ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT127-3-2-PFDA

PROGRAMMING FOR DATA ANALYSIS

APD2F2209CS(DA)

HAND OUT DATE:	**10 OTCTOBER 2022**
HAND IN DATE:	**2 DECEMBER 2022**
WEIGHTAGE:	**50%**
NAME:	**KOK HON KIT**
TP NO:	**TP059378**

INSTRUCTIONS TO CANDIDATES:

1 **Students are advised to underpin their answers with the use of references (cited using the American Psychological Association (APA) Referencing).**

2 **Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.**

3 **Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hardcopy.**

4 **You must obtain 50% overall to pass this module.**

Table of Contents

1.0 Introduction

Human Rights Measurement team has assigned task to study data problems given a dataset about rental house postings in India. To achieve to goal of investigating useful insights from the data, data analytics lifecycle such as data importing, data exploration, data cleaning and pre-processing, data transformation, and data visualizations are carried out in sequential manner. Techniques and concepts implemented in the project will be discussed in the report as well as the justifications for why the technique is used. In data importing phase, the dataset in CSV format will be read and parsed into usable form such as data frame data type in R. for data exploration phase, the distribution of data will be explored to identify the analysis techniques to be used in later phases. Transitioning into data cleaning phase, the data will be cleaned by removing outliers and filling in missing values. Afterward, additional columns will be computed to carry out the analysis in more detailed and in-depth manner in data transformation phase. In data visualization phase, data is used to plot various charts such as bar chart, pie chart, and scatter plot to visualize relationships between variables and extract insights for effective decision making. Throughout the phases, R programming language will be implemented to execute all required tasks. Upon completing analysis, all processes will be documented in this report as outlined in the Table of Contents.

2.0 Data Exploration

2.1 Environment Set Up and Data Importing

```
install.packages("tidyverse")
install.packages("dplyr")
install.packages("lubridate")
install.packages("ggplot2")
install.packages("scales")
install.packages("plotly")
install.packages("corrgram")
install.packages("viridis")  # Install
install.packages("corrplot")
install.packages("DescTools")
install.packages("stringr")
install.packages("smooth")
install.packages("ggpubr")
install.packages("treemapify")

# importing required packages
library(tidyverse) # data cleaning
library(dplyr) # data cleaning
library(lubridate) # data transformation (date manipulation and grouping)
library(ggplot2) # data visualization
library(scales) # data formatting in plots
library(RColorBrewer) # for colour palette
library(plotly) # data visualization library
library(corrgram) # for correlation plot
library(viridis) # colour palette for visualization
library(corrplot) # correlation plot
library(DescTools)
library(stringr) # data manipulation
library(smooth)
library(zoo) # moving average
library(ggpubr)
library(treemapify)
```

Figure 2-1: Environment Set Up

Before starting any data analytics process, all the required packages such as dplyr, ggplot2, and plotly must be installed and loaded into R environment to use its functions to perform necessary actions.

```
> df <- read.csv(file="C:\\Users\\bpvpe\\OneDrive - Asia Pacific University\\Degree\\Sem 1\\Programming For Data Analysis\\Assignment\\House_Rent_Dataset.csv")
```

Figure 2-2: Data Importing

The code above parse the data in Comma Separated Values (CSV) file given into data frame data type in R environment for data analysis. The parsed data frame is stored into variable “df” which stands for data frame which will be reused for the entire data analysis process.

2.2 Data Understanding

	Posted.On	BHK	Rent	Size	Floor	Area.Type	Area.Locality	City	Furnishing.Status	Tenant.Preferred	Bathroom	Point.of.Contact
1	5/18/2022	2	10000	1100	Ground out of 2	Super Area	Bandel	Kolkata	Unfurnished	Bachelors/Family	2	Contact Owner
2	5/13/2022	2	20000	800	1 out of 3	Super Area	Phool Bagan, Kankurgachi	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
3	5/16/2022	2	17000	1000	1 out of 3	Super Area	Salt Lake City Sector 2	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
4	7/4/2022	2	10000	800	1 out of 2	Super Area	Dumdum Park	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
5	5/9/2022	2	7500	850	1 out of 2	Carpet Area	South Dum Dum	Kolkata	Unfurnished	Bachelors	1	Contact Owner

Figure 2-3: Dataset Given

The image above illustrates the view of the dataset given to analyse the questions in data analysis phase.

Column	Description
Posted.On	The date when the rental house is available for renting and posted to the public.
BHK	The acronym stands for bedroom, hall, and kitchen. It represents the number of those facilities in the dataset.
Rent	The total rent price of the house.
Size	The size of the house unit.
Floor	The description about the current floor of the unit and total floor the house has represented in text format.
Area.Type	The method that the house size is calculated.
Area.Locality	The location where the rental house is located including the street and poscode.
City	The city where the rental house is located.
Furnishing.Status	The furnishing status of the rental house which includes “Unfurnished”, “Furnished”, and “Semi-Furnished”.
Tenant.Preferred	The tenants preferred to rent the house to. Tenants include “Bachelors”, “Family” and “Bachelors/Family”.
Bathroom	The number of bathrooms provided by the house.
Point.of.Contact	The person to contact when the tenants are interested in the rental house. Point of contact includes “Agent”, “Owner”, and “Builder”.

Table 2-1: Basic Data Description

2.3 Exploratory Data Analysis

```
> nrow(df) # 4767 rows
[1] 4746
>
> # number of columns for the dataset
> ncol(df) # 12 columns
[1] 12
>
> # dimensions of dataset
> dim(df) # 4767, 12 (row, column)
[1] 4746   12
>
> # all columns in the dataset
> names(df)
 [1] "Posted.On"         "BHK"               "Rent"              "Size"              "Floor"
 [6] "Area.Type"         "Area.Locality"     "City"              "Furnishing.Status" "Tenant.Preferred"
[11] "Bathroom"          "Point.of.Contact"
>
> # top 5 rows for the dataset
> head(df, 5)
  Posted.On BHK  Rent Size           Floor   Area.Type            Area.Locality    City Furnishing.Status
1 5/18/2022   2 10000 1100 Ground out of 2  Super Area                   Bandel Kolkata       Unfurnished
2 5/13/2022   2 20000  800      1 out of 3  Super Area Phool Bagan, Kankurgachi Kolkata    Semi-Furnished
3 5/16/2022   2 17000 1000      1 out of 3  Super Area  Salt Lake City Sector 2 Kolkata    Semi-Furnished
4  7/4/2022   2 10000  800      1 out of 2  Super Area              Dumdum Park Kolkata       Unfurnished
5  5/9/2022   2  7500  850      1 out of 2 Carpet Area            South Dum Dum Kolkata       Unfurnished
  Tenant.Preferred Bathroom Point.of.Contact
1 Bachelors/Family        2    Contact Owner
2 Bachelors/Family        1    Contact Owner
3 Bachelors/Family        1    Contact Owner
4 Bachelors/Family        1    Contact Owner
5        Bachelors        1    Contact Owner
>
```

Figure 2-4: Basic Data Exploration

Using built-in functions provided by R, basic data exploration such as getting dimension of data and top 5 rows in the dataset are conducted to understand the dataset in high level manner. From the image above, the dataset has a total of 4767 rows and 12 columns that contains various types of data including categorical and numeric variables.

```
> summary(df)
  Posted.On              BHK             Rent               Size            Floor           Area.Type
 Length:4746        Min.   :1.000   Min.   :   1200   Min.   :  10.0   Length:4746        Length:4746
 Class :character   1st Qu.:2.000   1st Qu.:  10000   1st Qu.: 550.0   Class :character   Class :character
 Mode  :character   Median :2.000   Median :  16000   Median : 850.0   Mode  :character   Mode  :character
                    Mean   :2.084   Mean   :  34993   Mean   : 967.5
                    3rd Qu.:3.000   3rd Qu.:  33000   3rd Qu.:1200.0
                    Max.   :6.000   Max.   :3500000   Max.   :8000.0
 Area.Locality          City           Furnishing.Status  Tenant.Preferred      Bathroom      Point.of.Contact
 Length:4746        Length:4746        Length:4746        Length:4746        Min.   : 1.000   Length:4746
 Class :character   Class :character   Class :character   Class :character   1st Qu.: 1.000   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Median : 2.000   Mode  :character
                                                                             Mean   : 1.966
                                                                             3rd Qu.: 2.000
                                                                             Max.   :10.000
>
```

Figure 2-5: Statistical Summary of the Dataset

R language provides “summary()” function to calculate statistical summary of the dataset which consists of minimum, maximum, first quartile range, median, mean, and third quartile range values to understand the datasets better. The images above shows the summary for all the variables available in the dataset. For example, the minimum value for Rent variable is

1200 while the maximum value is 35000000. The maximum value for Rent variable is clearly an outlier as the value stays extremely far away from the median and mean values. Therefore, the data cleaning and pre-processing are required for this dataset to make the analysis more precise and meaningful.

```
33  # unique categories of categorical variables
34  unique(df$Floor) # 480 unique categories
35  unique(df$Area.Type) # 3 unique categories
36  unique(df$Area.Locality) # 2235 unique categories
37  unique(df$City) # 6 unique categories
38  unique(df$Furnishing.Status) # 3 unique categories
39  unique(df$Tenant.Preferred) # 3 unique categories
40  unique(df$Point.of.Contact) # 3 unique categories
41
```

Figure 2-6: Get Unique Values of Categorical Variables

```
> unique(df$Area.Type)
[1] "Super Area"  "Carpet Area" "Built Area"
> length(unique(df$Area.Locality))
[1] 2235
>
```

Figure 2-7: Result of "unique()" Built-in Function

For categorical variables, “unique()” function is used to get the distinct values of the categorical class so that the analyst can plan the analysis better. With the usage of “length()” function, the number of unique categorical value is returned on the console as illustrated in the images above. Using the “length()” and unique()” functions, the number of distinct values for each categorical variable can be concluded as following table:

Variable	Number of Distinct Values
Floor	480
Area.Type	3
Area.Locality	2235
City	6
Furnishing.Status	3
Tenant.Preferred	3
Point.of.Contact	3

Table 2-2: Number of Distinct Values for Each Categorical Variable

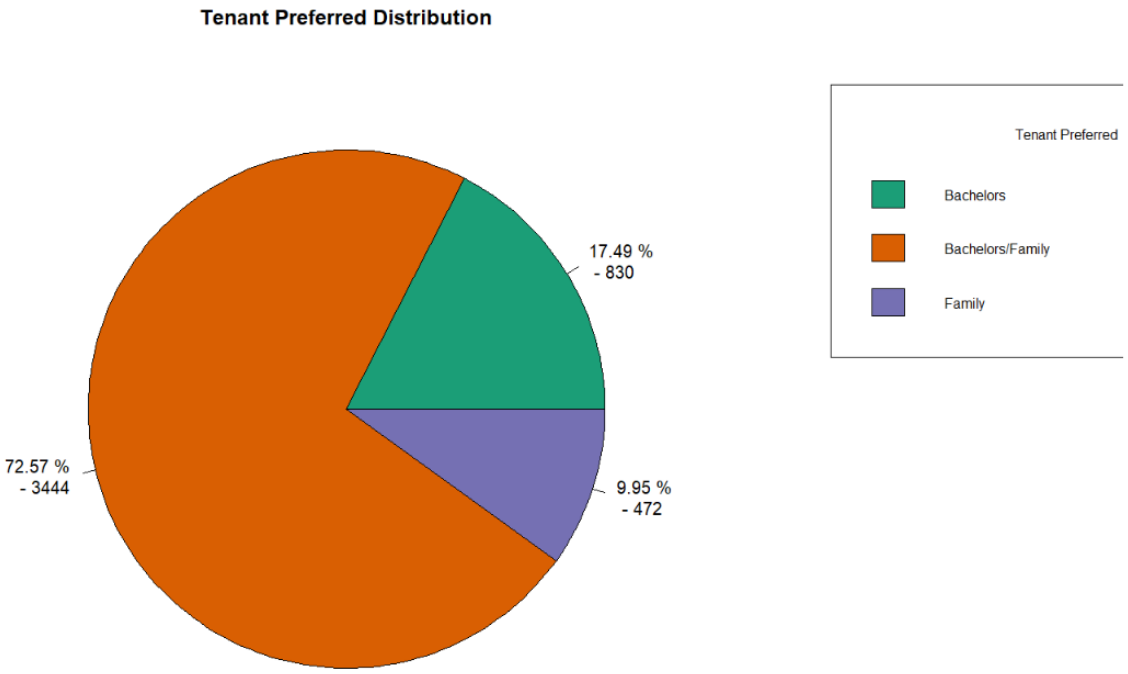


Figure 2-8: Data Distribution of Tenant Preferred Variable in Pie Chart

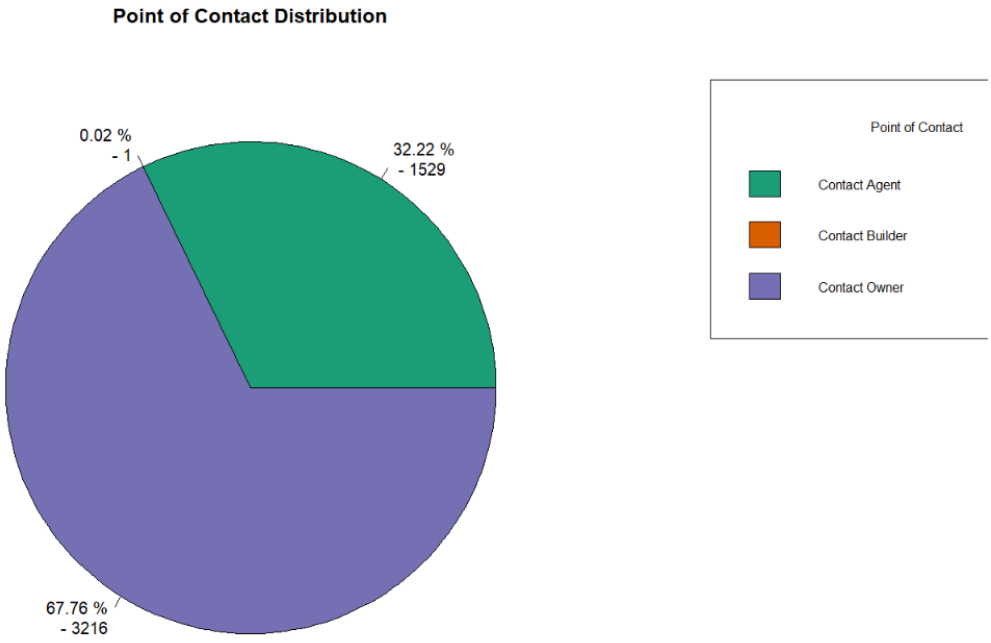


Figure 2-9: Data Distribution for Point of Contact Variable in Pie Chart

Pie chart is great tool to visualize the portion of a category to the whole as each pie's size is relative to the volume of certain category. Therefore, it is easy to identify the distribution of the data by slicing up different categories into their respective pie to investigate which area most data lies in (Gien, n.d.). For this project, pie chart is used to visualize the data distribution for Tenant Preferred and Point of Contact variable. As illustrated on the images above, the "Bachelors/Family" (72.57%) values in Tenant Preferred class occupies the largest portion of the whole followed by "Bachelors" (17.49%) then "Family" (9.95%). This means most of the rental postings target both Bachelor and Family as their primary client instead of either Bachelor or Family. For Point of Contact variable, the "Contact Owner" (67.76%) value occupies the largest portion followed by "Contact Agent" (32.22%) value while "Contact Builder" only present in 1 instance of the dataset. This means that most of the landlords of the rental house prefer the customers to contact the owner itself instead of contacting agent.

```
# data distribution by Tenant Preferred
# calculate the count frequency by different categorical values
tenant_preferred_count <-
  df %>%
    count(Tenant.Preferred) %>%
    mutate(percent = round(n / nrow(df) * 100, digit = 2))

# plot the pie chart
pie(
  tenant_preferred_count$n,
  labels = paste(unique(tenant_preferred_count$percent), "%\n -", unique(tenant_preferred_count$n)),
  main = "Tenant Preferred Distribution",
  col = brewer.pal(length(unique(df$Tenant.Preferred)), "Dark2")
)
legend(
  1.5, 1,
  title = "Tenant Preferred",
  legend = unique(tenant_preferred_count$Tenant.Preferred),
  cex = .8,
  fill = brewer.pal(length(unique(tenant_preferred_count$Tenant.Preferred)), "Dark2")
)
```

Figure 2-10: Code to Plot the Pie Chart

To plot the Pie Chart like images shown above, the frequency count needs to be calculated first. For this plot, the frequency count is computed using "count()" function from dplyr package that returns the frequency of occurrence by the targeted column's categorical value. Using pipe operator, the values returned are passed to the next function which is "mutate()" to calculate the percentage of distribution for each categorical values to easily compare the distribution. Then, the pie chart is plotted using "pie()" function by passing in the computed values and other configuration set-up such as labels, title, and fill colour. "legend()"

function is also used to add an appropriate legend for the pie chart so that every colour of the pie can be labelled correctly thus becoming easier for the audience to understand the chart.

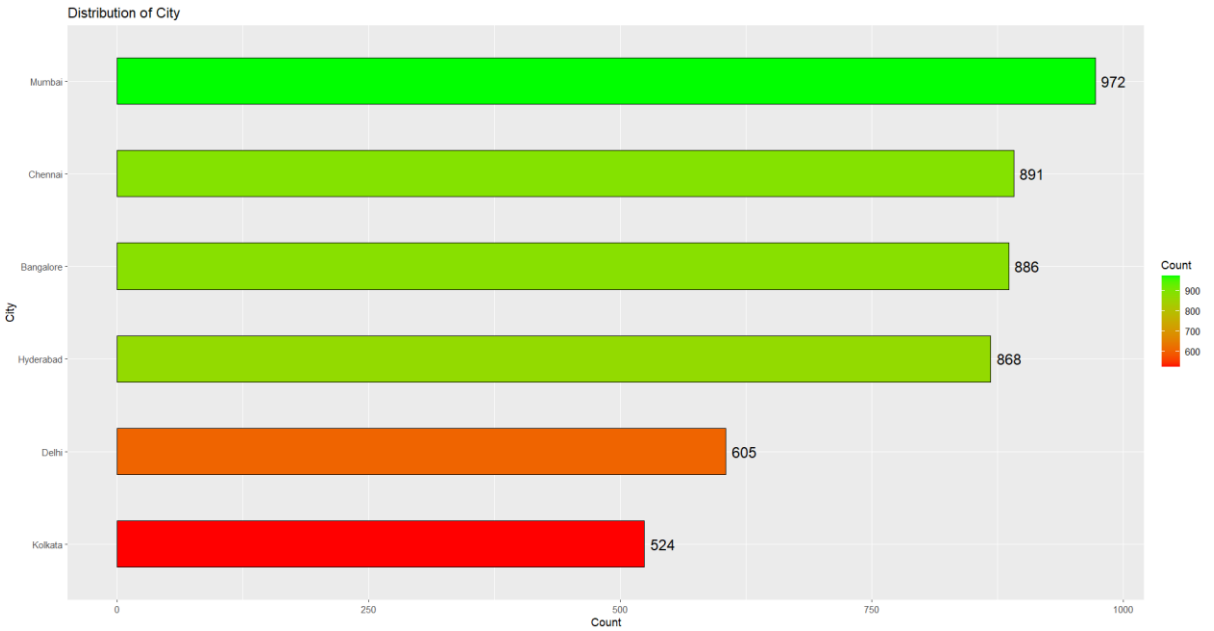


Figure 2-11: Data Distribution of City Variable in Bar Chart

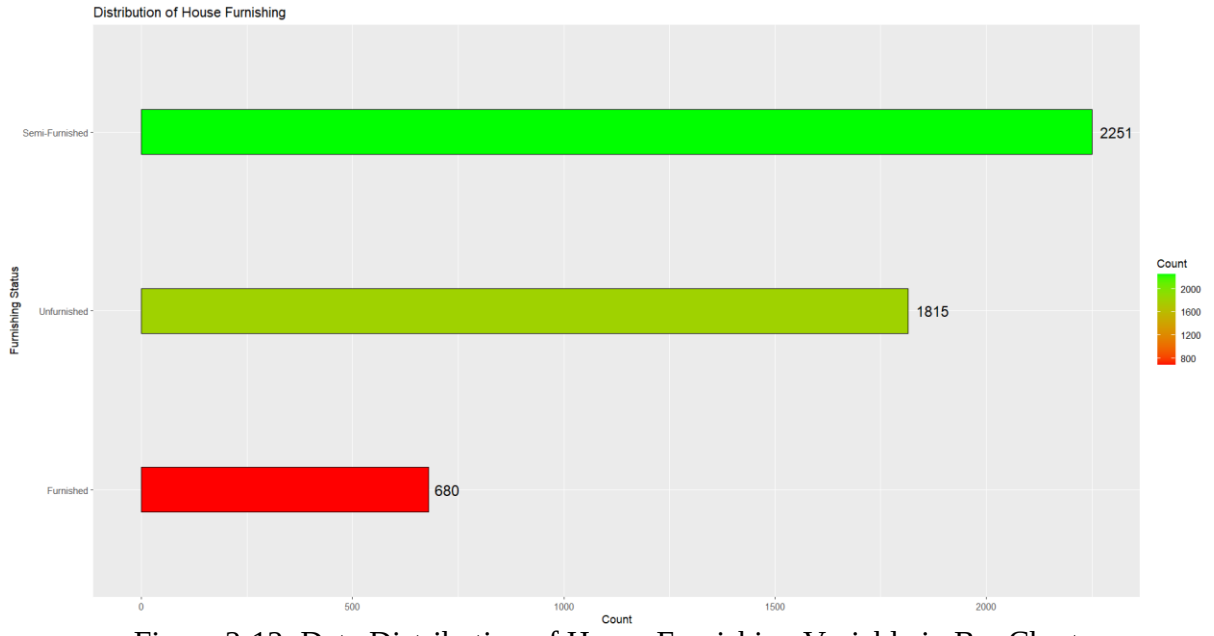


Figure 2-12: Data Distribution of House Furnishing Variable in Bar Chart

Bar chart is another tool that is often used for data understanding and exploration as it can plot numeric values for categorical value in the form of bar or strip. Each bar represents a categorical value, and the length of the bar represents the value of respective categorical value.

As a good practice, bar chart are always plotted on zero-based baseline to allow clear comparison (Yi, n.d.). In the images above, bar chart is used to visualize the distribution of City and Furnishing Status variable. For City variable, Mumbai contains highest frequency of house rental postings while Kolkata has the fewest rental house posting. Other cities has similar frequency of rental house posting which is from 860 to 900 range except Delhi has 605 rental house postings. For Furnishing Status variable, the semi-furnished houses has the largest portion in the dataset followed by unfurnished houses and furnished houses.

```
# distribution of data by city
df %>%
  count(City) %>%
  ggplot(mapping = aes(x = reorder(City, n), y = n)) +
    labs(title = "Distribution of City", x = "City", y = "Count") +
    geom_bar(mapping = aes(fill = n), stat = "identity", color = "black", width = 0.5) +
    scale_fill_gradient("Count", low = "red", high = "green", space = "Lab") + # gradient for fill
    geom_text(aes(label =  n), hjust = -0.25, size = 5) + # label at bar
    coord_flip() # horizontal bar
```

Figure 2-13: Code to Plot Bar Chart

Like plotting a pie chart, “count()” function is used to calculate the frequency count of the categorical values. Then, the frequency count is passed into the “ggplot()” function as arguments to plot a bar graph. The configurations are set by passing arguments and chaining up the functions using “+” operator as shown in the images above. “labs()” function configure the labels of bar chart such as title, x-axis label, and y-axis label. “geom_bar()” function set up the colour, border colour, and gradient fill of the bar chart while “scale_fill_gradient()” adjusts the colour used in the gradient fill. “geom_text()” function adds labels beside the bar for user to easily read the exact count of the categorical value. Meanwhile, “coord_flip()” function turns the vertical bar chart into horizontal bar chart.

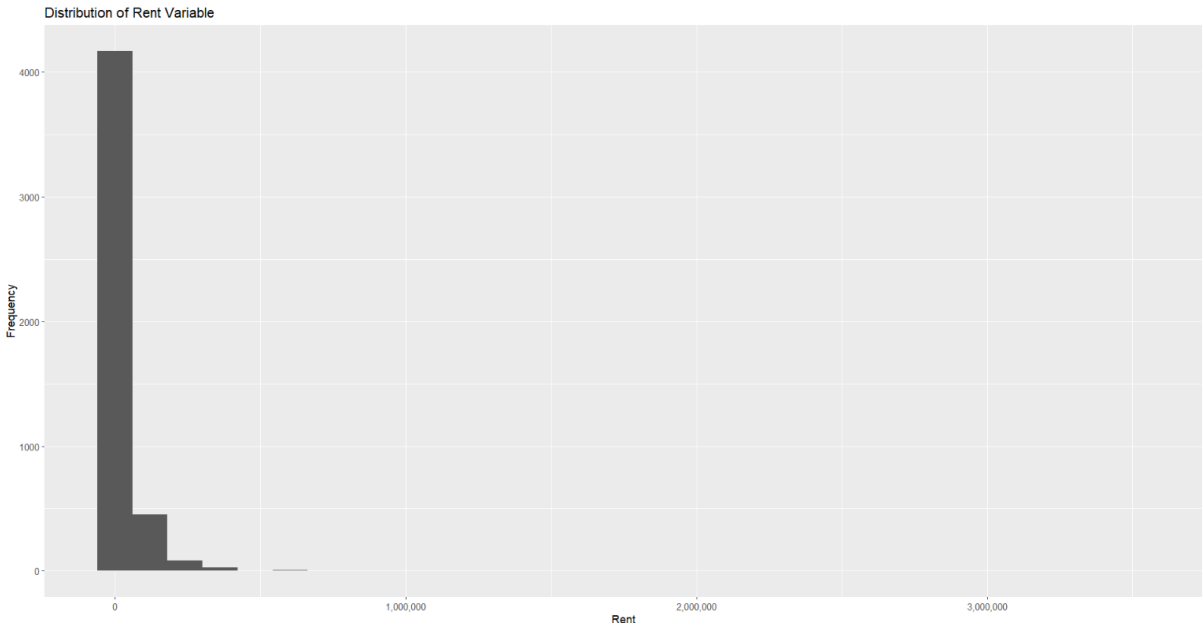


Figure 2-14: Distribution of Rent Variable with Histogram

Histogram is another great tool to visualize the frequency distribution of a continuous variable. It provides an overview of the frequency distribution by forming different shapes such as bell-shaped which represents normal distribution, right-skewed shape, and left-skewed shape (Six Sigma Daily, 2020). The images above illustrates the frequency distribution of Rent variable. Rent variable is continuous value so histogram is chosen instead of pie chart or bar chart for visualizing the distribution. For Rent variable, the histogram is extremely right skewed as most of the instances' rent price lies between 0 and 100,000 while other values scattered all around the places. This indicates that the Rent variable has outliers and will be fixed in data cleaning process later.

```
# distribution of Rent column with histogram
df %>%
  ggplot(mapping = aes(x = Rent)) +
  labs(title = "Distribution of Rent Variable", x = "Rent", y = "Frequency") +
  scale_x_continuous(labels = comma) +
  geom_histogram()
```

Figure 2-15: Code to Plot Histogram

Using ggplot2 package and "geom_histogram()" function, the histogram is plotted by passing in the necessary arguments. "scale_x_continuous()" function formats the scientific notation numbers on x-axis of the histogram to comma separated numbers using "comma" constant from scales package imported.

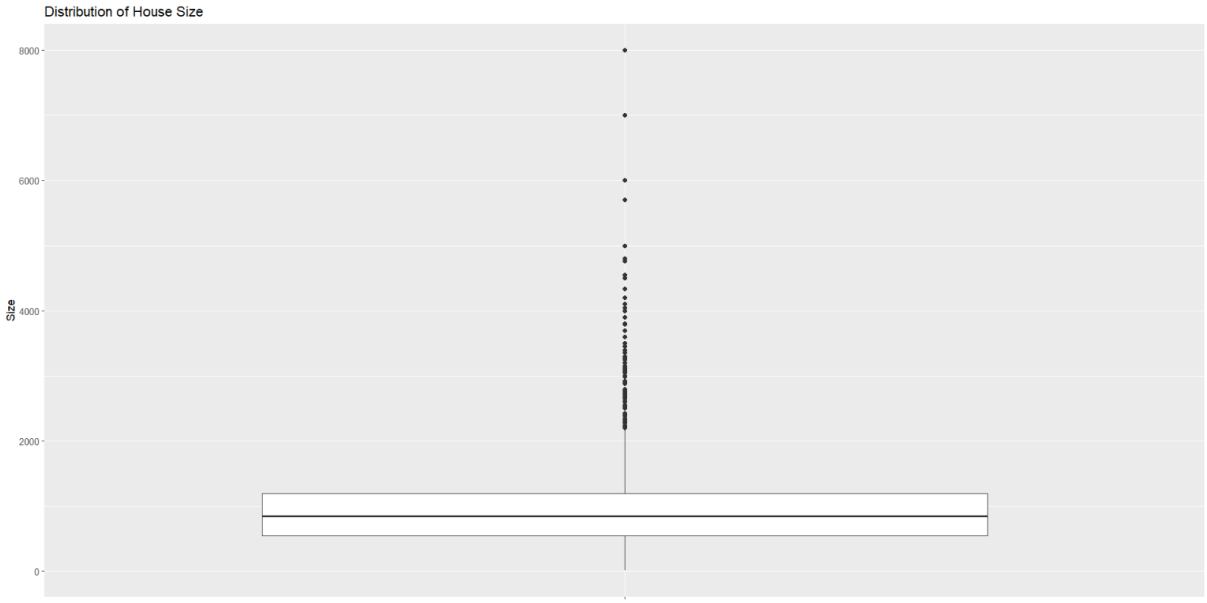


Figure 2-16: Distribution of House Size with Boxplot

Boxplot is also a type of visual aid that is often used for visualizing distribution of data based on statistical summary mentioned above including minimum, maximum, first quartile range, third quartile range, and median. The boxplot can help to identify whether the data is grouped together and the skewness of the distribution (Galarnyk, 2022). As shown in the image above, the boxplot is plotted for the distribution of house size. The boxplot indicates that the data is mostly lies between 550 and 1200 which is the box of the plot. The maximum value of House Size lies around 2200 and the dot outside the box indicates where the outliers are.

```
# for Size
df %>%
  ggplot(mapping = aes(x = " ", y = Size)) +
  labs(title = "Distribution of House Size", y = "Size", x = "") +
  geom_boxplot()
```

Figure 2-17: Code to Plot Boxplot for House Size Distribution

The boxplot is plotted using the ggplot2 package and “geom_boxplot()” function after passing all the required arguments.

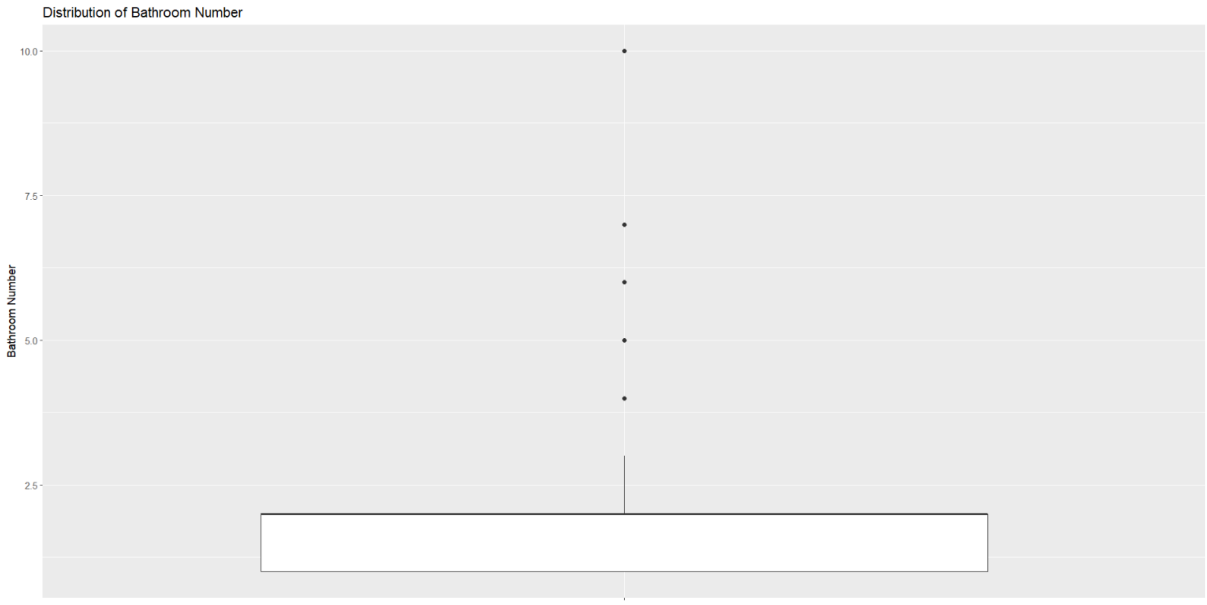


Figure 2-18: Distribution of Bathroom Number with Boxplot

The images above illustrates the distribution of Bathroom Number variable where the median of the Bathroom Number has the same value as the third quartile range value which is 2. The first quartile range value is 1 while there are some outliers which is represented by the dot outside the box.

```
# for Bathroom
df %>%
  ggplot(mapping = aes(x = " ", y = Bathroom)) +
  labs(title = "Distribution of Bathroom Number", x = "", y = "Bathroom Number") +
  geom_boxplot()
```

Figure 2-19: Code to Plot Boxplot for Bathroom Number Variable

The code to plot the boxplot above is the same as the code to plot boxplot for House Size but with different column.

3.0 Data Cleaning and Pre-Processing

3.1 Data Cleaning

Data cleaning is one of the most crucial phases in data analytics lifecycle as it ensures the quality data to be used for analysis. In short, data cleaning process includes correcting or removing corrupted, incorrectly formatted data, and missing values (Tableau, 2022).

```
> # change column name
> colnames(df) <- c(
+   "Posted.Date", "Facilities.Number", "Rental.Price",
+   "House.Size", "Floor", "Area.Type",
+   "House.Locality", "House.City", "House.Furnishing",
+   "Tenant.Targeted", "Bathroom.Number", "Contact.Person"
+ )
>
> # get summary of posted date
> summary(df)
 Posted.Date        Facilities.Number  Rental.Price       House.Size        Floor            Area.Type
 Length:4746        Min.   :1.000     Min.   :   1200   Min.   :  10.0   Length:4746        Length:4746
 Class :character   1st Qu.:2.000     1st Qu.:  10000   1st Qu.: 550.0   Class :character   Class :character
 Mode  :character   Median :2.000     Median :  16000   Median : 850.0   Mode  :character   Mode  :character
                    Mean   :2.084     Mean   :  34993   Mean   : 967.5
                    3rd Qu.:3.000     3rd Qu.:  33000   3rd Qu.:1200.0
                    Max.   :6.000     Max.   :3500000   Max.   :8000.0
 House.Locality      House.City        House.Furnishing   Tenant.Targeted    Bathroom.Number  Contact.Person
 Length:4746        Length:4746        Length:4746        Length:4746        Min.   : 1.000   Length:4746
 Class :character   Class :character   Class :character   Class :character   1st Qu.: 1.000   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Median : 2.000   Mode  :character
                                                                             Mean   : 1.966
                                                                             3rd Qu.: 2.000
                                                                             Max.   :10.000
>
```

Figure 3-1: Change Column Names

The first step in data cleaning phase is to change the column names of the original dataset to meaningful and descriptive names. As shown in the images above, the columns names are changed using “colnames()” function and assign it a vector of column names desired. After that, the column names are changed successfully according to the vector given.

```
                                                                             Max.   :10.000
> # Convert date column from character class to Date class
> df$Posted.Date <- as.Date(df$Posted.Date, format="%m/%d/%Y")
> class(df$Posted.Date)
[1] "Date"
>
> # Convert rental price to numeric instead of integer
> df$Rental.Price <- as.numeric(df$Rental.Price)
> class(df$Rental.Price)
[1] "numeric"
>
> # get summary of data
> summary(df)
  Posted.Date        Facilities.Number  Rental.Price       House.Size        Floor            Area.Type
 Min.   :2022-04-13  Min.   :1.000     Min.   :   1200   Min.   :  10.0   Length:4746        Length:4746
 1st Qu.:2022-05-20  1st Qu.:2.000     1st Qu.:  10000   1st Qu.: 550.0   Class :character   Class :character
 Median :2022-06-10  Median :2.000     Median :  16000   Median : 850.0   Mode  :character   Mode  :character
 Mean   :2022-06-07  Mean   :2.084     Mean   :  34993   Mean   : 967.5
 3rd Qu.:2022-06-28  3rd Qu.:3.000     3rd Qu.:  33000   3rd Qu.:1200.0
 Max.   :2022-07-11  Max.   :6.000     Max.   :3500000   Max.   :8000.0
 House.Locality      House.City        House.Furnishing   Tenant.Targeted    Bathroom.Number  Contact.Person
 Length:4746        Length:4746        Length:4746        Length:4746        Min.   : 1.000   Length:4746
 Class :character   Class :character   Class :character   Class :character   1st Qu.: 1.000   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Median : 2.000   Mode  :character
                                                                             Mean   : 1.966
                                                                             3rd Qu.: 2.000
                                                                             Max.   :10.000
>
```

Figure 3-2: Change Data Type

The next step of data cleaning is to change the data type of certain columns to the correct data type for analysis purpose. For instance, the Posted.Date column has character class before the data cleaning so the conversion to Date class is required as Date class allows data analysis using Date operations provided by R such as sorting, grouping and filtering.

```
> # sort in ascending order by date
> df <- df[order(df$Posted.Date), ]
>
> # check sorted data
> head(df, 5)
     Posted.Date Facilities.Number Rental.Price House.Size           Floor   Area.Type          House.Locality
1223  2022-04-13                3       260000       1800    10 out of 11 Carpet Area             JVPD Scheme
54    2022-04-23                2        15000       1000 Ground out of 2  Super Area               Bansdroni
245   2022-04-23                2         7000        450      1 out of 3 Carpet Area               safuipara
304   2022-04-23                2        20000        700      1 out of 2  Super Area Salt Lake City Sector 1
361   2022-04-23                1        15000       1000      2 out of 3  Super Area             Barrackpore
     House.City House.Furnishing  Tenant.Targeted Bathroom.Number Contact.Person
1223     Mumbai        Furnished           Family               4  Contact Agent
54      Kolkata      Unfurnished Bachelors/Family               2  Contact Owner
245     Kolkata        Furnished        Bachelors               2  Contact Owner
304     Kolkata        Furnished Bachelors/Family               1  Contact Owner
361     Kolkata      Unfurnished Bachelors/Family               1  Contact Owner
```

Figure 3-3: Sorting Data by Date

Afterwards, the dataset is sorted ascendingly by the Date column using “order()” function so that the analysis process can be eased.

```
> # check duplicated values
> df[duplicated(df), ]
 [1] Posted.Date       Facilities.Number Rental.Price      House.Size        Floor             Area.Type
 [7] House.Locality    House.City        House.Furnishing  Tenant.Targeted   Bathroom.Number   Contact.Person
<0 rows> (or 0-length row.names)
>
> # checking number of missing values in the dataframe
> sum(is.na(df))
[1] 0
> sum(is.null(df))
[1] 0
>
```

Figure 3-4: Check Duplicate and Missing Values

With “sum()”, “is.na()” and “is.null()” functions, the missing values are filtered and counted to identify if any missing values or N/A type value exists in the dataset to avoid conducting inaccurate analysis.

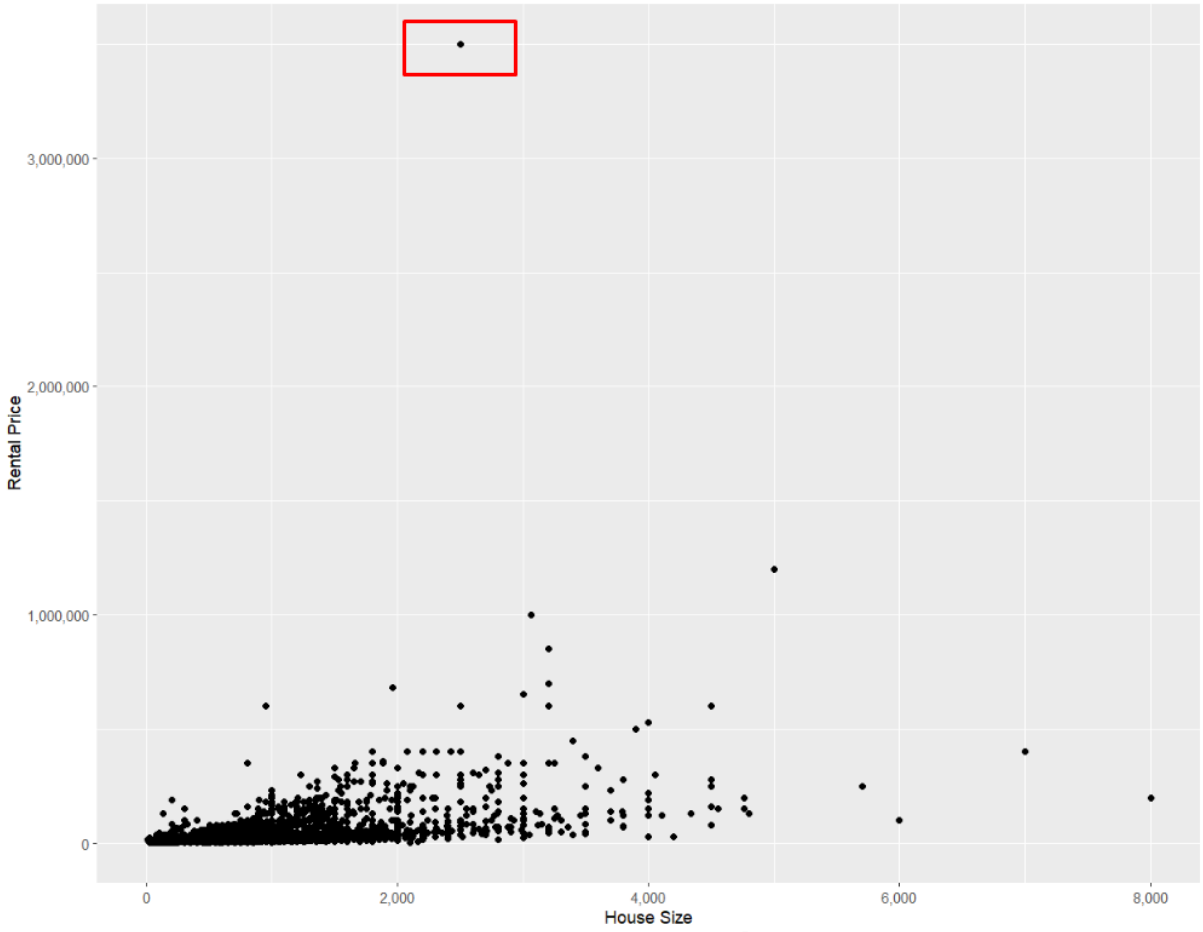


Figure 3-5: Detecting Outliers

As mentioned above, the dataset contains outliers that affect the quality of data according to the frequency distribution illustrated in histogram and boxplot. The scatterplot above plot Rental Price variable against the House Size variable and an outlier is found as it stays extremely far away from the other data points in the scatterplot. This outlier is occurred most likely due to human error during data entry process. To ensure the quality of data, the outlier must be fixed before conducting any analysis. As the data distribution of Rental Price is skewed, so median will be used as the replacement value for this outlier so that the replaced value can be much closer to the actual value.

```
> # outlier's data city is at Bangalore so we use Bangalore's median to replace
> df[df$Rental.Price == 3500000, ]$Rental.Price <- median(df[df$House.City == "Bangalore", ]$Rental.Price)
> df[df$Rental.Price == 3500000, ]
 [1] Posted.Date       Facilities.Number Rental.Price      House.Size        Floor             Area.Type
 [7] House.Locality    House.City        House.Furnishing  Tenant.Targeted   Bathroom.Number   Contact.Person
<0 rows> (or 0-length row.names)
>
```

Figure 3-6: Replacing Outlier Value

To replace the outlier, the median rental price of the rental houses in Bangalore city will be calculated first as the outlier is in the group of Bangalore city. Then, the computed median will be used to replace the outlier using the code shown above.

3.2 Data Transformation

```
> df$Rent.Per.Unit <- df %>% with(round(Rental.Price /House.Size, digit = 2))
> summary(df)
  Posted.Date         Facilities.Number  Rental.Price       House.Size        Floor            Area.Type
 Min.   :2022-04-13   Min.   :1.000     Min.   :   1200   Min.   :  10.0   Length:4746        Length:4746
 1st Qu.:2022-05-20   1st Qu.:2.000     1st Qu.:  10000   1st Qu.: 550.0   Class :character   Class :character
 Median :2022-06-10   Median :2.000     Median :  16000   Median : 850.0   Mode  :character   Mode  :character
 Mean   :2022-06-07   Mean   :2.084     Mean   :  34259   Mean   : 967.5
 3rd Qu.:2022-06-28   3rd Qu.:3.000     3rd Qu.:  33000   3rd Qu.:1200.0
 Max.   :2022-07-11   Max.   :6.000     Max.   :1200000   Max.   :8000.0
 House.Locality      House.City        House.Furnishing   Tenant.Targeted    Bathroom.Number  Contact.Person
 Length:4746        Length:4746        Length:4746        Length:4746        Min.   : 1.000   Length:4746
 Class :character   Class :character   Class :character   Class :character   1st Qu.: 1.000   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Median : 2.000   Mode  :character
                                                                             Mean   : 1.966
                                                                             3rd Qu.: 2.000
                                                                             Max.   :10.000

 Rent.Per.Unit
 Min.   :   0.57
 1st Qu.:  13.33
 Median :  20.00
 Mean   :  38.96
 3rd Qu.:  41.99
 Max.   :1500.00
>
```

Figure 3-7: Calculate Rental per Unit

To provide a more in-depth analysis, Rent per Unit is calculated by dividing the Rental Price variable with House Size variable then round the result values up to 2 decimal places to ensure standard values across the instances. The column is calculated using “with()” function that receive the data frame as arguments and “round()” function to round up the computed values.

```
> # parse floor to numbers, then calculate distance
> df[c("Current.Floor", "Total.Floor")] <- str_split_fixed(df$Floor, pattern=" out of ", 2)
> df$Current.Floor <- str_replace_all(df$Current.Floor, pattern="Ground", replacement="0")
> df$Current.Floor <- str_replace_all(df$Current.Floor, pattern="Upper Basement", replacement="-1")
> df$Current.Floor <- str_replace_all(df$Current.Floor, pattern="Lower Basement", replacement="-2")
> df$Current.Floor <- as.numeric(df$Current.Floor)
> df$Total.Floor <- as.numeric(df$Total.Floor)
> summary(df)
  Posted.Date         Facilities.Number  Rental.Price      House.Size        Floor           Area.Type
 Min.   :2022-04-13   Min.   :1.000     Min.   :   1200   Min.   :  10.0   Length:4746        Length:4746
 1st Qu.:2022-05-20   1st Qu.:2.000     1st Qu.:  10000   1st Qu.: 550.0   Class :character   Class :character
 Median :2022-06-10   Median :2.000     Median :  16000   Median : 850.0   Mode  :character   Mode  :character
 Mean   :2022-06-07   Mean   :2.084     Mean   :  34259   Mean   : 967.5
 3rd Qu.:2022-06-28   3rd Qu.:3.000     3rd Qu.:  33000   3rd Qu.:1200.0
 Max.   :2022-07-11   Max.   :6.000     Max.   :1200000   Max.   :8000.0

 House.Locality      House.City        House.Furnishing   Tenant.Targeted    Bathroom.Number  Contact.Person
 Length:4746        Length:4746        Length:4746        Length:4746        Min.   : 1.000   Length:4746
 Class :character   Class :character   Class :character   Class :character   1st Qu.: 1.000   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Median : 2.000   Mode  :character
                                                                             Mean   : 1.966
                                                                             3rd Qu.: 2.000
                                                                             Max.   :10.000

 Rent.Per.Unit     Current.Floor     Total.Floor
 Min.   :   0.57   Min.   :-2.000   Min.   : 1.000
 1st Qu.:  13.33   1st Qu.: 1.000   1st Qu.: 2.000
 Median :  20.00   Median : 2.000   Median : 4.000
 Mean   :  38.96   Mean   : 3.436   Mean   : 6.973
 3rd Qu.:  41.99   3rd Qu.: 3.000   3rd Qu.: 6.000
 Max.   :1500.00   Max.   :76.000   Max.   :89.000
                                    NA's   :4
>
```

Figure 3-8: Parse Floor Variable and Convert to Numeric Values

The Floor variable is character class value but contains some values such as current floor and total floor that can be parsed and used for analysis. Therefore, stringr package is installed and loaded to split the string given the delimiter and separate the values split into two columns. Using “str_split_fixed()” function from stringr package, the current and total floor of Floor variable is parsed and separated into Current.Floor and Total.Floor using the code above. Then some of categorical values parsed such as “Ground” are converted into integer value to standardize the records. The conversion of the categorical values to integer values is as the following table:

Categorical Value	Integer Value
Ground	0
Upper Basement	-1
Lower Basement	-2

Table 3-1: Conversion of Categorical Value to Integer Value

The conversion of values are achieved using “str_replace_all()” function from stringr package as well that replaces the string value given a pattern for the function to search. Afterwards, the parsed values of computed columns are converted to numeric data type so that mathematical operations can be applied on the columns later in the analysis phase.

```
> # having missing values in computed columns
> df[is.na(df$Total.Floor), ]
     Posted.Date Facilities.Number Rental.Price House.Size  Floor   Area.Type                          House.Locality
2884  2022-05-23                1        18000        450 Ground Carpet Area DDA Flat AD Block, Shalimar Bagh AD Block
4561  2022-05-31                3        15000       1270      1 Carpet Area                                  Tarnaka
4491  2022-06-12                3        15000        900      1  Super Area                          Malakpet, NH 9
2554  2022-06-18                2        20000        400      3  Super Area         Kasturba Niketan, Lajpat Nagar 2
     House.City House.Furnishing  Tenant.Targeted Bathroom.Number Contact.Person Rent.Per.Unit Current.Floor
2884      Delhi        Furnished Bachelors/Family               1  Contact Owner         40.00             0
4561  Hyderabad        Furnished           Family               2  Contact Owner         11.81             1
4491  Hyderabad   Semi-Furnished Bachelors/Family               3  Contact Owner         16.67             1
2554      Delhi      Unfurnished Bachelors/Family               1  Contact Owner         50.00             3
     Total.Floor
2884          NA
4561          NA
4491          NA
2554          NA
>
> # replace missing values for total floor
> df[is.na(df$Total.Floor) & df$House.City == "Delhi", ]$Total.Floor <- median(df[(!is.na(df$Total.Floor)) & (df$House.City == "Delhi"), ]$Total.Floor)
> df[is.na(df$Total.Floor) & df$House.City == "Hyderabad", ]$Total.Floor <- median(df[(!is.na(df$Total.Floor)) & (df$House.City == "Hyderabad"), ]$Total.Floor)
> df[is.na(df$Total.Floor), ]
 [1] Posted.Date       Facilities.Number Rental.Price      House.Size        Floor             Area.Type
 [7] House.Locality    House.City        House.Furnishing  Tenant.Targeted   Bathroom.Number   Contact.Person
[13] Rent.Per.Unit     Current.Floor     Total.Floor
<0 rows> (or 0-length row.names)
>
```

Figure 3-9: Eliminate Missing Values after Parsing Floor Variable

After parsing the Total.Floor variable, it is found that there are N/A type values in the computed results. Therefore, the N/A type value is replaced using the median of total floors in the same city with the code illustrated above.

```
> # calculate difference in floor (distance between current floor and ground floor)
> df$Difference.Floor <- abs(df$Current.Floor - 0)
>
> # remove floor variable
> df$Floor <- NULL
> summary(df)
  Posted.Date         Facilities.Number  Rental.Price       House.Size      Area.Type         House.Locality      House.City
 Min.   :2022-04-13   Min.   :1.000     Min.   :   1200   Min.   :  10.0   Length:4746        Length:4746        Length:4746
 1st Qu.:2022-05-20   1st Qu.:2.000     1st Qu.:  10000   1st Qu.: 550.0   Class :character   Class :character   Class :character
 Median :2022-06-10   Median :2.000     Median :  16000   Median : 850.0   Mode  :character   Mode  :character   Mode  :character
 Mean   :2022-06-07   Mean   :2.084     Mean   :  34259   Mean   : 967.5
 3rd Qu.:2022-06-28   3rd Qu.:3.000     3rd Qu.:  33000   3rd Qu.:1200.0
 Max.   :2022-07-11   Max.   :6.000     Max.   :1200000   Max.   :8000.0
 House.Furnishing   Tenant.Targeted    Bathroom.Number  Contact.Person     Rent.Per.Unit     Current.Floor     Total.Floor
 Length:4746        Length:4746        Min.   : 1.000   Length:4746        Min.   :   0.57   Min.   :-2.000   Min.   : 1.000
 Class :character   Class :character   1st Qu.: 1.000   Class :character   1st Qu.:  13.33   1st Qu.: 1.000   1st Qu.: 2.000
 Mode  :character   Mode  :character   Median : 2.000   Mode  :character   Median :  20.00   Median : 2.000   Median : 4.000
                                       Mean   : 1.966                      Mean   :  38.96   Mean   : 3.436   Mean   : 6.971
                                       3rd Qu.: 2.000                      3rd Qu.:  41.99   3rd Qu.: 3.000   3rd Qu.: 6.000
                                       Max.   :10.000                      Max.   :1500.00   Max.   :76.000   Max.   :89.000

 Difference.Floor
 Min.   : 0.000
 1st Qu.: 1.000
 Median : 2.000
 Mean   : 3.455
 3rd Qu.: 3.000
 Max.   :76.000
>
```

Figure 3-10: Calculate the Difference in Floor Variable and Remove Floor Variable

After ensuring the computed columns has no missing values or corrupted format, the Difference.Floor variable is computed by calculating the difference in the current floor with the ground floor. Using “abs()” function, mathematical absolute can be applied on the result value to avoid negative values in the records. After that, the Floor variable is removed from the dataset to avoid duplicated columns.

```
> # Remove unnecessary values (Remove "Area" and "Contact")
> df$Contact.Person <- str_replace_all(df$Contact.Person, pattern="Contact ", replacement="")
> df$Area.Type <- str_replace_all(df$Area.Type, pattern=" Area", replacement="")
> df[c("Contact.Person", "Area.Type")]
     Contact.Person Area.Type
1223          Agent    Carpet
54            Owner     Super
245           Owner    Carpet
304           Owner     Super
361           Owner     Super
```

Figure 3-11: Remove Unnecessary Value in Contact Person and Area Type Variable

The final step of data cleaning process is to remove unnecessary value in Contact.Person and Area.Type variables. The “Contact” and “Area” in the string values in both variables is redundant as the column names already stated the characteristics of the variable. Therefore, the string is removed from the values to avoid redundant data.

4.0 Data Analysis

4.1 Question 1: What is the factors that affect price of a rental house?

There are many factors that can affect a rental house's price such as the number of facilities provided, house size, and the location of the rental house. In this section, the analysis conducted aims to provide meaningful insights on the factors that affect the rental house price of the dataset.

4.1.1 Analysis 1-1: Changes of rental price according to date

The first analysis aims to provide an insight on how the rental price changes according to date ranging from 13rd April 2022 until 11st July 2022. The plot chosen for this analysis is a line plot that can easily visualize the changes of a dependent variable against independent variable (Turito, 2022). Before plotting a line chart, the data is grouped by respective dates and aggregated using mean function so that a continuous line of rental price can be plotted against the posted date.

```
# Analysis 1-1: Mean rental price in according to date
df %>%
  group_by(Posted.Date) %>%
  summarize(Rental.Price = mean(Rental.Price)) %>%
  mutate(Smoothed.Rental = rollmean(Rental.Price, k = 10, fill = NA, align = "center")) %>%
  ggplot() +
    labs(title = "Trend of Rental Price", x = "Posted Date", y = "Rental Price", colour = "Lines") +
    geom_line(mapping = aes(x = Posted.Date, y = Rental.Price, colour = "Rental Price"), size = 1.0) +
    geom_line(mapping = aes(x = Posted.Date, y = Smoothed.Rental, colour = "Smoothed Rental Price"), size = 1.0) +
    scale_y_continuous(labels = comma)
```

Figure 4-1: Code to Plot Line Chart

The code above groups the data by its posted date through "group_by()" function and aggregate the rental price with mean function through "summarize()" function. Then, a computed column that has the smoothed rental price is calculated and added to the dataset using "mutate()" function. To smoothen the line plot of original mean of rental price, "rollmean()" function is implemented to calculate the moving averages of the mean rental price with a window size of 10, so that the line in line plot can be smoothen and have less fluctuation to emphasize the trend. After the data is aggregated and computed, the data is fed into the "ggplot()" and "geom_line()" function to construct a line plot.

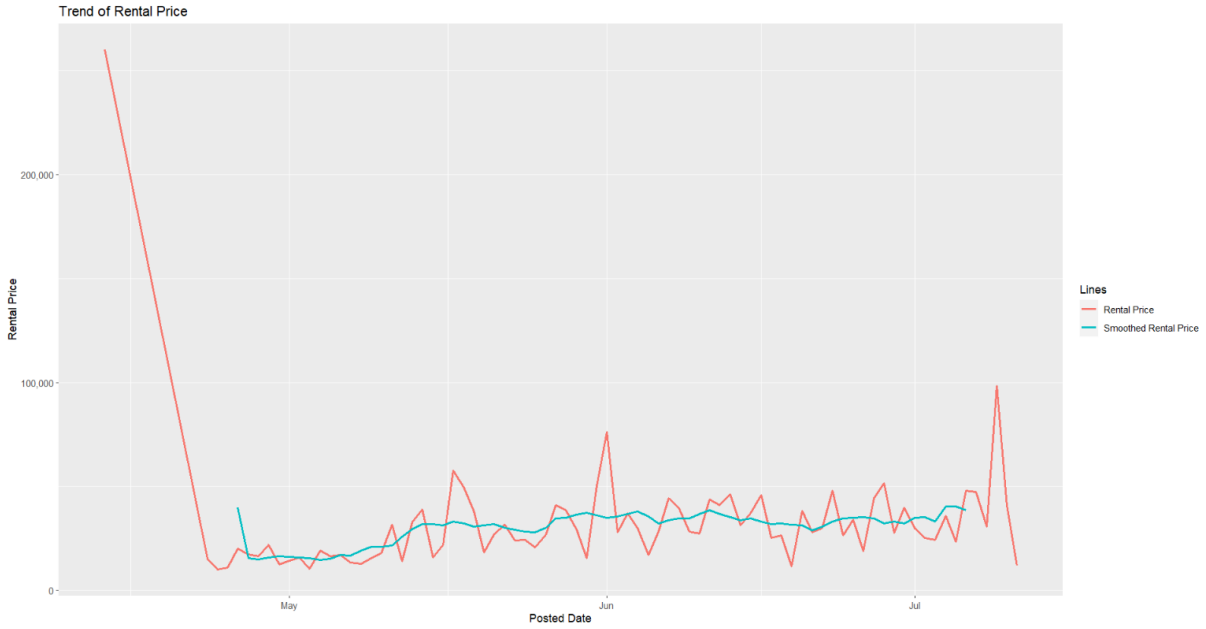


Figure 4-2: Line Chart Visualizing Rental Price Changes Across Time

Based on the line chart above, the rental price of the houses dropped quickly at the beginning then fluctuating up and down over long period of time. Although the prices are fluctuating, but there are no major increase or decrease in the rental price. This means that the change of rental price is seasonal as the data is experiencing a recurring change over time. To further visualize the changes in prices, the rolling average of the rental price of window size of 10 is calculated to smooth the line to identify the trend of the rental price. As shown in the image, there are no changes in the rental prices overall over the period given. To conclude, the date does not generate a significant impact on the rental price, but the rental price is seasonal as the data has recurring and predictable change every month.

4.1.2 Analysis 1-2: Rental price in each city by month

This analysis aims to analyse the changes in rental price of houses in each city by month. The visuals chosen for this analysis is 6 different line charts separated by city to isolate each city's rental price's changes. The data is grouped by month and aggregated using median function to identify the changes of rental price by month. Each city's line is also filled with different colour to easily differentiate among different cities.

```
# Analysis 1-2: Median rental price in each city by months
# plot in multiple line chart to visualize the changes in Rental Price by Month
# group data by city and month first
grouped_by_city_date <- df[, c("Posted.Date", "Rental.Price", "House.City")]

grouped_by_city_date <-
grouped_by_city_date %>%
  group_by(Month = lubridate::floor_date(Posted.Date, unit = "month"), House.City) %>%
  summarize(Median.Rental.Price = median(Rental.Price))

ggplot(data = grouped_by_city_date, mapping = aes(col = House.City, x = Month, y = Median.Rental.Price)) +
  labs(title = "Rental Price in Each City by Month", x = "Month", y = "Rental Price", colour = "City") +
  geom_line(size = 1.5) +
  geom_point(size = 3) +
  facet_wrap(~ House.City)
```

Figure 4-3: Code to Plot Separate Line Chart

The code above isolate the three columns out of the original dataset for aggregation purpose. The separated data is then grouped by months and city then the rental price is aggregated using the median function to get its median value. “ggplot()” function is used to plot the line chart where line’s colour is defined by the city categorical value. To emphasize the rental price value at each month, “geom_point()” function is used to plot the point on the graph. To separate the line chart into separate window of graph, “facet_wrap()” is implemented to separate the line plot by the city so that the trend in each city can be isolated and analysed easily.

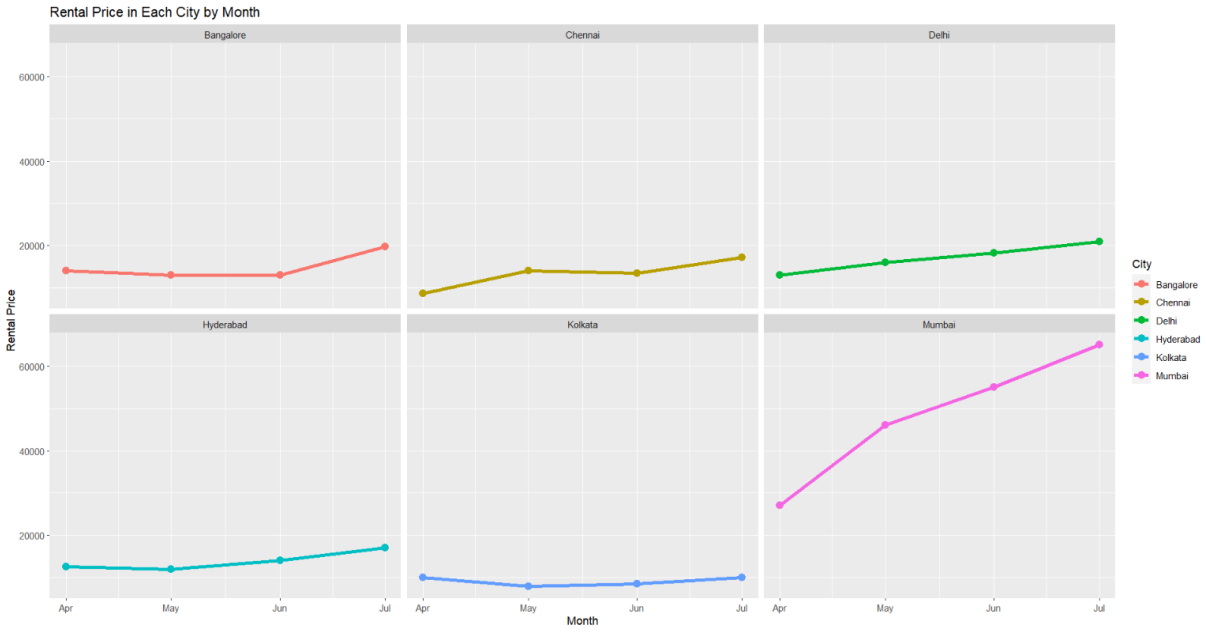


Figure 4-4: Visualizing Rental Price by Month in Each City

Based on the plot above, Bangalore, Chennai, Delhi, and Hyderabad has slight increase in the rental price over the months. For Kolkata, the rental prices of houses almost remain the same without increasing or decreasing much. For Mumbai, the rental price of houses has the

highest jump as compared to other cities. This is because Mumbai is the capital of Maharashtra state and the most famous city in India. Therefore, the population in Mumbai is undoubtedly one of the highest as compared to other cities (Mumbai Online Network, n.d.). As a result, the rental price of houses increases in a tendency higher than other cities.

4.1.3 Analysis 1-3: Rental price by facilities number

This analysis identifies whether the facilities number of the rental house will affect the rental price by comparing the rental price of houses with different number of facilities. This is achieved through grouping the houses with same facilities number then aggregate the rental price with median function. Then, the median of rental prices are compared against each other to identify the relationship between the facilities number and rental price.

```
# Analysis 1-3: Median Rental Price by Facilities Number
rent_by_facilities <-
df %>%
  group_by(Facilities.Number) %>%
  summarize(Median.Rental.Price = median(Rental.Price))

facilities_number <- unique(df$Facilities.Number)
ggplot(data = rent_by_facilities, mapping = aes(x = Facilities.Number, y = Median.Rental.Price)) +
  labs(
    title = "Median of Rental Price by Facilities Numbers",
    x = "Median of Rental Price",
    y = "Facilities Numbers",
    fill = "Median of Rental Price"
  ) +
  geom_bar(mapping = aes(fill = Median.Rental.Price), stat = "identity", col="#544c4b") +
  scale_fill_gradient(low = "green", high = "red", labels = comma) +
  scale_x_continuous("Facility Number", labels = as.character(facilities_number), breaks = facilities_number) +
  scale_y_continuous("Median Rental Price", labels = comma) +
  coord_flip()
```

Figure 4-5: Code to Plot Gradient Filled Horizontal Bar Chart

The code above groups the data by facility numbers then aggregate the rental price using median function. Then, “ggplot()” and “geom_bar()” function are used to plot the bar chart with the aggregated data. “scale_fill_gradient()” function is used to colour the bar based on the value of median rental price as indicated in the legend provided. “scale_x_continuous()” and “scale_y_continuous()” is used to format the x-axis and y-axis values into readable form instead of scientific notation. Lastly, “coord_flip()” function is used to switch the vertical bar chart into horizontal bar chart.

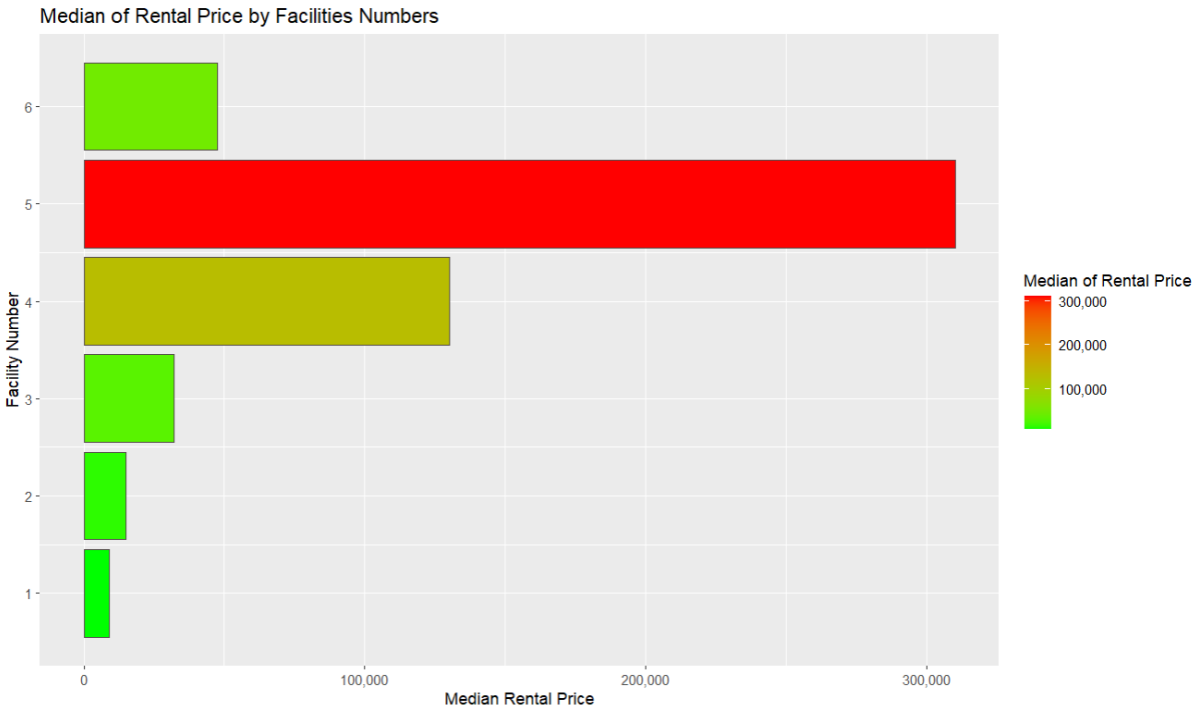


Figure 4-6: Comparing Rental Price by Facilities Number using Bar Chart

It is believed that more facilities such as bedroom, kitchen, and hall in a house will increase the house's value. The plot above proves that this belief is correct as the rental prices increases when the number of facilities increases. However, the increase of rental price stopped and decreases after the number of facilities passed 5. This may be due to there are no need for high number of facilities in the cities overall. To conclude, the higher the number of facilities, the higher the rental price will be until a certain point then the price will drop significantly.

4.1.4 Analysis 1-4: Relationship between house size and rental price

This analysis discover the relationship between house size and rental price using scatterplot and form a regression line to visualize the trend of the relationship. Scatterplot is chosen to visualize the relationship because scatterplot can effectively visualize the relationship between two variables (Corporate Finance Institute, 2022). Besides scatterplot, regression line is also calculated and plotted on the graph to identify whether the independent variable, House Size, has an impact on the dependent variable, Rental Price. The intercept and coefficient of the regression line is calculated using linear modelling method with "geom_smooth()" function.

```
# Analysis 1-4: House Size
# plot scatterplot and line of best fit
ggplot(data = df, mapping = aes(x = House.Size, y = Rental.Price, add = "reg.line")) +
  labs(title = "Relationship between House Size and Rental Price", x = "House Size", y = "Rental Price") +
  geom_point(col = "#9c1a11") +
  geom_smooth(method = lm) +
  scale_y_continuous("Rental Price", labels = comma) +
  stat_cor() +
  stat_regline_equation(label.y.npc = .9, size = 4, col = "black")
```

Figure 4-7: Code to Plot Scatterplot with Regression Line

The code above constructs a scatterplot and a regression line to show the relationship. "ggplot()" function and "geom_point()" function is used to plot the scatterplot given the data. Then, regression line is plotted on the graph using "geom_smooth()" function by passing in the method, linear model as arguments to the function. Using "stat_regline_equation()" and "stat_cor()", the equation of the regression line and correlation details are added to the graph to state the coefficients and intercepts of the regression line.

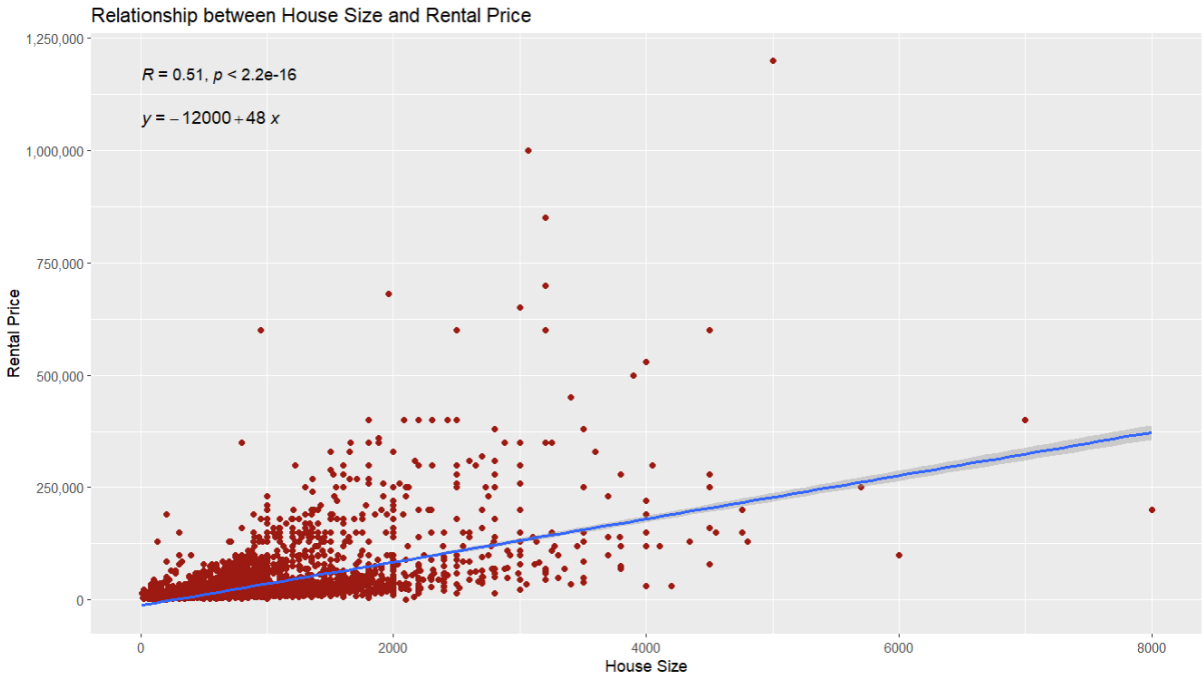


Figure 4-8: Scatterplot to Visualize Relationship

It is assumed that the house size are related to the rental price of houses as bigger house size means larger capacity to house more people. To prove this relationship, the combination of scatterplot and regression line is implemented for visualizing both the variables. Based on the plot above, the dot on the plot are mostly concentrated at the left bottom corner which means most of the rental houses has the house size below 2000 and rental price under 125000 Rupee. On overview, the scatterplot shows that the rental price of houses has a trend of

increasing when the house size increases. The regression line plotted also indicates that there are increasing relationship between rental price and house size. Besides, the p-values is also far smaller than 0.05 so the hypothesis where these 2 variables are independent can be rejected. To conclude, the house size has a positive impact on the rental price which means the rental price increases when house size increases.

4.1.5 Analysis 1-5: Relationship between area type and rental price

Scatterplot can also be used to visualize the relationship and distribution between categorical variable and continuous variable. This enables the isolation of distribution of continuous variable by its categorical variable to better identify the relationship and distribution of both variables.

```
# Analysis 1-5 Area Type
df %>%
  ggplot(mapping = aes(x = Area.Type, y = Rental.Price, colour = Area.Type)) +
    labs(title = "Relationship between Area Type and Rental Price", x = "Area Type", y = "Rental Price") +
    geom_point()
```

Figure 4-9: Code to Plot Relationship

The code above plot scatterplot that describes the relationship between area type and rental price using “ggplot()” and “geom_point()” functions.

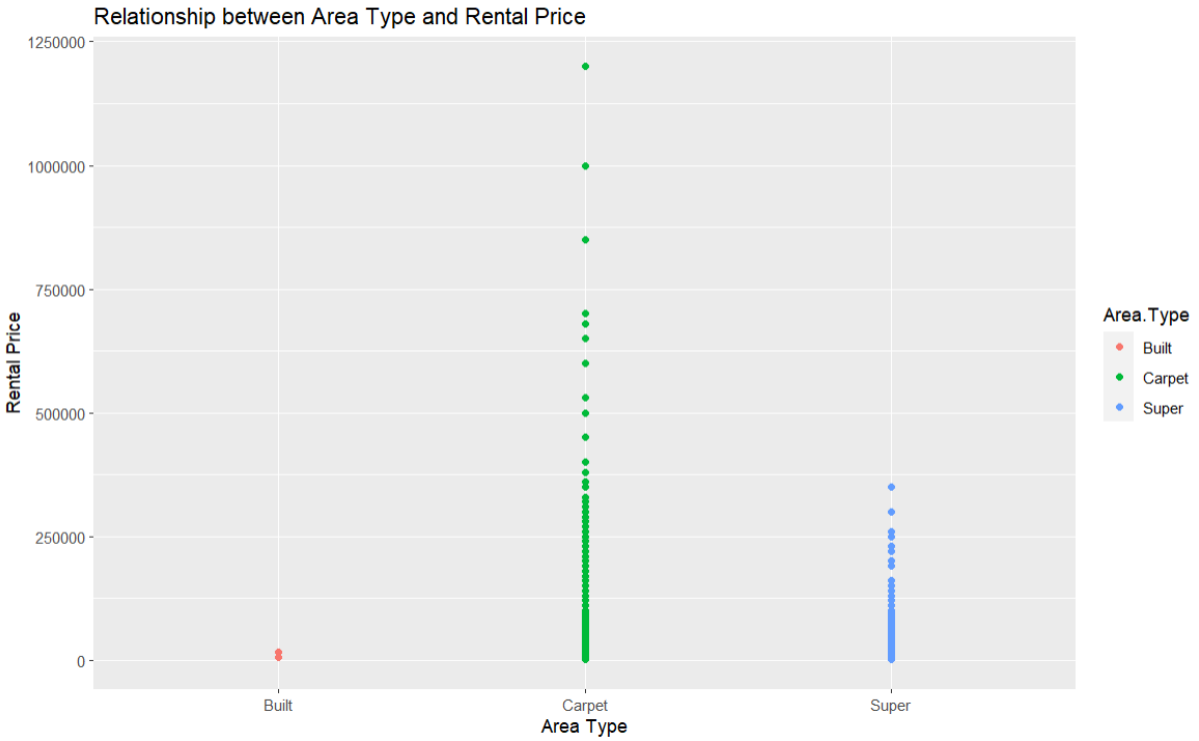


Figure 4-10: Relationship between Area Type and Rental Price

According to the scatterplot above, the built area type has very few samples so meaningful insights cannot be extracted effectively. For carpet area type, the rental price ranges from 0 and 1250000 where most of the rental houses lies between 0 and 300000. Super area type is ranging from 0 and 375000 where its price overall is lower than that of carpet area type. This is because carpet area type is the area that is covered by carpet or also known as usable area by the house owner according to Real Estate (Regulation and Development), RERA. This area includes the area of living room, bedroom, balconies and exclude the thickness of internal wall. In short, it represents the area covered by external wall. Unlike carpet area type, super area type consists of all the area of the house including corridor, elevator, and lift (ICICI Bank, 2022). Therefore, the carpet area type measures the house size in a more precise manner for the customers as it represents the net usable area. So, the carpet area type has more and higher range of the rental price than the super area type because the house size measured in terms of this area type include only usable area.

4.1.6 Analysis 1-6: Rental price per unit by area type in each city

To conduct an in-depth analysis on the rental price, the data is grouped by area type and city then aggregated with median function to calculate the median rental price per unit by the

area type and city located. This is to examine the relationship between rental price per unit between area types in different cities. To effectively visualize various categorical variables in a plot, grouped bar chart is used to conduct the analysis. This is because grouped bar chart works well when there are sub-categories in the dataset.

```
# Analysis 1-6 Median Rental per Unit of each Area Type by City
rent_by_area_city <-
  df %>%
  group_by(Area.Type, House.City) %>%
  summarize(Median.Rental.Per.Unit = median(Rent.Per.Unit))

ggplot(data = rent_by_area_city, mapping = aes(fill = Area.Type, x = House.City, y = Median.Rental.Per.Unit)) +
  ggtitle("Median of Rental per Unit for each Area Type by City") +
  xlab("City") +
  ylab("Median of Rental per Unit") +
  guides(fill = guide_legend(title = "Area Type")) +
  geom_bar(stat = "identity", position = "dodge", col = "#544c4b", width = .75)
```

Figure 4-11: Code to Plot a Grouped Bar Chart

As illustrated by the code above, the data is grouped by area type and house city first then the rental price per unit is aggregated through median function to calculate its median value. “ggplot()” and “geom_bar()” functions are used to plot a grouped bar chart by adjusting the position argument in “geom_bar()” function to “dodge” value so that grouped bar chart can be constructed. “guides()” function is used to modify the legend’s title to a descriptive title.

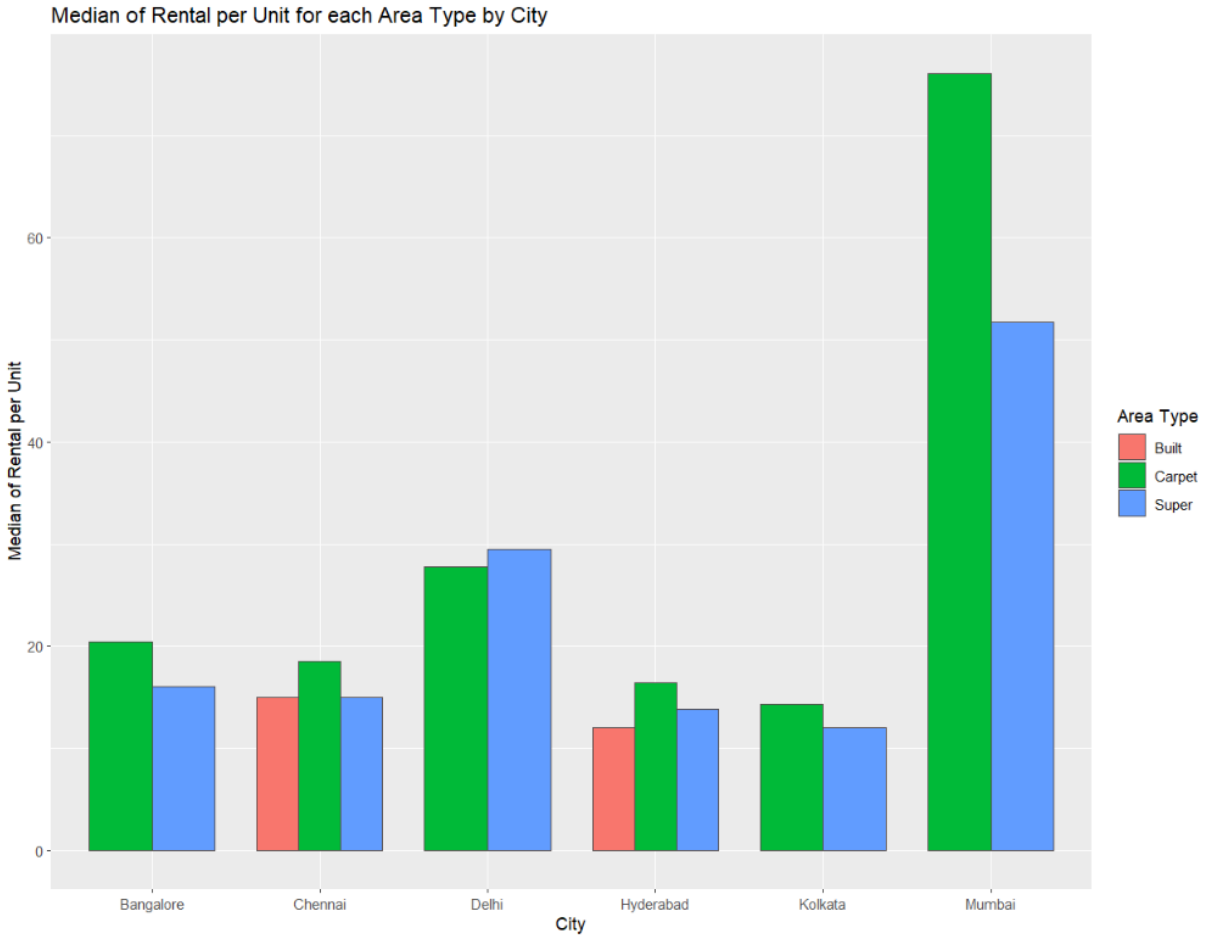


Figure 4-12: Rental Price per Unit by Area Type in Each City

Based on the grouped bar chart above, the rental price of carpet area type houses is higher than that of super area type except for Delhi where the super area type has higher rental price than that of carpet area type. The reason where most of the rental price of carpet area type is higher is because carpet area type measures house size in net usable area form. Therefore, all the house size measured can be used for activities such as bedroom and living room.

4.1.7 Analysis 1-7: Rental price in each city

This analysis provides the distribution of rental price in each city using violin plot which is a type of distribution plot like boxplot. Unlike boxplot, violin plot allows the analyst to identify the peak distribution of continuous variable and the density of the value in variables (Carron, 2021).

```
# Analysis 1-7: Rental Price by City
df %>%
  ggplot(mapping = aes(x = House.City, y = Rental.Price, fill = House.City)) +
    labs(title = "Rental Price by City", x = "House City", y = "Rental Price") +
    geom_violin()
```

Figure 4-13: Code to Plot Violin Plot

The code above illustrate plotting violin plot with “geom_violin()” function where the violin colour’s is determined by the city.

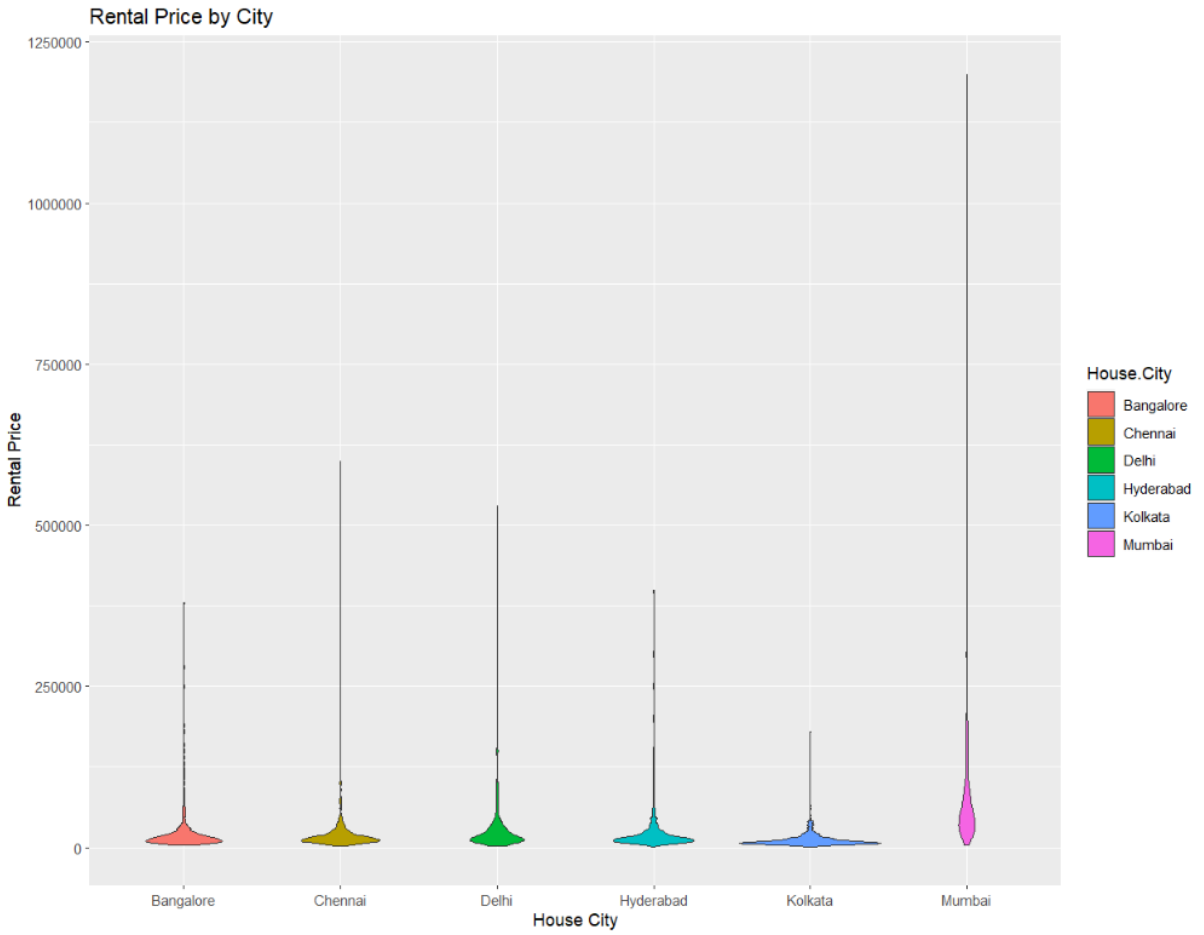


Figure 4-14: Violin Plot for Distribution of Rental Price by City

According to the plot above, the distribution for Bangalore, Chennai, Delhi, and Hyderabad has similar distribution of rental price. Chennai has higher rental price distribution as compared to the other two cities. This is because Chennai is city that has one of the highest speed Internet accesses in India which is important to most of the people nowadays (Britannica, 2022). Kolkata has overall cheaper rental houses as illustrated in the violin plot as the density of the violin spike at the lower side. This might be due to the culture of setting up a hut called

bastis on an open land as there are only three-fourths of the housing units are occupied and dwelled inside. So, the people living in Kolkata can choose not to need to rent a house but to set up basti for living due to cheaper housing fee (Brintannica, 2022). Therefore, the rental price is cheaper in overall in Kolkata as the demand for rental house is lower as compared to other cities. As mentioned above, Mumbai is one of the most prosperous cities in India, the rental price range of houses is overall higher than all the other cities in the dataset. To conclude, Mumbai has the highest rental price distribution followed by Bangalore, Chennai, Delhi, and Hyderabad while Kolkata has the lowest rental price of houses as compared to other cities.

4.1.8 Analysis 1-8: Median of rental price per unit by house furnishing in each city

This analysis provide insight on how the house furnishing will impact the rental price per unit in each city using a stack bar chart. The data is grouped by city and house furnishing first then aggregated using median function to calculate the median rental price per unit of houses.

```
# Analysis 1-8: House Furnishing
price_by_city_furnishing <- df[, c("House.City", "House.Furnishing", "Rent.Per.Unit")]

# group data by city and furnishing status
grouped_city_furnishing <-
  df %>%
  group_by(House.City, House.Furnishing) %>%
  summarize(Agg.Rental = median(Rent.Per.Unit))

# plot a stack bar chart
ggplot(data=grouped_city_furnishing, mapping=aes(fill = House.Furnishing, x = House.City, y = Agg.Rental)) +
  ggtitle("Rental Price in each City by Furnishing Status") +
  xlab("City") +
  ylab("Rental Price per Unit") +
  guides(fill=guide_legend(title="Furnishing Status")) +
  geom_bar(position = "stack", stat = "identity", col="black", width=0.45) +
  geom_text(aes(label = Agg.Rental), position = position_stack(vjust = 0.5)) +
  scale_fill_brewer(palette = "Dark2") +
  scale_y_continuous(labels = comma)
```

Figure 4-15: Code to Plot Horizontal Grouped Bar Chart

The code above selects the desired columns as stated in the “c()” function and store it in a variable. Then, the sliced data is grouped by city and house furnishing then the rental price per unit is aggregated with median function. Afterwards, the horizontal grouped bar chart is plotted with “ggplot()” and geom_bar()” function by passing in the required arguments.

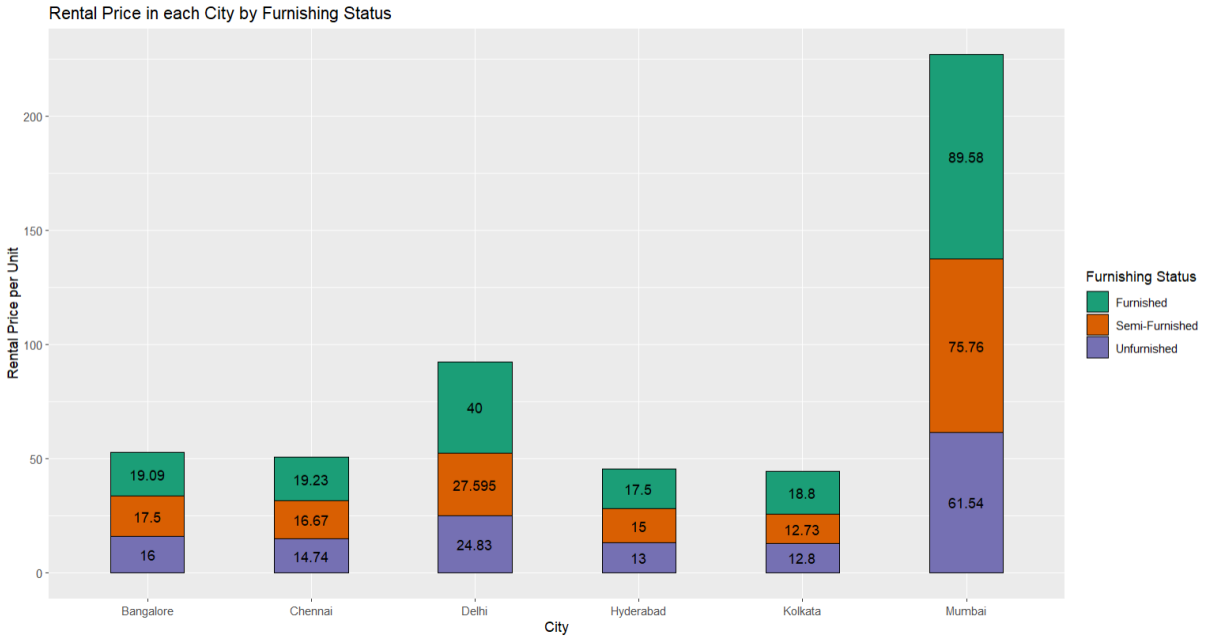


Figure 4-16: Rental Price per Unit in Each City by House Furnishing

Based on the plot above, the rental price of furnished house is the highest followed by semi-furnished then unfurnished among all cities. This is because furnished house includes the furniture that can improve the quality of living of the house owners and the cost of rent increases due to furniture installed. Therefore, the demand and cost to rent for furnished house will be higher as compared to houses that are semi-furnished or unfurnished. This results in higher rental price for furnished house regardless of the city.

4.1.9 Analysis 1-9: Relationship between tenant targeted and rental price

This analysis identifies the distribution of the rental price categorized by tenant targeted to compare against each other so that meaningful insights can be extracted. The data undergoes log transformation first before analysis as the rental price distribution for all three categories are extremely skewed which makes it difficult for comparison. Therefore, log transformation is applied to reduce the skewness of the rental price distribution (West, 2022).

```
# Analysis 1-9: Targeted Tenant
# after log transformation
ggplot(data = df, mapping = aes(x = Tenant.Targeted, y = log(Rental.Price), fill = Tenant.Targeted)) +
  labs(
    title = "Distribution of Rental Price by Tenant Targeted after Log Transformation",
    x = "Tenant Targeted",
    y = "Rental Price after Log Transformation",
    fill = "Tenant Targeted"
  ) +
  geom_boxplot()
```

Figure 4-17: Code to Plot Rental Price Distribution

The code above plot a boxplot categorized by the tenant targeted to explore the distribution of the rental price by tenant targeted. The y-axis value which is rental price undergoes log transformation using “log()” function first before plotting. This helps to reduce the skewness of the data so that the analyst can compare the price range better with the boxplot.

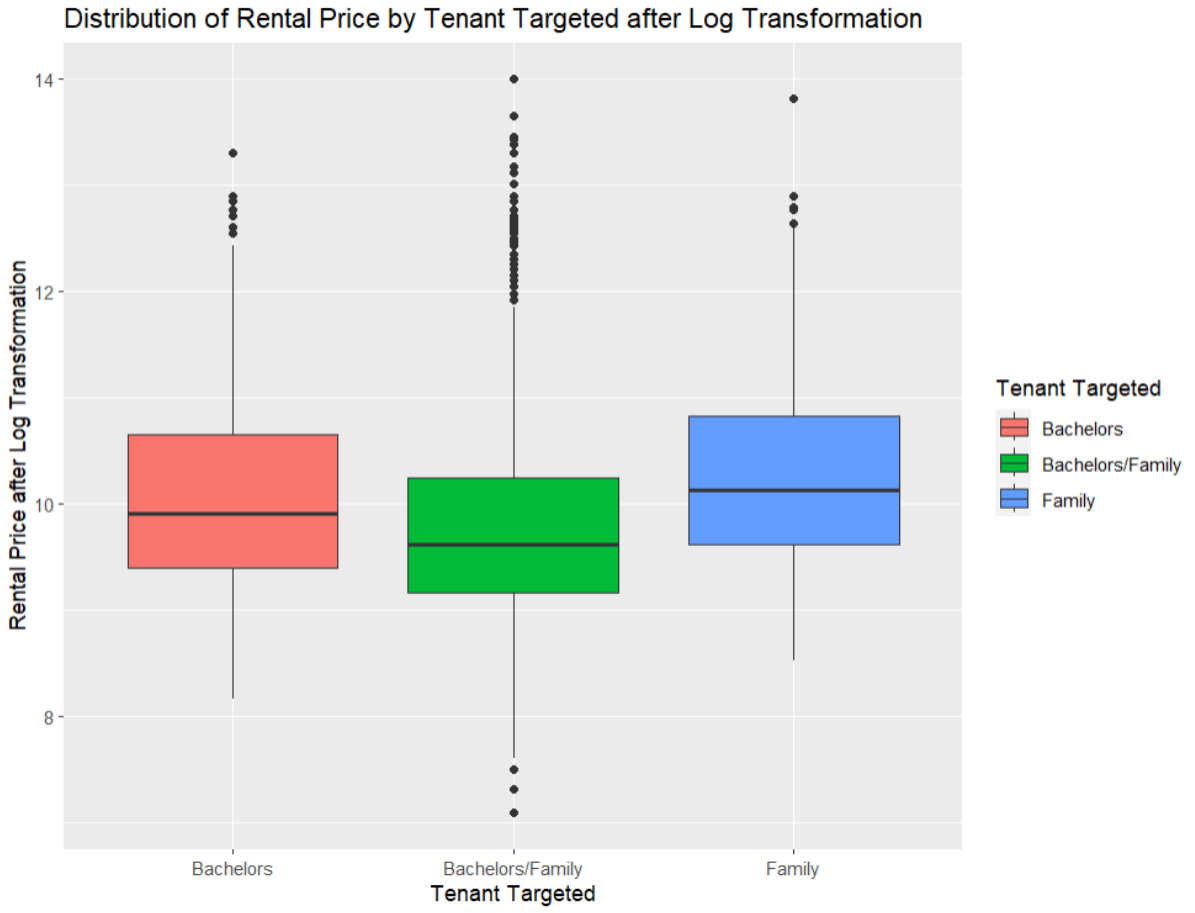


Figure 4-18: Distribution of Rental Price by Tenant Targeted

As illustrated in the plot above, the rental price distribution of houses that target family are slightly higher than that of houses that target bachelors. This is because the bachelors will be having less income than the family group as the bachelors are still studying in university.

This means that most bachelors are full-time students which does not have a stable income that can support them. The rental price distribution of "Bachelors/Family" is lower than the other two categories but it has more varieties in the rental price which means the rental price is that concentrated within the box as compared to other categories. This is because the rental house that target both bachelors and family ("Bachelors/Family") has less unique characteristics that can attract customers as the house does not target a specific group of people. On the other hand, the rental price of "Bachelors/Family" is more scattered as the rental houses has facilities that can satisfy the needs of bachelors or family group, so the rental price depends heavily on the features of the houses.

4.1.10 Analysis 1-10: Rental price per unit by bathroom number

This analysis provide an analysis on the relationship between the bathroom number of rental house and the rental price per unit. The data is first grouped by bathroom number that has already been converted to character class first then the rental price per unit is aggregated using median function. Afterwards, a line plot is charted using the aggregated data.

```
# Analysis 1-10: Bathroom Number
price_by_bathroom <- df[, c("Rental.Price", "Bathroom.Number")]
price_by_bathroom$Bathroom.Number <- as.character(df$Bathroom.Number)
grouped_bathroom <-
  df %>%
  group_by(Bathroom.Number) %>%
  summarize(Median.Rental = median(Rent.Per.Unit))

ggplot(data = grouped_bathroom, mapping = aes(x = Bathroom.Number, y = Median.Rental)) +
  labs(
    title = "Relationship between Bathroom Number and Median of Rental Price per Unit",
    x = "Bathroom Number",
    y = "Median of Rental Price per Unit"
  ) +
  geom_point(col = "#696969") +
  geom_line(col = "#7F7F7F", size = 0.75) +
  scale_x_continuous(labels = unique(df$Bathroom.Number), breaks = unique(df$Bathroom.Number))
```

Figure 4-19: Code to Plot the Line Chart

The code above group the bathroom number then aggregate the rental price per unit using median function. Then, "geom_line()" and "geom_point()" functions is used to plot the line chart with necessary arguments.

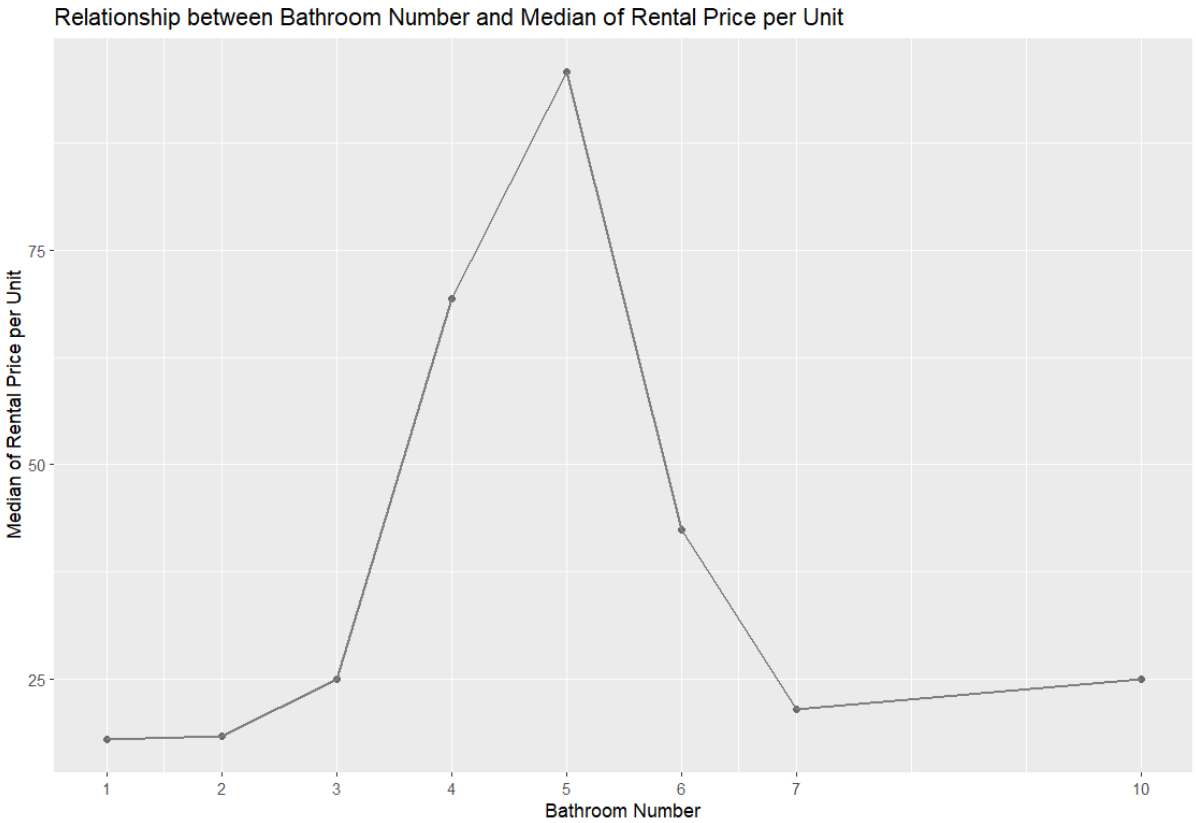


Figure 4-20: Relationship between Bathroom Number and Rental Price per Unit

Based on the plot above, the rental price per unit has the trend of increasing when the bathroom number of house increases. However, this trend has stopped after the bathroom number exceeds 5. This can be seen on the plot as the rental price per unit drops significantly after the bathroom number past 5. This may be due to the reason that the tenants do not need excessive bathrooms which causes the rental price to drop after the bathroom exceeds a certain number as the demand is decreasing. To conclude, the tenants in market only need up to 5 bathrooms in the rental house posted.

4.1.11 Analysis 1-11: Rental price per unit by contact person

The analysis aims to provide insights on the rental price distribution by the contact person. This is achieved through plotting scatterplot categorized by contact person to identify the rental price distribution.

```
# Analysis 1-11: Contact Person
ggplot(data = df, mapping = aes(x = Contact.Person, y = Rental.Price, colour = Contact.Person)) +
  labs(
    title = "Rental Price Distribution by Contact Person",
    x = "Contact Person",
    y = "Rental Price"
  ) +
  geom_point()
```

Figure 4-21: Code to Plot Violin Plot

The code above constructs the scatterplot demonstrated with "ggplot()" and "geom_point()" functions.

Figure 4-22: Rental Price Distribution by Contact Person

Based on the scatterplot above, the rental price distribution for builder is inaccurate as the sample for this category is too few which cannot generate meaningful insight. For agent type, the rental price distribution scattered at wider range than the owner type. The rental price distribution of agent type lies between 0 and 1250000 where the owner type has its distribution lies between 0 and 625000. This means that the rental price of house posted by the agents is higher than that from the owner. This is because the agent will need to extract some percentage of commission fee as the reward for completing a trade. This increases the cost of renting a house out which leads to higher rental price of the house (Aellina, 2019). Besides, the houses

offered by agent and its agency also has more varieties in terms of the house type from flat, apartment to luxury condominium and villa. This is because the agent has access to a platform known as Multiple Listing Service (MLS) which provides all the houses for rent or sale data to the agent (Key Inspection Services, n.d.). Therefore, the price range of houses provided by the agent has more range than the houses offered by the owner themselves.

4.1.12 Analysis 1-12: Rental price against difference in floor

This analysis generate insight on the relationship between difference in floor and rental price. The difference in floor is calculated by subtracting the current floor with the ground floor which is the distance between the current floor and ground floor. Scatterplot is used to visualize this relationship as it is effective in visualizing relationship between continuous variables.

```
# Analysis 1-12: Difference In Floor
ggplot(data = df, mapping = aes(x = Difference.Floor, y = Rental.Price)) +
  labs(
    title = "Relationship between Difference in Floor and Rental Price",
    x = "Difference in Floor",
    y = "Rental Price"
  ) +
  geom_point(col = "seagreen")
```

Figure 4-23: Code to Plot Scatterplot

The code above constructs the scatterplot demonstrated in the with “ggplot()” and “geom_point()” functions.

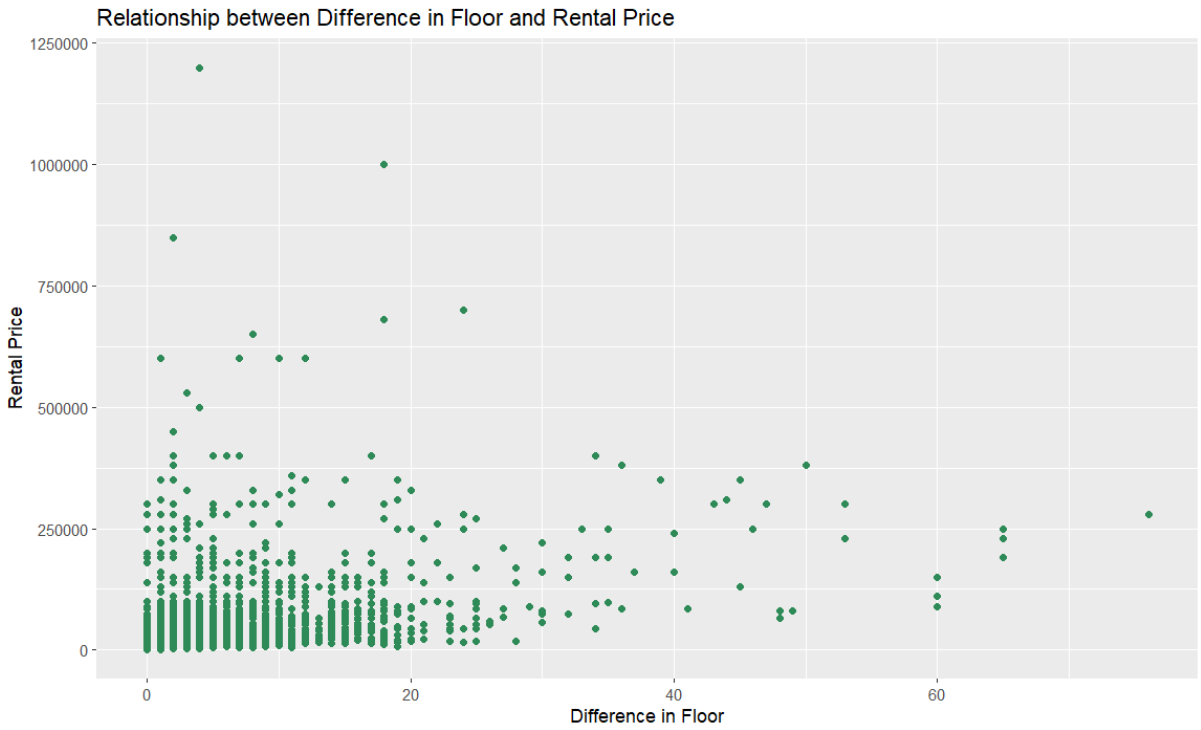


Figure 4-24: Relationship between Difference in Floor and Rental Price per Unit

Based on the scatterplot above, the rental price has the trend of slightly increasing when the difference in floor increases. This means that the difference floor has slight positive impact on the rental price although the magnitude in the increase is low. This might be due to lesser difference floor means the tenants only need to travel little distance to exit the house. The convenience of this feature increases little value to the house rented but it is not a necessary feature as there are facilities provided such as lift and elevator in most apartment nowadays. Therefore, the time saved is not that much of a difference when the difference in floor is low.

4.1.13 Analysis 1-13: Rental price by total floor group

This analysis aims to provide an analysis on the rental price distribution on different total floor group after grouping the total floor into their respective groups. For example, house with total floor of 5 will be included in “0-9” group. Then, boxplot is constructed to visualize the rental price distribution.

```r
# Analysis 1-13: Total Floor against Price
# separate the total floor into groups like 0-10, 11-20...
plot_total_floor <- function() {
  final_df <- df
  breaks <- seq(from = -1, to = 90, by = 10)
  labels <- c(
    "0 - 9",
    "10 - 19",
    "20 - 29",
    "30 - 39",
    "40 - 49",
    "50 - 59",
    "60 - 69",
    "70 - 79",
    "80 - 89"
  )

  final_df <-
    df %>%
      mutate(floor_group = cut(Total.Floor, breaks = breaks, labels = labels))

  final_df %>%
    ggplot(mapping = aes(x = floor_group, y = Rental.Price, fill = floor_group)) +
      labs(title = "Relationship between Floor Group and Rental Price", x = "Floor Group", y = "Rental Price") +
      geom_boxplot()

}
plot_total_floor()
```

Figure 4-25: Code to Plot Boxplot

The code above constructs the boxplot categorized by the total floor group demonstrated. “cut()” function is used to label each total floor by its group as stated in the vector “labels”. Then, “ggplot()” and “geom_boxplot()” functions is used to construct the graph by passing in all the necessary arguments.

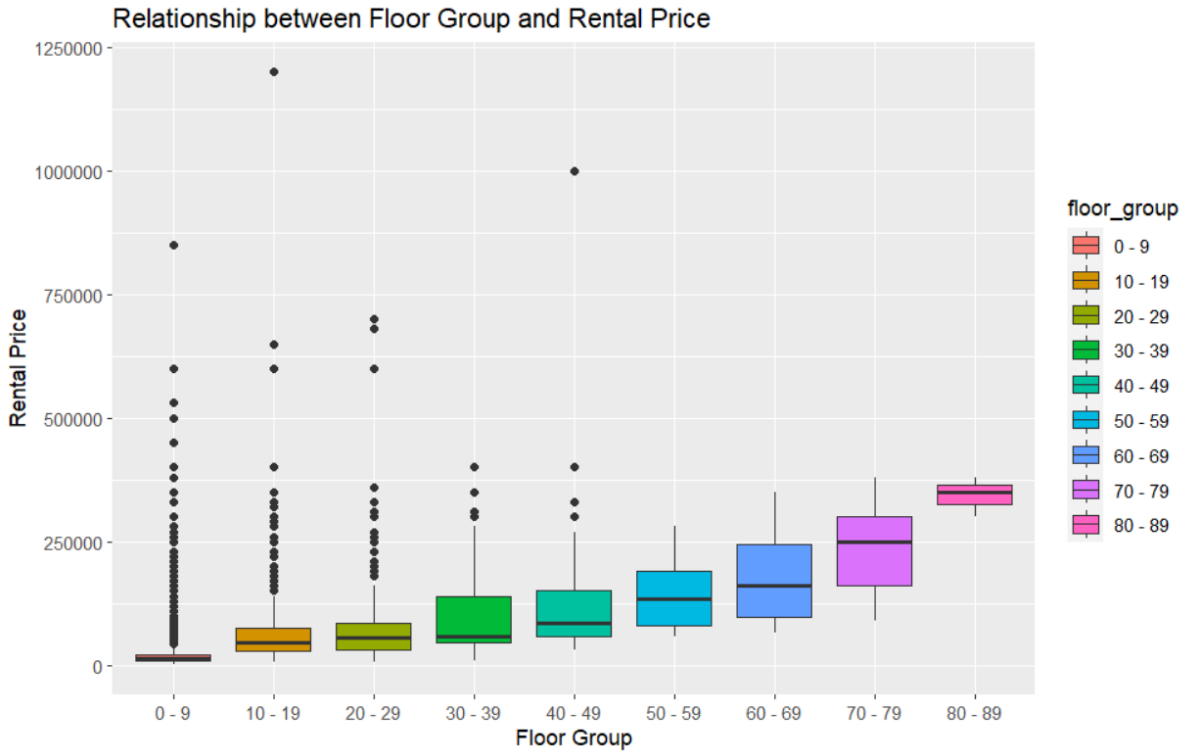


Figure 4-26: Relationship between Floor Group and Rental Price

According to the boxplot above, the rental price distribution categorized by 9 total floor groups ordered by the total floor number has the trend of increasing when the total floor increases. For the first 3 total floor groups, the rental price distribution lies between 0 and 125000 Rupee while the last 3 total floor groups' distribution is between 100000 and 300000. This means that the price range of the first 3 total floor groups are overall lower than the last 3 total floor groups. This is because the first 3 total floor groups indirectly prove that the house rented is single, double storey house, or flat apartment which is cheaper in overall as compared to the last 3 total floor groups which is usually luxury apartment. The reason why luxury apartment is more expensive is because that it provides various utilities and facilities such as swimming pool, 24-hours security guard, and sweat zone. To conclude, higher total floor group usually means luxury houses such as luxury apartment so the rental price overall will be higher than that of lower floor group.

4.1.14 Analysis 1-14: Correlation between all variables

This analysis provides an overview of the correlation between various variables in the dataset given. The visual used to achieve this is correlation plot where it visualizes the correlation matrix calculated with colour to emphasize the correlation between variables. The correlation coefficient is calculated using Pearson correlation formula for this analysis. Pearson correlation has value ranging between -1 and 1 that measures the strength of the correlation between two variables (Berg, n.d.). Primary goal for this visualization is to find out which variables are correlated with the rental price so that the factors that affect rental price can be discovered in high level view.

```r
final_df <- df

# Integer encoding for categorical variables to conduct correlation analysis
# Area Type convert to numbers
replace_number <- 1
for (area_type in unique(df$Area.Type)) {
  final_df[final_df$Area.Type == area_type, ]$Area.Type <- replace_number
  replace_number <- replace_number + 1
}
final_df$Area.Type <- as.numeric(final_df$Area.Type)

# House City convert to numbers
replace_number <- 1
for (city in unique(df$House.City)) {
  final_df[final_df$House.City == city, ]$House.City <- replace_number
  replace_number <- replace_number + 1
}
final_df$House.City <- as.numeric(final_df$House.City)

# House Furnishing convert to numbers
replace_number <- 1
for (furnishing in unique(df$House.Furnishing)) {
  final_df[final_df$House.Furnishing == furnishing, ]$House.Furnishing <- replace_number
  replace_number <- replace_number + 1
}
final_df$House.Furnishing <- as.numeric(final_df$House.Furnishing)

# Tenant Targeted convert to numbers
replace_number <- 1
for (tenant in unique(df$Tenant.Targeted)) {
  final_df[final_df$Tenant.Targeted == tenant, ]$Tenant.Targeted <- replace_number
  replace_number <- replace_number + 1
}
final_df$Tenant.Targeted <- as.numeric(final_df$Tenant.Targeted)

# Contact Person convert to numbers
replace_number <- 1
for (contact_person in unique(df$Contact.Person)) {
  final_df[final_df$Contact.Person == contact_person, ]$Contact.Person <- replace_number
  replace_number <- replace_number + 1
}
final_df$Contact.Person <- as.numeric(final_df$Contact.Person)
```

Figure 4-27: Data Preparation for Correlation Plot

The code above illustrates the data preparation algorithm for the correlation plot. Firstly, integer encoding for the categorical variable will be conducted as Pearson correlation only appropriate for quantitative variables. Therefore, the categorical values such as “Agent”, “Owner”, and “Builder” will be labelled as integer such as 1, 2, and 3 so that Pearson formula can be applied on the variables. Integer encoding is applied on 3 categorical variables which is Area Type, House City, House Furnishing, Tenant Targeted, and Contact Person. Afterwards, the transformed data will be used to calculate the correlation matrix and construct correlation plot as demonstrated above.

```
# House City convert to numbers
final_df <- final_df[
  c(
    "Facilities.Number",
    "House.Size",
    "Area.Type",
    "Bathroom.Number",
    "Difference.Floor",
    "House.City",
    "House.Furnishing",
    "Tenant.Targeted",
    "Contact.Person",
    "Rental.Price"
  )
]
correlation_matrix <- cor(final_df, method = "pearson")
corrplot(correlation_matrix, method = 'color', order = 'alphabet', addCoef.col = 'black', col = COL2('PuOr', 10))
```

Figure 4-28: Constructs Correlation Plot

The code above uses the transformed data to plot correlation graph. Firstly, the correlation matrix will be calculated first using “cor()” function provided by R after passing all required arguments. Then, “corrplot()” function will be used to plot the correlation graph with colour to indicate the strength of correlation.

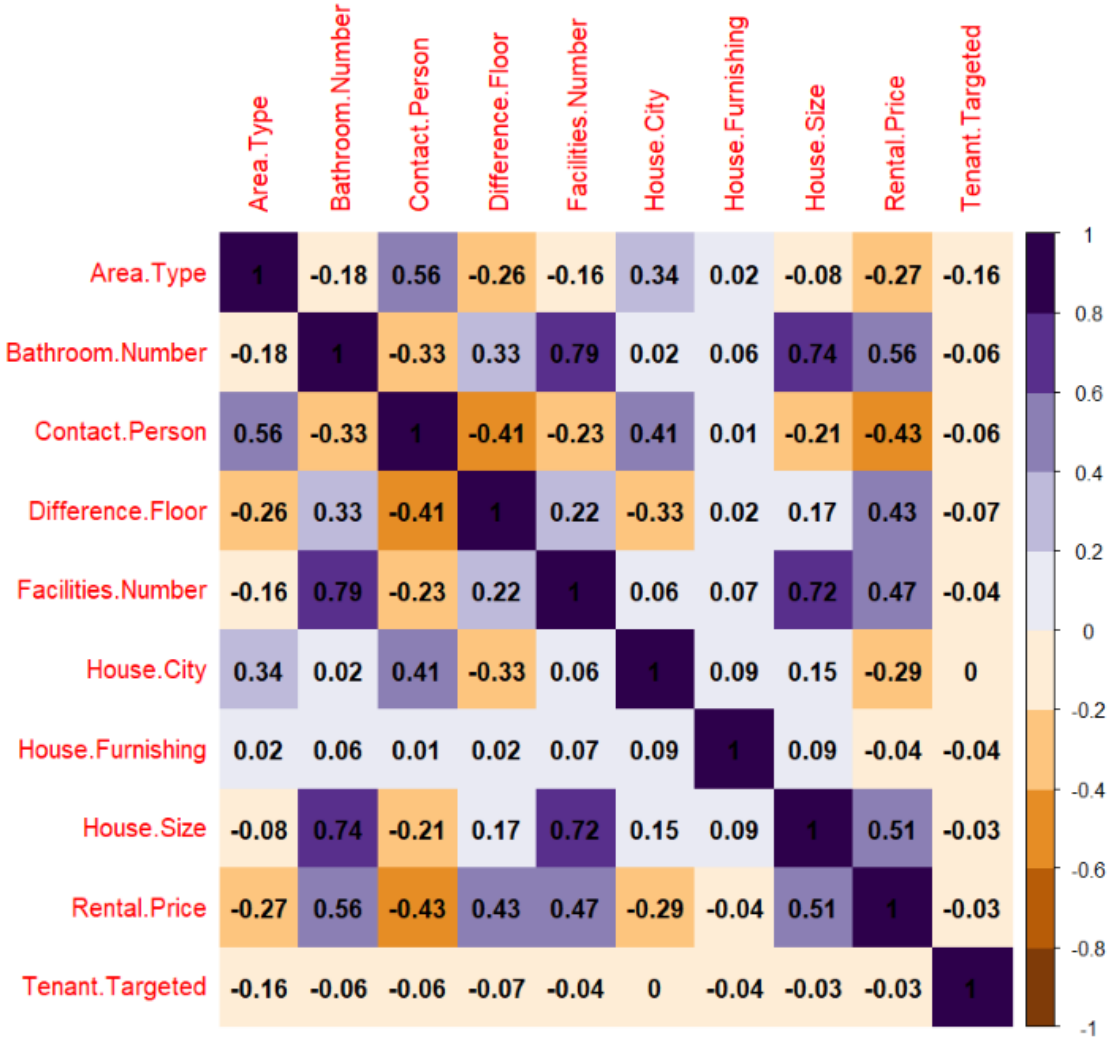


Figure 4-29: Correlation between All Variables

As illustrated by the correlation plot above, the correlation between variables can be visualized and identified whether the observed variable is correlated. For example, the bathroom number are positively correlated with the facilities number, house size, and the rental price of house. This is because houses with more bathroom number usually have higher number of facilities which results in higher rental price as the features add values to the house. Looking at rental price column, there are a few variables that are correlated with the rental price. To illustrate, the bathroom number, difference in floor, facility number, and house size has a positive correlation with the rental price. In other words, these rental price of the house increases when the value of variables mentioned increases. Besides, the contact person has negative correlation with the rental price which means when the contact person decreases the rental price will increases or vice versa. This is because the contact person is labelled 1, 2, and 3 for “Agent”, “Owner”, and “Builder” so the contact person is agent when contact person

decreases. As agent requires commission fee to complete the trade, rental house posted by agents usually cost more than the houses posted by the owner.

4.1.15 Conclusion

There are indeed many factors that can affect the rental price of houses in India based on the dataset given. For instance, the date variable has no significant impact on the rental price which means there are no specific trend on the rental price. However, when the data is partitioned by city, there are an upward trend in the rental price in most cities except Kolkata. For Mumbai, it has the highest jump of rental price as compared to other cities as it is one of the most prosperous cities in India. This also means that the city has an impact on the rental price of the houses. For example, Mumbai's overall rental price of houses is higher than the other cities as the population is dense thus the demand for houses are higher. Besides, the number of facilities in a house also have a positive impact on the rental price as more bedrooms add more values to the house. However, this trend stopped until the number of facilities exceed certain value and starts decreasing after that. This also applies to the number of bathrooms in the house as the tenants does not need that many facilities and bathrooms in the house thus causing the demand to decrease. Other than facilities, the house size also have positive impact on the rental price as larger house size can house more people and more facilities. On the other hand, the area type is also one of the factors that affect the rental price. This is because carpet area type that calculates net usable area focuses more on the area that add values to the house like bedroom and bathroom. Moreover, the overall rental price of houses that target the family are slightly higher than the houses that target bachelor. This is because the bachelors still do not have a stable income thus, they cannot afford expensive house. Meanwhile, the contact person also has an impact on the rental price. For instance, the overall rental price of houses that has agent as contact person is slightly higher than the rental houses that has owner as contact person. This is because the agent earns certain percentage of commission fees so the rental price will be slightly higher. Total floor of the house also have positive impact on the rental price as higher total floor usually indicates the type of house is luxury. For example, luxury apartments usually have around 60 floors with more facilities prepared so that the quality of life is ensured. Difference in floor have less or no impact on the rental price as lift and elevator are usually prepared in apartments.

4.2 Question 2: The characteristics of rental houses that targeting Bachelors

To identify what kind of houses that are preferred by the bachelors in India, this analysis is conducted based on different aspects of the houses' features to explore the preference of bachelors when choosing a house to rent.

4.2.1 Analysis 2-1: The distribution of facilities number of rental houses

This analysis aims to identify the facility numbers distribution of rental houses that target bachelors. The visual used for this analysis is tree map which is popular to visualize the values of each group where the area is proportional to its value.

```
# 2. What kind of houses targeted Bachelor? (only Bachelors)
# Analysis 2-1: Facilities Number
bachelors_df <- df[df$Tenant.Targeted == "Bachelors", ]

# group data and do aggregation
facility_dist_bachelor <-
  bachelors_df %>%
    group_by(Facilities.Number) %>%
    summarize(frequency = n()) %>%
    mutate(percent = calculate_percent(frequency))

# tree map
facility_dist_bachelor %>%
  mutate(Facilities.Number = as.character(Facilities.Number)) %>%
    ggplot(
      mapping = aes(
        area = frequency,
        fill = Facilities.Number,
        label = paste("Number of Facilities: ", Facilities.Number, "\n", percent, "%", sep = "")
      )
    ) +
    geom_treemap() +
    geom_treemap_text(size = 10, colour = "black", place = "centre") +
    scale_fill_brewer(palette = "Spectral")
```

Figure 4-30: Code to Plot Tree Map

The code above partition the houses data that targets bachelors only out from the original dataset so that analysis on bachelor group can be conducted. Then, the data is grouped by the facility number of houses which will be used to count the frequency of each facility number group. To ease the visualization of portion to the group of each category, the percentage that each group occupies is also calculated which will be included in the treemap. Treemap is plotted with "geom_treemap()" function and "geom_treemap_text()" function for adding customized labels after passing in all the necessary arguments. For the colour palette of this

treemap, RColorBrewer package is used to provide a colour palette to the plot function instead of using plain default colour palette.

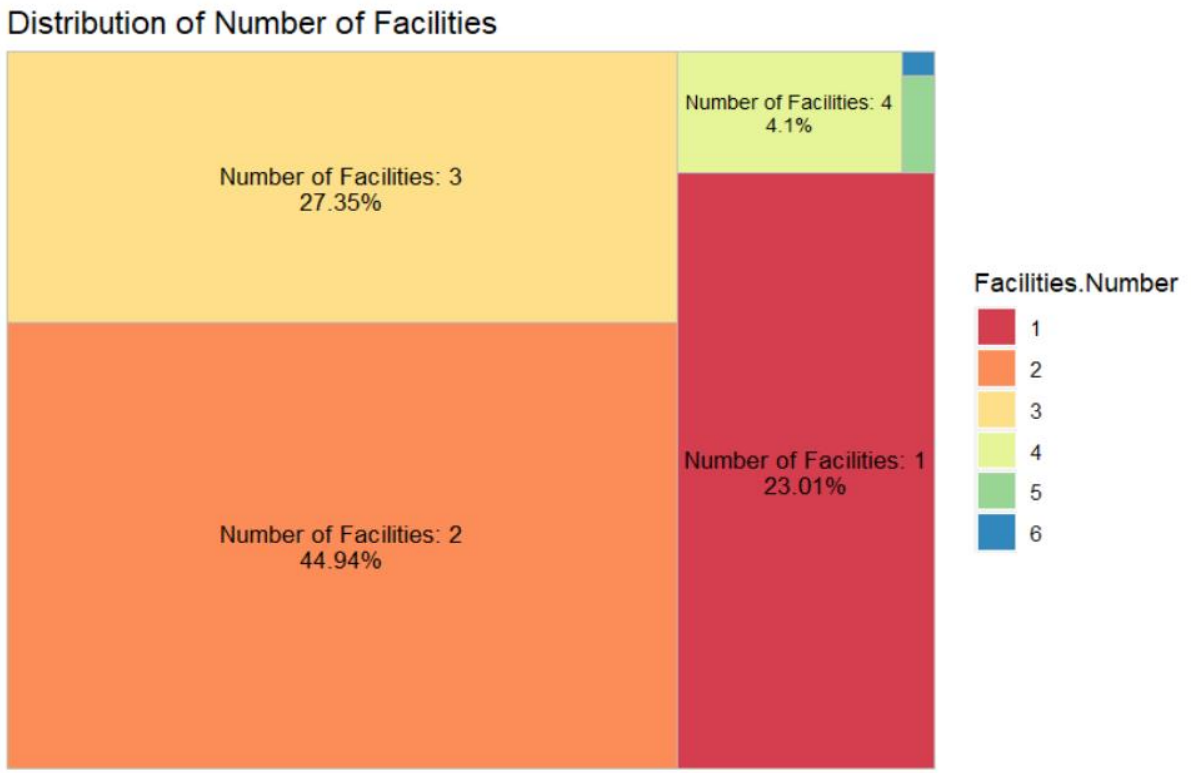


Figure 4-31: Distribution of Facility Numbers among Houses Targeting Bachelors

According to the tree map above, among the houses that target bachelors, houses with 2 facilities is the most popular followed by houses with 3 facilities then the rest. The facilities here include bedroom, hall, and kitchen. As bachelors usually rent the house themselves or with 1 or 2 friends, so higher number of facilities provided does not attract that much so the demand for higher number of facilities is not that high. Therefore, the houses with 1 to 3 facilities is the most preferred house among bachelors based on the dataset given. Houses with more than 4 facilities are not popular among bachelors as the need for hall and kitchen are lesser as compared to other group of target market.

4.2.2 Analysis 2-2: The house size distribution of rental houses

This analysis provides insight on the common house size for houses that targets bachelors as their preferred tenants. The plot used to achieve this is the histogram plot as it can effectively visualize the distribution of a continuous variable such as house size.

```r
# Analysis 2-2: House Size Distribution
bachelors_df %>%
  ggplot(mapping = aes(x = House.Size, fill = after_stat(count))) +
  labs(title = "Distribution of House Size targeting Bachelors", x = "House Size", y = "Frequency", fill = "Frequency") +
  scale_fill_gradient(low = "lightblue", high = "royalblue1") +
  geom_histogram()
```

Figure 4-32: Code to Plot House Size Distribution Histogram

The code above plot the house size distribution histogram with “ggplot()” and “geom_histogram()” functions after passing all arguments. The “after_stat()” function calculate the frequency count when plotting the histogram so there is no need for an extra variable when doing the data preparation.

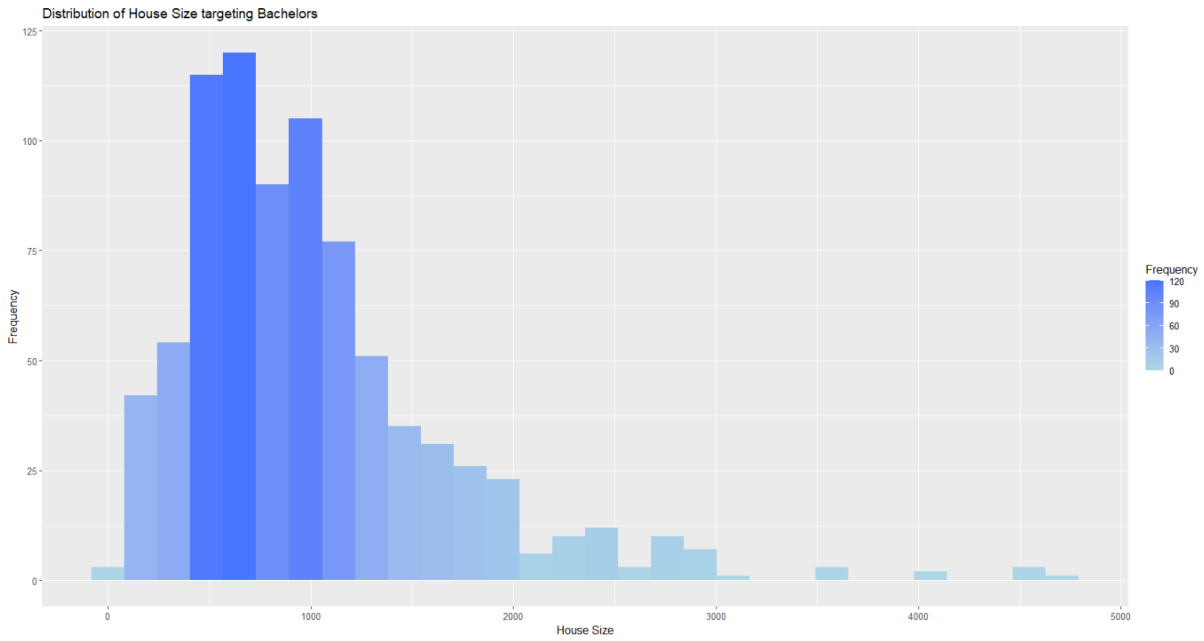


Figure 4-33: House Size Distribution for Houses Targeting Bachelors

Based on the histogram above, the data distribution for house size is skewed to the right as the most instances belong to the left side of the graph where the mean, median, and mode located at. This means that most houses that target bachelors have the house size between 500 and 1200 unit as bachelors does not require bigger house unit for living. This is because bachelors only comes to the city to rent a house for studying so they at most rent a house with their friends which does not require bigger house. To conclude, the bachelors does not need a big house for renting as there is no need for renting bigger house when the people living the house is few.

4.2.3 Analysis 2-3: The rental house distribution in each city

The analysis aims to discover the distribution of rent house posting posted in each city that targets bachelor as their preferred tenant. This is achieved by visualizing the distribution using pie chart as it can show the percentage of each city category immediately. The data is grouped by city first then the frequency of the posting is counted, and percentage count of each category is calculated for visualization.

```
# Analysis 2-2: House Size Distribution
hist(bachelors_df$House.Size, main = "Distribution of House Size Targeting Bachelor", xlab = "House Size")

# Analysis 2-3: City
city_dist_bachelor <-
  bachelors_df %>%
  group_by(House.City) %>%
  summarize(frequency = n()) %>%
  mutate(percent = calculate_percent(frequency))

pie(
  city_dist_bachelor$frequency,
  paste(unique(city_dist_bachelor$percent), "% - ", unique(city_dist_bachelor$frequency), sep = ""),
  main = "Distribution of Rental Houses in each City Targeting Bachelor",
  col = brewer.pal(length(unique(city_dist_bachelor$percent)), "Set3"),
  border = NA
)
legend(
  1.25, 1,
  title="City",
  legend = unique(city_dist_bachelor$House.City),
  cex = .8,
  fill = brewer.pal(length(unique(city_dist_bachelor$percent)), "Set3")
)
```

Figure 4-34: Code to Plot the Pie Chart

The code above construct the pie chart after grouping the data by city and calculate the frequency count and its respective percentage. Using “pie()” and “legend()” functions, the pie chart is plotted with appropriate labels and legend.

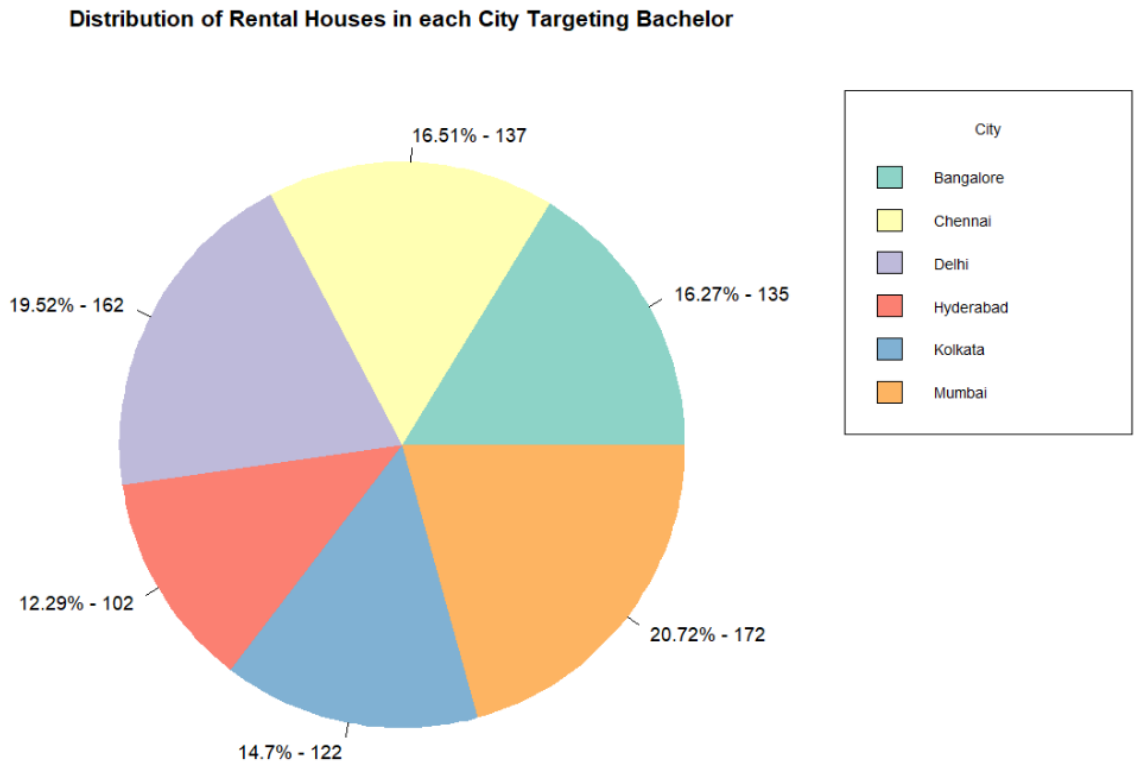


Figure 4-35: Distribution of Rental House in Each City Targeting Bachelors

Based on the pie chart above, the distribution of rental house in each city are balanced where no category has the majority count. Delhi and Mumbai has the highest percentage of distribution of rental house which is respectively 19.52% and 20.72%. This is because Delhi has one of the most popular universities, Delhi University, that provides world-class tertiary education that attracts all the bachelors around India (Mishra R. , 2022). Therefore, students outside Delhi wants to rent a house to study at the university which increases the demand for rental houses in Delhi. Mumbai has the highest percentage of distribution as it is one of the most prosperous cities in India so the many people around India wants to enters Mumbai for education and working purposes. The percentage of other cities are similar which can also be deduced that the cities provide similar education level to its people.

4.2.4 Analysis 2-4: The bathroom number distribution of rental houses

This analysis explore the bathroom number distribution of rental houses that target bachelors as its tenant preferred. It is achieved using horizontal bar chart that orders the bathroom number by descending order from up to down. The data is first grouped by the

bathroom number of rental houses first then the frequency count is calculated to compare against each other.

```
# Analysis 2-4: Bathroom Number
grouped_bachelor_bathroom <-
  bachelors_df %>%
  group_by(Bathroom.Number) %>%
  summarize(frequency = n())

ggplot(data = grouped_bachelor_bathroom, mapping = aes(x = Bathroom.Number, y = frequency)) +
  labs(title = "Bathroom Distribution of Houses Targeting Bachelors", x = "Bathroom Number", y = "Frequency") +
  geom_bar(stat = "identity", col = "#27408B", fill = "#4169E1") +
  geom_text(aes(label = frequency), hjust = -0.2) +
  scale_x_continuous(labels = unique(bachelors_df$Bathroom.Number), breaks = unique(bachelors_df$Bathroom.Number)) +
  coord_flip()
```

Figure 4-36: Code to Plot Bathroom Number Distribution Bar Chart

The code above constructs the horizontal bar chart first by grouping the data by bathroom number as category then calculate the frequency count for each category. Then, the data is passed into "ggplot()" and "geom_bar()" functions to construct a bar chart as demonstrated. The "geom_text()" function is implemented to add the frequency count as label beside the bar of each category for readability.

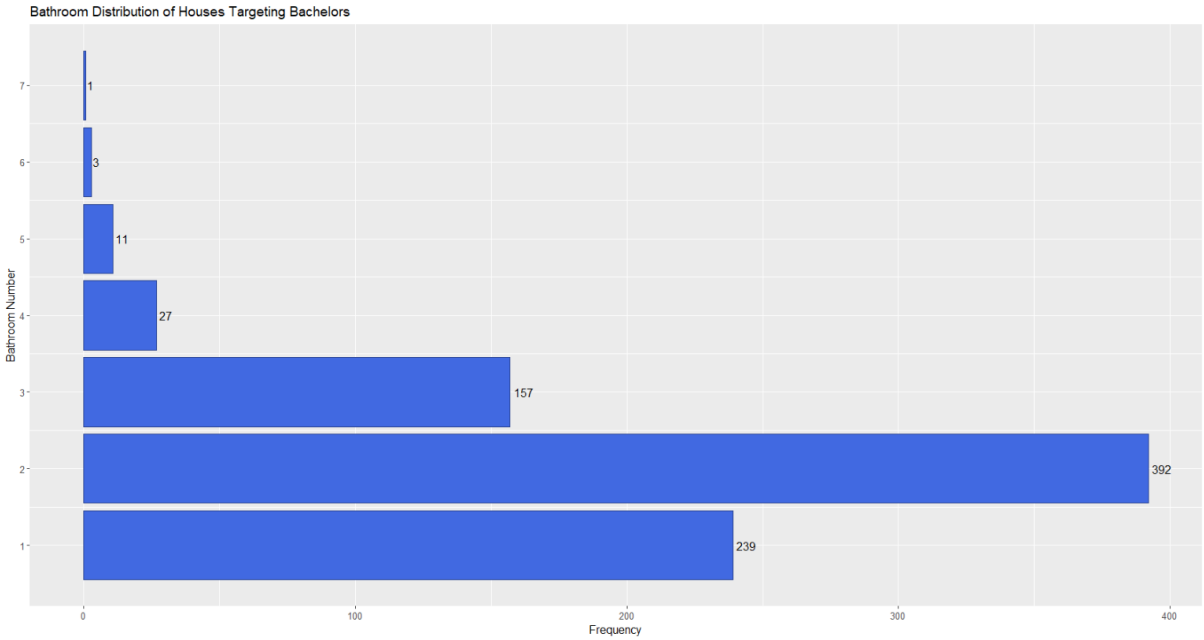


Figure 4-37: Bathroom Number Distribution for Rental House

According to the bar chart above, the rental houses that targets bachelors mostly has 1 to 3 bathrooms prepared. As mentioned before, bachelors usually rent the houses for themselves or with few friends only so more facilities does not add value to the house in the perspective of bachelors. Therefore, the houses that target bachelor mostly has up to 3

bathrooms, but the frequency count drops significantly after 3 bathrooms as there are no demand for more bathrooms in the market of bachelors. To conclude, bachelors does not need high number of bathrooms for their rental house as they only need to live temporarily for their study in university.

4.2.5 Analysis 2-5: The rental price per unit distribution of rental houses

This analysis aims to explore the rental price per unit range that is acceptable and most common for the bachelors. To extract insights for this analysis, histogram is used to visualize the rental price per unit distribution as it can effectively visualize the distribution for a continuous variable like rental price per unit.

```
# Analysis 2-5: Rent per Unit Distribution for Bachelors
bachelors_df %>%
  ggplot(mapping = aes(x = Rent.Per.Unit, fill = after_stat(count))) +
  labs(title = "Distribution of Rental Price per Unit Targeting Bachelors", x = "Rent per Unit", y = "Frequency") +
  scale_fill_gradient(low = "lightgreen", high = "springgreen3") +
  geom_histogram(bins = 20)
```

Figure 4-38: Code to Plot Histogram

The code above constructs the histogram using the bachelor data by passing in all the required arguments. “scale_fill_gradient()” function is used to determine the bar colour based on the value the bar has.

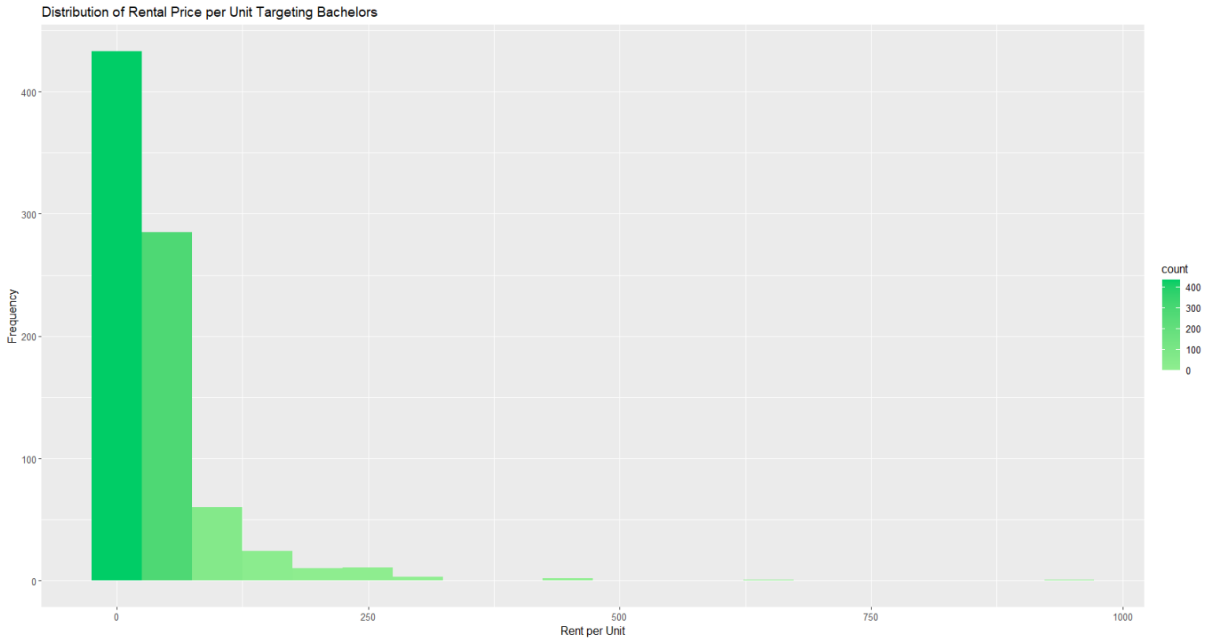


Figure 4-39: Rental Price per Unit Distribution of Rental House with Histogram

According to the histogram, the common rental price per unit of rental houses for bachelors is under 125 Rupee. This is because most bachelors are not yet working as they are studying full-time. This means that they can only afford cheaper rental price due to low and instable income. Therefore, the rental price per unit for houses targeting bachelors are cheaper in general to attract the bachelors to rent their houses. To conclude, the rental price per unit for houses targeting bachelors are lower in overall so that the bachelors can afford them.

4.2.6 Analysis 2-6: The contact person distribution of rental houses

This analysis aims to explore person distribution of the rental houses to identify whether which type of contact person is the most common contact person for bachelors and the preference of people the bachelors want to contact when they are renting a house. To explore the distribution, pie chart is chosen as the visualization tool as it can show the distribution and portion to the whole of categories immediately.

```r
# Analysis 2-6: Contact Person
contact_dist_bachelor <-
  bachelors_df %>%
  group_by(Contact.Person) %>%
  summarize(frequency = n()) %>%
  mutate(percent = calculate_percent(frequency))

pie(
  contact_dist_bachelor$frequency,
  paste(unique(contact_dist_bachelor$percent), "% - ", unique(contact_dist_bachelor$frequency), sep = ""),
  main = "Distribution of Contact Person Targeting Bachelor",
  col = brewer.pal(length(unique(contact_dist_bachelor$Contact.Person)), "Set1"),
  border = NA
)
legend(
  1.25, 1,
  title="Contact Person",
  legend = unique(contact_dist_bachelor$Contact.Person),
  cex = .8,
  fill = brewer.pal(length(unique(contact_dist_bachelor$Contact.Person)), "Set1")
)
```

Figure 4-40: Code to Plot Pie Chart

The code above constructs the pie chart describing the contact person distribution among houses that targeted bachelors as their tenants preferred. The data is first grouped by contact person then the frequency is calculated along with the percentage of each category. With “pie()” and “legend()” functions, the pie chart is plotted by having all the required arguments passed into the functions.

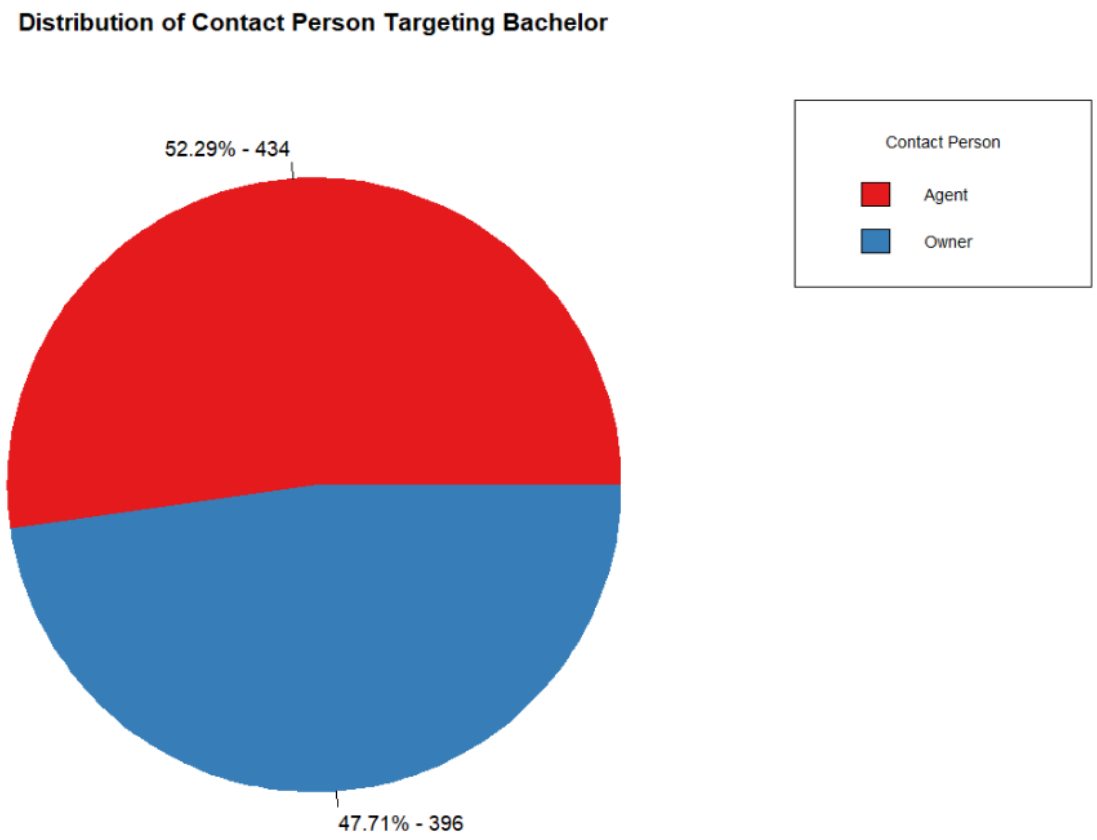


Figure 4-41: Pie Chart of Contact Person Distribution

Based on the pie chart above, the agent as a contact person has higher percentage of distribution as compared to the owner category. This is because the agent has higher exposure as compared to the owner so that the bachelors has higher chance of finding agent to rent a house. Due to this convenience, the bachelors more prefer to look for agent when finding a house to rent. However, this does not mean owner is not a popular choice for bachelor to contact as the difference in frequency count between these two categories is around 5% only. Therefore, the owner is also a popular point of contact for bachelors. To conclude, the frequency count of both these categories are similar so the bachelors does not have favour towards finding an agent or owner when looking for rental house. However, the bachelors does slightly lean towards finding an agent due to the high exposure of agent.

4.2.7 Analysis 2-7: The difference in floor distribution of rental houses

This analysis aims to explore what is the common difference in floor of rental house that targeting bachelors. Density plot is used for this analysis as it can effectively visualize the density and distribution of single continuous value.

```
# Analysis 2-7: Difference in Floor (distribution among houses that targets Bachelors)
bachelors_df %>%
  ggplot(mapping = aes(x = Difference.Floor)) +
  labs(
    title = "Distribution Difference in Floor among Houses Targeting Bachelors",
    x = "Difference in Floor",
    y = "Density"
  ) +
  geom_density(fill = "royalblue2", col = "royalblue1", alpha = 0.5)
```

Figure 4-42: Code to Plot Density Plot

The code above constructs the violin plot for the distribution of difference in floor with “ggplot()” and “geom_density()” functions by passing in the arguments to adjust the properties of the graph such as labels and title.

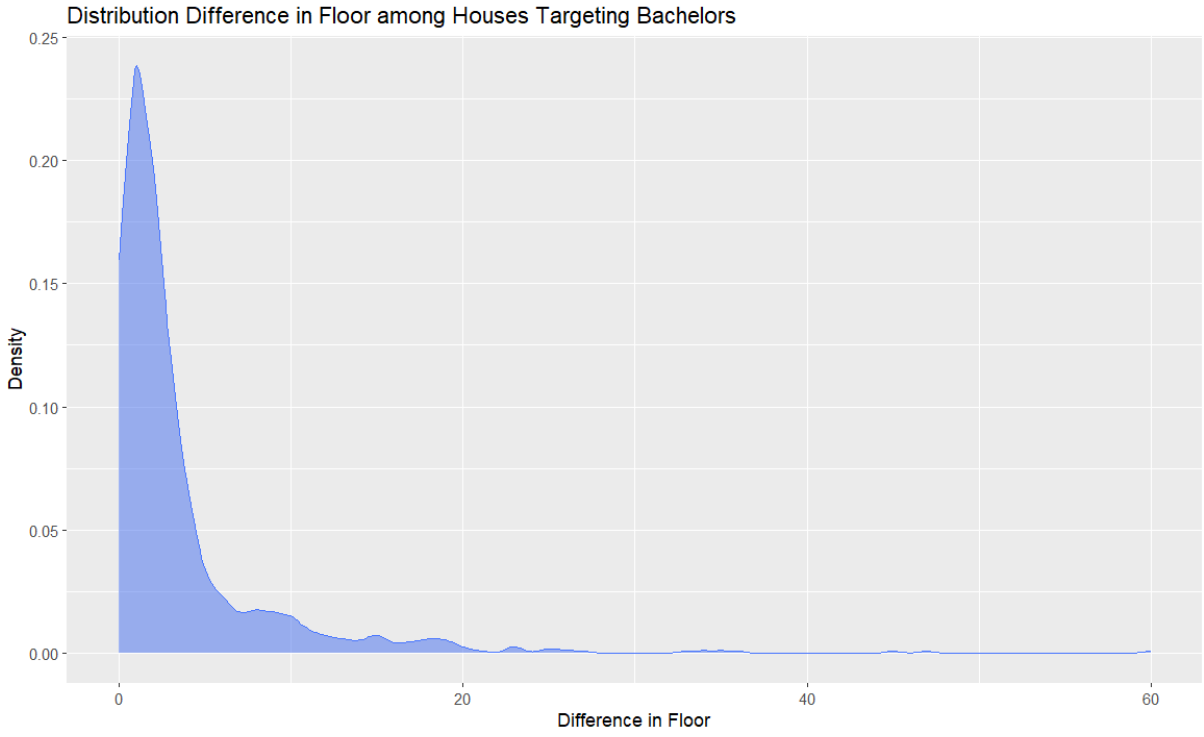


Figure 4-43: Violin Plot to Visualize Distribution of Difference in Floor

Based on the violin plot above, the distribution of difference in floor is peaked at the range between 0 and 8. This means that most houses rented targeting bachelors is single,

double-storey or flat apartment. There are also some houses with difference in floor in range between 5 and 20 which is usually apartment or condominium. This means that most bachelors prefer to live close to the ground floor as it is does not take much time to exit the housing unit.

4.2.8 Analysis 2-8: Total floor distribution of rental houses

This analysis provides insight on the total floor distribution among the houses that targets bachelors as its tenants preferred. Histogram is plotted to visualize the distribution of total floor as it works well in describing the distribution of continuous variable.

```
# Analysis 2-8: Total Floor
bachelors_df %>%
  ggplot(mapping = aes(x = Total.Floor, fill = after_stat(count))) +
  labs(title = "Distribution of Total Floor among Houses Targetting Bachelors", x = "Total Floor", y = "Frequency") +
  scale_fill_gradient(low = "lightsalmon", high = "lightsalmon4") +
  geom_histogram(bins = 20)
```

Figure 4-44: Code to Plot Histogram

The code above constructs the histogram with “ggplot()” and “geom_histogram()” functions after passing in all necessary arguments. “scale_fill_gradient()” function enables the gradient filling of the bar based on the frequency values.

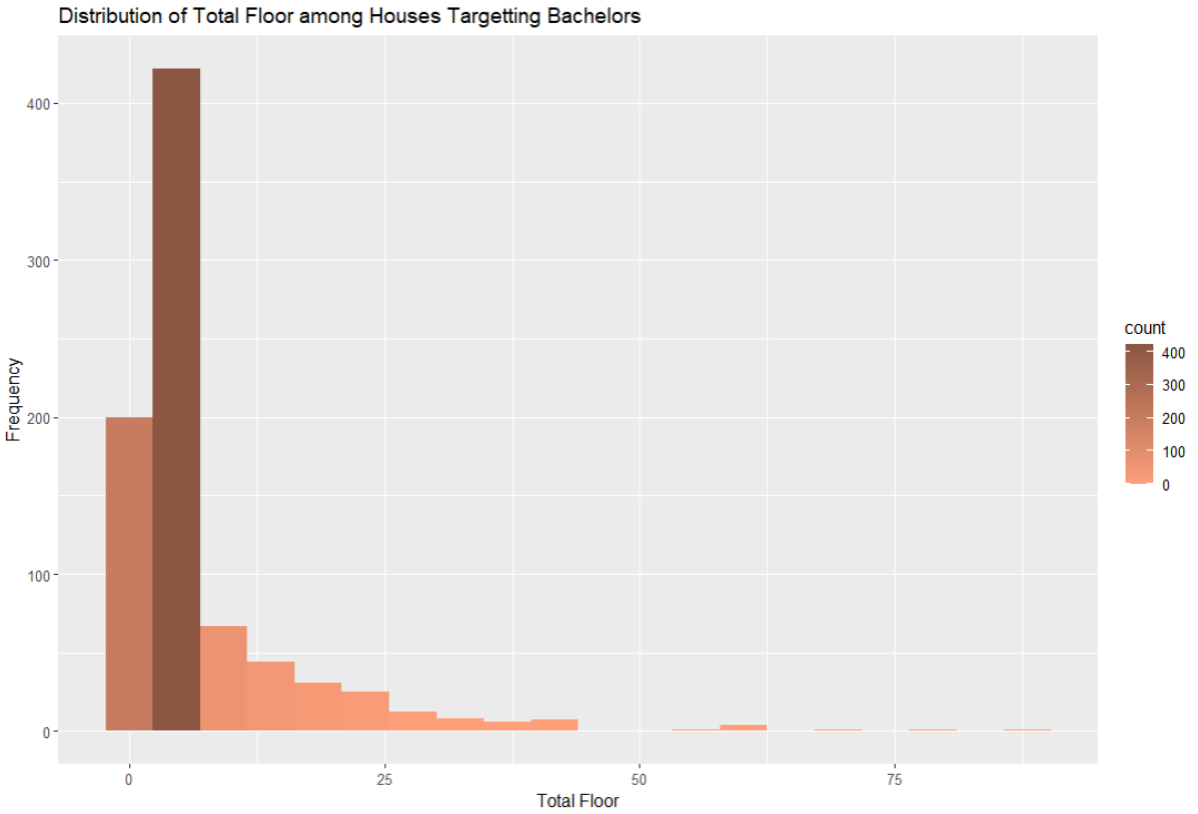


Figure 4-45: Distribution of Total Floor with Histogram

Based on the histogram above, the common total floor of houses that targets bachelors lies between 0 and 10 as the data distribution is skewed to right. This means that the houses that targets bachelors are usually single, double storey house, or units at flat apartment which is relatively cheaper as compared to other luxury apartment that has higher total floor. It can also be said that the bachelors prefer houses with lower total floor so that more houses with lower total floor are posted to target the bachelors group. To conclude, the bachelors prefer houses with lower total floor so that more houses with lower total floor target bachelors. This is also because the bachelors with instable income sources can only afford houses with lower total floor as they are relatively cheaper.

4.2.9 Analysis 2-9: The furnishing status distribution of rental houses

This analysis aims to discover the house furnishing distribution among houses that target bachelors as its tenants preferred. The visual used for this analysis is pie chart that can show the portion of each category to the whole.

```r
# Analysis 2-9: Furnishing Status
grouped_furnishing_bachelor <-
  bachelors_df %>%
    group_by(House.Furnishing) %>%
    summarize(frequency = n()) %>%
    mutate(percent = calculate_percent(frequency))

pie(
  grouped_furnishing_bachelor$frequency,
  paste(unique(grouped_furnishing_bachelor$percent), "% - ", grouped_furnishing_bachelor$frequency, sep = ""),
  main = "Distribution of House Furnishing Targeting Bachelor",
  col = brewer.pal(length(unique(grouped_furnishing_bachelor$House.Furnishing)), "Set3"),
  border = NA
)
legend(
  1.3, 1,
  title="House Furnishing",
  legend = unique(grouped_furnishing_bachelor$House.Furnishing),
  cex = .8,
  fill = brewer.pal(length(unique(grouped_furnishing_bachelor$House.Furnishing)), "Set3")
)
```

Figure 4-46: Code to Plot Pie Chart

The code above first groups the data by furnishing status then calculate the frequency count along with the percentage of the frequency for each category. Then, the pie chart is plotted using “pie()” and “legend()” functions after passing in all the necessary arguments.

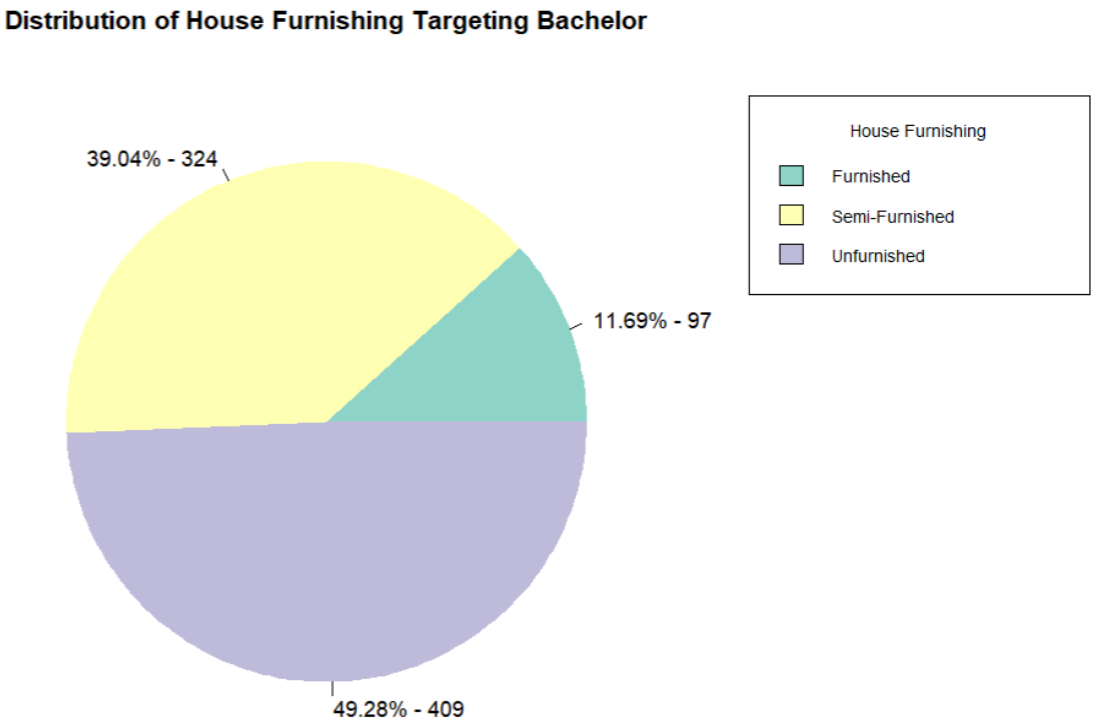


Figure 4-47: House Furnishing Distribution of Houses Targeting Bachelor

Based on the pie chart above, most of the houses that targets bachelors are unfurnished which occupies 49.28% of the entire portion followed by semi-furnished houses then furnished

houses. This is because the bachelors only live in the city temporarily for study, so the bachelors does not necessarily need the house to be furnished. This causes that most rental houses that target bachelors to be unfurnished. Besides, the reasons that why most houses are unfurnished is also because that furnishing a house increases the cost to rent a house. Therefore, most houses are unfurnished so that the renting cost can be reduced as much as possible. There are also few houses that furnishes their house to a certain extent. This is to add more values to the houses so that more tenants can be attracted.

4.2.10 Analysis 2-10: The area type distribution of rental houses

This analysis aims to discover the area type distribution of houses that target bachelors. The visualization used for this analysis is the doughnut chart as it can visualize the portion of each category to the whole effectively.

```r
# Analysis 2-10: Area Type (0 built area type housing for Bachelors)
grouped_area_bachelor <-
  bachelors_df %>%
  group_by(Area.Type) %>%
  summarize(frequency = n()) %>%
  mutate(percent = calculate_percent(frequency)) %>%
  mutate(
    ymax = cumsum(percent),
    ymin = c(0, head(ymax, n = -1)),
    label = paste0(Area.Type, "\nValue: ", frequency, "\nPercentage: ", percent, "%"),
    label_position = (ymax + ymin) / 2
  )

ggplot(data = grouped_area_bachelor, aes(ymax = ymax, ymin = ymin, xmax = 4, xmin = 3, fill = Area.Type)) +
  ggtitle("Distribution of Area Type Percentage Targeting Bachelor") +
  geom_rect() +
  geom_label(x = 3.5, aes(y = label_position, label = label), size = 3, col = "#363636") +
  scale_fill_brewer(palette = "Accent") +
  coord_polar(theta = "y") +
  xlim(c(2, 4)) +
  theme_void() +
  theme(legend.position = "none")
```

Figure 4-48: Code to Plot Pie Chart

The code above constructs the doughnut chart that describes area type distribution of houses that target bachelors. The data is first grouped by area type then the frequency is calculated along with the percentage of the frequency. Afterwards, all the parameters required to plot doughnut chart such as ymax and ymin are calculated. Then, the doughnut chart is plotted with the functions as shown in the image.

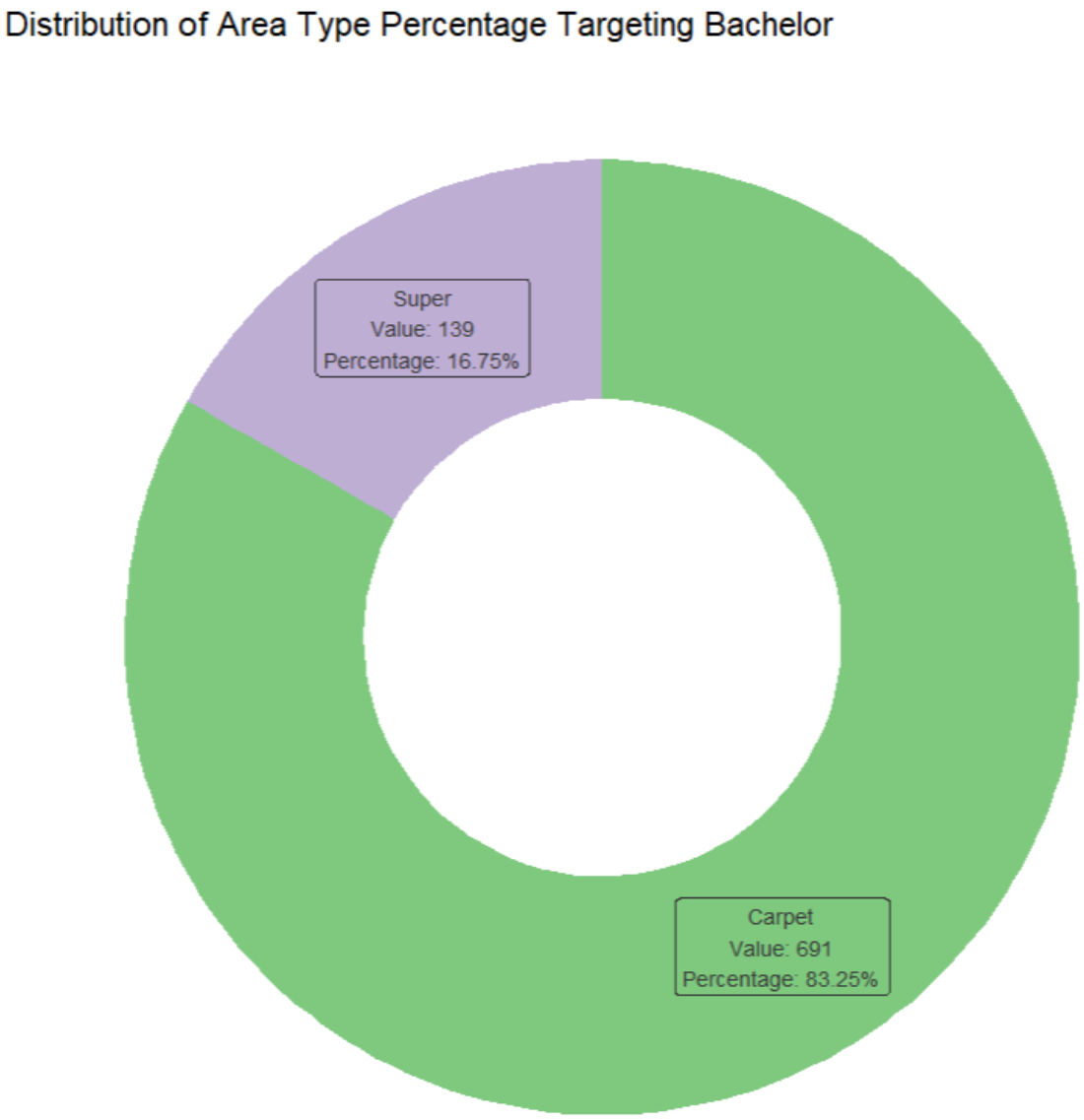


Figure 4-49: Area Type Distribution of Houses Targeting Bachelor

Based on the doughnut chart above, majority of the houses (83.25%) that target bachelors uses carpet area type to calculate the house size while 16.75% of the houses uses super area type to calculate the house size. This is because the carpet area type that only calculates the net usable area are more preferred among bachelors. Therefore, the calculation method of carpet area type that exclude all the public area helps the bachelors to quickly judge the house size and rental price so that quick decision making can be made. To conclude, carpet area type of houses are more popular among houses that target bachelors.

4.2.11 Conclusion

The characteristic of houses that is preferred by the bachelors can be concluded that the bachelors do not prefer houses that has more than 3 facilities and bathrooms. Besides, the house size preferred by bachelors is between 500 and 1200 as bigger house size is not necessary for bachelors because they are living alone or with a few friends only. Mumbai is the most popular city for bachelors as compared to other cities as it is one of the most famous cities in India. The rental price distribution for houses that target bachelor is skewed to right which is overall lower while the most popular contact person is agent. The difference in floor distribution is also low in general while the total floor distribution is ranged between 0 and 10. This means that most bachelors live in row houses or flat house type as those houses has lower total floor. Most of the houses that target bachelors are unfurnished. The most popular area calculation type is the carpet area type as bachelors prefer the net usable area like bedrooms instead of public area like corridor.

4.3 Question 3: Comparison between rental houses targeting Family and Bachelors

4.3.1 Analysis 3-1: Frequency of rental houses by furnishing status for each type of tenant

This analysis aims to provide insights on the house furnishing distribution among houses that targets each type of tenant. This is achieved using percent stacked bar chart that can effectively identify the distribution of a variable in each type of category represented by a bar in the chart.

```
# 3. Compare Houses targeting Family and Bachelors.
# Analysis 3-1: Frequency of Houses by Furnishing Status for Each type of Tenant
grouped_tenant_furnishing <-
  df %>%
  group_by(Tenant.Targeted, House.Furnishing) %>%
  summarize(frequency = n()) %>%
  mutate(percentage = calculate_percent(frequency))

ggplot(data = grouped_tenant_furnishing, mapping = aes(fill = House.Furnishing, x = Tenant.Targeted, y = frequency)) +
  ggtitle("Frequency of Houses by Furnishing Status for Each type of Tenant") +
  xlab("Tenant Targetted") +
  ylab("Percentage") +
  guides(fill = guide_legend(title = "House Furnishing")) +
  geom_bar(stat = "identity", position = "fill", width = 0.4) +
  geom_text(aes(label = paste0(percentage, "%")), size = 3, position = position_fill(vjust = 0.5)) +
  scale_y_continuous(labels = percent)
```

Figure 4-50: Code to Plot Percent Stacked Bar Chart

The code above groups the data by the tenant targeted and the house furnishing then calculate its frequency count. Using “ggplot()” and “geom_bar()” functions, the data is plotted on the graph as stacked percent bar chart as the position argument in “geom_bar()” function is set to “fill”.

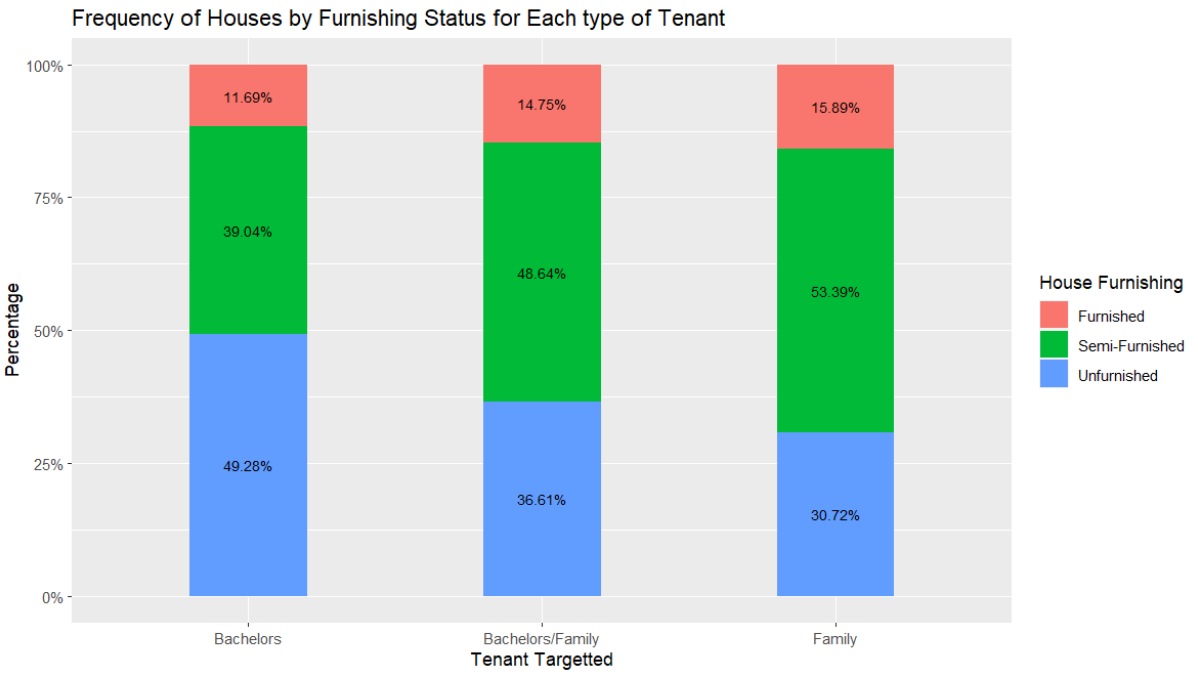


Figure 4-51: Percent Stacked Bar Chart

Based on the percent stacked bar chart above, the distribution of unfurnished house is slightly decreasing when moving from targeting bachelors to family. The distribution of furnished houses almost remain the similar but the semi-furnished houses increasing when the tenant targeted is family group. This is because as family groups are targeted the landlord of the houses needs to furnished or semi-furnished the house so that the family groups can be attracted. As family group is living for a longer period as compared to the bachelors, so they do care about the furnishing status of the house because it will impact their quality of life. Therefore, the landlords are furnishing or semi-furnishing their houses so that the family group can be targeted. For bachelors' groups, they only live in the city renting a house for temporary so that the furnishing status will not be taken into priority.

4.3.2 Analysis 3-2: Median of rental price for tenants in each city

This analysis aims to provide insights on the rental price of houses that targets each type of tenants in every city. Using a grouped bar chart, the median of rental price can be compared against different type of tenants in each city. This is because the grouped bar chart can effectively visualize the difference in values among category and sub-category.

```r
# Analysis 3-2: Median of Rental Price for each type of Tenant in each city
grouped_city_tenant <-
  df %>%
  group_by(House.City, Tenant.Targeted) %>%
  summarize(Median.Rental.Price = median(Rental.Price))

ggplot(data = grouped_city_tenant, mapping = aes(fill = Tenant.Targeted, x = House.City, y = Median.Rental.Price)) +
  ggtitle("Meidan of Rental Price for Each Type of Tenant in each City") +
  xlab("City") +
  ylab("Median of Rental Price") +
  geom_bar(stat = "identity", position = "dodge", width = 0.5, col = "#424242")
```

Figure 4-52: Code to Plot Grouped Bar Chart

The code above groups the data by the city and type of tenant then calculates the median of aggregated rental price. Then, the computed values will be fed into "ggplot()" and "geom_bar()" functions to plot a grouped bar chart where the sub-categories is differentiated by colours using "fill" argument.

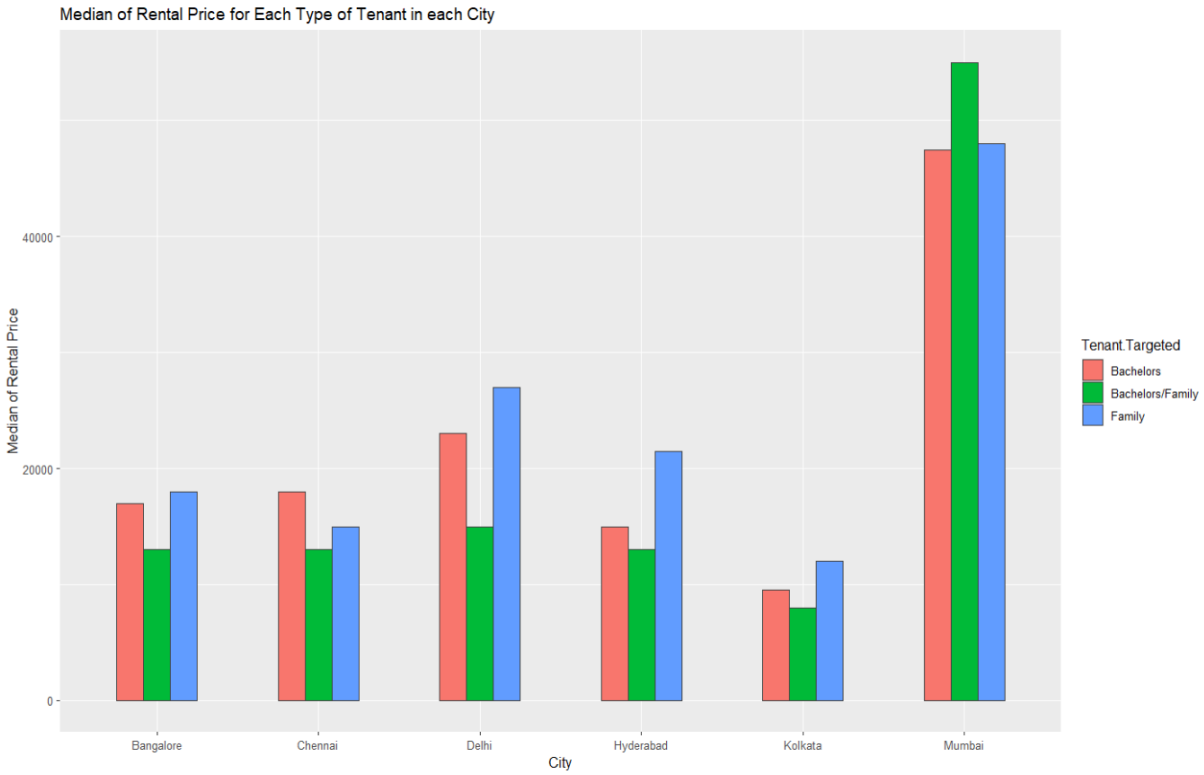


Figure 4-53: Rental Price for Each Type of Tenant in Each City

Based on the grouped bar chart above, in most cities, the rental price of houses target family are overall higher than houses that target bachelors. This is because a family usually requires bigger house size to house more people as family group usually has more residents. This causes the rental price of the house to rises as the house size has a positive impact on the rental price as mentioned in earlier analysis. Therefore, the rental price of houses that targets family are usually slightly higher than that of houses that target bachelors. For houses that

target “Bachelors/Family”, in most cities, the rental price is the lowest except for Mumbai. This might be due to the lack of unique characteristics that target specific group of market means that the houses is appropriate for all type of tenants which can reduce the demand. This is because the houses do not have characteristics that can attract a specific group. In Mumbai, the population is dense thus the rental price of houses that target every type of tenants is higher than the other cities.

4.3.3 Analysis 3-3: Area type for tenants in each city

This analysis aims to explore the area type distribution according to the tenant targeted in each city. The visuals used in this analysis is percent stacked bar chart that can effectively visualize the distribution of area type categorized by the tenant targeted in each city separated by facet.

```
# Analysis 3-3: Different Area Type of Houses Targeting Each Type of Tenant in Each City
grouped_area_city <-
  df %>%
  group_by(House.City, Tenant.Targeted, Area.Type) %>%
  summarize(frequency = n()) %>%
  mutate(percentage = calculate_percent(frequency))

ggplot(data = grouped_area_city, mapping = aes(fill = Area.Type, x = Tenant.Targeted, y = frequency)) +
  ggtitle("Different Area Type of Houses Targeting Each Type of Tenant in Each City") +
  xlab("Area Type") +
  ylab("Frequency") +
  geom_bar(stat = "identity", position = "fill", width = 0.5, col = "#404040") +
  geom_text(aes(label = paste0(percentage, "%")), size = 3, position = position_fill(vjust = 0.5)) +
  scale_y_continuous(labels = percent) +
  facet_wrap(~House.City)
```

Figure 4-54: Code to Plot Percent Stacked Bar Chart Separated by Facets

The code above groups the data by city, tenant, and area type then the frequency count and the percentage is calculated. Afterwards, the aggregated data is used to plot the percent stacked bar chart separated in different facets using “ggplot()” and “geom_bar()” functions.

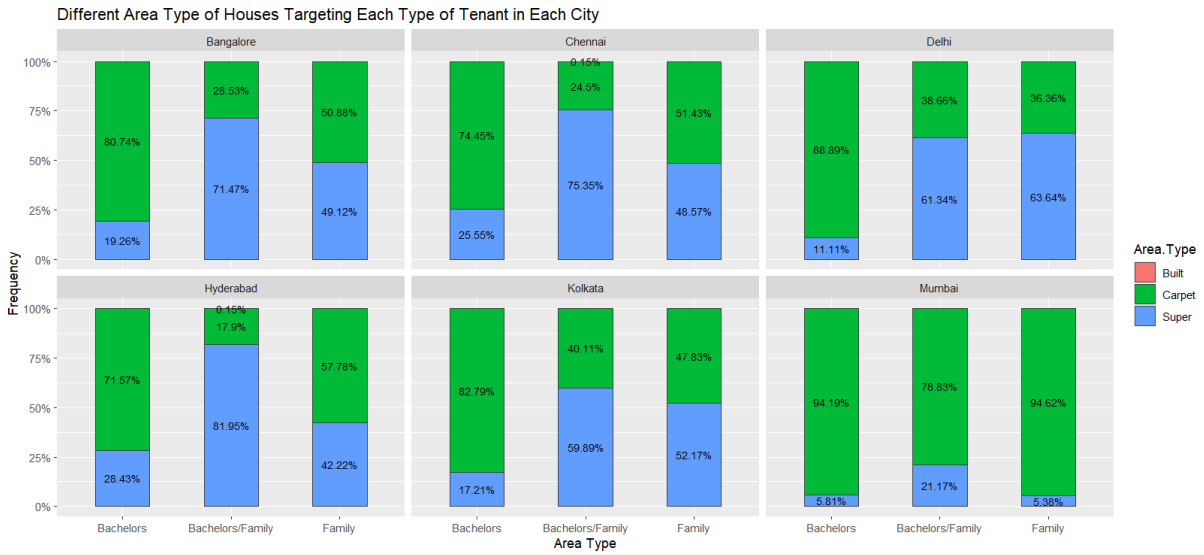


Figure 4-55: Area Type Distribution by Tenant Targeted

Based on the chart above, for houses that target bachelors, the carpet area type of house size calculation method is more popular in all cities. In most cities, most houses that target both bachelors and family uses super area type to measure house size. This might be due to that family group prefer area size that includes more public areas such as corridor and elevator. However, this might also a mean from the landlord to exaggerate the house size so that a greater deal can be made. For family targeted houses, the area type implemented is balanced but super area type is slightly more common than the carpet area type. Regardless of type of tenant, carpet area type is commonly implemented to measure the house size in Mumbai. This is because Mumbai as one of the most prosperous cities, so the people cares a lot when renting a house as the rental price is much higher than other cities. Therefore, carpet area type that calculate net usable area is popular among people in Mumbai.

4.3.4 Analysis 3-4: Distribution of rental price by facility numbers targeting each type of tenants

This analysis aims to provide insights on the rental price distribution of houses that targets each type of tenant. The visualizations graph to explore the rental price distribution is boxplot that is separated by facets according to each type of tenant. The fill colour of boxplot is defined based on the value of the facility numbers so that the audience can differentiate the boxplot by each facility number easily.

```
# Analysis 3-4: The distribution of rental price by facility number for each type of tenant
df %>%
  ggplot(mapping = aes(x = Facilities.Number, y = Rental.Price, fill = Facilities.Number, group = Facilities.Number)) +
    geom_boxplot() +
    labs(
      title = "Rental Price Distribution by Facility Number According To Tenant Type",
      x = "Facility Number",
      y = "Rental Price",
      fill = "Facility Number"
    ) +
    scale_x_continuous(labels = 1:6, breaks = 1:6) +
    scale_fill_viridis_c(option = "magma") +
    facet_wrap(~Tenant.Targeted)
```

Figure 4-56: Code to Plot Rental Price Distribution by Facility Number According to Tenant

The code above plot the boxplot with “ggplot()” and “geom_boxplot()” functions where the boxplot’s fill colour is determined by the value of facility numbers. The x-axis label is modified with sequence from 1 to 6 so that all facility numbers can be displayed on the plot. Then, the boxplot is separated to different facets with “facet_wrap()” function according to the type of tenants targeted.

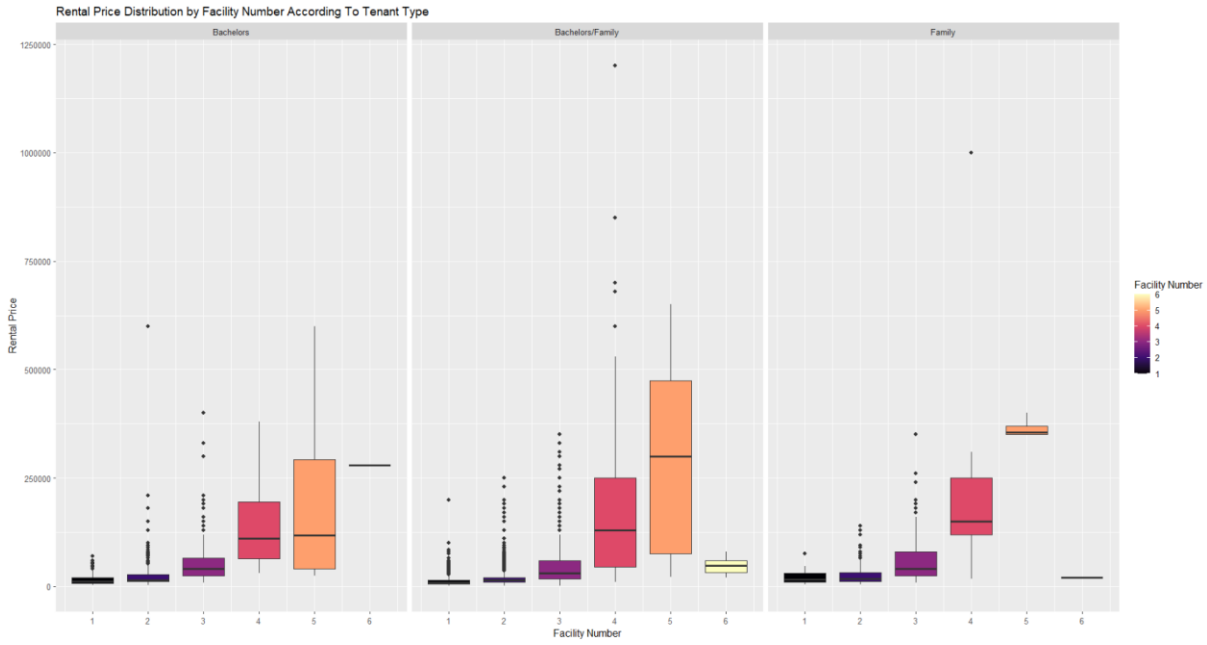


Figure 4-57: Boxplot for Rental Distribution According to Tenant

Based on the boxplot above, the rental price distribution for houses that target bachelor group increases overall when the facility number increases. For family group, the rental price distribution also increase in general when the facility number increases but the price range for bachelors and family targeted houses are different. Up to 3 facility numbers, the price range in general is almost similar but the price range for family targeted houses is slightly higher than bachelor targeted houses. Reaching 5 facility numbers, the price range for bachelor targeted houses has increased overall but the range variety is also extended which means the range is

getting larger. However, for family targeted houses, the price range of houses with 5 facilities is overall expensive than the houses with lesser facilities where the range is also narrowed down. To conclude, the rental price distribution for both type of tenants are increasing when the number of facilities increases but the overall price range for family tenants are higher than bachelor tenants.

4.3.5 Analysis 3-5: The difference in floor distribution for each type of tenants

This analysis discovers the distribution of difference in floor for houses by each type of tenants targeted. The visualization used for this analysis is the histogram separated by facet according to the type of tenant. This is because the histogram can effectively visualize the distribution of a continuous variable.

```
# Analysis 3-5: the common difference in floor for each type of tenants
ggplot(data = df, mapping = aes(x = Difference.Floor, fill = after_stat(count))) +
  labs(
    title = "Distribution of difference in floor for each type of tenants",
    x = "Difference in Floor",
    y = "Frequency"
  ) +
  geom_histogram(bins = 30) +
  scale_fill_distiller(palette = "PuOr") +
  facet_wrap(~Tenant.Targeted)
```

Figure 4-58: Code to Plot Difference in Floor Distribution

The code above plot the histogram of difference in floor distribution where the fill of the bar is determined by the count of the occurrences using “ggplot()” and “geom_histogram()” functions. “scale_fill_distiller()” function is used to change the gradient fill colour of the bar by setting a palette from RColorBrewer library. Then, the plot is separated into different facets using “facet_wrap()” function.

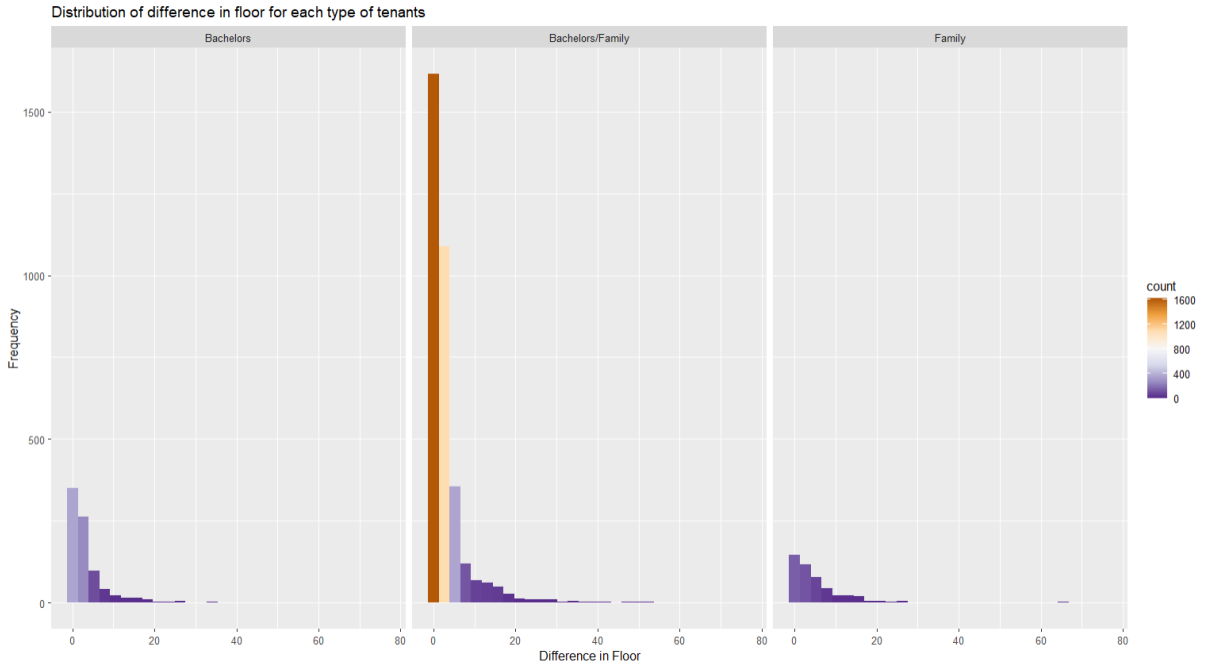


Figure 4-59: Difference in Floor Distribution

According to the plot above, the difference in floor distribution for all 3 types of tenants is similar which is right-skewed distribution. The magnitude of the frequency is different among the types of tenants, but the distribution shape is similar. Therefore, it can be concluded that the difference in floor of houses are overall lower for all types of tenants. This is because the common type of houses in India is flat, apartment, and row houses so the difference in floor will be lower as the houses' total floor is lower as well (Sharma, 2022).

4.3.6 Analysis 3-6: The common total floor for each type of tenants

This analysis aims to find out the most common total floor of houses that target each type of tenants. The visualization graph used for this analysis is the horizontal bar chart that records the frequency of occurrences for each floor group. The graph is separated by facets according to the type of tenants so that the analysis for each type of tenant can be isolated.

```r
# Analysis 3-6: the common total floor for each type of tenants
common_total_floor <- function() {
  final_df <- df
  breaks <- seq(from = -1, to = 90, by = 5)
  labels <- c(
    "0 - 4", "5 - 9",
    "10 - 14", "15 - 19",
    "20 - 24", "25 - 29",
    "30 - 34", "35 - 39",
    "40 - 44", "45 - 49",
    "50 - 54", "55 - 59",
    "60 - 64", "65 - 69",
    "70 - 74", "75 - 79",
    "80 - 84", "85 - 89"
  )

  final_df <-
    df %>%
      mutate(floor_group = cut(x = Total.Floor, labels = labels, breaks = breaks))

  final_df %>%
    group_by(Tenant.Targeted, floor_group) %>%
    summarize(frequency = n()) %>%
    ggplot(mapping = aes(x = floor_group, y = frequency, fill = Tenant.Targeted)) +
      labs(title = "Most Common Floor Group By Tenants", x = "Floor Group", y = "Frequency", fill = "Type of Tenants") +
      geom_bar(stat = "identity") +
      coord_flip() +
      facet_wrap(~Tenant.Targeted)
}
common_total_floor()
```

Figure 4-60: Code to Plot the Most Common Floor Group Chart

The code above categorize each instance into respective floor group using cut function as shown above. Then, the data is grouped by type of tenants and floor group to calculate the occurrence of each group so that the chart can be plotted. With “ggplot()” and “geom_bar()”, the bar chart is plotted with the computed data. “coord_flip()” is used to switch the bar chart into horizontal bar chart while “facet_wrap()” is used to separate the bar chart into different facets by the type of tenants.

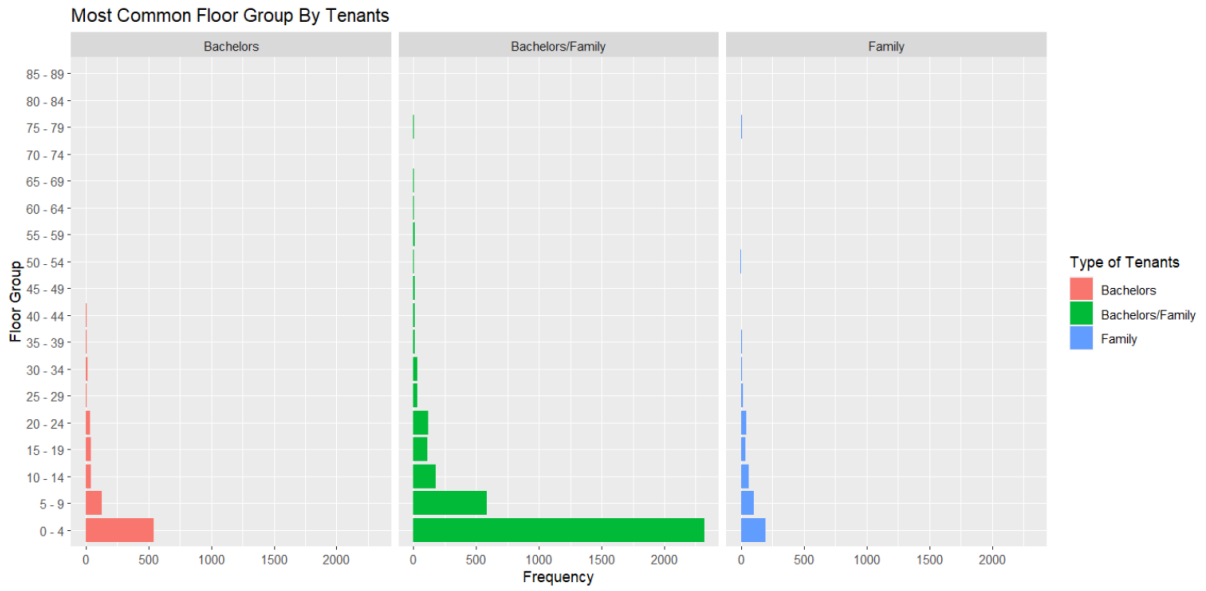


Figure 4-61: Most Common Floor Group by Tenant

Based on the chart above, the most common total floor group for bachelors is “0 – 4” which is usually the flat or row houses. The other two targeted tenants group also have the same common total floor group which indicates that the lower total floor houses such as flat and row houses may be the common type of houses rented in India. Although the most common floor group are the same across different types of tenants, but the distribution of floor group are different. For example, the bachelor group has lesser higher total floor group houses while the family group has a few occurrences. This means that the family group tend to rent luxury apartment that has higher floor. Therefore, to conclude, the total common floor for 3 types of tenants is “0 – 4” where the common house is row houses and flat.

4.3.7 Analysis 3-7: The common house size for each type of tenants

This analysis aims to find out the most common house size provided by houses that target each type of tenants. Violin plot is used for this analysis as it can describe the density and median value of the distribution in one plot which is useful to achieve the goal of this analysis.

```
# Analysis 3-7: For each city, the most common house size for each type of tenants
ggplot(data = df, mapping = aes(x = Tenant.Targeted, y = House.Size, fill = Tenant.Targeted)) +
  labs(title = "House Size Distribution for Each type of Tenants", x = "Tenant Targeted", y = "House Size") +
  geom_violin() +
  stat_summary(fun.y = median, geom = "point", color = "black")
```

Figure 4-62: Code to Plot House Size Distribution with Violin Plot

The code above plot the house size distribution with violin plot using “ggplot()” and “geom_violin()” functions. The median is calculated and plotted in the violin using “stat_summary()” function after adjusting all the parameters.

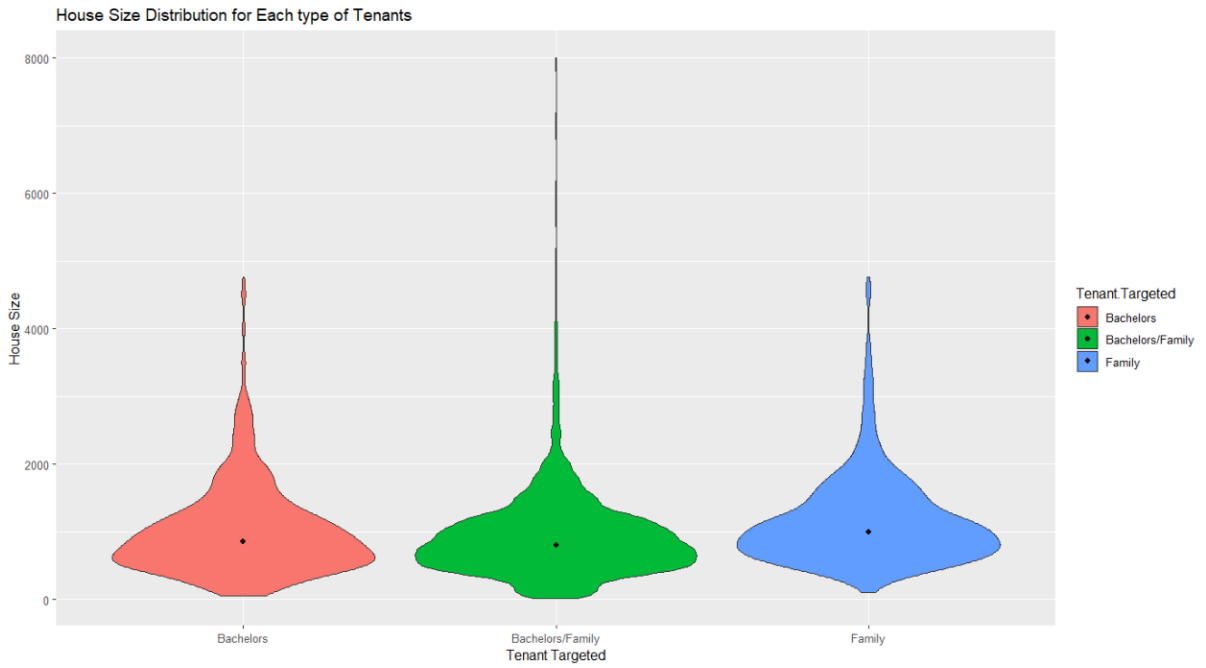


Figure 4-63: House Size Distribution for Tenants According to City

Based on the plot above, the house size of houses for all type tenants is similar but for family group, the house size distribution is slightly larger than that of bachelor group. For houses that target both bachelors and family, the house size distribution has wider range which means it has house size ranging from 0 to 8000. In other words, it has greater varieties in terms of house size as it targets both main groups in the market, so versatility means more chances to attract the customers.

4.3.8 Analysis 3-8: The most common contact person for each type of tenants

This analysis aims to find out the most common contact person by each type of tenants by exploring the distribution of the contact person among tenants. The visualizations for this analysis will be the pie charts that is aligned horizontally for comparison against different type of tenants.

```r
# Analysis 3-8: The most common contact person for each type of tenants
plot_dist_tenant <- function() {
  grouped_city_contact_tenant <-
    df %>%
    group_by(Contact.Person, Tenant.Targeted) %>%
    summarize(frequency = n())

  fig_contact_tenant <- plot_ly()
  fig_contact_tenant <- fig_contact_tenant %>%
    add_pie(
      data = grouped_city_contact_tenant[grouped_city_contact_tenant$Tenant.Targeted == "Bachelors", ],
      name = "Bachelors",
      value = ~frequency,
      labels = ~Contact.Person,
      title = "Distribution of Contact Person for Bachelors",
      domain = list(row = 0, column = 0)
    )

  fig_contact_tenant <- fig_contact_tenant %>%
    add_pie(
      data = grouped_city_contact_tenant[grouped_city_contact_tenant$Tenant.Targeted == "Family", ],
      name = "Family",
      value = ~frequency,
      labels = ~Contact.Person,
      title = "Distribution of Contact Person for Family",
      domain = list(row = 0, column = 1)
    )

  fig_contact_tenant <- fig_contact_tenant %>%
    add_pie(
      data = grouped_city_contact_tenant[grouped_city_contact_tenant$Tenant.Targeted == "Bachelors/Family", ],
      name = "Bachelors/Family",
      value = ~frequency, ===
      labels = ~Contact.Person,
      title = "Distribution of Contact Person for Bachelors/Family",
      domain = list(row = 0, column = 2)
    )

  fig_contact_tenant <- fig_contact_tenant %>% layout(
    title = "Distribution of Contact Person by Tenant Targeted",
    showlegend = TRUE,
    grid = list(rows = 1, columns = 3),
    xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
    yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
    margin = 1
  )
  fig_contact_tenant
}
```

Figure 4-64: Code to Plot Subplot of Pie Charts

The code above first group the data by contact person and type of tenants targeted before plotting. After the data is aggregated, the subplot is instantiated using “plot_ly()” function and the pie chart is plotted and added to the subplot using “add_pie()” function and pipe operator. By adjusting the arguments in “add_pie()” function such as data parameter for the data used to plot pie chart, the pie chart can be customized as desired. The layout of the subplot is adjusted after all pie charts is plotted and added to the subplot using “layout()” function by passing in all the necessary arguments.

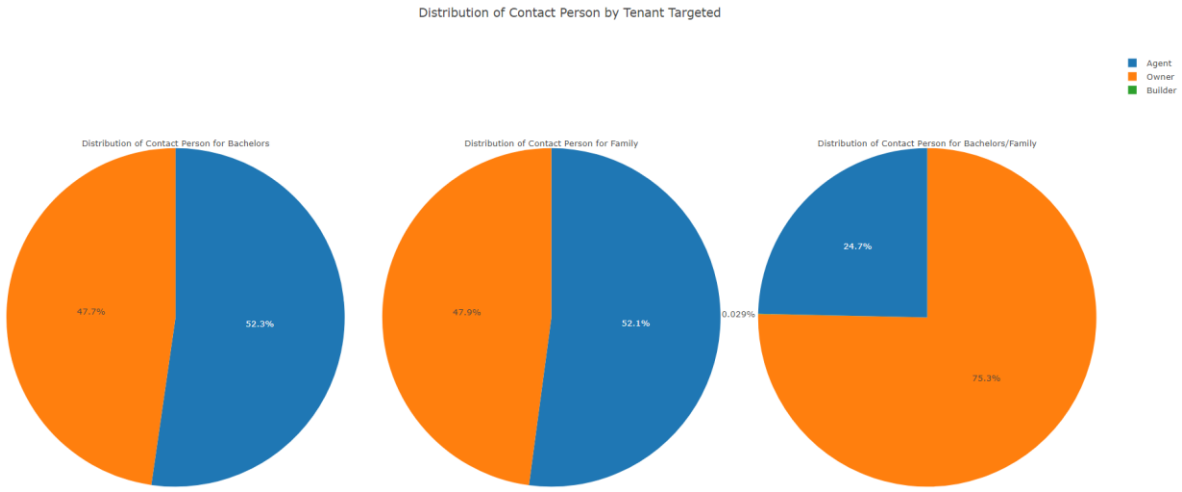


Figure 4-65: Pie Chart Describing Distribution of Contact Person by Tenant Targeted

Based on the pie charts above, the distribution of contact person for bachelors and family are almost the same where both type of contact person are preferred. However, for "Bachelors/Family" tenant type, the owner is overwhelmingly being preferred which has 75.3%. These findings can prove that the most house owners target both bachelor and family without a specific target as they do not understand the market as much as property agent. On the other side, the houses that target specific group of bachelor or family are managed by the property agent and the owners themselves with similar distribution.

4.3.9 Conclusion

After the comparison between the houses targeting family and bachelors, it can be concluded that the furnishing status is important when the family is targeted as the percentage of furnished house targeting family is more than that of bachelors group. For family group in most cities, the rental price of houses is higher than the bachelor group in general as family usually requires bigger house because of larger group of family members. For houses that target both bachelor and family in most cities, the rental price is lowest among types of tenants except for Mumbai because Mumbai's population is dense so rental price of houses will be slightly higher. Bachelors targeted houses commonly use carpet area type to measure house size, while the houses that target both bachelors and family use super area type. For family targeted house, super area type is slightly more common to be used than the carpet area type.

4.4 Question 4: What is the trend of rental houses?

This section aims to examine the trend on the characteristics of rental house posted in the dataset between 13rd April 2022 and 7th November 2022. To gain insights, various analysis is conducted on the features of rental houses such as facilities number, house size, and area type of rental houses.

4.4.1 Analysis 4-1: Facilities number of rental houses

This analysis aims to discover the trend for the number of facilities in the rental house posted between the date period mentioned. The graph selected for this analysis is line chart with two type of lines which is the facility number and the smoothed results to show the trend. The mean value for the number of facilities is calculated grouped by the posted date.

```
# Analysis 4-1: Facility Numbers against Date
df %>%
  group_by(Posted.Date) %>%
  summarize(agg = mean(Facilities.Number)) %>%
  mutate(Smoothed.Agg = rollmean(agg, k = 10, fill = NA, align = "center")) %>%
    ggplot() +
    labs(title = "Trend of the Number of Facilities", x = "Posted Date", y = "Number of Facility", colour = "Lines") +
    geom_line(mapping = aes(x = Posted.Date, y = agg, colour = "Facility Numbers")) +
    geom_line(mapping = aes(x = Posted.Date, y = Smoothed.Agg, colour = "Smoothed Facility Number"))
```

Figure 4-66: Code to Plot Number of Facilities Trend

The code above groups the data by the posted date first then the mean of number of facilities is calculated. To plot the smoothen line, the moving average of the facility numbers are calculated using “rollmean()” function. With “ggplot()” and “geom_line()” functions, line chart is plotted after passing in all the required arguments.

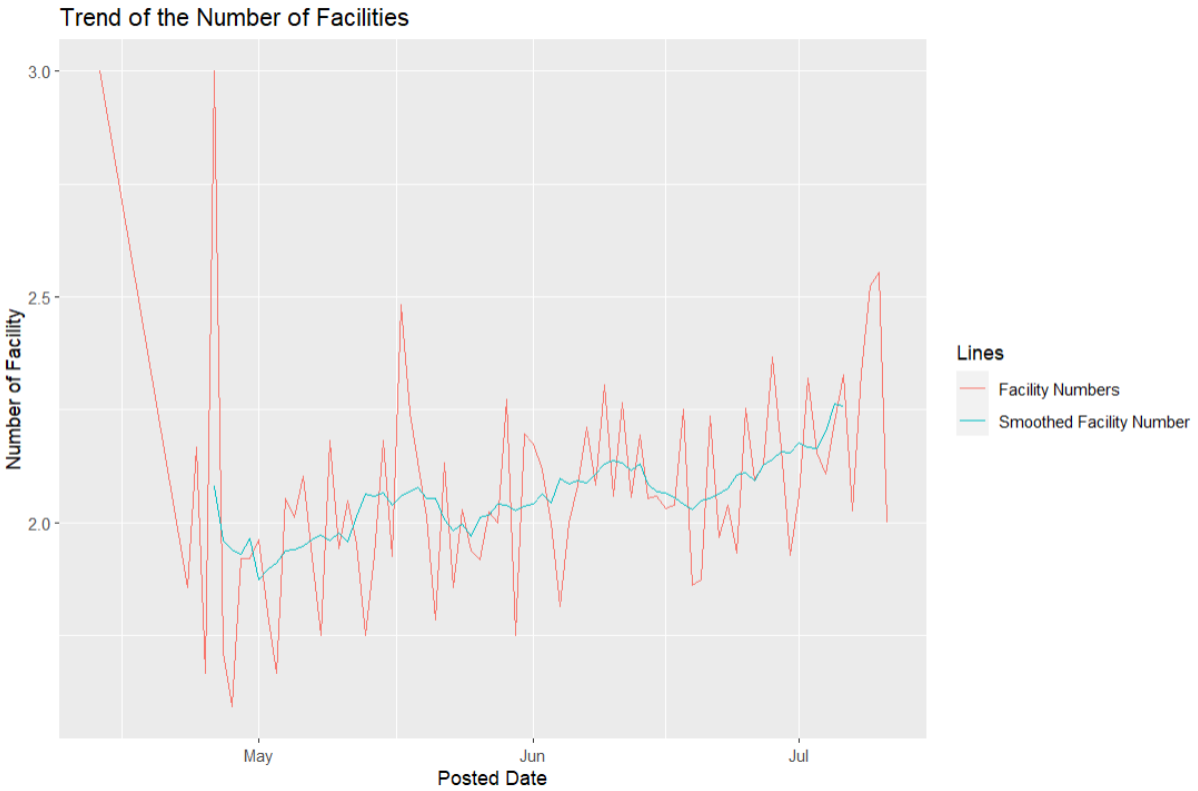


Figure 4-67: Trend of Facilities Number

Based on the plot above, the number of facilities has the trend of slightly increasing between the date of 13rd April 2022 and 7th November 2022. The original line of facility number is fluctuating and difficult to identify the trend based on that. Therefore, smoothed facility number is calculated and plotted on the graph so that the trend of the facility number can be emphasized. This trend indicates that there might be a demand for houses with more facilities as the mean of facilities number of houses are increasing over time.

4.4.2 Analysis 4-2: Size of rental houses

This analysis aims to find out the trend of the rental house's size over the period mentioned. The plot used in this analysis is the scatterplot which is effective to visualize the distribution of the continuous variable. The house size is grouped by each date to get its mean value before plotting because there might be few rental records on the same date. Therefore, mean which is the central tendency of the data is used for the plotting.

```
# Analysis 4-2: House Size against Date
df %>%
  group_by(Posted.Date) %>%
  summarize(agg = mean(House.Size)) %>%
  ggplot(mapping = aes(x = Posted.Date, y = agg)) +
    labs(title = "House Size against Date", x = "Posted Date", y = "House Size") +
    geom_point(col = "#424242")
```

Figure 4-68: Code to Plot Scatterplot

The code above groups the data by posted date then summarize each group of data with the mean value of the house size. Afterwards, “ggplot()” and “geom_point()” functions is used to plot the scatterplot by passing in the necessary requirements to customize the plot.

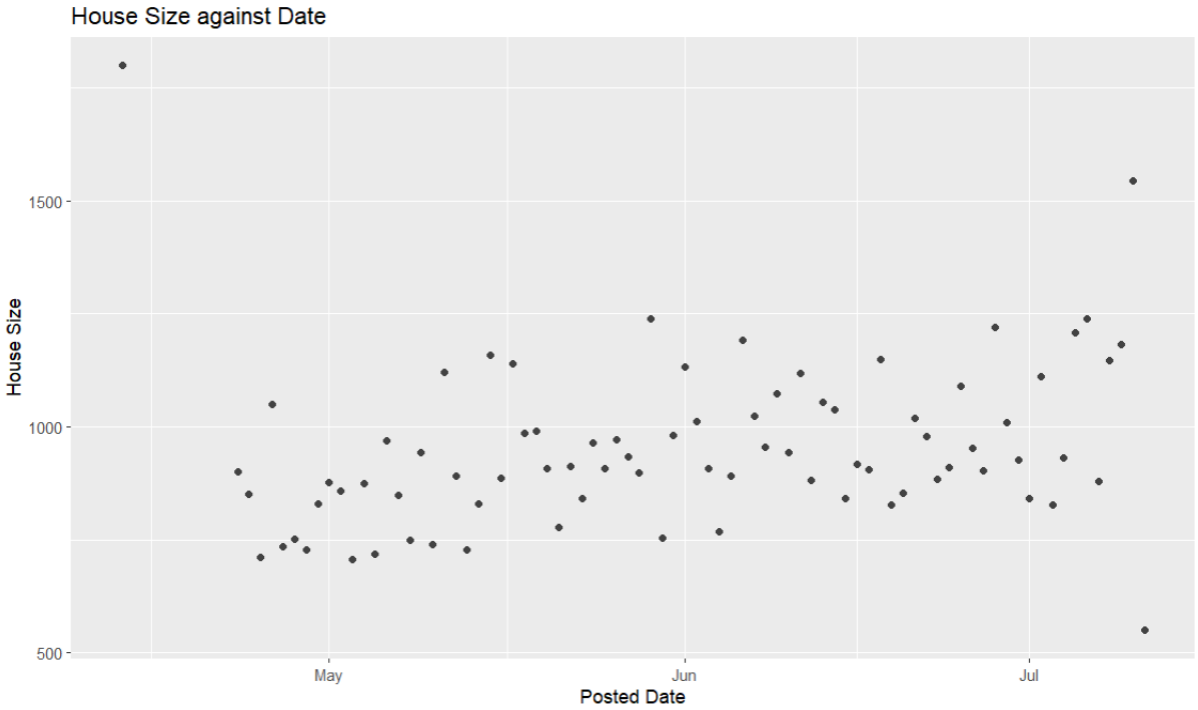


Figure 4-69: House Size Trend with Scatterplot

Based on the scatterplot above, the house size does not have a particular trend over the time as the data point on the plot remains scattered within a range randomly. This may indicate that there is no demand for bigger house to rent in the cities of the dataset, so the house size remains constant over months.

4.4.3 Analysis 4-3: Area type of rental houses

This analysis aims to explore the trend of the area type of houses over months. The visualizations used for this analysis is the line chart that separated in different facets by area type. Line chart is used because it can effectively describe the change in frequency of rental posting over time.

```
# Analysis 4-3: Count of Area Type against Date
df %>%
  group_by(Area.Type, Posted.Date) %>%
  summarize(frequency = n()) %>%
  ggplot(mapping = aes(x = Posted.Date, y = frequency, col = Area.Type)) +
    labs(title = "Trend of Area Type", x = "Posted Date", y = "Frequency") +
    geom_line() +
    facet_wrap(~Area.Type)
```

Figure 4-70: Code to Plot the Line Chart Facets

The code above first groups the data by area type and posted date then the frequency for each group is calculated. With “ggplot()” and “geom_line()” functions, the line chart is plotted where the colour of line is determined by the area type categorical value. The “facet_wrap()” function separate the line chart into different facets by the area type.

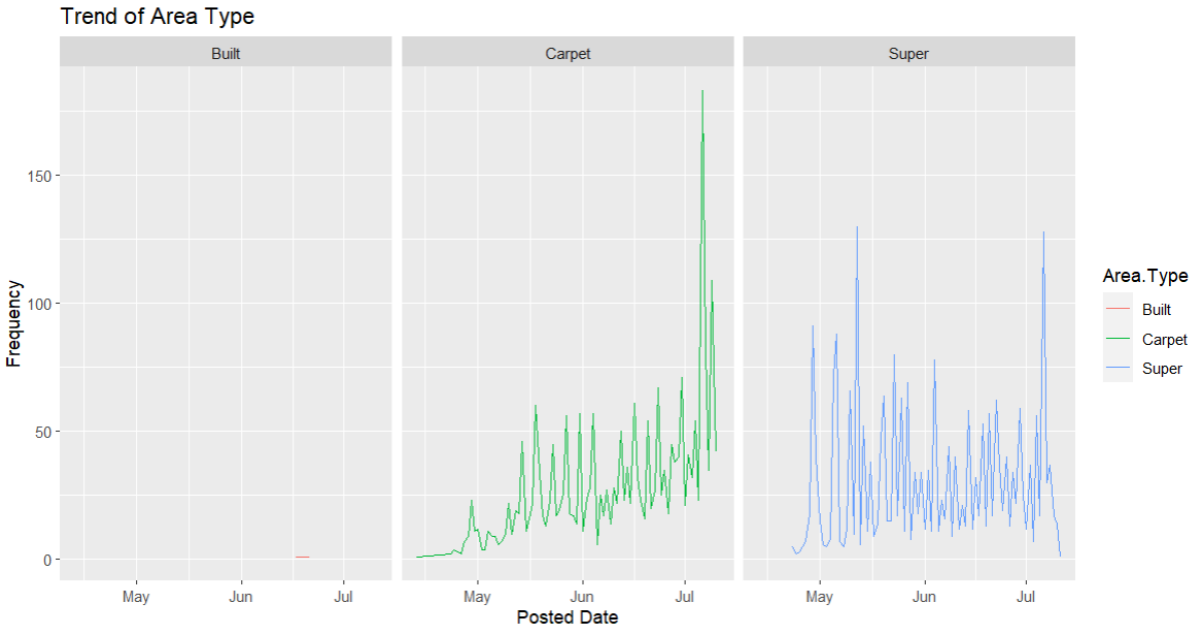


Figure 4-71: Trend of Area Type against Date

Based on the line charts above, the built area type has too few samples to plot a line chart, so the plot is empty. For carpet area type, the occurrences of rental posting has an

increasing trend from May to July. This means more people are towards the carpet area type of method to calculate the house size as it is more precise when comparing with other houses as it calculates the net usable area. Besides, this is also because that the RERA provides protection for carpet area type houses if the promised carpet area is different from the given carpet area then refund will be given (Mishra P. , 2022). The super area type has no trend of increasing or decreasing but remain constant over time. This is because the house developers like to market the projects with numbers calculated with super area type as the calculation includes common area such as playground and corridor (Mishra S. , 2022). Therefore, the trend of using super area type never dies although carpet area type is a method that is more precise.

4.4.4 Analysis 4-4: Posting of rental house by city

This analysis aims to provide insight on the trend of rental house posting in each city over date. The visualization graph used in this analysis is the grouped bar chart that can visualize the value of sub-category groups effectively. Data will be grouped by months first then plotted on the graph with months as the x-axis and the frequency of rental posting as the y-axis. Each city will be filled with different colour so that it can be differentiated easily.

```
# Analysis 4-4: House City against Date
grouped_city_date <-
  df %>%
  group_by(month = lubridate::floor_date(Posted.Date, unit = "month"), House.City) %>%
  summarize(frequency = n())

ggplot(data = grouped_city_date, mapping = aes(x = month, y = frequency, fill = House.City)) +
  labs(title = "Trend of Rental House Posted in City against Date", x = "Month", y = "Frequency of Rental House Posted") +
  guides(fill=guide_legend(title="House City")) +
  geom_bar(stat = "identity", position = "dodge", width = 20, col = "#424242") +
  scale_fill_brewer(palette = "Set2")
```

Figure 4-72: Code to Plot Grouped Bar Chart

The code above groups the data by months and city then calculate the frequency of each group. Then, the data is plotted with “ggplot()” and “geom_bar()” functions by passing in all the arguments to customize the bar chart. The “scale_fill_brewer()” function is used to set the colour palette of the grouped bar in the chart.

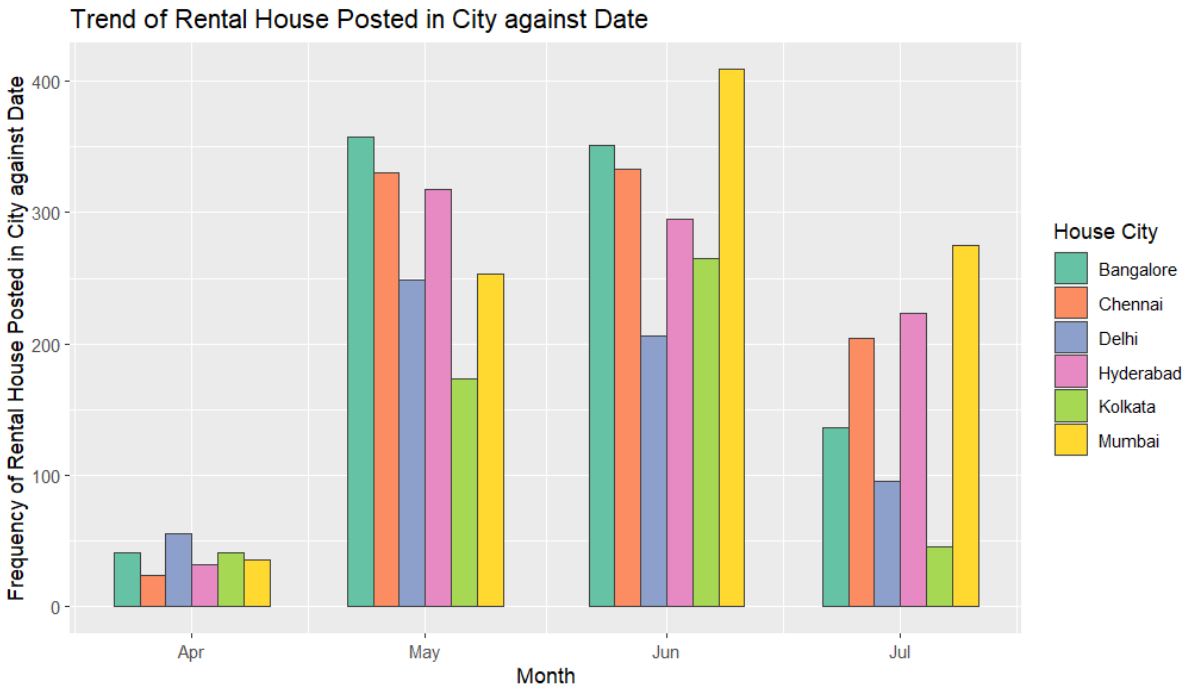


Figure 4-73: Trend of Rental House Posting over Date

Based on the grouped bar chart above, the trend of rental posting in all cities are almost the same as the trend increases until June then decreases in July. In Mumbai, the occurrences of rental posting has the greatest jump from the lowest among the cities to the highest in June. This is because Mumbai has dense population so the demand for rental houses increases significantly. In Bangalore, the increase from April to May also skyrocketed which makes it the top among the cities. Delhi has the lower magnitude in the increase of occurrences of rental posting and has the trend of decreasing the fastest as compared to other cities.

4.4.5 Analysis 4-5: Furnishing status of rental house

This analysis aims to explore the trend of furnishing status of houses over time. The graph selected for this analysis is lollipop chart that is similar to bar chart, but it can effectively visualize the relationship between a categorical and continuous variable even when there are many categories.

```
# Analysis 4-5: House Furnishing against Date (Bar chart group by Furnishing)
df %>%
  group_by(House.Furnishing, Posted.Date) %>%
  summarize(agg = n()) %>%
  ggplot(mapping = aes(x = Posted.Date, y = agg, col = House.Furnishing)) +
    labs(title = "Trend of Rental Posting by House Furnishing", x = "Posted Date", y = "Frequency") +
    geom_point() +
    geom_segment(mapping = aes(x = Posted.Date, xend = Posted.Date, y = 0, yend = agg)) +
    facet_wrap(~House.Furnishing)
```

Figure 4-74: Code to Plot Lollipop Chart

The code above groups the data by house furnishing and posted date then calculate the occurrences of records for each group. The “ggplot()” function is used to construct the basic layout of the plot. Then, “geom_point()” function adds the point on the lollipop chart while “geom_segment()” function adds the line that made up the lollipop. With “facet_wrap()” function, the lollipop charts are separated in different facets by the house furnishing type.

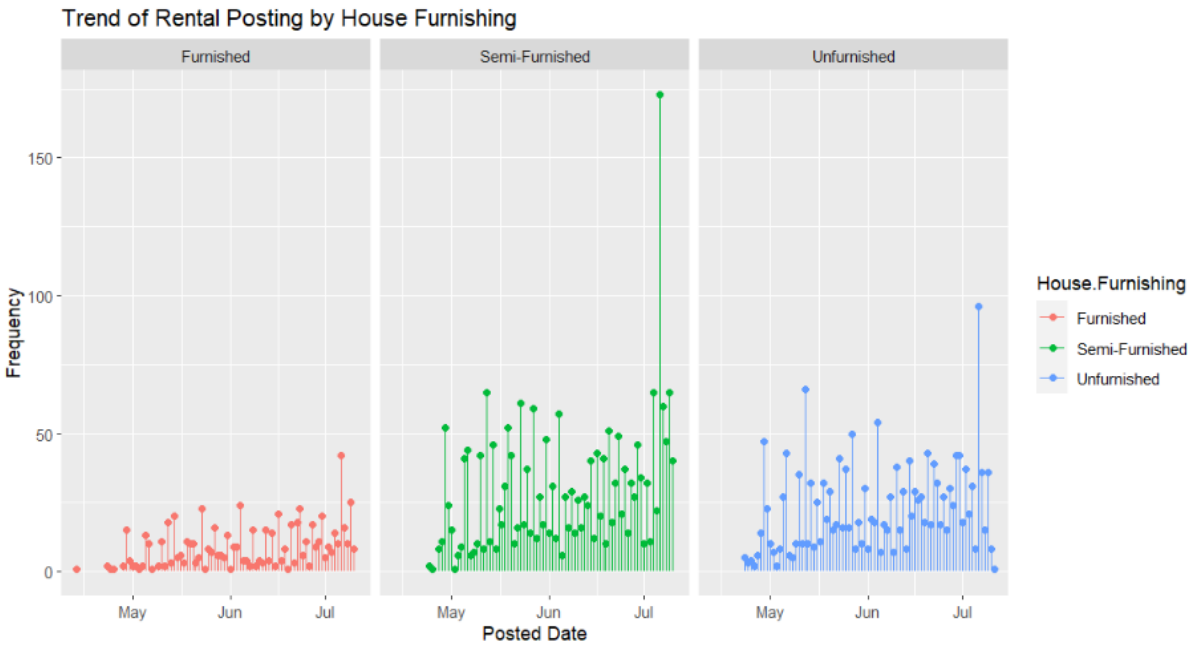


Figure 4-75: Lollipop Chart for Trend of Rental Posting by House Furnishing

Based on the lollipop chart above, the trends for 3 types of house furnishing are similar. This is because all the type of house furnishing show similar trend where the trend remains constant without major increasing or decreasing. In July, there are spikes in increases for all the house furnishing type. To conclude, there are no upward or downward trend for the occurrences of all house furnishing type.

4.4.6 Analysis 4-6: Tenant targeted of rental posting

This analysis aims to explore the trend of the tenant targeted over time. Grouped bar chart is selected for this analysis as it can visualize the value of each sub-categories group easily.

```
# Analysis 4-6: Tenant Targeted against Date
df %>%
  group_by(month = lubridate::floor_date(Posted.Date, unit = "month"), Tenant.Targeted) %>%
  summarize(frequency = n()) %>%
  ggplot(mapping = aes(x = month, y = frequency, fill = Tenant.Targeted)) +
    labs(title = "Trend of Tenant Targeted against Date", x = "Month", y = "Frequency of Tenant Targeted") +
    geom_bar(stat = "identity", position = "dodge", width = 10, col = "#424242") +
    scale_fill_brewer(palette = "Accent")
```

Figure 4-76: Code to Plot Grouped Bar Chart

The code above groups the data by months and type of tenants targeted then calculate the frequency of each group. Then, “ggplot()” and “geom_bar()” functions is used to plot the grouped bar chart after passing in all the arguments to customize the chart. Then, the “scale_fill_brewer()” function is used to set the colour palette used for the fill colour of bar of each sub-category.

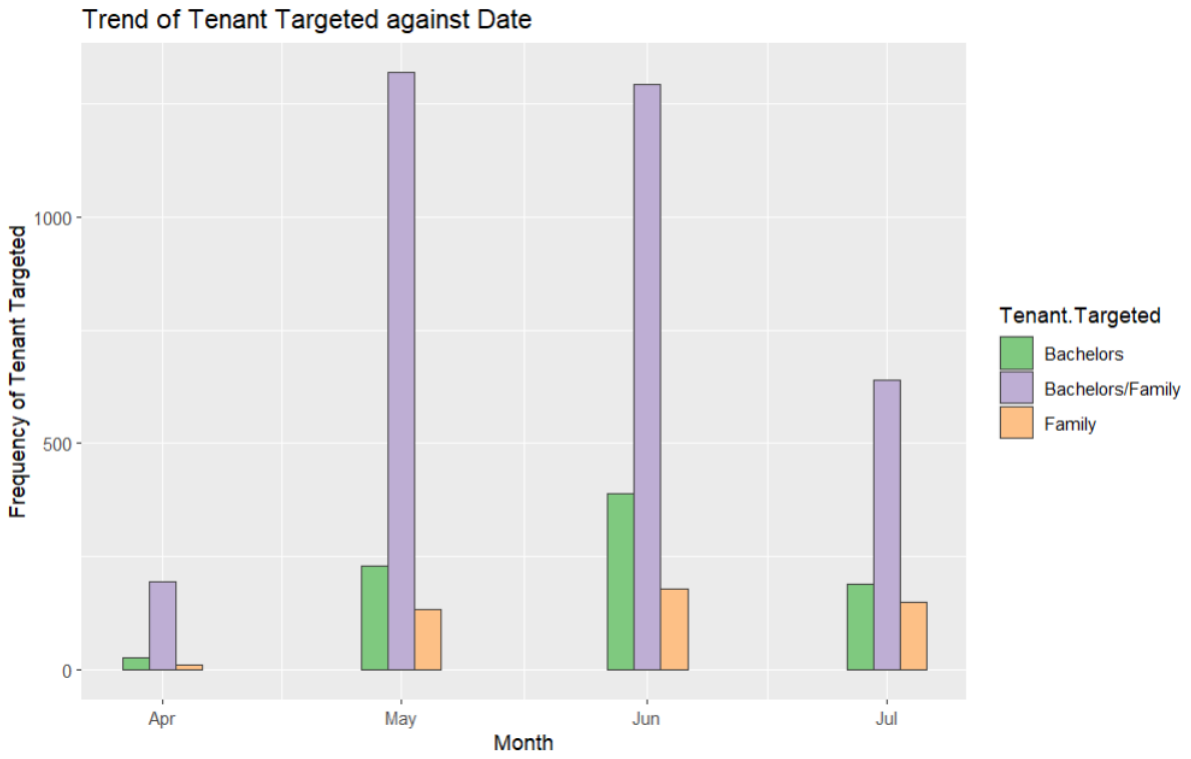


Figure 4-77: Grouped Bar Chart for Trend of Tenant Targeted

Based on the grouped bar chart above, the rental postings that target bachelors has a trend of increasing then experiences downward trend in July. This also applies to rental postings that target “Bachelors/Family” and family. For “Bachelors/Family” category, the trend starts experiencing decreasing in June while the other categories are still increasing. In other words, some of the rental postings has switched to targeting specific group of family or bachelors only.

4.4.7 Analysis 4-7: Bathroom number of rental house

This analysis aims to discover the trend of bathroom number of rental house over a period. The graph selected for this analysis is the scatterplot with a regression line to show the trend of the data points. This is because all the data instances can be shown by using scatterplot and the trend can be identified with the regression line.

```
# Analysis 4-7: Bathroom Number against Date (line)
df %>%
  group_by(Posted.Date) %>%
  summarize(agg = mean(Bathroom.Number)) %>%
  ggplot(mapping = aes(x = Posted.Date, y = agg)) +
    labs(title = "Trend of Bathroom Number of Rental House against Date", x = "Posted Date", y = "Bathroom Number") +
    geom_smooth(method = lm) +
    geom_point(col = "orange") +
    stat_regline_equation(label.y.npc = 0.9)
```

Figure 4-78: Code to Plot Bathroom Number Trend

The code above groups the data by the posted date first then calculate the mean of the bathroom numbers of each group. Mean is used for the aggregated value is because the scale of the bathroom number is small so median is not necessary. Afterwards, “ggplot()” and “geom_point()” functions are used to plot the scatterplot after passing in all the required arguments. Then, the “geom_smooth()” is used to plot the regression line on the scatterplot while “stat_regline_equation()” function is used to display the equation of the regression line.

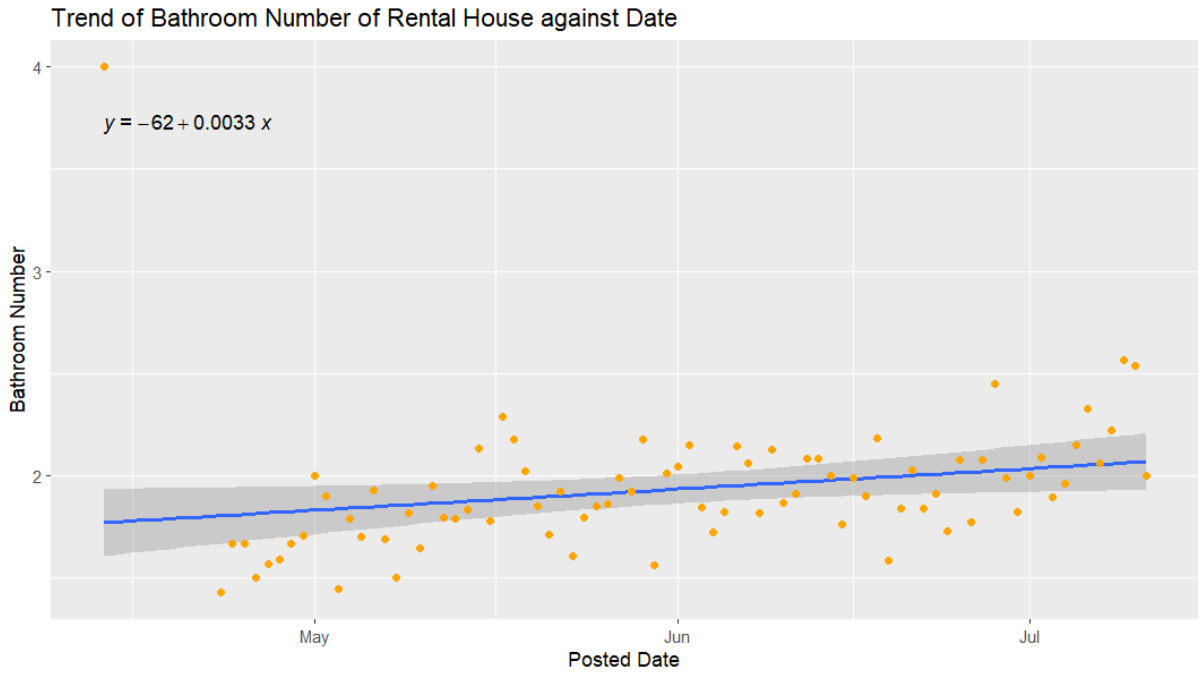


Figure 4-79: Trend of Bathroom Number in Scatterplot

According to the scatterplot above, the trend of the bathroom number is only slightly increasing or even no increasing trend. This means that there are no demand increases for the number of bathrooms as it is deemed as not an important factor when renting a house. This is because the number of bathrooms does not impact the quality of life anymore when it reaches certain number.

4.4.8 Analysis 4-8: Contact person of rental posting

This analysis provides insight on the trend of contact person over a period. The visualization used for this analysis is the bar chart separated in different facets by the type of contact person. This is because the bar chart is good for visualizing the relationship between categorical and continuous variable.

```
# Analysis 4-8: Trend of Contact Person against Date
df %>%
  group_by(month = lubridate::floor_date(Posted.Date, unit = "month"), Contact.Person) %>%
  summarize(frequency = n()) %>%
  ggplot(mapping = aes(x = month, y = frequency, fill = Contact.Person)) +
    labs(title = "Trend of Contact Person over Date", x = "Month", y = "Frequency of Contact Person") +
    geom_bar(stat = "identity", width = 10, col = "#424242") +
    facet_wrap(~Contact.Person)
```

Figure 4-80: Code to Plot Contact Person Distribution over Date

The data is grouped by the months and type of contact person first then the occurrences of records for each group is calculated. Then, “ggplot()” and “geom_bar()” functions will be sued to plot the bar chart. The “facet_wrap()” function is used to separate the charts into different facets by the type of contact person.

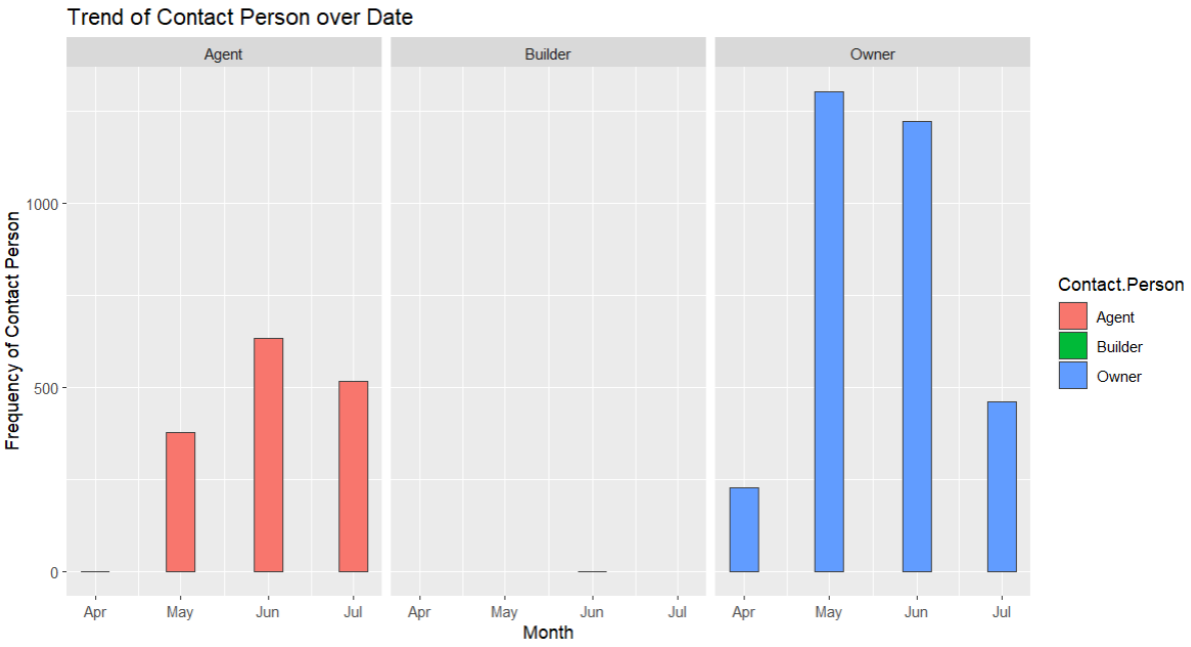


Figure 4-81: Trend of Contact Person over Date

Based on the charts above, the agent as contact person has a trend of increasing until June and decreases in July. For builder as contact person, the sample size is too less for plotting, so the plot is empty. Owner as contact person has trend of skyrocketing from April to May but decreases in June and July. Therefore, the trend of agent and owner as contact person is similar as they both experience an upward trend first then downward trend.

4.4.9 Analysis 4-9: Current floor of rental house

The analysis discovers the trend of current floor of the house being rented over a period. The visualization graph selected is scatterplot with a regression line to display the trend.

```
# Analysis 4-9: Trend of Current Floor against Date
df %>%
  group_by(Posted.Date) %>%
  summarize(agg = mean(Current.Floor)) %>%
  ggplot(mapping = aes(x = Posted.Date, y = agg)) +
    labs(title = "Trend of Current Floor against Date", x = "Posted Date", y = "Current Floor") +
    geom_smooth(method = lm) +
    geom_point(col = "#68228B") +
    stat_regline_equation(label.y.npc = 0.9)
```

Figure 4-82: Code to Plot Trend of Current Floor against Date

The data is grouped by posted date first the mean value for each date group is calculated. Mean value is chosen because the scale of the current floor is small so mean can be used as the aggregated value instead of median. Then, “ggplot()” and “geom_point()” functions are used to plot the scatterplot while “geom_smooth()” function is used to plot the regression line. The “stat_regline_equation()” function is used to show the equation of the regression line.

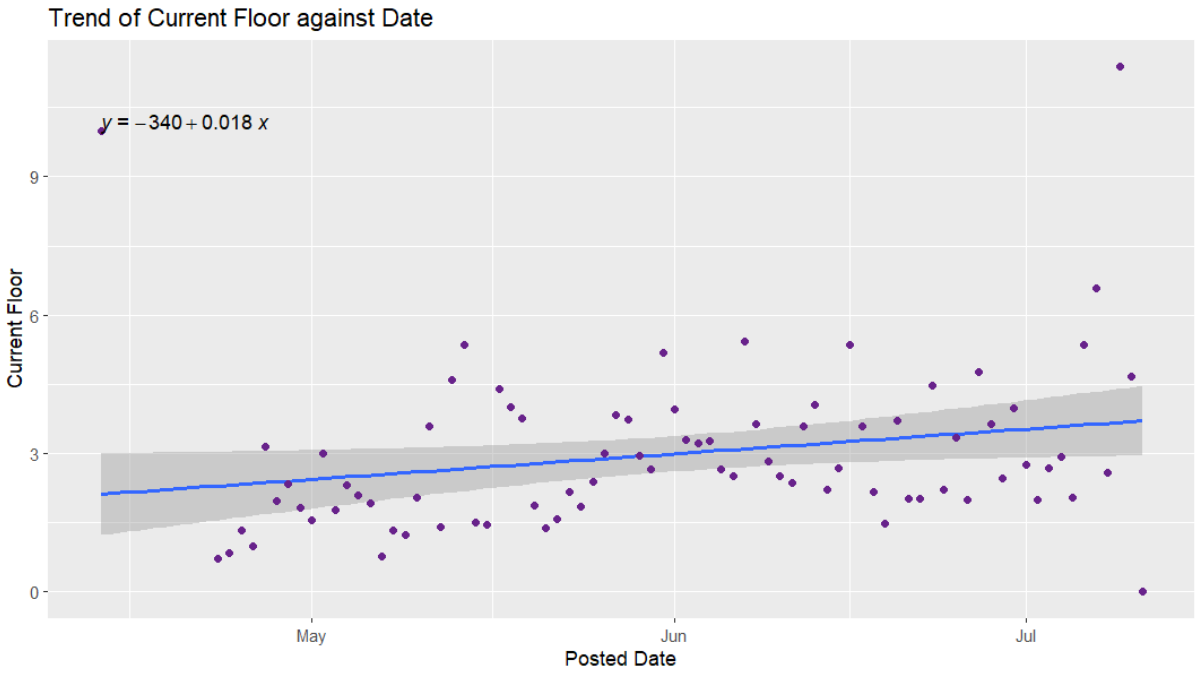


Figure 4-83: Trend of Current Floor against Date

Based on the scatterplot above, there are no obvious increasing trend being shown but the upward trend is weak or close to none. The data points are scattered within a certain range from 0 to 6 randomly. This indicates that there are no houses that are being rented located at higher floor over the months. Therefore, the current floor variable has no trend over the months.

4.4.10 Analysis 4-10: Total floor of rental house

This analysis aims to provide insight on the trend of total floor for the houses that are being rented. The visualizations chosen for this analysis is line chart as it can describes the changes of the variable over time effectively.

```
# Analysis 4-10: Trend of Total Floor against Date
df %>%
  group_by(Posted.Date) %>%
  summarize(Mean.Total.Floor = mean(Total.Floor)) %>%
  ggplot(mapping = aes(x = Posted.Date, y = Mean.Total.Floor)) +
    geom_line(col = "#5D478B") +
    labs(title = "Trend of Total Floor over Date", x = "Posted Date", y = "Total Floor")
```

Figure 4-84: Code to Plot the Trend of Total Floor against Date

The data is first grouped by the posted date then the mean value of the total floor is calculated for each group. Mean value is used instead of median is because the total floor only ranged from 0 to 100 so the scale is small.

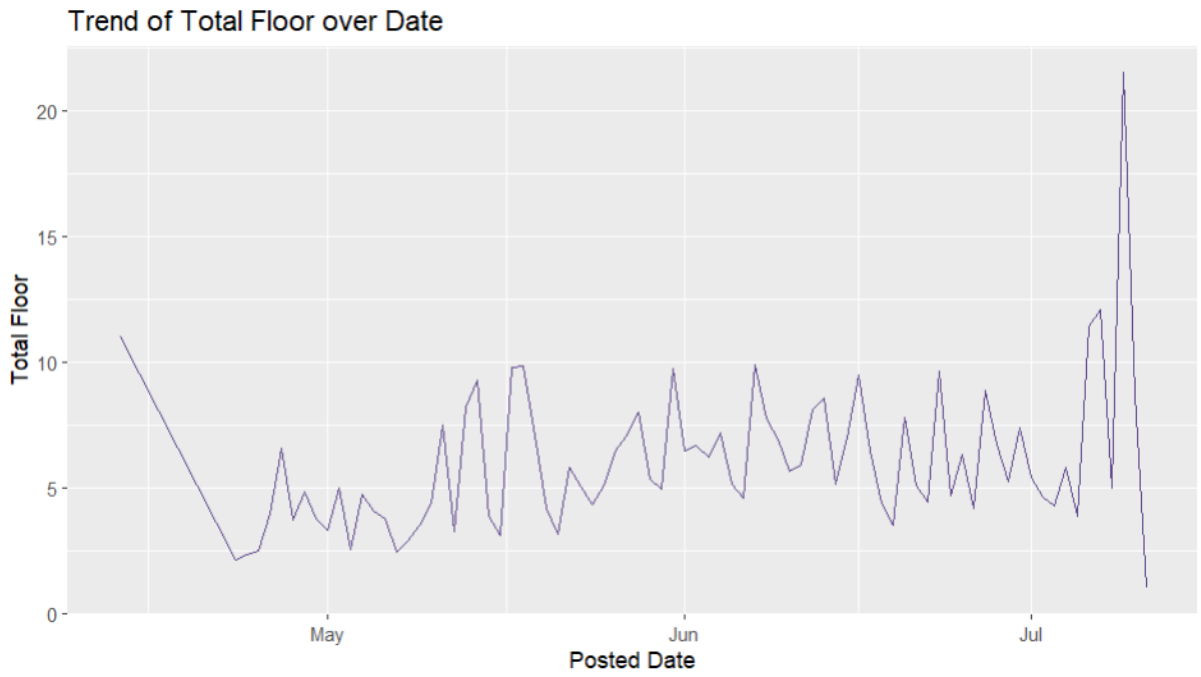


Figure 4-85: Trend of Total Floor over Date

Based on the line chart above, the total floor of the houses posted for rent has no trend of increasing as the line is fluctuating within a specific range with is from 2 to 13. There are spikes after July, but the overall trend is constant. This indicates that the house type posted for rent remains the same which is commonly flat or apartment as the common range of floor is

between 2 to 13. This also means that the common house type posted for the few months remains the same.

4.4.11 Conclusion

To conclude, there are trend in some variables while the other has remained the same for the few months in the dataset. There are few variables such as house size, house furnishing type, bathroom number, current floor, and total floor that has no trend over the months. For area type, the super area type has no obvious trend over the months while the carpet area type has an increasing trend as the formula to measure house size used in carpet area type is more precise when judging with the net usable area. The number of facilities in the rental house posted also has an increasing trend. For variables like city, type of tenants, and contact person such as owner and agent, they have an increasing trend then decreasing in July which is the last month of the dataset. This might be due to the lesser number of samples as compared to other month's number of samples. This is because the date range of the dataset is only between 13rd April 2022 and 7th November 2022 where the end date is only until the beginning of November 2022. Therefore, the increasing trend of those variables are expected to continue if the rental postings data provided in November lasts for 1 entire month.

4.5 Question 5: Comparison of the rental posting by owner and agent

As mentioned above, there are differences in the price range offered by the owner and agent as a contact person. Therefore, to further explore the difference between the houses managed by the agent and owner, some analysis tasks are conducted on the variables such as house size, house furnishing, and the common total floor of the houses provided by respective contact person.

4.5.1 Analysis 5-1: Comparison of house size by owner and agent

This analysis aims to find out the difference in the house size distribution that has the agent or the owner as contact person. The comparison was aided using histogram separated in different facets by the type of contact person as histogram can effectively visualize the distribution of a continuous variable.

```
# 5. Characteristics of House posted by Owner and Agent comaparison
# Analysis 5-1: House Size comparison between House posted by Owner and Agent
contact_df <- df %>% filter(Contact.Person == "Agent" | Contact.Person == "Owner")

contact_df %>%
  ggplot(mapping = aes(x = House.Size)) +
    labs(title = "House Size Distribution of Owner and Agent", x = "House Size", y = "Frequency") +
    geom_histogram(colour = "orange", aes(fill = ..count..)) +
    scale_fill_gradient("Count", low = "green", high = "red") +
    facet_wrap(~Contact.Person)
```

Figure 4-86: Code to Plot the Histogram

The code above constructs the histogram first by filtering all the builder contact person out and store the filtered data frame into a new variable. This is because the analysis only need to compare the characteristics between the agent and owner. Therefore, all the instances with builder as a contact person is removed. Afterwards, the data is plotted in histogram using “ggplot()” and “geom_histogram()” functions after passing in all the required arguments. The “scale_fill_gradient()” function is used to adjust the gradient fill colour palette while the “facet_wrap()” function separated the graphs into different facets by the type of contact person.

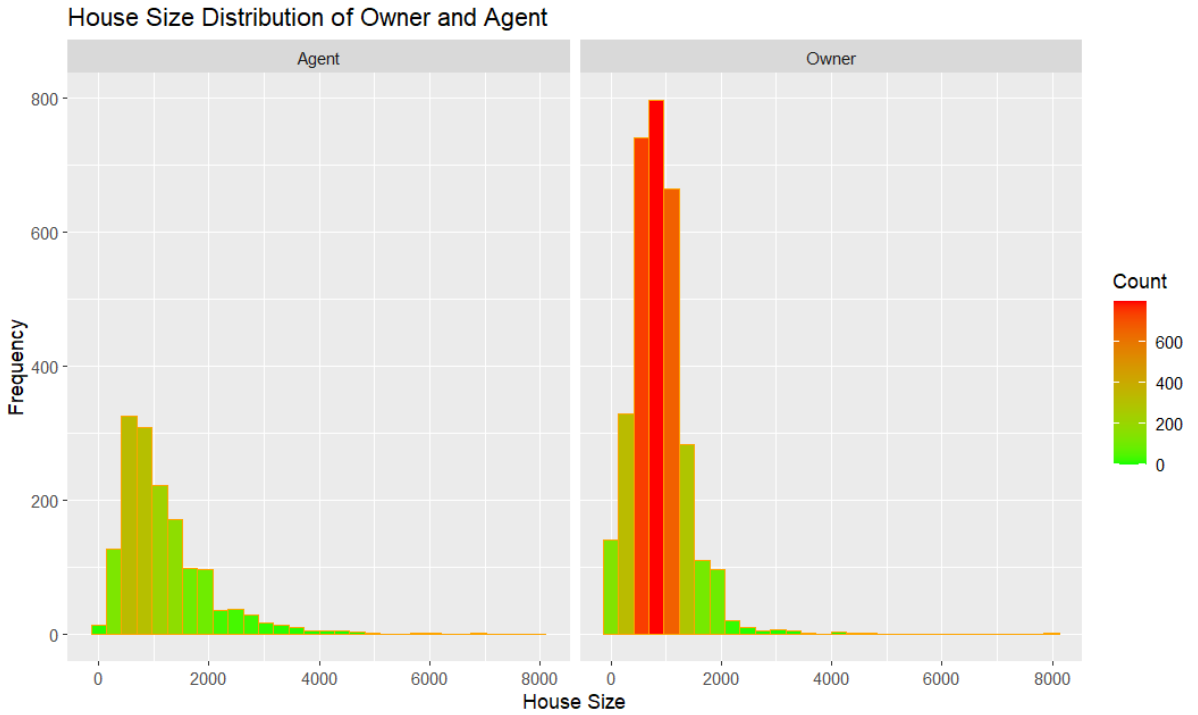


Figure 4-87: House Size Distribution for Both Contact Person

Based on the histogram above, the distribution for both type of contact person appears to be the same which is skewed to right. This means that the most common house size in the 6 cities of India is between 0 and 3000 unit. The difference in these two distributions is the value of the frequency where more owners has offered the house for rent postings as compared to that of agent. This may indicate that in the 6 cities of India, owner as a contact person is more popular as more people willing to find the house owners when renting a house.

4.5.2 Analysis 5-2: On each city, the rental postings count by owner or agent

This analysis aims to provide insight on the preference of contact person when renting a house. This analysis will be visualized using bar chart as the need to describe the relationship between categorical and continuous variable. The bar chart will be separated in different facets by the type of contact person.

```
# Analysis 5-2: On each city, the preference of owner or agent posting rental house
contact_df %>%
  group_by(House.City, Contact.Person) %>%
  summarize(total_count = n()) %>%
  ggplot(mapping = aes(x = House.City, y = total_count)) +
    labs(title = "Count of Rental House that has Agent or Owner as Contact Person in Each City", x = "House City", y = "Total Count") +
    geom_bar(stat = "identity", position = "dodge", width = 0.5, mapping = aes(fill = total_count)) +
    scale_fill_gradientn(colors = c("#00BFFF", "#1E90FF", "#104E8B")) +
    facet_wrap(~Contact.Person)
```

Figure 4-88: Code to Plot the Preference of Contact Person in Each City

The code above groups the data by city and contact person then calculate the occurrences of each group to analyse the preference of contact person in each city. Then, "ggplot()" and "geom_bar()" functions is used to plot the bar chart which then separated in different facets using "facet_wrap()" function by the type of contact person. "scale_fill_gradient()" function is used to adjust the gradient fill colour of the bar according to the value of the category.

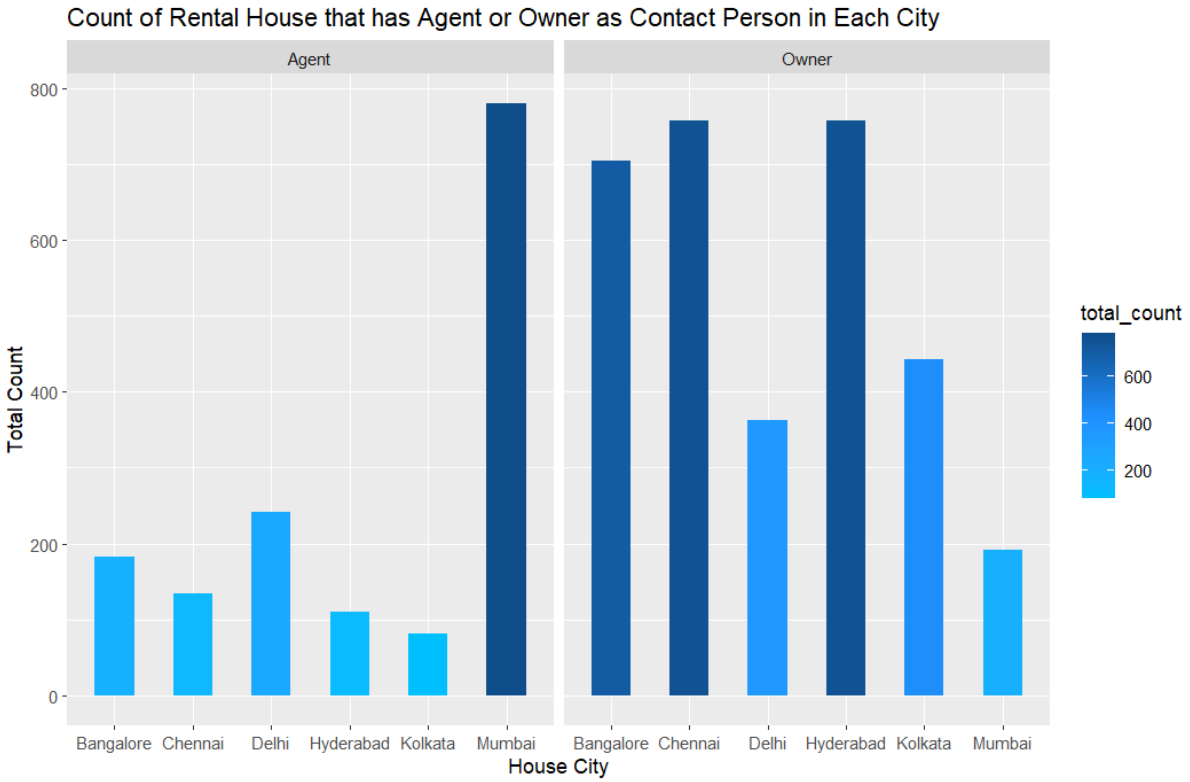


Figure 4-89: Rental Posting Occurrences Based on Cities by Owner and Agent

Based on the bar charts above, the frequency of rental posting in Mumbai city for agent as contact person is the highest compared to other cities. However, when owner as contact person, the frequency of rental posting in other cities beside Mumbai is higher in general. This means that there is trend for agent as contact person in Mumbai but owner as contact person in other cities besides Mumbai. This indicates that the tenants in Mumbai prefer to find agent while the tenants in other cities prefer to find the house owner directly when looking for house to rent.

4.5.3 Analysis 5-3: House furnishing of rental houses by owner and agent

This analysis aims to provide insight on the difference and similarity of the house furnishing of the rental house by owner and agent. The visualizations graph selected for this analysis is subplot of pie charts that is aligned side by side. This ease the process of doing comparison for both type of contact person. Pie chart is great for visualizing the distribution of a categorical value to the whole group.

```
# Analysis 5-3: House Furnishing of Rental House posted by Agent and Owner.
plot_furnishing_contact <- function() {
  data <- contact_df %>% group_by(House.Furnishing, Contact.Person) %>% summarize(Count = n())
  fig <- plot_ly()
  fig <- fig %>%
    add_pie(
      data = data %>% filter(Contact.Person == "Agent"),
      name = "Agent",
      value = ~Count,
      labels = ~House.Furnishing,
      title = "Agent Distribution",
      domain = list(row = 0, column = 0)
    )

  fig <- fig %>%
    add_pie(
      data = data %>% filter(Contact.Person == "Owner"),
      name = "Owner",
      value = ~Count,
      labels = ~House.Furnishing,
      title = "Owner Distribution",
      domain = list(row = 0, column = 1)
    )

  fig <- fig %>% layout(
    title = "Distribution of Furnishing Status of Rental House by Agent and Owner",
    showlegend = TRUE,
    grid = list(rows = 1, columns = 2),
    xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
    yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE)
  )
  fig
}
plot_furnishing_contact()
```

Figure 4-90: Code to Plot Subplots of Pie Chart

The code above groups the data by house furnishing and contact person then the frequency for each group is calculated. Firstly, the figure for the subplot is instantiated with “plot_ly()” function then each pie charts is added to the figure after customizing with the data and adjustments required. Then, the layout of the figure is constructed with “layout()” function by setting parameters such as title, row and columns count.

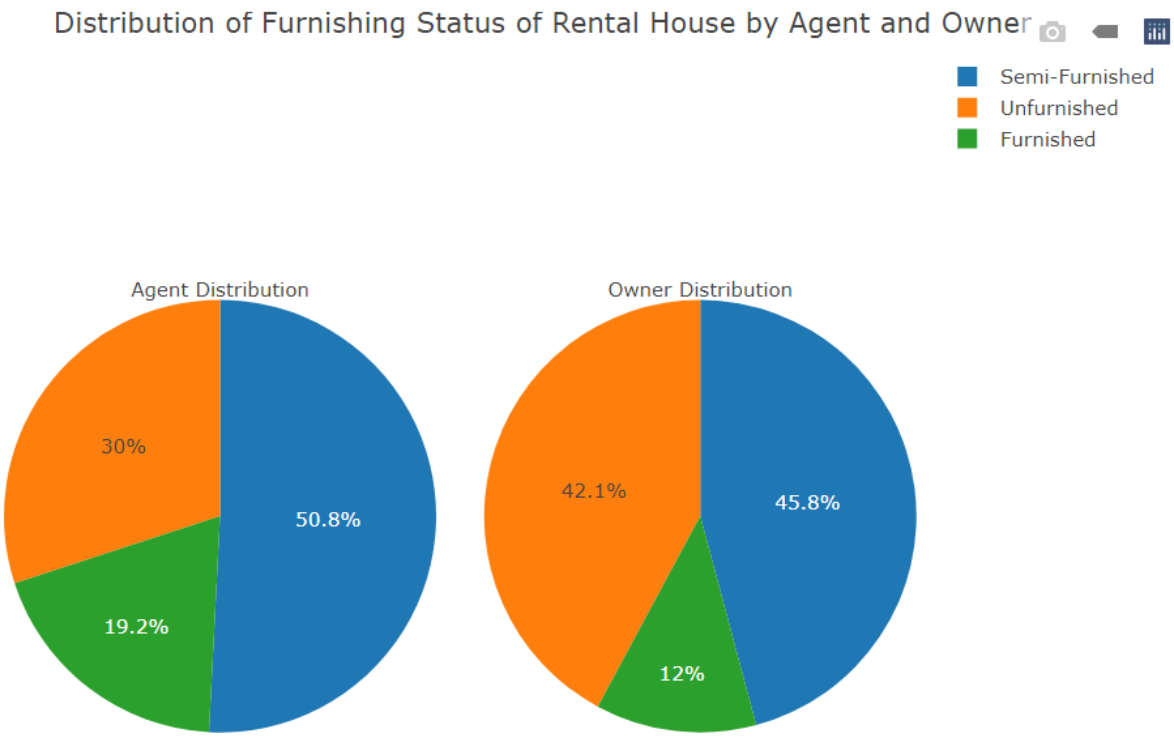


Figure 4-91: Distribution of House Furnishing According to Agent and Owner

Based on the pie charts above, for rental houses posted by the agent, majority of the houses are furnished (50.8%) while the minority of the houses are unfurnished (30%). This means that the houses that has agent has contact person usually are furnished with furniture which is also the reason the overall price posted by agent is higher in general. For rental houses that has their owner as contact person, more houses are unfurnished compared to the houses have agent as contact person, but still lesser than that of furnished houses. This means the house that has owner as contact person usually are unfurnished as the percentage of unfurnished houses are very close to half of the pie chart. To conclude, the rental houses by owner usually are unfurnished while the houses by agent usually are furnished.

4.5.4 Analysis 5-4: Tenant targeted by owner and agent

This analysis aims to find out the tenant targeted by the rental posting that has owner or agent as the contact person. The graph type selected for this analysis is bar chart that is separated in different facets by the type of contact person. Bar chart is selected as the visualization of the relationship between a categorical and continuous variable takes place.

```
# Analysis 5-4: Tenant Targeted by Agent or Owner
contact_df %>%
  ggplot(mapping = aes(x = Tenant.Targeted)) +
    labs(
      title = "Tenant Targeted by the Rental Posting that has Agent or Owner as Contact Person",
      x = "Tenant Targeted",
      y = "Frequency"
    ) +
    geom_bar(col = "#2F4F4F", width = 0.5, alpha = 0.7, mapping = aes(fill = after_stat(count))) +
    scale_fill_gradient("Count", low = "#00FFFF", high = "#008B8B") +
    facet_wrap(~Contact.Person)
```

Figure 4-92: Code to Plot Tenant Distribution by Contact Person

The code above constructs the bar chart with “ggplot()” and “geom_bar()” functions after passing all the required arguments. Then, the bar chart is separated in different facets with “facet_wrap()” function by the type of contact person.

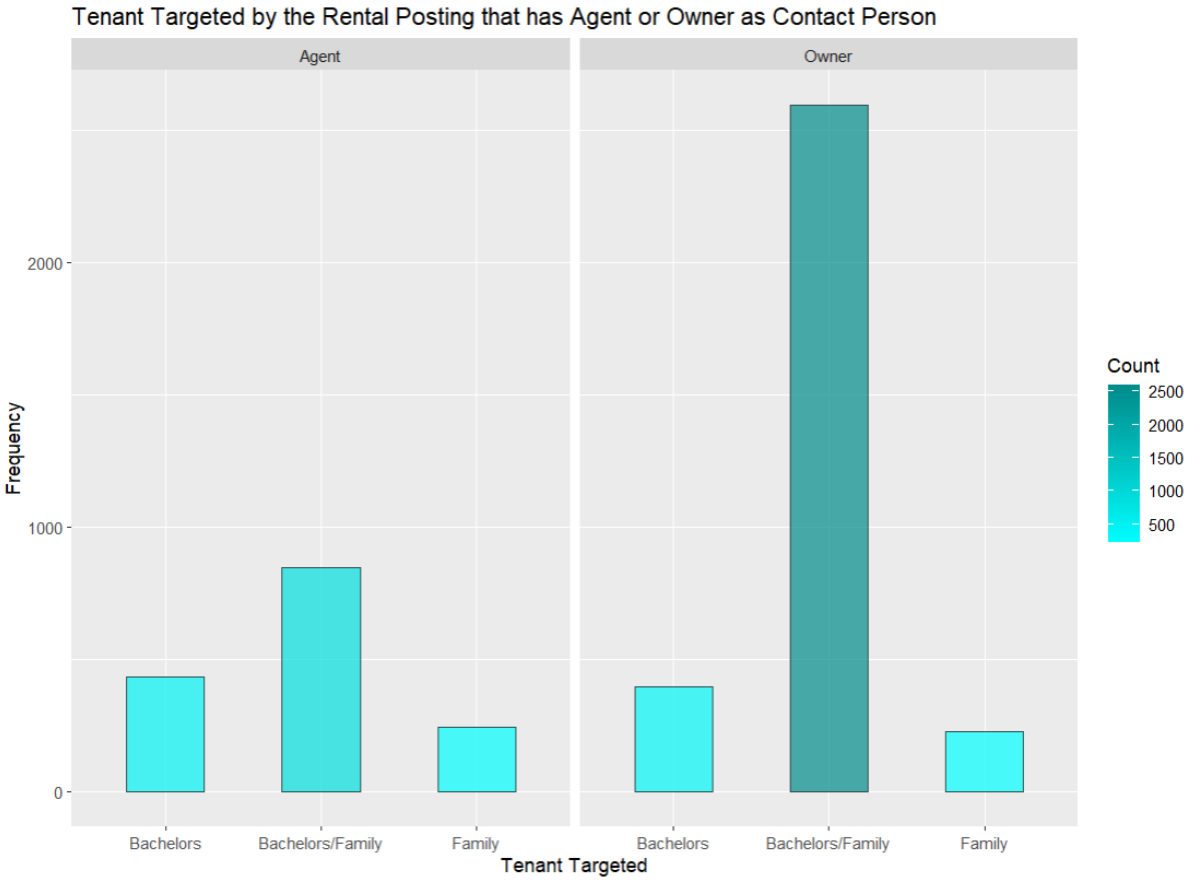


Figure 4-93: Comparison of Contact Person on Tenants Targeted

Based on the bar charts above, the distribution of tenants targeted by the rental posted that has agent or owner as contact person are similar. The most targeted type of tenants is both bachelors and family followed by the bachelors then the family group. For rental posting that has owner as contact person, the occurrences where both bachelors and family are targeted at

the same time has extremely high values as compared to other category. This might indicate that when house owners act as contact person, the renal posting does not target a specific group but rather all the type of tenants available in the market.

4.5.5 Analysis 5-5: Total floor of rental houses by owner and agent

This analysis provides insight on the difference and similarity of the total floor distribution of the rental house that has owner or agent as contact person. The visualization graph selected for this analysis is the histogram separated in different facets by the type of contact person.

```
# Analysis 5-5: Total Floor of the Rental House posted by Agent and Owner
contact_df %>%
  ggplot(mapping = aes(x = Total.Floor)) +
    labs(
      title = "Total Floor Distribution by Type of Contact Person",
      x = "Total Floor",
      y = "Frequency",
      fill = "Frequency"
    ) +
    geom_histogram(col = "#FF4500", aes(fill = after_stat(count))) +
    scale_fill_gradient(low = "#FFA07A", high = "#8B5742") +
    facet_wrap(~Contact.Person)
```

Figure 4-94: Code to Plot Total Floor Distribution of Rental Houses

The code above constructs the histogram plot using “ggplot()” and “geom_histogram()” after passing in all the arguments. With “facet_wrap()” function, the bar charts are separated in different facets by type of contact person. The “scale_fill_gradient()” function is used to adjust the gradient fill colour of the bar according to the values.

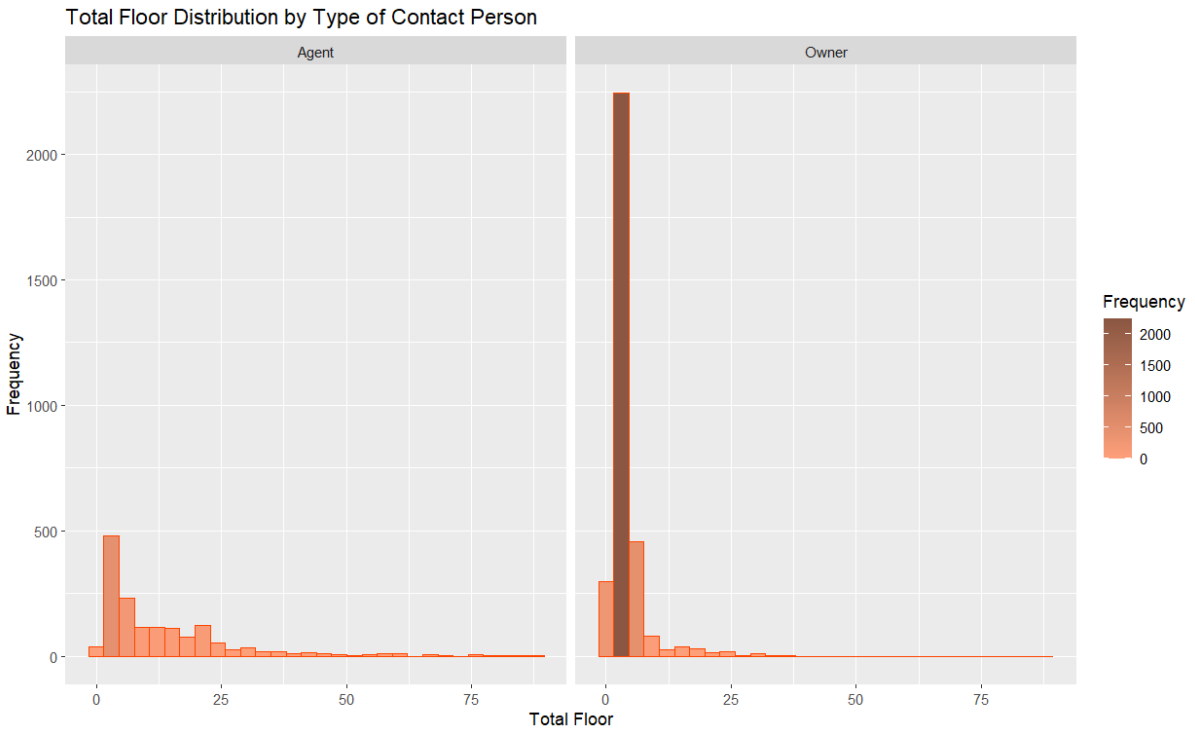


Figure 4-95: Total Floor Distribution by Type of Contact Person

Based on the histogram above, the total floor distribution of houses that has agent as contact person is similar to that of houses that has owner as contact person. Both distributions are skewed to the right where the total floor is commonly lies between 0 and 25 for both type of contact person. When agent as the contact person, the total floor distribution of house for rent has higher values in total floor in all range. Meanwhile, when the house owner acts as the contact person, the total floor distribution focuses on the range between 0 and 10. This indicates that the total floor managed by agent has higher varieties in the total floor variable while the houses that has owner as contact person are monotonous as the total floor of houses are focuses on small range.

4.5.6 Analysis 5-6: In each city, the facilities number of rental houses by owner and agent

The analysis provide insight on the facilities number distribution of rental houses that has agent and owner as contact person. The visualization selected for this analysis is boxplot separated in different facets by the city.

```r
# Analysis 5-7: In each city, Area Type of Rental House posted by Agent and Owner
contact_df %>%
  group_by(House.City, Area.Type, Contact.Person) %>%
  summarize(Count = n()) %>%
    ggplot(mapping = aes(x = House.City, y = Count, fill = Area.Type)) +
      labs(
        title = "Area Type of Rental House Posted by Agent and Owner in Each City",
        x = "City",
        y = "Count",
        fill = "Frequency"
      ) +
      geom_bar(stat="identity", position = "dodge", width = 0.5, col = "#4D4D4D") +
      facet_wrap(~Contact.Person)
```

Figure 4-96: Code to Plot the Distribution of Facility Number by City nad Contact Person

The code above constructs boxplot for the facilities number distribution using “ggplot()” and “geom_boxplot()” functions after passing in all required arguments.

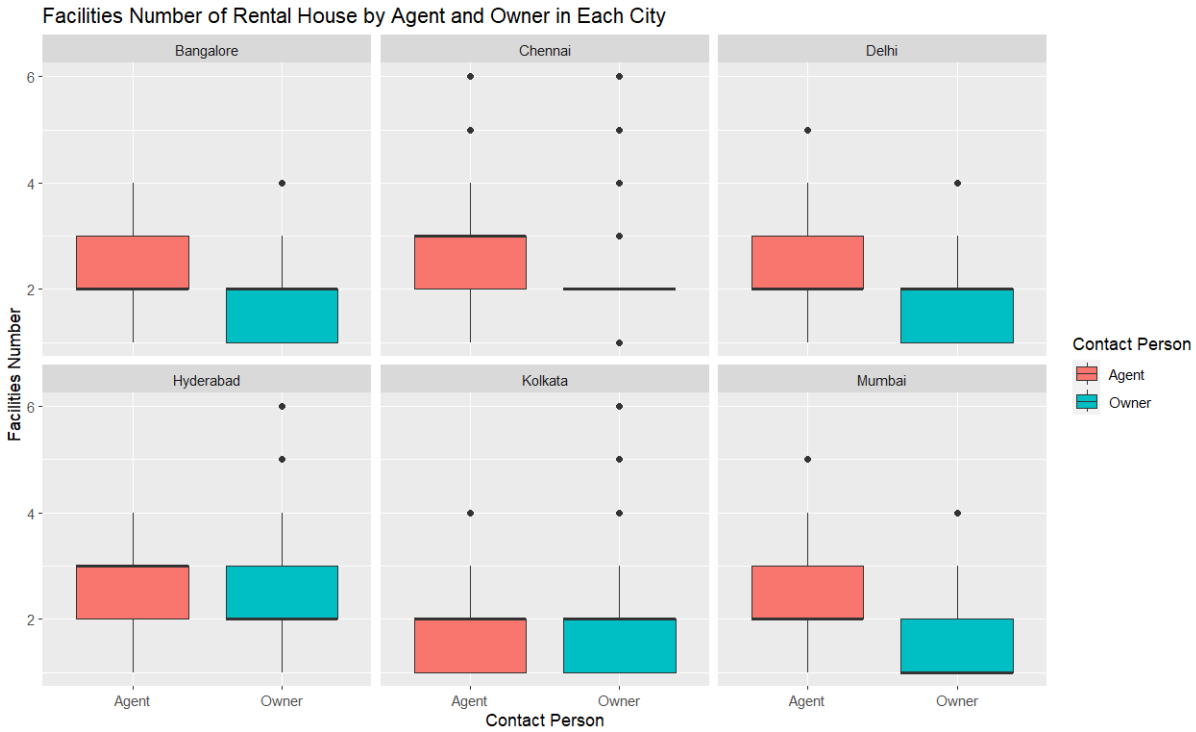


Figure 4-97: Facilities Number Distribution of Houses by Agent or Owner in Each City

Based on the chart above, the facility number distribution range for houses that has agent as contact person is always overall higher or equal to that of houses that has owner as contact person in all cities. In Chennai, the distribution for houses by owner has its distribution scattered at all range. This indicates that the instances has different number of facilities in the house. To conclude, the number of facilities in the houses by agent usually is higher than that

of houses by owner because the agent can have access to all the house for sale easily. Therefore, the houses managed by agent has more amount of facility.

4.5.7 Analysis 5-7: In each city, the area type of rental houses posted by owner and agent

This analysis aims to provide insight on the area type distribution of rental houses that has owner or agent as a contact person. The visualization graph selected for this analysis is the grouped bar chart that is separated in different facets by the type of contact person.

```
# Analysis 5-7: In each city, Area Type of Rental House posted by Agent and Owner
contact_df %>%
  group_by(House.City, Area.Type, Contact.Person) %>%
  summarize(Count = n()) %>%
    ggplot(mapping = aes(x = House.City, y = Count, fill = Area.Type)) +
      labs(
        title = "Area Type of Rental House by Agent and Owner in Each City",
        x = "City",
        y = "Count",
        fill = "Frequency"
      ) +
      geom_bar(stat="identity", position = "dodge", width = 0.5, col = "#4D4D4D") +
      facet_wrap(~Contact.Person)
```

Figure 4-98: Code to Plot Area Type Distribution Grouped Bar Chart

The code above groups the data by the city, area type, and the contact person then calculate the occurrences of each group with “n()” function. Then, the grouped bar chart is plotted with “ggplot()” and “geom_bar()” functions after passing in all the required arguments. The bar chart is then separated in different facets using “facet_wrap()” function by the contact person.

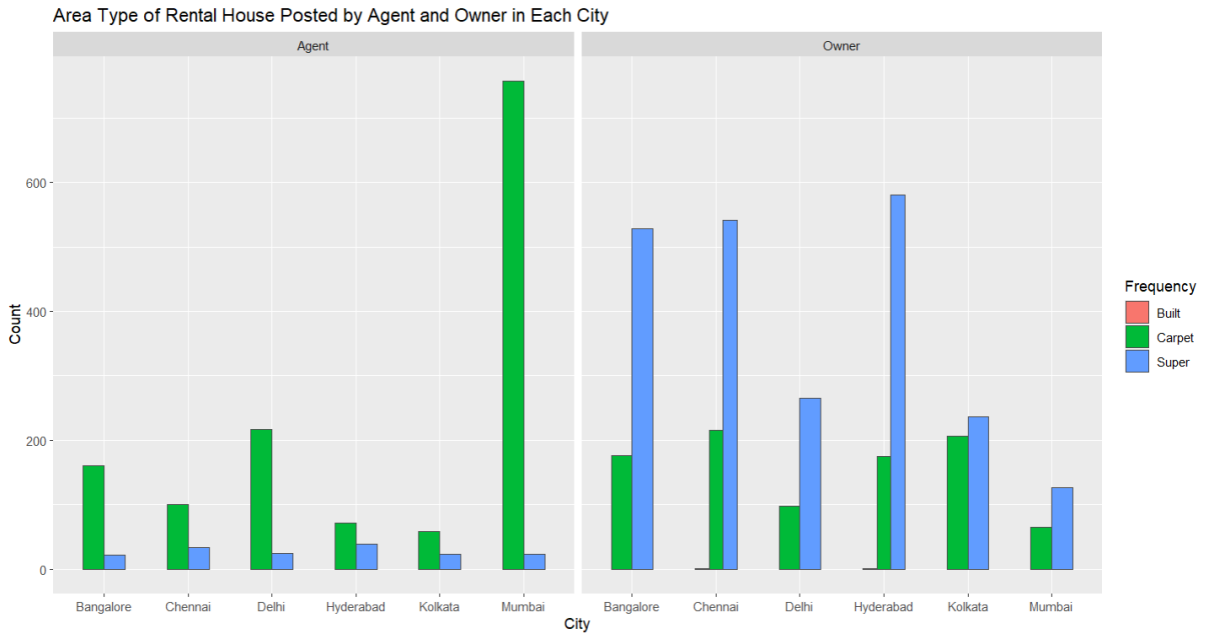


Figure 4-99: Area Type Distribution by Agent and Owner in Each City

Based on the bar charts above, when the agent acts as the contact person for the house, then the houses usually utilize carpet area type to calculate the house size. On the other hand, when the house owners act as the contact person, then the houses are commonly utilizing the super area type for house size calculation. This trend can be identified in the agent facet where all occurrences of carpet area type houses are higher than that of super area type in all cities and vice versa in the owner facet. As more houses managed by agent utilizing carpet area type, so it can be said that the house by agent is more protected as RERA will provide protection for houses that use carpet area type.

4.5.8 Analysis 5-8: The bathroom number of rental houses by owner and agent

This analysis aims to find out the bathroom number of rental houses managed by the owner and agent. The graph selected for this analysis is bar chart separated in different facets by the type of contact person. Bar chart is selected as it is appropriate to visualize the relationship between categorical and continuous variables.

```r
# Analysis 5-8: Bathroom Number of Rental Houses Posted by Agent and Owner
plot_percent_stack <- function() {
  final_df <- contact_df
  final_df$Bathroom.Number <- as.character(final_df$Bathroom.Number)

  final_df %>%
    group_by(Contact.Person, Bathroom.Number) %>%
    summarize(frequency = n()) %>%
      ggplot(mapping = aes(x = Bathroom.Number, y = frequency, fill = Contact.Person)) +
        geom_bar(stat = "identity", position = "fill") +
        labs(title = "Bathroom Number Count by Owner and Agent", x = "Bathroom Number", y = "Frequency") +
        scale_y_continuous(labels = percent)

}
plot_percent_stack()
```

Figure 4-100: Code to Plot Bathroom Distribution by Owner and Agent

The code above groups the data first by the contact person and bathroom number before plotting with “group_by()” function. Then, the occurrences of each group is calculated with the “summarize()” and “n()” functions. Afterwards, the bar chart is plotted with “ggplot()” and “geom_bar()” functions which then separated into different facets by the type of contact person with “facet_wrap()” function.

Figure 4-101: Bathroom Number by Owner and Agent

The percent stacked bar chart above visualize the composition for owner and agent percentage by the number of bathrooms. There is an upward trend of the houses managed by agent according to the composition when the number of bathrooms increases until it reaches 6 bathrooms. This indicates that the houses managed by the agent has more bathrooms while the houses managed by has less bathrooms in general. This trend stopped after the number of bathrooms exceed 6 and the houses that has 10 bathrooms is exclusive to the owner as contact person. This trend might be due to the rent of luxury houses are usually managed by the agents as luxury houses will be having more facilities inside the house such as bathrooms, bedroom, and kitchen. The trend stopped as number of bathrooms that exceed 6 is uncommon among all house types so that it is only exclusive to the houses that managed by the owner themselves as they are allowed to add more bathroom in their house by contacting the builder.

4.5.9 Conclusion

To conclude, the house size distribution for both contact person is almost the same which means the house size of house for rent is not affected by the contact person. However, the houses managed by agent has more varieties in terms of the house size. Besides, the tenants in Mumbai prefer to look for agent when renting a house while the tenants in other cities prefer to find the house owners directly. Other than that, more houses managed by the house owners are unfurnished, but houses managed by agent are overall furnished as agent is professional in their work. Therefore, the houses they help to rent will be furnished so that it can attract more tenants. Thanks to that, the houses managed by the agents are also commonly providing more facilities. On top of that, rental posting that has house owners as contact person usually does not target any specific group of tenants. Houses managed by the agent also utilize carpet area type for house size calculation which is more precise. The occurrences of houses managed by agents increases when the bathroom numbers increases. This trend stopped until the number of bathrooms reaches 6 as the houses with more than 6 bathrooms are mostly exclusive to the house owners.

4.6 Question 6: What is the potential in each Cities and Localities?

This section aims to analyse the potential that exists within all cities or localities of India given in the dataset. The potential mentioned includes the tenant groups that are more common in each city, top 10 cheapest rental price in localities, or even the total floor distribution in each city.

4.6.1 Analysis 6-1: Tenant distribution in each city

This analysis aims to provide insight on the tenant distribution in each city so that the potential of targeting specific groups in respective city can be identified. Percent stacked bar chart is selected for this analysis as it can visualize the distribution of each sub-category by the main category effectively.

```
# 6. Areas / Localities with highest potential
# Analysis 6-1 Tenant Distribution in each City
df %>%
  group_by(House.City, Tenant.Targeted) %>%
  summarize(Total.Tenant = n()) %>%
  mutate(percentage = calculate_percent(Total.Tenant)) %>%
  ggplot(mapping=aes(fill = Tenant.Targeted, x = House.City, y = Total.Tenant)) +
    labs(title = "Distribution of Tenant Preferred in Each City", x = "City", y = "Frequency") +
    geom_bar(position = "fill", stat = "identity", col="black", width=0.45) +
    scale_y_continuous(labels = scales::percent) +
    geom_text(aes(label = paste0(percentage, "%")), size = 3, position = position_fill(vjust = 0.5)) +
    scale_fill_brewer(palette = "Spectral") +
    theme(
      axis.text.x = element_text(angle = 45, vjust = 0.5, hjust = 1),
      axis.title.x = element_text(margin = margin(t=30))
    )
```

Figure 4-102: Code to Plot Tenant Distribution in Each City

The code above groups the data by the house city and tenant targeted which then the total tenants of each group is calculated. After that, the graph is plotted with the “ggplot()” and “geom_bar()” functions. Then, some labels adjustments is set using “scale_y_continuous()”, “geom_text()” and “theme()” functions.

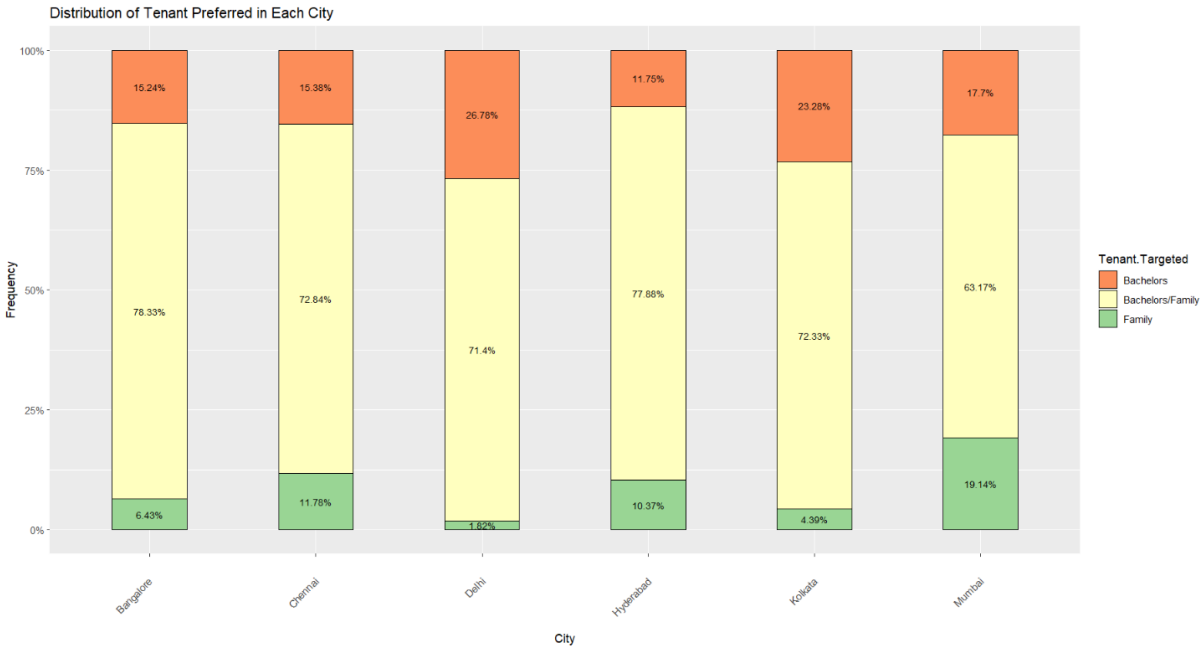


Figure 4-103: Distribution of Tenant Preferred in Each City

Based on the chart above, all cities has the majority of "Bachelors/Family" group which indicates that landlord do not target specific group of tenants in general. For cities other than Mumbai, more landlords are targeting bachelor as their tenant preferred as the percentage of bachelor groups are always higher than that of family group. In Mumbai, family group is slightly preferred over bachelor group maybe due to the prosperous economy which causes more family to stay in the city. To conclude, rental posting that does not target any specific groups is always the trend in every city while targeting family group is slightly favoured in Mumbai. In other cities, bachelor group is targeted more frequently when a house is for rent.

4.6.2 Analysis 6-2: The top 20 expensive of rental price per unit based on locality

This analysis aims to find out the top 20 localities that has the highest rental price per unit of the house for rent by comparing the maximum rental price per unit in each locality. The visualization graph used for this analysis is horizontal bar chart as it can compare values among many categorical values easily.

```r
# Analysis 6-2: Rental per Unit based on Locality (Top 20 Expensive)
plot_top20_expensive <- function() {
  # group data
  grouped_data <-
    df %>%
      mutate(House.Locality = paste(House.Locality, ", ", House.City, sep = "")) %>%
      group_by(House.Locality) %>%
      summarize(agg = max(Rent.Per.Unit))

  # sort in descending order and get the top 20 (which is the most expensive 20)
  top20_expensive <- head(grouped_data[order(-grouped_data$agg), ], 20)
  ggplot(data = top20_expensive, mapping = aes(x = reorder(House.Locality, +agg), y = agg)) +
    labs(
      title = "Top 20 Expensive Rental Price per Unit by Locality",
      x = "Localities",
      y = "Rental Price per Unit"
    ) +
    geom_bar(stat = "identity", mapping = aes(fill = agg), width = 0.5) +
    scale_fill_viridis("Median of Rent Per Unit Scale", option = "D") +
    geom_text(aes(label = agg), hjust = -0.2) +
    coord_flip()

}
plot_top20_expensive()
```

Figure 4-104: Code to Plot Top 20 Expensive Rental Price per Unit

The code above compute a new column by concatenating the locality string with the house city string separated by comma. Then, the house locality is grouped and maximum of rental price per unit is identified for each group. Afterwards, the data is sorted by descending order by the rental price per unit then first 20 rows of record is sliced out and stored into a variable. Then, the sliced data is used to plot the horizontal bar chart with functions such as "ggplot()", "geom_bar()", and "coord_flip()".

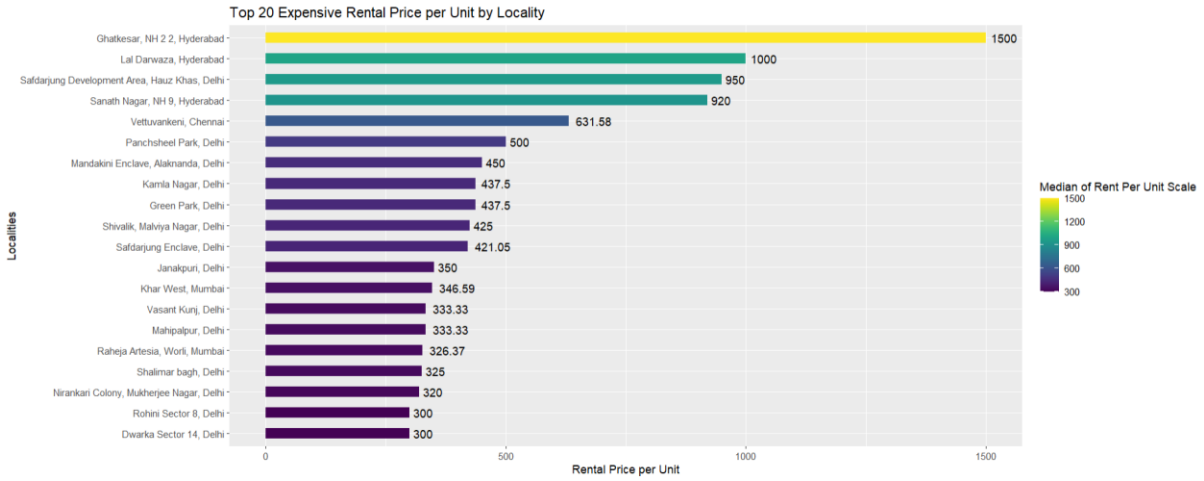


Figure 4-105: Top 20 Expensive Rental Price per Unit by Locality

Based on the bar chart above, the locality that has the most expensive rental price per unit, which is 1500 Rupee per unit, in the dataset is Ghatkesar, NH 2 2 which is in Hyderabad. The locality that has the lowest rental price per unit, which is 300 Rupee per unit, in the list is

Dwarka Sector 14 which is in Delhi. Although Mumbai is one of the most prosperous cities in India, the rental price per unit in Mumbai is only located at top 13.

4.6.3 Analysis 6-3: The top 20 cheapest of rental price per unit based on locality

This analysis aims to discover the top 20 localities that has the lowest rental price per unit of the house for rent by comparing the minimum rental price per unit in each locality. Horizontal bar chart is used for this analysis as it can do comparison between values among categorical values effectively.

```
# Analysis 6-3: Rental per Unit based on Locality (Top 20 Cheapest)
plot_top20_cheapest <- function() {
  grouped_data <-
    df %>%
      mutate(House.Locality = paste(House.Locality, ", ", House.City, sep = "")) %>%
      group_by(House.Locality) %>%
      summarize(agg = min(Rent.Per.Unit))

  top20_cheapest <- head(grouped_data[order(grouped_data$agg), ], 20)
  ggplot(data = top20_cheapest, mapping = aes(x = reorder(House.Locality, -agg), y = agg)) +
    labs(
      title = "Top 20 Cheapest Rental Price per Unit by Locality",
      x = "Localities",
      y = "Rental Price per Unit"
    ) +
    geom_bar(stat = "identity", mapping = aes(fill = agg), width = 0.5) +
    scale_fill_viridis("Median of Rent Per Unit Scale", option = "D") +
    geom_text(aes(label = agg), hjust = -0.2) +
    coord_flip()

}
plot_top20_cheapest()
```

Figure 4-106: Code to Plot Top 20 Cheapest Rental Price per Unit by Localities

The code above computes new column by concatenating the locality value with the city separated by comma. The locality is then grouped and the minimum value of the rental price per unit of each group is identified. After that, the data is reordered ascendingly and slice the top 20 rows of the data out and store into a variable. The data is used to plot the horizontal bar chart with “ggplot()”, “geom_bar()”, and “coord_flip()” functions.

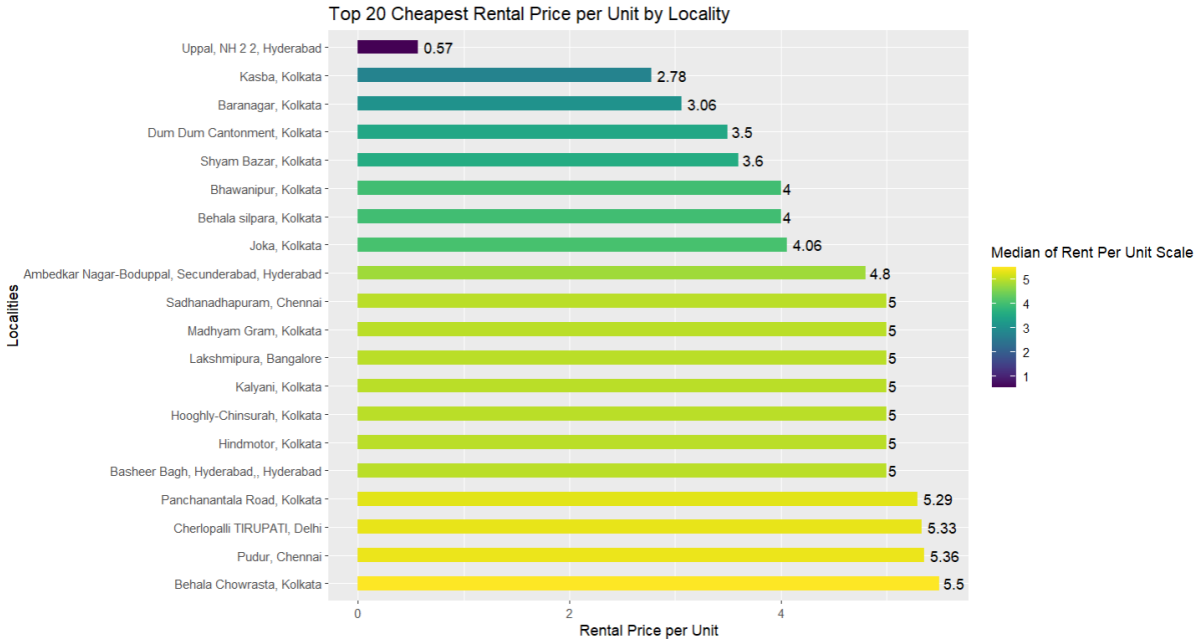


Figure 4-107: Top 20 Cheapest Rental Price per Unit by Localities

Based on the bar chart above, the locality that has the cheapest rental price per unit, which is 0.57 Rupee per unit, is Uppal, NH 2 2 which is in Hyderabad. Behala Chowrasta in Kolkata has the most expensive rental price per unit in the list which is 5.5 Rupee per unit. There is none rental posting from Mumbai enters the list above because Mumbai's house price is higher in overall.

4.6.4 Analysis 6-4: The total floor distribution in each city

This analysis aims to explore the distribution of the total floor of houses for rent in each city to identify the potential of house type in respective city. The visualization used for this analysis is scatterplot where the categorical values is the x-axis. Scatterplot is used because it can visualize the range where the instances lies at and the density of that range.

```
# Analysis 6-4 Total Floor in each city
ggplot(data = df, mapping = aes(x = House.City, y = Total.Floor, col = House.City)) +
  labs(
    title = "Total Floor Distribution in Each City",
    x = "City",
    y = "Total Floor",
    col = "City"
  ) +
  geom_point()
```

Figure 4-108: Code to Plot Total Floor Distribution in Each City

The code above constructs the scatterplot with “ggplot()” and “geom_point()” functions after passing in required arguments.

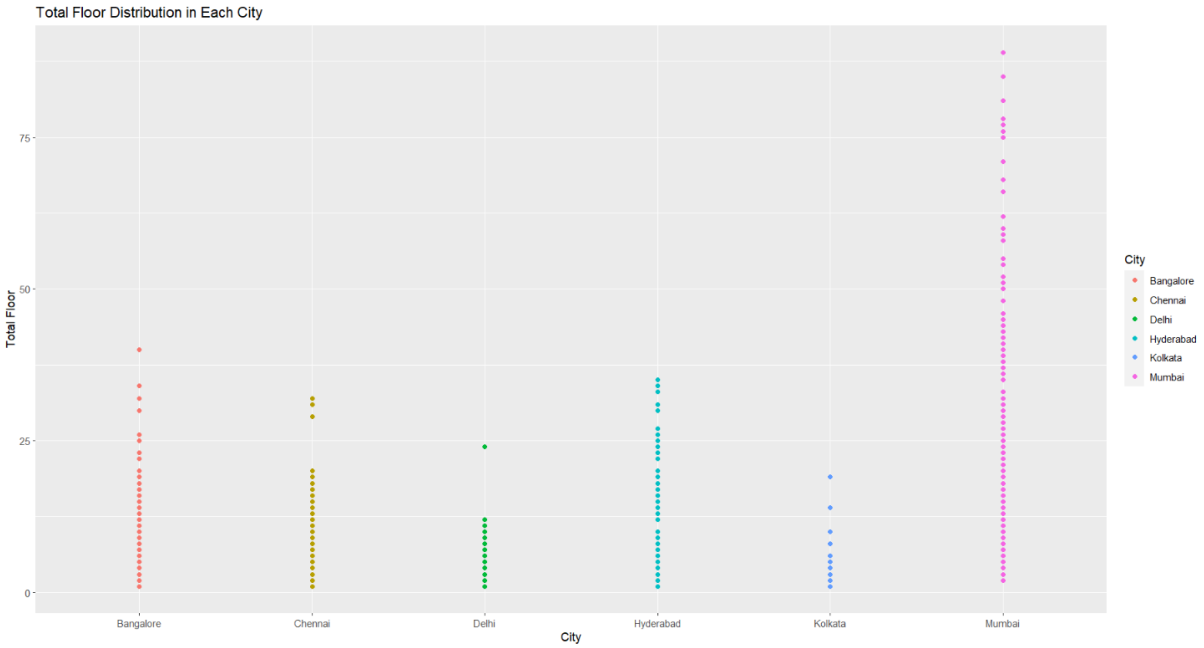


Figure 4-109: Total Floor Distribution in Each City

Based on the scatterplot above, the distribution for total floor of house for rent in Mumbai has larger range as compared to other cities. The total floor ranging all on the scale proves the varieties of house type in Mumbai from row houses to luxury apartments. Due to the dense population in Mumbai, the variation of house types are diverse so that the people can have more options when looking for a place to live. Bangalore, Chennai, and Hyderabad have slightly higher overall total floor range than Delhi and Kolkata according to the scatterplot.

4.6.5 Analysis 6-5: The difference in floor distribution for each city

This analysis aims to provide insight on the distribution for the total floor of the houses for rent in each city. Violin plot is selected for this analysis as it can visualize the density of the distribution effectively.

```
# Analysis 6-5: Difference in Floor for City (highest and lowest)
df %>%
  ggplot(mapping = aes(x = House.City, y = Difference.Floor, fill = House.City)) +
    scale_fill_brewer(palette = "Accent") +
    geom_violin() +
    labs(
      title = "Difference in Floor Distribution in Each City",
      x = "City",
      y = "Total Floor",
      fill = "City"
    ) +
    stat_summary(fun.y = median, geom = "point", color = "black")
```

Figure 4-110: Code to Plot Difference in Floor Distribution in Each City

The code above constructs violin plot with “ggplot()” and “geom_violin()” functions after setting all required adjustments such as title and labels.

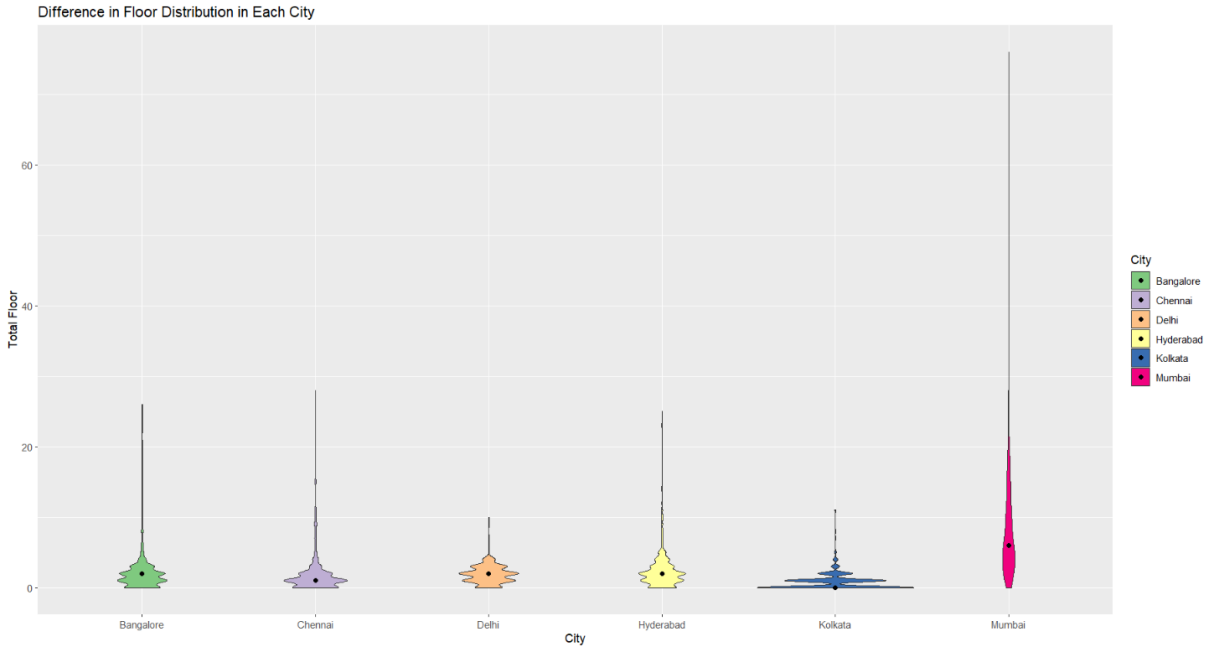


Figure 4-111: Difference in Floor Distribution in Each City

According to violin plot above, the range of difference in floor for Mumbai is higher than that from other cities. The distribution for the difference in floor for Bangalore, Chennai, and Hyderabad is similar. As for Mumbai, the total floor range is higher than that from other cities. The range of difference in floor for Delhi and Kolkata is also similar but Kolkata has spikes in range of the difference in floor while Delhi’s distribution is spread more evenly. In short, Mumbai has more variation in the distribution of difference in floor followed by Chennai, Bangalore, Hyderabad then Delhi and Kolkata.

4.6.6 Analysis 6-6: Common house furnishing in each city

This analysis aims to provide insight on the common house furnishing in each city by looking at the percentage of the distribution of each house furnishing category. This is achieved using percent stacked bar chart as it can effectively visualize the percentage of sub-category by many categories.

```
# Analysis 6-6: Common house furnishing each city
df %>%
  group_by(House.City, House.Furnishing) %>%
  summarize(frequency = n()) %>%
  mutate(percentage = calculate_percent(frequency)) %>%
  ggplot(mapping = aes(x = House.City, y = frequency, fill = House.Furnishing)) +
    geom_bar(stat = "identity", position = "fill", width = 0.5) +
    geom_text(aes(label = paste0(percentage, "%")), size = 3, position = position_fill(vjust = 0.5)) +
    labs(
      title = "Common House Furnishing in Each City",
      x = "City",
      y = "Percentage of Frequency",
      fill = "House Furnishing"
    ) +
    scale_y_continuous(labels = percent) +
    scale_fill_brewer(palette = "Pastel2")
```

Figure 4-112: Code to Plot Common House Furnishing in Each City

The code above groups the data by city and house furnishing then calculate the occurrences of records for each group. New column is computed by calculating the percentage through division of current occurrence with the total occurrence of the group. Then, the graph is plotted with “ggplot()” and “geom_bar()” functions along with some adjustments on the colour and labels.

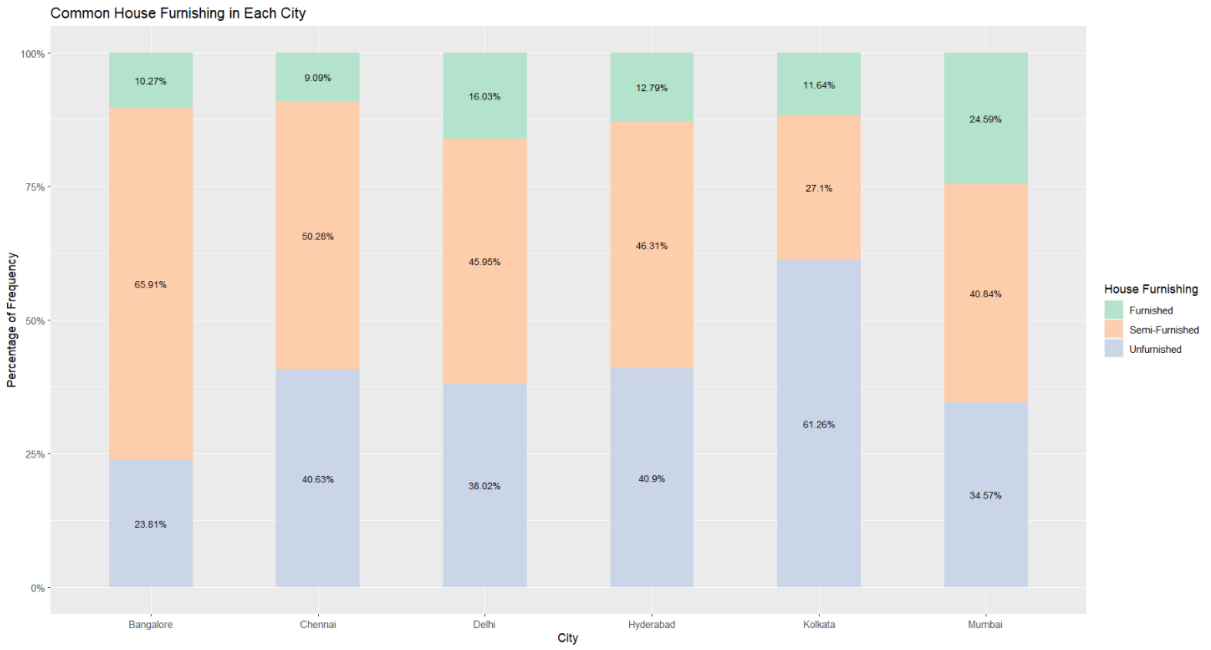


Figure 4-113: Common House Furnishing in Each City

As stated in the bar chart above, semi-furnished houses are the majority in most cities except Kolkata. In Kolkata, unfurnished houses is the most common furnishing type which is 61.26% of the total portion. In most cities, the number of unfurnished houses is higher than that of furnished houses. This may be due to the increase in cost to furnish the house when renting a house to the tenants. However, the number of furnished and unfurnished houses are close which means landlords in Mumbai are willing to furnish their house to attract more tenants.

4.6.7 Conclusion

To conclude, most of the rental posting does not target specific group in the 6 cities in dataset so that tenants can find their house to rent easily. Ghatkesar, NH 2 2 in Hyderabad has the highest rental price per unit which is 1500 Rupee per unit while Uppal, NH 2 2 in Hyderabad has the lowest rental price per unit which is 0.57 Rupee per unit. This means Hyderabad has the house with lowest and highest rental price per unit for rent. For total floor range, Mumbai has the highest total floor variations followed by Bangalore, Chennai, and Hyderabad then Delhi and Kolkata. This order also applies to the difference in floor distribution where Mumbai has the highest difference in floor distribution. In Mumbai, more furnished house in the rental posting as compared to other cities. This may indicate that the tenants in

Mumbai prefer furnished houses. However, unfurnished houses are still the majority among rental postings in all the cities mentioned in the dataset.

5.0 Additional Features

5.1 Additional Feature 1: with()

```
# Data Transformation
# calculate rent / size
df$Rent.Per.Unit <- df %>% with(round(Rental.Price /House.Size, digit = 2))
```

Figure 5-1: "with()" Function

The "with()" function is an alternative to "mutate()" function that helps to compute a new column according to the expression given. In the example above, the dataframe is passed into "with()" function then the new column is calculated by dividing the rental price with house size and round up to 2 decimal places with "round()" function.

```
> df %>% with(round(Rental.Price /House.Size, digit = 2))
  [1] 144.44  15.00  15.56  28.57  15.00  12.31  21.25   8.12  10.67  12.00  13.85  11.11  14.29
 [14]  11.00  27.27  13.85  16.67  20.00  14.44  13.00  10.00  27.27  46.15  15.00   7.50  39.35
 [27]  20.41  30.00  55.56  16.00  23.75  24.00  20.00  17.65  16.67  28.57  20.00   9.38  15.00
 [40]  14.29  18.86  52.63  12.00  75.47  66.67  60.71  39.00  19.84  12.50  16.67  18.67  17.39
 [53]  13.33  17.50  10.00  21.43  21.67  33.33  22.22  27.50  17.14  10.91  18.00  11.67  15.00
 [66]   9.01  10.00  12.50  18.75  20.00  12.50  11.11  61.54  40.00  52.63  41.67  66.67  32.31
 [79]  49.09  25.00  44.44  56.34  53.33  41.95  40.00  57.47  33.33  16.92  40.00  69.52  62.50
 [92]  51.56  57.14  66.67  80.79  40.06  21.58   8.57  13.33  20.00  27.78  24.15  16.00  13.00
[105]  12.83  18.00  10.00  16.67  14.40  15.00  18.52  13.33  16.30  21.18  66.67   6.82  13.91
[118]  19.16  14.17  23.08  37.14  18.33  17.50  31.25 150.00  22.00  16.67 300.00 187.50  61.54
[131]  19.23 140.00 133.33  16.67  36.36  11.25  15.79 130.00  44.44  14.00 136.84  11.11  22.00
[144]  14.44 125.00  33.33  75.47  18.00  15.00  11.11  20.00  22.22  20.00 185.71  25.00  22.22
[157]  21.28  17.50 150.00  24.69 109.38  27.78 200.00 200.00 238.10 183.33  25.56  13.33   8.82
[170]  16.25  11.82  12.77  11.00  17.93  18.57  18.57  33.85  12.14  30.77  12.31  15.17  12.00
[183]  17.00  14.12  80.00   6.67  21.67  16.00  10.00  40.00  17.78  12.86  10.00   6.25  15.00
[196]  13.33 130.00  18.89  17.50  18.92  13.33  15.00   9.35  10.83  25.00  23.00  13.33   8.75
[209]  14.42   7.69  16.28  12.00  12.50   7.37  14.29  11.00  10.29  12.94   7.50  11.58  26.67
[222]  15.00  11.36  23.33  90.00  15.62  16.00  13.33  68.63  11.54  15.00   7.78  10.81  15.00
[235]  25.00 266.67  11.14  15.38  16.43  16.07  19.23  10.50  23.00  14.62  16.43  20.00  25.71
[248]  21.54  10.00  18.00  10.91  15.00  11.82  10.00  10.59  29.59  27.42  26.67  20.41  16.67
```

Figure 5-2: Snippet of Output for "with()" function

The output with "with()" function is as illustrated as above. The computed output will be assigned to a new column of the original dataframe for data analysis later.

5.2 Additional Feature 2: str_split_fixed()

```
# parse floor to numbers, then calculate distance
df[c("Current.Floor", "Total.Floor")] <- str_split_fixed(df$Floor, pattern=" out of ", 2)
```

Figure 5-3: "str_split_fixed()" function

The "str_split_fixed()" function from stringr package is used to split the string into character matrix with n number of columns given the pattern of delimiter (RDocumentation,

2022). In the code above, the first argument of the function is the data column that the splitting of string is applied on. The pattern argument represents the pattern of the delimiter the function is finding to split the string into 2 columns.

```
> another_df$Floor
  [1] "Ground out of 2"          "1 out of 3"               "1 out of 3"               "1 out of 2"
  [5] "1 out of 2"               "Ground out of 1"          "Ground out of 4"          "1 out of 2"
  [9] "1 out of 2"               "1 out of 3"               "1 out of 4"               "1 out of 1"
 [13] "1 out of 4"               "1 out of 2"               "Ground out of 2"          "1 out of 1"
 [17] "1 out of 2"               "Ground out of 2"          "Ground out of 3"          "1 out of 2"
 [21] "1 out of 2"               "2 out of 3"               "Ground out of 4"          "Ground out of 2"
 [25] "1 out of 2"               "1 out of 1"               "1 out of 1"               "Ground out of 3"
 [29] "4 out of 5"               "Ground out of 2"          "Ground out of 2"          "Ground out of 2"
 [33] "1 out of 1"               "Ground out of 2"          "2 out of 3"               "1 out of 2"
 [37] "2 out of 2"               "Ground out of 3"          "1 out of 2"               "1 out of 2"
 [41] "Ground out of 1"          "Ground out of 2"          "2 out of 5"               "4 out of 14"
 [45] "Ground out of 3"          "Ground out of 1"          "3 out of 3"               "Ground out of 1"
 [49] "Ground out of 2"          "Ground out of 3"          "1 out of 1"               "2 out of 2"
 [53] "Ground out of 1"          "Ground out of 2"          "5 out of 5"               "Ground out of 2"
 [57] "1 out of 1"               "2 out of 2"               "Ground out of 3"          "4 out of 4"
 [61] "1 out of 3"               "Ground out of 2"          "7 out of 8"               "1 out of 3"
 [65] "Ground out of 4"          "Ground out of 3"          "Ground out of 2"          "Ground out of 2"
 [69] "Ground out of 2"          "1 out of 3"               "Ground out of 2"          "1 out of 4"
 [73] "1 out of 2"               "Ground out of 2"          "Ground out of 2"          "1 out of 4"
 [77] "Ground out of 2"          "2 out of 4"               "Ground out of 2"          "1 out of 2"
 [81] "Ground out of 2"          "1 out of 3"               "Ground out of 3"          "1 out of 1"
 [85] "1 out of 2"               "2 out of 3"               "Ground out of 2"          "4 out of 4"
 [89] "1 out of 3"               "2 out of 3"               "Ground out of 4"          "Ground out of 2"
```

Figure 5-4: Before Splitting the String

Before splitting the string, the floor column of the original dataframe is as shown in the image above. The common delimiter found across the records is “ out of “ that separates the current floor and total floor.

```
> str_split_fixed(another_df$Floor, pattern=" out of ", 2)
        [,1]           [,2]
   [1,] "Ground"       "2"
   [2,] "1"            "3"
   [3,] "1"            "3"
   [4,] "1"            "2"
   [5,] "1"            "2"
   [6,] "Ground"       "1"
   [7,] "Ground"       "4"
   [8,] "1"            "2"
   [9,] "1"            "2"
  [10,] "1"            "3"
  [11,] "1"            "4"
  [12,] "1"            "1"
  [13,] "1"            "4"
  [14,] "1"            "2"
  [15,] "Ground"       "2"
  [16,] "1"            "1"
  [17,] "1"            "2"
  [18,] "Ground"       "2"
  [19,] "Ground"       "3"
  [20,] "1"            "2"
  [21,] "1"            "2"
  [22,] "2"            "3"
  [23,] "Ground"       "4"
  [24,] "Ground"       "2"
  [25,] "1"            "2"
  [26,] "1"            "1"
  [27,] "1"            "1"
```

Figure 5-5: Snippet of Output for "str_split_fixed()" Function

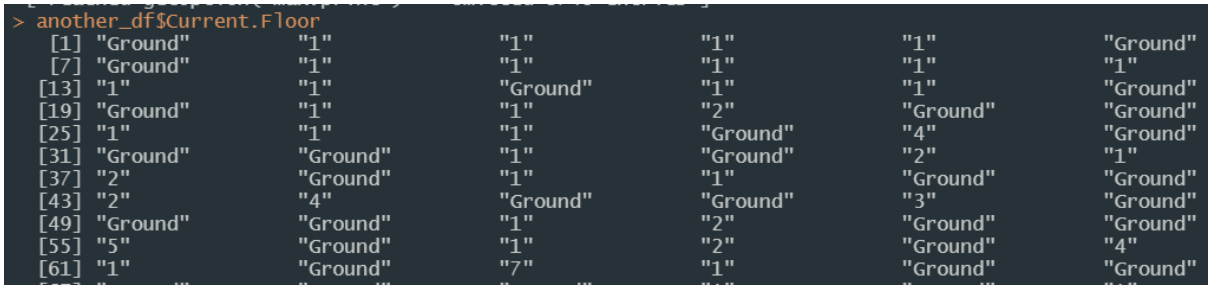
The output string after splitting the floor column of the dataframe is as illustrated in the image above. The returned character matrix is then used to create new columns such as “Current.Floor” and “Total.Floor”.

5.3 Additional Feature 3: str_replace_all()

```
df$Current.Floor <- str_replace_all(df$Current.Floor, pattern="Ground", replacement="0")
df$Current.Floor <- str_replace_all(df$Current.Floor, pattern="Upper Basement", replacement="-1")
df$Current.Floor <- str_replace_all(df$Current.Floor, pattern="Lower Basement", replacement="-2")
```

Figure 5-6: "str_replace_all()" Function

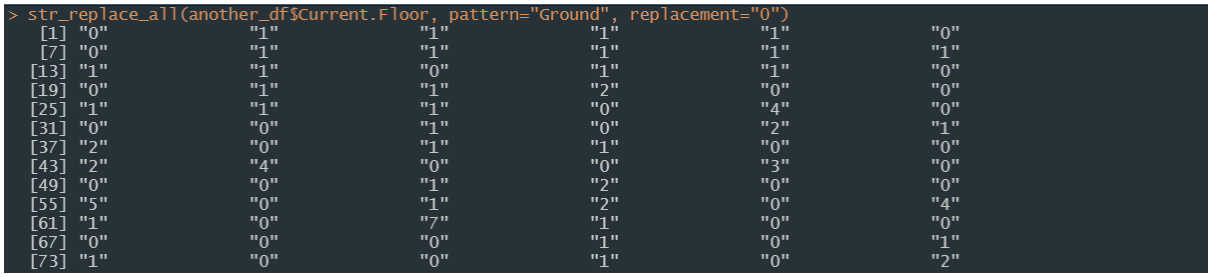
The “str_replace_all() from stringr package is used to replace the part of string given the pattern with a new string value. This is useful when the current floor with text values need to be replaced with a numeric text so that it can be converted into numeric data type later. The first argument represents the column that this function is applied on while pattern argument is the pattern the function is looking for when replacing string. The replacement value is the values that will be used to replace the pattern found.



```
> another_df$Current.Floor
  [1] "Ground"      "1"           "1"           "1"           "1"           "Ground"
  [7] "Ground"      "1"           "1"           "1"           "1"           "1"
 [13] "1"           "1"           "Ground"      "1"           "1"           "Ground"
 [19] "Ground"      "1"           "1"           "2"           "Ground"      "Ground"
 [25] "1"           "1"           "1"           "Ground"      "4"           "Ground"
 [31] "Ground"      "Ground"      "1"           "Ground"      "2"           "1"
 [37] "2"           "Ground"      "1"           "1"           "Ground"      "Ground"
 [43] "2"           "4"           "Ground"      "Ground"      "3"           "Ground"
 [49] "Ground"      "Ground"      "1"           "2"           "Ground"      "Ground"
 [55] "5"           "Ground"      "1"           "2"           "Ground"      "4"
 [61] "1"           "Ground"      "7"           "1"           "Ground"      "Ground"
```

Figure 5-7: Current Floor Values with Text Data

The image above shows the current floor values that has text data type which is an obstacle when converting the current floor into numeric data type.

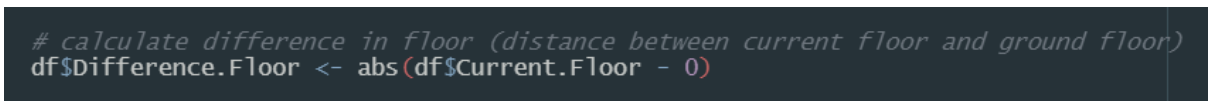


```
> str_replace_all(another_df$Current.Floor, pattern="Ground", replacement="0")
  [1] "0"           "1"           "1"           "1"           "1"           "0"
  [7] "0"           "1"           "1"           "1"           "1"           "1"
 [13] "1"           "1"           "0"           "1"           "1"           "0"
 [19] "0"           "1"           "1"           "2"           "0"           "0"
 [25] "1"           "1"           "1"           "0"           "4"           "0"
 [31] "0"           "0"           "1"           "0"           "2"           "1"
 [37] "2"           "0"           "1"           "1"           "0"           "0"
 [43] "2"           "4"           "0"           "0"           "3"           "0"
 [49] "0"           "0"           "1"           "2"           "0"           "0"
 [55] "5"           "0"           "1"           "2"           "0"           "4"
 [61] "1"           "0"           "7"           "1"           "0"           "0"
 [67] "0"           "0"           "0"           "1"           "0"           "1"
 [73] "1"           "0"           "0"           "1"           "0"           "2"
```

Figure 5-8: Replaced "Ground" value with "0"

The images above shows that the “Ground” value is replaced with “0” through “str_replace_all()” function. This is also applied to replace other values like “Upper Basement” in the current floor values with a valid numeric value.

5.4 Additional Feature 4: abs()



```
# calculate difference in floor (distance between current floor and ground floor)
df$Difference.Floor <- abs(df$Current.Floor - 0)
```

Figure 5-9: “abs()” Function

The “abs()” function is used to calculate the absolute of a numeric value where positive values have no changes while negative values are converted to positive values. This is used when calculating the absolute difference in floor as it cannot contains negative value.

5.5 Additional Feature 5: floor_date()

```
# group data by city and month first
grouped_by_city_date <- df[, c("Posted.Date", "Rental.Price", "House.City")]

grouped_by_city_date <-
grouped_by_city_date %>%
  group_by(Month = lubridate::floor_date(Posted.Date, unit = "month"), House.City) %>%
  summarize(Median.Rental.Price = median(Rental.Price))
```

Figure 5-10: "floor_date()" Function

The “floor_date()” function from lubridate package is used to rounds down the date time object to the nearest boundary given a time unit (Lubridate Tidyverse, n.d.). For example, in the image above, the posted date of rental posting is rounded down to the nearest month. If the posted date is 15th April 2022, then the posted date will be rounded down to 1st April 2022 when grouping.

```
   Month      House.City Median.Rental.Price
   <date>     <chr>                    <dbl>
 1 2022-04-01 Bangalore                14000
 2 2022-04-01 Chennai                   8750
 3 2022-04-01 Delhi                    13000
 4 2022-04-01 Hyderabad                12500
 5 2022-04-01 Kolkata                  10000
 6 2022-04-01 Mumbai                   27000
 7 2022-05-01 Bangalore                13000
 8 2022-05-01 Chennai                  14000
 9 2022-05-01 Delhi                    16000
10 2022-05-01 Hyderabad                12000
# … with 14 more rows
```

Figure 5-11: Output for "floor_date()" Function

As shown in the image above, all posted dates are rounded down to their nearest boundary so that aggregations of data by months can be achieved.

5.6 Additional Feature 6: labs()

```
facilities_number <- unique(df$Facilities.Number)
ggplot(data = rent_by_facilities, mapping = aes(x = Facilities.Number, y = Median.Rental.Price)) +
  labs(
    title = "Median of Rental Price by Facilities Numbers",
    x = "Median of Rental Price",
    y = "Facilities Numbers",
    fill = "Median of Rental Price"
  ) +
  geom_bar(mapping = aes(fill = Median.Rental.Price), stat = "identity", col="#544c4b") +
  scale_fill_gradient(low = "green", high = "red", labels = comma) +
  scale_x_continuous("Facility Number", labels = as.character(facilities_number), breaks = facilities_number) +
  scale_y_continuous("Median Rental Price", labels = comma) +
  coord_flip()
```

Figure 5-12: "labs()" Function

The “labs()” function from ggplot is used to adjust the labels such as title, x-axis label, and y-axis label in cleaner manner as every label is adjusted in single function. In the example above, the plot labels including legend title can be adjusted with the respective arguments.

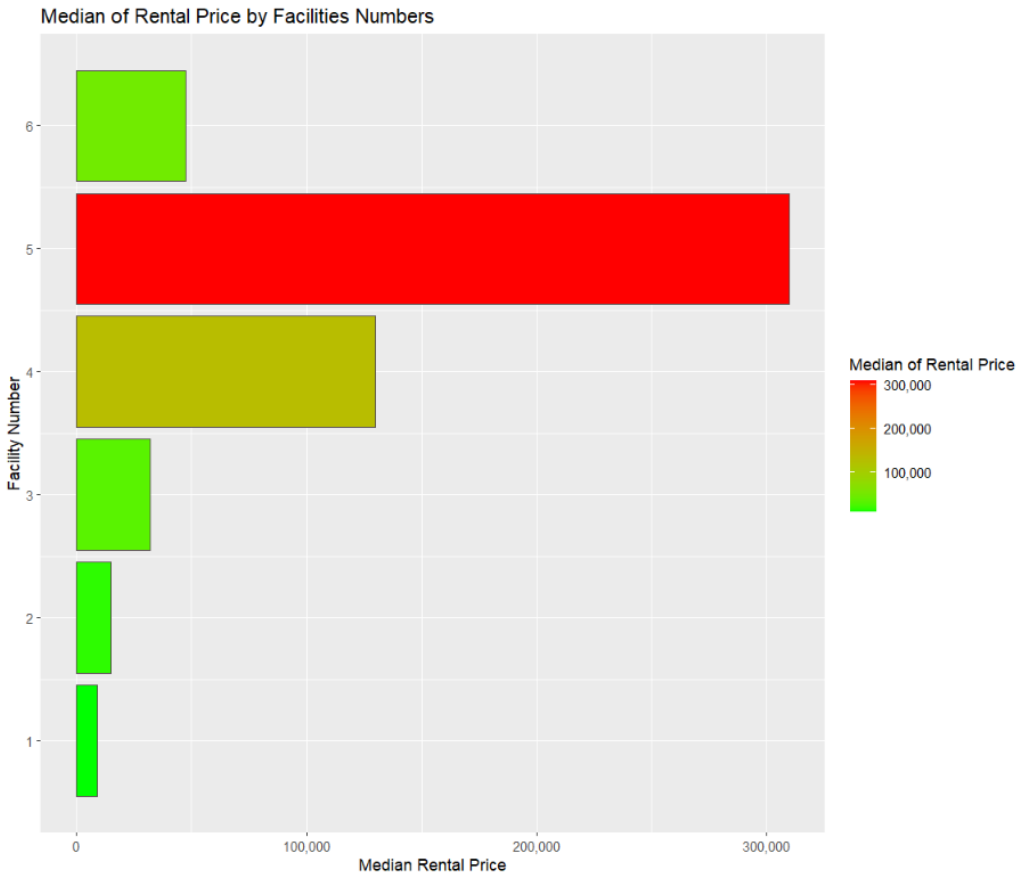


Figure 5-13: Output of "lab()" Function

As shown in the image, the plot title, x-axis title, y-axis title and legend title is set correctly.

5.7 Additional Feature 7: scale_y_continuous()

```
# plot a grouped bar chart
ggplot(data=grouped_city_furnishing, mapping=aes(fill = House.Furnishing, x = House.City, y = Agg.Rental)) +
  ggtitle("Rental Price in each City by Furnishing Status") +
  xlab("City") +
  ylab("Rental Price per Unit") +
  guides(fill=guide_legend(title="Furnishing Status")) +
  geom_bar(position = "dodge", stat = "identity", col="black", width=0.45) +
  scale_y_continuous(labels = comma) +
  coord_flip()
```

Figure 5-14: "scale_y_continuous()" Function

The “scale_y_continuous()” functions provide simple manipulation of y-axis parameters such as labels, breaks, and limits. In the example above, the function is commonly used to convert the number in scientific notation into comma separated format in y-axis labels for greater readability.

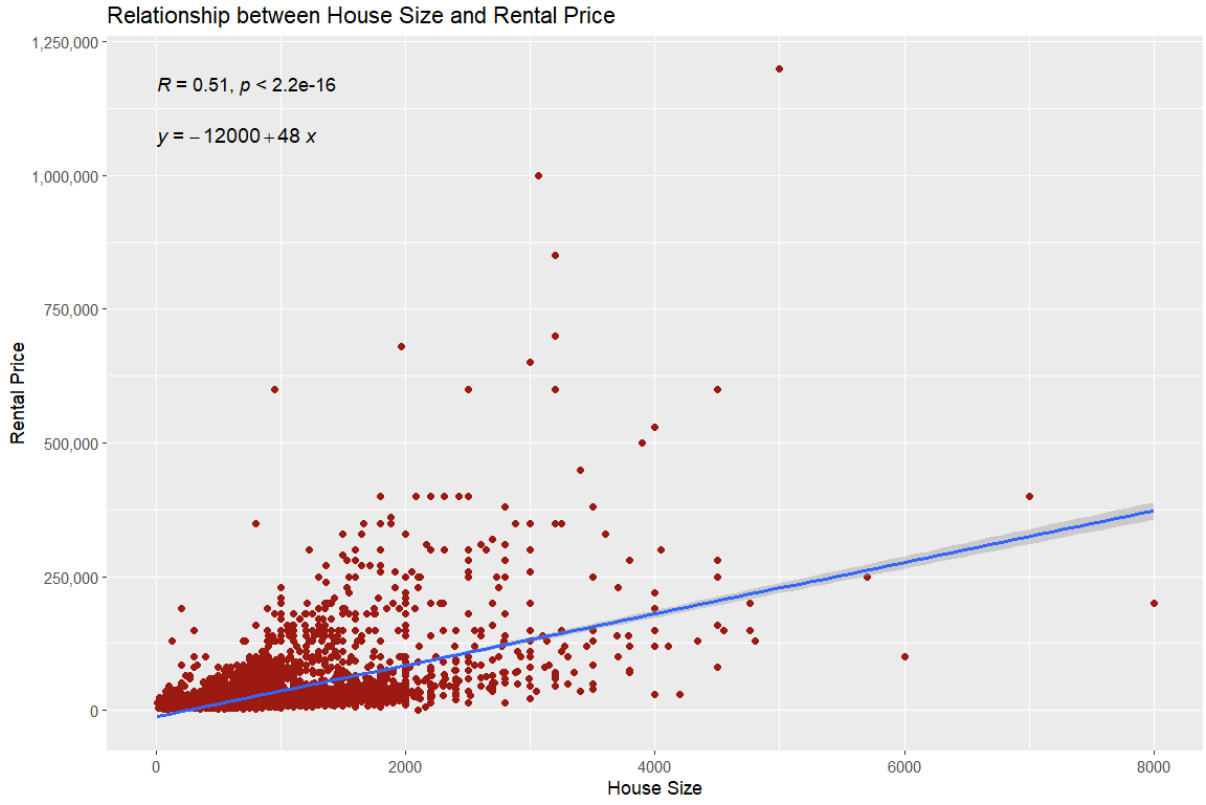


Figure 5-15: Output of "scale_y_continuous()" Function

5.8 Additional Feature 8: scale_x_continuous()

```
ggplot(data = grouped_bachelor_bathroom, mapping = aes(x = Bathroom.Number, y = frequency)) +
  labs(title = "Bathroom Distribution of Houses Targeting Bachelors", x = "Bathroom Number", y = "Frequency") +
  geom_bar(stat = "identity", col = "#27408B", fill = "#4169E1") +
  geom_text(aes(label = frequency), hjust = -0.2) +
  scale_x_continuous(labels = unique(bachelors_df$Bathroom.Number), breaks = unique(bachelors_df$Bathroom.Number)) +
  coord_flip()
```

Figure 5-16: "scale_x_continuous()" Function

The “scale_x_continuous()” function is like the “scale_y_continuous()” function except that it controls the adjustments of x-axis. In the example above, it is used to set the labels and breaks of the x-axis. This is often used to adjust the label text to improve readability of the plot.

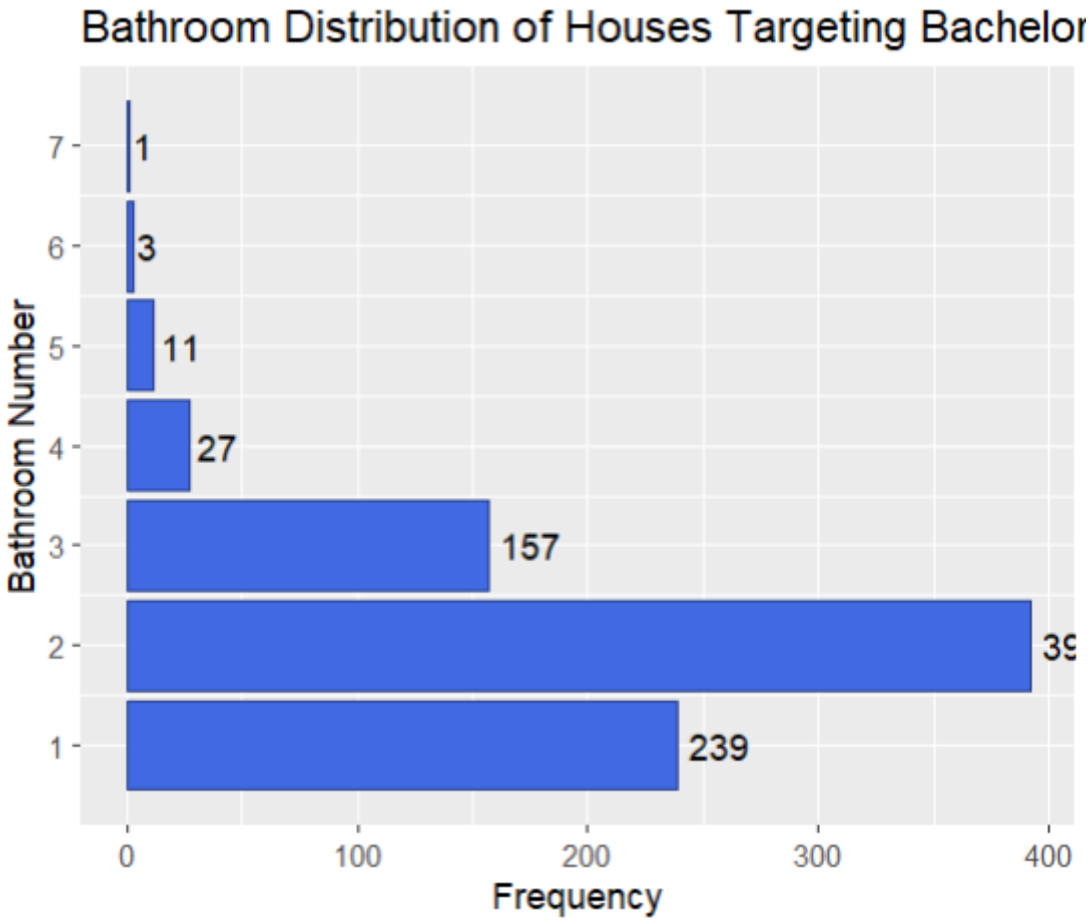


Figure 5-17: Output of "scale_x_continuous()" Function

As shown in the image above, all distinct values of bathroom number is shown on the y-axis as the plot undergoes “coord_flip()” function.

5.9 Additional Feature 9: rollmean()

```
df %>%
  group_by(Posted.Date) %>%
  summarize(Rental.Price = mean(Rental.Price)) %>%
  mutate(Smoothed.Rental = rollmean(Rental.Price, k = 10, fill = NA, align = "center")) %>%
  ggplot() +
    labs(title = "Trend of Rental Price", x = "Posted Date", y = "Rental Price", colour = "Lines") +
    geom_line(mapping = aes(x = Posted.Date, y = Rental.Price, colour = "Rental Price"), size = 1.0) +
    geom_line(mapping = aes(x = Posted.Date, y = Smoothed.Rental, colour = "Smoothed Rental Price"), size = 1.0) +
    scale_y_continuous(labels = comma)
```

Figure 5-18: "rollmean()" Function

The “rollmean()” function from “zoo” package is implemented to calculate the rolling average of the data to smoothen the line in line plot so that the trend can be identified and emphasized effectively. The k arguments is the window size that determines the number of records used to calculate the rolling average while fill argument decides the values to fill the missing values after computing the rolling average. The align argument decides the orientation of the index of results. The output is shown at analysis 1-1 in the graph displayed.

5.10 Additional Feature 10: is.null()

```
> sum(is.null(df))
[1] 0
>
```

Figure 5-19: "is.null()" Function

The “is.null()” function is used to find the null values in the dataset so that missing values can be identified and handled appropriately. When it is used with the “sum()” function, the total number of missing values can be calculated.

5.11 Additional Feature 11: coord_flip()

```
# sort in descending order and get the top 10 (which is the most expensive 10)
top20_expensive <- head(grouped_data[order(-grouped_data$agg), ], 20)
ggplot(data = top20_expensive, mapping = aes(x = reorder(House.Locality, +agg), y = agg)) +
  labs(
    title = "Top 20 Expensive Rental Price per Unit by Locality",
    x = "Localities",
    y = "Rental Price per Unit"
  ) +
  geom_bar(stat = "identity", mapping = aes(fill = agg), width = 0.5) +
  scale_fill_viridis("Median of Rent Per Unit Scale", option = "D") +
  geom_text(aes(label = agg), hjust = -0.2) +
  coord_flip()
```

Figure 5-20: "coord_flip()" Function

The “coord_flip()” function is used to flip the cartesian coordinates where the vertical turns to horizontal and vice versa. This is very useful when converting vertical bar chart into horizontal bar chart. Analysis 6-2 shows that the x-axis and y-axis has been swapped to form a horizontal bar chart.

5.12 Additional Feature 12: geom_smooth()

```
# plot scatterplot and line of best fit
ggplot(data = df, mapping = aes(x = House.Size, y = Rental.Price, add = "reg.line")) +
  labs(title = "Relationship between House Size and Rental Price", x = "House Size", y = "Rental Price") +
  geom_point(col = "#9c1a11") +
  geom_smooth(method = lm) +
  scale_y_continuous("Rental Price", labels = comma) +
  stat_cor() +
  stat_regline_equation(label.y.npc = .9, size = 4, col = "black")
```

Figure 5-21: "geom_smooth()" Function

The “geom_smooth()” function is essentially used to add a trend line to the existing plot, usually scatterplot to show the relationship between variables. There are many methods that can be used to model the trendline but in this example “lm” which stands for linear model is used to model the relationship in linear form. The output is shown at analysis 1-4 which shows the regression line is plotted on the chart.

5.13 Additional Feature 13: stat_regline_equation()

```
# plot scatterplot and line of best fit
ggplot(data = df, mapping = aes(x = House.Size, y = Rental.Price, add = "reg.line")) +
  labs(title = "Relationship between House Size and Rental Price", x = "House Size", y = "Rental Price") +
  geom_point(col = "#9c1a11") +
  geom_smooth(method = lm) +
  scale_y_continuous("Rental Price", labels = comma) +
  stat_cor() +
  stat_regline_equation(label.y.npc = .9, size = 4, col = "black")
```

Figure 5-22: "stat_regline_equation()" Function

The “stat_regline_equation()” is used to add the equation of regression line on the plot for reference. The position of the equation can be adjusted with arguments such as “label.x.npc” and “label.y.npc”. As shown in analysis 1-4, the equation is displayed on the graph.

5.14 Additional Feature 14: Grouped Bar Chart

```
# Analysis 4-6: Tenant Targeted against Date
df %>%
  group_by(month = lubridate::floor_date(Posted.Date, unit = "month"), Tenant.Targeted) %>%
  summarize(frequency = n()) %>%
  ggplot(data = grouped_tenant_date, mapping = aes(x = month, y = frequency, fill = Tenant.Targeted)) +
    labs(title = "Trend of Tenant Targeted against Date", x = "Month", y = "Frequency of Tenant Targeted") +
    geom_bar(stat = "identity", position = "dodge", width = 10, col = "#424242") +
    scale_fill_brewer(palette = "Accent")
```

Figure 5-23: Set "position" Argument to "dodge" Value in "geom_bar()" Function

The “position” argument is set to “dodge” usually when the bar of bar chart needs to be aligned side by side according to the sub-category. The final chart outputted is also known as grouped bar chart which can visualize the values of sub-category by main category effectively. In analysis 4-6, the grouped bar chart is constructed when position argument is set to “dodge”.

5.15 Additional Feature 15: Stacked Bar Chart

```
# plot a stack bar chart
ggplot(data=grouped_city_furnishing, mapping=aes(fill = House.Furnishing, x = House.City, y = Agg.Rental)) +
  ggtitle("Rental Price in each City by Furnishing Status") +
  xlab("City") +
  ylab("Rental Price per Unit") +
  guides(fill=guide_legend(title="Furnishing Status")) +
  geom_bar(position = "stack", stat = "identity", col="black", width=0.45) +
  geom_text(aes(label = Agg.Rental), position = position_stack(vjust = 0.5)) +
  scale_fill_brewer(palette = "Dark2") +
  scale_y_continuous(labels = comma)
```

Figure 5-24: Set "position" Argument to "stack" Value in "geom_bar()" Function

The “position” argument is set to “stack” when a stack bar chart wants to be plotted. The sub-category of the bar chart is stacked on the previous sub-category to visualize the distribution of each main category while comparing the values among main categories. The output is shown at analysis 1-8.

5.16 Additional Feature 16: Percent Stack Bar Chart

```
ggplot(data = grouped_tenant_furnishing, mapping = aes(fill = House.Furnishing, x = Tenant.Targeted, y = frequency)) +
  ggtitle("Frequency of Houses by Furnishing Status for Each type of Tenant") +
  xlab("Tenant Targetted") +
  ylab("Percentage") +
  guides(fill = guide_legend(title = "House Furnishing")) +
  geom_bar(stat = "identity", position = "fill", width = 0.4) +
  geom_text(aes(label = paste0(percentage, "%")), size = 3, position = position_fill(vjust = 0.5)) +
  scale_y_continuous(labels = percent)
```

sss

Figure 5-25: Set "position" Argument to "fill" Value in "geom_bar()" Function

The “position” argument is set to “fill” when a percent stack bar chart is required to be plotted. Like stack bar chart, the sub-category of bar chart is stacked on the previous sub-

category, but the bar of main categories are scaled to fit the y-axis between 0 and 1 to emphasize the percentage of each sub-category. The output is shown at analysis 3-1.

5.17 Additional Feature 17: guides()

```
ggplot(data = grouped_city_date, mapping = aes(x = month, y = frequency, fill = House.City)) +
  labs(title = "Trend of Rental House Posted in City against Date", x = "Month", y = "Frequency of Rental House Posted") +
  guides(fill=guide_legend(title="House City")) +
  geom_bar(stat = "identity", position = "dodge", width = 20, col = "#424242") +
  scale_fill_brewer(palette = "Set2")
```

Figure 5-26: "guides()" Function

The “guides()” function can be used along with “guide_legend()” function to adjust the parameters of the legend such as title. In the example above, the “fill” argument of “guides()” function is set to the values returned from “guide_legend()” function which in results set the title of the legend. The title of the legend at analysis 4-4 is adjusted as defined.

5.18 Additional Feature 18: Violin Plot

```
# Analysis 6-5: Difference in Floor for City (highest and lowest)
df %>%
  ggplot(mapping = aes(x = House.City, y = Difference.Floor, fill = House.City)) +
    scale_fill_brewer(palette = "Accent") +
    geom_violin() +
    labs(
      title = "Difference in Floor Distribution in Each City",
      x = "City",
      y = "Total Floor",
      fill = "City"
    ) +
    stat_summary(fun.y = median, geom = "point", color = "black")
```

Figure 5-27: "geom_violin()" Function

The “geom_violin()” function is used to plot a violin plot which is a hybrid of boxplot and density plot. Violin plot can be utilized to visualize the statistical summary like boxplot and the density of distribution like density plot. The violin plot constructed at analysis 6-5 is the output of “geom_violin()” function.

5.19 Additional Feature 19: cut()

```
final_df <- df
breaks <- seq(from = -1, to = 90, by = 10)
labels <- c(
  "0 - 9",
  "10 - 19",
  "20 - 29",
  "30 - 39",
  "40 - 49",
  "50 - 59",
  "60 - 69",
  "70 - 79",
  "80 - 89"
)

final_df <-
  df %>%
    mutate(floor_group = cut(Total.Floor, breaks = breaks, labels = labels))
```

Figure 5-28: "cut()" Function

The “cut()” function is often used for categorizing continuous values into different groups. as shown in the image above, the total floor which is numeric value is assigned to groups as defined in “labels” vector. The “breaks” argument indicates that the range of the numeric value and its respective group.

```
> head(final_df[c("Total.Floor", "floor_group")], 20)
     Total.Floor floor_group
1223          11     10 - 19
54             2        0 - 9
245            3        0 - 9
304            2        0 - 9
361            3        0 - 9
400            2        0 - 9
414            2        0 - 9
419            1        0 - 9
188            4        0 - 9
214            1        0 - 9
243            3        0 - 9
246            2        0 - 9
299            3        0 - 9
410            1        0 - 9
110            2        0 - 9
126            1        0 - 9
141            3        0 - 9
184            3        0 - 9
355            4        0 - 9
358            2        0 - 9
```

Figure 5-29: Output of "cut()" Function

The output of “cut()” function is as shown above where the instances are assigned the respective group name from “labels” vector according to their total floor value.

5.20 Additional Feature 20: cor()

```
correlation_matrix <- cor(final_df, method = "pearson")
print(correlation_matrix)
```

Figure 5-30: "cor()" Function

The “cor()” function is used to calculate the correlation coefficient between 2 variables or vectors given the method to use for that calculation. The example above illustrates calculating the correlation coefficient using Pearson coefficient.

```
                  Facilities.Number  House.Size   Area.Type Bathroom.Number Difference.Floor   House.City House.Furnishing
Facilities.Number        1.00000000  0.71614499 -0.15500945      0.79488544       0.22432154  0.056362210       0.07291622
House.Size               0.71614499  1.00000000 -0.08057717      0.74070304       0.17129932  0.147348088       0.08994938
Area.Type               -0.15500945 -0.08057717  1.00000000     -0.18435701      -0.25692304  0.342852452       0.02303896
Bathroom.Number          0.79488544  0.74070304 -0.18435701      1.00000000       0.33178813  0.016737938       0.06371636
Difference.Floor         0.22432154  0.17129932 -0.25692304      0.33178813       1.00000000 -0.330783709       0.01944519
House.City               0.05636221  0.14734809  0.34285245      0.01673794      -0.33078371  1.000000000       0.09227113
House.Furnishing         0.07291622  0.08994938  0.02303896      0.06371636       0.01944519  0.092271131       1.00000000
Tenant.Targeted         -0.04153379 -0.03149964 -0.15576730     -0.05689378      -0.07238134 -0.002354575      -0.03907173
Contact.Person          -0.23164285 -0.21475467  0.56142594     -0.32813342      -0.40585222  0.408899834       0.01377488
Rental.Price             0.46978027  0.51090426 -0.26926438      0.56240572       0.42740316 -0.294166345      -0.04109972
                  Tenant.Targeted Contact.Person Rental.Price
Facilities.Number     -0.04153379    -0.23164285   0.46978027
House.Size            -0.03149964    -0.21475467   0.51090426
Area.Type             -0.15576730     0.56142594  -0.26926438
Bathroom.Number       -0.05689378    -0.32813342   0.56240572
Difference.Floor      -0.07238134    -0.40585222   0.42740316
House.City            -0.00235457     0.40889983  -0.29416634
House.Furnishing      -0.03907173     0.01377488  -0.04109972
Tenant.Targeted        1.00000000    -0.06322798  -0.02981056
Contact.Person        -0.06322798     1.00000000  -0.42618565
Rental.Price          -0.02981056    -0.42618565   1.00000000
```

Figure 5-31: Output of "cor()" Function

The correlation coefficient is calculated and organized in matrix form as illustrated in the image above.

5.21 Additional Feature 21: corrplot()

```
correlation_matrix <- cor(final_df, method = "pearson")
print(correlation_matrix)
corrplot(correlation_matrix, method = 'color', order = 'alphabet', addCoef.col = 'black', col = COL2('PuOr', 10))
```

Figure 5-32: "corrplot()" Function

“corrplot()” function is used to visualize the correlation matrix calculated organized in grid box using various visualization approach such as number, circle, and colour. In the example above, the correlation matrix is plotted where each grid’s colour changes according to the

correlation coefficient as defined in “method” argument. The “order” argument determines how the variables in correlation plot is arranged while “addCoef.col” argument determines whether to include the correlation value in the graph. “col” argument receives colour palette to customize the colouring used in the plot. The correlation plot created at analysis 1-14 is the ouput of “corrplot()” function.

5.22 Additional Feature 22: brewer.pal()

```
pie(
  city_dist_bachelor$frequency,
  paste(unique(city_dist_bachelor$percent), "% - ", unique(city_dist_bachelor$frequency), sep = ""),
  main = "Distribution of Rental Houses in each City Targeting Bachelor",
  col = brewer.pal(length(unique(city_dist_bachelor$percent)), "Set3"),
  border = NA
)
legend(
  1.25, 1,
  title="City",
  legend = unique(city_dist_bachelor$House.City),
  cex = .8,
  fill = brewer.pal(length(unique(city_dist_bachelor$percent)), "Set3")
)
```

Figure 5-33: "brewer.pal()" Function

The “brewer.pal()” function receives the number of colours used and colour palette’s name as the arguments to customize the colouring used in the plot. The colour palette implemented for pie chart at analysis 2-3 is from “brewer.pal()” function.

5.23 Additional Feature 23: after_stat()

```
bachelors_df %>%
  ggplot(mapping = aes(x = Rent.Per.Unit, fill = after_stat(count))) +
  labs(title = "Distribution of Rental Price per Unit Targeting Bachelors", x = "Rent per Unit", y = "Frequency") +
  scale_fill_gradient(low = "lightgreen", high = "springgreen3") +
  geom_histogram(bins = 20)
```

Figure 5-34: "after_stat()" Function

The “after_stat()” function is used to calculate the occurrences of the records right before the plot is rendered. This helps to clean the code as there is no need to aggregate data before plotting.

5.24 Additional Feature 24: n()

```
contact_dist_bachelor <-
  bachelors_df %>%
  group_by(Contact.Person) %>%
  summarize(frequency = n()) %>%
  mutate(percent = calculate_percent(frequency))
```

Figure 5-35: "n()" Function

“n()” function is commonly used to calculate the number of occurrences of a group after grouping the data.

```
> contact_dist_bachelor
# A tibble: 2 × 3
  Contact.Person frequency percent
  <chr>              <int>   <dbl>
1 Agent                434    52.3
2 Owner                396    47.7
>
```

Figure 5-36: Output of "n()" Function

As shown in the image, the number of records for each group is calculated and assigned for the frequency column.

5.25 Additional Feature 25: legend()

```
legend(
  1.25, 1,
  title="Contact Person",
  legend = unique(contact_dist_bachelor$Contact.Person),
  cex = .8,
  fill = brewer.pal(length(unique(contact_dist_bachelor$Contact.Person)), "Set1")
)
```

Figure 5-37: "legend()" Function

The “legend()” function is often used to add legend for the plot after setting the position, title, legend labels. This function is used when adding a legend for the “pie()” function in R does not support adding a legend for the category in default. So, this function is used to customize the legend for the pie chart. The output of the legend is shown in the analysis 2-6.

5.26 Additional Feature 26: Horizontal Adjustment

```
# sort in descending order and get the top 10 (which is the most expensive 10)
top20_expensive <- head(grouped_data[order(-grouped_data$agg), ], 20)
ggplot(data = top20_expensive, mapping = aes(x = reorder(House.Locality, +agg), y = agg)) +
  labs(
    title = "Top 20 Expensive Rental Price per Unit by Locality",
    x = "Localities",
    y = "Rental Price per Unit"
  ) +
  geom_bar(stat = "identity", mapping = aes(fill = agg), width = 0.5) +
  scale_fill_viridis("Median of Rent Per Unit Scale", option = "D") +
  geom_text(aes(label = agg), hjust = -0.2) +
  coord_flip()
```

Figure 5-38: "hjust" Argument

The “hjust” arguments in some functions like “geom_text()” function can be used to adjust the horizontal position of the labels. For example, value of 0.5 will center the position of labels. This helps improve the readability of the plot as the audience can find the labels easily if positioned well.

5.27 Additional Feature 27: Setting Number of Bins for Histogram

```
bachelors_df %>%
  ggplot(mapping = aes(x = Rent.Per.Unit, fill = after_stat(count))) +
  labs(title = "Distribution of Rental Price per Unit Targeting Bachelors", x = "Rent per Unit", y = "Frequency") +
  scale_fill_gradient(low = "lightgreen", high = "springgreen3") +
  geom_histogram(bins = 20)
```

Figure 5-39: Adjusting Number of Bins

As some distributions are extremely skewed to the right, so the number of bins is adjusted so that the distribution can be seen more clearly. This is achieved through setting the “bins” argument to desired number. In analysis 2-5, the number of bins in the histogram is adjusted to 20.

5.28 Additional Feature 28: ggplot(group)

```
# Analysis 3-4: The distribution of rental price by facility number for each type of tenant
df %>%
  ggplot(mapping = aes(x = Facilities.Number, y = Rental.Price, fill = Facilities.Number, group = Facilities.Number)) +
    geom_boxplot() +
    labs(
      title = "Rental Price Distribution by Facility Number According To Tenant Type",
      x = "Facility Number",
      y = "Rental Price",
      fill = "Facility Number"
    ) +
    scale_x_continuous(labels = 1:6, breaks = 1:6) +
    scale_fill_viridis_c(option = "magma") +
    facet_wrap(~Tenant.Targeted)
```

Figure 5-40: "group" Argument

Usually, the grouping of the data is set implicitly if the “group_by()” function is used before plotting. However, this example does not implement “group_by()” function so “group” argument is mapped to a variable in the dataframe passed in to partition the data correctly for accurate analysis. In analysis 3-4, the number of facilities is grouped to visualize the distribution of rental price for each group.

5.29 Additional Feature 29: scale_fill_viridis_c()

```
df %>%
  ggplot(mapping = aes(x = Facilities.Number, y = Rental.Price, fill = Facilities.Number, group = Facilities.Number)) +
    geom_boxplot() +
    labs(
      title = "Rental Price Distribution by Facility Number According To Tenant Type",
      x = "Facility Number",
      y = "Rental Price",
      fill = "Facility Number"
    ) +
    scale_x_continuous(labels = 1:6, breaks = 1:6) +
    scale_fill_viridis_c(option = "magma") +
    facet_wrap(~Tenant.Targeted)
```

Figure 5-41: "scale_fill_viridis_c()" Function

The “scale_fill_viridis_c()” function provides pre-defined colour palette to the plot with various options provided. The gradient colour with magma theme is implemented at analysis 3-4.

option

A character string indicating the color map option to use. Eight options are available:

- "magma" (or "A")
- "inferno" (or "B")
- "plasma" (or "C")
- "viridis" (or "D")
- "cividis" (or "E")
- "rocket" (or "F")
- "mako" (or "G")
- "turbo" (or "H")

Figure 5-42: Options Available for "scale_fill_viridis_c()" Function (ggplot2, n.d.)

5.30 Additional Feature 30: scale_fill_distiller()

```
ggplot(data = df, mapping = aes(x = Difference.Floor, fill = after_stat(count))) +
  labs(
    title = "Distribution of difference in floor for each type of tenants",
    x = "Difference in Floor",
    y = "Frequency"
  ) +
  geom_histogram(bins = 30) +
  scale_fill_distiller(palette = "PuOr") +
  facet_wrap(~Tenant.Targeted)
```

Figure 5-43: "scale_fill_distiller()" Function

Like “scale_fill_viridis_c()” function, “scale_fill-distiller()” function is also function that provides colour palette to the plot. However, this function is only applicable to discrete data like categorical values. This colour palette is implemented at the graph for analysis 3-5.

5.31 Additional Feature 31: stat_summary()

```
ggplot(data = df, mapping = aes(x = Tenant.Targeted, y = House.Size, fill = Tenant.Targeted)) +
  labs(title = "House Size Distribution for Each type of Tenants", x = "Tenant Targeted", y = "House Size") +
  geom_violin() +
  stat_summary(fun.y = median, geom = "point", color = "black")
```

Figure 5-44: "stat_summary()" Function

“stat_summary()” function can calculate the aggregation of data that operates on either x-axis or y-axis. For example, the image above illustrates that the “stat_summary()” function aggregate the values of y-axis’s variable with median function and visualize them on the graph using point plot with black colour. In analysis 3-7, the black point in the violin plot indicates the median of the distribution.

5.32 Additional Feature 32: Subplot of Pie Charts

```
fig_contact_tenant <- plot_ly()
fig_contact_tenant <- fig_contact_tenant %>%
  add_pie(
    data = grouped_city_contact_tenant[grouped_city_contact_tenant$Tenant.Targeted == "Bachelors", ],
    name = "Bachelors",
    value = ~frequency,
    labels = ~Contact.Person,
    title = "Distribution of Contact Person for Bachelors",
    domain = list(row = 0, column = 0)
  )
```

Figure 5-45: Instantiate Plotly and Add Pie Chart

R does not support subplots of various chart in a single plot by default so “plotly” package is installed and imported for that purpose. Firstly, the subplot is instantiated with the “plot_ly()” function and stored inside a variable. Then, “add_pie()” function is called to add pie chart into the subplot instantiated. The position of the pie chart is adjusted by setting value at “domain” argument with a list containing the value of row and column.

```
fig_contact_tenant <- fig_contact_tenant %>% layout(
  title = "Distribution of Contact Person by Tenant Targeted",
  showlegend = TRUE,
  grid = list(rows = 1, columns = 3),
  xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
  yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
  margin = 1
)
fig_contact_tenant
```

Figure 5-46: Setting Layout of Subplot

After adding all the pie charts required, “layout()” function is called and the arguments such as number of rows and columns are set to plot the subplot as desired. The output for this script can be found at analysis 3-8.

5.33 Additional Feature 33: geom_segment()

```
df %>%
  group_by(House.Furnishing, Posted.Date) %>%
  summarize(agg = n()) %>%
  ggplot(mapping = aes(x = Posted.Date, y = agg, col = House.Furnishing)) +
    labs(title = "Trend of Rental Posting by House Furnishing", x = "Posted Date", y = "Frequency") +
    geom_point() +
    geom_segment(mapping = aes(x = Posted.Date, xend = Posted.Date, y = 0, yend = agg)) +
    facet_wrap(~House.Furnishing)
```

Figure 5-47: "geom_segment()" Function

“geom_segment()” function is used to draw straight line between the data points and the coordinates set which is namely “xend” and “yend”. This function is often used along with “geom_point()” to create lollipop chart as ggplot2 package does not provide a specific function for lollipop chart. The lollipop chart at analysis 4-5 is implemented using the script above.

5.34 Additional Feature 34: scale_fill_gradientn()

```
contact_df %>%
  group_by(House.City, Contact.Person) %>%
  summarize(total_count = n()) %>%
  ggplot(mapping = aes(x = House.City, y = total_count)) +
    labs(title = "Count of Rental House that has Agent or Owner as Contact Person in Each City", x = "House City", y = "To
    geom_bar(stat = "identity", position = "dodge", width = 0.5, mapping = aes(fill = total_count)) +
    scale_fill_gradientn(colors = c("#00BFFF", "#1E90FF", "#104E8B")) +
    facet_wrap(~Contact.Person)
```

Figure 5-48: "scale_fill_gradientn()" Function

This function is like other functions that provide colour palette, but it allows the user to control the gradient colour by passing in a vector of colour desired into the function. This enables the granular control over the colour gradient. This colour palette is implemented for the graph at analysis 5-2.

5.35 Additional Feature 35: theme()

```
df %>%
  group_by(House.City, Tenant.Targeted) %>%
  summarize(Total.Tenant = n()) %>%
  mutate(percentage = calculate_percent(Total.Tenant)) %>%
  ggplot(mapping=aes(fill = Tenant.Targeted, x = House.City, y = Total.Tenant)) +
    labs(title = "Distribution of Tenant Preferred in Each City", x = "City", y = "Frequency") +
    geom_bar(position = "fill", stat = "identity", col="black", width=0.45) +
    scale_y_continuous(labels = scales::percent) +
    geom_text(aes(label = paste0(percentage, "%")), size = 3, position = position_fill(vjust = 0.5)) +
    scale_fill_brewer(palette = "Spectral") +
    theme(
      axis.text.x = element_text(angle = 45, vjust = 0.5, hjust = 1),
      axis.title.x = element_text(margin = margin(t=30))
    )
```

Figure 5-49: "theme()" Function

“theme()” function is used to manipulate the graph element such as labels position, line element, and positioning. In the example above, this function is used to the margin and orientation of the x-axis element like x-axis’s label and text. As shown in analysis 6-1, the orientation of the x-axis’s text is adjusted to angled 45 degrees.

5.36 Additional Feature 36: order()

```
# sort in descending order and get the top 20 (which is the most expensive 20)
top20_expensive <- head(grouped_data[order(-grouped_data$agg), ], 20)
ggplot(data = top20_expensive, mapping = aes(x = reorder(House.Locality, +agg), y = agg)) +
  labs(
    title = "Top 20 Expensive Rental Price per Unit by Locality",
    x = "Localities",
    y = "Rental Price per Unit"
  ) +
  geom_bar(stat = "identity", mapping = aes(fill = agg), width = 0.5) +
  scale_fill_viridis("Median of Rent Per Unit Scale", option = "D") +
  geom_text(aes(label = agg), hjust = -0.2) +
  coord_flip()
```

Figure 5-50: "order()" Function

"order()" functions sorts the data by the column defined as the arguments in descending order if the column variable is preceded by "-" symbol. Otherwise, it will sort the data in ascending order by default.

```
> grouped_data[order(-grouped_data$agg), ]
# A tibble: 2,240 × 2
   House.Locality                                  agg
   <chr>                                         <dbl>
 1 Ghatkesar, NH 2 2, Hyderabad                  1500
 2 Lal Darwaza, Hyderabad                        1000
 3 Safdarjung Development Area, Hauz Khas, Delhi  950
 4 Sanath Nagar, NH 9, Hyderabad                  920
 5 Vettuvankeni, Chennai                          632.
 6 Panchsheel Park, Delhi                         500
 7 Mandakini Enclave, Alaknanda, Delhi            450
 8 Green Park, Delhi                              438.
 9 Kamla Nagar, Delhi                             438.
10 Shivalik, Malviya Nagar, Delhi                 425
```

Figure 5-51: Output of the "order()" Function

As shown in the image above, the data is sorted by the "agg" column in descending order.

5.37 Additional Feature 37: reorder()

```
# sort in descending order and get the top 20 (which is the most expensive 20)
top20_expensive <- head(grouped_data[order(-grouped_data$agg), ], 20)
ggplot(data = top20_expensive, mapping = aes(x = reorder(House.Locality, +agg), y = agg)) +
  labs(
    title = "Top 20 Expensive Rental Price per Unit by Locality",
    x = "Localities",
    y = "Rental Price per Unit"
  ) +
  geom_bar(stat = "identity", mapping = aes(fill = agg), width = 0.5) +
  scale_fill_viridis("Median of Rent Per Unit Scale", option = "D") +
  geom_text(aes(label = agg), hjust = -0.2) +
  coord_flip()
```

Figure 5-52: "reorder()" Function

The "reorder()" function is used to order the variable in ascending or descending order sorted by defined key during the plotting of a graph. In the example above, the first argument in the function is the variable that is required to be sorted by the second arguments which is

another variable in the data frame. As shown in analysis 6-2, the bar is ordered by descending order by the rent per unit variable.

5.38 Additional Feature 38: percent

```
df %>%
  group_by(House.City, House.Furnishing) %>%
  summarize(frequency = n()) %>%
  mutate(percentage = calculate_percent(frequency)) %>%
  ggplot(mapping = aes(x = House.City, y = frequency, fill = House.Furnishing)) +
    geom_bar(stat = "identity", position = "fill", width = 0.5) +
    geom_text(aes(label = paste0(percentage, "%")), size = 3, position = position_fill(vjust = 0.5)) +
    labs(
      title = "Common House Furnishing in Each City",
      x = "City",
      y = "Percentage of Frequency",
      fill = "House Furnishing"
    ) +
    scale_y_continuous(labels = percent) +
    scale_fill_brewer(palette = "Pastel2")
```

Figure 5-53: "percent" Function

The “percent” constant from “scales” package provides the function to format the decimals into percentage format. This is usually used in “scale_y_continuous()” function to format the y-axis values so that the decimals is in percentage form for readability.

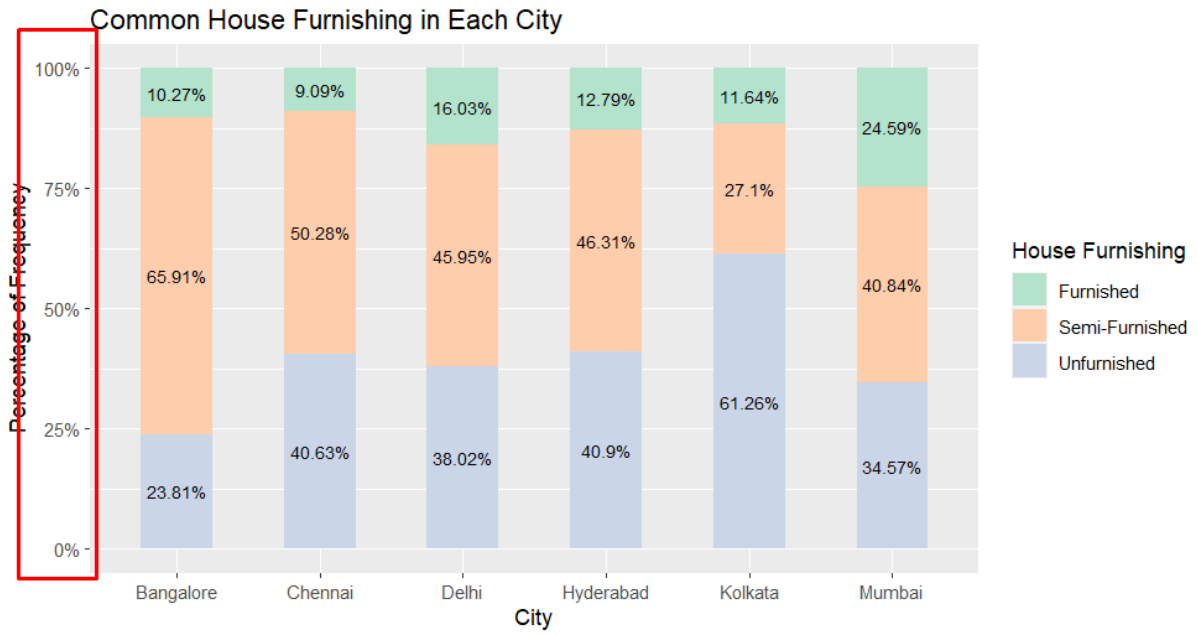


Figure 5-54: Output of Using Percent to Format Decimals

5.39 Additional Feature 39: comma

```
ggplot(data=grouped_city_furnishing, mapping=aes(fill = House.Furnishing, x = House.City, y = Agg.Rental)) +
  ggtitle("Rental Price in each City by Furnishing Status") +
  xlab("City") +
  ylab("Rental Price per Unit") +
  guides(fill=guide_legend(title="Furnishing Status")) +
  geom_bar(position = "stack", stat = "identity", col="black", width=0.45) +
  geom_text(aes(label = Agg.Rental), position = position_stack(vjust = 0.5)) +
  scale_fill_brewer(palette = "Dark2") +
  scale_y_continuous(labels = comma)
```

Figure 5-55: "comma" Function

The “comma” constant from scales package is like “percent” function except that it formats the numbers into numbers separated by comma. For example, “100000” will be formatted to “100, 000” to improve readability. As shown in analysis 1-8, the y-axis’s tick is formatted into comma separated numbers to improve readability.

5.40 Additional Feature 40: geom_text(position = position_fill())

```
# Analysis 6-1 Tenant Distribution in each City
df %>%
  group_by(House.City, Tenant.Targeted) %>%
  summarize(Total.Tenant = n()) %>%
  mutate(percentage = calculate_percent(Total.Tenant)) %>%
  ggplot(mapping=aes(fill = Tenant.Targeted, x = House.City, y = Total.Tenant)) +
    labs(title = "Distribution of Tenant Preferred in Each City", x = "City", y = "Frequency") +
    geom_bar(position = "fill", stat = "identity", col="black", width=0.45) +
    scale_y_continuous(labels = scales::percent) +
    geom_text(aes(label = paste0(percentage, "%")), size = 3, position = position_fill(vjust = 0.5)) +
    scale_fill_brewer(palette = "Spectral") +
    theme(
      axis.text.x = element_text(angle = 45, vjust = 0.5, hjust = 1),
      axis.title.x = element_text(margin = margin(t=30))
    )
```

Figure 5-56: "position_fill()" Function

“position_fill()” function is used to position the text labels of the values for sub-category in a percent stacked bar chart. As shown in the image above, “vjust” argument is set to 0.5 which indicates position the text labels in the center of the sub-category’s bar. In analysis 6-1, the text is well positioned inside the sub-category’s bar for enhanced readability.

5.41 Additional Feature 41: geom_text(position = position_stack())

```
# plot a stack bar chart
ggplot(data=grouped_city_furnishing, mapping=aes(fill = House.Furnishing, x = House.City, y = Agg.Rental)) +
  ggtitle("Rental Price in each City by Furnishing Status") +
  xlab("City") +
  ylab("Rental Price per Unit") +
  guides(fill=guide_legend(title="Furnishing Status")) +
  geom_bar(position = "stack", stat = "identity", col="black", width=0.45) +
  geom_text(aes(label = Agg.Rental), position = position_stack(vjust = 0.5)) +
  scale_fill_brewer(palette = "Dark2") +
  scale_y_continuous(labels = comma)
```

Figure 5-57: "position_stack()" Function

"position_stack" function is like "position_fill()" function which is used to position the text labels of values for the sub-category in a stacked bar chart. The label text for sub-categories of stacked bar chart at analysis 1-8 is well positioned as defined.

5.42 Additional Feature 42: paste0()

```
grouped_area_bachelor <-
  bachelors_df %>%
  group_by(Area.Type) %>%
  summarize(frequency = n()) %>%
  mutate(percent = calculate_percent(frequency)) %>%
  mutate(
    ymax = cumsum(percent),
    ymin = c(0, head(ymax, n = -1)),
    label = paste0(Area.Type, "\nValue: ", frequency, "\nPercentage: ", percent, "%"),
    label_position = (ymax + ymin) / 2
  )
```

Figure 5-58: "paste0()" Function

"paste0()" function is used to combine vectors given with some pre-defined string value after converting all the vectors to character class of vectors. As shown in the image above, all the arguments passed into the function is concatenated to form new string element by element in the vectors.

```
# A tibble: 2 × 7
  Area.Type frequency percent  ymax  ymin label                                      label_position
  <chr>         <int>   <dbl> <dbl> <dbl> <chr>                                              <dbl>
1 Carpet          691    83.2  83.2   0   "Carpet\nValue: 691\nPercentage: 83.25%"           41.6
2 Super           139    16.8 100    83.2 "Super\nValue: 139\nPercentage: 16.75%"           91.6
> grouped_area
```

Figure 5-59: Ouput of Concatenated String

The label column is the results from the concatenation of the arguments passed into "paste0()" function.

5.43 Additional Feature 43: duplicated()

```
> # check duplicated values
> df[duplicated(df), ]
 [1] Posted.Date       Facilities.Number Rental.Price      House.Size        Floor             Area.Type
 [7] House.Locality    House.City        House.Furnishing  Tenant.Targeted   Bathroom.Number   Contact.Person
[13] Rent.Per.Unit     Current.Floor     Total.Floor       Difference.Floor
<0 rows> (or 0-length row.names)
>
```

Figure 5-60: "duplicated()" Function

The “duplicated()” function is used to check whether there are duplicated rows in the dataset during data cleaning process. If there are duplicated rows, it will be shown in the console so that the user can proceed to handle duplicate values.

5.44 Additional Feature 44: stat_cor()

```
# Analysis 1-4: House Size
# plot scatterplot and line of best fit
ggplot(data = df, mapping = aes(x = House.Size, y = Rental.Price, add = "reg.line")) +
  labs(title = "Relationship between House Size and Rental Price", x = "House Size", y = "Rental Price") +
  geom_point(col = "#9c1a11") +
  geom_smooth(method = lm) +
  scale_y_continuous("Rental Price", labels = comma) +
  stat_cor() +
  stat_regline_equation(label.y.npc = .9, size = 4, col = "black")
```

Figure 5-61: "stat_cor()" Function

“stat_cor()” function is used to display the correlation summary such as R-Squared score and the p-values on the graph. In analysis 1-4, the R-Squared and p-values are displayed in the regression graph.

5.45 Additional Feature 45: Doughnut Chart

```
ggplot(data = grouped_area_bachelor, aes(ymax = ymax, ymin = ymin, xmax = 4, xmin = 3, fill = Area.Type)) +
  ggtitle("Distribution of Area Type Percentage Targeting Bachelor") +
  geom_rect() +
  geom_label(x = 3.5, aes(y = label_position, label = label), size = 3, col = "#363636") +
  scale_fill_brewer(palette = "Accent") +
  coord_polar(theta = "y") +
  xlim(c(2, 4)) +
  theme_void() +
  theme(legend.position = "none")
```

Figure 5-62: "geom_rect()" Function

“geom_rect()” function plots the rectangle on the plot based on four parameters which is “ymax”, “ymin”, “xmax”, and “xmin”. This function is used to constructs doughnut chart in this project. “geom_label()” function is implemented to add a bordered labels on the chart to provide necessary information. “coord_polar()” function is used to convert the desired axis into polar Cartesians which is rounded. Then, “theme_void()” function is called to remove unnecessary background for the doughnut chart such as the gridlines and axis values in the background. The doughnut chart is plotted for analysis 2-10.

5.46 Additional Feature 46: Treemap

```
# tree map
facility_dist_bachelor %>%
  mutate(Facilities.Number = as.character(Facilities.Number)) %>%
    ggplot(
      mapping = aes(
        area = frequency,
        fill = Facilities.Number,
        label = paste("Number of Facilities: ", Facilities.Number, "\n", percent, "%", sep = "")
      )
    ) +
    labs(title = "Distribution of Number of Facilities") +
    geom_treemap() +
    geom_treemap_text(size = 10, colour = "black", place = "centre") +
    scale_fill_brewer(palette = "Spectral")
```

Figure 5-63: Plotting a Tree Map with “geom_treemap()” Function

The “geom_treemap()” function from treemapify package is used to plot a tree map as shown in analysis 2-1. By mapping the required values in the aesthetics function, the treemap will be plotted successfully based on the mapping. Then, labels for each group in the treemap are added to aid in reading the graphs.

6.0 Conclusion

As a conclusion, the data problems identified has been answered after the data analytics lifecycle is carried out and completed. In data exploration phase, overview and distribution of data is visualized with graphs to understand the dataset so that the problems in dataset can be identified and handled. Afterwards, data cleaning handles problems that affect data quality such as missing values and outliers. After that, data transformation such as categorizing continuous variable, computing new variable with existing variables, and aggregation of data is executed to extract insight from more perspective of the dataset. In data visualization phase, the data problems identified is solved using techniques such as descriptive analysis through graphs and visuals such as lollipop chart, correlation matrix, and percent stacked bar chart to explore the dataset. Besides that, various additional features are implemented in the R program to solve the data problems elegantly.

References

Aellina. (17 June, 2019). *Commissions Brokers Make in Real Estate*. Retrieved from Gupta & Sen: https://www.guptasen.com/commissions-brokers-real-estate/

Berg, R. G. (n.d.). *Pearson Correlations – Quick Introduction*. Retrieved from SPSS Tutorials: https://www.spss-tutorials.com/pearson-correlation-coefficient/

Brintannica. (16 August, 2022). *Kolkata*. Retrieved from Britannica: https://www.britannica.com/place/Kolkata

Britannica. (7 Oct, 2022). *Chennai*. Retrieved from Britannica: https://www.britannica.com/place/Chennai

Carron, J. (13 December, 2021). *Violin Plots 101: Visualizing Distribution and Probability Density*. Retrieved from Mode: https://mode.com/blog/violin-plot-examples/

Corporate Finance Institute. (4 November, 2022). *What is a Scatter Plot?* Retrieved from Corporate Finance Institute: https://corporatefinanceinstitute.com/resources/data-science/scatter-plot/

Galarnyk, M. (9 August, 2022). *Understanding Boxplots*. Retrieved from builtin: https://builtin.com/data-science/boxplot

ggplot2. (n.d.). *Viridis colour scales from viridisLite*. Retrieved from ggplot2: https://ggplot2.tidyverse.org/reference/scale_viridis.html

Gien, S. (n.d.). *Pie Chart: Definition, Examples, Make one in Excel/SPSS*. Retrieved from StatisticsHowTo.com: Elementary Statistics for the rest of us!: https://www.statisticshowto.com/probability-and-statistics/descriptive-statistics/pie-chart/

ICICI Bank. (15 February, 2022). *Understanding the property area: Carpet, built-up and super built-up*. Retrieved from ICICI Bank: https://www.icicibank.com/knowledge-hub/blog/all-about-carpet-area-built-up-area-and-super-built-up-area.page

Key Inspection Services. (n.d.). *10 BENEFITS OF WORKING WITH A REAL ESTATE AGENT WHEN BUYING A HOME*. Retrieved from Key Inspection Services: https://www.keyinspectionservices.com/resources/10-benefits-of-working-with-a-realtor-when-buying-a-home/

Lubridate Tidyverse. (n.d.). *Round, floor and ceiling methods for date-time objects*. Retrieved from lubridate 1.9.0: https://lubridate.tidyverse.org/reference/round_date.html

Mishra, P. (21 July, 2022). *Difference between Carpet Area, Built up Area & Super Area*. Retrieved from magicbricks: https://www.magicbricks.com/blog/what-is-carpet-area/114783.html#Conclusion

Mishra, R. (26 April, 2022). *Why is Delhi University So Famous*. Retrieved from Duhook: https://duhook.com/why-is-delhi-university-so-famous/

Mishra, S. (1 November, 2022). *Carpet area, built up area and super built up area: Know the difference*. Retrieved from Housing.com: https://housing.com/news/real-estate-basics-part-1-carpet-area-built-up-area-super-built-up-area/

Mumbai Online Network. (n.d.). *About Mumbai*. Retrieved from mumbaionline.in: https://www.mumbaionline.in/city-guide/about-mumbai

RDocumentation. (27 November, 2022). *str_split_fixed: Split up a string into a fixed number of pieces.* Retrieved from RDocumentation: https://www.rdocumentation.org/packages/stringr/versions/0.6.2/topics/str_split_fixed

Sharma, P. G. (13 May, 2022). *15 different types of houses in India*. Retrieved from Housing.com: https://housing.com/news/types-of-houses-in-india/

Six Sigma Daily. (20 March, 2020). *Six Sigma Tools: What is a Histogram?* Retrieved from Six Sigma Daily: https://www.sixsigmadaily.com/six-sigma-tools-histogram/

Tableau. (18 November, 2022). *Guide To Data Cleaning: Definition, Benefits, Components, And How To Clean Your Data*. Retrieved from Tableau: https://www.tableau.com/learn/articles/what-is-data-cleaning

Turito. (9 April, 2022). *Line Plot: Definition, Types and Steps to Draw a Line Plot*. Retrieved from Turito: https://www.turito.com/learn/math/line-plot

West, R. M. (2022). Best practice in statistics: The use of log transformation. *Annals of Clinical Biochemistry*, 162-165.

Yi, M. (n.d.). *A Complete Guide to Bar Charts*. Retrieved from Chartio: https://chartio.com/learn/charts/bar-chart-complete-guide/