Práctica 2: Limpieza y validación de los datos

Autor: Alberto Giménez Aragón

Enero 2021

Contents

1 Descripción del dataset

Para la realización de esta práctica he escogido el dataset que generamos en la práctica 1. Este dataset recoge una gran variedad de datos sobre los distintos hoteles repartidos a lo largo de la ciudad de Barcelona y alrededores, los cuales fueron obtenidos mediante *web scraping* a fecha de 2 de noviembre de 2020.

A continuación, se detallan todos los atributos de los que dispone el dataset.

- **Name**: Nombre del Hotel
- **Stars**: Estrellas del hotel, entre 1 y 5 estrellas
- **Score**: Calificación del hotel según los usuarios.
- **Score Location**: Calificación de la ubicación del hotel según los usuarios (de 1.0 a 5.0).
- **Score Cleaning**: Calificación de la limpieza del hotel según los usuarios (de 1.0 a 5.0).
- **Score Service**: Calificación del servicio del hotel según los usuarios (de 1.0 a 5.0).
- **Score Value for Money**: Calificación de la relación calidad/precio del hotel según los usuarios (de 1.0 a 5.0).
- **Price**: Precio actual por noche del hotel
- **Price Range**: Rango de precios por noche en el que se encuentra el hotel

- **Ranking**: Ranking Tripadvisor del hotel respecto a otro hoteles de la ciudad
- **Number opinions**: Número de opiniones dejadas por los huéspedes a través de Tripadvisor
- **Number opinions excellent**: Número de opiniones excelentes dejadas por los huéspedes a través de Tripadvisor
- **Number opinions good**: Número de opiniones buenas dejadas por los huéspedes a través de Tripadvisor
- **Number opinions normal**: Número de opiniones normales dejadas por los huéspedes a través de Tripadvisor
- **Number opinions bad**: Número de opiniones malas dejadas por los huéspedes a través de Tripadvisor
- **Number opinions awful**: Número de opiniones pésimas dejadas por los huéspedes a través de Tripadvisor
- **Number QA**: Número de preguntas y respuestas de los usuarios
- **Nearby restaurants**: Número de restaurantes cercanos al hotel
- **Nearby attractions**: Número de atracciones turísticas cercanas al hotel
- **Zone**: Zona en la que se sitúa el hotel
- **Latitude/Longitude**: Latitud y longitud del hotel
- **Swimming pool**: Nos dice si el hotel dispone de este servicio
- **Bar**: Nos dice si el hotel dispone de este servicio
- **Restaurant**: Nos dice si el hotel dispone de este servicio
- **Breakfast**: Nos dice si el hotel dispone de este servicio
- **Gym**: Nos dice si el hotel dispone de este servicio
- **Reception 24h**: Nos dice si el hotel dispone de este servicio
- **Admit pets**: Nos dice si el hotel dispone de este servicio
- **Air conditioning**: Nos dice si el hotel dispone de este servicio
- **Strong box**: Nos dice si el hotel dispone de este servicio
- **Rooms**: Número de habitaciones del hotel
- **Suites**: Nos dice si el hotel dispone de este tipo de habitaciones
- **Sea View Rooms**: Nos dice si el hotel dispone de este tipo de habitaciones
- **Non-smoking Rooms**: Nos dice si el hotel dispone de este tipo de habitaciones
- **Landmark View Rooms**: Nos dice si el hotel dispone de este tipo de habitaciones
- **City View Rooms**: Nos dice si el hotel dispone de este tipo de habitaciones
- **Family Rooms**: Nos dice si el hotel dispone de este tipo de habitaciones
- **Style**: Estilo del hotel
- **Tripadvisor Clasification**: Calificación de Tripadvisor sobre la facilidad de realizar actividades y encontrar restaurantes a corta distancia del hotel.
- **Language Spanish**: Indica si el hotel habla este idioma o no
- **Language Catalan**: Indica si el hotel habla este idioma o no
- **Language French**: Indica si el hotel habla este idioma o no
- **Language English**: Indica si el hotel habla este idioma o no
- **Language Italian**: Indica si el hotel habla este idioma o no
- **Language Bulgarian**: Indica si el hotel habla este idioma o no
- **Language Russian**: Indica si el hotel habla este idioma o no
- **Language Portuguese**: Indica si el hotel habla este idioma o no
- **Prat Distance**: Distancia en kilómetros al aeropuerto del Prat
- **Timestamp**: Fecha/hora de recogida de los datos

Con la elección de este dataset, se pretende dar respuesta a diferentes preguntas sobre los hoteles, con el objetivo de ver si algunos atributos varían o no respecto a los barrios donde se sitúan los hoteles o entre diferentes rangos de precio. De esta forma, un posible turista puede ayudarse por este estudio para elegir entre los hoteles que más le convengan. Algunas de estas preguntas podrían ser las siguientes.

- ¿Hay relación entre las estrellas de un hotel y la valoración de los usuarios?
- ¿Hay diferencias en los precios de hoteles entre diferentes barrios de Barcelona?
- ¿Es cierto que los hoteles más caros ofrecen una mejor limpieza de sus instalaciones?
- ¿Son más caros aquellos hoteles que tienen ciertos servicios como aire acondicionado respecto los que

no tienen?
- ¿Hay relación entre la puntuación que los usuarios atribuyen a la ubicación del hotel respecto al número de restaurantes y atracciones cercanas?

A parte, se intentará generar algún modelo de clasificación para predecir los rangos de precios de los hoteles según el resto de características.

2 Integración y selección de los datos de interés a analizar

El primer paso que debemos hacer es cargar el dataset que se va a utilizar. Lo leemos y utilizamos la función `str` para ver su estructura y una muestra de los datos.

```
library(stringr)
library(ggplot2)
library(gridExtra)

df <- read.csv("tripadvisor_barcelona_hotels.csv", encoding = "UTF-8")
str(df, strict.width="wrap")
```

```
## 'data.frame':    1619 obs. of  50 variables:
## $ name : chr "Travelodge Barcelona Fira" "Four Points by Sheraton Barcelona
##    Diagonal" "Hotel Miramar Barcelona" "W Barcelona" ...
## $ stars : num 3 3 5 5 3 NA 4 3 4 4.5 ...
## $ score : num 3.5 4.5 4 4.5 4.5 4 5 4.5 4 5 ...
## $ score_location : num 4 4 4.5 4.5 4 4.5 5 4.5 4.5 5 ...
## $ score_cleaning : num 4.5 4.5 4.5 4.5 4.5 4 5 4.5 4.5 5 ...
## $ score_service : num 4 4.5 4 4.5 4.5 4 4.5 4.5 4 5 ...
## $ score_value_money : num 4 4.5 4 4 4.5 4.5 4.5 4.5 4 4.5 ...
## $ ranking : num 354 150 198 243 134 80 10 92 313 6 ...
## $ price : num NA 76 184 323 143 NA NA 98 126 160 ...
## $ price_range : chr "52 \200 - 166 \200" "79 \200 - 195 \200" "152 \200 -
##    283 \200" "287 \200 - 813 \200" ...
## $ opinions : num 234 1574 1298 7751 1404 ...
## $ opinions_excellent : num 77 685 708 4598 693 ...
## $ opinions_good : num 72 692 339 1789 566 ...
## $ opinions_normal : num 49 137 107 671 103 29 17 127 200 13 ...
## $ opinions_bad : num 19 31 66 329 31 11 4 21 56 4 ...
## $ opinions_awful : num 17 30 78 364 11 24 2 15 51 3 ...
## $ num_qa : num 15 99 28 182 71 10 23 44 34 94 ...
## $ nearby_restaurants : num 66 170 55 18 468 388 454 566 668 594 ...
## $ nearby_attractions : num 7 11 20 2 48 50 113 111 275 296 ...
## $ zone : chr "Barcelona" "Poblenou / Sant Martí" "Montjuic / Sants-Montjuïc"
##    "Ciutat Vella / El Port Vell / Barceloneta" ...
## $ latitude : num 41.4 41.4 41.4 41.4 41.4 ...
## $ longitude : num 2.13 2.2 2.17 2.19 2.15 ...
## $ has_swimming_pool : chr "False" "False" "True" "True" ...
## $ has_bar : chr "True" "True" "True" "True" ...
## $ has_restaurant : chr "True" "True" "True" "True" ...
## $ has_breakfast : chr "True" "True" "True" "True" ...
## $ has_gym : chr "False" "True" "True" "True" ...
## $ has_reception_24h : chr "True" "True" "True" "True" ...
## $ has_ac : chr "True" "True" "True" "True" ...
## $ has_strongbox : chr "False" "True" "True" "True" ...
## $ admits_pets : chr "False" "False" "False" "True" ...
```

```
## $ rooms : num 83 154 75 473 53 16 19 64 81 101 ...
## $ suites : chr "False" "False" "False" "True" ...
## $ sea_views_rooms : chr "False" "False" "True" "True" ...
## $ non_smoking_rooms : chr "True" "True" "True" "True" ...
## $ landmarks_views_rooms : chr "False" "False" "False" "False" ...
## $ city_views_rooms : chr "False" "False" "False" "False" ...
## $ family_rooms : chr "True" "False" "False" "False" ...
## $ style : chr "Familiar" "Ecológico" "Vistas al parque" "Vistas a la bahía" ...
## $ tripadvisor_clasification: chr "51 de 100" "94 de 100" "97 de 100" "75 de
##    100" ...
## $ language_spanish : chr "True" "True" "False" "True" ...
## $ language_catalan : chr "True" "True" "True" "False" ...
## $ language_french : chr "True" "True" "True" "False" ...
## $ language_english : chr "True" "True" "True" "False" ...
## $ language_italian : chr "True" "True" "False" "False" ...
## $ language_bulgarian : chr "True" "False" "False" "False" ...
## $ language_russian : chr "False" "False" "False" "False" ...
## $ language_portuguese : chr "False" "True" "False" "False" ...
## $ prat_distance : int 7 15 11 12 12 11 13 12 12 13 ...
## $ timestamp : chr "2020-11-02 15:37:45.938143" "2020-11-02 15:37:58.762559"
##    "2020-11-02 15:38:01.677586" "2020-11-02 15:38:06.467551" ...
```

En cuanto a la selección de datos, vamos a quedarnos con aquellos que tengan un rango de precio informado, ya que como vamos a explicar será una de las variable objetivo del análisis. Además, como veremos más tarde, vamos a quedarnos solo con los hoteles que están situados en los distritos más turísticos (aquellos con más hoteles). Concretamente, nos quedaremos con los distritos de "Ciutat Vella", "Eixample", "Sant Martí", "Gràcia" y "Sarrià". Sin embargo, esta selección no se puede realizar hasta más tarde, ya que necesitamos aplicar un preprocesado a la variable `zone` para extraer el distrito.

En cuanto a los diferentes atributos, vamos a empezar eliminando las columnas `name` y `timestamp`, ya que la primera contiene el nombre del hotel, el cual no es necesario para los análisis y el segundo la fecha de la captura de los datos, que tampoco nos aporta información relativa al hotel.

Observamos también dos columnas relativas al precio. `price` contiene el precio por noche en el momento de la extracción del dato proporcionado por terceros (Booking, Expedia...). `price_range` es una estimación más general del precio del hotel y es proporcionado por la propia Tripadvisor. Además, la variable `price`, a parte de tener un carácter temporal, tiene muchos valores nulos, como veremos más tarde. Por este motivo, eliminaremos la variable `price` y obtendremos los precios directamente de la variable `price_range`.

```
# eliminamos las 3 variables mencionadas
df <- df[, !(names(df) %in% c("name", "timestamp", "price"))]
# eliminamos las filas que tengan nulos en price_range (string vacío)
df <- df[df$price_range != "",]
```

3 Limpieza de los datos

3.1 Preprocesado

El primer paso será convertir el tipo de las variables. Vamos a empezar convirtiendo a tipo factor las variables categóricas.

```
catergorical_vars <- c("has_swimming_pool", "has_bar", "has_restaurant", "has_breakfast",
                       "has_gym", "has_reception_24h" , "has_ac", "has_strongbox",
                       "admits_pets", "suites", "sea_views_rooms",  "non_smoking_rooms",
```

```
                        "landmarks_views_rooms", "city_views_rooms", "family_rooms",
                        "style", "language_spanish", "language_catalan", "language_french",
                        "language_english", "language_italian", "language_bulgarian",
                        "language_russian", "language_portuguese")
for(v in catergorical_vars){
  df[[v]] <- factor(df[[v]])
}
```

A continuación, debemos estandarizar la variable `price_range`. Esta variable son rangos de precios medios. El problema es que estos rangos no están predefinidos de ninguna manera. Para solucionar este problema, vamos a transformar la variable para obtener el precio medio del intervalo y luego discretizar la variable en los valores "Barato", "Medio", "Caro" y "Lujo" según las siguientes reglas:

- **Barato:** precio $<$ 60 euros
- **Medio:** 60 $>=$ precio $<$ 150
- **Caro:** 150 $>=$ precio $<$ 300
- **Lujo:** precio $>=$ 300

A continuación, se muestra la variable antes y después de esta transformación.

```
head(df$price_range, 3)
price_range_clean <- str_replace_all(df$price_range, "€|[:space:]|\\.", "")
price_range_clean <- str_split(price_range_clean, "-")
new_price_range <- rep(NA, length(price_range_clean))
for(i in 1:length(price_range_clean)){
  range <- as.list(price_range_clean[i][[1]])
  if(range[[1]] != ""){
    m <- mean(c(strtoi(range[[1]]), strtoi(range[[2]])))
    new_price_range[i] <- m
  }
}
df$price_range <- new_price_range
head(df$price_range, 3)

# discretizar
df$price_range_disc[df$price_range < 60] <- "Barato"
df$price_range_disc[(df$price_range >= 60) & (df$price_range < 150)] <- "Medio"
df$price_range_disc[(df$price_range >= 150) & (df$price_range < 300)] <- "Caro"
df$price_range_disc[df$price_range >= 300] <- "Lujo"
df$price_range_disc <- factor(df$price_range_disc,
                              levels = c("Barato", "Medio", "Caro", "Lujo"))
head(df$price_range_disc, 3)
```

```
## [1] "52 \200 - 166 \200"  "79 \200 - 195 \200"  "152 \200 - 283 \200"
## [1] 109.0 137.0 217.5
## [1] Medio Medio Caro
## Levels: Barato Medio Caro Lujo
```

Ahora, vamos a realizar un procesado para la variable `tripadvisor_clasification`. Es una puntuación sobre 100 y está expresada como un string de la forma "88 de 100", por ejemplo. El objetivo es obtener la puntuación y convertirla a entero.

```
head(df$tripadvisor_clasification, 5)
new_class <- rep(NA, nrow(df))
for(i in 1:nrow(df)){
  cl <- df$tripadvisor_clasification[i]
```

```
  index <- str_locate(cl, " de 100")[1]
  new_class[i] <- as.integer(str_sub(cl, 1, index))
}
df$tripadvisor_clasification <- new_class
head(df$tripadvisor_clasification, 5)
```

```
## [1] "51 de 100"  "94 de 100"  "97 de 100"  "75 de 100"  "100 de 100"
## [1]  51  94  97  75 100
```

Por último, vamos a estandarizar la variable **zone**, la cual contiene la zona de Barcelona donde se encuentra el hotel. Sin embargo, vemos que esta variable no está normalizada y que según el hotel, la ubicación está más o menos detallada.

```
head(df$zone)
```

```
## [1] "Barcelona"
## [2] "Poblenou / Sant Martí"
## [3] "Montjuic / Sants-Montjuïc"
## [4] "Ciutat Vella / El Port Vell / Barceloneta"
## [5] "Sant Gervasi-Galvany"
## [6] "El Ensanche (Eixample) / Sant Antoni"
```

Si nos fijamos en esta muestra de zonas, vemos que el barrio está indicado en la primera parte (a la izquierda de la primera "/"), por lo que transformaremos esta variable para quedarnos solo con este fragmento.

```
df$zone <- factor(substr(df$zone, 1, str_locate(df$zone, " / |$") - 1))
head(df$zone)
```

```
## [1] Barcelona              Poblenou               Montjuic
## [4] Ciutat Vella           Sant Gervasi-Galvany   El Ensanche (Eixample)
## 33 Levels: Barcelona ... Vila de Gràcia
```

Sin embargo, se obtiene un número bastante elevado de barrios diferentes. Lo que haremos será crear una variable nueva a partir de esta (**district**) que indique el distrito al que pertence cada barrio.

```
df$district[df$zone %in% c("El Ensanche (Eixample)")] <- "Eixample"
df$district[df$zone %in% c("Sants-Montjuïc", "La Marina de Port",
                           "Montjuic")] <- "Sants-Montjuïc"
df$district[df$zone %in% c("Les Corts de Sarrià", "Pedralbes",
                           "La Maternitat i Sant Ramon")] <- "Les Corts"
df$district[df$zone %in% c("Sarrià", "Sant Gervasi-Galvany", "Sant Gervasi-La Bonanova",
                           "El Putxet i el Farró", "Les Tres Torres")] <- "Sarrià"
df$district[df$zone %in% c("Vallcarca i els Penitents", "El Coll", "La Salut",
                           "Vila de Gràcia",
                           "El Camp d'en Grassot i Gràcia Nova")] <- "Gràcia"
df$district[df$zone %in% c("La Vila Olímpica del Poblenou",
                           "El Parc i la Llacuna del Poblenou",
                           "Poblenou", "Diagonal Mar i el Front Marítim del Poblenou",
                           "El Clot", "Provençals del Poblenou",
                           "Sant Martí")] <- "Sant Martí"
df$district[df$zone %in% c("Sant Andreu de Palomar", "Sant Andreu", "La Sagrera",
                           "El Bon Pastor")] <- "Sant Andreu"
df$district[df$zone %in% c("Ciutat Vella")] <- "Ciutat Vella"
df$district[df$zone %in% c("Nou Barris")] <- "Nou Barris"
df$district[df$zone %in% c("La Vall d'Hebron", "Horta-Guinardó" )] <- "Horta-Guinardó"
df$district <- factor(df$district)
summary(df$district)
```

```
##   Ciutat Vella       Eixample         Gràcia Horta-Guinardó      Les Corts
##            268            440             72             18             33
##     Nou Barris    Sant Andreu     Sant Martí Sants-Montjuïc         Sarrià
##              4              6             90            105             62
##           NA's
##             15
```

De esta forma, tenemos datos más manejables y hemos conseguido reducir mucho el número de posibles valores para esta variable.

Como hemos menciando antes, vamos a quedarnos con los distritos que tengan más hoteles, ya que son los que potencialmente serán mas interesantes para los turistas. Por tanto, vamos a quedarnos con los hoteles que estén en "Ciutat Vella", "Eixample", "Sant Martí", "Gràcia" y "Sarrià".

```
df <- df[df$district %in% c("Ciutat Vella", "Eixample", "Sant Martí", "Gràcia", "Sarrià",
                            "Sants-Montjuïc"),]
```

3.2 Elementos nulos

A continuación, vamos a inspeccionar el dataset en busca de valores nulos.

```
colSums(is.na(df))
```

```
##                    stars                    score           score_location
##                      146                       97                      201
##           score_cleaning           score_service        score_value_money
##                      197                      180                      214
##                  ranking              price_range                 opinions
##                       98                        0                       96
##       opinions_excellent           opinions_good          opinions_normal
##                       97                       97                       97
##             opinions_bad           opinions_awful                  num_qa
##                       97                       97                       96
##       nearby_restaurants       nearby_attractions                     zone
##                        0                        0                        0
##                 latitude                longitude        has_swimming_pool
##                        0                        0                        0
##                  has_bar           has_restaurant            has_breakfast
##                        0                        0                        0
##                  has_gym        has_reception_24h                   has_ac
##                        0                        0                        0
##            has_strongbox             admits_pets                    rooms
##                        0                        0                      218
##                   suites          sea_views_rooms        non_smoking_rooms
##                        0                        0                        0
##     landmarks_views_rooms         city_views_rooms             family_rooms
##                        0                        0                        0
##                    style tripadvisor_clasification         language_spanish
##                        0                        0                        0
##          language_catalan          language_french         language_english
##                        0                        0                        0
##          language_italian       language_bulgarian         language_russian
##                        0                        0                        0
##       language_portuguese            prat_distance         price_range_disc
##                        0                        0                        0
```

```
##                   district
##                          0
```

Dependiendo de la variable, vamos a tratar los nulos de forma diferente.

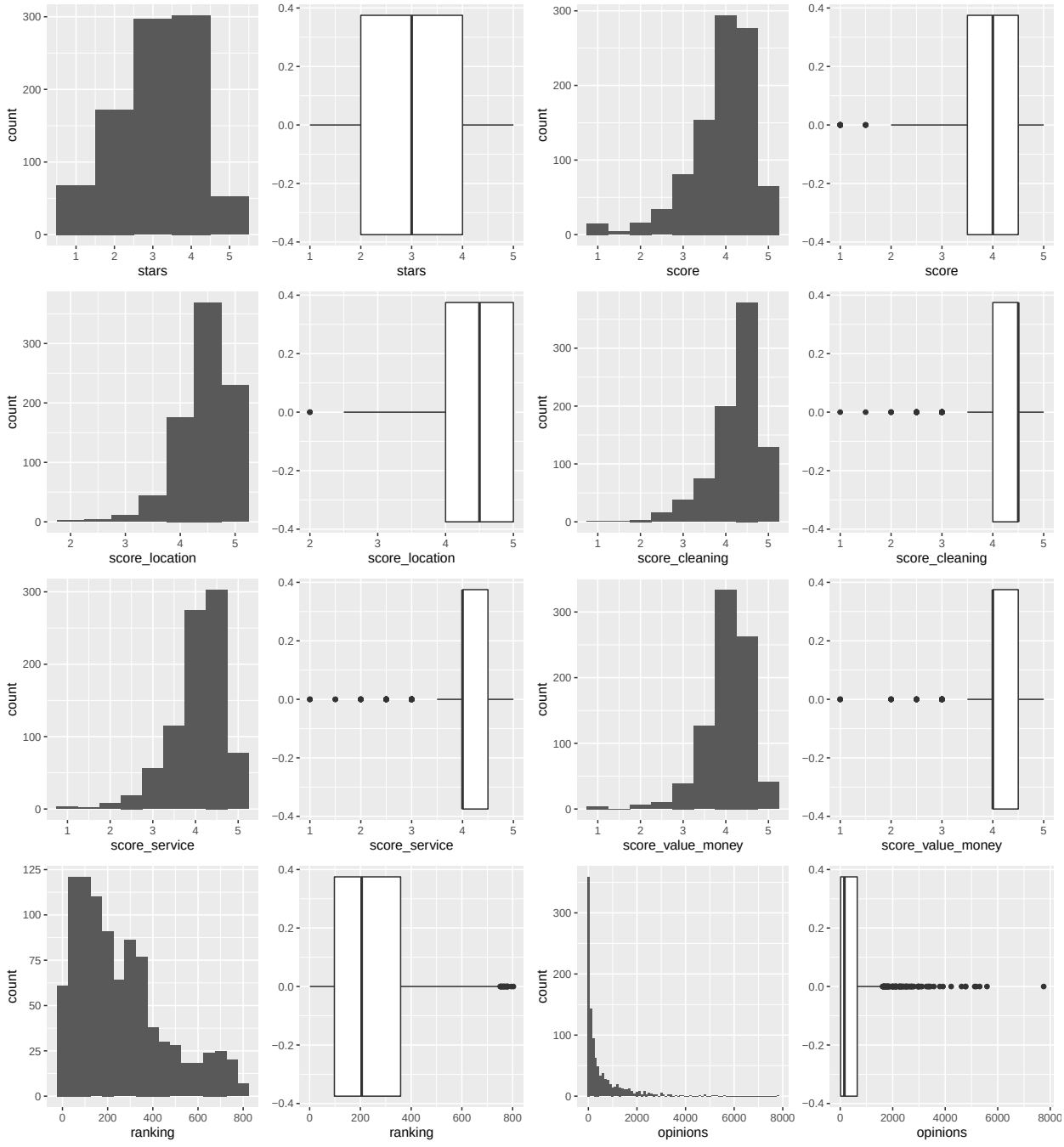
- Las que hacen referencia al número de opiniones las vamos a imputar con un 0, ya que si es nulo es que no se ha encontrado ninguna opinión.
- De momento no vamos a imputar el resto de nulos. Será en el momento que necesitemos usar estas columnas cuando eliminaremos aquellas filas que tengan NA's, ya que debido a que en algunas columnas el porcentaje de nulos está cerca del 20%, es mejor eliminarlas para no desvirtuar las posibles conclusiones. Sin embargo, no las vamos a eliminar ahora, ya que pueden tener información importante para el análisis del resto de variables.

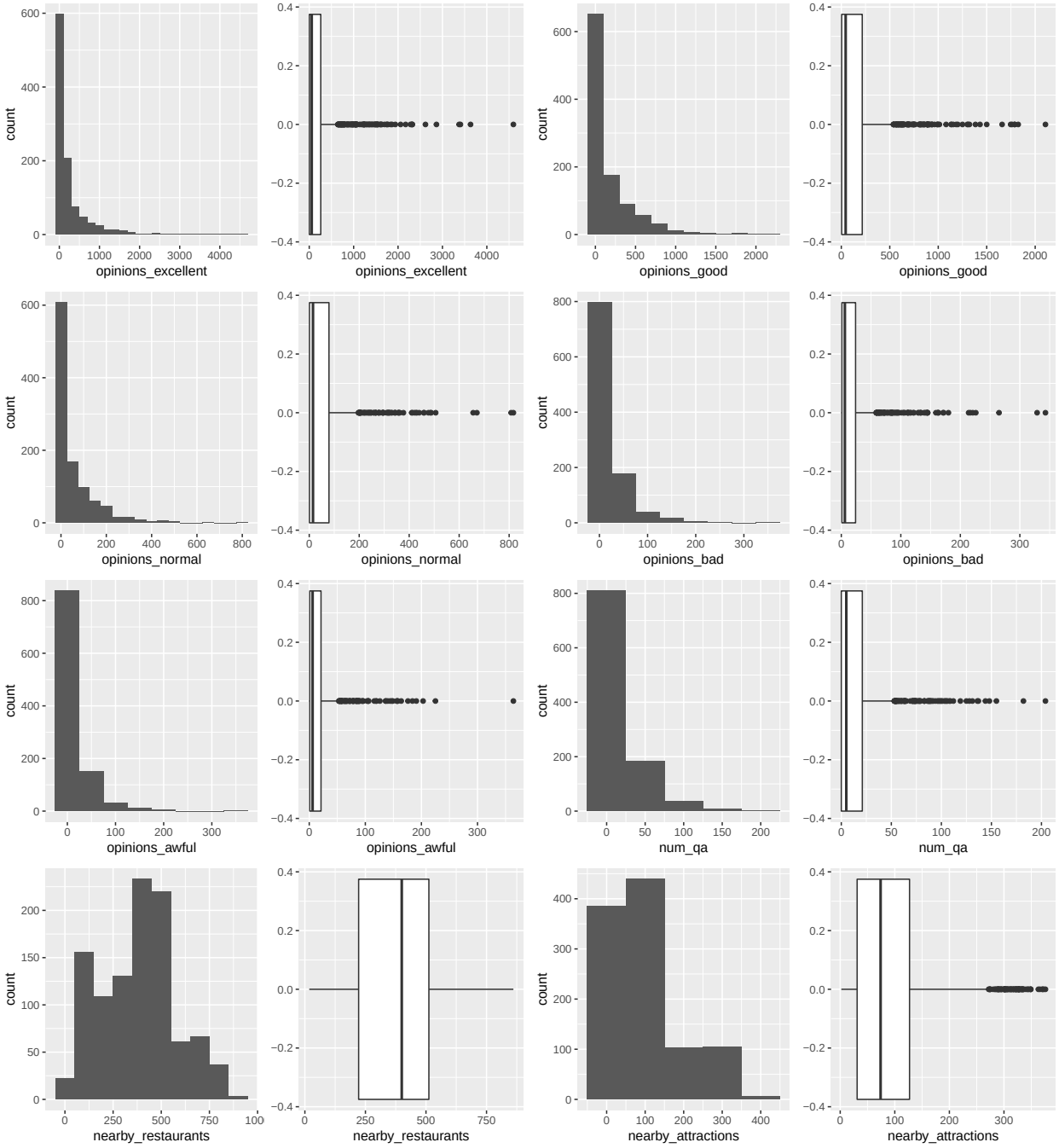
```
# opinión
opinion_vars <- c("opinions", "opinions_excellent", "opinions_good", "opinions_normal",
                  "opinions_bad", "opinions_awful", "num_qa")
for(var in opinion_vars){
  df[[var]][is.na(df[[var]])] <- 0
}
colSums(is.na(df))
```

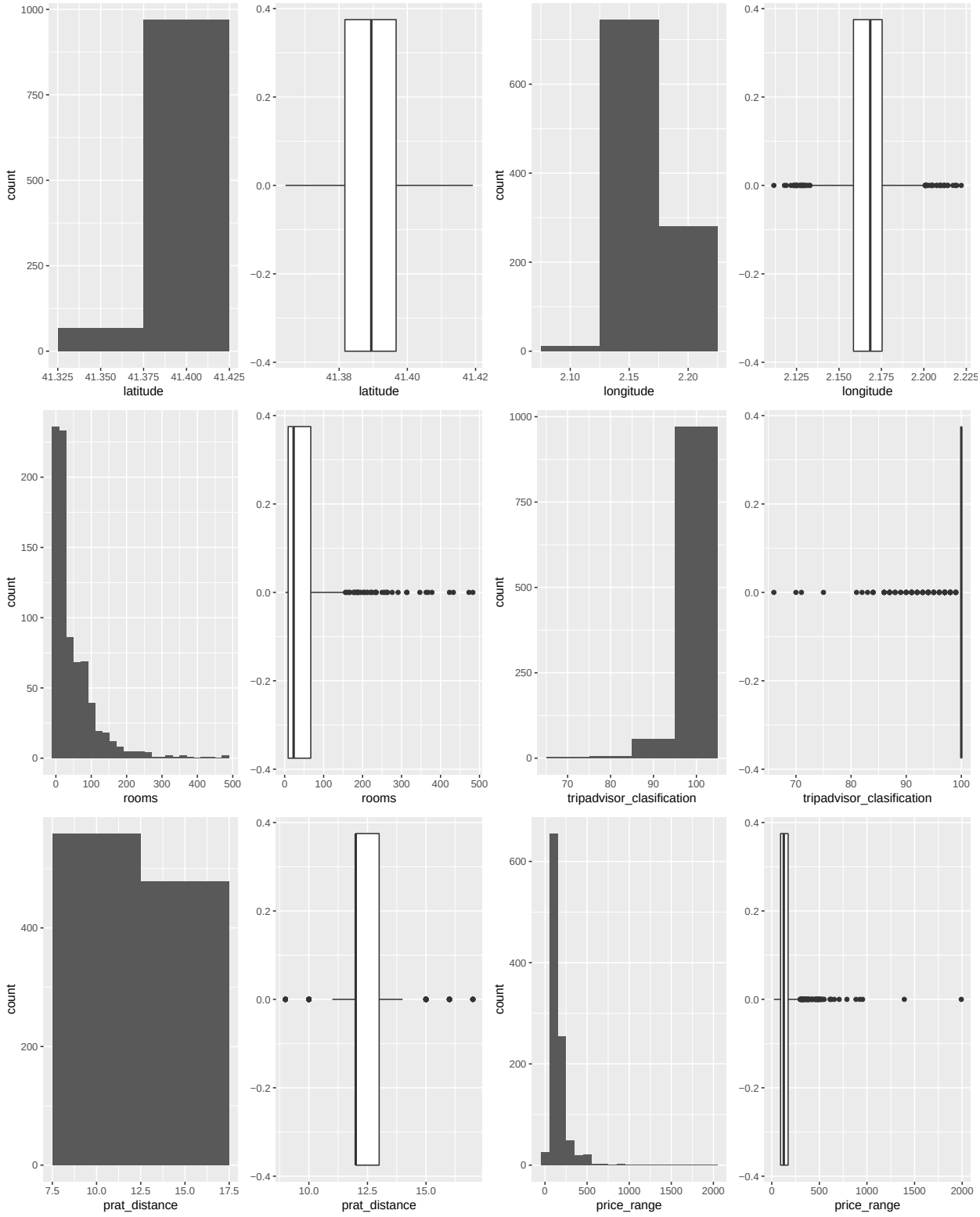
```
##                     stars                     score          score_location
##                       146                        97                       201
##            score_cleaning             score_service         score_value_money
##                       197                       180                       214
##                   ranking               price_range                  opinions
##                        98                         0                         0
##        opinions_excellent             opinions_good          opinions_normal
##                         0                         0                         0
##              opinions_bad            opinions_awful                    num_qa
##                         0                         0                         0
##        nearby_restaurants        nearby_attractions                      zone
##                         0                         0                         0
##                  latitude                 longitude         has_swimming_pool
##                         0                         0                         0
##                   has_bar            has_restaurant             has_breakfast
##                         0                         0                         0
##                   has_gym         has_reception_24h                    has_ac
##                         0                         0                         0
##             has_strongbox               admits_pets                     rooms
##                         0                         0                       218
##                    suites           sea_views_rooms         non_smoking_rooms
##                         0                         0                         0
##     landmarks_views_rooms          city_views_rooms              family_rooms
##                         0                         0                         0
##                     style tripadvisor_clasification          language_spanish
##                         0                         0                         0
##          language_catalan           language_french          language_english
##                         0                         0                         0
##          language_italian        language_bulgarian          language_russian
##                         0                         0                         0
##       language_portuguese             prat_distance          price_range_disc
##                         0                         0                         0
##                  district
##                         0
```

3.3 Valores extremos

Vamos a inspeccionar las variables numéricas en busca de outliers. Nos ayudaremos de un histograma y de un boxplot de cada variable.




count
opinions_excellent
opinions_excellent
opinions_good
opinions_good
opinions_normal
opinions_normal
opinions_bad
opinions_bad
opinions_awful
opinions_awful
num_qa
num_qa
nearby_restaurants
nearby_restaurants
nearby_attractions
nearby_attractions



Como observamos, si nos fijamos en los plots que hacer referencia a las variables de estrellas y puntuaciones en diferentes ámbitos observamos una media por encima de 4 sobre 5. Los únicos outliers que vemos son en puntuaciones por debajo de las 3 estrellas en general. Sin embargo, son valores perfectamente válidos, ya que están dentro del rango correcto de valores esperado (entre 1 y 5).

En cuanto a las diferentes variables sobre número de opiniones, se observa una clara asimetría y bastantes outliers por la parte superior. Sin embargo, no parecen valores incorrectos, ya que son valores razonables que y no están demasiado alejados del resto.

Luego, para las variables de atraciones y restaurantes cercanos, latitud y langitud, o no se observan outliers o los que se observan están muy cerca de los extremos, por lo que parecen valores perfectamente válidos.

También, en las variables `rooms` y `tripadvisor_classification`, vemos que los valores extremos parecen correctos y tienen sentido. Por ejemplo, en cuanto a esta última variable (la cual está sobre 100) vemos que la mayoría de hoteles tienen una puntuación muy alta y que todos están entre 50 y 100, valores que son correctos. Además, en la variable `prat_distance`, pese a haber outliers vemos que todos los valores están entre 10 y 17 kilómetros, distancias verosímiles para hoteles situados en los distritos mencionados de Barcelona.

Finalmente, en la variable `price_range`, vemos dos outliers claros, situados por encima de los 1250 euros. Los eliminaremos para que no afecten a las conclusiones de los análisis.

```
df <- df[df$price_range < 1000,]
```

Por lo tanto, en este punto ya tenemos el dataset preparado para su análisis.

4 Análisis de los datos

4.1 Comparación de la limpieza entre hoteles de diferente rango de precios

En este primer análisis vamos a ver si la calificación de limpieza de un hotel es superior a medida que se aumenta el rango de precios. Lo que se espera es que a cuanto más caro sea el hotel, mejor sea su limpieza, especialmente si estamos dispuestos a pagar una cantidad alta por el alojamiento, donde se espera una limpieza muy buena. Por tanto, se espera que los hoteles de precio medio tengan mejor puntuación en limpieza que los baratos, los caros mejor que los medios y los de lujo superior que en los caros.

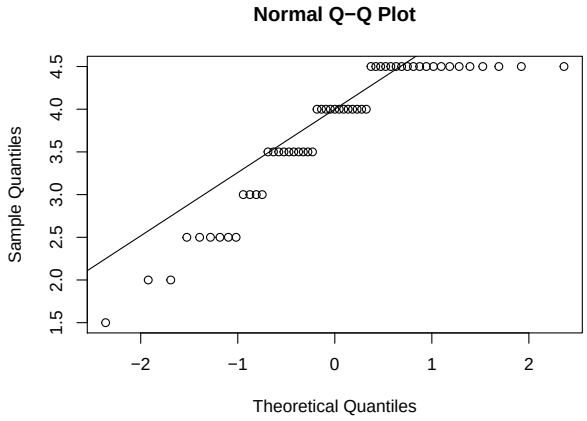
Veamos primero cuantos hoteles hay de cada tipo.

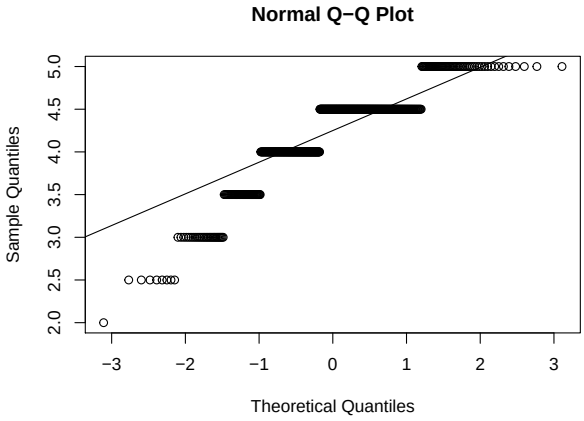
```
table(df$price_range_disc)
```

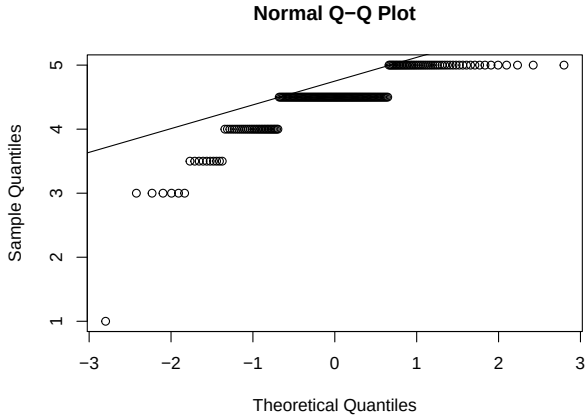
```
##
## Barato  Medio   Caro   Lujo
##     65    611    289     70
```

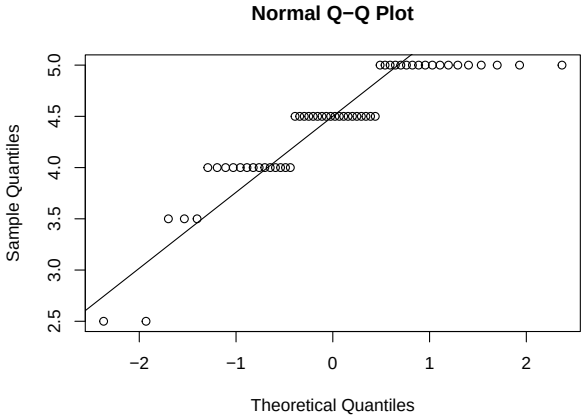
```
cleaning_cheap <- df$score_cleaning[df$price_range_disc == "Barato"]
cleaning_medium <- df$score_cleaning[df$price_range_disc == "Medio"]
cleaning_expensive <- df$score_cleaning[df$price_range_disc == "Caro"]
cleaning_luxury <- df$score_cleaning[df$price_range_disc == "Lujo"]
```

El primer paso será ver si la variable `price_range_disc` sigue o no una distribución normal. Para ello, vamos a visualizar los QQ Plots de cada grupo y hacer el Test de Shapiro-Wilk.










```
##
##  Shapiro-Wilk normality test
##
## data:  cleaning_cheap
## W = 0.84818, p-value = 6.006e-06
##
##
##  Shapiro-Wilk normality test
##
## data:  cleaning_medium
## W = 0.86049, p-value < 2.2e-16
##
##
##  Shapiro-Wilk normality test
##
## data:  cleaning_expensive
## W = 0.77327, p-value = 4.464e-16
##
##
##  Shapiro-Wilk normality test
##
## data:  cleaning_luxury
## W = 0.8245, p-value = 1.157e-06
```

Ya vemos en los plots (`cleaning_cheap`, `cleaning_medium`, `cleaning_expensive` y `cleaning_luxury`, respectivamente) que no se cumple la normalidad en los datos. La misma conclusión obtenemos con el Shapiro test, en los cuales obtenemos *p-values* muy inferiores a 0.05, con lo que se rechaza la hipótesis nula y se concluye que no siguen una distribución normal.

Comprobaremos también si las varianzas de los distintos grupos son iguales o no. Para ello aplicaremos el F-Test.

```
var.test(cleaning_cheap, cleaning_medium)
var.test(cleaning_medium, cleaning_expensive)
var.test(cleaning_expensive, cleaning_luxury)
```

```
##
##  F test to compare two variances
##
## data:  cleaning_cheap and cleaning_medium
## F = 2.1921, num df = 54, denom df = 532, p-value = 1.234e-05
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  1.519306 3.376596
## sample estimates:
## ratio of variances
##           2.192069
##
##
##  F test to compare two variances
##
## data:  cleaning_medium and cleaning_expensive
## F = 1.0531, num df = 532, denom df = 194, p-value = 0.6775
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.8288988 1.3209768
## sample estimates:
## ratio of variances
##           1.053119
##
##
##  F test to compare two variances
##
## data:  cleaning_expensive and cleaning_luxury
## F = 0.84246, num df = 194, denom df = 55, p-value = 0.3985
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.5358033 1.2583679
## sample estimates:
## ratio of variances
##          0.8424569
```

Vemos que las varianzas entre las puntuaciones de limpieza de hoteles baratos y medios son diferentes, mientras que entre medios y caros y caros y de lujo son iguales, según los *p-values* obtenidos.

Por tanto, para comparar las diferentes muestas, debemos recurrir a pruebas no paramétricas. Concretamente, aplicaremos el test de suma de rangos de Wilcoxon con las siguientes hipótesis.

$H_0 : \mu_{cheap} - \mu_{medium} = 0$
$H_1 : \mu_{cheap} - \mu_{medium} < 0$

$H_0 : \mu_{medium} - \mu_{expensive} = 0$
$H_1 : \mu_{medium} - \mu_{expensive} < 0$

$H_0 : \mu_{expensive} - \mu_{luxury} = 0$
$H_1 : \mu_{expensive} - \mu_{luxury} < 0$

```
wilcox.test(cleaning_cheap, cleaning_medium, alternative = "less")
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  cleaning_cheap and cleaning_medium
## W = 9649, p-value = 4.823e-06
## alternative hypothesis: true location shift is less than 0
```

```
wilcox.test(cleaning_medium, cleaning_expensive, alternative = "less")
```

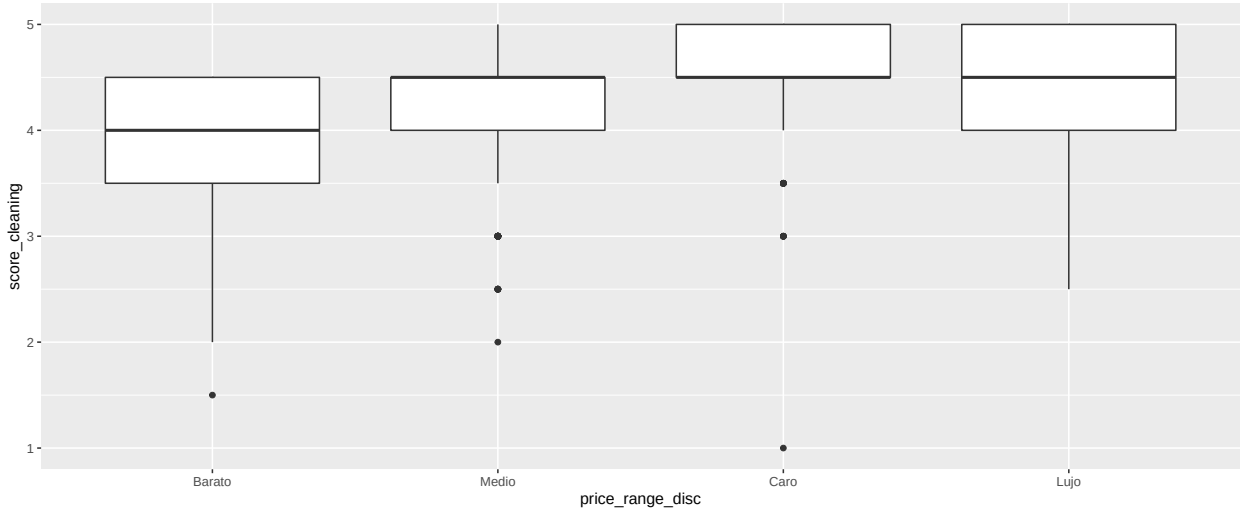
```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  cleaning_medium and cleaning_expensive
## W = 39156, p-value = 2.763e-08
## alternative hypothesis: true location shift is less than 0
```

```
wilcox.test(cleaning_expensive, cleaning_luxury, alternative = "less")
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  cleaning_expensive and cleaning_luxury
## W = 5533.5, p-value = 0.5656
## alternative hypothesis: true location shift is less than 0
```

Obtenemos *p-value* menores que 0.05 para las dos primeras pruebas, por lo que rechazamos la hipótesis nula y conlcuimos que los hoteles de precio medio tienen mejor limpieza que los baratos y que los caros tienen mejor limpieza que los medios. Sin embargo, no podemos rechazar la hipótesis nula para el último contraste, por lo que los hoteles de lujo no tienen mejor limpieza que los caros. Es algo llamativo, ya que en principio deberían tener la mejor limpieza posible.

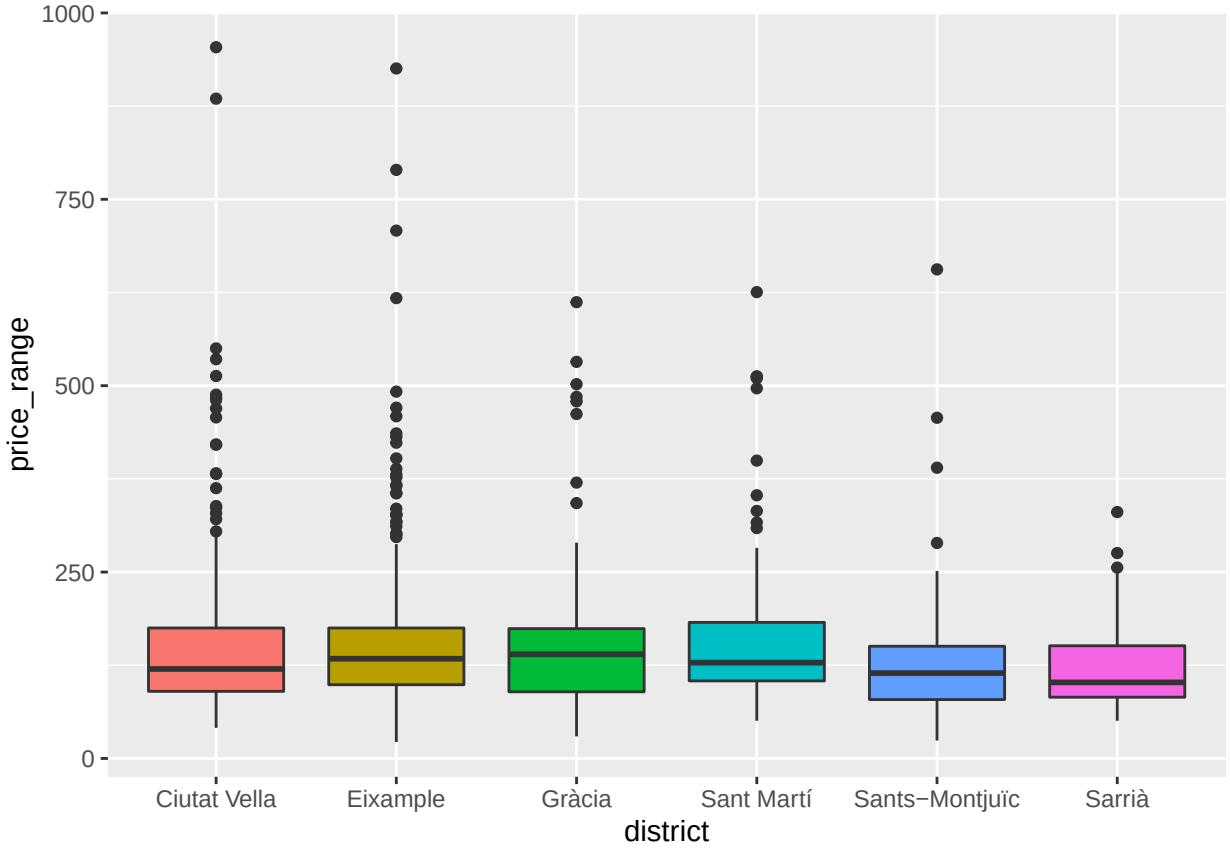
Si visualizamos la tabla con las medias de puntuación de limpieza y los boxplots para los diferentes rangos de precios, observamos las mismas conclusiones extraidas de las pruebas anteriores.



Barato	Medio	Caro	Lujo
3.736364	4.216698	4.433333	4.410714

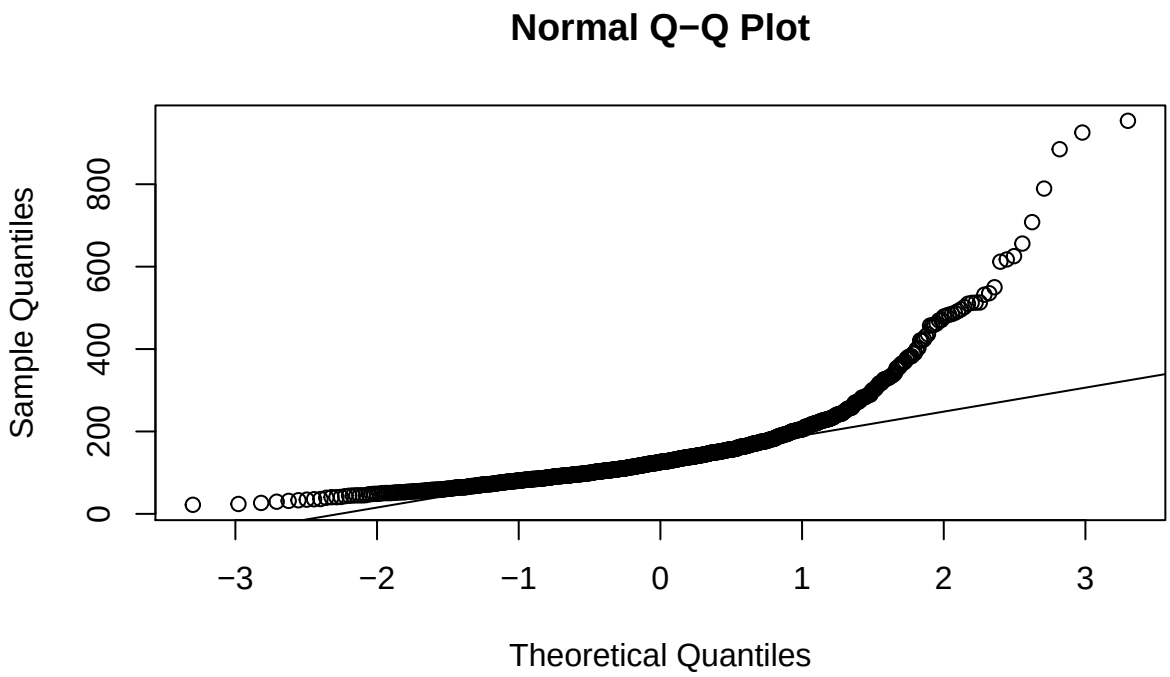
4.2 Comparación de precios entre hoteles de distintos distritos

En este apartado vamos a comprobar si hay diferencia de precios entre diferentes barrios de Barcelona. Como en muchas zonas no hay demasiados hoteles, vamos a comparar solo algunos distritos que aglomeren más hoteles. Para tener una idea a priori de cada barrio, vamos a visualizar los boxplots del precio en cada distrito.



Ya observamos que visualmente no se observan diferencias importantes entre los diferentes distritos. Solo vemos que el distrito de Sant Martí parece ser ligeramente más caro que el resto. Para salir de dudas, vamos a realizar los contrastes con todas las combinaciones de distritos, para ver si hay realmente diferencias o no entre los precios de los hoteles entre los diferentes distritos.

Para ver qué test aplicamos, debemos comprobar antes si el precio sigue una distribución normal.




```
shapiro.test(df$price_range)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  df$price_range
## W = 0.72408, p-value < 2.2e-16
```

A partir del plot y el Shapiro-Wilk vemos que no se cumple la normalidad en los datos. Vamos a comprobar también si las varianzas son iguales entre los distritos.

```
districts <- c("Sants-Montjuïc", "Ciutat Vella", "Eixample", "Sant Martí", "Gràcia", "Sarrià")
pairs <- combn(districts, 2)
for(i in 1:dim(pairs)[2]){
  dist1 <- pairs[1, i]; dist2 <- pairs[2, i]
  price1 <- df$price_range[df$district == dist1]
  price2 <- df$price_range[df$district == dist2]
  p <- var.test(price1, price2)$p.value
  cat(paste(dist1, "-", dist2, ":", p, "____ <0.05? ", p < 0.05, "\n"))
}
```

```
## Sants-Montjuïc - Ciutat Vella : 0.000344674957819093 ____ <0.05?  TRUE
## Sants-Montjuïc - Eixample : 0.159065441428888 ____ <0.05?  FALSE
## Sants-Montjuïc - Sant Martí : 0.00624859530621688 ____ <0.05?  TRUE
## Sants-Montjuïc - Gràcia : 0.000261060370197432 ____ <0.05?  TRUE
## Sants-Montjuïc - Sarrià : 0.00763647039456528 ____ <0.05?  TRUE
## Ciutat Vella - Eixample : 0.00028709744856048 ____ <0.05?  TRUE
## Ciutat Vella - Sant Martí : 0.751046297545528 ____ <0.05?  FALSE
## Ciutat Vella - Gràcia : 0.3583699002872 ____ <0.05?  FALSE
## Ciutat Vella - Sarrià : 6.34376391417391e-08 ____ <0.05?  TRUE
## Eixample - Sant Martí : 0.0341814204407929 ____ <0.05?  TRUE
```

```
## Eixample - Gràcia : 0.000879866830068676 ____ <0.05?  TRUE
## Eixample - Sarrià : 7.53277360843185e-05 ____ <0.05?  TRUE
## Sant Martí - Gràcia : 0.314802547160244 ____ <0.05?  FALSE
## Sant Martí - Sarrià : 2.0314200792626e-06 ____ <0.05?  TRUE
## Gràcia - Sarrià : 5.79652967847011e-08 ____ <0.05?  TRUE
```

Observamos que en todos los casos menos para los pares Sants-Montjuïc - Eixample y Ciutat Vella - Sant Martí las varianzas son diferentes. Lo que haremos será realizar contrastes de hipótesis en los que vamos a comprobar si los precios son iguales o no entre cada par de distritos (test bilateral). Debido a las condiciones de la muestra lo que haremos será aplicar de nuevo el test de suma de rangos de Wilcoxon con las siguientes hipótesis. Vamos a imprimir cada pareja de distritos con el *p-value* obtenido y un mensaje que diga si el valor es menor que 0.05 para poder rechazar la hipótesis nula.

$H_0 : \mu_{dist1} - \mu_{dist2} = 0$
$H_1 : \mu_{dist1} - \mu_{dist2} \neq 0$

```
for(i in 1:dim(pairs)[2]){
  dist1 <- pairs[1, i]; dist2 <- pairs[2, i]
  price1 <- df$price_range[df$district == dist1]
  price2 <- df$price_range[df$district == dist2]
  p <- wilcox.test(price1, price2, alternative = "two.sided")$p.value
  cat(paste(dist1, "-", dist2, ":", p, "____ <0.05? ", p < 0.05, "\n"))
}
```

```
## Sants-Montjuïc - Ciutat Vella : 0.0745704834589565 ____ <0.05?  FALSE
## Sants-Montjuïc - Eixample : 0.000443445051321382 ____ <0.05?  TRUE
## Sants-Montjuïc - Sant Martí : 0.00293696445772991 ____ <0.05?  TRUE
## Sants-Montjuïc - Gràcia : 0.0581194710184431 ____ <0.05?  FALSE
## Sants-Montjuïc - Sarrià : 0.789732012828323 ____ <0.05?  FALSE
## Ciutat Vella - Eixample : 0.0204109480314707 ____ <0.05?  TRUE
## Ciutat Vella - Sant Martí : 0.0451269066811619 ____ <0.05?  TRUE
## Ciutat Vella - Gràcia : 0.394054834312706 ____ <0.05?  FALSE
## Ciutat Vella - Sarrià : 0.0566461260324774 ____ <0.05?  FALSE
## Eixample - Sant Martí : 0.673846189499283 ____ <0.05?  FALSE
## Eixample - Gràcia : 0.759673141809608 ____ <0.05?  FALSE
## Eixample - Sarrià : 0.000795248644172622 ____ <0.05?  TRUE
## Sant Martí - Gràcia : 0.533802010662576 ____ <0.05?  FALSE
## Sant Martí - Sarrià : 0.00111173906212818 ____ <0.05?  TRUE
## Gràcia - Sarrià : 0.045106928228623 ____ <0.05?  TRUE
```

Resumimos en la siguiente tabla los resultados según si hipótesis que adoptamos según los *p-values*. Se mostrará el símbolo $=$ en caso de igualdad de precios entre distritosy $\neq$ en caso de que no lo sean (hipótesis alternativa).

districts	Sants-Montjuïc	Ciutat Vella	Eixample	Sant Martí	Gràcia	Sarrià
Sants-Montjuïc	*	$=$	$\neq$	$\neq$	$=$	$=$
Ciutat Vella		*	$\neq$	$\neq$	$=$	$=$
Eixample			*	$=$	$=$	$\neq$
Sant Martí				*	$=$	$\neq$
Gràcia					*	$\neq$
Sarrià						*

Ahora, para ver qué pasa en los casos donde los precios son diferentes (ver si es mayor o menor), vamos a realizar el mismo experimento pero cambiando la hipótesis alternativa por menor y mayor.

```
## Alternative: Greater
## Sants-Montjuïc - Ciutat Vella : 0.962801850782169 ____ <0.05?  FALSE
## Sants-Montjuïc - Eixample : 0.999778852313847 ____ <0.05?  FALSE
## Sants-Montjuïc - Sant Martí : 0.998543753119059 ____ <0.05?  FALSE
## Sants-Montjuïc - Gràcia : 0.971137597167615 ____ <0.05?  FALSE
## Sants-Montjuïc - Sarrià : 0.394866006414162 ____ <0.05?  FALSE
## Ciutat Vella - Eixample : 0.989804826144839 ____ <0.05?  FALSE
## Ciutat Vella - Sant Martí : 0.977500240750202 ____ <0.05?  FALSE
## Ciutat Vella - Gràcia : 0.803348299231291 ____ <0.05?  FALSE
## Ciutat Vella - Sarrià : 0.0283230630162387 ____ <0.05?  TRUE
## Eixample - Sant Martí : 0.663354500285184 ____ <0.05?  FALSE
## Eixample - Gràcia : 0.379836570904804 ____ <0.05?  FALSE
## Eixample - Sarrià : 0.000397624322086311 ____ <0.05?  TRUE
## Sant Martí - Gràcia : 0.266901005331288 ____ <0.05?  FALSE
## Sant Martí - Sarrià : 0.000555869531064088 ____ <0.05?  TRUE
## Gràcia - Sarrià : 0.0225534641143115 ____ <0.05?  TRUE
##
##
## Alternative: Less
## Sants-Montjuïc - Ciutat Vella : 0.0372852417294783 ____ <0.05?  TRUE
## Sants-Montjuïc - Eixample : 0.000221722525660691 ____ <0.05?  TRUE
## Sants-Montjuïc - Sant Martí : 0.00146848222886495 ____ <0.05?  TRUE
## Sants-Montjuïc - Gràcia : 0.0290597355092216 ____ <0.05?  TRUE
## Sants-Montjuïc - Sarrià : 0.606408777521093 ____ <0.05?  FALSE
## Ciutat Vella - Eixample : 0.0102054740157353 ____ <0.05?  TRUE
## Ciutat Vella - Sant Martí : 0.022563453340581 ____ <0.05?  TRUE
## Ciutat Vella - Gràcia : 0.197027417156353 ____ <0.05?  FALSE
## Ciutat Vella - Sarrià : 0.971772943201024 ____ <0.05?  FALSE
## Eixample - Sant Martí : 0.336923094749641 ____ <0.05?  FALSE
## Eixample - Gràcia : 0.620490525536407 ____ <0.05?  FALSE
## Eixample - Sarrià : 0.999603717560297 ____ <0.05?  FALSE
## Sant Martí - Gràcia : 0.734215522225625 ____ <0.05?  FALSE
## Sant Martí - Sarrià : 0.999451499509306 ____ <0.05?  FALSE
## Gràcia - Sarrià : 0.977684646045041 ____ <0.05?  FALSE
```

Vamos a actualizar la matriz anterior sustituyendo los signos "$\neq$" por los que se deducen a partir de esta última prueba.

districts	Sants-Montjuïc	Ciutat Vella	Eixample	Sant Martí	Gràcia	Sarrià
Sants-Montjuïc	*	=	<	<	=	=
Ciutat Vella		*	<	<	=	=
Eixample			*	=	=	>
Sant Martí				*	=	>
Gràcia					*	>
Sarrià						*

Como resumen de la tabla, vemos que los distritos que tienen los hoteles más caros son Eixample, Sant Martí y Gràcia, mientras que los más baratos son Sants-Montjuïc, Ciutat Vella y Sarrià.

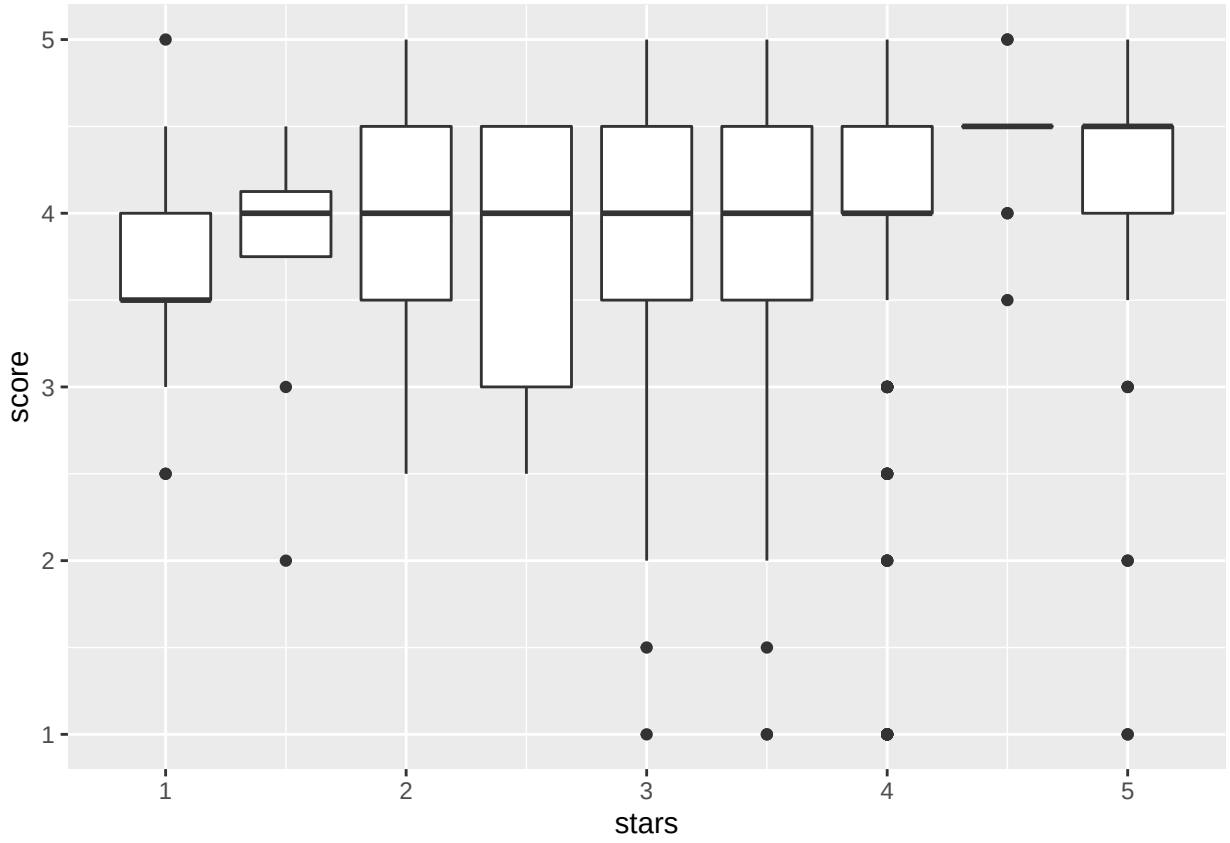
4.3 ¿Hay relación entre las estrellas de un hotel y la valoración de los usuarios?

Queremos ver si es cierto que cuantas más estrellas tenga un hotel, más puntuación le darán los usuarios. Para realizar este análisis, simplemente debemos calcular la correlación entre ambas variables. Nos ayudaremos también de un boxplot para analizar las tendencias.

```
cor(df$stars, df$score, use="complete.obs")
```

```
## [1] 0.105571
```

```
ggplot(df, aes(x = stars, y = score, group = stars)) + geom_boxplot()
```



Hemos obtenido una correlación de 0.11. Este valor indica que hay correlación positiva, aunque bastante débil. Algo similar se observa en el boxplot. A medida que aumentan las estrellas, se aprecia un leve crecimiento en el score, aunque no es demasiado claro como para afirmarlo contundentemente. Esto significa, que hay poca relación entre el aumento de estrellas del hotel y el aumento de puntuación de los clientes.

4.4 ¿Hay relación entre la puntuación que los usuarios atribuyen a la ubicación del hotel respecto al número de restaurantes y atracciones cercanas?

Muchas veces, cuando buscamos un hotel consideramos su ubicación un factor muy importante a la hora de escoger uno u otro. Muchas veces lo que se quiere es estar muy cerca de las principales atracciones de la ciudad o de restaurantes. Sin embargo, queremos comprobar si este hecho está reflejado en la puntuación que se le da a la ubicación del hotel. Para ello, vamos a ver si la puntuación que recibe la ubicación del hotel tiene relación con el número de restaurantes y atracciones situados en las inmediaciones del hotel.

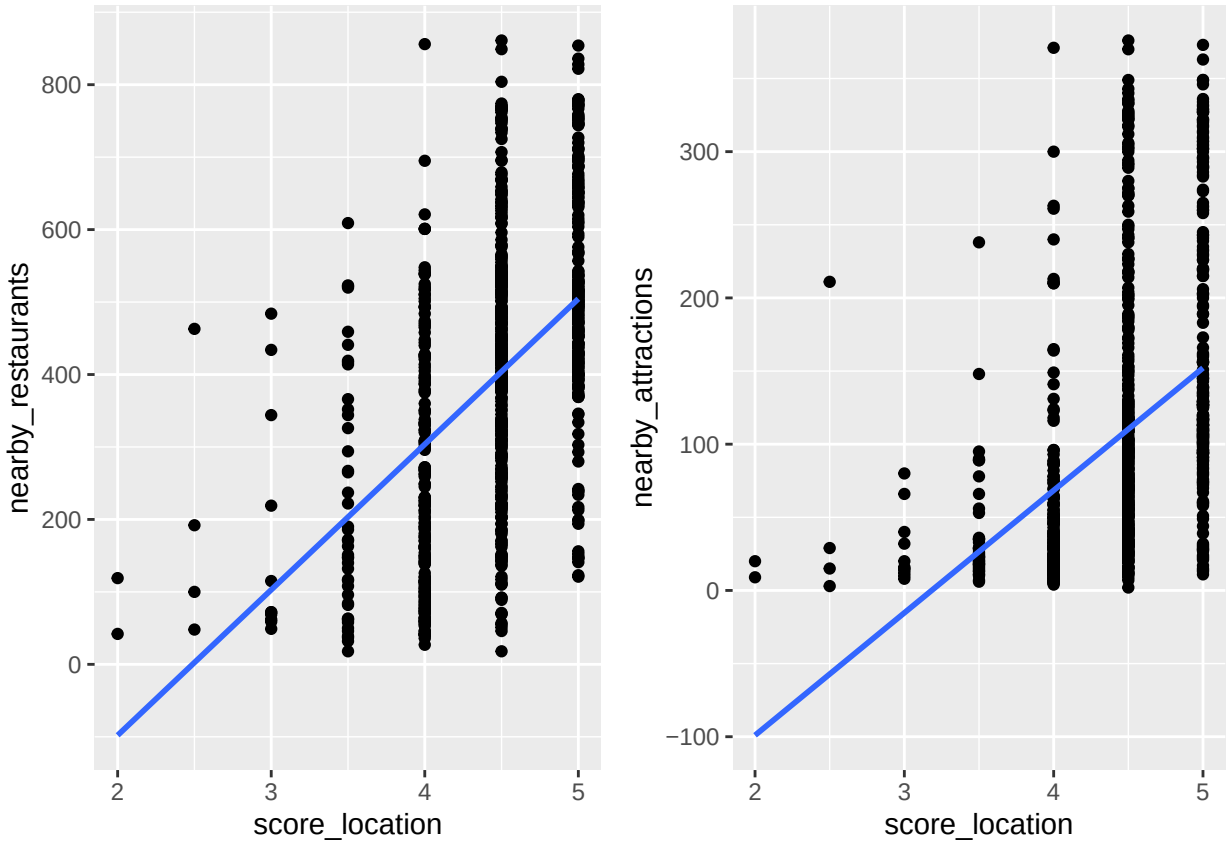
```
cor(df$score_location, df$nearby_restaurants, use="complete.obs")
```

```
## [1] 0.5039407
```

```
cor(df$score_location, df$nearby_attractions, use="complete.obs")
```

```
## [1] 0.441879
```

```
gg1 <- ggplot(df, aes(x = score_location, y = nearby_restaurants)) + geom_point() +
  geom_smooth(method = "lm", se = FALSE)
gg2 <- ggplot(df, aes(x = score_location, y = nearby_attractions)) + geom_point() +
  geom_smooth(method = "lm", se = FALSE)
grid.arrange(gg1, gg2, ncol=2)
```



Correlaciones	Num. Restaurantes	Num. Atracciones
Punt. Ubicación	0.504	0.442

Obtenemos correlaciones positivas moderadas entre la puntuación que se le da a la ubicación y el número de restaurantes y atracciones cercanas al hotel. Lo mismo vemos claramente en los *scatter plot*, donde la recta de regresión lineal tiene claramente pendiente positiva. Por tanto, vemos que cuando una persona puntúa la ubicación del hotel, indirectamente nos da información valiosa sobre el número de restaurantes y atracciones situados cerca del hotel.

4.5 ¿Son más caros aquellos hoteles que tienen aire acondicionado respecto los que no tienen?

Muchas veces los hoteles que tienen ciertas instalaciones o servicios incluyen el uso de estos en el precio. Por tanto, vamos a ver si los hoteles que disponen de aire acondicionado son más caros que los que no tienen. Para hacer una comparación lo más ajustada posible, vamos a comparar los hoteles que estén en el Eixample y que tengan entre 3 y 4 estrellas. Realizaremos también el Shapiro test para ver si cumplen la normalidad y el F Test para ver si las varianzas son iguales.

```
price_no_ac <- df$price_range[(!is.na(df$stars)) & (df$stars <= 4) &
                              (df$district == "Eixample") & (df$stars >= 3) &
                              (df$has_ac == "False")]
price_with_ac <- df$price_range[(!is.na(df$stars)) & (df$stars <= 4) &
                                (df$district == "Eixample") & (df$stars >= 3) &
                                (df$has_ac == "True")]
shapiro.test(price_no_ac)
shapiro.test(price_with_ac)
var.test(price_no_ac, price_with_ac)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  price_no_ac
## W = 0.87995, p-value = 1.328e-05
##
##
##  Shapiro-Wilk normality test
##
## data:  price_with_ac
## W = 0.57286, p-value < 2.2e-16
##
##
##  F test to compare two variances
##
## data:  price_no_ac and price_with_ac
## F = 0.37005, num df = 64, denom df = 197, p-value = 1.038e-05
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.2528460 0.5645395
## sample estimates:
## ratio of variances
##          0.3700538
```

Según los resultados, vemos que las muestras no siguen una normal y además las varianzas son diferentes.

Por tanto, volveremos a aplicar como antes el test de Wilkoxon con la hipótesis siguiente:

$H_0 : \mu_{ac} - \mu_{nac} = 0$
$H_1 : \mu_{ac} - \mu_{nac} > 0$

```
wilcox.test(price_with_ac, price_no_ac, alternative = "greater")
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  price_with_ac and price_no_ac
## W = 5485.5, p-value = 0.9629
```

```
## alternative hypothesis: true location shift is greater than 0
```

Hemos obtenido un *p-value* muy alto (0.9629), por lo que no podemos rechazar la hipótesis nula. Por tanto, no podemos afirmar que los hoteles con aire acondicionado sean más caros que los que no tienen. Para ver si es porque el aire acondicionado ya está muy extendido y es relativamente barato, vamos a probar con si tienen piscina o no, ya que en principio es un servicio que tiene un coste elevado.

```
price_no_sw <- df$price_range[(!is.na(df$stars)) & (df$stars <= 4) &
                              (df$district == "Eixample") & (df$stars >= 3) &
                              (df$has_swimming_pool == "False")]
price_with_sw <- df$price_range[(!is.na(df$stars)) & (df$stars <= 4) &
                                (df$district == "Eixample") & (df$stars >= 3) &
                                (df$has_swimming_pool == "True")]
wilcox.test(price_with_sw, price_no_sw, alternative = "greater")
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  price_with_sw and price_no_sw
## W = 4985, p-value = 0.8796
## alternative hypothesis: true location shift is greater than 0
```

Volvemos a no poder rechazar la hipótesis nula y por tanto se concluye que no son más caros los hoteles que tienen piscina. Es un resultado que no intuíamos, ya que se esperaba que los hoteles con piscina fueran más caros. Puede que el problema sea que no estamos comparando hoteles con otras características similares y que puedan ocultar el efecto de un servicio o instalación concreto del hotel, aunque se ha intentado minimizar este efecto seleccionando hoteles del mismo distirito y estrellas.

4.6 Predicción del precio de un hotel

En este apartado vamos a intentar clasificar el precio de un hotel según sus características a partir de un modelo de clasificación. El objetivo inicial era predecir el precio numérico del hotel mediante un modelo de regresión, pero debido a que el precio lo hemos obtenido a través de un intervalo y por lo tanto no es demasiado preciso, vamos a convertir el problema en uno de clasificación, en el que en lugar de obtener un valor exacto vamos a clasificar los hoteles según el rango de precios "Barato", "Medio", "Caro" y "Lujo". Como método de clasificación, utilizaremos un árbol de decisión, ya que es un algoritmo sencillo y que en general tiene buenos resultados.

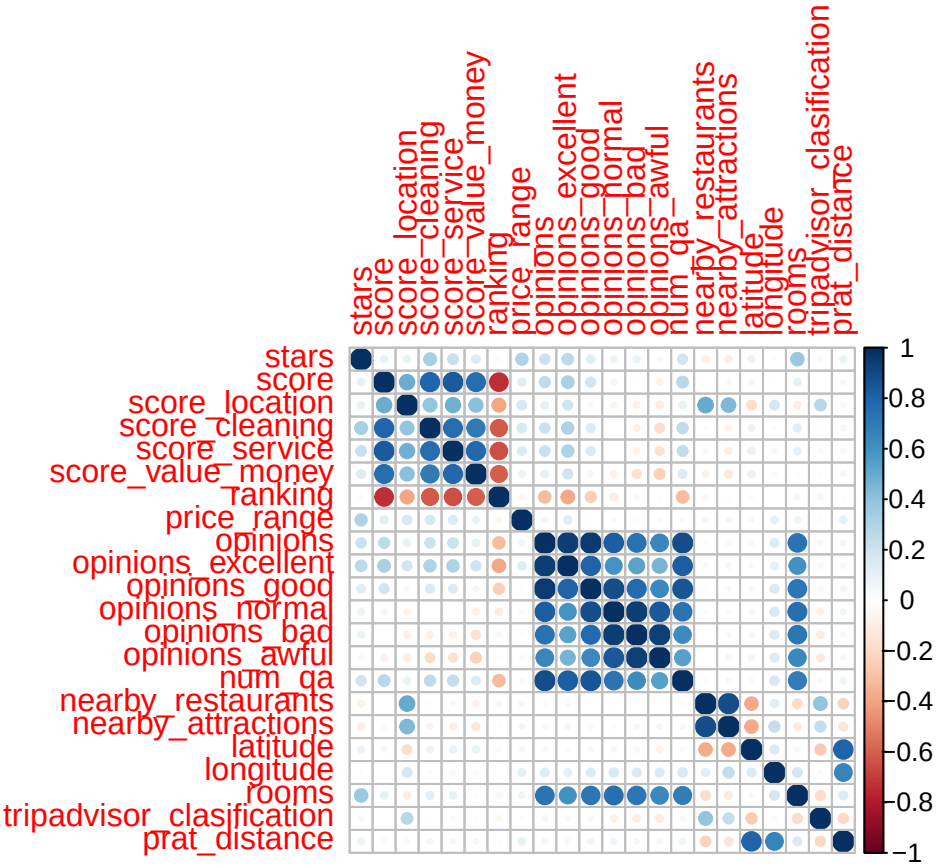
Predecir el precio de un hotel o simplemente saber en qué rango de precios de los definidos estaría un hotel puede ser muy útil para diferentes usuarios. Por ejemplo, imaginemos un turista que desea un alojamiento con unas características determinadas y ha encontrado algunos navegando por la web. Sin embargo, no sabe si los precios disponbles se adecuan a las características y servicios de los que dispone el hotel. Por tanto, el turista, siendo capaz de predecir el precio del hotel según sus atributos, podrá comparar los precios predichos con los que se le muestran, pudiendo valorar si una oferta es buena o si no lo es.

Otro caso podría ser el de una cadena que quiere abrir un hotel en algún punto de Barcelona con unas características concretas. Si predecimos el precio de este supuesto hotel, la cadena hotelera podrá determinar los precios a los que pondrá sus habitaciones, ya puede ser algo más caro que otros hoteles con características similares o bien tener un precio más reducido, para así poder competir con los demás hoteles. Por lo tanto, ya vemos que poder predecir el precio puede ser útil para diversos usuarios.

Para consturir el modelo, debemos seleccionar las variables independientes que nos ayudarán a predecir el precio. Para ello, vamos a calcular primero las correlaciones entre las diferentes variables.

```
library(corrplot)
not_num_vars <- c(catergorical_vars, "district", "zone", "price_range_disc")
```

```
corrs <- cor(df[, -which(names(df) %in% not_num_vars)], use="pairwise.complete.obs")
corrplot(corrs)
```



Observamos que no hay ninguna variable altamente correlacionada con el precio, por lo que vamos a construir diferentes modelos con variables distintas tanto numéricas como categóricas, de modo que el mejor modelo será aquel que tenga una precisión mayor. Para ello, vamos a dividir el dataset en dos trozos, uno para entrenar el modelo (training set) y el otro para el test (test set) con una proporción de 70/30.

```
set.seed(12)
df_rand <- df[sample(nrow(df)),]
X <- df_rand[, names(df_rand) != "price_range_disc"]
y <- df_rand[, "price_range_disc"]

indexes = sample(1:nrow(df), size=floor(0.7*nrow(df)))
trainX<-X[indexes,]
trainy<-y[indexes]
testX<-X[-indexes,]
testy<-y[-indexes]

m1 <- C50::C5.0(trainX[,c("stars", "nearby_restaurants")], trainy)
predicted_model <- predict(m1, testX, type="class")
ac1 <- 100*sum(predicted_model == testy) / length(predicted_model)

m2 <- C50::C5.0(trainX[,c("stars", "nearby_restaurants", "nearby_attractions")],
                trainy)
predicted_model <- predict(m2, testX, type="class")
```

```
ac2 <- 100*sum(predicted_model == testy) / length(predicted_model)

m3 <- C50::C5.0(trainX[,c("stars", "nearby_restaurants", "nearby_attractions",
                          "prat_distance", "district")], trainy)
predicted_model <- predict(m3, testX, type="class")
ac3 <- 100*sum(predicted_model == testy) / length(predicted_model)

m4 <- C50::C5.0(trainX[,c("stars", "nearby_restaurants", "nearby_attractions",
                          "prat_distance", "district", "admits_pets")], trainy)
predicted_model <- predict(m4, testX, type="class")
ac4 <- 100*sum(predicted_model == testy) / length(predicted_model)

m5 <- C50::C5.0(trainX[,c("stars", "nearby_restaurants", "latitude", "longitude",
                          "prat_distance", "district", "admits_pets",
                          "has_strongbox")], trainy)
predicted_model <- predict(m5, testX, type="class")
ac5 <- 100*sum(predicted_model == testy) / length(predicted_model)

m6 <- C50::C5.0(trainX[,c("stars", "nearby_restaurants", "nearby_attractions",
                          "latitude", "longitude", "has_swimming_pool", "has_bar",
                          "has_restaurant", "has_breakfast", "has_gym",
                          "has_reception_24h", "has_ac", "has_strongbox",
                          "admits_pets", "rooms", "suites", "sea_views_rooms",
                          "non_smoking_rooms", "landmarks_views_rooms",
                          "city_views_rooms", "family_rooms", "prat_distance",
                          "district")], trainy)
predicted_model <- predict(m6, testX, type="class")
ac6 <- 100*sum(predicted_model == testy) / length(predicted_model)

res <- data.frame(Modelo1 = ac1, Modelo2 = ac2, Modelo3 = ac3,
                  Modelo4 = ac4, Modelo5 = ac5, Modelo6 = ac6)
knitr::kable(res)
```

Modelo1	Modelo2	Modelo3	Modelo4	Modelo5	Modelo6
64.95177	64.95177	63.98714	61.09325	62.37942	65.27331

En la tabla anterior, vemos los resultados de las *accuracies* de cada uno de los árboles. Vemos que el modelo con el valor más alto es el modelo 6, el cual es el modelo que tiene más variables. No obstante, probando otras semillas, no siempre este modelo ha obtenido ha sido el mejor. Sin embargo, el *accuracy* de este modelo 6 ha sido del 65.27%, aunque en otras pruebas con semillas diferentes este porcentaje ha estado entre 55 y 65%. Veamos la matriz de confusión del mejor modelo.

	Barato	Medio	Caro	Lujo
Barato	3	16	1	0
Medio	4	167	17	0
Caro	0	67	24	0
Lujo	0	7	5	0

Vemos que la categoría que más acierta es la de rango medio, mientras que de hoteles baratos solo clasifica tres correctamente de un total de 20. De caros clasifica bien algunos, aunque la gran mayoría los ha clasificado como “Medio”. Los de lujo vemos que no los consigue clasificar, ya que los clasifica todos como medios o

caros.

En resumen, es un resultado aceptable pero tampoco demasiado bueno, ya que el porcentaje de hoteles clasificados correctamente no es excesivamente alto y muchos rangos de precios no los clasifica correctamente. La razón de este resultado es que no se están contemplando variables importantes que puedan ayudar a diferenciar los hoteles, como por ejemplo la cadena de hoteles o la calidad de otros servicios e instalaciones que no se han recogido en el dataset por motivos de dificultad del proceso de extracción o bien porque directamente no aparece la información. También podría ser problema de la calidad del dato, ya que en muchas ocasiones los rangos de precios de los que obtenemos el dato son muy amplios y pueden que no reflejen correctamente la realidad.

De todas formas, con los datos que se tiene se ha conseguido clasificar correctamente más de la mitad de los hoteles, por lo que está bastante bien para un dataset el cual se ha construido desde cero por nuestra parte sin estar preparado para obtener resultados muy precisos en tareas de clasificación.

5 Conclusiones

En la primera parte de la práctica se ha realizado un preproceso de los datos, el cual ha consistido en seleccionar los datos a analizar, limpiarlos y detectar valores nulos y extremos. La parte de limpieza ha implicado diversas tareas de formato, transformación y discretización de datos, con la cual hemos generado la variable que informa del rango de precios y que hemos intentado predecir posteriormente.

También, se han detectado algunos valores nulos. En algunas columnas referentes a número de opiniones de usuarios, estas se han inputado como 0, ya que es el significado real de los valores nulos en este tipo de variables. Para el resto de variables, hemos visto porcentajes bastante elevados de valores nulos (cerca del 20%), por lo que hemos preferido no inputar estos valores con las técnicas más habituales, como la mediana, para evitar así desvirtuar las conclusiones de los análisis. Lo que se ha hecho finalmente ha sido ignorar los registros con valores nulos en el momento que se han necesitado analizar ciertas variables, intentando obtener así conclusiones con observaciones completas.

Posteriormente, hemos analizado todas las variables en busca de valores extremos mediante histogramas y boxplots. Después de inspeccionar estos valores, solo hemos encontrado dos valores con alta probabilidad de ser outliers en la variable `price_range`, los cuales hemos eliminado para evitar que impacten en los resultados.

Ya en la parte de análisis, hemos respondido algunas preguntas sobre los hoteles. En primer lugar, se ha querido comprobar si es cierto que a cuanto más caro es un hotel, mejor es su limpieza. Hemos visto que que los hoteles de precio medio tienen mejor limpieza que los baratos y que los caros tienen mejor limpieza que los medios, pero no es cierto que los de lujo tengan mejor limpieza que los caros, hecho que no esperábamos antes de realizar el análisis.

Seguidamente, hemos querido comprobar las diferencias de precios de hoteles entre los diferentes distritos de Barcelona. Los tests realizados nos han llevado a la conclusion de que los distritos que tienen los hoteles más caros son Eixample, Sant Martí y Gràcia, mientras que los más baratos son Sants-Montjuïc, Ciutat Vella y Sarrià. Sin embargo, mirando el boxplot con la comparación entre barrios, estas diferencias parecen ser bastante pequeñas.

Además, hemos querido buscar correlaciones entre algunas variables. Primero, hemos querido validar si es cierto que cuantas más estrellas tenga un hotel, más satisfecho está el usuario (valoracion alta). El resultado ha sido una correlación bastante baja (0.11), con lo que la relacion entre ambas variables es bastante debil, significando que un hotel con muchas estrellas no tiene porqué tener valoraciones más altas que uno con menos estrellas.

Otra relación que hemos querido ver es si el número de hoteles y atracciones cercanas al hotel es motivo de tener alta puntuación de ubicación. Se ha obtenido una correlación moderada con ambas variables, concluyendo que existe relación entre el número dee atracciones y restaurantes en las inmediaciones del hotel y la puntuación obtenida por parte de los usuarios en cuanto a su ubicación.

Finalmente, se ha querido ver si existe diferencias de precios entre hoteles que disponen de algun servicio o instalación concreta respecto a otros similares que no la tengan. Para tener hoteles similares, se han utilizado los situados en el barrio de l'Eixample y que tuvieran entre 3 y 4 estrellas. Sin embargo, después de haber probado con tener o no aire acondicionado y luego piscina, hemos visto que no había diferencia en el precio entre los grupos que tenían y los que no tenian estos servicios, por lo que es probable que las diferencias de precios estén causadas por otros factores que no estamos teniendo en cuenta, como la cadena del hotel o la propia calidad de los servicios.

Por último, hemos intentado construir un modelo de clasificación de los hoteles según su rango de precios (Barato, Medio, Caro, Lujo) mediante árboles de decisión probando con distintas variables. Se ha hecho una separación de los datos en dos conjuntos: *training set* para el entrenamiento y *test set* para la prueba del modelo en datos desconocidos. Con el mejor modelo obtenido, se ha obtenido un 65.27% de *accuracy*. Es un resultado aceptable, aunque hemos visto que tiene bastantes problemas para clasificar correctamente los hoteles que no son de precio medio. Sin embargo, estamos satisfechos con el resultado obtenido, ya que con estos datos desconocidos previamente y sin saber si realmente obtendríamos algún modelo que funcionase más o menos bien, hemos conseguido predecir correctamente mas de la mitad de las observaciones que el modelo no había visto anteriormente.

Sin embargo, para resolver el problema descrito en el inicio y poder tener una herramienta totalmente últil y que funcione adecuadamente, se necesitaría un esfuerzo extra para encontrar nuevas variables que nos ayuden a predecir mejor el precio de un hotel y probar otros modelos de clasificación que funcionen mejor con este problema, aunque estas tareas podrían ser perfectamente un proyecto serio para una empresa, debido a los recursos necesarios para conseguir una precisión óptima del modelo.

6 Contribuciones

Contribuciones	Firma
Investigación previa	Alberto Giménez
Redacción de las respuestas	Alberto Giménez
Desarrollo código	Alberto Giménez