Melbourne Airbnb Dataset exploration

Srikar Manthatti

Table of Contents:

Introduction

About Airbnb:

Airbnb is the online hospitality service which enables the people to lease or rent their houses for short term which includes the apartment renting, private rooms rental, dorms or homestays.

Melbourne is ranked as the 6th in the top ten list globally for airbnb users. In this project I want to explore and visualize the Melbourne Airbnb dataset. An initiative took by insideairbnb provides the Airbnb datasets of many cities.

Contents of Dataset:

There are mainly two datasets listings and reviews. These datasets are in csv format. The listing dataset contains 105 columns in it. The dimensions of the listings and reviews datasets are 22909 * 105 and 526415 * 6 respectively.

Question:

By exploring this dataset I want to answer the following questions.

1. Which locations have the most Airbnb's in Melbourne
2. What kind of Airbnb are present in Melbourne
3. Which websites provide the most Airbnb's data

Description of Datasource:

The datasource is collected from melbourne_airbnb_datasource.

- The listing dataset has 106 different columns and 22909 entries in it. Although there are many columns we need only few columns for our exploration. These required columns are then extracted and cleaned which is explained in the data wrangling section of this document.
- My major analysis and exploration will focus on the listings dataset.
- The reviews dataset has 6 columns and 526414 rows in it. This dataset is not required for my initial questions. But this can be used to extend my project with the following questions
    1. On what days of the week does the reviews will high.
    2. What type of Airbnb are most likely to get good reviews.

```
|=================================================================================| 100%  138 MB
> dim(listings)
[1] 22909   106
> dim(reviews)
[1] 526414      6
>
```

Data Wrangling:

Libraries used:

readr	This library is used to read the data from csv files faster
dplyr	This library is used for manipulating the data
lubridate	This library is used to extract days and months from the given date
reticulate	This library is used to import python packages in the python chunk in RMD
ggplot2	This library is used to plot the maps and graphics on the given data
na.tools	This library is used to find any missing values in a given column or a vector

```
```{r}
library(readr)
library(dplyr)
library(lubridate)
library(reticulate)
library(ggplot2)
library(na.tools)
listings <- read_csv('melbourne-airbnb-open-data/From_insideairbnb/listings.csv')
reviews <- read_csv('melbourne-airbnb-open-data/From_insideairbnb/reviews.csv')

```
```

The above 6 libraries are used to analyze and explore the Airbnb dataset. The raw dataset is then loaded into listings and reviews using the read_csv() function. The column names of both dataset are shown with the help of colnames()

Columns in listings(only 78 columns are included in the screenshot for sample)

```
colnames(listings)
colnames(reviews)
```

```
 [1] "id"                                  "listing_url"
 [3] "scrape_id"                           "last_scraped"
 [5] "name"                                "summary"
 [7] "space"                               "description"
 [9] "experiences_offered"                 "neighborhood_overview"
[11] "notes"                               "transit"
[13] "access"                              "interaction"
[15] "house_rules"                         "thumbnail_url"
[17] "medium_url"                          "picture_url"
[19] "xl_picture_url"                      "host_id"
[21] "host_url"                            "host_name"
[23] "host_since"                          "host_location"
[25] "host_about"                          "host_response_time"
[27] "host_response_rate"                  "host_acceptance_rate"
[29] "host_is_superhost"                   "host_thumbnail_url"
[31] "host_picture_url"                    "host_neighbourhood"
[33] "host_listings_count"                 "host_total_listings_count"
[35] "host_verifications"                  "host_has_profile_pic"
[37] "host_identity_verified"              "street"
[39] "neighbourhood"                       "neighbourhood_cleansed"
[41] "neighbourhood_group_cleansed"        "city"
[43] "state"                               "zipcode"
[45] "market"                              "smart_location"
[47] "country_code"                        "country"
[49] "latitude"                            "longitude"
[51] "is_location_exact"                   "property_type"
[53] "room_type"                           "accommodates"
[55] "bathrooms"                           "bedrooms"
[57] "beds"                                "bed_type"
[59] "amenities"                           "square_feet"
[61] "price"                               "weekly_price"
[63] "monthly_price"                       "security_deposit"
[65] "cleaning_fee"                        "guests_included"
[67] "extra_people"                        "minimum_nights"
[69] "maximum_nights"                      "minimum_minimum_nights"
[71] "maximum_minimum_nights"              "minimum_maximum_nights"
[73] "maximum_maximum_nights"              "minimum_nights_avg_ntm"
[75] "maximum_nights_avg_ntm"              "calendar_updated"
[77] "has_availability"                    "availability_30"
```

Columns in reviews dataset

```
colnames(reviews)
```

```
[1] "listing_id"    "id"            "date"          "reviewer_id"   "reviewer_name"
[6] "comments"      "day"           "month"
```

Deleting the columns which have all null values:

I have identified the columns which have all null values in it and deleted the columns. Before that I have copied my listings dataset into listing_duplicate. All data manipulation will be done on this dataframe.

```
```{r}
listings_duplicate <- listings
#a <- apply(listings_duplicate,2,unique)
a<- sapply(listings_duplicate, function(y) sum(length(which(is.na(y)))))
#a[1:2]
#is.vector(a)
#a
listings_duplicate <- listings[,colSums(is.na(listings)) < nrow(listings)]
length(listings_duplicate)
```
```

```
[1] 100
```

This removed 6 columns which have entire nulls

In our dataset we have 6 columns which have all null values in it. After deleting this the total columns in listing_duplicate are 100.

Removing columns which have url's in it:

In our dataset we have few columns which have url data of the Airbnb houses, host profile, Airbnb pictures etc.,

As we don't have anything to do with the url data we can identify these columns and remove them.

```
URL's doesnt effect our analysis so we can remove those aswell
```{r}
listings_duplicate[,names(listings_duplicate)[grep("url",names(listings_duplicate))]] <- NULL
 length(listings_duplicate)
```
```

```
[1] 94
```

Deleting column which have same value in all the row:

In our dataset we have few columns which have same value in all rows, columns are scrape_id, state, country, country_code. As we are doing analysis on Melbourne city, we don't need columns such as state, country and country_code.

```
unique(listings$scrape_id)
unique(listings$state)
unique(listings$country_code)
unique(listings$country)
listings_duplicate[,c( "scrape_id", "state", "country_code", "country")] <- NULL

```
```
```

Deleting columns which have more than 70% null values:

Though we have deleted columns which have all null values, we still have columns which have more then 70% of the null data in it. As these nulls will affect our analysis we have two options to overcome with these.

i. We can replace all nulls with the median, mean or mode if those columns are necessary. But if our dataset is huge this might affect in the exploration.

ii. We can simply delete these columns if they are not necessary.

In my dataset exploration, these columns will not affect the analysis. So here am choosing the second option to delete these columns.

Deleting the columns which have more then 70% of null values in it

```{r}
total <- nrow(listings_duplicate)
b <-  round(total * 0.7)
b
listings_duplicate <- listings_duplicate[,colSums(is.na(listings_duplicate)) < b]
length(listings_duplicate)
```

```
[1] 16036
[1] 91
```

Columns which are not required:

There are few columns which are not required for exploring, so it's a good option to delete those columns and reduce the space.

c("security_deposit","weekly_price","monthly_price","first_review","jurisdiction_names","zipcode","street","market","cleaning_fee","name","interaction","access","space","notes","summary","description","host_name","host_has_profile_pic","host_verifications","host_neighborhood","require_guest_profile_picture","require_guest_phone_verification","calculated_host_listings_count",
"host_location","transit","neighborhood_overview","house_rules","host_about","license",
"requires_license","host_neighbourhood")

```{r}
listings_duplicate[,c("security_deposit","weekly_price","monthly_price","first_review","jurisdiction_names","zipcode","street","market","cleaning_fee","name","interaction","access","space","notes","summary","description","host_name","host_has_profile_pic","host_verifications","host_neighborhood","require_guest_profile_picture","require_guest_phone_verification","calculated_host_listings_count",
"host_location","transit","neighborhood_overview","house_rules","host_about","license",
"requires_license","host_neighbourhood")] <- NULL
colnames(listings_duplicate)
```

Redundant columns:

There might be columns which will provide same information. As this leads to data redundancy we have to delete these columns.

Check for the columns which have same data

```{r}
listings_duplicate[,names(listings_duplicate[(duplicated(t(listings_duplicate)))])] <- NULL
length(listings_duplicate)
```

```
[1] 62
```

Splitting the amenities column:

Amenities column have many variables in it such as pets, breakfast, internet, TV etc.,
As these all are combined and stored in one column, for data exploration we need to extract these from amenities column. So I have used grepl() to find the most entered amenities, extracted those and created new columns which have Boolean variables in it. For example if any Airbnb allows pets, then in the respective pet column we will have a TRUE values, likewise I have created 5 more columns to store the information.

```{r}
a<- strsplit(listings_duplicate$amenities,",")
listings_duplicate$TV <- ifelse(grepl("TV",a, ignore.case = T)==T,1,0)
listings_duplicate$Internet <- ifelse(grepl("Internet",a, ignore.case= T)==T,1,0)
listings_duplicate$AirCondition <- ifelse(grepl("conditioning",a, ignore.case =T)==T,1,0)
listings_duplicate$Pets <- ifelse(grepl("Pet",a, ignore.case = T)==T,1,0)
listings_duplicate$Pets <- ifelse(grepl("Dog",a, ignore.case = T)==T,1,listings_duplicate$Pets)
listings_duplicate$Pets <- ifelse(grepl("Cat",a, ignore.case = T)==T,1,listings_duplicate$Pets)
listings_duplicate$Kitchen <- ifelse(grepl("Kitchen",a, ignore.case = T)==T,1,0)
listings_duplicate$Breakfast <- ifelse(grepl("breakfast",a, ignore.case = T)==T,1,0)
listings_duplicate[,c("amenities")] <- NULL
```

Converting these column values in to Boolean

```{r}
listings_duplicate["Pets"] <- listings_duplicate[["Pets"]] == 1
listings_duplicate["TV"] <-  listings_duplicate[["TV"]] == 1
listings_duplicate["Internet"] <-  listings_duplicate[["Internet"]] == 1
listings_duplicate["AirCondition"] <-  listings_duplicate[["AirCondition"]] == 1
listings_duplicate["Kitchen"] <-  listings_duplicate[["Kitchen"]] == 1
listings_duplicate["Breakfast"] <-  listings_duplicate[["Breakfast"]] == 1
```

Data Checking:

Converting the price values from non numeric to numeric:

In our dataset the column price is non numeric and we have $ symbol appended to all prices. So we have to remove the symbol and change the datatype for visualization.

Converting the price values from non numeric to numeric

```{r}
typeof(listings$price)
listings_duplicate$price <- sub("\\$","",listings_duplicate$price)
listings_duplicate$price <- as.numeric(listings_duplicate$price)
typeof(listings_duplicate$price)
```

Removing the $ symbol for extra_people price column as well and converting the datatype

```
Removing $ symbol and chnaging the datatype of extra_people column
```{r}
typeof(listings$extra_people)
listings_duplicate$extra_people <- sub("\\$","",listings_duplicate$extra_people)
listings_duplicate$extra_people <- as.numeric(listings_duplicate$extra_people)
typeof(listings_duplicate$extra_people)
```
```

```
[1] "character"
[1] "double"
```

Converting few columns to factor:

Using as.factor(), I am converting few columns as factors. So that these levels can be accessed faster than usual data.

```
```{r}
listings_duplicate$host_response_time <- as.factor(listings_duplicate$host_response_time)
listings_duplicate$host_is_superhost <- as.factor(listings_duplicate$host_is_superhost)
listings_duplicate$host_identity_verified <- as.factor(listings_duplicate$host_identity_verified)
listings_duplicate$neighbourhood_cleansed <- as.factor(listings_duplicate$neighbourhood_cleansed)
listings_duplicate$is_location_exact <- as.factor(listings_duplicate$is_location_exact)
listings_duplicate$property_type <- as.factor(listings_duplicate$property_type)
listings_duplicate$room_type <- as.factor(listings_duplicate$room_type)
listings_duplicate$bed_type <- as.factor(listings_duplicate$bed_type)
listings_duplicate$calendar_updated <- as.factor(listings_duplicate$calendar_updated)
listings_duplicate$instant_bookable <- as.factor(listings_duplicate$instant_bookable)
 listings_duplicate$cancellation_policy<- as.factor(listings_duplicate$cancellation_policy)
```
```

Identifying outliers in price column:

To identify the outliers in the price column, I have plotted the distribution of price. In the below screenshot we can see that there few houses which have price 0. This might be caused due to some error while inserting the house price.

```
ggplot(listings_duplicate) +
  geom_bar(aes(price),fill = '#fd5c63',alpha = 0.85,binwidth = 10) +
  theme_minimal(base_size = 13) + xlab("Price") + ylab("Number") +
  ggtitle("The Distribution of Property Price")
```
```


```

To overcome this, am replacing the columns having 0 price with price mean value.

Finding outliers in price column, we see that few columns have 0, replacing them with price mean

```{r}

change <- which(listings_duplicate$price==0)
change
#summary(listings_duplicate$price)
mean_value <- 144
colnames(listings_duplicate)
for (i in 1:length(change))
  {
 row_number1 <- change[i]
  listings_duplicate[row_number1,29] <-mean_value
}


```

Extracting day and month from Date:

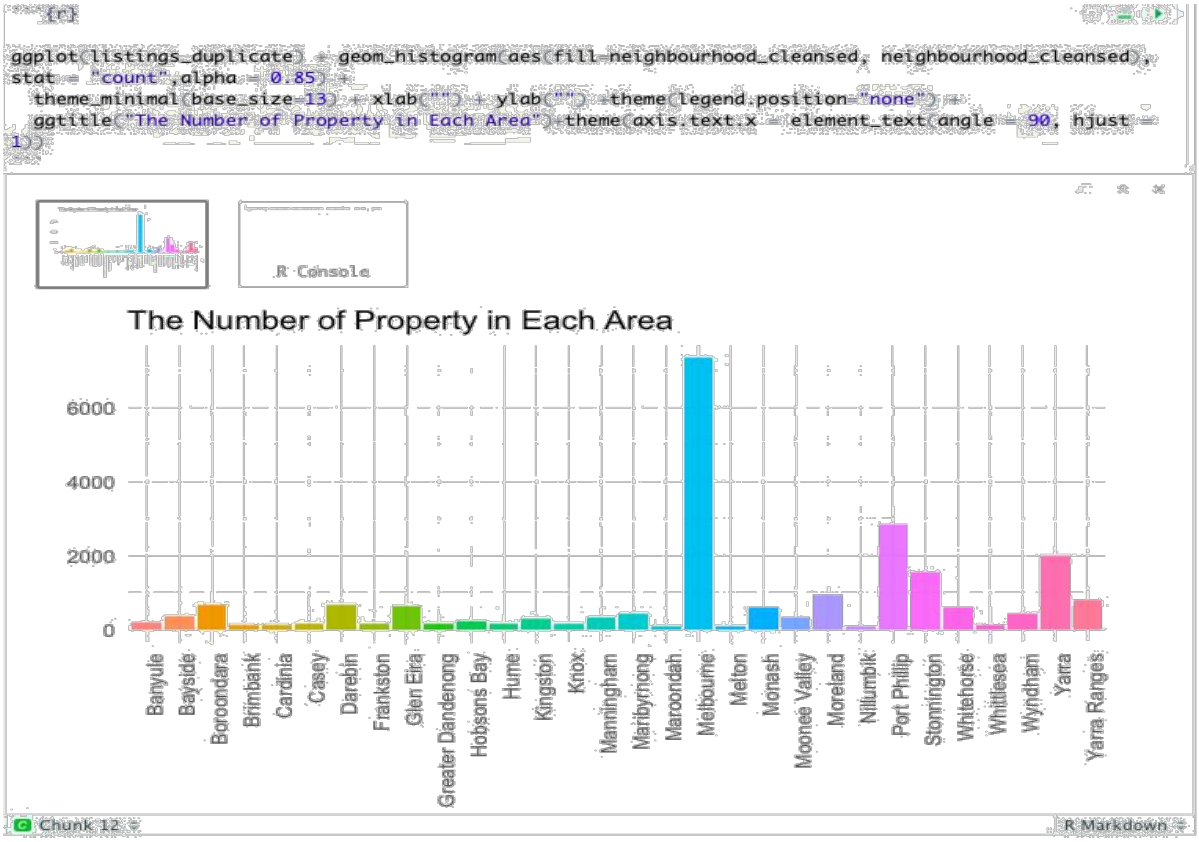
To explore the reviews dataset, we need to extract the day and month from the given date.

```{r}
bdays <- c(reviews$date)
reviews$day <- weekdays(bdays)
reviews$month <- month(reviews$date)
reviews$month <- sub("12","Dec",reviews$month)
reviews$month <- sub("11","Nov",reviews$month)
reviews$month <- sub("10","Oct",reviews$month)
reviews$month <- sub("9","Sep",reviews$month)
reviews$month <- sub("8","Aug",reviews$month)
reviews$month <- sub("7","Jul",reviews$month)
reviews$month <- sub("6","Jun",reviews$month)
reviews$month <- sub("5","May",reviews$month)
reviews$month <- sub("4","Apr",reviews$month)
reviews$month <- sub("3","Mar",reviews$month)
reviews$month <- sub("2","Feb",reviews$month)
reviews$month <- sub("1","Jan",reviews$month)
```

Data Exploration:

Council that has most airbnbs:

Neighbour_cleansed is the column which has the city council in which the Airbnb is located. So if we plot the histogram of these neighbour_cleaned then we can see which council has many Airbnb.

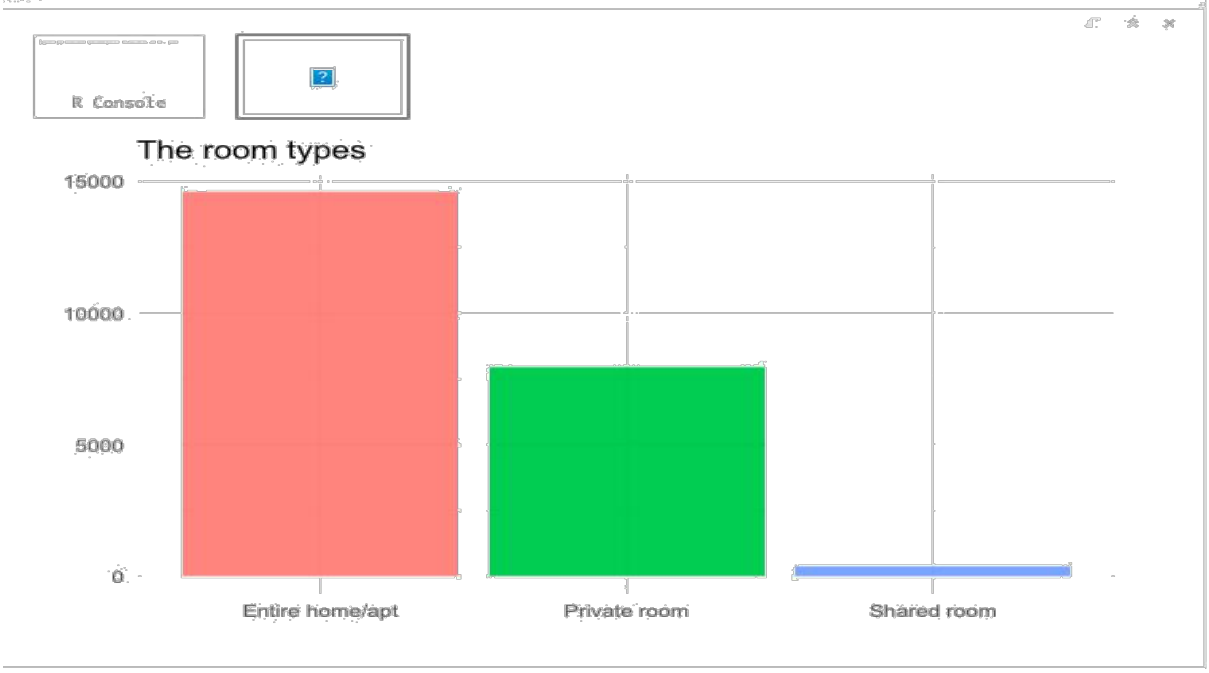


From the above graph we can see that Melbourne city council has many airbnbs in it than any other. This Melbourne city council consists of Central Business District (Melbourne CBD), Carlton, South Yarra, SouthBank etc.,

Number of different room types:

Room type is column which indicates the type of airbnb's, so here I wanted to plot the different types of room and their count to check which are more available.

```r
ggplot(listings_duplicate) + geom_histogram(aes(fill=room_type, room_type), stat = "count",alpha = 0.85) +
  theme_minimal(base_size = 13) + xlab("") + ylab("") + theme(legend.position="none") +
  ggtitle("The room types")#+theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



From the above graph, we can see that different airbnbs are classified into three room types. They are

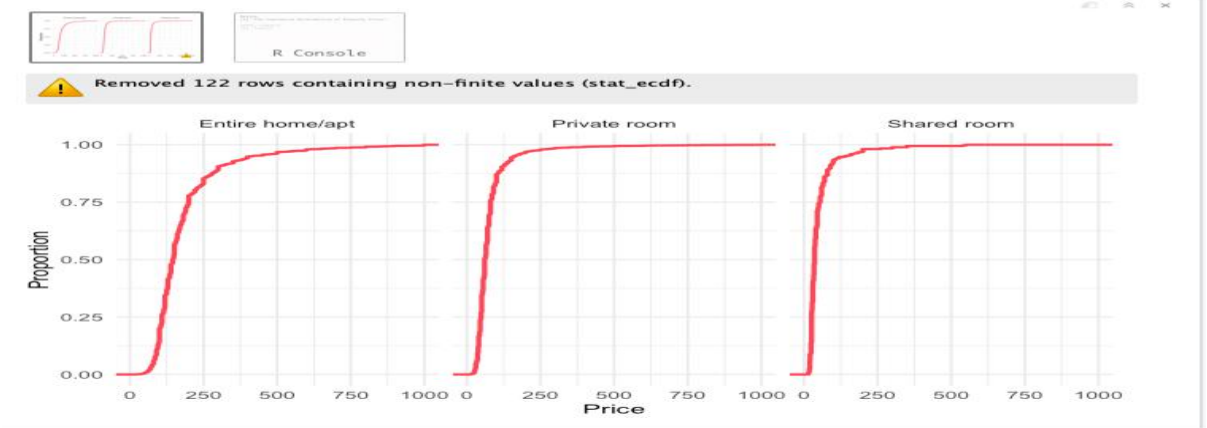
i. Entire Home/apt
ii. Private room
iii. Shared room

Out of all room types, 64% of airbnbs are offering entire home or apartment for share.
34% of airbnbs are offering private rooms as airbnbs. 2% of them are shared room.

Cumulative Distributive function of Price:

By plotting the cumulative distributive function of the price with respect to room type, we can get the 0.5 proportion of the price. i.e., exactly we can figure out the mean price of the house.

```r
ggplot(listings_duplicate, aes(price)) +
  stat_ecdf(geom = "step", color = '#fd5c63', lwd = 1.2) +
  ylab("Proportion") + xlab("Price") + theme_minimal(base_size = 13) + facet_grid(~room_type)
  ggtitle("The Cumulative Distrubition of Property Price")
```



By seeing at the below distribution we can tell that most of the Airbnb's price is in between $20 and $200.

```r
ggplot(na.rm(listings_duplicate)) +
  geom_bar(aes(price),fill = '#fd5c63',alpha = 0.85,binwidth = 10) +
  theme_minimal(base_size = 13) + xlab("Price") + ylab("Number") +
  ggtitle("The Distrubition of Property Price")
```

```r
ggplot(listings_duplicate) +
  geom_bar(aes(price),fill = '#fd5c63',alpha = 0.85,binwidth = 10) +
  theme_minimal(base_size = 13) + xlab("Price") + ylab("Number") +
  ggtitle("The Distrubition of Property Price")
```

R Console

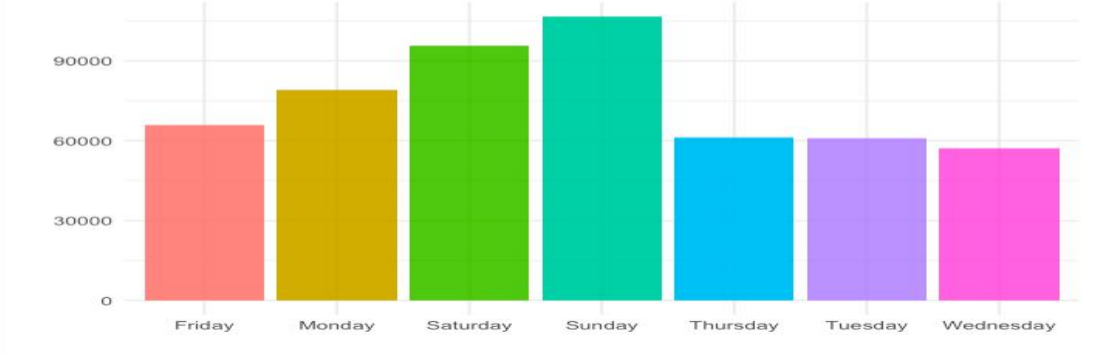
Removed 122 rows containing non-finite values (stat_bin).

Most reviews:

By plotting the last review date on the graph we can see that most reviews are given on Saturday and Sunday. This provides an information that people in their free time will give review to the Airbnb they have visited.

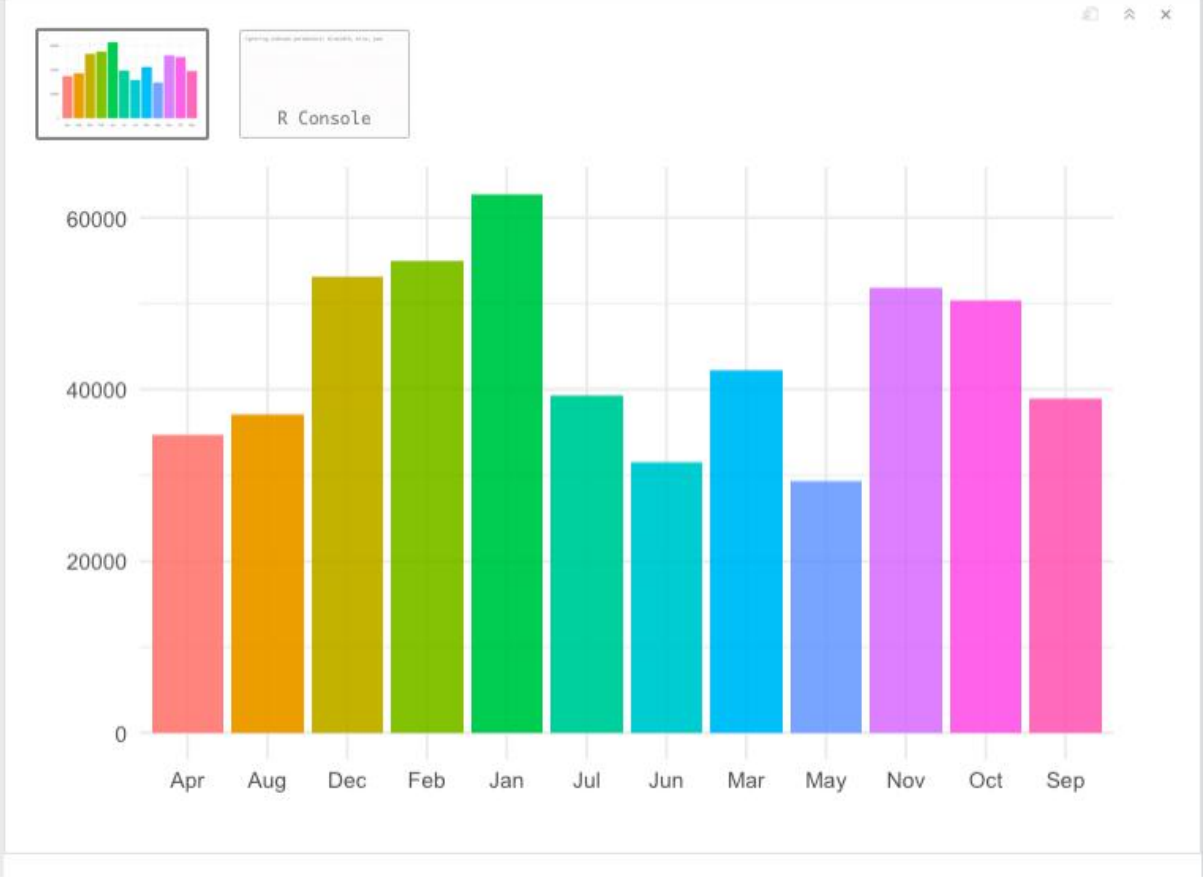
```r
```{r}
ggplot(reviews) + geom_histogram(aes(fill=day, day), stat = "count",alpha = 0.85)
+theme_minimal(base_size=13) + xlab("") + ylab("") +theme(legend.position="none")
```
```

R Console



If we plot the months in which reviews are the most we can see that January has the most reviews. This might be because most of the people will go on a vacation in December month. After returning they have given review to the Airbnb.

```r
ggplot(reviews) + geom_histogram(aes(fill=month, month), stat = "count",alpha = 0.85)
+theme_minimal(base_size=13) + xlab("") + ylab("") +theme(legend.position="none")
```



Conclusion:

After exploring the dataset, I have found the follow information.

1. Out of all city councils Melbourne city council has the most number of airbnb's in Melbourne.
2. There are total three types of airbnbs private room, shared room, entire home or apartment.
3. Out of all airbnbs 64% are entire home or apartment, 34% is private room and the remaining 2% is shared room.
4. The average price of home or apartment, private room and shared room are 189.1, 80, 49.2 respectively.
5. Most of the reviews are given on weekends ie, Saturday and Sunday. People are most likely to give reviews on weekends.
6. The January month has the most number of reviews submitted. This might be because people will be on vacation in December. After returning from vacation they are most likely to give review in January

This exploration has answered two of my initial questions,

i. Which locations have the most Airbnb's in Melbourne

ii. What kind of Airbnb are present in Melbourne

The third question which website provide most Airbnb data, I don't have enough data in the given dataset to answer this question. But in my analysis I have found answers to few other questions as well.

1. On what days of the week does the reviews will high.
2. Which month people are likely to give more review.

Reference:

Dates and times in R https://www.stat.berkeley.edu/~s133/dates.html

Empirical Cumulative distributive functions https://stat.ethz.ch/R-manual/R-patched/library/stats/html/ecdf.html

Inside Airbnb http://insideairbnb.com/get-the-data.html R

reference to Python https://rstudio.github.io/reticulate/