ITEC 724 CLASS PROJECT

KIPLIMO NANCY

2024-10-12

Table of contents

Part 1: Importing data and preprocessing

```
#install.packages("tinytex")
```

```
#tinytex::install_tinytex()
```

```
tinytex::is_tinytex()
```

```
[1] TRUE
```

load the required packages

```
library(readr)
library(tidytext)
```

```
Warning: package 'tidytext' was built under R version 4.5.1
```

```r
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
library(tm)
```

```
Warning: package 'tm' was built under R version 4.5.1

Loading required package: NLP
```

```r
library(caret)
```

```
Warning: package 'caret' was built under R version 4.5.1

Loading required package: ggplot2

Attaching package: 'ggplot2'

The following object is masked from 'package:NLP':

    annotate

Loading required package: lattice
```

```r
library(ggplot2)
library(tidyverse)
```

```
Warning: package 'tidyverse' was built under R version 4.5.1

Warning: package 'forcats' was built under R version 4.5.1

Warning: package 'lubridate' was built under R version 4.5.1

-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats   1.0.0     v stringr   1.5.1
v lubridate 1.9.4     v tibble    3.2.1
v purrr     1.0.4     v tidyr     1.3.1

-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x ggplot2::annotate() masks NLP::annotate()
x dplyr::filter()     masks stats::filter()
x dplyr::lag()        masks stats::lag()
x purrr::lift()       masks caret::lift()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```
library(e1071)
```

```
Warning: package 'e1071' was built under R version 4.5.1
```

```
library(wordcloud)
```

```
Warning: package 'wordcloud' was built under R version 4.5.1

Loading required package: RColorBrewer
```

```
library(SnowballC)
library(Matrix)
```

```


Attaching package: 'Matrix'

The following objects are masked from 'package:tidyr':

    expand, pack, unpack
```

```
library(quanteda)
```

```
Warning: package 'quanteda' was built under R version 4.5.1
```

```
Package version: 4.3.1
Unicode version: 15.1
ICU version: 74.1
Parallel computing: 12 of 12 threads used.
See https://quanteda.io for tutorials and examples.

Attaching package: 'quanteda'

The following object is masked from 'package:tm':

    stopwords

The following objects are masked from 'package:NLP':

    meta, meta<-
```

```
library(textcat)
```

```
Warning: package 'textcat' was built under R version 4.5.1
```

```
library(stringr)
library(tinytex)
```

```
Warning: package 'tinytex' was built under R version 4.5.1
```

```
#Import the data and save in an object called Airbnbreviews and Hotelreviews
Airbnbreviews<-read.csv("Airbnb reviews.csv")
Hotelreviews<-read.csv("tripadvisor_hotel_reviews.csv")
```

counts of the observations and visualize

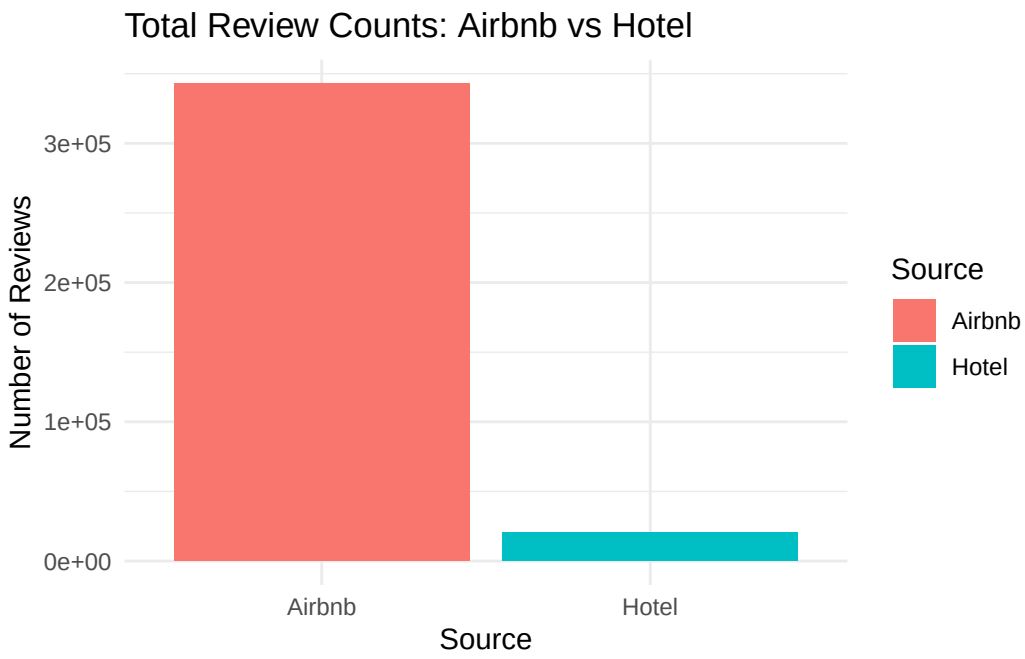
```r
# Count the number of reviews in each dataset
num_airbnb_reviews <- nrow(Airbnbreviews)
num_hotel_reviews <- nrow(Hotelreviews)

# Print counts
print(paste("Number of Airbnb reviews:", num_airbnb_reviews))
```

```
[1] "Number of Airbnb reviews: 342904"
```

```r
print(paste("Number of Hotel reviews:", num_hotel_reviews))
```

```
[1] "Number of Hotel reviews: 20491"
```

```r
# Create a summary data frame for review counts
review_counts <- data.frame(
  Source = c("Airbnb", "Hotel"),
  Count = c(num_airbnb_reviews, num_hotel_reviews)
)
# Create a bar chart to visualize the counts of reviews
ggplot(review_counts, aes(x = Source, y = Count, fill = Source)) +
  geom_bar(stat = "identity") +
  labs(title = "Total Review Counts: Airbnb vs Hotel",
       x = "Source",
       y = "Number of Reviews") +
  theme_minimal()
```



Check the head for both dataset

I am going to explore the datasets differently

Airbnb dataset

```
head(Airbnbreviews)
```

```
  listing_id      id      date reviewer_id reviewer_name
1       2818    1191 3/30/2009       10952           Lam
2     515749 1671407  7/9/2012     2640670       Gregory
3     515749 1715674 7/15/2012     1032804       Michael
4       2818    1771 4/24/2009       12798         Alice
5     515749 1963378 8/12/2012      503786         Brian
6     515749 2073958 8/23/2012     2869021        Nadége

1
2 If you want the authentic Amsterdam houseboat experience, this is it!  It is a great boat,
3
4
```

```
5
6
```

Hotel dataset

```
head(Hotelreviews)
```

```
1
2 ok nothing special charge diamond member hilton decided chain shot 20th anniversary seattle
3
4
5
6
  Rating
1      4
2      2
3      3
4      5
5      5
6      5
```

check for missing data

```
# Check for missing values in Hotelreviews dataset
sum(is.na(Hotelreviews))
```

```
[1] 0
```

```
# Check for missing values in Airbnbreviews dataset
sum(is.na(Airbnbreviews))
```

```
[1] 4
```

Remove missing values

```
# Remove rows with missing values in Airbnbreviews
Airbnbreviews <- Airbnbreviews[!rowSums(is.na(Airbnbreviews)), ]
```

```
#now recheck if missing values are removed
sum(is.na(Airbnbreviews))
```

```
[1] 0
```

check class

```
class(Hotelreviews)
```

```
[1] "data.frame"
```

```
class(Airbnbreviews)
```

```
[1] "data.frame"
```

#Tokenize both datasets

```
# Tokenize Hotelreviews dataset
Hotelreviews <- Hotelreviews %>%
  unnest_tokens(word, Review)
```

recheck again

```
head(Hotelreviews)
```

```
  Rating      word
1      4      nice
2      4     hotel
3      4 expensive
4      4   parking
5      4       got
6      4      good
```

Tokenize airbnb dataset

```
# Tokenize Airbnbreviews dataset
Airbnbreviews <- Airbnbreviews %>%
  unnest_tokens(word, comments)
```

check the heads

```
# check the few rows
head(Airbnbreviews)
```

```
  listing_id   id      date reviewer_id reviewer_name   word
1       2818 1191 3/30/2009       10952           Lam daniel
2       2818 1191 3/30/2009       10952           Lam     is
3       2818 1191 3/30/2009       10952           Lam really
4       2818 1191 3/30/2009       10952           Lam   cool
5       2818 1191 3/30/2009       10952           Lam    the
6       2818 1191 3/30/2009       10952           Lam  place
```

Create sentiment column for hotel dataset

```
# Now create sentiment column for hotel dataset
Hotelreviews <- Hotelreviews %>%
  mutate(sentiment = case_when(
    Rating >= 4 ~ "Positive",
    Rating == 3 ~ "Neutral",
    Rating <= 2 ~ "Negative"
  ))
```

```
# recheck again
head(Hotelreviews)
```

```
  Rating      word sentiment
1      4      nice  Positive
2      4     hotel  Positive
3      4 expensive  Positive
4      4   parking  Positive
5      4       got  Positive
6      4      good  Positive
```

display the created sentiments

```
# Count the number of reviews in each sentiment category
hotel_sentiment_count <- Hotelreviews %>%
  count(sentiment)

# Display the counts
print(hotel_sentiment_count)
```

```
  sentiment       n
1  Negative  392586
2   Neutral  254377
3  Positive 1522315
```

Perform word count for each datasets

word count for Hotel

```
# Count word frequencies for hotel dataset
hotel_word_freq <- Hotelreviews %>%
  count(word, sort = TRUE)
head(hotel_word_freq)
```

```
   word     n
1 hotel 49016
2  room 34669
3   not 31617
4 great 21175
5   n't 18984
6  good 17084
```

```
# Count word frequencies for airbnb Dataset
Airbnb_word_freq <- Airbnbreviews %>%
  count(word, sort = TRUE)
head(Airbnb_word_freq)
```

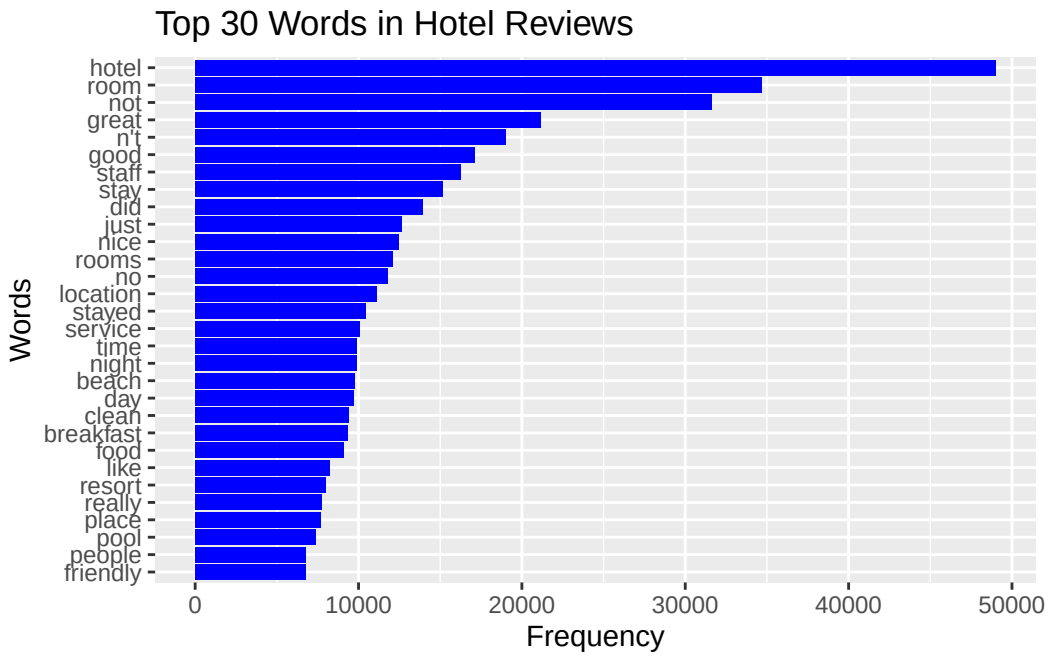
```
  word      n
1  the 661983
```

```
2  and 601385
3    a 413930
4   to 371302
5   is 269675
6  was 269323
```

Visualize both datasets before preprocessing

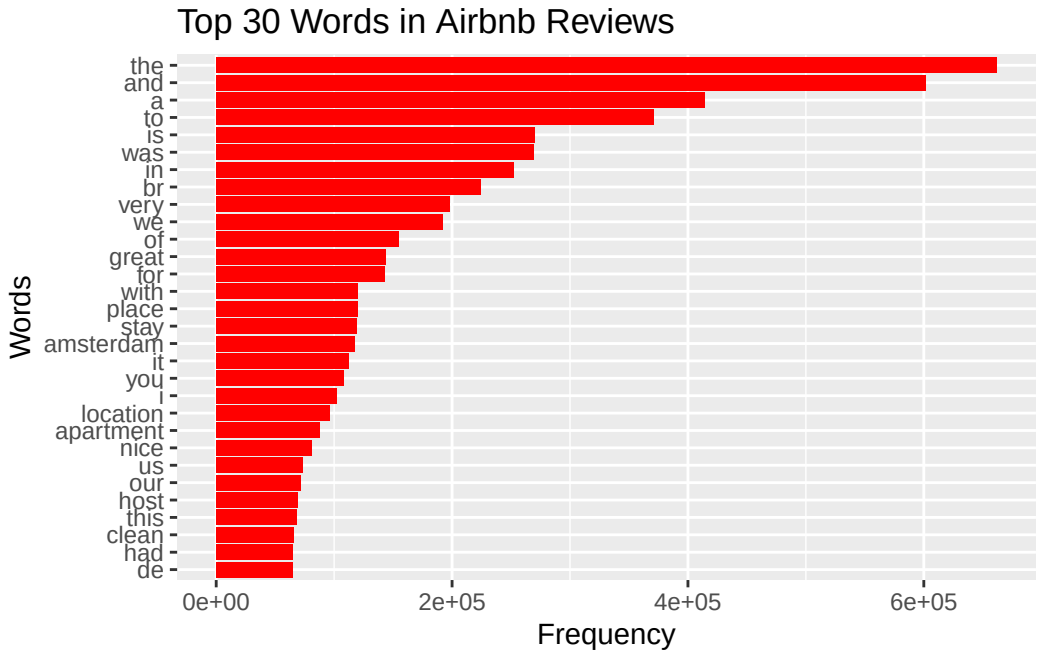
```
# Visualization of the top 30 frequent words in Hotel reviews
hotel_word_freq %>%
  top_n(30) %>%
  ggplot(aes(x = reorder(word, n), y = n)) +
  geom_bar(stat = 'identity', fill = 'blue') +
  coord_flip() +
  labs(title = "Top 30 Words in Hotel Reviews", x = "Words", y = "Frequency")
```

```
Selecting by n
```



```
# Visualization of the top 30 frequent words in Airbnb reviews
Airbnb_word_freq %>%
  top_n(30) %>%
  ggplot(aes(x = reorder(word, n), y = n)) +
  geom_bar(stat = 'identity', fill = 'red') +
  coord_flip() +
  labs(title = "Top 30 Words in Airbnb Reviews", x = "Words", y = "Frequency")
```

```
Selecting by n
```



Preprocess the Airbnb dataset

sentiment analysis for Airbnb Dataset by using Bing lexicon

```
# Load stopwords
data("stop_words")

#Remove stopwords and punctuation from the Airbnbreviews tokenized data
Airbnbreviews <- Airbnbreviews %>%
  anti_join(stop_words, by = "word") %>%
```

```
  filter(!word %in% c(".", ",", "!", "?", "-", "_", ":", ";", "(", ")"))
#Load the Bing sentiment lexicon
bing_lexicon <- get_sentiments("bing")

#Perform sentiment analysis by joining the cleaned data with the Bing lexicon
Airbnb_sentiment <- Airbnbreviews %>%
  inner_join(bing_lexicon, by = "word")
```

```
Warning in inner_join(., bing_lexicon, by = "word"): Detected an unexpected many-to-many rela
i Row 440748 of `x` matches multiple rows in `y`.
i Row 4835 of `y` matches multiple rows in `x`.
i If a many-to-many relationship is expected, set `relationship =
  "many-to-many"` to silence this warning.
```

```
#Summarize sentiment score for each review (assuming you have a review 'id' column)
Airbnb_sentiment_summary <- Airbnb_sentiment %>%
  group_by(id) %>%
  summarise(sentiment_score = sum(sentiment == "positive") - sum(sentiment == "negative"))

# Join the sentiment scores back to the original Airbnb dataset
# Force neutral sentiment when sentiment score is zero
Airbnbreviews <- Airbnbreviews %>%
  left_join(Airbnb_sentiment_summary, by = "id") %>%
  mutate(
    # Classify reviews as Positive, Neutral (when score = 0), or Negative
    sentimentbnb = case_when(
      sentiment_score > 0 ~ "Positive",
      sentiment_score == 0 ~ "Neutral",
      sentiment_score < 0 ~ "Negative"
    ),
    # Force zeros for neutral sentiment if needed
    sentiment_score = ifelse(is.na(sentiment_score), 0, sentiment_score)
  )

# Check the result
head(Airbnbreviews)
```

```
  listing_id   id      date reviewer_id reviewer_name         word
1       2818 1191 3/30/2009       10952           Lam       daniel
2       2818 1191 3/30/2009       10952           Lam         cool
3       2818 1191 3/30/2009       10952           Lam         nice
```

```
4       2818 1191 3/30/2009       10952           Lam        clean
5       2818 1191 3/30/2009       10952           Lam        quiet
6       2818 1191 3/30/2009       10952           Lam neighborhood
  sentiment_score sentimentbnb
1               2     Positive
2               2     Positive
3               2     Positive
4               2     Positive
5               2     Positive
6               2     Positive
```

```
# Count the number of reviews in each sentiment category
airbnb_sentiment_summary <- Airbnbreviews %>%
  count(sentimentbnb)

# View the sentiment distribution
print(airbnb_sentiment_summary)
```

```
  sentimentbnb       n
1     Negative  903341
2      Neutral  273816
3     Positive 5770578
4         <NA>  948431
```

```
# Classify reviews with NA in sentimentbnb as "Neutral" if sentiment_score is 0
Airbnbreviews <- Airbnbreviews %>%
  mutate(sentimentbnb = ifelse(is.na(sentimentbnb) & sentiment_score == 0, "Neutral", sentime

# Check the updated dataset to ensure it reflects the changes
head(Airbnbreviews)
```

```
  listing_id   id      date reviewer_id reviewer_name         word
1       2818 1191 3/30/2009       10952           Lam       daniel
2       2818 1191 3/30/2009       10952           Lam         cool
3       2818 1191 3/30/2009       10952           Lam         nice
4       2818 1191 3/30/2009       10952           Lam        clean
5       2818 1191 3/30/2009       10952           Lam        quiet
6       2818 1191 3/30/2009       10952           Lam neighborhood
  sentiment_score sentimentbnb
1               2     Positive
2               2     Positive
```

```
3               2     Positive
4               2     Positive
5               2     Positive
6               2     Positive
```

```r
# now check again
# Count the number of reviews in each sentiment category
airbnb_sentiment_summary <- Airbnbreviews %>%
  count(sentimentbnb)

# View the sentiment distribution
print(airbnb_sentiment_summary)
```

```
  sentimentbnb       n
1     Negative  903341
2      Neutral 1222247
3     Positive 5770578
```

```r
# Filter the Airbnb dataset to show only Positive reviews
positive_airbnb_reviews <- Airbnbreviews %>%
  filter(sentimentbnb == "Positive")

# Display the first few Positive Airbnb reviews
head(positive_airbnb_reviews)
```

```
  listing_id   id      date reviewer_id reviewer_name         word
1       2818 1191 3/30/2009       10952           Lam       daniel
2       2818 1191 3/30/2009       10952           Lam         cool
3       2818 1191 3/30/2009       10952           Lam         nice
4       2818 1191 3/30/2009       10952           Lam        clean
5       2818 1191 3/30/2009       10952           Lam        quiet
6       2818 1191 3/30/2009       10952           Lam neighborhood
  sentiment_score sentimentbnb
1               2     Positive
2               2     Positive
3               2     Positive
4               2     Positive
5               2     Positive
6               2     Positive
```

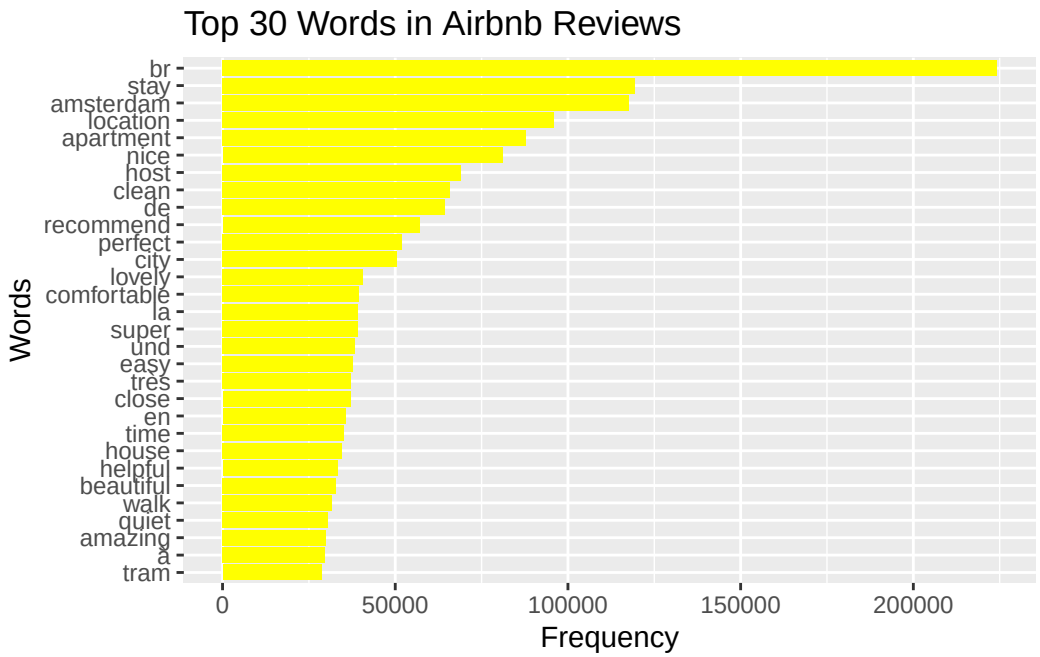
```r
# Load stopwords for Hotelreviews
data("stop_words")

# Remove stopwords from Hotelreviews tokenized data
Hotelreviews <- Hotelreviews %>%
  anti_join(stop_words, by = "word") %>%
  filter(!word %in% c(".", ",", "!", "?", "-", "_", ":", ";", "(", ")"))  # Remove punctuatio

# Check the cleaned Hotelreviews data
head(Hotelreviews)
```

```
  Rating      word sentiment
1      4      nice  Positive
2      4     hotel  Positive
3      4 expensive  Positive
4      4   parking  Positive
5      4      deal  Positive
6      4      stay  Positive
```

Part 2: Visualize term frequency for both datasets

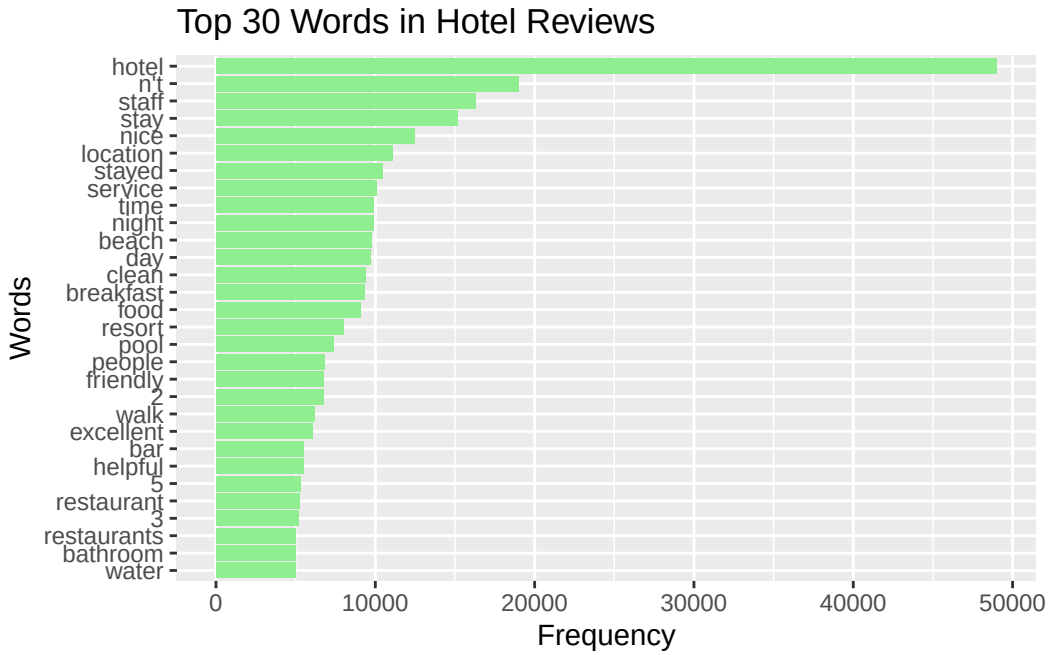
```r
# Count word frequency for Airbnb dataset(bar plot)
airbnb_word_freq <- Airbnbreviews %>%
  count(word, sort = TRUE)

# Visualize the top 30 frequent words in Airbnb reviews
ggplot(airbnb_word_freq %>% top_n(30), aes(x = reorder(word, n), y = n)) +
  geom_bar(stat = "identity", fill = "yellow") +
  coord_flip() +
  labs(title = "Top 30 Words in Airbnb Reviews", x = "Words", y = "Frequency")
```

```
Selecting by n
```




```
# Count word frequency for Hotel dataset
hotel_word_freq <- Hotelreviews %>%
  count(word, sort = TRUE)

# Visualize the top 30 frequent words in Hotel reviews
ggplot(hotel_word_freq %>% top_n(30), aes(x = reorder(word, n), y = n)) +
  geom_bar(stat = "identity", fill = "lightgreen") +
  coord_flip() +
  labs(title = "Top 30 Words in Hotel Reviews", x = "Words", y = "Frequency")
```

```
Selecting by n
```



Part 3: calculating the TF *IDF for both datasets

TF*IDF for Hotel dataset

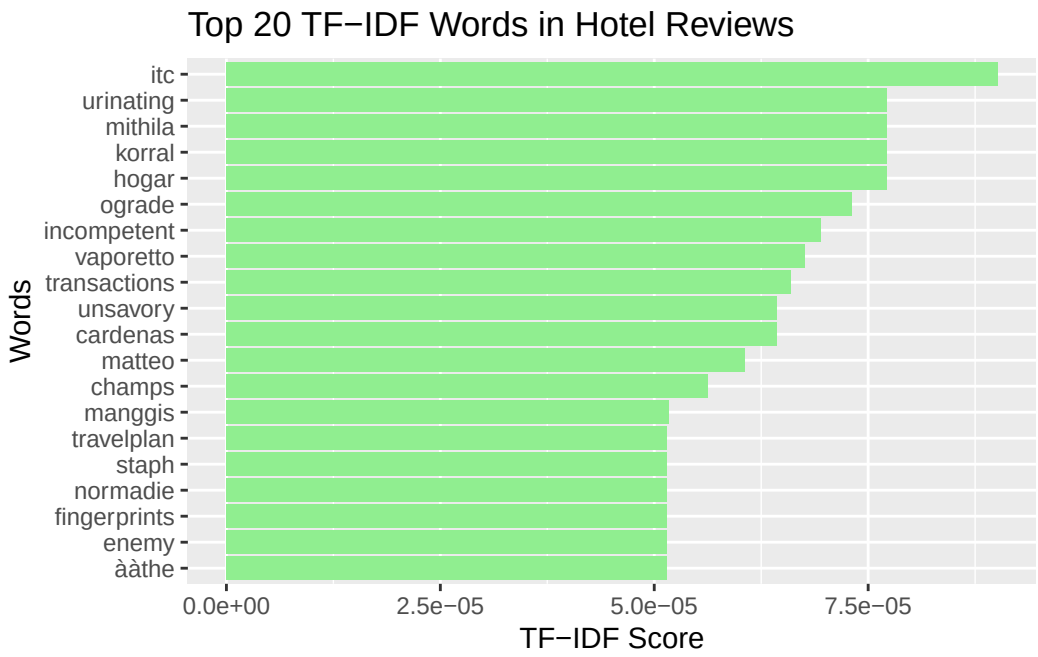
```
# Calculate TF-IDF for Hotel reviews, using 'Rating' to group by review
hotel_tfidf <- Hotelreviews %>%
  count(Rating, word) %>%  # Group by Rating (assuming it's your review identifier)
  bind_tf_idf(word, Rating, n)  # Calculate TF-IDF

# Arrange by the highest TF-IDF values
hotel_tfidf <- hotel_tfidf %>%
  arrange(desc(tf_idf))

# View the top TF-IDF words in Hotel reviews
head(hotel_tfidf)
```

```
  Rating      word n           tf      idf       tf_idf
1      1       itc 7 5.598477e-05 1.609438 9.010401e-05
2      1     hogar 6 4.798695e-05 1.609438 7.723201e-05
3      1    korral 6 4.798695e-05 1.609438 7.723201e-05
```

```
4       1   mithila 6 4.798695e-05 1.609438 7.723201e-05
5       1 urinating 6 4.798695e-05 1.609438 7.723201e-05
6       2    ograde 8 4.544448e-05 1.609438 7.314006e-05
```

```r
# Visualize the top 20 TF-IDF words in Hotel reviews
hotel_tfidf %>%
  top_n(20) %>%
  ggplot(aes(x = reorder(word, tf_idf), y = tf_idf)) +
  geom_bar(stat = "identity", fill = "lightgreen") +
  coord_flip() +
  labs(title = "Top 20 TF-IDF Words in Hotel Reviews", x = "Words", y = "TF-IDF Score")
```

```
Selecting by tf_idf
```



TF*IDF for Airbnb dataset

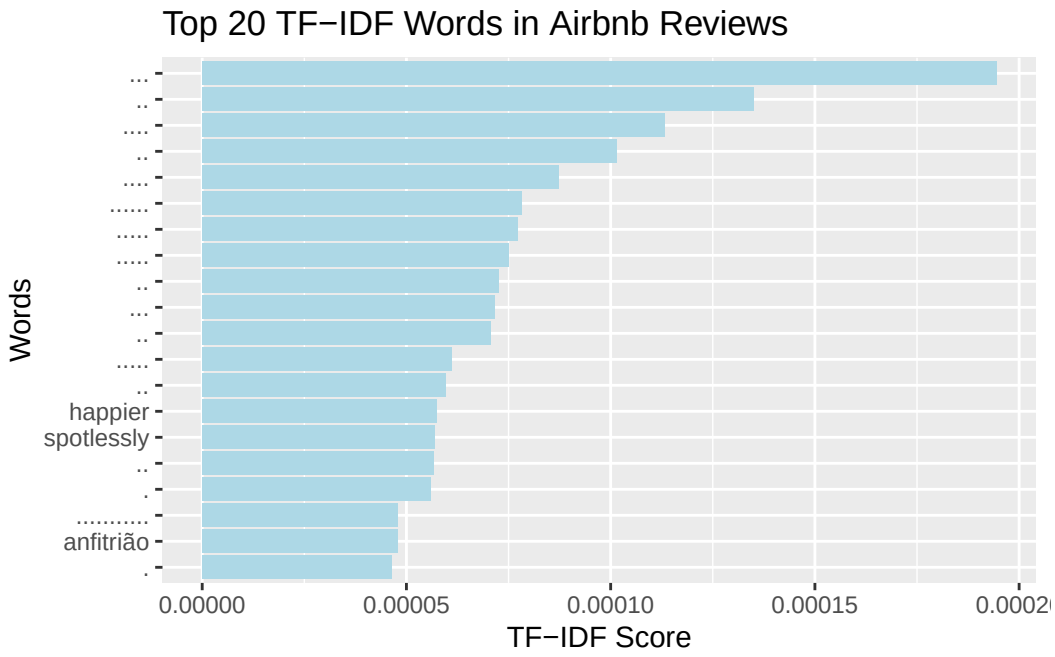
```r
# ## TF*IDF for Airbnb dataset
airbnb_tfidf <- Airbnbreviews %>%
  count(sentimentbnb, word) %>%  # Group by sentimentbnb
  bind_tf_idf(word, sentimentbnb, n)  # Calculate TF-IDF
```

```r
# Arrange by the highest TF-IDF values
airbnb_tfidf <- airbnb_tfidf %>%
  arrange(desc(tf_idf))

# View the top TF-IDF words in Airbnb reviews
head(airbnb_tfidf)
```

```
  sentimentbnb   word   n           tf       idf       tf_idf
1      Neutral       586 0.0004794448 0.4054651 1.943981e-04
2      Neutral       407 0.0003329932 0.4054651 1.350171e-04
3      Neutral       341 0.0002789943 0.4054651 1.131225e-04
4      Neutral       306 0.0002503586 0.4054651 1.015117e-04
5      Neutral       263 0.0002151775 0.4054651 8.724695e-05
6      Neutral       236 0.0001930870 0.4054651 7.829004e-05
```

```r
# Visualize the top 20 TF-IDF words in Airbnb reviews
airbnb_tfidf %>%
  top_n(20) %>%
  ggplot(aes(x = reorder(word, tf_idf), y = tf_idf)) +
  geom_bar(stat = "identity", fill = "lightblue") +
  coord_flip() +
  labs(title = "Top 20 TF-IDF Words in Airbnb Reviews", x = "Words", y = "TF-IDF Score")
```

```
Selecting by tf_idf
```

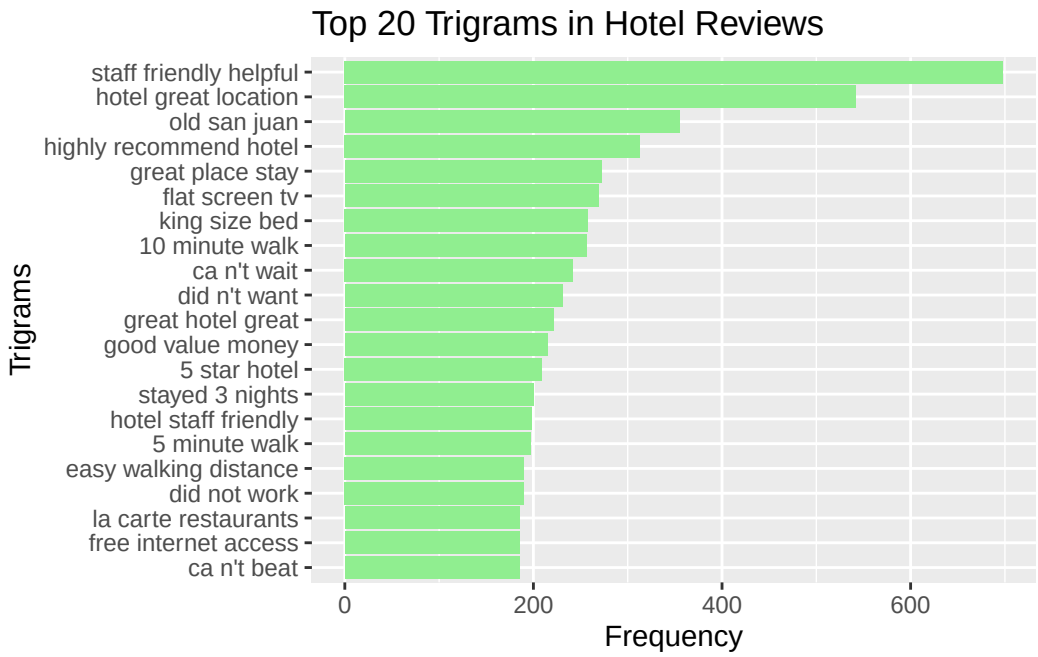


PART 4: NGRAMS FOR BOTH DATASETS

Bigrams visualization for hotel reviews

```
Airbnbreviews2<-read.csv("Airbnb reviews.csv")
Hotelreviews2<-read.csv("tripadvisor_hotel_reviews.csv")
# Tokenize Trigrams for Hotel reviews
hotel_trigrams <- Hotelreviews2 %>%
  unnest_tokens(trigram, Review, token = "ngrams", n = 3)

# Count Trigram frequencies
hotel_trigrams_freq <- hotel_trigrams %>%
  count(trigram, sort = TRUE)

# Visualize the top 20 Trigrams
ggplot(hotel_trigrams_freq %>% top_n(20), aes(x = reorder(trigram, n), y = n)) +
  geom_bar(stat = "identity", fill = "lightgreen") +
  coord_flip() +
  labs(title = "Top 20 Trigrams in Hotel Reviews", x = "Trigrams", y = "Frequency")
```

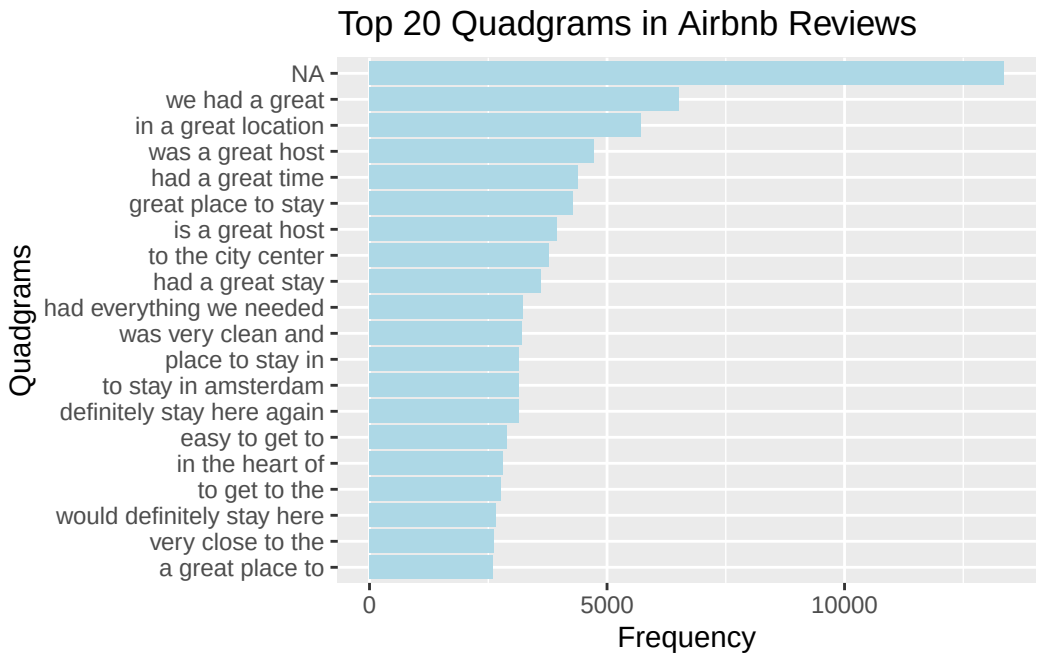
```
Selecting by n
```



Quadgrams visualization for Airbnb reviews

```r
# Tokenize quadgrams for Airbnb reviews
airbnb_quadgrams <- Airbnbreviews2 %>%
  unnest_tokens(quadgram, comments, token = "ngrams", n = 4)

# Count quadgram frequencies
airbnb_quadgrams_freq <- airbnb_quadgrams %>%
  count(quadgram, sort = TRUE)

# Visualize the top 20 quadgrams
ggplot(airbnb_quadgrams_freq %>% top_n(20), aes(x = reorder(quadgram, n), y = n)) +
  geom_bar(stat = "identity", fill = "lightblue") +
  coord_flip() +
  labs(title = "Top 20 Quadgrams in Airbnb Reviews", x = "Quadgrams", y = "Frequency")
```

```
Selecting by n
```

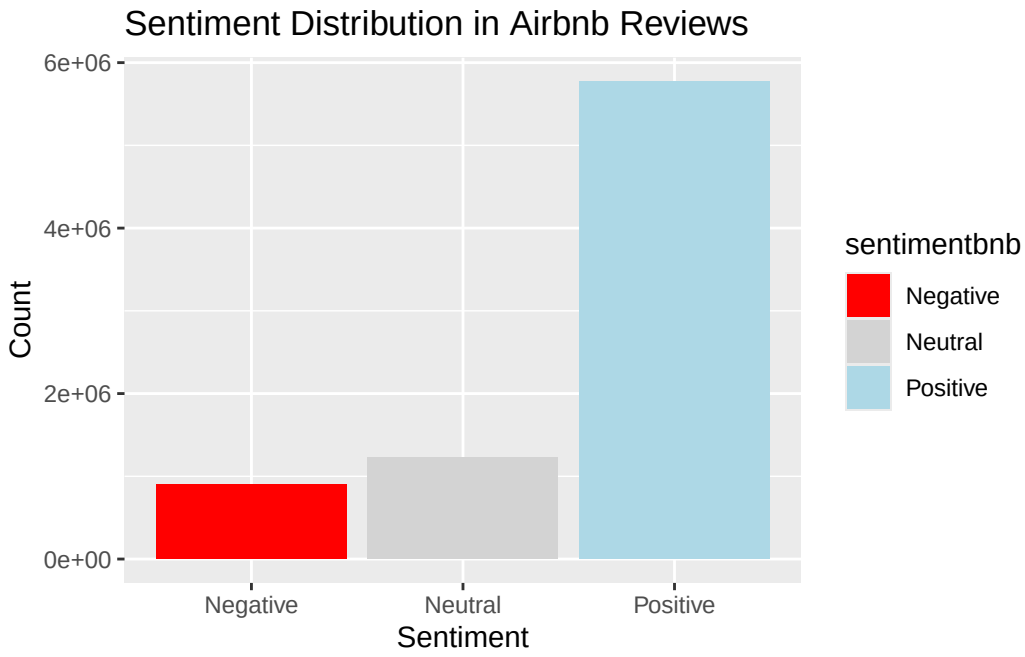


Sentiment distribution for hotel

```
# Visualize sentiment distribution for Hotel reviews with different colors for each sentiment
ggplot(Hotelreviews, aes(x = sentiment, fill = sentiment)) +
  geom_bar() +
  labs(title = "Sentiment Distribution in Hotel Reviews", x = "Sentiment", y = "Count") +
  scale_fill_manual(values = c("Positive" = "lightblue", "Neutral" = "lightgray", "Negative"
```

Sentiment distribution for Airbnb

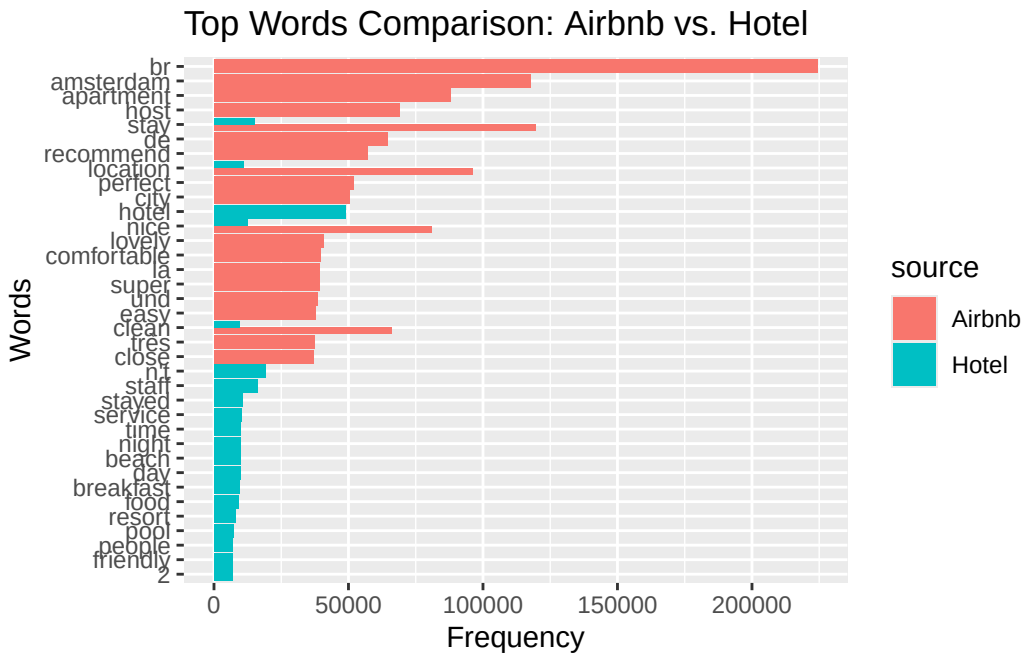
```
# Visualize sentiment distribution for Airbnb reviews with different colors for each sentimer
ggplot(Airbnbreviews, aes(x = sentimentbnb, fill = sentimentbnb)) +
  geom_bar() +
  labs(title = "Sentiment Distribution in Airbnb Reviews", x = "Sentiment", y = "Count") +
  scale_fill_manual(values = c("Positive" = "lightblue", "Neutral" = "lightgray", "Negative"
```




```
# Combine the top words for both datasets
combined_word_freq <- bind_rows(
  airbnb_word_freq %>% top_n(20) %>% mutate(source = "Airbnb"),
  hotel_word_freq %>% top_n(20) %>% mutate(source = "Hotel")
)
```

```
Selecting by n
Selecting by n
```

```
# Visualize the comparison
ggplot(combined_word_freq, aes(x = reorder(word, n), y = n, fill = source)) +
  geom_bar(stat = "identity", position = "dodge") +
  coord_flip() +
  labs(title = "Top Words Comparison: Airbnb vs. Hotel", x = "Words", y = "Frequency")
```

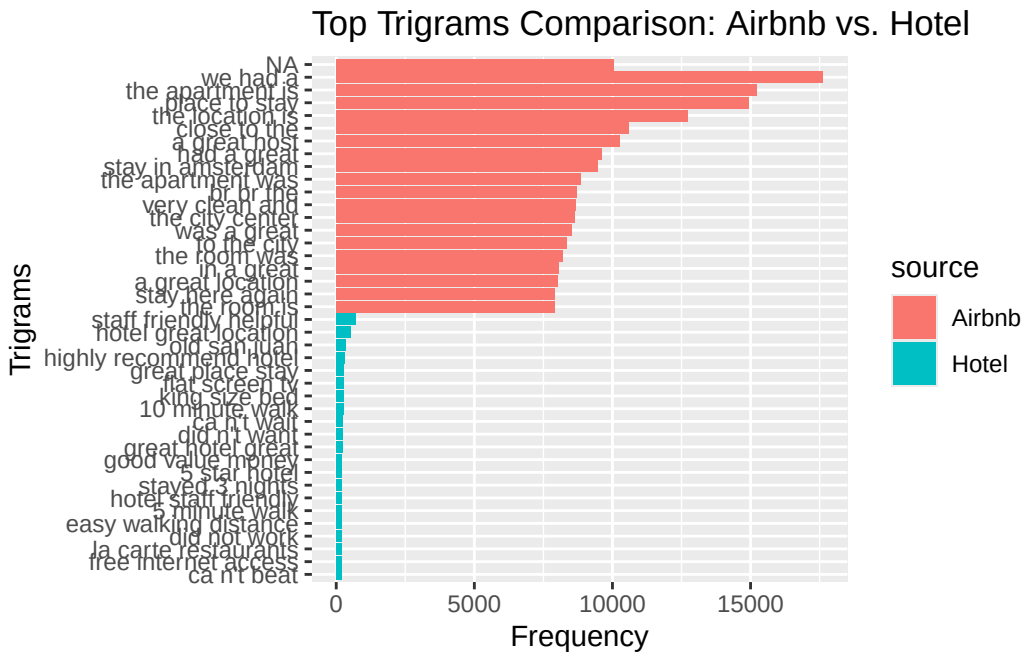


Comparison on the Ngrams

```
# Tokenize the datasets into trigrams
airbnb_trigrams <- Airbnbreviews2 %>%
  unnest_tokens(trigram, comments, token = "ngrams", n = 3)

hotel_trigrams <- Hotelreviews2 %>%
  unnest_tokens(trigram, Review, token = "ngrams", n = 3)

# Count the frequency of trigrams
airbnb_trigram_freq <- airbnb_trigrams %>%
  count(trigram, sort = TRUE)

hotel_trigram_freq <- hotel_trigrams %>%
  count(trigram, sort = TRUE)

# Combine the top trigrams for both datasets
combined_trigram_freq <- bind_rows(
  airbnb_trigram_freq %>% top_n(20) %>% mutate(source = "Airbnb"),
  hotel_trigram_freq %>% top_n(20) %>% mutate(source = "Hotel")
)
```

```
Selecting by n
Selecting by n
```

```
# Visualize the comparison of trigrams
ggplot(combined_trigram_freq, aes(x = reorder(trigram, n), y = n, fill = source)) +
  geom_bar(stat = "identity", position = "dodge") +
  coord_flip() +
  labs(title = "Top Trigrams Comparison: Airbnb vs. Hotel", x = "Trigrams", y = "Frequency")
```




```
# Box plot for Rating by Sentiment in Hotel reviews
ggplot(Hotelreviews, aes(x = sentiment, y = Rating, fill = sentiment)) +
  geom_boxplot() +
  labs(title = "Box Plot of Ratings by Sentiment (Hotel Reviews)", x = "Sentiment", y = "Rati
  scale_fill_manual(values = c("Positive" = "lightblue", "Neutral" = "lightgray", "Negative"
```


```
# Box plot for Rating by Sentiment in Airbnb reviews
ggplot(Airbnbreviews, aes(x = sentimentbnb, y = sentiment_score, fill = sentimentbnb)) +
  geom_boxplot() +
  labs(title = "Box Plot of  rating score by Sentiment (Airbnb Reviews)", x = "Sentiment", y
  scale_fill_manual(values = c("Positive" = "lightblue", "Neutral" = "lightgray", "Negative"
```


```
# Calculate word counts for each sentiment (Hotel and Airbnb)
#hotel_sentiment_word_counts <- Hotelreviews %>%
  #inner_join(get_sentiments("bing"), by = "word") %>%
  #count(word, sentiment, sort = TRUE) %>%
  #ungroup()

#airbnb_sentiment_word_counts <- Airbnbreviews %>%
  #inner_join(get_sentiments("bing"), by = "word") %>%
  #count(word, sentiment, sort = TRUE) %>%
  #ungroup()

# Combine both datasets
#combined_sentiment_counts <- bind_rows(
  #mutate(hotel_sentiment_word_counts, source = "Hotel"),
  #mutate(airbnb_sentiment_word_counts, source = "Airbnb")
#)

# Generate the comparison cloud
#library(wordcloud)
#combined_sentiment_counts %>%
  #count(word, sentiment, sort = TRUE) %>%
  #acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  #comparison.cloud(colors = c("red", "green"), max.words = 100)
```

```
library(reshape2)
```

```
Warning: package 'reshape2' was built under R version 4.5.1
```

```
Attaching package: 'reshape2'
```

```
The following object is masked from 'package:tidyr':

    smiths
```

```
# Count word frequencies for the Hotelreviews dataset
hotel_word_freq <- Hotelreviews %>%
  count(word, sort = TRUE)

# Create word cloud for Hotel reviews
wordcloud(words = hotel_word_freq$word,
          freq = hotel_word_freq$n,
          max.words = 100,
          random.order = FALSE,
          colors = brewer.pal(8, "Dark2"))
```

```r
# Count word frequencies for the Airbnbreviews dataset
airbnb_word_freq <- Airbnbreviews %>%
  count(word, sort = TRUE)

# Create word cloud for Airbnb reviews
wordcloud(words = airbnb_word_freq$word,
          freq = airbnb_word_freq$n,
          max.words = 100,
          random.order = FALSE,
          colors = brewer.pal(8, "Set1"))
```


```r
str(Airbnbreviews)
```

```
'data.frame':   7896166 obs. of  8 variables:
 $ listing_id     : num  2818 2818 2818 2818 2818 ...
 $ id             : num  1191 1191 1191 1191 1191 ...
 $ date           : chr  "3/30/2009" "3/30/2009" "3/30/2009" "3/30/2009" ...
 $ reviewer_id    : int  10952 10952 10952 10952 10952 10952 10952 10952 10952 10952 ...
 $ reviewer_name  : chr  "Lam" "Lam" "Lam" "Lam" ...
 $ word           : chr  "daniel" "cool" "nice" "clean" ...
 $ sentiment_score: num  2 2 2 2 2 2 2 2 2 2 ...
 $ sentimentbnb   : chr  "Positive" "Positive" "Positive" "Positive" ...
```

```
str(Hotelreviews)
```

```
'data.frame':   1704496 obs. of  3 variables:
 $ Rating   : int  4 4 4 4 4 4 4 4 4 4 ...
 $ word     : chr  "nice" "hotel" "expensive" "parking" ...
 $ sentiment: chr  "Positive" "Positive" "Positive" "Positive" ...
```

```
colnames(Airbnbreviews)
```

```
[1] "listing_id"      "id"              "date"            "reviewer_id"
[5] "reviewer_name"   "word"            "sentiment_score" "sentimentbnb"
```

```
colnames(Hotelreviews)
```

```
[1] "Rating"    "word"      "sentiment"
```

```
colnames(Airbnbreviews)
```

```
[1] "listing_id"      "id"              "date"            "reviewer_id"
[5] "reviewer_name"   "word"            "sentiment_score" "sentimentbnb"
```

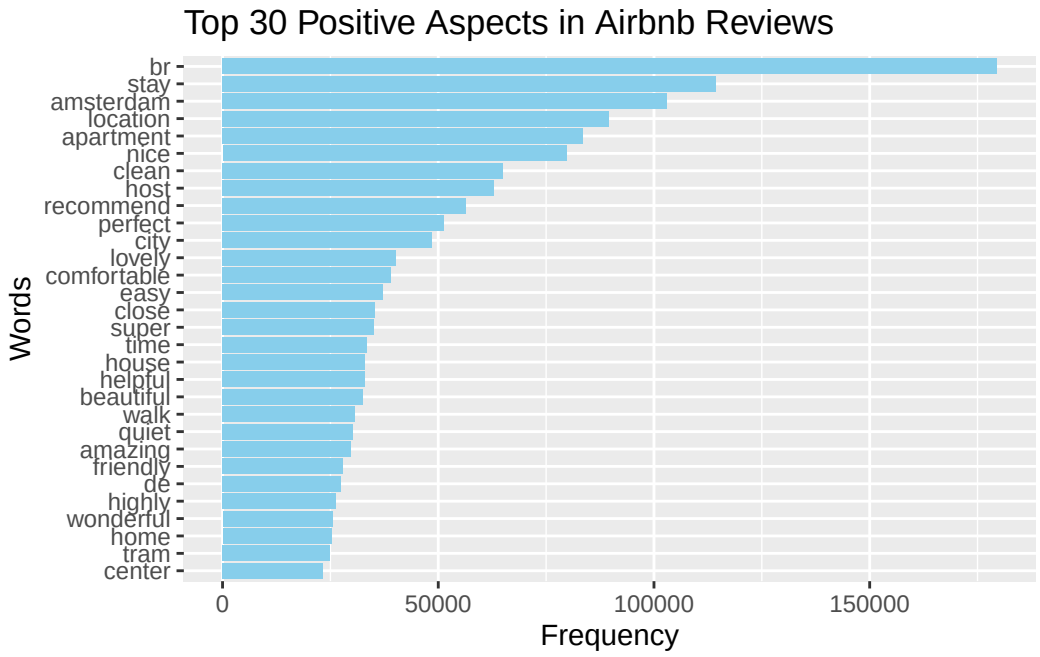
Part 5: RESEARCH QUESTIONS

QUESTION 1.1: Leveraging Positive Feedback from Airbnb Reviews(TF)

```
#Identify most frequently mentioned positive aspects using Bing lexicon
bing_lexicon <- get_sentiments("bing")

positive_airbnb <- Airbnbreviews %>%
  filter(sentimentbnb == "Positive") %>%
  anti_join(stop_words, by = "word") %>%  # Remove stop words
  filter(!word %in% c(".", ",", "!", "?", "-", "_", ":", ";", "(", ")")) %>%  # Remove punctu
  count(word, sort = TRUE) %>%
  top_n(30)
```

```
Selecting by n
```

```r
# Visualization - Bar Chart of Top 30 Positive Aspects
positive_airbnb %>%
  ggplot(aes(x = reorder(word, n), y = n)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  coord_flip() +
  labs(title = "Top 30 Positive Aspects in Airbnb Reviews", x = "Words", y = "Frequency")
```

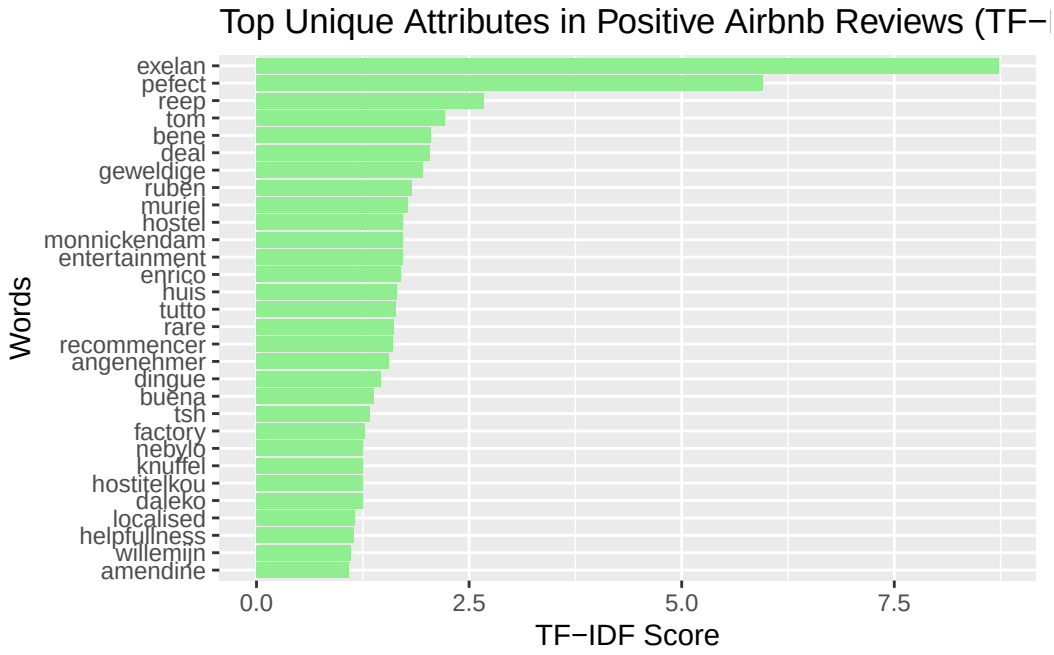



```r
# Visualization - Word Cloud of Positive Aspects
wordcloud(words = positive_airbnb$word, freq = positive_airbnb$n, max.words = 100, colors = 
```

QUESTION1:2 Unique Attributes Praised in Positive Airbnb Reviews(TF*IDF)

```
# Analytical Techniques: TF-IDF Analysis
# Data Wrangling: Remove stop words, group by sentiment and word, apply TF-IDF calculations
# Data Visualization: Word Cloud, Bar Chart

# Step 2: Remove Standard Stop Words
Airbnbreviews <- Airbnbreviews %>%
  anti_join(stop_words, by = "word") %>%  # Remove standard English stop words

  # Step 3: Filter for English Words Only
  filter(str_detect(word, "^[a-zA-Z]+$")) %>%  # Keep only words with alphabetic characters
  filter(str_length(word) > 1)  # Remove single character tokens (e.g., "a", "i")

# Step 4: Apply TF-IDF to Identify Unique Attributes for Positive Reviews
airbnb_tf_idf <- Airbnbreviews %>%
  filter(sentimentbnb == "Positive") %>%  # Filter for positive reviews
  count(listing_id, word, sort = TRUE) %>%
  bind_tf_idf(word, listing_id, n) %>%
  arrange(desc(tf_idf)) %>%
  slice_max(tf_idf, n = 30)  # Use slice_max to get the top 30 TF-IDF values
# Visualization - Bar Chart of Top TF-IDF Attributes
ggplot(airbnb_tf_idf, aes(x = reorder(word, tf_idf), y = tf_idf)) +
```

```
  geom_bar(stat = "identity", fill = "lightgreen") +
  coord_flip() +
  labs(title = "Top Unique Attributes in Positive Airbnb Reviews (TF-IDF)", x = "Words", y =
```

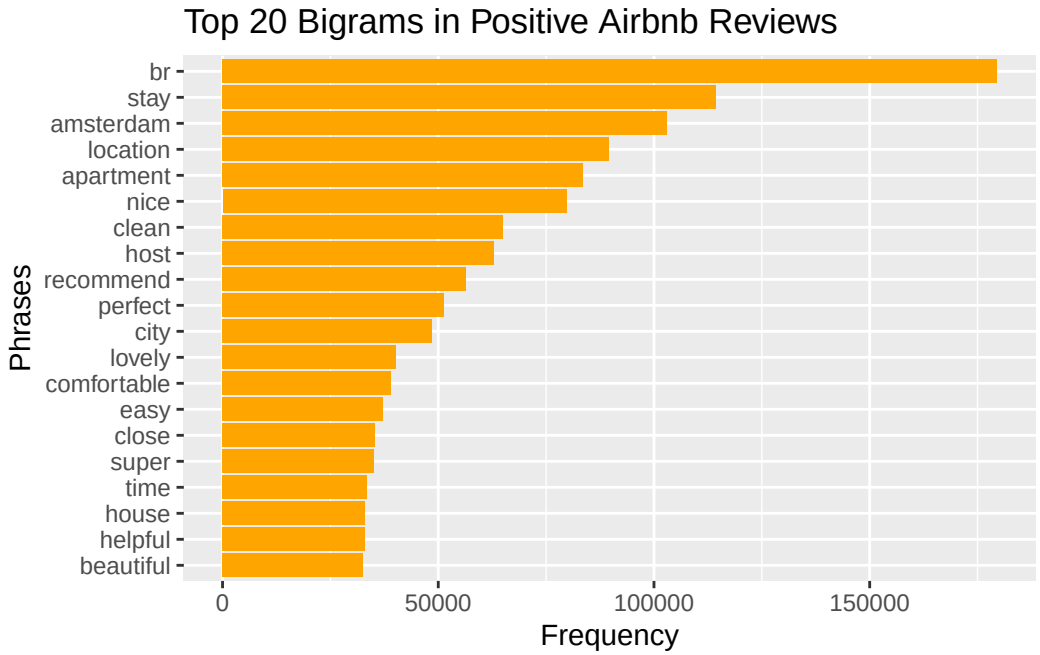


QUESTION 1:3 Common Phrases Expressing Satisfaction(NGRAMS)

```
#Filter positive reviews from Airbnb Dataset
positive_ngrams <- Airbnbreviews %>%
  filter(sentimentbnb == "Positive") %>%
  count(word, sort = TRUE) %>%
  top_n(20)
```

```
Selecting by n
```

```
# Visualization - Bar Chart of Top 20 Bigrams
positive_ngrams %>%
  ggplot(aes(x = reorder(word, n), y = n)) +
  geom_bar(stat = "identity", fill = "orange") +
  coord_flip() +
  labs(title = "Top 20 Bigrams in Positive Airbnb Reviews", x = "Phrases", y = "Frequency")
```




```
# Visualization - Word Cloud of Common Satisfaction-Related Phrases
wordcloud(words = positive_ngrams$word, freq = positive_ngrams$n, max.words = 100, colors = b
```

QUESTION 2:1 Common Service-Related Complaints

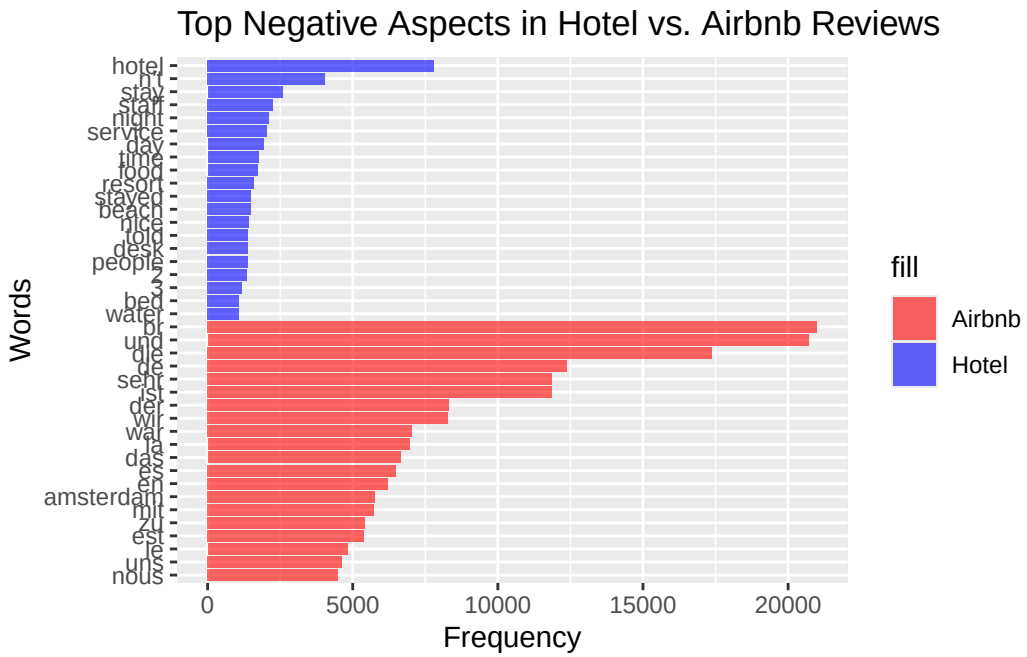
```
# Load necessary libraries
library(dplyr)
library(ggplot2)
library(tidytext)
library(wordcloud)
library(RColorBrewer)

# Load stop words
data("stop_words")

# Most frequently mentioned negative aspects for Hotel and Airbnb reviews
negative_airbnb <- Airbnbreviews %>%
  filter(sentimentbnb == "Negative") %>%
  anti_join(stop_words, by = "word") %>%
  filter(!word %in% c(".", ",", "!", "?", "-", "_", ":", ";", "(", ")")) %>%
  count(word, sort = TRUE) %>%
  top_n(20)
```

```
Selecting by n
```

```
negative_hotel <- Hotelreviews %>%
  filter(sentiment == "Negative") %>%
  anti_join(stop_words, by = "word") %>%
  filter(!word %in% c(".", ",", "!", "?", "-", "_", ":", ";", "(", ")")) %>%
  count(word, sort = TRUE) %>%
  top_n(20)
```

```
Selecting by n
```

```
#Comparative Bar Chart of Top Negative Aspects
ggplot() +
  geom_bar(data = negative_airbnb, aes(x = reorder(word, n), y = n, fill = "Airbnb"), stat =
  geom_bar(data = negative_hotel, aes(x = reorder(word, n), y = n, fill = "Hotel"), stat = "
  coord_flip() +
  labs(title = "Top Negative Aspects in Hotel vs. Airbnb Reviews", x = "Words", y = "Frequenc
  scale_fill_manual(values = c("Airbnb" = "red", "Hotel" = "blue"))
```




```r
#Word Cloud for Negative Aspects in Airbnb Reviews
set.seed(123)
wordcloud(
  words = negative_airbnb$word,
  freq = negative_airbnb$n,
  max.words = 100,
  random.order = FALSE,
  colors = brewer.pal(8, "Reds"),
  main = "Word Cloud of Negative Aspects in Airbnb Reviews"
)
```


```
#Word Cloud for Negative Aspects in Hotel Reviews
set.seed(123)
wordcloud(
  words = negative_hotel$word,
  freq = negative_hotel$n,
  max.words = 100,
  random.order = FALSE,
  colors = brewer.pal(8, "Blues"),
  main = "Word Cloud of Negative Aspects in Hotel Reviews"
)
```


```
colnames(Airbnbreviews)
```

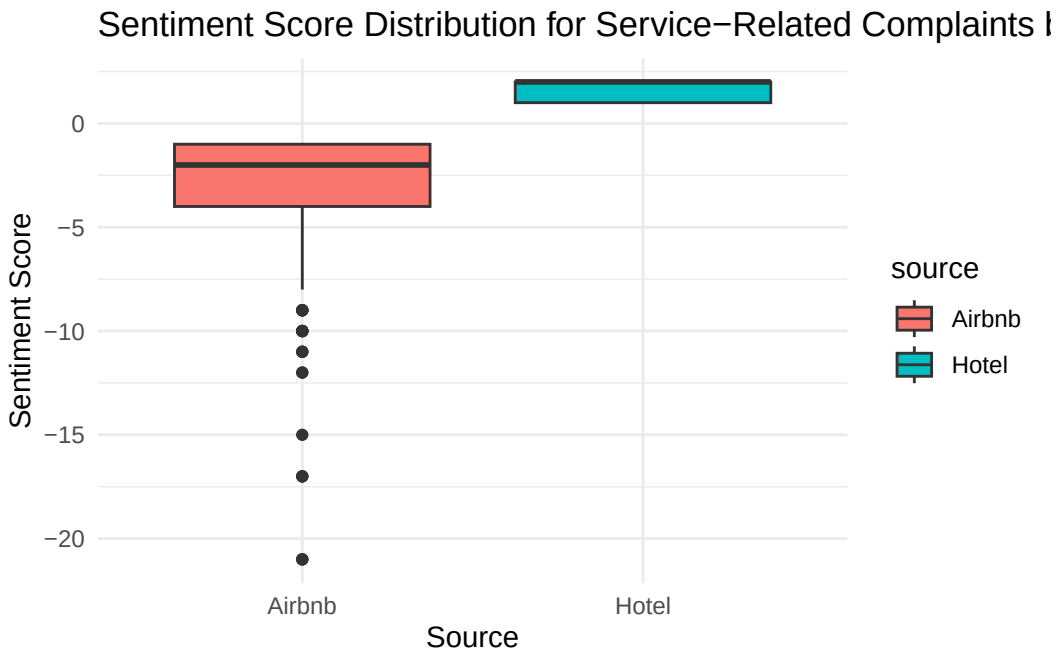
```
[1] "listing_id"      "id"              "date"            "reviewer_id"
[5] "reviewer_name"   "word"            "sentiment_score" "sentimentbnb"
```

```
# Rename the 'Rating' column in Hotelreviews to 'sentiment_score' for consistency
Hotelreviews <- Hotelreviews %>%
  rename(sentiment_score = Rating)
```

QUESTION 2:2 Sentiment Comparison Between Hotel and Airbnb Reviews

```
# Load necessary libraries
library(dplyr)
library(ggplot2)
#create a vector with service related words
# Filter for service-related words in both Airbnb and Hotel reviews
service_keywords <- c("service", "staff", "help", "support")
```

```r
# Filter Airbnb dataset for service-related words
airbnb_service_reviews <- Airbnbreviews %>%
  filter(word %in% service_keywords, sentimentbnb == "Negative")  # Filter for negative senti

# Filter Hotel dataset for service-related words
hotel_service_reviews <- Hotelreviews %>%
  filter(word %in% service_keywords, sentiment == "Negative")  # Filter for negative sentimer

# Combine the filtered datasets
combined_service_sentiment <- bind_rows(
  airbnb_service_reviews %>% mutate(source = "Airbnb"),
  hotel_service_reviews %>% mutate(source = "Hotel")
)

# Visualization - Box Plot of Service-Related Sentiment Scores by Source
ggplot(combined_service_sentiment, aes(x = source, y = sentiment_score, fill = source)) +
  geom_boxplot() +
  labs(title = "Sentiment Score Distribution for Service-Related Complaints by Source",
       x = "Source", y = "Sentiment Score") +
  theme_minimal()
```



```
colnames(Hotelreviews)
```

```
[1] "sentiment_score" "word"            "sentiment"
```

```
colnames(Airbnbreviews)
```

```
[1] "listing_id"      "id"              "date"            "reviewer_id"
[5] "reviewer_name"   "word"            "sentiment_score" "sentimentbnb"
```

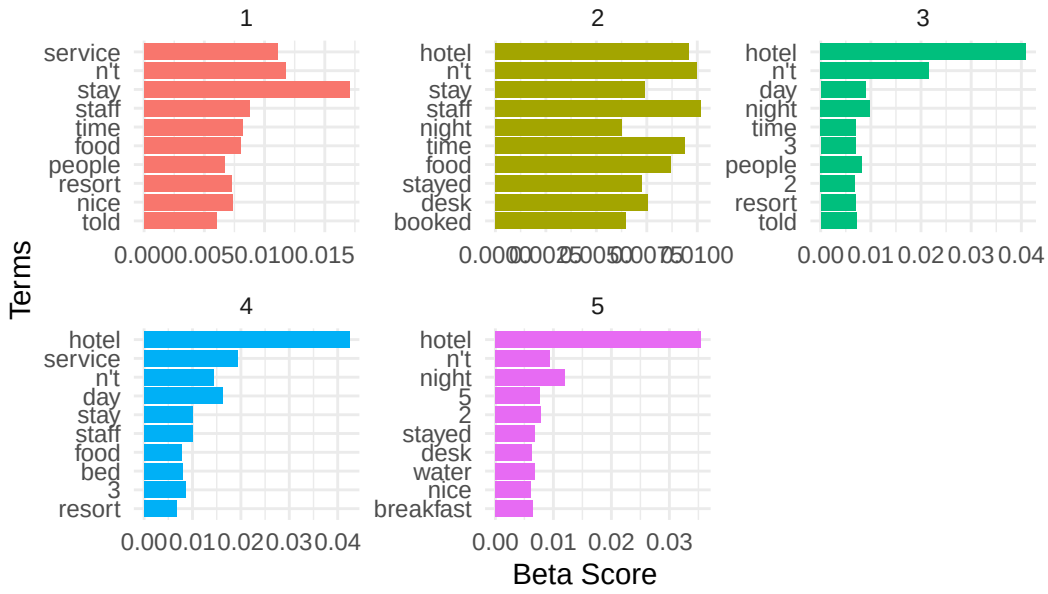
QUESTION 3:How can hotels address areas of dissatisfaction highlighted in negative Hotel reviews?

```
# Load required libraries
library(tidytext)
library(dplyr)
library(topicmodels)
```

```
Warning: package 'topicmodels' was built under R version 4.5.1
```

```
library(ggplot2)
library(wordcloud)

# Filter negative reviews from Hotelreviews dataset
negative_hotel_reviews <- Hotelreviews %>%
  filter(sentiment == "Negative")

# Step 2: Tokenize the negative reviews and remove stop words and punctuation
negative_hotel_tokens <- negative_hotel_reviews %>%
  unnest_tokens(word, word) %>%  # Unnest tokens to create a tokenized version of the text
  anti_join(stop_words, by = "word") %>%  # Remove stop words
  filter(!word %in% c(".", ",", "!", "?", "-", "_", ":", ";", "(", ")"))  # Remove punctuatio

# Step 3: Create a Document-Term Matrix (DTM) for LDA analysis
# Note: As there is no document ID in the data, I'll create one using row number
negative_hotel_tokens <- negative_hotel_tokens %>%
  mutate(document = row_number())  # Creating document ID

dtm <- negative_hotel_tokens %>%
```

```
  count(document, word) %>%
  cast_dtm(document = document, term = word, value = n)

# Apply LDA to extract topics from negative reviews
# Setting the number of topics to 5
lda_model <- LDA(dtm, k = 5, control = list(seed = 1234))

# Step 5: Extract topics and visualize
topics <- tidy(lda_model, matrix = "beta")

# Step 6: Visualize top terms for each topic
top_terms <- topics %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

# Step 7: Plot top terms for each topic
ggplot(top_terms, aes(x = reorder(term, beta), y = beta, fill = as.factor(topic))) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free", ncol = 3) +
  coord_flip() +
  labs(title = "Top Terms for Each Topic in Negative Hotel Reviews", x = "Terms", y = "Beta S
  theme_minimal()
```




```r
# Step 8: Create a word cloud for topics
# Get top terms for word cloud visualization
top_terms_wordcloud <- top_terms %>%
  group_by(topic) %>%
  summarise(word = paste(term, collapse = " "))

# Visualize the word cloud for topics
wordcloud(words = top_terms$term, freq = top_terms$beta, max.words = 100, random.order = FALS
```

QUESTION 4:Seasonal Trends in Airbnb Satisfaction

First check columns

```
# Check column names and types to see if 'date' is present
colnames(Airbnbreviews)
```

```
[1] "listing_id"      "id"              "date"            "reviewer_id"
[5] "reviewer_name"   "word"            "sentiment_score" "sentimentbnb"
```

Check date format

```
# Check the structure
str(Airbnbreviews)
```

```
'data.frame':   7159772 obs. of  8 variables:
 $ listing_id     : num  2818 2818 2818 2818 2818 ...
 $ id             : num  1191 1191 1191 1191 1191 ...
```

```
 $ date           : chr  "3/30/2009" "3/30/2009" "3/30/2009" "3/30/2009" ...
 $ reviewer_id    : int  10952 10952 10952 10952 10952 10952 10952 10952 10952 10952 ...
 $ reviewer_name  : chr  "Lam" "Lam" "Lam" "Lam" ...
 $ word           : chr  "daniel" "cool" "nice" "clean" ...
 $ sentiment_score: num  2 2 2 2 2 2 2 2 2 2 ...
 $ sentimentbnb   : chr  "Positive" "Positive" "Positive" "Positive" ...
```

```r
# Print the first few rows of the date column to confirm its format
head(Airbnbreviews$date)
```

```
[1] "3/30/2009" "3/30/2009" "3/30/2009" "3/30/2009" "3/30/2009" "3/30/2009"
```

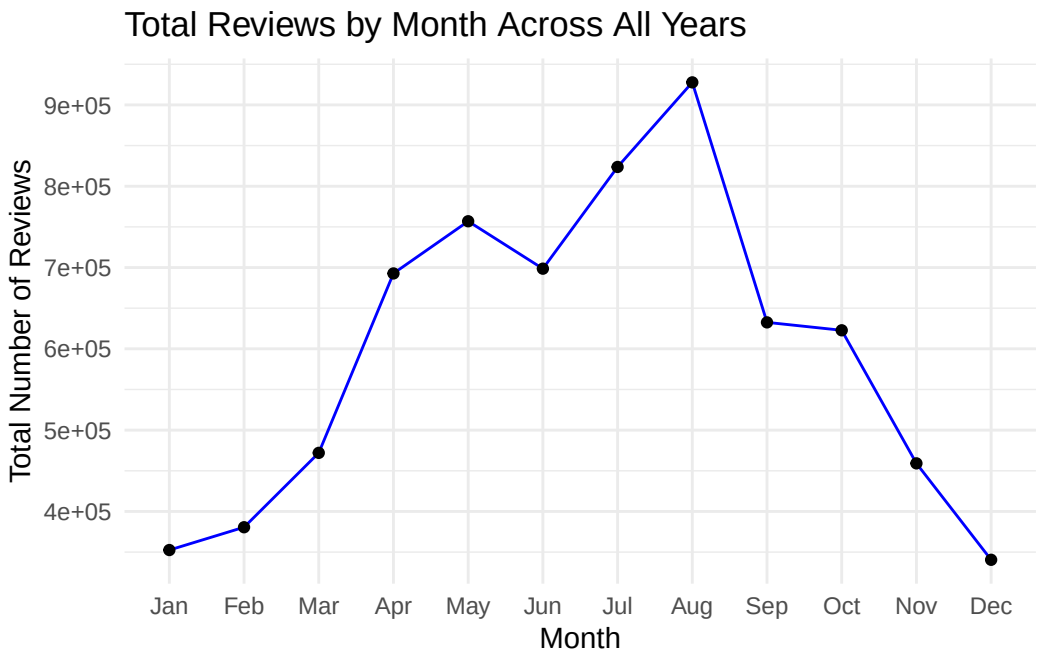
Parse the date

```r
# Load necessary libraries
library(lubridate)
library(dplyr)

# Correct the date format using mdy
Airbnbreviews <- Airbnbreviews %>%
  mutate(date_parsed = mdy(date))

# Check if the parsing is correct
head(Airbnbreviews)
```

```
  listing_id   id      date reviewer_id reviewer_name         word
1       2818 1191 3/30/2009       10952           Lam       daniel
2       2818 1191 3/30/2009       10952           Lam         cool
3       2818 1191 3/30/2009       10952           Lam         nice
4       2818 1191 3/30/2009       10952           Lam        clean
5       2818 1191 3/30/2009       10952           Lam        quiet
6       2818 1191 3/30/2009       10952           Lam neighborhood
  sentiment_score sentimentbnb date_parsed
1               2     Positive  2009-03-30
2               2     Positive  2009-03-30
3               2     Positive  2009-03-30
4               2     Positive  2009-03-30
5               2     Positive  2009-03-30
6               2     Positive  2009-03-30
```

Trend analysis on the monthly reviews

```
# Load necessary libraries
library(lubridate)
library(dplyr)
library(ggplot2)

# Group reviews by month to count the number of reviews per month across all years
monthly_trend <- Airbnbreviews %>%
  mutate(month = month(date_parsed, label = TRUE)) %>%  # Extract month with labels (e.g., Ja
  group_by(month) %>%
  summarize(total_reviews = n())

# Plot the trend by month across all years
ggplot(monthly_trend, aes(x = month, y = total_reviews)) +
  geom_line(group = 1, color = "blue") +
  geom_point() +
  labs(title = "Total Reviews by Month Across All Years",
       x = "Month",
       y = "Total Number of Reviews") +
  theme_minimal()
```

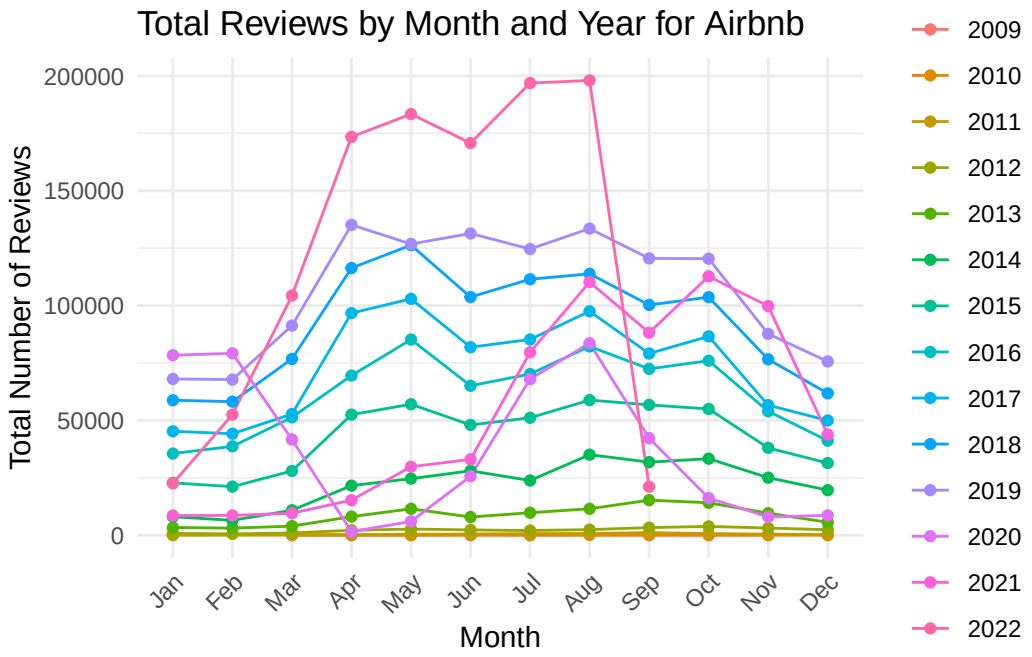


Trend Analysis Including Year and Month

```
# Load necessary libraries
library(lubridate)
library(dplyr)
library(ggplot2)

# Extract both year and month from the date_parsed column
Airbnbreviews <- Airbnbreviews %>%
  mutate(
    year = year(date_parsed),           # Extract year from date
    month = month(date_parsed, label = TRUE)  # Extract month with labels (e.g., Jan, Feb)
  )

# Group reviews by year and month to count the number of reviews
yearly_monthly_trend <- Airbnbreviews %>%
  group_by(year, month) %>%
  summarize(total_reviews = n()) %>%
  ungroup()
```

```
`summarise()` has grouped output by 'year'. You can override using the
`.groups` argument.
```

```
# Plot the trend by year and month
ggplot(yearly_monthly_trend, aes(x = month, y = total_reviews, group = year, color = factor(y
  geom_line() +
  geom_point() +
  labs(title = "Total Reviews by Month and Year for Airbnb",
       x = "Month",
       y = "Total Number of Reviews",
       color = "Year") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



QUESTION 5:Customer Experience Differences Between Hotel and Airbnb

```
colnames(Airbnbreviews)
```

```
 [1] "listing_id"      "id"              "date"            "reviewer_id"
 [5] "reviewer_name"   "word"            "sentiment_score" "sentimentbnb"
 [9] "date_parsed"     "year"            "month"
```

```
head(Airbnbreviews)
```

```
  listing_id   id      date reviewer_id reviewer_name         word
1       2818 1191 3/30/2009       10952           Lam       daniel
2       2818 1191 3/30/2009       10952           Lam         cool
3       2818 1191 3/30/2009       10952           Lam         nice
4       2818 1191 3/30/2009       10952           Lam        clean
5       2818 1191 3/30/2009       10952           Lam        quiet
6       2818 1191 3/30/2009       10952           Lam neighborhood
  sentiment_score sentimentbnb date_parsed year month
1               2     Positive  2009-03-30 2009   Mar
2               2     Positive  2009-03-30 2009   Mar
```

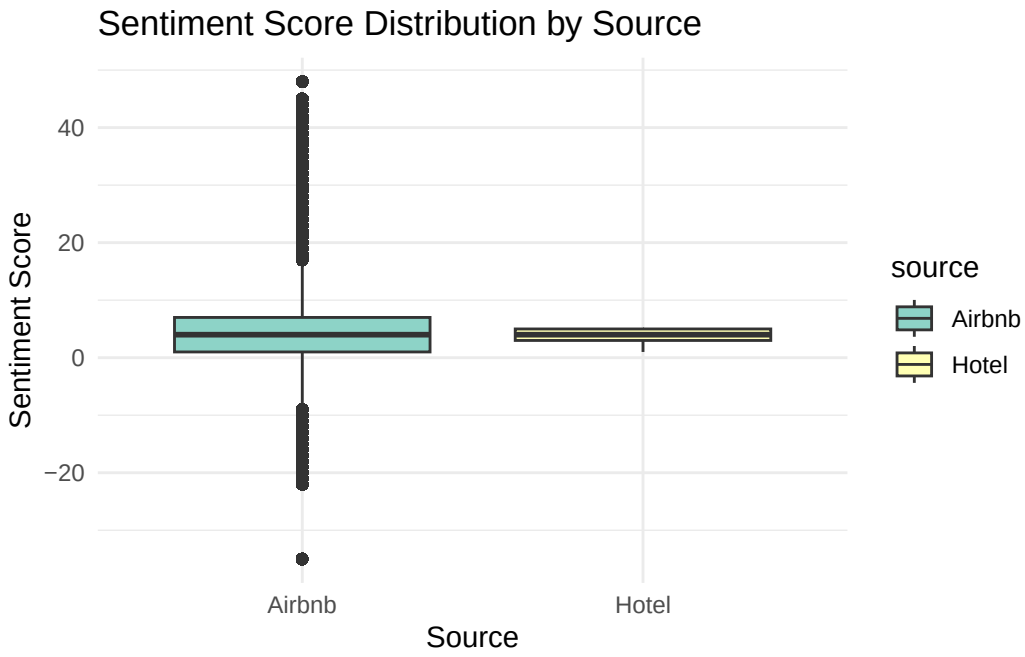
```
3               2     Positive  2009-03-30 2009   Mar
4               2     Positive  2009-03-30 2009   Mar
5               2     Positive  2009-03-30 2009   Mar
6               2     Positive  2009-03-30 2009   Mar
```

```r
colnames(Hotelreviews)
```

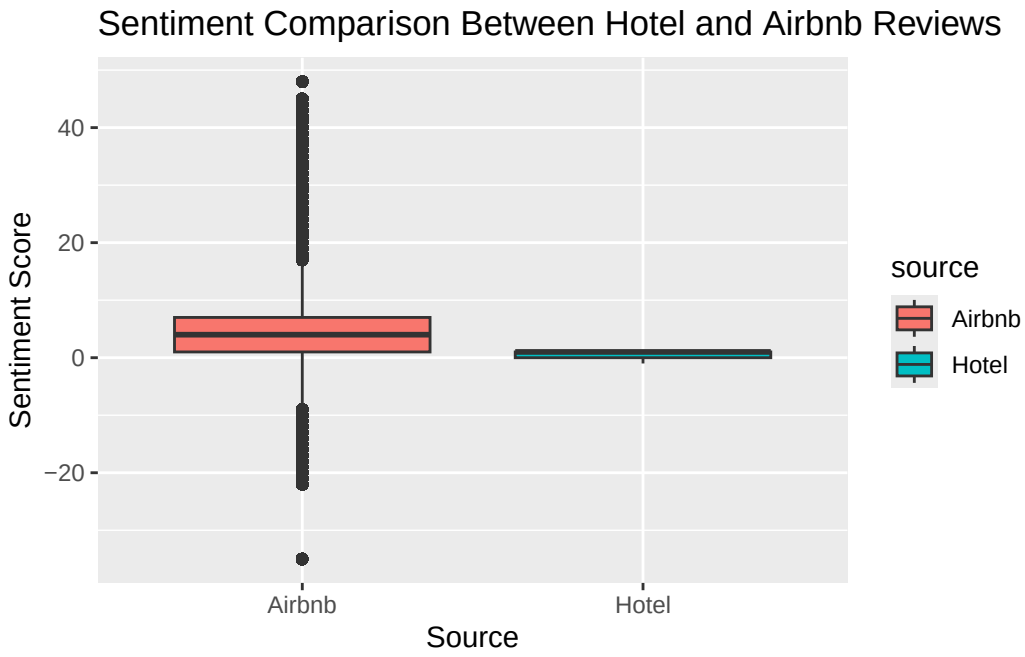
```
[1] "sentiment_score" "word"            "sentiment"
```

```r
# Load stop words
data("stop_words")

# Combine the Datasets

# Create a combined dataset for sentiment analysis
combined_sentiment <- bind_rows(
  Hotelreviews %>% mutate(source = "Hotel", sentimentbnb = sentiment),
  Airbnbreviews %>% mutate(source = "Airbnb")
) %>%
  count(source, sentimentbnb)

#Bar Chart of Sentiment Distribution by Source
ggplot(combined_sentiment, aes(x = source, y = n, fill = sentimentbnb)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Sentiment Distribution Between Hotel and Airbnb Reviews",
       x = "Source", y = "Count") +
  scale_fill_brewer(palette = "Set3") +
  theme_minimal()
```

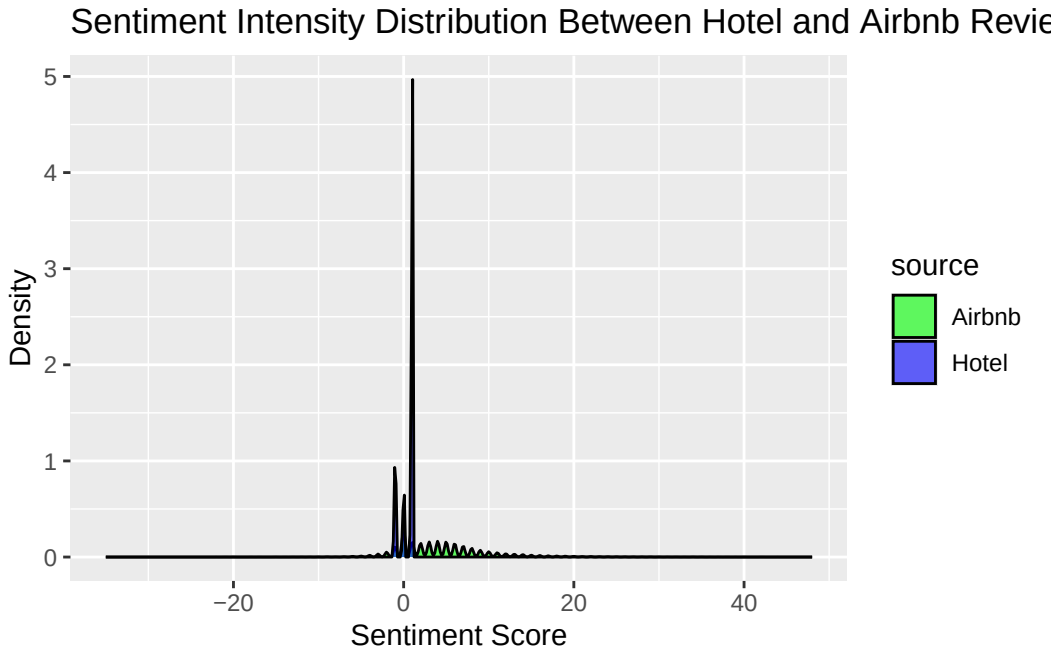

```r
# Sentiment Scores Comparison (Box Plot)
# Create a combined dataset for sentiment scores
combined_sentiment_scores <- bind_rows(
  Airbnbreviews %>% mutate(source = "Airbnb"),
  Hotelreviews %>% mutate(source = "Hotel")
)

# Box Plot of Sentiment Scores by Source
ggplot(combined_sentiment_scores, aes(x = source, y = sentiment_score, fill = source)) +
  geom_boxplot() +
  labs(title = "Sentiment Score Distribution by Source", x = "Source", y = "Sentiment Score")
  theme_minimal() +
  scale_fill_brewer(palette = "Set3")
```



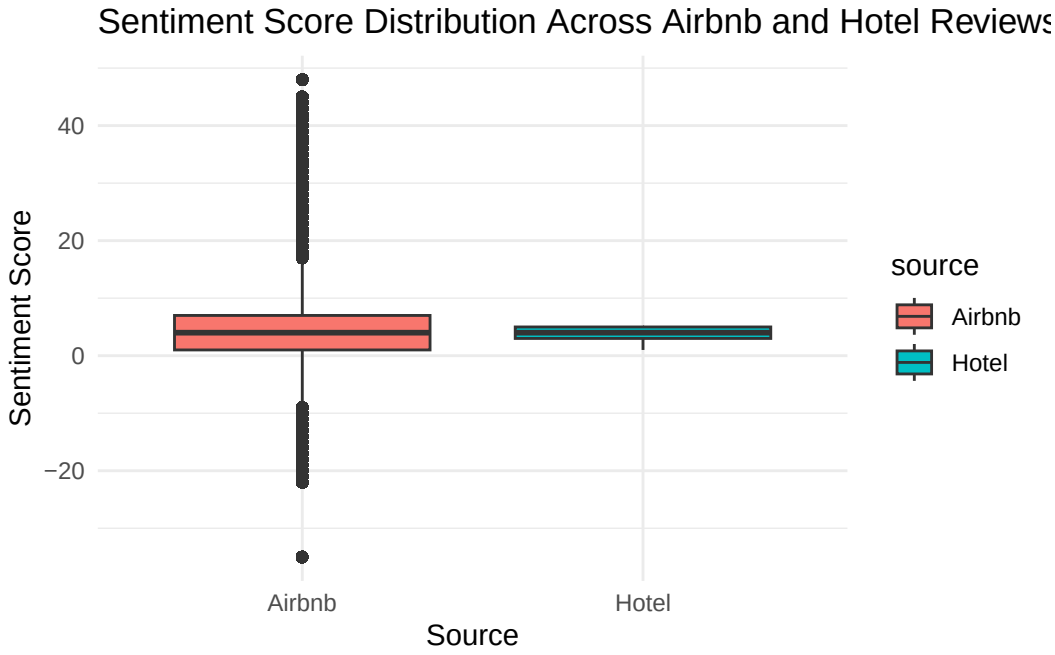

```
combined_reviews <- bind_rows(
  Hotelreviews %>% mutate(source = "Hotel", score = ifelse(sentiment == "Positive", 1, ifels
  Airbnbreviews %>% mutate(source = "Airbnb", score = sentiment_score)
)

# Visualization - Box Plot of Sentiment Distribution
ggplot(combined_reviews, aes(x = source, y = score, fill = source)) +
  geom_boxplot() +
  labs(title = "Sentiment Comparison Between Hotel and Airbnb Reviews", x = "Source", y = "S
```



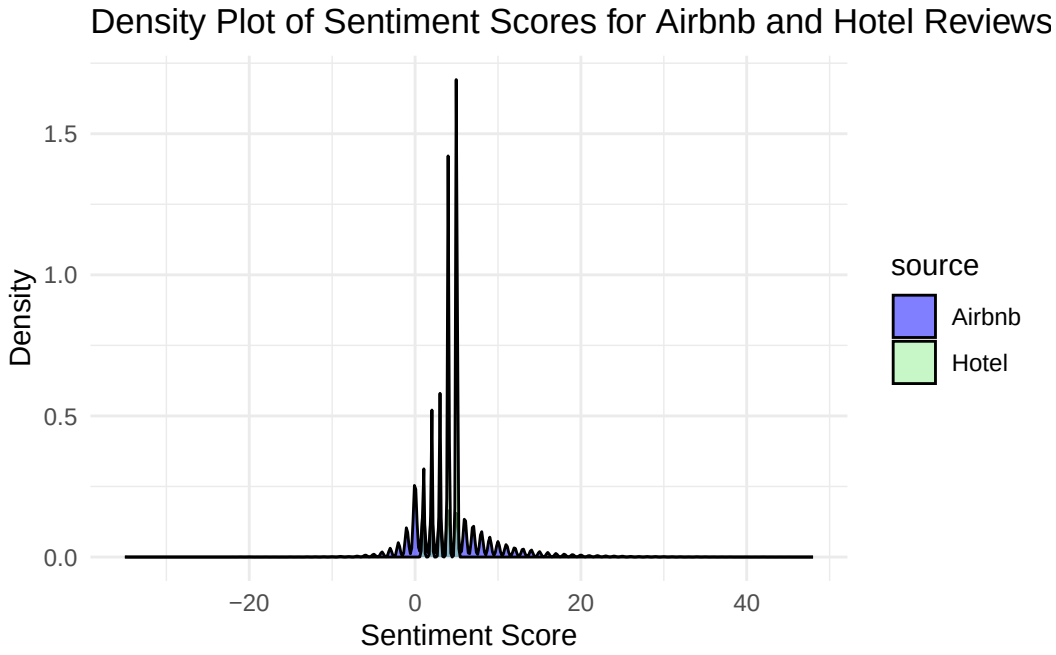

```
# Visualization - Density Plot of Sentiment Distribution
ggplot(combined_reviews, aes(x = score, fill = source)) +
  geom_density(alpha = 0.6) +
  labs(title = "Sentiment Intensity Distribution Between Hotel and Airbnb Reviews", x = "Sent
  scale_fill_manual(values = c("Hotel" = "blue", "Airbnb" = "green"))
```




```r
# Load necessary libraries
library(dplyr)
library(ggplot2)

# Combine both datasets for comparative analysis
combined_reviews <- bind_rows(
  Airbnbreviews %>%
    select(sentiment_score) %>%
    mutate(source = "Airbnb"),
  Hotelreviews %>%
    select(sentiment_score) %>%
    mutate(source = "Hotel")
)

# Box Plot: Comparing Sentiment Score Distribution Across Airbnb and Hotel Reviews
ggplot(combined_reviews, aes(x = source, y = sentiment_score, fill = source)) +
  geom_boxplot() +
  labs(title = "Sentiment Score Distribution Across Airbnb and Hotel Reviews",
       x = "Source",
       y = "Sentiment Score") +
  theme_minimal()
```




```
# Density Plot: Comparing Sentiment Intensity and Distribution Differences
ggplot(combined_reviews, aes(x = sentiment_score, fill = source)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density Plot of Sentiment Scores for Airbnb and Hotel Reviews",
       x = "Sentiment Score",
       y = "Density") +
  theme_minimal() +
  scale_fill_manual(values = c("Airbnb" = "blue", "Hotel" = "lightgreen"))
```


Density Plot of Sentiment Scores for Airbnb and Hotel Reviews
Density
1.5
1.0
0.5
0.0
−20
0
20
40
Sentiment Score
source
Airbnb
Hotel