STAT232 Final Project

Analysis of Austin, Texas Airbnb

Project Description

In this project, we will perform a descriptive and exploratory analysis of the Airbnb data of Austin Texas. In order to understand how the phenomena of each variable behave individually and transversely, in addition to to create prediction model for future price prediction, the whole analysis will follow a simple and direct structure, well detailed in all topics, at the same time, create an intuitive and simple guide to understand the data involved. Further, We did the comparison of number of Airbnb, to their respective prices, count and density according to different location in the city of Austin and used different EDA visualization tools to better visualize the relation between different variables. Similarly we are also using the help of Austin map to plot the area of significance according to different criteria for instance, price difference according to neibouhood, number of total private rooms, price per person, highest ratings etc.

Part I: Preparation

1. Install & Load All Libraries

```
# install.packages(c('maps', 'mapproj', 'leaflet',
# 'caTools', 'magick', 'ggrepel', 'ggmap', 'factoextra',
# 'randomForest', 'glmnet', 'webshot'))
```

```
library(tidyverse)
library(maps)
library(mapproj)
library(maptools)
library(ggmap)
library(mapview)
library(leaflet)
library(dplyr)
library(corrplot)
library(gridExtra)
library(ggrepel)
library(magick)
library(caTools)
library(randomForest)
library(glmnet)
library(factoextra)
library(webshot)
library(htmlwidgets)
webshot::install_phantomjs()
```

2. Load Data

```
googleid <- "1jDofhd3DH0qW8sHFfteEnF0MorS8-UjF"
googleurl <- sprintf("https://docs.google.com/uc?id=%s&export=download",
    googleid)
austin <- readr::read_csv(googleurl)
```

```
## Rows: 11269 Columns: 67
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (21): Name, Description, Neighborhood_overview, Host_name, Host_since, H...
## dbl (37): Id, Host_id, Host_listings_count, Host_total_listings_count, Neigh...
## lgl  (9): Host_is_superhost, Host_has_profile_pic, Host_identity_verified, N...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Part II: Data Cleaning

Make all column names lowercase (easier for typing!)

```
colnames(austin) <- tolower(colnames(austin))
```

1. Check columns

```
glimpse(austin)
```

```
## Rows: 11,269
## Columns: 67
## $ id                                          <dbl> 2265, 5245, 5456, 5769, 6~
## $ name                                        <chr> "Zen-East in the Heart of~
## $ description                                 <chr> "Zen East is situated in ~
## $ neighborhood_overview                       <chr> NA, NA, "My neighborhood ~
## $ host_id                                     <dbl> 2466, 2466, 8028, 8186, 1~
## $ host_name                                   <chr> "Paddy", "Paddy", "Sylvia~
## $ host_since                                  <chr> "8/23/2008", "8/23/2008",~
## $ host_location                               <chr> "Austin, Texas, United St~
## $ host_about                                  <chr> "I am a long time residen~
## $ host_response_time                          <chr> "within a day", "within a~
## $ host_response_rate                          <chr> "100%", "100%", "100%", "~
## $ host_acceptance_rate                        <chr> "71%", "71%", "93%", "100~
## $ host_is_superhost                           <lgl> TRUE, TRUE, TRUE, TRUE, T~
## $ host_neighbourhood                          <chr> "East Downtown", "East Do~
## $ host_listings_count                         <dbl> 3, 3, 1, 1, 1, 2, 1, 1, 1~
## $ host_total_listings_count                   <dbl> 3, 3, 1, 1, 1, 2, 1, 1, 1~
## $ host_verifications                          <chr> "['email', 'phone', 'face~
## $ host_has_profile_pic                        <lgl> TRUE, TRUE, TRUE, TRUE, T~
## $ host_identity_verified                      <lgl> TRUE, TRUE, TRUE, TRUE, T~
## $ neighbourhood                               <chr> NA, NA, "Austin, Texas, U~
## $ neighbourhood_cleansed                      <dbl> 78702, 78702, 78702, 7872~
## $ neighbourhood_group_cleansed                <lgl> NA, NA, NA, NA, NA, NA, N~
## $ latitude                                    <dbl> 30.27752, 30.27614, 30.26~
## $ longitude                                   <dbl> -97.71377, -97.71320, -97~
## $ property_type                               <chr> "Entire residential home"~
```

```
## $ room_type                                       <chr> "Entire home/apt", "Priva~
## $ accommodates                                    <dbl> 4, 2, 3, 2, 2, 3, 2, 3, 4~
## $ bathrooms                                       <lgl> NA, NA, NA, NA, NA, NA, N~
## $ bathrooms_text                                  <chr> "2 baths", "1 private bat~
## $ bedrooms                                        <dbl> 2, 1, 1, 1, NA, 1, 1, 2, ~
## $ beds                                            <dbl> 2, 2, 2, 1, 1, 2, 1, 2, 2~
## $ amenities                                       <chr> "[\"Paid parking off prem~
## $ price                                           <chr> "$179.00", "$114.00", "$1~
## $ minimum_nights                                  <dbl> 7, 30, 2, 1, 3, 3, 3, 30,~
## $ maximum_nights                                  <dbl> 180, 90, 90, 14, 365, 365~
## $ minimum_minimum_nights                          <dbl> 7, 30, 2, 1, 3, 3, 3, 30,~
## $ maximum_minimum_nights                          <dbl> 7, 30, 2, 1, 3, 3, 10, 30~
## $ minimum_maximum_nights                          <dbl> 180, 90, 90, 14, 1125, 11~
## $ maximum_maximum_nights                          <dbl> 180, 90, 90, 14, 1125, 11~
## $ minimum_nights_avg_ntm                          <dbl> 7.0, 30.0, 2.0, 1.0, 3.0,~
## $ maximum_nights_avg_ntm                          <dbl> 180, 90, 90, 14, 1125, 11~
## $ calendar_updated                                <lgl> NA, NA, NA, NA, NA, NA, N~
## $ has_availability                                <lgl> TRUE, TRUE, TRUE, TRUE, T~
## $ availability_30                                 <dbl> 6, 0, 8, 0, 0, 9, 0, 0, 3~
## $ availability_60                                 <dbl> 7, 0, 25, 0, 0, 18, 29, 2~
## $ availability_90                                 <dbl> 30, 0, 55, 0, 0, 36, 59, ~
## $ availability_365                                <dbl> 35, 0, 324, 0, 0, 201, 59~
## $ calendar_last_scraped                           <chr> "10/14/2021", "10/14/2021~
## $ number_of_reviews                               <dbl> 26, 9, 575, 264, 117, 254~
## $ number_of_reviews_ltm                           <dbl> 2, 0, 39, 7, 4, 22, 0, 0,~
## $ number_of_reviews_l30d                          <dbl> 0, 0, 3, 0, 0, 3, 0, 0, 1~
## $ first_review                                    <chr> "11/9/2015", "3/14/2018",~
## $ last_review                                     <chr> "7/2/2021", "2/24/2017", ~
## $ review_scores_rating                            <dbl> 4.68, 4.57, 4.83, 4.90, 4~
## $ review_scores_accuracy                          <dbl> 4.74, 4.80, 4.87, 4.90, 4~
## $ review_scores_cleanliness                       <dbl> 4.83, 4.20, 4.85, 4.86, 4~
## $ review_scores_checkin                           <dbl> 4.77, 5.00, 4.89, 4.91, 4~
## $ review_scores_communication                     <dbl> 4.83, 4.40, 4.80, 4.94, 4~
## $ review_scores_location                          <dbl> 4.26, 4.75, 4.73, 4.74, 4~
## $ review_scores_value                             <dbl> 4.35, 4.50, 4.78, 4.92, 4~
## $ license                                         <lgl> NA, NA, NA, NA, NA, NA, N~
## $ instant_bookable                                <lgl> FALSE, FALSE, FALSE, FALS~
## $ calculated_host_listings_count                  <dbl> 3, 3, 1, 1, 1, 1, 1, 2, 1~
## $ calculated_host_listings_count_entire_homes     <dbl> 2, 2, 1, 0, 1, 1, 1, 2, 1~
## $ calculated_host_listings_count_private_rooms    <dbl> 1, 1, 0, 1, 0, 0, 0, 0, 0~
## $ calculated_host_listings_count_shared_rooms     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ reviews_per_month                               <dbl> 0.36, 0.21, 24.16, 5.95, ~
```

There are NA, so first, let's select columns we are interested in and "useful", then remove empty columns and rows (so we don't lose all data). Then let's see what we can do with NA values. At last, we remove rows with NA values that we can't deal with.

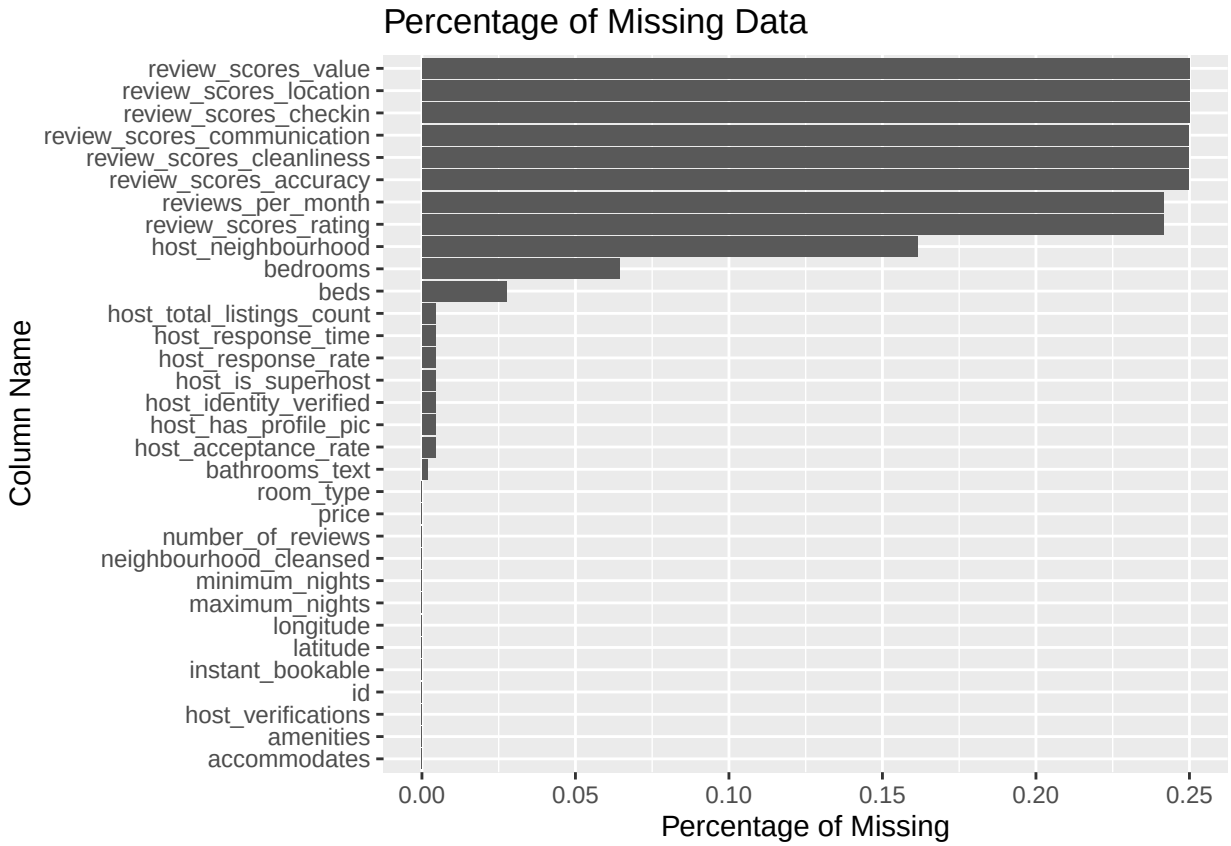
2. Select columns we are interested in and remove empty columns & rows.

```
austin <- austin %>%
    dplyr::select(id, host_response_time:host_neighbourhood,
        host_total_listings_count:host_identity_verified, neighbourhood_cleansed,
        latitude, longitude, room_type, accommodates, bathrooms_text:maximum_nights,
```

```
        number_of_reviews, review_scores_rating:review_scores_value,
        instant_bookable, reviews_per_month)

# remove empty columns
empty_columns <- colSums(is.na(austin) | austin == "") == nrow(austin)
austin <- austin[, !empty_columns]
empty_columns <- colSums(is.na(austin) | austin == "") == nrow(austin)

# remove empty rows
austin <- austin[rowSums(is.na(austin)) != ncol(austin), ]
```

3. Dive into rows with miss values, specifically numeric rows.

```
miss_data <- austin %>%
    summarise_all(~(sum(is.na(.))/n()))
miss_data <- gather(miss_data, key = "variables", value = "percentage_of_missing")

ggplot(miss_data, aes(x = reorder(variables, percentage_of_missing),
    y = percentage_of_missing)) + geom_bar(stat = "identity") +
    coord_flip() + ggtitle("Percentage of Missing Data") + xlab("Column Name") +
    ylab("Percentage of Missing")
```



Columns about reviews appear to have same scale of missing values. Let's remove those NA directly in next step.

Beds missing - fill with number of bedroom

Bedroom miss - fill with number of beds.

Others we just drop all NA.

```
# review_scores_rating is mean of all other rating, so just
# drop rows with NA in review_scores_rating
austin <- austin[!is.na(austin$review_scores_rating), ]

# fill values for bed & bedrooms missing
austin$beds <- ifelse(is.na(austin$beds), austin$bedrooms, austin$beds)
austin$bedrooms <- ifelse(is.na(austin$bedrooms), austin$beds,
    austin$bedrooms)
```

4. Remove rows with NA and duplicated rows

```
# remove rows with NA
austin <- drop_na(austin)

# remove duplicated rows
austin <- austin[!duplicated(austin), ]
```

5. Convert column to right type

```
# price to integer
austin$price <- gsub("\\$", "", austin$price)
austin$price <- gsub(",", "", austin$price)
austin$price <- as.numeric(austin$price)
```

```
# convert neighbourhood_cleansed(zipcode234) to character
austin$neighbourhood_cleansed <- as.factor(austin$neighbourhood_cleansed)
```

6. Work on categorical columns (that we are interested in!!)

```
# get the count of host_verification and amentities
host_cate <- gsub("\\[|\\]", "", austin$host_verifications)
host_cate_split <- str_split(host_cate, ",")

output <- NULL
for (i in 1:length(host_cate_split)) {
    output[i] <- length(host_cate_split[[i]])
}

amenities_cate <- gsub("\\[|\\]", "", austin$amenities)
amenities_cate <- gsub("\"", "", amenities_cate)
amenities_cate_split <- str_split(amenities_cate, ",")

output1 <- NULL
for (i in 1:length(amenities_cate_split)) {
    output1[i] <- length(amenities_cate_split[[i]])
}

count_host_verfitication <- as.data.frame(output)
colnames(count_host_verfitication) <- c("count_host_verfitication")
count_amenities_verfitication <- as.data.frame(output1)
```

```r
colnames(count_amenities_verfitication) <- c("count_amenities_verfitication")

# convert host_response_rate to numeric
host_response_rate1 <- as.numeric(gsub("[\\%,]", "", austin$host_response_rate))
host_response_rate1 <- as.data.frame(host_response_rate1/100)
colnames(host_response_rate1) <- c("host_response_rate1")

# convert host_acceptance_rate to numeric
host_acceptance_rate1 <- as.numeric(gsub("[\\%,]", "", austin$host_acceptance_rate))
host_acceptance_rate1 <- as.data.frame(host_acceptance_rate1/100)
colnames(host_acceptance_rate1) <- c("host_acceptance_rate1")

# get count of bathrooms from bathrooms_text (here we don't
# consider shared or private)
count_bathrooms <- as.data.frame(as.numeric(gsub("([0-9]+).*$",
    "\\1", austin$bathrooms_text)))
colnames(count_bathrooms) <- c("count_bathrooms")

# combine the new columns to austin dataset
austin <- cbind(austin, count_host_verfitication, count_amenities_verfitication,
    host_response_rate1, host_acceptance_rate1, count_bathrooms)
```

```r
# select useful columns for further analysis
austin <- dplyr::select(austin, -c("host_response_rate", "host_acceptance_rate",
    "host_verifications", "bathrooms_text", "amenities"))
head(austin)
```

```
##     id host_response_time host_is_superhost host_neighbourhood
## 1 2265       within a day              TRUE      East Downtown
## 2 5245       within a day              TRUE      East Downtown
## 3 5456 within a few hours              TRUE      East Downtown
## 4 5769                N/A              TRUE  SW Williamson Co.
## 5 6413     within an hour              TRUE     Travis Heights
## 6 6448     within an hour              TRUE             Zilker
##   host_total_listings_count host_has_profile_pic host_identity_verified
## 1                         3                 TRUE                    TRUE
## 2                         3                 TRUE                    TRUE
## 3                         1                 TRUE                    TRUE
## 4                         1                 TRUE                    TRUE
## 5                         1                 TRUE                    TRUE
## 6                         2                 TRUE                    TRUE
##   neighbourhood_cleansed latitude longitude       room_type accommodates
## 1                  78702 30.27752 -97.71377 Entire home/apt            4
## 2                  78702 30.27614 -97.71320    Private room            2
## 3                  78702 30.26057 -97.73441 Entire home/apt            3
## 4                  78729 30.45697 -97.78422    Private room            2
## 5                  78704 30.24885 -97.73587 Entire home/apt            2
## 6                  78704 30.26034 -97.76487 Entire home/apt            3
##   bedrooms beds price minimum_nights maximum_nights number_of_reviews
## 1        2    2   179              7            180                26
## 2        1    2   114             30             90                 9
## 3        1    2   108              2             90               575
## 4        1    1    39              1             14               264
## 5        1    1   109              3            365               117
## 6        1    2   146              3            365               254
```

```
##   review_scores_rating review_scores_accuracy review_scores_cleanliness
## 1                 4.68                   4.74                     4.83
## 2                 4.57                   4.80                     4.20
## 3                 4.83                   4.87                     4.85
## 4                 4.90                   4.90                     4.86
## 5                 4.97                   4.99                     4.99
## 6                 4.98                   4.96                     4.96
##   review_scores_checkin review_scores_communication review_scores_location
## 1                  4.77                        4.83                    4.26
## 2                  5.00                        4.40                    4.75
## 3                  4.89                        4.80                    4.73
## 4                  4.91                        4.94                    4.74
## 5                  4.99                        4.98                    4.86
## 6                  4.99                        4.97                    4.96
##   review_scores_value instant_bookable reviews_per_month
## 1                4.35           FALSE              0.36
## 2                4.50           FALSE              0.21
## 3                4.78           FALSE             24.16
## 4                4.92           FALSE              5.95
## 5                4.94            TRUE              1.27
## 6                4.89            TRUE              2.06
##   count_host_verfitication count_amenities_verfitication host_response_rate1
## 1                        5                            50                   1
## 2                        5                            23                   1
## 3                        4                            26                   1
## 4                        6                            18                  NA
## 5                        8                            28                   1
## 6                        5                            60                   1
##   host_acceptance_rate1 count_bathrooms
## 1                  0.71               2
## 2                  0.71               1
## 3                  0.93               1
## 4                  1.00               1
## 5                  1.00               1
## 6                  0.98               1
```

There are NA in host_response_time, but it fine, we will take it as a category for further analysis.

Now we have cleaned, full dataset, we can move to further analysis.

Part III: Exploratory Data Analysis & Visualization

1. Summary of data

```
summary(austin)
##        id           host_response_time host_is_superhost host_neighbourhood
##  Min.   :    2265   Length:7049        Mode :logical     Length:7049
##  1st Qu.:15942648   Class :character   FALSE:4184        Class :character
##  Median :31876373   Mode  :character   TRUE :2865        Mode  :character
##  Mean   :29934451
##  3rd Qu.:46420034
##  Max.   :52691975
##
##  host_total_listings_count host_has_profile_pic host_identity_verified
```

```
##  Min.   :   0.00          Mode :logical        Mode :logical
##  1st Qu.:   1.00          FALSE:26             FALSE:1062
##  Median :   1.00          TRUE :7023           TRUE :5987
##  Mean   :  29.82
##  3rd Qu.:   5.00
##  Max.   :1987.00
##
##  neighbourhood_cleansed    latitude       longitude       room_type
##  78704  :1361           Min.   :30.08   Min.   :-98.01   Length:7049
##  78702  :1122           1st Qu.:30.25   1st Qu.:-97.76   Class :character
##  78741  : 512           Median :30.27   Median :-97.74   Mode  :character
##  78701  : 445           Mean   :30.28   Mean   :-97.75
##  78703  : 372           3rd Qu.:30.30   3rd Qu.:-97.72
##  78751  : 359           Max.   :30.52   Max.   :-97.57
##  (Other):2878
##   accommodates      bedrooms           beds            price
##  Min.   : 1.00   Min.   : 0.000   Min.   : 0.000   Min.   :    5.0
##  1st Qu.: 2.00   1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.:   96.0
##  Median : 4.00   Median : 2.000   Median : 2.000   Median :  155.0
##  Mean   : 4.72   Mean   : 1.936   Mean   : 2.556   Mean   :  273.7
##  3rd Qu.: 6.00   3rd Qu.: 2.000   3rd Qu.: 3.000   3rd Qu.:  279.0
##  Max.   :16.00   Max.   :23.000   Max.   :61.000   Max.   :10000.0
##
##  minimum_nights     maximum_nights   number_of_reviews review_scores_rating
##  Min.   :   1.000   Min.   :   1.0   Min.   :  1.00    Min.   :0.000
##  1st Qu.:   1.000   1st Qu.:  30.0   1st Qu.:  3.00    1st Qu.:4.750
##  Median :   2.000   Median : 365.0   Median : 12.00    Median :4.930
##  Mean   :   6.578   Mean   : 590.5   Mean   : 43.34    Mean   :4.801
##  3rd Qu.:   3.000   3rd Qu.:1125.0   3rd Qu.: 46.00    3rd Qu.:5.000
##  Max.   :1100.000   Max.   :1825.0   Max.   :953.00    Max.   :5.000
##
##  review_scores_accuracy review_scores_cleanliness review_scores_checkin
##  Min.   :1.00           Min.   :1.000             Min.   :1.000
##  1st Qu.:4.83           1st Qu.:4.720             1st Qu.:4.910
##  Median :4.96           Median :4.910             Median :5.000
##  Mean   :4.84           Mean   :4.769             Mean   :4.896
##  3rd Qu.:5.00           3rd Qu.:5.000             3rd Qu.:5.000
##  Max.   :5.00           Max.   :5.000             Max.   :5.000
##
##  review_scores_communication review_scores_location review_scores_value
##  Min.   :0.000               Min.   :0.000          Min.   :0.000
##  1st Qu.:4.910               1st Qu.:4.810          1st Qu.:4.670
##  Median :5.000               Median :4.940          Median :4.860
##  Mean   :4.889               Mean   :4.839          Mean   :4.748
##  3rd Qu.:5.000               3rd Qu.:5.000          3rd Qu.:5.000
##  Max.   :5.000               Max.   :5.000          Max.   :5.000
##
##  instant_bookable reviews_per_month count_host_verfitication
##  Mode :logical    Min.   :  0.01    Min.   : 1.000
##  FALSE:3879       1st Qu.:  0.25    1st Qu.: 4.000
##  TRUE :3170       Median :  1.00    Median : 5.000
##                   Mean   :  2.77    Mean   : 5.486
##                   3rd Qu.:  2.78    3rd Qu.: 7.000
```

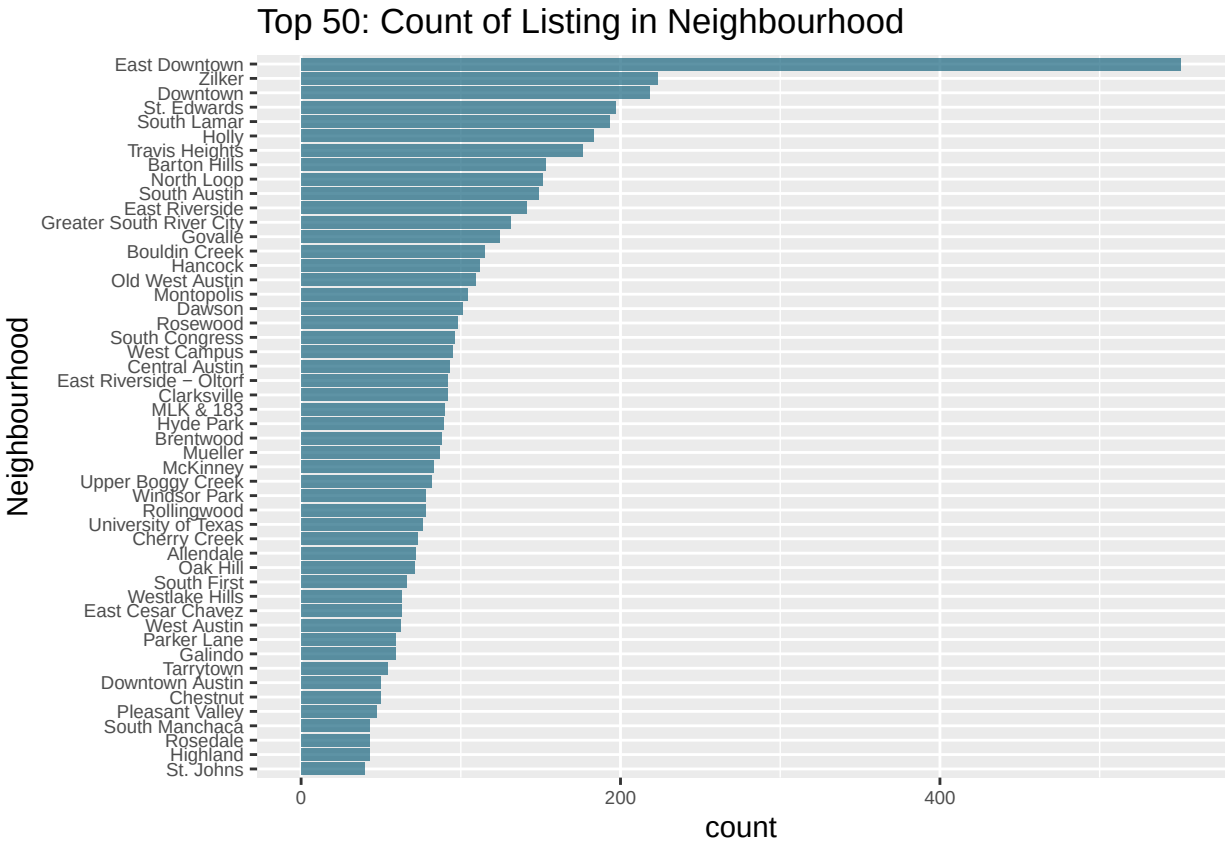
```
##                   Max.   :209.14    Max.   :12.000
##
##  count_amenities_verfitication host_response_rate1 host_acceptance_rate1
##  Min.   : 1.00                 Min.   :0.0000      Min.   :0.0000
##  1st Qu.:21.00                 1st Qu.:1.0000      1st Qu.:0.8600
##  Median :31.00                 Median :1.0000      Median :0.9800
##  Mean   :31.41                 Mean   :0.9693      Mean   :0.8748
##  3rd Qu.:39.00                 3rd Qu.:1.0000      3rd Qu.:1.0000
##  Max.   :89.00                 Max.   :1.0000      Max.   :1.0000
##                                NA's   :2151        NA's   :1852
##  count_bathrooms
##  Min.   : 0.000
##  1st Qu.: 1.000
##  Median : 1.000
##  Mean   : 1.487
##  3rd Qu.: 2.000
##  Max.   :17.000
##  NA's   :5
```

2. Descrptive Analysis

(1). Count of listing in each neighborhood

```
neighbourhood <- austin %>%
    group_by(host_neighbourhood) %>%
    summarise(count = n(), long = mean(longitude, na.rm = TRUE),
        lat = mean(latitude, na.rm = TRUE))
neighbourhood <- arrange(neighbourhood, desc(count))
head(neighbourhood)
## # A tibble: 6 x 4
##   host_neighbourhood count  long   lat
##   <chr>              <int> <dbl> <dbl>
## 1 East Downtown        551 -97.7  30.3
## 2 Zilker               223 -97.8  30.3
## 3 Downtown             218 -97.7  30.3
## 4 St. Edwards          197 -97.7  30.3
## 5 South Lamar          193 -97.8  30.2
## 6 Holly                183 -97.7  30.3
```

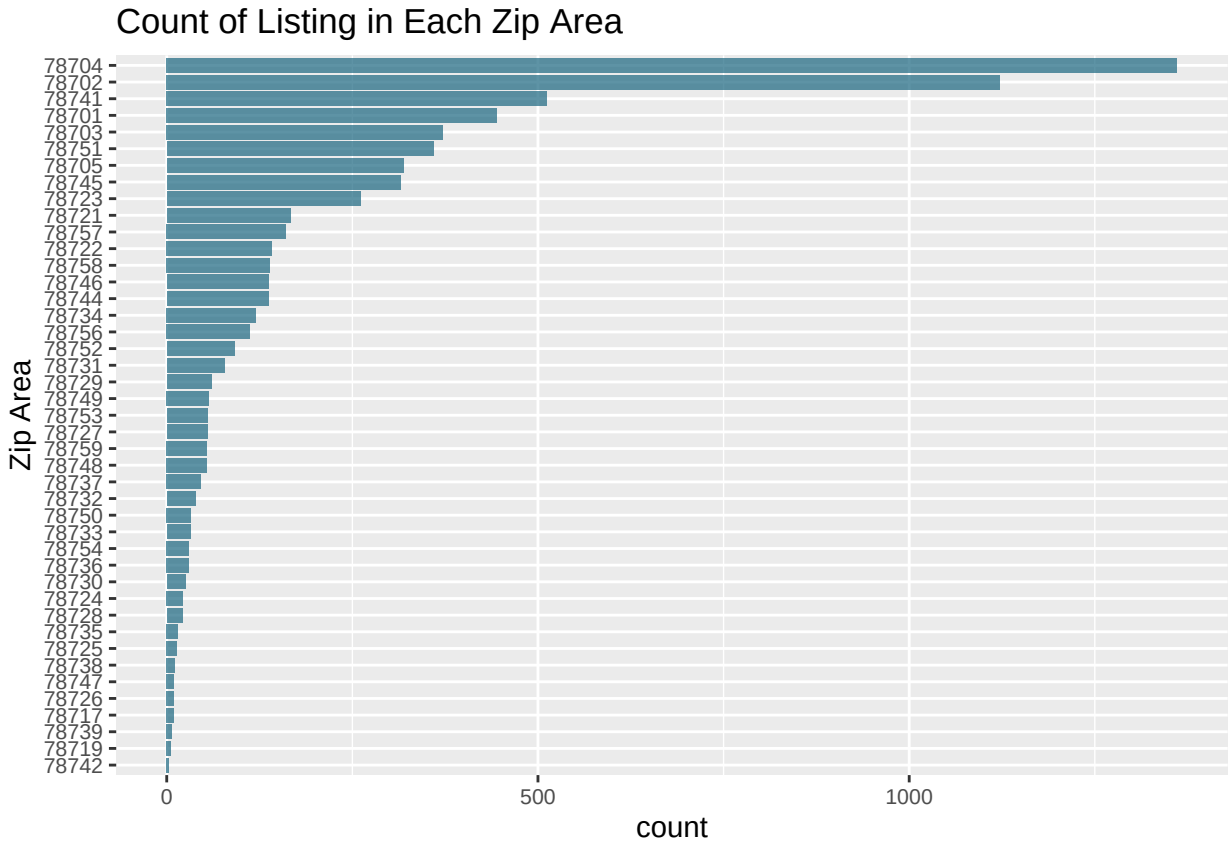
```
ggplot(head(neighbourhood, 50), mapping = aes(y = reorder(host_neighbourhood,
    count), x = count)) + geom_bar(stat = "identity", fill = rgb(0.1,
    0.4, 0.5, 0.7)) + ylab("Neighbourhood") + theme(axis.text = element_text(size = 7)) +
    ggtitle("Top 50: Count of Listing in Neighbourhood")
```



The abvoe bar plot has the listing count of top 50 neighborhood having airbnbs in the city of Austin. From the above barplot it can be figured out that East downtown has the highest number of listing with the count of 551 folllowed by Ziker, downtown, St Edwards and South Lamar which has the listing count of 223, 218 and 197 respectively. There are other neighborhoods whose count and rankings can be obseved from the above barplot easily.

(2). Count of listing in each zip area

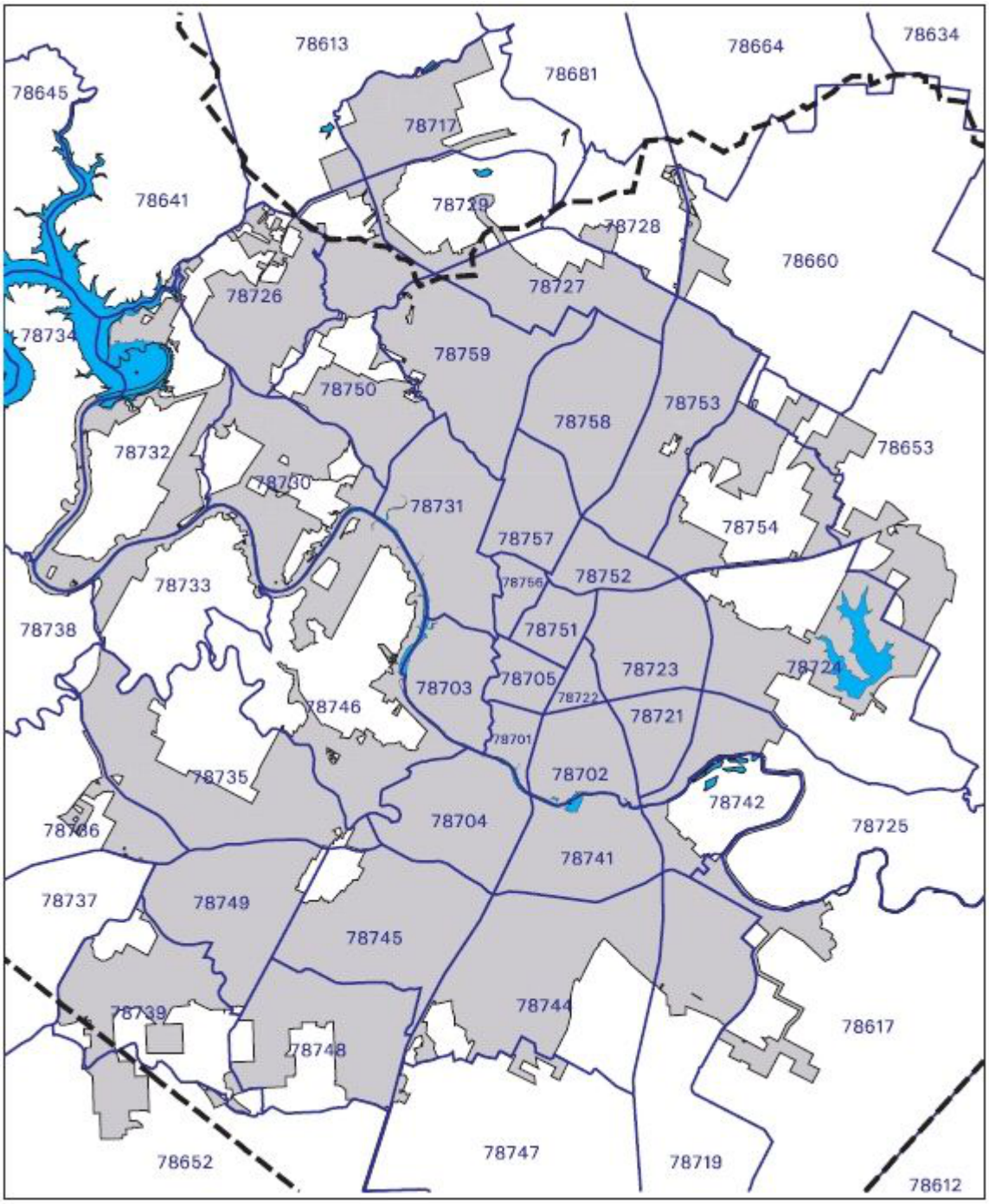
```
zip_area <- austin %>%
    group_by(neighbourhood_cleansed) %>%
    summarise(count = n())

ggplot(zip_area, mapping = aes(y = reorder(neighbourhood_cleansed,
    count), x = count)) + geom_bar(stat = "identity", fill = rgb(0.1,
    0.4, 0.5, 0.7)) + ylab("Zip Area") + theme(axis.text = element_text(size = 8)) +
    ggtitle("Count of Listing in Each Zip Area")
```



From the above barplot it can be figured out that Zip area having code of 78704 has the highest number of listing with the count of 1361 folllowed by Zip area 78702, 78741, 78701 and 78703 respectievly. There are other neighborhoods whose count and rankings can be observed from the above barplot easily. It can be infered that the zip code 78704 is a downtown where there is highest amount density of airbnbs

```
# get zip map
image_url <- "https://habitathunters.com/wp-content/uploads/2020/02/austin-zip-code-map.jpg"
pic <- image_read(image_url)
print(pic)
```

```
## # A tibble: 1 x 7
##   format width height colorspace matte filesize density
##   <chr>  <int>  <int> <chr>      <lgl>    <int> <chr>
## 1 JPEG     572    695 sRGB       FALSE    90811 96x96
```

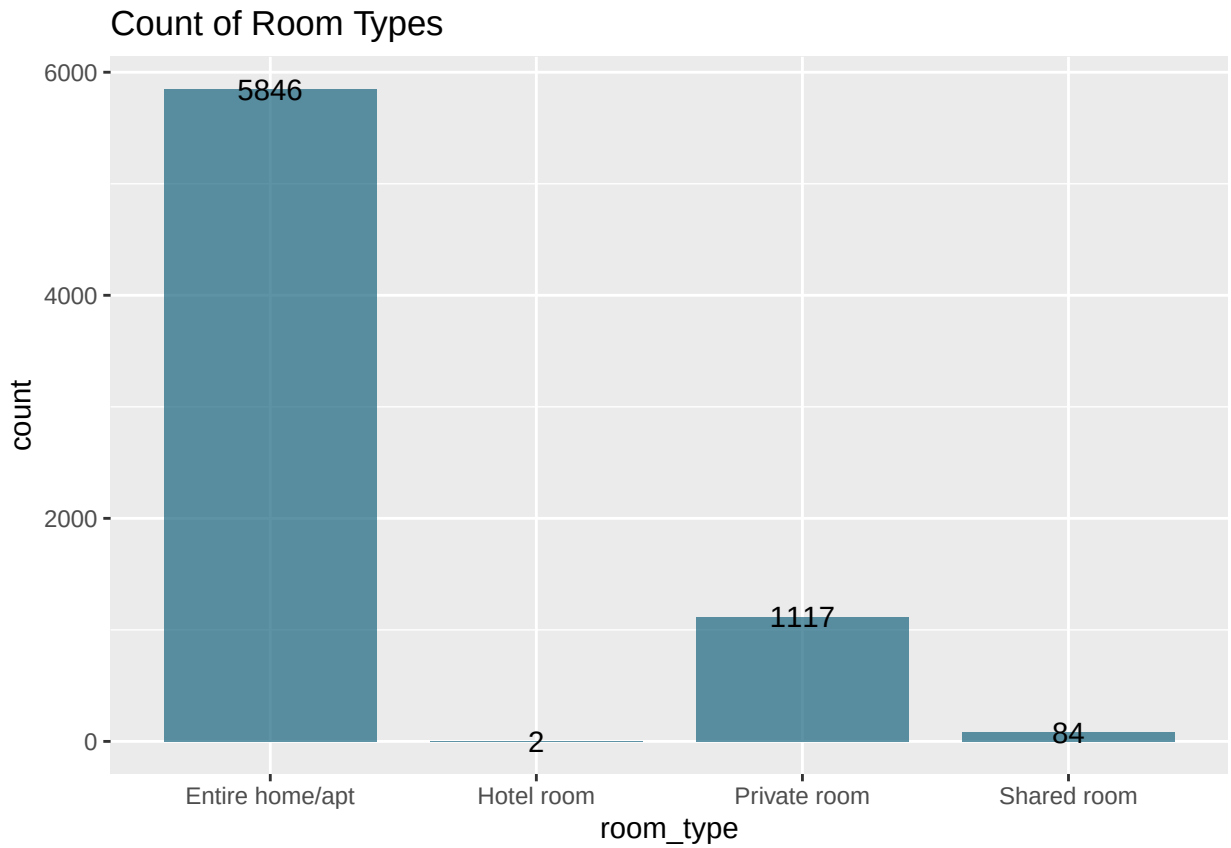


The result follows what we obtained from neighbourhood. From imported zip map of Austin, we can tell that.

(3). Count of Room Type

```
roommtype <- austin %>%
    group_by(room_type) %>%
    summarise(count = n())

ggplot(roommtype, aes(x = room_type, y = count)) + geom_bar(stat = "identity",
```

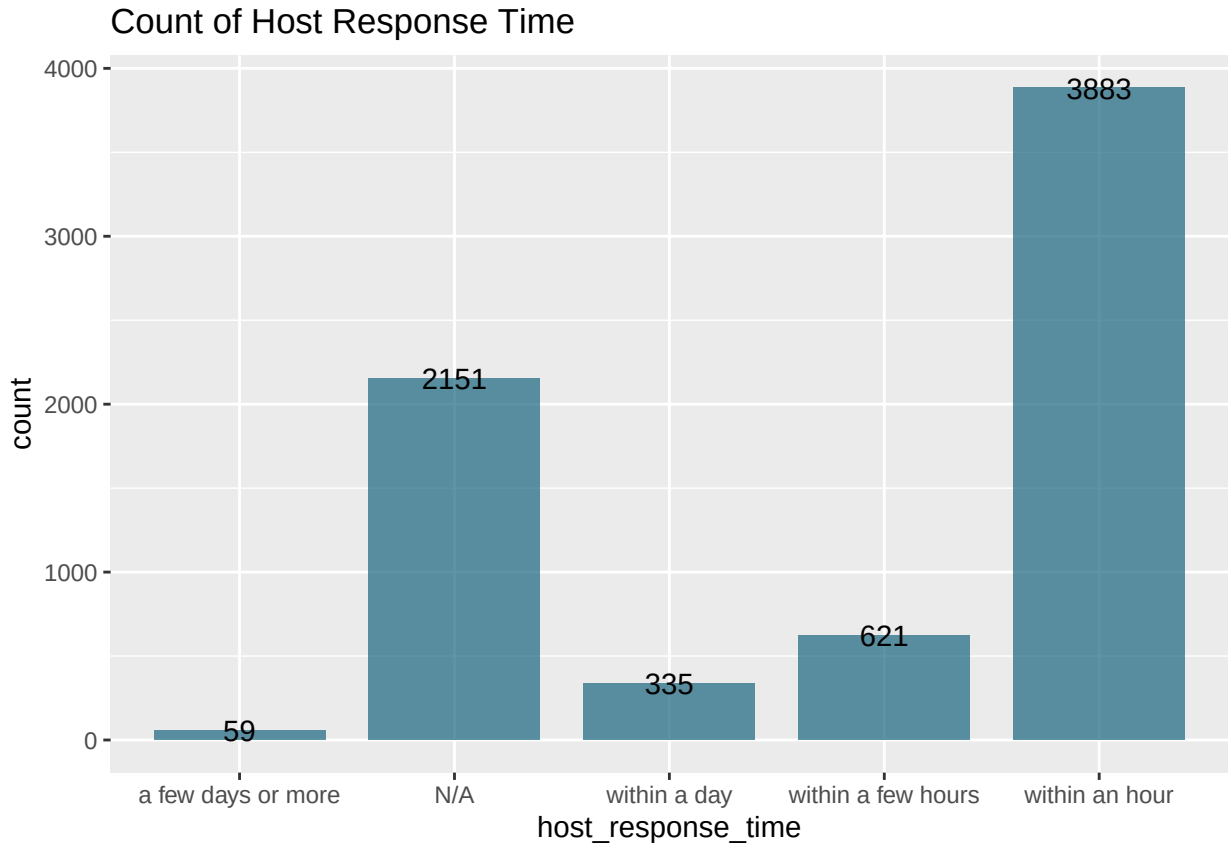
```
    fill = rgb(0.1, 0.4, 0.5, 0.7), width = 0.8) + geom_text(mapping = aes(label = count)) +
    ggtitle("Count of Room Types")
```



From the above barplot it can be seen that Entire home/apt room type has the highest numbers of listing which is 5846. second comes private room which has the count of 1117 followed by shared room with 84 and hotel room with 2 listing counts. The popularity of Entire home can be refered to the fact that many people who come to visit austin in groups tend to go for Entire home/apt in comparison to any other types of room type.

(4). Host Response Time

```
res_time <- austin %>%
    group_by(host_response_time) %>%
    summarise(count = n())

ggplot(res_time, aes(x = host_response_time, y = count)) + geom_bar(stat = "identity",
    fill = rgb(0.1, 0.4, 0.5, 0.7), width = 0.8) + geom_text(mapping = aes(label = count)) +
    ggtitle("Count of Host Response Time")
```

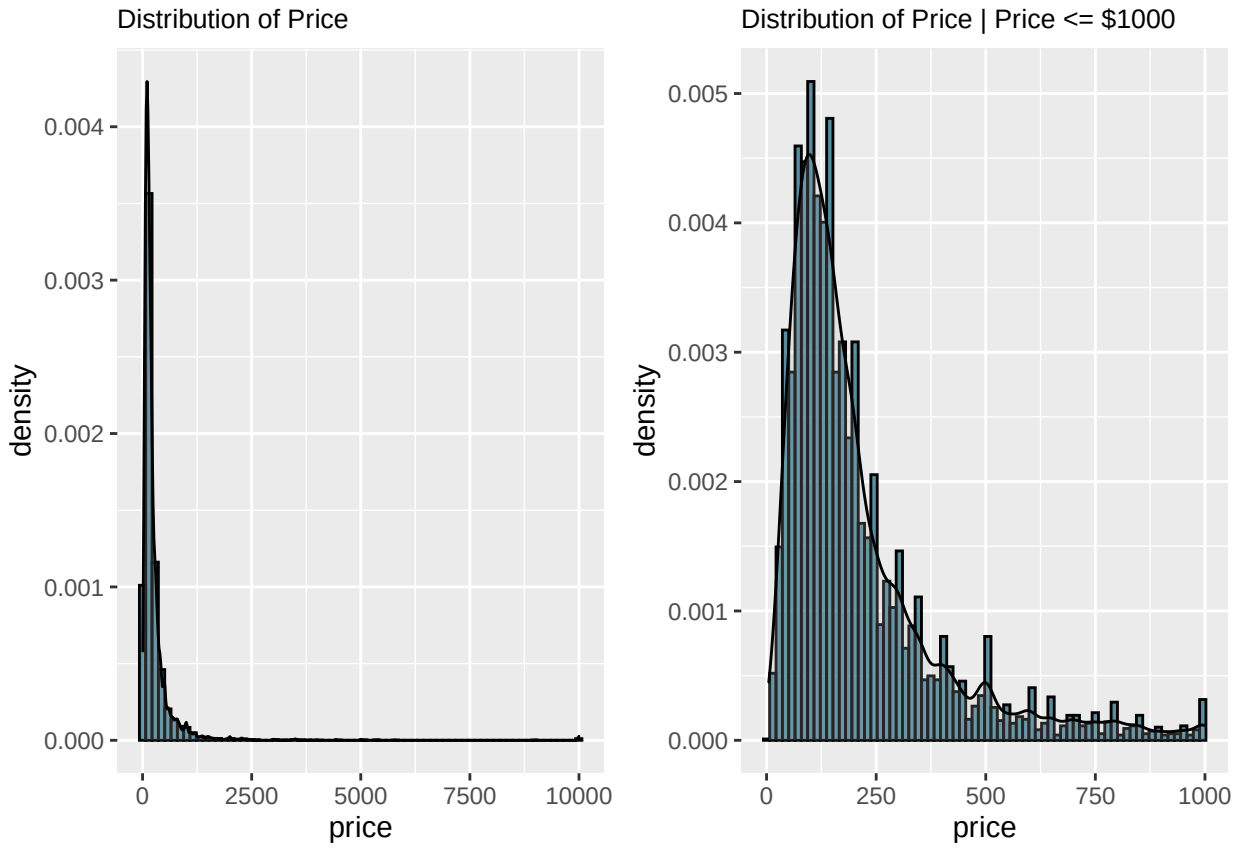


It can be observed that most host respond customers within the hour, which is 3883 in numbers. almost 2151 host never responded. 621 responded within a few hours, 335 within a day and 59 a few days and more.

Most hosts response customers within an hour.

(5). Price Distribution

```
p1 <- ggplot(data = austin, mapping = aes(x = price)) + geom_histogram(mapping = aes(y = ..density..),
    colour = "black", fill = rgb(0.1, 0.4, 0.5, 0.7), bins = 70) +
    geom_density(alpha = 0.2, fill = "grey") + ggtitle("Distribution of Price") +
    theme(plot.title = element_text(size = 10))

p2 <- ggplot(data = austin[austin$price <= 1000, ], mapping = aes(x = price)) +
    geom_histogram(mapping = aes(y = ..density..), colour = "black",
        fill = rgb(0.1, 0.4, 0.5, 0.7), bins = 70) + geom_density(alpha = 0.2,
    fill = "grey") + ggtitle("Distribution of Price | Price <= $1000") +
    theme(plot.title = element_text(size = 10))

grid.arrange(p1, p2, nrow = 1)
```



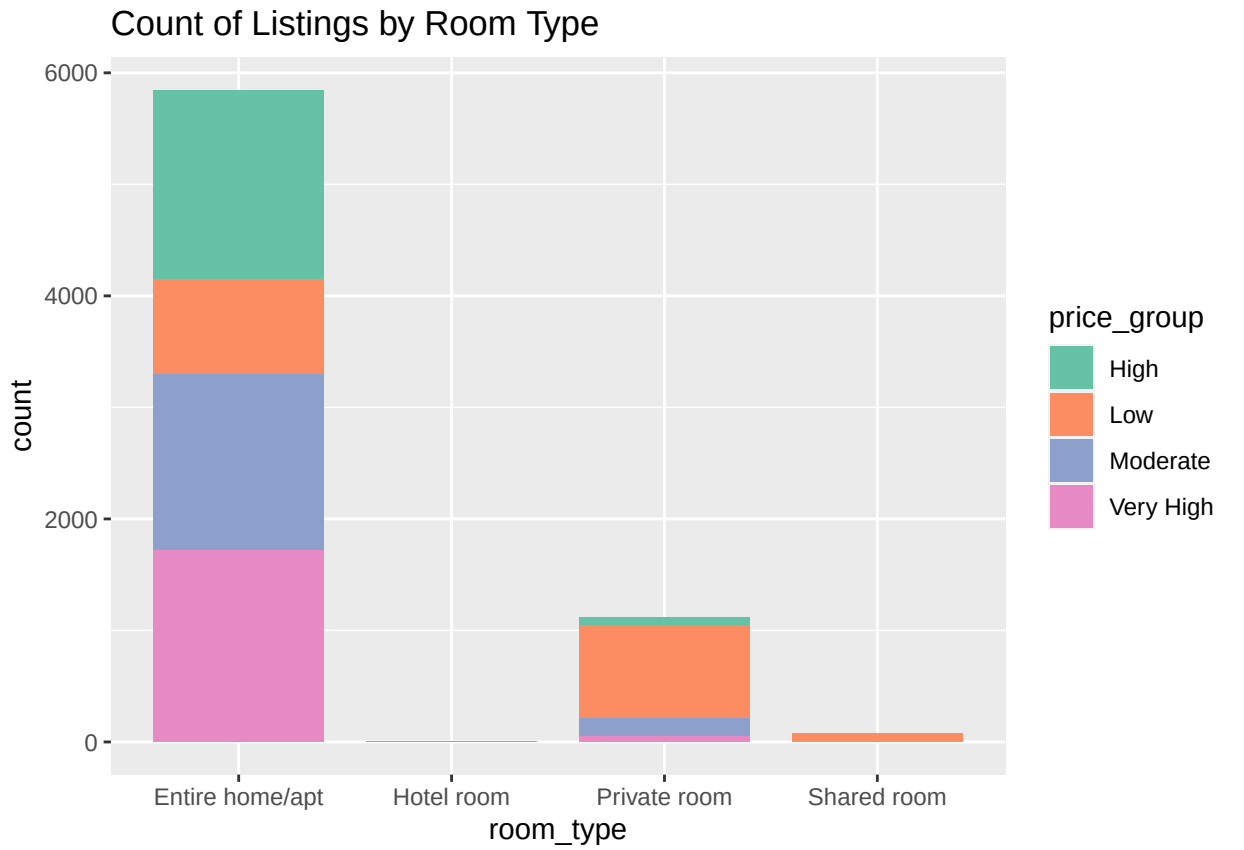
The distribution of price is heavily skewed to the right, even taking closer look to smaller price range(less than $1000). The highest concentration of price falls between o and $250. For a better price prediction later, we may apply data transformation.

3. Exploratory Analysis

(1). Price Distribution for Each room Type

```
price_group <- austin %>%
    mutate(price_group = ifelse(price < quantile(austin$price)[1],
        "Very Low", ifelse(price < quantile(austin$price)[2],
            "Low", ifelse(price < quantile(austin$price)[3],
                "Moderate", ifelse(price < quantile(austin$price)[4],
                  "High", "Very High")))))

ggplot(price_group, aes(room_type)) + geom_bar(aes(fill = price_group),
    width = 0.8) + ggtitle("Count of Listings by Room Type") +
    scale_fill_brewer(palette = "Set2")
```

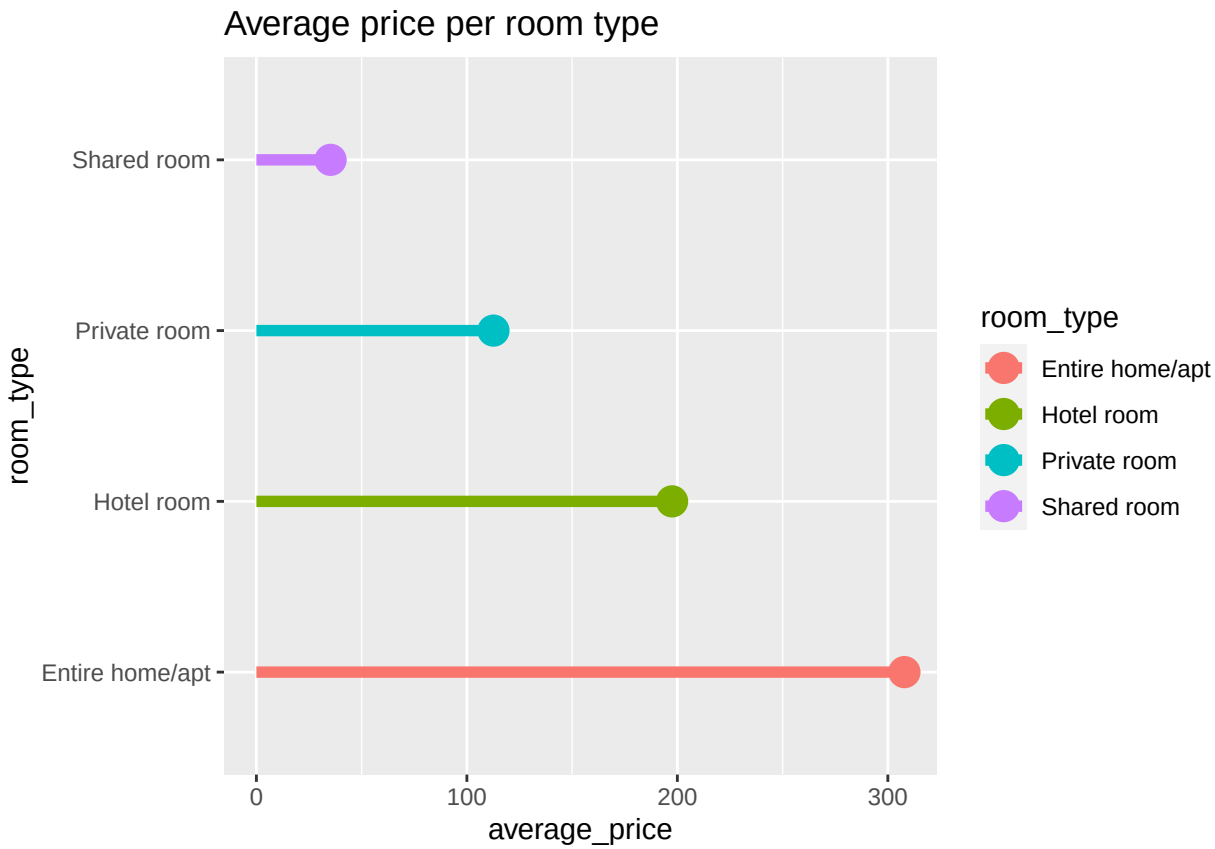
Count of Listing for Each room Type



From the above figure it can be observed that the price for entire home/apt varies widely. If we are considering price, we suggest to browse private room or shared room. Renting an entire home have many options for different price ranges

```
mean_room_type <- austin %>%
    group_by(room_type) %>%
    summarise(average_price = mean(price, na.rm = TRUE))
mean_room_type$Percent <- prop.table(mean_room_type$average_price) *
    100
mean_room_type
## # A tibble: 4 x 3
##   room_type       average_price Percent
##   <chr>                   <dbl>   <dbl>
## 1 Entire home/apt         308.    47.1
## 2 Hotel room              198.    30.2
## 3 Private room            113.    17.2
## 4 Shared room              35.2    5.40
ggplot(data = mean_room_type, aes(x = room_type, y = average_price)) +
    coord_flip() + geom_segment(aes(xend = room_type, yend = 0,
    color = room_type), size = 2) + geom_point(size = 5, mapping = aes(color = room_type)) +
    ggtitle("Average price per room type")
```

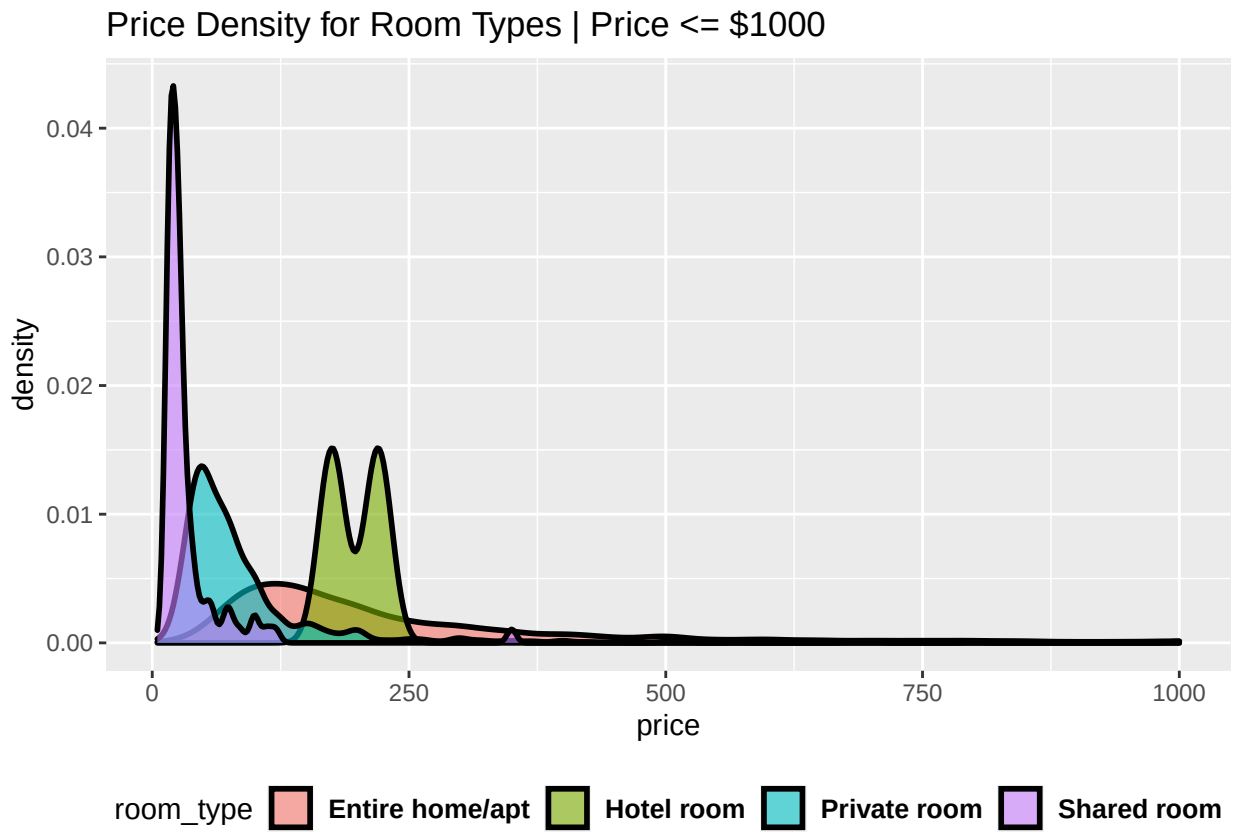
Average Price Per Room Type



In the graph above Geom point and geom segment has been used to show the the average price of different room types. Here it can be observed that average price of Entire home apartment is approximately $308 which is the highest followed by hotel room($197), private room($113) and shared room($35) respectively. The types of room is also represented by different colours.

```
ggplot(data = austin[austin$price <= 1000, ], mapping = aes(x = price,
    fill = room_type)) + geom_density(mapping = aes(fill = room_type),
    size = 1, color = "black", alpha = 0.6, size = 0.5) + ggtitle("Price Density for Room Types | Price
    theme(legend.position = "bottom", legend.text = element_text(colour = "black",
        size = 10, face = "bold"))
```

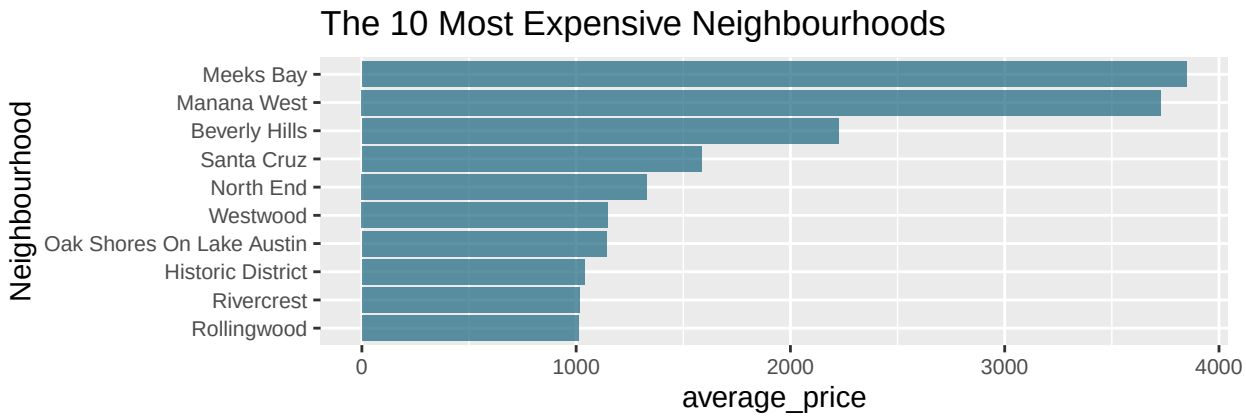
Price Distribution | Price <= $1000

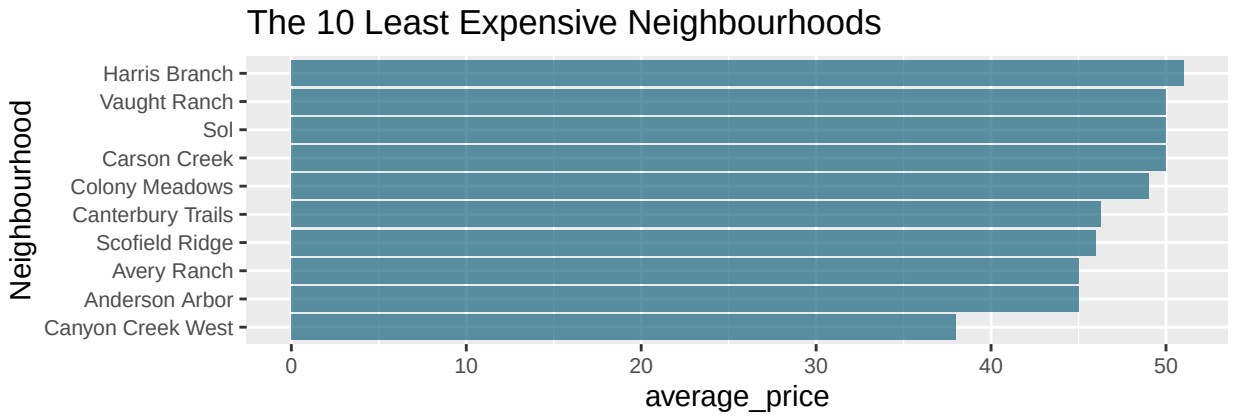


In the above density curve plot it can be observed that for the most of the shared room prices are cheap and and their prices are similar similar with the case with private room but they have a little more price options. Likewise, We can have a lots of price options for Entire home/apt till $500 . Similarly, most of the hotel rooms have the price range of $125 to $250.

(2). Avgerage Price for Most Expensive & Cheap 10 Neighbourhood

```
neighbourhood_price <- austin %>%
    group_by(host_neighbourhood) %>%
    summarise(average_price = mean(price, na.rm = TRUE))
neighbourhood_price <- arrange(neighbourhood_price, desc(average_price))

top10 <- head(neighbourhood_price, 10)
tail10 <- tail(neighbourhood_price, 10)

p1 <- ggplot(top10, aes(y = reorder(host_neighbourhood, average_price),
    x = average_price)) + geom_bar(stat = "identity", fill = rgb(0.1,
    0.4, 0.5, 0.7)) + ylab("Neighbourhood") + theme(axis.text = element_text(size = 8)) +
    ggtitle("The 10 Most Expensive Neighbourhoods")

p2 <- ggplot(tail10, aes(y = reorder(host_neighbourhood, average_price),
    x = average_price)) + geom_bar(stat = "identity", fill = rgb(0.1,
    0.4, 0.5, 0.7)) + ylab("Neighbourhood") + theme(axis.text = element_text(size = 8)) +
    ggtitle("The 10 Least Expensive Neighbourhoods")
grid.arrange(p1, p2, nrow = 2)
```





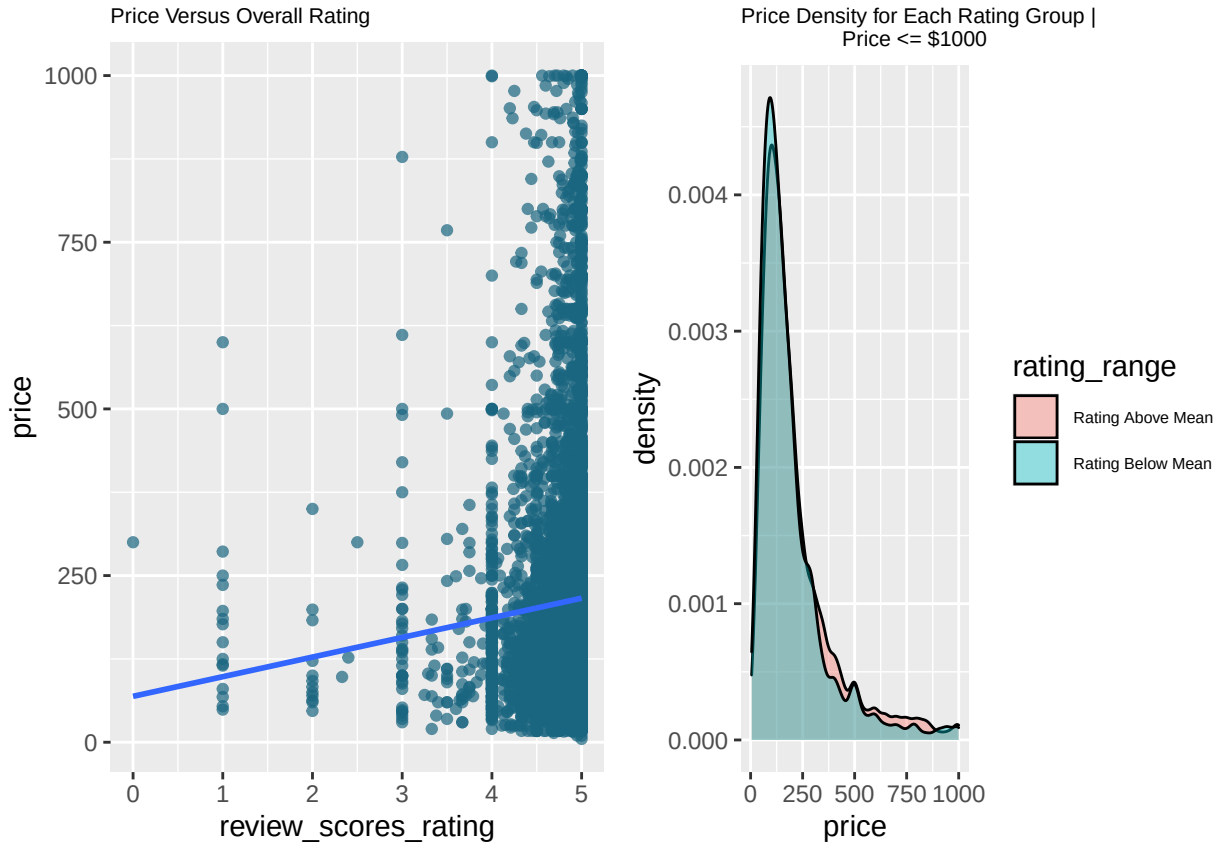
In the first plot we can observe the top 10 most expensive neighbourhoods in Austin. Here we can observe that Meeks bay has the highest average Airbnb price, which is $3850, Manana west which ranks 2nd gives close competition to Meeks bay which has the price of $3729. Beverly hills, Santa Cruz, North End and others join the list respectievely.

In the second plot we can observe the top 10 lest expensive neighbourhoods in Austin. Here we can observe that Canyon creek West has the lowest average Airbnb price, which is $38,all other lowest ranking neighbourhood has average prices which are close to each other and some of them like Vaught ranch, Sol and Carson Creek alongside Avery Ranch and Anderson Arbor has the exact same avergae price.

(3). Price Overall Rating | Price <= $1000

```
# scatter plot
p1 <- ggplot(austin[austin$price <= 1000, ], aes(x = review_scores_rating,
    y = price)) + geom_point(color = rgb(0.1, 0.4, 0.5, 0.7)) +
    geom_smooth(method = "lm", formula = y ~ x, se = FALSE) +
    ggtitle("Price Versus Overall Rating") + theme(plot.title = element_text(size = 8))

rating_range <- austin %>%
    mutate(rating_range = ifelse(review_scores_rating <= mean(austin$review_scores_rating,
        na.rm = TRUE), "Rating Below Mean", "Rating Above Mean"))
p2 <- ggplot(data = rating_range[rating_range$price <= 1000,
    ], mapping = aes(x = price, fill = rating_range)) + geom_density(mapping = aes(fill = rating_range)
    color = "black", alpha = 0.4, size = 0.5) + ggtitle("Price Density for Each Rating Group |
                Price <= $1000") +
    theme(legend.position = "right", legend.text = element_text(colour = "black",
        size = 6), plot.title = element_text(size = 8))
```

```
grid.arrange(p1, p2, nrow = 1)
```

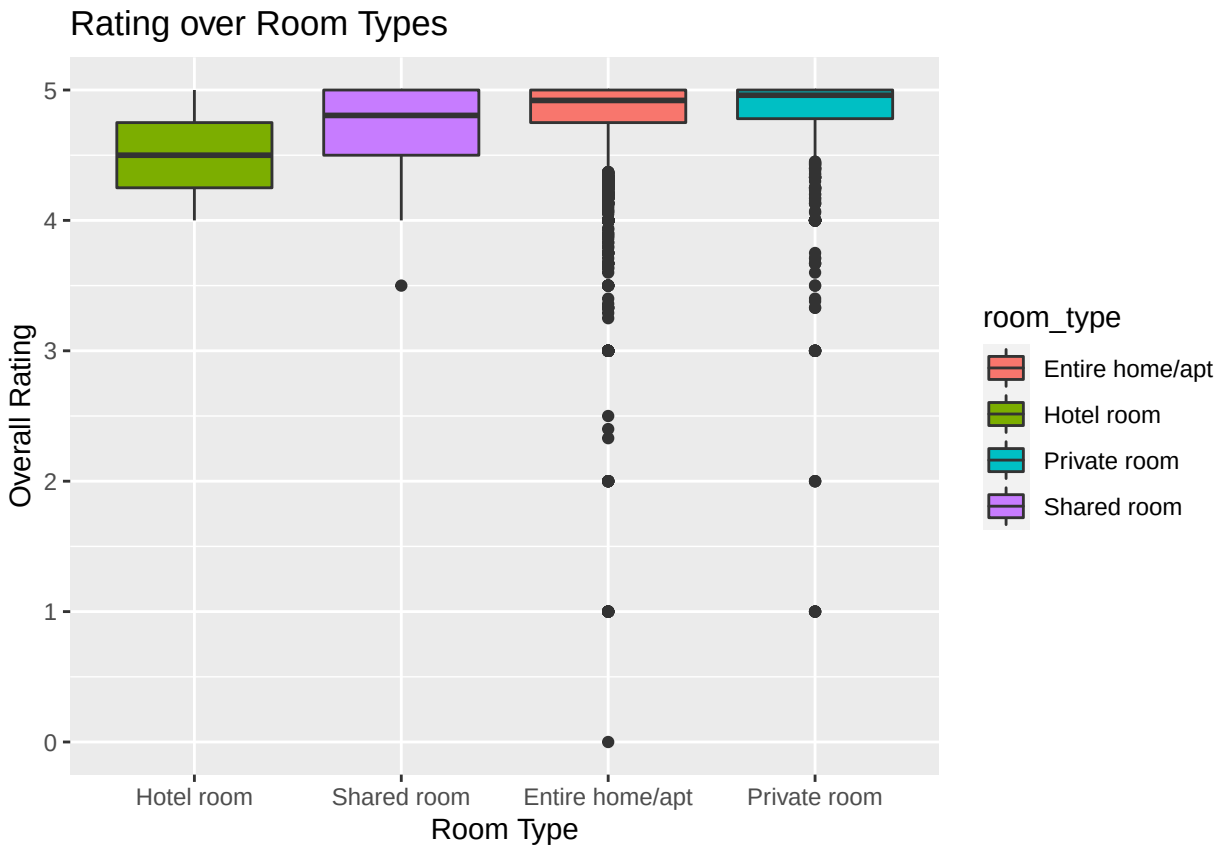


In the first plot it can be observed that despite the price range the rating and reviews of most of the airbnb in austin is on the highest side i.e most of the ratings are greater than 4. However if we look closely we can observe that higher concentration of the airbnb with high scores falls on the lower price range because the count(density) of airbnbs are highest on the lower price range which can be observed from the 2nd price density plot. It can also be observed that there are many low scoring airbnbs but most of them fall under the price range of $500 which can be verified by the second plot.

In the second plot it can be observed that

(4). Rating over Room Types

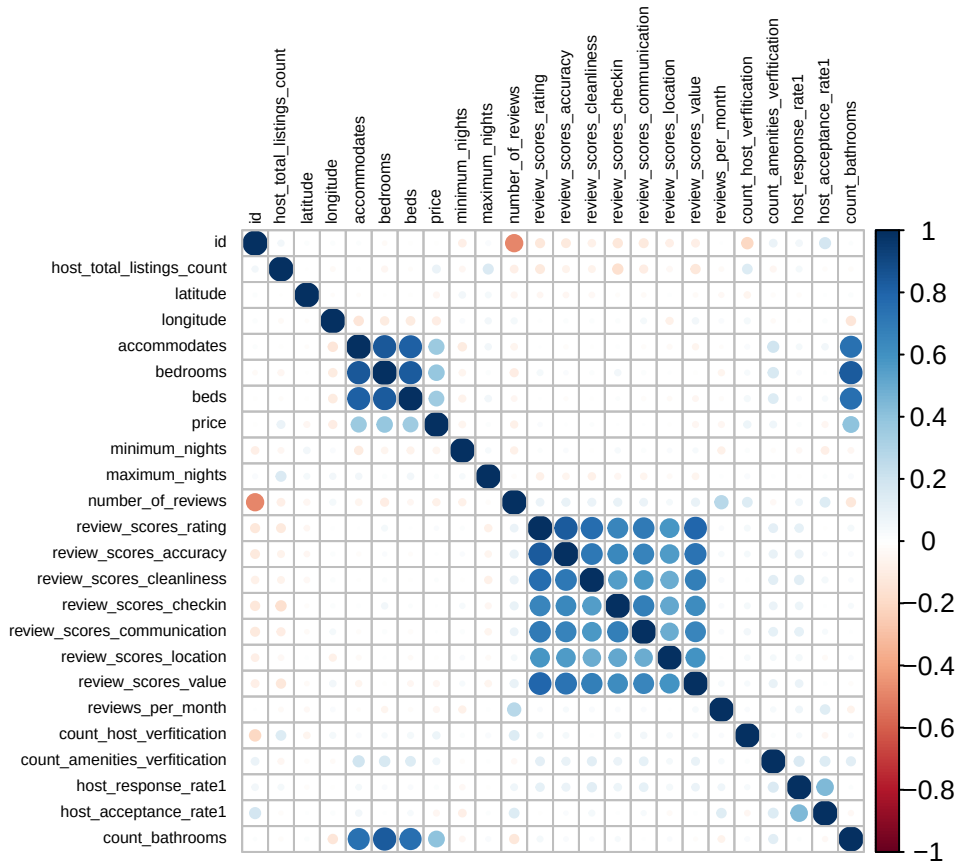
```
ggplot(austin, aes(x = reorder(room_type, review_scores_rating,
    FUN = median), y = review_scores_rating, fill = room_type)) +
    geom_boxplot() + ggtitle("Rating over Room Types") + xlab("Room Type") +
    ylab("Overall Rating")
```



The above box plot shows the overall rating of different room types of Airbnb. It can be observed that private room has overall average rating however, the shared room has the largest number of high ratings followed by hotel rooms. Private room has the highest overall rating followed by entire home/apt, shared room and hotel room respectively. However lots of outliers can be seen in the case of private and Entire home/apt, which is the sign that there is inconsistency in the quality in some of the private room and entire home/apt type.

(5). Price Factors Investigation

```
# select factors that may be influential
price_fac <- austin %>%
    dplyr::select_if(is.numeric)
price_fac <- na.omit(price_fac)
price_cor <- cor(price_fac)
corrplot(price_cor, tl.col = "black", tl.cex = 0.5)
```



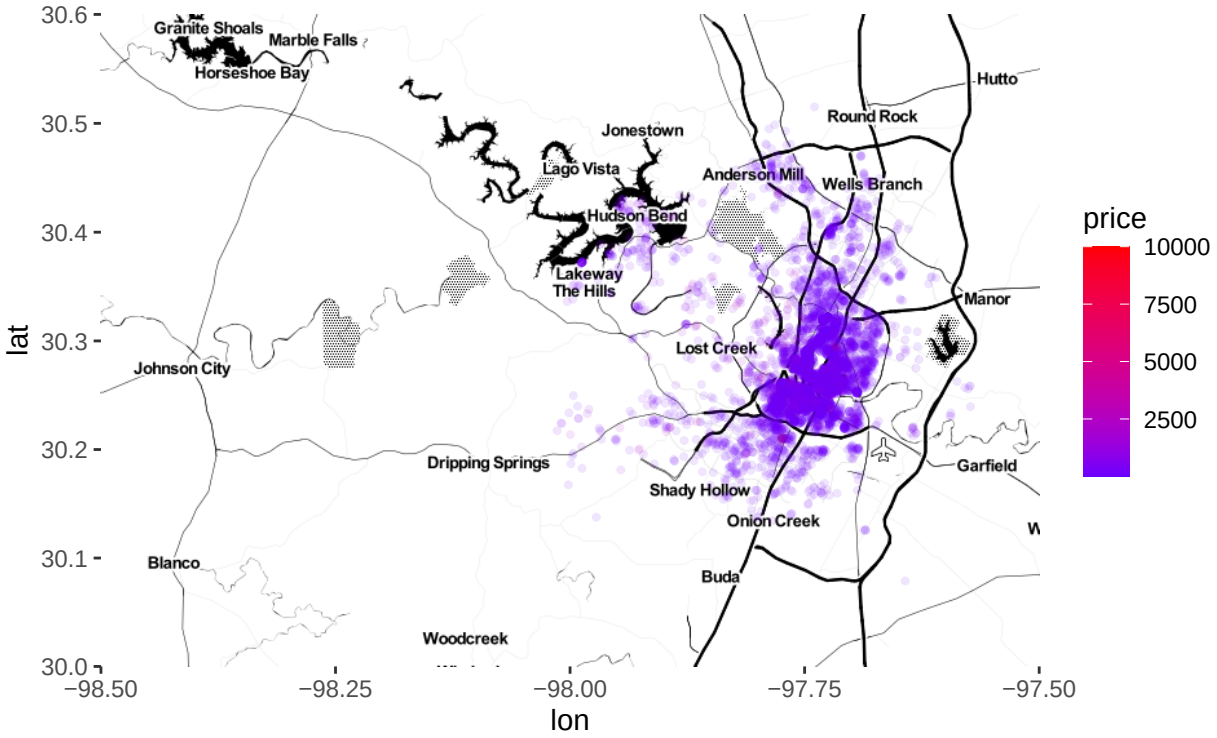
From correlation map, it's hard to investigate the factors that influence price. We will analyze price factors at prediction part.

4. Map Visualization

(1). Overall price on map

```
# get austin map
austin_map <- get_stamenmap(bbox = c(left = -98.5, bottom = 30,
    right = -97.5, top = 30.6), zoom = 10, maptype = "toner")
```

```
# overall price on map
ggmap(austin_map) + geom_point(data = austin, mapping = aes(x = longitude,
    y = latitude, color = price), alpha = 0.1, size = 0.9) +
    scale_colour_gradient(low = "#6A00FF", high = "#FF0000")
```

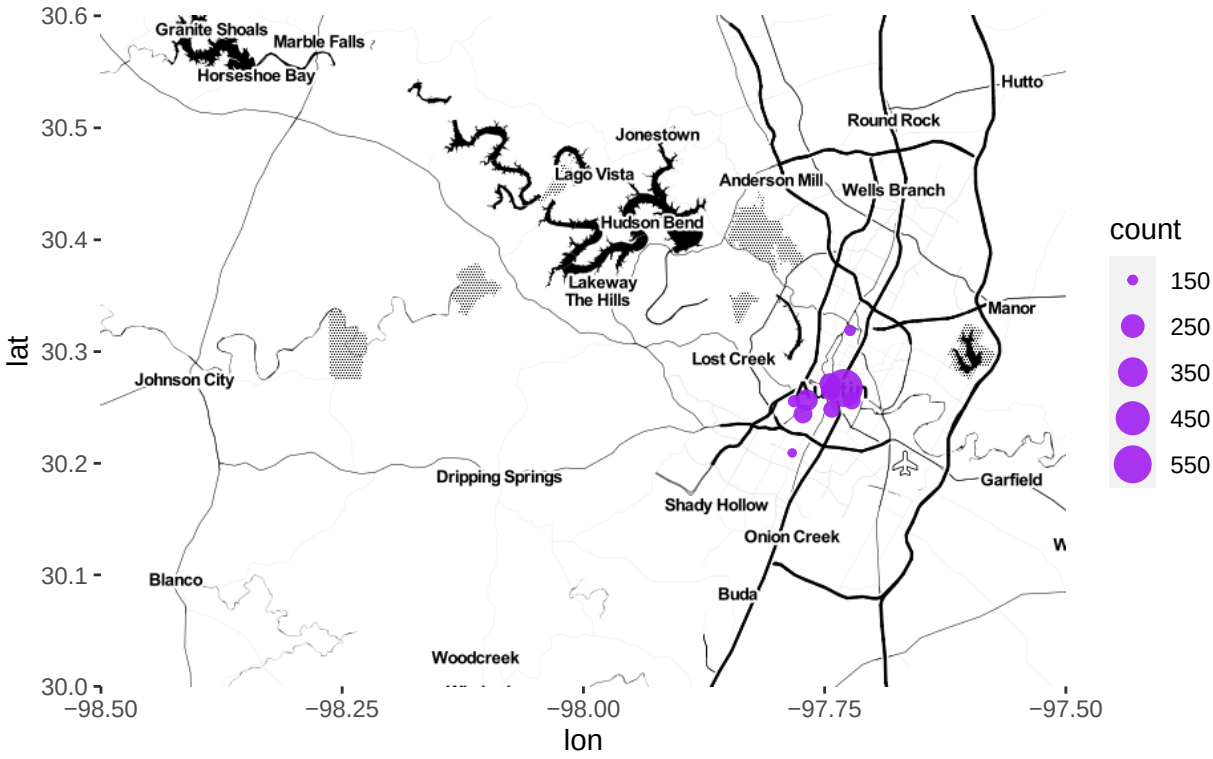


As stated previously, price for most of listings are below $2500.

In the map above of the city of Austin, Texas it can be observed that there is the high concentration of room listing which are purple in colour and according to the legend the color purple represents the price of $2500 and below, which can be observed from the above map and plot. It can also be inferred that the area might be the downtown or area where the population density is very high in comparison to the outskirts of Austin.

(2). Physical location for top 10 neighborhood

```
# get top 10 neighbourhood
top20 <- head(neighbourhood, 10)

ggmap(austin_map) + geom_point(data = top20, mapping = aes(x = long,
    y = lat, size = count), alpha = 0.9, color = "purple")
```

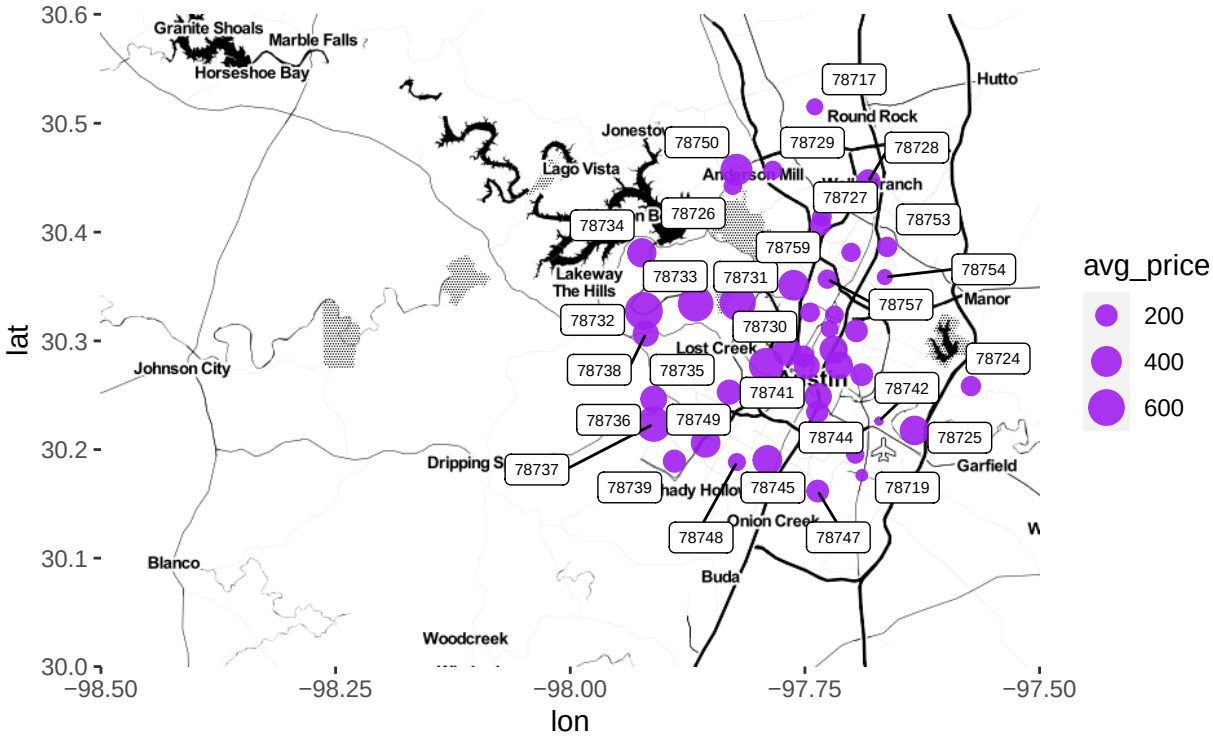


In the map above of the city of Austin, Texas it can be observed that there is the high concentration of airbnb hotels represented by the biggest purple circle. It can be infered that it might be the area where there is a lots of population concentration followed by large numbers of airbnbs. Similarly nearby outskirts areas have the lesser number of Airbnbs which can be infered from the descending size of the purple circles.

Neighborhoods near downtown Austin tend to have to more listings.

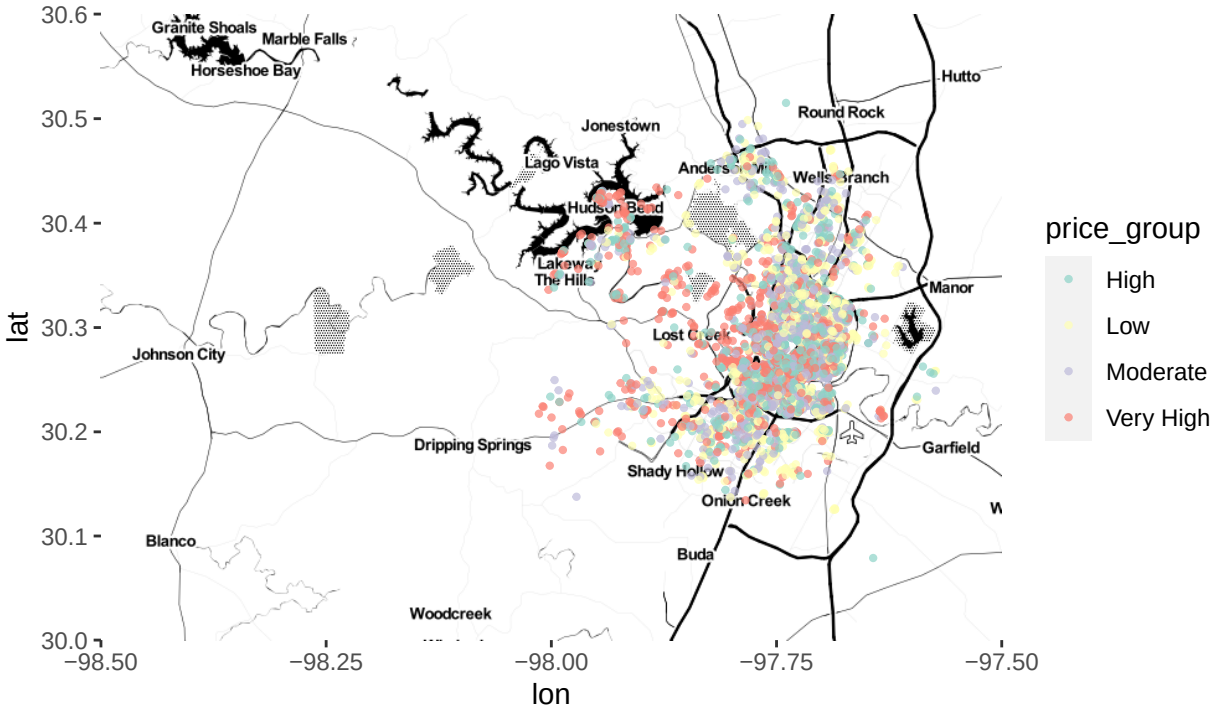
(3). Price Glimpse for Each Zip Area

```
# check average price for each zip area
austin_zip <- austin

austin_zip <- austin_zip %>%
    group_by(neighbourhood_cleansed) %>%
    summarise(latitude = first(latitude), longitude = first(longitude),
        avg_price = mean(price)) %>%
    ungroup()

ggmap(austin_map) + geom_point(data = austin_zip, mapping = aes(x = longitude,
    y = latitude, size = avg_price), alpha = 0.9, color = "purple") +
    geom_label_repel(data = austin_zip, aes(x = longitude, y = latitude,
        label = neighbourhood_cleansed), size = 2)
```



Average price according to different neighborhood is shown in the map, which is labelled by their respective zipcode in the top of the purple circle. biggest purple circle has the the average price of more than $600 which can be seen everywhere in austin which make it prevalent that most of the airbnbs have the average price of $600 or more

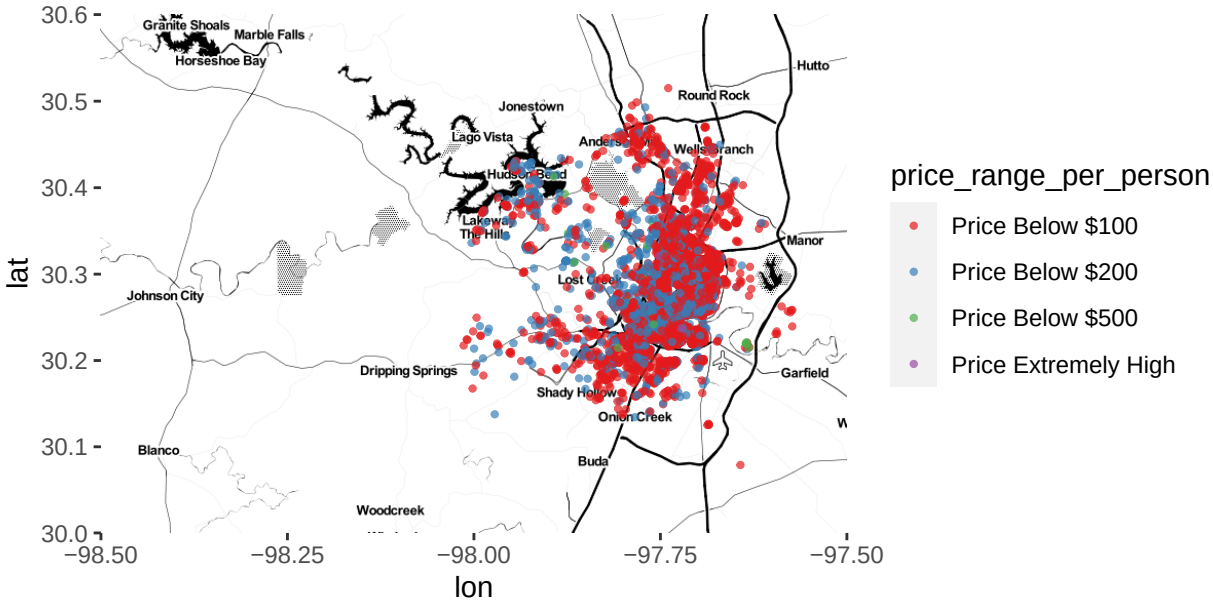
```
# Check price groups based on price quantiles
ggmap(austin_map) + geom_point(data = price_group, mapping = aes(x = longitude,
    y = latitude, color = price_group), alpha = 0.7, size = 0.8) +
    scale_color_brewer(palette = "Set3")
```



In the above map it can be observed that Airbnbs with different price groups are evenly distributed throughout the city, however it is evident that orange dot which represents the very high price range category is more observable and diffused from the bird eye view as it ranges from south from onion creek to max concentration on northwest hudson bend and few of them even at dripping springs in the west and obviously much more in the downtown area. The center of the city has all four price category coexisting on similar number.

(4). Average price per accommodates

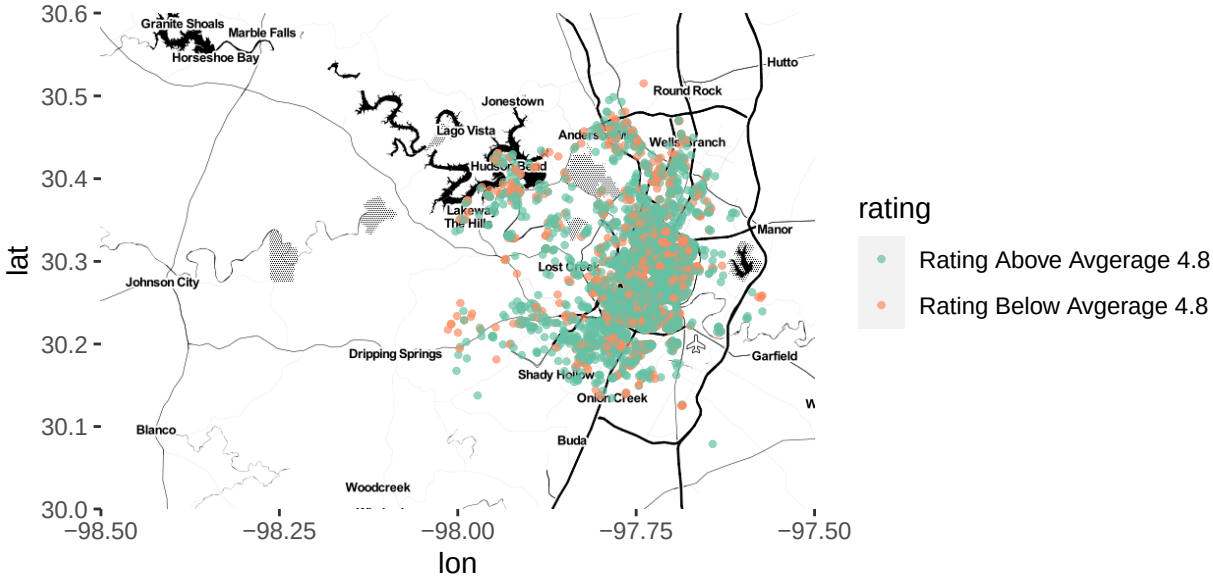
```
# check average price per person
avg_price <- austin %>%
    mutate(avg_price = price/accommodates) %>%
    mutate(price_range_per_person = ifelse(avg_price < 50, "Price Below $100",
        ifelse(avg_price < 200, "Price Below $200", ifelse(avg_price <
            500, "Price Below $500", "Price Extremely High"))))

ggmap(austin_map) + geom_point(data = avg_price, mapping = aes(x = longitude,
    y = latitude, color = price_range_per_person), alpha = 0.7,
    size = 0.8) + scale_color_brewer(palette = "Set1")
```



Most of price range per person are below $100 which is represented in the red dot which is high is number all around the map. Secondly there are price range per person below $200 that are represented in the blue dot. The price range per person that are green and brown dots are very low in numbers

(5). Overall rating on map

```
# overall rating on map
rating <- austin %>%
    mutate(rating = ifelse(review_scores_rating < mean(review_scores_rating),
        "Rating Below Avgerage 4.8", "Rating Above Avgerage 4.8"))

ggmap(austin_map) + geom_point(data = rating, mapping = aes(x = longitude,
    y = latitude, color = rating), alpha = 0.7, size = 0.8) +
    scale_color_brewer(palette = "Set2")
```

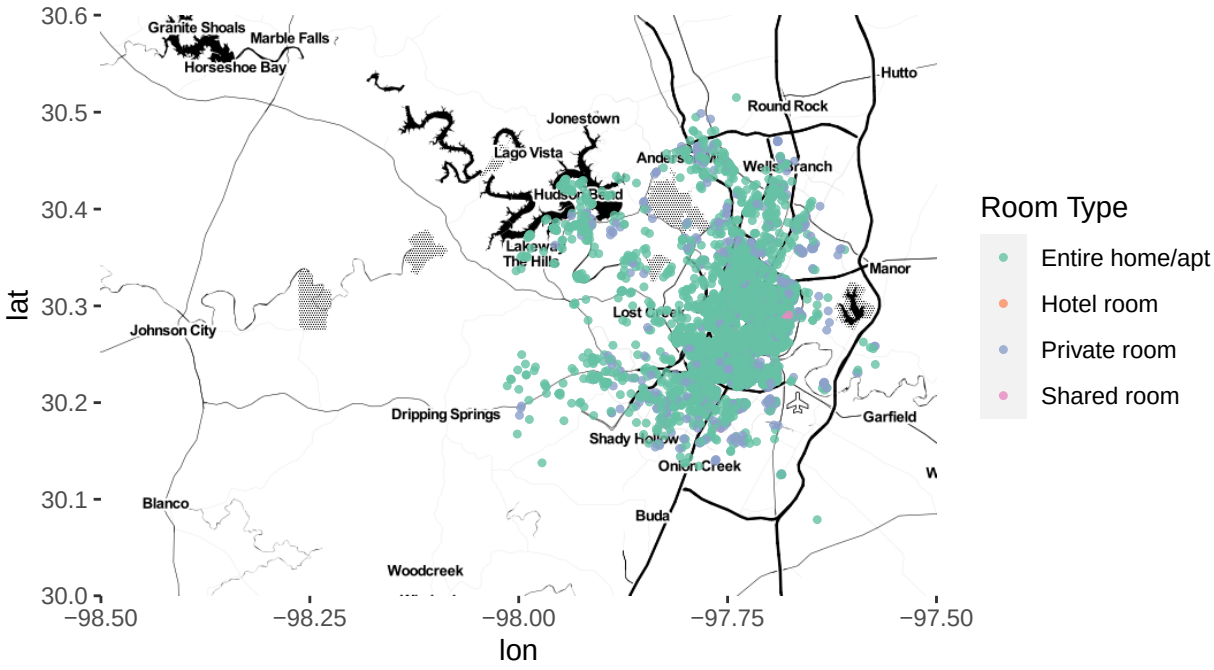


Here in the above map we can see that most of the rating for the hostels shows green which indicates rating

more than 4.8 and others are represents the rating reviews less than 4.8

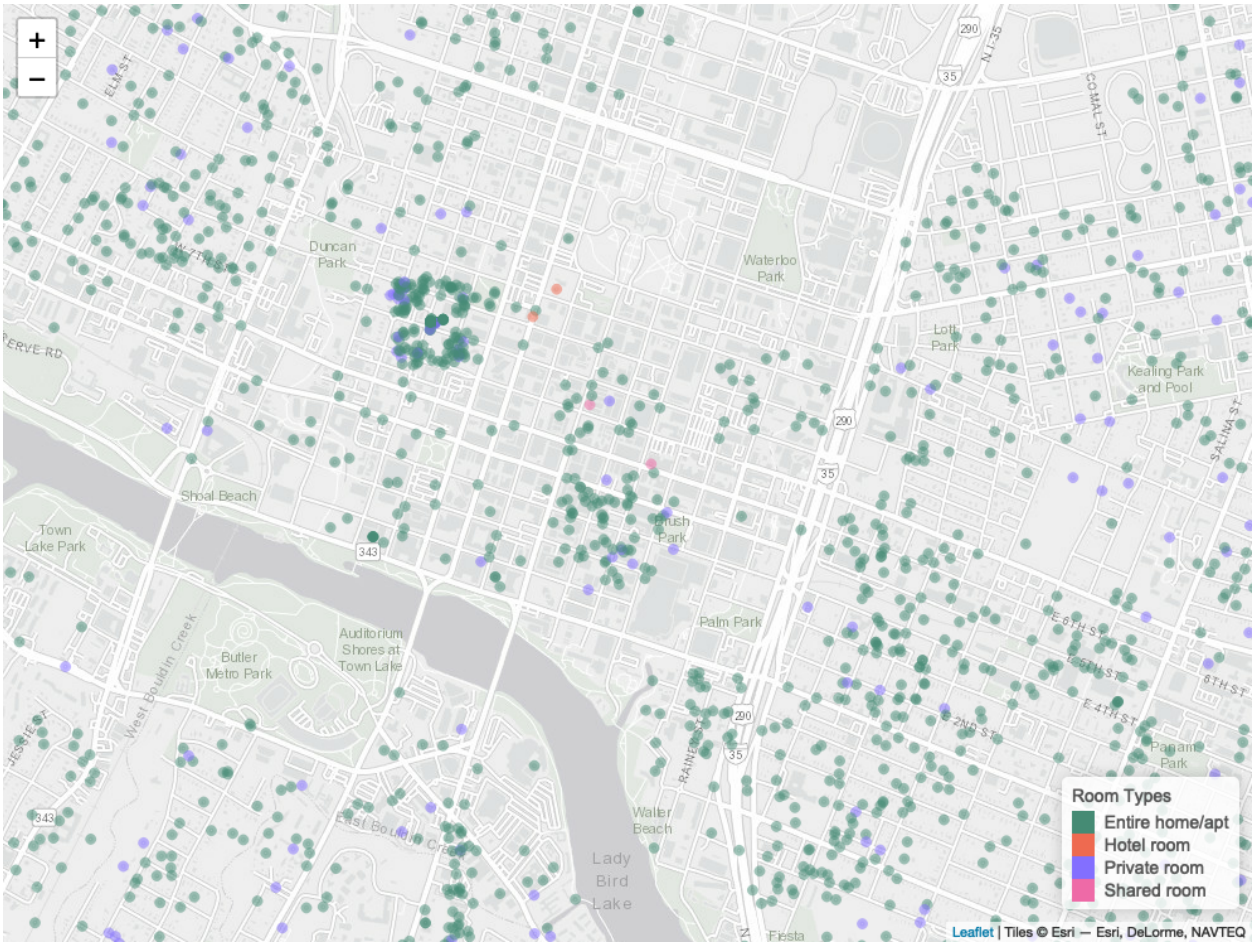
(6). Room type on map

```
# check room type
ggmap(austin_map) + geom_point(data = austin, mapping = aes(x = longitude,
    y = latitude, color = factor(room_type)), alpha = 0.8, size = 0.9) +
    scale_color_brewer(type = "qual", palette = "Set2", name = "Room Type")
```



In the map above it can be observed that most of the room types consists of Entire home/apt which is represented by the green dot, followed by private room which is shown by grey colour, we can see very less of hotel room and shared room in the map.

```
# let's check room_type on a zoomable map since from above
# map the entire home/apt covered other types of room
center_lon = median(austin$longitude, na.rm = TRUE)
center_lat = median(austin$latitude, na.rm = TRUE)

factpal <- colorFactor(c("aquamarine4", "coral2", "slateblue1",
    "hotpink2"), austin$room_type)

m <- leaflet(austin) %>%
    addProviderTiles("Esri.WorldGrayCanvas") %>%
    addCircles(lng = ~longitude, lat = ~latitude, color = ~factpal(room_type)) %>%
    setView(lng = center_lon, lat = center_lat, zoom = 15) %>%
    addLegend("bottomright", pal = factpal, values = ~room_type,
        title = "Room Types", opacity = 1)

saveWidget(m, "temp.html", selfcontained = FALSE)
webshot("temp.html", file = "leaflet_map.png", cliprect = "viewport")
```

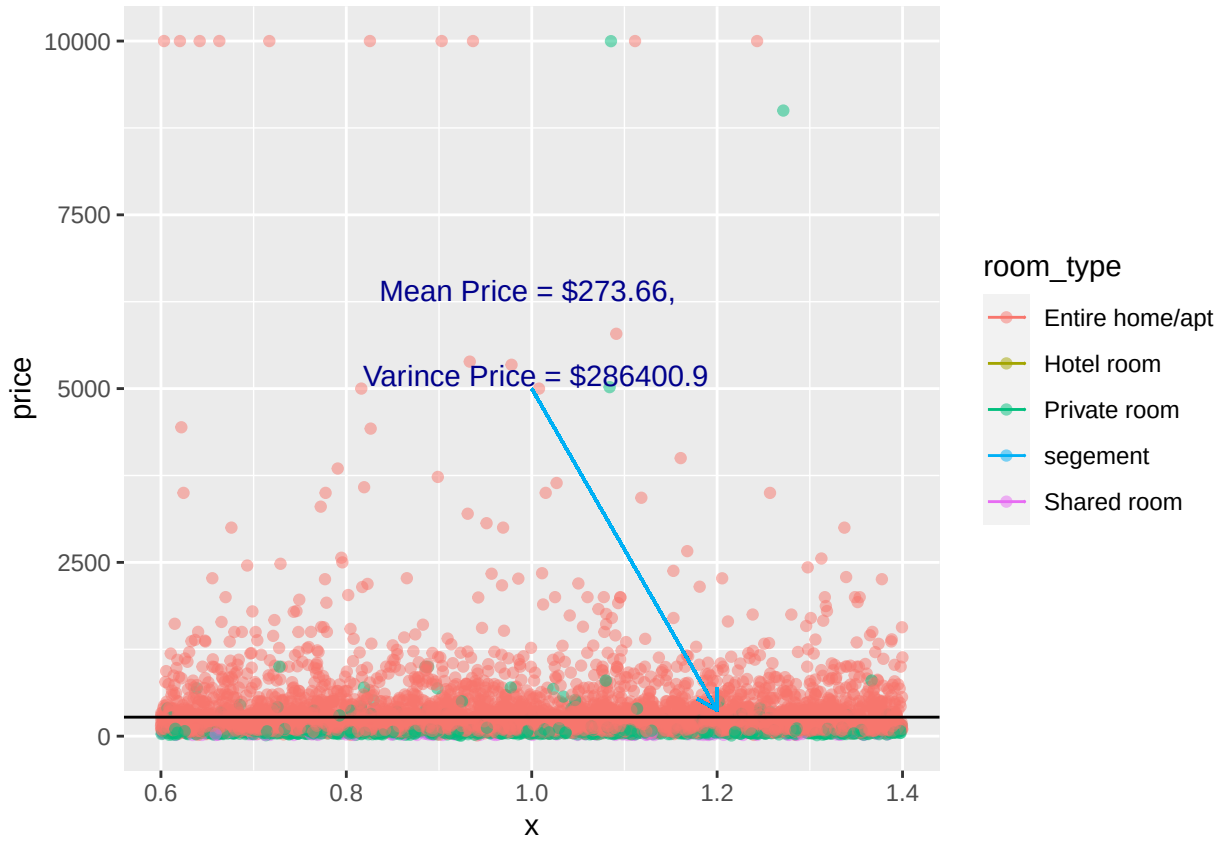


pdf is not able to show zoomed map, please check temp.html or in R if you are interested.

Part VI: Supervised/Unsupervised Learning

1. Check on price distribution

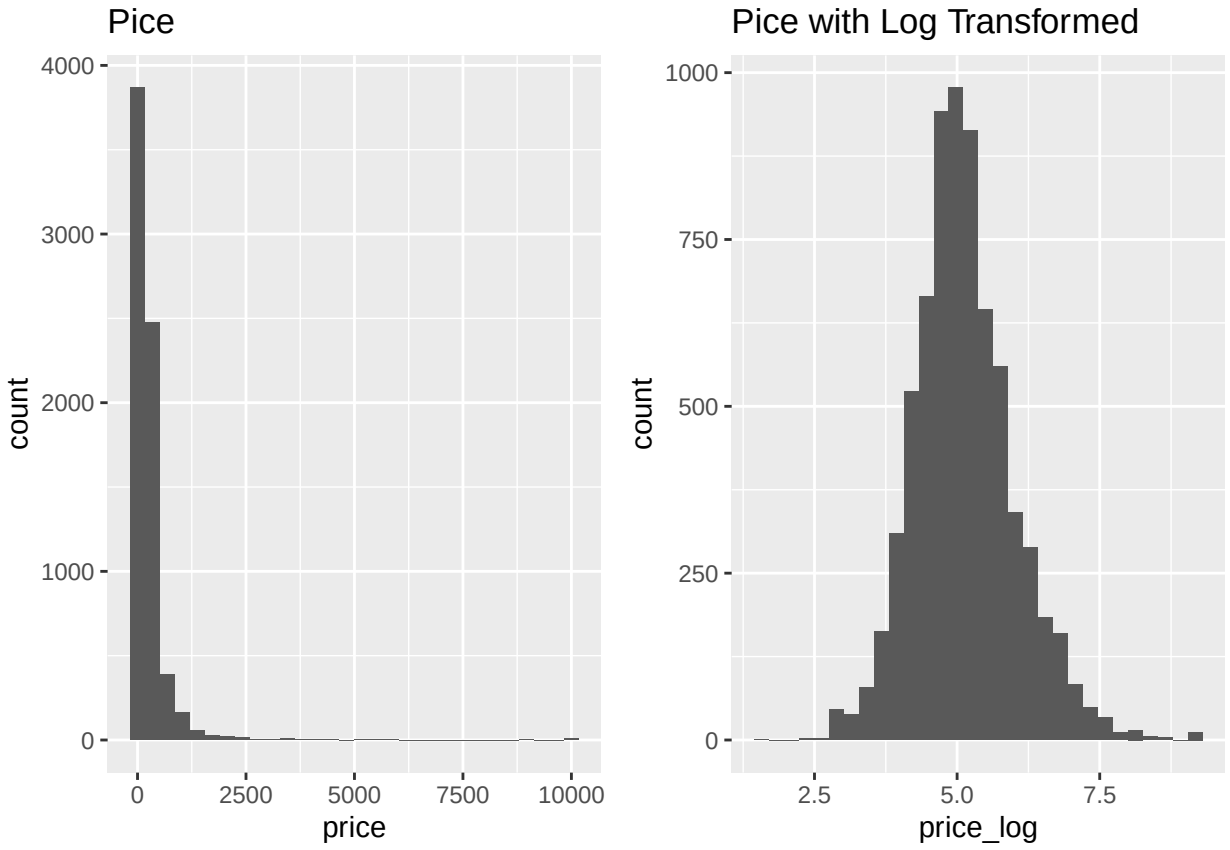
```
# plot of price
ggplot(austin, aes(x = 1, y = price, col = room_type)) + geom_jitter(alpha = 0.5) +
    geom_hline(yintercept = mean(austin$price)) + geom_segment(aes(x = 1,
    y = 5000, xend = 1.2, yend = mean(price) + 100, colour = "segement"),
    data = austin, arrow = arrow(length = unit(0.03, "npc")),
    size = 0.5) + annotate("text", x = 1, y = 5800, label = "Mean Price = $273.66,
           \n Varince Price = $286400.9",
    col = "darkblue")
```




```
# however, most price points are below $2500, check
# variance and plot
var(austin$price)
## [1] 286400.9
```

Transform price to log_price to improve skewness.

```
austin <- austin %>%
    mutate(price_log = log(price))

p1 <- ggplot(austin, aes(x = price)) + geom_histogram(bins = 30) +
    ggtitle("Pice")
p2 <- ggplot(austin, aes(x = price_log)) + geom_histogram(bins = 30) +
    ggtitle("Pice with Log Transformed")
grid.arrange(p1, p2, nrow = 1)
```



2. Applied different model for price prediction

Train-Test data is split with 50/50 ratio

(1). Step-wise

```
# get subset data from linear model
austin_reg <- austin %>%
    dplyr::select(-c("id", "latitude", "longitude", "host_neighbourhood",
        "price"))

austin_reg <- na.omit(austin_reg)
```

```
# randomly split data in r
set.seed(555)
sample_size = floor(0.5 * nrow(austin_reg))

picked = sample(seq_len(nrow(austin_reg)), size = sample_size)
train = austin_reg[picked, ]
test = austin_reg[-picked, ]

# options(max.print=999999)

reg <- as.formula(price_log ~ .)
austin_linreg <- lm(reg, data = train)
summary(austin_linreg)
##
```

```
## Call:
## lm(formula = reg, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3200 -0.2955 -0.0371  0.2377  4.3454
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       3.973e+00  2.699e-01  14.724  < 2e-16 ***
## host_response_timewithin a day    1.262e-02  1.953e-01   0.065 0.948466
## host_response_timewithin a few hours -1.085e-03  2.052e-01  -0.005 0.995783
## host_response_timewithin an hour -3.609e-03  2.073e-01  -0.017 0.986110
## host_is_superhostTRUE             1.931e-02  2.382e-02   0.811 0.417607
## host_total_listings_count         1.593e-04  7.807e-05   2.040 0.041451 *
## host_has_profile_picTRUE          2.156e-02  1.362e-01   0.158 0.874238
## host_identity_verifiedTRUE       -1.388e-02  3.392e-02  -0.409 0.682508
## neighbourhood_cleansed78702      -1.835e-01  5.044e-02  -3.638 0.000281 ***
## neighbourhood_cleansed78703      -3.000e-02  6.065e-02  -0.495 0.620920
## neighbourhood_cleansed78704      -9.769e-02  4.885e-02  -2.000 0.045633 *
## neighbourhood_cleansed78705      -4.575e-01  7.148e-02  -6.401 1.86e-10 ***
## neighbourhood_cleansed78717      -5.425e-01  2.127e-01  -2.550 0.010828 *
## neighbourhood_cleansed78719      -4.488e-01  2.591e-01  -1.732 0.083390 .
## neighbourhood_cleansed78721      -4.167e-01  8.257e-02  -5.047 4.83e-07 ***
## neighbourhood_cleansed78722      -3.733e-02  8.405e-02  -0.444 0.656968
## neighbourhood_cleansed78723      -3.546e-01  7.497e-02  -4.730 2.38e-06 ***
## neighbourhood_cleansed78724      -4.115e-01  1.988e-01  -2.070 0.038599 *
## neighbourhood_cleansed78725       6.475e-01  1.875e-01   3.454 0.000562 ***
## neighbourhood_cleansed78726      -6.658e-01  5.124e-01  -1.299 0.193929
## neighbourhood_cleansed78727      -6.115e-01  1.195e-01  -5.118 3.33e-07 ***
## neighbourhood_cleansed78728      -4.984e-01  1.859e-01  -2.681 0.007382 **
## neighbourhood_cleansed78729      -6.183e-01  1.290e-01  -4.793 1.74e-06 ***
## neighbourhood_cleansed78730      -5.076e-01  2.003e-01  -2.533 0.011357 *
## neighbourhood_cleansed78731      -1.683e-01  1.076e-01  -1.563 0.118090
## neighbourhood_cleansed78732      -1.058e-01  1.370e-01  -0.772 0.440108
## neighbourhood_cleansed78733       1.185e-01  1.482e-01   0.800 0.424034
## neighbourhood_cleansed78734      -1.928e-01  8.357e-02  -2.307 0.021144 *
## neighbourhood_cleansed78735      -2.864e-01  2.135e-01  -1.341 0.180041
## neighbourhood_cleansed78736      -3.232e-01  1.669e-01  -1.937 0.052844 .
## neighbourhood_cleansed78737      -3.902e-01  1.370e-01  -2.848 0.004436 **
## neighbourhood_cleansed78738      -5.029e-01  1.758e-01  -2.861 0.004257 **
## neighbourhood_cleansed78739      -6.877e-01  2.322e-01  -2.961 0.003099 **
## neighbourhood_cleansed78741      -4.110e-01  5.867e-02  -7.005 3.21e-12 ***
## neighbourhood_cleansed78742      -1.057e+00  3.644e-01  -2.900 0.003763 **
## neighbourhood_cleansed78744      -5.717e-01  8.749e-02  -6.535 7.78e-11 ***
## neighbourhood_cleansed78745      -3.530e-01  6.501e-02  -5.431 6.19e-08 ***
## neighbourhood_cleansed78746      -7.998e-02  8.156e-02  -0.981 0.326854
## neighbourhood_cleansed78747      -2.762e-02  2.978e-01  -0.093 0.926113
## neighbourhood_cleansed78748      -5.579e-01  1.167e-01  -4.781 1.85e-06 ***
## neighbourhood_cleansed78749      -1.611e-01  1.387e-01  -1.161 0.245584
## neighbourhood_cleansed78750      -7.203e-01  1.543e-01  -4.668 3.21e-06 ***
## neighbourhood_cleansed78751      -5.147e-01  6.433e-02  -8.002 1.90e-15 ***
## neighbourhood_cleansed78752      -4.845e-01  9.910e-02  -4.889 1.08e-06 ***
```

```
## neighbourhood_cleansed78753          -7.013e-01  1.250e-01  -5.610 2.26e-08 ***
## neighbourhood_cleansed78754          -7.043e-01  1.287e-01  -5.471 4.94e-08 ***
## neighbourhood_cleansed78756          -3.455e-01  1.032e-01  -3.347 0.000828 ***
## neighbourhood_cleansed78757          -6.326e-01  8.507e-02  -7.437 1.43e-13 ***
## neighbourhood_cleansed78758          -4.287e-01  8.206e-02  -5.225 1.89e-07 ***
## neighbourhood_cleansed78759          -3.813e-01  1.085e-01  -3.513 0.000451 ***
## room_typeHotel room                   1.074e-01  5.110e-01   0.210 0.833583
## room_typePrivate room                -4.346e-01  3.788e-02 -11.473  < 2e-16 ***
## room_typeShared room                 -1.449e+00  1.142e-01 -12.684  < 2e-16 ***
## accommodates                          1.089e-01  6.262e-03  17.385  < 2e-16 ***
## bedrooms                              9.859e-02  1.887e-02   5.224 1.90e-07 ***
## beds                                 -4.268e-02  7.124e-03  -5.992 2.39e-09 ***
## minimum_nights                       -1.956e-03  4.427e-04  -4.419 1.03e-05 ***
## maximum_nights                       -2.443e-05  2.138e-05  -1.143 0.253276
## number_of_reviews                    -7.885e-04  1.386e-04  -5.689 1.44e-08 ***
## review_scores_rating                  3.150e-01  6.846e-02   4.602 4.41e-06 ***
## review_scores_accuracy               -1.133e-01  6.140e-02  -1.846 0.065080 .
## review_scores_cleanliness             8.408e-02  4.396e-02   1.913 0.055898 .
## review_scores_checkin                 1.125e-02  5.388e-02   0.209 0.834642
## review_scores_communication           3.520e-02  5.250e-02   0.671 0.502577
## review_scores_location                5.465e-02  4.424e-02   1.235 0.216869
## review_scores_value                  -2.528e-01  5.143e-02  -4.915 9.50e-07 ***
## instant_bookableTRUE                  2.029e-02  2.431e-02   0.835 0.403935
## reviews_per_month                    -1.877e-03  1.068e-03  -1.758 0.078932 .
## count_host_verfitication              1.156e-02  5.328e-03   2.169 0.030173 *
## count_amenities_verfitication         2.554e-03  8.704e-04   2.934 0.003375 **
## host_response_rate1                   1.073e-01  1.987e-01   0.540 0.589159
## host_acceptance_rate1                -2.909e-01  7.063e-02  -4.118 3.95e-05 ***
## count_bathrooms                       2.192e-01  2.265e-02   9.679  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5085 on 2368 degrees of freedom
## Multiple R-squared:  0.6751, Adjusted R-squared:  0.6652
## F-statistic: 68.34 on 72 and 2368 DF,  p-value: < 2.2e-16
```

```
# try stepwise model (both direction)
austin_stepwise <- step(austin_linreg, direction = "both")
```

We have our linear model with lowest AIC = -3247.51, as price_log ~ host_total_listings_count + neighbourhood_cleansed + room_type + accommodates + bedrooms + beds + minimum_nights + number_of_reviews + review_scores_rating + review_scores_accuracy + review_scores_cleanliness + review_scores_value + reviews_per_month + count_host_verfitication + count_amenities_verfitication + host_acceptance_rate1 + count_bathrooms

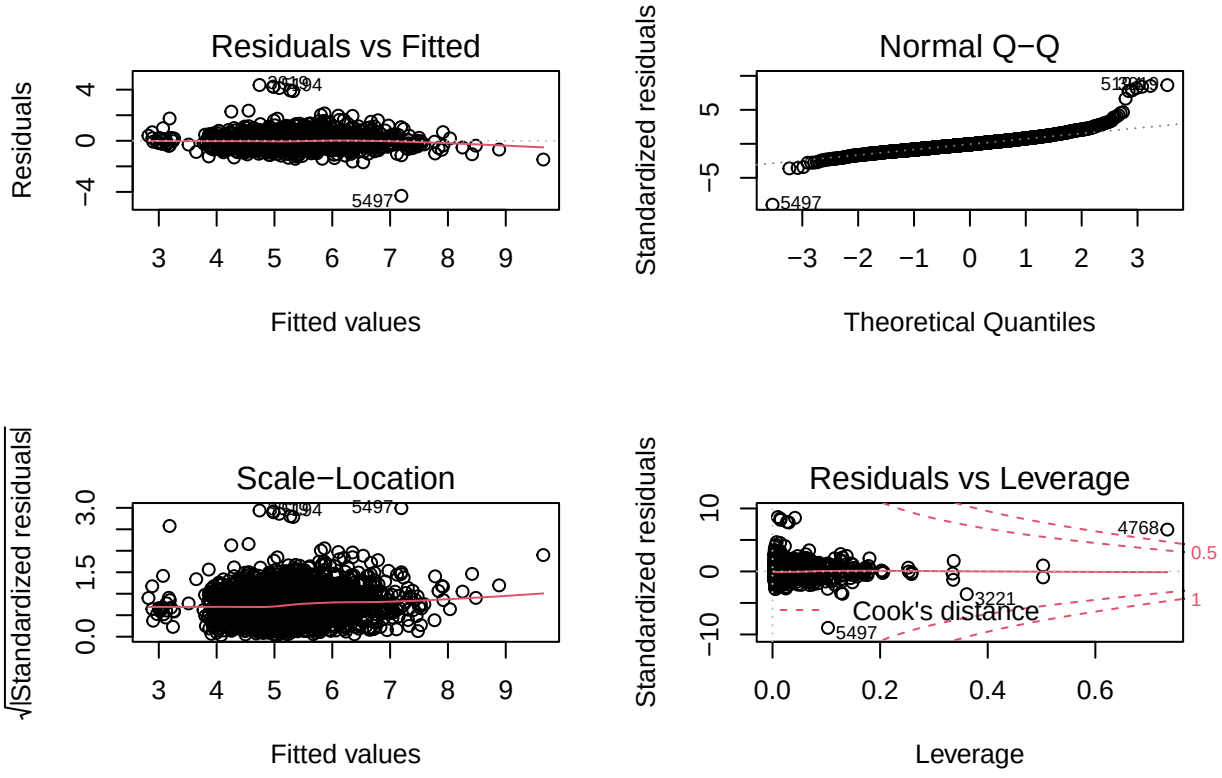
```
summary(austin_stepwise)
```

```
##
## Call:
## lm(formula = price_log ~ host_total_listings_count + neighbourhood_cleansed +
##     room_type + accommodates + bedrooms + beds + minimum_nights +
##     number_of_reviews + review_scores_rating + review_scores_accuracy +
##     review_scores_cleanliness + review_scores_value + reviews_per_month +
##     count_host_verfitication + count_amenities_verfitication +
##     host_acceptance_rate1 + count_bathrooms, data = train)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3046 -0.2974 -0.0399  0.2399  4.3632
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   4.230e+00  1.709e-01  24.744  < 2e-16 ***
## host_total_listings_count     1.451e-04  7.546e-05   1.923 0.054627 .
## neighbourhood_cleansed78702  -1.904e-01  4.982e-02  -3.822 0.000136 ***
## neighbourhood_cleansed78703  -3.183e-02  6.009e-02  -0.530 0.596323
## neighbourhood_cleansed78704  -9.921e-02  4.829e-02  -2.055 0.040036 *
## neighbourhood_cleansed78705  -4.652e-01  7.045e-02  -6.604 4.92e-11 ***
## neighbourhood_cleansed78717  -5.522e-01  2.119e-01  -2.605 0.009236 **
## neighbourhood_cleansed78719  -4.476e-01  2.582e-01  -1.733 0.083204 .
## neighbourhood_cleansed78721  -4.287e-01  8.173e-02  -5.246 1.69e-07 ***
## neighbourhood_cleansed78722  -4.003e-02  8.355e-02  -0.479 0.631909
## neighbourhood_cleansed78723  -3.622e-01  7.413e-02  -4.887 1.09e-06 ***
## neighbourhood_cleansed78724  -4.320e-01  1.975e-01  -2.188 0.028770 *
## neighbourhood_cleansed78725   6.216e-01  1.855e-01   3.350 0.000820 ***
## neighbourhood_cleansed78726  -6.651e-01  5.111e-01  -1.301 0.193318
## neighbourhood_cleansed78727  -6.179e-01  1.189e-01  -5.196 2.21e-07 ***
## neighbourhood_cleansed78728  -5.160e-01  1.852e-01  -2.786 0.005371 **
## neighbourhood_cleansed78729  -6.170e-01  1.281e-01  -4.818 1.54e-06 ***
## neighbourhood_cleansed78730  -5.126e-01  1.995e-01  -2.569 0.010262 *
## neighbourhood_cleansed78731  -1.755e-01  1.071e-01  -1.639 0.101257
## neighbourhood_cleansed78732  -1.080e-01  1.361e-01  -0.793 0.427654
## neighbourhood_cleansed78733   1.141e-01  1.477e-01   0.772 0.439929
## neighbourhood_cleansed78734  -1.890e-01  8.281e-02  -2.283 0.022532 *
## neighbourhood_cleansed78735  -2.842e-01  2.118e-01  -1.342 0.179661
## neighbourhood_cleansed78736  -3.130e-01  1.662e-01  -1.883 0.059788 .
## neighbourhood_cleansed78737  -4.014e-01  1.364e-01  -2.943 0.003277 **
## neighbourhood_cleansed78738  -5.067e-01  1.752e-01  -2.892 0.003868 **
## neighbourhood_cleansed78739  -6.892e-01  2.318e-01  -2.973 0.002975 **
## neighbourhood_cleansed78741  -4.247e-01  5.789e-02  -7.336 3.00e-13 ***
## neighbourhood_cleansed78742  -1.082e+00  3.622e-01  -2.988 0.002833 **
## neighbourhood_cleansed78744  -5.767e-01  8.674e-02  -6.649 3.66e-11 ***
## neighbourhood_cleansed78745  -3.582e-01  6.450e-02  -5.553 3.11e-08 ***
## neighbourhood_cleansed78746  -8.206e-02  8.084e-02  -1.015 0.310167
## neighbourhood_cleansed78747  -2.603e-02  2.972e-01  -0.088 0.930200
## neighbourhood_cleansed78748  -5.606e-01  1.161e-01  -4.828 1.46e-06 ***
## neighbourhood_cleansed78749  -1.581e-01  1.384e-01  -1.143 0.253338
## neighbourhood_cleansed78750  -7.125e-01  1.537e-01  -4.634 3.77e-06 ***
## neighbourhood_cleansed78751  -5.260e-01  6.352e-02  -8.280  < 2e-16 ***
## neighbourhood_cleansed78752  -4.949e-01  9.757e-02  -5.072 4.23e-07 ***
## neighbourhood_cleansed78753  -7.094e-01  1.243e-01  -5.707 1.29e-08 ***
## neighbourhood_cleansed78754  -7.148e-01  1.272e-01  -5.621 2.12e-08 ***
## neighbourhood_cleansed78756  -3.529e-01  1.025e-01  -3.444 0.000583 ***
## neighbourhood_cleansed78757  -6.380e-01  8.428e-02  -7.570 5.31e-14 ***
## neighbourhood_cleansed78758  -4.467e-01  8.131e-02  -5.494 4.34e-08 ***
## neighbourhood_cleansed78759  -3.845e-01  1.081e-01  -3.556 0.000383 ***
## room_typeHotel room           1.054e-01  5.099e-01   0.207 0.836277
## room_typePrivate room        -4.350e-01  3.750e-02 -11.600  < 2e-16 ***
## room_typeShared room         -1.461e+00  1.118e-01 -13.072  < 2e-16 ***
```

```
## accommodates                    1.085e-01  6.181e-03  17.554  < 2e-16 ***
## bedrooms                        9.996e-02  1.868e-02   5.352 9.55e-08 ***
## beds                           -4.301e-02  7.068e-03  -6.084 1.36e-09 ***
## minimum_nights                 -1.988e-03  4.399e-04  -4.519 6.52e-06 ***
## number_of_reviews              -7.681e-04  1.353e-04  -5.677 1.54e-08 ***
## review_scores_rating            3.368e-01  6.626e-02   5.083 4.01e-07 ***
## review_scores_accuracy         -9.490e-02  5.852e-02  -1.621 0.105046
## review_scores_cleanliness       9.029e-02  4.352e-02   2.075 0.038129 *
## review_scores_value            -2.331e-01  4.826e-02  -4.830 1.45e-06 ***
## reviews_per_month              -1.870e-03  1.065e-03  -1.756 0.079169 .
## count_host_verfitication        1.126e-02  4.843e-03   2.325 0.020147 *
## count_amenities_verfitication   2.717e-03  8.536e-04   3.184 0.001473 **
## host_acceptance_rate1          -2.504e-01  5.434e-02  -4.608 4.28e-06 ***
## count_bathrooms                 2.195e-01  2.253e-02   9.747  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5079 on 2380 degrees of freedom
## Multiple R-squared:  0.6742, Adjusted R-squared:  0.666
## F-statistic: 82.09 on 60 and 2380 DF,  p-value: < 2.2e-16
```

```
# check fit of step-wise linear regression
par(mfrow = c(2, 2))
plot(austin_stepwise)
```



Resides vs Fitted: ideal, linear assumptions hold;

Normal QQ: tails skewed but okay;

Scale-Location: residuals are not equally spread though;

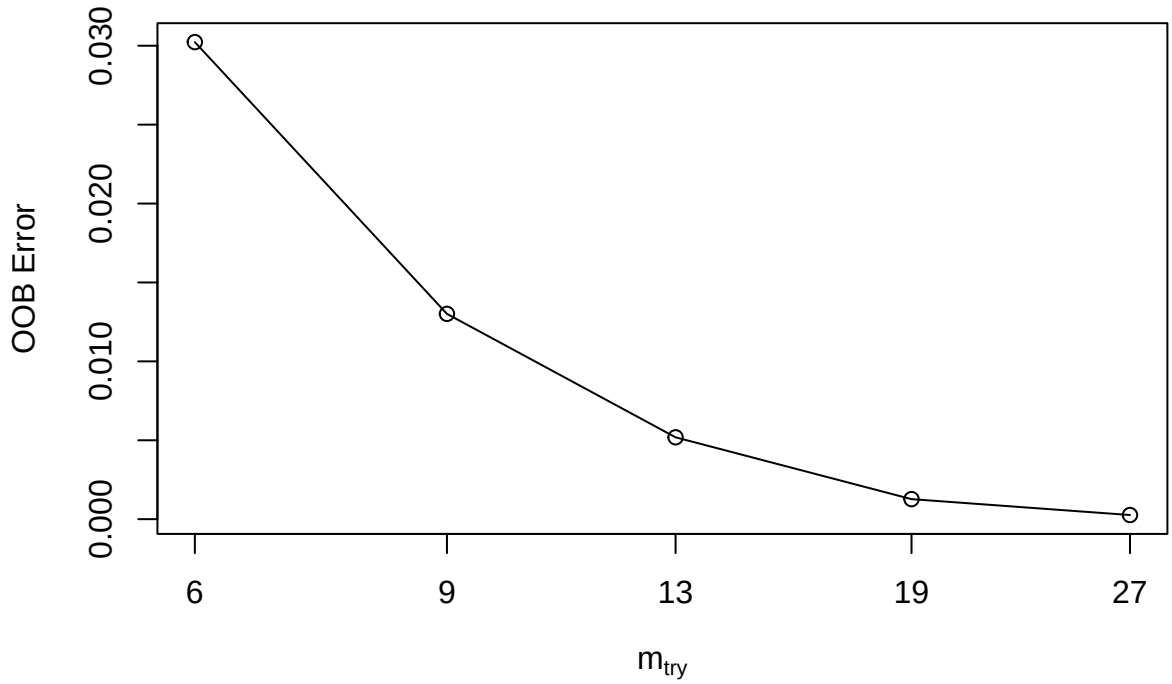
Residuals vs Leverage: some points are outside cook's distance.

In general the model works well, according to R^2 the model explains 67.42% of the fitted data.

```
# prediction & MSE
austin_stepwise_pred <- predict(austin_stepwise, test)
austin_stepwise_MSE <- mean((test$price_log - austin_stepwise_pred)^2)
austin_stepwise_MSE
## [1] 0.2609117
```

(2). Random Forest

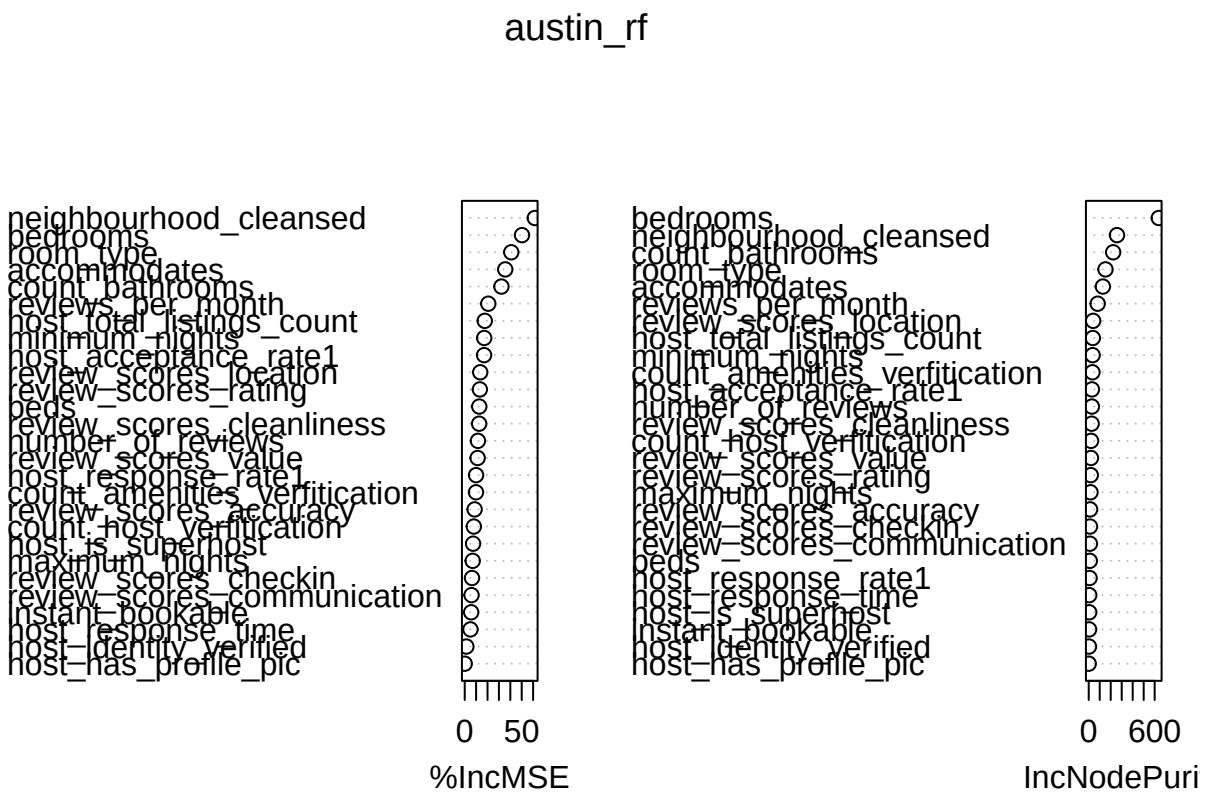
```
# select best mtry
mtry <- tuneRF(train[-1], train$price_log, ntreeTry = 500, stepFactor = 1.5,
    improve = 0.01, trace = TRUE, plot = TRUE)
## mtry = 9  OOB error = 0.01301169
## Searching left ...
## mtry = 6     OOB error = 0.03022996
## -1.323292 0.01
## Searching right ...
## mtry = 13    OOB error = 0.005186712
## 0.6013807 0.01
## mtry = 19    OOB error = 0.001270407
## 0.7550652 0.01
## mtry = 27    OOB error = 0.0002632028
## 0.79282 0.01
```




```
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]
print(mtry)
##    mtry     OOBError
## 6     6 0.0302299577
## 9     9 0.0130116937
## 13   13 0.0051867120
## 19   19 0.0012704065
## 27   27 0.0002632028
```

```
print(best.m)
## [1] 27
```

```
austin_rf <- randomForest(price_log ~ ., data = train, mtry = best.m,
    importance = TRUE)
austin_rf
##
## Call:
##  randomForest(formula = price_log ~ ., data = train, mtry = best.m,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##           Mean of squared residuals: 0.2276379
##                     % Var explained: 70.51
varImpPlot(austin_rf)
```



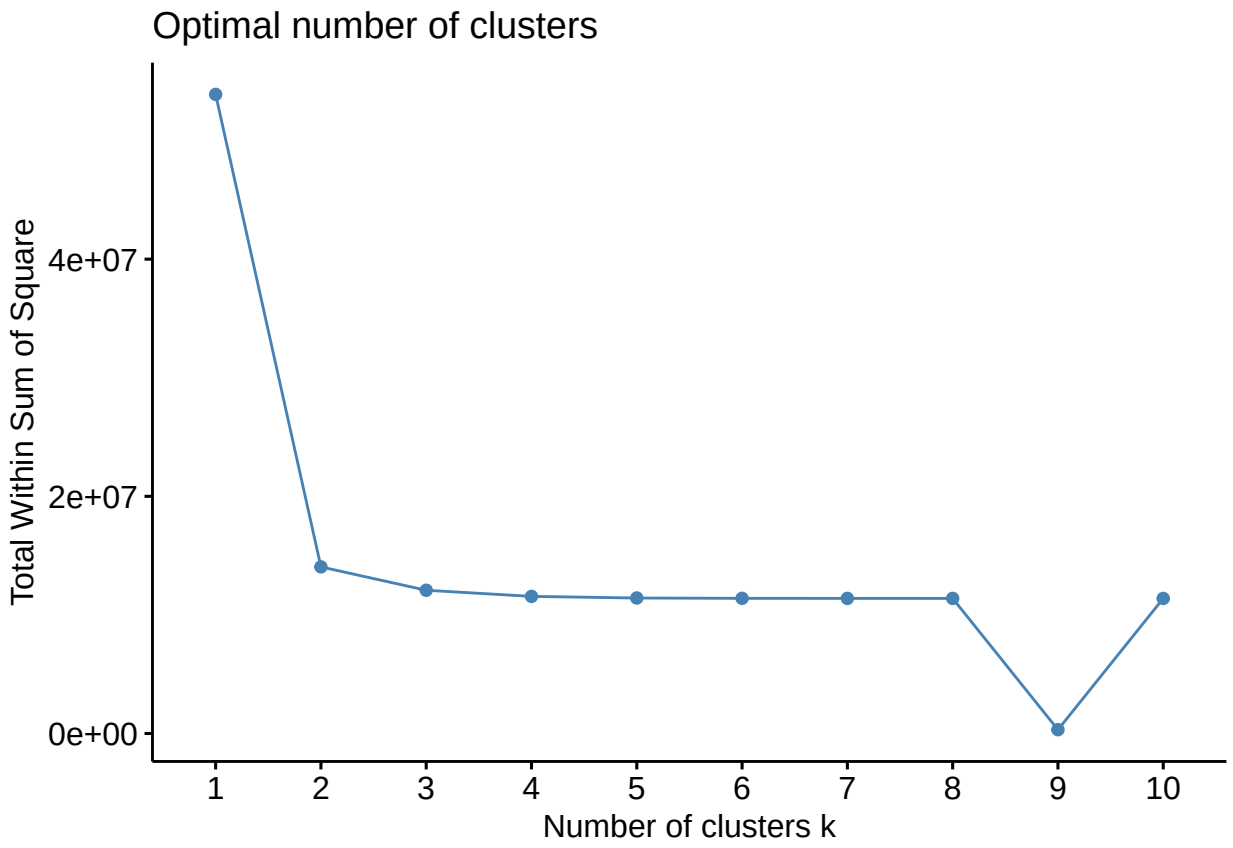

```
# prediction & MSE
austin_rf_pred <- predict(austin_rf, test)
austin_rf_MSE <- mean((test$price_log - austin_rf_pred)^2)
austin_rf_MSE
## [1] 0.2202858
calculateR2 <- function(y, yhat) {
    tss <- sum((y - mean(y))^2)
    rss <- sum((y - yhat)^2)
    r2 <- 1 - rss/tss
    return(r2)
}
```

```
austin_rf_r2 <- calculateR2(test$price_log, austin_rf_pred)
austin_rf_r2
## [1] 0.7048578
```

R^2 = 0.7045, in general the model works well, the model explains 70.45% of the fitted data.

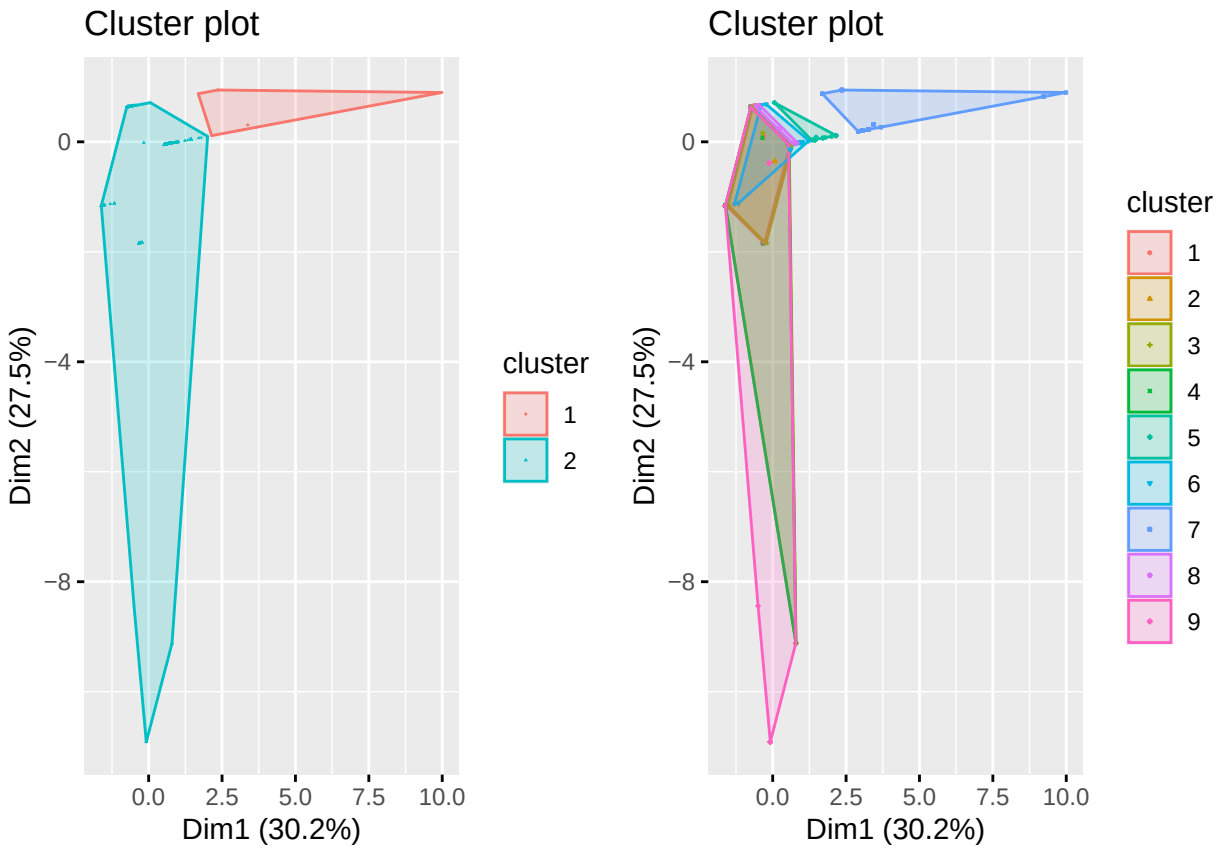
(3). Classification

```
# pick continuous column only

# str(train) column 2~5, 8 ~ 28

set.seed(555)
# determine k
fviz_nbclust(train[, 2:5, 8:28], kmeans, method = "wss")
```



Start with k = 2, there is no big change, but there is a great decreasing ar k = 9, so let's plot the cluster plot to determine k = 2 or k = 9.

```
austin_km2 <- kmeans(train[, 2:5, 8:28], centers = 2)
austin_km9 <- kmeans(train[, 2:5, 8:28], centers = 9)

p1 <- fviz_cluster(austin_km2, data = train[, 2:5, 8:28], pointsize = 0.1,
    labelsize = 0)
p2 <- fviz_cluster(austin_km9, data = train[, 2:5, 8:28], pointsize = 0.1,
    labelsize = 0)
grid.arrange(p1, p2, nrow = 1)
```



So we break train data into two cluster.

```
# access the cluster vector
o <- order(austin_km2$cluster)
clusters_df <- data.frame(row_id = as.numeric(rownames(train))[o],
    cluster_id = austin_km2$cluster[o])
clusters_df1 <- clusters_df[clusters_df$cluster_id == 1, ]
clusters_df2 <- clusters_df[clusters_df$cluster_id == 2, ]
```

```
# get dataframe for each cluster
austin_cf1 <- merge(train, clusters_df1, by = "row.names", all = TRUE)
austin_cf1 <- na.omit(austin_cf1)
austin_cf1 <- austin_cf1[-c(1, 7, 30)]
dim(austin_cf1)
## [1] 113  28
# 113 rows

austin_cf2 <- merge(train, clusters_df2, by = "row.names", all = TRUE)
austin_cf2 <- na.omit(austin_cf2)
austin_cf2 <- austin_cf2[-c(1, 7, 30)]
dim(austin_cf2)
## [1] 2328   28
# 2328 rows

austin_cf <- rbind(austin_cf1, austin_cf2)
```

```
# call glm() for each cluster
austin_cf1_lm <- lm(price_log ~ ., data = austin_cf1)
```

```
summary(austin_cf1_lm)
##
## Call:
## lm(formula = price_log ~ ., data = austin_cf1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6433 -0.3840 -0.1573  0.0559  3.9954
##
## Coefficients: (4 not defined because of singularities)
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       1.8448127  9.4897687   0.194  0.84631
## host_response_timewithin a few hours -1.1156820  1.9741306  -0.565  0.57341
## host_response_timewithin an hour     -0.1517586  2.3941396  -0.063  0.94960
## host_is_superhostTRUE                      NA         NA      NA       NA
## host_total_listings_count         -0.0002449  0.0006125  -0.400  0.69026
## host_has_profile_picTRUE                   NA         NA      NA       NA
## host_identity_verifiedTRUE                 NA         NA      NA       NA
## room_typePrivate room              0.3150584  0.9925479   0.317  0.75167
## accommodates                       0.1748694  0.1911080   0.915  0.36268
## bedrooms                           0.0132045  0.2603267   0.051  0.95966
## beds                              -0.2904809  0.3428326  -0.847  0.39913
## minimum_nights                    -0.0043594  0.0223816  -0.195  0.84602
## maximum_nights                     0.0002970  0.0004162   0.714  0.47741
## number_of_reviews                 -0.0024489  0.0047280  -0.518  0.60578
## review_scores_rating               0.9081564  0.6302078   1.441  0.15312
## review_scores_accuracy             0.1321897  0.7766097   0.170  0.86523
## review_scores_cleanliness         -0.5072374  0.3751928  -1.352  0.17986
## review_scores_checkin              0.0518752  0.3777391   0.137  0.89108
## review_scores_communication        0.2849652  0.4776627   0.597  0.55232
## review_scores_location            -1.3691127  0.4085301  -3.351  0.00119 **
## review_scores_value               -0.3541311  0.3485227  -1.016  0.31237
## instant_bookableTRUE              -0.1177236  0.6001048  -0.196  0.84493
## reviews_per_month                  0.0092413  0.0167019   0.553  0.58145
## count_host_verfitication          -0.0537489  0.0830775  -0.647  0.51933
## count_amenities_verfitication     -0.0120750  0.0346359  -0.349  0.72820
## host_response_rate1               10.5224069 11.6063350   0.907  0.36709
## host_acceptance_rate1             -2.3541640  3.1743091  -0.742  0.46029
## count_bathrooms                    0.1077205  0.3136179   0.343  0.73206
## cluster_id                                 NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8995 on 88 degrees of freedom
## Multiple R-squared:  0.2426, Adjusted R-squared:  0.03601
## F-statistic: 1.174 on 24 and 88 DF,  p-value: 0.2876
austin_cf2_lm <- lm(price_log ~ ., data = austin_cf2)
summary(austin_cf2_lm)
##
## Call:
## lm(formula = price_log ~ ., data = austin_cf2)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -4.4256 -0.3272 -0.0258  0.2595  4.5030
##
## Coefficients: (1 not defined because of singularities)
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       3.599e+00  2.714e-01  13.262  < 2e-16 ***
## host_response_timewithin a day   -2.830e-02  1.954e-01  -0.145   0.8849
## host_response_timewithin a few hours -2.978e-02  2.052e-01  -0.145   0.8846
## host_response_timewithin an hour -2.803e-02  2.070e-01  -0.135   0.8923
## host_is_superhostTRUE             5.144e-02  2.410e-02   2.135   0.0329 *
## host_total_listings_count         1.873e-04  3.624e-04   0.517   0.6053
## host_has_profile_picTRUE         -8.252e-02  1.356e-01  -0.609   0.5428
## host_identity_verifiedTRUE        1.078e-04  3.393e-02   0.003   0.9975
## room_typeHotel room               3.525e-01  5.138e-01   0.686   0.4928
## room_typePrivate room            -5.320e-01  3.587e-02 -14.831  < 2e-16 ***
## room_typeShared room             -1.634e+00  1.127e-01 -14.504  < 2e-16 ***
## accommodates                      1.078e-01  6.269e-03  17.201  < 2e-16 ***
## bedrooms                          9.569e-02  1.891e-02   5.060 4.53e-07 ***
## beds                             -3.968e-02  7.051e-03  -5.627 2.05e-08 ***
## minimum_nights                   -2.195e-03  4.429e-04  -4.957 7.70e-07 ***
## maximum_nights                   -4.026e-05  2.155e-05  -1.868   0.0619 .
## number_of_reviews                -6.334e-04  1.381e-04  -4.586 4.75e-06 ***
## review_scores_rating              3.165e-01  7.063e-02   4.481 7.81e-06 ***
## review_scores_accuracy           -1.248e-01  6.270e-02  -1.991   0.0466 *
## review_scores_cleanliness         1.875e-01  4.492e-02   4.175 3.10e-05 ***
## review_scores_checkin            -6.992e-02  5.672e-02  -1.233   0.2178
## review_scores_communication      -1.422e-02  5.466e-02  -0.260   0.7948
## review_scores_location            2.378e-01  4.415e-02   5.386 7.96e-08 ***
## review_scores_value              -3.335e-01  5.339e-02  -6.246 5.01e-10 ***
## instant_bookableTRUE              1.343e-02  2.482e-02   0.541   0.5885
## reviews_per_month                -2.480e-03  1.080e-03  -2.296   0.0218 *
## count_host_verfitication          1.001e-02  5.412e-03   1.850   0.0645 .
## count_amenities_verfitication     1.682e-03  8.668e-04   1.941   0.0524 .
## host_response_rate1               2.032e-01  1.975e-01   1.029   0.3036
## host_acceptance_rate1            -4.416e-01  7.011e-02  -6.298 3.59e-10 ***
## count_bathrooms                   2.241e-01  2.269e-02   9.877  < 2e-16 ***
## cluster_id                               NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5129 on 2297 degrees of freedom
## Multiple R-squared:  0.6622, Adjusted R-squared:  0.6578
## F-statistic: 150.1 on 30 and 2297 DF,  p-value: < 2.2e-16
```

Cluster 1 doesn't do well though, cluster 2 dose well.

Since clustering is unsupervised learning, we generally don't recommend this method for prediction.

Here we applied euclidean distance, to assign data in test into two clusters and check prediction.

```
austin_km2$centers
##   host_is_superhost host_total_listings_count host_has_profile_pic
## 1        0.08849558                 620.70796           1.0000000
## 2        0.49957045                  12.66366           0.9935567
##   host_identity_verified
```

```
## 1              1.0000000
## 2              0.8582474
```

```
# find cluster_id for austin_km2_test based on euclidean
# distance
clusters <- function(x, centers) {
    # compute squared euclidean distance from each sample
    # to each cluster center
    tmp <- sapply(seq_len(nrow(x)), function(i) apply(centers,
        1, function(v) sum((x[i, ] - v)^2)))
    max.col(-t(tmp))  # find index of min distance
}

# assign cluster_id to test dataset
cluster_id_test <- as.data.frame(clusters(test[, 2:5, 8:28],
    austin_km2[["centers"]]))
austin_cf_test <- cbind(test, cluster_id_test)
colnames(austin_cf_test)[29] <- "cluster_id"
head(austin_cf_test)
##    host_response_time host_is_superhost host_total_listings_count
## 6      within an hour             TRUE                        2
## 10     within an hour             TRUE                        1
## 11 within a few hours            FALSE                        1
## 12       within a day            FALSE                        2
## 13     within an hour             TRUE                        2
## 14     within an hour             TRUE                        2
##    host_has_profile_pic host_identity_verified neighbourhood_cleansed
## 6                  TRUE                  TRUE                 78704
## 10                 TRUE                  TRUE                 78741
## 11                 TRUE                 FALSE                 78731
## 12                 TRUE                  TRUE                 78705
## 13                 TRUE                 FALSE                 78704
## 14                 TRUE                 FALSE                 78704
##          room_type accommodates bedrooms beds minimum_nights maximum_nights
## 6  Entire home/apt            3        1    2              3            365
## 10 Entire home/apt            2        1    1             31            365
## 11 Entire home/apt            4        2    2              2             30
## 12 Entire home/apt            2        2    0             30           1125
## 13 Entire home/apt            6        1    3              1             15
## 14 Entire home/apt            4        1    1              2             90
##    number_of_reviews review_scores_rating review_scores_accuracy
## 6                254                 4.98                   4.96
## 10                44                 4.93                   5.00
## 11                32                 4.90                   4.93
## 12                30                 4.86                   4.81
## 13               810                 4.90                   4.93
## 14               146                 4.97                   4.95
##    review_scores_cleanliness review_scores_checkin review_scores_communication
## 6                       4.96                  4.99                        4.97
## 10                      5.00                  5.00                        5.00
## 11                      4.97                  4.97                        4.97
## 12                      4.77                  5.00                        4.96
## 13                      4.93                  4.96                        4.96
## 14                      4.99                  5.00                        5.00
```

```
##    review_scores_location review_scores_value instant_bookable
## 6                    4.96                4.89             TRUE
## 10                   4.79                4.89            FALSE
## 11                   4.90                4.77            FALSE
## 12                   4.81                4.88            FALSE
## 13                   4.94                4.81            FALSE
## 14                   4.95                4.83             TRUE
##    reviews_per_month count_host_verfitication count_amenities_verfitication
## 6               2.06                       5                           60
## 10              0.33                       4                           42
## 11              0.29                       6                           43
## 12              0.47                       6                           27
## 13             20.45                       4                           17
## 14            125.14                       4                           19
##    host_response_rate1 host_acceptance_rate1 count_bathrooms price_log
## 6                 1.00                 0.98               1  4.983607
## 10                1.00                 0.90               1  4.060443
## 11                1.00                 0.71               2  5.857933
## 12                0.71                 0.57               1  4.762174
## 13                1.00                 1.00               1  4.624973
## 14                1.00                 1.00               1  5.003946
##    cluster_id
## 6           2
## 10          2
## 11          2
## 12          2
## 13          2
## 14          2
```

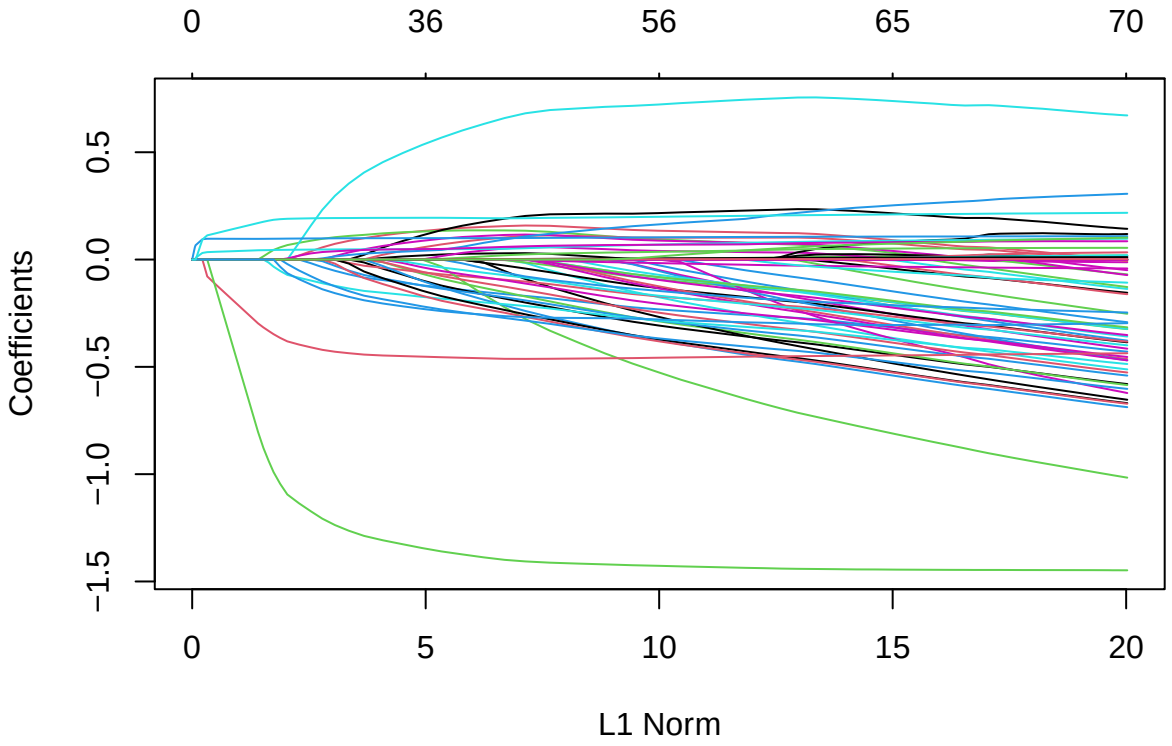
```
austin_cf_test1 <- austin_cf_test[austin_cf_test$cluster_id ==
    1, ]
austin_cf_test2 <- austin_cf_test[austin_cf_test$cluster_id ==
    2, ]

# prediction & MSE
austin_cf1_pred <- predict(austin_cf1_lm, austin_cf_test1)
austin_cf1_MSE <- mean((austin_cf_test1$price_log - austin_cf1_pred)^2)
austin_cf1_MSE
## [1] 0.9174165
austin_cf2_pred <- predict(austin_cf2_lm, austin_cf_test2)
austin_cf2_MSE <- mean((austin_cf_test2$price_log - austin_cf2_pred)^2)
austin_cf2_MSE
## [1] 0.2581971
# R-square
r2_cf1 <- calculateR2(austin_cf_test1$price_log, austin_cf1_pred)
r2_cf2 <- calculateR2(austin_cf_test2$price_log, austin_cf2_pred)
r2_cf1
## [1] -0.2679203
r2_cf2
## [1] 0.6535079
```

As we stated, cluster1 model doesn't fit well, while cluster 2 model does a good fit.

(4). Lasso

```
set.seed(555)
x <- model.matrix(price_log ~ ., austin_reg)[, -1]
y <- austin_reg$price_log
train <- sample(1:nrow(x), nrow(x)/2)
test <- (-train)
y.test <- y[test]
```

```
lasso.mod <- glmnet(x[train, ], y[train], alpha = 1, lambda = NULL)
plot(lasso.mod)
```




```
# best tuning parameter
cv.out <- cv.glmnet(x[train, ], y[train], alpha = 1)
bestlam <- cv.out$lambda.min
bestlam
## [1] 0.001924684
# model selection
austin_lasso <- glmnet(x, y, alpha = 1, lambda = NULL)
austin_lasso_coef <- predict(austin_lasso, type = "coefficients",
    s = bestlam)[, ]
austin_lasso_coef[austin_lasso_coef != 0]
##                          (Intercept) host_response_timewithin a few hours
##                         4.055356e+00                         -1.448651e-03
##     host_response_timewithin an hour           host_total_listings_count
##                         1.110452e-02                          1.965951e-04
##             host_has_profile_picTRUE         host_identity_verifiedTRUE
##                         9.918673e-03                         -2.945775e-02
##          neighbourhood_cleansed78702         neighbourhood_cleansed78703
##                        -3.125561e-02                          4.910431e-02
##          neighbourhood_cleansed78704         neighbourhood_cleansed78705
```

```
##                          2.369241e-02                          -2.910873e-01
##          neighbourhood_cleansed78717          neighbourhood_cleansed78719
##                         -4.600133e-01                         -4.362615e-01
##          neighbourhood_cleansed78721          neighbourhood_cleansed78722
##                         -2.596990e-01                         -4.131599e-02
##          neighbourhood_cleansed78723          neighbourhood_cleansed78724
##                         -3.006288e-01                         -4.108948e-01
##          neighbourhood_cleansed78725          neighbourhood_cleansed78726
##                          2.702211e-01                         -1.533183e-01
##          neighbourhood_cleansed78727          neighbourhood_cleansed78728
##                         -4.782448e-01                         -4.035627e-01
##          neighbourhood_cleansed78729          neighbourhood_cleansed78730
##                         -4.593081e-01                         -2.123619e-01
##          neighbourhood_cleansed78732          neighbourhood_cleansed78733
##                          2.256054e-02                          1.810165e-01
##          neighbourhood_cleansed78734          neighbourhood_cleansed78735
##                         -1.265101e-02                         -1.789137e-02
##          neighbourhood_cleansed78736          neighbourhood_cleansed78737
##                         -2.248130e-01                         -2.071102e-01
##          neighbourhood_cleansed78738          neighbourhood_cleansed78739
##                         -2.540207e-01                         -4.243220e-01
##          neighbourhood_cleansed78741          neighbourhood_cleansed78742
##                         -2.719501e-01                         -6.964440e-01
##          neighbourhood_cleansed78744          neighbourhood_cleansed78745
##                         -4.087137e-01                         -2.522023e-01
##          neighbourhood_cleansed78746          neighbourhood_cleansed78748
##                          5.887824e-03                         -4.931972e-01
##          neighbourhood_cleansed78749          neighbourhood_cleansed78750
##                         -2.234350e-01                         -4.665228e-01
##          neighbourhood_cleansed78751          neighbourhood_cleansed78752
##                         -3.868398e-01                         -4.094606e-01
##          neighbourhood_cleansed78753          neighbourhood_cleansed78754
##                         -4.972551e-01                         -4.838690e-01
##          neighbourhood_cleansed78756          neighbourhood_cleansed78757
##                         -2.870103e-01                         -4.267696e-01
##          neighbourhood_cleansed78758          neighbourhood_cleansed78759
##                         -3.224319e-01                         -2.460334e-01
##                 room_typeHotel room                 room_typePrivate room
##                          3.243953e-02                         -4.467498e-01
##                room_typeShared room                          accommodates
##                         -1.450864e+00                          8.065288e-02
##                               bedrooms                                  beds
##                          1.469840e-01                         -2.618906e-02
##                       minimum_nights                       maximum_nights
##                         -3.029767e-03                         -3.549079e-05
##                    number_of_reviews                  review_scores_rating
##                         -6.950568e-04                          2.263429e-01
##               review_scores_accuracy             review_scores_cleanliness
##                         -6.135267e-02                          1.602302e-01
##                review_scores_checkin                review_scores_location
##                         -6.374909e-03                          4.846910e-02
##                  review_scores_value                  instant_bookableTRUE
##                         -2.556716e-01                          3.030183e-02
```

```
##                   reviews_per_month            count_host_verfitication
##                        -2.146205e-03                        8.578830e-03
##       count_amenities_verfitication                  host_response_rate1
##                        2.527719e-03                        1.701000e-01
##               host_acceptance_rate1                      count_bathrooms
##                        -3.642403e-01                        1.890661e-01
```

We have significant variables above with their coefficents.

```
# MSE under the best tuning parameter
austin_lasso_pred <- predict(austin_lasso, s = bestlam, newx = x[test,
    ])
lasso_MSE <- mean((austin_lasso_pred - y[test])^2)
lasso_MSE
## [1] 0.2421914
# R-square
lasso_r2 <- calculateR2(y[test], austin_lasso_pred)
lasso_r2
## [1] 0.6755084
```

(5). Summary of 4 models

```
summary <- data.frame(MSE = c(austin_stepwise_MSE, austin_rf_MSE,
    austin_cf1_MSE, austin_cf2_MSE, lasso_MSE), Rsquared = c(summary(austin_stepwise)$r.squared,
    austin_rf_r2, r2_cf1, r2_cf2, lasso_r2))
row.names(summary) <- c("Step-wise", "Random Forest", "Classification-Cluster1",
    "Classification-Cluster2", "Lasso")
attr(summary, "row.names")
## [1] "Step-wise"               "Random Forest"
## [3] "Classification-Cluster1" "Classification-Cluster2"
## [5] "Lasso"
summary
##                               MSE   Rsquared
## Step-wise                0.2609117  0.6742034
## Random Forest            0.2202858  0.7048578
## Classification-Cluster1 0.9174165 -0.2679203
## Classification-Cluster2 0.2581971  0.6535079
## Lasso                    0.2421914  0.6755084
```

Here we have summary of 4 models we applied for Airbnb price prediction. Based on R^2 (the larger, the better) and MSE (the smaller, the better), the result shows that random forest forecast is the best, followed by Lasso.

Again, we don't suggest to use cluster method to do prediction since it's an unsupervised learning process. The clusters produced for each dataset could be different.