Statistical Learning Project

Aribnb data analysis

Anna Putina Federico Bottarelli Andrea Bello

2023-07-17

Project Description

This project aims at performing a regression analysis predicting the price of Airbnb places based on their characteristics. The price can vary depending on different factors such as location, number of guests allowed, cancellation policy, facilities provided and many others. The topic of interest is which characteristics influence the price the most, how strong the relationship is and whether it is linear or not. The answers to this questions provide interesting insights that can help hosts maximize their profits.

In this project, we perform the *exploratory data analysis*, followed by the *data preprocessing* and the *model selection*. As far as the methods are concerned, the *multiple linear regression* is investigated, together with its regularized variants, namely *Ridge regression* and *Lasso regression*.

Data Presentation

The dataset used in this project is taken from `Kaggle`.

Importing libraries and obtaining data.

We start off by retrieving the necessary libraries for the subsequent analysis, and then load the dataset from the file *airbnb.csv*:

```
library(tidyverse)
library(ggplot2)
library(corrplot)
library(ggcorrplot)
library(dplyr)
library(corrplot)
library(gridExtra)
library(grid)
library(GGally)
library(fastDummies)
library(stringr)
library(formatR)
library(MASS)
library(glmnet)
library(leaps)
```

```r
library(caret)
library(reshape2)
library(formatR)
airbnb <- read_csv("airbnb.csv")
```

The dataset contains 74111 instances of 29 instances each. We present the short the names of every column.

```r
glimpse(airbnb)
```

```
## Rows: 74,111
## Columns: 29
## $ id                       <dbl> 6901257, 6304928, 7919400, 13418779, 3808709, 1~
## $ log_price                <dbl> 5.010635, 5.129899, 4.976734, 6.620073, 4.74493~
## $ property_type            <chr> "Apartment", "Apartment", "Apartment", "House",~
## $ room_type                <chr> "Entire home/apt", "Entire home/apt", "Entire h~
## $ amenities                <chr> "{\"Wireless Internet\",\"Air conditioning\",Ki~
## $ accommodates             <dbl> 3, 7, 5, 4, 2, 2, 3, 2, 2, 2, 2, 4, 2, 6, 2, 2,~
## $ bathrooms                <dbl> 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.~
## $ bed_type                 <chr> "Real Bed", "Real Bed", "Real Bed", "Real Bed",~
## $ cancellation_policy      <chr> "strict", "strict", "moderate", "flexible", "mo~
## $ cleaning_fee             <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,~
## $ city                     <chr> "NYC", "NYC", "NYC", "SF", "DC", "SF", "LA", "L~
## $ description              <chr> "Beautiful, sunlit brownstone 1-bedroom in the ~
## $ first_review             <date> 2016-06-18, 2017-08-05, 2017-04-30, NA, 2015-0~
## $ host_has_profile_pic     <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,~
## $ host_identity_verified   <lgl> TRUE, FALSE, TRUE, TRUE, TRUE, TRUE, FALSE, TRU~
## $ host_response_rate       <chr> NA, "100%", "100%", NA, "100%", "100%", "100%",~
## $ host_since               <date> 2012-03-26, 2017-06-19, 2016-10-25, 2015-04-19~
## $ instant_bookable         <lgl> FALSE, TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, FAL~
## $ last_review              <date> 2016-07-18, 2017-09-23, 2017-09-14, NA, 2017-0~
## $ latitude                 <dbl> 40.69652, 40.76612, 40.80811, 37.77200, 38.9256~
## $ longitude                <dbl> -73.99162, -73.98904, -73.94376, -122.43162, -7~
## $ name                     <chr> "Beautiful brownstone 1-bedroom", "Superb 3BR A~
## $ neighbourhood            <chr> "Brooklyn Heights", "Hell's Kitchen", "Harlem",~
## $ number_of_reviews        <dbl> 2, 6, 10, 0, 4, 3, 15, 9, 159, 2, 82, 29, 0, 13~
## $ review_scores_rating     <dbl> 100, 93, 92, NA, 40, 100, 97, 93, 99, 90, 93, 9~
## $ thumbnail_url            <chr> "https://a0.muscache.com/im/pictures/6d7cbbf7-c~
## $ zipcode                  <chr> "11201", "10019", "10027", "94117.0", "20009", ~
## $ bedrooms                 <dbl> 1, 3, 1, 2, 0, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1,~
## $ beds                     <dbl> 1, 3, 3, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 3, 1, 1,~
```

$

$

Data Preprocessing

Before conducting exploratory data analysis, it is indeed crucial to clean and format the data properly. This process involves organizing and tidying up the data, removing what is not needed and identifying what is missing, and other issues that might affect the quality and integrity of the data. In this process, you may also need to convert the data from one format to another and consolidate everything into one standardized format across all data.

Variable	Description
id	Unique identifier for each accommodation
log_price	Logarithm of the price of the accommodation
property_type	Type of property (e.g., Apartment, House)
room_type	Type of room (e.g., Entire home/apt, Private room)
amenities	List of amenities provided
accommodates	Number of people the accommodation can host
bathrooms	Number of bathrooms
bed_type	Type of bed (e.g., Real Bed, Futon)
cancellation_policy	Cancellation policy (e.g., strict, moderate, flexible)
cleaning_fee	Whether a cleaning fee is charged (TRUE or FALSE)
city	City where the accommodation is located
description	Description of the accommodation
first_review	Date of the first review
host_has_profile_pic	Whether the host has a profile picture (TRUE or FALSE)
host_identity_verified	Whether the host's identity is verified (TRUE or FALSE)
host_response_rate	Response rate of the host
host_since	Date when the host joined Airbnb
instant_bookable	Whether the accommodation is instantly bookable (TRUE or FALSE)
last_review	Date of the last review
latitude	Latitude of the accommodation
longitude	Longitude of the accommodation
name	Name of the accommodation
neighbourhood	Neighbourhood where the accommodation is located
number_of_reviews	Number of reviews
review_scores_rating	Average rating score
thumbnail_url	URL of the accommodation's thumbnail image
zipcode	Zip code of the accommodation
bedrooms	Number of bedrooms
beds	Number of beds

Table 1: Description of Airbnb Accommodation Dataset Variables

Removing not relevant variables

In the initial dataset some features are not relevant for the analysis. They can be removed without additional exploration. The variables `id` and `thumbnail_url` represent identification characteristics. `Description` and `name` could be processed using some more advanced machine learning techniques which we do not apply in this project. We expect not to loose a lot of information by dropping them because there are other variables in the dataset that can partially replace `description`, for example, `amenities` and `neighbourhood`. We also discard `zipcode` refering to thet fact that it consists of fairly unique information (categorical feature with many categories) and indicates location which is represented in `city` and `neighbourhood` as well.

```
airbnb <- airbnb %>% dplyr::select(-id, -zipcode, -thumbnail_url, -name, -description)
```

Variables types

To ensure proper utilization of variables for modeling, it is important to appropriately change their types.

```
airbnb$beds = as.integer(airbnb$beds)
airbnb$property_type = as.factor(airbnb$property_type)
airbnb$room_type = as.factor(airbnb$room_type)
```

```r
airbnb$city = as.factor(airbnb$city)
airbnb$neighbourhood = as.factor(airbnb$neighbourhood)
airbnb$bed_type = as.factor(airbnb$bed_type)
airbnb$cancellation_policy = as.factor(airbnb$cancellation_policy)
airbnb$host_has_profile_pic = as.factor(airbnb$host_has_profile_pic)
airbnb$host_identity_verified = as.factor(airbnb$host_identity_verified)
airbnb$instant_bookable = as.factor(airbnb$instant_bookable)
airbnb$cleaning_fee = as.factor(airbnb$cleaning_fee)

# Replace spaces with underscores in the levels of the factor variables
airbnb$room_type <- gsub(" ", "_", airbnb$room_type)
airbnb$bed_type <- gsub(" ", "_", airbnb$bed_type)
# Replace "-" with underscores in the levels of the factor variables
airbnb$bed_type <- gsub("-", "_", airbnb$bed_type)
```

Now we will perform some preprocessing step, here divided by variables types ### Factors fixing We address two factor variables that may cause issues during modeling. The first variable is `amenities`, which represents a list of services or rules of the Airbnb's location. We preprocess the data by removing specified amenities and converting the remaining ones into a count of the number of amenities per hotel.

```r
amenities_to_remove <- c("translation missing: en.hosting_amenity_49", "translation missing: en.hosting
    "translationmissing:en.hosting_amenity_49", "translationmissing:en.hosting_amenity_50")

# Modify the existing code to remove curly braces and specified amenities
airbnb$amenities <- lapply(airbnb$amenities, function(x) {
    words <- strsplit(x, ",")[[1]]
    words <- gsub("[\" ]", "", words)
    words <- gsub("^\\{", "", words)  # Remove opening curly brace
    words <- gsub("\\}$", "", words)  # Remove closing curly brace
    words <- words[!words %in% amenities_to_remove]  # Remove specified amenities
    return(words)
})
```

We have added a variable representing the number of amenities per hotel instead of the complete list, but we are keeping the list for possible future reference.

```r
airbnb$amenities_count <- lapply(airbnb$amenities, function(x) {
  lengths <- length(x)
  return(as.integer(lengths))
})
airbnb$amenities_count <- unlist(airbnb$amenities_count)
```

Now we process `property_type`. To simplify the modeling process, we group similar property types together. We define two dictionaries (dict1 and dict2) to map the original property types to simplified categories. The grouping is done to create a simpler factorial variable for modeling.

```r
dict2 <- list(Apartment = c("Apartment", "Condominium", "Timeshare",
    "Loft", "Serviced apartment", "Guest suite"), House = c("House",
    "Vacation home", "Villa", "Townhouse", "In-law", "Casa particular"),
    Hotel = c("Hostel", "Guesthouse", "Boutique hotel", "Bed & Breakfast"),
    Cheap = c("Dorm", "Tent", "Parking", "Camper/RV", "Bungalow"),
    Other = c("Island", "Castle", "Yurt", "Hut", "Chalet", "Treehouse",
```

```
        "Earth House", "Tipi", "Cave", "Train", "Parking Space",
        "Lighthouse", "Boat", "Cabin"))
```

```
airbnb <- airbnb %>%
  mutate(property_type = case_when(
    property_type %in% dict2$Apartment ~ "Apartment",
    property_type %in% dict2$House ~ "House",
    property_type %in% dict2$Hotel ~ "Hotels",
    property_type %in% dict2$Cheap ~ "Cheap",
    TRUE ~ "Other"))
airbnb$property_type = as.factor(airbnb$property_type)
```

The category "Cheap" represents a group of accommodations that are commonly considered more affordable.

Dates to number

We must change dates in format that are usable in statistical modeling, `first_review`, `last_review` and `host_since` are all variables related to the host information, for a purpose of statistical modeling is more useful the `host_since` variable since we can compute the elapsed time since the host has been on the site. A distance is easier to model because it removes the absolute value from the date.

```
# host since
airbnb <- airbnb %>%
  mutate(host_since = as.integer(format(as.Date(airbnb$host_since,
                                                format = "%Y-%m-%d"),"%Y")))
# first review
airbnb <- airbnb %>%
  mutate(first_review = as.integer(format(as.Date(airbnb$first_review,
                                                format = "%Y-%m-%d"),"%Y")))
# last review
airbnb <- airbnb %>%
  mutate(last_review = as.integer(format(as.Date(airbnb$last_review,
                                                format = "%Y-%m-%d"),"%Y")))
```

Due to the logical positive correlation between `first_review` and `host_since`, we have decided to remove first_review from the analysis.

```
cor(airbnb$first_review, airbnb$host_since, use ="pairwise.complete.obs")
```

```
## [1] 0.5239194
```

```
airbnb <- airbnb %>% dplyr::select(-first_review)
cor(airbnb$last_review, airbnb$host_since, use ="pairwise.complete.obs")
```

```
## [1] 0.149301
```

We have regrouped the variable `last_review` into three levels based on the years. The new grouping reflects three categories: "2015 and earlier," "2016," and "2017." Each observation in the dataset has been assigned to one of these levels, allowing for easier analysis and interpretation.

```r
# Create a vector with the desired levels
levels <- c("2015 and earlier", "2016", "2017")

# Define the breakpoints for the cut function
breakpoints <- c(-Inf, 2015, 2016, Inf)

# Use the cut function to group the variable
airbnb$last_review <- cut(airbnb$last_review, breaks = breakpoints,
    labels = levels, right = FALSE)

# Convert the new variable to a factor
airbnb$last_review <- as.factor(airbnb$last_review)

# Get the current year
current_year <- 2017
# Transform host_since into numeric representation
airbnb$host_since <- ifelse(!is.na(airbnb$host_since), current_year -
    as.numeric(airbnb$host_since) + 1, NA)
```

Percentages fixing

We also fix `host_response_rate` and `review_scores_rating` both percentages. The code below converts these string into numeric values between 0 and 1.

```r
# host_response_rate
response <- airbnb$host_response_rate
airbnb$host_response_rate <- ifelse(
  grepl("%$", response),
  as.numeric(sub("%$", "", response))/100,
  suppressWarnings(as.numeric(response)))
# review_score_rating
airbnb$review_scores_rating <- airbnb$review_scores_rating/100
```

Missing Values

In the dataset some variables have missing values. To address the problem, we should know the amount of NA's and the information of the variables containing them. In some cases, the mode/median/mean imputation can be applied, while in other situations the variable can be removed from the dataset. The proportion of missing values for every column is shown below.

```r
na_counts = (round(colSums(is.na(airbnb))/nrow(airbnb), 4))
na_counts[na_counts>0]
```

```
##              bathrooms   host_has_profile_pic host_identity_verified
##                 0.0027                 0.0025                 0.0025
##     host_response_rate             host_since            last_review
##                 0.2469                 0.0025                 0.2136
##          neighbourhood   review_scores_rating               bedrooms
##                 0.0927                 0.2256                 0.0012
##                   beds
##                 0.0018
```

Point-Biserial Correlation: This is a correlation coefficient specifically designed for measuring the relationship between a continuous variable and a binary variable. It is an extension of the Pearson correlation coefficient and ranges from -1 to 1, where 0 indicates no relationship. This measure assumes that the continuous variable follows a normal distribution.

Binary variables For `host_identity_verified`, `host_profile_pictures_mode` and `last_review` mode imputation is implemented:

```
# Display the table of host_identity_verified levels
print(table(airbnb$host_identity_verified))
```

```
##
## FALSE  TRUE
## 24175 49748
```

```
# Assign the mode to missing values
host_identity_verified_mode <- "TRUE"
na_vector <- is.na(airbnb$host_identity_verified)
airbnb$host_identity_verified[na_vector] <- host_identity_verified_mode

# Display the table of host_has_profile_pic levels
print(table(airbnb$host_has_profile_pic))
```

```
##
## FALSE  TRUE
##   226 73697
```

```
# Assign the mode to missing values
host_profile_pictures_mode <- "TRUE"
na_vector <- is.na(airbnb$host_has_profile_pic)
airbnb$host_has_profile_pic[na_vector] <- host_profile_pictures_mode

# Last_review
print(table(airbnb$last_review))
```

```
##
## 2015 and earlier             2016             2017
##              655             2938            54691
```

```
last_review_mode = "2017"
na_vector = is.na(airbnb$last_review)
airbnb$last_review[na_vector] = last_review_mode
```

Minimal number of NA compared to the entries number, no further checks are needed.

Percentage variables For `review_scores_rating` and `host_response_rate` median imputation is used. By using the median, we aim to minimize the potential influence of outliers on the imputed values. Both this variables include more than 20% of missing values. Therefore, it should be explored carefully.

```r
# review_scores_rating
cor(airbnb$log_price, airbnb$review_scores_rating,
    use = "pairwise.complete.obs")
```

```
## [1] 0.09121864
```

```r
airbnb <- airbnb %>%
  mutate(review_scores_rating = if_else(is.na(review_scores_rating),
                                        median(review_scores_rating, na.rm = TRUE),
                                       review_scores_rating))
cor(airbnb$log_price, airbnb$review_scores_rating)
```

```
## [1] 0.08417997
```

```r
# host_response_rate
cor(airbnb$log_price, airbnb$host_response_rate,
    use = "pairwise.complete.obs")
```

```
## [1] -0.006776928
```

```r
airbnb <- airbnb %>%
  mutate(host_response_rate = if_else(is.na(host_response_rate),
                                        median(host_response_rate, na.rm = TRUE),
                                        host_response_rate))
cor(airbnb$log_price, airbnb$host_response_rate)
```

```
## [1] 0.001423241
```

It appears that the correlation doesn't changes too much with the NA imputation.

Numeric Variables For `host_since` the median imputation is implemented to address missing values. This variables includes more than 20% of missing values. Therefore, it should be explored carefully.

```r
# host since
cor(airbnb$log_price, airbnb$host_since,
    use = "pairwise.complete.obs")
```

```
## [1] 0.07685562
```

```r
airbnb <- airbnb %>%
  mutate(host_since = if_else(is.na(host_since),
                              median(host_since, na.rm = TRUE), host_since))

cor(airbnb$log_price, airbnb$host_since,
    use = "pairwise.complete.obs")
```

```
## [1] 0.07677723
```

For `bathrooms`, `bedrooms` and `beds` the median imputation is implemented without additional exploration because missing values are minimal.

```r
# bathrooms
airbnb <- airbnb %>%
  mutate(bathrooms = if_else(is.na(bathrooms),
                             median(bathrooms, na.rm = TRUE), bathrooms))
# bedrooms
airbnb <- airbnb %>%
  mutate(bedrooms = if_else(is.na(bedrooms),
                             median(bedrooms, na.rm = TRUE), bedrooms))
# beds
airbnb <- airbnb %>%
  mutate(beds = if_else(is.na(beds),
                        median(beds, na.rm = TRUE), beds))
```

Exploratory Data Analysis

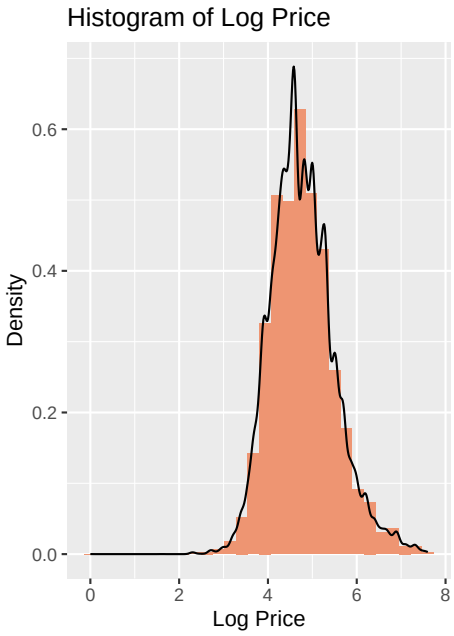
In this section, we will conduct an exploratory data analysis (EDA) on the Airbnb's dataset after the preprocessing phase. EDA is a crucial step in understanding the dataset and uncovering meaningful insights. By employing various data visualization techniques and descriptive statistics, we aim to identify significant patterns and trends in the data.

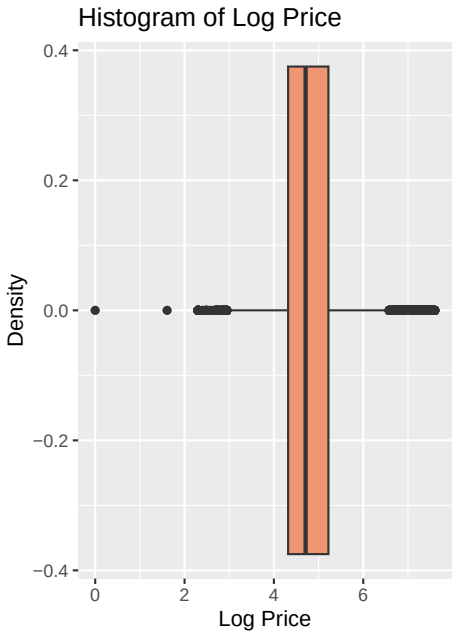
Response analysis

Log-price is our response, it is a numerical variable that represents the log value of the price of the airbnb's location. First we can check if our response is normally distributed, in order to be able to choose the right modeling technique.

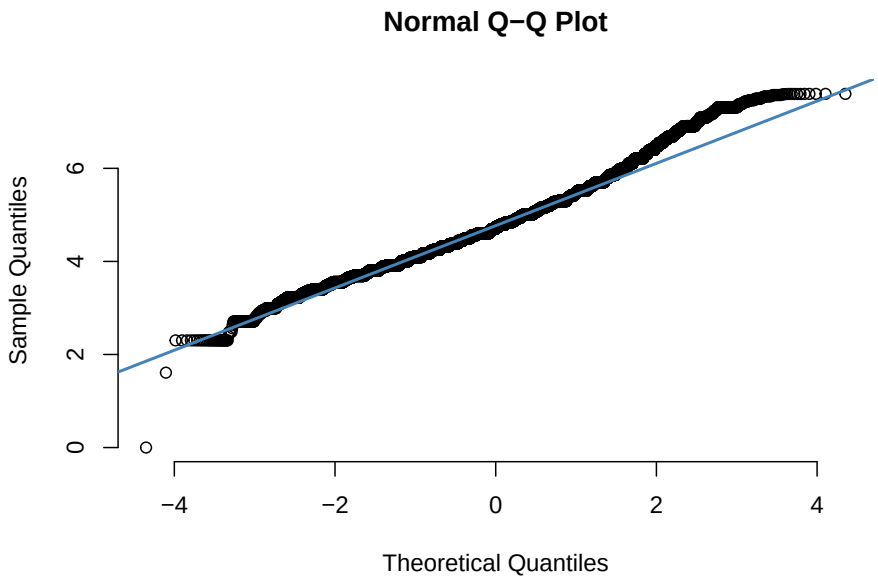
```r
hist_response = ggplot(airbnb, aes(x=log_price, y = ..density..)) +
  geom_histogram(fill = "#ee9572") +
  geom_density() +
  ylab("Density") +
  xlab("Log Price") +
  ggtitle("Histogram of Log Price")
boxplot_response = ggplot(airbnb, aes(x=log_price)) +
  geom_boxplot(fill = "#ee9572") +
  ylab("Density") +
  xlab("Log Price") +
  ggtitle("Histogram of Log Price")

grid.arrange(hist_response, boxplot_response, nrow = 1)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```






```
qqnorm(airbnb$log_price, pch = 1, frame = FALSE)
qqline(airbnb$log_price, col = "steelblue", lwd = 2)
```



Although there is a slight departure from the normal distribution in our data, it is characterized by negative skewness. However does not appear to be significantly extreme, and we can reasonably assume that our data is sufficiently similar to a normal distribution.

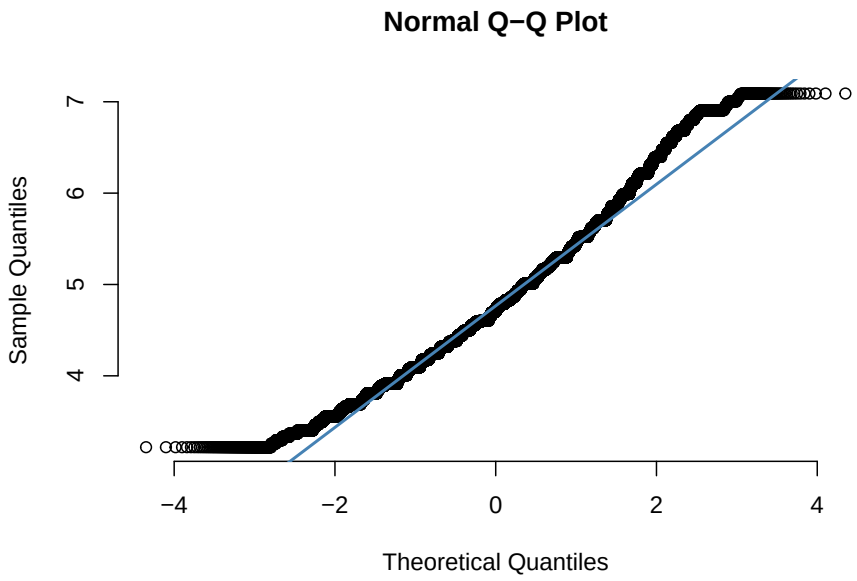
Outliers in response

```
outliers=boxplot.stats(airbnb$log_price)$out
length(outliers)/length(airbnb$log_price)
```

```
## [1] 0.0206717
```

By retaining the outliers, which account for approximately 2% of the total number of responses, we observe a distribution that deviates significantly from the normal distribution. Due to this substantial deviation, we have decided to retain the outliers in our analysis for the time being.

```
# trimming only values that are more extreme then 1 and 99 percenteline instead of  2.5 and 97.5
lower_bound <- quantile(airbnb$log_price, 0.005)
upper_bound <- quantile(airbnb$log_price, 0.995)
outlier_ind <- which(airbnb$log_price < lower_bound | airbnb$log_price > upper_bound)
airbnb_temp <- airbnb[- outlier_ind,]

qqnorm(airbnb_temp$log_price, pch = 1, frame = FALSE)
qqline(airbnb_temp$log_price, col = "steelblue", lwd = 2)
```



Upon observing the response boxplot, it is evident that there are two outliers that deviate significantly from the mean in comparison to the others. However, one of these outliers has a value of 0, which doesn't make sense in the context of the house prices.

```
# Calculate the mean of the variable "log_price"
mean_log_price <- mean(airbnb$log_price)
# Calculate the absolute difference between each value and the mean
diff_from_mean <- abs(airbnb$log_price - mean_log_price)
# Order the differences in descending order and select the first two
most_extreme_indices <- order(diff_from_mean, decreasing = TRUE)[1:2]
```

```r
# Get the most extreme values
most_extreme_values <- airbnb$log_price[most_extreme_indices]
# Print the most extreme values
print(most_extreme_values)
```

```
## [1] 0.000000 1.609438
```

```r
# Remove the most extreme values from the dataframe
airbnb <- airbnb[-most_extreme_indices, ]
```

Covariates analysis

Let's now analyze with some plots, all the covariates in the dataset and their relationship with the response `log_price`

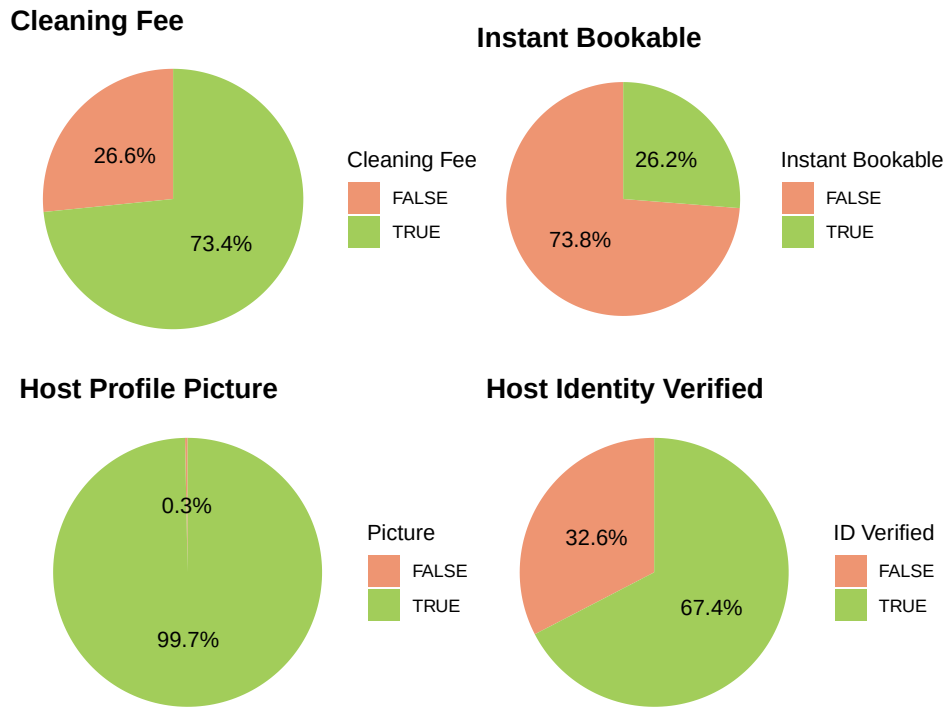
Binary variables

The variable `host_identity_verified` was explored previously along with `host_has_profile_pic`, among the booleans, we have to explore two more features, namely `cleaning_fee` and `instant_bookable`.

The pie charts and the boxplots illustrate the distribution and the dependence of the price on the category respectively.

```r
my_colors <- c("#ee9572", "#a2cd5a")

# pie chart for cleaning_fee
fee <- ggplot(airbnb, aes(x = "", fill = cleaning_fee)) +
    geom_bar(stat = "count", position = "fill") +
    coord_polar(theta = "y") +
    labs(fill = "Cleaning Fee", x = NULL, y = NULL,
       title = "Cleaning Fee") +
    theme_void() +
    theme(plot.title = element_text(size = 14, face = "bold")) +
    geom_text(aes(label = paste0(round(after_stat(count) / sum(after_stat(count)) * 100, 1), "%")),
            stat = "count", position = position_fill(vjust = 0.5))+
    scale_fill_manual(values = my_colors)

# pie chart for host_identity verified
book <- ggplot(airbnb, aes(x = "", fill = instant_bookable)) +
    geom_bar( stat = "count", position = "fill") +
    coord_polar(theta = "y") +
    labs(fill = "Instant Bookable", x = NULL, y = NULL,
        title = "Instant Bookable") +
    theme_void()+
    theme(plot.title = element_text(size = 14, face = "bold")) +
    geom_text(aes(label = paste0(round(after_stat(count) / sum(after_stat(count)) * 100, 1), "%")),
            stat = "count", position = position_fill(vjust = 0.5))+
    scale_fill_manual(values = my_colors)
# pie chart for host_has_profile_pic
pic <- ggplot(airbnb, aes(x = "", fill = host_has_profile_pic)) +
    geom_bar(stat = "count", position = "fill") +
```

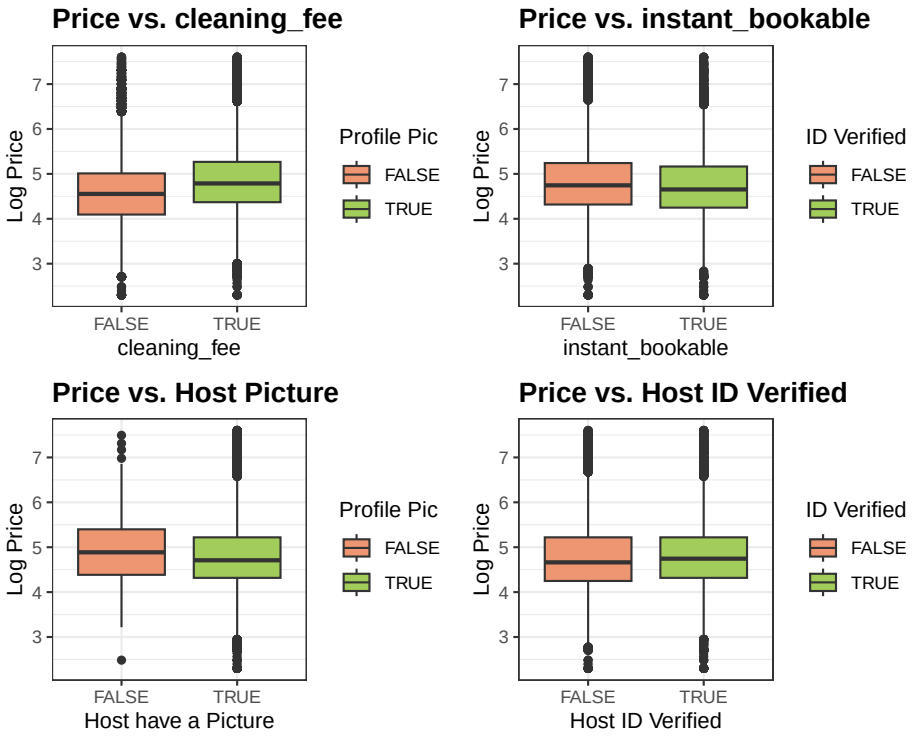
```
    coord_polar(theta = "y") +
    labs(fill = "Picture", x = NULL, y = NULL,
       title = "Host Profile Picture") +
    theme_void() +
    theme(plot.title = element_text(size = 14, face = "bold")) +
    geom_text(aes(label = paste0(round(after_stat(count) / sum(after_stat(count)) * 100, 1), "%")),
            stat = "count", position = position_fill(vjust = 0.5))+
    scale_fill_manual(values = my_colors)

# pie chart for host_identity verified
iden <- ggplot(airbnb, aes(x = "", fill = host_identity_verified)) +
    geom_bar( stat = "count", position = "fill") +
    coord_polar(theta = "y") +
    labs(fill = "ID Verified", x = NULL, y = NULL,
        title = "Host Identity Verified") +
    theme_void()+
    theme(plot.title = element_text(size = 14, face = "bold")) +
    geom_text(aes(label = paste0(round(after_stat(count) / sum(after_stat(count)) * 100, 1), "%")),
            stat = "count", position = position_fill(vjust = 0.5))+
    scale_fill_manual(values = my_colors)

grid.arrange(fee, book,pic,iden, nrow = 2)
```



Host profile picture exhibits a severe class imbalance, with 99.7% of the responses belonging to TRUE. The FALSE is notably underrepresented, comprising only a small fraction of the data.

```
# boxplot for cleaning_fee
fee_boxplot <- ggplot(airbnb, aes(x = cleaning_fee,
```

```
                    y = log_price, fill = cleaning_fee))+
    geom_boxplot()+
    ggtitle("Price vs. cleaning_fee")+
    scale_fill_manual(values = my_colors) +
    theme_bw() +
    theme(plot.title = element_text(size = 14, face = "bold")) +
    labs(fill = "Profile Pic", y = "Log Price", x = "cleaning_fee")

# boxplot for host_identity verified
book_boxplot <- ggplot(airbnb, aes(x = instant_bookable,
                    y = log_price, fill = instant_bookable))+
    geom_boxplot()+
    ggtitle("Price vs. instant_bookable")+
    scale_fill_manual(values = my_colors) +
    theme_bw() +
    theme(plot.title = element_text(size = 14, face = "bold")) +
    labs(fill = "ID Verified", y = "Log Price", x = "instant_bookable")

# boxplot for host_has_profile_pic
pic_price <- ggplot(airbnb, aes(x = host_has_profile_pic,
                    y = log_price, fill = host_has_profile_pic))+
    geom_boxplot()+
    ggtitle("Price vs. Host Picture")+
    scale_fill_manual(values = my_colors) +
    theme_bw() +
    theme(plot.title = element_text(size = 14, face = "bold")) +
    labs(fill = "Profile Pic", y = "Log Price", x = "Host have a Picture")

# boxplot for host_identity verified
iden_price <- ggplot(airbnb, aes(x = host_identity_verified,
                    y = log_price, fill = host_identity_verified))+
    geom_boxplot()+
    ggtitle("Price vs. Host ID Verified")+
    scale_fill_manual(values = my_colors) +
    theme_bw() +
    theme(plot.title = element_text(size = 14, face = "bold")) +
    labs(fill = "ID Verified", y = "Log Price", x = "Host ID Verified")

grid.arrange(fee_boxplot, book_boxplot,pic_price, iden_price ,nrow = 2)
```



Taking into consideration the plots above, we conclude that the variable `host_has_profile_pic` is not informative, and at the same time there is not a significant difference in price between hosts with pictures and without. Thus, this feature is dropped from the dataset.

```
airbnb <- airbnb %>% dplyr::select(-host_has_profile_pic)
```

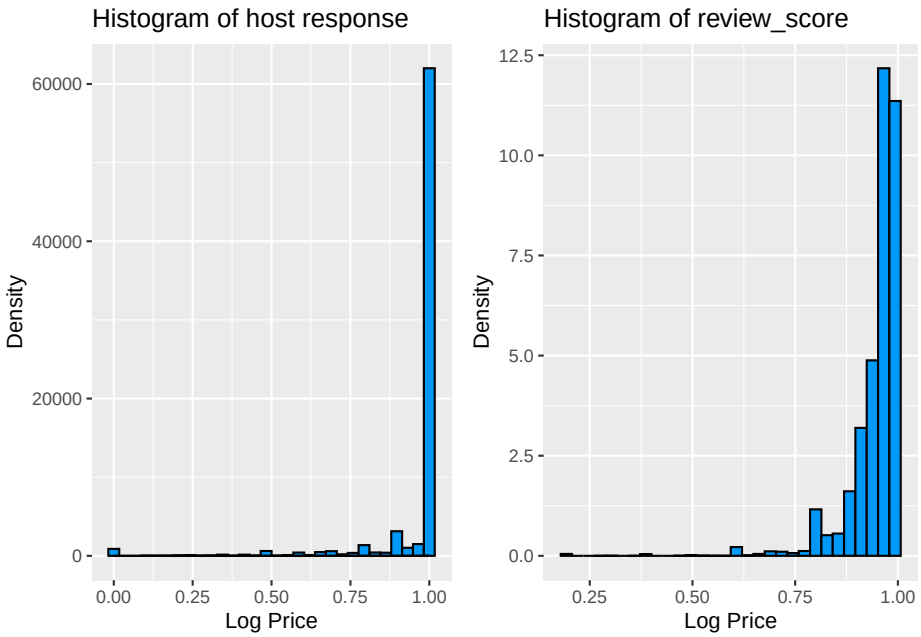
```
table(airbnb$host_response_rate)
```

Percentage variables: `host_resonse_rate` and `review_score_rating`

```
##
##     0  0.06   0.1  0.11  0.13  0.14  0.15  0.17   0.2  0.21  0.22  0.23  0.25
##   883     1    16     2     2     6     1    15    45     1     3     1    80
##  0.26  0.27  0.29   0.3  0.31  0.33  0.35  0.36  0.38  0.39   0.4  0.41  0.42
##     3     2    18    38     1   142     5     5    15     1   120     2     5
##  0.43  0.44  0.46  0.47   0.5  0.52  0.53  0.54  0.55  0.56  0.57  0.58  0.59
##    22    19     8     2   611     5    14    21    13    44    57    17     4
##   0.6  0.61  0.62  0.63  0.64  0.65  0.66  0.67  0.68  0.69   0.7  0.71  0.72
##   337     4     7    58    23    15     3   433    71    12   508   106    15
##  0.73  0.74  0.75  0.76  0.77  0.78  0.79   0.8  0.81  0.82  0.83  0.84  0.85
##    49    24   315    29    24   116    45  1113    88    82   279    62    62
##  0.86  0.87  0.88  0.89   0.9  0.91  0.92  0.93  0.94  0.95  0.96  0.97  0.98
##   243    90   316   310  2277   224   315   307   401   322   350   400   425
##  0.99     1
##   448 61551
```

```
host_response = ggplot(airbnb, aes(x=host_response_rate)) +
  geom_histogram( color = "#000000", fill = "#0099F8") +
  geom_density() +
  ylab("Density") +
  xlab("Log Price") +
  ggtitle("Histogram of host response")

review = ggplot(airbnb, aes(x=review_scores_rating)) +
  geom_histogram(aes(y = ..density..), color = "#000000", fill = "#0099F8") +
  ylab("Density") +
  xlab("Log Price") +
  ggtitle("Histogram of review_score")

grid.arrange(host_response,review, ncol=2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We calculate the percentage of the host response rate which are lower than 95%. And we also take into consideration the correlation between `log_price` and `host_response_rate`.

```
high_rate <- nrow(filter(airbnb, host_response_rate > 0.95))
na_rate <- sum(is.na(airbnb$host_response_rate))

# percentage of low rate responses
1 - (high_rate + na_rate)/nrow(airbnb)
```

```
## [1] 0.1475529
```

```r
# correlation between log_price and host_response_rate
cor(airbnb$log_price, airbnb$host_response_rate,
    use = "pairwise.complete.obs")
```

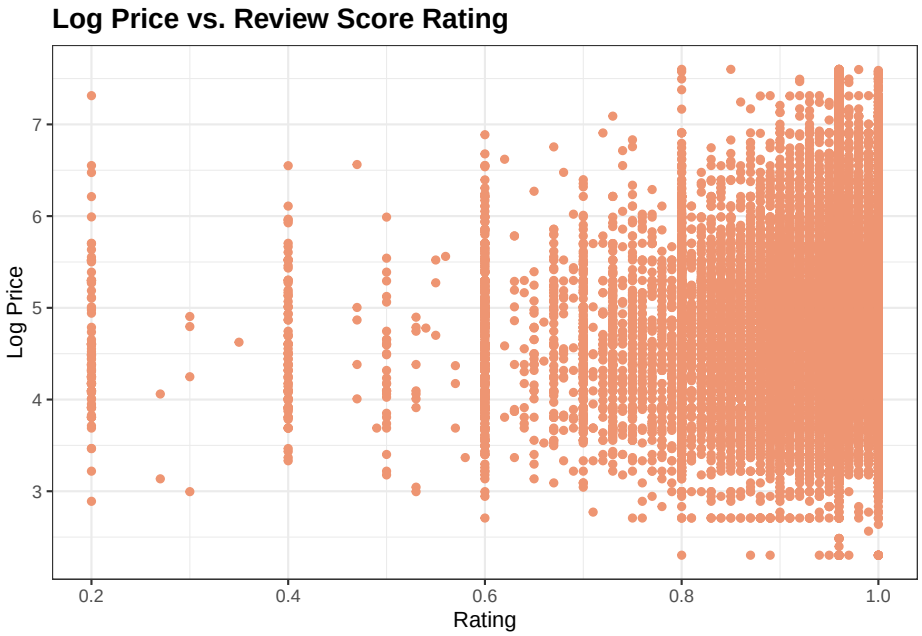
```
## [1] 0.001468106
```

Taking into account that there are less than 15% of the hosts who respond slowly in the dataset, and the correlation between `log_price` and `host_response_rate` is close to zero, we decide to discard the variable `host_response_rate`.

```r
airbnb <- airbnb %>% dplyr::select(-host_response_rate)
```

The following scatterplots provide more information about the variable `review_score_rating` and its relationship with `log_price`

```r
rating_price <- ggplot(airbnb, aes(x = review_scores_rating, y = log_price)) +
  geom_point(color = "#ee9572") +
  theme_bw() +
  theme(plot.title = element_text(size = 14, face = "bold")) +
  labs(x = "Rating", y = "Log Price") +
  ggtitle("Log Price vs. Review Score Rating")

grid.arrange(rating_price, nrow = 1)
```



The scatterplots indicate a slight increase in the log_price as the review_score_rating increases, suggesting a positive correlation between these two variables.

Factors variables

Neighbourhood and City Variables The variable `neighbourhood` can not be transformed to a numeric one. Moreover, it is connected with the variable `city` because each neighborhood belongs to a specific city.

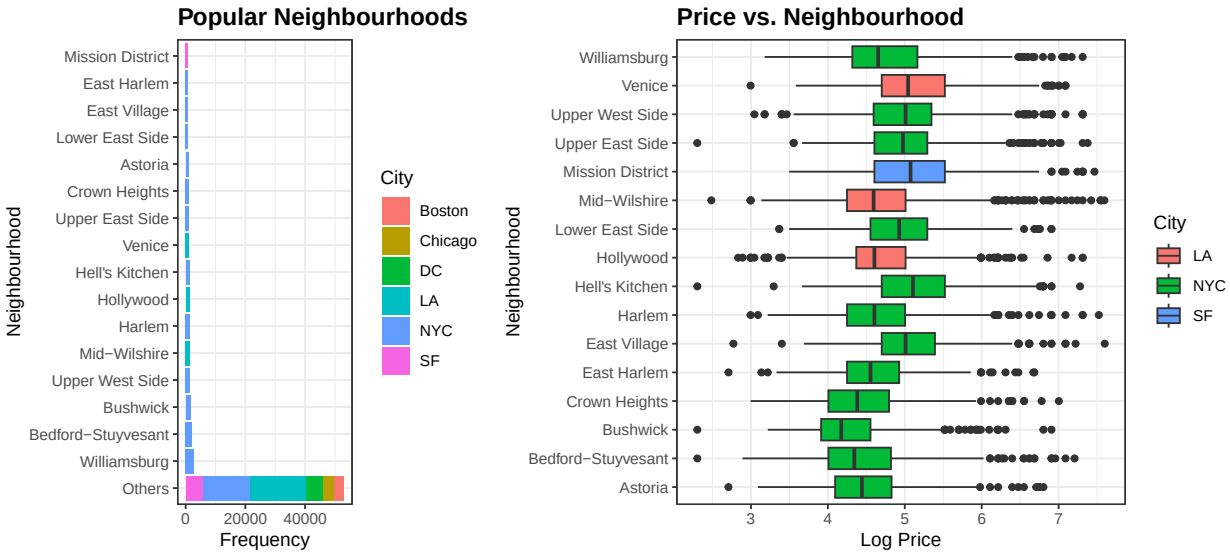
We have 619 categories that are to be explored in details.

We consider only those neighborhoods that contain at least 1% of the data. Others (along with missing values) are replaced with a category "Others".

```
# find the most frequent neighbourhood
neighbourhood_count = as.data.frame(table(airbnb$neighbourhood))
colnames(neighbourhood_count) <- c("name", "frequency")
neighbourhood_1 <- neighbourhood_count %>%
  filter(frequency > 0.01*nrow(airbnb))

# replace others with "Others"
airbnb <- airbnb %>%
  mutate(neighbourhood = if_else(neighbourhood %in% neighbourhood_1$name,
                         neighbourhood, "Others"))
```

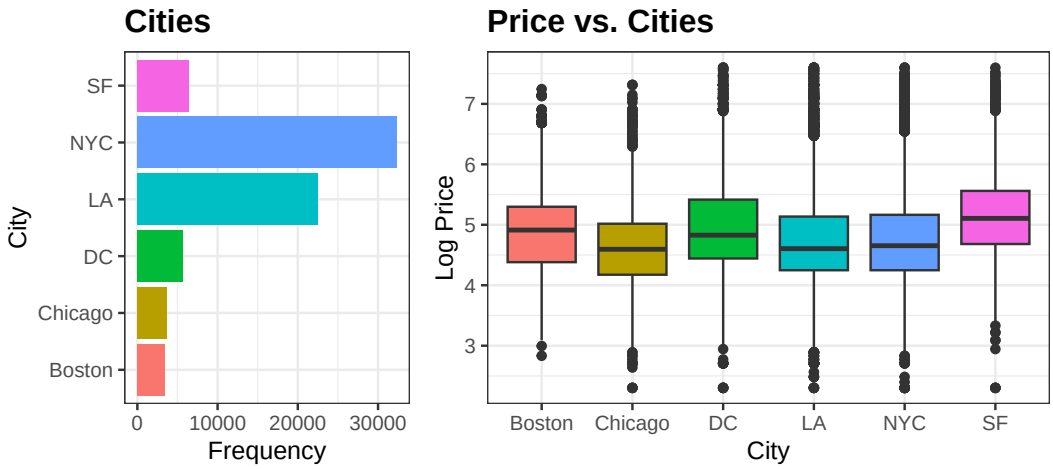
The following plots illustrate the distribution of `neighbourhood` and its dependence on `log_price` along with the information about a city.

```
# histogram
neigh_hist <- ggplot(airbnb,
                     aes(x = reorder(neighbourhood,
                                     -table(neighbourhood)[neighbourhood]),
                                     fill = city)) +
  geom_bar() +
  labs(fill="City", x = "Neighbourhood", y = "Frequency") +
  coord_flip()+
  theme_bw() +
  theme(plot.title = element_text(size = 14, face = "bold")) +
  ggtitle("Popular Neighbourhoods")

# relationship between different categories and price
neigh_price <- ggplot(filter(airbnb, neighbourhood != "Others"),
                      aes(x = neighbourhood, y = log_price, fill = city))+
    geom_boxplot()+
    ggtitle("Price vs. Neighbourhood")+
    coord_flip()+
    theme_bw() +
    theme(plot.title = element_text(size = 14, face = "bold")) +
    labs(fill="City", y = "Log Price", x = "Neighbourhood")

lay <- rbind(c(1,1,2,2,2),
             c(1,1,2,2,2))

grid.arrange(neigh_hist, neigh_price, layout_matrix = lay)
```



The following plots show the distribution of `city` and its dependence on `log_price`.

```
# histogram
city_hist <- ggplot(airbnb, aes(x = city, fill = city)) +
  geom_bar() +
  labs(fill="City", x = "City", y = "Frequency") +
  coord_flip()+
  theme_bw() +
  theme(plot.title = element_text(size = 14, face = "bold")) +
  ggtitle("Cities")+
  guides(fill = FALSE)

# relationship between different categories and price
city_price <- ggplot(airbnb, aes(x = city,
                                 y = log_price, fill = city))+
    geom_boxplot()+
    ggtitle("Price vs. Cities")+
    theme_bw() +
    theme(plot.title = element_text(size = 14, face = "bold")) +
    labs(fill="City", y = "Log Price", x = "City")+
    guides(fill = FALSE)

lay <- rbind(c(1,1,2,2,2),
             c(1,1,2,2,2))

grid.arrange(city_hist, city_price, layout_matrix = lay)
```



To assess the relevance of keeping the information about neighbourhoods, we estimate what percentage of all data the most popular neighbourhood makes up.

```
# the most popular neighbourhood
filter(neighbourhood_count, frequency == max(frequency))
```

```
##           name frequency
## 1 Williamsburg      2862
```

```
# its percentage
pop <- filter(neighbourhood_count, frequency == max(frequency))$frequency
pop/nrow(airbnb)
```

```
## [1] 0.03861879
```

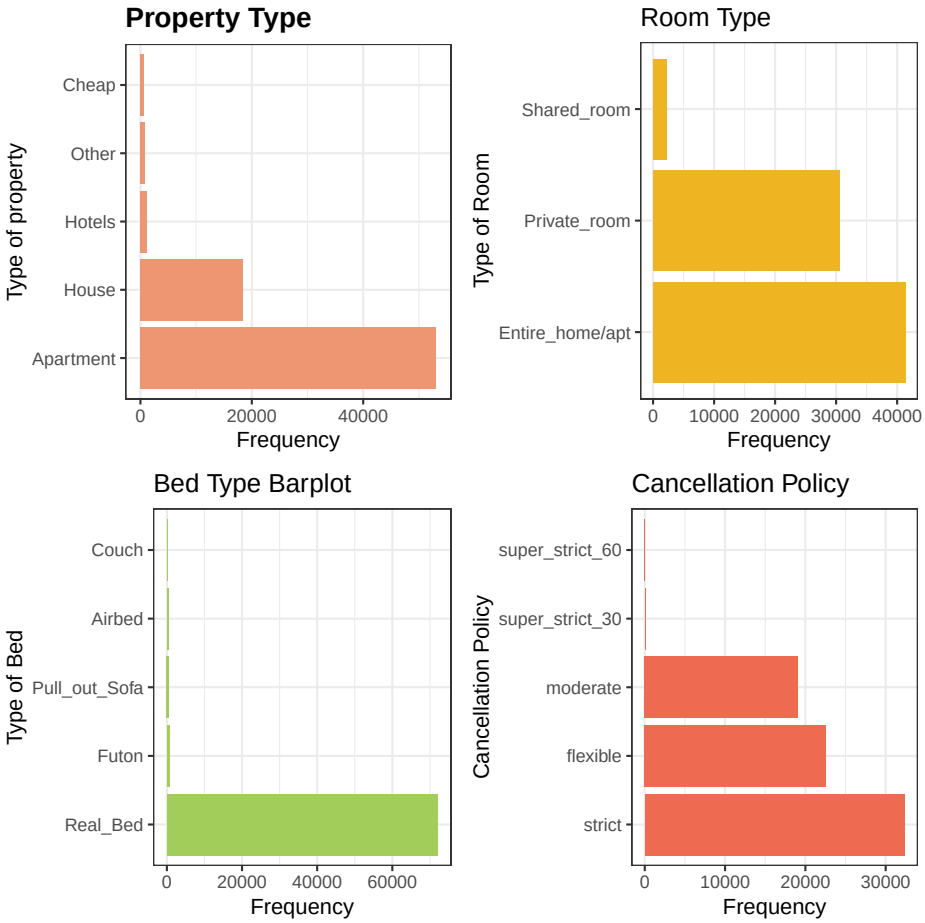
Since this value is less than 4%, we delete the variable `neighbourhood`, keeping the more general information about the cities.

```
airbnb <- airbnb %>% dplyr::select(-neighbourhood)
```

The next step is to explore the features `property_type`, `room_type`, `bed_type`, `cancellation_policy`. The boxplots and the histograms illustrate the information about the characteristics as usual.

```
# histogram for property_type
prop <- ggplot(airbnb, aes(x = reorder(property_type, -table(property_type)[property_type]))) +
  geom_bar(fill="#ee9572") +
  labs(x = "Type of property", y = "Frequency") +
  coord_flip()+
  theme_bw() +
  theme(plot.title = element_text(size = 14, face = "bold")) +
  ggtitle("Property Type")

# histogram for room_type
room <- ggplot(airbnb, aes(x =reorder(room_type, -table(room_type)[room_type]), fill = room_type)) +
  geom_bar(fill="#EEB422") +
```

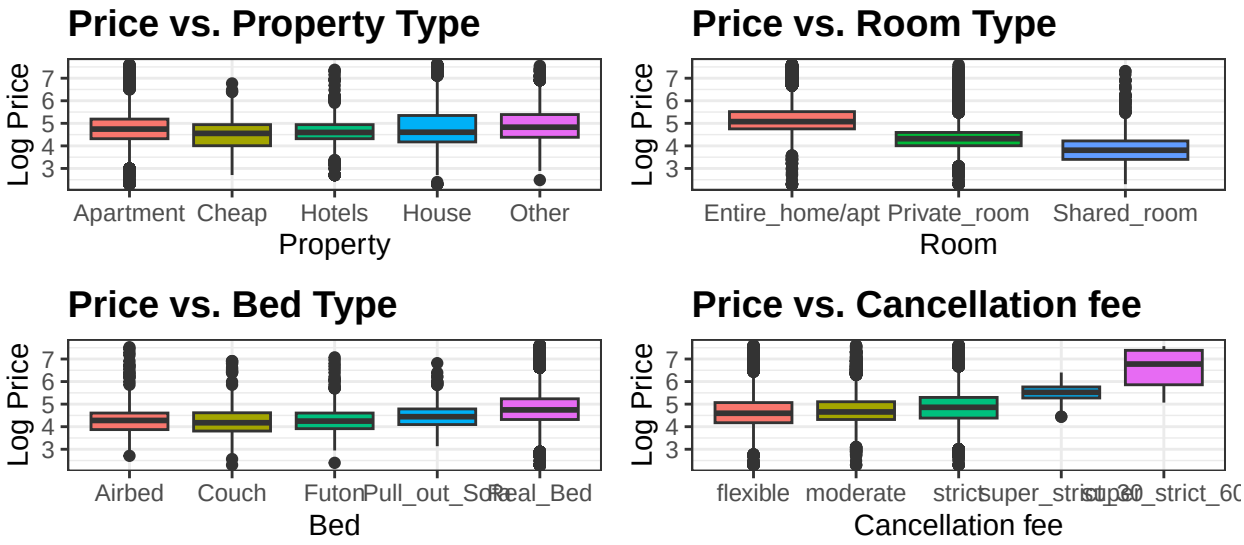
```
  labs(x = "Type of Room", y = "Frequency") +
  coord_flip()+
  theme(axis.text.x = element_text(angle = 180, vjust = 0.5),
        plot.title = element_text(size = 14, face = "bold")) +
  theme_bw() +
  ggtitle("Room Type")

# histogram for bed_type
bed <- ggplot(airbnb, aes(x = reorder(bed_type, -table(bed_type)[bed_type]), fill = bed_type)) +
  geom_bar(fill="#a2cd5a") +
  labs(x = "Type of Bed", y = "Frequency") +
  coord_flip()+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5),
          plot.title = element_text(size = 14, face = "bold")) +
  theme_bw() +
  ggtitle("Bed Type Barplot")

# histogram for cancellation_policy
canc <- ggplot(airbnb, aes(x = reorder(cancellation_policy,
                                       -table(cancellation_policy)[cancellation_policy]),
                           fill = cancellation_policy)) +
  geom_bar(fill="#EE6A50") +
  labs(x = "Cancellation Policy", y = "Frequency") +
  coord_flip()+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5),
          plot.title = element_text(size = 14, face = "bold")) +
  theme_bw() +
  ggtitle("Cancellation Policy")


# arrange these graphs
grid.arrange(prop, room, bed, canc, nrow=2)
```



Property type, room type, bed type and cancellation policy boxplot realted to the response:

```
# relationship between different categories and price
prop_price <- ggplot(airbnb, aes(x = property_type,
                                 y = log_price, fill = property_type))+
    geom_boxplot()+
    ggtitle("Price vs. Property Type")+
    theme_bw() +
    theme(plot.title = element_text(size = 14, face = "bold")) +
    labs(fill="Property", y = "Log Price", x = "Property")+
    guides(fill = FALSE)
# relationship between different categories and price
room_price <- ggplot(airbnb, aes(x = room_type,
                                 y = log_price, fill = room_type))+
    geom_boxplot()+
    ggtitle("Price vs. Room Type")+
    theme_bw() +
    theme(plot.title = element_text(size = 14, face = "bold")) +
    labs(fill="Property", y = "Log Price", x = "Room")+
    guides(fill = FALSE)
# relationship between different categories and price
bed_price <- ggplot(airbnb, aes(x = bed_type,
    y = log_price, fill = bed_type))+
```

```
    geom_boxplot()+
    ggtitle("Price vs. Bed Type")+
    theme_bw() +
    theme(plot.title = element_text(size = 14, face = "bold")) +
    labs(fill="Property", y = "Log Price", x = "Bed")+
    guides(fill = FALSE)
# relationship between different categories and price
canc_price <- ggplot(airbnb, aes(x = cancellation_policy,
    y = log_price, fill = cancellation_policy))+
    geom_boxplot()+
    ggtitle("Price vs. Cancellation fee")+
    theme_bw() +
    theme(plot.title = element_text(size = 14, face = "bold")) +
    labs(fill="Property", y = "Log Price", x = "Cancellation fee")+
    guides(fill = FALSE)

grid.arrange(prop_price, room_price,bed_price,canc_price,nrow=2)
```

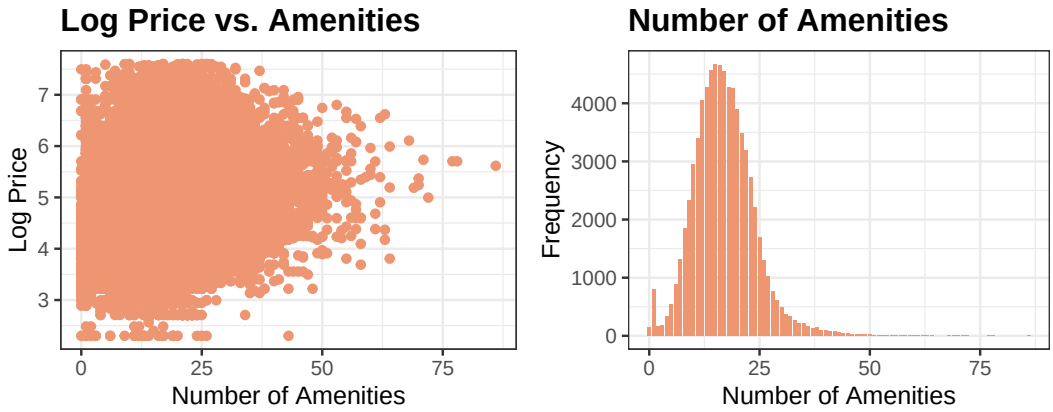


We decided to transform `Cancellation fee` in a numerical variable and put all the three strict levels in one:

```
# cancellation policy ca be transformed in numerical form
airbnb <- airbnb %>%
  mutate(cancellation_policy = case_when(
    cancellation_policy == "super_strict_30" | cancellation_policy == "super_strict_60"|
      cancellation_policy == "strict"  ~ 1,
      cancellation_policy == "moderate" ~ 2,
      cancellation_policy == "flexible" ~ 3
  ))
```

Now we look at the amenities count versus response

```r
amen_price <- ggplot(airbnb, aes(x = amenities_count, y = log_price)) +
 geom_point(color = "#ee9572") +
 theme_bw() +
 theme(plot.title = element_text(size = 14, face = "bold")) +
 labs(x = "Number of Amenities", y = "Log Price") +
 ggtitle("Log Price vs. Amenities")
amen_hist <- ggplot(airbnb, aes(x = amenities_count)) +
 geom_bar(fill="#ee9572") +
 labs(x = "Number of Amenities", y = "Frequency") +
 theme_bw() +
 theme(plot.title = element_text(size = 14, face = "bold")) +
 ggtitle("Number of Amenities")
grid.arrange(amen_price, amen_hist, nrow = 1)
```



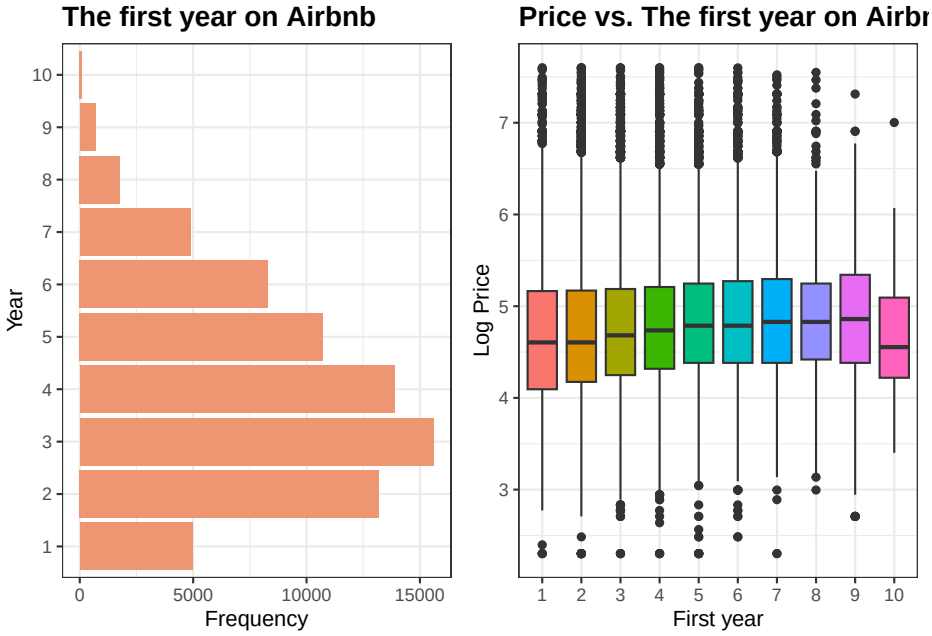
By looking the relationship between the response the amenities count it appears the response doesn't be affected with some not random patterns by the amenities count.

Date variables

We explore the distribution of years for the variables under investigation.

```r
# host since
host_since_hs <- ggplot(airbnb, aes(x = as.factor(host_since))) +
  geom_bar(fill="#ee9572") +
  labs(x = "Year", y = "Frequency") +
  coord_flip()+
  theme_bw() +
  theme(plot.title = element_text(size = 14, face = "bold")) +
  ggtitle("The first year on Airbnb")
# host since
host_since_hs_b <- ggplot(airbnb, aes(x = as.factor(host_since),
                    y = log_price, fill = as.factor(host_since)))+
    geom_boxplot()+
    ggtitle("Price vs. The first year on Airbnb")+
    theme_bw() +
    theme(plot.title = element_text(size = 14, face = "bold")) +
    labs(fill = "Year", y = "Log Price", x = "First year")+
    guides(fill = FALSE)
```

```
grid.arrange(host_since_hs, host_since_hs_b, nrow = 1)
```



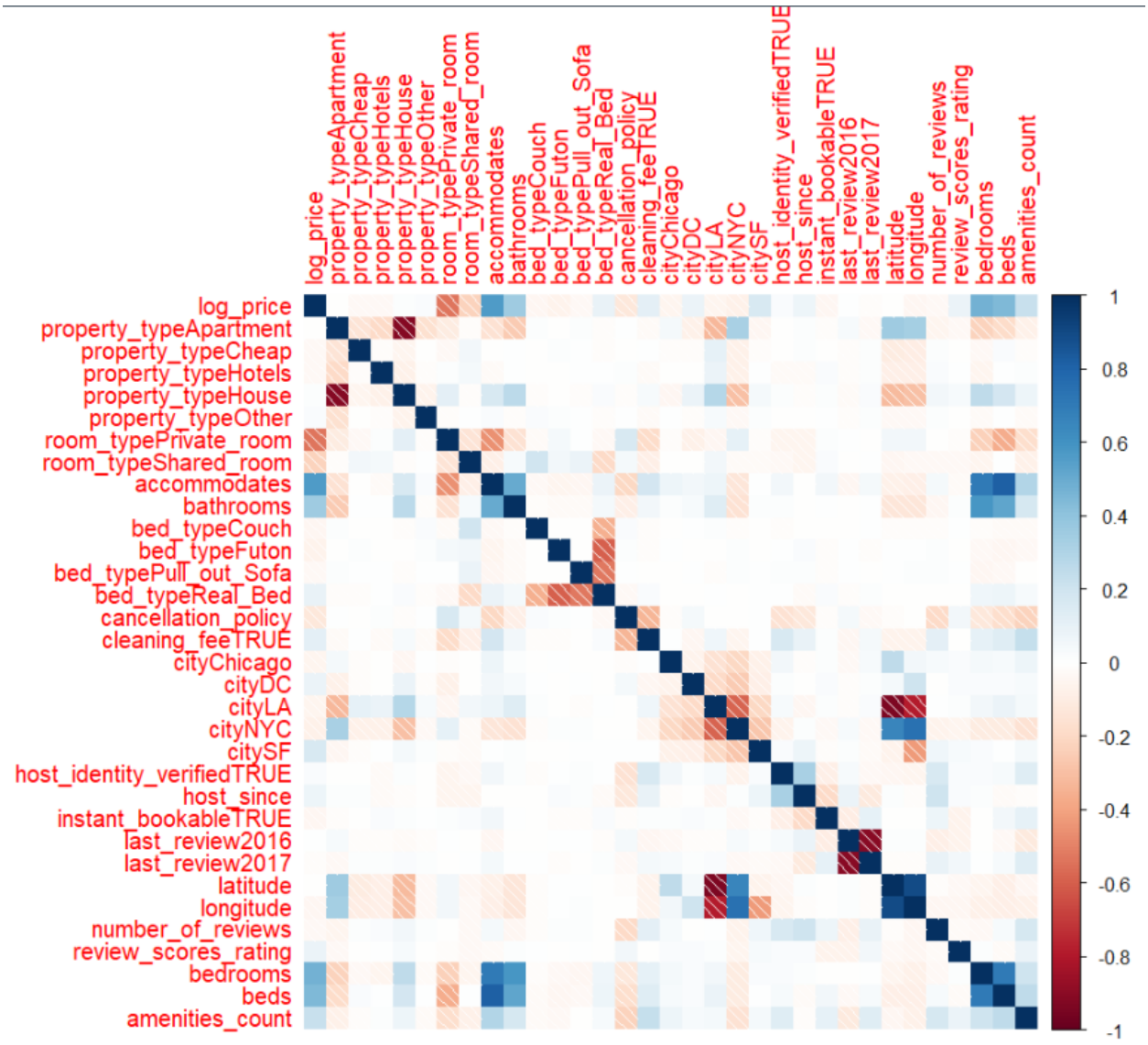
The boxplots illustrate the dependence of the price on the year. We can not observe a strong correlation from the plots below.

Despite the fact that the correlation in every case is small, we keep the variables `host_since`. Later, the significance of these variables is investigated.

Multicollinearity adress

Collinearity refers to the presence of strong linear relationships between independent variables in a regression model, which can have significant implications for models accuracy, stability, and interpretability. By leveraging cor.matrix in R, we can discern the extent of interrelationships between different covariates data types like categorical, continuous, binary covariates.

```
airbnb_temp <- airbnb %>%
  dplyr::select(-c("amenities"))

model.matrix(~0+., data=airbnb_temp) %>%
  cor(use="pairwise.complete.obs") %>%
  corrplot(method = "shade")
```



We removed highly correlated variables to avoid multicollinearity problems in our analysis, like beds that are logically correlated with the number of accommodates and number of bedrooms. So we removed beds and bedrooms.

```
# The highly correlated variables are removed in both airbnb and cor_airbnb
airbnb <- airbnb %>%
  dplyr::select(-c("beds", "bedrooms"))
```

Despite observing a high negative correlation between longitude and latitude due to the fact that the airbnb are concentrated in cities, we decided to retain both variables for potential modeling purposes.

Standardize numerical covariates

```
# Select numerical columns to standardize (excluding the response variable)
numerical_cols <- c("accommodates", "bathrooms", "host_since",
```

```
                    "latitude", "longitude", "number_of_reviews", "review_scores_rating",
                    "amenities_count")

# Standardize numerical columns
standardized_data <- airbnb
standardized_data[numerical_cols] <- scale(standardized_data[numerical_cols])


airbnb[numerical_cols] <- scale(airbnb[numerical_cols])
```

Linear Regression modeling

In this section, we present the methodology used to construct a statistical model and explore the factors that influence the response variable based on the dataset under investigation. In our analysis, we consider the response variable `log price` as the main focus of interest.

We want to estimate the relationship between the response variable and the covariates. We construct a statistical model that try to captures the dependencies and patterns observed in the data. We also discuss the significance of the covariates in explaining the variability in the response variable and interpret the estimated coefficients or effects.

In our analysis, we identify a set of covariates or independent variables that have the potential to exert an influence on the response variable. To ensure the robustness of our models and to avoid overfitting, we split our data into a training set and a testing set. The training set is used to build and tune our models, while the testing set serves as new, unseen data for evaluating the performance of these models

Variables not used for now:

```
airbnb <- airbnb %>% dplyr::select(-c("amenities", "longitude","latitude"))
```

Dataset splitting in training and test

```
# Split the data into training and testing sets (75% training, 25% testing)
index <- createDataPartition(airbnb$log_price, p = 0.75, list = FALSE)
train_set <- airbnb[index,]
test_set <- airbnb[-index,]

# Create the model matrix for glmnet
X_train <- model.matrix(log_price~.,data=train_set)
y_train <- train_set$log_price
X_test <- model.matrix(log_price~.,data=test_set)
y_test <- test_set$log_price
```

Multiple linear regression: model assumptions

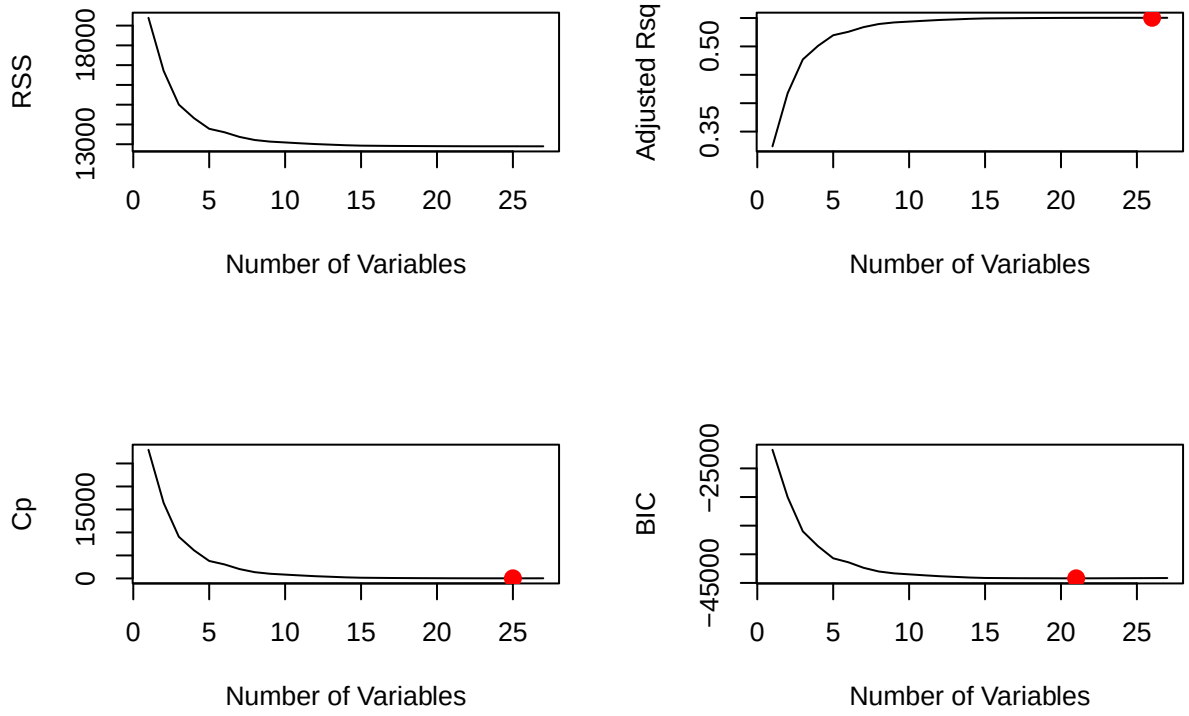
- Linear relationship: There exists a linear relationship between the predictors and the dependent variable, that is $Y = f(x) + \epsilon$, where $f(x) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$.
- Homoscedasticity: The error terms have constant variance, that is, $\text{Var}(\epsilon_i) = \sigma^2$ for all $i = 1, \ldots, n$.
- Independence of error terms: The error terms $\epsilon_1, \ldots, \epsilon_n$ are mutually independent.
- Normal distribution of error terms: $\epsilon_i \sim N(0, \sigma^2)$ for all $i = 1, \ldots, n$.

In this analysis, we will employ the Best Subset Selection method for variable selection in our regression model. This decision is motivated by the fact that our dataset has a manageable number of predictors, making the computational demand of this method feasible.

```
set.seed(1)
# Best Subset Selection on the training set
regfit.full <- regsubsets(log_price~., data=train_set, nvmax=27)
reg.summary <- summary(regfit.full)
length(reg.summary$rsq)
```

```
## [1] 27
```

```
par(mfrow=c(2,2))

# residual sum of squares
plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")

# adjusted-R^2 with its largest value
plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted Rsq",type="l")
best_value = which.max(reg.summary$adjr2)
points(best_value,reg.summary$adjr2[best_value], col="red",cex=2,pch=20)

# Mallow's Cp with its smallest value
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')
best_value=which.min(reg.summary$cp)
points(best_value,reg.summary$cp[best_value],col="red",cex=2,pch=20)

# BIC with its smallest value
plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
best_value=which.min(reg.summary$bic)
points(best_value,reg.summary$bic[best_value],col="red",cex=2,pch=20)
```



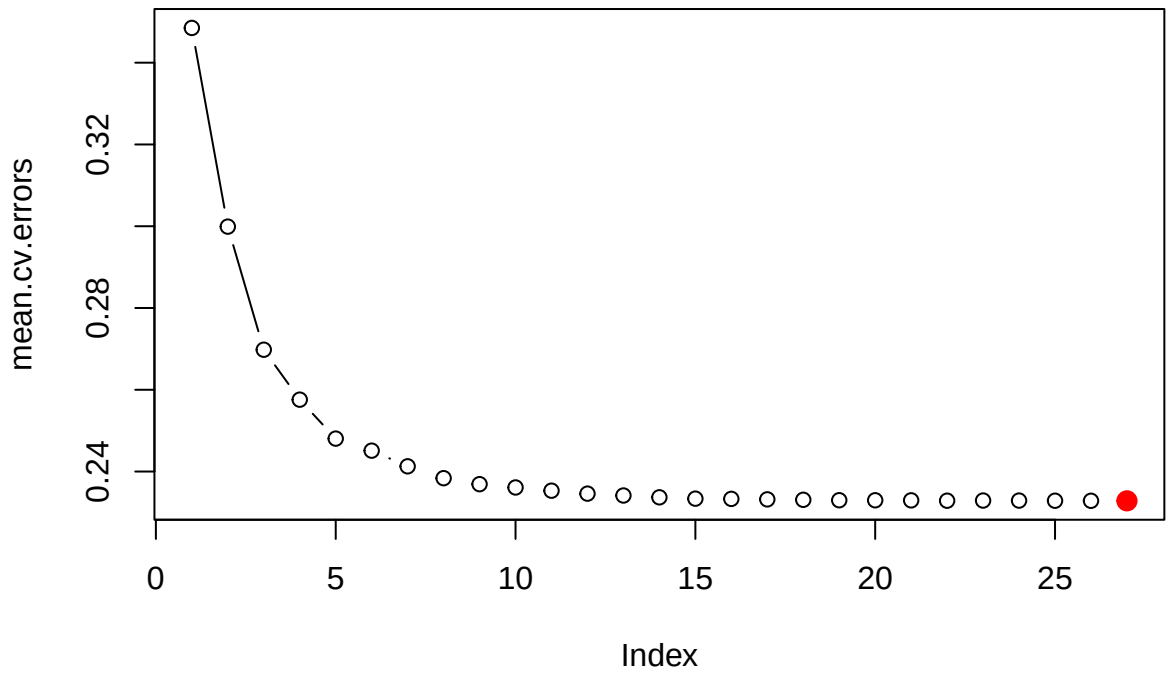

```
par(mfrow=c(1,1))
```

The model with 26 predictors maximizes the Adjusted R-squared. The model with 24 predictors minimizes Mallow's Cp. The model with 22 predictors minimizes the BIC.

Having analyzed the optimal number of predictors using Mallow's Cp, BIC, and adjusted R-squared, we will now proceed with a more comprehensive analysis using cross-validation. This method will provide us with a direct measure of our model's predictive performance.

```
# Cross-Validation
set.seed(1)
k = 10
folds <- sample(1:k, nrow(airbnb), replace=TRUE)
cv.errors <- matrix(NA, k, 27, dimnames=list(NULL, paste(1:27)))
for(j in 1:k){
  best.fit <- regsubsets(log_price~., data=airbnb[folds!=j,], nvmax=27)
  test.mat <- model.matrix(log_price~., data=airbnb[folds==j,])
  for(i in 1:27){
    coefi <- coef(best.fit, id=i)
    pred <- test.mat[,names(coefi)]%*%coefi
    cv.errors[j,i] <- mean((airbnb$log_price[folds==j]-pred)^2)
  }
}
mean.cv.errors <- apply(cv.errors, 2, mean)
```

```r
plot(mean.cv.errors, type='b')
best_value=which.min(mean.cv.errors)
mse_ml = min(mean.cv.errors)
points(best_value,mean.cv.errors[best_value], col="red",cex=2,pch=20)
```



Our cross-validation analysis revealed that the optimal number of predictors for our model is 27. However, it appears that there is no significant improvement in the error rate from 15 to 27 predictors. Even though we have a large dataset, we should still be cautious about model complexity. The relatively small improvement in the cross-validation error suggests that the additional complexity of the 27-predictor model may not be justified. To perform a better and more precise model selection we will try with ridge and lasso techniques

```r
# Convert the model matrix back to a data frame
train_model_matrix <- as.data.frame(X_train)
test_model_matrix <- as.data.frame(X_test)
# Add the response variable back to the data frame
train_model_matrix$log_price <- train_set$log_price
test_model_matrix$log_price <- test_set$log_price


reg.best <- regsubsets(log_price~.,data=train_set, nvmax=best_value)
best_predictors = coef(reg.best,best_value)
predictors = names(best_predictors)[-1]


# Create the formula as a string
```

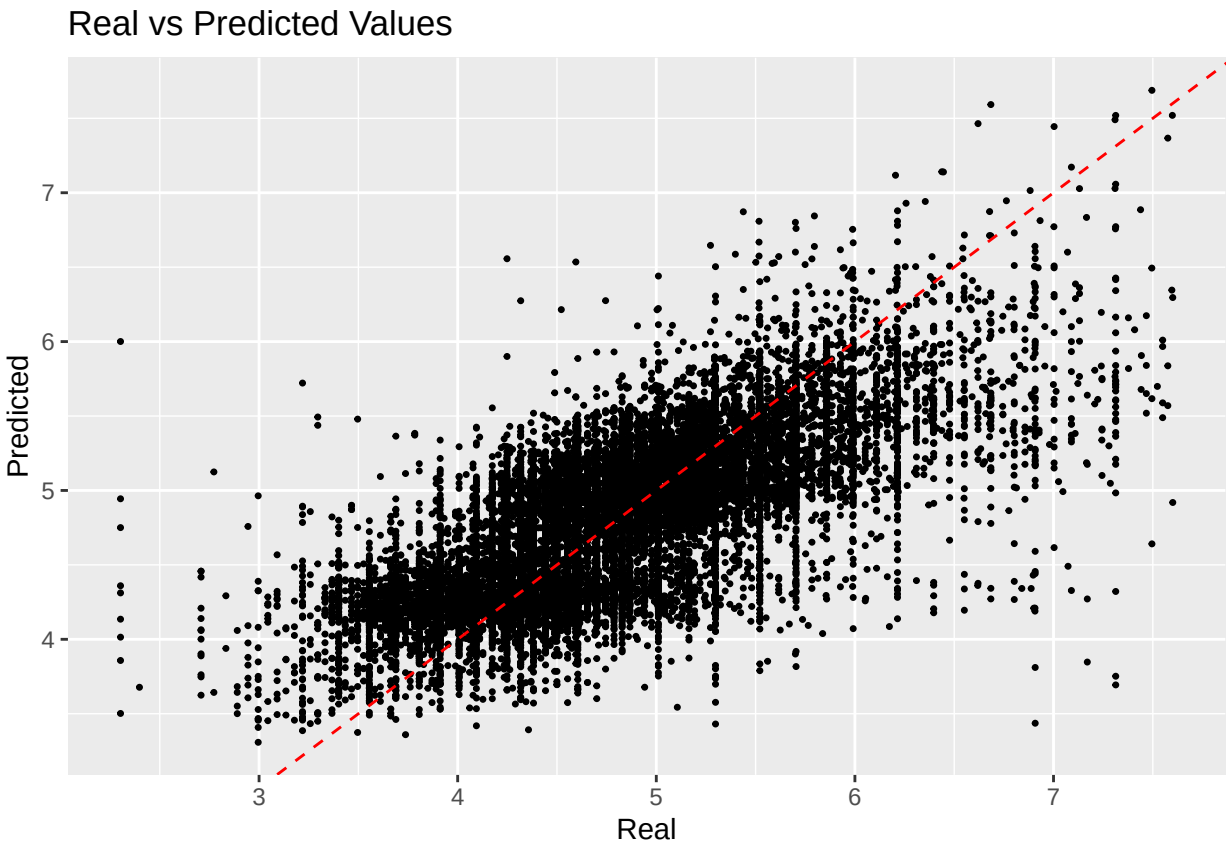
```r
formula_str <- paste("log_price ~", paste(predictors, collapse = " + "))
# Convert the string to a formula
formula_obj <- as.formula(formula_str)
# Fit the linear model
lm_model <- lm(formula_obj, data = train_model_matrix)
```

```r
reg.best <- regsubsets(log_price~.,data=train_set, nvmax=10)
best_predictors = coef(reg.best,10)
predictors = names(best_predictors)[-1]


# Create the formula as a string
formula_str <- paste("log_price ~", paste(predictors, collapse = " + "))
# Convert the string to a formula
formula_obj <- as.formula(formula_str)
# Fit the linear model
lm_model2 <- lm(formula_obj, data = train_model_matrix)
```

Indeed, the increase in the adjusted R-squared value from the first model (0.5395) to the second model (0.546) is quite small, despite the addition of 17 more predictor variables in the second model. This suggests that these additional variables are not contributing much to the explanation of the variance in the dependent variable.

Simpler models are often preferred. The first model, with fewer predictors, is simpler and easier to interpret than the second model. In our case we prefer to keep the more complex model.

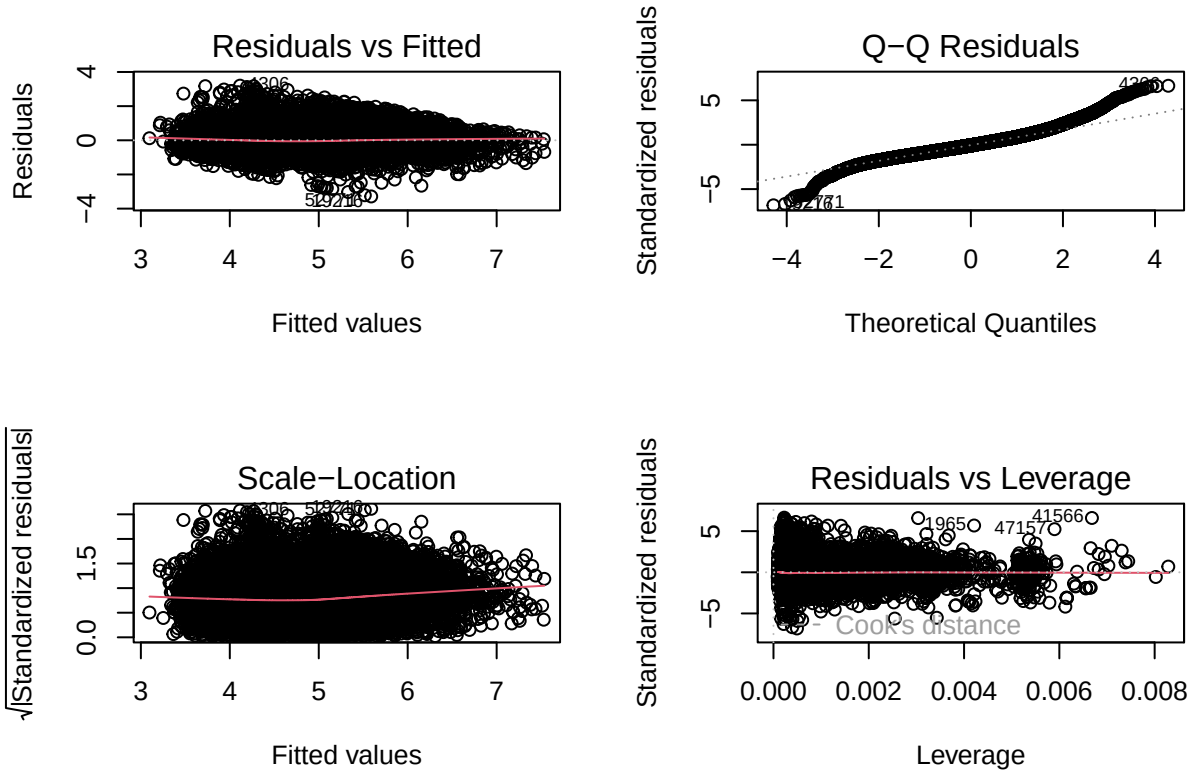
```r
predictions <- data.frame(Real = test_model_matrix$log_price,
                          Predicted = predict(lm_model, newdata = test_model_matrix))

# Calculate r squared and MSE of the MLM
rsquared_ml <- summary(lm_model)$r.squared
mse_ml <- mean((predictions$Real - predictions$Predicted)^2)

# Create a data frame to store the model results
model_results <- data.frame(
  Model = "Multiple Linear Regression",
  R_squared = rsquared_ml,
  MSE = mse_ml
)

# Create the plot
ggplot(predictions, aes(x = Real, y = Predicted)) +
  geom_point(size = 1, alpha = 1, shape = 20) +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
  labs(x = "Real", y = "Predicted") +
  ggtitle("Real vs Predicted Values")
```



The plot suggests a positive correlation between the relative and predictive values, as the line of best fit slopes upwards. However, the spread of the points around the line indicates some degree of prediction error as r-squared of 0.54 suggest. The outlier suggests that there may be some instances where the predictive value significantly underestimates the relative price.

Some coefficient interpretation are: - `room_typePrivate_room` is -0.596257. This suggests that, all else being equal, the log price is expected to decrease by about 0.596 for private rooms compared to the Entire_home/apt. - The coefficient for `accommodates` is 0.214186, suggesting that for each additional accommodation, the log price is expected to increase by about 0.214 units, assuming all other variables are held constant. - The coefficients for `cityChicago`, `cityLA`, and `citySF` are -0.289787, -0.145927, and 0.315473 respectively. This means that, all else being equal: Listings in Chicago are expected to have a log price that is about 0.29 units lower than listings in NYC. Listings in Los Angeles (LA) are expected to have a log price that is about 0.15 units lower than listings in NYC. Listings in San Francisco (SF) are expected to have a log price that is about 0.32 units higher than listings in NYC.

```
# Set the layout for the plot
par(mfrow = c(2, 2))
# Plot the model results
plot(lm_model)
```



A good residual plot has points randomly scattered around the horizontal axis. If the points in a residual plot are randomly dispersed around the horizontal axis, a linear regression model is appropriate for the data; otherwise, a non-linear model is more appropriate. In this plot, the points do not seem to be randomly scattered around the horizontal axis. Instead, they appear to follow a certain pattern. This suggests that the relationship between the variables is not purely linear.

The scale-location plot evidence a heteroscedasticity situation with the red line that bend towards low end instead of stay straight. Given that the response variable is already transformed, and further transformation is not preferred, here are some alternative solutions are consider using Weighted or Generalized Least Squares to give more weight to observations with smaller errors, or compute robust standard errors for more reliable hypothesis testing, another possibility can be switch to a different modeling approach because linear regression homoscedasticity assumptions isn't met.

Residuals are approximately normally distributed. But there is a clear sign of heavy tails, this could suggest that the residuals have more extreme values than would be expected under a normal distribution.

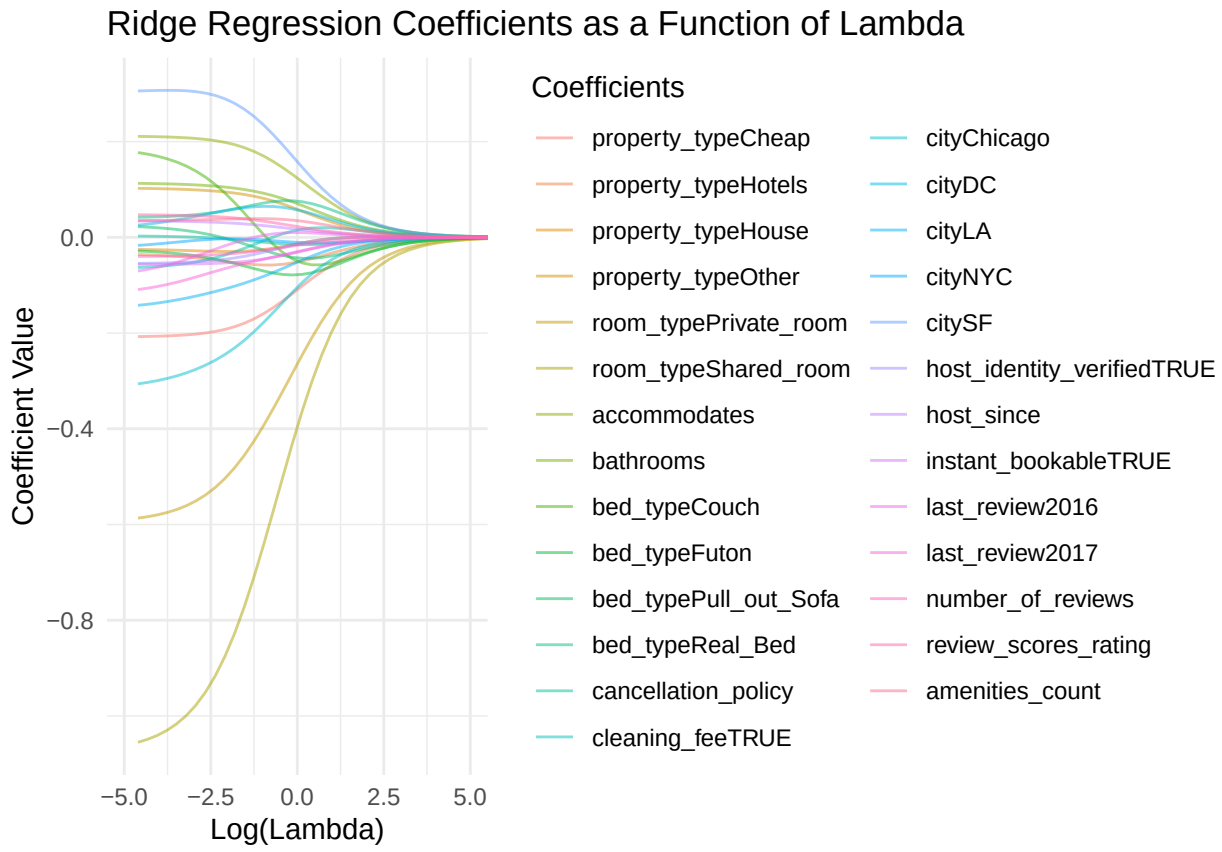
Ridge and Lasso Regression models

We will now try ridge and lasso regression to prevent overfitting and improve the performance of my linear regression model.

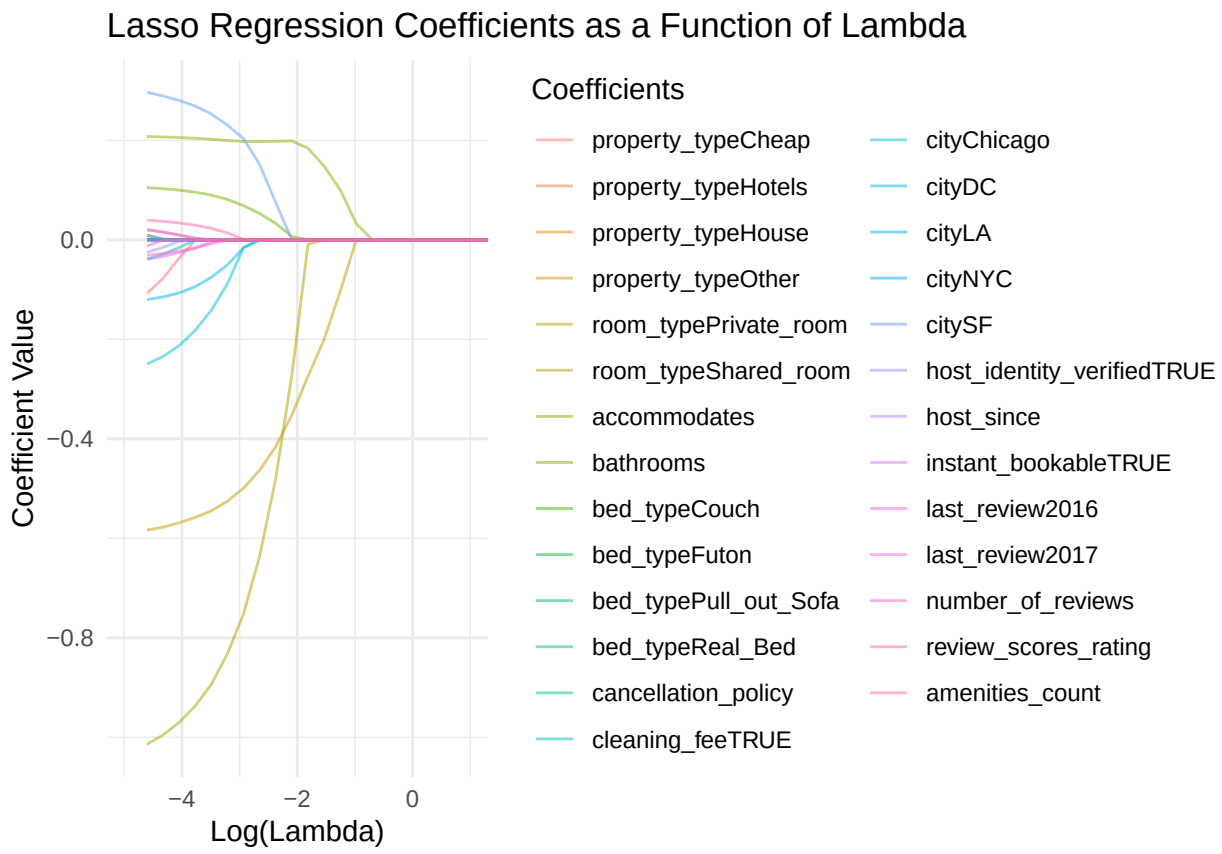
On the training dataset we perform cross validation to choose the best lambda value to use.

```
# First remove the intercept by X_train, because gmlnt add the intercept too
X_train_glmnt = X_train[,-1]
X_test_glmnt = X_test[,-1]
```

```r
# Ridge Regression
cv.out.ridge <- cv.glmnet(X_train_glmnt, y_train, alpha = 0, nfold=10)
bestlam.ridge <- cv.out.ridge$lambda.min
ridge.pred <- predict(cv.out.ridge, s = bestlam.ridge, newx = X_test_glmnt)
mse_ridge = mean((ridge.pred - y_test)^2)
rsquared_ridge <- 1 - sum((y_test - ridge.pred)^2) / sum((y_test - mean(y_test))^2)
ridge_model <- glmnet(X_train_glmnt, y_train, alpha=0,
                      lambda=10^seq(10, -2, length=100), thresh = 1e-12)

# Lasso Regression
cv.out.lasso <- cv.glmnet(X_train_glmnt, y_train, alpha = 1, nfold=10)
bestlam.lasso <- cv.out.lasso$lambda.min
lasso.pred <- predict(cv.out.lasso, s = bestlam.lasso, newx = X_test_glmnt)
mse_lasso = mean((lasso.pred - y_test)^2)
rsquared_lasso <- 1 - sum((y_test - lasso.pred)^2) / sum((y_test - mean(y_test))^2)
lasso_model <- glmnet(X_train_glmnt, y_train, alpha=1,
                      lambda=10^seq(10, -2, length=100), thresh = 1e-12)

# Update model_results data frame
model_results <- rbind(
  model_results,
  data.frame(
    Model = "Ridge Regression",
    R_squared = rsquared_ridge,
    MSE = mse_ridge
  ),
  data.frame(
    Model = "Lasso Regression",
    R_squared = rsquared_lasso,
    MSE = mse_lasso
  )
)

# Coefficients and Plot
df.ridge <- data.frame(t(as.matrix(coef(ridge_model))[-1,]))
df.ridge$lambda <- log(ridge_model$lambda)
df.lasso <- data.frame(t(as.matrix(coef(lasso_model))[-1,]))
df.lasso$lambda <- log(lasso_model$lambda)

df.ridge.melt <- melt(df.ridge, id.vars = "lambda",
                      variable.name = "coefficient", value.name = "value")
df.lasso.melt <- melt(df.lasso, id.vars = "lambda",
                      variable.name = "coefficient", value.name = "value")

ggplot() +
  geom_line(data = df.ridge.melt, aes(x = lambda, y = value, color = coefficient), alpha = 0.5) +
  scale_color_discrete(name = "Coefficients") +
  labs(x = "Log(Lambda)", y = "Coefficient Value") +
  ggtitle("Ridge Regression Coefficients as a Function of Lambda") +
  theme_minimal() +
   coord_cartesian(xlim = c(-5, 5))
```




```
ggplot() +
  geom_line(data = df.lasso.melt, aes(x = lambda, y = value, color = coefficient), alpha = 0.5) +
  scale_color_discrete(name = "Coefficients") +
  labs(x = "Log(Lambda)", y = "Coefficient Value") +
  ggtitle("Lasso Regression Coefficients as a Function of Lambda") +
  theme_minimal() +
   coord_cartesian(xlim = c(-5, 1))
```



We made a table to compare our three models by looking the R squared and the MSE

```
print(model_results)
```

```
##                         Model R_squared       MSE
## 1 Multiple Linear Regression 0.5505856 0.2346874
## 2           Ridge Regression 0.5379279 0.2348096
## 3           Lasso Regression 0.5381869 0.2346779
```

In our case it seems that the Ridge and Lasso regressions did not significantly improve the model's performance compared to the Multiple Linear Regression model.

Final comments

The results suggest that the predictors in our dataset are all contributing to the model's ability to explain the variance in the dependent variable. However, the performance of all three models - Multiple Linear Regression, Ridge Regression, and Lasso Regression - was quite similar, indicating that the regularization introduced by Ridge and Lasso did not significantly enhance the model's predictive power in this particular case.

Looking forward, to further improve the model's performance, it may be beneficial to explore more advanced machine learning models. Techniques such as Random Forests, Gradient Boosting, and Extreme Gradient Boosting (XGBoost) could potentially provide better predictive accuracy. These models are capable of capturing complex non-linear relationships and interactions between variables, which might be present in

our data and not captured by the linear models. However, these advanced models were not covered in the scope of this course.

In conclusion, while the linear models used in this study provided a decent baseline performance, there is potential for improvement. Future work could involve exploring more advanced machine learning models and techniques, to enhance the predictive power of the model.