INDIVIDUAL ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT127-3-2-PFDA

Programming for Data Analysis

APU2F2209SE

Student name: Jackson Tai

Student ID: TP059628

Handout Date: 10th October 2022

Hand-in Date: 2nd December 2022

Lecturer: Miss Minnu Helen Joseph & Miss Farhana Illiani Binti Hassan

Table of Contents

1.0 - Introduction and assumption

1.1 - Introduction

This documentation aims to analyse how people choose their rental house based on their family background, locality, and lifestyle. This provides the Human Rights Measurement (HRM) team with meaningful insight for decision-making. The given dataset contains the details of varied houses, apartments or flats available for rent in India. In total, the dataset contains 12 columns and approximately 4700 rows of records. This dataset will be explored and analysed using an integrated development environment for the R programming language, called RStudio.

Structure of the dataset:

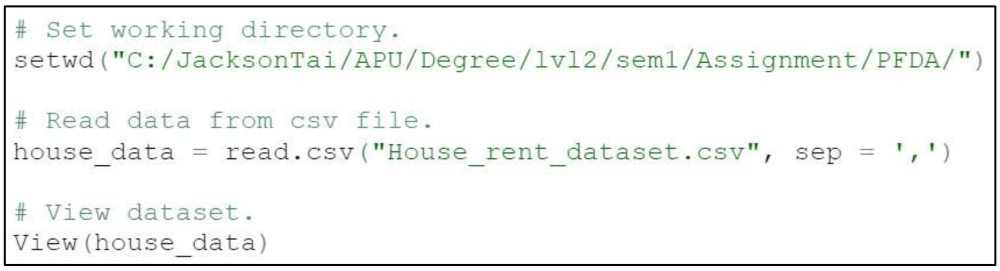
- **BHK**: Number of bedrooms in the houses/apartments/flats.
- **Rent**: Rent of the houses/apartments/flats.
- **Size**: Size of the houses/apartments/flats in square feet.
- **Floor**: Houses/apartments/flats situated on which floor and the total number of floors.
- **Area Type**: Size of the houses/apartments/flats calculated on.
- **Area Locality**: Locality of the houses/apartments/flats.
- **City**: City where the houses/apartments/flats are located.
- **Furnishing Status**: Furnishing status of the houses/apartments/flats.
- **Tenant Preferred**: The type of tenant preferred by the owner or agent.
- **Bathroom**: Number of bathrooms in the houses/apartments/flats.
- **Point of Contact**: The people who should be contacted for more information regarding the houses/apartments/flats.

1.2 - Assumption

The given dataset contains information on houses, apartments or flats available for rent. These data are obtained from the people who posted a list of houses they wish to be rented out. For this, the given dataset is useful for analysing house rent prediction. The properties posted in the given dataset have already been rented out. As such, this dataset can be used to analyse houses that are preferred by tenants.

2.0 - Data import, cleaning, pre-processing, and exploration

2.1 - Data Import

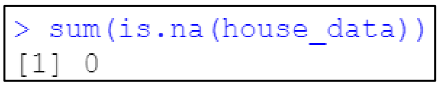


```
# Set working directory.
setwd("C:/JacksonTai/APU/Degree/lvl2/sem1/Assignment/PFDA/")

# Read data from csv file.
house_data = read.csv("House_rent_dataset.csv", sep = ',')

# View dataset.
View(house_data)
```

Figure 1: Data import source code

The first and yet most important step before the analysis process begins is to import data from the external files. Before importing the data, the working directory of the working environment is set using the `setwd()` built-in function. It takes a new working directory as the argument, `C:/JacksonTai/APU/Degree/lvl2/sem1/Assignment/PFDA/`. Once setting the directory, the rent data in a CSV file called `House_Rent_Dataset.csv` is imported by using the `read.csv()` built-in function. The first parameter of this function is the directory of the importing file while the second parameter is the separator `sep` with the assigned value of comma, which tells R that the file is comma-delimited. Lastly, the `View()` function is being used to view the dataset in a visual table form. It takes dataset `house_data` as the parameter and will automatically open another tab to display the dataset.

2.2 - Data Cleaning



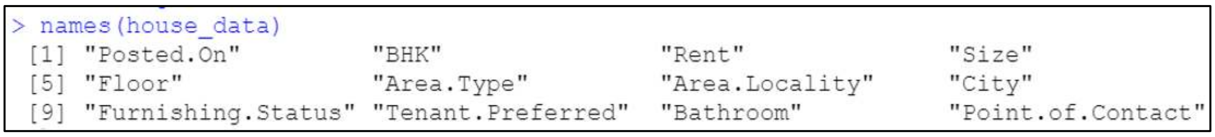
```
> sum(is.na(house_data))
[1] 0
```

Figure 2: Check for missing value

In R, missing values are represented by the symbol `NA` (not available). The `is.na()` function is used to check if `NA` values exist. The `sum()` is also being used to count all the possible `NA` in the dataset. For the given dataset, there is 0 number of `NA`, meaning there is no missing value found.

2.3 - Data Pre-processing

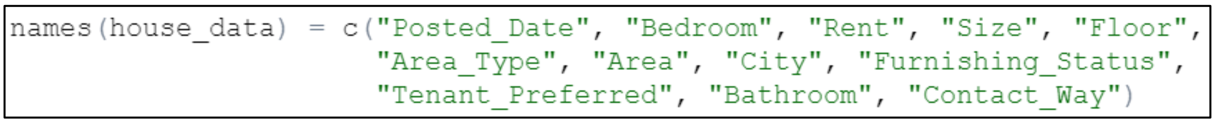
2.3.1 - View and rename column names



```
> names(house_data)
 [1] "Posted.On"         "BHK"              "Rent"             "Size"
 [5] "Floor"             "Area.Type"        "Area.Locality"    "City"
 [9] "Furnishing.Status" "Tenant.Preferred" "Bathroom"         "Point.of.Contact"
```

Figure 3: Output of dataset's heading name

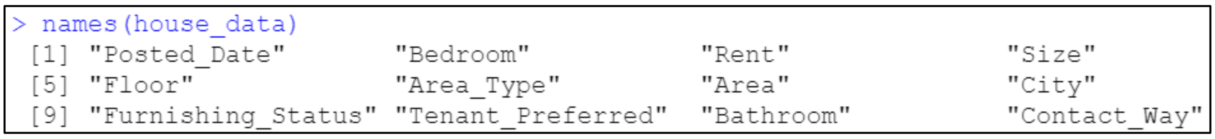
The `names()` function is used to return the names of a data object by passing the name of the variable that holds the data as the argument (Chidalu, n.d.). When the dataset is first imported in R, the column names of the `house_data` dataset are defaulted to be the same and the spaces is being replaced with a dot.



```
names(house_data) = c("Posted_Date", "Bedroom", "Rent", "Size", "Floor",
                      "Area_Type", "Area", "City", "Furnishing_Status",
                      "Tenant_Preferred", "Bathroom", "Contact_Way")
```

Figure 4: Rename column names

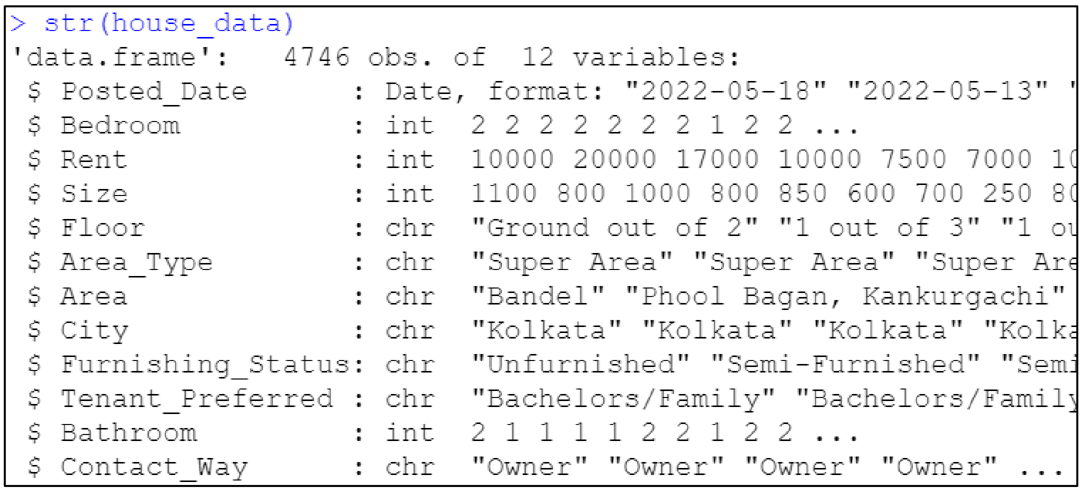
The column names of the given dataset are being renamed to be more meaningful for conducting the analysis. The `names()` function is also being used to rename the column names by assigning a vector of characters that contains the new column names.



```
> names(house_data)
 [1] "Posted_Date"       "Bedroom"          "Rent"             "Size"
 [5] "Floor"             "Area_Type"        "Area"             "City"
 [9] "Furnishing_Status" "Tenant_Preferred" "Bathroom"         "Contact_Way"
```

Figure 5: Output of renamed column names

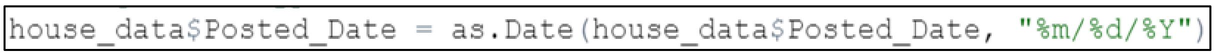
2.3.2 - View and change data types



```
> str(house_data)
'data.frame':   4746 obs. of  12 variables:
 $ Posted_Date      : Date, format: "2022-05-18" "2022-05-13" '
 $ Bedroom          : int  2 2 2 2 2 2 2 1 2 2 ...
 $ Rent             : int  10000 20000 17000 10000 7500 7000 10
 $ Size             : int  1100 800 1000 800 850 600 700 250 80
 $ Floor            : chr  "Ground out of 2" "1 out of 3" "1 ou
 $ Area_Type        : chr  "Super Area" "Super Area" "Super Are
 $ Area             : chr  "Bandel" "Phool Bagan, Kankurgachi"
 $ City             : chr  "Kolkata" "Kolkata" "Kolkata" "Kolka
 $ Furnishing_Status: chr  "Unfurnished" "Semi-Furnished" "Semi
 $ Tenant_Preferred : chr  "Bachelors/Family" "Bachelors/Family
 $ Bathroom         : int  2 1 1 1 1 2 2 1 2 2 ...
 $ Contact_Way      : chr  "Owner" "Owner" "Owner" "Owner" ...
```

Figure 6: Structure and information of each column in the dataset

The data types of each column in the dataset can be viewed using the `str()` function, which compactly displays the internal structure of an R object. It can display even the internal structure of large lists which are nested. It gives information about the columns and rows along with additional information like the column names, and the class of each column followed by a few of the initial observations of each of the columns.

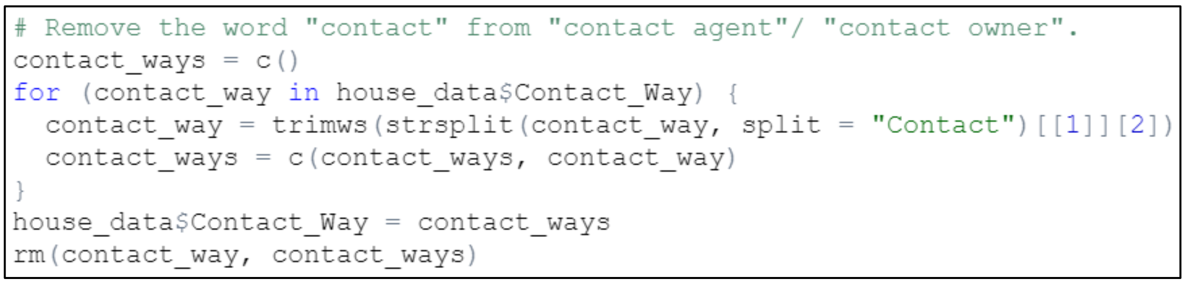


```
house_data$Posted_Date = as.Date(house_data$Posted_Date, "%m/%d/%Y")
```

Figure 7: Change data types

In Figure 6 above, the data type of the field `Posted.On` consists of dates with the data types of characters. This makes it difficult to perform analysis when there is a need to perform calculations on the date. Therefore, the data type of this column is being changed to date using the `as.Date()` functions. The first parameter consists of the column in which the data types will be changed, and the second parameter consists of the date format.

2.3.3 - Remove redundant words in the data



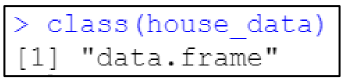
```r
# Remove the word "contact" from "contact agent"/ "contact owner".
contact_ways = c()
for (contact_way in house_data$Contact_Way) {
  contact_way = trimws(strsplit(contact_way, split = "Contact")[[1]][2])
  contact_ways = c(contact_ways, contact_way)
}
house_data$Contact_Way = contact_ways
rm(contact_way, contact_ways)
```

Figure 8: Remove redundant words in the data

The above source code aims to remove the redundant word “contact” from the data under the `Contact_Way` column, which was named in section 2.3.1. The `contact_ways` vector is being created to hold all extracted data such as “owner” and “agent”. A for loop is being used to iterate through each data in the `Contact_Way` column of `house_data` dataset. Within the for loop, the `strsplit()` function is used to remove the word `“Contact”` from the `contact_way` variable and return a list consisting of a vector that holds the remaining characters that are being split. The vector consists of an empty string and the extracted data (owner/agent). Thus, the `[[1]]` is specified to access the first element of the list, that is the vector. The `[2]` is then used to access the second element of the vector, which is the extracted data. Once extracted the data, it is being added to the `contact_ways` vector at the end of each iteration. Lastly, the `contact_ways` vector is updated to the dataset and the `rm()` function is used to remove the variable from the memory to avoid possible collapse of the same variable name later.

2.4 - Data Exploration

2.4.1 - Data types of the given dataset

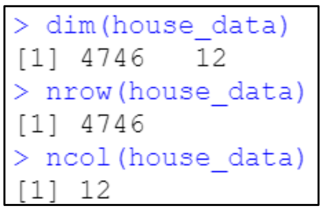


```
> class(house_data)
[1] "data.frame"
```

Figure 9: Data types of the given dataset

The `class()` function is being used to return the values of data types of the dataset, which is a data frame.

2.4.2 - Dimensions of the given dataset

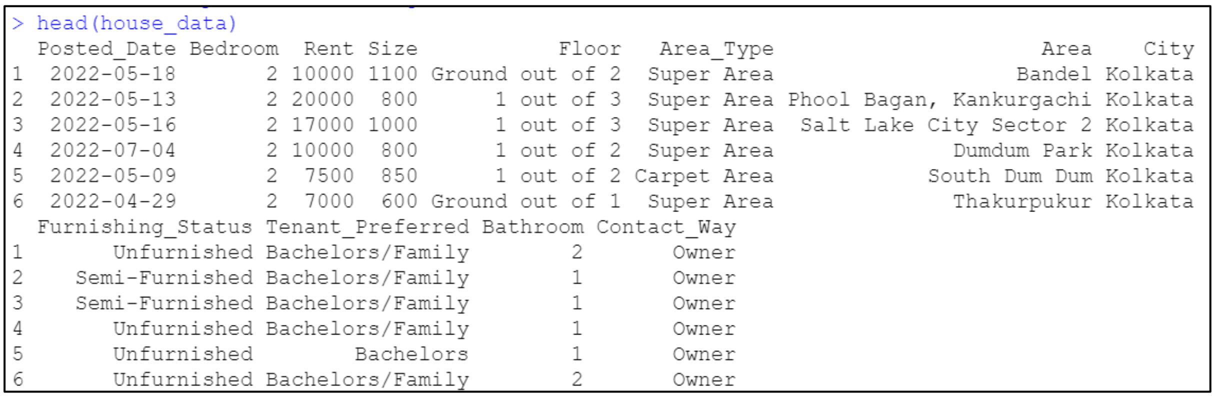


```
> dim(house_data)
[1] 4746   12
> nrow(house_data)
[1] 4746
> ncol(house_data)
[1] 12
```

Figure 10: Dimensions of the given dataset

The `dim()` function is used to get the dimension of the given dataset, which returns the number of rows and columns (Zach, 2022). Moreover, the `nrow()` and `ncol()` functions can also be used to get the row and column number of the dataset respectively. The given dataset has 4746 rows and 12 columns.

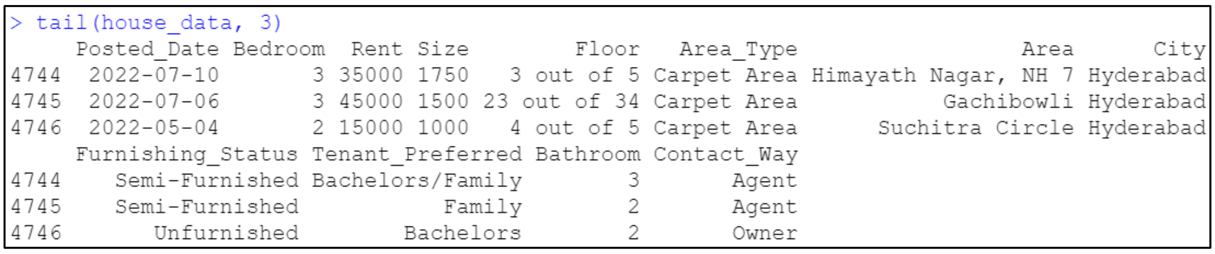
2.4.3 - Sample data of the given dataset



```
> head(house_data)
  Posted_Date Bedroom  Rent Size           Floor   Area_Type                      Area    City
1  2022-05-18       2 10000 1100 Ground out of 2  Super Area                    Bandel Kolkata
2  2022-05-13       2 20000  800      1 out of 3  Super Area Phool Bagan, Kankurgachi Kolkata
3  2022-05-16       2 17000 1000      1 out of 3  Super Area  Salt Lake City Sector 2 Kolkata
4  2022-07-04       2 10000  800      1 out of 2  Super Area              Dumdum Park Kolkata
5  2022-05-09       2  7500  850      1 out of 2 Carpet Area            South Dum Dum Kolkata
6  2022-04-29       2  7000  600 Ground out of 1  Super Area               Thakurpukur Kolkata
  Furnishing_Status Tenant_Preferred Bathroom Contact_Way
1       Unfurnished Bachelors/Family        2       Owner
2    Semi-Furnished Bachelors/Family        1       Owner
3    Semi-Furnished Bachelors/Family        1       Owner
4       Unfurnished Bachelors/Family        1       Owner
5       Unfurnished        Bachelors        1       Owner
6       Unfurnished Bachelors/Family        2       Owner
```

Figure 11: Sample data using head function

The sample data of the given dataset can be obtained by using the `head()` function. By default, the `head()` function returns the first 6 rows of the dataset if there is no value provided for the second parameter (CN, 2022).

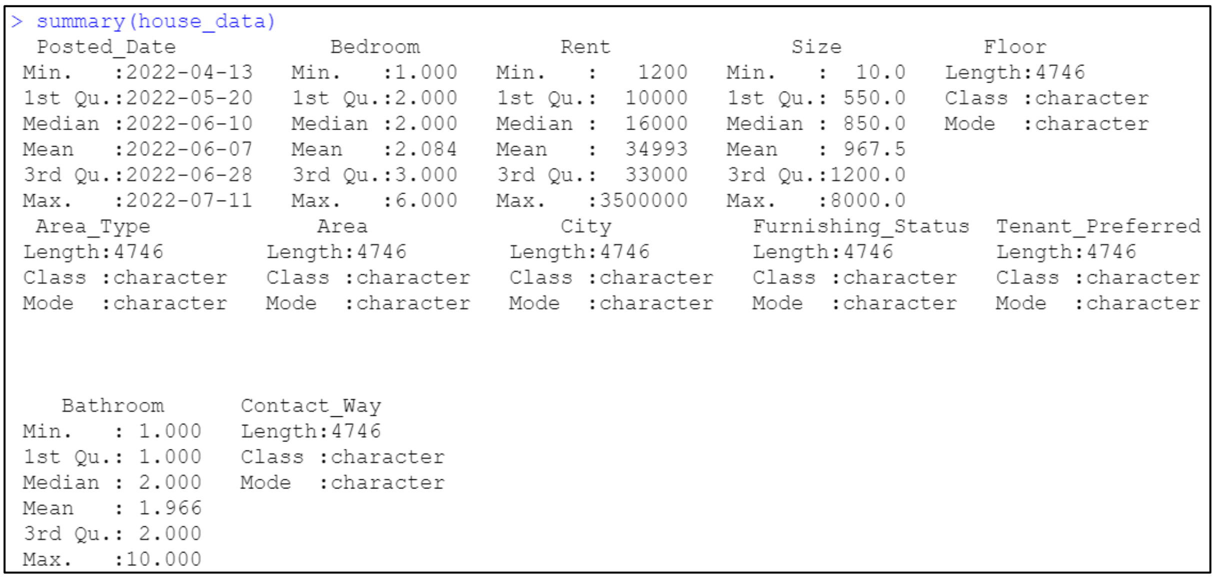


```
> tail(house_data, 3)
     Posted_Date Bedroom  Rent Size        Floor   Area_Type                 Area      City
4744  2022-07-10       3 35000 1750   3 out of 5 Carpet Area Himayath Nagar, NH 7 Hyderabad
4745  2022-07-06       3 45000 1500 23 out of 34 Carpet Area           Gachibowli Hyderabad
4746  2022-05-04       2 15000 1000   4 out of 5 Carpet Area      Suchitra Circle Hyderabad
     Furnishing_Status Tenant_Preferred Bathroom Contact_Way
4744     Semi-Furnished Bachelors/Family        3       Agent
4745     Semi-Furnished           Family        2       Agent
4746        Unfurnished        Bachelors        2       Owner
```

Figure 12: Sample data using tail function

On the other hand, the `tail()` function is being used to return the last n rows of the dataset and the second parameter is the number of rows to be returned.

2.4.4 - Summary of the given dataset



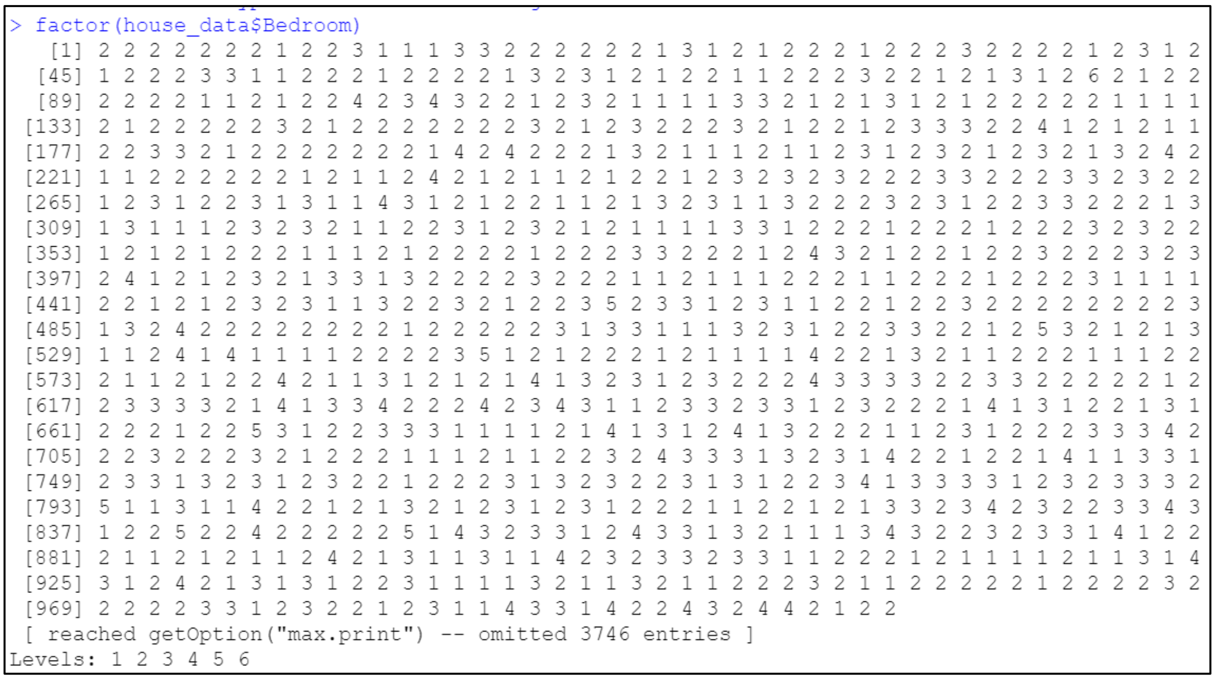
```
> summary(house_data)
  Posted_Date            Bedroom          Rent              Size            Floor          
 Min.   :2022-04-13   Min.   :1.000   Min.   :   1200   Min.   :  10.0   Length:4746       
 1st Qu.:2022-05-20   1st Qu.:2.000   1st Qu.:  10000   1st Qu.: 550.0   Class :character  
 Median :2022-06-10   Median :2.000   Median :  16000   Median : 850.0   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                     
  Area_Type             Area               City           Furnishing_Status  Tenant_Preferred  
 Length:4746        Length:4746        Length:4746        Length:4746        Length:4746       
 Class :character   Class :character   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character  



    Bathroom      Contact_Way       
 Min.   : 1.000   Length:4746       
 1st Qu.: 1.000   Class :character  
 Median : 2.000   Mode  :character  
 Mean   : 1.966                     
 3rd Qu.: 2.000                     
 Max.   :10.000                     
```

Figure 13: Summary of the given dataset

The `summary()` function is being used to quickly summarizes the values in the data frame, and provide information such as the minimum, 1st quartile, median, mean 3rd quartile, and max value of the numeric values. It provides information such as the length, class, and mode for categorical values.

2.4.5 - Categorical data values in the given dataset

In section 2.4.4, the `summary()` function does not provide what kind of values are there in the categorical data. For this, the `factor()` function is used to display all possible values.



```
> factor(house_data$Bedroom)
   [1] 2 2 2 2 2 2 2 1 2 2 3 1 1 1 3 3 2 2 2 2 2 2 1 3 1 2 1 2 2 2 1 2 2 2 3 2 2 2 2 1 2 3 1 2
  [45] 1 2 2 2 3 3 1 1 2 2 2 1 2 2 2 2 1 3 2 3 1 2 1 2 2 1 1 2 2 2 3 2 2 1 2 1 3 1 2 6 2 1 2 2
  [89] 2 2 2 2 1 1 2 1 2 2 4 2 3 4 3 2 2 1 2 3 2 1 1 1 1 3 3 2 1 2 1 3 1 2 1 2 2 2 2 2 1 1 1 1
 [133] 2 1 2 2 2 2 2 3 2 1 2 2 2 2 2 2 2 3 2 1 2 3 2 2 2 3 2 1 2 2 1 2 3 3 3 2 2 4 1 2 1 2 1 1
 [177] 2 2 3 3 2 1 2 2 2 2 2 2 2 1 4 2 4 2 2 2 1 3 2 1 1 1 2 1 1 2 3 1 2 3 2 1 2 3 2 1 3 2 4 2
 [221] 1 1 2 2 2 2 2 2 1 2 1 1 2 4 2 1 2 1 1 2 1 2 2 1 2 3 2 3 2 3 2 2 2 3 3 2 2 2 3 3 2 3 2 2
 [265] 1 2 3 1 2 2 3 1 3 1 1 4 3 1 2 1 2 2 1 1 2 1 3 2 3 1 1 3 2 2 2 3 2 3 1 2 2 3 3 2 2 2 1 3
 [309] 1 3 1 1 1 2 3 2 3 2 1 1 2 2 3 1 2 3 2 1 2 1 1 1 1 3 3 1 2 2 2 1 2 2 2 1 2 2 2 3 2 3 2 2
 [353] 1 2 1 2 1 2 2 2 1 1 1 2 1 2 2 2 2 1 2 2 2 3 3 2 2 2 1 2 4 3 2 1 2 2 1 2 2 3 2 2 2 3 2 3
 [397] 2 4 1 2 1 2 3 2 1 3 3 1 3 2 2 2 2 3 2 2 2 1 1 2 1 1 1 2 2 2 1 1 2 2 2 1 2 2 2 3 1 1 1 1
 [441] 2 2 1 2 1 2 3 2 3 1 1 3 2 2 3 2 1 2 2 3 5 2 3 3 1 2 3 1 1 2 2 1 2 2 3 2 2 2 2 2 2 2 2 3
 [485] 1 3 2 4 2 2 2 2 2 2 2 2 1 2 2 2 2 2 3 1 3 3 1 1 1 3 2 3 1 2 2 3 3 2 2 1 2 5 3 2 1 2 1 3
 [529] 1 1 2 4 1 4 1 1 1 1 2 2 2 2 3 5 1 2 1 2 2 2 1 2 1 1 1 1 4 2 2 1 3 2 1 1 2 2 2 1 1 1 2 2
 [573] 2 1 1 2 1 2 2 4 2 1 1 3 1 2 1 2 1 4 1 3 2 3 1 2 3 2 2 2 4 3 3 3 3 2 2 3 3 2 2 2 2 2 1 2
 [617] 2 3 3 3 3 2 1 4 1 3 3 4 2 2 2 4 2 3 4 3 1 1 2 3 3 2 3 3 1 2 3 2 2 2 1 4 1 3 1 2 2 1 3 1
 [661] 2 2 2 1 2 2 5 3 1 2 2 3 3 3 1 1 1 1 2 1 4 1 3 1 2 4 1 3 2 2 2 1 1 2 3 1 2 2 2 3 3 3 4 2
 [705] 2 2 3 2 2 2 3 2 1 2 2 2 1 1 1 2 1 1 2 2 3 2 4 3 3 3 1 3 2 3 1 4 2 2 1 2 2 1 4 1 1 3 3 1
 [749] 2 3 3 1 3 2 3 1 2 3 2 2 1 2 2 2 3 1 3 2 3 2 2 3 1 3 1 2 2 3 4 1 3 3 3 3 1 2 3 2 3 3 3 2
 [793] 5 1 1 3 1 1 4 2 2 1 2 1 3 2 1 2 3 1 2 3 1 2 2 2 1 1 2 2 1 2 1 3 3 2 3 4 2 3 2 2 3 3 4 3
 [837] 1 2 2 5 2 2 4 2 2 2 2 2 5 1 4 3 2 3 3 1 2 4 3 3 1 3 2 1 1 1 3 4 3 2 2 3 2 3 3 1 4 1 2 2
 [881] 2 1 1 2 1 2 1 1 2 4 2 1 3 1 1 3 1 1 4 2 3 2 3 3 2 3 3 1 1 2 2 2 1 2 1 1 1 1 2 1 1 3 1 4
 [925] 3 1 2 4 2 1 3 1 3 1 2 2 3 1 1 1 1 3 2 1 1 3 2 1 1 2 2 2 3 2 1 1 2 2 2 2 2 1 2 2 2 2 3 2
 [969] 2 2 2 2 3 3 1 2 3 2 2 1 2 3 1 1 4 3 3 1 4 2 2 4 3 2 4 4 2 1 2 2
 [ reached getOption("max.print") -- omitted 3746 entries ]
Levels: 1 2 3 4 5 6
```

Figure 14: Factor of bedroom column

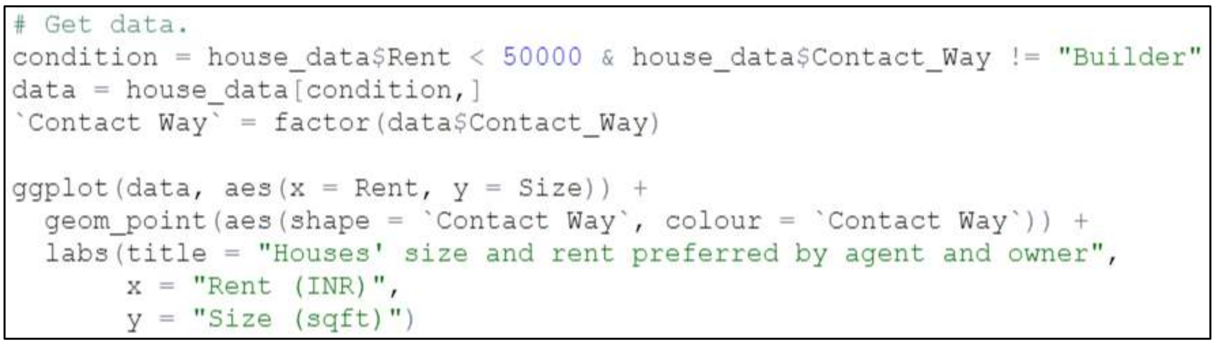
The `factor()` function takes a vector as the input, that is the `bedroom` column from the `house_data` dataset (Johnson, 2022). In figure 14, the data of the `bedroom` column is made up of numbers from 1 to 6.

3.0 - Question and Analysis

3.1 - Question 1: What kind of houses do people prefer to rent out?

Based on the given dataset, there are various types of houses being posted for people to rent. The HRM team must identify what kind of houses is being rented out by people. The following analysis provides adequate information on the attributes of houses being posted.

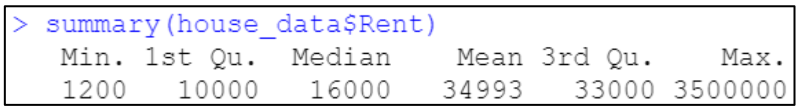
3.1.1 - Analysis 1.1: Find the house size and rent preferred by people.



```r
# Get data.
condition = house_data$Rent < 50000 & house_data$Contact_Way != "Builder"
data = house_data[condition,]
`Contact Way` = factor(data$Contact_Way)

ggplot(data, aes(x = Rent, y = Size)) +
  geom_point(aes(shape = `Contact Way`, colour = `Contact Way`)) +
  labs(title = "Houses' size and rent preferred by agent and owner",
       x = "Rent (INR)",
       y = "Size (sqft)")
```

Figure 15: Source code for analysis 1.1

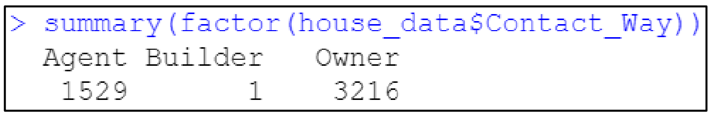
The above source code is to generate a scatter plot graph that is being used to visualize the relationship between the house size and rent with the contact way. The `data` dataset is derived from the `house_data` dataset with the condition of having a maximum rent of 100,000 INR and excluding the contact way of the builder. The reason for setting a maximum rent is to lower the high difference in rent, which ranges from 1,200 INR up to 3,500,000 INR.



```
> summary(house_data$Rent)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   1200   10000   16000   34993   33000 3500000 
```

Figure 16: Summary of house rent

The house with the contact way of the builder is being excluded from the dataset in this analysis as there is only one of it in the whole dataset (`house_data`), which is negligible to be excluded for the analysis.



```
> summary(factor(house_data$Contact_Way))
  Agent Builder   Owner 
   1529       1    3216 
```

Figure 17: Summary of houses' contact way

In addition to the dataset, the categories of contact way are also being extracted using the `factor()` built-in function, which takes a vector as the input, that is the contact way from the derived `data` dataset (Johnson, 2022). After obtaining the dataset to be plotted. The pre-written function `ggplot()` from the ggplot2 package is being used to plot the graph. The three key components of the grammar of graphics plots are:

- **Data:** the observations in the dataset
- **Aesthetics:** mappings from the data to visual properties
- **Geoms:** geometric objects that represent what is in the plot

The first parameter of the ggplot function would be the dataset to be plotted. The second parameter consists of the `aes()`, which includes the x-axis (Rent) and y-axis (Size) of the plotting graph. The `geom_point()` is also being used to create the scatterplots, which help display the relationship between two continuous variables. It takes `aes()` as the parameter, which specifies the shape and colour of the scatterplots, that is the contact way. Lastly, the `labs()` function is being used to assign the graph title as well as the name of x-axis and y-axis (Madhugiri, 2022).

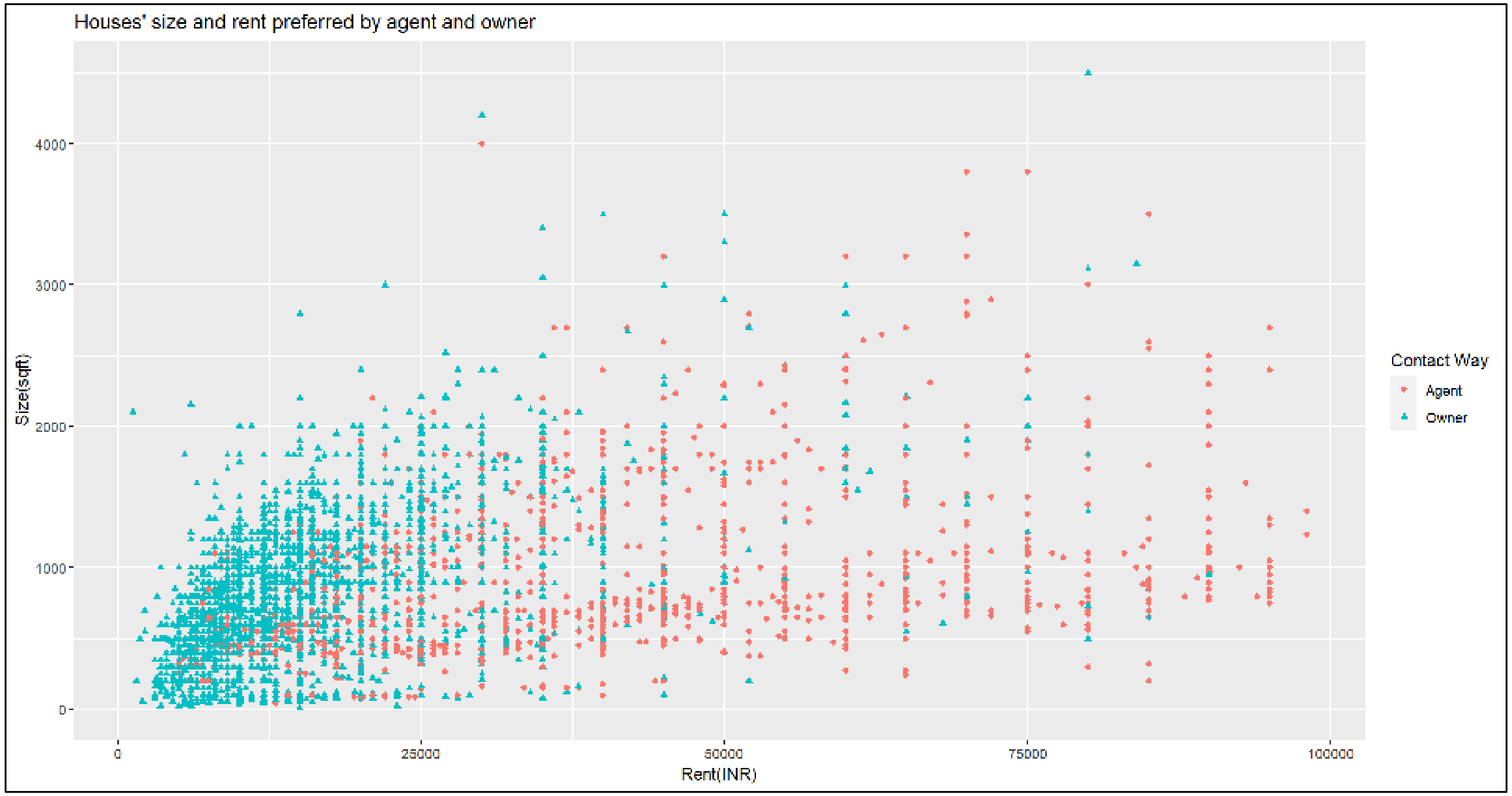
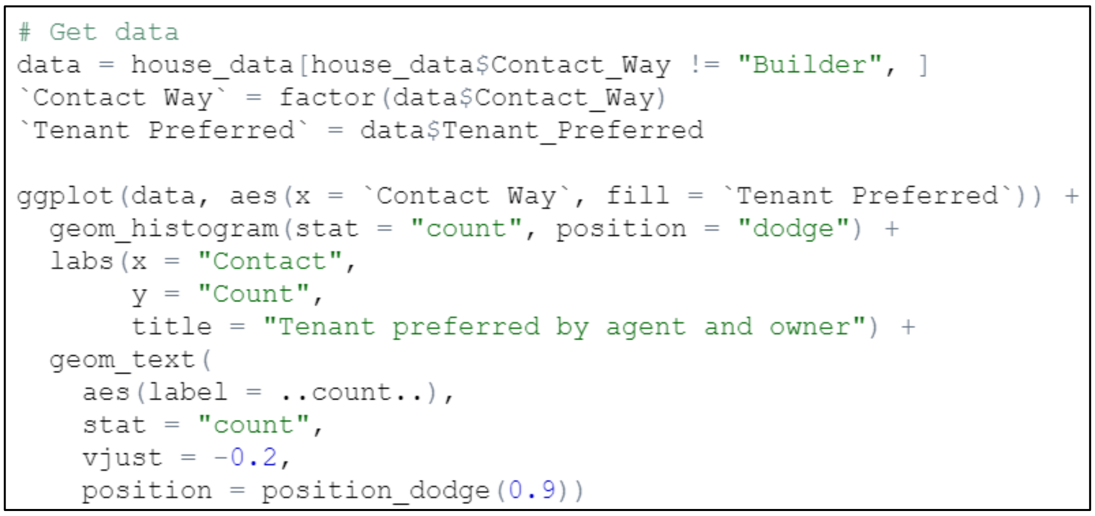


Figure 18: Analysis result 1.1

From the above graph, it can be observed that the owner prefers to rent out houses with lower rent compared to the agent. The majority of houses rent for the owner falls within the range of 25,000 INR whereas the agent has a wider range of rent. The vertical axis shows that agents prefer to rent out houses with larger sizes as opposed to the owner, where most of the house size starts around 500 square foot (sqft). These two metrics show the sign of agents tend to rent out luxury houses with higher rent since they will get higher commissions. In contrast, houses that are rented out by the owner are mostly common houses with sizes below 2,000 sqft.

3.1.2 - Analysis 1.2: Find the tenant preferred of houses



```r
# Get data
data = house_data[house_data$Contact_Way != "Builder", ]
`Contact Way` = factor(data$Contact_Way)
`Tenant Preferred` = data$Tenant_Preferred

ggplot(data, aes(x = `Contact Way`, fill = `Tenant Preferred`)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "Contact",
       y = "Count",
       title = "Tenant preferred by agent and owner") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9))
```

Figure 19: Source code for analysis 1.2

The above source code is to generate a histogram that is being used to visualize the relationship between the tenant preferred with the contact way. The `data` dataset is derived from the `house-data` dataset with the condition of excluding houses with contact way of the builder. Kindly refer to the reason for doing so under this section. The `fill` is used to display the graph grouped by tenants preferred so that it is more understandable. Moreover, the `geom_histogram()` is being used to generate the histogram, where the `stat` with the value of "count" counts the rows of each x-axis value and the *position* with the value of `"dodge"` is used to separate the bars plotted on the grid. The `labs()` function is used to name the histogram title and the axis on the graph. Lastly, the `geom_text()` function is used to display the count of every plot on top of the bar.

3.1.2 - Analysis

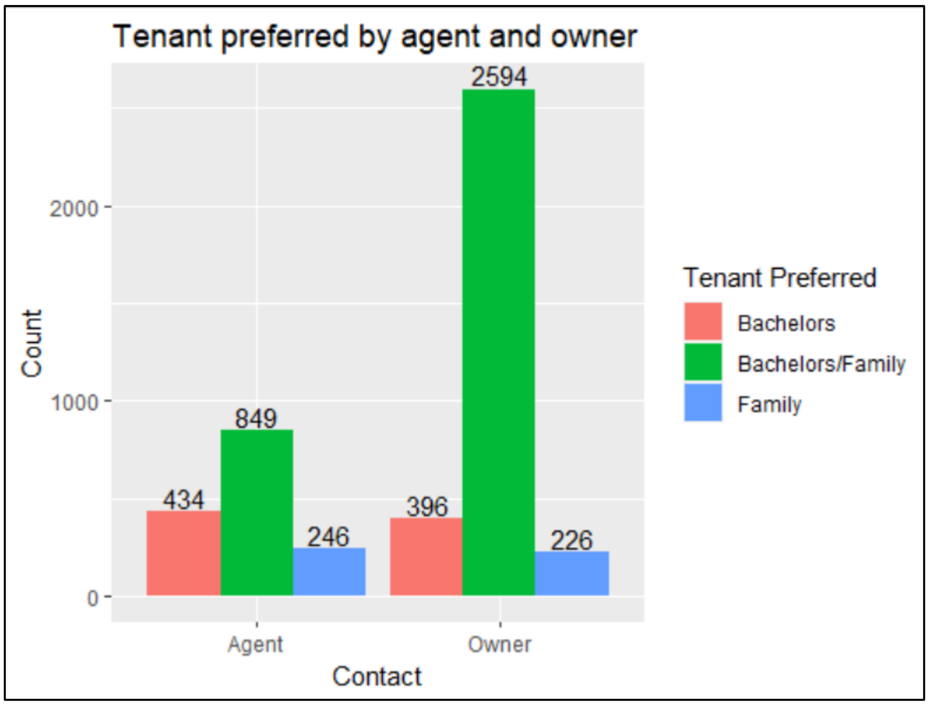
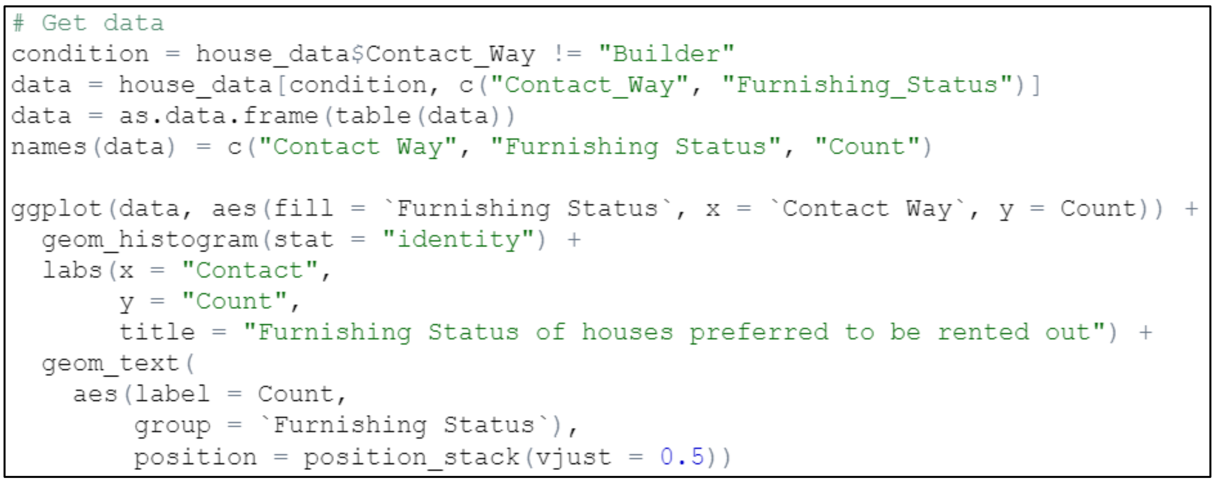


Figure 20: Analysis result 1.2

The result for analysis 1.2 shows that most agents and owners prefer renting out houses that are for tenants under the group of bachelor or family, with 849 and 2,594 of them respectively. There are 434 agents, and 396 owners prefer renting out houses for only the tenant that is a bachelor. The least preferred houses to be rented out by both agents and owners are the houses for tenants under the group of family, with only a total of 472. The reason for family people being the least preferred tenants could also be attributed to the size of houses they are renting out, where most of the houses posted are under 2,000 sqft as shown in analysis result 1.1.

3.1.3 - Analysis 1.3: Find the furnishing status of houses preferred by people



```r
# Get data
condition = house_data$Contact_Way != "Builder"
data = house_data[condition, c("Contact_Way", "Furnishing_Status")]
data = as.data.frame(table(data))
names(data) = c("Contact Way", "Furnishing Status", "Count")

ggplot(data, aes(fill = `Furnishing Status`, x = `Contact Way`, y = Count)) +
  geom_histogram(stat = "identity") +
  labs(x = "Contact",
       y = "Count",
       title = "Furnishing Status of houses preferred to be rented out") +
  geom_text(
    aes(label = Count,
        group = `Furnishing Status`),
        position = position_stack(vjust = 0.5))
```

Figure 21: Source code for analysis 1.3

The above source code generates a stacked bar chart to visualize the relationship between the furnishing status with the contact way. The `data` dataset derives only the records of contact way and furnishing status from the `house_data` dataset with the condition of excluding houses with contact way of the builder. Kindly refer to the reason for excluding the builder under this section. The `data` is then converted into a table using `table()` built-in function, which performs a tabulation of the furnishing status preferred by the owner or the agent. The table of data is then converted back into a data frame for renaming and plotting. The `names()` built-in function is used to rename the column heading name. The `stat` argument with the value of "`identity`" will take the y-axis value for the dependent variable. Lastly, the `geom_text()` built-in function is used to display the count of every plot on top of the bar.

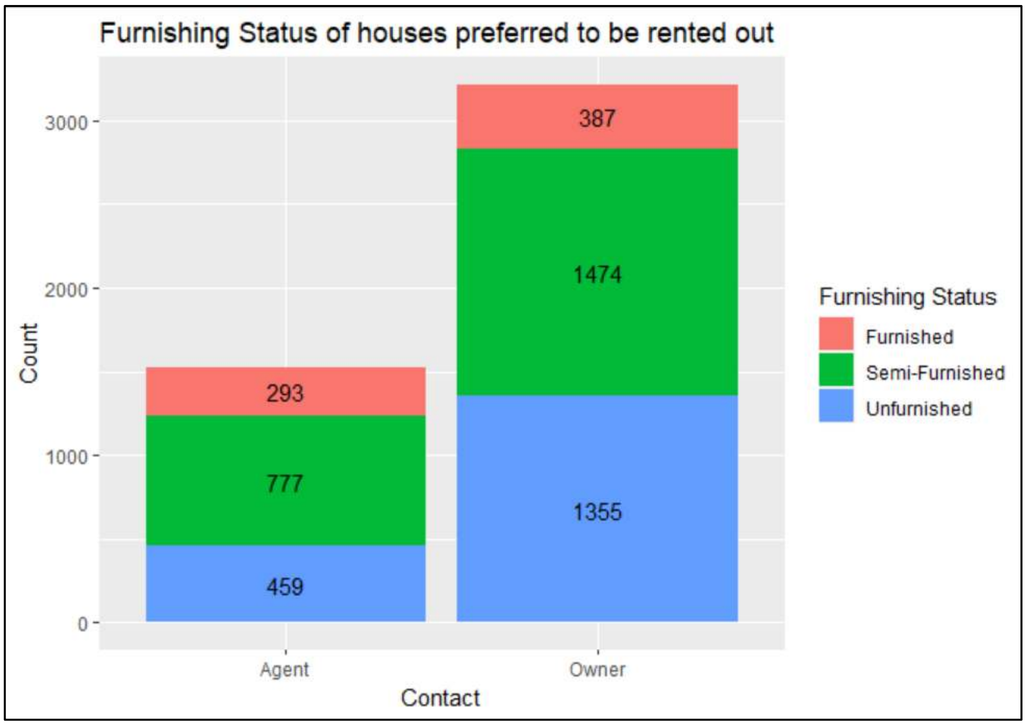
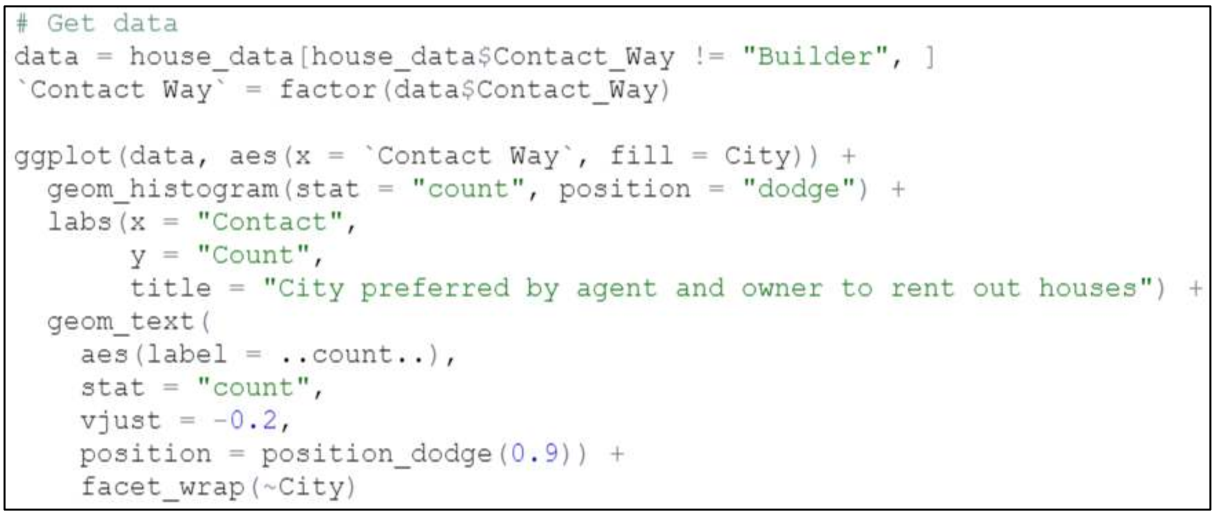


Figure 22: Analysis result 1.3

The result for analysis 1.3 shows that most agents and owners prefer to rent out houses that are semi-furnished, with 777 and 1,474 of them respectively. There are a total of 459 agents, and 1,355 owners prefer to rent out unfurnished houses. The least preferred furnishing status of houses to be rented out by both agent and owner is furnished, with only a total of 680 of them together. This could mean they are not willing to pay the upfront cost of furnishing the houses for the tenant as their goal is to maximize the return on investment (ROI).

3.1.4 - Analysis 1.4: Find the city in which the houses preferred by people located at



```
# Get data
data = house_data[house_data$Contact_Way != "Builder", ]
`Contact Way` = factor(data$Contact_Way)

ggplot(data, aes(x = `Contact Way`, fill = City)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "Contact",
       y = "Count",
       title = "City preferred by agent and owner to rent out houses") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9)) +
    facet_wrap(~City)
```

Figure 23: Source code for analysis 1.4

The above source code generates a histogram to visualize the relationship between the city with the contact way. The `data` dataset is derived from the `house_data` dataset with the condition of excluding houses with contact way of the builder. Kindly refer to the reason for doing so under this section. The `fill` argument is used to display the graph grouped by contact way. Moreover, the `geom_histogram()` is being used to generate the histogram. The `labs()` function is used to name the histogram title and the axis on the graph. The `geom_text()` function is also used to display the count of every plot on top of the bar. Lastly, the `facet_wrap()` function is used to split the single histogram into a multi-panel histogram that is related to each other (Haaf, 2019).

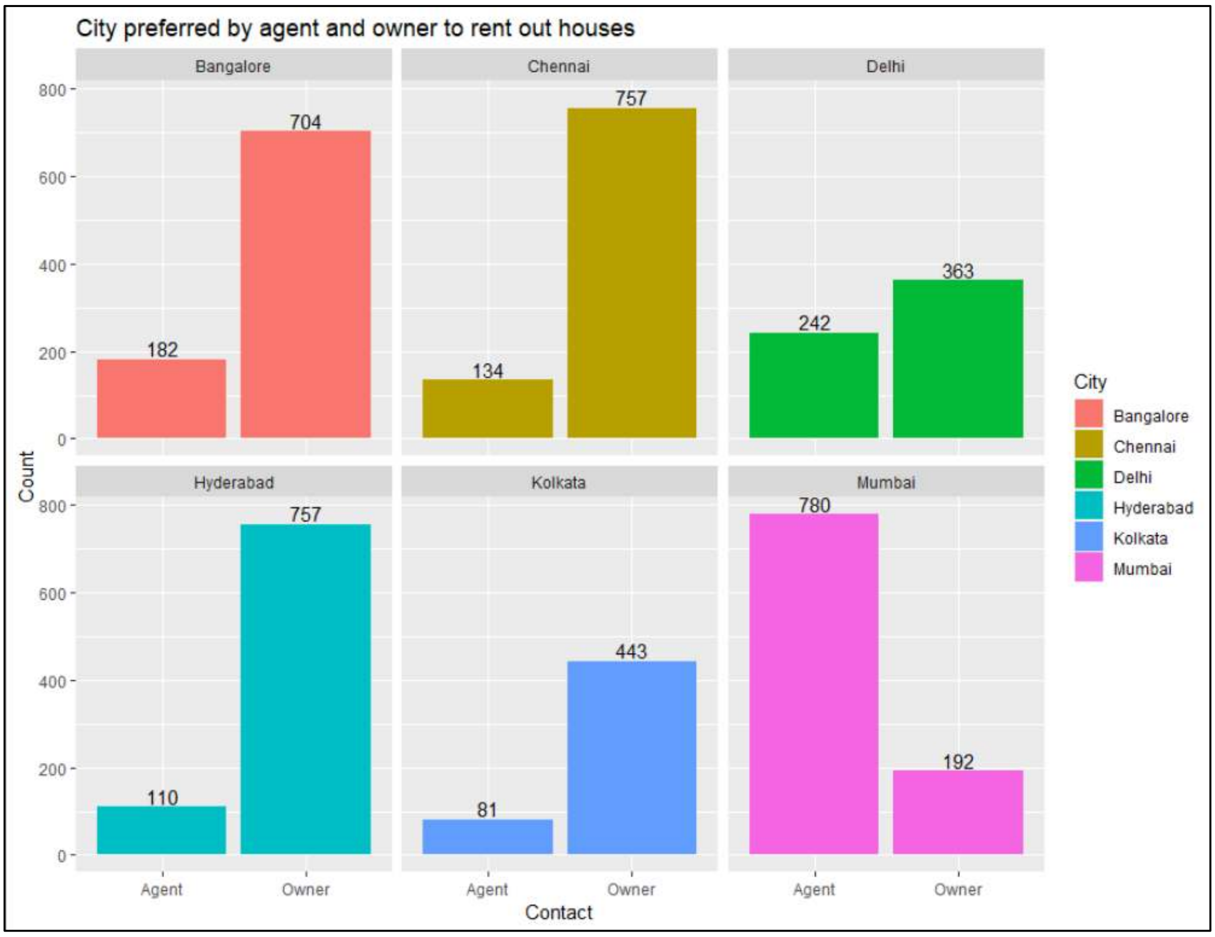
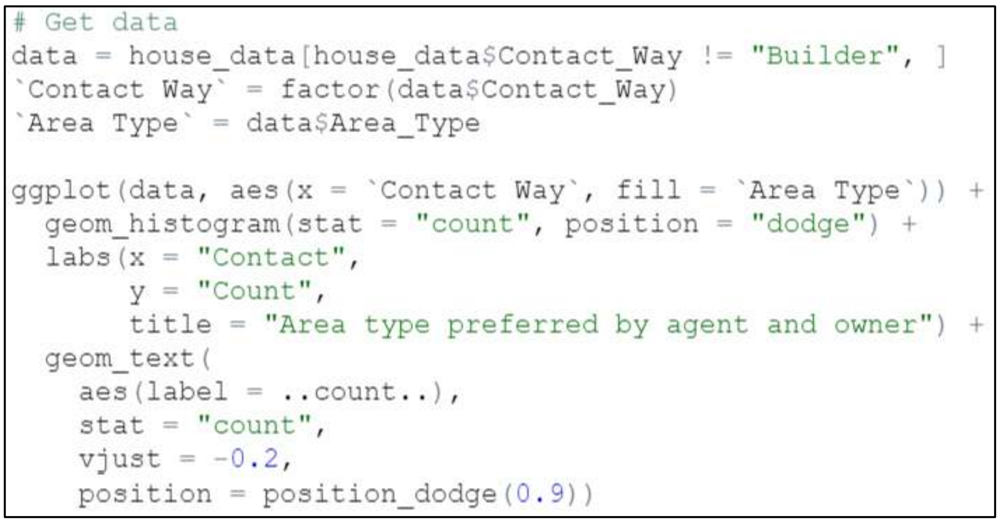


Figure 24: Analysis result 1.4

The most obvious result that can be observed through the histogram above is that most agent prefers to rent out houses that are located in Mumbai, with a total of 780 of them. As for owners, most of them prefer to rent out houses located in Chennai and Hyderabad. On the other hand, most agent does not prefer renting out houses in Kolkata, with only 81 agents renting out houses in this city. The owners do not prefer renting out houses in Mumbai, with only 192 of them. Overall, most people prefer to rent out houses in Mumbai, with a total of 972 of them. In contrast, Kolkata is the least preferred city by people to rent out houses, where only a total of 523 of them choose to rent out houses in this city.

3.1.5 - Analysis 1.5: Find the area type of houses preferred by people



```
# Get data
data = house_data[house_data$Contact_Way != "Builder", ]
`Contact Way` = factor(data$Contact_Way)
`Area Type` = data$Area_Type

ggplot(data, aes(x = `Contact Way`, fill = `Area Type`)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "Contact",
       y = "Count",
       title = "Area type preferred by agent and owner") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9))
```

Figure 25: Source code for analysis 1.5

The above source code generates a histogram to visualize the relationship between the area type with the contact way. Kindly refer to the source code explanation of analysis 1.4 for the details of obtaining data and plotting histogram.

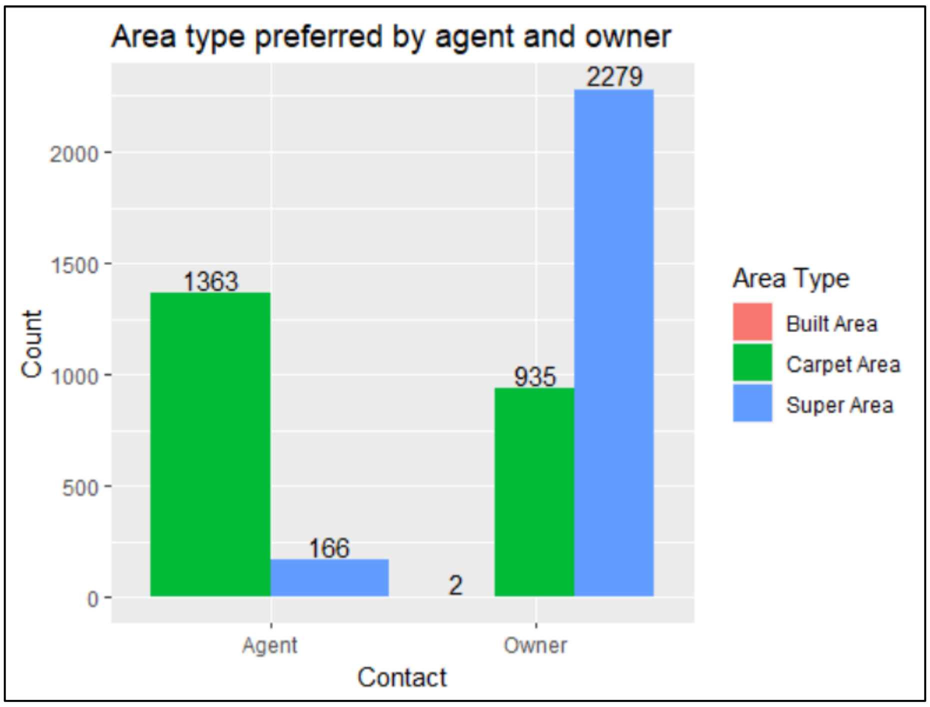
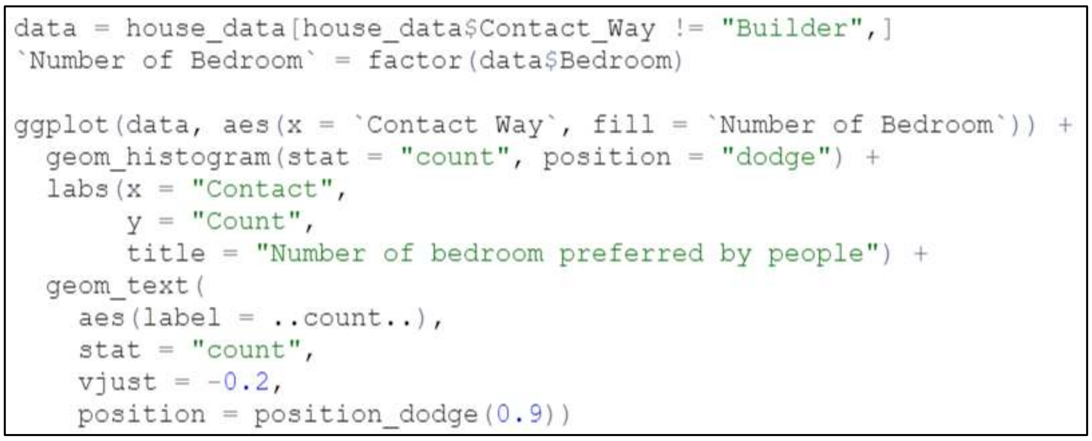


Figure 26: Analysis result 1.5

The above histogram shows that most agents prefer to rent out houses in which the size is calculated based on the carpet area and only a minority of them prefer super area. As for the owner, most of them prefer houses in which the size is calculated based on super area. Some owners prefer carpet areas and only two owners prefer built areas.

3.1.6 - Analysis 1.6: Find the number of bedrooms in houses preferred by people



```
data = house_data[house_data$Contact_Way != "Builder",]
`Number of Bedroom` = factor(data$Bedroom)

ggplot(data, aes(x = `Contact Way`, fill = `Number of Bedroom`)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "Contact",
       y = "Count",
       title = "Number of bedroom preferred by people") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9))
```

Figure 27: Source code for analysis 1.6

The above source code generates a histogram to visualize the relationship between the number of bedrooms with the contact way. Kindly refer to the source code explanation of analysis 1.4 for the details of obtaining data and plotting histogram.

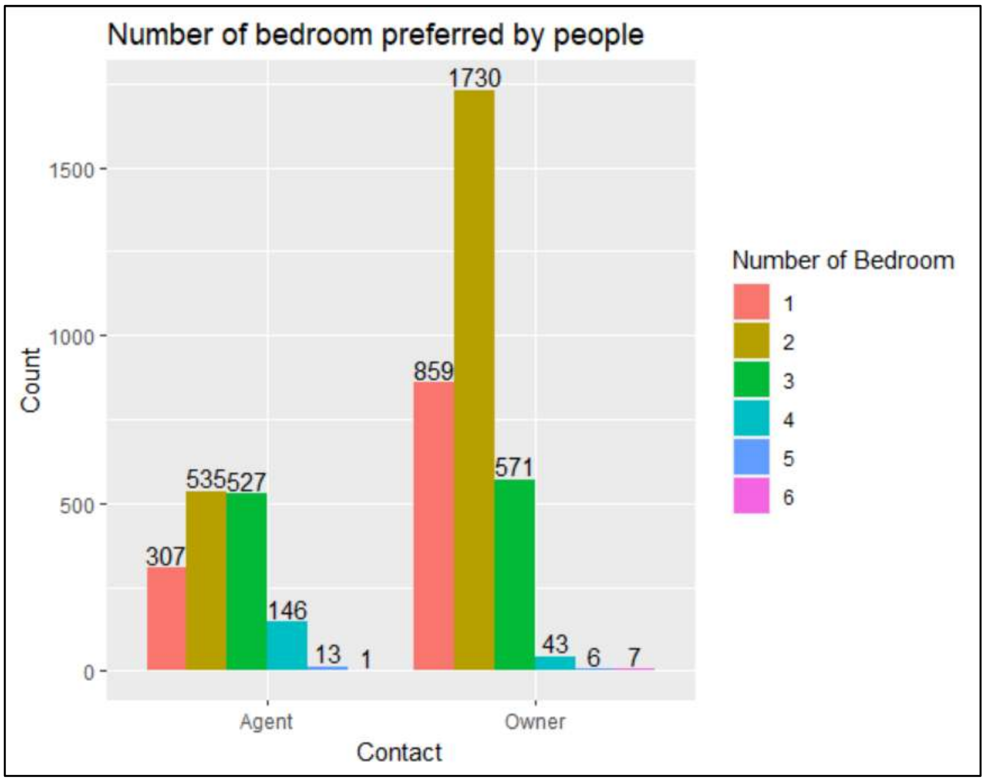
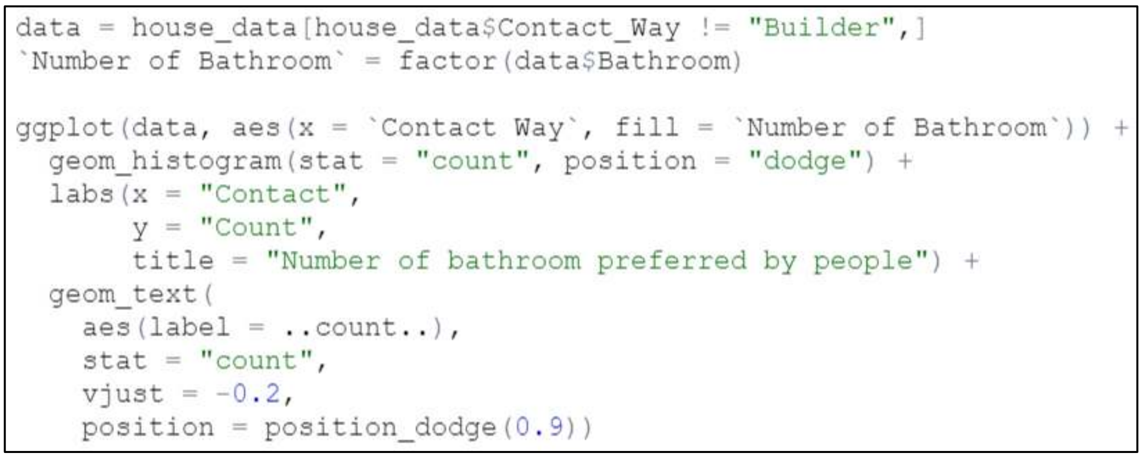


Figure 28: Analysis result 1.6

The above histogram shows that most of the agents and owners prefer to rent out houses with 2 bedrooms. More agents prefer to rent out houses with 3 bedrooms than one. Conversely, more owners prefer to rent out houses with one bedroom than 3.

3.1.7 - Analysis 1.7: Find the number of bathrooms in houses preferred by people



```
data = house_data[house_data$Contact_Way != "Builder",]
`Number of Bathroom` = factor(data$Bathroom)

ggplot(data, aes(x = `Contact Way`, fill = `Number of Bathroom`)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "Contact",
       y = "Count",
       title = "Number of bathroom preferred by people") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9))
```

Figure 29: Source code for analysis 1.7

The above source code generates a histogram to visualize the relationship between the number of bathrooms with the contact way. Kindly refer to the source code explanation of analysis 1.4 for the details of obtaining data and plotting histogram.

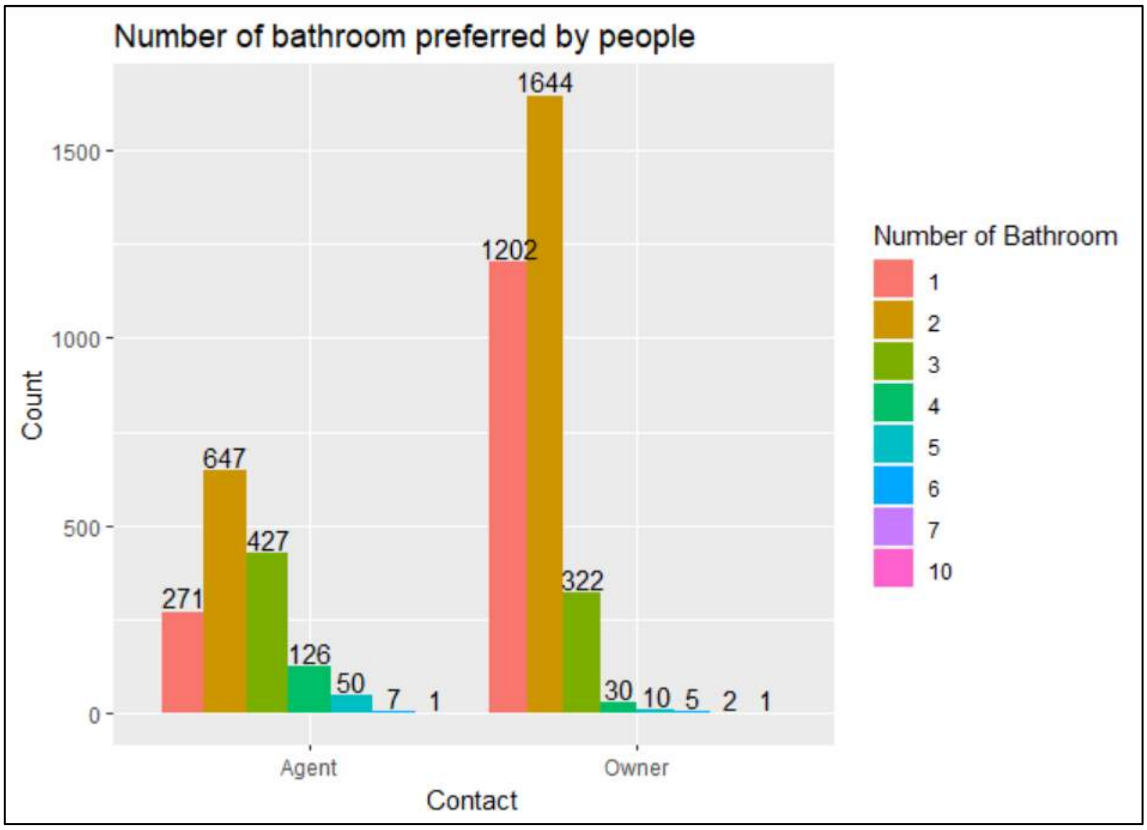


Figure 30: Analysis result 1.7

The above histogram shows that most of the agents and owners prefer to rent out houses with 2 bathrooms. More agents prefer to rent out houses with 3 bathrooms than one. Conversely, more owners prefer to rent out houses with one bathroom than 3.

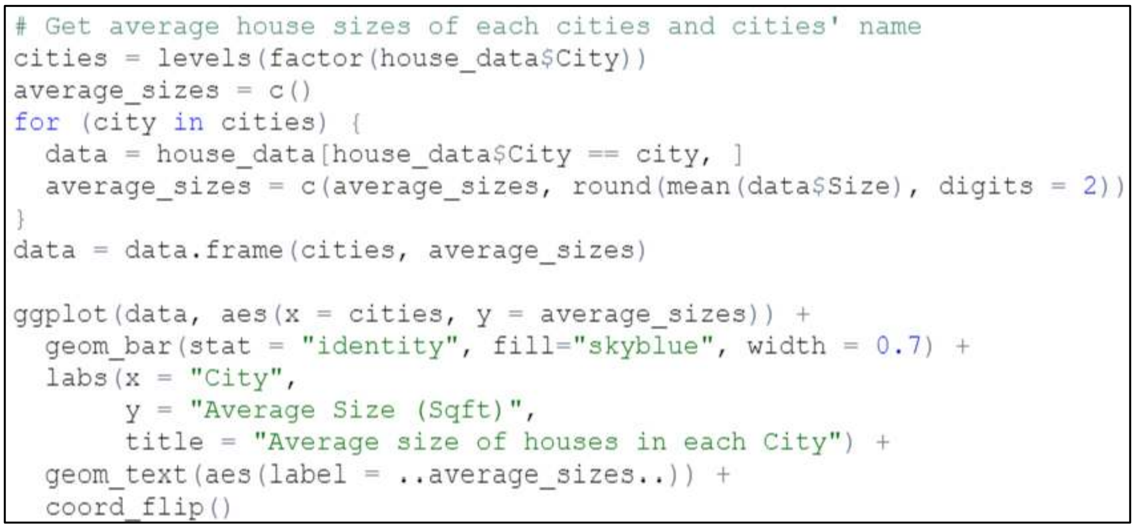
3.1.8 - Conclusion for question 1

There are a total of 7 analyses being done for question 1 to find out the houses that are preferred to be rented out by people. The interesting thing found in analysis 1.1 is most agents would prefer to rent out houses with higher prices as they are aiming for higher commissions. People would mostly prefer to rent out houses that are around 2,000 sqft and most of them prefer houses for either family or bachelors. They also prefer semi-furnished houses located in Mumbai. Apart from that, most people would prefer to rent out houses with 2 bedrooms and 2 bathrooms. Lastly, analysis 1.5 has found that agents would prefer to rent out houses in which the size is calculated based on carpet area whereas the owner would prefer super area.

3.2 - Question 2: What kind of houses are there in each city?

Analysis 1.4 shows that Mumbai is the most preferred city by the agent to rent out houses and it is also the least preferred city by the owner. On the other hand, the most preferred cities by the owner are Chennai and Hyderabad while the least preferred city by the agent is Kolkata. Thus, this question aims to find out what kind of houses there are in each city.

3.2.1 - Analysis 2.1: Find the average house size of each city



```
# Get average house sizes of each cities and cities' name
cities = levels(factor(house_data$City))
average_sizes = c()
for (city in cities) {
  data = house_data[house_data$City == city, ]
  average_sizes = c(average_sizes, round(mean(data$Size), digits = 2))
}
data = data.frame(cities, average_sizes)

ggplot(data, aes(x = cities, y = average_sizes)) +
  geom_bar(stat = "identity", fill="skyblue", width = 0.7) +
  labs(x = "City",
       y = "Average Size (Sqft)",
       title = "Average size of houses in each City") +
  geom_text(aes(label = ..average_sizes..)) +
  coord_flip()
```

Figure 31: Source code for analysis 2.1

The above source code is to generate a bar chart that is being used to visualize the average size of houses in each city. The `levels()` built-in function is being used to access the levels attributes of the city. A for loop is then used to obtain the average house size of each city by strong it in the `average_sizes` vector. The `round()` built-in function is also being used to round off the values of decimal values. The `digits` argument with the value of 2 indicates the number of the digit to be rounded. A data frame, `data` is created using `data.frame()` to store the cities' names and average house sizes. Once obtaining the data that needs to be plotted, the `geom_bar()` is used to plot a bar chart. The `fill` argument with the value of "`skyblue`" paints the bar colour, and the `width` argument defines the width of the bar. Lastly, the `coord_flip()` is being used to flip the cartesian coordinates.

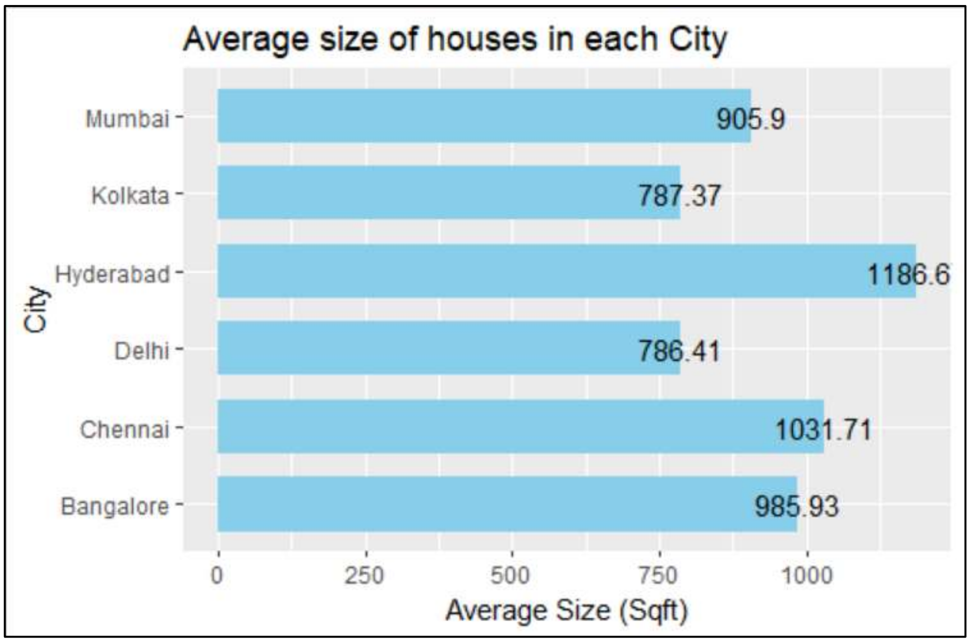
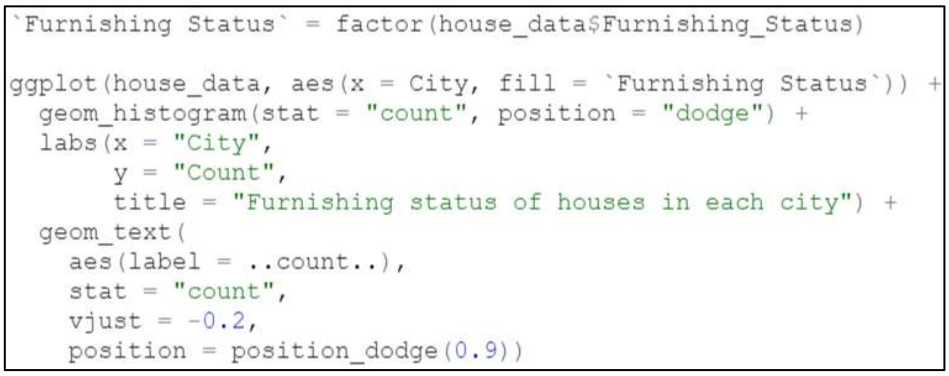


Figure 32: Analysis result 2.1

The above bar chart shows that most houses with bigger sizes are located in Hyderabad and Chennai, where the average house size is above 1,000 sqft. On the other hand, most small-sized houses are located in Kolkata and Delhi, where the average size is below 800 sqft. The medium-sized houses are mostly located in Mumbai and Bangalore, with an average house size of 905.9 and 985.93 sqft respectively.

3.2.2 - Analysis 2.2: Find the furnishing status of houses in each city



```
`Furnishing Status` = factor(house_data$Furnishing_Status)

ggplot(house_data, aes(x = City, fill = `Furnishing Status`)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "City",
       y = "Count",
       title = "Furnishing status of houses in each city") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9))
```

Figure 33: Source code for analysis 2.2

The above source code is to generate a histogram that is being used to visualize the furnishing status of houses in each city. Kindly refer to the source code explanation of analysis 1.4 for built-in functions details of plotting histogram.

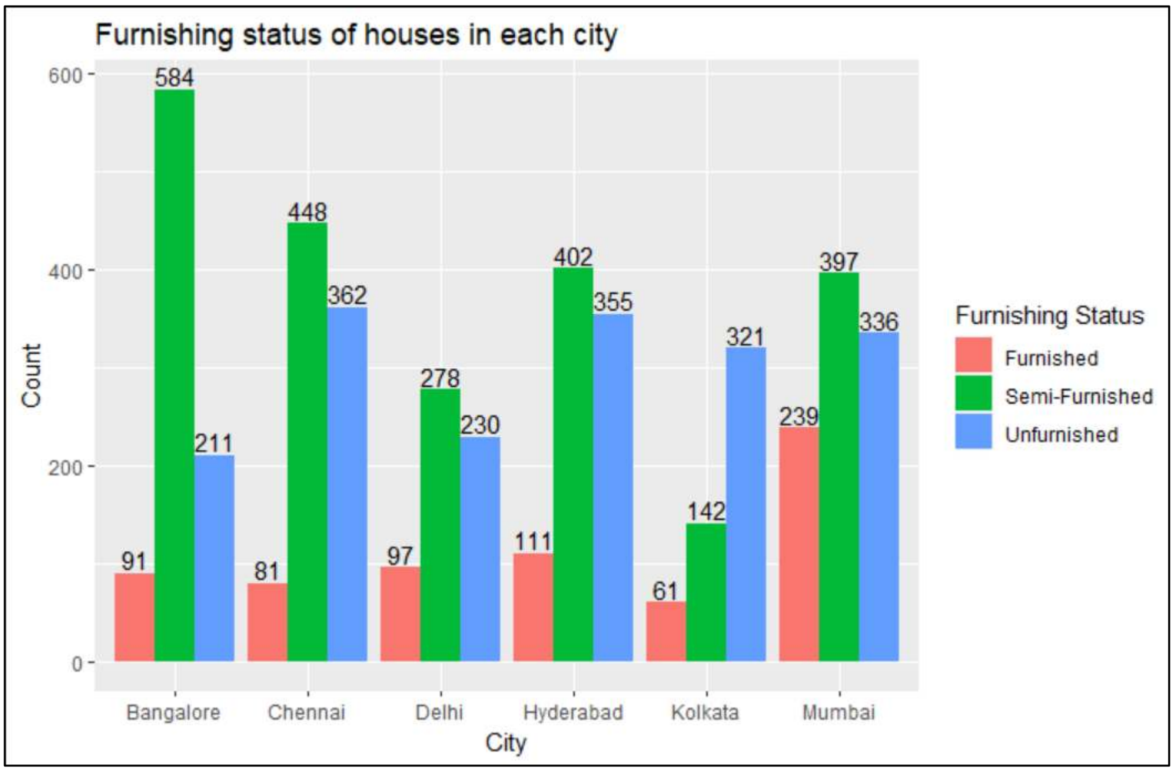
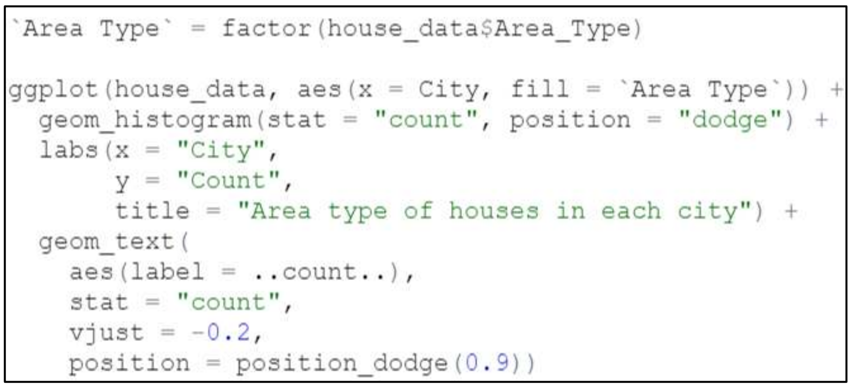


Figure 34: Analysis result 2.2

The histogram above shows that majority of the cities have the most semi-furnished houses following up with unfurnished and furnished houses. However, the percentage of unfurnished houses located in Kolkata is higher as opposed to the other cities.

3.2.3 - Analysis 2.3: Find the area type of houses in each city



```
`Area Type` = factor(house_data$Area_Type)

ggplot(house_data, aes(x = City, fill = `Area Type`)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "City",
       y = "Count",
       title = "Area type of houses in each city") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9))
```

Figure 35: Source code of analysis 2.3

The above source code is to generate a histogram that is being used to visualize the area type of houses in each city. Kindly refer to the source code explanation of analysis 1.4 for built-in functions details of plotting histogram.

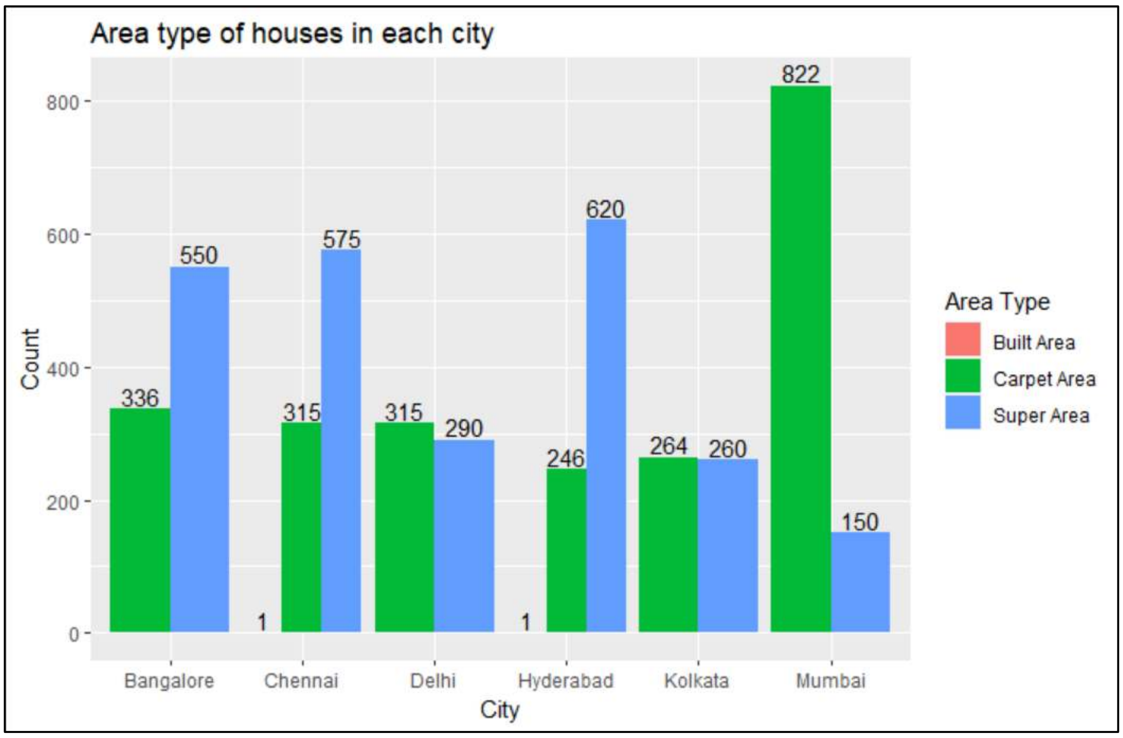
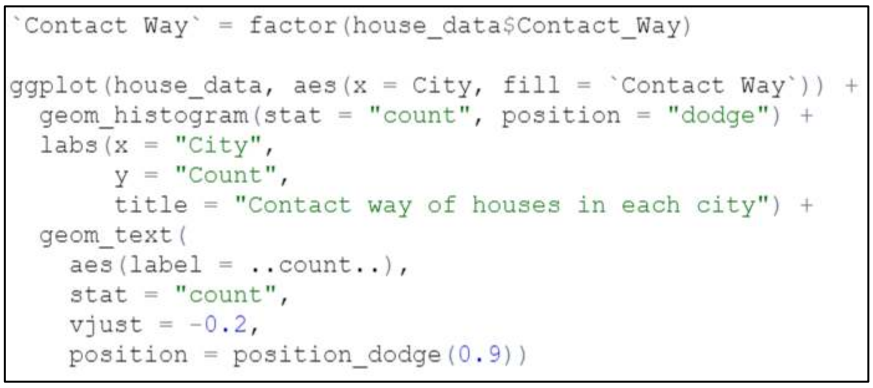


Figure 36: Analysis result 2.3

The above histogram shows that there are more houses in which the size is calculated based on the super area in cities such as Bangalore, Chennai, and Hyderabad. In contrast, there are more houses of which the size is calculated based on carpet area in cities such as Delhi, Kolkata, and Mumbai. Most houses calculated based on the super area are located in Hyderabad and carpet areas are mainly located in Mumbai. There are also two houses in which the size is calculated based on the built area located in Chennai and Hyderabad.

3.2.4 - Analysis 2.4: Find the contact way of renting houses in each city



```
`Contact Way` = factor(house_data$Contact_Way)

ggplot(house_data, aes(x = City, fill = `Contact Way`)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "City",
       y = "Count",
       title = "Contact way of houses in each city") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9))
```

Figure 37: Source code for analysis 2.4

The above source code is to generate a histogram that is being used to visualize the contact way of renting houses in each city. Kindly refer to the source code explanation of analysis 1.4 for built-in functions details of plotting histogram.

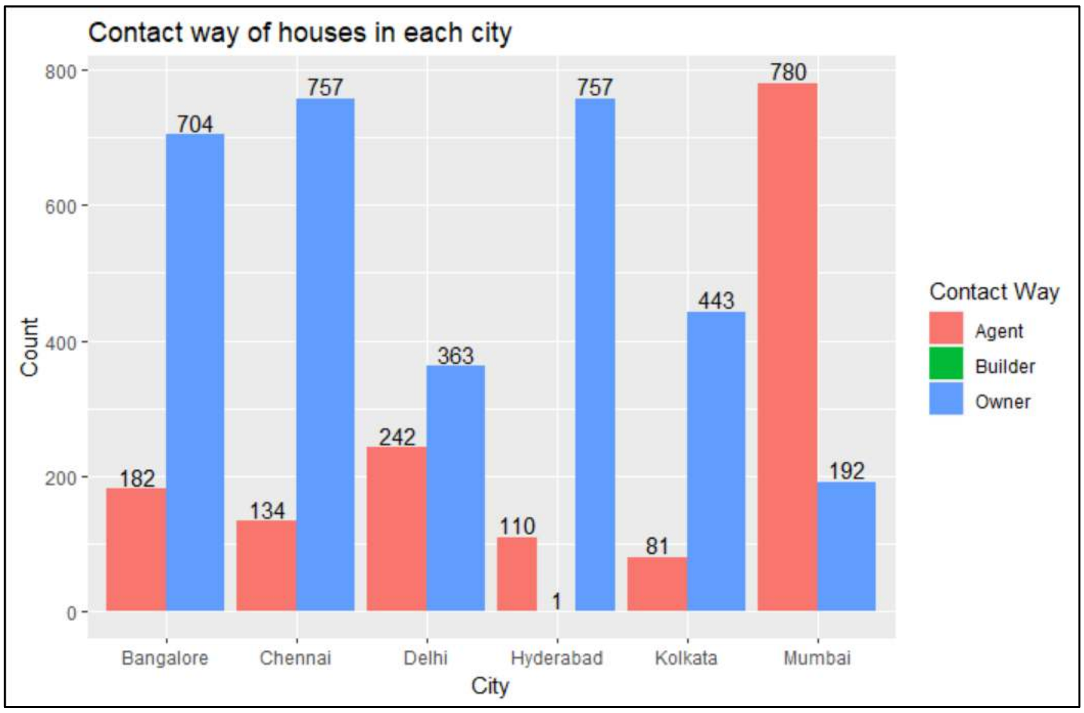
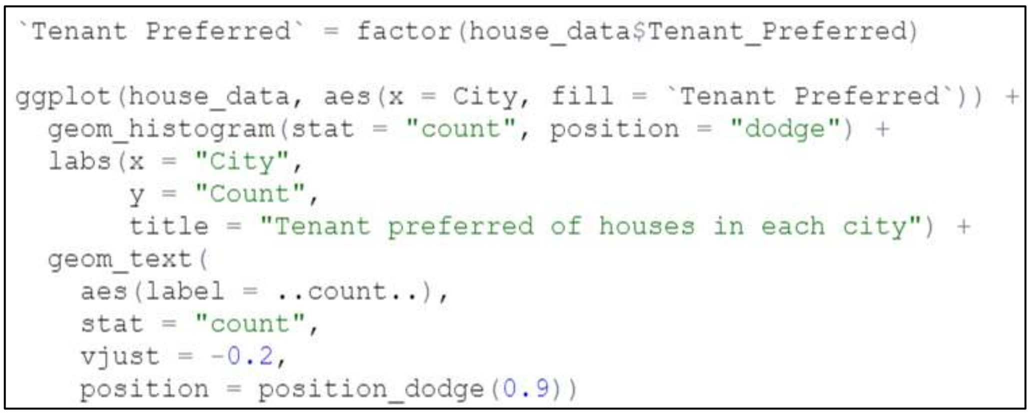


Figure 38: Analysis result 2.4

The histogram above shows that the majority of the cities have most houses that require tenants to contact the owner. In Mumbai, most houses require the tenant to contact the agent as there are a total of 780 houses that have the contact way of the agent and only 192 houses have the contact way of the owner. In Hyderabad, there is one house that requires the tenant to contact the builder.

3.2.5 - Analysis 2.5: Find the tenant preferred by houses in each city



```
`Tenant Preferred` = factor(house_data$Tenant_Preferred)

ggplot(house_data, aes(x = City, fill = `Tenant Preferred`)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "City",
       y = "Count",
       title = "Tenant preferred of houses in each city") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9))
```

Figure 39: Source code for analysis 2.5

The above source code is to generate a histogram that is being used to visualize the tenant preferred by houses in each city. Kindly refer to the source code explanation of analysis 1.4 for built-in functions details of plotting histogram.

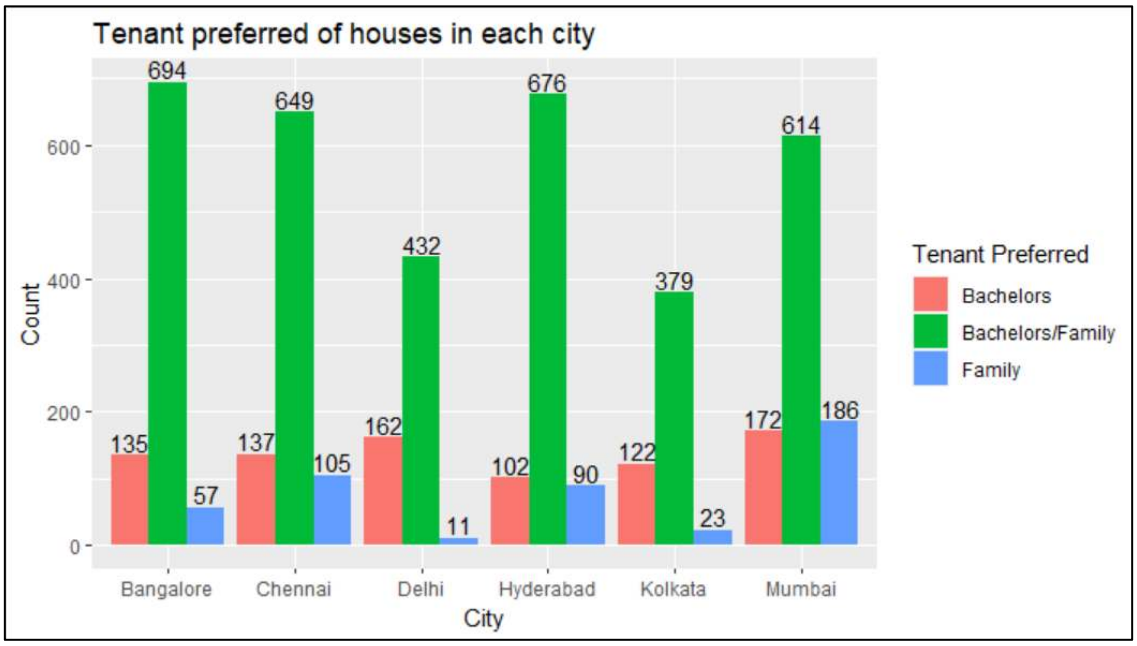
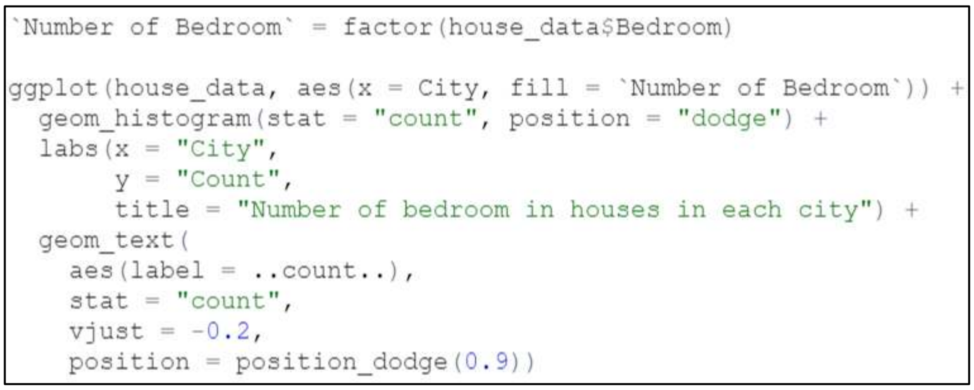


Figure 40: Analysis result 2.5

The histogram above shows that houses that are preferred to be rented out to bachelors or family have the largest proportion in all cities. Most of the cities have houses with tenants preferred by bachelors more than family except in Mumbai. The number of houses that are preferred to be rented to the family is higher than bachelors in Mumbai.

3.2.6 - Analysis 2.6: Find the number of bedrooms in houses in each city



```r
`Number of Bedroom` = factor(house_data$Bedroom)

ggplot(house_data, aes(x = City, fill = `Number of Bedroom`)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "City",
       y = "Count",
       title = "Number of bedroom in houses in each city") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9))
```

Figure 41: Source code for analysis 2.6

The above source code is to generate a histogram that is being used to visualize the number of bedrooms in houses in each city. Kindly refer to the source code explanation of analysis 1.4 for built-in functions details of plotting histogram.

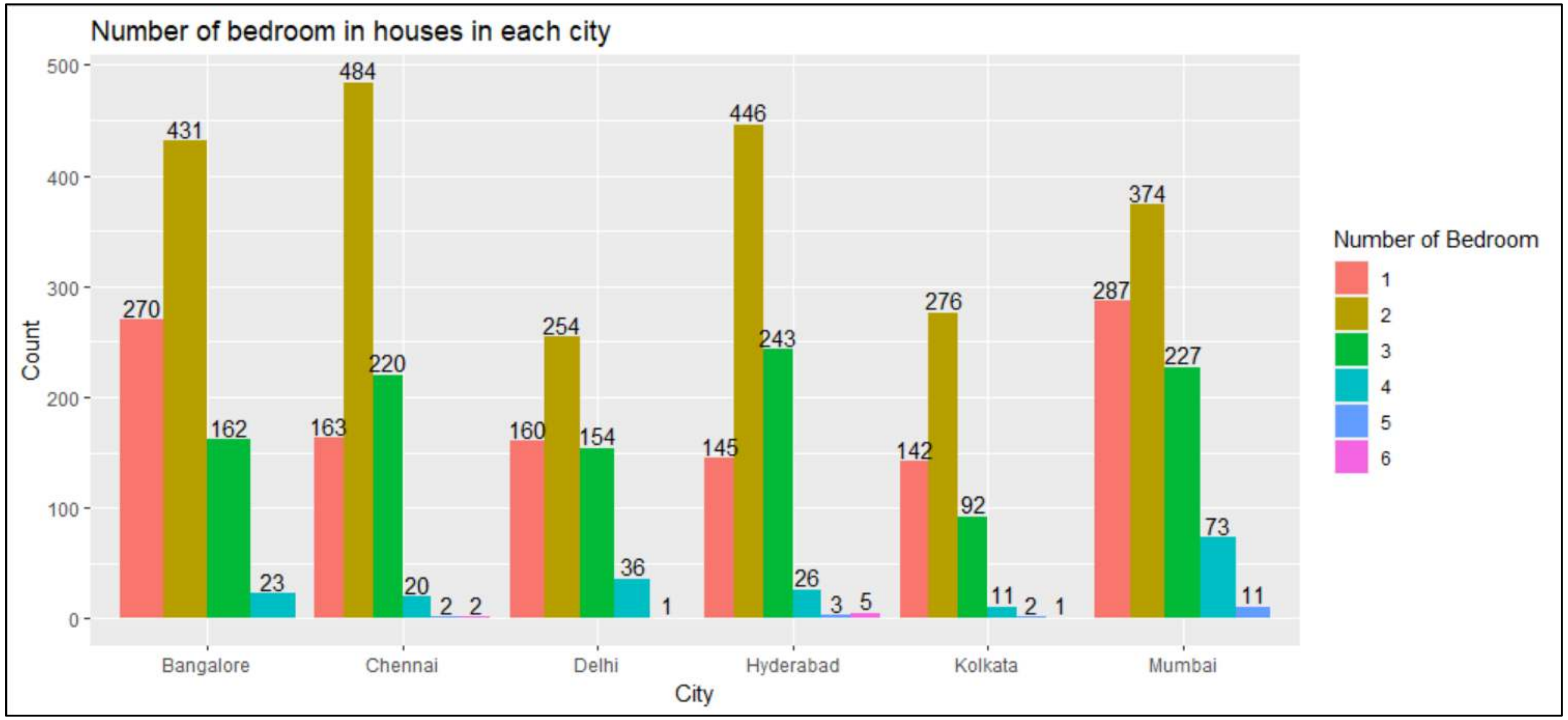


Figure 42: Analysis result 2.6

The most obvious result that can be observed through the histogram above is that most of the houses in all cities are having 2 bedrooms. The second most houses in cities such as Bangalore, Delhi, Kolkata, and Mumbai are having one bedroom. As for Chennai and Hyderabad, houses with 3 bedrooms are the second most houses. Houses with more than 4 bedrooms are located in all cities except Bangalore that is only having houses with up to 4 bedrooms.

3.2.7 - Analysis 2.7: Find the number of the bathroom in houses in each city

Due to the high similarity of analysis techniques, kindly refer to analysis 2.6 for detailed information.

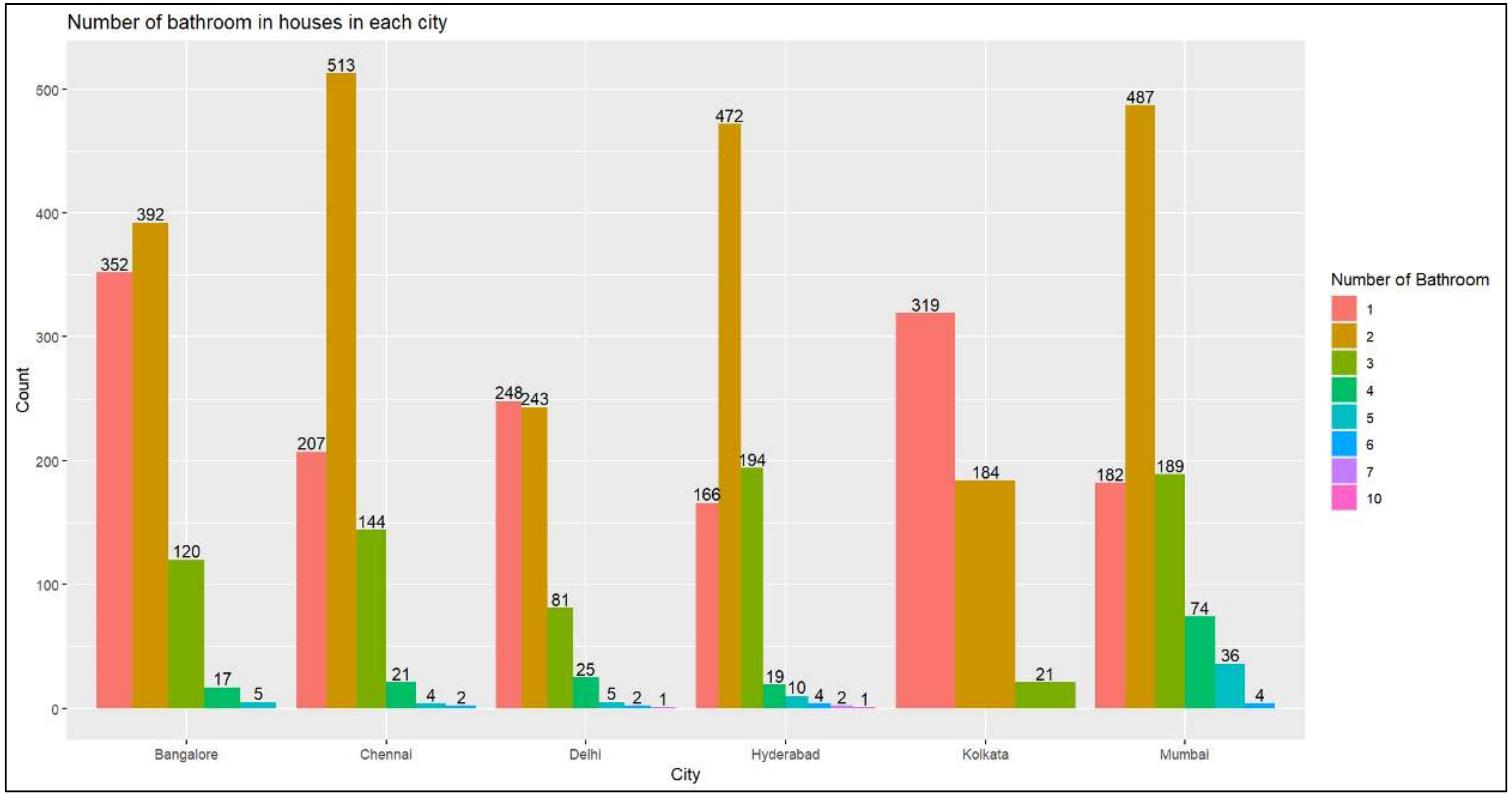


Figure 43: Analysis result 2.7

The majority of houses in all cities are having 2 bathrooms except in Delhi and Kolkata, which have houses with only one bathroom the most. Houses with more than 3 bathrooms are located in all cities except in Kolkata that only has houses with up to 3 bathrooms.

3.2.8 - Conclusion for question 2

There are a total of 7 analyses being done for question 2 to find out what kind of houses there are in each city. The table below summarizes all the attributes of houses in each city.

Table 1: Attributes of houses in each city

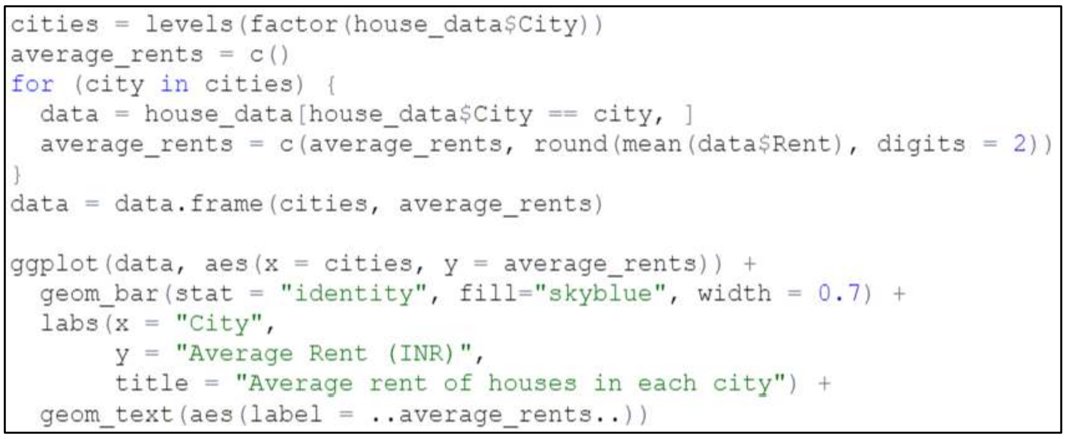
City/Attributes	Size	Furnishing status	Area type	Contact way	Preferred tenant
Bangalore	Medium	Semi-furnished	Super area	Owner	Family/Bachelor
Chennai	Large	Semi-furnished	Super area	Owner	Family/Bachelor
Delhi	Small	Semi-furnished	Carpet area	Owner	Family/Bachelor
Hyderabad	Large	Semi-furnished	Super area	Owner	Family/Bachelor
Kolkata	Small	Unfurnished	Carpet area	Owner	Family/Bachelor
Mumbai	Medium	Semi-furnished	Carpet area	Agent	Family/Bachelor

There are several interesting results have been found throughout those analyses. In analysis 2.2, it is found that all cities are mostly having semi-furnished houses, except for Kolkata that is mostly having unfurnished houses. Another result found in analysis 2.4 is that most houses in Mumbai are having agents as the contact way while houses in other cities are having owners as the most common contact way.

3.3 - Question 3: How expensive is it to rent a house in each city?

This question mainly focuses on finding different aspects of house rent in each city.

3.3.1 - Analysis 3.1: Find the average rent of houses in each city



```r
cities = levels(factor(house_data$City))
average_rents = c()
for (city in cities) {
  data = house_data[house_data$City == city, ]
  average_rents = c(average_rents, round(mean(data$Rent), digits = 2))
}
data = data.frame(cities, average_rents)

ggplot(data, aes(x = cities, y = average_rents)) +
  geom_bar(stat = "identity", fill="skyblue", width = 0.7) +
  labs(x = "City",
       y = "Average Rent (INR)",
       title = "Average rent of houses in each city") +
  geom_text(aes(label = ..average_rents..))
```

Figure 44: Source code for analysis 3.1

The above source code generates a bar chart to visualize the average rent of houses in each city. Kindly refers to the source code explanation of analysis 2.1 for the details of obtaining data and built-in functions.

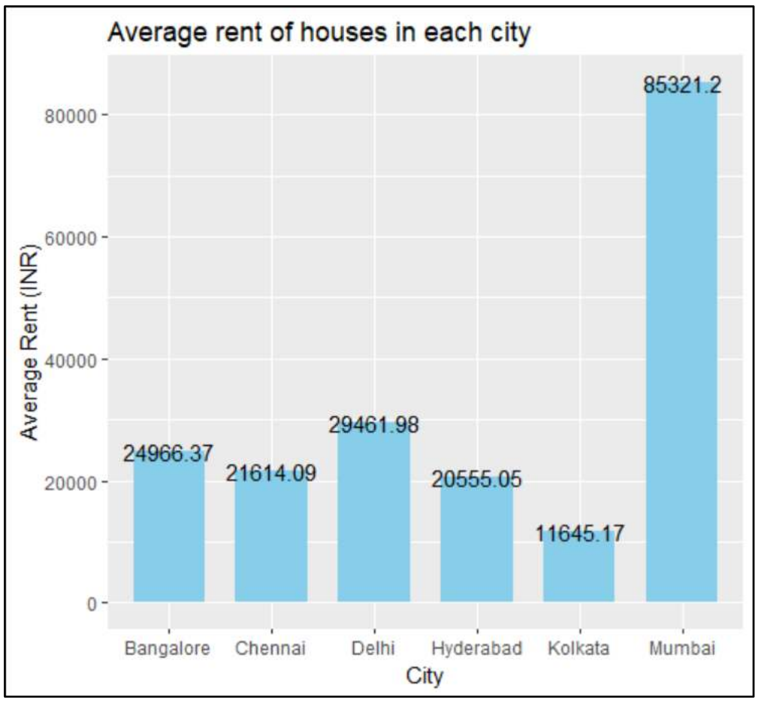
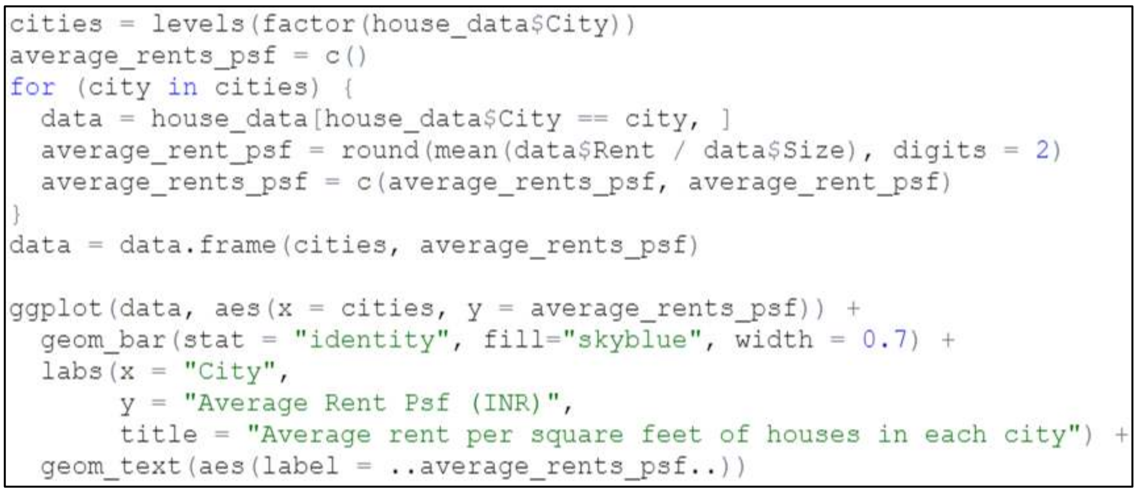


Figure 45: Analysis result 3.1

The bar chart above shows that houses in Mumbai have the highest average rent among all the other cities, which is around 85,000 INR. The average rent in cities such as Bangalore, Chennai, Delhi, and Hyderabad ranges from 20,000 INR to 30,000 INR. Kolkata is the cheapest city to rent a house as it has the lowest average rent of houses at around 11,600 INR.

3.3.2 - Analysis 3.2: Find the average rent per sqft of houses in each city



```
cities = levels(factor(house_data$City))
average_rents_psf = c()
for (city in cities) {
  data = house_data[house_data$City == city, ]
  average_rent_psf = round(mean(data$Rent / data$Size), digits = 2)
  average_rents_psf = c(average_rents_psf, average_rent_psf)
}
data = data.frame(cities, average_rents_psf)

ggplot(data, aes(x = cities, y = average_rents_psf)) +
  geom_bar(stat = "identity", fill="skyblue", width = 0.7) +
  labs(x = "City",
       y = "Average Rent Psf (INR)",
       title = "Average rent per square feet of houses in each city") +
  geom_text(aes(label = ..average_rents_psf..))
```

Figure 46: Source code for analysis 3.2

The above source code aims to generate a bar chart that is being used to visualize the average rent per sqft of houses in each city. Kindly refers to the source code explanation of analysis 2.1 for the details of obtaining data and plotting a bar chart.

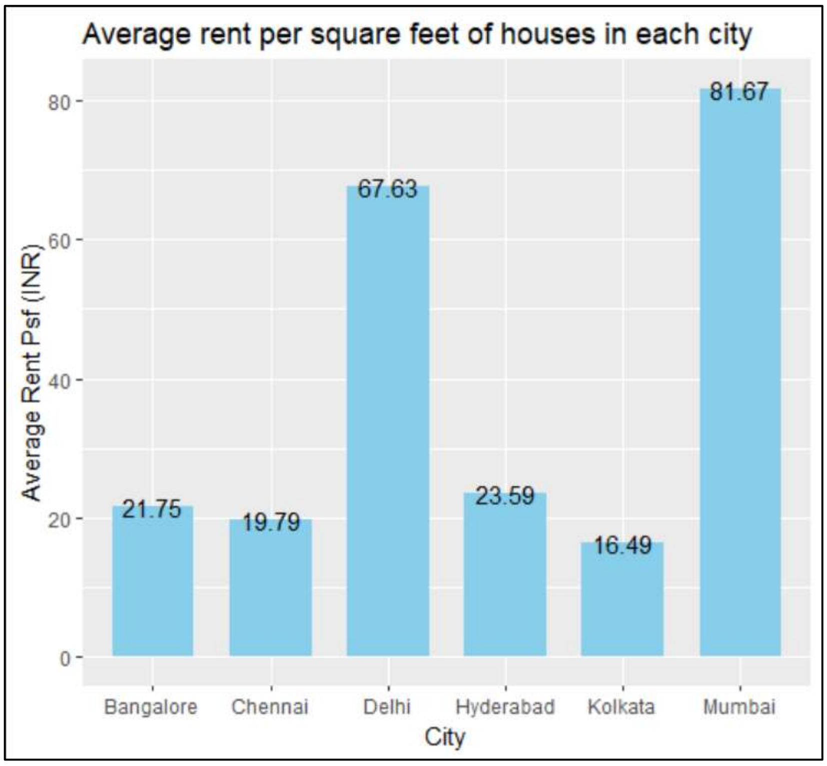


Figure 47: Analysis result 3.2

The analysis result 3.2 shows that houses in Mumbai have the highest average rent per square foot (psf) among all the other cities, which is around 80 INR psf. Delhi is the city that has the second highest average rent psf, which stands at 67.63 INR Psf. The average rent psf in cities such as Bangalore, Chennai, Hyderabad, and Kolkata ranges from 15 to 25 INR psf.

3.3.3 - Analysis 3.3: Find the average rent of houses' room in each city

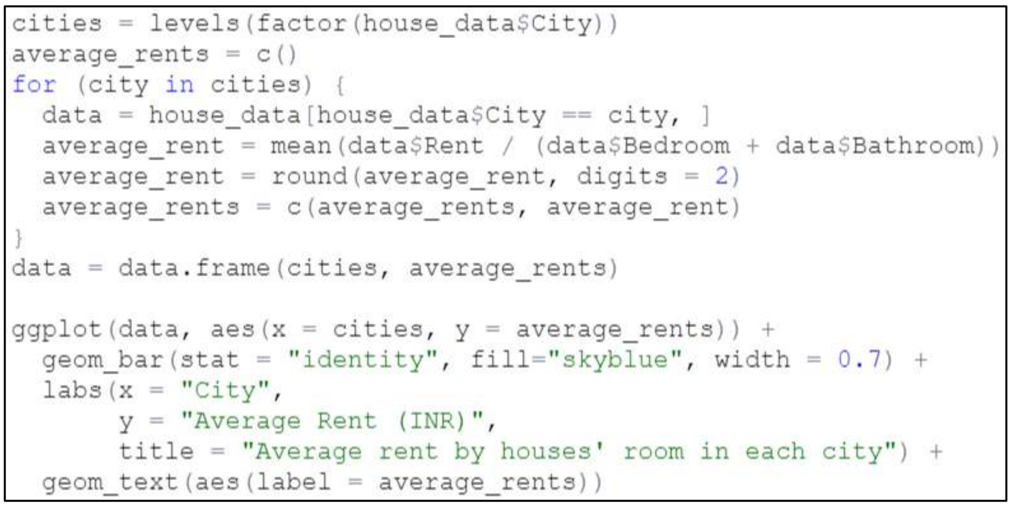

```
cities = levels(factor(house_data$City))
average_rents = c()
for (city in cities) {
  data = house_data[house_data$City == city, ]
  average_rent = mean(data$Rent / (data$Bedroom + data$Bathroom))
  average_rent = round(average_rent, digits = 2)
  average_rents = c(average_rents, average_rent)
}
data = data.frame(cities, average_rents)

ggplot(data, aes(x = cities, y = average_rents)) +
  geom_bar(stat = "identity", fill="skyblue", width = 0.7) +
  labs(x = "City",
       y = "Average Rent (INR)",
       title = "Average rent by houses' room in each city") +
  geom_text(aes(label = average_rents))
```

Figure 48: Source code for analysis 3.3

The above source code aims to generate a bar chart that is being used to visualize the average rent of houses room in each city. The number of house rooms includes both the bathroom and bedroom in a house. Kindly refers to the source code explanation of analysis 2.1 for the details of obtaining data and plotting a bar chart.

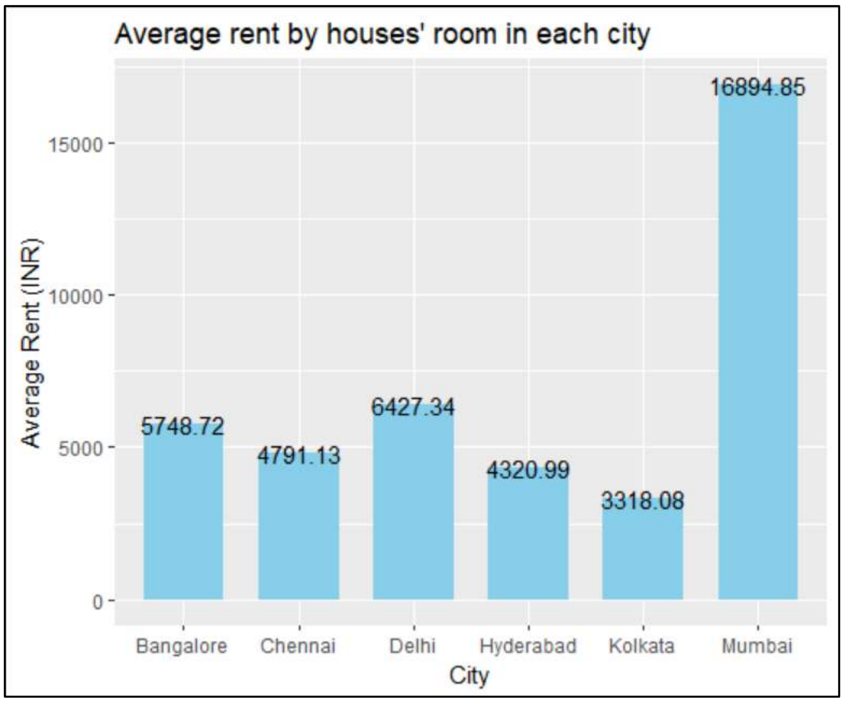


Figure 49: Analysis result 3.3

The analysis result 3.3 shows that Mumbai has the highest average rent of houses room, where each room is being rented at an average of 16,894.85 INR. In contrast, the average rent of a room in Kolkata's houses has the lowest rent of 3,318.08 INR. The average rent of a room in cities such as Bangalore, Chennai, Hyderabad, and Delhi ranges from 4,000 to 7,000 INR.

3.3.4 - Conclusion for question 3

There are a total of 3 analyses being done for question 3 to find out the rent of houses in each city from 3 different aspects, which are average house rent, rent psf and room. The table below summarizes all these aspects, where the city that is having more expensive rental houses will have a gradient colour that is more towards red and vice versa.

Table 2: Average rent of houses in each city

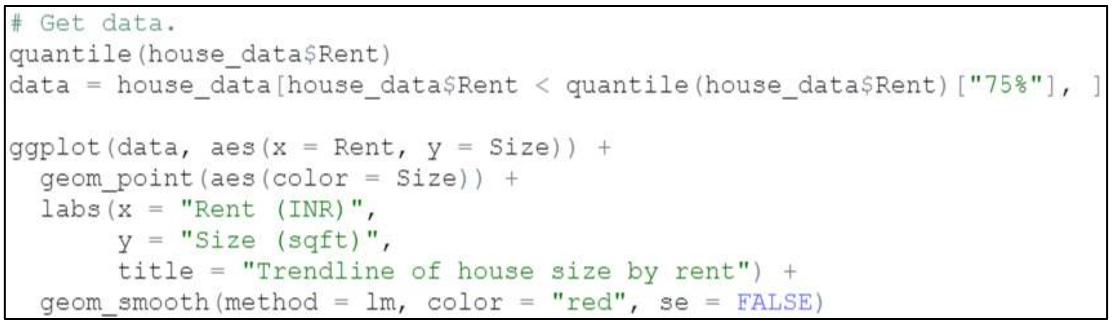
City/Rent (INR)	Average house rent	Average rent psf	Average room rent
Bangalore	INR 24,966.37	INR 21.75	INR 5,748.72
Chennai	INR 21,614.09	INR 19.79	INR 4,791.13
Delhi	INR 29,461.98	INR 67.63	INR 6,427.34
Hyderabad	INR 20,555.05	INR 23.59	INR 4,320.99
Kolkata	INR 11,645.17	INR 16.49	INR 3,318.08
Mumbai	INR 85,321.20	INR 81.67	INR 16,894.85

In short, Mumbai is the most expensive city to rent a house and Kolkata is the most affordable city to rent a house. An interesting thing has been found in analysis 3.2, which is the average rent psf in Delhi is significantly higher as compared to other aspects like average room rent. This means the houses in Delhi can be expensive for tenants who wish to rent a larger house and be affordable at the same time for tenants who are fine with living in a smaller house.

3.4 - Question 4: Does the rent directly affect the size of houses?

The house rent in each city is being analysed thoroughly in question 3, and this question aims to find out whether the rent will affects the size of houses.

3.4.1 - Analysis 4.1: Find the relationship between the rent with house size



```
# Get data.
quantile(house_data$Rent)
data = house_data[house_data$Rent < quantile(house_data$Rent)["75%"], ]

ggplot(data, aes(x = Rent, y = Size)) +
  geom_point(aes(color = Size)) +
  labs(x = "Rent (INR)",
       y = "Size (sqft)",
       title = "Trendline of house size by rent") +
  geom_smooth(method = lm, color = "red", se = FALSE)
```

Figure 50: Source code for analysis 4.1

The above source code aims to generate a scatter plot graph that is being used to visualize the relationship between house size and rent. The `quantile()` function is used to obtain the sample quantiles of a dataset. The maximum rent of this analysis uses the 75th percentile of the dataset to reduce the range of high differences in rent. The `geom_smooth()` function is being used to add a trendline over the existing plot and the default trendline is a loess line. Therefore, the `method` parameter with the value of “`lm`” which stands for the linear model is used to add a straight-line linear model (Ebner, 2022). The `colour` parameter is being used to specify the colour of the trendline. The `se` parameter is being specified as false to override the default value of true, which will add the confidence interval around the trendline.

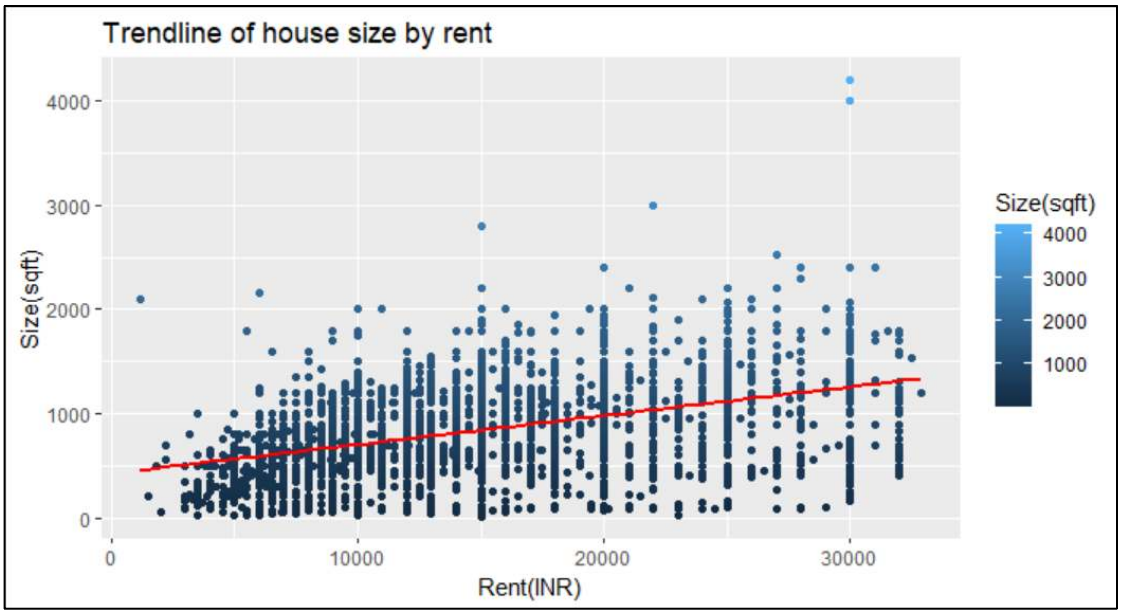
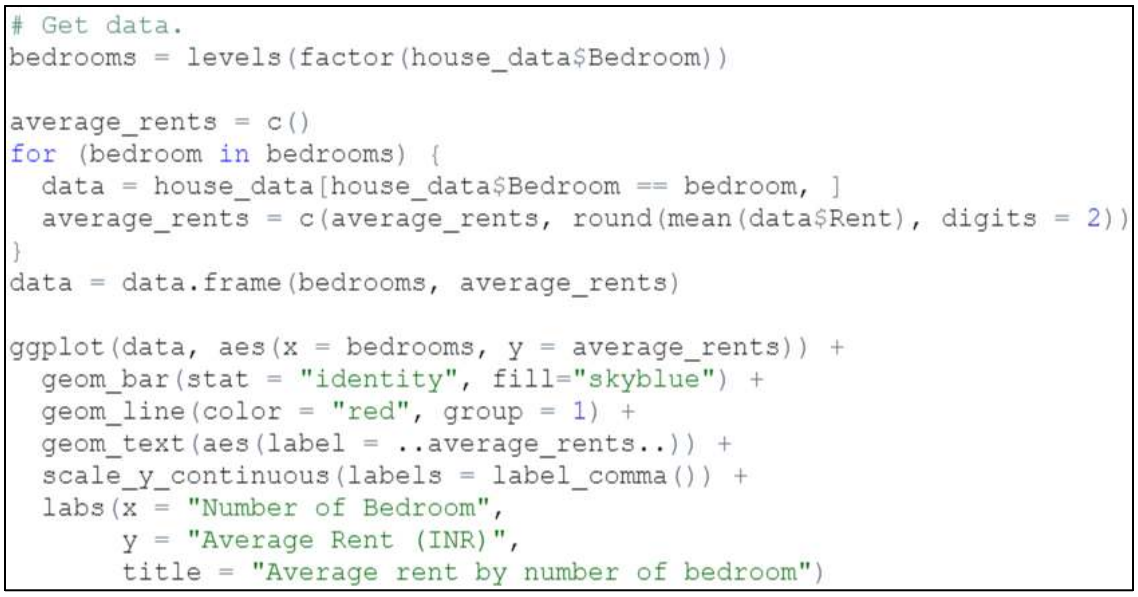


Figure 51: Analysis result 4.1

The scatter plots in the analysis result in 4.1 shows that the higher the rent, the wider the range of the plot is being scattered vertically. Overall, the rent will affect the size of houses as the trendline shows that houses with higher rent tend to have higher size psf and vice versa.

3.4.2 - Analysis 4.2: Find relationships between average rent with bedroom number



```
# Get data.
bedrooms = levels(factor(house_data$Bedroom))

average_rents = c()
for (bedroom in bedrooms) {
  data = house_data[house_data$Bedroom == bedroom, ]
  average_rents = c(average_rents, round(mean(data$Rent), digits = 2))
}
data = data.frame(bedrooms, average_rents)

ggplot(data, aes(x = bedrooms, y = average_rents)) +
  geom_bar(stat = "identity", fill="skyblue") +
  geom_line(color = "red", group = 1) +
  geom_text(aes(label = ..average_rents..)) +
  scale_y_continuous(labels = label_comma()) +
  labs(x = "Number of Bedroom",
       y = "Average Rent (INR)",
       title = "Average rent by number of bedroom")
```

Figure 52: Source code for analysis 4.2

The above source code aims to generate a bar chart that is being used to visualize the relationship between the average rent with the bedroom number. Kindly refers to the source code explanation of analysis 2.1 for the details of obtaining data and plotting a bar chart. In this analysis, the `geom_line()` function is being used to connect the observation of the bar chart. The `group` parameter is specified as one to override the default grouping of the x-axis value, which is the number of bedrooms. The `scale_y_continuous()` from scales package is used to avoid abbreviated axis labels (scientific notation) such as 1e+00.

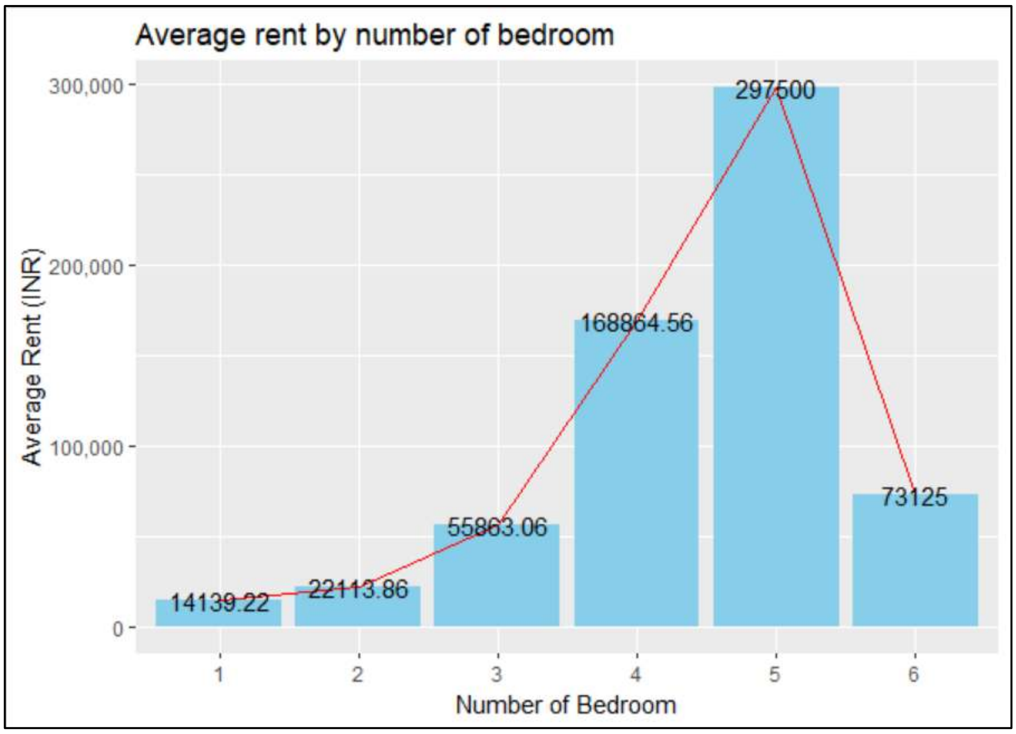
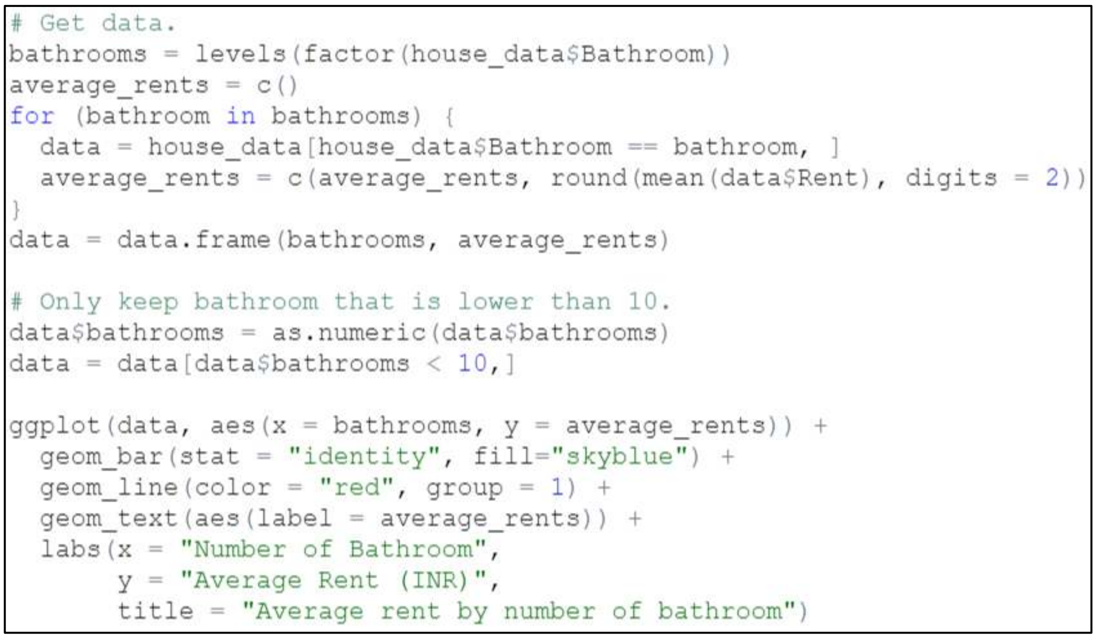


Figure 53: Analysis result 4.2

The bar chart above shows that the houses with higher average rent will be having a higher number of bedrooms. However, this trend stops as it reaches the houses with 5 bedrooms and starts to decline, where the average rent for a 6 bedrooms house is 73,125 INR.

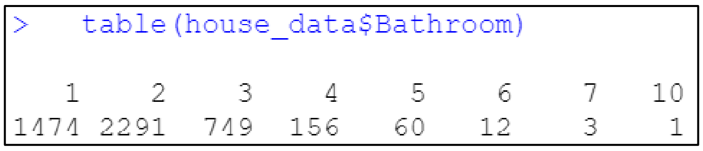
3.4.3 - Analysis 4.3: Find relationships between average rent with bathroom number



```
# Get data.
bathrooms = levels(factor(house_data$Bathroom))
average_rents = c()
for (bathroom in bathrooms) {
  data = house_data[house_data$Bathroom == bathroom, ]
  average_rents = c(average_rents, round(mean(data$Rent), digits = 2))
}
data = data.frame(bathrooms, average_rents)

# Only keep bathroom that is lower than 10.
data$bathrooms = as.numeric(data$bathrooms)
data = data[data$bathrooms < 10,]

ggplot(data, aes(x = bathrooms, y = average_rents)) +
  geom_bar(stat = "identity", fill="skyblue") +
  geom_line(color = "red", group = 1) +
  geom_text(aes(label = average_rents)) +
  labs(x = "Number of Bathroom",
       y = "Average Rent (INR)",
       title = "Average rent by number of bathroom")
```

Figure 54: Source code for analysis 4.3

The above source code aims to generate a bar chart that is being used to visualize the relationship between the average rent with the bathroom number. This analysis is highly similar to analysis 4.2, the only difference is that the data is being derived with the condition of having only houses with a maximum of 10 bathrooms. This is because there is only one house with 10 bathrooms in the whole dataset, which is negligible to be excluded from the analysis.



```
>   table(house_data$Bathroom)

   1    2    3    4    5    6    7   10
1474 2291  749  156   60   12    3    1
```

Figure 55: Count of houses' bathroom number

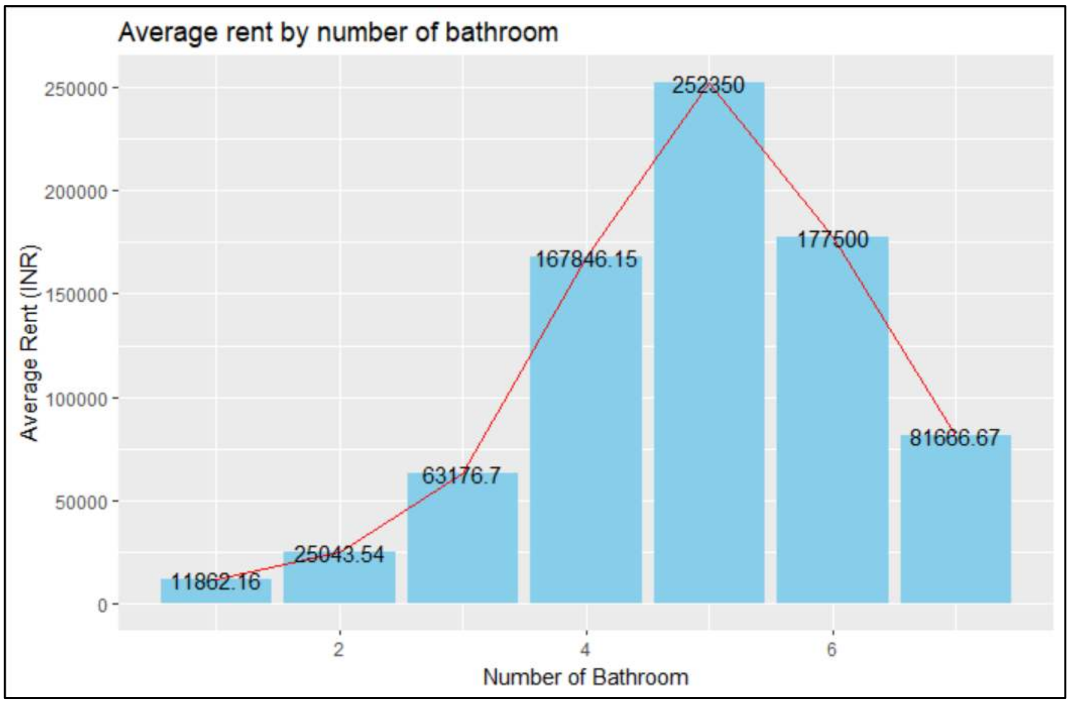


Figure 56: Analysis result 4.3

The bar chart above shows that the houses with higher average rent will be having a higher number of bathrooms. However, this trend stops as it reaches the houses with 5 bathrooms and starts to decline, where the average rent for houses with 6 and 7 bathrooms is 177,500 INR and 81,666.67 INR respectively.

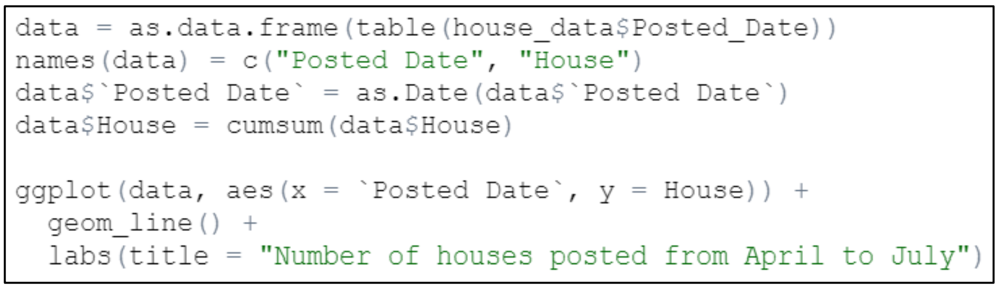
3.4.4 - Conclusion for question 4

There are a total of 3 analyses being done for question 4 to find out whether the rent will affects the size of houses. The analysis result 4.1 shows that the higher the house rent, the higher the size of the house in sqft. However, the result of analysis 4.2 and 4.3 shows that higher rent does not always mean more rooms in the house.

3.5 - Question 5: How is the property rental market performance?

This question aims to find out the performance of the property rental market in India.

3.5.1 - Analysis 5.1: Find the consistency of houses posted in the last few months



```
data = as.data.frame(table(house_data$Posted_Date))
names(data) = c("Posted Date", "House")
data$`Posted Date` = as.Date(data$`Posted Date`)
data$House = cumsum(data$House)

ggplot(data, aes(x = `Posted Date`, y = House)) +
  geom_line() +
  labs(title = "Number of houses posted from April to July")
```

Figure 57: Source code for analysis 5.1

The above source code generates a line graph to visualize the consistency of houses posted in the last few months. The `data` is assigned with the value returned from the `table()` function, which tabulates the count of houses posted on each date. The `data` is then converted into a data frame and named to plot the pie chart. The data type of `Posted Date` field is also being changed to date using `as.Date()` function. The `cumsum()` function is used to calculate the cumulative sum of the houses posted each day.

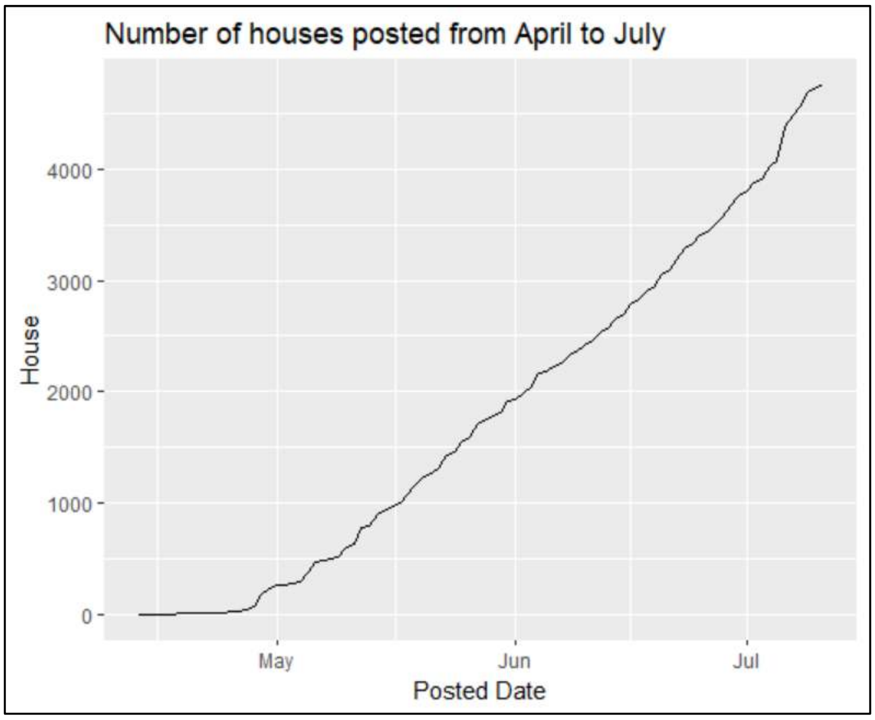
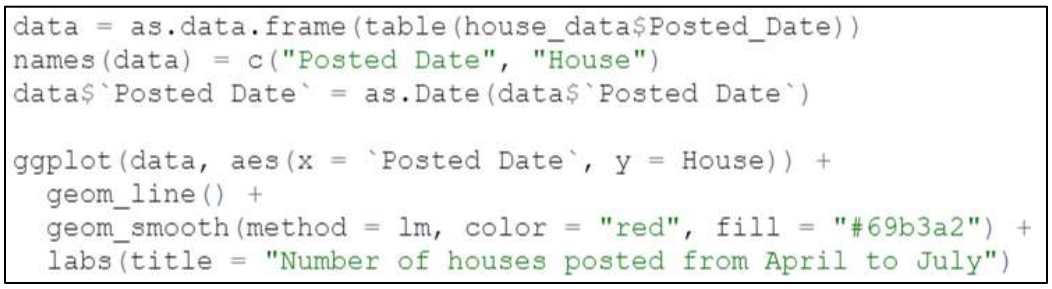


Figure 58: Analysis result 5.1

The line graph above shows that the number of houses posted has consistently increased from the beginning of May until July. This means the market is stable throughout this period as there is no sudden increase or decrease in the number of houses being posted.

3.5.2 - Analysis 5.2: Find the house posted daily in the last few months



```
data = as.data.frame(table(house_data$Posted_Date))
names(data) = c("Posted Date", "House")
data$`Posted Date` = as.Date(data$`Posted Date`)

ggplot(data, aes(x = `Posted Date`, y = House)) +
  geom_line() +
  geom_smooth(method = lm, color = "red", fill = "#69b3a2") +
  labs(title = "Number of houses posted from April to July")
```

Figure 59: Source code for analysis 5.2

The above source code is being used to generate a line graph to visualize the number of houses posted in the last few months. Kindly refers to the source code explanation of the analysis 5.1 for detailed information on obtaining data and built-in functions. Kindly refers to the source code explanation of `geom_smooth()` in analysis 4.1.

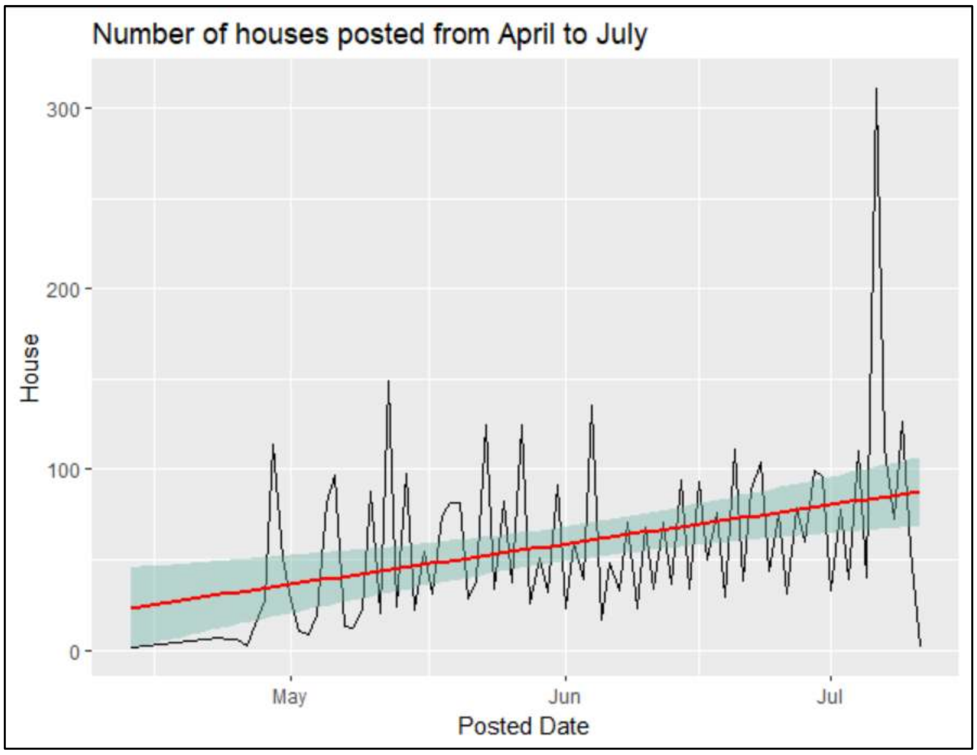
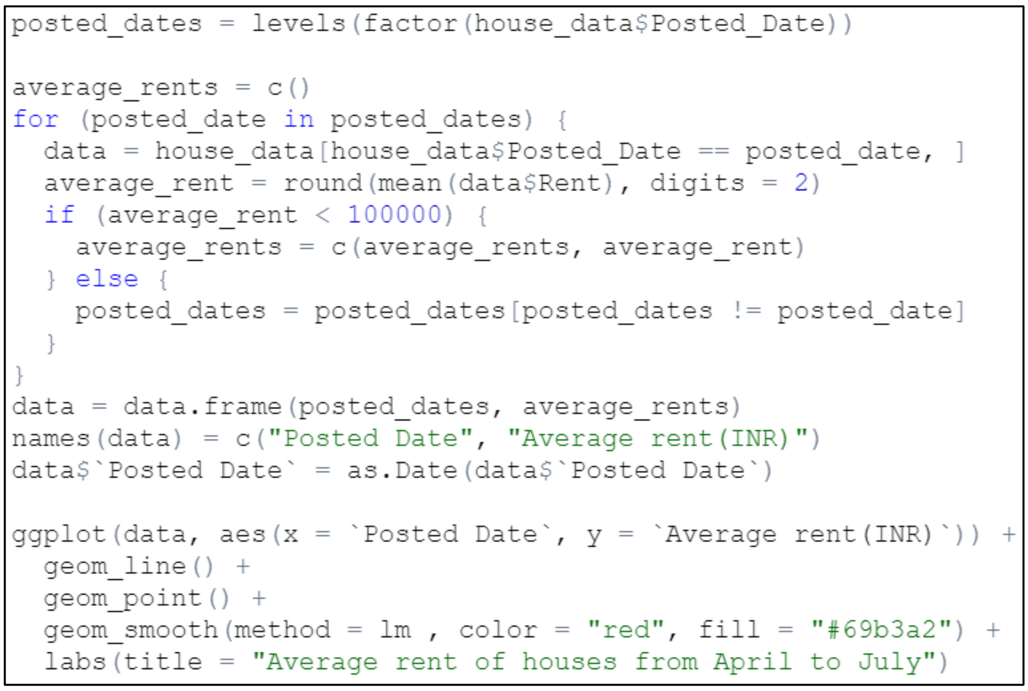


Figure 60: Analysis result 5.2

The line graph above shows that the number of houses posted from May until July has gradually increased based on the trendline. The number of houses posted during the mid of May until early June is relatively consistent. This is because this period has a narrower confidence interval (CI) as compared to April and July.

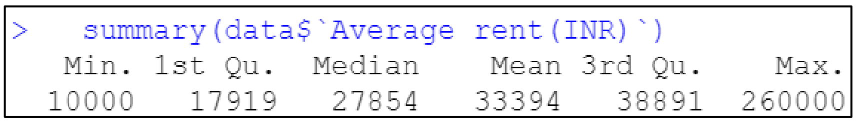
3.5.3 - Analysis 5.3: Find the average rent of houses in the last few months



```
posted_dates = levels(factor(house_data$Posted_Date))

average_rents = c()
for (posted_date in posted_dates) {
  data = house_data[house_data$Posted_Date == posted_date, ]
  average_rent = round(mean(data$Rent), digits = 2)
  if (average_rent < 100000) {
    average_rents = c(average_rents, average_rent)
  } else {
    posted_dates = posted_dates[posted_dates != posted_date]
  }
}
data = data.frame(posted_dates, average_rents)
names(data) = c("Posted Date", "Average rent(INR)")
data$`Posted Date` = as.Date(data$`Posted Date`)

ggplot(data, aes(x = `Posted Date`, y = `Average rent(INR)`)) +
  geom_line() +
  geom_point() +
  geom_smooth(method = lm , color = "red", fill = "#69b3a2") +
  labs(title = "Average rent of houses from April to July")
```

Figure 61: Source code for analysis 5.3

The above source code is being used to generate a connected scatterplot to visualize the average rent of houses in the last few months. The `data` consists of each posted date of the houses and the average rent of houses posted on that particular date. The `data` excludes date with average house rent above 100,000 INR to lower the high difference in average rent which range from 10,000 INR to 260,000 INR.



```
>  summary(data$`Average rent(INR)`)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  10000   17919   27854   33394   38891  260000 
```

Figure 62: Summary of average rent

Kindly refers to the source code explanation of `geom_smooth()` in analysis 4.1.

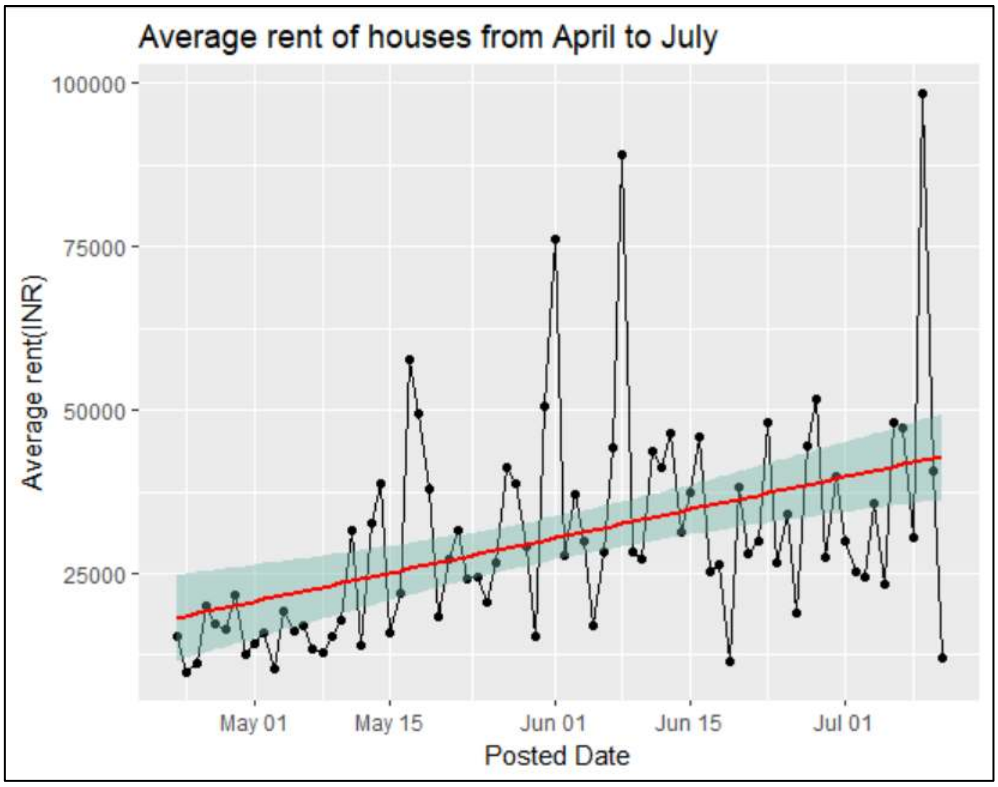


Figure 63: Analysis result 5.3

The connected scatterplot aboveanalysis__result_5__3 shows that the average rent of houses has gradually increased based on the trendline from May until July. The average rent in April and July has a wider confidence interval (CI) as compared to the duration during the mid of May until early June. This means the average rent of houses during the mid of May until early June did not fluctuate much as compared to the time during April and July.

3.5.4 - Analysis 5.4: Find the rent of houses posted in the last week

```
# Get data
last_week_date = max(house_data$Posted_Date) - 7
condition = house_data$Rent < 100000 & house_data$Posted_Date > last_week_date
data = house_data[condition, c("Posted_Date", "Rent")]

ggplot(data, aes(x = as.factor(Posted_Date), y = Rent)) +
  geom_boxplot(fill = "skyblue") +
  labs(x = "Posted Date",
       y = "Rent (INR)",
       title = "House rent on the last week")
```

Figure 64: Source code for analysis 5.4

The above source code is being used to generate box plots to visualize the rent of houses posted in the last week. The `data` dataset is derived from the `house_data` dataset with the condition of having a maximum rent of 100,000 INR. Kindly refer to the reason for doing so under this section. The way of obtaining the data that is posted in the last week is by subtracting the maximum posted date or the latest date with 7. The `geom_boxplot()` is being used to plot the bar plot where the value of `fill` parameter is being used to fill the bar colour.

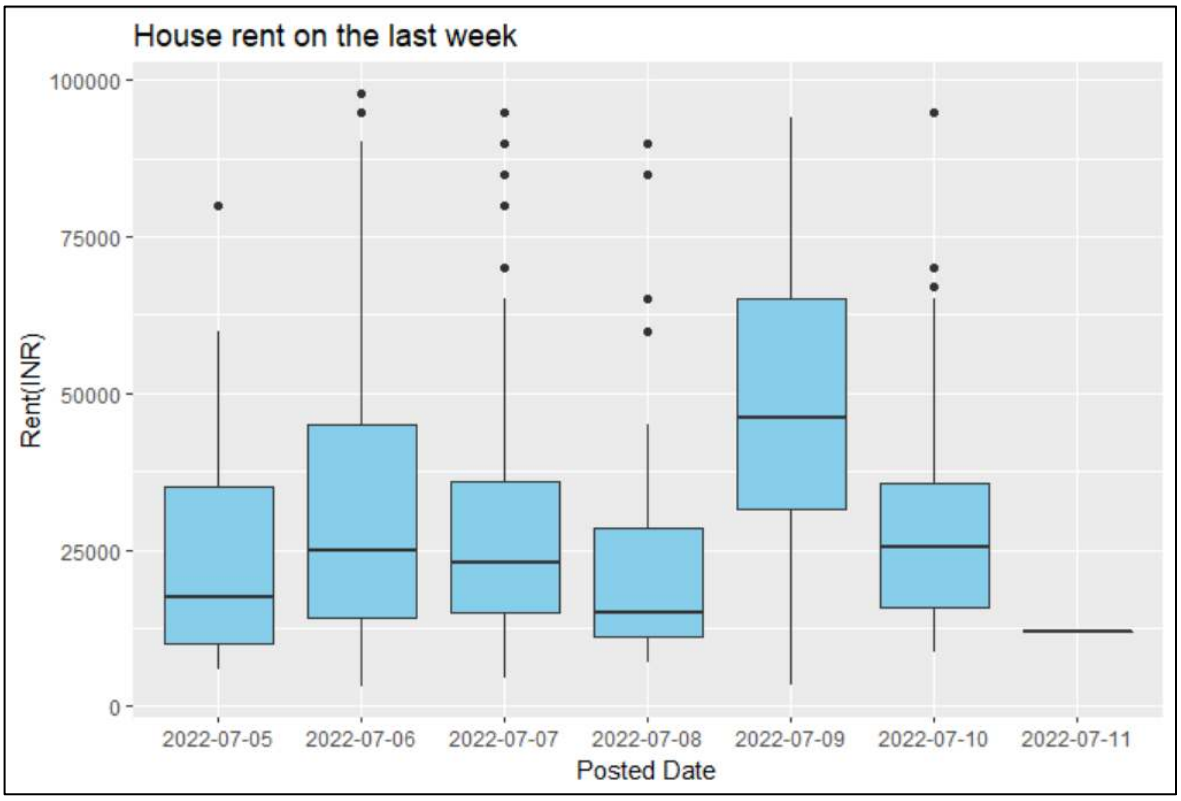


Figure 65: Analysis result 5.4

The box plot above shows that the rent of houses has increased from the 5th of July until the 9th of July. The median rent up to the 8th of July has been remaining below 25,000 INR and it has a sudden spike on the 9th of July, at around 45,000 INR. The rent started to drop after 2 days and remain at 17,500 INR on the 11th of July. On the 6th and 9th of July, the house rent range from 4,000 INR up to 90,000 INR.

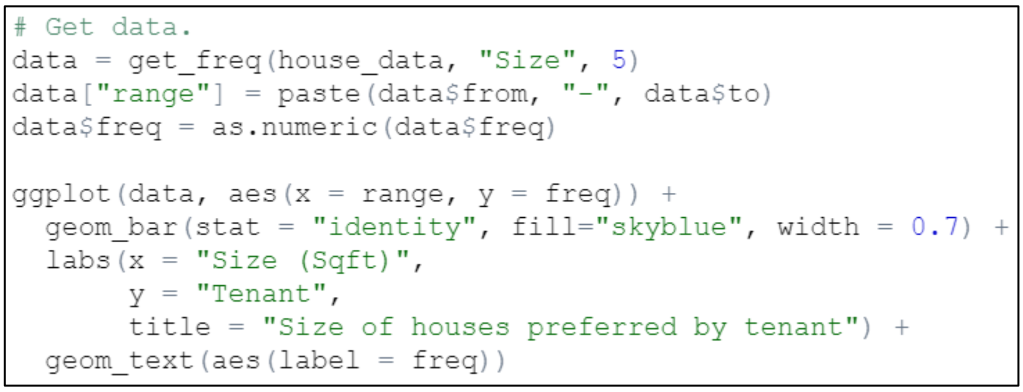
3.5.5 - Conclusion for question 5

There are a total of 4 analyses being done for question 5 to find out the property rental market performance in the last few months. The analysis results show that the market is performing well as the houses are posted consistently, and the average house rent has been increasing steadily from April until July.

3.6 - Question 6: What kind of houses do tenants prefer to live in?

This question focuses on finding the type of houses tenants prefer to rent and live in.

3.6.1 - Analysis 6.1: Find the size of houses preferred by tenants



```
# Get data.
data = get_freq(house_data, "Size", 5)
data["range"] = paste(data$from, "-", data$to)
data$freq = as.numeric(data$freq)

ggplot(data, aes(x = range, y = freq)) +
  geom_bar(stat = "identity", fill="skyblue", width = 0.7) +
  labs(x = "Size (Sqft)",
       y = "Tenant",
       title = "Size of houses preferred by tenant") +
  geom_text(aes(label = freq))
```

Figure 66: Source code for analysis 6.1

The above source code generates a bar chart to visualize the size of houses preferred by tenants. The `get_freq()` function is being used to obtain the frequency of the same data within a specific column, that is the count of tenants and the range of houses' sizes. Kindly refer to the source code explanation in analysis 2.1 for details of plotting a bar chart.

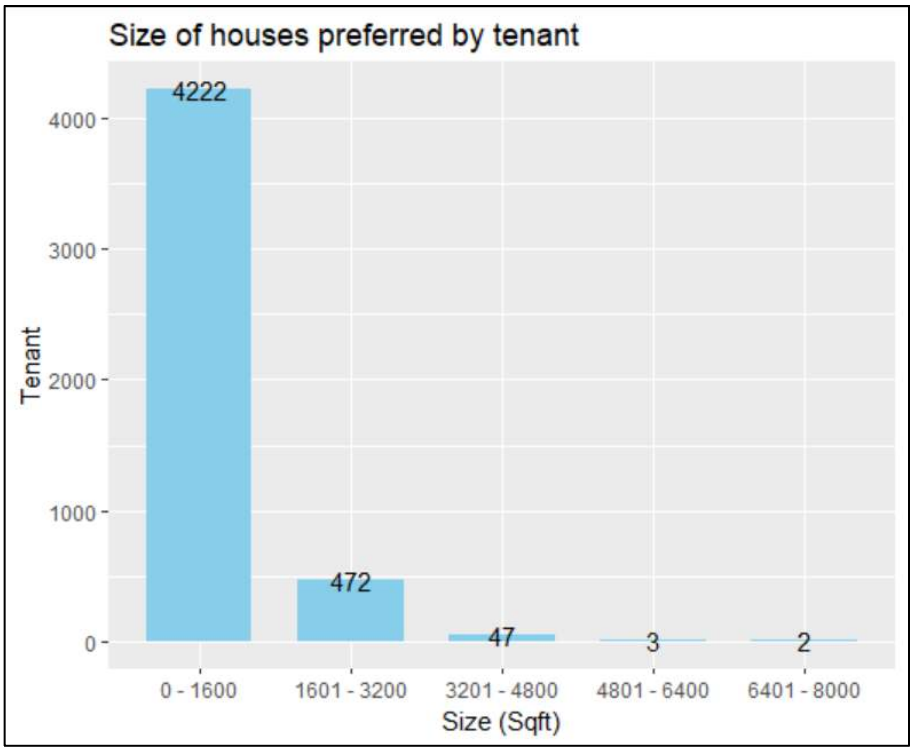


Figure 67: Analysis result 6.1

The bar chart above shows that most tenant prefers houses with a size that is below 1,600 sqft, with 4,222 of them. There are 472 tenants who prefer houses with sizes from 1,601 to 3,200 sqft. There is only a total of 52 tenants who prefer houses with 3,201 sqft and above.

3.6.2 - Analysis 6.2: Find the number of bedrooms preferred by tenants

```
data = as.data.frame(table(house_data$Bedroom))
names(data) = c("Bedroom", "Tenant")

ggplot(data, aes(x = Bedroom, y = Tenant)) +
  geom_bar(stat = "identity", fill="skyblue", width = 0.7) +
  labs(x = "Number of Bedroom",
       y = "Tenant",
       title = "Number of bedroom preferred by tenant") +
  geom_text(aes(label = Tenant))
```

Figure 68: Source code for analysis 6.2

The above source code is being used to generate a bar chart to visualize the bedroom number of houses preferred by tenants. The `data` is assigned with the value returned from the `table()` function, which tabulates the frequency of bedroom numbers. The `data` is then converted into a data frame and named to plot the pie chart. Kindly refers to the source code explanation in analysis 2.1 for the details of built-in functions in plotting the bar chart.

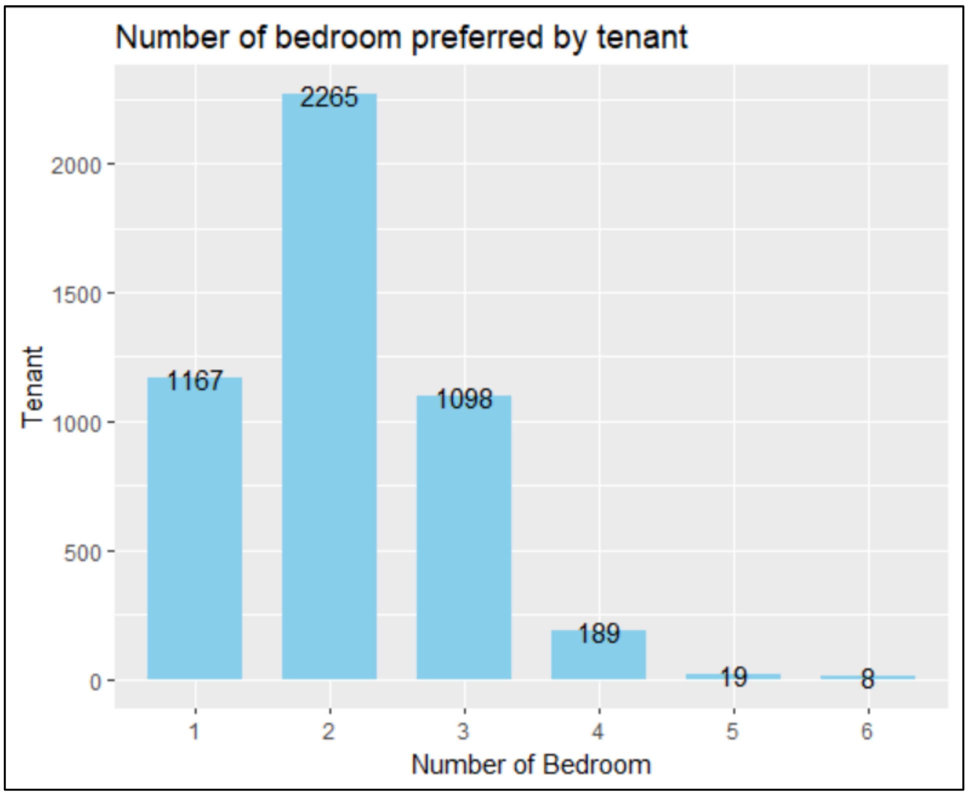


Figure 69: Analysis result 6.2

The bar chart above shows that most tenant prefers houses with 2 bedrooms, with 2,265 of them. There is a total of 1,167, and 1,098 tenants prefer houses with one and 3 bedrooms respectively. There is only a total of 216 tenants who prefer houses with more than 3 bedrooms.

3.6.3 - Analysis 6.3: Find the number of bathrooms preferred by tenants

```
data = as.data.frame(table(house_data$Bathroom))
names(data) = c("Bathroom", "Tenant")

ggplot(data, aes(x = Bathroom, y = Tenant)) +
  geom_bar(stat = "identity", fill="skyblue", width = 0.7) +
  labs(x = "Number of Bathroom",
       y = "Tenant",
       title = "Number of bathroom preferred by tenant") +
  geom_text(aes(label = Tenant))
```

Figure 70: Source code for analysis 6.3

The above source code generates a bar chart to visualize the bathroom number of houses preferred by tenants. Kindly refers to the source code explanation of analysis 6.2, which is highly similar to this analysis.

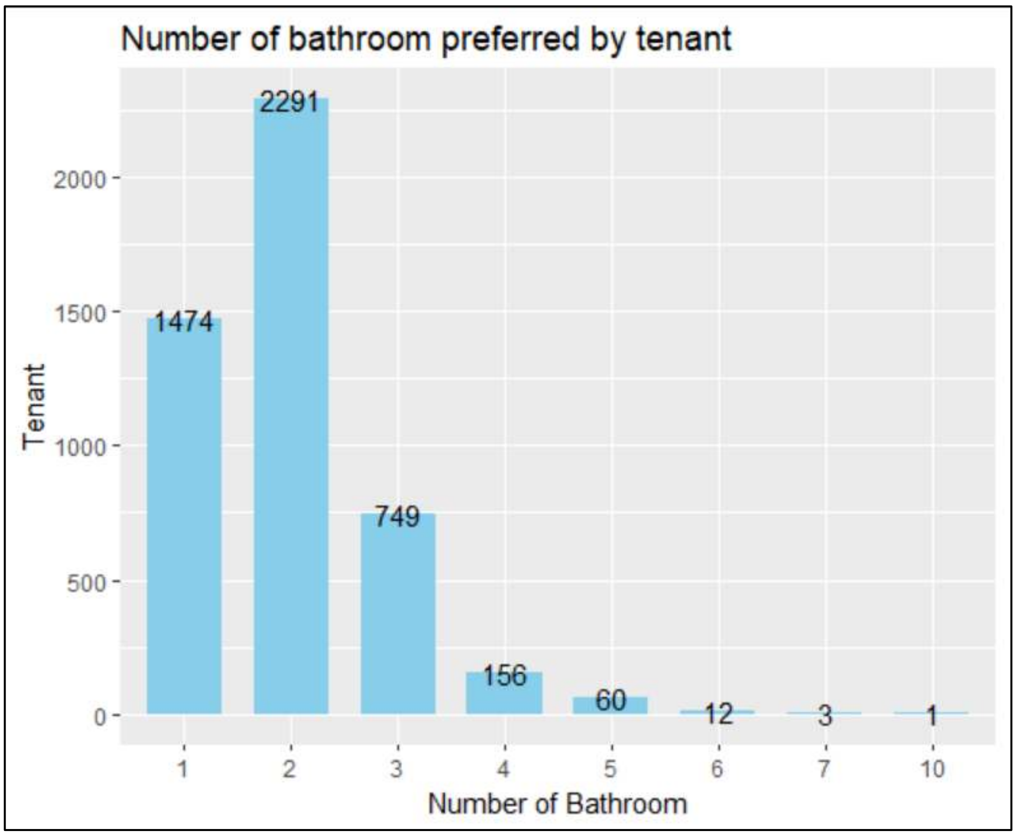
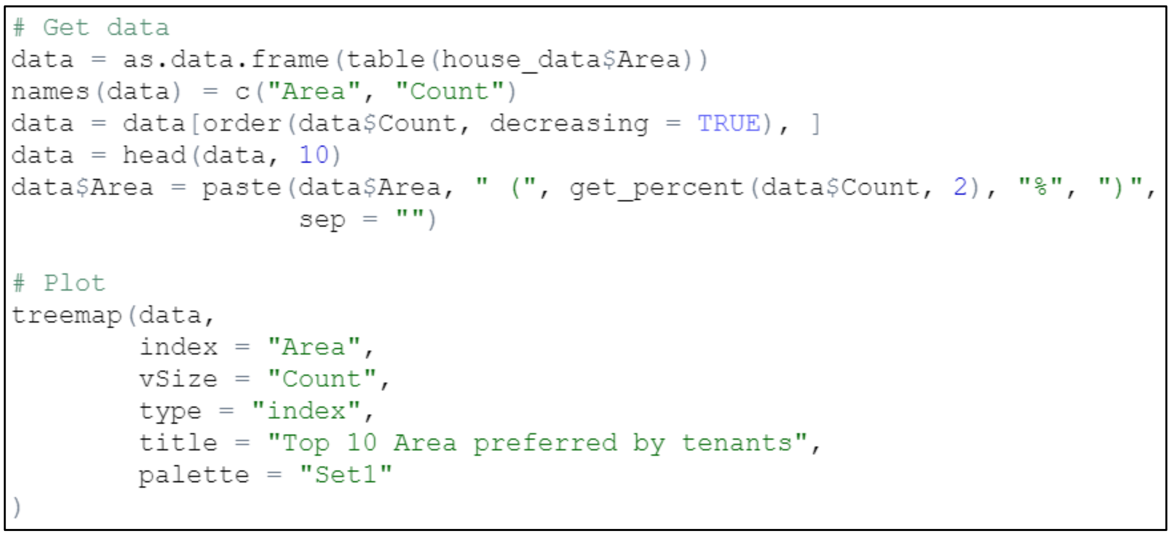


Figure 71: Analysis result 6.3

The bar chart above shows that most tenant prefers houses with 2 bathrooms, with 2,291 of them. There is a total of 1,474, and 749 tenants prefer houses with one and 3 bedrooms respectively. There is only a total of 232 tenants who prefer houses with more than 3 bathrooms.

3.6.4 - Analysis 6.4: Find the area in which the tenant's preferred house is located



```
# Get data
data = as.data.frame(table(house_data$Area))
names(data) = c("Area", "Count")
data = data[order(data$Count, decreasing = TRUE), ]
data = head(data, 10)
data$Area = paste(data$Area, " (", get_percent(data$Count, 2), "%", ")",
                  sep = "")

# Plot
treemap(data,
        index = "Area",
        vSize = "Count",
        type = "index",
        title = "Top 10 Area preferred by tenants",
        palette = "Set1"
)
```

Figure 72: Source code for analysis 6.4

The above source code generates a tree map to visualize the top 10 areas of houses preferred by tenants. The `table()` built-in function is used to perform a tabulation of the area preferred by the tenants. The table of data is then converted into a data frame for renaming using the `names()` function. The data is being reordered using the `order()` function with the parameter `decreasing` of true. The top 10 data is obtained using the `head()` built-in function with the value of 10 for the second parameter. The `Area` column is replaced with itself concatenated with the percentage value that is obtained using the `get_percent` function.

The `treemap()` function from the treemap package is used to plot the treemap where the first parameter is the data to be plotted. The `index` parameter is for the column that provides groups, that is the `Area` column. The `vSize` parameter is for the value of each group, that is the `Count` column. The `type` parameter is used to specify the type of the treemap, which determines how the rectangles are coloured. The provided value for this parameter is "`index`", which means the colours are determined by the index variables, that is the `Area` column.

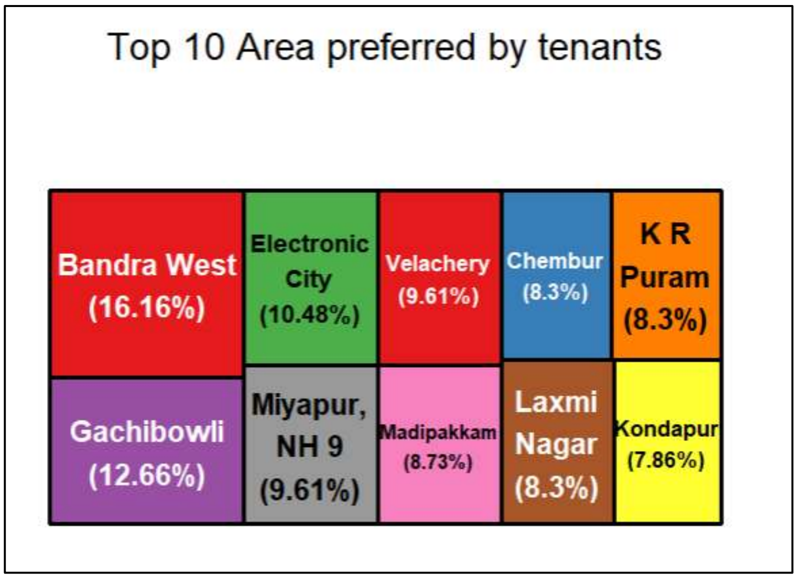
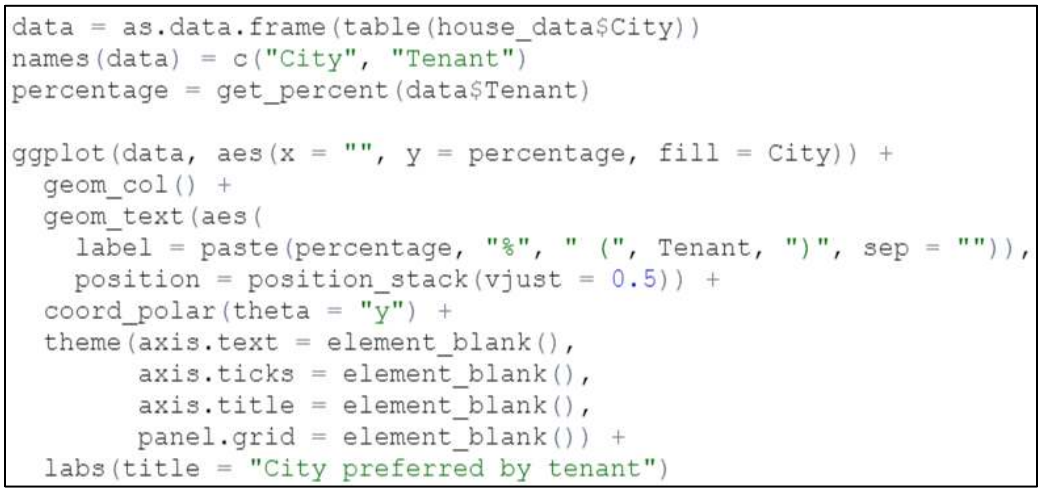


Figure 73: Analysis result 6.4

The above treemap shows that most tenants prefer to rent houses located in the Bandra West area, which is around 16.16% of them. The second most preferred area preferred by tenants is the Gachibowli, which is around 12.66%.

3.6.5 - Analysis 6.5: Find the city in which the tenant's preferred house is located



```
data = as.data.frame(table(house_data$City))
names(data) = c("City", "Tenant")
percentage = get_percent(data$Tenant)

ggplot(data, aes(x = "", y = percentage, fill = City)) +
  geom_col() +
  geom_text(aes(
    label = paste(percentage, "%", " (", Tenant, ")", sep = "")),
    position = position_stack(vjust = 0.5)) +
  coord_polar(theta = "y") +
  theme(axis.text = element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_blank(),
        panel.grid = element_blank()) +
  labs(title = "City preferred by tenant")
```

Figure 74: Source code for analysis 6.5

The above source code generates a pie chart to visualize the percentage of houses renting out in each city. Kindly refers to the source code explanation of analysis 6.2, which has the same technique used to obtain the data to be plotted. The percentages of each city's rental houses are obtained by using the `get_percent()` helper function, which calculates the percentage of data based on its value. `ggplot2` does not offer any specific geom to build pie charts. The trick for this is by building a stacked bar chart using the `geom_col()` function and making it circular with the `coord_polar()` function. The `theta` parameter with the value of `"y"` makes the variable map angle to the y-axis. The `paste()` function is used to concatenate the percentage with the count of renting houses. The `sep` argument has space as its default separator, and therefore it is being overridden using an empty string. The `theme()` function is used to override the default theme elements such as the axis text, ticks title and panel grid.

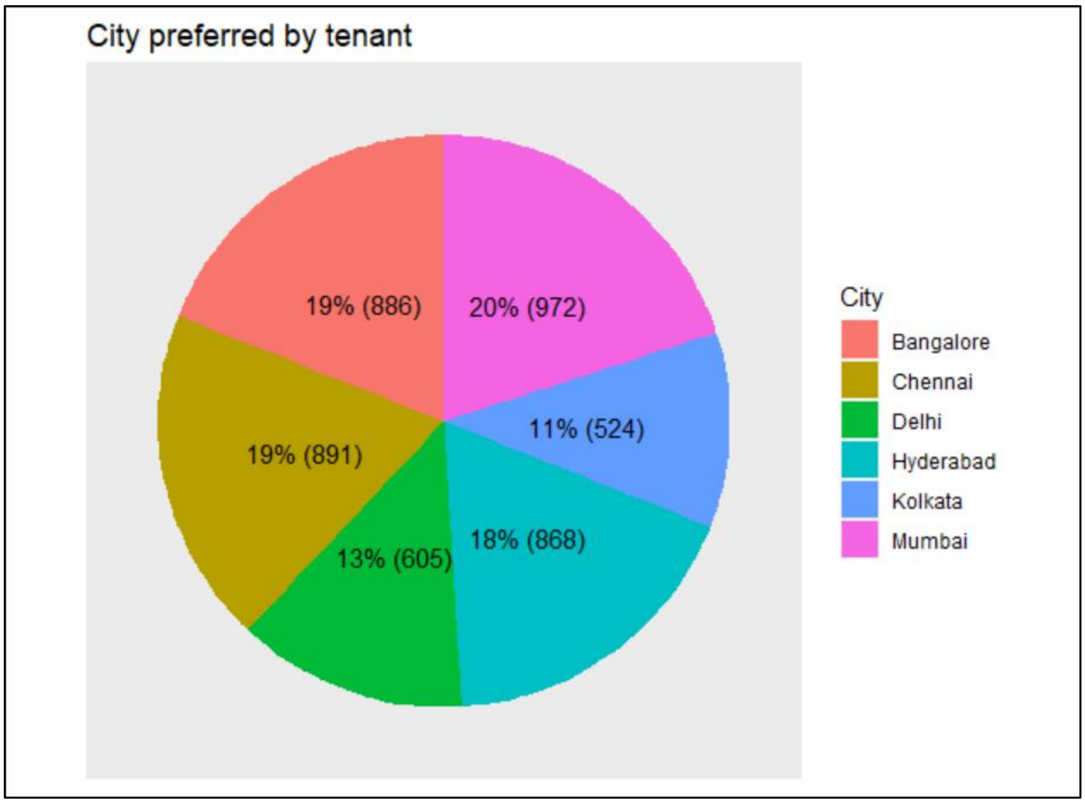
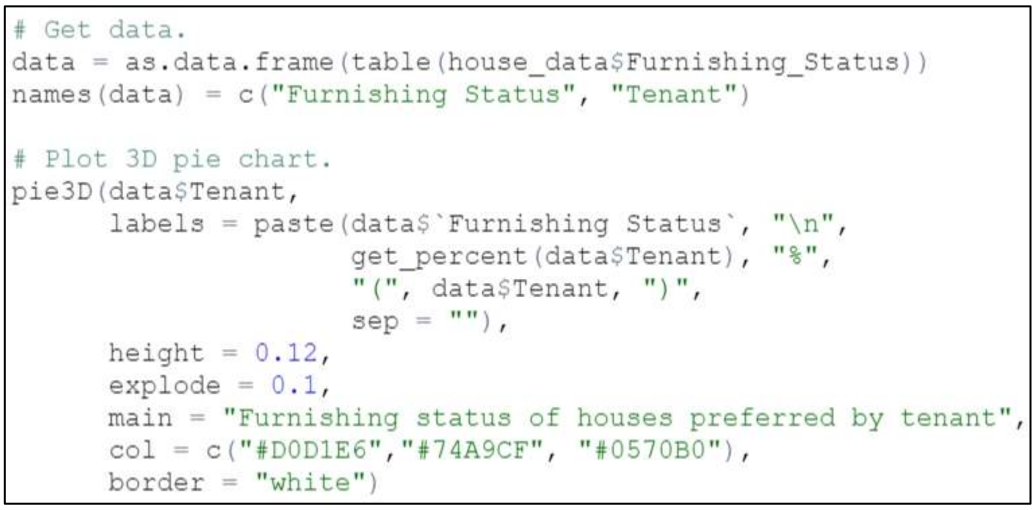


Figure 75: Analysis result 6.5

The above pie chart shows that most of the houses are renting out in Mumbai, that is around 20% or 972 houses. Both Bangalore and Chennai have the second most proportion of houses renting out, that is around 19% or 886 and 891 houses respectively. Hyderabad is also having a relatively close amount of houses renting out, that is around 18% or 868 houses. Delhi and Kolkata both have lesser houses renting out, that is around 13% and 11% or 605 and 868 houses respectively.

3.6.6 - Analysis 6.6: Find the furnishing status of houses preferred by tenants



```
# Get data.
data = as.data.frame(table(house_data$Furnishing_Status))
names(data) = c("Furnishing Status", "Tenant")

# Plot 3D pie chart.
pie3D(data$Tenant,
      labels = paste(data$`Furnishing Status`, "\n",
                     get_percent(data$Tenant), "%",
                     "(", data$Tenant, ")",
                     sep = ""),
      height = 0.12,
      explode = 0.1,
      main = "Furnishing status of houses preferred by tenant",
      col = c("#D0D1E6","#74A9CF", "#0570B0"),
      border = "white")
```

Figure 76: Source code for analysis 6.6

The above source code is being used to generate a 3D pie chart to visualize the furnishing status of houses preferred by tenants. Kindly refers to the source code explanation of analysis 6.2, which has the same technique used to obtain the data to be plotted. The `pie3D` function from the plotrix package is used to plot the 3D pie chart. The first parameters hold the data to be plotted and the `labels` parameter is used to specify the text on each slice of the pie chart, that is the furnishing status along with the percentage and tenants count. The percentage of each furnishing status is obtained using the `get_percent()` function. The `height` parameter is used to specify the height of the 3D pie chart while the `explode` parameter is used to separate the slices from the pie chart. The `col` parameter is used to specify the colours of slices in the pie chart whereas the `border` parameter is used to specify the colour of the slices' border.

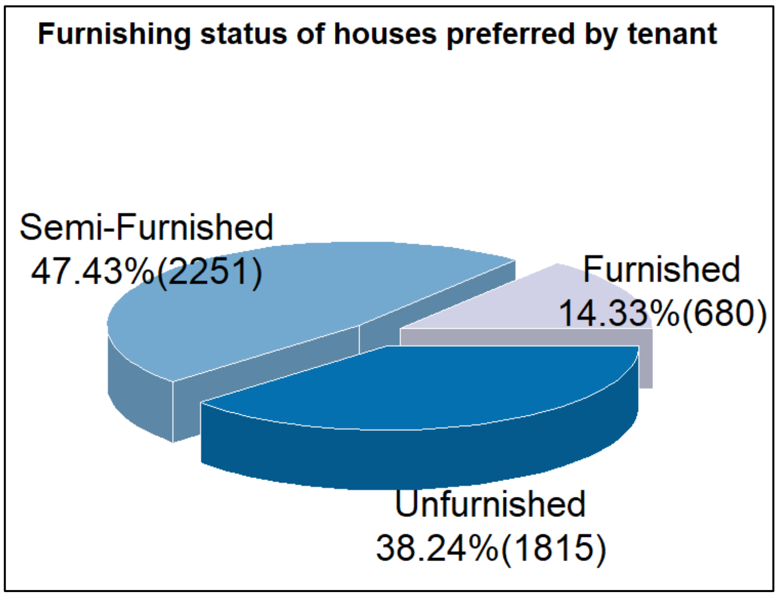
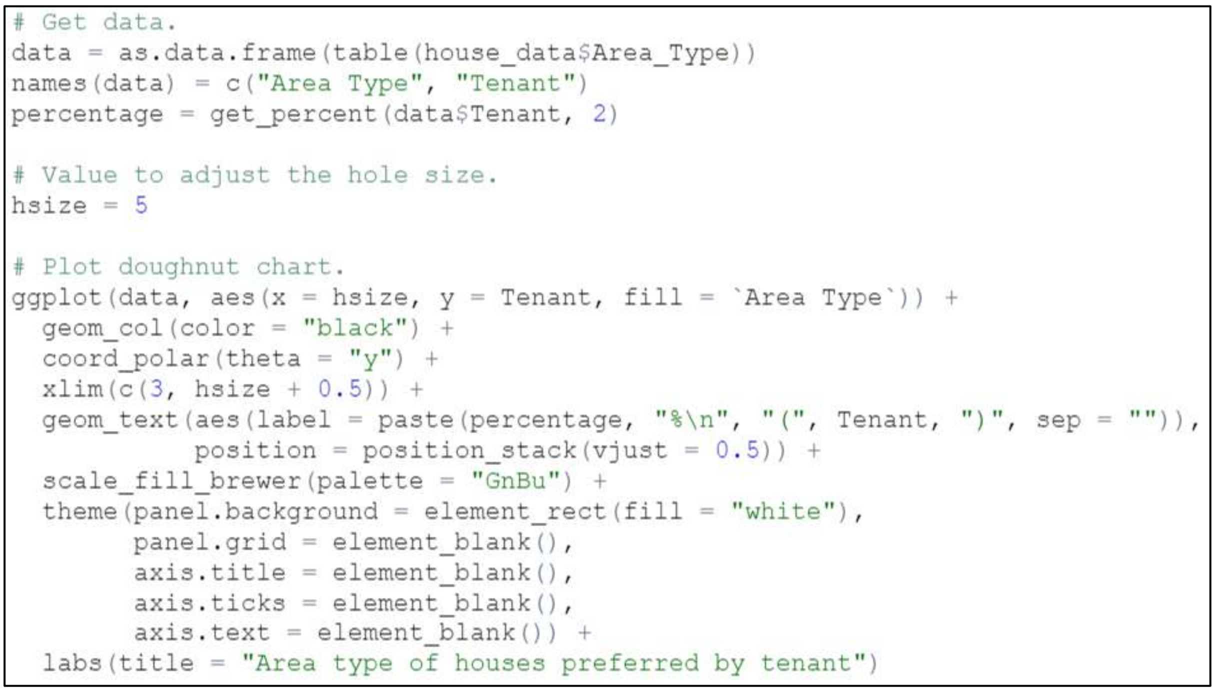


Figure 77: Analysis result 6.6

The 3D pie chart above shows that most tenant prefers houses that is semi-furnished, which is around 47.43% or 2,251 of them. There are around 38.24% or 1,815 tenants prefer houses that are unfurnished, and 14.33% or 680 tenants prefer fully-furnished houses.

3.6.7 - Analysis 6.7: Find the area type of houses preferred by tenants



```r
# Get data.
data = as.data.frame(table(house_data$Area_Type))
names(data) = c("Area Type", "Tenant")
percentage = get_percent(data$Tenant, 2)

# Value to adjust the hole size.
hsize = 5

# Plot doughnut chart.
ggplot(data, aes(x = hsize, y = Tenant, fill = `Area Type`)) +
  geom_col(color = "black") +
  coord_polar(theta = "y") +
  xlim(c(3, hsize + 0.5)) +
  geom_text(aes(label = paste(percentage, "%\n", "(", Tenant, ")", sep = "")),
            position = position_stack(vjust = 0.5)) +
  scale_fill_brewer(palette = "GnBu") +
  theme(panel.background = element_rect(fill = "white"),
        panel.grid = element_blank(),
        axis.title = element_blank(),
        axis.ticks = element_blank(),
        axis.text = element_blank()) +
  labs(title = "Area type of houses preferred by tenant")
```

Figure 78: Source code for analysis 6.7

The above source code generates a doughnut chart to visualize the area type of houses preferred by tenants. Kindly refers to the source code explanation of analysis 6.2, which has the same technique used to obtain the data to be plotted. The percentage of each area type is obtained using the `get_percent()` function. The `hsize` variable is used to hold the value that specifies the hole size of the doughnut chart. There is no specific function has been created for plotting a doughnut chart in `ggplot2`. The trick for this is by building a stacked bar chart using the `geom_col()` function and making it circular with the `coord_polar()` function. The `theta` parameter with the value of "`y`" makes the variable map angle to the y-axis. Finally, the `xlim()` function switches the pie to a doughnut by adding the empty circle in the middle. Besides, the `scale_fill_brewer()` function is used to change the filling colours of ggplot2 graphs. The `theme()` function is used to override the default theme elements such as the axis text, ticks title and panel grid.

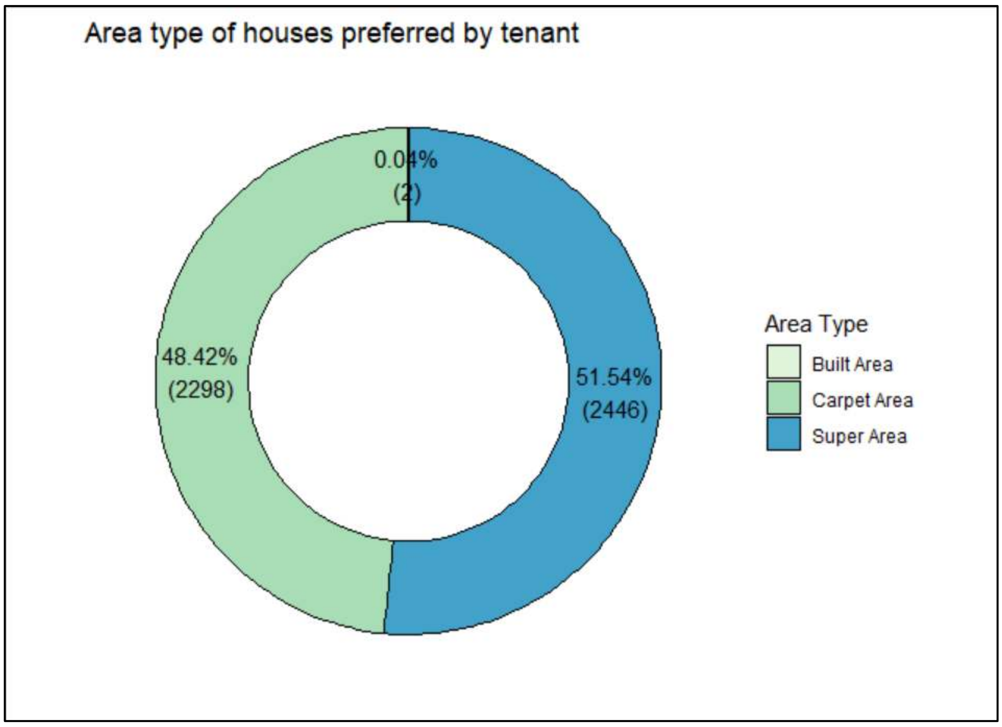


Figure 79: Analysis result 6.7

The above doughnut chart shows that more than half of the tenants prefer houses in which the size is calculated based on super area, with 51.54% or 2,446 of them. There are 48.42% or 2,298 tenants who prefer houses with the area type of carpet area and only 0.04% or 2 tenants prefer houses with the built area.

3.6.8 - Analysis 6.8: Find the contact way of houses preferred by tenants

```
data = as.data.frame(table(house_data$Contact_Way))
names(data) = c("Contact Way", "Tenant")

ggplot(data, aes(x = `Contact Way`, y = Tenant)) +
  geom_bar(stat = "identity", fill="skyblue", width = 0.7) +
  labs(x = "Contact Way",
       y = "Tenant",
       title = "Contact Way of houses preferred by tenant") +
  geom_text(aes(label = Tenant))
```

Figure 80: Source code for analysis 6.8

The above source code generates a bar chart to visualize the contact way of houses preferred by tenants. Kindly refers to the source code explanation of analysis 6.2, which is highly similar to this analysis.

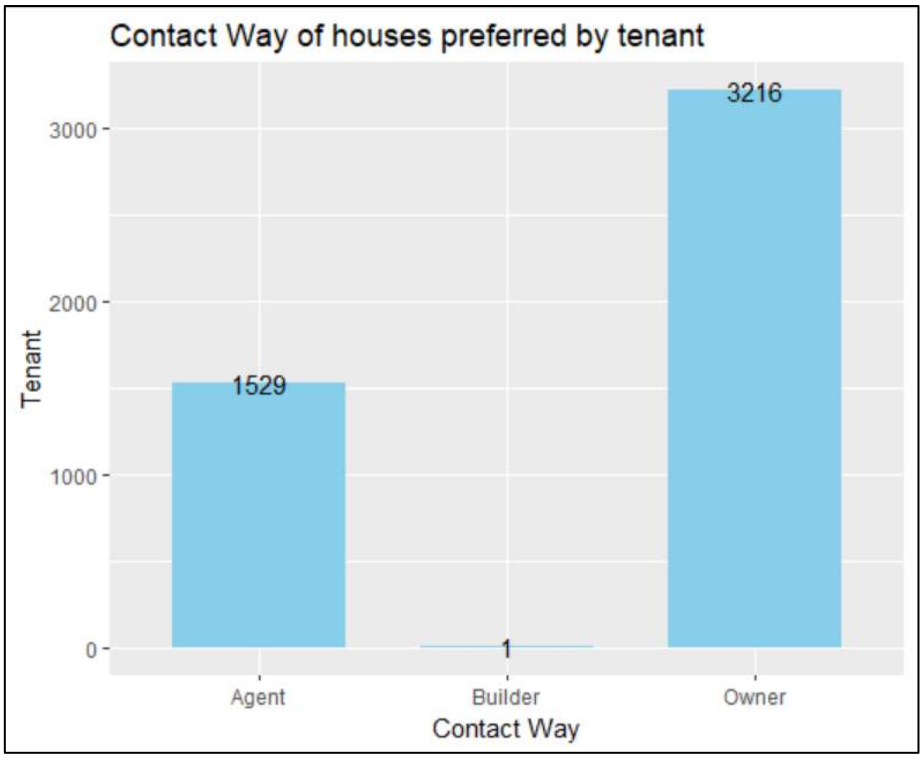


Figure 81: Analysis result 6.8

The bar chart above shows that most tenant prefers houses in which the contact way is the owner, with 3,216 of them. There are 1,529 tenants who prefer houses that require them to contact the agent and there is only one tenant who prefers houses with the contact way of the builder.

3.6.9 - Conclusion for question 6

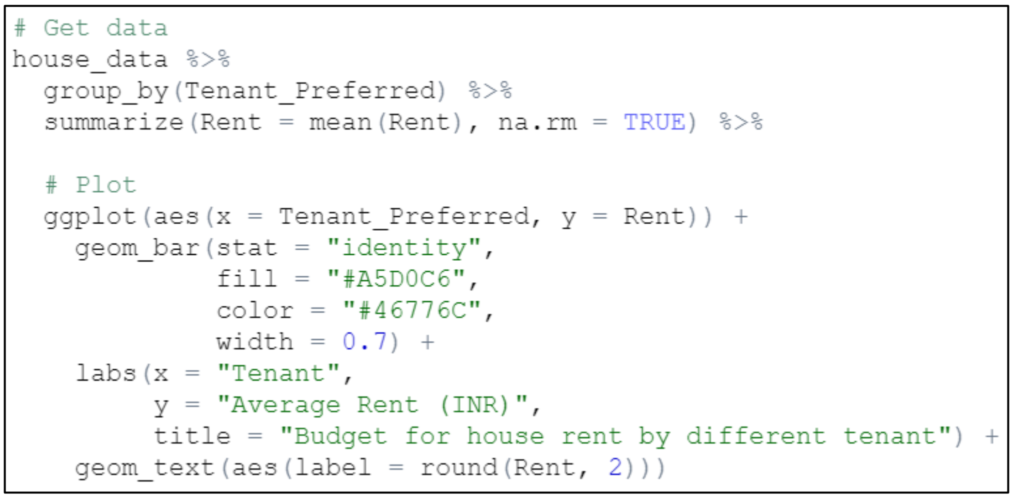
There are a total of 8 analyses being done for question 6 to find out what kind of house tenants prefer to live in. In summary, most tenants would prefer to rent houses that are:

- around 1600 sqft
- having 2 bedrooms and 2 bathrooms
- located in Bandra West, Mumbai
- semi-furnished
- calculated based on super area
- having the contact way of owners

3.7 - Question 7: How do different groups of tenants choose their rental houses?

Bracket subsetting can be cumbersome and difficult to read, especially for analyses such as 5.3, which involve complicated operations. As such, the subsequent question may involve various data manipulation techniques for modifying data to make the code more readable and organized (Naveen, 2022).

3.7.1 - Analysis 7.1: Find the budget for house rent by different tenants



```
# Get data
house_data %>%
  group_by(Tenant_Preferred) %>%
  summarize(Rent = mean(Rent), na.rm = TRUE) %>%

  # Plot
  ggplot(aes(x = Tenant_Preferred, y = Rent)) +
    geom_bar(stat = "identity",
             fill = "#A5D0C6",
             color = "#46776C",
             width = 0.7) +
    labs(x = "Tenant",
         y = "Average Rent (INR)",
         title = "Budget for house rent by different tenant") +
    geom_text(aes(label = round(Rent, 2)))
```

Figure 82: Source code for analysis 7.1

The above source code generates a bar chart to visualize the budget for house rent by different tenants. The pipe operator from `dplyr` package, written as `>%>` is used to take the output of one function and passes it into another function as an argument, allowing the sequence of analysis steps to be linked (Willeams, 2017). The `group_by()` function is used to group the data based on the `Tenant_Preferred` column in the `house_data` dataset. The `group_by()` function alone does not give any output unless it is used with the `summarize()` function, which is used to get aggregation results on specified columns, that is the mean value of rent. Thus, the `group by()` function is used to group the mean value of rent based on the data in `Tenant_Preferred` column. The `na.rm` argument is set as true to exclude missing value when calculating the mean value of rent. The `color` argument in the `geom_bar` function is used to specify the border colour of the bar. Kindly refers to the source code explanation in analysis 2.1 for more details of built-in functions in plotting the bar chart.

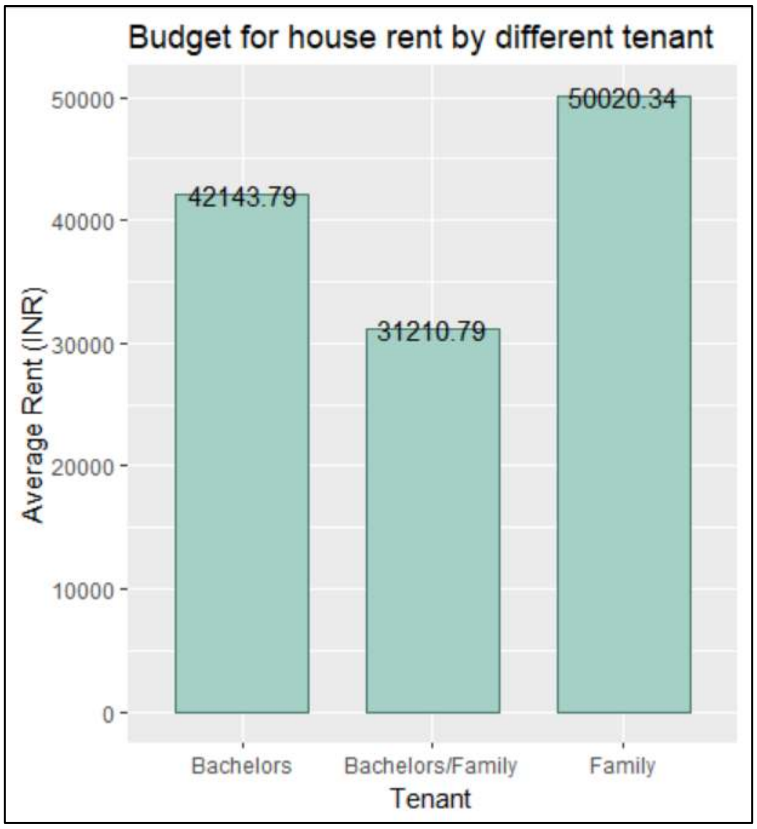
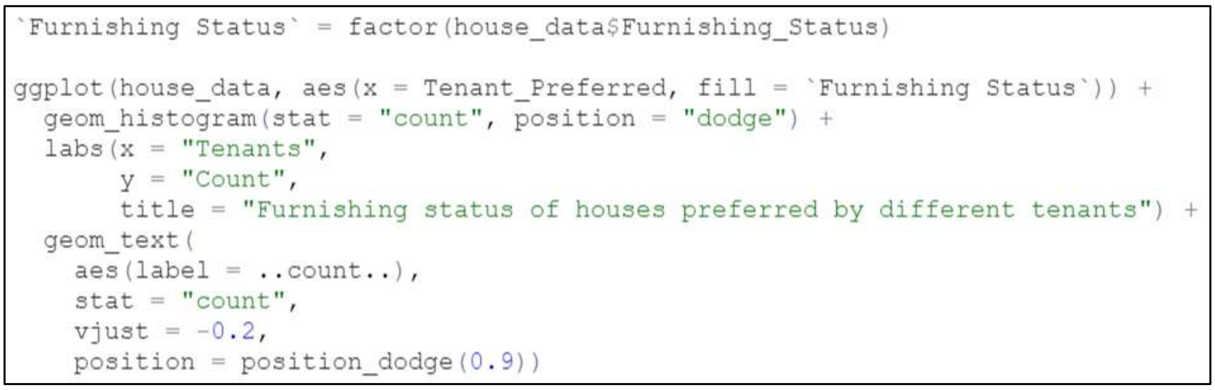


Figure 83: Analysis result 7.1

The above bar chart shows that the bachelor is having an average of 42,143.79 INR as their budget for renting a house. As for tenants who is either bachelor or family, their average budget for house rent is 31,210.79 INR. The family people will have a higher average budget for house rent, which is 50,020.34 INR.

3.7.2 - Analysis 7.2: Find the furnishing status of houses preferred by different tenants



```
`Furnishing Status` = factor(house_data$Furnishing_Status)

ggplot(house_data, aes(x = Tenant_Preferred, fill = `Furnishing Status`)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "Tenants",
       y = "Count",
       title = "Furnishing status of houses preferred by different tenants") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9))
```

Figure 84: Source code for analysis 7.2

The above source code is to generate a histogram that can be used to visualize the furnishing status of houses preferred by different tenants. Kindly refers to the source code explanation of plotting a histogram in analysis 1.2.

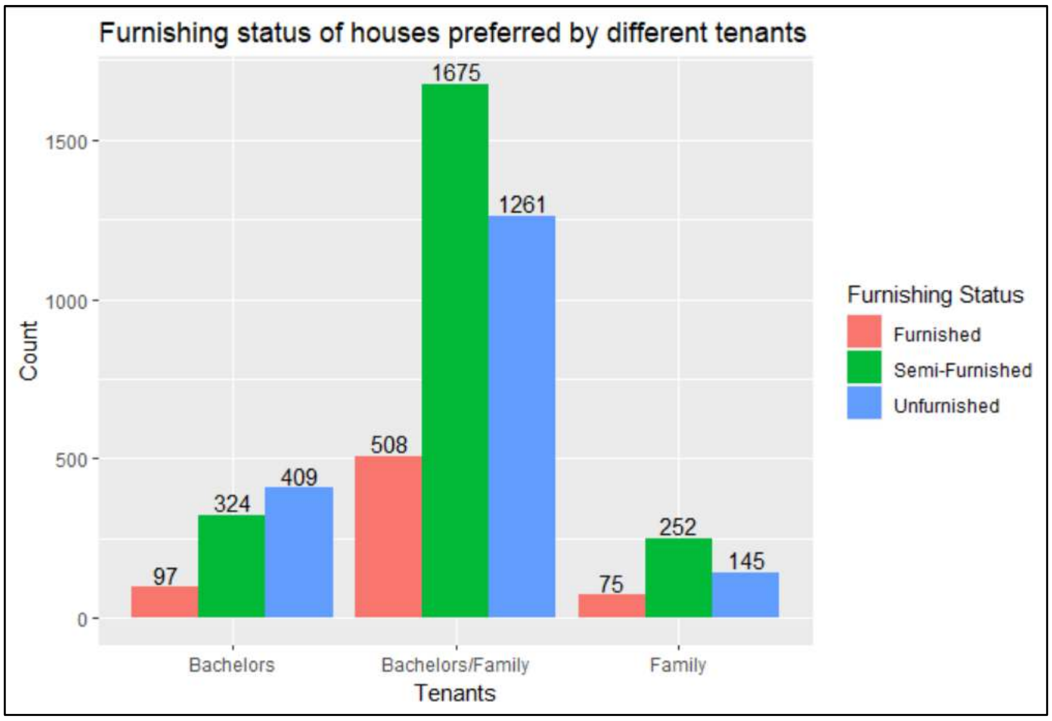
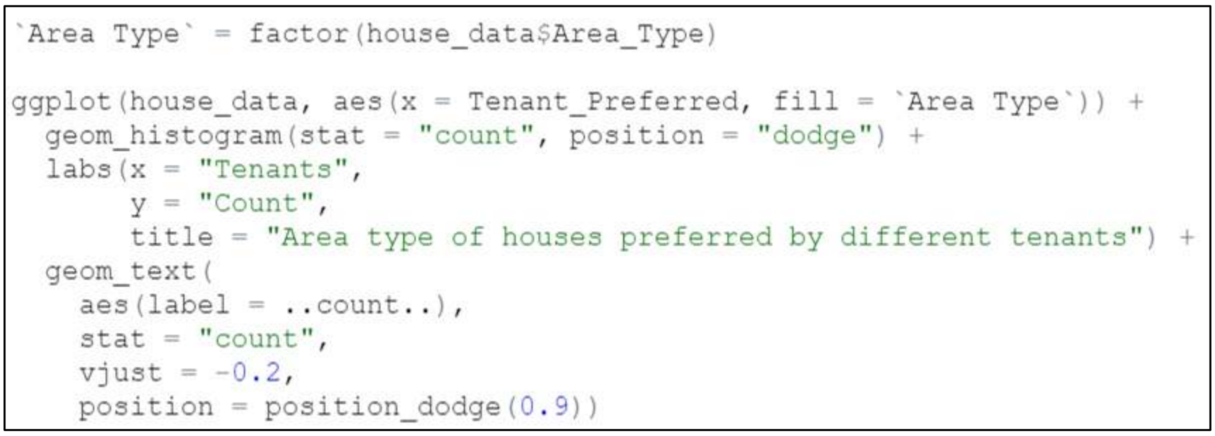


Figure 85: Analysis result 7.2

The above histogram shows that most bachelors choose to rent unfurnished houses, with 409 of them. As for tenants who is either bachelor or family, most of them prefer to choose semi-furnished houses, that is 1,675 of them. Most family people would also prefer semi-furnished houses, with 252 of them.

3.7.3 - Analysis 7.3: Find the area type of houses preferred by different tenants



```
`Area Type` = factor(house_data$Area_Type)

ggplot(house_data, aes(x = Tenant_Preferred, fill = `Area Type`)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "Tenants",
       y = "Count",
       title = "Area type of houses preferred by different tenants") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9))
```

Figure 86: Source code for analysis 7.3

The above source code is to generate a histogram that can be used to visualize the area type of houses preferred by different tenants. Kindly refers to the source code explanation of plotting a histogram in analysis 1.2.

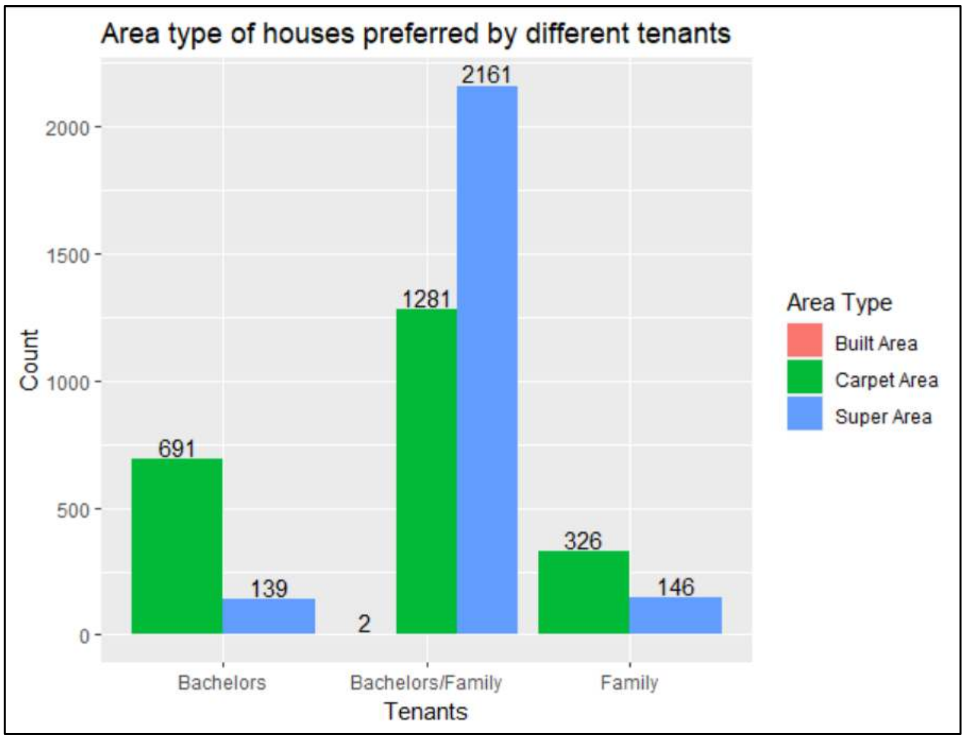
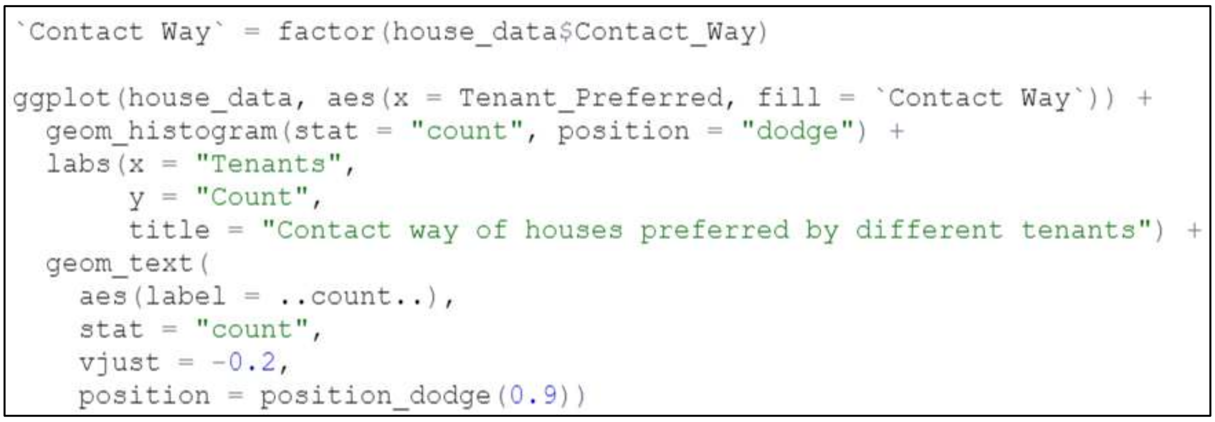


Figure 87: Analysis result 7.3

The above histogram shows that most bachelors choose to rent houses in which the size is calculated based on carpet area, that is 691 of them. As for tenants who is either bachelor or family, most of them prefer houses calculated based on super area, that is 2,161 of them. Most family people would prefer houses calculated on carpet area, with 326 of them.

3.7.4 - Analysis 7.4: Find the contact way of houses preferred by different tenants



```
`Contact Way` = factor(house_data$Contact_Way)

ggplot(house_data, aes(x = Tenant_Preferred, fill = `Contact Way`)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "Tenants",
       y = "Count",
       title = "Contact way of houses preferred by different tenants") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9))
```

Figure 88: Source code for analysis 7.4

The above source code is to generate a histogram that can be used to visualize the contact way of houses preferred by different tenants. Kindly refers to the source code explanation of plotting a histogram in analysis 1.2.

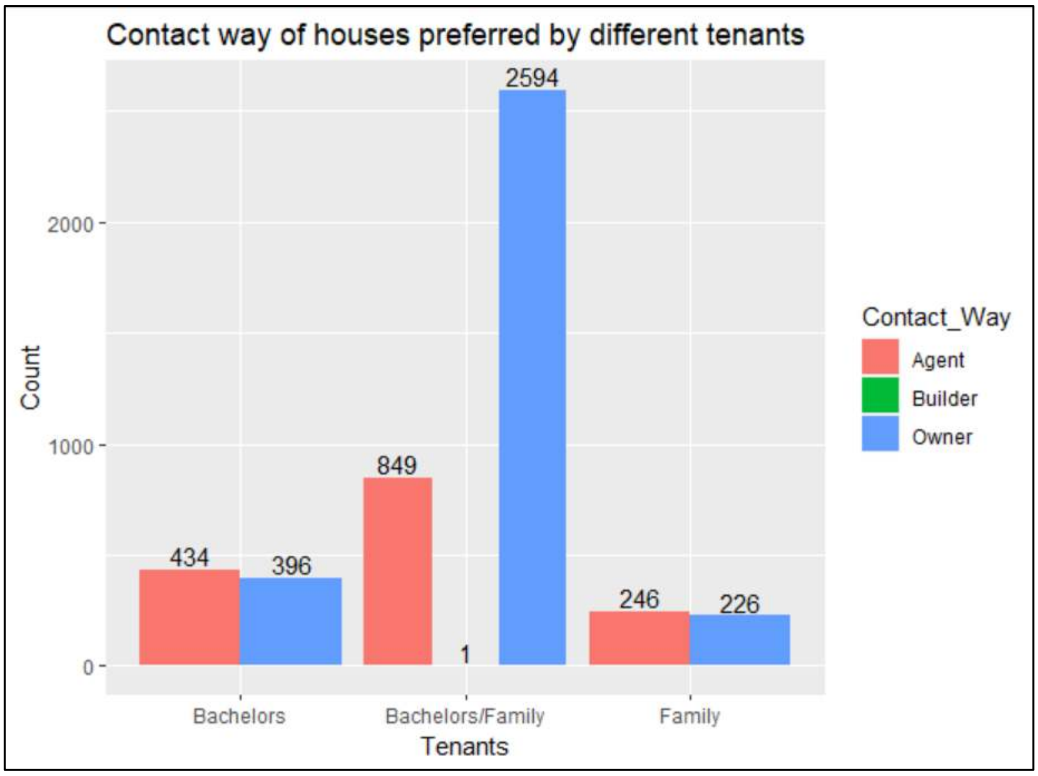


Figure 89: Analysis result 7.4

The above histogram shows that most bachelors and families prefer to contact the agent when choosing a rental house, that is 434 and 246 of them respectively. As for tenants who is either bachelor or family, most of them choose to rent a house with the contact way of the owner, that is 2,594 of them.

3.7.5 - Analysis 7.5: Find the city preferred by different tenants

```
ggplot(house_data, aes(x = Tenant_Preferred, fill = City)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "Tenants",
       y = "Count",
       title = "City preferred by different tenants") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9))
```

Figure 90: Source code for analysis 7.5

The above source code is to generate a histogram that can be used to visualize the city preferred by different tenants. Kindly refers to the source code explanation of plotting a histogram in analysis 1.2.

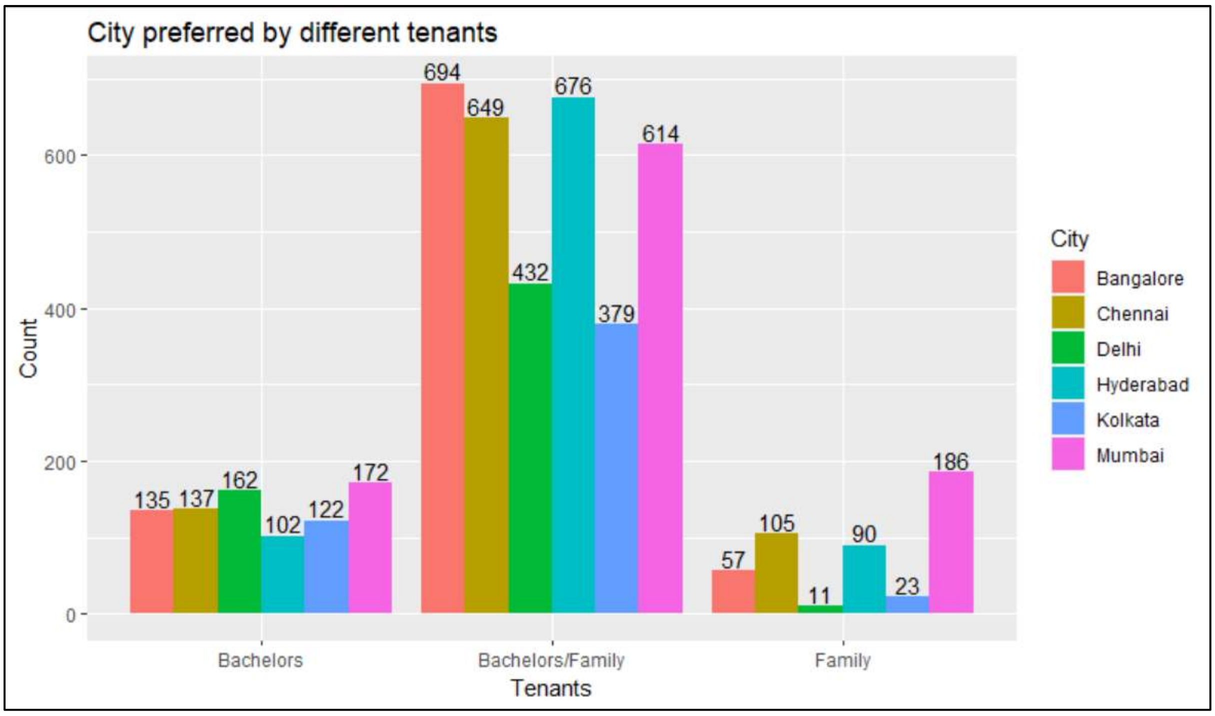


Figure 91: Analysis result 7.5

The above histogram shows that most bachelors and families prefer to rent a house located in Mumbai, that is 172 and 186 of them respectively. As for tenants who is either bachelor or family, most of them choose to rent a house located in Bangalore, that is 694 of them.

3.7.6 - Analysis 7.6: Find the area preferred by different tenants

```
data = house_data %>%
        {table(.$Area, .$Tenant_Preferred)} %>%
        data.frame() %>%
        set_colnames(c("Area", "Tenant", "Count")) %>%
        arrange(desc(Count)) %>%
        group_by(Tenant) %>%
        slice(1:3)

# Plot
data %>%
  ggplot(aes(fill = Area, x = Tenant, y = Count)) +
    geom_histogram(stat = "identity") +
    labs(x = "Tenant",
         y = "Count",
         title = "Top 3 area preferred by different tenants") +
    geom_text(
      aes(label = Count,
          group = Area),
      position = position_stack(vjust = 0.5))
```

Figure 92: Source code for analysis 7.6

The above source code generates a bar chart to visualize the top 3 areas preferred by different tenants. The `table()` function is also used to tabulate the frequency of data based on the `Area` and the `Tenant_Preferred` columns. The table is then converted to a data frame as well as named using the `set_colnames()` function. The data is arranged in descending order based on the `Count` value using the `arrange()` and `desc()` functions together. Areas with the top three frequencies will be obtained using the `group_by()` and `slice()` functions. Kindly refers to the source code explanation of plotting a histogram in analysis 1.2.

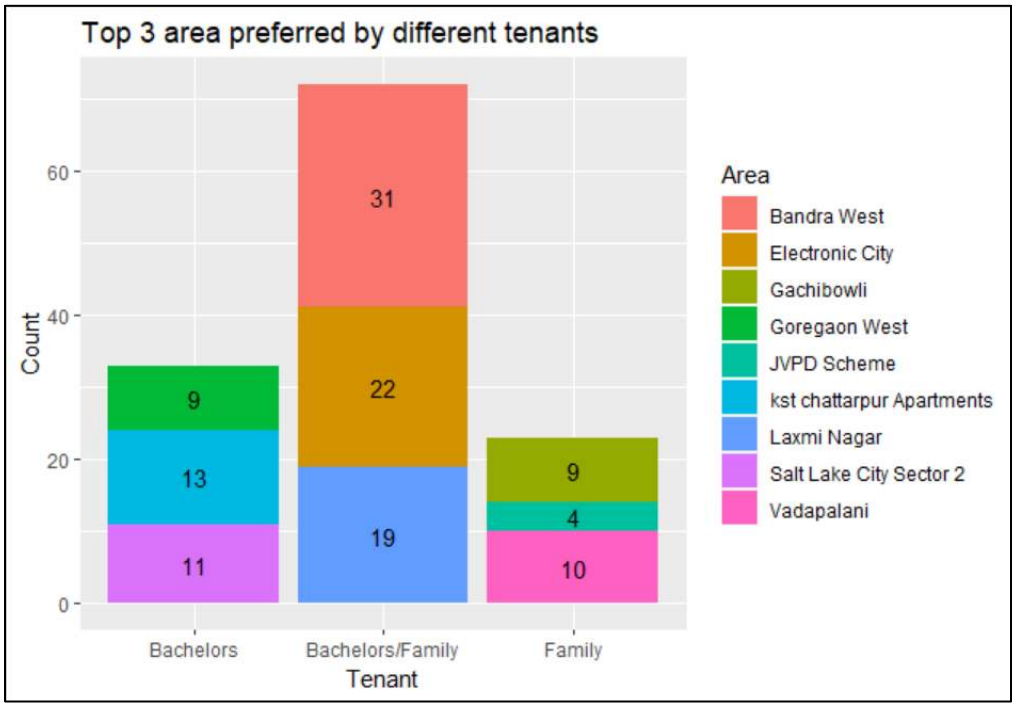


Figure 93: Analysis result 7.6

The above histogram shows that most bachelors would prefer to rent houses located at the kst chattarpur apartments, that is 13 of them. As for tenants who are either bachelors or family, most of them prefer to rent houses at the Bandra West, that is 31 of them. Most families would prefer to rent houses located in the Vadapalani area.

3.7.7 - Analysis 7.7: Find the floor preferred by different tenants

```r
# Extract the floor number.
floors = c()
for (floor in house_data$Floor) {
  floors = c(floors, trimws(strsplit(floor, split = "out of")[[1]][1]))
}

# Get top 3 floors preferred by different tenants.
data = house_data %>%
        mutate(Floor = floors) %>%
        {table(.$Floor, .$Tenant_Preferred)} %>%
        data.frame() %>%
        set_colnames(c("Floor", "Tenant", "Count")) %>%
        arrange(desc(Count)) %>%
        group_by(Tenant) %>%
        slice(1:3) %>%
        select(c("Tenant", "Floor", "Count"))
```

Figure 94: Source code for analysis 7.7 (Part 1)

The above source code generates a stacked bar chart to visualize the top three floors preferred by different tenants. The floor is extracted using several functions in a for loop and stored in the `floors` vector. Kindly refers to this section, which has the same technique for deriving the floor data. The `mutate()` function is being used to replace the `Floor` column in the `house_data` data frame with the extracted floor data. Kindly refers to the source code explanation in analysis 7.6 for details of built-in functions used in obtaining data. Lastly, the `select()` function is used to rearrange the column.

```r
# Rearrange the floor order.
data$Floor = factor(data$Floor, levels = c('2', '1', 'Ground'))

# Plot
ggplot(data, aes(fill = Floor, x = Tenant, y = Count)) +
  geom_histogram(stat = "identity") +
  labs(x = "Tenant",
       y = "Count",
       title = "Top 3 floors preferred by different tenants") +
  geom_text(
    aes(label = Count,
        group = Floor),
    position = position_stack(vjust = 0.5))
```

Figure 95: Source code for analysis 7.7 (Part 2)

The `factor()` function is used to rearrange the order of floors by specifying the factor levels for the stacked bar chart. Kindly refers to the source code explanation of plotting a histogram in analysis 1.2.

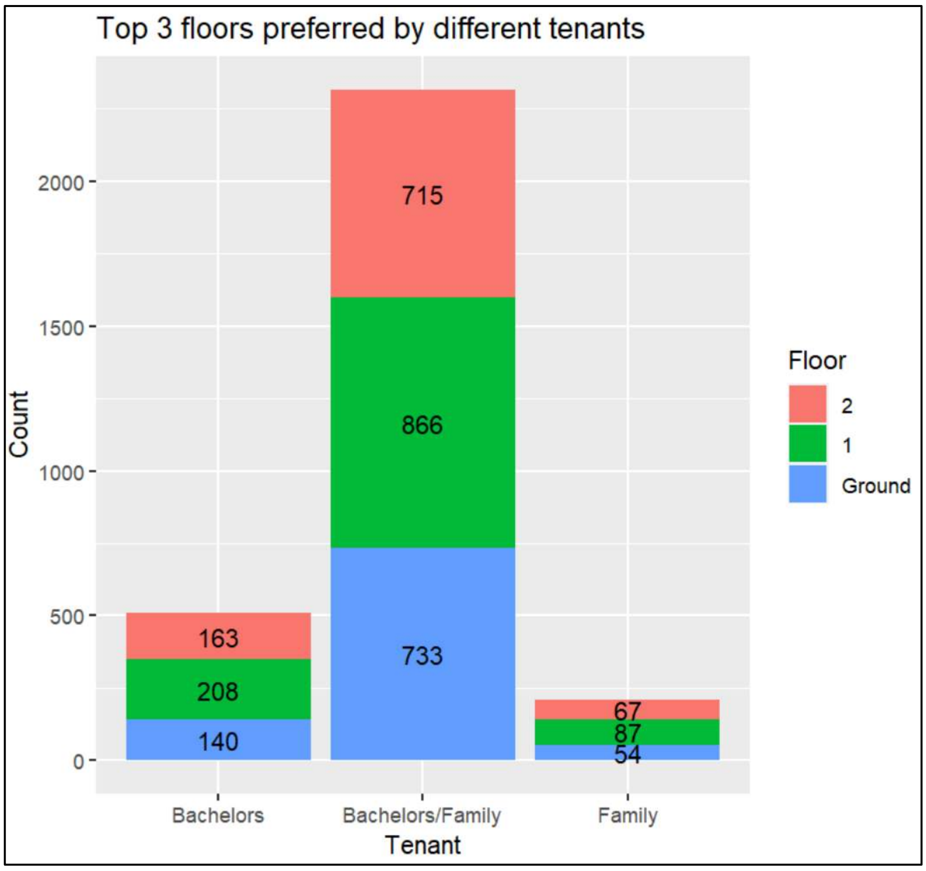


Figure 96: Analysis result 7.7

Each stacked bar chart above can be perceived as a building where each floor is separated using different colours. The result shows that all different tenants prefer to rent houses located on the first floor, following up with the second floor and the ground floor being the third preferred level.

3.7.8 - Analysis 7.8: Find the date of renting houses preferred by different tenants

```
# Get data.
data = house_data %>%
  {table(.$Posted_Date, .$Tenant_Preferred)} %>%
  data.frame() %>%
  set_colnames(c("Date", "Tenant", "Count")) %>%
  mutate(Date = as.Date(Date)) %>%
  filter(Count > 0)

# Plot
data %>%
  ggplot(aes(x = Date, y = Count, group = Tenant, color = Tenant)) +
  geom_line(size = 1) +
  theme_ipsum() +
  labs(x = "Date",
       y = "Count",
       title = "Date of renting house preferred by different tenants")
```

Figure 97: Source code for analysis 7.8

The above source code generates a multiline graph to visualize the date of renting houses preferred by different tenants. The `table()` function is also used to tabulate the frequency of data based on the `Posted_Date` and the `Tenant_Preferred` columns. The table is then converted to a data frame as well as named using the `set_colnames()` function. The `mutate()` function is being used to replace the `Date` column in the `house_data` data frame with the same column in which the data type is converted to date objects. Lastly, the `filter()` function is used to subset the data frame based on the condition where only the value of the `Count` column greater than 0 will be kept and vice versa.

The plotting of the multiline graph is very similar to the line graph in analysis 5.2. The only difference is that the `group` and `color` parameters are used to group the categorical data, that is the `Tenant`. Moreover, the `size` parameter in `geom_line` is used to specify the thickness of the line. The `theme_ipsum()` from the hrbrthemes package is used to provide the theme colour and appearance for the graph.

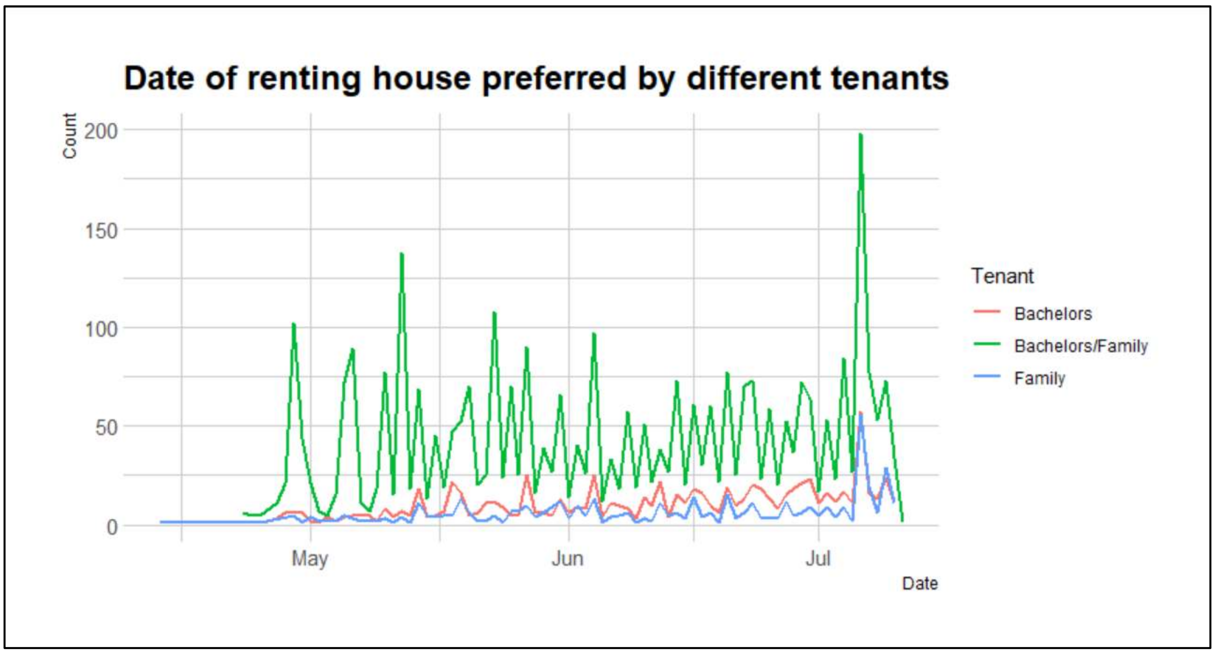


Figure 98: Analysis result 7.8

The above result shows that most of the different tenants would prefer to rent houses posted lately, that is during July. Tenants who are either bachelors or families also prefer to rent houses starting from the end of April. As for the bachelor and family, they prefer to rent houses posted starting from the mid of May.

3.7.9 - Conclusion for question 7

There are a total of 8 analyses being done for question 7 to find out how different groups of tenants choose their rental houses. The table below summarizes all the attributes of houses preferred by different tenants.

Table 3: Attributes of houses preferred by different tenants

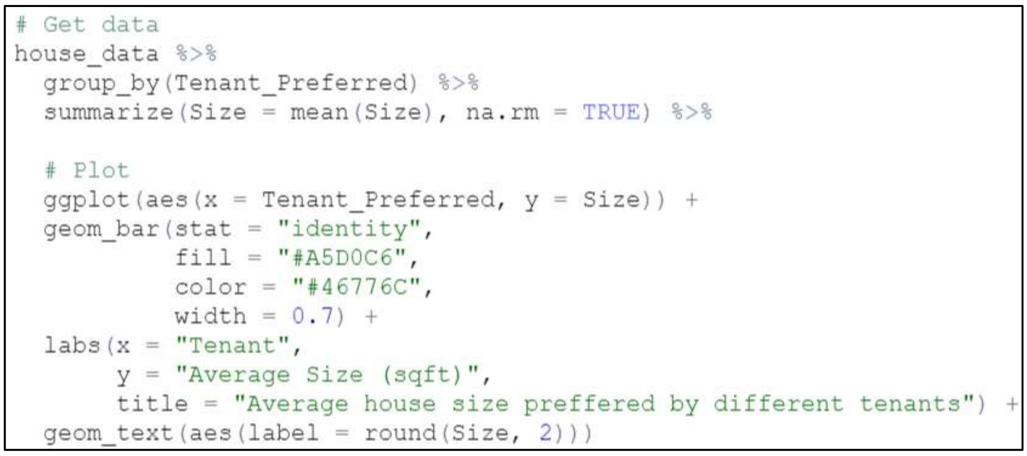
Tenants/Preference	Budget	Furnishing status	Area type	Contact way	City	Floor	Date
Bachelors	Low	Unfurnished	Carpet area	Agent	Mumbai	1	May - July
Bachelors/ Family	Medium	Semi-furnished	Super area	Owner	Bangalore	1	April - July
Family	High	Semi-furnished	Carpet area	Agent	Mumbai	1	May - July

Analysis 7.1 shows that different groups of tenants will have different budgets when choosing their rental houses. Bachelors would prefer to rent an unfurnished house while the other tenants prefer a semi-furnished house. Bachelors and family tenants prefer to choose rental houses in which the size is calculated based on carpet area, having the contact way of the agent, and located in Mumbai. In analysis 7.8, the result shows that all different tenants would prefer to live on the first floor of the building. It is also found that bachelors and families prefer to rent houses posted from May until July whereas tenants who are either bachelors or families would prefer to rent houses posted starting from May until July.

3.8 - Question 8: How big a house do different group of tenants prefer?

This question aims to find how different group of tenants choose their rental house in terms of house sizes.

3.8.1 - Analysis 8.1: Find the house size preferred by different tenants



```
# Get data
house_data %>%
  group_by(Tenant_Preferred) %>%
  summarize(Size = mean(Size), na.rm = TRUE) %>%

  # Plot
  ggplot(aes(x = Tenant_Preferred, y = Size)) +
  geom_bar(stat = "identity",
           fill = "#A5D0C6",
           color = "#46776C",
           width = 0.7) +
  labs(x = "Tenant",
       y = "Average Size (sqft)",
       title = "Average house size preffered by different tenants") +
  geom_text(aes(label = round(Size, 2)))
```

Figure 99: Source code for analysis 8.1

The above source code generates a bar chart to visualize the average house size preferred by different tenants. Kindly refers to analysis 7.1, which has similar analysis techniques.

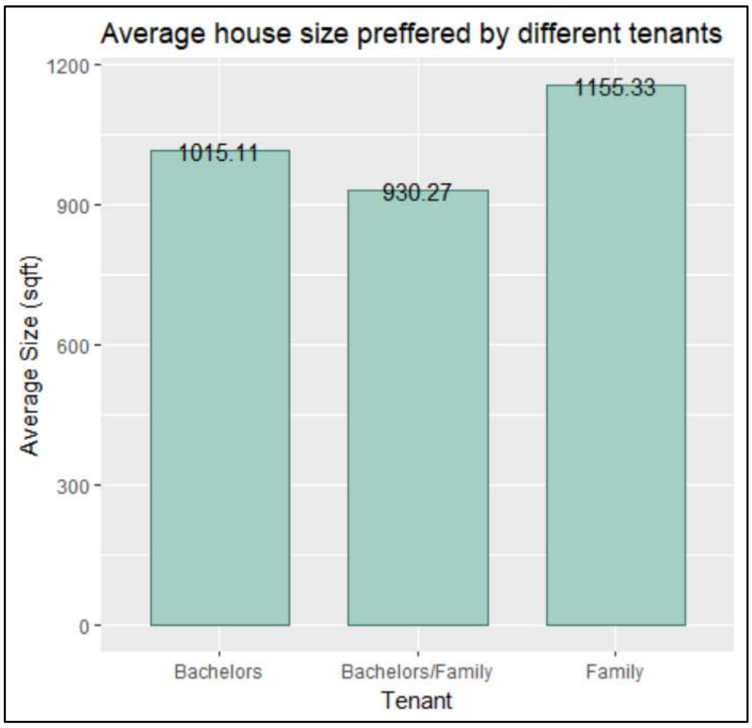
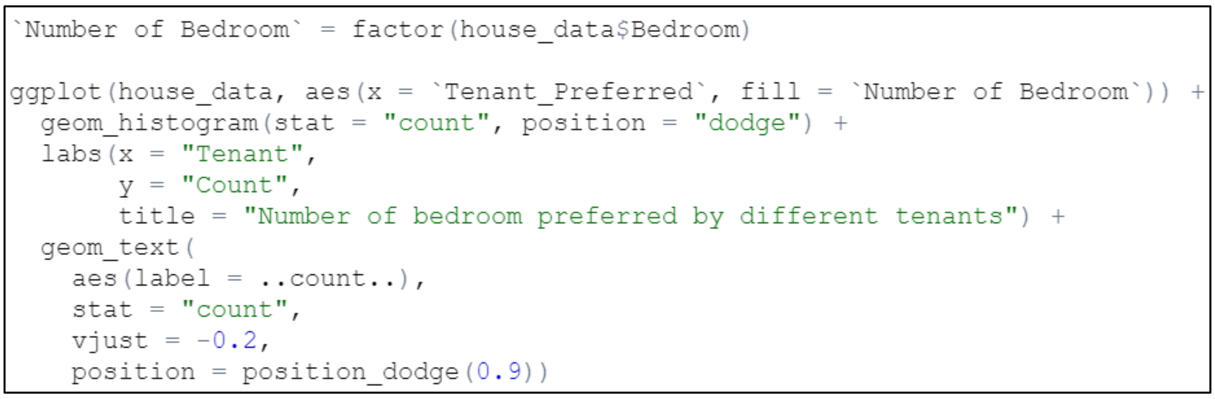


Figure 100: Analysis result 8.1

The above bar chart shows that the average house size preferred by bachelors is 1,015.11 sqft. As for tenants that are either bachelors or family, they preferred houses with an average size of 930.27 sqft. Family people prefer the highest average house size, which is 1,155.33 sqft.

3.8.2 - Analysis 8.2: Find the number of bedrooms preferred by different tenants



```
`Number of Bedroom` = factor(house_data$Bedroom)

ggplot(house_data, aes(x = `Tenant_Preferred`, fill = `Number of Bedroom`)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "Tenant",
       y = "Count",
       title = "Number of bedroom preferred by different tenants") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9))
```

Figure 101: Source code for analysis 8.2

The above source code is to generate a histogram that can be used to visualize the number of bedrooms preferred by different tenants. Kindly refers to the source code explanation of plotting a histogram in analysis 1.2.

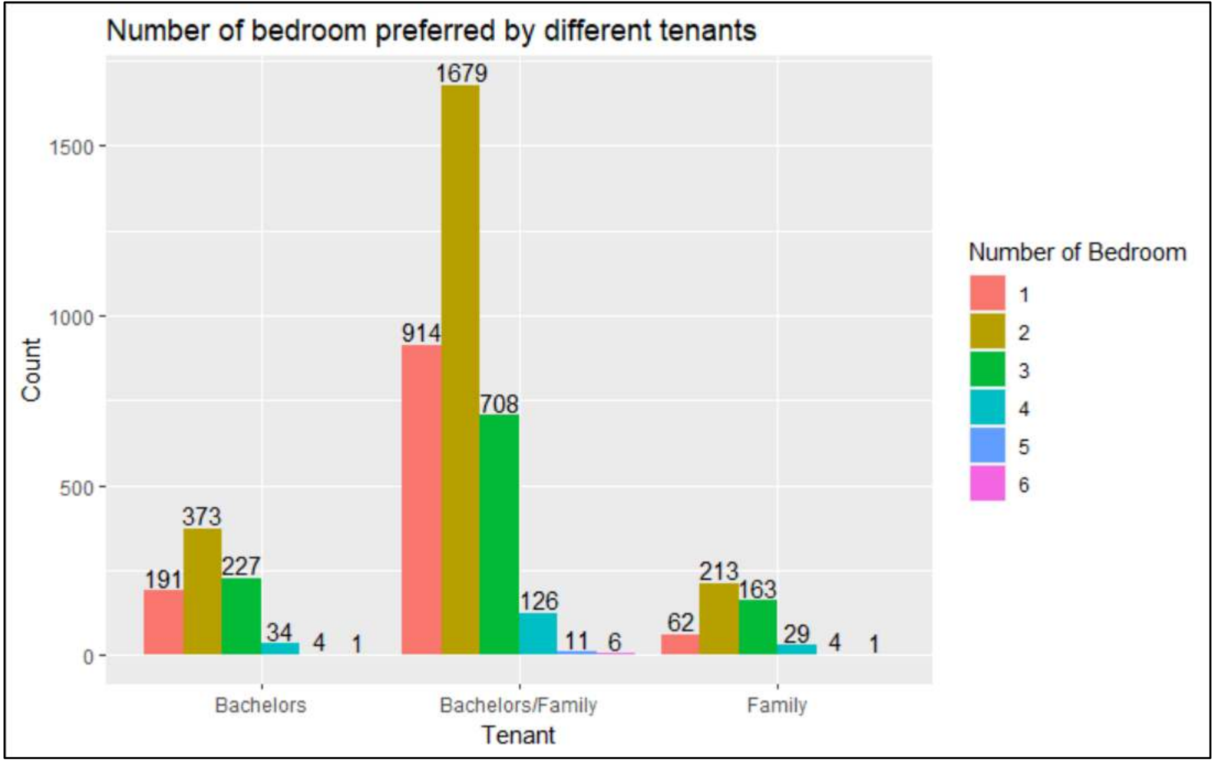
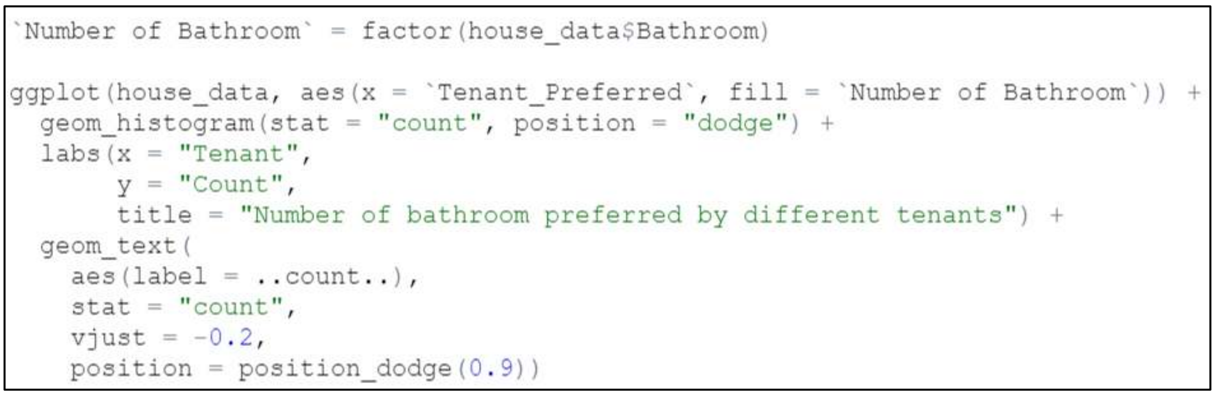


Figure 102: Analysis result 8.2

The histogram above shows that all different tenants mostly prefer to rent a house with 2 bedrooms. More bachelors and families prefer houses with 3 bedrooms than 2, that is 227 and 163 of them respectively. As for tenants that are either bachelor or family, there are more of them who prefers houses with one bedroom than 3, that is 914 of them.

3.8.3 - Analysis 8.3: Find the number of bathrooms preferred by different tenants



```
`Number of Bathroom` = factor(house_data$Bathroom)

ggplot(house_data, aes(x = `Tenant_Preferred`, fill = `Number of Bathroom`)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(x = "Tenant",
       y = "Count",
       title = "Number of bathroom preferred by different tenants") +
  geom_text(
    aes(label = ..count..),
    stat = "count",
    vjust = -0.2,
    position = position_dodge(0.9))
```

Figure 103: Source code for analysis 8.3

The above source code is to generate a histogram that can be used to visualize the number of bathrooms preferred by different tenants. Kindly refers to the source code explanation of plotting a histogram in analysis 1.2.

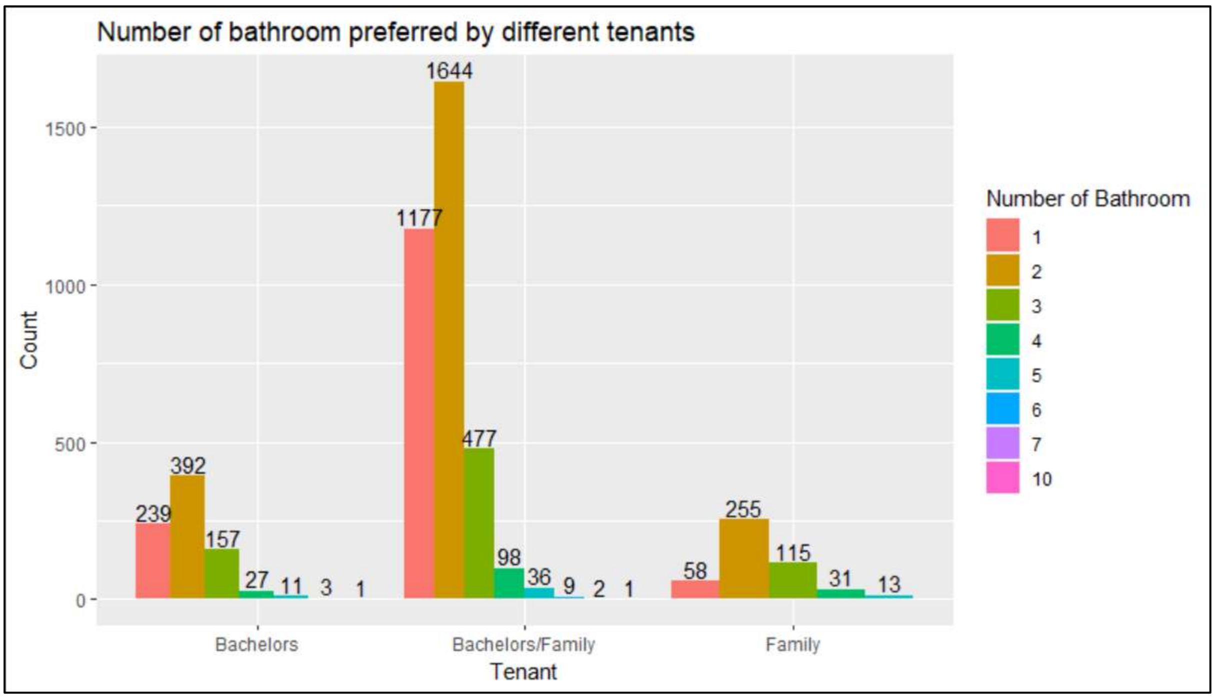


Figure 104: Analysis result 8.3

The histogram above shows that all different tenants mostly prefer to rent a house with 2 bathrooms. More bachelors and tenants that are either bachelor or family would prefer houses with one bedroom than 3, that is 239 and 1,177 of them respectively. As for family people, there are more of them who prefers houses with 3 bedroom than one, that is 115 of them.

3.8.4 - Conclusion for question 8

There are a total of 3 analyses being done for question 8 to find out how big a house different tenants prefer. Analysis 8.1 shows that family people prefer houses with an average size of 1,155.33 sqft, which is the biggest compared to other groups of tenants. All different tenants would prefer houses with 2 bedrooms and 2 bathrooms.

3.9 - Question 9: What kind of houses are posted in July?

In analysis 7.8, the result shows that there is a sudden increase of number tenants renting houses posted in July. Therefore, this question aims to find out what kind of houses are posted in July.

3.9.1 - Analysis 9.1: Find the furnishing status of houses posted in July.

```
# Get data.
data = house_data %>%
        filter(Posted_Date >= as.Date("2022-7-1")) %>%
        {table(.$Furnishing_Status)} %>%
        data.frame() %>%
        set_colnames(c("Furnishing Status", "Count"))

# Plot 3D pie chart.
pie3D(data$Count,
      labels = paste(data$`Furnishing Status`, "\n",
                     get_percent(data$Count), "%",
                     "(", data$Count, ")",
                     sep = ""),
      height = 0.12,
      explode = 0.1,
      main = "Furnishing status of houses posted in July",
      col = c("#D0D1E6","#74A9CF", "#0570D0"),
      border = "white")
```

Figure 105: Source code for analysis 9.1

The above source code is to generate a 3D pie chart that can be used to visualize the furnishing status of houses posted in July. The very first part is to obtain data in which the date is greater or equal to “`2022-7-1`” using the `filter()` function. This ensures the data obtained is later than the first of July. The `table()` built-in function is used to perform a tabulation of the furnishing status of each house. The table of data is then converted into a data frame using `data.frame()` function and the column name is renamed using the `set_colnames()` function. Kindly refers to the source code explanation of plotting a 3D pie chart in analysis 6.6.

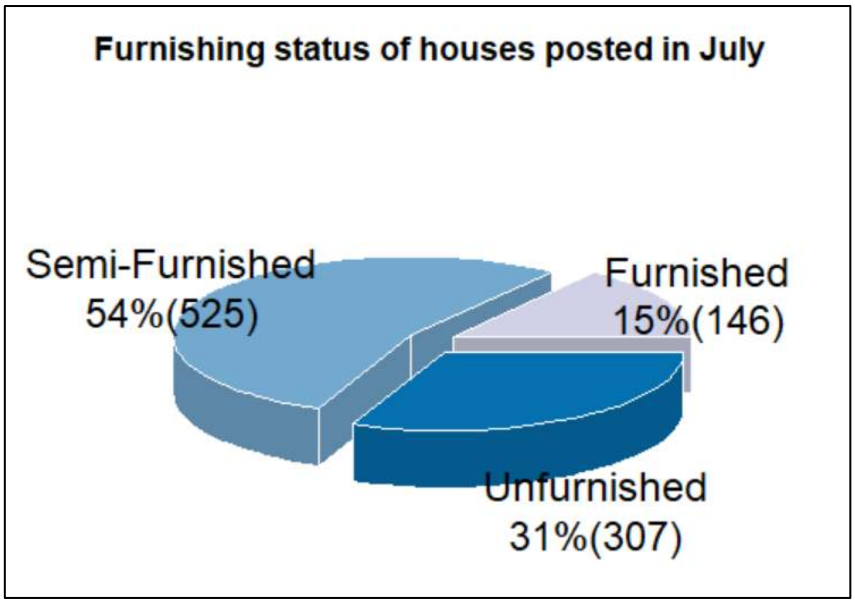
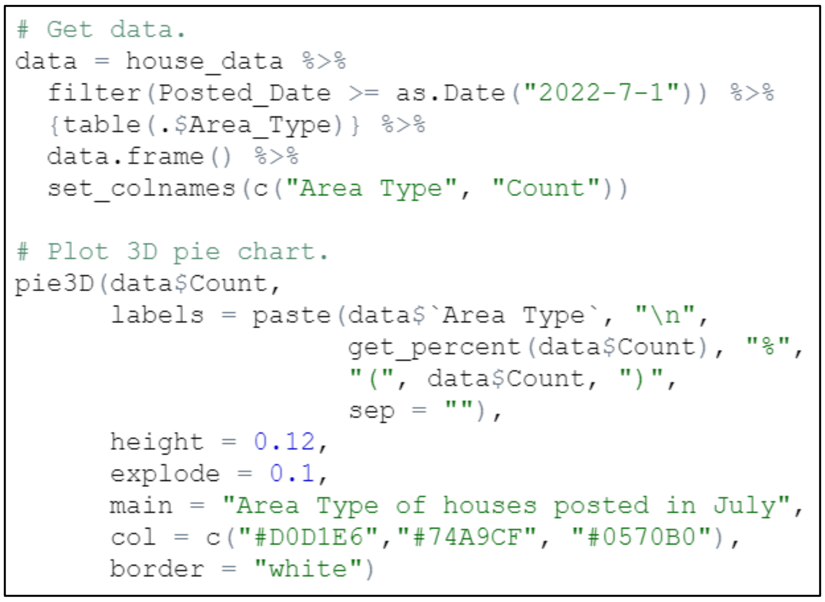


Figure 106: Analysis result 9.1

The above 3D pie chart shows that the majority of the houses posted in July are semi-furnished, that is 54% or 525 of them. There are also 31% or 307 unfurnished houses posted in July. The furnished houses are the least posted ones, which is only 15% or 146 of them.

3.9.2 - Analysis 9.2: Find the area type of houses posted in July.



```
# Get data.
data = house_data %>%
  filter(Posted_Date >= as.Date("2022-7-1")) %>%
  {table(.$Area_Type)} %>%
  data.frame() %>%
  set_colnames(c("Area Type", "Count"))

# Plot 3D pie chart.
pie3D(data$Count,
      labels = paste(data$`Area Type`, "\n",
                     get_percent(data$Count), "%",
                     "(", data$Count, ")",
                     sep = ""),
      height = 0.12,
      explode = 0.1,
      main = "Area Type of houses posted in July",
      col = c("#D0D1E6","#74A9CF", "#0570B0"),
      border = "white")
```

Figure 107: Source code for analysis 9.2

The above source code is to generate a 3D pie chart that can be used to visualize the area type of houses posted in July. Kindly refers to the source code explanation in analysis 9.1, which is highly similar to this analysis.

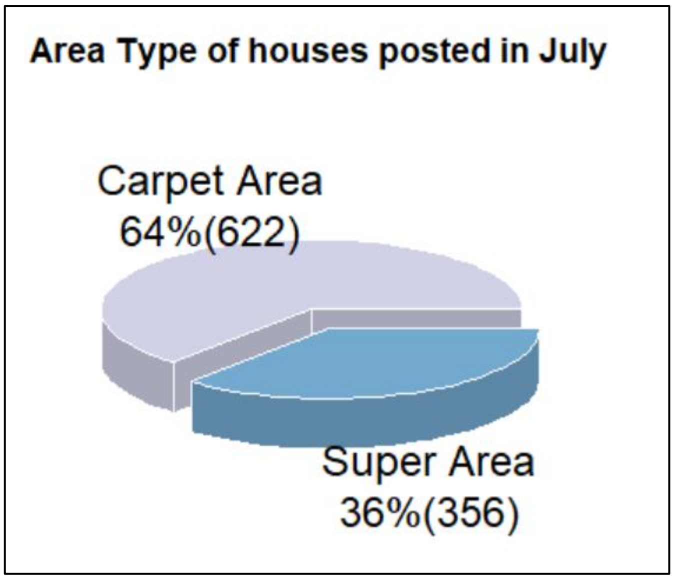
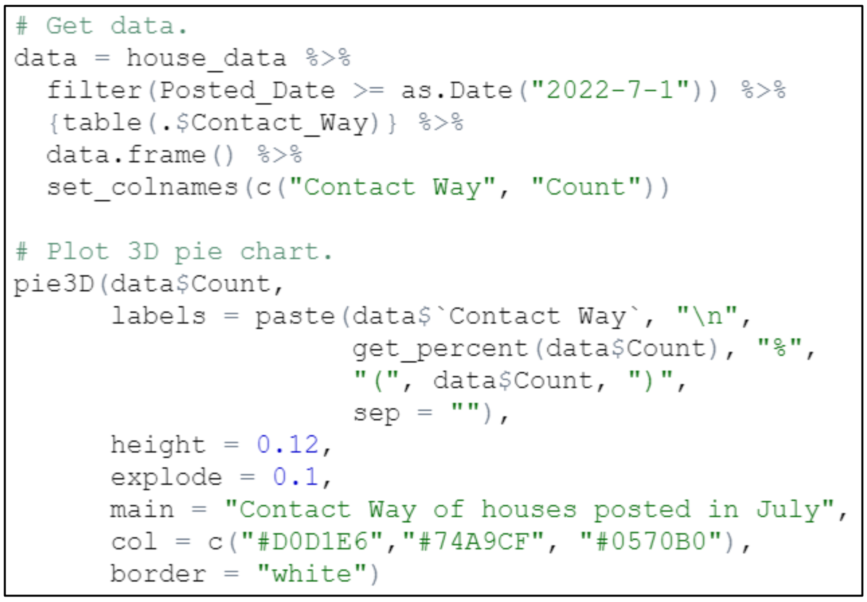


Figure 108: Analysis result 9.2

The above 3D pie chart shows that houses in which the size is calculated based on carpet area are more than the super area in July. The are 64% or 622 of houses with the area type of carpet area and there are only 36% or 356 with the area type of super area.

3.9.3 - Analysis 9.3: Find the contact way of houses posted in July.



```
# Get data.
data = house_data %>%
  filter(Posted_Date >= as.Date("2022-7-1")) %>%
  {table(.$Contact_Way)} %>%
  data.frame() %>%
  set_colnames(c("Contact Way", "Count"))

# Plot 3D pie chart.
pie3D(data$Count,
      labels = paste(data$`Contact Way`, "\n",
                     get_percent(data$Count), "%",
                     "(", data$Count, ")",
                     sep = ""),
      height = 0.12,
      explode = 0.1,
      main = "Contact Way of houses posted in July",
      col = c("#D0D1E6","#74A9CF", "#0570B0"),
      border = "white")
```

Figure 109: Source code for analysis 9.3

The above source code is to generate a 3D pie chart that can be used to visualize the contact way of houses posted in July. Kindly refers to the source code explanation in analysis 9.1, which is highly similar to this analysis.

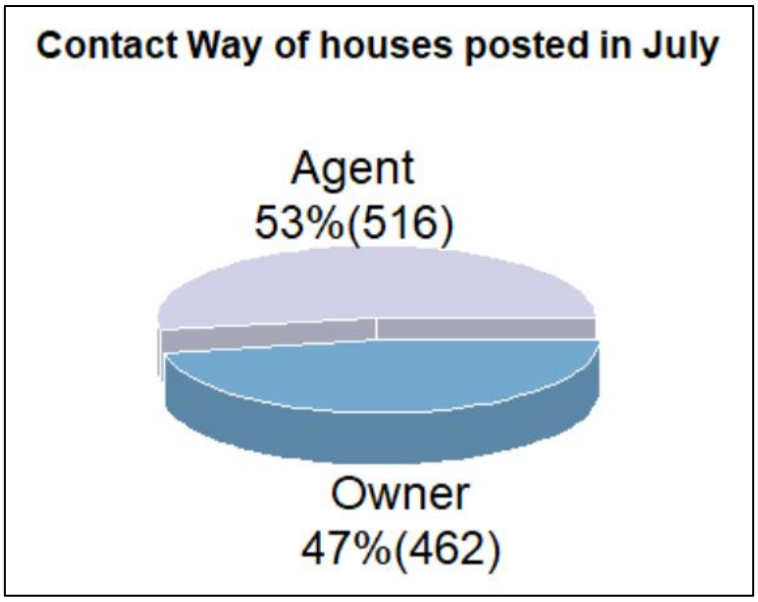
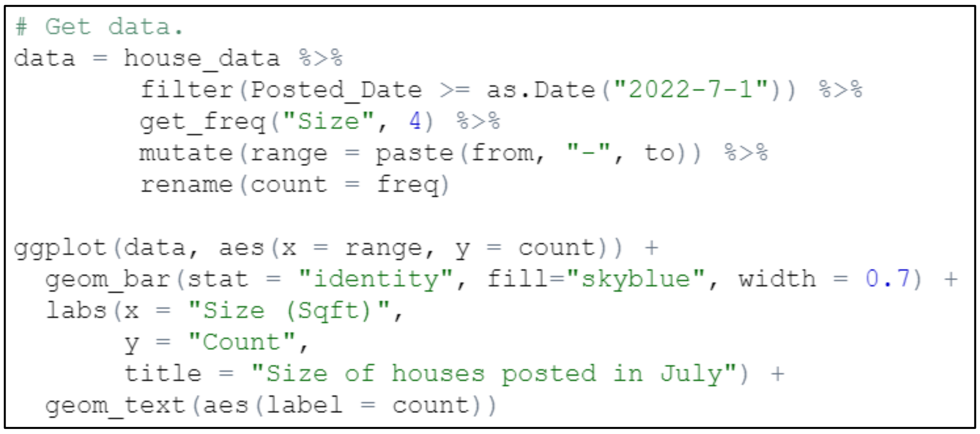


Figure 110: Analysis result 9.3

The above 3D pie chart shows that houses with the contact way of agents are more than owners in July. The are 53% or 516 of houses with the contact way of agents and there are 47% or 462 with the contact way of owners.

3.9.4 - Analysis 9.4: Find the size of houses posted in July.



```
# Get data.
data = house_data %>%
        filter(Posted_Date >= as.Date("2022-7-1")) %>%
        get_freq("Size", 4) %>%
        mutate(range = paste(from, "-", to)) %>%
        rename(count = freq)

ggplot(data, aes(x = range, y = count)) +
  geom_bar(stat = "identity", fill="skyblue", width = 0.7) +
  labs(x = "Size (Sqft)",
       y = "Count",
       title = "Size of houses posted in July") +
  geom_text(aes(label = count))
```

Figure 111: Source code for analysis 9.4

The above source code is to generate a bar chart that can be used to visualize the size of houses posted in July. The very first part is to obtain data in which the date is greater or equal to "`2022-7-1`" using the `filter()` function. This ensures the data obtained is later than the first of July. The `get_freq()` function is used to obtain the frequency of the same data in ranges within the `Size` column. The range of house sizes is obtained by concatenating the `from` and `to` columns provided by the `get_freq()` function. The `rename()` function is used to change the name of the `freq` column to `count`. Kindly refer to the source code explanation in analysis 2.1 for details of plotting a bar chart.

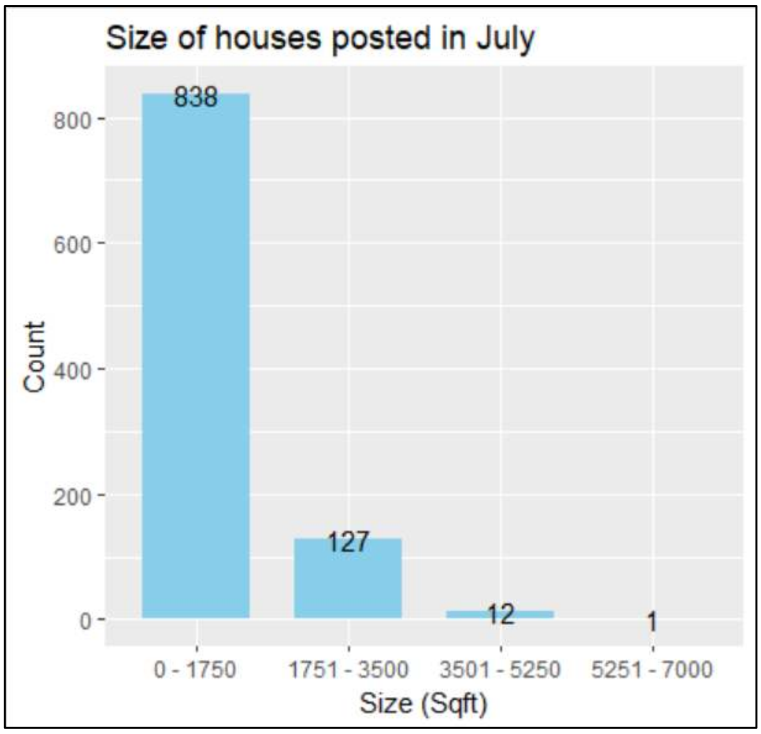


Figure 112: Analysis result 9.4

The above bar chart shows that most houses posted in July are below the size of 1,750 sqft, that is 838 of them. There are 127 houses posted in July that fall within the range of 1,751 to 3,500 sqft.

3.9.5 - Analysis 9.5: Find the number of bedrooms in houses posted in July.

```
# Get data.
data = house_data %>%
  filter(Posted_Date >= as.Date("2022-7-1")) %>%
  {table(.$Bedroom)} %>%
  data.frame() %>%
  set_colnames(c("Bedroom", "Count"))

ggplot(data, aes(x = Bedroom, y = Count)) +
  geom_bar(stat = "identity", fill="skyblue", width = 0.7) +
  labs(x = "Bedroom",
       y = "Count",
       title = "Number of bedrooms in houses posted in July") +
  geom_text(aes(label = Count))
```

Figure 113: Source code for analysis 9.5

The above source code is to generate a bar chart that can be used to visualize the number of bedrooms in houses posted in July. Kindly refers to the source code explanation in analysis 9.1, which is highly similar to this analysis. As for plotting a bar chart, kindly refer to the source code explanation in analysis 2.1.

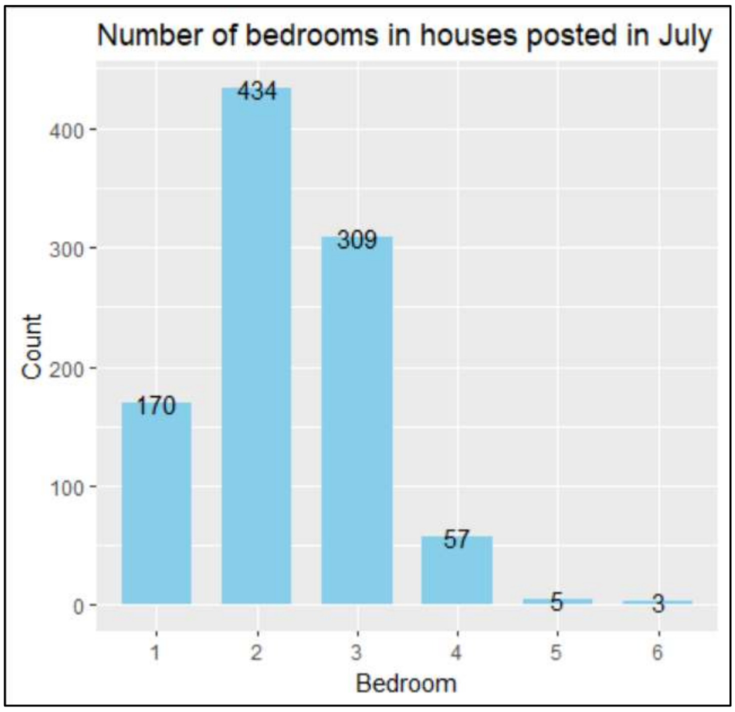


Figure 114: Analysis result 9.5

The above bar chart shows that most houses posted in July are having 2 bedrooms, that is 434 of them. There are 309 houses posted in July having 3 bedrooms and 170 houses having one bedroom.

3.9.6 - Analysis 9.6: Find the number of bathrooms in houses posted in July.

```
# Get data.
data = house_data %>%
  filter(Posted_Date >= as.Date("2022-7-1")) %>%
  {table(.$Bathroom)} %>%
  data.frame() %>%
  set_colnames(c("Bathroom", "Count"))

ggplot(data, aes(x = Bathroom, y = Count)) +
  geom_bar(stat = "identity", fill="skyblue", width = 0.7) +
  labs(x = "Bathroom",
       y = "Count",
       title = "Number of bathrooms in houses posted in July") +
  geom_text(aes(label = Count))
```

Figure 115: Source code for analysis 9.6

The above source code is to generate a bar chart that can be used to visualize the number of bathrooms in houses posted in July. Kindly refers to the source code explanation in analysis 9.1, which is highly similar to this analysis. As for plotting a bar chart, kindly refer to the source code explanation in analysis 2.1.

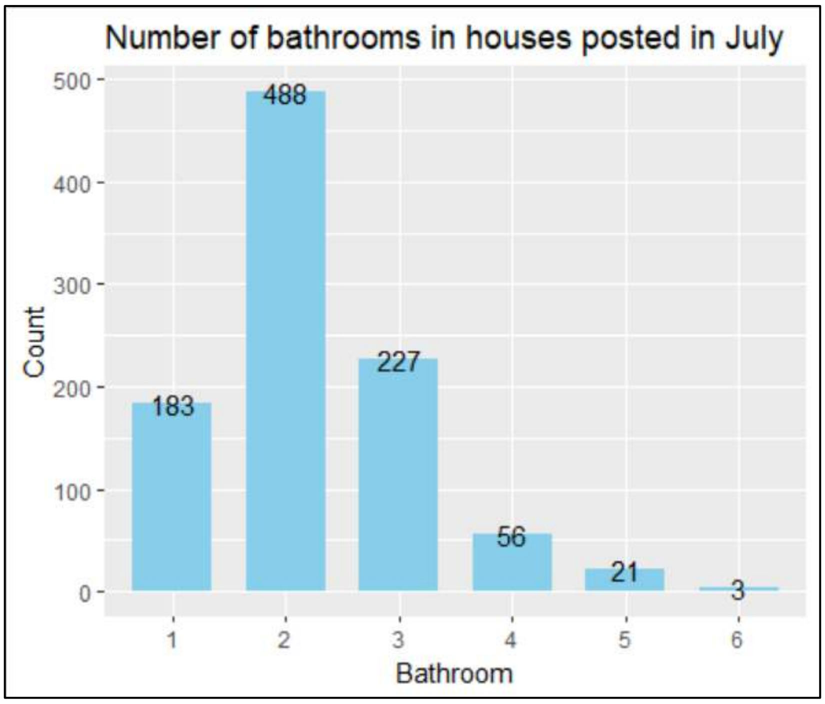


Figure 116: Analysis result 9.6

The above bar chart shows that most houses posted in July are having 2 bathrooms, that is 488 of them. There are 227 houses posted in July having 3 bedrooms and 183 houses having one bathroom.

3.9.7 - Analysis 9.7: Find the city of houses posted in July.

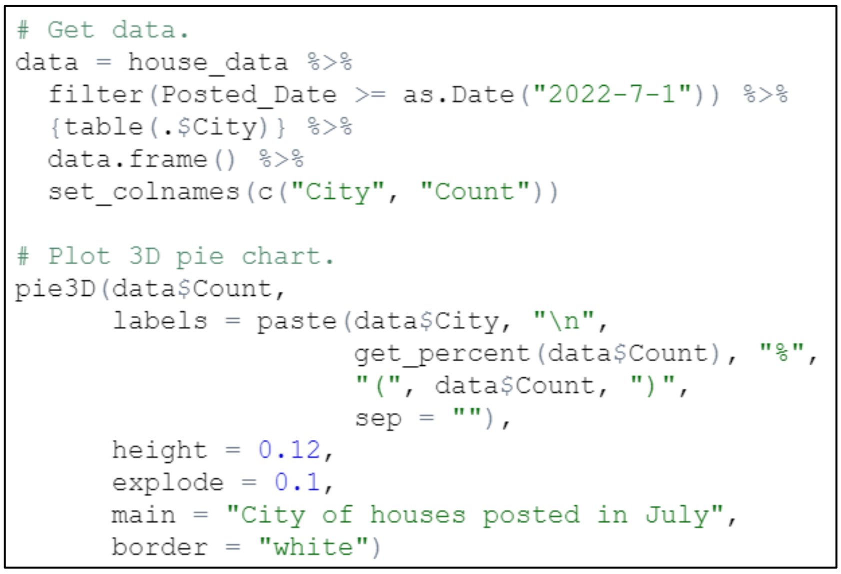

```
# Get data.
data = house_data %>%
  filter(Posted_Date >= as.Date("2022-7-1")) %>%
  {table(.$City)} %>%
  data.frame() %>%
  set_colnames(c("City", "Count"))

# Plot 3D pie chart.
pie3D(data$Count,
      labels = paste(data$City, "\n",
                     get_percent(data$Count), "%",
                     "(", data$Count, ")",
                     sep = ""),
      height = 0.12,
      explode = 0.1,
      main = "City of houses posted in July",
      border = "white")
```

Figure 117: Source code for analysis 9.7

The above source code is to generate a 3D pie chart that can be used to visualize the city of houses posted in July. Kindly refers to the source code explanation in analysis 9.1, which is highly similar to this analysis.

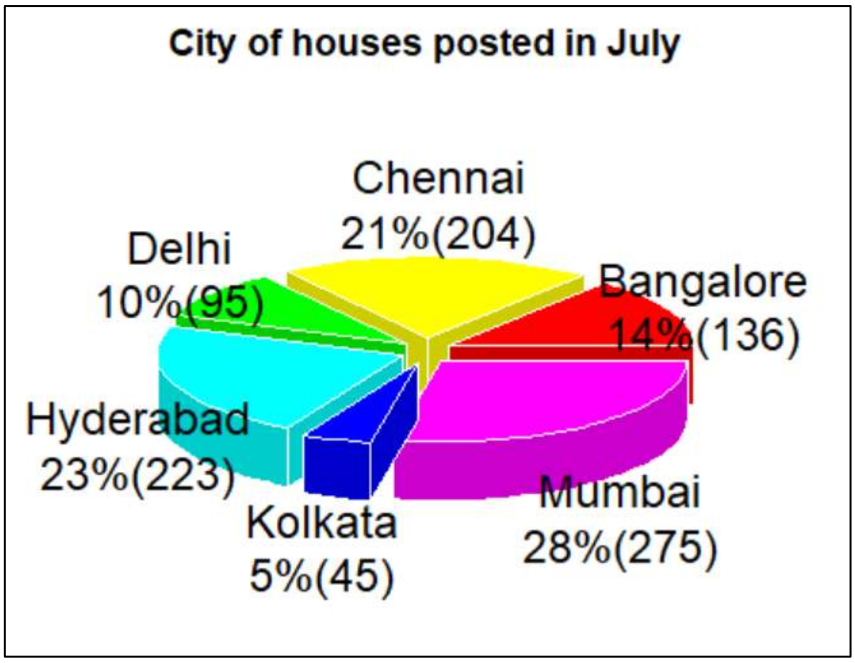


Figure 118: Analysis result 9.7

The above 3D pie chart shows that the majority of the houses posted in July are located in Mumbai, that is 28% or 275 of them. There are also 23% or 223 houses located in Hyderabad posted in July. Houses located in Kolkata are the least posted ones, which is only 5% or 45 of them.

3.9.8 - Conclusion for question 9

There are a total of 7 analyses being done for question 9 to find out what kind of house was posted in July. In summary, most houses posted in July are:

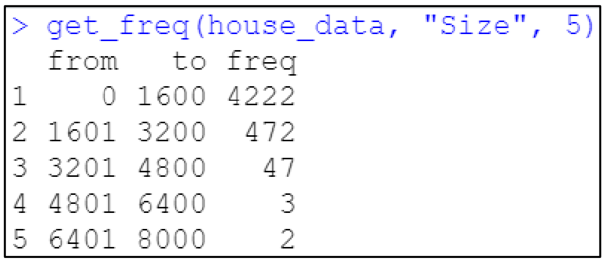
- below 1600 sqft
- having 2 bedrooms and 2 bathrooms
- located in Mumbai
- semi-furnished
- calculated based on carpet area
- having the contact way of agents

4.0 - Extra Feature

4.1 - User-defined function

4.1.1 - Get frequencies of the same data

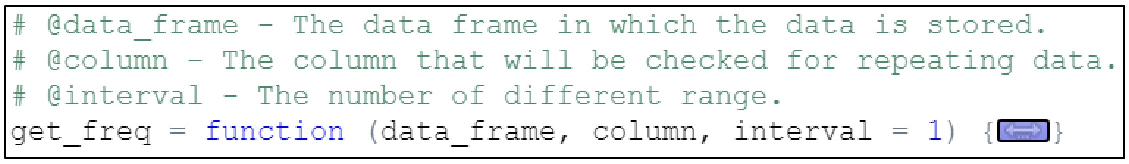
Unlike built-in functions, which already exist in the R language without having to define them, the user-defined function is a block of code written by the user to perform a specific task (Kosourova, 2022).



```
> get_freq(house_data, "Size", 5)
  from   to freq
1    0 1600 4222
2 1601 3200  472
3 3201 4800   47
4 4801 6400    3
5 6401 8000    2
```

Figure 119: Sample output of get_freq function

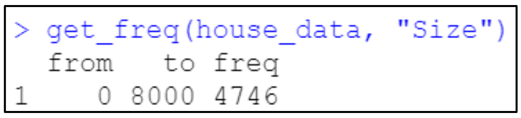
The `get_freq()` function is written to obtain the frequency of the same data occurring in a specific column. The reason for creating this function stems from many considerations, the first being its reusability as there are several analyses involving the counting of the same categorical data. For attributes such as house sizes and the floor that have a wide range of categorical data, it is difficult to plot a visible chart or graph. In analysis 6.1, instead of counting the frequency of each unique house size, this function allows the house sizes to be categorized and counted based on specific ranges.



```
# @data_frame - The data frame in which the data is stored.
# @column - The column that will be checked for repeating data.
# @interval - The number of different range.
get_freq = function (data_frame, column, interval = 1) {⟺}
```

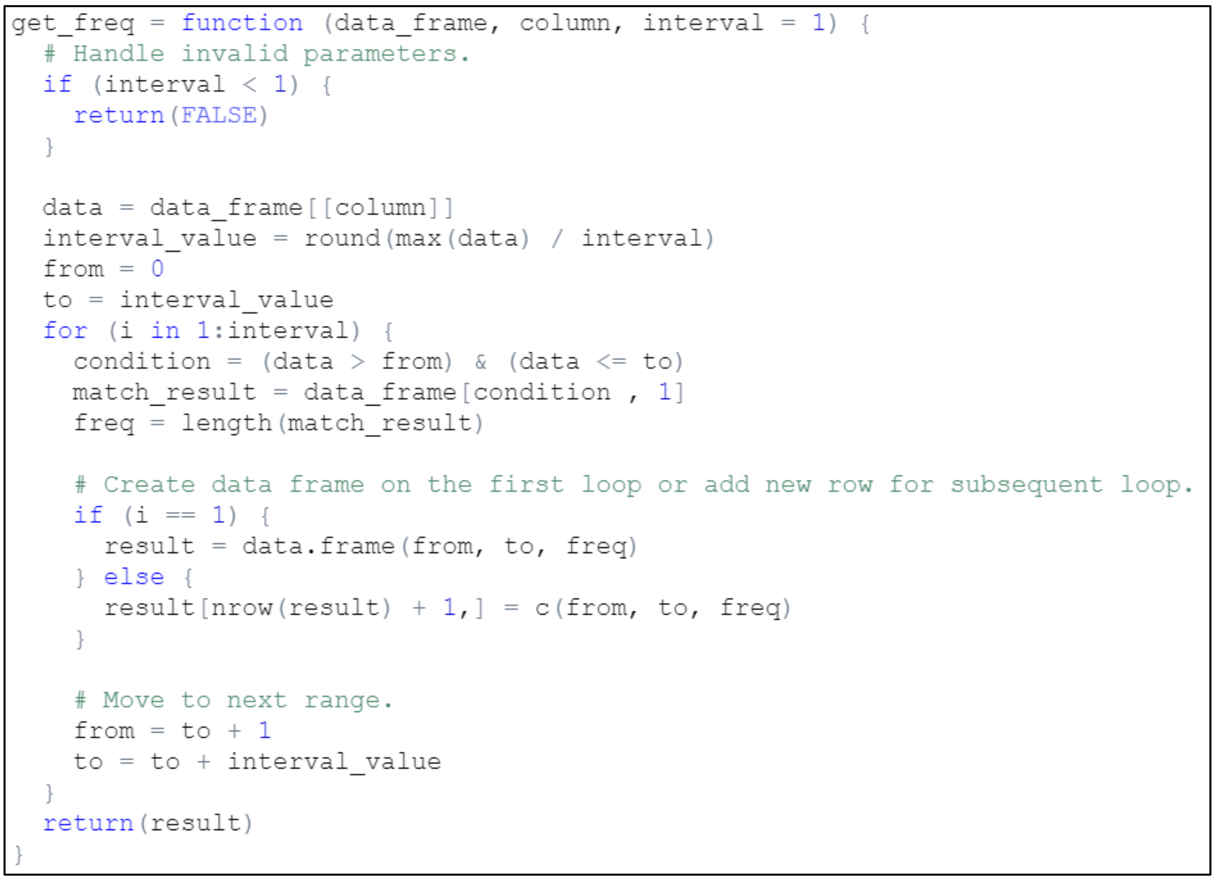
Figure 120: Overview of the get_freq function

There are three parameters that the `get_freq()` function will be receiving. The first and second parameters are pretty self-explanatory based on their naming. The `data_frame` parameter will be the data frame in which the data is stored, and the `column` refers to the name of the column that will be checked for repeating data. The `interval` parameter will be the number of different ranges based on the repeating data. Back to the example of houses' size, if the given value for the `interval` parameter is 5, there will be 5 different ranges of houses' size being counted as shown in figure 119. By default, the value of the `interval` parameter is set to one if none is provided.



```
> get_freq(house_data, "Size")
  from   to freq
1    0 8000 4746
```

Figure 121: Sample output of get_freq function with the default interval value



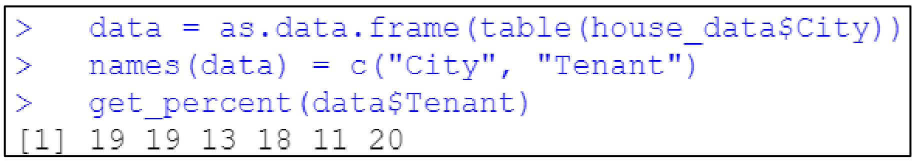
```
get_freq = function (data_frame, column, interval = 1) {
  # Handle invalid parameters.
  if (interval < 1) {
    return(FALSE)
  }

  data = data_frame[[column]]
  interval_value = round(max(data) / interval)
  from = 0
  to = interval_value
  for (i in 1:interval) {
    condition = (data > from) & (data <= to)
    match_result = data_frame[condition , 1]
    freq = length(match_result)

    # Create data frame on the first loop or add new row for subsequent loop.
    if (i == 1) {
      result = data.frame(from, to, freq)
    } else {
      result[nrow(result) + 1,] = c(from, to, freq)
    }

    # Move to next range.
    from = to + 1
    to = to + interval_value
  }
  return(result)
}
```

Figure 122: Source code of get_freq function

The first part of the function is to make sure that the number provided for the `interval` parameter is not less than 1, else a Boolean value of false will be returned. If the interval number is valid, the interval value will be calculated by dividing the maximum value of data in the given dataset column by the interval number. The `round()` function is also being used in this calculation to round off the decimal points that could lead to errors for the following algorithm. The `data` is being used to store the data of the specific column of the given data frame. The core idea of this function's algorithm is by keeping track of each data range and count the frequency of data that falls within this range. Therefore, two variables named `from` and `to` are being used to store the starting and ending point of each range of data. The initial value for `from` is 0, which will be the starting value for the first range of data whereas the `to` stores the first interval value obtained as the initial value.

The process to obtain the frequency of data within different ranges of value is iterated through a for loop. The number of iterations is dependent on the provided interval number. This means there will be 5 iterations in total if the provided value for the `interval` parameter is 5. Each iteration will extract the data in which the value is between the starting and ending points of the current iteration. This is done by specifying a condition where the value of the data shall be greater than the value of `from` and less than or equal to the value of `to`. The extracted data is then stored in the `match_result` variable. The frequency of data that falls within the range of value is counted using the `length()` function and stored in the `Freq` variable. Once setting the starting and ending point for the next range of data, the result for the current range of data will be added to a data frame named `result`. The data frame will be created along with the `From`, `To` and `Freq` variables during the first loop and the subsequence loop will add a new row to the data frame. After the for loop completes all the iterations, the result will be returned from the function and accessible.

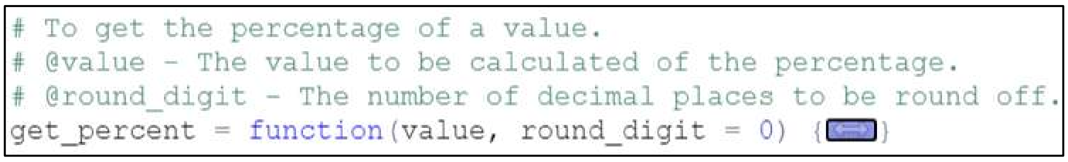
4.1.2 - Get percentages of data



```
>   data = as.data.frame(table(house_data$City))
>   names(data) = c("City", "Tenant")
>   get_percent(data$Tenant)
[1] 19 19 13 18 11 20
```

Figure 123: Sample output of get_percent function

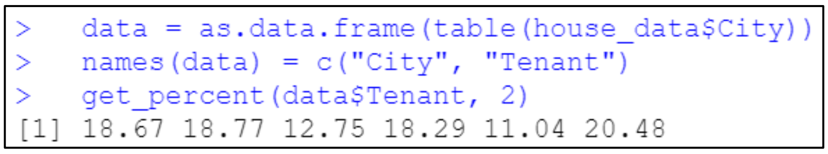
The `get_percent()` function is written to calculate the percentage of a value. This function is being created for the sake of reusability in several analyses that involve the counting of percentages. One of the use cases would be obtaining the percentages of houses in each city in analysis 6.4.



```
# To get the percentage of a value.
# @value - The value to be calculated of the percentage.
# @round_digit - The number of decimal places to be round off.
get_percent = function(value, round_digit = 0) {⟺}
```

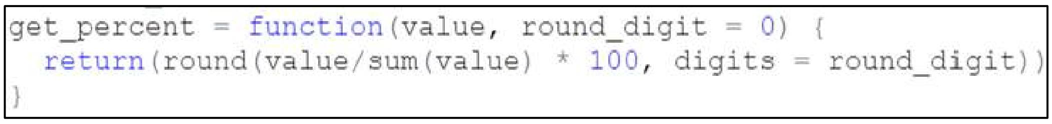
Figure 124: Overview of the get_percent function

There are two parameters that the `get_percent()` function will be receiving, which are the value to be calculated and the number of decimal places to be rounded off. By default, the value of the `round_digit` parameter is set to zero if none is provided.



```
>   data = as.data.frame(table(house_data$City))
>   names(data) = c("City", "Tenant")
>   get_percent(data$Tenant, 2)
[1] 18.67 18.77 12.75 18.29 11.04 20.48
```

Figure 125: Sample output of get_percent function with the default interval value



```
get_percent = function(value, round_digit = 0) {
  return(round(value/sum(value) * 100, digits = round_digit))
}
```

Figure 126: Source code of get_percent function

The first part of this function is by getting the denominator using the `sum()` function, which is used to find the sum of the values (Lathiya, 2021). The value is then divided by the value of the denominator and multiplied by 100. Lastly, the `round()` function is used to round off the decimal digits of the result.

4.2 – Additional ggplot 2 features

4.2.1 - Text label on charts

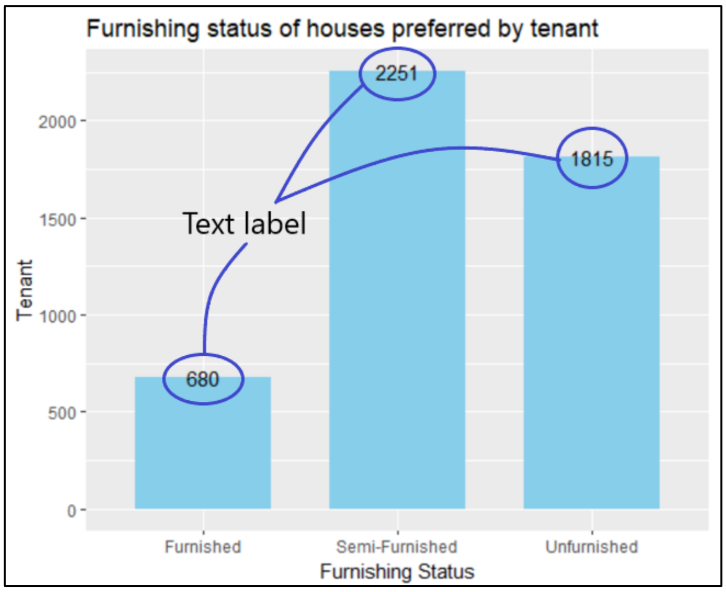
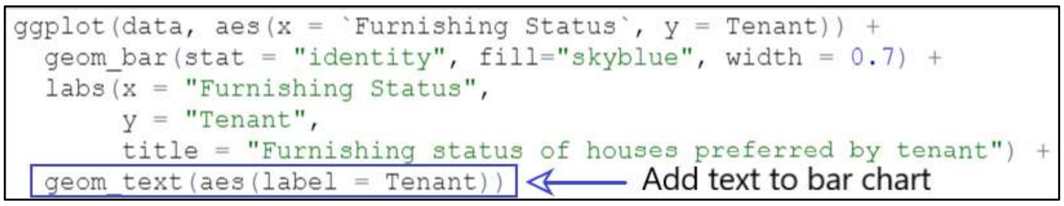


Figure 127: Text label on a bar chart

The bar charts and histograms provided for the analysis have included the text label that shows the value on each bar. Such a feature helps the HRM team to see the exact number of data present in the bar chart and make precise decisions. For example, it is easy for them to know how many tenants prefer houses with different furnishing statuses as shown in figure 127.



```
ggplot(data, aes(x = `Furnishing Status`, y = Tenant)) +
  geom_bar(stat = "identity", fill="skyblue", width = 0.7) +
  labs(x = "Furnishing Status",
       y = "Tenant",
       title = "Furnishing status of houses preferred by tenant") +
  geom_text(aes(label = Tenant))
```

Figure 128: Source code for adding text label on a chart

The text on the bar in a bar chart is added using the `geom_text()` function. This function consists of the `aes()` that includes the parameter of the `label` with the value of the y-axis, that is the `Tenant`.

4.2.2 - Trendline on graph

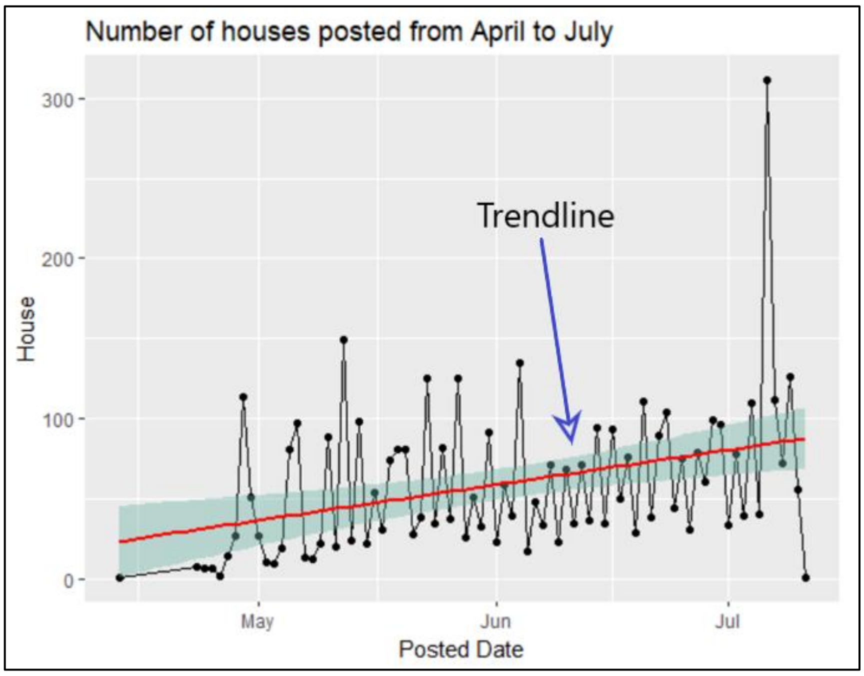
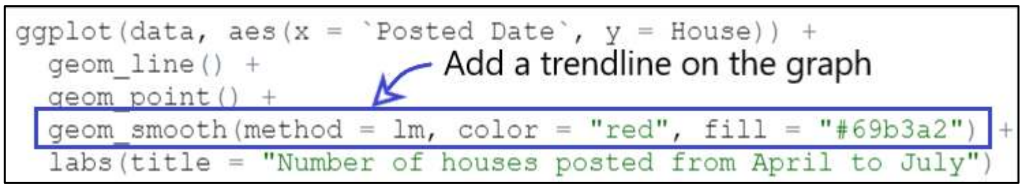


Figure 129: Trendline on a graph

Graph such as a connected scatter plot provided for the analysis has included a trendline that illustrates the prevailing direction of the data value. This feature helps the HRM team to see the overall direction of the data presented in the graph and make better decisions. This makes it easy for them to visualize the trend of the number of houses posted from April until July.



```
ggplot(data, aes(x = `Posted Date`, y = House)) +
  geom_line() +
  geom_point() +
  geom_smooth(method = lm, color = "red", fill = "#69b3a2") +
  labs(title = "Number of houses posted from April to July")
```

Figure 130: Source code for adding a trendline on a graph

The trendline is added to a graph using the `geom_smooth()` function. The default trendline is a loess line. The method parameter with the value of “lm” which stands for the linear model is used to specify a straight-line linear model (Ebner, 2022). The colour of the trendline can also be changed by specifying the value for the `color` parameter. By default, the `se` parameter is set to true, which will add the confidence interval around the trendline. The `fill` parameter would be the color that fills the area of the confidence interval.

4.2.3 - Doughnut chart

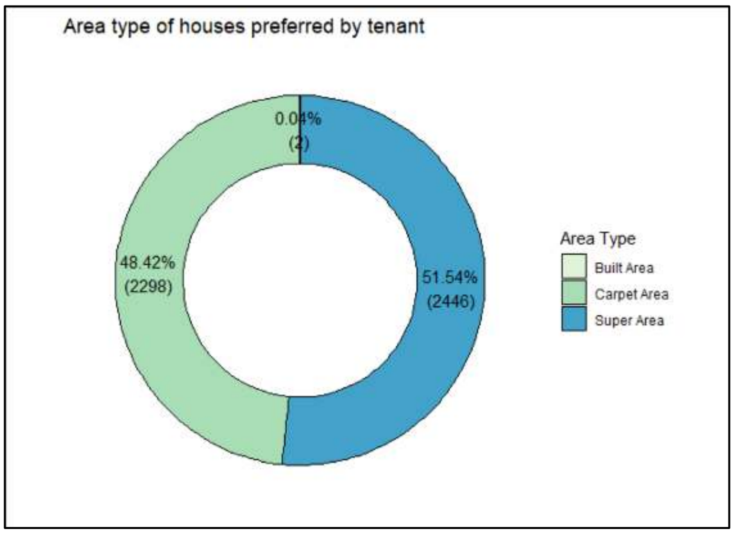


Figure 131: Doughnut chart

A doughnut chart is a pie chart with its centre removed. Instead of slices, the arc segments are used to display individual dimensions. Each of the doughnut arcs has the same width, but a different length. This makes it easier for the HRM team to compare individual categories or dimensions to the larger whole. This is because they just need to compare which arc is longer when comparing which data is greater.

4.2.4 - Stacked bar chart

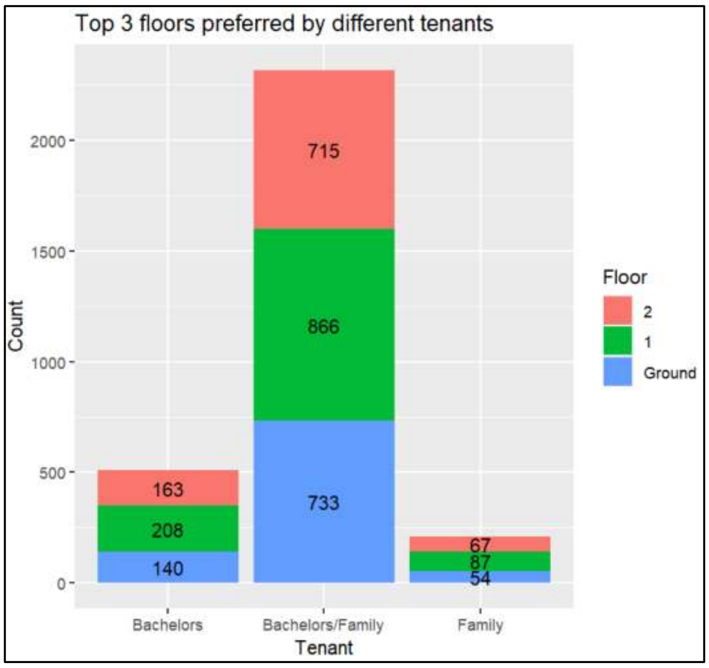


Figure 132: Stacked bar chart

When comparing data in analysis such as 1.3 and 7.6, the stacked bar chart makes it easier for the HRM team to see the comparison on a stacked bar chart versus a combined chart.

4.2.5 - Treemap

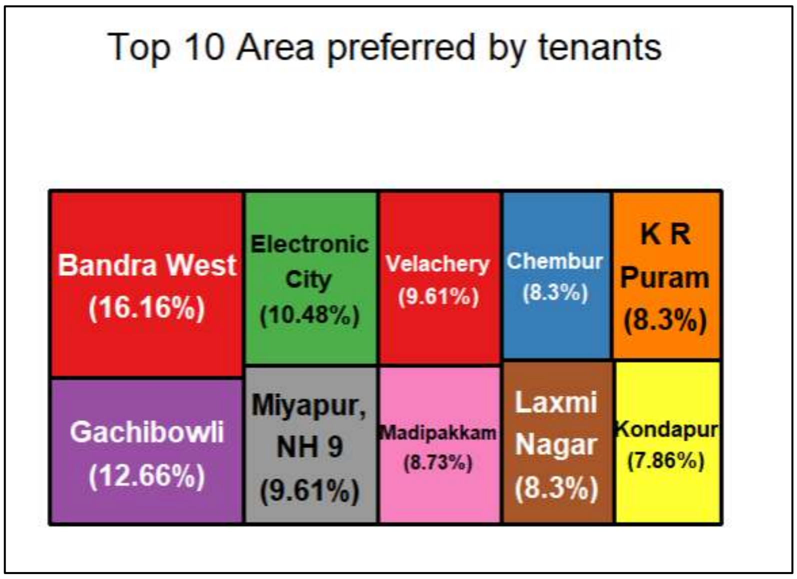


Figure 133: Treemap

The treemap capture relative sizes of data categories, allowing the HRM team to have a quick perception of the items that are large contributors to each category.

5.0 - Conclusion

In conclusion, there are several analyses have been performed with the given dataset to help the HRM team in decision-making. These analyses have involved various data analytics techniques such as pre-processing data, data exploration, manipulation, and visualization. Throughout the analysis, the relationship of multiple factors has been visualized and explained to determine how people choose their rental houses. However, the HRM team is not encouraged to fully rely on the analysis result due to the uncertainty of interest rates these days, which makes it difficult to predict the property market.

6.0 - References

Chidalu, O. T. (n.d.). *What is the names() function in R?* Retrieved from Educative: https://www.educative.io/answers/what-is-the-names-function-in-r

CN, P. (2022, August 4). *The head() and tail() function in R - Detailed Reference*. Retrieved from Digital Ocean: https://www.digitalocean.com/community/tutorials/head-and-tail-function-r

Ebner, J. (2022, July 19). *How to Use geom_smooth in R*. Retrieved from Sharp Sight: https://www.sharpsightlabs.com/blog/geom_smooth/

Haaf, S. (2019, April 2). *Easy multi-panel plots in R using facet_wrap() and facet_grid() from ggplot2*. Retrieved from Zev Ross: http://zevross.com/blog/2019/04/02/easy-multi-panel-plots-in-r-using-facet_wrap-and-facet_grid-from-ggplot2/

Johnson, D. (2022, November 19). *Factor in R: Categorical Variable & Continuous Variables.* Retrieved from Guru99: https://www.guru99.com/r-factor-categorical-continuous.html#:~:text=What%20is%20Factor%20in%20R,integer%20data%20values%20as%20levels.

Kosourova, E. (2022, June 15). *How to Write Functions in R (with 18 Code Examples).* Retrieved from Data Quest: https://www.dataquest.io/blog/write-functions-in-r/#:~:text=To%20declare%20a%20user%2Ddefined,at%20each%20of%20them%20separately.

Lathiya, K. (2021 , September 11). *Sum in R: How to Find Sum of Vectors.* Retrieved from R-Lang: https://r-lang.com/sum-in-r/

Madhugiri, D. (2022, March 7). *A Comprehensive Guide on ggplot2 in R.* Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2022/03/a-comprehensive-guide-on-ggplot2-in-r/

Naveen. (2022, Janruary 7). *Data Manipulation in R.* Retrieved from Intelli Paat: https://intellipaat.com/blog/tutorial/r-programming/data-manipulation-in-r/

Willeams, K. (2017, November). *Pipes in R Tutorial For Beginners.* Retrieved from Data Camp: https://www.datacamp.com/tutorial/pipe-r-tutorial

Zach. (2022, April 19). *How to Use the dim() Function in R*. Retrieved from Statology: https://www.statology.org/r-dim-function/