Cya-harvard-airbnb

Sarah Tomori

04/01/2020

1. Overview

1.1 Introduction

This project is part of the Harvard course "Capstone", which is one of 9 required in order to obtain a professional certificate in Data Science.

In this "Choose Your Own"-project, I have chosen to use the dataset from Kaggle.com, consisting of Airbnb data in Singapore. Airbnb has gained popularity all over the world and is among many travellers a preferred form of accomodation over hotels. Thus, the hotel industry faces a lot of competetion from these private house owners looking to rent out their homes to holiday visitors.

1.2 Purpose of the project

The sharing economy is becoming more and more widespread not only through housing options such as Airbnb but also through services such as Uber and general ridesharing services, crowdfunding, knowledge sharing and many other fields. For this reason, it is a very interesting area to look into and discover how such prices are actually determined.

Airbnb has, as mentioned, competed against many hotels due to there competitive prices and general flexibility. However, there are a various number of things could contribute to determining, how much should be charged for a visitor to stay in an Airbnb. Thus, the purpose of the project is to try to predict, which factors (variables) that affect the price of an Airbnb in Singapore and which machine learning algorithms do the best job at doing so.

In order to determine the best model, RMSE and R-squared will be used as to make informed decisions on the best model. The model that produces the lowest value of RMSE will be the final model and rerun on the test set in the end, which corresponds to the data that hasn't yet been touched by our model prior to this. The r-squared will also be used to give an indication of how much variation the model actually explains.

2. Method and analysis

In this section, the Singapore Airbnb dataset is explored with the help of different visualisation tools and data cleansing. This is an important step prior to our analysis, as it helps us understand the structure of the data and minimize the risk of disturbances that could bias results when training the model to be used for predicting Airbnb prices.

2.1 Preparing the data

First, we need to install and load all the packages required.

```r
#Install required packages
#Download the packages
library("readr")
library("caret")
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
```

```
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```r
library("magrittr")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library("wordcloud")
```

```
## Loading required package: RColorBrewer
```

```r
library("SnowballC")
library("corrplot")
```

```
## corrplot 0.84 loaded
```

```r
library("wesanderson")
library("maps")
library("mapproj")
library("ggmap")
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.

## Please cite ggmap if you use it! See citation("ggmap") for details.

##
## Attaching package: 'ggmap'

## The following object is masked from 'package:magrittr':
##
##     inset
```

```r
library("tm")
```

```
## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```r
library("Hmisc")
```

```
## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
```

```
##     cluster
```

```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
library("rpart")
library("rsq")
library("glmnet")
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-2
```

```
library("neuralnet")
```

```
##
## Attaching package: 'neuralnet'
```

```
## The following object is masked from 'package:dplyr':
##
##     compute
```

Read the file with the Singapore Airbnb data.

```
#Read the file with the Singapore Airbnb data.

#Singapore data set loaded from:
#http://data.insideairbnb.com/singapore/sg/singapore/2019-11-26/visualisations/listings.csv

dl <- tempfile()
download.file("http://data.insideairbnb.com/singapore/sg/singapore/2019-11-26/visualisations/listings.c

data <- read_csv(dl)
```

```
## Parsed with column specification:
## cols(
##   id = col_double(),
##   name = col_character(),
##   host_id = col_double(),
##   host_name = col_character(),
##   neighbourhood_group = col_character(),
##   neighbourhood = col_character(),
##   latitude = col_double(),
##   longitude = col_double(),
##   room_type = col_character(),
##   price = col_double(),
##   minimum_nights = col_double(),
##   number_of_reviews = col_double(),
##   last_review = col_date(format = ""),
##   reviews_per_month = col_double(),
```

```
##   calculated_host_listings_count = col_double(),
##   availability_365 = col_double()
## )
```

Then we need to create the test set, which will be 10% of the whole Airbnb Singapore data set. The test set will be kept till the very end and will ONLY be used to run the finally chosen algorithm on.

```r
# Create test set as 10% of the dataset
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
test_index <- createDataPartition(y = data$price, times = 1, p = 0.1, list = FALSE)
airbnb_singapore <- data[-test_index,]
temp <- data[test_index,]
```

You need to make sure that the host id that is present in the test set is also present in training set.

```r
# Make sure that host_id in the test set are present in the airbnb_singapore set
test_set <- temp %>%
semi_join(airbnb_singapore, by = "host_id")
# Add rows removed from test set back into airbnb_singapore set
removed <- anti_join(temp, test_set)
```

```
## Joining, by = c("id", "name", "host_id", "host_name",
## "neighbourhood_group", "neighbourhood", "latitude", "longitude",
## "room_type", "price", "minimum_nights", "number_of_reviews", "last_review",
## "reviews_per_month", "calculated_host_listings_count", "availability_365")
```

```r
airbnb_singapore <- rbind(airbnb_singapore, removed)
rm(temp, data, removed)
```

A validation set is also created and this set is meant to test the intermediary models when training, before deciding on the best and final model. The chosen final model will then be used on the test set.

```r
# Split airbnb_singapore data further in to validation and training sets
# Validation set will be 10% of current airbnb_singapore data
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
# if using R 3.5 or earlier, use `set.seed(1)` instead
test_index2 <- createDataPartition(y = airbnb_singapore$price, times = 1, p = 0.1, list = FALSE)
train_set <- airbnb_singapore[-test_index2,]
temp2 <- airbnb_singapore[test_index2,]
```

As done with the creation of the test set, we need to make sure that host_id in the validation set is also in the training set.

```r
# Make sure host_id in validation is also in training set
validation <- temp2 %>%
  semi_join(train_set, by = "host_id")
# Add rows removed from validation back into training set
removed <- anti_join(temp2, validation)
```

```
## Joining, by = c("id", "name", "host_id", "host_name",
## "neighbourhood_group", "neighbourhood", "latitude", "longitude",
## "room_type", "price", "minimum_nights", "number_of_reviews", "last_review",
```

```
## "reviews_per_month", "calculated_host_listings_count", "availability_365")
```

```
train_set <- rbind(train_set, removed)
rm(temp2, removed)
```

2.2 Data and exploration

2.2.1 Overview

The next step is to explore the data before analysing it. As the data is taken from Kaggle.com and is ready to use, not much data cleaning was necessesary to do beforehand but we need to do quite a bit of visualisation.

2.2.2 Data exploration

We are going to explore what kind of class our dataset falls into. Based in the class() function in RStudio, this is shown below. We can see that the data is a data frame.

```
# Check class of data
class(airbnb_singapore)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

We can see that our data is a data frame. Next, we can use the glimpse() function to get an overview of the different variables.

```
# Get brief overview of the data
glimpse(airbnb_singapore)
```

```
## Observations: 7,277
## Variables: 16
## $ id                             <dbl> 49091, 50646, 56334, 71609, 718...
## $ name                           <chr> "COZICOMFORT LONG TERM STAY ROO...
## $ host_id                        <dbl> 266763, 227796, 266763, 367042,...
## $ host_name                      <chr> "Francesca", "Sujatha", "France...
## $ neighbourhood_group            <chr> "North Region", "Central Region...
## $ neighbourhood                  <chr> "Woodlands", "Bukit Timah", "Wo...
## $ latitude                       <dbl> 1.44255, 1.33235, 1.44246, 1.34...
## $ longitude                      <dbl> 103.7958, 103.7852, 103.7967, 1...
## $ room_type                      <chr> "Private room", "Private room",...
## $ price                          <dbl> 82, 81, 68, 202, 93, 102, 205, ...
## $ minimum_nights                 <dbl> 180, 90, 6, 1, 1, 1, 1, 90, 90,...
## $ number_of_reviews              <dbl> 1, 18, 20, 15, 24, 45, 26, 174,...
## $ last_review                    <date> 2013-10-21, 2014-12-26, 2015-1...
## $ reviews_per_month              <dbl> 0.01, 0.26, 0.19, 0.16, 0.24, 0...
## $ calculated_host_listings_count <dbl> 2, 1, 2, 8, 8, 8, 8, 4, 4, 4, 3...
## $ availability_365               <dbl> 365, 365, 365, 352, 349, 342, 1...
```

From above, it is shown that data (without our test set which is kept for the end) contains 7,277 observations and there are 16 variables in the data set. id is used to identify the Airbnb listing and is unique for each listing. host_id is used to identify the host responsible for the respective listing. Thus, is is possible for a host to have more than one listing, if they are renting out more than one Airbnb.

Next, let's look at a summary of the data as well.

```
# See a summary of the data
summary(airbnb_singapore)
```

```
##        id                 name                 host_id
##  Min.   :   49091   Length:7277        Min.   :    23666
```

```
##  1st Qu.:16209070   Class :character   1st Qu.: 23243573
##  Median :25599788   Mode  :character   Median : 63448912
##  Mean   :24472903                      Mean   : 93818798
##  3rd Qu.:33922412                      3rd Qu.:156409670
##  Max.   :40382063                      Max.   :312037760
##
##   host_name         neighbourhood_group neighbourhood         latitude
##  Length:7277        Length:7277         Length:7277        Min.   :1.244
##  Class :character   Class :character    Class :character   1st Qu.:1.296
##  Mode  :character   Mode  :character    Mode  :character   Median :1.311
##                                                            Mean   :1.314
##                                                            3rd Qu.:1.322
##                                                            Max.   :1.455
##
##    longitude      room_type             price          minimum_nights
##  Min.   :103.6   Length:7277        Min.   :    0.0   Min.   :   1.00
##  1st Qu.:103.8   Class :character   1st Qu.:   66.0   1st Qu.:   1.00
##  Median :103.8   Mode  :character   Median :  120.0   Median :   3.00
##  Mean   :103.8                      Mean   :  169.8   Mean   :  18.49
##  3rd Qu.:103.9                      3rd Qu.:  195.0   3rd Qu.:  14.00
##  Max.   :104.0                      Max.   :10000.0   Max.   :1000.00
##
##  number_of_reviews  last_review         reviews_per_month
##  Min.   :  0.00    Min.   :2013-10-21   Min.   : 0.010
##  1st Qu.:  0.00    1st Qu.:2019-01-09   1st Qu.: 0.170
##  Median :  2.00    Median :2019-09-02   Median : 0.490
##  Mean   : 13.49    Mean   :2019-03-15   Mean   : 1.007
##  3rd Qu.: 11.00    3rd Qu.:2019-11-02   3rd Qu.: 1.290
##  Max.   :345.00    Max.   :2019-11-26   Max.   :13.450
##                    NA's   :2537         NA's   :2537
##  calculated_host_listings_count availability_365
##  Min.   :  1.00                 Min.   :  0.0
##  1st Qu.:  1.00                 1st Qu.: 50.0
##  Median :  9.00                 Median :260.0
##  Mean   : 40.48                 Mean   :206.9
##  3rd Qu.: 47.00                 3rd Qu.:354.0
##  Max.   :306.00                 Max.   :365.0
##
```

Looking at this summary, we see that the minimum minimum price shown is 0 Singaporean dollars per night while the maximum shown is 10000 Singaporean dollars per night. The maximum number of reviews found is 345, while the minimum is 0, due to some Airbnb's never having been reviewed.

```r
# Check whether there are any missing values
anyNA(airbnb_singapore)
```

```
## [1] TRUE
```

There are some missing values in the dataset. We might have to process that later before starting the analysis.

We can also show the number of unique price values can be found in the dataset.

```r
# Show unique values of airbnb prices
unique(airbnb_singapore$price)
```

```
##    [1]    82    81    68   202    93   102   205    50    55    41    44
##   [12]    40    64    45    30    48    60    56    37   272   164   100
```

```
##  [23]    26   173   899   149   206   128  2559   300   150    42    66
##  [34]    35   240    70   146    49   231   319   136    57   130   400
##  [45]    75    67   165   216    89    90   800   550    33   171   160
##  [56]   255   117   217   141    96   360   379    63   250    98   280
##  [67]    87    31   343   201   252   135   463   120   108    46   119
##  [78]  1200   105   115   111   248   276   126    91   138    85   180
##  [89]   220   221   179   237   121    29    78   388    61   950   306
## [100]   267    79   161   218   246   147    38   168   288   270    20
## [111]    25   186  1000   145   409   225   390   132   143    59   228
## [122]   190   229   500   349   109   175  1599   177   112   199   207
## [133]   188   304   198    76    52   123    71    18    53  1800   239
## [144]  2300   151    27   235   490   999   169    72   158   450   195
## [155]   345   278   191    83   261   210   222   700    22   308   408
## [166]   273   167   325   315  1019   310   287   751   441   257   224
## [177]   433   101    19   156   475   307   449  1350   407   289   131
## [188]   258  3501   232   480   139   104   540   785   730   330  7000
## [199]   650   299   399  3769  1295   469   116  1301    86   766   815
## [210]   254   203   385   680   184   212  2942   183   358   244  4000
## [221]   484    34  8900   570   269    23   560   124   162   351   134
## [232]   113   600   303   373   722 10000   106   213   375   214   580
## [243]    14   394  3000   460   318   364   420   386   192   439  5000
## [254]   251   172   142   182    97  2047   285     0   590   525   154
## [265]    15   197   359    94    74   153   333   314   370   233   313
## [276]   524   368   302   688   474   341   740   430   588   489   575
## [287]   328   682   568   259   520   298  1361  1096  1444  2631  4059
## [298]   727   631   340   344   322   127   412   334   243   888  1455
## [309]   263   393  3200   348   736   655  1830   585   454   453   209
## [320]   157   745   598   265   781  2500   332   338   775  3300   444
## [331]   528   725   661   415   295   242   429   194  6399   691   830
## [342]   355   438   721   545   839   535   875   247   378   620   465
## [353]   697  1500   646   491   521   515   510   336   495  1700   405
## [364]   695   565    16  4326   292   530   227   392  1580  1999   796
## [375]   790  1479
```

```r
#Show the number of unique values of prices
n_distinct(unique(airbnb_singapore$price))
```

```
## [1] 376
```

There is a great range of various unique prices, which is due to the continuous nature of this variable. In fact, there are 364 different prices in the dataset. This is just to get an overview, as counting the number of unique price values is not as explanative as doing so with discrete variables.

Next, we can show how many hosts, listings, regions, neighbourhoods and types of rooms the dataset contains in total.

```r
# Show number of hosts, listings, regions, neighbourhoods and room types
airbnb_singapore %>% summarise(n_hosts = n_distinct(host_id), n_listings = n_distinct(id),
n_regions = n_distinct(neighbourhood_group), n_neighbourhoods = n_distinct(neighbourhood),
n_types = n_distinct(room_type))
```

```
## # A tibble: 1 x 5
##   n_hosts n_listings n_regions n_neighbourhoods n_types
##     <int>      <int>     <int>            <int>   <int>
## 1    2626       7277         5               44       4
```

This shows that there are 2626 hosts, 7277 listings, 5 regions, 44 neighbourhoods and 3 types of rooms possible to rent. The fact that the number of hosts is less than half the number of listings, which confirms the earlier statement claiming that some hosts will likely be hosts for more than one rental or listing.

Next, we will look at the minimum nights required in order to stay in an Airbnb in Singapore.

```
# Show average minimum nights required
mean(airbnb_singapore$minimum_nights)
```

```
## [1] 18.4943
```

The average minimum number of nights required to stay at an Airbnb is 18.4943 nights. This indicates that many owners require you to stay for a longer period of time in order to rent out their accommodation. However, it could also be an indication that there are many listings available on the Singaporean Airbnb platform with the purpose of long duration rental.

We can also show what the average price level is per night in Singapore.

```
# Show average price per night
mean(airbnb_singapore$price)
```

```
## [1] 169.8464
```

The average price is 169.8464 Singaporean dollars.

2.3 Visualisation of the data

It is necessary to review how our data looks before we proceed to the analysing stage of the project.

First, a histogram is created to show the distribution of prices among the various listings.

```
# Price distributions in the training set
airbnb_singapore %>% ggplot(aes(price)) + geom_histogram(bins = 10, color = "#000000",
fill = "#FFCCCC") +
ggtitle("Distribution of prices across Airbnbs")
```

From the histogram above, we see that majority of the Airbnb listings are below a 1000 Singaporean dollars and only few are at much higher end of the scale. The further to the right one looks, the fewer Airbnbs are typically to find at that price level. It also shows that there are many various price levels once the Airbnb costs less than 25 Singaporean dollars per night.

Due to the continuous nature of the prices and the large variation in prices, it is a bit hard to interpret the above histogram. Instead, we can also see the top 20 most common prices.

```
#Check the most common prices
most_common_prices <- airbnb_singapore %>% group_by(price) %>%
summarise(count = n()) %>% top_n(20, count) %>% arrange(desc(count))
most_common_prices
```

```
## # A tibble: 20 x 2
##    price count
##    <dbl> <int>
##  1    50   208
##  2    60   207
##  3   100   192
##  4   130   188
##  5   111   156
##  6   150   155
##  7   201   153
##  8    70   152
##  9    81   150
## 10    40   144
## 11   120   140
## 12    55   132
```

```
## 13    90   109
## 14   180   106
## 15    66   104
## 16    45    99
## 17    96    99
## 18   300    97
## 19    35    86
## 20   141    83
```

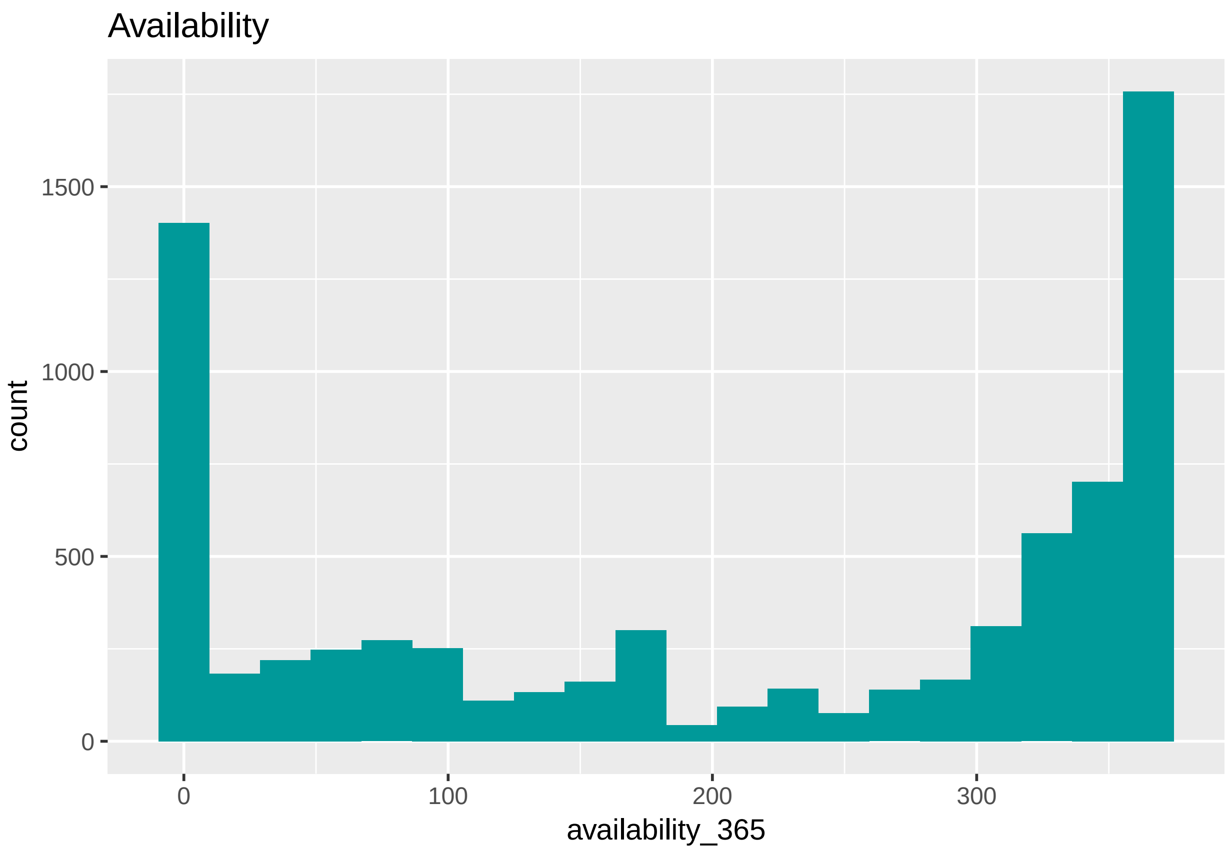
```r
#Check median price
median(airbnb_singapore$price)
```

```
## [1] 120
```

It is found that 50 Singaporean dollars is the most common price with 203 priced at this level. Secondly, 60 Singaporean dollars is commonly used to price Airbnbs as well. The median price for an Airbnb is 120 Singaporean dollars.

A display of the availability distribution can be shown as well.

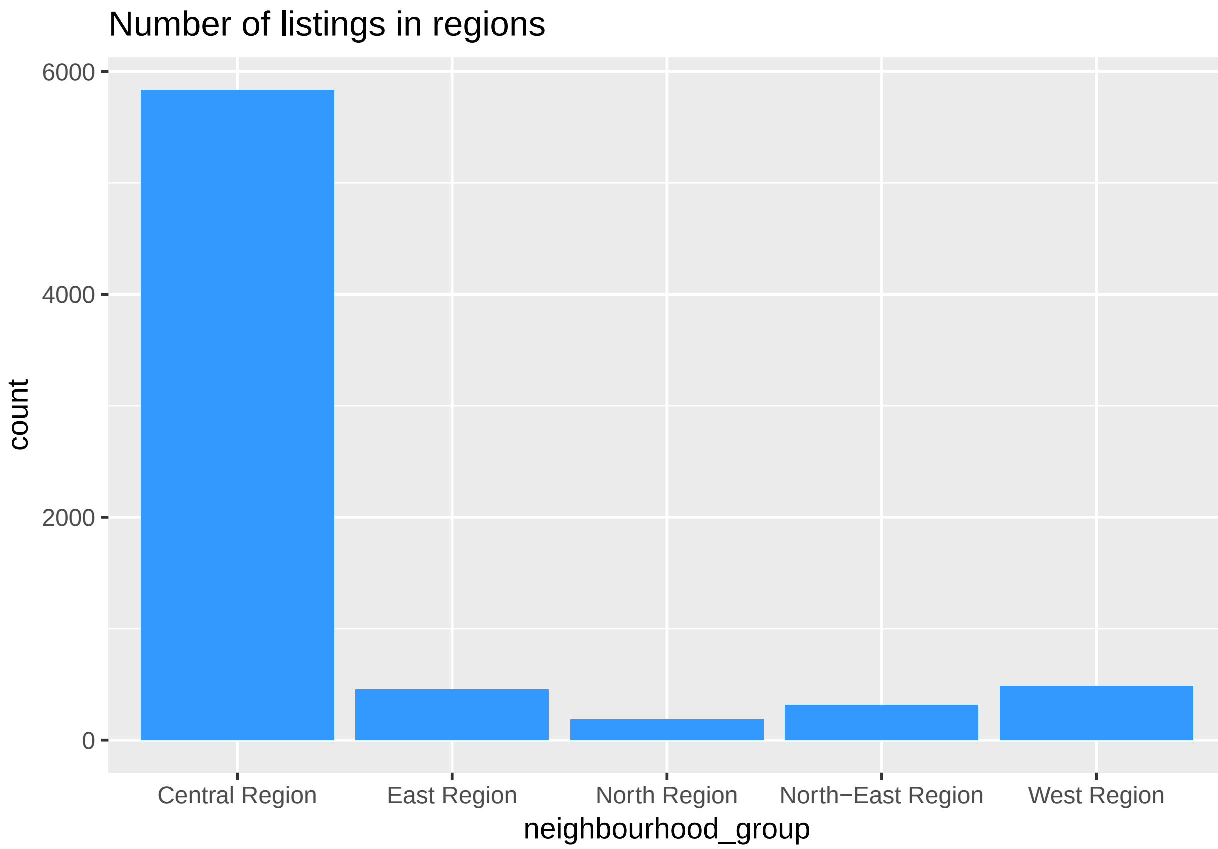
```r
#Check distributions of availability in the training set
airbnb_singapore %>% ggplot(aes(availability_365)) + geom_histogram(bins = 20,
fill = "#009999") +
ggtitle("Availability")
```



The graph shows that while some are only available less than 300 days a year, the far majority of Airbnbs are actually available for rent all 365 days or at least close to that.

The distribution below shows the number of listings in each regions. Each listing has a unique id. Thus, by counting the id variable, we assume this will give us the total number of listings, as an Airbnb should only be listed on the website once.

```r
#Check distributions of listings in the various regions in training set
airbnb_singapore %>% ggplot(aes(neighbourhood_group)) + geom_bar(fill = "#3399FF") +
ggtitle("Number of listings in regions")
```



From the graph above, we can see that the most popular region with the highest number of Airbnbs is the Central Region. This could indicate this number being very popular tourist destinations but also that there are many houses available, where owners like to rent it out to temporary visitors.

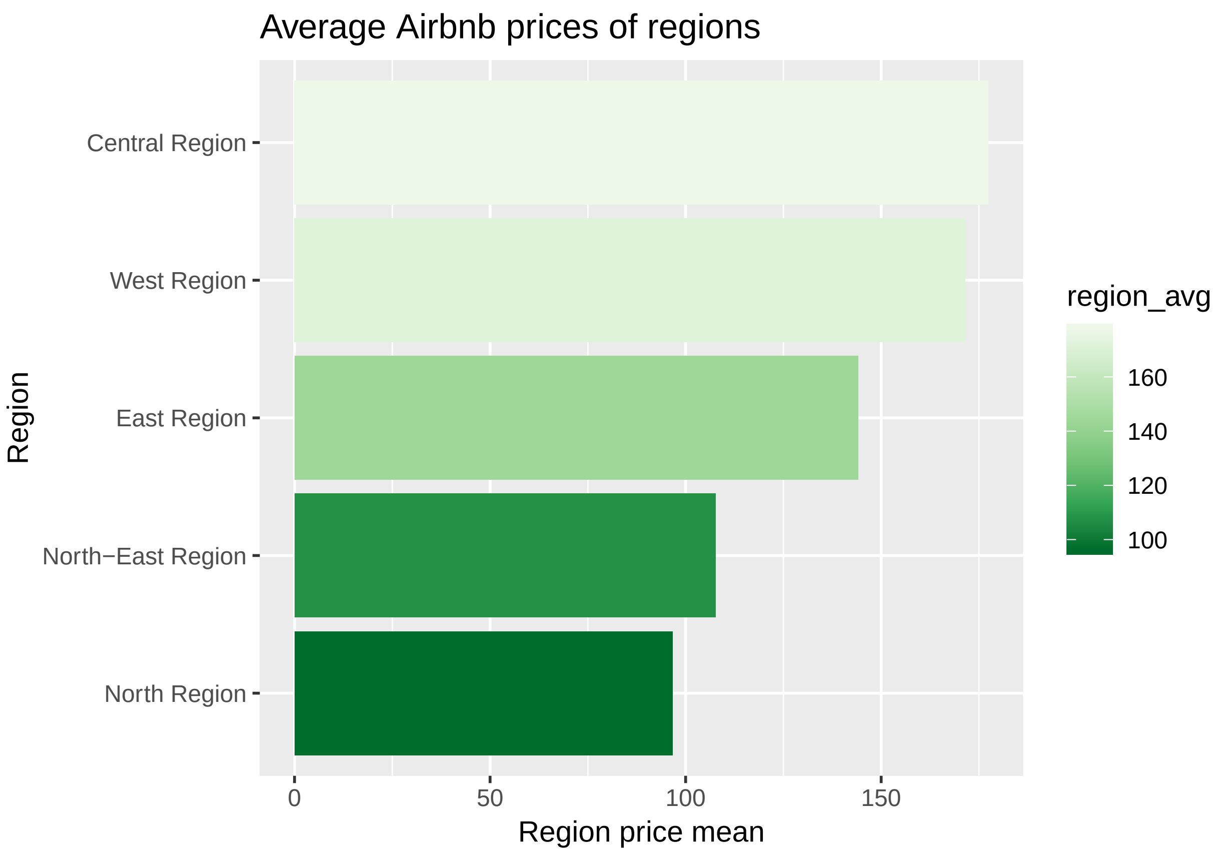
Next, let's see whether there is a difference in the average price levels across regions.

```r
#Check distributions of listings in the various regions in training set

avg_prices <- airbnb_singapore %>% group_by(neighbourhood_group) %>%
summarise(region_avg = mean(price)) %>%
arrange(-region_avg)
avg_prices
```

```
## # A tibble: 5 x 2
##   neighbourhood_group region_avg
##   <chr>                    <dbl>
## 1 Central Region           177.
## 2 West Region              172.
## 3 East Region              144.
## 4 North-East Region        108.
## 5 North Region              96.6
```

```r
#Display them in a histogram
avg_prices %>% ggplot(aes(reorder(neighbourhood_group, region_avg), region_avg,
fill = region_avg)) +
geom_bar(stat = "identity") + coord_flip() +
scale_fill_distiller(palette = "YlOrRed") + labs(y = "Region price mean", x = "Region") +
ggtitle("Average Airbnb prices of regions")
```

```
## Warning in pal_name(palette, type): Unknown palette YlOrRed
```



From this, we see that the Central Region is on average the most expense region to rent an Airbnb in Singapore, followed by West Region, East Region, North-East Region and North Region, which is the cheapest.

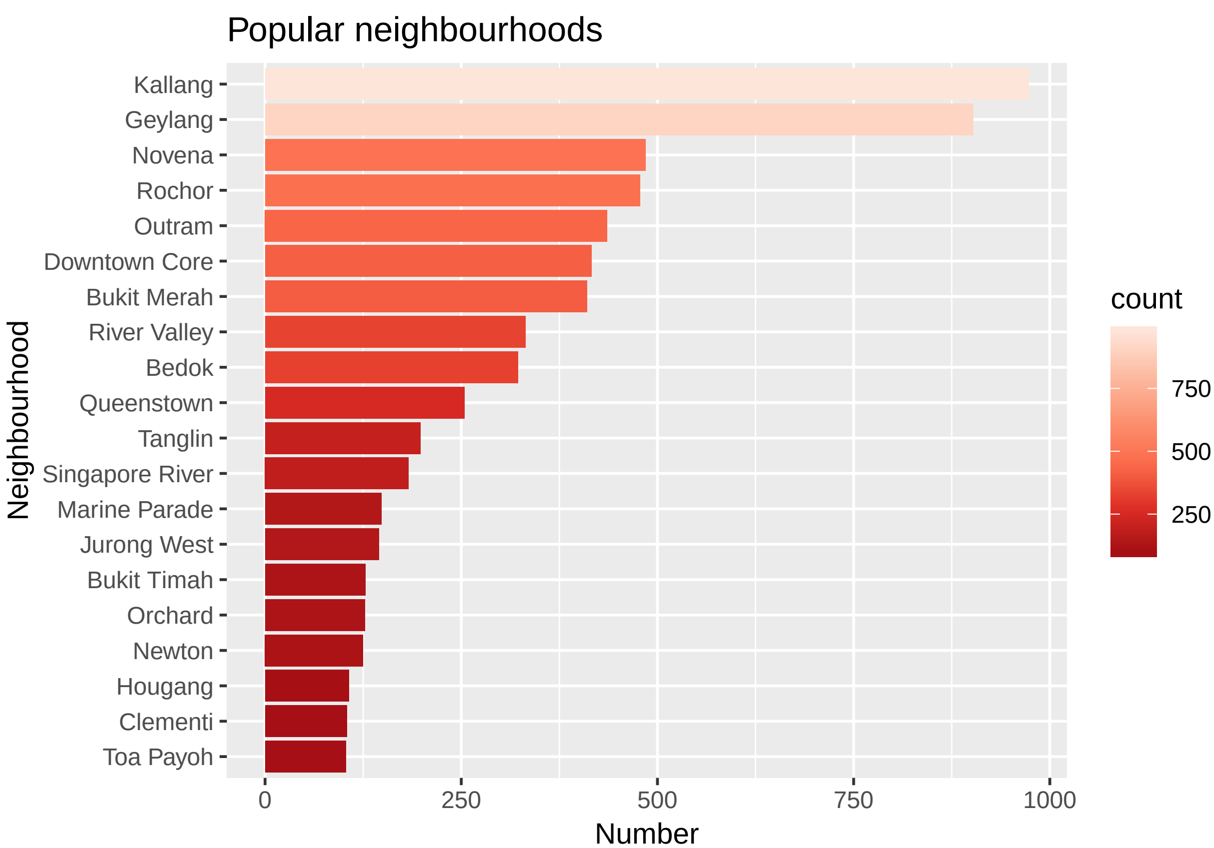
Additionally, we can show the neighbourhoods with the most listings.

```r
#First, save the number of listings in each neighbourhood.

top_neighbourhoods <- airbnb_singapore %>% group_by(neighbourhood) %>%
summarise(count = n()) %>% top_n(20, count) %>% arrange(desc(count))
top_neighbourhoods
```

```
## # A tibble: 20 x 2
##    neighbourhood   count
##    <chr>           <int>
##  1 Kallang           973
##  2 Geylang           902
##  3 Novena            485
##  4 Rochor            478
##  5 Outram            436
```

```
##   6 Downtown Core      416
##   7 Bukit Merah        410
##   8 River Valley       332
##   9 Bedok              322
##  10 Queenstown         254
##  11 Tanglin            198
##  12 Singapore River    183
##  13 Marine Parade      148
##  14 Jurong West        145
##  15 Bukit Timah        128
##  16 Orchard            127
##  17 Newton             125
##  18 Hougang            107
##  19 Clementi           104
##  20 Toa Payoh          103
```

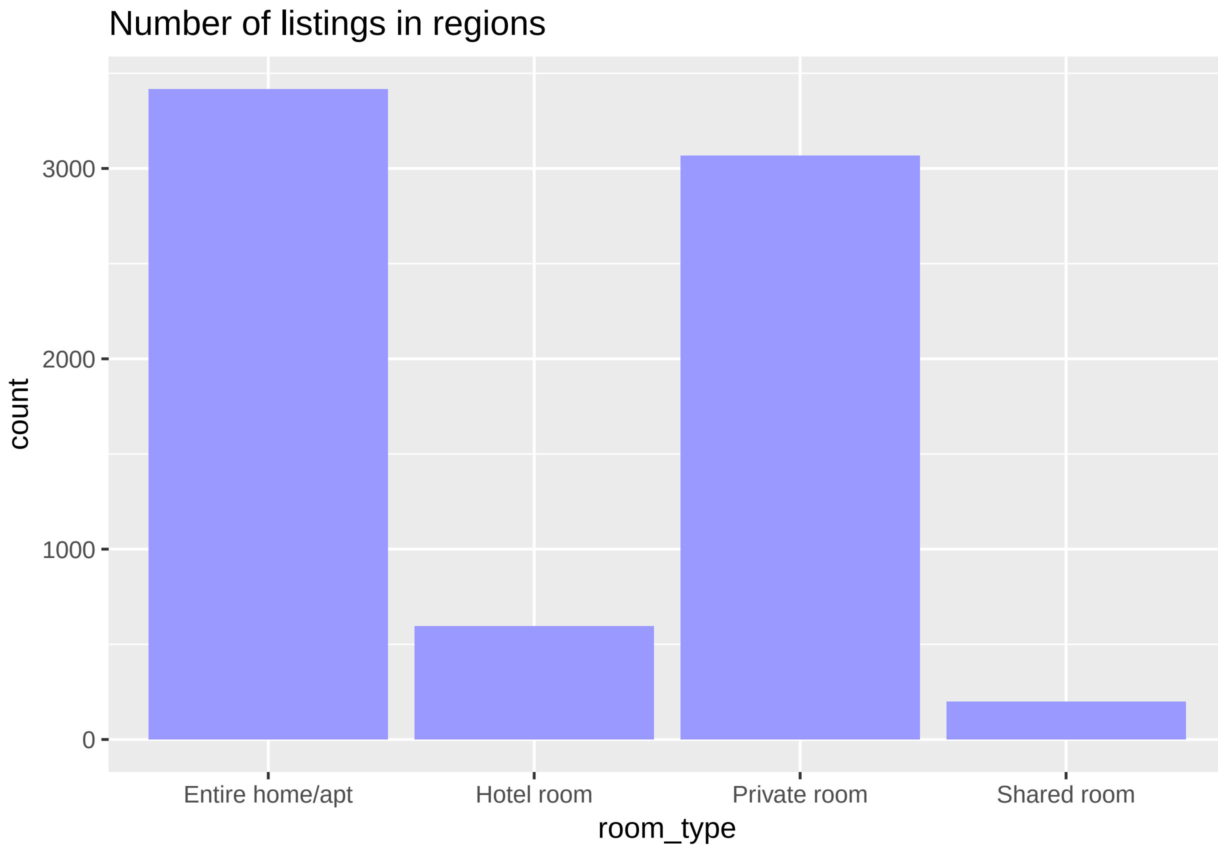
```r
#We can also show them in a bar chart.

top_neighbourhoods %>% ggplot(aes(reorder(neighbourhood, count), count, fill = count)) +
geom_bar(stat = "identity") + coord_flip() +
scale_fill_distiller(palette = "Reds") + labs(y = "Number", x = "Neighbourhood") +
ggtitle("Popular neighbourhoods")
```



From this, we can see that some of the neighbourhoods with most listings are Kallang, Geylang, Novena, Rochor and Bukit Merah.

Let's try to see a distribution of the number of the different room types.

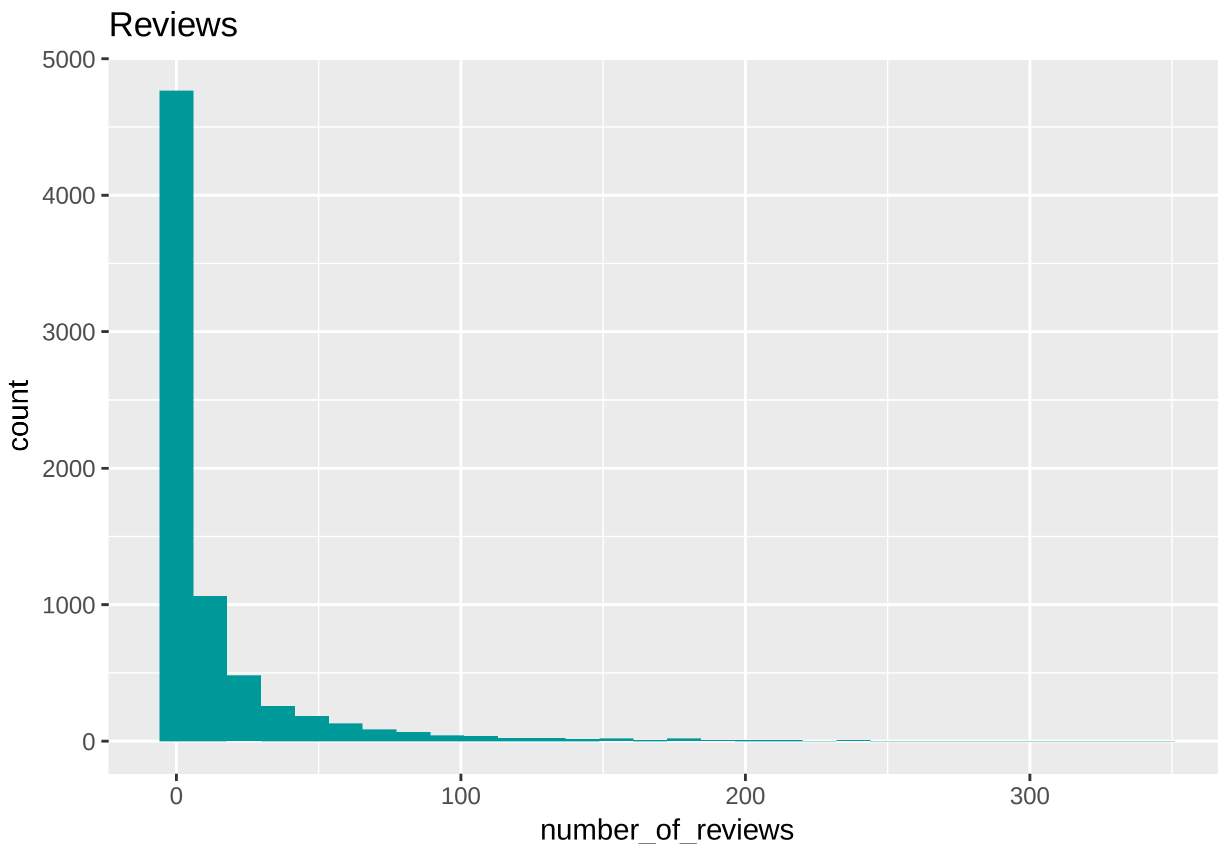
```r
#Check distributions of listings in the various room types in training set
airbnb_singapore %>% ggplot(aes(room_type)) + geom_bar(fill = "#9999FF") +
ggtitle("Number of listings in regions")
```



The most common room types are entire homes or apartments and private rooms. Hotel rooms and shared rooms are much less common.

We can also check the distribution of the number of reviews.

```r
#Check distributions of review count in the training set
airbnb_singapore %>% ggplot(aes(number_of_reviews)) + geom_histogram(bins = 30,
fill = "#009999") +
ggtitle("Reviews")
```



This shows that there are only very few who almost have up to a 5000 reviews, but most Airbnbs have below 1000 reviews. To be more specific, it is generally more common for the Airbnb's to have few or none reviews than to have a lot. This could be due to the large number of listings, leading to a great number of options for visitors. Therefore, some may not even get to be visited or reviewed very often or at all.

It's also interesting to look at which words are most commonly used in the Airbnb titles.

First, we need to clean the text. I first saved the columns with the Airbnb names as a text file.

```
#Now, we import the text file with the Airbnb names. We can read the file from GitHub
filePath <-
"https://raw.githubusercontent.com/sarahtomori/cyo-airbnb-singapore/master/singapore-airbnb-names.txt"
text <- readLines(filePath)
```

```
## Warning in readLines(filePath): incomplete final line found on 'https://
## raw.githubusercontent.com/sarahtomori/cyo-airbnb-singapore/master/
## singapore-airbnb-names.txt'
```

```
#Read the data as a corpus
docs <- Corpus(VectorSource(text))
```

Now, the text can be transformed, which is done using tm_map in order to replace special characters that could be found in the text. These could make it harder to analyse on the words.

```
#Replace special characters with text
#Replace special characters with text
toSpace <- content_transformer(function (x, pattern) gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "[[:punct:]]")
```

```
## Warning in tm_map.SimpleCorpus(docs, toSpace, "[[:punct:]]"):
## transformation drops documents
```

The next step is to further clean the text. This among others includes converting to lower case and removing numbers.

```
# Convert the text to lower case
docs <- tm_map(docs, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(docs, content_transformer(tolower)):
## transformation drops documents
```

```
# Remove numbers
docs <- tm_map(docs, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(docs, removeNumbers): transformation drops
## documents
```

```
# Remove english common stopwords
docs <- tm_map(docs, removeWords, stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, stopwords("english")):
## transformation drops documents
```

```
# Remove your own stop word
# specify your stopwords as a character vector
docs <- tm_map(docs, removeWords, c("blabla1", "blabla2"))
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, c("blabla1", "blabla2")):
## transformation drops documents
```

```
# Remove punctuations
docs <- tm_map(docs, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(docs, removePunctuation): transformation
## drops documents
```

```
# Eliminate extra white spaces
docs <- tm_map(docs, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(docs, stripWhitespace): transformation drops
## documents
```

```
# Text stemming
# docs <- tm_map(docs, stemDocument)
```

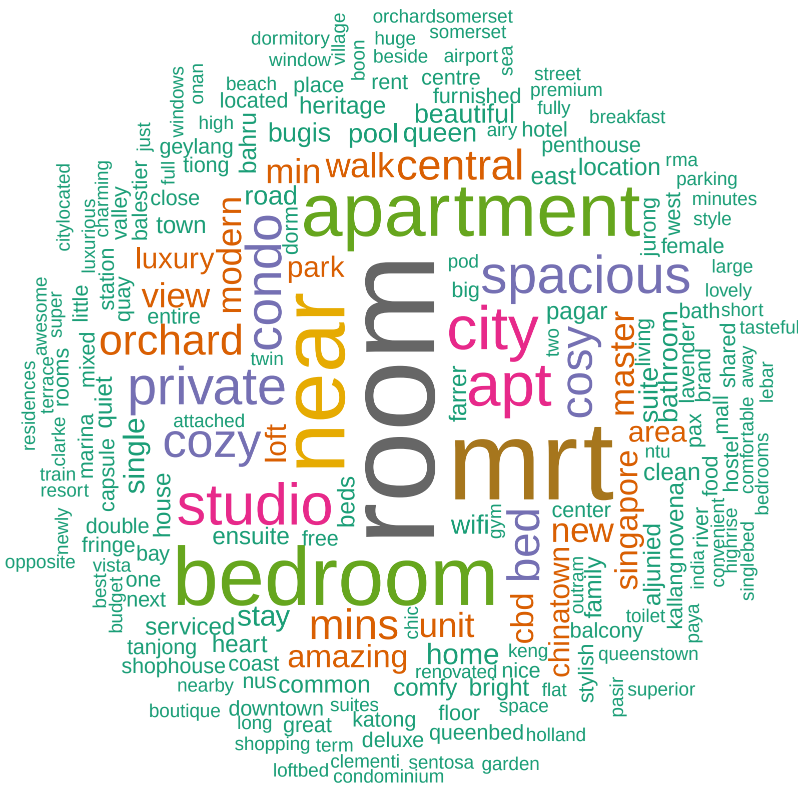
Afterwards, we can create a matrix that contains the frequence of the words.

```
#First, create a matrix that contains the frequency of the words.
dtm <- TermDocumentMatrix(docs)
m <- as.matrix(dtm)
v <- sort(rowSums(m), decreasing=TRUE)
d <- data.frame(word = names(v), freq=v)
head(d, 10)
```

```
##                word freq
## room           room 1709
## mrt             mrt 1477
## near           near 1139
## bedroom     bedroom 1004
## apartment apartment  875
## apt             apt  710
## city           city  688
```

```
## studio       studio  660
## private     private  588
## spacious   spacious  573
```

```r
#Create a wordcloud that visualises the most common words used to describe the Airbnbs.
wordcloud(words = d$word, freq = d$freq, min.freq = 1,
max.words=200, random.order=FALSE, rot.per=0.35,
colors=brewer.pal(8, "Dark2"))
```



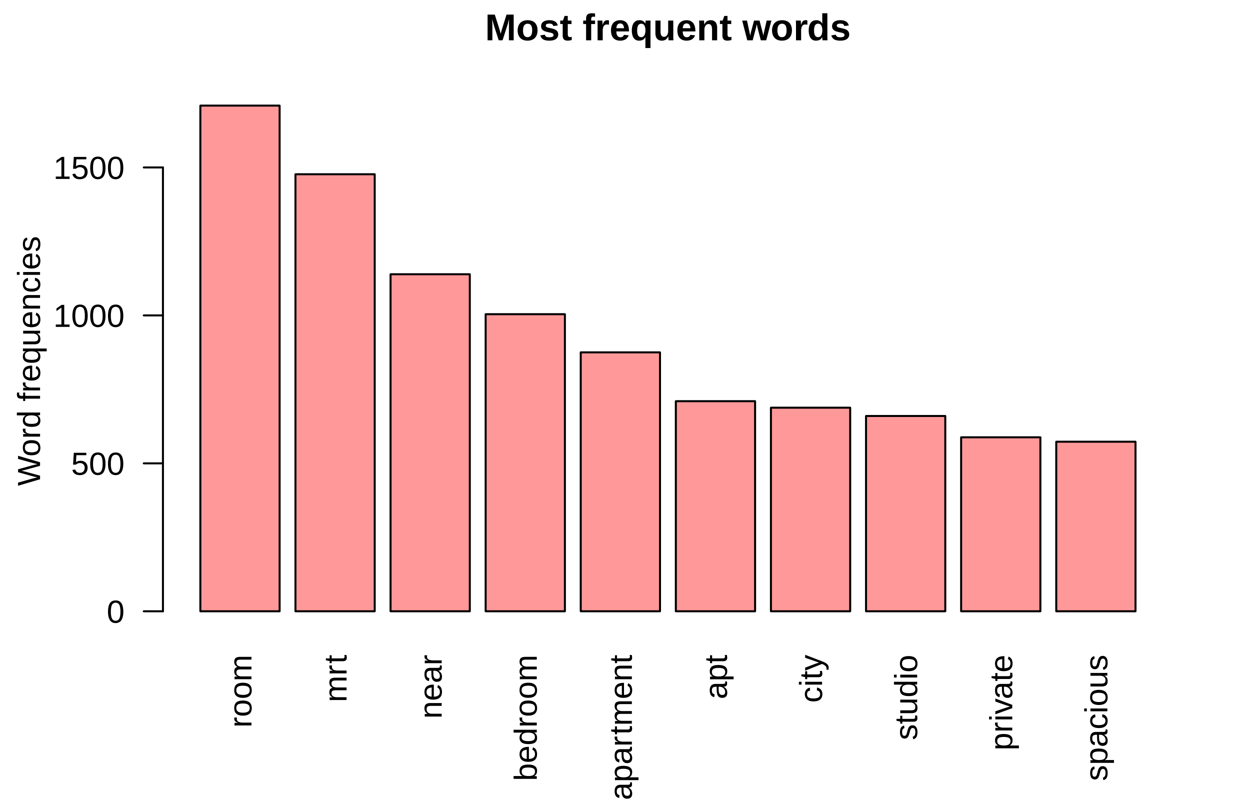
The largest letter words, are the ones most often found in name descriptions of the listings.

We can also see the 10 words that are used most frequently. We can use the following code:

```r
#Show the 10 most frequently used words.
head(d, 10)
```

```
##                  word freq
## room             room 1709
## mrt               mrt 1477
## near             near 1139
## bedroom       bedroom 1004
## apartment apartment  875
## apt               apt  710
## city             city  688
## studio         studio  660
## private       private  588
## spacious     spacious  573
```

```r
#and plot them
barplot(d[1:10,]$freq, las = 2, names.arg = d[1:10,]$word,
col ="#FF9999", main ="Most frequent words",
ylab = "Word frequencies")
```



Another step that is necessary before deciding how to predict Airbnb prices is to check how many of the variables are correlated with each other. If two or more variables are highly correlated, this could dilute the influence that the individual variable has on Airbnb prices, as they could initially explain the same thing. Note that only the numerical values are applied in this plot.

```r
#Check whether the variables are correlated.

cordata <- airbnb_singapore[, c(1,3,7,8,10,11,12,14,15,16)]
head(cordata)
```

```
## # A tibble: 6 x 10
##      id host_id latitude longitude price minimum_nights number_of_revie~
##   <dbl>   <dbl>    <dbl>     <dbl> <dbl>          <dbl>            <dbl>
## 1 49091  266763     1.44      104.    82            180                1
## 2 50646  227796     1.33      104.    81             90               18
## 3 56334  266763     1.44      104.    68              6               20
## 4 71609  367042     1.35      104.   202              1               15
## 5 71896  367042     1.35      104.    93              1               24
## 6 71903  367042     1.35      104.   102              1               45
## # ... with 3 more variables: reviews_per_month <dbl>,
## #   calculated_host_listings_count <dbl>, availability_365 <dbl>
```

```r
cordata.cor = cor(cordata, method = c("spearman"))
cordata.rcorr = rcorr(as.matrix(cordata))
cordata.rcorr
```

```
##                                   id host_id latitude longitude price
## id                              1.00    0.53    -0.10     -0.01  0.04
## host_id                         0.53    1.00    -0.04     -0.02  0.04
```

```
## latitude                          -0.10   -0.04     1.00      -0.04 -0.08
## longitude                         -0.01   -0.02    -0.04       1.00 -0.03
## price                              0.04    0.04    -0.08      -0.03  1.00
## minimum_nights                    -0.07   -0.08     0.08      -0.04  0.00
## number_of_reviews                 -0.33   -0.17    -0.02       0.09 -0.04
## reviews_per_month                  0.19    0.16    -0.02       0.12  0.00
## calculated_host_listings_count     0.21   -0.09    -0.15       0.02  0.01
## availability_365                   0.13    0.04    -0.06      -0.01  0.03
##                                  minimum_nights number_of_reviews
## id                                        -0.07             -0.33
## host_id                                   -0.08             -0.17
## latitude                                   0.08             -0.02
## longitude                                 -0.04              0.09
## price                                      0.00             -0.04
## minimum_nights                             1.00             -0.09
## number_of_reviews                         -0.09              1.00
## reviews_per_month                         -0.12              0.65
## calculated_host_listings_count             0.01             -0.14
## availability_365                           0.15             -0.06
##                                  reviews_per_month
## id                                            0.19
## host_id                                       0.16
## latitude                                     -0.02
## longitude                                     0.12
## price                                         0.00
## minimum_nights                               -0.12
## number_of_reviews                             0.65
## reviews_per_month                             1.00
## calculated_host_listings_count               -0.20
## availability_365                             -0.09
##                                  calculated_host_listings_count
## id                                                         0.21
## host_id                                                   -0.09
## latitude                                                  -0.15
## longitude                                                  0.02
## price                                                      0.01
## minimum_nights                                             0.01
## number_of_reviews                                         -0.14
## reviews_per_month                                         -0.20
## calculated_host_listings_count                             1.00
## availability_365                                           0.24
##                                  availability_365
## id                                           0.13
## host_id                                      0.04
## latitude                                    -0.06
## longitude                                   -0.01
## price                                        0.03
## minimum_nights                               0.15
## number_of_reviews                           -0.06
## reviews_per_month                           -0.09
## calculated_host_listings_count               0.24
## availability_365                             1.00
##
## n
```

```
##                                  id host_id latitude longitude price
## id                             7277    7277     7277      7277  7277
## host_id                        7277    7277     7277      7277  7277
## latitude                       7277    7277     7277      7277  7277
## longitude                      7277    7277     7277      7277  7277
## price                          7277    7277     7277      7277  7277
## minimum_nights                 7277    7277     7277      7277  7277
## number_of_reviews              7277    7277     7277      7277  7277
## reviews_per_month              4740    4740     4740      4740  4740
## calculated_host_listings_count 7277    7277     7277      7277  7277
## availability_365               7277    7277     7277      7277  7277
##                                minimum_nights number_of_reviews
## id                                       7277              7277
## host_id                                  7277              7277
## latitude                                 7277              7277
## longitude                                7277              7277
## price                                    7277              7277
## minimum_nights                           7277              7277
## number_of_reviews                        7277              7277
## reviews_per_month                        4740              4740
## calculated_host_listings_count           7277              7277
## availability_365                         7277              7277
##                                reviews_per_month
## id                                          4740
## host_id                                     4740
## latitude                                    4740
## longitude                                   4740
## price                                       4740
## minimum_nights                              4740
## number_of_reviews                           4740
## reviews_per_month                           4740
## calculated_host_listings_count              4740
## availability_365                            4740
##                                calculated_host_listings_count
## id                                                       7277
## host_id                                                  7277
## latitude                                                 7277
## longitude                                                7277
## price                                                    7277
## minimum_nights                                           7277
## number_of_reviews                                        7277
## reviews_per_month                                        4740
## calculated_host_listings_count                           7277
## availability_365                                         7277
##                                availability_365
## id                                         7277
## host_id                                    7277
## latitude                                   7277
## longitude                                  7277
## price                                      7277
## minimum_nights                             7277
## number_of_reviews                          7277
## reviews_per_month                          4740
## calculated_host_listings_count             7277
```

```
## availability_365                          7277
##
## P
##                                 id     host_id latitude longitude price
## id                                     0.0000  0.0000   0.5762    0.0002
## host_id                         0.0000         0.0017   0.1103    0.0001
## latitude                        0.0000 0.0017           0.0026    0.0000
## longitude                       0.5762 0.1103  0.0026             0.0085
## price                           0.0002 0.0001  0.0000   0.0085
## minimum_nights                  0.0000 0.0000  0.0000   0.0015    0.8703
## number_of_reviews               0.0000 0.0000  0.0534   0.0000    0.0007
## reviews_per_month               0.0000 0.0000  0.2172   0.0000    0.9195
## calculated_host_listings_count 0.0000 0.0000  0.0000   0.0876    0.3252
## availability_365                0.0000 0.0020  0.0000   0.2969    0.0036
##                                 minimum_nights number_of_reviews
## id                              0.0000         0.0000
## host_id                         0.0000         0.0000
## latitude                        0.0000         0.0534
## longitude                       0.0015         0.0000
## price                           0.8703         0.0007
## minimum_nights                                 0.0000
## number_of_reviews               0.0000
## reviews_per_month               0.0000         0.0000
## calculated_host_listings_count 0.5343         0.0000
## availability_365                0.0000         0.0000
##                                 reviews_per_month
## id                              0.0000
## host_id                         0.0000
## latitude                        0.2172
## longitude                       0.0000
## price                           0.9195
## minimum_nights                  0.0000
## number_of_reviews               0.0000
## reviews_per_month
## calculated_host_listings_count 0.0000
## availability_365                0.0000
##                                 calculated_host_listings_count
## id                              0.0000
## host_id                         0.0000
## latitude                        0.0000
## longitude                       0.0876
## price                           0.3252
## minimum_nights                  0.5343
## number_of_reviews               0.0000
## reviews_per_month               0.0000
## calculated_host_listings_count
## availability_365                0.0000
##                                 availability_365
## id                              0.0000
## host_id                         0.0020
## latitude                        0.0000
## longitude                       0.2969
## price                           0.0036
## minimum_nights                  0.0000
```

```
## number_of_reviews              0.0000
## reviews_per_month              0.0000
## calculated_host_listings_count 0.0000
## availability_365
```

```r
#Next, the following code can be used to extract the p-values from the data, using the following code:
cordata.coeff = cordata.rcorr$r
cordata.coeff
```

```
##                                           id     host_id    latitude
## id                              1.000000000  0.53221606 -0.09500779
## host_id                         0.532216056  1.00000000 -0.03686409
## latitude                       -0.095007787 -0.03686409  1.00000000
## longitude                      -0.006553396 -0.01872056 -0.03529924
## price                           0.043884875  0.04467944 -0.08408854
## minimum_nights                 -0.067412889 -0.08081562  0.08188552
## number_of_reviews              -0.328235966 -0.16547769 -0.02264226
## reviews_per_month               0.189178628  0.16462651 -0.01792562
## calculated_host_listings_count  0.212867066 -0.08653592 -0.14799899
## availability_365                0.128491438  0.03621371 -0.06289235
##                                   longitude        price minimum_nights
## id                             -0.006553396  0.043884875   -0.067412889
## host_id                        -0.018720555  0.044679444   -0.080815616
## latitude                       -0.035299242 -0.084088540    0.081885525
## longitude                       1.000000000 -0.030849192   -0.037190544
## price                          -0.030849192  1.000000000   -0.001914777
## minimum_nights                 -0.037190544 -0.001914777    1.000000000
## number_of_reviews               0.094141275 -0.039674939   -0.092994643
## reviews_per_month               0.117198337  0.001469162   -0.123858498
## calculated_host_listings_count  0.020026892  0.011534821    0.007286545
## availability_365               -0.012228880  0.034099811    0.150206431
##                                number_of_reviews reviews_per_month
## id                                   -0.32823597       0.189178628
## host_id                              -0.16547769       0.164626510
## latitude                             -0.02264226      -0.017925624
## longitude                             0.09414127       0.117198337
## price                                -0.03967494       0.001469162
## minimum_nights                       -0.09299464      -0.123858498
## number_of_reviews                     1.00000000       0.653048665
## reviews_per_month                     0.65304867       1.000000000
## calculated_host_listings_count       -0.14182282      -0.201242134
## availability_365                     -0.06108643      -0.092057565
##                                calculated_host_listings_count
## id                                                0.212867066
## host_id                                          -0.086535916
## latitude                                         -0.147998994
## longitude                                         0.020026892
## price                                             0.011534821
## minimum_nights                                    0.007286545
## number_of_reviews                                -0.141822815
## reviews_per_month                                -0.201242134
## calculated_host_listings_count                    1.000000000
## availability_365                                  0.237359399
##                                availability_365
## id                                   0.12849144
```

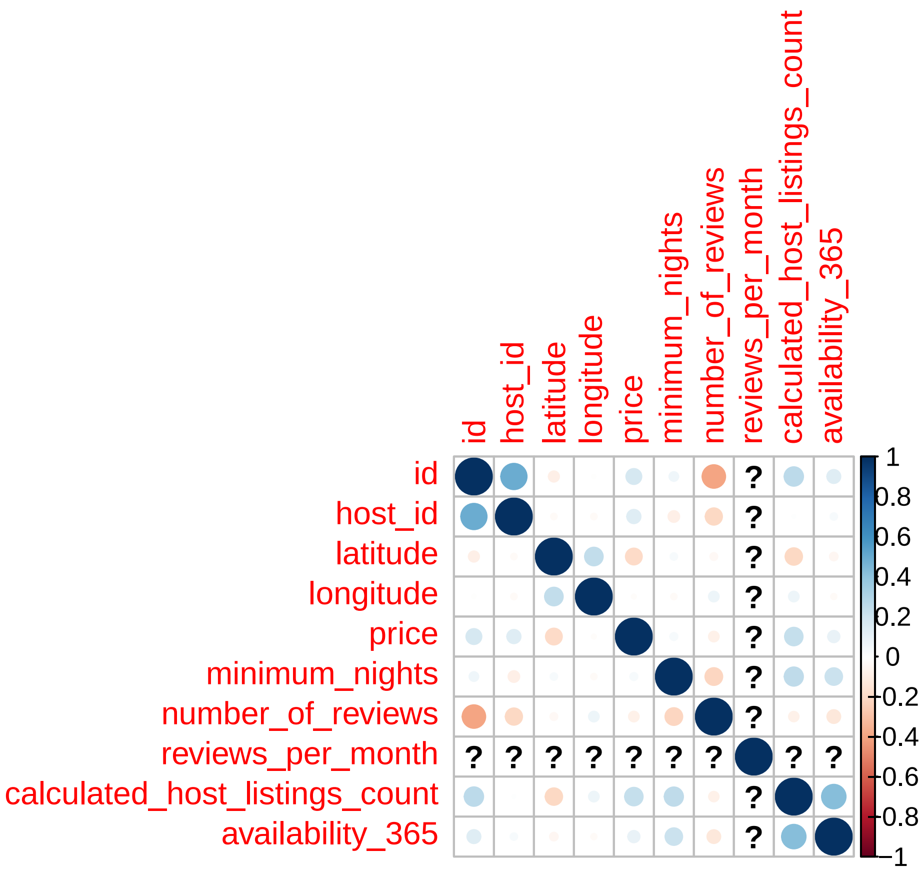
```
## host_id                          0.03621371
## latitude                        -0.06289235
## longitude                       -0.01222888
## price                            0.03409981
## minimum_nights                   0.15020643
## number_of_reviews               -0.06108643
## reviews_per_month               -0.09205756
## calculated_host_listings_count   0.23735940
## availability_365                 1.00000000
```

```
cordata.p = cordata.rcorr$P
cordata.p
```

```
##                                          id      host_id     latitude
## id                                       NA 0.000000e+00 4.440892e-16
## host_id                        0.000000e+00           NA 1.659550e-03
## latitude                       4.440892e-16 1.659550e-03           NA
## longitude                      5.761961e-01 1.103039e-01 2.598449e-03
## price                          1.805556e-04 1.374881e-04 6.736833e-13
## minimum_nights                 8.600824e-09 5.051737e-12 2.637002e-12
## number_of_reviews              0.000000e+00 0.000000e+00 5.343094e-02
## reviews_per_month              0.000000e+00 0.000000e+00 2.172361e-01
## calculated_host_listings_count 0.000000e+00 1.418865e-13 0.000000e+00
## availability_365               0.000000e+00 2.003550e-03 7.897843e-08
##                                   longitude        price minimum_nights
## id                             5.761961e-01 1.805556e-04   8.600824e-09
## host_id                        1.103039e-01 1.374881e-04   5.051737e-12
## latitude                       2.598449e-03 6.736833e-13   2.637002e-12
## longitude                                NA 8.494012e-03   1.508183e-03
## price                          8.494012e-03           NA   8.702722e-01
## minimum_nights                 1.508183e-03 8.702722e-01             NA
## number_of_reviews              8.881784e-16 7.111727e-04   1.776357e-15
## reviews_per_month              4.440892e-16 9.194538e-01   0.000000e+00
## calculated_host_listings_count 8.758467e-02 3.251915e-01   5.342821e-01
## availability_365               2.969258e-01 3.623062e-03   0.000000e+00
##                                number_of_reviews reviews_per_month
## id                                  0.000000e+00      0.000000e+00
## host_id                             0.000000e+00      0.000000e+00
## latitude                            5.343094e-02      2.172361e-01
## longitude                           8.881784e-16      4.440892e-16
## price                               7.111727e-04      9.194538e-01
## minimum_nights                      1.776357e-15      0.000000e+00
## number_of_reviews                             NA      0.000000e+00
## reviews_per_month                   0.000000e+00                NA
## calculated_host_listings_count      0.000000e+00      0.000000e+00
## availability_365                    1.838431e-07      2.156473e-10
##                                calculated_host_listings_count
## id                                               0.000000e+00
## host_id                                          1.418865e-13
## latitude                                         0.000000e+00
## longitude                                        8.758467e-02
## price                                            3.251915e-01
## minimum_nights                                   5.342821e-01
## number_of_reviews                                0.000000e+00
## reviews_per_month                                0.000000e+00
```

```
## calculated_host_listings_count                                    NA
## availability_365                                        0.000000e+00
##                                availability_365
## id                                 0.000000e+00
## host_id                            2.003550e-03
## latitude                           7.897843e-08
## longitude                          2.969258e-01
## price                              3.623062e-03
## minimum_nights                     0.000000e+00
## number_of_reviews                  1.838431e-07
## reviews_per_month                  2.156473e-10
## calculated_host_listings_count     0.000000e+00
## availability_365                             NA
```

```r
#The correlation plot can be visualised using the corrplot() function
corrplot(cordata.cor)
```



The correlation matrix above shows the variables that are positively correlated on a red sclae and the negatively correlated variables as blue scale.

3. Data analysis

Now that we have reviewed and visualised the data, it is time to decide, which algorithm that does the best job at predicting the price of renting an Airbnb in Singapore. It was quite clear that it is a regression problem, as we are interested in predicting Airbnb price, which is a continuous variable.

I first tried to look at the basic mean of prices.

Afterwards, a basic linear regression model (OLS regression) was tested and then the Ridge regression model.

In order to evaluate the best model, RMSE (residual mean squared error) was used to choose the model with the smallest loss, because this is what determines how close it is to predicting actual prices. R-squared is also used to predict, how much of the variation the algorithm manages to predict.

We can define the function for calculating RMSE as written below.

```r
#First, we define the function that calculates RMSE
RMSE <- function(true_price, predicted_price){
sqrt(mean((true_price - predicted_price)^2, na.rm=T))
}
```

3.1 The most basic mean

If we were to take the most basic guess, we would simply guess the price to be the average of all prices of listed Airbnb in Singapore. This value is calculated below.

```r
#Calculate average price of Airbnb prices in Singapore

avg_price <- mean(airbnb_singapore$price)
avg_price
```

```
## [1] 169.8464
```

In this case, the squared loss would be

```r
#Check squared loss
mean((avg_price - validation$price)^2)
```

```
## [1] 32512.62
```

```r
RMSE(validation$price, avg_price)
```

```
## [1] 180.3126
```

This squared loss is huge and obviously a very poor estimate of predicting the price of an Airbnb in Singapore. RMSE of 180.3126 also indicates quite a bit of error from the actual price.

Prices will vary a lot depending on various factors, so the mean is not at all a good estimate. In other words, many prices will vary from the mean and the sum of these differences will eventually add up to a large number as seen above.

3.1 The linear regression model

First, we can try commonly used linear regression model. This is often used to for ML predictions but is a very simple model that works particularly well if there is a high level of linearity.

Due to a lot of challenges with processing missing values, I decided to set the NA's of the 'last_review' variable to the same date of the first review in the data set. For other variables they were set to 0.

```r
#Replace missing values in the last_review variable with the date of the first review
airbnb_singapore$last_review <- ifelse(is.na(airbnb_singapore$last_review),
as.Date(2013-10-21,  origin = "2013-10-21", tz = "GMT"), airbnb_singapore$last_review)

airbnb_singapore[is.na(airbnb_singapore)] <- 0
```

For variables that are characters, they are converted to factors. It is also important to ensure that the training set and the validation set have the same levels to avoid problems when running the models later on.

```r
#Save neighbourhood, neighbourhood_group and room_type variables as factor in training and validation s
validation$neighbourhood <- as.factor(validation$neighbourhood)
airbnb_singapore$neighbourhood <- as.factor(airbnb_singapore$neighbourhood)
validation$room_type <- as.factor(validation$room_type)
airbnb_singapore$room_type <- as.factor(airbnb_singapore$room_type)
validation$neighbourhood_group <- as.factor(validation$neighbourhood_group)
airbnb_singapore$neighbourhood_group <- as.factor(airbnb_singapore$neighbourhood_group)

#Create lm model
fit <- lm(price ~ host_id + longitude + latitude + latitude + number_of_reviews +
calculated_host_listings_count + neighbourhood_group + neighbourhood + room_type +
calculated_host_listings_count + availability_365, data = airbnb_singapore)
#use backward elimination to remove redundant variables
step(fit, direction = "backward", trace = FALSE)
```

```
##
## Call:
## lm(formula = price ~ longitude + number_of_reviews + calculated_host_listings_count +
##     neighbourhood + room_type + availability_365, data = airbnb_singapore)
##
## Coefficients:
##                          (Intercept)
##                            1.830e+05
##                            longitude
##                           -1.761e+03
##                    number_of_reviews
##                           -4.880e-01
##       calculated_host_listings_count
##                           -2.907e-01
##                   neighbourhoodBedok
##                            2.095e+02
##                  neighbourhoodBishan
##                            5.516e+01
##            neighbourhoodBukit Batok
##                           -1.540e+02
##            neighbourhoodBukit Merah
##                           -5.744e+00
##          neighbourhoodBukit Panjang
##                            1.671e+02
##            neighbourhoodBukit Timah
##                           -5.087e+01
## neighbourhoodCentral Water Catchment
##                           -7.591e+00
##          neighbourhoodChoa Chu Kang
##                           -1.561e+02
##               neighbourhoodClementi
##                           -8.350e+01
##          neighbourhoodDowntown Core
##                            9.231e+01
##                neighbourhoodGeylang
##                            9.719e+01
##                neighbourhoodHougang
##                            1.128e+02
##            neighbourhoodJurong East
```

```
##                          -8.282e+01
##          neighbourhoodJurong West
##                          -2.237e+02
##              neighbourhoodKallang
##                           6.400e+01
##         neighbourhoodLim Chu Kang
##                          -2.152e+02
##               neighbourhoodMandai
##                          -6.723e+01
##         neighbourhoodMarina South
##                           3.990e+02
##        neighbourhoodMarine Parade
##                           1.250e+02
##               neighbourhoodMuseum
##                           8.081e+01
##               neighbourhoodNewton
##                           6.126e+01
##               neighbourhoodNovena
##                           3.362e+01
##              neighbourhoodOrchard
##                           1.441e+02
##               neighbourhoodOutram
##                           4.551e+01
##           neighbourhoodPasir Ris
##                           2.044e+02
##          neighbourhoodPaya Lebar
##                           1.042e+02
##              neighbourhoodPunggol
##                           1.014e+02
##          neighbourhoodQueenstown
##                          -7.699e+01
##        neighbourhoodRiver Valley
##                           4.589e+01
##               neighbourhoodRochor
##                           8.654e+01
##            neighbourhoodSembawang
##                          -3.114e+01
##             neighbourhoodSengkang
##                           9.793e+01
##            neighbourhoodSerangoon
##                           7.803e+01
##    neighbourhoodSingapore River
##                           6.276e+01
##   neighbourhoodSouthern Islands
##                           1.612e+03
##        neighbourhoodSungei Kadut
##                          -7.458e+01
##             neighbourhoodTampines
##                           2.034e+02
##              neighbourhoodTanglin
##                           2.724e+01
##            neighbourhoodToa Payoh
##                           7.136e+01
##                 neighbourhoodTuas
```

```
##                              9.462e+03
## neighbourhoodWestern Water Catchment
##                             -3.200e+02
##                 neighbourhoodWoodlands
##                             -1.046e+02
##                    neighbourhoodYishun
##                              3.011e+01
##                    room_typeHotel room
##                             -7.853e+01
##                  room_typePrivate room
##                             -1.303e+02
##                   room_typeShared room
##                             -1.561e+02
##                       availability_365
##                              9.423e-02
```

```r
#add levels of 'neighbourhood' in 'validation' dataset to fit$xlevels[["neighbourhood"]]
fit$xlevels[["neighbourhood"]] <- union(fit$xlevels[["neighbourhood"]],
levels(validation[["neighbourhood"]]))

#add levels of 'room_type' in 'validation' dataset to fit$xlevels[["room_type"]] as well
fit$xlevels[["room_type"]] <- union(fit$xlevels[["room_type"]],
levels(validation[["room_type"]]))

#add levels of 'neighbourhood_group' in 'validation' dataset to fit object as well
fit$xlevels[["neighbourhood_group"]] <- union(fit$xlevels[["neighbourhood_group"]],
levels(validation[["neighbourhood_group"]]))
```

Now, we can define and run the lm model. Backward elimination was used to remove redundant variables.

```r
#Set seed
set.seed(1)

#Create lm model
fit <- lm(price ~ host_id + longitude + latitude + latitude + number_of_reviews +
calculated_host_listings_count + neighbourhood_group + neighbourhood + room_type +
calculated_host_listings_count + availability_365, data = airbnb_singapore)
#use backward elimination to remove redundant variables
step(fit, direction = "backward", trace = FALSE)
```

```
##
## Call:
## lm(formula = price ~ longitude + number_of_reviews + calculated_host_listings_count +
##     neighbourhood + room_type + availability_365, data = airbnb_singapore)
##
## Coefficients:
##                            (Intercept)
##                              1.830e+05
##                              longitude
##                             -1.761e+03
##                      number_of_reviews
##                             -4.880e-01
##         calculated_host_listings_count
##                             -2.907e-01
##                     neighbourhoodBedok
```

```
##                               2.095e+02
##                      neighbourhoodBishan
##                               5.516e+01
##                 neighbourhoodBukit Batok
##                              -1.540e+02
##                 neighbourhoodBukit Merah
##                              -5.744e+00
##               neighbourhoodBukit Panjang
##                               1.671e+02
##                 neighbourhoodBukit Timah
##                              -5.087e+01
## neighbourhoodCentral Water Catchment
##                              -7.591e+00
##               neighbourhoodChoa Chu Kang
##                              -1.561e+02
##                    neighbourhoodClementi
##                              -8.350e+01
##               neighbourhoodDowntown Core
##                               9.231e+01
##                     neighbourhoodGeylang
##                               9.719e+01
##                     neighbourhoodHougang
##                               1.128e+02
##                 neighbourhoodJurong East
##                              -8.282e+01
##                 neighbourhoodJurong West
##                              -2.237e+02
##                     neighbourhoodKallang
##                               6.400e+01
##                neighbourhoodLim Chu Kang
##                              -2.152e+02
##                      neighbourhoodMandai
##                              -6.723e+01
##                neighbourhoodMarina South
##                               3.990e+02
##               neighbourhoodMarine Parade
##                               1.250e+02
##                      neighbourhoodMuseum
##                               8.081e+01
##                      neighbourhoodNewton
##                               6.126e+01
##                      neighbourhoodNovena
##                               3.362e+01
##                     neighbourhoodOrchard
##                               1.441e+02
##                      neighbourhoodOutram
##                               4.551e+01
##                   neighbourhoodPasir Ris
##                               2.044e+02
##                  neighbourhoodPaya Lebar
##                               1.042e+02
##                     neighbourhoodPunggol
##                               1.014e+02
##                  neighbourhoodQueenstown
```

```
##                                -7.699e+01
##             neighbourhoodRiver Valley
##                                 4.589e+01
##                   neighbourhoodRochor
##                                 8.654e+01
##                neighbourhoodSembawang
##                                -3.114e+01
##                 neighbourhoodSengkang
##                                 9.793e+01
##                neighbourhoodSerangoon
##                                 7.803e+01
##          neighbourhoodSingapore River
##                                 6.276e+01
##         neighbourhoodSouthern Islands
##                                 1.612e+03
##             neighbourhoodSungei Kadut
##                                -7.458e+01
##                 neighbourhoodTampines
##                                 2.034e+02
##                  neighbourhoodTanglin
##                                 2.724e+01
##                neighbourhoodToa Payoh
##                                 7.136e+01
##                     neighbourhoodTuas
##                                 9.462e+03
## neighbourhoodWestern Water Catchment
##                                -3.200e+02
##                neighbourhoodWoodlands
##                                -1.046e+02
##                   neighbourhoodYishun
##                                 3.011e+01
##                   room_typeHotel room
##                                -7.853e+01
##                 room_typePrivate room
##                                -1.303e+02
##                  room_typeShared room
##                                -1.561e+02
##                      availability_365
##                                 9.423e-02
```

```r
#Let's look at the coefficients
fit$coefficients
```

```
##                          (Intercept)                              host_id
##                         1.860712e+05                         2.925003e-08
##                            longitude                             latitude
##                        -1.782910e+03                        -5.049169e+02
##                    number_of_reviews       calculated_host_listings_count
##                        -4.734622e-01                        -2.869686e-01
##        neighbourhood_groupEast Region      neighbourhood_groupNorth Region
##                         1.412991e+02                         2.515185e+00
## neighbourhood_groupNorth-East Region       neighbourhood_groupWest Region
##                        -5.045459e+01                        -3.728100e+02
##                   neighbourhoodBedok                  neighbourhoodBishan
##                        -8.931896e+00                        -5.142866e+00
```

```
##              neighbourhoodBukit Batok             neighbourhoodBukit Merah
##                          1.560702e+02                        -1.052942e+02
##            neighbourhoodBukit Panjang             neighbourhoodBukit Timah
##                          4.839113e+02                        -1.291265e+02
## neighbourhoodCentral Water Catchment           neighbourhoodChoa Chu Kang
##                         -7.497771e+01                         1.700008e+02
##                 neighbourhoodClementi         neighbourhoodDowntown Core
##                          2.029164e+02                        -6.731607e+00
##                  neighbourhoodGeylang                 neighbourhoodHougang
##                          1.558572e+01                         1.071734e+02
##              neighbourhoodJurong East             neighbourhoodJurong West
##                          2.160720e+02                         7.891766e+01
##                  neighbourhoodKallang           neighbourhoodLim Chu Kang
##                         -1.789808e+01                        -2.524438e+02
##                   neighbourhoodMandai           neighbourhoodMarina South
##                         -9.453674e+01                         3.001878e+02
##            neighbourhoodMarine Parade                  neighbourhoodMuseum
##                          4.027929e+01                        -9.590687e+00
##                   neighbourhoodNewton                  neighbourhoodNovena
##                         -2.432477e+01                        -4.354177e+01
##                  neighbourhoodOrchard                  neighbourhoodOutram
##                          5.607210e+01                        -5.204784e+01
##               neighbourhoodPasir Ris              neighbourhoodPaya Lebar
##                          1.157798e+01                        -1.017150e+02
##                  neighbourhoodPunggol              neighbourhoodQueenstown
##                          1.159130e+02                        -1.706798e+02
##             neighbourhoodRiver Valley                  neighbourhoodRochor
##                         -4.449026e+01                        -1.221950e+00
##                neighbourhoodSembawang                neighbourhoodSengkang
##                         -4.946134e+01                         1.049026e+02
##                neighbourhoodSerangoon       neighbourhoodSingapore River
##                          7.058604e+01                        -3.195218e+01
##         neighbourhoodSouthern Islands           neighbourhoodSungei Kadut
##                          1.497832e+03                        -1.063664e+02
##                 neighbourhoodTampines                 neighbourhoodTanglin
##                                    NA                        -6.082011e+01
##                neighbourhoodToa Payoh                    neighbourhoodTuas
##                                    NA                         9.750783e+03
## neighbourhoodWestern Water Catchment               neighbourhoodWoodlands
##                                    NA                        -1.281895e+02
##                   neighbourhoodYishun                  room_typeHotel room
##                                    NA                        -7.857810e+01
##                 room_typePrivate room                 room_typeShared room
##                         -1.297815e+02                        -1.552914e+02
##                      availability_365
##                          9.330677e-02
```

```r
#Use the predict function
y_hat <- predict(fit, validation)
```

```
## Warning in predict.lm(fit, validation): prediction from a rank-deficient
## fit may be misleading
```

Next, we can see whether this model does any better than the basic mean by looking at our MSE, RMSE and r-squared.

```r
#Check squared loss
mean((y_hat - validation$price)^2)
```

```
## [1] 25809.42
```

```r
RMSE(validation$price, y_hat)
```

```
## [1] 160.6531
```

```r
#and the r-squared
rss_lm <- sum((validation$price - y_hat) ^ 2)  ## residual sum of squares
tss_lm <- sum((validation$price - mean(y_hat)) ^ 2)  ## total sum of squares
rsq_lm <- 1 - rss_lm/tss_lm
rsq_lm
```

```
## [1] 0.2061545
```

RMSE is 160.6531, which is just a bit lower than what the earlier calculated mean. R-squared is 0.2061545, so the model only explains about 21% of the variation from predicted prices.

3.1 The ridge regression model

Next, the ridge regression model can be used to see if it gives us a better result. The ridge regression is often used in order to deal with the problem of multicollinearity. It uses L2 regularization to penalize the square of magnitude for the coifficients in the regression in order to minimize them.

The formula looks as written below: LS Obj + L (sum of the square of coefficients)

where L is lambda. If L = 0, then our output will be similar to that of a basic linear regression.

If L is large, then coefficients will move towards zero. The glmnet package in R is used.

Note: I decided the last_review variable as it contains similar information as number of reviews, and it caused a lot of issues with NAs.

First, we need to define the independent variables.

```r
#Define the independent variables
x_var <- model.matrix(airbnb_singapore$price ~ longitude + number_of_reviews +
neighbourhood_group + calculated_host_listings_count + room_type + availability_365,
data = airbnb_singapore)

#Define the dependent variable
y_var <- airbnb_singapore$price

# Setting the range of lambda values
lambda_seq <- 10^seq(2, -2, by = -.1)

train_data <- data.frame(x_var, y_var)

model_glmnet <- train(y_var ~ ., data = train_data,
method = "glmnet",
metric = "RMSE",
na.action = na.replace,
lambda  = lambda_seq
)
```

We can check to see what the model looks like.

```r
#Check the model
summary(model_glmnet)
```

```
##             Length Class      Mode     
## a0           41    -none-     numeric  
## beta        451    dgCMatrix  S4       
## df           41    -none-     numeric  
## dim           2    -none-     numeric  
## lambda       41    -none-     numeric  
## dev.ratio    41    -none-     numeric  
## nulldev       1    -none-     numeric  
## npasses       1    -none-     numeric  
## jerr          1    -none-     numeric  
## offset        1    -none-     logical  
## call          6    -none-     call     
## nobs          1    -none-     numeric  
## lambdaOpt     1    -none-     numeric  
## xNames       11    -none-     character
## problemType   1    -none-     character
## tuneValue     2    data.frame list     
## obsLevels     1    -none-     logical  
## param         1    -none-     list
```

```r
model_glmnet
```

```
## glmnet 
## 
## 7277 samples
##   12 predictor
## 
## No pre-processing
## Resampling: Bootstrapped (25 reps) 
## Summary of sample sizes: 7277, 7277, 7277, 7277, 7277, 7277, ... 
## Resampling results across tuning parameters:
## 
##   alpha  lambda      RMSE      Rsquared    MAE     
##   0.10    0.1077453  326.1698  0.04265467  96.93823
##   0.10    1.0774527  326.1727  0.04259760  96.86982
##   0.10   10.7745273  326.2630  0.04200459  96.58345
##   0.55    0.1077453  326.1715  0.04263504  96.91247
##   0.55    1.0774527  326.1941  0.04241098  96.69880
##   0.55   10.7745273  326.8943  0.03933789  97.17688
##   1.00    0.1077453  326.1726  0.04261859  96.88968
##   1.00    1.0774527  326.2221  0.04222133  96.57073
##   1.00   10.7745273  327.8995  0.03383040  98.79496
## 
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.1 and lambda
##  = 0.1077453.
```

We see the different variables for lambda.

```r
#Set seed
set.seed(1)

#Choose the optimal lambda value
```

```r
# Using cross validation glmnet
ridge_cv <- cv.glmnet(x_var, y_var, lambda = lambda_seq)
# Best lambda value
best_lambda <- ridge_cv$lambda.min
best_lambda
```

```
## [1] 0.01
```

Cross validation was necessary in order to punish the loss function of high coefficient values in the model. By using cross validation and finding the best_lambda, we see that the lambda providing the best value of RMSE was 0.01.

Next, we continue to build the final model.

```r
#Find the best ridge model
best_ridge <- glmnet(x_var, y_var, alpha = 0, lambda =  best_lambda)

#Get the coefficients
coef(best_ridge)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                                                 s0
## (Intercept)                          65577.3349934
## (Intercept)                              .
## longitude                             -629.1479288
## number_of_reviews                       -0.4780145
## neighbourhood_groupEast Region          57.7356457
## neighbourhood_groupNorth Region        -60.8582529
## neighbourhood_groupNorth-East Region    -8.1879992
## neighbourhood_groupWest Region         -40.9632587
## calculated_host_listings_count          -0.3172961
## room_typeHotel room                    -86.1243746
## room_typePrivate room                 -140.5907611
## room_typeShared room                  -170.5964290
## availability_365                         0.0954530
```

After fitting the best lambda on the training set, the predict function can be used to fit the ridge regression on the validation set.

```r
#Define y and x variables from the validation set
y_val <- validation$price
x_val <- model.matrix(validation$price ~ longitude + number_of_reviews +
calculated_host_listings_count + neighbourhood_group + room_type
+ availability_365, data = validation)

#Save as a data frame
val_data <- data.frame(x_val, y_val)

#Save the data in a data frame
val_data <- data.frame(x_val, y_val)

#Use predict function and fit to validation data
glmnet_pred_val <- predict(best_ridge, s = best_lambda, newx = x_val)
```

Let's try to see the values of RMSE and r-squared.

```r
#Calculate the MSE
mean((glmnet_pred_val - y_val)^2)
```

```
## [1] 26946154
```

```r
#and the RMSE
RMSE(validation$price, glmnet_pred_val)
```

```
## [1] 5190.969
```

```r
#and the r-squared
rss <- sum((glmnet_pred_val - y_val) ^ 2)  ## residual sum of squares
tss <- sum((y_val - mean(glmnet_pred_val)) ^ 2)  ## total sum of squares
rsq <- 1 - rss/tss
rsq
```

```
## [1] -1.777756
```

It appears that the values of r-square and RMSE are doing worse when using the ridge regression. This indicates that tuning the parameters with this method does not help at predicting the prices better than the linear regression model. The basic linear regression model has a RMSE that is indeed lower than that of the ridge regression model.

Since we know that the linear basic regression model appears to be better than the ridge regression at predicting price, we can try to run it on the test set (our saved piece for the final model, which hasn't been touched throughout this report).

```r
#Use lm model created earlier and fit to validation data
y_test <- predict(fit, test_set)
```

```
## Warning in predict.lm(fit, test_set): prediction from a rank-deficient fit
## may be misleading
```

Next, we can determine the values RMSE and rsq.

```r
#Calculate the MSE
mean((y_hat - test_set$price)^2)
```

```
## Warning in y_hat - test_set$price: longer object length is not a multiple
## of shorter object length
```

```
## [1] 19355.44
```

```r
#and the RMSE
RMSE(test_set$price, y_test)
```

```
## [1] 106.4205
```

```r
#and the r-squared
rss <- sum((y_test - test_set$price) ^ 2)  ## residual sum of squares
tss <- sum((test_set$price - mean(y_test)) ^ 2)  ## total sum of squares
rsq <- 1 - rss/tss
rsq
```

```
## [1] 0.2742621
```

After running the lm on the test set, we get an R-square of 0.2743, indicating that the linear regression model managed to explain nearly 27.5% of variation in prices of Airbnb's in Singapore. This can be significantly better but it gave a better result than the former ridge regression model, indicating that tuning the coefficients did not help improve our outcome in this case.

The factors affecting the Airbnb prices appeared to be longitude, number of reviews, calculated host listings, neighbourhood group, room type and availability_365.

4. Conclusion

From the analysis it can be concluded that ridge regression in this case did not do a better job at predicting the price of Airbnb's in Singapore. Linear regression overall had a better performance.

It is important to note that there are many other ways that could improve the model even further. Due to time constraints, it was out of scope for the project at this time, but for future reference, other, more advanced methods could have been useful trying. This includes ensembling models.