Airbnb France : R Markdown

AYUSH RAGHUVANSHI

2024-07-09

GITHUB REPO LINK OF THIS PROJECT: https://github.com/ayushraghuvanshi2802/R-Programming-Summer-Training

Introduction

This document describes the preparation steps needed to construct the Rshiny app Airbnb Database Rstudio. Credits data source: Insideairbnb.com

```
# Import relevant packages
library(dplyr) library(lubridate)
library(stringr) library(ggplot2)
library(tm) library(wordcloud)

setwd("~/Documents/BoardInfinity/Project/Airbnb_Database_Rstudio")


#Disable scientific notation options(scipen=999)
```

1. Importing the data

Three French cities are available on Airbnbinside.com: Paris, Lyon and Bordeaux. For each city, 3 datasets are available:

- Listing: Information about the listing characteristic and the Airbnb hot
- Review: Comment made by guet
- Calendar: The future evolution of the price of each listing

For the present case study, I chose to analyze mainly the Listings dataset for 2 reasons:

- The dataset was already very dense and informative (more than 70 columns)
- The Rshiny app (free version) sets a limit regarding the size of the dataset (maximum threshold : 1 Go). Thus, I could not join and analyze too many data sources in the same project.

```
# Listings data
listings_Paris <- read.csv(paste0(string, "Airbnb_Paris/listings.csv"), encoding="UTF-8") listings_Bordeaux <-
read.csv(paste0(string, "Airbnb_Bordeaux/listings.csv"), encoding= listings_Lyon <- read.csv(paste0(string,
"Airbnb_Lyon/listings.csv"), encoding="UTF-8")

#Reviews data
reviews_Paris <- read.csv(paste0(string, "Airbnb_Paris/reviews.csv"), encoding="UTF-8") reviews_Bordeaux <-
read.csv(paste0(string, "Airbnb_Bordeaux/reviews.csv"), encoding= reviews_Lyon <- read.csv(paste0(string,
"Airbnb_Lyon/reviews.csv"), encoding="UTF-8")

# Add the city in each data source
listings_Paris = listings_Paris %>% mutate(city = 'Paris') listings_Bordeaux = listings_Bordeaux
%>% mutate(city = 'Bordeaux') listings_Lyon = listings_Lyon %>% mutate(city = 'Lyon')

reviews_Paris = reviews_Paris %>% mutate(city = 'Paris') reviews_Bordeaux =
reviews_Bordeaux %>% mutate(city = 'Bordeaux') reviews_Lyon = reviews_Lyon %>%
mutate(city = 'Lyon')

# Build a centralized dataset for listings and reviews
listings = rbind(listings_Paris, listings_Bordeaux, listings_Lyon) reviews = rbind(reviews_Paris,
reviews_Bordeaux, reviews_Lyon)

# Remove individual files to free memory
rm(listings_Paris, listings_Bordeaux, listings_Lyon, reviews_Paris,
reviews_Bordeaux, reviews_Lyon) gc()
```

I detail below how I concatenated the different datasets to build a central Listings data set. string="~/Documents/BoardInfinity/Project/Data/"

"UTF-8")

```
## $ id                                  : num 130420 23441 5396 132994 7397 ...
## $ listing_url                         : chr "https://www.airbnb.com/rooms/130420" "https://
## $ scrape_id                           : num 20220909140132 20220909140132 20220909140132 20
## $ last_scraped                        : chr "2022-09-10" "2022-09-10" "2022-09-10" "2022-09
## $ source                              : chr "city scrape" "city scrape" "city scrape" "prev
## $ name                                : chr "Charming Apartment 2BR in Paris 9e" "Charming
## $ description                         : chr "This quiet and bright flat is situated on the
## $ neighborhood_overview               : chr "The neighborhood of rue des Martyrs captures t
## $ picture_url                         : chr "https://a0.muscache.com/pictures/947410/5cb34c
```

$ host_id : int 641777 91706 7903 653074 2626 98012 670637 1075
"UTF-8")

2. Data cleaning

The listings dataset has 76 columns and 83,000+ rows. It contains information about :

- Listings scraped by Airbnb inside (description, price, number of bedrooms, bathrooms, neighborhood, number of reviews, availability in the near future,...).
- Hosts (name, verified ID, time since he/she joined Airbnb, Superhost status,...).

```
# Structure of the database "Listings" str(listings)
```

'data.frame': 83184 obs. of 76 variables:

```
## $ host_url                          : chr "https://www.airbnb.com/users/show/641777" "htt
## $ host_name                         : chr "Yassine" "Elise" "Borzou" "Victoire" ...
## $ host_since                        : chr "2011-05-30" "2010-03-12" "2009-02-14" "2011-06
## $ host_location                     : chr "Paris, France" "Paris, France" "İstanbul, Turk
## $ host_about                        : chr "Je suis juriste et je poursuis mes études pour
## $ host_response_time                : chr "within a few hours" "within a day" "within an
## $ host_response_rate                : chr "100%" "100%" "100%" "N/A" ...
## $ host_acceptance_rate              : chr "92%" "100%" "99%" "N/A" ...
## $ host_is_superhost                 : chr "t" "t" "f" "f" ...
## $ host_thumbnail_url                : chr "https://a0.muscache.com/im/pictures/user/285ce
## $ host_picture_url                  : chr "https://a0.muscache.com/im/pictures/user/285ce
## $ host_neighbourhood                : chr "Pigalle - Saint-Georges" "Montmartre" "Saint-P
## $ host_listings_count               : int 1 1 1 1 2 2 3 1 1 1 ...
## $ host_total_listings_count         : int 1 3 1 1 8 2 4 1 1 1 ...
## $ host_verifications                : chr "['email', 'phone']" "['email', 'phone']" "['em
## $ host_has_profile_pic              : chr "t" "t" "t" "t" ...
## $ host_identity_verified            : chr "t" "t" "t" "f" ...
## $ neighbourhood                     : chr "Paris, Ile-de-France, France" "" "Paris, Ile-d
## $ neighbourhood_cleansed            : chr "Opéra" "Buttes-Montmartre" "Hôtel-de-Ville" "T
## $ neighbourhood_group_cleansed      : chr NA NA NA NA ...
## $ latitude                          : num 48.9 48.9 48.9 48.9 48.9 ...
## $ longitude                         : num 2.34 2.33 2.36 2.36 2.35 ...
## $ property_type                     : chr "Entire rental unit" "Entire rental unit" "Enti
## $ room_type                         : chr "Entire home/apt" "Entire home/apt" "Entire hom
## $ accommodates                      : int 6 2 2 2 4 2 2 3 2 2 ...
## $ bathrooms                         : logi NA NA NA NA NA NA ...
## $ bathrooms_text                    : chr "1 bath" "1 bath" "1 bath" "1 bath" ...
## $ bedrooms                          : int 2 NA NA 1 2 1 1 2 1 NA ...
## $ beds                              : int 3 1 1 1 2 1 1 2 2 1 ...
## $ amenities                         : chr "[\"Ethernet connection\", \"Hair dryer\", \"Ha
## $ price                             : chr "$213.00" "$70.00" "$110.00" "$90.00" ...
## $ minimum_nights                    : int 1 30 1 365 10 30 3 6 4 30 ...
## $ maximum_nights                    : int 30 305 1125 365 130 180 365 21 730 1124 ...
## $ minimum_minimum_nights            : int 1 30 1 365 7 30 3 6 4 30 ...
## $ maximum_minimum_nights            : int 1 30 1 365 10 30 3 6 4 30 ...
## $ minimum_maximum_nights            : int 1125 305 1125 365 130 180 365 21 1125 1125 ...
## $ maximum_maximum_nights            : int 1125 305 1125 365 130 180 365 21 1125 1125 ...
## $ minimum_nights_avg_ntm            : num 1 30 1 365 9.9 30 3 6 4 30 ...
## $ maximum_nights_avg_ntm            : num 1125 305 1125 365 130 ...
## $ calendar_updated                  : logi NA NA NA NA NA NA ...
## $ has_availability                  : chr "t" "t" "t" "t" ...
## $ availability_30                   : int 4 0 4 30 0 2 0 1 0 2 ...
## $ availability_60                   : int 26 0 20 60 13 14 0 1 0 2 ...
## $ availability_90                   : int 38 0 50 90 22 44 7 1 0 2 ...
## $ availability_365                  : int 301 115 50 365 207 319 282 3 0 126 ...
## $ calendar_last_scraped             : chr "2022-09-10" "2022-09-10" "2022-09-10" "2022-09
## $ number_of_reviews                 : int 188 84 309 35 313 30 199 165 12 295 ...
## $ number_of_reviews_ltm             : int 32 3 48 0 35 0 7 0 2 3 ...
## $ number_of_reviews_l30d            : int 2 1 2 0 1 0 0 0 1 1 ...
```

```
## $ first_review                : chr "2011-06-30" "2010-04-04" "2009-06-30" "2011-06
## $ last_review                 : chr "2022-09-06" "2022-08-31" "2022-08-19" "2017-03
## $ review_scores_rating        : num 4.6 4.72 4.53 4.59 4.71 4.48 4.68 4.97 4.73 4.8
## $ review_scores_accuracy      : num 4.66 4.57 4.57 4.53 4.8 4.5 4.84 4.98 4.82 4.89
## $ review_scores_cleanliness   : num 4.37 4.6 4.49 4.66 4.43 4.05 4.36 4.97 4.45 4.9
```

$ review_scores_checkin : num 4.94 4.83 4.79 4.58 4.91 4.91 4.81 5 4.91 4.82
$ review_scores_communication : num 4.96 4.96 4.82 4.59 4.87 4.91 4.79 5 5 4.92 ...
$ review_scores_location : num 4.81 4.63 4.95 4.91 4.93 4.82 4.72 4.97 4.64 4.
$ review_scores_value : num 4.43 4.64 4.54 4.5 4.72 4.64 4.61 4.98 4.55 4.7
$ license : chr "7510900711502" "Available with a mobility leas ## $ instant_bookable : chr "f" "f" "f" "f" ...
$ calculated_host_listings_count: int 1 1 1 1 2 2 1 1 1 1 ... ## $
calculated_host_listings_count_entire_homes : int 1 1 1 1 2 2 1 1 1 1 ... ## $
calculated_host_listings_count_private_rooms: int 0 0 0 0 0 0 0 0 0 0 ... ## $
calculated_host_listings_count_shared_rooms : int 0 0 0 0 0 0 0 0 0 0 ...
$ reviews_per_month : num 1.38 0.55 1.92 0.26 2.25 0.2 1.46 1.1 0.11 2.16 ## $ city : chr "Paris" "Paris" "Paris" "Paris" ...

The results show that some columns in the listings dataset do not have the correct format. For instance, there are numerical or date columns that are considered as strings. The format of such columns is corrected so that they can be used for data analysis.

),

),

```
# Transform date columns (considered as strings) into date format listings <-listings
%>% mutate(host_since = ymd(host_since), last_scraped = ymd(last_scraped),
           calendar_last_scraped = ymd(calendar_last_scraped),
           first_review = ymd(first_review), last_review = ymd(last_review))

# Tranform percentage columns (considered as strings due to the '%' sign) into floats listings$host_response_rate <-
gsub("%","", listings$host_response_rate) listings$host_acceptance_rate <-gsub("%","",
listings$host_acceptance_rate) listings$host_response_rate = as.numeric(listings$host_response_rate) /100
listings$host_acceptance_rate = as.numeric(listings$host_acceptance_rate) /100

#Transform price column (considered as string due to the '$' sign) into & numeric variable listings %>%
filter(!grepl('$', price))
listings$price <- as.numeric(str_sub(listings$price, 2, -2))

# Transform boolean variables (considered as string) into integer flags listings = listings %>%
mutate(instant_bookable = case_when(instant_bookable=='f' ~ 0, instant_bookable=='t' ~ 1 has_availability =
case_when(has_availability=='f' ~ 0, has_availability=='t' ~ 1 host_identity_verified =
case_when(host_identity_verified=='f' ~ 0 host_has_profile_pic = case_when(host_has_profile_pic=='f' ~ 0
           host_is_superhost = case_when(host_is_superhost=='f' ~ 0, host_is_superhost=='t' )

#Transform strings into factors as they are categorical variables listings$host_response_time
= as.factor(listings$host_response_time) listings$room_type = as.factor(listings$room_type)
listings$property_type = as.factor(listings$property_type)
listings_table = as.data.frame(table(listings$room_type, listings$property_type))

# Transform IDs into strings listings$id =
as.character(listings$id) listings$host_id =
as.character(listings$host_id)
```

```
                                                                                    host_identity_verified=='t
                                                                  , host_has_profile_pic=='t' ~ 1) ~ 1),
```

```
# Finally, some date cleaning in the Reviews dataset
reviews$date = ymd(reviews$date) reviews$year =
year(reviews$date)
```

3. Data preparation

First, we print some basic descriptive statistics to :

- Check that every column has the correct format
- Understand a bit better the dataset

```
# Some descriptive statistics about the Listings database summary(listings)
```

```
##             id                          listing_url                       scrape_id
## Length:83184  Length:83184       Min.       :20220909140100 ## Class :character
           Class :character    1st Qu.:20220909140100 ## Mode :character
           Mode :character  Median :20220909140100
##          Mean     :20220909942800 ##         3rd Qu.:20220912200200 ##           Max.
           :20220912200200
##

##         last_scraped                source                      name                description
## Min.           :2022-09-09         Length:83184               Length:83184               Length:83184
## 1st Qu.:2022-09-10                  Class :character           Class :character           Class :character
## Median :2022-09-10                  Mode :character            Mode :character            Mode :character
## Mean :2022-09-10 ## 3rd
Qu.:2022-09-12    ##    Max.
:2022-09-15
##
## neighborhood_overview picture_url                          host_id                    host_url
## Length:83184                       Length:83184               Length:83184               Length:83184
## Class :character                    Class :character          Class :character           Class :character
## Mode :character                     Mode :character           Mode :character            Mode :character
##
##
##
##
##       host_name                   host_since                host_location              host_about
## Length:83184                 Min.       :2008-04-17         Length:83184               Length:83184
## Class :character              1st Qu.:2014-06-29            Class :character           Class :character
## Mode :character               Median :2015-12-14            Mode :character            Mode :character
##          Mean     :2016-06-28 ##    3rd Qu.:2018-05-06
##          Max.      :2022-09-10

##                                    NA's       :9
##                           host_response_time host_response_rate host_acceptance_rate
##                                    :        9        Min.       :0.00               Min.       :0.000
## a few days or more: 1810           1st Qu.:0.95        1st Qu.:0.710 ## N/A
:33489    Median :1.00       Median :0.940 ## within a day        : 7950    Mean     :0.93
Mean     :0.812
## within a few hours:10346                  3rd Qu.:1.00               3rd Qu.:1.000
## within an hour               :29580        Max.       :1.00               Max.       :1.000
##                                           NA's       :33498             NA's       :29681
## host_is_superhost host_thumbnail_url host_picture_url                        host_neighbourhood
## Min.           :0.0000          Length:83184               Length:83184               Length:83184
## 1st Qu.:0.0000 Class :character    Class :character    Class :character ## Median :0.0000 Mode
:character            Mode :character  Mode :character
## Mean :0.1356 ## 3rd
Qu.:0.0000    ##    Max.
:1.0000
## NA's           :49
## host_listings_count host_total_listings_count host_verifications
## Min.           :       0.00          Min.       :       1.00                    Length:83184
```

```
## 1st Qu.:          1.00       1st Qu.:  1.00       Class :character ## Median :          1.00
Median :          2.00       Mode :character
## Mean : 16.53    Mean     : 25.47 ## 3rd Qu.:
2.00       3rd Qu.: 4.00 ## Max.       :1732.00Max.
:2316.00
## NA's          :9                    NA's       :9
## host_has_profile_pic host_identity_verified neighbourhood
## Min.          :0.0000              Min.       :0.0000                  Length:83184
## 1st Qu.:1.0000 1st Qu.:1.0000     Class :character ## Median :1.0000 Median
:1.0000   Mode :character
## Mean :0.9879 Mean :0.8303 ## 3rd Qu.:1.0000
3rd Qu.:1.0000 ## Max. :1.0000 Max. :1.0000
## NA's          :9                    NA's       :9
## neighbourhood_cleansed neighbourhood_group_cleansed                  latitude
## Length:83184  Length:83184       Min.       :44.72 ## Class :character  Class :character
1st Qu.:45.78 ## Mode :character   Mode :character  Median :48.85
##                                                                      Mean       :47.93
##                                                                      3rd Qu.:48.87
##                                                                      Max.       :48.91
##
##         longitude                                property_type                        room_type
## Min.              :-0.8508Entire rental unit                  :57423          Entire home/apt:68253
## 1st Qu.: 2.3010          Private room in rental unit: 7893                Hotel room          : 1137
## Median : 2.3473          Entire condo                : 4551          Private room        :13280
## Mean              : 2.2887Entire home                  : 2255          Shared room         : 514
## 3rd Qu.: 2.3821          Room in boutique hotel      : 1701

## Max.              : 4.9178Entire loft                  : 1293

##                          (Other)                     : 8068

##      accommodates       bathrooms          bathrooms_text              bedrooms

## Min.          : 0.000      Mode:logical       Length:83184             Min.       : 1.000

## 1st Qu.: 2.000             NA's:83184           Class :character         1st Qu.: 1.000

## Median : 2.000                                  Mode :character          Median : 1.000

## Mean          : 3.135                                                    Mean       : 1.426

## 3rd Qu.: 4.000                                                           3rd Qu.: 2.000

## Max.          :16.000                                                    Max.       :50.000

##                                                                          NA's       :12341

##             beds             amenities                      price             minimum_nights

## Min.          : 1.000      Length:83184             Min.       : 0.0       Min.       :          1.00

## 1st Qu.: 1.000             Class :character           1st Qu.: 60.0        1st Qu.:          2.00
```

```
## Median : 1.000            Mode :character            Median : 90.0        Median :           3.00

## Mean          : 1.774                                Mean       :130.4       Mean          : 77.53
## 3rd Qu.: 2.000                                       3rd Qu.:150.0              3rd Qu.: 30.00
## Max.          :90.000                                Max.       :999.0       Max.          :9999.00
## NA's          :1246                                  NA's       :671
## maximum_nights            minimum_minimum_nights maximum_minimum_nights
## Min.          :            1      Min.       :       1.0                Min.       :       1.00
## 1st Qu.:                  60      1st Qu.:          2.0                 1st Qu.:          2.00
## Median :               1125       Median :          3.0                 Median :          3.00
## Mean          :          800      Mean         : 76.9                   Mean         : 79.83
## 3rd Qu.:               1125       3rd Qu.: 30.0                         3rd Qu.: 30.00
## Max.          :10000000       Max.       :9999.0                  Max.       :9999.00
##                                  NA's       :6                       NA's       :6
## minimum_maximum_nights maximum_maximum_nights minimum_nights_avg_ntm
## Min.          :                1          Min.       :                1          Min.       :       1.00
## 1st Qu.: 90 1st Qu.: 360 1st Qu.: 2.00 ## Median : 1125 Median : 1125 Median :
3.00 ## Mean : 104120 Mean : 388158 Mean : 79.37 ## 3rd Qu.: 1125 3rd Qu.: 1125
3rd Qu.: 30.00 ## Max. :2147483647 Max. :2147483647 Max. :9999.00
## NA's          :6                       NA's       :6                       NA's       :6
## maximum_nights_avg_ntm calendar_updated has_availability availability_30
## Min. : 1 Mode:logical Min. :0.0000 Min. : 0.000 ## 1st Qu.: 150 NA's:83184 1st Qu.:1.0000 1st
Qu.: 0.000 ## Median : 1125 Median :1.0000 Median : 0.000 ## Mean : 206122 Mean :0.9984
Mean : 4.392 ## 3rd Qu.: 1125 3rd Qu.:1.0000 3rd Qu.: 5.000 ## Max. :2147483647 Max. :1.0000
Max. :30.000
## NA's          :6
## availability_60 availability_90 availability_365 calendar_last_scraped
## Min. : 0 Min. : 0.00 Min. : 0.0 Min. :2022-09-09 ## 1st Qu.: 0 1st Qu.: 0.00 1st Qu.: 0.0 1st
Qu.:2022-09-10 ## Median : 0 Median : 1.00 Median : 33.0 Median :2022-09-10 ## Mean :12
Mean :21.59 Mean :109.3 Mean :2022-09-10 ## 3rd Qu.:19 3rd Qu.:42.00 3rd Qu.:240.0 3rd
Qu.:2022-09-12 ## Max. :60 Max. :90.00 Max. :365.0 Max. :2022-09-15
##
## number_of_reviews number_of_reviews_ltm number_of_reviews_l30d
## Min.          :       0.00       Min.       :       0.000            Min.         : 0.0000
## 1st Qu.:              1.00        1st Qu.:         0.000             1st Qu.: 0.0000
## Median :              7.00        Median :         1.000             Median : 0.0000
## Mean          : 25.23        Mean       :       7.573             Mean       : 0.6904
## 3rd Qu.: 24.00                  3rd Qu.:         8.000             3rd Qu.: 1.0000
## Max.          :2391.00        Max.       :1356.000             Max.       :92.0000
##
##          first_review                  last_review                  review_scores_rating
## Min.          :2009-06-30        Min.          :2010-05-28Min.          :0.000
## 1st Qu.:2017-06-08                   1st Qu.:2020-01-13    1st Qu.:4.530
## Median :2019-07-05                   Median :2022-07-10    Median :4.800
## Mean          :2019-05-04        Mean          :2021-04-25Mean          :4.623
## 3rd Qu.:2021-11-01                   3rd Qu.:2022-08-22    3rd Qu.:5.000
## Max.          :2022-09-12        Max.          :2022-09-12Max.          :5.000
## NA's          :15542                 NA's       :15542                 NA's       :15542
## review_scores_accuracy review_scores_cleanliness review_scores_checkin
## Min.          :0.000                 Min.       :0.000                       Min.       :0.000
```

1st Qu.:4.700 1st Qu.:4.480 1st Qu.:4.770 ## Median :4.890 Median :4.750 Median :4.930 ## Mean :4.765 Mean :4.609 Mean :4.803 ## 3rd Qu.:5.000 3rd Qu.:4.980 3rd Qu.:5.000 ## Max. :5.000 Max. :5.000 Max. :5.000 ## NA's :16355 NA's :16347 NA's :16368
review_scores_communication review_scores_location review_scores_value
Min. :0.000 Min. :0.000 Min. :0.000
1st Qu.:4.780 1st Qu.:4.690 1st Qu.:4.500
Median :4.940 Median :4.880 Median :4.720
Mean :4.811 Mean :4.774 Mean :4.618
3rd Qu.:5.000 3rd Qu.:5.000 3rd Qu.:4.890
Max. :5.000 Max. :5.000 Max. :5.000
NA's :16352 NA's :16370 NA's :16372
license instant_bookable calculated_host_listings_count
Length:83184 Min. :0.000 Min. : 1.00
Class :character 1st Qu.:0.000 1st Qu.: 1.00
Mode :character Median :0.000 Median : 1.00
Mean :0.329 Mean : 10.15
3rd Qu.:1.000 3rd Qu.: 2.00
Max. :1.000 Max. :269.00
##
calculated_host_listings_count_entire_homes
Min. : 0.000 ## 1st Qu.: 1.000 ## Median : 1.000 ## Mean : 9.516
3rd Qu.: 1.000
Max. :269.000
##
calculated_host_listings_count_private_rooms
Min. : 0.0000 ## 1st Qu.: 0.0000 ## Median : 0.0000 ## Mean : 0.4907
3rd Qu.: 0.0000 ## Max. :67.0000
##
calculated_host_listings_count_shared_rooms reviews_per_month
Min. : 0.00000 Min. : 0.010
1st Qu.: 0.00000 1st Qu.: 0.170
Median : 0.00000 Median : 0.580
Mean : 0.02601 Mean : 1.141
3rd Qu.: 0.00000 3rd Qu.: 1.530
Max. :24.00000 Max. :89.900
NA's :15542
city
Length:83184
Class :character ## Mode :character
##
##
##
##

Next, we compute some key summary statistics that will be used as BANs on the first Overview tab of the Rshiny app. We compute :

- The total number of listings available on Airbnb France website (when it was scraped by InsideAirbnb teams)
- The number of hosts
- The number of big cities
- The average review score
- The average price (in $, excluding cleaning and service fees)

```
# 3.1 BANS / some order of magnitude
BAN_listings = listings %>% summarise(nb_listings =
   n(), nb_hosts = n_distinct(host_id), nb_cities =
   n_distinct(city),
                    avg_satcli = round(mean(review_scores_rating, na.rm = TRUE),2),
                    avg_price = round(mean(price, na.rm = TRUE),2)) BAN_listings
```

```
##              nb_listings nb_hosts nb_cities avg_satcli avg_price
## 1              83184       64051                3            4.62         130.37
```

We also compute BANs for the Reviews dataset:

- Total number of reviews made on the French website
- Number of guests that made at least one review

```
BAN_reviews = reviews %>% filter(year=='2021') %>% summarise(nb_reviews = n(),
   nb_reviewers = n_distinct(reviewer_id))
BAN_reviews
```

```
##         nb_reviews nb_reviewers
## 1          332300            292435
```

Some listings characteristics are also refined:

```
# 3.2. Listings characteristics listings_summary = listings %>% group_by(city, room_type, property_type,
neighbourhood_cleansed, accommodates, bedrooms) summarise(nb_listings_charac = n()) %>%
   mutate(city_neighbourhood_cleansed = paste(city, "-", neighbourhood_cleansed))                    %>%
listings_summary = as.data.frame(listings_summary)

listings$property_type_clean = gsub("Private room in","", listings$property_type) listings = listings %>%
mutate(accommodates_bins = case_when(accommodates <3 ~ "[1;2]", accommodates>= 3 &
accommodates<= 5 ~ "[3;5]", accommodates>= 6 & accommodates<= 9 ~ "[6;9]", accommodates>= 10 ~
"[More than 9]") )
```

Moving on to prices, we compute the top 5 / bottom 5 expensive listings by property type. We also include the number of listings to detect whether or not there are some outliers / strange data in the top / bottom results.

```
# 3.3 Most and least expensive listings most_expensive = listings
%>% group_by(property_type) %>%
   summarise(median_price = median(price, na.rm = TRUE), avg_price =
                round(mean(price, na.rm =TRUE),2), nb_listings = n()
   ) %>%
   arrange(desc(avg_price)) %>%
slice(1:5) most_expensive
```

```
## # A tibble: 5 x 4
##       property_type                median_price avg_price nb_listings
##       <fct>                               <dbl>     <dbl>       <int>
## 1 Shared room in ice dome                   500       500           1
## 2 Floor                                     420       420           1
## 3 Shared room in cabin                      400       400           1
## 4 Castle                                    212.      377.          5
## 5 Room in boutique hotel                    342       368.       1701
```

```
least_expensive = listings %>% group_by(property_type) %>%
   summarise(median_price = median(price, na.rm = TRUE), avg_price =
                round(mean(price, na.rm =TRUE),2), nb_listings = n()
   ) %>%
   arrange(avg_price) %>%
slice(1:5) least_expensive
```

```
## # A tibble: 5 x 4
##       property_type                median_price avg_price nb_listings
##         <fct>      <dbl>    <dbl>    <int> ## 1 Private room in windmill 1          1
1 ## 2 Private room in tipi  21        21        1
## 3 Tent                                    25.5      25.5          2
## 4 Shared room in home                     22        29.6         49
## 5 Shared room in townhouse                30.5      30.2          4
```

We create a specific dataset that excludes listings with no price associated. We will use this dataset for the maps.

```
#Database excluding the few listings with no price specified listings_price =
listings %>% filter(price != "NA")
```

Finally, we prepare the wordcloud displaying the most frequent words used in the amenities column. To do so, we remove noise (numbers, punctuation, white spaces, stopwords, etc) and keep only recurring words (frequency > 50) so that the wordcloud is not overcrowded.

```
#Focus : Wordcloud text <-
listings$amenities docs <-
Corpus(VectorSource(text)) docs <- docs
%>% tm_map(removeNumbers) %>%
tm_map(removePunctuation) %>%
tm_map(stripWhitespace) %>%
    tm_map(function(x) removeWords(x, stopwords("english")))
dtm <- TermDocumentMatrix(docs) matrix <-
as.matrix(dtm)
words <- sort(rowSums(matrix),decreasing=TRUE) df <-
data.frame(word = names(words),freq=words) df = df %>%
filter(freq >50) rm(text, docs, dtm, matrix, words) gc()
```

4. Hosts segmentation

Who are the Airbnb hosts? We want to answer this answer with a segmentation analysis.

We want indeed to categorize hosts into groups so that hosts within a segment are similar enough to be treated similarly, yet different enough from hosts in other segments.

To do so, the Airbnb hosts are grouped into 5 different clusters thanks to an adapted RFM segmentation (Recency, Frequency, Monetary). The segmentation is performed thanks to the kmeans algorithm due to the size of the underlying data. It takes 5 different variables as input:

- *Recency*: When was the last time the host received a customer review on one of his/her listings ? (in months)
- *Frequency*: How many reviews did the host receive in total?
- *Monetary*: What is the average price of a listing (excluding service and cleaning fees)?
- *The length of relationship*: For how long has the host been on Airbnb.com (in years) ?
- *Superhost status*: Has the host been awarded 'Superhost' by Airbnb?

```
# Dataset preparation: Computing adapted RFM indicators for the listings dataset data_segmentation = listings
%>% select(host_id, host_since, last_review, price, host_is_superhost, number_of_reviews) filter(price != "NA",
host_is_superhost != "NA", !is.na(host_since),
                !is.na(last_review)) %>% mutate( length_relationship = as.numeric(difftime(Sys.Date(), host_since,
   units = "days")), recency = as.numeric(difftime(Sys.Date(), last_review, units = "days"))
   ) %>%
   group_by(host_id) %>% summarise( length_relationship_years =
   max(as.integer(length_relationship))/365, recency_months = min(recency)/30, monetary =
   mean(price, na.rm = TRUE),
      host_is_superhost = max(host_is_superhost, na.rm=TRUE),
      number_of_reviews = sum(number_of_reviews, na.rm = TRUE) )
```

%>%

```
# Assign contact id as row names, remove id from data rownames(data_segmentation) =
data_segmentation$host_id data_segmentation = data_segmentation[, -1]

# Perform kmeans segmentation on standardized data set.seed(10)
k = kmeans(x = scale(data_segmentation), centers = 5, nstart = 50)
```

We compute the number of hosts in each cluster.

```
# Print cluster size print(k$size)
```

```
## [1]              74 15773 18666 7938 10578
```

We then print the clusters characteristics to interpret them from a business point of view.

```
# Print standardized centers, and then un-standardized centers, one segment at a time print(k$centers)
```

```
##          length_relationship_years recency_months              monetary host_is_superhost
## 1                        -0.3438178      -0.7471923 1.13687662   0.2985220
## 2                         0.3534528       1.3392506 -0.41226797  -0.4195767
## 3                         0.5103884      -0.5706163 0.21376096   -0.4202866
## 4                         0.0610341      -0.6923706 0.17900173    2.3792837
## 5                        -1.4710688      -0.4652624 0.09525393   -0.4202866
##   number_of_reviews
## 1 19.0069460618 ##
2  -0.1948272240  ##  3
0.0009001132    ##    4
0.3392470594   ##   5  -
0.0986243974
for (i in 1:5) {
   print(colMeans(data_segmentation[k$cluster == i, ]))
}
```

```
## length_relationship_years                 recency_months                      monetary
##                  5.7689004                      2.0833333                215.4297275
##              host_is_superhost          number_of_reviews

##                  0.2567568                   2683.2702703

## length_relationship_years                 recency_months                      monetary
##                  7.5737394142                   54.5068915235            74.4179607708
##              host_is_superhost          number_of_reviews

##                  0.0002535979                   12.3540226970

## length_relationship_years                 recency_months                      monetary
##                  7.979957                       6.519947                 131.402600
##              host_is_superhost          number_of_reviews

##                  0.000000                       39.579181

## length_relationship_years                 recency_months                      monetary
##                  6.816833                       3.460771                 128.238621
##              host_is_superhost          number_of_reviews

##                  1.000000                       86.642353

## length_relationship_years                 recency_months                      monetary
##                  2.851085                       9.167048                 120.615430
##              host_is_superhost          number_of_reviews

##                  0.000000                       25.735583
```

Finally, we build the dataset that will be used for plotting the different segments.

```
#Final segmentation dataset cluster =
k[["cluster"]]
merged_data = cbind(data_segmentation, cluster) rm(data_segmentation)
gc()

# Clusters colors for the graphs couleurs = c("1" =
"hotpink",
                    "2" = "darkgoldenrod1",
                    "3" = "blue4",
                    "4" = "chocolate4",
                    "5" = "chartreuse4")

# Recap table about clusters characteristics seg_summary = merged_data %>% group_by(cluster) %>%
summarize(nb_hosts = n(), mean_length_relationship = round(mean(length_relationship_years),0),
mean_recency = round(mean(recency_months), 0), mean_price = round(mean(monetary),0),
                  pct_superhosts = round(mean(host_is_superhost), 1), mean_number_of_reviews =
                  round(mean(number_of_reviews),0)) %>%
  mutate(pct_hosts = round(nb_hosts / sum(nb_hosts),2)) seg_summary
```

```
## # A tibble: 5 x 8
##   cluster nb_hosts mean_length_relatio~1 mean_~2 mean_~3 pct_s~4 mean_~5 pct_h~6
##     <int>    <int>                 <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1       1       74                     6       2     215     0.3    2683       0
## 2       2    15773                     8      55      74       0      12     0.3
## 3       3    18666                     8       7     131       0      40    0.35 ## 4       4     7938       7
3     128       1      87    0.15
## 5       5    10578                     3       9     121       0      26     0.2
## # ... with abbreviated variable names 1: mean_length_relationship,
## #   2: mean_recency, 3: mean_price, 4: pct_superhosts,
## #   5: mean_number_of_reviews, 6: pct_hosts
```

5. Reviews analysis

In this section, we compute correlations between the reviews score and listings / hosts characteristics. We want to know what factors influence positively (respectively negatively) the guests satisfaction.

5.1. Correlation between the rating and listings/hosts characteristics

```
# 5.1 : Compute the correlation between the rating and listings/hosts characteristics corr_listings = listings
%>% mutate(is_paris = case_when(city=="Paris"~1, TRUE ~0), is_lyon = case_when(city=="Lyon"~1, TRUE ~0),
is_bordeaux = case_when(city=="Bordeaux"~1, TRUE ~0), shared_room = case_when(room_type=="Shared
room" ~1, TRUE ~0), entire_home = case_when(room_type=="Entire home/apt" ~1, TRUE ~0), private_room
= case_when(room_type=="Private room" ~1, TRUE ~0)) %>% select(review_scores_rating, price,
number_of_reviews, is_paris, is_lyon, is_bordeaux, shared_room, entire_home, private_room,
                host_is_superhost, host_response_rate, host_identity_verified, accommodates,
   beds, availability_30) %>% cor(use = "complete.obs")

# Rounding up the results
res_listings <- round(corr_listings, 2) res_listings
```

##	review_scores_rating	price	number_of_reviews	is_paris
## review_scores_rating	1.00	0.00	0.05	-0.04
## price	0.00	1.00	-0.04	0.21
## number_of_reviews	0.05	-0.04	1.00	-0.05
## is_paris	-0.04	0.21	-0.05	1.00
## is_lyon	0.00	-0.16	0.05	-0.62
## is_bordeaux	0.06	-0.12	0.02	-0.68
## shared_room	-0.03	-0.05	0.01	0.01
## entire_home	-0.02	0.11	-0.07	0.08

private_room 0.03 -0.16 0.06 -0.11

host_is_superhost 0.20 0.03 0.25 -0.07

host_response_rate 0.09 0.03 0.10 -0.03

host_identity_verified 0.00 0.06 0.08 0.04

accommodates -0.02 0.53 0.00 -0.07

beds -0.01 0.38 0.00 -0.04

availability_30 -0.11 0.15 -0.01 -0.22

##	is_lyon	is_bordeaux	shared_room	entire_home	private_room
## review_scores_rating	0.00	0.06	-0.03	-0.02	0.03
## price	-0.16	-0.12	-0.05	0.11	-0.16
## number_of_reviews	0.05	0.02	0.01	-0.07	0.06
## is_paris	-0.62	-0.68	0.01	0.08	-0.11
## is_lyon	1.00	-0.16	-0.01	-0.05	0.06
## is_bordeaux	-0.16	1.00	-0.01	-0.06	0.07
## shared_room	-0.01	-0.01	1.00	-0.16	-0.03
## entire_home	-0.05	-0.06	-0.16	1.00	-0.93
## private_room	0.06	0.07	-0.03	-0.93	1.00
## host_is_superhost	0.03	0.06	-0.02	-0.06	0.07
## host_response_rate	0.01	0.03	-0.01	0.02	-0.02
## host_identity_verified	0.00	-0.05	0.00	0.05	-0.06
## accommodates	0.00	0.09	-0.06	0.28	-0.27
## beds	-0.01	0.06	0.02	0.13	-0.13
## availability_30	0.13	0.15	0.07	-0.17	0.13

##	host_is_superhost	host_response_rate
## review_scores_rating	0.20	0.09
## price	0.03	0.03
## number_of_reviews	0.25	0.10
## is_paris	-0.07	-0.03
## is_lyon	0.03	0.01
## is_bordeaux	0.06	0.03
## shared_room	-0.02	-0.01

```
## entire_home                -0.06                0.02
## private_room                0.07               -0.02
## host_is_superhost           1.00                0.16
## host_response_rate          0.16                1.00
## host_identity_verified      0.07                0.06
## accommodates               -0.01                0.04
## beds                       -0.01                0.03
## availability_30            -0.03               -0.06
##                      host_identity_verified accommodates beds
## review_scores_rating                   0.00        -0.02 -0.01
## price                                  0.06         0.53 0.38
## number_of_reviews                      0.08         0.00 0.00
## is_paris                               0.04        -0.07 -0.04
## is_lyon                                0.00         0.00 -0.01
## is_bordeaux                           -0.05         0.09 0.06
## shared_room                            0.00        -0.06 0.02
## entire_home                            0.05         0.28 0.13
## private_room                          -0.06        -0.27 -0.13
## host_is_superhost                      0.07        -0.01 -0.01
## host_response_rate                     0.06         0.04 0.03
## host_identity_verified                 1.00         0.05 0.03
## accommodates                           0.05         1.00 0.72
## beds                                   0.03         0.72 1.00
## availability_30                       -0.01         0.05 0.05
##                      availability_30
## review_scores_rating           -0.11
## price                           0.15
## number_of_reviews              -0.01
## is_paris                       -0.22
## is_lyon                         0.13
## is_bordeaux                     0.15
```

```
## shared_room                    0.07
## entire_home                   -0.17
## private_room                   0.13
## host_is_superhost             -0.03
## host_response_rate            -0.06
## host_identity_verified        -0.01
## accommodates                   0.05
## beds                           0.05
## availability_30                1.00
```

Preparing the data that will be used for plots:

```
liste = as.data.frame(res_listings) %>% select(review_scores_rating) %>%
  arrange(desc(review_scores_rating)) object <- rownames(liste) liste = liste %>%
  cbind(object)
```

5.2. Correlation between the rating and available amenities

```
# We create flags for each amenity
listings$Pool = grepl("pool",listings$amenities) listings$BBQ =
grepl("BBQ",listings$amenities) listings$Garden =
grepl("garden",listings$amenities) listings$Balcony =
grepl("balcony",listings$amenities) listings$Washer =
grepl("washer",listings$amenities) listings$Dryer =
grepl("dryer",listings$amenities) listings$Oven =
grepl("oven",listings$amenities) listings$Fridge =
grepl("refrigerator",listings$amenities) listings$Microwave =
grepl("microwave",listings$amenities) listings$Dishwasher =
grepl("Dishwasher",listings$amenities) listings$Elevator =
grepl("Elevator",listings$amenities) listings$Freezer =
grepl("freezer",listings$amenities) listings$Iron =
grepl("iron",listings$amenities) listings$TV = grepl("TV",listings$amenities)
listings$Game_console = grepl("Game console",listings$amenities)
listings$Parking = grepl("parking",listings$amenities) listings$Aircon = grepl("Air
conditioning",listings$amenities) listings$Wifi = grepl("wifi",listings$amenities)
listings = listings %>% mutate(sum_amenities =
                  Pool + BBQ + Garden + Balcony +
                        Washer + Dryer + Oven + Fridge + Microwave + Dishwasher + Elevator +
                  Freezer + Iron + Parking + Aircon + Wifi + Game_console + TV )
```

We then compute correlations with the review score:

```
corr_amenities = listings %>% select(review_scores_rating,
   sum_amenities, Pool, BBQ, Garden, Balcony,
              Washer, Dryer, Oven, Fridge, Microwave, Dishwasher, Elevator ,
              Freezer, Iron, Parking, Aircon, Wifi, Game_console, TV,
              ) %>% cor(use =
   "complete.obs")

# Rounding up the results
res_amenities <- round(corr_amenities, 2) res_amenities
```

review_scores_rating sum_amenities Pool BBQ Garden ## review_scores_rating 1.00 0.15 0.04 0.04 0.06 ## sum_amenities 0.15 1.00 0.26 0.34 0.35 ## Pool 0.04 0.26 1.00 0.35 0.24 ## BBQ 0.04 0.34 0.35 1.00 0.32 ## Garden 0.06 0.35 0.24 0.32 1.00 ## Balcony 0.09 0.51 0.12 0.19 0.22 ## Washer 0.13 0.72 0.10 0.16 0.17 ## Dryer 0.11 0.47 0.04 0.05 0.07 ## Oven 0.06 0.32 0.11 0.07 0.13 ## Fridge 0.04 0.24 0.14 0.05 0.11 ## Microwave 0.00 0.00 0.03 0.00 0.00 ## Dishwasher 0.12 0.70 0.10 0.17 0.13 ## Elevator 0.01 0.31 -0.06 -0.09 -0.06 ## Freezer 0.01 0.03 0.00 0.00 0.02 ## Iron 0.00 0.00 0.00 0.00 0.00 ## Parking 0.12 0.56 0.13 0.18 0.20 ## Aircon -0.01 0.20 0.04 0.05 0.00 ## Wifi 0.07 0.37 0.05 0.08 0.14
Game_console 0.03 0.23 0.07 0.12 0.07
TV 0.00 0.41 0.06 0.07 0.03

##	Balcony	Washer	Dryer	Oven	Fridge	Microwave	Dishwasher
## review_scores_rating	0.09	0.13	0.11	0.06	0.04	0.00	0.12
## sum_amenities	0.51	0.72	0.47	0.32	0.24	0.00	0.70
## Pool	0.12	0.10	0.04	0.11	0.14	0.03	0.10
## BBQ	0.19	0.16	0.05	0.07	0.05	0.00	0.17
## Garden	0.22	0.17	0.07	0.13	0.11	0.00	0.13
## Balcony	1.00	0.28	0.12	0.10	0.07	0.01	0.27
## Washer	0.28	1.00	0.24	0.20	0.12	0.00	0.88
## Dryer	0.12	0.24	1.00	0.06	0.05	-0.01	0.23
## Oven	0.10	0.20	0.06	1.00	0.38	0.02	0.17
## Fridge	0.07	0.12	0.05	0.38	1.00	0.00	0.09
## Microwave	0.01	0.00	-0.01	0.02	0.00	1.00	0.00
## Dishwasher	0.27	0.88	0.23	0.17	0.09	0.00	1.00
## Elevator	0.16	0.07	0.08	0.03	0.01	0.00	0.09
## Freezer	0.00	0.01	0.00	0.01	0.10	0.00	0.01
## Iron	0.00	0.00	0.00	0.02	0.00	0.00	0.00
## Parking	0.25	0.28	0.17	0.13	0.10	0.00	0.24
## Aircon	0.01	-0.01	0.09	-0.04	-0.02	0.01	0.02
## Wifi	0.14	0.18	0.10	0.10	0.09	0.00	0.14

Game_console 0.08 0.12 0.04 0.07 0.04 0.00 0.12

TV 0.06 0.14 0.15 0.02 0.02 -0.01 0.15

Elevator Freezer Iron Parking Aircon Wifi Game_console
review_scores_rating 0.01 0.01 0.00 0.12 -0.01 0.07 0.03

sum_amenities 0.31 0.03 0.00 0.56 0.20 0.37 0.23

Pool -0.06 0.00 0.00 0.13 0.04 0.05 0.07

BBQ -0.09 0.00 0.00 0.18 0.05 0.08 0.12

Garden -0.06 0.02 0.00 0.20 0.00 0.14 0.07

Balcony 0.16 0.00 0.00 0.25 0.01 0.14 0.08

Washer 0.07 0.01 0.00 0.28 -0.01 0.18 0.12

Dryer 0.08 0.00 0.00 0.17 0.09 0.10 0.04

Oven 0.03 0.01 0.02 0.13 -0.04 0.10 0.07

Fridge 0.01 0.10 0.00 0.10 -0.02 0.09 0.04

Microwave 0.00 0.00 0.00 0.00 0.01 0.00 0.00

Dishwasher 0.09 0.01 0.00 0.24 0.02 0.14 0.12

Elevator 1.00 0.00 0.00 0.05 0.04 0.02 0.01

Freezer 0.00 1.00 0.00 0.02 0.00 0.01 0.00

Iron 0.00 0.00 1.00 0.00 0.00 0.00 0.00

Parking 0.05 0.02 0.00 1.00 0.04 0.21 0.09

Aircon 0.04 0.00 0.00 0.04 1.00 -0.01 0.01

Wifi 0.02 0.01 0.00 0.21 -0.01 1.00 0.08
Game_console 0.01 0.00 0.00 0.09 0.01 0.08 1.00
TV 0.06 0.00 0.00 0.12 0.13 0.05 0.08
TV ## review_scores_rating
0.00
sum_amenities 0.41
Pool 0.06

BBQ 0.07

Garden 0.03

Balcony 0.06

Washer 0.14

Dryer 0.15

Oven 0.02

```
## Fridge                    0.02
## Microwave                -0.01
## Dishwasher                0.15
## Elevator                  0.06
## Freezer                   0.00
## Iron                      0.00
## Parking                   0.12
## Aircon                    0.13
## Wifi                      0.05
## Game_console              0.08
## TV                        1.00
```

And we prepare the dataset that will be used for plots:

```
liste_amenities = as.data.frame(res_amenities) %>%
select(review_scores_rating) %>% arrange(desc(review_scores_rating))

Amenities <- rownames(liste_amenities)
liste_amenities = liste_amenities %>% cbind(Amenities)
```

6. Final cleaning / Preparation of the Rshiny app

We keep only the useful objects and reduce the size of dataframes so that the Rshiny app can be published with the free version on shinyapps.io

```
# Remove useless objects rm(i, k,
cluster, object)

#Reduce the size of datasets listings_price = listings_price %>% select(id, listing_url, name, property_type, neighbour
price, city, latitude, longitude, accommodates, number_of_reviews, review_scores_rating, host_is_superhost)

listings = listings %>% select(-starts_with("maximum")) %>% select(-
   starts_with("calculated")) %>% select(-starts_with("minimum"))
   %>%
   select(-c("scrape_id", "last_scraped", "source", "description",
                  "neighborhood_overview", "neighbourhood","picture_url",



                  "host_url", "host_name", "host_location", "host_about", "host_thumbnail_url", "host_picture_ "calendar_updated"))
```

We export all files to csv format. They will be used by the Rshiny app as inputs.

```
#Export all data into csv format to integrate them after in the Rshiny app
```

```
write.csv(Amenities, "Amenities.csv", row.names = FALSE) write.csv(BAN_listings, "BAN_listings.csv",
row.names = FALSE) write.csv(BAN_reviews, "BAN_reviews.csv", row.names = FALSE)
write.csv(corr_amenities, "corr_amenities.csv", row.names = FALSE) write.csv(corr_listings,
"corr_listings.csv", row.names = FALSE) write.csv(couleurs, "couleurs.csv", row.names = FALSE)
write.csv(df, "df.csv", row.names = FALSE)
write.csv(least_expensive, "least_expensive.csv", row.names = FALSE)
write.csv(liste, "liste.csv", row.names = FALSE)
write.csv(liste_amenities, "liste_amenities.csv", row.names = FALSE)
write.csv(listings, "listings.csv", row.names = FALSE) write.csv(listings_price, "listings_price.csv",
row.names = FALSE) write.csv(listings_summary, "listings_summary.csv", row.names = FALSE)
write.csv(listings_table, "listings_table.csv", row.names = FALSE) write.csv(merged_data,
"merged_data.csv", row.names = FALSE) write.csv(most_expensive, "most_expensive.csv", row.names
= FALSE) write.csv(res_amenities, "res_amenities.csv", row.names = FALSE) write.csv(res_listings,
"res_listings.csv", row.names = FALSE)
#write.csv(reviews, "reviews.csv") write.csv(seg_summary, "seg_summary.csv", row.names = FALSE)
```

Acknowledgments

I would like to acknowledge the following sources and individuals for their assistance and contributions to this project:

OpenAI's ChatGPT: For providing guidance and assistance in understanding R, RShiny, and data preparation techniques.

Stack Overflow: For offering valuable solutions to specific coding challenges encountered during the project.

InsideAirbnb.com: For providing the dataset used in the analysis.

Airbnb Database Rstudio: For serving as an inspiration and reference for the development of the RShiny app.