Airbnb Sentiment Analysis

Alessandra Bielli

2024-11-19

Part 1: Preparing for Analysis

Load the Data and View the Structure

```
# Read the data
library(readr)
reviews <- read_csv("Austin_Reviews.csv")
```

```
## Rows: 633196 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): reviewer_name, comments
## dbl  (3): listing_id, id, reviewer_id
## date (1): date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# View the Structure and show the first few rows of the data
str(reviews)
```

```
## spc_tbl_ [633,196 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ listing_id   : num [1:633196] 5456 5456 5456 5456 5456 ...
##  $ id           : num [1:633196] 865 977 1039 1347 1491 ...
##  $ date         : Date[1:633196], format: "2009-03-08" "2009-03-19" ...
##  $ reviewer_id  : num [1:633196] 5267 8102 8241 11152 12400 ...
##  $ reviewer_name: chr [1:633196] "Ellen" "Phil" "Galen" "April" ...
##  $ comments     : chr [1:633196] "Sylvia is a hostess who is gracious and helpful beyond words! Firs
##  - attr(*, "spec")=
##   .. cols(
##   ..   listing_id = col_double(),
##   ..   id = col_double(),
##   ..   date = col_date(format = ""),
##   ..   reviewer_id = col_double(),
##   ..   reviewer_name = col_character(),
##   ..   comments = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
head(reviews)
```

```
## # A tibble: 6 x 6
##   listing_id    id date       reviewer_id reviewer_name      comments
##        <dbl> <dbl> <date>           <dbl> <chr>              <chr>
```

```
## 1       5456   865 2009-03-08        5267 Ellen              "Sylvia is a hoste~
## 2       5456   977 2009-03-19        8102 Phil               "Highly recommende~
## 3       5456  1039 2009-03-22        8241 Galen              "A great place to ~
## 4       5456  1347 2009-04-08       11152 April              "Highly recommende~
## 5       5456  1491 2009-04-13       12400 Ivonne             "What a great litt~
## 6       5456  1535 2009-04-16       11071 Egan.Sturges.Regan "Sylvia was great;~
```

Check for Missing Values and Handle Them

```
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
```

```
library(tidytext)
library(stringr)

# Count the number of rows and colums
nrow(reviews)
```

```
## [1] 633196
```

```
ncol(reviews)
```

```
## [1] 6
```

```
# Count missing values in columns
colSums(is.na(reviews))
```

```
##    listing_id            id          date   reviewer_id reviewer_name
##             0             0             0             0             0
##      comments
##            32
```

```
# Remove rows with NA in the comments column
reviews <- reviews %>%
  filter(!is.na(comments))

# Verify there are no NAs left in the comments column
sum(is.na(reviews$comments))  # Should return 0
```

```
## [1] 0
```

```
# View the Structure and show the first few rows of the data
str(reviews)
```

```
## spc_tbl_ [633,164 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ listing_id   : num [1:633164] 5456 5456 5456 5456 5456 ...
##  $ id           : num [1:633164] 865 977 1039 1347 1491 ...
##  $ date         : Date[1:633164], format: "2009-03-08" "2009-03-19" ...
##  $ reviewer_id  : num [1:633164] 5267 8102 8241 11152 12400 ...
```

```
##  $ reviewer_name: chr [1:633164] "Ellen" "Phil" "Galen" "April" ...
##  $ comments     : chr [1:633164] "Sylvia is a hostess who is gracious and helpful beyond words! Firs
##  - attr(*, "spec")=
##   .. cols(
##   ..   listing_id = col_double(),
##   ..   id = col_double(),
##   ..   date = col_date(format = ""),
##   ..   reviewer_id = col_double(),
##   ..   reviewer_name = col_character(),
##   ..   comments = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
head(reviews)
```

```
## # A tibble: 6 x 6
##   listing_id    id date       reviewer_id reviewer_name      comments
##        <dbl> <dbl> <date>           <dbl> <chr>              <chr>
## 1       5456   865 2009-03-08        5267 Ellen              "Sylvia is a hoste~
## 2       5456   977 2009-03-19        8102 Phil               "Highly recommende~
## 3       5456  1039 2009-03-22        8241 Galen              "A great place to ~
## 4       5456  1347 2009-04-08       11152 April              "Highly recommende~
## 5       5456  1491 2009-04-13       12400 Ivonne             "What a great litt~
## 6       5456  1535 2009-04-16       11071 Egan.Sturges.Regan "Sylvia was great;~
```

Clean the Date Column

```r
library(dplyr)

# Convert 'date' to Date object if it is not already, then extract Year-Month
reviews <- reviews %>%
  mutate(date = as.Date(date, format = "%Y-%m-%d"),  # Ensure 'date' is in Date format
         year_month = format(date, "%Y-%m"))  # Extract Year-Month format

# Show the first and last few rows of the data
head(reviews)
```

```
## # A tibble: 6 x 7
##   listing_id    id date       reviewer_id reviewer_name      comments year_month
##        <dbl> <dbl> <date>           <dbl> <chr>              <chr>    <chr>
## 1       5456   865 2009-03-08        5267 Ellen              "Sylvia~ 2009-03
## 2       5456   977 2009-03-19        8102 Phil               "Highly~ 2009-03
## 3       5456  1039 2009-03-22        8241 Galen              "A grea~ 2009-03
## 4       5456  1347 2009-04-08       11152 April              "Highly~ 2009-04
## 5       5456  1491 2009-04-13       12400 Ivonne             "What a~ 2009-04
## 6       5456  1535 2009-04-16       11071 Egan.Sturges.Regan "Sylvia~ 2009-04
```

```r
tail(reviews)
```

```
## # A tibble: 6 x 7
##   listing_id      id date       reviewer_id reviewer_name comments    year_month
##        <dbl>   <dbl> <date>           <dbl> <chr>         <chr>       <chr>
## 1    1.23e18 1.24e18 2024-09-06   586704141 Caylen        enjoyed th~ 2024-09
## 2    1.23e18 1.24e18 2024-09-08   140429084 Lien          Shawn was ~ 2024-09
## 3    1.23e18 1.24e18 2024-09-11   112800383 Robert        Nicely fur~ 2024-09
## 4    1.23e18 1.24e18 2024-09-10    82486289 Joshua        A truly gr~ 2024-09
```

```
## 5    1.23e18 1.24e18 2024-09-02   163122238 Diana         We booked ~ 2024-09
## 6    1.24e18 1.24e18 2024-09-08    25187570 Roxanne       The place ~ 2024-09
```

Part 2: Analyze Comment Count and Length

Examine the Text of all Comments

```r
# Check the length of the first few comments
nchar(head(reviews$comments))
```

```
## [1] 524 205 350  80 369 280
```

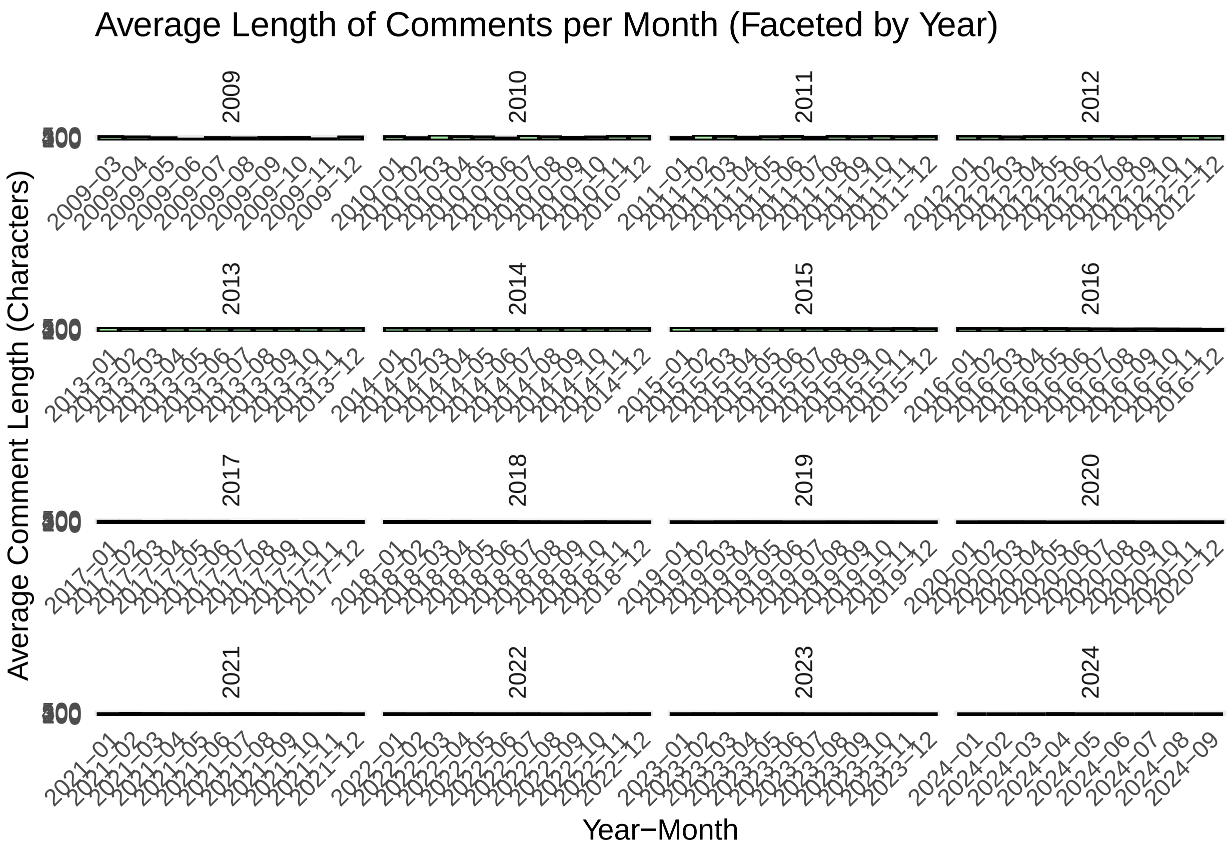
```r
# Get the mean length of comments
mean(nchar(reviews$comments), na.rm = TRUE)
```

```
## [1] 221.3008
```

This is the number of characters in the first 6 comments (reviews), and the average length of airbnb reviews in Austin is 221.3008.

Plot Average Length of Comments per Month per Year

```r
library(dplyr)
library(ggplot2)

# Calculate the average comment length for each Year-Month
average_comment_length <- reviews %>%
  mutate(comment_length = nchar(comments)) %>%
  group_by(year_month) %>%
  summarise(avg_comment_length = mean(comment_length, na.rm = TRUE), .groups = "drop") %>%
  filter(!is.na(avg_comment_length))  # Remove rows with NA values

# Extract the year for faceting
average_comment_length <- average_comment_length %>%
  mutate(year = substr(year_month, 1, 4))

# Plot the average length of comments per Year-Month (faceted by Year)
ggplot(average_comment_length, aes(x = year_month, y = avg_comment_length)) +
  geom_bar(stat = "identity", fill = "lightgreen", color = "black", alpha = 0.7) +
  geom_text(aes(label = round(avg_comment_length, 1)), vjust = -0.5, size = 3) +  # Add labels to bars
  facet_wrap(~ year, scales = "free_x") +  # Facet by Year
  labs(
    title = "Average Length of Comments per Month (Faceted by Year)",
    x = "Year-Month",
    y = "Average Comment Length (Characters)"
  ) +
  scale_y_continuous(limits = c(0, max(average_comment_length$avg_comment_length, na.rm = TRUE)), expand
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis labels for readability
    strip.text.x = element_text(angle = 90)  # Rotate facet labels for better readability
  )
```



As we can see, the average length of comments has significantly decreased over time. This could be due to individuals engaging with information in shorter bursts, reducing the likelihood in investing time in writing or reading long comments.

Average Length of Commemts over Time

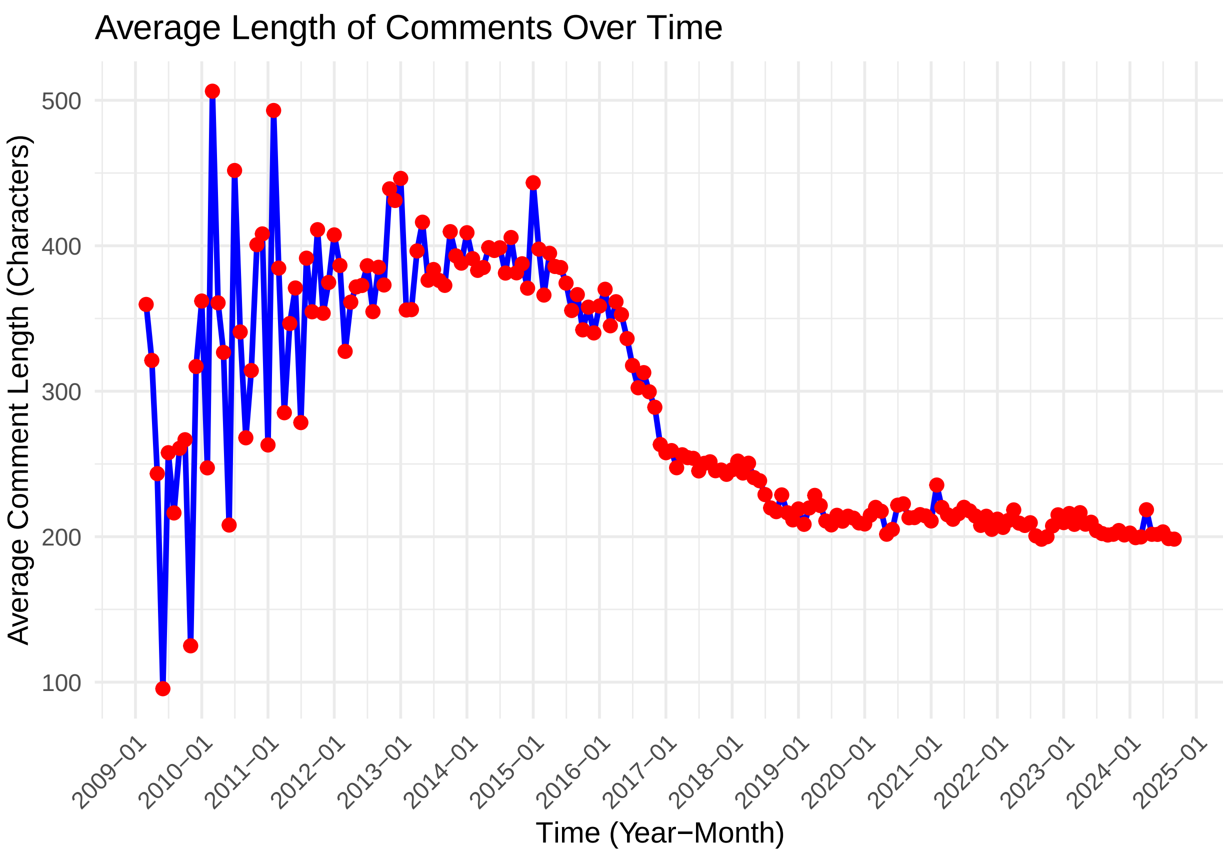
```
library(dplyr)
library(ggplot2)

# Calculate the average comment length for each Year-Month
average_comment_length <- reviews %>%
  mutate(comment_length = nchar(comments)) %>%
  group_by(year_month) %>%
  summarise(avg_comment_length = mean(comment_length, na.rm = TRUE), .groups = "drop")
print(average_comment_length)
```

```
## # A tibble: 187 x 2
##    year_month avg_comment_length
##    <chr>                   <dbl>
##  1 2009-03                 360.
##  2 2009-04                 321.
##  3 2009-05                 243.
##  4 2009-06                  95.5
##  5 2009-07                 258.
##  6 2009-08                 216.
##  7 2009-09                 261.
##  8 2009-10                 267.
##  9 2009-11                 125
```

```
## 10 2009-12                  317
## # i 177 more rows
```

```
# Plot the average length of comments over time
ggplot(average_comment_length, aes(x = as.Date(paste0(year_month, "-01")), y = avg_comment_length)) +
  geom_line(color = "blue", size = 1) +  # Line chart
  geom_point(color = "red", size = 2) +  # Points on the line to highlight data points
  labs(title = "Average Length of Comments Over Time",
       x = "Time (Year-Month)",
       y = "Average Comment Length (Characters)") +
  scale_x_date(date_labels = "%Y-%m", date_breaks = "1 year") +  # Format x-axis to show year-month
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis labels for readability
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

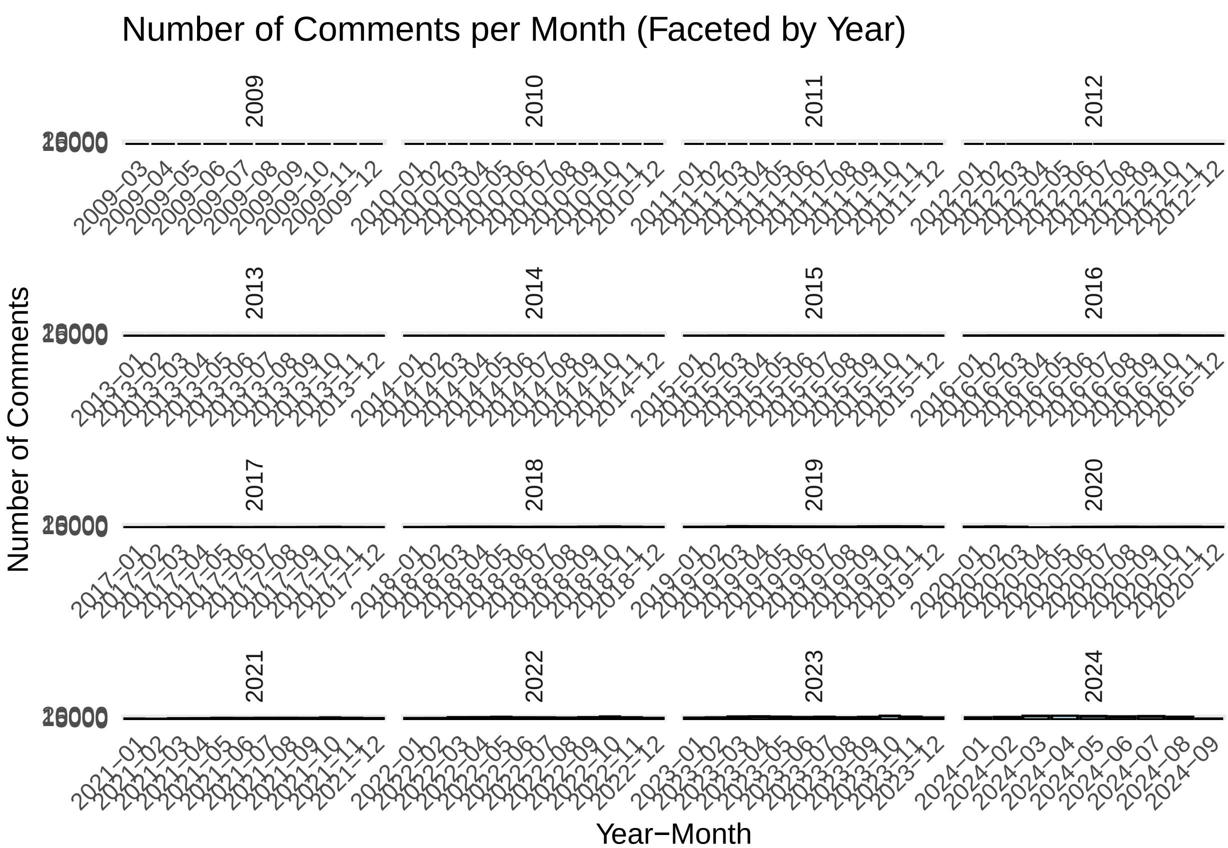


This visual may be easier to interpret as it clearly shows the peaks and dips in average length of comments over time.

Plot the Number of Comments per Month per Year

```
# Create the `comments_per_month` data frame
comments_per_month <- reviews %>%
  group_by(year_month) %>%  # Group by Year-Month
```

```r
  summarise(num_comments = n(), .groups = "drop") %>%  # Count the number of comments
  mutate(year = substr(year_month, 1, 4))  # Extract year for faceting

# Plot the Count of Comments per Year-Month, Faceted by Year
library(ggplot2)

ggplot(comments_per_month, aes(x = year_month, y = num_comments)) +
  geom_bar(stat = "identity", fill = "lightblue", color = "black", alpha = 0.7) +
  geom_text(aes(label = num_comments), vjust = -0.5, size = 3) +
  facet_wrap(~ year, scales = "free_x") +  # Facet by Year
  labs(title = "Number of Comments per Month (Faceted by Year)",
       x = "Year-Month",
       y = "Number of Comments") +
  scale_y_continuous(limits = c(0, 20000)) +  # Adjust Y-axis limit as needed
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),    # Rotate x-axis labels
        strip.text.x = element_text(angle = 90))  # Rotate facet labels
```



Unlike with the average length of comments over time, we see the number of comments over time increase.

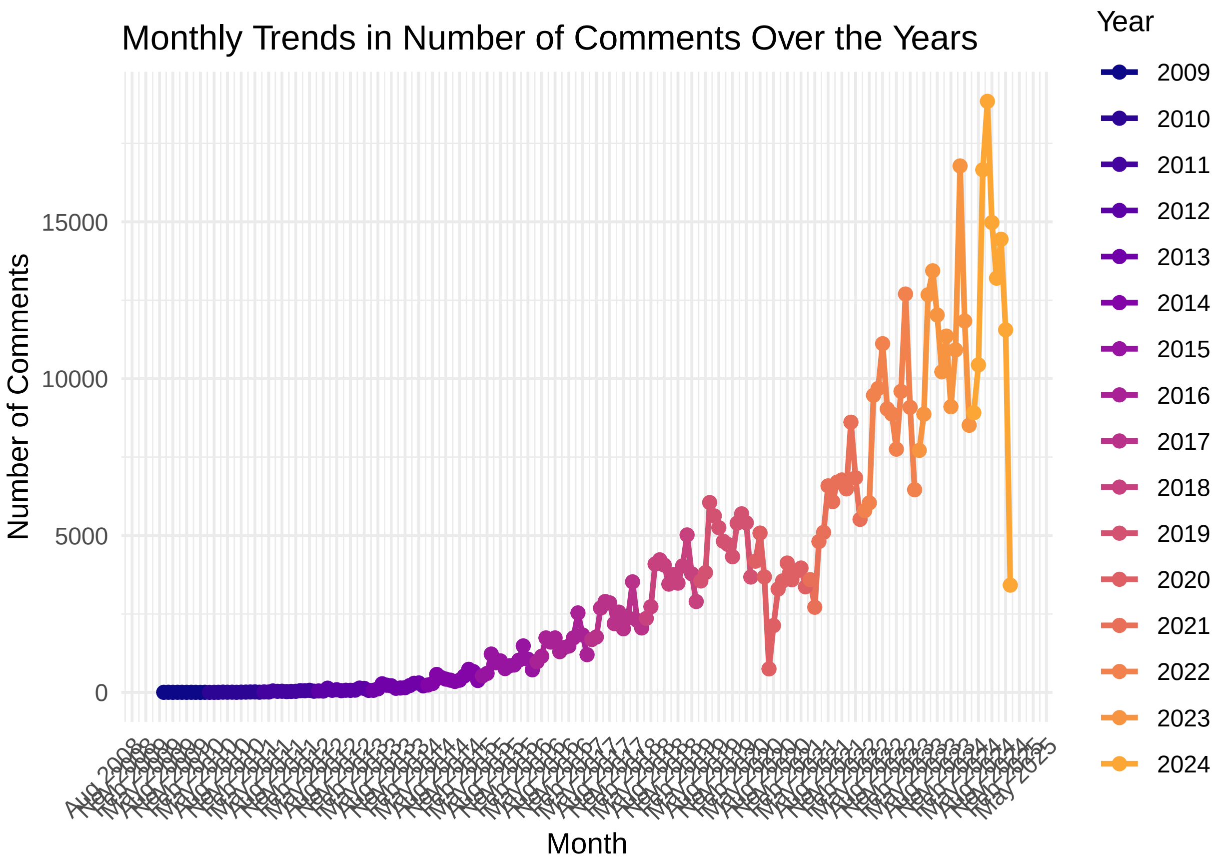
Plot the Increase of Comments over Time

```r
# Filter the data to include only dates up to August 2024
comments_per_month_filtered <- comments_per_month %>%
  filter(as.Date(paste0(year_month, "-01")) <= as.Date("2024-08-31"))

# Calculate percent change (percentage increase) from the previous month
```

```r
comments_per_month_filtered <- comments_per_month_filtered %>%
  arrange(as.Date(paste0(year_month, "-01"))) %>%
  mutate(percent_change = (num_comments - lag(num_comments)) / lag(num_comments) * 100)  # Percent chan

# View the percent change data
comments_per_month_filtered %>%
  select(year_month, num_comments, percent_change)
```

```
## # A tibble: 186 x 3
##    year_month num_comments percent_change
##    <chr>             <int>          <dbl>
##  1 2009-03               3           NA
##  2 2009-04               5           66.7
##  3 2009-05               3          -40
##  4 2009-06               4           33.3
##  5 2009-07               4            0
##  6 2009-08               3          -25
##  7 2009-09               4           33.3
##  8 2009-10               3          -25
##  9 2009-11               2          -33.3
## 10 2009-12               4          100
## # i 176 more rows
```

```r
# Plot this
ggplot(comments_per_month, aes(x = as.Date(paste0(year_month, "-01")), y = num_comments, color = as.fac
  geom_line(size = 1) +
  geom_point(size = 2) +  # Add points for emphasis
  scale_color_viridis_d(option = "plasma", begin = 0, end = 0.8) +  # Use a visually appealing color pa
  labs(title = "Monthly Trends in Number of Comments Over the Years",
       x = "Month",
       y = "Number of Comments",
       color = "Year") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_x_date(date_labels = "%b %Y", date_breaks = "3 months")  # Format x-axis with month and year
```



Here, we can see the peaks and dips in the number of comments. The dip off at the end is due to incomplete data (the data has yet to be added past mid August 2024).

Part 3: Analyzing the Most Common Words and TF-IDF

Calculate the Most Common Words and TF-IDF

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

# Clean the comments column
cleaned_comments <- reviews %>%
  mutate(comments = str_to_lower(comments),          # Convert to lowercase
         comments = str_replace_all(comments, "[[:punct:]]", " "), # Remove punctuation
         comments = str_replace_all(comments, "[[:digit:]]", ""),  # Remove numbers
         comments = str_replace_all(comments, "\\s+", " "))        # Remove extra spaces

# Remove stop words
data("stop_words")  # Load default stop words
cleaned_comments <- cleaned_comments %>%
  unnest_tokens(word, comments) %>%                # Tokenize comments
  anti_join(stop_words)                            # Remove stop words
```

```
## Joining with `by = join_by(word)`
```

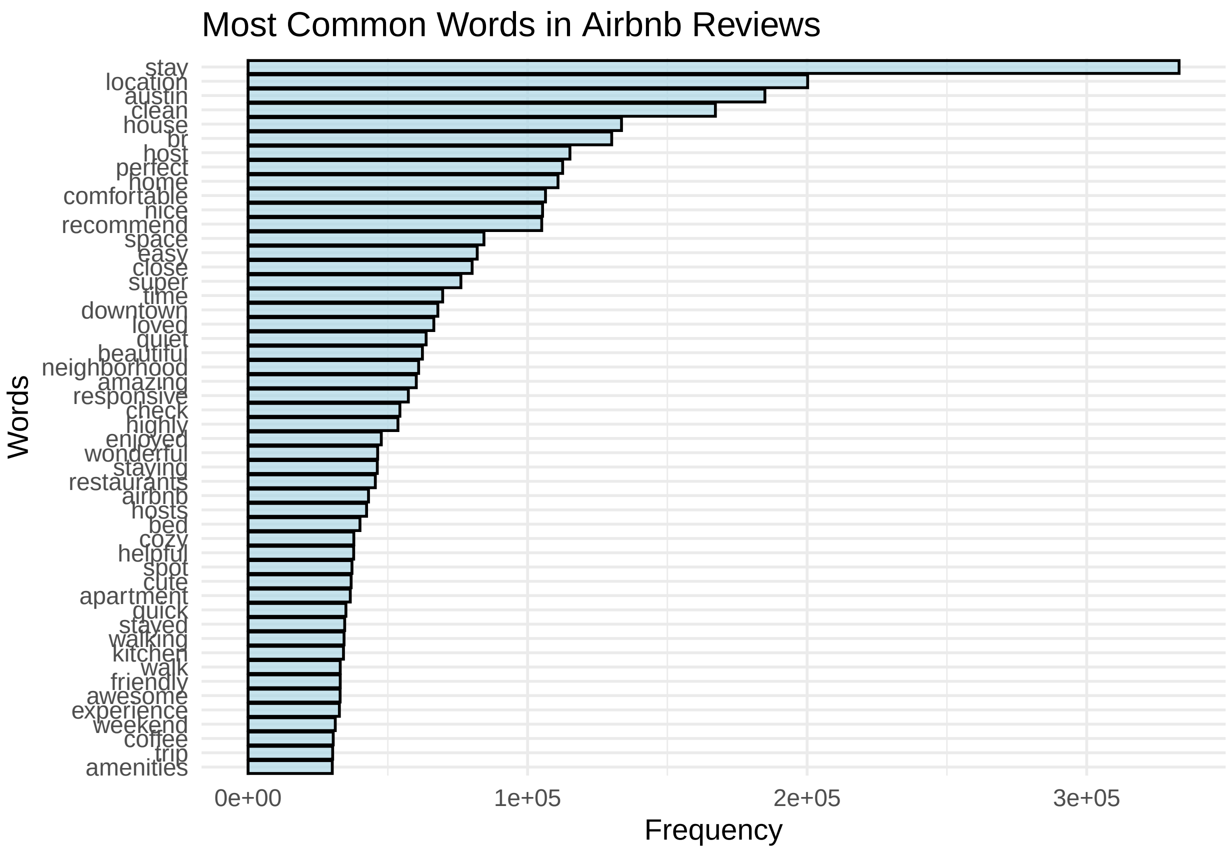
```r
# Count the most common words across all reviews
common_words <- cleaned_comments %>%
  count(word, sort = TRUE) %>%
  top_n(50, n)

# Calculate TF-IDF (Term Frequency-Inverse Document Frequency)
tf_idf <- cleaned_comments %>%
  count(listing_id, word) %>%                          # Count word occurrences per listing
  bind_tf_idf(word, listing_id, n) %>%                 # Calculate TF-IDF
  arrange(desc(tf_idf))                                # Sort by descending TF-IDF score

# Display the most common words
print(head(common_words))
```

```
## # A tibble: 6 x 2
##   word          n
##   <chr>     <int>
## 1 stay     333046
## 2 location 200180
## 3 austin   184876
## 4 clean    167176
## 5 house    133588
## 6 br       130075
```

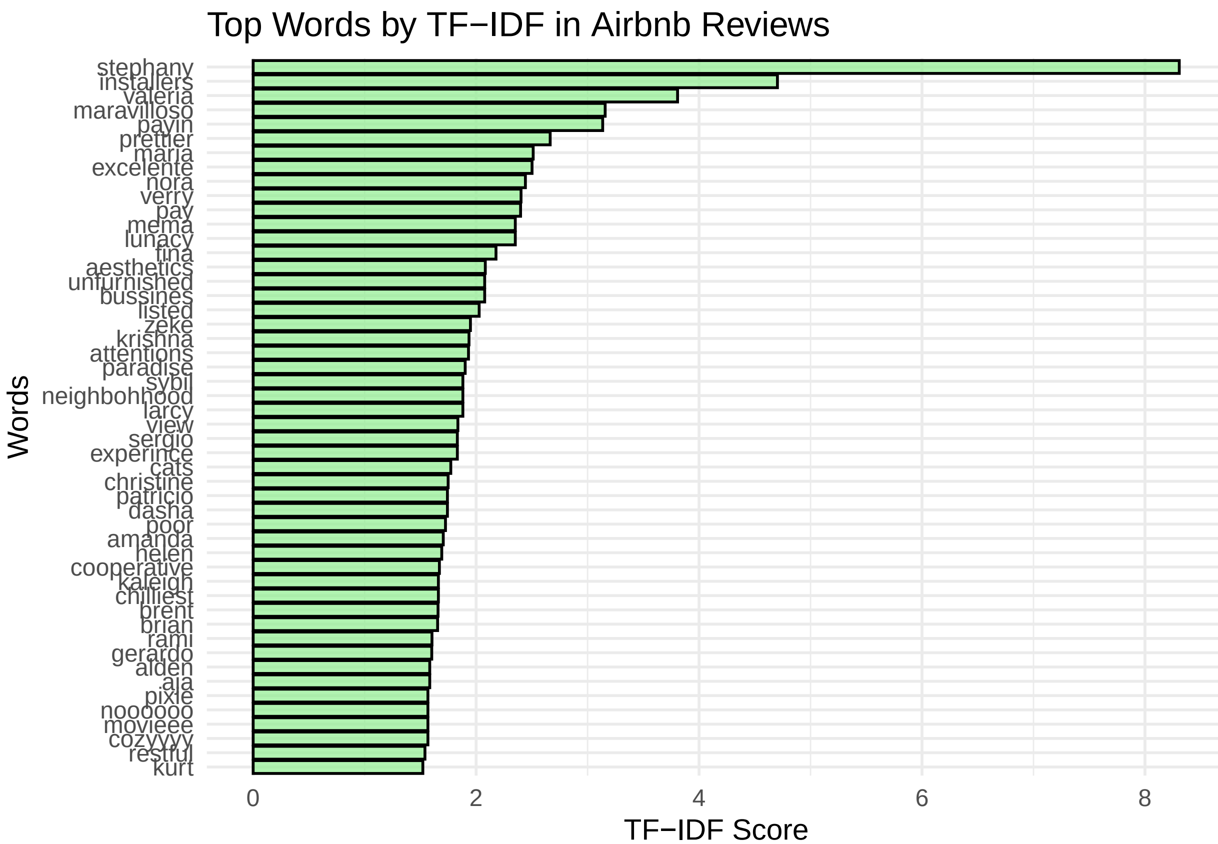
```r
# Display the top 10 words by TF-IDF
print(head(tf_idf))
```

```
## # A tibble: 6 x 6
##   listing_id word             n    tf   idf tf_idf
##        <dbl> <chr>        <int> <dbl> <dbl>  <dbl>
## 1    9.92e17 stephany         1 1      8.31   8.31
## 2    8.16e17 installers       1 0.5    9.41   4.70
## 3    9.88e17 valeria          1 0.5    7.61   3.81
## 4    9.19e17 maravilloso      1 0.5    6.32   3.16
## 5    1.18e18 payin            1 0.333  9.41   3.14
## 6    4.24e 7 prettier         1 0.5    5.33   2.66
```

```r
# Visualize the Most Common Words
top_common_words <- common_words %>%
  filter(n > 100) %>%   # Filter for words with frequency greater than 100
  ggplot(aes(x = reorder(word, n), y = n)) +
  geom_col(fill = "lightblue", color = "black", alpha = 0.7) +
  coord_flip() +
  labs(title = "Most Common Words in Airbnb Reviews", x = "Words", y = "Frequency") +
  theme_minimal()

# Display the plot
print(top_common_words)
```




```r
# Visualize the top words by TF-IDF
top_tf_idf_words <- tf_idf %>%
  top_n(50, tf_idf) %>%   # Show top 10 words by TF-IDF score
  ggplot(aes(x = reorder(word, tf_idf), y = tf_idf)) +
  geom_col(fill = "lightgreen", color = "black", alpha = 0.7) +
  coord_flip() +
  labs(title = "Top Words by TF-IDF in Airbnb Reviews", x = "Words", y = "TF-IDF Score") +
  theme_minimal()

# Display the plot
print(top_tf_idf_words)
```



We identify "stay," "location," and "Austin" as the top three most common words. Other commonly used words, such as "clean," "comfortable," and "nice," may offer valuable insights into what Airbnb guests prioritize. Although the TF-IDF analysis is less informative, it reveals that guests often mention their host's name, suggesting that hosts significantly influence customer satisfaction or dissatisfaction.

Part 4: Sentiment Analysis

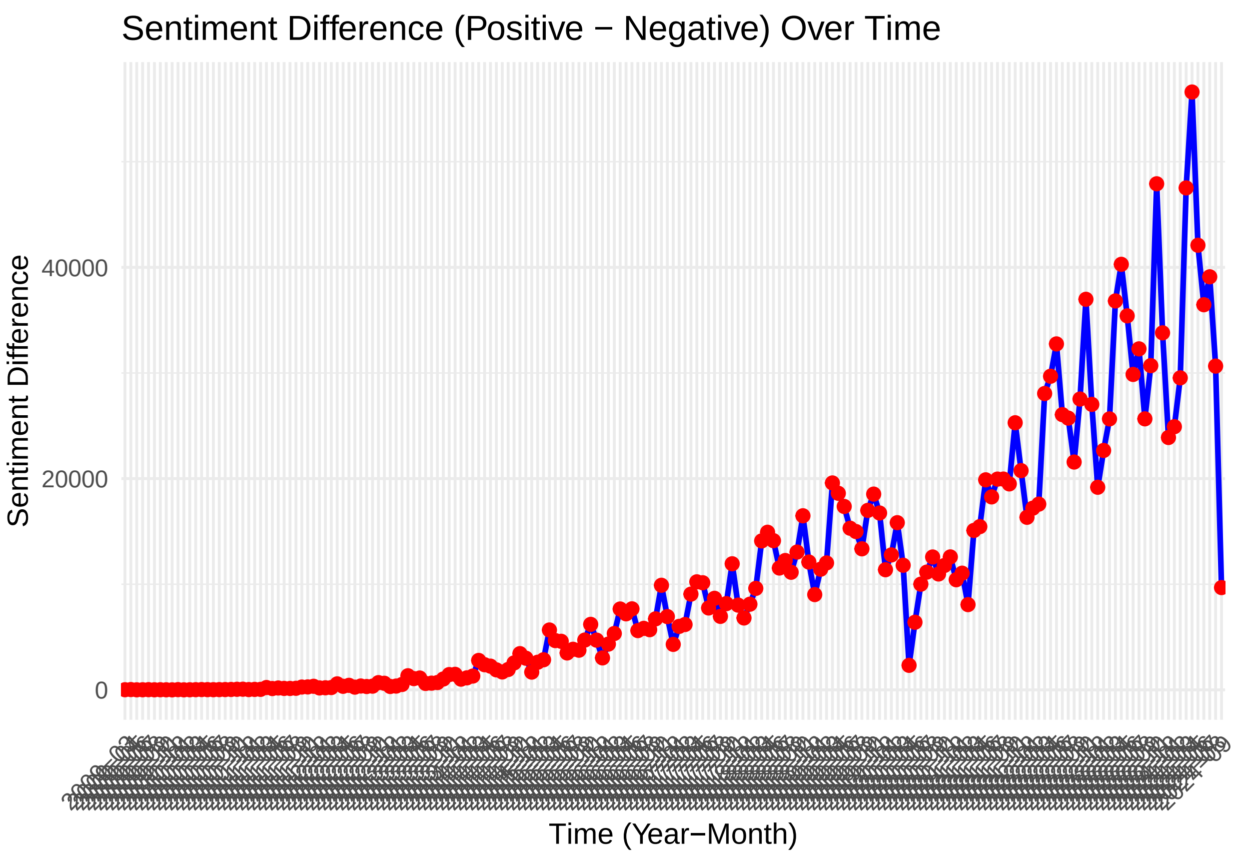
Plotting the Sentiment Difference Over Time

```r
# Load libraries
library(tidytext)
library(dplyr)
library(ggplot2)
library(tidyr)

# Perform sentiment analysis
sentiment_analysis <- cleaned_comments %>%
  count(word, year_month, sort = TRUE) %>% # Count words by year_month
  inner_join(get_sentiments("bing"), by = join_by(word)) %>% # Join Bing lexicon
  group_by(year_month, sentiment) %>% # Group by year_month and sentiment
  summarize(n = sum(n), .groups = "drop") %>% # Aggregate counts
  spread(sentiment, n, fill = 0) %>% # Spread into positive/negative columns
  mutate(sentiment_diff = positive - negative) # Calculate sentiment difference
```

```
## Warning in inner_join(., get_sentiments("bing"), by = join_by(word)): Detected an unexpected many-to-
## i Row 538983 of `x` matches multiple rows in `y`.
## i Row 868 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.
```

```r
# Aggregate by year_month for the line plot
sentiment_analysis <- sentiment_analysis %>%
  group_by(year_month) %>%
  summarize(sentiment_diff = sum(sentiment_diff, na.rm = TRUE), .groups = "drop")

# Plot sentiment difference over time
ggplot(sentiment_analysis, aes(x = year_month, y = sentiment_diff, group = 1)) +
  geom_line(color = "blue", size = 1) +
  geom_point(color = "red", size = 2) +
  labs(title = "Sentiment Difference (Positive - Negative) Over Time",
       x = "Time (Year-Month)",
       y = "Sentiment Difference") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Here, we see that the sentiment in Airbnb comments has increased dramatically over time with a peak between 2022 and 2025. Again, the dip off at the end is due to incomplete data (the data has yet to be added past mid August 2024). This indicates that customers are leaving more positive reviews than ever before.

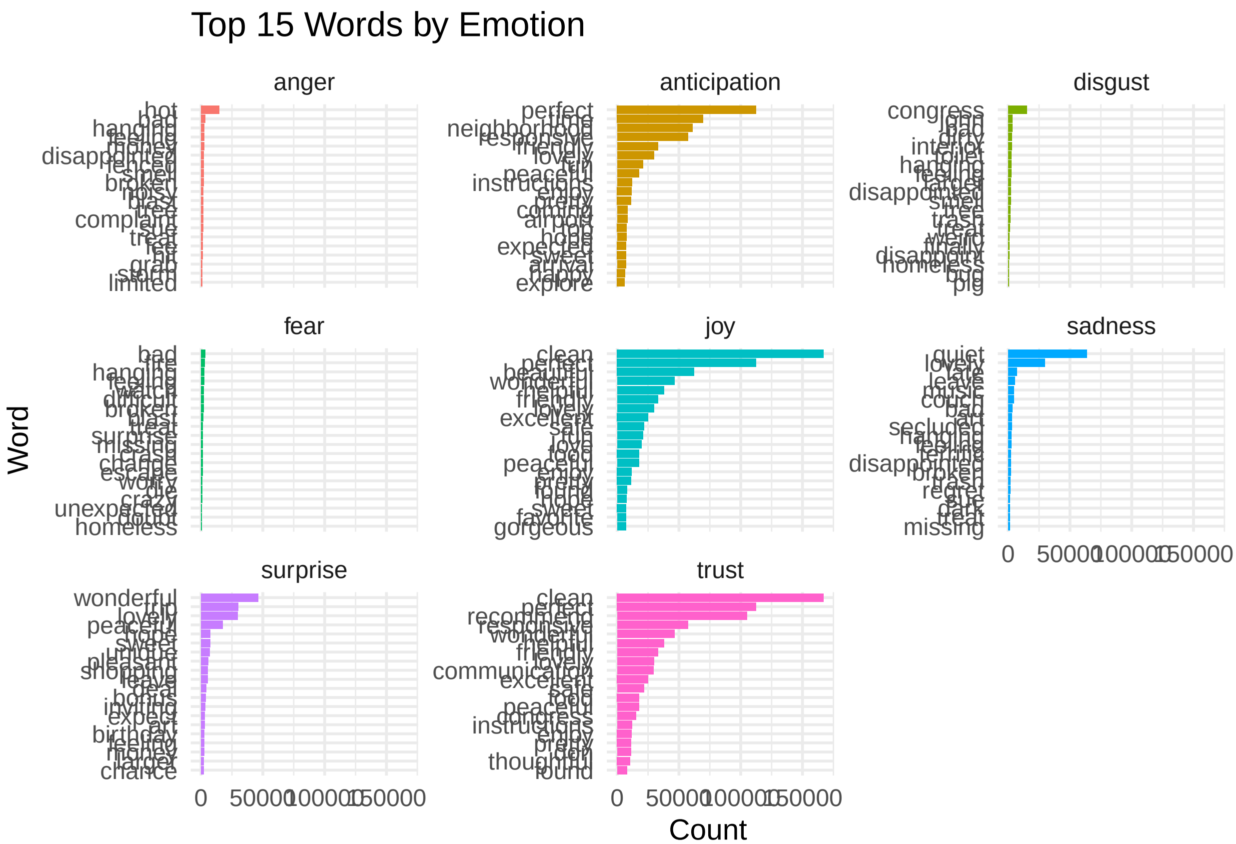
Identifying the Top Words by Emotion:

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

```r
# Perform sentiment analysis with the NRC lexicon for emotion analysis
emotion_sentiment <- cleaned_comments %>%
  inner_join(get_sentiments("nrc") %>% filter(!sentiment %in% c("positive", "negative"))) %>%  # Filter
  count(word, sentiment, sort = TRUE) %>%   # Count words by sentiment
  filter(n > 500)  # Keep words that appear at least 500 times, adjust this threshold if needed
```

```
## Joining with `by = join_by(word)`

## Warning in inner_join(., get_sentiments("nrc") %>% filter(!sentiment %in% : Detected an unexpected ma
## i Row 4 of `x` matches multiple rows in `y`.
## i Row 3614 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.
```

```r
# Get the top 15 words per emotion
top_emotion_words <- emotion_sentiment %>%
  group_by(sentiment) %>%  # Group by emotion
  slice_max(n, n = 20)     # Select top 15 most frequent words for each emotion

# Plot the top words by emotion with words on the y-axis and slanted labels
top_words_plot <- top_emotion_words %>%
  ggplot(aes(y = reorder(word, n), x = n, fill = sentiment)) +  # Swap x and y axes
  geom_bar(stat = "identity", show.legend = FALSE) +  # Bar plot with word count
  facet_wrap(~ sentiment, scales = "free_y") +  # Facet by emotion
  labs(title = "Top 15 Words by Emotion", x = "Count", y = "Word") +  # Axis labels
  theme_minimal() +
  theme(axis.text.y = element_text(angle = 0, hjust = 1))  # Slant y-axis labels for better readability

# Display the plot
print(top_words_plot)
```



It's interesting to compare words associated with opposing emotions. For instance, "clean" (linked to joy and trust) contrasts with "dirty" (associated with disgust). Additionally, some words, such as "pig," "sue," "crash," "die," and "homeless," were unexpected in the analysis.

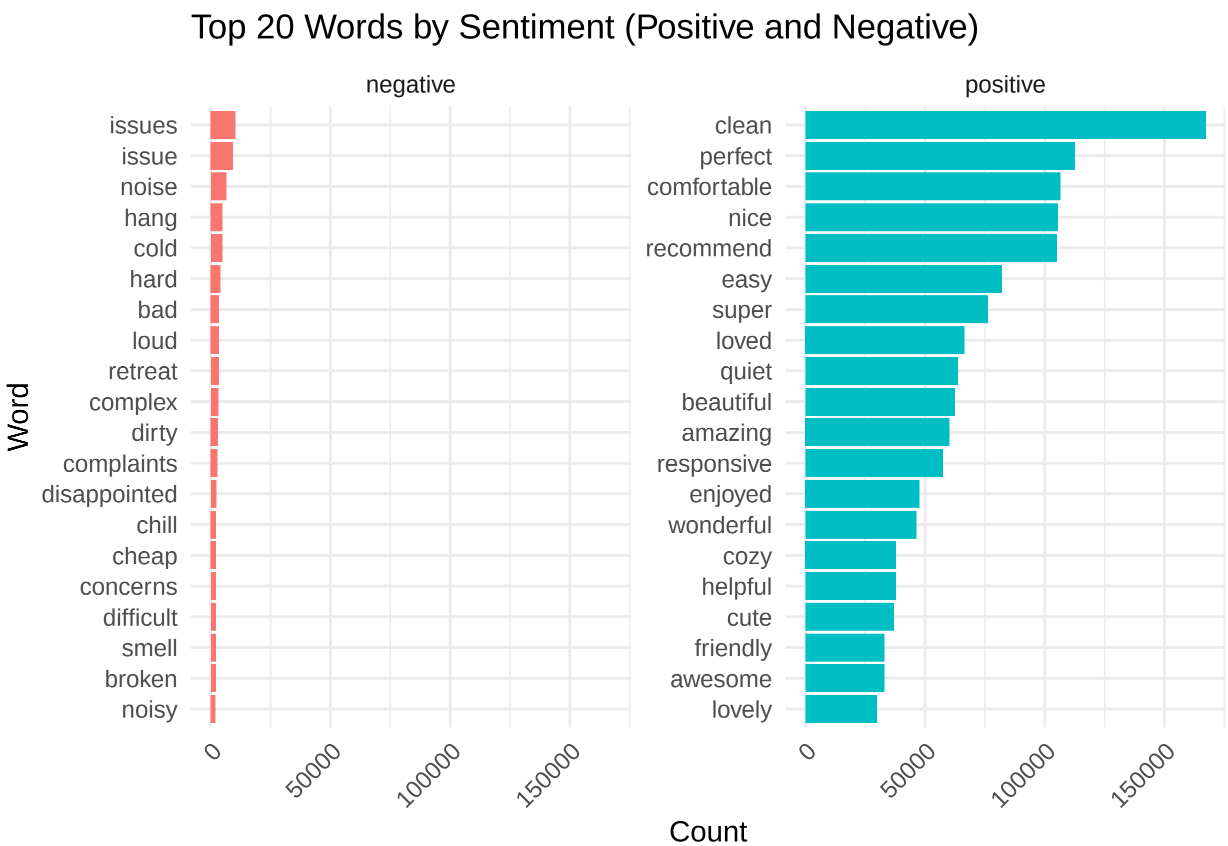
Identify Positive and Negative Words

```
# Perform sentiment analysis using the 'bing' lexicon for positive/negative words
sentiment_words <- cleaned_comments %>%
  inner_join(get_sentiments("bing")) %>%  # Join with the 'bing' sentiment lexicon
  count(word, sentiment, sort = TRUE)  # Count words by sentiment
```

```
## Joining with `by = join_by(word)`

## Warning in inner_join(., get_sentiments("bing")): Detected an unexpected many-to-many relationship be
## i Row 1195880 of `x` matches multiple rows in `y`.
## i Row 2929 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.
```

```
# Filter to get the top 20 words for positive and negative sentiments
top_sentiment_words <- sentiment_words %>%
  group_by(sentiment) %>%  # Group by sentiment (positive/negative)
  slice_max(n, n = 20)  # Select top 20 most frequent words for each sentiment

# Visualize the top 20 positive and negative words
sentiment_words_plot <- top_sentiment_words %>%
  ggplot(aes(x = reorder(word, n), y = n, fill = sentiment)) +  # Reorder by count
  geom_bar(stat = "identity", show.legend = FALSE) +  # Bar plot
  facet_wrap(~ sentiment, scales = "free_y") +  # Facet by sentiment (positive/negative)
```

```
  coord_flip() +  # Flip axes for better readability
  labs(title = "Top 20 Words by Sentiment (Positive and Negative)", x = "Word", y = "Count") +  # Label
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Slant x-axis labels for readability

# Display the plot
print(sentiment_words_plot)
```




```
# Optionally, display the most common positive and negative words
print(top_sentiment_words)
```

```
## # A tibble: 40 x 3
## # Groups:   sentiment [2]
##    word    sentiment     n
##    <chr>   <chr>     <int>
##  1 issues  negative  10364
##  2 issue   negative   9204
##  3 noise   negative   6440
##  4 hang    negative   4858
##  5 cold    negative   4751
##  6 hard    negative   4024
##  7 bad     negative   3399
##  8 loud    negative   3351
##  9 retreat negative   3309
## 10 complex negative   3128
## # i 30 more rows
```

Here, we conducted a similar analysis but categorized the top words into positive or negative sentiment using the Bing lexicon. Interestingly, the word "quiet" is classified as positive in the Bing lexicon, whereas in the NRC lexicon, it is associated with sadness and presumed negative. This analysis highlights customers' preferences and dislikes. They seem to favor a beautiful and quiet stay over a loud and noisy one. They also prefer an easy stay over a complex one, which may refer to the check-in and check-out process.

Part 5: Answering the Research Questions

Question 1: How do Airbnb reviews reflect customers' attitudes towards diversity and inclusion in Austin?

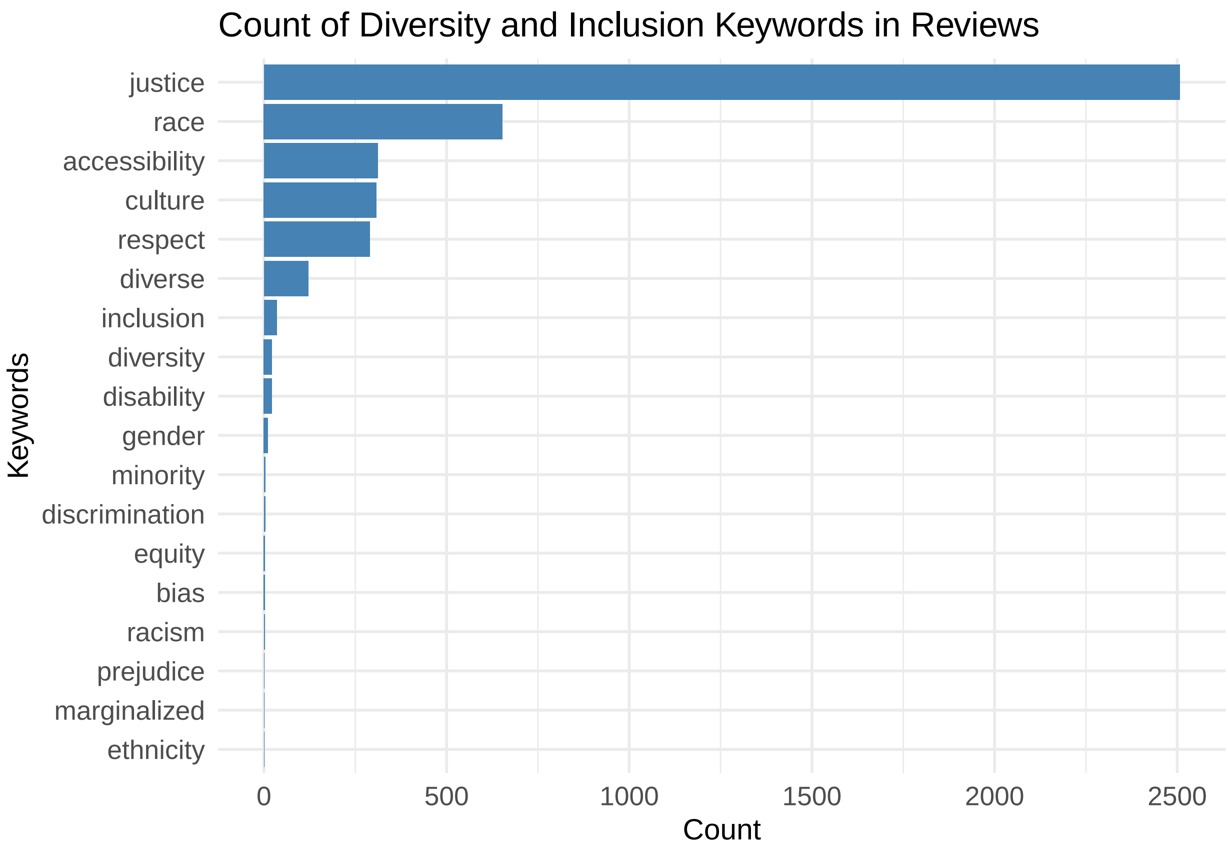
Search for Diversity Words

```r
library(dplyr)

# Define a list of keywords related to diversity and inclusion
keywords_di <- c("diverse", "inclusion", "culture", "accessibility", "respect",
                 "ethnicity", "race", "racism", "sexism", "homophobia",
                 "transphobia", "discrimination", "equality", "justice",
                 "prejudice", "bias", "marginalized", "minority", "equity",
                 "gender", "sexuality", "disability", "intersectionality", "diversity")

# Count occurrences of these words in reviews
di_words_count <- cleaned_comments %>%
  filter(word %in% keywords_di) %>%
  count(word, sort = TRUE)

# Display results
di_words_count
```

```
## # A tibble: 18 x 2
##    word                n
##    <chr>           <int>
##  1 justice          2507
##  2 race              653
##  3 accessibility     312
##  4 culture           308
##  5 respect           290
##  6 diverse           121
##  7 inclusion          35
##  8 disability         22
##  9 diversity          22
## 10 gender             11
## 11 discrimination      4
## 12 minority            4
## 13 bias                3
## 14 equity              3
## 15 racism              2
## 16 ethnicity           1
## 17 marginalized        1
## 18 prejudice           1
```

```r
# Create a bar chart
ggplot(di_words_count, aes(x = reorder(word, n), y = n)) +
  geom_bar(stat = "identity", fill = "steelblue") +
```

```r
  coord_flip() +
  labs(title = "Count of Diversity and Inclusion Keywords in Reviews",
       x = "Keywords",
       y = "Count") +
  theme_minimal() +
  theme(axis.text.x = element_text(size = 10),
        axis.text.y = element_text(size = 10))
```

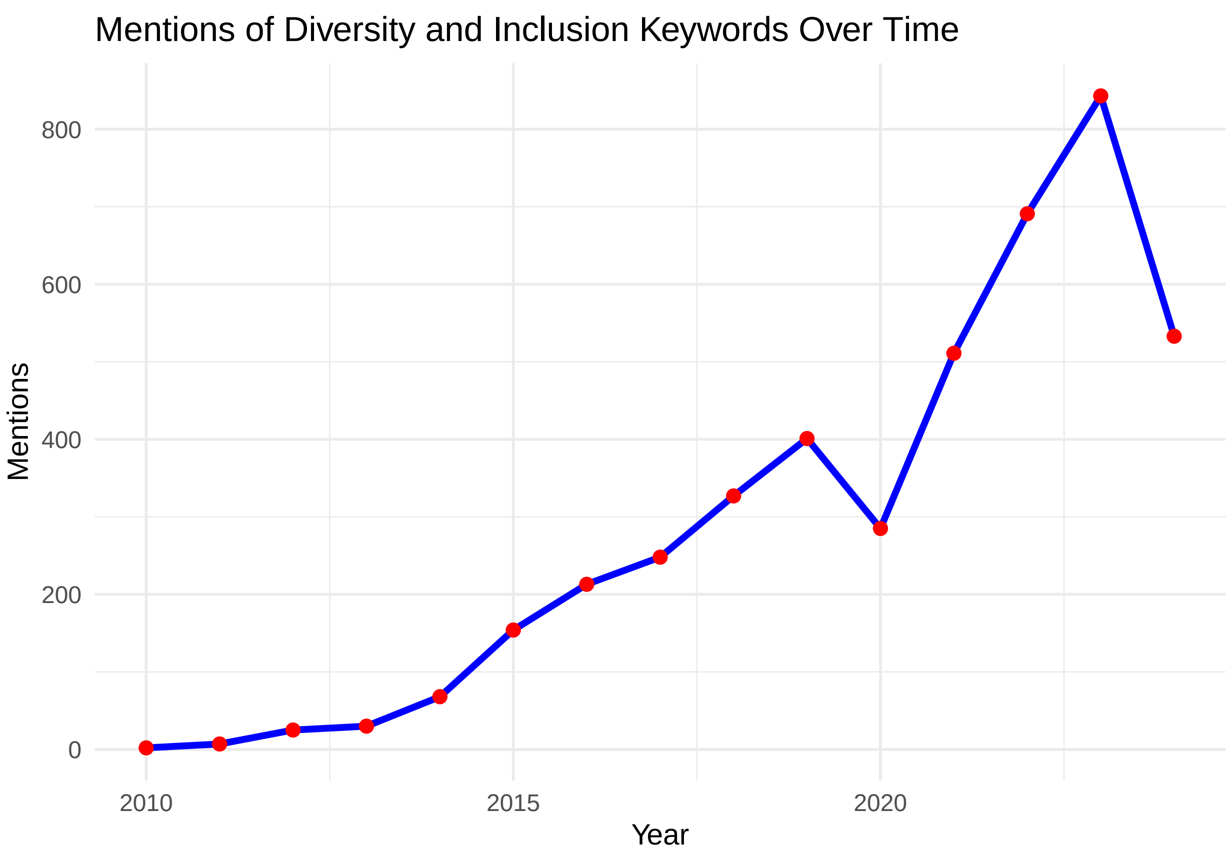


Surprisingly, "justice" was the top diversity word within the comments followed by "race," "accessibility," "culture," "respect," "diverse," and "inclusion."

Plot the Diversity Words Changing Over Years

```r
library(dplyr)
library(ggplot2)

# Define keywords related to diversity and inclusion
keywords_di <- c("diverse", "inclusion", "culture", "accessibility", "respect",
                 "ethnicity", "race", "racism", "sexism", "homophobia",
                 "transphobia", "discrimination", "equality", "justice",
                 "prejudice", "bias", "marginalized", "minority", "equity",
                 "gender", "sexuality", "disability", "intersectionality", "diversity",
                 "inclusive")

# Extract the year from the "Year-Month" column
cleaned_comments <- cleaned_comments %>%
  mutate(year = as.integer(substr(year_month, 1, 4)))  # Extract year from "Year-Month" column
```

```r
# Filter comments for keywords and count occurrences per year
di_mentions_by_year <- cleaned_comments %>%
  filter(word %in% keywords_di) %>%
  group_by(year, word) %>%
  summarise(count = n(), .groups = "drop")

# Summarize total mentions of all keywords per year
di_mentions_yearly <- di_mentions_by_year %>%
  group_by(year) %>%
  summarise(total_mentions = sum(count), .groups = "drop")

# Plot mentions per year
ggplot(di_mentions_yearly, aes(x = year, y = total_mentions)) +
  geom_line(color = "blue", size = 1.2) +
  geom_point(color = "red", size = 2) +
  labs(
    title = "Mentions of Diversity and Inclusion Keywords Over Time",
    x = "Year",
    y = "Mentions"
  ) +
  theme_minimal()
```



We observe a significant increase in the use of what I have defined as "diversity words" after 2020. This rise could be attributed to heightened discussions following events such as the death of George Floyd.

Calculate Diversity Words Sentiment Score

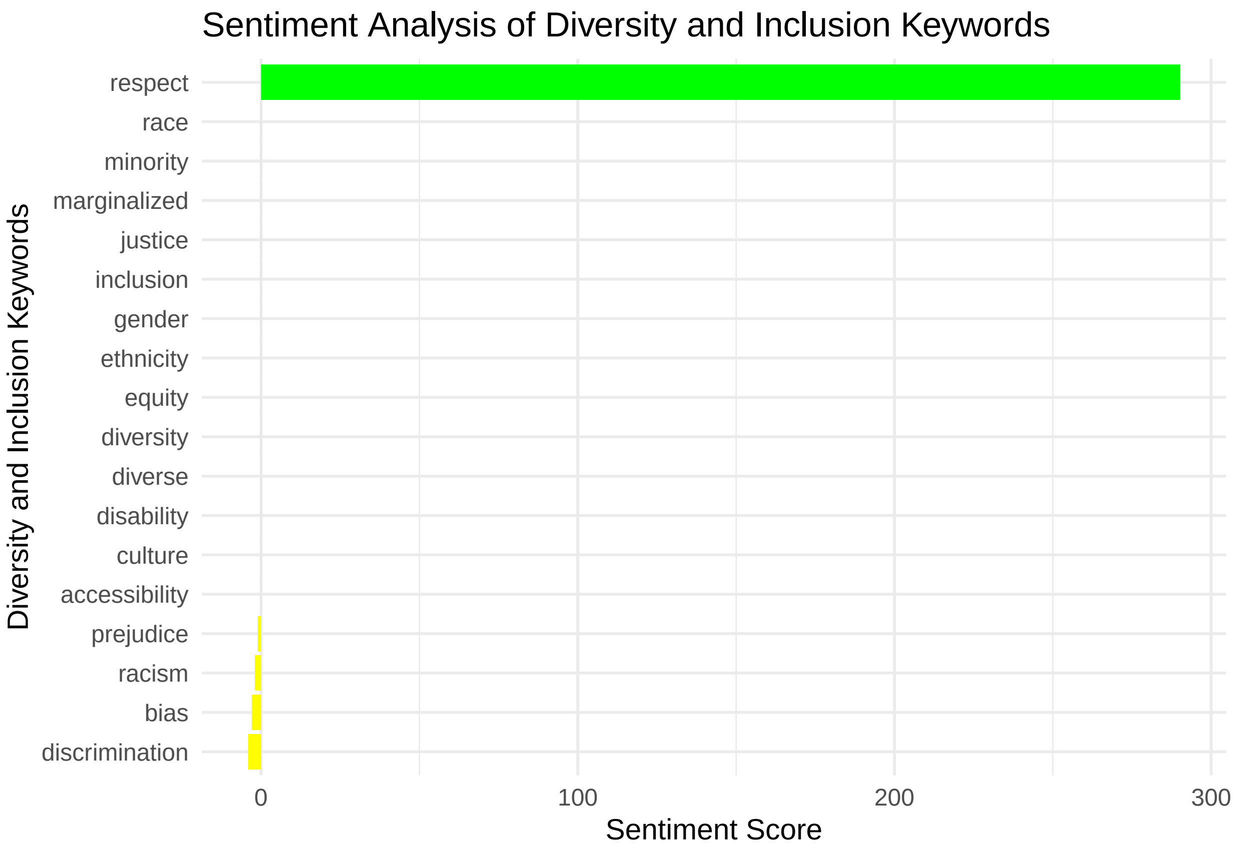
```r
library(dplyr)
library(tidytext)

# Perform sentiment analysis using the Bing lexicon
sentiment_lexicon <- get_sentiments("bing")  # You can also use "afinn" or "nrc"

# Assign numeric values to sentiment ('positive' = 1, 'negative' = -1)
sentiment_lexicon <- sentiment_lexicon %>%
  mutate(sentiment_score = ifelse(sentiment == "positive", 1, -1))

# Join sentiment lexicon with the counted words
di_sentiment <- di_words_count %>%
  left_join(sentiment_lexicon, by = "word") %>%
  group_by(word) %>%
  summarize(
    count = sum(n),  # Total count of occurrences
    sentiment_score = sum(n * sentiment_score, na.rm = TRUE)  # Sentiment score
  )

# Display results
di_sentiment
```

```
## # A tibble: 18 x 3
##    word           count sentiment_score
##    <chr>          <int>           <dbl>
##  1 accessibility    312               0
##  2 bias               3              -3
##  3 culture          308               0
##  4 disability        22               0
##  5 discrimination     4              -4
##  6 diverse          121               0
##  7 diversity         22               0
##  8 equity             3               0
##  9 ethnicity          1               0
## 10 gender            11               0
## 11 inclusion         35               0
## 12 justice         2507               0
## 13 marginalized       1               0
## 14 minority           4               0
## 15 prejudice          1              -1
## 16 race             653               0
## 17 racism             2              -2
## 18 respect          290             290
```

Plot Diversity Words Sentiment Score

```r
library(ggplot2)

# Plot the sentiment analysis results
ggplot(di_sentiment, aes(x = reorder(word, sentiment_score), y = sentiment_score, fill = sentiment_scor
  geom_bar(stat = "identity", show.legend = FALSE) +
  coord_flip() +  # Flip coordinates to make the plot horizontal
```

```
  labs(
    title = "Sentiment Analysis of Diversity and Inclusion Keywords",
    x = "Diversity and Inclusion Keywords",
    y = "Sentiment Score"
  ) +
  theme_minimal() +
  scale_fill_gradient2(low = "red", high = "green", mid = "yellow", midpoint = 0)
```



Interestingly, most diversity-related words do not have an associated sentiment score in the Bing lexicon. Words like "respect" carry a strong positive sentiment, while terms such as "prejudice," "racism," "bias," and "discrimination" are assigned moderate sentiment scores. I had anticipated these words to reflect a more intense sentiment.

Identify Diversity Words Emotions

```
library(tidytext)

# Use NRC lexicon for sentiment analysis
nrc_sentiment <- cleaned_comments %>%
  filter(word %in% keywords_di) %>%
  inner_join(get_sentiments("nrc")) %>%
  count(sentiment, sort = TRUE)
```

```
## Joining with `by = join_by(word)`

## Warning in inner_join(., get_sentiments("nrc")): Detected an unexpected many-to-many relationship be
## i Row 1 of `x` matches multiple rows in `y`.
## i Row 7326 of `y` matches multiple rows in `x`.
```

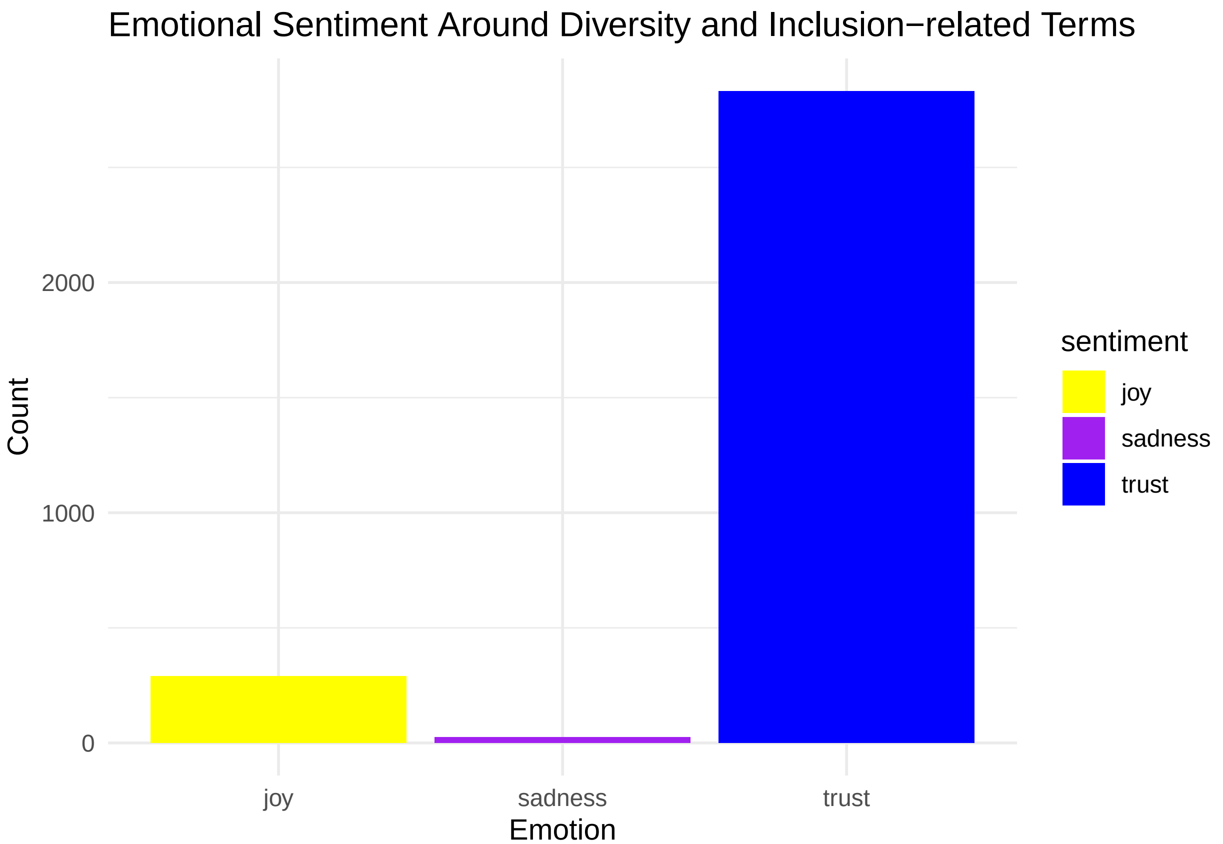
```
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```r
# View sentiment counts
nrc_sentiment
```

```
## # A tibble: 9 x 2
##   sentiment        n
##   <chr>        <int>
## 1 positive      3267
## 2 trust         2832
## 3 anticipation   290
## 4 joy            290
## 5 negative       155
## 6 sadness         26
## 7 anger            8
## 8 disgust          4
## 9 fear             4
```

Plot Diversity Words Emotions

```r
nrc_emotions <- nrc_sentiment %>%
  filter(sentiment %in% c("trust", "joy", "sadness"))

ggplot(nrc_emotions, aes(x = sentiment, y = n, fill = sentiment)) +
  geom_bar(stat = "identity") +
  labs(title = "Emotional Sentiment Around Diversity and Inclusion-related Terms",
       x = "Emotion", y = "Count") +
  theme_minimal() +
  scale_fill_manual(values = c("trust" = "blue", "joy" = "yellow", "sadness" = "purple"))
```



Most diversity-related words were identified as positive and associated with trust according to the NRC lexicon.

Customer Segmentation Based on Diversity and Inclusion Sentiments

```
# Libraries
library(dplyr)
library(tidytext)
library(textdata)
library(stringr)
library(cluster)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
# Step 1: Define custom list of diversity and inclusion-related keywords
keywords_di <- c("diverse", "inclusion", "culture", "accessibility", "respect",
                 "ethnicity", "race", "racism", "sexism", "homophobia",
                 "transphobia", "discrimination", "equality", "justice",
                 "prejudice", "bias", "marginalized", "minority", "equity",
                 "gender", "sexuality", "disability", "intersectionality",
                 "diversity", "inclusive")

# Step 2: Preprocess and clean text, filtering for diversity and inclusion-related keywords
cleaned_reviews_di <- reviews %>%
  unnest_tokens(word, comments) %>%
  anti_join(stop_words) %>%
  filter(!str_detect(word, "[0-9]")) %>%
```

```r
  filter(word %in% keywords_di)  # Filter for diversity & inclusion words only
```

```
## Joining with `by = join_by(word)`
```

```r
# Step 3: Sentiment analysis using the Bing lexicon (only for relevant words)
sentiments_di <- cleaned_reviews_di %>%
  inner_join(get_sentiments("bing")) %>%
  group_by(reviewer_id) %>%
  summarize(sentiment_score = sum(if_else(sentiment == "positive", 1, -1)))  # Calculate sentiment scor
```

```
## Joining with `by = join_by(word)`
```

```r
# Step 4: Merge sentiment scores with customer profiles
customer_profiles_di <- reviews %>%
  left_join(sentiments_di, by = "reviewer_id") %>%
  group_by(reviewer_id) %>%
  summarize(
    total_reviews = n(),
    avg_sentiment = mean(sentiment_score, na.rm = TRUE),  # Average sentiment score
    review_length = mean(nchar(comments), na.rm = TRUE),  # Average review length
    positive_review_count = sum(if_else(sentiment_score > 0, 1, 0)),  # Positive review count
    negative_review_count = sum(if_else(sentiment_score < 0, 1, 0))  # Negative review count
  )

# Step 5: Clean the data by removing rows with NA, NaN, or Inf values
customer_profiles_di_clean <- customer_profiles_di %>%
  filter_all(all_vars(!is.na(.) & !is.infinite(.) & !is.nan(.)))  # Remove rows with NA, Inf, or NaN va

# Step 6: Scale the data for clustering (excluding reviewer_id)
scaled_data_di <- scale(customer_profiles_di_clean %>% select(-reviewer_id))  # Exclude reviewer_id fro

# Step 7: Apply K-Means clustering (you can adjust the number of clusters as needed)
set.seed(123)  # For reproducibility
kmeans_model_di <- kmeans(scaled_data_di, centers = 3, nstart = 25)

# Step 8: Add cluster labels to the customer profiles
customer_profiles_di_clean$cluster <- kmeans_model_di$cluster

# Step 9: Output the customer profiles with cluster labels as a data frame (non-visual output)
customer_profiles_summary_di <- customer_profiles_di_clean %>%
  group_by(cluster) %>%
  summarize(
    avg_sentiment = mean(avg_sentiment, na.rm = TRUE),
    avg_review_length = mean(review_length, na.rm = TRUE),
    avg_positive_reviews = mean(positive_review_count, na.rm = TRUE),
    avg_negative_reviews = mean(negative_review_count, na.rm = TRUE),
    total_customers_in_cluster = n()
  )

# Step 10: View the output summary of customer segmentation
print(customer_profiles_summary_di)
```

```
## # A tibble: 3 x 6
##   cluster avg_sentiment avg_review_length avg_positive_reviews
##     <int>         <dbl>             <dbl>                <dbl>
```

```
## 1       1         -1.14            1382.                0
## 2       2          1.02             636.                1.22
## 3       3          1                262.                7.44
## # i 2 more variables: avg_negative_reviews <dbl>,
## #   total_customers_in_cluster <int>
```

We segment reviewers into three clusters based on their engagement with diversity and inclusion-related keywords and their sentiment scores. The clusters reveal meaningful differences in reviewer behavior and sentiment:

Cluster 1: Reviewers with negative sentiment scores, longer reviews, and more mentions of diversity-related issues with a critical tone. Cluster 2: Reviewers with moderately positive sentiment, shorter reviews, and a balanced engagement with diversity topics. Cluster 3: Highly positive reviewers with the shortest reviews and the highest count of positive diversity-related mentions.

These clusters provide insights into how customers engage with diversity-related themes, helping identify areas for improved messaging or service.

Question 2: What are the key factors that influence customer satisfaction on the platform?

Identify Top Words in Positive and Negative Reviews and their Frequency

```r
# Find top words in positive and negative reviews
top_positive_words <- cleaned_comments %>%
  inner_join(get_sentiments("bing")) %>%
  filter(sentiment == "positive") %>%
  count(word, sort = TRUE) %>%
  top_n(10)
```

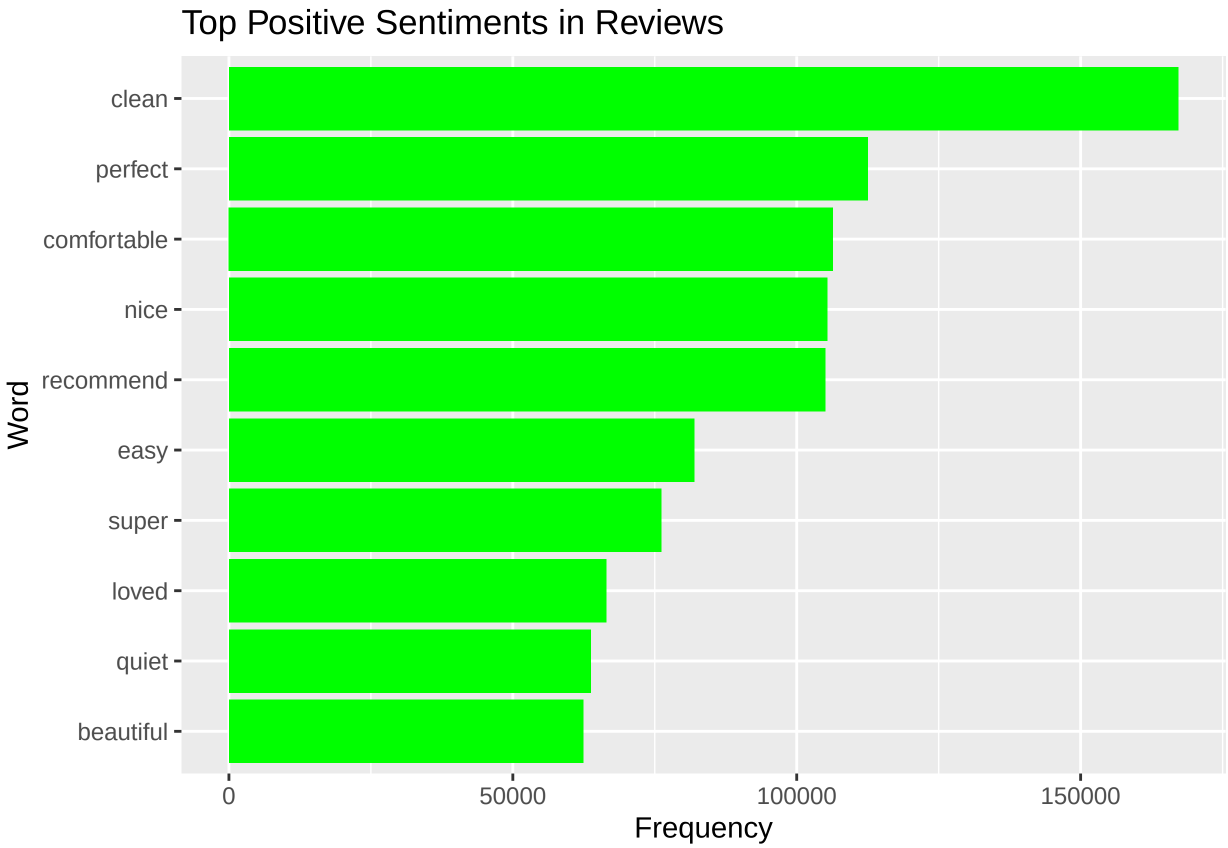
```
## Joining with `by = join_by(word)`

## Warning in inner_join(., get_sentiments("bing")): Detected an unexpected many-to-many relationship be
## i Row 1195880 of `x` matches multiple rows in `y`.
## i Row 2929 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.

## Selecting by n
```

```r
top_negative_words <- cleaned_comments %>%
  inner_join(get_sentiments("bing")) %>%
  filter(sentiment == "negative") %>%
  count(word, sort = TRUE) %>%
  top_n(10)
```

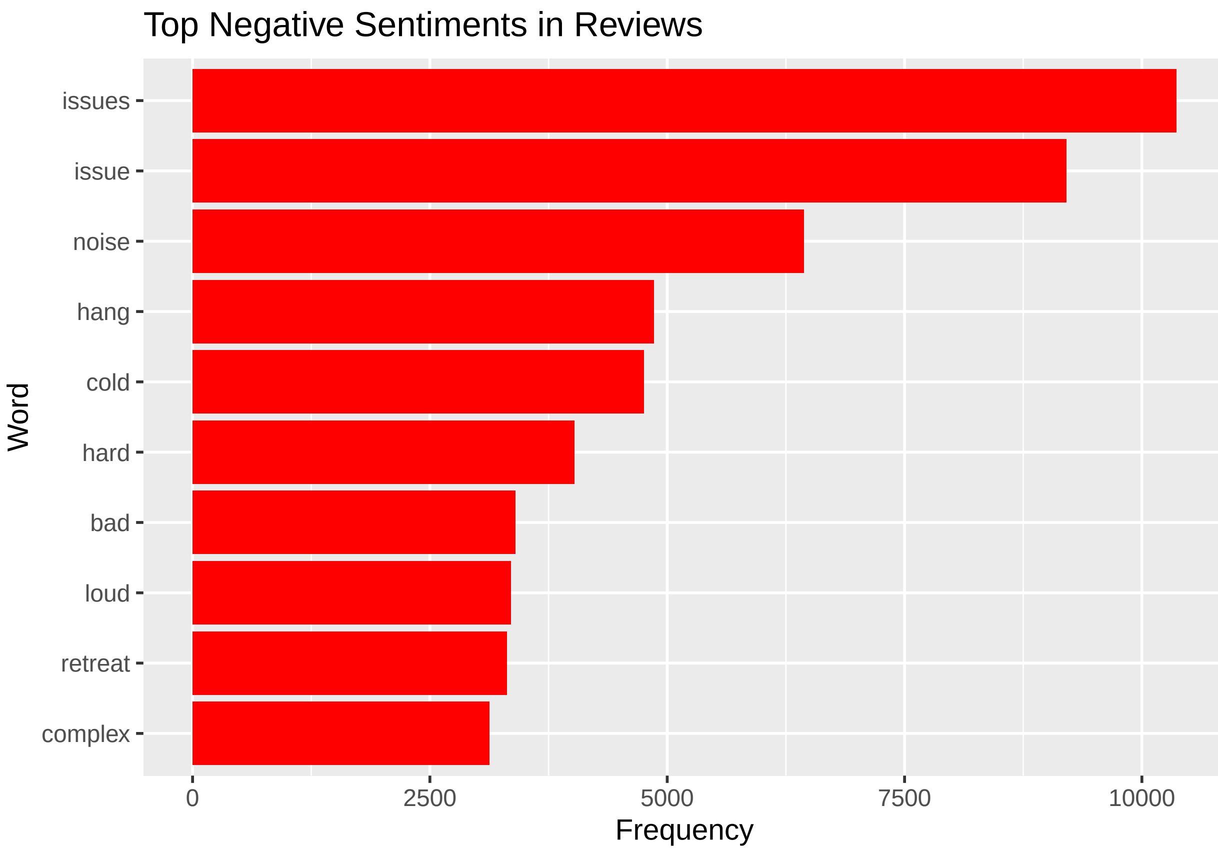
```
## Joining with `by = join_by(word)`

## Warning in inner_join(., get_sentiments("bing")): Detected an unexpected many-to-many relationship be
## i Row 1195880 of `x` matches multiple rows in `y`.
## i Row 2929 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.

## Selecting by n
```

```r
# Plot the top words for positive and negative sentiments
ggplot(top_positive_words, aes(x = reorder(word, n), y = n)) +
  geom_bar(stat = "identity", fill = "green") +
```

```
  coord_flip() +
  labs(title = "Top Positive Sentiments in Reviews", x = "Word", y = "Frequency")
```




```
ggplot(top_negative_words, aes(x = reorder(word, n), y = n)) +
  geom_bar(stat = "identity", fill = "red") +
  coord_flip() +
  labs(title = "Top Negative Sentiments in Reviews", x = "Word", y = "Frequency")
```



We answered this question previously while looking at the top words in this dataset and their sentiment, but here is another visual.

Question 3: How do hosts' practices and neighborhood characteristics impact guests' experiences and reviews?

Identify Most Frequent Words by Listing

```
# Load necessary libraries
library(dplyr)
library(tidytext)
library(stringr)
library(ggplot2)

# Ensure listing_id is a character type
cleaned_comments <- cleaned_comments %>%
  mutate(listing_id = as.character(listing_id))

# Find the most common word for each listing_id
top_word_by_listing <- cleaned_comments %>%
  count(listing_id, word, sort = TRUE) %>%          # Count word occurrences per listing_id
  group_by(listing_id) %>%                          # Group by listing_id
  slice_max(n, n = 1, with_ties = FALSE)            # Select the word with the highest count (no ties)

# Display the results
print(top_word_by_listing)
```

```
## # A tibble: 12,167 x 3
```

```
## # Groups:   listing_id [12,167]
##    listing_id          word       n
##    <chr>               <chr>  <int>
##  1 1.000797929335e+18 airbnb     2
##  2 1.016805032206e+18 house     18
##  3 1.022608702149e+18 nice       2
##  4 1.030750700746e+18 stay       4
##  5 1.039997958937e+18 easy       6
##  6 1.045794183008e+18 austin     6
##  7 1.049520689799e+18 stay      10
##  8 1.050854100196e+18 blake     10
##  9 1.061480881084e+18 space      2
## 10 1.074699991468e+18 stay      23
## # i 12,157 more rows
```

This block identifies the single most common word used in reviews for each listing. It uses slice_max to ensure only the top word (without ties) is selected for each listing_id.

Keyword Mentions in Reviews

```r
# Load necessary libraries
library(dplyr)
library(tidytext)
library(stringr)

# Ensure listing_id is a character type
cleaned_comments <- cleaned_comments %>%
  mutate(listing_id = as.character(listing_id))

# Create a list of words of interest
keywords <- c("clean", "hospitable", "responsive", "friendly", "helpful", "organized", "welcoming", "lo

# Filter the words that match the keywords and count occurrences
keyword_mentions <- cleaned_comments %>%
  filter(word %in% keywords) %>%     # Filter for the keywords: location, Austin, host
  count(word, sort = TRUE)            # Count the occurrences of each word

# Display the results
print(keyword_mentions)
```

```
## # A tibble: 14 x 2
##    word            n
##    <chr>       <int>
##  1 location   200180
##  2 clean      167176
##  3 quiet       63698
##  4 responsive  57333
##  5 helpful     37776
##  6 friendly    32971
##  7 convenient  27402
##  8 safe        21821
##  9 welcoming   13661
## 10 central      8525
## 11 accessible   4787
## 12 organized    3745
```

```
## 13 hospitable   3513
## 14 noisy        1996
```

Here, we observe the frequency of words associated with hospitality. "Location" is mentioned over 200,000 times, followed by "clean," "quiet," "responsive," "helpful," and "friendly." This suggests that these aspects are highly valued by customers in their hosts and the experience they provide.

Calculate Positive/Negative Sentiment

```r
## Categorize into Positive and Negtive Sentiment

library(tidytext)

# Perform sentiment analysis using the 'bing' lexicon
sentiment <- cleaned_comments %>%
  inner_join(get_sentiments("bing")) %>%  # Join with the 'bing' lexicon
  count(listing_id, sentiment, sort = TRUE) %>%
  spread(sentiment, n, fill = 0)  # Spread the sentiment into columns
```

```
## Joining with `by = join_by(word)`
```

```
## Warning in inner_join(., get_sentiments("bing")): Detected an unexpected many-to-many relationship be
## i Row 1195880 of `x` matches multiple rows in `y`.
## i Row 2929 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.
```

```r
# Calculate sentiment score by subtracting negative from positive
sentiment <- sentiment %>%
  mutate(sentiment_score = positive - negative)

# View the sentiment data for each listing
head(sentiment)
```

```
## # A tibble: 6 x 4
##   listing_id         negative positive sentiment_score
##   <chr>                 <dbl>    <dbl>           <dbl>
## 1 1.000797929335e+18        1       11              10
## 2 1.016805032206e+18       25       61              36
## 3 1.022608702149e+18        0        7               7
## 4 1.030750700746e+18        0       19              19
## 5 1.039997958937e+18        4       56              52
## 6 1.045794183008e+18        1       28              27
```

This section calculates sentiment scores for each listing by categorizing words as positive or negative using the Bing lexicon. It subtracts the count of negative words from positive ones to create a sentiment score.

Sentiment Score Summary

```r
# Calculate the lowest, highest, and average sentiment score
sentiment_stats <- sentiment %>%
  summarise(
    min_sentiment = min(sentiment_score, na.rm = TRUE),
    max_sentiment = max(sentiment_score, na.rm = TRUE),
    avg_sentiment = mean(sentiment_score, na.rm = TRUE)
  )
```

```r
# View the calculated statistics
sentiment_stats
```

```
## # A tibble: 1 x 3
##   min_sentiment max_sentiment avg_sentiment
##           <dbl>         <dbl>         <dbl>
## 1           -27          4947          162.
```

We find that the minimum sentiment score is -27, the maximum sentiment score is 4949, and the average sentiment score is 162. This suggests that, overall, comments are positive, though there is considerable variation among listings.

Overall Sentiment Breakdown

```r
library(tidytext)
library(ggplot2)

sentiments <- cleaned_comments %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

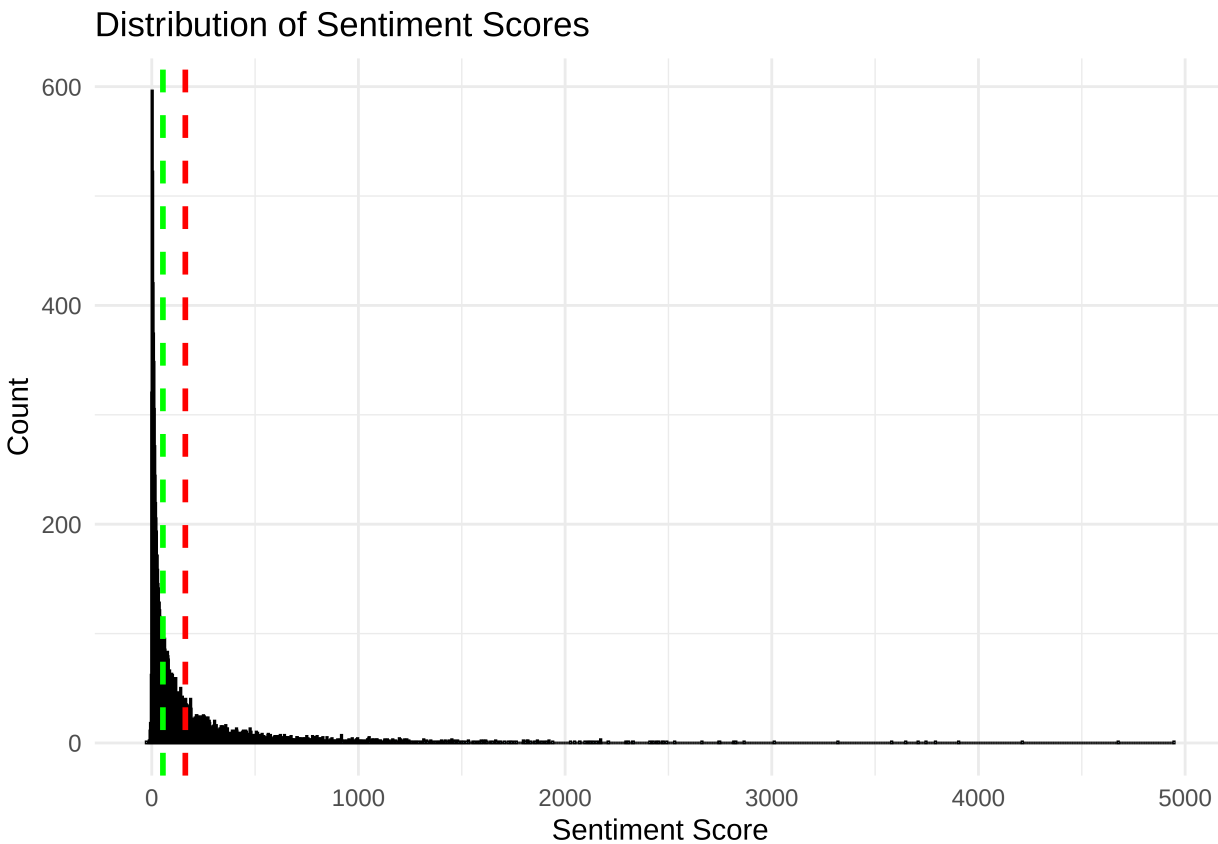
```
## Joining with `by = join_by(word)`

## Warning in inner_join(., get_sentiments("bing")): Detected an unexpected many-to-many relationship b
## i Row 1195880 of `x` matches multiple rows in `y`.
## i Row 2929 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.
```

```r
ggplot(sentiments, aes(x = sentiment, fill = sentiment)) +
  geom_bar() +
  labs(title = "Distribution of Sentiments in Reviews", x = "Sentiment", y = "Count") +
  theme_minimal()
```

We see that there are more negative reviews than positive. Previously we saw an increase in positive reviews in recent years, so this may be due to older ones.

Plot Sentiment Score Distribution

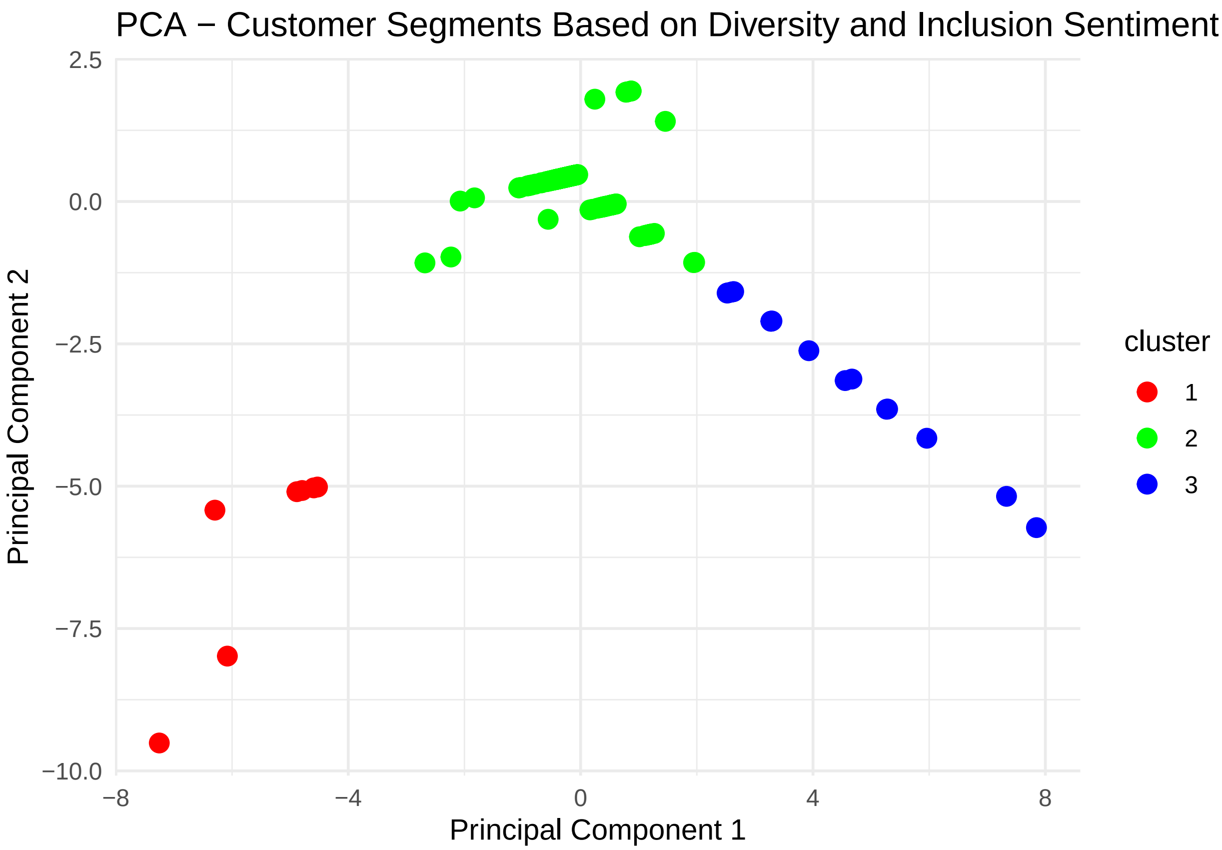
```r
# Load necessary libraries for plotting
library(ggplot2)

# Adjust binwidth to reduce the number of bars
ggplot(sentiment, aes(x = sentiment_score)) +
  geom_histogram(binwidth = 2, fill = "skyblue", color = "black", alpha = 0.7) +
  labs(title = "Distribution of Sentiment Scores",
       x = "Sentiment Score",
       y = "Count") +
  theme_minimal() +
  geom_vline(aes(xintercept = mean(sentiment_score, na.rm = TRUE)),
             color = "red", linetype = "dashed", size = 1) +
  geom_vline(aes(xintercept = median(sentiment_score, na.rm = TRUE)),
             color = "green", linetype = "dashed", size = 1)
```



The distribution is highly skewed to the right.

Perform K Means Clustering to Segment Customers

```
# Apply PCA (Principal Component Analysis) to reduce the data to 2 components for visualization
pca_result <- prcomp(scaled_data_di, center = TRUE, scale. = TRUE)

# Extract the first two principal components
pca_data <- as.data.frame(pca_result$x[, 1:2]) # First two principal components
pca_data$cluster <- factor(customer_profiles_di_clean$cluster) # Add cluster labels

# Plot the clusters on a 2D scatter plot
library(ggplot2)

ggplot(pca_data, aes(x = PC1, y = PC2, color = cluster)) +
  geom_point(size = 3) +
  scale_color_manual(values = c("red", "green", "blue")) + # Adjust colors for clusters
  labs(
    title = "PCA - Customer Segments Based on Diversity and Inclusion Sentiment",
    x = "Principal Component 1",
    y = "Principal Component 2"
  ) +
  theme_minimal()
```



Cluster 1: This cluster represents customers with negative sentiment on average, and they tend to write longer reviews. However, these customers have a very low number of positive reviews. Cluster 2: This cluster has positive sentiment on average, and customers in this group tend to write moderately long reviews. They have a modest number of positive reviews. Cluster 3: This cluster has a slightly positive sentiment overall, with the shortest reviews on average. However, these customers leave a high number of positive reviews compared to others.