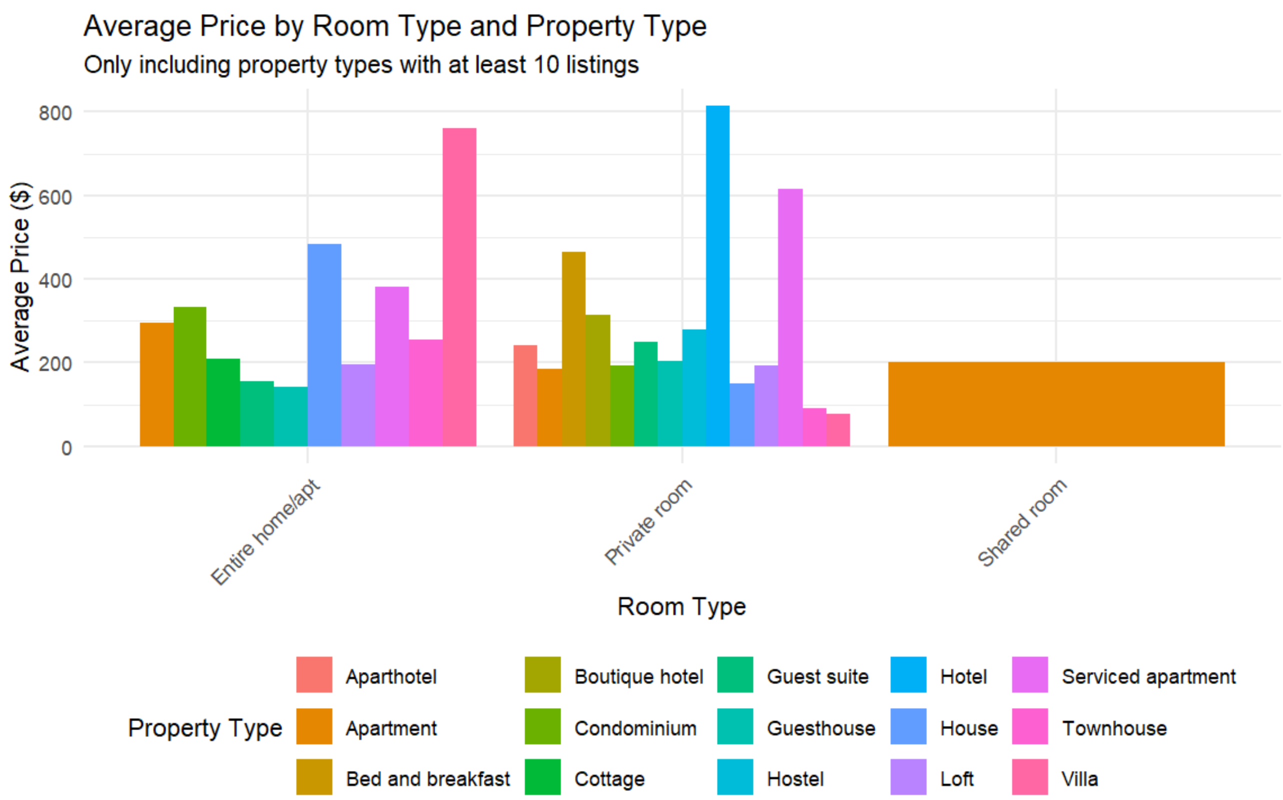
Appendix

Appendix 1: Description of the "Average Price by Room Type and Property Type" visualization



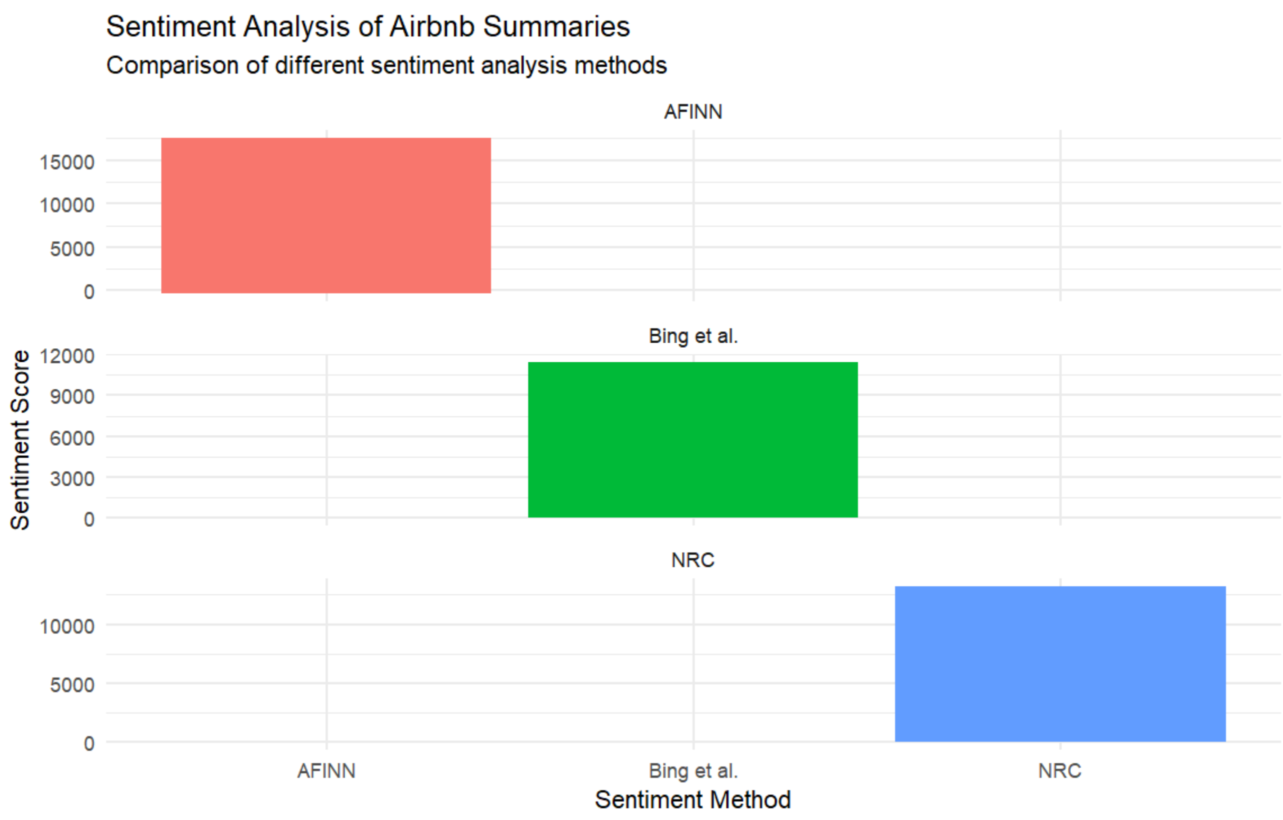
Axes:

- **X-axis**: Represents Room Type. The three room types are "Entire home/apt," "Private room," and "Shared room."
- **Y-axis**: Represents Average Price ($). The scale ranges from 0 to 800.

Data Representation:

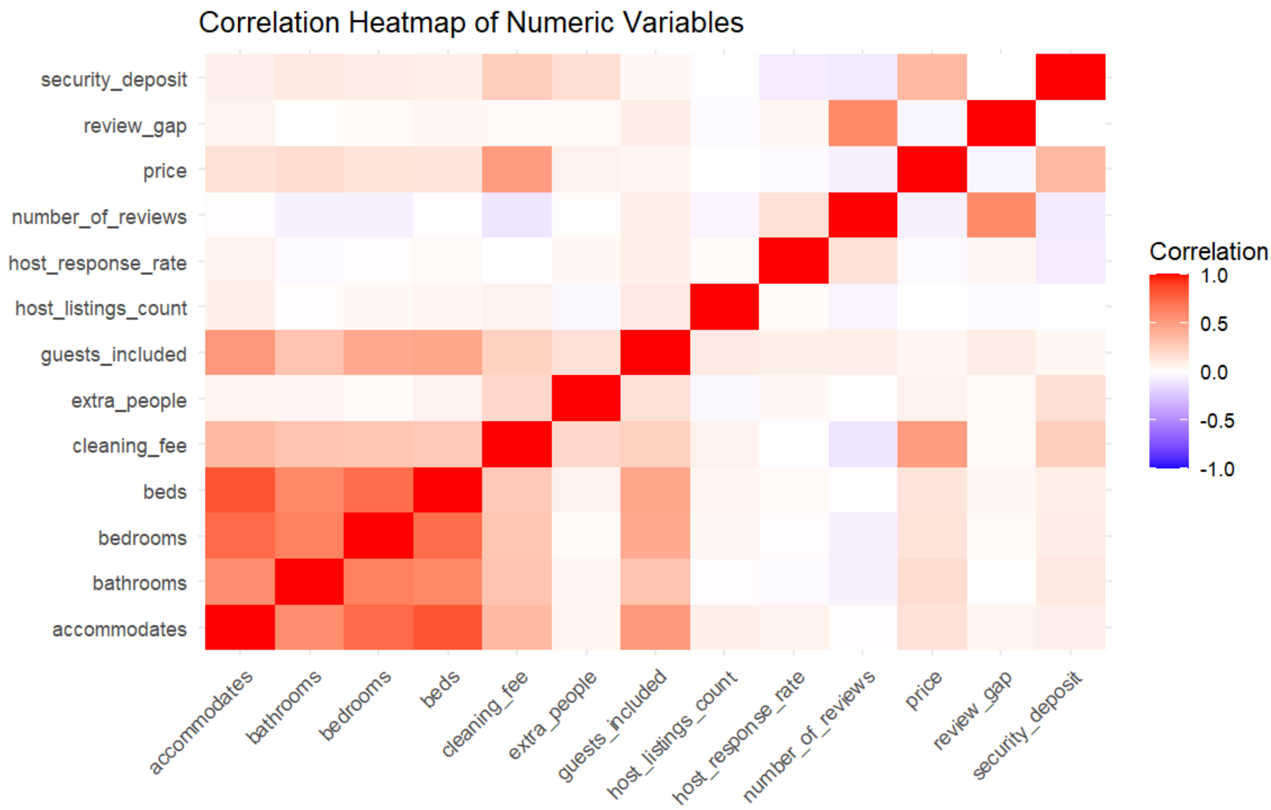
- Each bar represents a specific combination of room type and property type.
- The height of each bar corresponds to the average price for that combination.
- Bars are color-coded to distinguish between different property types, with a legend provided beneath the chart.

Appendix 2: Description of the sentiment analysis visualization for the Airbnb dataset:



The y-axis represents the **Sentiment Score**, and the x-axis represents the **Sentiment Method**.

Appendix 3: Description of the "Correlation Heatmap of Numeric Variables" visualization for the Airbnb dataset:



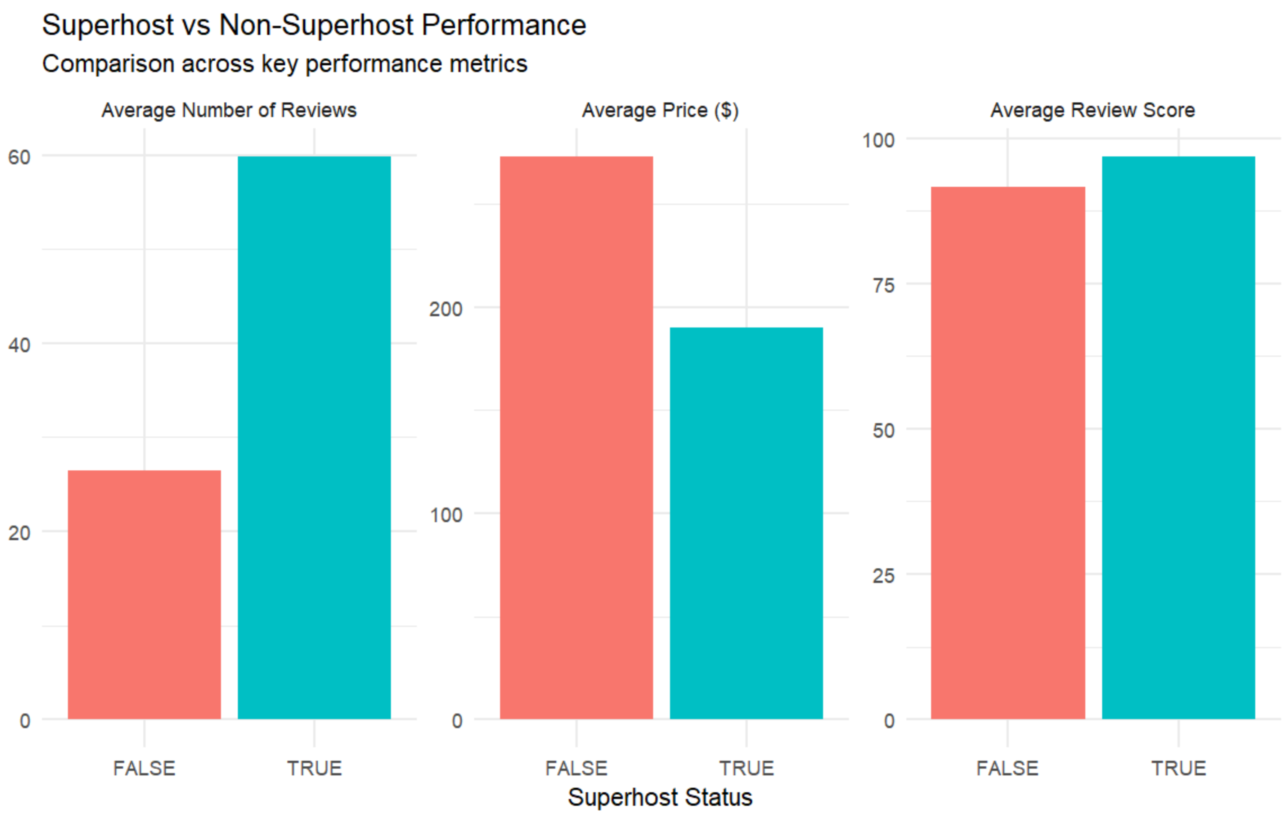
Cells: Each cell at the intersection of a row and a column represents the correlation coefficient between the two corresponding variables.

Color Coding: The cells are colored according to the strength and direction of the correlation:

- **Red** indicates a positive correlation. The intensity of the red color represents the strength of the positive correlation (darker red = stronger positive correlation).
- **White**: Indicates a correlation close to zero (no or very weak correlation).
- **Blue** Indicates a negative correlation. The intensity of the blue color represents the strength of the negative correlation (darker blue = stronger negative correlation).

Color Scale: A color scale (legend) is provided to the right of the heatmap, mapping the color intensities to correlation values ranging from -1.0 to 1.0.

Appendix 4: "Superhost vs Non-Superhost Performance: Comparison across key performance metrics", comparing the performance of Airbnb listings based on whether the host is a Superhost (TRUE) or not (FALSE)



X-axis: Superhost Status (FALSE/TRUE).

Average Number of Reviews: **Y-axis**: *Average Number of Reviews.* The turquoise bar is significantly higher than the coral bar, indicating that listings managed by Superhosts tend to have a much higher average number of reviews than those managed by non-Superhosts.

Average Price: **Y-axis**: *Average Price.* On average, the listings managed by non-Superhosts have a higher price than those managed by Superhosts, possibly to break even owing to fewer customers.

Average Review Score: **Y-axis**: *Average Review Score.* The bar graph suggests that listings managed by Superhosts generally have higher average review scores.

Appendix 5: **Word Cloud** titled “Most Common Words in Airbnb Descriptions.”

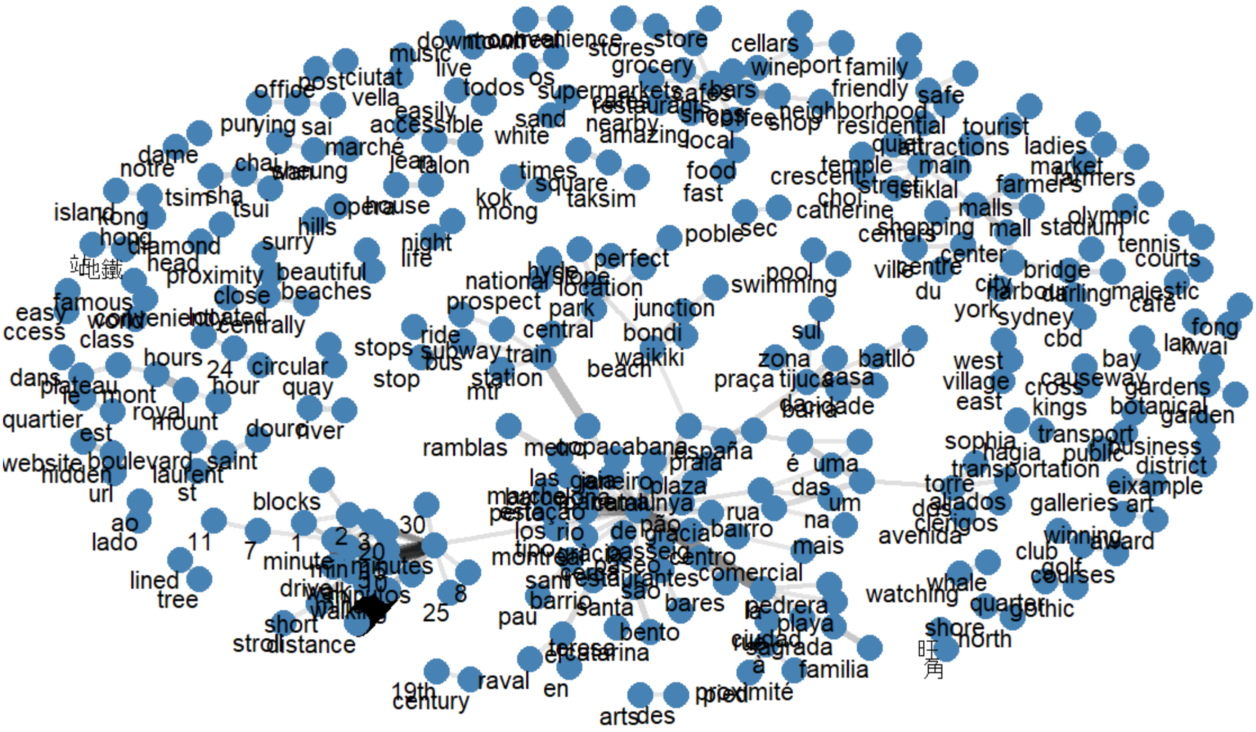
Dominant Words: Due to its size, the most prominent word is “apartment.” Other significant words include “minutes,” “located,” “restaurants,” “kitchen,” and “bedroom.” These suggest that descriptions often focus on property type, location, amenities, and the number of bedrooms.

Descriptive Adjectives: Adjectives like “beautiful,” “cozy,” “comfortable,” “modern,” and “quiet” give insight into how hosts are describing their properties to attract customers.

Appendix 6: **Bigram Network of Neighborhood Descriptions**

Bigram Network of Neighborhood Descriptions

Showing bigrams appearing at least 15 times

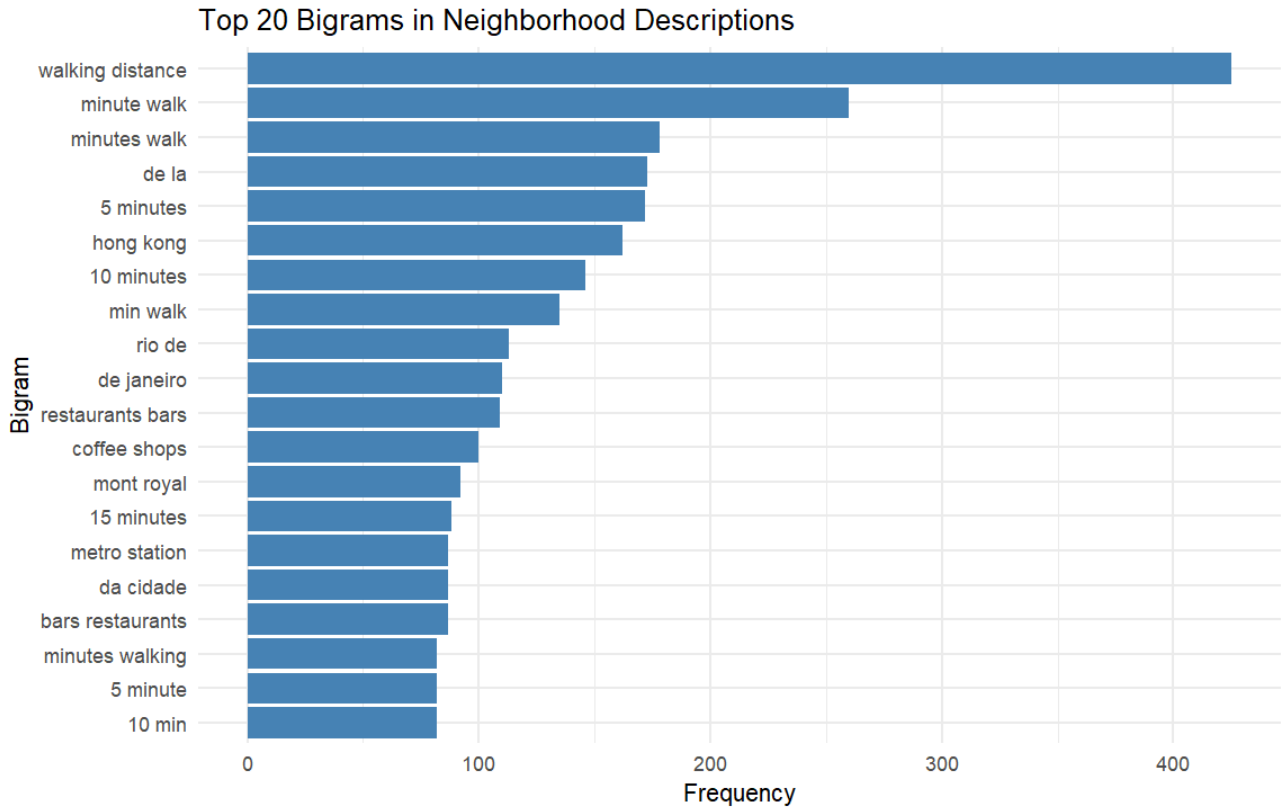


Nodes: Each word or phrase (bigram) is represented as a node (a blue circle). The label of each node is the bigram itself.

Edges: Lines connect the nodes, representing the co-occurrence of these words or phrases within the neighborhood descriptions. The thickness of the line suggests the frequency of their co-occurrence, with thicker lines indicating more frequent pairings.

Layout: The nodes are arranged in a force-directed layout where connected nodes are drawn closer together and disconnected nodes are pushed further apart. This allows clusters of related terms to emerge.

Appendix 7: Horizontal bar chart titled "Top 20 Bigrams in Neighborhood Descriptions".



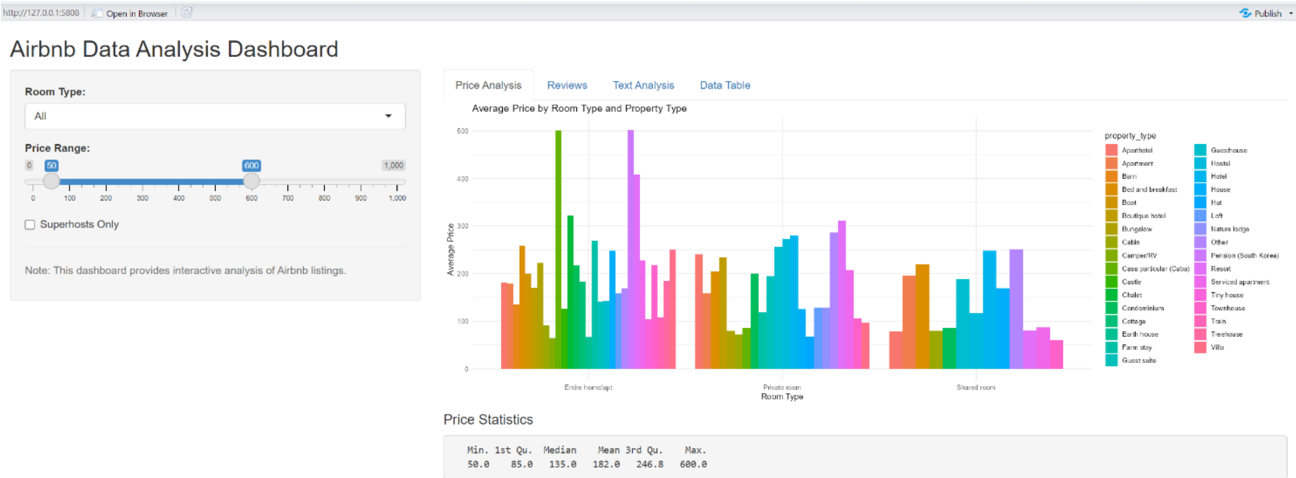
Y-axis: The y-axis lists the 20 most frequent bigrams. Each bigram is a pair of words often appearing together in the neighborhood descriptions. Examples include "walking distance," "minute walk," "restaurants bars," and location-specific phrases like "hong kong," "rio de," and "de janeiro." The bigrams are ordered from top to bottom according to frequency (highest to lowest).

X-axis: The x-axis represents the frequency or count of each bigram in the dataset. The scale extends from 0 to 400.

Bars: Each bigram has a horizontal bar extending from the y-axis to a point corresponding to its frequency on the x-axis. The bars are all colored in blue.

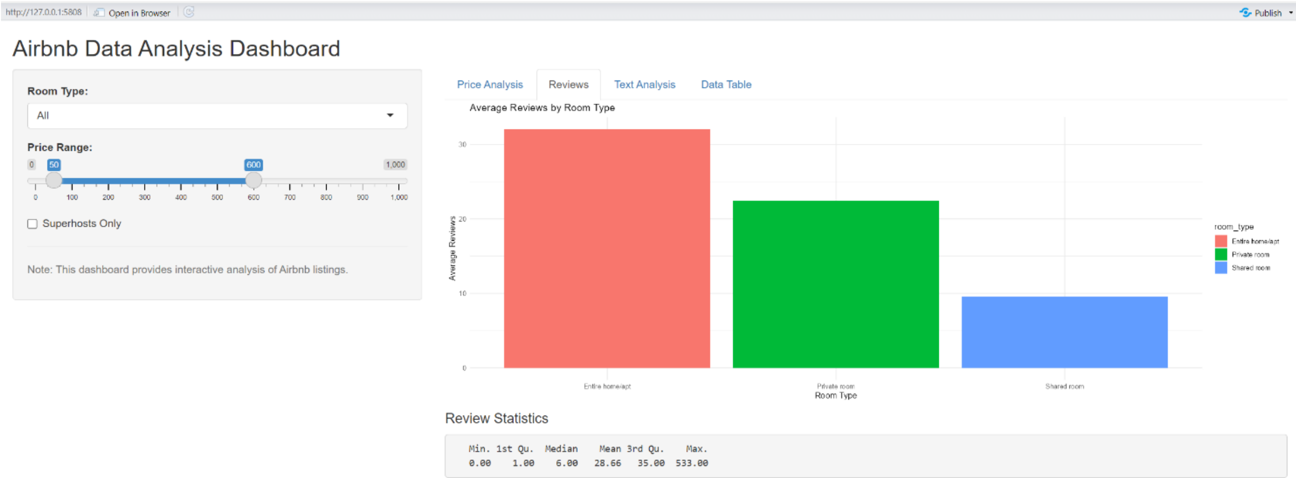
Appendix 8.A: Description of the Airbnb Data Analysis Dashboard

Price Analysis Tab:



This tab features a bar chart comparing average prices by room and property types. The chart allows users to identify which combinations command the highest prices quickly. Below the chart are price statistics, such as the minimum, first quartile, median, mean, third quartile, and maximum listing prices.

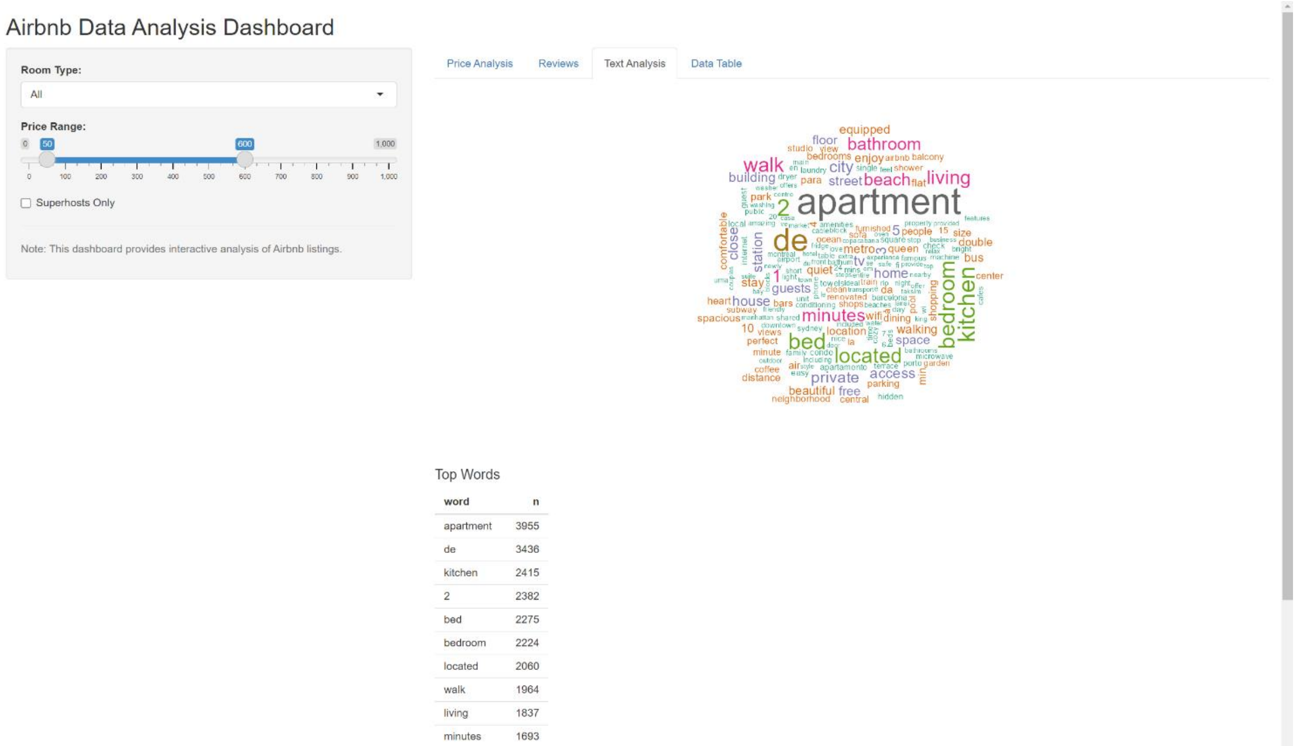
Reviews Tab:



The section displays a bar chart showing the average number of reviews across all room types. The Review statistics present descriptive statistics about the reviews.

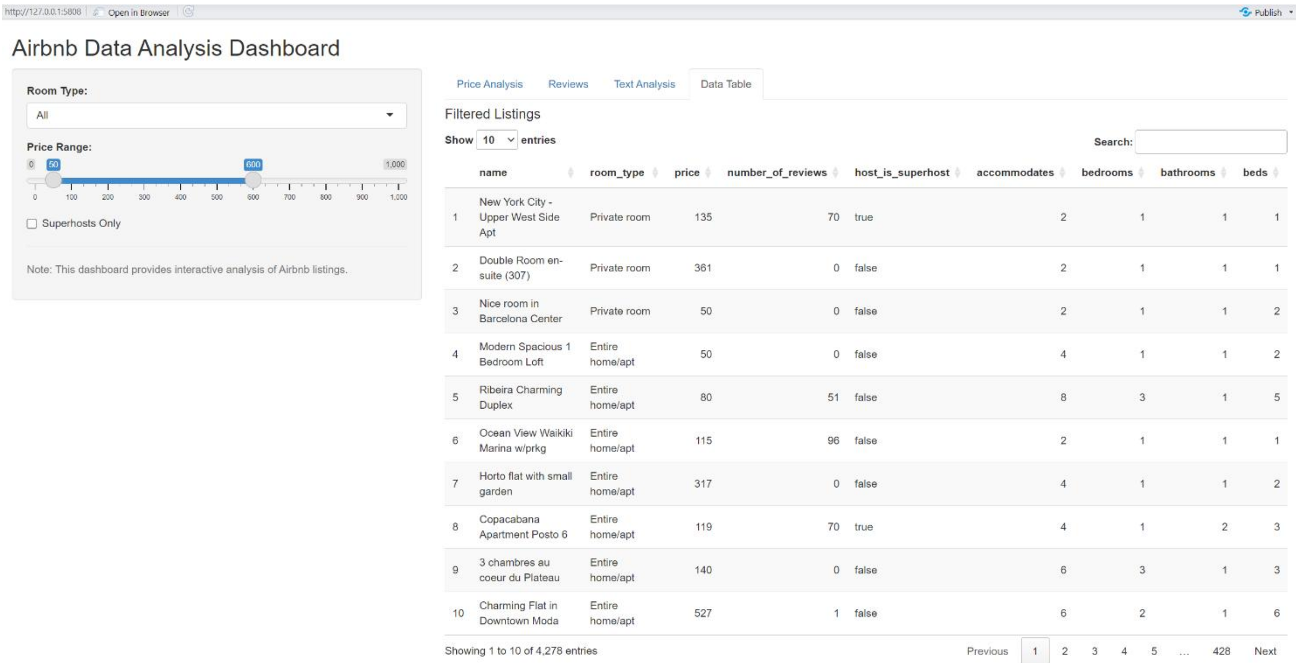
Appendix 8.B: Description of the Airbnb Data Analysis Dashboard

Text Analysis Tab:



The central visualization in this tab is a word cloud. The size of each word in the cloud corresponds to its frequency in the Airbnb descriptions, providing a visual representation of the most common themes and features highlighted. The **Top Words Table** shows a list of top-occurring words in the listings, ranked by how often they occur.

Data Table Tab:



Displays raw data in a tabular form, allowing users to browse individual Airbnb listings. The table includes key listing attributes like name, room type, price, number of reviews, and host status.

Controls for the data table include filtering and search capabilities, as well as pagination to navigate through multiple pages of listings.

Appendix 9: R-Script

Appendix 9.A: Setup and Configuration

```
# ==============================================================================
# Assignment A1 : AIRBNB DATA ANALYSIS PIPELINE
# ==============================================================================
# Author: Tirthankar Mukherjee
# University: Hult International Business School
# Course: Business Analysis with Unstructured Data - DAT-7471 - FMBANDD1
# Program: Master of Business Analytics Dual Degree
# Professor: Thomas Kurnicki
# Date: March 13, 2025
# Description: Comprehensive analysis of Airbnb listing data to identify key
#              factors influencing listing performance and revenue
# ==============================================================================

#---------------------------------------------------------------------------
# ----------------- SETUP AND CONFIGURATION ------------------------------
#---------------------------------------------------------------------------
# Install required packages if not already installed
required_packages <- c("mongolite", "tidyverse", "tidytext", "wordcloud",
                       "ggplot2", "caret", "shiny", "lubridate", "dplyr",
                       "tidyr", "igraph", "ggraph", "RColorBrewer")

# Check and install missing packages
new_packages <- required_packages[!(required_packages %in% installed.packages()[,"Package"])]
if(length(new_packages) > 0) {
  install.packages(new_packages)
}

# Load all required libraries
lapply(required_packages, library, character.only = TRUE)

# Set default theme for consistent visualization
theme_set(theme_minimal())

```

In this code snippet, the required packages are installed and loaded, which are necessary for the analysis.

Appendix 9.B: Data Connection and Loading

```
#---------------------------------------------------------------------------
# ----------------- DATA CONNECTION AND LOADING ---------------------------
#---------------------------------------------------------------------------
# MongoDB connection details
# Note: In production, consider storing credentials in environment variables
connection_string <- 'mongodb+srv://mukherjeetirthankar:tirthA25@a1textmining.9gdz0.mongodb.net/?retryWr

# Initialize MongoDB connection
airbnb_collection <- mongo(
  collection = "listingsAndReviews",
  db = "sample_airbnb",
  url = connection_string
)

# Function to clean and preprocess Airbnb data
clean_airbnb_data <- function(data) {
  # Data cleaning and type conversion
  data %>%
    mutate(
      # Convert currency string to numeric
      price = as.numeric(gsub("[\\$,]", "", price)),

      # Convert date strings to Date objects
      last_review = as.Date(last_review),
      first_review = as.Date(first_review),

      # Calculate duration between first and last review in days
      review_gap = as.numeric(difftime(last_review, first_review, units = "days")),

      # Extract and clean host response rate
      host_response_rate = as.numeric(gsub("%", "", host$host_response_rate)) / 100,

      # Convert listing details to numeric
      accommodates = as.numeric(accommodates),
      bedrooms = as.numeric(bedrooms),
      bathrooms = as.numeric(bathrooms),
      beds = as.numeric(beds),

      # Extract additional host information
      host_is_superhost = host$host_is_superhost,
      host_identity_verified = host$host_identity_verified,
      host_listings_count = as.numeric(host$host_listings_count)
    )
}

# Fetch and preprocess all data
# Warning: This operation may require significant memory
airbnb <- airbnb_collection$find() %>% clean_airbnb_data()
```

In this code snippet, a MongoDB connection is initiated to load the dataset into R.

Appendix 9.C: Exploratory Data Analysis

```r
#---------------------------------------------------------------------------
#------------------ EXPLORATORY DATA ANALYSIS ---------------------------
#---------------------------------------------------------------------------

# Function to summarize dataset
summarize_dataset <- function(data) {
  # Basic dataset summary
  cat("Dataset Summary:\n")
  cat("Number of listings:", nrow(data), "\n")
  cat("Number of variables:", ncol(data), "\n")
  cat("Date range:", min(data$first_review, na.rm = TRUE), "to",
      max(data$last_review, na.rm = TRUE), "\n")

  # Missing values summary
  missing_values <- data %>%
    summarise(across(everything(), ~sum(is.na(.)))) %>%
    pivot_longer(everything(), names_to = "variable", values_to = "missing_count") %>%
    filter(missing_count > 0) %>%
    arrange(desc(missing_count))

  if(nrow(missing_values) > 0) {
    cat("\nVariables with missing values:\n")
    print(missing_values)
  }
}

# Run initial data summary
summarize_dataset(airbnb)
```

In this code snippet, an exploratory data analysis of the Airbnb dataset is conducted.

Appendix 9.D: Price Analysis

```r
#----------------------------------------------------------------------
# -------------------- PRICE ANALYSIS ---------------------------------
#----------------------------------------------------------------------
# Analyze how price varies by room type and property type
price_analysis <- airbnb %>%
  # Group data for aggregation
  group_by(room_type, property_type) %>%
  # Calculate metrics for each group
  summarise(
    avg_price = mean(price, na.rm = TRUE),
    median_price = median(price, na.rm = TRUE),
    min_price = min(price, na.rm = TRUE),
    max_price = max(price, na.rm = TRUE),
    total_listings = n(),
    avg_reviews = mean(number_of_reviews, na.rm = TRUE),
    .groups = "drop"  # Drop grouping after summarization
  ) %>%
  # Filter out uncommon property types for clearer visualization
  filter(total_listings >= 10)

# Visualization: Average Price by Room Type and Property Type
plot_price_by_room_property <- function(data) {
  ggplot(data, aes(x = room_type, y = avg_price, fill = property_type)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(
      title = "Average Price by Room Type and Property Type",
      subtitle = "Only including property types with at least 10 listings",
      x = "Room Type",
      y = "Average Price ($)",
      fill = "Property Type"
    ) +
    theme(
      axis.text.x = element_text(angle = 45, hjust = 1),
      legend.position = "bottom",
      legend.box = "horizontal"
    )
}

# Generate and display the price analysis plot
plot_price_by_room_property(price_analysis)
```

Now, the code snippet shows the price analysis and visualizes it with the ggplot function.

Appendix 9.E: Host Performance Analysis

```r
#---------------------------------------------------------------------------
#---------------- HOST PERFORMANCE ANALYSIS ------------------------------
#---------------------------------------------------------------------------
# Analyze how host characteristics affect listing performance
host_performance <- airbnb %>%
  # Filter out entries with missing response rate
  filter(!is.na(host$host_response_rate)) %>%
  # Group by superhost status
  group_by(host_is_superhost) %>%
  # Calculate metrics for each group
  summarise(
    avg_response_rate = mean(host_response_rate, na.rm = TRUE),
    avg_price = mean(price, na.rm = TRUE),
    avg_reviews = mean(number_of_reviews, na.rm = TRUE),
    avg_review_scores = mean(review_scores$review_scores_rating, na.rm = TRUE),
    total_listings = n(),
    .groups = "drop"
  )

# Visualization: Superhost vs Non-Superhost Performance
plot_superhost_performance <- function(data) {
  # Prepare data for visualization by pivoting to long format
  plot_data <- data %>%
    select(host_is_superhost, avg_price, avg_reviews, avg_review_scores) %>%
    pivot_longer(
      cols = c(avg_price, avg_reviews, avg_review_scores),
      names_to = "metric",
      values_to = "value"
    ) %>%
    mutate(
      # Create readable labels for the metrics
      metric = case_when(
        metric == "avg_price" ~ "Average Price ($)",
        metric == "avg_reviews" ~ "Average Number of Reviews",
        metric == "avg_review_scores" ~ "Average Review Score",
        TRUE ~ metric
      )
    )

  # Create the plot
  ggplot(plot_data, aes(x = host_is_superhost, y = value, fill = host_is_superhost)) +
    geom_bar(stat = "identity") +
    facet_wrap(~ metric, scales = "free_y") +
    labs(
      title = "Superhost vs Non-Superhost Performance",
      subtitle = "Comparison across key performance metrics",
      x = "Superhost Status",
      y = NULL
    ) +
    theme(legend.position = "none")
}

# Generate and display the host performance plot
plot_superhost_performance(host_performance)
```

The code snippet compares the superhost with the non-superhost across various metrics like Price, Reviews, and so on. Then, it visualizes it with the ggplot function.

Appendix 9.F: Text Mining on Descriptions

```r
#---------------------------------------------------------------------------
#---------------- TEXT MINING ON DESCRIPTIONS ----------------------------
#---------------------------------------------------------------------------
# Process and analyze listing descriptions to identify key terms

# Function to process text and remove stop words
process_text <- function(data, text_column, id_column = "name") {
  data %>%
    select(!!sym(id_column), !!sym(text_column)) %>%
    # Filter out missing text values
    filter(!is.na(!!sym(text_column))) %>%
    # Tokenize text into individual words
    unnest_tokens(word, !!sym(text_column)) %>%
    # Remove stop words that don't add meaning
    anti_join(stop_words) %>%
    # Count word frequency
    count(word, sort = TRUE)
}

# Process descriptions for analysis
description_tokens <- process_text(airbnb, "description")

# Create word cloud of most common words in descriptions
create_wordcloud <- function(token_data, max_words = 200, title = "Word Cloud") {
  # Set up plotting device
  par(mar = c(0, 0, 2, 0))

  # Generate word cloud
  wordcloud(
    words = token_data$word,
    freq = token_data$n,
    max.words = max_words,
    colors = brewer.pal(8, "Dark2"),
    random.order = FALSE,
    rot.per = 0.35,
    scale = c(3, 0.5)
  )

  # Add title
  title(main = title)
}

# Generate and display the description word cloud
create_wordcloud(
  description_tokens,
  max_words = 200,
  title = "Most Common Words in Airbnb Descriptions"
)
```

In this code snippet, descriptions of Airbnb properties are processed and then visualized through a word cloud.

Appendix 9.G: Sentiment Analysis

```r
#---------------- SENTIMENT ANALYSIS ON LISTING SUMMARIES ----------------
#-------------------------------------------------------------------------
# Analyze the sentiment of listing summaries to identify emotional tone

# Function to perform sentiment analysis using multiple lexicons
analyze_sentiment <- function(token_data) {
  # AFINN sentiment analysis (numerical scores)
  afinn_sentiment <- token_data %>%
    inner_join(get_sentiments("afinn")) %>%
    group_by(name) %>%
    summarise(sentiment = sum(value), .groups = "drop") %>%
    mutate(method = "AFINN")

  # Bing and NRC sentiment analysis (categorical)
  bing_and_nrc_sentiment <- bind_rows(
    # Bing lexicon (positive/negative)
    token_data %>%
      inner_join(get_sentiments("bing")) %>%
      mutate(method = "Bing et al."),

    # NRC lexicon (positive/negative only)
    token_data %>%
      inner_join(get_sentiments("nrc") %>%
                   filter(sentiment %in% c("positive", "negative"))) %>%
      mutate(method = "NRC")
  ) %>%
    # Count positive and negative words
    count(method, sentiment) %>%
    # Reshape data for visualization
    pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
    # Calculate net sentiment
    mutate(sentiment = positive - negative)

  # Combine results from different methods
  bind_rows(afinn_sentiment, bing_and_nrc_sentiment)
}

# Process and tokenize summaries
summary_tokens <- airbnb %>%
  select(name, summary) %>%
  filter(!is.na(summary)) %>%
  unnest_tokens(word, summary) %>%
  filter(!is.na(word)) %>%
  anti_join(stop_words)

# Perform sentiment analysis
combined_sentiments <- analyze_sentiment(summary_tokens)

# Visualization: Sentiment Analysis Results
plot_sentiment_analysis <- function(sentiment_data) {
  ggplot(sentiment_data, aes(method, sentiment, fill = method)) +
    geom_col(show.legend = FALSE) +
    facet_wrap(~method, ncol = 1, scales = "free_y") +
    labs(
```

A sentiment analysis is performed on the tokenized description column in the code snippet.

Appendix 9.H: Neighborhood Bigram Analysis

```r
#---------------------- NEIGHBORHOOD BIGRAM ANALYSIS ---------------------------
#--------------------------------------------------------------------------------
# Analyze neighborhood descriptions using bigrams to identify location features

# Function to process text into n-grams
process_ngrams <- function(data, text_column, n = 2, id_column = "name", min_count = 1) {
  # Create ngrams from text
  ngrams <- data %>%
    filter(!is.na(!!sym(text_column))) %>%
    unnest_tokens(
      ngram,
      !!sym(text_column),
      token = "ngrams",
      n = n
    )

  # For bigrams, separate into component words and filter stop words
  if (n == 2) {
    ngrams <- ngrams %>%
      separate(ngram, c("word1", "word2"), sep = " ") %>%
      filter(!word1 %in% stop_words$word) %>%
      filter(!word2 %in% stop_words$word) %>%
      filter(!is.na(word1) & !is.na(word2)) %>%
      count(word1, word2, sort = TRUE) %>%
      filter(n >= min_count)
  } else {
    # For other n-grams, just count occurrences
    ngrams <- ngrams %>%
      count(ngram, sort = TRUE) %>%
      filter(n >= min_count)
  }

  return(ngrams)
}

# Process neighborhood descriptions into bigrams
neighborhood_bigrams <- process_ngrams(
  airbnb,
  "neighborhood_overview",
  n = 2,
  min_count = 10
)

# Visualization: Top Bigrams Bar Chart
plot_top_bigrams <- function(bigram_data, top_n = 20) {
  bigram_data %>%
    head(top_n) %>%
    unite(bigram, word1, word2, sep = " ") %>%
    ggplot(aes(x = reorder(bigram, n), y = n)) +
    geom_col(fill = "steelblue") +
    coord_flip() +
    labs(
      title = paste("Top", top_n, "Bigrams in Neighborhood Descriptions"),
      x = "Bigram",
```

This code snippet shows the bigram analysis and visualizes the top 20 bigrams with a Bar Chart.

```r
# Visualization: Bigram Network Graph
plot_bigram_network <- function(bigram_data, min_count = 10) {
  # Filter bigrams by minimum count
  filtered_bigrams <- bigram_data %>%
    filter(n >= min_count)

  # Create network graph
  bigram_graph <- filtered_bigrams %>%
    graph_from_data_frame()

  # Plot the graph
  ggraph(bigram_graph, layout = "fr") +
    geom_edge_link(aes(edge_alpha = n, edge_width = n), show.legend = FALSE) +
    geom_node_point(color = "steelblue", size = 5) +
    geom_node_text(aes(label = name), vjust = 1.5, hjust = 1) +
    labs(
      title = "Bigram Network of Neighborhood Descriptions",
      subtitle = paste("Showing bigrams appearing at least", min_count, "times")
    ) +
    theme_void()
}

# Generate and display the bigram network
plot_bigram_network(neighborhood_bigrams, min_count = 15)
```

This follow-up code on Bigram Analysis shows the visualization of a bigram network with the help of the bigram data created with the help of the ggraph function.

Appendix 9.I: TF-IDF Analysis

```r
#--------------------------------------------------------------------------
#------------------------ TF-IDF ANALYSIS ----------------------------------
#--------------------------------------------------------------------------

# Identify distinctive terms in neighborhood descriptions using TF-IDF

# Function to perform TF-IDF analysis on bigrams
analyze_tfidf <- function(data, text_column, grouping_var = "neighborhood_cleansed") {
  # Process text into bigrams
  bigrams <- data %>%
    filter(!is.na(!!sym(text_column)) & !is.na(!!sym(grouping_var))) %>%
    unnest_tokens(
      bigram,
      !!sym(text_column),
      token = "ngrams",
      n = 2
    ) %>%
    separate(bigram, c("word1", "word2"), sep = " ") %>%
    filter(!word1 %in% stop_words$word) %>%
    filter(!word2 %in% stop_words$word) %>%
    filter(!is.na(word1) & !is.na(word2)) %>%
    unite(bigram, word1, word2, sep = " ")

  # Calculate TF-IDF
  bigrams %>%
    count(!!sym(grouping_var), bigram) %>%
    bind_tf_idf(bigram, !!sym(grouping_var), n) %>%
    arrange(desc(tf_idf))
}

# Analyze TF-IDF if neighborhood_cleansed is available
if ("neighborhood_cleansed" %in% names(airbnb)) {
  neighborhood_tfidf <- analyze_tfidf(
    airbnb,
    "neighborhood_overview",
    "neighborhood_cleansed"
  )

  # Display top TF-IDF terms by neighborhood
  top_tfidf_terms <- neighborhood_tfidf %>%
    group_by(neighborhood_cleansed) %>%
    slice_max(order_by = tf_idf, n = 5) %>%
    ungroup()

  print(top_tfidf_terms)
}
```

This code snippet shows the UDF of the TF-IDF analysis to identify distinctive terms in neighborhood descriptions.

Appendix 9.J: Correlation Analysis

```r
#--------------------------------------------------------------------------
#---------------------- CORRELATION ANALYSIS -------------------------------
#--------------------------------------------------------------------------
# Analyze correlations between numeric variables

# Function to calculate correlations between numeric variables
calculate_correlations <- function(data) {
  # Select numeric columns
  numeric_cols <- data %>%
    select(where(is.numeric)) %>%
    # Remove columns with too many NAs
    select_if(function(x) mean(!is.na(x)) > 0.5)

  # Calculate correlation matrix
  cor_matrix <- cor(numeric_cols, use = "pairwise.complete.obs")

  # Convert to long format for visualization
  cor_data <- as.data.frame(cor_matrix) %>%
    rownames_to_column("variable1") %>%
    pivot_longer(-variable1, names_to = "variable2", values_to = "correlation")

  return(cor_data)
}

# Calculate correlations
correlations <- calculate_correlations(airbnb)

# Visualization: Correlation Heatmap
plot_correlation_heatmap <- function(cor_data) {
  ggplot(cor_data, aes(x = variable1, y = variable2, fill = correlation)) +
    geom_tile() +
    scale_fill_gradient2(
      low = "blue",
      mid = "white",
      high = "red",
      midpoint = 0,
      limits = c(-1, 1)
    ) +
    theme(
      axis.text.x = element_text(angle = 45, hjust = 1),
      axis.title = element_blank()
    ) +
    labs(
      title = "Correlation Heatmap of Numeric Variables",
      fill = "Correlation"
    )
}

# Generate and display correlation heatmap
plot_correlation_heatmap(correlations)
```

This code snippet shows the correlation analysis between all the numeric variables and visualizes the relationship between these variables with a heat map.