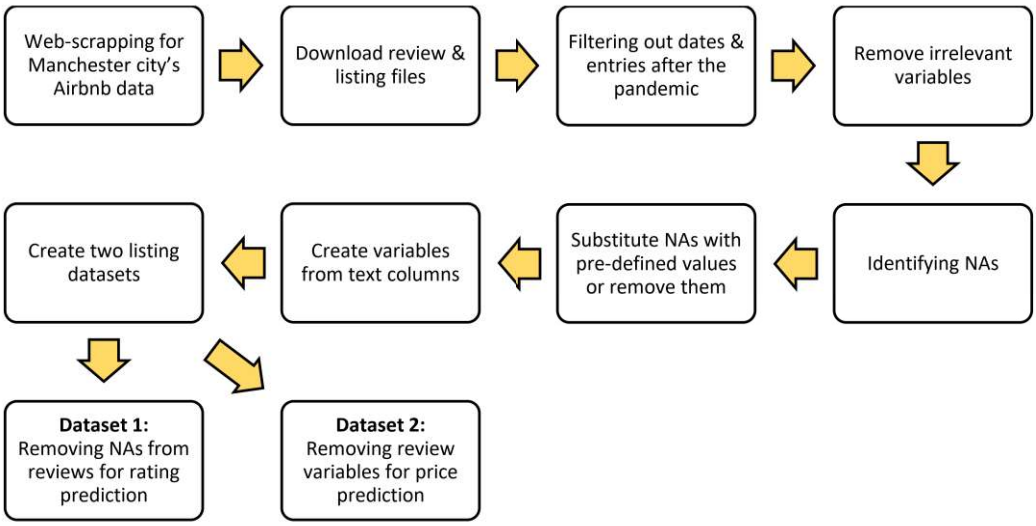
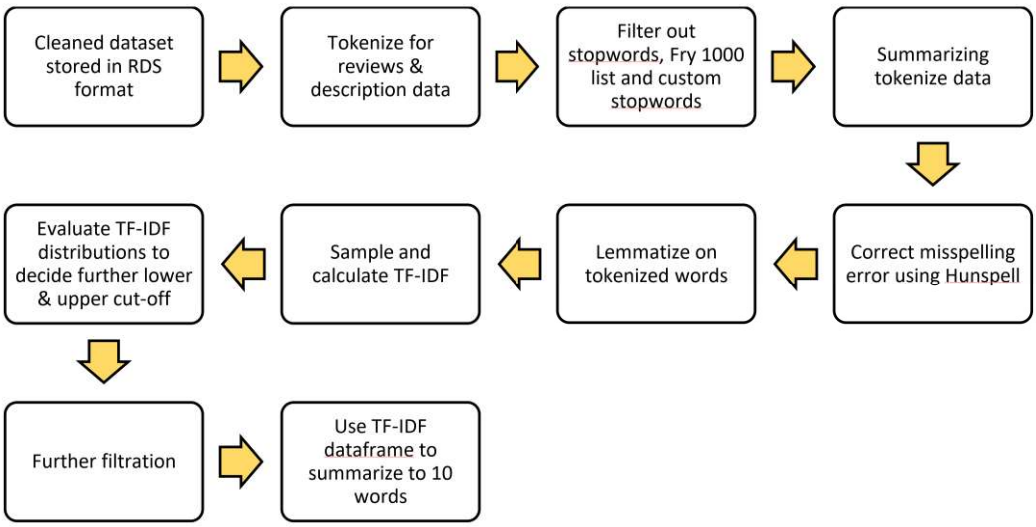
Table of Contents

Data Preparation

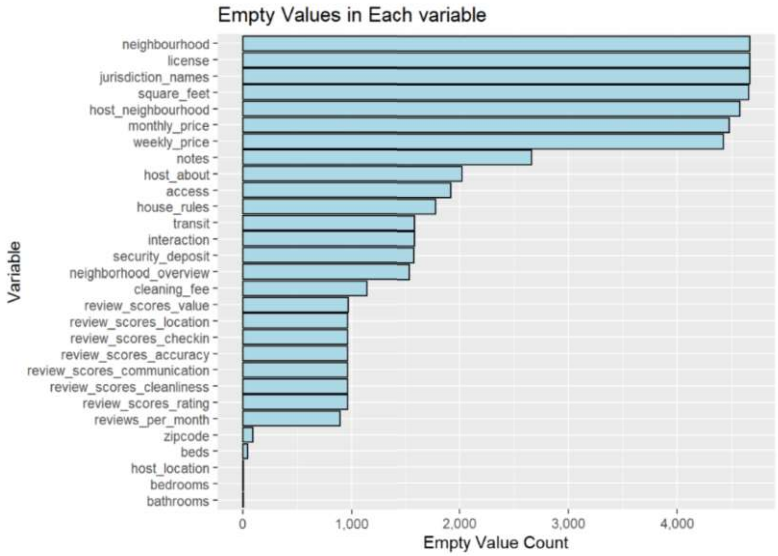
Data preparation



Text cleaning

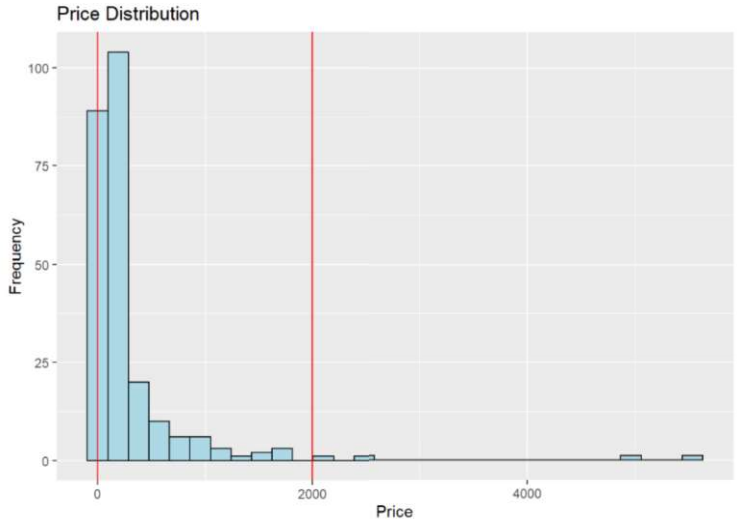


After downloading review and listing data of May 2020 we have removed variables which were not useful for analysis such as URLs and details of scrapping. After that we have dealt with the null values, we have assumed some values while removing the rest. Following is the graph for empty observations in each variable.

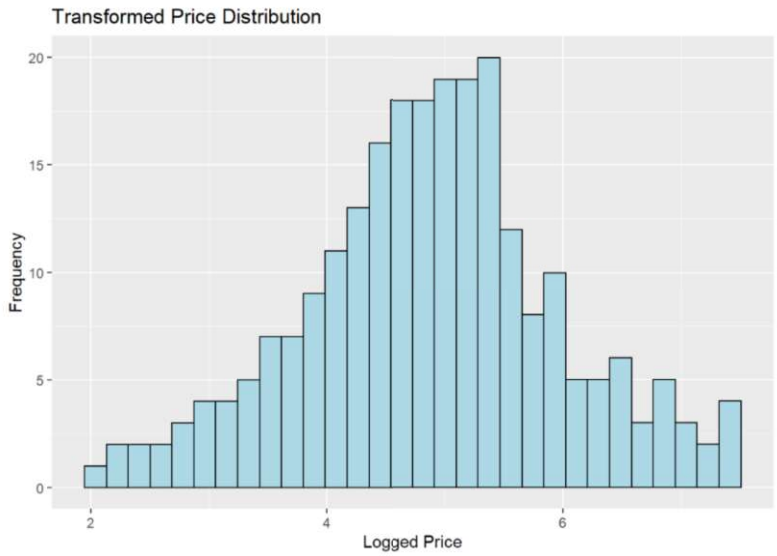


Steps and assumptions for tidying the dataset include:

1. Performed analysis only on English language descriptions and comments.
2. Removed listings with fewer than 4 reviews (based on quantile) as they could exaggerate results.
3. NAs in security deposit, cleaning fee, extra people, and minimum nights observations are assumed to be 0.
4. Removed reviews by professional reviewers who have reviewed more than 5 times within a year.
5. Removed outliers in price equal to $0 and greater than $2000.



6. Transformed price with log to normalise distribution as it was left skewed by 0.956, skewness after log was -0.065.



7. Considered only those amenities that are not very common or rare. The set threshold was between 5% to 80%.
8. Created two data frames with listing details, first with review information after removing nulls (for review analysis) and second without review variables (pricing analysis - review variables are not helpful).
9. Removed reviews with less than 5 words as they do not aid in sentiment analysis.

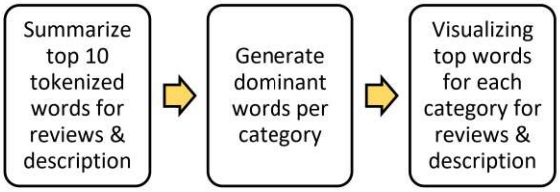
Additional variables have been extracted to test their significance as shown below:

1. Gender of host using their first name
2. Number of characters and words in a listing name
3. Extracted total number of amenities for each listing and created new variable for amenities using one-hot encoding
4. Total number of verifications in each listing

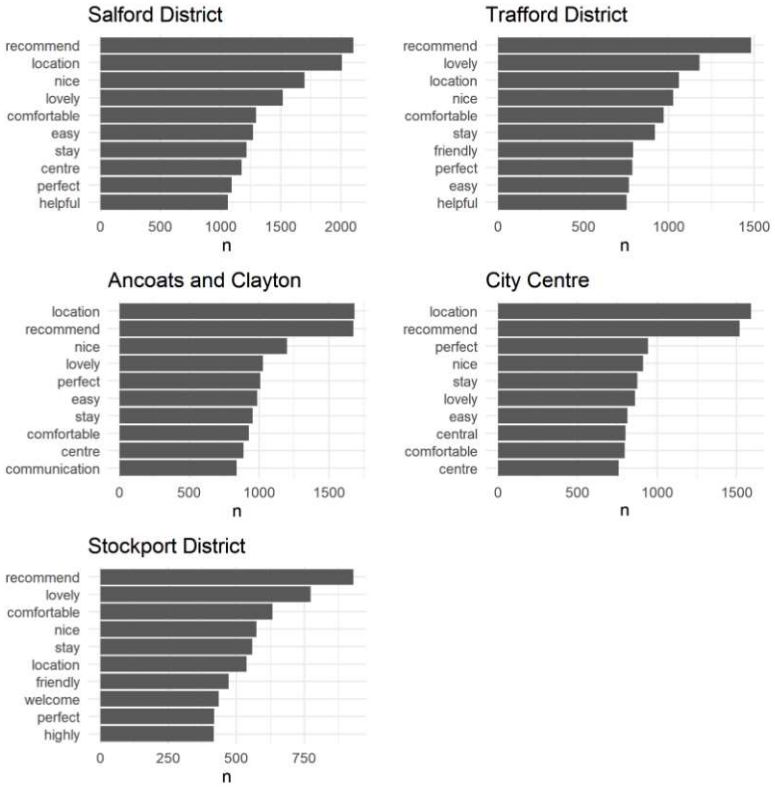
Part A

Part A (a)

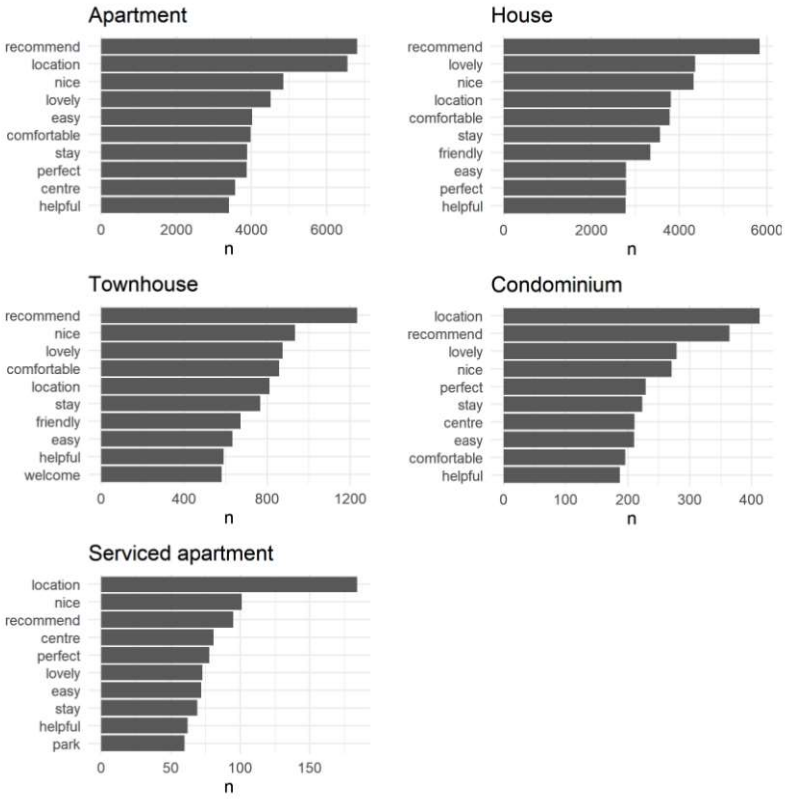
In this part, cleaned description and review text will be grouped by variables such as neighbourhood_cleansed, property_type, room_type, host_verified, and host_is_superhost and dominant words will be generated for each of these categories for both review and description text.



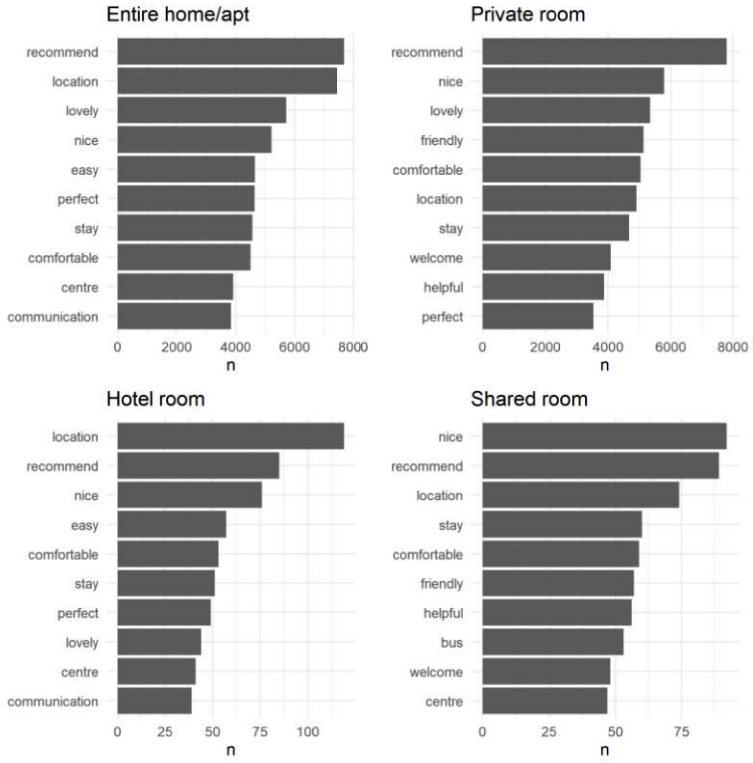
Review Text



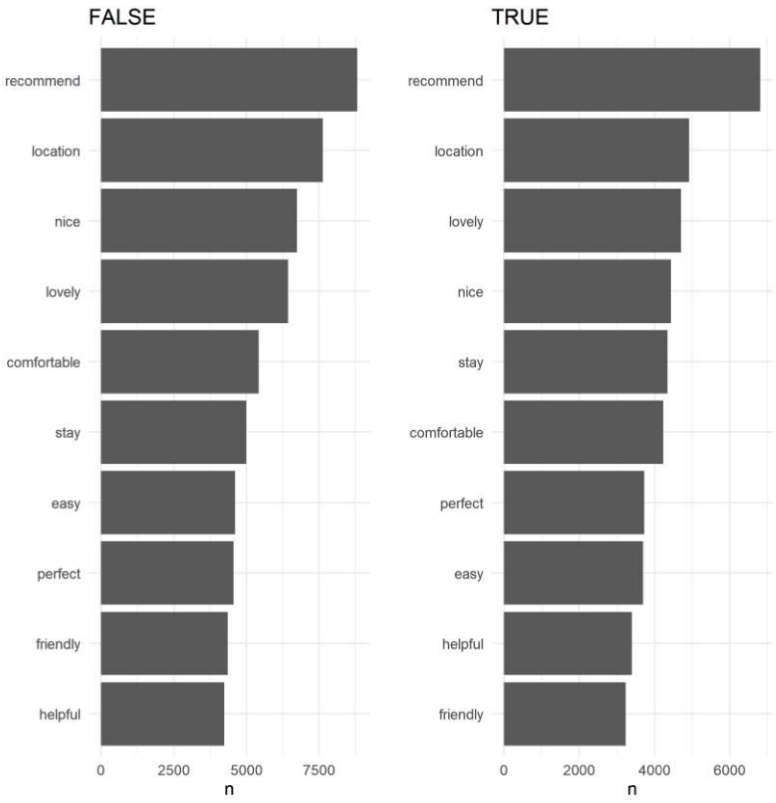
Dominant words by neighbourhood



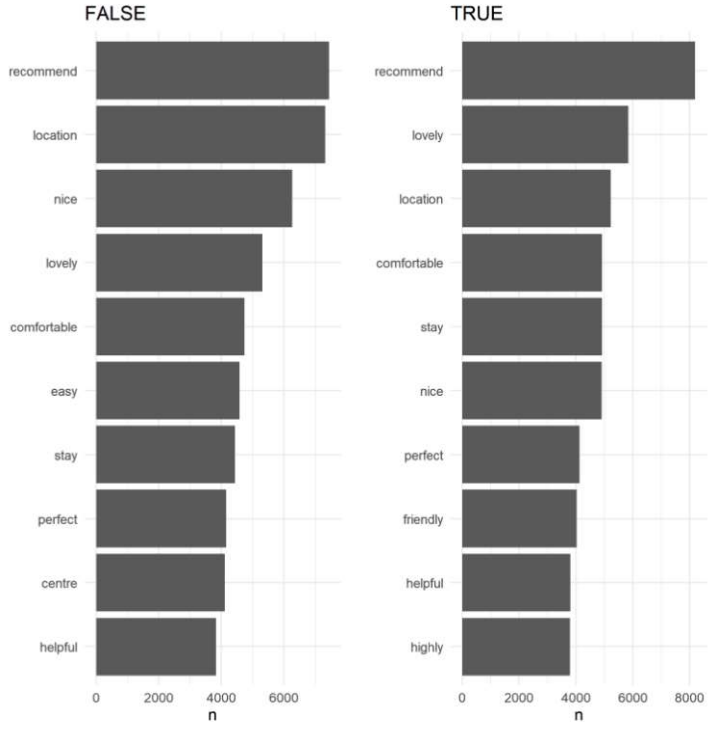
Dominant words by property type



Dominant words by room type

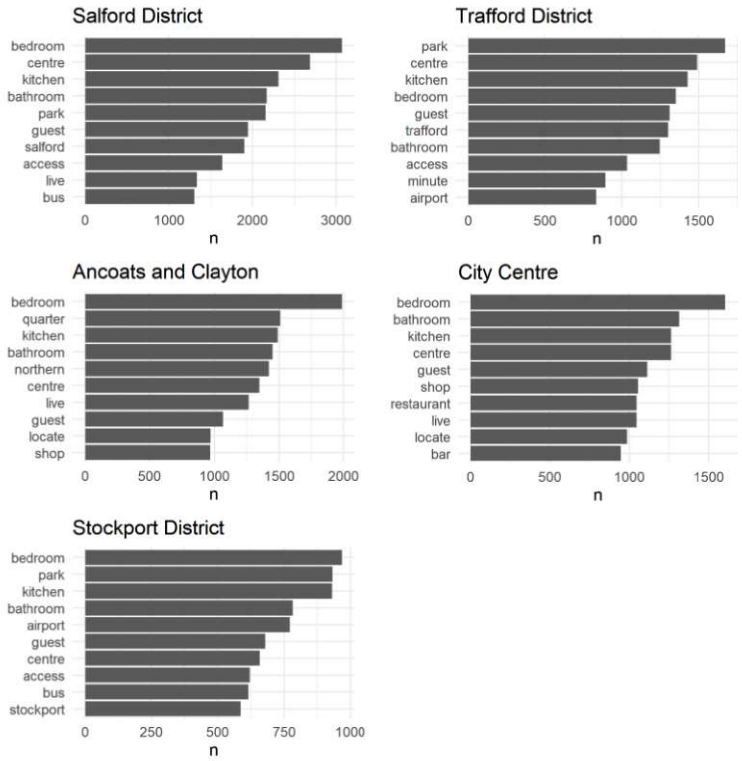


Dominant words by host verified

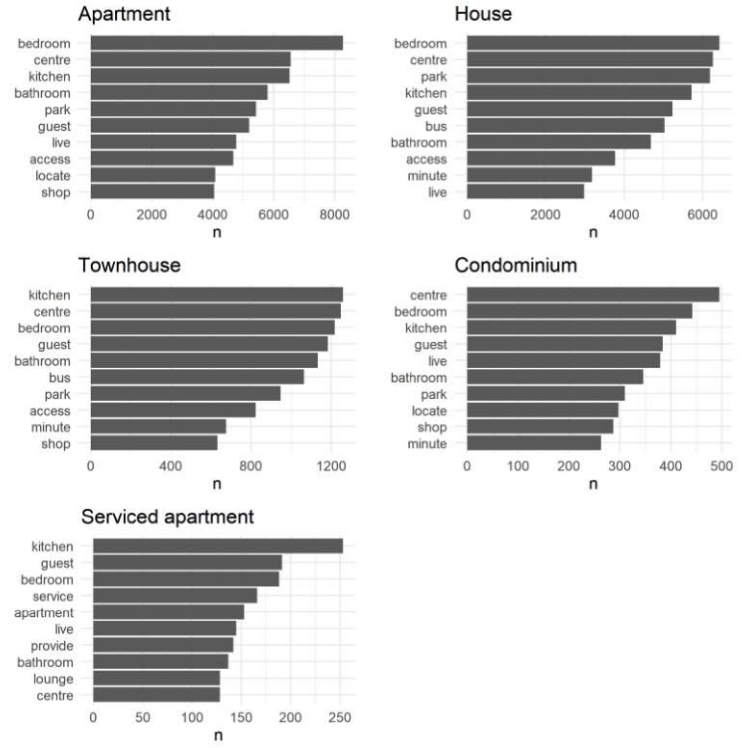


Dominant words by super host

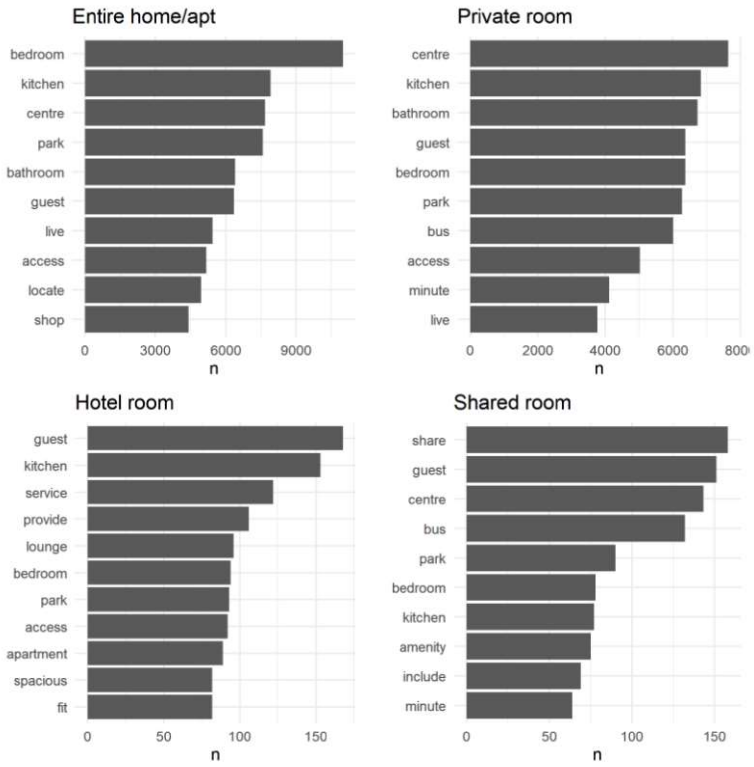
Description Text



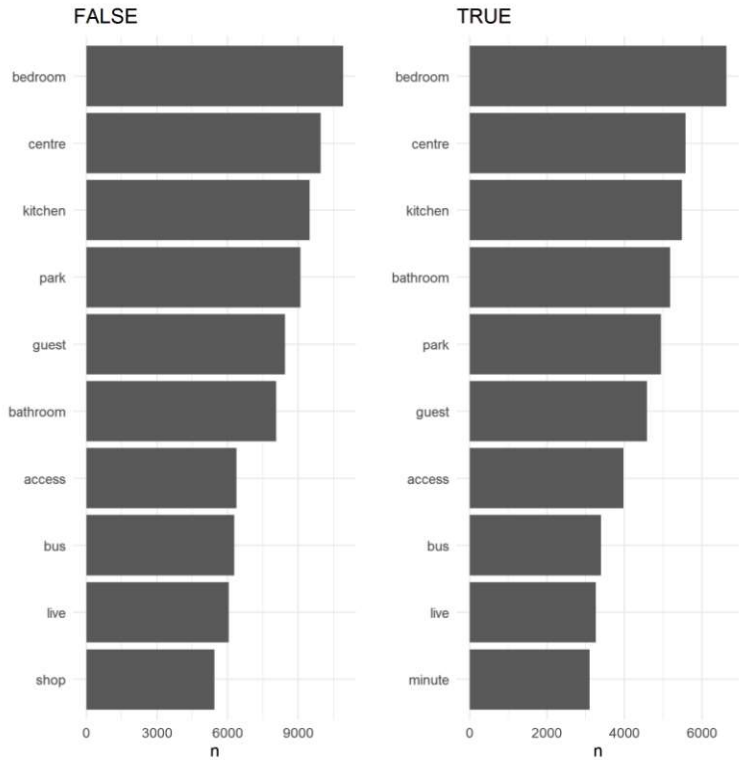
Dominant words by neighbourhood



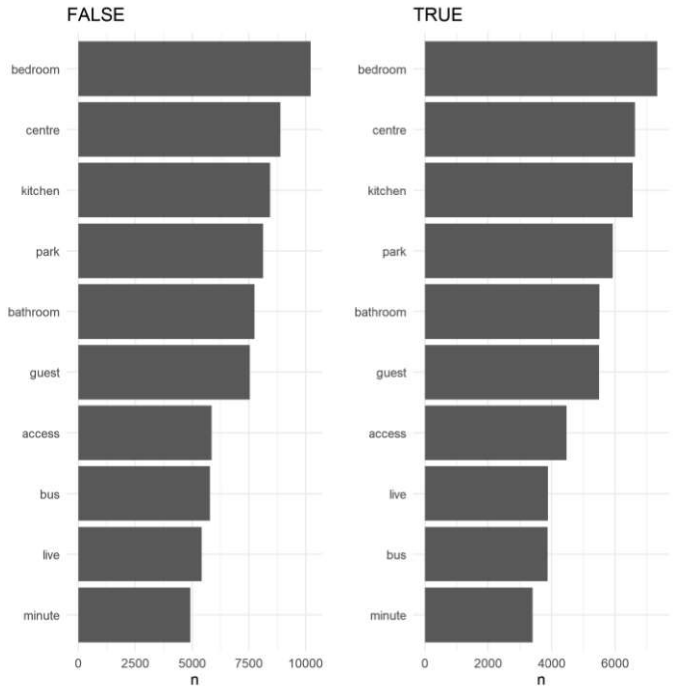
Dominant words by property type



Dominant words by room type



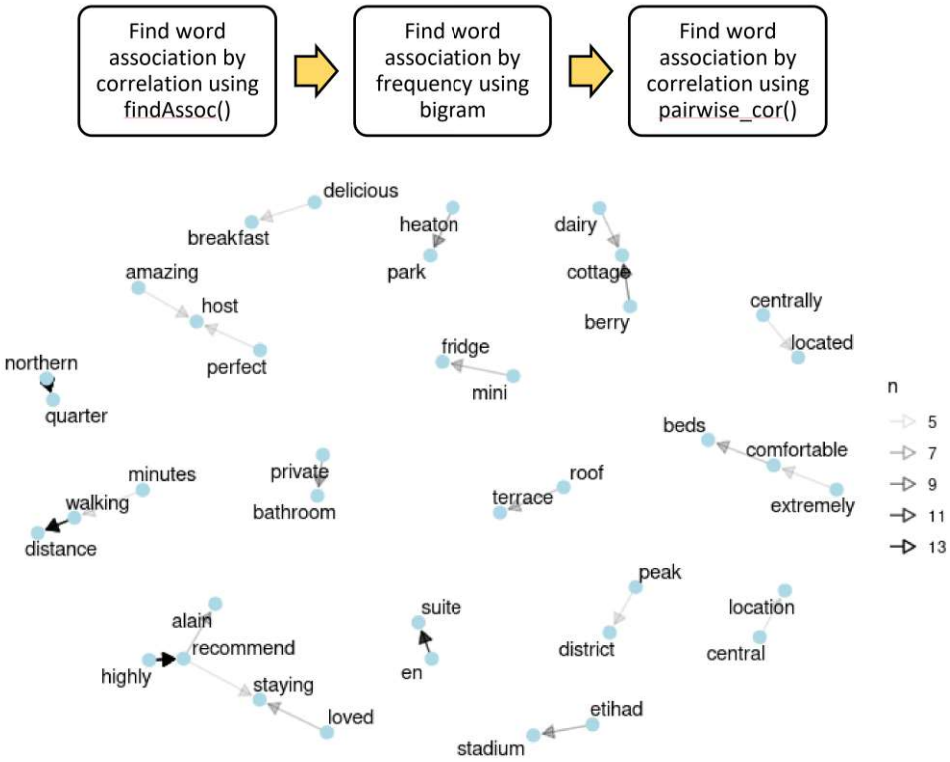
Dominant words by host verified



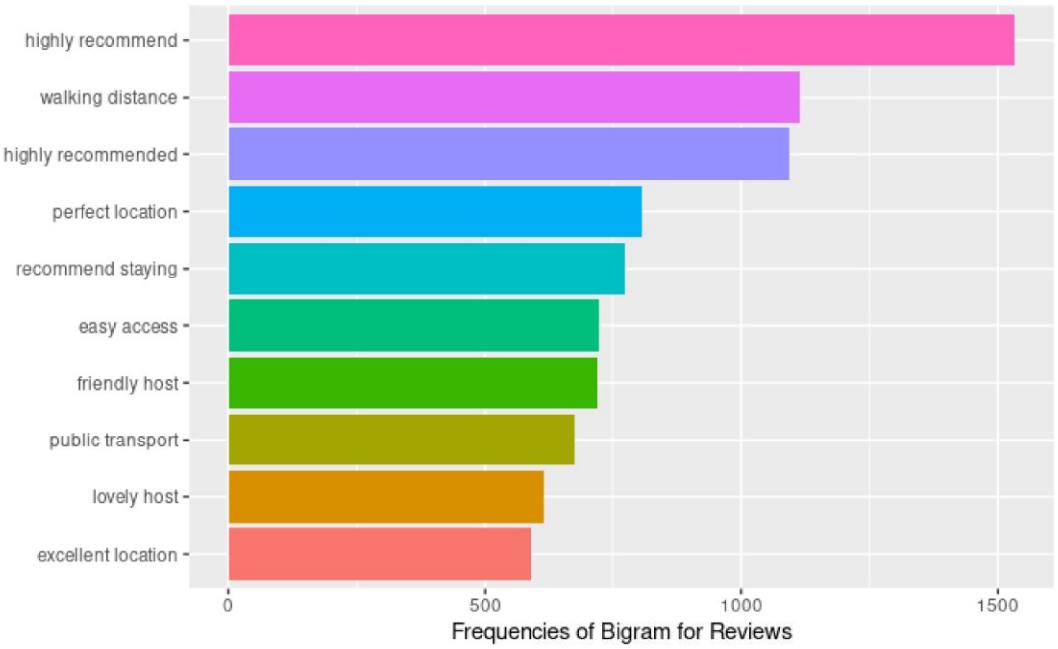
Dominant words by super host

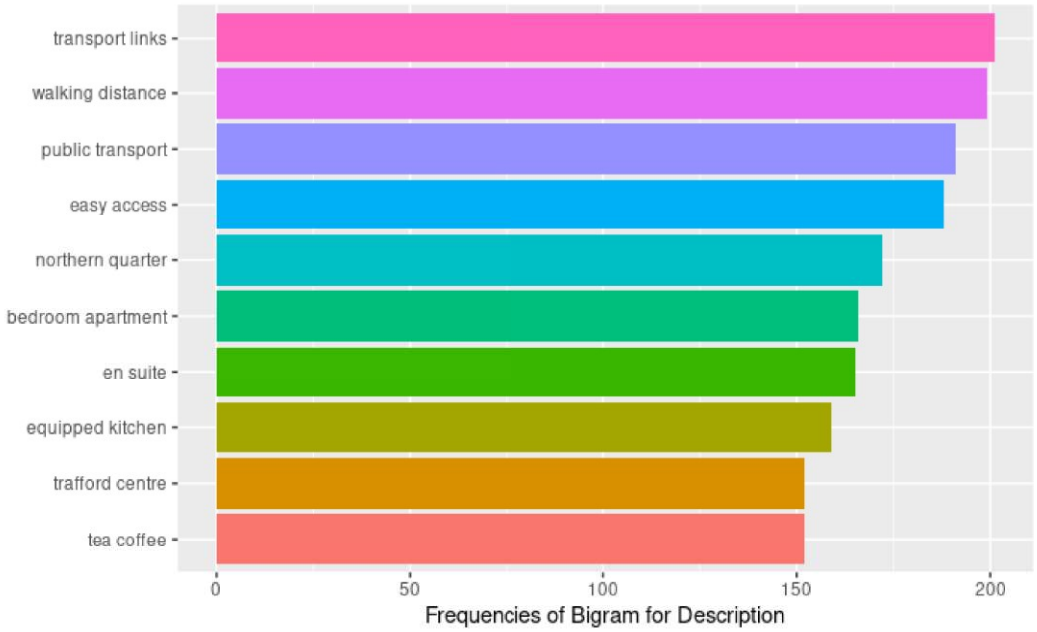
Part A (b)

In this part, two perspectives will be explored: reviewers' and hosts'. We will find correlations of word combinations for both perspectives using findAssoc; performing bigram and finding the top 10 frequent words and using pairwise_cor() to find correlations of word combinations for reviewers' perspective.



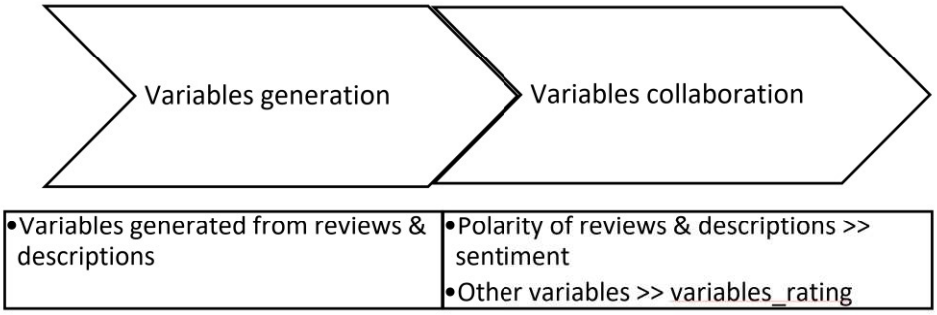
The cluster of words shows bigram formation patterns for the 'reviews' column. The arrows indicate how the words are ordered and the bolder the arrow is, the higher the frequency of the word association gets.





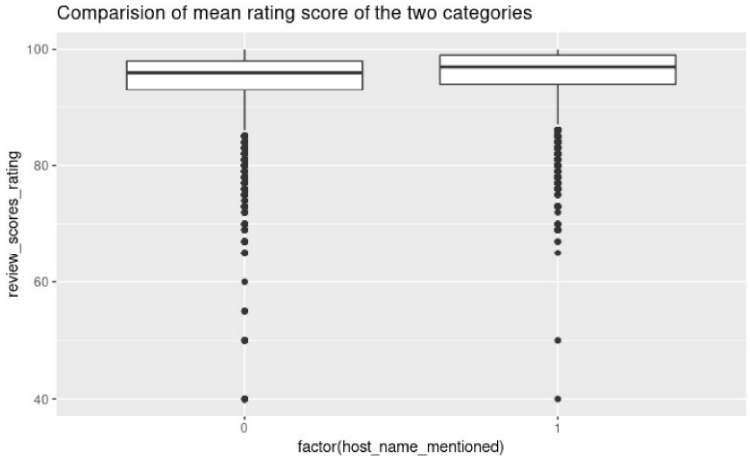
For the scope of this dataset, bigram better represents the most common word combinations used to describe a property. Bigrams provide more meaningful interpretations of the context. For future analysis, a researcher can consider exploring n-grams when n=3 and decide which analysis would provide a more meaningful representation.

Part A (c)



In this part, we firstly categorized the rating score to have an overview of rating scores' distribution. We divided it into two rating scores after seeing the distribution. Rating scores below 96 were assigned to rating 1 while scores above 96 were assigned to rating 2. We then saw the top 10 words with highest tf_idf for both the rating categories. Then, several variables such as number of characters in comments, formality, polarity, readability and diversity were generated from textual columns such as comments and description. For the further analysis in part B, these variables apart from polarity of comments and description are combined and stored in a data frame named *variables_rating*. Whereas, polarities are saved in data frame named *sentiment*, which should combine with sentiment scores latter in part B.

Furthermore, a new variable was created which was whether host name is mentioned in comment or not. By using t.test it was concluded that the mention of host name has a significant impact on the rating score of the listing. The plot below shows the difference in the two groups.



Part A (d)

From 'description' variable, we have evaluated features such as readability, number of words, number of sentences, and formality. The results were all significant, but the R-sq value for description was very low. Thus, we drop 'description' from our analysis.

```
model1 <- lm(price_log~FK_grd.lvl, data = listings_price_df)
summary(model1) #Multiple R-squared:  0.00966,  Adjusted R-squared:  0.009439
model2 <- lm(price_log~log(FK_grd.lvl), data = listings_price_df)
summary(model2) #Multiple R-squared:  0.03078,  Adjusted R-squared:  0.03056
model3 <- lm(price_log~FK_read.ease, data = listings_price_df)
summary(model3) #Multiple R-squared:  0.02882,  Adjusted R-squared:  0.0286
model4 <- lm(price_log~log(FK_read.ease), data = listings_price_df)
summary(model4) #Multiple R-squared:  0.03178,  Adjusted R-squared:  0.03156
model5 <- lm(price_log~formality, data = listings_price_df)
summary(model5) #Multiple R-squared:  0.00599,  Adjusted R-squared:  0.005768
model6 <- lm(price_log~log(formality), data = listings_price_df)
summary(model6) #Multiple R-squared:  0.006606, Adjusted R-squared:  0.006384
model_7 <- lm(price_log~word.count, data = listings_price_df)
summary(model_7) #Multiple R-squared:  0.003697,  Adjusted R-squared:  0.003475
model_8 <- lm(price_log~syllable.count, data = listings_price_df)
summary(model_8) #Multiple R-squared:  0.01394, Adjusted R-squared:  0.01372
```

The features of description are significant, but they have very low R-squared value. These variables were individually placed in base model, but there was no significant improvement in the model. stepAIC was used (works on forward selection method) to shortlist the variables and then tested the recommended variable. On checking multivariate regression grade level of description was tested insignificant variable.

We identified that 'zip-code' might affect our analysis significantly after evaluating our regression result. It appears that 'zip-code' created a huge difference between the R-squared and adjusted R-squared results. However, the regression performance dropped significantly when we try discarding the variable. Thus, to avoid overfitting, the base model was constructed after running

individual regression test and selecting only the most significant model [forward selection]. We also discarded insignificant variables from the analysis.

```
#Recommended model by stepAIC
model_b6 <- lm(formula = price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + bathrooms + beds + zipcode + log(FK_grd.lvl),
    data = listings_price_df)

summary(model_b6)
#Multiple R-squared:  0.8884, Adjusted R-squared:  0.7538

#Base model
summary(lm(price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + bathrooms + beds + zipcode,
    data = model_data))
#Multiple R-squared:  0.8882, Adjusted R-squared:  0.7535

anova(model_base, model_b6)
#p - 0.0765
```

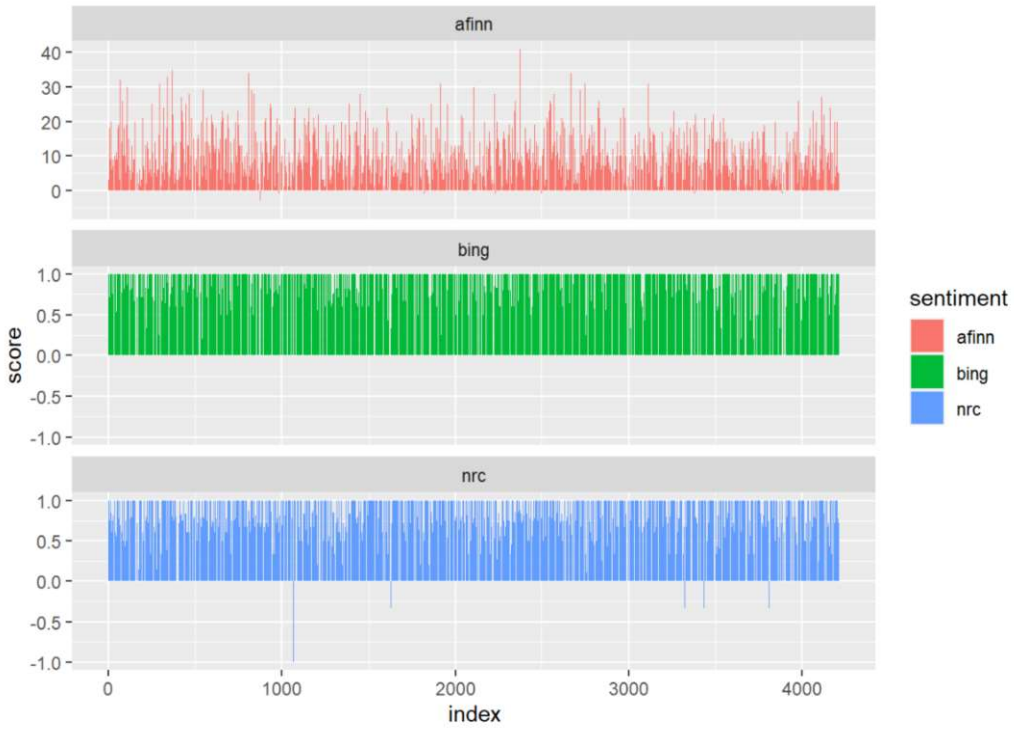
We can conclude that by adding the textual descriptions do not affect price of the property significantly. Additionally, we identified other variables which provide higher property price prediction in Manchester.

Part B

Data Preparation	Regression 1	Regression 2
•Generating sentiment scores •Counting exclamations and capital letters •Further data cleaning	•Test significance of individual variables •**Step AIC:** Generate combinations of variables that will result in the highe adjusted Rsquared •Baseline regression for sentiment analysis •Run regression on model with its sentiment score that will improve baseline regression most	•Looping •**Step AIC** •Baseline regression •Run regression on model with its sentiment score that will improve baseline regression most •Most appropriate regression

Sentiment Analysis and Price Prediction

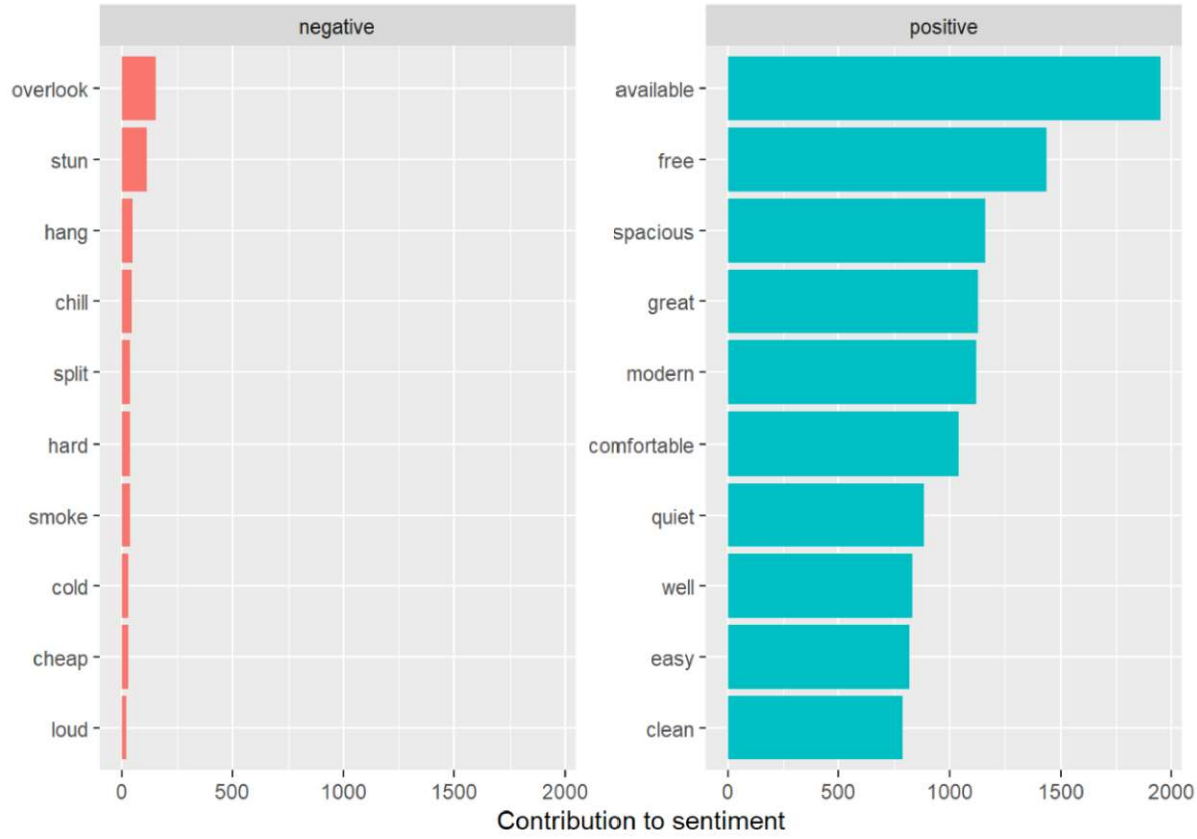
Sentiment analysis is performed on the variables to improve the price prediction model. Three dictionaries were used Afinn, Bing and nrc, Loughran was not included as it is a specialised dictionary for financial documents. By using these dictionaries, we got multiple sentiments and polarity through our calculation with the qdap function.



Based on these results, Afinn indicates variation in the score of sentiment in each description while other dictionaries give highly positive scores except a few in nrc. After individually testing the significance of variables, we added them in a base model for forward selection, selected variables were anticipation, joy, positive, sadness, surprise, and number of capital letters in description. In

the multivariate model, variables extracted from sentiment and features in text such as expressive punctuations and capital words were not a significant addition to the model.

An interaction between variables was selected and the R-sq between both were equivalent to base model with 7 variables. The final variables were zip-code and log of accommodation, which were selected on the principle of parsimony. The analysis is that in Greater Manchester the price is highly dependent on location and number of people a listing allows.



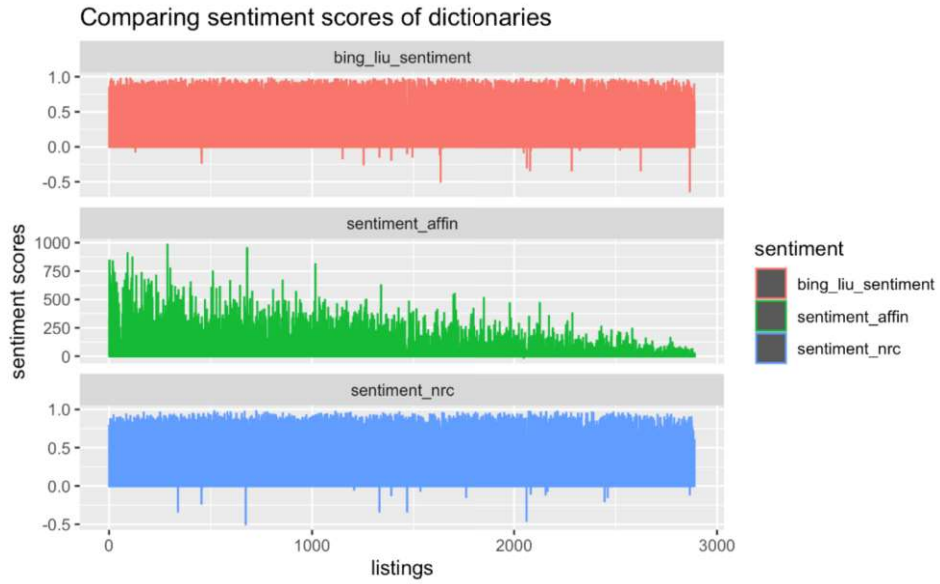
This graph of bing represents that positivity is highly focused in description which will attract customer and help in building initial trust.

Sentiment Analysis and Rating Score Prediction

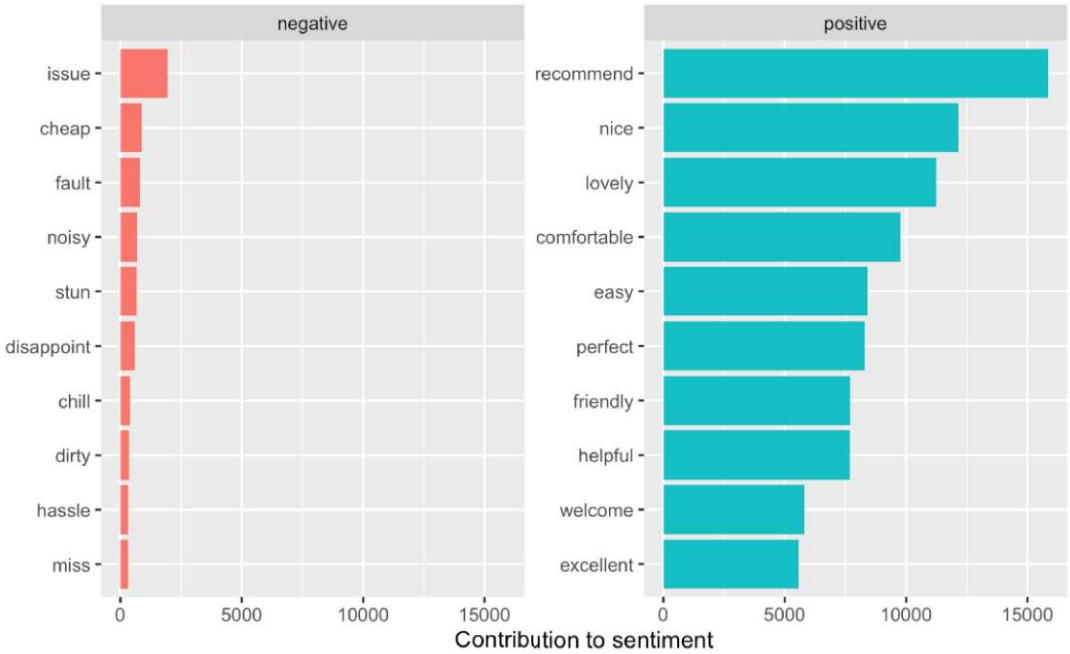
The rating score part of sentiment analysis processed through data preparation and two groups of regressions on after another. Generating and collaborating these variables with is the major task for the data preparation. At this stage, sentiment scores of three dictionaries which are Bing Liu, NRC and Affin, are generated and their results are saved in data frame sentiment, which stored polarity of comments and description in part A (c). Meanwhile, variables which illustrate average number of capital letters and exclamation are extracted from comments as the following regression might require. These newly generated variables from comments then collaborate with data frame named variables_rating which generate in partA (c). At the same time, the dependent variable in the following regressions, review_socres_rating, is placed in the first column while collaborating

with the variables_rating. Then, the data preparation is finished after cleaning on NA observations and irrelevant variables.

The graph below shows the comparision of different sentiment scores of the three different dictionaries we used.



The graphs below show the top 10 positive and negative words from comments and how they contribute to the sentiment score.



In first group of regressions, a loop is processed to check the significance of each variables that their p-value is smaller than 0.05. These significant variables are manually selected and imported to stepAIC function to find out the best combination of variables with the lowest AIC of 5525.78. As there are insignificant variables from the result of stepAIC function, the baseline model for further analysis is generated after removing these insignificant variables manually. To select which sentiment score improve the baseline regression most, data frame *sentiment* is combined with *variable_rating* and five regressions are built that each regression improves baseline regression by increase a sentiment score. By comparing adjusted R^2, regression with bing_liu_sentiment has the best performance that its figure reaches 0.6976. By removing insignificant variables such as wordcount_description, host_greets_you and microwave, the adjusted R^2 increase 0.0003. The performance of the regression is considerable, but some feasible methods exist to improve the regression further.

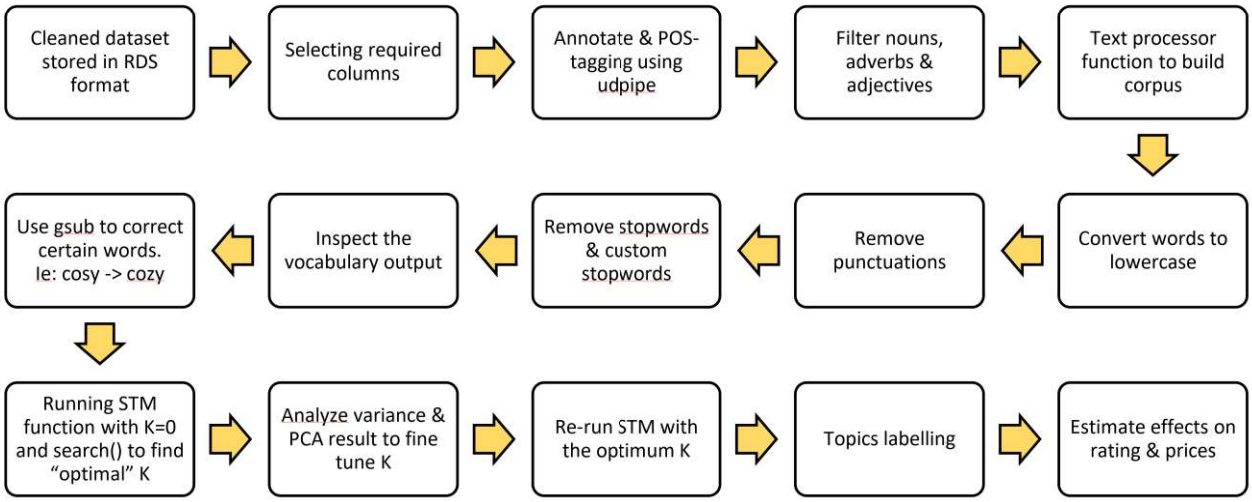
	Multiple R square	Adjusted R square
Model After Step AIC	0.3748	0.3675
Model after removing insignificant variables	0.3706	0.3649
Model with bing dictionary	0.7029	0.6976
Model with Nrc dictionary	0.6599	0.6539
Model with Affin dictionary	0.5415	0.5334
Model with Polarity of Description	0.4658	0.4564
Model of bing dictionary with only significant	0.7012	0.6979

To improve the predictability of regression, sub rating scores of rating score, such as review scores of check-in, are input into *variable_rating* and apply as independent variables. By following functionally and manually selection as first group of regression, the overall level of adjusted R^2 is significantly higher than the first group of regression. As a result, regression with score of bing liu dictionary still the most efficient one to improve the regression, that its adjusted R^2 is 0.814. By manually removing insignificant variables. the final regression of part B contains various independent variables with adjusted R^2 of 0.8137.

	Multiple R-square	Adjusted R square
Model After Step AIC	0.7856	0.7804
Model after removing insignificant variables	0.7818	0.778
Model with bing dictionary	0.8174	0.814
Model with Nrc dictionary	0.8109	0.8074
Model with Affin dictionary	0.7914	0.7875
Model with Polarity of Description	0.7819	0.7778
Model with Bing dictionary and significant variables removed	0.8169	0.8137

	Estimate	*Std. Error*	*t value*	*Pr(>|t|)*
(Intercept)	32.3181	2.7848	11.605	< 2e-16
avg_wordcount	0.2191	0.0772	2.836	0.0046
avg_syllable	-0.1569	0.0556	-2.821	0.0049
formality_comments	-0.0840	0.0254	-3.303	0.0010
avg_host_name	0.6678	0.2799	2.386	0.0172
hot_water	0.4028	0.1877	2.146	0.0321
private_entrance	0.4164	0.1426	2.919	0.0035
refrigerator	-0.3929	0.1702	-2.308	0.0211
shampoo	0.3433	0.1534	2.238	0.0254
patio_or_balcony	0.4267	0.1577	2.706	0.0069
dishwasher	0.5004	0.1476	3.390	0.0007
host_is_superhost	1.0915	0.1572	6.941	6.73e-12
avg_capital	-0.0018	0.0005	-3.581	0.0003
avg_count_excamation	0.0177	0.0039	4.467	8.75e-06
review_scores_accuracy	2.5018	0.1922	13.016	< 2e-16
review_scores_checkin	0.6895	0.1949	3.537	0.000422
review_scores_communication	0.5382	0.2070	2.599	0.009472
review_scores_value	1.4948	0.1581	9.453	< 2e-16
review_scores_location	1.0067	0.1392	7.229	9.21e-13
bing_liu_sentiment	8.2675	0.5689	14.531	< 2e-16
	R2	0.8169	*Adjusted R2*	0.8137

Part C



Part C: Topic Modelling and Latent Dirichlet Allocation

In this part, we will investigate topic prevalence in the Manchester Airbnb review corpus and discover words that contribute most to each topic. In addition, we will perform various steps to find the optimal number of topics and the additive predictability that some topics add on estimating the renting price and rating score.

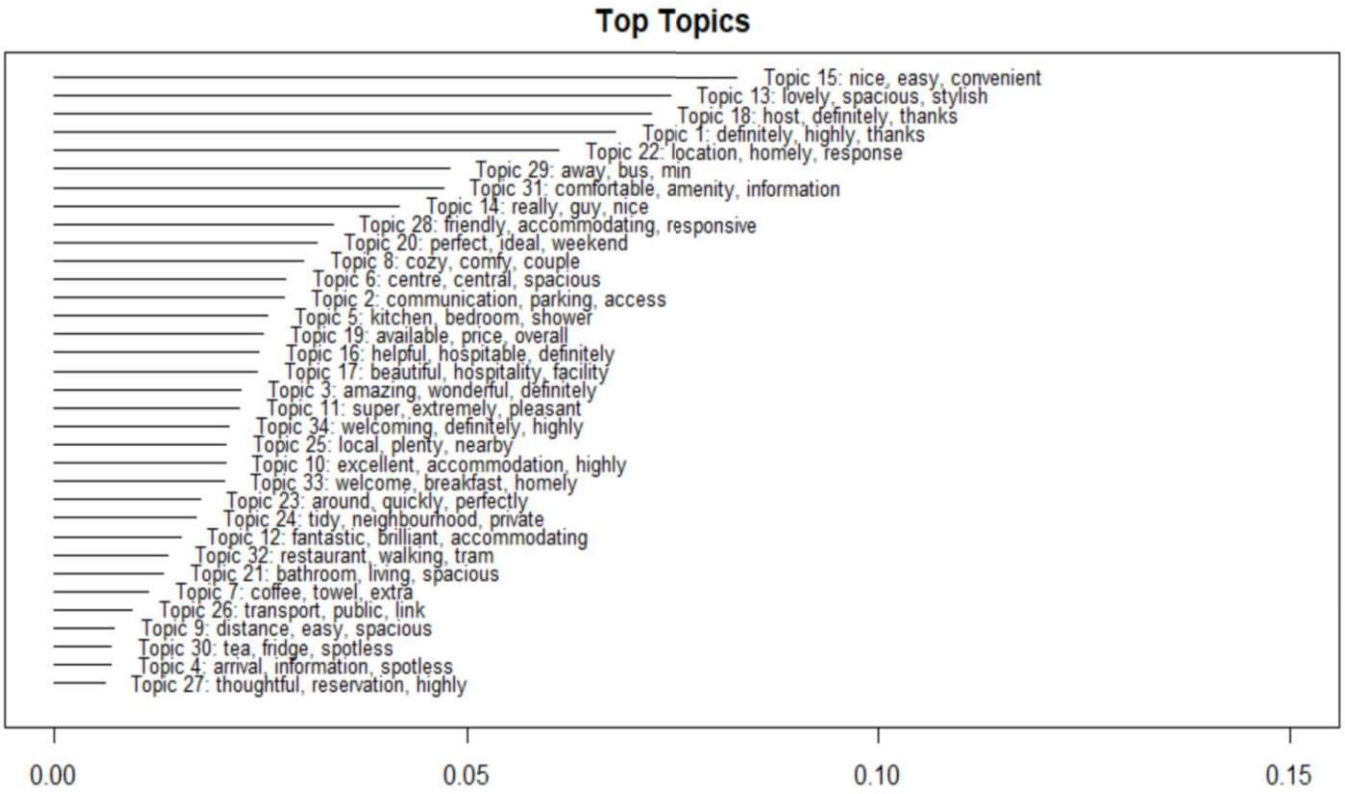
Before getting started on topic modelling, we performed POS tagging, which was a straightforward process with udpipe. The POS tagging process aims to add value to the topic modelling analysis by classifying nouns, adjectives, and adverbs. This helps with the interpretability of topics, wherein nouns capture aspects and adverbs, and adjectives capture sentiment. Once the corpus has been prepared for structural modelling, we can follow various approaches highlighted below to determine the optimal number of topic models.

Optimal Number of Topics

We have used the following approaches to find the optimal number of K:

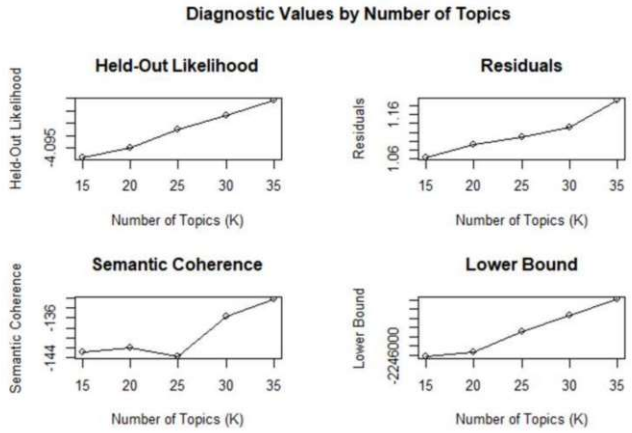
1. Set K = 0

Setting K = 0 and initialization type as "Spectral" in the stm function as suggested by Lee and Mimno (2014) ran multiple iterations and determined the appropriate number of topics to be 34. The function does not result in an optimal number of models but offers a good starting point to explore further. We will use the searchK function to explore topic numbers around 34 and decide on the final number of topics to go with based on measures such as exclusivity, held-out likelihood, semantic coherence, and residuals. Let us take a quick look at the expected topic proportions, to inspect the topics returned by the stm function.

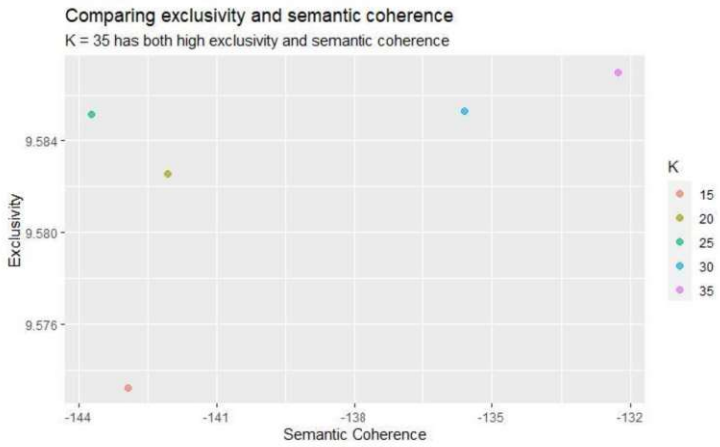


2. SearchK Function

The searchK function from the stm package ran hundreds of iterations with K = 15, 20, 25, 30, and 35. This was a computationally expensive process and had our RStudio Cloud crashing consistently. Our intention here was to use the input achieved from the previous step to search around K = 34 and to see whether K could be improved. The semantic coherence and the held-out likelihood are highest at 35, while the residuals are lowest until K = 15.

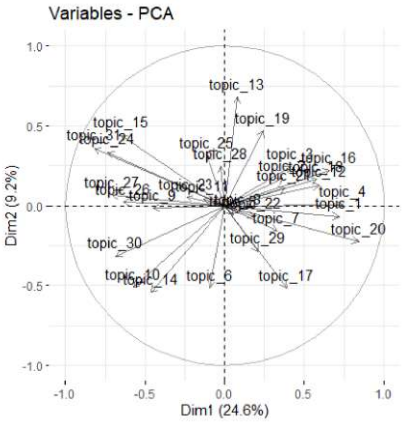


Even when comparing semantic coherence and exclusivity, K = 35 appears to be the best number of topic models.



3. Further Inspection and Tweaking

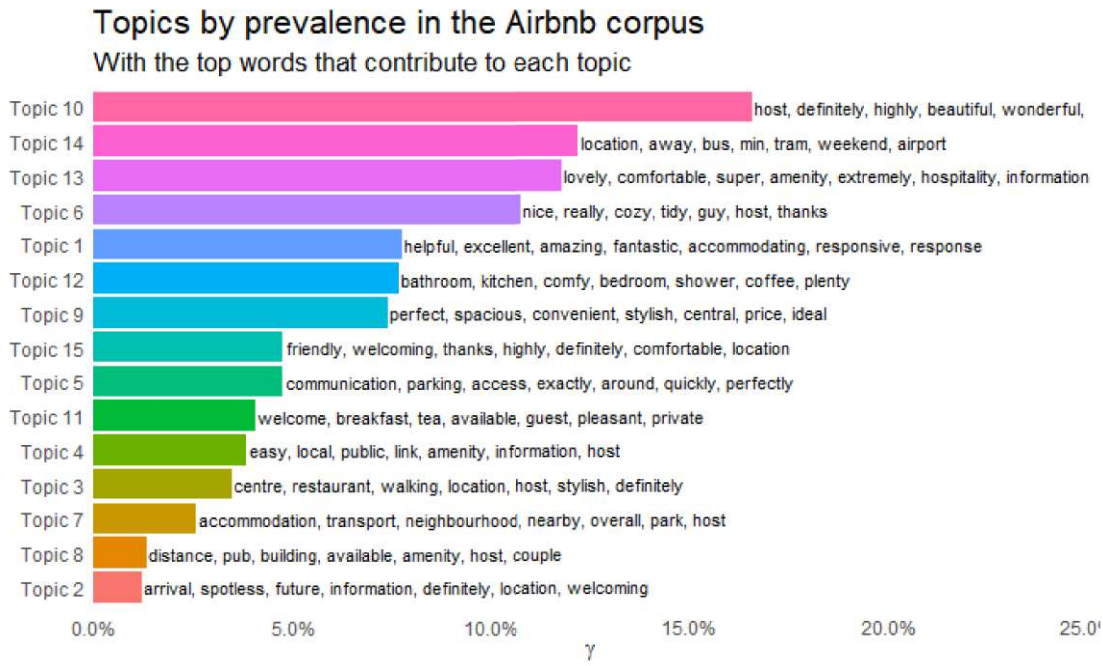
We will be exploring the topic models obtained through K = 34 in the steps above and evaluating them to further fine tune K. There is a significant overlap in the topic models obtained with K = 34, we can look at PCA to see how good the model is and confirm the overlaps. The model explains 35.8% of the variance as can be seen from.



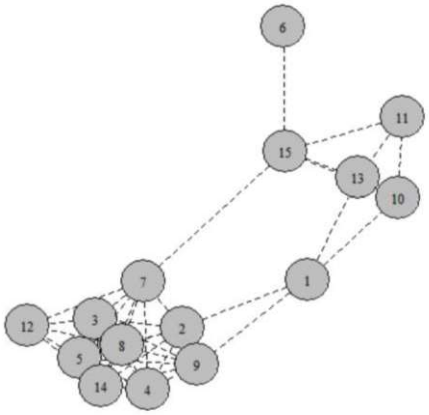
Next, we explored K = 15, 20, 25 and 30 and supplied these to the stm function in a descending order. Figure below summarises the results from the stm model.

K	Variance Explained
30	35.4%
25	36.6%
20	39.7%
15	44.0%

We decided to stop at K = 15, as the topics are distinct enough for us to label and use in estimating their effect on covariates of interest – rating score and price. It must be noted that this method is not perfect as it is a heuristic. The final topics and their top terms are shown.



We also explored the correlation among different topics, some topics are more correlated than others.

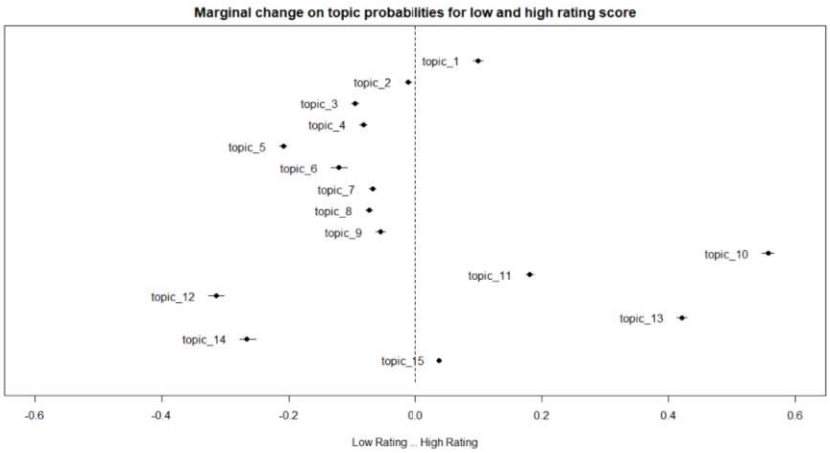


Examine Effect of Topics on Covariates of Interest

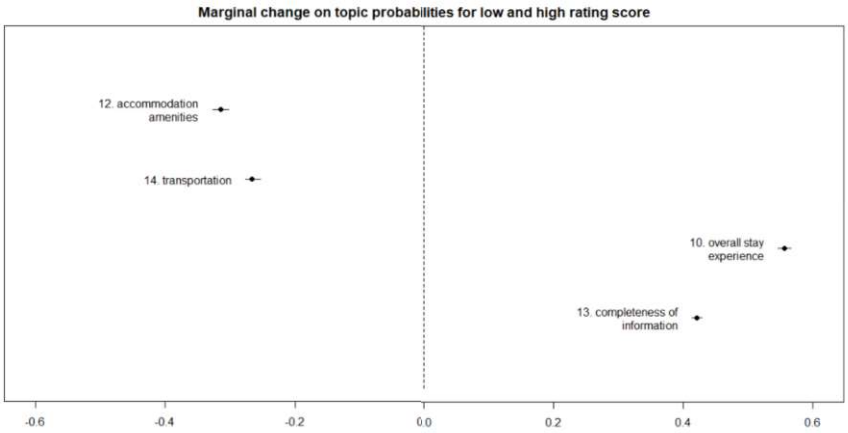
We will be using the estimateEffect function from the stm package to examine the relationship between various topics and covariates of interest. In this instance, the covariates of interest are rating score and price that the property lists for. Since we estimated the 'optimal' number (K = 15) of models in the previous section, we supply the same to the estimateEffect function and plot the effects.

Marginal Effects on Rating Score

Let us first explore marginal effects on rating score for all topics. Clearly, topics 12 and 14 lead to relatively lower ratings compared to topics 10 and 13, which lead to relatively higher ratings.



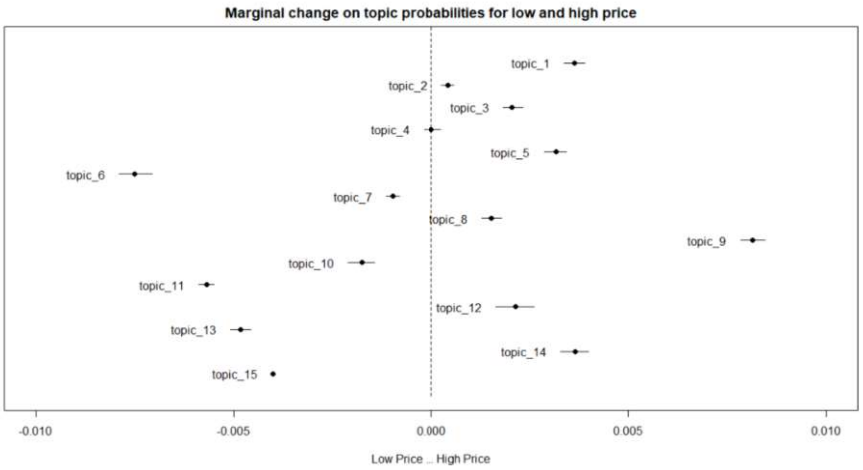
Labelling these topics using the top terms (beta) in each topic would shed further light on topics that are associated with relatively lower and relatively higher ratings.



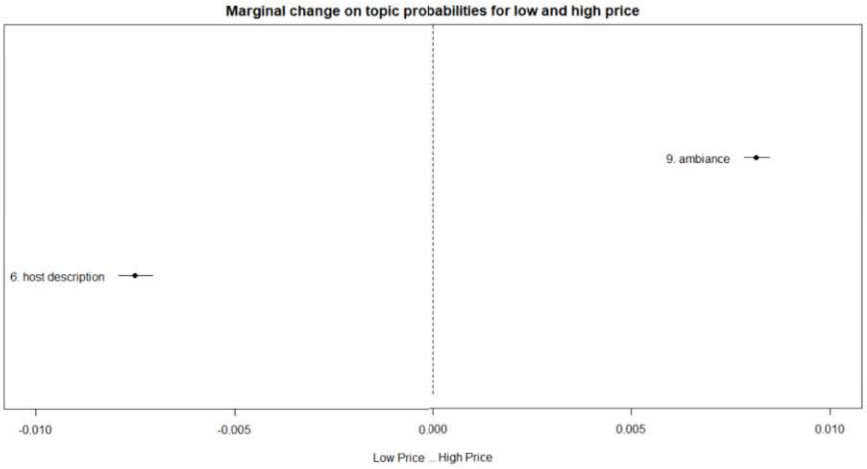
It is quite intuitive that a relatively high rating score would come from a good overall stay experience and accurate information provided by the host and that local transportation and accommodation amenities such as bathroom, kitchen, shower would not result in significantly high rating on their own.

Marginal Effects on Price

Let us first explore marginal effects on price for all topics. Clearly, topics 6 and 9 lead to relatively lower and higher price, respectively.



It can be seen from the figure below that ambiance has a relatively bigger impact on price.



References

Hadley Wickham (2021). rvest: Easily Harvest (Scrape) Web Pages. https://rvest.tidyverse.org/, https://github.com/tidyverse/rvest.

Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686

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/.

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>.

Silge, J., & Robinson, D. (2017). Text mining with R: A tidy approach. " O'Reilly Media, Inc.".

Jeroen Ooms (2021). cld3: Google's Compact Language Detector 3. https://docs.ropensci.org/cld3/, https://github.com/ropensci/cld3 (devel) https://github.com/google/cld3 (upstream).

Kurt Hornik (2020). NLP: Natural Language Processing Infrastructure. R package version 0.2-1.

Ingo Feinerer and Kurt Hornik (2020). tm: Text Mining Package. R package version 0.7-8. http://tm.r-forge.r-project.org/

Ingo Feinerer, Kurt Hornik, and David Meyer (2008). Text Mining Infrastructure in R. Journal of Statistical Software 25(5): 1-54. URL: https://www.jstatsoft.org/v25/i05/.

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

Hadley Wickham, Jim Hester and Winston Chang (2021). devtools: Tools to Make Developing R Packages Easier. https://devtools.r-lib.org/, https://github.com/r-lib/devtools.

Rinker, T. W. (2018). textclean: Text Cleaning Tools version 0.9.3. Buffalo, New York. https://github.com/trinker/textclean

Rinker, T. W. (2018). textstem: Tools for stemming and lemmatizing text version 0.1.4. Buffalo, New York. http://github.com/trinker/textstem

R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Jeroen Ooms (2020). hunspell: High-Performance Stemmer, Tokenizer, and Spell Checker. https://docs.ropensci.org/hunspell/ (docs), https://github.com/ropensci/hunspell (devel), https://hunspell.github.io (upstream).

Baptiste Auguie (2017). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.3.

David Robinson (2020). widyr: Widen, Process, then Re-Tidy Data. R package version 0.1.3. http://github.com/dgrtwo/widyr

Jan Wijffels (2020). udpipe: Tokenization, Part of Speech Tagging, Lemmatization and Dependency Parsing with the 'UDPipe' 'NLP' Toolkit. https://bnosac.github.io/udpipe/en/index.html, https://github.com/bnosac/udpipe.

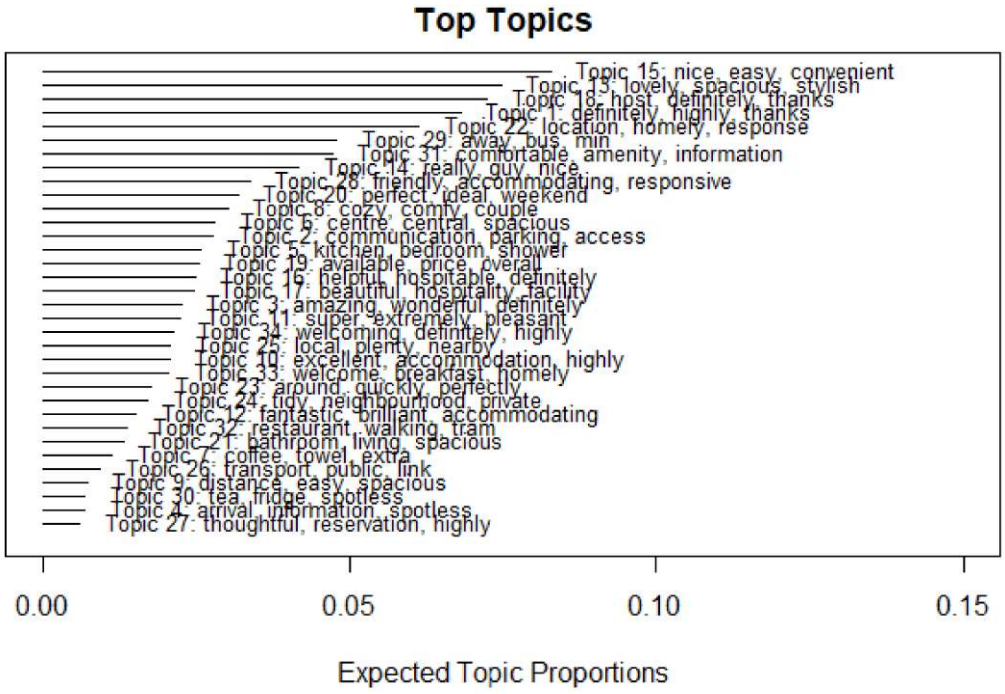
Emil Hvitfeldt (2020). textdata: Download and Load Various Text Datasets. R package version 0.4.1. https://github.com/EmilHvitfeldt/textdata

Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.

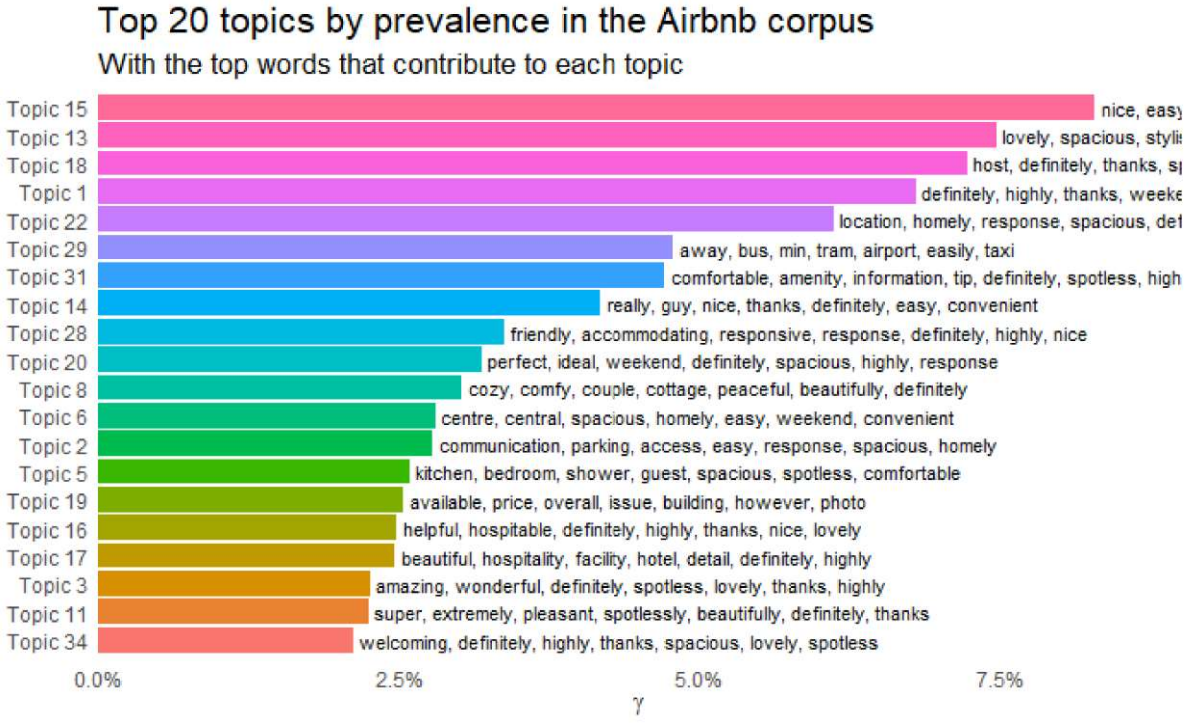
Appendix A - Figures

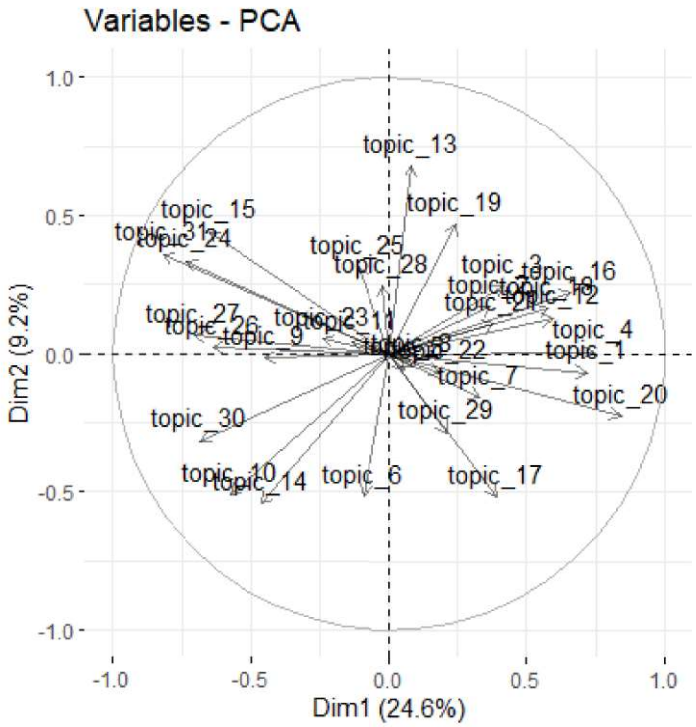
Part C

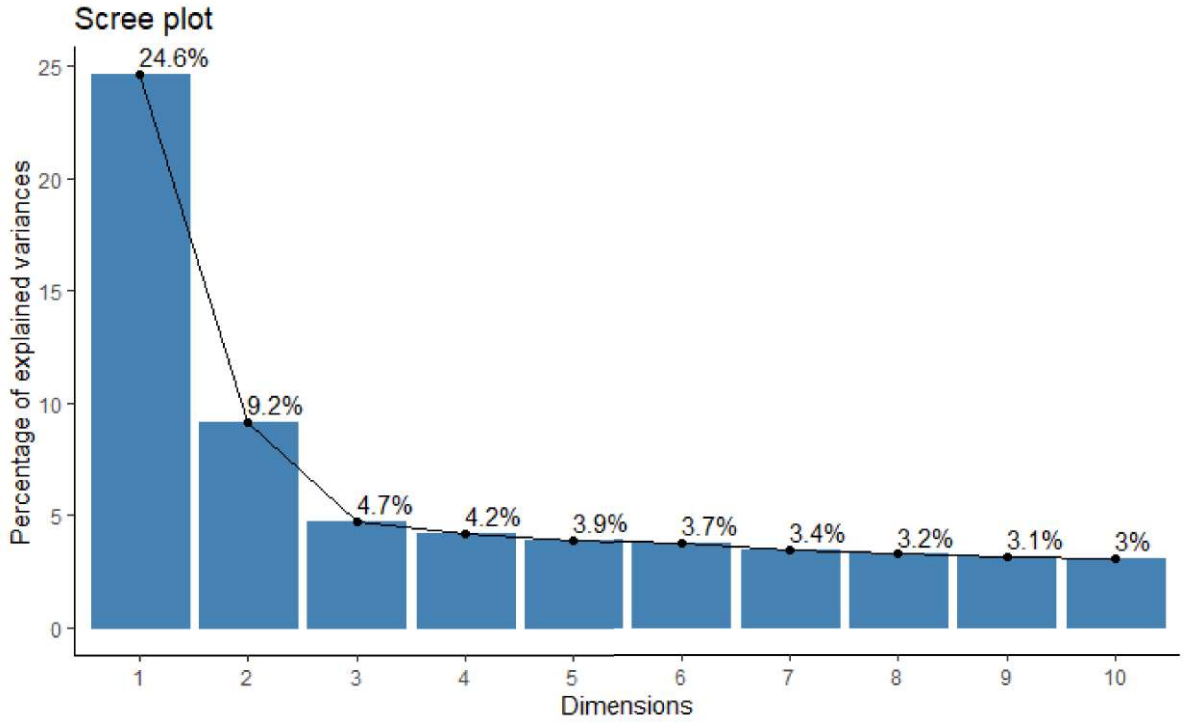
K = 0



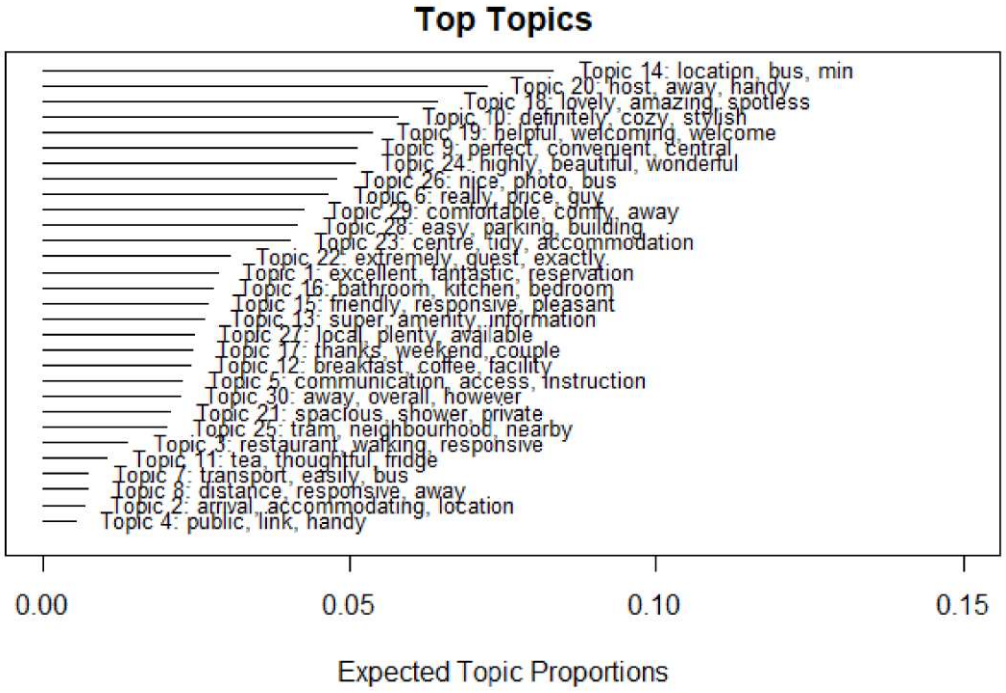

definitely
helpful
host
accommodating
airport
spotless
bathroom
thanks
amazing
neighbourhood
distance
absolutely
arrival
fantastic
parking
public
information
spacious
highly
super
homely
building
future
hospitable
local
facility
perfect
stylish
tip
cottage
peaceful
brilliant
away
overall
pub
transport
ideal
perfectly
responsive
weekend
however
coffee
nearby
tram
message
price
extra
hospitality
centre
bedroom
breakfast
excellent
shower
bus
extremely
easily
convenient
response
issue
exactly
tidy
guest
min
hotel
flexible
walking
quickly
available
tea
handy
instruction
around
communication
outside
amenity
easy
accommodation
restaurant
guy
plenty
comfy
welcome
cozy
pleasant
nice
kitchen
wonderful
couple
private
beautiful
location
central
towel
access
thoughtful
welcoming
comfortable
friendly

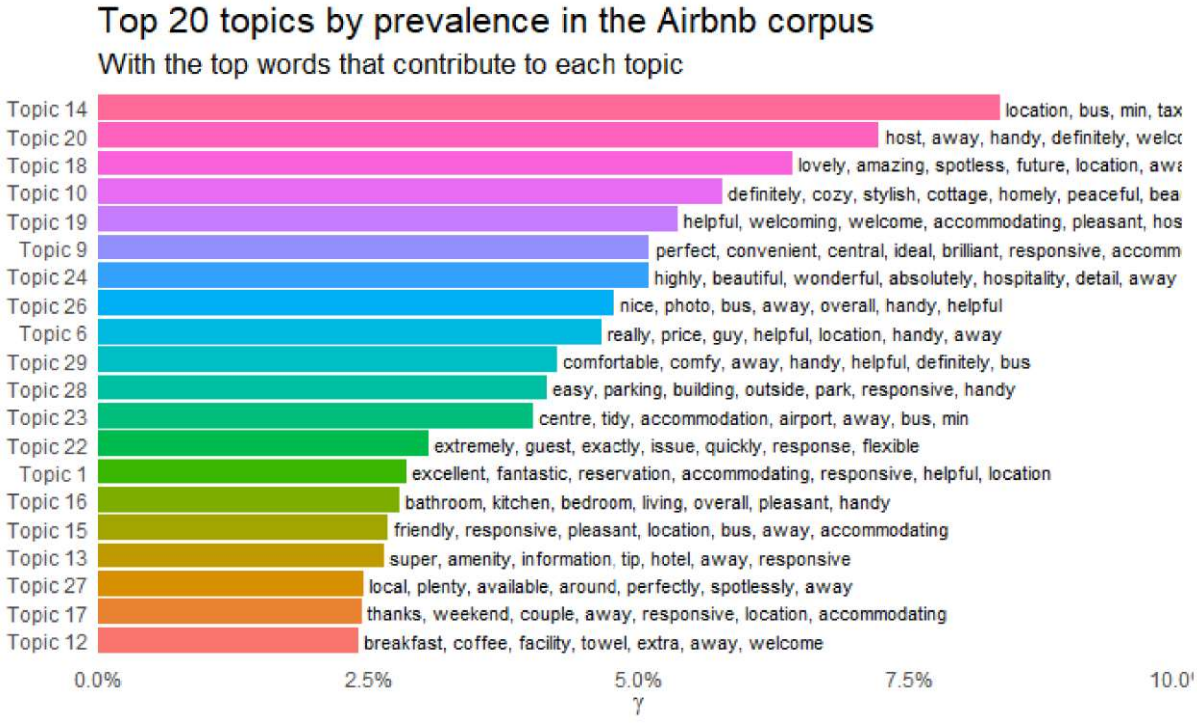

Top 20 topics by prevalence in the Airbnb corpus
With the top words that contribute to each topic
Topic 15
nice, easy
Topic 13
lovely, spacious, styli
Topic 18
host, definitely, thanks, sp
Topic 1
definitely, highly, thanks, weeke
Topic 22
location, homely, response, spacious, def
Topic 29
away, bus, min, tram, airport, easily, taxi
Topic 31
comfortable, amenity, information, tip, definitely, spotless, high
Topic 14
really, guy, nice, thanks, definitely, easy, convenient
Topic 28
friendly, accommodating, responsive, response, definitely, highly, nice
Topic 20
perfect, ideal, weekend, definitely, spacious, highly, response
Topic 8
cozy, comfy, couple, cottage, peaceful, beautifully, definitely
Topic 6
centre, central, spacious, homely, easy, weekend, convenient
Topic 2
communication, parking, access, easy, response, spacious, homely
Topic 5
kitchen, bedroom, shower, guest, spacious, spotless, comfortable
Topic 19
available, price, overall, issue, building, however, photo
Topic 16
helpful, hospitable, definitely, highly, thanks, nice, lovely
Topic 17
beautiful, hospitality, facility, hotel, detail, definitely, highly
Topic 3
amazing, wonderful, definitely, spotless, lovely, thanks, highly
Topic 11
super, extremely, pleasant, spotlessly, beautifully, definitely, thanks
Topic 34
welcoming, definitely, highly, thanks, spacious, lovely, spotless
0.0%
2.5%
5.0%
7.5%
γ

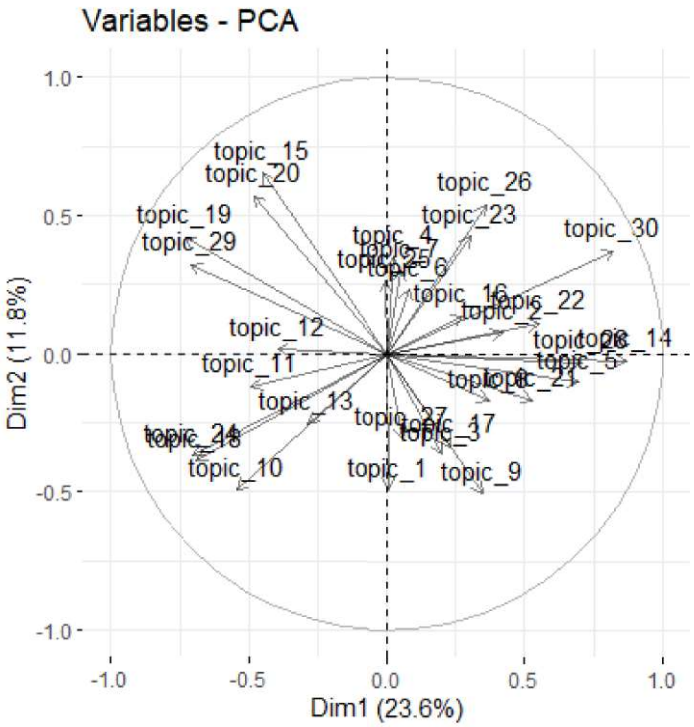

Variables - PCA
topic_13
topic_19
topic_15
topic_25
topic_28
topic_16
topic_4
topic_1
topic_7
topic_20
topic_29
topic_30
topic_10
topic_14
topic_6
topic_17
topic_22
Dim2 (9.2%)
Dim1 (24.6%)
1.0
0.5
0.0
-0.5
-1.0

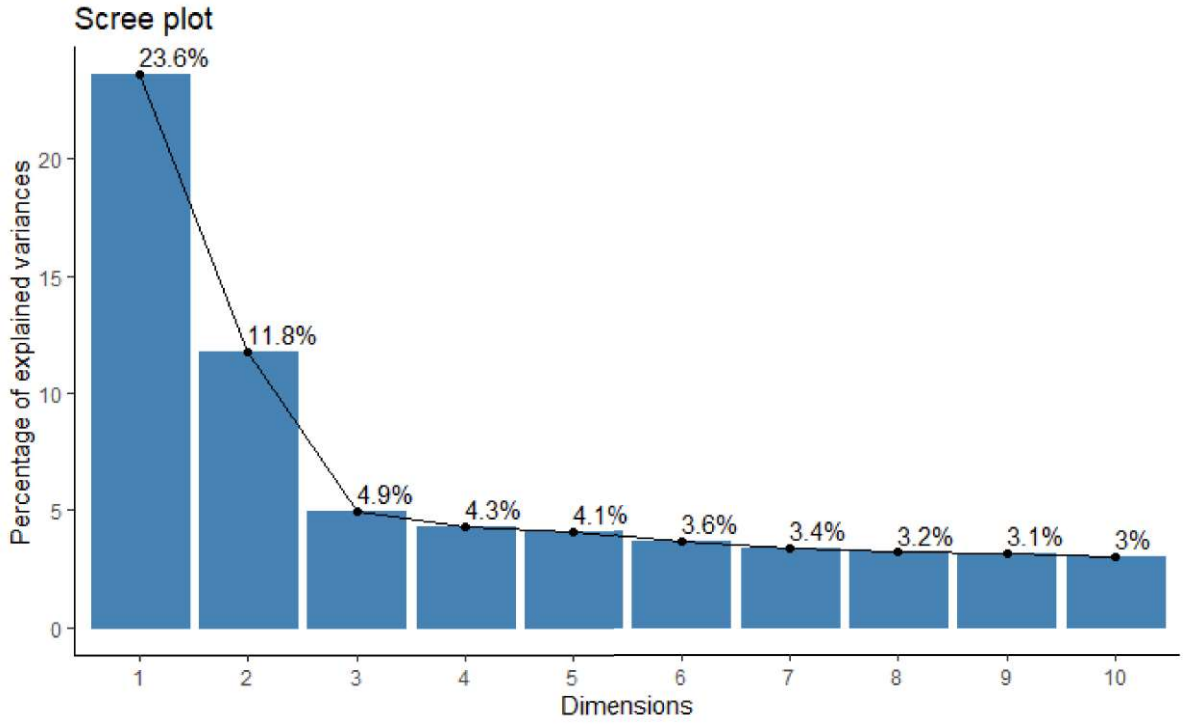

Scree plot
24.6%
9.2%
4.7%
4.2%
3.9%
3.7%
3.4%
3.2%
3.1%
3%
Percentage of explained variances
Dimensions
25
20
15
10
5
0
1
2
3
4
5
6
7
8
9
10

K = 30

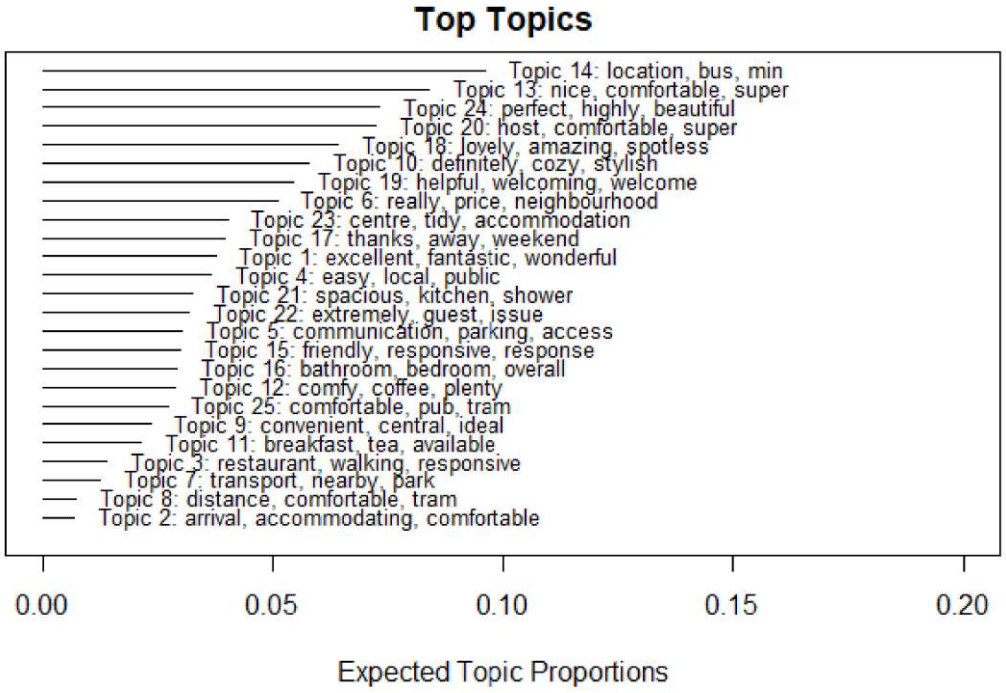

Top Topics
Topic 14: location, bus, min
Topic 20: host, away, handy
Topic 18: lovely, amazing, spotless
Topic 10: definitely, cozy, stylish
Topic 19: helpful, welcoming, welcome
Topic 9: perfect, convenient, central
Topic 24: highly, beautiful, wonderful
Topic 26: nice, photo, bus
Topic 6: really, price, guy
Topic 29: comfortable, comfy, away
Topic 28: easy, parking, building
Topic 23: centre, tidy, accommodation
Topic 22: extremely, guest, exactly
Topic 1: excellent, fantastic, reservation
Topic 16: bathroom, kitchen, bedroom
Topic 15: friendly, responsive, pleasant
Topic 13: super, amenity, information
Topic 27: local, plenty, available
Topic 17: thanks, weekend, couple
Topic 12: breakfast, coffee, facility
Topic 5: communication, access, instruction
Topic 30: away, overall, however
Topic 21: spacious, shower, private
Topic 25: tram, neighbourhood, nearby
Topic 3: restaurant, walking, responsive
Topic 11: tea, thoughtful, fridge
Topic 7: transport, easily, bus
Topic 8: distance, responsive, away
Topic 2: arrival, accommodating, location
Topic 4: public, link, handy
0.00
0.05
0.10
0.15
Expected Topic Proportions

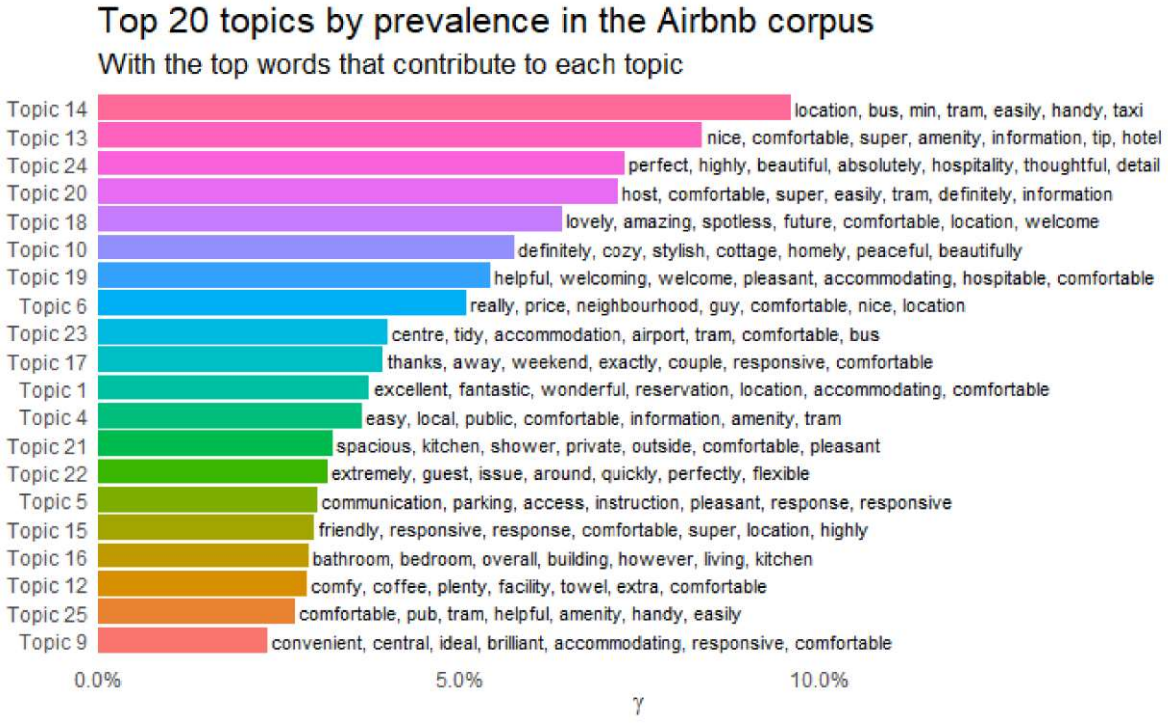

Top 20 topics by prevalence in the Airbnb corpus
With the top words that contribute to each topic
Topic 14 location, bus, min, tax
Topic 20 host, away, handy, definitely, welc
Topic 18 lovely, amazing, spotless, future, location, awa
Topic 10 definitely, cozy, stylish, cottage, homely, peaceful, bea
Topic 19 helpful, welcoming, welcome, accommodating, pleasant, hos
Topic 9 perfect, convenient, central, ideal, brilliant, responsive, accomm
Topic 24 highly, beautiful, wonderful, absolutely, hospitality, detail, away
Topic 26 nice, photo, bus, away, overall, handy, helpful
Topic 6 really, price, guy, helpful, location, handy, away
Topic 29 comfortable, comfy, away, handy, helpful, definitely, bus
Topic 28 easy, parking, building, outside, park, responsive, handy
Topic 23 centre, tidy, accommodation, airport, away, bus, min
Topic 22 extremely, guest, exactly, issue, quickly, response, flexible
Topic 1 excellent, fantastic, reservation, accommodating, responsive, helpful, location
Topic 16 bathroom, kitchen, bedroom, living, overall, pleasant, handy
Topic 15 friendly, responsive, pleasant, location, bus, away, accommodating
Topic 13 super, amenity, information, tip, hotel, away, responsive
Topic 27 local, plenty, available, around, perfectly, spotlessly, away
Topic 17 thanks, weekend, couple, away, responsive, location, accommodating
Topic 12 breakfast, coffee, facility, towel, extra, away, welcome
0.0%
2.5%
5.0%
7.5%
10.0
γ

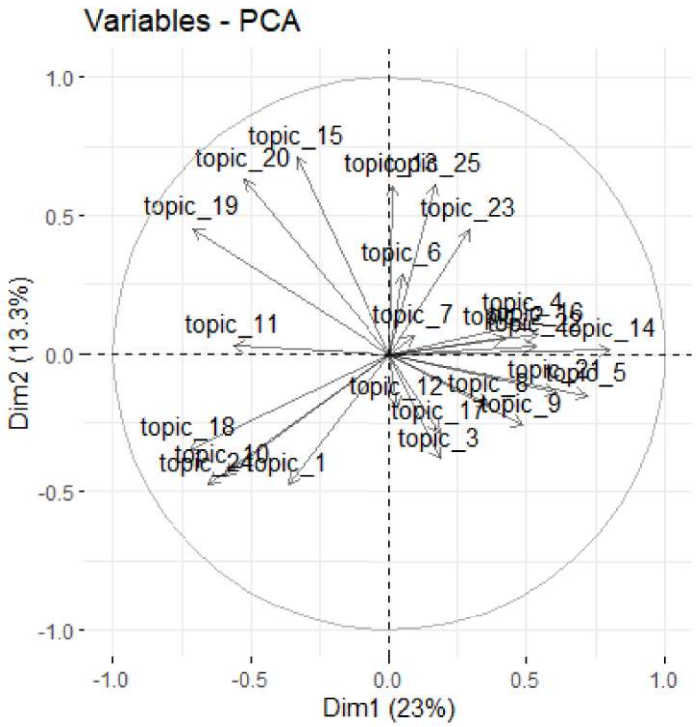

Variables - PCA
Dim2 (11.8%)
Dim1 (23.6%)
1.0
0.5
0.0
-0.5
-1.0
topic_15
topic_20
topic_26
topic_19
topic_23
topic_30
topic_29
topic_4
topic_16
topic_22
topic_12
topic_14
topic_11
topic_5
topic_13
topic_17
topic_3
topic_10
topic_1
topic_9

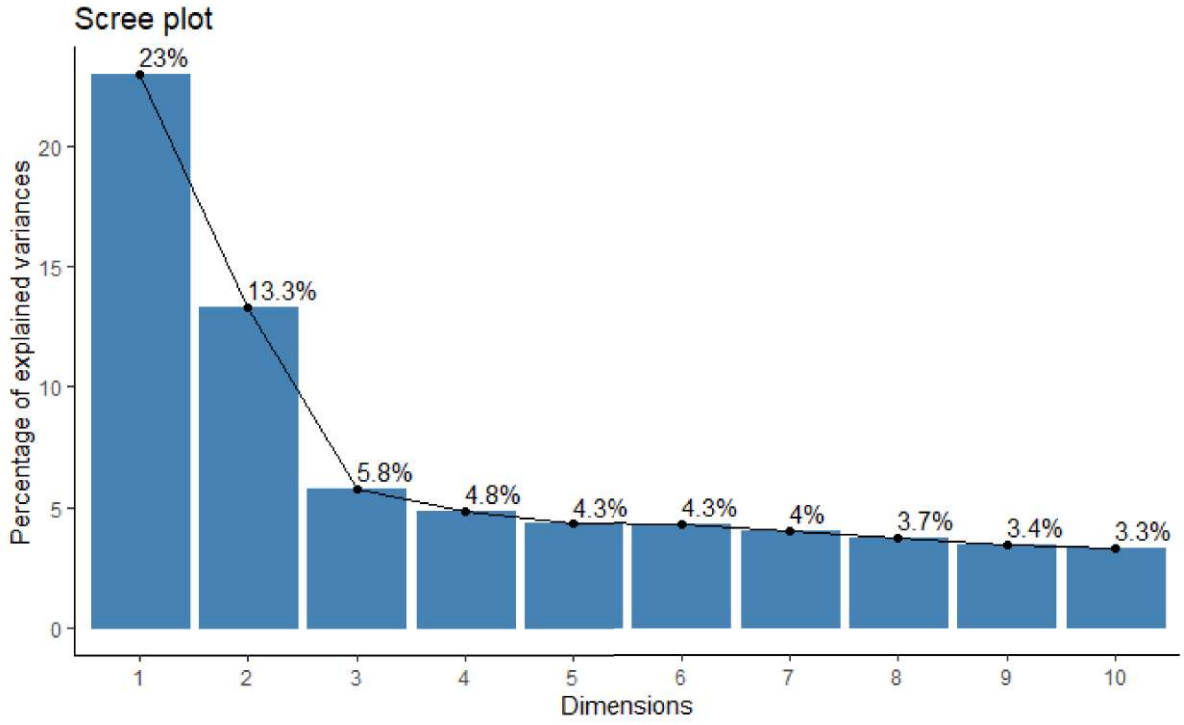

Scree plot
Percentage of explained variances
Dimensions
23.6%
11.8%
4.9%
4.3%
4.1%
3.6%
3.4%
3.2%
3.1%
3%
0
5
10
15
20
1
2
3
4
5
6
7
8
9
10

K = 25

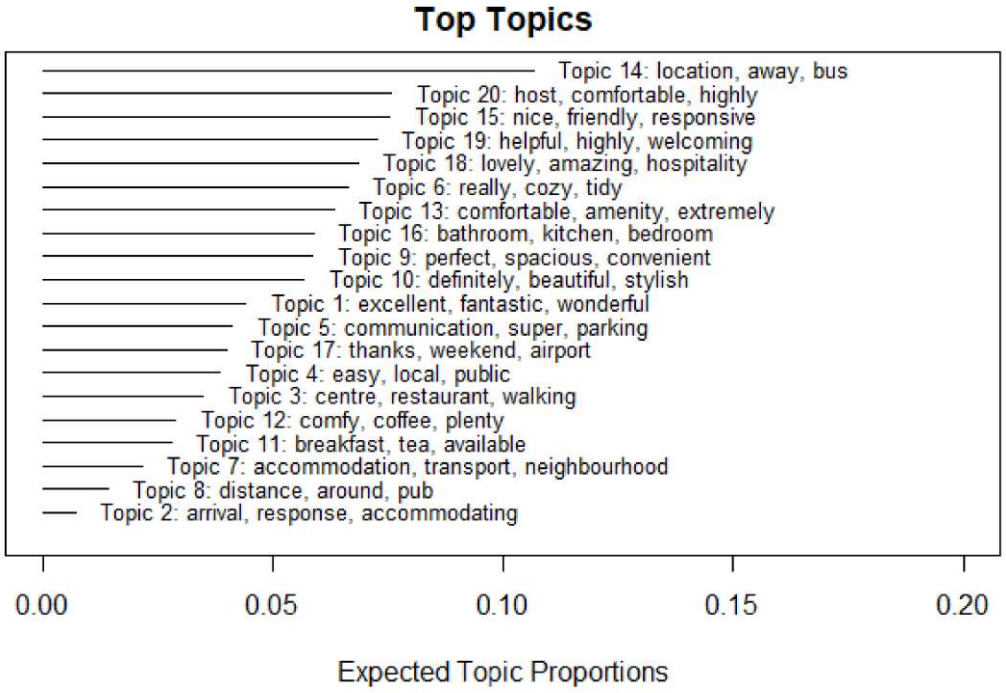

Top Topics
Topic 14: location, bus, min
Topic 13: nice, comfortable, super
Topic 24: perfect, highly, beautiful
Topic 20: host, comfortable, super
Topic 18: lovely, amazing, spotless
Topic 10: definitely, cozy, stylish
Topic 19: helpful, welcoming, welcome
Topic 6: really, price, neighbourhood
Topic 23: centre, tidy, accommodation
Topic 17: thanks, away, weekend
Topic 1: excellent, fantastic, wonderful
Topic 4: easy, local, public
Topic 21: spacious, kitchen, shower
Topic 22: extremely, guest, issue
Topic 5: communication, parking, access
Topic 15: friendly, responsive, response
Topic 16: bathroom, bedroom, overall
Topic 12: comfy, coffee, plenty
Topic 25: comfortable, pub, tram
Topic 9: convenient, central, ideal
Topic 11: breakfast, tea, available
Topic 3: restaurant, walking, responsive
Topic 7: transport, nearby, park
Topic 8: distance, comfortable, tram
Topic 2: arrival, accommodating, comfortable
0.00
0.05
0.10
0.15
0.20
Expected Topic Proportions

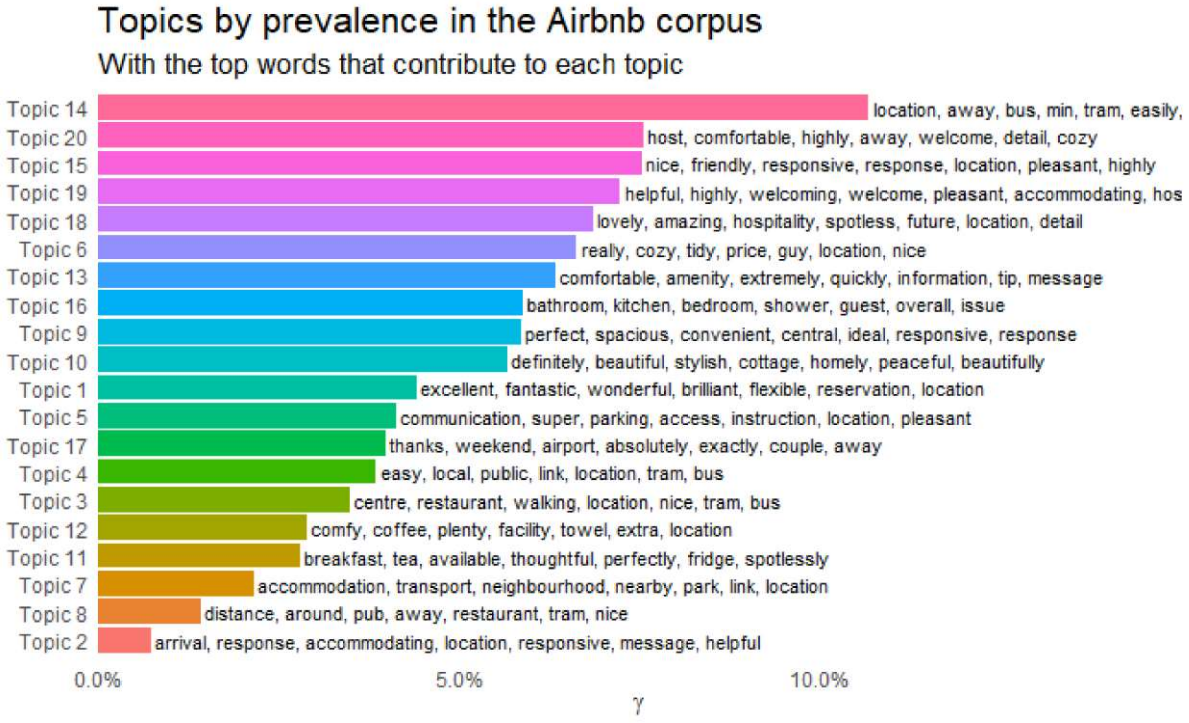

Top 20 topics by prevalence in the Airbnb corpus
With the top words that contribute to each topic
Topic 14 location, bus, min, tram, easily, handy, taxi
Topic 13 nice, comfortable, super, amenity, information, tip, hotel
Topic 24 perfect, highly, beautiful, absolutely, hospitality, thoughtful, detail
Topic 20 host, comfortable, super, easily, tram, definitely, information
Topic 18 lovely, amazing, spotless, future, comfortable, location, welcome
Topic 10 definitely, cozy, stylish, cottage, homely, peaceful, beautifully
Topic 19 helpful, welcoming, welcome, pleasant, accommodating, hospitable, comfortable
Topic 6 really, price, neighbourhood, guy, comfortable, nice, location
Topic 23 centre, tidy, accommodation, airport, tram, comfortable, bus
Topic 17 thanks, away, weekend, exactly, couple, responsive, comfortable
Topic 1 excellent, fantastic, wonderful, reservation, location, accommodating, comfortable
Topic 4 easy, local, public, comfortable, information, amenity, tram
Topic 21 spacious, kitchen, shower, private, outside, comfortable, pleasant
Topic 22 extremely, guest, issue, around, quickly, perfectly, flexible
Topic 5 communication, parking, access, instruction, pleasant, response, responsive
Topic 15 friendly, responsive, response, comfortable, super, location, highly
Topic 16 bathroom, bedroom, overall, building, however, living, kitchen
Topic 12 comfy, coffee, plenty, facility, towel, extra, comfortable
Topic 25 comfortable, pub, tram, helpful, amenity, handy, easily
Topic 9 convenient, central, ideal, brilliant, accommodating, responsive, comfortable
0.0%
5.0%
10.0%
γ

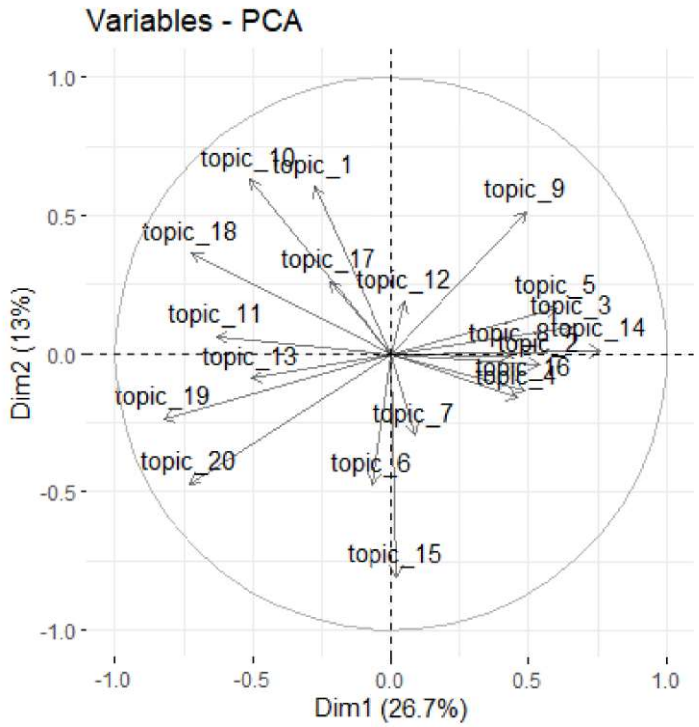

Variables - PCA
topic_15
topic_20
topic_25
topic_19
topic_23
topic_6
topic_11
topic_7
topic_14
topic_12
topic_9
topic_17
topic_3
topic_18
topic_1
Dim2 (13.3%)
Dim1 (23%)
1.0
0.5
0.0
-0.5
-1.0

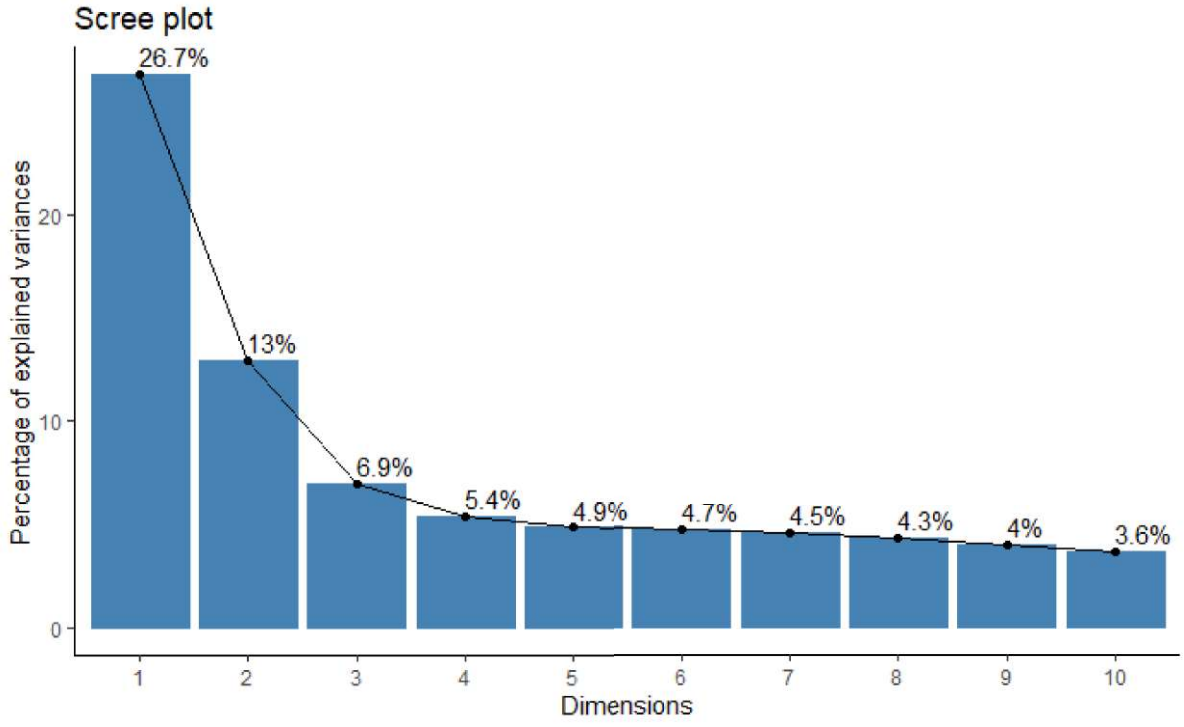

Scree plot
23%
13.3%
5.8%
4.8%
4.3%
4.3%
4%
3.7%
3.4%
3.3%
Percentage of explained variances
Dimensions
0
5
10
15
20
1
2
3
4
5
6
7
8
9
10

K = 20

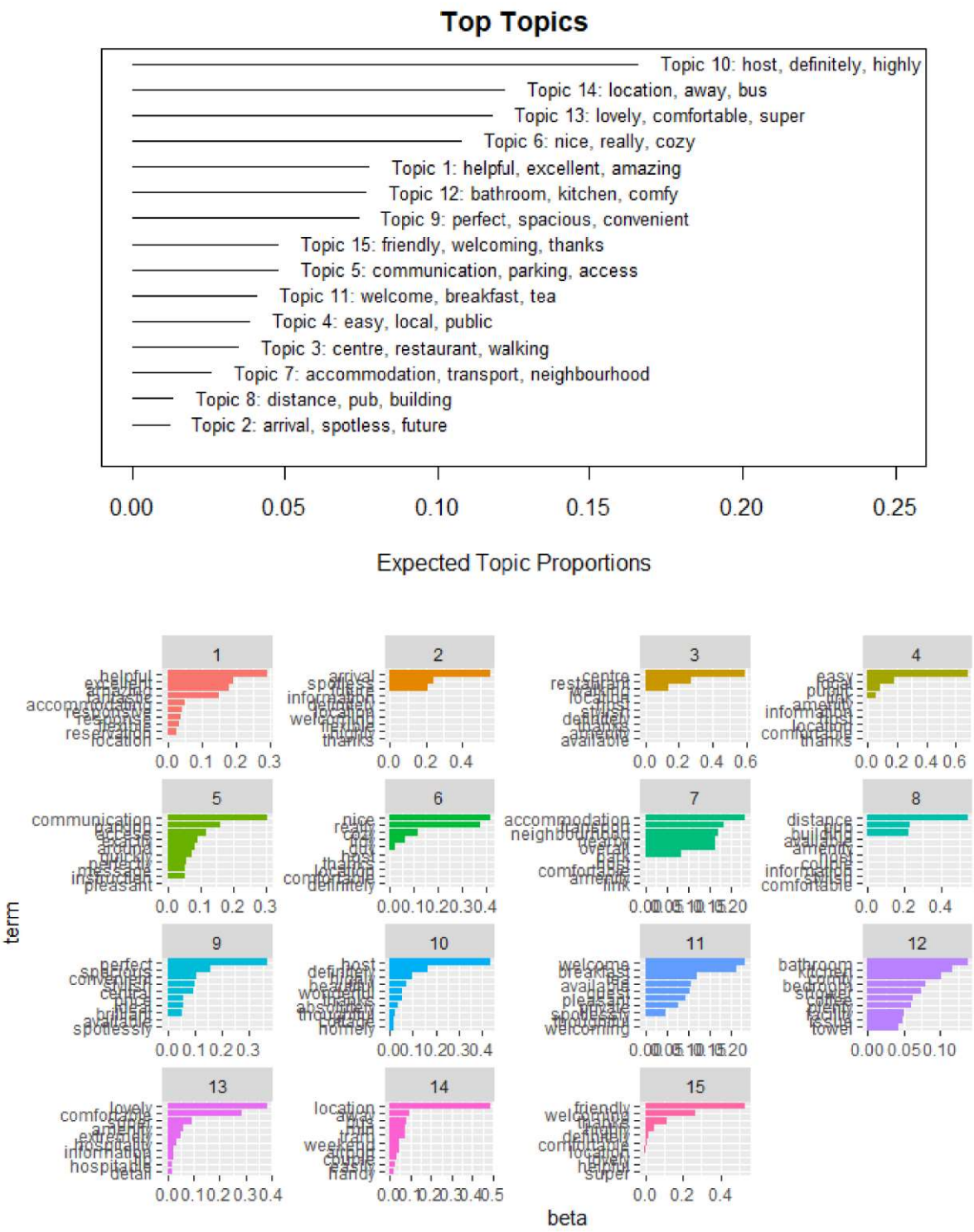

Top Topics
Topic 14: location, away, bus
Topic 20: host, comfortable, highly
Topic 15: nice, friendly, responsive
Topic 19: helpful, highly, welcoming
Topic 18: lovely, amazing, hospitality
Topic 6: really, cozy, tidy
Topic 13: comfortable, amenity, extremely
Topic 16: bathroom, kitchen, bedroom
Topic 9: perfect, spacious, convenient
Topic 10: definitely, beautiful, stylish
Topic 1: excellent, fantastic, wonderful
Topic 5: communication, super, parking
Topic 17: thanks, weekend, airport
Topic 4: easy, local, public
Topic 3: centre, restaurant, walking
Topic 12: comfy, coffee, plenty
Topic 11: breakfast, tea, available
Topic 7: accommodation, transport, neighbourhood
Topic 8: distance, around, pub
Topic 2: arrival, response, accommodating
0.00
0.05
0.10
0.15
0.20
Expected Topic Proportions

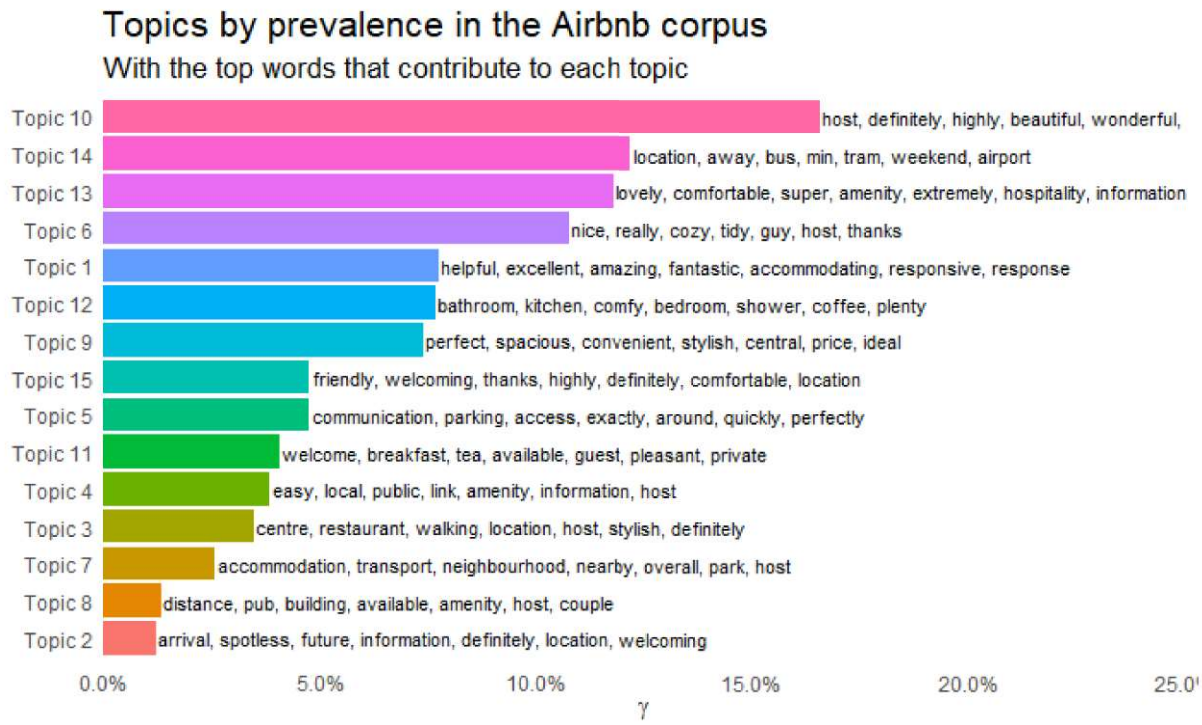

Topics by prevalence in the Airbnb corpus
With the top words that contribute to each topic
Topic 14 location, away, bus, min, tram, easily,
Topic 20 host, comfortable, highly, away, welcome, detail, cozy
Topic 15 nice, friendly, responsive, response, location, pleasant, highly
Topic 19 helpful, highly, welcoming, welcome, pleasant, accommodating, hos
Topic 18 lovely, amazing, hospitality, spotless, future, location, detail
Topic 6 really, cozy, tidy, price, guy, location, nice
Topic 13 comfortable, amenity, extremely, quickly, information, tip, message
Topic 16 bathroom, kitchen, bedroom, shower, guest, overall, issue
Topic 9 perfect, spacious, convenient, central, ideal, responsive, response
Topic 10 definitely, beautiful, stylish, cottage, homely, peaceful, beautifully
Topic 1 excellent, fantastic, wonderful, brilliant, flexible, reservation, location
Topic 5 communication, super, parking, access, instruction, location, pleasant
Topic 17 thanks, weekend, airport, absolutely, exactly, couple, away
Topic 4 easy, local, public, link, location, tram, bus
Topic 3 centre, restaurant, walking, location, nice, tram, bus
Topic 12 comfy, coffee, plenty, facility, towel, extra, location
Topic 11 breakfast, tea, available, thoughtful, perfectly, fridge, spotlessly
Topic 7 accommodation, transport, neighbourhood, nearby, park, link, location
Topic 8 distance, around, pub, away, restaurant, tram, nice
Topic 2 arrival, response, accommodating, location, responsive, message, helpful
0.0%
5.0%
10.0%
γ

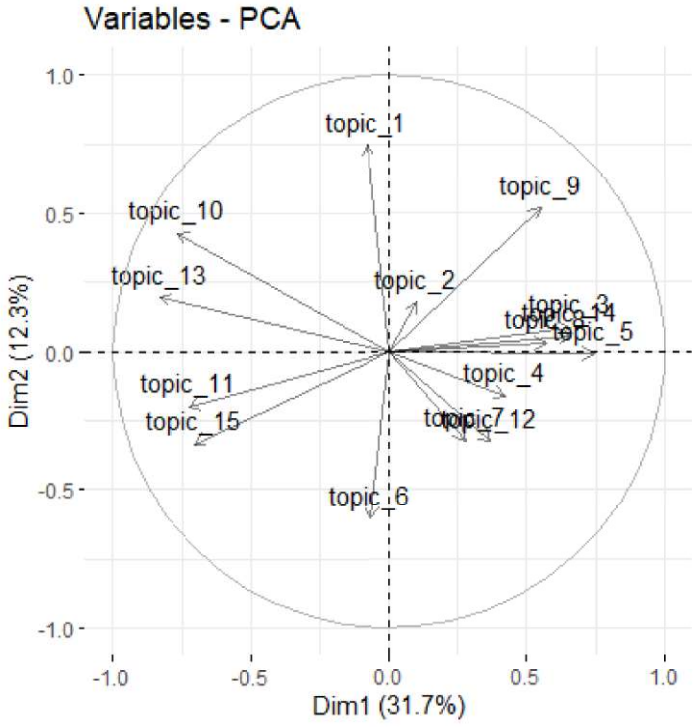

Variables - PCA
topic_10
topic_1
topic_9
topic_18
topic_17
topic_12
topic_5
topic_3
topic_11
topic_14
topic_13
topic_19
topic_7
topic_20
topic_6
topic_15
Dim2 (13%)
Dim1 (26.7%)
1.0
0.5
0.0
-0.5
-1.0

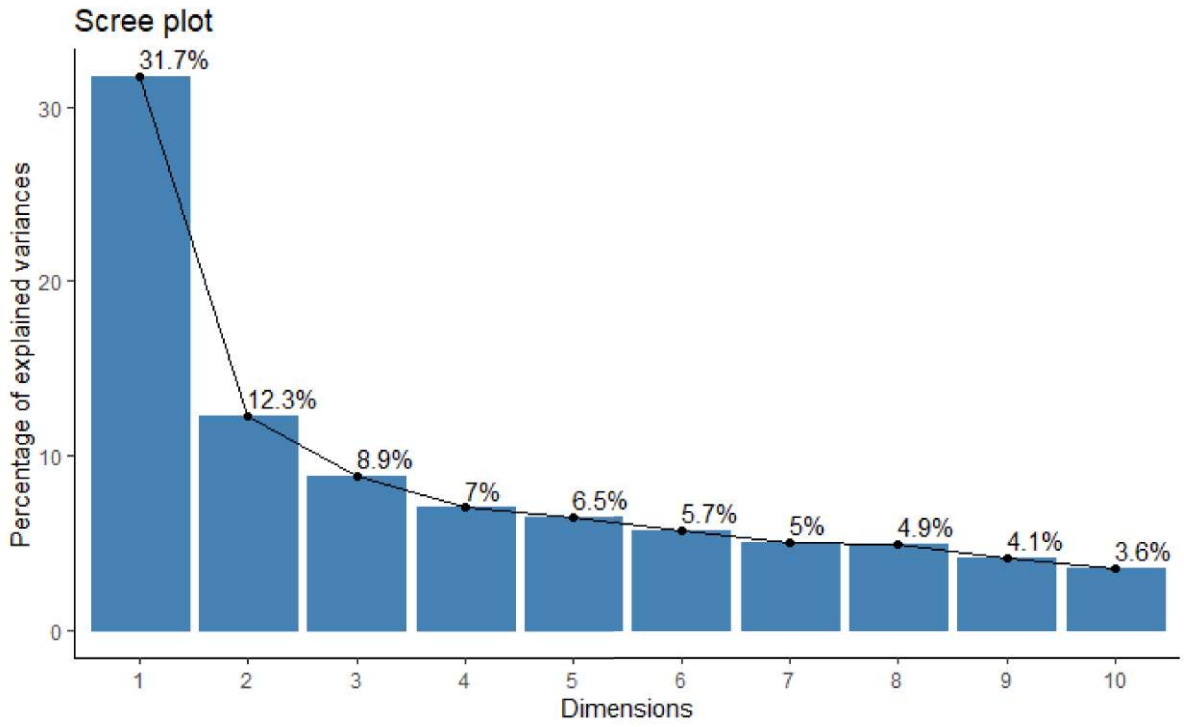

Scree plot
26.7%
13%
6.9%
5.4%
4.9%
4.7%
4.5%
4.3%
4%
3.6%
Percentage of explained variances
Dimensions
0
10
20
1
2
3
4
5
6
7
8
9
10

K = 15


Top Topics
Topic 10: host, definitely, highly
Topic 14: location, away, bus
Topic 13: lovely, comfortable, super
Topic 6: nice, really, cozy
Topic 1: helpful, excellent, amazing
Topic 12: bathroom, kitchen, comfy
Topic 9: perfect, spacious, convenient
Topic 15: friendly, welcoming, thanks
Topic 5: communication, parking, access
Topic 11: welcome, breakfast, tea
Topic 4: easy, local, public
Topic 3: centre, restaurant, walking
Topic 7: accommodation, transport, neighbourhood
Topic 8: distance, pub, building
Topic 2: arrival, spotless, future
0.00
0.05
0.10
0.15
0.20
0.25
Expected Topic Proportions
term
beta


Topics by prevalence in the Airbnb corpus
With the top words that contribute to each topic
Topic 10
host, definitely, highly, beautiful, wonderful,
Topic 14
location, away, bus, min, tram, weekend, airport
Topic 13
lovely, comfortable, super, amenity, extremely, hospitality, information
Topic 6
nice, really, cozy, tidy, guy, host, thanks
Topic 1
helpful, excellent, amazing, fantastic, accommodating, responsive, response
Topic 12
bathroom, kitchen, comfy, bedroom, shower, coffee, plenty
Topic 9
perfect, spacious, convenient, stylish, central, price, ideal
Topic 15
friendly, welcoming, thanks, highly, definitely, comfortable, location
Topic 5
communication, parking, access, exactly, around, quickly, perfectly
Topic 11
welcome, breakfast, tea, available, guest, pleasant, private
Topic 4
easy, local, public, link, amenity, information, host
Topic 3
centre, restaurant, walking, location, host, stylish, definitely
Topic 7
accommodation, transport, neighbourhood, nearby, overall, park, host
Topic 8
distance, pub, building, available, amenity, host, couple
Topic 2
arrival, spotless, future, information, definitely, location, welcoming
0.0%
5.0%
10.0%
15.0%
20.0%
γ


Variables - PCA
topic_1
topic_9
topic_10
topic_13
topic_2
topic_3
topic_14
topic_5
topic_4
topic_11
topic_15
topic_12
topic_6
1.0
0.5
0.0
-0.5
-1.0
Dim2 (12.3%)
Dim1 (31.7%)



Appendix B – R codes

Group Number 20

Student IDs: 2043274, 2090491, 2091680, 2092358, 2095515

Contents

```
rm(list=ls())
library(rvest) #For scrapping the data
library(tidyverse) #For data transformation
library(lubridate) #dealing with dates
library(tidytext) #For text analytical pipelines
library(cld3) #for filtering English reviews
library(NLP)
library(tm)
library(qdap) #For transformation of strings
#install.packages("devtools")
devtools::install_github("ropensci/genderdata")
library(textclean)# For replacing strings
library(textstem) #for lemmatization
library(parallel) # For speeding up process execution
library(hunspell) #for spelling correction
library(gridExtra) #for grid.arrange()
library(widyr) #pairwise correlation
library(udpipe) #for annotate
```

Data Preparation

We have selected the Manchester City from Airbnb insider data sets. We have selected the archived data of May 2020 because of the pandemic effect. Here we are scrapping the files related to listings details and reviews and filtered them for time period before Feburary 2020. The data is further filtered for only English descriptions and comments for further analysis.

Downloading the files for Manchester

```
#Setting URL of Airbnb insider
url <- "http://insideairbnb.com/get-the-data.html"

#Read HTML
pg_source <- read_html(url)

#Read all cities to a table
tables <- pg_source %>% html_nodes("table")

#Extracting the index number for Greater Manchester
city_index <- pg_source %>%
  html_nodes("h2") %>%
  html_text() %>%
  grep("Manchester", .) #provide city name to extract the index of the city

#Filter the details of the city through index number of the city
tables <- tables[[city_index]] %>%
  html_table()

#Correction in column names as per requirement
colnames(tables) <- gsub(" ","_",tolower(colnames(tables)))
colnames(tables) <- gsub("/","_",colnames(tables))

#Now getting links of data file so we can download them
```

```
links <- pg_source %>%
  html_node(".greater-manchester") %>%
  html_node("tbody") %>%
  html_nodes("tr") %>%
  #html_nodes("td") %>%
  html_node("a") %>%
  html_attr("href")

#Combining the dataset with links
tables$link <- links

#Filtering data for review and listings files through finding index by grep
tables <- tables[grep("listings.csv.gz|reviews.csv.gz", tables$file_name),]

#Casting date
tables$date_compiled <- as.Date(tables$date_compiled,format = "%d %b,%Y")

#Filtering data for May 2020 files as least data loss due to pandemic filtering
tables <- tables %>% filter(date_compiled == min(date_compiled))

#Downloading the file to local machine for quicker operations
#for (i in 1:nrow(tables)){
#download.file(url = tables$link[i], tables$file_name[i])
#}

#Remove tables and pg_source to keep environment space
rm(tables)
rm(pg_source)
```

Reading files into R

```
#Reading files to environment of review of Manchester
reviews_df <- read_csv(
  gzfile("reviews.csv.gz"))
#Review have 137240 observations and 6 variables


#Reading files to environment of listing of Manchester
listings_df <- read_csv(
  gzfile("listings.csv.gz"))
#Listing have 4924 observations and 106 variables
```

Combining listings and reviews into one dataframe

```
#filtering reviews after February 2020 due to COVID'19 pandemic
reviews_df <- reviews_df %>% filter(date > "01-02-2021")
#Observations 137240
```

Dealing with listings with few reviews

We will use listings with higher reviews so we can get more information for each listing. Low reviews listings can be baised or manipulative.

```r
#Lets filter data for number of reviews per listing
reviews_df %>% group_by(listing_id) %>% summarise(reviews_sum = n()) %>% arrange(desc(reviews_sum)) -> r

#Visualise the counts per listing
ggplot(reviews_count, aes(x = reviews_sum)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(title = "Counts per listing", x = "Count of Reviews", y = "Count of listing")

#Find the quartile to decide the threshold
quantile(reviews_count$reviews_sum)

#Visualise boxplot
ggplot(reviews_count, aes(reviews_sum)) +
  geom_boxplot() +
  labs(title = "Distribution of Reviews per Listing")


#Filtering reviews below 4
reviews_count <- reviews_count %>% filter(reviews_sum<4)
#Observations of reviews are 137240

#Filter the reviews below lower quantile as it will be tough to generalize of these listings
reviews_df <- reviews_df %>% filter(!(listing_id %in% (reviews_count %>% pull(listing_id))))

#Filtering for reviews in English
reviews_df <- reviews_df %>%
  mutate(review_language = cld3::detect_language(comments))%>%
  filter(review_language == "en")
#Observations are 123428 now

#Filtering for listing description in English
listings_df <- listings_df %>%
  mutate(description_language = cld3::detect_language(description))%>%
  filter(description_language == "en")
#Observations are 4666 now
```

Preparing review and description text for analysis

We have removed punctuation, made all to lower cases, useless symbols, extra space and numbers. We have created a different column as these few functions run on raw data for better output.

```r
#set the text to lowercase in review and listing dataframe
reviews_df$comments_new <- tolower(reviews_df$comments)
listings_df$description_new <- tolower(listings_df$description)

#Convert contractions to base words in review and listing dataframe
reviews_df$comments_new <- replace_contraction(reviews_df$comments_new)
listings_df$description_new <- replace_contraction(listings_df$description)

#Removing punctuation from review comments and listing description and transit
reviews_df$comments_new <- gsub('[[:punct:]]+',' ', reviews_df$comments_new)
listings_df$description_new <- gsub('[[:punct:]]+',' ', listings_df$description)

#replacing \r\n in comments in review
```

```
reviews_df$comments_new <- gsub('[\r\n]+', ' ', reviews_df$comments_new)

#Trim the data in review and listing dataframe
reviews_df$comments_new <- trimws(reviews_df$comments_new)
listings_df$description_new <- trimws(listings_df$description)

#Remove extra spaces in review and listing dataframe
reviews_df$comments_new <- stripWhitespace(reviews_df$comments_new)
listings_df$description_new <- stripWhitespace(listings_df$description)

#Replace numbers from comments, description, transit
reviews_df$comments_new <- replace_number(reviews_df$comments_new)
listings_df$description_new <- replace_number(listings_df$description)

#converting $ to numeric values in listing data frame
listings_df$price <- listings_df$price %>% str_extract_all("\\(?[0-9,.]+\\)?") %>% gsub(",", "", .) %>%
listings_df$security_deposit <- listings_df$security_deposit  %>% str_extract_all("\\(?[0-9,.]+\\)?") %
listings_df$cleaning_fee <- listings_df$cleaning_fee   %>% str_extract_all("\\(?[0-9,.]+\\)?") %>% gsub
listings_df$extra_people <- listings_df$extra_people   %>% str_extract_all("\\(?[0-9,.]+\\)?") %>% gsub

#Saving dataframe for efficiency
saveRDS(listings_df, "listing.rds")
saveRDS(reviews_df, "review.rds")
```

```
listings_df <- readRDS("listing.rds")
reviews_df <- readRDS("review.rds")
```

We will remove columns that do not add addtional information or aid our analysis:

```
#The variables which doesn't have effect on price or rating are been removed
listings_df <- listings_df %>%
  select(-c("listing_url", "scrape_id", "last_scraped","experiences_offered","thumbnail_url", "medium_u

#Removing columns with repetitive information
listings_df <- listings_df %>%
  select(-c("summary", "space")) #summary is covered by description and space is covered by multiple va
#Number of varaiables in listing 69
```

Dealing with NAs

Let's get the NA count per column

```
na_count <- data.frame(sapply(listings_df, function(y) sum(length(which(is.na(y))))))

#Let us now set the rownames as the column

na_count <- rownames_to_column(na_count, "column")

names(na_count)[2] <- "na_sum"

#filter for the columns with NA greater than 0
na_count <- na_count %>%
  filter(na_sum > 0) %>%
  arrange(desc(na_sum))
```

```
#Let's plot these values
ggplot(na_count, aes(na_sum, reorder(column, na_sum))) +
  geom_col(fill = "lightblue", color = "black") +
  scale_x_continuous(labels = scales::comma) +
  labs(title = "Empty Values in Each variable", x = "Empty Value Count", y = "Variable")

#Removing columns due to high NA observations
#Cannot replace such high number of NA as it will effect the analysis
#Generalization is not possible for these
listings_df <- listings_df %>%
  select(-c("neighbourhood", "jurisdiction_names","license", "square_feet", "host_neighbourhood", "montl

#Variables now are 55
```

Replacing and setting assumptions for NA values. We have assumed that if there is no value for security deposit, cleaning fee, extra people and minimum nights it will be zero.

```
#replacing NAs in the listings columns
listings_df$security_deposit[is.na(listings_df$security_deposit)] <- 0
listings_df$cleaning_fee[is.na(listings_df$cleaning_fee)] <- 0
listings_df$extra_people[is.na(listings_df$extra_people)] <- 0
listings_df$minimum_nights[is.na(listings_df$minimum_nights)] <- 0
```

Further analyzing NA values for betterment

```
na_count <- data.frame(sapply(listings_df, function(y) sum(length(which(is.na(y))))))

#Let us now set the rownames as the column

na_count <- rownames_to_column(na_count, "column")

names(na_count)[2] <- "na_sum"

#filter for the columns with NA greater than 0
na_count <- na_count %>%
  filter(na_sum > 0) %>%
  arrange(desc(na_sum))

#Let's plot these values
ggplot(na_count, aes(na_sum, reorder(column, na_sum))) +
  geom_col(fill = "lightblue", color = "black") +
  scale_x_continuous(labels = scales::comma) +
  labs(x = "Empty Value Count", y = "Variable")
```

Dealing with professional reviews

We will also be removing professional reviewers:

```
#Converting date format
reviews_df$date <- ymd(reviews_df$date)

#Extracting year
reviews_df$year <- year(reviews_df$date)

#Finding possible fraudulent reviewers
reviews_df %>% group_by(reviewer_id, listing_id, year) %>% count(reviewer_id, sort = T) %>% filter(n >5
```

```r
#Visualize the count of possible fraudulent reviews
ggplot(multiple_reviews, aes(x=n)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(title = "Possible fraudulent reviews", x = "Count of Reviews", y = "Count of Reviewers")

#Removing reviews of reviewers have more than 5 reviews for same accommodation within a year
#They can be fraudulant removing them is for data integrity

reviews_df <- reviews_df %>% filter(!(reviewer_id %in% (multiple_reviews %>% pull(reviewer_id))))

#Observations now are 122,260
```

Analyzing price variable for outliers and skewness

```r
#Lets visualize the data to identify outliers and distribution
listings_df %>%
  select(price) %>%
  group_by(price) %>%
  summarise(count = n()) %>%
  arrange(price) %>%
  ggplot(., aes(x = price)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(title = "Price Distribution", x = "Price", y = "Frequency") +
  geom_vline(xintercept = 1, color = "red") +
  geom_vline(xintercept = 2000, color = "red")


#Removing zero price and 2000 as after that there are discrete values
listings_df <- listings_df %>% filter(between(price, 1, 2000))

nrow(listings_df)

#There is evidence of skewness of price
e1071::skewness(listings_df$price)
#0.9565388 is the skewness which is too high so we have transformed

#Checking log and it's skewness
e1071::skewness(log(listings_df$price))
#-0.06549465 skewness modification

#Checking Square root and it's skewness modification
e1071::skewness(sqrt(listings_df$price))
#0.4520159 skewness modification

#Check if outlier effect the model
mod_1 <- lm(price~room_type, data = listings_df)

summary(mod_1)
plot(mod_1)

EnvStats::boxcox(mod_1)

bc <- MASS::boxcox(mod_1, lambda = seq(-3,3))
```

```
best_lambda <- bc$x[which(bc$y == max(bc$y))]
#Lambda Value -0.03030303

#Checking Box-Cox for skewness transformation
e1071::skewness((listings_df$price)^-0.03030303)
#0.06587 skewness modification


#Price has been transformed to normalise by log as it is most normalized transformation
listings_df$price_log <- log(listings_df$price)
```

Checking distribution of other numeric variables to see if there is skewness and if transformation improves results.

```
#Visualise transformed price
listings_df %>%
  select(price_log) %>%
  group_by(price_log) %>%
  summarise(count = n()) %>%
  arrange(price_log) %>%
  ggplot(., aes(x = price_log)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(title = "Transformed Price Distribution", x = "Logged Price", y = "Frequency")

#Visualise property type distribution
listings_df %>%
  select(property_type) %>%
  group_by(property_type) %>%
  summarise(count = n()) %>%
  ggplot(., aes(count,  reorder(property_type, count))) +
  geom_col(fill = "lightblue", color = "black") +
  labs(title = "Property Types", x = "Frequency", y = "")

#Visualise cancellation policy distribution
listings_df %>%
  select(cancellation_policy) %>%
  group_by(cancellation_policy) %>%
  summarise(count = n()) %>%
  ggplot(., aes(count,  reorder(cancellation_policy, count))) +
  geom_col(fill = "lightblue", color = "black") +
  labs(title = "Cancellation Policy", x = "Frequency", y = "")

#Visualise cleaning fee distribution
listings_df %>%
  select(cleaning_fee, room_type) %>%
  group_by(cleaning_fee, room_type) %>%
  summarise(count = n()) %>%
  arrange(cleaning_fee) %>%
  ggplot(., aes(x = cleaning_fee, color = "black")) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(title = "Cleaning Fee Distribution", x = "Cleaning Fee", y = "Frequency")
```

Checking improvement on regression model after pricing improvement

```r
#Checking model again for improvement
mod_2 <- lm(price~room_type, data = listings_df)

summary(mod_2)
#Room type is significant variable and there is huge improvement in model
#The R2 value have improved to 0.387

#Testing transformed price if that improves model performance
mod_3 <- lm(price_log~room_type, data = listings_df)

#Model summary
summary(mod_3)

#Comparing both models
anova(mod_2,mod_3)

#Transformed price model have a significant difference
#Transformed price gives Multiple R-squared:  0.3535,   Adjusted R-squared:  0.3531
```

```r
#Checking other variable outliers effect too
#Creating a datframe
list <- listings_df

#Model with outliers of property type
mod_4 <- lm(price~property_type, data = list)

#Check details
summary(mod_4)

#Filter the properties which have less than 5 listings
list %>%
  select(property_type) %>%
  group_by(property_type) %>%
  summarise(count = n()) %>%
  filter(count <5) %>%
  pull(property_type) -> prop

temp <- list %>% filter(!(property_type %in% prop))

#Create model with filtered variables
mod_5 <- lm(price~property_type, data = temp)

#Checking details
summary(mod_5)

#There isn't huge change so these outliers aren't removed

#Checking for cleaning fees
#Model created with outliers
mod_6 <- lm(price~cleaning_fee, data = list)

#Details of model
summary(mod_6)
```

```r
#Filter the value above 130 as there was a gap in histogram from this point nearly
temp <- list %>% filter(cleaning_fee < 130)
#Removed 11 observations

#Created a model without outliers
mod_7 <- lm(price~cleaning_fee, data = temp)

#Checking details of the model
summary(mod_7)

#There is a significant change in the R2 valuefrom 0.157 to 0.179

#Updating main dataframe
listings_df <- listings_df %>% filter(cleaning_fee < 130)

#Checking for cancellation policy
#Model created with outliers
mod_8 <- lm(price~cancellation_policy, data = list)

#Details of model
summary(mod_8)

#Filter the data set with strict 60 days as it has very low value
temp <- list %>% filter(cancellation_policy != "super_strict_60")

#Check model without outlier
mod_9 <- lm(price~cancellation_policy, data = temp)

#Details of model
summary(mod_9)

#There isn't significant change in the model so outlier not removed

#Removing load from environment
rm(list)
rm(temp)
rm(mod_1)
rm(mod_2)
rm(mod_3)
rm(mod_4)
rm(mod_5)
rm(mod_6)
rm(mod_7)
rm(mod_8)
rm(mod_9)
```

Creating new variables

Finding gender of host through their first name:

```r
#Create a dataframe for gender
#Find host gender through gender library
#Using first name
gender_df <-  unique(gender::gender(stringr::word(listings_df$host_name,1)))
```

```r
#Select name and gender from dataframe
gender_df <- gender_df %>% select(c(name, gender))

#Extracting first name of host
listings_df$host_first_name <- stringr::word(listings_df$host_name,1)

#Create name dataframe
name_df <- listings_df %>% select(c(id, host_first_name))

#Rename the column for joining
name_df <- rename(name_df, name = host_first_name)

#Join the dataframes of name and gender
name_df <- name_df %>%
  left_join(gender_df)


#Calculate gender distribution
table(name_df$gender)

# #Calculate the gender distribution rate
#count(filter(name_df$gender == "male")/sum(!is.na(name_df$gender))

#Assign gender to NAs by the proportion of identified genders
name_df$gender <- ifelse(is.na(name_df$gender), ifelse(which(is.na(name_df$gender)) < 2422, "female", "

#Extract gender from dataframe
name_df <- name_df %>% select(gender)

#Combine with main dataframe of listings
listings_df <- cbind(listings_df, name_df)
```

Creating variables with listing name to check if they can effect price

```r
#Number of words in listing name
listings_df$name_word <- sapply(gregexpr("\\W+", listings_df$name), length) + 1

#Number of characters in listing name
listings_df$name_character <- str_count(listings_df$name, "\\S+")

#Variables now are 60
```

Creating Variables from verification column and calculating number of verifications each listing have:

```r
#Cleaning amenities as it have extra symbols
listings_df$host_verifications <- tolower(listings_df$host_verifications)
listings_df$host_verifications <- gsub("\\[", "", listings_df$host_verifications)
listings_df$host_verifications <- gsub("\\]", "", listings_df$host_verifications)
listings_df$host_verifications <- gsub("\\'", "", listings_df$host_verifications)

#Subset of amenities
host_verifications_df <- listings_df %>% select(c(id, host_verifications))

#Tokenize verifications through separator
host_verifications_token <- host_verifications_df %>%
```

```
  unnest_tokens(word,host_verifications,token="regex",pattern="\\,")

#Calculate count of each host_verifications
host_verifications_count <- host_verifications_token %>%
  group_by(word) %>%
  summarise(count=n()) %>%
  arrange(desc(count))

#Visualize the count of host_verifications
host_verifications_count %>% slice_max(count, n =10) %>%
ggplot(aes(x = count, y = reorder(word,count))) + geom_col(fill = "lightblue", color = "black") +labs(t

#Creating a dataframe to make a variable of how many host_verifications each listings have
host_verifications_df <- host_verifications_token %>%
  group_by(id) %>%
  summarise(verification_count=n()) %>%
  arrange(desc(verification_count))

#Join the variable with listings main data set
listings_df <- listings_df %>%
  left_join(host_verifications_df)

#Replacing NA with zero
listings_df$verification_count[is.na(listings_df$verification_count)] <- 0
```

Visualizing distribution of numeric variables

```
#Visualize bedrooms in listings name
ggplot(listings_df, aes(x = bedrooms)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(title = "Distribution of Bedrooms", x = "", y ="Frequency")

#Visualize beds in listings name
ggplot(listings_df, aes(x = beds)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(title = "Distribution of Beds", x = "", y ="Frequency")

#Visualize bathrooms in listings name
ggplot(listings_df, aes(x = bathrooms)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(title = "Distribution of Bathrooms", x = "", y ="Frequency")

#Visualize accommodates in listings name
ggplot(listings_df, aes(x = accommodates)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(title = "Distribution of Accommodates", x = "", y ="Frequency")

#Filter zero beds and bedrooms
#There would not be any accommodation with zero beds and bedrooms
#We cannot assume any value for 207 observations
listings_df <- listings_df %>% filter(bedrooms > 0 | beds > 0)
```

```
#Observations of listing now are 4646
```

Creating variables with amenities so we can check their significance on rating and price of a listing.

```
#Cleaning amenities as it have extra symbols
listings_df$amenities <- tolower(listings_df$amenities)
listings_df$amenities <- gsub("\\{", "", listings_df$amenities)
listings_df$amenities <- gsub("\\}", "", listings_df$amenities)
listings_df$amenities <- gsub('\\\"', "", listings_df$amenities)

#Subset of amenities
amenities_df <- listings_df %>% select(c(id, amenities, price))

#Tokenize amenities through separator
amenities_token <- amenities_df %>%
  unnest_tokens(word,amenities,token="regex",pattern="\\,")

#Calculate count of each amenity
amenities_token_review <- amenities_token %>%
  group_by(word) %>%
  summarise(count=n()) %>%
  arrange(desc(count))

#Visualize the count of amenities
amenities_token_review %>% slice_max(count, n =10) %>%
ggplot(aes(x = count, y = reorder(word,count))) +
  geom_col(fill = "lightblue", color = "black") +
  labs(title = "Top Amenities in Listing", x = "Frequency", y = "Amenity")

#Finding percentage of listings with each amenity
amenities_token_review$percentage <- (amenities_token_review$count / nrow(listings_df)) *100

#Creating dataframe to make a variable of how many amenities each listings have
amenities_total_df <- amenities_token %>%
  group_by(id) %>%
  summarise(amenities_count=n()) %>%
  arrange(desc(amenities_count))

#Join the variable with listings main data set
listings_df <- listings_df %>%
  left_join(amenities_total_df)


#Replacing NA with zero
#9 observations were NA
listings_df$amenities_count[is.na(listings_df$amenities_count)] <- 0

#Filter the amenities which can influence price
#Amenities which are almost in each of the listing is removed
#Amenity which are rare are been removed
amenities_filter <- amenities_token_review %>% filter(between(percentage, 5, 80)) %>% pull(word)

#Do one hot coding
#Create a dataframe with ID
```

```
amenities_token <- amenities_token %>%
  group_by(id, word) %>%
  summarise(count=n()) %>%
  arrange(desc(count)) %>%
  mutate(count = ifelse(count > 1, 1, count)) #replace any number above 1 to 1 for one hot coding

#Pivot for one hot coding
amenities_token <- amenities_token %>%
  filter(word %in% amenities_filter) %>%
  pivot_wider(names_from = word, values_from = count, values_fill = 0L)

#Modify colnames according to proper name
colnames(amenities_token) <- gsub(" ","_",colnames(amenities_token))
colnames(amenities_token) <- gsub("/","_",colnames(amenities_token))
colnames(amenities_token) <- gsub("-","_",colnames(amenities_token))
colnames(amenities_token) <- gsub("_'n","",colnames(amenities_token))

#Save amenities dataframe
saveRDS(amenities_token, "amenities_token.RDS")
```

Creating a dataframe with price log and amenities so we can check the significance and filter them to join in main dataframe.

```
amenities_token <- readRDS("amenities_token.RDS")
#Create a dataframe with price and id
list_price <- listings_df %>% select(c("id", "price_log"))

#Combine amenities with the dataframe
amenities_token <- amenities_token %>% inner_join(list_price)

#Create a dataframe without id to check significant amenities
amenities_token_df <- amenities_token

#Remove id variable
amenities_token_df$id <- NULL

# #Create a linear regression model with all variables
model_8 <- lm(price_log~.,data = amenities_token_df)
#
# #Put in stepAIC for initial variable filtering
MASS::stepAIC(model_8, k = log(nrow(listings_df)))
```

```
#Creating model with variables suggested by AIC
summary(lm(price_log ~ tv + cooking_basics + dryer + free_street_parking +
    host_greets_you + hot_water + long_term_stays_allowed + luggage_dropoff_allowed +
    microwave + private_entrance + refrigerator + `translation_missing:_en.hosting_amenity_49` +
    washer + breakfast + elevator + family_kid_friendly + internet +
    first_aid_kit + fire_extinguisher + pets_allowed + smoking_allowed +
    lockbox + self_check_in + bathtub + lock_on_bedroom_door +
    dishwasher + shower_gel, data = amenities_token_df))
#Multiple R-squared:  0.2635,   Adjusted R-squared:  0.2592

#Removed translation missing: en.hosting_amenity_49
summary(lm(price_log ~ tv + cooking_basics + dryer + free_street_parking +
```

```
    host_greets_you + hot_water + long_term_stays_allowed + luggage_dropoff_allowed +
    microwave + private_entrance + refrigerator +
    washer + breakfast + elevator + family_kid_friendly + internet +
    first_aid_kit + fire_extinguisher + pets_allowed + smoking_allowed +
    lockbox + self_check_in + bathtub + lock_on_bedroom_door +
    dishwasher + shower_gel, data = amenities_token_df))
#Multiple R-squared:  0.2569,    Adjusted R-squared:  0.2527
```

```
#Filter significant variables
amenities_token <- amenities_token %>% dplyr::select(c("id", "tv", "cooking_basics", "dryer", "free_str
    "host_greets_you", "hot_water", "long_term_stays_allowed", "luggage_dropoff_allowed",
    "microwave", "private_entrance", "refrigerator",
    "washer", "breakfast", "elevator", "family_kid_friendly", "internet",
    "first_aid_kit", "fire_extinguisher", "pets_allowed", "smoking_allowed",
    "lockbox", "self_check_in", "bathtub", "lock_on_bedroom_door",
    "dishwasher", "shower_gel"))
```

Dealing with NA values in reviews

Visualize the NA values in the reviews:

```
#defining function to get total NAs per column
na_count <- data.frame(sapply(listings_df, function(y) sum(length(which(is.na(y))))))

#Let us now set the rownames as the column

na_count <- rownames_to_column(na_count, "column")

names(na_count)[2] <- "na_sum"

#filter for columns with NA greater than 0
na_count <- na_count %>%
  filter(na_sum > 0) %>%
  arrange(desc(na_sum))

#Let's plot these values
ggplot(na_count, aes(na_sum, reorder(column, na_sum))) +
  geom_col(fill = "lightblue", color = "black") +
  scale_x_continuous(labels = scales::comma) +
  labs(x = "Empty Value Count", y = "Variable")
```

Making two dataframes for listings, one is for reviews analysis and other one is for pricing analysis. There is a high number of NA values in reviews score so they are removed for reviews but for pricing the variable has been removed as removal of those observations would have cost information loss.

```
#Creating listings dataframe with reviews
listings_review_df <- na.omit(listings_df)
#Observations are 3600

#There are variables which cannot be generalized through replacement because of the influence
#it will have on the variables
#Filtering variables which have more than 500 NA values
na_count %>%
  filter(na_sum > 0) %>%
  arrange(desc(na_sum)) %>%
```

```
  filter(na_sum > 500) %>%
  pull(column) -> variable

#Filter the main data set
listings_price_df <- listings_df %>% dplyr::select(-variable)

#Omit NA values from the data set
listings_price_df <- na.omit(listings_price_df)

listings_price_df$host_response_rate <- as.numeric(gsub("%", "", listings_price_df$host_response_rate))
listings_price_df$host_response_rate[is.na(listings_price_df$host_response_rate)] <- 0

listings_price_df$host_acceptance_rate <- as.numeric(gsub("%", "", listings_price_df$host_acceptance_ra
listings_price_df$host_acceptance_rate[is.na(listings_price_df$host_acceptance_rate)] <- 0

#Join the amenities with listings main data
listings_price_df <- listings_df %>% left_join(amenities_token)

#Only consider those listings that do not have too many common/rare amenities
#replacing NAs with 0s for amenities
listings_price_df[is.na(listings_price_df)] <- 0L
#Observations are 4476
```

```
#using inner join to only keep those listings that have reviews
combined_df <- inner_join(listings_review_df, reviews_df, by = c("id" = "listing_id"), suffix = c("_lis
#Combined dataframe is 119416 observations and 96 variables

#Checking for duplicate rows
combined_df <- distinct(combined_df) #There was no duplication as observations count remains the same
```

```
#compute number of characters in review

combined_df$review_length_words <- str_count(combined_df$comments, '\\w+')

#compute summary of review_length_chars
summary(combined_df$review_length_words)

#plot review_length_chars
ggplot(combined_df, aes(review_length_words)) +
  geom_histogram(fill = "lightblue", color = "black") +
  xlim(0,500) +
  labs(title = "Review Length(All)", x = "Word Length", y = "frequency")

#arranging combined_df in descending order of review length in characters
combined_df <- combined_df %>%
  arrange(review_length_words) %>%
  filter(review_length_words > 5) #5 word reviews are removed because it is not a complete sentence and

#113397 observations remain after implementing both conditions
```

```
saveRDS(combined_df, "combined.RDS")
saveRDS(listings_price_df, "listings_price_clean.RDS")
saveRDS(listings_review_df, "listings_review_clean.RDS")
saveRDS(reviews_df, "reviews_clean.RDS")
```

```
rm(list=ls())
```

Part A

Part a

```
combined_df <- readRDS("combined.RDS")
```

Tokenising reviews

```
#loading the stop words data
data(stop_words)
data("Fry_1000")

#splitting the data into chunks of 10000 to make processing quicker
split_size <- 10000
tokens_list <- split(combined_df,
                     rep(1:ceiling(nrow(combined_df)
                                   /split_size),
                         each=split_size,
                         length.out=nrow(combined_df)))

#creating dataframe to bind the tokens to
tokens_all_comments <- data.frame()

#tokenising revies and removing stopwords
for(i in 1:length(tokens_list)){
   tokens_h <- tokens_list[[i]] %>%
    unnest_tokens(word, comments_new) %>%
    count(id, word, sort = TRUE) %>%
    anti_join(stop_words) %>%
    filter(!(word %in% Fry_1000)) %>%
    mutate(token_length = nchar(word))

   print(i)

   tokens_all_comments <- bind_rows(tokens_all_comments, tokens_h)
}
```

```
#edited host name to be added to stop words
host_name <- combined_df$host_name %>%
  unique() %>%
  replace_symbol() %>%
  tolower() %>%
  strsplit(., " ") %>%
  unlist() %>%
  unique() %>%
  removePunctuation()

#creating custom stop words
custom_stopwords <- c("airbnb","manchester", "apartment", "england", "united", "kingdom", "wouldn", "co
```

```r
#cleaning tokens by removing stop words
tokens_all_comments <- tokens_all_comments %>%
  filter(!(word %in% custom_stopwords))
```

Creating custom stopwords

```r
#review the token length
tokens_all_comments %>%
  group_by(token_length) %>%
  summarise(freq = sum(n))

#only keeping tokens with number of characters between 3 and 15
tokens_all_comments <- tokens_all_comments %>%
  filter(between(token_length,3,15))

#Inspecting most frequent words across all listings
tokens_all_comments %>%
  group_by(word) %>%
  summarise(freq = sum(n)) %>%
  arrange(desc(freq))

tokens_all_comments$token_length <- NULL
```

Summarizing tokenised data We considered using the hunspell package to correct spelling mistakes. While it is quick, the accuracy is not that high, it's a classic speed vs accuracy tradeoff. Hence, we have decided not to proceed with it.

```r
# correct <- hunspell_check(tokens_all_comments$word)
# summary(correct)
# #There are around 57,114 incorrect words
#
# #extracting the ones that are misspelled according to hunspell check
# misspelled <- tokens_all_comments[!correct, 2]
#
#
# uniq_misspelled <- unique(misspelled)
#
# # Using parallelism to make the process faster
# #extracting the element of each sublist
# suggestions <- map(map(
#         mclapply(uniq_misspelled, hunspell_suggest, mc.cores = 6),
#         1),1)
# # Some words don't appear in the spelling dictionary and are given NULL values
# # NULL messes up downstream processing and needs to be replaced with NA
# counter <- 0
# for(element in suggestions){
#     counter <- counter + 1 # indexing
#     if(is.null(element)){
#         suggestions[counter] <- NA # replacement
#     }
# }
# suggestions <- unlist(suggestions) # easier for downstream processing
#
```

```
# corrected_index <- match(misspelled, uniq_misspelled)
# misspelled <- suggestions[corrected_index]
#
# # feed corrections back into the original data frame
# dataBlogs[!correct, 2] <- misspelled
#
# # Because of the NULL correction up above, some lines now have NA values for
# # words. These lines need to be entirely removed.
# na_lines <- dataBlogs[is.na(dataBlogs$words), "linenumber"]
# dataBlogs <- dataBlogs %>%
#         filter(!(linenumber %in% na_lines))

#take screenshot of the spelling suggestions....
```

```
#lemmatization
tokens_all_comments$word <- lemmatize_words(tokens_all_comments$word,
                                            dictionary = lexicon::hash_lemmas)
```

Lemmatization on reviews

```
#computing the tf-idf for comments
tf_idf_reviews <- tokens_all_comments %>%
  group_by(id) %>%
  count(word, sort = TRUE) %>%
  ungroup() %>%
  bind_tf_idf(word, id, n) %>%
  arrange(desc(tf_idf))

# lets try to filter important words
# using the left and right trim
tf_idf_reviews <- tf_idf_reviews %>%
  filter(tf_idf < 0.2)

# Filter important words using left and right trim
ggplot(tf_idf_reviews, aes(tf_idf)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(title = "TF-IDF plot", x = "tf-idf", y = "Frequency")

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

#Visualise the graph
ggplot(tf_idf_reviews, aes(tf_idf)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(title = "TF-IDF plot", x = "tf-idf", y = "Frequency")

# Lets now remove also very common terms
# those with tf-idf < 0.001 as shown in the chart below
tf_idf_reviews <- tf_idf_reviews %>%
```

```
  filter(tf_idf > 0.001)

ggplot(tf_idf_reviews, aes(tf_idf)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(title = "TF-IDF plot", x = "tf-idf", y = "Frequency")

#Saving file for efficiency
saveRDS(tf_idf_reviews, "tf_idf_reviews.RDS")
```

```
tf_idf_reviews <- readRDS("tf_idf_reviews.RDS")
```

Plotting the tf-idf and filtering based on the upper and lower cutoffs.

Tokenising Listing Description

```
#creating description to bind tokens to
tokens_all_description <- data.frame()

#tokenising listing description and removing stopwords
for(i in 1:length(tokens_list)){
   tokens_h <- tokens_list[[i]] %>%
    unnest_tokens(word, description_new) %>%
    count(id, word, sort = TRUE) %>%
    anti_join(stop_words) %>%
    filter(!(word %in% Fry_1000)) %>%
    mutate(token_length = nchar(word))

   print(i)

   tokens_all_description <- bind_rows(tokens_all_description, tokens_h)
}
```

```
#creating custom stop words
custom_stopwords_desc <- c("airbnb","manchester", "apartment", "england", "united", "kingdom")

#cleaning tokens by removing stop words
tokens_all_description <- tokens_all_description %>%
  filter(!(word %in% custom_stopwords_desc))
```

Creating custom stopwords

```
#review the token length
tokens_all_description %>%
  group_by(token_length) %>%
  summarise(freq = sum(n))

#only keeping tokens with number of characters between 3 and 15
tokens_all_description <- tokens_all_description %>%
  filter(between(token_length,3,15))

#Inspecting most frequent words across all listings
```

```
tokens_all_description %>%
  group_by(word) %>%
  summarise(freq = sum(n)) %>%
  arrange(desc(freq))

tokens_all_description$token_length <- NULL
```

Summarizing tokenised data

```
#lemmatization
tokens_all_description$word <- lemmatize_words(tokens_all_description$word,
                                               dictionary = lexicon::hash_lemmas)
```

Lemmatization for listing description

```
#computing the tf-idf for listing description
tf_idf_description <- tokens_all_description %>%
  group_by(id) %>%
  count(word, sort = TRUE) %>%
  ungroup() %>%
  bind_tf_idf(word, id, n) %>%
  arrange(desc(tf_idf))

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

# Ok the tfidf is cut
# lets try to filter important words
# using the left and right trim
tf_idf_description <- tf_idf_description %>%
  filter(tf_idf < 0.5)

# Filter important words using left and right trim
hist(tf_idf_description$tf_idf, breaks = 200, main="TF-IDF plot")

# Ok from the plot we see that the cut-off value
# is at 0.3
tf_idf_description <- tf_idf_description %>%
  filter(tf_idf < 0.3)

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

# Lets now remove also very common terms
# those with tf-idf < 0.001 as shown in the chart below
tf_idf_ <- tf_idf_description %>%
  filter(tf_idf > 0.001)

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

Plotting the tf-idf and filtering based on the upper and lower cutoffs. What are the dominant words per aggregation category (neighborhood, access to public transport etc.)?

Reviews

Let's review the top 10 words

```
top_10_tokens_reviews <- tf_idf_reviews %>%
  group_by(word) %>%
  summarise(total = sum(n)) %>%
  arrange(desc(total)) %>%
  top_n(10)

top_10_tokens_reviews
```

Creating a function to generate dominant words per category

```
#creating a function to get the most dominant words
#per different category to avoid re-writing the code
top_words_by_category <- function(col, top_tokens) {

    top_5_cat <- combined_df %>%
        select(id, !!as.name(col)) %>%
        unique(.) %>%
        group_by(!!as.name(col)) %>%
        summarise(total = n()) %>%
        arrange(desc(total)) %>%
        top_n(5)

    top_5_tokens <- combined_df %>%
        select(id, !!as.name(col)) %>%
        filter(!!as.name(col) %in% top_5_cat[[1]]) %>%
        unique(.)

    tokens <- top_tokens %>%
        right_join(top_5_tokens) %>%
        group_by(!!as.name(col), word) %>%
        summarise(total = sum(n)) %>%
        arrange(desc(total))

    for(subcat in 1:nrow(top_5_cat)){
       print(paste0("For ", top_5_cat, ": ",  top_5_cat[[1]][subcat]))

       toprint <- tokens %>%
            ungroup() %>%
            filter(!!as.name(col) == top_5_cat[[1]][subcat]) %>%
            top_n(10, total)

       print(toprint)

       plot <- toprint %>%
           ggplot(aes(total, reorder(word, total)), fill = word) +
           geom_col(show.legend = FALSE)

       print(plot)
      }
}
```

```
top_words_by_category("neighbourhood_cleansed", tf_idf_reviews)
```

Reviews - Top words by Neighbourhood

```
top_words_by_category("property_type", tf_idf_reviews)
```

Reviews - Top words by Property Type

```
top_words_by_category("room_type", tf_idf_reviews)
```

Reviews - Top words by Room Type

```
top_words_by_category("host_identity_verified", tf_idf_reviews)
```

Reviews - Top words by Host Verified

```
top_words_by_category("host_is_superhost", tf_idf_reviews)
```

Reviews - Top words by Host Superhost

Listing Description

Let's review the top 10 words

```
top_10_tokens_description <- tf_idf_description %>%
  group_by(word) %>%
  summarise(total = sum(n)) %>%
  arrange(desc(total)) %>%
  top_n(10)

top_10_tokens_description
```

```
top_words_by_category("neighbourhood_cleansed", tf_idf_description)
```

Description - Top words by Neighbourhood

```
top_words_by_category("property_type", tf_idf_description)
```

Description - Top words by Property Type

```
top_words_by_category("room_type", tf_idf_description)
```

Description - Top words by Room Type

```
top_words_by_category("host_identity_verified", tf_idf_description)
```

Description - Top words by Host Verified

```
top_words_by_category("host_is_superhost", tf_idf_description)
```

Description - Top words by Host Superhost

Part b

In this part we are trying to answer the question of finding the most common word combinations for describing a property listing by exploring two perspectives: reviewers' point of view and hosts' point of view.

We will first consider finding correlations of word combinations for both perspectives using findAssoc(); performing bigram and find the top 10 frequent words for both perspectives and finally, we consider finding correlations of word combinations for reviewers' perspective using pairwise_cor().

Analyzing correlations using findAssoc()

We try finding associations from two textual columns: reviews and property descriptions and we will then compare which set yields to the best representation.

```
# from reviewers' perspective

#cast tf-idf for reviews to document-term matrix
reviews_dtm <- tf_idf_reviews %>%
  cast_dtm(id, word, n)

#assigning empty data-frame
associations <- data.frame()

#find associations between words & their corresponding correlations from the tokenized top 10 list
for (token in top_10_tokens_reviews$word) {
  assocs_list <- findAssocs(reviews_dtm, token, corlimit = 0.7)
  assocs <- data.frame(unlist(assocs_list))
  associations <- bind_rows(associations, assocs)
}

associations <- rownames_to_column(associations)
colnames(associations) <- c("word_combo", "corr")
associations <-  associations %>% separate(word_combo, c("word", "associated_word"))
associations

#visualizing words association with correlation more than 0.7
assoc_plot_list <- list()

counter = 1

for (term in unique(associations$word)) {

  word_df <- associations %>%
    filter(word == term) %>%
    top_n(10, corr)

  assoc_plot <- ggplot(word_df, aes(corr, associated_word)) + geom_point(size = 2) + theme_minimal()+th

  assoc_plot_list[[counter]] <- assoc_plot
```

```
  counter = counter + 1
  print(counter)

}
jpeg(file="saving_plot1.jpeg", width = 1080, height = 1080)
gridExtra::grid.arrange(grobs = assoc_plot_list, top="Correlations for word association", ncol=2)
```

```
# from hosts' perspective

#casting tf-idf words from the description column to document-term matrix
description_dtm <- tf_idf_description %>%
  cast_dtm(id, word, n)

#assigning empty data frame to begin with
associations1 <- data.frame()

#find associations between words & their corresponding correlations from the tokenized top 10 list
for (token1 in top_10_tokens_description$word) {
  assocs_list1 <- findAssocs(description_dtm, token1, corlimit = 0.35)
  assocs1 <- data.frame(unlist(assocs_list1))
  associations1 <- bind_rows(associations1, assocs1)
}

associations1 <- rownames_to_column(associations1)
colnames(associations1) <- c("word_combo", "corr")
associations1 <-  associations1 %>% separate(word_combo, c("word", "associated_word"))
associations1


top_n(associations1, 20)

#visualizing words association with correlation more than 0.7
assoc_plot_list1 <- list()

counter1 = 1

for (term1 in unique(associations1$word)) {

  word_df1 <- associations1 %>%
    filter(word == term1) %>%
    top_n(10, corr)

  assoc_plot1 <- ggplot(word_df1, aes(corr, associated_word)) + geom_point(size = 2) + theme_minimal()+

  assoc_plot_list1[[counter1]] <- assoc_plot1

  counter1 = counter1 + 1
  print(counter1)

}
jpeg(file="saving_plot2.jpeg", width = 1080, height = 1080)
gridExtra::grid.arrange(grobs = assoc_plot_list1, top="Correlations for word association", ncol=2)
```

Discussion: From the customer reviews plot we can deduce that 'comfortable' was being the most correlated

word used in describing a property, with a correlation of above 0.75 to its respective correlated words. However, from the 'findAssoc' function we can see that it also displayed the same pair twice (ie: comfortable lovely, corr=0.74 and lovely comfortable, corr=0.74). Thus, we can also consider exploring an alternative method to find the most common word combinations used to describe a property listing by using n-gram when n=2 in the next step.

It seems that the 'reviews' column yields to higher word association correlation values than those from the listing 'description' column. From the second plot, the highest word pair correlation we could extract from the 'description' column was at 0.45 and we are unable to derive useful / coherent insights from its word combinations (ie: kitchen-bathroom).

Analyzing bigrams

We are also interested in other alternatives in describing property listing from both reviewers' and hosts' point of views. Instead of looking at the word association like we did in the previous step, we are now interested in checking the top 10 word-pairs by its frequency / popularity to describe property listings from reviewers' and hosts' perspective and visualize them.

```
# from reviewers' perspective

data(stop_words)
data("Fry_1000")

#creating custom stop words
custom_stopwords <- c("airbnb","manchester", as.character(unique(tolower(combined_df$host_name))))

#splitting the dataset into smaller chuncks for faster processing time
split_size <- 10000
tokens_list <- split(combined_df,
                     rep(1:ceiling(nrow(combined_df)
                                   /split_size),
                         each=split_size,
                         length.out=nrow(combined_df)))

#constructing bigram from the dataset
n_grams <- data.frame()
for(i in 1:length(tokens_list)){
   tokens_g <- tokens_list[[i]] %>%
     unnest_tokens(word, comments_new, token="ngrams", n=2)%>%
       separate(word, c("word1","word2"),sep = " ") %>%

       #filtering out stopwords
       filter(!word1 %in% stop_words$word) %>%
       filter(!word2 %in% stop_words$word) %>%
       filter(!(word1 %in% Fry_1000)) %>%
       filter(!(word2 %in% Fry_1000)) %>%
       filter(!(word1 %in% custom_stopwords)) %>%
       filter(!(word2 %in% custom_stopwords))

      token_counts <- tokens_g %>%
      count(id, word1, word2, sort = TRUE)
      united_token_counts <- token_counts %>%
      unite(word, word1,word2, sep = " ")

      print(i)
```

```
      n_grams <- bind_rows(n_grams, united_token_counts)
}


#arranging tokenized bigram in descending order by tf-idf
n_grams_tf_idf <- n_grams %>%
    group_by(id) %>%
    count(word, sort = TRUE) %>%
    ungroup() %>%
    bind_tf_idf(word, id, n) %>%
    arrange(desc(tf_idf))

#visualizing a histogram
hist(n_grams_tf_idf$tf_idf, breaks = 200, main="TF_IDF Plot Before Trimming")

#from the plot we can see that the right cut-off value is at 0.4
n_grams_tf_idf <- n_grams_tf_idf %>% filter(tf_idf<0.4)
hist(n_grams_tf_idf$tf_idf, breaks = 200, main="TF_IDF Plot")

#checking for the upper / lower quantile of tf-idf
quantile(n_grams_tf_idf$tf_idf, na.rm = TRUE, c(0.025, 0.975))
#2.5%       97.5%
#0.006451368 0.270069455

#imposing further upper and lower trim using quantile
n_grams_tf_idf <- n_grams_tf_idf %>%
  filter(between(tf_idf, 0.006451368, 0.270069455)) %>%
  arrange(desc(tf_idf))

#plotting histogram
hist(n_grams_tf_idf$tf_idf, breaks = 200, main="TF_IDF Plot After Trimming")

#arranging the filtered tf-idf bigram in descending order and extract the top 10 pairs
n_grams_tf_idf <- n_grams_tf_idf %>% group_by(word) %>%
  summarise(total = n()) %>%
  arrange(desc(total)) %>%
  top_n(10)
n_grams_tf_idf

#plotting the top 10 bigram
bigram_plot <- n_grams_tf_idf %>%
              arrange(desc(total=n())) %>%
              slice_max(total, n = 10) %>%
              ungroup() %>%
              mutate(word = reorder(word, total))%>%
              ggplot(aes(total, word, fill=word))+

              geom_col(show.legend = FALSE) +

              labs(x = "Frequencies of Bigram", y = NULL)
bigram_plot
ggsave(bigram_plot, filename = "bigram_plot.png")
```

```
#we can also consider plotting bigram relationships
#in directed graphs to visualize the correlations and
#clusters of words that were found by the widyr package
library(igraph)
#the original counts
token_counts <- token_counts %>%
                select(-c("id"))
token_counts

#filter for only relatively common combinations
n_gram_graph <- token_counts %>%
                filter(n>4) %>%
                graph_from_data_frame()
n_gram_graph

library(ggraph)
set.seed(1234)

word_cluster <- ggraph(n_gram_graph, layout = "fr") +
                geom_edge_link(aes(edge_alpha = n),
                               arrow = arrow(length = unit(2, "mm")), end_cap = circle(.07, 'inches')) +
                geom_node_point(color = "lightblue", size = 3) +
                geom_node_text(aes(label = name)) +
                theme_void()
word_cluster
ggsave(word_cluster, filename = "word_cluster.png")
```

From the Frequencies of Bigram for Reviews chart, the word combination 'highly recommend' was being assigned as the top of the word combination list. It is evaluated based on the frequencies of their occurrences together. This is expected, especially when we are analysing from reviewers' perspective. Rather than discarding them from the analysis, we can keep them as they are and evaluate our remaining list. The term 'walking distance' appears as the most popular combination to describe property listing with a total of 1113 observations. Subsequently, by having a 'perfect location' for property listings comes in favour among reviewers. Other qualities such as lovely & friendly host and by having easy access to public transport also hold importance in reviewers' perspectives in describing property listings.

From the cluster of words diagram above, it shows bigram formation patterns for the 'reviews' column. In here, the arrows indicate how the words are ordered and the bolder the arrow is, the higher the frequency of the word association gets. For instance, as can be seen from the previous bar chart that the phrase "highly recommend" have the highest frequency and we can see that it has the boldest arrow in the diagram.

We are also interested in other alternatives in describing property listing from hosts' point of view. Instead of looking at the word association, we are now interested in checking the top 10 word pairs by its frequency / popularity to describe property listings from reviewers perspective and visualize them.

```
# from hosts' perspective

data(stop_words)
data("Fry_1000")

#creating custom stop words
custom_stopwords <- c("airbnb","manchester", as.character(unique(tolower(combined_df$host_name))))

#splitting the dataset into smaller chuncks for faster processing time
split_size <- 10000
```

```
tokens_list <- split(combined_df,
                     rep(1:ceiling(nrow(combined_df)
                                   /split_size),
                         each=split_size,
                         length.out=nrow(combined_df)))

#constructing bigram from the dataset
n_grams <- data.frame()
for(i in 1:length(tokens_list)){
   tokens_g <- tokens_list[[i]] %>%
     unnest_tokens(word, description_new, token="ngrams", n=2)%>%
       separate(word, c("word1","word2"),sep = " ") %>%

       #filtering out stopwords
       filter(!word1 %in% stop_words$word) %>%
       filter(!word2 %in% stop_words$word) %>%
       filter(!(word1 %in% Fry_1000)) %>%
       filter(!(word2 %in% Fry_1000)) %>%
       filter(!(word1 %in% custom_stopwords)) %>%
       filter(!(word2 %in% custom_stopwords))

      token_counts <- tokens_g %>%
      count(id, word1, word2, sort = TRUE)
      united_token_counts <- token_counts %>%
      unite(word, word1,word2, sep = " ")

      print(i)
      n_grams <- bind_rows(n_grams, united_token_counts)
}


#arranging tokenized bigram in descending order by tf-idf
n_grams_tf_idf <- n_grams %>%
    group_by(id) %>%
    count(word, sort = TRUE) %>%
    ungroup() %>%
    bind_tf_idf(word, id, n) %>%
    arrange(desc(tf_idf))

#visualizing a histogram
hist(n_grams_tf_idf$tf_idf, breaks = 200, main="TF_IDF Plot Before Trimming")

#from the plot we can see that the right cut-off value is at 1
n_grams_tf_idf <- n_grams_tf_idf %>% filter(tf_idf<1)
hist(n_grams_tf_idf$tf_idf, breaks = 200, main="TF_IDF Plot")

#checking for the upper / lower quantile of tf-idf
quantile(n_grams_tf_idf$tf_idf, na.rm = TRUE, c(0.025, 0.975))
#2.5%       97.5%
#0.1001823 0.7830426

#imposing further upper and lower trim using quantile
n_grams_tf_idf <- n_grams_tf_idf %>%
```

```
  filter(between(tf_idf, 0.1001823, 0.7830426)) %>%
  arrange(desc(tf_idf))

#plotting histogram
hist(n_grams_tf_idf$tf_idf, breaks = 200, main="TF_IDF Plot After Trimming")

#arranging the filtered tf-idf bigram in descending order and extract the top 10 pairs
n_grams_tf_idf <- n_grams_tf_idf %>% group_by(word) %>%
  summarise(total = n()) %>%
  arrange(desc(total)) %>%
  top_n(10)
n_grams_tf_idf

#plotting the top 10 bigram
library(ggplot2)
bigram_plot1 <- n_grams_tf_idf %>%
              arrange(desc(total=n())) %>%
              slice_max(total, n = 10) %>%
              ungroup() %>%
              mutate(word = reorder(word, total))%>%
              ggplot(aes(total, word, fill=word))+

              geom_col(show.legend = FALSE) +

              labs(x = "Frequencies of Bigram", y = NULL)
bigram_plot1

ggsave(bigram_plot1, filename = "bigram_plot1.png")
```

From the Frequencies of Bigram for Description chart, the word combination 'transport link' was being assigned as the top of the word combination list with a total of 201 observations. It is a lot fewer observations when we compare them with those from reviewers because there are significantly a lot of review observations than those from hosts' listing description. After reviewing the remaining observations, hosts like to use the accessibility (how easy to commute to / from the accommodation) of the property to describe property listings.

Analysing correlation using pairwise_cor()

One useful function from widyr is the pairwise_cor() function. It lets us find the phi coefficient between words based on how often they appear in the same section:

```
word_correlations <-  tokens_all_comments %>%
  semi_join(top_10_tokens_reviews, by = "word") %>%
  pairwise_cor(item = word, feature = id) %>%
  filter(correlation >= 0.2) %>%
  arrange(desc(correlation))
print(word_correlations)

graph_from_data_frame(d = word_correlations,
                      vertices = top_10_tokens_reviews %>%
                        semi_join(word_correlations, by = c("word" = "item1"))) %>%
                           ggraph(layout = "fr") +
                          geom_edge_link(aes(alpha = correlation)) +
                          geom_node_point() +
```

```
                              geom_node_text(aes(color = top_10_tokens_reviews$total, label = top_10_tokens
```

The network above shows the highly correlated unigrams based on reviewers' description of the property listing. The shade of edge colour shows the degree of correlation of each token pair. The token that has the highest occurrences / frequency for the overall reviews are shown in the lightest blue shade.

All in all, for the scope of this dataset, bigram seems to be a better choice in representing the most common word combinations used to describe a property listing because they provide more meaningful interpretations of the context. For future analysis, researcher can consider exploring n-gram when n=3 and decide which analysis would provide a more meaningful representation.

Part c

```
combined_df <- readRDS("combined.RDS")
tf_idf_reviews <- readRDS("tf_idf_reviews.RDS")
readability <- readRDS("readability_all_review.RDS")
reviews_df <- readRDS("reviews_clean.RDS")
listings_review_df <-  readRDS("listings_review_clean.RDS")
```

```
#average rating for each listing id
rating_categories <- combined_df %>%
  group_by(id) %>%
  summarise(avg_rating = mean(review_scores_rating)) %>%
  ungroup()

# Lets find the levels that we want to aggregate the words
quantile(rating_categories$avg_rating)

# now assign them in a rating group. Looking at the distibution we deicded to divide it into two rating
# the two rating categories are created
rating_categories$rating_category <- ifelse(rating_categories$avg_rating<96,1,2)

ratings_categories_tokens <- tf_idf_reviews %>% left_join(rating_categories) %>%
  group_by(rating_category,word) %>% summarise(total =sum(tf_idf))
# words with highest tf-idf belong to rating 1
ratings_categories_tokens %>% filter(rating_category==1) %>% arrange(desc(total)) %>% top_n(10)
# words with highest tf-idf belong to rating 2
ratings_categories_tokens %>% filter(rating_category==2) %>% arrange(desc(total)) %>% top_n(10)
```

```
#creating more variables in combined_df that can relate to rating score


#1 Number of characters in comments
combined_df$no_of_characters <-  str_count(combined_df$comments, "\\S+")
characters <- combined_df %>%
  group_by(id) %>%
  summarise(avg_chars = mean(no_of_characters))
```

```
# 2 gender of the host is another variable
# this variable was found in the data prepapration part and will be used for our future analysis
```

###Readability

```
# For readability for comments as a variable
readability_all_review <- data.frame()
```

```
for(i in 1:nrow(reviews_df)){
#for(i in 1:1000)



  readability_h <- data.frame()



  this_text <- iconv(all_data$comments[i])
  this_text <- removeNumbers(this_text)
#  this_text <- removePunctuation(this_text) should not remove punctuation so sentence count can be cal
  this_text <- str_replace_all(this_text, "[^A-Za-z0-9 ]","")



 tryCatch(readability_h <- flesch_kincaid(this_text),error=function(e){
    cat("Error parsing")})



 if(!is.null(readability_h$Readability)){



    readability_h <- readability_h$Readability
    readability_h$id <- reviews_df$id[i]
    readability_all_review <- bind_rows(readability_all_review,readability_h)
   }
}



#Average words in a sentence is 15 words https://techcomm.nz/Story?Action=View&Story_id=106#:~:text=How
readability_all_review$sent_n <- ceiling(readability_all_review$word.count/15)



readability_all_review$FK_grd.lvl <- 0.39 * (readability_all_review$word.count/readability_all_review$s
readability_all_review$FK_read.ease <- 206.835 - 1.015 * (readability_all_review$word.count/readability



#saveRDS(readability_all_review, "readability_all_review.RDS")
```

```
#using readability of comments and grouping the variables in them for each listing

readability <- left_join(readability_all_review,reviews_df,by ="id")
read1 <- readability %>% select(word.count,listing_id)
read1 <- read1 %>% group_by(listing_id) %>%
  summarise(avg_wordcount = mean(word.count))
```

```
read2 <- readability %>% select(listing_id,syllable.count)
read2 <- read2 %>% group_by(listing_id) %>%
  summarise(avg_syllable = mean(syllable.count))

read3 <- readability %>% select(listing_id,FK_grd.lvl)
read3 <- read3 %>% group_by(listing_id) %>%
  summarise(avg_FK_GRD = mean(FK_grd.lvl))

read4 <- readability %>% select(listing_id,FK_read.ease)
read4 <- read4 %>% group_by(listing_id) %>%
  summarise(avg_Fk_ease = mean(FK_read.ease))

read5 <- readability %>% select(listing_id,sentence.count)
read5 <- read5 %>% group_by(listing_id) %>%
  summarise(avg_sentence = mean(sentence.count))
```

```
# 4 number of verfications is also a variable that we found in the data preparation and it will be used
```

##Variables from listing_df

```
# verfication count
extractvariables <- listings_review_df %>% select(id,verification_count)
```

```
#data frame created for all the variables that will be helpful to predict rating
#charaters, word count, amentities
variables_rating <- left_join(extractvariables,characters, by ="id")
```

```
#alll readablity variables combined with the initial variables
variables_rating <- variables_rating %>% rename(listing_id = id)
variables_rating <- left_join(variables_rating,read1, by = "listing_id")
variables_rating<- left_join(variables_rating,read2, by = "listing_id")
variables_rating <- left_join(variables_rating,read3, by = "listing_id")
variables_rating <- left_join(variables_rating,read4, by = "listing_id")
variables_rating <- left_join(variables_rating,read5, by = "listing_id")
```

##Formality and diversity of commemts and description

```
all_listings_comments <- combined_df %>% select(id,comments)  %>%  group_by(id) %>% summarise(comments=

# 6 formality of comments as a variable
formality_comments<-with(all_listings_comments,formality(comments, id))
formality_comments_1 <-formality_comments[["formality"]]

# 7 diversity of comments as variable
diversity_comments<-with(all_listings_comments,diversity(comments, id))

#generating a dataframw which combined polarity, formality, diversity of comments according to id colum
all_listings_comments<- merge(all_listings_comments,formality_comments_1,by="id", all=TRUE)
all_listings_comments<- merge(all_listings_comments,diversity_comments,by="id", all=TRUE)

all_listings_comments<-all_listings_comments %>% select(id,review_scores_rating,"word.count",formality,

#rename columns to make them easier to interpret
all_listings_comments<-all_listings_comments %>% rename(wordcount_comments=wc)
all_listings_comments<-all_listings_comments %>% rename(formality_comments=formality)
```

```
all_listings_comments<-all_listings_comments %>% rename(diversity_comments=simpson)
```

```
 all_listings_description <- combined_df %>% select(id,description) %>%
    unique(.)
# 8 polarity of description as a variable
polarity_description <- with(all_listings_description,polarity(description, id))
polarity_description_1<-polarity_description[["all"]]

# 9 formality of description as a variable
formality_description<-with(all_listings_description,formality(description, id))
formality_description_1 <-formality_description[["formality"]]

# 10 diversity of description as a variable
diversity_description<-with(all_listings_description,diversity(description, id))


#generating a dataframe which combined polarity, formality, diversity of description according to id co
all_listings_description<- merge(all_listings_description,polarity_description_1,by="id", all=TRUE)
all_listings_description<- merge(all_listings_description,formality_description_1,by="id", all=TRUE)
all_listings_description<- merge(all_listings_description,diversity_description,by="id", all=TRUE)

all_listings_description<-all_listings_description %>% select(id,review_scores_rating,"word.count",pola

sentiment<-all_listings_description %>% select(id,polarity)

#rename columns to make them easier to interpret
all_listings_description<-all_listings_description %>% rename(wordcount_description=word.count)
sentiment<-sentiment %>% rename(polarity_description=polarity)
all_listings_description<-all_listings_description %>% rename(formality_description=formality)
all_listings_description<-all_listings_description %>% rename(diversity_description=simpson)
```

```
#Data combination of alll variables that could be possible in predicting the rating score
partA_c<-left_join(all_listings_comments,all_listings_description,by="id")
variables_rating <- left_join(variables_rating,partA_c, by = c("listing_id"="id"))

#initial variables that we will test in the model
variables_rating <- variables_rating %>% na.omit()
```

##other variables generates from raw data

```
library(ggplot2)
# We check the influence on whether the name of
# the owner is present in the review using the review rating score
# For this we are going to add a binary 0,1 variable
 combined_df$host_name_mentioned <-NA
 for(i in 1:nrow(combined_df)){
   check_host <- as.numeric(grepl(combined_df$host_name[i],
                                  combined_df$comments[i],
                                  ignore.case = T))
   combined_df$host_name_mentioned[i] <- check_host
 }

# How does it look graphically
ggplot(subset(combined_df,!is.na(host_name_mentioned)),aes(x=factor(host_name_mentioned),y=review_scores
#t.test used to show the significance
```

```
t.test(combined_df$review_scores_rating~factor(combined_df$host_name_mentioned))
```

Can be seen from t.test result that mention of host name can have a significant impact on the rating score. SO can me concluded that mentioning the host name is important. The mean review rating was 94 for no host name mentioned. While it was 96 when host name was mentioned.

```
# 11 avg amount of times host name is mentioned as a variable, grouped it by id since we are dealing wi
host_name <- combined_df %>%
  group_by(id) %>%
  summarise(avg_host_name = mean(host_name_mentioned))
```

```
# 12 amenities is another variable that we generated in the data preparation
```

```
# adding mention of host name as a variable
#host_name <- host_name %>% rename(id= listing_id)
host_rating <- left_join(host_name,rating_categories,by="id")
variables_rating <- left_join(variables_rating,host_rating, by = c("listing_id"="id"))

# adding amenities as variables
amenities_token <- readRDS("amenities_token.RDS")
variables_rating <- left_join(variables_rating, amenities_token, by = c("listing_id" = "id"))

#more variables to add from the listing data frame that can be helpful in prediction rating score

more_variables <- listings_review_df %>% select(id,host_is_superhost,gender,bedrooms,bathrooms,accommoda
variables_rating <- left_join(variables_rating,more_variables,by = c("listing_id"="id"))
#encoding the variables into binary
variables_rating$host_is_superhost =ifelse(variables_rating$host_is_superhost=="TRUE",1,0)
variables_rating$is_location_exact =ifelse(variables_rating$is_location_exact=="TRUE",1,0)
```

```
#write the data frame
saveRDS(variables_rating,"variables_rating.RDS")
saveRDS(sentiment,"sentiment.RDS")
```

Part d

```
rm(list=ls())
#Reading all the files
listings_price_df <- readRDS(file = "listings_price_clean.rds")
combined_df <- readRDS(file = "combined.RDS")
```

```
#Create a dataframe for readability results
readability_all <- data.frame()

#Constructing a for-loop for putting each readability result in dataframe
for(i in 1:nrow(listings_price_df)){

#Creating dataframe of each test
readability_h <- data.frame()

#Remove any symbol in text (/n,/r)
 this_text <- iconv(listings_price_df$description[i])
#Removing numbers from descriptions
 this_text <- removeNumbers(this_text)
#Remove anything beside English
```

```
 this_text <- str_replace_all(this_text, "[^A-Za-z0-9 ]","")

 #For any error creating a function
 #This will help completing all observations
 tryCatch(readability_h <- flesch_kincaid(this_text),error=function(e){
   cat("Error parsing")
 })

 #If results are not null lets put them in readability_all dataframe
 if(!is.null(readability_h$Readability)){
    readability_h <- readability_h$Readability
    readability_h$id <- listings_price_df$id[i] #We are grabbing ID so we can join the results
    readability_all <- bind_rows(readability_all,readability_h) #Putting each individual result togethe
  }
}

#There is incorrect count of sentences
#Function of grade level and ease has been created

#Create dataframe with description and ID
descrip <- listings_price_df %>% select(c("id", "description"))
#Remove whitespace and numbers from description so sentences can be read properly
descrip$description <- stripWhitespace(removeNumbers(descrip$description))


#Counting number of sentences in description
sentence_count <- sentSplit(descrip,"description") %>%
 group_by(id) %>%
 summarise(sent_n = n())

#Join number of sentences with the readablility dataframe
readability_all <- readability_all %>%
 left_join(sentence_count)

#Creating a function of grade level
readability_all$FK_grd.lvl <- 0.39 * (readability_all$word.count/readability_all$sent_n) + 11.8 * (read

#Creating a function of ease
readability_all$FK_read.ease <- 206.835 - 1.015 * (readability_all$word.count/readability_all$sent_n) -

#Save the result for efficiency
saveRDS(readability_all, "readability_all.RDS")
```

There was an incorrect counts of number of sentence in readability function, hence we have created a function and counted number of sentences through other function. This have given have proper value, there were negative value in one observation but grade level is positive so we have transformed negative to positive.

```
#Read the file of readability
readability_all <- readRDS(file = "readability_all.RDS")

#Join the results with listing dataframe for analysis
listings_price_df <- listings_price_df %>%
  inner_join(readability_all)
```

```
#Removing negative score
listings_price_df$FK_grd.lvl <- abs(listings_price_df$FK_grd.lvl)
```

```
#Finding formality level in description as per ID
formality <- formality(listings_price_df$description,listings_price_df$id)

#Select the formality
formality$formality %>% select(id,formality) -> formality_calc

#The ID was character so change it to numeric
formality_calc$id <- as.numeric(formality_calc$id)

#Save the results
saveRDS(formality_calc, "formality_calc.RDS")
```

```
#Read the file to environment
formality_calc <- readRDS(file = "formality_calc.RDS")

#Join the data with listing dataframe
 listings_price_df<- listings_price_df %>%
  inner_join(formality_calc)
# Joining, by = "id"
```

```
#Checking formality on the basis of property type
formality <- formality(listings_price_df$description,listings_price_df$property_type)
#Extracting formality results
formality$formality %>% select(property_type,formality) -> formality_property

#Save the file
saveRDS(formality_property, "formality_property.RDS")
```

```
#Read file
formality_property <- readRDS("formality_property.RDS")

#Visualise the results
ggplot(formality_property, aes(x = formality, y = reorder(property_type, formality))) +
  geom_col(fill = "lightblue", color = "black") +
  labs(title = "Formality of Description w.r.t Property Type", x = "Formality", y = "Property Type")
```

```
#Checking formality on the basis of room type
formality <- formality(listings_price_df$description,listings_price_df$room_type)
#Extracting formality results
formality$formality %>% select(room_type,formality) -> formality_room

#Save the file
saveRDS(formality_room, "formality_room.RDS")
```

```
#Read file
formality_room <- readRDS("formality_room.RDS")

#Visualise the results
ggplot(formality_room, aes(x = formality, y = reorder(room_type, formality))) +
  geom_col(fill = "lightblue", color = "black") +
  labs(title = "Formality of Description w.r.t Room Type", x = "Formality", y = "Room Type")
```

```
#Calculate the formality correlation with price based on property type
price_property_type <- listings_price_df %>% select(property_type, price) %>% group_by(property_type) %

#Join the results
formality_property <- formality_property %>% left_join(price_property_type)

#Calculate the correlation
cor(formality_property$formality,formality_property$avg_price)
#0.07430969 correlation
#This is negligible correlation
```

```
#Visualize the description features with price
#Formality
formality <- ggplot(listings_price_df, aes(x=formality,y=price)) + geom_smooth(method = "lm") + geom_po
#Grade level
grade_level <- ggplot(listings_price_df, aes(x=FK_grd.lvl,y=price)) + geom_smooth(method = "lm") + geom
#Read ease
read_ease <- ggplot(listings_price_df, aes(x=FK_read.ease,y=price)) + geom_smooth(method = "lm") + geom
# For working with the price
# we need to parse it first as a numeric
# variable

#Checking correlation with price
listings_price_df %>% select(c(price_log, FK_grd.lvl, FK_read.ease, formality)) %>%
  na.omit %>% cor() %>% corrplot::corrplot(method = "number")

# Ok basic regression model
model1 <- lm(price_log~FK_grd.lvl, data = listings_price_df)
summary(model1) #Multiple R-squared:  0.00966,  Adjusted R-squared:  0.009439
model2 <- lm(price_log~log(FK_grd.lvl), data = listings_price_df)
summary(model2) #Multiple R-squared:  0.03078,  Adjusted R-squared:  0.03056
model3 <- lm(price_log~FK_read.ease, data = listings_price_df)
summary(model3) #Multiple R-squared:  0.02882,  Adjusted R-squared:  0.0286
model4 <- lm(price_log~log(FK_read.ease), data = listings_price_df)
summary(model4) #Multiple R-squared:  0.03178,  Adjusted R-squared:  0.03156
model5 <- lm(price_log~formality, data = listings_price_df)
summary(model5) #Multiple R-squared:  0.00599,  Adjusted R-squared:  0.005768
model6 <- lm(price_log~log(formality), data = listings_price_df)
summary(model6) #Multiple R-squared:  0.006606, Adjusted R-squared:  0.006384
model_7 <- lm(price_log~word.count, data = listings_price_df)
summary(model_7) #Multiple R-squared:  0.003697,    Adjusted R-squared:  0.003475
model_8 <- lm(price_log~syllable.count, data = listings_price_df)
summary(model_8) #Multiple R-squared:  0.01394, Adjusted R-squared:  0.01372

#Create regression analysis results from several models side-by-side
stargazer::stargazer(model1,model2,model3,model4,model5,model6,type = "text")

# So we can clearly see from the regression models
# that more expensive properties have more complex description
# you may expect to find the same with the formality that we
# calculated together in the lab
```

The stargazer help in comparing all models and it help us in analyzing that only log of formality is significant.

```
#Creating a dataframe for testing variables
model_data <- listings_price_df %>% select(c("price_log","host_response_time",,"host_response_rate", "h

#Select variables for factor
variable <- c(2,5,6,7,8,9,10,15,21,22)

#Factorize all character variables
model_data[,variable] <- lapply(model_data[,variable] , factor)

#Individually checking variables to reduce Adj R sq
for(i in 2:length(colnames(model_data))){
  variable_df <- model_data %>%
    select(c("price_log",colnames(model_data[i])))
  print(colnames(model_data[i]))
  model <- lm(price_log ~ ., data = variable_df)
  print(summary(model))
}
```

We have not considered any factor related to rating in predicting price because review in post-experience factors. If we incorporate those variables our model would not be able to predict price of new listing.

```
#Using variables with higher than 0.01 R sq for efficiency
model_1 <- lm(price_log ~ host_response_time +
host_acceptance_rate +
neighbourhood_group_cleansed +
zipcode +
property_type +
room_type +
accommodates +
bathrooms +
bedrooms +
beds +
security_deposit +
cleaning_fee +
guests_included +
extra_people +
cancellation_policy +
amenities_count, data = model_data)

MASS::stepAIC(model_1)
```

```
summary(lm(price_log ~ host_response_time + host_acceptance_rate +
    neighbourhood_group_cleansed + zipcode + property_type +
    room_type + accommodates + bathrooms + bedrooms + beds +
    guests_included + cancellation_policy + amenities_count,
    data = model_data))
#Multiple R-squared:  0.8881,   Adjusted R-squared:  0.7514

summary(lm(price_log ~ property_type + room_type + accommodates +
    bathrooms + bedrooms + beds + cleaning_fee + guests_included + amenities_count + zipcode,
    data = model_data))
#Multiple R-squared:  0.8869,   Adjusted R-squared:  0.7503

#Removing insignificant variables
summary(lm(price_log ~ property_type + room_type + accommodates +
```

```
    bathrooms + bedrooms + beds + guests_included + zipcode,
    data = model_data))
#Multiple R-squared:  0.8868,   Adjusted R-squared:  0.7503

#Log accomodates due to skewness
summary(lm(price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + bathrooms + beds + zipcode,
    data = model_data))
#Multiple R-squared:  0.8882,   Adjusted R-squared:  0.7535

summary(lm(price_log ~ property_type + room_type + log(accommodates) +
    beds + zipcode,
    data = model_data))
#Multiple R-squared:  0.8867,   Adjusted R-squared:  0.7504

#Selecting the best model
model_base <- lm(price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + bathrooms + beds + zipcode,
    data = model_data)
```

```
model_b1 <- lm(price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + bathrooms + beds + zipcode + FK_read.ease,
    data = listings_price_df)

#Likelihood ratio test
anova(model_base,model_b1)
# p - 0.1174

model_b2 <- lm(price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + bathrooms + beds + zipcode + formality,
    data = listings_price_df)

#Likelihood ratio test
anova(model_a,model_b2)
# p - 0.2606

model_b3 <- lm(price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + bathrooms + beds + zipcode + log(formality),
    data = listings_price_df)

#Likelihood ratio test
anova(model_a,model_b3)
# p - 0.319

model_b4 <- lm(price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + bathrooms + beds + zipcode + log(FK_grd.lvl),
    data = listings_price_df)

#Likelihood ratio test
anova(model_a,model_b4)
# p - 0.0765

model_b5 <- lm(price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + bathrooms + beds + zipcode + FK_read.ease + formality + log(FK_grd.lvl),
```

```
    data = listings_price_df)


#Likelihood ratio test
anova(model_a,model_b5)
# p - 0.2609

#Checking the variable through forward selection method
# MASS::stepAIC(model_b5)

#Recommended model by stepAIC
model_b6 <- lm(formula = price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + bathrooms + beds + zipcode + log(FK_grd.lvl),
    data = listings_price_df)

summary(model_b6)
#Multiple R-squared:  0.8884,   Adjusted R-squared:  0.7538

#Base model
summary(lm(price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + bathrooms + beds + zipcode,
    data = model_data))
#Multiple R-squared:  0.8882,   Adjusted R-squared:  0.7535

anova(model_base, model_b6)
#p - 0.0765

#Adding formality variables provide significant difference in base model but still isn't highly signifi
#These variables addition does not improve model highly
#formality & log(FK_grd.lvl) are insignificant in the model
```

After running through we got log of grade level as significant variable in multivariate regression for predicting price. But, with overall model the variable is insignificant. We have removed the variables which are explaining the features of description as they does not have significant affect on the model. Keeping these model will cause over fitting of the model.

```
#Creating a dataframe with base model and amenities
#Creating a regression model
model_comb <- lm(price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + bathrooms + beds + log(FK_grd.lvl) + zipcode + tv + cooking_basics + dryer + free_street
    host_greets_you + hot_water + long_term_stays_allowed + luggage_dropoff_allowed +
    microwave + private_entrance + refrigerator +
    washer + breakfast + elevator + family_kid_friendly + internet +
    first_aid_kit + fire_extinguisher + pets_allowed + smoking_allowed +
    lockbox + self_check_in + bathtub + lock_on_bedroom_door +
    dishwasher + shower_gel, data = listings_price_df)

#Checking the variable through forward selection method
MASS::stepAIC(model_comb)
```

```
#Variables filtered by stepAIC
summary(lm(price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + bathrooms + beds + log(FK_grd.lvl) +
    tv + cooking_basics + free_street_parking + long_term_stays_allowed +
```

```
    microwave + private_entrance + family_kid_friendly + internet +
    first_aid_kit + fire_extinguisher + pets_allowed + smoking_allowed +
    bathtub + lock_on_bedroom_door+ zipcode, data = listings_price_df))
#Multiple R-squared:  0.8928,    Adjusted R-squared:  0.7619

#Removing insignificant variables
summary(lm(price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + bathrooms + beds +
    free_street_parking + long_term_stays_allowed +
    family_kid_friendly + zipcode, data = listings_price_df))
#Multiple R-squared:  0.891,    Adjusted R-squared:  0.7594
```

```
#Removing insignificant variables
summary(lm(price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + free_street_parking +
    family_kid_friendly +
    smoking_allowed + zipcode, data = listings_price_df))
#Multiple R-squared:  0.8895,    Adjusted R-squared:  0.7563
#There is not any significant effect on removing variables
```

```
#Calculating diversity in the descriptions
diversity_property <- diversity(listings_price_df$description, listings_price_df$property_type)

#Visualizing diversit for better understanding
plot(diversity_property)
```

The result of diversity is not important and the measures it calculate would not helo in description understanding and word count is also been analysed before with readability.

Part B

```
nrc_dictionary <- tidytext::get_sentiments("nrc")
bing_dictionary <- tidytext::get_sentiments("bing")
afinn_dictionary <- tidytext::get_sentiments("afinn")


#Download Udpipe model of English
#Load the model
langmodel_download <- udpipe::udpipe_download_model("english")
langmodel <- udpipe::udpipe_load_model("english-ewt-ud-2.5-191206.udpipe")
```

Sentiment on Description

```
#Reading all the files from the local machine
listings_price_df <- readRDS(file = "listings_price_clean.rds")
reviews_df <- readRDS(file = "reviews_clean.rds")
combined_df <- readRDS(file = "combined.rds")
```

```
# #Clean the description from symbols and other languages
#  #cld library missed some descriptions in other language
# listings_price_df$description_new <- iconv(listings_price_df$description_new)
# listings_price_df$description_new <- str_replace_all(listings_price_df$description_new, "[^A-Za-z0-9
#
```

```
# #Use annotate function to extract everything from description
# piped_data <- udpipe::udpipe_annotate(langmodel,
# listings_price_df$description_new,
# parallel.cores = 8,
# trace = 1000)
#
# #Make list as a dataframe
# piped_data <- as.data.frame(piped_data)
#
# #Save the data to local machine
# saveRDS(piped_data, "piped_data.RDS")

#Read annotoated result from loal machine
piped_data <- readRDS(file = "piped_data.RDS")
```

```
#Creating document ID column for joining results from udpipe
listings_price_df <- listings_price_df %>% mutate(doc_id = paste0("doc", row_number()))

#Filter the words which are adverb and adjective because sentiment will be found on them
lematized <- piped_data %>% filter(upos %in% c("ADJ",
                                               "ADV",
                                               "VERB")) %>%
                            select(c("doc_id","lemma")) %>%
                            na.omit()


#Identification the incorrect spellings
lematized <- lematized %>% mutate(spell_check = hunspell::hunspell_check(lematized$lemma))

#Filter the incorrect value to review them
lematized_filter <- lematized %>% filter(spell_check == F) %>% group_by(lemma) %>% count(lemma, sort =

#Replacing words for better sentiments analysis as they are related to sentiment
lematized$lemma <- gsub("Cosy", "cosy", lematized$lemma)
lematized$lemma <- gsub("onest", "honest", lematized$lemma)
lematized$lemma <- gsub("Lovly", "lovely", lematized$lemma)
lematized$lemma <- gsub("confortable", "comfortable", lematized$lemma)

#Update column name to word for dictionary
lematized <- lematized %>% rename(word = lemma)
```

```
#Creating dataframe for price and doc_id
listings_token <- listings_price_df %>% select(c("doc_id", "id")) %>% mutate(index = row_number())

#Joining the data with lemmatized data
listings_token <- listings_token %>% inner_join(lematized, by = ("doc_id" = "doc_id"))

#Removing doc_id variable and spell variable
listings_token$doc_id <- NULL
listings_token$spell_check <- NULL
```

```
#Creating visualization
top_frequency <- listings_token %>% select(word) %>% group_by(word) %>% summarise(frequency = n()) %>% a
```

```
#Creating visualisation for results
wordcloud(words = top_frequency$word, freq = top_frequency$frequency, max.words = 100, colors = "#AD1DA
```

```
#NRC sentiment dictionary
#Joining data with the dictionary
listings_token_nrc <- listings_token %>% inner_join(nrc_dictionary)

#Counting sentiments in each listing
description_nrc <- listings_token_nrc %>%
  group_by(id, index, sentiment)%>%
summarise(count=n())

#Visualise the data
ggplot(description_nrc, aes(x=sentiment, y=count, fill=sentiment)) +
      geom_bar(stat="identity") + coord_polar() +
      theme( axis.text.y = element_blank()) +
      labs(x="", y="", title="Description sentiments")

#Pivoting the sentiment for each listings
description_nrc <- description_nrc %>%
  pivot_wider(names_from=sentiment, values_from=count, values_fill = 0L) %>% mutate(nrc_sentiment = (pos
```

```
#Joining data with main dataframe
listings_price_df <- listings_price_df %>% inner_join(description_nrc, by = "id", suffix =c("","nrc"))

#Removing index column
listings_price_df$index <- NULL
```

```
#AFINN sentiment dictionary
#Joining data with the dictionary
listings_token_afinn <- listings_token %>%
  inner_join(afinn_dictionary)

#Counting sentiments in each listing
description_afinn <- listings_token_afinn %>%
  group_by(id, index)%>%
summarise(afinn_sentiment=sum(value))

#Visualise the data
ggplot(description_afinn, aes(afinn_sentiment)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(title = "Sentiment Score of Afinn", x = "Distribution", y = "Frequency")
```

```
#Joining data with main dataframe
listings_price_df <- listings_price_df %>% inner_join(description_afinn, by = "id", suffix =c("","afinn

#Removing index column
listings_price_df$index <- NULL
```

```
#Bing sentiment dictionary
#Joining data with the dictionary
listings_token_bing <- listings_token %>%
  inner_join(bing_dictionary)

#Counting sentiments in each listing
```

```
description_bing <- listings_token_bing %>%
  group_by(id, index, sentiment)%>%
summarise(count=n())

#Pivot the dataframe for better understanding and joining with main data
description_bing <- description_bing %>%
  pivot_wider(names_from=sentiment, values_from=count, values_fill = 0L) %>%
  mutate(bing_sentiment = (positive-negative)/(positive+negative)) #polarity variable

#Visualise the data
ggplot(description_bing, aes(bing_sentiment)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(title = "Sentiment Score of Bing", x = "Distribution", y = "Frequency")

listings_token_bing %>%
  group_by(word, sentiment)%>%
  summarise(count=n()) %>%
  ungroup() %>%
  group_by(sentiment) %>%
  slice_max(count, n = 10) %>%
  ungroup() %>%
  mutate(word = reorder(word, count)) %>%
  ggplot(aes(count, word, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(x = "Contribution to sentiment",
       y = NULL)
```

```
#Joining data with main dataframe
listings_price_df <- listings_price_df %>% inner_join(description_bing, by = "id", suffix =c("","_bing")

#Removing index column
listings_price_df$index <- NULL
```

```
# Create dataframes with selected variables of each dictionary
# Create a column with dictionary name
results_bing <- description_bing %>% select(c("index", "bing_sentiment")) %>% mutate(sentiment = "bing")
results_nrc <- description_nrc %>% select(c("index", "nrc_sentiment")) %>% mutate(sentiment = "nrc") %>
results_afinn <- description_afinn %>% select(c("index", "afinn_sentiment")) %>% mutate(sentiment = "af

#Combine all the dataframes in rows
#This will give us a dataframe for visuals
all_results <- results_bing %>% bind_rows(results_nrc) %>% bind_rows(results_afinn)
```

```
#Visualize results for each description by different sentiment dictionary
ggplot(all_results, aes(x=index,y=score,fill=sentiment)) +
  geom_bar(stat="identity") +
  facet_wrap(~sentiment,ncol=1,scales="free_y")

#Visualize results by room type with each dictionary
all_results <- all_results %>% inner_join(listings_price_df, by = "id") %>% select(c(index, room_type,

#Visualize the sentiments with room type
ggplot(all_results, aes(x=index,y=score,fill=sentiment)) +
```

```
  geom_bar(stat="identity") +
  labs(title = "Sentiment through Dictionaries", x = "Index", y = "Score", fill = "Dictionary") +
  facet_grid(room_type~sentiment,scales="free_y")
```

```
#Creating a variable in main dataframe of listing
#Variables with count of exclamation mark
listings_price_df$expressive_features <- str_count(listings_price_df$description, pattern = "!")

#Visualize the variable
ggplot(listings_price_df, aes(expressive_features)) +
  geom_histogram(fill = "lightblue", color = "black") +
  facet_grid(~room_type) +
  labs(title = "Number of Exclamation Marks!", x = "Numbers", y = "Frequency")

#Variable with count of capital words
listings_price_df$capital_description <- str_count(listings_price_df$description, pattern = "[A-Z]")

#Visualize the variable
ggplot(listings_price_df, aes(capital_description)) +
  geom_histogram(fill = "lightblue", color = "black") +
  facet_grid(~room_type) +
  labs(title = "Number of Capital Words", x = "Numbers", y = "Frequency")
```

```
#Calculate the polarity for description
#Using function too for polarity check
polarity_description <- with(listings_price_df,polarity(description, id))

#Extract the polarity information from the list
polarity_description_all <-polarity_description[["all"]]

#Save the file for quicker process
saveRDS(polarity_description_all, "polarity_description_all.RDS")
```

```
#Read the results of polarity output
polarity_description_all <- readRDS(file = "polarity_description_all.RDS")

#Select useful variables from polarity output
polarity_description_all <- polarity_description_all %>% select(c(id, polarity))

#Join the data with main dataframe of listing
listings_price_df <- listings_price_df %>% inner_join(polarity_description_all, by = "id")
```

```
#Create a dataframe with created variables to test the significance in regression
model_data <- listings_price_df %>% select(c(price_log, anticipation:polarity))

#Creating a for loop for reading individual regression
for(i in 2:length(colnames(model_data))){
  variable_df <- model_data %>%
    select(c("price_log",colnames(model_data[i])))
  print(colnames(model_data[i]))
  model <- lm(price_log ~ ., data = variable_df)
  print(summary(model))
}
```

```
#Taken the significant variables
#Add them with the base model's variables
model <- lm(price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + free_street_parking +
    family_kid_friendly +
    smoking_allowed + anticipation + joy + positive + sadness + surprise + capital_description + zipcod

#Checking step-forward method for variables reduction
MASS::stepAIC(model)
```

```
#Testing the recommended model
summary(lm(price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + free_street_parking + family_kid_friendly + smoking_allowed +
    anticipation + positive + zipcode, data = listings_price_df))
#Multiple R-squared:  0.8934,    Adjusted R-squared:  0.7529
```

```
#Removing insignificant variables from the model
summary(lm(formula = price_log ~ property_type + room_type + log(accommodates) +
    bedrooms + free_street_parking + family_kid_friendly + zipcode, data = listings_price_df))
#Multiple R-squared:  0.8934,    Adjusted R-squared:  0.7529

#Taking two most significant variables in the model only
#Interacting the variables to check the cross over
summary(lm(price_log ~ log(accommodates) * zipcode, data = listings_price_df))
#Multiple R-squared:  0.9172,    Adjusted R-squared:  0.7577
#Taking this model on the basis of principle of parsimony
```

Rating Prediction

```
#read the data
variables_rating <- readRDS("variables_rating.RDS")
sentiment <- readRDS("sentiment.RDS")
tokens_all_comments <- readRDS("token_all_comments.RDS")
reviews_df <- readRDS("reviews_clean.RDS")
```

```
#Add review_scores_rating as dependent variable

listing_id_rating <- listings_review_df %>% select(id, review_scores_rating)

#readability <- readability %>% rename(listing_id = id)
variables_rating <- left_join(variables_rating,listing_id_rating,by = c("listing_id"="id"))
# relocated so that we cab easily run our for loop
variables_rating <- variables_rating %>% relocate(review_scores_rating, .before = listing_id)
```

```
# converted price into a numerical variable
listing_id_price <- variables_rating %>%
  select(listing_id,price) %>%
  mutate(price = as.numeric(gsub("\\$","",price)))

#removing NA value
variables_rating <- na.omit(variables_rating)

# removing irrelavant variables
a <- c("X1","avg_sentence","rating_category")
```

```
variables_rating <- variables_rating[,!names(variables_rating) %in% a]
```

Regression 1

```
# for loop to try out all the variables and see which ones are significant

(variables_rating$review_scores_rating)
for(i in 2:length(colnames(variables_rating))){
  variable_df <- variables_rating%>%
    select(c("review_scores_rating",colnames(variables_rating[i])))
  print(colnames(variables_rating[i]))
  model <- lm(review_scores_rating ~., data = variable_df)
  print(summary(model))
}
```

```
#created model with all the significant variables
modelinitial <- lm(review_scores_rating~wordcount_comments+formality_comments+diversity_comments+wordcou

summary(modelinitial)
MASS::stepAIC(modelinitial)
#conducted step AIC on the inital model which had the significant variables. Step AIC gave us the best
```

```
# model based on best results from step AIC
summary(lm(formula= review_scores_rating ~ wordcount_comments + formality_comments +
    diversity_comments + wordcount_description + avg_host_name +
    cooking_basics + hair_dryer + host_greets_you + microwave +
    oven + private_entrance + family_kid_friendly + patio_or_balcony +
    free_parking_on_premises + lock_on_bedroom_door + dishwasher +
    shower_gel + host_is_superhost + gender + accommodates +
    is_location_exact + avg_capital + avg_count_excamation,
    data = variables_rating))

#used the same model from Step AIC but  removing insignificant variables  oven,family_kid_friendly,show
summary(lm(formula= review_scores_rating ~ wordcount_comments + formality_comments +
    diversity_comments + wordcount_description + avg_host_name +
    cooking_basics + hair_dryer + host_greets_you + microwave +
    private_entrance + patio_or_balcony +
    free_parking_on_premises + lock_on_bedroom_door + dishwasher +
    host_is_superhost + gender + accommodates +
    avg_capital + avg_count_excamation,
    data = variables_rating))

#used the same model above but  removing insignificant variables cooking_basics
summary(lm(formula= review_scores_rating ~ wordcount_comments + formality_comments +
    diversity_comments + wordcount_description + avg_host_name +
    hair_dryer + host_greets_you + microwave +
    private_entrance + patio_or_balcony +
    free_parking_on_premises + lock_on_bedroom_door + dishwasher +
    host_is_superhost + gender + accommodates +
    avg_capital + avg_count_excamation,
    data = variables_rating))
```

```
# combining the variables with the sentiment scores of dictionaries because now we will add sentiment s
#sentiment <- sentiment %>% rename(id = listing_id)
```

```r
variables_rating <- left_join(variables_rating,sentiment,by =c("listing_id"= "id"))
```

```r
#trying out different sentiment dictionary results into out model

#model with sentiment bing
summary(lm(formula = review_scores_rating ~ wordcount_comments + formality_comments +
    diversity_comments + wordcount_description + avg_host_name +
    hair_dryer + host_greets_you + microwave +
    private_entrance + patio_or_balcony +
    free_parking_on_premises + lock_on_bedroom_door + dishwasher +
    host_is_superhost + gender + accommodates +
    avg_capital + avg_count_excamation+bing_liu_sentiment,
    data = variables_rating))

#model with  sentiment nrc
summary(lm(formula = review_scores_rating ~ wordcount_comments + formality_comments +
    diversity_comments + wordcount_description + avg_host_name +
    hair_dryer + host_greets_you + microwave +
    private_entrance + patio_or_balcony +
    free_parking_on_premises + lock_on_bedroom_door + dishwasher +
    host_is_superhost + gender + accommodates +
    avg_capital + avg_count_excamation+sentiment_nrc,
    data = variables_rating))

#model with sentiment affin
summary(lm(formula = review_scores_rating ~ wordcount_comments + formality_comments +
    diversity_comments + wordcount_description + avg_host_name +
    hair_dryer + host_greets_you + microwave +
    private_entrance + patio_or_balcony +
    free_parking_on_premises + lock_on_bedroom_door + dishwasher +
    host_is_superhost + gender + accommodates +
    avg_capital + avg_count_excamation+sentiment_affin,
    data = variables_rating))

#model with polarity description
summary(lm(formula = review_scores_rating ~ wordcount_comments + formality_comments +
    diversity_comments + wordcount_description + avg_host_name +
    hair_dryer + host_greets_you + microwave +
    private_entrance + patio_or_balcony +
    free_parking_on_premises + lock_on_bedroom_door + dishwasher +
    host_is_superhost + gender + accommodates +
    avg_capital + avg_count_excamation+polarity_description,
    data = variables_rating))
```

```r
#By comparing the adjusted R squared, regression with bing dictionary results has the best performance.
summary(lm(formula = review_scores_rating ~ wordcount_comments + formality_comments +
    diversity_comments  + avg_host_name +
    hair_dryer+ private_entrance + patio_or_balcony +
    lock_on_bedroom_door + dishwasher +
    host_is_superhost + avg_count_excamation+bing_liu_sentiment,
    data = variables_rating))

#the adjusted R squared has decrease slightly
#Multiple Adj R sq 0.701 and R sq 0.6979
```

Regression 2

The most appropriate regression from above has adjusted R square of 0.6979, which is acceptable to predict the rating scores. To improve the regression, new variables are added.

```
# review score accuracy, review score communication, review score value and review score location is ad
reveiw_score <- listings_review_df %>% select(id, review_scores_accuracy,review_scores_checkin,review_s

variables_rating <- left_join(variables_rating,reveiw_score,by= c("listing_id"="id"))
variables_rating <- variables_rating%>%na.omit()

# relocated the position so that we can run the for loop without the independent variable
variables_rating <- variables_rating %>% relocate(review_scores_rating, .before = id)

# for loop to check all the dependant variables against the independent variable and see which ones are
for(i in 2:length(colnames(variables_rating))){
  variable_df <- variables_rating%>%
    select(c("review_scores_rating",colnames(variables_rating[i])))
  print(colnames(variables_rating[i]))
  model <- lm(review_scores_rating ~., data = variable_df)
  print(summary(model))
}
```

```
#variables_rating$price.y <- NULL
#variables_rating<- variables_rating%>%rename(price=price.x)
# initial model with all the significant variables that effect review score rating

modelinitial <- lm(review_scores_rating~avg_wordcount+avg_syllable+avg_Fk_ease+wordcount_comments+forma



summary(modelinitial)
#install.packages("MASS")
MASS::stepAIC(modelinitial)
```

```
## After Step AIC using the best combination of variables
summary(lm(formula = review_scores_rating ~ avg_wordcount + avg_syllable +
    avg_Fk_ease + formality_comments + avg_host_name + bed_linens +
    free_street_parking + hair_dryer + hot_water + private_entrance +
    refrigerator + shampoo + patio_or_balcony + dishwasher +
    ethernet_connection + host_is_superhost + bedrooms + bathrooms +
    accommodates + avg_capital + avg_count_excamation + review_scores_accuracy +
    review_scores_checkin + review_scores_communication + review_scores_value +
    review_scores_location, data = variables_rating))

#removing insignificant variables:avg_Fk_ease,bed_linens,free_street_parking,hair_dryer,ethernet_connec
summary(lm(formula = review_scores_rating ~ avg_wordcount + avg_syllable +
    formality_comments + avg_host_name  +
    hot_water + private_entrance +
    refrigerator + shampoo + patio_or_balcony + dishwasher +
    host_is_superhost + bedrooms + bathrooms +
    avg_capital + avg_count_excamation + review_scores_accuracy +
    review_scores_checkin + review_scores_communication + review_scores_value +
    review_scores_location, data = variables_rating))

#further removing more insignificant variables: bedrooms
```

```
summary(lm(formula = review_scores_rating ~ avg_wordcount + avg_syllable +
    formality_comments + avg_host_name  + hot_water + private_entrance +
    refrigerator + shampoo + patio_or_balcony + dishwasher +
    host_is_superhost +bathrooms + avg_capital + avg_count_excamation + review_scores_accuracy +
    review_scores_checkin + review_scores_communication + review_scores_value +
    review_scores_location, data = variables_rating))
```

```
#Now testing out all the dictionary sentiment scores on the model. Will later choose which one is the b
#model with sentiment bing
summary(lm(formula = review_scores_rating ~ avg_wordcount + avg_syllable +
    formality_comments + avg_host_name  + hot_water + private_entrance +
    refrigerator + shampoo + patio_or_balcony + dishwasher +
    host_is_superhost +bathrooms + avg_capital + avg_count_excamation + review_scores_accuracy +
    review_scores_checkin + review_scores_communication + review_scores_value +
    review_scores_location+bing_liu_sentiment, data = variables_rating))

#model with  sentiment nrc
summary(lm(formula = review_scores_rating ~ avg_wordcount + avg_syllable +
    formality_comments + avg_host_name  + hot_water + private_entrance +
    refrigerator + shampoo + patio_or_balcony + dishwasher +
    host_is_superhost +bathrooms + avg_capital + avg_count_excamation + review_scores_accuracy +
    review_scores_checkin + review_scores_communication + review_scores_value +
    review_scores_location+sentiment_nrc,
    data = variables_rating))

#model with sentiment affin
summary(lm(formula =  review_scores_rating ~ avg_wordcount + avg_syllable +
    formality_comments + avg_host_name  + hot_water + private_entrance +
    refrigerator + shampoo + patio_or_balcony + dishwasher +
    host_is_superhost +bathrooms + avg_capital + avg_count_excamation + review_scores_accuracy +
    review_scores_checkin + review_scores_communication + review_scores_value +
    review_scores_location+sentiment_affin,
    data = variables_rating))

#model with polarity description
summary( lm(formula = review_scores_rating ~ avg_wordcount + avg_syllable +
    formality_comments + avg_host_name  + hot_water + private_entrance +
    refrigerator + shampoo + patio_or_balcony + dishwasher +
    host_is_superhost +bathrooms + avg_capital + avg_count_excamation + review_scores_accuracy +
    review_scores_checkin + review_scores_communication + review_scores_value +
    review_scores_location+polarity_description, data = variables_rating))

# sentiment scores from bing liu dictionary gave the best results so we continued with that model
summary(lm(formula = review_scores_rating ~ avg_wordcount + avg_syllable +
    formality_comments + avg_host_name  + hot_water + private_entrance +
    refrigerator + shampoo + patio_or_balcony + dishwasher +
    host_is_superhost +bathrooms + avg_capital + avg_count_excamation + review_scores_accuracy +
    review_scores_checkin + review_scores_communication + review_scores_value +
    review_scores_location+bing_liu_sentiment, data = variables_rating))
```

```
# since there were still some insigificant variables those were removed. bathrooms was removed due to i
summary(lm(formula = review_scores_rating ~ avg_wordcount + avg_syllable +
    formality_comments + avg_host_name  + hot_water + private_entrance +
    refrigerator + shampoo + patio_or_balcony + dishwasher +
```

```
    host_is_superhost + avg_capital + avg_count_excamation + review_scores_accuracy +
    review_scores_checkin + review_scores_communication + review_scores_value +
    review_scores_location+bing_liu_sentiment, data = variables_rating))
#Multiple Adjusted R sq 0.8137 and R square 0.8169
```

Part C

```
all_data_df <- readRDS("combined.RDS")
all_data_df <- all_data_df %>%
  rename(review_id = idreview_)
```

```
# we want to build a topic model
# where the topic is dependent on the review score
# and the price
data_for_stm <- all_data_df %>%
  select(review_id, comments, review_scores_rating, id,
  price, date) %>%
  mutate(price = as.numeric(gsub("\\$|,","",price)))
```

```
# loading udpipe english model
language <- udpipe_download_model(language="english",overwrite = F)
ud_model <- udpipe_load_model("english-ewt-ud-2.5-191206.udpipe")
#creating a dataframe for annotated reviews
annotated_reviews_all <- data.frame()
```

```
# parallelization

split_size <- 10000

for_pos_list <- split(all_data_df,
rep(1:ceiling(nrow(all_data_df)/split_size), each=split_size,  length.out=nrow(all_data_df)))

for(i in 1:length(for_pos_list)){
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)
}
```

```
# Prepare metadata for the STM model
annotated_reviews_all$review_id <- as.integer(annotated_reviews_all$review_id)

data_for_stm <- annotated_reviews_all %>%
  left_join(data_for_stm) %>%
  na.omit()
```

```r
# Should be done from the very beginning but
# make sure price is parsed as numeric
data_for_stm$price <- gsub("\\$|,","",data_for_stm$price)
data_for_stm$price <- as.numeric(data_for_stm$price)

# saveRDS(data_for_stm, "data_for_stm.rds")

data_for_stm <- readRDS("data_for_stm.rds")
```

```r
# load the stm package
library(stm)

# First we engage with the
# textProcessor function
# Notice that you should augment the custom
# stopwords

# gsub for lovly, neighborhood and cosy
data_for_stm$annotated_comments <- gsub("lovly", "lovely", data_for_stm$annotated_comments, ignore.case
data_for_stm$annotated_comments <- gsub("cosy", "cozy", data_for_stm$annotated_comments, ignore.case =
data_for_stm$annotated_comments <- gsub("neighborhood", "neighbourhood", data_for_stm$annotated_comments

processed <- textProcessor(data_for_stm$annotated_comments,
metadata = data_for_stm,
customstopwords = c("airbnb","manchester",
"apartment", Fry_1000),
stem = F)

# Keep only those words that appear
# on 1% of the corpus
# considering the size of the corpus
threshold <- round(1/100 * length(processed$documents),0)

out <- prepDocuments(processed$documents,
processed$vocab,
processed$meta,
lower.thresh = threshold)

# saveRDS(out, "out.rds")
out <- readRDS("out.rds")
# look at out$vocab to inspect the vocab
# spotted a few spelling errors which we corrected
```

```r
#running the stm algorithm with K set to 0 to
#get the right number of topics
airbnbfit_0 <- stm(documents = out$documents,
              vocab = out$vocab,
              K = 0,
              prevalence =~ price+review_scores_rating,
              max.em.its = 75,
              data = out$meta,
              reportevery=3,
              # gamma.prior = "L1",
```

```
              sigma.prior = 0.7,
              seed = 123,
              init.type = "Spectral")
#Results in 34 topics
```

```
#searching for optimal k
numtopics <- searchK(out$documents, out$vocab, K = c(15, 20, 25, 30, 35), prevalence=~ price + review_s

#   K   exclus    semcoh   heldout residual    bound   lbound em.its
# 1 10 9.490771 -153.3682 -3.992954  48.9021 -1798308 -1798293    106
# 2 15  9.51254 -149.8583 -3.986032 13246.46 -1795714 -1795686    122
# 3 20 9.516951 -148.0616 -3.982698 125325.6 -1795754 -1795712    152
# 4 25  9.52385 -145.7082 -3.979536 758353.4 -1794214 -1794156    191
# 5 29 9.525607 -140.8816 -3.969915 95147221 -1791493 -1791422    206
```

```
plot(numtopics)
```

```
#let's look at exclusivity vs semantic coherence
numtopics2$results %>%
  select(K, exclus, semcoh) %>%
  filter(K %in% c(15, 20, 25, 30, 35)) %>%
  unnest() %>%
  mutate(K = as.factor(K)) %>%
  ggplot(aes(semcoh, exclus, color = K)) +
  geom_point(size = 2, alpha = 0.7) +
  labs(x = "Semantic Coherence",
       y = "Exclusivity",
       title = "Comparing exclusivity and semantic coherence",
       subtitle = "K = 35 has both high exclusivity and semantic coherence")
```

```
# after running multiple models with various values of K
# we determined to go with K = 15 as it has the highest variance explained
# and topics are distinct enough to label

airbnbfit <-  readRDS("airbnbfit_5.rds")

airbnbfit <- stm(documents = out$documents,
              vocab = out$vocab,
              K = 15,
              prevalence =~ price+review_scores_rating,
              max.em.its = 75,
              data = out$meta,
              reportevery=3,
              # gamma.prior = "L1",
              sigma.prior = 0.7,
              seed = 123,
              init.type = "Spectral")
```

```
#plot to see the percentage of topics in the corpus
plot(airbnbfit, type = "summary", text.cex = 0.8, xlim = c(0, 0.15))
```

Extracting the theta matrix

```r
#extract the theta matrix
convergence_theta <- as.data.frame(airbnbfit$theta)
colnames(convergence_theta)<-paste0("topic_", 1:15)
```

```r
#assigning topic summary, proportions and labels to variables
topic_summary <- summary(airbnbfit)
topic_proportions <- colMeans(airbnbfit$theta)
topic_labels <- paste0("topic_",1:15)
```

```r
#wordcloud for topics
stm::cloud(airbnbfit)
```

```r
td_beta <- tidy(airbnbfit)
td_beta
```

```r
top_terms <-
  td_beta %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  arrange(topic, desc(beta))

plot <- top_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()
```

```r
td_gamma <- tidy(airbnbfit, matrix = "gamma",
                 document_names = rownames(out$documents))

td_gamma
```

```r
library(ggthemes)
library(scales)

top_terms <- td_beta %>%
  arrange(beta) %>%
  group_by(topic) %>%
  top_n(7, beta) %>%
  arrange(-beta) %>%
  select(topic, term) %>%
  summarise(terms = list(term)) %>%
  mutate(terms = map(terms, paste, collapse = ", ")) %>%
  unnest()

gamma_terms <- td_gamma %>%
  group_by(topic) %>%
  summarise(gamma = mean(gamma)) %>%
  arrange(desc(gamma)) %>%
  left_join(top_terms, by = "topic") %>%
  mutate(topic = paste0("Topic ", topic),
         topic = reorder(topic, gamma))
```

```
gamma_terms %>%
  top_n(20, gamma) %>%
  ggplot(aes(topic, gamma, label = terms, fill = topic)) +
  geom_col(show.legend = FALSE) +
  geom_text(hjust = 0, nudge_y = 0.0005, size = 3,
            family = "IBMPlexSans") +
  coord_flip() +
  scale_y_continuous(expand = c(0,0),
                     limits = c(0, 0.20),
                     labels = percent_format()) +
  theme_tufte(base_family = "IBMPlexSans", ticks = FALSE) +
  theme(plot.title = element_text(size = 16,
                                  family="IBMPlexSans-Bold"),
        plot.subtitle = element_text(size = 13)) +
  labs(x = NULL, y = expression(gamma),
       title = "Topics by prevalence in the Airbnb corpus",
       subtitle = "With the top words that contribute to each topic")
```

```
gamma_terms %>%
  select(topic, gamma, terms) %>%
  knitr::kable(digits = 3,
        col.names = c("Topic", "Expected topic proportion", "Top 7 terms"))
```

```
# Tidying it up
gamma_topics <- td_gamma %>%
pivot_wider(names_from = topic, values_from = gamma)

# Adding column name
colnames(gamma_topics) <- c("document",topic_labels)

#remove the document from the column
gamma_topics <- as.data.frame(gamma_topics)
rownames(gamma_topics) <- gamma_topics$document
gamma_topics$document <- NULL
```

```
#perform PCA
pcav <- FactoMineR::PCA(gamma_topics, graph = FALSE)
factoextra::fviz_pca_var(pcav, alpha.var = .5)
factoextra::fviz_screeplot(pcav, addlabels = TRUE) + theme_classic()
```

```
#As expected there are no negative reviews
hist(all_data_df$review_scores_rating, xlab="Review Score", main="Review Distribution", breaks = 50)
```

```
#Let us also look at the distribution of price
hist(all_data_df$price, xlab="Price", main="Price Distribution", breaks = 50)
```

STM Effect Estimation

```
# Estimate the effects of price and review score
# on topic probability
effects <- estimateEffect(~ as.numeric(review_scores_rating) + as.numeric(price),
                          stmobj = airbnbfit,
                          metadata = out$meta)
```

```r
# Effect of review score on topic
# probability
plot(effects, covariate = "review_scores_rating",
     topics = c(1:15),
     model = airbnbfit,
     method = "difference",
     cov.value1 = "100",
     cov.value2 = "0",
     xlab = "Low Rating ... High Rating",
     xlim = c(-0.6,0.6),
     main = "Marginal change on topic probabilities for low and high rating score",
     custom.labels = topic_labels[c(1:15)],
     labeltype = "custom")
```

```r
# let's add labels to each topic
# and dig into specific topics
# that lead to relatively lower and
# relatively higher ratings

custom_labels = c("1. responsive host ", "2. arrival experience", "3. central location", "4. local acce

# summarizing topic labels below:
label_review <- knitr::kable(custom_labels)
label_review

# The code below picks three topics and
# plots them according to their association with the
# low/high rating variable.


plot(effects, covariate = "review_scores_rating",
     topics = c(12, 14, 10, 13),
     model = airbnbfit,
     method = "difference",
     cov.value1 = "100",
     cov.value2 = "0",
     xlab = "Low Rating ... High Rating",
     xlim = c(-0.6,0.6),
     main = "Marginal change on topic probabilities for low and high rating score",
     custom.labels = custom_labels[c(12, 14, 10, 13)],
     labeltype = "custom")
```

```r
# 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 upper quantile
     # and low quantile of the price
     xlab = "Price",
     # xlim = c(0,800),
     main = custom_labels[i],
     printlegend = FALSE,
     custom.labels = custom_labels[i],
     labeltype = "custom")
}
```

```
# 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])

plot(effects,
     covariate = "price",
     topics = c(1:15),
     model = airbnbfit,
     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,
     labeltype = "custom")

# digging into topics with higher impact
# on low and high price
plot(effects,
     covariate = "price",
     topics = c(9, 6),
     model = airbnbfit,
     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 = custom_labels[c(9, 6)],
     labeltype = "custom")
```

```
# checking correlation between topics
plot(topicCorr(airbnbfit),
      topics = c(1:15),
      vlabels = c(1:15),
      layout = NULL,
      vertex.color = "grey",
      vertex.label.cex = 0.8,
      vertex.size = 24,
      main = "Topic correlation in reviews")
```