Final Project- K-Nearest Neighbors & Naive Bayes

Step III: Classification

Part I: K-nearest neighbors

A. Using k-nearest neighbors, predict whether a rental in your neighborhood will have some particular amenity, or combination of amenities. Use any set of numerical predictors in order to build this model. You can decide which amenity, or set of amenities, to use as your outcome variable.(Hint: the grepl() function is worth exploring in order to perform this step). Show the code you used to run your model, and the code you used to assess your model.

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.0     ✔ readr     2.1.4
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ ggplot2   3.4.1     ✔ tibble    3.2.1
## ✔ lubridate 1.9.2     ✔ tidyr     1.3.0
## ✔ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflic
ts to become errors
```

```
hk2<- read_csv("hong_kong_cleaned.csv")
```

```
## New names:
## Rows: 1424 Columns: 50
## ── Column specification
## ──────────────────────────────────────────────────────── Delimiter: "," chr
## (10): description, neighborhood_overview, host_location, host_response_... dbl
## (34): ...1, host_id, host_response_rate, host_acceptance_rate, host_lis... lgl
## (5): host_is_superhost, host_has_profile_pic, host_identity_verified, ... date
## (1): host_since
## 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.
## • `` -> `...1`
```

```
library(dplyr)
library(caret)
```

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

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

Creating the binary variables for input variables:

```
hk3 <- hk2 %>%
  mutate(has_washer = grepl("Washer", amenities, ignore.case = TRUE))
```

Data partition:

```
library(e1071)
set.seed(42)
train_size <- floor(0.6 * nrow(hk3))
train_indices <- sample(nrow(hk3), train_size)
train.df <- hk3[train_indices, c("minimum_nights", "maximum_nights", "price", "has_wa
sher", "host_listings_count")]
valid.df <- hk3[-train_indices, c("minimum_nights", "maximum_nights", "bedrooms", "pr
ice", "has_washer", "host_listings_count")]
```

I convert the logical outcome variables to factors:

```
train.df$has_washer <- factor(train.df$has_washer, levels = c(FALSE, TRUE), labels =
c(0, 1))

valid.df$has_washer <- factor(valid.df$has_washer, levels = c(FALSE, TRUE), labels =
c(0, 1))
```

I will normalize the data:

```
library(caret)

norm_values<-preProcess(train.df, method=c("center", "scale"))

train.norm <-predict(norm_values,train.df)
valid.norm<-predict(norm_values,valid.df)

print(train.norm)
```

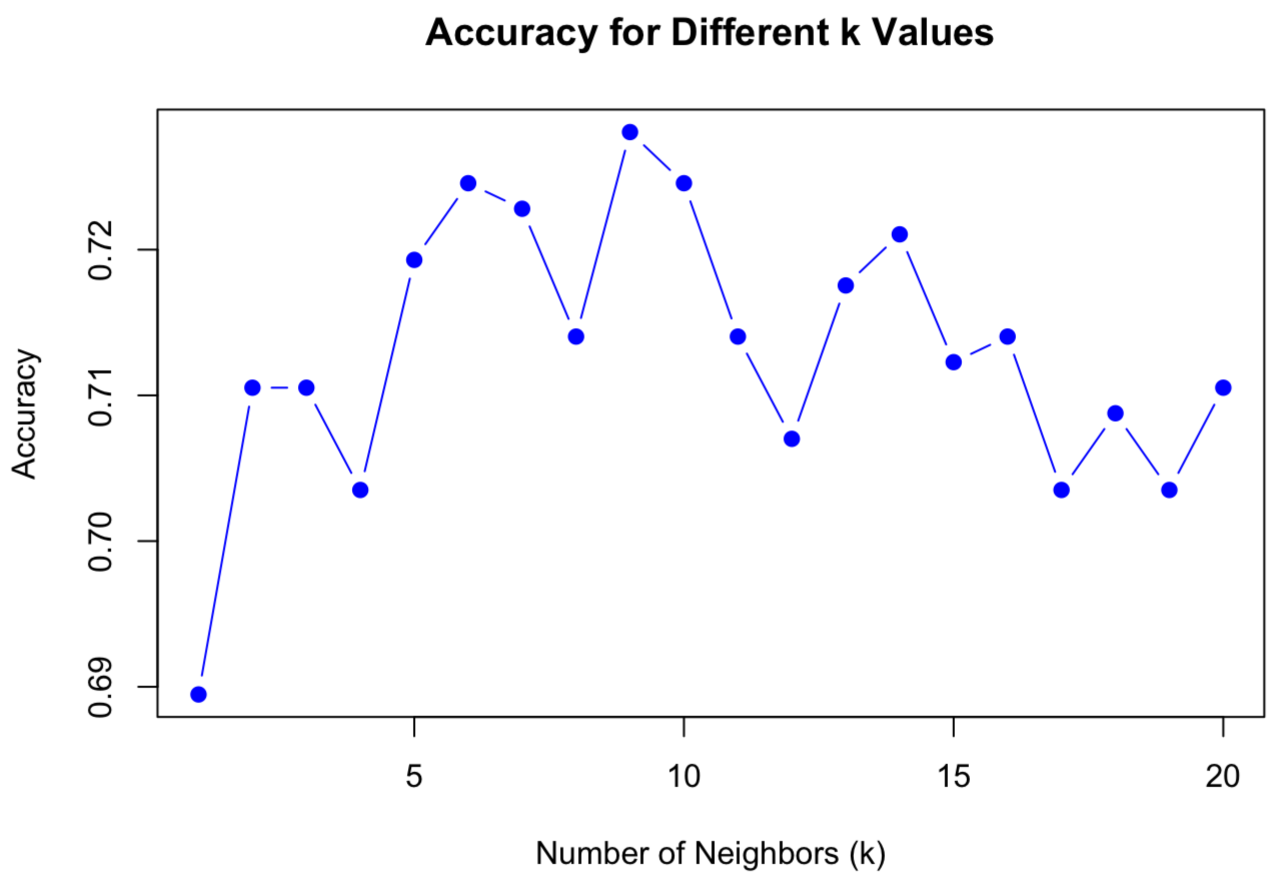
```
## # A tibble: 854 × 5
##    minimum_nights maximum_nights   price has_washer host_listings_count
##             <dbl>          <dbl>   <dbl> <fct>                    <dbl>
##  1         0.0691          0.714 -0.0864 0                         1.95
##  2        -0.448           0.714  0.511  1                        -1.01
##  3         0.0906          0.714 -0.247  0                         0.914
##  4        -0.512          -1.75  -0.0684 1                        -0.967
##  5         1.40            0.714  0.204  1                        -0.983
##  6         0.0691          0.714  0.211  1                         1.95
##  7         0.0691         -1.09  -0.136  1                        -0.993
##  8         0.0691         -1.75  -0.193  0                        -0.845
##  9         1.40            0.714  0.233  1                        -0.983
## 10         0.112           0.714 -0.236  1                         1.32
## # … with 844 more rows
```

```
print(valid.norm)
```

```
## # A tibble: 570 × 6
##    minimum_nights maximum_nights bedrooms   price has_washer host_listings_count
##             <dbl>          <dbl>    <dbl>   <dbl> <fct>                    <dbl>
##  1         0.0906          0.714        1 -0.247  1                        0.914
##  2         0.112          -1.09         1  0.293  1                       -0.972
##  3         0.112          -0.297        1 -0.236  1                        1.32
##  4         0.112           0.714        1 -0.236  1                        1.32
##  5         0.0906          0.714        1 -0.247  0                        0.591
##  6        -0.319           0.714        3  0.811  1                       -1.01
##  7         1.40           -1.09         1 -0.0934 1                       -0.977
##  8         0.112           0.714        1 -0.247  1                        1.32
##  9         0.0906          0.714        1 -0.253  0                        0.914
## 10         0.0906          0.714        1 -0.198  0                        0.591
## # … with 560 more rows
```

I determine the optimal value for k:

```
library(class)

predictors <- c("minimum_nights", "maximum_nights", "host_listings_count","price")
target_variable <- "has_washer"

train_matrix <- data.matrix(train.norm[, predictors])
valid_matrix <- data.matrix(valid.norm[, predictors])
train_labels <- train.df[[target_variable]]
valid_labels <- valid.df[[target_variable]]

k_values <- seq(1, 20, by = 1)

accuracy_values <- numeric(length(k_values))

for (i in seq_along(k_values)) {
  knn_model <- knn(train_matrix, valid_matrix, train_labels, k = k_values[i])
  accuracy_values[i] <- mean(knn_model == valid_labels)
}

plot(k_values, accuracy_values, type = "b", pch = 19, col = "blue",
     xlab = "Number of Neighbors (k)", ylab = "Accuracy",
     main = "Accuracy for Different k Values")
```



```
optimal_k <- k_values[which.max(accuracy_values)]
cat("Optimal k:", optimal_k, "\n")
```

```
## Optimal k: 9
```

The optimal k value is 9, so I will create the model with this k-value:

```
optimal_k <- 9
knn_model <- knn(train_matrix, valid_matrix, train_labels, k = optimal_k)
```

I Assess the model:

```
accuracy <- mean(knn_model == valid_labels)
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.7245614
```

```
confusion <- table(Actual = valid_labels, Predicted = knn_model)
print(confusion)
```

```
##       Predicted
## Actual   0   1
##      0 171  75
##      1  82 242
```

The Naive rule. Our model is 23.6% more accurate than if we used the Naive rule.

```
mean(hk3$has_washer)
```

```
## [1] 0.5533708
```

B. Write a two-paragraph narrative that describes how you did this. In your narrative, be sure to describe your predictor choices, and mention how you arrived at the particular k value that you used.

The amenity we decided to predict was “washer”. The numeric variables we decided to use for the model are: minimum_nights, “maximum_nights, price and”host_listings_count”. We figured these variables have predicting power because if the minimum night stay is low then probably the apartment won’t have a washer. However, if the maximum night stay is high, it is probably more likely that the apartment has a washer because the customers are staying more time and probably would need to do laundry. Also, the host_listings_count” is important because listings with high count maybe belong to a big company like a hotel or apart hotel that probably does not include washers in each unit but a shared one in the building. Likewise, a low count means the listings belong to independent hosts who rent their own apartment and probably have a washer inside.

After normalizing the data and converting the washer variable into a factor, we determined the optimal value of k by calculating the different accuracy values for different k-values. The resulting plot revealed the optimum k-value is 9. Therefore we run the model with k-9. Finally we assessed the accuracy of the model which turned out to be 0.7245614. The confusion matrix also revealed a bigger proportion of true positives and true negatives. Finally we compared the accuracy of the naïve rule with the accuracy of our model. We calculated the naïve rule by determining the mean of the number of apartments with washer. The naïve rule turned out to give a predictive accuracy of 0.5533708. This means our model is 23.6% more accurate than if we used the naïve rule.

Classification, Part II. Naive Bayes

A. Using any set of predictors, build a model using the naive Bayes algorithm, with the purpose of predicting host rental ratings. Use review_scores_rating as your response variable, after binning it using an equal frequency binning method. Do not use any of the other review_scores variables as model inputs.

I will convert amenities into variables

```
data<-hk2
```

```
data2 <- data %>%
  mutate(has_washer = grepl("Washer", amenities, ignore.case = TRUE),
         has_wifi = grepl("Wifi", amenities, ignore.case = TRUE),
         has_air_conditioning = grepl("Air conditioning", amenities, ignore.case = TR
UE),
         has_microwave = grepl("Microwave", amenities, ignore.case = TRUE),
         has_dishes_silverware = grepl("Dishes and silverware", amenities, ignore.cas
e = TRUE),
         has_fridge = grepl("Mini fridge", amenities, ignore.case = TRUE),
         has_hair_dryer = grepl("Hair dryer", amenities, ignore.case = TRUE))
```

First I will convert character variables into factors:

```
character <- c("host_response_time","host_verifications",
               "property_type","room_type", "bathroom_type",
               "amenities")


data2[character] <- lapply(data2[character],factor)
```

I will convert logical outcome variables into factors:

```
logical <- c("host_is_superhost","host_has_profile_pic","host_identity_verified", "ha
s_availability","instant_bookable", "has_wifi", "has_air_conditioning","has_microwav
e", "has_dishes_silverware","has_fridge","has_hair_dryer","has_washer")
data2[logical] <- lapply(data2[logical],factor)
```

The possible numeric variables are:

```
num_columns <- c("host_response_rate", "host_acceptance_rate", "host_listings_count",
"host_total_listings_count", "latitude", "longitude", "accommodates", "bathroom_nb",
"bedrooms", "beds", "price", "minimum_nights", "maximum_nights", "availability_30",
"availability_60", "availability_90", "availability_365", "number_of_reviews", "numbe
r_of_reviews_ltm", "number_of_reviews_l30d", "review_scores_rating", "calculated_host
_listings_count_entire_homes", "calculated_host_listings_count_private_rooms", "calcu
lated_host_listings_count_shared_rooms", "reviews_per_month", "got_reviewed")
```

I will bin numeric variables using equal frequency binning:

```
library(Hmisc)
```

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

```
## The following object is masked from 'package:e1071':
##
##     impute
```

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

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

```
for (col in num_columns){
data2[[col]] <- Hmisc::cut2(data2[[col]], m = nrow(data2)/5)
}
```

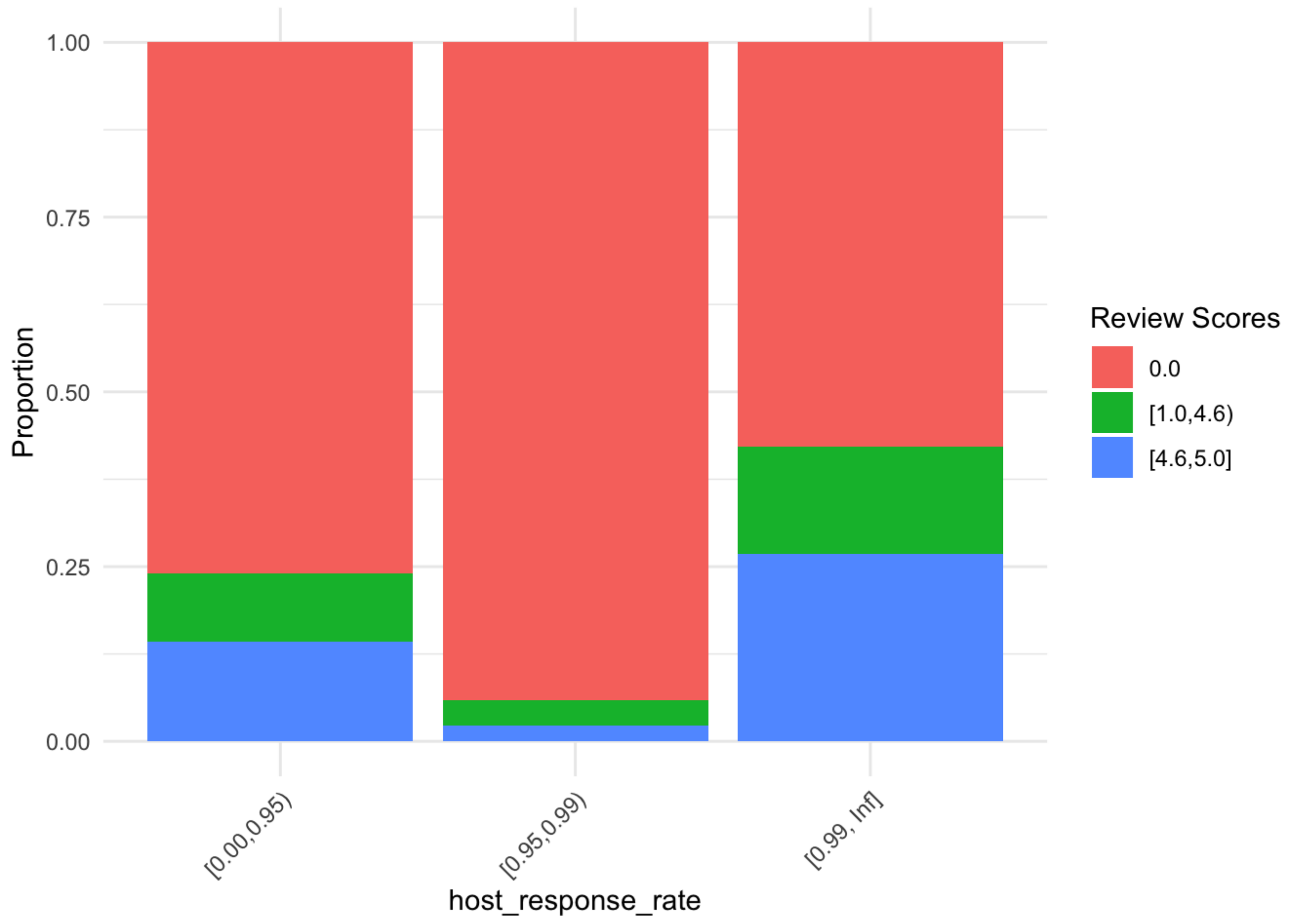
```
## Warning in min(xx[xx > upper]): no non-missing arguments to min; returning Inf

## Warning in min(xx[xx > upper]): no non-missing arguments to min; returning Inf

## Warning in min(xx[xx > upper]): no non-missing arguments to min; returning Inf
```

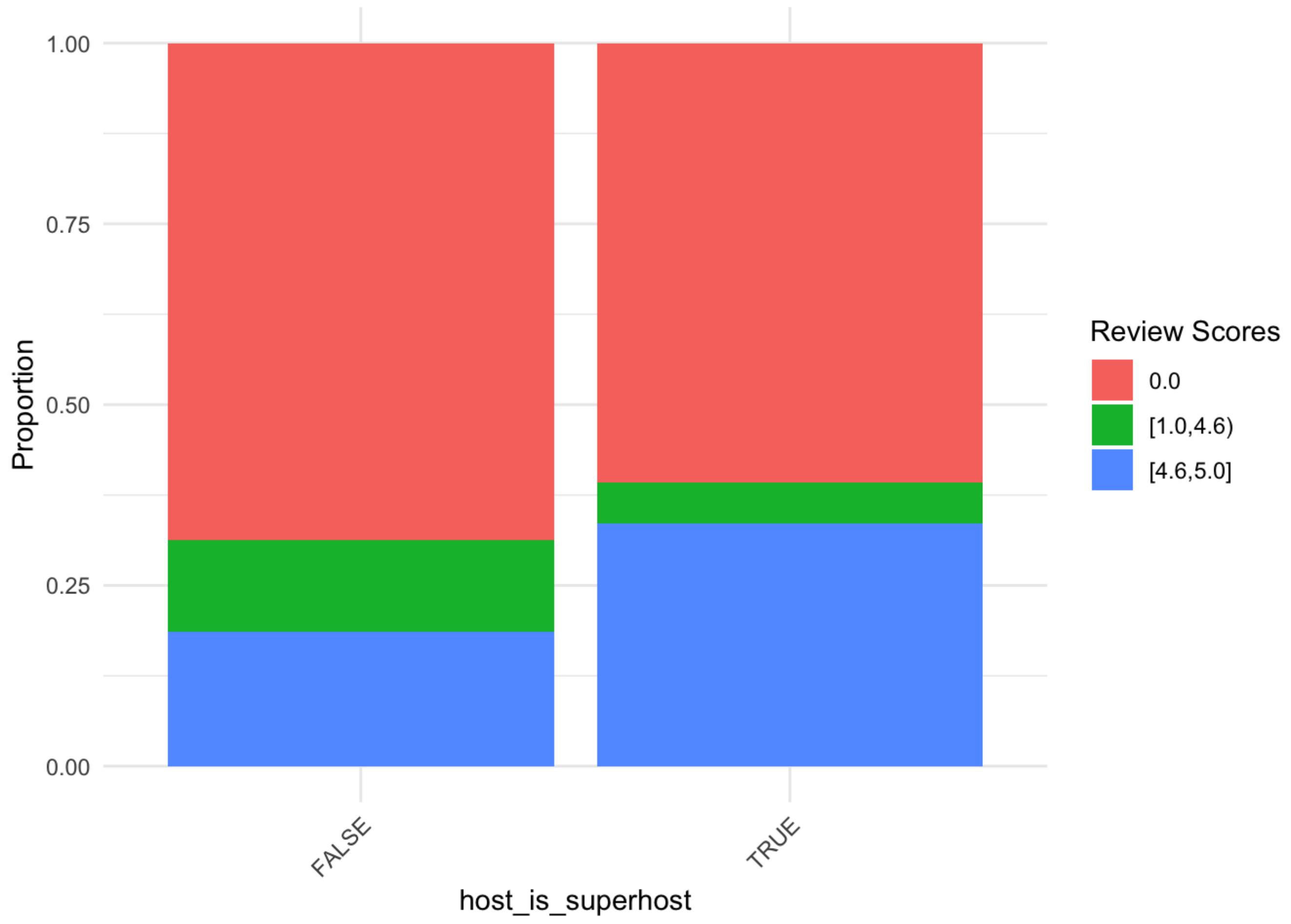
```
table(data2$host_response_rate)
```

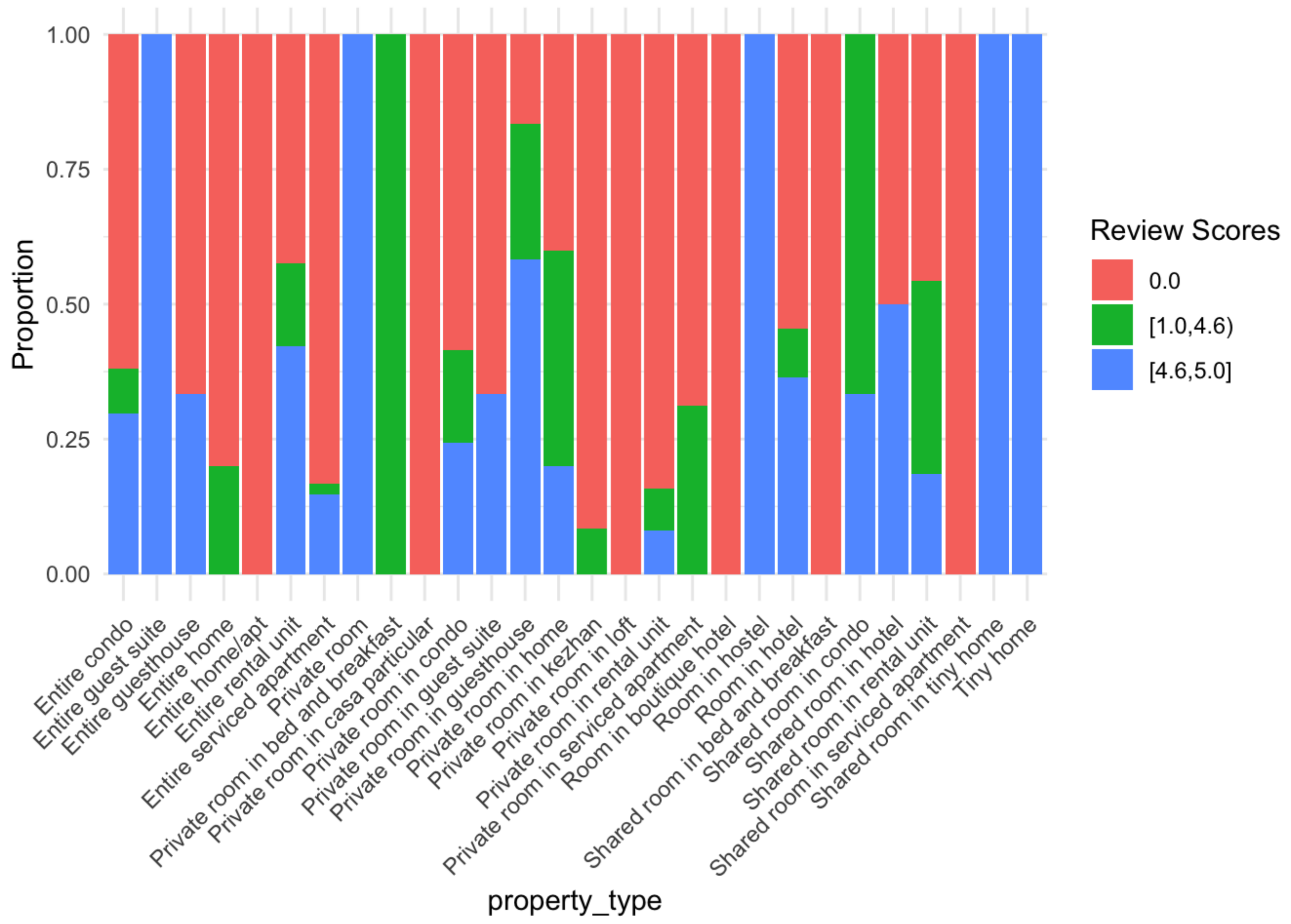
```
## 
## [0.00,0.95) [0.95,0.99) [0.99, Inf] 
##         357         219         848
```

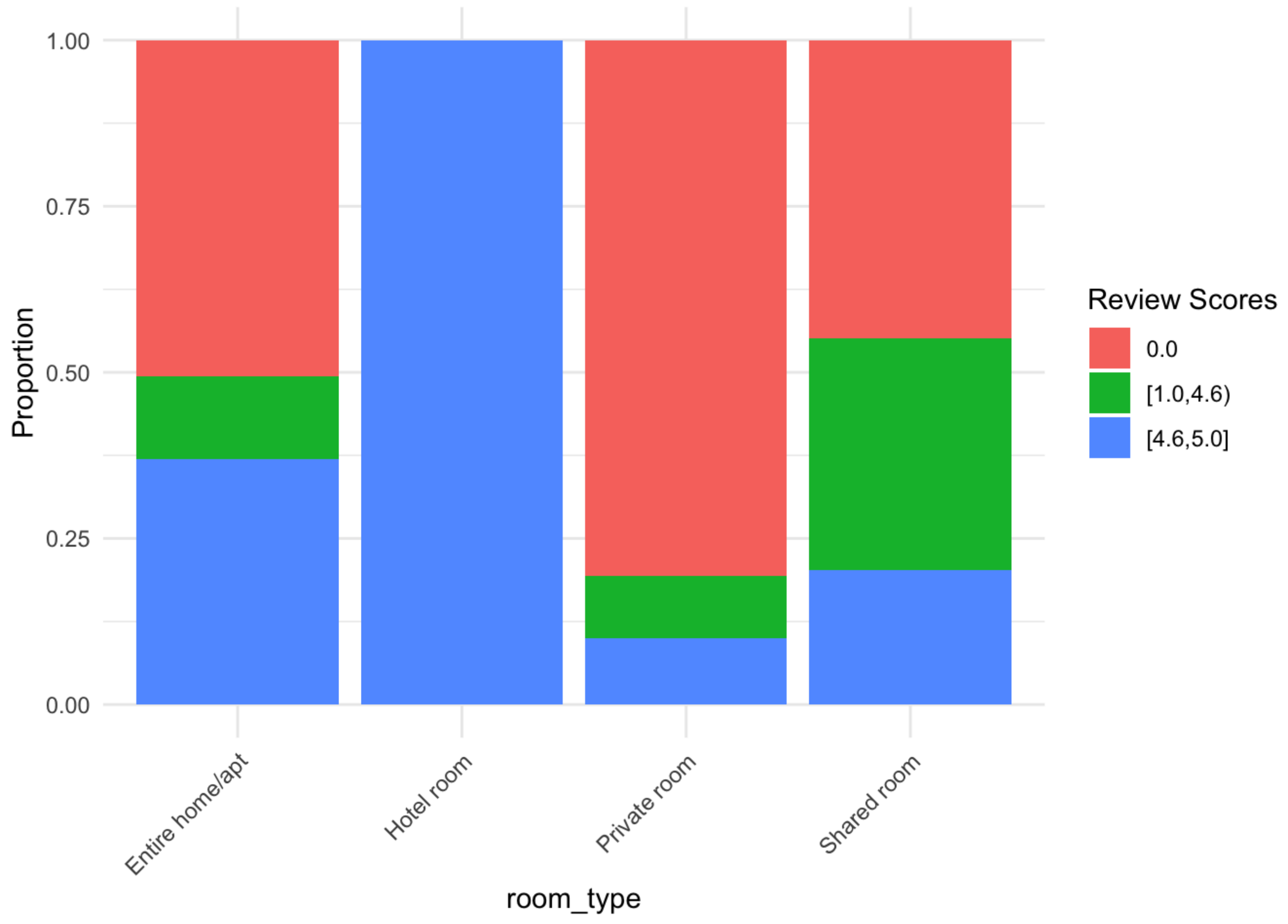
Now I assess their impact of the possible variables that might affect review_scores_rating:

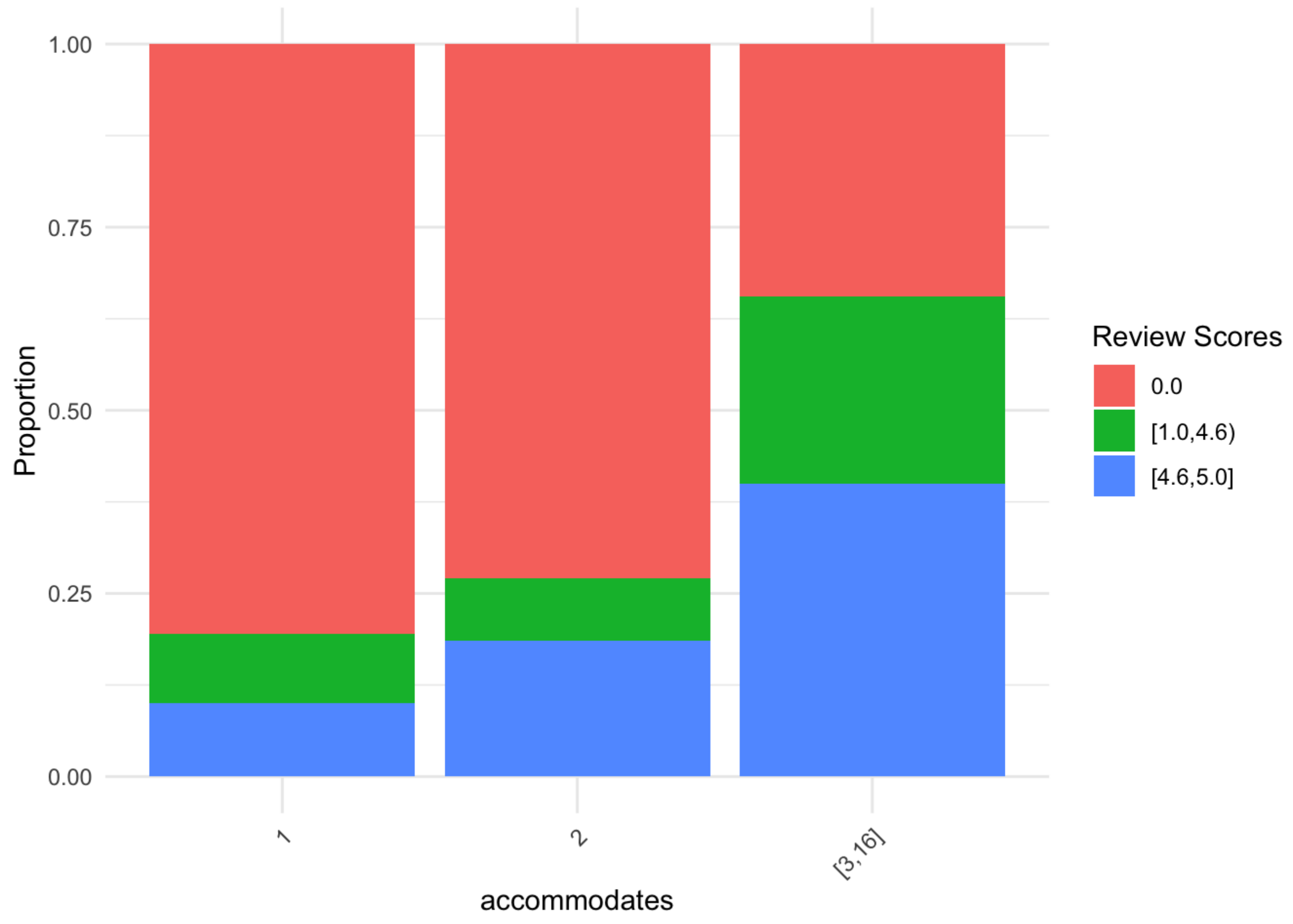
```
library(ggplot2)

create_proportional_barplot <- function(data2, x_var) {
  ggplot(data2, aes(x = !!sym(x_var), fill = review_scores_rating)) +
    geom_bar(position = "fill") +
    labs(x = x_var, y = "Proportion", fill = "Review Scores") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
}

prospective_input_variables <- c("host_response_rate", "host_is_superhost", "property
_type", "room_type", "accommodates", "bathroom_nb", "bedrooms", "beds", "price", "min
imum_nights", "maximum_nights", "availability_30", "availability_60", "availability_3
65","has_availability", "has_wifi", "has_air_conditioning","has_microwave", "has_dish
es_silverware","has_fridge","has_hair_dryer","has_washer" )

for (var in prospective_input_variables) {
  plot <- create_proportional_barplot(data2, var)
  print(plot)
}
```

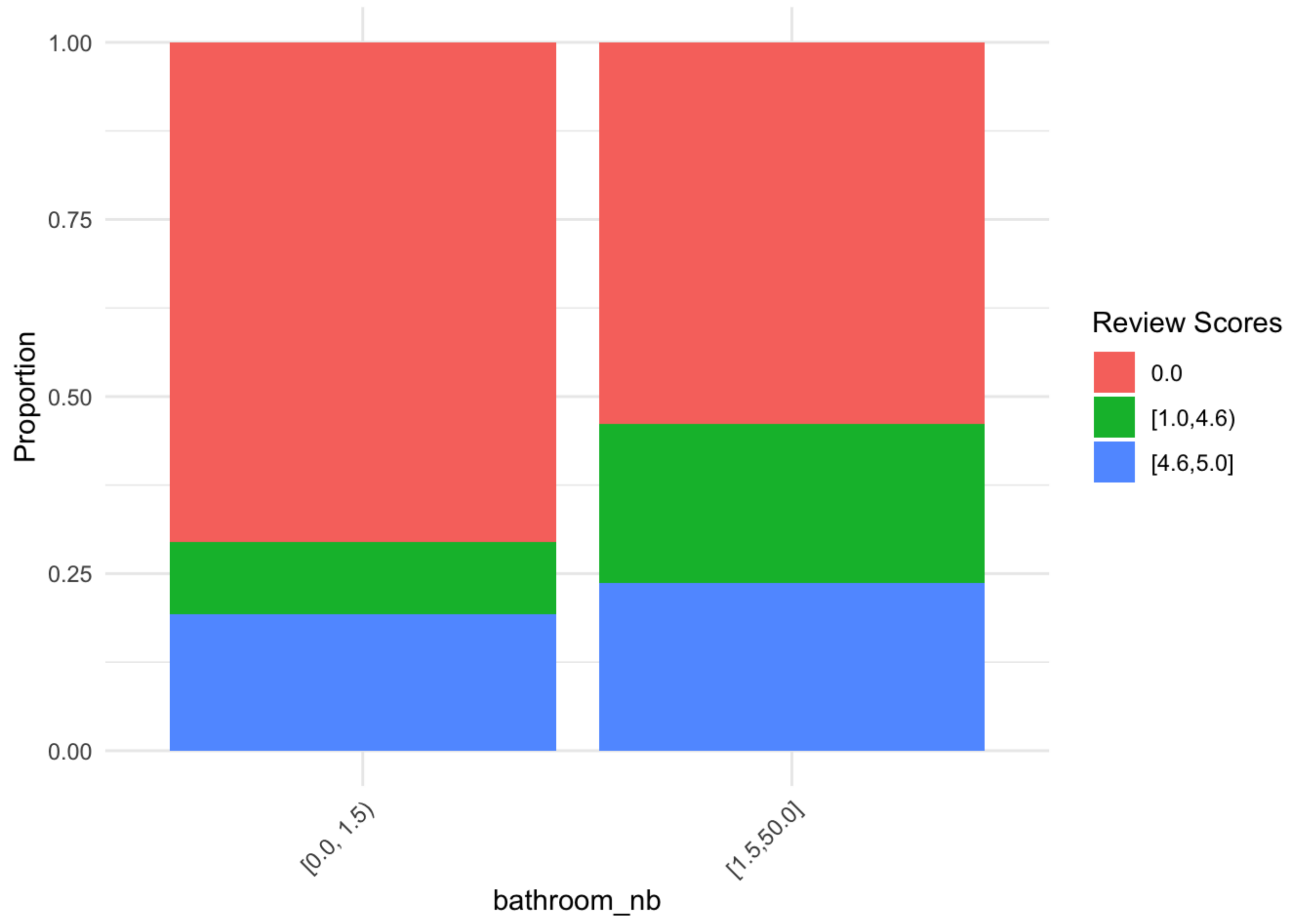

Proportion
1.00
0.75
0.50
0.25
0.00
[0.00,0.95)
[0.95,0.99)
[0.99, Inf]
host_response_rate
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

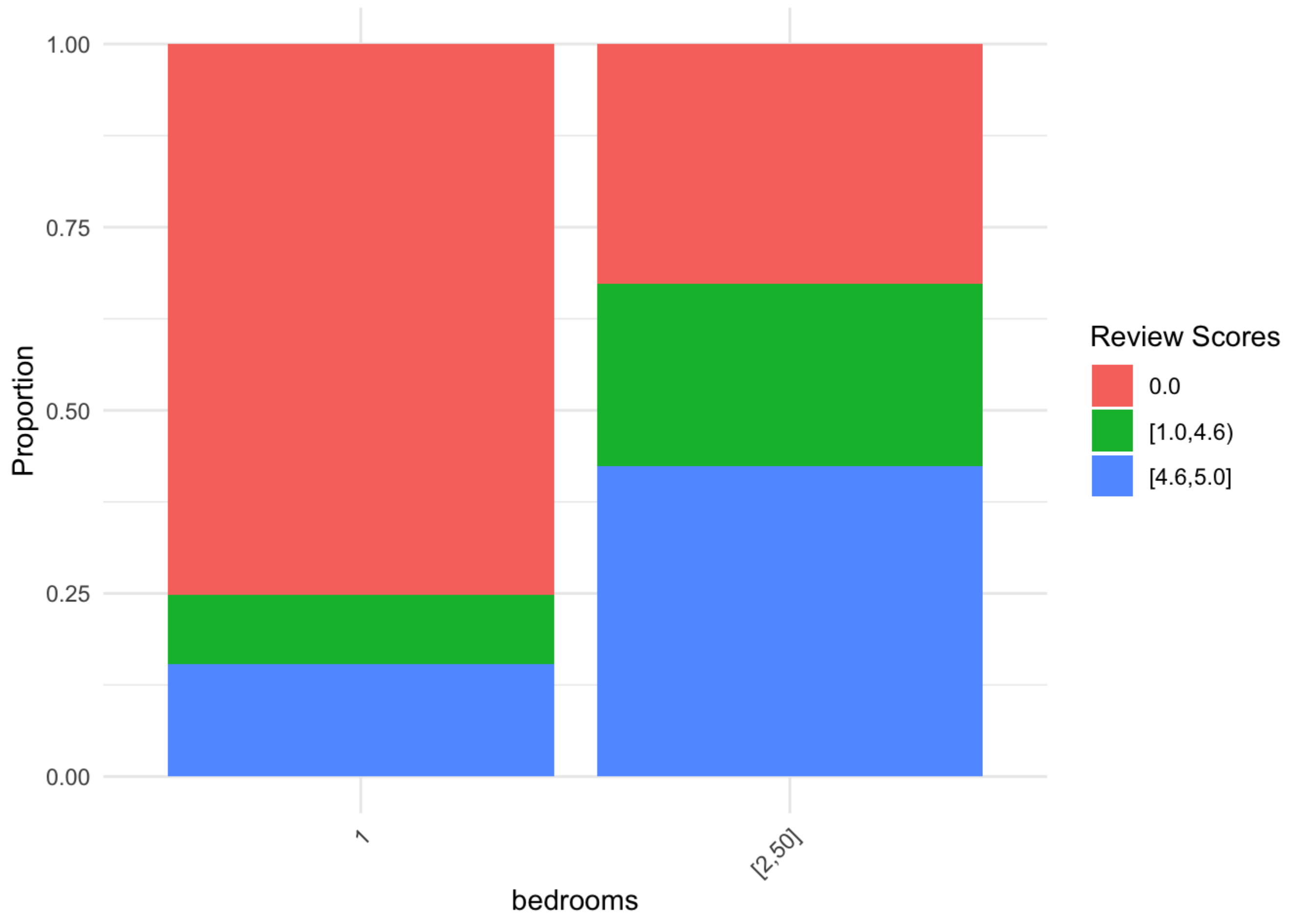

Proportion
1.00
0.75
0.50
0.25
0.00
FALSE
TRUE
host_is_superhost
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

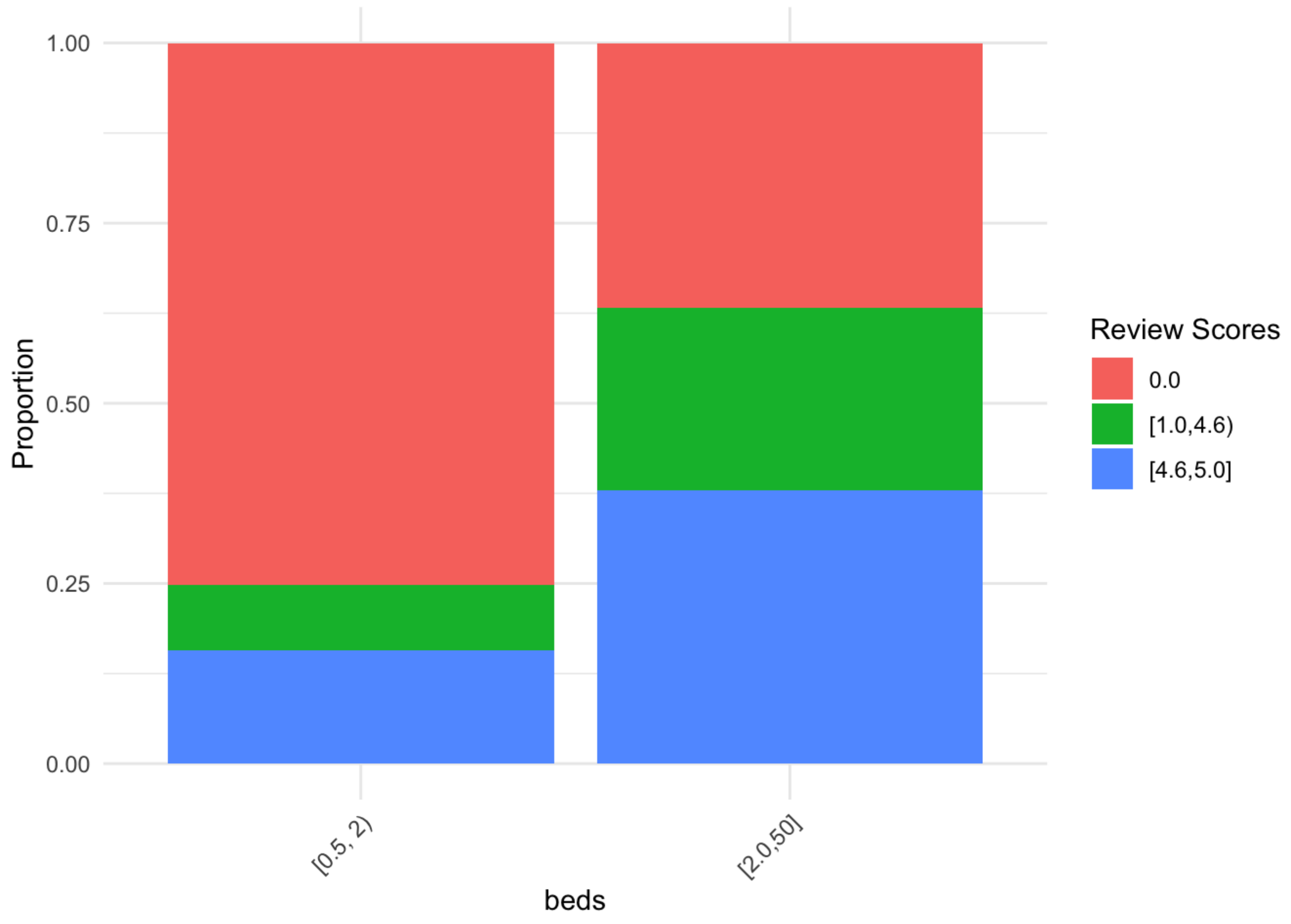

Proportion
1.00
0.75
0.50
0.25
0.00
Entire condo
Entire guest suite
Entire guesthouse
Entire home
Entire home/apt
Entire rental unit
Entire serviced apartment
Private room
Private room in bed and breakfast
Private room in casa particular
Private room in condo
Private room in guest suite
Private room in guesthouse
Private room in home
Private room in kezhan
Private room in loft
Private room in rental unit
Private room in serviced apartment
Room in boutique hotel
Room in hostel
Room in hotel
Shared room in bed and breakfast
Shared room in condo
Shared room in hotel
Shared room in rental unit
Shared room in serviced apartment
Shared room in tiny home
Tiny home
property_type
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

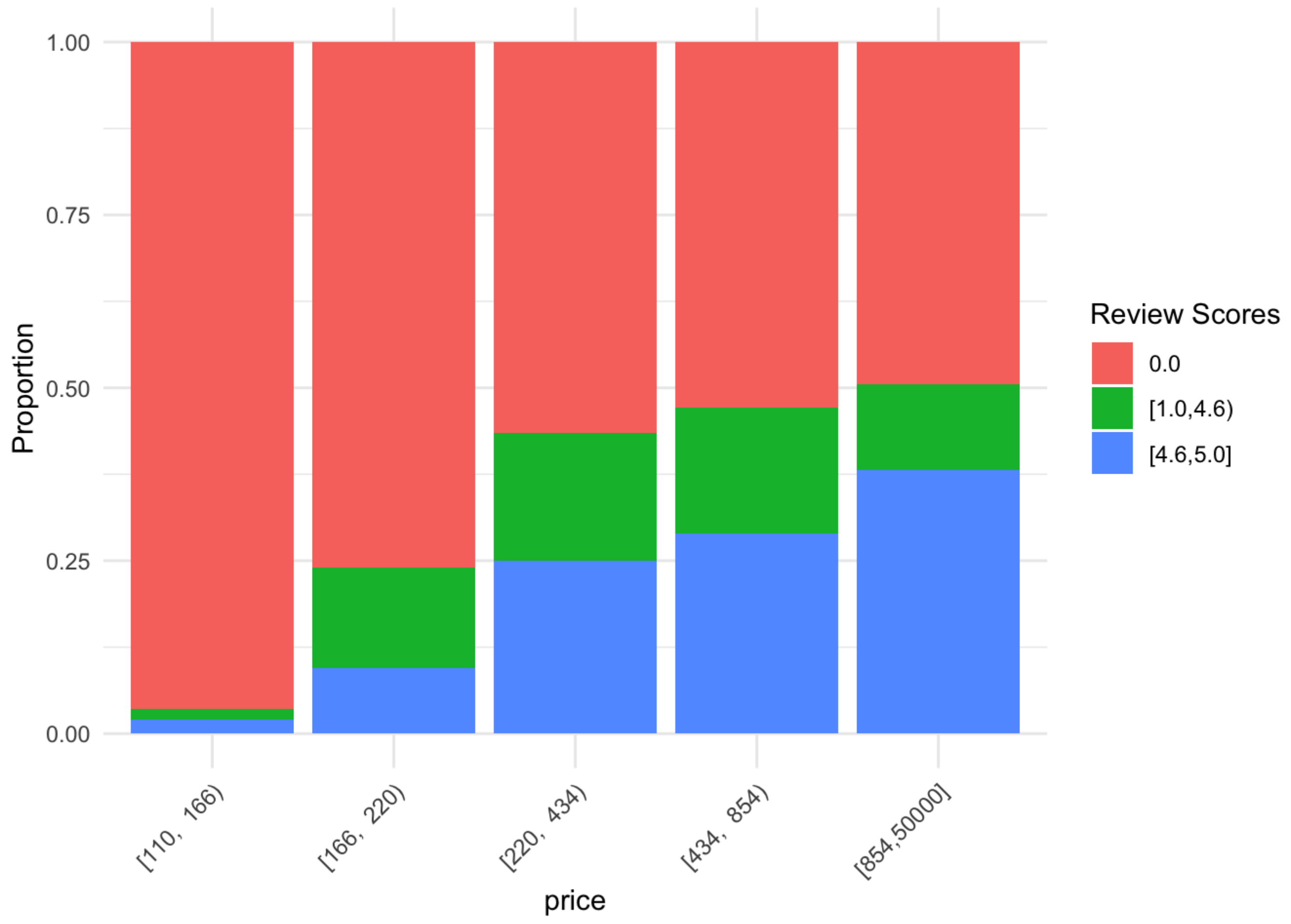

Proportion
1.00
0.75
0.50
0.25
0.00
Entire home/apt
Hotel room
Private room
Shared room
room_type
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

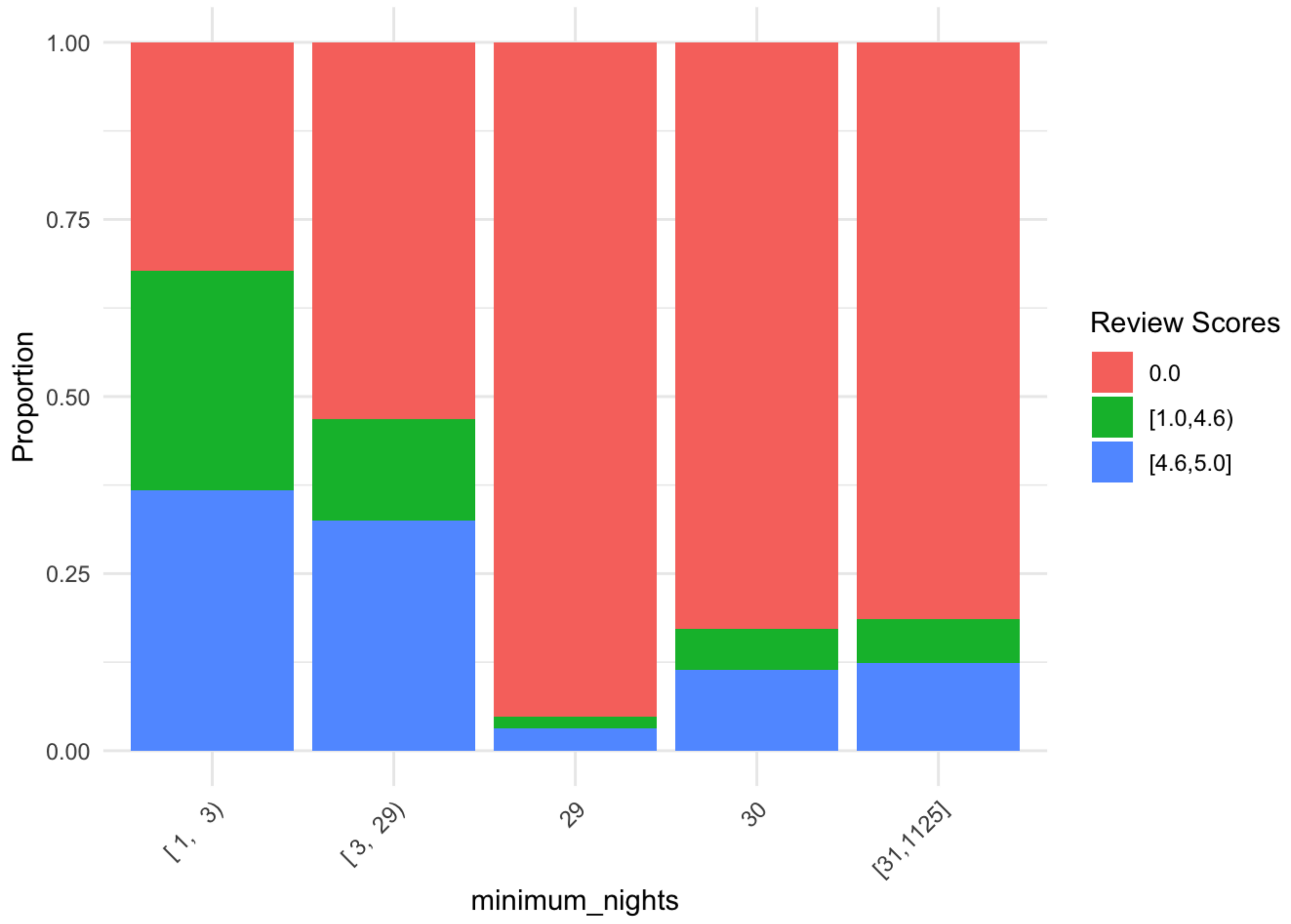

Proportion
1.00
0.75
0.50
0.25
0.00
1
2
[3,16]
accommodates
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

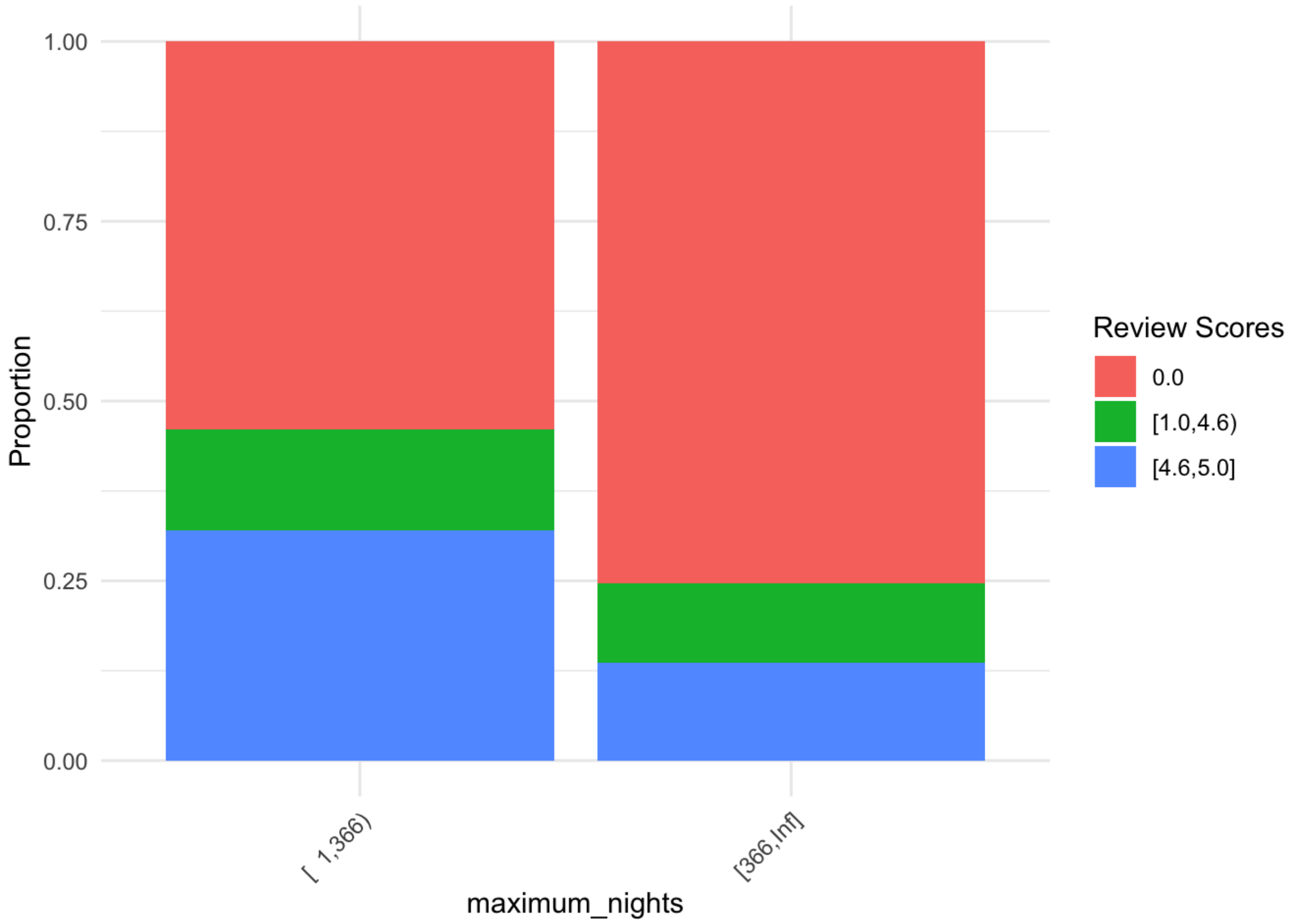

Proportion
1.00
0.75
0.50
0.25
0.00
[0.0, 1.5)
[1.5,50.0]
bathroom_nb
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

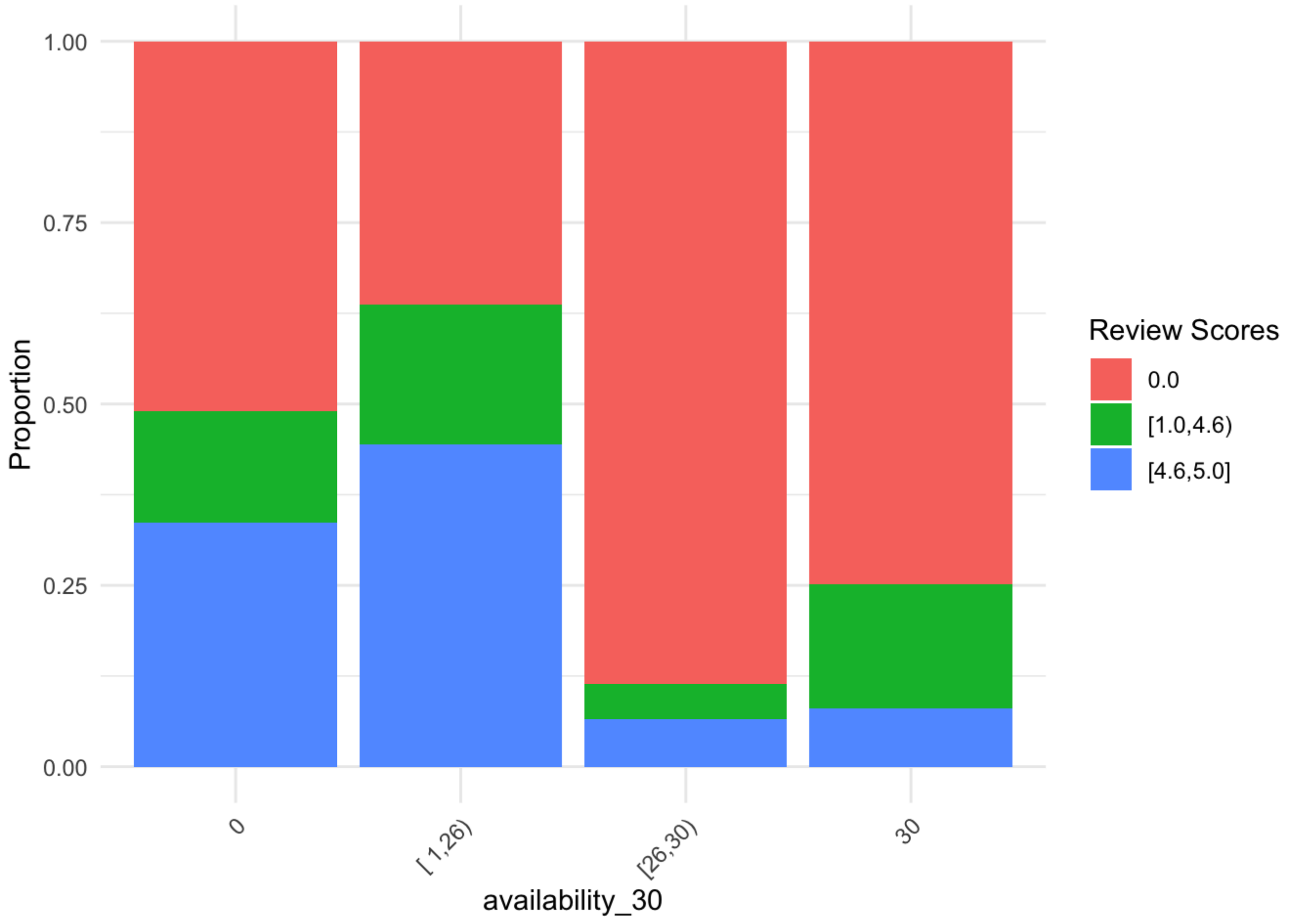

Proportion
1.00
0.75
0.50
0.25
0.00
1
[2,50]
bedrooms
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

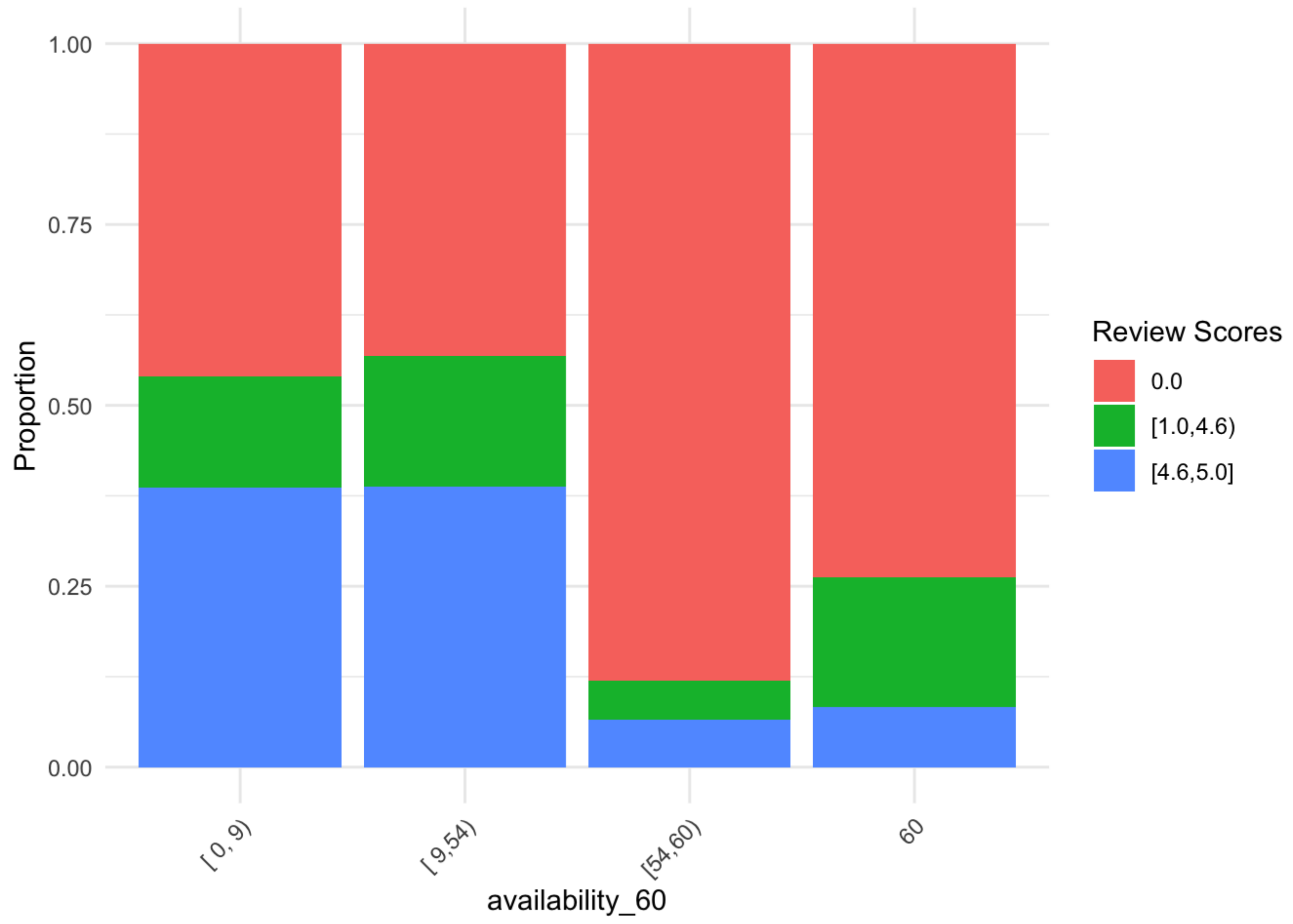

Proportion
1.00
0.75
0.50
0.25
0.00
[0.5, 2)
[2.0,50]
beds
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

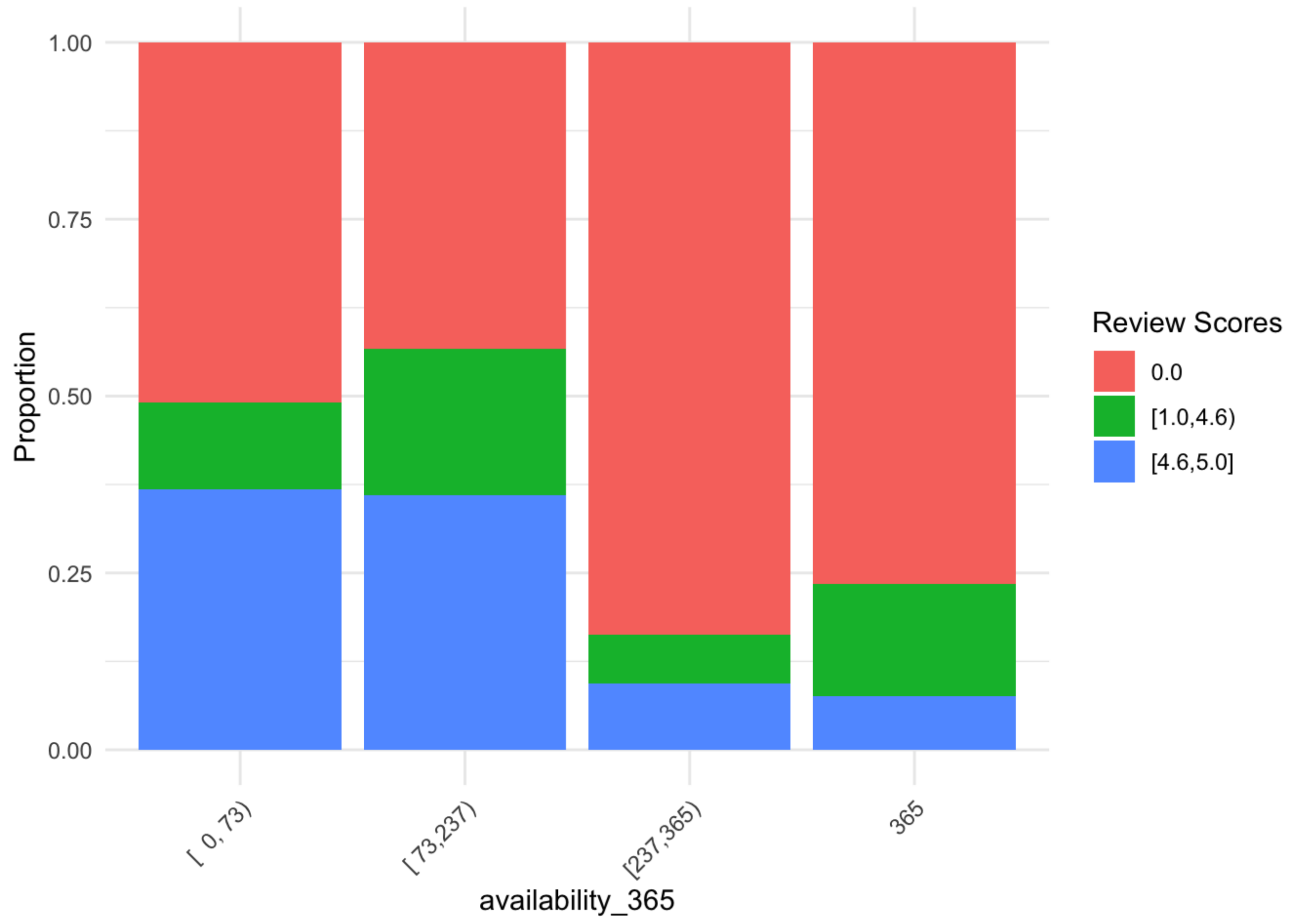

1.00
0.75
0.50
0.25
0.00
Proportion
[110, 166)
[166, 220)
[220, 434)
[434, 854)
[854,50000]
price
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

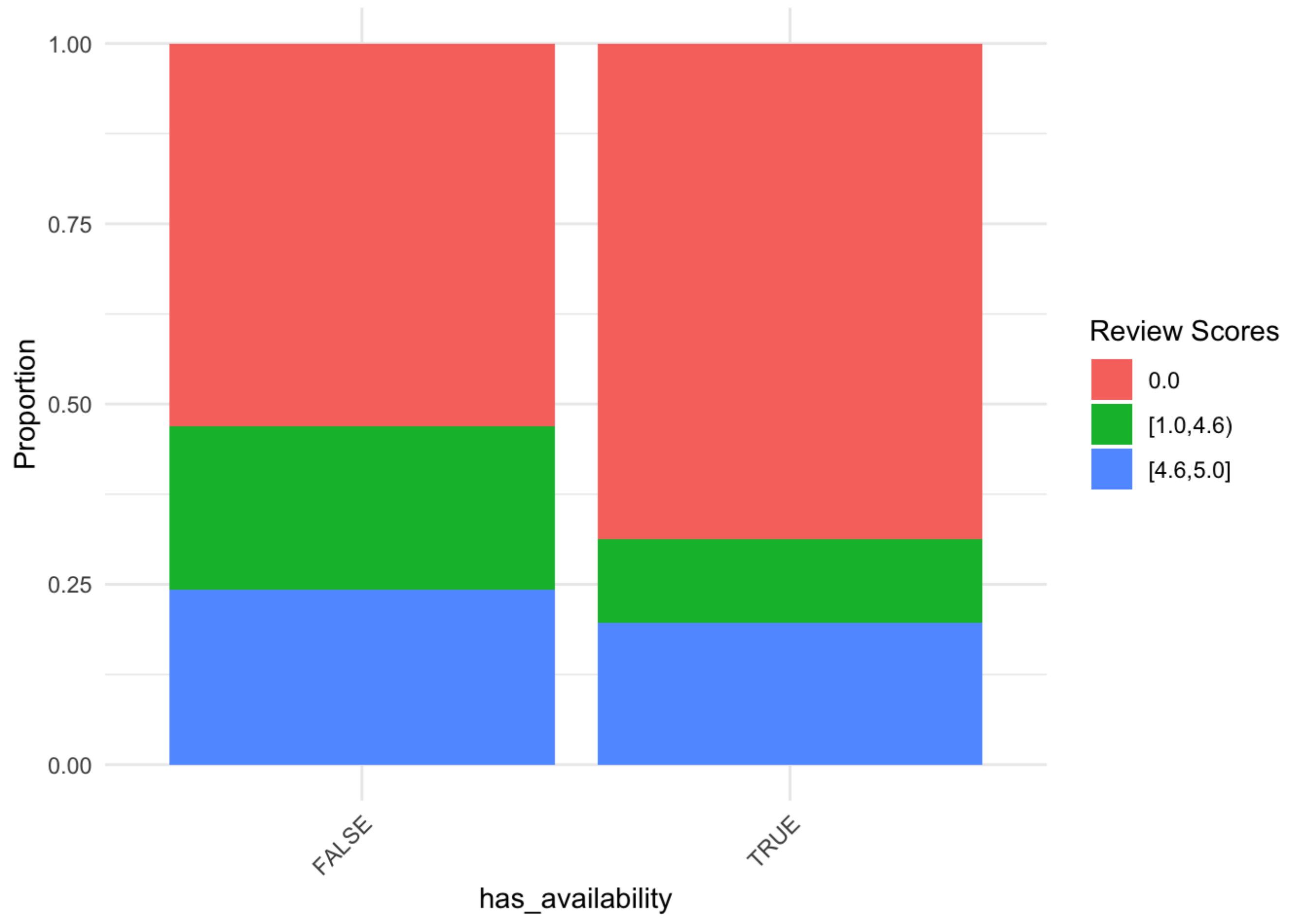

1.00
0.75
0.50
0.25
0.00
Proportion
[1, 3)
[3, 29)
29
30
[31,1125]
minimum_nights
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

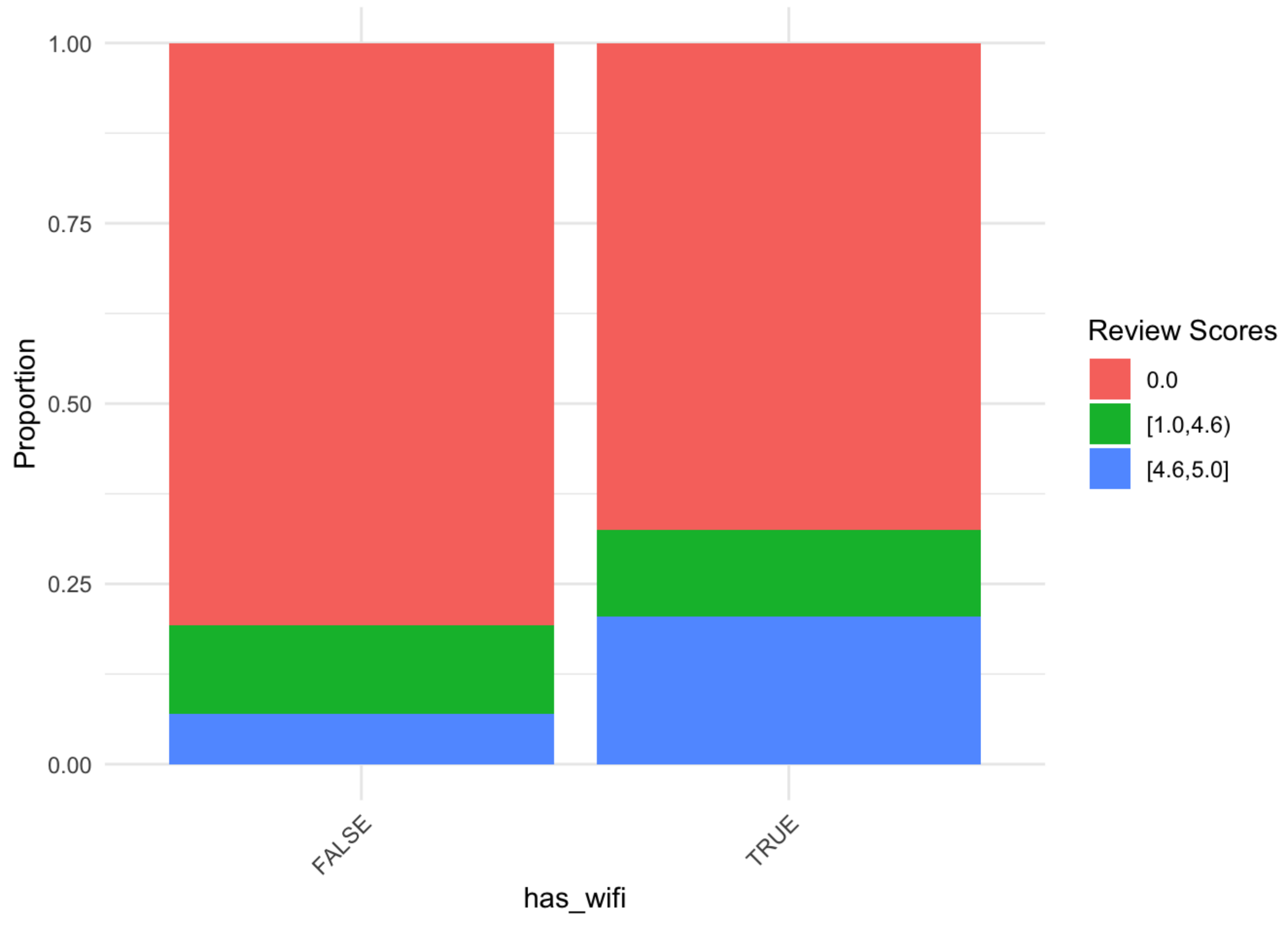

1.00
0.75
0.50
0.25
0.00
Proportion
[1,366)
[366,Inf]
maximum_nights
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

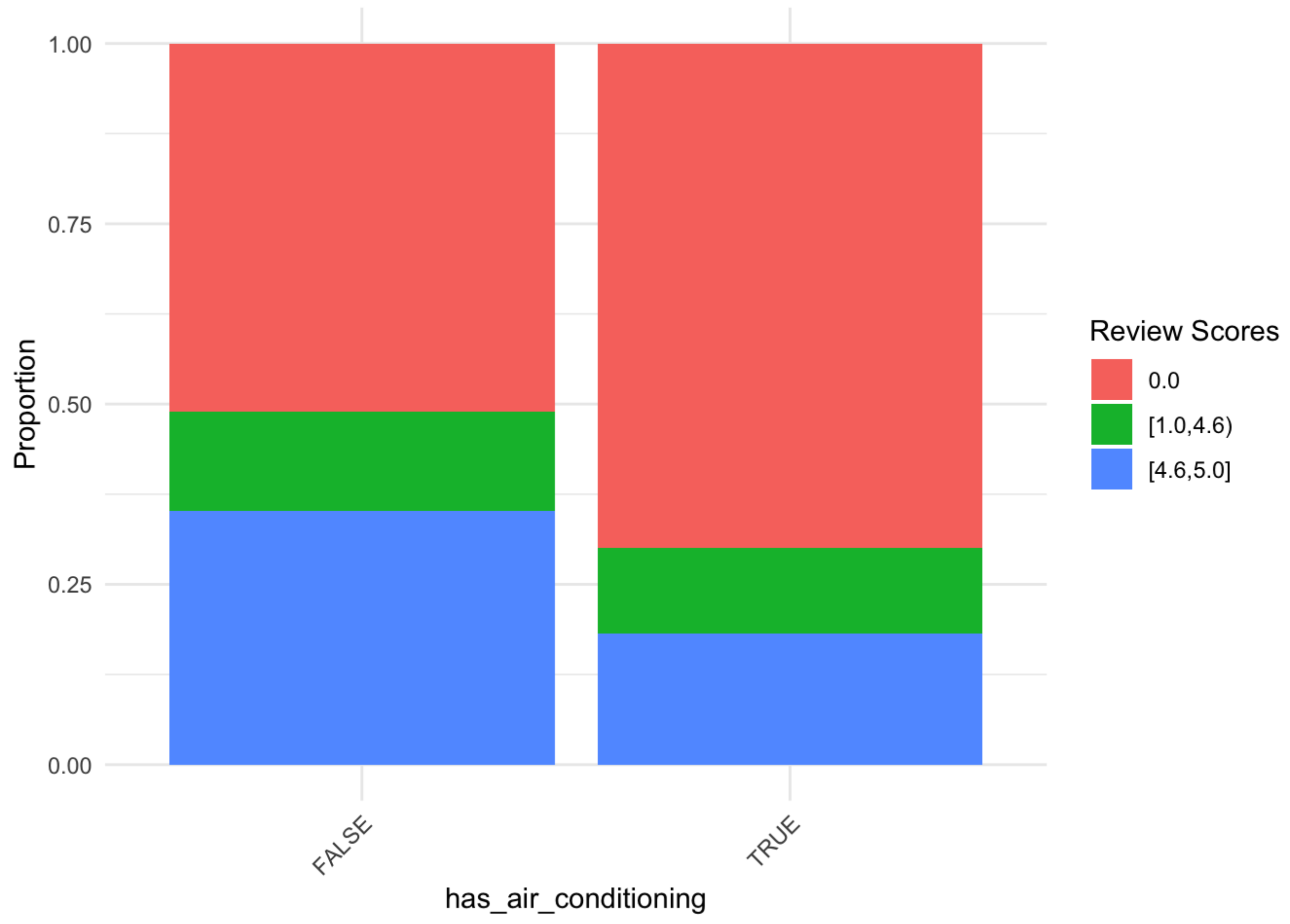

1.00
0.75
0.50
0.25
0.00
Proportion
0
[1,26)
[26,30)
30
availability_30
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

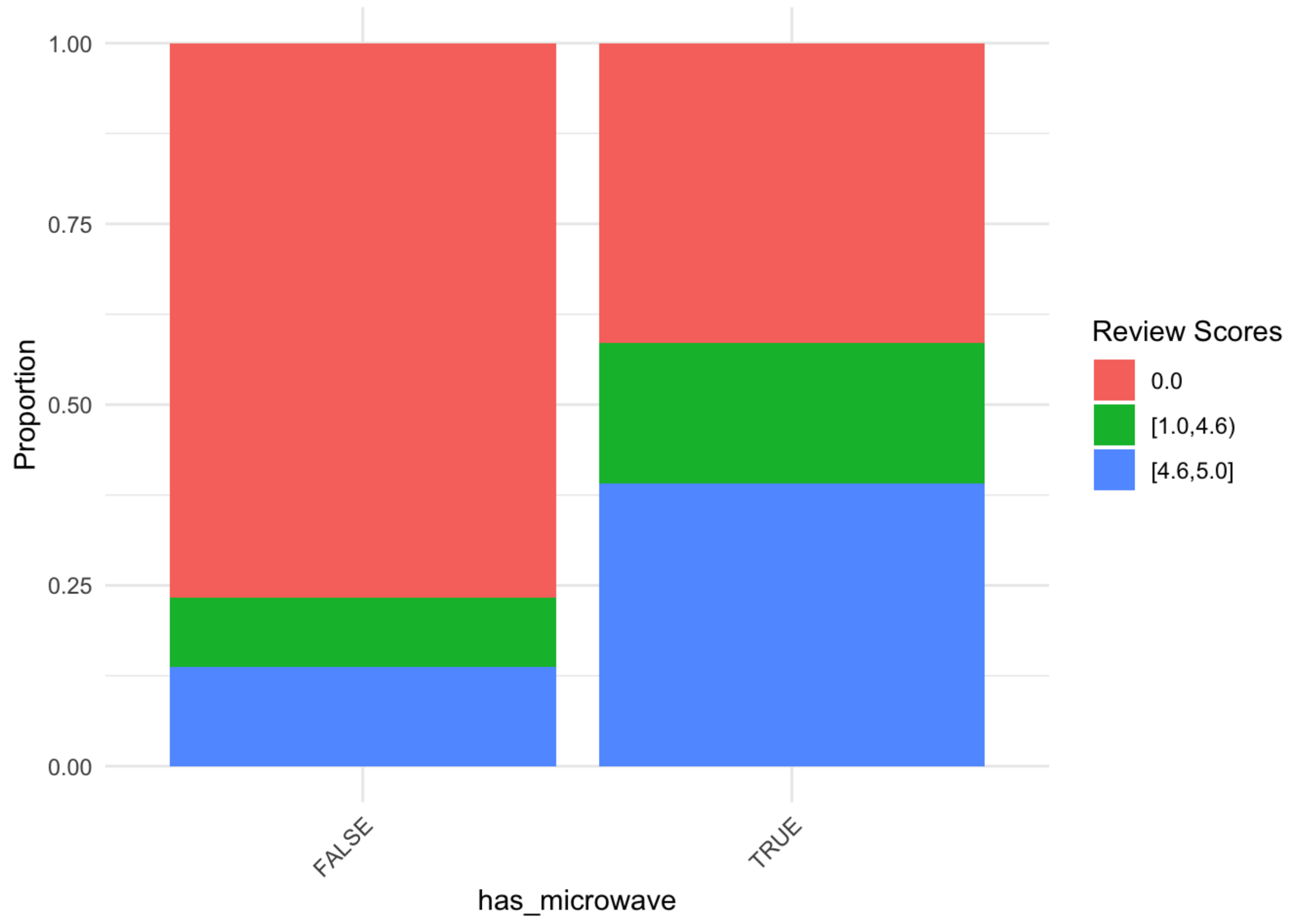

Proportion
1.00
0.75
0.50
0.25
0.00
[0, 9)
[9,54)
[54,60)
60
availability_60
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

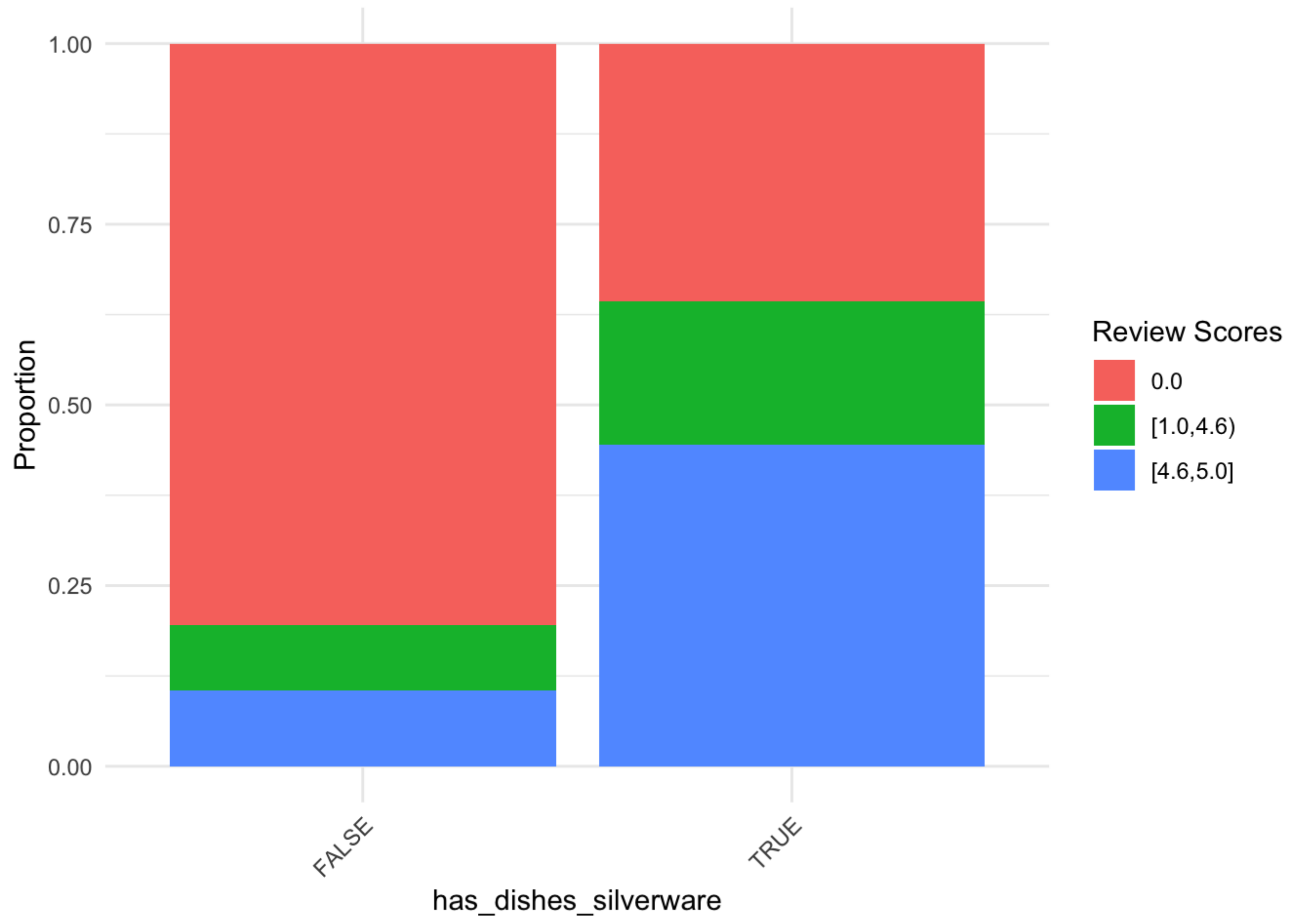

Proportion
1.00
0.75
0.50
0.25
0.00
[0, 73)
[73,237)
[237,365)
365
availability_365
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

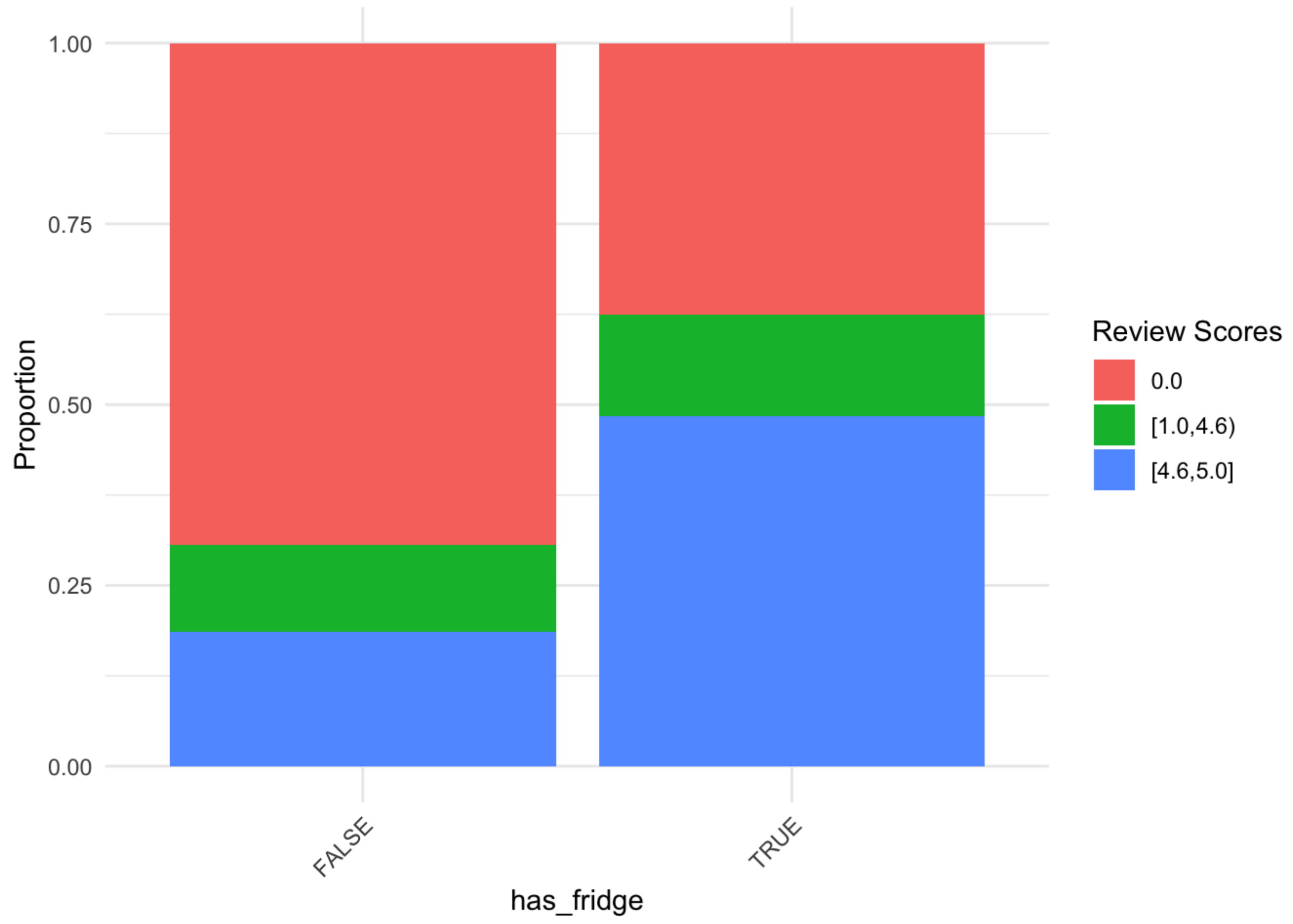

1.00
0.75
0.50
0.25
0.00
Proportion
FALSE
TRUE
has_availability
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

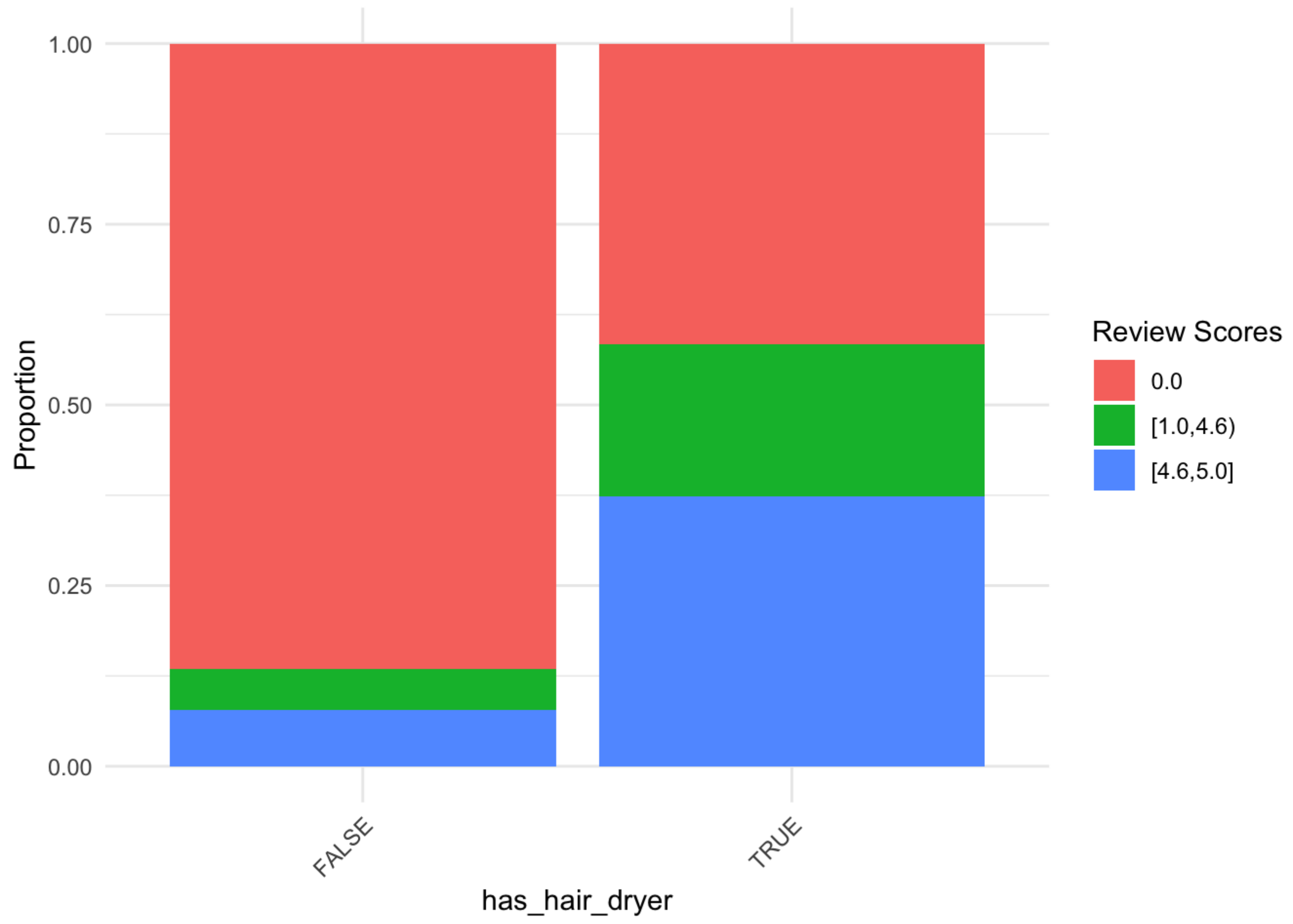

1.00
0.75
0.50
0.25
0.00
Proportion
FALSE
TRUE
has_wifi
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]

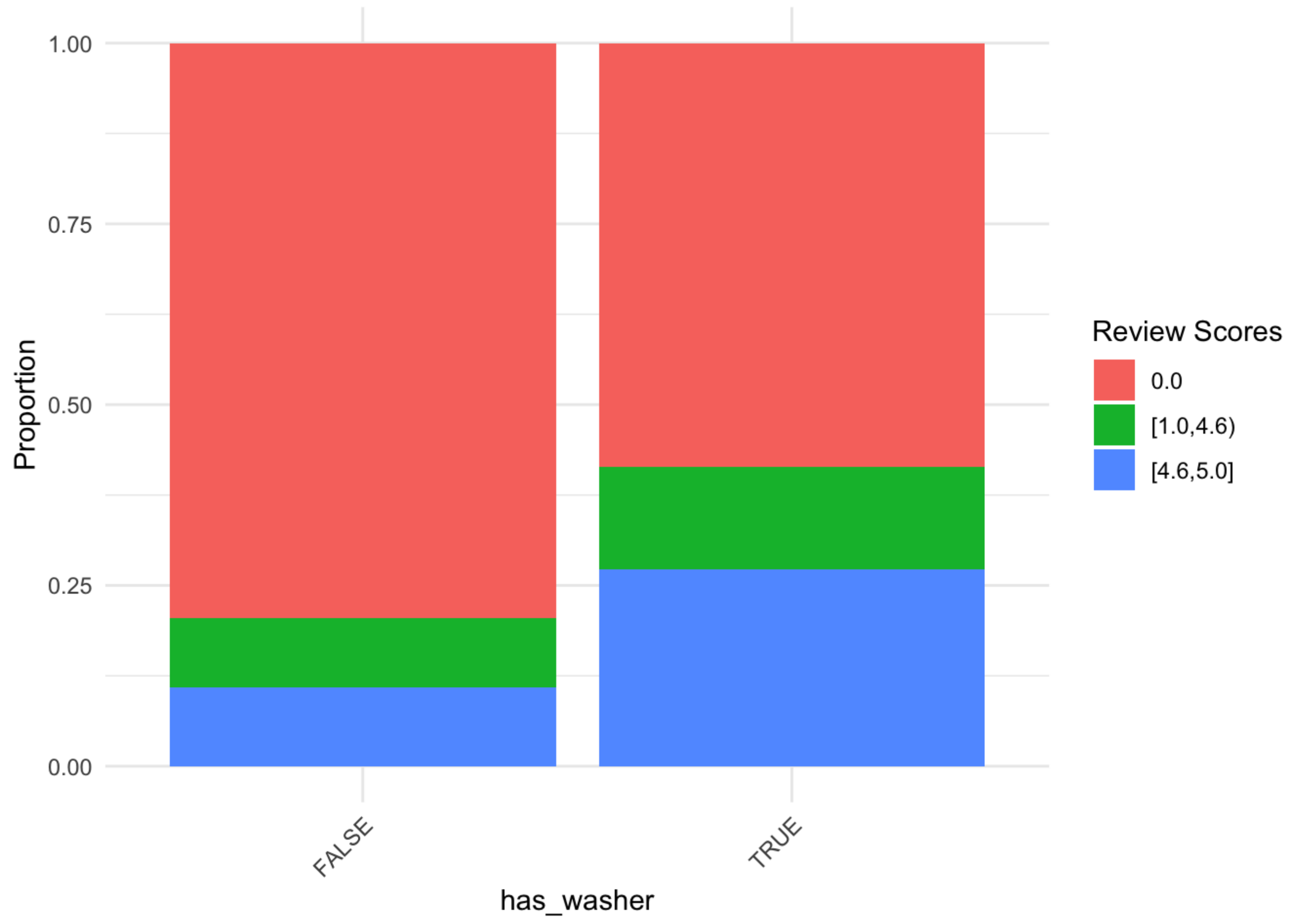

1.00
0.75
0.50
0.25
0.00
Proportion
FALSE
TRUE
has_air_conditioning
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]


1.00
0.75
0.50
0.25
0.00
Proportion
FALSE
TRUE
has_microwave
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]


1.00
0.75
0.50
0.25
0.00
Proportion
FALSE
TRUE
has_dishes_silverware
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]


1.00
0.75
0.50
0.25
0.00
Proportion
FALSE
TRUE
has_fridge
Review Scores
0.0
[1.0,4.6)
[4.6,5.0]





I decide to keep all the variables because they all have predictive power. They can all differentiate the outcome, according to the plots.

```r
str(data)
```

```
## spc_tbl_ [1,424 × 50] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ...1                                   : num [1:1424] 1 2 3 4 5 6 7 8 9 10
...
##  $ description                            : chr [1:1424] "Flat is very quiet.
A/C in each room and living room." "*共居時光*<br /><br />關於我們: <br />共居時光是一家現
代化的服務式公寓公司，為您提供不同類型的舒適房間，從合租"| __truncated__ "near MTR<br />8 rooms
sharing apartment with a kitchen and washing machine<br />with private bathroom<br />
<br "| __truncated__ "This listing is for 1 private room in a shared apartment.<br />
If you like to meet new friends, expand your Net"| __truncated__ ...
##  $ neighborhood_overview                  : chr [1:1424] "None" "None" "None"
"None" ...
##  $ host_id                                : num [1:1424] 1.25e+08 3.81e+07 7.
52e+06 7.52e+06 1.20e+08 ...
##  $ host_since                             : Date[1:1424], format: "2017-04-1
0" "2015-07-10" ...
##  $ host_location                          : chr [1:1424] "Hong Kong" "Hong Ko
ng" "Hong Kong" "Hong Kong" ...
##  $ host_response_time                     : chr [1:1424] "within a few hours"
"within a few hours" "within a few hours" "within a few hours" ...
##  $ host_response_rate                     : num [1:1424] 1 1 0.98 0.98 0.33
0.94 0.94 1 1 1 ...
##  $ host_acceptance_rate                   : num [1:1424] 0.57 0.57 0.57 0.57
0 0.2 0.2 0.26 0.2 1 ...
##  $ host_is_superhost                      : logi [1:1424] FALSE FALSE FALSE F
ALSE FALSE FALSE ...
##  $ host_neighbourhood                     : chr [1:1424] "Wan Chai" "Causeway
Bay" "Wan Chai" "Wan Chai" ...
##  $ host_listings_count                    : num [1:1424] 2 1 365 365 9 441 44
1 367 304 7 ...
##  $ host_total_listings_count              : num [1:1424] 2 1 396 396 11 505 5
05 379 323 9 ...
##  $ host_verifications                     : chr [1:1424] "['email', 'phone',
'work_email']" "['email', 'phone']" "['email', 'phone']" "['email', 'phone']" ...
##  $ host_has_profile_pic                   : logi [1:1424] TRUE TRUE TRUE TRUE
TRUE TRUE ...
##  $ host_identity_verified                 : logi [1:1424] FALSE TRUE TRUE TRU
E FALSE TRUE ...
##  $ latitude                               : num [1:1424] 22.3 22.3 22.3 22.3
22.3 ...
##  $ longitude                              : num [1:1424] 114 114 114 114 114
...
##  $ property_type                          : chr [1:1424] "Entire rental unit"
"Shared room in serviced apartment" "Private room in rental unit" "Private room in re
ntal unit" ...
##  $ room_type                              : chr [1:1424] "Entire home/apt" "S
hared room" "Private room" "Private room" ...
##  $ accommodates                           : num [1:1424] 2 7 2 2 3 1 1 1 2 2
...
##  $ bathroom_nb                            : num [1:1424] 1.5 2 1 1 1 1 1 1 1
1 ...
##  $ bathroom_type                          : chr [1:1424] "normal" "shared" "n
ormal" "private" ...
##  $ bedrooms                               : num [1:1424] 2 7 1 1 1 1 1 1 1 1
...
##  $ beds                                   : num [1:1424] 2 7 1 1 1 1 1 1 1 1
```

```
...
##  $ amenities                                   : chr [1:1424] "[\"Wifi\", \"Kitche
n\", \"Washer\", \"Air conditioning\", \"Smoke alarm\"]" "[\"Hangers\", \"Wifi\", \"H
ot water\", \"TV\", \"Shampoo\", \"Kitchen\", \"Washer\", \"Air conditioning\", \"Ele
vator\"]" "[\"Iron\", \"Air conditioning\", \"Wifi\", \"Kitchen\"]" "[\"Iron\", \"Han
gers\", \"Wifi\", \"Lock on bedroom door\", \"Kitchen\", \"Washer\", \"Air conditioni
ng\"]" ...
##  $ price                                       : num [1:1424] 470 500 217 160 1150
180 180 180 160 950 ...
##  $ minimum_nights                              : num [1:1424] 30 30 29 29 30 30 30
29 29 28 ...
##  $ maximum_nights                              : num [1:1424] 43 365 1125 1125 366
...
##  $ has_availability                            : logi [1:1424] FALSE TRUE TRUE TRU
E TRUE TRUE ...
##  $ availability_30                             : num [1:1424] 0 23 29 29 0 30 30 2
9 29 0 ...
##  $ availability_60                             : num [1:1424] 0 53 59 59 0 60 60 5
9 59 0 ...
##  $ availability_90                             : num [1:1424] 0 83 89 89 0 90 90 8
9 89 0 ...
##  $ availability_365                            : num [1:1424] 0 83 364 364 0 365 3
65 364 364 0 ...
##  $ number_of_reviews                           : num [1:1424] 1 0 0 0 0 1 0 0 0 0
...
##  $ number_of_reviews_ltm                       : num [1:1424] 0 0 0 0 0 0 0 0 0 0
...
##  $ number_of_reviews_l30d                      : num [1:1424] 0 0 0 0 0 0 0 0 0 0
...
##  $ review_scores_rating                        : num [1:1424] 5 0 0 0 0 4 0 0 0 0
...
##  $ review_scores_accuracy                      : num [1:1424] 5 0 0 0 0 4 0 0 0 0
...
##  $ review_scores_cleanliness                   : num [1:1424] 5 0 0 0 0 3 0 0 0 0
...
##  $ review_scores_checkin                       : num [1:1424] 5 0 0 0 0 3 0 0 0 0
...
##  $ review_scores_communication                 : num [1:1424] 5 0 0 0 0 5 0 0 0 0
...
##  $ review_scores_location                      : num [1:1424] 5 0 0 0 0 5 0 0 0 0
...
##  $ review_scores_value                         : num [1:1424] 5 0 0 0 0 3 0 0 0 0
...
##  $ instant_bookable                            : logi [1:1424] FALSE FALSE FALSE F
ALSE FALSE FALSE ...
##  $ calculated_host_listings_count_entire_homes : num [1:1424] 1 0 15 15 5 13 13 18
26 6 ...
##  $ calculated_host_listings_count_private_rooms: num [1:1424] 0 0 322 322 2 384 38
4 332 265 0 ...
##  $ calculated_host_listings_count_shared_rooms : num [1:1424] 0 1 28 28 0 8 8 16 1
2 0 ...
##  $ reviews_per_month                           : num [1:1424] 0.05 0 0 0 0 0.01 0
0 0 0 ...
##  $ got_reviewed                                : num [1:1424] 1 0 0 0 0 1 0 0 0 0
...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ...1 = col_double(),
##   ..   description = col_character(),
##   ..   neighborhood_overview = col_character(),
##   ..   host_id = col_double(),
##   ..   host_since = col_date(format = ""),
##   ..   host_location = col_character(),
##   ..   host_response_time = col_character(),
##   ..   host_response_rate = col_double(),
##   ..   host_acceptance_rate = col_double(),
##   ..   host_is_superhost = col_logical(),
##   ..   host_neighbourhood = col_character(),
##   ..   host_listings_count = col_double(),
##   ..   host_total_listings_count = col_double(),
##   ..   host_verifications = col_character(),
##   ..   host_has_profile_pic = col_logical(),
##   ..   host_identity_verified = col_logical(),
##   ..   latitude = col_double(),
##   ..   longitude = col_double(),
##   ..   property_type = col_character(),
##   ..   room_type = col_character(),
##   ..   accommodates = col_double(),
##   ..   bathroom_nb = col_double(),
##   ..   bathroom_type = col_character(),
##   ..   bedrooms = col_double(),
##   ..   beds = col_double(),
##   ..   amenities = col_character(),
##   ..   price = col_double(),
##   ..   minimum_nights = col_double(),
##   ..   maximum_nights = col_double(),
##   ..   has_availability = col_logical(),
##   ..   availability_30 = col_double(),
##   ..   availability_60 = col_double(),
##   ..   availability_90 = col_double(),
##   ..   availability_365 = col_double(),
##   ..   number_of_reviews = col_double(),
##   ..   number_of_reviews_ltm = col_double(),
##   ..   number_of_reviews_l30d = col_double(),
##   ..   review_scores_rating = col_double(),
##   ..   review_scores_accuracy = col_double(),
##   ..   review_scores_cleanliness = col_double(),
##   ..   review_scores_checkin = col_double(),
##   ..   review_scores_communication = col_double(),
##   ..   review_scores_location = col_double(),
##   ..   review_scores_value = col_double(),
##   ..   instant_bookable = col_logical(),
##   ..   calculated_host_listings_count_entire_homes = col_double(),
##   ..   calculated_host_listings_count_private_rooms = col_double(),
##   ..   calculated_host_listings_count_shared_rooms = col_double(),
##   ..   reviews_per_month = col_double(),
##   ..   got_reviewed = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

I create a dataframe with the variables I will use:

```
variables_of_interest <- c("host_response_rate", "host_is_superhost", "room_type", "a
ccommodates", "bathroom_nb", "bedrooms", "beds","price", "minimum_nights", "maximum_n
ights", "availability_30", "availability_60","availability_365","has_availability","h
as_wifi", "has_air_conditioning","has_microwave", "has_dishes_silverware","has_fridg
e","has_hair_dryer","has_washer", "review_scores_rating")
```

```
set.seed(60)
train.index <- sample(c(1:dim(data2)[1]), dim(data2)[1]*0.6)
train.df <- data2[train.index, variables_of_interest]
valid.df <- data2[-train.index, variables_of_interest]
```

A. The Naive Bayes Model:

```
library(e1071)
nb_model <- naiveBayes(review_scores_rating ~ . , data = train.df)

nb_model
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##       0.0 [1.0,4.6) [4.6,5.0]
## 0.6569087 0.1334895 0.2096019
##
## Conditional probabilities:
##            host_response_rate
## Y           [0.00,0.95) [0.95,0.99) [0.99, Inf]
##   0.0        0.28163993  0.20499109  0.51336898
##   [1.0,4.6)  0.21929825  0.04385965  0.73684211
##   [4.6,5.0]  0.18435754  0.02234637  0.79329609
##
##            host_is_superhost
## Y                FALSE       TRUE
##   0.0       0.92156863 0.07843137
##   [1.0,4.6) 0.97368421 0.02631579
##   [4.6,5.0] 0.84357542 0.15642458
##
##            room_type
## Y           Entire home/apt  Hotel room Private room Shared room
##   0.0           0.249554367 0.000000000  0.711229947 0.039215686
##   [1.0,4.6)     0.333333333 0.000000000  0.491228070 0.175438596
##   [4.6,5.0]     0.608938547 0.005586592  0.312849162 0.072625698
##
##            accommodates
## Y                    1          2     [3,16]
##   0.0       0.38502674 0.51871658 0.09625668
##   [1.0,4.6) 0.23684211 0.36842105 0.39473684
##   [4.6,5.0] 0.14525140 0.46368715 0.39106145
##
##            bathroom_nb
## Y           [0.0, 1.5) [1.5,50.0]
##   0.0        0.8752228  0.1247772
##   [1.0,4.6)  0.7105263  0.2894737
##   [4.6,5.0]  0.7988827  0.2011173
##
##            bedrooms
## Y                    1     [2,50]
##   0.0       0.91087344 0.08912656
##   [1.0,4.6) 0.66666667 0.33333333
##   [4.6,5.0] 0.60893855 0.39106145
##
##            beds
## Y             [0.5, 2)  [2.0,50]
##   0.0       0.8930481 0.1069519
##   [1.0,4.6) 0.5964912 0.4035088
##   [4.6,5.0] 0.6312849 0.3687151
##
##            price
## Y           [110,  166) [166,  220) [220,  434) [434,  854) [854,50000]
##   0.0        0.39572193  0.14616756  0.16577540  0.15686275  0.13547237
##   [1.0,4.6)  0.04385965  0.15789474  0.28947368  0.28947368  0.21929825
##   [4.6,5.0]  0.02793296  0.08379888  0.23463687  0.27932961  0.37430168
##
##            minimum_nights
## Y             [ 1,   3)  [ 3,  29)         29         30  [31,1125]
##   0.0       0.09803922 0.21568627 0.39037433 0.22281640 0.07308378
##   [1.0,4.6) 0.54385965 0.28947368 0.04385965 0.10526316 0.01754386
##   [4.6,5.0] 0.37988827 0.41340782 0.04469274 0.12290503 0.03910615
##
##            maximum_nights
## Y           [  1,366) [366,Inf]
##   0.0        0.285205  0.714795
##   [1.0,4.6)  0.377193  0.622807
##   [4.6,5.0]  0.547486  0.452514
##
##            availability_30
## Y                    0    [ 1,26)    [26,30)         30
##   0.0       0.17112299 0.10338681 0.53119430 0.19429590
##   [1.0,4.6) 0.28070175 0.28070175 0.20175439 0.23684211
##   [4.6,5.0] 0.33519553 0.41899441 0.15642458 0.08938547
##
##            availability_60
## Y               [ 0, 9)    [ 9,54)    [54,60)         60
##   0.0       0.14438503 0.13903743 0.53832442 0.17825312
##   [1.0,4.6) 0.28947368 0.25438596 0.21929825 0.23684211
##   [4.6,5.0] 0.35754190 0.39106145 0.16201117 0.08938547
##
##            availability_365
## Y             [  0, 73)  [ 73,237)  [237,365)        365
##   0.0       0.16399287 0.14260250 0.53475936 0.15864528
##   [1.0,4.6) 0.21929825 0.33333333 0.25438596 0.19298246
##   [4.6,5.0] 0.35195531 0.34636872 0.23463687 0.06703911
##
##            has_availability
## Y                FALSE       TRUE
##   0.0       0.03565062 0.96434938
##   [1.0,4.6) 0.11403509 0.88596491
##   [4.6,5.0] 0.06145251 0.93854749
##
##            has_wifi
## Y                FALSE       TRUE
##   0.0       0.04634581 0.95365419
##   [1.0,4.6) 0.04385965 0.95614035
##   [4.6,5.0] 0.01675978 0.98324022
##
##            has_air_conditioning
## Y                FALSE       TRUE
##   0.0       0.06773619 0.93226381
##   [1.0,4.6) 0.08771930 0.91228070
##   [4.6,5.0] 0.17877095 0.82122905
##
##            has_microwave
## Y               FALSE      TRUE
##   0.0       0.8538324 0.1461676
##   [1.0,4.6) 0.6140351 0.3859649
##   [4.6,5.0] 0.5027933 0.4972067
##
##            has_dishes_silverware
## Y                FALSE      TRUE
##   0.0       0.8502674 0.1497326
##   [1.0,4.6) 0.5175439 0.4824561
##   [4.6,5.0] 0.3854749 0.6145251
##
##            has_fridge
## Y                 FALSE       TRUE
##   0.0       0.97860963 0.02139037
##   [1.0,4.6) 0.96491228 0.03508772
##   [4.6,5.0] 0.92178771 0.07821229
##
##            has_hair_dryer
## Y                FALSE      TRUE
##   0.0       0.7540107 0.2459893
##   [1.0,4.6) 0.2807018 0.7192982
##   [4.6,5.0] 0.2625698 0.7374302
##
##            has_washer
## Y                FALSE      TRUE
##   0.0       0.4955437 0.5044563
##   [1.0,4.6) 0.3508772 0.6491228
##   [4.6,5.0] 0.2625698 0.7374302
```

```
levels(data2$review_scores_rating)
```

```
## [1] "0.0"       "[1.0,4.6)" "[4.6,5.0]"
```

B. Describe a fictional apartment, and use your model to predict which bin it will fall into.

```
fictional_apartment <- data.frame(
  host_response_rate = 0.8,
  host_is_superhost = TRUE,
  room_type = "Private room",
  accommodates= 3,
  bathroom_nb = 1,
  bedrooms = 3,
  beds = 3,
  price = 300,
  minimum_nights = 2,
  maximum_nights= 10,
  availability_30 = 26,
  availability_60= 25,
  availability_365= 30,
  availability_365= 30,
  has_availability = TRUE,
  has_wifi=TRUE,
  has_air_conditioning=TRUE,
  has_dishes_silverware=TRUE,
  has_fridge=TRUE,
  has_hair_dryer=FALSE,
  has_washer=FALSE
)

predicted_bin <- predict(nb_model, newdata = fictional_apartment)

cat("Predicted Review Scores Rating Bin:", predicted_bin, "\\n")
```

```
## Predicted Review Scores Rating Bin: 2 \n
```

The model predicts this apartment will fall in bin 2 which covers Score ratings in the range: 1.0 - 4.6.

C. Assessing performance The predictions in the training set have an accuracy of 0.7143, while the predictions in the validation set have an accuracy of 0.7474.This means the model is not overfitting to the training data and is making accurate predictions on unseen data.

```
train_predictions <- predict(nb_model, newdata = train.df)

confusion_train <- confusionMatrix(data = train_predictions, reference = train.df$rev
iew_scores_rating)

accuracy_train <- confusion_train$overall["Accuracy"]

confusion_train
```

```
## Confusion Matrix and Statistics
## 
##             Reference
## Prediction   0.0 [1.0,4.6) [4.6,5.0]
##   0.0        441        29        31
##   [1.0,4.6)   17        40        19
##   [4.6,5.0] 103        45       129
## 
## Overall Statistics
## 
##                Accuracy : 0.7143
##                  95% CI : (0.6827, 0.7444)
##     No Information Rate : 0.6569
##     P-Value [Acc > NIR] : 0.0001971
## 
##                   Kappa : 0.4657
## 
##  Mcnemar's Test P-Value : 2.486e-11
## 
## Statistics by Class:
## 
##                      Class: 0.0 Class: [1.0,4.6) Class: [4.6,5.0]
## Sensitivity              0.7861          0.35088           0.7207
## Specificity              0.7952          0.95135           0.7807
## Pos Pred Value           0.8802          0.52632           0.4657
## Neg Pred Value           0.6601          0.90488           0.9133
## Prevalence               0.6569          0.13349           0.2096
## Detection Rate           0.5164          0.04684           0.1511
## Detection Prevalence     0.5867          0.08899           0.3244
## Balanced Accuracy        0.7907          0.65111           0.7507
```

Prediction on Training Data:

```
valid_predictions <- predict(nb_model, newdata = valid.df)

confusion_valid <- confusionMatrix(valid_predictions, valid.df$review_scores_rating)

confusion_valid
```

```
## Confusion Matrix and Statistics
## 
##            Reference
## Prediction  0.0 [1.0,4.6) [4.6,5.0]
##   0.0       326        13        13
##   [1.0,4.6)  17        16         8
##   [4.6,5.0]  64        29        84
## 
## Overall Statistics
## 
##                Accuracy : 0.7474          
##                  95% CI : (0.7096, 0.7826)
##     No Information Rate : 0.714           
##     P-Value [Acc > NIR] : 0.04193         
## 
##                   Kappa : 0.4891          
## 
##  Mcnemar's Test P-Value : 5.064e-10       
## 
## Statistics by Class:
## 
##                      Class: 0.0 Class: [1.0,4.6) Class: [4.6,5.0]
## Sensitivity              0.8010          0.27586           0.8000
## Specificity              0.8405          0.95117           0.8000
## Pos Pred Value           0.9261          0.39024           0.4746
## Neg Pred Value           0.6284          0.92060           0.9466
## Prevalence               0.7140          0.10175           0.1842
## Detection Rate           0.5719          0.02807           0.1474
## Detection Prevalence     0.6175          0.07193           0.3105
## Balanced Accuracy        0.8207          0.61352           0.8000
```

D. Write a two-paragraph narrative that describes how you did this. In your narrative, be sure to talk about things like feature selection and assessing performance against your training data and validation data.

First, we selected the amenities that we thought had the biggest influence on rating. These are “washer”, “wifi”, “microwave”, “dishes and silverware”, mini fridge” and “hair dryer” Then we converted these amenities into variables using the grepl() function. Second, we converted the character variables and the amenities logical variables into factors. For feature selection, we focused on the variables we thought had the greatest influence on rating and we assessed their impact with proportional barplots. According to the barplots, all the variables have high predictive power, as they can all differentiate the rating outcome. Therefore, we decided to keep them all in the model.

After partitioning the data, we run the model and created a fictional apartment which was predicted to fall in bin number 2 which is the one with Score ratings in the range: 1.0 - 4.6. Finally, we assessed the performance making predictions on the training set and the validation set by using a confusion matrix. The predictions in the training set have an accuracy of 0.7143, while the predictions in the validation set have an accuracy of 0.7474. This means the model has a high predictive power. It is not overfitting to the training data and is making accurate predictions on unseen data.