# Airbnb EDA for Paris.*

Jerry Lu

March 5, 2024

The dataset comes from Inside Airbnb (Cox 2021), and we'll read it from their website before saving a copy locally. We can send read_csv() a link to the dataset, and it will download it.

```
url <-
  paste0(

    "http://data.insideairbnb.com/france/ile-de-france/",
    "paris/2023-12-12/data/listings.csv.gz"

    )

airbnb_data <-
  read_csv(
    file = url,
    guess_max = 20000
  )
```

```
Rows: 74329 Columns: 75
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (24): listing_url, source, name, neighborhood_overview, picture_url, ho...
dbl  (37): id, scrape_id, host_id, host_listings_count, host_total_listings_...
lgl   (9): description, host_is_superhost, host_has_profile_pic, host_identi...
date  (5): last_scraped, host_since, calendar_last_scraped, first_review, la...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

*Code and data are available at: https://github.com/Jerryluuuu/TUT8.git

```
write_csv(airbnb_data, "airbnb_data.csv")

airbnb_data
```

```
# A tibble: 74,329 x 75
      id listing_url             scrape_id last_scraped source name  description
   <dbl> <chr>                       <dbl> <date>       <chr>  <chr> <lgl>
 1  3109 https://www.airbnb.com~   2.02e13 2023-12-12   city ~ Rent~ NA
 2  5396 https://www.airbnb.com~   2.02e13 2023-12-14   city ~ Rent~ NA
 3 81106 https://www.airbnb.com~   2.02e13 2023-12-13   city ~ Rent~ NA
 4  7397 https://www.airbnb.com~   2.02e13 2023-12-13   city ~ Rent~ NA
 5  7964 https://www.airbnb.com~   2.02e13 2023-12-12   city ~ Rent~ NA
 6 81615 https://www.airbnb.com~   2.02e13 2023-12-13   city ~ Rent~ NA
 7  9359 https://www.airbnb.com~   2.02e13 2023-12-13   city ~ Rent~ NA
 8 81870 https://www.airbnb.com~   2.02e13 2023-12-13   city ~ Rent~ NA
 9  9952 https://www.airbnb.com~   2.02e13 2023-12-14   city ~ Rent~ NA
10 86053 https://www.airbnb.com~   2.02e13 2023-12-13   city ~ Rent~ NA
# i 74,319 more rows
# i 68 more variables: neighborhood_overview <chr>, picture_url <chr>,
#   host_id <dbl>, host_url <chr>, host_name <chr>, host_since <date>,
#   host_location <chr>, host_about <chr>, host_response_time <chr>,
#   host_response_rate <chr>, host_acceptance_rate <chr>,
#   host_is_superhost <lgl>, host_thumbnail_url <chr>, host_picture_url <chr>,
#   host_neighbourhood <chr>, host_listings_count <dbl>, ...
```

Extract variable that interested in the project from dataset. Including: host_id, host_response_time, host_is_superhost, host_total_listings_count, neighbourhood_cleansed, bathrooms, bedrooms, price, number_of_reviews, review_scores_rating, review_scores_accuracy, review_scores_value

```
airbnb_data_selected <-
  airbnb_data |>
  select(
    host_id,
    host_response_time,
    host_is_superhost,
    host_total_listings_count,
    neighbourhood_cleansed,
    bathrooms,
    bedrooms,
    price,
    number_of_reviews,
    review_scores_rating,
    review_scores_accuracy,
    review_scores_value
  )

write_parquet(
  x = airbnb_data_selected,
  sink =
    "2023-12-12-paris-airbnblistings-select_variables.parquet"
  )

rm(airbnb_data)
```

Price might be the first thing we want to know. Right now it's a character, so we need to change it to a number. There are a lot of these, so we need to be careful that they don't all just turn into NAs. It will go to "NA" if we only force the price variable to be a number, since there are many characters that don't have a clear number equivalent, like "$." To begin, we need to get rid of those characters.

```
airbnb_data_selected$price |>
  head()
```

```
[1] "$150.00" "$146.00" "$110.00" "$140.00" "$180.00" "$71.00"
```

```
airbnb_data_selected$price |>
  str_split("") |>
  unlist() |>
  unique()
```

```
 [1] "$" "1" "5" "0" "." "4" "6" "8" "7" "3" "2" "9" NA  ","
```

```
airbnb_data_selected |>
  select(price) |>
  filter(str_detect(price, ","))
```

```
# A tibble: 1,550 x 1
   price
   <chr>
 1 $1,200.00
 2 $8,000.00
 3 $7,000.00
 4 $1,997.00
 5 $1,000.00
 6 $1,286.00
 7 $2,300.00
 8 $1,500.00
 9 $1,200.00
10 $1,357.00
# i 1,540 more rows
```

### 0.1 Distribution and properties of individual variables

```
airbnb_data_selected <-
  airbnb_data_selected |>
  mutate(
    price = str_remove_all(price, "[\\$,]"),
    price = as.integer(price)
  )
```

We'll get rid of all prices above $999 for now. Superhosts are Airbnb hosts with a lot of knowledge, and we might want to find out more about them. We wouldn't expect any NAs because a host is either a superhost or it's not. We can see that there are NAs, though. It's possible that the host took down a post or something similar, but we need to find out more about this. We will also need to turn this into a binary variable. The current value is true/false, which works fine for modelling. However, there are a few times when a 0/1 would be more useful. And for now, we'll just get rid of anyone who gave us a NA for "superhost." It's harder to deal with the NAs in "review_scores_rating" because there are so many of them. This could be because they don't have any reviews.

```
airbnb_data_less_1000 <-
  airbnb_data_selected |>
  filter(price < 1000)

airbnb_data_no_superhost_nas <-
  airbnb_data_less_1000 |>
  filter(!is.na(host_is_superhost)) |>
  mutate(
    host_is_superhost_binary =
      as.numeric(host_is_superhost)
  )
```

```
airbnb_data_has_reviews <-
  airbnb_data_no_superhost_nas |>
  filter(!is.na(review_scores_rating))
```

Guests can rate an Airbnb listing with one to five stars based on many factors, such as cleaning, accuracy, value, and more. However, when we look at the reviews in our dataset, it's clear that it's really a binary, with ratings being either five stars or not so much.

```
airbnb_data_no_superhost_nas |>
  ggplot(aes(x = review_scores_rating)) +
  geom_bar() +
  theme_classic() +
  labs(
    x = "Review scores rating",
    y = "Number of properties"
  )
```

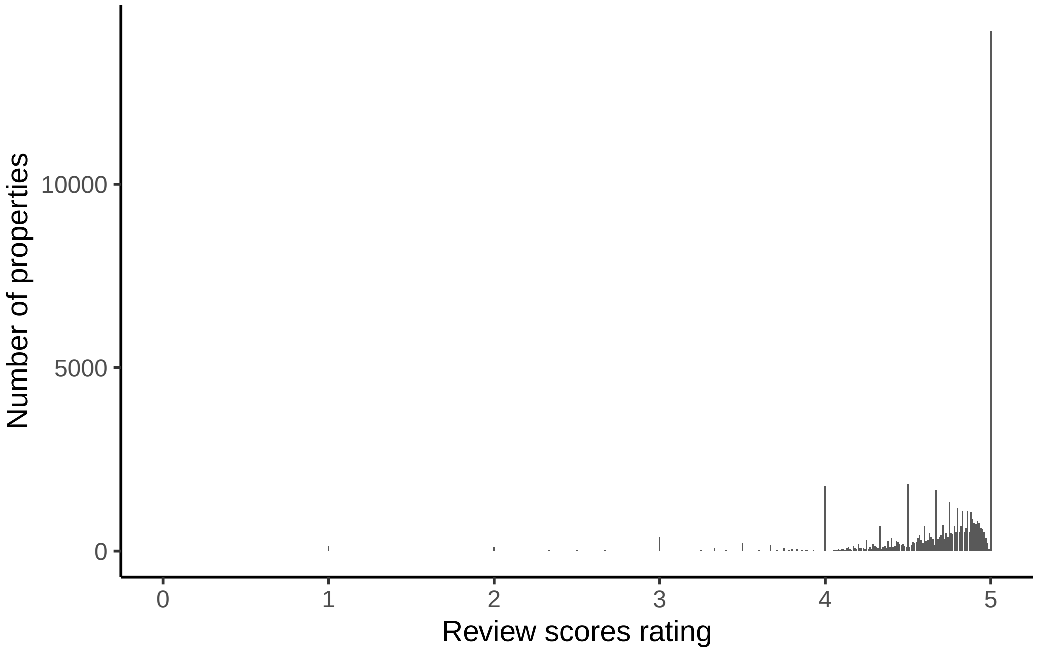

Figure 1: Distribution of review scores rating for Paris Airbnb rentals

Another key consideration is how quickly a host replies to a request. Airbnb gives hosts up to 24 hours to react, but prefers responses within an hour. It is not evident how a host might have a response time of NA. Perhaps this is related to another variable. Interestingly, what appear to be "NAs" in the "host_response_time" variable are not classified as proper NAs, but are instead considered as a different category. We'll recode them as actual NAs and modify the variable to a factor. The abundance of NAs creates a problem. For example, we could want to examine if there is a correlation with the review score. There are numerous reviews with an overall rating of 100.

```
airbnb_data_has_reviews |>
  count(host_response_time)
```

```
# A tibble: 6 x 2
  host_response_time     n
  <chr>              <int>
1 N/A                16531
2 a few days or more  1243
3 within a day        5297
4 within a few hours  6811
5 within an hour     22094
6 <NA>                   2
```

```
airbnb_data_has_reviews <-
  airbnb_data_has_reviews |>
  mutate(
    host_response_time = if_else(
      host_response_time == "N/A",
      NA_character_,
      host_response_time
    ),
    host_response_time = factor(host_response_time)
  )
airbnb_data_has_reviews |>
  filter(is.na(host_response_time)) |>
  ggplot(aes(x = review_scores_rating)) +
  geom_histogram(binwidth = 1) +
  theme_classic() +
  labs(
    x = "Average review score",
    y = "Number of properties"
  )
```

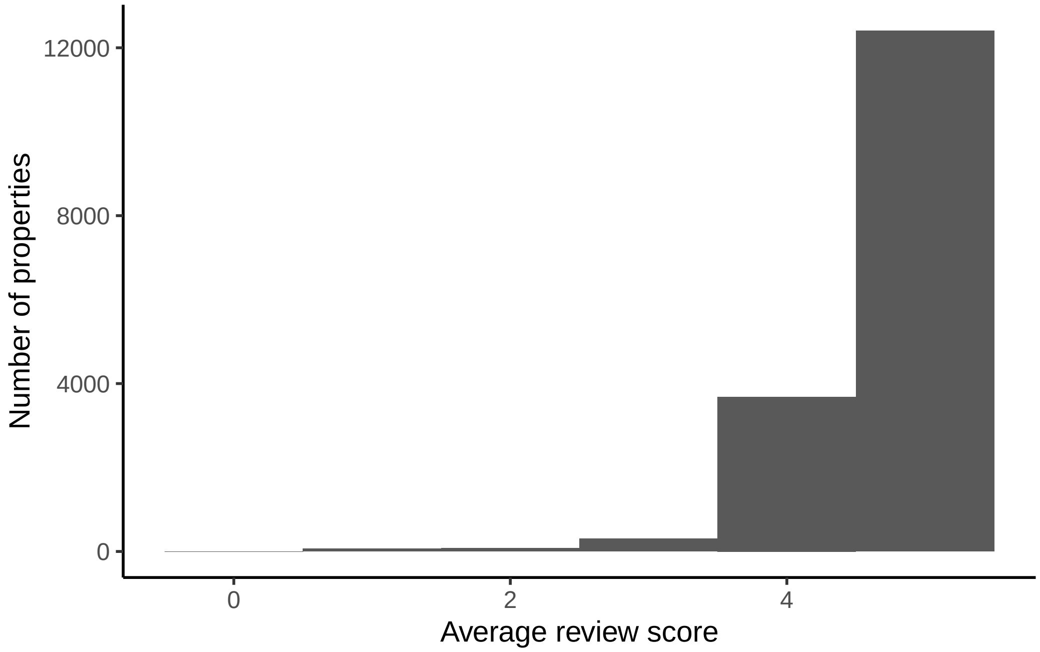

Figure 2: Distribution of review scores for properties with NA response time, for Paris Airbnb rentals

Usually missing values are dropped by `ggplot2`. We can use `geom_miss_point()` from `naniar` to include them in the graph

```r
airbnb_data_has_reviews |>
  ggplot(aes(
    x = host_response_time,
    y = review_scores_accuracy
  )) +
  geom_miss_point() +
  labs(
    x = "Host response time",
    y = "Review score accuracy",
    color = "Is missing?"
  ) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

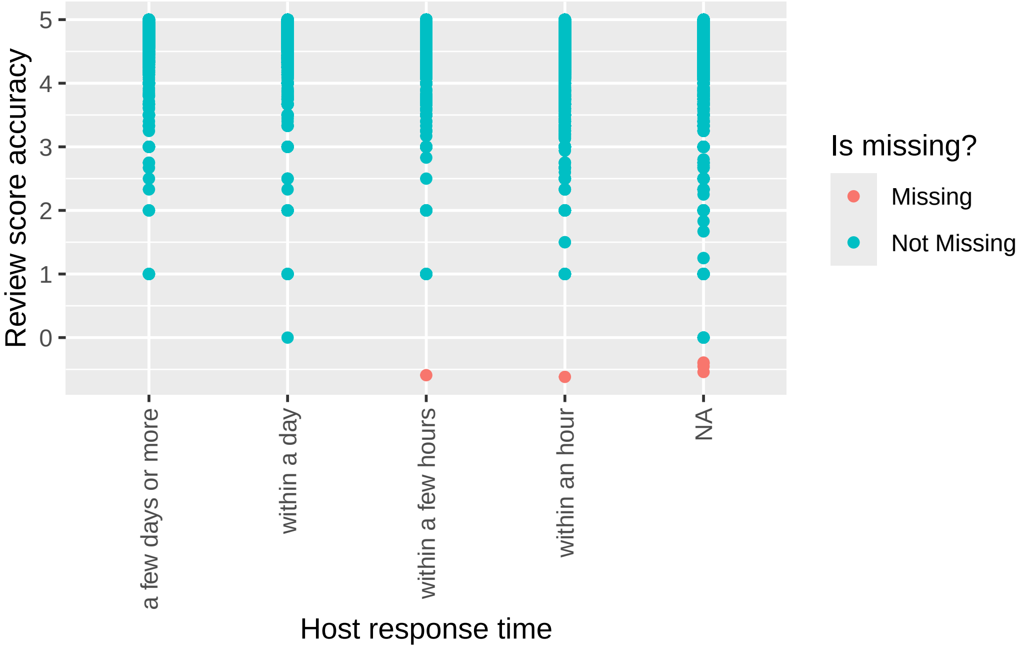

Figure 3: Missing values in Paris Airbnb data, by host response time

```r
airbnb_data_selected <-
  airbnb_data_has_reviews |>
  filter(!is.na(host_response_time))
```

We might be interested in how many properties a host has on Airbnb

```r
airbnb_data_selected |>
  ggplot(aes(x = host_total_listings_count)) +
  geom_histogram() +
  scale_x_log10() +
  labs(
    x = "Total number of listings, by host",
    y = "Number of hosts"
  )
```

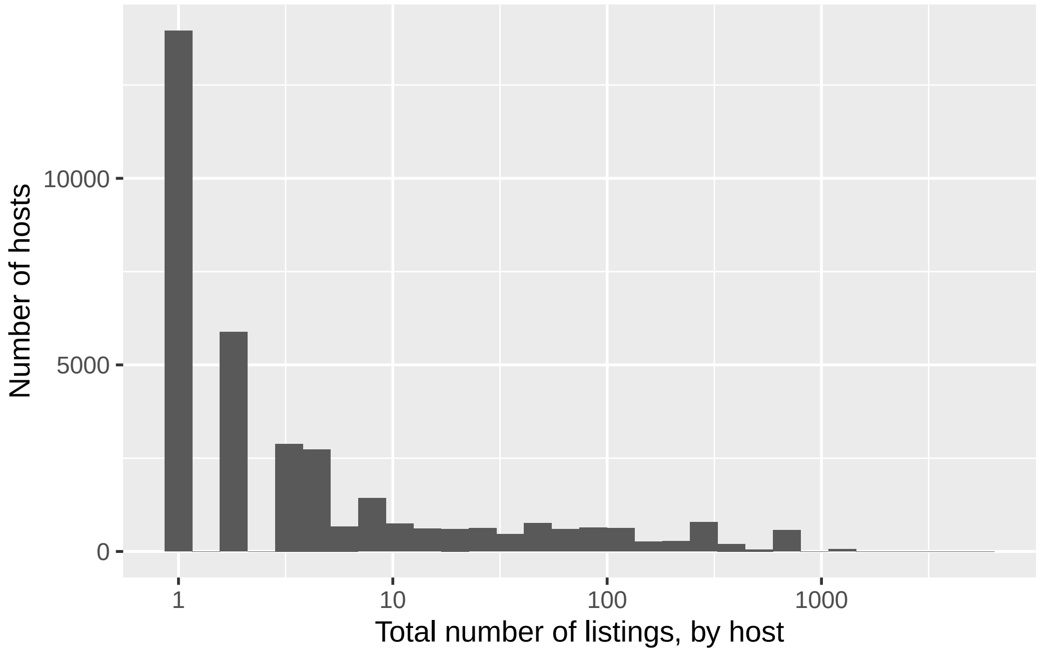

Figure 4: Distribution of the number of properties a host has on Airbnb, forParis Airbnb rentals

```
airbnb_data_selected |>
  filter(host_total_listings_count >= 500) |>
  head()
```

```
# A tibble: 6 x 13
   host_id host_response_time host_is_superhost host_total_listings_count
     <dbl> <fct>              <lgl>                                 <dbl>
1 50502817 within an hour     FALSE                                   778
2 50502817 within an hour     FALSE                                   778
3 50502817 within an hour     FALSE                                   778
4 50502817 within an hour     FALSE                                   778
5 50502817 within an hour     FALSE                                   778
6 50502817 within an hour     FALSE                                   778
# i 9 more variables: neighbourhood_cleansed <chr>, bathrooms <lgl>,
#   bedrooms <dbl>, price <int>, number_of_reviews <dbl>,
#   review_scores_rating <dbl>, review_scores_accuracy <dbl>,
#   review_scores_value <dbl>, host_is_superhost_binary <dbl>
```

```
airbnb_data_selected <-
  airbnb_data_selected |>
  add_count(host_id) |>
  filter(n == 1) |>
  select(-n)
```

We might want to create some graphs to see if any links between variables become apparent. Some things that come to mind include comparing costs to ratings, superhosts, the amount of properties, and the neighbourhood. For properties with multiple reviews, we may analyse the relationship between pricing and reviews, as well as if they are a super-host.

```
airbnb_data_selected |>
  filter(number_of_reviews > 1) |>
  ggplot(aes(x = price, y = review_scores_rating,
             color = host_is_superhost)) +
  geom_point(size = 1, alpha = 0.1) +
  theme_classic() +
  labs(
    x = "Price per night",
    y = "Average review score",
    color = "Superhost"
  ) +
  scale_color_brewer(palette = "Set1")
```

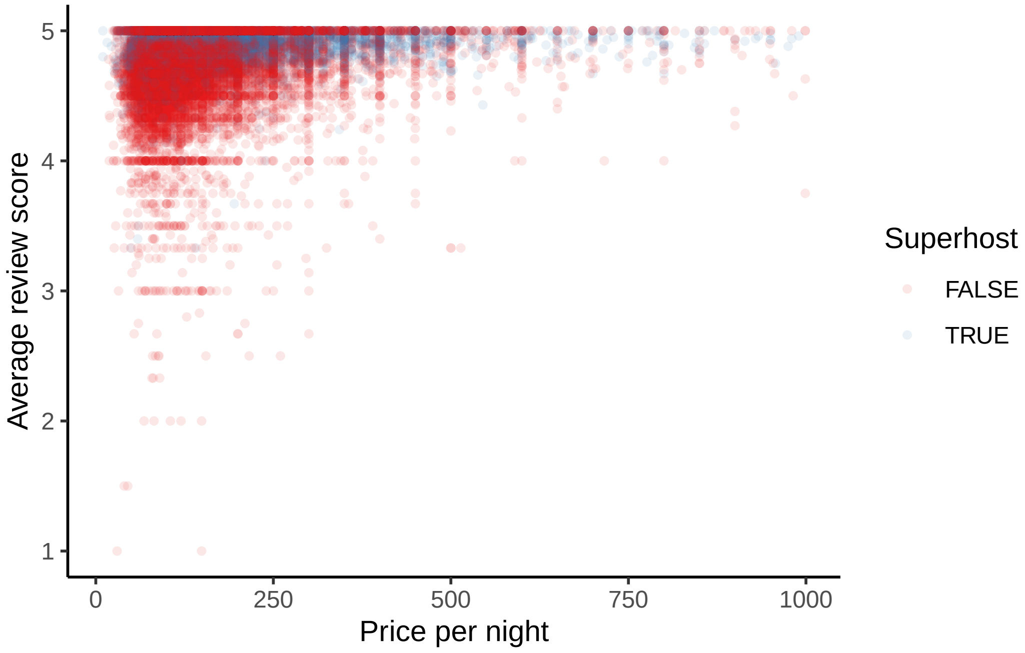

Figure 5: Relationship between price and review and whether a host is a superhost, for Paris Airbnb rentals

One of the characteristics that may distinguish a superhost is how rapidly they answer to requests. One may believe that being a superhost entails fast responding yes or no to requests. Let's look at the data. First, we'll look at the possible values of superhost based on their response times. Fortunately, it appears that when we removed the review rows, we also removed any NAs from whether they were a superhost, but if we go back and investigate, we may need to double-check. Using the `tabyl()` function from `janitor`, we might create a table that compares a host's response time to whether they are a superhost. It is evident that if a host does not answer within an hour, they are unlikely to be a Superhost. Finally, we could look at the neighbourhood. The data provider has attempted to clean the neighbourhood variable for us, thus we will continue to use it for now. However, if we were to use this variable in our actual research, we would want to investigate how it was produced.

```
airbnb_data_selected |>
  count(host_is_superhost) |>
  mutate(
    proportion = n / sum(n),
    proportion = round(proportion, digits = 2)
  )
```

```
# A tibble: 2 x 3
```

```
  host_is_superhost     n proportion
  <lgl>             <int>      <dbl>
1 FALSE             15820       0.72
2 TRUE               6227       0.28
```

```
airbnb_data_selected |>
  tabyl(host_response_time, host_is_superhost) |>
  adorn_percentages("col") |>
  adorn_pct_formatting(digits = 0) |>
  adorn_ns() |>
  adorn_title()
```

```
                    host_is_superhost
 host_response_time             FALSE         TRUE
 a few days or more         6%   (953)  0%     (24)
       within a day        22% (3,511) 12%    (770)
 within a few hours        24% (3,802) 26% (1,614)
     within an hour        48% (7,554) 61% (3,819)
```

```
airbnb_data_selected |>
  tabyl(neighbourhood_cleansed) |>
  adorn_pct_formatting() |>
  arrange(-n) |>
  filter(n > 100) |>
  adorn_totals("row") |>
  head()
```

```
 neighbourhood_cleansed    n percent
      Buttes-Montmartre 2842   12.9%
             Popincourt 2202   10.0%
               Entrepôt 1713    7.8%
              Vaugirard 1681    7.6%
            Ménilmontant 1438    6.5%
        Buttes-Chaumont 1430    6.5%
```

During EDA, we can use models to help us understand how different factors in a dataset may be related to each other. As an example, we might want to see if we can guess if someone is a superhost and what factors could explain that. Because the result is either yes or no, this is a good time to use logistic regression. We think that being a superhost will lead to faster replies and better reviews.

Table 1: Explaining whether a host is a superhost based on their response time

| | (1) |
|---|---|
| (Intercept) | −16.262 |
| | (0.481) |
| host_response_timewithin a day | 2.019 |
| | (0.211) |
| host_response_timewithin a few hours | 2.695 |
| | (0.210) |
| host_response_timewithin an hour | 2.972 |
| | (0.209) |
| review_scores_rating | 2.624 |
| | (0.089) |
| Num.Obs. | 22 047 |
| AIC | 24 165.0 |
| BIC | 24 205.0 |
| Log.Lik. | −12 077.507 |
| RMSE | 0.43 |

```r
logistic_reg_superhost_response_review <-
  glm(
    host_is_superhost ~
      host_response_time +
      review_scores_rating,
    data = airbnb_data_selected,
    family = binomial
  )
```

```r
modelsummary(logistic_reg_superhost_response_review)
```

```r
write_parquet(
  x = airbnb_data_selected,
  sink = "2023-12-12-paris-airbnblistings-analysis_dataset.parquet"
  )
```

# 1 Reference