Analysis on Airbnb Dataset

Group2

11/13/2024

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```r
#Install Package
install.packages("readxl")
install.packages("tidyverse")
install.packages("leaflet")
install.packages("corrplot")

#Load libraries
library(readxl)
library(tidyverse)  # This includes dplyr, ggplot2, and tidyr
library(tidyr)      # Explicitly load tidyr for pivot_longer
library(leaflet)    # Load leaflet for mapping
library(ggcorrplot)   # Load corrplot for correlation visualization
```

Load data into datframe

```r
# Read the Excel file
airbnb <- readxl::read_xlsx("data/AirbnbLA_2023.xlsx")

# View the data
head(airbnb) #use head for a sample
```

```
## # A tibble: 6 x 32
##      Id `Host Id` `Host Name`    `Host Is Superhost` `Host Acceptance Rate`
##   <dbl>     <dbl> <chr>          <lgl>               <chr>
## 1   109       521 Paolo          FALSE               50%
## 2  2708      3008 Chas.          TRUE                100%
## 3  2732      3041 Yoga Priestess FALSE               42%
## 4 63416    309512 Vincenzo       TRUE                96%
## 5 67089    210344 Brenna         TRUE                95%
## 6  5728      9171 Sanni          FALSE               79%
## # i 27 more variables: `Host Response Rate` <chr>, `Host Response Time` <chr>,
## #   `Host Since` <dttm>, `Neighbourhood Group` <chr>, Neighbourhood <chr>,
## #   Latitude <dbl>, Longitude <dbl>, `Room Type` <chr>, Accommodates <dbl>,
## #   Beds <dbl>, Price <dbl>, `Instant Bookable` <lgl>, `First Review` <dttm>,
## #   `Last Review` <dttm>, License <chr>, `Reviews Per Month` <dbl>,
```

```
## #   `Minimum Nights` <dbl>, `Number Of Reviews` <dbl>,
## #   `Number Of Reviews L30D` <dbl>, `Number Of Reviews Ltm` <dbl>, ...
```

Perform Data Cleaning

```r
# Initial data cleaning and renaming columns
airbnb_cleaned <- airbnb %>%
  rename('id' = 'Id',
         'host_id' = 'Host Id',
         'host_name' = 'Host Name',
         'host_is_superhost' = 'Host Is Superhost',
         'host_acceptance_rate' = 'Host Acceptance Rate',
         'host_response_rate' = 'Host Response Rate',
         'host_response_time' = 'Host Response Time',
         'host_since' = 'Host Since',
         'neighbourhood_group' = 'Neighbourhood Group',
         'neighbourhood' = 'Neighbourhood',
         'latitude' = 'Latitude',
         'longitude' = 'Longitude',
         'room_type' = 'Room Type',
         'accommodates' = 'Accommodates',
         'beds' = 'Beds',
         'price' = 'Price',
         'instant_bookable' = 'Instant Bookable',
         'first_review' = 'First Review',
         'last_review' = 'Last Review',
         'license' = 'License',
         'reviews_per_month' = 'Reviews Per Month',
         'minimum_nights' = 'Minimum Nights',
         'number_of_reviews' = 'Number Of Reviews',
         'number_of_reviews_l30d' = 'Number Of Reviews L30D',
         'number_of_reviews_ltm' = 'Number Of Reviews Ltm',
         'review_scores_rating' = 'Review Scores Rating',
         'review_scores_accuracy' = 'Review Scores Accuracy',
         'review_scores_checkin' = 'Review Scores Checkin',
         'review_scores_cleanliness' = 'Review Scores Cleanliness',
         'review_scores_communication' = 'Review Scores Communication',
         'review_scores_location' = 'Review Scores Location',
         'review_scores_value' = 'Review Scores Value'
  )

#drop licence column
airbnb_cleaned <- select(airbnb_cleaned, -license)

# Convert "N/A" values to NA
airbnb_cleaned <- airbnb_cleaned %>%
  mutate(
    host_acceptance_rate = if_else(host_acceptance_rate == "N/A", NA, host_acceptance_rate),
    host_response_rate = if_else(host_response_rate == "N/A", NA, host_response_rate),
    host_response_time = if_else(host_response_time == "N/A", NA, host_response_time)
  )

# Impute missing review scores with the mean value of each column
airbnb_cleaned <- airbnb_cleaned %>%
```

```r
  mutate(
    review_scores_accuracy = if_else(is.na(review_scores_accuracy), mean(review_scores_accuracy, na.rm =
    review_scores_checkin = if_else(is.na(review_scores_checkin), mean(review_scores_checkin, na.rm = TR
    review_scores_cleanliness = if_else(is.na(review_scores_cleanliness), mean(review_scores_cleanliness
    review_scores_communication = if_else(is.na(review_scores_communication), mean(review_scores_communi
    review_scores_location = if_else(is.na(review_scores_location), mean(review_scores_location, na.rm =
    review_scores_value = if_else(is.na(review_scores_value), mean(review_scores_value, na.rm = TRUE), r
  )

# Check for missing values again
colSums(is.na(airbnb_cleaned))
```

```
##                          id                     host_id
##                           0                           0
##                   host_name           host_is_superhost
##                           0                           0
##        host_acceptance_rate          host_response_rate
##                        4903                        6076
##          host_response_time                  host_since
##                        6076                           0
##         neighbourhood_group               neighbourhood
##                           0                           0
##                    latitude                   longitude
##                           0                           0
##                   room_type                accommodates
##                           0                           0
##                        beds                       price
##                           0                           0
##            instant_bookable                first_review
##                           0                           0
##                 last_review           reviews_per_month
##                           0                           0
##              minimum_nights           number_of_reviews
##                           0                           0
##      number_of_reviews_l30d       number_of_reviews_ltm
##                           0                           0
##        review_scores_rating      review_scores_accuracy
##                           0                           0
##       review_scores_checkin   review_scores_cleanliness
##                           0                           0
## review_scores_communication      review_scores_location
##                           0                           0
##         review_scores_value
##                           0
```

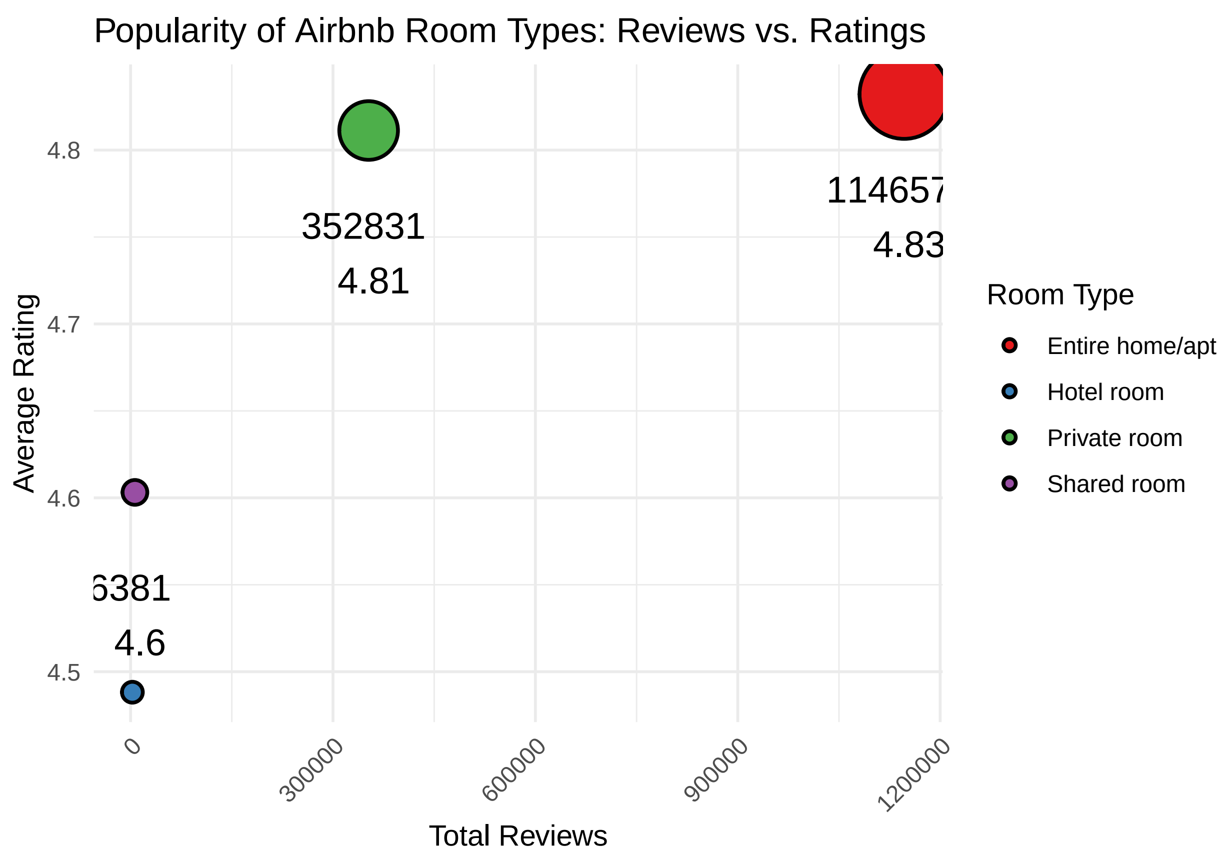
```r
sum(is.na(airbnb_cleaned))
```

```
## [1] 17055
```

Question 1: Which type of Airbnb properties garner the most reviews, indicating popularity?

```r
#Summarise the total review, avg rating, no. of reviews across room type
```

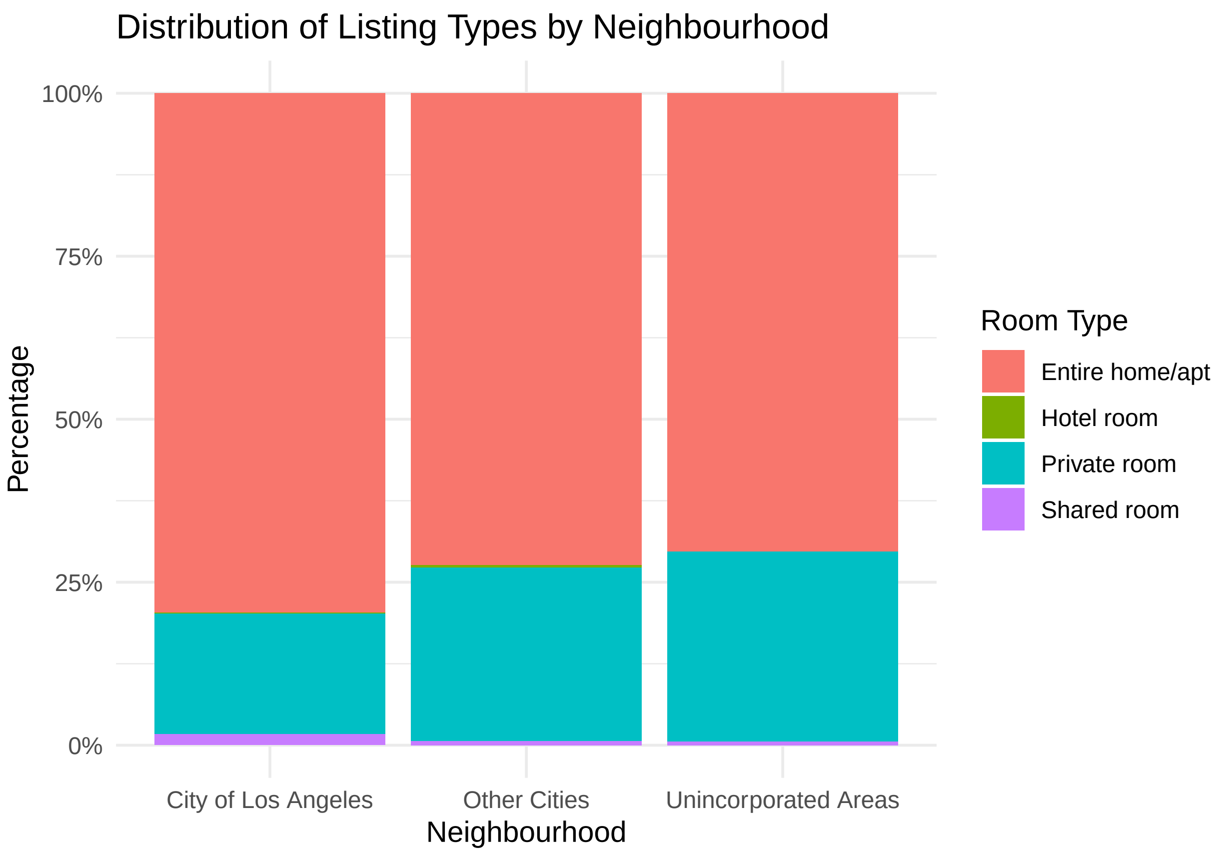
```r
airbnb_popularity <- airbnb_cleaned %>%
  group_by(room_type) %>%
  summarise(total_reviews = sum(number_of_reviews),
            avg_rating = sum(review_scores_rating * number_of_reviews) / sum(number_of_reviews))

# Bubble chart for total reviews and avg_rating

ggplot(airbnb_popularity, aes(x = total_reviews, y = avg_rating, size = total_reviews, fill = room_type
  geom_point(shape = 21, color = "black", stroke = 1) +
  geom_text(aes(label = paste(total_reviews, "\n", round(avg_rating, 2))),
            vjust = 2, color = "black", size = 5) +
  scale_size(range = c(3, 15), guide = "none") +
  scale_fill_brewer(palette = "Set1") +
  labs(title = "Popularity of Airbnb Room Types: Reviews vs. Ratings",
       x = "Total Reviews",
       y = "Average Rating",
       fill = "Room Type") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Question 2: How does the distribution of listing types vary across different neighborhoods or regions?

```r
airbnb_summary <- airbnb_cleaned %>%
  group_by(neighbourhood_group, room_type) %>%
  summarise(count = n(), .groups='drop') %>%
```

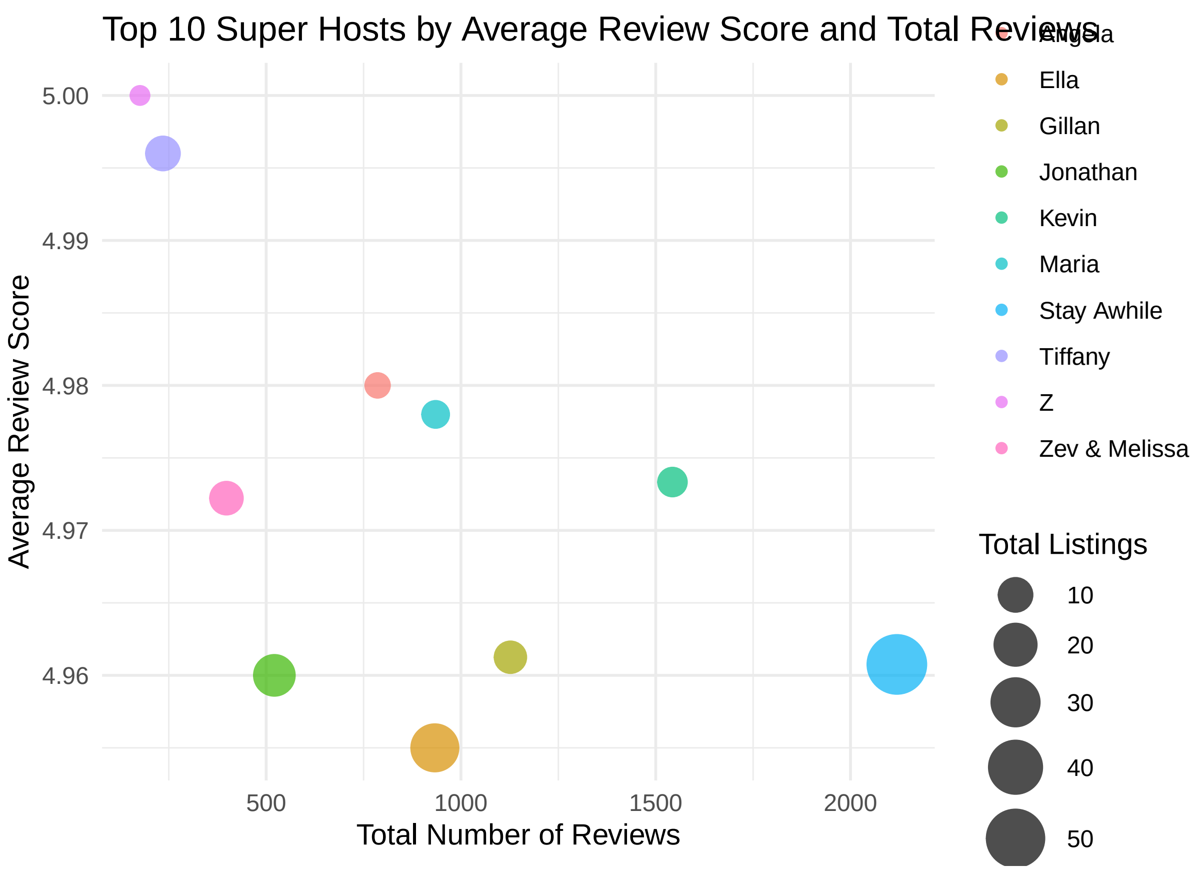
```
  mutate(percentage = count / sum(count) * 100)

# Plot the 100% stacked bar chart for distribution of listing types vary across different neighborhoods

ggplot(airbnb_summary, aes(x = neighbourhood_group, y = percentage, fill = room_type)) +
  geom_bar(stat = "identity", position = "fill") +
  scale_y_continuous(labels = scales::percent) +
  labs(x = "Neighbourhood", y = "Percentage", fill = "Room Type",
       title = "Distribution of Listing Types by Neighbourhood") +
  theme_minimal()
```



Question 3: Who are the top 10 Super hosts based on listings, review scores, and number of reviews? How do their listing and review score distributions vary?

```
#Assign the rank to each host based on review score rating, number of reviews and total listings
airbnb_groupby_unique_host <- airbnb_cleaned %>%
  filter(host_is_superhost = TRUE) %>%
  group_by(host_id, host_name) %>%
  summarise(avg_review_score = mean(review_scores_rating),
            total_reviews = sum(number_of_reviews),
            total_listing = n()) %>%
  ungroup() %>%  # Corrected to 'ungroup()'
  mutate(
    rank_review_score = rank(-avg_review_score),  # Rank by average review score rating
    rank_number_of_reviews = rank(-total_reviews), # Rank by number of reviews
    rank_number_of_listings = rank(-total_listing) # Rank by maximum number of listings
```

```
)


# Combine the ranks (e.g., by summing them up for an overall rank)
filtered_data <- airbnb_groupby_unique_host %>%
  mutate(overall_rank = rank_review_score + rank_number_of_reviews + rank_number_of_listings) %>%
  arrange(overall_rank) # Sort by the combined rank

# Select the top 10 super hosts
top_10_hosts <- filtered_data %>%
  slice_head(n = 10) %>%
  select(host_id, host_name, avg_review_score, total_reviews, total_listing, overall_rank)


#Visualize top 10 Super hosts based on listings, review scores, and number of reviews
ggplot(top_10_hosts, aes(x = total_reviews, y = avg_review_score, color = host_name, size=total_listing
  geom_point(alpha = 0.7) +
  scale_size_continuous(range = c(3, 10)) +
  labs(title = "Top 10 Super Hosts by Average Review Score and Total Reviews",
       x = "Total Number of Reviews",
       y = "Average Review Score",
       size = "Total Listings",
       color = "Super Host Name") +
  theme_minimal()
```



Question 4: What is the overall price trend for different room types on Airbnb?

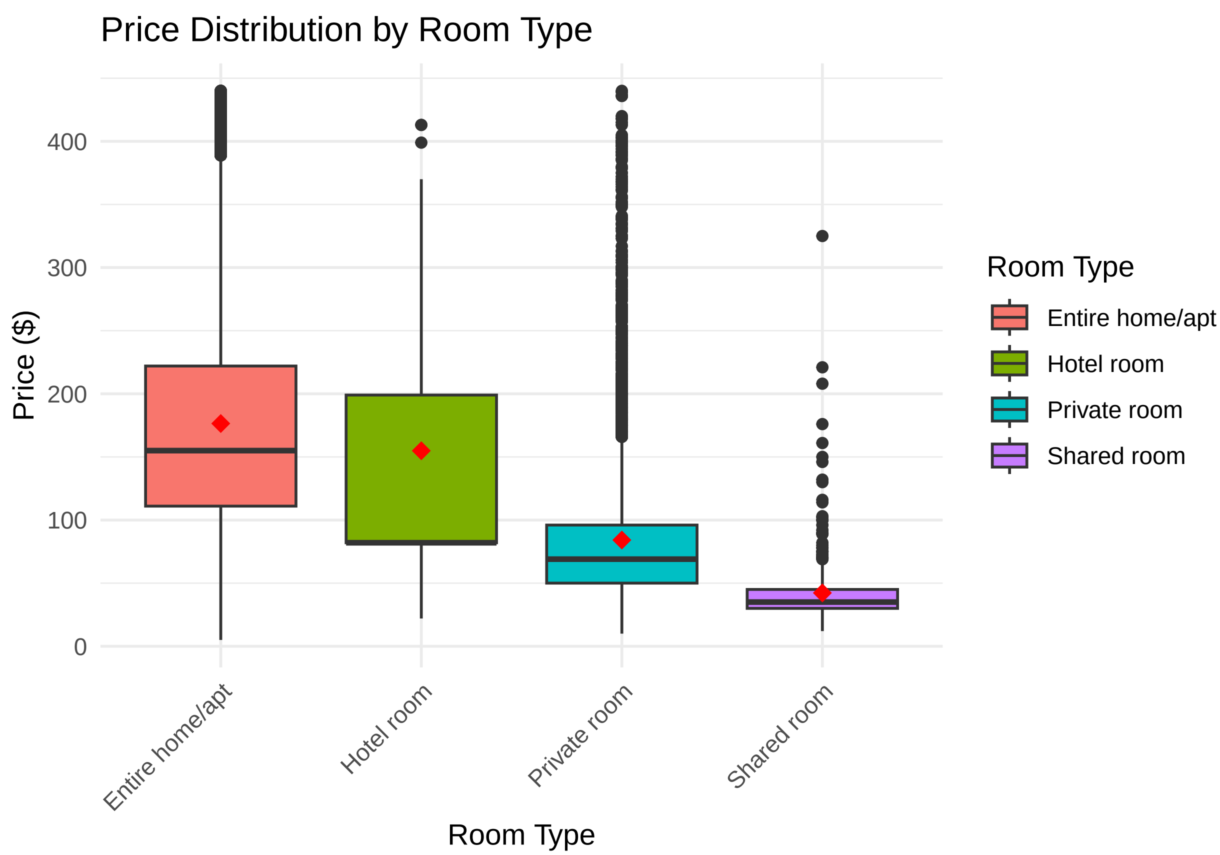
```
# Calculate summary statistics for room types and see for price outliers

summary_stats_by_roomType <- airbnb_cleaned %>%
  group_by(room_type) %>%
  summarise(
    Average_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),
    SD_Price = sd(price, na.rm = TRUE)
  )

print(summary_stats_by_roomType)
```

```
## # A tibble: 4 x 6
##   room_type       Average_Price Median_Price Min_Price Max_Price SD_Price
##   <chr>                   <dbl>        <dbl>     <dbl>     <dbl>    <dbl>
## 1 Entire home/apt          268.          170         5     99999    777.
## 2 Hotel room               798.          100.       22      9999   2439.
## 3 Private room             118.           69        10     99999   1204.
## 4 Shared room               53.7          35        12      1200     95.3
```

```
# Data Cleaning: Remove outliers in price using the IQR method
remove_price_outliers <- function(data) {
  Q1 <- quantile(data$price, 0.25, na.rm = TRUE)
  Q3 <- quantile(data$price, 0.75, na.rm = TRUE)
  IQR_value <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR_value
  upper_bound <- Q3 + 1.5 * IQR_value
  data %>% filter(price >= lower_bound & price <= upper_bound)
}

airbnb_filtered <- remove_price_outliers(airbnb_cleaned)
```

Plot Analysis Question4

```
# Box plot of Price Distribution by Room Type
ggplot(airbnb_filtered, aes(x = room_type, y = price, fill = room_type)) +
  geom_boxplot() +
  stat_summary(fun = "mean", geom = "point", shape = 18, size = 3, color = "red", fill = "red",
               position = position_dodge(width = 0.75)) +  # Adjusts the position of the mean marker
  labs(title = "Price Distribution by Room Type",
       x = "Room Type",
       y = "Price ($)",
       fill = "Room Type") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Question 5: How does the average price of Airbnb listings vary across different neighborhoods in Los Angeles?

```r
# Group the data by neighborhood and calculate average price
df_grouped <- airbnb_filtered %>%
  group_by(neighbourhood) %>%
  summarise(
    Avg_Price = mean(price, na.rm = TRUE),
    latitude = first(latitude),
    longitude = first(longitude),
    .groups = 'drop'
  )

print(df_grouped)
```

```
## # A tibble: 265 x 4
##    neighbourhood   Avg_Price latitude longitude
##    <chr>               <dbl>    <dbl>     <dbl>
##  1 Acton               171.      34.5     -118.
##  2 Adams-Normandie      90.5     34.0     -118.
##  3 Agoura Hills        174.      34.2     -119.
##  4 Agua Dulce          158.      34.5     -118.
##  5 Alhambra            128.      34.1     -118.
##  6 Alondra Park        178.      33.9     -118.
##  7 Altadena            155.      34.2     -118.
##  8 Angeles Crest       168.      34.4     -118.
```

```
##   9 Arcadia             123.      34.1     -118.
## 10 Arleta              100       34.2     -118.
## # i 255 more rows
```

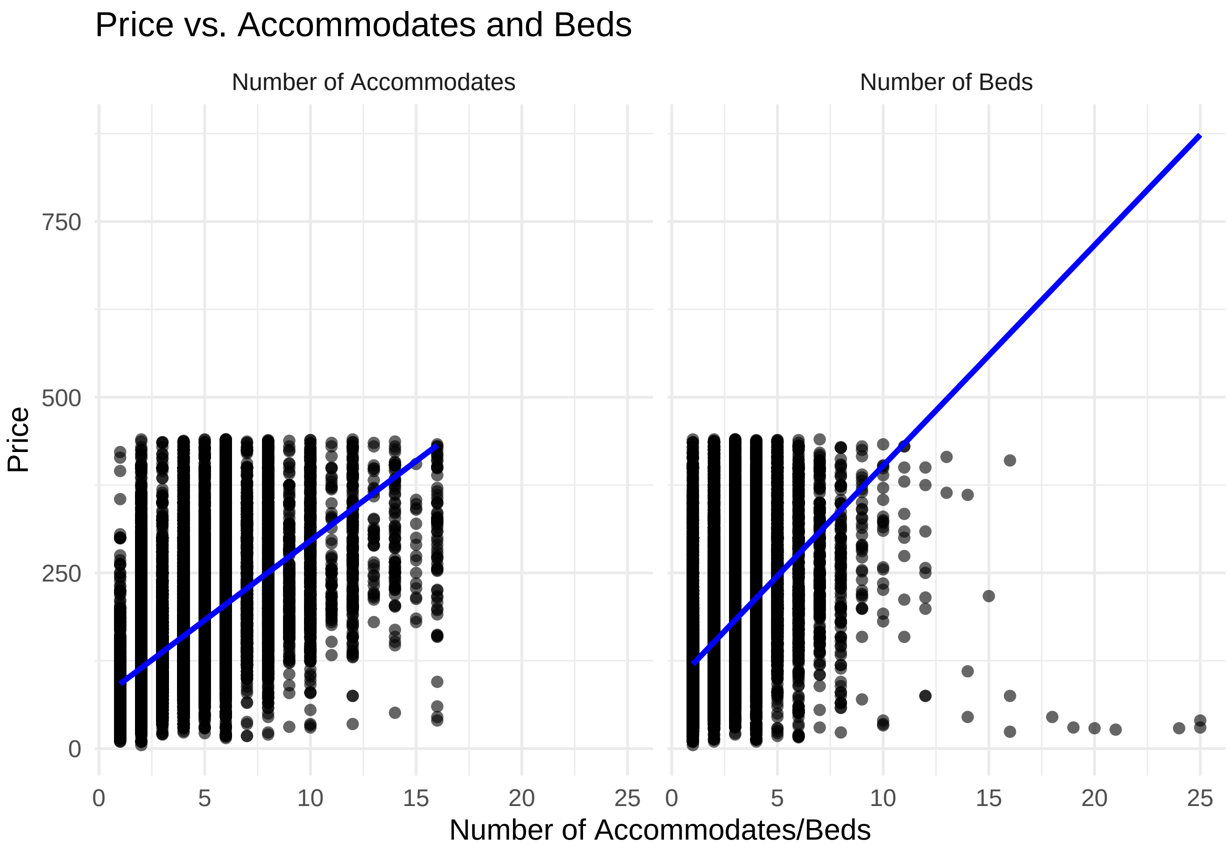
```
# Create a color palette based on average prices
pal <- colorNumeric(palette = "viridis", domain = df_grouped$Avg_Price)

# Create the interactive map with color tones
leaflet(df_grouped) %>%
  addTiles() %>%
  addCircleMarkers(
    lng = ~longitude,
    lat = ~latitude,
    radius = ~Avg_Price / 50,
    popup = ~paste(neighbourhood, ": $", round(Avg_Price, 2)),
    color = ~pal(Avg_Price),
    fillOpacity = 0.7
  ) %>%
  setView(lng = mean(df_grouped$longitude), lat = mean(df_grouped$latitude), zoom = 11) %>%
  addLegend("bottomright", pal = pal, values = ~Avg_Price,
            title = "Average Price",
            opacity = 0.7)
```

Data Modeling Visulaization

```
# Impact of Beds and Accommodates on Price of Room Types
# Checking which has more impact: Accommodates or Beds
correlation_matrix <- airbnb_filtered %>%
  select(price, accommodates, beds) %>%
  cor()
print(correlation_matrix)
```

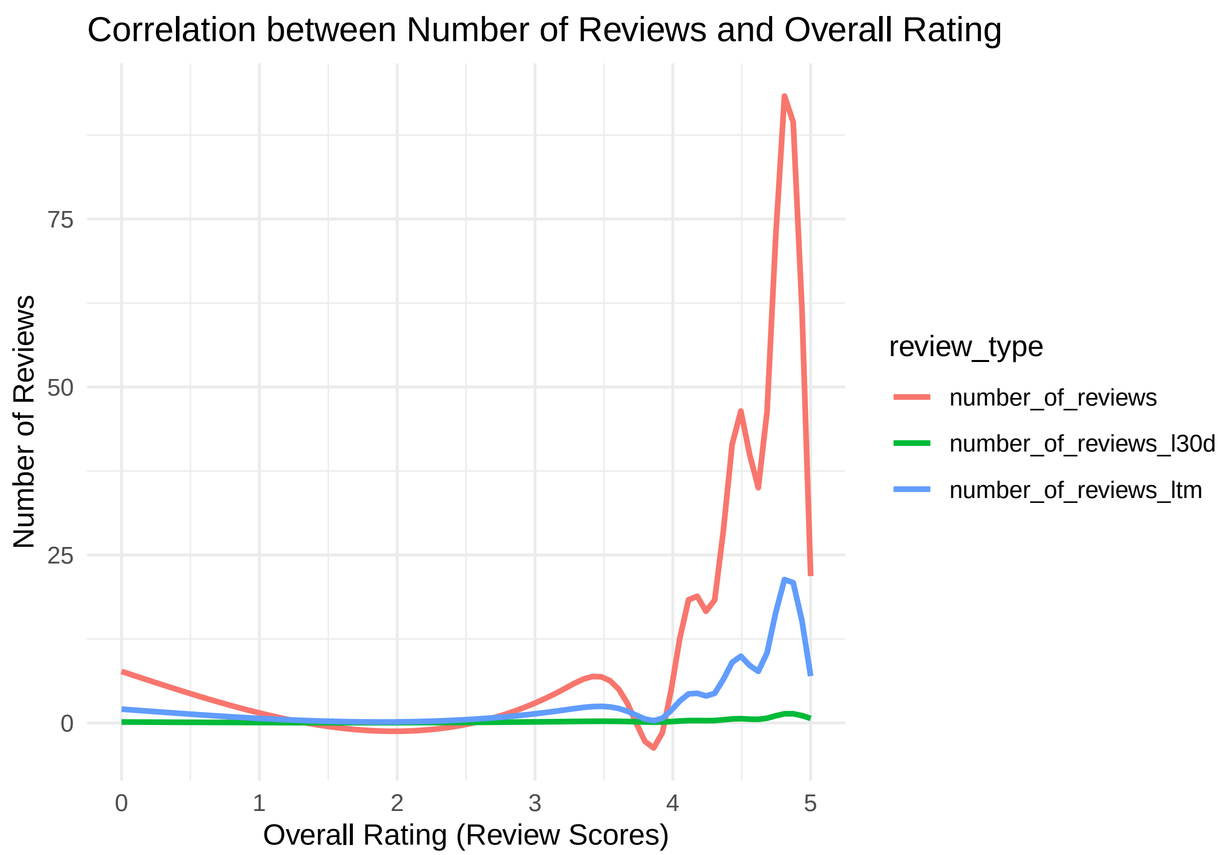
```
##                  price accommodates      beds
## price        1.0000000    0.6022625 0.4964012
## accommodates 0.6022625    1.0000000 0.8219663
## beds         0.4964012    0.8219663 1.0000000
```

```
# Reshape the data for combined plotting
airbnb_long <- airbnb_filtered %>%
  pivot_longer(cols = c(beds, accommodates), names_to = "Type", values_to = "Value")

# Create a single graph for Price vs. Beds and Price vs. Accommodates
ggplot(airbnb_long, aes(x = Value, y = price)) +
  geom_point(alpha = 0.6) +  # Adjust transparency for better visibility
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Add linear regression line
  labs(title = "Price vs. Accommodates and Beds",
       x = "Number of Accommodates/Beds",
       y = "Price") +
  facet_wrap(~ Type, labeller = as_labeller(c(beds = "Number of Beds", accommodates = "Number of Accomm
  theme_minimal()
```



Question 6: Is there a correlation between the number of reviews and overall ratings? Do hosts with more reviews tend to have better ratings?

```
# Transform data for faceting
airbnb_long <- airbnb_cleaned %>%
  pivot_longer(cols = c(number_of_reviews, number_of_reviews_l30d, number_of_reviews_ltm),
               names_to = "review_type",
               values_to = "review_count")

# Creating a scatter plot with a trend line
ggplot(airbnb_long, aes(x =review_scores_rating , y = review_count)) +
  geom_smooth(aes(color = review_type), se=FALSE) +
  labs(title = "Correlation between Number of Reviews and Overall Rating",
       x = "Overall Rating (Review Scores)",
       y = "Number of Reviews") +
  theme_minimal()
```



Question 7: Which factors, such as the check-in process, cleanliness, accuracy of listing descriptions, etc., most significantly impact review ratings?

```
cor_matrix <- cor(airbnb_cleaned[, c("review_scores_rating", "review_scores_accuracy",
                                     "review_scores_cleanliness", "review_scores_communication",
                                     "review_scores_checkin", "review_scores_value",
                                     "review_scores_location")],
                  use = "complete.obs")

print("Correlation matrix of factors impacting review ratings:")
```

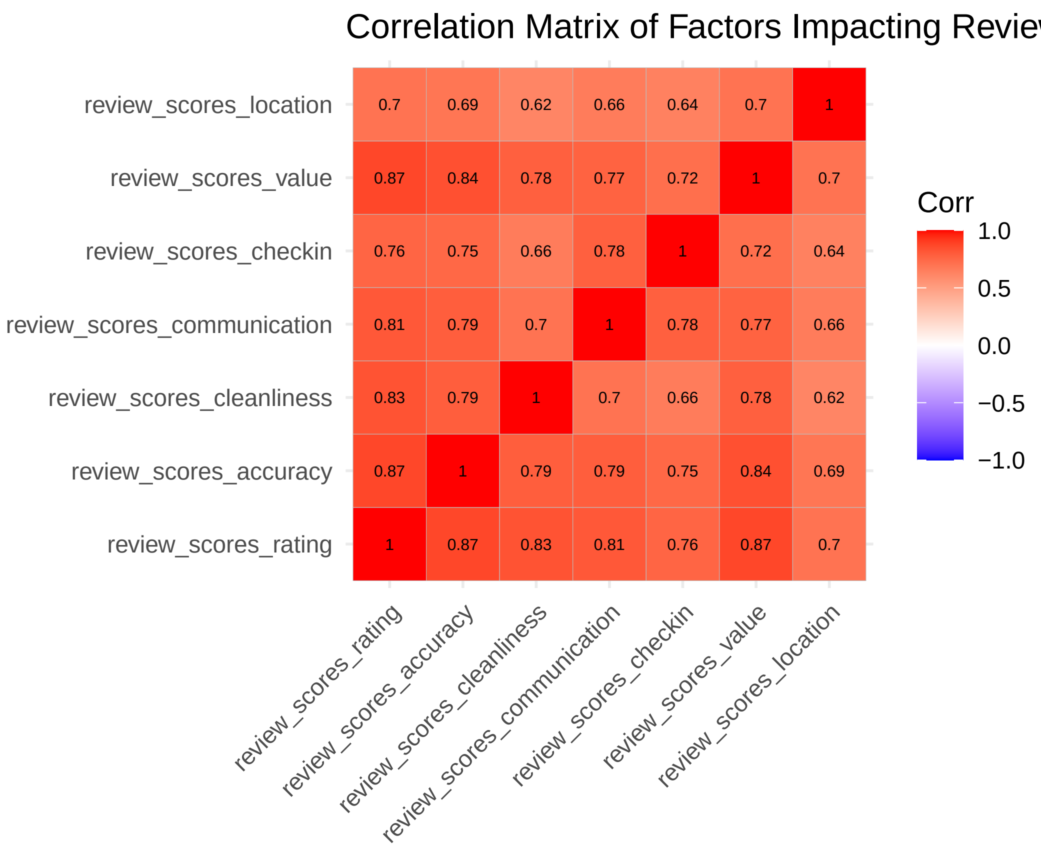
```
## [1] "Correlation matrix of factors impacting review ratings:"
```

```
print(cor_matrix)
```

```
##                             review_scores_rating review_scores_accuracy
## review_scores_rating                   1.0000000              0.8731444
## review_scores_accuracy                 0.8731444              1.0000000
## review_scores_cleanliness              0.8281561              0.7925776
## review_scores_communication            0.8103266              0.7936822
## review_scores_checkin                  0.7569332              0.7532662
## review_scores_value                    0.8691172              0.8431411
## review_scores_location                 0.6952753              0.6936656
##                             review_scores_cleanliness
## review_scores_rating                        0.8281561
## review_scores_accuracy                      0.7925776
## review_scores_cleanliness                   1.0000000
```

```
## review_scores_communication                  0.6999623
## review_scores_checkin                        0.6647535
## review_scores_value                          0.7751937
## review_scores_location                       0.6201472
##                             review_scores_communication review_scores_checkin
## review_scores_rating                          0.8103266             0.7569332
## review_scores_accuracy                        0.7936822             0.7532662
## review_scores_cleanliness                     0.6999623             0.6647535
## review_scores_communication                   1.0000000             0.7836596
## review_scores_checkin                         0.7836596             1.0000000
## review_scores_value                           0.7681240             0.7180183
## review_scores_location                        0.6556051             0.6442605
##                             review_scores_value review_scores_location
## review_scores_rating                  0.8691172              0.6952753
## review_scores_accuracy                0.8431411              0.6936656
## review_scores_cleanliness             0.7751937              0.6201472
## review_scores_communication           0.7681240              0.6556051
## review_scores_checkin                 0.7180183              0.6442605
## review_scores_value                   1.0000000              0.6993991
## review_scores_location                0.6993991              1.0000000
```

```
# Visualize correlations
ggcorrplot(cor_matrix, lab = TRUE, lab_size = 2, title = "Correlation Matrix of Factors Impacting Revie
  theme(plot.title = element_text(size = 13), axis.text.x = element_text(size = 9), axis.text.y = elemen
```



Summary Statistics

```
#Summary Statistics for Listing Capacity across Room Type

get_mode <- function(x) {
  uniqx <- unique(x)
  uniqx[which.max(tabulate(match(x, uniqx)))]
}

summary_stats_accommodates <- airbnb_cleaned %>%
  group_by(room_type) %>%
  summarize(
    mean = mean(accommodates, na.rm = TRUE),
    median = median(accommodates, na.rm = TRUE),
    min = min(accommodates, na.rm = TRUE),
    max = max(accommodates, na.rm = TRUE),
    sd = sd(accommodates, na.rm = TRUE),
    mode = get_mode(accommodates),
    Inter_quertile = IQR(accommodates, na.rm=TRUE),
    count = n())

summary_stats_accommodates
```

```
## # A tibble: 4 x 9
##   room_type        mean median   min   max    sd  mode Inter_quertile count
##   <chr>           <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>          <dbl> <int>
## 1 Entire home/apt  4.66      4     1    16 2.81      2              4 24493
## 2 Hotel room       2.34      2     1     6 0.922     2              0    62
## 3 Private room     1.99      2     1    16 1.11      2              1  7530
## 4 Shared room      2.24      1     1    16 2.33      1              1   364
```

Build and Evaluate Price Prediction Model

```
#Load the libraries

library(rpart) # for creating decision tree model
library(rattle) # for plotting decision tree model
library(caret) # for evaluating decision tree model
library(randomForest) # for creating random forest model
library(rpart.plot) #for creating rpart plot
library(ISLR)

# Data Preparation and Feature Engineering

#Select relevant features from the original dataset

airbnb_model <- airbnb_filtered %>%
  select(
    host_is_superhost,
    neighbourhood_group,
    latitude, longitude,
    room_type,
    accommodates,
    beds,
```

```
    price,
    instant_bookable,
    minimum_nights
  )

# Apply One-Hot Encoding for categorical variables

dummy <- dummyVars(price ~ host_is_superhost + neighbourhood_group + room_type + instant_bookable, data
 # Generate dummy variables
one_hot_encoded <- predict(dummy, newdata = airbnb_model)
# Convert the result into a data frame
one_hot_encoded_df <- as.data.frame(one_hot_encoded)

# Update the dataset: Replace original categorical columns with one-hot encoded variables
airbnb_model <- cbind(
  airbnb_model %>% select(-host_is_superhost, -neighbourhood_group, -room_type, -instant_bookable),  #
  one_hot_encoded_df  # Add the new one-hot encoded columns
)

# Apply RobustScaler to numeric features
robust_scale <- function(x) {
  (x - median(x, na.rm = TRUE)) / IQR(x, na.rm = TRUE)  # Formula: (x - median) / IQR
}

# Specify numeric columns to scale
num_cols <- c("latitude", "longitude", "accommodates", "beds", "minimum_nights")

# Scale the numeric columns
airbnb_model[num_cols] <- lapply(airbnb_model[num_cols], robust_scale)

# -------------------Feature Engineering------------------#

# Create new features to improve the model
airbnb_model$accommodates_beds <- airbnb_model$accommodates * airbnb_model$beds  # Interaction feature:
airbnb_model$NEW_total_cost <- airbnb_model$price * airbnb_model$minimum_nights  # Total cost: price ×

# Clean column names

# Replace spaces with underscores, remove special characters, and convert to lowercase for consistency
colnames(airbnb_model) <- colnames(airbnb_model) %>%
  gsub(" ", "_", .) %>%
  gsub("[^A-Za-z0-9_]", "", .) %>%
  tolower()
```

Split the Data

```
# Separate the target variable (y) from the features (X)
y <- airbnb_model$price  # Target variable: price
X <- airbnb_model %>% select(-price)

# Split the dataset into training (70%) and testing (30%) sets

set.seed(17)  # Set random seed for reproducibility
```

```r
train_indices <- sample(1:nrow(airbnb_model), 0.7 * nrow(airbnb_model))  # Randomly select 70% of rows
X_train <- X[train_indices, ]  # Training features
X_test <- X[-train_indices, ]  # Testing features
y_train <- y[train_indices]    # Training target
y_test <- y[-train_indices]    # Testing target
```

Linear Regression Model

```r
#Train the Linear Regression Model

linear_model <- lm(price ~ . + I(accommodates^2), data = cbind(X_train, price = y_train))

#Predictions and Evaluation
predictions_lm <- predict(linear_model, newdata = X_test)

# Calculate evaluation metrics for Linear Regression
mse_lm <- mean(((y_test) - predictions_lm)^2)  # Mean Squared Error
rmse_lm <- sqrt(mse_lm)                        # Root Mean Squared Error
SS_res_lm <- sum(((y_test) - predictions_lm)^2)  # Residual sum of squares
SS_tot_lm <- sum(((y_test) - mean((y_test)))^2)    # Total sum of squares
r_squared_lm <- 1 - (SS_res_lm / SS_tot_lm)    # R-squared

# Number of observations and predictors
n <- nrow(X_test)
k <- ncol(X_train) + 1

# Adjusted R-squared
adjusted_r_squared_lm <- 1 - ((1 - r_squared_lm) * (n - 1) / (n - k - 1))

# Akaike Information Criterion (AIC)
RSS <- SS_res_lm  # Residual Sum of Squares
AIC_lm <- n * log(RSS / n) + 2 * k

# Print evaluation metrics for Linear Regression
cat("Linear Regression Model - Mean Squared Error (MSE):", mse_lm, "\n")
```

```
## Linear Regression Model - Mean Squared Error (MSE): 3687.971
```

```r
cat("Linear Regression Model - Root Mean Squared Error (RMSE):", rmse_lm, "\n")
```

```
## Linear Regression Model - Root Mean Squared Error (RMSE): 60.72867
```

```r
cat("Linear Regression Model - R-squared (R²):", r_squared_lm, "\n")
```

```
## Linear Regression Model - R-squared (R²): 0.5256395
```

```r
cat("Linear Regression Model - Adjusted R-squared (Adj R²):", adjusted_r_squared_lm, "\n")
```

```
## Linear Regression Model - Adjusted R-squared (Adj R²): 0.5246211
```

```r
cat("Linear Regression Model - Akaike Information Criterion (AIC):", AIC_lm, "\n")
```

```
## Linear Regression Model - Akaike Information Criterion (AIC): 72885.82
```

Decision Tree

```r
#Train the Decision Tree Model
train_data <- cbind(X_train, price = y_train)

# Set stopping parameters
control_params <- rpart.control(
  minsplit = 20,    # Minimum observations to split a node
  minbucket = 5,    # Minimum observations in a leaf node
  cp = 0.01,        # Complexity parameter
  maxdepth = 30     # Maximum depth of the tree
)


#Train the model
dt_model <- rpart(price ~ ., data = train_data, method =  "anova", control = control_params)

#Make Predictions for Decision Tree
predictions_dt <- predict(dt_model, newdata = X_test)

#Evaluate the Decision Tree Model

# Mean Squared Error (MSE)
mse_dt <- mean((y_test - predictions_dt)^2)
# Root Mean Squared Error (RMSE)
rmse_dt <- sqrt(mse_dt)
# Residual sum of squares
SS_res_dt <- sum((y_test - predictions_dt)^2)
# Total sum of squares
SS_tot_dt <- sum((y_test - mean(y_test))^2)
# R-squared (proportion of variance explained)
r_squared_dt <- 1 - (SS_res_dt / SS_tot_dt)

# Number of observations and predictors
n <- nrow(X_test)
k <- ncol(X_train) + 1

# Adjusted R-squared
adjusted_r_squared_dt <- 1 - ((1 - r_squared_dt) * (n - 1) / (n - k - 1))

# Akaike Information Criterion (AIC)
RSS <- SS_res_dt  # Residual Sum of Squares
AIC_dt <- n * log(RSS / n) + 2 * k

# Print evaluation metrics for Decision Tree
cat("Decision Tree Model - Mean Squared Error (MSE):", mse_dt, "\n")
```

```
## Decision Tree Model - Mean Squared Error (MSE): 1856.663
```

```r
cat("Decision Tree Model - Root Mean Squared Error (RMSE):", rmse_dt, "\n")
```

```
## Decision Tree Model - Root Mean Squared Error (RMSE): 43.08901
```

```r
cat("Decision Tree Model - R-squared (R²):", r_squared_dt, "\n")
```

```
## Decision Tree Model - R-squared (R²): 0.7611892
```

```r
cat("Decision Tree Model - Adjusted R-squared (Adj R²):", adjusted_r_squared_dt, "\n")
```

```
## Decision Tree Model - Adjusted R-squared (Adj R²): 0.7606765
```

```r
cat("Decision Tree Model - Akaike Information Criterion (AIC):", AIC_dt, "\n")
```

```
## Decision Tree Model - Akaike Information Criterion (AIC): 66798.37
```

Check if Decision Tree model is overfitted or not.

```r
#Predictions for Decision Tree on Training Data
predictions_dt_train <- predict(dt_model, newdata = X_train)

mse_dt_train <- mean((y_train - predictions_dt_train)^2)
rmse_dt_train <- sqrt(mse_dt_train)
SS_res_dt_train <- sum((y_train - predictions_dt_train)^2)
SS_tot_dt_train <- sum((y_train - mean(y_train))^2)

# R-squared (proportion of variance explained)
r_squared_dt_train <- 1 - (SS_res_dt_train / SS_tot_dt_train)

# Print the Training Metrics
cat("Training Metrics for Decision Tree Model:\n")
```

```
## Training Metrics for Decision Tree Model:
```

```r
cat("DT MSE (Train):", mse_dt_train, "\n")
```

```
## DT MSE (Train): 1796.165
```

```r
cat("DT RMSE (Train):", rmse_dt_train, "\n")
```

```
## DT RMSE (Train): 42.38118
```

```r
cat("DT R-squared (Train):", r_squared_dt_train, "\n")
```

```
## DT R-squared (Train): 0.7716274
```

Random Forest

```r
#Train the Random Forest Model
rf_model <- randomForest(x = X_train, y = y_train, ntree = 100, mtry = 3)


# Predictions for Random Forest
predictions_rf <- predict(rf_model, newdata = X_test)

# Evaluate the Random Forest Model
# Mean Squared Error (MSE)
mse_rf <- mean((y_test - predictions_rf)^2)
# Root Mean Squared Error (RMSE)
rmse_rf <- sqrt(mse_rf)
# Residual sum of squares
SS_res_rf <- sum((y_test - predictions_rf)^2)
# Total sum of squares
SS_tot_rf <- sum((y_test - mean(y_test))^2)
```

```r
# R-squared (proportion of variance explained)
r_squared_rf <- 1 - (SS_res_rf / SS_tot_rf)
#

# Number of observations and predictors
n <- nrow(X_test)
k <- ncol(X_train) + 1

# Adjusted R-squared
adjusted_r_squared_rf <- 1 - ((1 - r_squared_rf) * (n - 1) / (n - k - 1))

# Akaike Information Criterion (AIC)
RSS <- SS_res_rf  # Residual Sum of Squares
AIC_rf <- n * log(RSS / n) + 2 * k

# Print evaluation metrics for Random Forest
cat("Random Forest Model - Mean Squared Error (MSE):", mse_rf, "\n")
```

```
## Random Forest Model - Mean Squared Error (MSE): 1184.576
```

```r
cat("Random Forest Model - Root Mean Squared Error (RMSE):", rmse_rf, "\n")
```

```
## Random Forest Model - Root Mean Squared Error (RMSE): 34.41767
```

```r
cat("Random Forest Model - R-squared (R²):", r_squared_rf, "\n")
```

```
## Random Forest Model - R-squared (R²): 0.8476354
```

```r
cat("Random Forest Model - Adjusted R-squared (Adj R²):", adjusted_r_squared_rf, "\n")
```

```
## Random Forest Model - Adjusted R-squared (Adj R²): 0.8473083
```

```r
cat("Random Forest Model - Akaike Information Criterion (AIC):", AIC_rf, "\n")
```

```
## Random Forest Model - Akaike Information Criterion (AIC): 62812.23
```

Check if the Random Forest model is overfitted or not.

```r
# Predictions for Random Forest on Training Data
predictions_rf_train <- predict(rf_model, newdata = X_train)

# Evaluate the Random Forest Model on Training Data

mse_rf_train <- mean((y_train - predictions_rf_train)^2)
rmse_rf_train <- sqrt(mse_rf_train)
SS_res_rf_train <- sum((y_train - predictions_rf_train)^2)
SS_tot_rf_train <- sum((y_train - mean(y_train))^2)

# R-squared (proportion of variance explained)
r_squared_rf_train <- 1 - (SS_res_rf_train / SS_tot_rf_train)

# Print the Training Metrics
cat("Training Metrics for Random Forest Model:\n")
```

```
## Training Metrics for Random Forest Model:
```

```r
cat("RF MSE (Train):", mse_rf_train, "\n")
```

```
## RF MSE (Train): 758.1557
```

```r
cat("RF RMSE (Train):", rmse_rf_train, "\n")
```

```
## RF RMSE (Train): 27.53463
```

```r
cat("RF R-squared (Train):", r_squared_rf_train, "\n")
```

```
## RF R-squared (Train): 0.9036046
```

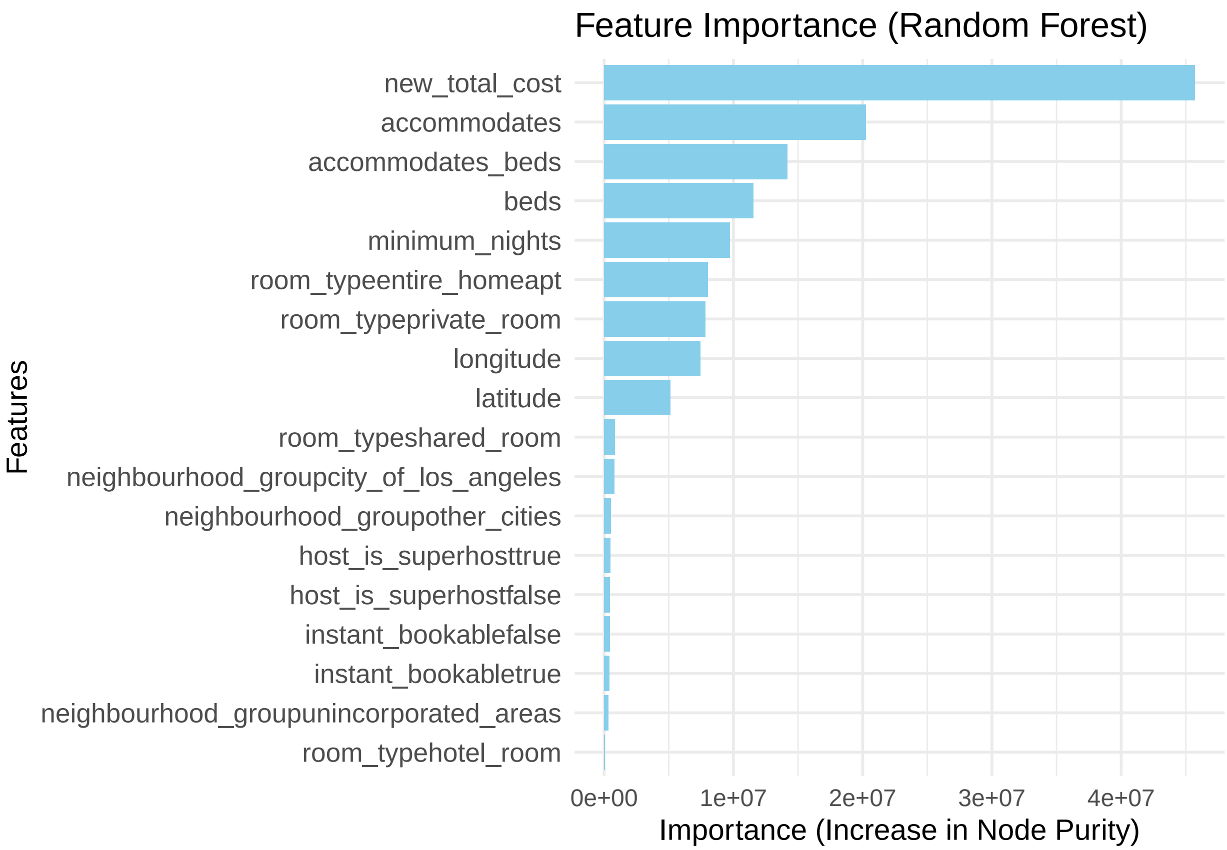
Feature Importance

```r
# Extract feature importance from the Random Forest model
feature_importance <- importance(rf_model)

# Print the feature importance to inspect its structure
print(feature_importance)
```

```
##                                         IncNodePurity
## latitude                                   5128972.44
## longitude                                  7458894.96
## accommodates                              20248698.87
## beds                                      11550359.77
## minimum_nights                             9727291.41
## host_is_superhostfalse                      451090.19
## host_is_superhosttrue                       496588.36
## neighbourhood_groupcity_of_los_angeles      797465.34
## neighbourhood_groupother_cities             512443.26
## neighbourhood_groupunincorporated_areas     342215.25
## room_typeentire_homeapt                    8028195.33
## room_typehotel_room                          58104.14
## room_typeprivate_room                      7851310.62
## room_typeshared_room                        813016.59
## instant_bookablefalse                       442452.78
## instant_bookabletrue                        402938.98
## accommodates_beds                         14161565.40
## new_total_cost                            45716653.79
```

```r
# Convert importance values into a data frame for visualization
feature_importance_df <- data.frame(
  Feature = rownames(feature_importance),
  Importance = feature_importance[, "IncNodePurity"]
) %>%
  arrange(desc(Importance))  # Sort features by importance

# Plot the feature importance
ggplot(feature_importance_df, aes(x = Importance, y = reorder(Feature, Importance))) +
  geom_bar(stat = "identity", fill = "skyblue") +  # Horizontal bar chart
  theme_minimal() +
  labs(
    title = "Feature Importance (Random Forest)",  # Chart title
    x = "Importance (Increase in Node Purity)",    # X-axis label
    y = "Features"                                # Y-axis label
  ) +
```

```r
  theme(axis.text.y = element_text(size = 10))  # Adjust font size for readability
```



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.