Lets get some idea about the dataset, The reviews dataset consits of 2 columns and neighborhoods consistins of 2 columns. I found that the reviews column has reviews data and id also the number of rows is much more bigger than needed. Also I don't find this dataset that much important to do the analysis as most of of the data are already available in the listing data set. Where, the neighbourhoods consist of very few rows with that has the names of the location. And I am not using the names instead I will be using the longitude and latitude. So I will not be using these two dataset for my machine learning model. The main and the important dataset is the listing data set which consist of 75 columns.

```
     id             listing_url        scrape_id          last_scraped          source             name             description        neighborhood_overview
 Length:88044       Length:88044       Length:88044       Length:88044       Length:88044       Length:88044       Length:88044       Length:88044
 Class :character   Class :character   Class :character   Class :character   Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character
 picture_url          host_id            host_url           host_name          host_since         host_location       host_about        host_response_time
 Length:88044       Length:88044       Length:88044       Length:88044       Length:88044       Length:88044       Length:88044       Length:88044
 Class :character   Class :character   Class :character   Class :character   Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character
 host_response_rate host_acceptance_rate host_is_superhost  host_thumbnail_url host_picture_url   host_neighbourhood host_listings_count host_total_listings_count
 Length:88044       Length:88044         Length:88044       Length:88044       Length:88044       Length:88044       Length:88044        Length:88044
 Class :character   Class :character     Class :character   Class :character   Class :character   Class :character   Class :character    Class :character
 Mode  :character   Mode  :character     Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character    Mode  :character
 host_verifications host_has_profile_pic host_identity_verified neighbourhood      neighbourhood_cleansed neighbourhood_group_cleansed  latitude
 Length:88044       Length:88044         Length:88044           Length:88044       Length:88044           Length:88044                  Length:88044
 Class :character   Class :character     Class :character       Class :character   Class :character       Class :character              Class :character
 Mode  :character   Mode  :character     Mode  :character       Mode  :character   Mode  :character       Mode  :character              Mode  :character
  longitude         property_type       room_type          accommodates          bathrooms          bathrooms_text      bedrooms              beds
 Length:88044       Length:88044       Length:88044       Length:88044       Length:88044       Length:88044       Length:88044       Length:88044
 Class :character   Class :character   Class :character   Class :character   Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character
  amenities            price           minimum_nights     maximum_nights     minimum_minimum_nights maximum_minimum_nights minimum_maximum_nights maximum_maximum_nights
 Length:88044       Length:88044       Length:88044       Length:88044       Length:88044           Length:88044           Length:88044           Length:88044
 Class :character   Class :character   Class :character   Class :character   Class :character       Class :character       Class :character       Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character       Mode  :character       Mode  :character       Mode  :character
 minimum_nights_avg_ntm maximum_nights_avg_ntm calendar_updated   has_availability   availability_30    availability_60    availability_90    availability_365
 Length:88044           Length:88044           Length:88044       Length:88044       Length:88044       Length:88044       Length:88044       Length:88044
 Class :character       Class :character       Class :character   Class :character   Class :character   Class :character   Class :character   Class :character
 Mode  :character       Mode  :character       Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character
 calendar_last_scraped number_of_reviews  number_of_reviews_ltm number_of_reviews_l30d first_review       last_review        review_scores_rating review_scores_accuracy
 Length:88044          Length:88044       Length:88044          Length:88044           Length:88044       Length:88044       Length:88044         Length:88044
 Class :character      Class :character   Class :character      Class :character       Class :character   Class :character   Class :character     Class :character
 Mode  :character      Mode  :character   Mode  :character      Mode  :character       Mode  :character   Mode  :character   Mode  :character     Mode  :character
 review_scores_cleanliness review_scores_checkin review_scores_communication review_scores_location review_scores_value  license            instant_bookable
 Length:88044              Length:88044          Length:88044                Length:88044           Length:88044         Length:88044       Length:88044
 Class :character          Class :character      Class :character            Class :character       Class :character     Class :character   Class :character
 Mode  :character          Mode  :character      Mode  :character            Mode  :character       Mode  :character     Mode  :character   Mode  :character
 calculated_host_listings_count calculated_host_listings_count_entire_homes calculated_host_listings_count_private_rooms calculated_host_listings_count_shared_rooms
 Length:88044                   Length:88044                                Length:88044                                 Length:88044
 Class :character               Class :character                            Class :character                             Class :character
 Mode  :character               Mode  :character                            Mode  :character                             Mode  :character
 reviews_per_month
 Length:88044
```

What I understood from the data set and what I did in the beginning:

1. Id: It is the unique identifier of the property listed. I will not be using this column. Code:
```
> listing$id=NULL
```

2. Listing_url: The url of the listing page on Airbnb. I will not be using this column. Code:
```
> listing$Listing_url=NULL
```

3. Scrap_id: It is also the id of the time that when the data was retrived from the source. I will not be using this id as I don't find it as a useful information for now. Code: listing$scrape_id=NULL

4. Last_scraped: It is the date when the data was retrieved from the source. I will be using this column to derive for how long the host has been listed in Airbnb. Code: listing$last_scraped=as.Date(listing$last_scraped) #converting into date data type
Last_scraped and calender_last_scraped are almost same so I am droping this column.
   Code: listing$last_scraped=NULL

5. Source: It gives the information about the origin of data. I will not be using this column. Code: listing$source=NULL

6. Name: it is the name of the host who has listed in Airbnb. Here, I can find an useful information which is rating of the host. I will be using only that information from this column. Code: library(stringr)
   listing$host_rating=str_extract(listings$name, "★\\d+\\.\\d+") # Extracting rating form the column
   #remove the star sign: listing$host_rating=as.numeric(str_replace(listing$host_rating, "★", "")) After analyzing, I don't find it that much useful so I am droping the column.

   #Droping the name column: listing$name=NULL

7. Description: It gives the extra information about the hosted property, but similar information can be seen in other columns too, so I will not be using this column too.Code: listing$description=NULL

8. Neighbourhood_overview: it is also some kind of description about the periferi of the property. I are not using NLP so it is better to drop the column. Code: listing$neighborhood_overview=NULL

9. Picture_url: I will not be using any Urls as it is difficult to convert it into meaningful data in numeric form. Code: listing$picture_url=NULL

10. Host_id: It is the unique identifier for the host. I will not be using this column as well. Code: listing$host_id=NULL

11. Host url: It is the url of host profile on Airbnb. And I will not be using this. Code: listing$host_url=NULL

12. Host_name: it is the name of the host. I am not using NLP so it is better to drop the column. Code: listing$host_name=NULL

13. Host_since: It is the date about when the host has first registered in Airbnb. I will be using this column to get the days and years. Which will help me to find how long the host is registered in Airbnb. Code: listing$host_since=as.Date(listing$host_since) #converting into date data type

14. Host_location: It shows the location of the host. I will not be using this column as I will be using longitude and latitude to get some insites if needed. Code: listing$host_location=NULL

15. Host_about: It shows the short description about the host. I will not be using this column too. Code: listing$host_about=NULL

16. Host_response_time: it shows how fast the host respondes to the clienst s. Converting into ordered factor. Code: listing$host_response_time = factor(listing$host_response_time, levels = c("a few days or more", "within a day", "within a few hours", "within an hour"), ordered = TRUE)

    As we cannot use non numeric dataset to train the model I am going to convert this coiumn into numeric form.

```
> unique(listing$host_response_time)
[1] <NA>               within a few hours within a day       within an hour
[5] a few days or more
4 Levels: a few days or more < within a day < ... < within an hour
> listing$host_response_time_numeric=as.numeric(listing$host_response_time)
> 
```

After conversion lets remove the existing column:

Figure: Before removing NA

```
> listing$host_response_time_numeric=as.numeric(listing$host_response_time)
> listing$host_response_time=NULL
> listing$host_response_time_numeric[is.na(listing$host_response_time_numeric)]=mean(listing$host_response_time_numeric,na.rm = TRUE)
> 
```

listing$host_response_time_numeric=as.numeric(listing$host_response_time)

listing$host_response_time=NULL

listing$host_response_time_numeric[is.na(listing$host_response_time_numeric)]=mean (listing$host_response_time_numeric,na.rm = TRUE)

hist(listing$host_response_time_numeric)

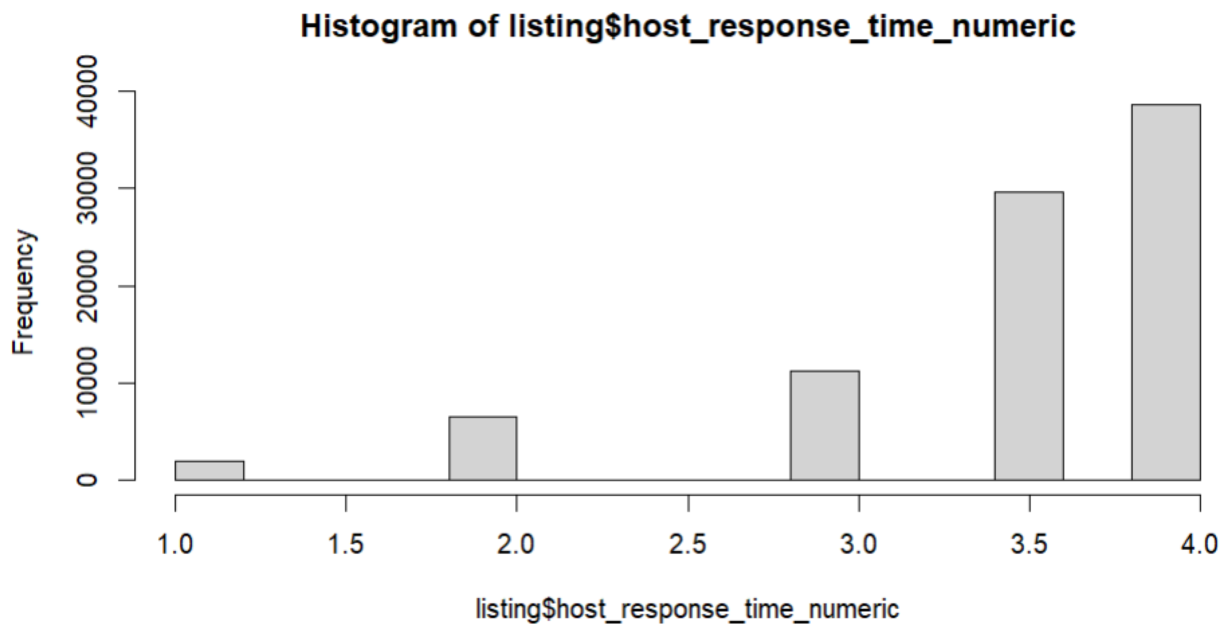


Figure: After replacing Na's

I found this column not much useful so droping this column

listing$host_response_time_numeric=NULLthe percentage sign first before converting to numeric value.

Code:
listing$host_response_rate=gsub("%","",listing$host_response_rate)

listing$host_response_rate=as.numeric(listing$host_response_rate)

(It will show a warning messgae: Nas introduced by coercion because, there are some non numeric values which will be replaced by Nas, and I will do the cleanings of all such data before model generation.)

Visualization:

hist(listing$host_response_rate)

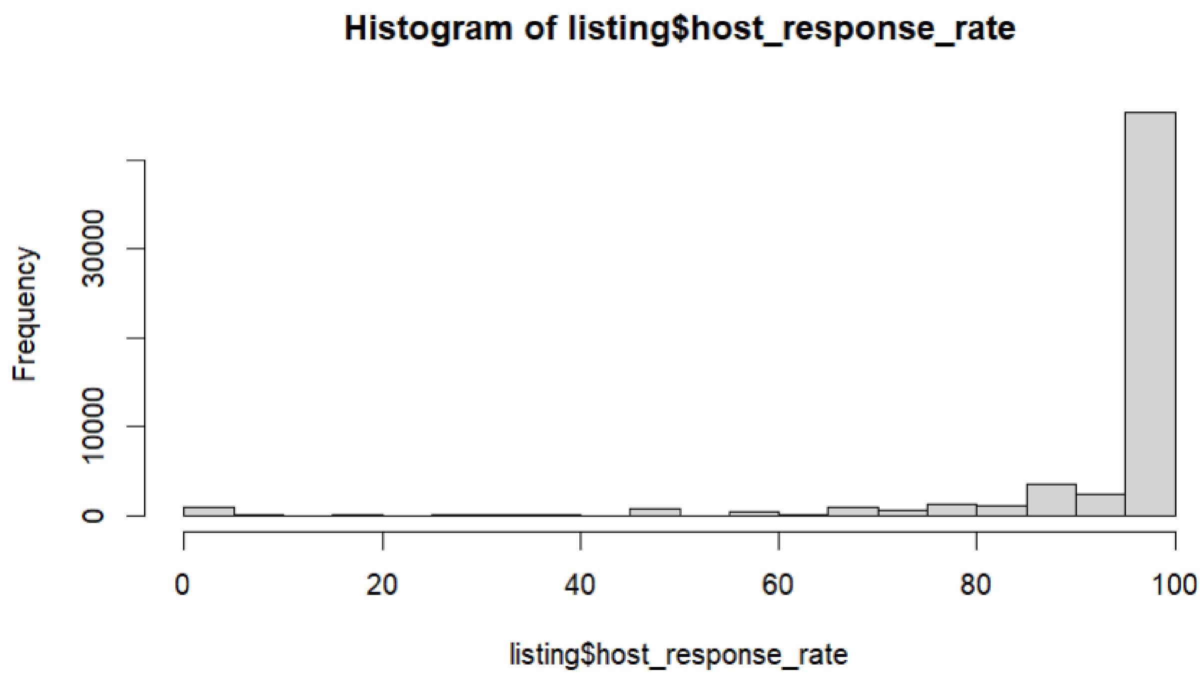


Fig: before filling NA

mean_host_response_rate=mean(listing$host_response_rate,na.rm = TRUE)

listing$host_response_rate[is.na(listing$host_response_rate)]=mean_host_response_ra
te

Figure: After Filling up Na’s

17. Host_acceptance_rate: It shows the percentage of booking requests that the host accepts. I will be using this column to derive the dependent variable. Doing the same thing as point 17.

Code: listing$host_acceptance_rate=gsub("%","",listing$host_acceptance_rate)
listing$host_acceptance_rate=as.numeric(listing$host_acceptance_rate)

```
> sum(is.na(listing$host_acceptance_rate))
[1] 25994
> sum(listing$host_acceptance_rate=="")
[1] NA
>
```

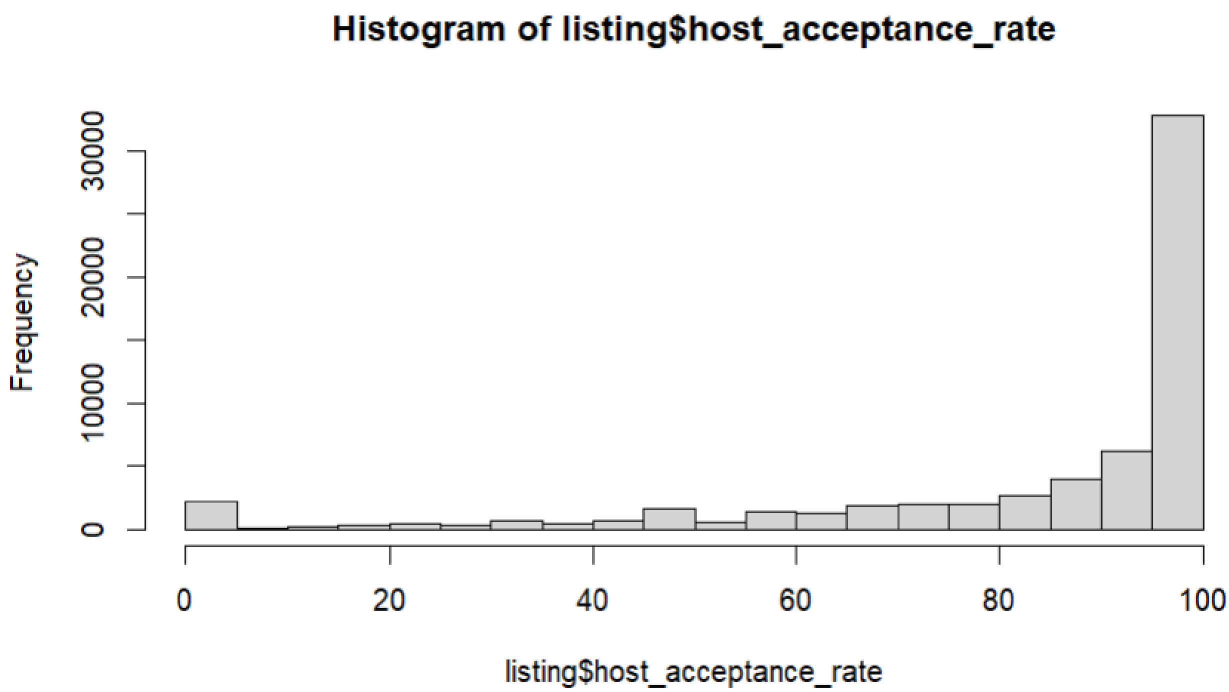


Figure: Before Replacing NA’s

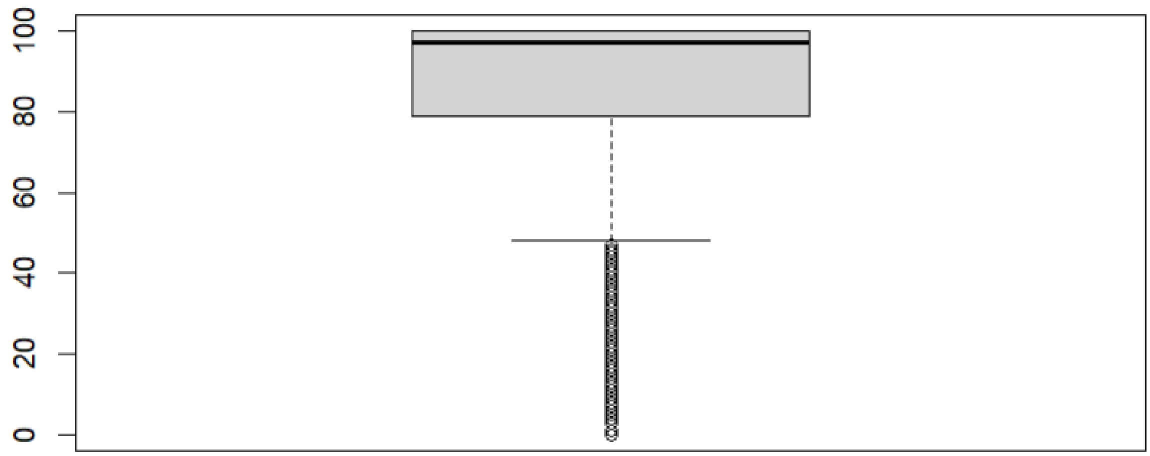


Figure: Boxplot before replacing Na’s

Code:
listing$host_acceptance_rate[is.na(listing$host_acceptance_rate)]=mean(listing$host_acceptance_rate,na.rm = TRUE)

```
> median(listing$host_acceptance_rate,na.rm = TRUE)
[1] 97
> mean(listing$host_acceptance_rate,na.rm = TRUE)
[1] 84.02298
> listing$host_acceptance_rate[is.na(listing$host_acceptance_rate)]=mean(listing$host_acceptance
_rate,na.rm = TRUE)
> hist(listing$host_acceptance_rate)
> 
```

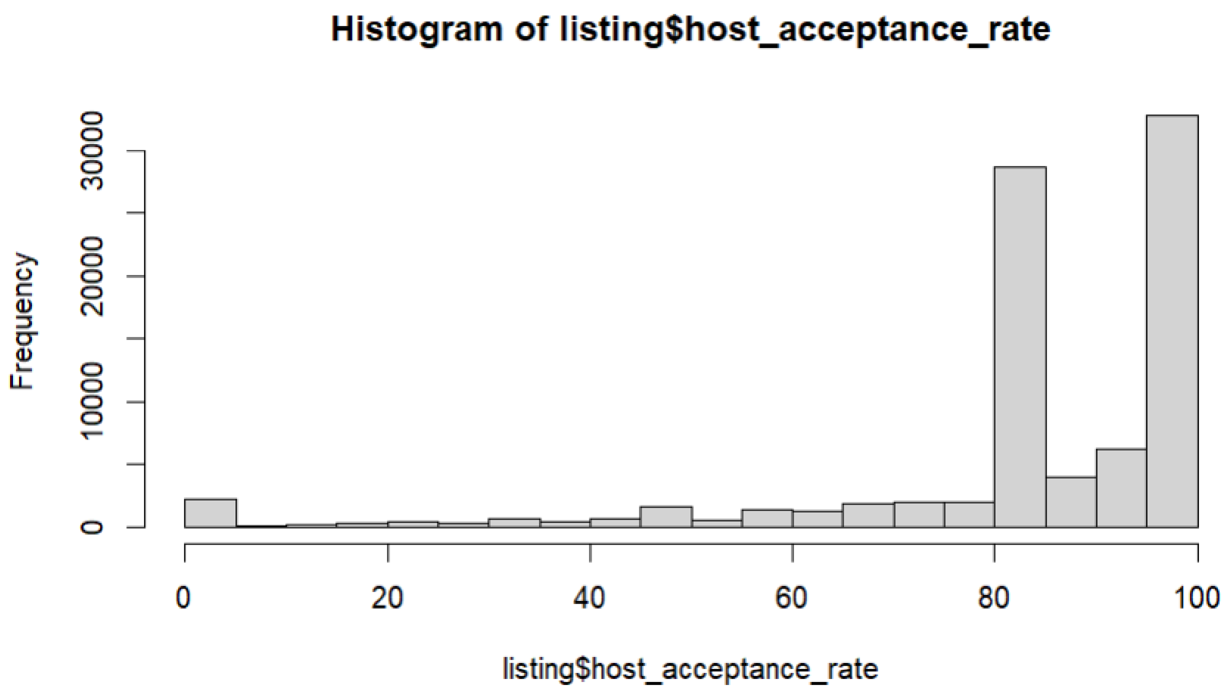


Figure: after replacing Na's

18. Host_is_superhost: indicates if the host is a good host having high rating and reliable. **I** will be using this column to derive the dependent variable**.** Converting into factor (T,F)

Code: listing$host_is_superhost=as.factor(listing$host_is_superhost)

```
> sum(is.na(listing$host_is_superhost))
[1] 0
> sum(listing$host_is_superhost=="")
[1] 1547
>
```

```
> unique(listing$host_is_superhost)
  [1] "f"
  [2] "t"
  [3] ""
  [4] " BBQ grill"
  [5] " Free street parking"
  [6] " Long term stays allowed"
  [7] " Kitchen"
  [8] " Smoke alarm"
  [9] " Carbon monoxide alarm"
 [10] " Free parking on premises"
 [11] " with shops and post office just around the corner and two large supermarkets w
hin walking distance. Our street has free parking.<br /><br />At home"
 [12] " Oven"
 [13] " Stove"
 [14] " Hair dryer"
 [15] " United Kingdom,I'm a bubbly"
 [16] " it comes with all the basic facilities a person needs for short stay.,,https:/
```

After detail inspectio, I found that this column is currupted hence it should be removed.

listing$host_is_superhost=NULL

```
> listing$host_is_superhost=NULL
```

I am creating a function for determining the mode of this column so that I can replace all the null values with mode.

Code: Modenew=function(x) {na.omit(x)[which.max(tabulate(match(x, na.omit(x))))]}

19. Host_thumbnail_url: It is an url of thumbnail profile and I will not be using this column. Code: listing$host_thumbnail_url=NULL

20. Host_picture_url: It is also an url and I will not be using this column. Code: listing$host_picture_url=NULL

21. Host_neighbourhood: this shows the neighbourhood of the host. And I will not be using this column. Code: listing$host_neighbourhood=NULL

22. Host_listing_count: This shows how many active properties has the host listed in Airbnb. Code: listing$host_listings_count=as.numeric(listing$host_listings_count)

```
> sum(is.na(listing$host_listings_count))
[1] 1263
> sum((listing$host_listings_count==""))
[1] NA
```

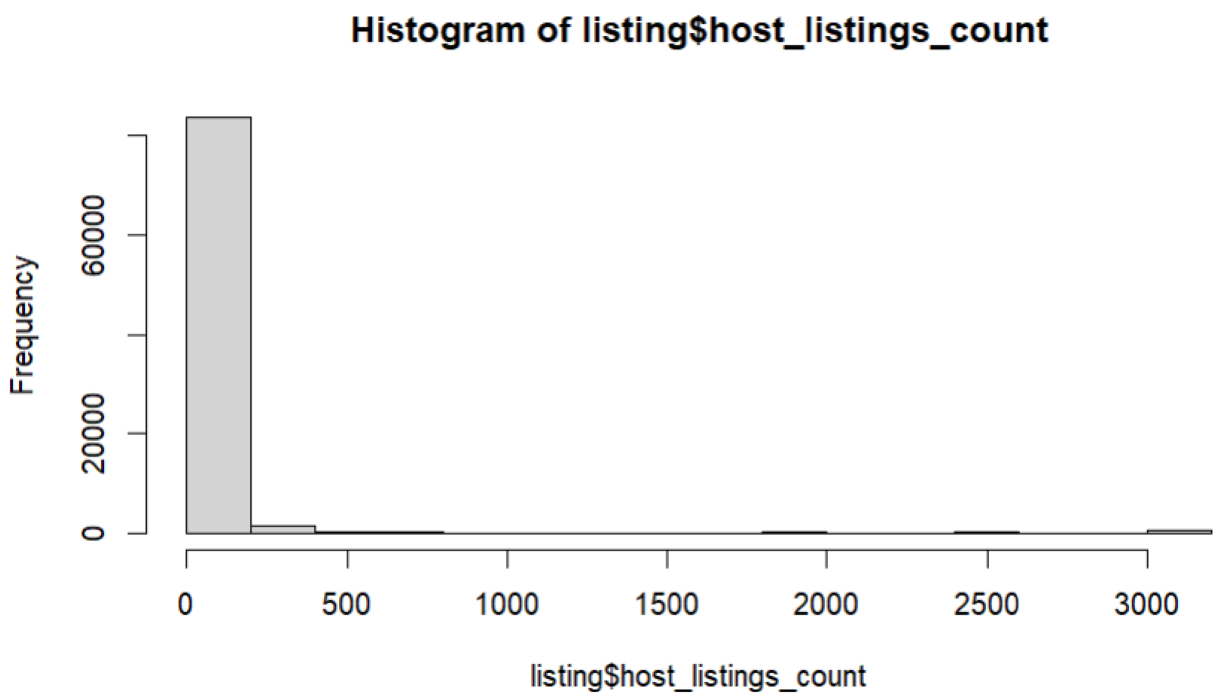


Figure: Before filling Na’s

We will fill the Na with most repeated item.

Code:

Modenew=function(x) {na.omit(x)[which.max(tabulate(match(x, na.omit(x))))]}

host_listing_count_mode=Modenew(listing$host_listings_count)

```
> Modenew=function(x) {na.omit(x)[which.max(tabulate
(match(x, na.omit(x))))]}
> host_listing_count_mode=Modenew(listing$host_listin
gs_count)
>
> host_listing_count_mode=Modenew(listing$host_list
gs_count)
> unique(listing$host_listings_count)
  [1]    1    3   10    4    2   12    7    5    9
 [10]   13    8   26    6   11   25   22   48   15
 [19]   83   74   24   44   14   20   86  118   32
 [28]   NA   36  196   33   30   91   17   43   84
 [37]   39   95   57   19   67   55  499   21  637
 [46]   38  555   18   53   23   79   54   16   65
 [55]   28   92   60   47   35  124   72   73   29
 [64]   62   97   61  106   52   49  149   46   31
 [73]  204   37   63  215   64   34   27  267   71
 [82]   41  139  148   42 1830   76   51   59  181
 [91]  310  154   68 1085 1469  302 1205   99 3023
[100]   75  360  375   69   56  110   40   66  138
[109]  381  245  501   50  200   45 2538  128  178
[118]  183  115  225   89  252  114  191  241 1360
[127]  349  189 1401  130   80
```

listing$host_listings_count[is.na(listing$host_listings_count)]=Modenew(listing$host_listings_count)

```
> listing$host_listings_count[is.na(listing$host_list
ings_count)]=Modenew(listing$host_listings_count)
> unique(listing$host_listings_count)
  [1]    1    3   10    4    2   12    7    5    9
 [10]   13    8   26    6   11   25   22   48   15
 [19]   83   74   24   44   14   20   86  118   32
 [28]   36  196   33   30   91   17   43   84   39
 [37]   95   57   19   67   55  499   21  637   38
 [46]  555   18   53   23   79   54   16   65   28
 [55]   92   60   47   35  124   72   73   29   62
 [64]   97   61  106   52   49  149   46   31  204
 [73]   37   63  215   64   34   27  267   71   41
 [82]  139  148   42 1830   76   51   59  181  310
 [91]  154   68 1085 1469  302 1205   99 3023   75
[100]  360  375   69   56  110   40   66  138  381
[109]  245  501   50  200   45 2538  128  178  183
[118]  115  225   89  252  114  191  241 1360  349
[127]  189 1401  130   80
```

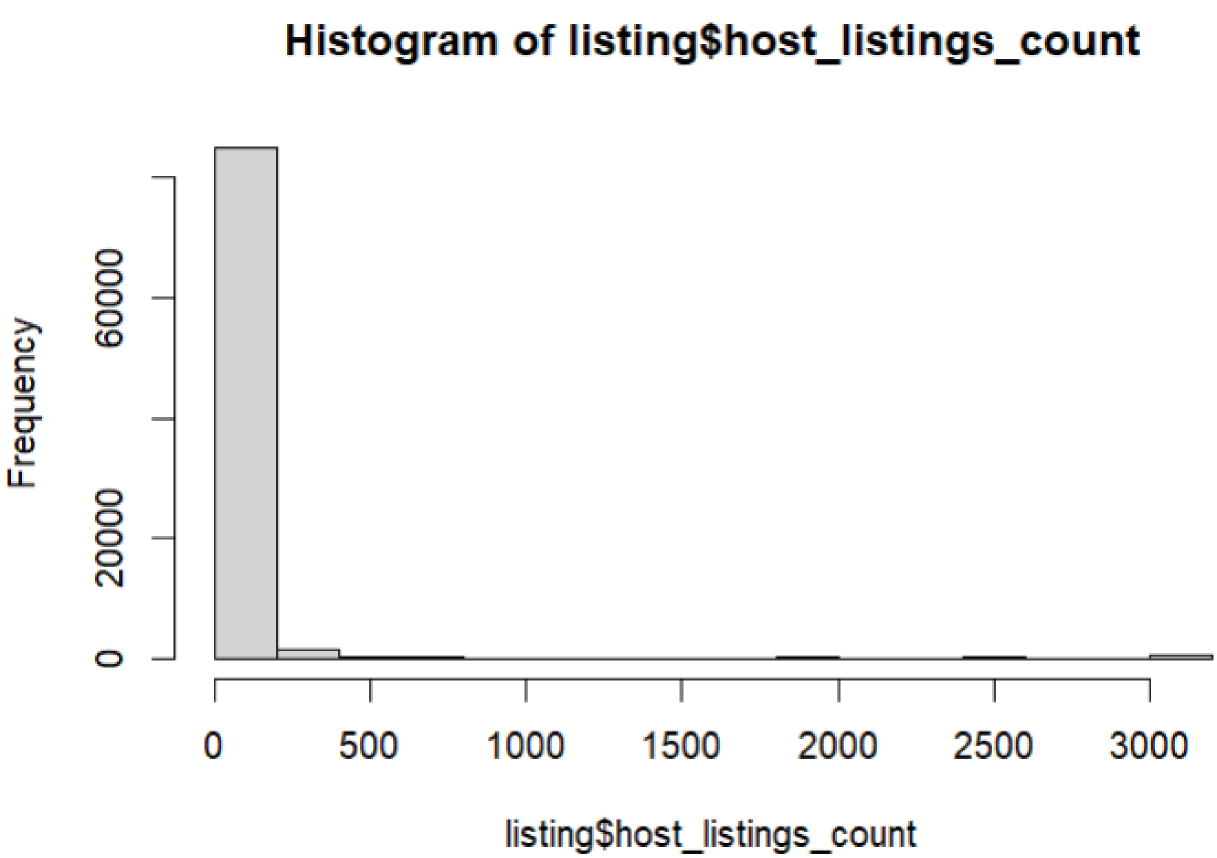


Figure: after replacing Na's

23. Host_total_listing_count: this shows the total number of properties both active and inactive managed by the host. Code: listing$host_total_listings_count=as.numeric(listing$host_total_listings_count)

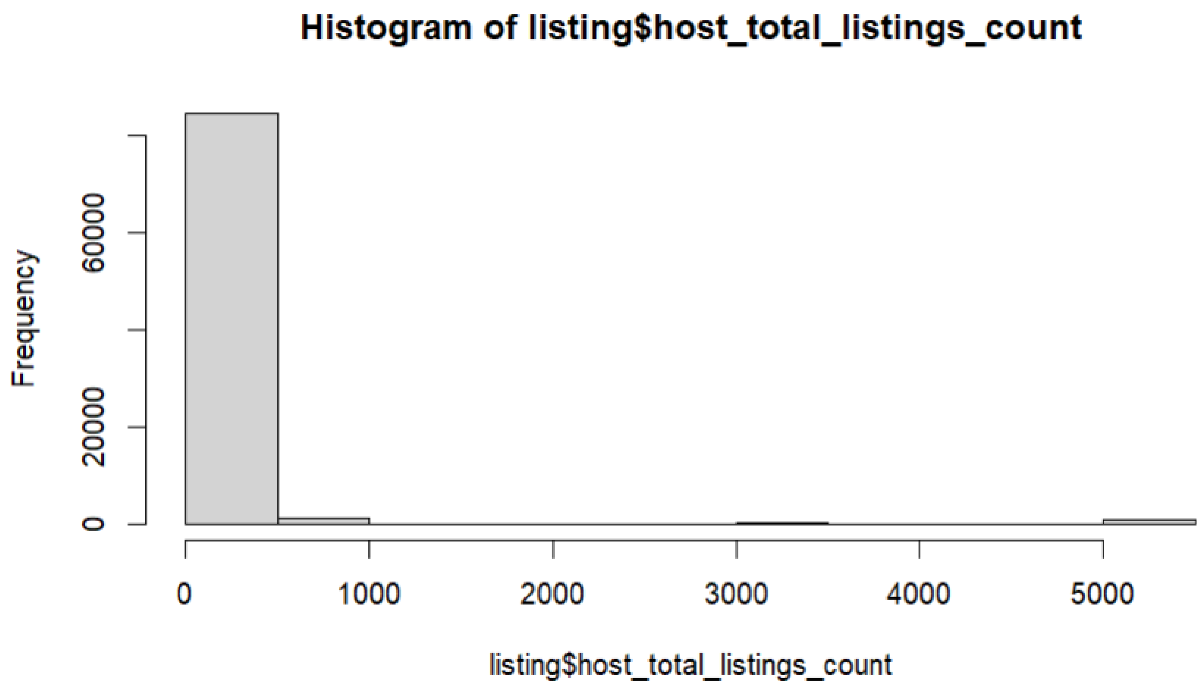


Figure: Before replacing Na’s.

Code:
listing$host_total_listings_count[is.na(listing$host_total_listings_count)]=mean(listing$host_total_listings_count, na.rm = TRUE)

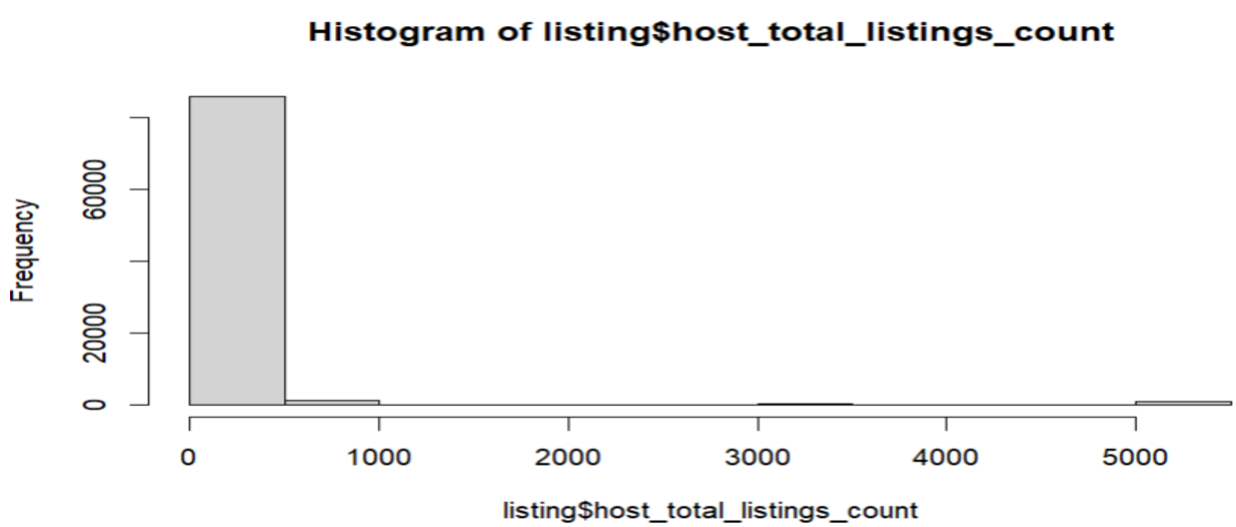


Figure: After Na’s

```
> listing$host_total_listings_count[is.na(listing$host_total_listings_count)]=mean(listing$host_total
_listings_count, na.rm = TRUE)
> hist(listing$host_total_listings_count)
```

24. host_verifications: This column meant to have email and phone number. But I will not be using this column.
    Code: listing$host_verifications=NULL

25. host_has_profile_pic: this is the column that indicates of the host has profile pictire or not.
    Code: listing$host_has_profile_pic=as.factor(listing$host_has_profile_pic)
    The data here is currupted so better to drop this column

    Code: listing$host_has_profile_pic=NULL

26. host_identity_verified: this is the column that indicates if the identity of host is verified or not. Code: listing$host_identity_verified=NULL

27. Neighbourhood: It shows the neighbourbood. I will not be using this column. Code: listing$neighbourhood=NULL

28. Neighbourhood_cleansed: Similar to neighbourhood. I will not be using this column. Code: listing$neighbourhood_cleansed=NULL

29. neighbourhood_group_cleansed: Groping of neighbourhood. I will not be using this column.

    Code: listing$neighbourhood_group_cleansed=NULL
    list
30. Latitude: Latitude of the listing. Code: listing$latitude=as.numeric(listing$latitude)

31. Longitude: Longitude of the listing. Code: listing$longitude=as.numeric(listing$longitude)

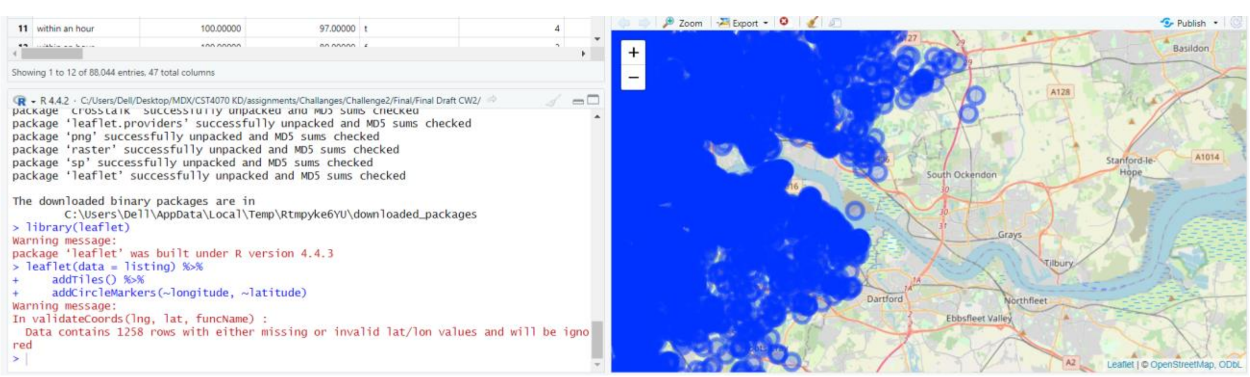


Figure one.

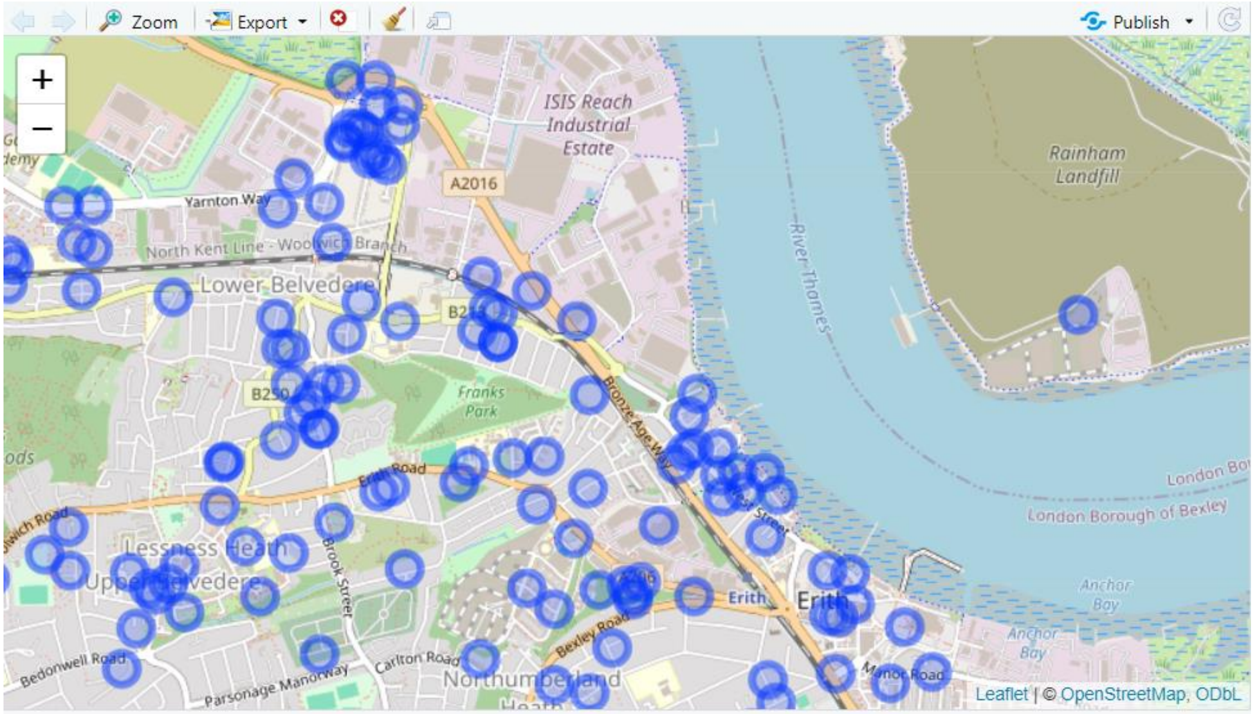


Figure two.

Figure one and two showing the location of property as per longitude and latitude.

During my research I have found that using the longitude and latitude we can actually visualize the position of property on the real map. For that we need to know about the library called "leaflet" which helps to plot the points on maps based on the given longitude and latitude.
But, in my case I don't find this feature that much useful for now. So I am droping this as I will not be able use this on my glm model.
I will be droping this column.

```
> listing$longitude=NULL
> listing$latitude=NULL
>
```

32. Property_type: It indicates the property of the listed property.
The column is currupted and has plenty of non relevent data, I will try to clean it.

Code: listing$property_type= factor(listing$property_type,levels = c("Shared room in house","Private room in apartment","Private room in house","Entire rental unit","Entire condominium (condo)","Serviced apartment","Tiny home","Bed and breakfast","Hotel room"),ordered = TRUE)

Check the number of NA's in the column
sum(is.na(listing$property_type)) #54799
Since it is not good idea to fill this much of data with mean or mode. I think it will be the better idea to drop this column. Even after converting the column into numeric and

replacing the NA with zero, filling that much of data may cause my model to do wrong predictions so I will drop this column.

Code: listing$property_type=NULL

33. Room_type: It gives the information about the type of the room. Ordered factor: This column is also currupted and non relevent data are there but I will try to make it clean.

```
 [86] " Outdoor furniture"
 [87] " Free washer \\u2013 In building"
 [88] " Coffee maker"
 [89] " United Kingdom,Hi I'm James"
 [90] " and the Victoria and Albert Museum (V&A) all
close by. Other notable landm,https://a0.muscache.cc
m/pictures/2314791d-8f26-49be-8b93-b02a58a5ff69.jpg,
19793375,https://www.airbnb.com/users/show/21979337
Bruce,2018-10-10,,\",N/A,N/A,N/A,f,https://a0.muscac
e.com/im/pictures/user/8045bcdd-24b6-46d0-b218-094bb
1d5d2b.jpg?aki_policy=profile_small,https://a0.musca
he.com/im/pictures/user/8045bcdd-24b6-46d0-b218-094b
31d5d2b.jpg?aki_policy=profile_x_medium,Knightsbridg
e,1,1,['phone'],t,t,Greater London"
 [91] " United Kingdom,Lambeth,,51.45869,-0.13324,En
ire rental unit,Entire home/apt,4,,2 baths,2,2,[Self
check-in"
```

Figure: sample of data in this column

Code: listing$room_type=factor(listing$room_type,levels = c("Shared room", "Private room", "Entire home/apt"),ordered = TRUE)

```
> sum(is.na(listing$room_type))
[1] 1462
```

Since the missing and non relevent data is less so I will be converting this column into numeric data type and replace the missing values with mode of the room type.

Code: listing$room_type=as.numeric(listing$room_type)

listing$room_type[is.na(listing$room_type)]=Modenew(listing$room_type)

```
> listing$room_type=as.numeric(listing$room_type)
> listing$room_type[is.na(listing$room_type)]=Modenew
(listing$room_type)
> sum(is.na(listing$room_type))
[1] 0
```

```
> unique(listing$room_type)
[1] 2 3 1
```

Here we can see three unique room types,

I will convert this column into factors.

Code: listing$room_type=as.factor(listing$room_type)

34. Accommodates: It gives the information about max number of people that people can live in the property.
    Code: listing$accommodates=as.numeric(listing$accommodates)

```
> sum(is.na(listing$accommodates))
[1] 1258
```

Since missing data is few, I will replace missing data with mean of this column
Code:
listing$accommodates[is.na(listing$accommodates)]=mean(listing$accommodates)

```
> listing$accommodates[is.na(listing$accommodates)]=m
ean(listing$accommodates,na.rm = TRUE)
> sum(is.na(listing$accommodates))
[1] 0
```

35. Bathrooms: Shows the number of bathrooms Code: listing$bathrooms=NULL

36. Bathroom_text: Shows type and number of bathrooms

    Code: listing$bathrooms_text=NULL

37. Bedrooms: Shows the number of bedrooms Code: listing$bedrooms=as.numeric(listing$bedrooms)

```
> sum(is.na(listing$bedrooms))
[1] 33525
```

Better to drop.

Code: listing$bedrooms=NULL

38. Beds: Shows number of beds. Code: listing$beds=as.numeric(listing$beds)

```
> sum(is.na(listing$beds))
[1] 2384
```

I can fill this data with mode of this column

Code: listing$beds[is.na(listing$beds)]=Modenew(listing$beds)

```
> listing$beds[is.na(listing$beds)]=Modenew(listing$b
eds)
> sum(is.na(listing$beds))
[1] 0
> |
```

39. Amenities: Shows the types of amenities available in the property.

```
# Define amenity tiers
good_amenities   = c("Wifi", "Heating", "Essentials", "Smoke alarm", "Shampoo")
better_amenities = c("Kitchen", "TV", "Hair dryer", "Iron", "Washer", "Dryer", "Coffee
maker")
best_amenities   = c("Pool", "Gym", "Hot tub", "Free parking", "Air conditioning", "EV
charger", "Indoor fireplace")

# Clean and split amenities
clean_amenities = gsub("\\[|\\]|\"", "", listing$amenities)
amenity_list = strsplit(clean_amenities, ",\\s*")

# Assign tier based on presence
listing$amenity_tier = sapply(amenity_list, function(a) { a = trimws(a)  # remove extra
spaces
 if (any(a %in% best_amenities)) {return("Best") }
 else if (any(a %in% better_amenities)) {
      return("Better")}
 else if (any(a %in% good_amenities)) {
   return("Good") }
else { return(NA)  # or "Unknown" } })

# Convert to ordered factor
listing$amenity_tier = factor(listing$amenity_tier, levels = c("Good", "Better", "Best"),
ordered = TRUE)
```

I found the data to be currupted so better to drop.

Code: listing$amenities=NULL

40. Price: shows the price per night for the property.

install.packages("readr")

library(readr)
listing$price=parse_number(listing$price)

```
> sum(is.na(listing$price))
[1] 1187
```

I will fill this missing values with mean of price.

Code: listing$price[is.na(listing$price)]=mean(listing$price,na.rm = TRUE)

```
> listing$price[is.na(listing$price)]=mean(listing$pr
ice,na.rm = TRUE)
> sum(is.na(listing$price))
[1] 0
```

41. Minimum_nights: Shows minimun nights required for booking the property. Code: listing$minimum_nights=as.numeric(listing$minimum_nights)

Code: listing$minimum_nights[is.na(listing$maximum_nights)]=mean(listing$minimum_nights, na.rm = TRUE)

```
> sum(is.na(listing$minimum_nights))
[1] 1258
> listing$minimum_nights[is.na(listing$maximum_nights)]=mean(listing$minimum_nights,na.
rm = TRUE)
> sum(is.na(listing$minimum_nights))
[1] 0
```

42. Maximum_nights: Shows maximum nights that the guest can book the property. Code: listing$maximum_nights=as.numeric(listing$maximum_nights)

listing$maximum_nights[is.na(listing$maximum_nights)]=mean(listing$maximum_nights ,na.rm = TRUE)

```
> sum(is.na(listing$maximum_nights))
[1] 1258
> listing$maximum_nights[is.na(listing$maximum_nights)]=mean(listing$maximum_nights,na.
rm = TRUE)
> sum(is.na(listing$maximum_nights))
[1] 0
```

43. Minimum_minimum_nights: Shows minimum value for minimum nights.
    Code:
    listing$minimum_minimum_nights=as.numeric(listing$minimum_minimum_nights)

```
> sum(is.na(listing$minimum_minimum_nights))
[1] 1259
> listing$minimum_minimum_nights[is.na(listing$minimum_minimum_nights)]=mean(listing$mi
nimum_minimum_nights,na.rm = TRUE)
> sum(is.na(listing$minimum_minimum_nights))
[1] 0
> |
```

    Code:
    listing$minimum_minimum_nights[is.na(listing$minimum_minimum_nights)]=mean(listing$minimum_minimum_nights,na.rm = TRUE)

44. Maximum_minimum_nights: Shows maximum value for minimum nights.
    Code:
    listing$maximum_minimum_nights=as.numeric(listing$maximum_minimum_nights)

```
> sum(is.na(listing$maximum_minimum_nights))
[1] 1259
> listing$maximum_minimum_nights[is.na(listing$maximum_minimum_nights)]=mean(listing$ma
ximum_minimum_nights,na.rm = TRUE)
> sum(is.na(listing$maximum_minimum_nights))
[1] 0
> |
```

    Code:
    listing$maximum_minimum_nights[is.na(listing$maximum_minimum_nights)]=mean(listing$maximum_minimum_nights,na.rm = TRUE)

45. Minimum_maximum_nights: Shows minimum value for maximum nights.
    Code:
    listing$minimum_maximum_nights=as.numeric(listing$minimum_maximum_nights)

    listing$minimum_maximum_nights[is.na(listing$minimum_maximum_nights)]=mean(listing$minimum_maximum_nights,na.rm = TRUE)

```
> sum(is.na(listing$minimum_maximum_nights))
[1] 1259
> listing$minimum_maximum_nights[is.na(listing$minimum_maximum_nights)]=mean(listing$minimum_maximum_nights,na.rm = TRUE)
> sum(is.na(listing$minimum_maximum_nights))
[1] 0
```

46. Maximum_maximum_nights: Shows maximum value for maximum nights.
    Code:
    listing$maximum_maximum_nights=as.numeric(listing$maximum_maximum_nights)

```
> sum(is.na(listing$maximum_maximum_nights))
[1] 1259
> listing$maximum_maximum_nights[is.na(listing$maximum_maximum_nights)]=mean(listing$maximum_maximum_nights,na.rm = TRUE)
> sum(is.na(listing$maximum_maximum_nights))
[1] 0
```

    listing$maximum_maximum_nights[is.na(listing$maximum_maximum_nights)]=mean(listing$maximum_maximum_nights,na.rm = TRUE)

47. Minimum_nights_avg_ntm: average number of minimum nights.
    Code: listing$minimum_nights_avg_ntm=as.numeric(listing$minimum_nights_avg_ntm)

    listing$minimum_nights_avg_ntm[is.na(listing$minimum_nights_avg_ntm)]=mean(listing$minimum_nights_avg_ntm,na.rm = TRUE)

```
> sum(is.na(listing$minimum_nights_avg_ntm))
[1] 1259
> listing$minimum_nights_avg_ntm[is.na(listing$minimum_nights_avg_ntm)]=mean(listing$minimum_nights_avg_ntm,na.rm = TRUE)
> sum(is.na(listing$minimum_nights_avg_ntm))
[1] 0
```

48. Maximum_nights_avg_ntm: average number of maximum nights.
    Code:
    listing$maximum_nights_avg_ntm=as.numeric(listing$maximum_nights_avg_ntm)

    listing$maximum_nights_avg_ntm[is.na(listing$maximum_nights_avg_ntm)]=mean(listing$maximum_nights_avg_ntm, na.rm = TRUE)

```
> sum(is.na(listing$maximum_nights_avg_ntm))
[1] 1259
> listing$maximum_nights_avg_ntm[is.na(listing$maximum_nights_avg_ntm)]=mean(listing$maximum_nights_avg_ntm, na.rm = TRUE )
> sum(is.na(listing$maximum_nights_avg_ntm))
[1] 0
```

49. Calander_updated: when was the listing calander last updated.
Code: listing$calendar_updated=NULL

50. Has_availability: Shows of the listing is available for booking or not.
Code: listing$has_availability=as.factor(listing$has_availability)

Column is currupted. So better to drop.

listing$has_availability=NULL

51. Availability_30: number of available nights in 30 days.

Code: listing$availability_30=as.numeric(listing$availability_30)

listing$availability_30[is.na(listing$availability_30)]=Modenew(listing$availability_30)

```
> sum(is.na(listing$availability_30))
[1] 1258
> listing$availability_30[is.na(listing$availability_30)]=Modenew(listing$availability_3
0)
> sum(is.na(listing$availability_30))
[1] 0
> |
```

52. Availability_60: number of available nights in 60 days.
Code: listing$availability_60=as.numeric(listing$availability_60)

listing$availability_60[is.na(listing$availability_60)]=Modenew(listing$availability_60)

```
> sum(is.na(listing$availability_60))
[1] 1258
> listing$availability_60[is.na(listing$availability_60)]=Modenew(listing$availability_6
0)
> sum(is.na(listing$availability_60))
[1] 0
> |
```

53. Availability_90: number of available nights in 90 days.
listing$availability_90=as.numeric(listing$availability_90)
listing$availability_90[is.na(listing$availability_90)]=Modenew(listing$availability_90)

```
> sum(is.na(listing$availability_90))
[1] 1258
> listing$availability_90[is.na(listing$availability_90)]=Modenew(listing$availability_
90)
> sum(is.na(listing$availability_90))
[1] 0
```

54. Availability_365: number of available nights in 365 days

Code: listing$availability_365=as.numeric(listing$availability_365) listing$availability_365[is.na(listing$availability_365)]=Modenew(listing$availability_365)

```
> sum(is.na(listing$availability_365))
[1] 1258
> listing$availability_365[is.na(listing$availability_365)]=Modenew(listing$availabilit
y_365)
> sum(is.na(listing$availability_365))
[1] 0
> |
```

55. Calander_last_scrapped: This shows when was the calander data last retrieved.
    Code: listing$calendar_last_scraped=as.Date(listing$calendar_last_scraped)

56. Number_of_reviews: shows total number of reviews the listing has received.
    Code: listing$number_of_reviews=as.numeric(listing$number_of_reviews)

    listing$number_of_reviews[is.na(listing$number_of_reviews)]=mean(listing$number_of_reviews,na.rm = TRUE)

```
> sum(is.na(listing$number_of_reviews))
[1] 1257
> listing$number_of_reviews[is.na(listing$number_of_reviews)]=mean(listing$number_of_re
views,na.rm = TRUE)
> sum(is.na(listing$number_of_reviews))
[1] 0
```

57. Number_of_reviews_ltm: shows the number of reviews of items in the property.
    Code: listing$number_of_reviews_ltm=as.numeric(listing$number_of_reviews_ltm)

    listing$number_of_reviews_ltm[is.na(listing$number_of_reviews_ltm)]=Modenew(listing$number_of_reviews_ltm)

```
> sum(is.na(listing$number_of_reviews_ltm))
[1] 1256
> listing$number_of_reviews_ltm[is.na(listing$number_of_reviews_ltm)]=Modenew(listing$n
umber_of_reviews_ltm)
> sum(is.na(listing$number_of_reviews_ltm))
[1] 0
```

58. Number_of_reviews_l30d: shows the number of reviewsin last 30 days.
    Code: listing$number_of_reviews_l30d=as.numeric(listing$number_of_reviews_l30d)

    listing$number_of_reviews_l30d[is.na(listing$number_of_reviews_l30d)]=Modenew(listing$number_of_reviews_l30d)

```
> sum(is.na(listing$number_of_reviews_l30d))
[1] 1257
> listing$number_of_reviews_l30d[is.na(listing$number_of_reviews_l30d)]=Modenew(listing
$number_of_reviews_l30d)
> sum(is.na(listing$number_of_reviews_l30d))
[1] 0
```

59. First_review: shows first review date.

    Code: listing$first_review=NULL

60. Last_review: shows last review date.

    Code: listing$last_review=NULL

61. Review_scores_rating: shows average of all ratings.
    Code: listing$review_scores_rating=as.numeric(listing$review_scores_rating)
    listing$review_scores_rating[is.na(listing$review_scores_rating)]=mean(listing$review_scores_rating,na.rm = TRUE)

```
> sum(is.na(listing$review_scores_rating))
[1] 23192
> listing$review_scores_rating[is.na(listing$review_scores_rating)]=mean(listing$review
_scores_rating,na.rm = TRUE)
> sum(is.na(listing$review_scores_rating))
[1] 0
```

62. Review_scores_accuracy: Shows the accuracy of rating.
    Code: listing$review_scores_accuracy=as.numeric(listing$review_scores_accuracy)
    listing$review_scores_accuracy[is.na(listing$review_scores_accuracy)]=Modenew(listing$review_scores_accuracy)

```
> sum(is.na(listing$review_scores_accuracy))
[1] 24105
> listing$review_scores_accuracy[is.na(listing$review_scores_accuracy)]=Modenew(listing
$review_scores_accuracy)
> sum(is.na(listing$review_scores_accuracy))
[1] 0
```

63. Review_score_cleanliness: Shows the rating for cleanliness.
    Code:
    listing$review_scores_cleanliness=as.numeric(listing$review_scores_cleanliness)

    listing$review_scores_cleanliness[is.na(listing$review_scores_cleanliness)]=mean(listing$review_scores_cleanliness,na.rm = TRUE)

```
> sum(is.na(listing$review_scores_cleanliness))
[1] 24093
> listing$review_scores_cleanliness[is.na(listing$review_scores_cleanliness)]=mean(listing$review_scores_cleanliness,na.rm = TRUE)
> sum(is.na(listing$review_scores_cleanliness))
[1] 0
```

64. Review_scores_checkin: Shows review while checkin.

Code: listing$review_scores_checkin=as.numeric(listing$review_scores_checkin)

listing$review_scores_checkin[is.na(listing$review_scores_checkin)]=mean(listing$review_scores_checkin,na.rm = TRUE)

```
> sum(is.na(listing$review_scores_checkin))
[1] 24137
> listing$review_scores_checkin[is.na(listing$review_scores_checkin)]=mean(listing$review_scores_checkin,na.rm = TRUE)
> sum(is.na(listing$review_scores_checkin))
[1] 0
```

65. Review_scores_communication: Shows rating for communication.

Code:
listing$review_scores_communication=as.numeric(listing$review_scores_communication)

listing$review_scores_communication[is.na(listing$review_scores_communication)]=mean(listing$review_scores_communication,na.rm = TRUE)

```
> sum(is.na(listing$review_scores_communication))
[1] 24106
> listing$review_scores_communication[is.na(listing$review_scores_communication)]=mean(listing$review_scores_communication,na.rm = TRUE)
> sum(is.na(listing$review_scores_communication))
[1] 0
```

66. Review_scores_location: Shows rating about the location of property.

Code: listing$review_scores_location=as.numeric(listing$review_scores_location)
listing$review_scores_location[is.na(listing$review_scores_location)]=mean(listing$review_scores_location,na.rm = TRUE)

```
> sum(is.na(listing$review_scores_location))
[1] 24137
> listing$review_scores_location[is.na(listing$review_scores_location)]=mean(listing$review_scores_location,na.rm = TRUE)
> sum(is.na(listing$review_scores_location))
[1] 0
```

67. Review_score_value: Shows review score value.

Code: listing$review_scores_value=as.numeric(listing$review_scores_value)

listing$review_scores_value[is.na(listing$review_scores_value)]=mean(listing$review_scores_value,na.rm = TRUE)

```
> sum(is.na(listing$review_scores_value))
[1] 24138
> listing$review_scores_value[is.na(listing$review_scores_value)]=mean(listing$review_s
cores_value,na.rm = TRUE)
> sum(is.na(listing$review_scores_value))
[1] 0
```

68. Licence: Shows licence.

Code: listing$license=NULL

69. Instant_bookable: If its is instantly bookable or not.

Code: listing$instant_bookable=as.factor(listing$instant_bookable)

```
> unique(listing$instant_bookable)
  [1] t
  [2] f
  [3]
  [4]  Self check-in
  [5]  Free dryer \\u2013 In building
  [6]  DLR station (Langdon Park station).<br /><br />STRICTLY - No Pork
  [7]  Bed linens
  [8]  Clothing storage: closet
  [9]  Freezer
 [10]  Kitchen
 [11]  Iron
 [12]  Room-darkening shades
 [13]  Paid parking on premises
 [14]  'phone'],t,t,London
 [15]  standard cable
 [16]  Extra pillows and blankets
 [17]  Washer
```

```
[53]  Paid street parking off premises
[54]  post office
[55]  Brockwell Park and Sydenham Woods all within a 20-25 minute walk
[56]  Paid parking off premises
[57]  Pack \\u2019n play/Travel crib],$300.00,5,365,5,5,1125,1125,5.0,1125.0,,t,1,1,1,
210,2023-09-07,0,0,0,,,,,,,,,,,t,1,1,0,0,\n29247609,https://www.airbnb.com/rooms/292476
09,20230906022807,2023-09-06,city scrape,Home in Greater London · ★4.73 · 1 bedroom ·
1 bed · 1 bath,<b>The space</b><br />I live on a quiet residential road.<br /><br /><b>
Guest access</b><br />Bedroom and Bathroom & is the only available open space<br /><br
/><b>During your stay</b><br />Mobile phone number is 07807443563<br /><br /><b>Other t
hings to note</b><br />Please do not invite other people into my home.<br /><br />There
is a Laundry Service nearby and local supermarkets and takeaways within walking distanc
e.,,https://a0.muscache.com/pictures/770bc777-0e49-4d73-96ce-0b0f909abee0.jpg,13714312
0,https://www.airbnb.com/users/show/137143120,Charmaine,2017-06-26,Mitcham
[58]  awesome cafes
[59]  Security cameras on property
[60]  United Kingdom,",N/A,N/A,N/A,f,https://a0.muscache.com/im/pictures/user/966f6504
-5f39-4076-946c-771051c080ca.jpg?aki_policy=profile_small,https://a0.muscache.com/im/pi
```

Since the data is currupted, better to drop the column.
listing$instant_bookable=NULL

70. Calculated_host_listings_count: listings managed by host.
Code:
listing$calculated_host_listings_count=as.numeric(listing$calculated_host_listings_count)

```
> sum(is.na(listing$calculated_host_listings_count))
[1] 1258
> listing$calculated_host_listings_count[is.na(listing$calculated_host_listings_count)]
=mean(listing$calculated_host_listings_count,na.rm = TRUE)
> sum(is.na(listing$calculated_host_listings_count))
[1] 0
```

listing$calculated_host_listings_count[is.na(listing$calculated_host_listings_count)]=mean(listing$calculated_host_listings_count,na.rm = TRUE)

71. Calculated_host_listings_count_entire_homes: Number of homes managed by host.
Code:
listing$calculated_host_listings_count_entire_homes=as.numeric(listing$calculated_host_listings_count_entire_homes)

listing$calculated_host_listings_count_entire_homes[is.na(listing$calculated_host_listings_count_entire_homes)]=mean(listing$calculated_host_listings_count_entire_homes,na.rm = TRUE)

```
> sum(is.na(listing$calculated_host_listings_count_entire_homes))
[1] 1258
> listing$calculated_host_listings_count_entire_homes[is.na(listing$calculated_host_lis
tings_count_entire_homes)]=mean(listing$calculated_host_listings_count_entire_homes,na.
rm = TRUE)
> sum(is.na(listing$calculated_host_listings_count_entire_homes))
[1] 0
```

72. Calculated_host_listings_count_private_rooms: Number of private rooms managed by host.

Code:
listing$calculated_host_listings_count_private_rooms=as.numeric(listing$calculated_host_listings_count_private_rooms)

```
> sum(is.na(listing$calculated_host_listings_count_private_rooms))
[1] 1258
> listing$calculated_host_listings_count_private_rooms[is.na(listing$calculated_host_listings_count_private_rooms)]=mean(listing$calculated_host_listings_count_private_rooms,na.rm = TRUE)
> sum(is.na(listing$calculated_host_listings_count_private_rooms))
[1] 0
```

listing$calculated_host_listings_count_private_rooms[is.na(listing$calculated_host_listings_count_private_rooms)]=mean(listing$calculated_host_listings_count_private_rooms, na.rm = TRUE)

73. Calculated_host_listings_count_shared_rooms: Number of shared rooms managed by host.

Code:
listing$calculated_host_listings_count_shared_rooms=as.numeric(listing$calculated_host_listings_count_shared_rooms)

listing$calculated_host_listings_count_shared_rooms[is.na(listing$calculated_host_listings_count_shared_rooms)]=mean(listing$calculated_host_listings_count_shared_rooms, na.rm = TRUE)

```
> sum(is.na(listing$calculated_host_listings_count_shared_rooms))
[1] 1258
> listing$calculated_host_listings_count_shared_rooms[is.na(listing$calculated_host_listings_count_shared_rooms)]=mean(listing$calculated_host_listings_count_shared_rooms,na.rm = TRUE)
> sum(is.na(listing$calculated_host_listings_count_shared_rooms))
[1] 0
```

74. Reviews_per_month: shows reviews per month.

Code: listing$reviews_per_month=as.numeric(listing$reviews_per_month)

listing$reviews_per_month[is.na(listing$reviews_per_month)]=mean(listing$reviews_per_month,na.rm = TRUE)

```
> sum(is.na(listing$reviews_per_month))
[1] 23193
> listing$reviews_per_month[is.na(listing$reviews_per_month)]=mean(listing$reviews_per_month,na.rm = TRUE)
> sum(is.na(listing$reviews_per_month))
[1] 0
```

Lets create some new columns:

No of days the host has been registered in airbnb:

We will do some feature engineering.

Which can be calculated by substracting host_since from calender_last_scraped
Code: listing$host_engaged_days=as.numeric(listing$calendar_last_scraped-listing$host_since)

Now convert this into years:

Code: listing$host_engaged_years=listing$host_engaged_days/365

listing$host_engaged_years[is.na(listing$host_engaged_years)]=mean(listing$host_engaged_years,na.rm = TRUE)

Now we can drop the earlier column

listing$host_engaged_days=NULL

listing$host_since=NULL

listing$calendar_last_scraped=NULL

```
R 4.4.2 · C:/Users/Dell/Desktop/MDX/CST4070 KD/assignments/Challanges/Challenge2/Final/Final Draft CW2/
> listing$host_engaged_days=as.numeric(listing$calendar_last_scraped-listing$host_since)
> listing$host_engaged_years=listing$host_engaged_days/365
> listing$host_engaged_days=NULL
> listing$host_since=NULL
> listing$calendar_last_scraped=NULL
>
```

```
> sum(is.na(listing$host_engaged_years))
[1] 1263
> listing$host_engaged_years[is.na(listing$host_engaged_years)]=mean(listing$host_engaged_years,na.rm = TRUE)
> sum(is.na(listing$host_engaged_years))
[1] 0
>
```

From the point of view of Airbnb, a lising can be a good listing if:

A. It has got good reviews score(this indicates that the clients are happy with the listing) like minimum reviews shows the popularity and no of reviews per month shows the engagement of guests in the property.
B. The property can be booked for more days(availability_365 is more), it can generate more revinue.
C. good responding host makes the experience of clients better.

D. Price should not be too low or too high.

So I will be taking number_of_reviews, reviews_per_month, review_score_rating, , availability_365, price,

Lets see the summary of the listing todetermine the values that needs to be added as the condition while determining the dependent variable.

```
host_response_rate host_acceptance_rate host_listings_count host_total_listings_count   room_type      accommodates        beds            price
Min.   :  0.00     Min.   :  0.00       Min.   :   1.00     Min.   :   1.00           Min.   :1.000   Min.   : 1.000   Min.   : 1.000   Min.   :    0.0
1st Qu.: 93.46     1st Qu.: 84.02       1st Qu.:   1.00     1st Qu.:   1.00           1st Qu.:2.000   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.:   64.0
Median : 97.00     Median : 84.02       Median :   2.00     Median :   3.00           Median :3.000   Median : 2.000   Median : 1.000   Median :  111.0
Mean   : 93.46     Mean   : 84.02       Mean   :  48.44     Mean   :  86.65           Mean   :2.624   Mean   : 3.168   Mean   : 1.783   Mean   :  181.9
3rd Qu.:100.00     3rd Qu.: 99.00       3rd Qu.:   7.00     3rd Qu.:  12.00           3rd Qu.:3.000   3rd Qu.: 4.000   3rd Qu.: 2.000   3rd Qu.:  191.0
Max.   :100.00     Max.   :100.00       Max.   :3023.00     Max.   :5272.00           Max.   :3.000   Max.   :16.000   Max.   :50.000   Max.   :80100.0
minimum_nights     maximum_nights       minimum_minimum_nights maximum_minimum_nights minimum_maximum_nights maximum_maximum_nights minimum_nights_avg_ntm
Min.   :   1.000   Min.   :        1    Min.   :   1.000       Min.   :   1.000       Min.   :1.000e+00      Min.   :1.000e+00      Min.   :   1.00
1st Qu.:   1.000   1st Qu.:       60    1st Qu.:   1.000       1st Qu.:   2.000       1st Qu.:9.900e+01      1st Qu.:1.800e+02      1st Qu.:   1.30
Median :   2.000   Median :      365    Median :   2.000       Median :   3.000       Median :3.650e+02      Median :9.990e+02      Median :   2.70
Mean   :   5.624   Mean   :     6794    Mean   :   5.289       Mean   :   9.541       Mean   :5.018e+05      Mean   :5.018e+05      Mean   :   7.87
3rd Qu.:   4.000   3rd Qu.:     1125    3rd Qu.:   4.000       3rd Qu.:   5.000       3rd Qu.:1.125e+03      3rd Qu.:1.125e+03      3rd Qu.:   5.00
Max.   :1125.000   Max.   :524855552    Max.   :1125.000       Max.   :1125.000       Max.   :2.147e+09      Max.   :2.147e+09      Max.   :1125.00
maximum_nights_avg_ntm availability_30  availability_60 availability_90 availability_365 number_of_reviews number_of_reviews_ltm number_of_reviews_l30d
Min.   :1.000e+00      Min.   : 0.000   Min.   : 0.00   Min.   : 0.00   Min.   :  0.0    Min.   :   0.00   Min.   :  0.000       Min.   :  0.0000
1st Qu.:1.800e+02      1st Qu.: 0.000   1st Qu.: 0.00   1st Qu.: 0.00   1st Qu.:  0.0    1st Qu.:   1.00   1st Qu.:  0.000       1st Qu.:  0.0000
Median :7.310e+02      Median : 1.000   Median : 6.00   Median :13.00   Median : 63.0    Median :   4.00   Median :  1.000       Median :  0.0000
Mean   :5.018e+05      Mean   : 7.733   Mean   :18.47   Mean   :31.01   Mean   :119.8    Mean   :  17.94   Mean   :  5.732       Mean   :  0.5073
3rd Qu.:1.125e+03      3rd Qu.:14.000   3rd Qu.:36.00   3rd Qu.:64.00   3rd Qu.:249.0    3rd Qu.:  17.00   3rd Qu.:  6.000       3rd Qu.:  0.0000
Max.   :2.147e+09      Max.   :30.000   Max.   :60.00   Max.   :90.00   Max.   :365.0    Max.   :1536.00   Max.   :640.000       Max.   :109.0000
```

```
review_scores_rating review_scores_accuracy review_scores_cleanliness review_scores_checkin review_scores_communication review_scores_location review_scores_value
Min.   :  0.000      Min.   :0.000          Min.   :0.000             Min.   :0.000         Min.   :  0.000             Min.   :0.00           Min.   :0.000
1st Qu.:  4.604      1st Qu.:4.750          1st Qu.:4.623             1st Qu.:4.777         1st Qu.:  4.802             1st Qu.:4.72           1st Qu.:4.590
Median :  4.640      Median :5.000          Median :4.623             Median :4.810         Median :  4.850             Median :4.72           Median :4.593
Mean   :  4.604      Mean   :4.791          Mean   :4.623             Mean   :4.777         Mean   :  4.802             Mean   :4.72           Mean   :4.593
3rd Qu.:  4.950      3rd Qu.:5.000          3rd Qu.:4.930             3rd Qu.:5.000         3rd Qu.:  5.000             3rd Qu.:4.95           3rd Qu.:4.860
Max.   :452.000      Max.   :5.000          Max.   :5.000             Max.   :5.000         Max.   :253.000             Max.   :5.00           Max.   :5.000
calculated_host_listings_count calculated_host_listings_count_entire_homes calculated_host_listings_count_private_rooms reviews_per_month host_engaged_years
Min.   :  1.00                 Min.   :  0.00                              Min.   :  0.000                              Min.   : 0.010    Min.   : 0.005479
1st Qu.:  1.00                 1st Qu.:  0.00                              1st Qu.:  0.000                              1st Qu.: 0.260    1st Qu.: 4.276712
Median :  2.00                 Median :  1.00                              Median :  0.000                              Median : 1.000    Median : 7.136986
Mean   : 18.25                 Mean   : 13.92                              Mean   :  4.173                              Mean   : 1.018    Mean   : 6.526133
3rd Qu.:  6.00                 3rd Qu.:  3.00                              3rd Qu.:  1.000                              3rd Qu.: 1.018    3rd Qu.: 8.882192
Max.   :595.00                 Max.   :312.00                              Max.   :285.000                              Max.   :50.250    Max.   :14.767123
listing_is_good listing_is_goodnew
Mode :logical   Mode :logical
FALSE:78248     FALSE:81581
TRUE :9796      TRUE :6463
```

From the summary I can say that thre are outliers in the data sets. I will check each and every dataset, remove the outliers normalize the data and then derive the dependent variable.

I need to take export the edited data. To do so, we need a library called openxlsx.

install.packages("openxlsx")
library(openxlsx)

write.csv(listing,"edited on R dropped independent.csv")

Now lets do some data visualization.

To do data visualization and get some insights of the data, we need to install some libraries like tidyverse, e1071

I will begin by installing and importing the libraries first.

install.packages(“e1071”)

library(e1071)

The use of e1071 library in R is used for ML and statistical analysis. It provides the features like “support vector machines for classification and regression”, “probabilistic classification”, “grouping of similar data points”,”finding skewness” etc. I am using this library to check the skewness of my data set.

install.packages("tidyverse")

library(tidyverse)

The use of tidyverse library is used for data manipulation, visualization and cleaning.

library(dplyr)

It is the part of tidyverse library, for data manipulation

```
> str(listing)
'data.frame':   88044 obs. of  29 variables:
 $ host_acceptance_rate                      : num  100 25 88 41 75 ...
 $ host_listings_count                       : num  1 1 3 1 1 10 1 1 1 3 ...
 $ host_total_listings_count                 : num  1 2 4 12 1 32 3 3 2 3 ...
 $ room_type                                 : Factor w/ 3 levels "1","2","3": 2 3 2 3 2 3 3 3 3 2 ...
 $ accommodates                              : num  2 6 1 2 2 7 5 3 5 2 ...
 $ beds                                      : num  2 3 1 1 1 3 3 1 1 1 ...
 $ minimum_nights                            : num  2 5 1 7 4 3 5 2 2 10 ...
 $ maximum_nights                            : num  730 240 29 30 365 ...
 $ minimum_minimum_nights                    : num  2 5 1 7 2 3 5 2 2 10 ...
 $ maximum_minimum_nights                    : num  2 5 1 7 4 3 5 2 2 10 ...
 $ minimum_maximum_nights                    : num  1125 240 29 30 365 ...
 $ maximum_maximum_nights                    : num  1125 240 29 30 365 ...
 $ minimum_nights_avg_ntm                    : num  2 5 1 7 4 3 5 2 2 10 ...
 $ maximum_nights_avg_ntm                    : num  1125 240 29 30 365 ...
 $ availability_30                           : num  0 13 25 7 5 14 26 0 0 0 ...
 $ availability_60                           : num  0 18 55 7 5 26 56 0 0 0 ...
 $ availability_90                           : num  0 38 85 7 5 50 86 0 0 0 ...
 $ number_of_reviews_ltm                     : num  9 2 11 5 25 4 3 0 0 0 ...
 $ number_of_reviews_l30d                    : num  0 0 0 0 3 0 0 0 0 0 ...
 $ review_scores_accuracy                    : num  4.74 4.76 4.72 4.85 4.7 4.83 4.57 4.89 4.93 4.7 ...
 $ review_scores_cleanliness                 : num  4.86 4.62 4.72 4.88 4.59 4.71 4.7 4.91 4.71 4.94 ...
 $ review_scores_checkin                     : num  4.71 4.85 4.74 4.88 4.63 4.71 5 4.9 4.93 4.91 ...
 $ review_scores_communication               : num  4.67 4.88 4.82 4.83 4.81 4.71 4.96 4.93 5 4.89 ...
 $ review_scores_value                       : num  4.68 4.74 4.69 4.74 4.67 4.6 4.39 4.65 4.8 4.74 ...
 $ calculated_host_listings_count            : num  1 1 2 1 1 9 1 1 1 3 ...
 $ calculated_host_listings_count_entire_homes : num  0 1 1 1 0 9 1 1 1 0 ...
 $ calculated_host_listings_count_private_rooms: num  1 0 1 0 1 0 0 0 0 3 ...
 $ host_engaged_years                        : num  12.4 12.4 13.8 13.8 12.4 ...
 $ listing_is_good                           : Factor w/ 2 levels "0","1": 2 1 1 1 2 1 1 1 1 1 ...
> |
```

```
> summary(listing)
 host_acceptance_rate host_listings_count host_total_listings_count room_type  accommodates         beds         minimum_nights      maximum_nights
 Min.   :  0.00       Min.   :   1.00     Min.   :   1.00           1:  436   Min.   : 1.000   Min.   : 1.000   Min.   :   1.000   Min.   :        1
 1st Qu.: 84.02       1st Qu.:   1.00     1st Qu.:   1.00           2:32236   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.:   1.000   1st Qu.:       60
 Median : 84.02       Median :   2.00     Median :   3.00           3:55372   Median : 2.000   Median : 1.000   Median :   2.000   Median :      365
 Mean   : 84.02       Mean   :  48.44     Mean   :  86.65                     Mean   : 3.558   Mean   : 1.783   Mean   :   5.624   Mean   :     6794
 3rd Qu.: 99.00       3rd Qu.:   7.00     3rd Qu.:  12.00                     3rd Qu.: 5.000   3rd Qu.: 2.000   3rd Qu.:   4.000   3rd Qu.:     1125
 Max.   :100.00       Max.   :3023.00     Max.   :5272.00                     Max.   :17.000   Max.   :50.000   Max.   :1125.000   Max.   :524855552
 minimum_minimum_nights maximum_minimum_nights minimum_maximum_nights maximum_maximum_nights minimum_nights_avg_ntm maximum_nights_avg_ntm availability_30
 Min.   :   1.000       Min.   :   1.000       Min.   :1.000e+00      Min.   :1.000e+00      Min.   :   1.00        Min.   :1.000e+00      Min.   : 0.000
 1st Qu.:   1.000       1st Qu.:   2.000       1st Qu.:9.900e+01      1st Qu.:1.800e+02      1st Qu.:   1.30        1st Qu.:1.800e+02      1st Qu.: 0.000
 Median :   2.000       Median :   3.000       Median :3.650e+02      Median :9.990e+02      Median :   2.70        Median :7.310e+02      Median : 1.000
 Mean   :   5.289       Mean   :   9.541       Mean   :5.018e+05      Mean   :5.018e+05      Mean   :   7.87        Mean   :5.018e+05      Mean   : 7.733
 3rd Qu.:   4.000       3rd Qu.:   5.000       3rd Qu.:1.125e+03      3rd Qu.:1.125e+03      3rd Qu.:   5.00        3rd Qu.:1.125e+03      3rd Qu.:14.000
 Max.   :1125.000       Max.   :1125.000       Max.   :2.147e+09      Max.   :2.147e+09      Max.   :1125.00        Max.   :2.147e+09      Max.   :30.000
 availability_60 availability_90 number_of_reviews_ltm number_of_reviews_l30d review_scores_accuracy review_scores_cleanliness review_scores_checkin
 Min.   : 0.00   Min.   : 0.00   Min.   :  0.000       Min.   :  0.0000       Min.   :0.000          Min.   :0.000             Min.   :0.000
 1st Qu.: 0.00   1st Qu.: 0.00   1st Qu.:  0.000       1st Qu.:  0.0000       1st Qu.:4.750          1st Qu.:4.623             1st Qu.:4.777
 Median : 6.00   Median :13.00   Median :  1.000       Median :  0.0000       Median :5.000          Median :4.623             Median :4.810
 Mean   :18.47   Mean   :31.01   Mean   :  5.732       Mean   :  0.5073       Mean   :4.791          Mean   :4.623             Mean   :4.777
 3rd Qu.:36.00   3rd Qu.:64.00   3rd Qu.:  6.000       3rd Qu.:  0.0000       3rd Qu.:5.000          3rd Qu.:4.930             3rd Qu.:5.000
 Max.   :60.00   Max.   :90.00   Max.   :640.000       Max.   :109.0000       Max.   :5.000          Max.   :5.000             Max.   :5.000
 review_scores_communication review_scores_value calculated_host_listings_count calculated_host_listings_count_entire_homes
 Min.   :  0.000             Min.   :0.000       Min.   :  1.00                 Min.   :  0.00
 1st Qu.:  4.802             1st Qu.:4.590       1st Qu.:  1.00                 1st Qu.:  0.00
 Median :  4.850             Median :4.593       Median :  2.00                 Median :  1.00
 Mean   :  4.802             Mean   :4.593       Mean   : 18.25                 Mean   : 13.92
 3rd Qu.:  5.000             3rd Qu.:4.860       3rd Qu.:  6.00                 3rd Qu.:  3.00
 Max.   :253.000             Max.   :5.000       Max.   :595.00                 Max.   :312.00
 calculated_host_listings_count_private_rooms host_engaged_years  listing_is_good
 Min.   :  0.000                              Min.   : 0.005479   0:81581
 1st Qu.:  0.000                              1st Qu.: 4.276712   1: 6463
 Median :  0.000                              Median : 7.136986
 Mean   :  4.173                              Mean   : 6.526133
 3rd Qu.:  1.000                              3rd Qu.: 8.882192
 Max.   :285.000                              Max.   :14.767123
```

Checking the skewness of each column.

If the value of skewness is >+1 it is called right skewed, if value is <-1 it is called left skewed. If value is near zero then it is supported to be moderate to to low skewed. Whereas, if the skewness is equal to zero, it shows the normal distribution.

```
> apply(listing_numeric,MARGIN = 2,skewness)
Error in x - mean(x) : non-numeric argument to binary operator
In addition: Warning message:
In mean.default(x) : argument is not numeric or logical: returning NA
```

The skewness function only takes numeric values. So we need to filter the numeric values first.

```
> library(dplyr)

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union

Warning message:
package ‘dplyr’ was built under R version 4.4.3
> listing_numeric=select_if(listing,is.numeric)
> 
```

listing_numeric=select_if(listing,is.numeric) apply(listing_numeric,MARGIN = 2, skewness)

```
> listing_numeric=select_if(listing,is.numeric)
> apply(listing_numeric,MARGIN = 2, skewness)
                       host_acceptance_rate                         host_listings_count
                                 -2.3470979                                   8.9340093
                  host_total_listings_count                                accommodates
                                  8.8039468                                   1.3233655
                                       beds                              minimum_nights
                                  3.8798835                                  23.3312285
                             maximum_nights                      minimum_minimum_nights
                                296.4232581                                  23.8729972
                     maximum_minimum_nights                      minimum_maximum_nights
                                 11.2880787                                  66.0768113
                     maximum_maximum_nights                      minimum_nights_avg_ntm
                                 66.0768113                                  13.8165055
                     maximum_nights_avg_ntm                             availability_30
                                 66.0768113                                   1.0770463
                            availability_60                             availability_90
                                  0.7488694                                   0.5359917
                      number_of_reviews_ltm                      number_of_reviews_l30d
                                  7.5599836                                  11.3541392
                     review_scores_accuracy                 review_scores_cleanliness
                                 -4.5767488                                  -3.7227164
                      review_scores_checkin                 review_scores_communication
                                 -5.5566414                                 221.7342511
                        review_scores_value              calculated_host_listings_count
                                 -3.7870559                                   6.4345258
 calculated_host_listings_count_entire_homes calculated_host_listings_count_private_rooms
                                  5.0002217                                  10.2379289
                          host_engaged_years
                                 -0.3761691
> |
```

From the above table from R we can clearly see that most of the data are right skewed and some of them are left skewed.

Now, I am doing some data manipulation to normalize the data.

At first I will check if there are any values that are less than zero in my dataset.

```
> sum(listing_numeric<0)
[1] 0
> |
```

Since there are no values that are less than zero,

Function	Equivalent to	Is safe for small(x)?	Can handle x=0
Log1p(x)	Log(1+x)	Yes	Yes
Log(1+x)	Standard log form	Precision might be lost	Yes

It is the most common way to transform the highly skewed positive numeric data. What it does is, it will compress the large values and spreads the small values, this will make the distribution more symmetric or say closer to normal. I have found that, this is beneficial for regression and classification. It also handles the values that are zero in the better way than other type of log transformation.

Are there any other ways too?

Yes, the below table shows what types of transformation is used in different data sets.

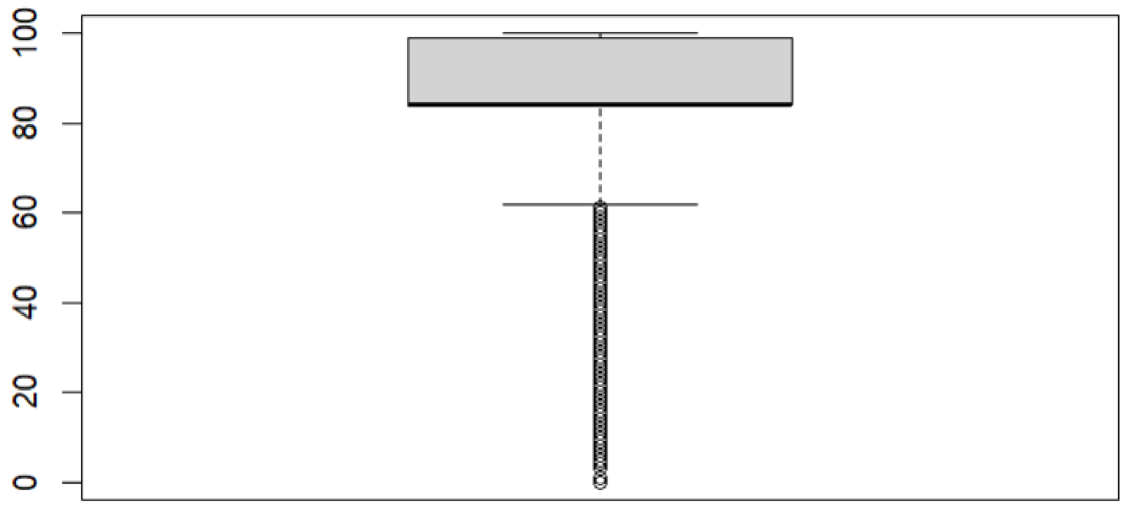
Transformation type	Used for	R function
log(x)	Positive and non zero values	log()
log1p(x)	Positive data having zero values also (right skewed data)	log1p()
Sqrt(x)	For less skewed data	Sqrt()
IQR(Interquartile Range) based Capping	Preserves scale/handles outliers	IQR=Q3−Q1

I will be using IQR-Based Outlier Capping method to handle the outliers.
It will help me to replace the extreme values with boundry values.

Generalized Function to Clean Outliers:
Code:

```
clean_column = function(data, column, min_val = -Inf, max_val = Inf) {
  median_val = median(data[[column]], na.rm = TRUE)
  Q1 = quantile(data[[column]], 0.25, na.rm = TRUE)
  Q3 = quantile(data[[column]], 0.75, na.rm = TRUE)
  IQR = Q3 - Q1
  lower_bound = Q1 - 1.5 * IQR
  upper_bound = Q3 + 1.5 * IQR

  data[[column]] = case_when(
    is.na(data[[column]]) |
    data[[column]] < min_val |
    data[[column]] > max_val ~ median_val,
    data[[column]] < lower_bound ~ pmax(lower_bound, min_val),
    data[[column]] > upper_bound ~ pmin(upper_bound, max_val),
    TRUE ~ data[[column]]
  )

  return(data)}
```

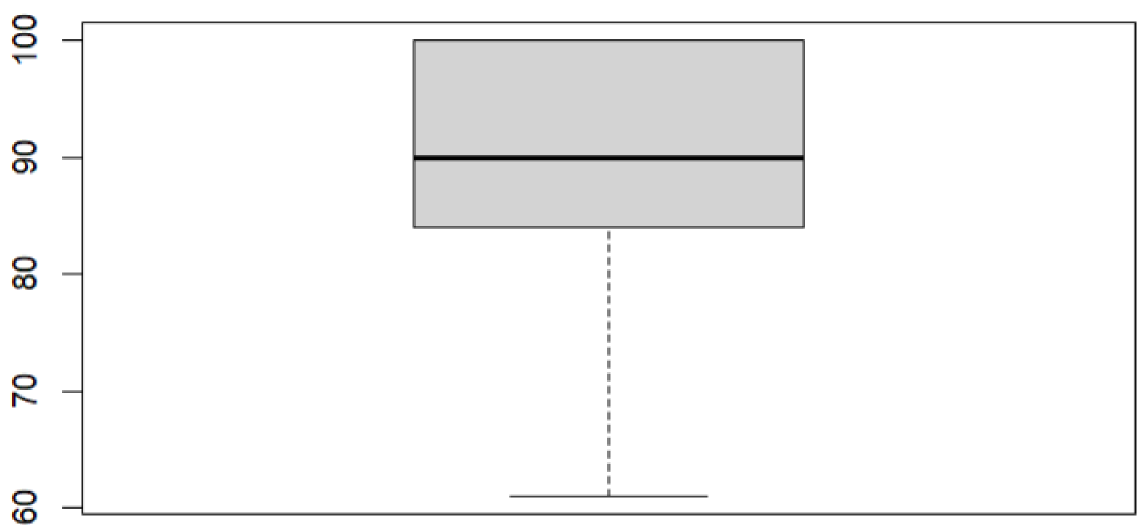
1. $ host_acceptance_rate



We can see that there are outliers in this data set.

```
> listing=listing[listing$host_acceptance_rate>60,]
> boxplot(listing$host_acceptance_rate)
>
```

After removing outlier, boxplot looks like below.

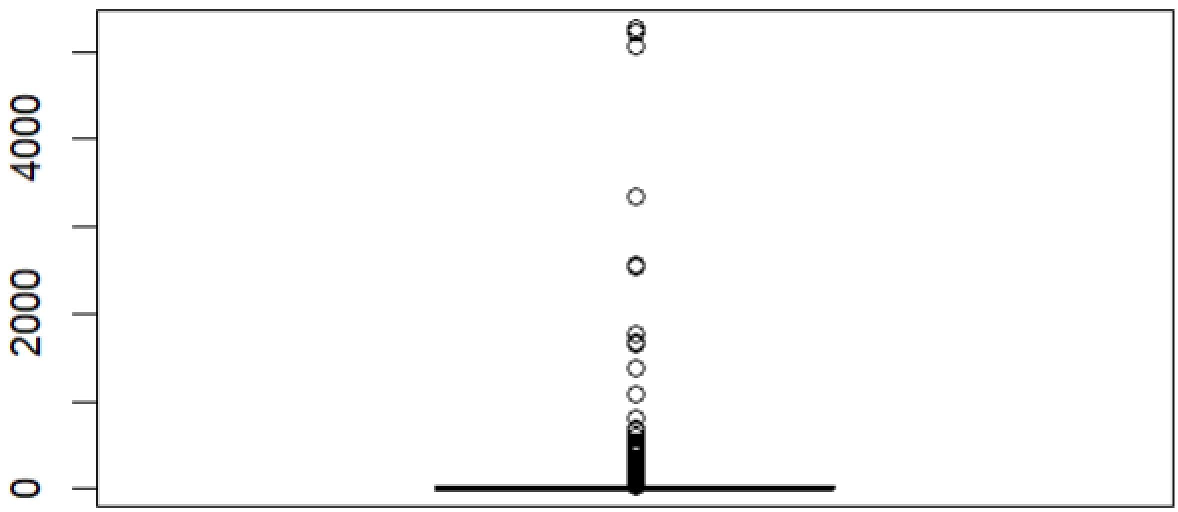


2. $host_listings_count

   listing = clean_column(listing, "host_listings_count", min_val = 1, max_val = 7)

   > boxplot(listing$host_listings_count)

3. $ host_total_listings_count

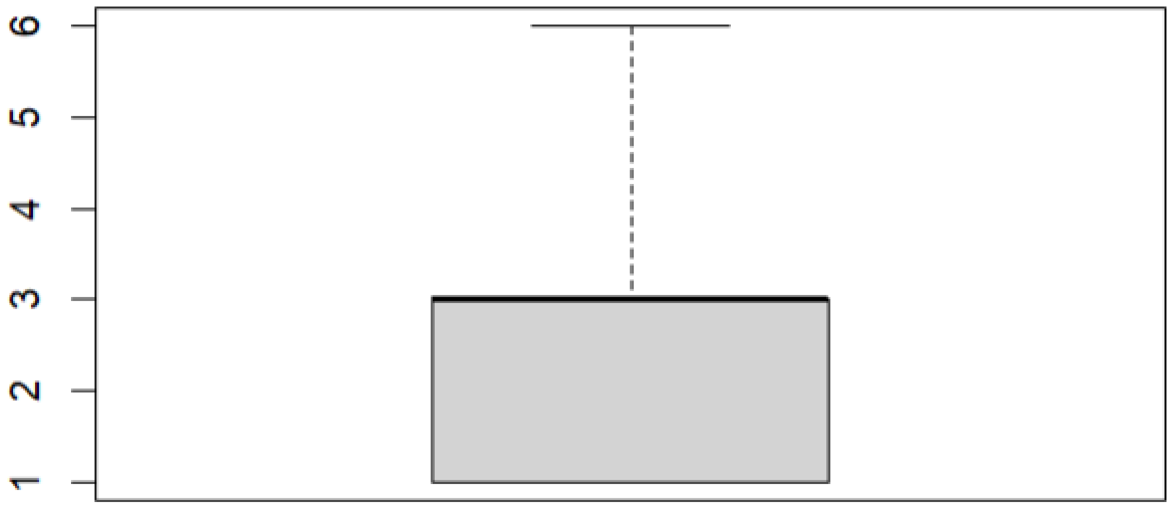


> boxplot(listing$host_total_listings_count)

> listing = clean_column(listing, "host_total_listings_count", min_val = 1, max_val = 6)

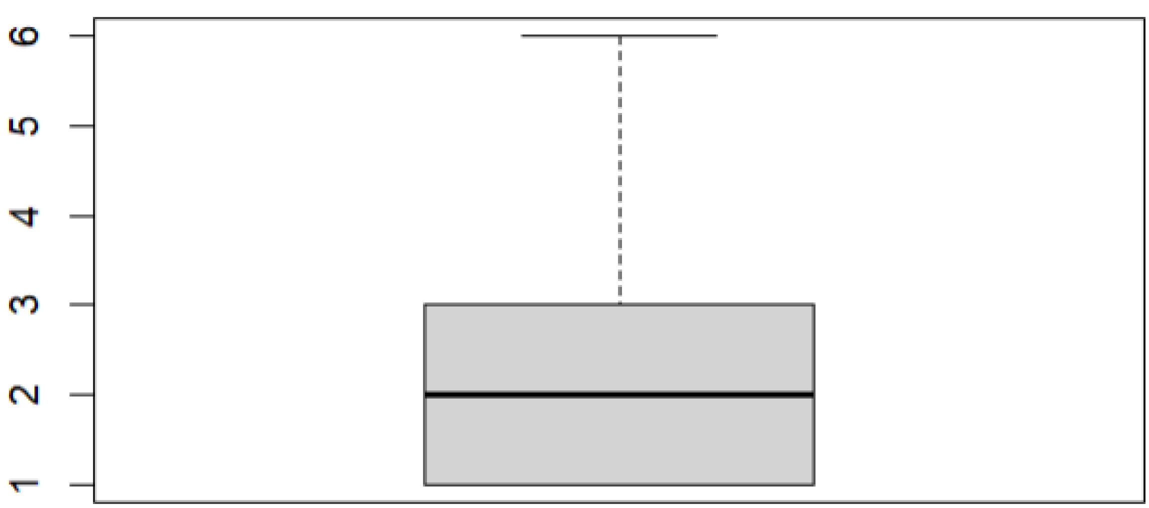
> boxplot(listing$host_total_listings_count)

```
> listing = clean_column(listing, "host_total_listings_count", min_val = 1, max_val = 6)
> boxplot(listing$host_total_listings_count)
```



4. $ room_type
5. $ accommodates
6. $ beds
7. $ minimum_nights

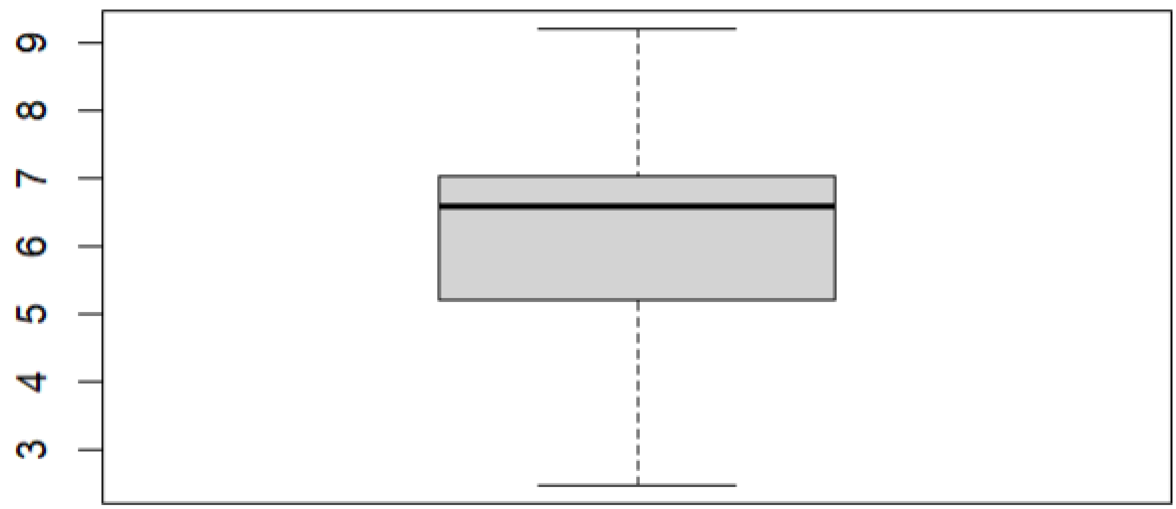
```
> listing = clean_column(listing, "minimum_nights", min_val = 1, max_val = 6)
> boxplot(listing$minimum_nights)
```



8. $ maximum_nights

> listing = clean_column(listing, "maximum_nights", min_val = 1, max_val = 10)

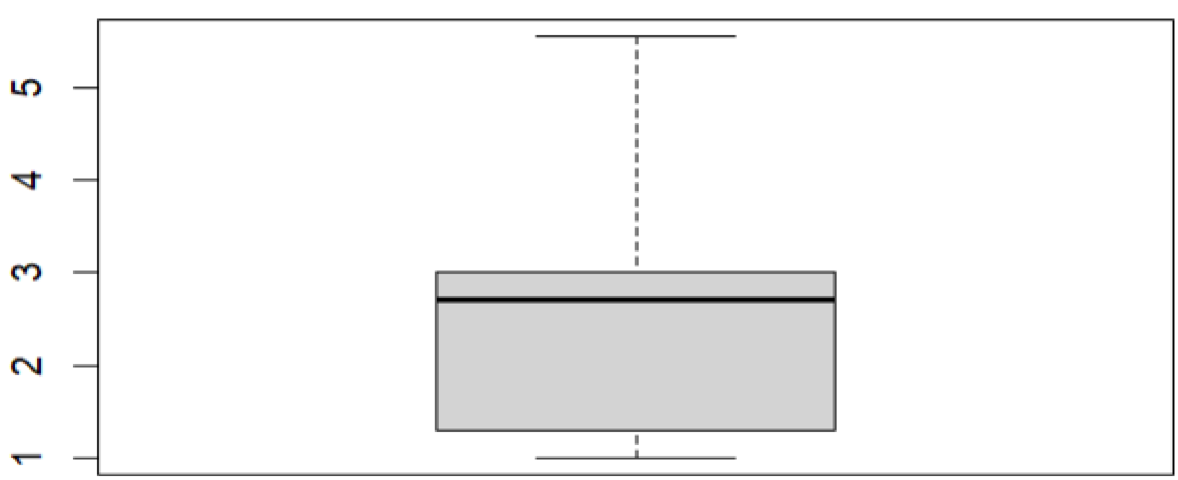
> boxplot(listing$maximum_nights)



9. $ minimum_minimum_nights
10. $ maximum_minimum_nights
11. $ minimum_maximum_nights
12. $ maximum_maximum_nights
13. $ minimum_nights_avg_ntm

> listing = clean_column(listing, "minimum_nights_avg_ntm", min_val = 0, max_val = 6)

> boxplot(listing$minimum_nights_avg_ntm)



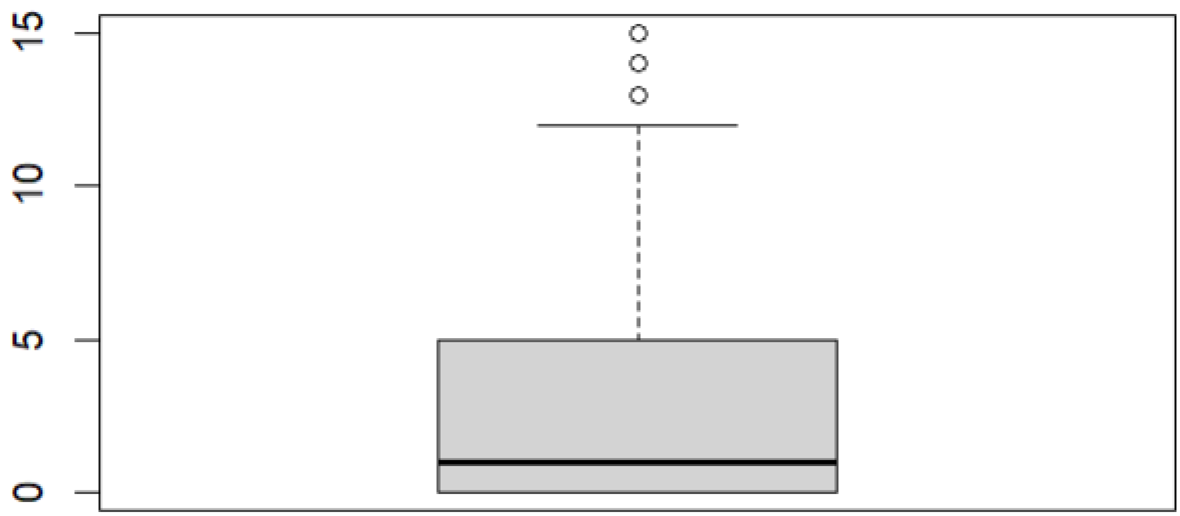
14. $ maximum_nights_avg_ntm
    listing$maximum_nights_avg_ntm=NULL

15. $ availability_30
16. $ availability_60
17. $ availability_90
18. $ number_of_reviews_ltm

> listing = clean_column(listing, "number_of_reviews_ltm", min_val = 0, max_val = 50)

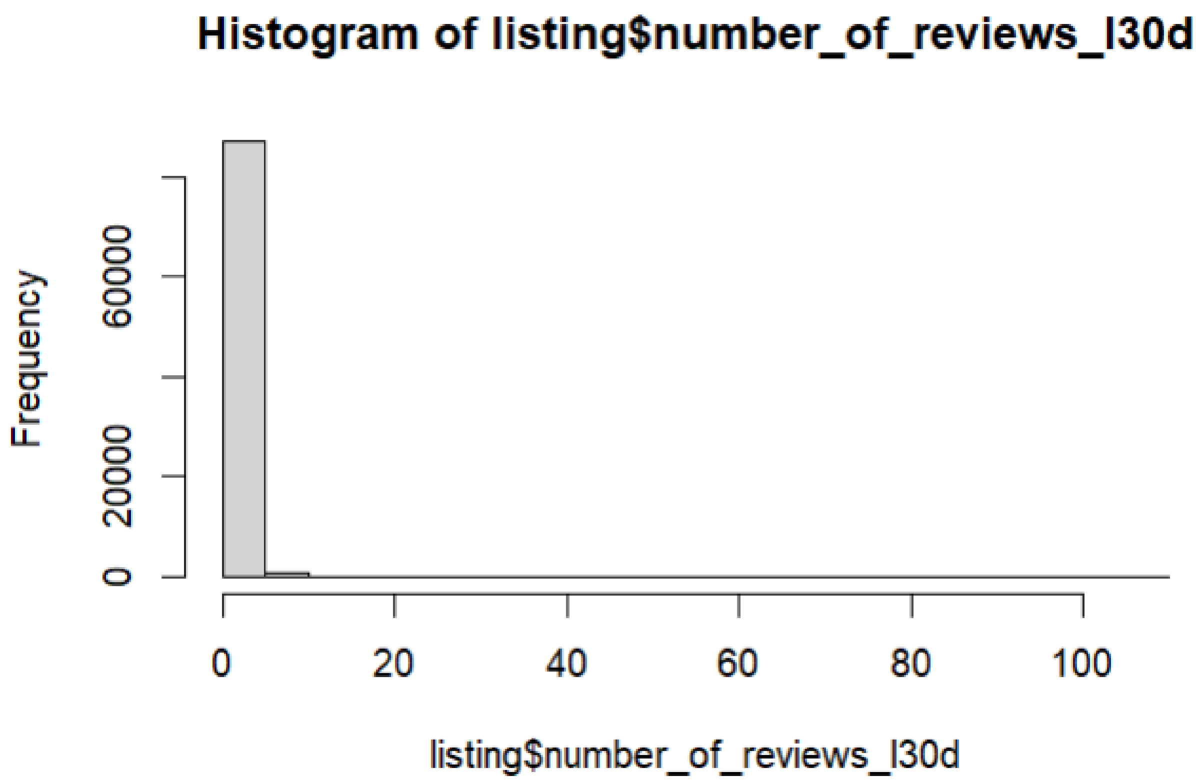
> hist(listing$number_of_reviews_ltm)

> boxplot(listing$number_of_reviews_ltm)



19. $ number_of_reviews_l30d

```
> hist(listing$number_of_reviews_l30d)
>
> sum(listing$number_of_reviews_l30d>10)
[1] 71
> sum(listing$number_of_reviews_l30d>5)
[1] 969
> sum(listing$number_of_reviews_l30d>6)
[1] 523
> sum(listing$number_of_reviews_l30d>2)
[1] 5803
> sum(listing$number_of_reviews_l30d>2)
[1] 5803
> sum(listing$number_of_reviews_l30d>1)
[1] 10782
> sum(listing$number_of_reviews_l30d<1)
[1] 67365
>
```



20. $ review_scores_accuracy
21. $ review_scores_cleanliness’
22. $ review_scores_checkin
23. $ review_scores_communication
24. $ review_scores_value
25. $ calculated_host_listings_count
26. $ calculated_host_listings_count_entire_homes
27. $ calculated_host_listings_count_private_room
28. $ host_engaged_years
29. $ listing_is_good

Now lets drop the correlated data among above 29 columns :

```
> cor(listing_importedbackup[,c("host_listings_count","host_total_listings_count")], use = "complete.obs", method = "pearson")
                          host_listings_count host_total_listings_count
host_listings_count                 1.0000000                 0.9869852
host_total_listings_count           0.9869852                 1.0000000
>
```

Here, from the table, I can see that host_listing_count and host_total_listing_count are highly correlated. So I will drop the host_listing_count and take only host_listing_count as it shows only the active listings which is essential for the Airbnb.

cor(listing[, c("minimum_nights","maximum_nights","minimum_minimum_nights","maximum_minimum_nights","minimum_maximum_nights","maximum_maximum_nights","minimum_nights_avg_ntm","maximum_nights_avg_ntm")], use = "complete.obs", method = "pearson")

```
> cor(listing[, c("minimum_nights","maximum_nights","minimum_minimum_nights","maximum_minimum_nights","minimum_maximum_nights","maximum_maximum_nights","minimum_nights_avg_ntm","maximum
_nights_avg_ntm")], use = "complete.obs", method = "pearson")
                       minimum_nights maximum_nights minimum_minimum_nights maximum_minimum_nights minimum_maximum_nights maximum_maximum_nights minimum_nights_avg_ntm
minimum_nights           1.0000000000  -0.0006589026           0.9116381665           0.6872055687           -0.003005489           -0.003005515           0.7512429375
maximum_nights          -0.0006589026   1.0000000000          -0.0006214875          -0.0008233416            0.054543589            0.054543587          -0.0007256602
minimum_minimum_nights   0.9116381665  -0.0006214875           1.0000000000           0.6976062698           -0.002841607           -0.002841665           0.7727470020
maximum_minimum_nights   0.6872055687  -0.0008233416           0.6976062698           1.0000000000           -0.001124964           -0.001124779           0.9624746834
minimum_maximum_nights  -0.0030054892   0.0545435885          -0.0028416070          -0.0011249641            1.000000000            1.000000000          -0.0028152811
maximum_maximum_nights  -0.0030055147   0.0545435871          -0.0028416654          -0.0011247794            1.000000000            1.000000000          -0.0028152064
minimum_nights_avg_ntm   0.7512429375  -0.0007256602           0.7727470020           0.9624746834           -0.002815281           -0.002815206           1.0000000000
maximum_nights_avg_ntm  -0.0030055062   0.0545435876          -0.0028416440          -0.0011248689            1.000000000            1.000000000          -0.0028152495
                       maximum_nights_avg_ntm
minimum_nights                   -0.003005506
maximum_nights                    0.054543588
minimum_minimum_nights           -0.002841644
maximum_minimum_nights           -0.001124869
minimum_maximum_nights            1.000000000
maximum_maximum_nights            1.000000000
minimum_nights_avg_ntm           -0.002815249
maximum_nights_avg_ntm            1.000000000
>
```

From the table we can see that,

minimum_maximum_nights and maximum_maximum_nights are perfectly correrelated. Thus I can keep any one of them and drop the remaining.

Also, minimum_maximum_nights and maximum_nights_avg_ntm are perfectly related.

Thus I can keep any one of them and drop the remaining.

Also, maximum_maximum_nights and maximum_nights_avg_ntm are perfectly related so, I can keep any one of them.

Again, maximum_minimum_nights and minimum_nights_avg_ntm are very highly correlated (0.96) so I can keep one of them.

Also, minimum_nights and minimum_minimum_nights have the high correrelation.

Also, maximum_nights_avg_ntm has perfect correrelation with maximum_maximum_nights and minimum_maximum_nights. so I will keep maximum_nights_avg_ntm out of them.

After analysing the data, I came to the conclusion that. I will be keeping:

Minimum_nights [as it is more general]

Maximum_nights[as it is not highly correlated with any of the data set]

Minimum_nights_avg_ntm [taking out of two as it is the average also.]

Maximum_nights_avg_ntm

And dropping the remaining columns.

```
> listing$minimum_minimum_nights=NULL
> listing$maximum_maximum_nights=NULL
> listing$maximum_minimum_nights=NULL
> listing$minimum_maximum_nights=NULL
> 
```

Code:

cor(listing[,c("number_of_reviews_ltm","number_of_reviews_l30d","review_scores_accuracy","review_scores_cleanliness","review_scores_checkin","review_scores_communication","review_scores_value")], use = "complete.obs", method = "pearson")

```
> cor(listing[, c("number_of_reviews_ltm","number_of_reviews_l30d","review_scores_accuracy","review_scores_cleanliness","review_scores_checkin","review_scores_communication","review_sco
res_value")], use = "complete.obs", method = "pearson")
                            number_of_reviews_ltm number_of_reviews_l30d review_scores_accuracy review_scores_cleanliness review_scores_checkin review_scores_communication
number_of_reviews_ltm                  1.00000000             0.62773096            -0.03153897                0.06474521            0.05121928                  0.01968352
number_of_reviews_l30d                 0.62773096             1.00000000            -0.02532928                0.06166245            0.04514420                  0.02005256
review_scores_accuracy                -0.03153897            -0.02532928             1.00000000                0.70117127            0.66190518                  0.29532059
review_scores_cleanliness              0.06474521             0.06166245             0.70117127                1.00000000            0.58520290                  0.25373291
review_scores_checkin                  0.05121928             0.04514420             0.66190518                0.58520290            1.00000000                  0.32247200
review_scores_communication            0.01968352             0.02005256             0.29532059                0.25373291            0.32247200                  1.00000000
review_scores_value                    0.04857445             0.04799568             0.78100639                0.72313465            0.66784861                  0.29957449
                            review_scores_value
number_of_reviews_ltm                0.04857445
number_of_reviews_l30d               0.04799568
review_scores_accuracy               0.78100639
review_scores_cleanliness            0.72313465
review_scores_checkin                0.66784861
review_scores_communication          0.29957449
review_scores_value                  1.00000000
> 
```

Number_of _reviews_itm and number_of_reviews_l30d are 0.627 correlated. But these are different things so I would like to keep these both.

So lets drop number_of_reviews_itm to remove data redundancy.

review_scores_accuracy and review_scores_cleanliness are 0.70 correlated.

review_scores_accuracy and review_scores_checkin are 0.66 correlated.

review_scores_accuracy and review_scores_value are 0.78 correlated.

review_scores_accuracy and review_scores_communication are 0.295 correlated

Also,

review_scores_cleanliness and review_scores_checkin, review_scores_communication, review_scores_value are highly correlated.

Instead of droping I will create a mean called total_review_score of all these rating and remove these five columns.

Code:

listing$total_review_score = rowMeans(listing[, c("review_scores_accuracy", "review_scores_cleanliness","review_scores_checkin","review_scores_communication", "review_scores_value")],na.rm = TRUE)

```
> listing$total_review_score = rowMeans(listing[, c("review_scores_accuracy", "rev
iew_scores_cleanliness","review_scores_checkin","review_scores_communication",
"review_scores_value")],na.rm = TRUE)
> listing$review_scores_accuracy=NULL
> listing$review_scores_checkin=NULL
> listing$review_scores_cleanliness=NULL
> listing$review_scores_communication=NULL
> listing$review_scores_value=NULL
> |
```

Next step:

```
> cor(listing[, c("availability_30","availability_60","availability_90"
+ )], use = "complete.obs", method = "pearson")
                availability_30 availability_60 availability_90
availability_30       1.0000000       0.9394774       0.8831743
availability_60       0.9394774       1.0000000       0.9755635
availability_90       0.8831743       0.9755635       1.0000000
> 
```

From the table we can see that,

Availability_30 and availability_60 are 0.93 correlated.

availability_30 and availability_90 are 0.88 correlated.

availability_60 and availability_90 are 0.975 correlated.

Since, availability_60 is highly correlated with both the columns, I will keep this column and drop the remaining.

```
> listing$availability_30=NULL
> listing$availability_90=NULL
> |
```

I will scale the columns that will be used to derrive the dependent variable first.

lets make a generalised function so that I can fill the reduce the outliers and extreme values.

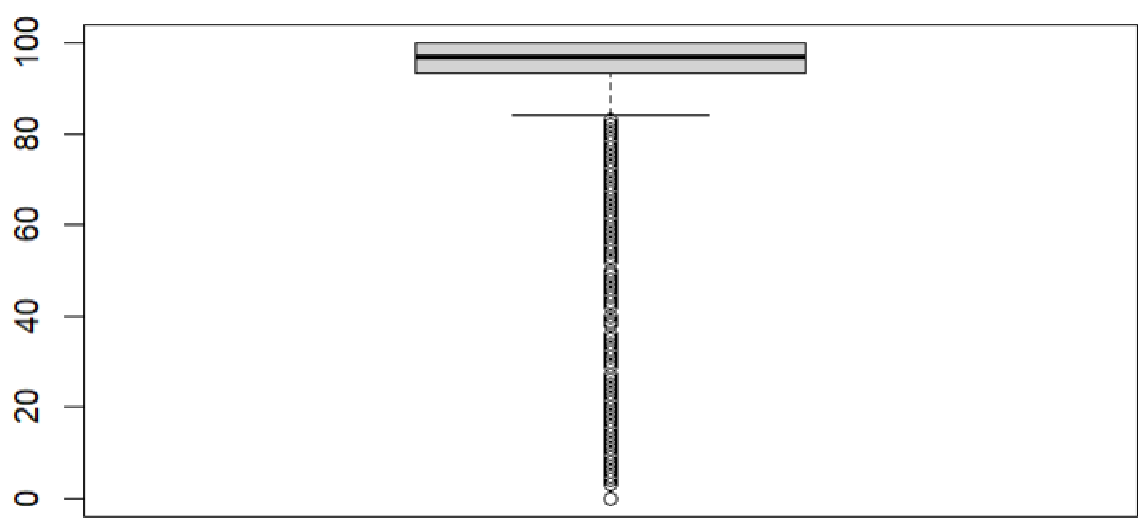
I will be using IQR-Based Outlier Capping method to handle the outliers.
It will help me to replace the extreme values with boundry values.

```
# --- Generalized Function to Clean Outliers
clean_column = function(data, column, min_val = -Inf, max_val = Inf) {
  median_val = median(data[[column]], na.rm = TRUE)
  Q1 = quantile(data[[column]], 0.25, na.rm = TRUE)
  Q3 = quantile(data[[column]], 0.75, na.rm = TRUE)
  IQR = Q3 - Q1
  lower_bound = Q1 - 1.5 * IQR
  upper_bound = Q3 + 1.5 * IQR

  data[[column]] = case_when(
    is.na(data[[column]]) |
    data[[column]] < min_val |
    data[[column]] > max_val ~ median_val,
    data[[column]] < lower_bound ~ pmax(lower_bound, min_val),
    data[[column]] > upper_bound ~ pmin(upper_bound, max_val),
    TRUE ~ data[[column]]
  )

  return(data)}
```

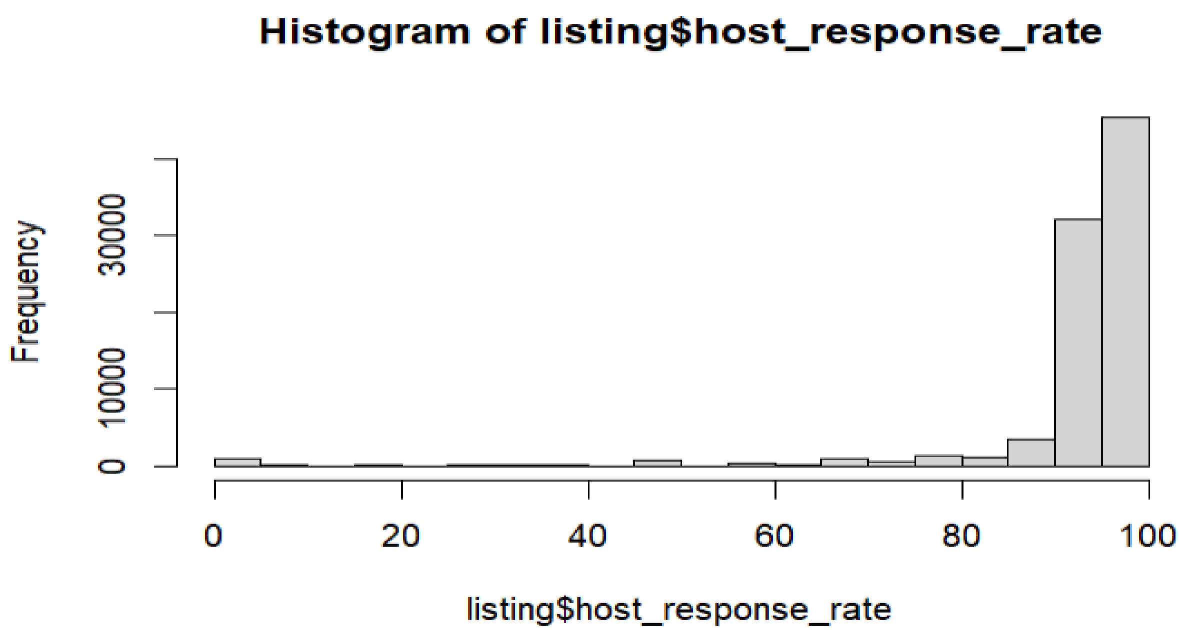
```
# dataset_name = clean_column(dataset_name, "Variable_name", min_val = x, max_val = y)
```

```
> # --- Generalized Function to Clean Outliers
> clean_column = function(data, column, min_val = -Inf, max_val = Inf) {
+     median_val = median(data[[column]], na.rm = TRUE)
+     Q1 = quantile(data[[column]], 0.25, na.rm = TRUE)
+     Q3 = quantile(data[[column]], 0.75, na.rm = TRUE)
+     IQR = Q3 - Q1
+     lower_bound = Q1 - 1.5 * IQR
+     upper_bound = Q3 + 1.5 * IQR
+
+     data[[column]] = case_when(
+         is.na(data[[column]]) |
+             data[[column]] < min_val |
+             data[[column]] > max_val ~ median_val,
+         data[[column]] < lower_bound ~ pmax(lower_bound, min_val),
+         data[[column]] > upper_bound ~ pmin(upper_bound, max_val),
+         TRUE ~ data[[column]]
+     )
+
+     return(data)}
>
```

1. host_response_rate



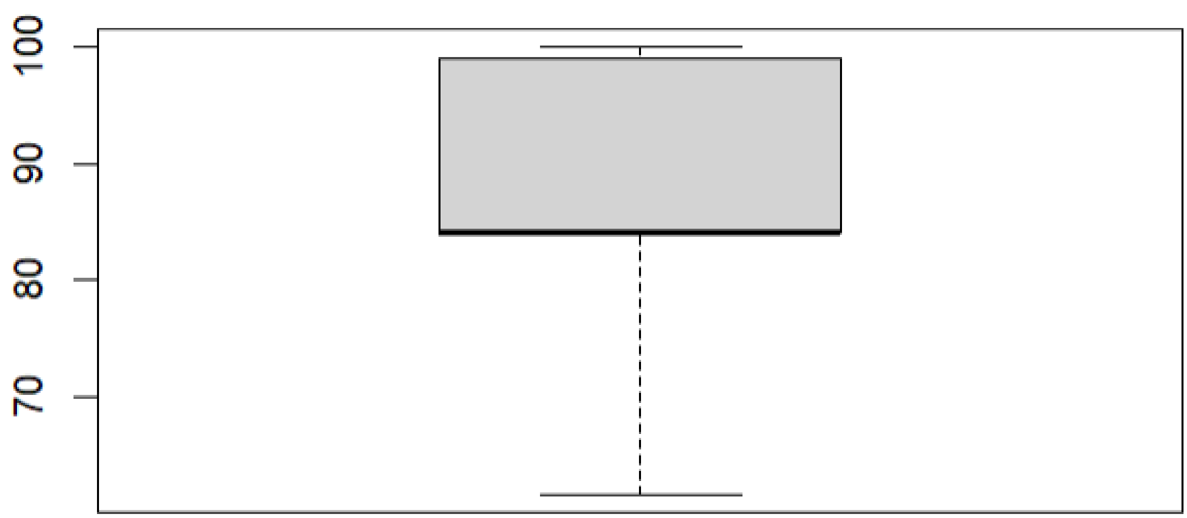

```
> boxplot(listing$host_response_rate)
> sum(listing$host_response_rate<80)
[1] 5217
> sum(listing$host_response_rate<75)
[1] 4444
```



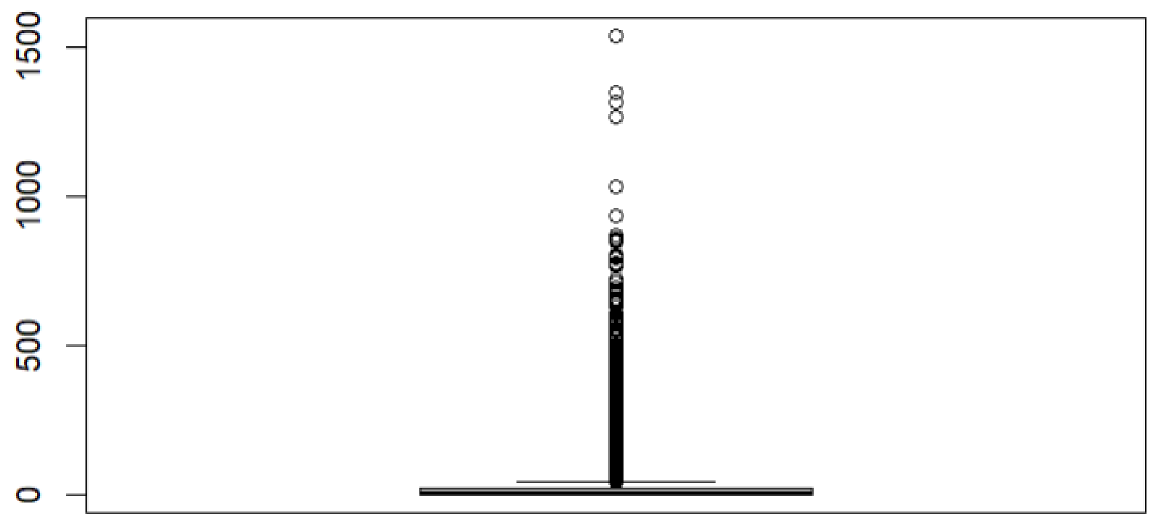

```
> hist(listing$host_response_rate)
> listing = clean_column(listing, "host_response_rate", min_val = 80, max_val = 100)
> boxplot(airbnb_data$host_acceptance_rate)
```

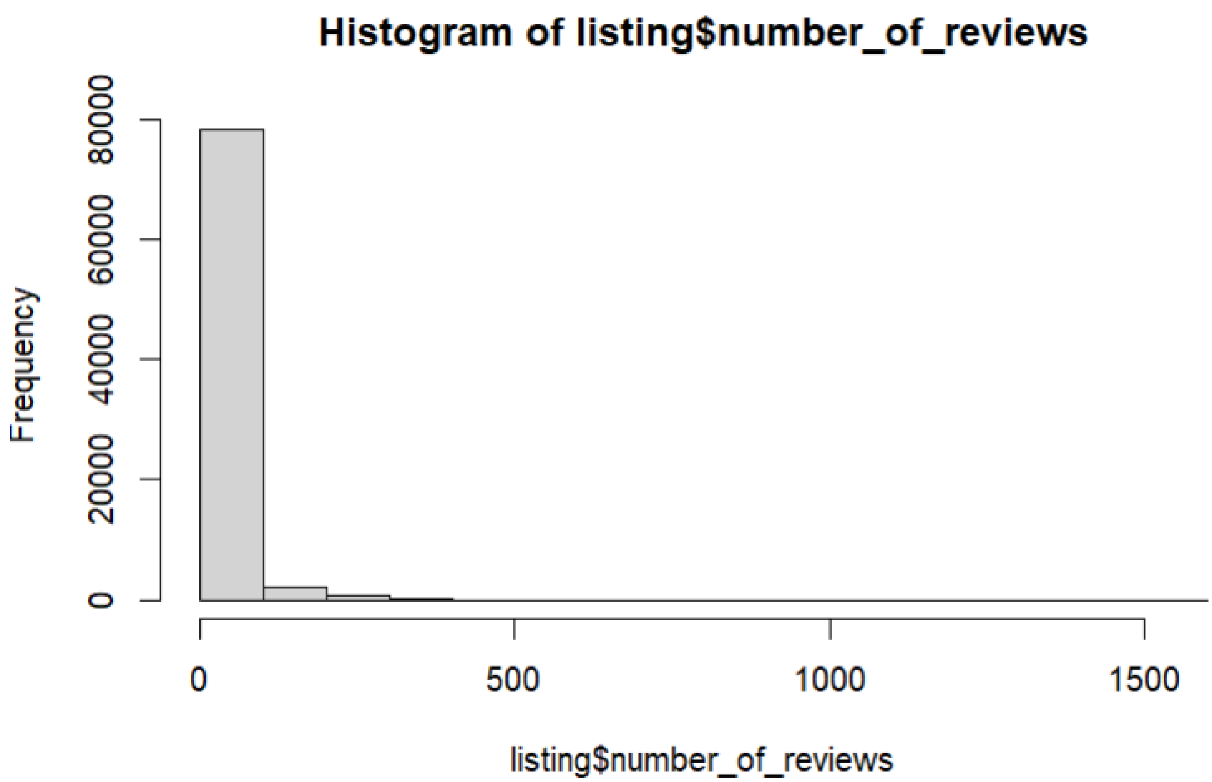
Code:

listing = clean_column(listing, "host_response_rate", min_val = 80, max_val = 100)

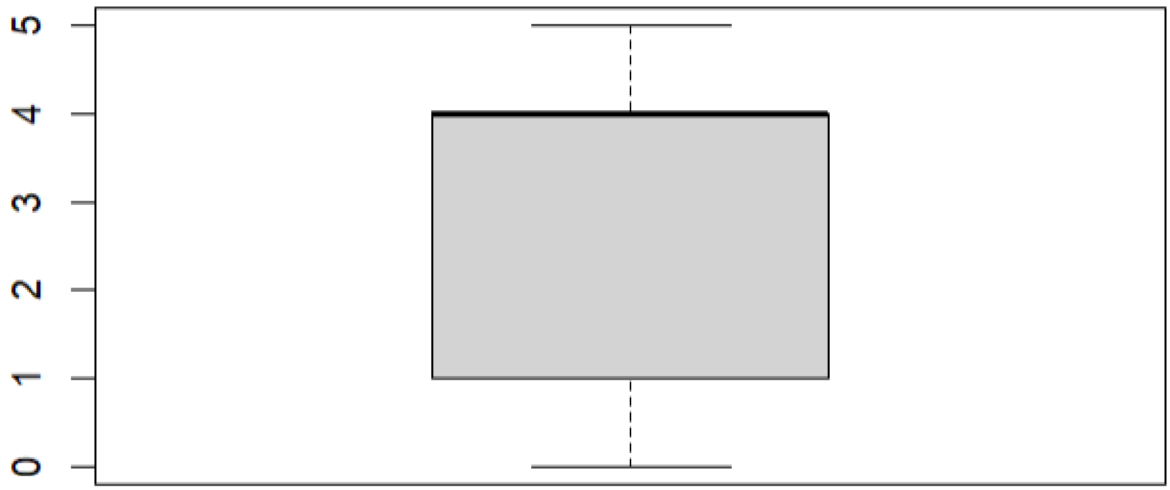


2. number_of_reviews

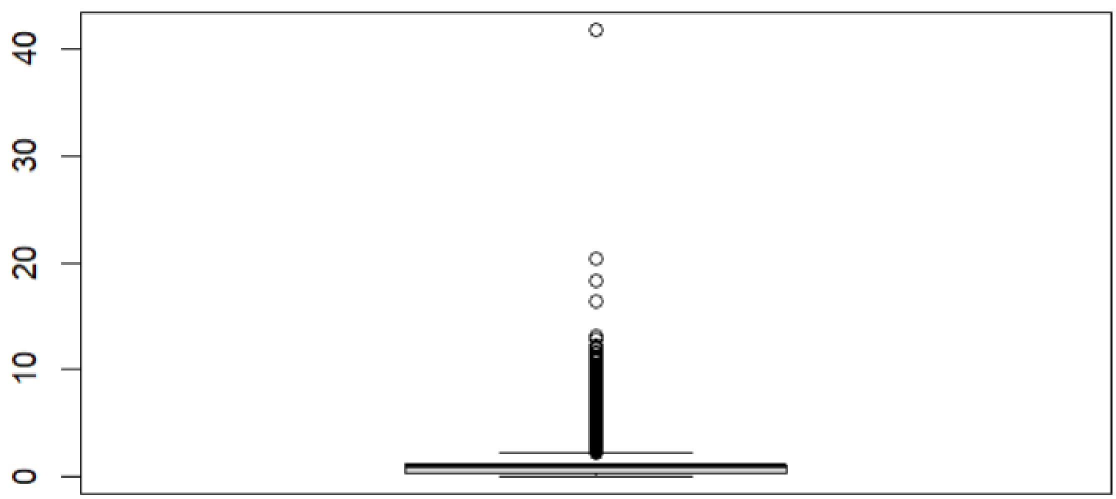


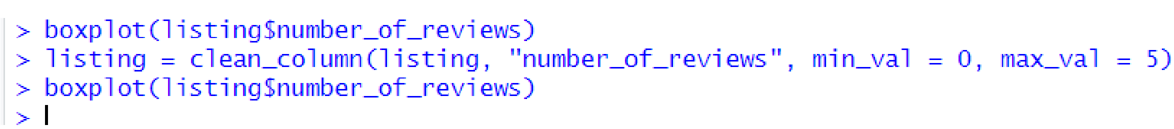


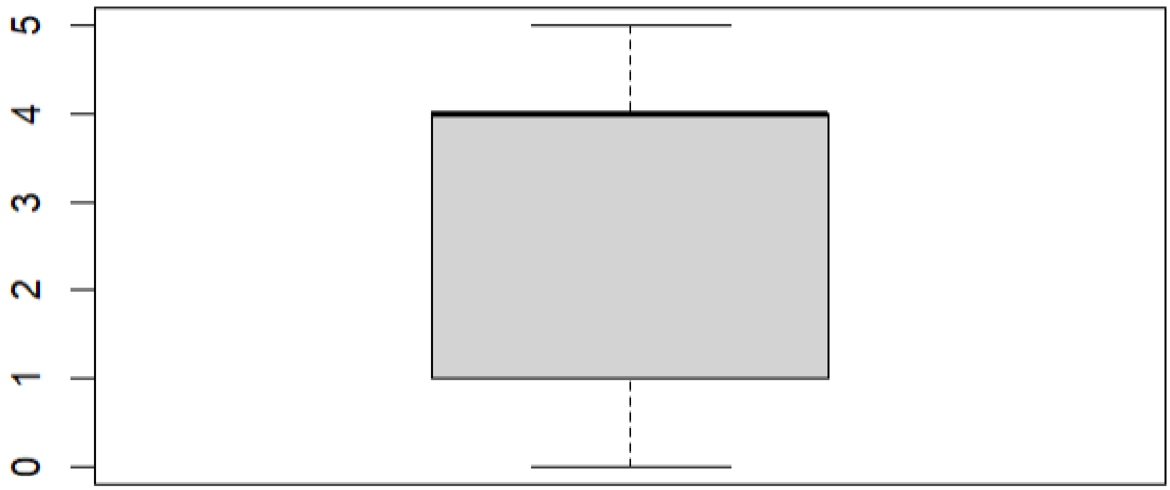

```
> boxplot(listing$number_of_reviews)
> listing = clean_column(listing, "number_of_reviews", min_val = 0, max_val = 5)
> boxplot(listing$number_of_reviews)
>
```



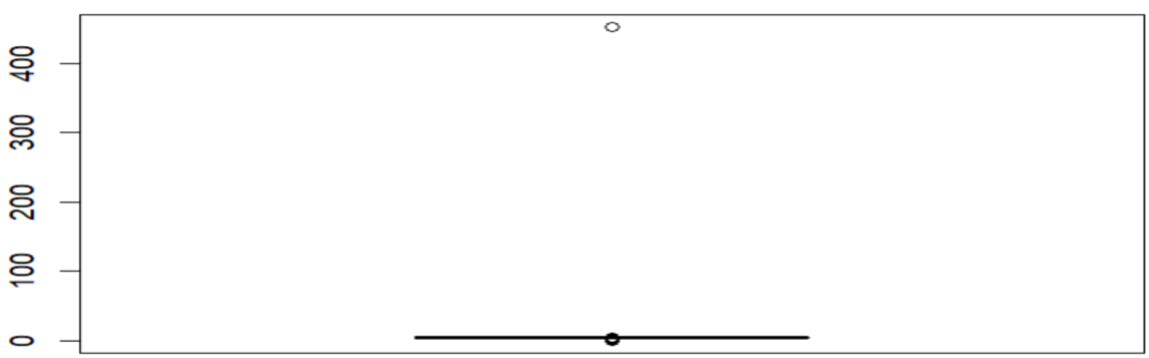
3. reviews_per_month







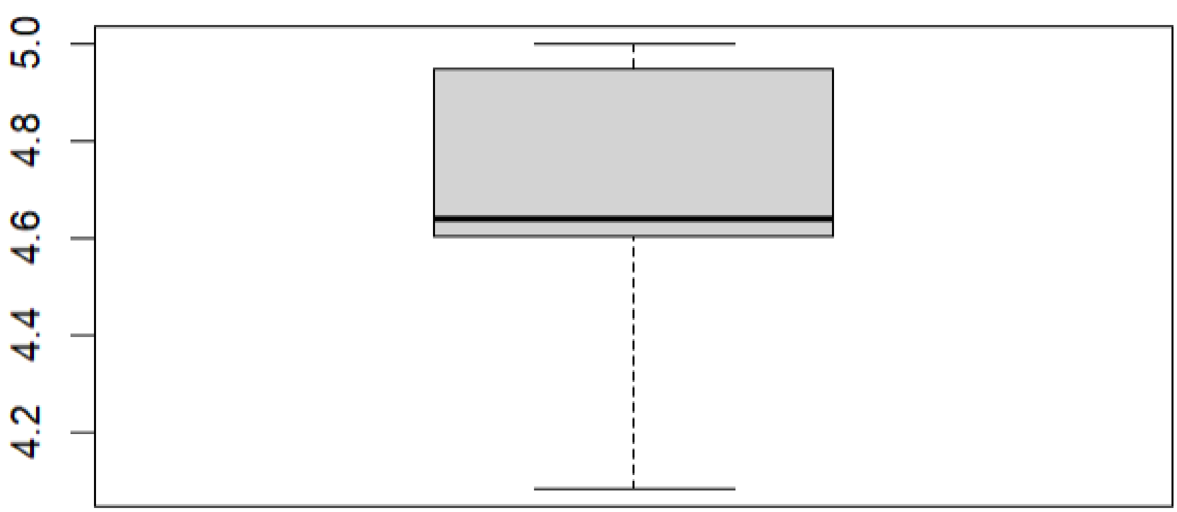
4. review_scores_rating



```
> boxplot(listing$number_of_reviews)
> listing = clean_column(listing, "number_of_reviews", min_val = 0, max_val = 5)
> boxplot(listing$number_of_reviews)
> boxplot(listing$review_scores_rating)
> hist(listing$review_scores_rating)
> sum(listing$review_scores_rating>5)
[1] 1
> sum(listing$review_scores_rating>4)
[1] 80894
> sum(listing$review_scores_rating>4.9)
[1] 25291
> listing = clean_column(listing, "review_scores_rating", min_val = 0, max_val = 5)
> boxplot(listing$review_scores_rating)
> |
```

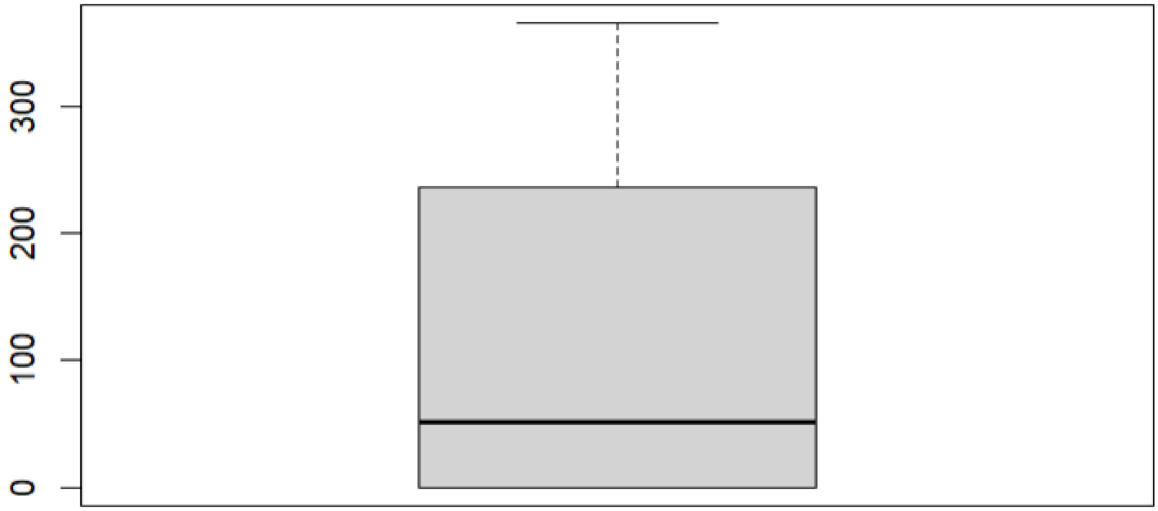
listing=listing[listing$review_scores_rating>4.2,]

> boxplot(listing$review_scores_rating)

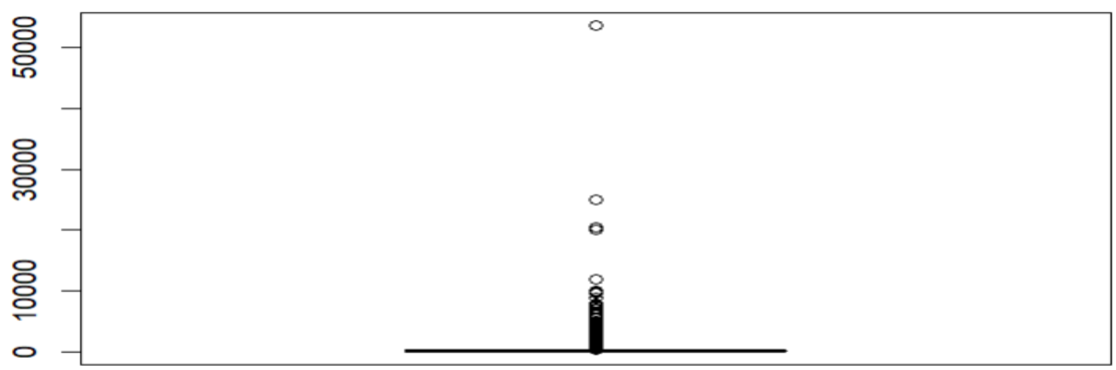


5. availability_365

boxplot(listing$availability_365)

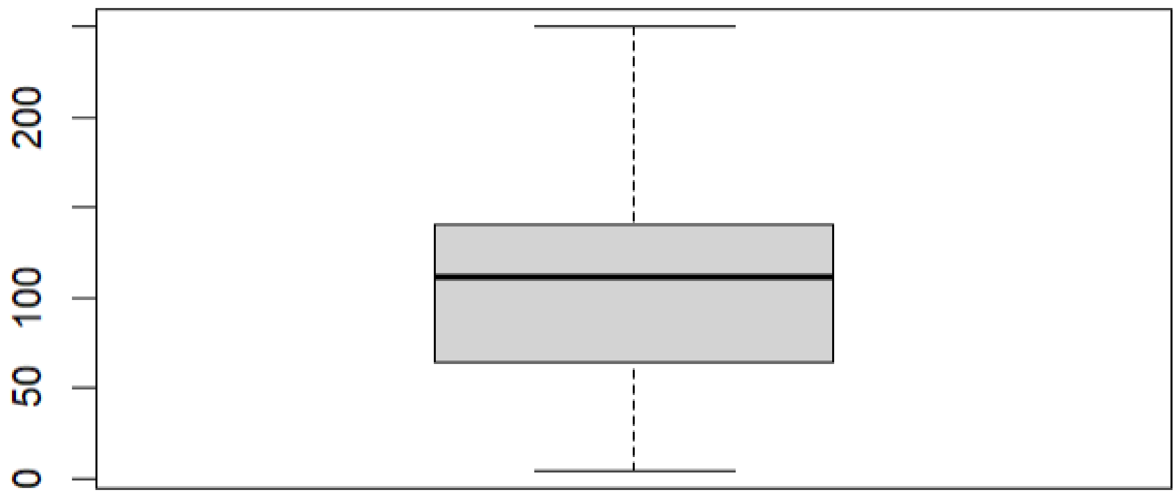


6. Price

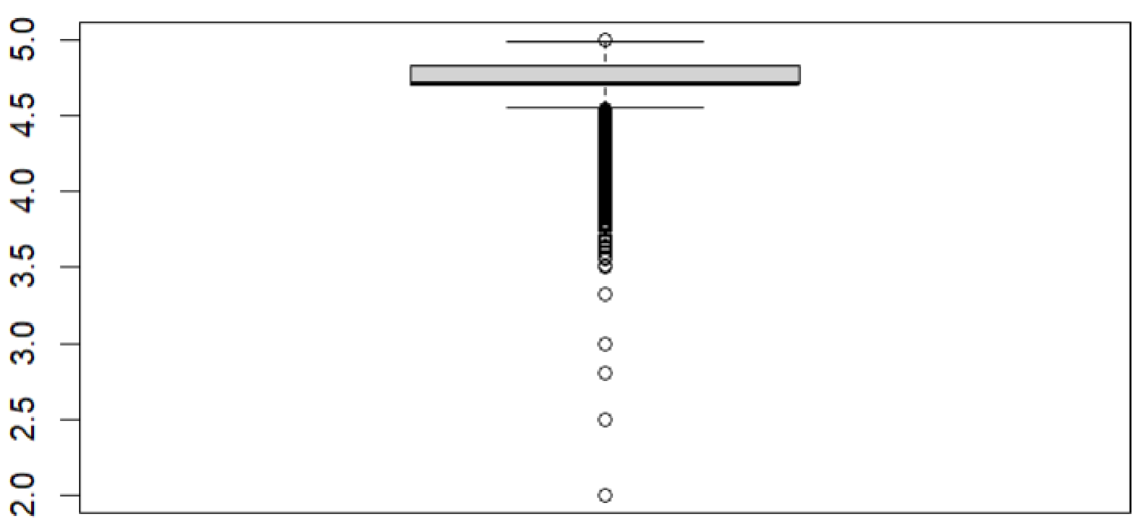



```
> listing=listing[listing$price<500,]
> boxplot(listing$price)
> listing$price=log1p(listing$price)
> boxplot(listing$price)
> sum(listing$price<3)
[1] 259
> listing=listing[listing$price>3,]
> boxplot(listing$price)
> 
```

```
> boxplot(listing$price)
> listing = clean_column(listing, "price", min_val = 3, max_val = 500)
> boxplot(listing$price)
> listing = clean_column(listing, "price", min_val = 3, max_val = 300)
> boxplot(listing$price)
> listing = clean_column(listing, "price", min_val = 3, max_val = 250)
> boxplot(listing$price)
```

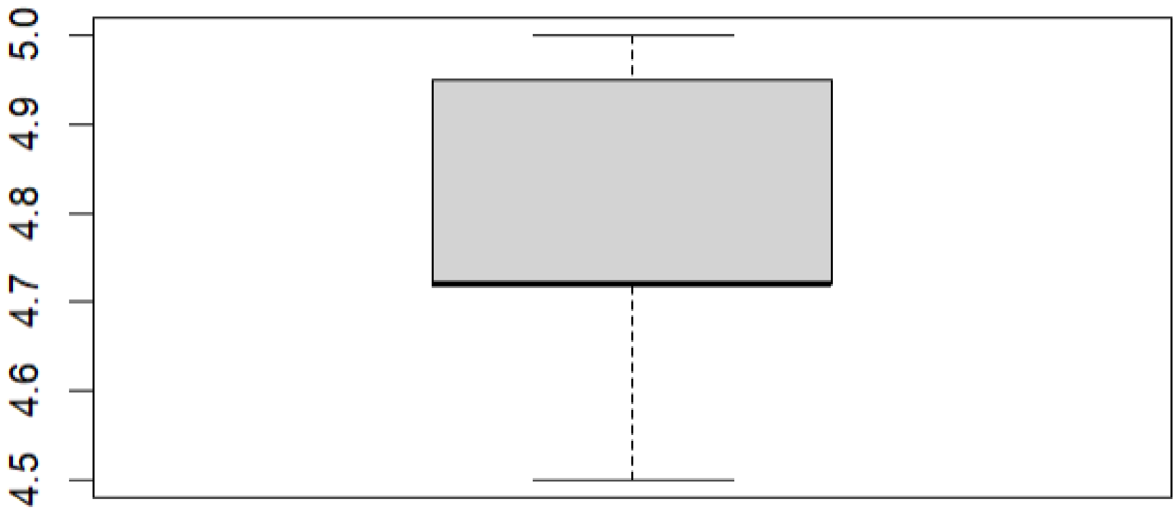


7. listing$review_scores_location



```
> listing=listing[listing$review_scores_location>4.5,]
> boxplot(listing$review_scores_location)
>
```

```
listing = clean_column(listing, "review_scores_location", min_val = 4.5, max_val = 5)
boxplot(listing$review_scores_location)
```



Now for derriving the new dependent column named : listing_is_good

I will check the summary first.

```
> summary(listing)
 host_response_rate host_acceptance_rate host_listings_count host_total_listings_count   room_type      accommodates          beds             price
 Min.    : 83.64    Min.    :  0.00       Min.    :   1.00     Min.    :   1.00            Min.    :1.000   Min.    : 1.000   Min.    : 1.000   Min.    :  4.0
 1st Qu.: 93.46     1st Qu.: 84.02        1st Qu.:   1.00      1st Qu.:   1.00             1st Qu.:2.000    1st Qu.: 2.000    1st Qu.: 1.000    1st Qu.: 64.0
 Median : 97.00     Median : 84.02        Median :   2.00      Median :   3.00             Median :3.000    Median : 2.000    Median : 1.000    Median :111.0
 Mean   : 96.54     Mean   : 84.02        Mean   :  48.44      Mean   :  86.65             Mean   :2.624    Mean   : 3.168    Mean   : 1.783    Mean   :107.8
 3rd Qu.:100.00     3rd Qu.: 99.00        3rd Qu.:   7.00      3rd Qu.:  12.00             3rd Qu.:3.000    3rd Qu.: 4.000    3rd Qu.: 2.000    3rd Qu.:140.0
 Max.   :100.00     Max.   :100.00        Max.   :3023.00      Max.   :5272.00             Max.   :3.000    Max.   :16.000    Max.   :50.000    Max.   :250.0
 minimum_nights      maximum_nights      minimum_nights_avg_ntm maximum_nights_avg_ntm availability_30  availability_60 availability_90 availability_365
 Min.   :   1.000   Min.   :        1   Min.   :   1.00     Min.   :1.000e+00     Min.   : 0.000   Min.   : 0.00   Min.   : 0.00   Min.   :  0.0
 1st Qu.:   1.000   1st Qu.:       60   1st Qu.:   1.30     1st Qu.:1.800e+02     1st Qu.: 0.000   1st Qu.: 0.00   1st Qu.: 0.00   1st Qu.:  0.0
 Median :   2.000   Median :      365   Median :   2.70     Median :7.310e+02     Median : 1.000   Median : 6.00   Median :13.00   Median : 63.0
 Mean   :   5.624   Mean   :     6794   Mean   :   7.87     Mean   :5.018e+05     Mean   : 7.733   Mean   :18.47   Mean   :31.01   Mean   :119.8
 3rd Qu.:   4.000   3rd Qu.:     1125   3rd Qu.:   5.00     3rd Qu.:1.125e+03     3rd Qu.:14.000   3rd Qu.:36.00   3rd Qu.:64.00   3rd Qu.:249.0
 Max.   :1125.000   Max.   :524855552   Max.   :1125.00     Max.   :2.147e+09     Max.   :30.000   Max.   :60.00   Max.   :90.00   Max.   :365.0
 number_of_reviews number_of_reviews_ltm number_of_reviews_l30d review_scores_rating review_scores_location calculated_host_listings_count
 Min.   :0.000     Min.   :  0.000      Min.   :  0.0000      Min.   :4.084       Min.   :4.500          Min.   :  1.00
 1st Qu.:1.000     1st Qu.:  0.000      1st Qu.:  0.0000      1st Qu.:4.604       1st Qu.:4.720          1st Qu.:  1.00
 Median :4.000     Median :  1.000      Median :  0.0000      Median :4.640       Median :4.720          Median :  2.00
 Mean   :2.529     Mean   :  5.732      Mean   :  0.5073      Mean   :4.685       Mean   :4.801          Mean   : 18.25
 3rd Qu.:4.000     3rd Qu.:  6.000      3rd Qu.:  0.0000      3rd Qu.:4.950       3rd Qu.:4.950          3rd Qu.:  6.00
 Max.   :5.000     Max.   :640.000      Max.   :109.0000      Max.   :5.000       Max.   :5.000          Max.   :595.00
 calculated_host_listings_count_entire_homes calculated_host_listings_count_private_rooms reviews_per_month host_engaged_years  listing_is_good
 Min.   :  0.00                              Min.   :  0.000                              Min.   : 0.010    Min.   : 0.005479   Min.   :0.00000
 1st Qu.:  0.00                              1st Qu.:  0.000                              1st Qu.: 0.260    1st Qu.: 4.276712   1st Qu.:0.00000
 Median :  1.00                              Median :  0.000                              Median : 1.000    Median : 7.136986   Median :0.00000
 Mean   : 13.92                              Mean   :  4.173                              Mean   : 1.018    Mean   : 6.526133   Mean   :0.02954
 3rd Qu.:  3.00                              3rd Qu.:  1.000                              3rd Qu.: 1.018    3rd Qu.: 8.882192   3rd Qu.:0.00000
 Max.   :312.00                              Max.   :285.000                              Max.   :50.250    Max.   :14.767123   Max.   :1.00000
>
```

Code:

listing$listing_is_good=with(listing,as.logical(listing$host_response_rate>90 &

listing$review_scores_rating>4.6&

listing$availability_365>60&

listing$review_scores_location>=4.6

))

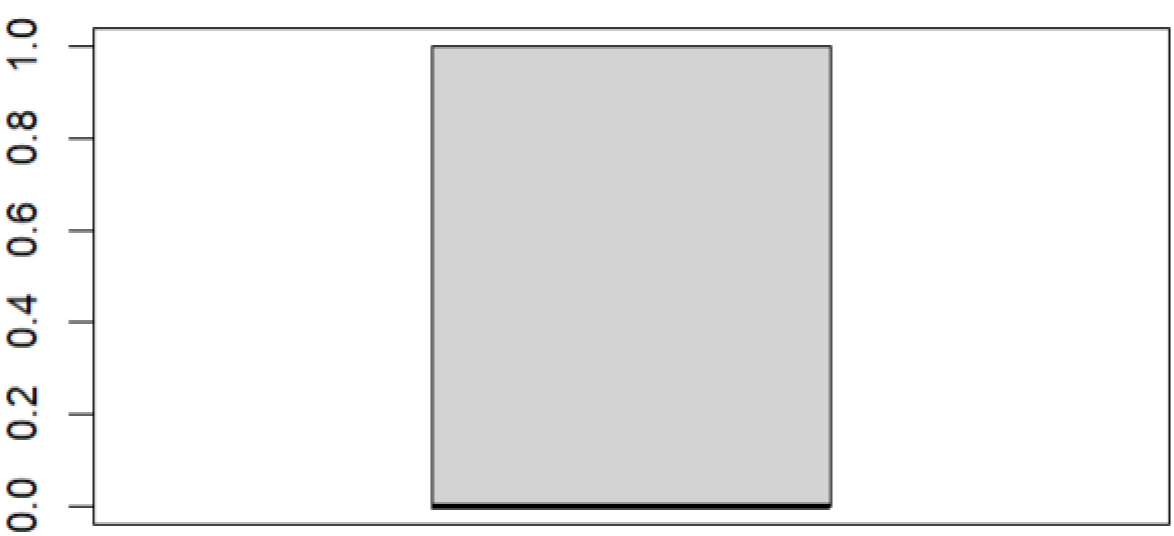
The above code gives the logical value which needs to be converted into numeric value so that I can convert it into factoral value later for my model.

Code: listing$listing_is_good=ifelse(listing$listing_is_good=="TRUE", yes = 1, no=0)

I will convert this column into categorical factors:

Code: listing$listing_is_good=as.factor(listing$listing_is_good)

```
> listing$listing_is_good=with(listing,as.logical(listing$host_response_rate>95 &
+                                                  listing$number_of_reviews>2&
+                                                  listing$reviews_per_month>2 &
+                                                  listing$review_scores_rating>4.7&
+                                                  listing$availability_365>120&
+                                                  listing$review_scores_location>=4.6
+ ))
>
> listing$listing_is_good=ifelse(listing$listing_is_good=="TRUE", yes = 1, no=0)
> listing$listing_is_good=as.factor(listing$listing_is_good)
> length(listing$listing_is_good)
[1] 88044
```



Now we cannot use the independent variable that has been used to derive the dependent variable called listing_is_good. If I do so, it will directly influence the results and may lead to 100% accuracy of my model which is not good in terms of modeling the machine learning model. So I will drop the independent columns that has been used to derive the dependent variable.

listing$host_response_rate=NULL

listing$review_scores_rating=NULL

listing$review_scores_location=NULL

listing$availability_365=NULL

```
> summary(listing)
 host_acceptance_rate host_listings_count host_total_listings_count   room_type      accommodates   
 Min.   :  0.00       Min.   :1.000       Min.   :1.000             Min.   :1.000   Min.   : 1.000  
 1st Qu.: 84.02       1st Qu.:1.000       1st Qu.:1.000             1st Qu.:2.000   1st Qu.: 2.000  
 Median : 84.02       Median :2.000       Median :3.000             Median :3.000   Median : 2.000  
 Mean   : 84.02       Mean   :1.663       Mean   :2.421             Mean   :2.624   Mean   : 3.168  
 3rd Qu.: 99.00       3rd Qu.:2.000       3rd Qu.:3.000             3rd Qu.:3.000   3rd Qu.: 4.000  
 Max.   :100.00       Max.   :3.500       Max.   :6.000             Max.   :3.000   Max.   :16.000  
      beds            price         minimum_nights maximum_nights  minimum_nights_avg_ntm availability_30
 Min.   : 1.000   Min.   :  4.0   Min.   :1.00   Min.   :2.457   Min.   :1.00          Min.   : 0.000  
 1st Qu.: 1.000   1st Qu.: 64.0   1st Qu.:1.00   1st Qu.:5.198   1st Qu.:1.30          1st Qu.: 0.000  
 Median : 1.000   Median :111.0   Median :2.00   Median :6.596   Median :2.70          Median : 1.000  
 Mean   : 1.783   Mean   :107.8   Mean   :2.32   Mean   :5.810   Mean   :2.45          Mean   : 7.733  
 3rd Qu.: 2.000   3rd Qu.:140.0   3rd Qu.:3.00   3rd Qu.:7.026   3rd Qu.:3.00          3rd Qu.:14.000  
 Max.   :50.000   Max.   :250.0   Max.   :6.00   Max.   :9.210   Max.   :5.55          Max.   :30.000  
 availability_60 availability_90 number_of_reviews number_of_reviews_ltm number_of_reviews_l30d
 Min.   : 0.00   Min.   : 0.00   Min.   :0.000     Min.   : 0.000        Min.   :  0.0000      
 1st Qu.: 0.00   1st Qu.: 0.00   1st Qu.:1.000     1st Qu.: 0.000        1st Qu.:  0.0000      
 Median : 6.00   Median :13.00   Median :4.000     Median : 1.000        Median :  0.0000      
 Mean   :18.47   Mean   :31.01   Mean   :2.529     Mean   : 3.434        Mean   :  0.5073      
 3rd Qu.:36.00   3rd Qu.:64.00   3rd Qu.:4.000     3rd Qu.: 5.000        3rd Qu.:  0.0000      
 Max.   :60.00   Max.   :90.00   Max.   :5.000     Max.   :15.000        Max.   :109.0000      
 calculated_host_listings_count calculated_host_listings_count_entire_homes
 Min.   :  1.00                 Min.   :  0.00                             
 1st Qu.:  1.00                 1st Qu.:  0.00                             
 Median :  2.00                 Median :  1.00                             
 Mean   : 18.25                 Mean   : 13.92                             
 3rd Qu.:  6.00                 3rd Qu.:  3.00                             
 Max.   :595.00                 Max.   :312.00                             
 calculated_host_listings_count_private_rooms reviews_per_month host_engaged_years  listing_is_good
 Min.   :  0.000                              Min.   : 0.010    Min.   : 0.005479   Mode :logical  
 1st Qu.:  0.000                              1st Qu.: 0.260    1st Qu.: 4.276712   FALSE:58318    
 Median :  0.000                              Median : 1.000    Median : 7.136986   TRUE :29726    
 Mean   :  4.173                              Mean   : 1.018    Mean   : 6.526133                  
 3rd Qu.:  1.000                              3rd Qu.: 1.018    3rd Qu.: 8.882192                  
 Max.   :285.000                              Max.   :50.250    Max.   :14.767123                  
> |
```

Final touch up with data types:

```
> str(listing)
'data.frame':	88044 obs. of  22 variables:
 $ host_acceptance_rate                        : num  100 25 88 41 75 ...
 $ host_listings_count                         : num  1 1 3 1 1 2 1 1 1 3 ...
 $ host_total_listings_count                   : num  1 2 4 3 1 3 3 3 2 3 ...
 $ room_type                                   : int  2 3 2 3 2 3 3 3 3 2 ...
 $ accommodates                                : num  2 5 1 2 2 6 4 3 4 2 ...
 $ beds                                        : int  2 3 1 1 1 3 3 1 1 1 ...
 $ price                                       : num  42 175 79 150 46 111 111 250 75 29 ...
 $ minimum_nights                              : num  2 5 1 2 4 3 5 2 2 2 ...
 $ maximum_nights                              : num  7.03 5.48 3.4 3.43 5.9 ...
 $ minimum_nights_avg_ntm                      : num  2 5 1 2.7 4 3 5 2 2 2.7 ...
 $ availability_30                             : int  0 13 25 7 5 14 26 0 0 0 ...
 $ availability_60                             : int  0 18 55 7 5 26 56 0 0 0 ...
 $ availability_90                             : int  0 38 85 7 5 50 86 0 0 0 ...
 $ number_of_reviews                           : num  4 4 4 4 4 4 4 4 4 4 ...
 $ number_of_reviews_ltm                       : num  9 2 11 5 15 4 3 0 0 0 ...
 $ number_of_reviews_l30d                      : int  0 0 0 0 3 0 0 0 0 0 ...
 $ calculated_host_listings_count              : num  1 1 2 1 1 9 1 1 1 3 ...
 $ calculated_host_listings_count_entire_homes : num  0 1 1 1 0 9 1 1 1 0 ...
 $ calculated_host_listings_count_private_rooms: num  1 0 1 0 1 0 0 0 0 3 ...
 $ reviews_per_month                           : num  1.45 0.27 0.26 0.56 1.21 0.36 0.16 0.62 0.64 0.79 ...
 $ host_engaged_years                          : num  12.4 12.4 13.8 13.8 12.4 ...
 $ listing_is_good                             : logi  FALSE FALSE TRUE TRUE FALSE TRUE ...
> listing$listing_is_good=as.factor(listing$listing_is_good)
> listing$room_type=as.numeric(listing$room_type)
> listing$beds=as.numeric(listing$beds)
> listing$availability_30=as.numeric(listing$availability_30)
> listing$availability_60=as.numeric(listing$availability_60)
> listing$availability_90=as.numeric(listing$availability_90)
> listing$number_of_reviews_l30d=as.numeric(listing$number_of_reviews_l30d)
> |
```

listing$number_of_reviews_l30d=NULL

> listing$availability_30=NULL

> listing$availability_90=NULL

Now, I will be deploying decision trees for analysing if the property is a good listing or not.

Code:

install.packages("tree")

library(tree)

#I will be using rsample to split the data.

install.packages("rsample")

library(rsample)

split_tree_data=initial_split(listing,prop = 0.7) #splitting the data as 70-30 ratio

train_tree_data=training(split_tree_data) #assigining the train data

test_tree_data=testing(split_tree_data) #assigning the test_data tree_model_good_notgood=tree(formula =listing_is_good~.,data=train_tree_data, control = tree.control(nobs = nrow(train_tree_data), mindev = 0.0001, mincut = 5, minsize = 10))

predict_good_notgood_train=predict(tree_model_good_notgood,train_tree_data,type = "class")

table(train_tree_data$listing_is_good,predict_good_notgood_train)

train_model_accuracy_tree=mean(train_tree_data$listing_is_good==predict_good_notgood_train)

train_model_accuracy_tree

predict_good_notgood_test=predict(tree_model_good_notgood,test_tree_data,type = "class")

table(test_tree_data$listing_is_good,predict_good_notgood_test)

test_model_accuracy_tree=mean(test_tree_data$listing_is_good==predict_good_notgood_test)

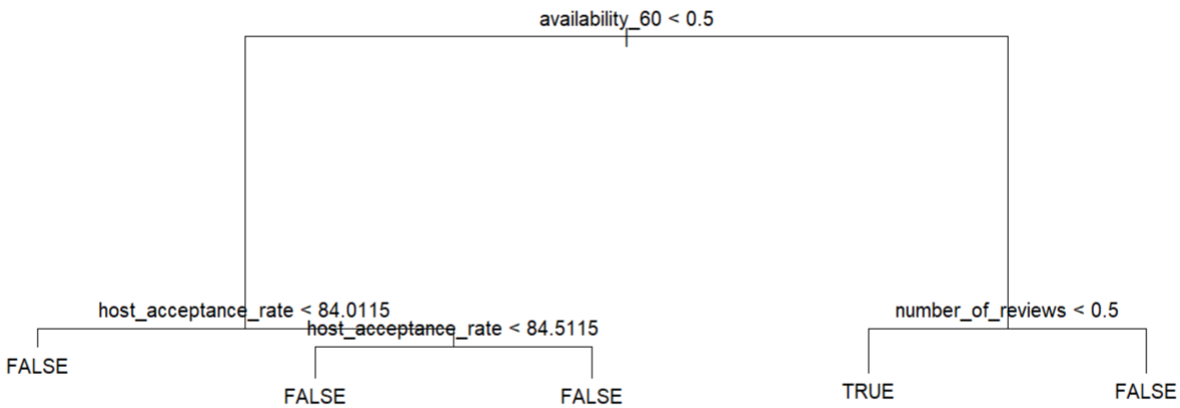
test_model_accuracy_tree

plot(tree_model_good_notgood)

text(tree_model_good_notgood)

```
> split_tree_data=initial_split(listing,prop = 0.7) #splitting the data as 70-30 ratio
> train_tree_data=training(split_tree_data) #assigining the train data
> test_tree_data=testing(split_tree_data) #assigning the test_data
> tree_model_good_notgood=tree(formula =listing_is_good~.,data=train_tree_data)
> predict_good_notgood_train=predict(tree_model_good_notgood,train_tree_data,type = "class")
> table(train_tree_data$listing_is_good,predict_good_notgood_train)
       predict_good_notgood_train
        FALSE  TRUE
  FALSE 39000  1871
  TRUE  14502  6257
> train_model_accuracy_tree=mean(train_tree_data$listing_is_good==predict_good_notgood_train)
> train_model_accuracy_tree
[1] 0.7343339
> predict_good_notgood_test=predict(tree_model_good_notgood,test_tree_data,type = "class")
>
> table(test_tree_data$listing_is_good,predict_good_notgood_test)
       predict_good_notgood_test
        FALSE  TRUE
  FALSE 16629   818
  TRUE   6260  2707
> test_model_accuracy_tree=mean(test_tree_data$listing_is_good==predict_good_notgood_test)
> test_model_accuracy_tree
[1] 0.732036
>
> |
```

```
> dim(test_tree_data)
[1] 26414    19
> dim(train_tree_data)
[1] 61630    19
> 
```



Now, I will split the data for that I need to install the library called “rsample”

install.packages("rsample")

library(rsample)

split_glm =initial_split(listing,prop = 0.7) #splitting the data as 70-30 ratio

traindata=training(split_glm)

testdata=testing(split_glm)

model1=glm(listing_is_good~.,family = binomial(link = "logit"), data = traindata)

traindata_predict=predict(model1,traindata,type = "response")

traindata_classified=ifelse(traindata_predict>0.5,yes = 1,no=0)

table(traindata$listing_is_good,traindata_classified)

trainmodel_accuracy=mean(traindata$listing_is_good==traindata_classified)

trainmodel_accuracy

testdata_predict=predict(model1,testdata,type = "response")

testdata_classified=ifelse(testdata_predict>0.5,yes = 1,no=0)

table(testdata$listing_is_good,testdata_classified)

testdata_accuracy=mean(testdata$listing_is_good== testdata_classified)

testdata_accuracy

```
> model1=glm(listing_is_good~.,family = binomial(link = "logit"), data = traindata )
> summary(model1)

Call:
glm(formula = listing_is_good ~ ., family = binomial(link = "logit"),
    data = traindata)
```

```
Number of Fisher Scoring iterations: 8

> traindata_predict=predict(model1,traindata,type = "response")
> traindata_classified=ifelse(traindata_predict>0.5,yes = 1,no=0)
> table(traindata$listing_is_good,traindata_classified)
   traindata_classified
        0     1
  0 57413   475
  1  3451   291
> trainmodel_accuracy=mean(traindata$listing_is_good==traindata_classified)
>
> trainmodel_accuracy
[1] 0.9362973
> testdata_predict=predict(model1,testdata,type = "response")
> testdata_classified=ifelse(testdata_predict>0.5,yes = 1,no=0)
> table(testdata$listing_is_good,testdata_classified)
   testdata_classified
        0     1
  0 24596   204
  1  1485   129
> testdata_accuracy=mean(testdata$listing_is_good==testdata_classified)
> testdata_accuracy
[1] 0.9360566
> |
```

```
> dim(testdata)
[1] 26414    30
> dim(traindata)
[1] 61630    30
> |
```

Here form the above two models, I can see that the model accuracy of decision tree is 73% on both train and test dataset and the model accuracy for GLM model is 93% for both train and test dataset. Though the model accuracy looks good in glm model what I found doubtful is that out of 26414 of test data, only 129 are correctly predicted. i.e. only 0.48% of data is correctly predicted. The reasons may be as follows:

A. Either the splitted data are not distributed uniformly.
B. either I have replaced too many overfitted values with Inter Quartile Range.

C. Either I have replaced too many Null values and empty spaces with mean and median.
D. Either I have selected the wrong independent variables to create the dependent variable.
E. Either the conditions that I set for determining the dependent variable is biased.
F. Either the whole dataset is biased.

Things I can try to improve:

A. I will try to split the data in more uniform way.

B. I will not replace all outliers with IQR.
C. I will try calculate the total blank space and null values saperately and see if I can drop them.
D. I will try to change the independent variables.
E. I will try to change the conditions set for determining the dependent variable.
F. I will try to reduce the size of dataset.

Next approach:
Here, I reimported the dataset and did the similar data cleaning task but I neglected the outliers. And generated the dependent variable. And began to train the model. But to test the model, I did not removed the independent variables that were used to derive the dependent variable.

```
> listing$listing_is_good=with(listing,as.logical(listing$host_response_rate>60 &
+                                                  listing$review_scores_rating>4.2&
+                                                  listing$availability_365>40&
+                                                  listing$review_scores_location>=4.5
+ ))
>
```

split_glm =initial_split(listing,prop = 0.7) #splitting the data as 70-30 ratio

traindata=training(split_glm)

testdata=testing(split_glm)

model1=glm(listing_is_good~.,family = binomial(link = "logit"), data = traindata)

traindata_predict=predict(model1,traindata,type = "response")

traindata_classified=ifelse(traindata_predict>0.5,yes = 1,no=0)

table(traindata$listing_is_good,traindata_classified)

trainmodel_accuracy=mean(traindata$listing_is_good==is.logical(traindata_classified))

trainmodel_accuracy

testdata_predict=predict(model1,testdata,type = "response")

testdata_classified=ifelse(testdata_predict>0.5,yes = 1,no=0)

table(testdata$listing_is_good,testdata_classified)

testdata_accuracy=mean(testdata$listing_is_good== is.logical(testdata_classified)) testdata_accuracy

```
> split_glm =initial_split(listing,prop = 0.7) #splitting the data as 70-30 ratio
> traindata=training(split_glm)
> testdata=testing(split_glm)
> model1=glm(listing_is_good~.,family = binomial(link = "logit"), data = traindata )
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> traindata_predict=predict(model1,traindata,type = "response")
> traindata_classified=ifelse(traindata_predict>0.5,yes = 1,no=0)
> table(traindata$listing_is_good,traindata_classified)
       traindata_classified
            0     1
  FALSE 31630  2604
  TRUE   3442 23101
> trainmodel_accuracy=mean(traindata$listing_is_good==is.logical( traindata_classified))
> trainmodel_accuracy
[1] 0.5693169
>
> testdata_predict=predict(model1,testdata,type = "response")
> testdata_classified=ifelse(testdata_predict>0.5,yes = 1,no=0)
> table(testdata$listing_is_good,testdata_classified)
       testdata_classified
            0     1
  FALSE 13614  1058
  TRUE   1476  9861
> testdata_accuracy=mean(testdata$listing_is_good== is.logical(testdata_classified))
> testdata_accuracy
[1] 0.5707958
>
>
```

```
> summary(listing$listing_is_good)
FALSE  TRUE
50164 37880
```

```
> summary(testdata$listing_is_good)
FALSE  TRUE
15077 11337
> summary(traindata$listing_is_good)
FALSE  TRUE
35087 26543
>
```

Now, what I saw is just opposite of the previous model. Here I can clearly see that the model accuracy is low, but the detection rate is kinda believable.

I am eger to check If removing the independent variable will affect the prediction.

```
> listing$availability_365=NULL
> listing$review_scores_rating=NULL
> listing$host_response_rate=NULL
> listing$review_scores_location=NULL
>
```

```
> listing$availability_365=NULL
> listing$review_scores_rating=NULL
> listing$host_response_rate=NULL
> listing$review_scores_location=NULL
> split_glm =initial_split(listing,prop = 0.7) #splitting the data as 70-30 ratio
> traindata=training(split_glm)
> testdata=testing(split_glm)
> model1=glm(listing_is_good~.,family = binomial(link = "logit"), data = traindata )
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> traindata_predict=predict(model1,traindata,type = "response")
> traindata_classified=ifelse(traindata_predict>0.5,yes = 1,no=0)
> table(traindata$listing_is_good,traindata_classified)
       traindata_classified
            0     1
  FALSE 29805  4395
  TRUE   5816 20750
> trainmodel_accuracy=mean(traindata$listing_is_good==is.logical( traindata_classified))
> trainmodel_accuracy
[1] 0.5689437
>
> testdata_predict=predict(model1,testdata,type = "response")
> testdata_classified=ifelse(testdata_predict>0.5,yes = 1,no=0)
> table(testdata$listing_is_good,testdata_classified)
       testdata_classified
            0     1
  FALSE 12820  1886
  TRUE   2515  8799
> testdata_accuracy=mean(testdata$listing_is_good== is.logical(testdata_classified))
>
> testdata_accuracy
[1] 0.5716665
>
```

I found that for this data and this type of cleaning, removing of the independent variable that were used to derrive the dependent variable did not affect that much.

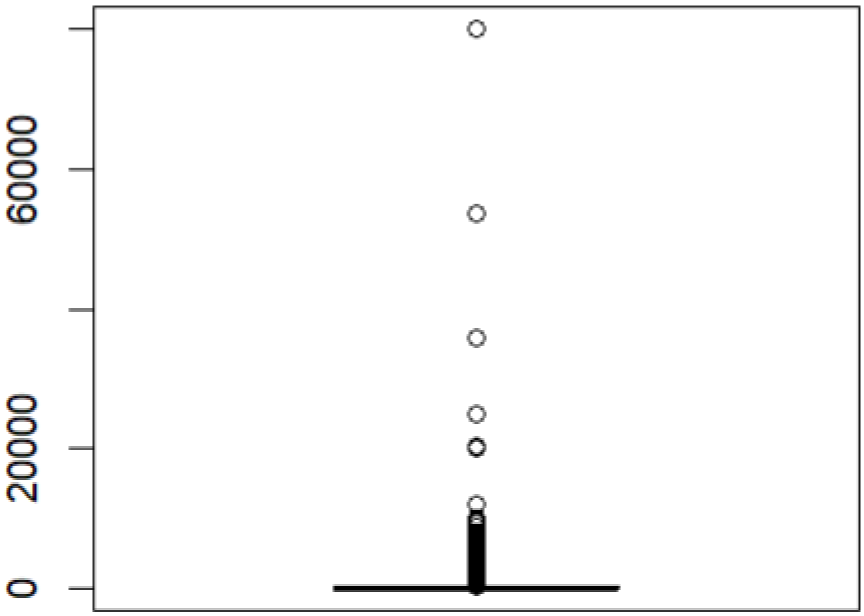
But why?

Lets re-import those columns and check the correrelation.

```
> listing$availability_365=listing_backup2$availability_365
> listing$review_scores_rating=listing_backup2$review_scores_rating
> listing$host_response_rate=listing_backup2$host_response_rate
> listing$review_scores_location=listing_backup2$review_scores_location
>
```

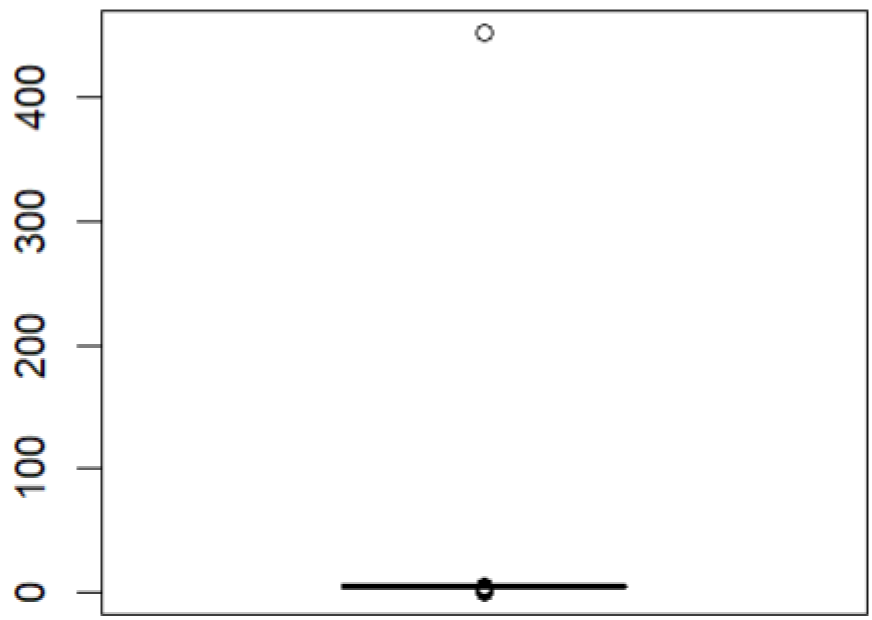
```
> listing$listing_is_good_numeric=ifelse(listing$listing_is_good=="TRUE", yes = 1, no=0)
```

```
> cor(listing$availability_365,listing$listing_is_good_numeric)
[1] 0.6000827
>
```

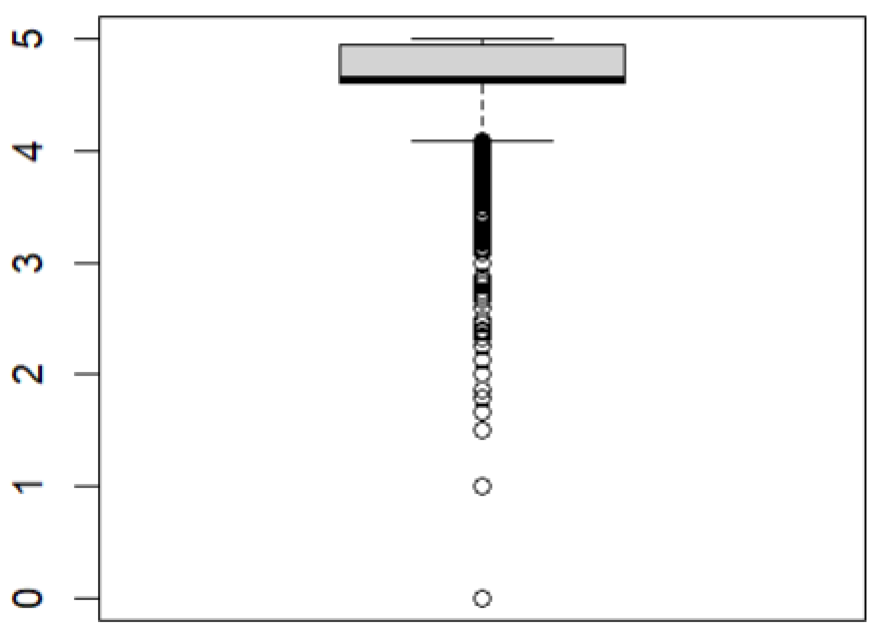


Here we can see that, the values above 20000 are outliers.

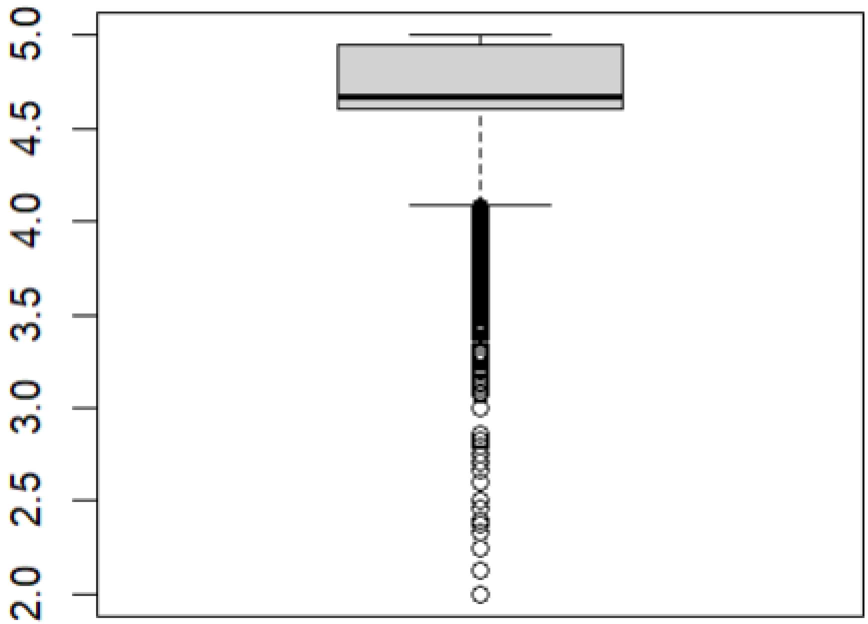
```
> sum(listing$price>20000)
[1] 6
```



The above figure is of review_scores_rating. Where we can see the outlier which lies above 400. And I will remove the outlier to see how it affects the model.

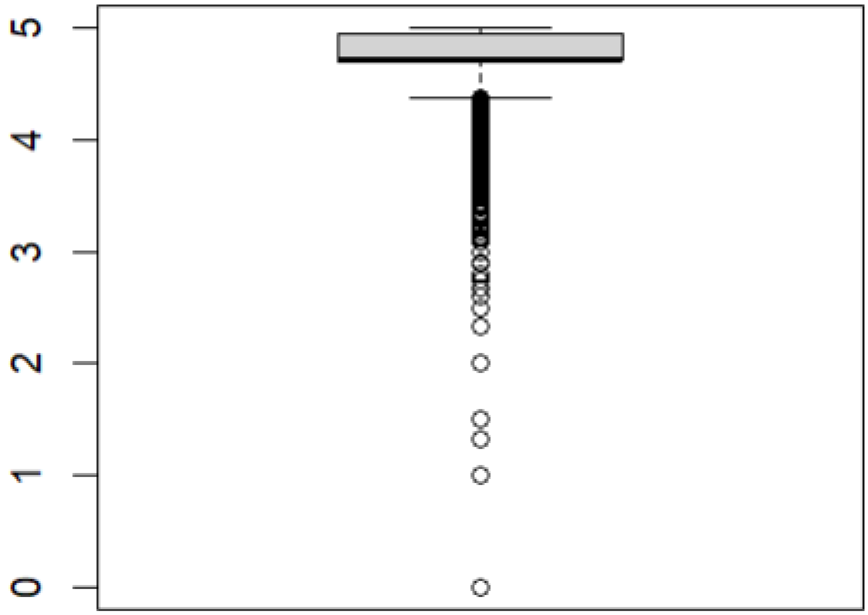


```
> listing= listing[listing$review_scores_rating<=100,]
> boxplot(listing$review_scores_rating)
>
```



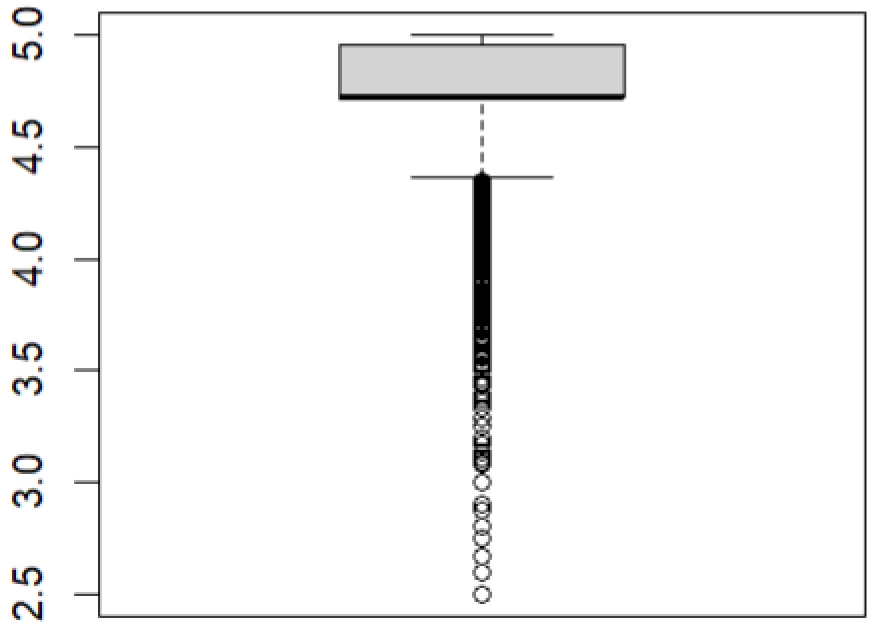

```
> listing= listing[listing$review_scores_rating>=2,]
> boxplot(listing$review_scores_rating)
>
```

Next:

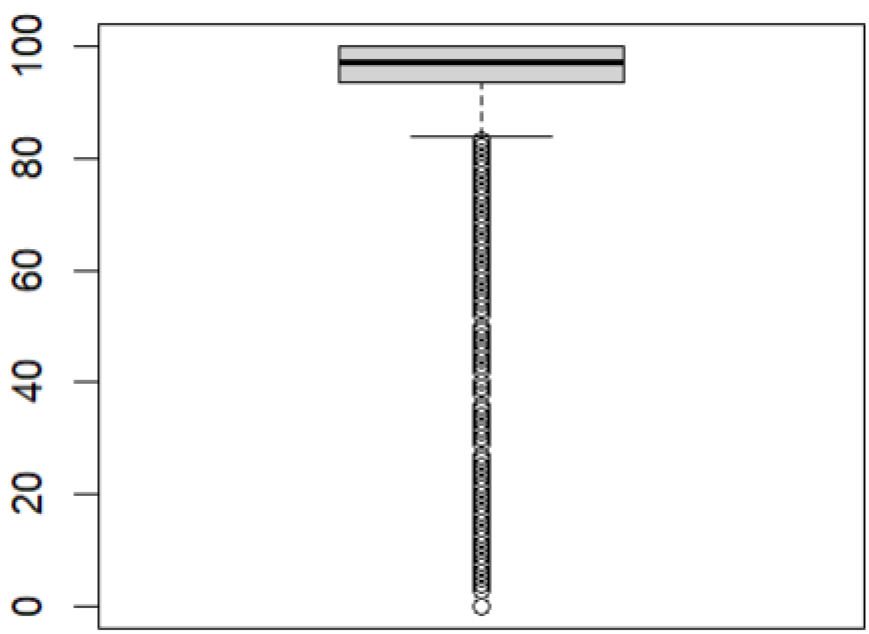


In review_score_location also the outliers can be clearly seen. I will remove those roes and generate the dependent variables to test the model.

```
> listing= listing[listing$review_scores_location>=2.5,]
> boxplot(listing$review_scores_location)
> 
```

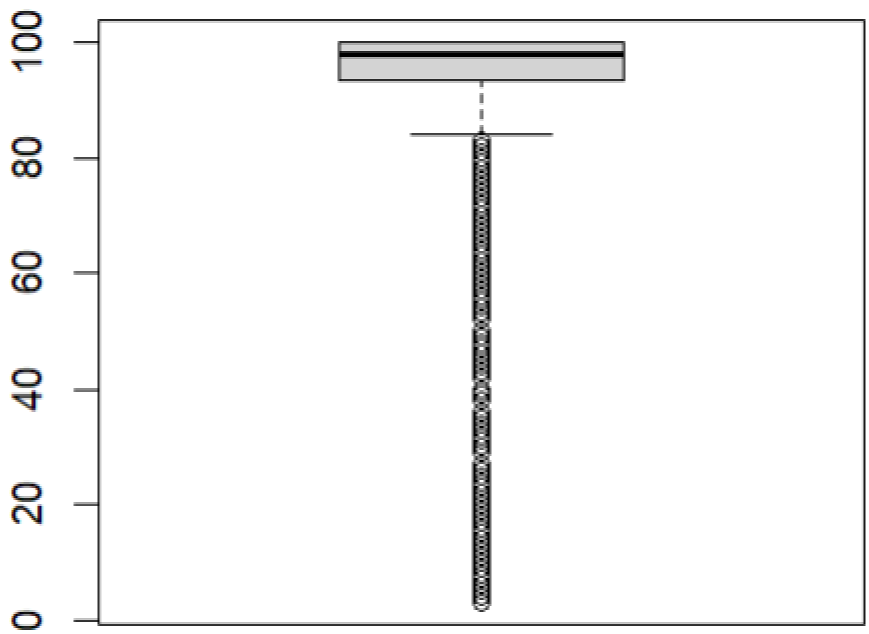


Next:



This is the boxplot of host_response_rate, here I can see that outliers lies near to the zeros. Since there is no gap between the data we cannot conside these values as outliers.

```
> listing= listing[listing$host_response_rate>1,]
> boxplot(listing$host_response_rate)
> 
```



After removal of outliers:

```
> dim(listing)
[1] 85700    37
> 
```

```
> listing$listing_is_good=with(listing,as.logical(listing$host_response_rate>90 &
+                                                        listing$review_scores_rating>4.6&
+                                                        listing$availability_365>60&
+                                                        listing$review_scores_location>=4.6
+ ))
>
> : listing$listing_is_good=as.factor(listing$listing_is_good)
Error: unexpected ':' in ":"
> listing$listing_is_good=as.factor(listing$listing_is_good)
> listing$host_response_rate=NULL
> listing$review_scores_rating=NULL
> listing$review_scores_location=NULL
> listing$availability_365=NULL
>
> dim(listing)
[1] 85700    33
> |
```

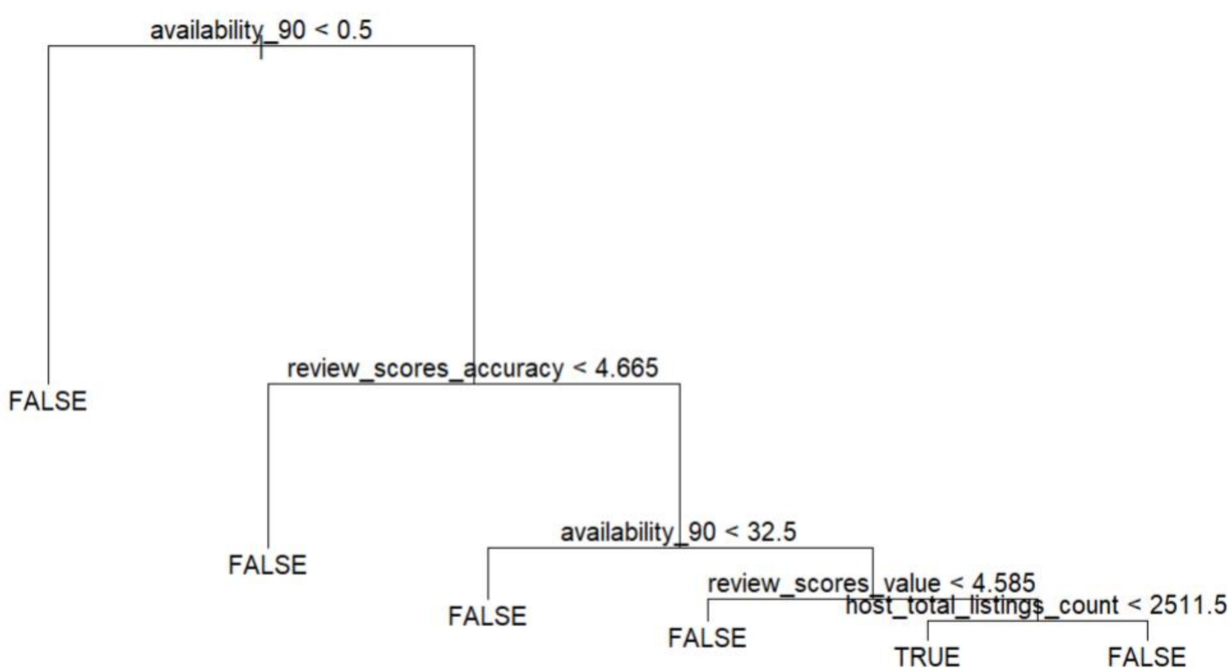
```
> listing$listing_is_good_numeric=NULL
> str(listing)
'data.frame':   85700 obs. of  32 variables:
 $ host_total_listings_count                   : num  1 2 4 12 1 32 3 3 2 3 ...
 $ latitude                                    : num  51.4 51.5 51.6 51.5 51.5 ...
 $ longitude                                   : num  -0.1874 -0.2171 -0.1127 -0.1681 0.0144 ...
 $ accommodates                                : num  2 5 1 2 2 6 4 3 4 2 ...
 $ beds                                        : num  2 3 1 1 1 3 3 1 1 1 ...
 $ price                                       : num  42 175 79 150 46 476 371 250 75 29 ...
 $ minimum_nights                              : num  2 5 1 7 4 3 5 2 2 10 ...
 $ maximum_nights                              : num  730 240 29 30 365 ...
 $ minimum_minimum_nights                      : num  2 5 1 7 2 3 5 2 2 10 ...
 $ maximum_minimum_nights                      : num  2 5 1 7 4 3 5 2 2 10 ...
 $ minimum_maximum_nights                      : num  1125 240 29 30 365 ...
 $ maximum_maximum_nights                      : num  1125 240 29 30 365 ...
 $ minimum_nights_avg_ntm                      : num  2 5 1 7 4 3 5 2 2 10 ...
 $ maximum_nights_avg_ntm                      : num  1125 240 29 30 365 ...
 $ availability_30                             : num  0 13 25 7 5 14 26 0 0 0 ...
 $ availability_60                             : num  0 18 55 7 5 26 56 0 0 0 ...
 $ availability_90                             : num  0 38 85 7 5 50 86 0 0 0 ...
 $ number_of_reviews                           : num  216 38 41 94 180 54 24 96 42 129 ...
 $ number_of_reviews_ltm                       : num  9 2 11 5 25 4 3 0 0 0 ...
 $ number_of_reviews_l30d                      : num  0 0 0 0 3 0 0 0 0 0 ...
 $ review_scores_accuracy                      : num  4.74 4.76 4.72 4.85 4.7 4.83 4.57 4.89 4.93 4.7 ...
 $ review_scores_cleanliness                   : num  4.86 4.62 4.72 4.88 4.59 4.71 4.7 4.91 4.71 4.94 ...
 $ review_scores_checkin                       : num  4.71 4.85 4.74 4.88 4.63 4.71 5 4.9 4.93 4.91 ...
 $ review_scores_communication                 : num  4.67 4.88 4.82 4.83 4.81 4.71 4.96 4.93 5 4.89 ...
 $ review_scores_value                         : num  4.68 4.74 4.69 4.74 4.67 4.6 4.39 4.65 4.8 4.74 ...
 $ calculated_host_listings_count              : num  1 1 2 1 1 9 1 1 1 3 ...
 $ calculated_host_listings_count_entire_homes : num  0 1 1 1 0 9 1 1 1 0 ...
 $ calculated_host_listings_count_private_rooms: num  1 0 1 0 1 0 0 0 0 3 ...
 $ calculated_host_listings_count_shared_rooms : num  0 0 0 0 0 0 0 0 0 0 ...
 $ reviews_per_month                           : num  1.45 0.27 0.26 0.56 1.21 0.36 0.16 0.62 0.64 0.79 ...
 $ host_engaged_years                          : num  12.4 12.4 13.8 13.8 12.4 ...
 $ listing_is_good                             : Factor w/ 2 levels "FALSE","TRUE": 1 1 2 2 1 2 2 1 1 1 ...
> |
```

So I will run the glm model again:

```
> split_glm =initial_split(listing,prop = 0.7) #splitting the data as 70-30 ratio
> traindata=training(split_glm)
> testdata=testing(split_glm)
> model1=glm(listing_is_good~.,family = binomial(link = "logit"), data = traindata )
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> traindata_predict=predict(model1,traindata,type = "response")
> traindata_classified=ifelse(traindata_predict>0.5,yes = 1,no=0)
> table(traindata$listing_is_good,traindata_classified)
       traindata_classified
            0     1
  FALSE 36409  4878
  TRUE   5898 11923
> trainmodel_accuracy=mean(traindata$listing_is_good==is.logical( traindata_classified))
> trainmodel_accuracy
[1] 0.7029289
>
> testdata_predict=predict(model1,testdata,type = "response")
> testdata_classified=ifelse(testdata_predict>0.5,yes = 1,no=0)
> table(testdata$listing_is_good,testdata_classified)
       testdata_classified
            0     1
  FALSE 15652  2126
  TRUE   2446  5111
> testdata_accuracy=mean(testdata$listing_is_good== is.logical(testdata_classified))
>
> testdata_accuracy
[1] 0.7060791
> summary(listing$listing_is_good)
FALSE  TRUE
60322 25378
> |
```

Further more:

The tree model:



```
> split_tree_data=initial_split(listing,prop = 0.7) #splitting the data as 70-30 ratio
> train_tree_data=training(split_tree_data) #assigining the train data
> test_tree_data=testing(split_tree_data) #assigning the test_data
> tree_model_good_notgood=tree(formula =listing_is_good~.,data=train_tree_data,)
> predict_good_notgood_train=predict(tree_model_good_notgood,train_tree_data,type = "class")
> table(train_tree_data$listing_is_good,predict_good_notgood_train)
       predict_good_notgood_train
        FALSE  TRUE
  FALSE 37243  4961
  TRUE   5831 11954
> train_model_accuracy_tree=mean(train_tree_data$listing_is_good==predict_good_notgood_train
> train_model_accuracy_tree
[1] 0.8201004
> predict_good_notgood_test=predict(tree_model_good_notgood,test_tree_data,type = "class")
>
> table(test_tree_data$listing_is_good,predict_good_notgood_test)
       predict_good_notgood_test
        FALSE  TRUE
  FALSE 15985  2133
  TRUE   2572  5021
> test_model_accuracy_tree=mean(test_tree_data$listing_is_good==predict_good_notgood_test)
> test_model_accuracy_tree
[1] 0.8170044
>
> plot(tree_model_good_notgood)
> text(tree_model_good_notgood)
>
> |
```

Here, I can see that now the accuracy of the tree model has increase than the previous time. Also the predicted values looks convinsing. In the earlier model, my model acuuracy was 73% and also the data that the model has predicted doesnot looks convincing.

Conclusion:

I found that I was overfitting my data on the earlier stages by assuming the datas above 95% in boxplot as outliers. Also, the model accuracy depends on the conditions that are applied to generate the dependent variable. If I bottleneck the data to the dependent variable most of the rows will be marked as false hence will affect the quality and accuracy of model. Also, the data distribution and ratio of spit will affect the accuracy of model. There is also difference between the results of glm model and decision tree model. I found from my data and models that usually, the model accuracy of the decision tree is higher than the glm model. In this case this is natural because, this is the case of categorization of listed property is good or bad. I think if this problem was about making logical decisions rather than categorization than glm would perform better than decision trees.

The data that was available, had many errors like where there should be numeric values there were descriptions and lengthy text. I found that replacing those vales with mean, median, mode has also affected the expected result. But the thing is that if I had removed those vales, the whole row would have been deleted which is also not good because I would have lost the good data also. I did not find the proper way for treating those data espicially when their volume is large and that column is the most important column.

I also found that, good model accuracy doesnot means that the model is good. In my understanding till now, a model is said to be good if it’s prediction is reasonable, sensable and convinsing. Through this coursework I learned so much. I did planty of research though I couldn’t document all the them, it helped me to increse the analytical thinking.I my view, it is a never-ending process and we cannot say that the model is perfect or we cannot make the perfect model. We can improvise the model at any point.