
# AD 699 Data Mining for Business Analytics

## Semester Project

## Airbnb Bastille

Team Bravo Members
Zhen Fu
Rong Gao
Keyu Mi
Haonan Kou
Tong Shen
Xin Wang

Prof. Greg Page

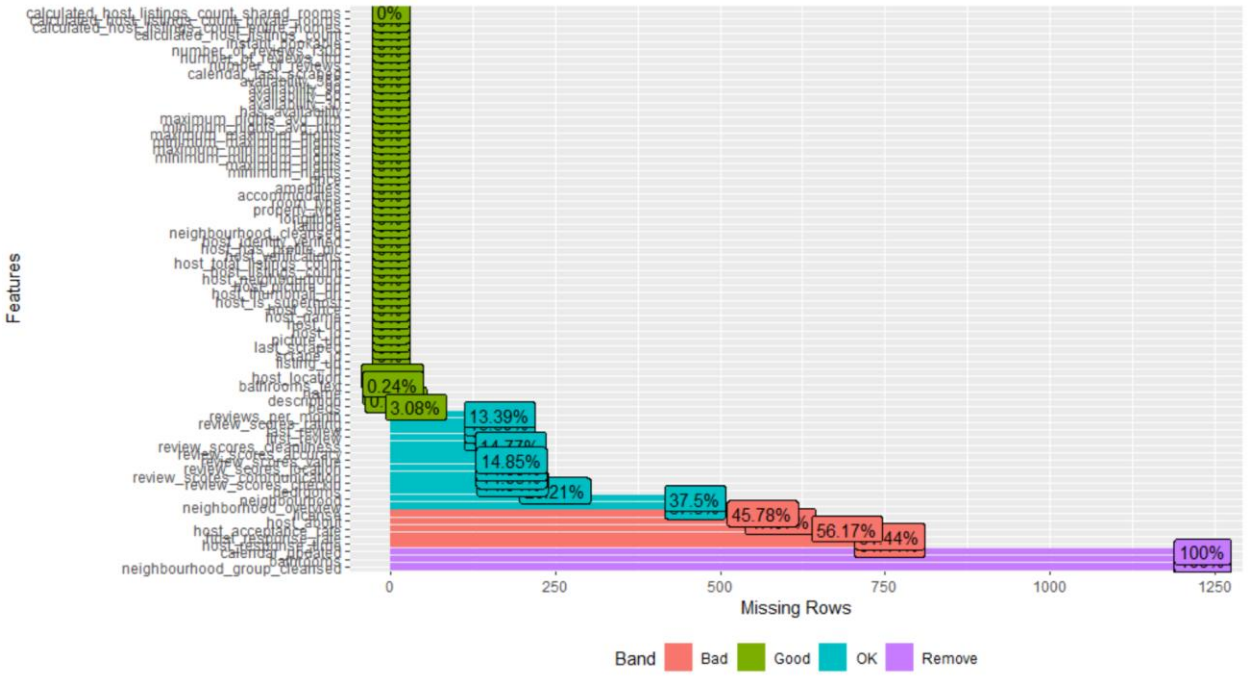
May 5, 2022

# Contents

# Step I: Data Preparation & Exploration

## I. Missing Values

```
> data <- read_csv("C:/Users/41141/Desktop/699 final/paris_listings.csv")
Rows: 49429 Columns: 74

-- Column specification --------------------------------------------------------
Delimiter: ","
chr (23): listing_url, name, description, neighborhood_overview, pictu...
dbl (43): id, scrape_id, last_scraped, host_id, host_since, host_listi...
lgl  (8): host_is_superhost, host_has_profile_pic, host_identity_verif...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this mes
sage.
Warning message:
One or more parsing issues, see `problems()` for details
> Airbnb_Bastille <- data %>% filter(host_neighbourhood == "Bastille")
> sum(is.na(Airbnb_Bastille))
[1] 7834
> Airbnb_Bastille[Airbnb_Bastille=="N/A"]<-NA
> sum(is.na(Airbnb_Bastille))
[1] 10040
> NA_data <- t(data.frame((lapply(lapply(Airbnb_Bastille, is.na), sum))))
> plot_missing(Airbnb_Bastille)
```



```
> a <- colSums(is.na(Airbnb_Bastille))
> a
                                          id                                  listing_url
                                           0                                            0
                                    scrape_id                                last_scraped
                                           0                                            0
                                        name                                 description
                                           3                                            7
                        neighborhood_overview                                 picture_url
                                         462                                            0
                                     host_id                                    host_url
                                           0                                            0
                                   host_name                                 host_since
                                           0                                            0
                               host_location                                 host_about
                                           2                                          591
                          host_response_time                             host_response_rate
                                         757                                          757
                       host_acceptance_rate                              host_is_superhost
                                         692                                            0
                         host_thumbnail_url                               host_picture_url
                                           0                                            0
                         host_neighbourhood                           host_listings_count
                                           0                                            0
                                  bathrooms                                bathrooms_text
                                        1232                                            3
                                   bedrooms                                         beds
                                         249                                           38
                                  amenities                                        price
                                           0                                            0
                              minimum_nights                               maximum_nights
                                           0                                            0
                         minimum_minimum_nights                       maximum_minimum_nights
                                           0                                            0
                         minimum_maximum_nights                       maximum_maximum_nights
                                           0                                            0
                         minimum_nights_avg_ntm                       maximum_nights_avg_ntm
                                           0                                            0
                               calendar_updated                             has_availability
                                        1232                                            0
                                availability_30                              availability_60
                                           0                                            0
                                availability_90                              availability_365
                                           0                                            0
                         calendar_last_scraped                            number_of_reviews
                                           0                                            0
                               calendar_updated                             has_availability
                                        1232                                            0
                                availability_30                              availability_60
                                           0                                            0
                                availability_90                              availability_365
                                           0                                            0
                         calendar_last_scraped                            number_of_reviews
                                           0                                            0
                         number_of_reviews_ltm                        number_of_reviews_l30d
                                           0                                            0
                                  first_review                                 last_review
                                         165                                          165
                          review_scores_rating                         review_scores_accuracy
                                         165                                          182
                    review_scores_cleanliness                          review_scores_checkin
                                         182                                          184
                  review_scores_communication                          review_scores_location
                                         183                                          183
                         review_scores_value                                      license
                                         183                                          564
                             instant_bookable                 calculated_host_listings_count
                                           0                                            0
calculated_host_listings_count_entire_homes calculated_host_listings_count_private_rooms
                                           0                                            0
```

As shown, the values of bathrooms and neighbourhood_group_cleansed are all NA, so we drop them directly. Later models will select some of the useful predictive variables, so we need to remove the existing NA values or perform interpolation operations when using them.

## II. Summary Statistics

```
> #II. Summary Statistics
> # Price
> sum(is.na(Bastille$price))
[1] 0
> str(Bastille$price)
 num [1:1232] 146 60 45 139 50 344 130 250 95 180 ...
> summary(Bastille$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   22.0    62.0    85.0   109.3   123.0  1000.0
> sd(Bastille$price)
[1] 87.04059
> mean(Bastille$price)
[1] 109.3214
> min(Bastille$price)
[1] 22
> max(Bastille$price)
[1] 1000
> median(Bastille$price)
[1] 85
```

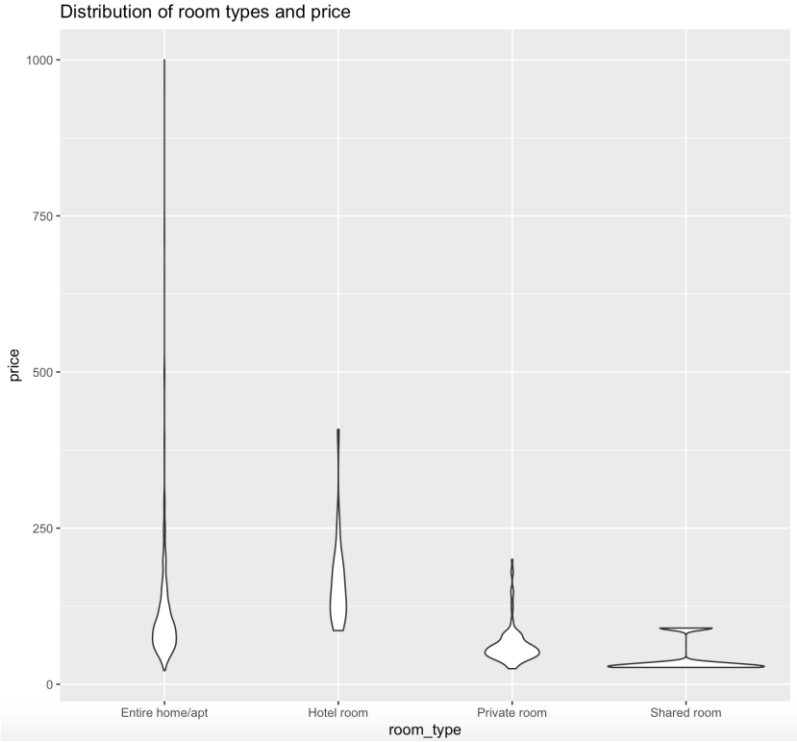
After making sure there were no null values under the price variable, I used str(), summary(), sd(), mean(), min(), max() and median () function to obtain some summary statistics of price. As you can see from the results of the above exploration, Airbnb prices in the Bastille area range from $22 to $1000, which is a surprisingly large difference. Also, the standard deviation of up to 87 shows that there is a huge difference between prices. And the average value price is $109, the median is $85.

```
> # rating
> summary(Bastille$review_scores_rating)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  0.000   4.580   4.790   4.638   4.940   5.000     165
> sd(Bastille$review_scores_rating,na.rm = TRUE)
[1] 0.6640745
> mean(Bastille$review_scores_rating,na.rm = TRUE)
[1] 4.637619
> min(Bastille$review_scores_rating,na.rm = TRUE)
[1] 0
> max(Bastille$review_scores_rating,na.rm = TRUE)
[1] 5
> median(Bastille$review_scores_rating,na.rm = TRUE)
[1] 4.79
```

In addition to price, review ratings are also very important for Airbnb. So we explored the summary statistics of the ratings with the same function. From the above results, we can understand that after ignoring null values, the lowest rating is 0 and the highest rating is 5. Although there seems to be a lot of variability among the ratings, it is clear from the mean of 4.6 and the median of 4.79 that most of the ratings are concentrated above 4.5. This means that the overall rating of Airbnb in the Bastille area is relatively high.

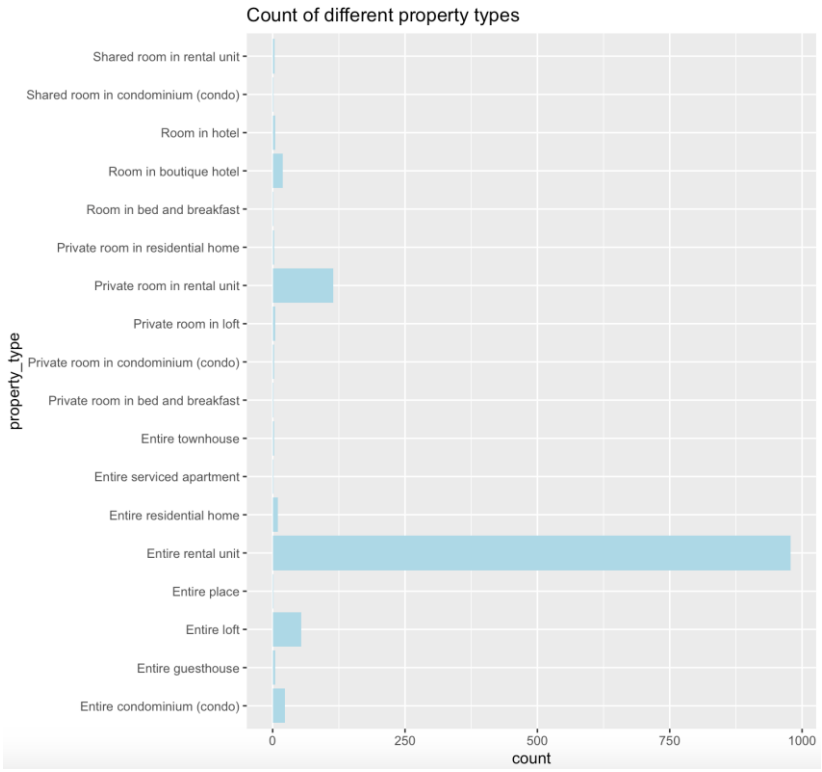
## III. Data Visualization

```
> #III. Data Visualization
> ggplot(data=Bastille, aes(x=room_type,y=price)) + geom_violin()+
+   ggtitle('Distribution of room types and price')
```



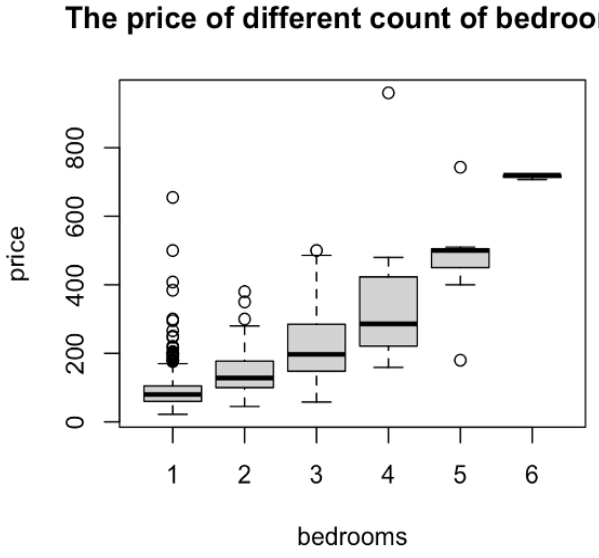

With this violin plot showing the relationship between room type and price, we can see that the entire home/apt has the largest price range, and it has the largest counts. Hotel rooms, on the other hand, have the highest minimum price.

```
> ggplot(Bastille,aes(x=property_type))+geom_histogram(stat="count",binwidth=4,fill="lightblue")+
+   coord_flip()+ggtitle('Count of different property types')
```



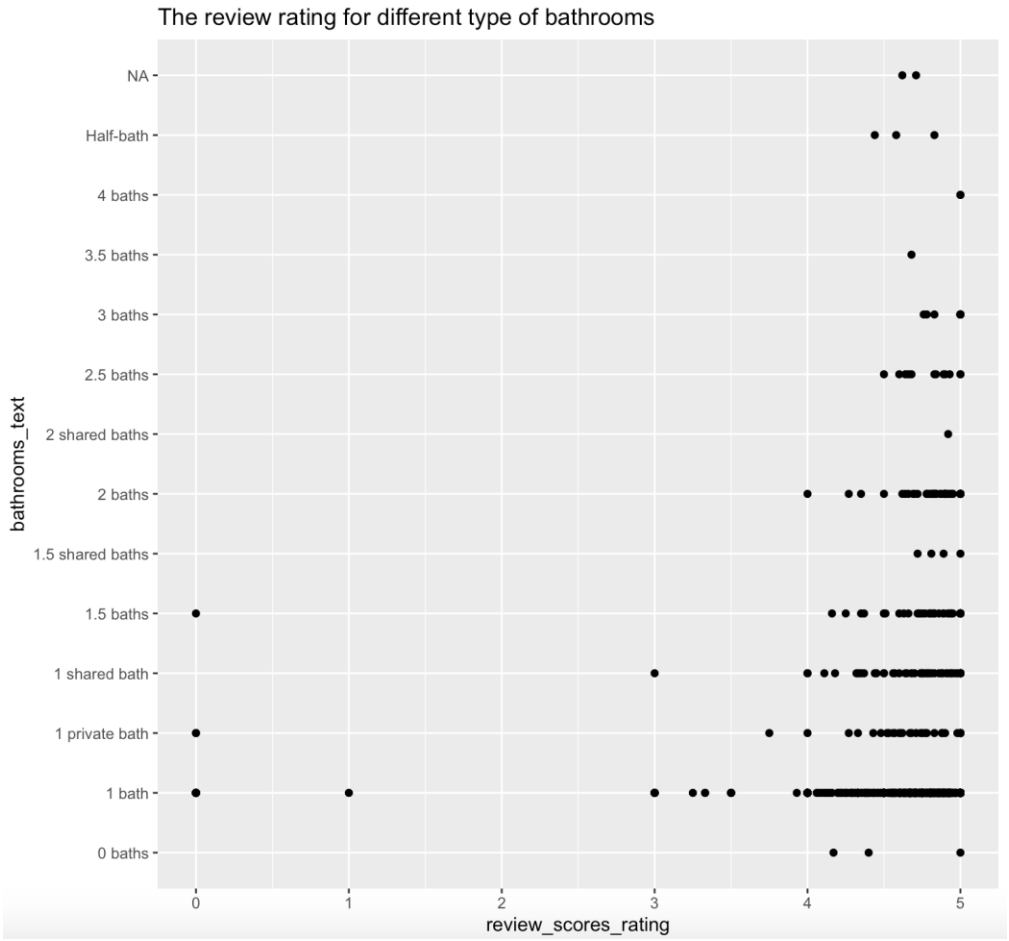

In this plot above, we explore the number of different property types. It is clear that the number of entire rental units is much higher than the other types, almost 875 more than the number of private rooms in rental units, which is the second-ranked category. And other than that, the difference in numbers between the remaining types is not very large.

```
> boxplot(Bastille$price~Bastille$bedrooms,xlab = "bedrooms",ylab = "price",
+         main="The price of different count of bedrooms")
```



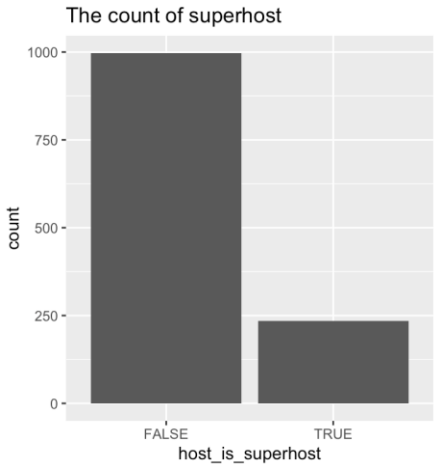
This box plot above gives us an idea of the difference in price for the different numbers of rooms. Not surprisingly, when the number of bedrooms becomes greater, the median price becomes higher. However, it is worth noting that the price range is largest when the number of bedrooms is 3 and the quartile range of prices is largest when the number of bedrooms is 4.

```
> ggplot(data=Bastille, aes(x=review_scores_rating,y=bathrooms_text)) + geom_point()+
+   ggtitle('The review rating for different type of bathrooms')
```




This scatter plot explores the distribution of customer ratings for different numbers and types of bathrooms. Apparently, the low scores rating by customers are found in the bathrooms of 1 to 1.5. When the number of bathrooms is 2, customers' scores rating are greater than 4, and when the number of bathrooms is greater than 2, customers' scores are generally higher than 4.5.

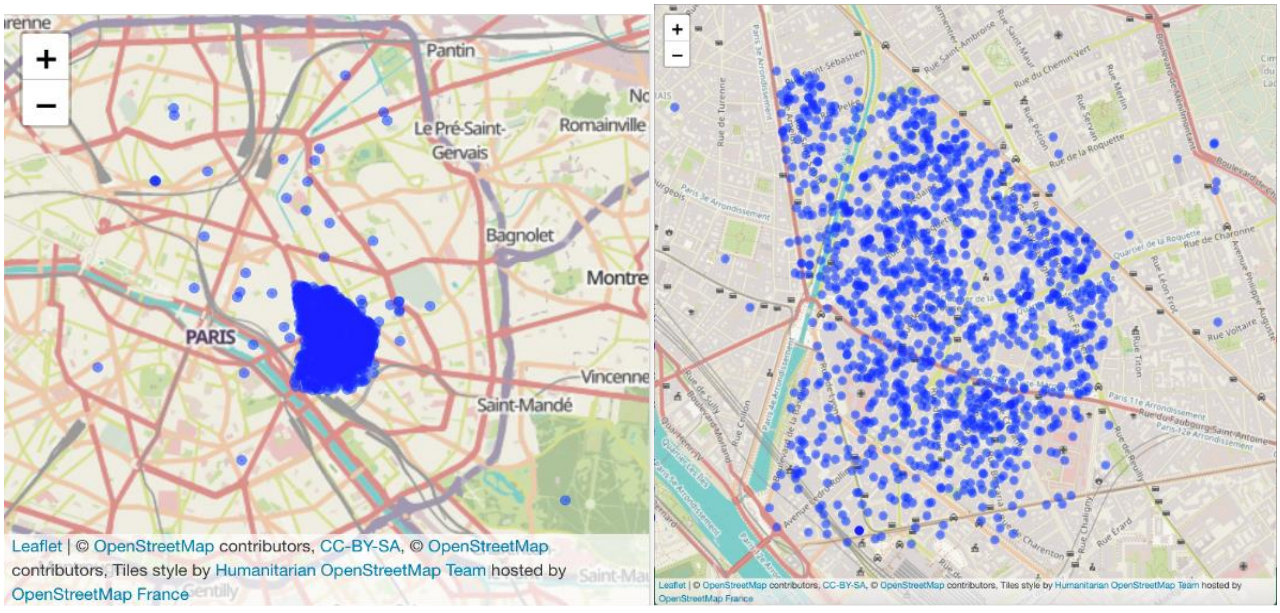
```
> ggplot(Bastille,aes(x=host_is_superhost))+geom_bar()+
+   ggtitle('The count of superhost')
```




This plot above shows the count of superhost, we can find that nearly 250 of the 1232 entries were superhosts, which is about 20% of the total. According to the research, only 7% of the hosts in the whole Airbnb are superhosts. This means that the overall quality of Airbnb service in the Bastille area is really good.

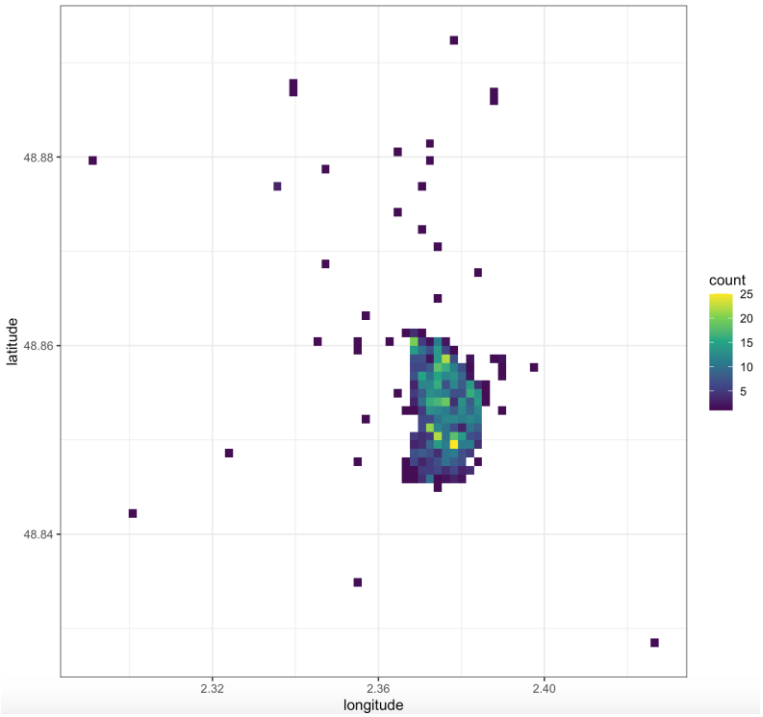
## IV. Mapping

```
> m <- leaflet() %>% addTiles() %>% addCircles(lng= Bastille$longitude, lat= Bastille$latitude ) %>%
+   addProviderTiles(providers$OpenStreetMap.HOT)
> m
```



By generating a map of the Bastille area, we found that most of the listings are concentrated in one area, although for all of them the dispersion is wide. After zooming in on the map, we can view the streets where the listings are clustered.

```
> ggplot(Bastille, aes(x=longitude, y=latitude) ) +
+   geom_bin2d(bins = 70) +
+   scale_fill_continuous(type = "viridis") +
+   theme_bw()
```




By creating a heat map showing the density of listings in Bastille, it is clear that the listings are most dense at a longitude of 3.68 and latitude of 48.85, in addition, with spreading around, the density decreases.

## V. Wordcloud

```
> Bastille_neigh <- subset(Bastille,select = c("neighborhood_overview"))
> Bastille_neigh <- na.omit(Bastille_neigh)
```

```
> crp <- VCorpus(VectorSource(Bastille_neigh))
> crp <- tm_map(crp,stripWhitespace)
> crp <- tm_map(crp, removePunctuation)
> crp <- tm_map(crp, removeWords,stopwords("english"))
> crp <- tm_map(crp, stemDocument)
> tdm <- TermDocumentMatrix(crp)
> m <- as.matrix(tdm)
> v <- sort(rowSums(m),decreasing=TRUE)
> Bastille_neigh1 <- data.frame(word = names(v),freq=v)
> head(Bastille_neigh1, 10)
            word freq
bastill   bastill  607
restaur   restaur  570
place       place  478
pari         pari  450
quartier quartier  444
des           des  382
bar           bar  355
the           the  329
les           les  325
rue           rue  307
> set.seed(1000)
> wordcloud(words = Bastille_neigh1$word, freq = Bastille_neigh1$freq, min.freq = 1,
+           max.words=100, random.order=FALSE, rot.per=0.35,
+           colors=brewer.pal(8, "Dark2"))
```

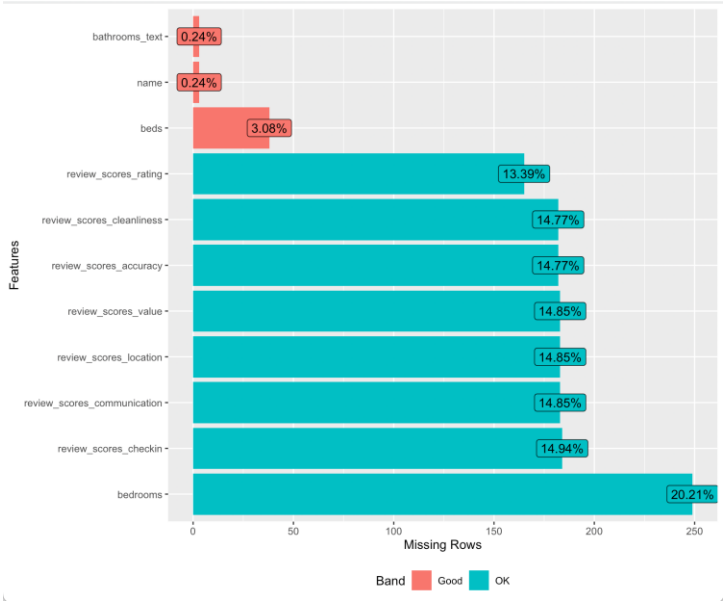

Using the neighborhood overview column to generate a word cloud, after removing the white space, punctuation, stopwords, and stemming, the result is shown in the figure above. We can see that the most emphasized words are bastill, restaur, place, pari, quartier, etc., which means that bastille hosts are more likely to use these words to describe their rented rooms.

# Step II Prediction

To set up a prediction of 'price,' we removed variables that were not directly related to prices, including scrape_iid and listing URL, etc., and kept variables included: property type, room type, accommodates, beds, price, availability, number of views, reviews scores rating, review scores accuracy, review scores cleanliness, review scores check-in, review scores communication, review scores location, review scores value, and instantly bookable, also shown as below.

```
var_b <- c("property_type",
           "room_type",
           "accommodates","bathrooms_text",
           "bedrooms","beds","price",
           "availability_30","number_of_reviews",
           "review_scores_rating","review_scores_accuracy",
           "review_scores_cleanliness",
           "review_scores_checkin",
           "review_scores_communication",
           "review_scores_location","review_scores_value",
           "instant_bookable"
)
bas_bnb <-  Airbnb_Bastille[, var_b]
```

Among the selected 17 variables, we checked their missing values. According to the results shown in the below plot, we dropped all the NAs since the number of missing values removed is relatively a small portion, which will not have much impact on the overall data volume.

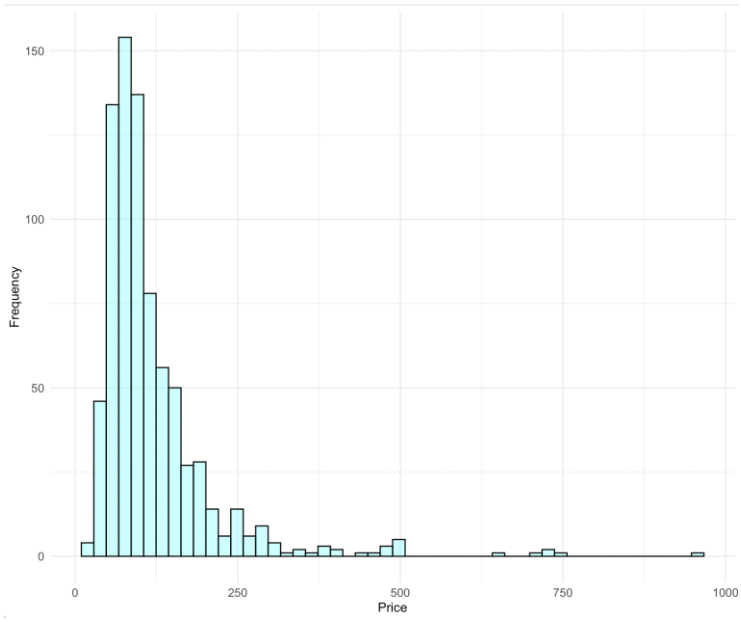


The numerical variables consist of accommodates, bedrooms, beds, review numbers, and scores, and also include availability in 30 days. Additionally, the bathroom as a class character in the original dataset, we extracted the bathroom numbers and converted them into the numerical variables. Variables that property_types and room_type are converted to factor variables which will dummy in the prediction. However, considering the too many levels in variable property_type will make the model complex and misleading the result, so we only keep the type more than 5. Consequently, we reduced to  6 different types of properties “Entire home/apt,” “Hotel room”, “Private room”, and “Shared room,” the most common property type.

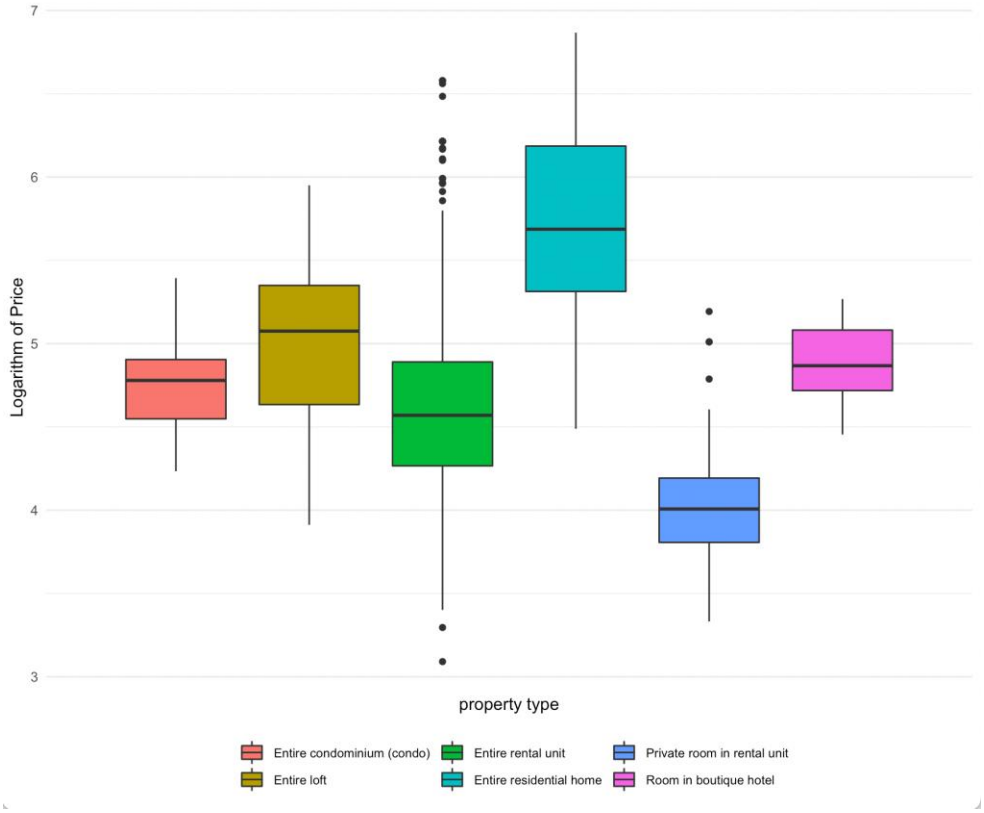
```
> table(bas_bnb$property_type)

          Entire condominium (condo)                     Entire guesthouse
                                  15                                     4
                        Entire loft                    Entire rental unit
                                  43                                   622
             Entire residential home             Entire serviced apartment
                                   9                                     1
                     Entire townhouse    Private room in bed and breakfast
                                   2                                     2
Private room in condominium (condo)                  Private room in loft
                                   2                                     5
          Private room in rental unit     Private room in residential home
                                  88                                     3
            Room in bed and breakfast                Room in boutique hotel
                                   1                                    15
                       Room in hotel           Shared room in rental unit
                                   3                                     3
```

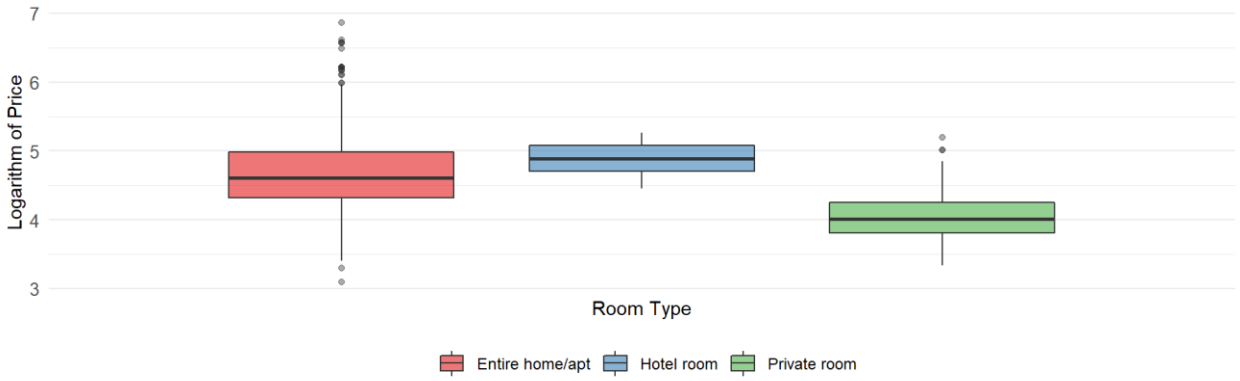
```
> bas_bnb <-bas_bnb[!as.numeric(bas_bnb$property_type) %in%which(table(bas_bnb$property_type) <= 5), ]
> bas_bnb$property_type <-droplevels(bas_bnb$property_type)
> levels(bas_bnb$property_type) # reduce to 6 levels
[1] "Entire condominium (condo)"
[2] "Entire loft"
[3] "Entire rental unit"
[4] "Entire residential home"
[5] "Private room in rental unit"
[6] "Room in boutique hotel"
>
```

Before the prediction, we generated a few visualizations of variables we were interested in to ensure their changes impact the price on some extent. For the predictive variable, price, the histogram shown below provides information regarding its abnormal distribution, rather a skewed to the right.



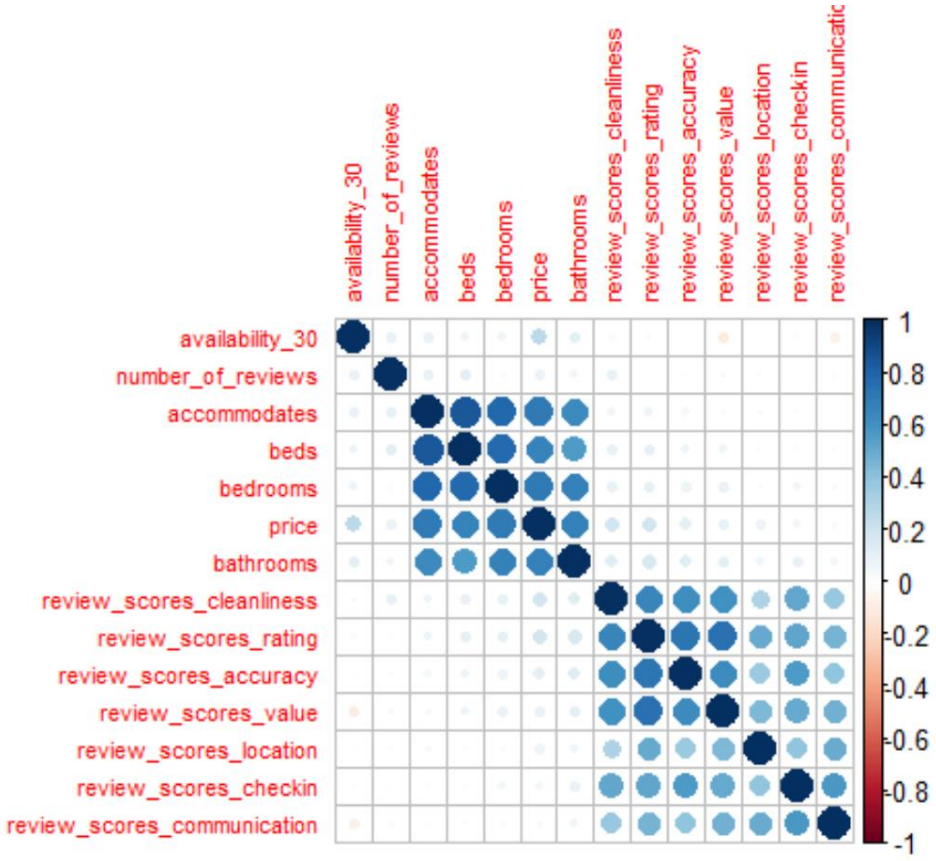

Two boxplots were created according to the type of different properties and rooms. Plots showed that entire residential homes and hotel rooms are at the more expensive price. Also, we noticed that properties of ‘entire rental units’ and ’entire home/apt’ rooms contained lots of outliers indicating there are large deviations from the average level.






We hope that with more beds and accommodates, the price will be higher. So we generated a scatter plot below to examine that.


The correlation between our 17 variables with prices is shown below. We can see that prices are highly correlated with accommodates, bedrooms, beds, and bathrooms. The similarity between variables other than the outcome variable does not exceed 90%. So the variables can all be used as predictor. Gathering information above, we set up prediction model and start with l re-check the  proper variables.




## Build Multiple Linear Regression

After partitioning 60% of data into training and others into the valid set, we set the prediction model with all selected variables at first. Considering there are a few variables that are not significant, and the adjusted R square is low as 0.6612, we need to reduce the variables.

```
> mlr1 = lm(price ~ ., data = train)
> summary(mlr1)

Call:
lm(formula = price ~ ., data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-125.23  -22.57   -3.78   19.35  419.70

Coefficients: (1 not defined because of singularities)
                                    Estimate Std. Error
(Intercept)                        -94.44443   62.27633
property_typeEntire loft            -4.67417   18.64307
property_typeEntire rental unit     -0.03732   16.41062
```

```
room_typeHotel room                 0.248   0.8046
room_typePrivate room                  NA       NA
accommodates                        3.991 7.66e-05 ***
bedrooms                            5.016 7.60e-07 ***
beds                                1.333   0.1831
availability_30                     5.679 2.42e-08 ***
number_of_reviews                   0.488   0.6260
review_scores_rating                2.003   0.0458 *
review_scores_accuracy             -1.168   0.2434
review_scores_cleanliness           2.508   0.0125 *
review_scores_checkin              -0.988   0.3238
review_scores_communication         0.238   0.8122
review_scores_location              0.441   0.6592
review_scores_value                -1.113   0.2664
instant_bookableTRUE               -1.070   0.2850
bathrooms                           6.638 9.10e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 45.34 on 454 degrees of freedom
Multiple R-squared:  0.6755,    Adjusted R-squared:  0.6612
F-statistic: 47.24 on 20 and 454 DF,  p-value: < 2.2e-16
```

We used the step function and set direction equals to backward for abbreviated values.

```
> #step reducing
> library(stats)
> mlr2=step(mlr1,direction='backward')
Start:  AIC=3643.92
price ~ property_type + room_type + accommodates + bedrooms +
    beds + availability_30 + number_of_reviews + review_scores_rating +
    review_scores_accuracy + review_scores_cleanliness + review_scores_c
heckin +
    review_scores_communication + review_scores_location + review_scores
_value +
    instant_bookable + bathrooms

                             Df Sum of Sq     RSS    AIC
- review_scores_communication  1       116  933254 3642.0
- room_type                    1       126  933263 3642.0
- review_scores_location       1       400  933538 3642.1
- number_of_reviews            1       489  933626 3642.2
- review_scores_checkin        1      2005  935143 3642.9
- instant_bookable             1      2355  935492 3643.1
```

```
Step:  AIC=3633.83
price ~ property_type + accommodates + bedrooms + beds + availability_30
 +
    review_scores_rating + review_scores_accuracy + review_scores_cleanl
iness +
    bathrooms

                           Df Sum of Sq     RSS    AIC
<none>                                   940855 3633.8
- beds                      1      4241  945097 3634.0
- review_scores_accuracy    1      5512  946367 3634.6
- review_scores_rating      1      7027  947883 3635.4
- review_scores_cleanliness 1     10025  950880 3636.9
- property_type             5     42314  983169 3644.7
- accommodates              1     33620  974475 3648.5
- bedrooms                  1     50124  990979 3656.5
- availability_30           1     75156 1016012 3668.3
- bathrooms                 1     91797 1032652 3676.1
```

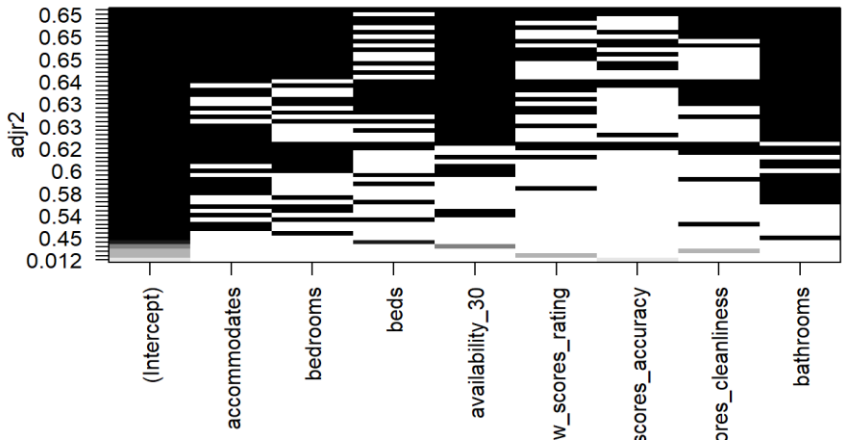
```
> summary(mlr2)

Call:
lm(formula = price ~ property_type + accommodates + bedrooms +
    beds + availability_30 + review_scores_rating + review_scores_accura
cy +
    review_scores_cleanliness + bathrooms, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-126.49  -23.10   -2.88   18.83  420.19

availability_30                          6.068 2.70e-09 ***
review_scores_rating                     1.856   0.0641 .
review_scores_accuracy                  -1.643   0.1010
review_scores_cleanliness                2.216   0.0272 *
bathrooms                                6.707 5.85e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 45.18 on 461 degrees of freedom
Multiple R-squared:  0.6728,    Adjusted R-squared:  0.6635
F-statistic: 72.91 on 13 and 461 DF,  p-value: < 2.2e-16
```

After the step function performs automatic variable filters on the model based on the AIC minimization principle (shown above), the AIC starts with 3643.92 running down to 3633.83. The adjusted R square increased to 0.6635.

Then, we use the vif function to examine the multicollinearity between variables. The vif of ‘accommodates’ is 4.6, is slightly higher but still within the threshold = 5. Considering our goal is to build the model with balance performance, selecting the ‘accommodates’ will not cause too many concerns about collinearity.

```
> vif(mlr2)
                            GVIF Df GVIF^(1/(2*Df))
property_type           1.508992  5        1.042002
accommodates            4.618059  1        2.148967
bedrooms                3.475007  1        1.864137
beds                    4.081370  1        2.020240
availability_30         1.224483  1        1.106564
review_scores_rating    2.569037  1        1.602822
review_scores_accuracy  2.279080  1        1.509662
review_scores_cleanliness 1.953724  1        1.397757
bathrooms               2.068093  1        1.438086
```

The adjusted R square is not ideal, although the P value less than 0.05 means that the model still good fit. To improve accuracy and further select the appropriate variables, we're using the leaps() function to find which quantitative variables lead to better adjusted R squared values.



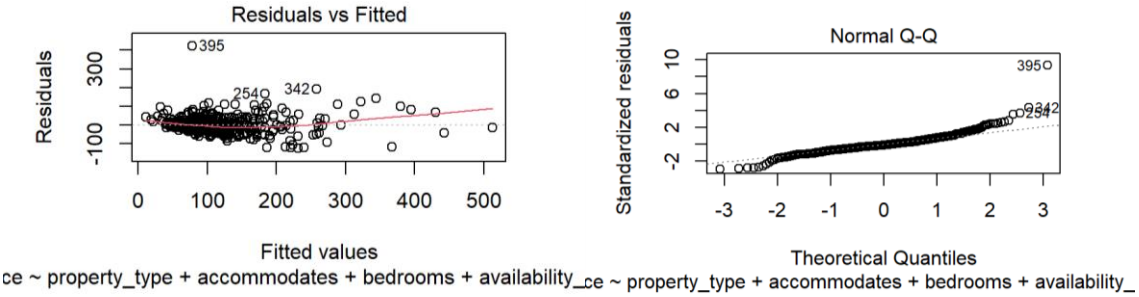

The plot above shows the adjusted R^2 is the most  when all numerical values included. Combined with the significant result in last model, the variable  'beds' and 'review scores accuracy'' showed non-significant, even if we delete them the value of adjusted R^2 does not change much. So finally we remove these two variables to build the prediction model.

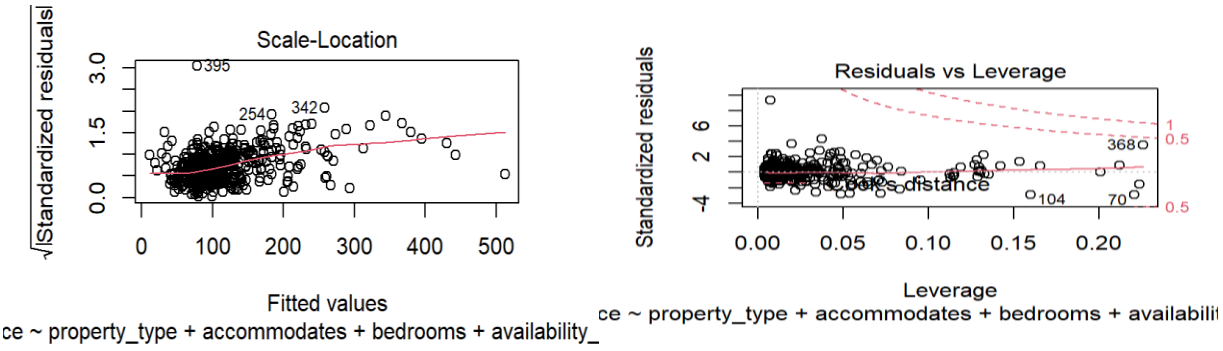
We also tried to improve the adjusted R^2 value by removing the variable 'review_score_rating ' and 'review_score_cleanliness' since these two variables have low P value and not strong significant, however the adjusted R^2 are reduced, so I keep them. The final prediction model is shown below.

```
> summary(MLR)

Call:
lm(formula = price ~ property_type + accommodates + bedroo
ms +
    availability_30 + review_scores_rating + review_scores
_cleanliness +
    bathrooms, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-124.99  -23.27   -4.99   19.01  421.57

                                  0.000228 ***
(Intercept)
property_typeEntire loft          0.862219
property_typeEntire rental unit   0.995048
property_typeEntire residential home  0.033299 *
property_typePrivate room in rental unit 0.170822
property_typeRoom in boutique hotel  0.644539
accommodates                      1.27e-09 ***
bedrooms                          1.57e-08 ***
availability_30                   3.85e-09 ***
review_scores_rating              0.223523
review_scores_cleanliness         0.062125 .
bathrooms                         1.47e-10 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 45.31 on 463 degrees of freedom
Multiple R-squared:  0.6693,    Adjusted R-squared:  0.6615
F-statistic:  85.2 on 11 and 463 DF,  p-value: < 2.2e-16
```

Thus, the final prediction equation is :
Price = -136.4815 -3.2192*property_typeEntire loft + 0.1011*property_typeEntire rental unit + 56.3045*property_typeEntire residential home - 23.9418*property_typePrivate room in rental uni  +10.5853*property_typeRoom in boutique hotel + 12.6786*accommodates + 27.3194* bedrooms  + 2.0317*availability_30 + 11.2893*review_scores_rating+12.0073* review_scores_cleanliness+ 49.1352*bathrooms

The adjusted R^2 in the model is 0.6615 indicating that the model can explain the 66.15% of data. Then we used the Q-Q plot checking the normality. The distribution of residuals shows there exists few outliers as the residuals not well evenly distributed.




The prediction result based on the train set showed below. As it's hard to access the fit of the model, we conduct cross validation.

```
> pre1 <- predict(MLR,train)
> train_accruacy <-accuracy(pre1,train$price)
> train_accruacy
                    ME     RMSE      MAE       MPE     MAPE
Test set 7.663426e-13 44.73886 29.97618 -10.63871 29.34084
>
```

Compared the result with valid set, the mean absolute error(MAE) increase a little but not too much. The difference RMSE between test and valid data is 26.25 as we didn't exclude the outlier in some variables. Thus, we focus on  MAE. The reason for this is that MAE is an easily interpretable metric and more robust to outliers compared to RMSE. Thus, we believe the prediction with this model.

```
> pre2 <- predict(MLR,valid)
> valid_accruacy <- accuracy(pre2,valid$price)
> valid_accruacy
                ME     RMSE      MAE       MPE     MAPE
Test set 7.780678 70.98425 38.16444 -10.02983 31.04067
>
```

# Step III: Classification

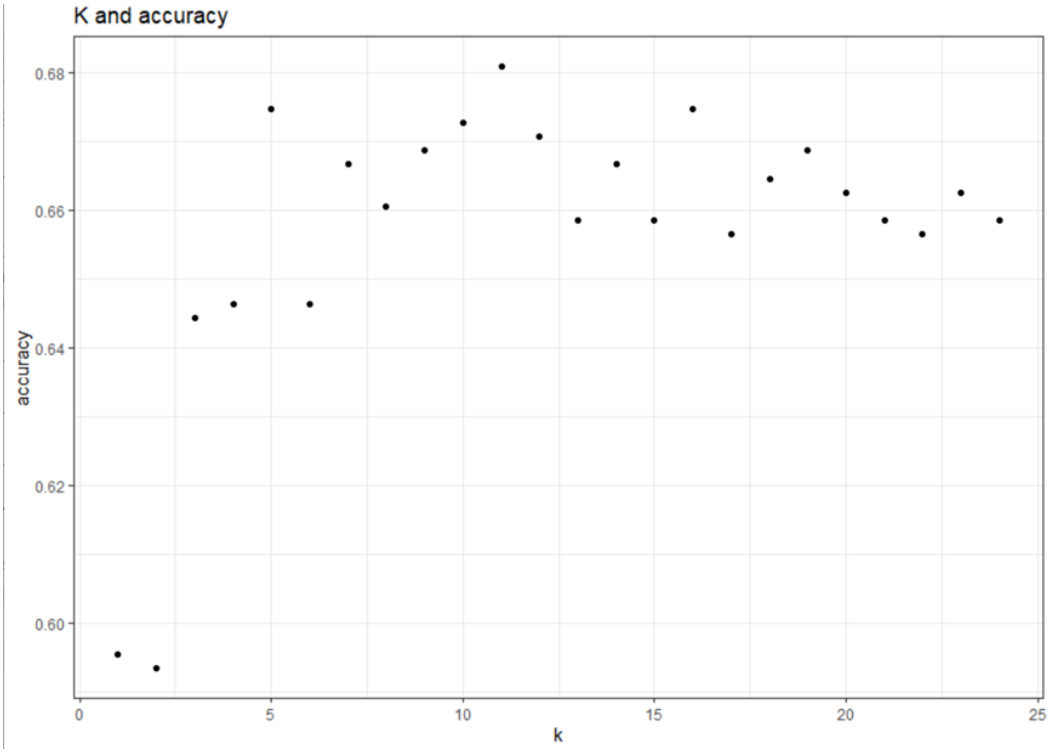
## I. K-nearest neighbors

```
> data <- read_csv("C:/Users/41141/Desktop/699 final/paris_listings.csv")
Rows: 49429 Columns: 74

-- Column specification --------------------------------------------------------
Delimiter: ","
chr (23): listing_url, name, description, neighborhood_overview, picture_url, hos
dbl (43): id, scrape_id, last_scraped, host_id, host_since, host_listings_count,
lgl  (8): host_is_superhost, host_has_profile_pic, host_identity_verified, neighb

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
Warning message:
One or more parsing issues, see `problems()` for details
> Airbnb_Bastille <- data %>% filter(host_neighbourhood == "Bastille")
> sum(is.na(Airbnb_Bastille))
[1] 7834
> Airbnb_Bastille[Airbnb_Bastille=="N/A"]<-NA
> sum(is.na(Airbnb_Bastille))
[1] 10040
> NA_data <- t(data.frame((lapply(lapply(Airbnb_Bastille, is.na), sum))))
> plot_missing(Airbnb_Bastille)
> str(Airbnb_Bastille)
spec_tbl_df [1,232 x 74] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ id                                          : num [1:1232] 32082 34453 38650
 $ listing_url                                 : chr [1:1232] "https://www.airbn
bnb.com/rooms/34453" "https://www.airbnb.com/rooms/38650" "https://www.airbnb.co
 $ scrape_id                                   : num [1:1232] 2.02e+13 2.02e+13
 $ last_scraped                                : num [1:1232] 44539 44538 44539
 $ name                                        : chr [1:1232] "BEAUTIFUL APARTME
ra Bastille" "Bastille/10m2 à la Roquette" "Stylish & Design 1BR flat- Voltaire
 $ description                                 : chr [1:1232] "<b>The space</b><
 place des Vosges and the Marais,in the lively Bastille nei"| __truncated__ "My
nt on the 1st floor of a building typical of the XIX th century Paris arc"| __tr
is studio has been entirely renovated. It is brand new.<br />The studio was a fo
ght and redesigned completely my flat to make it my comfy nest. Located in the E
__ ...
 $ neighborhood_overview                       : chr [1:1232] NA "Bastille is a
 animated, in the center of Paris with lots of cafes and resta"| __truncated__ N
 (11th), in a quiet and green courtyard where you can listen to sparrows singing
 $ picture_url                                 : chr [1:1232] "https://a0.muscac
ginal.jpg" "https://a0.muscache.com/pictures/653066/ed7af65a_original.jpg" "http
95/8fd16da7_original.jpg" "https://a0.muscache.com/pictures/1659858e-41a7-4c15-b
 $ host_id                                     : num [1:1232] 138401 148773 1536
```

```
> knn_data_origin <- Airbnb_Bastille %>% select(host_response_rate,
+                                               host_acceptance_rate,
+                                               host_listings_count,
+                                               host_total_listings_count,
+                                               accommodates,
+                                               bathrooms_text,
+                                               bedrooms,beds,price,
+                                               minimum_nights,maximum_nights,
+                                               amenities)
> knn_data <- knn_data_origin
> knn_data$host_response_rate <- as.numeric(knn_data$host_response_rate)
> knn_data$host_acceptance_rate <- as.numeric(knn_data$host_acceptance_rate)
> table(knn_data$bathrooms_text)

         0 baths           1 bath   1 private bath    1 shared bath       1.5 baths
               3              960              41               71              54
1.5 shared baths          2 baths   2 shared baths       2.5 baths         3 baths
               5               65                1              13              10
       3.5 baths          4 baths        Half-bath
               1                2                3
> knn_data$bathrooms_text <- ifelse(knn_data$bathrooms_text == "Half-bath",
+                                   "0.5",
+                                   knn_data$bathrooms_text)
```

```
> knn_data$bathrooms_text <- gsub('baths','',knn_data$bathrooms_text)
> knn_data$bathrooms_text <- gsub('bath','',knn_data$bathrooms_text)
> knn_data$bathrooms_text <- gsub('shared','',knn_data$bathrooms_text)
> knn_data$bathrooms_text <- gsub('private','',knn_data$bathrooms_text)
> table(knn_data$bathrooms_text)

   0    0.5    1    1    1.5  1.5      2    2    2.5     3   3.5     4
   3     3   960   112    54     5    65     1    13    10     1     2
> knn_data$bathrooms_text <- as.numeric(knn_data$bathrooms_text)
> knn_data$host_response_rate[is.na(knn_data$host_response_rate)] <- median(knn_data$host_response_rate,na.rm=TRUE)
> knn_data$host_acceptance_rate[is.na(knn_data$host_acceptance_rate)] <- median(knn_data$host_acceptance_rate,na.rm
=TRUE)
> #anyNA(knn_data)
> data.frame((lapply(lapply(knn_data, is.na), sum)))
  host_response_rate host_acceptance_rate host_listings_count host_total_listings_count
1                  0                    0                   0                         0
  accommodates bathrooms_text bedrooms beds price minimum_nights maximum_nights amenities
1            0             3      249   38     0              0              0         0
> knn_data$beds <- ifelse((is.na(knn_data$beds)),
+                         knn_data$accommodates,
+                         knn_data$beds)
> knn_data$bedrooms <- ifelse((is.na(knn_data$bedrooms)),
+                          knn_data$accommodates/2,
+                          knn_data$bedrooms)
> knn_data$amenities <- knn_data_origin$amenities
> knn_data$amenities <- gsub('[""]',' ',knn_data$amenities)
> knn_data$amenities <- gsub('[[]',' ',knn_data$amenities)
> knn_data$amenities <- gsub('[]]',' ',knn_data$amenities)
> knn_data$amenities <- gsub('[ ]','',knn_data$amenities)
> #const <- as.data.frame(knn_data$amenities)
> const <- as.data.frame(gsub('[ ]','',knn_data$amenities))
> colnames(const)[1] <- "text"
> tidy_const <- const %>% unnest_tokens(word, text) #%>% anti_join(stop_words)
> tidy_const %>% count(word, sort = TRUE) %>% head(50)
                         word    n
1                        wifi 1170
2                  essentials 1165
3                     kitchen 1160
4                     heating 1155
5       longtermstaysallowed 1071
6                      washer  931
7                     hangers  881
8                   hairdryer  869
9                  smokealarm  867
10                       iron  855
> #tidy_const %>% count(word) %>% with(wordcloud(word, n, max.words=50))
> knn_data$amenities <- grepl(pattern = "Wifi.*Kitchen.*Essentials|Kitchen.*Wifi.*Essentials|
+                                       Kitchen.*Wifi.*Essentials|Kitchen.*Essentials.*Wifi|
+                                       Essentials.*Kitchen.*Wifi|Essentials.*Wifi.*Kitchen",
+                                       knn_data$amenities)
> knn_data$amenities <- as.factor(knn_data$amenities)
> table(knn_data$amenities)

FALSE  TRUE
  661   571
> #data partition
> anyNA(knn_data)
[1] FALSE
```

```
> set.seed(699)
> n_knn_data <-slice_sample(knn_data, prop = 1)
> train_knn <- slice_head(n_knn_data, prop=0.6)
> valid_knn <- slice_tail(n_knn_data, prop=0.4)
> train_knn.norm <-train_knn
> valid_knn.norm <-valid_knn
> norm<-preProcess(train_knn.norm[,-c(12)], method=c("center", "scale"))
> train_knn.norm[,-c(12)] <- predict(norm, train_knn[,-c(12)])
> valid_knn.norm[,-c(12)] <- predict(norm, valid_knn[,-c(12)])
> accuracy.df <- data.frame(k = seq(1, 24, 1), accuracy = rep(0, 24))
> for(i in 1:24) {
+   knn.pred <- knn(train_knn.norm[,-c(12)], valid_knn.norm[,-c(12)],
+                   cl = train_knn.norm$amenities, k = i)
+   accuracy.df[i, 2] <- confusionMatrix(knn.pred, valid_knn.norm$amenities)$overall[1]
+ }
> accuracy.df
    k  accuracy
1   1 0.5955285
2   2 0.5934959
3   3 0.6443089
4   4 0.6463415
5   5 0.6747967
6   6 0.6463415
7   7 0.6666667
8   8 0.6605691
> accuracy.df %>%filter(accuracy==max(accuracy))
   k  accuracy
1 11 0.6808943
```




```
> testing <- valid_knn.norm[c(x),]
> view(testing)
> x <- round(runif(1, min=0, max=nrow(valid_knn.norm)))
> x
[1] 204
> testing <- valid_knn.norm[c(x),]
> view(testing)
```

```
> testing <- valid_knn.norm[c(x),]
> view(testing)
> nn <- knn(train_knn.norm[1:11], test = testing[1:11], cl = train_knn$amenities, k = 1
1)
> nn
[1] FALSE
attr(,"nn.index")
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
[1,]  154  633  295  468  659  596  669  258  544    41   520
attr(,"nn.dist")
          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
[1,] 0.8637098 1.153357 1.248748 1.24954 1.275117 1.346167 1.403597 1.431455
         [,9]    [,10]    [,11]
[1,] 1.433811 1.460524 1.464048
Levels: FALSE
> row.names(train_knn)[attr(nn,"nn.index")]
 [1] "154" "633" "295" "468" "659" "596" "669" "258" "544" "41"  "520"
> neighbors <- Airbnb_Bastille[c(697,499,345,557,365,515,88,252,380,381,5),]
> View(neighbors)
```

| | id | listing_url | scrape_id | last_scraped | name | description |
|---|---|---|---|---|---|---|
| 1 | 19398667 | https://www.airbnb.com/rooms/19398667 | 2.021121e+13 | 44539 | Central and cosy apartment near Bastille 4 people | Located in the heart of pari |
| 2 | 13227605 | https://www.airbnb.com/rooms/13227605 | 2.021121e+13 | 44538 | Cozy and Stylish Duplex in a Trendy area of Paris | Located near metro Voltaire |
| 3 | 8519822 | https://www.airbnb.com/rooms/8519822 | 2.021121e+13 | 44539 | Loft avec terrasse  dans passage privé | Cet ancien atelier d artiste c |
| 4 | 14243657 | https://www.airbnb.com/rooms/14243657 | 2.021121e+13 | 44539 | Beautiful Apt close to Bastille. Sleeps 4 | Freshly renovated flat, situa |
| 5 | 9058536 | https://www.airbnb.com/rooms/9058536 | 2.021121e+13 | 44539 | MyPrettyFlat | Hello everyone! <br /><br |
| 6 | 13500398 | https://www.airbnb.com/rooms/13500398 | 2.021121e+13 | 44538 | 70 m² 2-bedroom appartment near Bastille | My accommodation is close |
| 7 | 1496750 | https://www.airbnb.com/rooms/1496750 | 2.021121e+13 | 44539 | Cosy place! Bastille, Marais' Gates | The space<br /> |
| 8 | 6265263 | https://www.airbnb.com/rooms/6265263 | 2.021121e+13 | 44538 | Cool Roomate at Bastille/Republique | Ideally located in the 11th ( |
| 9 | 9570215 | https://www.airbnb.com/rooms/9570215 | 2.021121e+13 | 44538 | Charming oasis the heart of Paris with AC! | Dans un quartier plein de vi |
| 10 | 9656400 | https://www.airbnb.com/rooms/9656400 | 2.021121e+13 | 44538 | charmant studio/ rue mignonne | Studio de 20 m2 <br /><br |
| 11 | 72427 | https://www.airbnb.com/rooms/72427 | 2.021121e+13 | 44538 | Bastille -Night Life/ Market/ Metro | My humble apartment is th |

```
> neighbors$amenities <- gsub('[""]',' ',neighbors$amenities)
> neighbors$amenities <- gsub('[[]',' ',neighbors$amenities)
> neighbors$amenities <- gsub('[]]',' ',neighbors$amenities)
> neighbors$amenities <- gsub('[ ]','',neighbors$amenities)
> #const <- as.data.frame(knn_data$amenities)
> const <- as.data.frame(gsub('[ ]','',neighbors$amenities))
> colnames(const)[1] <- "text"
> tidy_const <- const %>% unnest_tokens(word, text) #%>% anti_join(stop_words)
> tidy_const %>% count(word, sort = TRUE) %>% head(50)
                    word  n
1              essentials 11
2                 heating 11
3                 kitchen 11
4                    wifi 11
5     longtermstaysallowed 10
6                   washer 10
7                hairdryer  8
8                     iron  8
9                 shampoo  8
10      dedicatedworkspace  7
11                hangers  7
12               hotwater  7
13             smokealarm  7
14           cookingbasics  6
```

The 74 variables in the Paris Listings dataset provide brief background information on Airbnb's operations and data collection framework. However, not all data is applicable to implementing the k nearest neighbors model. Cleaning up the neighborhood data is the first step to start. By exploring the data, We chose host_response_rate,

host_acceptance_rate, host_listings_count, host_total_listings_count, accommodates, Bathrooms_text, bedrooms, beds, price, minimum_nights, maximum_nights, and amenities as the predicting variables. We used the data cleaning method to change bathrooms and amenities variables into num and boolean types for prediction.

After sampling and standardization, we look for the optimal k value. When k=11, the maximum accuracy of the model is 0.68. Therefore, if The k value is set to 11, we find the nearest 11 neighbors of one random record in the validation set and get their amenities. By using grepl() to carry out word frequency analysis, we found that essentials, heating, kitchen, wifi, and long term stay allowed, the washer was most frequently mentioned. According to the results of word frequency analysis, it is not difficult to find that most of the 11 nearest neighbors support long-term renting. So they provide great kitchen and bathroom appliances to accommodate their tenants for long periods.

# II. Naive Bayes

## Building & Testing

First, let us look at the numeric variables and logic variables in the original dataset. Out of these more than forty available variables, we have selected 8 of them plus the result variable as listed in the table below:

| | | |
|---|---|---|
| host_response_time (pred) | host_response_rate (pred) | host_is_superhost (pred) |
| host_identity_verified (pred) | Price (pred) | number_of_reviews (pred) |
| review_scores_rating (pred) | reviews_per_month (pred) | ***instant_bookable (Result)*** |

```
navie_data <- Airbnb_Bastille %>% select(host_response_time,
                                         host_response_rate,
                                         host_is_superhost,
                                         host_identity_verified,
                                         price,
                                         number_of_reviews,
                                         review_scores_rating,
                                         reviews_per_month,
                                         instant_bookable)
```

However, we also noticed that there are too many missing values in [host_response_time] and [host_response_rate]. We dropped these two variables, and other rows that contain "NA" values. The drop of "NA" rows would only reduce 10% of the total sample size. And we believe the drop is reasonable.

```
> data.frame((lapply(lapply(navie_data, is.na), sum)))
  host_response_time host_response_rate host_is_superhost host_identity_verified price
1                757                757                 0                        0     0
  number_of_reviews review_scores_rating reviews_per_month instant_bookable
1                 0                  165               165                 0
```

After tidying up the predictors such as changing them into factor variables and binning the numeric variables, here are the final predictors:

| Predictors | Variable Type: |
|---|---|
| host_is_superhost | Factor w/ 2 levels "FALSE","TRUE" |
| host_identity_verified | Factor w/ 2 levels "FALSE","TRUE": |
| price | Factor w/ 4 levels "low price","average price", "high price","top price" |
| number_of_reviews | Factor w/ 4 levels "low review#", "average review#", "high review#","top review#" |
| review_scores_rating | Factor w/ 4 levels "low rating", "average rating", "high rating","top rating" |
| reviews_per_month | Factor w/ 4 levels "low reviews/m", "average reviews/m", "high reviews/m","top reviews/m" |

*(Low*, average*, high*, top*) = (min, mean – standard deviation, mean, mean + standard deviation, max)*

Next, we created two data sets: training set and validation set

```
set.seed(699)
n_navie_data <- slice_sample(navie_data,prop = 1)
train_nb <- slice_head(n_navie_data, prop=0.7)
valid_nb <- slice_tail(n_navie_data, prop=0.3)
```

Building the Navie model and test its accuracy:

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
    FALSE      TRUE
0.7613941 0.2386059

Conditional probabilities:
       host_is_superhost
Y           FALSE      TRUE
  FALSE 0.7975352 0.2024648
  TRUE  0.7359551 0.2640449

       host_identity_verified
Y           FALSE      TRUE
  FALSE 0.1514085 0.8485915
  TRUE  0.1629213 0.8370787

       price
Y        low price average price high price  top price
  FALSE 0.04416961    0.49116608 0.44169611 0.02296820
  TRUE  0.05617978    0.41573034 0.49438202 0.03370787

       number_of_reviews
Y       low review# average review# high review# top review#
  FALSE   0.2207294       0.2341651    0.4165067   0.1285988
  TRUE    0.2678571       0.2023810    0.3571429   0.1726190

       review_scores_rating
Y       low rating average rating high rating top rating
  FALSE 0.01776199     0.25577265  0.72646536 0.00000000
  TRUE  0.02285714     0.34285714  0.63428571 0.00000000

       reviews_per_month
Y       low reviews/m average reviews/m high reviews/m top reviews/m
  FALSE   0.021126761       0.693661972    0.200704225   0.084507042
  TRUE    0.005617978       0.612359551    0.219101124   0.162921348
```

```
> pred_t <- predict(navie_data_nb, train_nb)
> confusionMatrix(pred_t,train_nb$instant_bookable)$overall[1]
 Accuracy
0.7520107
> pred_v <- predict(navie_data_nb, valid_nb)
> confusionMatrix(pred_v,valid_nb$instant_bookable)$overall[1]
Accuracy
 0.74375
```

## Fictional Apartment

```
test <- data.frame(host_is_superhost = TRUE,
                   host_identity_verified = FALSE,
                   price = "average price",
                   number_of_reviews = "average review#",
                   review_scores_rating = "high rating",
                   reviews_per_month = "average reviews/m")
```

Here is the result:

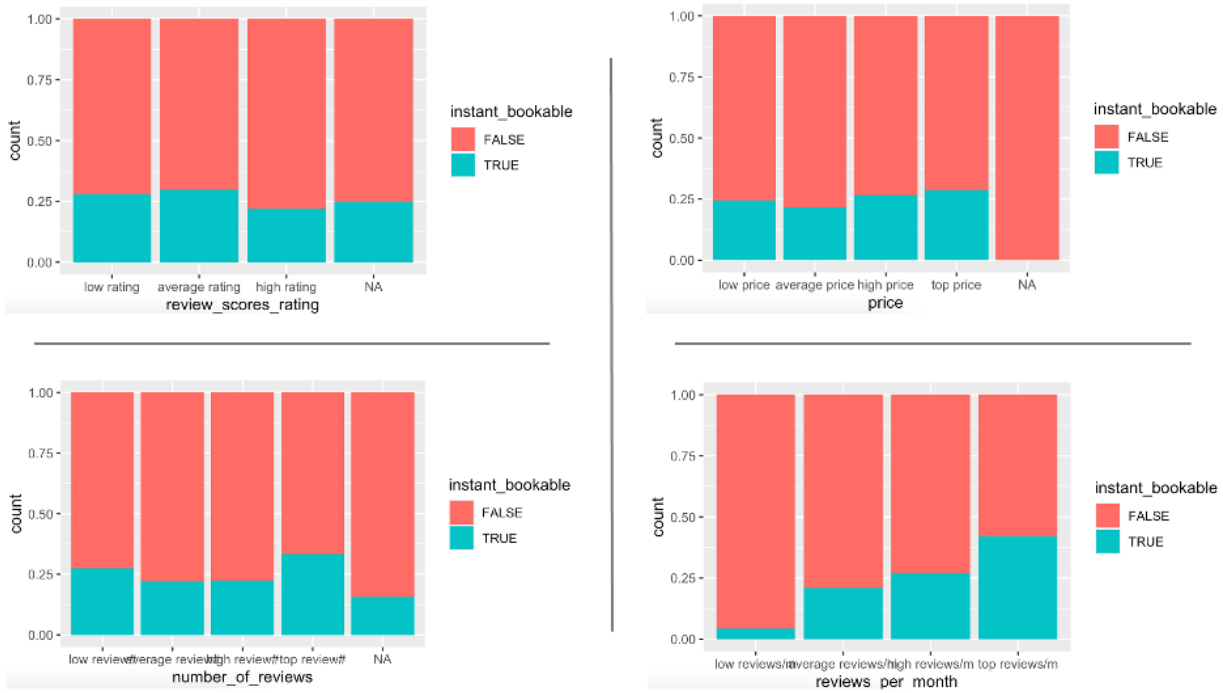
```
> predict(navie_data_nb, test)
[1] FALSE
Levels: FALSE TRUE
> predict(navie_data_nb, test, type="raw")
          FALSE      TRUE
[1,] 0.8013056 0.1986944
```

## Conclusions

The confusion matrix result for the seen dataset has an accuracy of 75%. When the navie model faces unseen data, it has a comparable accuracy of 72%. These two numbers are very close, which showing the model is not overfitting and its predicting power is convincing.

However, when we were checking the partition on each variable against the result (whether an Airbnb is instant bookable), we found that the result of “False” is dominating in every variable in the two datasets (Training and Validation). This could help to explain why the accuracy is so high because most of the Airbnb units in Bastille are not instant bookable.

```
> table(train_nb$instant_bookable)

FALSE  TRUE
  568   178
> table(valid_nb$instant_bookable)

FALSE  TRUE
  240    80
```




But with the combination of "[Super_Host_Yes] + [host_Identified_no] + [High Price] + [Average Rating] + [Low review#] + [Top review/month]", the model says that the Airbnb unit has a 64% chance to be instant bookable. And this number is much better than just random guessing.

```
> 0.23*0.26*0.16*0.49*0.17*0.34*0.16
[1] 4.335758e-05
> 0.76*0.2*0.15*0.44*0.12*0.25*0.08
[1] 2.40768e-05
> 4.335758e-05/(4.335758e-05+2.40768e-05)
[1] 0.6429596
```

# III. Classification tree

## Building & Choosing size

First, we needed to do some cleaning on the original dataset. For example, changing the [bathrooms_text] into readable numeric input:

```
tree_data$bathrooms_text <- ifelse(tree_data$bathrooms_text == "Half-bath",
                                   "0.5",
                                   tree_data$bathrooms_text)
tree_data$bathrooms_text <- gsub('baths','',tree_data$bathrooms_text)
tree_data$bathrooms_text <- gsub('bath','',tree_data$bathrooms_text)
tree_data$bathrooms_text <- gsub('shared','',tree_data$bathrooms_text)
tree_data$bathrooms_text <- gsub('private','',tree_data$bathrooms_text)
```

And removing the variables that contain too many “NA”.

```
tree_data_cleaned <- tree_data_cleaned %>% select(-c(host_response_time,
                                                     host_acceptance_rate))
```

The way that we binned the [review_scores_rating] variable is the same as how we binned predictors in the naïve model:
(Low*, average*, high*, top*) = (min, mean – standard deviation, mean, mean + standard deviation, max)

```
tree_data_cleaned$review_scores_rating <- cut(tree_data_cleaned$review_scores_rating,
                                  c(min(tree_data_cleaned$review_scores_rating),
                                    mean(tree_data_cleaned$review_scores_rating)-sd(tree_data_cleaned$review_scores_rating),
                                    mean(tree_data_cleaned$review_scores_rating),
                                    mean(tree_data_cleaned$review_scores_rating)+sd(tree_data_cleaned$review_scores_rating),
                                    max(tree_data_cleaned$review_scores_rating)),
                                  labels = c("low rating", "average rating", "high rating","top rating"))
```

Building the training and validation datasets:

```
set.seed(699)
n_tree_data_cleaned <-slice_sample(tree_data_cleaned, prop = 1)
train_tree <- slice_head(n_tree_data_cleaned, prop=0.6)
valid_tree <- slice_tail(n_tree_data_cleaned, prop=0.4)
```

So, we determined the CP value as 0.0160550:

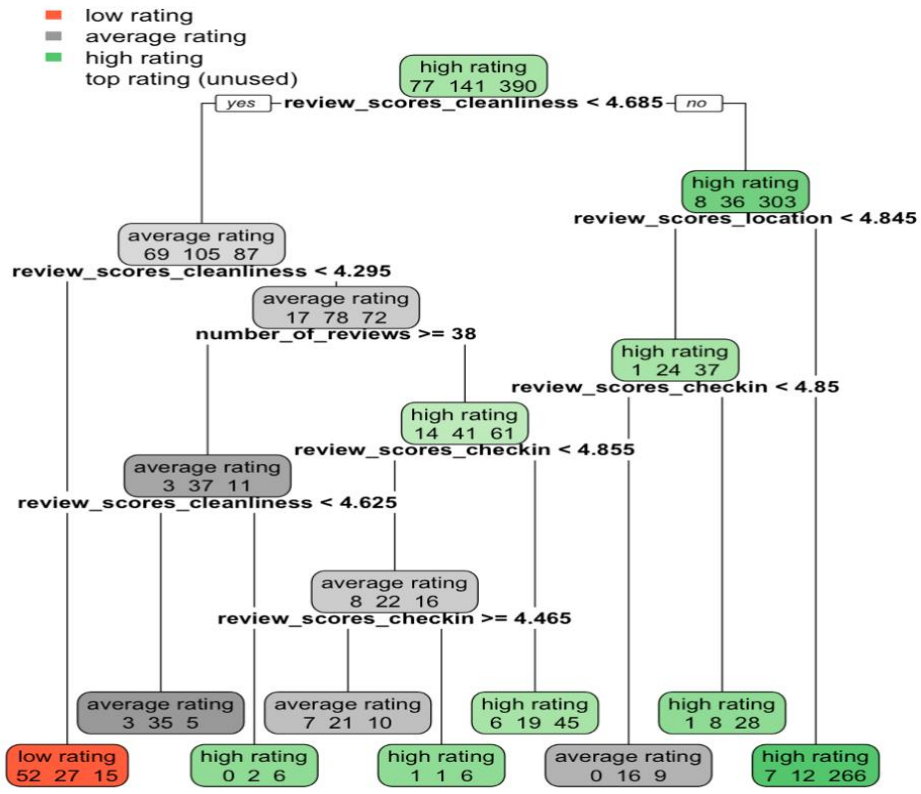
```
model <- rpart(review_scores_rating~.,train_tree,cp = 0.00, xval = 5)
a <- printcp(model)
a <- data.frame(a)
which.min(a$xerror)
> which.min(a$xerror)
[1] 6
```

```
          CP nsplit rel error  xerror     xstd
1  0.0986239      0   1.00000 1.00000 0.054244
2  0.0917431      2   0.80275 0.88532 0.052650
3  0.0275229      3   0.71101 0.81193 0.051383
4  0.0229358      4   0.68349 0.80734 0.051297
5  0.0183486      5   0.66055 0.81193 0.051383
6  0.0160550      6   0.64220 0.79358 0.051034
7  0.0152905      8   0.61009 0.79358 0.051034
8  0.0137615     11   0.56422 0.79358 0.051034
9  0.0091743     12   0.55046 0.81193 0.051383
10 0.0045872     16   0.51376 0.82110 0.051552
11 0.0022936     17   0.50917 0.84404 0.051962
12 0.0000000     19   0.50459 0.85321 0.052120
```

## Modeling

```
class.tree <- rpart(review_scores_rating~.,train_tree,cp = 0.0160550 )
rpart.plot(class.tree, digit = -5, extra = 1, cex = 0.65)
```

Here is the plot:



## Conclusions

Although we included other 11 variables for the tree model such as [price], [room_type], [host_is_superhost], and so on. But the model only uses 4 of them: [review_scores_checkin], [review_scores_location],[review_scores_rating], and [number_of_reviews]. Therefore, to get a good overall rating, quality and quantity of the reviews are equally important.

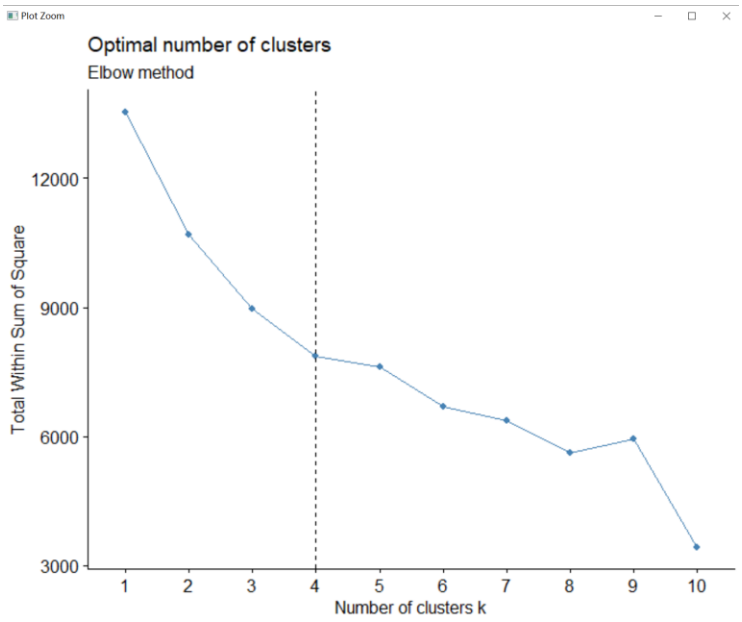
Intuitively, a cleanliness score that is below 4.295 will straightaway drag the overall rating of an Airbnb unit into the “low rating” category, no matter how good other scores are. And the confusion matrix on unseen data also proved predicting power of this tree model since the accuracy on training dataset and validation dataset are pretty close.

```
> class.tree.pred <- predict(class.tree,train_tree,type = "class")
> confusionMatrix(class.tree.pred, as.factor(train_tree$review_scores_rating))$overall[1]
Accuracy
 0.78125
> class.tree.pred2 <- predict(class.tree,valid_tree,type = "class")
> confusionMatrix(class.tree.pred2, as.factor(valid_tree$review_scores_rating))$overall[1]
Accuracy
0.726601
```

# Step IV: Clustering

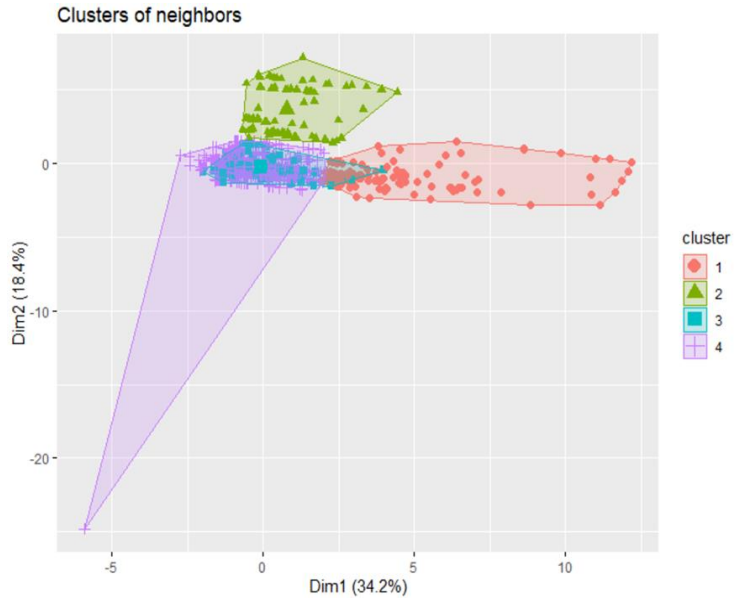
```
> #Clustering
> kmeans_data_origin <- knn_data
> dim(kmeans_data_origin)
[1] 1232   12
> kmeans_data_origin <- subset(kmeans_data_origin, select = -c(12) )
> col_name <- Airbnb_Bastille %>% select(id)
> view(col_name)
> dim(kmeans_data_origin)
[1] 1232   11
> kmeans_data <- cbind(kmeans_data_origin,col_name)
> view(kmeans_data)
> dim(kmeans_data)
[1] 1232   12
> row.names(kmeans_data) <- kmeans_data[,12]
> kmeans_data <- kmeans_data[,-12]
> view(kmeans_data)
> str(kmeans_data)
'data.frame':   1232 obs. of  11 variables:
 $ host_response_rate       : num  1 1 0.8 1 1 1 1 1 1 1 ...
 $ host_acceptance_rate     : num  1 0.905 0.83 0.905 1 0.5 0.47 1 0.77 0.2
5 ...
 $ host_listings_count      : num  1 1 1 2 1 0 1 1 2 1 ...
 $ host_total_listings_count: num  1 1 1 2 1 0 1 1 2 1 ...
 $ accommodates             : num  2 3 2 4 3 6 4 4 3 3 ...
 $ bathrooms_text           : num  1 1 1 1 1 2.5 2 2 1 1 ...
 $ bedrooms                 : num  1 1 1 1 1 3 2 2 1 1 ...
 $ beds                     : num  1 2 1 1 1 4 2 4 2 1 ...
 $ price                    : num  146 60 45 139 50 344 130 250 95 180 ...
 $ minimum_nights           : num  3 3 7 365 6 13 3 5 4 5 ...
 $ maximum_nights           : num  60 30 1125 365 20 ...
> kmeans.norm <- sapply(kmeans_data, scale)
> row.names(kmeans.norm) <- row.names(kmeans_data)
```

```
> kmeans.norm <- sapply(kmeans_data, scale)
> row.names(kmeans.norm) <- row.names(kmeans_data)
> library(factoextra)
> library(NbClust)
> library(cluster)
> fviz_nbclust(kmeans.norm, kmeans, method = "wss") +
+   geom_vline(xintercept = 4, linetype = 2)+
+   labs(subtitle = "Elbow method")
> set.seed(699)
> km <- kmeans(kmeans.norm, centers = 4, nstart=50)
> km$cluster
   32082    34453    38650    40143    72427    97773   182744   252049
       4        4        4        4        4        1        3        1
  314294   314903   419747   440257   470361   475005   496821   509332
       4        3        1        4        4        4        4        1
  514562   530942   541521   554876   590712   599220   607523   612563
       4        1        4        4        4        4        4        4
  615971   618641   621838   652430   675260   676690   678427   693726
       4        4        3        4        3        1        4        3
  693809   713716   717607   718556   728737   740902   747874   748212
       3        4        4        1        4        4        1        4
```




```
> km$centers
  host_response_rate host_acceptance_rate host_listings_count
1          0.1012097          -0.01325006           0.1075294
2          0.1123693           0.24158198           3.0037324
3         -1.6838783          -2.64449291          -0.2374218
4          0.1637934           0.27188080          -0.2676490
  host_total_listings_count accommodates bathrooms_text    bedrooms
1                 0.1075294    2.0332388      2.1916640  2.33375417
2                 3.0037324   -0.1200430     -0.1284603 -0.25758019
3                -0.2374218   -0.1019996     -0.1023247 -0.08064068
4                -0.2676490   -0.2223366     -0.2405923 -0.24802688
         beds       price minimum_nights maximum_nights
1  2.06073366  1.93854553     -0.2164313    -0.09124445
2 -0.02433899  0.25302447     -0.3711010    -0.06455665
3 -0.02336356  0.06490587     -0.3185842    -0.06351457
4 -0.24330993 -0.26431778      0.0960648     0.02406216
```

```
> fviz_cluster(km,kmeans.norm,geom = "point",repel = TRUE,main = "Clusters of neighbors",po
intsize = 2)
```



We use K-means to place rental units within our neighborhood into clusters.
Firstly, we use predictive variables similar to k-nn, because k-means conducts clustering by calculating distance, so we need variables of num. We then went through the normalization data and used elbow methods to find the optimal number of clustering centers. Finally, the clustering center was selected as 4, and PCA and k-means were used to reduce the dimension of the data set with the dimension of 11 to display it on the plane to obtain images of 4 clusters.

```
> x <- as.character(km$cluster)
> class(x)
[1] "character"
> kmeans_data['clusters'] = km$cluster
> view(kmeans_data)
> kmeans_data$clusters <- as.factor(kmeans_data$clusters)
> str(kmeans_data)
'data.frame':   1232 obs. of  12 variables:
 $ host_response_rate       : num  1 1 0.8 1 1 1 1 1 1 1 ...
 $ host_acceptance_rate     : num  1 0.905 0.83 0.905 1 0.5 0.47 1 0.77 0.25 ...
 $ host_listings_count      : num  1 1 1 2 1 0 1 1 2 1 ...
 $ host_total_listings_count: num  1 1 1 2 1 0 1 1 2 1 ...
 $ accommodates             : num  2 3 2 4 3 6 4 4 3 3 ...
 $ bathrooms_text           : num  1 1 1 1 1 2.5 2 2 1 1 ...
 $ bedrooms                 : num  1 1 1 1 1 3 2 2 1 1 ...
 $ beds                     : num  1 2 1 1 1 4 2 4 2 1 ...
 $ price                    : num  146 60 45 139 50 344 130 250 95 180 ...
 $ minimum_nights           : num  3 3 7 365 6 13 3 5 4 5 ...
 $ maximum_nights           : num  60 30 1125 365 20 ...
 $ clusters                 : Factor w/ 4 levels "1","2","3","4": 4 4 4 4 4 1 3 1 4 3 ...

> ggplot(kmeans_data) + geom_bar(aes(x=clusters),color="skyblue3",fill="skyblue3") +
+   xlab("Clusters") +
+   ylab("Number of units") +
+   ggtitle("Number of units in each cluster")+
+   theme(plot.title = element_text(hjust = 0.5))
> ggplot(kmeans_data,aes(y=price,fill=clusters)) +
+   geom_histogram(binwidth=50)+
+   facet_wrap(~clusters)+
+   ggtitle("Price in each cluster")+
+   theme(plot.title = element_text(hjust = 0.5))
> ggplot(kmeans_data,aes(x=bedrooms,y=beds,fill=clusters)) + geom_point()+
+   facet_wrap(~clusters)+
+   ggtitle("Number of bedrooms and beds in each cluster")+
+   theme(plot.title = element_text(hjust = 0.5))
> ggplot(kmeans_data,aes(x=clusters,y=accommodates,fill=clusters)) + geom_violin()+
+   ggtitle("Accommodates in each cluster")+
+   theme(plot.title = element_text(hjust = 0.5))
> ggplot(kmeans_data,aes(x=clusters,y=host_acceptance_rate,fill=clusters)) + geom_violin()+
+   ggtitle("Acceptance rate in each cluster")+
+   theme(plot.title = element_text(hjust = 0.5))
```

Figure 5.1 shows the number of units within each cluster. Cluster1, cluster2 and cluster3 have a similar number of units, about 100 units in each. Cluster 4 has the most significant number of units, about 7 times as many as other clusters.
Figure5.2 shows the price distribution for each cluster. We found that the price of cluster1 is about $200 to $300, which is the highest one, and the highest price in it can

reach up to $1000. Cluster2 and cluster 3 have similar price distribution, with prices concentrated between $50 and $200. Cluster4 has the lowest price, mostly under $125, but more units can be chosen between $125 and $200 than cluster2 and cluster3.

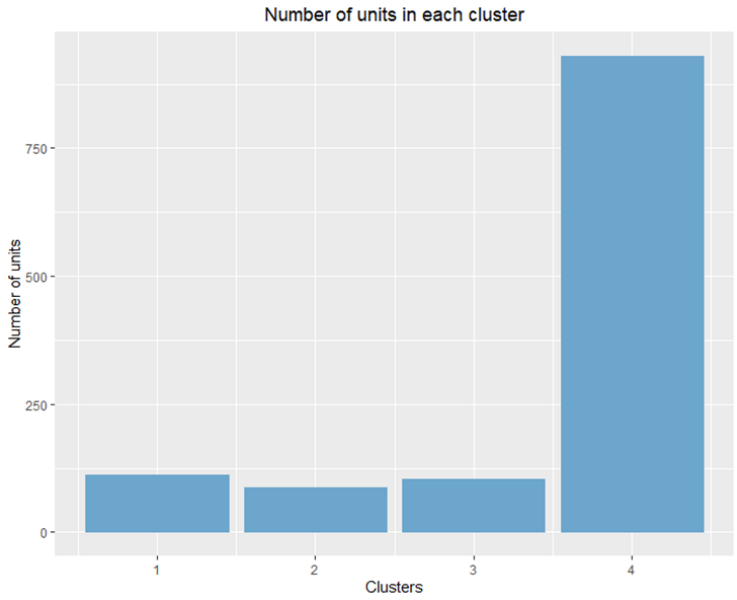



Figure5.1

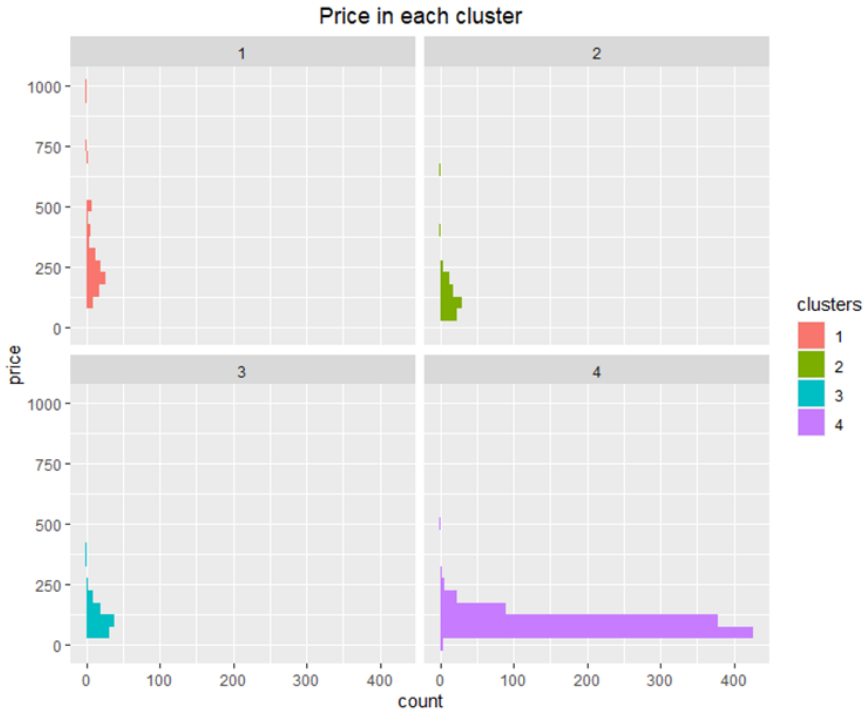



Figure5.2

Figure5.3 shows the distribution of bedrooms and beds in each cluster. Cluster1 has more options for the number of bedrooms and beds, with most having 3 or more bedrooms and

beds. Cluster2 and cluster3 have no more than 2 bedrooms, and most beds are under 3. Most Cluster4 have no more than 2 bedrooms, but there is a wide choice of beds. Figure 5.4 shows the distribution density of each cluster's capacity. Most units of Cluster1 can accommodate 4 to 7 people, or even more than 12 people. Cluster2 is similar to Cluster3, and cluster2 can accommodate 2 to 4 people and evenly distribute with no more than 7 people. Most units in Cluster4 can also accommodate 2 to 4 people, but most of them are 2 people.

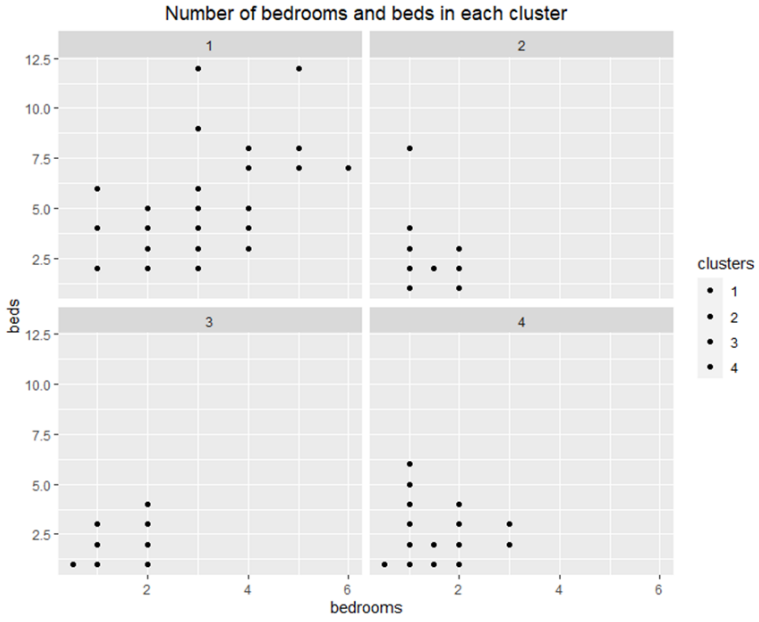



Figure5.3

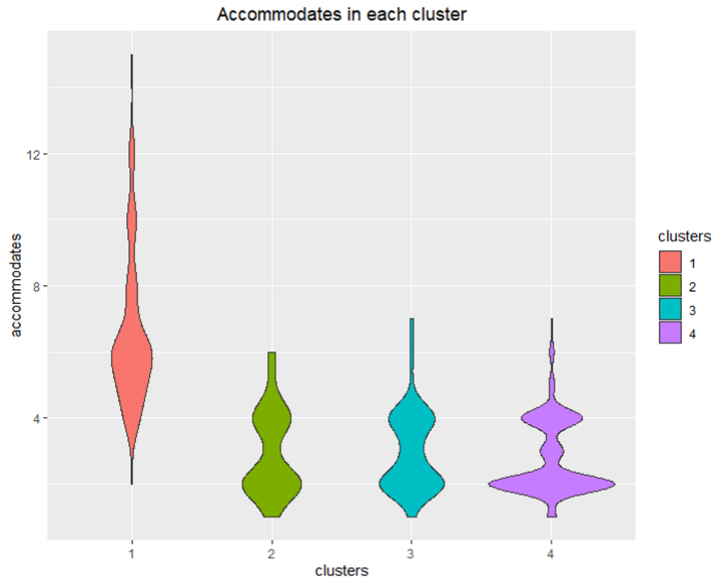



Figure5.4

In order to further explore the differences between Cluster2 and Cluster3, as shown in Figure 5.5, we explored the frequency of host acceptance rate of each cluster. The acceptance rate of Clster1 and Cluster4 is basically around 87%, while the acceptance rate of Cluster2 is concentrated at about 90%. However, the acceptance rate of Cluster3 is meager and generally below 50%.

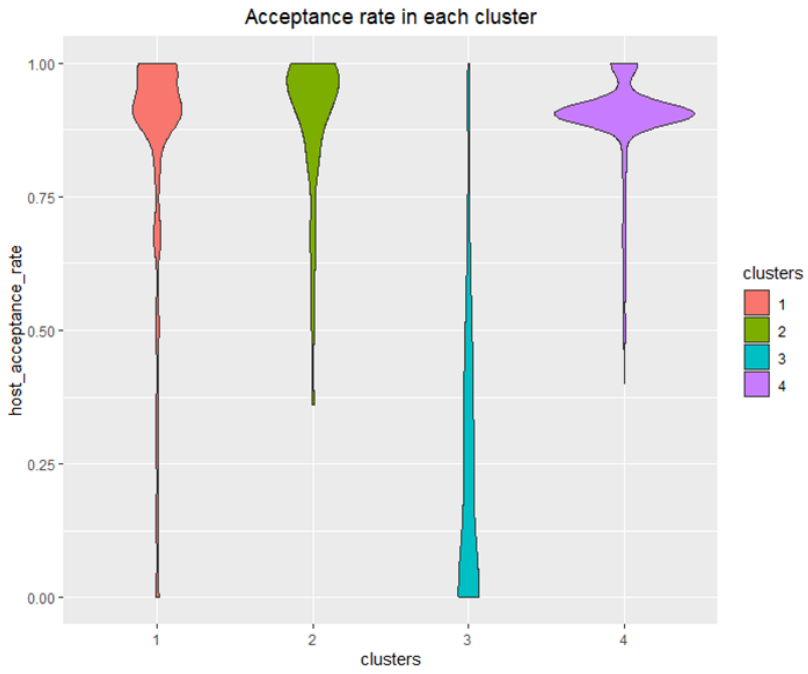



Figure5.5

Based on the above analysis of the four clusters, we decided to name cluster1 as ‘Party size and the choice of the wealthy’, cluster2 as ‘The minority experience of light luxury’, cluster3 as ‘It’s up to hosts’, and cluster4 as ‘The majority with cost-effective’.

# Step V: Conclusions

We first explored the unit data of the Bastille area in Paris, France, and learned that the average score of the units in this area is 4.6, and the average price is $109, but the prices of different units vary greatly. The Entire rental unit is the most available. The price is proportional to the number of units, and 20% of hosts are superhosts. The distribution of units in this area is very concentrated. For the hosts, the high rating and high percentage of superhosts mean that hosts in the Bastille region need to improve service quality to become competitive. A score of 4.6 or above is a good choice for tenants.

Then, we use step()function and select the variables that can be used for prediction according to the minimum AIC value. The VIF value after fitting shows that they do not appear multicollinearity. Finally, the goodness of fit of the model was 0.6615, and the cross validation results were also very close, indicating that the model's prediction results were relatively accurate. The final prediction equation shows as follows:

$$
\begin{aligned}
Price \;=\; & -136.4815 \;-\; 3.2192 \;\times property\_typeEntire\ loft \\
& +\; 0.1011 \times property\_typeEntire\ rental\ unit \\
& +\; 56.3045 \times property\_typeEntire\ residential\ home \\
& -\; 23.9418 \times property\_typePrivate\ room\ in\ rental\ uni \\
& + 10.5853 \times property\_typeRoom\ in\ boutique\ hotel \\
& +\; 12.6786 \times accommodates \;+\; 27.3194 \times\; bedrooms \\
& +\; 2.0317 \times availability\_30 \\
& +\; 11.2893 \times review\_scores\_rating \\
& + 12.0073 \times\; review\_scores\_cleanliness + 49.1352 \times bathrooms
\end{aligned}
$$

In the classification analysis, most long-term rental units in Bastille will provide kitchen and bathroom supplies, wifi, and heating. Meanwhile, most Airbnb listings here can't be booked immediately. At the Bastille, the quality of reviews is as important as the quantity of reviews to get a good overall rating. On the other hand, cleanliness scores below 4.295 will drag the unit's overall rating into the ‘low rating’ category, no matter how good the other scores are. So for hosts to improve their rating, they need to improve the reviews' number, quality, and cleanliness. According to tenants, units in the Bastille must be booked in advance and can be rented for a long time without a kitchen or sanitary essentials.

In the part of clustering, we divided these units into four clusters: first is Party size, second is Light luxury, third is Up to hosts, and fourth is Cost-effective. For the tenants,

if four or more people travel together and want to live in the Bastille, they can choose units in the first category that have more space and allow more people to rest. If the tenant has a sufficient budget for the travel and less than four people together, we recommend choosing the second type of house with a light luxury experience. If tenants are traveling with fewer than four people and don't want to spend too much money on accommodation, we recommend choosing cost-effective housing types. In addition, we do not recommend tenants to select units within ‘Up to hosts’, because the rejection rate of these units is very high. We do not want tenants to waste their precious time. We know that Uber will reduce the number of orders it sends to drivers with high rejection rates. Therefore, we also suggest Airbnb platform remind hosts with high rejection rates and carry out additional qualification verification and other measures.

In a word, through various methods' of analysis and exploration of unit data in the Bastille region, we can say that we have a relatively clear and comprehensive understanding of the online marketplace of homestay houses in this region. And different customer groups, such as tenants, hosts and platforms, can benefit from our analysis results.