ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT127-3-2-PFDA

PROGRAMMING FOR DATA ANALYSIS

APU2F2209CS

HAND OUT DATE: 10 OCTOBER 2022

HAND IN DATE: 28 NOVEMBER 2022

WEIGHTAGE: 50%

INSTRUCTIONS TO CANDIDATES:

1 **Submit your assignment at the administrative counter.**

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

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

4 **Cases of plagiarism will be penalized.**

5 **The assignment should be bound in an appropriate style (comb bound or stapled).**

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

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

Table of Contents

1.0 Introduction and assumptions

1.1 Introduction

According to Stevens (2022), data analytics is the **process** of preprocessing, analyzing, and generating useful insights from raw data. One crucial keyword here is: process. Anything that pose great meaning in this world, do not build up in one day. Instead, they require an extraordinarily long series of processes to build up from the ground. Take one of the greatest software – the iOS from Apple Inc. – for example, it had taken thousands of software engineers' years of time to make it such a success today. Data analytics is no exception – it takes a series of process in order to generate valuable insights and findings which can ultimately bring enormous values to the community.

Hence, this documentation will be process oriented. By process oriented, it means that the structure of this report will be mainly according to the main sub-processes of the entire data analytic process.

This report consists of several fundamental sections:

1.0 Introduction and Assumptions: Introduce readers to this documentation and provide any necessary assumptions.

2.0 Data Preparation: This section includes all the data preprocessing procedures that every data analytics project will go through before going into the actual analysis works.

3.0 Data Analysis: This section is where the real works kick in. All the analysis and findings will be outlined in this section.

4.0 Extra Features: This section explains all the advanced concepts, techniques, and programs that had been implemented in this project.

5.0 Conclusion: This section concludes the entire project. It will include a personal reflection report that discuss the lessons learnt, limitations that can be addressed in future, etc.

1.2 Assumptions

Several assumptions have been made before progressing with the data analytic process:

1. The "*Tenant Preferred*" column means the kind of tenants (families, bachelors, or both) that prefer that particular rental unit.
2. The data given are a real representation of the housing conditions in India.
3. Values of the "*Rent*" column are represented in Indian Rupee (INR). The symbol is ₹.

2.0 Data Preparation

2.1 Data Acquisition and Import

Data acquisition (DAQ) is the process of collecting and gathering data that can be further utilized for any purposes, mostly related to analysis and statistics creation (First San Francisco Partners, 2018). In other words, DAQ provides people with raw data. For this project, the initial dataset has been given and it is ready to be imported anytime.

```
data <- read.csv("House_Rent_Dataset.csv")
```

Figure 2.1.1: Import Data from CSV file

In order to import the data from the CSV file, we will use the *read.csv* function to help us do that. It needs at least 1 argument – the path that will lead the compiler to the CSV file (Figure 2.1.1).

2.2 Data Exploration

2.2.1 View Column Headings

Now that we have the data, the first thing to do would be to explore the headings of the dataset so that we have an idea what is the dataset about. This can be done by using the *names* function.

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

Figure 2.2.1: Retrieving the column headings

From the output, we can make 2 initial notes:

1. The dataset is mostly about the characteristics of houses.
2. There are a few columns about people – *Tenant.Preferred* and *Point.of.Contact.*
3. The dataset has 12 columns in total.

2.2.2 Looking into the Data

Next, we will take a quick look into the data to understand their types, characteristics, and structure.

```
head(data, 10)
```

Figure 2.2.2: Understanding first 10 rows of data

	Posted.On	BHK	Rent	Size	Floor	Area.Type	Area.Locality	City	Furnishing.Status
1	5/18/2022	2	10000	1100	Ground out of 2	Super Area	Bandel	Kolkata	Unfurnished
2	5/13/2022	2	20000	800	1 out of 3	Super Area	Phool Bagan, Kankurgachi	Kolkata	Semi-Furnished
3	5/16/2022	2	17000	1000	1 out of 3	Super Area	Salt Lake City Sector 2	Kolkata	Semi-Furnished
4	7/4/2022	2	10000	800	1 out of 2	Super Area	Dumdum Park	Kolkata	Unfurnished
5	5/9/2022	2	7500	850	1 out of 2	Carpet Area	South Dum Dum	Kolkata	Unfurnished
6	4/29/2022	2	7000	600	Ground out of 1	Super Area	Thakurpukur	Kolkata	Unfurnished
7	6/21/2022	2	10000	700	Ground out of 4	Super Area	Malancha	Kolkata	Unfurnished
8	6/21/2022	1	5000	250	1 out of 2	Super Area	Malancha	Kolkata	Unfurnished
9	6/7/2022	2	26000	800	1 out of 2	Carpet Area	Palm Avenue Kolkata, Ballygunge	Kolkata	Unfurnished
10	6/20/2022	2	10000	1000	1 out of 3	Carpet Area	Natunhat	Kolkata	Semi-Furnished

	Tenant.Preferred	Bathroom	Point.of.Contact
1	Bachelors/Family	2	Contact Owner
2	Bachelors/Family	1	Contact Owner
3	Bachelors/Family	1	Contact Owner
4	Bachelors/Family	1	Contact Owner
5	Bachelors	1	Contact Owner
6	Bachelors/Family	2	Contact Owner
7	Bachelors	2	Contact Agent
8	Bachelors	1	Contact Agent
9	Bachelors	2	Contact Agent
10	Bachelors/Family	2	Contact Owner

Figure 2.2.3: Output of the head function

From the output, we can see that:

1. There is a column of *date* data type – *Posted.On*.
2. There are 4 numeric columns – *BHK*, *Rent*, *Size*, and *Bathroom*.
3. There are 7 alphabetic columns – *Floor*, *Area.Type*, *Area.Locality*, *City*, *Furnishing.Status*, *Tenant.Preferred*, and *Point.of.Contact*.

This brings up a few exploratory questions:

- What are the unique values existing for each of the alphabetic columns?
- How are the numeric data distributed?

2.2.3 Studying Structure of the Dataset

To answer the abovementioned questions, we will take a look into the structure of the dataset.

```
summary(data$BHK)
summary(data$Rent)
summary(data$Size)
summary(data$Bathroom)
```

Figure 2.2.4: Make summary on numeric data

```
> summary(data$BHK)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   2.000   2.084   3.000   6.000
> summary(data$Rent)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1200   10000   16000   34993   33000 3500000
> summary(data$Size)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   10.0   550.0   850.0   967.5  1200.0  8000.0
> summary(data$Bathroom)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   2.000   1.966   2.000  10.000
```

Figure 2.2.5: Output of the code above

Based on the output, we can make a few observations:

1. The maximum value of *Rent* is very much different from the 3rd Quartile value. This indicates that **there might be outlier data point** which might be result of inaccurate data acquisition. We will further investigate on that later.
2. The smallest data point of the column *Size* is 10 square feet, which is **unrealistic** if the measurement is taken as the built-up / super built-up area of the house. This data point is required for further investigation as well.

The issues raised above will be taken care of during the Data Cleaning stage.

The columns of data type *character* will also be explored. We will make use of the *unique* function to understand their structure.

```
unique(data$Floor)
```

Figure 2.2.6: the unique function

```
[433] "Lower Basement out of 1"  "13 out of 17"             "1 out of 14"              "Upper Basement out of 2"
[437] "2 out of 14"              "24 out of 31"             "2 out of 32"              "2 out of 16"
[441] "9 out of 13"              "1 out of 11"              "6 out of 29"              "9 out of 9"
[445] "28 out of 31"             "1 out of 15"              "Ground out of 14"         "2 out of 11"
[449] "19 out of 31"             "1 out of 16"              "25 out of 32"             "11 out of 16"
[453] "11 out of 17"             "Upper Basement out of 3"  "4 out of 24"               "1 out of 19"
[457] "7 out of 30"              "16 out of 19"             "Upper Basement out of 5"  "Ground out of 13"
[461] "2 out of 25"              "23 out of 30"             "4 out of 30"              "13 out of 25"
[465] "23 out of 35"             "Ground out of 10"         "5 out of 34"              "20 out of 35"
[469] "1"                        "4 out of 31"              "4 out of 26"              "24 out of 33"
[473] "4 out of 17"              "1 out of 35"              "11 out of 35"             "Ground out of 15"
[477] "Ground out of 27"         "15 out of 30"             "12 out of 30"             "23 out of 34"
```

Figure 2.2.7: Output of the function above

From the output we can see that the *Floor* column has 480 different values. This type of data is very difficult to be analyzed. It requires a proper transformation in order for it to maximize its use. This is similar for the *Area Locality* column as well (Figure below).

```
unique(data$Area.Locality)
length(unique(data$Area.Locality))
```

Figure 2.2.8: imposing the unique function on Area Locality and find its length

```
 [988] "Bedarahalli, Magadi Road"
 [989] "Kodigehalli"
 [990] "Rukmini Nagar"
 [991] "Vibhutipura"
 [992] "Mallasandra, Jalahalli West"
 [993] "Hebbal Kempapura"
 [994] "Jp Nagar Phase 5, J P Nagar, Outer Ring Road"
 [995] "Kannur"
 [996] "Bhattarahalli"
 [997] "Raghuvanahalli-Bcmc Layout"
 [998] "Chandapura"
 [999] "Marutinagar,"
[1000] "Hoysala Nagar"
 [ reached getOption("max.print") -- omitted 1235 entries ]
> length(unique(data$Area.Locality))
[1] 2235
```

Figure 2.2.9: Output of the code in Figure 2.2.8

The *Area Locality* column has a total number of 2235 unique values. We will continue to explore other columns as well:

```
unique(data$Area.Type)
unique(data$City)
unique(data$Furnishing.Status)
unique(data$Tenant.Preferred)
unique(data$Point.of.Contact)
```

Figure 2.2.10: Find the unique values

```
> unique(data$Area.Type)
[1] "Super Area"  "Carpet Area" "Built Area"
> unique(data$City)
[1] "Kolkata"   "Mumbai"    "Bangalore" "Delhi"     "Chennai"   "Hyderabad"
> unique(data$Furnishing.Status)
[1] "Unfurnished"    "Semi-Furnished" "Furnished"
> unique(data$Tenant.Preferred)
[1] "Bachelors/Family" "Bachelors"        "Family"
> unique(data$Point.of.Contact)
[1] "Contact Owner"   "Contact Agent"   "Contact Builder"
```

Figure 2.2.11: Output of the code in Figure 2.2.10

Now we get to have an idea what values these columns contain:

Table 2.2.1: Observations made from the code output shown in Figure 2.2.11

Column Name	Observations and Explorations
Area Type	Carpet Area is the area that we can cover with a carpet. Whereas Built Area is carpet area plus area that are used by walls and balcony. Meanwhile, Super Area is area that is calculated by built area plus the common area of the residency facilities such as lobby, clubhouses, swimming pool, etc. (Mishra, 2022).
City	There are 6 cities of India involved in this dataset namely Kolkata, Delhi, Bangalore, Chennai, and Hyderabad.
Furnishing Status	There are 3 levels of furnishing status: Unfurnished, Semi-Furnished, and Furnished.
Tenant Preferred	The houses are preferred by either Bachelors, Family, or both.
Point of Contact	Tenants will be required to either contact the owner, agent, or builder of the house.

2.2.4 Higher-Level Overview

Now we will look into the bigger picture of the dataset.

```
> ncol(data)
[1] 12
> nrow(data)
[1] 4746
>
```

Figure 2.2.12: Find the number of column and row

From the output of the *ncol* and *nrow* functions, we get to know that there are 12 columns and 4746 rows / records. The size of this dataset is perfect for visualizations and statistical modelling, but maybe not so for the advanced data analytic techniques such as Machine Learning and Deep Learning. Most of the time, these techniques require a bigger dataset so that the trained model would be accurate enough to generate real business values.

```
> str(data)
'data.frame':   4746 obs. of  12 variables:
 $ Posted.On        : chr  "5/18/2022" "5/13/2022" "5/16/2022" "7/4/2022" ...
 $ BHK              : int  2 2 2 2 2 2 2 1 2 2 ...
 $ Rent             : int  10000 20000 17000 10000 7500 7000 10000 5000 26000 10000 ...
 $ Size             : int  1100 800 1000 800 850 600 700 250 800 1000 ...
 $ Floor            : chr  "Ground out of 2" "1 out of 3" "1 out of 3" "1 out of 2" ...
 $ Area.Type        : chr  "Super Area" "Super Area" "Super Area" "Super Area" ...
 $ Area.Locality    : chr  "Bandel" "Phool Bagan, Kankurgachi" "Salt Lake City Sector 2" "Dumdum Park" ...
 $ City             : chr  "Kolkata" "Kolkata" "Kolkata" "Kolkata" ...
 $ Furnishing.Status: chr  "Unfurnished" "Semi-Furnished" "Semi-Furnished" "Unfurnished" ...
 $ Tenant.Preferred : chr  "Bachelors/Family" "Bachelors/Family" "Bachelors/Family" "Bachelors/Family" ...
 $ Bathroom         : int  2 1 1 1 1 2 2 1 2 2 ...
 $ Point.of.Contact : chr  "Contact Owner" "Contact Owner" "Contact Owner" "Contact Owner" ...
```

Figure 2.2.13: Output of the str function

By using the *str* function (which stands for structure), we can see the data type of every column at once. The result is the same as the observation in the “Looking into the Data” section, except that the *Posted On* column is of character data type. This means it needs a transformation so that it would be presented in the date data type.

Now that the dataset has been explored and we have gained some knowledge about it, we will continue and progress to the Data Cleaning stage.

2.3 Data Cleaning

2.3.1 Checking and Handling Null Values

In the data cleaning stage, the first thing we need to do is to check if there are any null values in the dataset. Null values can greatly impact the final analytic results in the bad way.

```
colSums(is.na(data))
```

Figure 2.3.1: Get the number of null values in each column

```
colSums(is.na(data))
     Posted.On            BHK           Rent           Size          Floor     Area.Type
             0              0              0              0              0             0
 Area.Locality           City Furnishing.Status Tenant.Preferred      Bathroom Point.of.Contact
             0              0              0              0              0             0
```

Figure 2.3.2: Output of the code in Figure 2.3.1

From the output we can see that all the columns in the dataset have no null values. This means the dataset has been cleaned, which is great.

2.3.2 Detecting and Treating Outliers

Any unusual data points that will interfere with the data analytic process are known as outliers (Frost, n.d.). In the industry, there are several common approaches to detect and locate the outliers such as the Z-score method and Interquartile Range (IQR) method (John, 2020). The advantage of using IQR method is that it uses metrics like quartiles and median, which is much less sensitive to outliers compared to common metrics like mean and range. Hence, we will leverage on this method to help us locate the outliers for this project.

Previously, we found that the *Rent* column might has outlier. Hence, we will use the IQR method to locate and eliminate the outliers in the dataset. To use that method, we will first have to find the interquartile range, first quartile, and third quartile of the *Rent* column.

```
Q_rent <- quantile(data$Rent, probs=c(.25, .75))
iqr_rent <- IQR(data$Rent)
```

Figure 2.3.3: Finding the quartiles and interquartile range of Rent

After calculating the quartiles, we can apply them to the formula now. We are using **1.5** as the constant to multiply with the interquartile range because it will eliminate around 0.28% of the extreme values from the dataset, which is rational.

```
cleaned_dataset <- subset(data, data$Rent > (Q_rent[1] - 1.5*iqr_rent) & data$Rent < (Q_rent[2]+1.5*iqr_rent))
```

Figure 2.3.4: Eliminating the outliers

Now the dataset looks more normal (Figure 2.3.5).

```
summary(cleaned_dataset$Rent)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 1200    9500   15000   19286   25000   67000
```

Figure 2.3.5: Finding the summary of Rent

On the other hand, we had noticed that the *Size* column might contain outliers as well. Hence, we will locate and eliminate them also. However, we will use manual approach for this process because the size of a house can vary greatly and requires more careful care. First, we will locate the houses that has a size of 30 square feet or below.

```
cleaned_dataset[cleaned_dataset$Size < 30, ]
```

Figure 2.3.6: Code to find size below 30 square feet

```
> cleaned_dataset[cleaned_dataset$Size < 30, ]
     Posted.On BHK  Rent Size      Floor   Area.Type      Area.Locality      City Furnishing.Status Tenant.Preferred
117  5/30/2022   1  5000   20 2 out of 3  Super Area       Dakshineswar   Kolkata       Unfurnished Bachelors/Family
2461 5/18/2022   1  5000   25 2 out of 5 Carpet Area        Ganga Vihar     Delhi       Unfurnished Bachelors/Family
2914  6/4/2022   1  3500   25 3 out of 4  Super Area             Burari     Delhi       Unfurnished Bachelors/Family
2974 6/16/2022   1  6000   25 4 out of 6  Super Area  Hari Nagar Ashram     Delhi    Semi-Furnished Bachelors/Family
3923 5/31/2022   3 23000   25 5 out of 6 Carpet Area Sanath Nagar, NH 9 Hyderabad    Semi-Furnished Bachelors/Family
4654 6/29/2022   3 15000   10 1 out of 1 Carpet Area  Ghatkesar, NH 2 2 Hyderabad         Furnished Bachelors/Family
     Bathroom Point.of.Contact
117         1    Contact Owner
2461        1    Contact Owner
2914        1    Contact Owner
2974        1    Contact Owner
3923        3    Contact Owner
4654        3    Contact Owner
```

Figure 2.3.7: Output of the code in Figure 2.3.6

Based on the output, we can observe that there are 2 houses with unreasonably small size despite being measured as carpet area, while having very high rent – row 3923 and 4654. They are obviously outliers and should be eliminated from the dataset.

To remove these 2 records, we will make use of the *subset* syntax (Figure below).

```
cleaned_dataset <- subset(cleaned_dataset, ! cleaned_dataset$Size == 10)
cleaned_dataset <- subset(cleaned_dataset, ! (cleaned_dataset$Size == 25 & cleaned_dataset$Rent == 23000))
```

Figure 2.3.8: Removing the 2 records

Now the dataset has been cleaned. We can move on to the data transformation stage.

2.4 Data Transformation

To enhance the current dataset we have, we can leverage data transformation techniques such as creating new columns, splitting existing columns, etc.

2.4.1 Renaming Data

The first transformation we will do to our dataset is to rename the *BHK* column so that it could have a more meaningful heading.

```
colnames(cleaned_dataset)[which(names(cleaned_dataset) == "BHK")] <- "Bedroom"
```

Figure 2.4.1: Renaming the BHK column to Bedroom

```
> names(cleaned_dataset)
 [1] "Posted.On"         "Bedroom"          "Rent"             "Size"              "Floor"
 [6] "Area.Type"         "Area.Locality"    "City"             "Furnishing.Status" "Tenant.Preferred"
[11] "Bathroom"          "Point.of.Contact"
```

Figure 2.4.2: Verifying the renaming operation

2.4.2 Splitting Data

The first piece of data we are going to split is the *Floor* column. It is actually consisting of the floor the house belongs to as well as the total floor of the entire building. They can have more value being separated. To split the data, we will need a R library called *stringr*.

```
install.packages("stringr")
library(stringr)
```

Figure 2.4.3: Installing and loading stringr

After that, we will utilize the *str_split_fixed* function to do the splitting work.

```
cleaned_dataset[c('Floor', 'Highest Floor')] <- str_split_fixed(cleaned_dataset$Floor, " out of ", 2)
```

Figure 2.4.4: Splitting Floor and add the result into the dataset

2.4.3 Replacing Values

In the *Floor* column, there are 2 redundant values that pose the exact same meaning – *Ground* and *1*. Hence, we will eliminate the redundancy to make the data consistent.

```
cleaned_dataset$Floor <- replace(cleaned_dataset$Floor, cleaned_dataset$Floor=="Ground", "1")
```

Figure 2.4.5: Replacing the value "Ground" with "1" in the Floor column

Then, we will use the unique function to verify this change (Figure below).

```
> unique(cleaned_dataset$Floor)
 [1] "1"              "2"              "4"              "3"              "5"              "7"
 [7] "8"              "Upper Basement" "11"             "Lower Basement" "6"              "14"
[13] "9"              "19"             "34"             "12"             "26"             "13"
[19] "16"             "10"             "18"             "20"             "17"             "15"
[25] "21"             "25"             "48"             "23"             "30"             "24"
[31] "28"
```

Figure 2.4.6: The value "Ground" has been eliminated successfully

2.4.4 Changing Data Type

There are several instances of inappropriate data type assignation within the dataset. One of them is the *Posted.On* column – it should be of date data type, but it belongs to the character data type now.

```
cleaned_dataset$Posted.On <- as.POSIXct(cleaned_dataset$Posted.On, format = "%m/%d/%Y")
```

Figure 2.4.7: Changing the column Posted.On to date (POSIXct) data type

Besides, the Highest.Floor column should also be of numeric data type, instead of the current character data type.

```
cleaned_dataset$Highest.Floor <- as.numeric(cleaned_dataset$Highest.Floor)
```

Figure 2.4.8: Changing the column Highest.Floor to numeric data type

We can verify the changes using the *class* function.

```
> class(cleaned_dataset$Posted.On)
[1] "POSIXct" "POSIXt"
> class(cleaned_dataset$Highest.Floor)
[1] "numeric"
```

Figure 2.4.9: Verifying the data type modifications

2.4.5 Adding Relevant Data

In order to produce better results for the data analysis process, we can add some relevant and meaningful data to the original dataset. For our project, we will bring several additional data to our dataset – part of India, average housing price, and GDP of the cities.

Figure 2.4.10: Map of India

Based on the map of India (Figure 2.4.10), we can see that Delhi is located at Northern India, while Kolkata is at East, Mumbai being West, and the other 3 cities – Hyderabad, Bangalore, and Chennai are at Southern India. We will add this observed data to the dataset (Figure below).

```
cleaned_dataset$Part.of.India <- NA
cleaned_dataset$Part.of.India[cleaned_dataset$City=="Delhi"] <- "North"
cleaned_dataset$Part.of.India[cleaned_dataset$City=="Kolkata"] <- "East"
cleaned_dataset$Part.of.India[cleaned_dataset$City=="Mumbai"] <- "West"
cleaned_dataset$Part.of.India[cleaned_dataset$City=="Bangalore"] <- "South"
cleaned_dataset$Part.of.India[cleaned_dataset$City=="Chennai"] <- "South"
cleaned_dataset$Part.of.India[cleaned_dataset$City=="Hyderabad"] <- "South"
```

Figure 2.4.11: Adding new data – Part of India

The average housing price of cities in India, on the other hand, is also an interesting and meaningful data to be combined with our dataset. According to Press Trust of India (2022), the average housing price of the cities in India is as follow:

Table 2.4.1: Average Housing Price of Cities in India

City	**Price (INR per square feet)**
Delhi	7434
Kolkata	6362
Hyderabad	9218
Bangalore	7848
Mumbai	19677
Chennai	7129

This housing price will be added to the dataset accordingly.

```
cleaned_dataset$Average.Housing.Price <- NA
cleaned_dataset$Average.Housing.Price[cleaned_dataset$City=="Delhi"] <- 7434
cleaned_dataset$Average.Housing.Price[cleaned_dataset$City=="Kolkata"] <- 6362
cleaned_dataset$Average.Housing.Price[cleaned_dataset$City=="Mumbai"] <- 19677
cleaned_dataset$Average.Housing.Price[cleaned_dataset$City=="Bangalore"] <- 7848
cleaned_dataset$Average.Housing.Price[cleaned_dataset$City=="Chennai"] <- 7129
cleaned_dataset$Average.Housing.Price[cleaned_dataset$City=="Hyderabad"] <- 9218
```

Figure 2.4.12: Adding new data – Average Housing Price

Next, we will continue and add the GDP of the cities to the dataset as well. The data is being acquired from an article written by the 3rd biggest English newspaper publisher in India - The Times of India.

Table 2.4.2: GDP of cities in India (Tiwari, 2021)

City	**GDP in 2021 (Billion USD)**
Delhi	293.6
Kolkata	150.1
Hyderabad	75.2
Bangalore	110
Mumbai	310
Chennai	78.6

```
cleaned_dataset$GDP <- NA
cleaned_dataset$GDP[cleaned_dataset$City=="Delhi"] <- 293.6
cleaned_dataset$GDP[cleaned_dataset$City=="Kolkata"] <- 150.1
cleaned_dataset$GDP[cleaned_dataset$City=="Mumbai"] <- 310
cleaned_dataset$GDP[cleaned_dataset$City=="Bangalore"] <- 110
cleaned_dataset$GDP[cleaned_dataset$City=="Chennai"] <- 78.6
cleaned_dataset$GDP[cleaned_dataset$City=="Hyderabad"] <- 75.2
```

Figure 2.4.13: Adding new data – GDP of the cities

The population in each of the cities is considered as a relevant data point as well. Hence, it will be added. The data is provided by United Nation.

Table 2.4.3: Population of cities in India (United Nation, 2021)

City	Population in 2020	Population in 2021	Changes (%)
Delhi	30290936	31181377	2.94
Kolkata	14850066	14974073	0.84
Hyderabad	10004144	10268653	2.64
Bangalore	12326532	12764935	3.56
Mumbai	20411274	20667655	1.26
Chennai	10971108	11235018	2.41

```
cleaned_dataset$Population.2020 <- NA
cleaned_dataset$Population.2020[cleaned_dataset$City=="Delhi"] <- 30290936
cleaned_dataset$Population.2020[cleaned_dataset$City=="Kolkata"] <- 14850066
cleaned_dataset$Population.2020[cleaned_dataset$City=="Mumbai"] <- 20411274
cleaned_dataset$Population.2020[cleaned_dataset$City=="Bangalore"] <- 12326532
cleaned_dataset$Population.2020[cleaned_dataset$City=="Chennai"] <- 10971108
cleaned_dataset$Population.2020[cleaned_dataset$City=="Hyderabad"] <- 10004144

cleaned_dataset$Population.2021 <- NA
cleaned_dataset$Population.2021[cleaned_dataset$City=="Delhi"] <- 31181377
cleaned_dataset$Population.2021[cleaned_dataset$City=="Kolkata"] <- 14974073
cleaned_dataset$Population.2021[cleaned_dataset$City=="Mumbai"] <- 20667655
cleaned_dataset$Population.2021[cleaned_dataset$City=="Bangalore"] <- 12764835
cleaned_dataset$Population.2021[cleaned_dataset$City=="Chennai"] <- 11235018
cleaned_dataset$Population.2021[cleaned_dataset$City=="Hyderabad"] <- 10268653

cleaned_dataset$Population.Changes <- NA
cleaned_dataset$Population.Changes[cleaned_dataset$City=="Delhi"] <- 2.94
cleaned_dataset$Population.Changes[cleaned_dataset$City=="Kolkata"] <- 0.84
cleaned_dataset$Population.Changes[cleaned_dataset$City=="Mumbai"] <- 1.26
cleaned_dataset$Population.Changes[cleaned_dataset$City=="Bangalore"] <- 3.56
cleaned_dataset$Population.Changes[cleaned_dataset$City=="Chennai"] <- 2.41
cleaned_dataset$Population.Changes[cleaned_dataset$City=="Hyderabad"] <- 2.64
```

Figure 2.4.14: Adding new data – Population in 2020, population in 2021, and population changes

3.0 Data Analysis

Before proceeding, we will have to load the *ggplot2* library so that we can use it for data visualization throughout the project.

```
library(ggplot2)
```

Figure 3.0.1: Loading ggplot2

3.1 Question 1 – Property Market

Question: Which is the most profitable city to invest in for the property market?

This question comes from the standing of any individuals that are involved in investment of the property market. This may include businesspeople, investors, property owners, tenants, etc. This question intends to figure out which city or cities, out of the 6 which is involved in the dataset, are the most profitable city if one intends to invest in the property market. This question will provide enormous values for the respective stakeholders.

Analysis 1-1: Find out how the average housing price of cities in India looks like.

This analysis wishes to find out how the average housing price of India looks like. This data is important for the property owners because housing price means their cost of owning a property in India.

```
ggplot(cleaned_dataset, aes(x=City, y=Average.Housing.Price)) +
  geom_segment( aes(x=City, xend=City, y=0, yend=Average.Housing.Price), color="grey") +
  geom_point( color="blue", size=4) +
  theme_light() +
  theme(
    panel.grid.major.x = element_blank(),
    panel.border = element_blank(),
    axis.ticks.x = element_blank()
  ) +
  xlab("City") +
  ylab("Average Housing Price")
```

Figure 3.1.1: Visualizing average housing price of cities in India

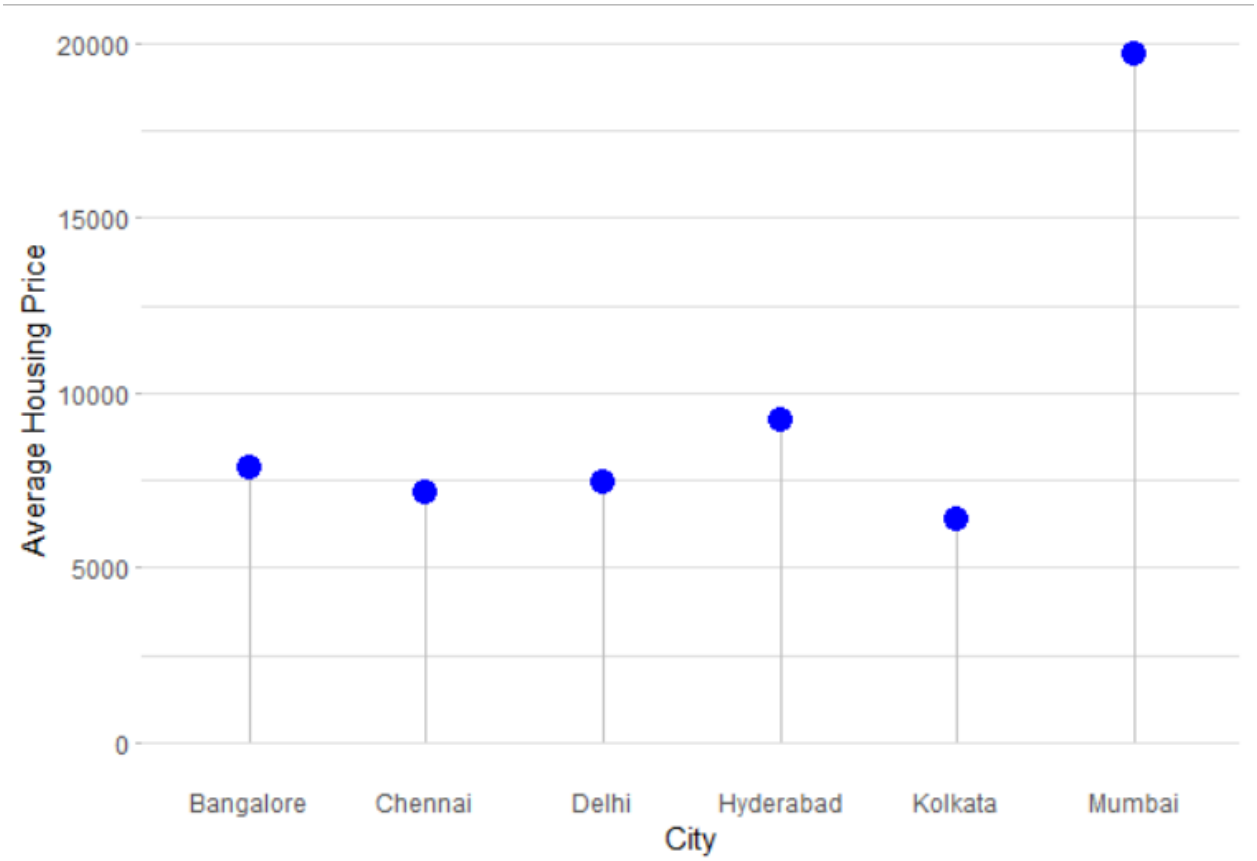


Figure 3.1.2: Plot for Analysis 1-1

From the lollipop plot above, we can see that Mumbai has the highest housing price in India. And it is much higher compared to all the other cities. Meanwhile, Kolkata has the lowest housing price, which is approximately **300%** lower compared to Mumbai.

Conclusion: Houses in Mumbai **might** be overvalued. We need to proceed with further analysis to figure out more about this.

Analysis 1-2: Find out how the housing price compared to rent in every city in India.

This analysis is to find out if there is any relationship between housing price and rent. This analysis might also determine if there is any undervaluation happening in the cities in India.

```
x <- c(rep(sort(unique(cleaned_dataset$City)), 2))
Data <- c(rep("Average Housing Price", 6), rep("Rent", 6))
price_rent <- cleaned_dataset %>% group_by(City) %>% summarise(price = mean(Average.Housing.Price), rent = mean(Rent))
values = c(pull(price_rent, price), pull(price_rent, rent))
data_temp <- data.frame(x,condition,values)

ggplot(data_temp, aes(fill=Data, y=values, x=x)) +
  geom_bar(position="dodge", stat="identity") +
  ggtitle("Housing Price compared to Rent") +
  xlab("") +
  ylab("INR")

rm(x,condition,price_rent, values, data_temp, Data)
```

Figure 3.1.3: Visualizing average housing price and rent of houses across cities

To create a grouped bar chart with *ggplot2*, we will need to create another data frame which is compatible to the aesthetics of the bar chart. During the process, we will make use of the *group_by* function and *summarise* function of the *dplyr* library.

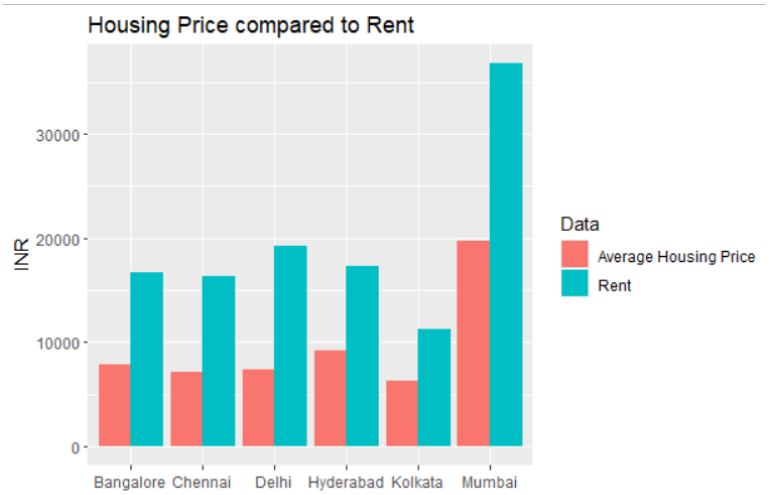


Figure 3.1.4: Plot for Analysis 1-2

Both the average housing price and rent in Mumbai are much higher compared to other cities. For other cities, the data is within a close range.

Conclusion: The living cost of Mumbai is much higher than other cities. Across the 6 cities, Kolkata offers the most affordable houses for property buyers.

Analysis 1-3: Find the relationship between furnishing status and rent in India's cities.

This analysis is to figure out if there is any relationship between furnishing status of the property and its rent. We will interpret every city in India separately.

```
x <- c(sort(rep(unique(cleaned_dataset$City), 3)))
Data <- c(rep(c("Unfurnished", "Semi-Furnished", "Furnished"), 6))
Data <- factor(Data, levels = c('Unfurnished', 'Semi-Furnished', 'Furnished'))
table <- cleaned_dataset %>%
  group_by(City, Furnishing.Status) %>%
  summarise(rent = mean(Rent)) %>%
  arrange(City, desc(Furnishing.Status))
values = pull(table, rent)
data_temp <- data.frame(x,Data,values)

ggplot(data_temp, aes(fill=Data, y=values, x=x)) +
  geom_bar(position="dodge", stat="identity") +
  ggtitle("Furnishing Status and Rent") +
  xlab("") +
  ylab("INR") +
  scale_fill_viridis(discrete = T, option = "E") +
  theme_ipsum() +
  theme(legend.position="bottom")

rm(x, table, values, data_temp, Data)
```

Figure 3.1.5: Visualizing furnishing status and rent

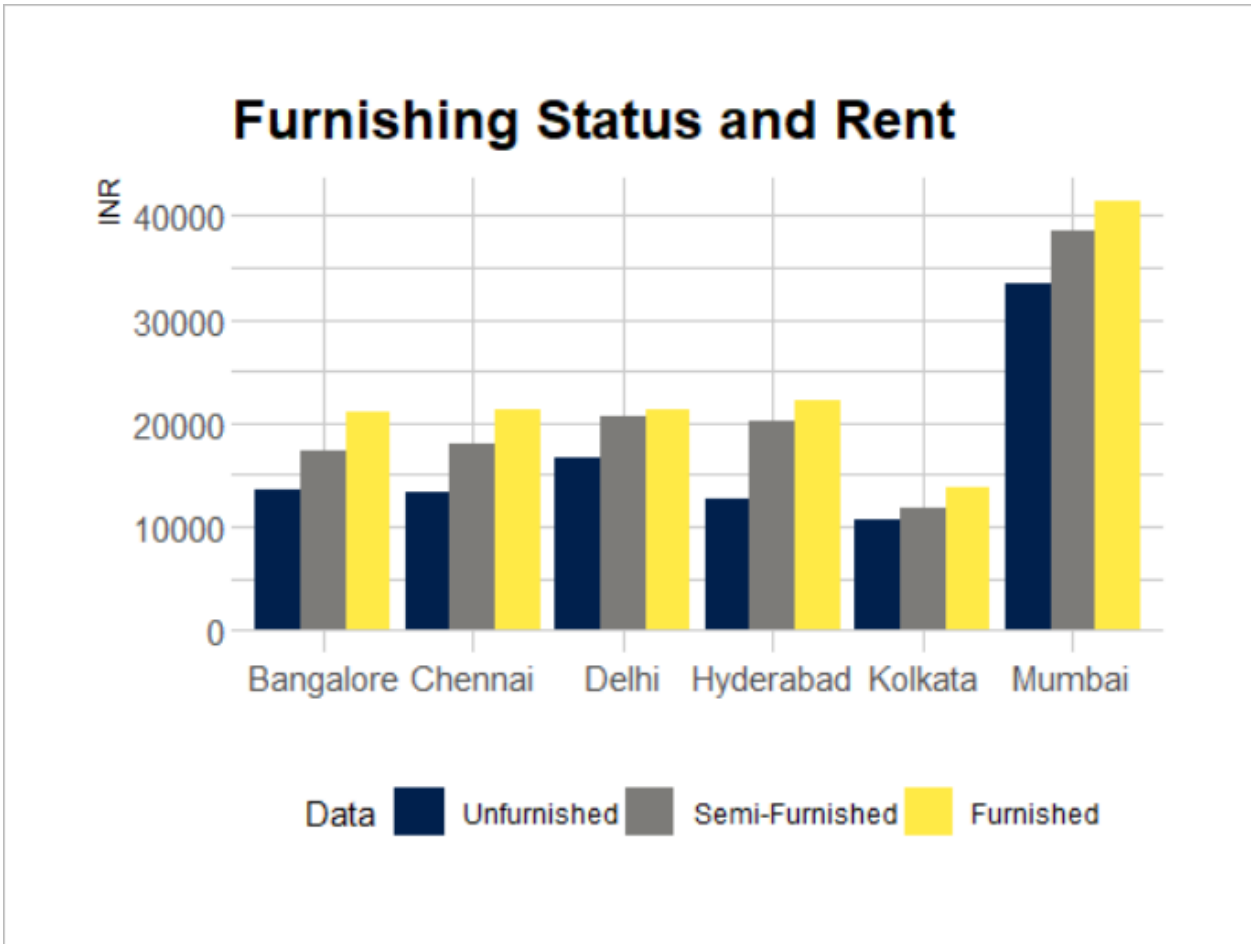


Figure 3.1.6: Plot for Analysis 1-3

From the bar chart above, we can observe a trend: the more the house is furnished, the higher the rent is. This makes sense because furnishing the house is a cost for the owner and it is an added value for the tenants. However, the differences of rent vary across the cities. Kolkata is the city with the least difference of rent for the 3 furnishing statuses.

Conclusion: Property owners at Kolkata are **not** suggested to furnish their house for the sake of increased rent.

Analysis 1-4: Find the usual ways tenant look for a spot at the city Bangalore.

This analysis intends to figure out what are the usual ways tenants look for a property in Bangalore. This analysis is important to act as an indicator which will determine the current conditions for the property market at Bangalore.

```
data_temp <- data.frame(
  category=sort(unique(cleaned_dataset$Point.of.Contact)),
  count=c(
    length(which(cleaned_dataset$Point.of.Contact == "Contact Agent" & cleaned_dataset$City == "Bangalore")),
    length(which(cleaned_dataset$Point.of.Contact == "Contact Builder" & cleaned_dataset$City == "Bangalore")),
    length(which(cleaned_dataset$Point.of.Contact == "Contact Owner" & cleaned_dataset$City == "Bangalore"))
  )
)

data_temp$fraction <- data_temp$count / sum(data_temp$count)
data_temp$ymax <- cumsum(data_temp$fraction)
data_temp$ymin <- c(0, head(data_temp$ymax, n=-1))
data_temp$label_position <- (data_temp$ymax + data_temp$ymin) / 2
data_temp$label <- paste0(data_temp$category, "\n value: ", data_temp$count)

ggplot(data_temp, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=category)) +
  geom_rect() +
  geom_text( x=1.5, aes(y=label_position, label=label, color=category), size=6) +
  scale_fill_brewer(palette="Dark2") +
  scale_color_brewer(palette="Dark2") +
  coord_polar(theta="y") +
  xlim(c(-1, 4)) +
  theme_void() +
  theme(legend.position = "none")
```

Figure 3.1.7: Visualizing point of contact data at Bangalore

We will plot a donut chart to visualize the point of contact data at Bangalore. To do this, we will need to compute the portion of each category, the maximum value, and the minimum value. Besides, we will also determine the position of the label.

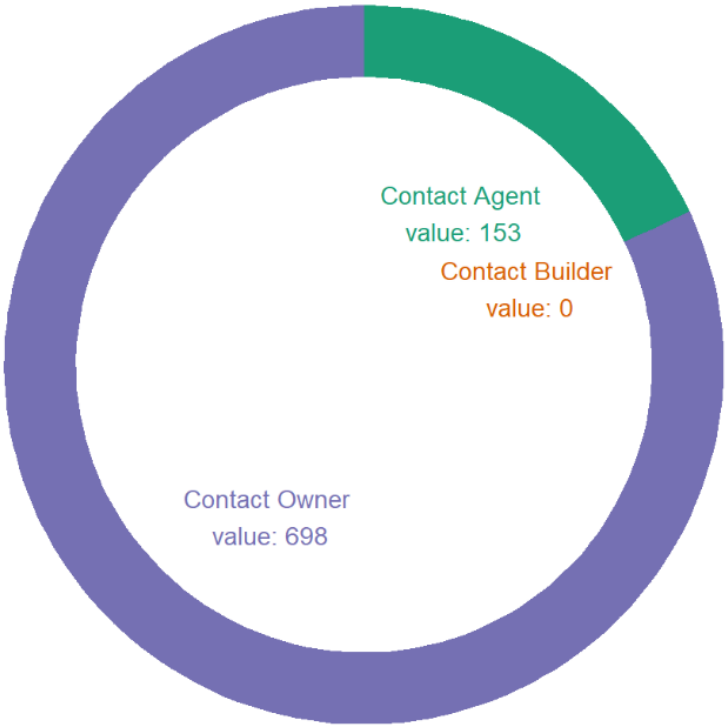


Figure 3.1.8: Plot for Analysis 1-4

At Bangalore, more than **75%** of the tenants directly contact with the owner of the houses that they have an interest in. A small number of tenants go through an agent, and nobody go to the house builder straightly.

Conclusion: Bangalore's properties are quite in-demand. Property owners do not rely too much on agents to help them approach tenants.

Analysis 1-5: Find the usual ways tenant look for a spot at the city Chennai.

This analysis intends to figure out what are the usual ways tenants look for a property in Chennai. This analysis is important to act as an indicator which will determine the current conditions for the property market at Chennai.

```
data_temp <- data.frame(
  category=sort(unique(cleaned_dataset$Point.of.Contact)),
  count=c(
    length(which(cleaned_dataset$Point.of.Contact == "Contact Agent" & cleaned_dataset$City == "Chennai")),
    length(which(cleaned_dataset$Point.of.Contact == "Contact Builder" & cleaned_dataset$City == "Chennai")),
    length(which(cleaned_dataset$Point.of.Contact == "Contact Owner" & cleaned_dataset$City == "Chennai"))
  )
)

data_temp$fraction <- data_temp$count / sum(data_temp$count)
data_temp$ymax <- cumsum(data_temp$fraction)
data_temp$ymin <- c(0, head(data_temp$ymax, n=-1))
data_temp$label_position <- (data_temp$ymax + data_temp$ymin) / 2
data_temp$label <- paste0(data_temp$category, "(", round(data_temp$fraction*100) , "%)")

ggplot(data_temp, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=category)) +
  geom_rect() +
  geom_text( x=1.5, aes(y=label_position, label=label, color=category), size=6) +
  scale_fill_brewer(palette="Blues") +
  scale_color_brewer(palette="Blues") +
  coord_polar(theta="y") +
  xlim(c(-1, 4)) +
  theme_void() +
  theme(legend.position = "none")
```

Figure 3.1.9: Visualizing point of contact data at Chennai

Similar to the previous analysis, we will make a donut plot for this analysis as well. However, we will compute the percentage of each of the portion instead of giving the values directly this time.

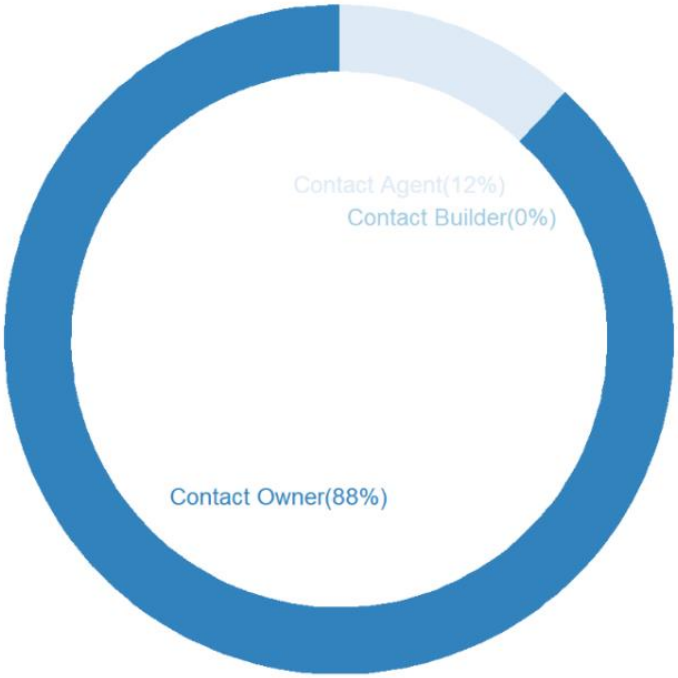


Figure 3.1.10: Plot for Analysis 1-5

For Chennai, the situation is more or less the same as Bangalore – majority **(88%)** of the rental transactions happen between owner and tenant. Agents play a small role of **12%** of the transactions. No one go to the house builder directly.

Conclusion: The property market at Chennai is healthy. Property owners do not rely on agents to approach tenants.

Analysis 1-6: Find the usual ways tenant look for a spot at the city Delhi.

```
data_temp <- data.frame(
  category=sort(unique(cleaned_dataset$Point.of.Contact)),
  count=c(
    length(which(cleaned_dataset$Point.of.Contact == "Contact Agent" & cleaned_dataset$City == "Delhi")),
    length(which(cleaned_dataset$Point.of.Contact == "Contact Builder" & cleaned_dataset$City == "Delhi")),
    length(which(cleaned_dataset$Point.of.Contact == "Contact Owner" & cleaned_dataset$City == "Delhi"))
  )
)

data_temp$fraction <- data_temp$count / sum(data_temp$count)
data_temp$ymax <- cumsum(data_temp$fraction)
data_temp$ymin <- c(0, head(data_temp$ymax, n=-1))
data_temp$label_position <- (data_temp$ymax + data_temp$ymin) / 2
data_temp$label <- paste0(data_temp$category, "(", round(data_temp$fraction*100) , "%)")

ggplot(data_temp, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=category)) +
  geom_rect() +
  geom_text( x=1.5, aes(y=label_position, label=label, color=category), size=6) +
  scale_fill_brewer(palette="Paired") +
  scale_color_brewer(palette="Paired") +
  coord_polar(theta="y") +
  xlim(c(-1, 4)) +
  theme_void() +
  theme(legend.position = "none")
```

Figure 3.1.11: Visualizing point of contact data at Delhi

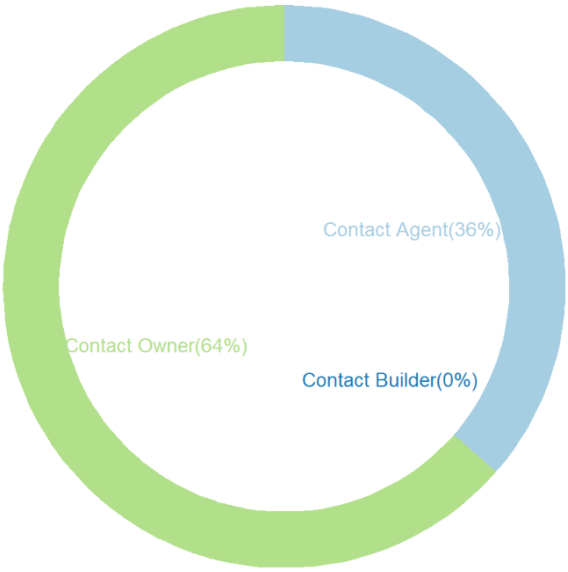


Figure 3.1.12: Plot for Analysis 1-6

Although most of the tenants (**64%**) at Delhi go to the house owner directly, the statistic is lower compared to Bangalore and Chennai. The other **36%** of tenants seek help from property agents.

Conclusion: The property market at Delhi is still within the healthy boundary. Most of the property owners do not rely on agents to approach tenants.

Analysis 1-7: Find the usual ways tenant look for a spot at the city Hyderabad.

```
data_temp <- data.frame(
  category=sort(unique(cleaned_dataset$Point.of.Contact)),
  count=c(
    length(which(cleaned_dataset$Point.of.Contact == "Contact Agent" & cleaned_dataset$City == "Hyderabad")),
    length(which(cleaned_dataset$Point.of.Contact == "Contact Builder" & cleaned_dataset$City == "Hyderabad")),
    length(which(cleaned_dataset$Point.of.Contact == "Contact Owner" & cleaned_dataset$City == "Hyderabad"))
  )
)

data_temp$fraction <- data_temp$count / sum(data_temp$count)
data_temp$ymax <- cumsum(data_temp$fraction)
data_temp$ymin <- c(0, head(data_temp$ymax, n=-1))
data_temp$label_position <- (data_temp$ymax + data_temp$ymin) / 2
data_temp$label <- paste0(data_temp$category, "(", round(data_temp$fraction*100) , "%)")

ggplot(data_temp, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=category)) +
  geom_rect() +
  geom_text( x=1.5, aes(y=label_position, label=label, color=category), size=6) +
  scale_fill_brewer(palette="BrBG") +
  scale_color_brewer(palette="BRBG") +
  coord_polar(theta="y") +
  xlim(c(-1, 4)) +
  theme_void() +
  theme(legend.position = "none")

rm(data_temp)
```

Figure 3.1.13: Visualizing point of contact data at Hyderabad

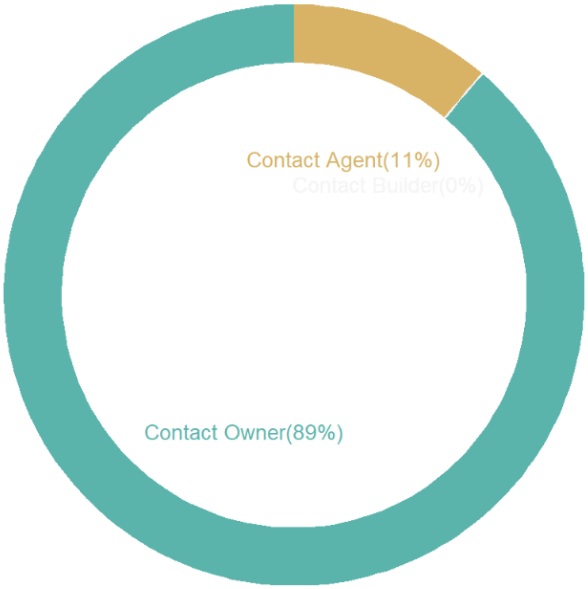


Figure 3.1.14: Plot for Analysis 1-7

In Hyderabad, almost all (**89%**) of the tenants contact the house owner directly and the other **11%** of tenants go through agents. No one contact the builder directly.

Conclusion: The property market at Hyderabad is healthy. The demand for rental agents in the city is not high. Hence, being a rental agent at Hyderabad might not be a good idea.

Analysis 1-8: Find the usual ways tenant look for a spot at the city Kolkata.

```
data_temp <- data.frame(
  category=sort(unique(cleaned_dataset$Point.of.Contact)),
  count=c(
    length(which(cleaned_dataset$Point.of.Contact == "Contact Agent" & cleaned_dataset$City == "Kolkata")),
    length(which(cleaned_dataset$Point.of.Contact == "Contact Builder" & cleaned_dataset$City == "Kolkata")),
    length(which(cleaned_dataset$Point.of.Contact == "Contact Owner" & cleaned_dataset$City == "Kolkata"))
  )
)

data_temp$fraction <- data_temp$count / sum(data_temp$count)
data_temp$ymax <- cumsum(data_temp$fraction)
data_temp$ymin <- c(0, head(data_temp$ymax, n=-1))
data_temp$label_position <- (data_temp$ymax + data_temp$ymin) / 2
data_temp$label <- paste0(data_temp$category, "(", round(data_temp$fraction*100) , "%)")

ggplot(data_temp, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=category)) +
  geom_rect() +
  geom_text( x=1.5, aes(y=label_position, label=label, color=category), size=6) +
  scale_fill_brewer(palette="RdGy") +
  scale_color_brewer(palette="RdGy") +
  coord_polar(theta="y") +
  xlim(c(-1, 4)) +
  theme_void() +
  theme(legend.position = "none")
```

Figure 3.1.15: Visualizing point of contact data at Kolkata

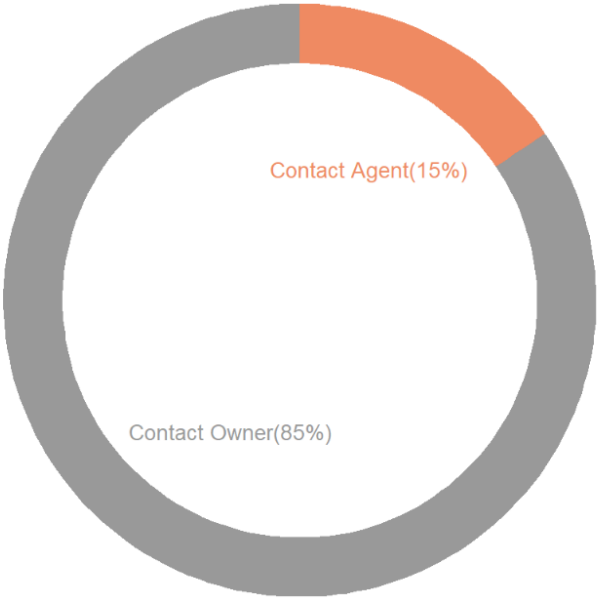


Figure 3.1.16: Plot for Analysis 1-8

Similar to other cities, most of the tenants (**85%**) in Kolkata contact the house owners directly. The other **15%** of tenants go though an agent.

Conclusion: The property market at Kolkata is healthy as well. The demand for rental agents in the city is not high.

Analysis 1-9: Find the usual ways tenant look for a spot at the city Mumbai.

```
data_temp <- data.frame(
  category=sort(unique(cleaned_dataset$Point.of.Contact)),
  count=c(
    length(which(cleaned_dataset$Point.of.Contact == "Contact Agent" & cleaned_dataset$City == "Mumbai")),
    length(which(cleaned_dataset$Point.of.Contact == "Contact Builder" & cleaned_dataset$City == "Mumbai")),
    length(which(cleaned_dataset$Point.of.Contact == "Contact Owner" & cleaned_dataset$City == "Mumbai"))
  )
)

data_temp$fraction <- data_temp$count / sum(data_temp$count)
data_temp$ymax <- cumsum(data_temp$fraction)
data_temp$ymin <- c(0, head(data_temp$ymax, n=-1))
data_temp$label_position <- (data_temp$ymax + data_temp$ymin) / 2
data_temp$label <- paste0(data_temp$category, "(", round(data_temp$fraction*100) , "%)")

ggplot(data_temp, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=category)) +
  geom_rect() +
  geom_text( x=1.5, aes(y=label_position, label=label, color=category), size=6) +
  scale_fill_brewer(palette="PiYG") +
  scale_color_brewer(palette="PiYG") +
  coord_polar(theta="y") +
  xlim(c(-1, 4)) +
  theme_void() +
  theme(legend.position = "none")
```

Figure 3.1.17: Visualizing point of contact data at Mumbai

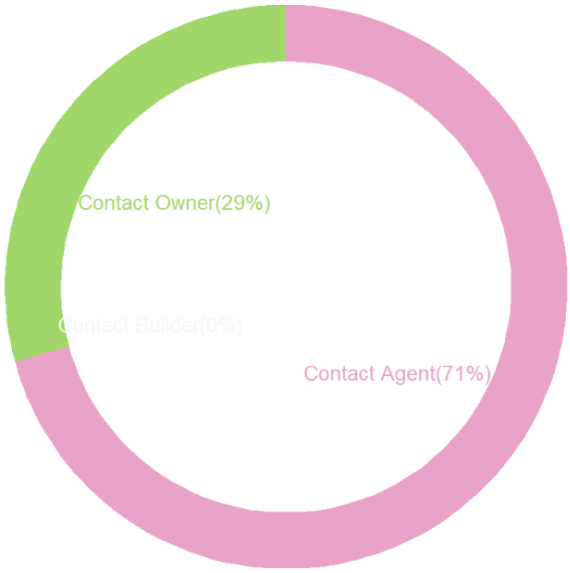


Figure 3.1.18: Plot for Analysis 1-9

As opposed to other cities in India, Mumbai shows a very different phenomenon whereby most of the tenants (**71%**) rely on rental agents to help them with rental unit search. The other **29%** of tenants contact the owner directly.

Conclusion: Tenants in Mumbai rely on agents very much. It is not a good thing for the tenants because the existence of a middleman, undoubtedly, will increase their rental cost. Being an agent in Mumbai would be a good occupation given the huge demand.

Findings for Analysis 1-4 to 1-9

After analysing how tenants approach their interested rental units in each of the cities in India, we can make a few observations / conclusions:

1. If one aspires to be a rental agent, go to the city Mumbai, there are much more opportunities and demand in the city.
2. Tenants in Hyderabad (89%) and Chennai (88%) usually go to the house owners directly. Hence, try not to be an agent in these 2 cities.
3. As a house owner or investor, Hyderabad and Chennai would be good choices given the low reliance on rental agents – it means bigger chance of profits.

Analysis 1-10: Find out how population in every city in India look like.

This analysis intends to find out what are the populations in each and every city in India. Population is a very important indicator for the property market at a particular place. This is because properties, ultimately, are to be occupied by the population. Hence, the bigger the population, the better the property market or the rental market of the place would be.

```
data_temp<- data.frame(
  city <- c(sort(unique(cleaned_dataset$City))),
  population <- c(
    mean(cleaned_dataset[cleaned_dataset$City == "Bangalore", ]$Population.2021),
    mean(cleaned_dataset[cleaned_dataset$City == "Chennai", ]$Population.2021),
    mean(cleaned_dataset[cleaned_dataset$City == "Delhi", ]$Population.2021),
    mean(cleaned_dataset[cleaned_dataset$City == "Hyderabad", ]$Population.2021),
    mean(cleaned_dataset[cleaned_dataset$City == "Kolkata", ]$Population.2021),
    mean(cleaned_dataset[cleaned_dataset$City == "Mumbai", ]$Population.2021)
  )
)

ggplot(data_temp, aes(x=city, y=population/1000000)) +
  geom_segment( aes(x=city, xend=city, y=0, yend=population/1000000), color="skyblue") +
  geom_point( color="blue", size=4, alpha=0.6) +
  theme_light() +
  ylab("Population (in million)") +
  xlab("City") +
  coord_flip() +
  theme(
    panel.grid.major.y = element_blank(),
    panel.border = element_blank(),
    axis.ticks.y = element_blank()
  )
```

Figure 3.1.19: Visualizing population of every city in India

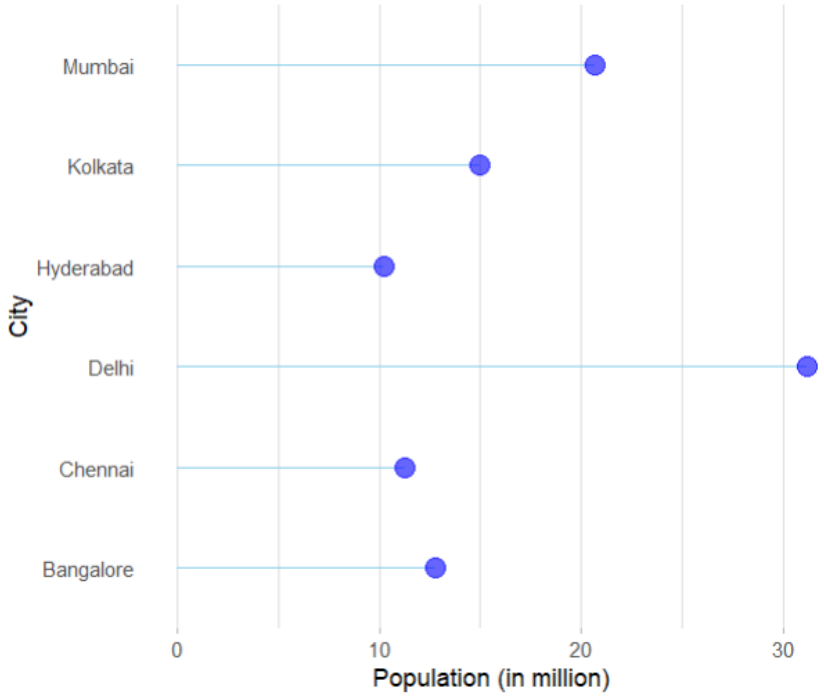


Figure 3.1.20: Plot for Analysis 1-10

From the lollipop chart we can observe that Delhi and Mumbai are the two most populated cities. On the other hand, Hyderabad is the least populated one among the 6 cities that involve in the analysis.

Conclusion: Delhi and Mumbai have the biggest demand for house rental.

Analysis 1-11: Find out the GDP per capita in every city in India.

This analysis wishes to figure out and visualize the GDP per capita of the people in every city in India. This analysis is important in a way that GDP per capita is one of the most important indicators for the purchasing power of the people in a particular area. Hence, the higher the GDP per capita of the city, the bigger the purchasing power of people, the better the property market or rental market of the city would be.

```
data_temp <- data.frame(
  city=c(sort(unique(cleaned_dataset$City))) ,
  gdp_per_capita <- c(
    mean(cleaned_dataset[cleaned_dataset$City == "Bangalore", ]$GDP) * 1000
    / (mean(cleaned_dataset[cleaned_dataset$City == "Bangalore", ]$Population.2021) / 1000000),
    mean(cleaned_dataset[cleaned_dataset$City == "Chennai", ]$GDP) * 1000
    / (mean(cleaned_dataset[cleaned_dataset$City == "Chennai", ]$Population.2021) / 1000000),
    mean(cleaned_dataset[cleaned_dataset$City == "Delhi", ]$GDP) * 1000
    / (mean(cleaned_dataset[cleaned_dataset$City == "Delhi", ]$Population.2021) / 1000000),
    mean(cleaned_dataset[cleaned_dataset$City == "Hyderabad", ]$GDP) * 1000
    / (mean(cleaned_dataset[cleaned_dataset$City == "Hyderabad", ]$Population.2021) / 1000000),
    mean(cleaned_dataset[cleaned_dataset$City == "Kolkata", ]$GDP) * 1000
    / (mean(cleaned_dataset[cleaned_dataset$City == "Kolkata", ]$Population.2021) / 1000000),
    mean(cleaned_dataset[cleaned_dataset$City == "Mumbai", ]$GDP) * 1000
    / (mean(cleaned_dataset[cleaned_dataset$City == "Mumbai", ]$Population.2021) / 1000000)
  )
)

ggplot(data_temp, aes(x=city, y=gdp_per_capita)) +
  geom_bar(stat = "identity", color="blue", fill=rgb(0.1,0.4,0.5,0.7)) +
  xlab("City") +
  ylab("GDP Per Capita")
```

Figure 3.1.21: Visualizing GDP per capita of every city in India

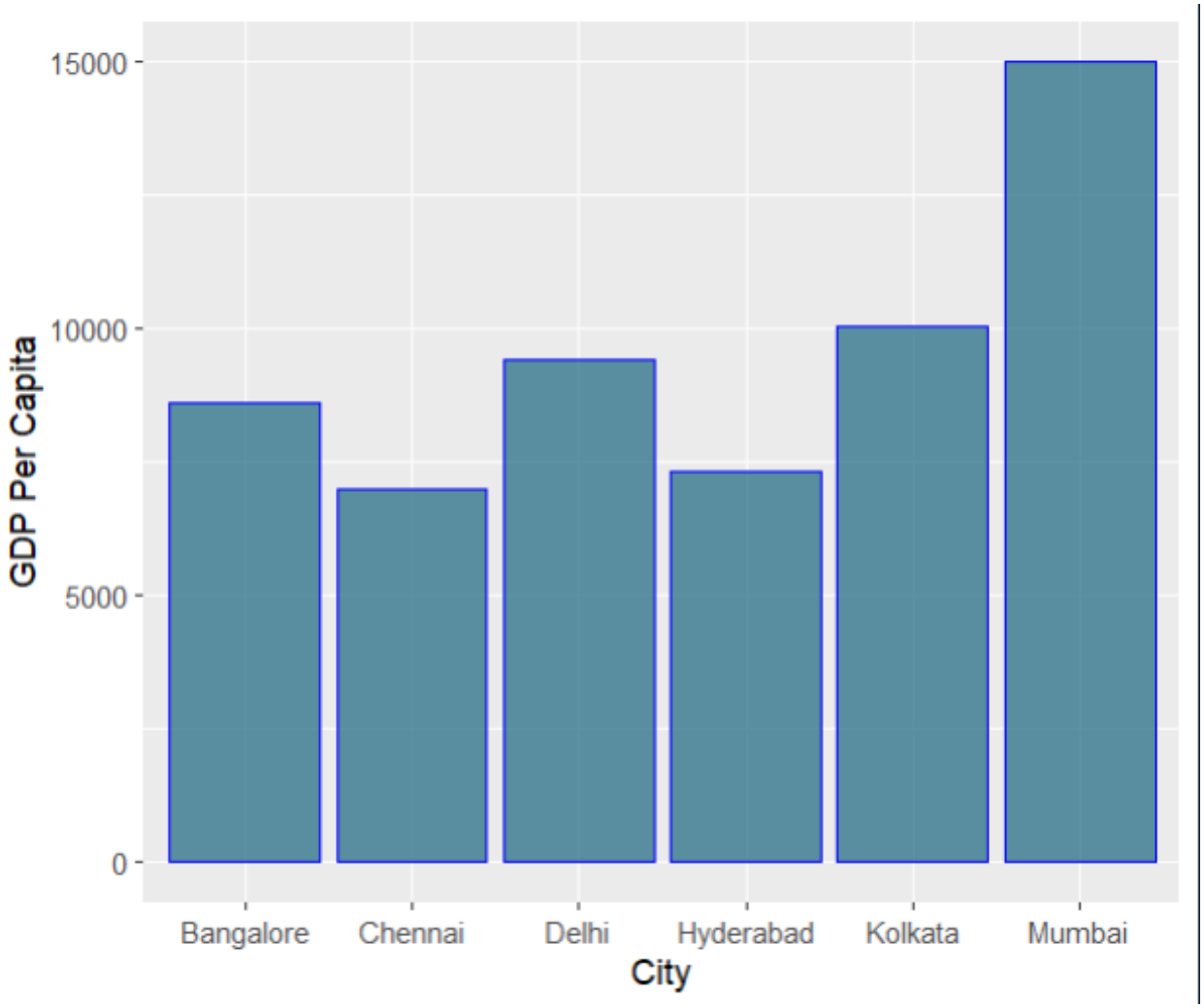


Figure 3.1.22: Plot for Analysis 1-11

Mumbai has the highest GDP per capita, followed by Kolkata and Delhi. Meanwhile, Chennai has the lowest GDP per capita.

Conclusion: People in Mumbai, Kolkata, and Delhi have higher purchasing power in general. Hence, people living in these cities can afford higher rent.

Analysis 1-12: Find the relationship between GDP per capita and rent in every city in India.

This analysis wishes to find out if there is any relationship between GDP per capita in a city and its average rent. This analysis is important to determine the level of undervaluation of rent in the cities in India. Besides, this also provide tenants a guideline that explains which city has more rent-worthy properties.

```
data_temp <- data.frame(
  city=c(sort(unique(cleaned_dataset$City))) ,
  gdp_per_capita <- c(
    mean(cleaned_dataset[cleaned_dataset$City == "Bangalore", ]$GDP) * 1000
    / (mean(cleaned_dataset[cleaned_dataset$City == "Bangalore", ]$Population.2021) / 1000000),
    mean(cleaned_dataset[cleaned_dataset$City == "Chennai", ]$GDP) * 1000
    / (mean(cleaned_dataset[cleaned_dataset$City == "Chennai", ]$Population.2021) / 1000000),
    mean(cleaned_dataset[cleaned_dataset$City == "Delhi", ]$GDP) * 1000
    / (mean(cleaned_dataset[cleaned_dataset$City == "Delhi", ]$Population.2021) / 1000000),
    mean(cleaned_dataset[cleaned_dataset$City == "Hyderabad", ]$GDP) * 1000
    / (mean(cleaned_dataset[cleaned_dataset$City == "Hyderabad", ]$Population.2021) / 1000000),
    mean(cleaned_dataset[cleaned_dataset$City == "Kolkata", ]$GDP) * 1000
    / (mean(cleaned_dataset[cleaned_dataset$City == "Kolkata", ]$Population.2021) / 1000000),
    mean(cleaned_dataset[cleaned_dataset$City == "Mumbai", ]$GDP) * 1000
    / (mean(cleaned_dataset[cleaned_dataset$City == "Mumbai", ]$Population.2021) / 1000000)
  ),
  rent <- c(
    mean(cleaned_dataset[cleaned_dataset$City == "Bangalore", ]$Rent),
    mean(cleaned_dataset[cleaned_dataset$City == "Chennai", ]$Rent),
    mean(cleaned_dataset[cleaned_dataset$City == "Delhi", ]$Rent),
    mean(cleaned_dataset[cleaned_dataset$City == "Hyderabad", ]$Rent),
    mean(cleaned_dataset[cleaned_dataset$City == "Kolkata", ]$Rent),
    mean(cleaned_dataset[cleaned_dataset$City == "Mumbai", ]$Rent)
  )
)

ggplot(data_temp) +
  geom_bar(aes(x=city, y=gdp_per_capita), stat = "identity", color="blue", fill=rgb(0.1,0.4,0.5,0.7)) +
  geom_line(aes(x=city, y=rent), col="red", group = 1, size = 2) +
  xlab("City") +
  ylab("GDP Per Capita")
```

Figure 3.1.23: Visualizing relationship between GDP per capita and rent in every city in India

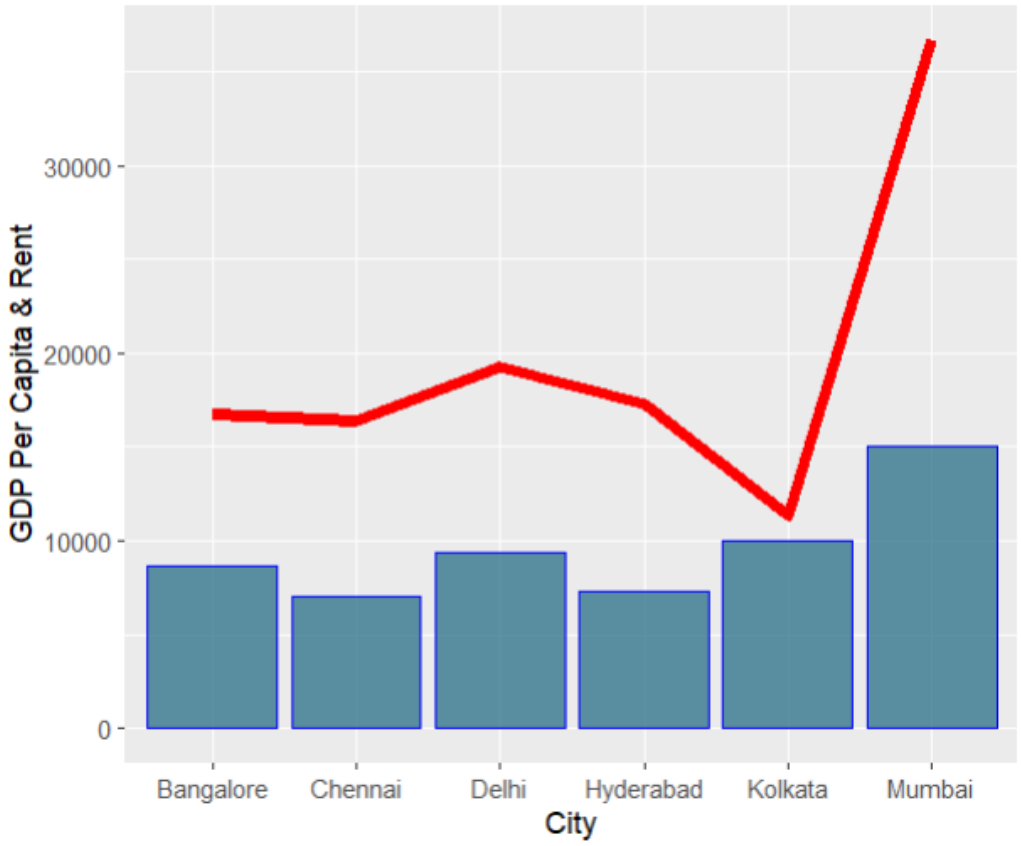


Figure 3.1.24: Plot for Analysis 1-12

From the plot, we can see that all data seem to be normal except for the city Kolkata and Mumbai. Kolkata, the city with second highest GDP per capita, has the lowest average rent across the cities. Furthermore, the difference of GDP per capita between Mumbai and Kolkata is around **50%**. However, their difference in rent is around **230%**.

Conclusion: From an investor's point of view, Kolkata is not a good choice to invest in given the undervalued rent. As a tenant, on the other hand, renting a house in Kolkata will be a wise decision. Mumbai is a good spot for investors. Tenants in Mumbai are used to higher rent.

Analysis 1-13: Find out how the change of population in every city in India look like.

Change of population is a very important indicator for the future property market of a particular area. This is because change of population means the increased number of people of a certain area, which will be taking part in the property market in future. Hence, the bigger the change of population of a city, the better the property market of that city in the future will be.

```
data_temp <-data.frame(
  mean(cleaned_dataset[cleaned_dataset$City == "Bangalore", ]$Population.Changes),
  mean(cleaned_dataset[cleaned_dataset$City == "Chennai", ]$Population.Changes),
  mean(cleaned_dataset[cleaned_dataset$City == "Delhi", ]$Population.Changes),
  mean(cleaned_dataset[cleaned_dataset$City == "Hyderabad", ]$Population.Changes),
  mean(cleaned_dataset[cleaned_dataset$City == "Kolkata", ]$Population.Changes),
  mean(cleaned_dataset[cleaned_dataset$City == "Mumbai", ]$Population.Changes)
)
colnames(data_temp) <- c("Bangalore", "Chennai", "Delhi", "Hyderabad", "Kolkata", "Mumbai")
max <- max(cleaned_dataset$Population.Changes)

data_temp <- rbind(rep(max,6) , rep(0,6) , data_temp)

head(data_temp)

radarchart(data_temp)
```

Figure 3.1.25: Visualizing the population growth in every city in India

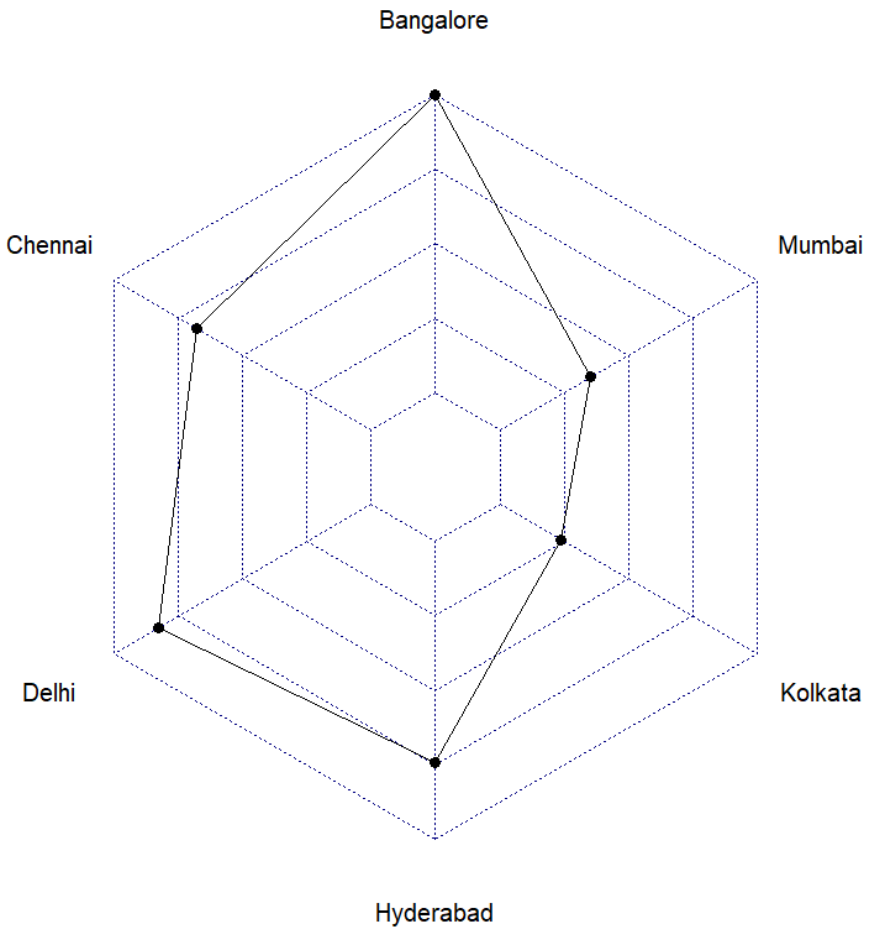


Figure 3.1.26: Plot for Analysis 1-13

From the radar plot in Figure 3.1.26 we can observe that the population growth in Bangalore is the highest among all cities. It is followed by Delhi and Hyderabad. The city with lowest population growth is Kolkata.

Conclusion: If this population growth trend continues for years or even decades, investing in the property market of Bangalore and Delhi will be a wise decision because population growth will boost rental demand. And if demand increase, the profitability of owning a property will increase as well.

Conclusion for Question 1

1. Mumbai has the highest average housing price, while Kolkata has the least.
2. The property as well as rental market in Mumbai is hot.
3. Investors that own properties in Kolkata are not suggested to furnish their unit for the sake of increased rent.
4. Rental transactions in Bangalore, Chennai, Delhi, Hyderabad, and Kolkata are mostly through the owner directly. Mumbai, on the other hand, shows different finding: most of the tenants in Mumbai rely on agents to find rental units.
5. There are much more opportunities in Mumbai to work as a rental agent.
6. Hyderabad and Chennai have the least reliance on agents. This means the property market in these cities is healthy, and hence do not need a middleman to facilitate transactions.
7. Delhi and Mumbai have the biggest population among all cities. Hence, these cities have bigger demand for rental properties.
8. People in Mumbai, Kolkata, and Delhi achieved the highest GDP per capita, hence the greater purchasing power they have. These people can afford higher rent.
9. Rent in Kolkata is undervalued. This is favorable for tenants but not the property owners.
10. Tenants in Mumbai are used to higher rent. The city is a good spot for investors.
11. The population growth in Bangalore and Delhi is the highest among all cities. Hence, owning a property in these cities will be a wise choice.

3.2 Question 2 – Housing Condition

Question: How is the housing condition in India?

This question is an important one. It is for any individuals who wish to know about the current housing condition in India. This might include the government authorities, international investors, businesspeople, tenants, etc. Housing condition which is good enough is necessary for the economic growth of any given country. Hence, the conclusion to this question will be involved in a lot of parties' decisions.

Analysis 2-1: Find the ratio of single-floor landed property against high-rise property in India.

This analysis aims to figure out the ratio of single-floor properties against high-rise properties in India. The finding to this analysis can answer questions such as "is India overdeveloped". Too many high-rise properties in a certain area means that the area is overdeveloped.

```
install.packages("waffle")
library(waffle)

# Vector
data_temp <- c(
  Landed = as.numeric(round(length(which(cleaned_dataset$Highest.Floor==1)) / nrow(cleaned_dataset) * 80)),
  Condo = as.numeric(round(length(which(cleaned_dataset$Highest.Floor!=1)) / nrow(cleaned_dataset) * 80))
  )

# Waffle chart
waffle(data_temp, rows = 8,
       legend_pos = "bottom")
```

Figure 3.2.1: Visualizing the ratio of landed property against high-rise property in India

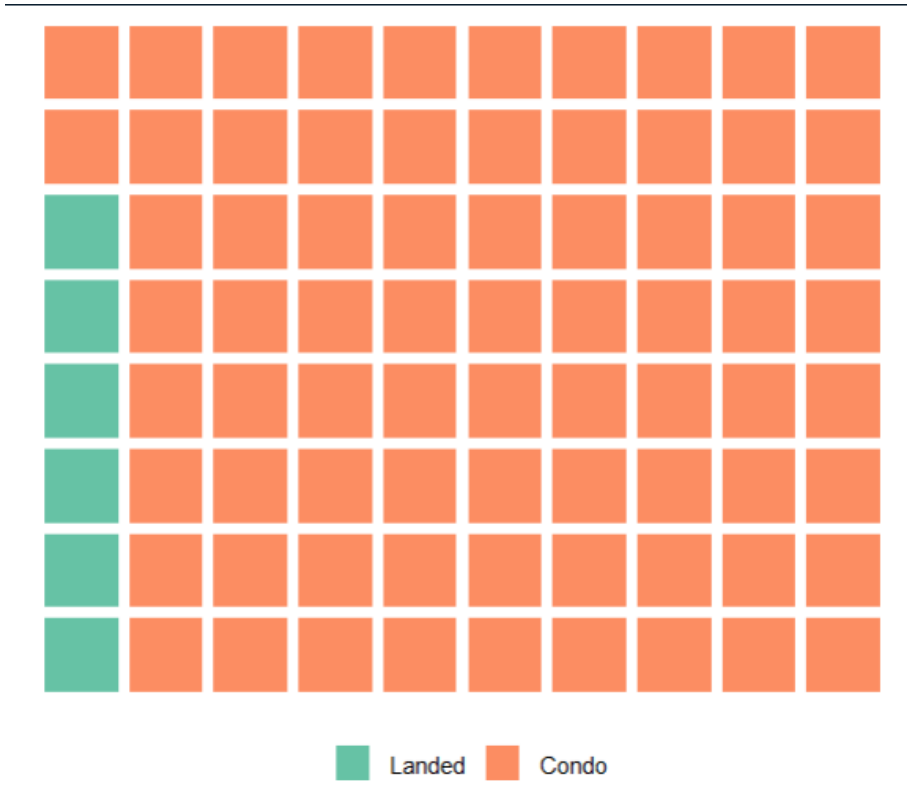


Figure 3.2.2: Plot for Analysis 2-1

From the waffle chart above we can observe that the ratio of landed properties against high-rise properties in India is quite imbalance – approximately **7.5%** of the properties is single-floor, and the other **92.5%** is high-rise properties.

Conclusion: Due to modernization, the population density in India is rising. Hence, a lot of high-rise properties are built instead of single-floor ones.

Analysis 2-2: Find the ratio of single-floor landed property against high-rise property in each city in India.

The previous analysis is about India as a whole. For this analysis, we will do the same thing, but the dataset will be grouped by city. This is to allow us to analyze the development level and modernization level of each city in India.

```
city <- c(rep("Bangalore", 2), rep("Chennai", 2), rep("Delhi", 2),
          rep("Hyderabad", 2), rep("Kolkata", 2), rep("Mumbai", 2))
condition <- rep(c("Landed" , "Condo") , 6)
value <- c(
  length(which(cleaned_dataset$Highest.Floor==1 & cleaned_dataset$City=="Bangalore")),
  length(which(cleaned_dataset$Highest.Floor!=1 & cleaned_dataset$City=="Bangalore")),
  length(which(cleaned_dataset$Highest.Floor==1 & cleaned_dataset$City=="Chennai")),
  length(which(cleaned_dataset$Highest.Floor!=1 & cleaned_dataset$City=="Chennai")),
  length(which(cleaned_dataset$Highest.Floor==1 & cleaned_dataset$City=="Delhi")),
  length(which(cleaned_dataset$Highest.Floor!=1 & cleaned_dataset$City=="Delhi")),
  length(which(cleaned_dataset$Highest.Floor==1 & cleaned_dataset$City=="Hyderabad")),
  length(which(cleaned_dataset$Highest.Floor!=1 & cleaned_dataset$City=="Hyderabad")),
  length(which(cleaned_dataset$Highest.Floor==1 & cleaned_dataset$City=="Kolkata")),
  length(which(cleaned_dataset$Highest.Floor!=1 & cleaned_dataset$City=="Kolkata")),
  length(which(cleaned_dataset$Highest.Floor==1 & cleaned_dataset$City=="Mumbai")),
  length(which(cleaned_dataset$Highest.Floor!=1 & cleaned_dataset$City=="Mumbai"))
)
data_temp <- data.frame(city,condition,value)

# Stacked + percent
ggplot(data_temp, aes(fill=condition, y=value, x=city)) +
  geom_bar(position="fill", stat="identity")
```

Figure 3.2.3: Visualizing the ratio of landed property against high-rise property in every city

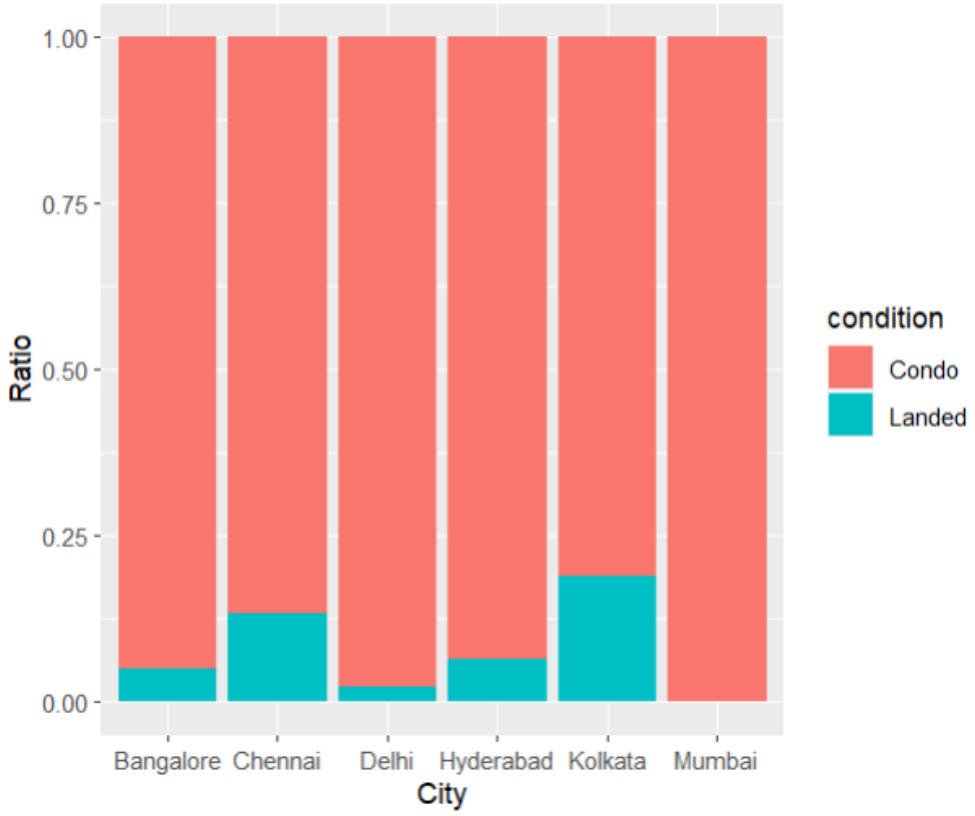


Figure 3.2.4: Plot for Analysis 2-2

From the percent stacked chart above we can see that properties in every city are mostly high-rise ones. Kolkata and Chennai are the two cities with the most landed properties among all. Mumbai, on the other hand, has almost no landed property.

Conclusion: Mumbai is the most modernized city among all. Meanwhile, Kolkata and Chennai are the least modernized cities.

Analysis 2-3: Find out how the height of properties in India look like.

This analysis is to find out how are the heights of property in India as a whole. This analysis aims to help property builders and developers determine which kind of property may be in more demand in India.

```
library(tidyverse)
library(hrbrthemes)

p <- cleaned_dataset %>%
  ggplot( aes(x=Highest.Floor)) +
  geom_histogram( fill="#69b3a2", color="#e9ecef", alpha=0.9, binwidth = 1) +
  ggtitle("Height of Properties") +
  theme_ipsum() +
  xlab("Number of floor") +
  ylab("Number of property") +
  theme(
    plot.title = element_text(size=15)
  )
p
```

Figure 3.2.5: Visualizing height of properties in India

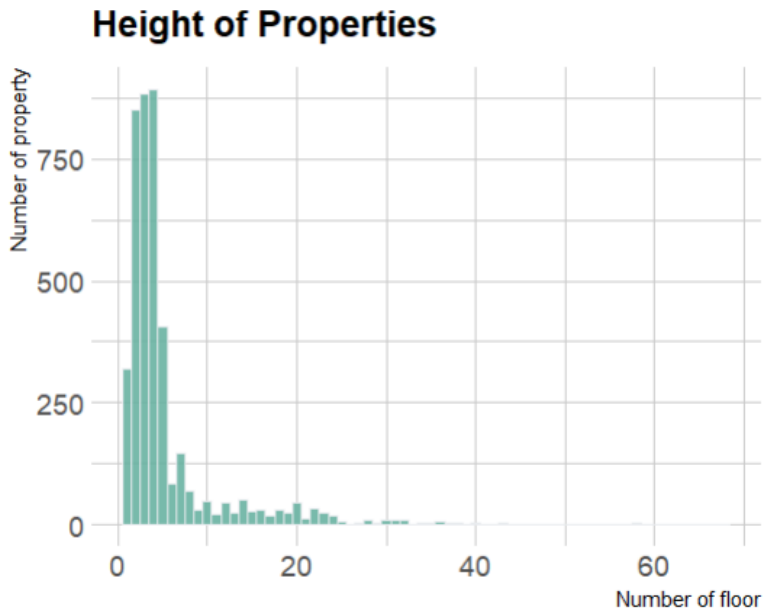


Figure 3.2.6: Plot for Analysis 2-3

From the histogram in Figure 3.2.6 we can see that most of the properties in India are within 2 to 4 floors. The highest property in the dataset is between 60 to 70 floors.

Conclusion: India has many double-storey housing as well as low-rise apartments.

Analysis 2-4: Find out how the height of properties in Bangalore look like.

In the previous analysis, we analyzed the height of properties in India as a whole. For this analysis, we will analyze the height of properties in one of the cities – Bangalore. The purpose of this study is to find out what is the most common type of property in Bangalore.

```
p <- cleaned_dataset %>%
  filter( City == "Bangalore" ) %>%
  ggplot( aes(x=Highest.Floor)) +
  geom_histogram( fill="#0B1354", color="#e9ecef", alpha=0.9, binwidth = 1) +
  ggtitle("Height of Properties in Bangalore") +
  theme_ipsum() +
  xlab("Number of floor") +
  ylab("Number of property") +
  theme(
    plot.title = element_text(size=15)
  )
p
```

Figure 3.2.7: Visualizing height of properties in Bangalore

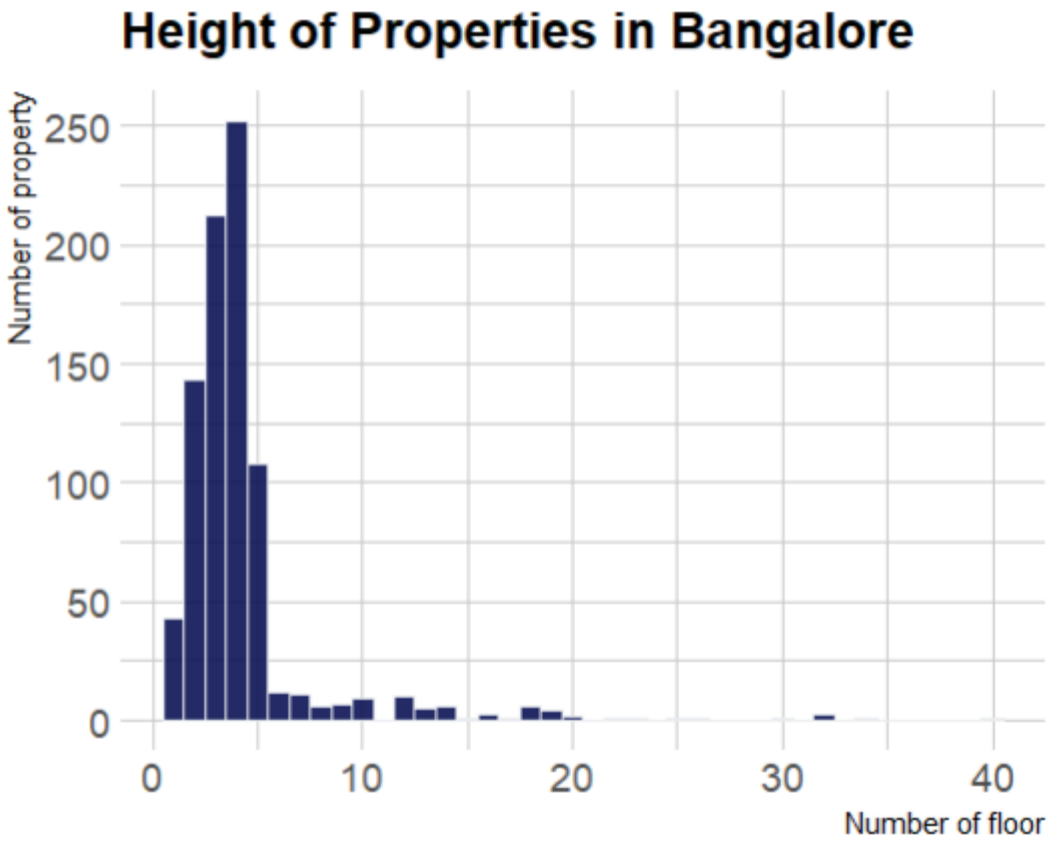


Figure 3.2.8: Plot for Analysis 2-4

Most of the properties in Bangalore has 4 floors. The highest property in Bangalore exceeds 40 floors.

Conclusion: The most common type of property in Bangalore is low-rise apartments which the height is mostly within 5 floors.

Analysis 2-5: Find out how the height of properties in Chennai look like.

This analysis is about the height of properties in another city in India – Chennai. The purpose of this analysis, again, is also to find out what is the most common type of property in Chennai.

```
p <- cleaned_dataset %>%
  filter( City == "Chennai" ) %>%
  ggplot( aes(x=Highest.Floor)) +
  geom_histogram( fill="#FCDC4C", color="#e9ecef", alpha=0.9, binwidth = 1) +
  ggtitle("Height of Properties in Chennai") +
  theme_ipsum() +
  xlab("Number of floor") +
  ylab("Number of property") +
  theme(
    plot.title = element_text(size=15)
  )
p
```

Figure 3.2.9: Visualizing height of properties in Chennai

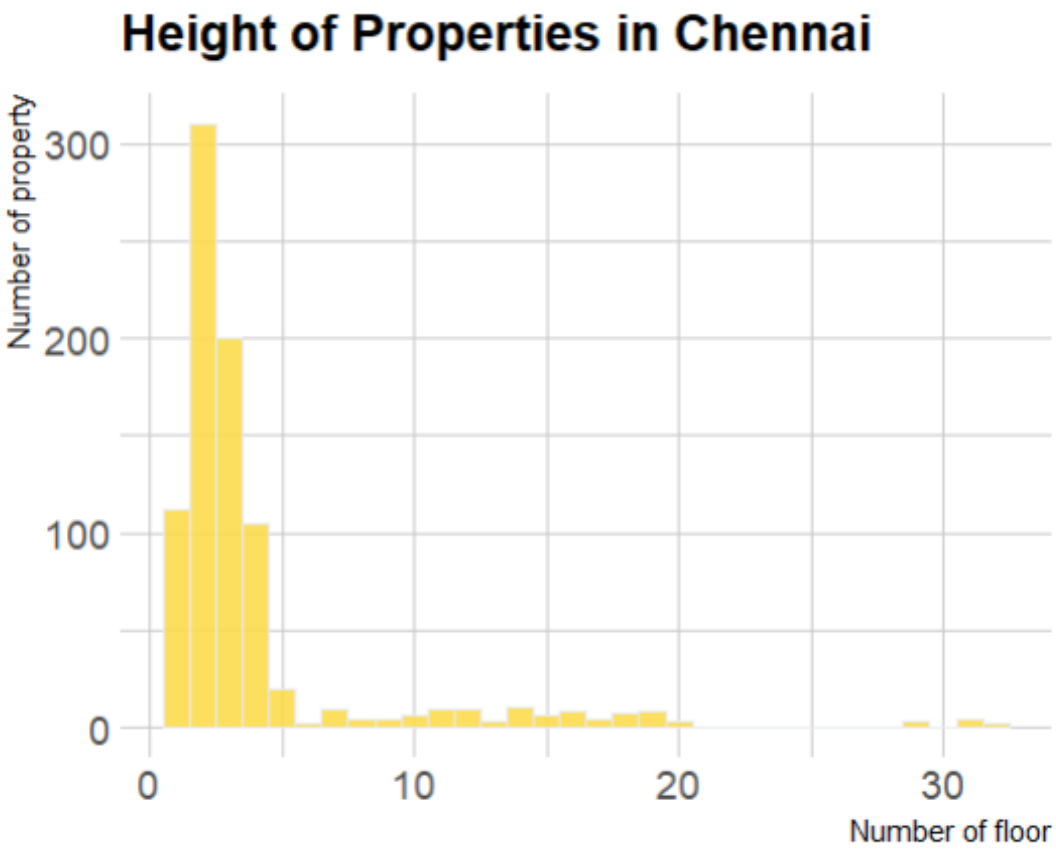


Figure 3.2.10: Plot for Analysis 2-5

Most of the properties in Chennai have 2 floors. The highest property in Chennai is around 35 floors.

Conclusion: The most common type of property in Chennai is double storey landed properties. Low-rise properties are also quite common in the city.

Analysis 2-6: Find out how the height of properties in Delhi look like.

This analysis is about the height of properties in another city in India – Delhi. The purpose of this analysis, again, is also to find out what is the most common type of property in Delhi.

```
p <- cleaned_dataset %>%
  filter( City == "Delhi" ) %>%
  ggplot( aes(x=Highest.Floor)) +
  geom_histogram( fill="#00C489", color="#e9ecef", alpha=0.9, binwidth = 1) +
  ggtitle("Height of Properties in Delhi") +
  theme_ipsum() +
  xlab("Number of floor") +
  ylab("Number of property") +
  theme(
    plot.title = element_text(size=15)
  )
p
```

Figure 3.2.11: Visualizing height of properties in Delhi

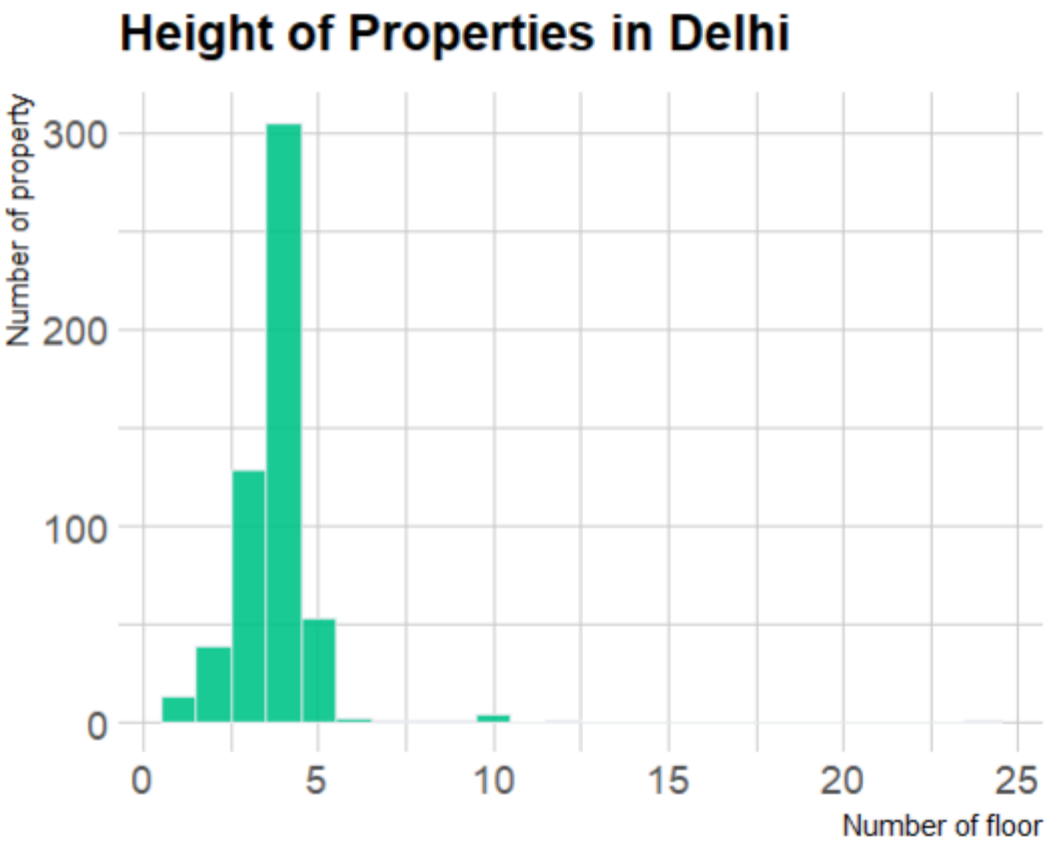


Figure 3.2.12: Plot for Analysis 2-6

Almost all the properties in Delhi are between 1 to 5 floors. 4-floor properties are the most common ones among all.

Conclusion: Properties in Delhi are mostly low-rise ones.

Analysis 2-7: Find out how the height of properties in Hyderabad look like.

This analysis is about the height of properties in another city in India – Hyderabad. The purpose of this analysis, again, is also to find out what is the most common type of property in Hyderabad.

```
p <- cleaned_dataset %>%
  filter( City == "Hyderabad" ) %>%
  ggplot( aes(x=Highest.Floor)) +
  geom_histogram( fill="#995757", color="#e9ecef", alpha=0.9, binwidth = 1) +
  ggtitle("Height of Properties in Hyderabad") +
  theme_ipsum() +
  xlab("Number of floor") +
  ylab("Number of property") +
  theme(
    plot.title = element_text(size=15)
  )
p
```

Figure 3.2.13: Visualizing height of properties in Hyderabad

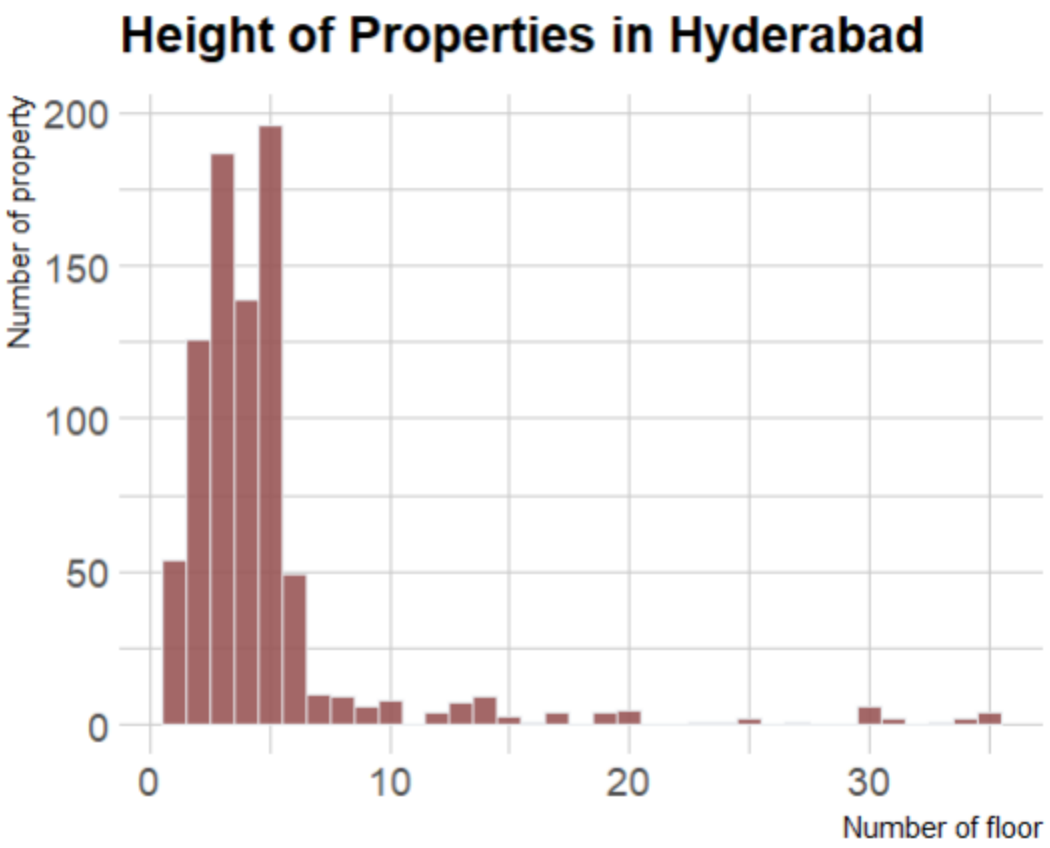


Figure 3.2.14: Plot for Analysis 2-7

Hyderabad shows quite a different finding: 3-floor and 5-floor properties are the 2 top majorities, followed by 4-floor and 2-floor properties. The highest property has around 35 floors.

Conclusion: Just like other cities, Hyderabad has a lot of low-rise properties.

Analysis 2-8: Find out how the height of properties in Kolkata look like.

This analysis is about the height of properties in another city in India – Kolkata. The purpose of this analysis, again, is also to find out what is the most common type of property in Kolkata.

```
p <- cleaned_dataset %>%
  filter( City == "Kolkata" ) %>%
  ggplot( aes(x=Highest.Floor)) +
  geom_histogram( fill="#5C5E6A", color="#e9ecef", alpha=0.9, binwidth = 1) +
  ggtitle("Height of Properties in Kolkata") +
  theme_ipsum() +
  xlab("Number of floor") +
  ylab("Number of property") +
  theme(
    plot.title = element_text(size=15)
  )
p
```

Figure 3.2.15: Visualizing height of properties in Kolkata

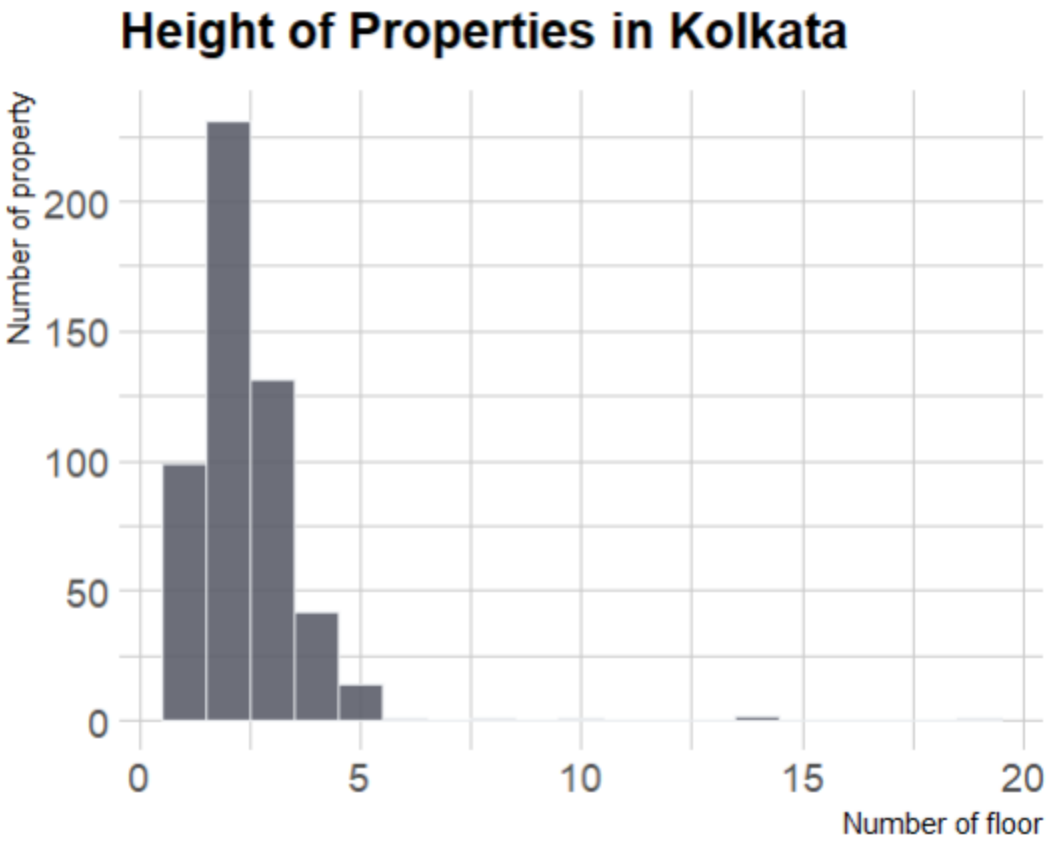


Figure 3.2.16: Plot for Analysis 2-8

The height of almost all properties in Kolkata are within 5 floors. 2-floor properties are the most common ones, followed by 3-floor and single-floor properties.

Conclusion: Properties in Kolkata have low height.

Analysis 2-9: Find out how the height of properties in Mumbai look like.

This analysis is about the height of properties in another city in India – Mumbai. The purpose of this analysis, again, is also to find out what is the most common type of property in Mumbai.

```
p <- cleaned_dataset %>%
  filter( City == "Kolkata" ) %>%
  ggplot( aes(x=Highest.Floor)) +
  geom_histogram( fill="#5C5E6A", color="#e9ecef", alpha=0.9, binwidth = 1) +
  ggtitle("Height of Properties in Kolkata") +
  theme_ipsum() +
  xlab("Number of floor") +
  ylab("Number of property") +
  theme(
    plot.title = element_text(size=15)
  )
p
```

Figure 3.2.17: Visualizing height of properties in Mumbai

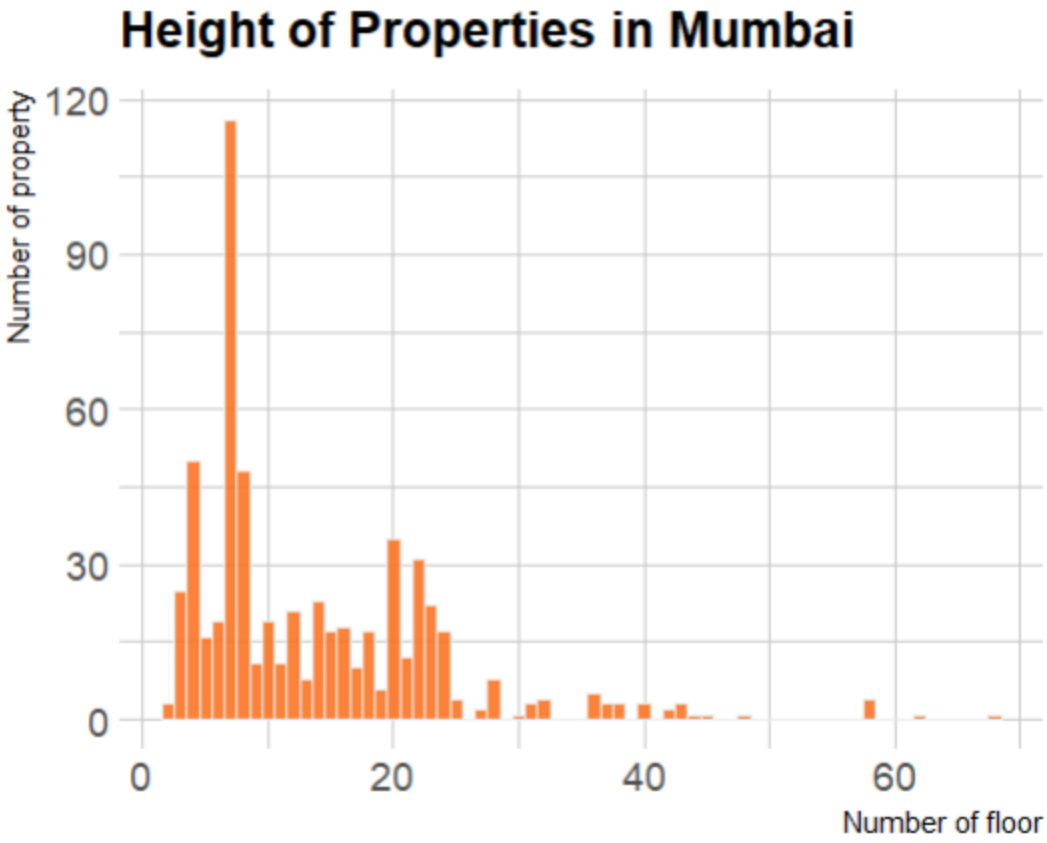


Figure 3.2.18: Plot for Analysis 2-8

The finding of the Mumbai city is an interesting one. The most common property in Mumbai is 6-floor low-rise apartments. Mumbai has also quite a number of properties that are within 10 to 25 floors. Moreover, the highest property in India is also located in Mumbai as well.

Conclusion: Mumbai has the most high-rise apartments compared to all the other cities. It is the most advanced city among all in terms of housing condition.

Analysis 2-10: Find the overall size of properties in India.

This analysis is to find out how is the size condition of the properties in India as a whole. This can help people to determine the most common range of size they can expect to have for their properties. Size is also a very important indicator of the housing condition in India.

```
data_temp <- data.frame(
  name=c(
    rep("Carpet Area" , length(which(cleaned_dataset$Area.Type == "Carpet Area"))),
    rep("Built Area" , length(which(cleaned_dataset$Area.Type == "Built Area"))),
    rep("Super Area" , length(which(cleaned_dataset$Area.Type == "Super Area")))
  ),
  value=c(
    cleaned_dataset[cleaned_dataset$Area.Type == "Carpet Area", ]$Size,
    cleaned_dataset[cleaned_dataset$Area.Type == "Built Area", ]$Size,
    cleaned_dataset[cleaned_dataset$Area.Type == "Super Area", ]$Size
  )
)

p <- ggplot(data_temp, aes(x=name, y=value, fill=name)) +
  geom_violin() +
  ggtitle("House Size") +
  xlab("Area Type") +
  ylab("Size in sq ft.")
p
```

Figure 3.2.19: Visualizing overall size of properties in India

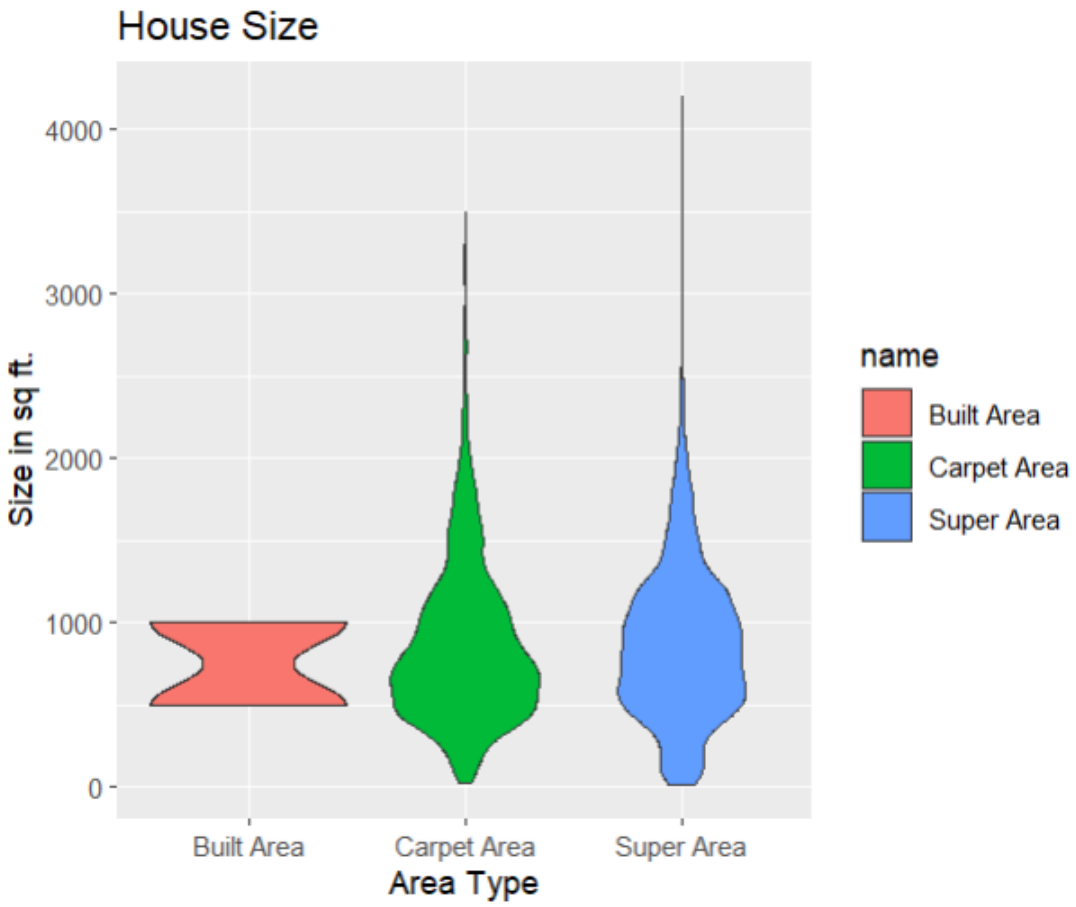


Figure 3.2.20: Plot for Analysis 2-10

As area types will influence the measurement of house size, we will separate the houses using area type and analyse accordingly. Firstly, the houses that are measured by built area are all between 500 to 1000 square feet in size. Unbelievably, it shows a perfectly symmetrical shape. It requires further analysis as this is abnormal. For properties measured by carpet area and super area, they show normal shape. Properties measured by carpet area mostly have a size of 450 to 1000 square feet. Meanwhile, properties measured by super area are mostly in between 500 to 1200 square feet. The biggest property measured in carpet area is around 3500 square feet in size, while for super area, it is approximately 4200 square feet.

Conclusion: The size of properties is generally bigger when it is measured by super area. This is because the super area's measurement includes much more things. It is calculated by a property's built area plus area of residency facilities. Refer to Table 2.2.1 for more information.

Analysis 2-11: Find out why built area of the properties show an utterly symmetrical shape in Analysis 2-10.

```
length(which(cleaned_dataset$Area.Type == "Built Area"))
```

Figure 3.2.21: Finding the number of records that are measured by built area

First of all, we will find how many records are there that are of "*built area*" area type.

```
[1] 2
```

Figure 3.2.22: Output of the code shown above

From the output we know that there are **only** 2 rows that use the "built area" area type. We will continue to see what the size of these 2 houses is.

```
min(cleaned_dataset[cleaned_dataset$Area.Type == "Built Area", ]$Size)
max(cleaned_dataset[cleaned_dataset$Area.Type == "Built Area", ]$Size)
```

Figure 3.2.23: Find the size of the properties that are measured by built area

```
> min(cleaned_dataset[cleaned_dataset$Area.Type == "Built Area", ]$Size)
[1] 500
> max(cleaned_dataset[cleaned_dataset$Area.Type == "Built Area", ]$Size)
[1] 1000
>
```

Figure 3.2.24: Output of the code above

The 2 values are 500 and 1000.

Conclusion: The reason of built area being symmetrical in the violin chart plotted for Analysis 2-10 is because there are only 2 values. We can make a conclusion that built area measurement is not a common way people measure houses in India.

Analysis 2-12: Find the overall size of properties in Bangalore.

In Analysis 2-10, we analyzed the size of properties in India as a whole. This analysis will continue and find the size of properties only in Bangalore. This will help people understand the housing condition in Bangalore, which might be helpful in their further analysis such as standard of living, average living space, etc.

```
data_temp <- cleaned_dataset[cleaned_dataset$City == "Bangalore", ]

ggplot(data_temp, aes(x=Area.Type, y=Size, fill=Area.Type)) +
  geom_boxplot(alpha=0.3) +
  theme(legend.position="none")
```

Figure 3.2.25: Visualizing size of properties in Bangalore

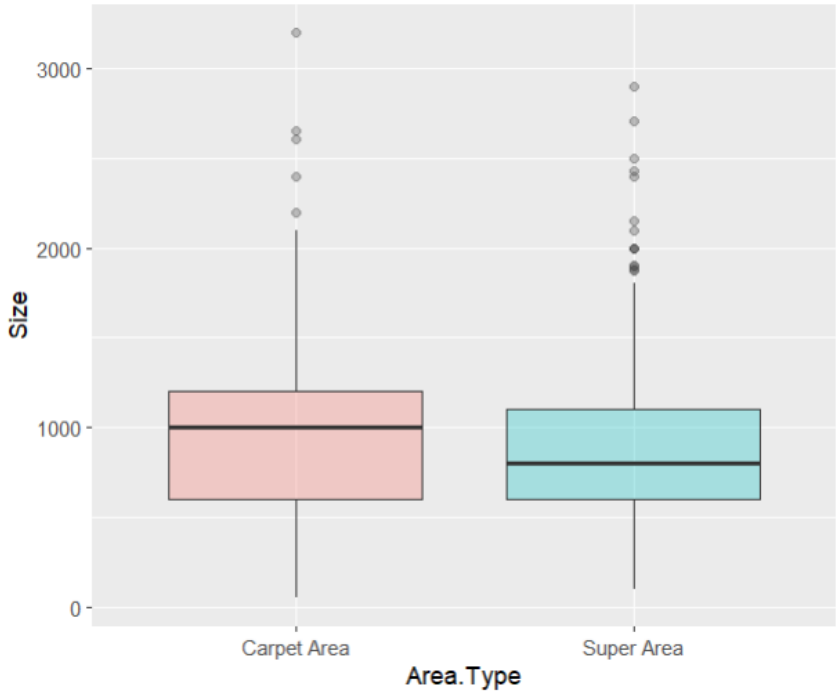


Figure 3.2.26: Plot for Analysis 2-12

From the output we can observe that the size of the properties being measured by carpet area is generally bigger than the ones being measured by super area, which is an abnormal case. The potential reason behind this might be the carpet area's data are mostly consisting of landed properties, which is bigger than high-rise properties. Besides, the median size of carpet area measurement is around 1000 square feet, while for super area, it is around 750 square feet.

Analysis 2-13: Find the overall size of properties in Chennai.

This analysis will continue and find the size of properties only in Chennai. This will help people understand the housing condition in Chennai, which might be helpful in their further analysis such as standard of living, average living space, etc.

```
data_temp <- cleaned_dataset[cleaned_dataset$City == "Chennai", ]

ggplot(data_temp, aes(x=Area.Type, y=Size, fill=Area.Type)) +
  geom_boxplot(alpha=0.3) +
  theme(legend.position="none")
```

Figure 3.2.27: Visualizing size of properties in Chennai

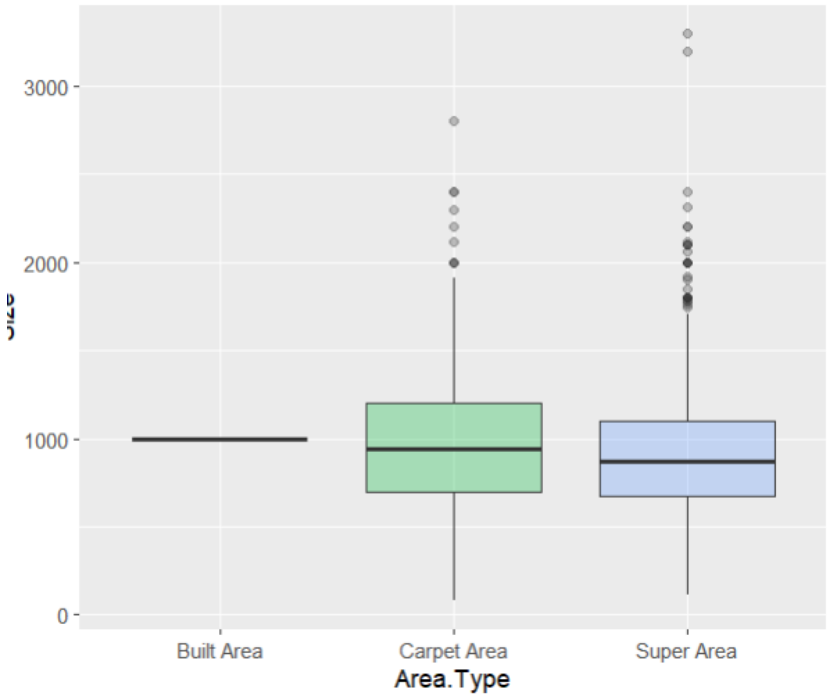


Figure 3.2.28: Plot for Analysis 2-13

From the output we can see that one of the two houses that are measured by built area is located in Chennai. Just like Bangalore, the data measured by super area is generally lower than carpet area as well. The median size measured by carpet area is around 950 square feet. Meanwhile, it is around 850 square feet for super area.

Analysis 2-14: the overall size of properties in Delhi.

This analysis will continue and find the size of properties only in Delhi. This will help people understand the housing condition in Delhi, which might be helpful in their further analysis such as standard of living, average living space, etc.

```
data_temp <- cleaned_dataset[cleaned_dataset$City == "Delhi", ]

ggplot(data_temp, aes(x=Area.Type, y=Size, fill=Area.Type)) +
  geom_boxplot(alpha=0.3) +
  theme(legend.position="none") +
  scale_fill_brewer(palette="BuPu")
```

Figure 3.2.29: Visualizing size of properties in Delhi

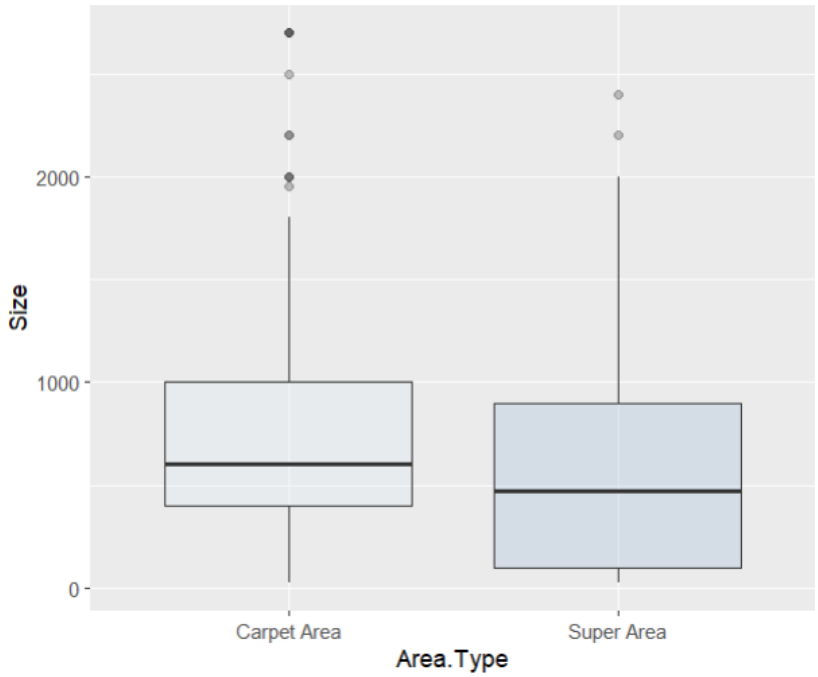


Figure 3.2.30: Plot for Analysis 2-14

From the box plot above we can see that houses that are measured by carpet area are still generally bigger in size compared to the ones measured by super area. The median size of properties measured by carpet area is approximately 600 square feet. On the other hand, the median size of properties measured by super area is around 500 square feet.

Analysis 2-15: the overall size of properties in Hyderabad.

This analysis will continue and find the size of properties only in Hyderabad. This will help people understand the housing condition in Hyderabad, which might be helpful in their further analysis such as standard of living, average living space, etc.

```
data_temp <- cleaned_dataset[cleaned_dataset$City == "Hyderabad", ]

ggplot(data_temp, aes(x=Area.Type, y=Size, fill=Area.Type)) +
  geom_boxplot(alpha=0.3) +
  theme(legend.position="none") +
  scale_fill_brewer(palette="PRGn")
```

Figure 3.2.31: Visualizing size of properties in Hyderabad

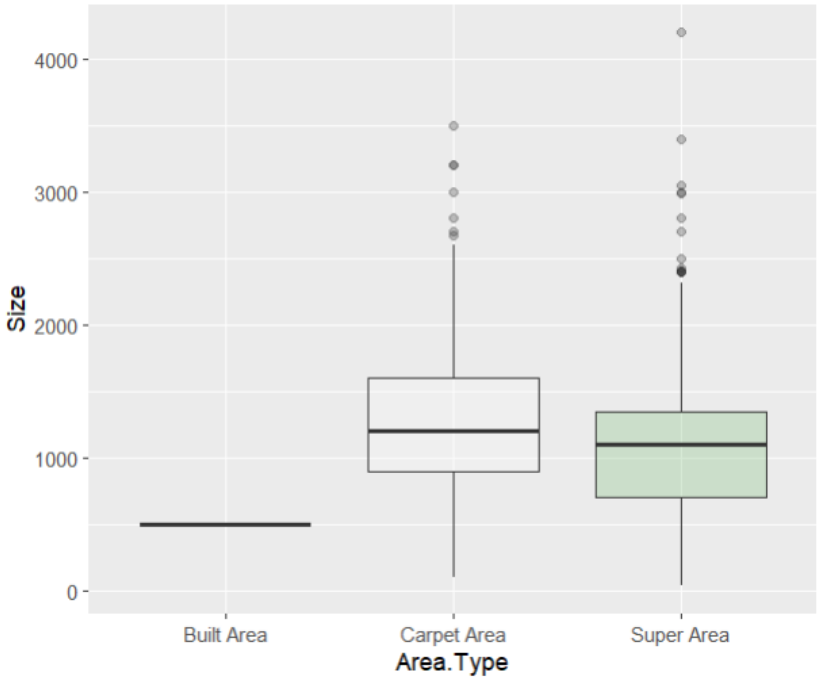


Figure 3.2.32: Plot for Analysis 2-15

For Hyderabad, the median size of properties being measured by carpet area is around 1200 square feet. The ones being measured by super area is around 1100 square feet.

Analysis 2-16: the overall size of properties in Kolkata.

This analysis will continue and find the size of properties only in Kolkata. This will help people understand the housing condition in Kolkata, which might be helpful in their further analysis such as standard of living, average living space, etc.

```
data_temp <- cleaned_dataset[cleaned_dataset$City == "Kolkata", ]

ggplot(data_temp, aes(x=Area.Type, y=Size, fill=Area.Type)) +
  geom_boxplot(alpha=0.3) +
  theme(legend.position="none") +
  scale_fill_brewer(palette="OrRd")
```

Figure 3.2.33: Visualizing size of properties in Kolkata

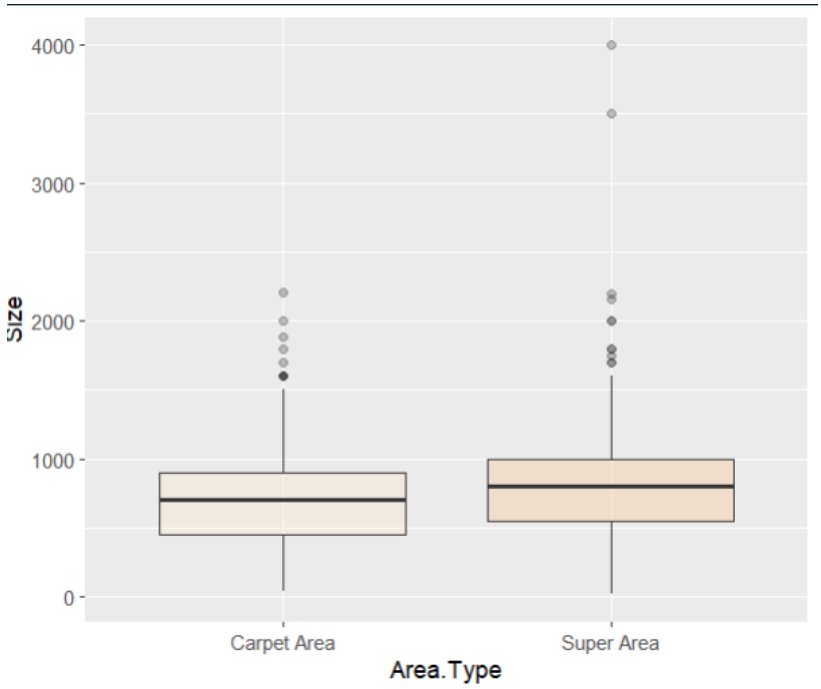


Figure 3.2.34: Plot for Analysis 2-16

The median size of properties measured by carpet area in Kolkata is around 750 square feet, and it is around 800 square feet for the properties that are measured by super area. Unlike other cities, the data of super area in Kolkata is generally higher than the data of carpet area.

Analysis 2-17: Find the overall size of properties in Mumbai.

This analysis will continue and find the size of properties only in Mumbai. This will help people understand the housing condition in Mumbai, which might be helpful in their further analysis such as standard of living, average living space, etc.

```
data_temp <- cleaned_dataset[cleaned_dataset$City == "Mumbai", ]

ggplot(data_temp, aes(x=Area.Type, y=Size, fill=Area.Type)) +
  geom_boxplot(alpha=0.3) +
  theme(legend.position="none") +
  scale_fill_brewer(palette="Set1")
```

Figure 3.2.35: Visualizing size of properties in Mumbai

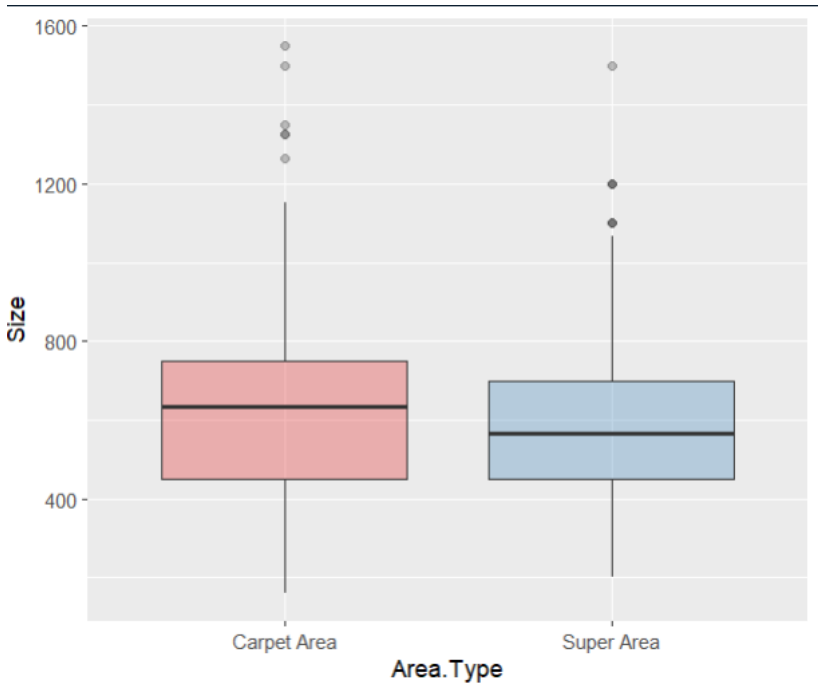


Figure 3.2.36: Plot for Analysis 2-17

For Mumbai, the median property size measured by carpet area is around 610 square feet and around 590 for the ones measured by super area.

Findings for Analysis 2-12 to 2-17

After analysing the overall size of rental units in each of the cities in India, we can make a few observations / conclusions:

1. The properties in Hyderabad are the biggest among all cities.
2. The properties in Delhi are the smallest among all cities.
3. The properties measured by carpet area is generally bigger than the ones measured by super area in all cities except Kolkata. This is abnormal. There might be some errors in the data acquisition steps.

Analysis 2-18: Find out how the number of bedroom of properties in India look like.

Number of bedrooms is one of the most common characteristics of a property. This is because it can give its owner quite a big impact in a way that it influences the number of people that can live in that particular property. Hence, this analysis is about the number of bedrooms of properties in India as a whole.

```
library(fmsb)

data_temp <-data.frame(
  length(which(cleaned_dataset$Bedroom == 1)),
  length(which(cleaned_dataset$Bedroom == 2)),
  length(which(cleaned_dataset$Bedroom == 3)),
  length(which(cleaned_dataset$Bedroom == 4)),
  length(which(cleaned_dataset$Bedroom == 5)),
  length(which(cleaned_dataset$Bedroom == 6))
)
colnames(data_temp) <- c("1", "2", "3", "4", "5", "6")
max <- max(data_temp[1,])
min <- min(data_temp[1,])

data_temp <- rbind(rep(max,6) , rep(min,6) , data_temp)

radarchart(data_temp, title = "Number of bedroom")
```

Figure 3.2.37: Visualizing the condition of bedroom number of properties in India

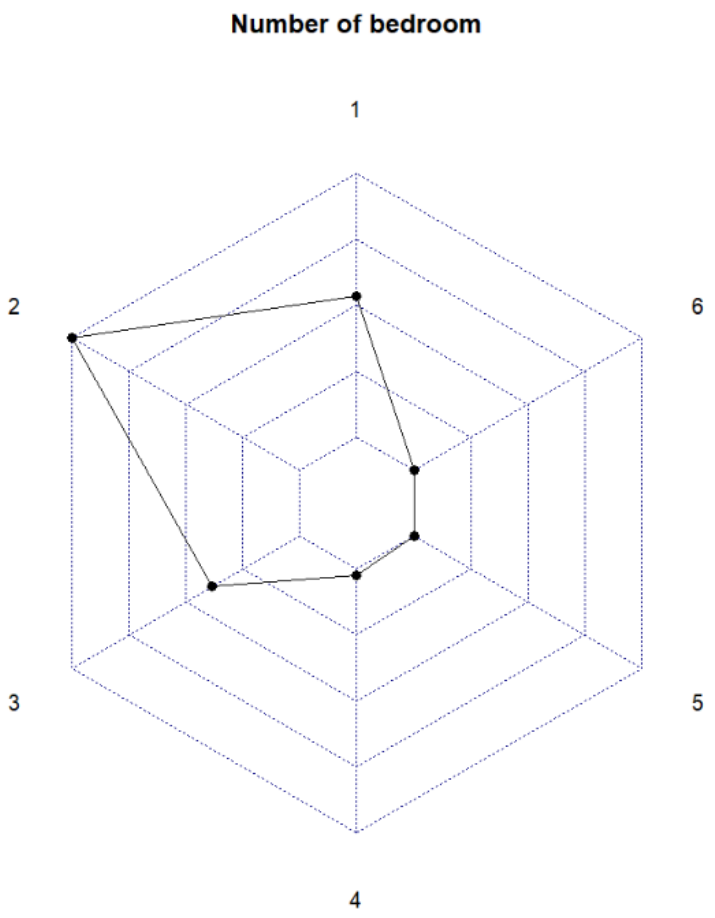


Figure 3.2.38: Plot for Analysis 2-18

The radar chart above shows that most of the properties in India have 2 bedrooms. It is immediately followed by 1-bedroom and 3-bedroom properties.

Conclusion: The properties in India are mostly for small family which has fewer members.

Analysis 2-19: Find out how the number of bedrooms in every city in India look like.

This analysis is an extension of the previous analysis in a way that this analysis group the dataset by cities. It is important to know about the bedroom condition in every city in India as well.

```
max(cleaned_dataset$Bedroom)
min(cleaned_dataset$Bedroom)
```

Figure 3.2.39: Find the maximum and minimum number of bedrooms

The code needed to generate this heatmap chart is a little complex, hence we will explain the process entirely. The first step is to find the maximum and minimum number of bedrooms the properties in India have. These 2 pieces of data will be needed later (Figure above).

```
data_temp <- data.frame(
  one = c(
    length(which(cleaned_dataset$Bedroom == 1 & cleaned_dataset$City == "Bangalore")),
    length(which(cleaned_dataset$Bedroom == 1 & cleaned_dataset$City == "Chennai")),
    length(which(cleaned_dataset$Bedroom == 1 & cleaned_dataset$City == "Delhi")),
    length(which(cleaned_dataset$Bedroom == 1 & cleaned_dataset$City == "Hyderabad")),
    length(which(cleaned_dataset$Bedroom == 1 & cleaned_dataset$City == "Kolkata")),
    length(which(cleaned_dataset$Bedroom == 1 & cleaned_dataset$City == "Mumbai"))
  ),
  two = c(
    length(which(cleaned_dataset$Bedroom == 2 & cleaned_dataset$City == "Bangalore")),
    length(which(cleaned_dataset$Bedroom == 2 & cleaned_dataset$City == "Chennai")),
    length(which(cleaned_dataset$Bedroom == 2 & cleaned_dataset$City == "Delhi")),
    length(which(cleaned_dataset$Bedroom == 2 & cleaned_dataset$City == "Hyderabad")),
    length(which(cleaned_dataset$Bedroom == 2 & cleaned_dataset$City == "Kolkata")),
    length(which(cleaned_dataset$Bedroom == 2 & cleaned_dataset$City == "Mumbai"))
  ),
  three = c(
    length(which(cleaned_dataset$Bedroom == 3 & cleaned_dataset$City == "Bangalore")),
    length(which(cleaned_dataset$Bedroom == 3 & cleaned_dataset$City == "Chennai")),
    length(which(cleaned_dataset$Bedroom == 3 & cleaned_dataset$City == "Delhi")),
    length(which(cleaned_dataset$Bedroom == 3 & cleaned_dataset$City == "Hyderabad")),
    length(which(cleaned_dataset$Bedroom == 3 & cleaned_dataset$City == "Kolkata")),
    length(which(cleaned_dataset$Bedroom == 3 & cleaned_dataset$City == "Mumbai"))
  ),
  four = c(
    length(which(cleaned_dataset$Bedroom == 4 & cleaned_dataset$City == "Bangalore")),
    length(which(cleaned_dataset$Bedroom == 4 & cleaned_dataset$City == "Chennai")),
    length(which(cleaned_dataset$Bedroom == 4 & cleaned_dataset$City == "Delhi")),
    length(which(cleaned_dataset$Bedroom == 4 & cleaned_dataset$City == "Hyderabad")),
    length(which(cleaned_dataset$Bedroom == 4 & cleaned_dataset$City == "Kolkata")),
    length(which(cleaned_dataset$Bedroom == 4 & cleaned_dataset$City == "Mumbai"))
  ),
  five = c(
    length(which(cleaned_dataset$Bedroom == 5 & cleaned_dataset$City == "Bangalore")),
    length(which(cleaned_dataset$Bedroom == 5 & cleaned_dataset$City == "Chennai")),
    length(which(cleaned_dataset$Bedroom == 5 & cleaned_dataset$City == "Delhi")),
    length(which(cleaned_dataset$Bedroom == 5 & cleaned_dataset$City == "Hyderabad")),
    length(which(cleaned_dataset$Bedroom == 5 & cleaned_dataset$City == "Kolkata")),
    length(which(cleaned_dataset$Bedroom == 5 & cleaned_dataset$City == "Mumbai"))
  ),
  six = c(
    length(which(cleaned_dataset$Bedroom == 6 & cleaned_dataset$City == "Bangalore")),
    length(which(cleaned_dataset$Bedroom == 6 & cleaned_dataset$City == "Chennai")),
    length(which(cleaned_dataset$Bedroom == 6 & cleaned_dataset$City == "Delhi")),
    length(which(cleaned_dataset$Bedroom == 6 & cleaned_dataset$City == "Hyderabad")),
    length(which(cleaned_dataset$Bedroom == 6 & cleaned_dataset$City == "Kolkata")),
    length(which(cleaned_dataset$Bedroom == 6 & cleaned_dataset$City == "Mumbai"))
  )
)
```

Figure 3.2.40: Data transformation for chart plotting's use

Next, we will create a new data frame called *data_temp* in order to be used later for heatmap chart plotting (Figure above).

```
row.names(data_temp) <- c(sort(unique(cleaned_dataset$City)))
heatmap(as.matrix(data_temp), cexRow = 1, cexCol = 1, xlab = "Number of bedroom")
```

Figure 3.2.41: Continue with data transformation procedure and plot the heatmap

Then, we will assign city names to the rows. And plot the heatmap chart.

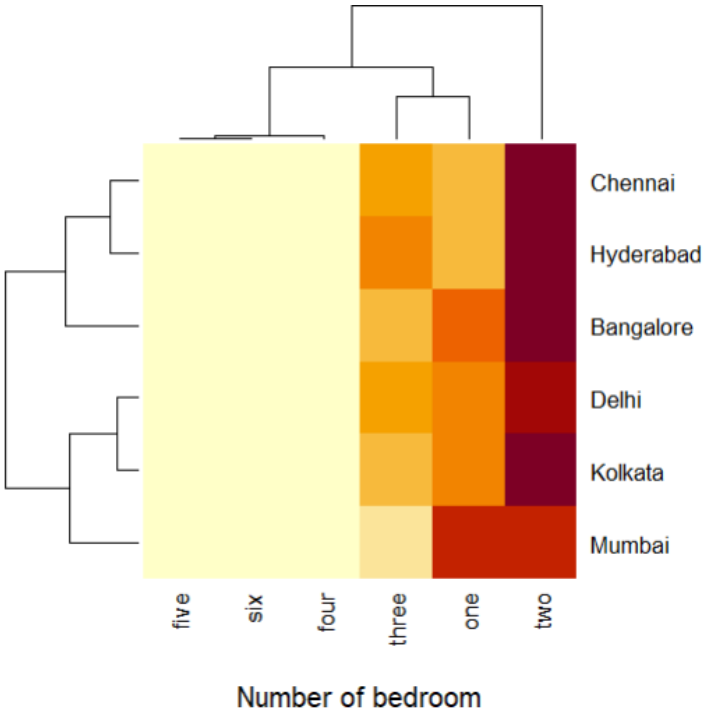


Figure 3.2.42: Plot for Analysis 2-19

From the heatmap chart above we can make a few interesting findings:

1. 2-bedroom properties are the most common ones among all.
2. In Mumbai, 1-bedroom properties are as common as 2-bedroom properties.
3. There are very few properties have more than 3 bedrooms in all the cities.
4. In Chennai and Hyderabad, there are more 3-bedroom properties than 1-bedroom properties.

Conclusion: Big families which have more than 10 family members in India, regardless of city, would be difficult to fit in one property given the lack of bedroom.

Analysis 2-20: Find out how the number of washrooms of properties in India look like.

Besides bedrooms, washrooms are also one of the key components of a property. This analysis gives people an idea how many washrooms a property in India will have.

```
max(cleaned_dataset$Bathroom)
min(cleaned_dataset$Bathroom)

data_temp <- data.frame(
  category=c("1","2","3","4","5","6","7") ,
  num = c(
  length(which(cleaned_dataset$Bathroom == 1)),
  length(which(cleaned_dataset$Bathroom == 2)),
  length(which(cleaned_dataset$Bathroom == 3)),
  length(which(cleaned_dataset$Bathroom == 4)),
  length(which(cleaned_dataset$Bathroom == 5)),
  length(which(cleaned_dataset$Bathroom == 6)),
  length(which(cleaned_dataset$Bathroom == 7))
  )
)

ggplot(data_temp, aes(x=category, y=num)) +
  geom_bar(stat = "identity", color="black", fill="#023123") +
  xlab("Number of Bathroom") +
  ylab("Number of units")
```

Figure 3.2.43: Visualizing the number of bathrooms of properties in India

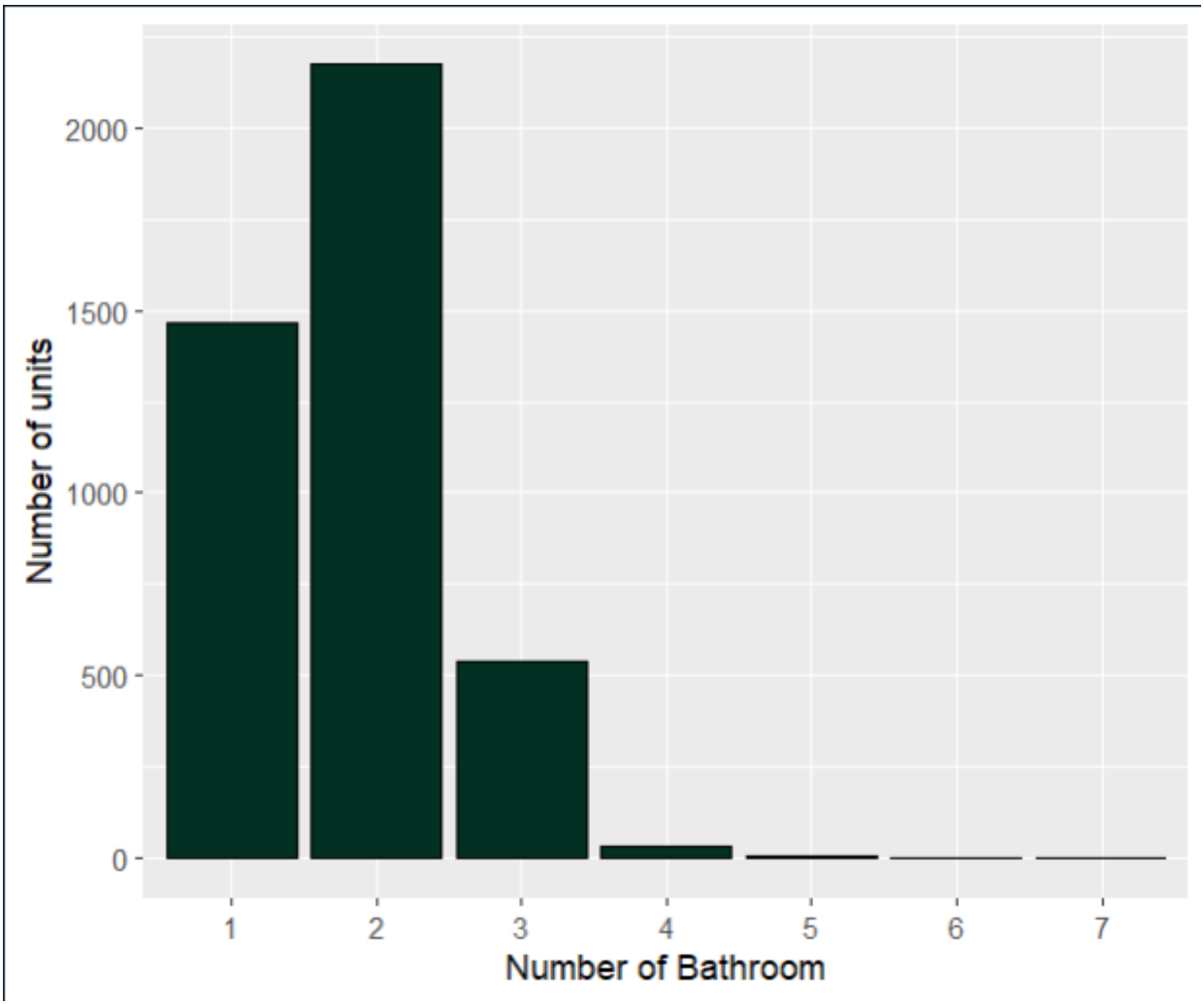


Figure 3.2.44: Plot for Analysis 2-20

Most of the properties in India have 2 bathrooms. This is followed by 1-bathroom and 3-bathroom properties.

Conclusion: The number of bathrooms in India's properties are more or less the same as the number of bedrooms. There might be a correlation between these 2 pieces of data. This will be addressed by the next analysis.

Analysis 2-21: Find the relationship between number of bedrooms and number of bathrooms.

This analysis is to figure out if there is any relationship between number of bedrooms and the number of bathrooms. The purpose of this analysis is to provide people, especially interior designers, and architects a reference on the appropriate combinations of bedroom and bathroom number.

```
data_temp <- cleaned_dataset %>% count(Bedroom, Bathroom)

ggplot(data_temp, aes(x=Bedroom, y=Bathroom, size = n)) +
  geom_point() +
  scale_size(range = c(1, 10), name="Count (units)")
```

Figure 3.2.45: Visualizing the correlation between number of bathrooms and number of bedrooms

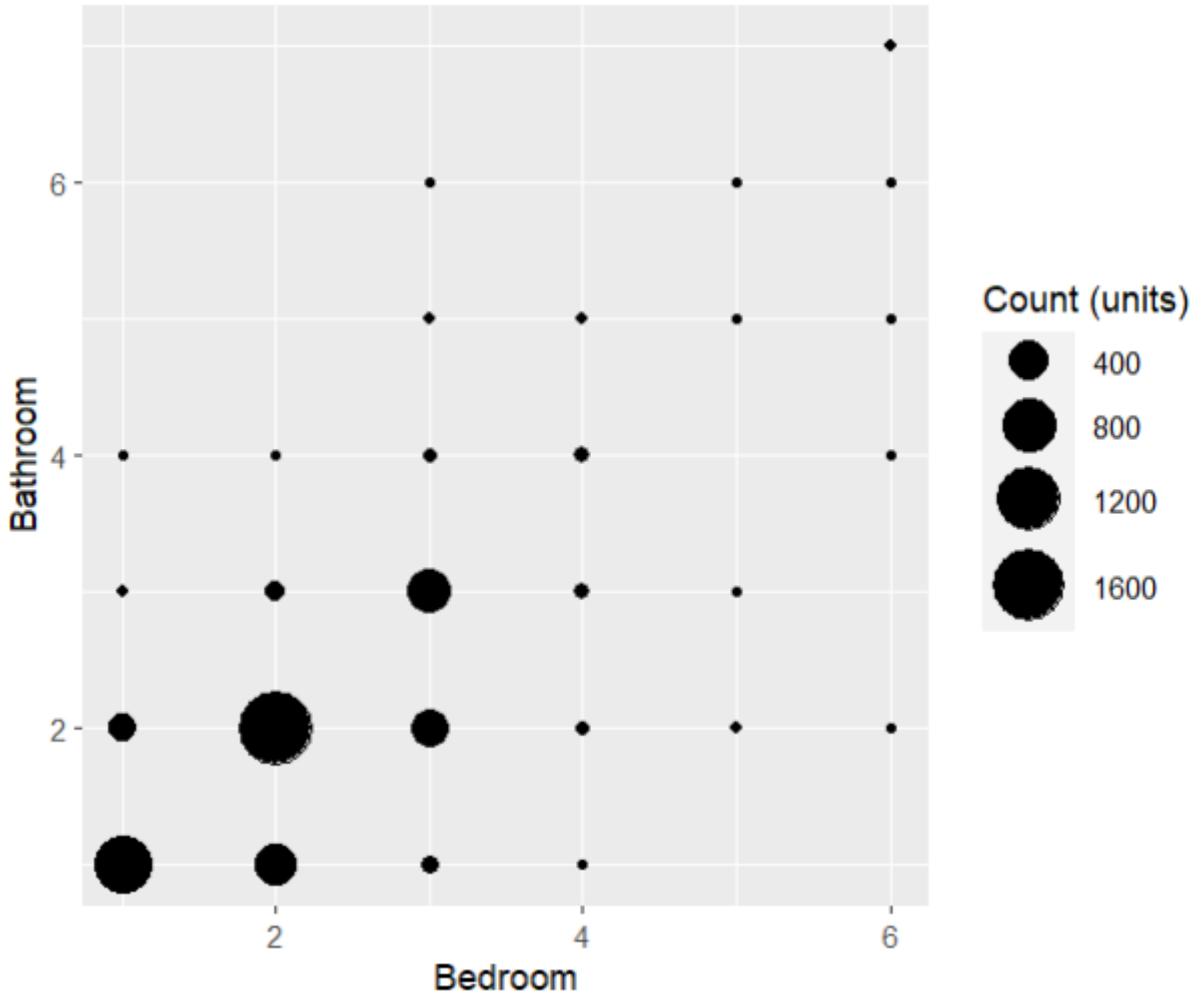


Figure 3.2.46: Plot for Analysis 2-21

From the bubble plot we can see that the 2-bedroom-2-bathroom combination forms the majority of the data. This is then followed by 1-bedroom-1-bathroom combination and 3-bedroom-3-bathroom combination. Interestingly, there are also quite a number of units which have 2 bedrooms and 1 bathroom, as well as units which have 3 bedrooms and 2 bathrooms.

Conclusion: The number of bathrooms and number of bedrooms do show a positive correlation. Generally speaking, the more the number of bedrooms, the more the number of bathrooms.

Conclusion for Question 2

1. More than 90% of properties in India have more than 1 floor.
2. Mumbai is the most modernized city – almost no single-floor property can be found in this city.
3. Kolkata and Chennai are the least modernized cities among all.
4. The most common properties in India are double-storey houses and low-rise apartments.
5. Bangalore's most common property is low-rise apartments that are within 5 floors.
6. Chennai's most common property is double-storey houses.
7. Delhi's most common property is 4-floor apartments.
8. Hyderabad's most common property is low-rise apartments that are within 6 floors.
9. Almost all properties in Kolkata are within 5-floor in height. Double-storey houses are the most common property in the city.
10. Mumbai has many high-rise apartments. Many apartments go up to 24 floors in height. The most common property is 6-floor apartments.
11. Properties in India are mostly 450 to 1000 square feet in carpet area, and 500 to 1200 square feet in super area.
12. The biggest property in India has 3500 square feet in carpet area, as well as another property with 4200 square feet being measured in super area.
13. Only 2 properties are being measured in built area.
14. Hyderabad has biggest average property size, whereas Delhi has the smallest.
15. Properties measured in carpet area is generally bigger than the ones measured in super area, which is an abnormal case.
16. Most properties in India have 2 bedrooms.
17. Mumbai has the least number of bedrooms, whereas Chennai and Hyderabad have the most.
18. Number of bathrooms in India is similar to number of bedrooms.
19. Properties in India show a trend: the more the number of bedrooms, the more the number of bathrooms.

3.3 Question 3 – Factors Influencing Rent

Question: What are the factors that will influence rent?

Rent, undoubtedly, is one of the most fundamental factors when tenants are to pick rental unit. Hence, this question is a view from any parties who are looking to find a rental unit or understand the factors that have an impact on the rent of a property. This may include tenants, property owners, real estate investment trusts (REITs), etc. Hence, this question can bring values to individuals as well as business organizations.

Analysis 3-1: Find out the condition of rent in India as a whole.

This analysis is to figure out the condition of rent in India as a whole. It is very important to understand the context before going deeper into any data analysis. Hence, this analysis serves as a context-understanding analysis, which will tell us about the current rent situation in the nation.

```
library(hrbrthemes)

ggplot(cleaned_dataset, aes(x=Rent)) +
geom_density(fill="#3cb3a2", color="#e9ecef") +
ggtitle("Rent in India") +
theme_ipsum()
```

Figure 3.3.1: Visualizing the rent in India as a whole

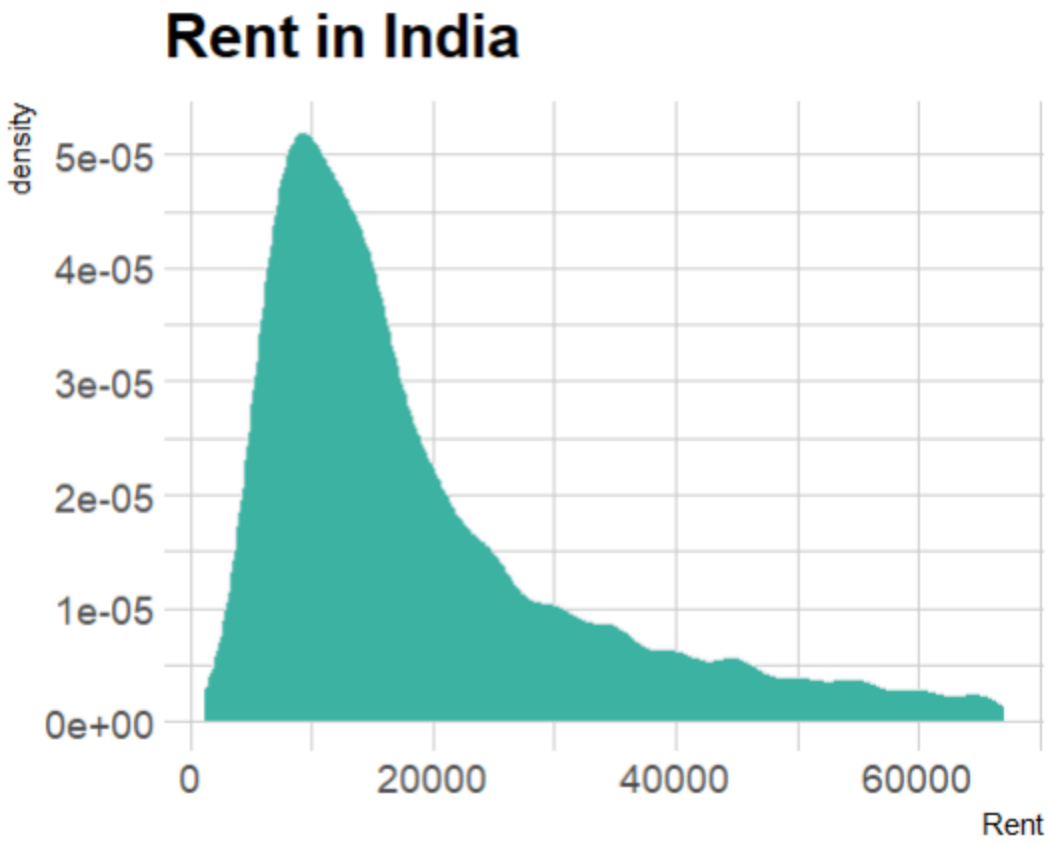


Figure 3.3.2: Plot for Analysis 3-1

From the density plot above we can observe that the highest density level is at the rent level of 9000 ₹ approximately. The density then declines steeply and form a **logarithmic curve**. The highest rent of this dataset is around 67000 ₹.

Conclusion: Most of the properties in India have a rent of 4000 ₹ to 28000 ₹.

Analysis 3-2: Find out the condition of rent in each of the part of India

This analysis is a context-understanding analysis as well. It is about finding out the rent condition in the different parts of India. This would give people an idea how the rent in every part of India works.

```
library(ggridges)

ggplot(cleaned_dataset, aes(x = Rent, y = Part.of.India, fill = Part.of.India)) +
  geom_density_ridges() +
  theme_ridges() +
  theme(legend.position = "none")
```

Figure 3.3.3: Visualizing rent in each of the part of India

Figure 3.3.4: Plot for Analysis 3-2

From the ridgeline plot above we can see that the rent is consistent at western India. At eastern India, the rent is generally lower and more densed at the level of 8000 ₹. Southern and northern India, on the other hand, are more or less the same, which is in the middle range.

Conclusion: Eastern India has the lowest rent across all the parts. Southern and northern India, on the other hand, are more to middle range on the rent scale. Western India has the highest average rent, which consists of a very consistent density of rent across the range between 10000 ₹ to 65000 ₹.

Analysis 3-3: Find out the condition of rent in each of the city of India.

This analysis is an extension of the previous analysis in a way that this analysis goes deeper into the rent condition in India. This analysis group every city in India and inspect their rent condition separately. This will give people an idea which city has higher and lower rent, which may help them decide which city suits them more.

```
ggplot(cleaned_dataset, aes(x = Rent, y = City, fill = City)) +
  geom_density_ridges() +
  theme_ridges() +
  theme(legend.position = "none") +
  ylab("City") +
  xlab("Rent") +
  ggtitle("Rent at Every City")
```

Figure 3.3.5: Visualizing rent in each of the city in India

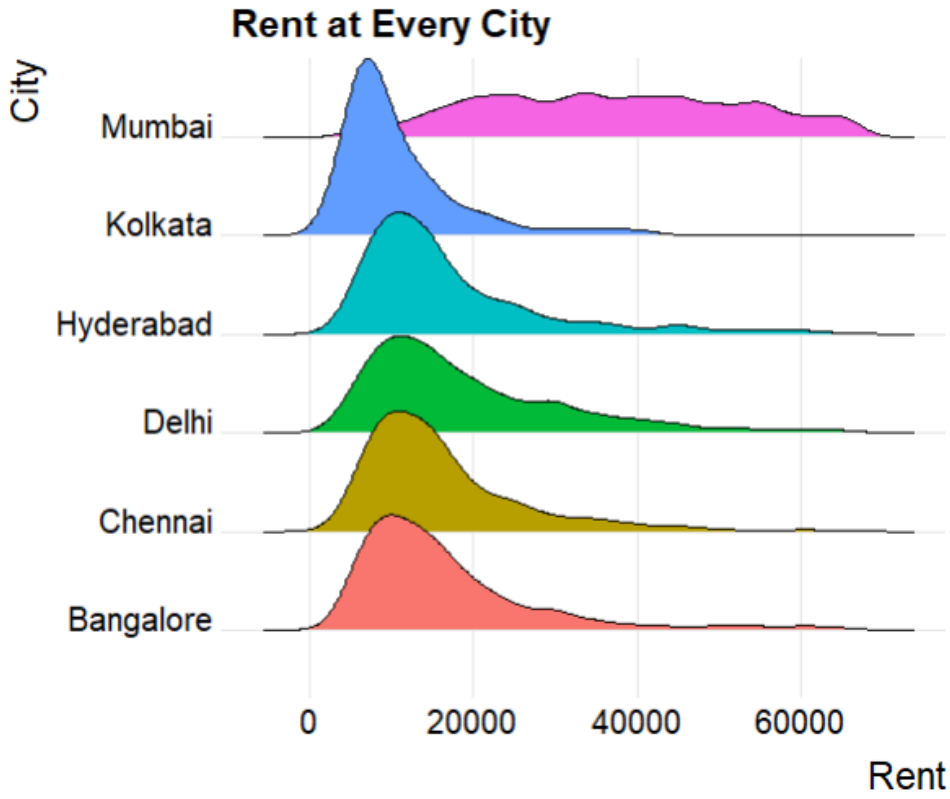


Figure 3.3.6: Plot for Analysis 3-3

Mumbai has a consistent density of rent between the range 20000 and 60000 ₹. Kolkata's rent is relatively more densed towards the lower range. Hyderabad, Delhi, Chennai, and Bangalore are very similar in terms of the density of the entire rent range.

Conclusion: Mumbai is the city with highest rent in average. It is not encouraged for tenants to live in the city, investors are suggested to take part in the city's property market. Kolkata's lowest rent across the city is favorable for tenants. Hyderabad, Delhi, Chennai, and Bangalore have middle range rent. Both tenants and investors may consider these cities.

Analysis 3-4: Find the relationship between rent and size of the units.

This analysis aims to figure out if there is any relationship between the size of a property and its rent. If yes, then we will find out more about that relationship.

```
ggplot(cleaned_dataset, aes(x=Size, y=Rent)) +
  geom_point(color="#69b3a2") +
  geom_smooth(method=lm , color="red", se=FALSE) +
  theme_ipsum()
```

Figure 3.3.7: Visualizing relationship between unit size and rent

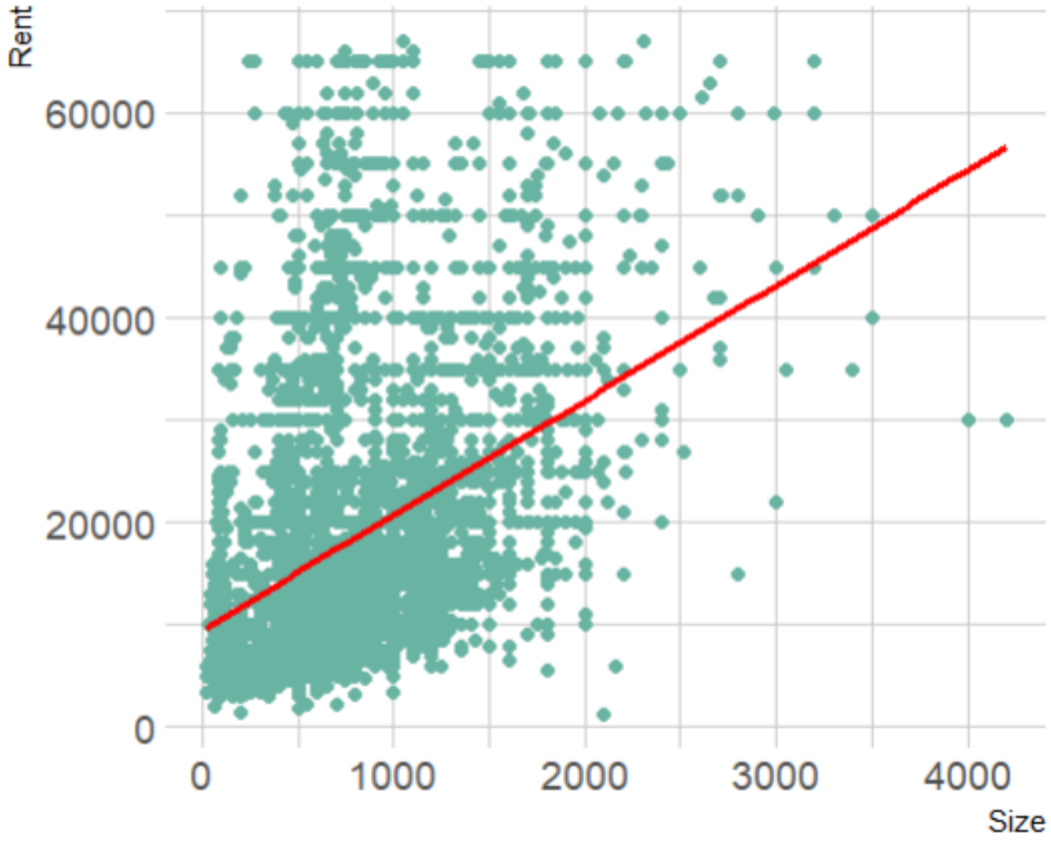


Figure 3.3.8: Plot for Analysis 3-4

The scatter plot of rent against size is shown above. From the plot we can see that there are some rental units with different sizes but demand the same amount of rent. The linear trend line shows a positive relationship between rent and size.

Conclusion: The trend line shows that the bigger the size of the rental units, the higher the rent will be.

Analysis 3-5: Find the relationship between furnishing status and rent of rental units.

This analysis is to find out if there is any relationship between furnishing status and rent of properties. Furniture is a cost for the property owner. Hence, it is expected that the higher the furnishing status, the higher the rent would be.

```
library(viridis)

ggplot(cleaned_dataset, aes(x=Furnishing.Status, y=Rent, fill=Furnishing.Status)) +
geom_violin() +
scale_fill_viridis(discrete = TRUE, alpha=0.6, option="A") +
theme_ipsum() +
theme(
  legend.position="none",
  plot.title = element_text(size=11)
) +
ggtitle("Rent against Furnishing Status") +
xlab("Furnishing Status")
```

Figure 3.3.9: Visualizing relationship between furnishing status and rent

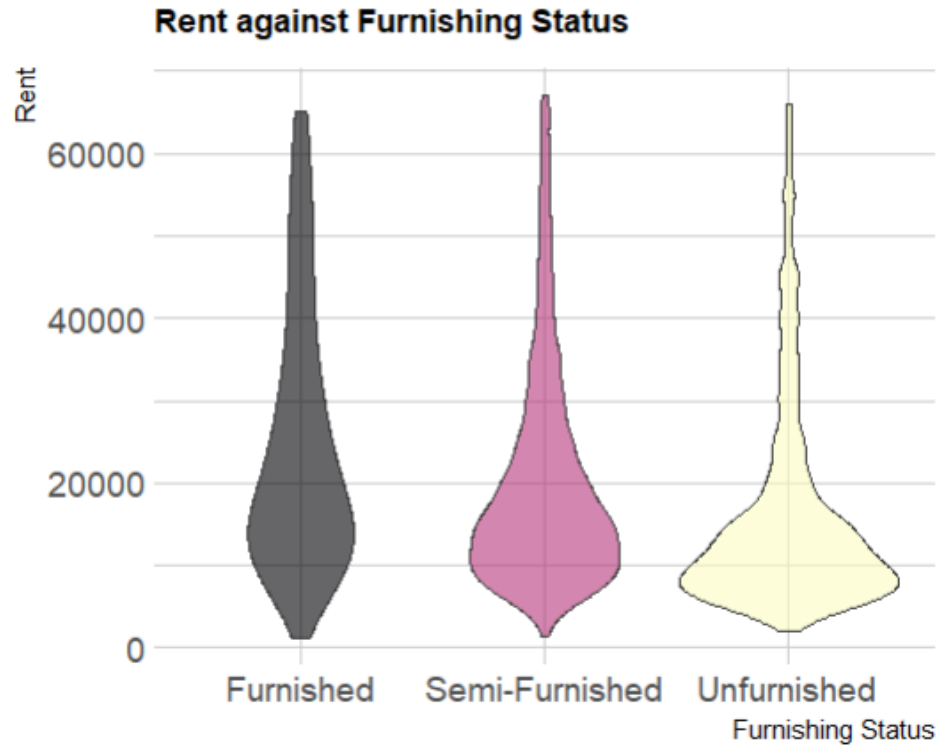


Figure 3.3.10: Plot for Analysis 3-5

Unfurnished rental units are generally lower in rent. Semi-furnished rental units are generally in mid-range of rent but have the highest rent among all. Lastly, fully furnished rental units have the highest average rent.

Conclusion: Furnishing status has some impacts on rent, but it is not the deterministic factor.

Analysis 3-6: Find the relationship between properties' level (floor) and rent.

This analysis is to find out if there is any relationship between the property's level and its rent. The unique selling proposition (USP) of some high-rise properties is the view from the high-level units. Hence, it is expected that these properties will have higher rent for the units with higher level.

```
data_temp <- data.frame(floor = cleaned_dataset$Floor, rent = cleaned_dataset$Rent)

unique(data_temp$floor)
data_temp <- subset(data_temp, ! data_temp$floor == "Upper Basement")
data_temp <- subset(data_temp, ! data_temp$floor == "Lower Basement")
class(data_temp$floor) = "numeric"

ggplot(data_temp, aes(x=floor, y=rent)) +
  geom_point() +
  geom_smooth(method=lm , color="red", se=FALSE) +
  theme_ipsum()
```

Figure 3.3.11: Visualizing the relationship between property's level and rent

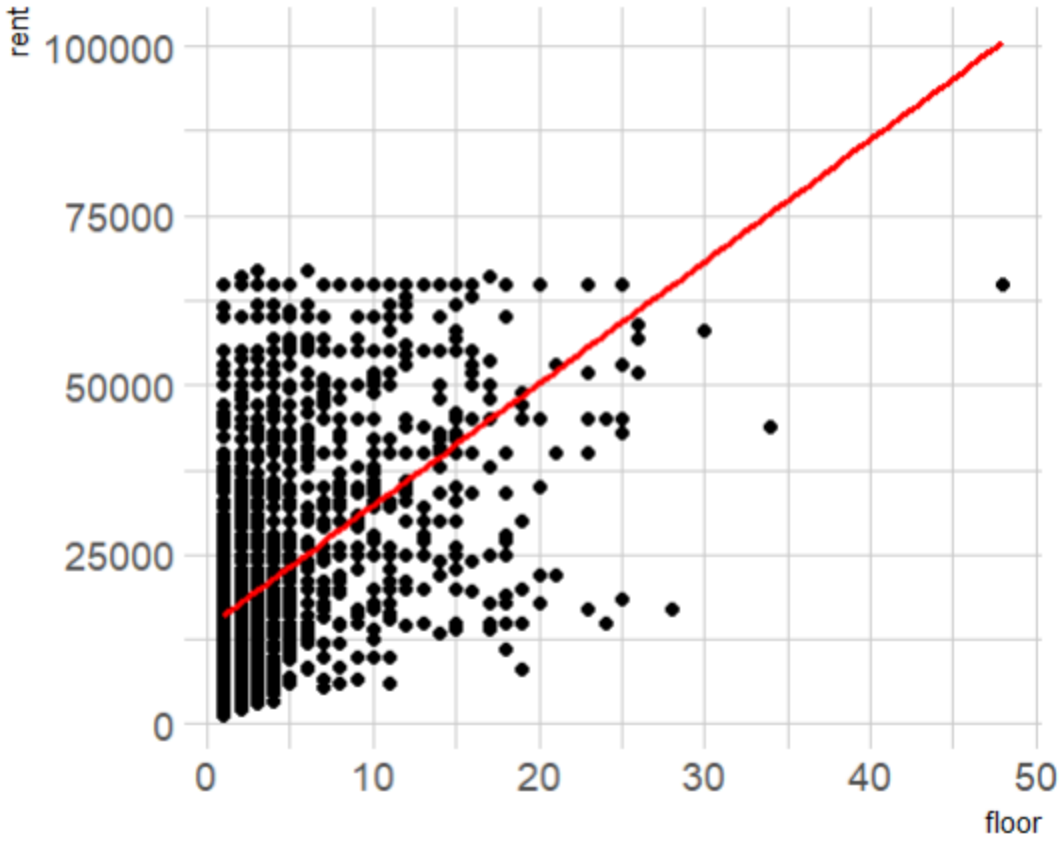


Figure 3.3.12: Plot for Analysis 3-6

The scatter plot of rent against floor shows that the properties located at lower levels (from range level 0 to 6) are normally have lower rent. The properties that are at higher levels scatter around on the rent scale.

Conclusion: The linear trend line shows that the floor is actually having a positively linear impact to rent. This means that the higher the property is located, the rent is likely to be higher.

Analysis 3-7: Find the relationship between number of bedrooms and rent.

Bedrooms, the place where people can be the most relaxed at, are very important for a property. For this analysis, we will determine if there is any relationship between the number of bedrooms and rent of a rental unit.

```
data_temp <- data.frame(bedroom = cleaned_dataset$Bedroom, rent = cleaned_dataset$Rent)
data_temp$bedroom <- factor(data_temp$bedroom)

ggplot(data_temp, aes(x=bedroom, y=rent, fill=bedroom)) +
  geom_boxplot() +
  scale_fill_viridis(discrete = TRUE, alpha=0.6, option="A") +
  theme_ipsum() +
  theme(
    legend.position="none",
    plot.title = element_text(size=11)
  ) +
  ggtitle("Rent against Number of Bedrooms") +
  xlab("Number of Bedroom") +
  ylab("Rent")
```

Figure 3.3.13: Visualizing number of bedrooms against rent

Figure 3.3.14: Plot for Analysis 3-7

From the box plot we can see that the rent is generally higher when the number of bedrooms is higher. The median of the sixth box is lower than fifth and fourth box's median. This might be due to the inaccuracy caused by small dataset of 6-bedroom units. We can also see that first and second box have many individual points at the top.

Conclusion: The number of bedrooms has a slight / non-deterministic impact on rent. The individual points on top of the first and second box indicates that properties with small number of bedrooms might also demand a high rent.

Analysis 3-8: Find the relationship between number of bathrooms and rent.

This analysis is about figuring out if there is any relationship between the number of bathrooms and rent of a rental unit. Bathrooms are one of the not-so-important characteristics of a property because the rate of utilization is very low. However, we would like to understand the relationship as well.

```
data_temp <- data.frame(bathroom = cleaned_dataset$Bathroom, rent = cleaned_dataset$Rent)
data_temp$bathroom <- factor(data_temp$bathroom)

ggplot(data_temp, aes(y=bathroom, x=rent, fill=bathroom)) +
  geom_density_ridges(alpha=0.6, stat="binline", bins=20) +
  theme_ridges() +
  theme(
    legend.position="none",
    panel.spacing = unit(0.1, "lines"),
    strip.text.x = element_text(size = 8)
  ) +
  xlab("Rent") +
  ylab("Number of Bathroom")
```

Figure 3.3.15: Visualizing the relationship between number of bathrooms and rent

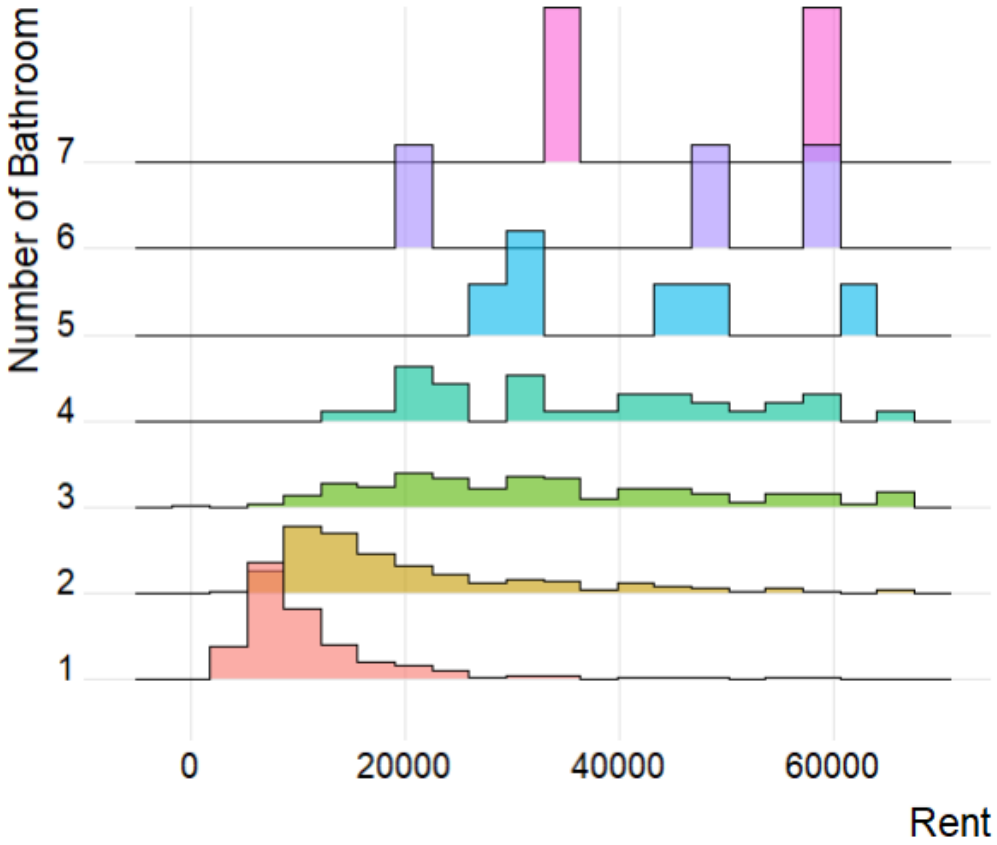


Figure 3.3.16: Plot for Analysis 3-8

From the ridgeline plot above we can observe that the 1-bathroom and 2-bathroom properties are clearly more densed at the range of lower rent. For 3-bathroom and 4-bathroom properties, the rent is relatively more consistent throughout the entire scale of rent. The 5-bathroom, 6-bathroom, and 7-bathroom properties, on the other hand, are intermittent on the range of rent that is higher than 20000 ₹.

Conclusion: For the properties that have less bathrooms, the rent is generally lower. For properties that have more than 2 bathrooms, the rent is not much related to rent anymore.

Analysis 3-9: Find the relationship between point of contact and rent.

This analysis will find out if there is any relationship between the way tenants approach rental units and the rent. This will provide a guideline for tenants when they are to find properties.

```
ggplot(cleaned_dataset, aes(x=Point.of.Contact, y=Rent, fill=Point.of.Contact)) +
  geom_violin() +
  theme(
    legend.position="none",
  ) +
  xlab("Point of Contact") +
  ggtitle("Rent against Point of Contact")
```

Figure 3.3.17: Visualizing the relationship between point of contact and rent

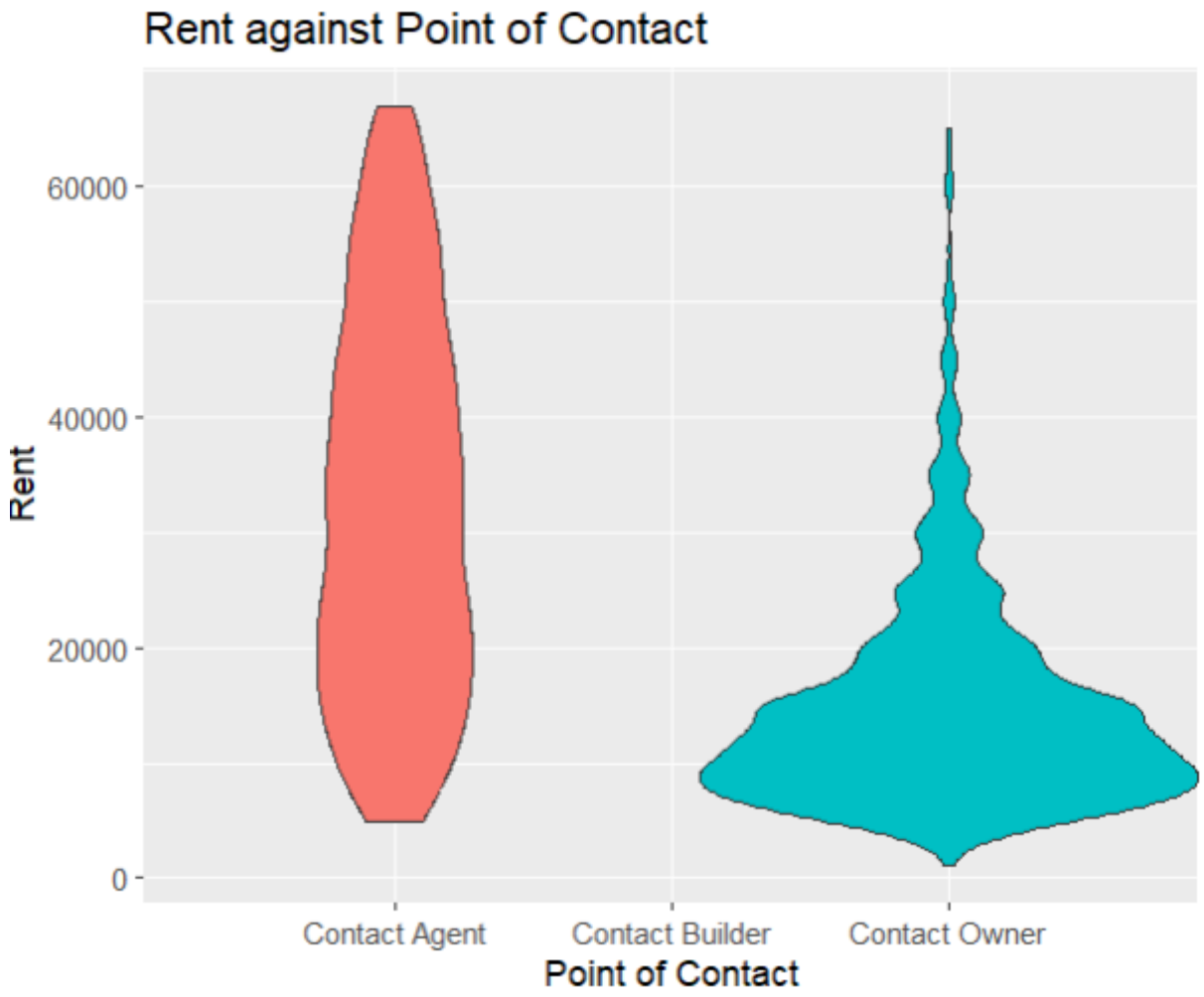


Figure 3.3.18: Plot for Analysis 3-9

From the violin plot above we can see that the "*Contact Owner*" data is more densed towards the lower range of rent. The rent of properties that require tenants to contact agent, on the other hand, shows a relatively consistent trend throughout the entire rent range. Lastly, the "*Contact Builder*" data is dropped by the *ggplot2* library due to its neglectable size of data.

Conclusion: The properties that go through a middleman – the agents – demand **higher** rent compared to the ones that do not. Hence, we can safely say that the point of contact does influence the amount of rent.

Analysis 3-10: Find the relationship between average housing price and rent.

This analysis aims to figure out if there is any relationship between the average housing price and rent. Housing price indicates the cost of owning a property, while rent indicates the income of that particular property. They are opposing one another. Hence, by analyzing them we can know if the valuation of the property in India is within a rational range.

```
data_temp <- data.frame(rent = cleaned_dataset$Rent, price = cleaned_dataset$Average.Housing.Price)
data_temp$price <- factor(data_temp$price)

ggplot(data_temp, aes(x=price, y=rent, fill=price)) +
  geom_boxplot() +
  theme_ipsum() +
  theme(
    legend.position="none",
    plot.title = element_text(size=11)
  ) +
  ggtitle("Rent against Average Housing Price") +
  xlab("Average Housing Price") +
  ylab("Rent")
```

Figure 3.3.19: Visualizing the relationship between average housing price and rent

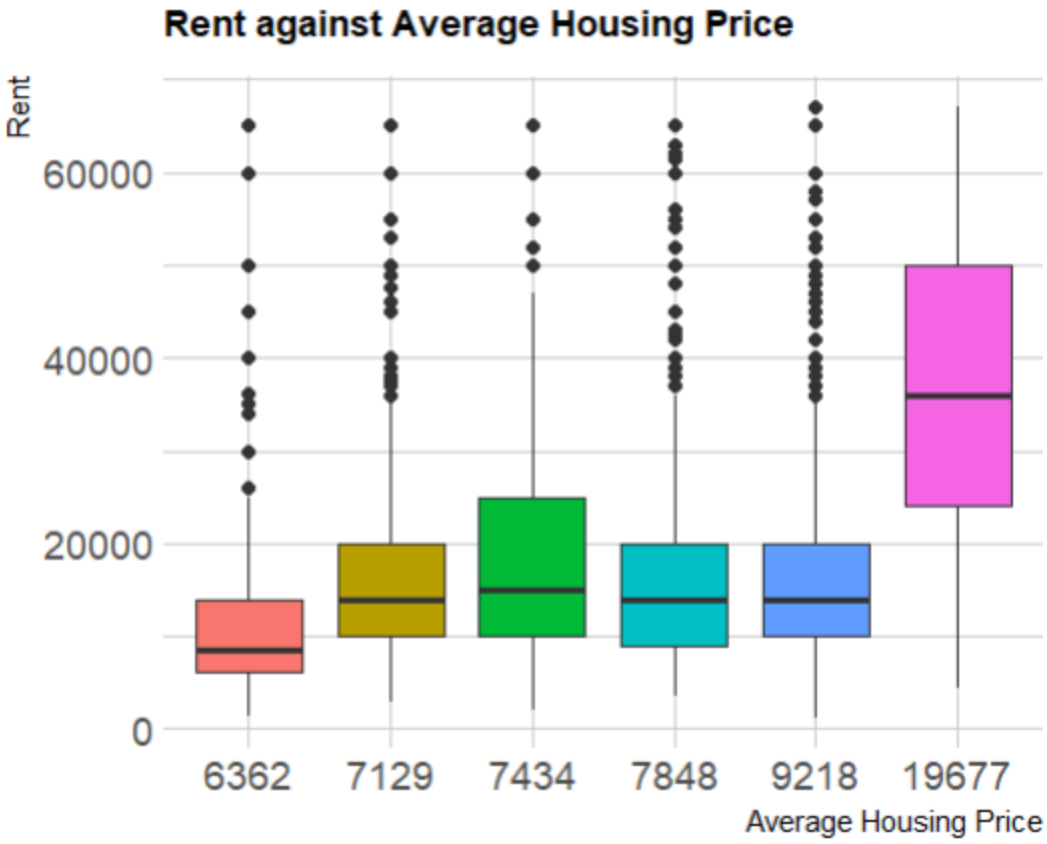


Figure 3.3.20: Plot for Analysis 3-10

Generally speaking, the rent is higher when the average housing price is higher. However, there are exceptions.

Conclusion: The average housing price is an influencing factor of rent. However, due to the different market conditions such as lower demand on the rental market in some cities, the rent of properties might be lower than it should be. This is known as undervalued. When the rental market goes back to normal, the rent will increase accordingly.

Analysis 3-11: Find the relationship between population and rent.

This analysis is to figure out if there is any relationship between population and rent in each of the city in India. Population means people, and people need shelter. Hence, we expect there is some kind of relationship between these 2 variables, even if it is a weak one.

```
sort(unique(cleaned_dataset$Population.2021))
data_temp <- list(
  "10268653" = cleaned_dataset[cleaned_dataset$Population.2021 == 10268653, ]$Rent,
  "11235018" = cleaned_dataset[cleaned_dataset$Population.2021 == 11235018, ]$Rent,
  "12764835" = cleaned_dataset[cleaned_dataset$Population.2021 == 12764835, ]$Rent,
  "14974073" = cleaned_dataset[cleaned_dataset$Population.2021 == 14974073, ]$Rent,
  "20667655" = cleaned_dataset[cleaned_dataset$Population.2021 == 20667655, ]$Rent,
  "31181377" = cleaned_dataset[cleaned_dataset$Population.2021 == 31181377, ]$Rent
)

stripchart(data_temp,
           main="Multiple stripchart for comparision",
           xlab="Rent",
           ylab="Population",
           method="jitter",
           col=c("orange","red"),
           pch=16
)
```

Figure 3.3.21: Visualizing the relationship between population and rent

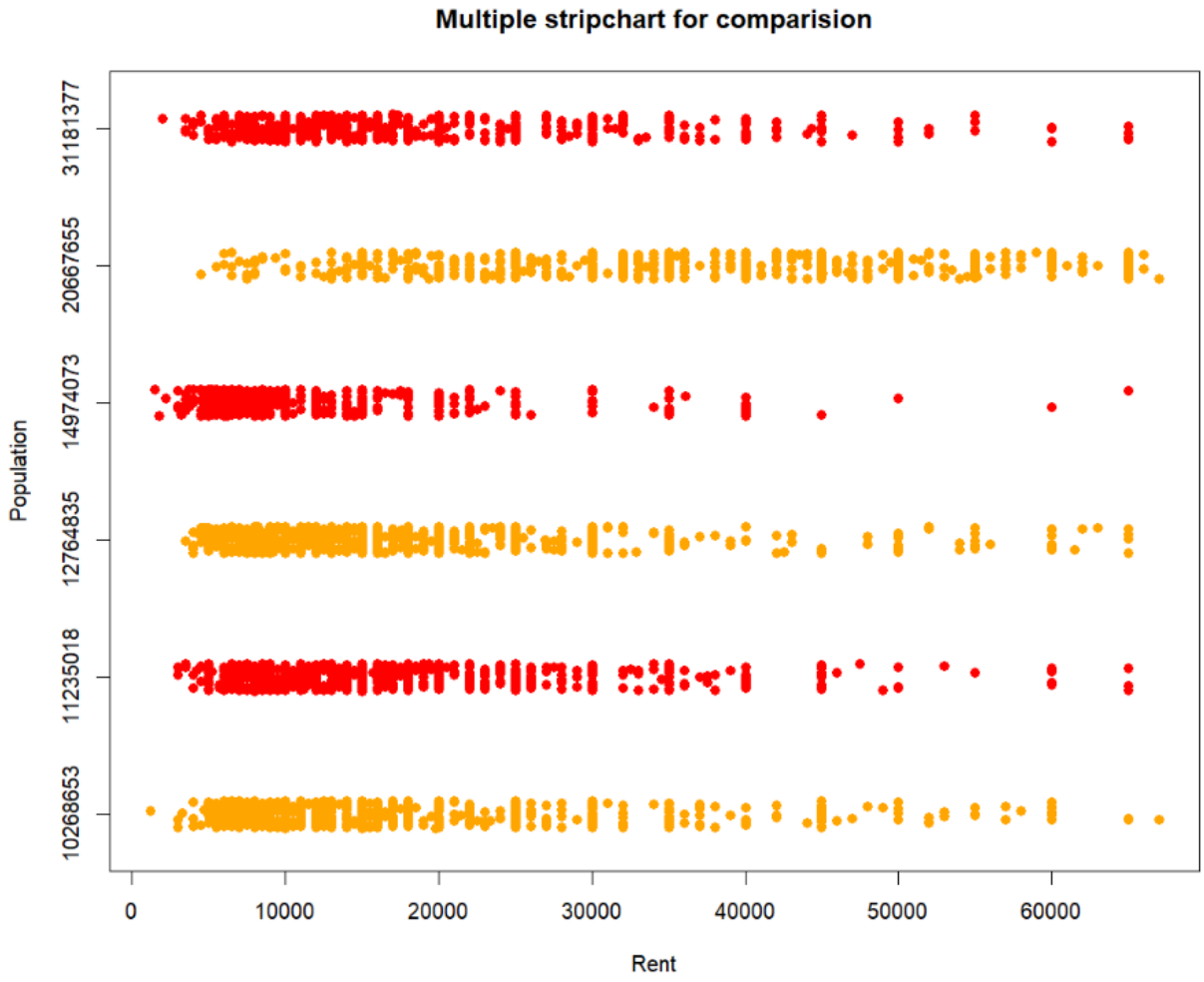


Figure 3.3.22: Plot for Analysis 3-11

From the multiple strip chart above we can see that the rent all densed towards the lower end of the rent scale regardless of the condition of population.

Conclusion: The population data is **not** related to rent.

Conclusion for Question 3

1. Most of the properties in India have a rent of 4000 ₹ to 28000 ₹.
2. Eastern India has the lowest rent across all the parts. Southern and northern India, on the other hand, are more to middle range on the rent scale. Western India has the highest average rent, which consists of a very consistent density of rent across the range between 10000 ₹ to 65000 ₹.
3. The bigger the size of the rental units, the higher the rent will be.
4. Furnishing status has some impacts on rent, but it is not the deterministic factor.
5. The higher the property is located, the rent is likely to be higher.
6. The number of bedrooms has a slight / non-deterministic impact on rent.
7. For the properties that have less bathrooms, the rent is generally lower. For properties that have more than 2 bathrooms, the rent is not much related to rent anymore.
8. The properties that go through a middleman – the agents – demand higher rent compared to the ones that do not.
9. The average housing price is an influencing factor of rent.
10. The population data is **not** related to rent.

3.4 Question 4 – Rental Unit Preferences

Question: What kind of properties does bachelor and family like?

All the previous topics are more towards the economic side of analysis. This topic we are more concerned with the social demographic conditions in India. We would like to know about the lifestyle of the different groups of people in India. Hence, this question is from the view of any parties that would like to know the house preferences of 2 groups of people – bachelors and families – in India. This may include government authorities, academic researchers, property investors, etc. This question will bring values to the industry as well as the academia.

Analysis 4-1: Find the preference of bachelor and family on number of bedrooms.

Given the difference in the number of individuals involved, bachelors and families are expected to have different preference on the number of bedrooms in their properties. Hence, this analysis aims to find out if this hypothesis is true.

```
data_Q4 <- subset(cleaned_dataset, ! cleaned_dataset$Tenant.Preferred == "Bachelors/Family")

ggplot(data_Q4, aes(x=Tenant.Preferred, y=Bedroom, fill=Tenant.Preferred)) +
  geom_violin() +
  theme(
    legend.position="none",
  ) +
  xlab("Tenant Type") +
  ylab("Number of Bedroom") +
  ggtitle("Number of Bedroom against Tenant Type")
```

Figure 3.4.1: Visualizing the preference of bachelor and family on number of bedrooms

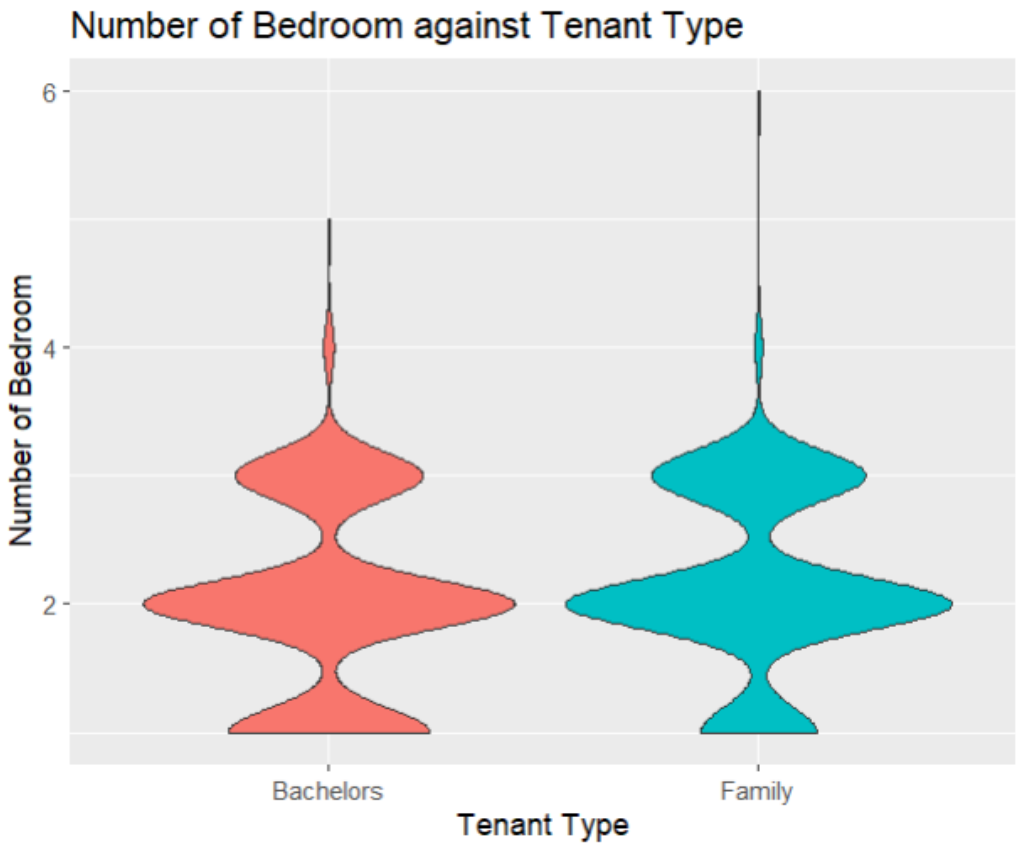


Figure 3.4.2: Plot for Analysis 4-1

From the violin plot above we can see that the bachelors and family actually have very similar preference on the number of bedrooms. However, we can see that more bachelors prefer 1-bedroom properties than families. Both bachelors and families prefer 2-bedroom properties the most.

Conclusion: Number of bedrooms are not the deterministic factor for bachelors and families when they choose rental units. However, bachelors are a little more inclined towards properties with less number of bedrooms because they do not have the need.

Analysis 4-2: Find the preference of bachelor and family on number of bathrooms.

Bathroom is very similar to bedroom in terms of their nature when tenants are to pick rental unit. Hence, just like bathrooms, we expect bachelors and families have different preference on number of bathrooms.

```
ggplot(data_Q4, aes(x=Bathroom, group=Tenant.Preferred, fill=Tenant.Preferred)) +
  geom_density(adjust=1.5) +
  theme_ipsum() +
  facet_wrap(~Tenant.Preferred) +
  theme(
    legend.position="none",
    panel.spacing = unit(0.1, "lines"),
    axis.ticks.x=element_blank()
  ) +
  xlab("Number of Bathrooms") +
  ylab("Density")
```

Figure 3.4.3: Visualizing the preference of bachelor and family on number of bathrooms

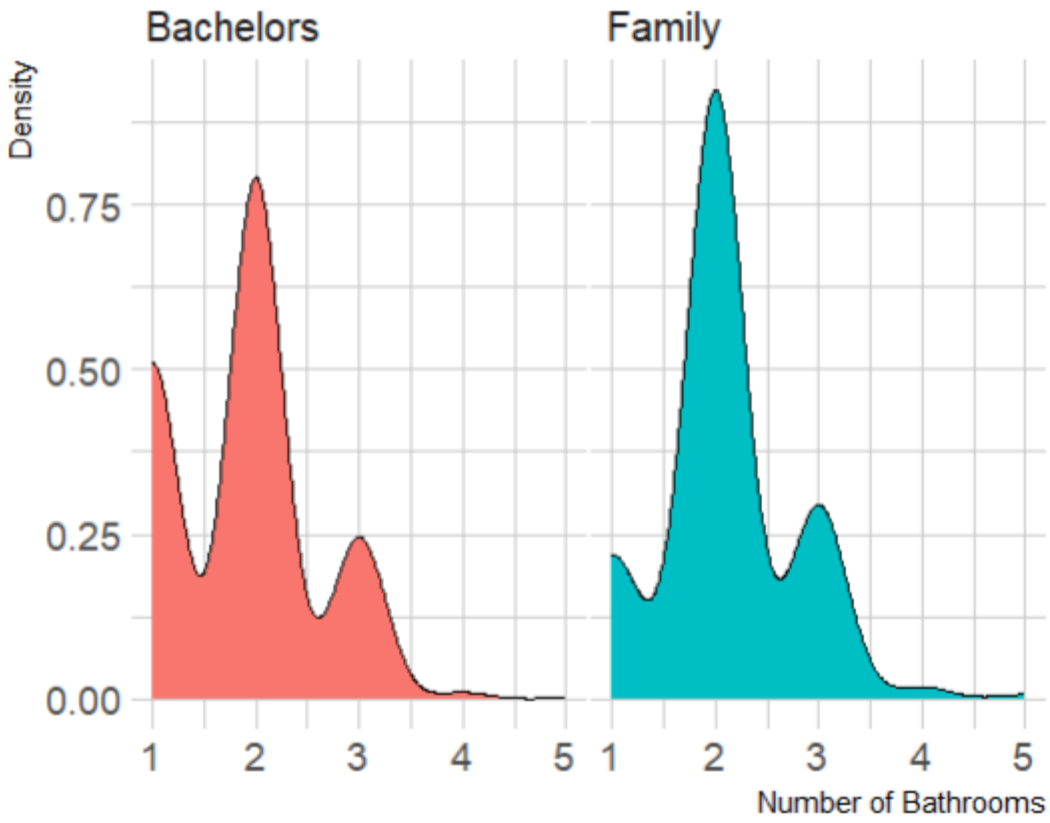


Figure 3.4.4: Plot for Analysis 4-2

From the density plot above we can observe that families generally prefer properties with more bathrooms. Meanwhile, bachelors mostly prefer properties with 1 or 2 bathrooms.

Conclusion: Bachelors do not need a lot of bathrooms. Not like empty bedrooms which can be used for purposes other than sleeping, there are no point having extra bathrooms. Families, on the other hand, need more bathrooms due to the bigger number of people.

Analysis 4-3: Find the property rent that bachelors and families prefer.

Rent is one of the main factors when tenants are to pick rental unit, especially the youths. Hence, this analysis aims to understand and analyze the rent that is preferred by bachelors and families.

```
ggplot(data_Q4, aes(x=Rent, fill=Tenant.Preferred)) +
geom_histogram( color="#e9ecef", alpha=0.6, position = 'identity') +
scale_fill_manual(values=c("#69b3a2", "#404080")) +
theme_ipsum() +
labs(fill="", y="Count")
```

Figure 3.4.5: Visualizing the property rent that bachelors and families prefer

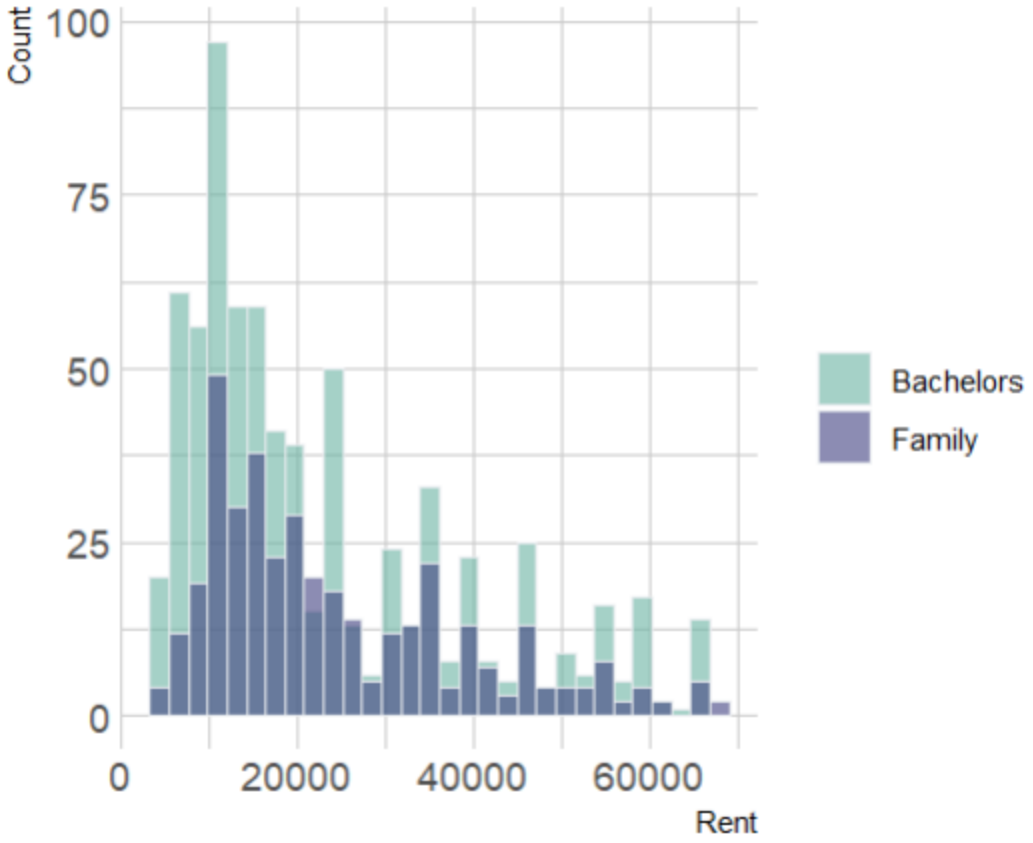


Figure 3.3.6: Plot for Analysis 4-3

The bachelors and families show a very similar trend on the preference of rent. Both the bachelors and families prefer properties that demand lower rent. The preference peaks at the range of 8000 ₹ to 10000 ₹.

Conclusion: Both bachelors and families prefer properties with lower rent.

Analysis 4-4: Find the bachelors' preference on size of rental units.

This analysis intends to figure out the bachelors' preferences on the size of rental unit. This would give property owners some idea on the size of property that they should invest in.

```
data_temp <- subset(data_Q4, ! data_Q4$Tenant.Preferred == "Family")

ggplot(data_temp, aes(x = Size, y = Area.Type, fill = Area.Type)) +
  geom_density_ridges() +
  theme_ridges() +
  theme(legend.position = "none") +
  ylab("Area Type") +
  ggtitle("Bachelor's Preference on Property Size")
```

Figure 3.3.7: Visualizing bachelors' preference on size of rental units.

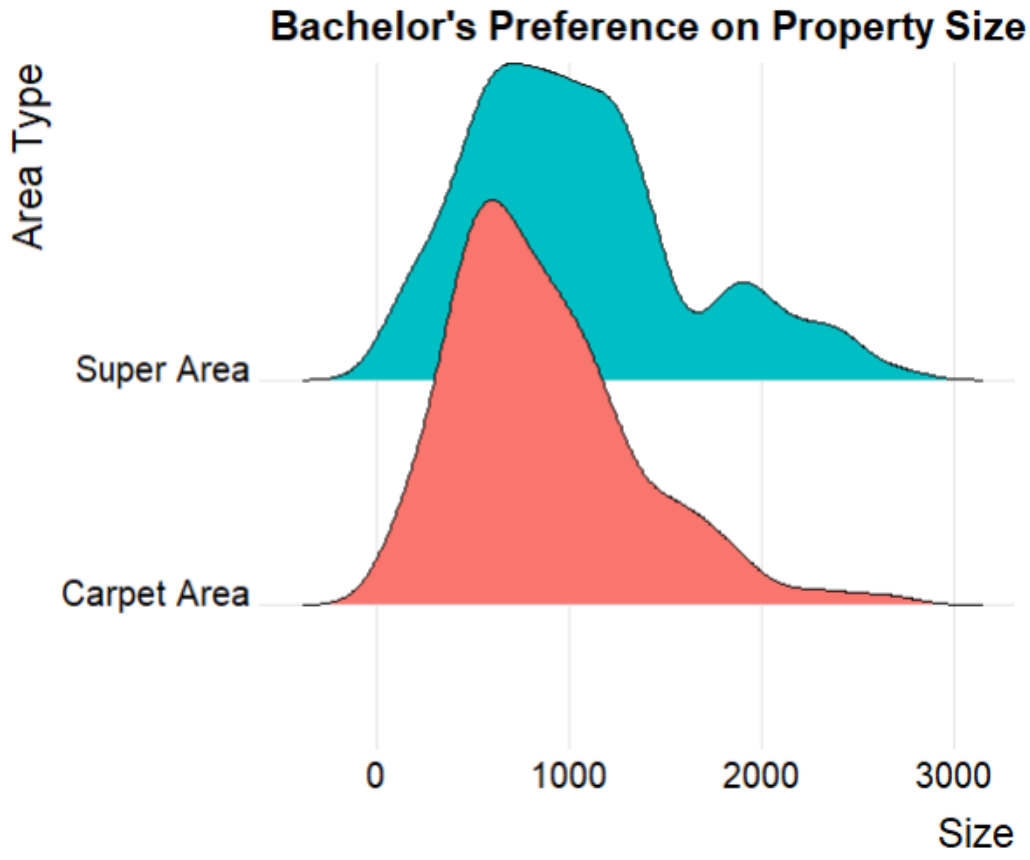


Figure 3.4.8: Plot for Analysis 4-4

From the ridgeline plot above we can see that both the lines peak around 800 square feet. For properties that are measured by super area, the line has a small rise at the 1900 square feet level.

Conclusion: Most of the bachelors prefer properties that are around 800 square feet.

Analysis 4-5: Find the families' preference on size of rental units.

The previous analysis is about bachelors. This analysis, meanwhile, is about the families. It aims to figure out the property size that families prefer.

```
data_temp <- subset(data_Q4, ! data_Q4$Tenant.Preferred == "Bachelors")

ggplot(data_temp, aes(x = Size, y = Area.Type, fill = Area.Type)) +
  geom_density_ridges() +
  theme_ridges() +
  theme(legend.position = "none") +
  ylab("Area Type") +
  ggtitle("Families's Preference on Property Size")
```

Figure 3.4.9: Visualizing families' preference on size of rental units.

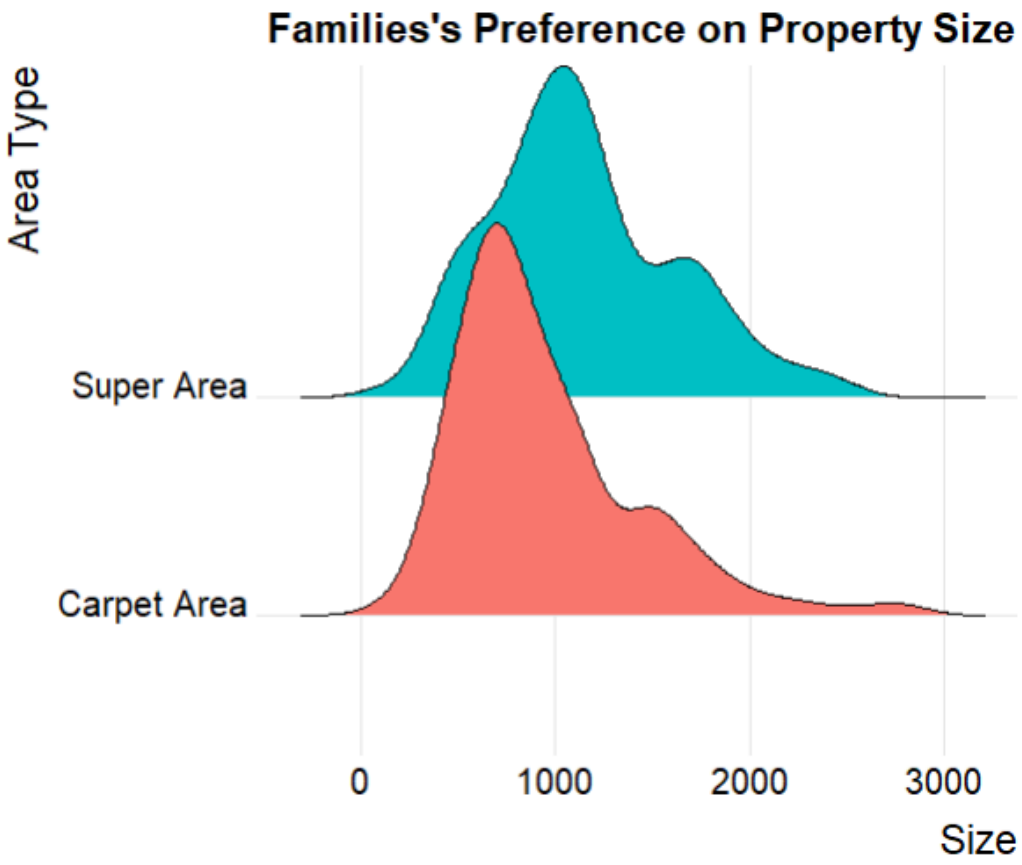


Figure 3.4.10: Plot for Analysis 4-5

The ridgeline plot above shows different peaks for the 2 different area types. The line peaks around 1050 square feet for super area while it is around 800 square feet for carpet area.

Conclusion: The properties that families prefer is a little bigger compared to the ones preferred by bachelors, which is around 1050 square feet for super area.

Analysis 4-6: Find the property level that bachelors prefer.

This analysis wishes to find out the preference of bachelors on the level of rental unit. In other words, figuring out which floor they would like to rent.

```
library(tidyverse)
library(hrbrthemes)

data_Q4 %>%
  filter( Tenant.Preferred == "Bachelors" & Floor != "Upper Basement" & Floor != "Lower Basement" ) %>%
  mutate(Floor= as.integer (Floor)) %>%
  ggplot( aes(x=Floor)) +
  geom_histogram( binwidth=1, fill="#69b3a2", color="#e9ecef", alpha=0.9) +
  ggtitle("Bachelors' Preference on Property Level") +
  theme_ipsum() +
  theme(
    plot.title = element_text(size=15)
  ) +
  ylab("Count") +
  xlab("Level")
```

Figure 3.4.11: Visualizing the property level that bachelors prefer

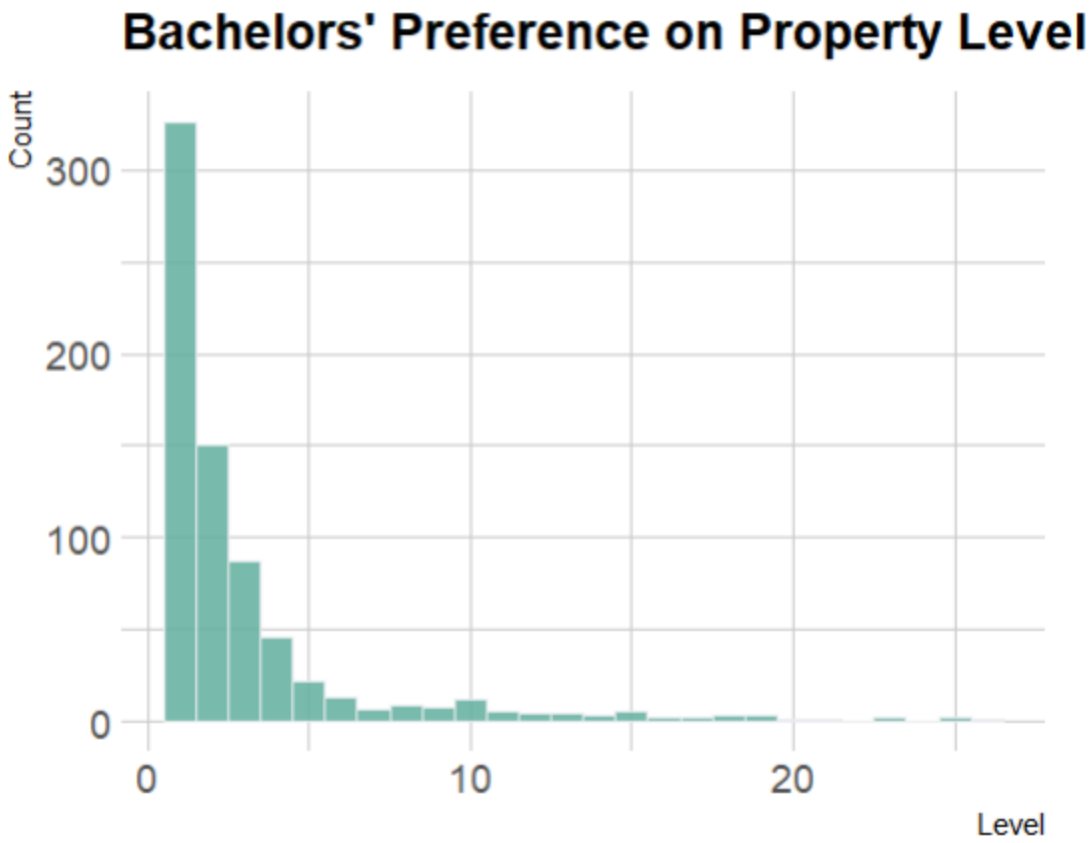


Figure 3.4.12: Plot for Analysis 4-6

The histogram above shows most of the bachelors prefer lower levels. The bar peaks at first floor and decreases linearly to 6-th floor.

Conclusion: Bachelors prefer lower floors.

Analysis 4-7: Find the property level that families prefer.

This analysis aims to figure out which level do the families prefer. The finding of this analysis might help academic researchers with their study on the topic of behavioral sciences.

```
# Analysis 4-7: Find the property level that families prefer.

data_Q4 %>%
  filter( Tenant.Preferred == "Family" & Floor != "Upper Basement" & Floor != "Lower Basement" ) %>%
  mutate(Floor= as.integer (Floor)) %>%
  ggplot( aes(x=Floor)) +
  geom_histogram( binwidth=1, fill="#69b3a2", color="#e9ecef", alpha=0.9) +
  ggtitle("Families' Preference on Property Level") +
  theme_ipsum() +
  theme(
    plot.title = element_text(size=15)
  ) +
  ylab("Count") +
  xlab("Level")
```

Figure 3.4.13: Visualizing the property level that families prefer.

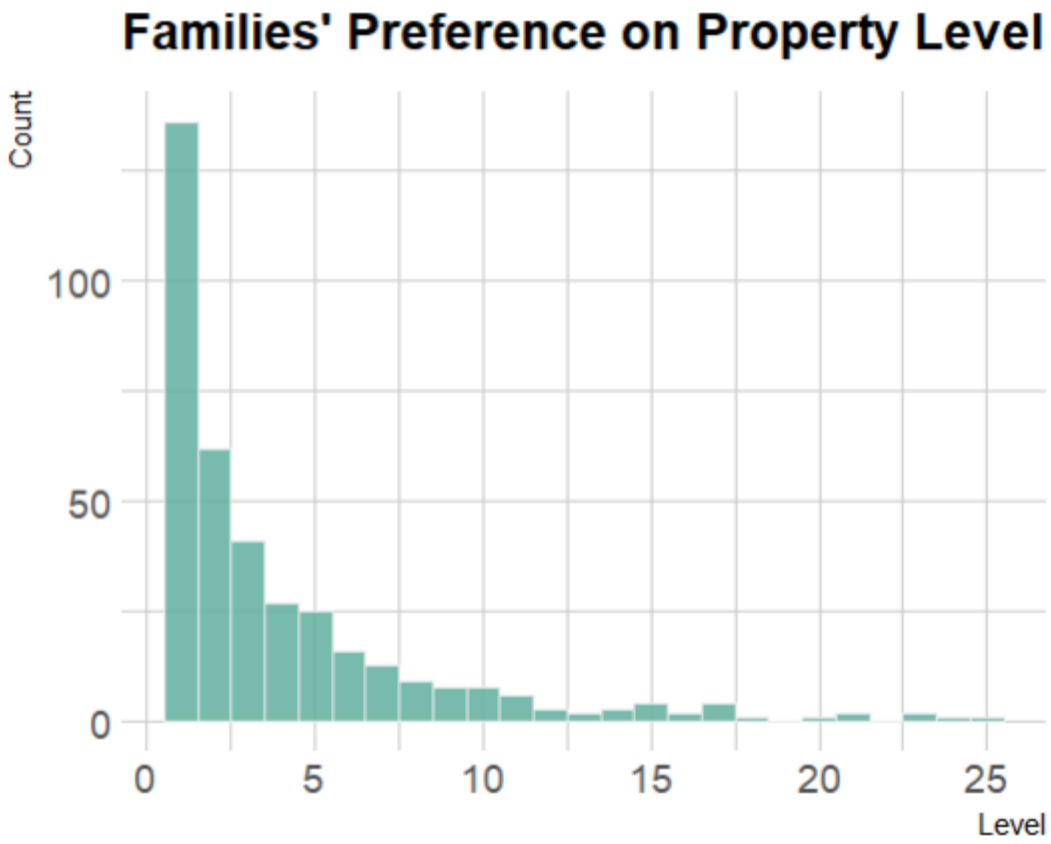


Figure 3.4.14: Plot for Analysis 4-7

From the histogram above we can see that it is similar to the bachelor's plot. Most of the families prefer lower levels as well. However, the decreasing trend of families is not as steep as the one of bachelors'.

Conclusion: Although families prefer rental units with lower level, some of them can accept high-level properties as well. Their acceptance level is higher than bachelors.

Analysis 4-8: Find out the cities that bachelors prefer.

Nowadays, many people have certain cities that they personally prefer. For example, youths usually prefer big cities because there are much more career growth opportunities. Hence, the aim of this analysis is to find out the cities that bachelors would like to stay in.

```
data_bachelor <- data_Q4 %>% filter(Tenant.Preferred == "Bachelors")
data_temp <- data.frame(
  city = sort(unique(data_bachelor$City)),
  count = c(
    length(which(data_bachelor$City == "Bangalore")),
    length(which(data_bachelor$City == "Chennai")),
    length(which(data_bachelor$City == "Delhi")),
    length(which(data_bachelor$City == "Hyderabad")),
    length(which(data_bachelor$City == "Kolkata")),
    length(which(data_bachelor$City == "Mumbai"))
  )
)

ggplot(data_temp, aes(x=city, y=count)) +
  geom_segment( aes(x=city, xend=city, y=0, yend=count)) +
  geom_point( size=5, color="red", fill=alpha("red", 0.3), alpha=0.7, shape=21, stroke=2) +
  ggtitle("City that Bachelors Prefer") +
  ylab("Count") +
  xlab("City")
```

Figure 3.4.15: Visualizing the cities that bachelors prefer.

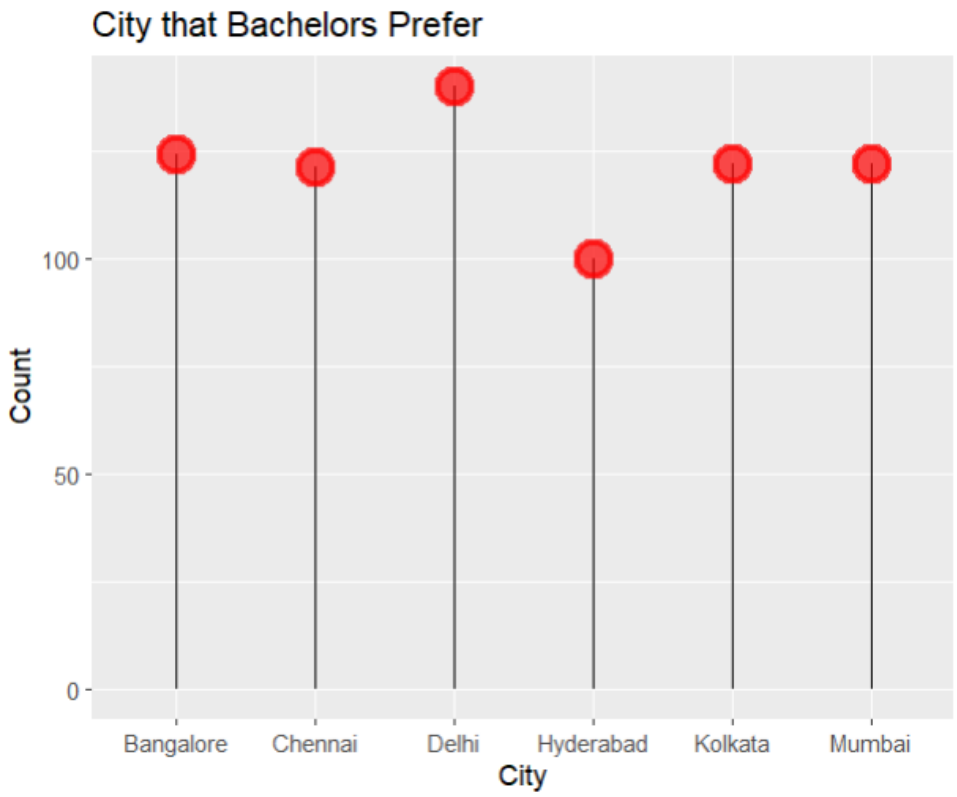


Figure 3.4.16: Plot for Analysis 4-8

From the lollipop chart above we can see that the longest lollipop is Delhi, followed by Bangalore. However, all the lollipops are not much different in their length.

Conclusion: Delhi is the city that most bachelors prefer. However, the preference level for different cities has not much difference. This means that city is not quite a factor that weight much when bachelors are to pick a rental unit.

Analysis 4-9: Find out the cities that families prefer.

The previous analysis is about bachelors. This analysis is about the families. The purpose of this analysis is to figure out the cities that families would prefer to stay in.

```
data_family <- data_Q4 %>% filter(Tenant.Preferred == "Family")
data_temp <- data.frame(
  city = sort(unique(data_family$City)),
  count = c(
    length(which(data_family$City == "Bangalore")),
    length(which(data_family$City == "Chennai")),
    length(which(data_family$City == "Delhi")),
    length(which(data_family$City == "Hyderabad")),
    length(which(data_family$City == "Kolkata")),
    length(which(data_family$City == "Mumbai"))
  )
)

ggplot(data_temp, aes(x=city, y=count)) +
  geom_segment( aes(x=city, xend=city, y=0, yend=count)) +
  geom_point( size=5, color="blue", fill=alpha("blue", 0.5), alpha=0.7, shape=21, stroke=2) +
  ggtitle("City that Families Prefer") +
  ylab("Count") +
  xlab("City")
```

Figure 3.4.17: Visualizing the cities that families prefer

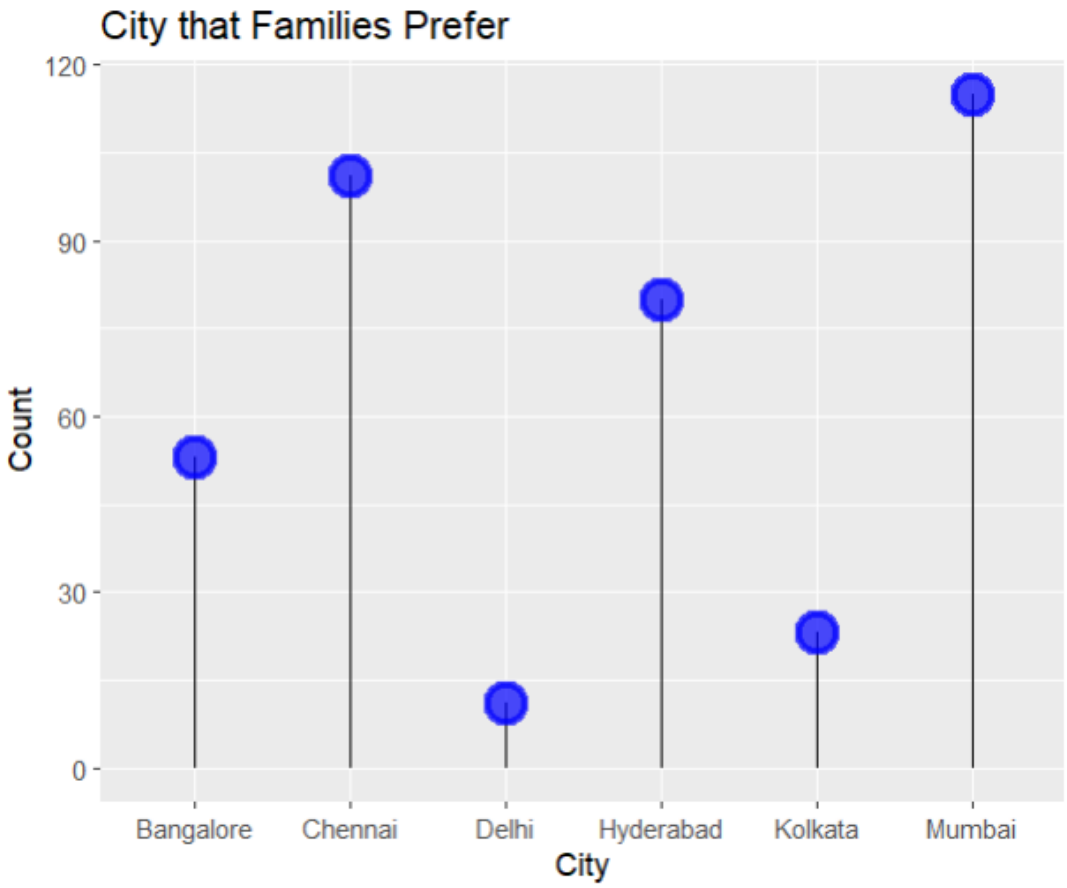


Figure 3.4.18: Plot for Analysis 4-9

From the lollipop plot above we can observe that Mumbai has the longest lollipop, followed by Chennai and Hyderabad. Interestingly, Delhi is the shortest one.

Conclusion: Mumbai and Chennai are the 2 cities that most families prefer. Delhi, on the other hand, is the least popular option for families. This might be due to the stressful work condition in Delhi.

Analysis 4-10: Find out what kind of furnishing status do bachelors and families prefer.

This analysis wishes to find out which level of furnishing status would the bachelors and families prefer. It is important in a way that it can act as the guideline for property owners to give them an idea on whether they should furnish their property.

```
tenant.type <- c(rep("Bachelors" , 3) , rep("Family" , 3))
furnishing.status <- rep(c("Unfurnished", "Semi-Furnished", "Furnished") , 2)
value <- c(
  length(which(data_Q4$Tenant.Preferred == "Bachelors" & data_Q4$Furnishing.Status == "Unfurnished")),
  length(which(data_Q4$Tenant.Preferred == "Bachelors" & data_Q4$Furnishing.Status == "Unfurnished")),
  length(which(data_Q4$Tenant.Preferred == "Bachelors" & data_Q4$Furnishing.Status == "Unfurnished")),

  length(which(data_Q4$Tenant.Preferred == "Family" & data_Q4$Furnishing.Status == "Unfurnished")),
  length(which(data_Q4$Tenant.Preferred == "Family" & data_Q4$Furnishing.Status == "Semi-Furnished")),
  length(which(data_Q4$Tenant.Preferred == "Family" & data_Q4$Furnishing.Status == "Furnished"))
)
data_temp <- data.frame(tenant.type,furnishing.status,value)

ggplot(data_temp, aes(fill=furnishing.status, y=value, x=tenant.type)) +
  geom_bar(position="fill", stat="identity") +
  ggtitle("Bachelor and Family's Preference on Furnishing Status") +
  labs(fill="Furnishing Status", y="Ratio", x="Tenant Type")
```

Figure 3.4.19: Visualizing the preferred furnishing status of bachelors and families

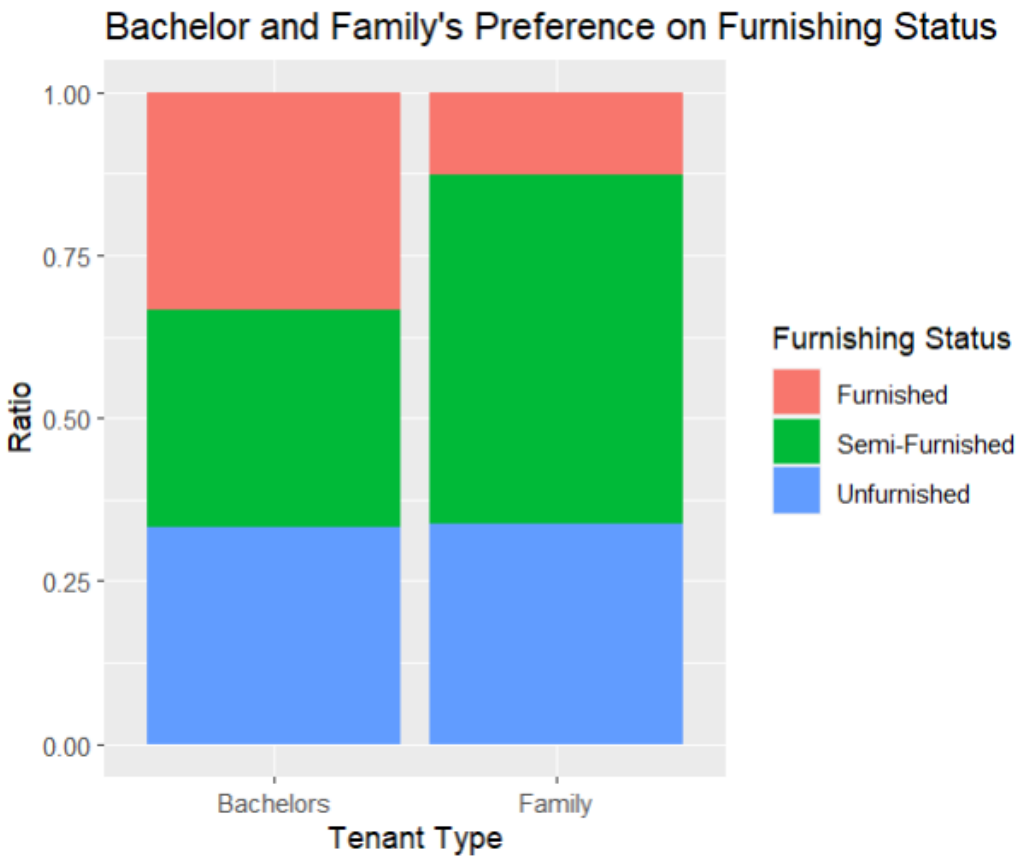


Figure 3.4.20: Plot for Analysis 4-10

The percent stacked graph above shows that bachelors are okay with properties with any kind of furnishing status. Families, on the other hand, prefer semi-furnished rental units and they do not like fully furnished units.

Conclusion: Bachelors do not care too much on rental unit's furnishing status. This might because bachelors do not have a partner to discuss with about how they should make the house cozy. Families, on the other hand, usually have higher intention to furnish their houses according to their wills. This is why they do not like rental units that are fully furnished.

Analysis 4-11: Find out the preferred way bachelors and families use to approach rental units.

This analysis is about figuring out the ways that bachelors and families prefer to approach the rental units that they like. In other words, who would they like to contact in order to know more about the rental unit they are interested in.

```
library(fmsb)

data_bachelor <- data_Q4 %>% filter(Tenant.Preferred == "Bachelors")
data_family <- data_Q4 %>% filter(Tenant.Preferred == "Family")
data_temp <- data.frame(
  owner = c(
    bachelor = length(which(data_bachelor$Point.of.Contact == "Contact Owner")),
    family = length(which(data_family$Point.of.Contact == "Contact Owner"))
  ),
  agent = c(
    bachelor = length(which(data_bachelor$Point.of.Contact == "Contact Agent")),
    family = length(which(data_family$Point.of.Contact == "Contact Agent"))
  ),
  builder = c(
    bachelor = length(which(data_bachelor$Point.of.Contact == "Contact Builder")),
    family = length(which(data_family$Point.of.Contact == "Contact Builder"))
  )
)

data_temp <- rbind(rep(390,3) , rep(0,3) , data_temp)

# Color vector
colors_border=c( rgb(0.2,0.5,0.5,0.9), rgb(0.8,0.2,0.5,0.9) , rgb(0.7,0.5,0.1,0.9) )
colors_in=c( rgb(0.2,0.5,0.5,0.4), rgb(0.8,0.2,0.5,0.4) , rgb(0.7,0.5,0.1,0.4) )

radarchart( data_temp, title = "Tenants' Preferred Point of Contact",
            #custom polygon
            pcol=colors_border , pfcol=colors_in , plwd=4 , plty=1
)

# Add a legend
legend(x=0.7, y=1, legend = rownames(data_temp[-c(1,2),]), bty = "n", pch=20 ,
       col=colors_in , text.col = "grey", cex=1.2, pt.cex=3)
```

Figure 3.4.21: Visualizing the preferred way bachelors and families use to approach rental units

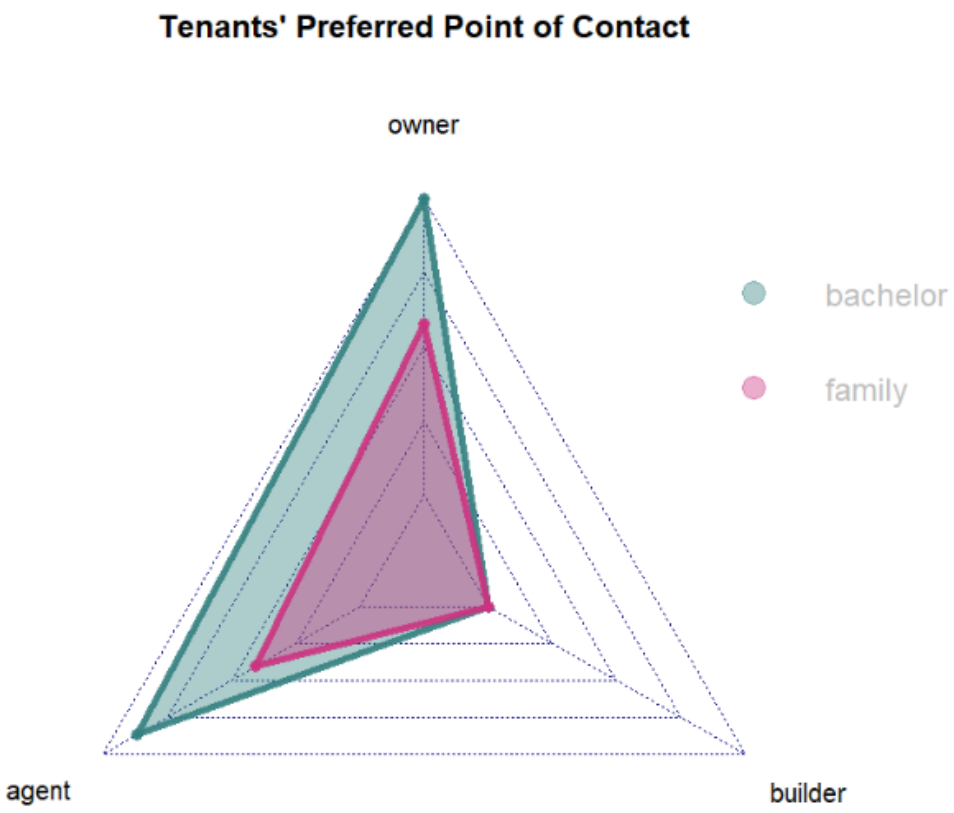


Figure 3.4.22: Plot for Analysis 4-11

From the radar chart above we can see that both the bachelors and families prefer to make contact with the owner of the rental units instead of going through an agent. Directly contact with property builders is not an option for the people.

Conclusion: People in India, regardless their marital status, prefer to make contact with rental units' owner directly. This might due to the lower rent thanks to the absence of a middleman's cut.

Analysis 4-12: Find out bachelors' preference on the height of property.

This analysis is about finding out the bachelors' preference on the height of property. This means to find out whether they prefer high-rise property, low-rise property, or landed property. It is an important step towards understanding their behaviors and lifestyles.

```
data_temp <- data_bachelor %>% count(Highest.Floor)

# Plot
ggplot(data_temp, aes(x=Highest.Floor, y=n)) +
  geom_line( color="#69b3a2", size=1, alpha=0.9) +
  theme_ipsum() +
  ggtitle("Bachelors' Preference on Height of Property") +
  xlab("Highest Level") +
  ylab("Count")
```

Figure 3.4.23: Visualizing the bachelors' preference on the height of property

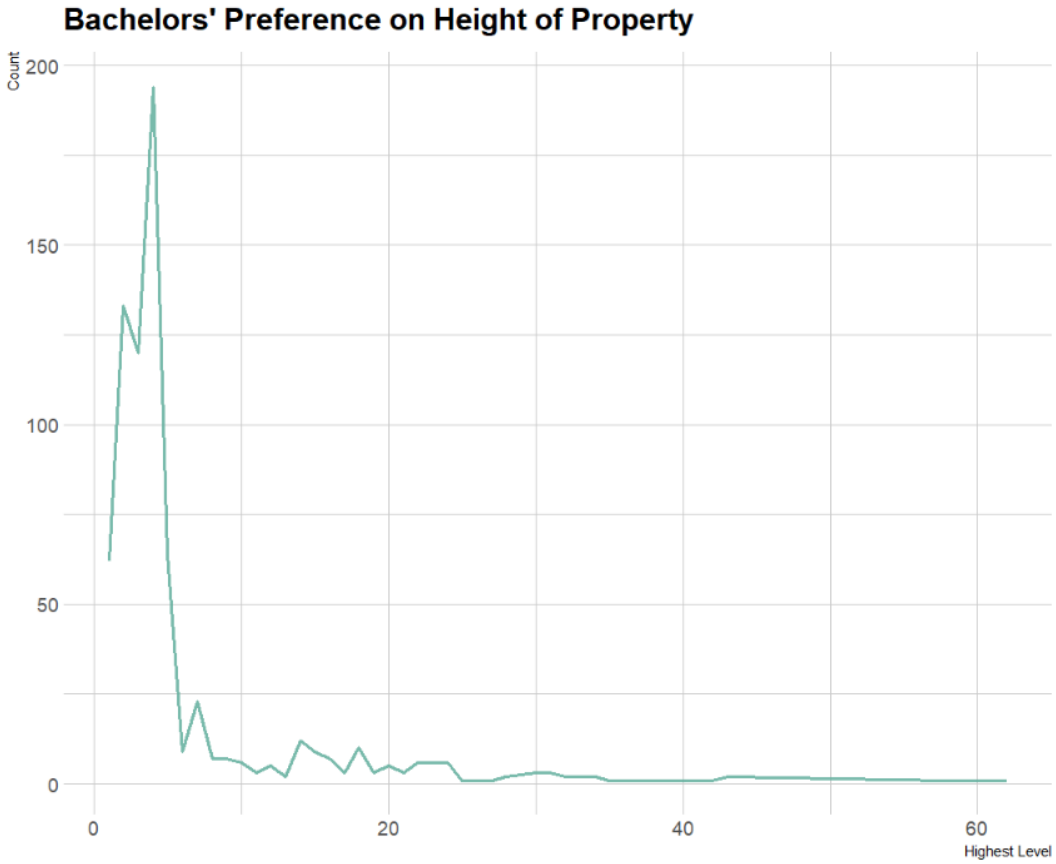


Figure 3.4.24: Plot for Analysis 4-12

From the line chart above we can see that the line peaks at 4-level property. Besides, we can also observe that the line drops very steeply after that.

Conclusion: Bachelors prefer low-rise properties, especially the ones which are below 6 floors.

Analysis 4-13: Find out families' preference on the height of property.

This analysis is about finding out the preference of families towards the height of property. This is as well significant for people to understand families' lifestyles and make comparison with the findings related to bachelors (the previous analysis).

```
data_temp <- data_family %>% count(Highest.Floor)

# Plot
ggplot(data_temp, aes(x=Highest.Floor, y=n)) +
  geom_line( color="#69b3a2", linewidth=1, alpha=0.9) +
  theme_ipsum() +
  ggtitle("Families' Preference on Height of Property") +
  xlab("Highest Level") +
  ylab("Count")
```

Figure 3.4.25: Visualizing the families' preference on the height of property

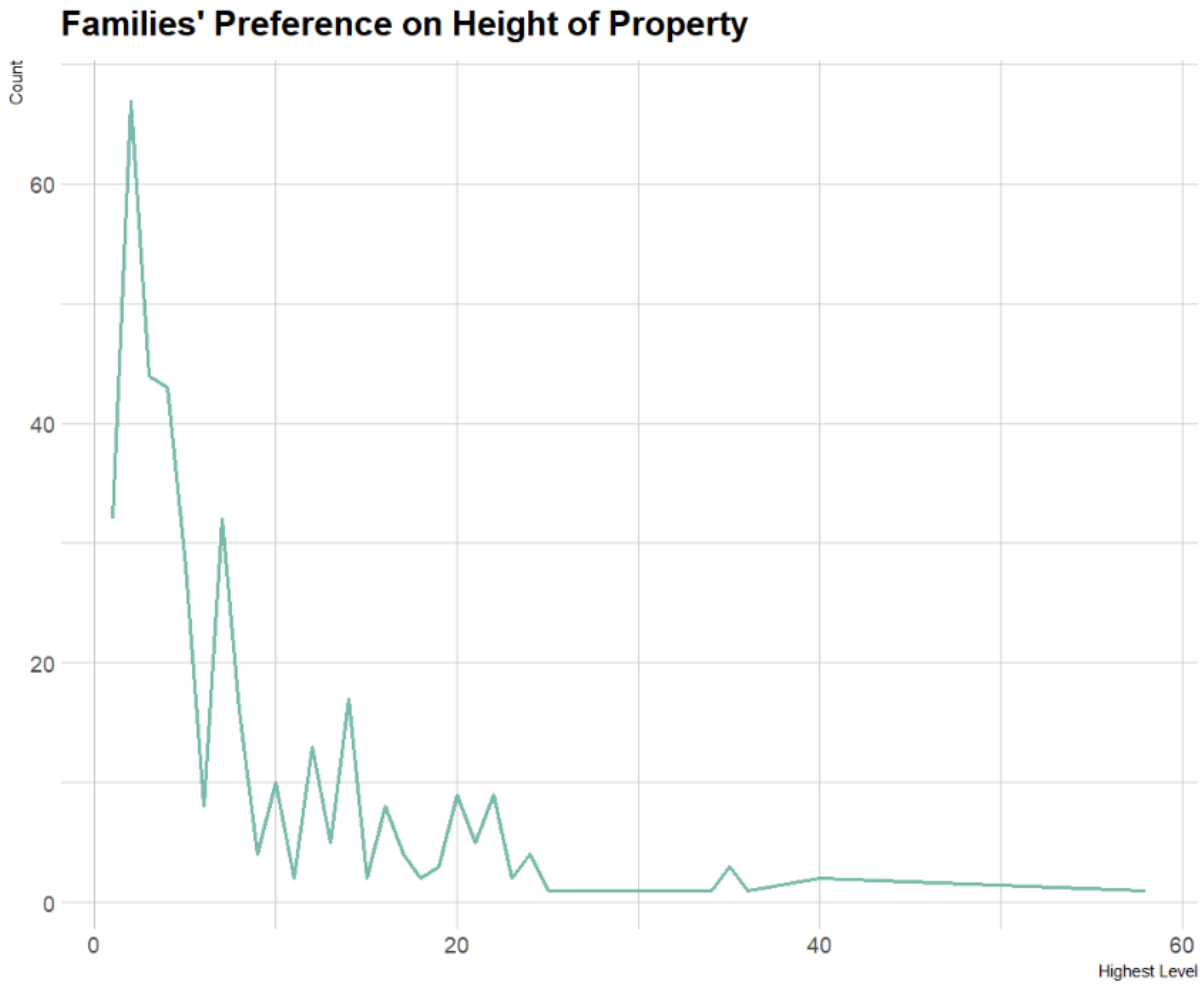


Figure 3.4.26: Plot for Analysis 4-13

From the line plot above we can see that the graph peaks at x=2 and drops steeply after that. The line fluctuates greatly in the range between x=6 to x=20.

Conclusion: Most of the families prefer double storey landed properties. Some also like high-rise properties that are below 20-floor height.

Conclusion for Question 4

1. Bachelors are a little more inclined towards properties with less number of bedrooms.
2. Families prefer properties with more number of bathrooms compared to bachelors.
3. Both bachelors and families prefer properties with lower rent.
4. Most of the bachelors prefer properties that are around 800 square feet.
5. Most of the families prefer properties that are around 1050 square feet.
6. Bachelors prefer lower floors.
7. Families prefer lower floors but have a higher acceptance level towards higher floors compared to bachelors.
8. Delhi is the city that most bachelors prefer. However, the preference level for different cities has not much difference.
9. Families prefer Mumbai and Chennai and do not like Delhi.
10. Bachelors do not care too much on rental unit's furnishing status.
11. Families do not like rental units that are fully furnished.
12. Most people in India, regardless their marital status, prefer to make contact with rental units' owner directly
13. Bachelors prefer low-rise properties, especially the ones which are below 6 floors.
14. Most of the families prefer double storey landed properties. Some also like high-rise properties that are below 20-floor height.

4.0 Extra Features

4.1 Data Preprocessing Procedures

Data preprocessing is fundamental for any given data analytic project. It is so important that all real projects will, deterministically, go through certain degree of data preprocessing to make sure the data is appropriate and suitable to be analysed.

4.1.1 Data Cleaning

Data cleaning is important for data analytics project in a way that it makes sure the data that will be used for analysis later has no inappropriate data such as null values and outlier values. For some data analytic algorithms, especially the advanced ones, dirty data will generate extremely inaccurate data. For example, machine learning algorithms such as linear regression, support vector machine, and K-nearest neighbors are very sensitive to outlier data (R, 2020).

Hence, as an extra feature, data cleaning procedure is added to this project. Null values have been checked for. Outlier values, on the other hand, has been detected using the IQR method. The R's built-in *subset()* function has been used to eliminate those outlier data.

4.1.2 Data Transformation

Data transformation is about transforming the dataset we have into something which might generate even more meaningful and accurate insights. It is crucial for any data analytic project in many ways. For example, by changing column names into something that are more self-explanatory, future data analysts in the team would not have to waste time figuring out what the columns mean anymore. Hence, data transformation not only can help data analyst team to generate better results, but also improve the overall efficiency of the team.

In this project, we have renamed the *BHK* column to *Bedroom*. It is more verbose and easier to understand. Besides, we have also split the *Floor* column into *Floor* and *Highest.Floor* column. Data splitting will help the analysis in future. Then, we have also replaced some inappropriate values exist in the dataset with data that are more suitable. For example, the *Floor* column has "*Ground*" and "*1*" values which basically mean the same thing. Hence, all the "*Ground*" values had been replaced with "*1*" so that the data will be more consistent.

Furthermore, some data type also had been changed for the sake of smoothening the data analytic process later. For instance, the *Highest.Floor* column which is originally in character data type was changed into numeric data type. Lastly, some relevant data that will aid the data analysis had also been added and combined with the original dataset. This includes data such as GDP, population, as well as the average housing price in every city that is involved in the dataset.

4.2 Extra Functions Used

4.2.1 summary()

Code

```
summary(data$BHK)
summary(data$Rent)
summary(data$Size)
summary(data$Bathroom)
```

Output

```
> summary(data$BHK)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   2.000   2.084   3.000   6.000
> summary(data$Rent)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1200   10000   16000   34993   33000 3500000
> summary(data$Size)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   10.0   550.0   850.0   967.5  1200.0  8000.0
> summary(data$Bathroom)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   2.000   1.966   2.000  10.000
```

Explanation

The summary() function is used to display the statistical data of a column in the dataset. Summarized information such as minimum, maximum, median, and mean are all shown.

4.2.2 unique()

Code

```
unique(data$Floor)
```

Output

```
[433] "Lower Basement out of 1"  "13 out of 17"             "1 out of 14"              "Upper Basement out of 2"
[437] "2 out of 14"              "24 out of 31"             "2 out of 32"              "2 out of 16"
[441] "9 out of 13"              "1 out of 11"              "6 out of 29"              "9 out of 9"
[445] "28 out of 31"             "1 out of 15"              "Ground out of 14"         "2 out of 11"
[449] "19 out of 31"             "1 out of 16"              "25 out of 32"             "11 out of 16"
[453] "11 out of 17"             "Upper Basement out of 3"  "4 out of 24"               "1 out of 19"
[457] "7 out of 30"              "16 out of 19"             "Upper Basement out of 5"  "Ground out of 13"
[461] "2 out of 25"              "23 out of 30"             "4 out of 30"              "13 out of 25"
[465] "23 out of 35"             "Ground out of 10"         "5 out of 34"              "20 out of 35"
[469] "1"                        "4 out of 31"              "4 out of 26"              "24 out of 33"
[473] "4 out of 17"              "1 out of 35"              "11 out of 35"             "Ground out of 15"
[477] "Ground out of 27"         "15 out of 30"             "12 out of 30"             "23 out of 34"
```

Explanation

It is used to display all the unique values of a column.

4.2.3 length()

Code

```
length(unique(data$Area.Locality))
```

Output

```
[1] 2235
```

Explanation

This function is used to find the number of elements in a vector.

4.2.4 str()

Code

```
> str(data)
```

Output

```
> str(data)
'data.frame':   4746 obs. of  12 variables:
 $ Posted.On        : chr  "5/18/2022" "5/13/2022" "5/16/2022" "7/4/2022" ...
 $ BHK              : int  2 2 2 2 2 2 2 1 2 2 ...
 $ Rent             : int  10000 20000 17000 10000 7500 7000 10000 5000 26000 10000 ...
 $ Size             : int  1100 800 1000 800 850 600 700 250 800 1000 ...
 $ Floor            : chr  "Ground out of 2" "1 out of 3" "1 out of 3" "1 out of 2" ...
 $ Area.Type        : chr  "Super Area" "Super Area" "Super Area" "Super Area" ...
 $ Area.Locality    : chr  "Bandel" "Phool Bagan, Kankurgachi" "Salt Lake City Sector 2" "Dumdum Park" ...
 $ City             : chr  "Kolkata" "Kolkata" "Kolkata" "Kolkata" ...
 $ Furnishing.Status: chr  "Unfurnished" "Semi-Furnished" "Semi-Furnished" "Unfurnished" ...
 $ Tenant.Preferred : chr  "Bachelors/Family" "Bachelors/Family" "Bachelors/Family" "Bachelors/Family" ...
 $ Bathroom         : int  2 1 1 1 1 2 2 1 2 2 ...
 $ Point.of.Contact : chr  "Contact Owner" "Contact Owner" "Contact Owner" "Contact Owner" ...
```

Explanation

The *str()* function returns the structure of the R object.

4.2.5 is.na()

Code

```
is.na(data)
```

Output

```
is.na(data)
      Posted.On   BHK  Rent  Size Floor Area.Type Area.Locality  City Furnishing.Status Tenant.Preferred Bathroom
 [1,]     FALSE FALSE FALSE FALSE FALSE     FALSE         FALSE FALSE              FALSE             FALSE    FALSE
 [2,]     FALSE FALSE FALSE FALSE FALSE     FALSE         FALSE FALSE              FALSE             FALSE    FALSE
 [3,]     FALSE FALSE FALSE FALSE FALSE     FALSE         FALSE FALSE              FALSE             FALSE    FALSE
 [4,]     FALSE FALSE FALSE FALSE FALSE     FALSE         FALSE FALSE              FALSE             FALSE    FALSE
 [5,]     FALSE FALSE FALSE FALSE FALSE     FALSE         FALSE FALSE              FALSE             FALSE    FALSE
 [6,]     FALSE FALSE FALSE FALSE FALSE     FALSE         FALSE FALSE              FALSE             FALSE    FALSE
 [7,]     FALSE FALSE FALSE FALSE FALSE     FALSE         FALSE FALSE              FALSE             FALSE    FALSE
 [8,]     FALSE FALSE FALSE FALSE FALSE     FALSE         FALSE FALSE              FALSE             FALSE    FALSE
 [9,]     FALSE FALSE FALSE FALSE FALSE     FALSE         FALSE FALSE              FALSE             FALSE    FALSE
[10,]     FALSE FALSE FALSE FALSE FALSE     FALSE         FALSE FALSE              FALSE             FALSE    FALSE
```

Explanation

This function returns TRUE if the given data is null value.

4.2.6 colSums()

Code

```
colSums(is.na(data))
```

Output

```
colSums(is.na(data))
     Posted.On            BHK           Rent           Size          Floor      Area.Type
             0              0              0              0              0              0
 Area.Locality           City Furnishing.Status Tenant.Preferred      Bathroom Point.of.Contact
             0              0              0              0              0              0
```

Explanation

colSums() function returns the sum of values in every column.

4.2.7 quantile()

Code

```
Q_rent <- quantile(data$Rent, probs=c(.25, .75))
```

Output

```
  25%   75%
10000 33000
```

Explanation

This function returns the quantiles of any given data. The probs argument defines which quantile to compute.

4.2.8 IQR()

Code

```
iqr_rent <- IQR(data$Rent)
```

Output

```
[1] 23000
```

Explanation

The *IQR()* function returns the interquartile range of any given data.

4.2.9 str_split_fixed()

Code

```
str_split_fixed(cleaned_dataset$Floor, " out of ", 2)
```

Output

```
[487,] "Ground"          "2"
[488,] "5"               "10"
[489,] "11"              "14"
[490,] "Ground"          "1"
[491,] "Ground"          "2"
[492,] "2"               "3"
[493,] "Ground"          "4"
[494,] "Ground"          "1"
[495,] "2"               "3"
[496,] "Ground"          "2"
[497,] "Ground"          "2"
[498,] "Ground"          "3"
[499,] "1"               "4"
[500,] "3"               "3"
[ reached getOption("max.print") -- omitted 4246 rows ]
```

Explanation

This function help to split a column at a given delimiter.

4.2.10 replace()

Code

```
cleaned_dataset$Floor <- replace(cleaned_dataset$Floor, cleaned_dataset$Floor=="Ground", "1")
```

Output

"*Ground*" values being replaced with "*1*".

```
> unique(cleaned_dataset$Floor)
 [1] "1"              "2"              "4"              "3"              "5"              "7"
 [7] "8"              "Upper Basement" "11"             "Lower Basement" "6"              "14"
[13] "9"              "19"             "34"             "12"             "26"             "13"
[19] "16"             "10"             "18"             "20"             "17"             "15"
[25] "21"             "25"             "48"             "23"             "30"             "24"
[31] "28"
```

Explanation

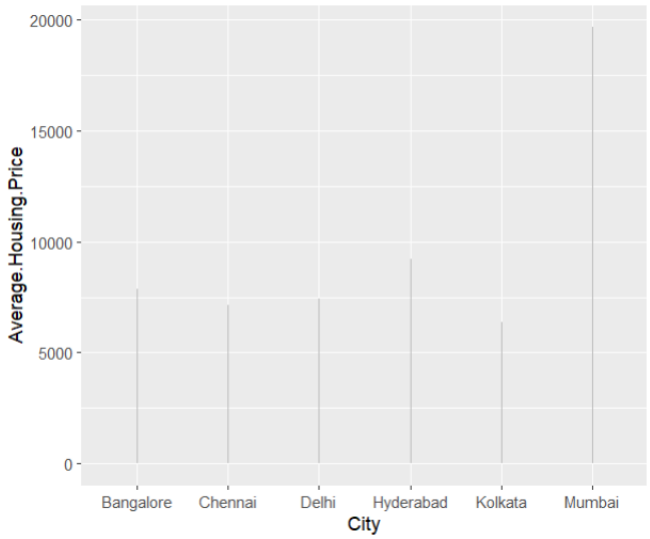
This function searches for the given values and replace them with the new values given.

4.2.11 geom_segment()

Code

```
geom_segment( aes(x=City, xend=City, y=0, yend=Average.Housing.Price), color="grey")
```

Output



Explanation

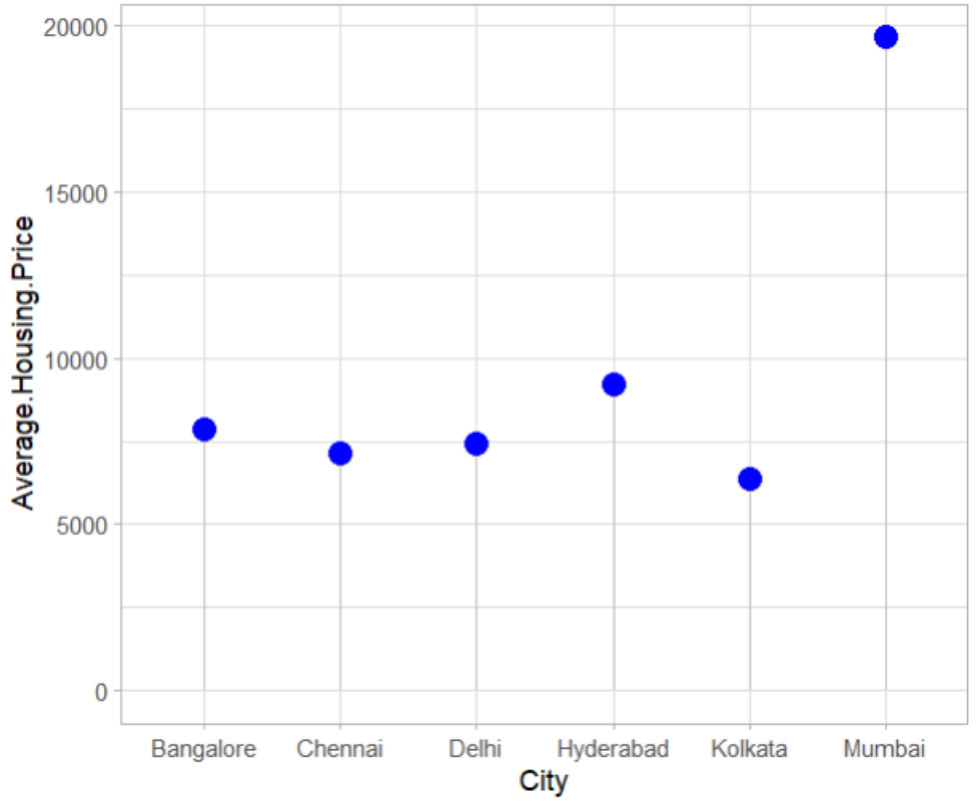
Used along with the ggplot2 library to draw straight lines between the given points.

4.2.12 theme_light()

Code

```
theme_light()
```

Output



Explanation

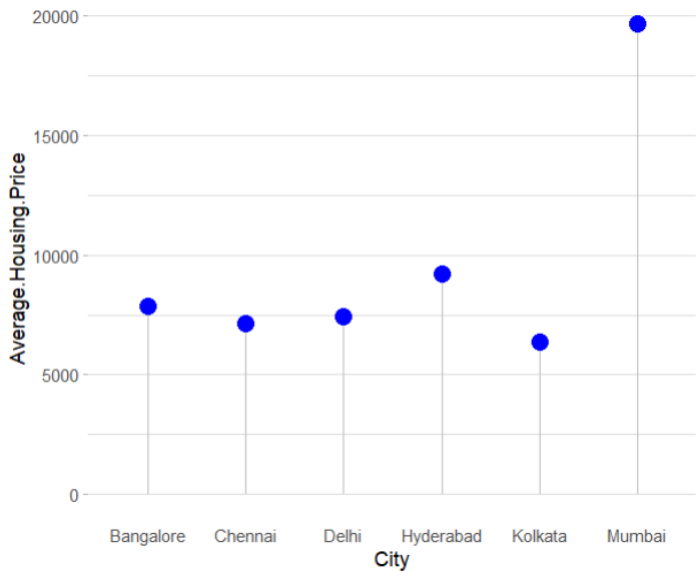
This function makes the background white. Meanwhile, axis and lines will be grey in color.

4.2.13 theme()

Code

```
theme(
  panel.grid.major.x = element_blank(),
  panel.border = element_blank(),
  axis.ticks.x = element_blank()
) +
```

Output



Explanation

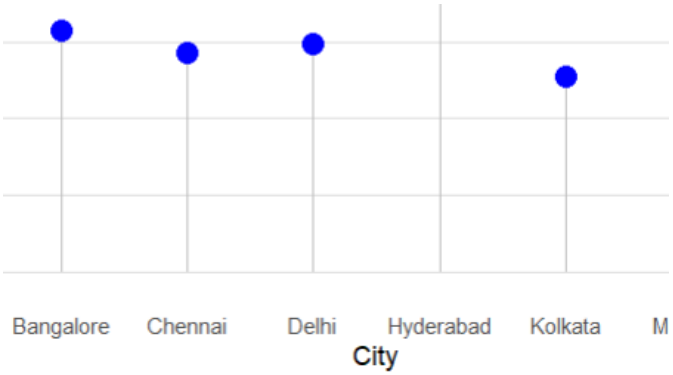
This function let us define more specifically the theme of the plot.

4.2.14 xlab()

Code

```
xlab("City")
```

Output



Explanation

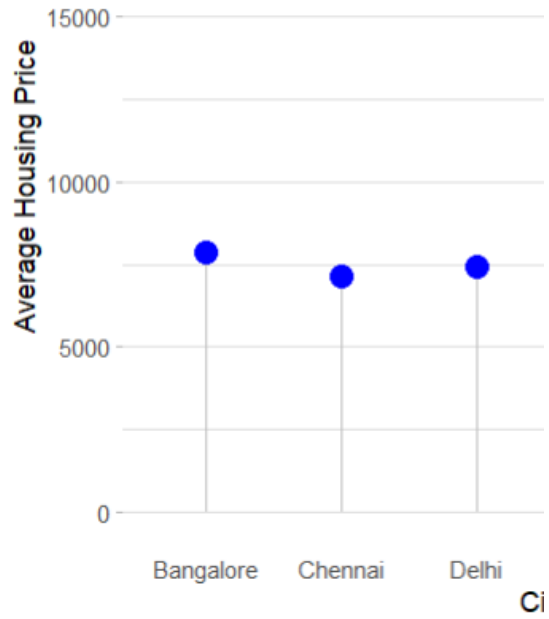
This function defines the label of the x-axis of the plot.

4.2.15 ylab()

Code

```
ylab("Average Housing Price")
```

Output



Explanation

This function defines the label of the y-axis of the plot.

4.2.16 pull()

Code

```
pull(price_rent, rent))
```

Output

```
[1] 16734.67 16351.95 19259.96 17264.27 11323.27 36723.33
```

Explanation

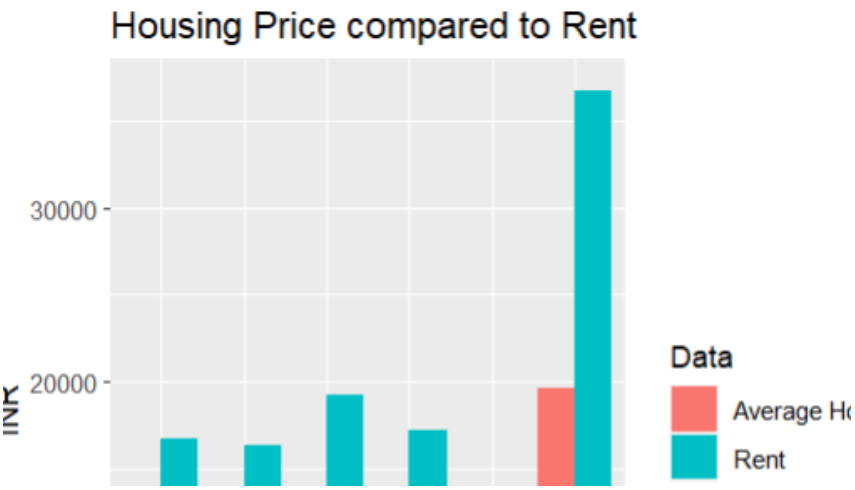
This function extracts the column of an R object.

4.2.17 ggtitle()

Code

```
ggtitle("Housing Price compared to Rent")
```

Output



Explanation

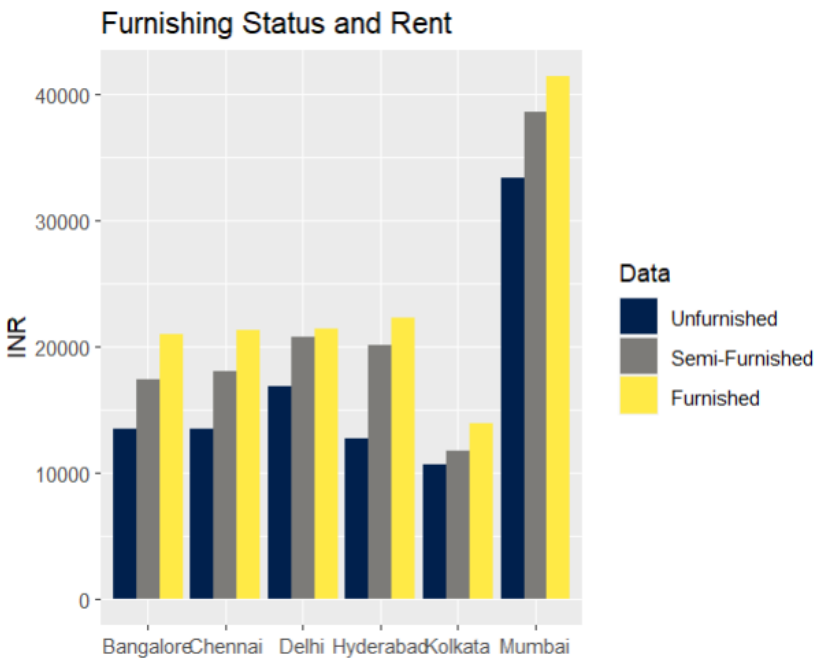
This function alters the title of a plot.

4.2.18 scale_fill_viridis()

Code

```
scale_fill_viridis(discrete = T, option = "E")
```

Output



Explanation

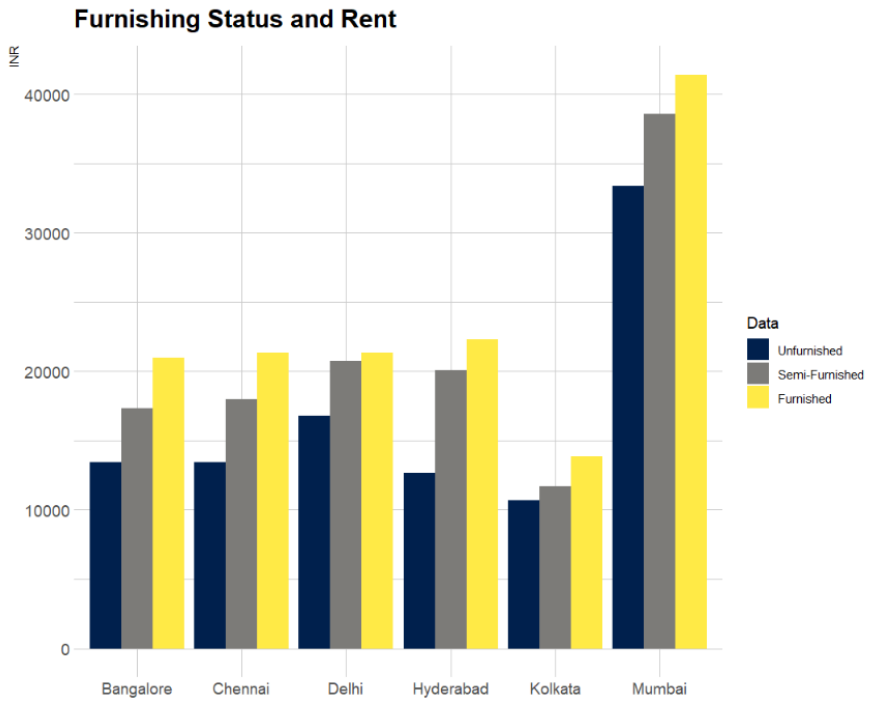
This function gives nice colors to plot. There are in total 8 options to be chosen from. The discrete argument let us decide if we want continuous color palette or a discrete one.

4.2.19 theme_ipsum()

Code

```
theme_ipsum()
```

Output



Explanation

This function gives a different theme to the plot.

4.2.20 which()

Code

```
length(which(cleaned_dataset$Point.of.Contact == "Contact Owner" & cleaned_dataset$City == "Bangalore"))
```

Output

```
[1] 750
```

Explanation

This function returns the rows that comply with the conditions given.

4.2.21 cumsum()

Code

```
cumsum(data_temp$fraction)
```

Output

```
[1] 0.1176471 0.1176471 1.0000000
```

Explanation

This function returns the cumulative sum of an R object.

4.2.22 paste0()

Code

```
paste0(data_temp$category, "(", round(data_temp$fraction*100) , "%)")
```

Output

```
[1] "Contact Agent(12%)"  "Contact Builder(0%)" "Contact Owner(88%)"
```

Explanation

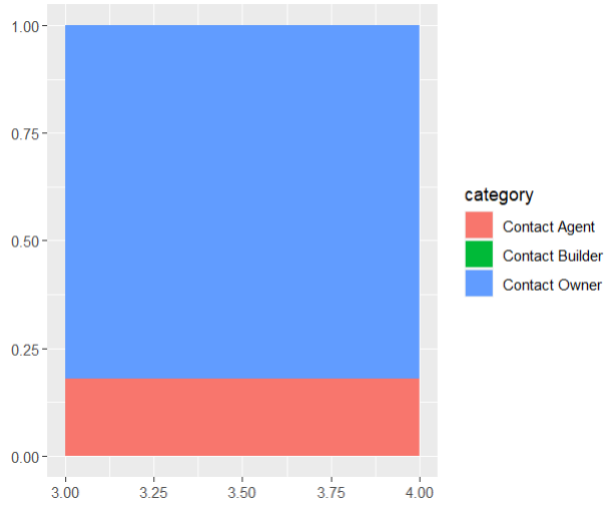
This function concatenates vector values and generate new strings.

4.2.23 geom_rect()

Code

```
geom_rect()
```

Output



Explanation

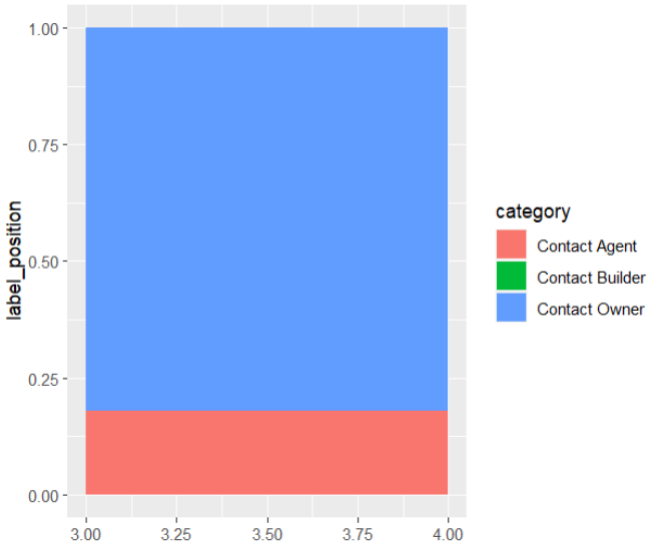
This function plot a rectangular graph by using the 4 corners (xmin, xmax, ymin, ymax).

4.2.24 geom_text()

Code

```
geom_text( x=1.5, aes(y=label_position, label=label, color=category), size=6)
```

Output



Explanation

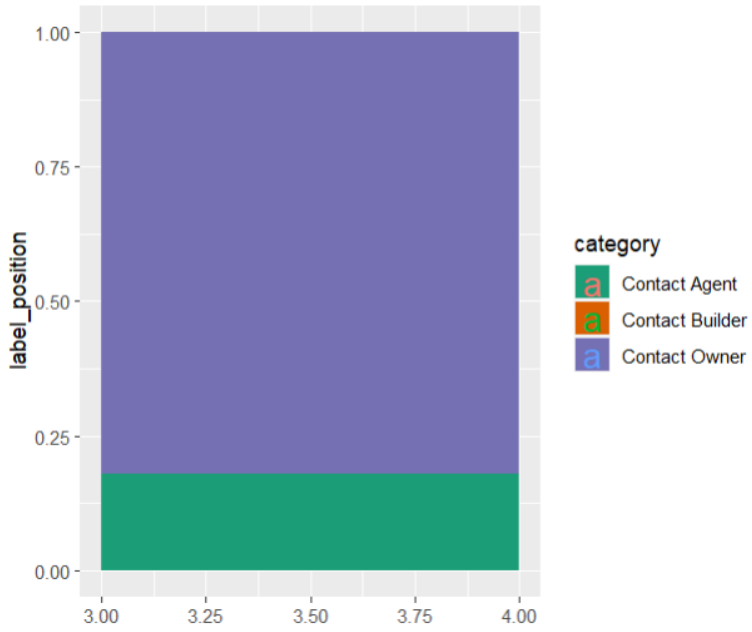
This function generates labels for plot.

4.2.25 scale_fill_brewer()

Code

```
scale_fill_brewer(palette="Dark2")
```

Output



Explanation

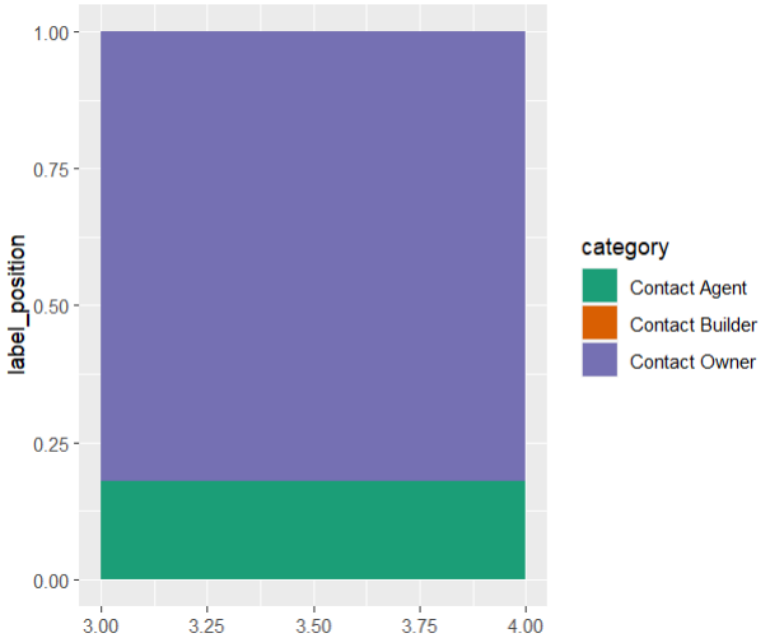
This function gives the plot different colors based on the palette determined.

4.2.26 scale_color_brewer()

Code

```
scale_color_brewer(palette="Dark2")
```

Output



Explanation

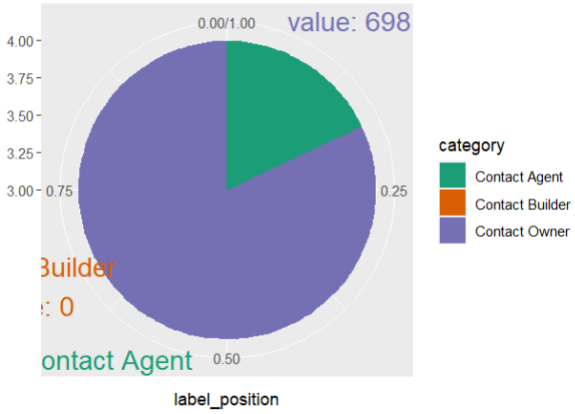
This function changes the color of the border of the plot.

4.2.27 coord_polar()

Code

```
coord_polar(theta="y")
```

Output



Explanation

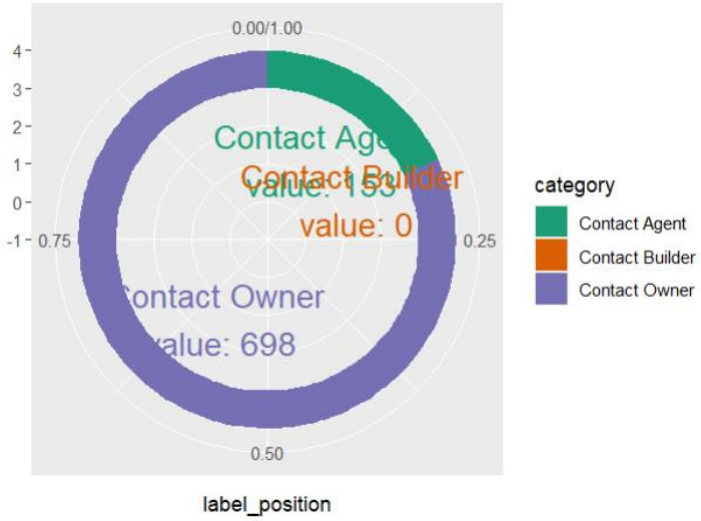
This function changes the rectangular bar plot into pie chart (polar).

4.2.28 xlim()

Code

```
xlim(c(-1, 4))
```

Output



Explanation

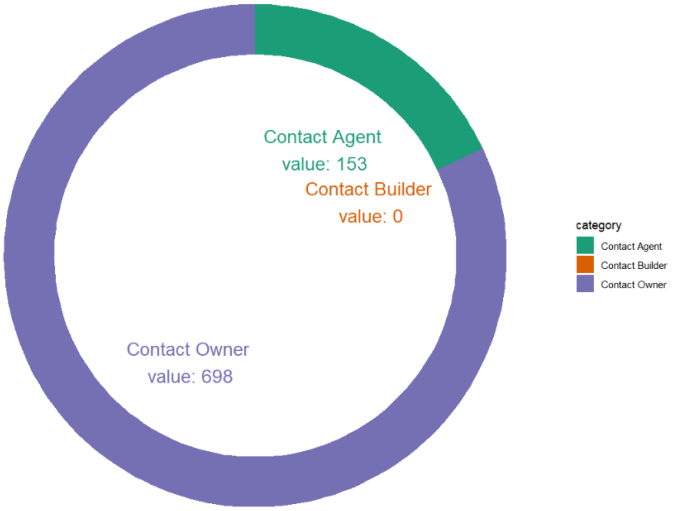
This function sets the limit of the x-axis, which in our case, helps to create a donut chart.

4.2.29 theme_void()

Code

```
theme_void()
```

Output



Explanation

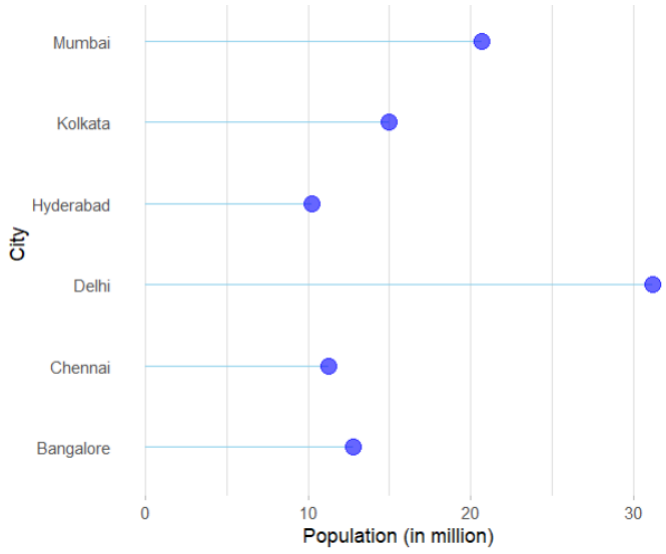
This function applies another theme – which makes the noisy background disappear to the plot.

4.2.30 coord_flip()

Code

```
coord_flip()
```

Output



Explanation

This function exchanges the two axes.

4.2.31 rbind()

Code

```
rbind(rep(max,6) , rep(0,6) , data_temp)
```

Output

	Bangalore	Chennai	Delhi	Hyderabad	Kolkata	Mumbai
1	3.56	3.56	3.56	3.56	3.56	3.56
2	0.00	0.00	0.00	0.00	0.00	0.00
3	3.56	2.41	2.94	2.64	0.84	1.26

Explanation

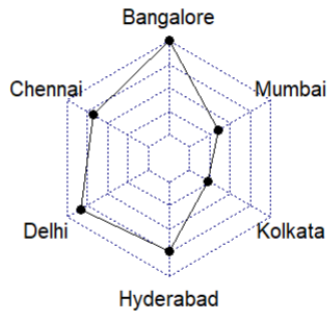
This function stands for row-bind. It is used to combine R objects such as vectors and data frames. In our case, we used it to add vectors into a data frame.

4.2.32 radarchart()

Code

```
radarchart(data_temp)
```

Output



Explanation

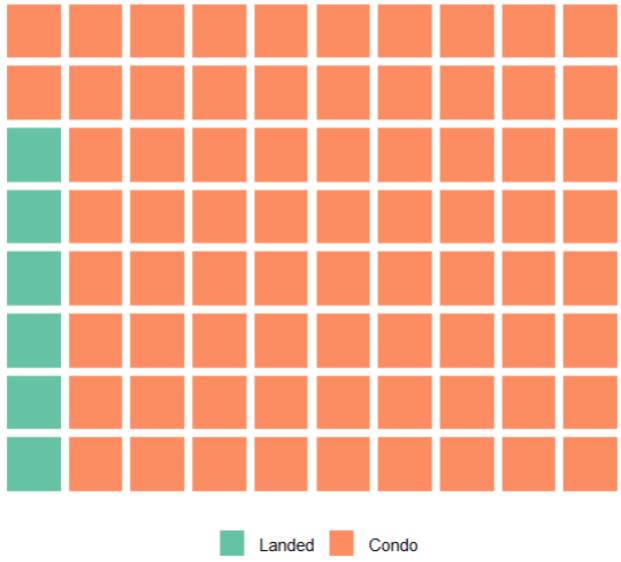
This function generates a radar chart based on the data given.

4.2.33 waffle()

Code

```
waffle(data_temp, rows = 8,
       legend_pos = "bottom")
```

Output



Explanation

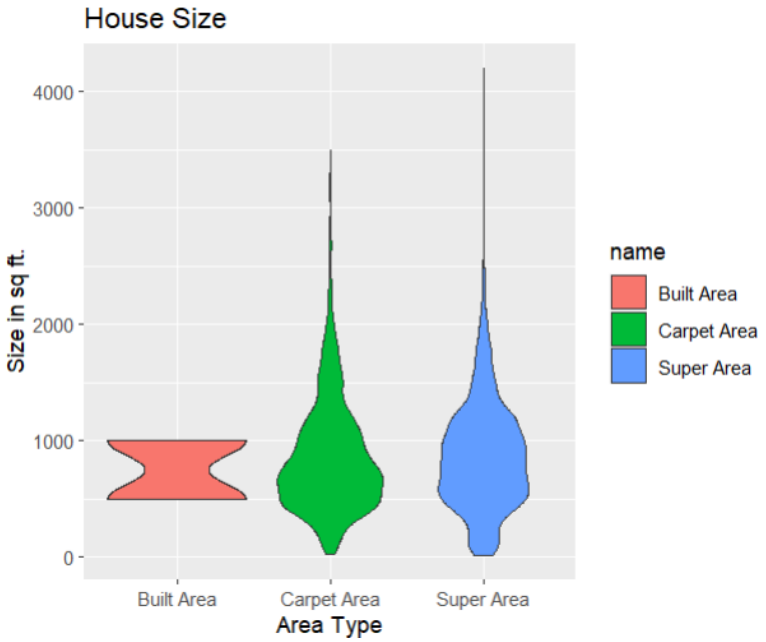
This function generates a waffle chart based on the data given.

4.2.34 geom_violin()

Code

```
geom_violin()
```

Output



Explanation

This function generates a violin plot.

4.2.35 row.names()

Code

```
row.names(data_temp) <- c(sort(unique(cleaned_dataset$City)))
```

Output

```
[1] "Bangalore" "Chennai"   "Delhi"     "Hyderabad" "Kolkata"   "Mumbai"
```

Explanation

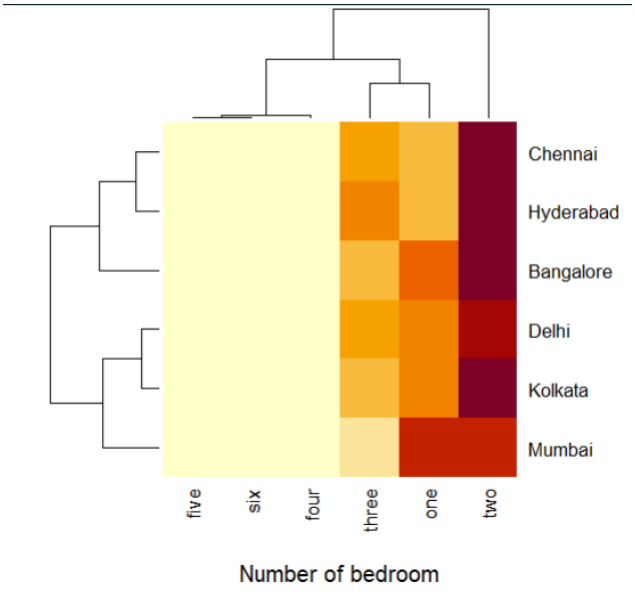
This function sets the row names for the rows.

4.2.36 heatmap()

Code

```
heatmap(as.matrix(data_temp), cexRow = 1, cexCol = 1, xlab = "Number of bedroom")
```

Output



Explanation

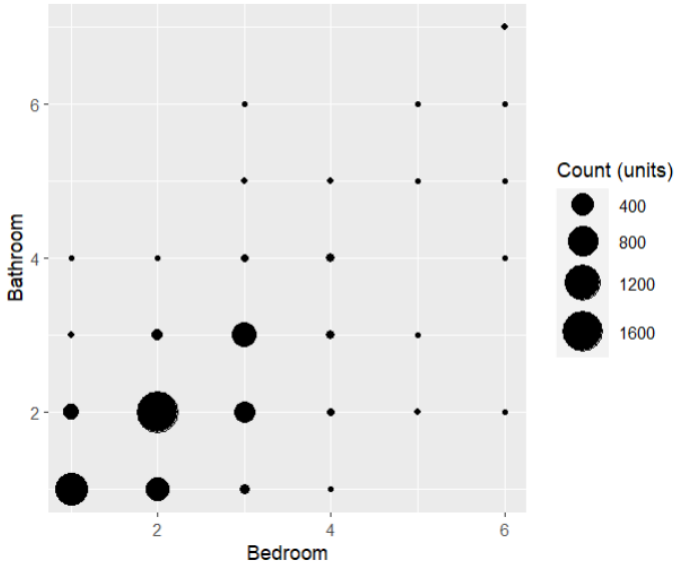
This function generates a heatmap plot based on the given data.

4.2.37 scale_size()

Code

```
scale_size(range = c(1, 10), name="Count (units)")
```

Output



Explanation

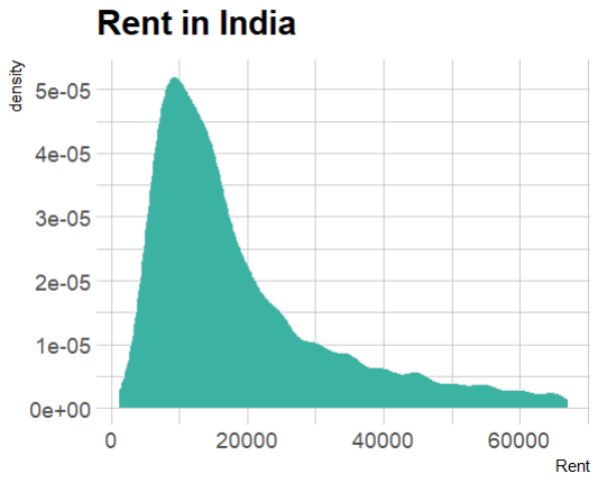
This function gives the plot different scale size.

4.2.38 geom_density()

Code

```
geom_density(fill="#3cb3a2", color="#e9ecef")
```

Output



Explanation

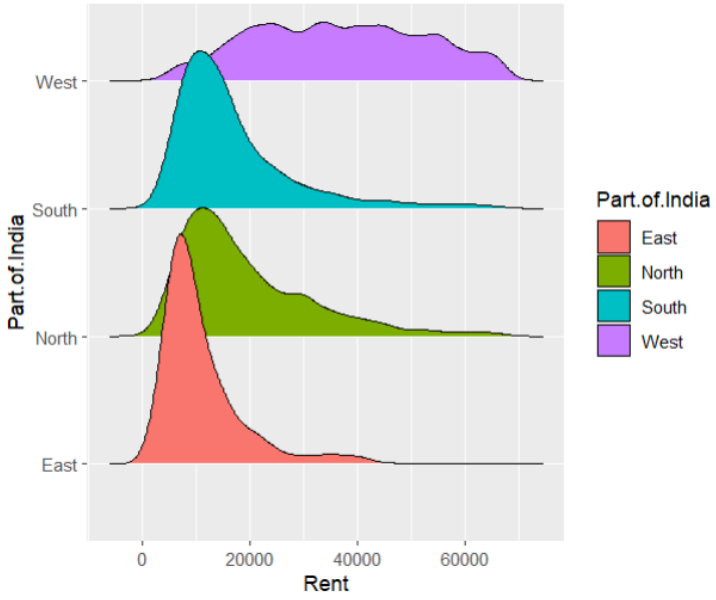
This function generates a density plot based on the given data.

4.2.39 geom_density_ridges()

Code

```
geom_density_ridges()
```

Output



Explanation

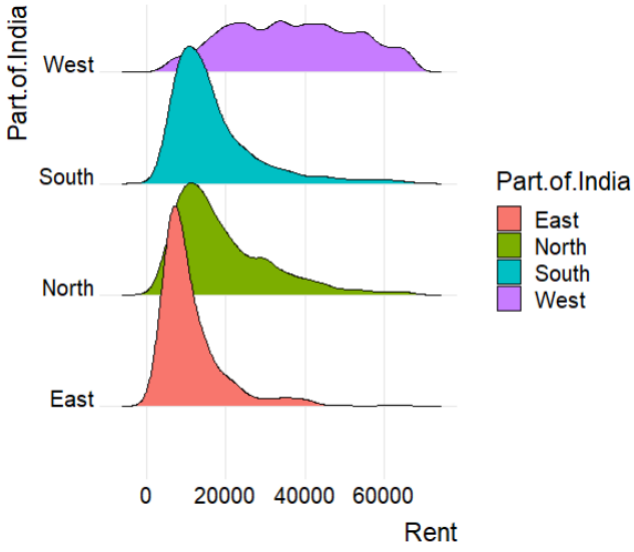
This function generates ridgeline plot from the given data.

4.2.40 theme_ridges()

Code

```
theme_ridges()
```

Output



Explanation

This function gives a better-looking theme to the ridgeline plot.

4.2.41 geom_smooth()

Code

```
geom_smooth(method=lm , color="red", se=FALSE)
```

Output

Explanation

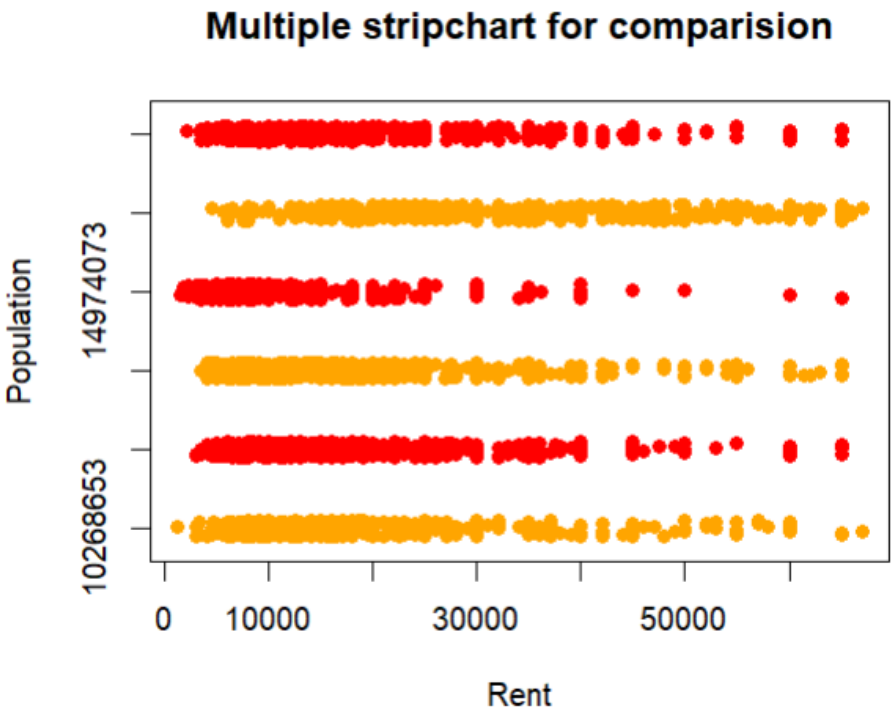
This function gives a trend line to the scatter plot.

4.2.42 stripchart()

Code

```
stripchart(data_temp,
           main="Multiple stripchart for comparision",
           xlab="Rent",
           ylab="Population",
           method="jitter",
           col=c("orange","red"),
           pch=16
)
```

Output



Explanation

This function generates a strip chart based on the given data.

4.2.43 facet_wrap()

Code

```
facet_wrap(~Tenant.Preferred)
```

Output

Explanation

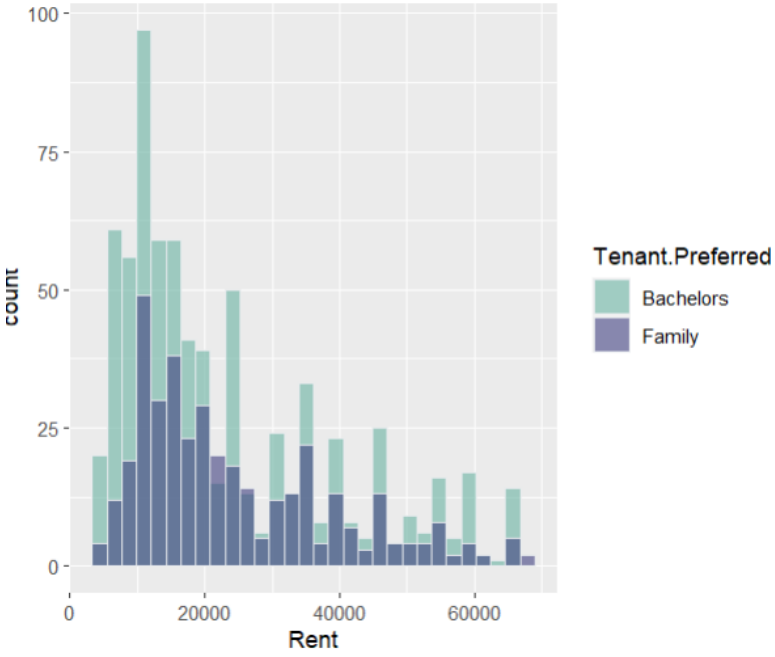
This function makes the plot being displayed in multiple panels.

4.2.44 scale_fill_manual()

Code

```
scale_fill_manual(values=c("#69b3a2", "#404080"))
```

Output



Explanation

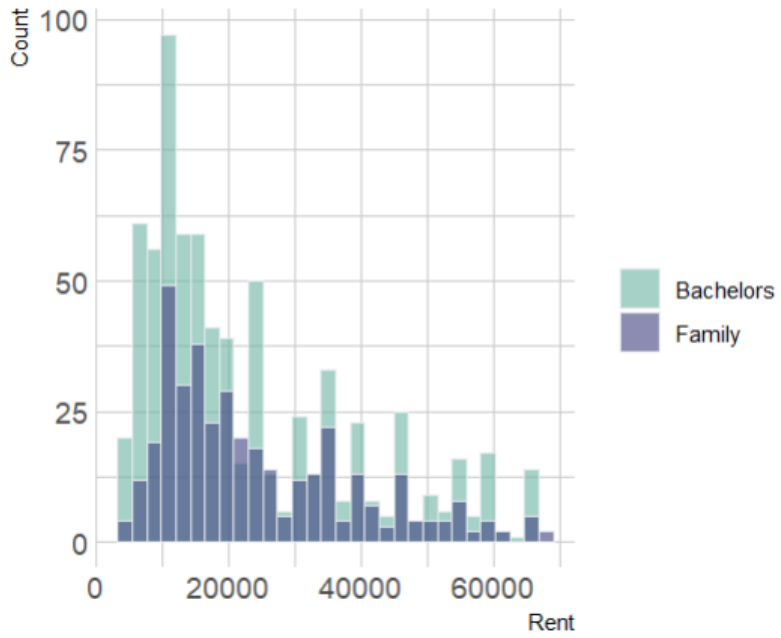
This function gives the plot colors manually.

4.2.45 labs()

Code

```
labs(fill="", y="Count")
```

Output



Explanation

This function lets us define all the labels on the plot at one place.

4.2.46 rgb()

Code

```
rgb(0.2,0.5,0.5,0.9)
```

Output

```
"#338080E6"
```

Explanation

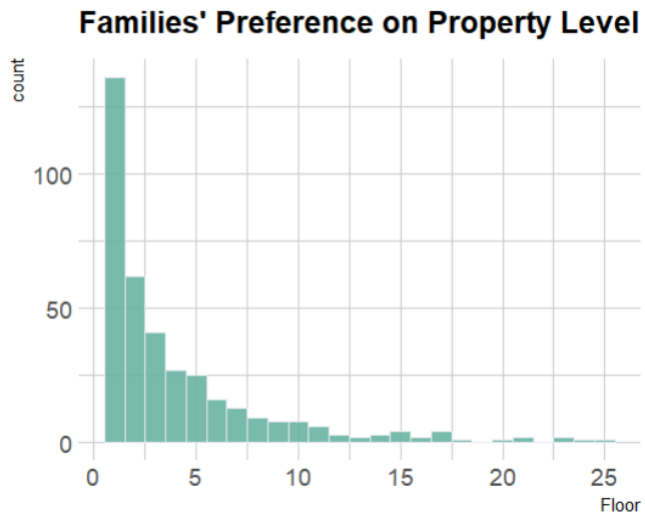
This function creates colors based on the values given.

4.2.47 element_text()

Code

```
plot.title = element_text(size=15)
```

Output



Explanation

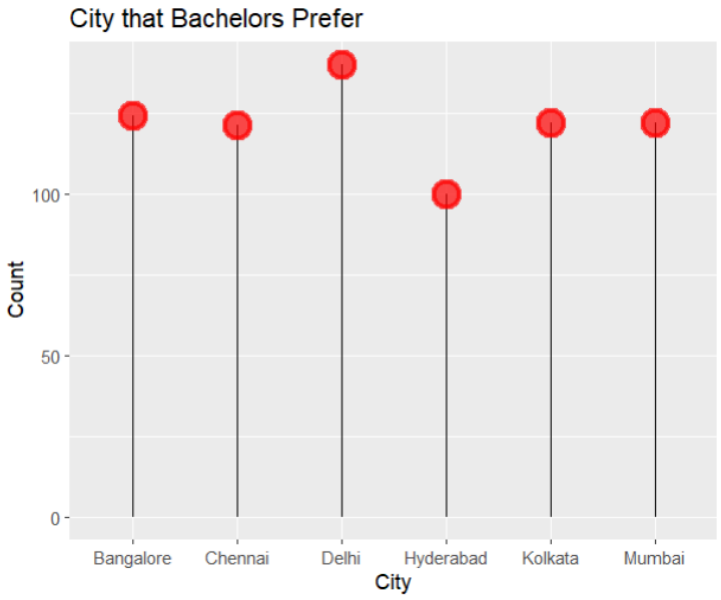
This function is used to gives texts nicer-looking themes.

4.2.48 alpha()

Code

```
fill=alpha("red", 0.3)
```

Output



Explanation

This function sets the color as well as the alpha (transparency).

5.0 Conclusion

This project has come to an end. To conclude, this project includes a total of 58 analysis, 57 plots had been created in order to aid the corresponding analysis. The questions of this project revolve around some of the most important aspects of India – economy and social status of the nation.

5.1 Personal Reflection Report

This project has provided me with an opportunity to practically carry out data analysis using a newly learnt programming language – R. Generally speaking, the learning curve of R is not very steep compared to other technologies. However, I am impressed with the abilities of R. Before learning this module, I only know how to carry out data analysis graphically, thanks to the Introduction to Data Analysis (IDA) module during the diploma study in Asia Pacific University. The experience of using R is very different from using graphical tools (The tool I used was KNIME). Throughout the course of the entire project, I have learnt a lot of R programming techniques, but not very much on the theoretical part. It is a very practical module which can be immediately deployed into real business use cases in future.

The knowledge gained from this project was enormous. I have had the opportunity to experience the day-to-day job as a data analyst thanks to the practical assignment topic. I have learnt a lot of different charts and plots through self-study for the sake of completing this project distinctly. Moreover, I had also learnt how to implement them and capture the significant findings from these charts through analysis. I would say this module has opened a new domain to me – the data analysis domain. After completing this project, I have become much more confident in data analysis and would like to try more about this domain in my future undertaking. **I have learnt how to come up with excellent data analysis solutions.**

To be honest, there are some limitations to this project as well. Due to the lack of time as well as manpower, I had not considered much on the sustainability and maintainability of the R programs being produced. In the real world, the R program should actually be designed in a way that it can be more flexible to different datasets. For instance, the R program should be able to adapt to the dataset after some slight changes being made such as a new column being added to the dataset. Besides, the R code would be much more maintainable if it is divided into several files which each file would be responsible for only one responsibility so that it will be easier to track if there is any bug in future. This is known as the single-responsibility principle (Oloruntoba, 2021). These are the limitations that can be improved in future data analysis projects.

References

First San Francisco Partners. (2018, September 6). *Defining Data Acquisition and Why it Matters*. Retrieved from FIRST SAN FRANCISCO PARTNERS: https://www.firstsanfranciscopartners.com/blog/defining-data-acquisition-importance/

Frost, J. (n.d.). *Guidelines for Removing and Handling Outliers in Data*. Retrieved from Statistics By Jim: https://statisticsbyjim.com/basics/remove-outliers/

John. (2020, January 19). *How to Remove Outliers in R*. Retrieved from R-Bloggers: https://www.r-bloggers.com/2020/01/how-to-remove-outliers-in-r/

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

Oloruntoba, S. (2021, December 1). *SOLID: The First 5 Principles of Object Oriented Design*. Retrieved from DigitalOcean: https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design

Press Trust of India. (2022, August 16). *Average housing prices rise 5% in April-June across 8 cities: Report*. Retrieved from Business Standard: https://www.business-standard.com/article/economy-policy/average-housing-prices-rise-5-in-april-june-across-8-cities-report-122081600204_1.html

R, S. A. (2020, November 1). *ML Algorithms' sensitivity towards outliers*. Retrieved from Medium: https://arsrinevetha.medium.com/ml-algorithms-sensitivity-towards-outliers-f3862a13c94d

Stevens, E. (2022, November 26). *What is Data Analytics? A Complete Guide for Beginners*. Retrieved from CareerFoundry Blog: https://careerfoundry.com/en/blog/data-analytics/what-is-data-analytics/#what-is-data-analytics

Tiwari, A. (2021, October 22). *From Mumbai To Delhi, Here's A List Of The Top 10 Richest Cities In India 2021*. Retrieved from India Times: https://www.indiatimes.com/trending/social-relevance/top-10-richest-cities-in-india-2021-552132.html

United Nation. (2021, September 9). *Population of Cities in India*. Retrieved from Statistics Times: https://statisticstimes.com/demographics/country/india-cities-population.php