# MSc Business Analytics

## GROUP ASSIGNMENT


**Submitted by:** **GROUP 15**

2084551

2082046

2055211

2058395

2048022


**Date Sent:** June 9, 2021

**Module Title:** Text Analytics

**Module Code:** IB9CW0

**Date/Year of Module:** 2021

***"I declare that this work is entirely my own in accordance with the University's Regulation 11 and the WBS guidelines on plagiarism and collusion. All external references and sources are clearly acknowledged and identified within the contents.***

***No substantial part(s) of the work submitted here has also been submitted by me in other assessments for accredited courses of study, and I acknowledge that if this has been done it may result in me being reported for self-plagiarism and an appropriate reduction in marks may be made when marking this piece of work."***

# Table of Contents

**Table of Figures**

## Table of Tables

# Part A – Construction of Corpus –Airbnb Reviews

## Data Preparation


*Figure 1: Pipeline overview for the data preparation stage*

**Pipeline overview for the data preparation stage**

The data was first crawled from the website and loaded into the listings, and reviews data frame using read_csv. The data frames were inspected to check the columns and observations.

**Part I – Reviews dataset**

The data for dates before Feb 2020 was filtered out first. It was observed that a lot of reviews were not in English. Since this could be a problem in removing stop words and in further analysis for sentiments and topic modelling, it was decided to remove the non-English comments using the cld2 dictionary. This was chosen over cld3 because it is faster for large datasets.

```
library(cld2)
reviews_pre_corona$comments <- iconv(reviews_pre_corona$comments)
reviews_pre_corona$language <- cld2::detect_language(reviews_pre_corona$comments)
reviews_pre_corona%>% filter(language =="en") -> reviews_pre_corona
```

Post this, the comments column in the reviews dataset was cleaned for digits and punctuation marks as these don't add much value to the analysis. This was done using the gsub function and the using the pattern search for digit and punctuation marks.

```
# Substitute digits with space using gsub function
reviews_pre_corona$comments <- gsub('[[:digit:]]+',' ', reviews_pre_corona$comments)

# Substitute punctuations with space using gsub
reviews_pre_corona$comments <- gsub('[[:punct:]]+','', reviews_pre_corona$comments)
```

The character length of the reviews is trimmed using Luhn's method. The reviews of more than 1500 characters and less than 100 characters are filtered out. This is to eliminate very long reviews and extremely short reviews. The limits are decided through the histogram inspection and the knowledge of acceptable review lengths for analysis in Amazon.com.

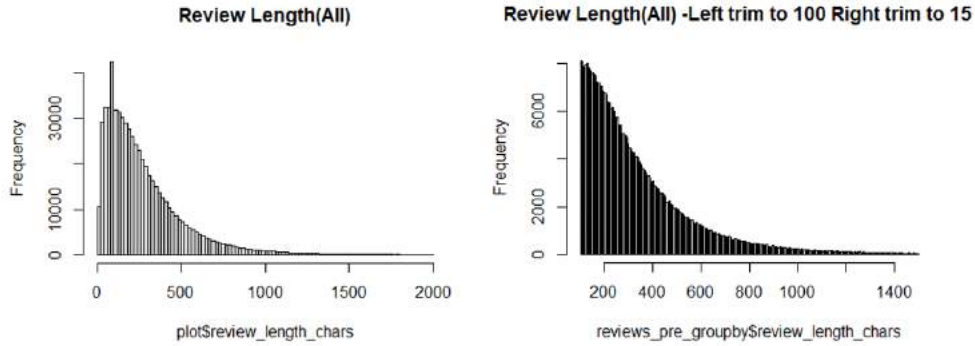



*Figure 2: Distribution of review character length*

**Part II – Listings dataset**

The columns that won't be used for the analysis are removed. These include listing URL, scrape_id, last_scraped etc. The property type, room type variables are converted to factors. The name and description of the property are merged into one column so that this column will have all the information about the property. There are new columns created that has number of amenities, verifications and bathrooms per listing. The price column is converted to numeric, and the "$" sign is removed.

The descriptions in non-English languages are removed, and they are cleaned for digits, punctuation marks and other alphanumeric characters in them. The character lengths of description that are less than 120 and more than 1040 are removed using Luhn's method. This is decided through the observation of the long tail in the histogram.

```
# Filter for the minimum length of 120, 1040

listings_cleaned <- listings_cleaned %>%

filter(listings_length_chars>120) %>% filter(listings_length_chars<1040)
```

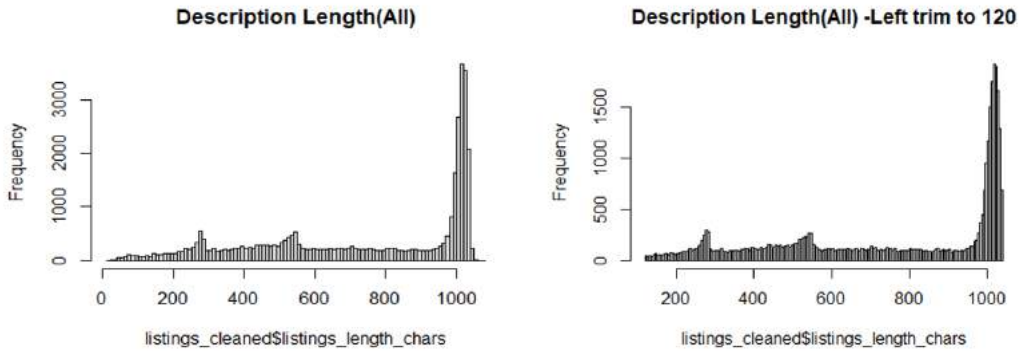



*Figure 3:Distribution of description character length*

Now, the two cleaned datasets reviews and listings are left joined.

```
listings_reviews <- listings_cleaned %>% left_join(reviews_groupbylisting, by = "listing_id")
```

Post this, stop words are removed from the comments and the description columns. This is done by splitting the listing_reviews data frame into 4 chunks of 10,000 and storing them in a list using the split function. Then the unnest tokens and anti-join functions are used in these 4 chunks over a loop to tokenise and remove stop words. After inspecting the tokens, it is observed that some tokens still have repeated letters like "aa" or "bbb", and some have two words without spaces in them like "lovelyplace". These have to be filtered out using the character length. The words with less than 4 and more than 17 characters are filtered out.

Next, the TF/IDF scores are computed for the tokenised words. The document here is defined to be the listing id since it's the primary key of the dataset. The TF/IDF scores below 0.001 and more than 0.05 are filtered out using Luhn's method by inspecting the histogram. Post this, common words in both reviews and descriptions are tabulated. This gives insight into what words occur most frequently in the documents. This is then used to create the custom stop word dictionary for further removal of stop words in the upcoming questions. The following words were common in reviews and descriptions.

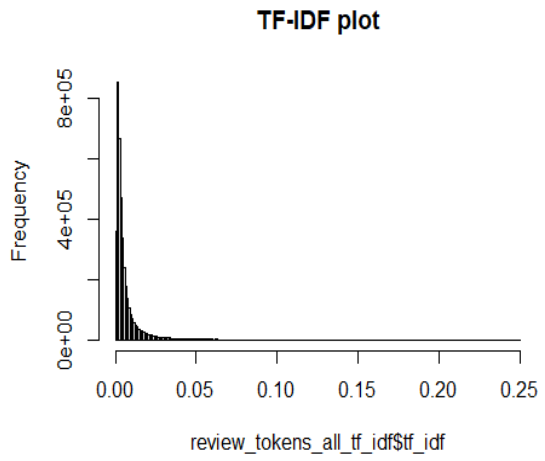



*Figure 4: TF-IDF plot of review tokens- after*

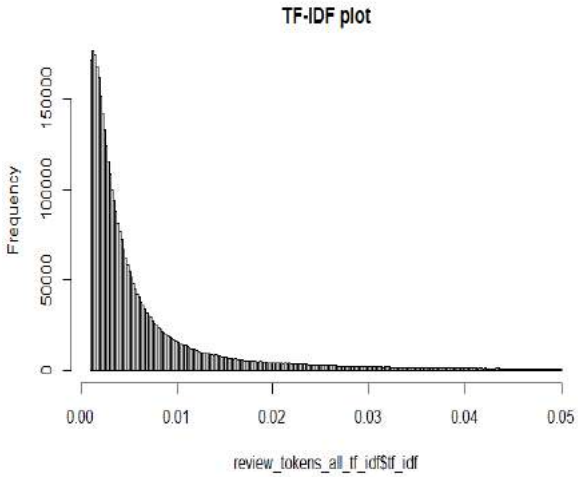



*Figure 5: TF-IDF plot of review tokens- before*

Some common words that qualify as stop words are neighbourhood", "manhattan", "located", "brooklyn", "queens", "staten", "island", "bronx", "stay", "staying".

## Question a – Dominant words

In the question a, we will examine the dominant words of the listing descriptions and customer reviews within two main group categories, the neighbourhood group cleansed and the room type.

**Aggregation by Neighbourhoods**

There are 5 neighbourhood groups in our data, namely Brooklyn, Manhattan, Queens, Staten Island, Bronx. To check the dominant word of reviews for each listing, we first get a list of neighbourhood groups and then use the review tokens cleaned to calculate the frequency of each token for each review or description of each listing. The stopwords and tokens with extreme TF-IDF score are removed before calculating the word frequency.

A for loop is created to get the top 10 frequent tokens in each neighbourhood group. The tokens are ranked based on their frequencies showing in reviews. To better capture the dominant word, we use bar charts to plot the results based on word frequency.

From the bar chart, we can find there are common terms in the top10 frequent words among 5 neighbourhood groups. The common words are "apartment", "clean", "location", "host", "nice", "comfortable" and "recommend". "Apartment" indicates the property type of the listing while "clean" and "location" show customers' focus (e.g. cleanliness, location) when they rate a listing. "Subway" is missing in the top 10 words of Staten Island. Considering the geographical features of Staten Island, the subway might not be the main transportation method for people staying at Staten Island. Besides, "house" appears more than "apartment" in the reviews regarding the Staten Island neighbourhood, implying the main property type on Staten Island is house.

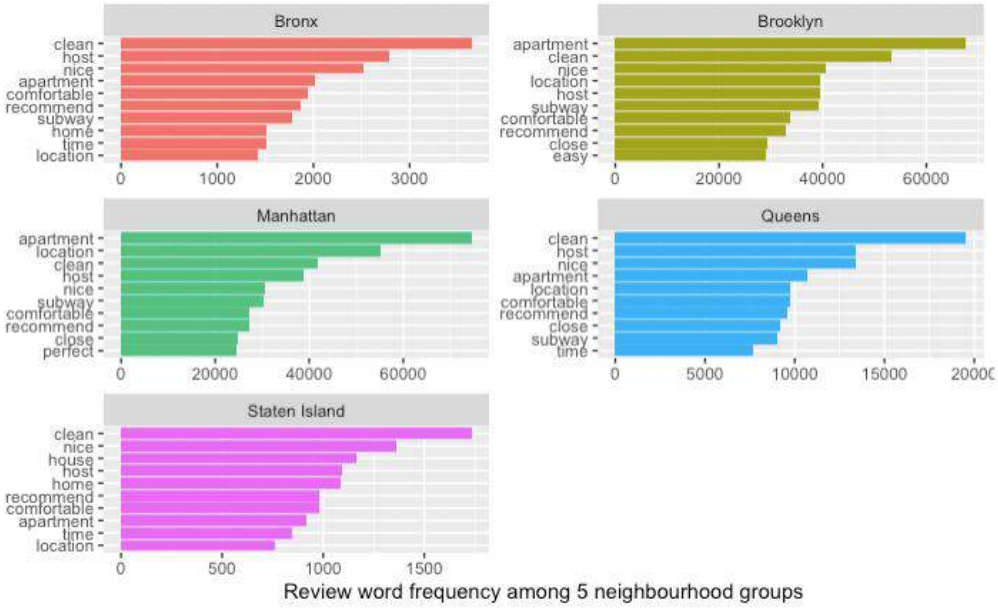



*Figure 6: Review word frequency among 5 neighbourhood groups*

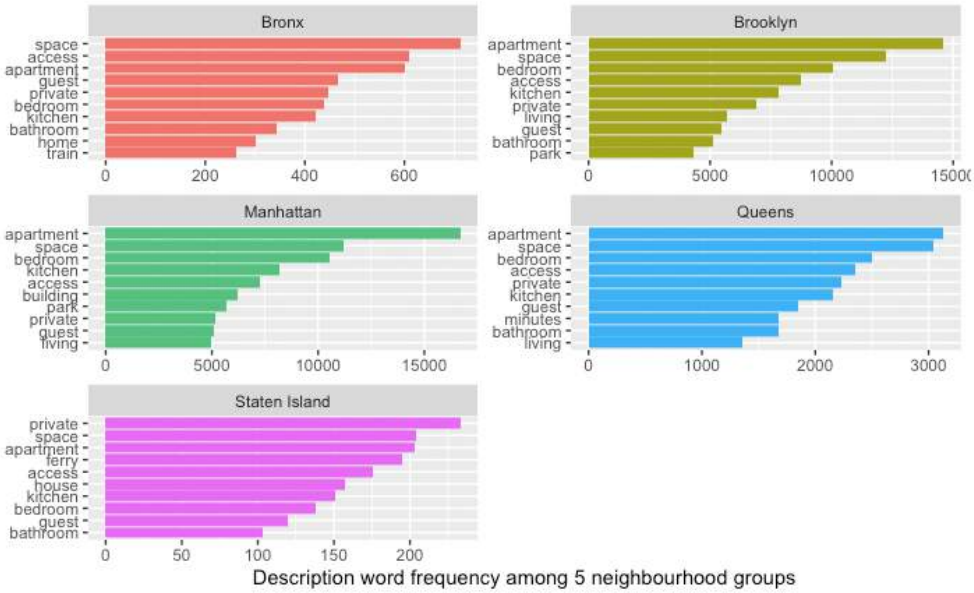



*Figure 7: Description word frequency among 5 neighbourhood groups*

For descriptions, the common words in reviews of 5 neighbourhood groups are "apartment", "space", "bedroom", "access", "kitchen", "private", "guest". We can conclude that the number of rooms and if there is private space are the main focus when hosts describe their listings. "ferry" becomes a unique frequent word in the reviews towards Staten Island, which also shows the different transportation methods in this neighbourhood group.

We then checked the word frequency in 5 neighbourhoods with the most listings within each neighbourhood group. Figure 8 shows the word frequency in reviews of the top 25 neighbourhoods. The figure of description the top words of each neighbourhood are basically consistent with terms of their parent neighbourhood group, while some specific words might indicate the special attraction of that neighbourhood. For example, "restaurant" has a high frequency in neighbourhood East Village and Ditmars Steinway. The description word frequency among 25 neighbourhoods is included in the Appendix.

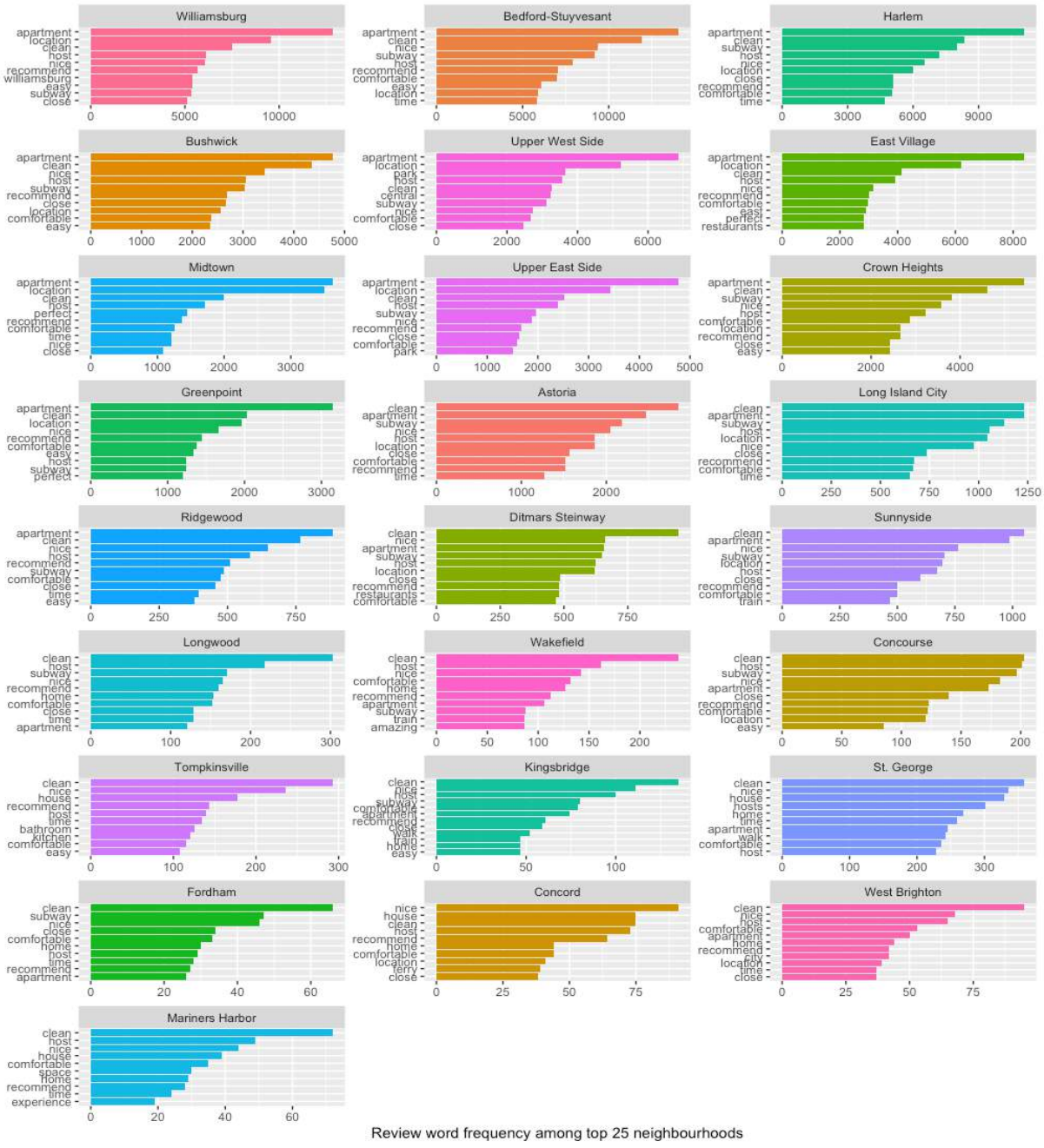



*Figure 8: Review word frequency among top 15 neighbourhoods*

**Aggregation by Room type**

The dominant words in reviews and descriptions do not differ much among 4 room types and from top words generated by neighbourhoods. The word "staff" frequently shows in both the reviews and descriptions of hotel rooms. We could infer that for listing belonging to hotel rooms, the host and the customers care about the hotel services provided by staff, which could be a unique selling point for

hotel rooms.

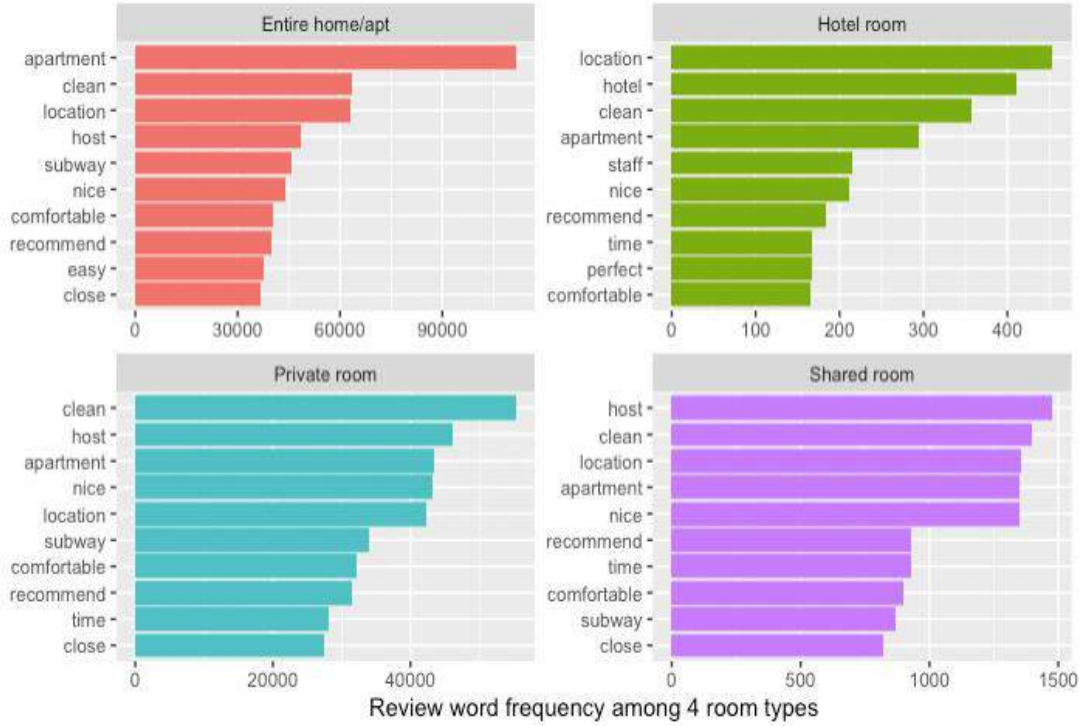



*Figure 9: Review word frequency among 4 room types*

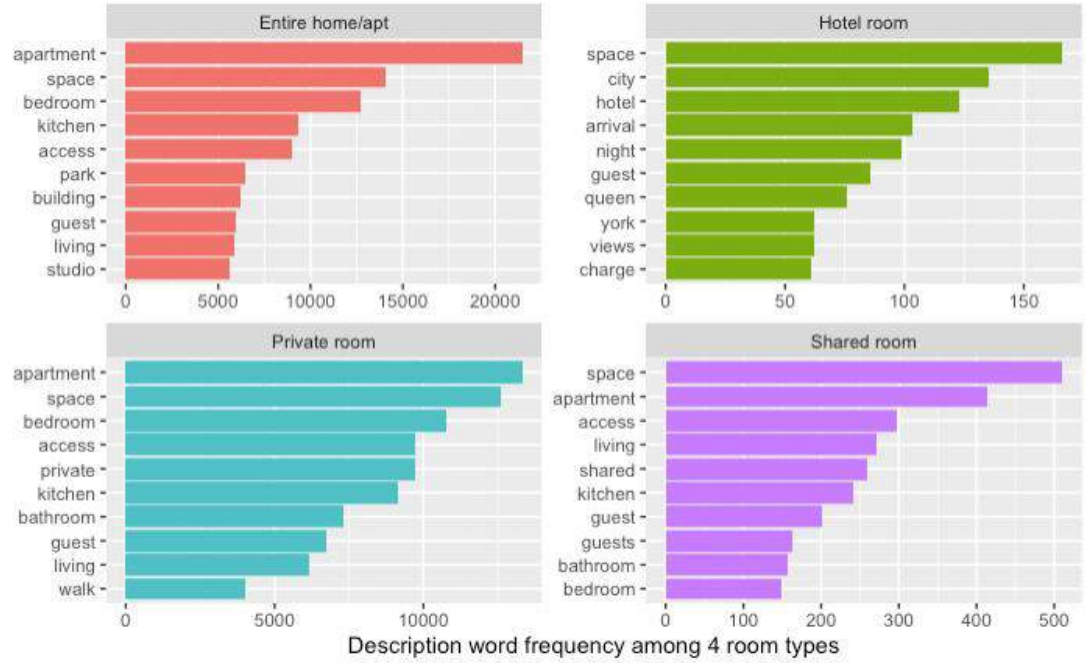



*Figure 10: Description word frequency among 4 room types*

## Question b – Word Combinations

In this section, we focus on the word combinations used to describe the listings. The hosts describe the listings in the 'description' column, and the guests describe the listings in the 'comments' column and elaborate their stay. In the following analysis, we have addressed the following aspects of the scope:

1. Which words/ word combinations are used to describe the listing by the host?
    a) What are the average rating and price for these frequently used terms?
    b) Are these word combinations different if the host is "superhost"? If yes, how?
2. Which word/ word combinations are used by the guests to describe their experience in the listings?
    a) What are the average price and rating for these commented words/ word combinations?
3. Can we observe the correlation between these frequently occurring terms?

**Data Preparation**

For this analysis, we needed reviews per listing. Hence, we grouped the reviews per listing and joined it with the listings dataset. We filter the reviews data to keep only English comments and filtered out the abnormally long or short comments. Then we grouped the reviews per listing id and joined it with the listings dataset. For this analysis, one of the features used is whether a listing has a host who is "super host", so we assumed that the missing cells in this column are not "super host". Next, we checked and handled the inappropriate data and missing data for all relevant columns as shown in the subsection' a'. We saved the cleaned and tidy format as 'df' for this section.

**Tokenisation**

For the 'description' and 'comments' columns, we have chosen to analyse uni-grams, bi-grams, trigrams and quad-grams. In order to filter only relevant tokens, we have removed the custom list of 'stopwords' which do provide useful information about the textual content. In the case of ngrams = 2 to 4, we have chosen to filter out tokens that start or end with words within the 'custom stopwords'.

After tokenisation, we check the top 10 frequently occurring tokens and compute the average price and ratings for listings where the token appears (this is done for both the description and comments column).

**Observations from the Plots**

As per the data provided, the data is not uniformly distributed w.r.t review ratings and prices. The ratings are generally in the higher range, and the prices are right-skewed since there are fewer listings on the expensive side as expected.

From the unigrams, we can see that the hosts explain mostly about the **apartment**, **spaces** for guests and several other perks of their listing as expected. By comparing the prices for these terms, we can see that hosts which provide **private** rooms or spaces are rated the highest, without being on the relatively expensive side of the price distribution. From the Bigram frequency plots, we can observe that guests prefer listings that are **newly renovated** and within **walking distance** of important places around. It is also clear that descriptions with a touch of homeliness (**home away from home**) get higher ratings than others.

Following are the plots for the Description Column:

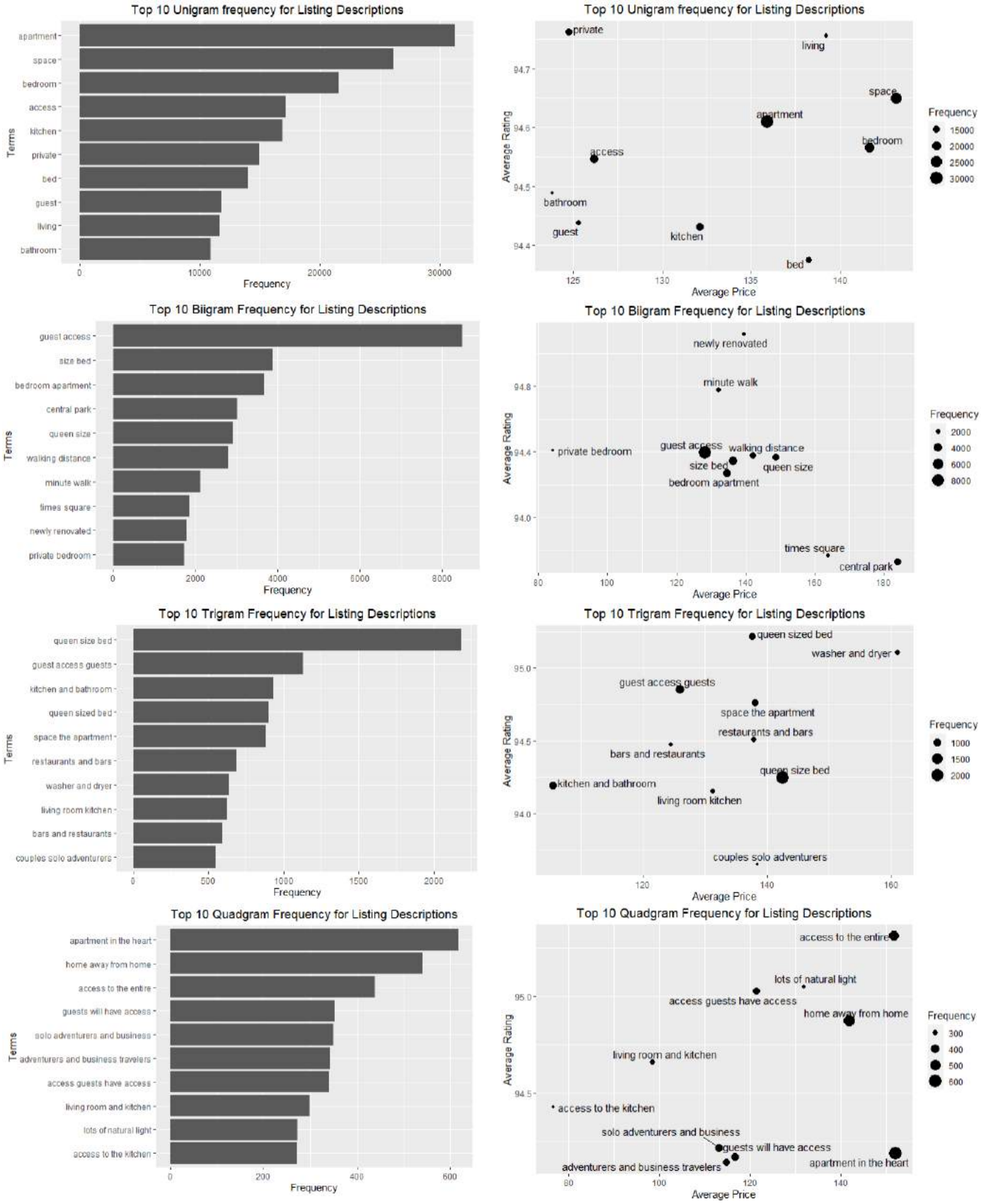



*Figure 11: Top description word combination frequency*

From the figure below, we can check if the super hosts are using something special since they are relatively more experienced in the scenario and if 'non-superhosts' can change their wordings for a higher rating. We can conclude that 'super hosts have perks of **private** spaces and work the extra mile of having **tree linings** around the space, which differentiates them from the relatively new hosts on Airbnb. However, further research is required for checking this aspect of the analysis since most of the wordings used in the descriptions are similar regardless of being super host or not.

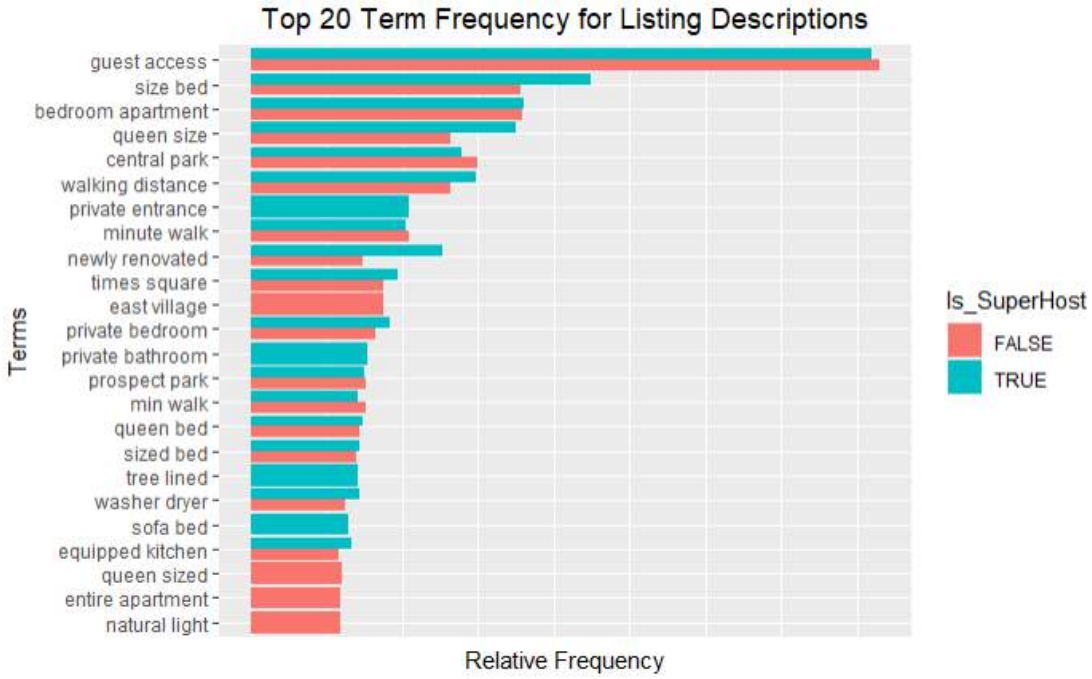
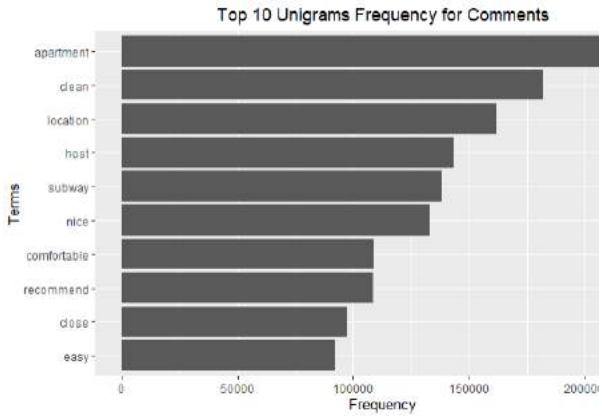
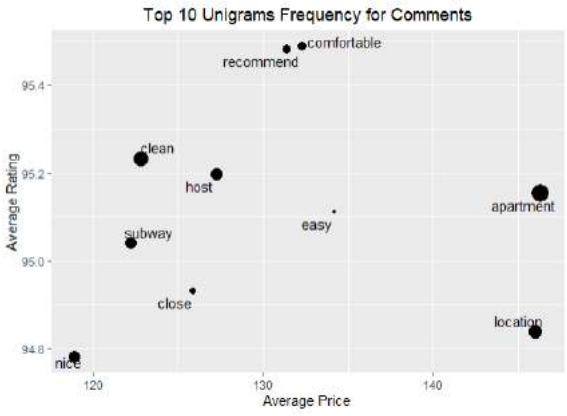



*Figure 12: Top 20 term frequency for listing descriptions*

From the unigrams for comments, most of the comments focus on the **location** and **cleanliness** of the listing. The guests have clearly rated highly where they used **comfortable** (**feel at home, home away from home**) as the word to describe the listing. As discussed earlier, guests also rated highly for listings with **easy access** and within walking distance (**short walk, minute walk**) from places around (**subway station, central park**).



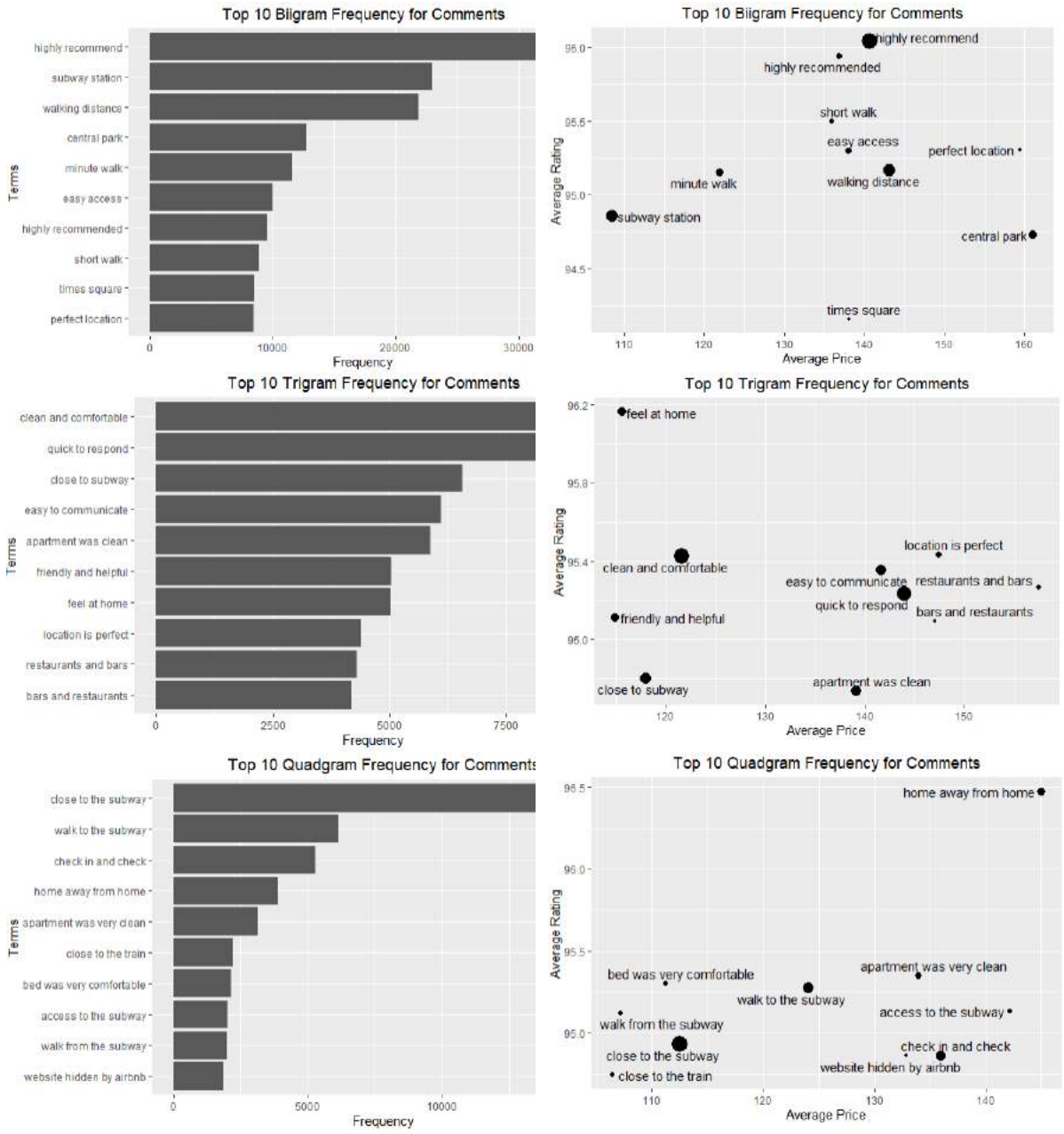



*Figure 13: Top comment word combination frequency*

The above inferences can be confirmed with word correlations and check which words generally occur together. This is a very crude way of estimating topics in the corpus. For example, the word 'recommend' is used frequently, so checking which words highly correlate with this can give us an idea about what are the things users are recommending.

```r
findAssocs(dtm_1_c,"recommend",corlimit = 0.7)
```

```
$recommend
       highly   comfortable         clean          time          nice          easy       helpful         close
         0.92          0.91          0.90          0.90          0.89          0.88          0.88          0.87
      perfect       enjoyed       stayed        subway          city           day    experience           nyc
         0.87          0.85          0.85          0.85          0.84          0.84          0.84          0.84
        super          trip          walk          host       airbnb         visit       walking     wonderful
         0.84          0.84          0.84          0.83          0.82          0.82          0.82          0.82
accommodating    convenient     excellent         short     extremely     apartment          feel      friendly
         0.81          0.81          0.81          0.81          0.81          0.80          0.80          0.80
       loved            ny           lot       amazing      location     questions   recommended           bed
         0.80          0.80          0.80          0.79          0.79          0.79          0.79          0.78
       check          days          home        lovely         quiet          safe         space    absolutely
         0.78          0.78          0.78          0.78          0.78          0.78          0.78          0.78
      access          lots       awesome communication      distance       minutes        nearby         night
         0.78          0.78          0.77          0.77          0.77          0.77          0.77          0.77
   fantastic       arrived         comfy          love         quick     welcoming      provided   restaurants
         0.77          0.76          0.76          0.76          0.76          0.76          0.76          0.76
        cozy         local        minute          warm      visiting   appreciated       arrival        easily
         0.75          0.75          0.74          0.74          0.74          0.73          0.73          0.73
       found         happy          late        nights       station           eat          food neighbourhood
         0.73          0.73          0.73          0.73          0.73          0.73          0.73          0.73
      plenty        couple       explore        people          spot        street          book        helped
         0.73          0.72          0.71          0.71          0.71          0.71          0.71          0.71
         bit        future        public
         0.70          0.70          0.70
```

*Figure 14: Correlation result of word combination*

This is further visualised in a graphical way to get insights about the word correlations:

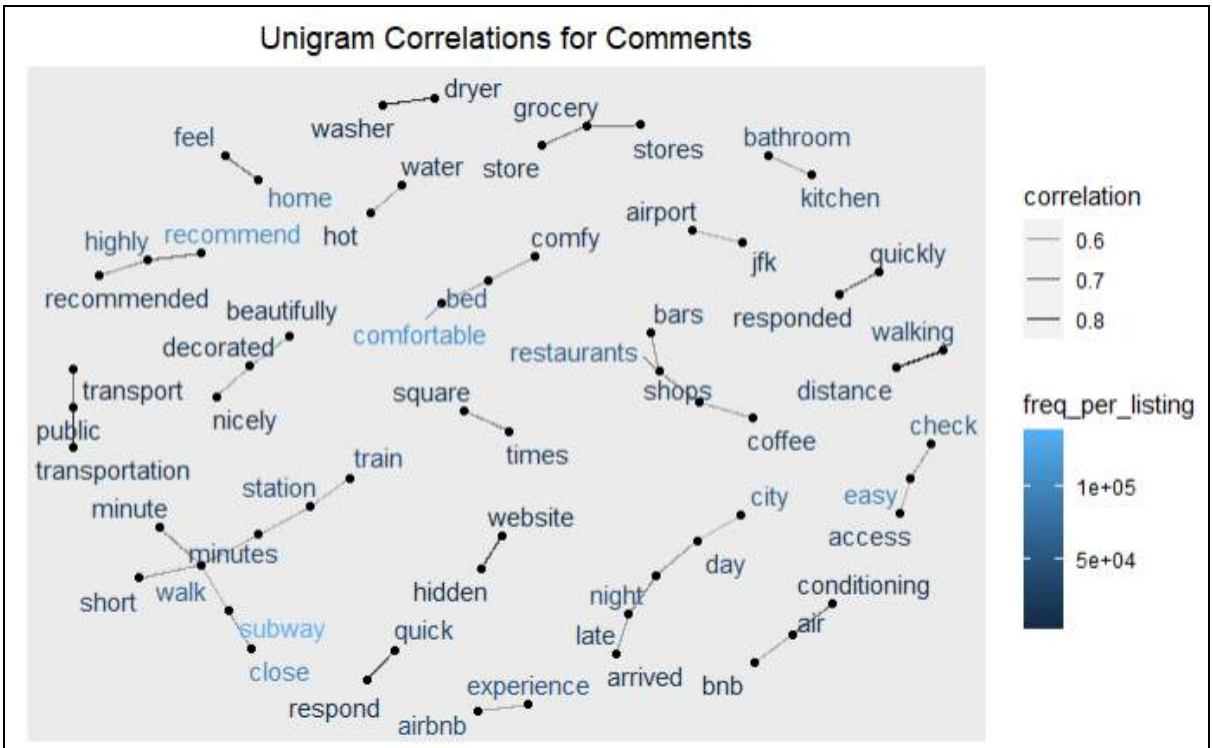



*Figure 15: Unigram correlations for Comments*

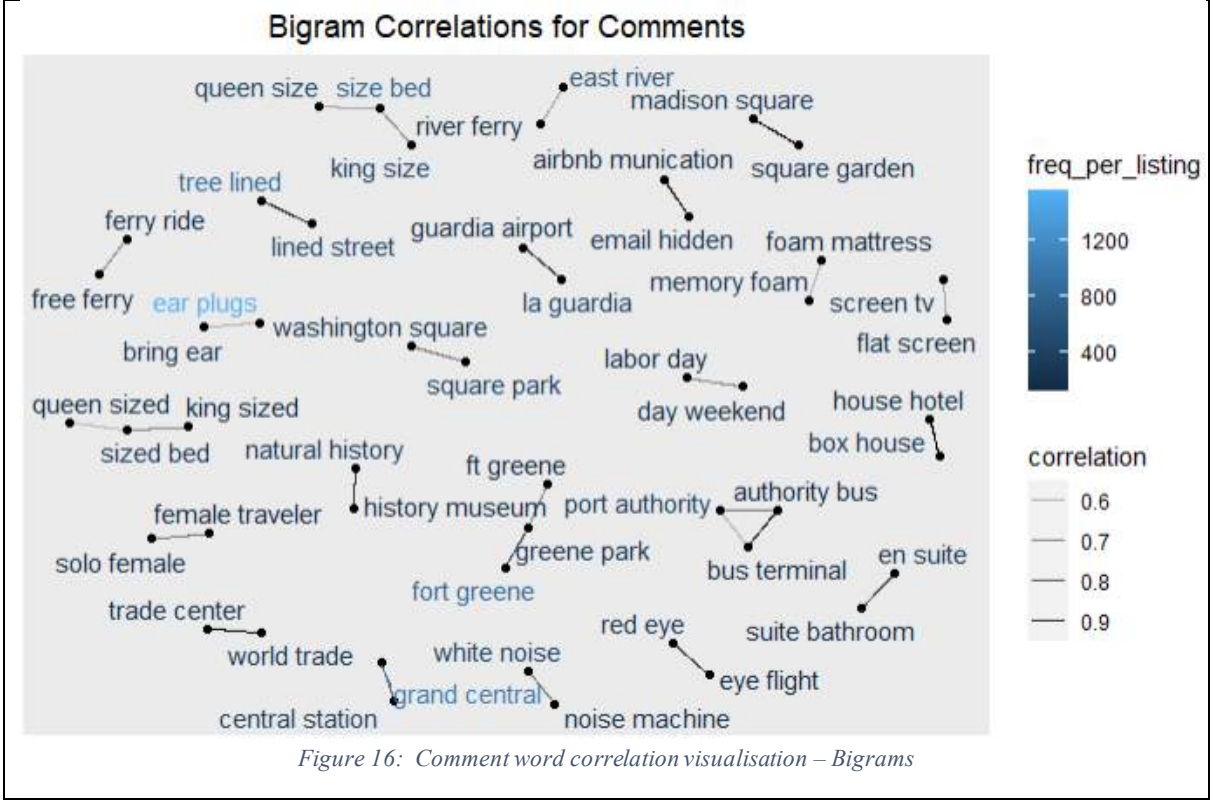



*Figure 16: Comment word correlation visualisation – Bigrams*

## Question c – Variable Extraction

As writing comments is more time-consuming for reviewers than the numeric ratings, more effort is given to the accurate assessment. Therefore, extracting some meaningful variables from the textual comments can support studying customer satisfaction which is shown as score ratings. For question C, except the mentioning name of the owner, we also try many other variables that can affect the score ratings of the listings, as follows:

(1) How many times did reviewers mention the name of the most famous attraction point in New York City in their comments?
(2) the average word count of comments
(3) how many times did reviewers mention words like "recommend"?
(4) how many amenities does each property have?
(5) readability of the comments.

First of all, we checked the distribution of the rating scores of listings, which is extremely skewed. The amounts of listings with rating scores under 80 is 1012, equal to 100 is 7744. We decided to remove the records with the rating under 80 as they are outliers and remove the records with the rating equal to 100 due to the rule of thumb that some of the customers who score the renting experience with full mark are more likely to not think deeper with the ratings at all, so that, need to be removed to conduct a more stable analysis. And also, we removed 8979 rows that have missing value in rating scores. After filtering out the above observations, we got a rating score cleaned listing data and joined the review data with it to prepare for question C. The following plots are distributions of rating scores before cleaning and after cleaning.

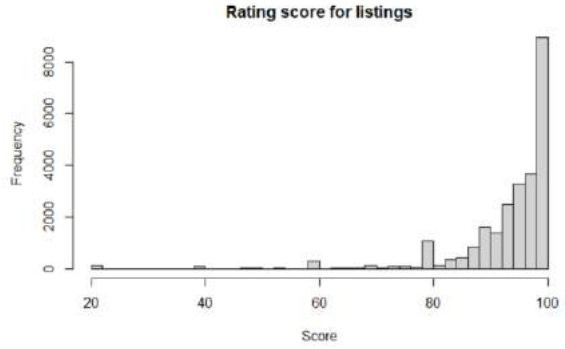



*Figure 17: Distribution of listing rating score*

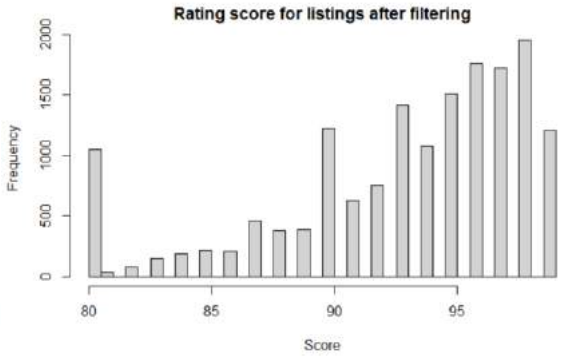



*Figure 18: Distribution of listing score after filtering*

**Mentioning the name of a host**

For detecting the hostname from the comments, the first step is cleaning the hostname column by substituting digits, punctuations and special words such as "new york", "hotel", "apartment". Also, we removed the rows with missing values in the hostname column. After adding one column to display whether the comments contain the name of the host or not, we grouped by the listing_id to calculate the proportion of comments with the hostname.

The following plot shows that those properties with high rating scores have higher possibilities of hostname being mentioned in reviews. However, the linear regression line is almost horizontal, and the correlation coefficient in the table is considerably low, which mean that mentioning hostname or not does matter how high is the corresponding rating score.

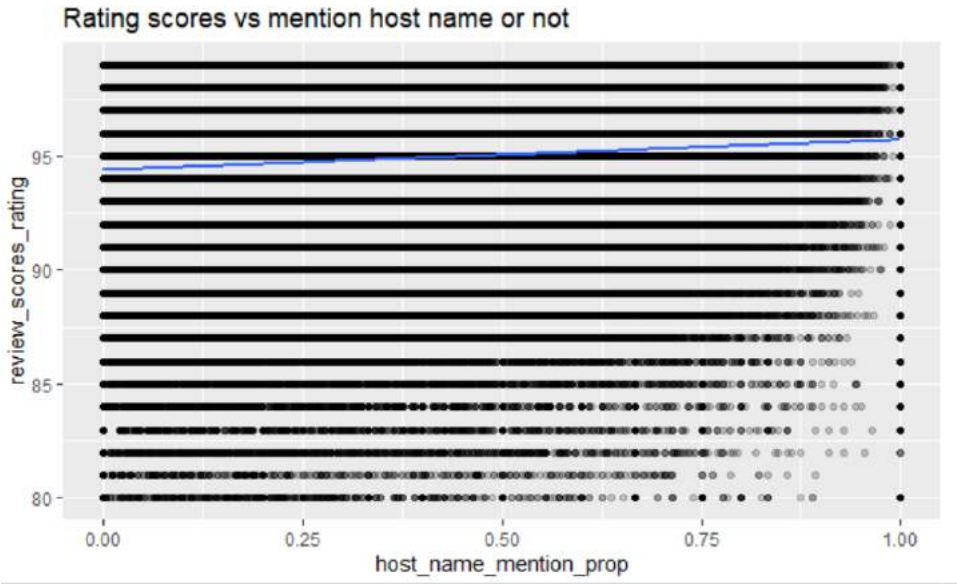



*Figure 19: Rating scores vs host's mention*

```
        Pearson's product-moment correlation

data:  data_Ac$review_scores_rating and data_Ac$host_name_mention_prop
t = 59.974, df = 413229, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.08986931 0.09591462
sample estimates:
       cor
0.09289282
```

*Figure 20: Pearson's product- moment correlation*

For further study, we tried to count the times reviewers mentioned the name of the host in each comment and summed up to a total count of mentioning for every listing. The plot shows a more significant correlation between rating scores and mentioning times than the previous one, which is also confirmed by the correlation coefficient.

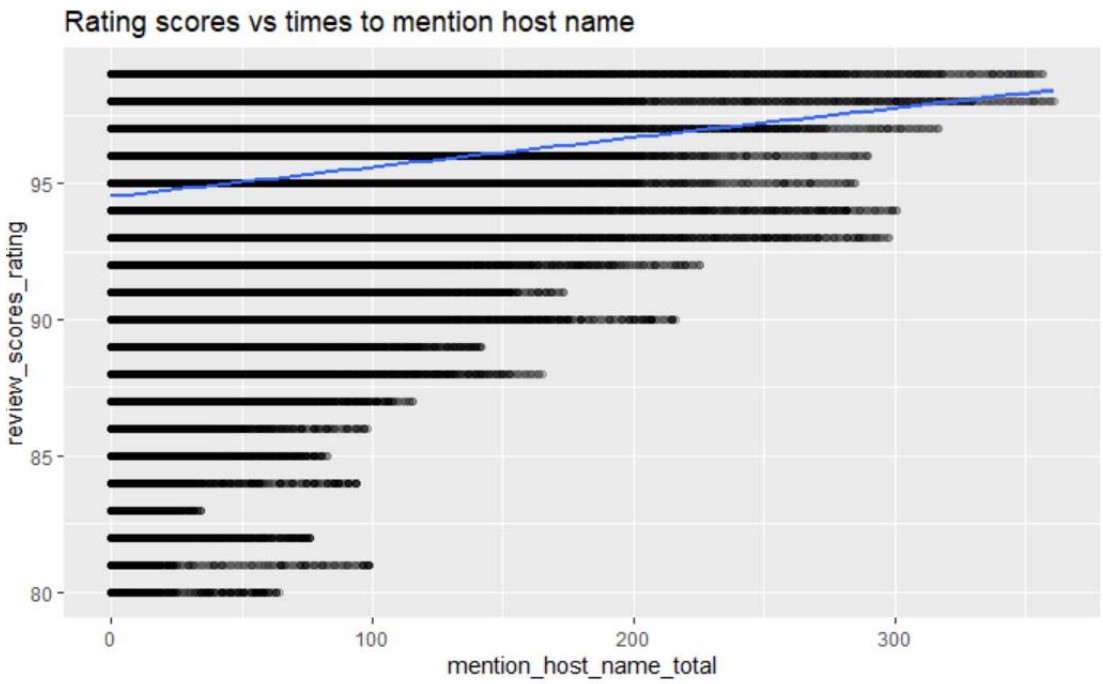



*Figure 21: Rating scores vs times to mention host name*

```
        Pearson's product-moment correlation

data:  data_Ac$review_scores_rating and data_Ac$mention_host_name_total
t = 67.539, df = 413229, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.1014741 0.1075054
sample estimates:
      cor
0.1044907
```

*Figure 22: Pearson's product-moment correlation*

**Length of reviews**

In addition, we calculated the length of the review by applying the string count function in the stringr package, then grouped them by listing id to get the average length of reviews for each property. The following plot shows that for most of the Then plot the scatter diagram to see if the average length of review affects the rating scores.

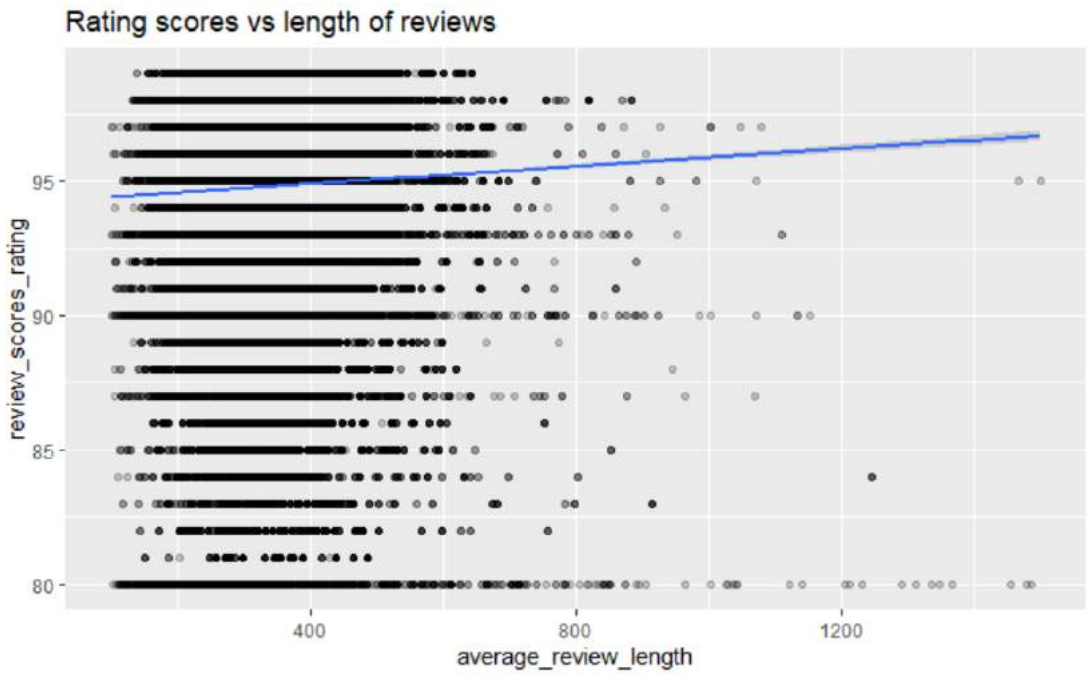



*Figure 23: Rating scores vs length of reviews*

```
        Pearson's product-moment correlation

data:  data_Ac$review_scores_rating and data_Ac$average_review_length
t = 18.814, df = 413229, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.02620877 0.03230148
sample estimates:
      cor
0.0292554
```

*Figure 24: Pearson's product-moment correlation*

**Mentioning "recommend"**

We also searched for if mentioning word such as "recommend" in the reviews is important to rating scores. The plot and correlation table shows a more significant effect of the word "recommend".

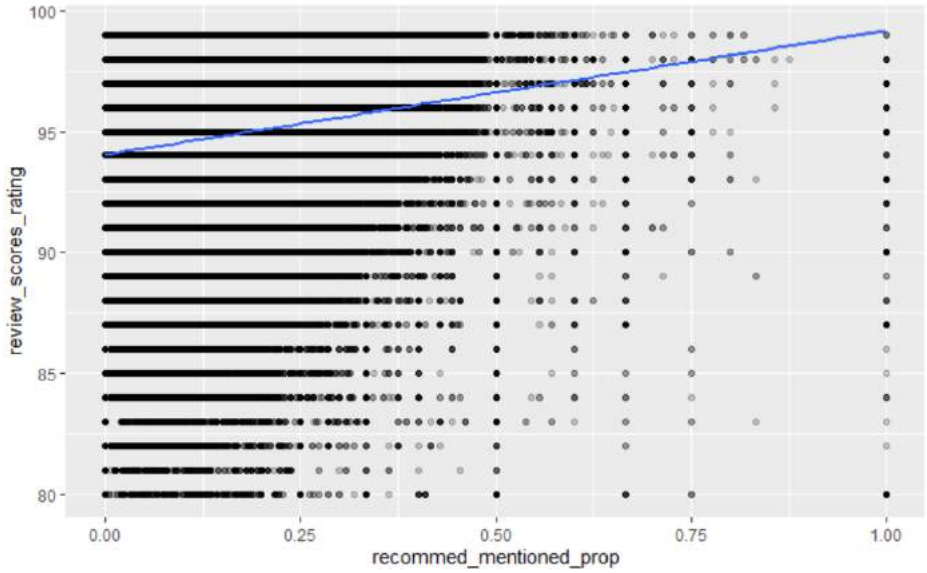



*Figure 25: Review scores rating vs recommend mentioned proportions*

```
        Pearson's product-moment correlation

data:  data_Ac$review_scores_rating and data_Ac$recommed_mentioned_prop
t = 98.752, df = 413229, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.1488604 0.1548177
sample estimates:
      cor
0.1518404
```

*Figure 26: Pearson's correlation statistics*

**Number of amenities**

Availability of equipment in the room also can make a big difference to the user experience and further affect the rating scores. We calculated the number of amenities for each listing and plot the amount of the amenities versus rating scores.

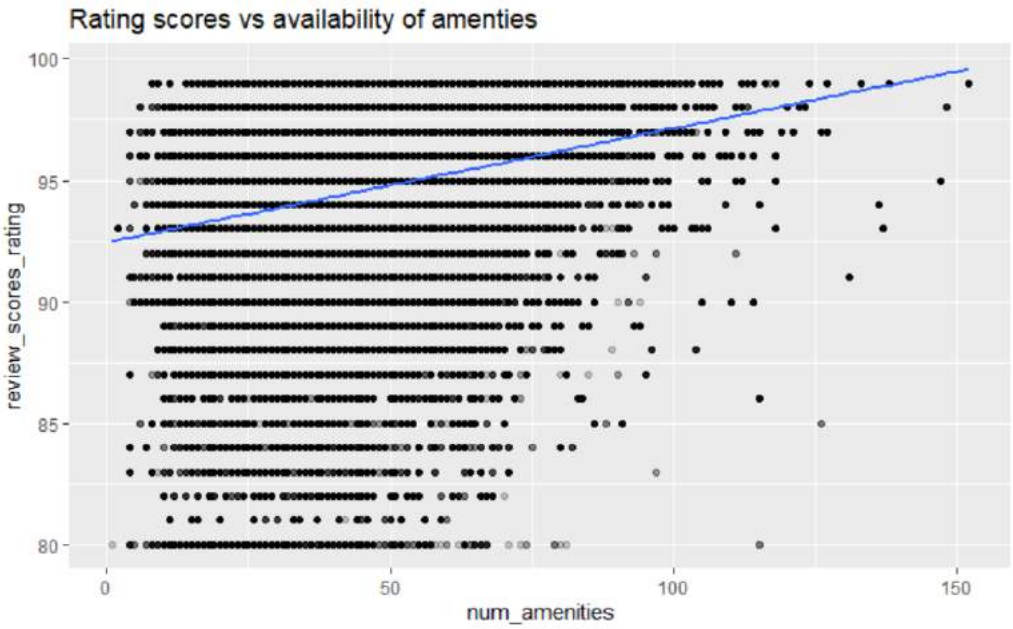



*Figure 27: Rating scores vs availability of amenities*

```
        Pearson's product-moment correlation

data:  data_Ac$review_scores_rating and data_Ac$num_amenities
t = 161.8, df = 427720, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.2373338 0.2429819
sample estimates:
      cor
0.2401599
```

*Figure 28: Pearson's correlation statistics*

**Mentioning tourist attraction spots in NYC**

We crawled the name list of the top 100 tourist attraction spots in New York city from Tripadvisor (https://www.tripadvisor.co.uk/) and tokenised the names into words. After removing the stopwords, we used the %in% function to check if the comments contained the words of attraction points and if this has some impact on the rating scores. We used the tokenised comments for this part to match the words in the attraction points name list. After that, we detected the attraction words showed up in tokenised word column and grouped the data by listing to get an insight of the average level of mentioning the attraction words in comments. From the following chart and correlation table, we can see a weak correlation between the rating scores and mentioning the tourist attraction spots in comments.

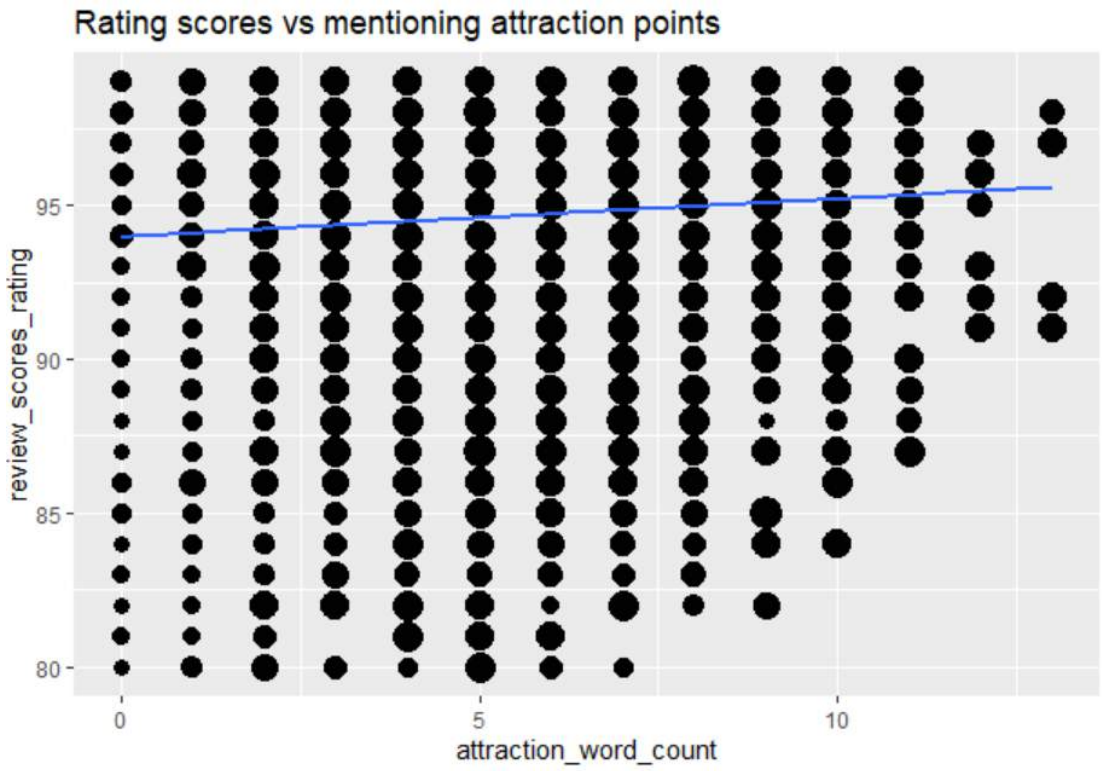



*Figure 29: Rating scores vs mentioning attraction points*

```
        Pearson's product-moment correlation

data:  data_Ac_attraction_detect$review_scores_rating and
data_Ac_attraction_detect$attraction_word_count
t = 134.55, df = 3286478, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.07293928 0.07508972
sample estimates:
       cor
0.07401459
```

*Figure 30: Pearson's correlation statistics*

**Readability of comments**

As comments need to be read by subsequent guests who are interested in the properties, how easy the comments are to be understood is also worthy of studying. To calculate the readability of the comments, we used the flesch_kincaid function from the qdap package, which is based on the Flesch-Kincaid Grade Level formula. Surprisingly, the readability throughout a different level of rating scores remains similar. The correlation coefficient also extremely small, which mean the readability of the reviews does not make a big difference to the rating scores of the listing.

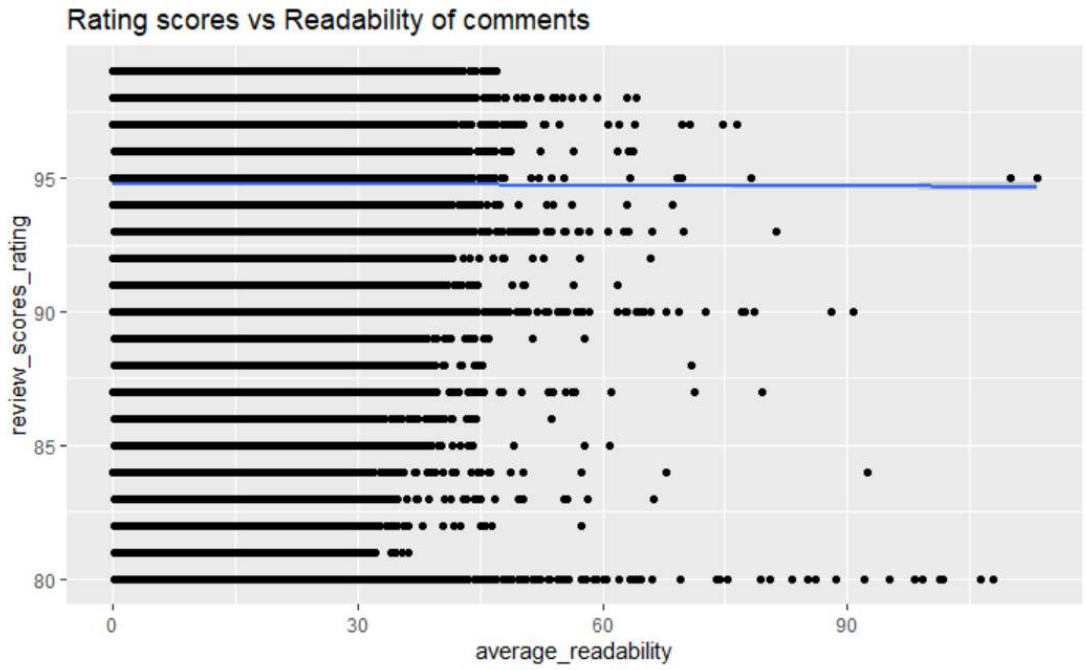



*Figure 31: Rating scores vs readability*

```
        Pearson's product-moment correlation

data:  comment_readability_check$review_scores_rating and comment_readability_check$FK_grd.lvl
t = 5.2356, df = 427720, p-value = 1.646e-07
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.005008416 0.011001766
sample estimates:
       cor
0.008005163
```

*Figure 32: Pearson's correlation statistics*

With all the above analysis, we can see that the features extracted from the textual content do have some effects on rating scores, but most of them are not significant enough to be used to predict the rating scores except mentioning "recommend" in the reviews and the availability of amenities in properties.

### Question d – Text association with Price

First of all, the iconv function was applied to all of the textual descriptions of the listing_id so that the text is ready to be worked with. Moreover, using the cld3 package, the language of each description was detected, and only those which are in English were kept.

```
#Pre-work to get some text to work with from listing description
listings_reviews = listings_reviews %>% mutate(document_id = row_number())
listings_reviews$description_combined <- iconv(listings_reviews$description_combined)
```

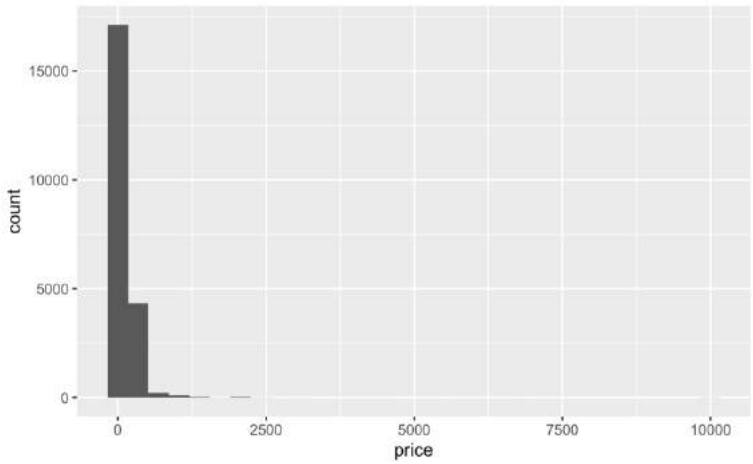



*Figure 33: Price distribution*

```
listings_reviews$language = cld3::detect_language(listings_reviews$description_combined)
listings_reviews = listings_reviews %>% filter(language == "en")
```

Checking price distribution demonstrated that the log of the price should be used in the consequent plots and models, as distribution was not normal.

By using the qdap package readability, formality and polarity were calculated for every description. Their scores were added to the listings_reviews data frame. A column with a word count was also added to this data frame.

```
#Readability calculation
listings_read = qdap::flesch_kincaid(gsub("[^ -~]", "",
 listings_reviews$description_combined),listings_reviews$id)

formality <- qdap::formality(gsub("[^ -~]", "",
 listings_reviews$description_combined),listings_reviews$id)

polarity <- qdap::polarity(gsub("[^ -~]", "",
 listings_reviews$description_combined),listings_reviews$id)

listings_reviews$wordcount = as.numeric(qdap::wc
(listings_reviews$description_combined, byrow = TRUE))
```

The plots of these variables against the log of price demonstrated that readability has a negative relationship with the price even though the slope is fairly low, formality has an evident positive relationship, while polarity and word count are also positively related to the price, but their relationships are slightly weaker than the formality.

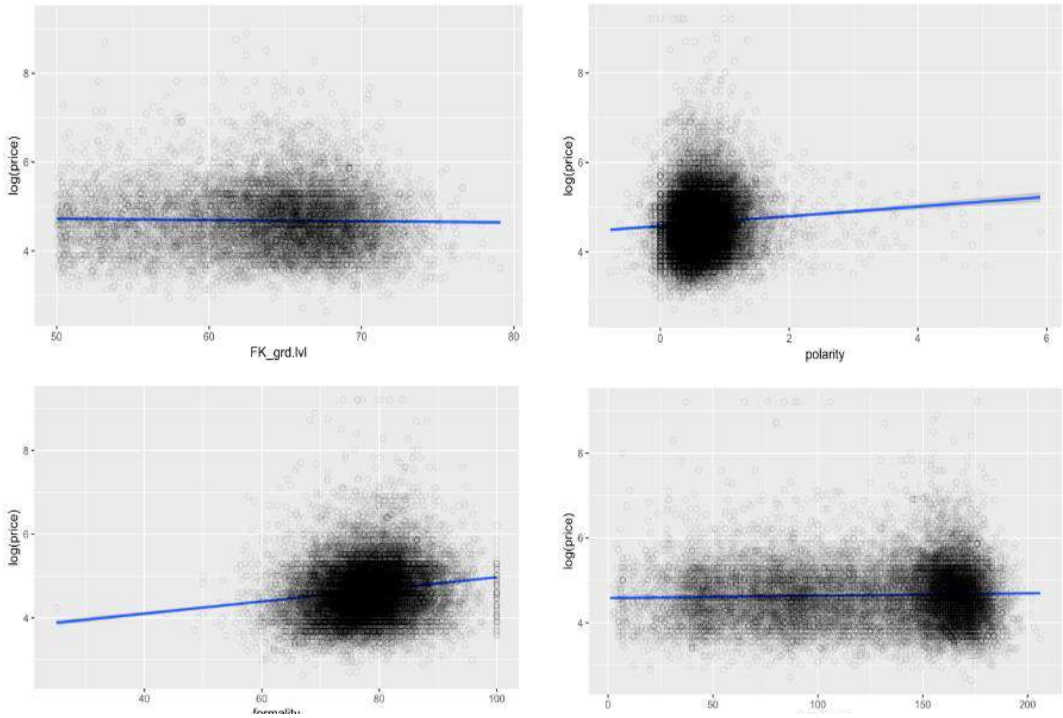



*Figure 34: Polarity and formality graphs vs Log(price)*

Four linear regressions of the log of price against each of the variables outlined above were run in order to measure their effects on the price. The results turned out to show that an increase in readability, formality, polarity and word count by 1 point increases the price on average by $e^{0.002}$= 1.002, $e^{0.014}$= 1.014, $e^{0.108}$= 1.114, $e^{0.001}$= 1.001 dollars correspondingly. All coefficients appeared to be significant. However, the R-squares of all models were rather low, with only 2.2% being the highest for the model

*Table 1: Linear regression of log(price) vs text derived features*

| | Dependent variable: | | | |
|---|---|---|---|---|
| | price) | | | |
| | (1) | (2) | (3) | (4) |
| wordcount | 0.001*** | | | |
| | (0.0001) | | | |
| formality | | 0.014*** | | |
| | | (0.001) | | |
| FK_grd.lvl | | | 0.002*** | |
| | | | (0.0003) | |
| polarity | | | | 0.108*** |
| | | | | (0.011) |
| Constant | 4.582*** | 3.525*** | 4.553*** | 4.580*** |
| | (0.013) | (0.051) | (0.014) | (0.009) |
| Observations | 21,883 | 21,883 | 21,883 | 21,883 |
| R2 | 0.002 | 0.022 | 0.003 | 0.004 |
| Adjusted R2 | 0.001 | 0.022 | 0.003 | 0.004 |
| Residual Std. Error (df = 21881) | 0.676 | 0.669 | 0.675 | 0.675 |
| F Statistic (df = 1; 21881) | 33.238*** | 500.696*** | 60.342*** | 97.507*** |
| Note: | | | | *p<0.1; **p<0.05; ***p<0.01 |

with formality, while for other models, R-squared values were between 0.02 and 0.04%. This implies that, despite effects being fully significant, models lack some explanatory power, meaning that variability in independent variables almost does not explain any variability in price.

Next, the price was split into three categories in order to break down the effects of readability, formality, polarity and word count and assess how these effects actually change depending on whether the price is low, medium or high. Three separate data frames were created, by filtering the price to be below 100$ for low price, between 100$ and 300$ for medium price and above 300$ for the high price. Then, the same models as above were run on three different data sets.

```
#Split the price
price_low = listings_reviews %>% select(id,price, wordcount, formality, FK_grd.lvl, polarity) %>%
filter(price <= 100)

price_med = listings_reviews %>% select(id,price, wordcount, formality, FK_grd.lvl, polarity) %>%
filter(price > 100 & price <= 300)

price_high = listings_reviews %>% select(id,price, wordcount, formality, FK_grd.lvl, polarity) %>%
filter(price > 300)
```

After checking the distribution of each price group, the high price distribution turned to be right-skewed, so for high price models, a log of price was used.

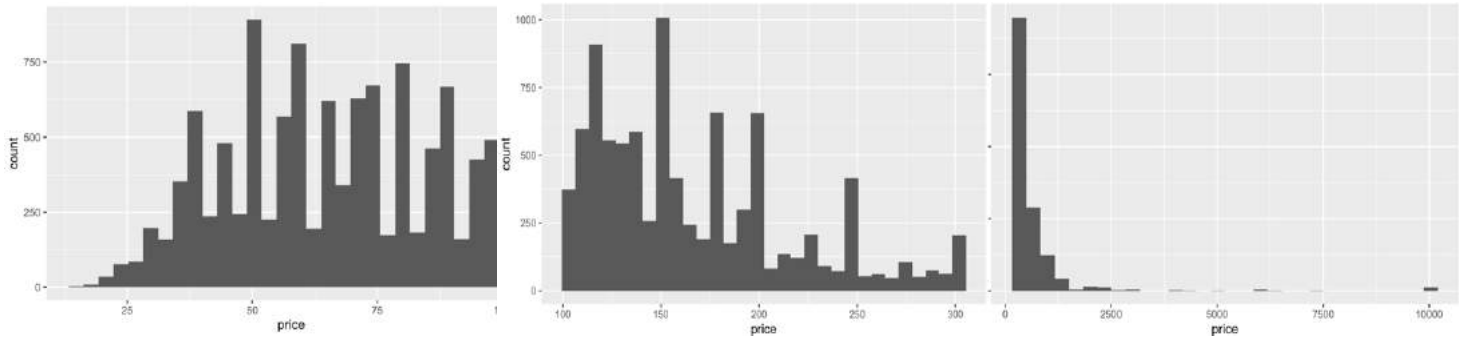



*Figure 35: Distribution of price*

Firstly, models of price against word count were run, and coefficients appeared to be insignificant for low and high prices, while for medium prices, the effect was significant with the value of 0.027, meaning that word count has a considerable effect only on the prices between 100$ and 300$. Nevertheless, R-squared for the medium price model was below 1%, highlighting the same issue as outlined in the models above.

*Table 2: Linear regression of price (1&2) and log(price)(3) vs wordcount*

| | Dependent variable: | | |
|---|---|---|---|
| | price (1) | price (2) | price) (3) |
| wordcount | -0.003 (0.004) | | |
| wordcount | | 0.027** (0.011) | |
| wordcount | | | -0.0004 (0.0004) |
| Constant | 67.239*** (0.537) | 163.502*** (1.523) | 6.309*** (0.051) |
| Observations | 11,476 | 9,242 | 1,165 |
| R2 | 0.00005 | 0.001 | 0.001 |
| Adjusted R2 | -0.00004 | 0.0005 | 0.0003 |
| Residual Std. Error | 20.890 (df = 11474) | 50.297 (df = 9240) | 0.545 (df = 1163) |
| F Statistic | 0.559 (df = 1; 11474) | 5.501** (df = 1; 9240) | 1.344 (df = 1; 1163) |
| Note: | | | *p<0.1; **p<0.05; ***p<0.01 |

Formality models yielded significant coefficients for low and medium prices, with the effects being 0.298 and 0.443, implying that increase in formality score by 1 point on average increases price by 0.298$ for a low price and by 0.443$ for medium price. There was no formality effect present for high prices. In terms of R-squared, the same issue arose with R-squares being low.

*Table 3: Linear regression of price(1&2) and log(price)(3) vs formality*

| | Dependent variable: | | |
|---|---|---|---|
| | price | price | price) |
| | (1) | (2) | (3) |
| formality | 0.298*** | | |
| | (0.027) | | |
| formality | | 0.443*** | |
| | | (0.078) | |
| formality | | | 0.001 |
| | | | (0.002) |
| Constant | 43.957*** | 131.924*** | 6.177*** |
| | (2.093) | (6.191) | (0.195) |
| Observations | 11,476 | 9,242 | 1,165 |
| R2 | 0.010 | 0.003 | 0.0001 |
| Adjusted R2 | 0.010 | 0.003 | -0.001 |
| Residual Std. Error | 20.781 (df = 11474) | 50.225 (df = 9240) | 0.545 (df = 1163) |
| F Statistic | 120.870*** (df = 1; 11474) | 32.064*** (df = 1; 9240) | 0.150 (df = 1; 1163) |
| Note: | | | *p<0.1; **p<0.05; ***p<0.01 |

For readability, the only significant coefficient was in the medium price model, with the effect of 0.078. Therefore, it was concluded that readability only has a real impact on the price within the 100 to 300$ range. R-squared was below 1%.

*Table 4: Linear regression of price(1&2) and log(price)(3) vs FK-grd.lvl*

| | price | price | price) |
|---|---|---|---|
| | (1) | (2) | (3) |
| FK_grd.lvl | 0.002 | | |
| | (0.011) | | |
| FK_grd.lvl | | 0.078*** | |
| | | (0.030) | |
| FK_grd.lvl | | | -0.001 |
| | | | (0.001) |
| Constant | 66.753*** | 162.894*** | 6.317*** |
| | (0.558) | (1.599) | (0.053) |
| Observations | 11,476 | 9,242 | 1,165 |
| R2 | 0.00000 | 0.001 | 0.001 |
| Adjusted R2 | -0.0001 | 0.001 | 0.001 |
| Residual Std. Error | 20.890 (df = 11474) | 50.293 (df = 9240) | 0.545 (df = 1163) |
| F Statistic | 0.046 (df = 1; 11474) | 6.878*** (df = 1; 9240) | 1.604 (df = 1; 1163) |
| Note: | | | *p<0.1; **p<0.05; ***p<0.01 |

Polarity was the only measure, which appeared to be significant across all three different price groups. The biggest impact was for the medium price with the magnitude of 3.792. For higher prices effect was negative, meaning that every increase in polarity score by 1 point leads to an average $e^{0.107}$= 1.112 decreases in price.

*Table 5: Linear regression of price(1&2) and log(price)(3) vs polarity*

| | Dependent variable: | | |
|---|---|---|---|
| | price | price | price) |
| | (1) | (2) | (3) |
| polarity | 2.131*** | | |
| | (0.465) | | |
| polarity | | 3.792*** | |
| | | (1.272) | |
| polarity | | | -0.107*** |
| | | | (0.040) |
| Constant | 65.501*** | 164.252*** | 6.328*** |
| | (0.356) | (1.018) | (0.033) |
| Observations | 11,476 | 9,242 | 1,165 |
| R2 | 0.002 | 0.001 | 0.006 |
| Adjusted R2 | 0.002 | 0.001 | 0.005 |
| Residual Std. Error | 20.871 (df = 11474) | 50.287 (df = 9240) | 0.544 (df = 1163) |
| F Statistic | 20.983*** (df = 1; 11474) | 8.894*** (df = 1; 9240) | 7.034*** (df = 1; 1163) |
| Note: | | | *p<0.1; **p<0.05; ***p<0.01 |

Overall, all four variables appeared to have an impact on medium prices, while for high prices, the only polarity was significant. For low prices, in addition to polarity, impact from formality was also observed. Moreover, when compared to the full dataset models, the values of the effects declined, meaning that in the cases where effects are present for the split datasets, they are of a lower magnitude

Finally, multiple regression models were run on the full dataset and on each of the price that datasets separate. Each model included all four variables that were being analysed. For the first model, all of the coefficients were significant, and R-squared was 4.9%, which is still relatively low but higher than in

*Table 6: Linear regression of log(price) vs all text derived features*

| | Dependent variable: |
|---|---|
| | log(price) |
| wordcount | -0.018*** |
| | (0.001) |
| polarity | 0.050*** |
| | (0.011) |
| FK_grd.lvl | 0.050*** |
| | (0.003) |
| formality | 0.016*** |
| | (0.001) |
| Constant | 3.060*** |
| | (0.061) |
| Observations | 21,883 |
| R2 | 0.049 |
| Adjusted R2 | 0.049 |
| Residual Std. Error | 0.660 (df = 21878) |
| F Statistic | 281.929*** (df = 4; 21878) |
| Note: | *p<0.1; **p<0.05; ***p<0.01 |

the models discussed previously. All of the effects appeared to be positive except for word count, which impacted price negatively.

For models which ran on different datasets, the results showed that for a low price, all of the coefficients were significant and carried positive relationships with price except for word count, which, again, turned out to have a negative effect on price. For medium price wordcount coefficient was not significant, while others still remained significant, however polarity and readability were only 90% significant, as opposed to the low price dataset, where all coefficients were 99% significant. Finally, for a high price, the only polarity was significant, and its effect appeared to be negative. R-squared values were low for all of the models, with a maximum value of 2%.

*Table 7: Linear regression of price(1&2) and log(price)(3) vs variables*

| | Dependent variable: | | |
|---|---|---|---|
| | price | | log(price) |
| | (1) | (2) | (3) |
| wordcount | -0.412*** | -0.170 | 0.002 |
| | (0.049) | (0.130) | (0.004) |
| polarity | 1.444*** | 2.293* | -0.097** |
| | (0.475) | (1.317) | (0.043) |
| FK_grd.lvl | 1.110*** | 0.596* | -0.004 |
| | (0.127) | (0.339) | (0.009) |
| formality | 0.290*** | 0.581*** | 0.001 |
| | (0.030) | (0.087) | (0.003) |
| Constant | 39.684*** | 110.581*** | 6.297*** |
| | (2.560) | (7.514) | (0.226) |
| Observations | 11,476 | 9,242 | 1,165 |
| R2 | 0.020 | 0.007 | 0.007 |
| Adjusted R2 | 0.020 | 0.007 | 0.003 |
| Residual Std. Error | 20.683 (df = 11471) | 50.143 (df = 9237) | 0.544 (df = 1160) |
| F Statistic | 58.510*** (df = 4; 11471) | 16.287*** (df = 4; 9237) | 1.908 (df = 4; 1160) |
| Note: | | | *p<0.1; **p<0.05; ***p<0.01 |

# Part B – Sentiment Association With Prices & Ratings

In this part, we used sentiment dictionaries and syntactical features (e.g. exclamation marks and capital letters) to conduct sentiment analysis on the customers' reviews and established regression models to examine the relationship between sentiment against price and individual ratings. Apart from sentiment, we identified other variables, such as the hotel_acceptance_rate, the host_response_rate, and the num_of_amenities, that contribute to the regression model.

### Affection categorisation

First, a separate data frame listing_id_price_score was created with listing_id, price, and review_scores_rating columns. This data frame was later used for joining with the tables containing sentiment scores so that price and review_score_rating were present at those tables. The sentiment scores were calculated using Bing-Liu, NRC, Afinn and Loughran dictionaries and put into a separate table for each dictionary. Consequently, all these tables were joined together into all_sentiments data frame, which contained listing_id, price and review_scores_rating columns, along with four columns for different sentiment scores. The formula which was used to obtain the value of the sentiment score is (positive-negative)/(positive+negative), however for Afinn dictionary, sum function was used instead, as Afinn dictionary assigns a numerical value to each word, so grouping by listing_id and summing across value column gave the sentiment score.

A regression model was run for each dictionary separately, with price as a dependent variable and a sentiment score as an independent. Moreover, a log of price was used, as price's distribution turned out to be right-skewed.

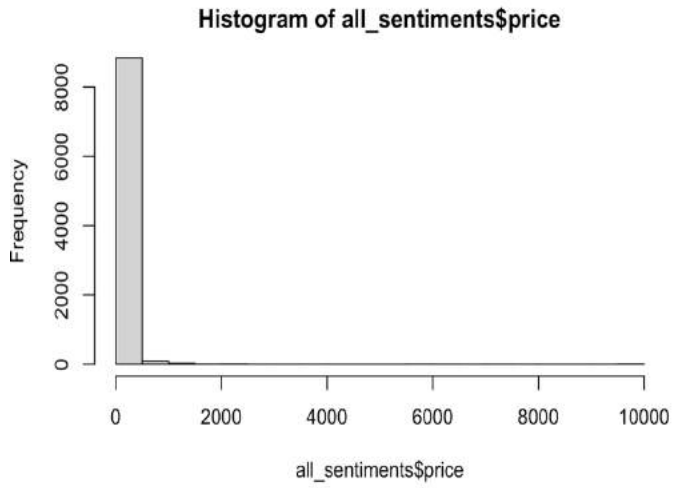



*Figure 36: Histogram of price*

Coefficients for all four regressions were highly significant, and their values were around 0.3 except for the Afinn dictionary, where the coefficient had a value of 0.001. Therefore, for Afinn dictionary, it would mean that increasing a sentiment score by 1 point would increase the price by $e^{0.001}$= 1.001$ on average, while for other models, an increase in sentiment score by 1 would lead to a price increase of roughly $e^{0.3}$= 1.35$ on average. Intercepts in all four models were also significant, with a similar value of around $e^{4.4}$= 81.45. Nevertheless, the R-squared of each model appeared to be extremely low, with the value not being higher than 1.5%. Significant coefficients but low R-squares indicate that the sentiment score is correlated with the price in each model. However, much of the variability in price is not explained. This can also be seen in the visualisation of each regression, where data points are very much spread around the predicted mean. To sum up, there is a positive relationship between price and sentiment score for each dictionary. However, only around 1% variability in price can be explained by the variability in the sentiment score, indicating that even though a relationship exists, it is rather weak.

Considering the low variation explained by the simple regression model, we considered establishing a multiple regression model including other relevant variables to explore the association and to predict price and individual ratings.

*Table 8: Linear regression of log(price) vs sentiment*

| | Dependent variable: | | | |
|---|---|---|---|---|
| | price) | | | |
| | (1) | (2) | (3) | (4) |
| bing_liu_sentiment | 0.385*** | | | |
| | (0.035) | | | |
| nrc_sentiment | | 0.338*** | | |
| | | (0.042) | | |
| afinn_sentiment | | | 0.001*** | |
| | | | (0.0002) | |
| loughran_sentiment | | | | 0.291*** |
| | | | | (0.025) |
| Constant | 4.406*** | 4.439*** | 4.585*** | 4.558*** |
| | (0.024) | (0.027) | (0.013) | (0.011) |
| Observations | 8,970 | 8,970 | 8,970 | 8,970 |
| R2 | 0.013 | 0.007 | 0.004 | 0.015 |
| Adjusted R2 | 0.013 | 0.007 | 0.004 | 0.015 |
| Residual Std. Error (df = 8968) | 0.645 | 0.647 | 0.648 | 0.645 |
| F Statistic (df = 1; 8968) | 118.418*** | 66.049*** | 39.528*** | 134.458*** |
| Note: | | | | *p<0.1; **p<0.05; ***p<0.01 |

Once nine different feelings, including sentiment itself, were extracted from the NRC dictionary, nrc_feelings_modelling data frame was created, which contained a score for each feeling, along with price and review_score_rating columns. This data frame was used to run two multiple regressions with price and review_score_rating as dependent variables and nine feelings as independent variables. This was done in order to check the effect each feeling has on the price and review_score_rating. Log of price and log of review_score_rating were used, as their distributions were not normal.

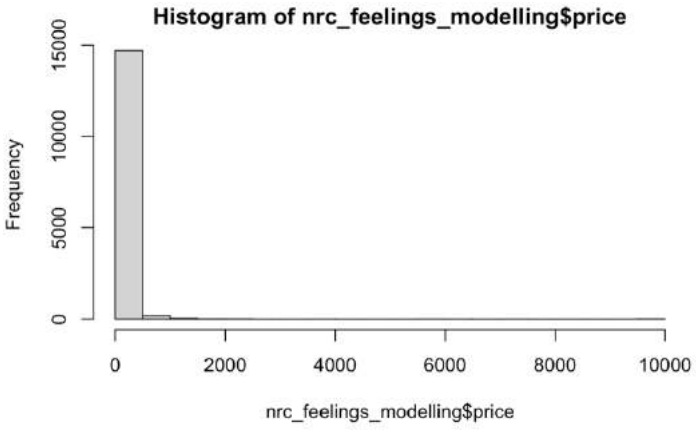



*Figure 38: Histogram of price of NRC sentiments*

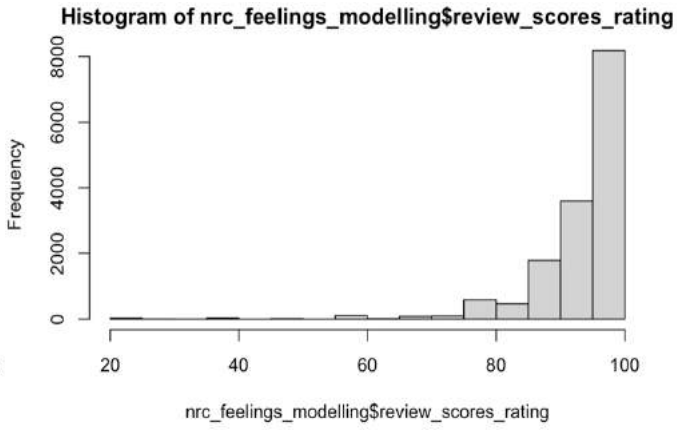



*Figure 37Histogram of review scores of NRC feelings*

For the case with the price, anger, anticipation, disgust, and sadness appeared to be insignificant, with fear, joy, sentiment, surprise and trust being significant. R-squared of the model was only 1.8%, highlighting just like in the previous models that not much of the variation of price can be explained by feelings. For the case with review_score_rating, only sadness was not significant and R-squared of 6.2%, meaning that this model has more explanatory power than the one with a price. Overall, NRC

*Table 9: Linear regression of log(price) vs NRC feelings*

| | Dependent variable: | |
|---|---|---|
| | log(price) | log(review_scores_rating) |
| | (1) | (2) |
| anger | 0.0003 | -0.002*** |
| | (0.005) | (0.001) |
| anticipation | -0.002 | -0.002*** |
| | (0.002) | (0.0004) |
| disgust | 0.006 | -0.009*** |
| | (0.004) | (0.001) |
| fear | -0.021*** | -0.001* |
| | (0.004) | (0.001) |
| joy | 0.022*** | 0.003*** |
| | (0.002) | (0.0003) |
| nrc_sentiment | 0.050*** | 0.016*** |
| | (0.019) | (0.003) |
| sadness | 0.0005 | 0.00002 |
| | (0.003) | (0.001) |
| surprise | 0.011*** | 0.001** |
| | (0.003) | (0.001) |
| trust | -0.017*** | 0.001* |
| | (0.002) | (0.0003) |
| Constant | 4.598*** | 4.518*** |
| | (0.010) | (0.002) |
| Observations | 14,916 | 14,916 |
| R2 | 0.018 | 0.062 |
| Adjusted R2 | 0.018 | 0.061 |
| Residual Std. Error (df = 14906) | 0.645 | 0.101 |
| F Statistic (df = 9; 14906) | 30.821*** | 108.922*** |
| Note: | | *p<0.1; **p<0.05; ***p<0.01 |

feeling analysis showed which feelings, in particular, had a significant effect on the price and review_score_rating and what were the magnitudes of the effects.

Now all the models that were previously run with price as a dependent variable were run with review_score_rating instead. Regressing the log of review_score_rating against each of the sentiment scores yielded significant coefficients in every case, with Afinn once again having the smallest effect of $e^{0.0003}$= 1.0003, while for other dictionaries, the effects were ranging from $e^{0.083}$= 1.086 to $e^{0.165}$= 1.179. R-squared values were different across the models. For example, regression with Bing-Liu sentiment had a 30.2% R-squared, which was the highest, while the lowest was for Afinn with only 11.5%. Still, all of these R-squares turned out to be considerably higher than in the price models. It implies that variability in sentiment scores can explain more variability in review_score_rating rather than in price.

*Table 10:Linear regression of log(review scores) vs different sentiment scores*

| | (1) | (2) | (3) | (4) |
|---|---|---|---|---|
| bing_liu_sentiment | 0.143*** | | | |
| | (0.0004) | | | |
| nrc_sentiment | | 0.165*** | | |
| | | (0.001) | | |
| afinn_sentiment | | | 0.0003*** | |
| | | | (0.00000) | |
| loughran_sentiment | | | | 0.083*** |
| | | | | (0.0003) |
| Constant | 4.468*** | 4.452*** | 4.518*** | 4.528*** |
| | (0.0002) | (0.0003) | (0.0002) | (0.0001) |
| Observations | 298,894 | 298,894 | 298,894 | 298,894 |
| R2 | 0.302 | 0.225 | 0.115 | 0.183 |
| Adjusted R2 | 0.302 | 0.225 | 0.115 | 0.183 |
| Residual Std. Error (df = 298892) | 0.036 | 0.038 | 0.040 | 0.039 |
| F Statistic (df = 1; 298892) | 129,281.600*** | 86,776.580*** | 38,681.470*** | 66,911.150*** |
| Note: | | | | *p<0.1; **p<0.05; ***p<0.01 |

## Syntactical features

For the syntactical features, we focus on the exclamation marks and the capital letters and study the possible connections among the syntactical features, price and review ratings.

We calculated the number of exclamation marks in each listing review and the proportion of exclamation marks containing in each review. The proportion is calculated by the total number of exclamation marks in each listing review divided by the character length of each listing review and is used to adjust the effects from review length.

```
# Calculate the number of exclamation marks in each listing
data_syntactical %>%
  mutate(excl_count = str_count(comments, "!")) %>%
  group_by(listing_id) %>%
  mutate(total_excl_count = sum(excl_count),
         exl_prop = total_excl_count/listings_length_chars) -> data_syntactical
```

A simple regression of the number and the proportion of exclamation marks are conducted against listing prices and ratings. According to the regression results, the total number of exclamation marks are significantly related to the price and the rating score (p-value <0.01). The proportion of exclamation marks are significantly related to the rating but not the price. For models with significant relationships, the total number of exclamations marks has a positive relationship with both price and individual ratings, even though the coefficients are 0.0001 ( e^0.0001 = 1.0001) shows a small effect size.

*Table 11: Linear regression of log(review scores) and log(price) vs exclamation count*

| | Dependent variable: | | | |
|---|---|---|---|---|
| | log(as.numeric(price)) (1) | log(review_scores_rating) (2) | log(as.numeric(price)) (3) | log(review_scores_rating) (4) |
| total_excl_count | 0.0001*** | 0.0001*** | | |
| | (0.00001) | (0.00000) | | |
| exl_prop | | | -0.020 | 0.064*** |
| | | | (0.012) | (0.001) |
| Constant | 4.629*** | 4.544*** | 4.637*** | 4.545*** |
| | (0.002) | (0.0001) | (0.002) | (0.0001) |
| Observations | 316,486 | 316,479 | 316,486 | 316,479 |
| R2 | 0.0001 | 0.019 | 0.00001 | 0.015 |
| Adjusted R2 | 0.0001 | 0.019 | 0.00001 | 0.015 |
| Residual Std. Error | 0.603 (df = 316484) | 0.047 (df = 316477) | 0.603 (df = 316484) | 0.047 (df = 316477) |
| F Statistic | 22.492*** (df = 1; 316484) | 6,203.554*** (df = 1; 316477) | 2.702 (df = 1; 316484) | 4,736.100*** (df = 1; 316477) |
| Note: | | | | *p<0.1; **p<0.05; ***p<0.01 |

Since the model with the number of exclamation marks included showed higher adjusted R^2 values (0.0001, 0.019), we chose to add the total number of exclamation marks rather than the proportion in the model predicting listing prices and review ratings.

Similar to methods applied to exclamation marks, we calculated the number of capital letters and the proportion of capital letters containing in each listing review. Table 12 shows the simple regression results. The model with cap_prop running against price has a higher adjusted R^2 (0.0002) and larger size of coefficients (-0.011) than the model with total_cap_count included. For models against rating, there is a larger coefficient size (0.003) but a lower adjusted R^2 (0.003) when cap_prop is the independent variable. Overall, the proportion of capital letters outperforms the total number of capital letters in the baseline models and therefore is included in the multiple regression model predicting the listing price and review rating.

*Table 12: Linear regression of log(review scores) and log(price) vs capital letters*

| | Dependent variable: | | | |
|---|---|---|---|---|
| | log(as.numeric(price)) (1) | log(review_scores_rating) (2) | log(as.numeric(price)) (3) | log(review_scores_rating) (4) |
| total_cap_count | -0.00000 | 0.00001*** | | |
| | (0.00000) | (0.00000) | | |
| cap_prop | | | -0.011*** | 0.003*** |
| | | | (0.001) | (0.0001) |
| Constant | 4.635*** | 4.547*** | 4.644*** | 4.548*** |
| | (0.002) | (0.0001) | (0.002) | (0.0001) |
| Observations | 316,486 | 316,479 | 316,486 | 316,479 |
| R2 | 0.000 | 0.005 | 0.0002 | 0.003 |
| Adjusted R2 | -0.00000 | 0.005 | 0.0002 | 0.003 |
| Residual Std. Error | 0.603 (df = 316484) | 0.047 (df = 316477) | 0.603 (df = 316484) | 0.047 (df = 316477) |
| F Statistic | 0.002 (df = 1; 316484) | 1,681.768*** (df = 1; 316477) | 66.713*** (df = 1; 316484) | 794.352*** (df = 1; 316477) |
| Note: | | | | *p<0.1; **p<0.05; ***p<0.01 |

From the regression results, syntactical features show significant relationships with price and individual rating. To understand how the syntactical features are related to price and rating, we run regression models between syntactical features and the sentiments to check if frequent appearances of exclamation marks or capital letter indicate stronger sentiment. The following are the liner model plots for the number of exclamation marks and the proportion of capital letters against sentiments calculated by 4 dictionaries.

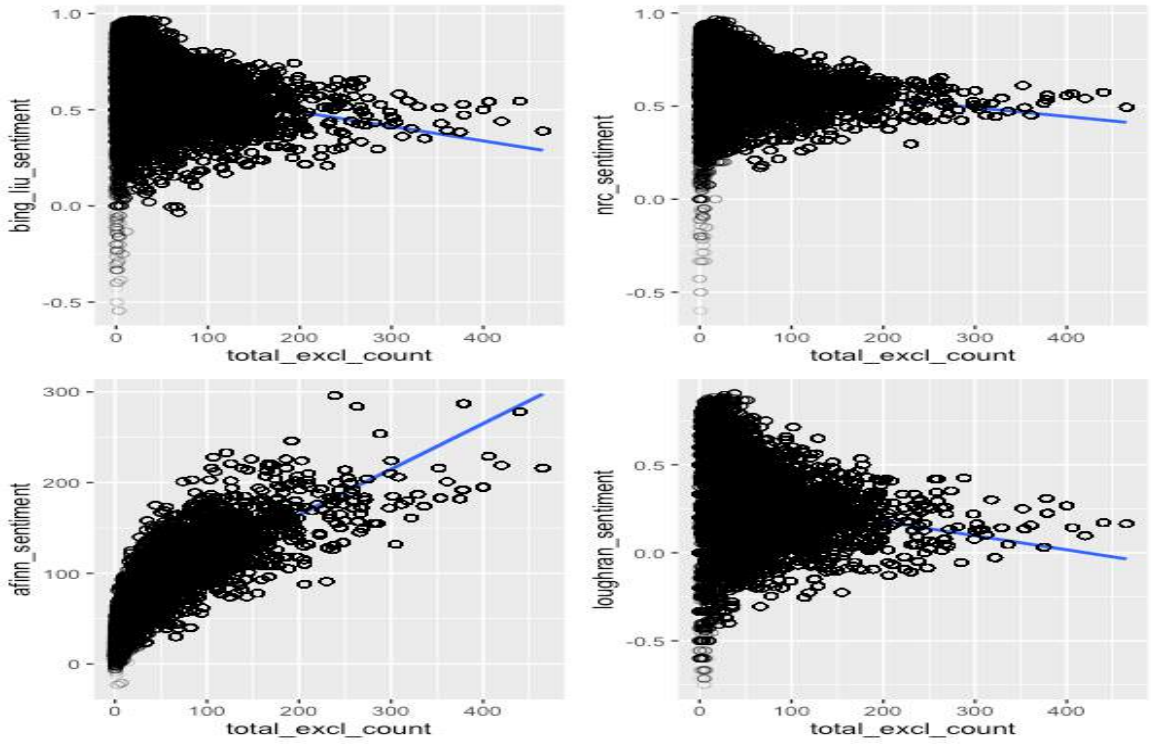



*Figure 39 Scatter plot of exclamation points vs sentiments*

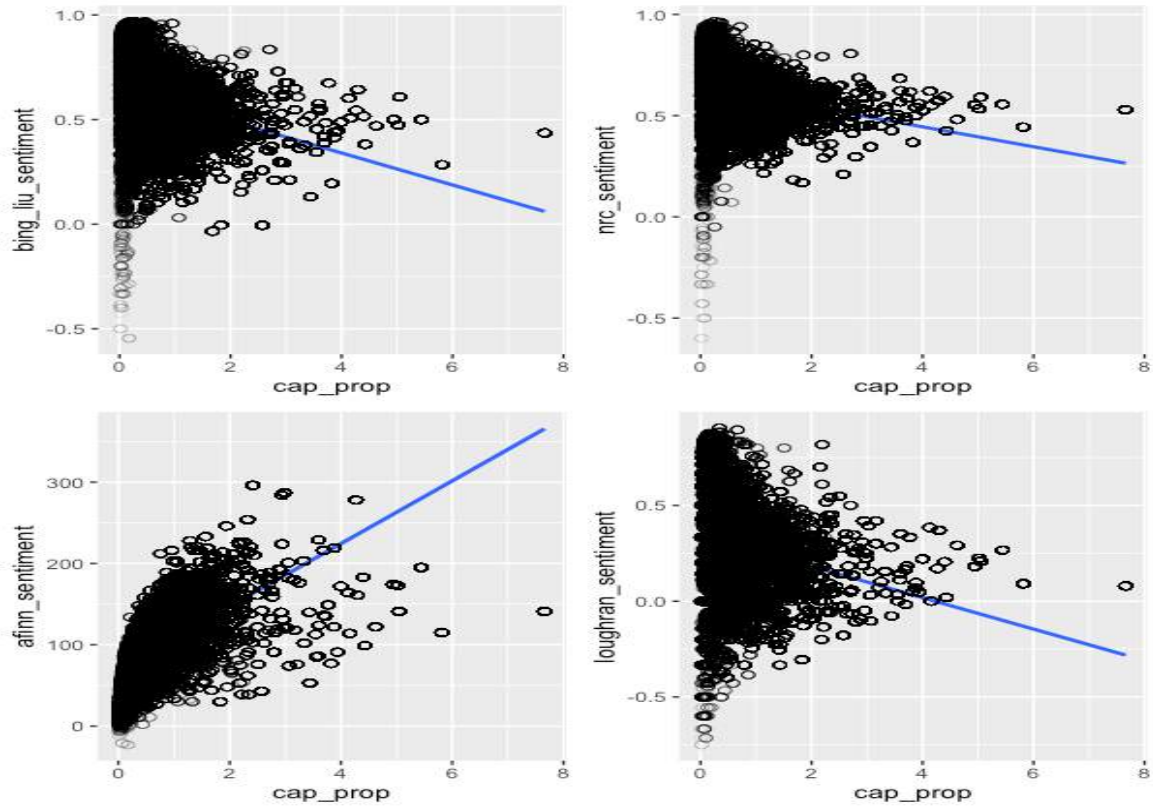



*Figure 40: Scatter plot of capital words counts vs sentiments*

From Figure 40 & 41, we can find that the syntactical features show a clear positive relationship with sentiments when the Afinn dictionary is applied and slightly negative relationships with the sentiment calculated by the other three dictionaries. For both the number of exclamation marks and the proportion of capital letters, more appearances indicate stronger sentiments in a positive way when sentiments are

captured by Afinn dictionaries. Table 13 and 14 shows the regression results for the model with total_excl_count and cap_prop against price and ratings. For both the two models using Afinn sentiments, the syntactical features have positive relationships with sentiments and models show relatively high adjusted R^2 values (0.646, 0.446) than the other three models, which indicates the syntactical features could explain more variation in the sentiments captured by Afinn dictionary than the other dictionaries. Therefore, syntactical features could be applied simultaneously with the Afinn dictionaries to better capture the sentiments from the text.

*Table 13: Linear regression of different sentiment vs exclamamtion count*

| | Dependent variable: | | | |
|---|---|---|---|---|
| | bing_liu_sentiment | nrc_sentiment | afinn_sentiment | loughran_sentiment |
| | (1) | (2) | (3) | (4) |
| total_excl_count | -0.001*** | -0.0005*** | 0.503*** | -0.001*** |
| | (0.00000) | (0.00000) | (0.001) | (0.00001) |
| Constant | 0.638*** | 0.636*** | 63.812*** | 0.337*** |
| | (0.0004) | (0.0003) | (0.074) | (0.001) |
| Observations | 298,894 | 298,894 | 298,894 | 298,894 |
| R2 | 0.113 | 0.083 | 0.646 | 0.072 |
| Adjusted R2 | 0.113 | 0.083 | 0.646 | 0.072 |
| Residual Std. Error (df = 298892) | 0.155 | 0.118 | 27.413 | 0.212 |
| F Statistic (df = 1; 298892) | 38,137.150*** | 27,098.760*** | 545,662.200*** | 23,101.090*** |
| Note: | | | | *p<0.1; **p<0.05; ***p<0.01 |

*Table 14: Linear regression of different sentiment and afinn sentiment vs capital letters*

| | Dependent variable: | | | |
|---|---|---|---|---|
| | bing_liu_sentiment | nrc_sentiment | afinn_sentiment | loughran_sentiment |
| | (1) | (2) | (3) | (4) |
| cap_prop | -0.076*** | -0.049*** | 38.569*** | -0.082*** |
| | (0.0003) | (0.0003) | (0.078) | (0.0005) |
| Constant | 0.645*** | 0.641*** | 69.625*** | 0.345*** |
| | (0.0004) | (0.0003) | (0.093) | (0.001) |
| Observations | 306,681 | 306,681 | 306,681 | 306,681 |
| R2 | 0.137 | 0.102 | 0.446 | 0.088 |
| Adjusted R2 | 0.137 | 0.102 | 0.446 | 0.088 |
| Residual Std. Error (df = 306679) | 0.153 | 0.117 | 34.297 | 0.211 |
| F Statistic (df = 1; 306679) | 48,660.140*** | 34,711.050*** | 247,032.300*** | 29,456.140*** |
| Note: | | | | *p<0.1; **p<0.05; ***p<0.01 |

## Other variables against price & rating

To establish a multiple regression model, we identified other possible variables from both structured and unstructured data to check their relationships against prices and individual ratings.

Host_response_rate, host_acceptance_rate and num_of_amenities variables were taken in order to check their effect on the price and review_score_rating. These numerical variables represented the structured part of the dataset and were used to increase the explanatory power of the models. Model_data data frame was created, which contained all of these variables, along with listing_id, price and review_scores_rating. Three simple regressions were run for price against each of these numerical variables. This was done to get the rough idea of whether the relationships exist and, if yes, what are their magnitudes. Same regressions were run with the review_scores_rating as a dependent variable instead of the price. Models and their visualisations can be found in the Appendix.

What is more important is including host_response_rate, host_acceptance_rate and num_of_amenities into multiple regression, along with sentiment scores obtained from different dictionaries. Moreover, the total number of exclamation marks (total_excl_count) and proportion of capital letters (cap_prop) variables were also added to the model. For this purpose, a mult_reg_data data frame was created, which contained all of the necessary variables.

*Table 15: Table 13: Linear regression of log(price) and different sentiments*

| | Dependent variable: | | | |
|---|---|---|---|---|
| | log(price) | | | |
| | (1) | (2) | (3) | (4) |
| bing_liu_sentiment | 0.640*** | | | |
| | (0.010) | | | |
| nrc_sentiment | | 0.532*** | | |
| | | (0.013) | | |
| afinn_sentiment | | | 0.005*** | |
| | | | (0.0001) | |
| loughran_sentiment | | | | 0.485*** |
| | | | | (0.007) |
| host_acceptance_rate | -0.002*** | -0.002*** | -0.001*** | -0.002*** |
| | (0.0001) | (0.0001) | (0.0001) | (0.0001) |
| host_response_rate | 0.001*** | 0.001*** | 0.001*** | 0.001*** |
| | (0.0001) | (0.0001) | (0.0001) | (0.0001) |
| num_amenities | 0.005*** | 0.005*** | 0.005*** | 0.005*** |
| | (0.0001) | (0.0001) | (0.0001) | (0.0001) |
| total_excl_count | 0.0003*** | 0.0003*** | -0.002*** | 0.0003*** |
| | (0.00003) | (0.00003) | (0.00004) | (0.00003) |
| cap_prop | 0.022*** | 0.004 | -0.036*** | 0.016*** |
| | (0.003) | (0.003) | (0.003) | (0.003) |
| Constant | 4.002*** | 4.045*** | 4.097*** | 4.238*** |
| | (0.010) | (0.011) | (0.009) | (0.008) |
| Observations | 178,832 | 178,832 | 178,832 | 178,832 |
| R2 | 0.054 | 0.039 | 0.071 | 0.056 |
| Adjusted R2 | 0.054 | 0.039 | 0.071 | 0.056 |
| Residual Std. Error (df = 178825) | 0.601 | 0.605 | 0.595 | 0.600 |
| F Statistic (df = 6; 178825) | 1,704.775*** | 1,224.204*** | 2,275.098*** | 1,761.945*** |
| Note: | | | | *p<0.1; **p<0.05; ***p<0.01 |

As a result, four models of log of price against host_response_rate, host_acceptance_rate, num_of_amenities and the sentiment score were run, with one model for each dictionary. R-squares of the models ranged from 3.9% to 7.1%. The explanatory power of these models at least doubled when compared to baseline regressions, with dictionary sentiment score being the only variable. All of the coefficients in every model were significant except for the host_response_rate and cap_prop. The former appeared to be insignificant in all four models, meaning that this variable had no effect on the price, while the latter was insignificant only for the NRC dictionary. Host_acceptance_rate turned out to have a negative relationship with the price across all of the models. However, the effect was very low, only around $e^{0.002}$= 1.002 in each regression. Num_amenities had a positive effect with the value of around 0.005 for each model, meaning that increasing the number of amenities by 1, increases the price by $e^{0.005}$= 1.005$ on average. Yet again, the effect is rather low. The values of the sentiment score coefficients increased slightly in every case comparing to the baseline models, with Afinn sentiment having a 0.002 value, while for other dictionaries, the values ranged from 0.35 to 0.412, with the true effect being $e^{0.002}$= 1.002, $e^{0.35}$ =1.42 and $e^{0.412}$= 1.51. As for total_excl_count, the effect was persistent at $e^{0.0003}$ = 1.0003 level for every model, but for Afinn dictionary, the relationship with

the price was negative, and the effect's magnitude was $e^{0.002} = 1.0002$. Cap_prop had a positive impact on the price for Bing-Liu and Loughran dictionaries, while for Afinn, the effect was negative.

Other than variables included in the multiple regression model against price, we added variables related to sub-category rating derived from customer reviews. For the individual rating, we notice there are variables (i.e. review_score_accuracy, review_scores_cleanliness, review_scores_checkin, review_scores_communication, review_scores_location, review_scores_value) about sub-categories review ratings. A correlation matrix among the dominant rating and sub-rating is checked to find what customers focus most when rating.

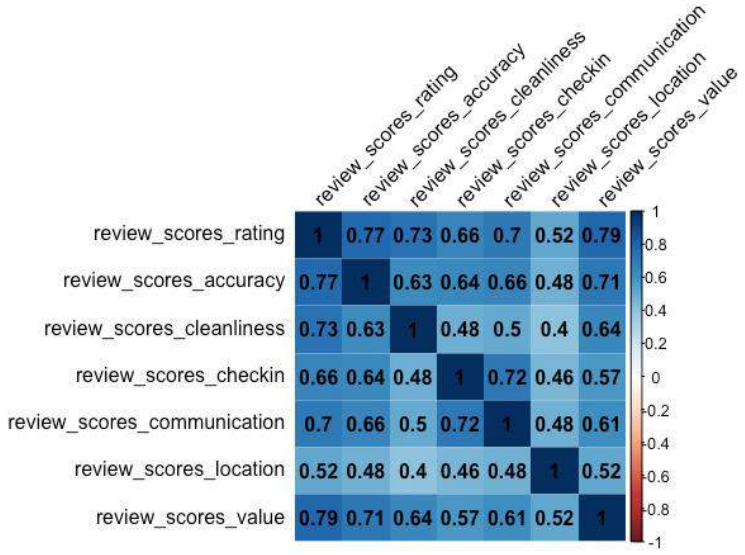



*Figure 41: Correlation matrix*

From the correlation matrix, we find the main review_scores_rating has the highest two correlations with review_scores_value (0.79) and review_scores_accuracy (0.77). Therefore, we infer that customers might give a high rating when they consider the listing valuable and with accurate descriptions. Following this inference, we calculate the frequency of word "value" (total_value_count) and "accurate" (total_accuracy_count) appearing in the review as the proxy of customers opinions towards the listing's value and accuracy and add the two variables into the multiple regression model against individual rating.

Simple regression was conducted first to check the linear relationship between total_value_count, total_accuracy_count and the rating scores.

*Table 16: Linear regression of log(review score rating) vs total value and accuracy*

| | Dependent variable: | |
|---|---|---|
| | log(review_scores_rating) | |
| | (1) | (2) |
| total_value_count | -0.002*** | |
| | (0.00003) | |
| total_accuracy_count | | -0.001*** |
| | | (0.0001) |
| Constant | 4.556*** | 4.552*** |
| | (0.0001) | (0.0001) |
| Observations | 316,479 | 316,479 |
| R2 | 0.019 | 0.001 |
| Adjusted R2 | 0.019 | 0.001 |
| Residual Std. Error (df = 316477) | 0.047 | 0.047 |
| F Statistic (df = 1; 316477) | 6,032.938*** | 243.861*** |
| Note: | *p<0.1; **p<0.05; ***p<0.01 | |

Table 16 shows that total_value_count, total_accuracy_count have a significant relationship with review ratings even though the two variable could not explain much variation in the rating score (based on low R^2 value). We then added the two new identified variables to the multiple regression model against rating to check the model performance.

*Table 17: Linear regression of log(review scores rating) vs features such as sentiment, host characteristics*

| | log(review_scores_rating) | | | |
|---|---|---|---|---|
| | (1) | (2) | (3) | (4) |
| bing_liu_sentiment | 0.180*** | | | |
| | (0.00004) | | | |
| nrc_sentiment | | 0.215*** | | |
| | | (0.0001) | | |
| afinn_sentiment | | | 0.0005*** | |
| | | | (0.00000) | |
| loughran_sentiment | | | | 0.106*** |
| | | | | (0.00004) |
| host_acceptance_rate | 0.00001*** | 0.00001*** | 0.00002*** | 0.00000*** |
| | (0.00000) | (0.00000) | (0.00000) | (0.00000) |
| host_response_rate | 0.0001*** | 0.0001*** | 0.0002*** | 0.0001*** |
| | (0.00000) | (0.00000) | (0.00000) | (0.00000) |
| num_amenities | 0.0002*** | 0.0002*** | 0.0002*** | 0.0003*** |
| | (0.00000) | (0.00000) | (0.00000) | (0.00000) |
| total_excl_count | 0.0002*** | 0.0002*** | 0.00003*** | 0.0002*** |
| | (0.00000) | (0.00000) | (0.00000) | (0.00000) |
| cap_prop | -0.0004*** | -0.001*** | -0.008*** | -0.002*** |
| | (0.00001) | (0.00001) | (0.00001) | (0.00001) |
| total_value_count | -0.001*** | -0.002*** | -0.002*** | -0.002*** |
| | (0.00000) | (0.00000) | (0.00000) | (0.00000) |
| total_accuracy_count | -0.0002*** | -0.001*** | -0.002*** | -0.001*** |
| | (0.00000) | (0.00000) | (0.00000) | (0.00000) |
| Constant | 4.425*** | 4.398*** | 4.481*** | 4.497*** |
| | (0.00004) | (0.0001) | (0.0001) | (0.00004) |
| Observations | 17,169,038 | 17,169,038 | 17,169,038 | 17,169,038 |
| R2 | 0.610 | 0.518 | 0.361 | 0.473 |
| Adjusted R2 | 0.610 | 0.518 | 0.361 | 0.473 |
| Residual Std. Error (df = 17169029) | 0.023 | 0.025 | 0.029 | 0.026 |
| F Statistic (df = 8; 17169029) | 3,358,945.000*** | 2,302,678.000*** | 1,214,715.000*** | 1,924,088.000*** |
| Note: | | | | *p<0.1; **p<0.05; ***p<0.01 |

The regression results indicate the model with bing_liu_sentiment performs best with an adjusted R^2 of 0.610, which shows an improvement of 0.302 in adjusted R^2 compared to the baseline regression model with only sentiments scores are included. All the variables included are significantly related to the rating score. Regarding the bing_liu_sentiment model, sentiments have the largest coefficient (0.18), showing the dominant effect (e^0.18 = 1.19721736312) on rating compared with other variables. The cap_prop, total_value_count and total_accuracy_count show negative relationships with rating with a relatively small effect size. Based on the results, we could guess the capital letters and word "value" and "accurate" appears in the reviews in a neutral but slight negative way. While the total number of exclamation marks show a slightly positive relation with rating, the appearance of exclamation marks might imply the customers' positive attitudes towards listings.

# Part C – Topic Modelling

## Data preparation

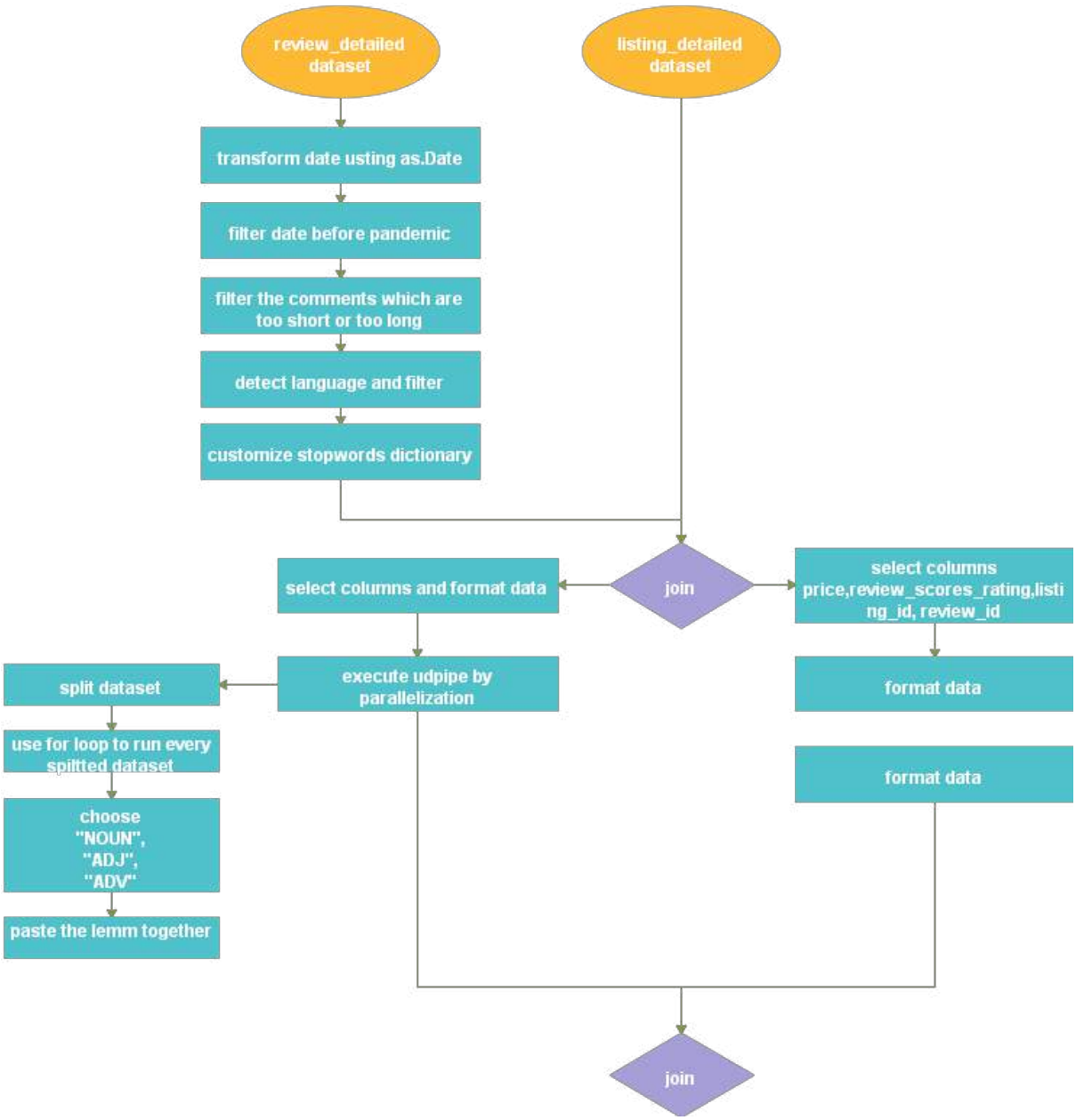



*Figure 42: Pipeline overview of data preparation*

In data preparation for part C, we chose POS tagging classifier with udpipe_annotate( ) function rather than the unnest_tokens( ) function. As POS taggers predict the part of speech depending on the language structure, so we only left certain steps used in the data preparation process in part A, such as filtering the review data by the date before the epidemic, filtering by the language, removing the observations with too short or too long. After joining with the listing_detailed data, we selected review_id and comments columns to prepare for the POS tagging. As the dataset is relatively huge, we chose to split the data into the subsets with 1000 observation in each, then using for loop to execute the udpipe one by one. In the pipeline for every iteration, we included the filtering of the terms tagged as "NOUN", "ADJ", "ADV", which are more likely to be the meaningful words that can perform as a more interpretable topic. At the end of each iteration, we used the bind_rows( ) function to add the outputs to the output we got from the previous iteration. As the POS tagging can also label the misspelt words, it is easier to detect and remove typo from the textual content than the tokenisation method.

After the documents are lemmatised and stemmed, it is ready for topic modelling analysis. There are different levels of aggregation that can be chosen to conduct this analysis. The levels chosen were

a. Per each review
b. Per each listing
c. Per each neighbourhood

These levels were chosen because the review level is the most granular and hence will yield good insights about topics discussed in each review. The listing aggregation was done to analyse the topics discussed in each listing, and this can give insights into what are the topics discussed for listings, including the property itself, closeness to public locations, access to transit etc. The last aggregation was chosen as the neighbourhood as this could give insights into common topics in particular neighbourhoods. Some neighbourhoods may be better to live in and are more popular than the rest.

### Data pre-processing

The data frame to be analysed needs to have the price and review rating columns so that the regressions can be run with the sentiment and topic scores. Hence, we left join the output of udpipe with the data frame that has the required columns pre-selected.

### Aggregation by Reviews

Since the pre-processing is done at the level of the reviews, there is no further grouping required. We cleaned up the price column and omitted the NA values. Post this, we used the Textprocessor function to remove stopwords and perform basic text processing. The stopwords are decided iteratively through steps 1 and 2 and augmented into the stopwords list. This output is stored and used in the prepDocuments function to prepare the data for topic modelling. We choose to remove words that appear in less than 1% of the documents and set this as the threshold. Since the price and review scores are per listing, we do not have to create an average value for these attributes.

**Step 1: Choosing Kappa: Number of topics**

We use the stm function to calculate the number of topics by setting K = 0 and setting the prevalence measure to be price and review rating. We found out a total of 39 topics. These topics had high overlapping, and after observing the cross-loadings, an estimate of 5 to 10 topics were decided as per human judgement. This range is further analysed using the SearchK function to identify the optimum number of latent topics in the corpus (reviews). The parameters chosen to finalise the number of topics are Held-Out Likelihood, Residuals and Semantic Coherence. The following image clearly shows that 8 topics were the ideal choice for this aggregation.

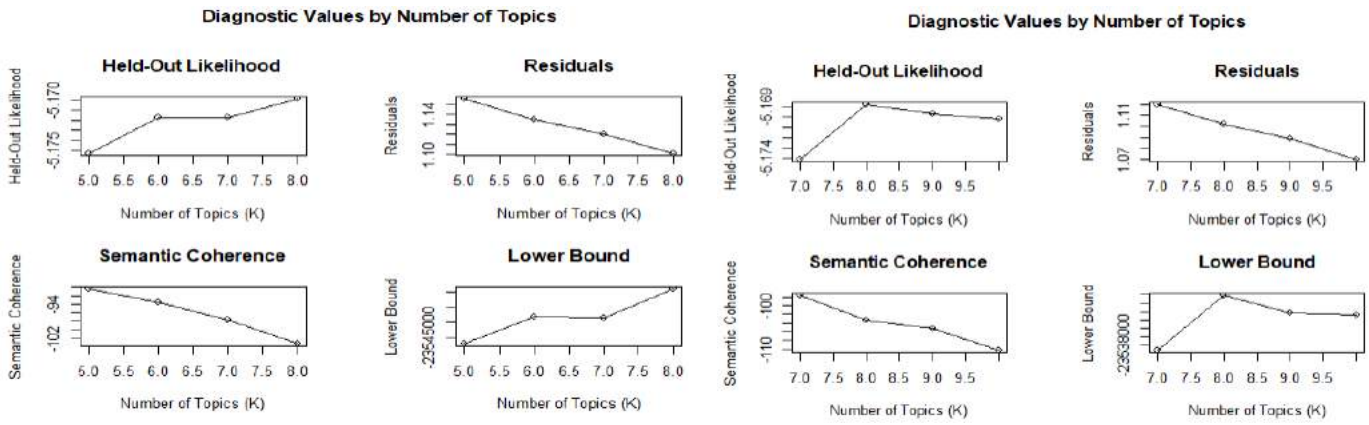



*Figure 43: DIagnostinc values - Finding Kappa*

**Step 2: Building beta matrix and Topic Labelling**

Hence we fit the clean data to the stm package with K=8 and prevalence = ~review_scores_rating + price. We can see a glimpse of the 8 topics from the corpus in the following image:

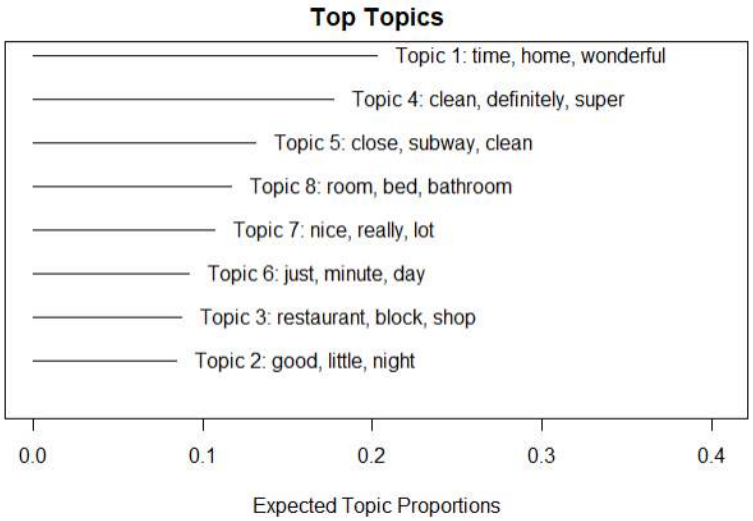



*Figure 44: Description of topics*

Note: Probably, Topic 1 can be observed as generic.
The 8 topics were studied, and according to human interpretation, topic labels were assigned to each of the topics. The following image shows the overall term frequency in each of the topics.

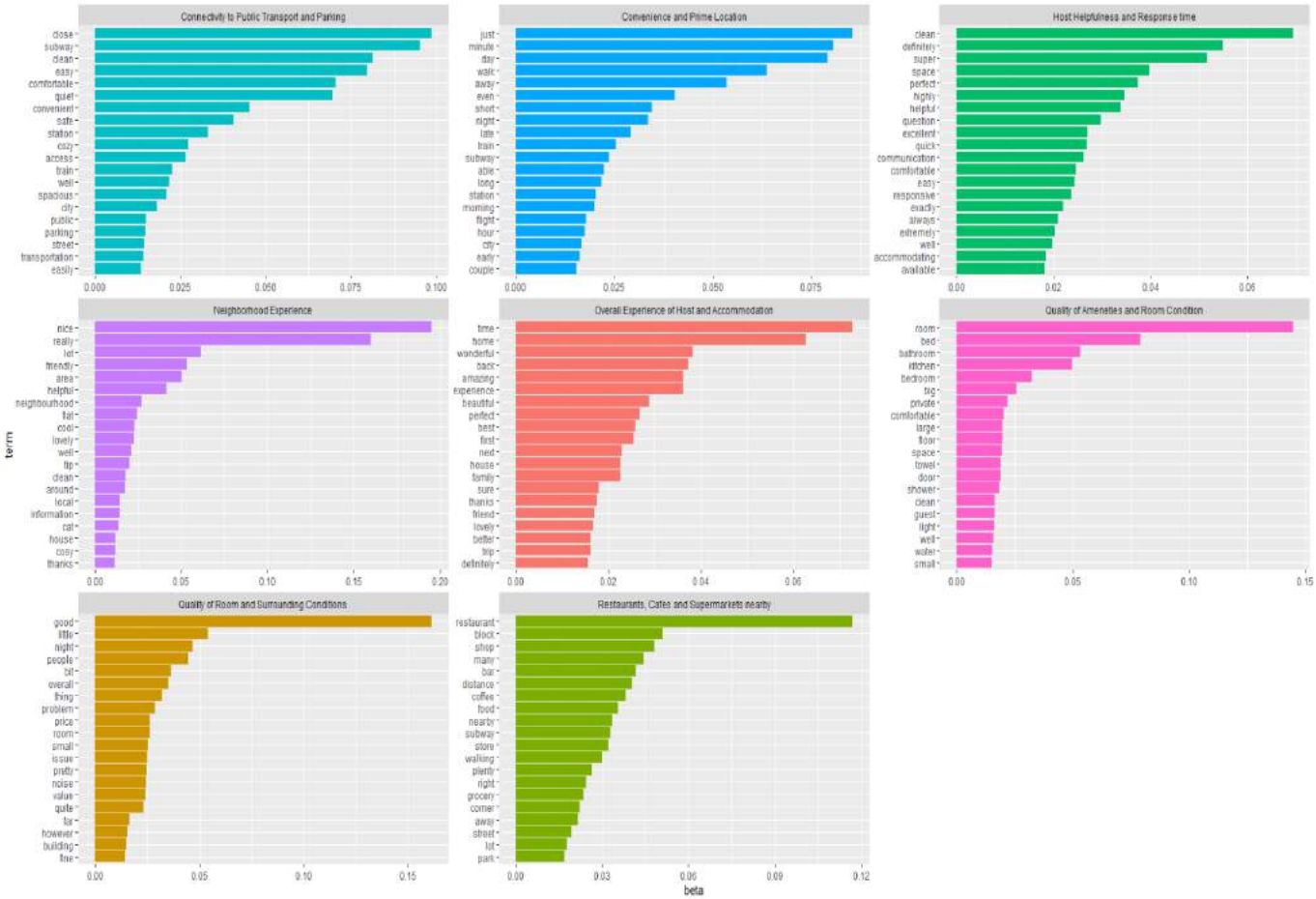



*Figure 45: Topic labelling with frequerncy counts of words in each topic*

In order to confirm there is no overlapping between topics and our topics are without high cross-loadings, we used the stm:: toLDAvis() package to visualise the topics. While choosing the topic names, two things were considered, words that have high term frequency in that topic and the words which were exclusive(almost) to that topic.
Hence, the following are the 8 topics and short reasoning for the title:

*Table 18: Topic labels*

| Topic | Label |
|---|---|
| 1 | Overall Experience of Host and Accommodation |
| 2 | Quality of Room and Surrounding Conditions |
| 3 | Restaurants, Cafes and Supermarkets nearby |
| 4 | Host Helpfulness and Response time |
| 5 | Connectivity to Public Transport and Parking |
| 6 | Convenience and Prime Location |
| 7 | Neighbourhood Experience |
| 8 | Quality of Amenities and Room Condition |

Interpreting the latent topic is highly dependent on how different the textual data is as per the aggregation level. If the data is very similar as per the aggregation, it is not possible to use such analysis for Topic Modeling. However, since the textual data for the given data is reasonably dissimilar per reviews to identify 8 different latent topics and we can visualise their PCA components with a biplot.

**Step 3: Compute the gamma matrix of the topics and plot the Biplot.**

We computed the gamma matrix of document and topic using the tidy function and using matrix = "gamma". We pivoted the matrix such that the columns had the topic names, and the rows were the documents.

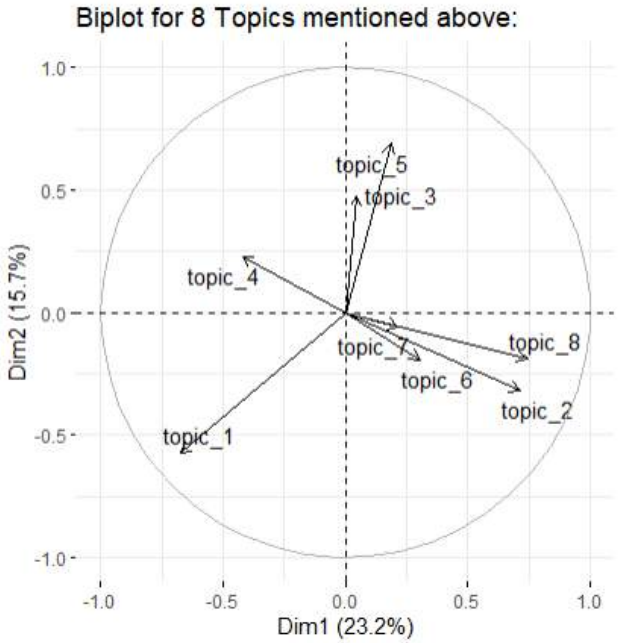



*Figure 46: Biplot of 8 topics*

The above biplot shows that almost all the latent topics are different, and the PCA analysis explains about 40% of the variance. This can be improved with rotation and Factor analysis. However, this out of the scope of this project. Also, according to the biplot, topic and 2 and 8 seem to be similar. This seems accurate since the words used to describe things inside the room and outside(surroundings) will be similar with a possibility of cross-loading.

**Step 4: Effects estimates of the review rating with respect to topics**

We estimated the effects of price and review rating on topic proportions using the estimated effect function in R.

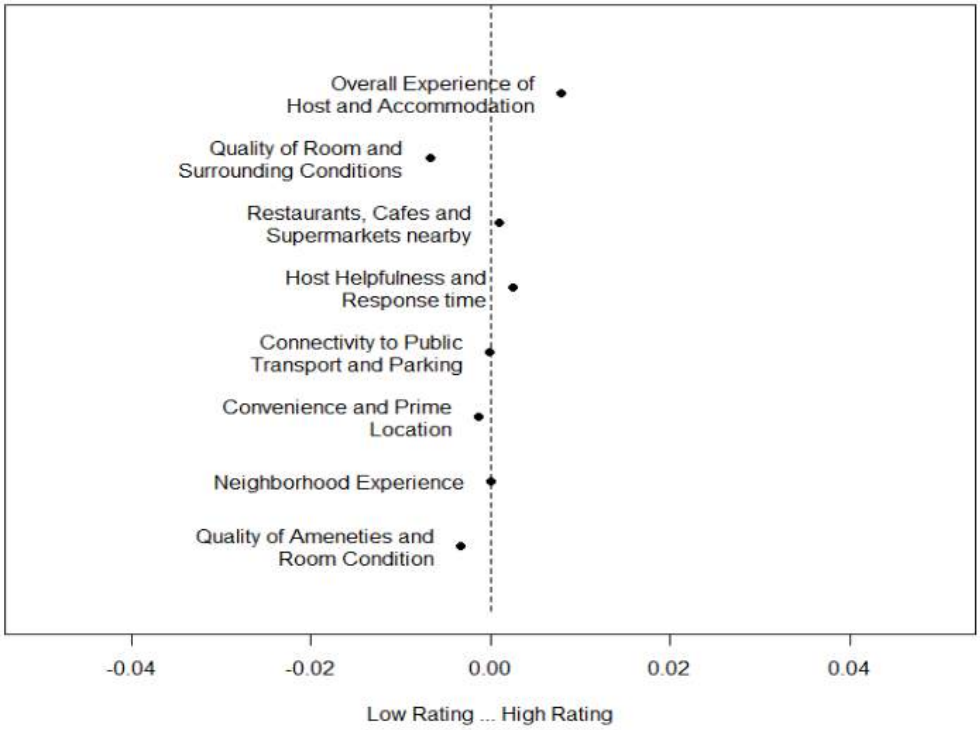



*Figure 47: Effect of Review rating on Topic proportions*

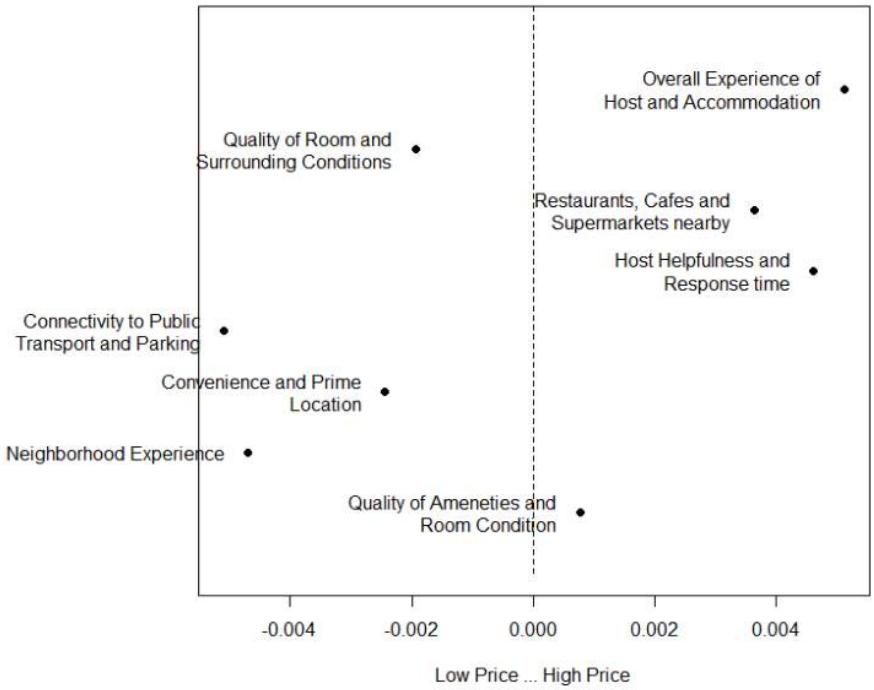



*Figure 48: Estimate of price on top proportions*

From the above figure, it is clear that Review Rating has any relation with the contribution of the topics. This is because of the highly left-skewed data, as most of the ratings are above 80 out of 100, resulting in low variance. Whereas, in spite of the skewness in price distribution, the "**Overall Experience of Host and Accommodation**", "**Restaurants, Cafes and Supermarkets nearby**", "**Host Helpfulness and Response time**" and "**Quality of Amenities and Room Condition**" have high contribution when the price is high where are rest of the topic have probabilities high when the price is at the lower end.

**Step 5: Regression analysis of topics with Reviews and Price**

We first do some data preparation to get the number of bathrooms, beds and amenities in each listing. This is now left joined with the sentiment scores that were obtained in part B. The all_senitments dataframe is used to left join, and the average sentiment per neighbourhood is calculated. The stepAIC function is used to select variables from the dataframe using a backward elimination procedure. The results for Reviews are as follows:

*Table 19:Linear regression of topics with review for review level aggregation*

```
Coefficients:
                                                 Estimate Std. Error t value Pr(>|t|)
(Intercept)                                     5.556e+01  1.918e-01 289.666  < 2e-16 ***
price                                           5.487e-04  3.986e-05  13.766  < 2e-16 ***
`Overall Experience of Host and Accommodation`  3.357e+01  2.135e-01 157.230  < 2e-16 ***
`Quality of Room and Surrounding Conditions`    2.614e+01  2.590e-01 100.937  < 2e-16 ***
`Restaurants, Cafes and Supermarkets nearby`    3.277e+01  2.245e-01 145.993  < 2e-16 ***
`Host Helpfulness and Response time`            3.279e+01  2.173e-01 150.914  < 2e-16 ***
`Connectivity to Public Transport and Parking`  3.272e+01  2.273e-01 143.946  < 2e-16 ***
`Convinience and Prime Location`                3.196e+01  2.334e-01 136.940  < 2e-16 ***
`Neighborhood Experience`                       3.247e+01  2.337e-01 138.913  < 2e-16 ***
total_excl_count                                7.858e-03  1.206e-04  65.142  < 2e-16 ***
bing_liu_sentiment                              1.113e+01  7.069e-02 157.498  < 2e-16 ***
nrc_sentiment                                   1.312e+00  8.136e-02  16.126  < 2e-16 ***
afinn_sentiment                                 1.587e-02  1.954e-04  81.204  < 2e-16 ***
loughran_sentiment                              1.962e-01  3.862e-02   5.081 3.77e-07 ***
num_bath                                        2.760e-01  1.715e-02  16.096  < 2e-16 ***
num_amenities                                   2.187e-02  2.900e-04  75.404  < 2e-16 ***
beds                                           -1.304e-01  4.786e-03 -27.249  < 2e-16 ***
room_typeHotel room                            -5.405e-01  1.180e-01  -4.579 4.67e-06 ***
room_typePrivate room                          -1.972e-01  1.125e-02 -17.530  < 2e-16 ***
room_typeShared room                            5.027e-01  4.289e-02  11.721  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.392 on 236096 degrees of freedom
Multiple R-squared:  0.5948,    Adjusted R-squared:  0.5948
F-statistic: 1.824e+04 on 19 and 236096 DF,  p-value: < 2.2e-16
```

Interpretations:

All the coefficients are significant with p value < 0.05. The review ratings of a listing increases with all the latent topic proportions. Hence directly proportional. However, we can see that the review ratings are indirectly proportional to the number of beds and ratings are lower if the room types are "hotel", "private room" but higher if "shared room".
The adjusted R-squared of the model is good at 59.48% indicating that the review scores are well explained by this model.

The results of price are as follows:

*Table 20 Linear Regression of topics with price*

```
Coefficients:
                                                 Estimate Std. Error  t value Pr(>|t|)
(Intercept)                                    413.25110   11.49196   35.960   <2e-16 ***
review_scores_rating                             1.46153    0.10617   13.766   <2e-16 ***
`Overall Experience of Host and Accommodation` -496.11272   11.53694  -43.002   <2e-16 ***
`Quality of Room and Surrounding Conditions`   -693.53078   13.57581  -51.086   <2e-16 ***
`Restaurants, Cafes and Supermarkets nearby`   -415.56813   12.06493  -34.444   <2e-16 ***
`Host Helpfulness and Response time`           -489.42850   11.69894  -41.835   <2e-16 ***
`Connectivity to Public Transport and Parking` -758.87754   12.13586  -62.532   <2e-16 ***
`Convinience and Prime Location`               -704.36262   12.43106  -56.662   <2e-16 ***
`Neighborhood Experience`                      -742.03318   12.45241  -59.590   <2e-16 ***
total_excl_count                                -0.05669    0.00628   -9.027   <2e-16 ***
bing_liu_sentiment                              40.91918    3.83436   10.672   <2e-16 ***
nrc_sentiment                                  -58.98146    4.19923  -14.046   <2e-16 ***
afinn_sentiment                                 0.11860    0.01022   11.605   <2e-16 ***
loughran_sentiment                             37.12516    1.99161   18.641   <2e-16 ***
num_bath                                       98.34078    0.86213  114.067   <2e-16 ***
num_amenities                                  -0.13231    0.01514   -8.737   <2e-16 ***
beds                                           15.30173    0.24538   62.359   <2e-16 ***
room_typeHotel room                           237.58188    6.07136   39.132   <2e-16 ***
room_typePrivate room                         -62.92417    0.56621 -111.132   <2e-16 ***
room_typeShared room                          -60.73404    2.21038  -27.477   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 123.5 on 236096 degrees of freedom
Multiple R-squared:  0.2524,    Adjusted R-squared:  0.2523
F-statistic:  4195 on 19 and 236096 DF,  p-value: < 2.2e-16
```

All the coefficients are significant with p values <0.05 with p values >0.05. The price of the listings decreases with all the latent topics. More research needs to be done regarding the price regression model since this analysis is done by presuming that the required conditions for multiple linear regression are satisfied. The adjusted R squared is at 25.23%. This can be improved by normalising the price distribution and feature engineering with additional hidden features.

### Aggregation by Listing:

We used the dataframe from the last step to group by listing id and pasted the annotated comments in one row per listing. We cleaned up the price column and omitted the NA values. Post this we used the Textprocessor function to remove stopwords and perform basic text processing. The stopwords are decided iteratively through steps 1 and 2 and augmented into the stopwords list. This output is stored and used in the prepDocuments function to prepare the data for topic modelling. We choose to remove words that appear in less than 1% of the documents and set this as the threshold. Since the price and review scores are per listing, we don't have to create an average value for these attributes.

**Step 1: Choosing Kappa: Number of topics**

We use the stm function to calculate the number of topics by setting K = 0 and setting the prevalence measure to be price and review rating. We found out a total of 40 topics. These are the total number of possible topics. We inspected these topics and found out that there is a considerable number of overlapping topics that had similar words and described the same themes. Hence we could consolidate them into 12 topics and decided to run the searchK function from K = 5 to K = 14 to decide on the exact K that was needed. The plot of the held-out likelihood, residuals and semantic coherence is shown below.

It can be seen that the held-out likelihood is maximum at K= 8 topics and the residuals also are the lowest at 8. This indicates that the ideal number of topics are 8. The semantic coherence drops at 8 indicating that at K =8, the most probable words don't frequently co-occur together. In the next step, the topic labelling is done.

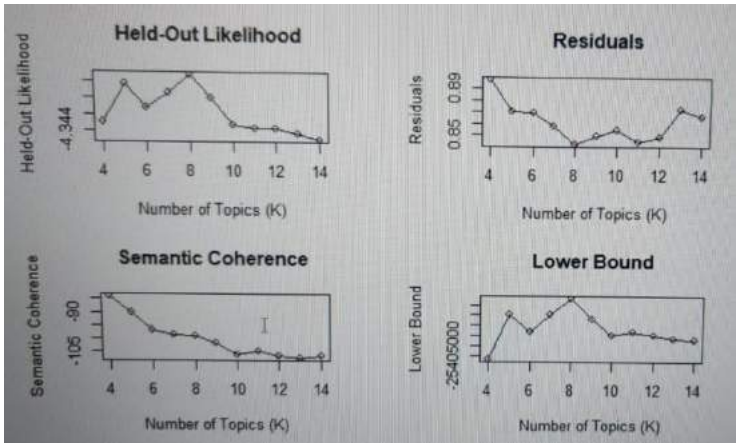



*Figure 49: Diagnostic values for Listin. Held out likelihood and Semantic coherence*

**Step 2: Building beta matrix and Topic Labelling**

Next, we used the tidy function to build the beta matrix for the topics. We then sliced the top 10 words according to descending beta values for each topic and plotted them as horizontal bar plots to inspect the words in each topic. This helped us decide the topic labels.

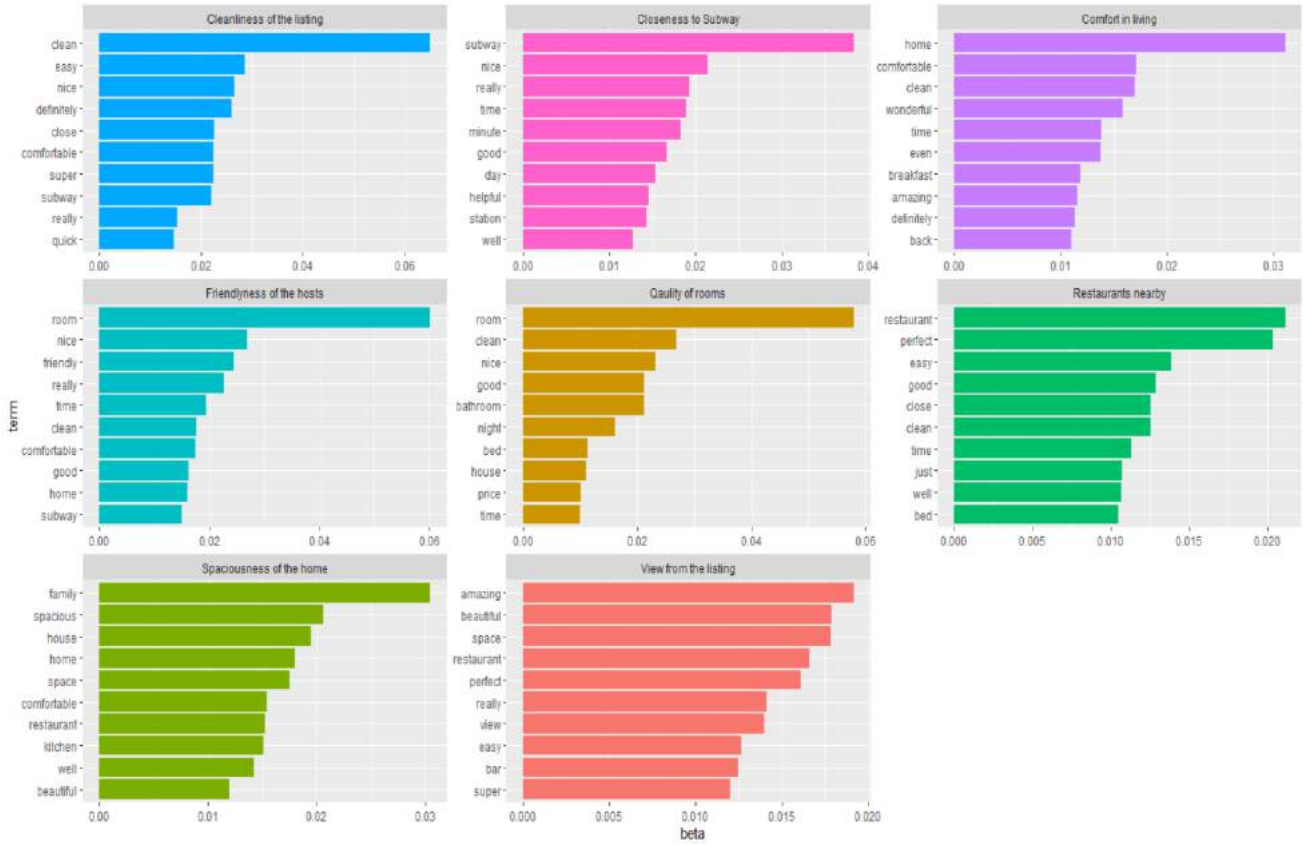



*Figure 50: Topics for Listings*

*Table 21: Topic Labelling*

| Topic# | Topic 1 | Topic 2 | Topic 3 | Topic 4 |
|---|---|---|---|---|
| Label | **View from the listing** | **Quality of rooms** | **Spaciousness of the home** | **Restaurants nearby** |
| Topic# | Topic 7 | Topic 8 | Topic 9 | Topic 10 |
| Label | **Friendlyness of the hosts** | **Cleanliness of the listing** | **Comfort of living** | **Closeness to Subway** |

The topics are reasonably different from each other though topics 2,8 and 3 seem to be similar. We plotted the biplot in the next step to inspect this.

**Step 3: Compute the gamma matrix of the topics and plot the Biplot**

We computed the gamma matrix of document and topic using the tidy function and using matrix = "gamma". We pivoted the matrix such that the columns had the topic names and the rows were the documents.

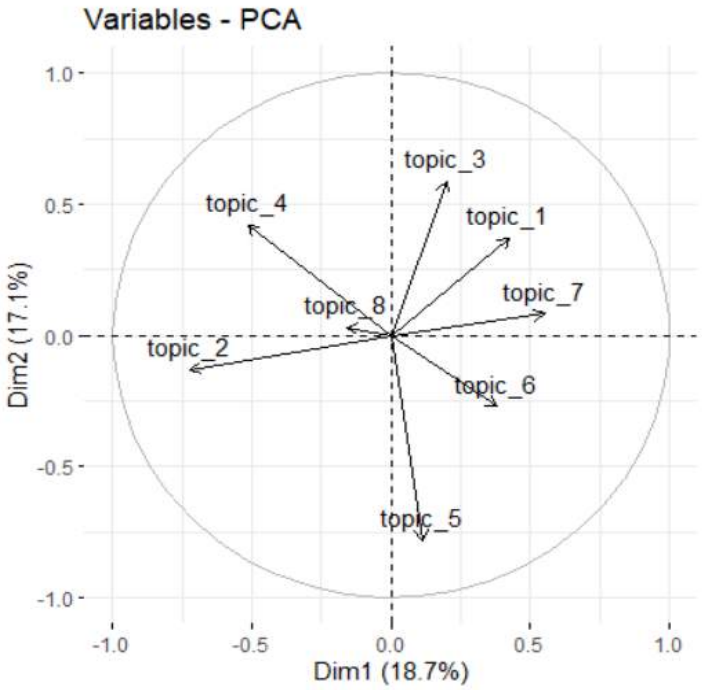



*Figure 51: Biplot of topics in listings*

The topics above are reasonably differentiated from each other in the two dimensions of the PCA variables. Topics 2 and 8 are a little close to each other which was expected.

**Step 4: Effects estimates of the review rating with respect to topics.**

WE estimated the effects of price and review rating on topic proportions using the estimateEffect function in R.

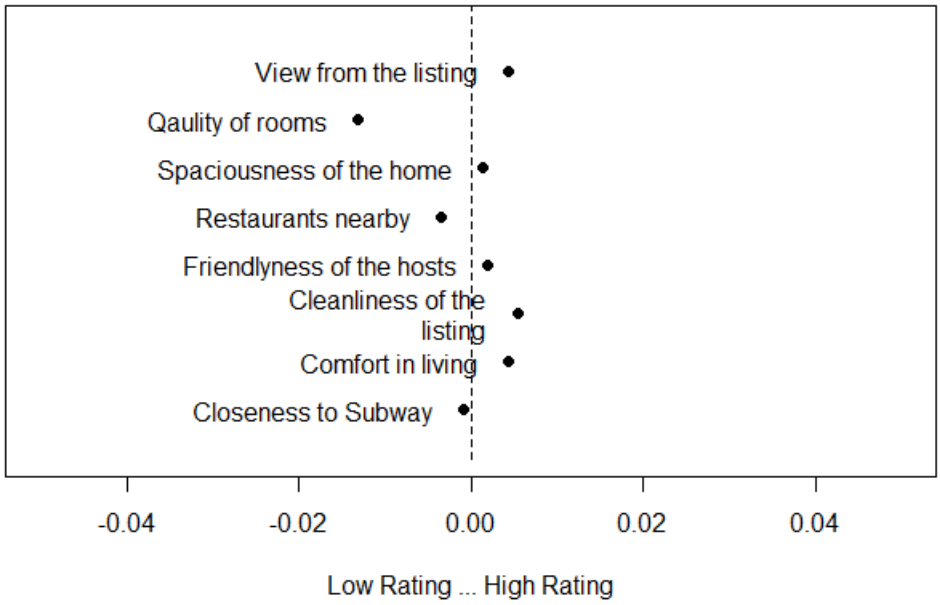



*Figure 52: Effects of Rating on Listing topics*

It can be seen that high rating contributes to the topics "View from the listing", "Spaciousness of the home", "Friendliness of the hosts", Cleanliness and comfort of the living whereas low ratings contribute to Restaurants nearby and Quality of rooms topics.

**Step 5: Regression analysis of topics with Reviews and Price**

We first do some data preparation to get the number of bathrooms, beds and amenities in each listing. This is now left joined with the sentiment scores that were obtained in part B. The all_senitments dataframe is used to left join and the average sentiment per neighbourhood is calculated. The stepAIC function is used to select variables from the dataframe using a backward elimination procedure. The results for Reviews are as follows:

*Table 22: Linear Regression of topics with reviews*

```
Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)              81.832766   0.552431 148.132  < 2e-16 ***
`View from the listing`  11.599987   0.606292  19.133  < 2e-16 ***
`Qaulity of rooms`       -7.788855   0.661084 -11.782  < 2e-16 ***
`Spaciousness of the home`  9.431334   0.752192  12.538  < 2e-16 ***
`Restaurants nearby`        3.873671   0.612332   6.326 2.64e-10 ***
`Friendlyness of the hosts`  6.376862   0.686551   9.288  < 2e-16 ***
`Cleanliness of the listing` 16.148254   0.656703  24.590  < 2e-16 ***
`Comfort in living`      15.835799   0.701273  22.581  < 2e-16 ***
bing_liu_sentiment        4.804221   0.309706  15.512  < 2e-16 ***
afinn_sentiment           0.014687   0.001309  11.224  < 2e-16 ***
num_bath                  0.520614   0.133143   3.910 9.29e-05 ***
num_amenities             0.025837   0.002770   9.328  < 2e-16 ***
beds                     -0.188276   0.050248  -3.747 0.000180 ***
room_typeHotel room       1.224529   0.641896   1.908 0.056466 .
room_typePrivate room     1.079644   0.144159   7.489 7.59e-14 ***
room_typeShared room      1.463901   0.407177   3.595 0.000326 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.128 on 8909 degrees of freedom
Multiple R-squared:  0.5054,    Adjusted R-squared:  0.5045
F-statistic: 606.8 on 15 and 8909 DF,  p-value: < 2.2e-16
```

Interpretations:

All the coefficients are significant with p value < 0.05 except of room_typeHotel room variable which has a p value of 0.056 which is a borderline case. The review ratings of a listing increases when most of the topic proportions increase except for "Quality of rooms" topic and beds variable increases. We can see that the review scores are increasing with increasing sentiment and other topics.

The adjusted R-squared of the model is good at 50.45% indicating that the review scores are well explained by this model.

The results of price are as follows:

*Table 23 Linear Regression of topics with price*

```
Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)               -292.4608    41.0988  -7.116 1.20e-12 ***
`View from the listing`    587.1105    44.7070  13.132  < 2e-16 ***
`Qaulity of rooms`         280.7589    48.3516   5.807 6.59e-09 ***
`Spaciousness of the home` 721.4250    55.0949  13.094  < 2e-16 ***
`Restaurants nearby`       504.7197    44.9740  11.222  < 2e-16 ***
`Friendlyness of the hosts`  76.5034    50.2649   1.522 0.128044
`Cleanliness of the listing` 113.1761  47.9560   2.360 0.018296 *
`Comfort in living`        342.1305    51.2872   6.671 2.69e-11 ***
total_excl_count             0.3450     0.1474   2.340 0.019297 *
bing_liu_sentiment         111.5770    26.8592   4.154 3.30e-05 ***
afinn_sentiment             -0.4116     0.1585  -2.597 0.009416 **
loughran_sentiment         -26.6533    15.5459  -1.714 0.086473 .
num_bath                    57.7224     9.6957   5.953 2.73e-09 ***
num_amenities               -0.4519     0.2018  -2.239 0.025179 *
beds                        17.9795     3.6600   4.913 9.15e-07 ***
room_typeHotel room         33.9461    46.7595   0.726 0.467874
room_typePrivate room       27.8272    10.5011   2.650 0.008065 **
room_typeShared room       113.6630    29.6517   3.833 0.000127 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 300.6 on 8907 degrees of freedom
Multiple R-squared:  0.1017,    Adjusted R-squared:  0.09995
F-statistic: 59.29 on 17 and 8907 DF,  p-value: < 2.2e-16
```

All the coefficients are significant with p values <0.05 except for "Friendlyness of the hosts", Loughran sentiment and Hotel room which have p values >0.05. The price of the listings increases with most of the topics and variables except for afinn_sentiment, Loughran sentiment and the number of amenities. The topic proportions indicate the quality, spaciousness, proximity to public transport etc. and this has a positive effect on price which is expected. The R squared of this model is low since the price is a variable that can be affected by many factors which may not have been captured in this data. The adjusted R squared is at 10%.

## Aggregation by Neighbourhood

We used the dataframe from the last step to group by neighbourhood_cleansed and paste the annotated comments in one row per neighbourhood. We cleaned up the price column and omitted the NA values. Post this we used the Textprocessor function to remove stopwords and perform basic text processing. The stopwords are decided iteratively through steps 1 and 2 and augmented into the stopwords list. This output is stored and used in the prepDocuments function to prepare the data for topic modelling. We choose to remove words that appear in less than 1% of the documents and set this as the threshold. We also created average price and review score columns per neighbourhood.

### Step 1: Choosing Kappa: Number of topics

We use the stm function to calculate the number of topics by setting K = 0 and setting the prevalence measure to be price and review rating. We found out a total of 28 topics. These are the total number of possible topics. We inspected these topics and found out that there is a considerable number of overlapping topics that had similar words and described the same themes. Hence we could consolidate

them into 12 topics and decided to run the searchK function from K = 4 to K = 14 to decide on the exact K that was needed. The plot of the held-out likelihood, residuals and semantic coherence is shown below.

It can be seen that the held-out likelihood is maximum at K= 12 topics and the residuals also are the lowest at 12. This indicates that the ideal number of topics is 12. The semantic coherence drops sharply at 12 indicating that at K =12 the most probable words don't frequently co-occur together.

Hence the number of topics is chosen to be 12. In the next step, the topic labelling is done.

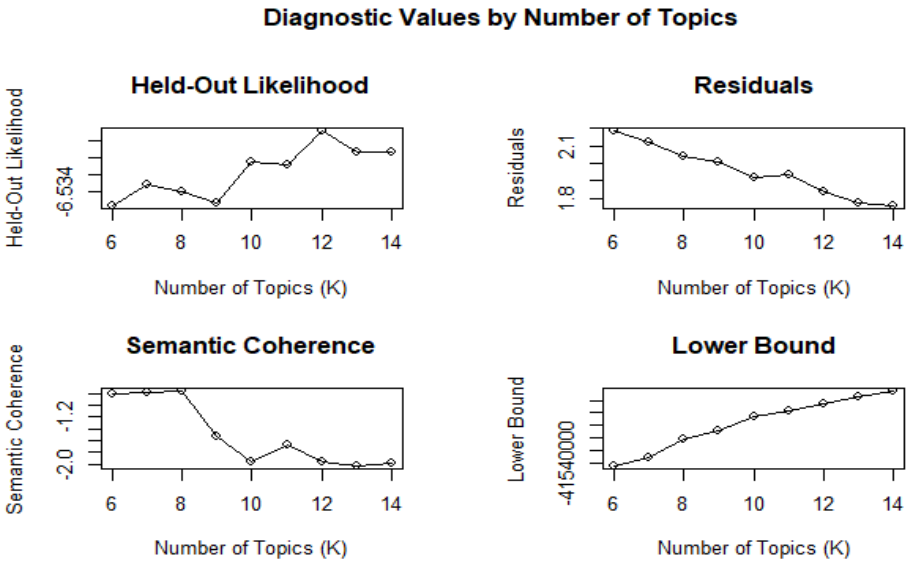



*Figure 53: Diagnostic values - Held-out likelihood, Semnatic Coherence*

**Step 2: Building beta matrix and Topic Labelling**

Next, we used the tidy function to build the beta matrix for the topics. We then sliced the top 10 words according to descending beta values for each topic and plotted them as horizontal bar plots to inspect the words in each topic. This helped us decide the topic labels.


*Figure 54: Topics for Neighbourhoods*

*Table 24: Topic labelling for neighborhoods*

| Topic# | Topic 1 | Topic 2 | Topic 3 | Topic 4 | Topic 5 | Topic 6 |
|---|---|---|---|---|---|---|
| Label | **Proximity to Subway** | **Comfiness and coziness of the room** | **Room check-in time** | **Closeness to Ferry and beach** | **Comfort of the room** | **Niceness of the room** |
| Topic# | Topic 7 | Topic 8 | Topic 9 | Topic 10 | Topic 11 | Topic 12 |
| Label | **Spaciousness of the home** | **Restaurants near the listing** | **Comfort of the house** | **Quality of restaurants nearby** | **Comfort of reaching subway** | **Perfect restaurants nearby** |

There are a couple of topics that are overlapping like topics 1, 11 and topics 8 and 12. This is expected as the aggregation is on the neighbourhood and there may not be too much variation among different neighbourhoods and many may have similar themes like closeness to subway and comfortable living space.

**Step 3: Compute the gamma matrix of the topics and plot the Biplot**

We computed the gamma matrix of document and topic using the tidy function and using matrix = "gamma". We pivoted the matrix such that the columns had the topic names and the rows were the documents.

We then plotted the biplot of the topics by first calculating the PCA values using the FactorMineR::PCA function and then plotting using the factoextra::fviz_pca_var function.

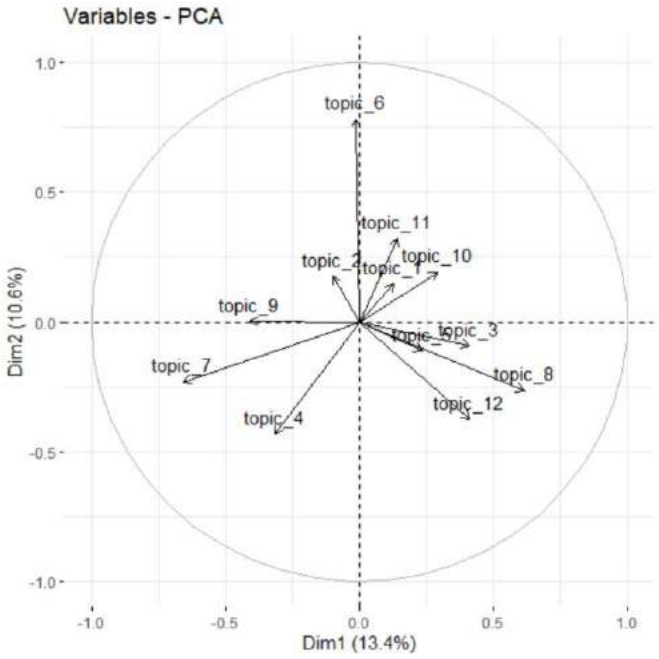



*Figure 55: Biplot of Neighbourhood Topics*

The biplot shows the correlation of topics concerning different dimensions. As suspected, the topics that are similar are also nearby in the biplots. Topics 8 and 12 which talk about restaurants are clustered together and so are topics 1 and 11 that are about subway proximity.

**Step 4: Regression with Price and Reviews**

We first do some data preparation to get the number of bathrooms, beds and amenities in each neighbourhood by grouping by and find the mean values. This is now left joined with the sentiment scores that were obtained in part B. The all_senitments dataframe is used to left join and the average sentiment per neighbourhood is calculated. The stepAIC function is used to select variables from the data frame using a backward elimination procedure. The results for Reviews are as follows:

*Table 25: Linear Regression of topics with review*

```
Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                  91.721246   1.000996  91.630  < 2e-16 ***
`Room check in time`         -3.054034   0.836167  -3.652 0.000349 ***
`Niceness of the room`       -3.129232   0.480571  -6.511 8.58e-10 ***
`Restaurants near the listing` -1.531288   0.632672  -2.420 0.016591 *
`Comfort of the house`       -2.268489   0.530807  -4.274 3.24e-05 ***
`Quality of restuarants nearby` -1.728999   1.178311  -1.467 0.144183
avg_bing                      6.208512   1.312385   4.731 4.78e-06 ***
avg_afinn                     0.009016   0.005008   1.800 0.073651 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.386 on 165 degrees of freedom
Multiple R-squared:  0.4486,    Adjusted R-squared:  0.4252
F-statistic: 19.17 on 7 and 165 DF,  p-value: < 2.2e-16
```

Interpretations: The coefficients are all significant with p values < 0.05 except for "Quality of restaurants nearby" topic and avg_afinn with a p-value of 0.07, which is a borderline case. The review rating of a neighbourhood

1. Decreases with increasing check-in times topic proportions in the neighbourhood.
2. Decreases with increasing niceness of the room topic proportions in the neighbourhood. This is because when there are a greater number of nice rooms in a neighbourhood, then the competition increases, and so does the expectation from the customer.
3. Decreases with increasing restaurants topic proportions in the neighbourhood.
4. Decrease with the comfort of the house topic proportions in the neighbourhood.
5. Decreases with Quality of restaurants nearby – although this coefficient is not significant
6. Increases with average bing_liu and afinn sentiment scores and is predominant with bing sentiment.

The predictability of review from this model is given by the adjusted R-squared value, which is at 42.52%

The results of Price are as follows:

*Table 26 Linear Regression of topics with price*

```
Coefficients:
                               Estimate Std. Error t value Pr(>|t|)
(Intercept)                      78.744     18.085   4.354 2.36e-05 ***
`Proximity to Subway`           -79.974     13.942  -5.736 4.63e-08 ***
`Room check in time`             69.646     19.753   3.526 0.000549 ***
`Comfort of the room`           -96.553     36.190  -2.668 0.008407 **
`Niceness of the room`          -96.395     12.292  -7.842 5.61e-13 ***
`Spaciousness of home`          -51.465     11.625  -4.427 1.75e-05 ***
`Restaurants near the listing`   99.076     16.765   5.910 1.96e-08 ***
`Comfort of the house`          -62.716     13.057  -4.803 3.53e-06 ***
`Comfort of reaching subway`    -46.446     27.197  -1.708 0.089596 .
avg_bath                         45.240     13.453   3.363 0.000962 ***
avg_beds                         11.946      4.351   2.745 0.006726 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 31.14 on 162 degrees of freedom
Multiple R-squared:  0.6741,    Adjusted R-squared:  0.654
F-statistic: 33.51 on 10 and 162 DF,  p-value: < 2.2e-16
```

Figure 3.5.6

Interpretations: The coefficients are all significant except for the "comfort of reaching subway" topic, which has a p-value of 0.09. The price of listings in a neighbourhood

1. Decreases as proximity to subway topic proportions increases.
2. Increases as Room check in time topic proportions increases.
3. Decreases as comfort of the room topic proportions increases.

4. Decreases as Niceness of the room topic proportions increases.
5. Decreases as Spacoiusness of the home topic proportions increases.
6. Increases as Restaurants near the lsiting topic proportions increases.
7. Decreases as comfort of the house topic proportions increases.
8. Decreases as comfort of reaching subway topic proportions increases.
9. Increases as avg_bathrooms increase.
10. Increase as avg beds increases.

The predictability of review from this model is given by the adjusted R-squared value, which is at 65.4%

The regression model has good predictability. However, there are a few odd signs in the coefficients which may need further investigation. The variable needs to be checked for normality, homoscedasticity, normality of residuals and multicollinearity. The normality and multicollinearity assumptions may have been violated, and hence the coefficients are negative for some topics.

# Bibliography


Baptiste Auguie (2017). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.3. https://CRAN.R-project.org/package=gridExtra

Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006. https://igraph.org

Cheng, M., & Jin, X. (2019). What do Airbnb users care about? An analysis of online review comments. International Journal of Hospitality Management,76, 58-70.

Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1-25. URL https://www.jstatsoft.org/v40/i03/.

Grün B, Hornik K (2011). "topicmodels: An R Package for Fitting Topic Models." _Journal of Statistical Software_, *40*(13), 1-30. doi: 10.18637/jss.v040.i13 (URL: https://doi.org/10.18637/jss.v040.i13).

Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2021). dplyr: A Grammar of Data Manipulation. R package version 1.0.6. https://CRAN.R-project.org/package=dplyr

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Hadley Wickham and Jim Hester (2020). readr: Read Rectangular Text Data. R package version 1.4.0. https://CRAN.R-project.org/package=readr

Hadley Wickham (2021). rvest: Easily Harvest (Scrape) Web Pages. R package version 1.0.0. https://CRAN.R-project.org/package=rvest

Hadley Wickham (2019). stringr: Simple, Consistent Wrappers for Common String Operations. R package version 1.4.0. https://CRAN.R-project.org/package=stringr

Hadley Wickham (2021). tidyr: Tidy Messy Data. R package version 1.1.3. https://CRAN.R-project.org/package=tidyr

Hadley Wickham, Jim Hester and Jeroen Ooms (2020). xml2: Parse XML. R package version 1.3.2. https://CRAN.R-project.org/package=xml2

Hornik K, Mair P, Rauch J, Geiger W, Buchta C, Feinerer I (2013). "The textcat Package for $n$-Gram Based Text Categorization in R." _Journal of Statistical Software_, *52*(6), 1-17. doi: 10.18637/jss.v052.i06 (URL: https://doi.org/10.18637/jss.v052.i06).

Ian Fellows (2018). wordcloud: Word Clouds. R package version 2.6. https://CRAN.R-project.org/package=wordcloud

Ingo Feinerer and Kurt Hornik (2020). tm: Text Mining Package. R package version 0.7-8. https://CRAN.R-project.org/package=tm

Jan Wijffels (2020). udpipe: Tokenisation, Parts of Speech Tagging, Lemmatization and Dependency Parsing with the 'UDPipe' 'NLP' Toolkit. R package version 0.8.5. https://CRAN.R-project.org/package=udpipe

Jeroen Ooms (2020). cld2: Google's Compact Language Detector 2. R package version 1.2.1. https://CRAN.R-project.org/package=cld2

Jeroen Ooms (2014). The jsonlite Package: A Practical and Consistent Mapping Between JSON Data and R Objects. arXiv:1403.2805 [stat.CO] URL https://arxiv.org/abs/1403.2805.

Julia Silge (2017). janeaustenr: Jane Austen's Complete Novels. R package version 0.1.5. https://CRAN.R-project.org/package=janeaustenr

Kamil Slowikowski (2021). ggrepel: Automatically Position Non-Overlapping Text Labels with 'ggplot2'. R package version 0.9.1. https://CRAN.R-project.org/package=ggrepel

Luo, Y., & Tang, R. L. (2019). Understanding hidden dimensions in textual reviews on Airbnb: An application of modified latent aspect rating analysis (LARA). International Journal of Hospitality Management, 80, 144-154.

Mouselimis L (2021). _geojsonR: A GeoJson Processing Toolkit_. R package version 1.1.0, <URL: https://CRAN.R-project.org/package=geojsonR>.

Rinker, T. W. (2020). qdap: Quantitative Discourse Analysis Package. 2.4.2. Buffalo, New York. https://github.com/trinker/qdap

Roberts ME, Stewart BM, Tingley D (2019). "stm: An R Package for Structural Topic Models." _Journal of Statistical Software_, *91*(2), 1-40. doi: 10.18637/jss.v091.i02 (URL: https://doi.org/10.18637/jss.v091.i02).

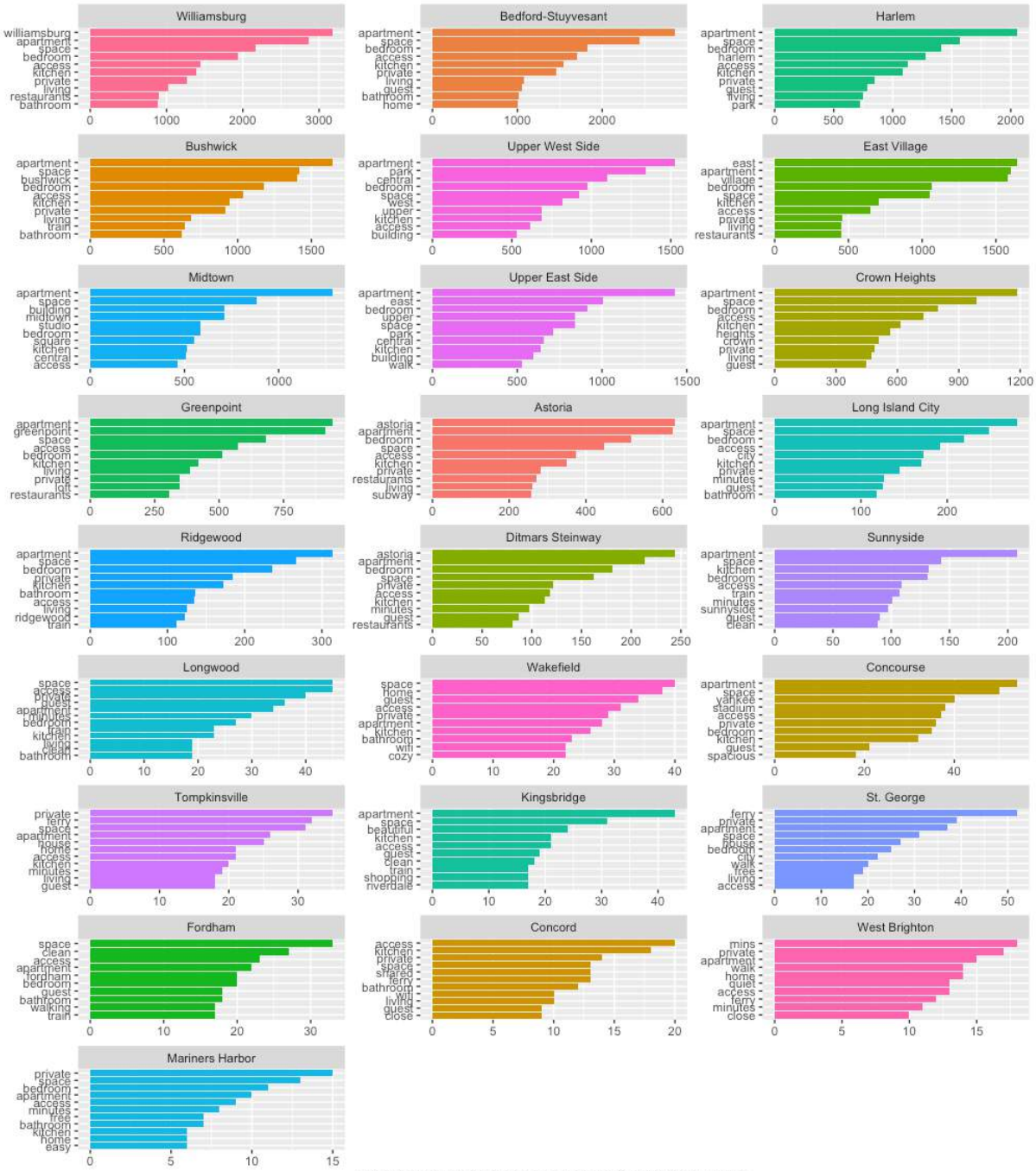
Silge J, Robinson D (2016). "tidytext: Text Mining and Analysis Using Tidy Data Principles in R." _JOSS_, *1*(3). doi: 10.21105/joss.00037 (URL: https://doi.org/10.21105/joss.00037), <URL: http://dx.doi.org/10.21105/joss.00037>.

Simon Urbanek (2021). rJava: Low-Level R to Java Interface. R package version 1.0-4. https://CRAN.R-project.org/package=rJava

Stefan Milton Bache and Hadley Wickham (2020). magrittr: A Forward-Pipe Operator for R. R package version 2.0.1. https://CRAN.R-project.org/package=magrittr

# Appendix

Figure A.1 Description word frequency among top 25 neighbourhoods




Description word frequency among top 25 neighbourhoods

Figure A.2

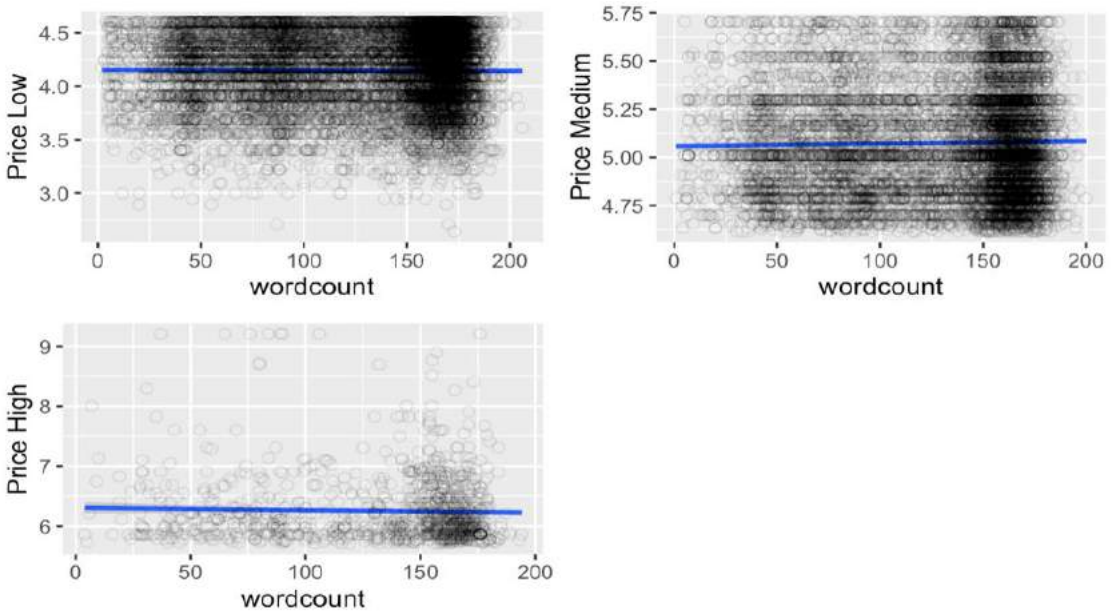



Figure A.3

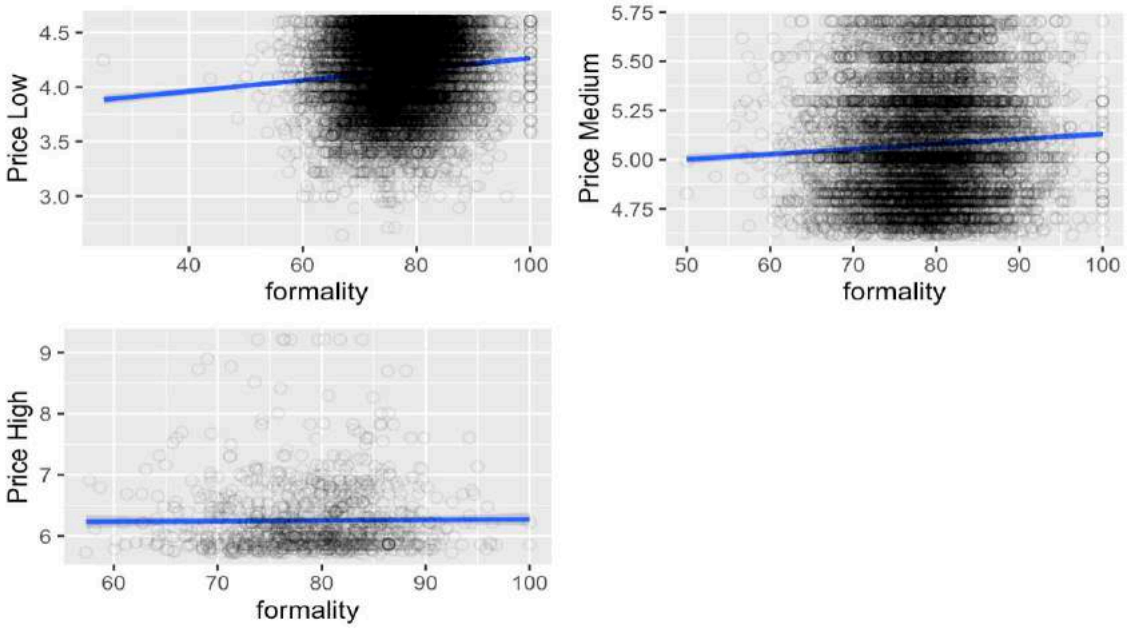


Figure A.4

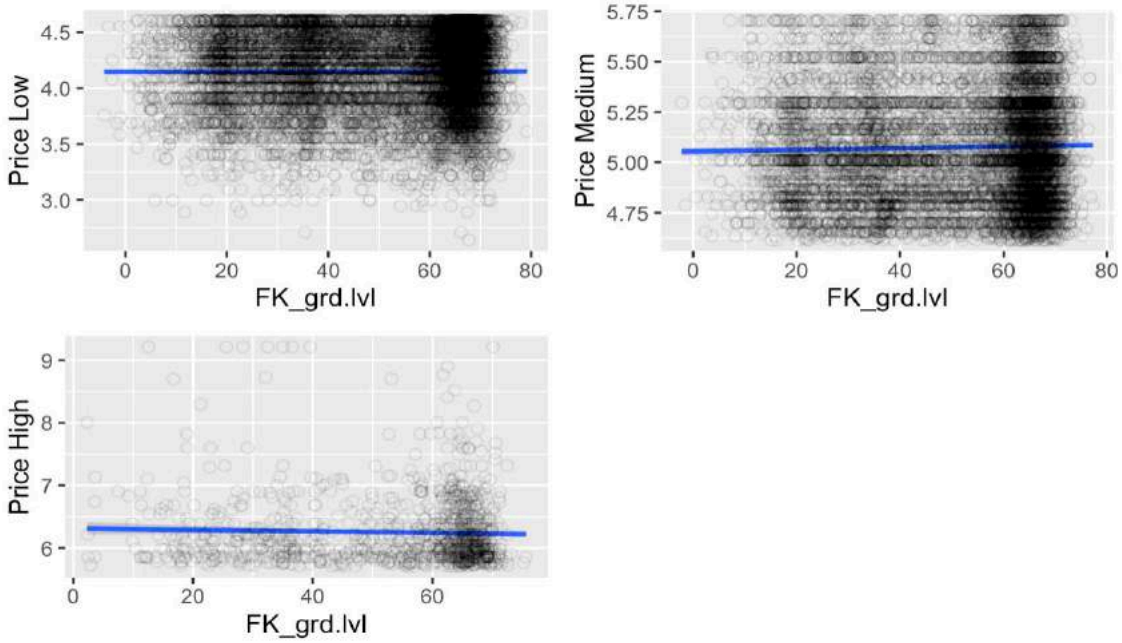



Figure A.5

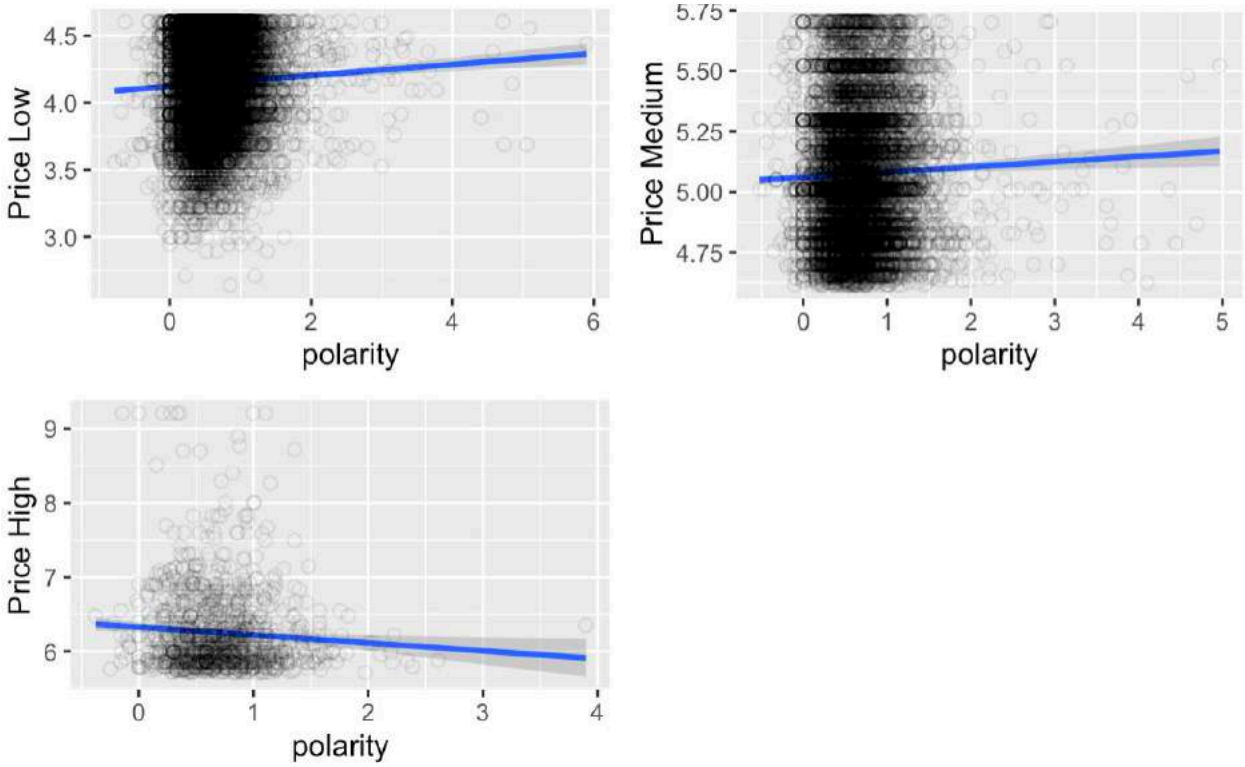


Figure B.1

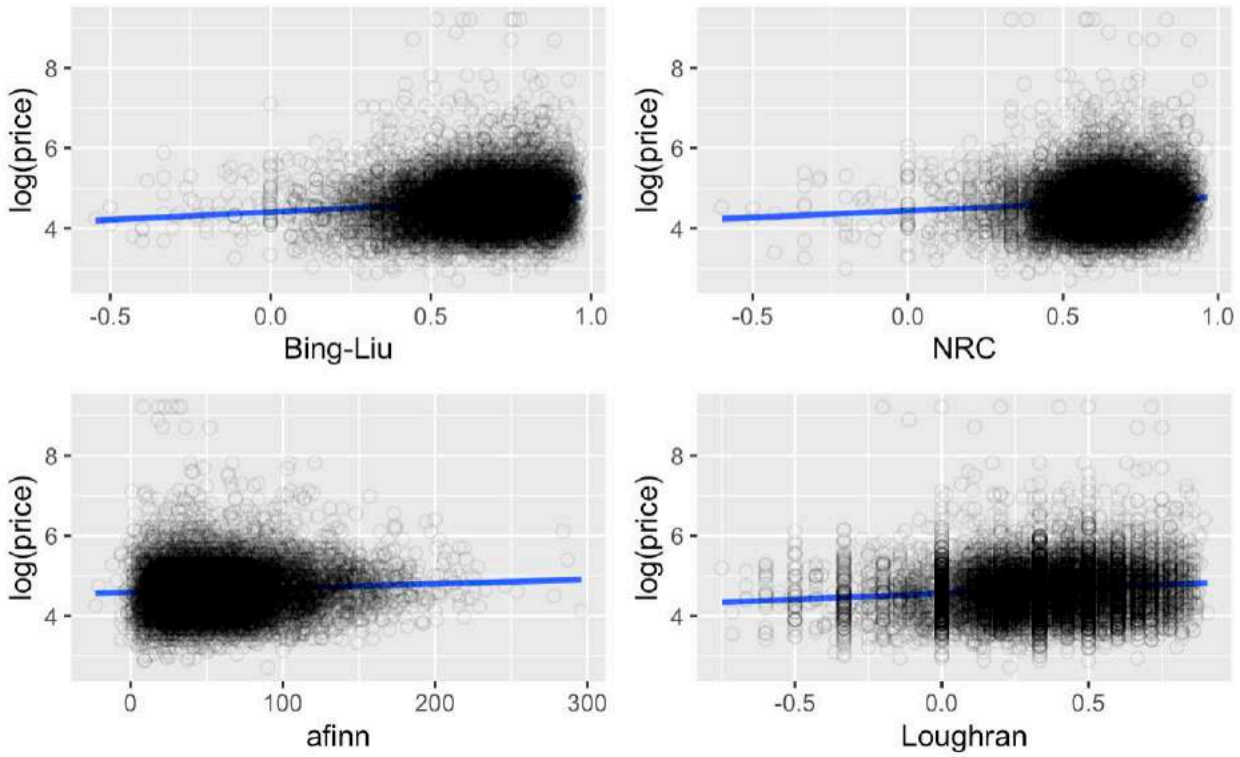



Figure B.2

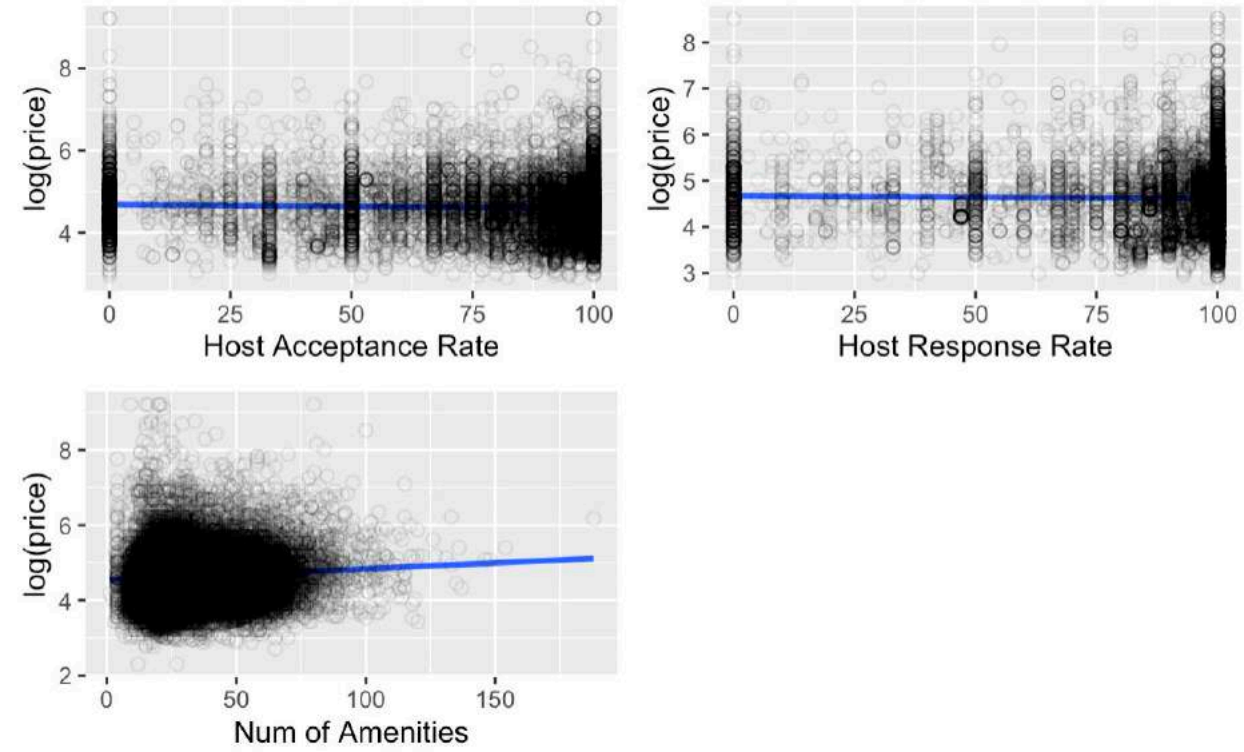
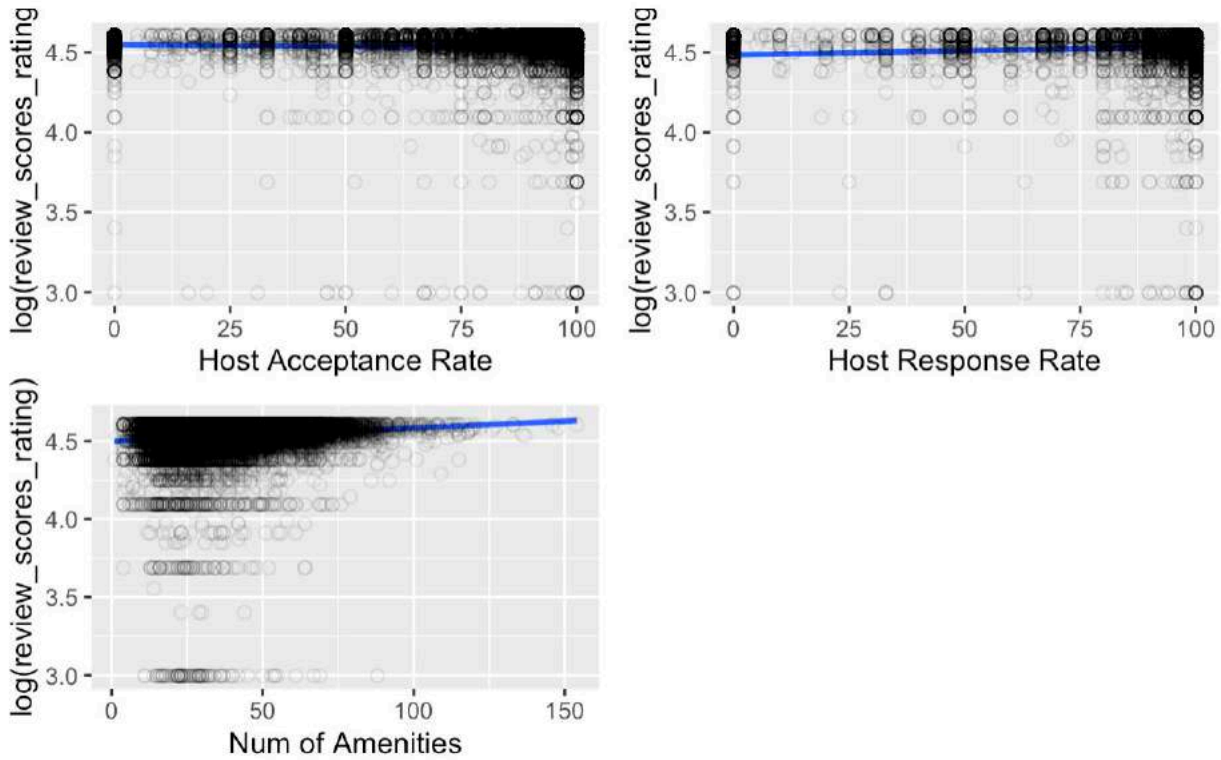


Figure B.3

| | Dependent variable: | | |
|---|---|---|---|
| | price) | | |
| | (1) | (2) | (3) |
| host_acceptance_rate | -0.001*** | | |
| | (0.0002) | | |
| host_response_rate | | -0.001** | |
| | | (0.0003) | |
| num_amenities | | | 0.003*** |
| | | | (0.0002) |
| Constant | 4.683*** | 4.673*** | 4.535*** |
| | (0.017) | (0.024) | (0.010) |
| Observations | 12,186 | 11,212 | 25,432 |
| R2 | 0.002 | 0.0004 | 0.006 |
| Adjusted R2 | 0.002 | 0.0003 | 0.006 |
| Residual Std. Error | 0.716 (df = 12184) | 0.729 (df = 11210) | 0.696 (df = 25430) |
| F Statistic | 20.994*** (df = 1; 12184) | 4.440** (df = 1; 11210) | 156.206*** (df = 1; 25430) |
| Note: | | | *p<0.1; **p<0.05; ***p<0.01 |

Figure B.4




]

Figure B.5

| | Dependent variable: | | |
|---|---|---|---|
| | review_scores_rating) | | |
| | (1) | (2) | (3) |
| host_acceptance_rate | -0.0002*** | | |
| | (0.0001) | | |
| host_response_rate | | 0.001*** | |
| | | (0.0001) | |
| num_amenities | | | 0.001*** |
| | | | (0.0001) |
| Constant | 4.548*** | 4.484*** | 4.498*** |
| | (0.004) | (0.006) | (0.003) |
| Observations | 9,540 | 8,528 | 18,537 |
| R2 | 0.002 | 0.007 | 0.011 |
| Adjusted R2 | 0.002 | 0.007 | 0.011 |
| Residual Std. Error | 0.152 (df = 9538) | 0.150 (df = 8526) | 0.150 (df = 18535) |
| F Statistic | 21.596*** (df = 1; 9538) | 57.497*** (df = 1; 8526) | 199.801*** (df = 1; 18535) |
| Note: | | | *p<0.1; **p<0.05; ***p<0.01 |

Figure B.6

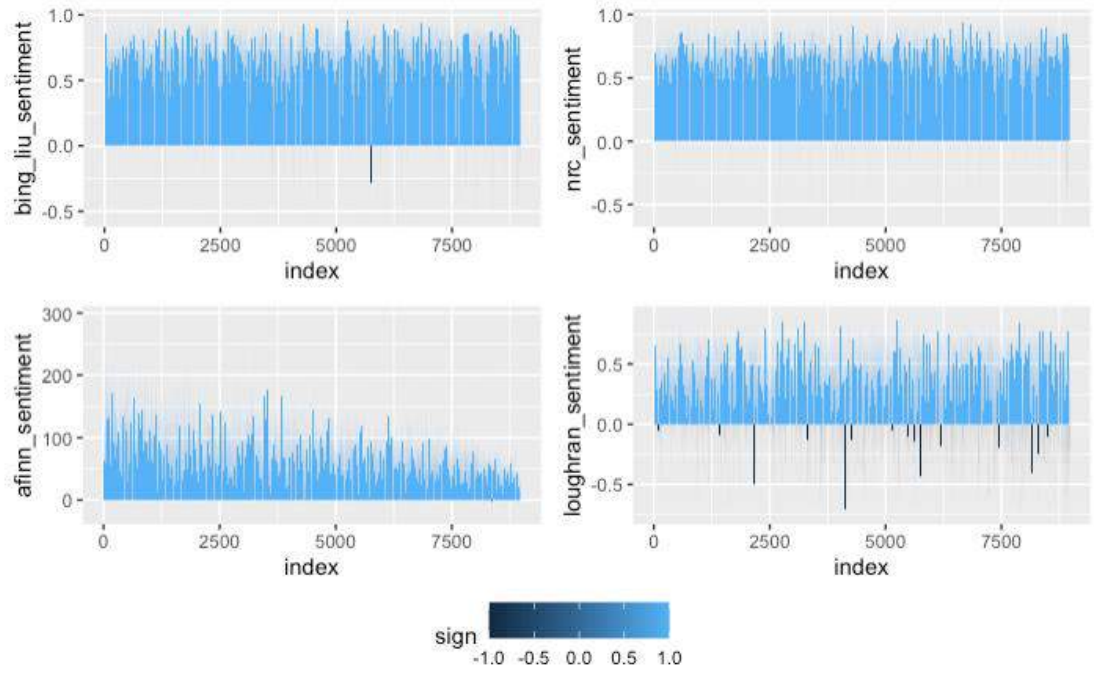
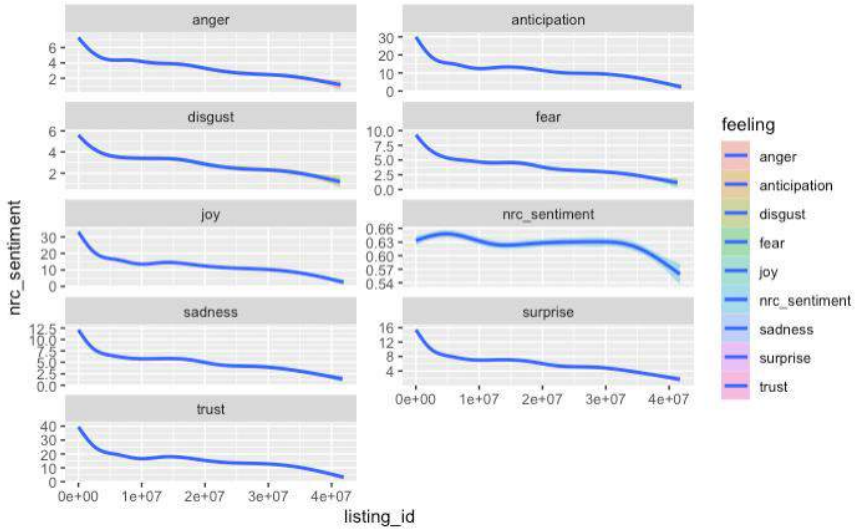


Figure B.7


anger
anticipation
disgust
fear
joy
nrc_sentiment
sadness
surprise
trust
nrc_sentiment
listing_id
0e+00
1e+07
2e+07
3e+07
4e+07
feeling
anger
anticipation
disgust
fear
joy
nrc_sentiment
sadness
surprise
trust

# Part A

2084551

05/06/2021

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE, eval = FALSE)
```

```
rm(list=ls())
library(ggplot2)
library(dplyr)
library(tidytext)
library(jsonlite)
library(tidyr)
library(readr)
library(topicmodels)
library(geojsonR)
library(stringr)
library(dplyr)
library(tidyr)
library(stm)
library(udpipe)
library(cld2)
library(wordcloud)
library(xml2)
library(rvest)
library(janeaustenr)
library(rjson)
library(lubridate)
library(textcat)
library(rJava)
library(qdap)
library(magrittr)
library(jsonlite)
library(topicmodels)
library(geojsonR)
library(ggrepel)
library(gridExtra)
library(widyr)
library(igraph)
library(ggraph)
library(stringr)
library(tm)
set.seed(123113)
```

# Data Preparation

Download the dataset from airbnb.com

```
#crawl data from the website
path <- "http://insideairbnb.com/get-the-data.html"

#table preparation
data_set <- read_html(path)
list_of_tables <- data_set %>% html_nodes("table")

#City index for New York is 63.
city_index <- 63
table <-list_of_tables[[city_index]] %>% html_table()

#Getting the links to download the data
links <- list_of_tables[[city_index]] %>% html_nodes("a") %>% html_attr("href")

# Add it on the dataframe
table$links <- links

# cleaning the column names (lower case, replace _with spaces)
colnames(table) <- gsub(" ","_",tolower(colnames(table)))
colnames(table) <- gsub("/","_",colnames(table))

# Convert the date to a date using the as.Date function.
table$date_compiled <- as.Date(table$date_compiled,format = "%d %b,%Y")
```

```
# Get Review Data
Review_data <- table %>% arrange(desc(date_compiled)) %>%
  filter(grepl("Detailed Review",description)) %>%
  top_n(1)

# Get Listings Data
Listings_data <- table %>% arrange(desc(date_compiled)) %>%
  filter(grepl("Detailed Listings",description)) %>%
  top_n(1)
```

```
#download the files

NYC <- tolower(Listings_data$country_city[1])

download.file(url = Listings_data$links[1],destfile = "listings.csv.gz")

download.file(url = Review_data$links[1], destfile = "reviews.csv.gz")
```

```
listings_detailed <- readr::read_csv("listings.csv.gz")
reviews_detailed <- readr::read_csv("reviews.csv.gz")
```

# Cleaning for Review file

```
# Check the data structure of review file
str(reviews_detailed)
reviews_detailed$date <- as.Date(reviews_detailed$date)

# Filtering out reviews after feb 2020 from reviews_detailed dataset
(records_per_year <- reviews_detailed %>% group_by(lubridate::year(date)) %>% summari
se(count = n()))
reviews_pre_corona <- reviews_detailed %>% filter(date<"2020-02-01")
head(reviews_pre_corona)
```

```
# Step 1
# Cleaning up the language
# For some computers the character set is not
# automatically set to latin or ASCII
library(cld2)
reviews_pre_corona$comments <- iconv(reviews_pre_corona$comments)
reviews_pre_corona$language <- cld2::detect_language(reviews_pre_corona$comments)
reviews_pre_corona%>%
  filter(language =="en") -> reviews_pre_corona
```

```
# Step 2
# Substitute digits with space
reviews_pre_corona$comments <- gsub('[[:digit:]]+',' ', reviews_pre_corona$comments)

# Substitute punctuations with space
reviews_pre_corona$comments <- gsub('[[:punct:]]+',' ', reviews_pre_corona$comments)
```

```
# Step3
# Check the character length of each review
reviews_pre_corona$review_length_chars <- nchar(reviews_pre_corona$comments)
plot <- reviews_pre_corona %>% filter(review_length_chars<2000)

# Plot historgram of review length
hist(plot$review_length_chars,breaks = 100,main = "Review Length(All)")

# Check the characteristics of review length chars
reviews_pre_corona %>% summarise(mean = mean(review_length_chars), median = median(re
view_length_chars),sd = sd(review_length_chars), min = min(review_length_chars), max
 = max(review_length_chars))
```

```
# Filter for minimum length of 100 and 1500 - come back to experiment with this.
reviews_pre_groupby <- reviews_pre_corona %>%
  filter(review_length_chars>100) %>%
  filter(review_length_chars<1500)
# Plot the histogram of review length
hist(reviews_pre_groupby$review_length_chars,breaks = 200,main = "Review Length(All)
 -Left trim to 100 Right trim to 1500")

reviews_pre_groupby %>% summarise(mean = mean(review_length_chars), median = median(r
eview_length_chars),sd = sd(review_length_chars), min = min(review_length_chars), max
= max(review_length_chars))

# # Do a right trim to maximum 1000 chars
# reviews_pre_groupby<- reviews_pre_groupby %>%
#   filter(review_length_chars<1000)
#
# hist(reviews_pre_groupby$review_length_chars,breaks = 200,main = "Review Length(Al
l) -Right trim to 1000")
```

```
# Step 4
# Group by Reviews by listing ids
reviews_groupbylisting <- reviews_pre_groupby %>%
  unnest_tokens(word,comments)%>%
  group_by(listing_id) %>%
  summarise(comments_grouped  = paste(word,collapse = "  "))
```

# Cleaning for listing file

```
# Step 5
# Filtering unwanted columns from listings dataset:
listings_cleaned <- listings_detailed %>% dplyr::select(-c(listing_url, scrape_id, la
st_scraped , picture_url,host_url, host_thumbnail_url,host_picture_url, host_neighbou
rhood, host_listings_count, neighbourhood, bathrooms, minimum_minimum_nights, maximum
_minimum_nights, minimum_maximum_nights, maximum_maximum_nights, minimum_nights_avg_n
tm, maximum_nights_avg_ntm,calendar_updated, calendar_last_scraped, has_availability,
availability_30, availability_60, availability_90, availability_365, calendar_last_sc
raped, first_review, last_review, license))

# Format property type and room type for future analysis
listings_cleaned$property_type <- as.factor(listings_cleaned$property_type)
listings_cleaned$room_type <- as.factor(listings_cleaned$room_type)

# Combine names and descriptions in one column
listings_cleaned$description_combined <- str_c(listings_cleaned$name, ' ', listings_c
leaned$description)
listings_cleaned <- listings_cleaned %>% dplyr::select(-c(name, description))

# Finding number of verifications, amenities, removing $ from price
i <- 1
for (i in 1:length(listings_cleaned$host_verifications)) {

  listings_cleaned$verifications[i] <- length(strsplit(listings_cleaned$host_verifica
tions[i], split= " ")[[1]])
  listings_cleaned$num_amenities[i] <- length(strsplit(listings_cleaned$amenities[i],
split= " ")[[1]])
  listings_cleaned$price[i] <- as.numeric(gsub("\\D", "",listings_cleaned$price[i]))/
100
  listings_cleaned$num_bath[i] <- as.numeric(gsub("[^0-9.]", "",listings_cleaned$bath
rooms_text[i]))
    i <- i+1
}
# Rename the id to listing_ids
colnames(listings_cleaned)[1] <- "listing_id"
```

```
# Step 6
# Cleaning up the language
# For some computers the character set is not
# automatically set to latin or ASCII

listings_cleaned$description_combined <- iconv(listings_cleaned$description_combined,
from = 'UTF-8', to = 'ASCII//TRANSLIT')
listings_cleaned$description_language <- cld2::detect_language(listings_cleaned$descr
iption_combined)

# Filter English descriptions
listings_cleaned <- listings_cleaned %>%
  filter(description_language =="en")
```

```
# Step 7
# Substitute digits with space
listings_cleaned$description_combined <- gsub('[[:digit:]]+',' ', listings_cleaned$de
scription_combined)

# Substitute punctuations with space
listings_cleaned$description_combined<- gsub('[[:punct:]]+',' ', listings_cleaned$des
cription_combined)


listings_cleaned$description_combined<- gsub( '[^[:alnum:]]',' ', listings_cleaned$de
scription_combined)
```

```
# Step 8
# Check the character length of each description
listings_cleaned$listings_length_chars <- nchar(listings_cleaned$description_combine
d)

# Plot the histogram
hist(listings_cleaned$listings_length_chars,breaks = 100,main = "Description Length(A
ll)")

# Check the characteristics of description length chars
listings_cleaned %>% summarise(mean = mean(listings_length_chars), median = median(li
stings_length_chars),sd = sd(listings_length_chars), min = min(listings_length_char
s), max = max(listings_length_chars) )
```

```
# Filter for minimum length of 120, 1040
listings_cleaned <- listings_cleaned %>%
  filter(listings_length_chars>120) %>%
  filter(listings_length_chars<1040)

hist(listings_cleaned$listings_length_chars,breaks = 200,main = "Description Length(A
ll) -Left trim to 120")

# Lets do a right trim to maximum 1000 chars
# listings_cleaned <- listings_cleaned %>%
#   filter(listings_length_chars<1040)
#
# hist(listings_cleaned$listings_length_chars,breaks = 200,main = "Description Length
(All) -Right trim to 1000")
```

```
#Step9
listings_reviews <- listings_cleaned %>% left_join(reviews_groupbylisting,  by = "lis
ting_id")
head(listings_reviews)
```

```
metadata <- listings_reviews
```

# Tokenization

```
# Step 10
# Split the dataset at a size of 1000
split_size <- 1000
tokens_list <- split(listings_reviews,
                     rep(1:ceiling(nrow(listings_reviews)
                                   /split_size),
                         each=split_size,
                         length.out=nrow(listings_reviews)))

# Tokenization for both comments (review) and descriptions
review_tokens_all <- data.frame()
for(i in 1:length(tokens_list)){
   review_tokens_h <- tokens_list[[i]] %>%
    unnest_tokens(word,comments_grouped) %>%
    count(word,listing_id) %>%
    anti_join(stop_words)
   print(i)
   review_tokens_all <- bind_rows(review_tokens_all,review_tokens_h)
}
```

```
description_tokens_all <- data.frame()
for(i in 1:length(tokens_list)){
   description_tokens_h <- tokens_list[[i]] %>%
    unnest_tokens(word,description_combined) %>%
    count(word,listing_id) %>%
    anti_join(stop_words)
   print(i)
   description_tokens_all <- bind_rows(description_tokens_all,description_tokens_h)
}
```

# Review tokens

```
# Step 11
# Calculate the token length

# review_tokens_all$token_length <- nchar(review_tokens_all$clean)
#
# # Lets have a look on the distribution
# review_tokens_all %>% group_by(token_length) %>% summarise(total =n())
```

```
#filter out non english words
review_tokens_all$word <- iconv(review_tokens_all$word,from = 'UTF-8', to = 'ASCII//T
RANSLIT')
review_tokens_all$clean<- gsub('[[:punct:]]+','', review_tokens_all$word)

review_tokens_all$token_length <- nchar(review_tokens_all$clean)

# Lets have a look on the distribution
review_tokens_all %>% group_by(token_length) %>% summarise(total =n())
# review_tokens_all$language <- cld3::detect_language(review_tokens_all$clean)

# # Filter English descriptions
# review_tokens_all <- review_tokens_all %>%
#   filter(language =="en")
```

```
# Remove tokens with length less than 3
review_tokens_all <- review_tokens_all %>%
  filter(token_length>3)

# Lets have a look at the distribution of tokens again
review_tokens_all %>% group_by(token_length) %>%
  summarise(total =n()) %>%
  arrange(desc(token_length))

# Remove those tokens with chars more than 17
review_tokens_all <- review_tokens_all %>%
  filter(token_length<=17)

review_tokens_all$language <- NULL
```

```
# Step 12
# Calculate tf-idf using the listing as a document
review_tokens_all_tf_idf <- review_tokens_all %>%
  bind_tf_idf(clean,listing_id,n)
```

```
# Plot the distribution of tf-idf
hist(review_tokens_all_tf_idf$tf_idf,breaks = 100,main="TF-IDF plot")
```

```
review_tokens_all_tf_idf <- review_tokens_all_tf_idf %>%
  filter(tf_idf<0.25)
hist(review_tokens_all_tf_idf$tf_idf,breaks = 200,main="TF-IDF plot")

# Ok from the plot we see that the cut-off value
# is at 0.05
review_tokens_all_tf_idf <- review_tokens_all_tf_idf %>%
  filter(tf_idf<0.05)

hist(review_tokens_all_tf_idf$tf_idf,breaks = 200,main="TF-IDF plot")

# Remove also very common terms those with tf-idf <0.001 as shown in the chart below
review_tokens_all_tf_idf <- review_tokens_all_tf_idf %>%
  filter(tf_idf>0.001)

review_tokens_all_tf_idf %>% group_by(word) %>%
  summarise(total =n()) %>%
  arrange(desc(total)) %>%
  top_n(40)

# Add manhattan to the stopword dictionary
```

# Description tokens

```
# Step 11
# Calculate the token length
description_tokens_all$word <- iconv(description_tokens_all$word,from = 'UTF-8', to =
'ASCII//TRANSLIT')
description_tokens_all$clean<- gsub('[[:punct:]]+','', description_tokens_all$word)
description_tokens_all$token_length <- nchar(description_tokens_all$clean)

# Look at the distribution
description_tokens_all %>% group_by(token_length) %>% summarise(total =n())
```

```
# Remove tokens with 1 and 2 characters
description_tokens_all <- description_tokens_all %>%
  filter(token_length>3)

# Look at the distribution of tokens again on the oposite side
description_tokens_all %>% group_by(token_length) %>%
  summarise(total =n()) %>%
  arrange(desc(token_length))

# Remove tokens with length greater than 17
description_tokens_all <- description_tokens_all %>%
  filter(token_length<=17)
```

```
# Step 12
# Calculate tf-idf using the listing as a document
description_tokens_all_tf_idf <- description_tokens_all %>%
  bind_tf_idf(clean,listing_id,n)
```

```
# Ok the tfidf is cut
# lets try to filter important words
# using the left and right trim
hist(description_tokens_all_tf_idf$tf_idf,breaks = 100,main="TF-IDF plot")
```

```
# Check the distribution with left trim to 0.5
description_tokens_all_tf_idf <- description_tokens_all_tf_idf %>%
  filter(tf_idf<0.5)

hist(description_tokens_all_tf_idf$tf_idf,breaks = 200,main="TF-IDF plot")

# Remove also very common terms those with tf-idf <0.001 as shown in the chart below

description_tokens_all_tf_idf <- description_tokens_all_tf_idf %>%
  filter(tf_idf>0.001)

hist(description_tokens_all_tf_idf$tf_idf,breaks = 200,main="TF-IDF plot")

description_tokens_all_tf_idf %>% group_by(word) %>%
  summarise(total =n()) %>%
  arrange(desc(total)) %>%
  top_n(40)

# Notice that manhattan could be a stopword and can augment the stopwords
```

```
# Customize stopwords dictionary
tibble(word = c("neighborhood",
                "manhattan",
                "located",
                "brooklyn",
                "queens",
                "staten",
                "island",
                "bronx",
                "stay",
                "staying"),lexicon = rep("custom",10)) %>%
  bind_rows(stop_words) -> custom_dict
```

# Part A -a

##a. dominant word ### Review vs Neighborhood Group * What are the dominant words per aggregation category (neighborhood, access to public transport etc.)?

## Neighborhood group

Let’s first check the dominant word of reviews and descriptions within each neighborhood group
(i.e. Brooklyn, Manhattan, Queens, Staten Island, Bronx)

```
# Get the neighborhood group list
neighborhood_group <- listings_reviews %>%
  select(neighbourhood_group_cleansed) %>%
  unique()

# Check top 10 tokens in each neighbourhood group
neighbourhood_review_tokens_group <- listings_reviews %>%
  select(listing_id,neighbourhood_group_cleansed) %>%
  right_join(review_tokens_all_tf_idf) %>%
  anti_join(custom_dict) %>%
  group_by(neighbourhood_group_cleansed,clean) %>%
  summarise(total=sum(n)) %>%
  arrange(desc(total))

# Establish a df and list
neighbourhood_group_print <- data.frame()
neighbourhood_print <- list()

# Now loop through and get the top 10 tokens
# per neighbourhood group for the review

for(neighb in 1:nrow(neighborhood_group)){
  print(paste0("For neighbourhood group: ",neighborhood_group$neighbourhood_group_cle
ansed[neighb]))

  neighbourhood_print[[neighb]] <- neighbourhood_review_tokens_group %>%
    ungroup() %>%
    filter(neighbourhood_group_cleansed == neighborhood_group$neighbourhood_group_cle
ansed[neighb]) %>%
    top_n(10,total) %>%
    mutate(rank = row_number())
  print(neighbourhood_print[[neighb]])

  neighbourhood_group_print <- rbind(neighbourhood_group_print,neighbourhood_print[[n
eighb]])
}
```

```
# Plot the word frequency
neighbourhood_group_print %>%
  mutate(clean = reorder_within(clean, total, neighbourhood_group_cleansed)) %>%
  ggplot(aes(x= total, y= clean , fill = factor(neighbourhood_group_cleansed))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~neighbourhood_group_cleansed, ncol=2 , scales = "free") +
  labs(x = "Review word frequency among 5 neighbourhood groups", y = NULL) +
  scale_y_reordered()
```

## 5 Neighborhoods among each group

Check the top 5 neighborhoods among each neighborhood group.

```
neighbourhood_group_print_1 <- data.frame()
 neighbourhood_print_1 <- list()
 top_5_neighb_list <- data.frame()
# Get the top 5 neighbourhoods with the most listings in each neighbourhood group

for (group in 1:nrow(neighborhood_group)){
  top_5_neighb <- listings_reviews %>%
    select(listing_id, neighbourhood_cleansed, neighbourhood_group_cleansed) %>%
    unique(.) %>%
    filter(neighbourhood_group_cleansed == neighborhood_group$neighbourhood_group_cle
ansed[group]) %>%
    group_by(neighbourhood_cleansed) %>%
    summarise(total =n())%>%
    arrange(desc(total))%>%
    top_n(5)

  top_5_neighb_list <- rbind(top_5_neighb_list,top_5_neighb)

  tokens_top_5 <- listings_reviews %>%
    select(listing_id,neighbourhood_cleansed) %>%
    filter(neighbourhood_cleansed %in% top_5_neighb$neighbourhood_cleansed) %>%
    unique(.)

  neighbourhood_review_tokens <- review_tokens_all_tf_idf %>%
    right_join(tokens_top_5) %>%
    anti_join(custom_dict) %>%
    group_by(neighbourhood_cleansed,clean) %>%
    summarise(total=sum(n)) %>%
    arrange(desc(total))

  for(neighb in 1:nrow(top_5_neighb)){
    print(paste0("For neighbourhood: ",top_5_neighb$neighbourhood_cleansed[neighb], "
in ",neighborhood_group$neighbourhood_group_cleansed[group]))

    neighbourhood_print_1[[neighb]] <- neighbourhood_review_tokens %>%
      ungroup() %>%
      filter(neighbourhood_cleansed == top_5_neighb$neighbourhood_cleansed[neighb]) %
>%
      top_n(10,total) %>%
      mutate(rank = row_number())

    print(neighbourhood_print_1[[neighb]])

    neighbourhood_group_print_1 <- rbind(neighbourhood_group_print_1,neighbourhood_pr
int_1[[neighb]])
  }

}
```

```
# Order the neighbourhoods based on frequency
top_5_neighb_list <- arrange(top_5_neighb_list,desc(total))
neighbourhood_group_print_1$order <- factor(neighbourhood_group_print_1$neighbourhood
_cleansed, levels = top_5_neighb_list$neighbourhood_cleansed)

# Plot the word frequency
neighbourhood_group_print_1 %>%
  mutate(clean = reorder_within(clean, total, neighbourhood_cleansed)) %>%
  ggplot(aes(x= total, y= clean , fill = factor(neighbourhood_cleansed))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~order, ncol=3 , scales = "free") +
  labs(x = "Review word frequency among top 25 neighbourhoods", y = NULL) +
  scale_y_reordered()
```

# Description vs Neighborhood Group

## Neighborhood group

Check the dominant key words of description in each neiborhood group

```
# Establish a df and list

neighbourhood_group_print_2 <- data.frame()
neighbourhood_print_2 <- list()

# Check tokens
neighbourhood_description_tokens_group <- listings_reviews %>%
  select(listing_id,neighbourhood_group_cleansed) %>%
  right_join(description_tokens_all_tf_idf) %>%
  anti_join(custom_dict) %>%
  group_by(neighbourhood_group_cleansed,clean) %>%
  summarise(total=sum(n)) %>%
  arrange(desc(total))

# Now loop through and get the top 10 tokens
# per neighbourhood group for the review

for(neighb in 1:nrow(neighborhood_group)){
  print(paste0("For neighbourhood group: ",neighborhood_group$neighbourhood_group_cle
ansed[neighb]))

  neighbourhood_print_2[[neighb]] <- neighbourhood_description_tokens_group %>%
    ungroup() %>%
    filter(neighbourhood_group_cleansed == neighborhood_group$neighbourhood_group_cle
ansed[neighb]) %>%
    top_n(10,total) %>%
    mutate(rank = row_number())

  print(neighbourhood_print_2[[neighb]])

  neighbourhood_group_print_2 <- rbind(neighbourhood_group_print_2,neighbourhood_prin
t_2[[neighb]])

}
```

```
# Plot the frequency
neighbourhood_group_print_2 %>%
  mutate(clean = reorder_within(clean, total, neighbourhood_group_cleansed)) %>%
  ggplot(aes(x= total, y= clean , fill = factor(neighbourhood_group_cleansed))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~neighbourhood_group_cleansed, ncol=2 , scales = "free") +
  labs(x = "Review word frequency among 5 neighbourhood groups", y = NULL) +
  scale_y_reordered()
```

## 5 Neighborhoods among each group

```
# Establish a df and list

neighbourhood_group_print_3 <- data.frame()
neighbourhood_print_3 <- list()

# Get the top 5 neighbourhoods with the most listings in each neighbourhood group

for (group in 1:nrow(neighborhood_group)){
  top_5_neighb <- listings_reviews %>%
    select(listing_id, neighbourhood_cleansed, neighbourhood_group_cleansed) %>%
    unique(.) %>%
    filter(neighbourhood_group_cleansed == neighborhood_group$neighbourhood_group_cle
ansed[group]) %>%
    group_by(neighbourhood_cleansed) %>%
    summarise(total =n())%>%
    arrange(desc(total))%>%
    top_n(5)

  tokens_top_5 <- listings_reviews %>%
  select(listing_id,neighbourhood_cleansed) %>%
  filter(neighbourhood_cleansed %in% top_5_neighb$neighbourhood_cleansed) %>%
  unique(.)

  neighbourhood_description_tokens <- description_tokens_all %>%
    right_join(tokens_top_5) %>%
    anti_join(custom_dict) %>%
    group_by(neighbourhood_cleansed,clean) %>%
    summarise(total=sum(n)) %>%
    arrange(desc(total))

  for(neighb in 1:nrow(top_5_neighb)){
    print(paste0("For neighbourhood: ",top_5_neighb$neighbourhood_cleansed[neighb], "
in ",neighborhood_group$neighbourhood_group_cleansed[group], " group"))

    neighbourhood_print_3[[neighb]] <- neighbourhood_description_tokens %>%
      ungroup() %>%
      filter(neighbourhood_cleansed == top_5_neighb$neighbourhood_cleansed[neighb]) %
>%
      top_n(10,total) %>%
      mutate(rank = row_number())
  print(neighbourhood_print_3[[neighb]])

 neighbourhood_group_print_3 <- rbind(neighbourhood_group_print_3,neighbourhood_print
_3[[neighb]])

  }
}
```

```
# Order the neighbourhoods based on frequency
top_5_neighb_list <- arrange(top_5_neighb_list,desc(total))
neighbourhood_group_print_3$order <- factor(neighbourhood_group_print_3$neighbourhood
_cleansed, levels = top_5_neighb_list$neighbourhood_cleansed)

# Plot the frequency
neighbourhood_group_print_3 %>%
  mutate(clean = reorder_within(clean, total, neighbourhood_cleansed)) %>%
  ggplot(aes(x= total, y= clean , fill = factor(neighbourhood_cleansed))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~order, ncol=2 , scales = "free") +
  labs(x = "Description word frequency among top 25 neighbourhoods", y = NULL) +
  scale_y_reordered()
```

# Review vs Room type

```
# Establish a df and list
roomtype_group_print <- data.frame()
roomtype_print <- list()

# Get the roomtype list
room_type_group <- listings_reviews %>%
  select(room_type) %>%
  unique()

# Check tokens
roomtype_review_tokens <- listings_reviews %>%
  select(listing_id,room_type) %>%
  right_join(review_tokens_all) %>%
  anti_join(custom_dict) %>%
  group_by(room_type,clean) %>%
  summarise(total=sum(n)) %>%
  arrange(desc(total))

# Now loop through and get the top 10 tokens
# per room type for the review
for(room in 1:nrow(room_type_group)){
  print(paste0("For room type: ",room_type_group$room_type[room]))
  roomtype_print[[room]] <- roomtype_review_tokens%>%
    ungroup() %>%
    filter(room_type == room_type_group$room_type[room]) %>%
    top_n(10,total) %>%
    mutate(rank = row_number())
  print(roomtype_print[[room]])
  roomtype_group_print <- rbind(roomtype_group_print,roomtype_print[[room]])
}
```

```
# Plot the frequency
roomtype_group_print %>%
  mutate(clean = reorder_within(clean, total, room_type)) %>%
  ggplot(aes(x= total, y= clean , fill = factor(room_type))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~room_type, ncol=2 , scales = "free") +
  labs(x = "Review word frequency among 4 room types", y = NULL) +
  scale_y_reordered()
```

## Description vs Room type

```
# Establish a df and list
roomtype_group_print_1 <- data.frame()
roomtype_print_1 <- list()

# Check tokens
roomtype_description_tokens <- listings_reviews %>%
  select(listing_id,room_type) %>%
  right_join(description_tokens_all) %>%
  anti_join(custom_dict) %>%
  group_by(room_type,clean) %>%
  summarise(total=sum(n)) %>%
  arrange(desc(total))

# Now loop through and get the top 10 tokens
# per room type for the review
for(room in 1:nrow(room_type_group)){
  print(paste0("For room type: ",room_type_group$room_type[room]))
  roomtype_print_1[[room]] <- roomtype_description_tokens%>%
    ungroup() %>%
    filter(room_type == room_type_group$room_type[room]) %>%
    top_n(10,total) %>%
    mutate(rank = row_number())
  print(roomtype_print_1[[room]])
  roomtype_group_print_1 <- rbind(roomtype_group_print_1,roomtype_print_1[[room]])
}
```

```
roomtype_group_print_1 %>%
  mutate(clean = reorder_within(clean, total, room_type)) %>%
  ggplot(aes(x= total, y= clean , fill = factor(room_type))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~room_type, ncol=2 , scales = "free") +
  labs(x = "Description word frequency among 4 room types", y = NULL) +
  scale_y_reordered()
```

# PART A - b

# Handling Inappropriate Data and Mising data for Reviews and Listings Dataset

## Reviews Dataset

```
# Filter Reviews pre-pandemic (filtering out reviews after feb 2020 from reviews_deta
iled dataset)

reviews_detailed$date <- as.Date(reviews_detailed$date)
reviews_pre_corona <- reviews_detailed %>% filter(date<"2020-02-01")
```

```
# Filtering on English comments/reviews as per the scope of the project

reviews_pre_corona$comments <- iconv(reviews_pre_corona$comments)


reviews_pre_corona$language <- cld3::detect_language(reviews_pre_corona$comments)
reviews_pre_corona%>%
  filter(language=="en") -> reviews_pre_corona


paste(dim(reviews_detailed)[1]-dim(reviews_pre_corona)[1], "non-english records remov
ed")
```

```
# Create new feature for Total Words per Review----> "totalwords_comments"

reviews_pre_corona$totalwords_comments <- stringr::str_count(reviews_pre_corona$comme
nts, "\\S+")


# Amazon inside info -> 15 to 250 words required for analysis

reviews_pre_corona %>%
  filter(totalwords_comments>14 & totalwords_comments<251) %>%
  ggplot(., aes(totalwords_comments)) +
  geom_histogram(binwidth = 1)

# # Saving the reviews before group_by listing_id ---> reviews_pre_corona_filter_word
count

reviews_pre_corona %>%
  filter(totalwords_comments>14) %>%
  filter(totalwords_comments < 251) -> reviews_cleaned
```

# Listings Dataset

```
# Select only relevant features from the dataset

listings_cleaned <- listings_detailed %>% select(-c(listing_url,
                                                    scrape_id,
                                                    last_scraped,
                                                    picture_url,
                                                    host_url,
                                                    host_thumbnail_url,
                                                    host_picture_url,
                                                    host_neighbourhood,
                                                    host_listings_count,
                                                    neighbourhood, bathrooms,
                                                    minimum_minimum_nights,
                                                    maximum_minimum_nights,
                                                    minimum_maximum_nights,
                                                    maximum_maximum_nights,
                                                    minimum_nights_avg_ntm,
                                                    maximum_nights_avg_ntm,
                                                    calendar_updated,
                                                    calendar_last_scraped,
                                                    has_availability,
                                                    availability_30,
                                                    availability_60,
                                                    availability_90,
                                                    availability_365,
                                                    calendar_last_scraped,
                                                    first_review,
                                                    last_review,
                                                    license))
```

```
# If the superhost information is missing, we assume that the host is not a "superhos
t"
listings_cleaned$host_is_superhost <- ifelse(listings_cleaned$host_is_superhost == ""
,
                                             "f",
                                             listings_cleaned$host_is_superhost)
listings_cleaned$host_is_superhost <- as.factor(listings_cleaned$host_is_superhost)
```

```
# Property type as factor

listings_cleaned$property_type <- as.factor(listings_cleaned$property_type)
```

```
# Combining Name of property and description as description_combined,
# because the Name describes the property as well

listings_cleaned$description_combined <- str_c(listings_cleaned$name,
                                               ' ',
                                               listings_cleaned$description)
```

```
# Creating a new feature to check how many:
    # 1. amenities are available per listing----------------> "num_amenities"
    # 2. Verification certs are available per listing-------> "num_verifications"
    # 3. Bathrooms are available per listing----------------> "num_bath"

# Cleaning the Price column <- remove the "$" and load as numeric

i <- 1
for (i in 1:length(listings_cleaned$host_verifications)) {

  listings_cleaned$num_verifications[i] <- length(strsplit(listings_cleaned$host_veri
fications[i], split= " ")[[1]])
  listings_cleaned$num_amenities[i] <- length(strsplit(listings_cleaned$amenities[i],
split= " ")[[1]])
  listings_cleaned$price[i] <- as.numeric(gsub("\\D", "",listings_cleaned$price[i]))/
100
  listings_cleaned$num_bath[i] <- as.numeric(gsub("[^0-9.]", "",listings_cleaned$bath
rooms_text[i]))
    i <- i+1
}
```

```
# Filtering on English Descriptions as per the scope of the project

listings_cleaned$description_combined <- iconv(listings_cleaned$description_combined)

listings_cleaned$language <- cld3::detect_language(listings_cleaned$description_combi
ned)
listings_cleaned%>%
  filter(language=="en") -> listings_cleaned


paste(dim(listings_detailed)[1]-dim(listings_cleaned)[1], "non-english records remove
d")
```

```
# Create new feature for Total Words per Review----> "totalwords_description_combine
d"

listings_cleaned$totalwords_description_combined <- stringr::str_count(listings_clean
ed$description_combined, "\\S+")


listings_cleaned %>%
  filter(totalwords_description_combined>0 & totalwords_description_combined<1000) %
>%
  ggplot(., aes(totalwords_description_combined)) +
  geom_histogram(binwidth = 1)
```

```
# No need for filtering by word count

listings_cleaned %>%
  filter(totalwords_description_combined>0) %>%
  filter(totalwords_description_combined < 250) -> listings_cleaned
```

```
# Clean Reviews  -> reviews_cleaned (including stopwords) (language + wordcount)
# Clean Listings -> listings_cleaned (including stopwords) (language + wordcount)
```

```
# Combine reviews per listing_id

reviews_groupbylisting <- reviews_cleaned %>%
  unnest_tokens(word,comments) %>%
  group_by(listing_id) %>%
  summarise(comments_grouped  = paste(word,collapse = "  "))
```

```
# Joining the Listings Dataset and the Reviews Dataset (Left Join)

listings_reviews <- listings_cleaned %>% left_join(reviews_groupbylisting,  by = c("i
d" = "listing_id"))

# Removing Records with no comments

listings_reviews <- listings_reviews[!is.na(listings_reviews$comments_grouped), ]

# Saving listing_reviews as metadeta for back_up

metadata <- listings_reviews
```

```
# 1. word frequency for listing description DONE
#    a. unigrams DONE
#    b. bigrams DONE
#    c. trigrams DONE
#    d. quadgrams DONE

# 2. word frequency for review comments
#    a. unigrams DONE
#    b. bigrams DONE
#    c. trigrams DONE
#    d. quadgrams DONE

# 3. Word association for review comments
#    a. unigrams DONE (reccomend, subway)
#    b. bigrams DONE ()


# 4. Word frequency according to superhost
#    a. for description - should have these words to appear reliable (host) DONE
#    b. for comments - customer reactions for f and t

# 5. Average price for top 10 word frequency
#    a. comments
#    b. description

# 6. Conclusion - What words to include in Description if you are a host for hiring r
ating?
#    a. Word frequency for high ratings DONE
```

```
# write.csv(metadata,"metadata_A_b.csv", row.names = FALSE)

listings_reviews <- read_csv("metadata_A_b.csv")

# 1. word frequency for listing description
#    a. unigrams



# Data frame for Part A b: Save as df

df <- listings_reviews %>% select('id',
                                  'description_combined',
                                  'comments_grouped',
                                  'price',
                                  'review_scores_rating',
                                  'host_is_superhost')
rm(listings_reviews)


df <- df %>% rename(description = description_combined) %>%
  rename(rating = review_scores_rating) %>%
  rename(listing_id = id)


# Check for missing data

colSums(is.na(df))

# Handling Missing data in rating
df[is.na(df$rating), 'rating'] <- mean(na.omit(df$rating))


# Handling Missing data in rating
df[is.na(df$rating), 'rating'] <- mean(na.omit(df$rating))

# Cleaning description column

df$description <- str_replace_all(df$description, "(<br>|<br />|<b>|</b>|<|>|Ã|â|-|
_)", " ") %>%
  iconv(from = 'UTF-8', to = 'ASCII//TRANSLIT')

df$description <- str_replace_all(df$description, "[:digit:]", "")

# df$description <- str_replace_all(df$description, "[[:punct:]]", "")
#
# df$description <- str_replace_all(df$description, "[^[:alnum:]]", " ")


df$comments_grouped <- str_replace_all(df$comments_grouped, "aaahhhmmmaaazzzzing|aaaa
mmmaazziiinngg|a.m.a.z.i.n.g|aaaamaaazing", "amazing")

df$comments_grouped <- str_replace_all(df$comments_grouped, "metro", "subway")

df$comments_grouped <- str_replace_all(df$comments_grouped, "reccomended", "reccomen
d")
```

```
df$comments_grouped <- str_replace_all(df$comments_grouped, "(<br>|<br />|<b>|</b>|<|
>|Ã|â|-|_)", " ") %>%
  iconv(from = 'UTF-8', to = 'ASCII//TRANSLIT')

df$comments_grouped <- str_replace_all(df$comments_grouped, "[:digit:]", "")

# df$comments_grouped <- str_replace_all(df$comments_grouped, "[[:punct:]]", "")
#
# df$comments_grouped <- str_replace_all(df$comments_grouped, "[^[:alnum:]]", " ")

# Custom Dictionary for anti-joining stopwords

remove_words <- remove_words <- c("neighborhood",
                "manhattan",
                "located",
                "brooklyn",
                "queens",
                "staten",
                "island",
                "bronx",
                "stay",
                "staying",
                "york",
                "williamsburg")

tibble(word = remove_words,lexicon = rep("custom",length(remove_words))) %>% bind_row
s(stop_words) -> custom_stopwords
```

```
# Visualising Unigram Frequency for Description

ngram_1_d <- df %>%
  unnest_tokens(word, description, token = "ngrams", n =1) %>%
  anti_join(custom_stopwords)

ngram_1_d_top10 <- ngram_1_d %>%
  group_by(word) %>%
  summarise(Frequency = n(),
            Average_Price = mean(price),
            Average_Rating = mean(rating)) %>%
  top_n(10,Frequency)


df %>%
  unnest_tokens(word, description, token = "ngrams", n =1) %>%
  anti_join(custom_stopwords) %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  ggplot(aes(reorder(word, n), n)) +
  geom_col() +
  labs(title="Top 10 Unigram frequency for Listing Descriptions",
         y = "Frequency",
         x = "Terms") +
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_flip()

ngram_1_d_top10 %>%
  ggplot(aes(Average_Price, Average_Rating)) +
  geom_point(aes(size = Frequency)) +
  geom_text_repel(aes(label=word)) +
  labs(title="Top 10 Unigram frequency for Listing Descriptions",
         y = "Average Rating",
         x = "Average Price") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
# 1. word frequency for listing description
#    b. bigrams

ngram_2_d <- df %>%
  unnest_tokens(word, description, token = "ngrams", n =2)

ngram_2_d$index <- seq.int(nrow(ngram_2_d))

df_temp <- ngram_2_d %>%
  separate(word, c("word1", "word2"), sep = " ") %>%
  select('index', 'word1','word2')

ngram_2_d <- left_join(ngram_2_d, df_temp, by = "index")

ngram_2_d <- ngram_2_d %>%
  filter(!word1 %in% custom_stopwords$word) %>%
  filter(!word2 %in% custom_stopwords$word)

ngram_2_d_top10 <- ngram_2_d %>%
  group_by(word) %>%
  summarise(Frequency = n(),
            Average_Price = mean(price),
            Average_Rating = mean(rating)) %>%
  top_n(10,Frequency)

ngram_2_d %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  ggplot(aes(reorder(word, n), n)) +
  geom_col() +
  labs(title="Top 10 Biigram Frequency for Listing Descriptions",
         y = "Frequency",
         x = "Terms") +
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_flip()


ngram_2_d_top10 %>%
  ggplot(aes(Average_Price, Average_Rating)) +
  geom_point(aes(size = Frequency)) +
  geom_text_repel(aes(label=word)) +
  labs(title="Top 10 Biigram Frequency for Listing Descriptions",
         y = "Average Rating",
         x = "Average Price") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
# 1. word frequency for listing description
#    c. trigrams

ngram_3_d <- df %>%
  unnest_tokens(word, description, token = "ngrams", n =3)

ngram_3_d$index <- seq.int(nrow(ngram_3_d))

df_temp <- ngram_3_d %>%
  separate(word, c("word1", "word2", "word3"), sep = " ") %>%
  select('index', 'word1','word2', 'word3')

ngram_3_d <- left_join(ngram_3_d, df_temp, by = "index")

rm(df_temp)

ngram_3_d <- ngram_3_d %>%
  filter(!word1 %in% custom_stopwords$word) %>%
  filter(!word3 %in% custom_stopwords$word)

ngram_3_d_top10 <- ngram_3_d %>%
  group_by(word) %>%
  summarise(Frequency = n(),
            Average_Price = mean(price),
            Average_Rating = mean(rating)) %>%
  top_n(10,Frequency)


ngram_3_d %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  ggplot(aes(reorder(word, n), n)) +
  geom_col() +
  labs(title="Top 10 Trigram Frequency for Listing Descriptions",
         y = "Frequency",
         x = "Terms") +
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_flip()

ngram_3_d_top10 %>%
  ggplot(aes(Average_Price, Average_Rating)) +
  geom_point(aes(size = Frequency)) +
  geom_text_repel(aes(label=word)) +
  labs(title="Top 10 Trigram Frequency for Listing Descriptions",
         y = "Average Rating",
         x = "Average Price") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
# 1. word frequency for listing description
#    c. ngram = 4

ngram_4_d <- df %>%
  unnest_tokens(word, description, token = "ngrams", n =4)

ngram_4_d$index <- seq.int(nrow(ngram_4_d))

df_temp <- ngram_4_d %>%
  separate(word, c("word1", "word2", "word3", "word4"), sep = " ") %>%
  select('index', "word1","word2", "word3","word4")

ngram_4_d <- left_join(ngram_4_d, df_temp, by = "index")

rm(df_temp)

ngram_4_d <- ngram_4_d %>%
  filter(!word1 %in% custom_stopwords$word) %>%
  filter(!word4 %in% custom_stopwords$word)

ngram_4_d_top10 <- ngram_4_d %>%
  group_by(word) %>%
  summarise(Frequency = n(),
            Average_Price = mean(price),
            Average_Rating = mean(rating)) %>%
  top_n(10,Frequency)


ngram_4_d %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  ggplot(aes(reorder(word, n), n)) +
  geom_col() +
  labs(title="Top 10 Quadgram Frequency for Listing Descriptions",
         y = "Frequency",
         x = "Terms") +
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_flip()

ngram_4_d_top10 %>%
  ggplot(aes(Average_Price, Average_Rating)) +
  geom_point(aes(size = Frequency)) +
  geom_text_repel(aes(label=word)) +
  labs(title="Top 10 Quadgram Frequency for Listing Descriptions",
         y = "Average Rating",
         x = "Average Price") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
# 4. Word frequency according to superhost
#    a. for description - should have these words to appear reliable (host) DONE


super <- df %>% group_by(host_is_superhost) %>% count()
f_count <- as.numeric(super$n[1])
t_count <- as.numeric(super$n[2])

a <- ngram_2_d %>%
  filter(host_is_superhost == TRUE) %>%
  group_by(word) %>%
  summarise(Frequency = (n()/t_count), Is_SuperHost = TRUE) %>%
  top_n(20,Frequency)

b <- ngram_2_d %>% filter(host_is_superhost == FALSE) %>%
  group_by(word) %>%
  summarise(Frequency = (n()/f_count), Is_SuperHost = FALSE) %>%
  top_n(20,Frequency)

ngram_4_d_super <- rbind(a, b)

rm(list = c(a,b))

ngram_4_d_super %>% ggplot(aes(x=reorder(word, Frequency), y=Frequency, fill=Is_Super
Host)) +
  geom_bar(stat="identity", position="dodge") +
  theme(axis.text.x = element_text(angle = 90)) +
  labs(title="Top 20 Term Frequency for Listing Descriptions",
         y = "Relative Frequency",
         x = "Terms") +
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank()) +
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_flip()
```

```
# 2. word frequency for review comments
#    a. unigrams

# Visualising Unigram Frequency

ngram_1_c <- df %>%
  unnest_tokens(word, comments_grouped, token = "ngrams", n =1) %>%
  anti_join(custom_stopwords)%>%
  filter(str_detect(word, "[:alpha:]"))

ngram_1_c_top10 <- ngram_1_c %>%
  group_by(word) %>%
  summarise(Frequency = n(),
            Average_Price = mean(price),
            Average_Rating = mean(rating)) %>%
  top_n(10,Frequency)
```

```
ngram_1_c %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  ggplot(aes(reorder(word, n), n)) +
  geom_col() +
  labs(title="Top 10 Unigrams Frequency for Comments",
         y = "Frequency",
         x = "Terms") +
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_flip()

ngram_1_c_top10 %>%
  ggplot(aes(Average_Price, Average_Rating)) +
  geom_point(aes(size = Frequency)) +
  geom_text_repel(aes(label=word)) +
  labs(title="Top 10 Unigrams Frequency for Comments",
         y = "Average Rating",
         x = "Average Price") +
  theme(plot.title = element_text(hjust = 0.5))
```

```r
# 2. word frequency for review comments
#    b. bigrams

# Visualizing Bigrams Frequency

ngram_2_c <- df %>%
  unnest_tokens(word, comments_grouped, token = "ngrams", n =2)%>%
  filter(str_detect(word, "[:alpha:]"))

ngram_2_c$index <- seq.int(nrow(ngram_2_c))

df_temp <- ngram_2_c %>%
  separate(word, c("word1", "word2"), sep = " ") %>%
  select('index', 'word1','word2')

ngram_2_c <- left_join(ngram_2_c, df_temp, by = "index")

rm(df_temp)

ngram_2_c <- ngram_2_c %>%
  filter(!word1 %in% custom_stopwords$word) %>%
  filter(!word2 %in% custom_stopwords$word)

ngram_2_c_top10 <- ngram_2_c %>%
  group_by(word) %>%
  summarise(Frequency = n(),
            Average_Price = mean(price),
            Average_Rating = mean(rating)) %>%
  top_n(10,Frequency)

ngram_2_c %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  ggplot(aes(reorder(word, n), n)) +
  geom_col() +
  labs(title="Top 10 Biigram Frequency for Comments",
         y = "Frequency",
         x = "Terms") +
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_flip()


ngram_2_c_top10 %>%
  ggplot(aes(Average_Price, Average_Rating)) +
  geom_point(aes(size = Frequency)) +
  geom_text_repel(aes(label=word)) +
  labs(title="Top 10 Biigram Frequency for Comments",
         y = "Average Rating",
         x = "Average Price") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
# 2. word frequency for comments
#    c. trigrams

ngram_3_c <- df %>%
  unnest_tokens(word, comments_grouped, token = "ngrams", n =3)%>%
  filter(str_detect(word, "[:alpha:]"))

ngram_3_c$index <- seq.int(nrow(ngram_3_c))

df_temp <- ngram_3_c %>%
  separate(word, c("word1", "word2", "word3"), sep = " ") %>%
  select('index', 'word1','word2', 'word3')

ngram_3_c <- left_join(ngram_3_c, df_temp, by = "index")

rm(df_temp)

ngram_3_c <- ngram_3_c %>%
  filter(!word1 %in% custom_stopwords$word) %>%
  filter(!word3 %in% custom_stopwords$word)

ngram_3_c_top10 <- ngram_3_c %>%
  group_by(word) %>%
  summarise(Frequency = n(),
            Average_Price = mean(price),
            Average_Rating = mean(rating)) %>%
  top_n(10,Frequency)


ngram_3_c %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  ggplot(aes(reorder(word, n), n)) +
  geom_col() +
  labs(title="Top 10 Trigram Frequency for Comments",
         y = "Frequency",
         x = "Terms") +
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_flip()

ngram_3_c_top10 %>%
  ggplot(aes(Average_Price, Average_Rating)) +
  geom_point(aes(size = Frequency)) +
  geom_text_repel(aes(label=word)) +
  labs(title="Top 10 Trigram Frequency for Comments",
         y = "Average Rating",
         x = "Average Price") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
# 2. word frequency for comments
#    d. ngram = 4

ngram_4_c <- df %>%
  unnest_tokens(word, comments_grouped, token = "ngrams", n =4) %>%
  filter(str_detect(word, "[:alpha:]"))

ngram_4_c$index <- seq.int(nrow(ngram_4_c))

df_temp <- ngram_4_c %>%
  separate(word, c("word1", "word2", "word3", "word4"), sep = " ") %>%
  select('index', "word1","word2", "word3","word4")

ngram_4_c <- left_join(ngram_4_c, df_temp, by = "index")

rm(df_temp)

ngram_4_c <- ngram_4_c %>%
  filter(!word1 %in% custom_stopwords$word) %>%
  filter(!word4 %in% custom_stopwords$word)

ngram_4_c_top10 <- ngram_4_c %>%
  group_by(word) %>%
  summarise(Frequency = n(),
            Average_Price = mean(price),
            Average_Rating = mean(rating)) %>%
  top_n(10,Frequency)


ngram_4_c %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  ggplot(aes(reorder(word, n), n)) +
  geom_col() +
  labs(title="Top 10 Quadgram Frequency for Comments",
         y = "Frequency",
         x = "Terms") +
  theme(plot.title = element_text(hjust = 0.5)) +
  coord_flip()

ngram_4_c_top10 %>%
  ggplot(aes(Average_Price, Average_Rating)) +
  geom_point(aes(size = Frequency)) +
  geom_text_repel(aes(label=word)) +
  labs(title="Top 10 Quadgram Frequency for Comments",
         y = "Average Rating",
         x = "Average Price") +
  theme(plot.title = element_text(hjust = 0.5))
```

# Word Associations

```
# Document Term Matrix needed to find word correlations

dtm_1_c <- df %>%
  unnest_tokens(word, comments_grouped, token = "ngrams", n =1) %>%
  anti_join(custom_stopwords) %>%
  count(listing_id, word) %>%
  cast_dtm(listing_id, word, n)

inspect(dtm_1_c)
```

```
# Note DTM for n = 4, is not very populated

dtm_4_c <- ngram_4_c %>%
  count(listing_id, word) %>%
  cast_dtm(listing_id, word, n)

inspect(dtm_4_c)
```

```
# we can check which words correlate with "recommend"

findAssocs(dtm_1_c,"recommend",corlimit = 0.7)
```

```
# To check and confirm what word combinations have reccomend to get an idea about thi
ngs which are reccomended in the comments by the user
ngram_3_c %>% group_by(word) %>%
  count(sort = TRUE) %>%
  filter(stringr::str_detect(word,"recommend | recommended"))
```

```
# Now lets visualise all the high frequency terms and their respective correlated
# words so that we can get an overall idea about the things discussed in the comment
s.
# This is a crude version of checking which topics are explicitly discussed within th
e comments,
# But we need Topic Modeling to check Latent topics.

listings_which_mention_word <- ngram_1_c %>%
  filter(str_detect(word, "[:alpha:]")) %>%
  count(word, name = "freq_per_listing", sort = TRUE) %>%
  filter(freq_per_listing >= 2000)


word_corr <- ngram_1_c %>%
  semi_join(listings_which_mention_word, by = "word") %>%
  pairwise_cor(item = word, feature = listing_id) %>%
  filter(correlation >= 0.5)


graph_from_data_frame(d = word_corr,
                      vertices = listings_which_mention_word %>%
                        semi_join(word_corr, by = c("word" = "item1"))) %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(alpha = correlation)) +
  geom_node_point() +
  geom_node_text(aes(color = freq_per_listing, label = name), repel = TRUE) +
  labs(title="Unigram Correlations for Comments") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
# Similarly we can check the same using bigrams to see if we get any more insights.

listings_which_mention_word_2 <- ngram_2_c %>%
  count(word, name = "freq_per_listing", sort = TRUE) %>%
  filter(freq_per_listing >= 100)

word_corr_2 <- ngram_2_c %>%
  semi_join(listings_which_mention_word_2, by = "word") %>%
  pairwise_cor(item = word, feature = listing_id) %>%
  filter(correlation >= 0.5)


graph_from_data_frame(d = word_corr_2,
                      vertices = listings_which_mention_word_2 %>%
                        semi_join(word_corr_2, by = c("word" = "item1"))) %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(alpha = correlation)) +
  geom_node_point() +
  geom_node_text(aes(color = freq_per_listing, label = name), repel = TRUE) +
  labs(title="Bigram Correlations for Comments") +
  theme(plot.title = element_text(hjust = 0.5))

# Here we can see several clusters of discussions within the comments and mostly
# customers have discussed either the things provided by the host or locations
# around the listing. We can confirm this in the Topic Modeling section (Part C)
```

##Part A - c

```r
## check the distribution of the rating scores
hist(listings_cleaned$review_scores_rating,
     breaks = 50,
     main = "Rating score for listings",
     xlab = "Score",
     ylbs = "Listing count")

# check the amount of observations those rating scores are lower than 80
# or equal to 100
# or NA
listings_cleaned %>% filter(review_scores_rating<80) %>% nrow()
listings_cleaned %>% filter(review_scores_rating==100) %>% nrow()
sum(is.na(listings_cleaned$review_scores_rating))

# filter [80,100)
listing_Ac <- listings_cleaned %>%
  filter(review_scores_rating >= 80) %>%
  filter(review_scores_rating < 100)
nrow(listing_Ac)/nrow(listings_cleaned)

# plot again
hist(listing_Ac$review_scores_rating,
     breaks = 50,
     main = "Rating score for listings after filtering",
     xlab = "Score",
     ylbs = "Listing count")

# join the review data with listing data by listing id
data_Ac <- reviews_pre_groupby %>%
  inner_join(listing_Ac) %>%
  mutate(review_id = row_number())
data_Ac$comments <- tolower(data_Ac$comments)
```

```
# clean the host name column
data_Ac$host_name <- tolower(data_Ac$host_name)
# Substitute digits with space
data_Ac$host_name <- gsub(pattern = '[[:digit:]]+',
                          replace = ' ',
                          data_Ac$host_name)

# Substitute punctuations with space
data_Ac$host_name <- gsub(pattern = '[[:punct:]]+',
                          replace = ' ',
                          data_Ac$host_name)

# Substitute special words with space
special_words <- paste(c("new york",
                         "city",
                         "NY",
                         "hotel",
                         "apartment",
                         "hostel",
                         "flat",
                         "garden",
                         "building",
                         "center",
                         "centre"),
                       collapse = "|")
data_Ac$host_name <- gsub(pattern = special_words,
                          replace = ' ',
                          data_Ac$host_name,
                          ignore.case = TRUE)
# cleaned the rows with missing values in host_name
data_Ac <- data_Ac[!is.na(data_Ac$host_name),]
```

```
# add a column for mentioning the host name or not
data_Ac <- data_Ac %>%
  mutate(mention_host_name = stringr::str_detect(data_Ac$comments,
                                                 data_Ac$host_name))
data_Ac <- data_Ac %>%
  group_by(listing_id) %>%
  mutate(host_name_mention_prop=sum(mention_host_name)/n()) %>%
  ungroup()

# plot
data_Ac %>% ggplot(.,aes(x = host_name_mention_prop,
                         y = review_scores_rating))+
  geom_point(alpha=0.2) +
  geom_smooth(method="lm") +
  ggtitle("Rating scores vs mention host name or not")
# cor test
cor.test(data_Ac$review_scores_rating,
         data_Ac$host_name_mention_prop,
         method = "pearson")
```

```
# add a column for times of mentioning host name in comments
data_Ac <- data_Ac %>%
  mutate(mention_host_name_count = stringr::str_detect(data_Ac$comments,
                                                       data_Ac$host_name))
# plot
data_Ac <- data_Ac %>%
  group_by(listing_id) %>%
  mutate(mention_host_name_total = sum(mention_host_name_count)) %>%
  ungroup()

data_Ac %>%
  ggplot(.,aes(x = mention_host_name_total, y = review_scores_rating)) +
    geom_point(alpha=0.2) +
    geom_smooth(method = "lm") +
  ggtitle("Rating scores vs times to mention host name")

# cor test
cor.test(data_Ac$review_scores_rating,
         data_Ac$mention_host_name_total,
         method = "pearson")
```

```
data_Ac <- data_Ac %>%
  mutate(review_length = str_count(comments)) %>%
  group_by(listing_id) %>%
  mutate(average_review_length = mean(review_length)) %>%
  ungroup()

# plot
data_Ac %>% ggplot(aes(x=average_review_length,
                       y=review_scores_rating)) +
  geom_point(alpha=0.2) +
  geom_smooth(method="lm") +
  ggtitle("Rating scores vs length of reviews")

# cor test
cor.test(data_Ac$review_scores_rating,
         data_Ac$average_review_length,
         method="pearson")
```

```
data_Ac <- data_Ac %>%
  mutate(recommend_mentioned = stringr::str_detect(data_Ac$comments,
                                                   "recommend|recommendation")) %>%
  group_by(listing_id) %>%
  mutate(recommed_mentioned_prop = sum(recommend_mentioned)/n()) %>%
  ungroup()

# plot
data_Ac %>%
  ggplot(.,aes(x = recommed_mentioned_prop, y = review_scores_rating)) +
  geom_point(alpha=0.2) +
  geom_smooth(method = "lm") +
  ggtitle("Rating scores vs recommend")

# cor test
cor.test(data_Ac$review_scores_rating,
         data_Ac$recommed_mentioned_prop,
         method="pearson")
```

```
library(tm)
# plot the amount of the amenities and the scores rating
data_Ac %>%
  ggplot(.,aes(x = num_amenities, y = review_scores_rating)) +
  geom_point(alpha=0.2) +
  geom_smooth(method = "lm")  +
  ggtitle("Rating scores vs availability of amenties")

# cor test
cor.test(data_Ac$review_scores_rating,
         data_Ac$num_amenities,
         method="pearson")
```

```
# abstract a list of attractions from the tripadvisor
ny_places = data.frame()
page
for (page_url in seq(0,91,30)) {
  url <- paste0("https://www.tripadvisor.co.uk/Attractions-g60763-Activities-oa",page
_url , "-New_York_City_New_York.html")
  page = read_html(url)

  page <- page %>% html_nodes("._1zP41Z7X") %>% html_text()

  ny_places <- rbind(ny_places, data.frame(page))

}

ny_places <- head(ny_places,99)

i <- 1
for (ele in ny_places$page) {
  if (i<10){
    ny_places$page[i] <-  substr(ny_places$page[i],4,nchar(ny_places$page[i]))
    i <- i+1
  }
  else {
    ny_places$page[i] <- substr(ny_places$page[i],5,nchar(ny_places$page[i]))
    i <- i+1
  }
}

# combine the name of the attractions together and tokenize it
tokenized_attraction <- ny_places %>% unnest_tokens(word, page) %>% count(word, sort=
TRUE) %>% view(.)
# choose most common words for attraction spots
attraction_stop_words <- tibble(word = c("the", "of", "at", "on", "street", "st", "av
enue", "road","new","east","west","north","south"), lexicon = "mine")
attraction_word <- tokenized_attraction %>%
  filter(n>2) %>%
  anti_join(attraction_stop_words) %>%
  bind_rows(filter(tokenized_attraction, word==c("broadway","church","sea","bridge"
)))

# use the tokenized data
data_Ac_attraction_detect <- review_tokens_all_tf_idf %>%
  mutate(attraction_detect = review_tokens_all_tf_idf$word %in% attraction_word$word)
%>%
  group_by(listing_id) %>%
  mutate(attraction_word_count=sum(attraction_detect)) %>%
  left_join(select(listing_Ac,
                   c(listing_id,
                     review_scores_rating)))
# plot
data_Ac_attraction_detect %>%
  select(listing_id,attraction_word_count,review_scores_rating) %>%
  group_by(listing_id) %>%
  mutate(count=n()) %>%
  ggplot(.,aes(x = attraction_word_count, y = review_scores_rating, size=count)) +
  geom_point(alpha=0.2) +
  geom_smooth(method = "lm") +
```

```
  ggtitle("Rating scores vs mentioning attraction points")

# cor test
cor.test(data_Ac_attraction_detect$review_scores_rating,
         data_Ac_attraction_detect$attraction_word_count,
         method="pearson")
```

```
library(qdap)
# calculate the readability of each review
comment_readability <- qdap::flesch_kincaid(gsub("[^ -~]", "", data_Ac$comments),
                                            data_Ac$review_id)

to_join <- comment_readability$Readability %>%
  select(review_id, FK_grd.lvl)
comment_readability_check <- data_Ac %>% left_join(to_join)

# plot the relation between the readability of the reviews and the rating scores
comment_readability_check %>%
  select(FK_grd.lvl, review_scores_rating, listing_id) %>%
  group_by(listing_id) %>%
  mutate(average_readability = sum(FK_grd.lvl)/n()) %>%
  na.omit() %>%
  ggplot(aes(x=average_readability, y=review_scores_rating)) +
  geom_point() +
  geom_smooth(method = "lm") +
  ggtitle("Rating scores vs Readability of comments")

cor.test(comment_readability_check$review_scores_rating,
         comment_readability_check$FK_grd.lvl,
         method="pearson")
```

#Part A - d - Using the textual description of the property supplied by the owner, how does this relate with the price that the property is listed for rent?

```
listings_reviews <- listings_reviews %>% rename("id" = "listing_id")

#Pre-work to get some text to work with from listing description
listings_reviews = listings_reviews %>% mutate(document_id = row_number())
listings_reviews$description_combined <- iconv(listings_reviews$description_combined)

listings_reviews$language = cld3::detect_language(listings_reviews$description_combin
ed)
listings_reviews = listings_reviews %>% filter(language == "en")
```

```
#Check Price Distribution

listings_reviews$price = as.numeric(listings_reviews$price)

listings_reviews %>% dplyr::select(id,price) %>%
  ggplot(aes(price))+geom_histogram()



#No need to trim, as Tianhui gets rid of some descriptions in {r cleanup length} Step
8. Use my code for the cleaned dataset with description column (code can be adjusted
 for the dataset)
```

```
#Readability calculation
listings_read = qdap::flesch_kincaid(gsub("[^ -~]", "", listings_reviews$description_
combined),listings_reviews$id)
```

```
#Adding readability score as a column
listings_read$Readability %>% dplyr::select(id,FK_grd.lvl ) -> join_table

listings_reviews %>% left_join(join_table, by = "id") -> listings_reviews


#FK_grd.lvl distributions

listings_reviews %>% dplyr::select(id, FK_grd.lvl) %>% ggplot(aes(FK_grd.lvl))+geom_h
istogram()

#Plotting Readability against price
listings_reviews %>%
  dplyr::select(FK_grd.lvl,price) %>%
  na.omit() %>%
  filter(FK_grd.lvl>50) %>%
  ggplot(aes(x=FK_grd.lvl,y=log(price)))  + geom_smooth(method="lm") + geom_point(siz
e = 2, shape=1,alpha=0.1)
```

```
#Adding Formality column

formality <- qdap::formality(gsub("[^ -~]", "", listings_reviews$description_combine
d),listings_reviews$id)
```

```
#Adding Formality column
formality$formality %>% dplyr::select(id,formality ) -> formality_join
formality_join$id = as.integer(formality_join$id)

listings_reviews %>% left_join(formality_join, by = "id") -> listings_reviews

#Plotting Formality against price
listings_reviews %>%
  dplyr::select(formality,price) %>%
  na.omit() %>%
  ggplot(aes(x=formality,y=log(price)))  + geom_smooth(method="lm") + geom_point(size
= 2, shape=1,alpha=0.1)
```

```
#Polarity calculations
polarity <- qdap::polarity(gsub("[^ -~]", "", listings_reviews$description_combined),
listings_reviews$id)
```

```
#Adding Polarity column

polarity$all %>% dplyr::select(id,polarity ) -> polarity_join

listings_reviews %>% left_join(polarity_join, by = "id") -> listings_reviews

#Plotting Polarity against price

listings_reviews %>%
  dplyr::select(polarity,price) %>%
  na.omit() %>%
  ggplot(aes(x=polarity,y=log(price)))  + geom_smooth(method="lm") + geom_point(size
 = 2, shape=1,alpha=0.1)
```

```
#Adding word count column
listings_reviews$wordcount = as.numeric(qdap::wc(listings_reviews$description_combine
d, byrow = TRUE))

#Plotting Word Count against price

listings_reviews %>%
  dplyr::select(wordcount,price) %>%
  na.omit() %>%
  ggplot(aes(x=wordcount,y=log(price)))  + geom_smooth(method="lm") + geom_point(size
= 2, shape=1,alpha=0.1)
```

```
#Models

model1 <- lm(log(listings_reviews$price)~listings_reviews$wordcount)
model2 <- lm(log(listings_reviews$price)~listings_reviews$formality)
model3 <- lm(log(listings_reviews$price)~listings_reviews$FK_grd.lvl)
model4 <- lm(log(listings_reviews$price)~listings_reviews$polarity)

stargazer::stargazer(model1,model2,model3,model4,type = "text")
```

```
summary(listings_reviews$price)

#Split the price

price_low = listings_reviews %>% dplyr::select(id,price, wordcount, formality, FK_gr
d.lvl, polarity) %>% filter(price <= 100)

price_med = listings_reviews %>% dplyr::select(id,price, wordcount, formality, FK_gr
d.lvl, polarity) %>% filter(price > 100 & price <= 300)

price_high = listings_reviews %>% dplyr::select(id,price, wordcount, formality, FK_gr
d.lvl, polarity) %>% filter(price > 300)


# 0-100, 100-300, 300-above
```

```
#Plot wordcount for different prices

pl_wc = price_low %>%
  dplyr::select(wordcount,price) %>%
  na.omit() %>%
  ggplot(aes(x=wordcount,y=log(price)))  + geom_smooth(method="lm") + geom_point(size
= 2, shape=1,alpha=0.1) + ylab("Price Low")

pm_wc = price_med %>%
  dplyr::select(wordcount,price) %>%
  na.omit() %>%
  ggplot(aes(x=wordcount,y=log(price)))  + geom_smooth(method="lm") + geom_point(size
= 2, shape=1,alpha=0.1)+  ylab("Price Medium")

ph_wc = price_high %>%
  dplyr::select(wordcount,price) %>%
  na.omit() %>%
  ggplot(aes(x=wordcount,y=log(price)))  + geom_smooth(method="lm") + geom_point(size
= 2, shape=1,alpha=0.1) + ylab("Price High")

library(gridExtra)

grid.arrange(pl_wc, pm_wc, ph_wc,
          ncol = 2, nrow = 2)

#Check price distribution for all groups

price_low %>% dplyr::select(id,price) %>%
  ggplot(aes(price))+geom_histogram()

price_med %>% dplyr::select(id,price) %>%
  ggplot(aes(price))+geom_histogram()

price_high %>% dplyr::select(id,price) %>%
  ggplot(aes(price))+geom_histogram()



#Run separate models
model5 <- lm(price_low$price~price_low$wordcount)
model6 <- lm(price_med$price~price_med$wordcount)
model7 <- lm(log(price_high$price)~price_high$wordcount)


stargazer::stargazer(model5,model6,model7,type = "text")
```

```
#Plot formality for different prices

pl_form = price_low %>%
  dplyr::select(formality,price) %>%
  na.omit() %>%
  ggplot(aes(x=formality,y=log(price)))  + geom_smooth(method="lm") + geom_point(size
= 2, shape=1,alpha=0.1) + ylab("Price Low")

pm_form = price_med %>%
  dplyr::select(formality,price) %>%
  na.omit() %>%
  ggplot(aes(x=formality,y=log(price)))  + geom_smooth(method="lm") + geom_point(size
= 2, shape=1,alpha=0.1) + ylab("Price Medium")

ph_form = price_high %>%
  dplyr::select(formality,price) %>%
  na.omit() %>%
  ggplot(aes(x=formality,y=log(price)))  + geom_smooth(method="lm") + geom_point(size
= 2, shape=1,alpha=0.1) + ylab("Price High")


grid.arrange(pl_form, pm_form, ph_form,
          ncol = 2, nrow = 2)



#Run separate models
model8 <- lm(price_low$price~price_low$formality)
model9 <- lm(price_med$price~price_med$formality)
model10 <- lm(log(price_high$price)~price_high$formality)


stargazer::stargazer(model8,model9,model10,type = "text")
```

```
#Plot readability for different prices

pl_read = price_low %>%
  dplyr::select(FK_grd.lvl,price) %>%
  na.omit() %>%
  ggplot(aes(x=FK_grd.lvl,y=log(price)))  + geom_smooth(method="lm") + geom_point(siz
e = 2, shape=1,alpha=0.1) + ylab("Price Low")

pm_read = price_med %>%
  dplyr::select(FK_grd.lvl,price) %>%
  na.omit() %>%
  ggplot(aes(x=FK_grd.lvl,y=log(price)))  + geom_smooth(method="lm") + geom_point(siz
e = 2, shape=1,alpha=0.1) + ylab("Price Medium")

ph_read = price_high %>%
  dplyr::select(FK_grd.lvl,price) %>%
  na.omit() %>%
  ggplot(aes(x=FK_grd.lvl,y=log(price)))  + geom_smooth(method="lm") + geom_point(siz
e = 2, shape=1,alpha=0.1) + ylab("Price High")

library(gridExtra)

grid.arrange(pl_read, pm_read, ph_read,
          ncol = 2, nrow = 2)



#Run separate models
model11 <- lm(price_low$price~price_low$FK_grd.lvl)
model12 <- lm(price_med$price~price_med$FK_grd.lvl)
model13 <- lm(log(price_high$price)~price_high$FK_grd.lvl)


stargazer::stargazer(model11,model12,model13,type = "text")
```

```
#Plot polarity for different prices

pl_pol = price_low %>%
  dplyr::select(polarity,price) %>%
  na.omit() %>%
  ggplot(aes(x=polarity,y=log(price)))  + geom_smooth(method="lm") + geom_point(size
 = 2, shape=1,alpha=0.1) + ylab("Price Low")

pm_pol = price_med %>%
  dplyr::select(polarity,price) %>%
  na.omit() %>%
  ggplot(aes(x=polarity,y=log(price)))  + geom_smooth(method="lm") + geom_point(size
 = 2, shape=1,alpha=0.1) + ylab("Price Medium")

ph_pol = price_high %>%
  dplyr::select(polarity,price) %>%
  na.omit() %>%
  ggplot(aes(x=polarity,y=log(price)))  + geom_smooth(method="lm") + geom_point(size
 = 2, shape=1,alpha=0.1) + ylab("Price High")

library(gridExtra)

grid.arrange(pl_pol, pm_pol, ph_pol,
          ncol = 2, nrow = 2)



#Run separate models
model14 <- lm(price_low$price~price_low$polarity)
model15 <- lm(price_med$price~price_med$polarity)
model16 <- lm(log(price_high$price)~price_high$polarity)


stargazer::stargazer(model14,model15,model16,type = "text")
```

```
#Full multiple regression

model17 <- lm(log(price)~wordcount + polarity + FK_grd.lvl + formality, data = listin
gs_reviews)

stargazer::stargazer(model17,type = "text")
```

```
#Multiple regressions for different price ranges

model18 <- lm(log(price)~wordcount + polarity + FK_grd.lvl + formality, data = price_
low)
model19 <- lm(log(price)~wordcount + polarity + FK_grd.lvl + formality, data = price_
med)
model20 <- lm(log(price)~wordcount + polarity + FK_grd.lvl + formality, data = price_
high)

stargazer::stargazer(model18,model19, model20, type = "text")
```

# Part B dataset

## 2021/6/8

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)
```

```
rm(list=ls())
library(ggplot2)
library(dplyr)
library(tidytext)
library(jsonlite)
library(tidyr)
library(readr)
library(topicmodels)
library(geojsonR)
library(stringr)
library(dplyr)
library(gridExtra)
library(tidyr)
library(stm)
library(udpipe)
library(textcat)
library(cld2)
library(wordcloud)
set.seed(123113)

library(xml2)
library(rvest)
memory.limit(size=64000)
```

```
## [1] 64000
```

Download the dataset from airbnb.com

```
#crawl data from the website
path <- "http://insideairbnb.com/get-the-data.html"

#table preparation
data_set <- read_html(path)
list_of_tables <- data_set %>% html_nodes("table")

#City index for New York is 63.
city_index <- 63
table <-list_of_tables[[city_index]] %>% html_table()

#Getting the links to download the data
links <- list_of_tables[[city_index]] %>% html_nodes("a") %>% html_attr("href")

# Add it on the dataframe
table$links <- links

# cleaning the column names (lower case, replace _with spaces)
colnames(table) <- gsub(" ","_",tolower(colnames(table)))
colnames(table) <- gsub("/","_",colnames(table))

# Convert the date to a date using the as.Date function.
table$date_compiled <- as.Date(table$date_compiled,format = "%d %b,%Y")
```

```
# Get Review Data
Review_data <- table %>% arrange(desc(date_compiled)) %>%
  filter(grepl("Detailed Review",description)) %>%
  top_n(1)

# Get Listings Data
Listings_data <- table %>% arrange(desc(date_compiled)) %>%
  filter(grepl("Detailed Listings",description)) %>%
  top_n(1)
```

```
#download the files

NYC <- tolower(Listings_data$country_city[1])

download.file(url = Listings_data$links[1],destfile = "listings.csv.gz")

download.file(url = Review_data$links[1], destfile = "reviews.csv.gz")
```

```
listings_detailed <- readr::read_csv("listings.csv.gz")
reviews_detailed <- readr::read_csv("reviews.csv.gz")
```

# Cleaning for Review file

```
# Check the data structure of review file
str(reviews_detailed)
```

```
## spec_tbl_df [836,586 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ listing_id   : num [1:836586] 2595 2595 2595 2595 2595 ...
##  $ id           : num [1:836586] 17857 19176 19760 34320 46312 ...
##  $ date         : Date[1:836586], format: "2009-11-21" "2009-12-05" ...
##  $ reviewer_id  : num [1:836586] 50679 53267 38960 71130 117113 ...
##  $ reviewer_name: chr [1:836586] "Jean" "Cate" "Anita" "Kai-Uwe" ...
##  $ comments     : chr [1:836586] "Notre séjour de trois nuits.\r\nNous avons appré
cier L'appartement qui est très bien situé. Agréable, propre et"| __truncated__ "Grea
t experience." "I've stayed with my friend at the Midtown Castle for six days and it
was a lovely place to be. A big spacious r"| __truncated__ "We've been staying here f
or about 9 nights, enjoying to be in the center of the city, that never sleeps...shor
t"| __truncated__ ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   listing_id = col_double(),
##   ..   id = col_double(),
##   ..   date = col_date(format = ""),
##   ..   reviewer_id = col_double(),
##   ..   reviewer_name = col_character(),
##   ..   comments = col_character()
##   .. )
```

```
reviews_detailed$date <- as.Date(reviews_detailed$date)

# Filtering out reviews after feb 2020 from reviews_detailed dataset
(records_per_year <- reviews_detailed %>% group_by(lubridate::year(date)) %>% summari
se(count = n()))
```

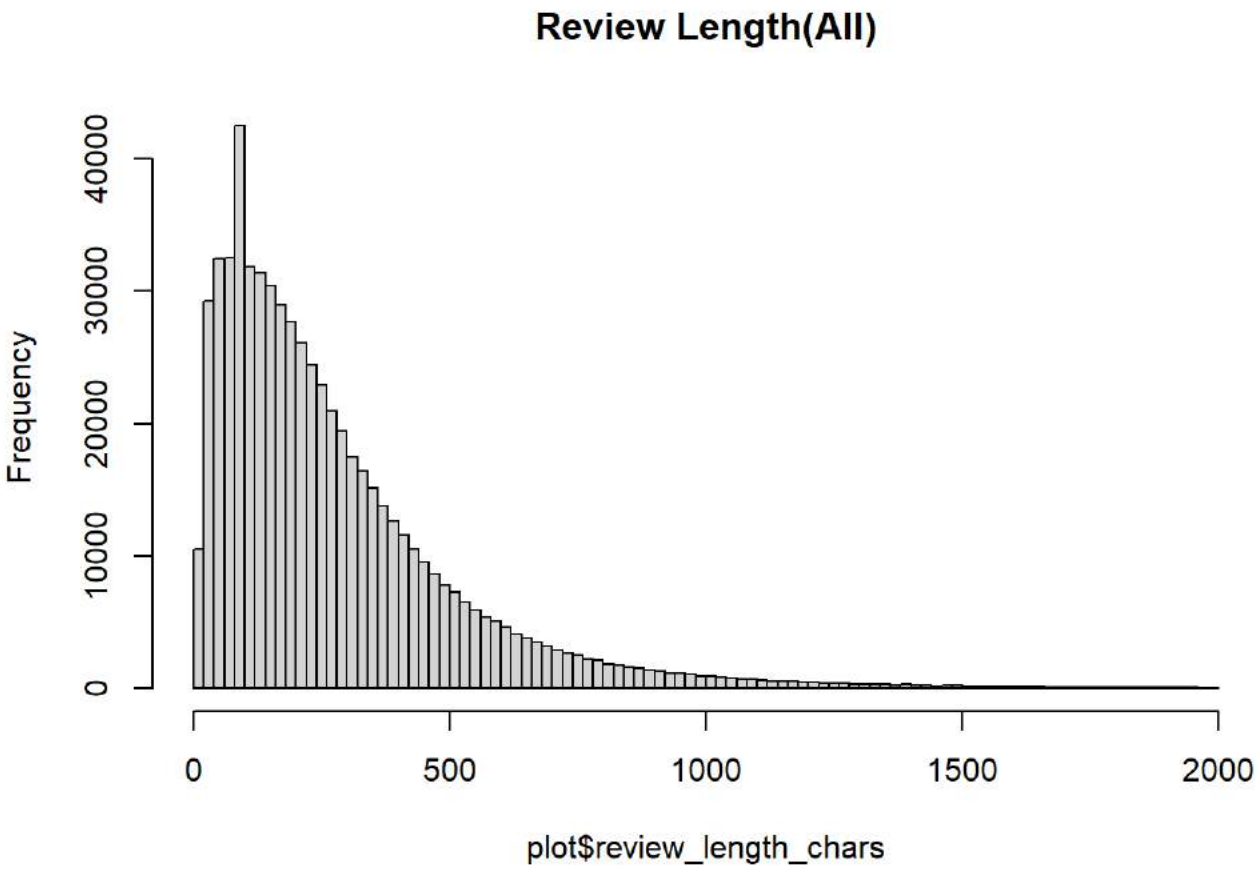
```
## # A tibble: 13 x 2
##    `lubridate::year(date)`  count
##                      <dbl>  <int>
##  1                    2009     93
##  2                    2010    696
##  3                    2011   2952
##  4                    2012   6176
##  5                    2013  11845
##  6                    2014  23391
##  7                    2015  46326
##  8                    2016  80251
##  9                    2017 115833
## 10                    2018 170301
## 11                    2019 239223
## 12                    2020 106937
## 13                    2021  32562
```

```
reviews_pre_corona <- reviews_detailed %>% filter(date<"2020-02-01")
head(reviews_pre_corona)
```

```
## # A tibble: 6 x 6
##   listing_id      id date       reviewer_id reviewer_name comments
##        <dbl>   <dbl> <date>           <dbl> <chr>         <chr>
## 1       2595   17857 2009-11-21       50679 Jean          "Notre séjour de troi~
## 2       2595   19176 2009-12-05       53267 Cate          "Great experience."
## 3       2595   19760 2009-12-10       38960 Anita         "I've stayed with my ~
## 4       2595   34320 2010-04-09       71130 Kai-Uwe       "We've been staying h~
## 5       2595   46312 2010-05-25      117113 Alicia        "We had a wonderful s~
## 6       2595 1238204 2012-05-07     1783688 Sergey        "Hi to everyone!\r\nW~
```

```r
# Step 1
# Cleaning up the language
# For some computers the character set is not
# automatically set to latin or ASCII
library(cld2)
reviews_pre_corona$comments <- iconv(reviews_pre_corona$comments)
reviews_pre_corona$language <- cld2::detect_language(reviews_pre_corona$comments)
reviews_pre_corona%>%
  filter(language =="en") -> reviews_pre_corona
```

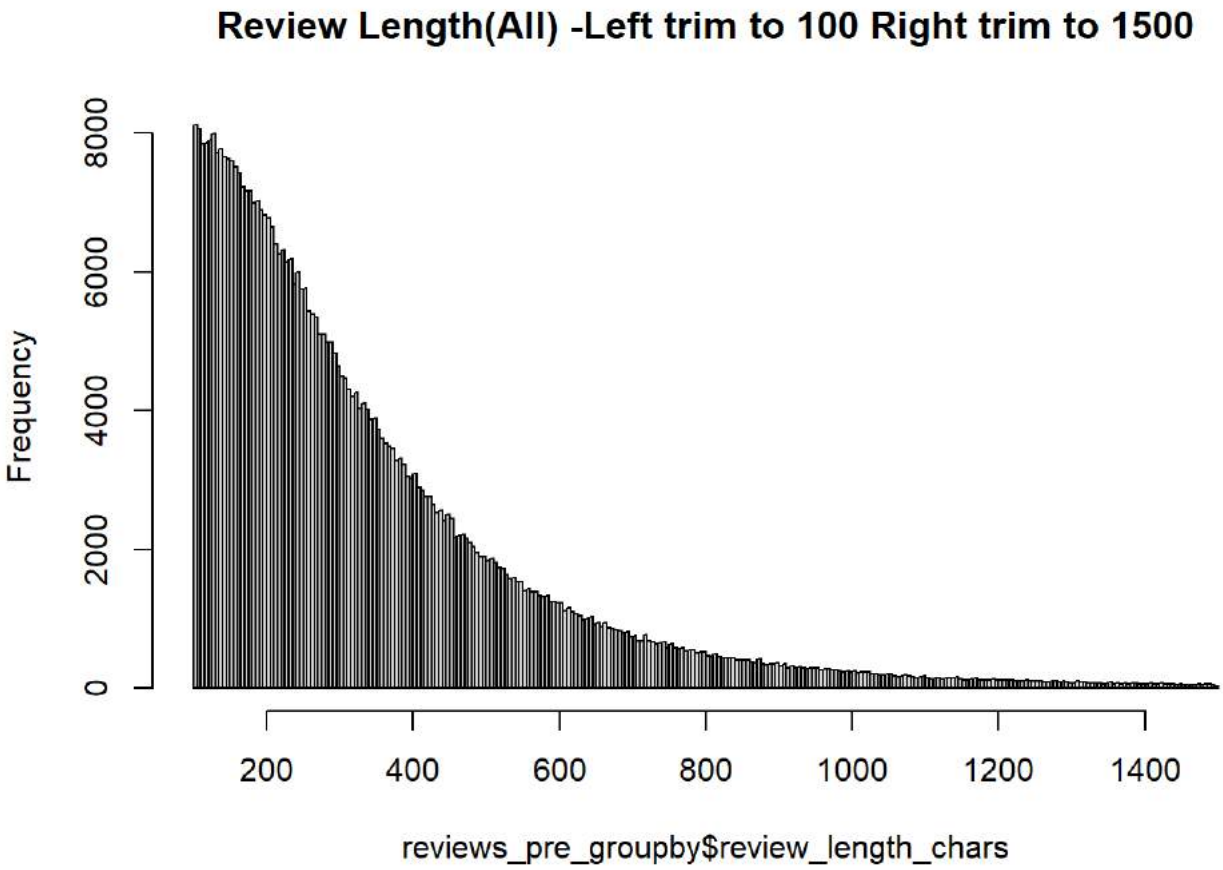
```r
# Step2
# Check the character length of each review
reviews_pre_corona$review_length_chars <- nchar(reviews_pre_corona$comments)
plot <- reviews_pre_corona %>% filter(review_length_chars<2000)

# Plot historgram of review length
hist(plot$review_length_chars,breaks = 100,main = "Review Length(All)")
```



```
# Check the characteristics of review length chars
reviews_pre_corona %>% summarise(mean = mean(review_length_chars), median = median(re
view_length_chars),sd = sd(review_length_chars), min = min(review_length_chars), max
 = max(review_length_chars))
```

```
## # A tibble: 1 x 5
##    mean median    sd   min   max
##   <dbl>  <int> <dbl> <int> <int>
## 1  286.    212  276.     7  5905
```

```
# Filter for minimum length of 100 and 1500 - come back to experiment with this.
reviews_pre_groupby <- reviews_pre_corona %>%
  filter(review_length_chars>100) %>%
  filter(review_length_chars<1500)
# Plot the histogram of review length
hist(reviews_pre_groupby$review_length_chars,breaks = 200,main = "Review Length(All)
 -Left trim to 100 Right trim to 1500")
```




```
reviews_pre_groupby %>% summarise(mean = mean(review_length_chars), median = median(r
eview_length_chars),sd = sd(review_length_chars), min = min(review_length_chars), max
= max(review_length_chars))
```

```
## # A tibble: 1 x 5
##    mean median    sd   min   max
##   <dbl>  <int> <dbl> <int> <int>
## 1  341.    273  230.   101  1499
```

```
# # Do a right trim to maximum 1000 chars
# reviews_pre_groupby<- reviews_pre_groupby %>%
#   filter(review_length_chars<1000)
#
# hist(reviews_pre_groupby$review_length_chars,breaks = 200,main = "Review Length(Al
l) -Right trim to 1000")
```

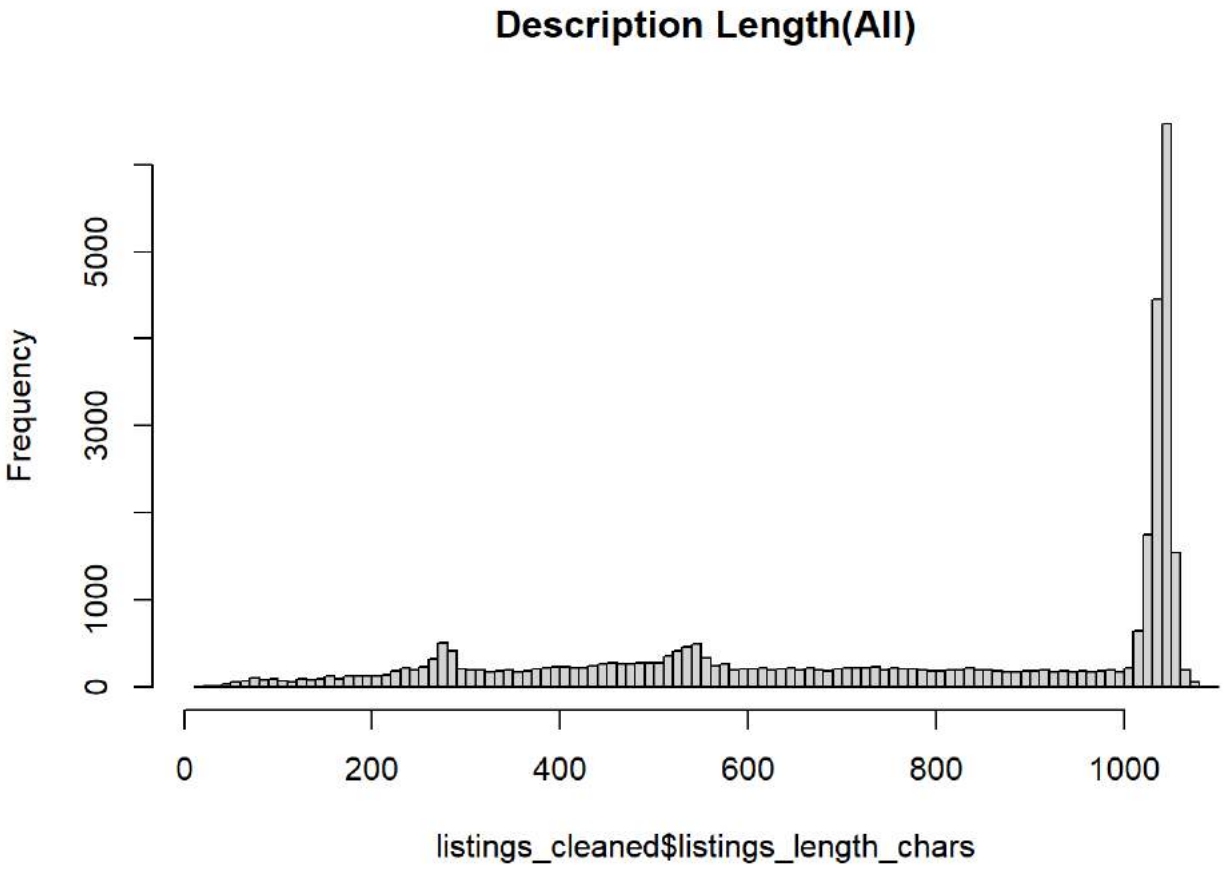
```
# Step 3
# Group by Reviews by listing ids
reviews_groupbylisting <- reviews_pre_groupby %>%
  unnest_tokens(word,comments)%>%
  group_by(listing_id) %>%
  summarise(comments_grouped  = paste(word,collapse = "  "))
```

# Cleaning for listing file

```
# Step 4
# Filtering unwanted columns from listings dataset:
listings_cleaned <- listings_detailed %>% dplyr::select(-c(listing_url, scrape_id, la
st_scraped , picture_url,host_url, host_thumbnail_url,host_picture_url, host_neighbou
rhood, host_listings_count, neighbourhood, bathrooms, minimum_minimum_nights, maximum
_minimum_nights, minimum_maximum_nights, maximum_maximum_nights, minimum_nights_avg_n
tm, maximum_nights_avg_ntm,calendar_updated, calendar_last_scraped, has_availability,
calendar_last_scraped, first_review, last_review, license))

# Format property type and room type for future analysis
listings_cleaned$property_type <- as.factor(listings_cleaned$property_type)
listings_cleaned$room_type <- as.factor(listings_cleaned$room_type)

# Combine names and descriptions in one column
listings_cleaned$description_combined <- str_c(listings_cleaned$name, ' ', listings_c
leaned$description)
listings_cleaned <- listings_cleaned %>% dplyr::select(-c(name, description))

# Finding number of verifications, amenities, removing $ from price
i <- 1
for (i in 1:length(listings_cleaned$host_verifications)) {

  listings_cleaned$verifications[i] <- length(strsplit(listings_cleaned$host_verifica
tions[i], split= " ")[[1]])
  listings_cleaned$num_amenities[i] <- length(strsplit(listings_cleaned$amenities[i],
split= " ")[[1]])
  listings_cleaned$price[i] <- as.numeric(gsub("\\D", "",listings_cleaned$price[i]))/
100
  listings_cleaned$num_bath[i] <- as.numeric(gsub("[^0-9.]", "",listings_cleaned$bath
rooms_text[i]))
    i <- i+1
}
# Rename the id to listing_ids
colnames(listings_cleaned)[1] <- "listing_id"
```

```
# Step 5
# Cleaning up the language
# For some computers the character set is not
# automatically set to latin or ASCII

listings_cleaned$description_combined <- iconv(listings_cleaned$description_combined,
from = 'UTF-8', to = 'ASCII//TRANSLIT')
listings_cleaned$description_language <- cld2::detect_language(listings_cleaned$descr
iption_combined)

# Filter English descriptions
listings_cleaned <- listings_cleaned %>%
  filter(description_language =="en")
```

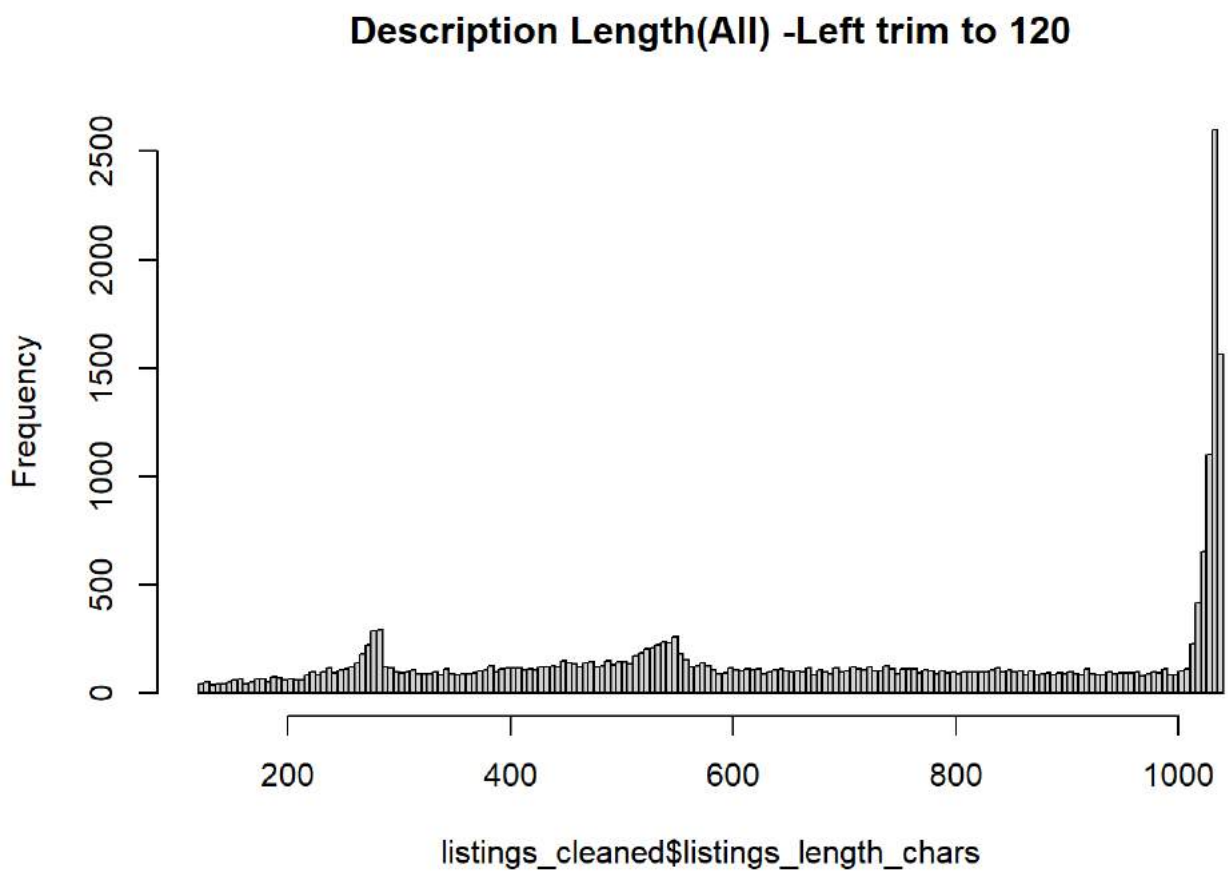
```
# Step 6
# Check the character length of each description
listings_cleaned$listings_length_chars <- nchar(listings_cleaned$description_combine
d)

# Plot the histogram
hist(listings_cleaned$listings_length_chars,breaks = 100,main = "Description Length(A
ll)")
```




```
# Check the characteristics of description length chars
listings_cleaned %>% summarise(mean = mean(listings_length_chars), median = median(li
stings_length_chars),sd = sd(listings_length_chars), min = min(listings_length_char
s), max = max(listings_length_chars) )
```

```
## # A tibble: 1 x 5
##    mean median    sd   min   max
##   <dbl>  <int> <dbl> <int> <int>
## 1  762.    880  306.    14  1098
```

```
# Filter for minimum length of 120, 1040
listings_cleaned <- listings_cleaned %>%
  filter(listings_length_chars>120) %>%
  filter(listings_length_chars<1040)

hist(listings_cleaned$listings_length_chars,breaks = 200,main = "Description Length(All) -Left trim to 120")
```




```
# Lets do a right trim to maximum 1000 chars
# listings_cleaned <- listings_cleaned %>%
#   filter(listings_length_chars<1040)
#
# hist(listings_cleaned$listings_length_chars,breaks = 200,main = "Description Length(All) -Right trim to 1000")
```

```
#Step 7
listings_reviews <- listings_cleaned %>% left_join(reviews_groupbylisting,  by = "listing_id")
head(listings_reviews)
```

```
## # A tibble: 6 x 56
##   listing_id neighborhood_overview  host_id host_name  host_since host_location
##        <dbl> <chr>                    <dbl> <chr>      <date>     <chr>
## 1       2595 Centrally located in ~    2845 Jennifer   2008-09-09 New York, New~
## 2       5121 <NA>                      7356 Garon      2009-02-03 New York, New~
## 3       5178 Theater district, man~    8967 Shunichi   2009-03-03 New York, New~
## 4       5203 Our neighborhood is f~    7490 MaryEllen  2009-02-05 New York, New~
## 5       6848 <NA>                     15991 Allen & I~ 2009-05-06 New York, New~
## 6       6990 Location: Five minute~   16800 Cyn        2009-05-12 New York, New~
## # ... with 50 more variables: host_about <chr>, host_response_time <chr>,
## #   host_response_rate <chr>, host_acceptance_rate <chr>,
## #   host_is_superhost <lgl>, host_total_listings_count <dbl>,
## #   host_verifications <chr>, host_has_profile_pic <lgl>,
## #   host_identity_verified <lgl>, neighbourhood_cleansed <chr>,
## #   neighbourhood_group_cleansed <chr>, latitude <dbl>, longitude <dbl>,
## #   property_type <fct>, room_type <fct>, accommodates <dbl>,
## #   bathrooms_text <chr>, bedrooms <dbl>, beds <dbl>, amenities <chr>,
## #   price <chr>, minimum_nights <dbl>, maximum_nights <dbl>,
## #   availability_30 <dbl>, availability_60 <dbl>, availability_90 <dbl>,
## #   availability_365 <dbl>, number_of_reviews <dbl>,
## #   number_of_reviews_ltm <dbl>, number_of_reviews_l30d <dbl>,
## #   review_scores_rating <dbl>, review_scores_accuracy <dbl>,
## #   review_scores_cleanliness <dbl>, review_scores_checkin <dbl>,
## #   review_scores_communication <dbl>, review_scores_location <dbl>,
## #   review_scores_value <dbl>, instant_bookable <lgl>,
## #   calculated_host_listings_count <dbl>,
## #   calculated_host_listings_count_entire_homes <dbl>,
## #   calculated_host_listings_count_private_rooms <dbl>,
## #   calculated_host_listings_count_shared_rooms <dbl>, reviews_per_month <dbl>,
## #   description_combined <chr>, verifications <int>, num_amenities <int>,
## #   num_bath <dbl>, description_language <chr>, listings_length_chars <int>,
## #   comments_grouped <chr>
```

```
# Step 8
# Join reviews without removing syntactical features
data_syntactical <- listings_cleaned %>% left_join(reviews_pre_groupby)

# Data Backup
metadata <- listings_reviews
```

```
# Step 9
# Substitute digits with space
listings_reviews$comments_grouped <- gsub('[[:digit:]]+',' ', listings_reviews$commen
ts_grouped)
listings_reviews$description_combined <- gsub('[[:digit:]]+',' ', listings_reviews$de
scription_combined)

# Substitute punctuations with space
listings_reviews$comments_grouped <- gsub('[[:punct:]]+',' ', listings_reviews$commen
ts_grouped)
listings_reviews$description_combined<- gsub('[[:punct:]]+',' ', listings_reviews$des
cription_combined)

listings_reviews$description_combined<- gsub( '[^[:alnum:]]',' ', listings_reviews$de
scription_combined)
```

# Tokenization

```
# Step 10
# Split the dataset at a size of 10000
split_size <- 10000
tokens_list <- split(listings_reviews,
                     rep(1:ceiling(nrow(listings_reviews)
                                   /split_size),
                         each=split_size,
                         length.out=nrow(listings_reviews)))

# Tokenization for both comments (review) and descriptions
review_tokens_all <- data.frame()
for(i in 1:length(tokens_list)){
   review_tokens_h <- tokens_list[[i]] %>%
    unnest_tokens(word,comments_grouped) %>%
    count(word,listing_id) %>%
    anti_join(stop_words)
   print(i)
   review_tokens_all <- bind_rows(review_tokens_all,review_tokens_h)
}
```

```
## [1] 1
## [1] 2
## [1] 3
```

```
description_tokens_all <- data.frame()
for(i in 1:length(tokens_list)){
   description_tokens_h <- tokens_list[[i]] %>%
    unnest_tokens(word,description_combined) %>%
    count(word,listing_id) %>%
    anti_join(stop_words)
   print(i)
   description_tokens_all <- bind_rows(description_tokens_all,description_tokens_h)
}
```

```
## [1] 1
## [1] 2
## [1] 3
```

# Review tokens

```
# Step 11
# Calculate the token length

# review_tokens_all$token_length <- nchar(review_tokens_all$clean)
#
# # Lets have a look on the distribution
# review_tokens_all %>% group_by(token_length) %>% summarise(total =n())
```

```
#filter out non english words
review_tokens_all$word <- iconv(review_tokens_all$word,from = 'UTF-8', to = 'ASCII//T
RANSLIT')
review_tokens_all$clean<- gsub('[[:punct:]]+','', review_tokens_all$word)

review_tokens_all$token_length <- nchar(review_tokens_all$clean)

# Lets have a look on the distribution
review_tokens_all %>% group_by(token_length) %>% summarise(total =n())
```

```
## # A tibble: 26 x 2
##    token_length  total
##           <int>  <int>
##  1            0     64
##  2            1   5904
##  3            2  40991
##  4            3 169624
##  5            4 502564
##  6            5 509993
##  7            6 500126
##  8            7 452466
##  9            8 353169
## 10            9 255900
## # ... with 16 more rows
```

```
# review_tokens_all$language <- cld3::detect_language(review_tokens_all$clean)

# # Filter English descriptions
# review_tokens_all <- review_tokens_all %>%
#   filter(language =="en")
```

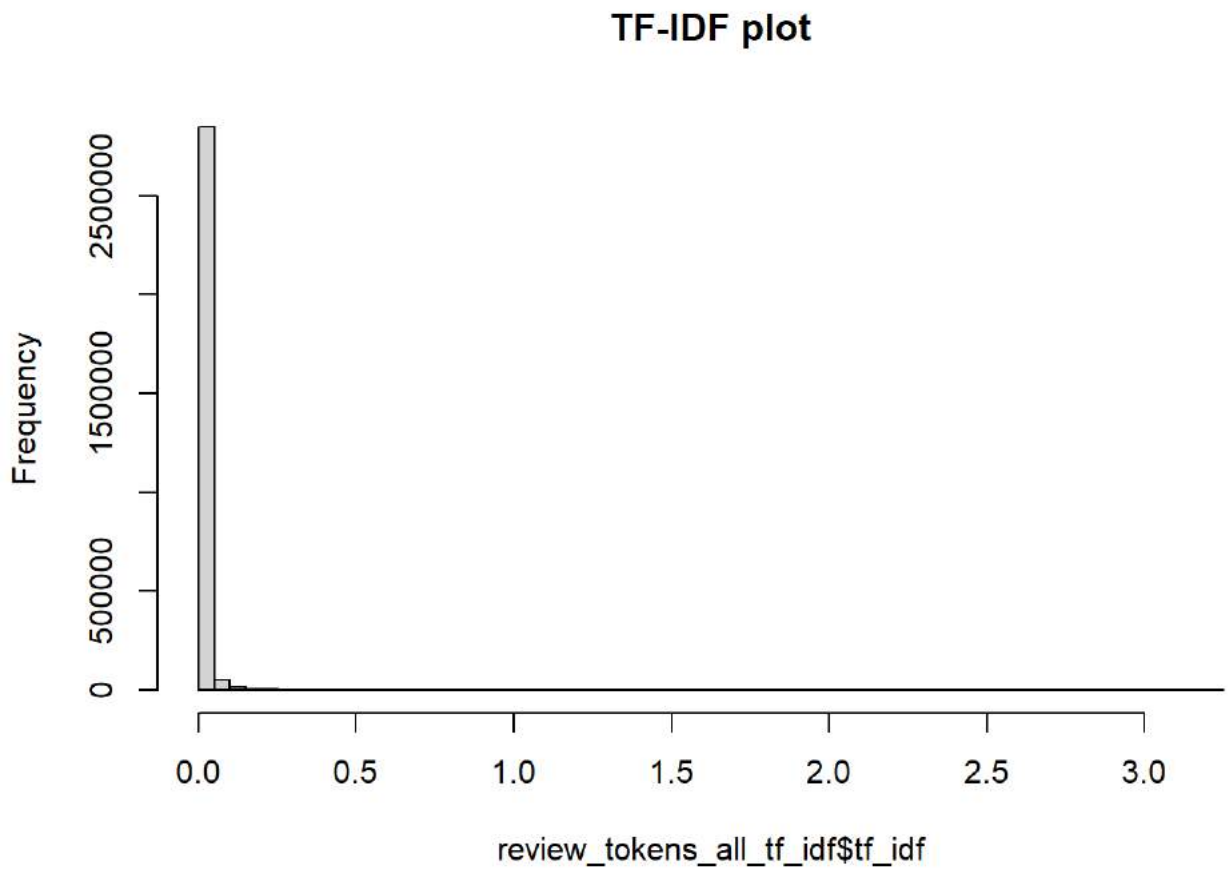
```
# Remove tokens with length less than 3
review_tokens_all <- review_tokens_all %>%
  filter(token_length>3)

# Lets have a look at the distribution of tokens again
review_tokens_all %>% group_by(token_length) %>%
  summarise(total =n()) %>%
  arrange(desc(token_length))
```

```
## # A tibble: 21 x 2
##    token_length total
##           <int> <int>
##  1           39     1
##  2           24     1
##  3           22     1
##  4           21     2
##  5           20     4
##  6           19     2
##  7           18    41
##  8           17   179
##  9           16   367
## 10           15  4031
## # ... with 11 more rows
```

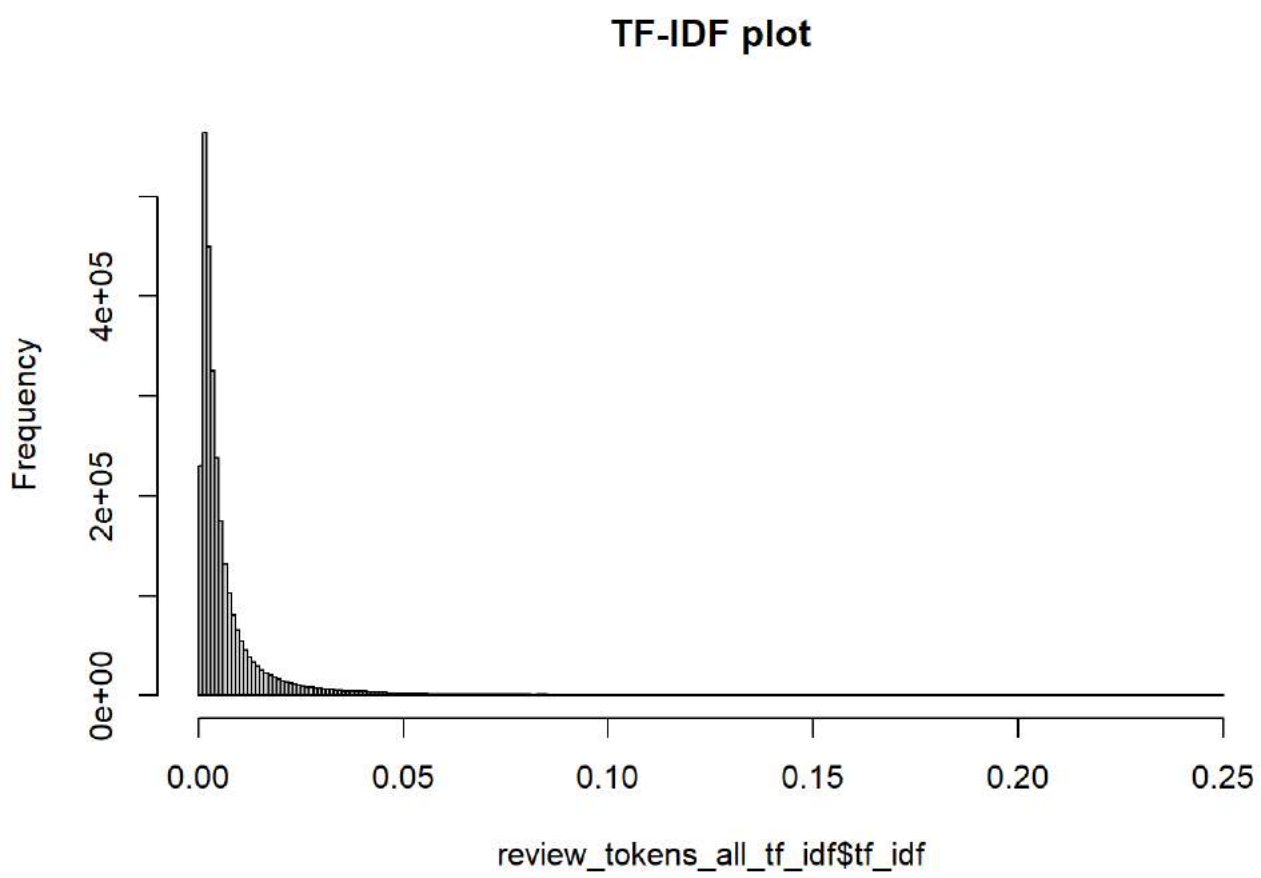
```
# Remove those tokens with chars more than 17
review_tokens_all <- review_tokens_all %>%
  filter(token_length<=17)

review_tokens_all$language <- NULL
```

```
# Step 12
# Calculate tf-idf using the listing as a document
review_tokens_all_tf_idf <- review_tokens_all %>%
  bind_tf_idf(clean,listing_id,n)
```

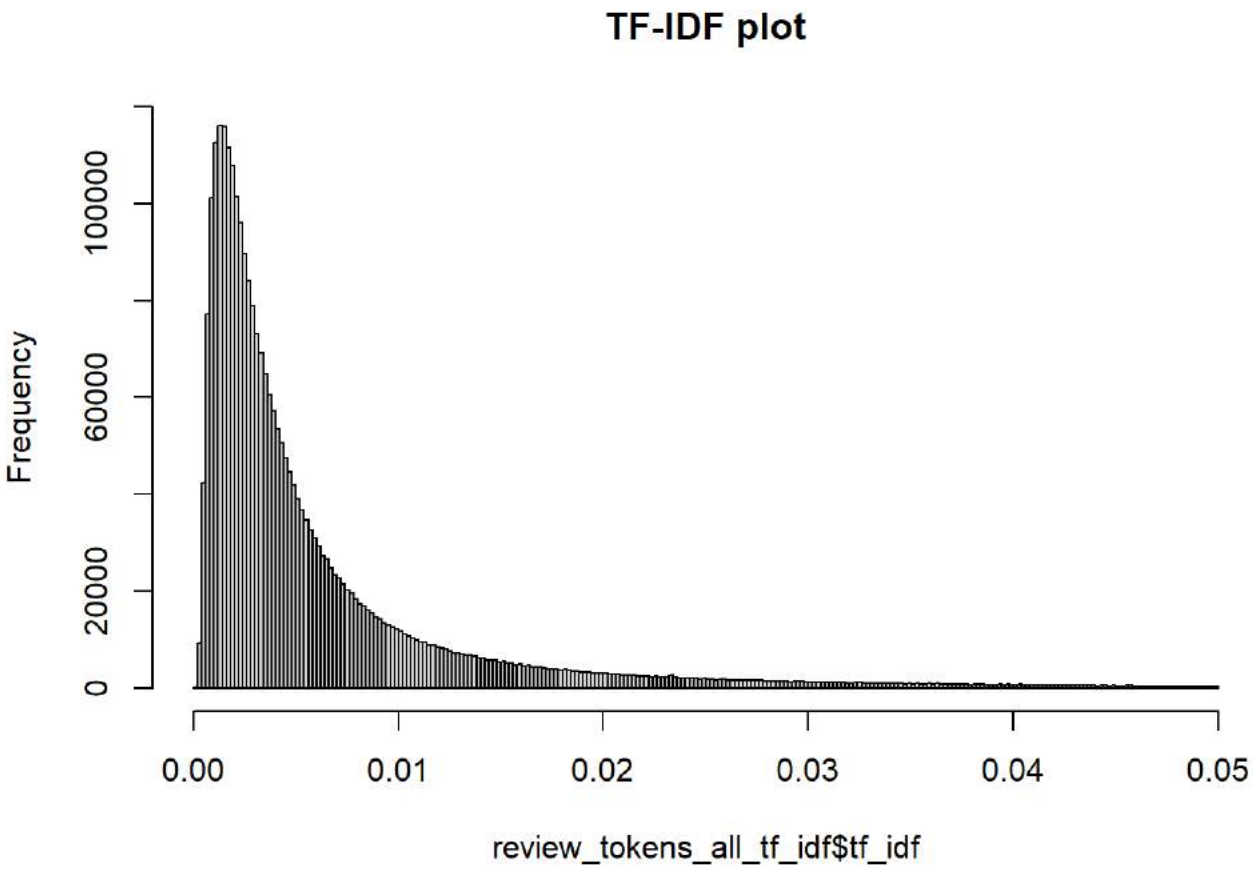
```
# Plot the distribution of tf-idf
hist(review_tokens_all_tf_idf$tf_idf,breaks = 100,main="TF-IDF plot")
```




```
review_tokens_all_tf_idf <- review_tokens_all_tf_idf %>%
  filter(tf_idf<0.25)
hist(review_tokens_all_tf_idf$tf_idf,breaks = 200,main="TF-IDF plot")
```




```
# Ok from the plot we see that the cut-off value
# is at 0.05
review_tokens_all_tf_idf <- review_tokens_all_tf_idf %>%
  filter(tf_idf<0.05)

hist(review_tokens_all_tf_idf$tf_idf,breaks = 200,main="TF-IDF plot")
```




```
# Remove also very common terms those with tf-idf <0.001 as shown in the chart below
review_tokens_all_tf_idf <- review_tokens_all_tf_idf %>%
  filter(tf_idf>0.001)

review_tokens_all %>% group_by(word) %>%
  summarise(total =n()) %>%
  arrange(desc(total)) %>%
  top_n(40)
```

```
## # A tibble: 40 x 2
##    word        total
##    <chr>       <int>
##  1 stay        12994
##  2 location    11979
##  3 clean       11934
##  4 apartment   11795
##  5 host        11303
##  6 nice        11047
##  7 recommend   10947
##  8 comfortable 10379
##  9 subway      10282
## 10 close       10209
## # ... with 30 more rows
```

```
# Add manhattan to the stopword dictionary
```

# Description tokens

```
# Step 11
# Calculate the token length
description_tokens_all$word <- iconv(description_tokens_all$word,from = 'UTF-8', to =
'ASCII//TRANSLIT')
description_tokens_all$clean<- gsub('[[:punct:]]+','', description_tokens_all$word)
description_tokens_all$token_length <- nchar(description_tokens_all$clean)

# Look at the distribution
description_tokens_all %>% group_by(token_length) %>% summarise(total =n())
```

```
## # A tibble: 26 x 2
##    token_length  total
##           <int>  <int>
##  1            2  43002
##  2            3  65064
##  3            4 187669
##  4            5 216707
##  5            6 192299
##  6            7 175738
##  7            8 136915
##  8            9  86674
##  9           10  31834
## 10           11  30416
## # ... with 16 more rows
```

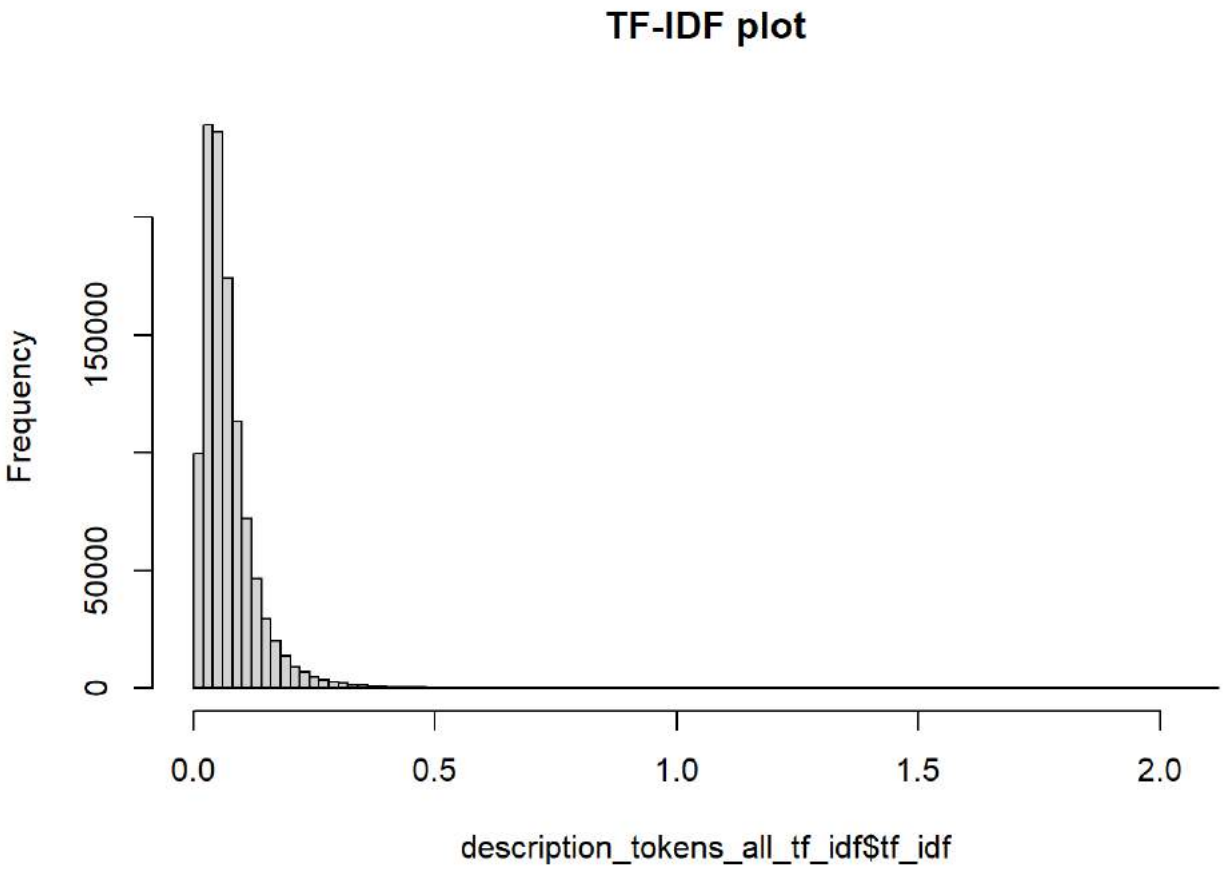
```
# Remove tokens with 1 and 2 characters
description_tokens_all <- description_tokens_all %>%
  filter(token_length>3)

# Look at the distribution of tokens again on the oposite side
description_tokens_all %>% group_by(token_length) %>%
  summarise(total =n()) %>%
  arrange(desc(token_length))
```

```
## # A tibble: 24 x 2
##    token_length total
##           <int> <int>
##  1           33     1
##  2           26     1
##  3           25     1
##  4           24     1
##  5           23     1
##  6           22     1
##  7           21     2
##  8           20     3
##  9           19     5
## 10           18    14
## # ... with 14 more rows
```

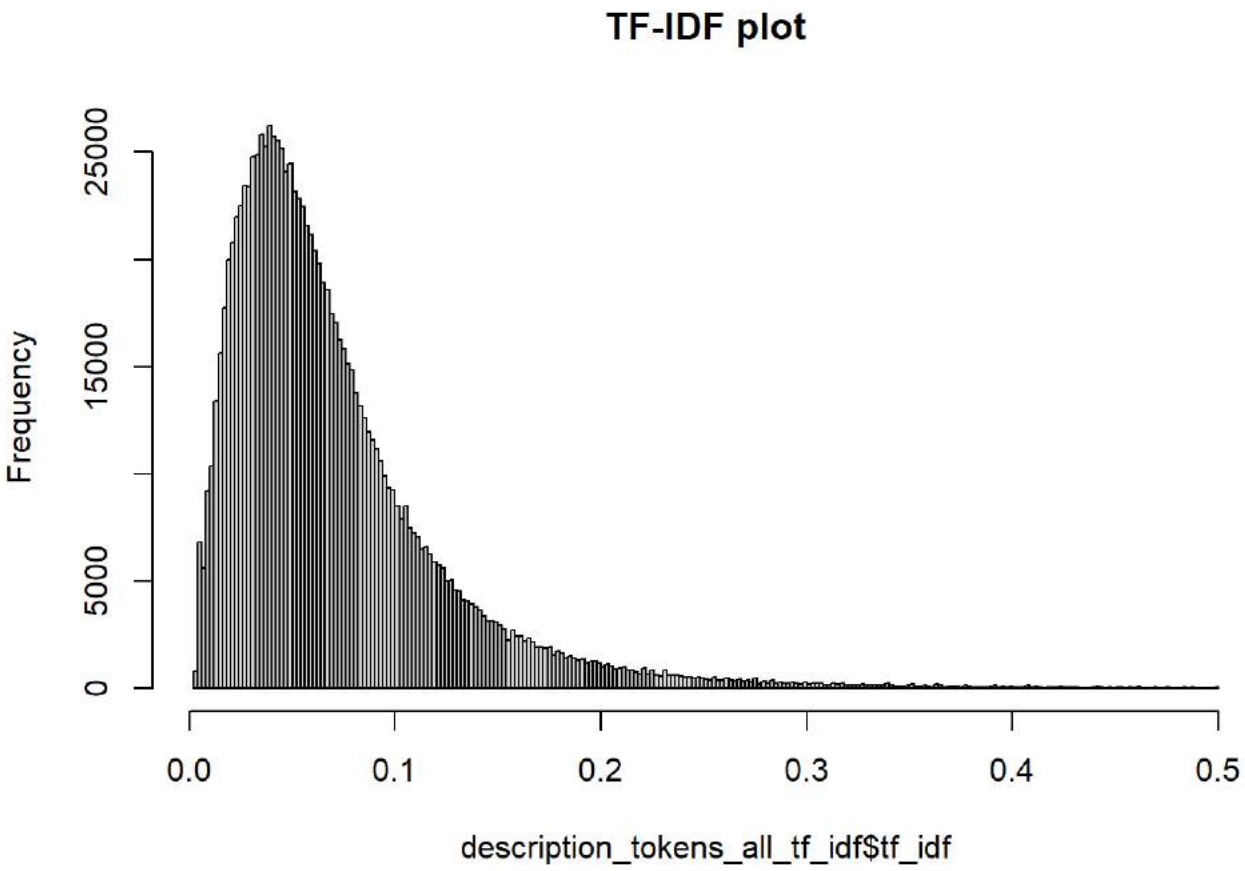
```
# Remove tokens with length greater than 17
description_tokens_all <- description_tokens_all %>%
  filter(token_length<=17)
```

```
# Step 12
# Calculate tf-idf using the listing as a document
description_tokens_all_tf_idf <- description_tokens_all %>%
  bind_tf_idf(clean,listing_id,n)
```

```
# Ok the tfidf is cut
# lets try to filter important words
# using the left and right trim
hist(description_tokens_all_tf_idf$tf_idf,breaks = 100,main="TF-IDF plot")
```




```
# Check the distribution with left trim to 0.5
description_tokens_all_tf_idf <- description_tokens_all_tf_idf %>%
  filter(tf_idf<0.5)

hist(description_tokens_all_tf_idf$tf_idf,breaks = 200,main="TF-IDF plot")

# Remove also very common terms those with tf-idf <0.001 as shown in the chart below

description_tokens_all_tf_idf <- description_tokens_all_tf_idf %>%
  filter(tf_idf>0.001)

hist(description_tokens_all_tf_idf$tf_idf,breaks = 200,main="TF-IDF plot")
```




```
description_tokens_all %>% group_by(word) %>%
  summarise(total =n()) %>%
  arrange(desc(total)) %>%
  top_n(40)
```

```
## # A tibble: 40 x 2
##    word      total
##    <chr>     <int>
##  1 space     18578
##  2 apartment 16863
##  3 kitchen   13802
##  4 bedroom   13367
##  5 access    12412
##  6 guest     10876
##  7 living     9392
##  8 manhattan  9312
##  9 private    9208
## 10 bathroom   9184
## # ... with 30 more rows
```

```
# Notice that manhattan could be a stopword and can augment the stopwords
```

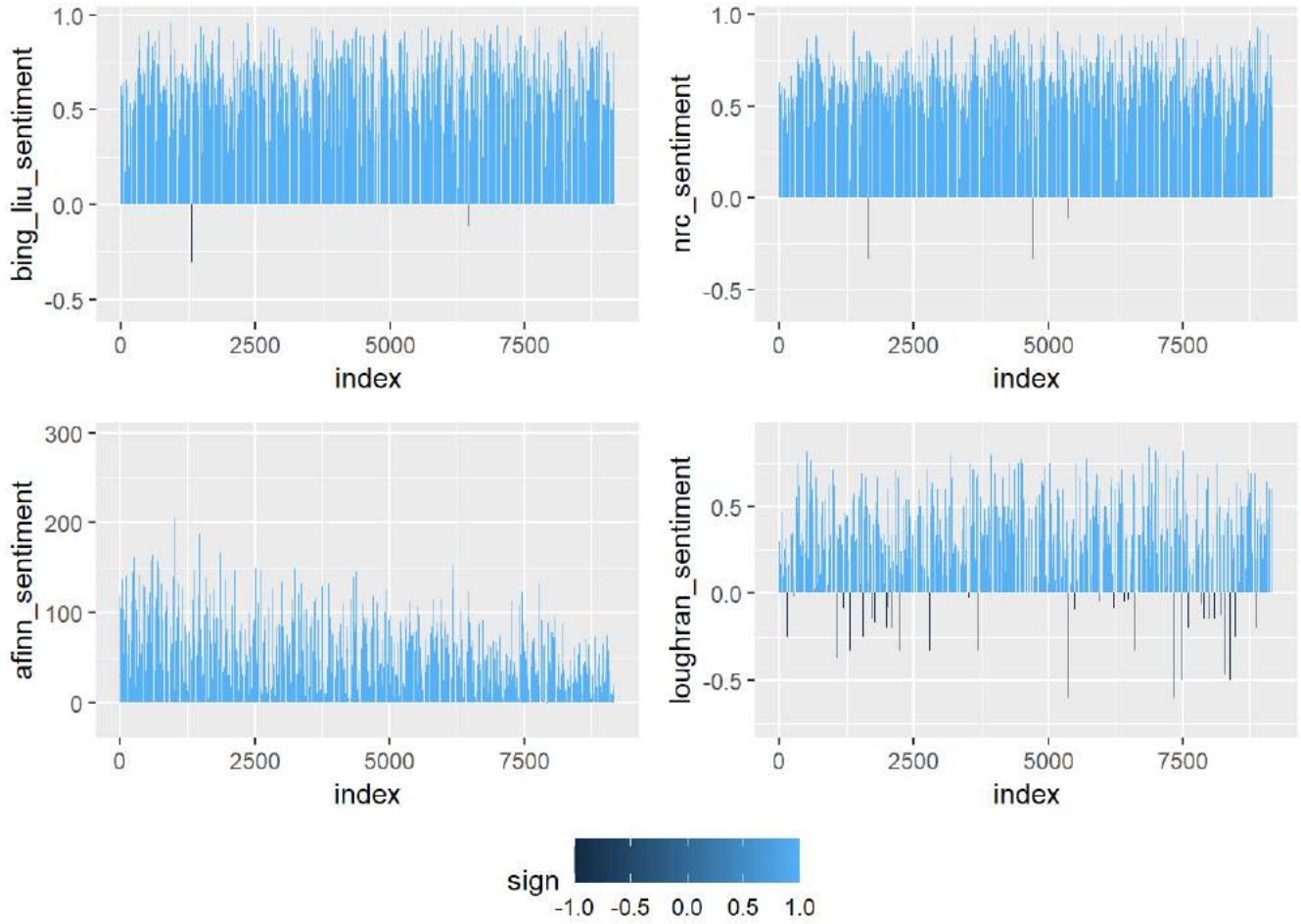
```
# Customize stopwords dictionary
tibble(word = c("neighborhood",
                "manhattan",
                "located",
                "brooklyn",
                "queens",
                "staten",
                "island",
                "bronx",
                "stay",
                "staying"),lexicon = rep("custom",10)) %>%
  bind_rows(stop_words) -> custom_dict
```

# Part B

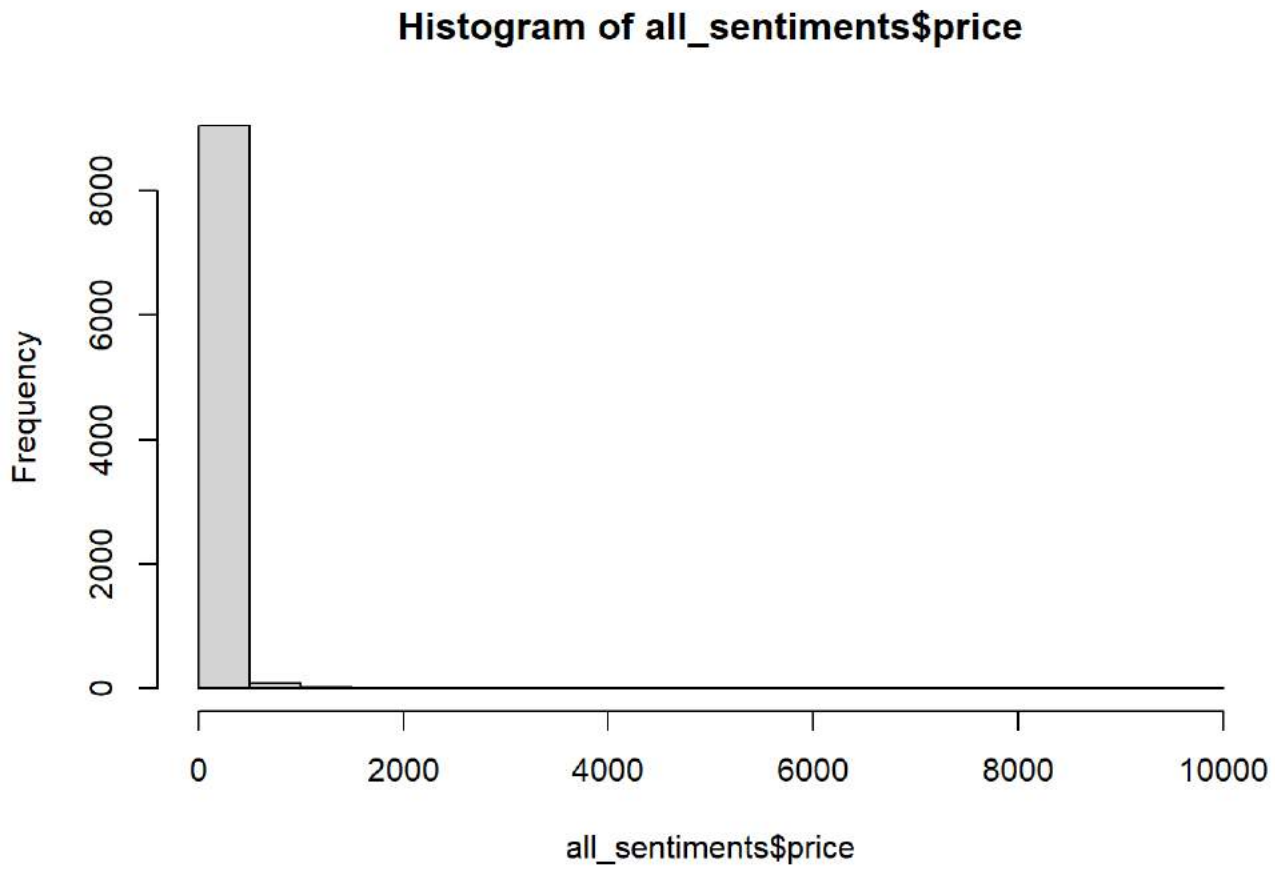
## Affection Categorization

```
# Creating df with listing_id and price

listing_id_price_score <- listings_reviews %>%
  dplyr::select(listing_id,price, review_scores_rating)

# Computing Sentiment of reviews using different dictionaries

# Bing-Liu
review_tokens_all %>% inner_join(get_sentiments("bing"), by = c("clean" = "word")) %
>%
  count(sentiment,listing_id) %>% spread(sentiment,n) %>%
  mutate(bing_liu_sentiment = (positive-negative)/(positive+negative)) %>%
  dplyr::select(listing_id,bing_liu_sentiment) -> bing_liu_sentiment

bing_liu_sentiment <- bing_liu_sentiment %>%
  left_join(listing_id_price_score)

# NRC
review_tokens_all %>% inner_join(get_sentiments("nrc"), by = c("clean" = "word")) %>%
  count(sentiment,listing_id) %>% spread(sentiment,n) %>%
  mutate(nrc_sentiment = (positive-negative)/(positive+negative)) %>%
  dplyr::select(listing_id,nrc_sentiment) -> nrc_sentiment

nrc_sentiment <- nrc_sentiment %>%
  left_join(listing_id_price_score)

# Afinn
review_tokens_all %>% inner_join(get_sentiments("afinn"), by = c("clean" = "word")) %
>%
  group_by(listing_id) %>%
  summarise(afinn_sentiment = sum(value)) -> afinn_sentiment

afinn_sentiment <- afinn_sentiment %>%
  left_join(listing_id_price_score)

# Loughran
review_tokens_all %>% inner_join(get_sentiments("loughran"),by = c("clean" = "word"))
%>%
  count(sentiment,listing_id) %>% spread(sentiment,n) %>%
  mutate(loughran_sentiment = (positive-negative)/(positive+negative)) %>%
  dplyr::select(listing_id,loughran_sentiment) -> loughran_sentiment

loughran_sentiment <- loughran_sentiment %>%
  left_join(listing_id_price_score)

# Combine all into single df

all_sentiments <- bing_liu_sentiment %>%
  left_join(nrc_sentiment) %>%
  left_join(afinn_sentiment) %>%
  left_join(loughran_sentiment) %>%
  na.omit()
```

```
library(ggpubr)
# Plot the sentiment
p1 <- all_sentiments %>%
  mutate(index = row_number(), sign = sign(bing_liu_sentiment)) %>%
  ggplot(aes(x=index,y=bing_liu_sentiment, fill = sign))+
  geom_bar(stat="identity")

p2 <- all_sentiments %>%
  mutate(index = row_number(), sign = sign(nrc_sentiment)) %>%
  ggplot(aes(x=index,y=nrc_sentiment, fill = sign))+
  geom_bar(stat="identity")

p3 <- all_sentiments %>%
  mutate(index = row_number(), sign = sign(afinn_sentiment)) %>%
  ggplot(aes(x=index,y=afinn_sentiment, fill = sign))+
  geom_bar(stat="identity")

p4 <- all_sentiments %>%
  mutate(index = row_number(), sign = sign(loughran_sentiment)) %>%
  ggplot(aes(x=index,y=loughran_sentiment, fill = sign))+
  geom_bar(stat="identity")

ggarrange(p1,p2,p3,p4,common.legend = TRUE, legend = "bottom")
```



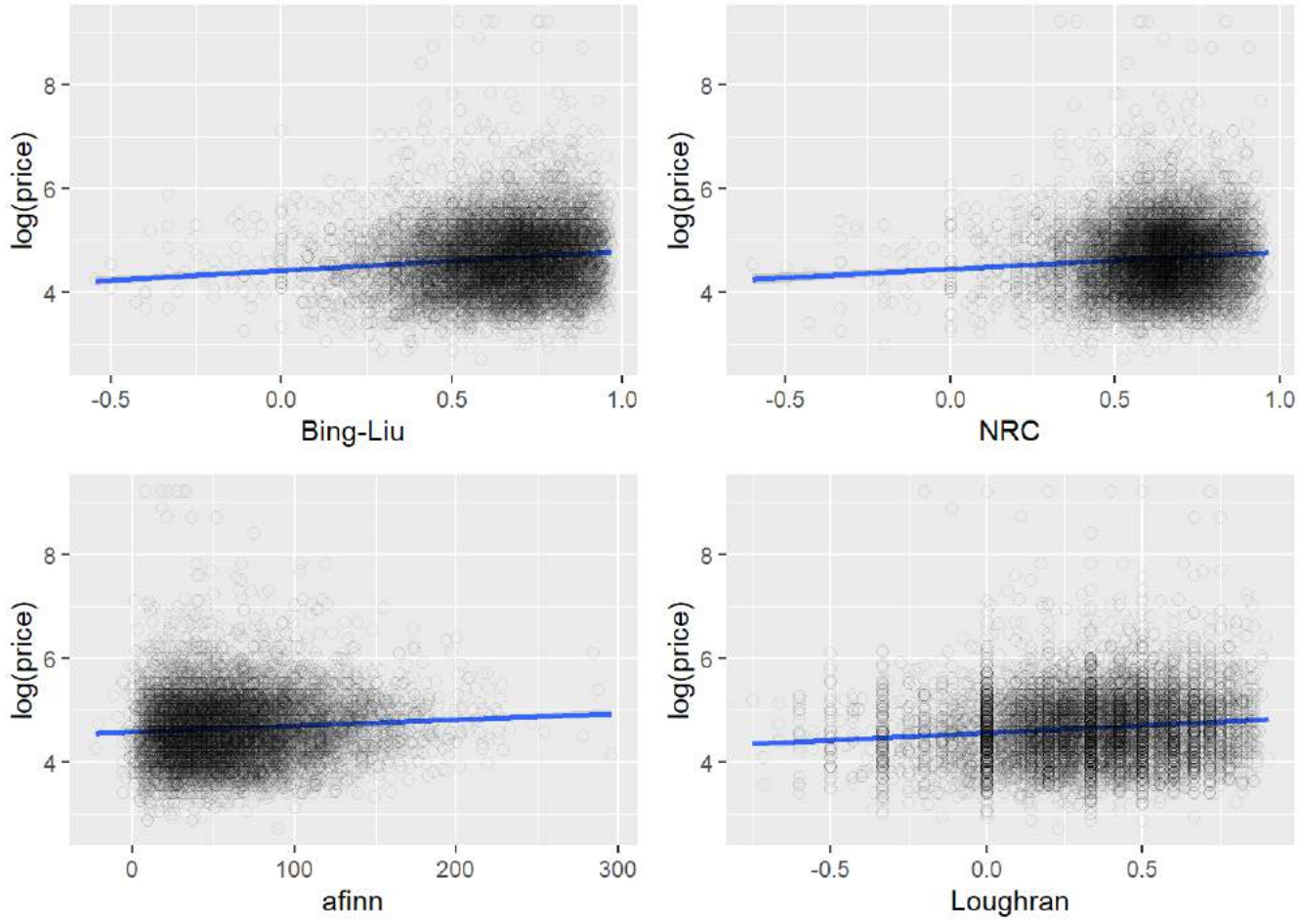

```
# Models of Price against Sentiment
all_sentiments$price = as.numeric(all_sentiments$price)

#To check distribution to decide whether to use log(price)
hist(all_sentiments$price)
```



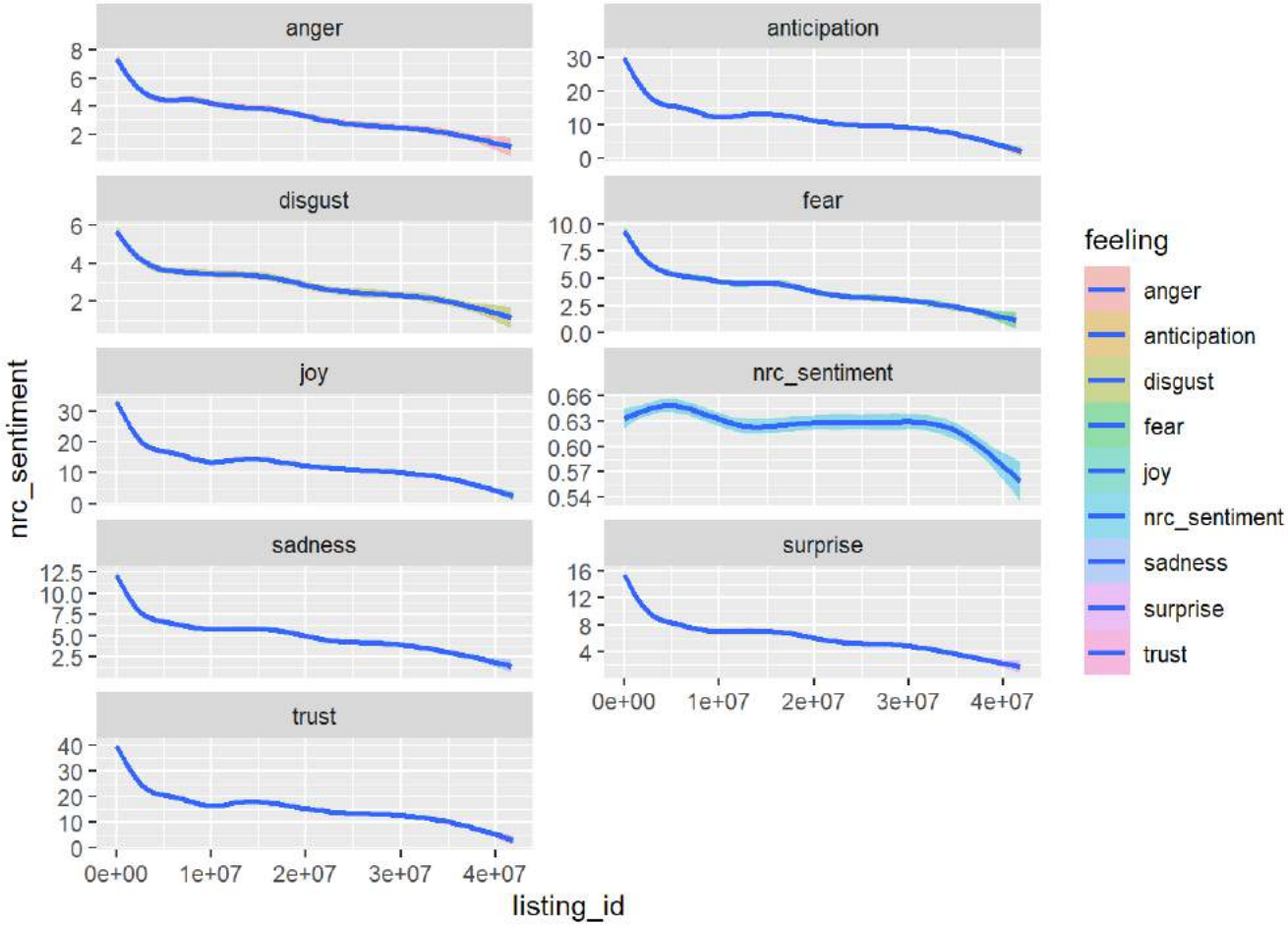

```
# Run regression models (Price against Sentiment)
model1 <- lm(log(all_sentiments$price)~all_sentiments$bing_liu_sentiment)
model2 <- lm(log(all_sentiments$price)~all_sentiments$nrc_sentiment)
model3 <- lm(log(all_sentiments$price)~all_sentiments$afinn_sentiment)
model4 <- lm(log(all_sentiments$price)~all_sentiments$loughran_sentiment)

# Check the regression results
stargazer::stargazer(model1,model2,model3,model4, type = "text")
```

```
##
## ================================================================================
##                                          Dependent variable:
##                                  ---------------------------------------
##                                                    price)
##                                     (1)        (2)       (3)       (4)
## --------------------------------------------------------------------------------
## bing_liu_sentiment                0.372***
##                                   (0.035)
##
## nrc_sentiment                               0.331***
##                                              (0.041)
##
## afinn_sentiment                                        0.001***
##                                                        (0.0002)
##
## loughran_sentiment                                                    0.286***
##                                                                       (0.025)
##
## Constant                          4.413***  4.442***  4.577***   4.558***
##                                   (0.023)    (0.027)   (0.013)   (0.011)
##
## --------------------------------------------------------------------------------
## Observations                       9,173      9,173     9,173     9,173
## R2                                 0.012      0.007     0.005     0.014
## Adjusted R2                        0.012      0.007     0.005     0.014
## Residual Std. Error (df = 9171)    0.647      0.649     0.649     0.646
## F Statistic (df = 1; 9171)      112.341*** 64.401*** 49.038*** 131.678***
## ================================================================================
## Note:                                          *p<0.1; **p<0.05; ***p<0.01
```

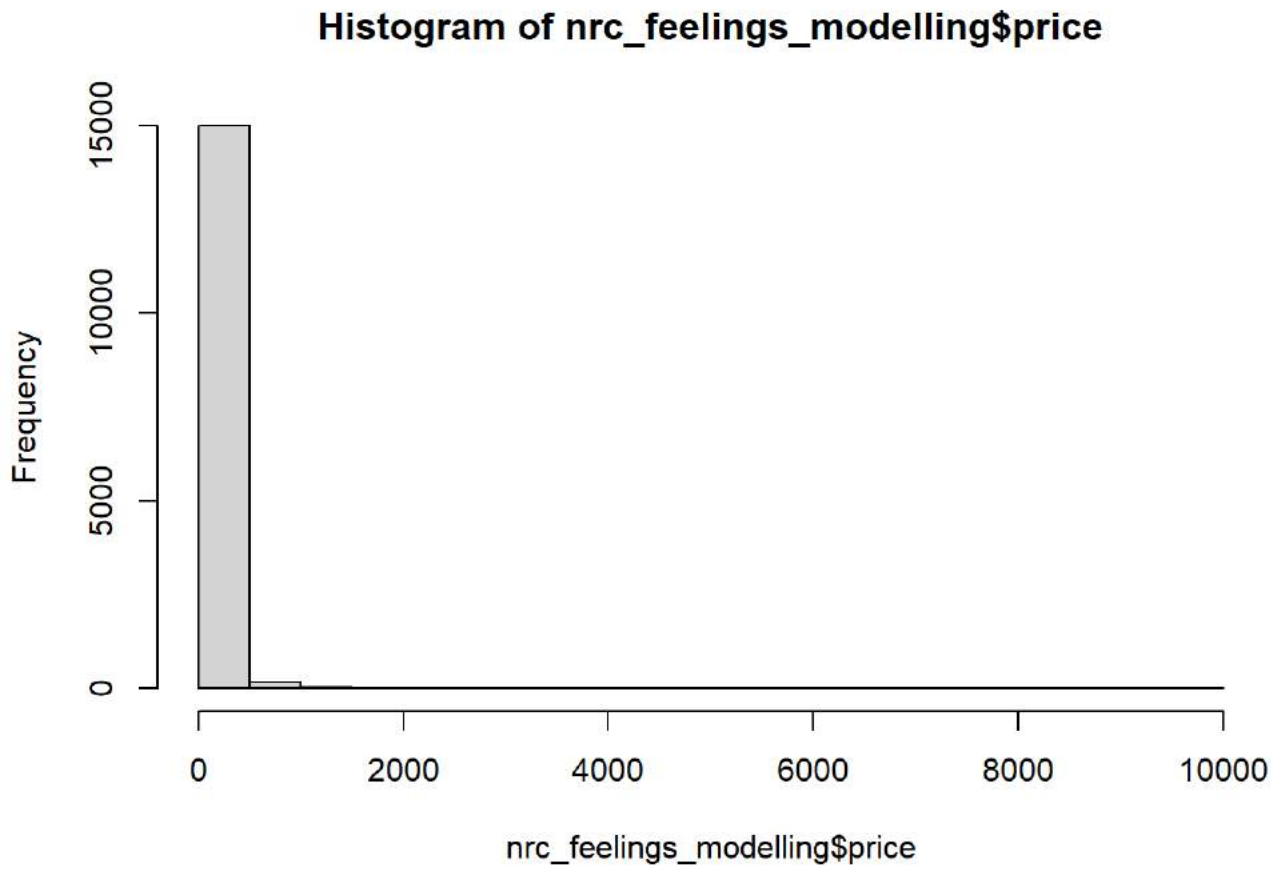
```
# Plot the simple regression
gr_bl = all_sentiments %>%
  dplyr::select(bing_liu_sentiment,price) %>%
  na.omit() %>%
  ggplot(aes(x=bing_liu_sentiment,y=log(price)))  + geom_smooth(method="lm") + geom_p
oint(size = 2, shape=1,alpha=0.1) + xlab("Bing-Liu")

gr_nrc = all_sentiments %>%
  dplyr::select(nrc_sentiment,price) %>%
  na.omit() %>%
  ggplot(aes(x=nrc_sentiment,y=log(price)))  + geom_smooth(method="lm") + geom_point
(size = 2, shape=1,alpha=0.1) + xlab("NRC")

gr_afinn = all_sentiments %>%
  dplyr::select(afinn_sentiment,price) %>%
  na.omit() %>%
  ggplot(aes(x=afinn_sentiment,y=log(price)))  + geom_smooth(method="lm") + geom_poin
t(size = 2, shape=1,alpha=0.1) + xlab("afinn")

gr_loughran = all_sentiments %>%
  dplyr::select(loughran_sentiment,price) %>%
  na.omit() %>%
  ggplot(aes(x=loughran_sentiment,y=log(price)))  + geom_smooth(method="lm") + geom_p
oint(size = 2, shape=1,alpha=0.1)+ xlab("Loughran")

grid.arrange(gr_bl, gr_nrc, gr_afinn, gr_loughran,
          ncol = 2, nrow = 2)
```



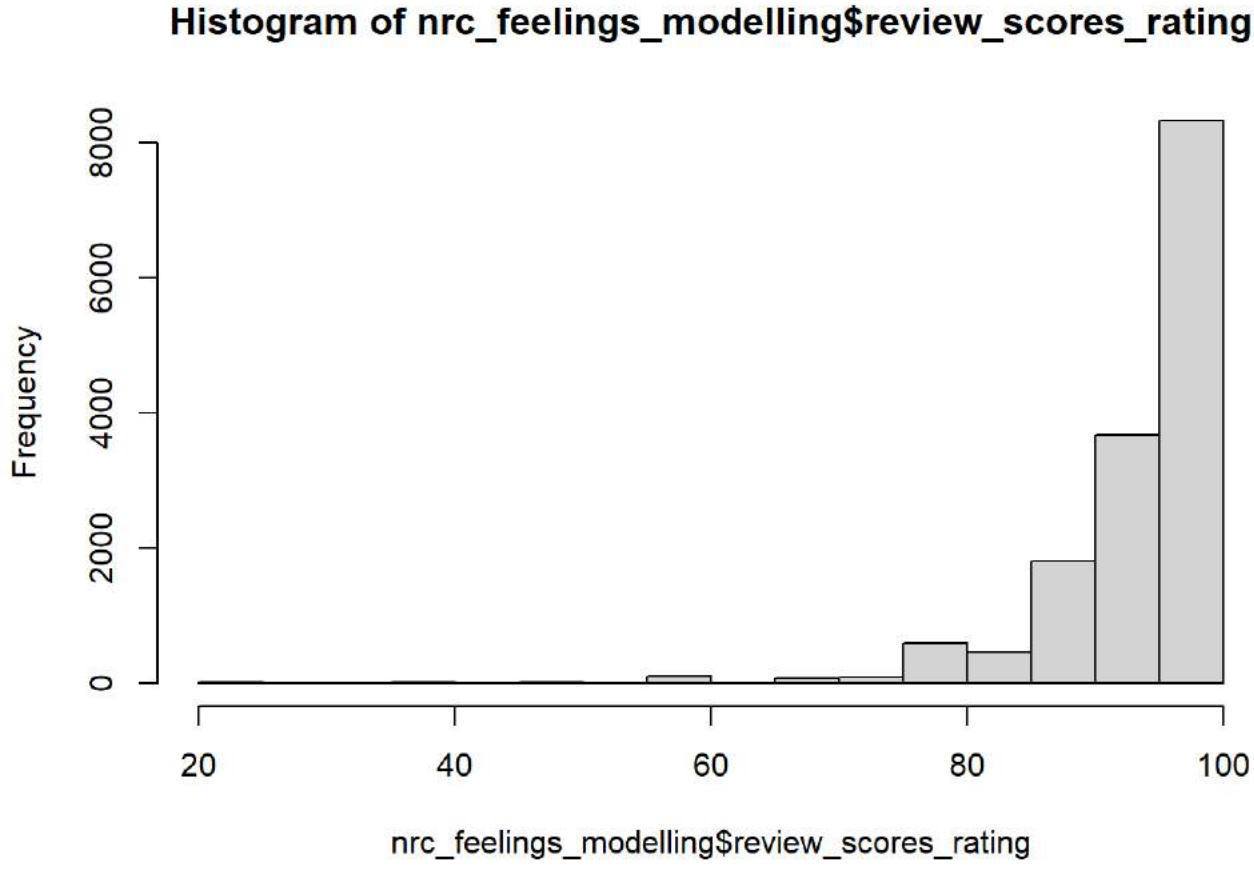
```
# Extract feelings from NRC dictionary
nrc_feelings <- review_tokens_all %>%
  inner_join(get_sentiments("nrc"), by = c("clean" = "word")) %>%
  count(sentiment,listing_id) %>%
  spread(sentiment,n) %>%
  mutate(nrc_sentiment = (positive-negative)/(positive+negative)) %>%
  dplyr::select(-c(positive,negative)) %>%
  pivot_longer(anger:nrc_sentiment,names_to = "feeling",values_to="nrc_sentiment")

# Plot the feelings extracted
nrc_feelings %>%
  ggplot(aes(x=listing_id,y=nrc_sentiment,fill=feeling))+
  geom_smooth()+
  facet_wrap(~feeling,scales="free_y",ncol=2)
```




```
#Running regressions for nrc feelings
nrc_feelings_modelling = nrc_feelings %>% spread(feeling, nrc_sentiment, fill = 0)

nrc_feelings_modelling = left_join(nrc_feelings_modelling, listing_id_price_score)

nrc_feelings_modelling$price = as.numeric(nrc_feelings_modelling$price)

#Check distrbution to see if log shiuld be used
hist(nrc_feelings_modelling$price)
```




```
hist(nrc_feelings_modelling$review_scores_rating)
```

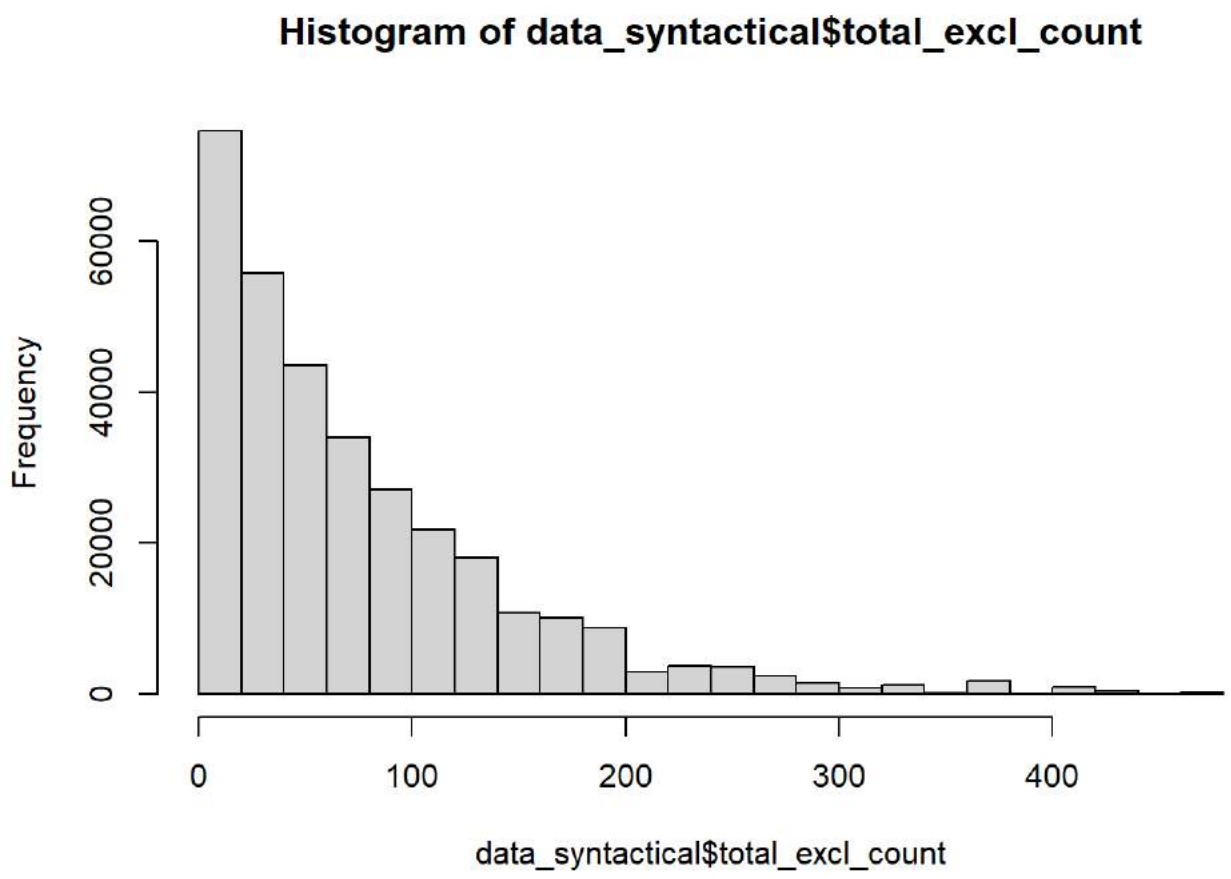


```
model5 <- lm(log(price)~.-review_scores_rating -listing_id, data=nrc_feelings_modelli
ng)
model6 <- lm(log(review_scores_rating)~. -price -listing_id, data=nrc_feelings_modell
ing)

stargazer::stargazer(model5,model6, type = "text")
```

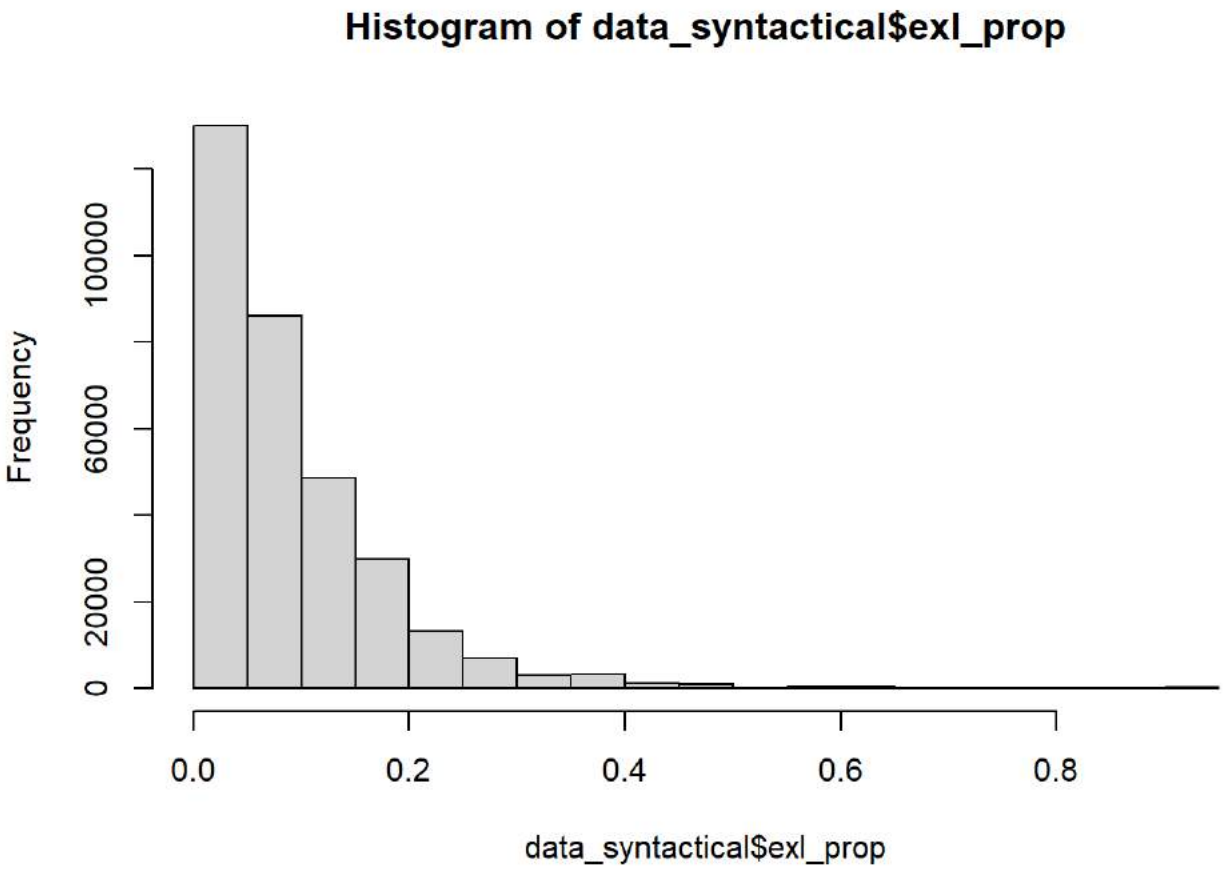
```
##
## ===================================================================
##                                        Dependent variable:
##                                    -----------------------------------
##                                    log(price) log(review_scores_rating)
##                                       (1)                (2)
## -------------------------------------------------------------------
## anger                                -0.001           -0.002***
##                                     (0.005)            (0.001)
##
## anticipation                         -0.001           -0.002***
##                                     (0.002)           (0.0004)
##
## disgust                               0.006           -0.009***
##                                     (0.004)            (0.001)
##
## fear                                -0.020***           -0.001*
##                                     (0.004)            (0.001)
##
## joy                                 0.022***          0.003***
##                                     (0.002)           (0.0003)
##
## nrc_sentiment                        0.046**          0.016***
##                                     (0.019)            (0.003)
##
## sadness                               0.001             0.0001
##                                     (0.003)            (0.001)
##
## surprise                            0.011***           0.001**
##                                     (0.003)            (0.001)
##
## trust                               -0.017***           0.0004
##                                     (0.002)           (0.0003)
##
## Constant                            4.596***          4.518***
##                                     (0.010)            (0.002)
##
## -------------------------------------------------------------------
## Observations                         15,208             15,208
## R2                                   0.019              0.062
## Adjusted R2                          0.018              0.062
## Residual Std. Error (df = 15198)     0.647              0.101
## F Statistic (df = 9; 15198)        32.087***          112.208***
## ===================================================================
## Note:                                    *p<0.1; **p<0.05; ***p<0.01
```

# Syntactical features

```
# Calculate the number of exclamation marks in each listing
data_syntactical %>%
  mutate(excl_count = str_count(comments, "!")) %>%
  group_by(listing_id) %>%
  mutate(total_excl_count = sum(excl_count),
         exl_prop = total_excl_count/listings_length_chars) -> data_syntactical

# Check the distribution of the number and proportion of exclamations marks
hist(data_syntactical$total_excl_count)
```



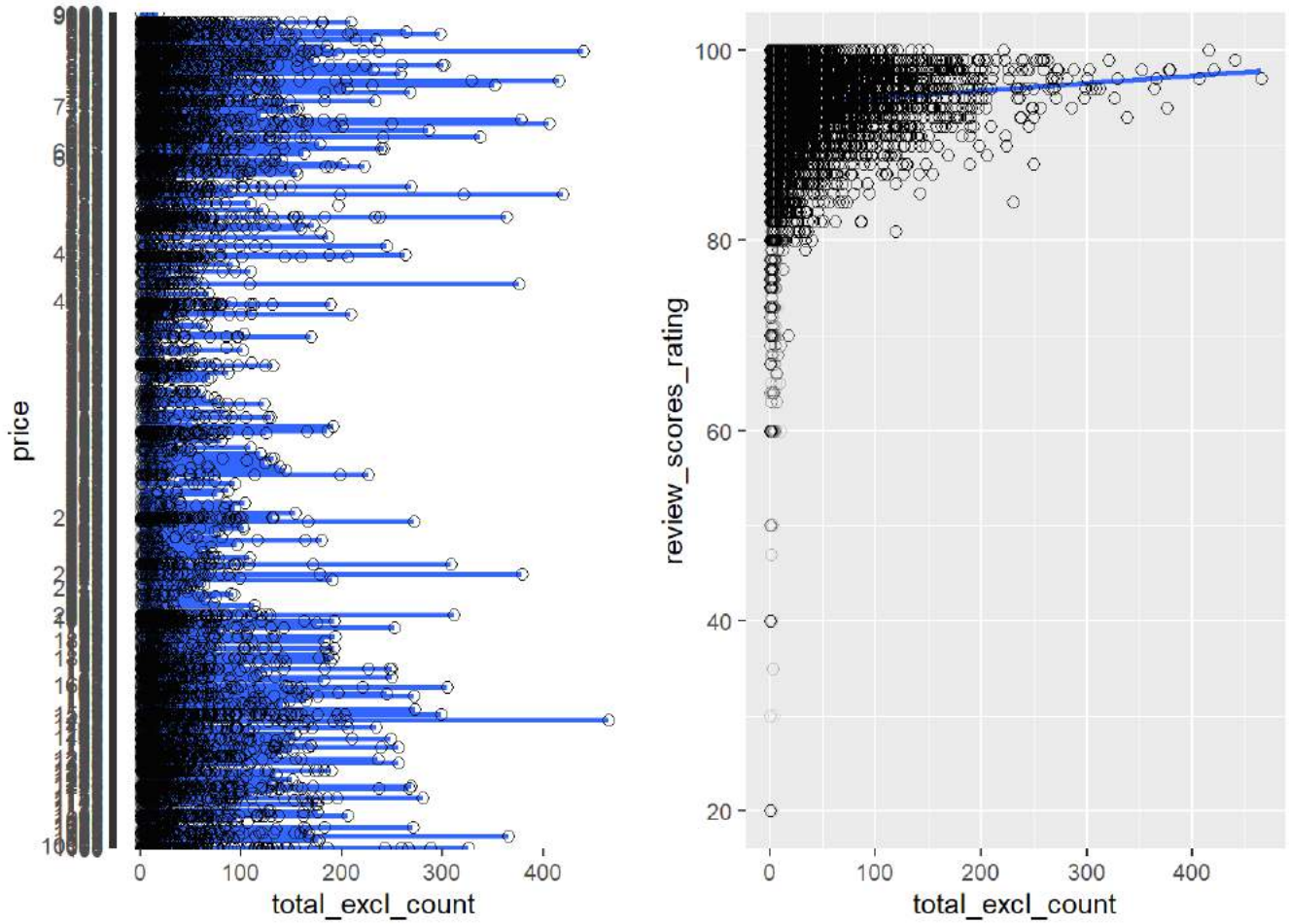

```
hist(data_syntactical$exl_prop)
```



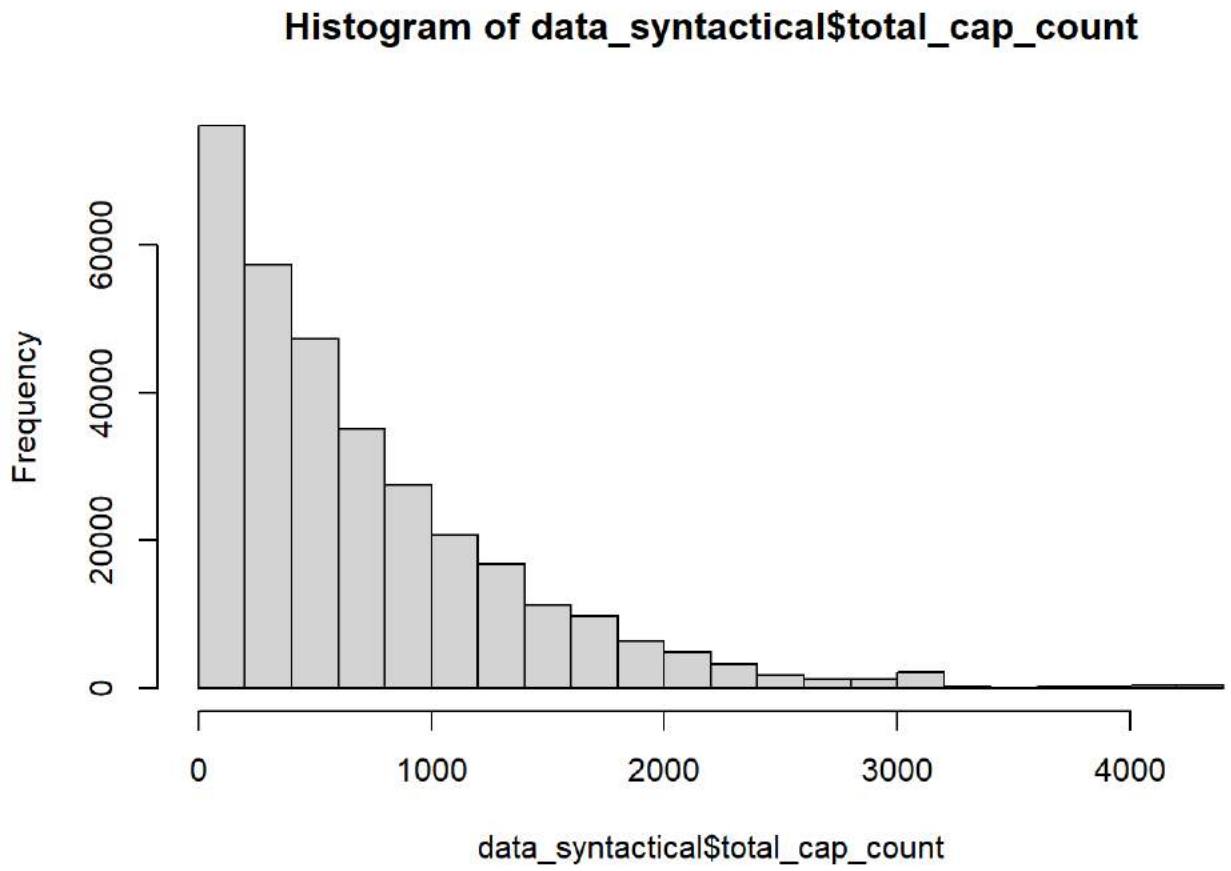

```
# Run regression model Price/Rating against
model7 <- lm(log(as.numeric(price))~total_excl_count,
             data=data_syntactical)

model8 <- lm(log(review_scores_rating)~total_excl_count,
             data = data_syntactical)

model9 <- lm(log(as.numeric(price))~exl_prop,
             data=data_syntactical)

model10 <- lm(log(review_scores_rating)~exl_prop,
             data = data_syntactical)

stargazer::stargazer(model7,model8,model9,model10,type = "text")
```

```
##
## ================================================================================
==============================================
##                                                                  Dependent variabl
e:
##                     ---------------------------------------------------------------
-----------------------------------------------
##                       log(as.numeric(price))     log(review_scores_rating)   log(a
s.numeric(price))   log(review_scores_rating)
##                                 (1)                          (2)
(3)                         (4)
## ------------------------------------------------------------------------------------
-----------------------------------------------
## total_excl_count            0.0001***                    0.0001***
##                             (0.00001)                    (0.00000)
##
## exl_prop
0.005                     0.065***
##
(0.012)                     (0.001)
##
## Constant                     4.628***                    4.544***
4.637***                   4.545***
##                              (0.002)                    (0.0001)
(0.002)                    (0.0001)
##
## ------------------------------------------------------------------------------------
-----------------------------------------------
## Observations                 324,577                      324,570
324,577                    324,570
## R2                            0.0002                      0.019
0.00000                     0.015
## Adjusted R2                   0.0002                      0.019
-0.00000                    0.015
## Residual Std. Error    0.603 (df = 324575)          0.046 (df = 324568)       0.60
4 (df = 324575)        0.047 (df = 324568)
## F Statistic         70.040*** (df = 1; 324575) 6,441.039*** (df = 1; 324568) 0.186
(df = 1; 324575) 4,933.101*** (df = 1; 324568)
## ================================================================================
==============================================
## Note:
*p<0.1; **p<0.05; ***p<0.01
```

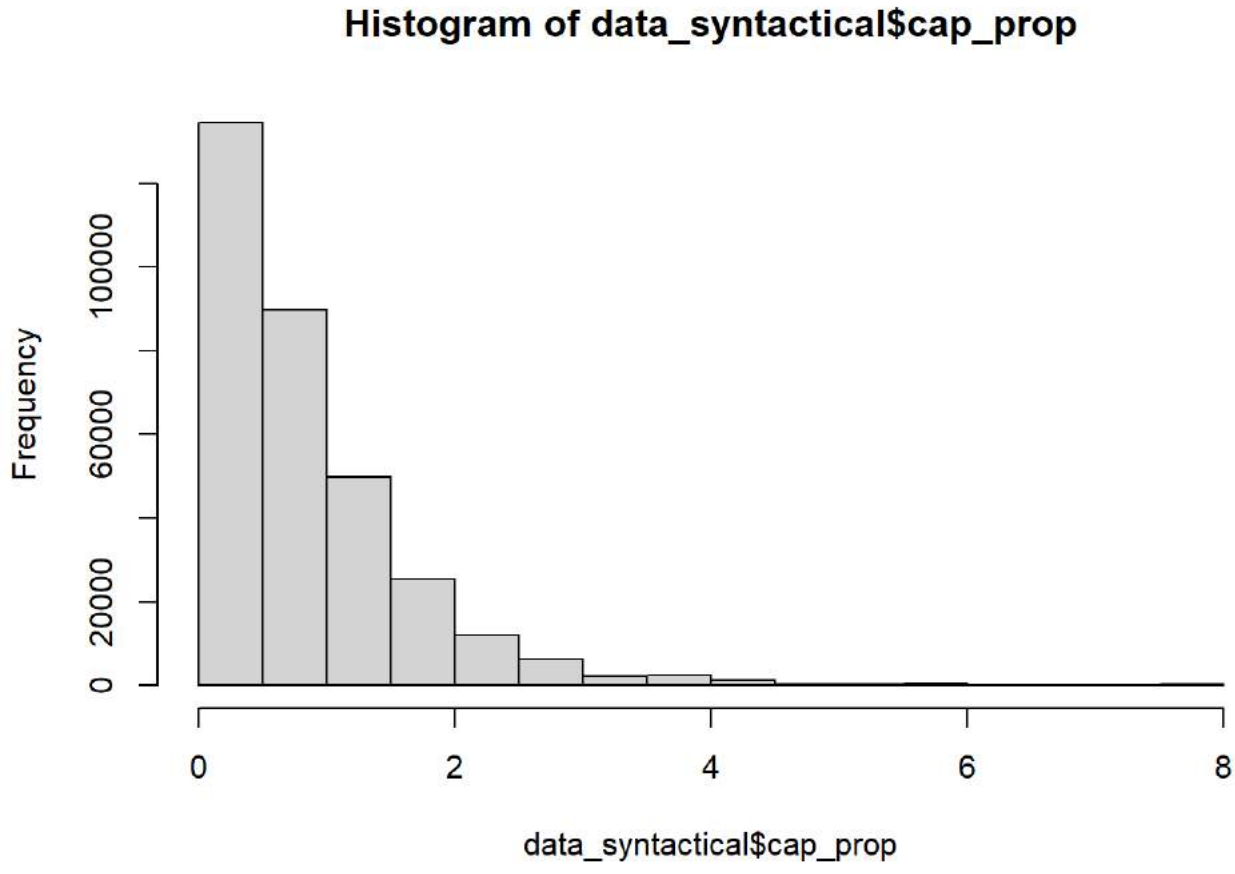
```
# Plot the regression result (total_excl_count vs. price/rating)
gr_excl_price = data_syntactical %>%
  dplyr::select(total_excl_count,price) %>%
  na.omit() %>%
  ggplot(aes(x=total_excl_count,y=price))  + geom_smooth(method="lm") + geom_point(si
ze = 2, shape=1,alpha=0.1)

gr_excl_rating= data_syntactical %>%
  dplyr::select(total_excl_count,review_scores_rating) %>%
  na.omit() %>%
  ggplot(aes(x=total_excl_count,y=review_scores_rating))  + geom_smooth(method="lm")
 + geom_point(size = 2, shape=1,alpha=0.1)

grid.arrange(gr_excl_price, gr_excl_rating,
          ncol = 2, nrow = 1)
```




```
# Calculate the number and proportion of capital letters in each listing
data_syntactical %>%
  mutate(cap_count = str_count(comments, "[[:upper:]]")) %>%
  group_by(listing_id) %>%
  mutate(total_cap_count = sum(cap_count),
         cap_prop = total_cap_count/listings_length_chars) -> data_syntactical

# Check the distribution of the number and proportion of capital letters
hist(data_syntactical$total_cap_count)
```



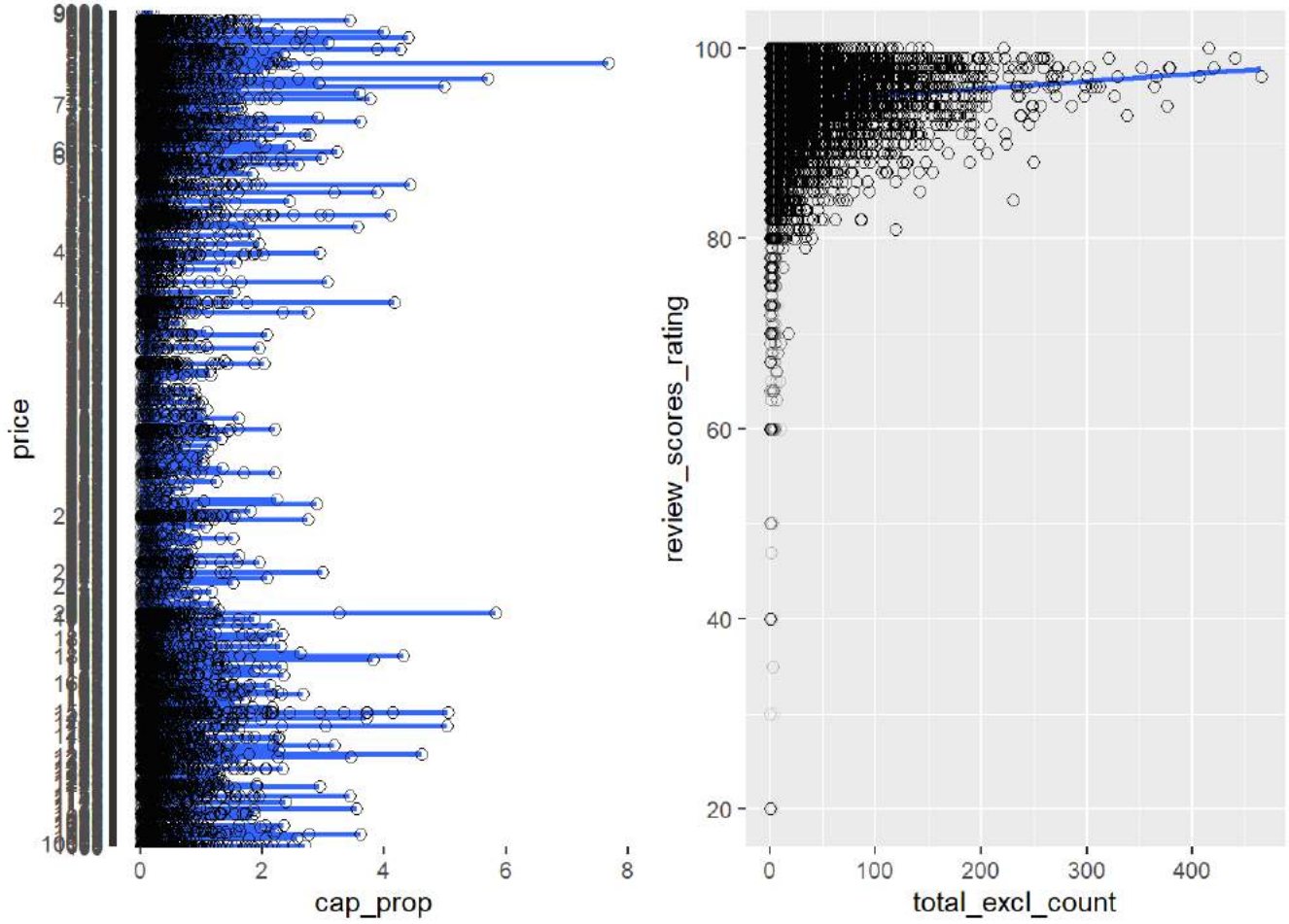

```
hist(data_syntactical$cap_prop)
```



```
# Run regression model Price/Rating against number of capital letters
model11 <- lm(log(as.numeric(price))~total_cap_count,
              data=data_syntactical)

model12 <- lm(log(review_scores_rating)~total_cap_count,
              data = data_syntactical)

model13 <- lm(log(as.numeric(price))~cap_prop,
            data=data_syntactical)

model14 <- lm(log(review_scores_rating)~cap_prop,
              data = data_syntactical)

stargazer::stargazer(model11,model12,model13,model14,type = "text")
```

```
##
## ================================================================================
================================================
##                                                                  Dependent variabl
e:
##                      ---------------------------------------------------------
-----------------------------------------------
##                       log(as.numeric(price))     log(review_scores_rating)     log
(as.numeric(price))    log(review_scores_rating)
##                                 (1)                          (2)
(3)                          (4)
## ------------------------------------------------------------------------------
-----------------------------------------------
## total_cap_count            0.00000***                  0.00001***
##                             (0.00000)                   (0.00000)
##
## cap_prop
-0.009***                   0.003***
##
(0.001)                    (0.0001)
##
## Constant                     4.634***                    4.547***
4.645***                   4.548***
##                               (0.002)                    (0.0001)
(0.002)                    (0.0001)
##
## ------------------------------------------------------------------------------
-----------------------------------------------
## Observations                  324,577                     324,570
324,577                     324,570
## R2                            0.00002                      0.005
0.0002                     0.002
## Adjusted R2                   0.00002                      0.005
0.0002                     0.002
## Residual Std. Error    0.604 (df = 324575)         0.047 (df = 324568)         0.6
03 (df = 324575)         0.047 (df = 324568)
## F Statistic         7.292*** (df = 1; 324575) 1,711.150*** (df = 1; 324568) 50.502
*** (df = 1; 324575) 800.940*** (df = 1; 324568)
## ================================================================================
================================================
## Note:
*p<0.1; **p<0.05; ***p<0.01
```

```
# Plot the regression results
gr_cap_price = data_syntactical %>%
  dplyr::select(cap_prop,price) %>%
  na.omit() %>%
  ggplot(aes(x=cap_prop,y=price))  + geom_smooth(method="lm") + geom_point(size = 2,
 shape=1,alpha=0.1)

gr_cap_rating= data_syntactical %>%
  dplyr::select(cap_prop,review_scores_rating) %>%
  na.omit() %>%
  ggplot(aes(x=cap_prop,y=review_scores_rating))  + geom_smooth(method="lm") + geom_p
oint(size = 2, shape=1,alpha=0.1)

grid.arrange(gr_cap_price, gr_excl_rating,
          ncol = 2, nrow = 1)
```

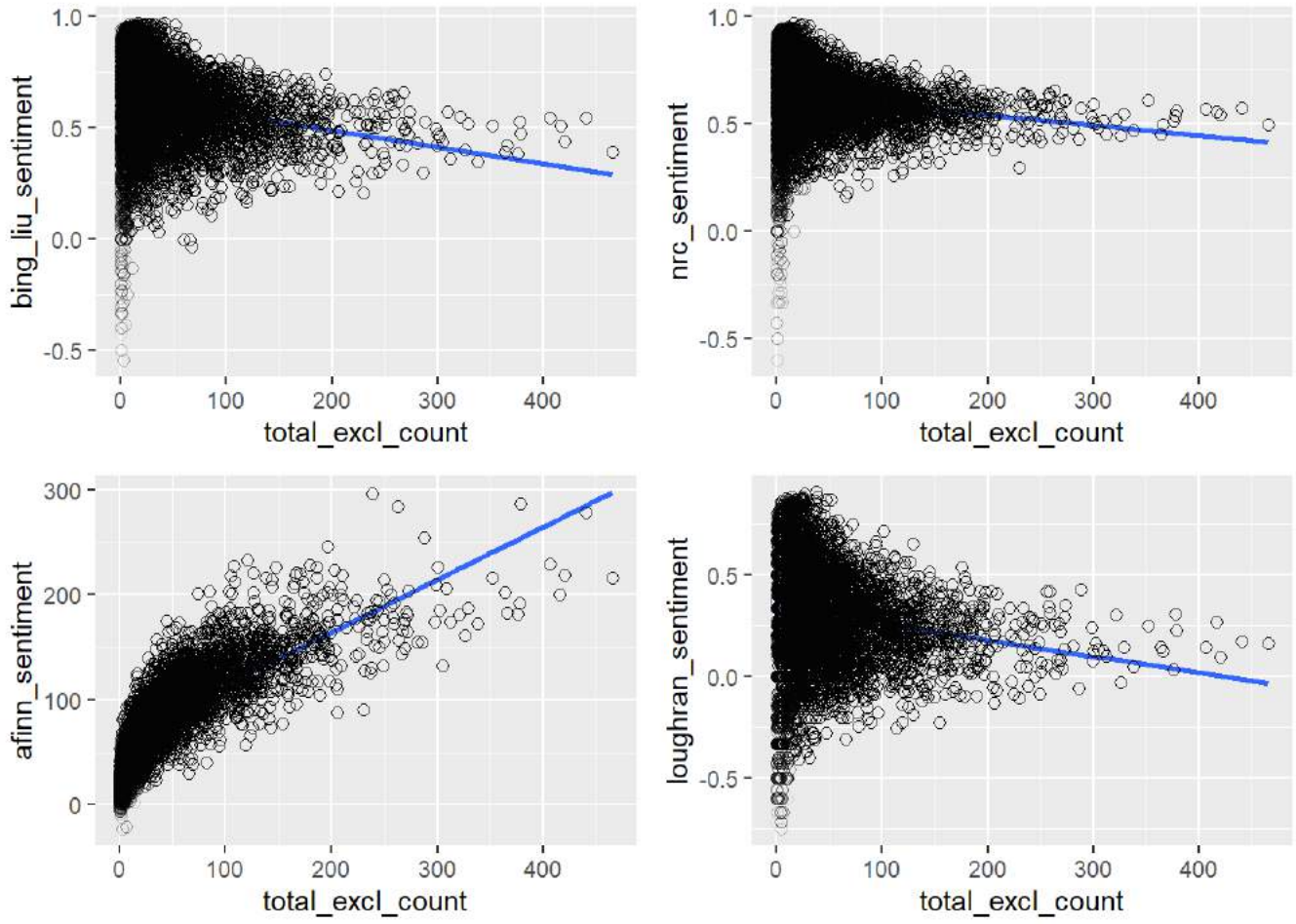



```
# Check the relation between syntactical features and sentiments
all_sentiments<- data_syntactical%>%
  group_by(listing_id) %>%
  summarise(total_excl_count= sum(excl_count),
            cap_prop = total_cap_count/listings_length_chars) %>%
  na.omit()%>%
  right_join(all_sentiments)
```

```
# Run regression model total against sentiment
model_excl_sentiment_bing <- lm(bing_liu_sentiment~total_excl_count,
             data = all_sentiments)

model_excl_sentiment_nrc <- lm(nrc_sentiment~total_excl_count,
             data = all_sentiments)

model_excl_sentiment_afinn <- lm(afinn_sentiment~total_excl_count,
             data = all_sentiments)

model_excl_sentiment_loughran <- lm(loughran_sentiment~total_excl_count,
             data = all_sentiments)

stargazer::stargazer(model_excl_sentiment_bing,model_excl_sentiment_nrc, model_excl_s
entiment_afinn, model_excl_sentiment_loughran,type = "text")
```

```
##
## ================================================================================
==================
##                                                   Dependent variable:
##                                 ---------------------------------------------
------------------
##                                 bing_liu_sentiment nrc_sentiment afinn_sentiment
loughran_sentiment
##                                        (1)              (2)            (3)
(4)
## --------------------------------------------------------------------------------
------------------
## total_excl_count                    -0.001***       -0.0005***      0.502***
-0.001***
##                                    (0.00000)        (0.00000)       (0.001)
(0.00001)
##
## Constant                            0.637***        0.636***       63.657***
0.336***
##                                     (0.0004)        (0.0003)        (0.074)

(0.001)
##
## --------------------------------------------------------------------------------
------------------
## Observations                        306,681          306,681        306,681
306,681
## R2                                   0.112            0.082          0.643
0.070
## Adjusted R2                          0.112            0.082          0.643
0.070
## Residual Std. Error (df = 306679)    0.155            0.118          27.550
0.213
## F Statistic (df = 1; 306679)      38,787.260***    27,296.980*** 551,449.100***
23,134.420***
## ================================================================================
==================
## Note:                                                               *p<0.1;
**p<0.05; ***p<0.01
```

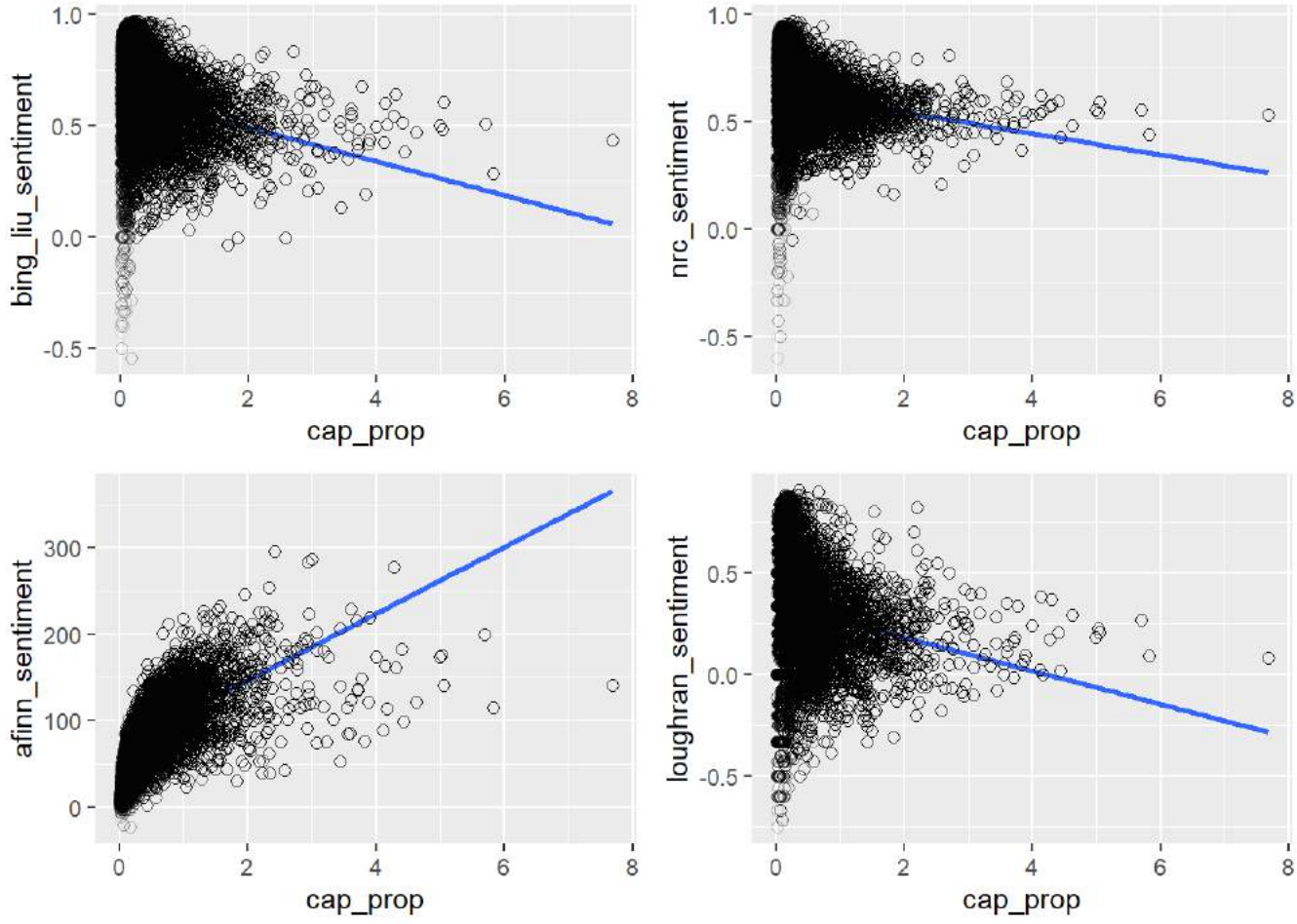
```
# Plot the regression against sentiment
gr_excl_bing = all_sentiments %>%
  dplyr::select(total_excl_count,bing_liu_sentiment) %>%
  na.omit() %>%
  ggplot(aes(x=total_excl_count,y=bing_liu_sentiment))  + geom_smooth(method="lm") +
 geom_point(size = 2, shape=1,alpha=0.1)

gr_excl_nrc = all_sentiments %>%
  dplyr::select(total_excl_count,nrc_sentiment) %>%
  na.omit() %>%
  ggplot(aes(x=total_excl_count,y=nrc_sentiment))  + geom_smooth(method="lm") + geom_
point(size = 2, shape=1,alpha=0.1)

gr_excl_afinn = all_sentiments %>%
  dplyr::select(total_excl_count,afinn_sentiment) %>%
  na.omit() %>%
  ggplot(aes(x=total_excl_count,y=afinn_sentiment))  + geom_smooth(method="lm") + geo
m_point(size = 2, shape=1,alpha=0.1)

gr_excl_loughran = all_sentiments %>%
  dplyr::select(total_excl_count,loughran_sentiment) %>%
  na.omit() %>%
  ggplot(aes(x=total_excl_count,y=loughran_sentiment))  + geom_smooth(method="lm") +
 geom_point(size = 2, shape=1,alpha=0.1)

grid.arrange(gr_excl_bing, gr_excl_nrc,gr_excl_afinn,gr_excl_loughran,
          ncol = 2, nrow = 2)
```



```
# Run regression model total against sentiment
model_cap_sentiment_bing <- lm(bing_liu_sentiment~cap_prop,
             data = all_sentiments)

model_cap_sentiment_nrc <- lm(nrc_sentiment~cap_prop,
             data = all_sentiments)

model_cap_sentiment_afinn <- lm(afinn_sentiment~cap_prop,
             data = all_sentiments)

model_cap_sentiment_loughran <- lm(loughran_sentiment~cap_prop,
             data = all_sentiments)

stargazer::stargazer(model_cap_sentiment_bing,model_cap_sentiment_nrc, model_cap_sent
iment_afinn, model_cap_sentiment_loughran,type = "text")
```

```
##
## ================================================================================
==================
##                                                   Dependent variable:
##                                 ---------------------------------------------
------------------
##                                 bing_liu_sentiment nrc_sentiment afinn_sentiment
loughran_sentiment
##                                        (1)              (2)            (3)
(4)
## --------------------------------------------------------------------------------
------------------
## cap_prop                             -0.076***        -0.049***      38.569***
-0.082***
##                                       (0.0003)        (0.0003)        (0.078)
(0.0005)
##
## Constant                              0.645***        0.641***       69.625***
0.345***
##                                       (0.0004)        (0.0003)        (0.093)

(0.001)
##
## --------------------------------------------------------------------------------
------------------
## Observations                         306,681          306,681        306,681
306,681
## R2                                    0.137            0.102          0.446
0.088
## Adjusted R2                           0.137            0.102          0.446
0.088
## Residual Std. Error (df = 306679)     0.153            0.117          34.297
0.211
## F Statistic (df = 1; 306679)        48,660.140***    34,711.050*** 247,032.300***
29,456.140***
## ================================================================================
==================
## Note:                                                                *p<0.1;
**p<0.05; ***p<0.01
```

```
gr_cap_bing = all_sentiments %>%
  dplyr::select(cap_prop,bing_liu_sentiment) %>%
  na.omit() %>%
  ggplot(aes(x=cap_prop,y=bing_liu_sentiment))  + geom_smooth(method="lm") + geom_poi
nt(size = 2, shape=1,alpha=0.1)

gr_cap_nrc = all_sentiments %>%
  dplyr::select(cap_prop,nrc_sentiment) %>%
  na.omit() %>%
  ggplot(aes(x=cap_prop,y=nrc_sentiment))  + geom_smooth(method="lm") + geom_point(si
ze = 2, shape=1,alpha=0.1)

gr_cap_afinn = all_sentiments %>%
  dplyr::select(cap_prop,afinn_sentiment) %>%
  na.omit() %>%
  ggplot(aes(x=cap_prop,y=afinn_sentiment))  + geom_smooth(method="lm") + geom_point
(size = 2, shape=1,alpha=0.1)

gr_cap_loughran = all_sentiments %>%
  dplyr::select(cap_prop,loughran_sentiment) %>%
  na.omit() %>%
  ggplot(aes(x=cap_prop,y=loughran_sentiment))  + geom_smooth(method="lm") + geom_poi
nt(size = 2, shape=1,alpha=0.1)

grid.arrange(gr_cap_bing, gr_cap_nrc,gr_cap_afinn,gr_cap_loughran,
          ncol = 2, nrow = 2)
```

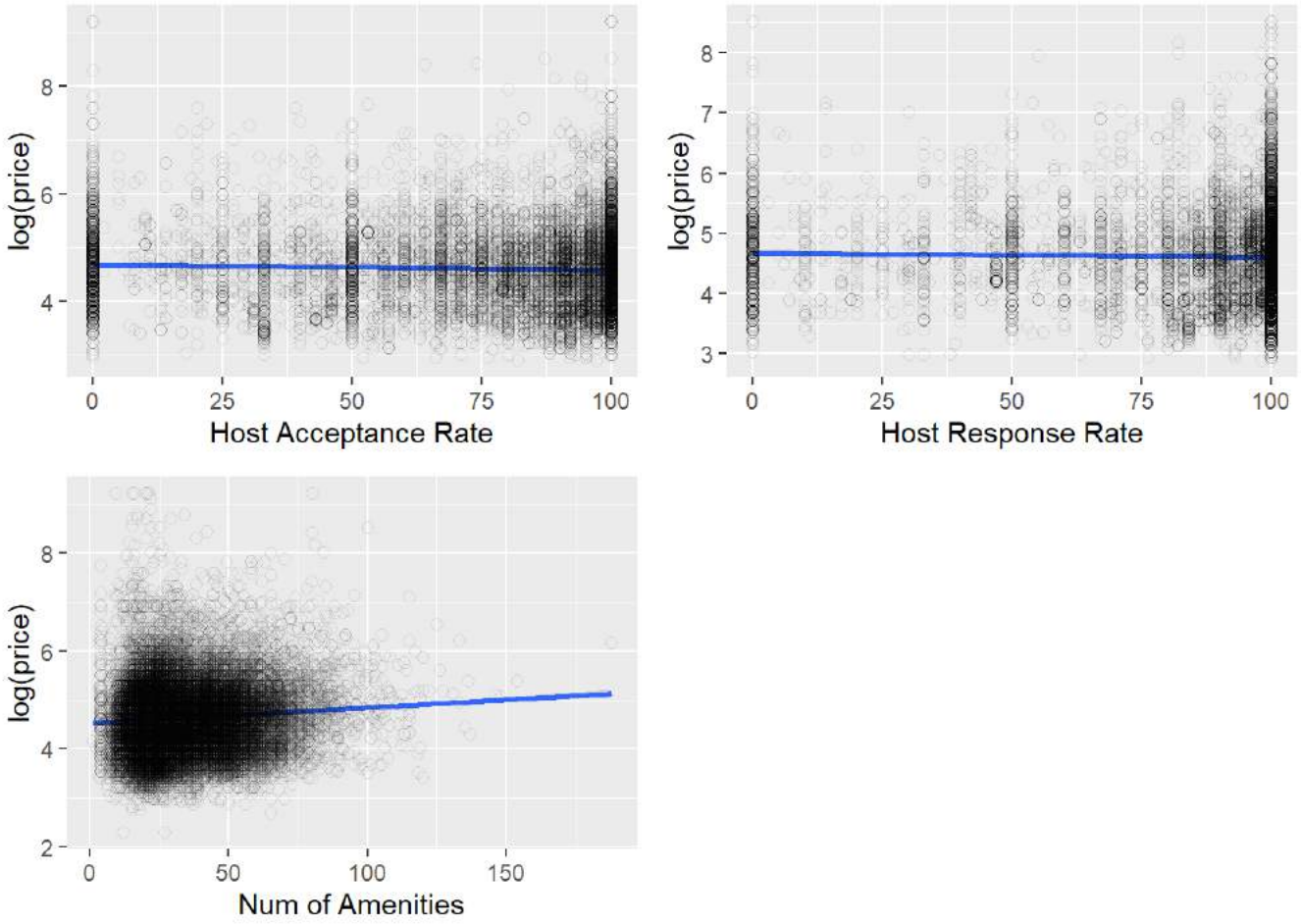



## Other variables against prices

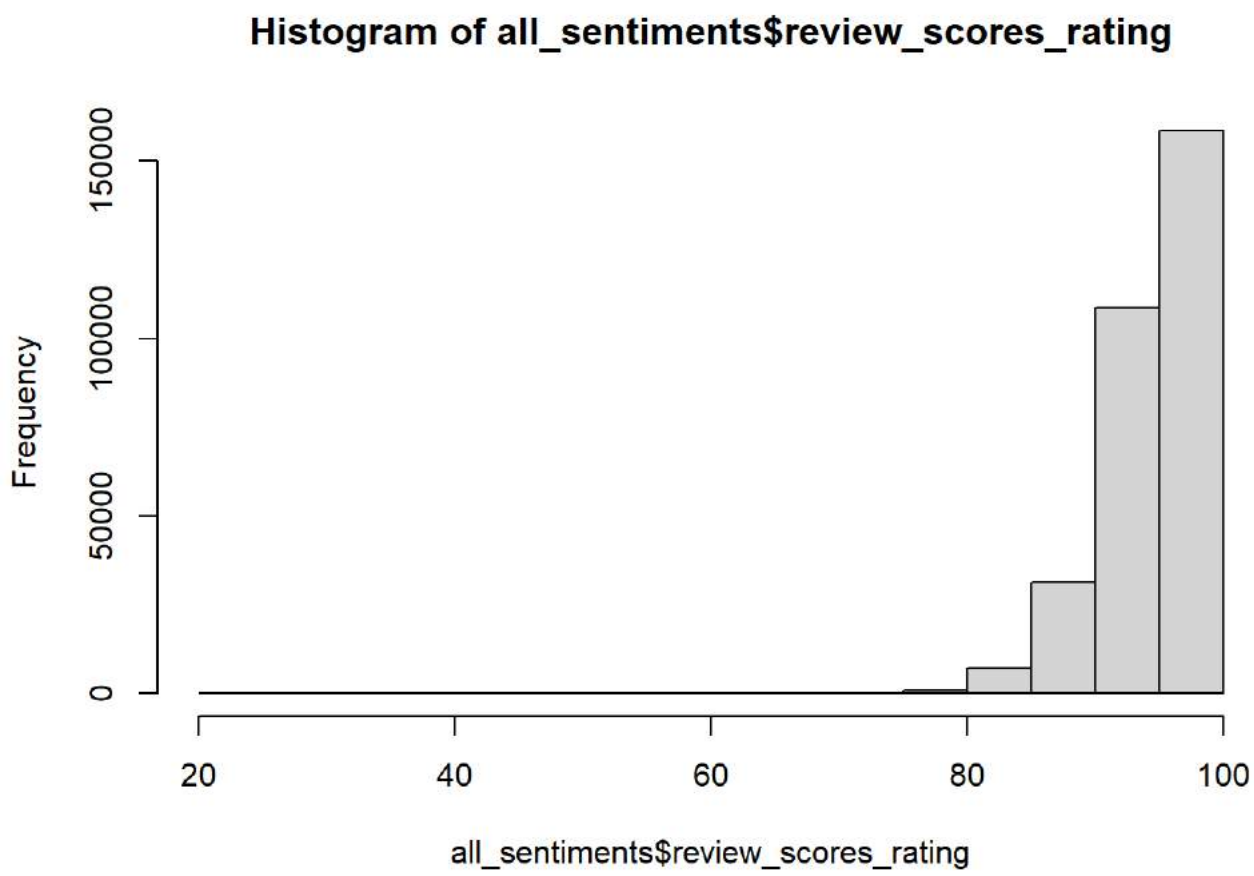
```
#Try models of Price against host_response_rate, host_acceptance_rate and num_of_amen
ities

#Extract host_response_rate, host_acceptance_rate and num_of_amenities and have one d
f with all these variables

model_data <- listings_reviews %>%
  dplyr::select(listing_id,price, review_scores_rating, host_acceptance_rate, host_re
sponse_rate, num_amenities)

model_data$host_acceptance_rate = as.numeric(sub("%","",model_data$host_acceptance_ra
te))
model_data$host_response_rate = as.numeric(sub("%","",model_data$host_response_rate))
model_data$price = as.numeric(model_data$price)

#Models (other variables against pricess)

model15 <- lm(log(model_data$price)~model_data$host_acceptance_rate)
model16 <- lm(log(model_data$price)~model_data$host_response_rate)
model17 <- lm(log(model_data$price)~model_data$num_amenities)


stargazer::stargazer(model15,model16,model17, type = "text")
```

```
##
## ================================================================================
===============
##                                              Dependent variable:
##                     ---------------------------------------------------------
---------------
##                                                    price)
##                                (1)                   (2)
(3)
## --------------------------------------------------------------------------------
---------------
## host_acceptance_rate         -0.001***
##                              (0.0002)
##
## host_response_rate                                 -0.001**
##                                                    (0.0003)
##
## num_amenities                                                                0.
003***
##
(0.0002)
##
## Constant                     4.681***              4.675***                  4.
526***
##                               (0.017)               (0.024)
(0.010)
##
## --------------------------------------------------------------------------------
---------------
## Observations                  12,492                11,503                    2
5,933
## R2                             0.002                 0.001
0.007
## Adjusted R2                    0.002                 0.0005
0.007
## Residual Std. Error     0.721 (df = 12490)       0.734 (df = 11501)        0.699
(df = 25931)
## F Statistic          24.139*** (df = 1; 12490) 6.482** (df = 1; 11501) 172.856***
(df = 1; 25931)
## ================================================================================
===============
## Note:                                                             *p<0.1; **p<
0.05; ***p<0.01
```

```
gr_har = model_data %>%
  dplyr::select(host_acceptance_rate,price) %>%
  na.omit() %>%
  ggplot(aes(x=host_acceptance_rate,y=log(price)))  + geom_smooth(method="lm") + geom
_point(size = 2, shape=1,alpha=0.1) + xlab("Host Acceptance Rate")

gr_hrr = model_data  %>%
  dplyr::select(host_response_rate,price) %>%
  na.omit() %>%
  ggplot(aes(x=host_response_rate,y=log(price)))  + geom_smooth(method="lm") + geom_p
oint(size = 2, shape=1,alpha=0.1) + xlab("Host Response Rate")

gr_na = model_data  %>%
  dplyr::select(num_amenities,price) %>%
  na.omit() %>%
  ggplot(aes(x=num_amenities,y=log(price)))  + geom_smooth(method="lm") + geom_point
(size = 2, shape=1,alpha=0.1) + xlab("Num of Amenities")

grid.arrange(gr_har, gr_hrr, gr_na,
          ncol = 2, nrow = 2)
```



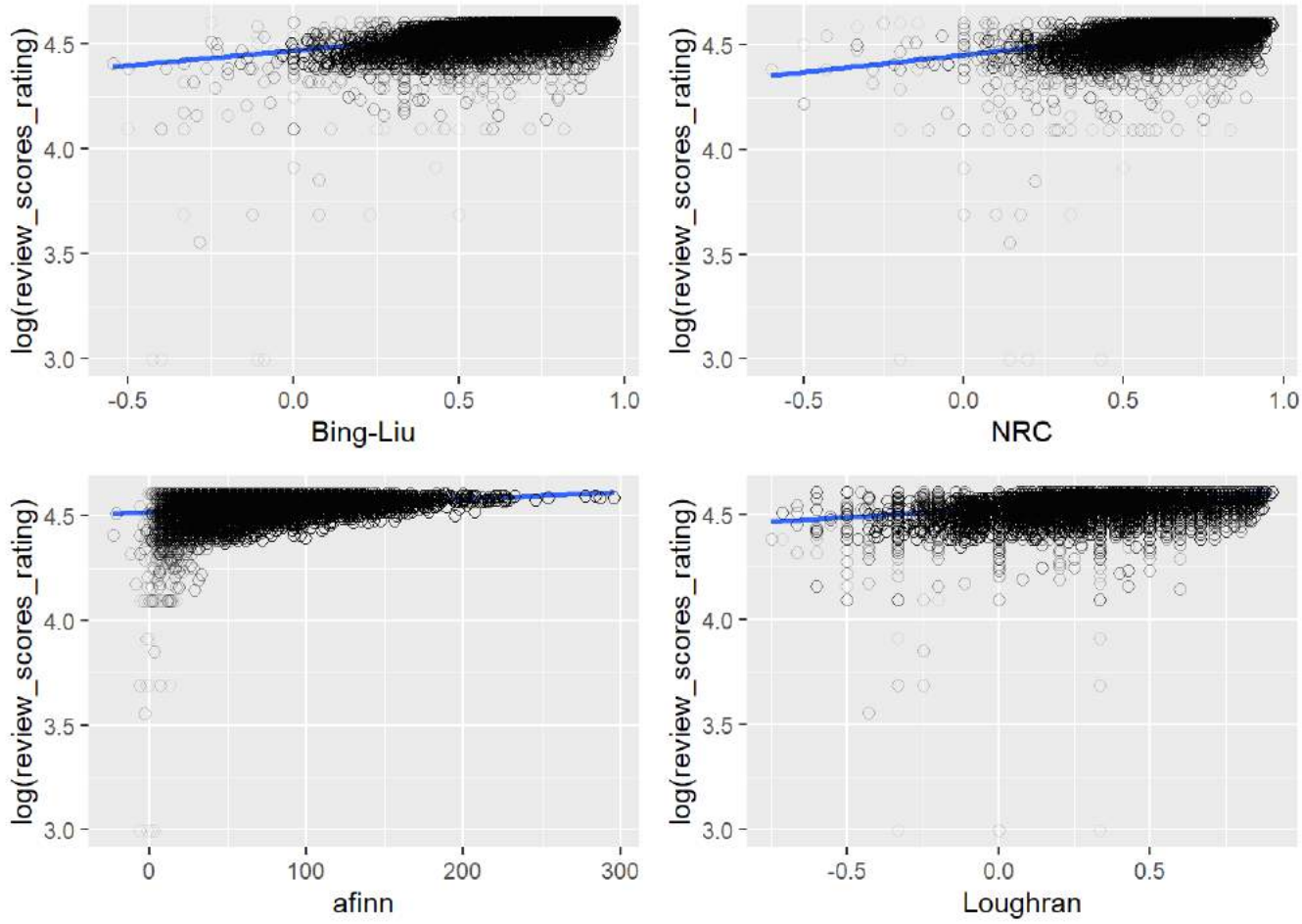

```
#Do same 7 models with review_scores_rating instead of price

hist(all_sentiments$review_scores_rating)
```




```
model18 <- lm(log(all_sentiments$review_scores_rating)~all_sentiments$bing_liu_sentim
ent)
model19 <- lm(log(all_sentiments$review_scores_rating)~all_sentiments$nrc_sentiment)
model20 <- lm(log(all_sentiments$review_scores_rating)~all_sentiments$afinn_sentimen
t)
model21 <- lm(log(all_sentiments$review_scores_rating)~all_sentiments$loughran_sentim
ent)


stargazer::stargazer(model18,model19,model20,model21, type = "text")
```

```
##
## ================================================================================
========
##                                                 Dependent variable:
##                                     ---------------------------------------------
--------
##                                                review_scores_rating)
##                                       (1)            (2)           (3)
(4)
## --------------------------------------------------------------------------------
--------
## bing_liu_sentiment                  0.142***
##                                     (0.0004)
##
## nrc_sentiment                                      0.165***
##                                                     (0.001)
##
## afinn_sentiment                                                  0.0003***
##                                                                  (0.00000)
##
## loughran_sentiment                                                              0.0
83***
##                                                                                 (0.
0003)
##
## Constant                            4.469***      4.452***      4.518***      4.5
28***
##                                     (0.0002)      (0.0003)      (0.0002)      (0.
0001)
##
## --------------------------------------------------------------------------------
--------
## Observations                        306,681        306,681       306,681       30
6,681
## R2                                   0.302          0.228         0.116
0.183
## Adjusted R2                          0.302          0.228         0.116
0.183
## Residual Std. Error (df = 306679)    0.036          0.037         0.040
0.038
## F Statistic (df = 1; 306679)      132,629.200*** 90,325.490*** 40,354.100*** 68,77
0.100***
## ================================================================================
========
## Note:                                                          *p<0.1; **p<0.05; *
**p<0.01
```

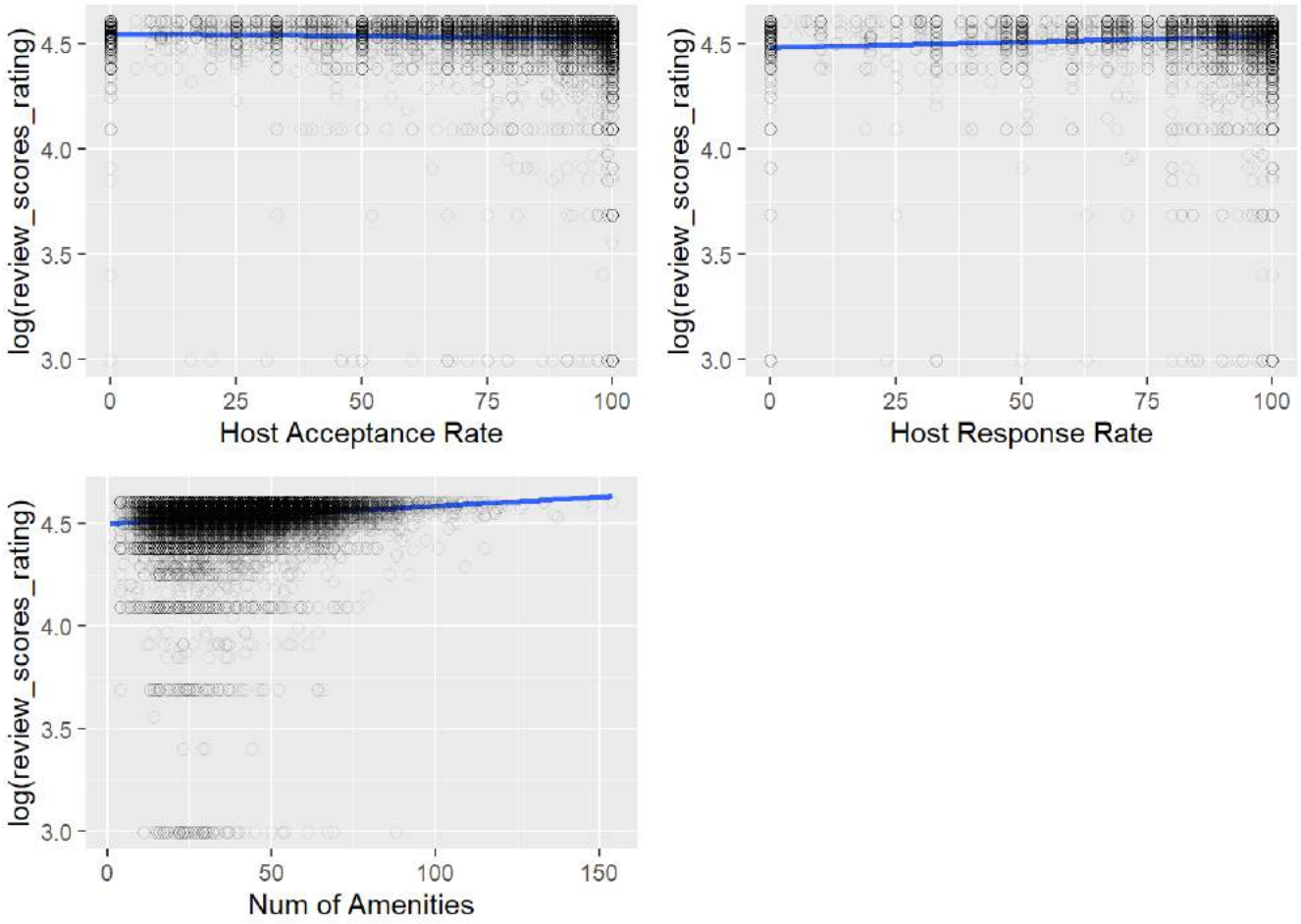
```
gr_bl_1 = all_sentiments %>%
  dplyr::select(bing_liu_sentiment,review_scores_rating) %>%
  na.omit() %>%
  ggplot(aes(x=bing_liu_sentiment,y=log(review_scores_rating)))  + geom_smooth(method
="lm") + geom_point(size = 2, shape=1,alpha=0.1) + xlab("Bing-Liu")

gr_nrc_1 = all_sentiments %>%
  dplyr::select(nrc_sentiment,review_scores_rating) %>%
  na.omit() %>%
  ggplot(aes(x=nrc_sentiment,y=log(review_scores_rating)))  + geom_smooth(method="lm"
) + geom_point(size = 2, shape=1,alpha=0.1) + xlab("NRC")

gr_afinn_1 = all_sentiments %>%
  dplyr::select(afinn_sentiment,review_scores_rating) %>%
  na.omit() %>%
  ggplot(aes(x=afinn_sentiment,y=log(review_scores_rating)))  + geom_smooth(method="l
m") + geom_point(size = 2, shape=1,alpha=0.1) + xlab("afinn")

gr_loughran_1 = all_sentiments %>%
  dplyr::select(loughran_sentiment,review_scores_rating) %>%
  na.omit() %>%
  ggplot(aes(x=loughran_sentiment,y=log(review_scores_rating))) + geom_smooth(method=
"lm") + geom_point(size = 2, shape=1,alpha=0.1) + xlab("Loughran")

grid.arrange(gr_bl_1, gr_nrc_1, gr_afinn_1, gr_loughran_1,
          ncol = 2, nrow = 2)
```



```
#Models with host_acceptance_rate, host_response_rate, num_amenities

model22 <- lm(log(model_data$review_scores_rating)~model_data$host_acceptance_rate)
model23 <- lm(log(model_data$review_scores_rating)~model_data$host_response_rate)
model24 <- lm(log(model_data$review_scores_rating)~model_data$num_amenities)


stargazer::stargazer(model22,model23,model24, type = "text")
```

```
##
## ===============================================================================
===============
##                                          Dependent variable:
##                     ---------------------------------------------------------
---------------
##                                         review_scores_rating)
##                              (1)                    (2)
(3)
## -------------------------------------------------------------------------------
---------------
## host_acceptance_rate        -0.0002***
##                             (0.00005)
##
## host_response_rate                                0.001***
##                                                   (0.0001)
##
## num_amenities                                                                0.
001***
##
(0.0001)
##
## Constant                     4.548***                4.482***                4.
498***
##                              (0.004)                 (0.006)
(0.003)
##
## -------------------------------------------------------------------------------
---------------
## Observations                  9,768                   8,734                    1
8,905
## R2                            0.002                   0.007
0.011
## Adjusted R2                   0.002                   0.007

0.011
## Residual Std. Error     0.151 (df = 9766)       0.150 (df = 8732)         0.149
(df = 18903)
## F Statistic          21.746*** (df = 1; 9766) 63.261*** (df = 1; 8732) 209.961***
(df = 1; 18903)
## ===============================================================================
===============
## Note:                                                              *p<0.1; **p<
0.05; ***p<0.01
```

```
gr_har_1 = model_data %>%
  dplyr::select(host_acceptance_rate,review_scores_rating) %>%
  na.omit() %>%
  ggplot(aes(x=host_acceptance_rate,y=log(review_scores_rating)))+ geom_smooth(method
="lm") + geom_point(size = 2, shape=1,alpha=0.1) + xlab("Host Acceptance Rate")

gr_hrr_1 = model_data  %>%
  dplyr::select(host_response_rate,review_scores_rating) %>%
  na.omit() %>%
  ggplot(aes(x=host_response_rate,y=log(review_scores_rating)))  + geom_smooth(method
="lm") + geom_point(size = 2, shape=1,alpha=0.1) + xlab("Host Response Rate")

gr_na_1 = model_data  %>%
  dplyr::select(num_amenities,review_scores_rating) %>%
  na.omit() %>%
  ggplot(aes(x=num_amenities,y=log(review_scores_rating)))  + geom_smooth(method="lm"
) + geom_point(size = 2, shape=1,alpha=0.1) + xlab("Num of Amenities")

grid.arrange(gr_har_1, gr_hrr_1, gr_na_1,
          ncol = 2, nrow = 2)
```



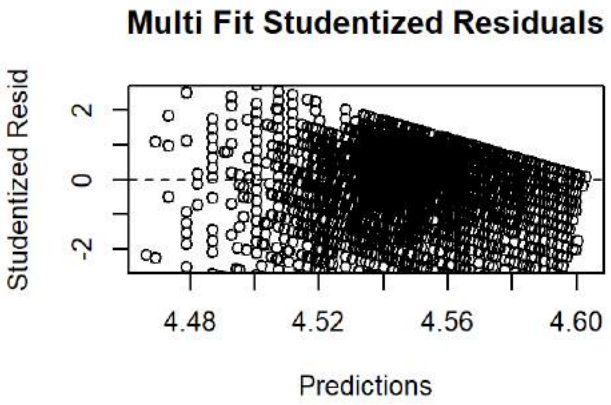

```
#Multiple regressions for price and review_score using different dictionaries
model_data_join = model_data %>%
  dplyr::select(listing_id, host_acceptance_rate, host_response_rate, num_amenities)

mult_reg_data = left_join(all_sentiments, model_data_join, by = "listing_id")
```

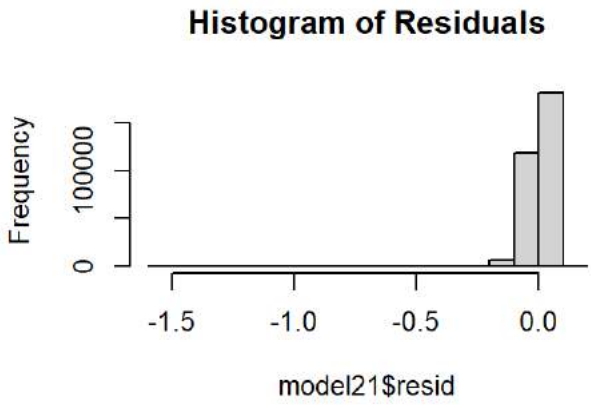
```
#Multiple regressions for price
model25 = lm(log(price)~bing_liu_sentiment + host_acceptance_rate + host_response_rat
e + num_amenities + total_excl_count + cap_prop, data = mult_reg_data)
model26 = lm(log(price)~nrc_sentiment + host_acceptance_rate + host_response_rate + n
um_amenities + total_excl_count + cap_prop, data = mult_reg_data)
model27 = lm(log(price)~afinn_sentiment + host_acceptance_rate + host_response_rate +
num_amenities + total_excl_count + cap_prop, data = mult_reg_data)
model28 = lm(log(price)~loughran_sentiment + host_acceptance_rate + host_response_rat
e + num_amenities + total_excl_count + cap_prop, data = mult_reg_data)

stargazer::stargazer(model25,model26,model27,model28, type = "text")
```

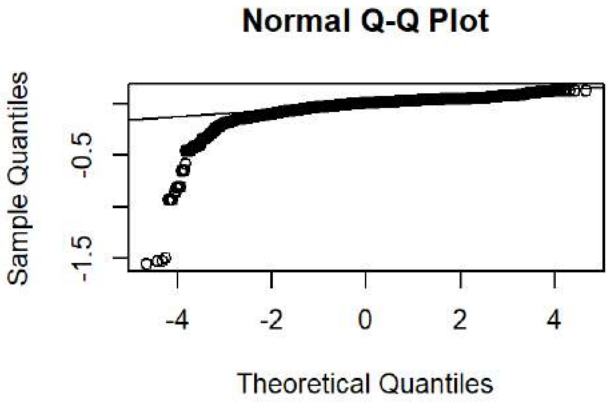
```
##
## ================================================================================
===
##                                          Dependent variable:
##                                 ----------------------------------------------
---
##                                              log(price)
##                                   (1)          (2)          (3)          (4)
## --------------------------------------------------------------------------------
---
## bing_liu_sentiment              0.590***
##                                 (0.009)
##
## nrc_sentiment                                0.490***
##                                              (0.013)
##
## afinn_sentiment                                           0.005***
##                                                           (0.0001)
##
## loughran_sentiment                                                     0.465**
*
##                                                                        (0.007)
##
## host_acceptance_rate            -0.002***    -0.002***    -0.002***    -0.002**
*
##                                 (0.0001)     (0.0001)     (0.0001)     (0.000
1)
##
## host_response_rate              0.001***     0.001***     0.001***     0.001**
*
##                                 (0.0001)     (0.0001)     (0.0001)     (0.000
1)
##
## num_amenities                   0.005***     0.006***     0.005***     0.005**
*
##                                 (0.0001)     (0.0001)     (0.0001)     (0.000
1)
##
## total_excl_count                0.0005***    0.0004***    -0.002***    0.0004**
*
##                                (0.00003)    (0.00003)    (0.00004)    (0.0000
3)
##
## cap_prop                        0.013***      -0.003      -0.043***     0.009**
*
##                                 (0.003)      (0.003)      (0.003)      (0.003)
##
## Constant                        4.073***     4.109***     4.149***     4.283**
*
##                                 (0.010)      (0.011)      (0.008)      (0.008)
##
## --------------------------------------------------------------------------------
---
## Observations                    184,248      184,248      184,248      184,248
## R2                               0.054        0.042        0.074        0.057
## Adjusted R2                      0.054        0.042        0.074        0.057
## Residual Std. Error (df = 184241)  0.601        0.605        0.595        0.600
```

```
## F Statistic (df = 6; 184241)      1,759.919*** 1,334.829*** 2,449.172*** 1,864.061
***
## ============================================================================
===
## Note:                                                  *p<0.1; **p<0.05; ***p<
0.01
```

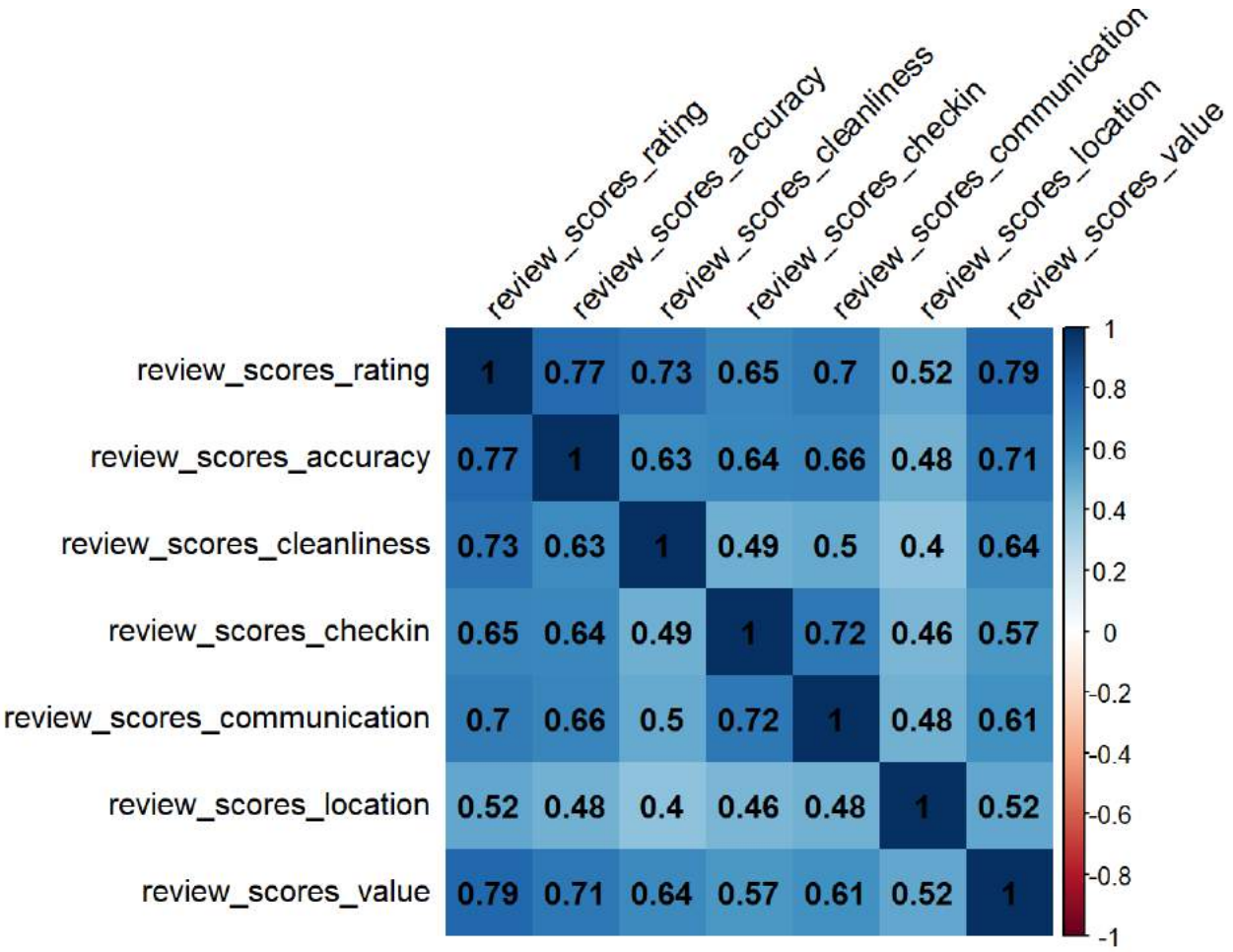
```
# Check if the assumptions are met in the best performance model
# Model 21
layout(matrix(c(1,2,3,4),2,2,byrow=T))
plot(model21$fitted, rstudent(model21),
 main="Multi Fit Studentized Residuals",
 xlab="Predictions",ylab="Studentized Resid",
 ylim=c(-2.5,2.5))
abline(h=0, lty=2)

hist(model21$resid,main="Histogram of Residuals")
qqnorm(model21$resid)
qqline(model21$resid)
```



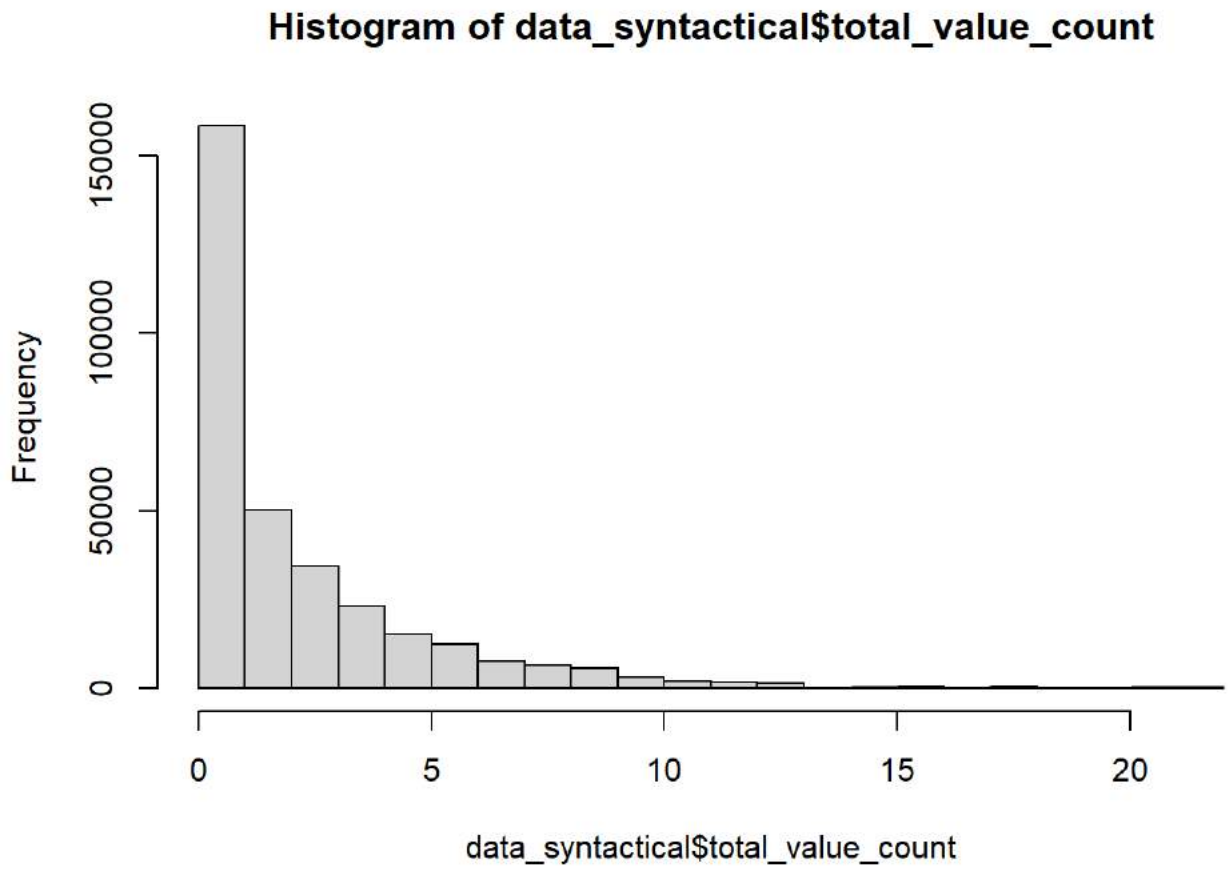



```
#Correlation matrix for score reviews
reviews_data = listings_reviews[ , grepl( "review_scores", names(listings_reviews))]
 %>% na.omit()

#install.packages("corrplot")
library(corrplot)

corrplot(cor(reviews_data), method="shade", addCoef.col = "black", tl.col="black", t
l.srt=45)
```

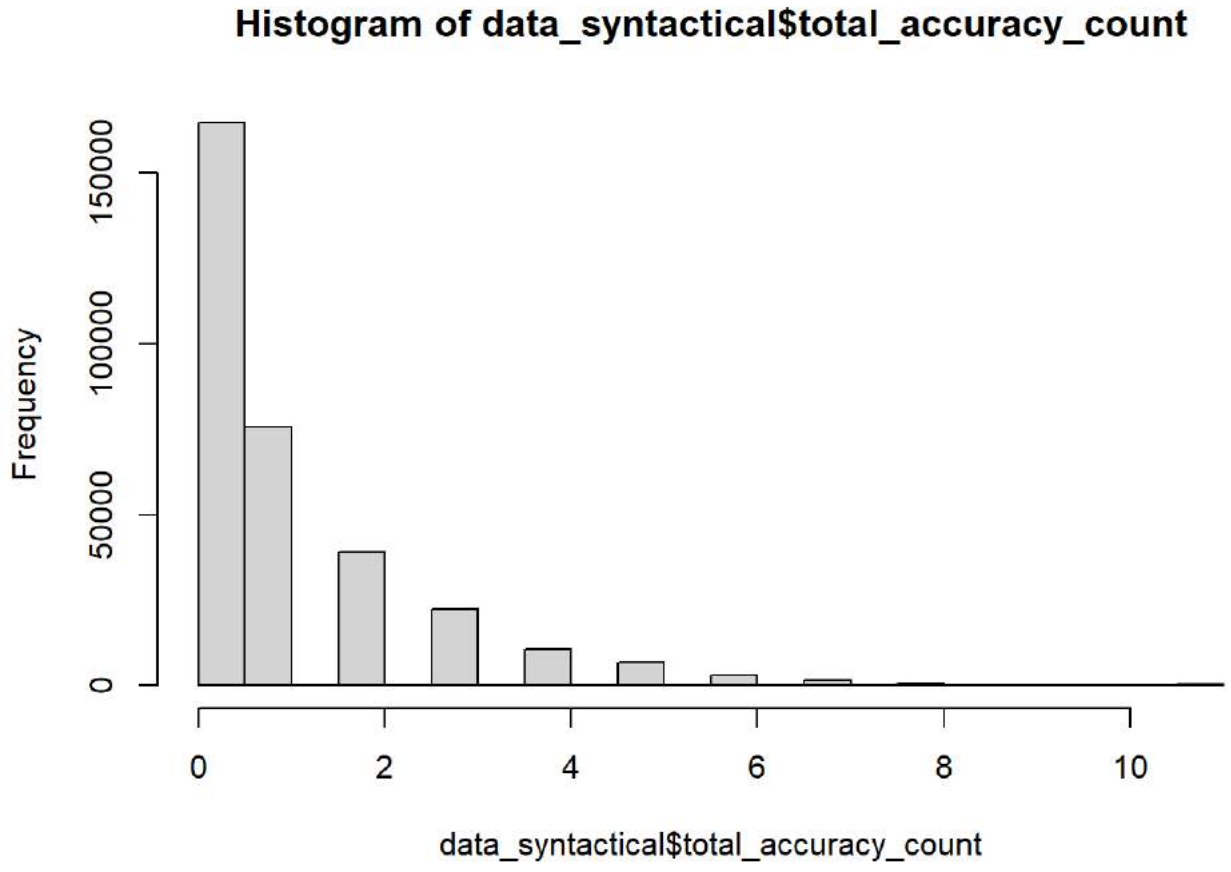



```
#Calculate the number of words indicating value or accuracy
data_syntactical %>%
  mutate(value_count = str_count(comments, "value"),
         accuracy_count = str_count(comments, "accurate")) %>%
  group_by(listing_id) %>%
  mutate(total_value_count = sum(value_count),
         total_accuracy_count = sum(accuracy_count))-> data_syntactical
```

```
# Check the distribution of total_value_count and total_accuracy_count
hist(data_syntactical$total_value_count)
```



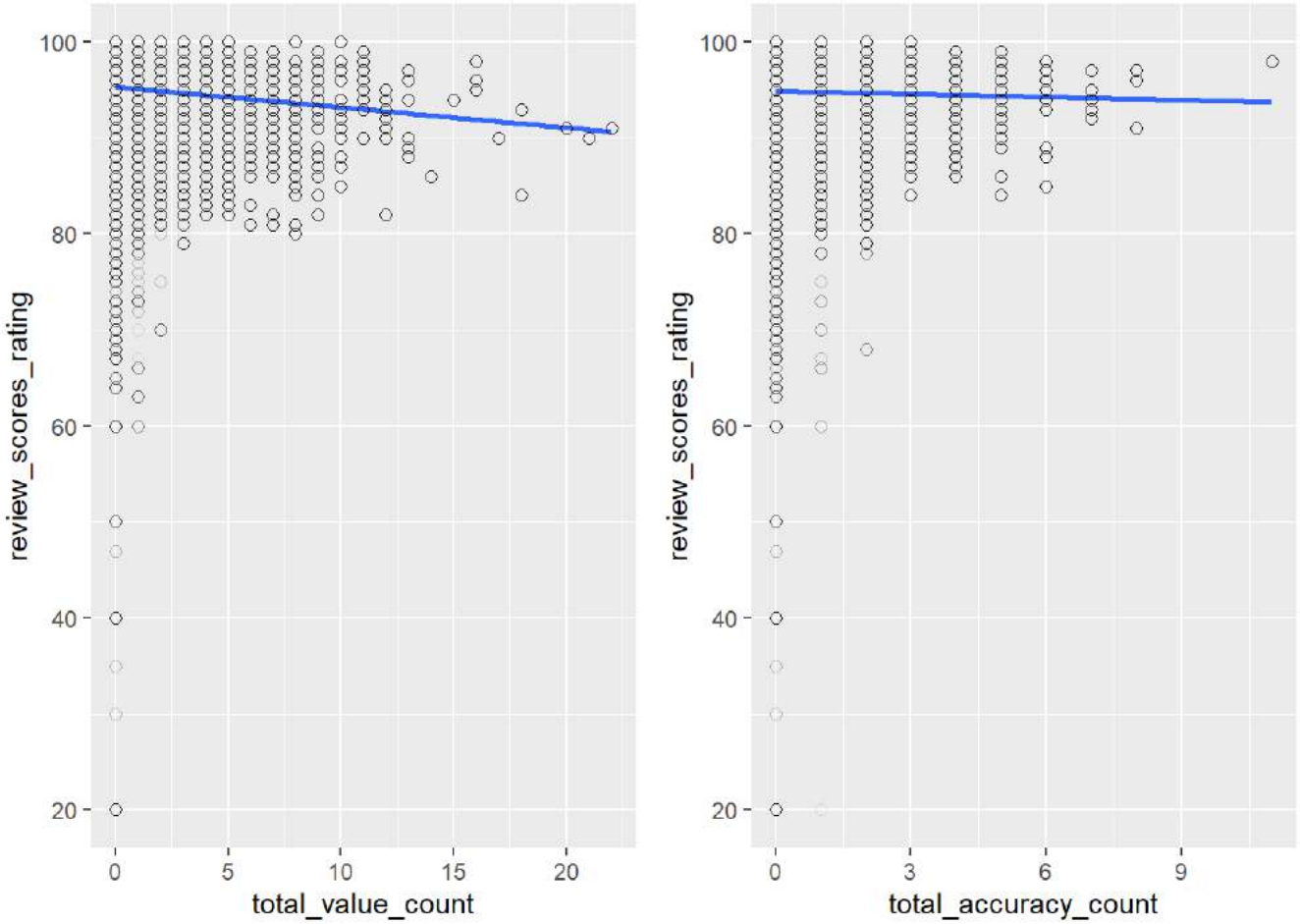

```
hist(data_syntactical$total_accuracy_count)
```



```
# Run regression model Price/Rating against total_value_count and total_accuracy_coun
t
model29 <- lm(log(review_scores_rating)~total_value_count,
              data=data_syntactical)

model30 <- lm(log(review_scores_rating)~total_accuracy_count,
              data = data_syntactical)

stargazer::stargazer(model29,model30,type = "text")
```

```
##
## ===============================================================
##                                        Dependent variable:
##                                    ----------------------------
##                                    log(review_scores_rating)
##                                         (1)           (2)
## ---------------------------------------------------------------
## total_value_count                     -0.002***
##                                       (0.00003)
##
## total_accuracy_count                                -0.001***
##                                                      (0.0001)
##
## Constant                              4.556***       4.552***
##                                       (0.0001)       (0.0001)
##
## ---------------------------------------------------------------
## Observations                           324,570       324,570
## R2                                      0.019         0.001
## Adjusted R2                             0.019         0.001
## Residual Std. Error (df = 324568)       0.046         0.047
## F Statistic (df = 1; 324568)       6,432.481***    276.340***
## ===============================================================
## Note:                                *p<0.1; **p<0.05; ***p<0.01
```

```
# Plot the regression results
gr_value_rating = data_syntactical %>%
  dplyr::select(total_value_count,review_scores_rating) %>%
  na.omit() %>%
  ggplot(aes(x=total_value_count,y=review_scores_rating))  + geom_smooth(method="lm")
+ geom_point(size = 2, shape=1,alpha=0.1)

gr_accuracy_rating= data_syntactical %>%
  dplyr::select(total_accuracy_count,review_scores_rating) %>%
  na.omit() %>%
  ggplot(aes(x=total_accuracy_count,y=review_scores_rating))  + geom_smooth(method="l
m") + geom_point(size = 2, shape=1,alpha=0.1)

grid.arrange(gr_value_rating, gr_accuracy_rating,
          ncol = 2, nrow = 1)
```



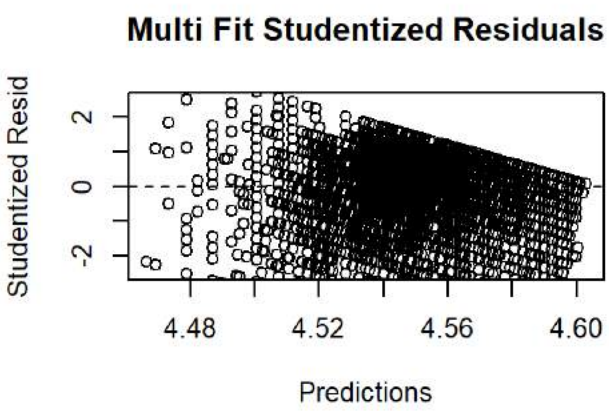

```
# Join the data
mult_reg_data <- data_syntactical%>%
  dplyr::select(listing_id,total_value_count,total_accuracy_count)%>%
  right_join(mult_reg_data)
```

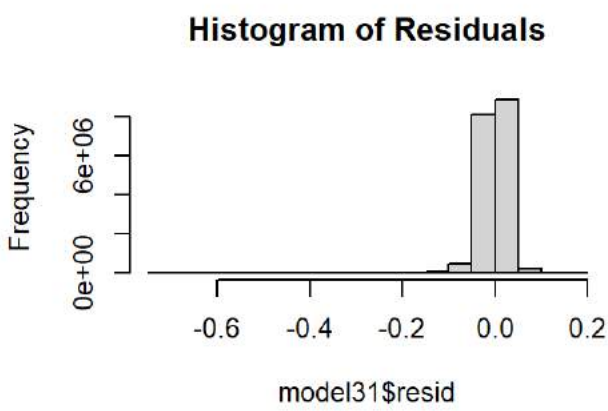
```
#Multiple regressions for rating
model31 = lm(log(review_scores_rating)~bing_liu_sentiment + host_acceptance_rate + ho
st_response_rate + num_amenities+ total_excl_count + cap_prop +total_value_count +tot
al_accuracy_count, data = mult_reg_data)
model32 = lm(log(review_scores_rating)~nrc_sentiment + host_acceptance_rate + host_re
sponse_rate + num_amenities+ total_excl_count+cap_prop+total_value_count +total_accur
acy_count, data = mult_reg_data)
model33 = lm(log(review_scores_rating)~afinn_sentiment + host_acceptance_rate + host_
response_rate + num_amenities+ total_excl_count+cap_prop+total_value_count +total_acc
uracy_count, data = mult_reg_data)
model34 = lm(log(review_scores_rating)~loughran_sentiment + host_acceptance_rate + ho
st_response_rate + num_amenities+ total_excl_count+cap_prop+total_value_count +total_
accuracy_count, data = mult_reg_data)

stargazer::stargazer(model31,model32,model33,model34, type = "text")
```

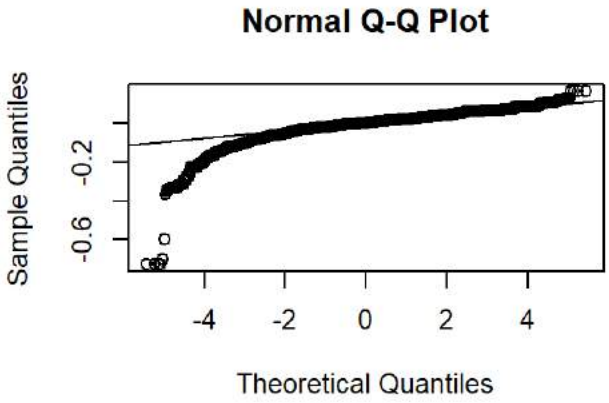
```
##
## ================================================================================
=====================
##                                                  Dependent variable:
##                                -----------------------------------------------
---------------------
##                                                 log(review_scores_rating)
##                                    (1)              (2)              (3)
(4)
## --------------------------------------------------------------------------------
---------------------
## bing_liu_sentiment                  0.178***
##                                    (0.00004)
##
## nrc_sentiment                                        0.213***
##                                                      (0.0001)
##
## afinn_sentiment                                                                0.0005***
##                                                                                (0.00000)
##
## loughran_sentiment
0.105***
##
(0.00004)
##
## host_acceptance_rate                 0.00000         0.00001***       0.00001**
*        -0.00000
##                                    (0.00000)        (0.00000)        (0.00000)
(0.00000)
##
## host_response_rate                 0.0001***        0.0001***        0.0002***
0.0001***
##                                    (0.00000)        (0.00000)        (0.00000)
(0.00000)
##
## num_amenities                      0.0002***        0.0002***        0.0002***
0.0003***
##                                    (0.00000)        (0.00000)        (0.00000)
(0.00000)
##
## total_excl_count                   0.0002***        0.0002***        0.00004**
*       0.0002***
##                                    (0.00000)        (0.00000)        (0.00000)
(0.00000)
##
## cap_prop                           -0.001***        -0.002***        -0.008***
-0.003***
##                                    (0.00001)        (0.00001)        (0.00001)
(0.00001)
##
## total_value_count                  -0.001***        -0.002***        -0.002***
-0.002***
##                                    (0.00000)        (0.00000)        (0.00000)
(0.00000)
##
## total_accuracy_count               -0.0003***       -0.001***        -0.002***
-0.001***
```

```
##                                          (0.00000)        (0.00000)        (0.00000)
(0.00000)
##
## Constant                                  4.428***         4.400***         4.483***
4.497***
##                                          (0.00004)         (0.0001)        (0.00005)
(0.00004)
##
## ------------------------------------------------------------------------------------
---------------------
## Observations                            17,655,084       17,655,084       17,655,08
4       17,655,084
## R2                                         0.611            0.523            0.369
0.477
## Adjusted R2                                0.611            0.523            0.369
0.477
## Residual Std. Error (df = 17655075)        0.023            0.025            0.029
0.026
## F Statistic (df = 8; 17655075)      3,466,539.000*** 2,416,442.000*** 1,288,859.00
0*** 2,014,632.000***
## ====================================================================================
=====================
## Note:                                                                    *p<0.
1; **p<0.05; ***p<0.01
```

```
# Check if the assumptions are met in the best performance model
# Model 31
layout(matrix(c(1,2,3,4),2,2,byrow=T))
plot(model21$fitted, rstudent(model21),
 main="Multi Fit Studentized Residuals",
 xlab="Predictions",ylab="Studentized Resid",
 ylim=c(-2.5,2.5))
abline(h=0, lty=2)

hist(model31$resid,main="Histogram of Residuals")
qqnorm(model31$resid)
qqline(model31$resid)
```


Multi Fit Studentized Residuals
Studentized Resid
-2
0
2
4.48
4.52
4.56
4.60
Predictions


Histogram of Residuals
Frequency
0e+00
6e+06
-0.6
-0.4
-0.2
0.0
0.2
model31$resid


Normal Q-Q Plot
Sample Quantiles
-0.6
-0.2
-4
-2
0
2
4
Theoretical Quantiles

# part C

2084551

30/05/2021

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)
```

# Data Preparation

```
rm(list=ls())
library(ggplot2)
library(dplyr)
library(tidytext)
library(jsonlite)
library(tidyr)
library(readr)
library(topicmodels)
library(geojsonR)
library(stringr)
library(dplyr)
library(tidyr)
library(stm)
library(udpipe)
library(cld2)
library(wordcloud)
library(xml2)
library(rvest)
set.seed(123113)
```

Download the dataset from airbnb.com

```
#crawl data from the website
path <- "http://insideairbnb.com/get-the-data.html"

#table preparation
data_set <- read_html(path)
list_of_tables <- data_set %>% html_nodes("table")

#City index for New York is 63.
city_index <- 63
table <-list_of_tables[[city_index]] %>% html_table()

#Getting the links to download the data
links <- list_of_tables[[city_index]] %>% html_nodes("a") %>% html_attr("href")

# Add it on the dataframe
table$links <- links

# cleaning the column names (lower case, replace _with spaces)
colnames(table) <- gsub(" ","_",tolower(colnames(table)))
colnames(table) <- gsub("/","_",colnames(table))

# Convert the date to a date using the as.Date function.
table$date_compiled <- as.Date(table$date_compiled,format = "%d %b,%Y")
```

```
# Get Review Data
Review_data <- table %>% arrange(desc(date_compiled)) %>%
  filter(grepl("Detailed Review",description)) %>%
  top_n(1)

# Get Listings Data
Listings_data <- table %>% arrange(desc(date_compiled)) %>%
  filter(grepl("Detailed Listings",description)) %>%
  top_n(1)
```

```
#download the files

NYC <- tolower(Listings_data$country_city[1])

download.file(url = Listings_data$links[1],destfile = "listings.csv.gz")

download.file(url = Review_data$links[1], destfile = "reviews.csv.gz")
```

```
listings_detailed <- readr::read_csv("listings.csv.gz")
reviews_detailed <- readr::read_csv("reviews.csv.gz")

listings_detailed <- listings_detailed %>% rename(listing_id = id)
```

# Cleaning for Review file

```
# Check the data structure of review file
str(reviews_detailed)
```

```
## spec_tbl_df [836,586 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ listing_id   : num [1:836586] 2595 2595 2595 2595 2595 ...
##  $ id           : num [1:836586] 17857 19176 19760 34320 46312 ...
##  $ date         : Date[1:836586], format: "2009-11-21" "2009-12-05" ...
##  $ reviewer_id  : num [1:836586] 50679 53267 38960 71130 117113 ...
##  $ reviewer_name: chr [1:836586] "Jean" "Cate" "Anita" "Kai-Uwe" ...
##  $ comments     : chr [1:836586] "Notre séjour de trois nuits.\r\nNous avons appré
cier L'appartement qui est très bien situé. Agréable, propre et"| __truncated__ "Grea
t experience." "I've stayed with my friend at the Midtown Castle for six days and it
was a lovely place to be. A big spacious r"| __truncated__ "We've been staying here f
or about 9 nights, enjoying to be in the center of the city, that never sleeps...shor
t"| __truncated__ ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   listing_id = col_double(),
##   ..   id = col_double(),
##   ..   date = col_date(format = ""),
##   ..   reviewer_id = col_double(),
##   ..   reviewer_name = col_character(),
##   ..   comments = col_character()
##   .. )
```

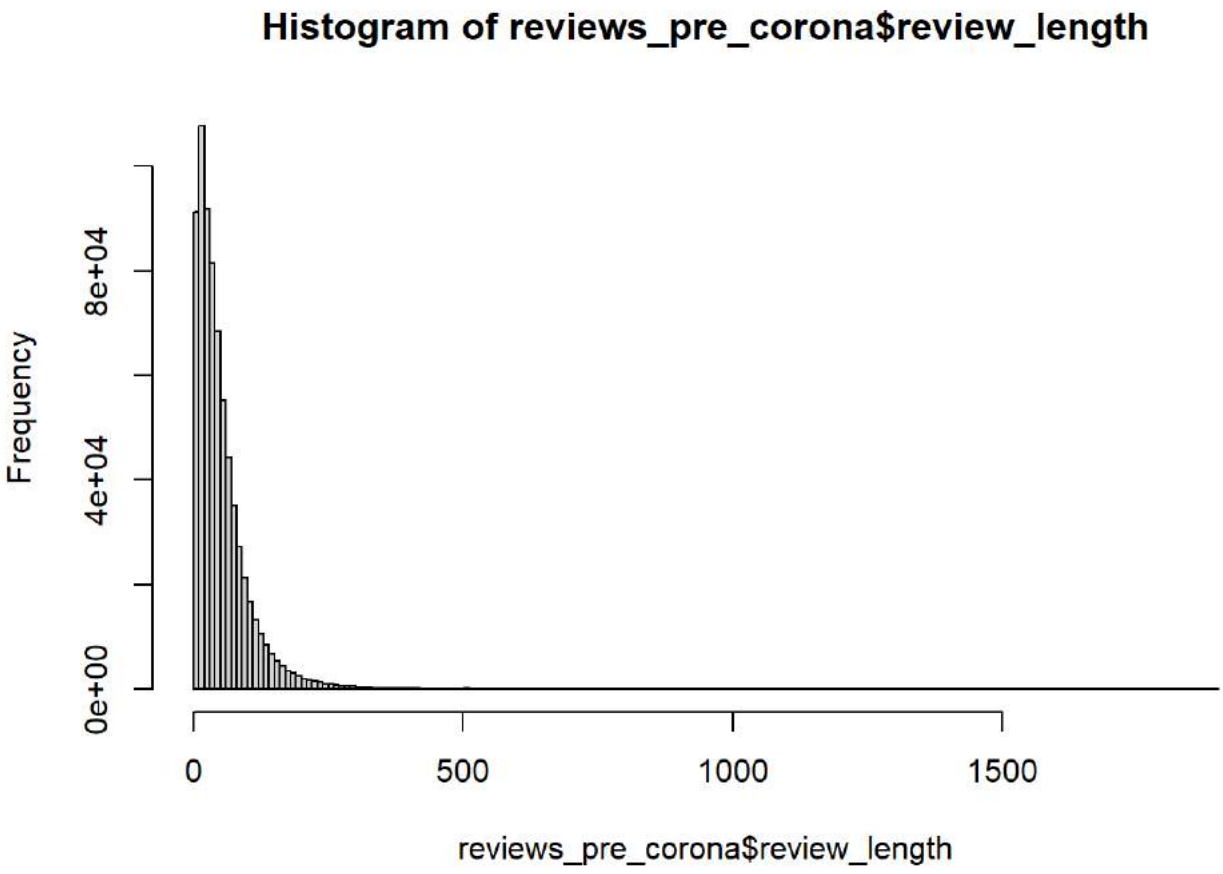
```
reviews_detailed$date <- as.Date(reviews_detailed$date)

# Filtering out reviews after feb 2020 from reviews_detailed dataset
(records_per_year <- reviews_detailed %>% group_by(lubridate::year(date)) %>% summari
se(count = n()))
```

```
## # A tibble: 13 x 2
##    `lubridate::year(date)`  count
##                      <dbl>  <int>
##  1                    2009     93
##  2                    2010    696
##  3                    2011   2952
##  4                    2012   6176
##  5                    2013  11845
##  6                    2014  23391
##  7                    2015  46326
##  8                    2016  80251
##  9                    2017 115833
## 10                    2018 170301
## 11                    2019 239223
## 12                    2020 106937
## 13                    2021  32562
```

```
reviews_pre_corona <- reviews_detailed %>% filter(date<"2020-02-01")
head(reviews_pre_corona)
```

```
## # A tibble: 6 x 6
##   listing_id      id date       reviewer_id reviewer_name comments
##        <dbl>   <dbl> <date>           <dbl> <chr>         <chr>
## 1       2595   17857 2009-11-21       50679 Jean          "Notre séjour de troi~
## 2       2595   19176 2009-12-05       53267 Cate          "Great experience."
## 3       2595   19760 2009-12-10       38960 Anita         "I've stayed with my ~
## 4       2595   34320 2010-04-09       71130 Kai-Uwe       "We've been staying h~
## 5       2595   46312 2010-05-25      117113 Alicia        "We had a wonderful s~
## 6       2595 1238204 2012-05-07     1783688 Sergey        "Hi to everyone!\r\nW~
```

```
# Step 1

#finding the number of words in each review so that we can filter out reviews with ve
ry few words. this will improve the accuracy of language detection
reviews_pre_corona$comments <- iconv(reviews_pre_corona$comments)

reviews_pre_corona$review_length <- sapply(reviews_pre_corona$comments, function(x) l
ength(unlist(strsplit(as.character(x), "\\W+"))))

hist(reviews_pre_corona$review_length, breaks = 200)
```




```
summary(reviews_pre_corona$review_length)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   18.00   39.00   53.15   69.00 1900.00
```

```
#select the lower limit as 15 and upper as 300

reviews_pre_corona <- reviews_pre_corona %>%
  filter(review_length>14) %>%
  filter(review_length<301)


# Now we are ready to Clean up the language
# For some computers the character set is not
# automatically set to latin or ASCII
library(cld2)
#reviews_pre_corona$comments <- iconv(reviews_pre_corona$comments)
reviews_pre_corona$language <- cld2::detect_language(reviews_pre_corona$comments)
reviews_pre_corona%>%
  filter(language =="en") -> reviews_pre_corona

reviews_pre_corona$language <- NULL
reviews_pre_corona <- reviews_pre_corona %>%
  rename("review_id"="id")
```

```
englishmodel <- udpipe::udpipe_download_model("english")

englishmodel <- udpipe::udpipe_load_model(englishmodel$file_model)
```

```
#join the listings and reviews to get the review score rating

#prepare reviews_pre_corona to left join with listing.

reviews_pre_corona_metadata <- reviews_pre_corona

reviews_pre_corona <- reviews_pre_corona %>%
  group_by(listing_id) %>%
  summarise(comments_grouped = paste(comments, collapse = " "))

reviews_joined <- reviews_pre_corona_metadata %>%
  left_join(reviews_pre_corona)


listing_reviews <- listings_detailed %>%
  inner_join(reviews_joined)
```

```
# library(udpipe)
#
# # No need if you have already downloaded it once
# language <- udpipe_download_model(language="english",overwrite = F)
# ud_model <- udpipe_load_model(language)
#
# annotated_reviews_all <- data.frame()
#

#
#
# split_size <- 1000
#
# for_pos_list <- split(data_for_stm,
#                       rep(1:ceiling(nrow(data_for_stm)
#                                     /split_size),
#                           each=split_size,
#                           length.out=nrow(data_for_stm)))
#
#
#
# for(i in 1:length(for_pos_list)){
# #for(i in 1:10){
#
#     udpipe_annotate(for_pos_list[[i]]$comments,
#                 doc_id = for_pos_list[[i]]$review_id,
#                 object= ud_model) %>%
#     as.data.frame() %>%
#     filter(upos %in% c("NOUN","ADJ","ADV")) %>%
#     select(doc_id,lemma) %>%
#     group_by(doc_id) %>%
#     summarise(annotated_comments = paste(lemma, collapse = " ")) %>%
#     rename(review_id = doc_id) -> this_annotated_reviews
#
#   print(paste(i,"from",length(for_pos_list)))
#   annotated_reviews_all <- bind_rows(annotated_reviews_all,
#                                      this_annotated_reviews)
# }
#
# #}
#
# write.csv(annotated_reviews_all,"F:\\WBS\\Term 3\\Text Analytics\\Assignment\\New Y
ork\\Data_NY\\Part C\\annotated_reviews.csv", row.names = FALSE)
```

```
# select the relevant columns for joining with the annotated reviews
data <- listing_reviews %>%
  dplyr::select(review_id, comments, review_scores_rating,listing_id,
         price,date, neighbourhood_cleansed) %>%
  mutate(price = as.numeric(gsub("\\$|,","",price)))


annotated_reviews_all <- readr::read_csv("F:\\WBS\\Term 3\\Text Analytics\\Assignment
\\New York\\Data_NY\\Part C\\annotated_reviews.csv")

annotated_reviews_all$review_id <- as.integer(annotated_reviews_all$review_id)
#left join the annotated reviews with the columns from the main dataset.
dataprep <- annotated_reviews_all %>%
  left_join(data)
```

# STM execution

# Neighborhood level of aggregation

```
# Prepare metadata for the STM model
#Ensuring that the price is numeric.
dataprep$price <- gsub("\\$|,","",dataprep$price)
dataprep$price <- as.numeric(dataprep$price)

#creating mean rating and mean price columns per neighborhood.
data_for_stm <- dataprep %>%
  group_by(neighbourhood_cleansed) %>%
  summarise(comments_grouped_neigh = paste(annotated_comments, collapse = " "), mean_
rating = mean(review_scores_rating), mean_price = mean(price))


data_for_stm <- data_for_stm %>% na.omit()
data_for_stm$rownum <- 1:nrow(data_for_stm)
#Use textprocessor function to remove stopwords and custom stopwords

text <- textProcessor(data_for_stm$comments_grouped_neigh,
                          metadata = data_for_stm,
                          customstopwords = c("neighborhood",
                "manhattan",
                "located",
                "brooklyn",
                "queens",
                "staten",
                "island",
                "bronx",
                "stay",
                "staying",
                "airbnb",
                "line",
                "much",
                "min",
                "also","location", "apartment", "suite", "place", "host", "great","cl
ean"),removestopwords = TRUE
                          ,stem = F)
```

```
## Building corpus...
## Converting to Lower Case...
## Removing punctuation...
## Removing stopwords...
## Remove Custom Stopwords...
## Removing numbers...
## Creating Output...
```

```
# Remove words that appear in less than 1% of the corpus
thresh <- round(1/100 * length(text$documents),0)

output <- prepDocuments(text$documents,
                        text$vocab,
                        text$meta,
                        lower.thresh = thresh)
```

```
## Removing 27811 of 42139 terms (32588 of 408220 tokens) due to frequency
## Your corpus now has 207 documents, 14328 terms and 375632 tokens.
```

```
# numtopics <- searchK(output$documents,output$vocab,K=seq(from=6, to=14,by=1))
# plot(numtopics)
```

```
#Run the stm function to get the 12 topics that we want.
newyorkfit <- stm(documents = output$documents,
                  vocab = output$vocab,
                  K = 12,
                  prevalence = ~mean_rating + mean_price,
                  max.em.its = 75,
                  data = output$meta,
                  reportevery=3,
                  # gamma.prior = "L1",
                  sigma.prior = 0.7,
                  init.type = "Spectral")
```

```
## Beginning Spectral Initialization
##   Calculating the gram matrix...
##   Using only 10000 most frequent terms during initialization...
##   Finding anchor words...
##      ............
##   Recovering initialization...
##
.....................................................................................

## Initialization complete.
##
.....................................................................................

## Completed E-Step (1 seconds).
## Completed M-Step.
## Completing Iteration 1 (approx. per word bound = -6.549)
##
.....................................................................................

## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 2 (approx. per word bound = -6.521, relative change = 4.251e-
03)
##
.....................................................................................

## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 3 (approx. per word bound = -6.509, relative change = 1.989e-
03)
## Topic 1: room, nice, subway, comfortable, really
##  Topic 2: comfortable, just, definitely, many, safe
##  Topic 3: room, time, perfect, nice, good
##  Topic 4: beach, nice, comfortable, perfect, time
##  Topic 5: subway, nice, really, room, comfortable
##  Topic 6: room, nice, subway, good, time
##  Topic 7: nice, home, definitely, comfortable, room
##  Topic 8: perfect, restaurant, comfortable, nice, really
##  Topic 9: nice, room, comfortable, definitely, house
##  Topic 10: subway, nice, room, really, good
##  Topic 11: subway, room, nice, really, time
##  Topic 12: nice, comfortable, room, subway, home
##
.....................................................................................

## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 4 (approx. per word bound = -6.503, relative change = 7.806e-
04)
##
.....................................................................................

## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 5 (approx. per word bound = -6.501, relative change = 3.612e-
04)
##
```

```
.....................................................................................

## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 6 (approx. per word bound = -6.500, relative change = 1.979e-
04)
## Topic 1: room, nice, comfortable, subway, good
##  Topic 2: comfortable, just, definitely, many, safe
##  Topic 3: room, time, perfect, nice, good
##  Topic 4: beach, ferry, nice, time, comfortable
##  Topic 5: nice, subway, really, room, comfortable
##  Topic 6: room, nice, subway, good, really
##  Topic 7: nice, home, definitely, comfortable, time
##  Topic 8: perfect, restaurant, really, comfortable, nice
##  Topic 9: nice, room, comfortable, definitely, home
##  Topic 10: subway, nice, room, really, close
##  Topic 11: subway, room, nice, really, time
##  Topic 12: comfortable, subway, nice, restaurant, room
##
.....................................................................................

## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 7 (approx. per word bound = -6.499, relative change = 1.067e-
04)
##
.....................................................................................

## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 8 (approx. per word bound = -6.499, relative change = 5.351e-
05)
##
.....................................................................................

## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 9 (approx. per word bound = -6.499, relative change = 3.312e-
05)
## Topic 1: room, nice, comfortable, subway, really
##  Topic 2: comfortable, just, definitely, safe, many
##  Topic 3: room, time, perfect, nice, good
##  Topic 4: ferry, beach, nice, house, time
##  Topic 5: nice, subway, really, room, comfortable
##  Topic 6: room, nice, subway, good, really
##  Topic 7: home, nice, definitely, comfortable, time
##  Topic 8: perfect, restaurant, really, comfortable, nice
##  Topic 9: nice, room, comfortable, definitely, home
##  Topic 10: subway, nice, really, close, comfortable
##  Topic 11: subway, room, nice, really, time
##  Topic 12: comfortable, subway, restaurant, nice, home
##
.....................................................................................

## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 10 (approx. per word bound = -6.498, relative change = 2.647e
-05)
```

```
##
.....................................................................................

## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 11 (approx. per word bound = -6.498, relative change = 2.019e
-05)
##
.....................................................................................

## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 12 (approx. per word bound = -6.498, relative change = 1.537e
-05)
## Topic 1: room, nice, comfortable, subway, really
##  Topic 2: comfortable, safe, definitely, just, many
##  Topic 3: room, time, perfect, nice, good
##  Topic 4: ferry, beach, nice, house, time
##  Topic 5: nice, subway, really, room, comfortable
##  Topic 6: room, nice, subway, good, really
##  Topic 7: home, nice, definitely, comfortable, time
##  Topic 8: perfect, restaurant, really, comfortable, nice
##  Topic 9: nice, room, comfortable, definitely, home
##  Topic 10: subway, nice, comfortable, really, close
##  Topic 11: subway, room, nice, really, time
##  Topic 12: comfortable, subway, restaurant, nice, home
##
.....................................................................................

## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 13 (approx. per word bound = -6.498, relative change = 1.237e
-05)
##
.....................................................................................

## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 14 (approx. per word bound = -6.498, relative change = 1.040e
-05)
##
.....................................................................................

## Completed E-Step (0 seconds).
## Completed M-Step.
## Model Converged
```

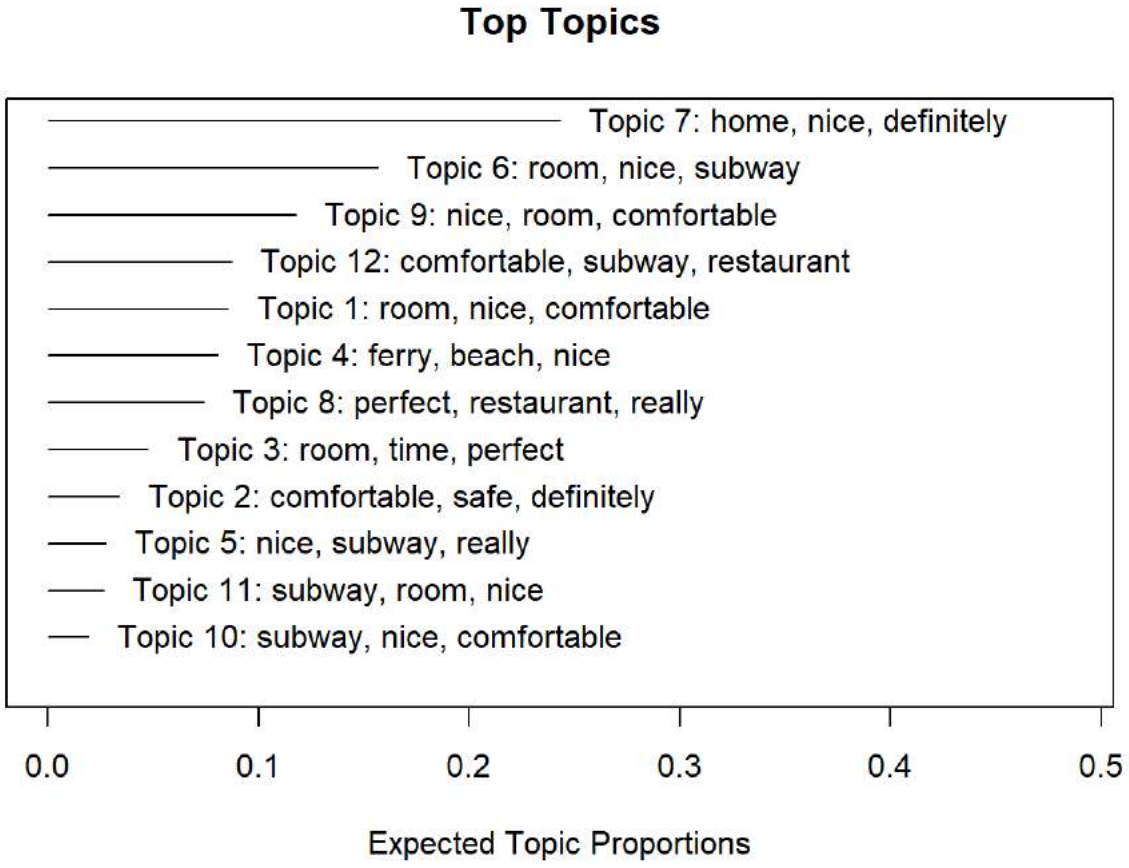
```
summary(newyorkfit)
```

```
## A topic model with 12 topics, 207 documents and a 14328 word dictionary.
```

```
## Topic 1 Top Words:
##       Highest Prob: room, nice, comfortable, subway, really, good, time
##       FREX: tram, campus, hospital, victor, cloister, river, medical
##       Lift: oculus, stackable, undoubtly, manhatthan, sunnyside, relentlessly, est
er
##       Score: room, train, cloister, sunnyside, trish, subway, building
## Topic 2 Top Words:
##       Highest Prob: comfortable, safe, definitely, just, many, cozy, couple
##       FREX: reason, glad, message, couple, quickly, many, safe
##       Lift: reason, monica, intimately, glad, outcome, defininitely, bogeda
##       Score: reason, safe, many, couple, cozy, quickly, message
## Topic 3 Top Words:
##       Highest Prob: room, time, perfect, nice, good, comfortable, really
##       FREX: staff, square, theater, theatre, doorman, show, district
##       Lift: javit, rockerfeller, ucuceuac, macys, martini, ktown, unlivable
##       Score: building, chelsea, perfect, square, highline, central, doorman
## Topic 4 Top Words:
##       Highest Prob: ferry, beach, nice, house, time, comfortable, perfect
##       FREX: beach, ferry, boat, boardwalk, cottage, bungalow, ocean
##       Lift: dolphin, rockaways, beachy, nautical, houseboat, birder, wharf
##       Score: ferry, beach, houseboat, boardwalk, marina, cottage, boat
## Topic 5 Top Words:
##       Highest Prob: nice, subway, really, room, comfortable, easy, time
##       FREX: brownstone, bodega, stuy, neighbourhood, peaches, peach, cafe
##       Lift: stuy, peaches, oxtail, bedford, bushwick, putnam, renee
##       Score: brownstone, subway, peaches, stuy, train, really, station
## Topic 6 Top Words:
##       Highest Prob: room, nice, subway, good, really, close, time
##       FREX: chinese, stadium, diverse, asian, budget, game, station
##       Lift: woodside, herman, coxy, dominant, partement, malaysian, luz
##       Score: room, train, subway, station, house, good, minute
## Topic 7 Top Words:
##       Highest Prob: home, nice, definitely, comfortable, time, house, really
##       FREX: juice, fruit, snack, egg, milk, driveway, caribbean
##       Lift: emaculately, arrea, yonker, trini, kureig, pple, vie
##       Score: home, house, parking, driveway, caribbean, bus, snack
## Topic 8 Top Words:
##       Highest Prob: perfect, restaurant, really, comfortable, nice, easy, time
##       FREX: village, loft, east, bar, west, lower, nightlife
##       Lift: bdfm, bleeker, loath, nyu, villiage, orchard, delancey
##       Score: bar, village, restaurant, soho, loft, perfect, east
## Topic 9 Top Words:
##       Highest Prob: nice, room, comfortable, definitely, home, house, close
##       FREX: airport, layover, flushing, overnight, flight, lirr, basement
##       Lift: hospitalization, acres, bayside, confort<U+FFFD>vel, aditionally, shir
ley, layover
##       Score: airport, layover, flushing, bus, flight, house, driveway
## Topic 10 Top Words:
##       Highest Prob: subway, nice, comfortable, good, really, close, time
##       FREX: museum, upper, east, met, central, side, green
##       Lift: streak, ues, astoria, ditmar, guggenheim, aldi, geoff
##       Score: subway, restaurant, museum, astoria, block, ues, east
## Topic 11 Top Words:
##       Highest Prob: subway, room, nice, really, time, comfortable, good
##       FREX: harlem, express, downtown, agnes, north, jazz, uptown
##       Lift: trinity, agnes, confronting, sylvia, uueuufuuau, uueuuuufuuuucudu, las
ting
```

```
##       Score: harlem, agnes, subway, brownstone, downtown, room, express
## Topic 12 Top Words:
##       Highest Prob: comfortable, subway, restaurant, nice, home, easy, perfect
##       FREX: brownstone, garden, prospect, tree, toy, slope, park
##       Lift: bergen, pre-wedding, brookling, parkslope, bergan, barbes, threes
##       Score: brownstone, prospect, restaurant, pierce, shop, beautiful, subway
```

```
plot(newyorkfit)
```




```
newyorkfit_topics<-tidy(newyorkfit,matrix="beta")
```

```
#We get the top 10 terms:

newyorkfit_terms<-newyorkfit_topics %>%
  group_by(topic) %>%
  slice_max(beta,n=10) %>%
  ungroup() %>%
  arrange(topic,desc(beta))
newyorkfit_terms
```

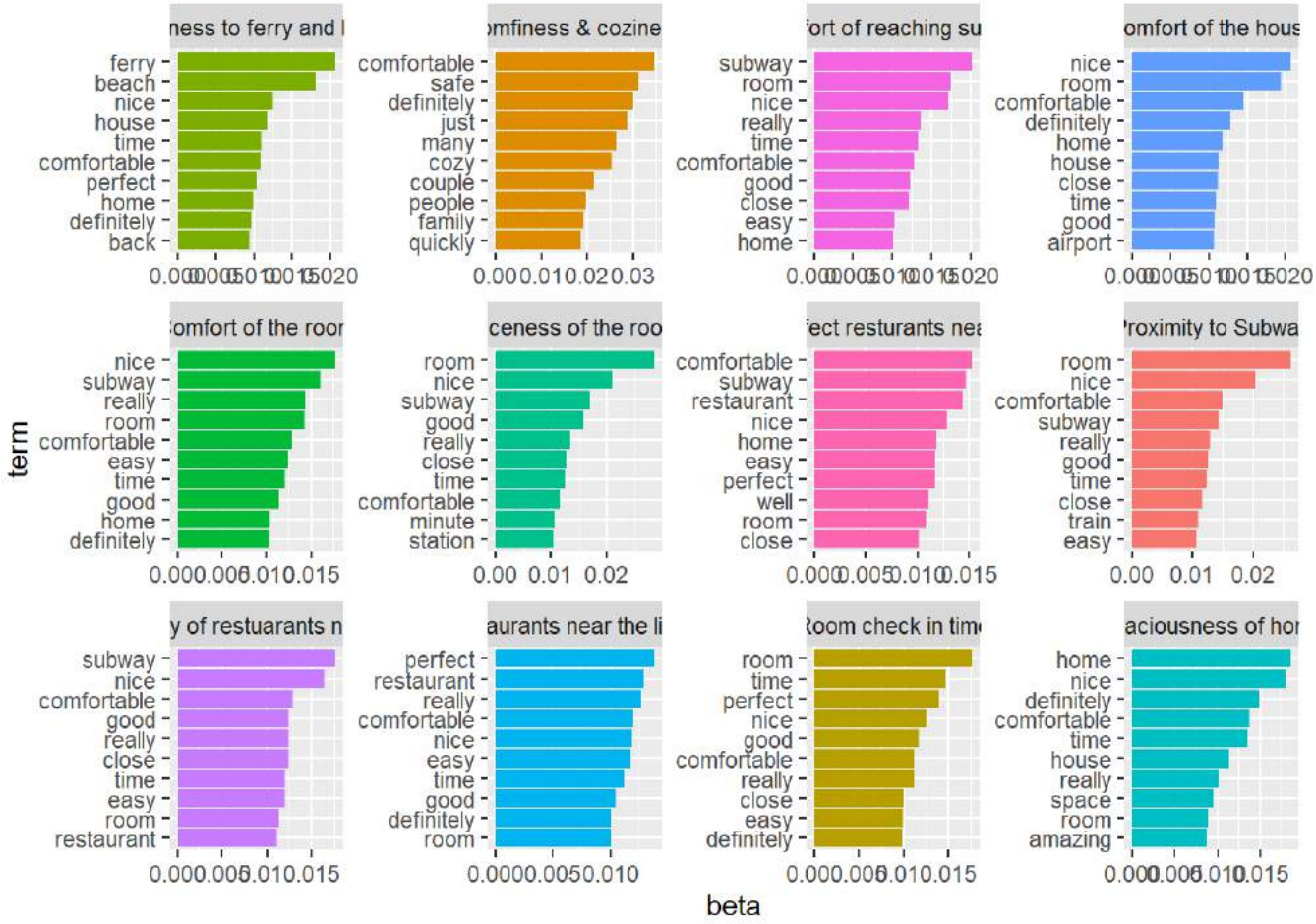
```
## # A tibble: 120 x 3
##    topic term          beta
##    <int> <chr>        <dbl>
##  1     1 room        0.0262
##  2     1 nice        0.0204
##  3     1 comfortable 0.0150
##  4     1 subway      0.0144
##  5     1 really      0.0129
##  6     1 good        0.0125
##  7     1 time        0.0124
##  8     1 close       0.0115
##  9     1 train       0.0109
## 10     1 easy        0.0106
## # ... with 110 more rows
```

***Topic labeling***

```
#we look at the most frequent words appearing in each topic and give a name to each t
opic.
topic_labels <- c("Proximity to Subway", "Comfiness & coziness","Room check in time",
"Closeness to ferry and beach", "Comfort of the room", "Niceness of the room", "Spaci
ousness of home", "Restaurants near the listing", "Comfort of the house", "Quality of
restuarants nearby","Comfort of reaching subway","Perfect resturants nearby")
```

And we can plot the results:

```
#Creat a column to name the topics.
newyorkfit_terms <-  newyorkfit_terms %>%
 mutate(topic_label = case_when((topic == 1) ~ topic_labels[1],
                                (topic == 2) ~ topic_labels[2],
                                (topic == 3) ~ topic_labels[3],
                                (topic == 4) ~ topic_labels[4],
                                (topic == 5) ~ topic_labels[5],
                                (topic == 6) ~ topic_labels[6],
                                (topic == 7) ~ topic_labels[7],
                                (topic == 8) ~ topic_labels[8],
                                (topic == 9) ~ topic_labels[9],
                                (topic == 10) ~ topic_labels[10],
                                (topic == 11) ~ topic_labels[11],
                                (topic == 12) ~ topic_labels[12]))


newyorkfit_terms %>%
mutate(term = reorder_within(term, beta, topic)) %>%
ggplot(aes(beta, term, fill = factor(topic))) +
geom_col(show.legend = FALSE) +
facet_wrap(~ topic_label, scales = "free") +
scale_y_reordered()
```



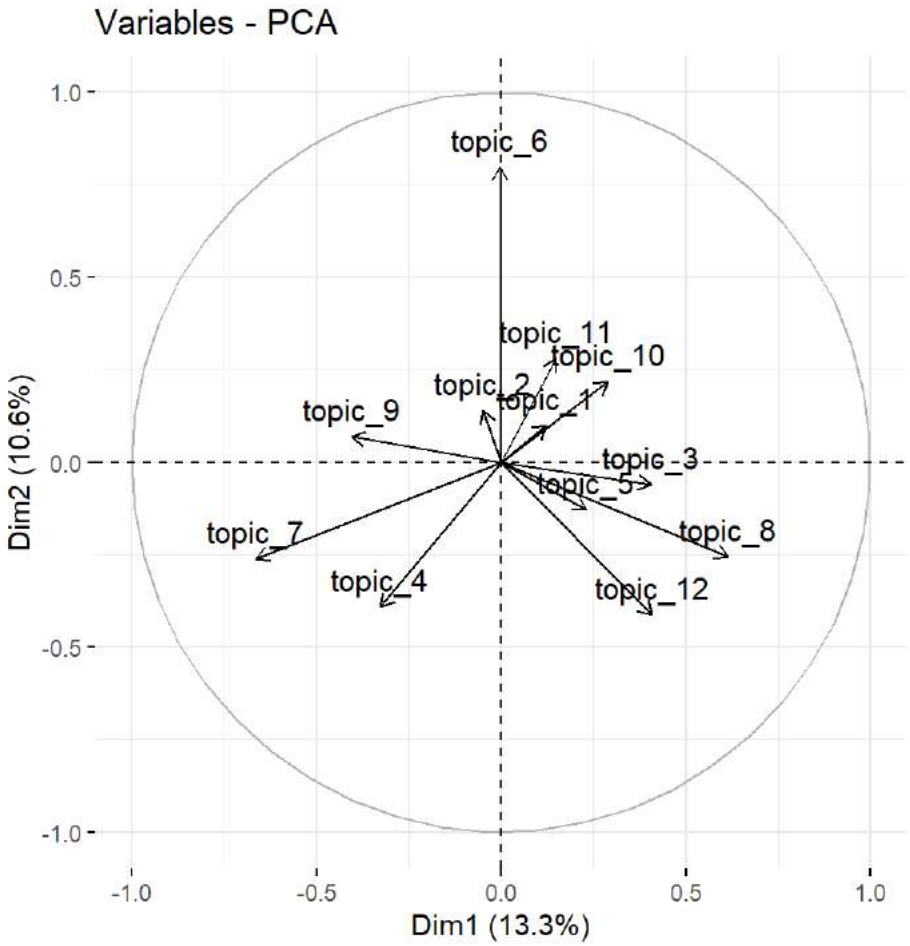

```
#install.packages("wordcloud")
library(wordcloud)
# stm::cloud(newyorkfit,topic = 1)
# stm::cloud(newyorkfit,topic = 2)
# stm::cloud(newyorkfit,topic = 3)
# stm::cloud(newyorkfit,topic = 4)
# stm::cloud(newyorkfit,topic = 5)
# stm::cloud(newyorkfit,topic = 6)
# stm::cloud(newyorkfit,topic = 7)
# stm::cloud(newyorkfit,topic = 8)
```

```
gamma_topics<-tidy(newyorkfit,matrix="gamma")
#pivot the gamma_topics
gamma_topics<-gamma_topics %>%
  pivot_wider(names_from=topic,values_from=gamma)

topic_labels <- paste0("topic_",1:12)
colnames(gamma_topics)<-c("document",topic_labels)

rownames(gamma_topics) <- gamma_topics$document
```

```
gamma_topics$document <- NULL
gamma_topics<-as.data.frame(gamma_topics)
head(gamma_topics)
```

```
##        topic_1     topic_2      topic_3     topic_4      topic_5      topic_6
## 1 0.0009040755 0.052111993 0.0015496820 0.001564376 8.792515e-03 2.134715e-01
## 2 0.0001227789 0.013496545 0.0612907623 0.033758830 2.692640e-04 6.269883e-01
## 3 0.0277530156 0.093451706 0.0019620580 0.295557704 5.785120e-04 1.112041e-01
## 4 0.0471467424 0.005256529 0.0000619471 0.913020549 3.915277e-05 2.141963e-05
## 5 0.0530623547 0.016803928 0.0001663602 0.009483750 1.422739e-02 3.477360e-01
## 6 0.0093283088 0.085686159 0.0002914644 0.053285631 5.650643e-02 2.273566e-01
##        topic_7      topic_8      topic_9     topic_10     topic_11     topic_12
## 1 0.6789053168 0.0008909546 0.0008120950 2.790146e-02 0.0005193678 0.0125766401
## 2 0.0003377518 0.0087188141 0.2526262227 1.688447e-03 0.0005945110 0.0001077546
## 3 0.3324846767 0.0011784056 0.1250569011 2.005200e-03 0.0004497273 0.0083179518
## 4 0.0005971949 0.0001864113 0.0313910625 1.892094e-05 0.0000150557 0.0022450148
## 5 0.0157252059 0.0003390068 0.0004400003 5.102165e-01 0.0001276188 0.0316718903
## 6 0.4553719117 0.0121888581 0.0640574642 9.362946e-04 0.0008950375 0.0340958017
```

```
#Run PCA and plot the Biplot
#install.packages("factoextra")
library(factoextra)
pcah2 <- FactoMineR::PCA(gamma_topics,graph = FALSE)
factoextra::fviz_pca_var(pcah2)
```




***Expected topic proportions***

extract the theta matrix from the fitted object

```
convergence_theta <- as.data.frame(newyorkfit$theta)

topic_summary <- summary(newyorkfit)
```

```
## A topic model with 12 topics, 207 documents and a 14328 word dictionary.
```

```
topic_proportions <- colMeans(newyorkfit$theta)
```

```
# Finding number  amenities, bathrooms,
i <- 1
for (i in 1:length(listings_detailed$host_verifications)) {

  listings_detailed$num_amenities[i] <- length(strsplit(listings_detailed$amenities
[i], split= " ")[[1]])
  listings_detailed$num_bath[i] <- as.numeric(gsub("[^0-9.]", "",listings_detailed$ba
throoms_text[i]))
    i <- i+1
}

all_sentiments <- readr::read_csv("sentiments.csv")

#joining listing data with the sentiments data. Before that we need to select the new
ly created columns to include in the regressions analysis.
join_data <- listings_detailed %>% dplyr::select(listing_id, neighbourhood_cleansed,
 num_bath,num_amenities,beds, room_type)

sentiment <- all_sentiments %>% left_join(join_data)
#Group by neighborhood and get a mean sentiment, mean beds, mean bathrooms and amenit
ies.
sentiment <- sentiment %>%
  group_by(neighbourhood_cleansed) %>%
  summarise(avg_bing = mean(bing_liu_sentiment),
            avg_nrc = mean(nrc_sentiment),
            avg_afinn = mean(afinn_sentiment),
            avg_loughran = mean(loughran_sentiment),
            avg_bath = mean(num_bath),
            avg_beds = mean(beds),
            avg_amenities = mean(num_amenities))
sentiment <- sentiment %>% na.omit()
```

```
stm_object<- newyorkfit$theta

colnames(stm_object) <- c("Proximity to Subway", "Comfiness & coziness","Room check i
n time","Closeness to ferry and beach", "Comfort of the room", "Niceness of the room"
, "Spaciousness of home", "Restaurants near the listing", "Comfort of the house", "Qu
ality of restuarants nearby","Comfort of reaching subway","Perfect resturants nearby"
)

causal_topic_df <- cbind(output$meta,stm_object)

causal_topic_df %>%
  left_join(sentiment) %>%
  dplyr::select(-c(comments_grouped_neigh, neighbourhood_cleansed, rownum)) %>%
  na.omit() -> regress_stm_price_review
```

```
model_search_review <- lm(mean_rating~.,data=regress_stm_price_review)
MASS::stepAIC(model_search_review)
```

Start: AIC=138.39 mean_rating ~ mean_price + `Proximity to Subway` + `Comfiness & coziness` + `Room check in time` + `Closeness to ferry and beach` + `Comfort of the room` + `Niceness of the room` + `Spaciousness of home` + `Restaurants near the listing` + `Comfort of the house` + `Quality of restuarants nearby` + `Comfort of reaching subway` + `Perfect resturants nearby` + avg_bing + avg_nrc + avg_afinn + avg_loughran + avg_bath + avg_beds + avg_amenities

Step: AIC=138.39 mean_rating ~ mean_price + `Proximity to Subway` + `Comfiness & coziness` + `Room check in time` + `Closeness to ferry and beach` + `Comfort of the room` + `Niceness of the room` + `Spaciousness of home` + `Restaurants near the listing` + `Comfort of the house` + `Quality of restuarants nearby` + `Comfort of reaching subway` + avg_bing + avg_nrc + avg_afinn + avg_loughran + avg_bath + avg_beds + avg_amenities

| | Df | Sum of Sq | RSS | AIC |
|---|---|---|---|---|

- avg_nrc 1 0.004 305.53 136.39
- avg_beds 1 0.021 305.55 136.40
- `Closeness to ferry and beach` 1 0.071 305.60 136.43
- `Comfiness & coziness` 1 0.116 305.64 136.46
- mean_price 1 0.158 305.68 136.48
- avg_bath 1 0.191 305.72 136.50
- `Spaciousness of home` 1 0.759 306.28 136.82
- avg_amenities 1 0.933 306.46 136.92
- `Comfort of reaching subway` 1 1.140 306.66 137.04
- `Proximity to Subway` 1 1.958 307.48 137.50
- `Comfort of the room` 1 2.642 308.17 137.88
- avg_loughran 1 3.013 308.54 138.09
- avg_afinn 1 3.545 309.07 138.39 305.52 138.39
- `Quality of restuarants nearby` 1 4.582 310.11 138.97
- `Restaurants near the listing` 1 6.105 311.63 139.81
- `Room check in time` 1 19.395 324.92 147.04
- avg_bing 1 19.942 325.47 147.33
- `Comfort of the house` 1 21.799 327.32 148.31
- `Niceness of the room` 1 34.188 339.71 154.74

Step: AIC=136.39 mean_rating ~ mean_price + `Proximity to Subway` + `Comfiness & coziness` + `Room check in time` + `Closeness to ferry and beach` + `Comfort of the room` + `Niceness of the room` + `Spaciousness of home` + `Restaurants near the listing` + `Comfort of the house` + `Quality of restuarants nearby` + `Comfort of reaching subway` + avg_bing + avg_afinn + avg_loughran + avg_bath + avg_beds + avg_amenities

| | Df | Sum of Sq | RSS | AIC |
|---|---|---|---|---|

- avg_beds 1 0.028 305.56 134.41
- `Closeness to ferry and beach` 1 0.067 305.60 134.43
- `Comfiness & coziness` 1 0.113 305.64 134.46
- mean_price 1 0.155 305.68 134.48
- avg_bath 1 0.188 305.72 134.50
- `Spaciousness of home` 1 0.760 306.29 134.82
- avg_amenities 1 0.941 306.47 134.93
- `Comfort of reaching subway` 1 1.195 306.72 135.07
- `Proximity to Subway` 1 1.956 307.48 135.50
- `Comfort of the room` 1 2.643 308.17 135.88

- avg_loughran 1 3.125 308.65 136.15
- avg_afinn 1 3.542 309.07 136.39 305.53 136.39
- `Quality of restuarants nearby` 1 4.582 310.11 136.97
- `Restaurants near the listing` 1 6.464 311.99 138.02
- `Room check in time` 1 19.625 325.15 145.16
- `Comfort of the house` 1 21.796 327.32 146.31
- avg_bing 1 32.914 338.44 152.09
- `Niceness of the room` 1 34.270 339.80 152.78

Step: AIC=134.41 mean_rating ~ mean_price + `Proximity to Subway` + `Comfiness & coziness` + `Room check in time` + `Closeness to ferry and beach` + `Comfort of the room` + `Niceness of the room` + `Spaciousness of home` + `Restaurants near the listing` + `Comfort of the house` + `Quality of restuarants nearby` + `Comfort of reaching subway` + avg_bing + avg_afinn + avg_loughran + avg_bath + avg_amenities

```
                                Df Sum of Sq    RSS    AIC
```

- `Closeness to ferry and beach` 1 0.061 305.62 132.44
- `Comfiness & coziness` 1 0.108 305.66 132.47
- mean_price 1 0.135 305.69 132.49
- avg_bath 1 0.219 305.77 132.53
- `Spaciousness of home` 1 0.757 306.31 132.84
- avg_amenities 1 0.924 306.48 132.93
- `Comfort of reaching subway` 1 1.246 306.80 133.11
- `Proximity to Subway` 1 2.024 307.58 133.55
- `Comfort of the room` 1 2.628 308.18 133.89
- avg_loughran 1 3.101 308.66 134.16
- avg_afinn 1 3.515 309.07 134.39 305.56 134.41
- `Quality of restuarants nearby` 1 4.634 310.19 135.01
- `Restaurants near the listing` 1 6.892 312.45 136.27
- `Room check in time` 1 20.369 325.93 143.57
- `Comfort of the house` 1 22.048 327.60 144.46
- avg_bing 1 33.522 339.08 150.42
- `Niceness of the room` 1 34.941 340.50 151.14

Step: AIC=132.44 mean_rating ~ mean_price + `Proximity to Subway` + `Comfiness & coziness` + `Room check in time` + `Comfort of the room` + `Niceness of the room` + `Spaciousness of home` + `Restaurants near the listing` + `Comfort of the house` + `Quality of restuarants nearby` + `Comfort of reaching subway` + avg_bing + avg_afinn + avg_loughran + avg_bath + avg_amenities

```
                                Df Sum of Sq    RSS    AIC
```

- `Comfiness & coziness` 1 0.078 305.69 130.49
- mean_price 1 0.150 305.77 130.53
- avg_bath 1 0.229 305.85 130.57
- avg_amenities 1 0.869 306.49 130.94
- `Spaciousness of home` 1 1.507 307.12 131.29
- `Comfort of reaching subway` 1 1.593 307.21 131.34
- avg_loughran 1 3.041 308.66 132.16
- `Proximity to Subway` 1 3.182 308.80 132.24
- `Comfort of the room` 1 3.354 308.97 132.33 305.62 132.44
- avg_afinn 1 3.760 309.38 132.56
- `Quality of restuarants nearby` 1 5.672 311.29 133.62

- `Restaurants near the listing` 1 9.659 315.28 135.83
- `Room check in time` 1 24.721 330.34 143.90
- `Comfort of the house` 1 31.508 337.12 147.42
- avg_bing 1 34.308 339.92 148.85
- `Niceness of the room` 1 49.281 354.90 156.31

Step: AIC=130.49 mean_rating ~ mean_price `+ Proximity to Subway + Room check in time + Comfort of the room + Niceness of the room + Spaciousness of home + Restaurants near the listing + Comfort of the house + Quality of restuarants nearby + Comfort of reaching subway` + avg_bing + avg_afinn + avg_loughran + avg_bath + avg_amenities

```
                                Df Sum of Sq    RSS    AIC
```

- mean_price 1 0.178 305.87 128.59
- avg_bath 1 0.200 305.89 128.60
- avg_amenities 1 0.979 306.67 129.04
- `Spaciousness of home` 1 1.695 307.39 129.44
- `Comfort of reaching subway` 1 1.730 307.42 129.46
- avg_loughran 1 3.065 308.76 130.21
- `Comfort of the room` 1 3.532 309.23 130.47 305.69 130.49
- `Proximity to Subway` 1 3.654 309.35 130.54
- avg_afinn 1 3.682 309.38 130.56
- `Quality of restuarants nearby` 1 5.940 311.63 131.82
- `Restaurants near the listing` 1 10.024 315.72 134.07
- `Room check in time` 1 25.810 331.50 142.51
- `Comfort of the house` 1 33.066 338.76 146.26
- avg_bing 1 34.401 340.10 146.94
- `Niceness of the room` 1 52.645 358.34 155.98

Step: AIC=128.59 mean_rating ~ `Proximity to Subway + Room check in time + Comfort of the room + Niceness of the room + Spaciousness of home + Restaurants near the listing + Comfort of the house + Quality of restuarants nearby + Comfort of reaching subway` + avg_bing + avg_afinn + avg_loughran + avg_bath + avg_amenities

```
                                Df Sum of Sq    RSS    AIC
```

- avg_bath 1 0.115 305.99 126.65
- avg_amenities 1 1.025 306.90 127.17
- `Spaciousness of home` 1 1.518 307.39 127.44
- `Comfort of reaching subway` 1 1.614 307.49 127.50
- avg_loughran 1 3.049 308.92 128.30
- `Comfort of the room` 1 3.357 309.23 128.48 305.87 128.59
- `Proximity to Subway` 1 3.668 309.54 128.65
- avg_afinn 1 3.680 309.55 128.66
- `Quality of restuarants nearby` 1 5.837 311.71 129.86
- `Restaurants near the listing` 1 12.908 318.78 133.74
- `Room check in time` 1 28.247 334.12 141.87
- avg_bing 1 34.427 340.30 145.04
- `Comfort of the house` 1 35.334 341.21 145.50
- `Niceness of the room` 1 65.733 371.61 160.26

Step: AIC=126.65 mean_rating ~ `Proximity to Subway + Room check in time + Comfort of the room + Niceness of the room + Spaciousness of home + Restaurants near the listing + Comfort of the house + Quality of restuarants nearby + Comfort of reaching subway +`

avg_bing + avg_afinn + avg_loughran + avg_amenities

```
                                    Df Sum of Sq    RSS    AIC
```

- avg_amenities 1 0.968 306.96 125.20
- `Spaciousness of home` 1 1.780 307.77 125.66
- `Comfort of reaching subway` 1 1.787 307.77 125.66
- avg_loughran 1 3.160 309.15 126.43
- `Comfort of the room` 1 3.515 309.50 126.63 305.99 126.65
- avg_afinn 1 3.571 309.56 126.66
- `Proximity to Subway` 1 4.216 310.20 127.02
- `Quality of restuarants nearby` 1 6.111 312.10 128.07
- `Restaurants near the listing` 1 13.687 319.67 132.22
- `Room check in time` 1 29.819 335.81 140.74
- avg_bing 1 34.333 340.32 143.05
- `Comfort of the house` 1 37.841 343.83 144.82
- `Niceness of the room` 1 72.199 378.19 161.30

Step: AIC=125.2 mean_rating ~ `Proximity to Subway + Room check in time + Comfort of the room + Niceness of the room + Spaciousness of home + Restaurants near the listing + Comfort of the house + Quality of restuarants nearby + Comfort of reaching subway +` avg_bing + avg_afinn + avg_loughran

```
                                    Df Sum of Sq    RSS    AIC
```

- `Comfort of reaching subway` 1 1.639 308.59 124.12
- `Spaciousness of home` 1 2.125 309.08 124.39
- `Comfort of the room` 1 3.291 310.25 125.05 306.96 125.20
- avg_afinn 1 3.600 310.56 125.22
- avg_loughran 1 3.649 310.61 125.25
- `Proximity to Subway` 1 4.384 311.34 125.65
- `Quality of restuarants nearby` 1 5.704 312.66 126.39
- `Restaurants near the listing` 1 12.732 319.69 130.23
- `Room check in time` 1 28.969 335.93 138.80
- avg_bing 1 36.709 343.66 142.74
- `Comfort of the house` 1 37.713 344.67 143.25
- `Niceness of the room` 1 71.265 378.22 159.32

Step: AIC=124.12 mean_rating ~ `Proximity to Subway + Room check in time + Comfort of the room + Niceness of the room + Spaciousness of home + Restaurants near the listing + Comfort of the house + Quality of restuarants nearby +` avg_bing + avg_afinn + avg_loughran

```
                                    Df Sum of Sq    RSS    AIC
```

- `Spaciousness of home` 1 1.366 309.96 122.89
- avg_loughran 1 3.063 311.66 123.83
- `Comfort of the room` 1 3.169 311.76 123.89 308.59 124.12
- `Proximity to Subway` 1 3.607 312.20 124.13
- avg_afinn 1 4.732 313.33 124.75
- `Quality of restuarants nearby` 1 5.821 314.42 125.35
- `Restaurants near the listing` 1 11.381 319.98 128.39
- `Room check in time` 1 27.385 335.98 136.83
- `Comfort of the house` 1 36.481 345.08 141.45

- avg_bing 1 37.759 346.35 142.09
- `Niceness of the room` 1 70.469 379.06 157.70

Step: AIC=122.89 mean_rating ~ `Proximity to Subway + Room check in time + Comfort of the room + Niceness of the room + Restaurants near the listing + Comfort of the house + Quality of restuarants nearby` + avg_bing + avg_afinn + avg_loughran

```
                                Df Sum of Sq    RSS    AIC
```

- `Comfort of the room` 1 2.293 312.25 122.16
- `Proximity to Subway` 1 2.368 312.33 122.20
- avg_loughran 1 2.500 312.46 122.28 309.96 122.89
- `Quality of restuarants nearby` 1 4.648 314.61 123.46
- avg_afinn 1 5.479 315.44 123.92
- `Restaurants near the listing` 1 10.301 320.26 126.54
- `Room check in time` 1 26.737 336.70 135.20
- avg_bing 1 37.622 347.58 140.70
- `Comfort of the house` 1 39.018 348.98 141.40
- `Niceness of the room` 1 85.742 395.70 163.14

Step: AIC=122.16 mean_rating ~ `Proximity to Subway + Room check in time + Niceness of the room + Restaurants near the listing + Comfort of the house + Quality of restuarants nearby` + avg_bing + avg_afinn + avg_loughran

```
                                Df Sum of Sq    RSS    AIC
```

- `Proximity to Subway` 1 2.086 314.34 121.31
- avg_loughran 1 2.645 314.90 121.62 312.25 122.16
- `Quality of restuarants nearby` 1 4.195 316.45 122.47
- avg_afinn 1 5.754 318.01 123.32
- `Restaurants near the listing` 1 10.554 322.81 125.91
- `Room check in time` 1 25.456 337.71 133.72
- `Comfort of the house` 1 37.161 349.41 139.61
- avg_bing 1 39.277 351.53 140.66
- `Niceness of the room` 1 84.396 396.65 161.55

Step: AIC=121.31 mean_rating ~ `Room check in time + Niceness of the room + Restaurants near the listing + Comfort of the house + Quality of restuarants nearby` + avg_bing + avg_afinn + avg_loughran

```
                                Df Sum of Sq    RSS    AIC
```

- avg_loughran 1 2.362 316.70 120.61 314.34 121.31
- `Quality of restuarants nearby` 1 3.739 318.08 121.36
- avg_afinn 1 6.727 321.07 122.98
- `Restaurants near the listing` 1 9.401 323.74 124.41
- `Room check in time` 1 24.888 339.23 132.50
- `Comfort of the house` 1 35.178 349.52 137.66
- avg_bing 1 37.266 351.61 138.69
- `Niceness of the room` 1 82.461 396.80 159.62

Step: AIC=120.61 mean_rating ~ `Room check in time + Niceness of the room + Restaurants near the listing + Comfort of the house + Quality of restuarants nearby` + avg_bing + avg_afinn

```
                                  Df Sum of Sq    RSS    AIC
```

316.70 120.61 - `Quality of restuarants nearby` 1 3.892 320.59 120.72 - avg_afinn 1 6.218 322.92 121.97 - `Restaurants near the listing` 1 11.381 328.08 124.72 - `Room check in time` 1 26.045 342.75 132.28 - `Comfort of the house` 1 34.740 351.44 136.61 - avg_bing 1 43.028 359.73 140.65 - `Niceness of the room` 1 81.413 398.11 158.19

Call: lm(formula = mean_rating ~ `Room check in time` + `Niceness of the room` + `Restaurants near the listing` + `Comfort of the house` + `Quality of restuarants nearby` + avg_bing + avg_afinn, data = regress_stm_price_review)

Coefficients: (Intercept) `Room check in time`
91.719056 -3.097702
`Niceness of the room` `Restaurants near the listing`
-3.094847 -1.535182
`Comfort of the house` `Quality of restuarants nearby`
-2.273719 -1.696616
avg_bing avg_afinn
6.211151 0.009007

```
summary(lm(mean_rating ~ `Room check in time` + `Niceness of the room` +
    `Restaurants near the listing` + `Comfort of the house` +
    `Quality of restuarants nearby` + avg_bing + avg_afinn, data = regress_stm_price_
review))
```

```
##
## Call:
## lm(formula = mean_rating ~ `Room check in time` + `Niceness of the room` +
##     `Restaurants near the listing` + `Comfort of the house` +
##     `Quality of restuarants nearby` + avg_bing + avg_afinn, data = regress_stm_pri
ce_review)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5181 -0.9258 -0.0541  0.8158  4.8340
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    91.719056   1.000117  91.708  < 2e-16 ***
## `Room check in time`           -3.097702   0.840933  -3.684 0.000311 ***
## `Niceness of the room`         -3.094847   0.475199  -6.513 8.52e-10 ***
## `Restaurants near the listing` -1.535182   0.630451  -2.435 0.015955 *
## `Comfort of the house`         -2.273719   0.534450  -4.254 3.51e-05 ***
## `Quality of restuarants nearby` -1.696616   1.191525  -1.424 0.156363
## avg_bing                        6.211151   1.311829   4.735 4.70e-06 ***
## avg_afinn                       0.009007   0.005004   1.800 0.073698 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.385 on 165 degrees of freedom
## Multiple R-squared:  0.449,  Adjusted R-squared:  0.4256
## F-statistic: 19.21 on 7 and 165 DF,  p-value: < 2.2e-16
```

```
model_search_price <- lm((mean_price)~.,data=regress_stm_price_review)
MASS::stepAIC(model_search_price)
```

```
## Start:  AIC=1212.19
## (mean_price) ~ mean_rating + `Proximity to Subway` + `Comfiness & coziness` +
##     `Room check in time` + `Closeness to ferry and beach` + `Comfort of the room`
+
##     `Niceness of the room` + `Spaciousness of home` + `Restaurants near the listin
g` +
##     `Comfort of the house` + `Quality of restuarants nearby` +
##     `Comfort of reaching subway` + `Perfect resturants nearby` +
##     avg_bing + avg_nrc + avg_afinn + avg_loughran + avg_bath +
##     avg_beds + avg_amenities
##
##
## Step:  AIC=1212.19
## (mean_price) ~ mean_rating + `Proximity to Subway` + `Comfiness & coziness` +
##     `Room check in time` + `Closeness to ferry and beach` + `Comfort of the room`
+
##     `Niceness of the room` + `Spaciousness of home` + `Restaurants near the listin
g` +
##     `Comfort of the house` + `Quality of restuarants nearby` +
##     `Comfort of reaching subway` + avg_bing + avg_nrc + avg_afinn +
##     avg_loughran + avg_bath + avg_beds + avg_amenities
##
##                                   Df Sum of Sq    RSS    AIC
## - avg_afinn                        1      37.8 151629 1210.2
## - mean_rating                      1      78.4 151669 1210.3
## - `Closeness to ferry and beach`   1     144.5 151735 1210.3
## - avg_amenities                    1     176.3 151767 1210.4
## - avg_loughran                     1     274.7 151866 1210.5
## - avg_bing                         1     340.8 151932 1210.6
## - `Quality of restuarants nearby`  1    1090.1 152681 1211.4
## - `Comfiness & coziness`           1    1734.2 153325 1212.2
## <none>                                       151591 1212.2
## - avg_nrc                          1    1845.4 153436 1212.3
## - `Comfort of reaching subway`     1    2106.9 153698 1212.6
## - avg_beds                         1    4297.2 155888 1215.0
## - `Room check in time`             1    5333.1 156924 1216.2
## - `Comfort of the room`            1    7907.0 159498 1219.0
## - avg_bath                         1    9780.5 161371 1221.0
## - `Comfort of the house`           1   15359.1 166950 1226.9
## - `Spaciousness of home`           1   15381.8 166973 1226.9
## - `Restaurants near the listing`   1   16919.3 168510 1228.5
## - `Proximity to Subway`            1   26614.1 178205 1238.2
## - `Niceness of the room`           1   31117.0 182708 1242.5
##
## Step:  AIC=1210.23
## (mean_price) ~ mean_rating + `Proximity to Subway` + `Comfiness & coziness` +
##     `Room check in time` + `Closeness to ferry and beach` + `Comfort of the room`
+
##     `Niceness of the room` + `Spaciousness of home` + `Restaurants near the listin
g` +
##     `Comfort of the house` + `Quality of restuarants nearby` +
##     `Comfort of reaching subway` + avg_bing + avg_nrc + avg_loughran +
##     avg_bath + avg_beds + avg_amenities
##
##                                   Df Sum of Sq    RSS    AIC
## - mean_rating                      1        68 151697 1208.3
## - `Closeness to ferry and beach`   1       127 151755 1208.4
```

```
## - avg_amenities                    1       171 151799 1208.4
## - avg_loughran                     1       275 151904 1208.5
## - avg_bing                         1       364 151992 1208.6
## - `Quality of restuarants nearby`  1      1125 152754 1209.5
## <none>                                       151629 1210.2
## - `Comfiness & coziness`           1      1812 153441 1210.3
## - avg_nrc                          1      1839 153468 1210.3
## - `Comfort of reaching subway`     1      2249 153877 1210.8
## - avg_beds                         1      4263 155891 1213.0
## - `Room check in time`             1      5320 156949 1214.2
## - `Comfort of the room`            1      7943 159572 1217.1
## - avg_bath                         1      9798 161427 1219.1
## - `Spaciousness of home`           1     15580 167209 1225.2
## - `Comfort of the house`           1     15962 167590 1225.5
## - `Restaurants near the listing`   1     16903 168531 1226.5
## - `Proximity to Subway`            1     27283 178911 1236.8
## - `Niceness of the room`           1     32453 184081 1241.8
##
## Step:  AIC=1208.31
## (mean_price) ~ `Proximity to Subway` + `Comfiness & coziness` +
##     `Room check in time` + `Closeness to ferry and beach` + `Comfort of the room`
+
##     `Niceness of the room` + `Spaciousness of home` + `Restaurants near the listin
g` +
##     `Comfort of the house` + `Quality of restuarants nearby` +
##     `Comfort of reaching subway` + avg_bing + avg_nrc + avg_loughran +
##     avg_bath + avg_beds + avg_amenities
##
##                                   Df Sum of Sq    RSS    AIC
## - `Closeness to ferry and beach`   1       133 151829 1206.5
## - avg_amenities                    1       185 151881 1206.5
## - avg_loughran                     1       251 151948 1206.6
## - avg_bing                         1       472 152168 1206.8
## - `Quality of restuarants nearby`  1      1073 152769 1207.5
## <none>                                       151697 1208.3
## - `Comfiness & coziness`           1      1820 153516 1208.4
## - avg_nrc                          1      1839 153536 1208.4
## - `Comfort of reaching subway`     1      2202 153899 1208.8
## - avg_beds                         1      4265 155962 1211.1
## - `Room check in time`             1      6113 157809 1213.1
## - `Comfort of the room`            1      7875 159572 1215.1
## - avg_bath                         1      9790 161486 1217.1
## - `Spaciousness of home`           1     15515 167211 1223.2
## - `Comfort of the house`           1     16731 168428 1224.4
## - `Restaurants near the listing`   1     17745 169442 1225.5
## - `Proximity to Subway`            1     27262 178959 1234.9
## - `Niceness of the room`           1     36172 187868 1243.3
##
## Step:  AIC=1206.46
## (mean_price) ~ `Proximity to Subway` + `Comfiness & coziness` +
##     `Room check in time` + `Comfort of the room` + `Niceness of the room` +
##     `Spaciousness of home` + `Restaurants near the listing` +
##     `Comfort of the house` + `Quality of restuarants nearby` +
##     `Comfort of reaching subway` + avg_bing + avg_nrc + avg_loughran +
##     avg_bath + avg_beds + avg_amenities
##
##                                   Df Sum of Sq    RSS    AIC
## - avg_amenities                    1       141 151970 1204.6
```

```
## - avg_loughran                     1       328 152157 1204.8
## - avg_bing                         1       457 152287 1205.0
## - `Quality of restuarants nearby`  1       940 152769 1205.5
## - `Comfiness & coziness`           1      1687 153517 1206.4
## <none>                                       151829 1206.5
## - `Comfort of reaching subway`     1      2117 153947 1206.8
## - avg_nrc                          1      2194 154023 1206.9
## - avg_beds                         1      4315 156145 1209.3
## - `Comfort of the room`            1      8214 160043 1213.6
## - `Room check in time`             1      8675 160504 1214.1
## - avg_bath                         1      9830 161659 1215.3
## - `Spaciousness of home`           1     21315 173144 1227.2
## - `Comfort of the house`           1     21942 173771 1227.8
## - `Restaurants near the listing`   1     28612 180441 1234.3
## - `Proximity to Subway`            1     35839 187669 1241.1
## - `Niceness of the room`           1     52789 204618 1256.1
##
## Step:  AIC=1204.62
## (mean_price) ~ `Proximity to Subway` + `Comfiness & coziness` +
##     `Room check in time` + `Comfort of the room` + `Niceness of the room` +
##     `Spaciousness of home` + `Restaurants near the listing` +
##     `Comfort of the house` + `Quality of restuarants nearby` +
##     `Comfort of reaching subway` + avg_bing + avg_nrc + avg_loughran +
##     avg_bath + avg_beds
##
##                                  Df Sum of Sq    RSS    AIC
## - avg_loughran                    1       278 152248 1202.9
## - avg_bing                        1       474 152444 1203.2
## - `Quality of restuarants nearby`  1      1015 152985 1203.8
## <none>                                       151970 1204.6
## - `Comfiness & coziness`          1      1899 153869 1204.8
## - avg_nrc                         1      2113 154084 1205.0
## - `Comfort of reaching subway`    1      2198 154168 1205.1
## - avg_beds                        1      4478 156449 1207.6
## - `Comfort of the room`           1      8391 160362 1211.9
## - `Room check in time`            1      8535 160505 1212.1
## - avg_bath                        1      9935 161906 1213.6
## - `Spaciousness of home`          1     21202 173172 1225.2
## - `Comfort of the house`          1     21913 173884 1225.9
## - `Restaurants near the listing`  1     29320 181290 1233.1
## - `Proximity to Subway`           1     35707 187677 1239.1
## - `Niceness of the room`          1     54152 206123 1255.3
##
## Step:  AIC=1202.94
## (mean_price) ~ `Proximity to Subway` + `Comfiness & coziness` +
##     `Room check in time` + `Comfort of the room` + `Niceness of the room` +
##     `Spaciousness of home` + `Restaurants near the listing` +
##     `Comfort of the house` + `Quality of restuarants nearby` +
##     `Comfort of reaching subway` + avg_bing + avg_nrc + avg_bath +
##     avg_beds
##
##                                  Df Sum of Sq    RSS    AIC
## - avg_bing                        1       825 153073 1201.9
## - `Quality of restuarants nearby`  1       961 153210 1202.0
## <none>                                       152248 1202.9
## - `Comfiness & coziness`          1      1848 154096 1203.0
## - avg_nrc                         1      1908 154156 1203.1
## - `Comfort of reaching subway`    1      2037 154286 1203.2
```

```
## - avg_beds                          1      4487 156735 1206.0
## - `Comfort of the room`             1      8274 160522 1210.1
## - `Room check in time`              1      8706 160954 1210.6
## - avg_bath                          1     10124 162372 1212.1
## - `Spaciousness of home`            1     21190 173439 1223.5
## - `Comfort of the house`            1     21734 173982 1224.0
## - `Restaurants near the listing`    1     29212 181460 1231.3
## - `Proximity to Subway`             1     35715 187964 1237.4
## - `Niceness of the room`            1     54584 206833 1253.9
##
## Step:  AIC=1201.87
## (mean_price) ~ `Proximity to Subway` + `Comfiness & coziness` +
##     `Room check in time` + `Comfort of the room` + `Niceness of the room` +
##     `Spaciousness of home` + `Restaurants near the listing` +
##     `Comfort of the house` + `Quality of restuarants nearby` +
##     `Comfort of reaching subway` + avg_nrc + avg_bath + avg_beds
##
##                                   Df Sum of Sq    RSS    AIC
## - `Quality of restuarants nearby`  1       821 153894 1200.8
## - avg_nrc                          1      1133 154206 1201.2
## <none>                                       153073 1201.9
## - `Comfiness & coziness`           1      1940 155014 1202.0
## - `Comfort of reaching subway`     1      1983 155057 1202.1
## - avg_beds                         1      6482 159555 1207.0
## - `Comfort of the room`            1      7804 160877 1208.5
## - avg_bath                         1      9731 162805 1210.5
## - `Room check in time`             1     10039 163113 1210.9
## - `Spaciousness of home`           1     20426 173499 1221.5
## - `Comfort of the house`           1     21065 174139 1222.2
## - `Restaurants near the listing`   1     29285 182359 1230.2
## - `Proximity to Subway`            1     35393 188466 1235.9
## - `Niceness of the room`           1     54377 207451 1252.5
##
## Step:  AIC=1200.8
## (mean_price) ~ `Proximity to Subway` + `Comfiness & coziness` +
##     `Room check in time` + `Comfort of the room` + `Niceness of the room` +
##     `Spaciousness of home` + `Restaurants near the listing` +
##     `Comfort of the house` + `Comfort of reaching subway` + avg_nrc +
##     avg_bath + avg_beds
##
##                                  Df Sum of Sq    RSS    AIC
## - avg_nrc                         1      1077 154971 1200.0
## - `Comfiness & coziness`          1      1736 155630 1200.7
## <none>                                      153894 1200.8
## - `Comfort of reaching subway`    1      2069 155963 1201.1
## - avg_beds                        1      6669 160563 1206.1
## - `Comfort of the room`           1      7210 161104 1206.7
## - avg_bath                        1     10371 164265 1210.1
## - `Room check in time`            1     11061 164956 1210.8
## - `Spaciousness of home`          1     19821 173715 1219.8
## - `Comfort of the house`          1     20309 174203 1220.2
## - `Restaurants near the listing`  1     32089 185983 1231.6
## - `Proximity to Subway`           1     34767 188661 1234.0
## - `Niceness of the room`          1     54069 207963 1250.9
##
## Step:  AIC=1200
## (mean_price) ~ `Proximity to Subway` + `Comfiness & coziness` +
##     `Room check in time` + `Comfort of the room` + `Niceness of the room` +
```

```
##     `Spaciousness of home` + `Restaurants near the listing` +
##     `Comfort of the house` + `Comfort of reaching subway` + avg_bath +
##     avg_beds
##
##                                  Df Sum of Sq    RSS    AIC
## - `Comfiness & coziness`          1      1693 156665 1199.9
## <none>                                       154971 1200.0
## - `Comfort of reaching subway`    1      2907 157878 1201.2
## - avg_beds                        1      7248 162220 1205.9
## - `Comfort of the room`           1      7472 162443 1206.2
## - avg_bath                        1      9449 164420 1208.2
## - `Room check in time`            1     10220 165192 1209.0
## - `Spaciousness of home`          1     20673 175644 1219.7
## - `Comfort of the house`          1     22834 177805 1221.8
## - `Restaurants near the listing`  1     31269 186240 1229.8
## - `Proximity to Subway`           1     33902 188873 1232.2
## - `Niceness of the room`          1     60852 215823 1255.3
##
## Step:  AIC=1199.88
## (mean_price) ~ `Proximity to Subway` + `Room check in time` +
##     `Comfort of the room` + `Niceness of the room` + `Spaciousness of home` +
##     `Restaurants near the listing` + `Comfort of the house` +
##     `Comfort of reaching subway` + avg_bath + avg_beds
##
##                                  Df Sum of Sq    RSS    AIC
## <none>                                       156665 1199.9
## - `Comfort of reaching subway`    1      2497 159162 1200.6
## - `Comfort of the room`           1      7039 163704 1205.5
## - avg_beds                        1      7707 164372 1206.2
## - avg_bath                        1     10765 167430 1209.4
## - `Room check in time`            1     11692 168357 1210.3
## - `Spaciousness of home`          1     19467 176132 1218.2
## - `Comfort of the house`          1     21787 178452 1220.4
## - `Proximity to Subway`           1     32257 188921 1230.3
## - `Restaurants near the listing`  1     33357 190022 1231.3
## - `Niceness of the room`          1     59228 215893 1253.4
```

```
##
## Call:
## lm(formula = (mean_price) ~ `Proximity to Subway` + `Room check in time` +
##     `Comfort of the room` + `Niceness of the room` + `Spaciousness of home` +
##     `Restaurants near the listing` + `Comfort of the house` +
##     `Comfort of reaching subway` + avg_bath + avg_beds, data = regress_stm_price_r
eview)
##
## Coefficients:
##                     (Intercept)           `Proximity to Subway`
##                           79.04                          -80.56
##           `Room check in time`           `Comfort of the room`
##                           69.06                          -95.34
##         `Niceness of the room`          `Spaciousness of home`
##                          -95.91                          -52.00
## `Restaurants near the listing`          `Comfort of the house`
##                           98.60                          -62.50
##   `Comfort of reaching subway`                        avg_bath
##                          -45.80                           44.87
##                        avg_beds
##                           12.25
```

```
summary(lm((mean_price) ~ `Proximity to Subway` + `Comfiness & coziness` +
    `Room check in time` + `Comfort of the room` + `Niceness of the room` +
    `Spaciousness of home` + `Restaurants near the listing` +
    `Comfort of the house` + `Comfort of reaching subway` + avg_bath +
    avg_beds, data = regress_stm_price_review))
```

```
##
## Call:
## lm(formula = (mean_price) ~ `Proximity to Subway` + `Comfiness & coziness` +
##     `Room check in time` + `Comfort of the room` + `Niceness of the room` +
##     `Spaciousness of home` + `Restaurants near the listing` +
##     `Comfort of the house` + `Comfort of reaching subway` + avg_bath +
##     avg_beds, data = regress_stm_price_review)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.605 -17.261  -3.346  11.484 154.370
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      85.876     18.823   4.562 9.98e-06 ***
## `Proximity to Subway`           -84.060     14.164  -5.935 1.75e-08 ***
## `Comfiness & coziness`          -50.844     38.332  -1.326  0.18659
## `Room check in time`             65.241     20.022   3.259  0.00137 **
## `Comfort of the room`           -98.441     35.332  -2.786  0.00597 **
## `Niceness of the room`          -98.088     12.336  -7.951 3.07e-13 ***
## `Spaciousness of home`          -54.072     11.668  -4.634 7.35e-06 ***
## `Restaurants near the listing`   96.074     16.856   5.700 5.59e-08 ***
## `Comfort of the house`          -64.340     13.210  -4.871 2.64e-06 ***
## `Comfort of reaching subway`    -49.673     28.583  -1.738  0.08415 .
## avg_bath                         42.435     13.544   3.133  0.00206 **
## avg_beds                         11.904      4.338   2.744  0.00676 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 31.03 on 161 degrees of freedom
## Multiple R-squared:  0.6785, Adjusted R-squared:  0.6565
## F-statistic: 30.89 on 11 and 161 DF,  p-value: < 2.2e-16
```

# Listing level aggregation

```
data <- listing_reviews %>%
  dplyr::select(review_id, comments, review_scores_rating,listing_id,
         price,date) %>%
  mutate(price = as.numeric(gsub("\\$|,","",price)))

annotated_reviews_all <- readr::read_csv("F:\\WBS\\Term 3\\Text Analytics\\Assignment
\\New York\\Data_NY\\Part C\\annotated_reviews.csv")

annotated_reviews_all$review_id <- as.integer(annotated_reviews_all$review_id)

data_for_stm <- annotated_reviews_all %>%
  left_join(data)


# load("annotated_reviews.rda")
# annotated_reviews <- data.table::data.table(annotated_reviews)
```

# STM execution

```
# Prepare metadata for the STM model


data_for_stm <- data_for_stm %>%
  group_by(listing_id) %>%
  summarise(comments_grouped_listing = paste(annotated_comments, collapse = " "))

data_for_joining <- listings_detailed %>%
  dplyr::select(price,review_scores_rating,listing_id)

data_for_joining$price <- gsub("\\$|,","",data_for_joining$price)
data_for_joining$price <- as.numeric(data_for_joining$price)


data_for_stm <- data_for_stm %>%
  left_join(data_for_joining)

data_for_stm <- data_for_stm %>% na.omit()

# First we engage with the
# textProcessor function
# Notice that you should augment the custom
# stopwords
text <- textProcessor(data_for_stm$comments_grouped_listing,
                          metadata = data_for_stm,
                          customstopwords = c("neighborhood",
                "manhattan",
                "located",
                "brooklyn",
                "queens",
                "staten",
                "island",
                "bronx",
                "stay",
                "staying",
                "airbnb",
                "line",
                "much",
                "min",
                "also","apartment","suite","location", "place", "great"),removestopwo
rds = TRUE,
                          stem = F)
```

```
## Building corpus...
## Converting to Lower Case...
## Removing punctuation...
## Removing stopwords...
## Remove Custom Stopwords...
## Removing numbers...
## Creating Output...
```

```
# Keep only those words that appear
# on the 1% of the corpus
#
thresh <- round(1/100 * length(text$documents),0)

output <- prepDocuments(text$documents,
                        text$vocab,
                        text$meta,
                        lower.thresh = thresh)
```

```
## Removing 45672 of 47649 terms (390000 of 3778304 tokens) due to frequency
## Your corpus now has 21114 documents, 1977 terms and 3388304 tokens.
```

```
# numtopics <- searchK(out$documents,out$vocab,K=seq(from=8, to=20,by=5))
# plot(numtopics)
```

```
newyorkfit <- stm(documents = output$documents,
                  vocab = output$vocab,
                  K = 8,
                  prevalence = ~review_scores_rating + price,
                  max.em.its = 75,
                  data = output$meta,
                  reportevery=3,
                  # gamma.prior = "L1",
                  sigma.prior = 0.7,
                  init.type = "Spectral")
```

```
## Beginning Spectral Initialization
##   Calculating the gram matrix...
##   Finding anchor words...
##      ........
##   Recovering initialization...
##      ...................
## Initialization complete.
##
.....................................................................................

## Completed E-Step (7 seconds).
## Completed M-Step.
## Completing Iteration 1 (approx. per word bound = -6.189)
##
.....................................................................................

## Completed E-Step (7 seconds).
## Completed M-Step.
## Completing Iteration 2 (approx. per word bound = -6.157, relative change = 5.112e-
03)
##
.....................................................................................

## Completed E-Step (7 seconds).
## Completed M-Step.
## Completing Iteration 3 (approx. per word bound = -6.144, relative change = 2.138e-
03)
## Topic 1: host, perfect, clean, restaurant, easy
##  Topic 2: clean, subway, host, comfortable, nice
##  Topic 3: clean, subway, easy, host, nice
##  Topic 4: host, nice, really, time, subway
##  Topic 5: host, clean, room, nice, comfortable
##  Topic 6: host, home, clean, comfortable, time
##  Topic 7: clean, room, nice, host, good
##  Topic 8: room, nice, clean, host, subway
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 4 (approx. per word bound = -6.137, relative change = 1.228e-
03)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 5 (approx. per word bound = -6.132, relative change = 7.412e-
04)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 6 (approx. per word bound = -6.129, relative change = 4.647e-
04)
## Topic 1: perfect, host, restaurant, clean, easy
```

```
##  Topic 2: clean, subway, comfortable, easy, host
##  Topic 3: clean, easy, good, subway, host
##  Topic 4: host, really, time, nice, good
##  Topic 5: host, clean, nice, room, comfortable
##  Topic 6: home, host, house, beautiful, comfortable
##  Topic 7: clean, room, nice, good, house
##  Topic 8: room, nice, clean, host, subway
##
.....................................................................................

## Completed E-Step (5 seconds).
## Completed M-Step.
## Completing Iteration 7 (approx. per word bound = -6.127, relative change = 3.156e-
04)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 8 (approx. per word bound = -6.126, relative change = 2.289e-
04)
##
.....................................................................................

## Completed E-Step (5 seconds).
## Completed M-Step.
## Completing Iteration 9 (approx. per word bound = -6.125, relative change = 1.737e-
04)
## Topic 1: perfect, host, restaurant, amazing, clean
##  Topic 2: clean, subway, comfortable, easy, nice
##  Topic 3: clean, perfect, good, easy, restaurant
##  Topic 4: host, really, time, nice, good
##  Topic 5: host, clean, nice, comfortable, definitely
##  Topic 6: home, host, house, beautiful, wonderful
##  Topic 7: clean, room, nice, good, house
##  Topic 8: room, nice, clean, host, subway
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 10 (approx. per word bound = -6.124, relative change = 1.356e
-04)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 11 (approx. per word bound = -6.123, relative change = 1.098e
-04)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 12 (approx. per word bound = -6.123, relative change = 8.953e
-05)
## Topic 1: perfect, host, amazing, space, restaurant
```

```
##  Topic 2: subway, clean, easy, comfortable, nice
##  Topic 3: perfect, clean, restaurant, good, easy
##  Topic 4: host, really, time, nice, friendly
##  Topic 5: host, clean, nice, definitely, comfortable
##  Topic 6: home, host, house, beautiful, family
##  Topic 7: clean, nice, room, good, house
##  Topic 8: room, nice, clean, host, subway
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 13 (approx. per word bound = -6.122, relative change = 7.430e
-05)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 14 (approx. per word bound = -6.122, relative change = 6.275e
-05)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 15 (approx. per word bound = -6.121, relative change = 5.394e
-05)
## Topic 1: perfect, amazing, host, space, super
##  Topic 2: subway, clean, easy, nice, comfortable
##  Topic 3: perfect, restaurant, clean, easy, good
##  Topic 4: host, really, time, friendly, nice
##  Topic 5: host, clean, nice, definitely, comfortable
##  Topic 6: home, house, host, beautiful, family
##  Topic 7: clean, good, nice, room, space
##  Topic 8: room, nice, clean, host, subway
##
.....................................................................................

## Completed E-Step (7 seconds).
## Completed M-Step.
## Completing Iteration 16 (approx. per word bound = -6.121, relative change = 4.724e
-05)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 17 (approx. per word bound = -6.121, relative change = 4.217e
-05)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 18 (approx. per word bound = -6.121, relative change = 3.818e
-05)
## Topic 1: amazing, space, perfect, host, super
```

```
##  Topic 2: subway, clean, easy, nice, close
##  Topic 3: perfect, restaurant, clean, easy, good
##  Topic 4: host, really, time, friendly, nice
##  Topic 5: host, clean, definitely, nice, comfortable
##  Topic 6: home, house, host, beautiful, family
##  Topic 7: clean, good, nice, room, space
##  Topic 8: room, clean, nice, host, subway
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 19 (approx. per word bound = -6.120, relative change = 3.488e
-05)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 20 (approx. per word bound = -6.120, relative change = 3.144e
-05)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 21 (approx. per word bound = -6.120, relative change = 2.745e
-05)
## Topic 1: space, amazing, perfect, host, super
##  Topic 2: subway, clean, easy, nice, close
##  Topic 3: perfect, restaurant, clean, easy, good
##  Topic 4: host, really, time, friendly, nice
##  Topic 5: clean, host, definitely, nice, comfortable
##  Topic 6: home, house, host, family, beautiful
##  Topic 7: clean, good, nice, space, night
##  Topic 8: room, clean, nice, host, subway
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 22 (approx. per word bound = -6.120, relative change = 2.370e
-05)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 23 (approx. per word bound = -6.120, relative change = 2.065e
-05)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 24 (approx. per word bound = -6.120, relative change = 1.810e
-05)
## Topic 1: space, amazing, perfect, super, host
```

```
##  Topic 2: subway, clean, easy, nice, close
##  Topic 3: perfect, restaurant, easy, clean, good
##  Topic 4: host, really, time, friendly, nice
##  Topic 5: clean, host, definitely, nice, comfortable
##  Topic 6: home, house, family, host, beautiful
##  Topic 7: good, clean, nice, night, space
##  Topic 8: room, clean, nice, host, subway
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 25 (approx. per word bound = -6.120, relative change = 1.616e
-05)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 26 (approx. per word bound = -6.120, relative change = 1.444e
-05)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 27 (approx. per word bound = -6.119, relative change = 1.323e
-05)
## Topic 1: space, amazing, super, perfect, host
##  Topic 2: subway, clean, easy, nice, close
##  Topic 3: perfect, restaurant, easy, clean, good
##  Topic 4: host, really, time, friendly, home
##  Topic 5: clean, host, definitely, nice, comfortable
##  Topic 6: home, house, family, beautiful, host
##  Topic 7: good, clean, nice, night, space
##  Topic 8: room, clean, nice, bathroom, host
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 28 (approx. per word bound = -6.119, relative change = 1.182e
-05)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 29 (approx. per word bound = -6.119, relative change = 1.089e
-05)
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Completing Iteration 30 (approx. per word bound = -6.119, relative change = 1.003e
-05)
## Topic 1: space, amazing, super, beautiful, perfect
```

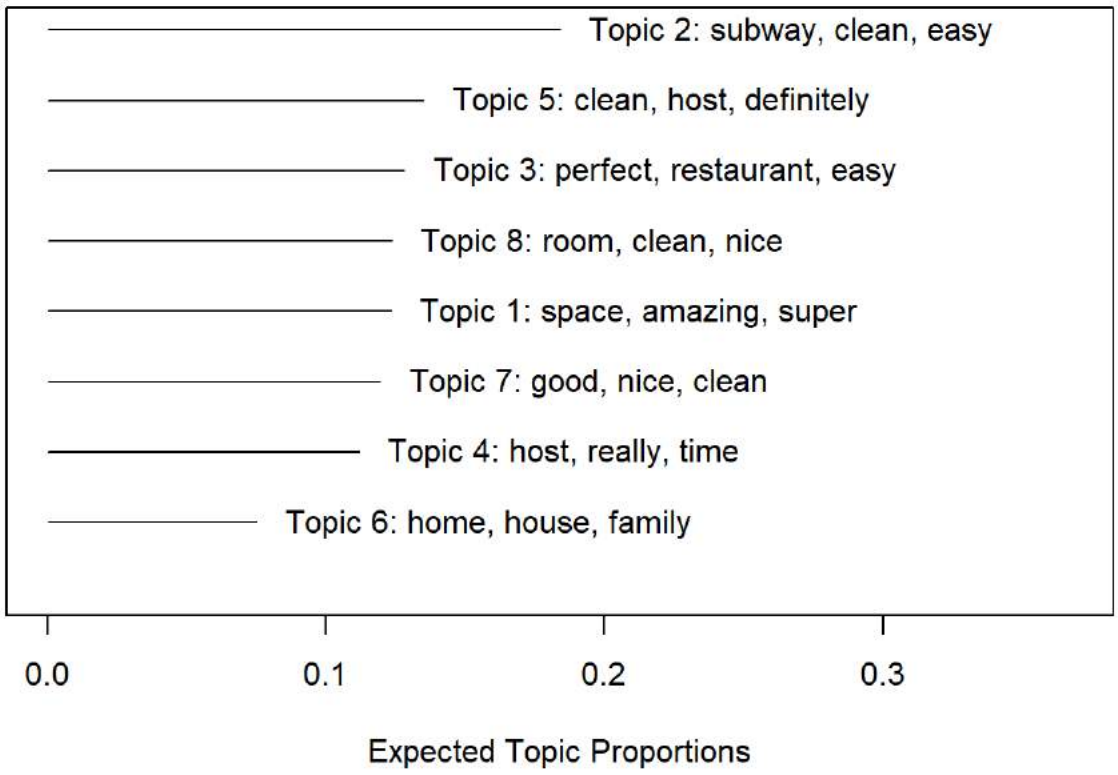
```
##  Topic 2: subway, clean, easy, nice, close
##  Topic 3: perfect, restaurant, easy, clean, host
##  Topic 4: host, really, time, friendly, home
##  Topic 5: clean, host, definitely, comfortable, nice
##  Topic 6: home, house, family, beautiful, host
##  Topic 7: good, nice, clean, night, space
##  Topic 8: room, clean, nice, bathroom, host
##
.....................................................................................

## Completed E-Step (6 seconds).
## Completed M-Step.
## Model Converged
```

```
summary(newyorkfit)
```

```
## A topic model with 8 topics, 21114 documents and a 1977 word dictionary.
```

```
## Topic 1 Top Words:
##       Highest Prob: space, amazing, super, beautiful, perfect, host, definitely
##       FREX: view, loft, stylish, rooftop, balcony, roof, terrace
##       Lift: recommended, views, loft, rooftop, terrace, balcony, skyline
##       Score: recommended, loft, view, rooftop, beautiful, stylish, bar
## Topic 2 Top Words:
##       Highest Prob: subway, clean, easy, nice, close, comfortable, well
##       FREX: store, grocery, subway, supermarket, spacious, walk, train
##       Lift: delighted, grocery, supermarket, washer, dishwasher, grocer, store
##       Score: delighted, subway, restaurant, station, spacious, train, clean
## Topic 3 Top Words:
##       Highest Prob: perfect, restaurant, easy, clean, host, good, close
##       FREX: studio, stair, noise, central, building, key, bar
##       Lift: quieter, walkup, fifth, theater, village, theatre, exercise
##       Score: quieter, studio, bar, restaurant, building, village, perfect
## Topic 4 Top Words:
##       Highest Prob: host, really, time, friendly, home, nice, subway
##       FREX: cat, dog, tip, advice, welcoming, guy, friendly
##       Lift: immensely, kitty, cat, dog, funny, animal, chat
##       Score: cat, immensely, dog, friendly, welcoming, home, tip
## Topic 5 Top Words:
##       Highest Prob: clean, host, definitely, comfortable, nice, home, time
##       FREX: airport, bus, hospitality, snack, thoughtful, cozy, flight
##       Lift: mum, layover, snacks, toothbrush, candy, toothpaste, airport
##       Score: mum, clean, host, snack, bus, airport, parking
## Topic 6 Top Words:
##       Highest Prob: home, house, family, beautiful, host, wonderful, comfortable
##       FREX: family, garden, brownstone, breakfast, kid, backyard, beach
##       Lift: memorable, toy, beach, townhouse, brownstone, garden, historic
##       Score: memorable, family, house, home, brownstone, beautiful, beach
## Topic 7 Top Words:
##       Highest Prob: good, nice, clean, night, space, people, bed
##       FREX: dirty, basement, staff, bad, toilet, however, group
##       Lift: steal, mold, dirt, horrible, stain, bug, rude
##       Score: steal, group, parking, house, price, dirty, ferry
## Topic 8 Top Words:
##       Highest Prob: room, clean, nice, bathroom, host, subway, good
##       FREX: room, bathroom, private, privacy, roommate, station, guest
##       Lift: mother, room, roommate, housemate, share, private, restroom
##       Score: mother, room, bathroom, roommate, house, private, clean
```

```
plot(newyorkfit)
```




```
newyorkfit_topics<-tidy(newyorkfit,matrix="beta")
```

```
#And we get the top 10 terms:

newyorkfit_terms<-newyorkfit_topics %>%
  group_by(topic) %>%
  slice_max(beta,n=10) %>%
  ungroup() %>%
  arrange(topic,desc(beta))
newyorkfit_terms
```

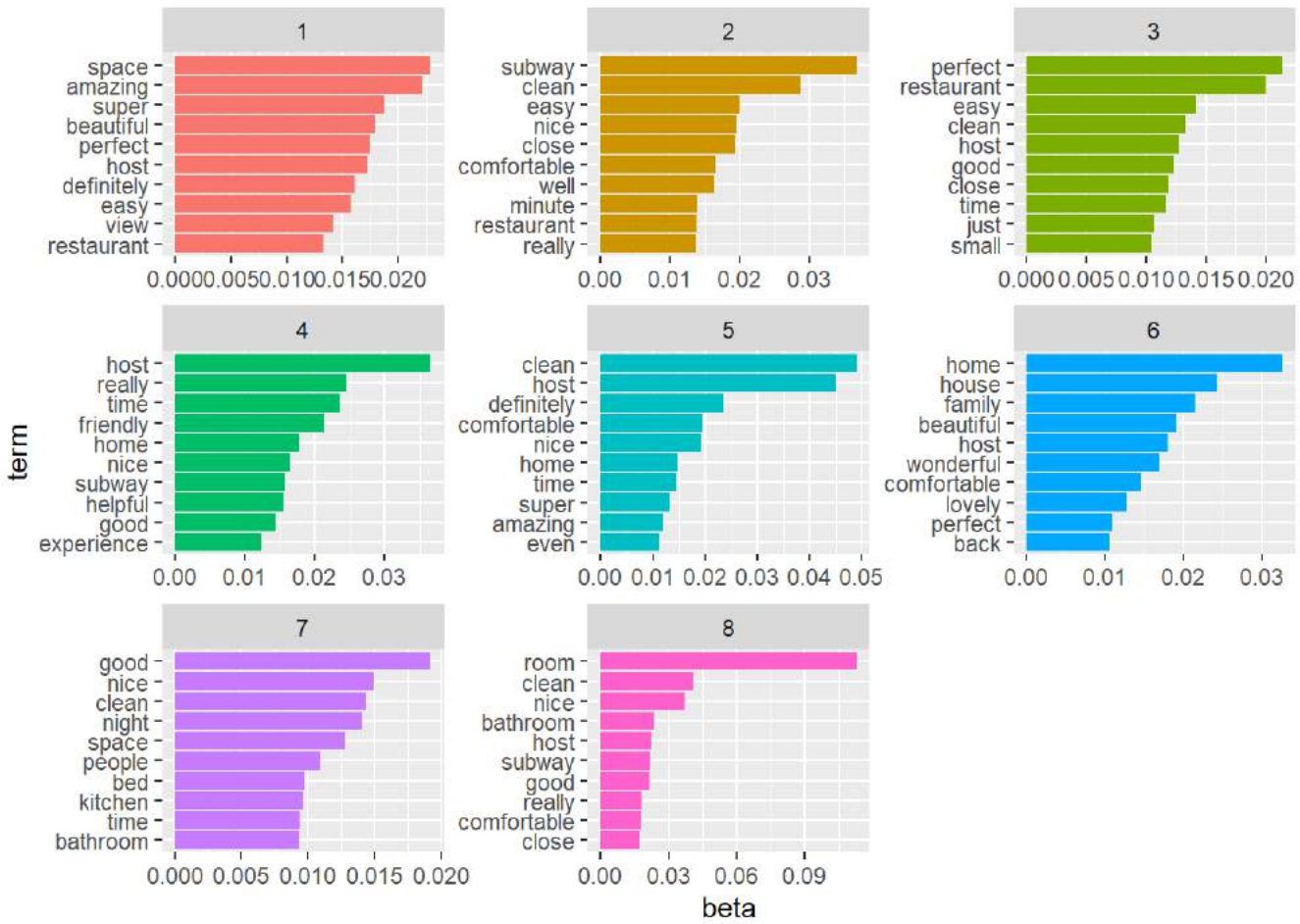
```
## # A tibble: 80 x 3
##    topic term         beta
##    <int> <chr>       <dbl>
##  1     1 space      0.0229
##  2     1 amazing    0.0222
##  3     1 super      0.0187
##  4     1 beautiful  0.0179
##  5     1 perfect    0.0175
##  6     1 host       0.0173
##  7     1 definitely 0.0161
##  8     1 easy       0.0157
##  9     1 view       0.0142
## 10     1 restaurant 0.0133
## # ... with 70 more rows
```

***Topic labeling***

```
#we look at the most frequent words appearing in each topic and give a name to each t
opic.
topic_labels <- paste0("topic_",1:8)
topic_labels <- c("View from the listing", "Qaulity of rooms","Spaciousness of the ho
me","Restaurants nearby", "Friendlyness of the hosts", "Cleanliness of the listing",
"Comfort in living", "Closeness to Subway")
```

And we can plot the results:

```
newyorkfit_terms <-  newyorkfit_terms %>%
 mutate(topic_label = case_when((topic == 1) ~ topic_labels[1],
                                (topic == 2) ~ topic_labels[2],
                                (topic == 3) ~ topic_labels[3],
                                (topic == 4) ~ topic_labels[4],
                                (topic == 5) ~ topic_labels[5],
                                (topic == 6) ~ topic_labels[6],
                                (topic == 7) ~ topic_labels[7],
                                (topic == 8) ~ topic_labels[8]
                                ))
newyorkfit_terms %>%
mutate(term = reorder_within(term, beta, topic)) %>%
ggplot(aes(beta, term, fill = factor(topic))) +
geom_col(show.legend = FALSE) +
facet_wrap(~ topic, scales = "free") +
scale_y_reordered()
```



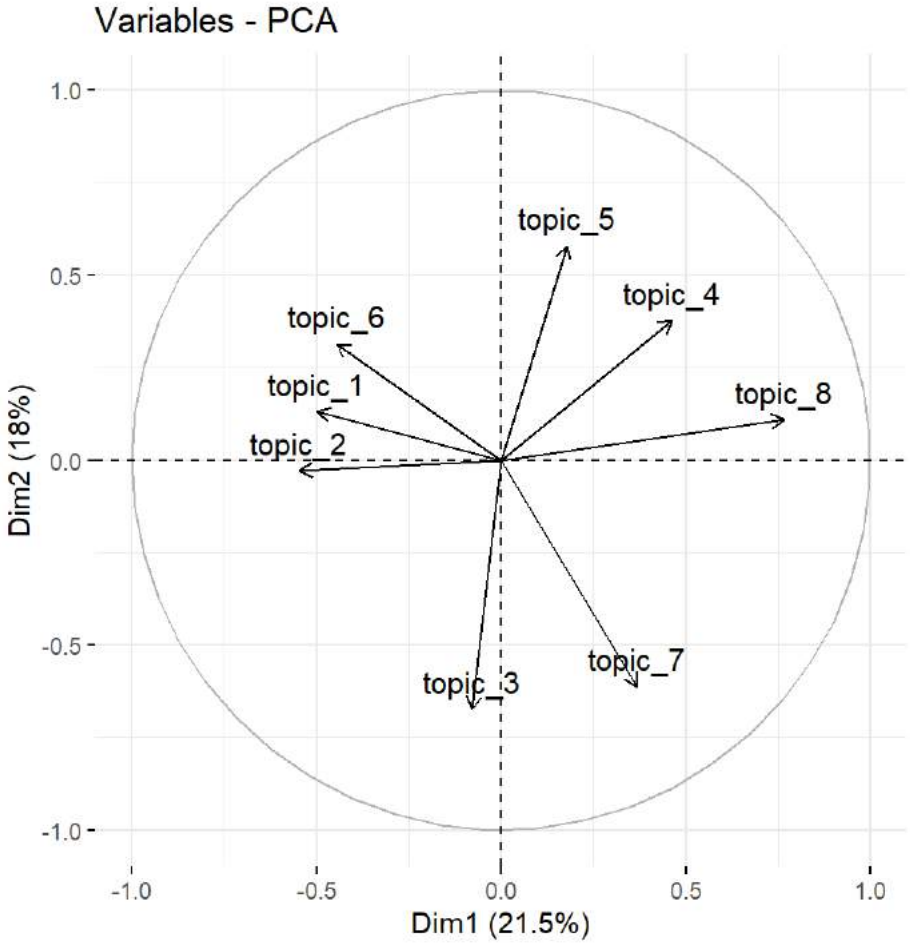
```
# #install.packages("wordcloud")
# library(wordcloud)
# stm::cloud(newyorkfit,topic = 1)
# stm::cloud(newyorkfit,topic = 2)
# stm::cloud(newyorkfit,topic = 3)
# stm::cloud(newyorkfit,topic = 4)
# stm::cloud(newyorkfit,topic = 5)
# stm::cloud(newyorkfit,topic = 6)
# stm::cloud(newyorkfit,topic = 7)
# stm::cloud(newyorkfit,topic = 8)
# stm::cloud(newyorkfit,topic = 9)
# stm::cloud(newyorkfit,topic = 10)
```

```
# mod = newyorkfit
# docs = output$documents
# stm::toLDAvis(
# mod,
# docs,
# R = 30,
# plot.opts = list(xlab = "PC1", ylab = "PC2"),
# lambda.step = 0.1,
# out.dir = tempfile(),
# open.browser = interactive(),
# as.gist = FALSE,
# reorder.topics = TRUE
# )
```

```
gamma_topics<-tidy(newyorkfit,matrix="gamma")



gamma_topics<-gamma_topics %>%
  pivot_wider(names_from=topic,values_from=gamma)

topic_labels <- paste0("topic_",1:8)
colnames(gamma_topics)<-c("document",topic_labels)

rownames(gamma_topics) <- gamma_topics$document

gamma_topics$document <- NULL
gamma_topics<-as.data.frame(gamma_topics)
head(gamma_topics)
```

```
##       topic_1     topic_2    topic_3    topic_4    topic_5     topic_6
## 1 0.168562918 0.019429327 0.45356380 0.16432223 0.05010869 0.016957638
## 2 0.005271278 0.181159793 0.13768181 0.10657477 0.11115604 0.186615204
## 3 0.017551530 0.010954960 0.01084113 0.76527827 0.02452295 0.011513378
## 4 0.196376322 0.049127645 0.03787470 0.05605068 0.02995659 0.496785562
## 5 0.001792272 0.004806522 0.34960311 0.06555366 0.02790075 0.006039486
## 6 0.005144037 0.008130834 0.04841625 0.22103418 0.03410827 0.401947558
##       topic_7    topic_8
## 1 0.085369659 0.04168574
## 2 0.147653114 0.12388799
## 3 0.034223045 0.12511474
## 4 0.056922451 0.07690605
## 5 0.254991587 0.28931260
## 6 0.006088099 0.27513077
```

```
#install.packages("factoextra")
library(factoextra)
pcah2 <- FactoMineR::PCA(gamma_topics,graph = FALSE)
factoextra::fviz_pca_var(pcah2)
```




***Expected topic proportions***

We extract the theta matrix from the fitted object

```
convergence_theta <- as.data.frame(newyorkfit$theta)

topic_summary <- summary(newyorkfit)
```

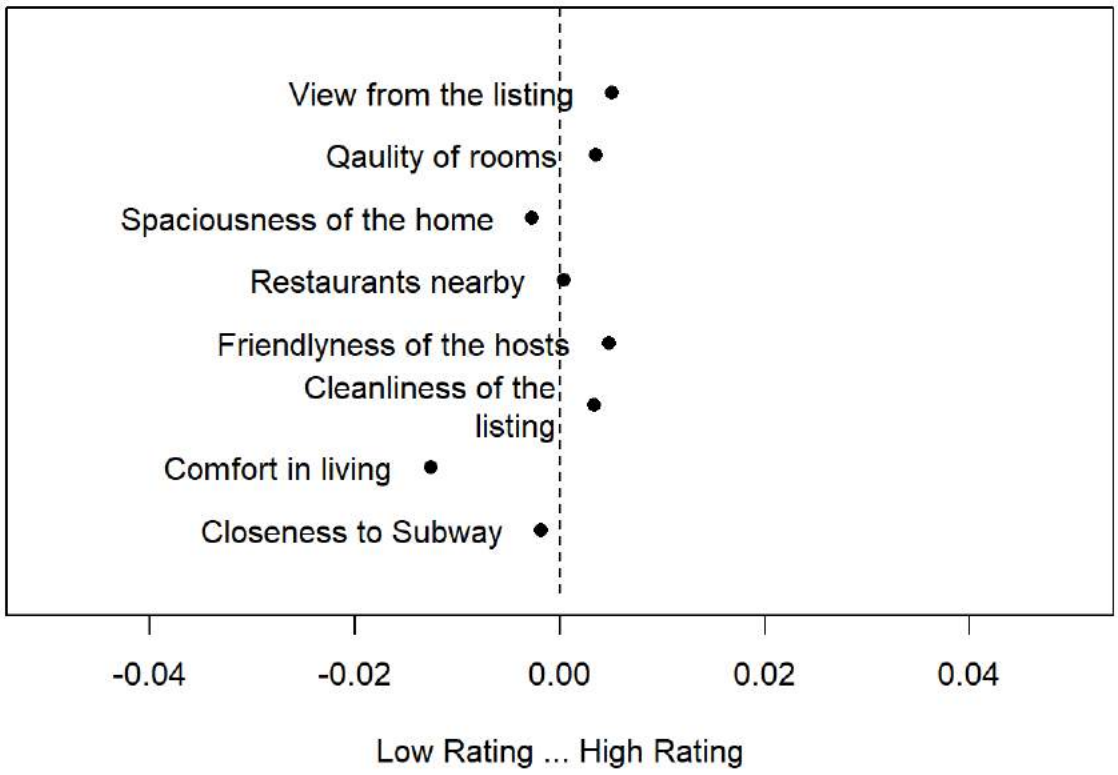
```
## A topic model with 8 topics, 21114 documents and a 1977 word dictionary.
```

```
topic_proportions <- colMeans(newyorkfit$theta)
```

# STM Effect Estimation

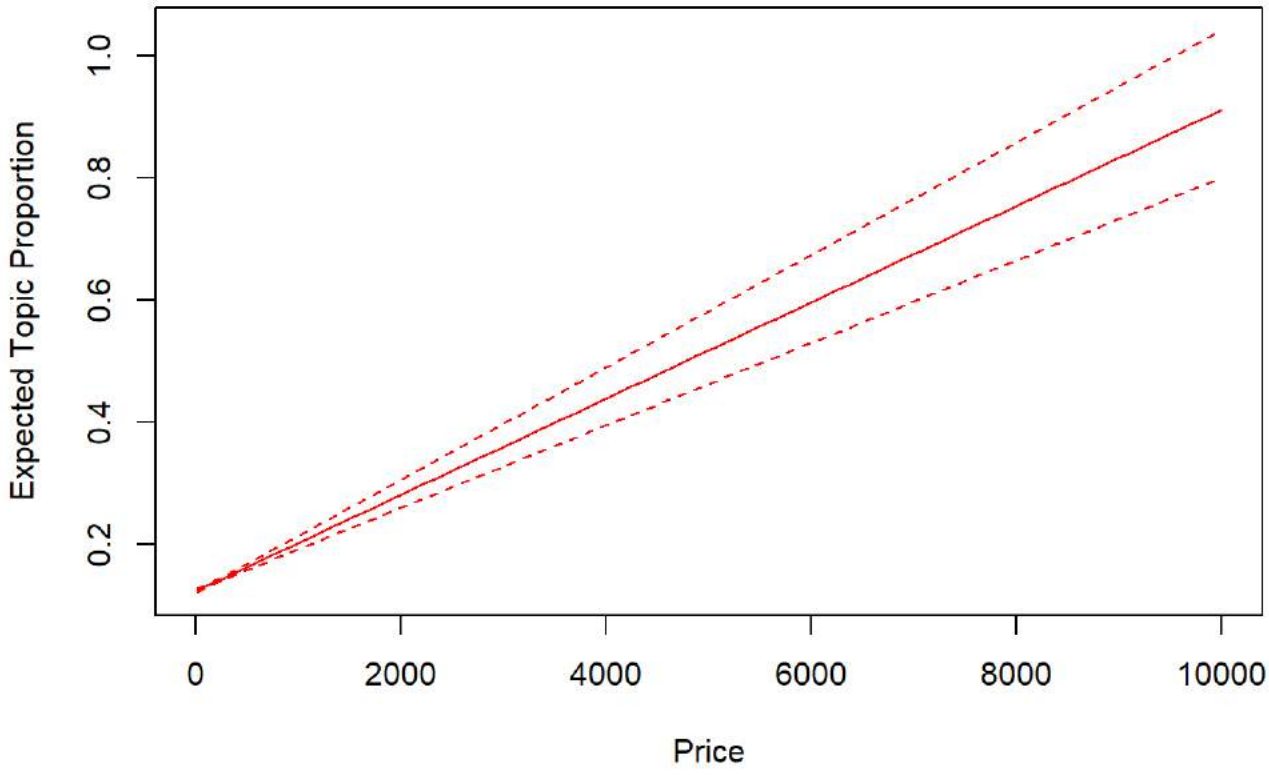
```
#
# Estimate the effects of price and review score
# on topic probability

effects <- estimateEffect(~review_scores_rating+price,
                          stmobj = newyorkfit,
                          metadata = output$meta)
```

```
# Effect of review score on topic
# probability
topic_labels <- c("View from the listing", "Qaulity of rooms","Spaciousness of the ho
me","Restaurants nearby", "Friendlyness of the hosts", "Cleanliness of the listing",
"Comfort in living", "Closeness to Subway")
plot(effects, covariate = "review_scores_rating",
     topics = c(1:8),
     model = newyorkfit, method = "difference",
     cov.value1 = "100", cov.value2 = "0",
     xlab = "Low Rating ... High Rating",
     xlim = c(-0.05,0.05),
     main = "",
     ci.level = 0.05,
     custom.labels =topic_labels[c(1:8)],
     labeltype = "custom")
```

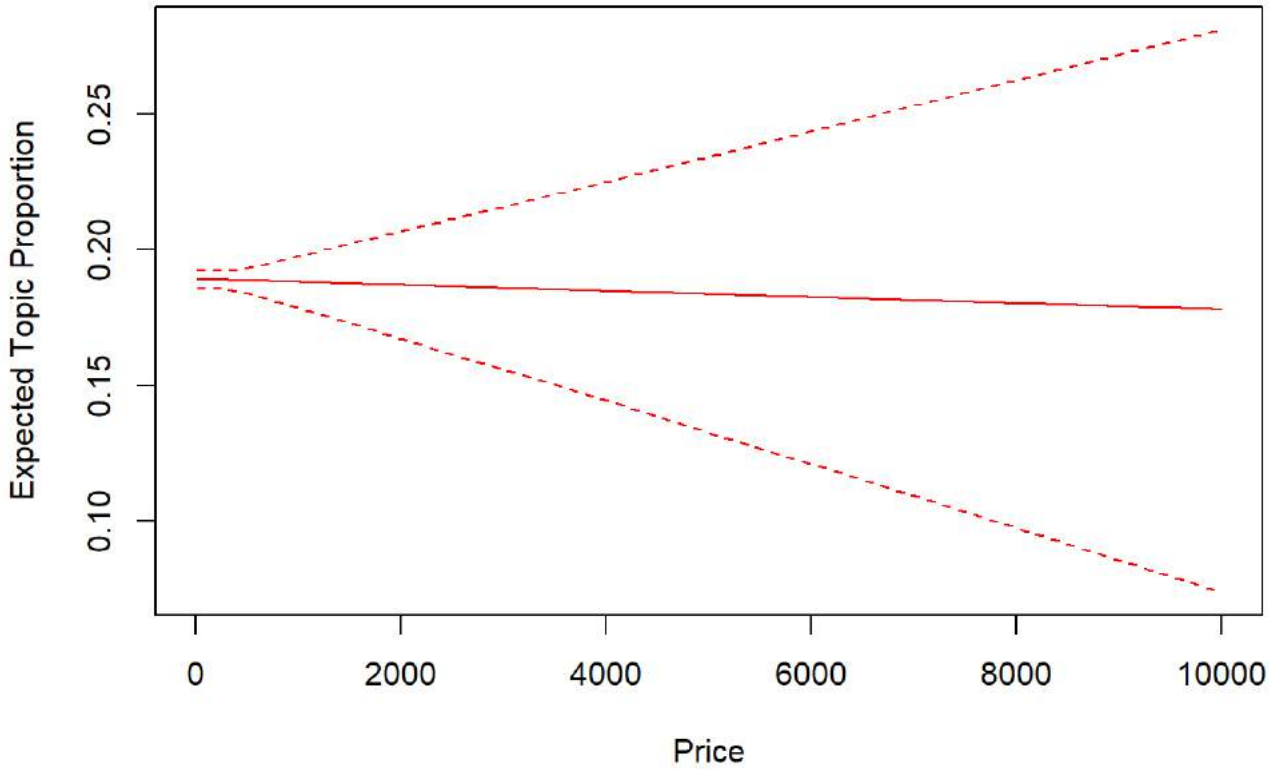



```
# Effect of price on topic
# probability treating price as
# a continuous variable.
#
# Ploting each topic separately
# we can see that some of them increase
# substantially with the price

for(i in 1:length(topic_labels)){
plot(effects, covariate = "price",
     topics = i,
     model = airbnbfit, method = "continuous",
     # For this plotting we get the uper quantile
     # and low quantile of the price
     xlab = "Price",
     # xlim = c(0,800),
     main = topic_labels[i],
     printlegend = FALSE,
     custom.labels =topic_labels[i],
     labeltype = "custom")
}
```

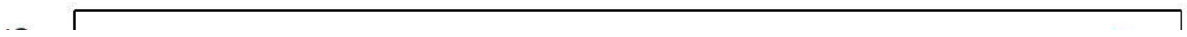
## View from the listing

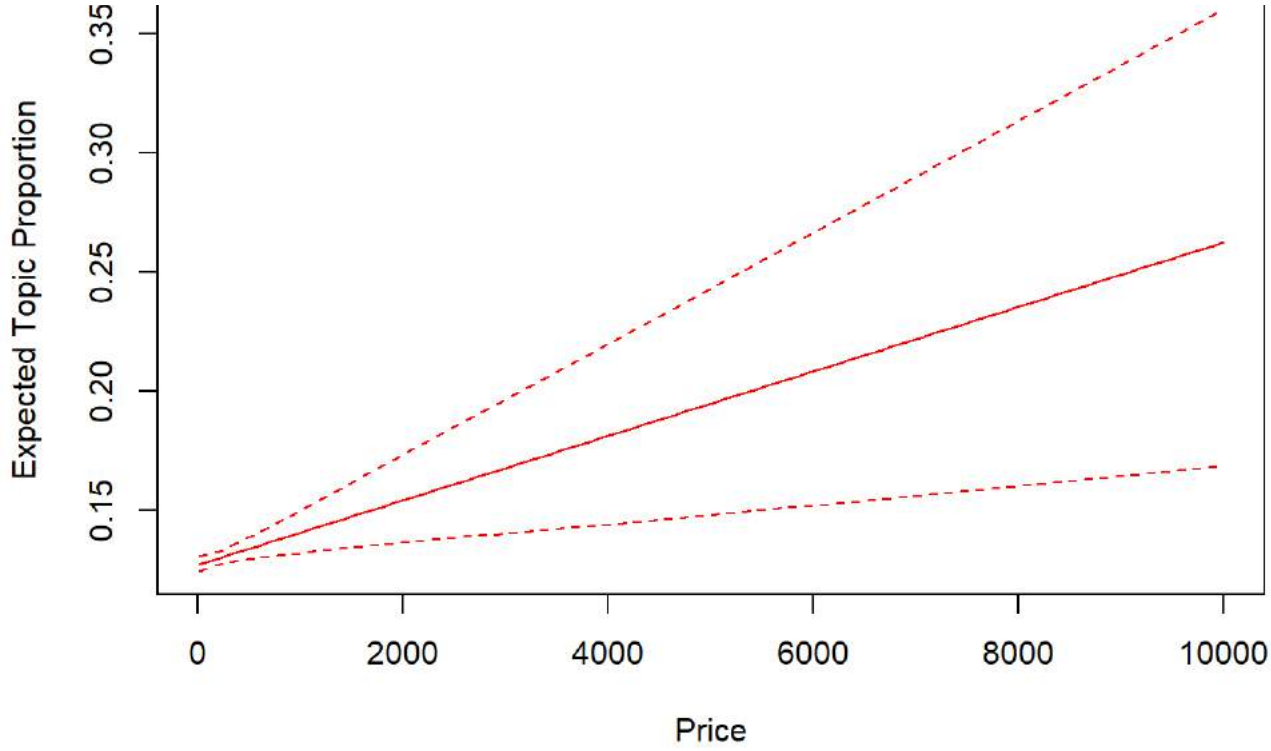



## Qaulity of rooms

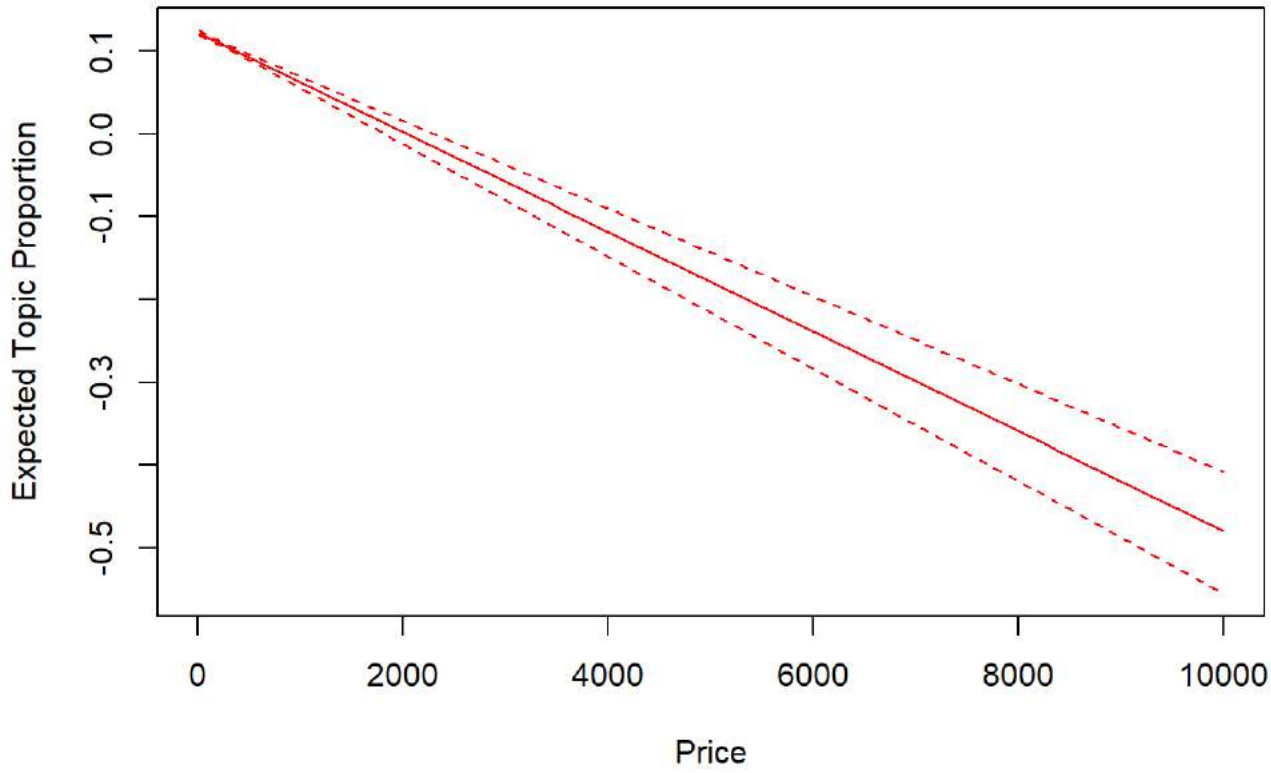


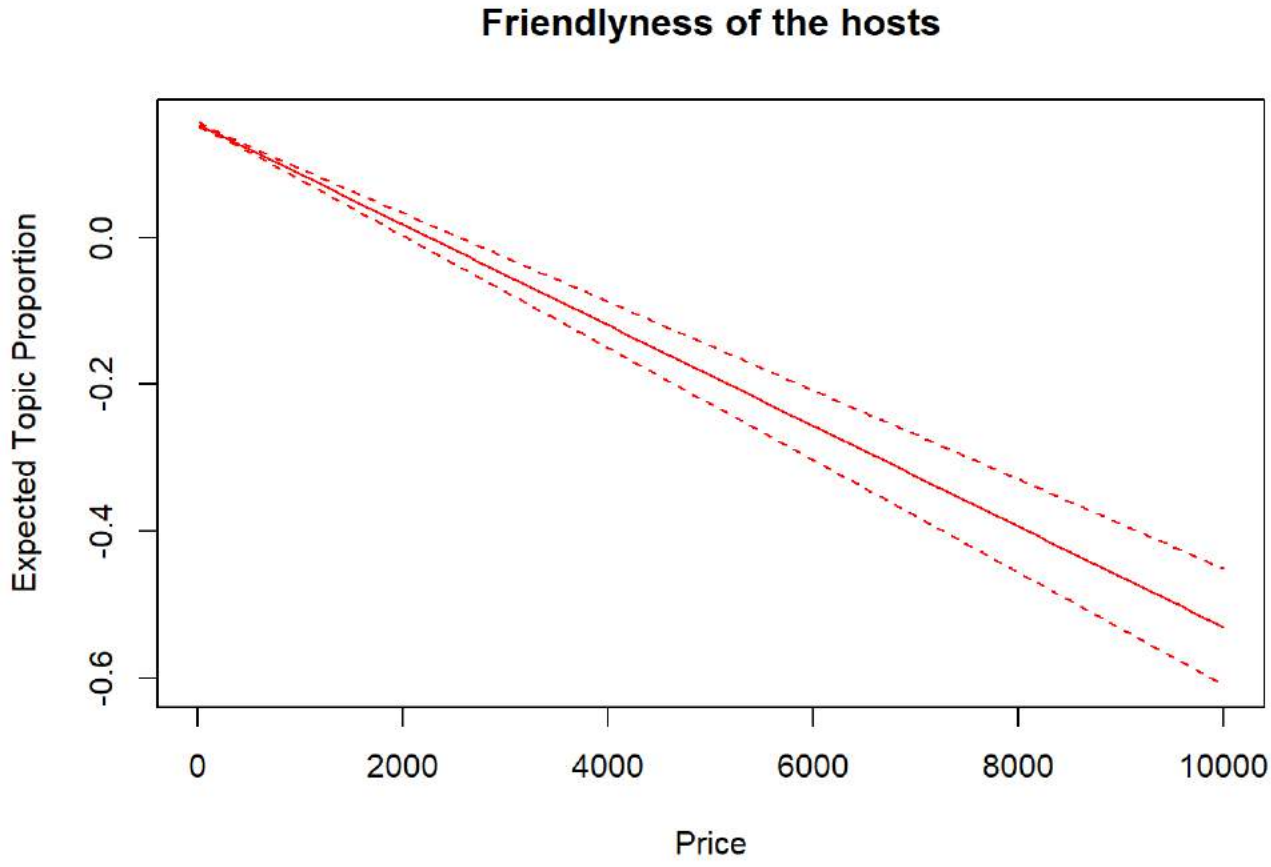

## Spaciousness of the home

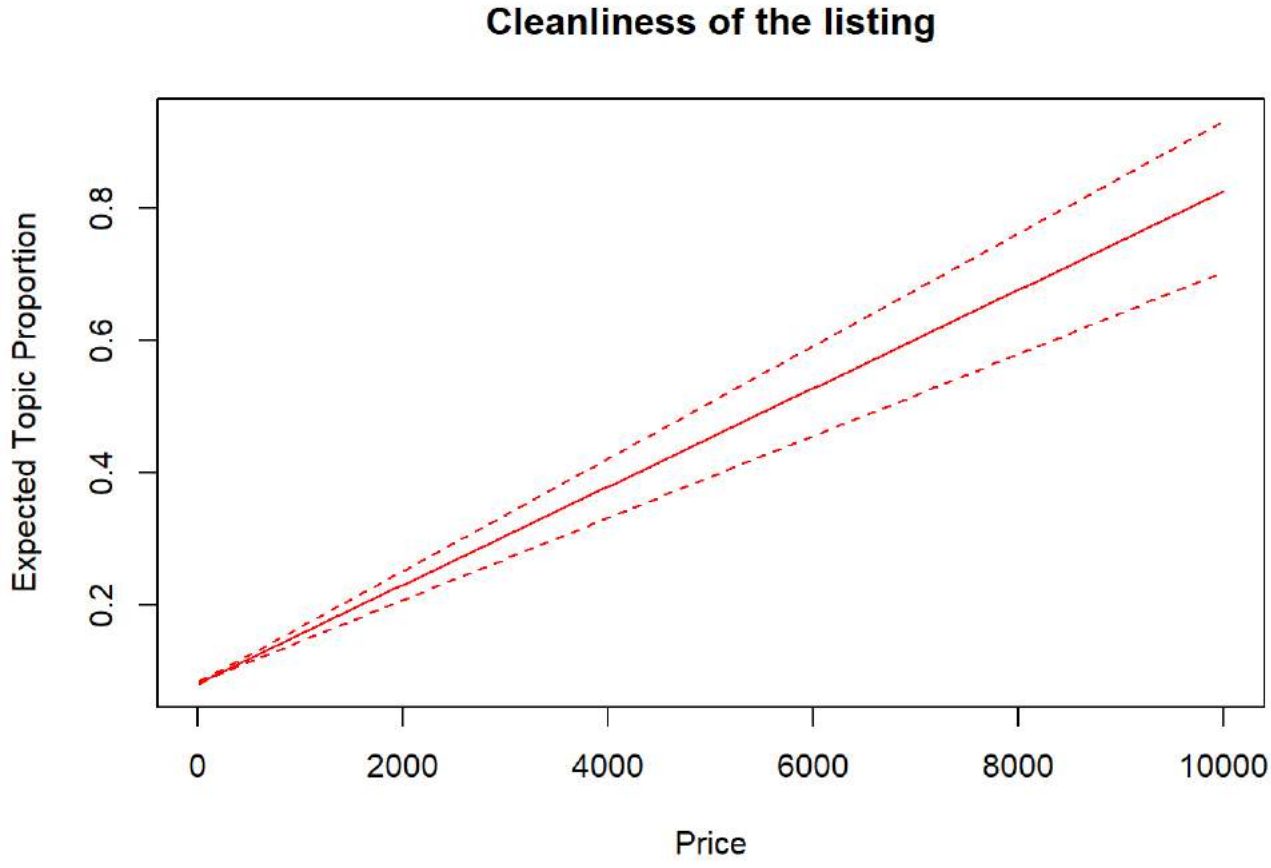


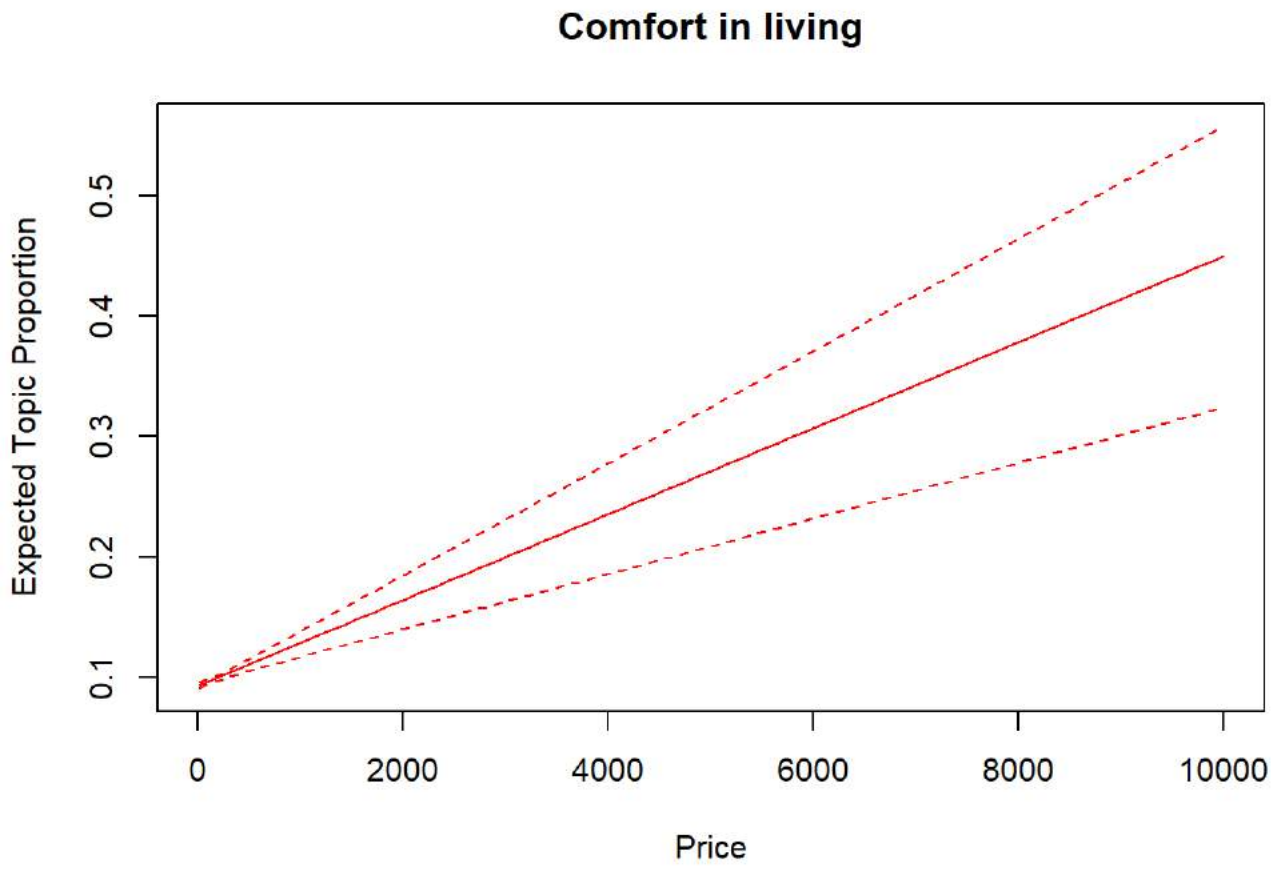


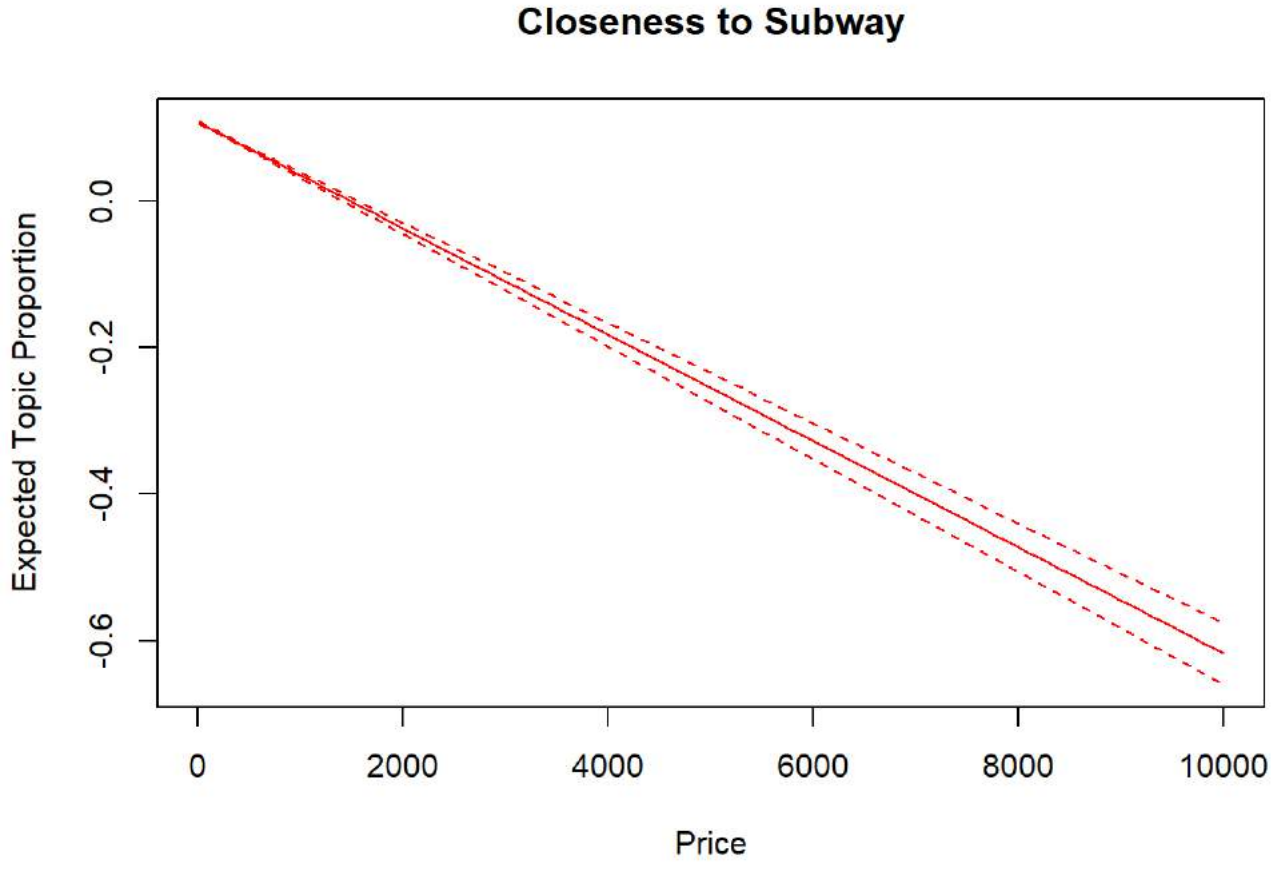

**Restaurants nearby**



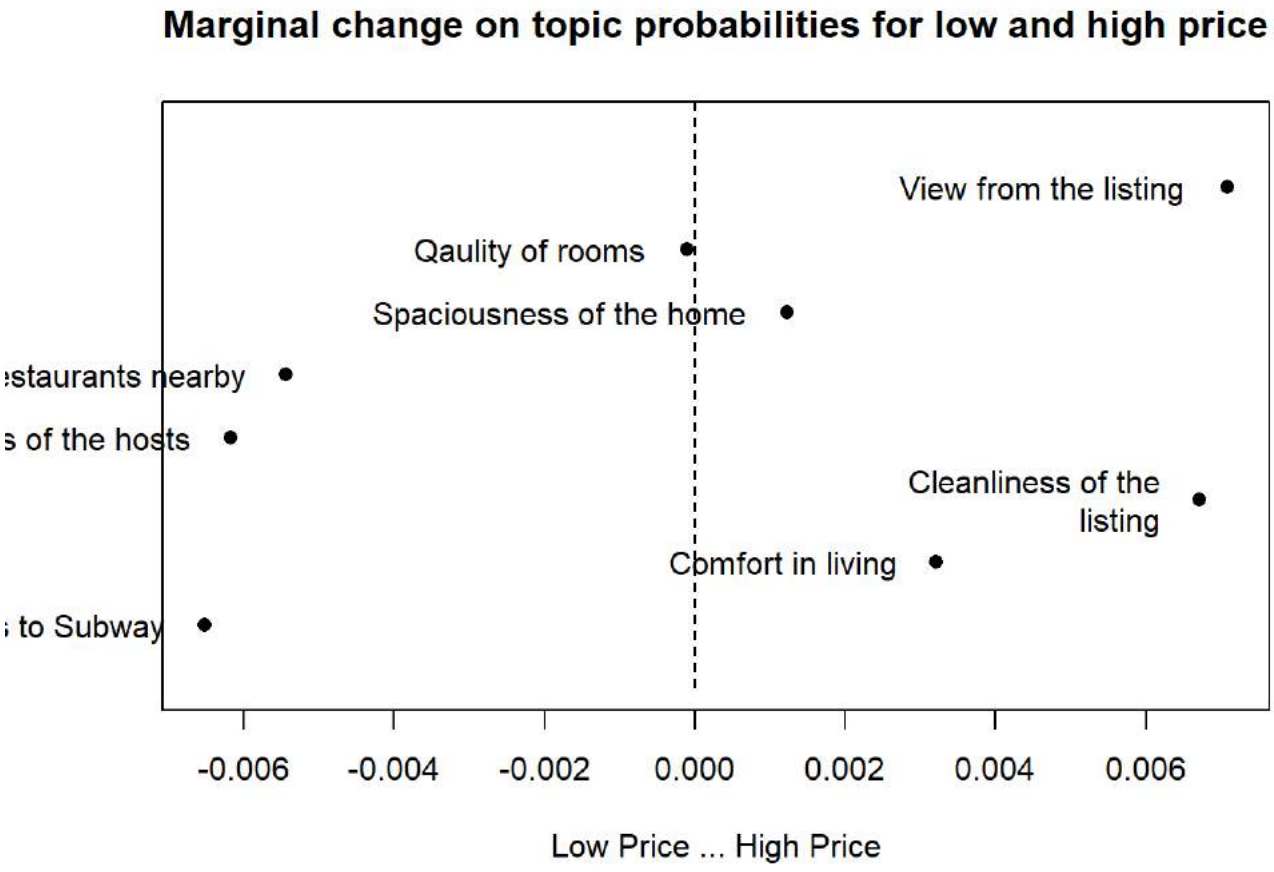

Friendlyness of the hosts
Expected Topic Proportion
-0.6
-0.4
-0.2
0.0
0
2000
4000
6000
8000
10000
Price


Cleanliness of the listing
Expected Topic Proportion
0.2
0.4
0.6
0.8
0
2000
4000
6000
8000
10000
Price


Comfort in living
Expected Topic Proportion
0.1
0.2
0.3
0.4
0.5
0
2000
4000
6000
8000
10000
Price


Closeness to Subway
Expected Topic Proportion
-0.6
-0.4
-0.2
0.0
0
2000
4000
6000
8000
10000
Price

```
# Lets also plot them as a contrast between
# the minimum and maximum price
# for that we need the margins of the upper and lower quantile
margin1 <- as.numeric(quantile(output$meta$price)[2])
margin2 <- as.numeric(quantile(output$meta$price)[4])


plot(effects, covariate = "price",
     topics = c(1:8),
     model = newyorkfit, method = "difference",
     cov.value1 = margin2, cov.value2 = margin1,
     xlab = "Low Price ... High Price",
     # xlim = c(-0.01,0.01),
     main = "Marginal change on topic probabilities for low and high price",
     custom.labels =topic_labels,
     ci.level = 0.05,
     labeltype = "custom")
```



```
# Finding number  amenities, bathrooms,
i <- 1
for (i in 1:length(listings_detailed$host_verifications)) {

  listings_detailed$num_amenities[i] <- length(strsplit(listings_detailed$amenities
[i], split= " ")[[1]])
  listings_detailed$num_bath[i] <- as.numeric(gsub("[^0-9.]", "",listings_detailed$ba
throoms_text[i]))
    i <- i+1
}

all_sentiments <- readr::read_csv("sentiments.csv")

join_data <- listings_detailed %>% dplyr::select(listing_id, neighbourhood_cleansed,
 num_bath,num_amenities,beds, room_type)

sentiment <- all_sentiments %>% left_join(join_data)
sentiment <- sentiment %>% na.omit()
```

```
stm_object<- newyorkfit$theta
colnames(stm_object) <- c("View from the listing", "Qaulity of rooms","Spaciousness o
f the home","Restaurants nearby", "Friendlyness of the hosts", "Cleanliness of the li
sting", "Comfort in living", "Closeness to Subway")


causal_topic_df <- cbind(output$meta,stm_object)

causal_topic_df  %>%
  left_join(sentiment) %>%
  dplyr::select(-c(listing_id, comments_grouped_listing,X1)) %>%
  na.omit() -> regress_stm_price_review


# model1_review <- lm(review_scores_rating~mtopic1+bing_liu_sentiment,data=regress_st
m_price_review)
# model2_review <- lm(review_scores_rating~mtopic2+bing_liu_sentiment,data=regress_st
m_price_review)
# model3_review <- lm(review_scores_rating~mtopic3+bing_liu_sentiment,data=regress_st
m_price_review)
# model4_review <- lm(review_scores_rating~mtopic4+bing_liu_sentiment,data=regress_st
m_price_review)
# model5_review <- lm(review_scores_rating~mtopic5+bing_liu_sentiment,data=regress_st
m_price_review)
# model6_review <- lm(review_scores_rating~mtopic6+bing_liu_sentiment,data=regress_st
m_price_review)
# model7_review <- lm(review_scores_rating~mtopic7+bing_liu_sentiment,data=regress_st
m_price_review)
# model8_review <- lm(review_scores_rating~mtopic8+bing_liu_sentiment,data=regress_st
m_price_review)
# model9_review <- lm(review_scores_rating~mtopic9+bing_liu_sentiment,data=regress_st
m_price_review)
# model10_review <- lm(review_scores_rating~mtopic10+bing_liu_sentiment,data=regress_
stm_price_review)
#
# summary(model1_review)
# summary(model2_review)
# summary(model3_review)
# summary(model4_review)
# summary(model5_review)
# summary(model6_review)
# summary(model7_review)
# summary(model8_review)
# summary(model9_review)
# summary(model10_review)
#
#
#
# stargazer::stargazer(model1_review,model2_review,
#                      model3_review,model4_review,
#                      model5_review,model6_review,
#                      model7_review,model8_review,
# #                      model9_review,model10_review,
#                      type = "text")
```

```
model_search_review <- lm(review_scores_rating~.,data=regress_stm_price_review)
MASS::stepAIC(model_search_review)
```

Start: AIC=25187.85 review_scores_rating ~ price + `View from the listing` + `Qaulity of rooms` + `Spaciousness of the home` + `Restaurants nearby` + `Friendlyness of the hosts` + `Cleanliness of the listing` + `Comfort in living` + `Closeness to Subway` + total_excl_count + bing_liu_sentiment + nrc_sentiment + afinn_sentiment + loughran_sentiment + neighbourhood_cleansed + num_bath + num_amenities + beds + room_type

Step: AIC=25187.85 review_scores_rating ~ price + `View from the listing` + `Qaulity of rooms` + `Spaciousness of the home` + `Restaurants nearby` + `Friendlyness of the hosts` + `Cleanliness of the listing` + `Comfort in living` + total_excl_count + bing_liu_sentiment + nrc_sentiment + afinn_sentiment + loughran_sentiment + neighbourhood_cleansed + num_bath + num_amenities + beds + room_type

```
                                Df Sum of Sq    RSS   AIC
```

- neighbourhood_cleansed 197 4387.5 147324 25064
- beds 1 2.3 142938 25186
- nrc_sentiment 1 6.1 142942 25186
- `Spaciousness of the home` 1 8.5 142944 25186
- loughran_sentiment 1 10.4 142946 25187
- price 1 14.9 142951 25187
- total_excl_count 1 15.5 142952 25187 142936 25188
- `Restaurants nearby` 1 49.0 142985 25189
- room_type 3 177.3 143113 25193
- afinn_sentiment 1 226.1 143162 25200
- num_bath 1 255.8 143192 25202
- bing_liu_sentiment 1 685.3 143621 25229
- `Qaulity of rooms` 1 1221.8 144158 25262
- `Cleanliness of the listing` 1 1355.8 144292 25270
- num_amenities 1 1906.4 144842 25304
- `View from the listing` 1 3164.6 146101 25381
- `Friendlyness of the hosts` 1 3795.6 146732 25420
- `Comfort in living` 1 7222.4 150158 25626

Step: AIC=25063.69 review_scores_rating ~ price + `View from the listing` + `Qaulity of rooms` + `Spaciousness of the home` + `Restaurants nearby` + `Friendlyness of the hosts` + `Cleanliness of the listing` + `Comfort in living` + total_excl_count + bing_liu_sentiment + nrc_sentiment + afinn_sentiment + loughran_sentiment + num_bath + num_amenities + beds + room_type

```
                                Df Sum of Sq    RSS   AIC
```

- `Spaciousness of the home` 1 0.1 147324 25062
- nrc_sentiment 1 2.0 147326 25062
- beds 1 2.6 147326 25062
- total_excl_count 1 4.3 147328 25062
- price 1 17.9 147341 25063
- loughran_sentiment 1 19.2 147343 25063 147324 25064
- `Restaurants nearby` 1 38.6 147362 25064
- room_type 3 233.7 147557 25072
- afinn_sentiment 1 223.0 147547 25075
- num_bath 1 258.5 147582 25077

- bing_liu_sentiment 1 813.3 148137 25111
- `Qaulity of rooms` 1 1153.4 148477 25131
- `Cleanliness of the listing` 1 1378.3 148702 25145
- num_amenities 1 1958.9 149283 25180
- `View from the listing` 1 3712.8 151036 25284
- `Friendlyness of the hosts` 1 4398.8 151722 25324
- `Comfort in living` 1 7097.9 154421 25482

Step: AIC=25061.7 review_scores_rating ~ price + `View from the listing` + `Qaulity of rooms` + `Restaurants nearby` + `Friendlyness of the hosts` + `Cleanliness of the listing` + `Comfort in living` + total_excl_count + bing_liu_sentiment + nrc_sentiment + afinn_sentiment + loughran_sentiment + num_bath + num_amenities + beds + room_type

```
                          Df Sum of Sq    RSS   AIC
```

- nrc_sentiment 1 2.1 147326 25060
- beds 1 2.5 147326 25060
- total_excl_count 1 4.2 147328 25060
- price 1 18.6 147342 25061
- loughran_sentiment 1 19.3 147343 25061 147324 25062
- `Restaurants nearby` 1 82.8 147406 25065
- afinn_sentiment 1 223.0 147547 25073
- room_type 3 319.8 147643 25075
- num_bath 1 259.2 147583 25075
- bing_liu_sentiment 1 815.8 148139 25109
- num_amenities 1 1960.4 149284 25178
- `Cleanliness of the listing` 1 2781.0 150105 25227
- `Qaulity of rooms` 1 3193.8 150517 25251
- `View from the listing` 1 8781.9 156106 25577
- `Friendlyness of the hosts` 1 11762.2 159086 25745
- `Comfort in living` 1 18639.7 165963 26123

Step: AIC=25059.82 review_scores_rating ~ price + `View from the listing` + `Qaulity of rooms` + `Restaurants nearby` + `Friendlyness of the hosts` + `Cleanliness of the listing` + `Comfort in living` + total_excl_count + bing_liu_sentiment + afinn_sentiment + loughran_sentiment + num_bath + num_amenities + beds + room_type

```
                          Df Sum of Sq    RSS   AIC
```

- beds 1 2.6 147328 25058
- total_excl_count 1 3.9 147330 25058
- loughran_sentiment 1 18.3 147344 25059
- price 1 18.7 147344 25059 147326 25060
- `Restaurants nearby` 1 81.9 147408 25063
- afinn_sentiment 1 221.2 147547 25071
- room_type 3 319.2 147645 25073
- num_bath 1 259.1 147585 25074
- bing_liu_sentiment 1 1037.7 148363 25121
- num_amenities 1 1962.2 149288 25176
- `Cleanliness of the listing` 1 2778.9 150105 25225
- `Qaulity of rooms` 1 3202.5 150528 25250
- `View from the listing` 1 8780.0 156106 25575
- `Friendlyness of the hosts` 1 11765.7 159091 25744

- `Comfort in living` 1 18642.8 165969 26121

Step: AIC=25057.98 review_scores_rating ~ price + `View from the listing` + `Qaulity of rooms` + `Restaurants nearby` + `Friendlyness of the hosts` + `Cleanliness of the listing` + `Comfort in living` + total_excl_count + bing_liu_sentiment + afinn_sentiment + loughran_sentiment + num_bath + num_amenities + room_type

```
                            Df Sum of Sq    RSS    AIC
```

- total_excl_count 1 4.0 147332 25056
- price 1 18.0 147346 25057
- loughran_sentiment 1 18.4 147347 25057 147328 25058
- `Restaurants nearby` 1 80.4 147409 25061
- afinn_sentiment 1 221.4 147550 25069
- num_bath 1 267.0 147595 25072
- room_type 3 342.4 147671 25073
- bing_liu_sentiment 1 1036.9 148365 25119
- num_amenities 1 1970.3 149299 25175
- `Cleanliness of the listing` 1 2885.3 150214 25229
- `Qaulity of rooms` 1 3258.9 150587 25251
- `View from the listing` 1 8792.2 156120 25573
- `Friendlyness of the hosts` 1 11774.1 159102 25742
- `Comfort in living` 1 19378.6 166707 26159

Step: AIC=25056.22 review_scores_rating ~ price + `View from the listing` + `Qaulity of rooms` + `Restaurants nearby` + `Friendlyness of the hosts` + `Cleanliness of the listing` + `Comfort in living` + bing_liu_sentiment + afinn_sentiment + loughran_sentiment + num_bath + num_amenities + room_type

```
                            Df Sum of Sq    RSS    AIC
```

- price 1 17.5 147350 25055
- loughran_sentiment 1 21.4 147354 25056 147332 25056
- `Restaurants nearby` 1 80.7 147413 25059
- num_bath 1 266.3 147599 25070
- room_type 3 342.2 147675 25071
- afinn_sentiment 1 476.2 147809 25083
- bing_liu_sentiment 1 1128.9 148461 25122
- num_amenities 1 1976.1 149308 25173
- `Cleanliness of the listing` 1 2917.8 150250 25229
- `Qaulity of rooms` 1 3257.0 150589 25249
- `View from the listing` 1 8797.1 156129 25572
- `Friendlyness of the hosts` 1 11837.1 159169 25744
- `Comfort in living` 1 19459.0 166791 26161

Step: AIC=25055.28 review_scores_rating ~ `View from the listing` + `Qaulity of rooms` + `Restaurants nearby` + `Friendlyness of the hosts` + `Cleanliness of the listing` + `Comfort in living` + bing_liu_sentiment + afinn_sentiment + loughran_sentiment + num_bath + num_amenities + room_type

```
                            Df Sum of Sq    RSS    AIC
```

- loughran_sentiment 1 21.0 147371 25055 147350 25055
- `Restaurants nearby` 1 78.2 147428 25058

- room_type 3 338.2 147688 25070
- num_bath 1 278.9 147629 25070
- afinn_sentiment 1 476.6 147826 25082
- bing_liu_sentiment 1 1144.2 148494 25122
- num_amenities 1 1965.8 149316 25172
- `Cleanliness of the listing` 1 3013.4 150363 25234
- `Qaulity of rooms` 1 3254.8 150605 25248
- `View from the listing` 1 9014.5 156364 25583
- `Friendlyness of the hosts` 1 11819.7 159170 25742
- `Comfort in living` 1 19508.9 166859 26163

Step: AIC=25054.55 review_scores_rating ~ `View from the listing` + `Qaulity of rooms` + `Restaurants nearby` + `Friendlyness of the hosts` + `Cleanliness of the listing` + `Comfort in living` + bing_liu_sentiment + afinn_sentiment + num_bath + num_amenities + room_type

```
                                Df Sum of Sq    RSS   AIC
```

147371 25055 - `Restaurants nearby` 1 75.9 147447 25057 - room_type 3 336.8 147708 25069 - num_bath 1 278.8 147650 25069 - afinn_sentiment 1 495.8 147867 25083 - bing_liu_sentiment 1 1710.7 149081 25156 - num_amenities 1 1981.2 149352 25172 - `Cleanliness of the listing` 1 3018.1 150389 25234 - `Qaulity of rooms` 1 3234.0 150605 25246 - `View from the listing` 1 9086.3 156457 25587 - `Friendlyness of the hosts` 1 11848.4 159219 25743 - `Comfort in living` 1 19610.9 166982 26168

Call: lm(formula = review_scores_rating ~ `View from the listing` + `Qaulity of rooms` + `Restaurants nearby` + `Friendlyness of the hosts` + `Cleanliness of the listing` + `Comfort in living` + bing_liu_sentiment + afinn_sentiment + num_bath + num_amenities + room_type, data = regress_stm_price_review)

Coefficients: (Intercept) `View from the listing`
87.403624 10.562595
`Qaulity of rooms` `Restaurants nearby`
5.900780 1.002313
`Friendlyness of the hosts` `Cleanliness of the listing`
11.306291 7.051446
`Comfort in living` bing_liu_sentiment
-16.812693 3.216825
afinn_sentiment num_bath
0.007196 0.512118
num_amenities room_typeHotel room
0.029582 1.199672
room_typePrivate room room_typeShared room
0.401019 1.079460

```
summary(lm(review_scores_rating ~ `View from the listing` +
    `Qaulity of rooms` + `Spaciousness of the home` + `Restaurants nearby` +
    `Friendlyness of the hosts` + `Cleanliness of the listing` +
    `Comfort in living` + bing_liu_sentiment + afinn_sentiment +
    num_bath + num_amenities + beds + room_type, data = regress_stm_price_review))
```

```
##
## Call:
## lm(formula = review_scores_rating ~ `View from the listing` +
##     `Qaulity of rooms` + `Spaciousness of the home` + `Restaurants nearby` +
##     `Friendlyness of the hosts` + `Cleanliness of the listing` +
##     `Comfort in living` + bing_liu_sentiment + afinn_sentiment +
##     num_bath + num_amenities + beds + room_type, data = regress_stm_price_review)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.933  -1.598   0.244   2.073  18.455
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   87.248002   0.667626 130.684  < 2e-16 ***
## `View from the listing`       10.695219   0.686883  15.571  < 2e-16 ***
## `Qaulity of rooms`             6.075279   0.715854   8.487  < 2e-16 ***
## `Spaciousness of the home`     0.181063   0.692745   0.261  0.79381
## `Restaurants nearby`           1.150665   0.705196   1.632  0.10278
## `Friendlyness of the hosts`   11.457513   0.695845  16.466  < 2e-16 ***
## `Cleanliness of the listing`   7.235019   0.758789   9.535  < 2e-16 ***
## `Comfort in living`          -16.616820   0.797433 -20.838  < 2e-16 ***
## bing_liu_sentiment             3.225405   0.317856  10.147  < 2e-16 ***
## afinn_sentiment                0.007210   0.001315   5.484 4.28e-08 ***
## num_bath                       0.527641   0.130788   4.034 5.52e-05 ***
## num_amenities                  0.029711   0.002721  10.921  < 2e-16 ***
## beds                          -0.017611   0.048295  -0.365  0.71538
## room_typeHotel room            1.196308   0.627663   1.906  0.05669 .
## room_typePrivate room          0.418004   0.148927   2.807  0.00501 **
## room_typeShared room           1.092379   0.400706   2.726  0.00642 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.067 on 8909 degrees of freedom
## Multiple R-squared:  0.5198, Adjusted R-squared:  0.519
## F-statistic: 642.8 on 15 and 8909 DF,  p-value: < 2.2e-16
```

```
model_search_price <- lm(price~.,data=regress_stm_price_review)
MASS::stepAIC(model_search_price)
```

```
## Start:  AIC=101927.3
## price ~ review_scores_rating + `View from the listing` + `Qaulity of rooms` +
##     `Spaciousness of the home` + `Restaurants nearby` + `Friendlyness of the hosts
` +
##     `Cleanliness of the listing` + `Comfort in living` + `Closeness to Subway` +
##     total_excl_count + bing_liu_sentiment + nrc_sentiment + afinn_sentiment +
##     loughran_sentiment + neighbourhood_cleansed + num_bath +
##     num_amenities + beds + room_type
##
##
## Step:  AIC=101927.3
## price ~ review_scores_rating + `View from the listing` + `Qaulity of rooms` +
##     `Spaciousness of the home` + `Restaurants nearby` + `Friendlyness of the hosts
` +
##     `Cleanliness of the listing` + `Comfort in living` + total_excl_count +
##     bing_liu_sentiment + nrc_sentiment + afinn_sentiment + loughran_sentiment +
##     neighbourhood_cleansed + num_bath + num_amenities + beds +
##     room_type
##
##                                Df Sum of Sq       RSS    AIC
## - neighbourhood_cleansed      197  19132237 794156818 101751
## - nrc_sentiment                 1      8550 775033131 101925
## - review_scores_rating          1     80745 775105326 101926
## - loughran_sentiment            1    161652 775186233 101927
## <none>                                     775024581 101927
## - afinn_sentiment               1    274789 775299370 101928
## - total_excl_count              1    563155 775587736 101932
## - num_amenities                 1    799002 775823583 101934
## - beds                          1   1290624 776315206 101940
## - bing_liu_sentiment            1   1998718 777023299 101948
## - num_bath                      1   3276699 778301281 101963
## - room_type                     3   4411312 779435893 101972
## - `Restaurants nearby`          1   4575465 779600046 101978
## - `Friendlyness of the hosts`   1   5473200 780497781 101988
## - `Spaciousness of the home`    1   7501862 782526443 102011
## - `Qaulity of rooms`            1   9484030 784508612 102034
## - `Comfort in living`           1  20027070 795051651 102153
## - `View from the listing`       1  22995903 798020484 102186
## - `Cleanliness of the listing`  1  24122538 799147119 102199
##
## Step:  AIC=101750.9
## price ~ review_scores_rating + `View from the listing` + `Qaulity of rooms` +
##     `Spaciousness of the home` + `Restaurants nearby` + `Friendlyness of the hosts
` +
##     `Cleanliness of the listing` + `Comfort in living` + total_excl_count +
##     bing_liu_sentiment + nrc_sentiment + afinn_sentiment + loughran_sentiment +
##     num_bath + num_amenities + beds + room_type
##
##                               Df Sum of Sq       RSS    AIC
## - nrc_sentiment                1      5253 794162071 101749
## - review_scores_rating         1     96285 794253103 101750
## - loughran_sentiment           1    143758 794300576 101751
## <none>                                    794156818 101751
## - afinn_sentiment              1    277642 794434460 101752
## - total_excl_count             1    549632 794706450 101755
## - num_amenities                1    934523 795091341 101759
## - beds                         1   1778867 795935685 101769
```

```
## - bing_liu_sentiment            1   2620025 796776843 101778
## - num_bath                      1   3661100 797817918 101790
## - room_type                     3   5181311 799338129 101803
## - `Restaurants nearby`          1   6120226 800277044 101817
## - `Friendlyness of the hosts`   1   6148092 800304911 101818
## - `Qaulity of rooms`            1   9919913 804076731 101860
## - `Spaciousness of the home`    1  18233445 812390263 101952
## - `Comfort in living`           1  22069512 816226330 101994
## - `Cleanliness of the listing`  1  25874228 820031046 102035
## - `View from the listing`       1  28582905 822739723 102064
##
## Step:  AIC=101749
## price ~ review_scores_rating + `View from the listing` + `Qaulity of rooms` +
##     `Spaciousness of the home` + `Restaurants nearby` + `Friendlyness of the hosts
` +
##     `Cleanliness of the listing` + `Comfort in living` + total_excl_count +
##     bing_liu_sentiment + afinn_sentiment + loughran_sentiment +
##     num_bath + num_amenities + beds + room_type
##
##                                Df Sum of Sq       RSS    AIC
## - review_scores_rating          1     96455 794258526 101748
## - loughran_sentiment            1    150258 794312329 101749
## <none>                                      794162071 101749
## - afinn_sentiment               1    282992 794445063 101750
## - total_excl_count              1    557586 794719657 101753
## - num_amenities                 1    932896 795094967 101757
## - beds                          1   1777464 795939535 101767
## - bing_liu_sentiment            1   3330671 797492742 101784
## - num_bath                      1   3661809 797823880 101788
## - room_type                     3   5191877 799353948 101801
## - `Restaurants nearby`          1   6124964 800287035 101816
## - `Friendlyness of the hosts`   1   6155579 800317650 101816
## - `Qaulity of rooms`            1   9916063 804078134 101858
## - `Spaciousness of the home`    1  18308226 812470297 101950
## - `Comfort in living`           1  22175605 816337676 101993
## - `Cleanliness of the listing`  1  25888424 820050495 102033
## - `View from the listing`       1  28617188 822779259 102063
##
## Step:  AIC=101748.1
## price ~ `View from the listing` + `Qaulity of rooms` + `Spaciousness of the home`
+
##     `Restaurants nearby` + `Friendlyness of the hosts` + `Cleanliness of the listi
ng` +
##     `Comfort in living` + total_excl_count + bing_liu_sentiment +
##     afinn_sentiment + loughran_sentiment + num_bath + num_amenities +
##     beds + room_type
##
##                                Df Sum of Sq       RSS    AIC
## - loughran_sentiment            1    147648 794406174 101748
## <none>                                      794258526 101748
## - afinn_sentiment               1    270807 794529333 101749
## - total_excl_count              1    555333 794813859 101752
## - num_amenities                 1    877038 795135564 101756
## - beds                          1   1774438 796032964 101766
## - bing_liu_sentiment            1   3451220 797709746 101785
## - num_bath                      1   3719533 797978059 101788
## - room_type                     3   5245646 799504172 101801
## - `Restaurants nearby`          1   6153360 800411886 101815
```

```
## - `Friendlyness of the hosts`   1   6619303 800877829 101820
## - `Qaulity of rooms`            1  10174862 804433388 101860
## - `Spaciousness of the home`    1  18314939 812573464 101950
## - `Comfort in living`           1  22600375 816858900 101996
## - `Cleanliness of the listing`  1  26469131 820727657 102039
## - `View from the listing`       1  29941420 824199946 102076
##
## Step:  AIC=101747.7
## price ~ `View from the listing` + `Qaulity of rooms` + `Spaciousness of the home`
+
##     `Restaurants nearby` + `Friendlyness of the hosts` + `Cleanliness of the listi
ng` +
##     `Comfort in living` + total_excl_count + bing_liu_sentiment +
##     afinn_sentiment + num_bath + num_amenities + beds + room_type
##
##                                Df Sum of Sq       RSS    AIC
## <none>                                      794406174 101748
## - afinn_sentiment               1    350530 794756703 101750
## - total_excl_count              1    657026 795063199 101753
## - num_amenities                 1    899558 795305731 101756
## - beds                          1   1778893 796185066 101766
## - bing_liu_sentiment            1   3533231 797939405 101785
## - num_bath                      1   3713222 798119396 101787
## - room_type                     3   5211259 799617433 101800
## - `Restaurants nearby`          1   6133754 800539927 101814
## - `Friendlyness of the hosts`   1   6668788 801074962 101820
## - `Qaulity of rooms`            1  10217707 804623881 101860
## - `Spaciousness of the home`    1  18210657 812616831 101948
## - `Comfort in living`           1  22610722 817016896 101996
## - `Cleanliness of the listing`  1  26378627 820784801 102037
## - `View from the listing`       1  29794652 824200826 102074
```

```
##
## Call:
## lm(formula = price ~ `View from the listing` + `Qaulity of rooms` +
##     `Spaciousness of the home` + `Restaurants nearby` + `Friendlyness of the hosts
` +
##     `Cleanliness of the listing` + `Comfort in living` + total_excl_count +
##     bing_liu_sentiment + afinn_sentiment + num_bath + num_amenities +
##     beds + room_type, data = regress_stm_price_review)
##
## Coefficients:
##                   (Intercept)       `View from the listing`
##                     -652.1011                     921.9076
##           `Qaulity of rooms`    `Spaciousness of the home`
##                     562.8972                     727.0966
##         `Restaurants nearby`   `Friendlyness of the hosts`
##                     429.4678                     443.4288
## `Cleanliness of the listing`           `Comfort in living`
##                     958.7998                     932.6248
##             total_excl_count            bing_liu_sentiment
##                       0.3936                     155.5903
##              afinn_sentiment                     num_bath
##                      -0.3111                      61.9683
##                num_amenities                         beds
##                      -0.6348                      15.8408
##          room_typeHotel room        room_typePrivate room
##                      -9.0351                      75.3557
##         room_typeShared room
##                     145.1895
```

```
summary(lm(price ~ `View from the listing` + `Qaulity of rooms` +
    `Spaciousness of the home` + `Restaurants nearby` + `Friendlyness of the hosts` +
    `Cleanliness of the listing` + `Comfort in living` + total_excl_count +

    bing_liu_sentiment + afinn_sentiment + loughran_sentiment +
    num_bath + num_amenities + beds + room_type, data = regress_stm_price_review))
```

```
##
## Call:
## lm(formula = price ~ `View from the listing` + `Qaulity of rooms` +
##     `Spaciousness of the home` + `Restaurants nearby` + `Friendlyness of the hosts
` +
##     `Cleanliness of the listing` + `Comfort in living` + total_excl_count +
##     bing_liu_sentiment + afinn_sentiment + loughran_sentiment +
##     num_bath + num_amenities + beds + room_type, data = regress_stm_price_review)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -559.2  -63.5  -11.8   37.5 9536.7
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 -657.9590    49.2390 -13.363  < 2e-16 ***
## `View from the listing`      926.1227    50.5414  18.324  < 2e-16 ***
## `Qaulity of rooms`           561.7908    52.5927  10.682  < 2e-16 ***
## `Spaciousness of the home`   729.7941    50.9229  14.331  < 2e-16 ***
## `Restaurants nearby`         430.1781    51.7854   8.307  < 2e-16 ***
## `Friendlyness of the hosts`  441.8992    51.2900   8.616  < 2e-16 ***
## `Cleanliness of the listing` 960.8264    55.7687  17.229  < 2e-16 ***
## `Comfort in living`          932.4149    58.5688  15.920  < 2e-16 ***
## total_excl_count               0.3658     0.1466   2.496  0.01260 *
## bing_liu_sentiment           171.1157    27.5054   6.221 5.16e-10 ***
## afinn_sentiment               -0.2772     0.1591  -1.743  0.08143 .
## loughran_sentiment           -19.8226    15.4050  -1.287  0.19821
## num_bath                      62.0215     9.6031   6.458 1.11e-10 ***
## num_amenities                 -0.6270     0.1999  -3.136  0.00172 **
## beds                          15.8211     3.5467   4.461 8.26e-06 ***
## room_typeHotel room           -9.3450    46.1000  -0.203  0.83937
## room_typePrivate room         75.6238    10.9378   6.914 5.04e-12 ***
## room_typeShared room         145.6369    29.4231   4.950 7.57e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 298.6 on 8907 degrees of freedom
## Multiple R-squared:  0.1133, Adjusted R-squared:  0.1116
## F-statistic: 66.93 on 17 and 8907 DF,  p-value: < 2.2e-16
```

# Review level aggregation

```
annotated_reviews_all <- readr::read_csv("annotated_reviews.csv")

annotated_reviews_all$review_id <- as.integer(annotated_reviews_all$review_id)

data_for_stm <- annotated_reviews_all %>%
  left_join(data)
```

# STM execution

```
# Prepare metadata for the STM model


data_for_stm <- data_for_stm %>%
  group_by(review_id) %>%
  summarise(comments_grouped = paste(annotated_comments, collapse = " "))

data_for_joining <- listing_reviews %>%
  dplyr::select(price,review_scores_rating,listing_id, review_id)

data_for_joining$price <- gsub("\\$|,","",data_for_joining$price)
data_for_joining$price <- as.numeric(data_for_joining$price)


data_for_stm <- data_for_stm %>%
  left_join(data_for_joining)

data_for_stm <- data_for_stm %>% na.omit()

processed <- textProcessor(data_for_stm$comments_grouped,
                           metadata = data_for_stm,
                           customstopwords = c("neighborhood",
                "manhattan",
                "located",
                "brooklyn",
                "queens",
                "staten",
                "island",
                "bronx",
                "stay",
                "staying",
                "airbnb",
                "line",
                "much",
                "min",
                "also","location", "apartment", "suite", "place", "host", "great"),
                           stem = F)
```

```
## Building corpus...
## Converting to Lower Case...
## Removing punctuation...
## Removing stopwords...
## Remove Custom Stopwords...
## Removing numbers...
## Creating Output...
```

```
# Keep only those words that appear
# on the 1% of the corpus
#
threshold <- round(1/100 * length(processed$documents),0)

out <- prepDocuments(processed$documents,
                     processed$vocab,
                     processed$meta,
                     lower.thresh = threshold)
```

```
## Removing 47348 of 47645 terms (2314185 of 8233512 tokens) due to frequency
## Removing 426 Documents with No Words
## Your corpus now has 500852 documents, 297 terms and 5919327 tokens.
```

```
# Building corpus...
# Converting to Lower Case...
# Removing punctuation...
# Removing stopwords...
# Remove Custom Stopwords...
# Removing numbers...
# Creating Output...
# Removing 40466 of 40770 terms (1688986 of 6045427 tokens) due to frequency
# Removing 278 Documents with No Words
# Your corpus now has 352869 documents, 304 terms and 4356441 tokens.
```

```
# numtopics <- searchK(out$documents,out$vocab,K=seq(from=5, to=8,by=1))
# plot(numtopics)
```

```
# numtopics <- searchK(out$documents,out$vocab,K=seq(from=7, to=10,by=1))
# plot(numtopics)
```

```
newyorkfit <- stm(documents = out$documents,
                  vocab = out$vocab,
                  K = 8,
                  prevalence = ~review_scores_rating + price,
                  max.em.its = 75,
                  data = out$meta,
                  reportevery=3,
                  # gamma.prior = "L1",
                  sigma.prior = 0.7,
                  init.type = "Spectral")
```

```
## Beginning Spectral Initialization
##   Calculating the gram matrix...
##   Finding anchor words...
##      ........
##   Recovering initialization...
##      ..
## Initialization complete.
##
.....................................................................................

## Completed E-Step (107 seconds).
## Completed M-Step.
## Completing Iteration 1 (approx. per word bound = -5.307)
##
.....................................................................................

## Completed E-Step (93 seconds).
## Completed M-Step.
## Completing Iteration 2 (approx. per word bound = -5.282, relative change = 4.678e-
03)
##
.....................................................................................

## Completed E-Step (80 seconds).
## Completed M-Step.
## Completing Iteration 3 (approx. per word bound = -5.267, relative change = 2.823e-
03)
## Topic 1: home, time, wonderful, amazing, experience
##  Topic 2: good, nice, price, really, value
##  Topic 3: restaurant, subway, shop, distance, close
##  Topic 4: clean, easy, super, definitely, comfortable
##  Topic 5: really, nice, close, subway, clean
##  Topic 6: minute, walk, subway, away, just
##  Topic 7: room, bed, bathroom, clean, nice
##  Topic 8: public, subway, clean, comfortable, nice
##
.....................................................................................

## Completed E-Step (82 seconds).
## Completed M-Step.
## Completing Iteration 4 (approx. per word bound = -5.254, relative change = 2.464e-
03)
##
.....................................................................................

## Completed E-Step (75 seconds).
## Completed M-Step.
## Completing Iteration 5 (approx. per word bound = -5.243, relative change = 2.082e-
03)
##
.....................................................................................

## Completed E-Step (69 seconds).
## Completed M-Step.
## Completing Iteration 6 (approx. per word bound = -5.234, relative change = 1.650e-
03)
## Topic 1: home, time, wonderful, amazing, experience
```

```
##  Topic 2: good, bit, overall, price, value
##  Topic 3: restaurant, shop, subway, distance, bar
##  Topic 4: clean, super, easy, definitely, comfortable
##  Topic 5: really, nice, close, subway, clean
##  Topic 6: minute, walk, away, just, subway
##  Topic 7: room, bed, bathroom, kitchen, clean
##  Topic 8: clean, comfortable, subway, public, quiet
##
.....................................................................................

## Completed E-Step (70 seconds).
## Completed M-Step.
## Completing Iteration 7 (approx. per word bound = -5.228, relative change = 1.238e-
03)
##
.....................................................................................

## Completed E-Step (74 seconds).
## Completed M-Step.
## Completing Iteration 8 (approx. per word bound = -5.223, relative change = 8.910e-
04)
##
.....................................................................................

## Completed E-Step (78 seconds).
## Completed M-Step.
## Completing Iteration 9 (approx. per word bound = -5.220, relative change = 6.227e-
04)
## Topic 1: time, home, amazing, wonderful, back
##  Topic 2: good, overall, bit, price, little
##  Topic 3: restaurant, shop, subway, distance, bar
##  Topic 4: clean, easy, super, definitely, comfortable
##  Topic 5: nice, really, close, subway, clean
##  Topic 6: minute, walk, just, away, subway
##  Topic 7: room, bed, bathroom, kitchen, bedroom
##  Topic 8: comfortable, quiet, clean, well, subway
##
.....................................................................................

## Completed E-Step (82 seconds).
## Completed M-Step.
## Completing Iteration 10 (approx. per word bound = -5.218, relative change = 4.253e
-04)
##
.....................................................................................

## Completed E-Step (85 seconds).
## Completed M-Step.
## Completing Iteration 11 (approx. per word bound = -5.216, relative change = 2.845e
-04)
##
.....................................................................................

## Completed E-Step (90 seconds).
## Completed M-Step.
## Completing Iteration 12 (approx. per word bound = -5.215, relative change = 1.843e
-04)
## Topic 1: time, home, amazing, wonderful, back
```

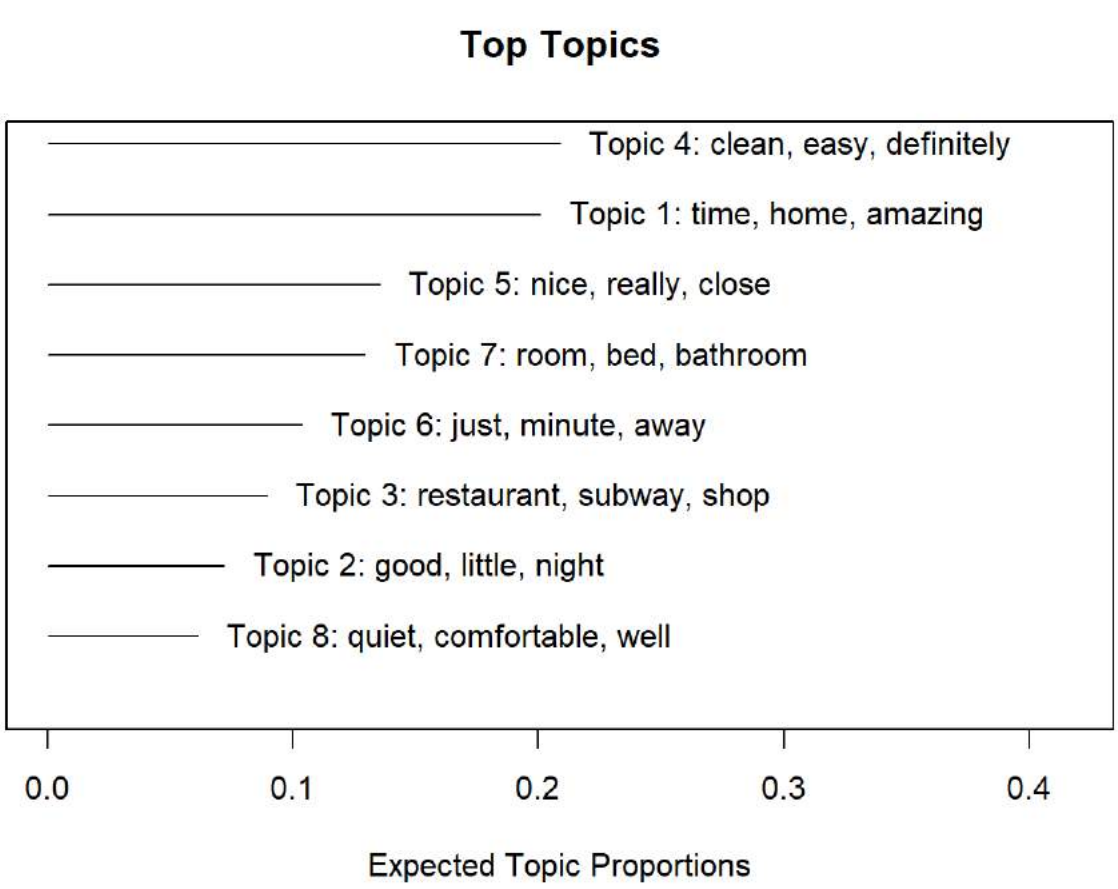
```
##  Topic 2: good, little, overall, bit, night
##  Topic 3: restaurant, shop, subway, many, distance
##  Topic 4: clean, easy, super, definitely, comfortable
##  Topic 5: nice, really, close, subway, friendly
##  Topic 6: minute, just, away, walk, day
##  Topic 7: room, bed, bathroom, kitchen, small
##  Topic 8: quiet, comfortable, well, clean, subway
##
.....................................................................................

## Completed E-Step (94 seconds).
## Completed M-Step.
## Completing Iteration 13 (approx. per word bound = -5.215, relative change = 1.141e
-04)
##
.....................................................................................

## Completed E-Step (96 seconds).
## Completed M-Step.
## Completing Iteration 14 (approx. per word bound = -5.214, relative change = 6.496e
-05)
##
.....................................................................................

## Completed E-Step (95 seconds).
## Completed M-Step.
## Completing Iteration 15 (approx. per word bound = -5.214, relative change = 3.060e
-05)
## Topic 1: time, home, amazing, back, wonderful
##  Topic 2: good, little, night, overall, bit
##  Topic 3: restaurant, subway, shop, many, distance
##  Topic 4: clean, easy, definitely, super, comfortable
##  Topic 5: nice, really, close, friendly, subway
##  Topic 6: just, minute, away, walk, day
##  Topic 7: room, bed, bathroom, kitchen, small
##  Topic 8: quiet, comfortable, well, clean, safe
##
.....................................................................................

## Completed E-Step (97 seconds).
## Completed M-Step.
## Model Converged
```

```
summary(newyorkfit)
```

```
## A topic model with 8 topics, 500852 documents and a 297 word dictionary.
```

```
## Topic 1 Top Words:
##       Highest Prob: time, home, amazing, back, wonderful, experience, beautiful
##       FREX: home, amazing, best, wonderful, first, beautiful, next
##       Lift: second, truly, soon, ever, hospitality, home, incredible
##       Score: second, home, amazing, time, wonderful, experience, beautiful
## Topic 2 Top Words:
##       Highest Prob: good, little, night, overall, bit, people, price
##       FREX: good, price, bit, overall, value, little, pretty
##       Lift: money, good, price, value, worth, noisy, bit
##       Score: money, good, bit, price, value, overall, little
## Topic 3 Top Words:
##       Highest Prob: restaurant, subway, shop, many, distance, bar, coffee
##       FREX: distance, shop, bar, restaurant, walking, store, many
##       Lift: walking, grocery, distance, bar, store, shop, cafe
##       Score: walking, restaurant, shop, distance, bar, store, grocery
## Topic 4 Top Words:
##       Highest Prob: clean, easy, definitely, super, comfortable, highly, space
##       FREX: question, super, quick, responsive, communication, accommodating, exac
tly
##       Lift: response, process, communicative, stylish, question, responsive, quick
##       Score: response, super, clean, easy, quick, definitely, question
## Topic 5 Top Words:
##       Highest Prob: nice, really, close, friendly, subway, helpful, area
##       FREX: really, nice, neighbourhood, flat, close, cool, friendly
##       Lift: transport, cosy, flat, neighbourhood, really, transportation, cool
##       Score: transport, nice, really, close, friendly, transportation, flat
## Topic 6 Top Words:
##       Highest Prob: just, minute, away, day, walk, subway, station
##       FREX: minute, walk, late, away, short, flight, just
##       Lift: last, minute, less, ride, flight, walk, late
##       Score: last, minute, walk, away, station, just, day
## Topic 7 Top Words:
##       Highest Prob: room, bed, bathroom, kitchen, small, bedroom, big
##       FREX: bathroom, bedroom, kitchen, private, shower, bed, room
##       Lift: living, window, bathroom, bedroom, shower, hot, water
##       Score: living, room, bathroom, bed, kitchen, bedroom, small
## Topic 8 Top Words:
##       Highest Prob: quiet, comfortable, well, clean, safe, public, area
##       FREX: quiet, public, safe, well, comfortable, parking, convenient
##       Lift: public, parking, quiet, free, safe, convenient, pleasant
##       Score: public, quiet, comfortable, well, parking, safe, convenient
```

```
plot(newyorkfit)
```




```
newyorkfit_topics<-tidy(newyorkfit,matrix="beta")
```

```
#And we get the top 10 terms:

newyorkfit_terms<-newyorkfit_topics %>%
  group_by(topic) %>%
  slice_max(beta,n=10) %>%
  ungroup() %>%
  arrange(topic,desc(beta))
# newyorkfit_terms
```

***Topic labeling***

```
# we look at the most frequent words appearing in each topic and give a name to each
 topic.
topic_labels_1 <- paste0("topic_",1:8)

topic_labels <- c("Overall Experience of Host and Accommodation",
                  "Quality of Room and Surrounding Conditions",
                  "Restaurants, Cafes and Supermarkets nearby",
                  "Host Helpfulness and Response time",
                  "Connectivity to Public Transport and Parking",
                  "Convinience and Prime Location",
                  "Neighborhood Experience",
                  "Quality of Ameneties and Room Condition")
```

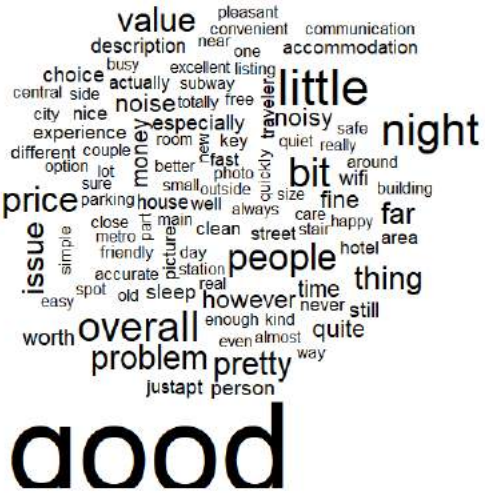
And we can plot the results:

```
newyorkfit_terms <- newyorkfit_terms %>%
mutate(topic_label = case_when((topic == 1) ~ topic_labels[1],
(topic == 2) ~ topic_labels[2],
(topic == 3) ~ topic_labels[3],
(topic == 4) ~ topic_labels[4],
(topic == 5) ~ topic_labels[5],
(topic == 6) ~ topic_labels[6],
(topic == 7) ~ topic_labels[7],
(topic == 8) ~ topic_labels[8]))

newyorkfit_terms %>%
mutate(term = reorder_within(term, beta, topic)) %>%
ggplot(aes(beta, term, fill = factor(topic))) +
geom_col(show.legend = FALSE) +
facet_wrap(~ topic_label, scales = "free") +
scale_y_reordered()
```

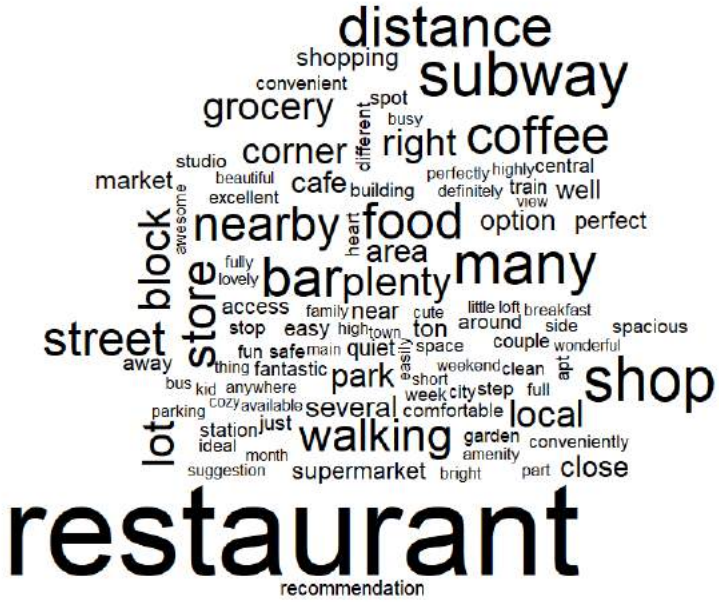
```
mod = newyorkfit
docs = out$documents
toLDAvis(
  mod,
  docs,
  R = 30,
  plot.opts = list(xlab = "PC1", ylab = "PC2"),
  lambda.step = 0.1,
  out.dir = tempfile(),
  open.browser = interactive(),
  as.gist = FALSE,
  reorder.topics = TRUE
)
```

```
#install.packages("wordcloud")
library(wordcloud)
stm::cloud(newyorkfit,topic = 1)
```


```
stm::cloud(newyorkfit,topic = 2)
```



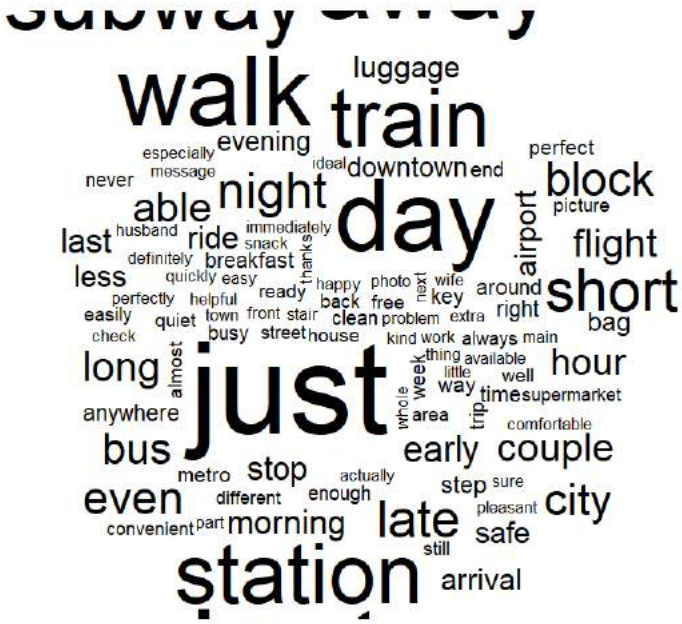

```
stm::cloud(newyorkfit,topic = 3)
```



```
stm::cloud(newyorkfit,topic = 4)
```

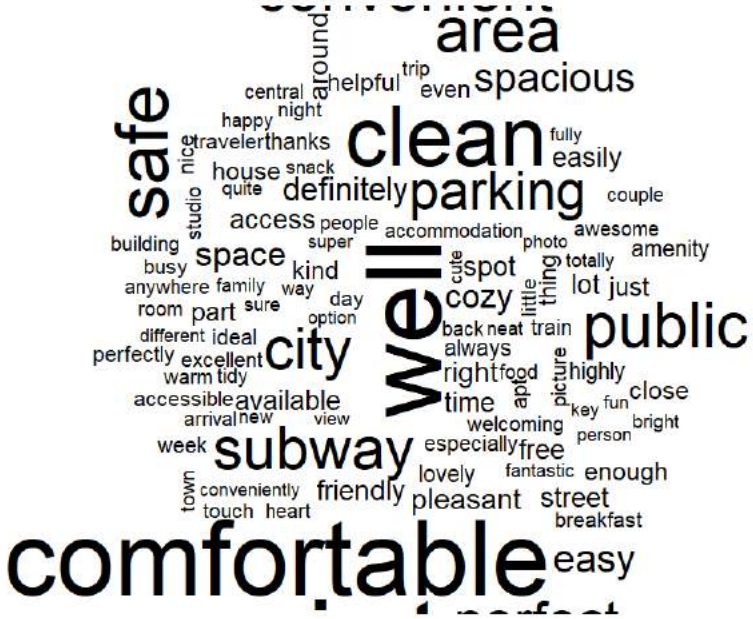

```
stm::cloud(newyorkfit,topic = 5)
```


```
stm::cloud(newyorkfit,topic = 6)
```



```
stm::cloud(newyorkfit,topic = 7)
```

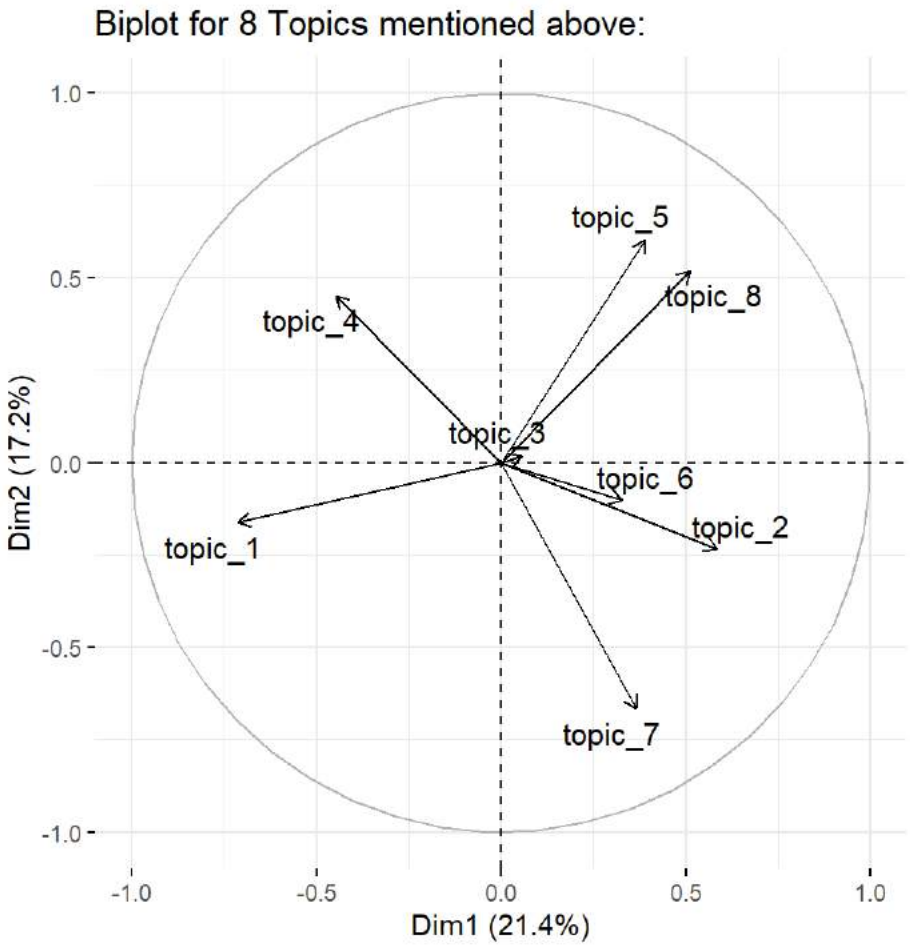

```
stm::cloud(newyorkfit,topic = 8)
```


area
around
helpful
trip
even
spacious
central
night
safe
clean
happy
traveler
thanks
fully
easily
nice
house
snack
quite
definitely
parking
couple
studio
access
people
accommodation
awesome
building
super
photo
amenity
space
kind
well
cute
spot
totally
busy
anywhere
family
way
thing
lot
just
day
cozy
room
part
sure
option
little
public
different
ideal
back
neat
train
perfectly
excellent
city
always
warm
tidy
right
food
highly
picture
accessible
available
time
apt
close
arrival
new
view
key
fun
welcoming
bright
person
week
subway
especially
free
lovely
fantastic
enough
town
conveniently
friendly
pleasant
street
touch
heart
breakfast
comfortable
easy

```
# Creating the Gamma Matrix (Theta Matrix)

gamma_topics<-tidy(newyorkfit,matrix="gamma")


# Create Tidy Format with Pivot Wider


gamma_topics<-gamma_topics %>%
  pivot_wider(names_from=topic,values_from=gamma)


# Create another Gamma Matix (Theta) for visualising the Bi-plot clearly.
gamma_2 <- gamma_topics
colnames(gamma_2)<-c("document",topic_labels_1)

# Add pre-defined column names:


colnames(gamma_topics)<-c("document",topic_labels)

rownames(gamma_topics) <- gamma_topics$document
rownames(gamma_2) <- gamma_2$document


gamma_topics$document <- NULL
gamma_2$document <- NULL


gamma_topics<-as.data.frame(gamma_topics)
gamma_2<-as.data.frame(gamma_2)


head(gamma_topics)
```

```
##   Overall Experience of Host and Accommodation
## 1                                    0.25116833
## 2                                    0.07275295
## 3                                    0.36701179
## 4                                    0.13008629
## 5                                    0.30436558
## 6                                    0.29622997
##   Quality of Room and Surrounding Conditions
## 1                                 0.06454657
## 2                                 0.20059824
## 3                                 0.05219799
## 4                                 0.30347424
## 5                                 0.07560815
## 6                                 0.05436337
##   Restaurants, Cafes and Supermarkets nearby Host Helpfulness and Response time
## 1                                 0.07197829                           0.1701640
## 2                                 0.06134577                           0.1609985
## 3                                 0.05388193                           0.1272751
## 4                                 0.04444396                           0.1384599
## 5                                 0.05530319                           0.2222074
## 6                                 0.04950692                           0.3047618
##   Connectivity to Public Transport and Parking Convinience and Prime Location
## 1                                    0.14922767                     0.09891960
## 2                                    0.17268787                     0.06974892
## 3                                    0.18617042                     0.06923517
## 4                                    0.14324715                     0.10277446
## 5                                    0.10958922                     0.06979748
## 6                                    0.08347011                     0.06386370
##   Neighborhood Experience Quality of Ameneties and Room Condition
## 1              0.12095308                              0.07304247
## 2              0.20138800                              0.06047980
## 3              0.08405062                              0.06017701
## 4              0.07722850                              0.06028547
## 5              0.09766703                              0.06546194
## 6              0.08374299                              0.06406111
```

```r
# Biplot

#install.packages("factoextra")
library(factoextra)
pcah2 <- FactoMineR::PCA(gamma_2,graph = FALSE)
factoextra::fviz_pca_var(pcah2, repel = TRUE, title = "Biplot for 8 Topics mentioned
 above:")
```




***Expected topic proportions***

We extract the theta matrix from the fitted object

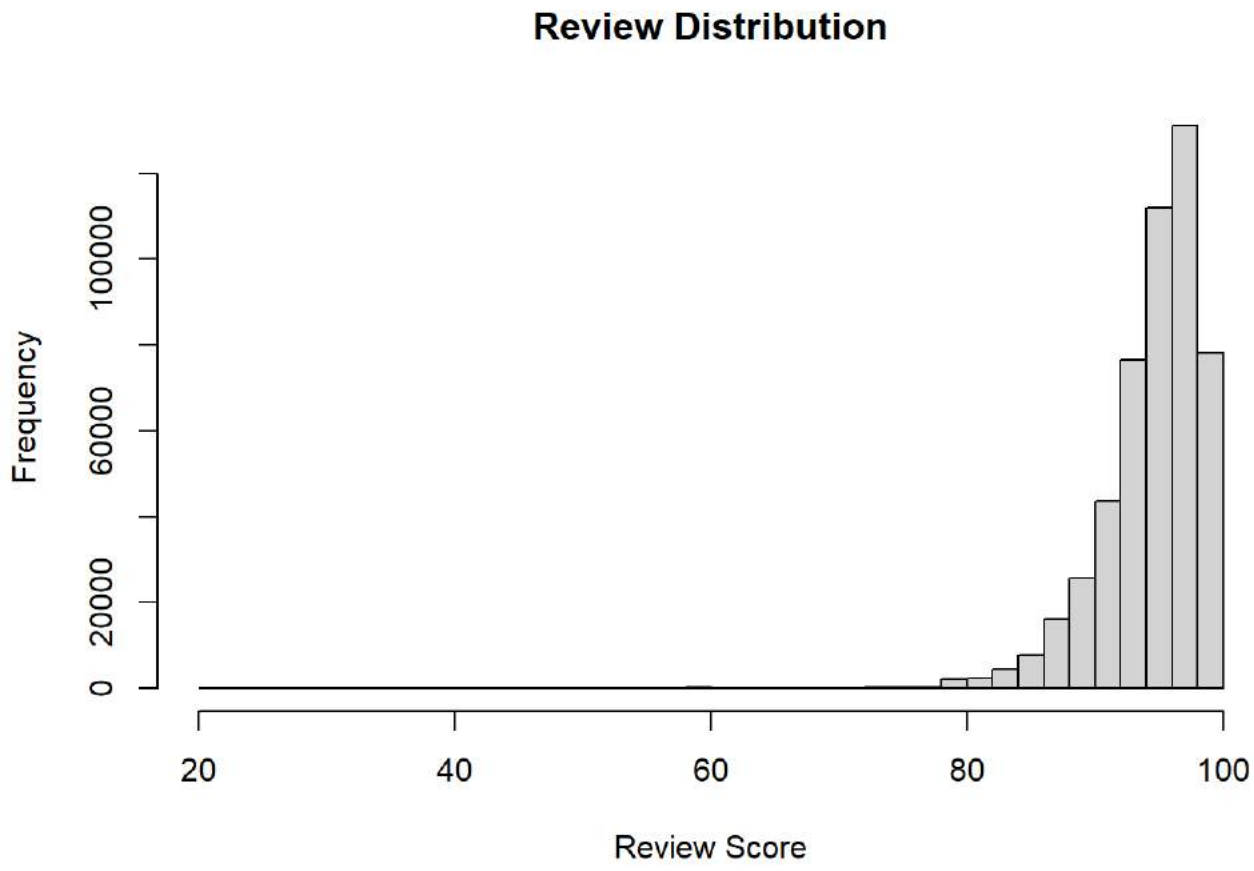
```
convergence_theta <- as.data.frame(newyorkfit$theta)

topic_summary <- summary(newyorkfit)
```

```
## A topic model with 8 topics, 500852 documents and a 297 word dictionary.
```

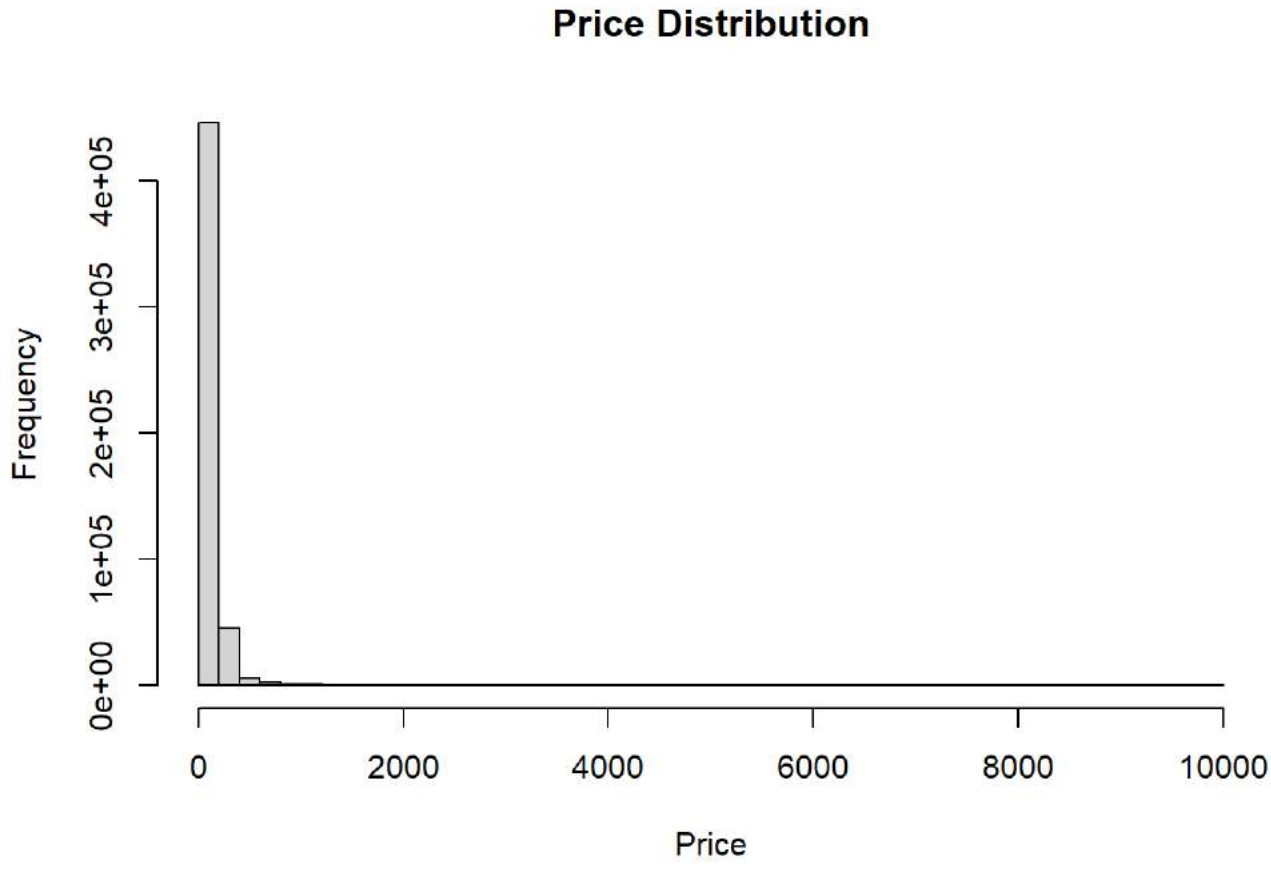
```
topic_proportions <- colMeans(newyorkfit$theta)
```

***Why no negative topics?***

```
#we can see that the reviews for the listings range from 80 to a 100, so we only have positive reviews.

hist(data_for_stm$review_scores_rating,xlab="Review Score", main="Review Distributio
n",breaks = 50)
```




```
hist(data_for_stm$price,xlab="Price", main="Price Distribution",breaks = 50)
```

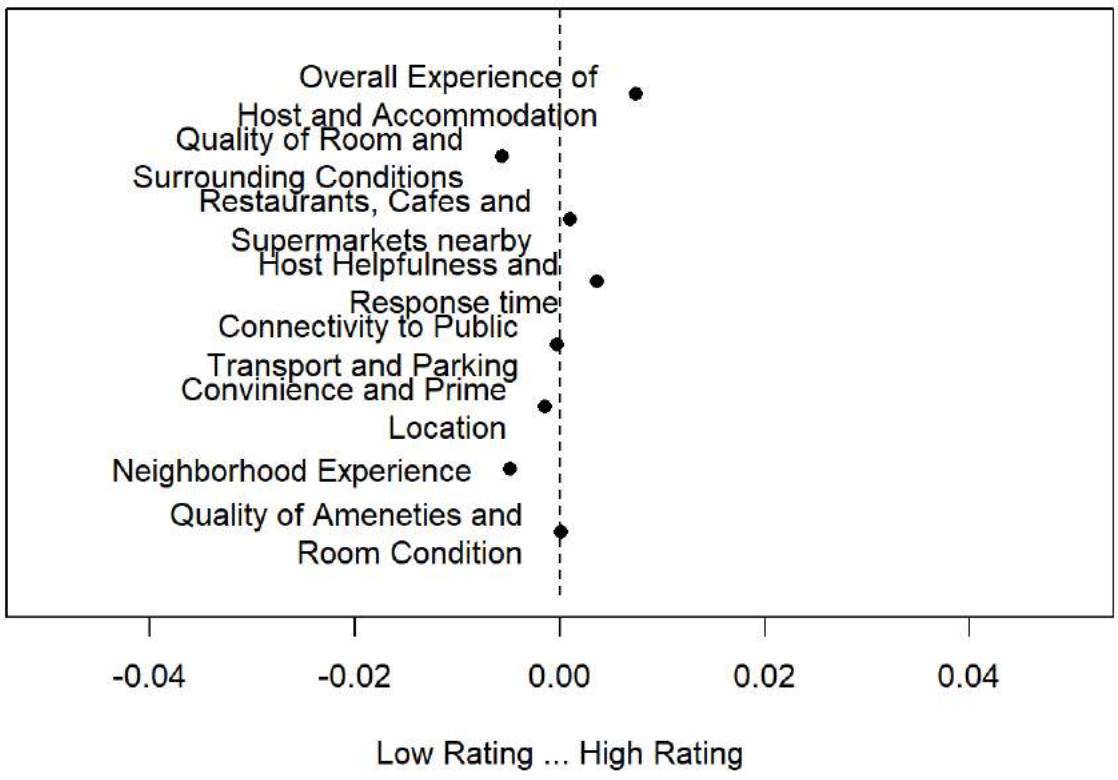


```
# Notice that, both Review ratings and Price are highly skewed. This will affect the
 analysis.
```

# STM Effect Estimation

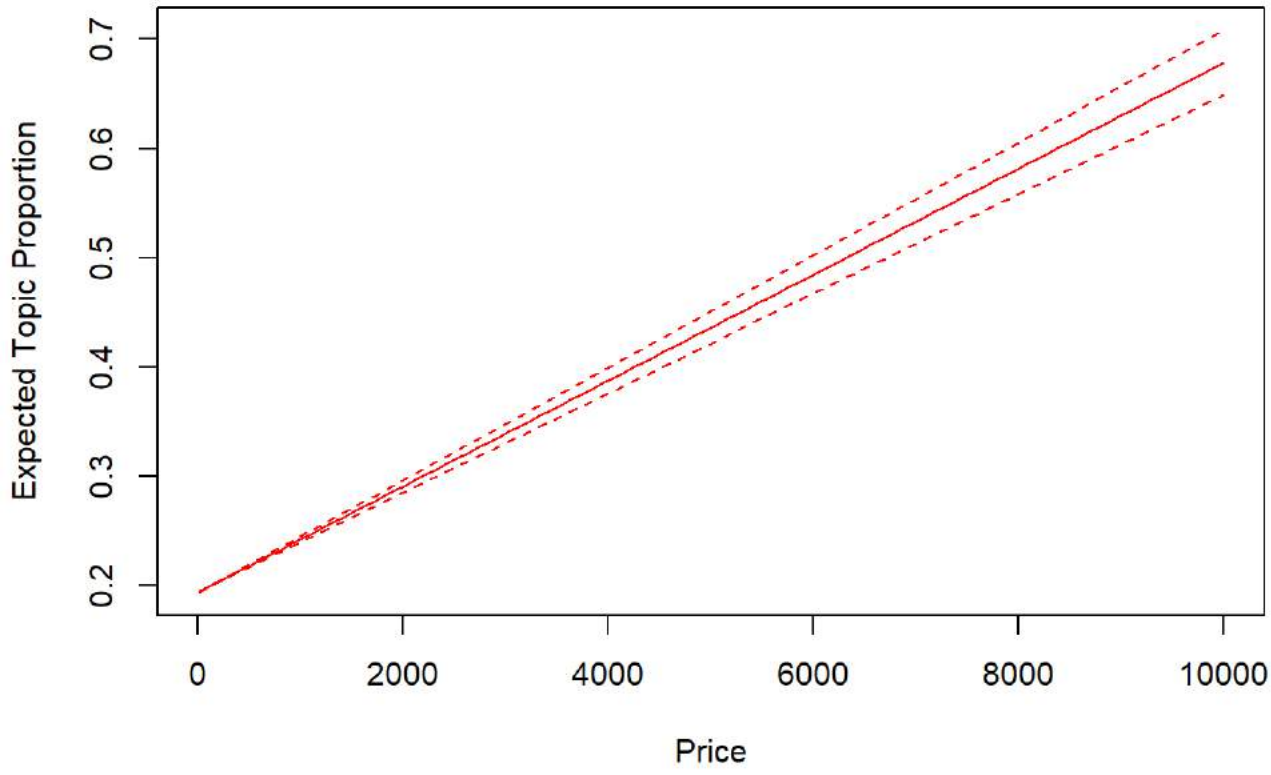
```
# Estimate the effects of price and review score
# on topic probability

effects <- estimateEffect(~review_scores_rating+price,
                          stmobj = newyorkfit,
                          metadata = out$meta)
```

```
# Effect of review score on topic
# probability
plot(effects, covariate = "review_scores_rating",
     topics = c(1:8),
     model = newyorkfit, method = "difference",
     cov.value1 = "100", cov.value2 = "0",
     xlab = "Low Rating ... High Rating",
     xlim = c(-0.05,0.05),
     main = "",
     ci.level = 0.05,
     custom.labels =topic_labels[c(1:8)],
     labeltype = "custom")
```

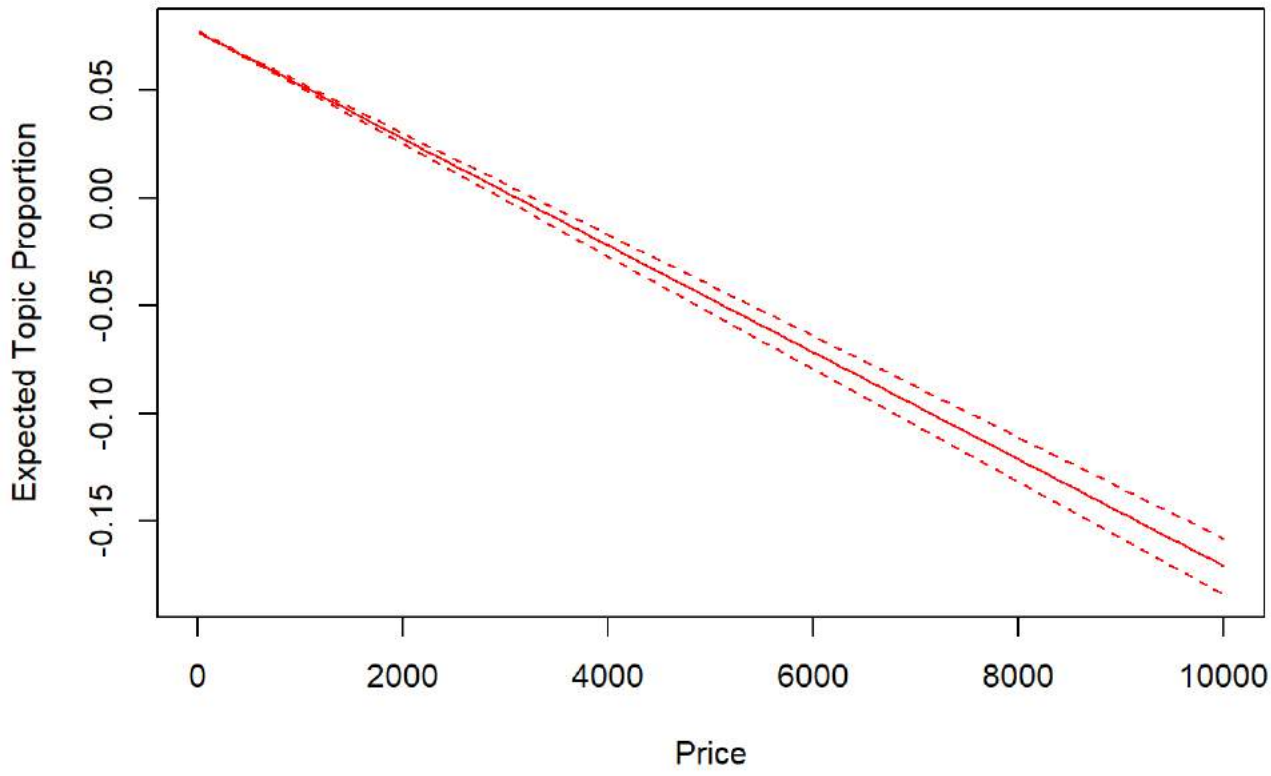


```
# Effect of price on topic
# probability treating price as
# a continuous variable.
#
# Ploting each topic separately
# we can see that some of them increase
# substantially with the price

for(i in 1:length(topic_labels)){
plot(effects, covariate = "price",
     topics = i,
     model = airbnbfit, method = "continuous",
     # For this plotting we get the uper quantile
     # and low quantile of the price
     xlab = "Price",
     # xlim = c(0,800),
     main = topic_labels[i],
     printlegend = FALSE,
     custom.labels =topic_labels[i],
     labeltype = "custom")
}
```

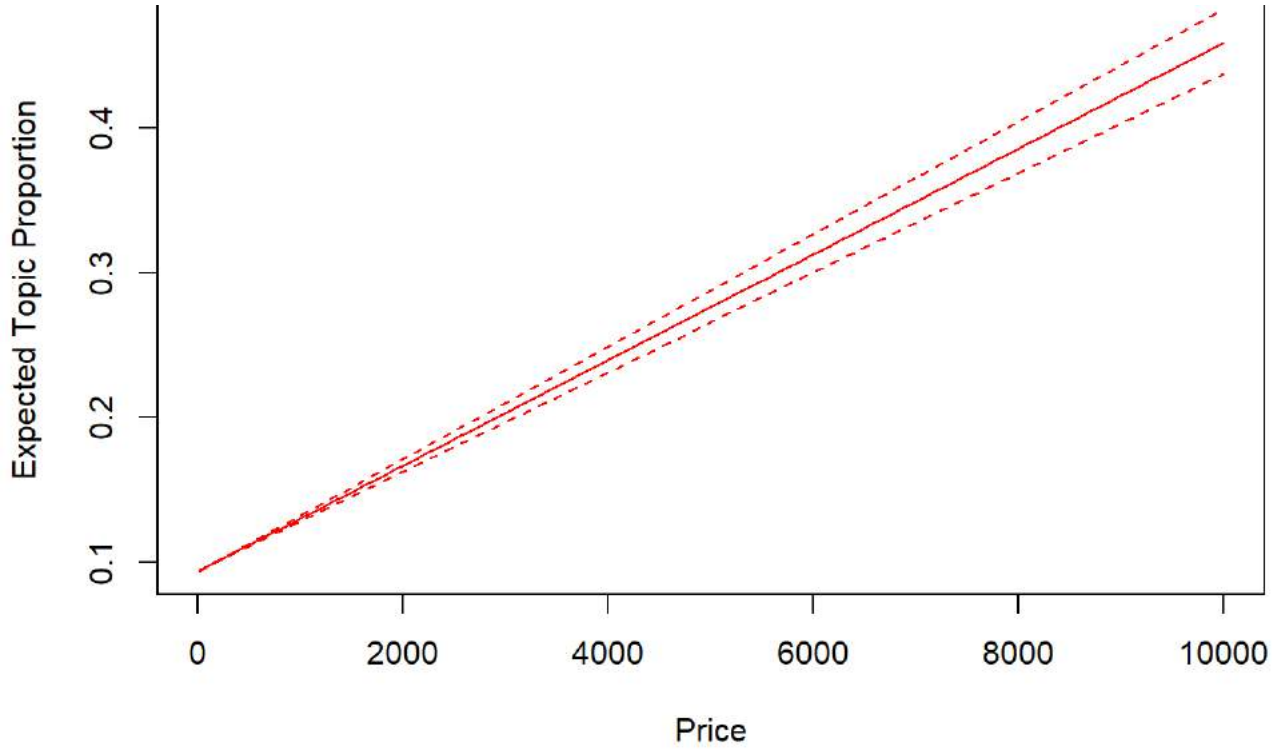
## Overall Experience of Host and Accommodation



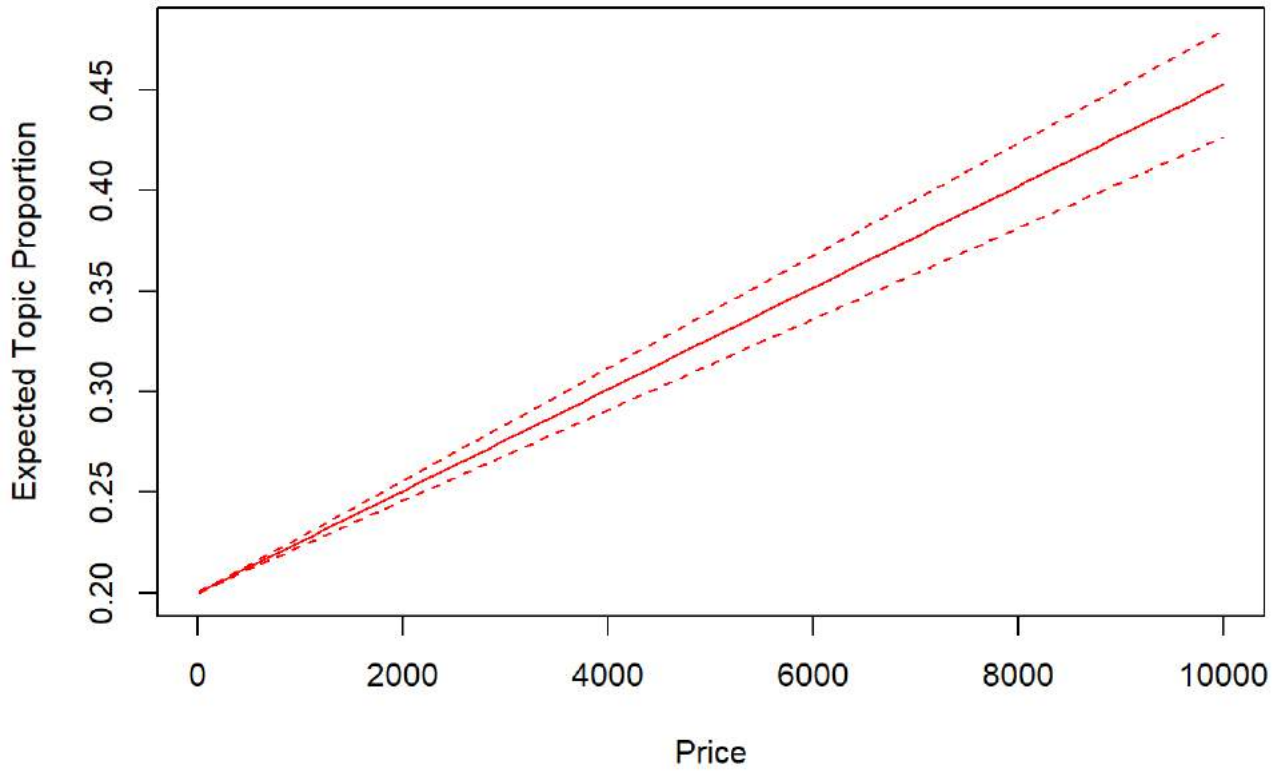

## Quality of Room and Surrounding Conditions



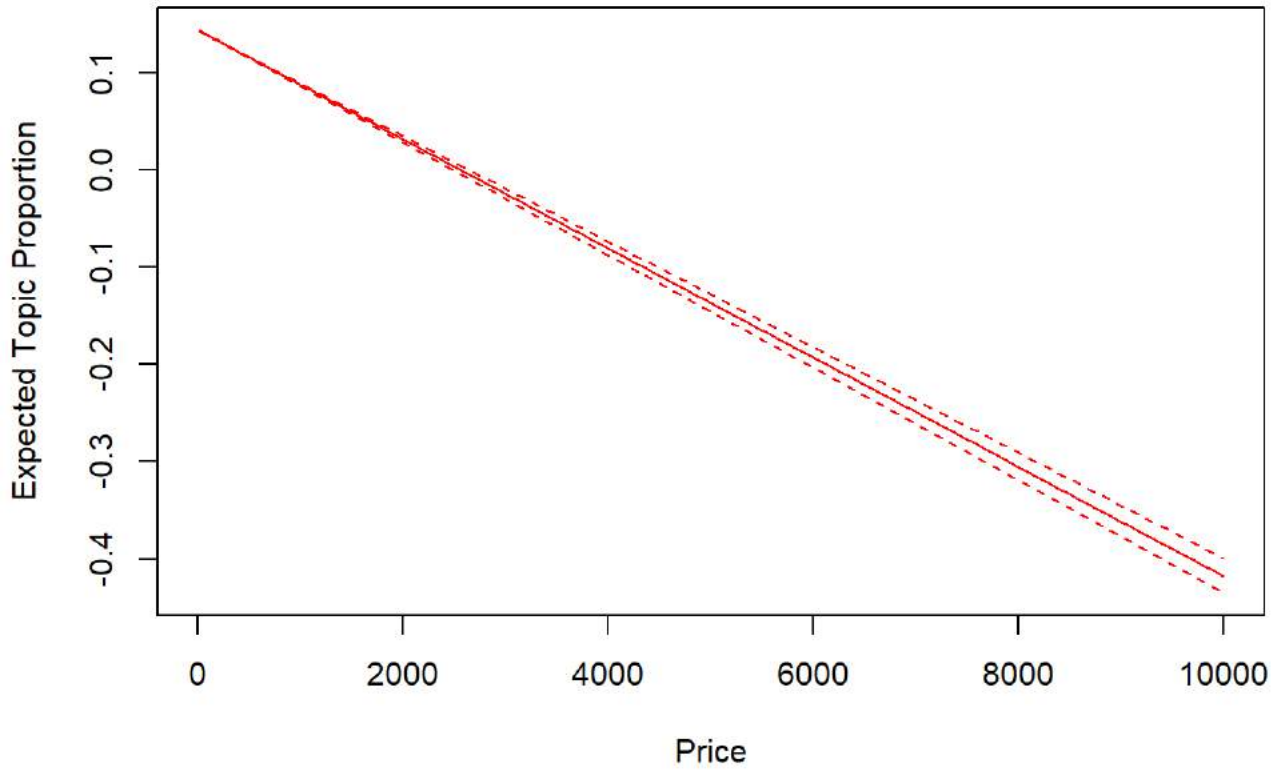

## Restaurants, Cafes and Supermarkets nearby



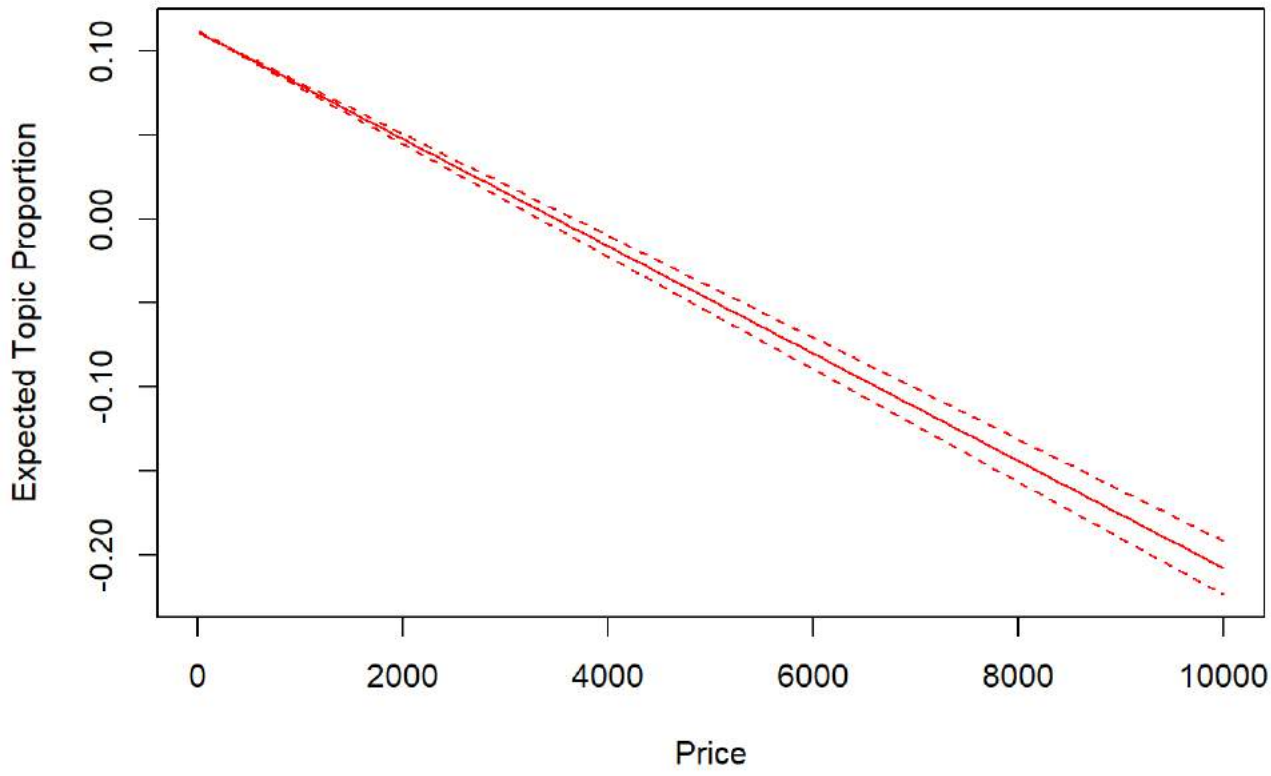

**Host Helpfulness and Response time**

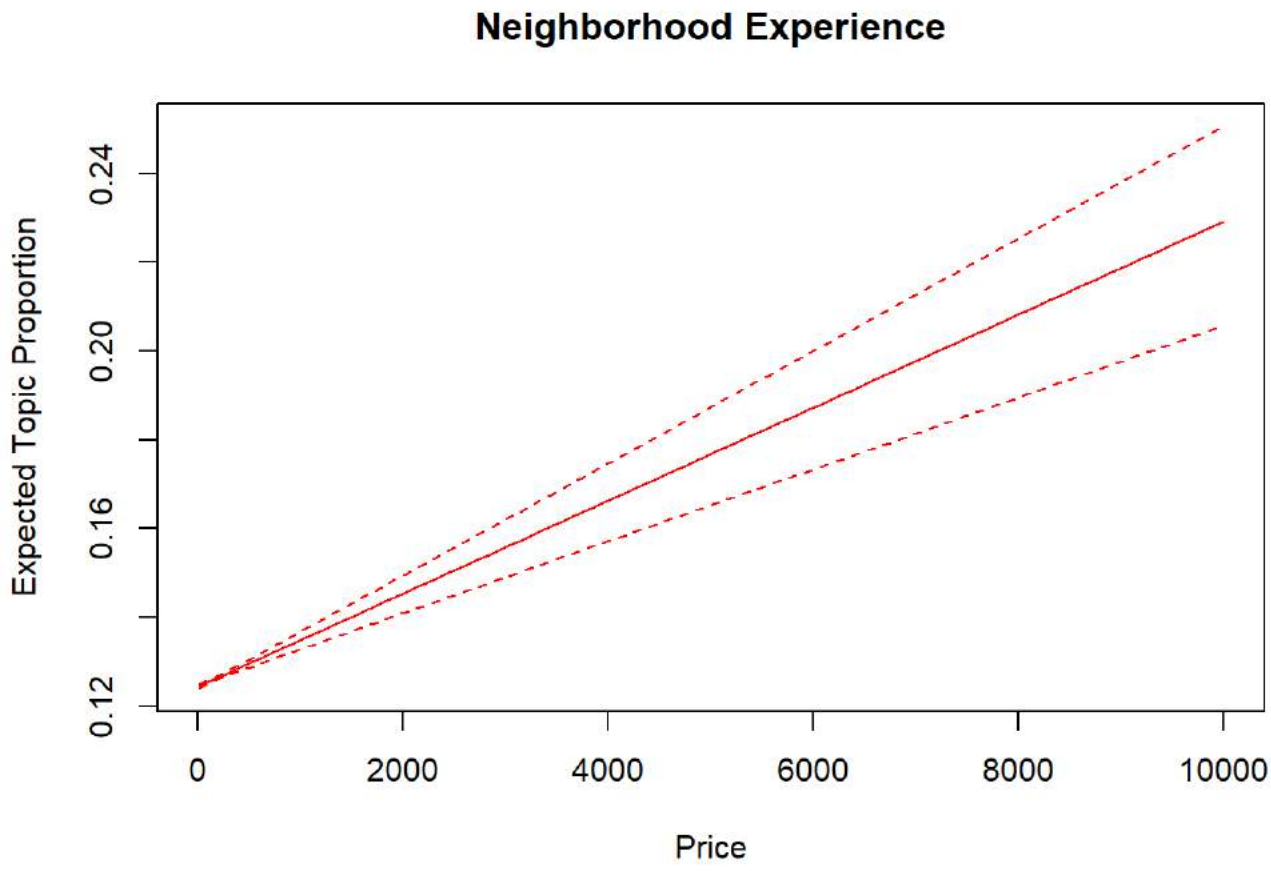


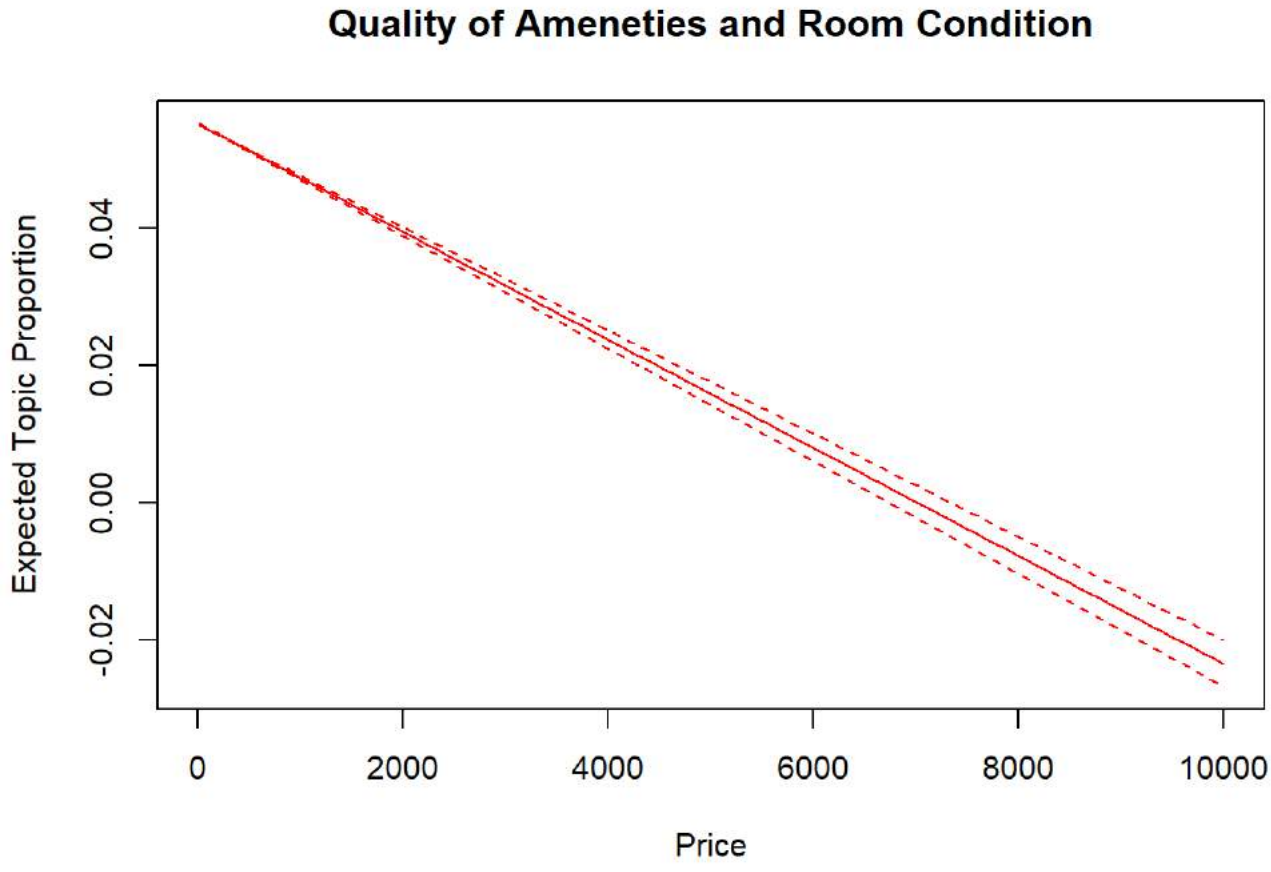
## Connectivity to Public Transport and Parking



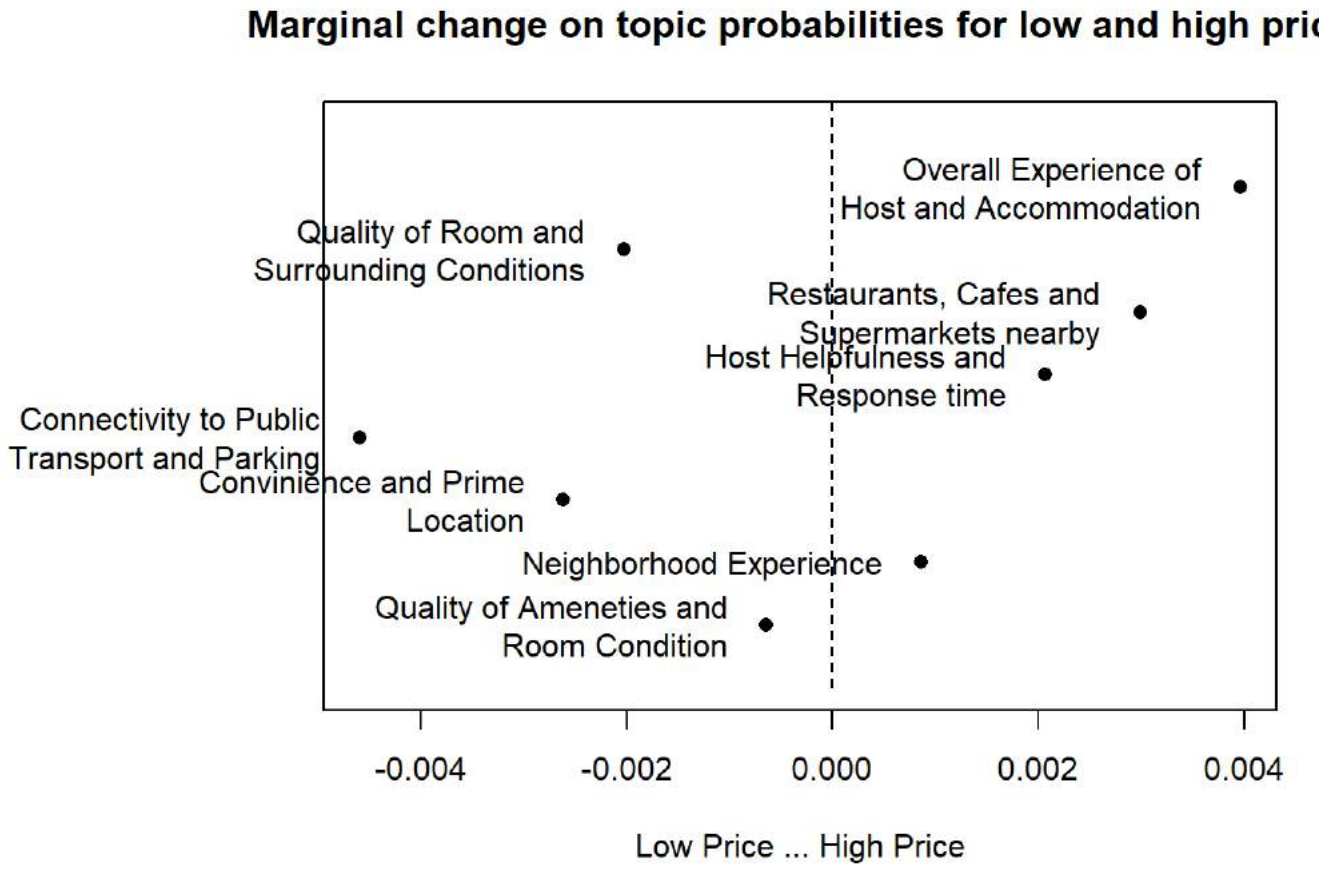

## Convinience and Prime Location




Neighborhood Experience
Expected Topic Proportion
0.12
0.16
0.20
0.24
0
2000
4000
6000
8000
10000
Price


Quality of Ameneties and Room Condition
Expected Topic Proportion
-0.02
0.00
0.02
0.04
0
2000
4000
6000
8000
10000
Price

```
# Lets also plot them as a contrast between
# the minimum and maximum price
# for that we need the margins of the upper and lower quantile
margin1 <- as.numeric(quantile(out$meta$price)[2])
margin2 <- as.numeric(quantile(out$meta$price)[4])


par(mar=c(5,9,4,1)+.1)
plot(effects, covariate = "price",
     topics = c(1:8),
     model = newyorkfit, method = "difference",
     cov.value1 = margin2, cov.value2 = margin1,
     xlab = "Low Price ... High Price",
     # xlim = c(-0.01,0.01),
     main = "Marginal change on topic probabilities for low and high price",
     custom.labels =topic_labels,
     ci.level = 0.05,
     labeltype = "custom")
```



```
# Finding number  amenities, bathrooms
i <- 1
for (i in 1:length(listings_detailed$host_verifications)) {

  listings_detailed$num_amenities[i] <- length(strsplit(listings_detailed$amenities
[i], split= " ")[[1]])
  listings_detailed$num_bath[i] <- as.numeric(gsub("[^0-9.]", "",listings_detailed$ba
throoms_text[i]))
    i <- i+1
}

all_sentiments <- readr::read_csv("sentiments.csv")

listings_detailed %>% dplyr::select(listing_id,num_bath,num_amenities,beds, room_typ
e) -> var_data

var <- all_sentiments %>% left_join(var_data)
var <- var %>% na.omit()
```

```
# We allready have the price and occupancy rate
# What we are going to do now is to regress price and occupancy rate from
# part B and this time we are going to add the average topic membership for this mont
h

# Lets merge the dataframe: to_regress_sentiment_price_occ
# with the topic probabilities
# to do that we access again the stm object with each metadata for each topic
# do not forget that we need the review date as well
# so we average it by month and year.

stm_object<- newyorkfit$theta
# remember that we just rename the column names for the
# topics so we have a better naming convention than just
# '1' or '2' etc.
colnames(stm_object) <- topic_labels

# date_df <- reviews_pre_corona_metadata %>%
#   select(listing_id, date)

causal_topic_df <- cbind(out$meta,stm_object)

# convert the date object to the month year format
#causal_topic_df$month_year <- format(causal_topic_df$date,"%Y-%m")




# Bonus point if you find a smarter way to
# create topic summaries
causal_topic_df %>%
  left_join(var) %>%
  na.omit() -> regress_stm_price_review

regress_stm_price_review <- regress_stm_price_review %>% dplyr::select(-c(listing_id,
X1, review_id, comments_grouped))
```

```
model_search_review <- lm(review_scores_rating~.,data=regress_stm_price_review)
MASS::stepAIC(model_search_review)
```

Start: AIC=596827 review_scores_rating ~ price + `Overall Experience of Host and Accommodation` + `Quality of Room and Surrounding Conditions` + `Restaurants, Cafes and Supermarkets nearby` + `Host Helpfulness and Response time` + `Connectivity to Public Transport and Parking` + `Convinience and Prime Location` + `Neighborhood Experience` + `Quality of Ameneties and Room Condition` + total_excl_count + bing_liu_sentiment + nrc_sentiment + afinn_sentiment + loughran_sentiment + num_bath + num_amenities + beds + room_type

Step: AIC=596827 review_scores_rating ~ price + `Overall Experience of Host and Accommodation` + `Quality of Room and Surrounding Conditions` + `Restaurants, Cafes and Supermarkets nearby` + `Host Helpfulness and Response time` + `Connectivity to Public Transport and Parking` + `Convinience and Prime Location` + `Neighborhood Experience` + total_excl_count + bing_liu_sentiment + nrc_sentiment + afinn_sentiment + loughran_sentiment + num_bath + num_amenities + beds + room_type

```
                                                Df Sum of Sq     RSS    AIC
```

- price 1 5 2117996 596826 2117991 596827
- loughran_sentiment 1 433 2118424 596889
- num_bath 1 1908 2119899 597105
- nrc_sentiment 1 1924 2119914 597107
- room_type 3 7296 2125287 597891
- beds 1 9616 2127607 598235
- `Overall Experience of Host and Accommodation` 1 25727 2143718 600582
- `Host Helpfulness and Response time` 1 26701 2144692 600723
- `Restaurants, Cafes and Supermarkets nearby` 1 27661 2145652 600863
- total_excl_count 1 27908 2145899 600898
- `Connectivity to Public Transport and Parking` 1 29783 2147774 601170
- `Convinience and Prime Location` 1 29949 2147940 601194
- `Neighborhood Experience` 1 31961 2149952 601486
- `Quality of Room and Surrounding Conditions` 1 41465 2159456 602858
- afinn_sentiment 1 45399 2163390 603424
- num_amenities 1 59809 2177800 605490
- bing_liu_sentiment 1 256744 2374735 632428

Step: AIC=596825.7 review_scores_rating ~ `Overall Experience of Host and Accommodation` + `Quality of Room and Surrounding Conditions` + `Restaurants, Cafes and Supermarkets nearby` + `Host Helpfulness and Response time` + `Connectivity to Public Transport and Parking` + `Convinience and Prime Location` + `Neighborhood Experience` + total_excl_count + bing_liu_sentiment + nrc_sentiment + afinn_sentiment + loughran_sentiment + num_bath + num_amenities + beds + room_type

```
                                                Df Sum of Sq     RSS    AIC
```

2117996 596826 - loughran_sentiment 1 435 2118431 596888 - nrc_sentiment 1 1919 2119915 597106 - num_bath 1 1981 2119976 597115 - room_type 3 7344 2125339 597897 - beds 1 9655 2127651 598239 - `Overall Experience of Host and Accommodation` 1 26821 2144816 600739 - `Host Helpfulness and Response time` 1 27817 2145813 600884 - total_excl_count 1 27908 2145903 600897 - `Restaurants, Cafes and Supermarkets nearby` 1 28888 2146883 601039 - `Connectivity to Public Transport and Parking` 1 30931 2148927 601335 - `Convinience and Prime Location` 1 31119 2149115 601362 - `Neighborhood Experience` 1 33331

2151326 601682 - `Quality of Room and Surrounding Conditions` 1 43106 2161101 603093 - afinn_sentiment 1 45403 2163399 603424 - num_amenities 1 59821 2177817 605491 - bing_liu_sentiment 1 257013 2375008 632462

Call: lm(formula = review_scores_rating ~ `Overall Experience of Host and Accommodation` + `Quality of Room and Surrounding Conditions` + `Restaurants, Cafes and Supermarkets nearby` + `Host Helpfulness and Response time` + `Connectivity to Public Transport and Parking` + `Convinience and Prime Location` + `Neighborhood Experience` + total_excl_count + bing_liu_sentiment + nrc_sentiment + afinn_sentiment + loughran_sentiment + num_bath + num_amenities + beds + room_type, data = regress_stm_price_review)

Coefficients: (Intercept)
132.339645
`Overall Experience of Host and Accommodation`
-47.930850
`Quality of Room and Surrounding Conditions`
-62.573406
`Restaurants, Cafes and Supermarkets nearby`
-50.202915
`Host Helpfulness and Response time`
-49.313106
`Connectivity to Public Transport and Parking`
-53.110718
`Convinience and Prime Location`
-52.520511
`Neighborhood Experience`
-53.726889
total_excl_count
0.007925
bing_liu_sentiment
11.756275
nrc_sentiment
1.156498
afinn_sentiment
0.015315
loughran_sentiment
0.259930
num_bath
0.263504
num_amenities
0.026202
beds
-0.172228
room_typeHotel room
-1.695830
room_typePrivate room
-0.215179
room_typeShared room
0.575567

```
summary(lm(review_scores_rating ~ price + `Overall Experience of Host and Accommodati
on` +
    `Quality of Room and Surrounding Conditions` + `Restaurants, Cafes and Supermarke
ts nearby` +
    `Host Helpfulness and Response time` + `Connectivity to Public Transport and Park
ing` +
    `Convinience and Prime Location` + `Neighborhood Experience` +
    total_excl_count + bing_liu_sentiment + nrc_sentiment + afinn_sentiment +
    loughran_sentiment + num_bath + num_amenities + beds + room_type,
    data = regress_stm_price_review))
```

```
##
## Call:
## lm(formula = review_scores_rating ~ price + `Overall Experience of Host and Accomm
odation` +
##     `Quality of Room and Surrounding Conditions` + `Restaurants, Cafes and Superma
rkets nearby` +
##     `Host Helpfulness and Response time` + `Connectivity to Public Transport and P
arking` +
##     `Convinience and Prime Location` + `Neighborhood Experience` +
##     total_excl_count + bing_liu_sentiment + nrc_sentiment + afinn_sentiment +
##     loughran_sentiment + num_bath + num_amenities + beds + room_type,
##     data = regress_stm_price_review)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -55.154  -1.355   0.212   1.616  17.458
##
## Coefficients:
##                                              Estimate Std. Error t value
## (Intercept)                                 1.325e+02  7.435e-01 178.163
## price                                       2.136e-05  2.560e-05   0.834
## `Overall Experience of Host and Accommodation` -4.807e+01  7.819e-01 -61.478
## `Quality of Room and Surrounding Conditions`   -6.271e+01  8.035e-01 -78.048
## `Restaurants, Cafes and Supermarkets nearby`   -5.035e+01  7.898e-01 -63.747
## `Host Helpfulness and Response time`           -4.945e+01  7.896e-01 -62.631
## `Connectivity to Public Transport and Parking` -5.325e+01  8.050e-01 -66.146
## `Convinience and Prime Location`               -5.266e+01  7.939e-01 -66.331
## `Neighborhood Experience`                      -5.387e+01  7.861e-01 -68.523
## total_excl_count                            7.925e-03  1.238e-04  64.031
## bing_liu_sentiment                          1.175e+01  6.053e-02 194.210
## nrc_sentiment                               1.158e+00  6.891e-02  16.810
## afinn_sentiment                             1.531e-02  1.875e-04  81.667
## loughran_sentiment                          2.593e-01  3.252e-02   7.973
## num_bath                                    2.616e-01  1.562e-02  16.743
## num_amenities                               2.621e-02  2.796e-04  93.736
## beds                                       -1.726e-01  4.591e-03 -37.585
## room_typeHotel room                        -1.698e+00  8.449e-02 -20.095
## room_typePrivate room                      -2.139e-01  1.059e-02 -20.204
## room_typeShared room                        5.767e-01  3.987e-02  14.465
##                                             Pr(>|t|)
## (Intercept)                                  < 2e-16 ***
## price                                          0.404
## `Overall Experience of Host and Accommodation`  < 2e-16 ***
## `Quality of Room and Surrounding Conditions`    < 2e-16 ***
## `Restaurants, Cafes and Supermarkets nearby`    < 2e-16 ***
## `Host Helpfulness and Response time`            < 2e-16 ***
## `Connectivity to Public Transport and Parking`  < 2e-16 ***
## `Convinience and Prime Location`                < 2e-16 ***
## `Neighborhood Experience`                       < 2e-16 ***
## total_excl_count                             < 2e-16 ***
## bing_liu_sentiment                           < 2e-16 ***
## nrc_sentiment                                < 2e-16 ***
## afinn_sentiment                              < 2e-16 ***
## loughran_sentiment                          1.55e-15 ***
## num_bath                                     < 2e-16 ***
## num_amenities                                < 2e-16 ***
## beds                                         < 2e-16 ***
```

```
## room_typeHotel room                               < 2e-16 ***
## room_typePrivate room                             < 2e-16 ***
## room_typeShared room                              < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.609 on 311149 degrees of freedom
## Multiple R-squared:  0.5425, Adjusted R-squared:  0.5425
## F-statistic: 1.942e+04 on 19 and 311149 DF,  p-value: < 2.2e-16
```

```
model_search_price <- lm(price~.,data=regress_stm_price_review)
MASS::stepAIC(model_search_price)
```

```
## Start:  AIC=3241161
## price ~ review_scores_rating + `Overall Experience of Host and Accommodation` +
##     `Quality of Room and Surrounding Conditions` + `Restaurants, Cafes and Superma
rkets nearby` +
##     `Host Helpfulness and Response time` + `Connectivity to Public Transport and P
arking` +
##     `Convinience and Prime Location` + `Neighborhood Experience` +
##     `Quality of Ameneties and Room Condition` + total_excl_count +
##     bing_liu_sentiment + nrc_sentiment + afinn_sentiment + loughran_sentiment +
##     num_bath + num_amenities + beds + room_type
##
##
## Step:  AIC=3241161
## price ~ review_scores_rating + `Overall Experience of Host and Accommodation` +
##     `Quality of Room and Surrounding Conditions` + `Restaurants, Cafes and Superma
rkets nearby` +
##     `Host Helpfulness and Response time` + `Connectivity to Public Transport and P
arking` +
##     `Convinience and Prime Location` + `Neighborhood Experience` +
##     total_excl_count + bing_liu_sentiment + nrc_sentiment + afinn_sentiment +
##     loughran_sentiment + num_bath + num_amenities + beds + room_type
##
##                                                Df Sum of Sq        RSS
## - total_excl_count                              1      8786 1.0389e+10
## - review_scores_rating                          1     23250 1.0389e+10
## <none>                                                      1.0389e+10
## - afinn_sentiment                               1    127081 1.0389e+10
## - loughran_sentiment                            1   6084599 1.0395e+10
## - num_amenities                                 1   6719516 1.0396e+10
## - bing_liu_sentiment                            1   7177063 1.0396e+10
## - nrc_sentiment                                 1  10314539 1.0400e+10
## - beds                                          1  84148383 1.0473e+10
## - room_type                                     3 229452126 1.0619e+10
## - num_bath                                      1 237153402 1.0626e+10
## - `Quality of Room and Surrounding Conditions`  1 450558346 1.0840e+10
## - `Connectivity to Public Transport and Parking`  1 450787239 1.0840e+10
## - `Convinience and Prime Location`              1 456377927 1.0846e+10
## - `Host Helpfulness and Response time`          1 490505839 1.0880e+10
## - `Neighborhood Experience`                     1 495200941 1.0885e+10
## - `Overall Experience of Host and Accommodation`  1 499613832 1.0889e+10
## - `Restaurants, Cafes and Supermarkets nearby`  1 516978419 1.0906e+10
##                                                    AIC
## - total_excl_count                              3241160
## - review_scores_rating                          3241160
## <none>                                          3241161
## - afinn_sentiment                               3241163
## - loughran_sentiment                            3241342
## - num_amenities                                 3241361
## - bing_liu_sentiment                            3241374
## - nrc_sentiment                                 3241468
## - beds                                          3243670
## - room_type                                     3247953
## - num_bath                                      3248182
## - `Quality of Room and Surrounding Conditions`  3254370
## - `Connectivity to Public Transport and Parking`  3254376
## - `Convinience and Prime Location`              3254537
## - `Host Helpfulness and Response time`          3255514
```

```
## - `Neighborhood Experience`                       3255648
## - `Overall Experience of Host and Accommodation` 3255775
## - `Restaurants, Cafes and Supermarkets nearby`   3256270
##
## Step:  AIC=3241160
## price ~ review_scores_rating + `Overall Experience of Host and Accommodation` +
##     `Quality of Room and Surrounding Conditions` + `Restaurants, Cafes and Superma
rkets nearby` +
##     `Host Helpfulness and Response time` + `Connectivity to Public Transport and P
arking` +
##     `Convinience and Prime Location` + `Neighborhood Experience` +
##     bing_liu_sentiment + nrc_sentiment + afinn_sentiment + loughran_sentiment +
##     num_bath + num_amenities + beds + room_type
##
##                                              Df Sum of Sq        RSS
## - review_scores_rating                        1     20370 1.0389e+10
## <none>                                                    1.0389e+10
## - afinn_sentiment                             1    204692 1.0390e+10
## - loughran_sentiment                          1   6189024 1.0396e+10
## - num_amenities                               1   6725113 1.0396e+10
## - bing_liu_sentiment                          1   7541893 1.0397e+10
## - nrc_sentiment                               1  10314329 1.0400e+10
## - beds                                        1  84224473 1.0474e+10
## - room_type                                   3 229622552 1.0619e+10
## - num_bath                                    1 237965077 1.0627e+10
## - `Quality of Room and Surrounding Conditions`    1 450549701 1.0840e+10
## - `Connectivity to Public Transport and Parking`  1 450780005 1.0840e+10
## - `Convinience and Prime Location`                1 456371601 1.0846e+10
## - `Host Helpfulness and Response time`            1 490500846 1.0880e+10
## - `Neighborhood Experience`                       1 495192155 1.0885e+10
## - `Overall Experience of Host and Accommodation`  1 499605592 1.0889e+10
## - `Restaurants, Cafes and Supermarkets nearby`    1 516969663 1.0906e+10
##                                                 AIC
## - review_scores_rating                       3241158
## <none>                                       3241160
## - afinn_sentiment                            3241164
## - loughran_sentiment                         3241343
## - num_amenities                              3241359
## - bing_liu_sentiment                         3241383
## - nrc_sentiment                              3241466
## - beds                                       3243670
## - room_type                                  3247956
## - num_bath                                   3248204
## - `Quality of Room and Surrounding Conditions`   3254368
## - `Connectivity to Public Transport and Parking` 3254374
## - `Convinience and Prime Location`               3254535
## - `Host Helpfulness and Response time`           3255512
## - `Neighborhood Experience`                      3255646
## - `Overall Experience of Host and Accommodation` 3255773
## - `Restaurants, Cafes and Supermarkets nearby`   3256268
##
## Step:  AIC=3241158
## price ~ `Overall Experience of Host and Accommodation` + `Quality of Room and Surr
ounding Conditions` +
##     `Restaurants, Cafes and Supermarkets nearby` + `Host Helpfulness and Response
time` +
##     `Connectivity to Public Transport and Parking` + `Convinience and Prime Locati
on` +
```

```
##     `Neighborhood Experience` + bing_liu_sentiment + nrc_sentiment +
##     afinn_sentiment + loughran_sentiment + num_bath + num_amenities +
##     beds + room_type
##
##                                                Df Sum of Sq        RSS
## <none>                                                      1.0389e+10
## - afinn_sentiment                               1    298133 1.0390e+10
## - loughran_sentiment                            1   6191377 1.0396e+10
## - num_amenities                                 1   6793043 1.0396e+10
## - bing_liu_sentiment                            1   8649755 1.0398e+10
## - nrc_sentiment                                 1  10298410 1.0400e+10
## - beds                                          1  84457770 1.0474e+10
## - room_type                                     3 229818603 1.0619e+10
## - num_bath                                      1 238220156 1.0628e+10
## - `Connectivity to Public Transport and Parking`  1 456633176 1.0846e+10
## - `Quality of Room and Surrounding Conditions`    1 458891825 1.0848e+10
## - `Convinience and Prime Location`                1 462335170 1.0852e+10
## - `Host Helpfulness and Response time`            1 496208107 1.0886e+10
## - `Neighborhood Experience`                       1 502222508 1.0892e+10
## - `Overall Experience of Host and Accommodation`  1 505224459 1.0895e+10
## - `Restaurants, Cafes and Supermarkets nearby`    1 523288434 1.0913e+10
##                                                    AIC
## <none>                                          3241158
## - afinn_sentiment                               3241165
## - loughran_sentiment                            3241342
## - num_amenities                                 3241360
## - bing_liu_sentiment                            3241415
## - nrc_sentiment                                 3241465
## - beds                                          3243676
## - room_type                                     3247960
## - num_bath                                      3248211
## - `Connectivity to Public Transport and Parking` 3254541
## - `Quality of Room and Surrounding Conditions`   3254606
## - `Convinience and Prime Location`               3254704
## - `Host Helpfulness and Response time`           3255674
## - `Neighborhood Experience`                      3255846
## - `Overall Experience of Host and Accommodation` 3255932
## - `Restaurants, Cafes and Supermarkets nearby`   3256447
```

```
##
## Call:
## lm(formula = price ~ `Overall Experience of Host and Accommodation` +
##     `Quality of Room and Surrounding Conditions` + `Restaurants, Cafes and Superma
rkets nearby` +
##     `Host Helpfulness and Response time` + `Connectivity to Public Transport and P
arking` +
##     `Convinience and Prime Location` + `Neighborhood Experience` +
##     bing_liu_sentiment + nrc_sentiment + afinn_sentiment + loughran_sentiment +
##     num_bath + num_amenities + beds + room_type, data = regress_stm_price_review)
##
## Coefficients:
##                                     (Intercept)
##                                       -6.108e+03
## `Overall Experience of Host and Accommodation`
##                                        6.578e+03
##   `Quality of Room and Surrounding Conditions`
##                                        6.456e+03
##   `Restaurants, Cafes and Supermarkets nearby`
##                                        6.757e+03
##           `Host Helpfulness and Response time`
##                                        6.586e+03
## `Connectivity to Public Transport and Parking`
##                                        6.453e+03
##               `Convinience and Prime Location`
##                                        6.401e+03
##                      `Neighborhood Experience`
##                                        6.595e+03
##                             bing_liu_sentiment
##                                        6.731e+01
##                                  nrc_sentiment
##                                       -8.458e+01
##                                afinn_sentiment
##                                        2.303e-02
##                             loughran_sentiment
##                                        3.086e+01
##                                       num_bath
##                                        9.126e+01
##                                   num_amenities
##                                       -2.791e-01
##                                             beds
##                                        1.610e+01
##                           room_typeHotel room
##                                        9.564e+01
##                         room_typePrivate room
##                                       -5.853e+01
##                          room_typeShared room
##                                       -5.154e+01
```

```
summary(lm(price ~ review_scores_rating + `Overall Experience of Host and Accommodati
on` +
    `Quality of Room and Surrounding Conditions` + `Restaurants, Cafes and Supermarke
ts nearby` +
    `Host Helpfulness and Response time` + `Connectivity to Public Transport and Park
ing` +
    `Convinience and Prime Location` + `Neighborhood Experience` +
    total_excl_count + bing_liu_sentiment + nrc_sentiment + afinn_sentiment +
    loughran_sentiment + num_bath + num_amenities + beds + room_type,
    data = regress_stm_price_review))
```

```
##
## Call:
## lm(formula = price ~ review_scores_rating + `Overall Experience of Host and Accomm
odation` +
##     `Quality of Room and Surrounding Conditions` + `Restaurants, Cafes and Superma
rkets nearby` +
##     `Host Helpfulness and Response time` + `Connectivity to Public Transport and P
arking` +
##     `Convinience and Prime Location` + `Neighborhood Experience` +
##     total_excl_count + bing_liu_sentiment + nrc_sentiment + afinn_sentiment +
##     loughran_sentiment + num_bath + num_amenities + beds + room_type,
##     data = regress_stm_price_review)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -559.2  -50.6   -7.1   34.2 9453.8
##
## Coefficients:
##                                              Estimate Std. Error  t value
## (Intercept)                                -6.122e+03  5.355e+01 -114.312
## review_scores_rating                        1.048e-01  1.256e-01    0.834
## `Overall Experience of Host and Accommodation`  6.583e+03  5.382e+01  122.323
## `Quality of Room and Surrounding Conditions`    6.462e+03  5.563e+01  116.163
## `Restaurants, Cafes and Supermarkets nearby`    6.762e+03  5.434e+01  124.431
## `Host Helpfulness and Response time`            6.591e+03  5.438e+01  121.203
## `Connectivity to Public Transport and Parking`  6.458e+03  5.558e+01  116.192
## `Convinience and Prime Location`                6.407e+03  5.480e+01  116.910
## `Neighborhood Experience`                       6.600e+03  5.420e+01  121.782
## total_excl_count                           -4.476e-03  8.725e-03   -0.513
## bing_liu_sentiment                          6.579e+01  4.487e+00   14.661
## nrc_sentiment                              -8.482e+01  4.826e+00  -17.576
## afinn_sentiment                             2.590e-02  1.327e-02    1.951
## loughran_sentiment                          3.074e+01  2.277e+00   13.499
## num_bath                                    9.121e+01  1.082e+00   84.276
## num_amenities                              -2.816e-01  1.985e-02  -14.186
## beds                                        1.612e+01  3.210e-01   50.201
## room_typeHotel room                         9.583e+01  5.919e+00   16.191
## room_typePrivate room                      -5.850e+01  7.346e-01  -79.634
## room_typeShared room                       -5.159e+01  2.792e+00  -18.483
##                                            Pr(>|t|)
## (Intercept)                                  <2e-16 ***
## review_scores_rating                         0.4040
## `Overall Experience of Host and Accommodation`  <2e-16 ***
## `Quality of Room and Surrounding Conditions`    <2e-16 ***
## `Restaurants, Cafes and Supermarkets nearby`    <2e-16 ***
## `Host Helpfulness and Response time`            <2e-16 ***
## `Connectivity to Public Transport and Parking`  <2e-16 ***
## `Convinience and Prime Location`                <2e-16 ***
## `Neighborhood Experience`                       <2e-16 ***
## total_excl_count                             0.6080
## bing_liu_sentiment                           <2e-16 ***
## nrc_sentiment                                <2e-16 ***
## afinn_sentiment                              0.0511 .
## loughran_sentiment                           <2e-16 ***
## num_bath                                     <2e-16 ***
## num_amenities                                <2e-16 ***
## beds                                         <2e-16 ***
```

```
## room_typeHotel room                                  <2e-16 ***
## room_typePrivate room                                <2e-16 ***
## room_typeShared room                                 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 182.7 on 311149 degrees of freedom
## Multiple R-squared:  0.168,  Adjusted R-squared:  0.168
## F-statistic:  3308 on 19 and 311149 DF,  p-value: < 2.2e-16
```