



WELCOME TO HOTEL

WHERE COMFORT MEETS LUXURY

OUR TEAM

AKUL GARG - 24095

NIKITA TRIPATHI- 24109

BHAVYANSHU JAIN- 24108

AYUSH GOYAL- 24074





INTRODUCTION

THIS ANALYSIS PROVIDES VALUABLE INSIGHTS INTO THE PERFORMANCE OF HOTEL PROPERTIES ACROSS MULTIPLE CITIES, LEVERAGING A COMPREHENSIVE DATASET. THE DATA CAPTURES KEY ASPECTS OF HOTEL OPERATIONS, INCLUDING ROOM BOOKINGS, OCCUPANCY RATES, AND PROPERTY CATEGORIES. BY EXAMINING THESE FACTORS, WE AIM TO UNCOVER TRENDS, PERFORMANCE METRICS, AND ACTIONABLE INSIGHTS FOR OPTIMIZING HOTEL MANAGEMENT AND CUSTOMER SATISFACTION.



DATASET DESCRIPTION

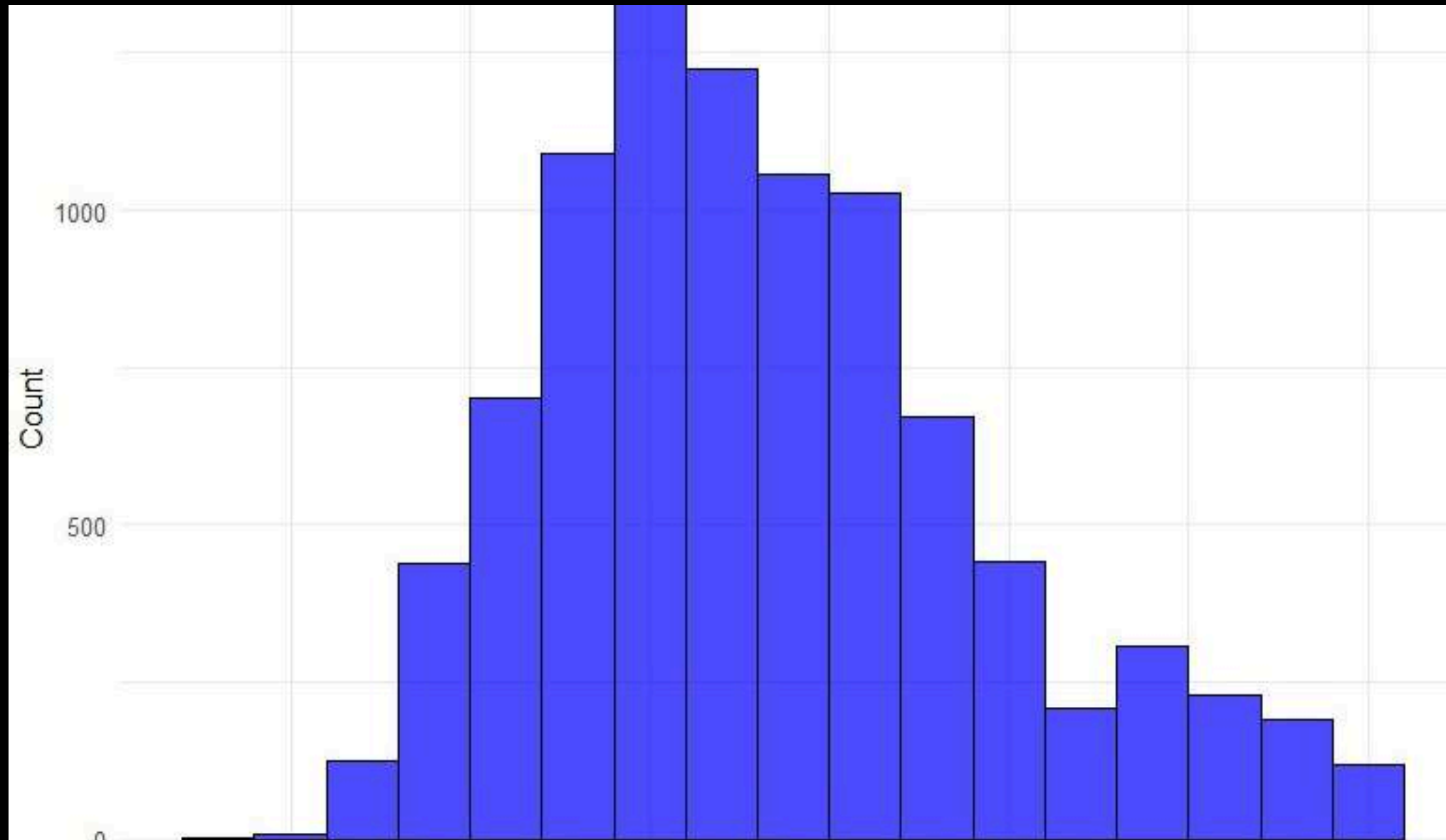
contains 9,196 rows and 11 columns, representing information about hotel bookings.

Key Features:

- 1.property_id: Unique identifier for the property (integer).
- 2.check_in_date: Date of check-in (datetime).
- 3.room_category: Type of room (e.g., Standard, Deluxe).
- 4.successful_bookings: Number of successful bookings (integer).
- 5.capacity: Maximum capacity of the property (integer).
- 6.occupancy: Occupancy rate (float, 0-1 scale).
- 7.property_name: Name of the property (categorical).
- 8.city: City where the property is located (categorical).
- 9.category: Type of property (e.g., Luxury, Business).
- 10.mm-yy: Month and year of booking (datetime).
- 11.day_type: Day type (Weekend or Weekday).



Q1. DISTRIBUTION OF OCCUPANCY RATES ACROSS ALL PROPERTIES



```
hotel_data %>%
```

```
  summarise(
```

```
    min_occupancy = min(occupancy, na.rm = TRUE),
```

```
    max_occupancy = max(occupancy, na.rm = TRUE),
```

```
    mean_occupancy = mean(occupancy, na.rm = TRUE),
```

```
    median_occupancy = median(occupancy, na.rm = TRUE),
```

```
    sd_occupancy = sd(occupancy, na.rm = TRUE)
```

```
)
```

Key Observation

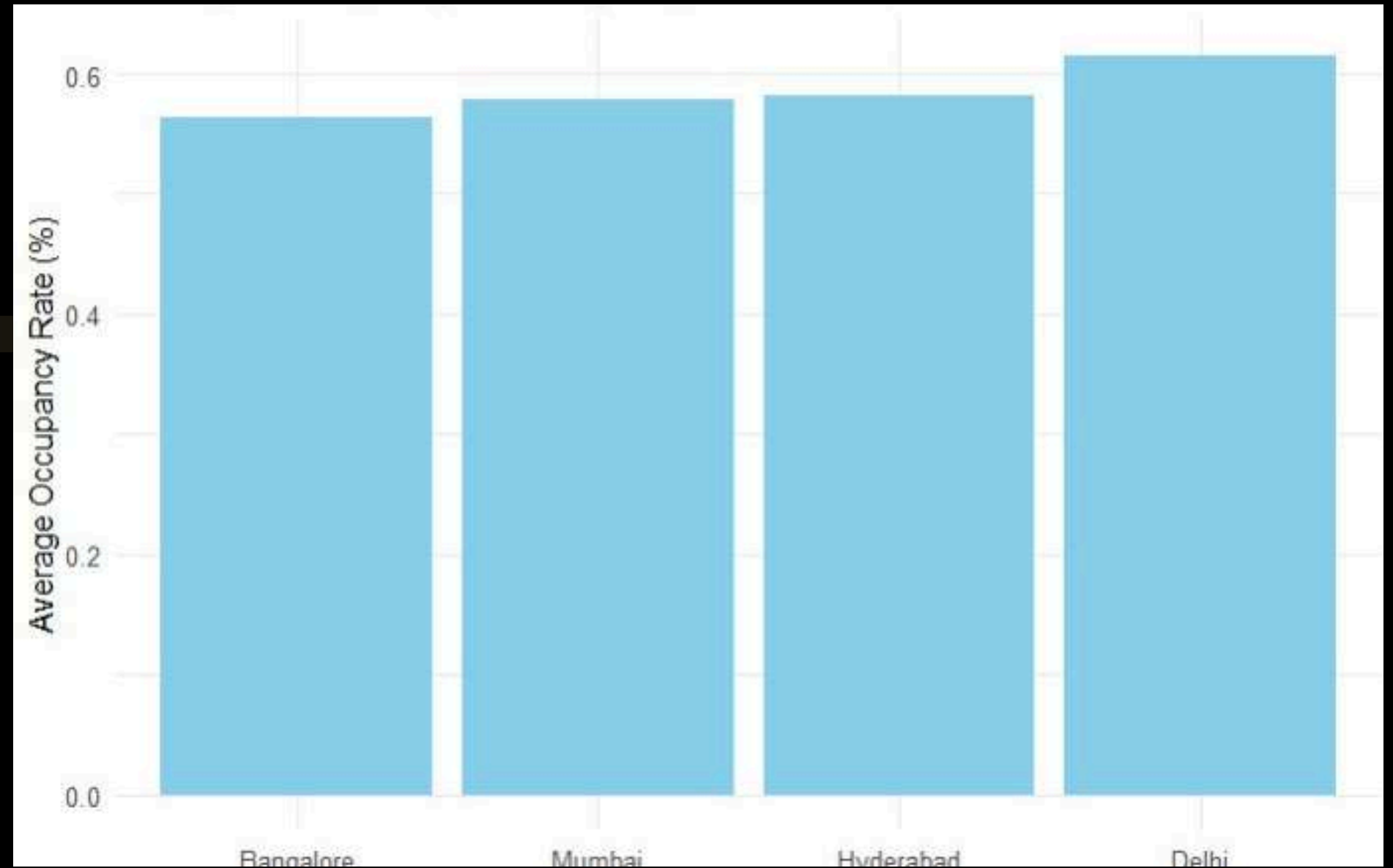
The histogram shows a bell-shaped curve, indicating a normal distribution. This means that most occupancy rates are clustered around the center (around 0.50), with fewer observations at the extremes.

Q2. AVERAGE OCCUPANCY RATE PER CITY

```
city_occupancy <- hotel_data %>%  
  group_by(city) %>%  
  summarize(  
    avg_occupancy = mean(occupancy, na.rm = TRUE),  
    sd_occupancy = sd(occupancy, na.rm = TRUE),  
    count = n()  
  ) %>%  
  arrange(desc(avg_occupancy))
```

Key Observations:

- Delhi has the highest average occupancy rate, exceeding 60%.
- Bangalore has the lowest average occupancy rate, around 55%.
- Mumbai and Hyderabad have similar average occupancy rates, both around 60%.

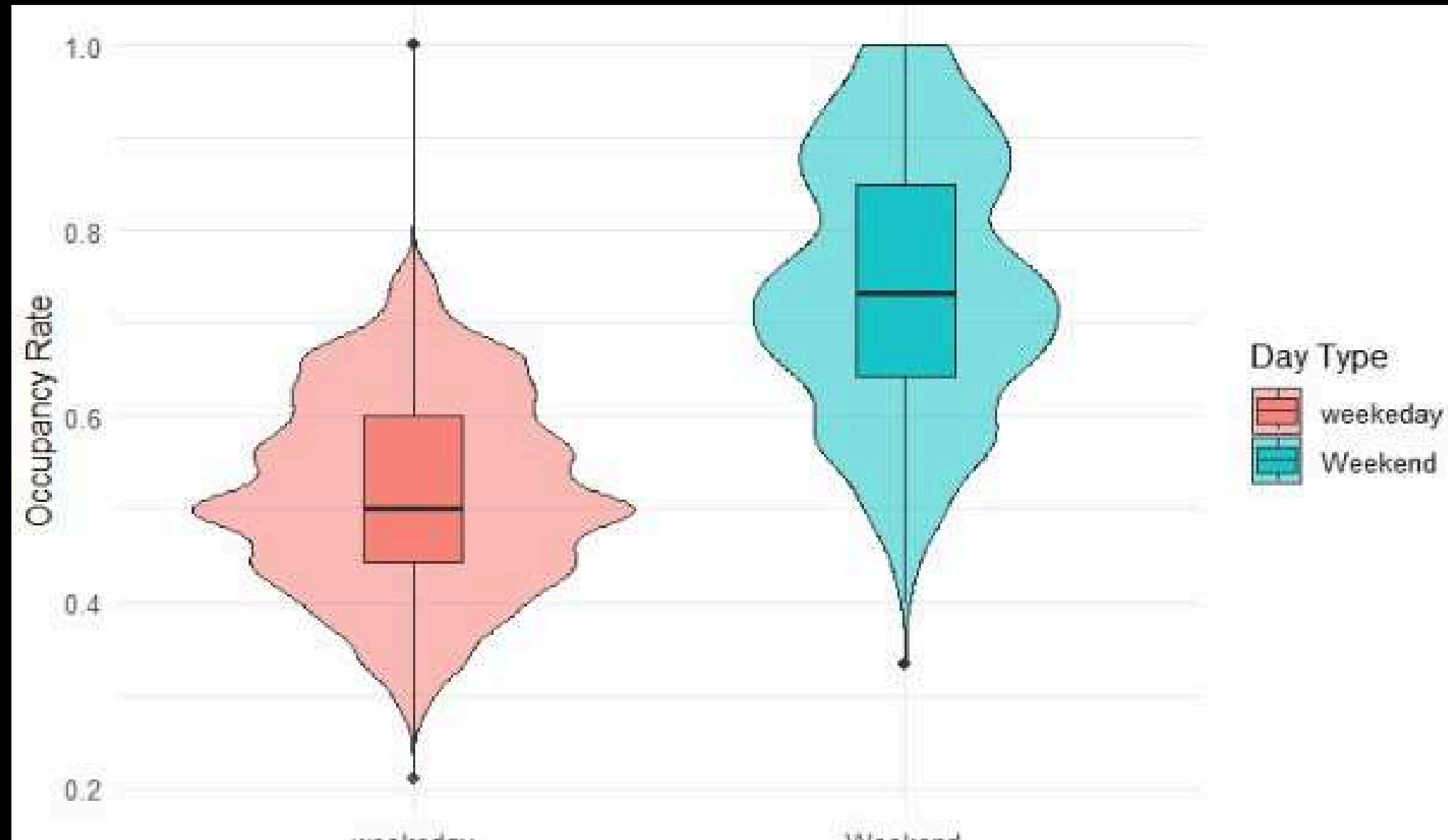


Q3. OCCUPANCY VARIATION BETWEEN WEEKDAYS AND WEEKENDS

```
OCCUPANCY_SUMMARY <- HOTEL_DATA %>%  
  GROUP_BY(DAY_TYPE) %>%  
  SUMMARIZE(  
    AVG_OCCUPANCY = MEAN(OCCUPANCY,  
      NA.RM = TRUE),  
    SD_OCCUPANCY = SD(OCCUPANCY, NA.RM  
      = TRUE),  
    MEDIAN_OCCUPANCY =  
      MEDIAN(OCCUPANCY, NA.RM = TRUE),  
    COUNT = N())
```

INTERPRETATION:

- OVERALL: THE GRAPH SUGGESTS THAT OCCUPANCY RATES ARE GENERALLY HIGHER ON WEEKENDS COMPARED TO WEEKDAYS.
- WEEKDAYS: THE BOX PLOT FOR WEEKDAYS SHOWS A LOWER MEDIAN OCCUPANCY RATE COMPARED TO WEEKENDS. THE DISTRIBUTION IS ALSO MORE SPREAD OUT WITH A WIDER RANGE OF VALUES.
- WEEKENDS: THE BOX PLOT FOR WEEKENDS SHOWS A HIGHER MEDIAN OCCUPANCY RATE AND A NARROWER RANGE OF VALUES, INDICATING A MORE CONCENTRATED DISTRIBUTION AROUND THE MEDIAN.

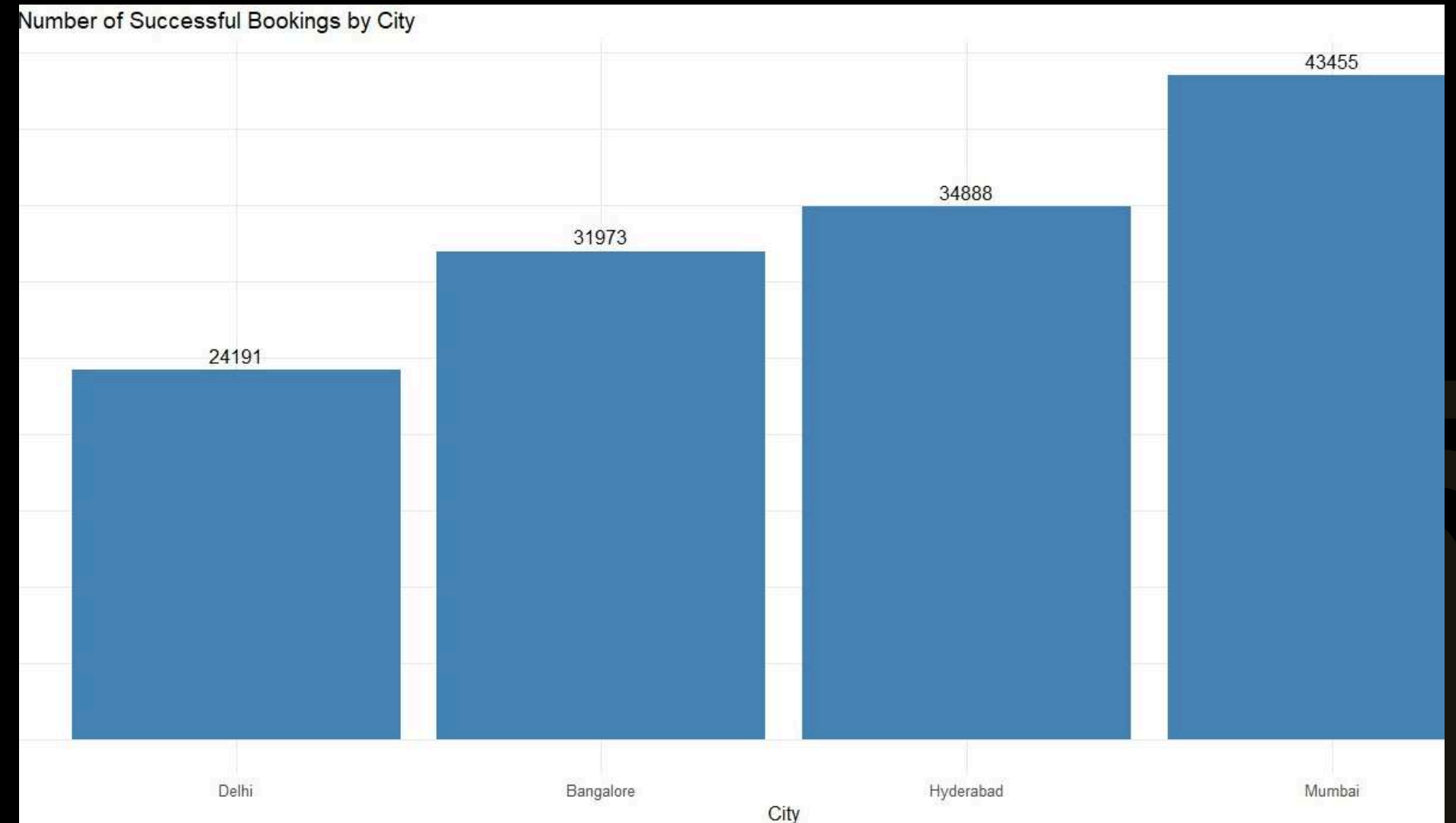


Q4. DISTRIBUTION OF SUCCESSFUL BOOKINGS ACROSS CITIES

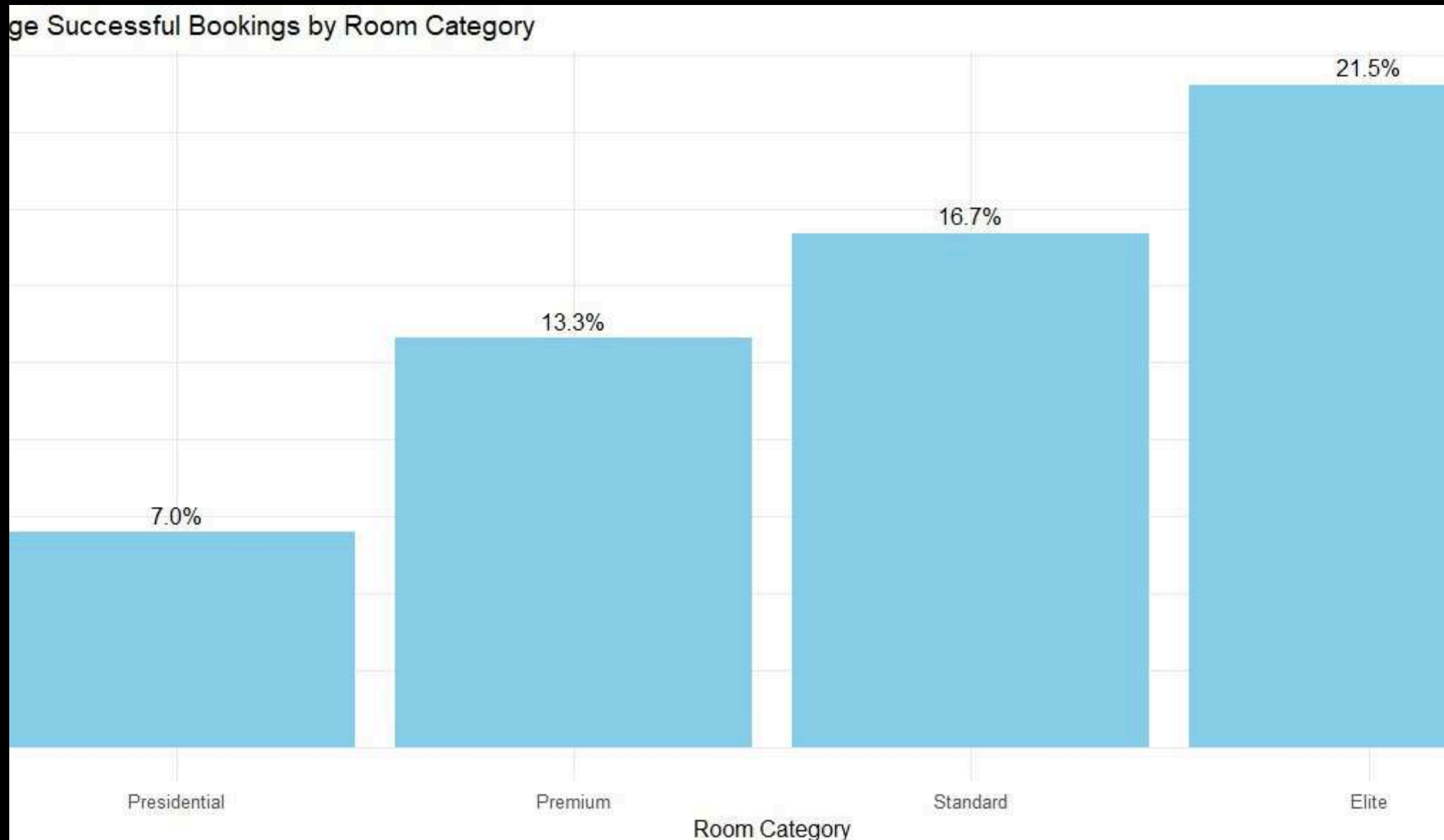
```
booking_stats <- hotel_data %>%  
  group_by(city) %>%  
  summarize(  
    total_bookings = n(),  
    successful_bookings = sum(successful_bookings),  
    success_rate = (successful_bookings / total_bookings) * 100  
  ) %>%  
  arrange(desc(successful_bookings))
```

Key Observations:

- Mumbai has the highest number of successful bookings, with approximately 43,455 bookings.
- Hyderabad comes in second with around 34,888 bookings.
- Bangalore has the third-highest number of bookings, with about 31,973.
- Delhi has the lowest number of successful bookings among the cities shown, with approximately 24,191 bookings.



Q5. ROOM CATEGORY WITH HIGHEST AVERAGE SUCCESSFUL BOOKINGS



```
room_stats <- hotel_data %>%  
  group_by(room_category) %>%  
  summarize(  
    total_bookings = n(),  
    avg_bookings = mean(successful_bookings, na.rm = TRUE)  
  ) %>%  
  arrange(desc(avg_bookings))
```

Key Observations:

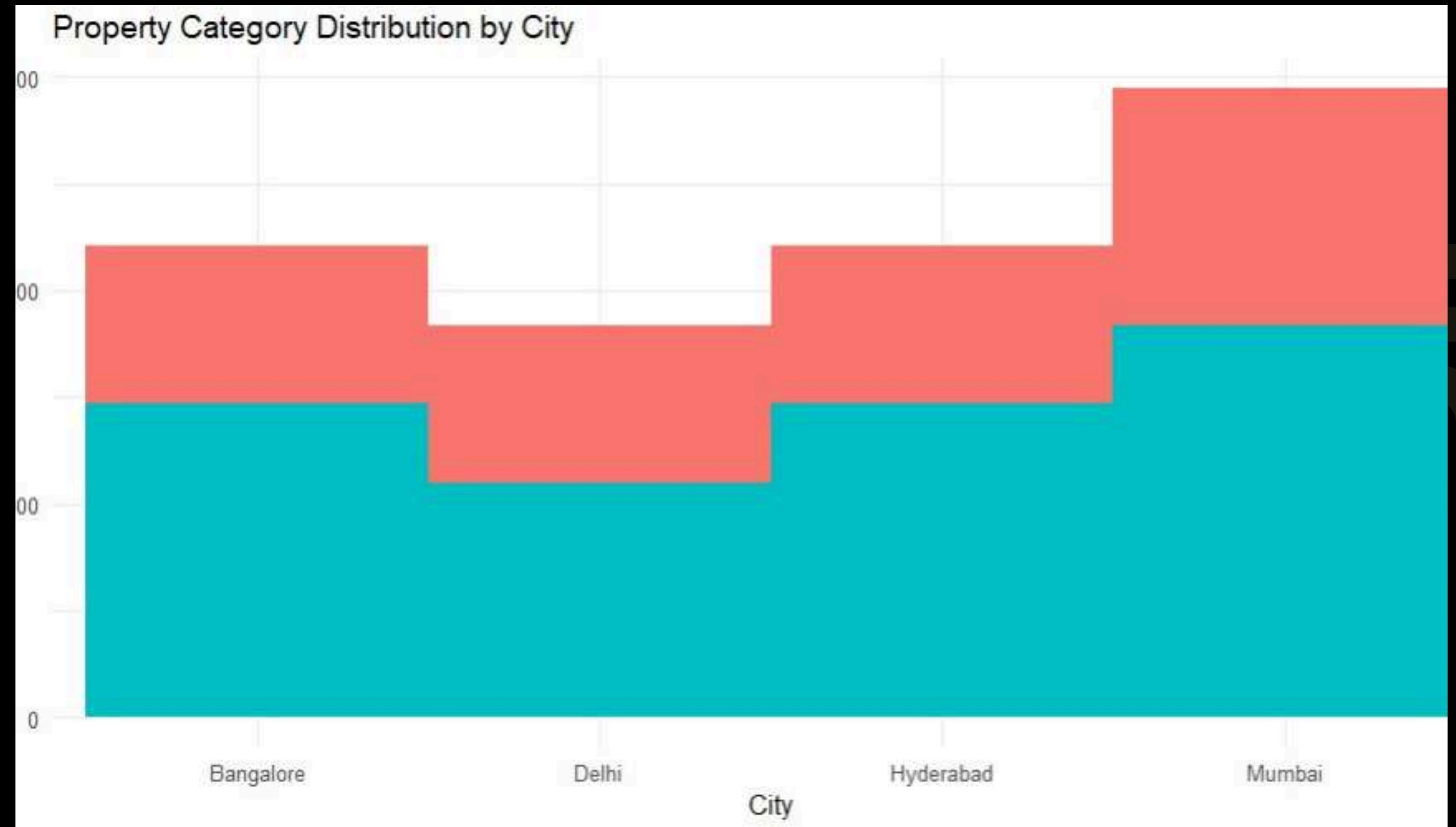
- Elite rooms have the highest average successful booking rate at 21.5%.
- Standard rooms have the second-highest average successful booking rate at 16.7%.
- Premium rooms have an average successful booking rate of 13.3%.
- Presidential rooms have the lowest average successful booking rate at 7.0%.

Q6. PROPERTY CATEGORY DISTRIBUTION DIFFER ACROSS CITIES

```
hproperty_distribution <- hotel_data %>%  
  group_by(city, category) %>%  
  summarize(count = n()) %>%  
  mutate(percentage = count / sum(count) * 100) %>%  
  arrange(category, city, desc(count))
```

Key Observations:

- Mumbai has the highest total number of properties, with a significant portion being "Business" properties.
- Bangalore has the second-highest total number of properties, with a relatively balanced distribution between "Business" and "Luxury" properties.
- Delhi has a slightly higher number of "Business" properties compared to "Luxury" properties.
- Hyderabad has the lowest total number of properties among the cities shown, with a majority being "Business" properties.



Q7. WHICH CITY HAS THE MOST PROPERTIES LISTED

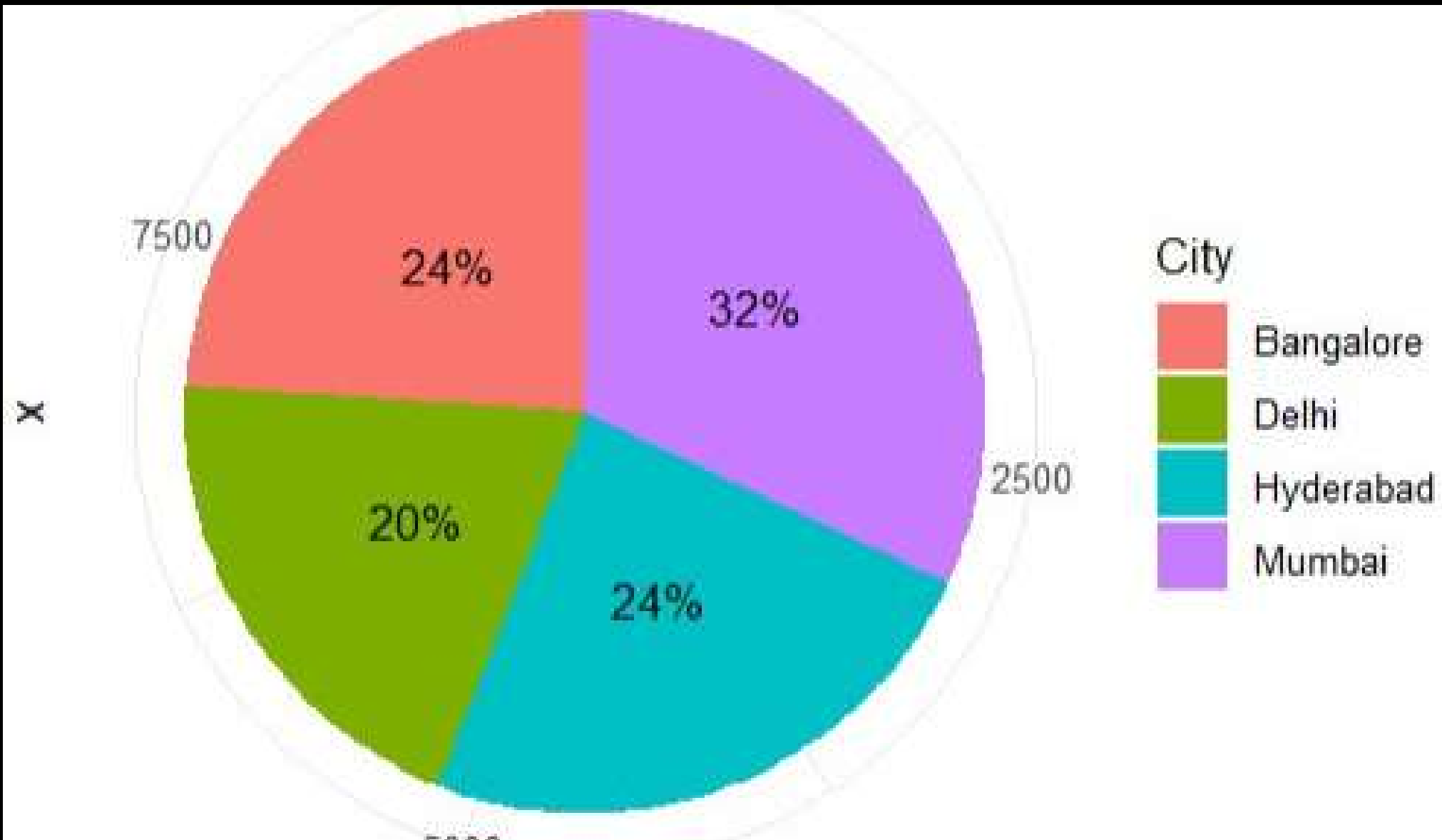
```
hcity_properties <- hotel_data %>%  
  group_by(city) %>%  
  summarize(  
    property_count = n(),  
    percentage = (n() / nrow(hotel_data) * 100)  
  ) %>%  
  arrange(desc(property_count))
```

Key Observations:

- Mumbai has the largest share of properties, accounting for 32% of the total.
- Bangalore and Hyderabad have similar shares, both at 24% each.
- Delhi has the smallest share, representing 20% of the total properties.

Overall:

The pie chart indicates that Mumbai has the highest concentration of properties, followed by Bangalore and Hyderabad, with Delhi having the lowest proportion.

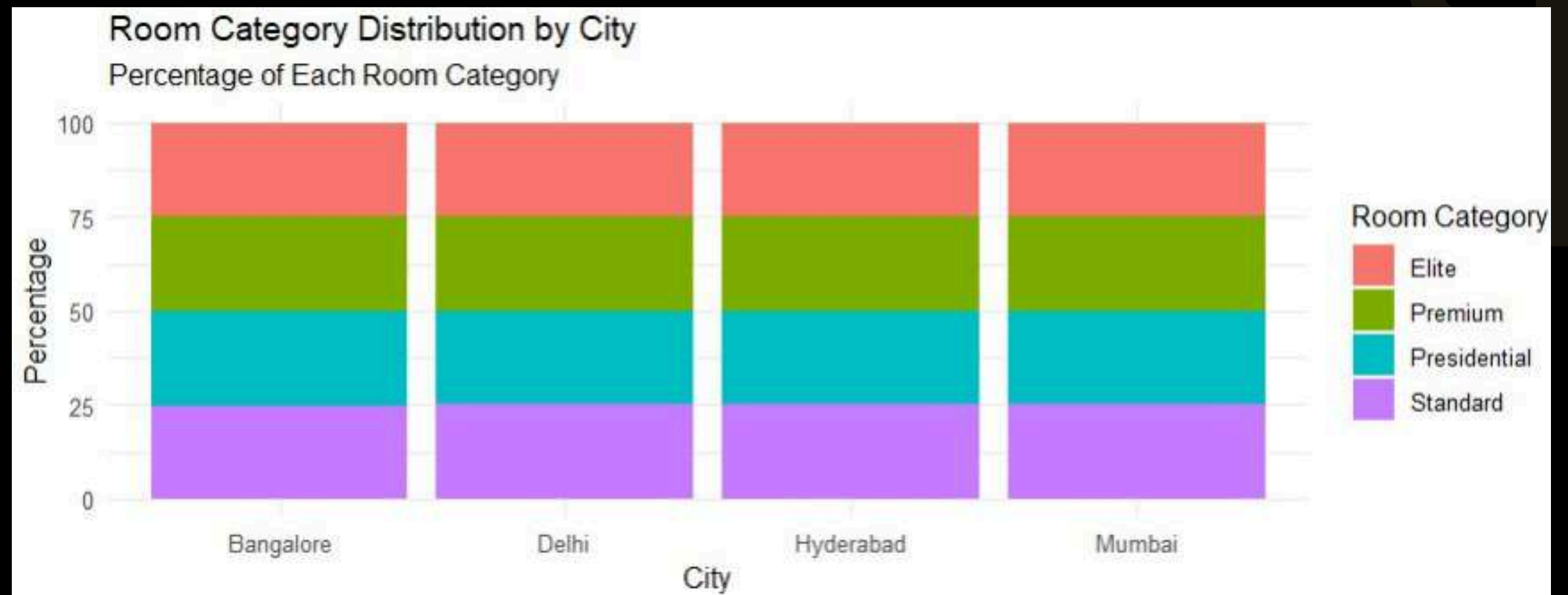


Q8. ROOM CATEGORIES MORE POPULAR IN SPECIFIC CITIES

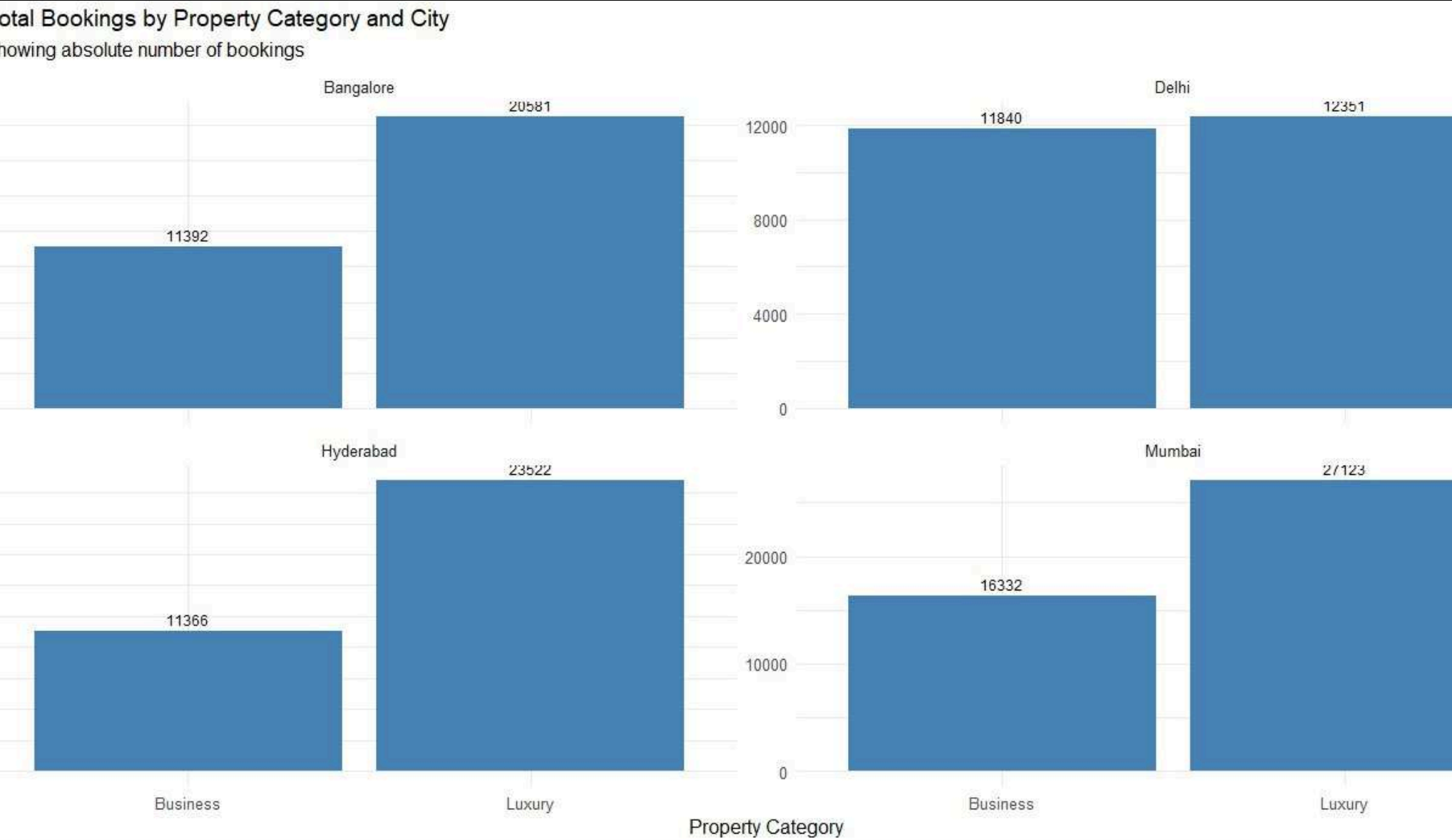
```
room_city_dist <- hotel_data %>%  
  group_by(city, room_category) %>%  
  summarize(  
    total_bookings = n(),  
  ) %>%  
  group_by(city) %>%  
  mutate(  
    percentage = (total_bookings /  
sum(total_bookings)) * 100,  
  )
```

Key Observations:

- Elite rooms have the highest percentage in all cities, indicating they are the most common room category.
- Standard rooms have the second-highest percentage in all cities, after Elite rooms.
- Premium and Presidential rooms have lower percentages compared to Elite and Standard rooms in all cities.
- The distribution of room categories is relatively similar across all cities, with Elite and Standard rooms dominating in each.



Q9. HIGHEST PROPERTY CONTRIBUTION IN EACH CITY



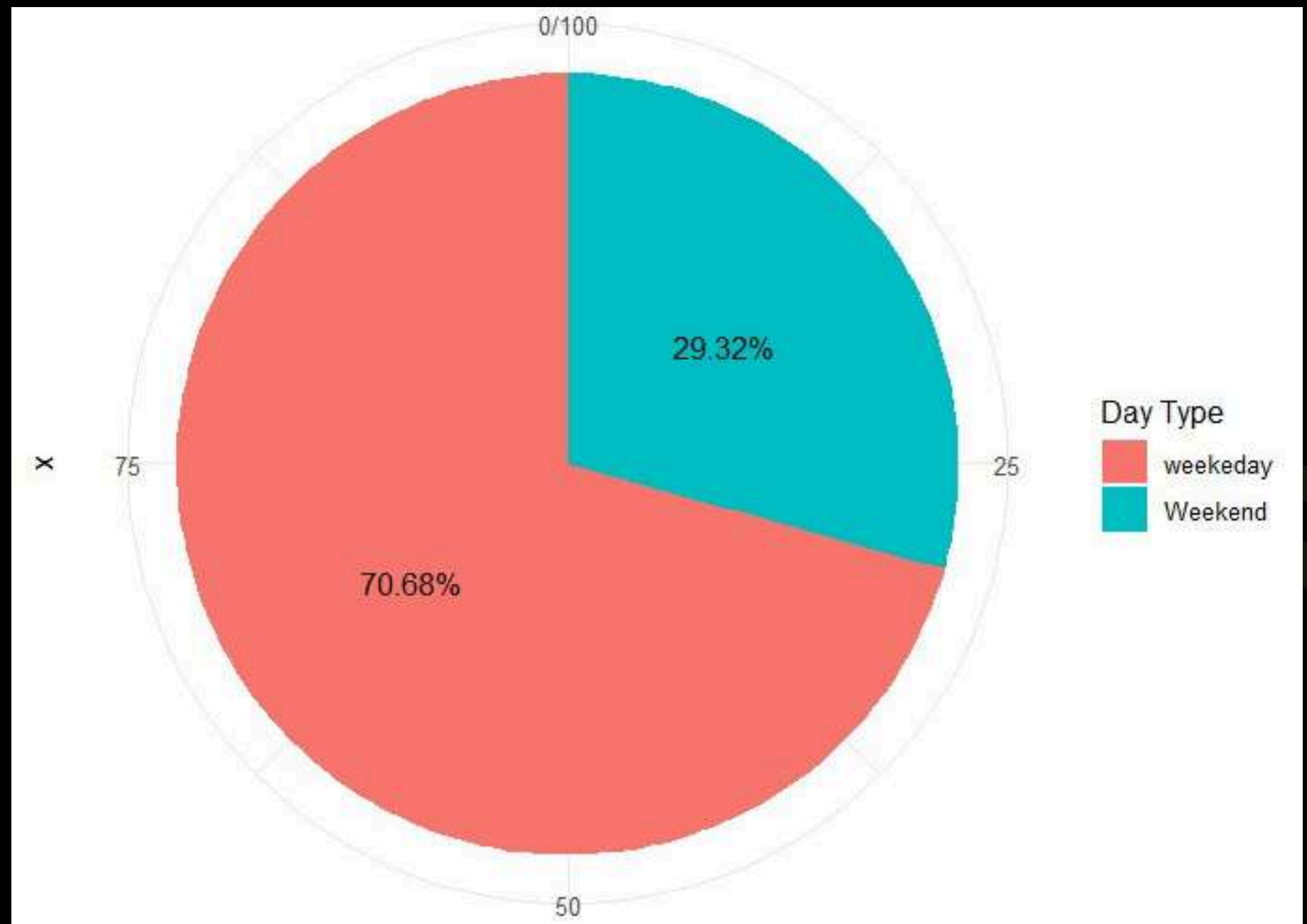
```
category_analysis <- hotel_data %>%
  group_by(city, category) %>%
  summarise(
    total_bookings = sum(successful_bookings),
    .groups = 'drop'
  ) %>%
  group_by(city) %>%
  mutate(
    booking_percentage = (total_bookings / sum(total_bookings) * 100),
    booking_percentage = round(booking_percentage, 2)
  ) %>%
  arrange(city, desc(total_bookings))
```

Q10. PROPORTION OF WEEKENDS TO WEEKDAYS

```
day_type_analysis <- hotel_data %>%  
  group_by(day_type) %>%  
  summarise(  
    count = n(),  
    .groups = 'drop'  
  ) %>%  
  mutate(  
    proportion = count / sum(count),  
    percentage = round(proportion * 100, 2)  
  )
```

Key Observations:

- Weekdays account for a significantly larger portion of the distribution, representing 70.68%.
- Weekends make up a smaller portion, representing 29.32%.



Q11. SIGNIFICANT DIFFERENCE IN OCCUPANCY RATES BETWEEN WEEKDAYS AND WEEKENDS

dPerform t-test

```
t_test_weekday_weekend <- t.test(  
  occupancy ~ day_type,  
  data = hotel_clean  
)  
print("T-Test: Occupancy Rates (Weekday vs Weekend)")  
print(t_test_weekday_weekend)
```






12. RELATIONSHIP BETWEEN ROOM CATEGORY AND SUCCESSFUL BOOKINGS (CHI-SQUARE TEST)

Create a contingency table

```
if ("room_category" %in% names(hotel_clean)
    && "successful_bookings" %in%
    names(hotel_clean)) {
  table_room_booking <- table(hotel_clean$room_category,
    hotel_clean$successful_bookings > 0)
```

Perform chi-square test

```
chi_square_test <- chisq.test(table_room_booking)
print("Chi-Square Test: Room Category vs Successful Bookings")
print(chi_square_test)
} else {
  print("Columns 'room_category' or 'successful_bookings' are missing!")
}
```

13. SUCCESSFUL BOOKINGS HIGHER DURING SPECIFIC MONTHS (T-TEST FOR SELECTED MONTHS)

```
if ("mm-yy" %in% names(hotel_clean) && "successful_bookings" %in% names(hotel_clean)) {  
  
  # Filter data for May and December  
  data_selected_months <- hotel_clean %>%  
    filter(format(as.Date(`mm-yy`, format="%m-%y"), "%m") %in% c("05", "12"))  
  
  # Create a month column  
  data_selected_months$month <- format(as.Date(data_selected_months$`mm-yy`,  
    format="%m-%y"), "%m")  
  
  # Check unique values in the 'month' column  
  print(unique(data_selected_months$month))  
  
  # If the filtering is working, there should be only May (05) and December (12)  
  if(length(unique(data_selected_months$month)) == 2) {  
  
    # Ensure month is treated as a factor with only two levels  
    data_selected_months$month <- factor(data_selected_months$month, levels = c("05", "12"))  
  
    # Perform t-test  
    t_test_months <- t.test(successful_bookings ~ month, data = data_selected_months)  
    print("T-Test: Successful Bookings (May vs December)")  
    print(t_test_months)  
  } else {  
    print("Error: More than two months found.")]}}
```




14. OCCUPANCY RATES OF PROPERTIES IN LUXURY AND BUSINESS CATEGORIES (T-TEST)

```
if ("category" %in% names(hotel_clean)) {  
  
  # Filter data for Luxury and Business categories  
  data_luxury_business <- hotel_clean %>% filter(category %in% c("Luxury",  
"Business"))  
  
  # Perform t-test  
  t_test_luxury_business <- t.test(  
    occupancy ~ category,  
    data = data_luxury_business  
  )  
  print("T-Test: Occupancy Rates (Luxury vs Business Categories)")  
  print(t_test_luxury_business)  
}  
}
```


15. WEEKEND VS WEEKDAY OCCUPANCY RATES ACROSS ALL CITIES (PAIRED T-TEST)

```
if ("city" %in% names(hotel_clean) && "day_type" %in% names(hotel_clean)) {  
  
  # Prepare data for paired t-test  
  paired_data <- hotel_clean %>%  
    group_by(city, day_type) %>%  
    summarize(mean_occupancy = mean(occupancy, na.rm = TRUE)) %>%  
    pivot_wider(names_from = day_type, values_from = mean_occupancy)  
  
  if ("Weekday" %in% names(paired_data) && "Weekend" %in% names(paired_data)) {  
  
    # Perform paired t-test  
    t_test_paired <- t.test(  
      paired_data$Weekday,  
      paired_data$Weekend,  
      paired = TRUE  
    )  
    print("Paired T-Test: Occupancy Rates (Weekend vs Weekday)")  
    print(t_test_paired)  
  } else {  
    print("Paired data is incomplete! Ensure both 'Weekday' and 'Weekend' columns are present.")  
  }  
}
```




16.

```
# ENSURE `DAY_TYPE` IS A FACTOR AND PREDICTORS ARE NUMERIC
HOTEL_DATA$DAY_TYPE <- AS.FACTOR(HOTEL_DATA$DAY_TYPE)
HOTEL_DATA$OCCUPANCY <- AS.NUMERIC(HOTEL_DATA$OCCUPANCY)
HOTEL_DATA$CAPACITY <- AS.NUMERIC(HOTEL_DATA$CAPACITY)

# CHECK FOR NA VALUES (OPTIONAL, BUT RECOMMENDED)
IF (ANYNA(HOTEL_DATA)) {
  PRINT("DATA CONTAINS NA VALUES. HANDLING THEM IS RECOMMENDED.")
}

# SPLIT DATA INTO TRAINING AND TESTING SETS
SET.SEED(123)
TRAIN_INDEX <- CREATEDATAPARTITION(HOTEL_DATA$DAY_TYPE, P = 0.8, LIST
= FALSE)
TRAIN_DATA <- HOTEL_DATA[TRAIN_INDEX, ]
TEST_DATA <- HOTEL_DATA[-TRAIN_INDEX, ]

# FIT A LOGISTIC REGRESSION MODEL
LOG_MODEL <- GLM(DAY_TYPE ~ OCCUPANCY + CAPACITY, DATA =
TRAIN_DATA, FAMILY = "BINOMIAL")

# SUMMARY OF THE MODEL
SUMMARY(LOG_MODEL)

# PREDICT ON THE TEST SET
TEST_DATA$PREDICTED <- PREDICT(LOG_MODEL, NEWDATA = TEST_DATA, TYPE
= "RESPONSE")
TEST_DATA$PREDICTED_CLASS <- IFELSE(TEST_DATA$PREDICTED > 0.5,
"WEEKEND", "WEEKDAY")

# CHECK UNIQUE LEVELS IN THE TEST SET
ACTUAL_LEVELS <- LEVELS(TEST_DATA$DAY_TYPE)
PREDICTED_LEVELS <- LEVELS(FACTOR(TEST_DATA$PREDICTED_CLASS))

# IF NECESSARY, ADJUST LEVELS OF PREDICTED CLASS TO MATCH THE ACTUAL
CLASS LEVELS
PREDICTED_FACTOR <- FACTOR(TEST_DATA$PREDICTED_CLASS, LEVELS =
ACTUAL_LEVELS)

# EVALUATE THE MODEL
CONFUSION_MATRIX <- CONFUSIONMATRIX(PREDICTED_FACTOR,
TEST_DATA$DAY_TYPE)
PRINT(CONFUSION_MATRIX)
```




16.

```
# ENSURE `DAY_TYPE` IS A FACTOR AND PREDICTORS ARE NUMERIC
HOTEL_DATA$DAY_TYPE <- AS.FACTOR(HOTEL_DATA$DAY_TYPE)
HOTEL_DATA$OCCUPANCY <- AS.NUMERIC(HOTEL_DATA$OCCUPANCY)
HOTEL_DATA$CAPACITY <- AS.NUMERIC(HOTEL_DATA$CAPACITY)

# CHECK FOR NA VALUES (OPTIONAL, BUT RECOMMENDED)
IF (ANYNA(HOTEL_DATA)) {
  PRINT("DATA CONTAINS NA VALUES. HANDLING THEM IS RECOMMENDED.")
}

# SPLIT DATA INTO TRAINING AND TESTING SETS
SET.SEED(123)
TRAIN_INDEX <- CREATEDATAPARTITION(HOTEL_DATA$DAY_TYPE, P = 0.8, LIST
= FALSE)
TRAIN_DATA <- HOTEL_DATA[TRAIN_INDEX, ]
TEST_DATA <- HOTEL_DATA[-TRAIN_INDEX, ]

# FIT A LOGISTIC REGRESSION MODEL
LOG_MODEL <- GLM(DAY_TYPE ~ OCCUPANCY + CAPACITY, DATA =
TRAIN_DATA, FAMILY = "BINOMIAL")

# SUMMARY OF THE MODEL
SUMMARY(LOG_MODEL)

# PREDICT ON THE TEST SET
TEST_DATA$PREDICTED <- PREDICT(LOG_MODEL, NEWDATA = TEST_DATA, TYPE
= "RESPONSE")
TEST_DATA$PREDICTED_CLASS <- IFELSE(TEST_DATA$PREDICTED > 0.5,
"WEEKEND", "WEEKDAY")

# CHECK UNIQUE LEVELS IN THE TEST SET
ACTUAL_LEVELS <- LEVELS(TEST_DATA$DAY_TYPE)
PREDICTED_LEVELS <- LEVELS(FACTOR(TEST_DATA$PREDICTED_CLASS))

# IF NECESSARY, ADJUST LEVELS OF PREDICTED CLASS TO MATCH THE ACTUAL
CLASS LEVELS
PREDICTED_FACTOR <- FACTOR(TEST_DATA$PREDICTED_CLASS, LEVELS =
ACTUAL_LEVELS)

# EVALUATE THE MODEL
CONFUSION_MATRIX <- CONFUSIONMATRIX(PREDICTED_FACTOR,
TEST_DATA$DAY_TYPE)
PRINT(CONFUSION_MATRIX)
```


16.

```
# ENSURE `DAY_TYPE` IS A FACTOR AND  
PREDICTORS ARE NUMERIC  
HOTEL_DATA$DAY_TYPE <-  
AS.FACTOR(HOTEL_DATA$DAY_TYPE)  
HOTEL_DATA$OCCUPANCY <-  
AS.NUMERIC(HOTEL_DATA$OCCUPANCY)  
HOTEL_DATA$CAPACITY <-  
AS.NUMERIC(HOTEL_DATA$CAPACITY)
```

```
# CHECK FOR NA VALUES (OPTIONAL, BUT  
RECOMMENDED)  
IF (ANYNA(HOTEL_DATA)) {  
  PRINT("DATA CONTAINS NA VALUES. HANDLING  
  THEM IS RECOMMENDED.")  
}
```

```
# SPLIT DATA INTO TRAINING AND TESTING SETS  
SET.SEED(123)  
TRAIN_INDEX <-  
CREATEDATAPARTITION(HOTEL_DATA$DAY_TYPE, P  
= 0.8, LIST = FALSE)  
TRAIN_DATA <- HOTEL_DATA[TRAIN_INDEX, ]  
TEST_DATA <- HOTEL_DATA[-TRAIN_INDEX, ]
```

```
# FIT A LOGISTIC REGRESSION MODEL  
LOG_MODEL <- GLM(DAY_TYPE ~ OCCUPANCY +  
CAPACITY, DATA = TRAIN_DATA, FAMILY =  
"BINOMIAL")
```

```
# SUMMARY OF THE MODEL  
SUMMARY(LOG_MODEL)
```




```
# Predict on the test set
test_data$predicted <- predict(log_model,
newdata = test_data, type = "response")
test_data$predicted_class <-
ifelse(test_data$predicted > 0.5,
"Weekend", "Weekday")

# Check unique levels in the test set
actual_levels <-
levels(test_data$day_type)
predicted_levels <-
levels(factor(test_data$predicted_class))

# If necessary, adjust levels of predicted
class to match the actual class levels
predicted_factor <-
factor(test_data$predicted_class, levels =
actual_levels)

# Evaluate the model
confusion_matrix <-
confusionMatrix(predicted_factor,
test_data$day_type)
print(confusion_matrix)

}
```


PREDICTIVE


MODELLING




```
CONVERT OCCUPANCY TO BINARY: 1 IF > 75%, ELSE 0
DATA <- DATA %>%
  MUTATE(OCCUPANCY_HIGH = IFELSE(OCCUPANCY > 0.75, 1, 0))

# CALCULATE MEDIAN BOOKINGS AND CLASSIFY PERFORMANCE
MEDIAN_BOOKINGS <- MEDIAN(DATA$SUCCESSFUL_BOOKINGS, NA.RM =
TRUE)
DATA <- DATA %>%
  MUTATE(PERFORMANCE = IFELSE(SUCCESSFUL_BOOKINGS >
MEDIAN_BOOKINGS, "HIGH", "LOW"))

# ONE-HOT ENCODE CATEGORICAL VARIABLES
DATA <- DATA %>%
  SELECT(CHECK_IN_DATE, MM_YY) %>% # REMOVE DATE COLUMNS IF NOT
NEEDED FOR MODELING
  MUTATE_IF(IS.CHARACTER, AS.FACTOR) %>% # CONVERT CHARACTER
COLUMNS TO FACTORS
  MUTATE_IF(IS.FACTOR, AS.NUMERIC) # ONE-HOT ENCODE FACTORS
```



```
SPLIT DATA INTO TRAINING AND TEST SETS
SET.SEED(123)
TRAIN_INDEX <- CREATEDATAPARTITION(DATA$OCCUPANCY_HIGH, P
= 0.8, LIST = FALSE)
TRAIN_DATA <- DATA[TRAIN_INDEX,]
TEST_DATA <- DATA[TRAIN_INDEX,]

# SPLIT DATA INTO TRAINING AND TEST SETS
SET.SEED(123)
TRAIN_INDEX <- CREATEDATAPARTITION(DATA$OCCUPANCY_HIGH, P
= 0.8, LIST = FALSE)
TRAIN_DATA <- DATA[TRAIN_INDEX,]
TEST_DATA <- HOTEL_DATA[-TRAIN_INDEX,]

# TRAIN A LOGISTIC REGRESSION MODEL FOR OCCUPANCY
OCCUPANCY_MODEL <- TRAIN(OCCUPANCY_HIGH ~ ., DATA =
TRAIN_DATA, METHOD = "GLM", FAMILY = "BINOMIAL")

# PREDICT ON TEST DATA AND EVALUATE
OCCUPANCY_PREDICTIONS <- PREDICT(OCCUPANCY_MODEL, NEWDATA
= TEST_DATA)
CONFUSIONMATRIX(OCCUPANCY_PREDICTIONS,
TEST_DATA$OCCUPANCY_HIGH)
```



```
TRAIN A CLASSIFICATION MODEL FOR PERFORMANCE
PERFORMANCE_MODEL <- TRAIN(PERFORMANCE ~ ., DATA =
TRAIN_DATA, METHOD = "RPART")

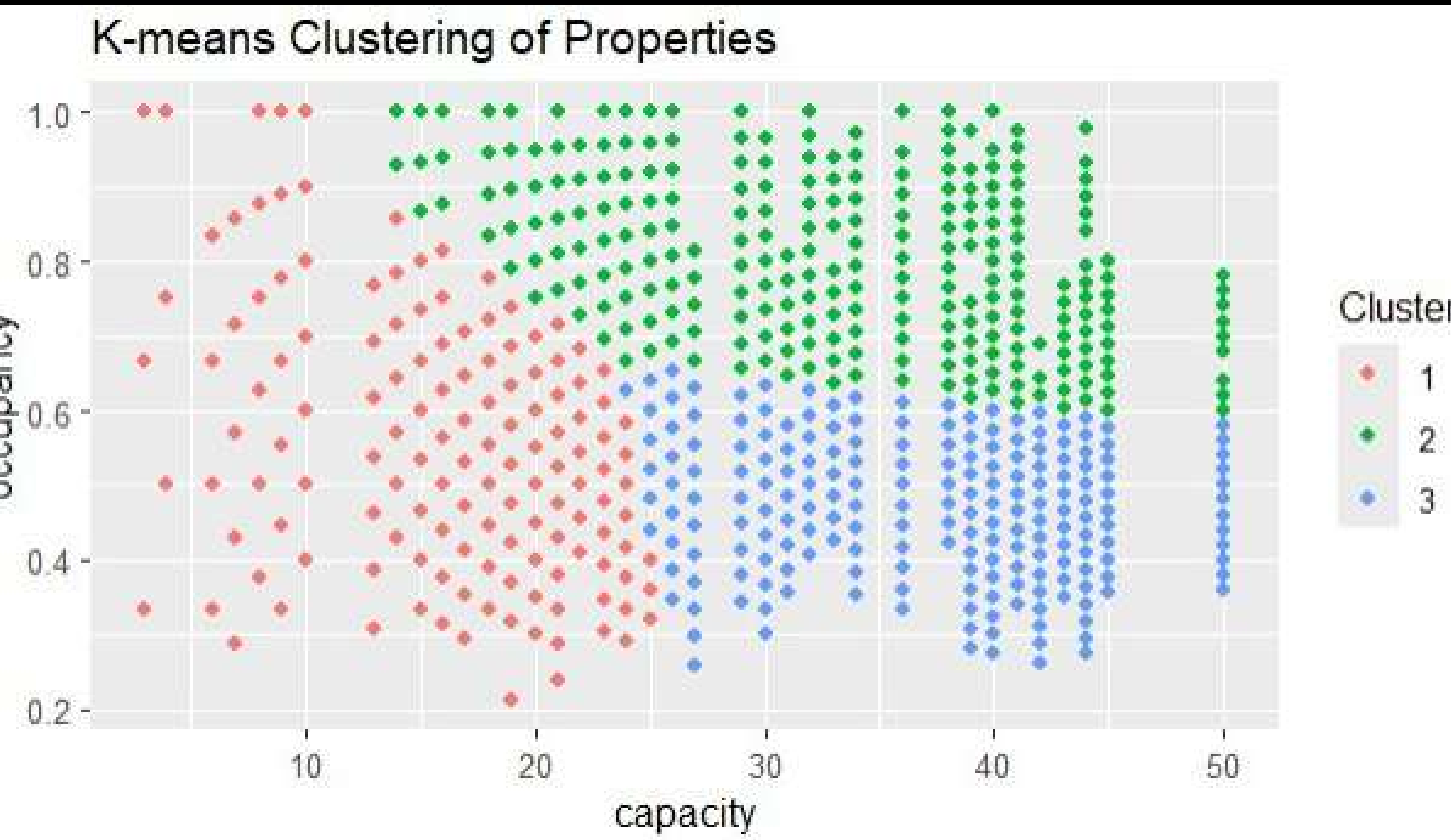
# PREDICT ON TEST DATA AND EVALUATE
PERFORMANCE_PREDICTIONS <- PREDICT(PERFORMANCE_MODEL,
NEWDATA = TEST_DATA)
CONFUSIONMATRIX(PERFORMANCE_PREDICTIONS,
TEST_DATA$PERFORMANCE)

# CLUSTERING WITH K-MEANS
# SELECT NUMERIC COLUMNS FOR CLUSTERING
CLUSTERING_DATA <- DATA %>% SELECT(CAPACITY,
SUCCESSFUL_BOOKINGS, OCCUPANCY)

# NORMALIZE THE DATA
CLUSTERING_DATA <- SCALE(CLUSTERING_DATA)
```



CLUSTERING WITH K-MEANS



```
# Select numeric columns for clustering
clustering_data <- hotel_data %>% select(capacity,
successful_bookings, occupancy)
```

```
# Normalize the data
clustering_data <- scale(clustering_data)
```

```
# Perform K-means clustering
set.seed(123)
kmeans_result <- kmeans(clustering_data, centers = 3, nstart = 25)
```

```
# Add cluster results to the original data
data$cluster <- kmeans_result$cluster
```

```
# Visualize clusters
ggplot(data, aes(x = capacity, y = occupancy, color =
as.factor(cluster))) +
  geom_point() +
  labs(title = "K-means Clustering of Properties", color = "Cluster")
```


THANK YOU

FOR ATTENTION

