Práctica 2: Booking Data Cleaning

Gerard Alcalde and Guillem Rochina

2022-12-24

## 1. DESCRIPCIÓN DEL DATASET

### ¿Por qué es importante y qué pregunta/problema pretende responder?

#### 1.1. Descripción general

El conjunto de datos ha tratar es el obtenido como resultado de la práctica anterior de Web Scraping. El dataset presenta los datos e indicadores más relevantes de cada uno de los hoteles encontrados en función de determinados criterios de búsqueda (ciudad, fecha de check-in, fecha de check-out, número de adultos, niños y habitaciones). En este dataset en concreto presentamos datos para distintas fechas (diciembre, marzo y junio), ciudades (barcelona, madrid, valencia), número de adultos, niños y habitaciones a fin de tener una muestra más amplia e informativa que la que nos daría una búsqueda con tan sólo unos parámetros fijos.

A continuación, se realiza una breve descripción de cada una de las variables disponibles antes de realizar la limpieza:

* **Name (str)**: Nombre del hotel.
* **City (str)**: Nombre de la ciudad donde se realiza la búsqueda.
* **Check-in (str)**: Fecha de entrada al hotel.
* **Check-out (str)**: Fecha de salida del hotel.
* **Adults (int)**: Nº de adultos para los que se realiza la reserva.
* **Num\_rooms (int)**: Nº de habitaciones reservadas.
* **Address (str)**: Dirección postal del hotel, nombre de calle, barrio de la ciudad, código postal, etc.
* **Hotel\_coordinates (str)**: Latitud y longitud de la ubicación del hotel.
* **Hotel\_score (int)**: Nota general que recibe el hotel por los usuarios.
* **Hotel\_scores (dict)**: Puntuaciones de cada una de las dimensiones que puede valorar un usuario.
* **Hotel description (list)**: Descripción aportada por el propietario del hotel
* **Features (list)**: Lista de servicios añadidos del hotel.
* **Room\_data (dict)**: Conjunto de diccionarios que recogen las características de cada una de las habitaciones disponibles en el hotel.
* **Page\_count (int)**: Posición en la que ha aparecido el hotel en el buscador.
* **Current\_page (int)**: Página en la que ha aparecido el hotel en el buscador.

#### 1.2. Importancia y problema a responder

Aparecer en los primeros resultados de una búsqueda de un usuario por internet se ha convertido en los últimos años en una de las preocupaciones principales de cualquier negocio que busca hacerse un hueco en el mundo digital. Este fenómeno explica la proliferación de herramientas como el SEO y el SEM, mediante las cuales se intenta innovar (más allá del pago de tarifas para aparecer en las primeras páginas de búsqueda) a fin de lograr un mejor “posicionamiento” en buscadores como Google o Bing. El negocio hotelero no es una excepción en este caso.

No obstante, el uso de dichas herramientas no resulta útil si nos enfrentamos a un buscador dentro de una página web, como el caso que nos ocupa con Booking.com. ¿Qué hacer entonces? ¿Cómo mejorar el posicionamiento dentro de la web?

En la web del metabuscador de alojamientos encontramos que define a dichas posiciones entre los resultados como “ranking”, descrito por ellos mismos como el orden en el que se muestran los alojamientos disponibles en los resultados de búsqueda. En su blog “Booking.com Partner Hub” indican que los resultados son ordenados según la relevancia en base a las preferencias particulares de cada “cliente”, incluyendo en este caso también las “dinámicas del mercado”, el “rendimiento del alojamiento”, etc. Variable que poca información revela al interesado en que sus hoteles/alojamientos aparezcan en posiciones más “atractivas” para este.

El objetivo de este dataset es obtener las características que hacen que un hotel esté mejor posicionado que otro. El posicionamiento de los hoteles suele ser algo estático y por ello con una sola lectura de la página web puede ser suficiente.

En concreto, en el mismo post analizado se destaca la relevancia de revisar regularmente las “condiciones flexibles”, los comentarios, los precios externos, las promociones, la puntuación de la página del alojamiento… a fin de lograr escalar puestos en la clasificación.

Dado el impacto que puede tener en el negocio de un hotel el aparecer en posiciones más inmediatas del buscador, el presente proyecto pretende ser la continuación de la práctica 1, ofreciendo un análisis inferencial que destaque que factores tienen un mayor peso en el posicionamiento de un alojamiento en Booking.com.

## 2. LIMPIEZA DE LOS DATOS

A fin de iniciar el proceso de limpieza de datos, en primer lugar se debe realizar la lectura del fichero csv que obtuvimos en la práctica anterior. Como resultado de la función de R-base “read.csv” obtenemos un objeto data.frame, que manipularemos en la presente sección.

# Lectura y breve descripción de los datos  
  
booking <- read.csv("hotels\_data.csv", header = TRUE)  
summary(booking)

## name city check.in adults   
## Length:3912 Length:3912 Length:3912 Min. :2.000   
## Class :character Class :character Class :character 1st Qu.:2.000   
## Mode :character Mode :character Mode :character Median :4.000   
## Mean :3.329   
## 3rd Qu.:5.000   
## Max. :5.000   
##   
## children check.out num\_rooms address   
## Min. :1.000 Length:3912 Min. :2.000 Length:3912   
## 1st Qu.:1.000 Class :character 1st Qu.:2.000 Class :character   
## Median :3.000 Mode :character Median :2.000 Mode :character   
## Mean :2.475 Mean :2.268   
## 3rd Qu.:3.000 3rd Qu.:3.000   
## Max. :3.000 Max. :3.000   
##   
## hotel\_coordinates hotel\_score hotel\_scores hotel\_description   
## Length:3912 Min. :-1.000 Length:3912 Length:3912   
## Class :character 1st Qu.: 8.000 Class :character Class :character   
## Mode :character Median : 8.400 Mode :character Mode :character   
## Mean : 8.259   
## 3rd Qu.: 8.800   
## Max. :10.000   
## NA's :8   
## features room\_data page\_count current\_page   
## Length:3912 Length:3912 Min. : 0.00 Min. :1.000   
## Class :character Class :character 1st Qu.: 24.00 1st Qu.:1.000   
## Mode :character Mode :character Median : 53.00 Median :3.000   
## Mean : 57.19 Mean :2.741   
## 3rd Qu.: 88.00 3rd Qu.:4.000   
## Max. :133.00 Max. :5.000   
##   
## in\_page\_count search\_date   
## Min. : 0.00 Length:3912   
## 1st Qu.: 6.00 Class :character   
## Median :13.00 Mode :character   
## Mean :12.66   
## 3rd Qu.:19.00   
## Max. :27.00   
##

Como observamos tanto con la función “summary” como con la función “class” nos muestran que a cada variable se le ha asignado o bien la clase de entero o la de string. En concreto, muchas de las variables interpretadas como string se tratan de diccionarios y listas, factor que deberemos tener en cuenta a la hora de trabajar con estas variables.

# Tipo de dato asignado a cada variable  
  
res <- sapply(booking, function(x) class(x))  
kable(data.frame(var=names(res), clase=as.vector(res)))

| var | clase |
| --- | --- |
| name | character |
| city | character |
| check.in | character |
| adults | integer |
| children | integer |
| check.out | character |
| num\_rooms | integer |
| address | character |
| hotel\_coordinates | character |
| hotel\_score | numeric |
| hotel\_scores | character |
| hotel\_description | character |
| features | character |
| room\_data | character |
| page\_count | integer |
| current\_page | integer |
| in\_page\_count | integer |
| search\_date | character |

#### 2.1. Transformación de las variables en el formato adecuado

Como se ha comentado, existen algunas variables que presentan todavía una estructura no válida para tratarlos estadísticamente o introducirlos en un modelo de data mining o machine learning. En esta sección nos encargamos de tratar dichas columnas a fin de obtener nuevas variables con un formato adecuado, eliminando a su vez cualquier informaión que no nos sea de utilidad.

En primer lugar, cabe destacar que no todas las columnas que tenemos disponibles en el dataset nos ofrecen información relevante para nuestro problema. Por tanto, a fin de reducir la dimensionalidad de este, nos encargamos de eliminar las columnas con información redundante.

# Seleccionamos con la función "select" de dplyr todas las columnas excepto name y search\_date, que nos van a ser de poca utilidad.  
  
booking <- booking %>%  
 select(-c("name", "search\_date"))

La columna hotel\_coordinates incluye tanto los valores de latitud como los de longitud, separados por una coma. Dado que ambos dos valores nos indica información distinta, pues la latitud nos proporcionar información de la posición en dirección norte o sur del ecuador y la longitud información de la posición en dirección este u oeste, en este caso hemos optado por separarlos en dos columnas distintas.

# Separamos los valores de la variable hotel coordinates en dos nuevas columnas, latitud y longitud  
# La columna hotel\_coordinates se elimina en el proceso  
  
booking <- booking %>%  
 separate(hotel\_coordinates, c("latitude", "longitude"), sep = ",")

## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 16 rows [202,  
## 203, 223, 224, 225, 226, 227, 664, 665, 765, 1598, 2455, 2672, 2789, 2993,  
## 2999].

booking$latitude <- as.numeric(booking$latitude)  
booking$longitude <- as.numeric(booking$longitude)

Por su parte, “hotel\_scores” alberga diccionarios en cada uno de los registros. Cada diccionario consta del nombre de cada dimensión (caracterísitca a valorar) como clave y la valoración que recibe como valor. Dado que R lo ha interpretado como un string, nuestro enfoque en su limpieza se ha basado en separar la información con comas y, una vez obtenidas todas las columnas resultantes, extraer los datos numéricos de cada registro para quedarnos solo con los scores de cada dimensión.

# Definimos un vector con los nombres de las columnas a crear  
  
columns <- c("staff\_score", "facilities\_score", "cleanliness\_score", "comfort\_score", "value\_for\_money\_score", "location\_score", "free\_wifi\_score")  
  
# Separamos los strings (diccionarios) en función de las comas y creamos las columnas correspondientes. La primera columna creada es una NA porque todos los primeros registros en cada diccionario están vacíos (por un fallo en la extracción de datos). Estableciendo la columna como NA, indicamos a la función separate que ignore dichos datos y no cree columna alguna para ellos.  
  
booking <- booking %>%  
 separate(hotel\_scores, c(NA, "staff\_score", "facilities\_score", "cleanliness\_score", "comfort\_score", "value\_for\_money\_score", "location\_score", "free\_wifi\_score"), sep = ",")

## Warning: Expected 8 pieces. Missing pieces filled with `NA` in 423 rows [9, 80,  
## 105, 125, 141, 202, 203, 218, 223, 224, 225, 226, 227, 247, 303, 311, 322, 327,  
## 336, 370, ...].

# Extraemos los valores numéricos de cada columna con la función del paquete readr, "parse\_number"  
  
booking[columns] <- apply(booking[columns], 2, readr::parse\_number)

Dado que de las fechas tan sólo nos interesa el mes en el que se realiza la reserva (un alojamiento con unas características determinadas pdría estar mejor posicionado en un determinado mes que en otro) procedemos a extraer dicha información de las variables “check-in” y “check-out”. No obstante, por el momento no eliminamos dichas columnas y las transformamos a formato date por si a caso podemos hacer uso de ellas más adelante.

booking <- booking %>%  
 # Separamos el nombre del mes del resto de elementos del registro para check\_in y check\_out  
 separate(check.in, c(NA, "month\_in", NA), sep = "-", remove = FALSE) %>%  
 separate(check.out, c(NA, "month\_out", NA), sep = "-", remove = FALSE) %>%  
 # Modificamos la columna check\_in para que tenga el formato de fecha normal  
 separate(check.in, c("day", "month", "year"), sep = "-") %>%  
 mutate(month = ifelse(month == "March", "03", ifelse(month == "June", "06", "12"))) %>%  
 unite("check\_in", c(day, month, year), sep = "-") %>%  
 # Modificamos la columna check\_out para que tenga el formato de fecha normal  
 separate(check.out, c("day", "month", "year"), sep = "-") %>%  
 mutate(month = ifelse(month == "March", "03", ifelse(month == "June", "06", "12"))) %>%  
 unite("check\_out", c(day, month, year), sep = "-")  
  
# Transformamos las columnas check\_in y check\_out al format date  
booking["check\_in"] <- as.Date.character(booking$check\_in, "%d-%m-%Y")  
booking["check\_out"] <- as.Date(booking$check\_out, "%d-%m-%Y")

De la columna address tan sólo nos interesa el código postal (el barrio no se incluye en todos los registros a diferencia del CP, así que hemos optado por quedarnos con estos valores), por lo que extraemos dicha información para crear una nueva columna llamada “postal\_code”, eliminando la información redundante en el proceso.

# Hemos tenido que cambiar esta función para que el extract no diera problemas  
Sys.setlocale('LC\_ALL', 'C')

## [1] "C"

# Extraemos los valores de codigo postal de la columna addres utilizando expressiones regulares  
booking <- booking %>%  
 extract(address, c("postal\_code"), regex = "( [0-9]{5} )", remove = FALSE)

Los features, entendidos como servicios adicionales más allá de la propia habitación, fueron guardados como una lista. No obstante, de nuevo nos encontramos como R lo ha identificado como string. Es por ello por lo que realizamos un tratamiento similar al de la columna de scores. En este caso creamos tantas variables dummy (columnas dicotómicas con valores de 1, en caso de que el feature exista en dicho registro, o 0, en caso de que no existe) como servicios adicionales de hotel consideramos interesantes.

# Definimos las features que consideramos que son interesantes. Utilizamos este formato para filtrar por expresión regular.  
feature\_list <- c("('Free Wifi')", "('Air conditioning')", "('24-hour front desk')", "('Safe')", "('Heating')", "('Elevator')", "('Private Bathroom')", "('Non-smoking rooms')", "('Aparments')", "('City view')","('Kitchen')", "('Pet Friendly')", "('Swimming pool')", "('Balcony')")  
  
# Creamos un bucle en el que se recorre cada elemento de la lista de features  
for (feature in feature\_list){  
 # Se eliminan los parentésis y comillas de la variable local feature para crear el nombre de la columna.  
 col\_name <- str\_replace(str\_replace(str\_to\_lower(str\_extract(feature, "([A-Z][a-z]\*( |-)?[A-Z]?[a-z]\* ? ?[a-z]\*)")), " ", "\_"), "-", "\_")  
 booking <- booking %>%  
 # Extraemos el nombre del feature de cada uno de los registros (strings). En caso de que no encuentre ningún valor devuelve un NA.  
 extract(features, c(col\_name), regex = feature, remove = FALSE) %>%  
 # Transformamos la columna que acaba de ser creada para que indique con un 1 si el registro tenia dicho servicio y 0 si el valor era NA (no tenida dicho servicio)  
 mutate\_(.dots = setNames(list(paste0("as.integer(!is.na(",col\_name,"))")), col\_name))  
}

## Warning: `mutate\_()` was deprecated in dplyr 0.7.0.  
## i Please use `mutate()` instead.  
## i See vignette('programming') for more help  
## i The deprecated feature was likely used in the dplyr package.  
## Please report the issue at <]8;;https://github.com/tidyverse/dplyr/issueshttps://github.com/tidyverse/dplyr/issues]8;;>.

# Eliminamos la columna de features para finalizar  
booking <- booking %>%  
 select(-c("features"))

#### 2.2. Ceros y elementos vacíos

Dando paso a el tratamiento de ceros y elementos vacios, lo primero que debemos hacer es identificar aquellas variables en las que encontramos valores faltantes. Como observamos, “postal\_code” (19), “longitude” (16), “hotel\_score” (8) y el resto de variables relacionadas con los scores (60) presentan valores faltantes.

sapply(booking, function(y) sum(length(which(is.na(y)))))

## city check\_in month\_in   
## 0 0 0   
## adults children check\_out   
## 0 0 0   
## month\_out num\_rooms address   
## 0 0 0   
## postal\_code latitude longitude   
## 19 16 16   
## hotel\_score staff\_score facilities\_score   
## 8 60 60   
## cleanliness\_score comfort\_score value\_for\_money\_score   
## 60 60 60   
## location\_score free\_wifi\_score hotel\_description   
## 60 423 0   
## balcony swimming\_pool pet\_friendly   
## 0 0 0   
## kitchen city\_view aparments   
## 0 0 0   
## non\_smoking\_rooms private\_bathroom elevator   
## 0 0 0   
## heating safe air\_conditioning   
## 0 0 0   
## free\_wifi room\_data page\_count   
## 0 0 0   
## current\_page in\_page\_count as.integer(!is.na(NA))   
## 0 0 0

# c(1:4) es por el output de knitr, más adelante lo cambiamos.  
head(booking[is.na(booking$postal\_code),c(1:4)], 5)

## city check\_in month\_in adults  
## 202 Barcelona 2022-12-12 December 4  
## 203 Barcelona 2022-12-12 December 4  
## 223 Barcelona 2022-12-12 December 4  
## 224 Barcelona 2022-12-12 December 4  
## 225 Barcelona 2022-12-12 December 4

booking <- booking[!is.na(booking$postal\_code),]

sapply(booking, function(y) sum(is.na(y)))

## city check\_in month\_in   
## 0 0 0   
## adults children check\_out   
## 0 0 0   
## month\_out num\_rooms address   
## 0 0 0   
## postal\_code latitude longitude   
## 0 0 0   
## hotel\_score staff\_score facilities\_score   
## 5 44 44   
## cleanliness\_score comfort\_score value\_for\_money\_score   
## 44 44 44   
## location\_score free\_wifi\_score hotel\_description   
## 44 407 0   
## balcony swimming\_pool pet\_friendly   
## 0 0 0   
## kitchen city\_view aparments   
## 0 0 0   
## non\_smoking\_rooms private\_bathroom elevator   
## 0 0 0   
## heating safe air\_conditioning   
## 0 0 0   
## free\_wifi room\_data page\_count   
## 0 0 0   
## current\_page in\_page\_count as.integer(!is.na(NA))   
## 0 0 0

head(booking[is.na(booking$hotel\_score),c(1:4)], 5)

## city check\_in month\_in adults  
## 310 Barcelona 2022-12-12 December 5  
## 844 Valencia 2022-12-12 December 4  
## 958 Valencia 2022-12-12 December 5  
## 1210 Barcelona 2023-03-12 March 2  
## 2708 Barcelona 2023-06-12 June 2

tail(booking[booking$hotel\_score == "-1",c(1:4)], 5)

## city check\_in month\_in adults  
## 3690 Madrid 2023-06-12 June 2  
## 3736 Madrid 2023-03-12 March 2  
## 3855 Madrid 2023-03-12 March 5  
## 3877 Madrid 2023-03-12 March 2  
## 3909 Madrid 2023-03-12 March 4

booking$hotel\_score <- ifelse(booking$hotel\_score == "-1", NA, booking$hotel\_score)

booking$hotel\_score <- kNN(booking)$hotel\_score  
booking$staff\_score <- kNN(booking)$staff\_score   
booking$facilities\_score <- kNN(booking)$facilities\_score  
booking$cleanliness\_score <- kNN(booking)$cleanliness\_score  
booking$comfort\_score <- kNN(booking)$comfort\_score  
booking$value\_for\_money\_score <- kNN(booking)$value\_for\_money\_score  
booking$location\_score <- kNN(booking)$location\_score  
# Do not use the same treatement  
booking$free\_wifi\_score <- ifelse(is.na(booking$free\_wifi\_score), 0, booking$free\_wifi\_score)

sapply(booking, function(y) sum(is.na(y)))

## city check\_in month\_in   
## 0 0 0   
## adults children check\_out   
## 0 0 0   
## month\_out num\_rooms address   
## 0 0 0   
## postal\_code latitude longitude   
## 0 0 0   
## hotel\_score staff\_score facilities\_score   
## 0 0 0   
## cleanliness\_score comfort\_score value\_for\_money\_score   
## 0 0 0   
## location\_score free\_wifi\_score hotel\_description   
## 0 0 0   
## balcony swimming\_pool pet\_friendly   
## 0 0 0   
## kitchen city\_view aparments   
## 0 0 0   
## non\_smoking\_rooms private\_bathroom elevator   
## 0 0 0   
## heating safe air\_conditioning   
## 0 0 0   
## free\_wifi room\_data page\_count   
## 0 0 0   
## current\_page in\_page\_count as.integer(!is.na(NA))   
## 0 0 0

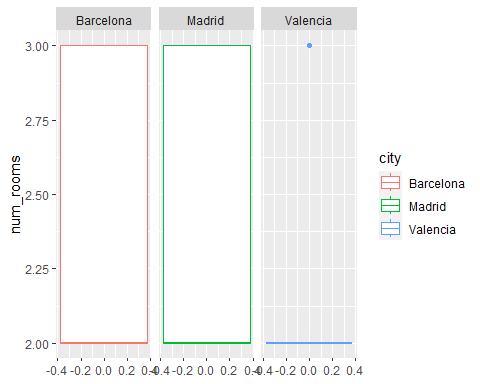
sapply(booking, function(y) sum(y == ""))

## city check\_in month\_in   
## 0 NA 0   
## adults children check\_out   
## 0 0 NA   
## month\_out num\_rooms address   
## 0 0 0   
## postal\_code latitude longitude   
## 0 0 0   
## hotel\_score staff\_score facilities\_score   
## 0 0 0   
## cleanliness\_score comfort\_score value\_for\_money\_score   
## 0 0 0   
## location\_score free\_wifi\_score hotel\_description   
## 0 0 0   
## balcony swimming\_pool pet\_friendly   
## 0 0 0   
## kitchen city\_view aparments   
## 0 0 0   
## non\_smoking\_rooms private\_bathroom elevator   
## 0 0 0   
## heating safe air\_conditioning   
## 0 0 0   
## free\_wifi room\_data page\_count   
## 0 0 0   
## current\_page in\_page\_count as.integer(!is.na(NA))   
## 0 0 0

#### 2.3. Valores extremos

Los outliers son aquellos datos extremos que dada su distancia con respecto al grueso de la distribución a priori resultan no ser congruentes si los comparamos con la población/muestra analizada. Existen varias vías para detectarlos, no obstante el método más rápdio y común se basa en el uso de boxplots, donde se detectan aquellos valores que se sitúan 1.5 veces más allá del rango intercuartílico con respecto a la mediana. Asimismo, en este trabajo también haremos uso de la función “boxplots.stats()” a fin de detectar, con un output más allá del visual, los outliers que encontramos usando boxplots. Dado que muchas de las variables que hemos generado en la fase de limpieza son dicotómicas, tan solo nos encargaremos de comprobar si existen valores extremos en las columnas de **num\_rooms**, **latitude**, **longitude**, **hotel\_score** y el resto de **scores del hotel**.

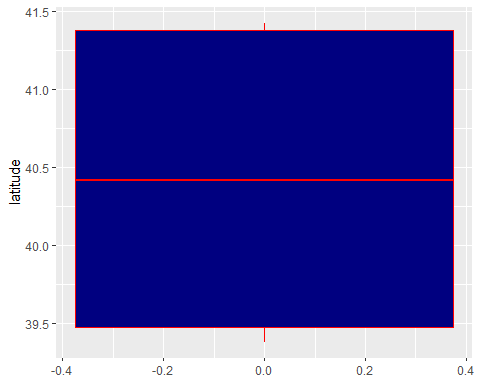
ggplot(booking, aes(y = num\_rooms, color = city))+  
 geom\_boxplot() +  
 facet\_grid(~city)



boxplot.stats(booking$num\_rooms)$out

## integer(0)

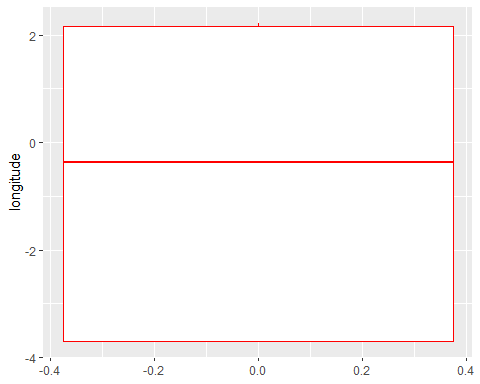
ggplot(booking, aes(y = latitude))+  
 geom\_boxplot(color = "red", fill = "navy")



boxplot.stats(booking$latitude)$out

## numeric(0)

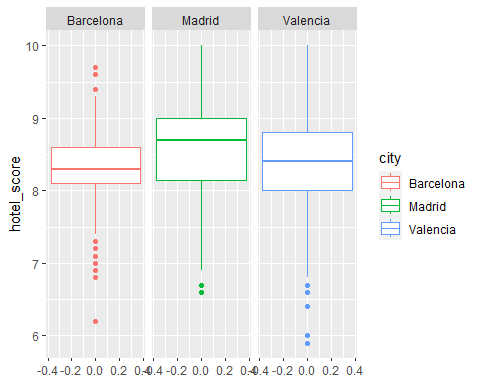
ggplot(booking, aes(y = longitude))+  
 geom\_boxplot(color = "red", fill = "white")



boxplot.stats(booking$longitude)$out

## numeric(0)

ggplot(booking, aes(y = hotel\_score, color = city))+  
 geom\_boxplot() +  
 facet\_grid(~city)



boxplot.stats(booking$hotel\_score)$out

## [1] 6.9 6.9 6.9 6.9 6.9 10.0 6.6 6.6 5.9 7.0 6.6 6.0 6.6 5.9 6.7  
## [16] 10.0 5.9 7.0 6.8 6.6 6.7 6.9 6.0 6.9 6.4 10.0 6.6 6.6 10.0 7.0  
## [31] 6.9 6.9 6.9 10.0 7.0 6.6 7.0 6.6 6.7 7.0 6.4 7.0 6.9 6.8 10.0  
## [46] 6.9 6.9 7.0 6.2 6.8 7.0 6.6 6.7 7.0 6.9 6.9 6.9 6.9 7.0 7.0  
## [61] 6.9 6.9 7.0 5.9 6.0 7.0 6.9 7.0 6.8 6.8 5.9 7.0 10.0 6.6 6.6  
## [76] 6.6 7.0 6.9 10.0 6.6 10.0 10.0 6.6

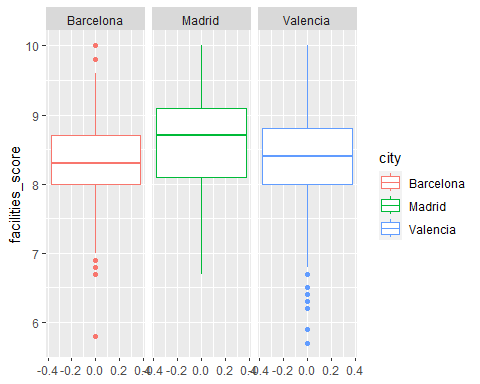
booking %>%  
 select(-c("hotel\_description", "room\_data")) %>%  
 filter(hotel\_score <= 7.0) %>%  
 sample\_n(3)

## city check\_in month\_in adults children check\_out month\_out num\_rooms  
## 1 Barcelona 2023-03-12 March 2 3 2023-03-16 March 2  
## 2 Madrid 2022-12-12 December 5 3 2022-12-16 December 3  
## 3 Valencia 2022-12-12 December 2 3 2022-12-16 December 2  
## address  
## 1 Paris, 101, Eixample, 08029 Barcelona, Spain  
## 2 11 Calle de la Luna, Madrid City Center, 28004 Madrid, Spain  
## 3 Av. del Bar\357\277\275 de C\357\277\275rcer, 37., Ciutat Vella, 46001 Valencia, Spain  
## postal\_code latitude longitude hotel\_score staff\_score facilities\_score  
## 1 08029 41.38942 2.1475435 6.9 8.2 6.8  
## 2 28004 40.42239 -3.7062110 6.6 7.9 6.7  
## 3 46001 39.47047 -0.3795761 6.0 6.4 5.7  
## cleanliness\_score comfort\_score value\_for\_money\_score location\_score  
## 1 7.5 7.3 7.0 8.2  
## 2 7.1 6.7 5.8 8.8  
## 3 6.4 6.1 6.7 8.9  
## free\_wifi\_score balcony swimming\_pool pet\_friendly kitchen city\_view  
## 1 7.2 0 0 0 0 0  
## 2 0.0 1 0 0 1 1  
## 3 10.0 0 0 0 1 1  
## aparments non\_smoking\_rooms private\_bathroom elevator heating safe  
## 1 0 0 0 0 0 0  
## 2 0 1 1 0 0 1  
## 3 0 1 1 0 1 0  
## air\_conditioning free\_wifi page\_count current\_page in\_page\_count  
## 1 0 0 48 2 23  
## 2 1 0 76 3 25  
## 3 1 0 99 4 22  
## as.integer(!is.na(NA))  
## 1 0  
## 2 0  
## 3 0

booking %>%  
 select(-c("hotel\_description", "room\_data")) %>%  
 filter(hotel\_score == 10.0) %>%  
 sample\_n(3)

## city check\_in month\_in adults children check\_out month\_out num\_rooms  
## 1 Valencia 2023-06-12 June 4 1 2023-06-16 June 2  
## 2 Madrid 2023-03-12 March 2 3 2023-03-16 March 2  
## 3 Madrid 2022-12-12 December 5 3 2022-12-16 December 3  
## address  
## 1 38 Passeig de l'Albereda, Camins al Grau, 46023 Valencia, Spain  
## 2 Calle de los Reyes, Madrid City Center, 28015 Madrid, Spain  
## 3 44 Carrera de San Jer\357\277\275nimo, Madrid City Center, 28014 Madrid, Spain  
## postal\_code latitude longitude hotel\_score staff\_score facilities\_score  
## 1 46023 39.46342 -0.3575133 10 10 10  
## 2 28015 40.42396 -3.7098710 10 10 10  
## 3 28014 40.41612 -3.6975489 10 10 10  
## cleanliness\_score comfort\_score value\_for\_money\_score location\_score  
## 1 10 10 10 10  
## 2 10 10 10 10  
## 3 10 10 10 10  
## free\_wifi\_score balcony swimming\_pool pet\_friendly kitchen city\_view  
## 1 0 0 0 0 1 0  
## 2 0 1 0 0 1 0  
## 3 0 0 0 0 1 0  
## aparments non\_smoking\_rooms private\_bathroom elevator heating safe  
## 1 0 0 0 0 0 0  
## 2 0 0 1 0 1 0  
## 3 0 0 1 1 0 0  
## air\_conditioning free\_wifi page\_count current\_page in\_page\_count  
## 1 1 0 94 4 19  
## 2 1 0 55 3 4  
## 3 1 0 36 2 11  
## as.integer(!is.na(NA))  
## 1 0  
## 2 0  
## 3 0

ggplot(booking, aes(y = facilities\_score, color = city))+  
 geom\_boxplot() +  
 facet\_grid(~city)



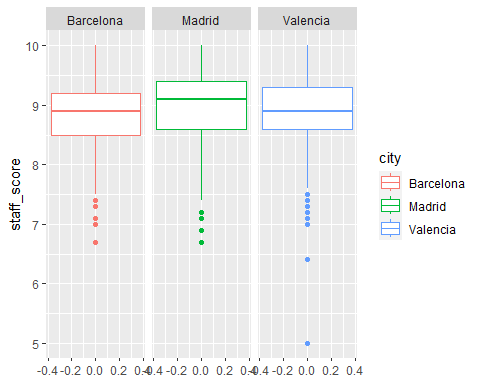
boxplot.stats(booking$facilities\_score)$out

## [1] 5.9 6.5 5.7 6.4 5.9 5.9 6.5 6.4 6.3 5.9 6.2 6.5 6.5 5.8 6.5 5.9 5.7 6.5 6.5  
## [20] 5.9

booking %>%  
 select(-c("hotel\_description", "room\_data")) %>%  
 filter(facilities\_score <= 6.5) %>%  
 sample\_n(3)

## city check\_in month\_in adults children check\_out month\_out num\_rooms  
## 1 Valencia 2022-12-12 December 5 3 2022-12-16 December 3  
## 2 Valencia 2023-06-12 June 2 3 2023-06-16 June 2  
## 3 Barcelona 2022-12-12 December 5 3 2022-12-16 December 3  
## address postal\_code  
## 1 Carrer del Peu de la Creu 5, Ciutat Vella, 46001 Valencia, Spain 46001   
## 2 Calle Navardera, 2 , Poblats Maritims, 46024 Valencia, Spain 46024   
## 3 264 Carrer de Sic\357\277\275lia, Eixample, 08013 Barcelona, Spain 08013   
## latitude longitude hotel\_score staff\_score facilities\_score  
## 1 39.47310 -0.3803926 6.4 7.3 6.2  
## 2 39.45995 -0.3338888 5.9 7.1 5.9  
## 3 41.40005 2.1758251 6.2 7.1 5.8  
## cleanliness\_score comfort\_score value\_for\_money\_score location\_score  
## 1 6.5 6.6 6.9 8.7  
## 2 5.9 6.1 6.7 7.6  
## 3 5.4 7.1 5.4 7.1  
## free\_wifi\_score balcony swimming\_pool pet\_friendly kitchen city\_view  
## 1 8.0 0 0 0 0 0  
## 2 7.7 0 0 0 1 0  
## 3 0.0 1 0 0 1 0  
## aparments non\_smoking\_rooms private\_bathroom elevator heating safe  
## 1 0 0 0 0 0 0  
## 2 0 1 1 1 1 0  
## 3 0 1 0 1 1 0  
## air\_conditioning free\_wifi page\_count current\_page in\_page\_count  
## 1 0 0 118 5 13  
## 2 1 0 11 1 11  
## 3 0 0 69 3 18  
## as.integer(!is.na(NA))  
## 1 0  
## 2 0  
## 3 0

ggplot(booking, aes(y = staff\_score, color = city))+  
 geom\_boxplot() +  
 facet\_grid(~city)



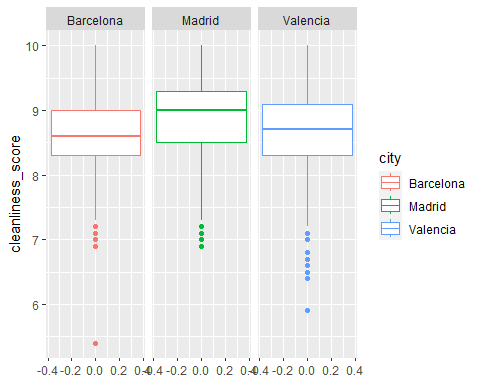
boxplot.stats(booking$staff\_score)$out

## [1] 7.0 7.5 7.4 7.4 7.5 7.4 7.5 7.5 7.5 7.5 6.7 7.5 7.1 7.3 7.3 6.4 7.1 7.2  
## [19] 7.1 7.4 7.4 7.5 7.0 7.4 7.1 7.3 7.2 7.4 7.5 7.5 7.5 6.7 7.4 7.5 7.2 7.1  
## [37] 6.9 7.5 7.4 7.5 7.5 7.5 7.4 7.4 5.0 5.0 7.3 7.3 7.2 7.5 7.1 7.4 7.2 7.3  
## [55] 7.3 7.4 7.3 7.1 5.0 7.3 7.4 7.5 7.0 7.5 7.3 7.4 7.4 7.1 6.7 7.5 7.4 7.5  
## [73] 7.4 7.5 6.7 7.5 7.5 7.4 7.4 7.1 7.5 7.5 7.4 7.5 7.5 7.5 7.4 7.2 7.3 7.5  
## [91] 7.5 7.4 6.7 7.5 7.4 7.4 7.5 7.3 7.3 7.1 6.4 7.3 7.3 7.3 7.3 7.1 7.3 7.5  
## [109] 7.5 6.7 7.5 7.5 7.4 7.5 7.5 7.1 7.4 6.9 7.4 7.5 7.4 6.7 7.4 7.5 6.7 7.4  
## [127] 7.1 7.5 7.5 7.5 7.2

booking %>%  
 select(-c("hotel\_description", "room\_data")) %>%  
 filter(staff\_score <= 6.7) %>%  
 sample\_n(3)

## city check\_in month\_in adults children check\_out month\_out num\_rooms  
## 1 Madrid 2023-03-12 March 5 3 2023-03-16 March 3  
## 2 Valencia 2023-03-12 March 4 1 2023-03-16 March 2  
## 3 Madrid 2023-06-12 June 2 3 2023-06-16 June 2  
## address  
## 1 6 Calle de Hern\357\277\275n Cort\357\277\275s, Madrid City Center, 28004 Madrid, Spain  
## 2 c/Bonrempos i Mirambell n\357\277\27517, Bloque2 pta9, 46200 Valencia, Spain  
## 3 6 Calle de Hern\357\277\275n Cort\357\277\275s, Madrid City Center, 28004 Madrid, Spain  
## postal\_code latitude longitude hotel\_score staff\_score facilities\_score  
## 1 28004 40.42347 -3.700148 7.3 6.7 7  
## 2 46200 39.43310 -0.408367 9.0 5.0 10  
## 3 28004 40.42347 -3.700148 7.3 6.7 7  
## cleanliness\_score comfort\_score value\_for\_money\_score location\_score  
## 1 7.5 7.4 7.7 9.4  
## 2 10.0 10.0 10.0 7.5  
## 3 7.5 7.4 7.7 9.4  
## free\_wifi\_score balcony swimming\_pool pet\_friendly kitchen city\_view  
## 1 0 0 0 0 1 0  
## 2 0 0 0 0 1 0  
## 3 0 0 0 0 1 0  
## aparments non\_smoking\_rooms private\_bathroom elevator heating safe  
## 1 0 0 0 0 0 0  
## 2 0 0 0 0 0 0  
## 3 0 0 0 0 0 0  
## air\_conditioning free\_wifi page\_count current\_page in\_page\_count  
## 1 0 0 46 2 21  
## 2 1 0 18 1 18  
## 3 0 0 68 3 17  
## as.integer(!is.na(NA))  
## 1 0  
## 2 0  
## 3 0

ggplot(booking, aes(y = cleanliness\_score, color = city))+  
 geom\_boxplot() +  
 facet\_grid(~city)



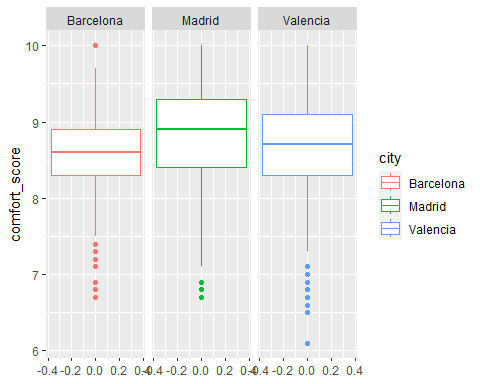
boxplot.stats(booking$cleanliness\_score)$out

## [1] 7.2 6.9 7.2 7.0 6.9 7.1 5.9 6.8 7.0 7.1 6.4 7.1 5.9 7.2 6.7 7.1 7.1 7.2 5.9  
## [20] 6.4 6.7 7.1 6.8 6.6 6.5 7.0 6.9 7.1 7.0 7.1 7.1 7.0 6.9 6.8 7.0 6.7 7.1 7.0  
## [39] 7.0 6.8 7.0 6.8 7.1 7.1 7.2 7.2 6.9 6.9 5.4 7.1 7.2 7.0 6.9 6.8 7.2 6.5 7.2  
## [58] 6.9 6.8 5.9 6.4 6.8 6.5 7.1 6.8 6.7 7.1 6.4 5.9 6.8 7.2 6.9 6.9 7.0 7.1 7.1  
## [77] 7.0 7.0 7.2 7.1 6.9 6.9 7.0

booking %>%  
 select(-c("hotel\_description", "room\_data")) %>%  
 filter(cleanliness\_score < 7.0) %>%  
 sample\_n(3)

## city check\_in month\_in adults children check\_out month\_out num\_rooms  
## 1 Madrid 2022-12-12 December 2 3 2022-12-16 December 2  
## 2 Valencia 2023-06-12 June 2 3 2023-06-16 June 2  
## 3 Valencia 2022-12-12 December 5 3 2022-12-16 December 3  
## address  
## 1 Madrid City Center, 28005 Madrid, Spain  
## 2 78 Avinguda del Cardenal Benlloch, Algiros, 46021 Valencia, Spain  
## 3 Carrer d'Empar Ballester 25, Poblats Maritims, 46022 Valencia, Spain  
## postal\_code latitude longitude hotel\_score staff\_score facilities\_score  
## 1 28005 40.40919 -3.7113892 6.6 7.5 7.2  
## 2 46021 39.47228 -0.3536916 7.0 7.3 7.4  
## 3 46022 39.47170 -0.3353337 7.5 8.6 7.4  
## cleanliness\_score comfort\_score value\_for\_money\_score location\_score  
## 1 6.9 6.9 7.5 9.1  
## 2 6.8 7.1 7.5 6.8  
## 3 6.4 7.6 8.7 7.9  
## free\_wifi\_score balcony swimming\_pool pet\_friendly kitchen city\_view  
## 1 0.0 1 0 0 1 0  
## 2 5.0 0 0 0 1 0  
## 3 7.5 0 0 0 1 0  
## aparments non\_smoking\_rooms private\_bathroom elevator heating safe  
## 1 0 1 1 0 0 0  
## 2 0 1 1 0 0 0  
## 3 0 1 1 0 1 0  
## air\_conditioning free\_wifi page\_count current\_page in\_page\_count  
## 1 1 0 78 4 1  
## 2 1 0 13 1 13  
## 3 1 0 61 3 10  
## as.integer(!is.na(NA))  
## 1 0  
## 2 0  
## 3 0

ggplot(booking, aes(y = comfort\_score, color = city))+  
 geom\_boxplot() +  
 facet\_grid(~city)



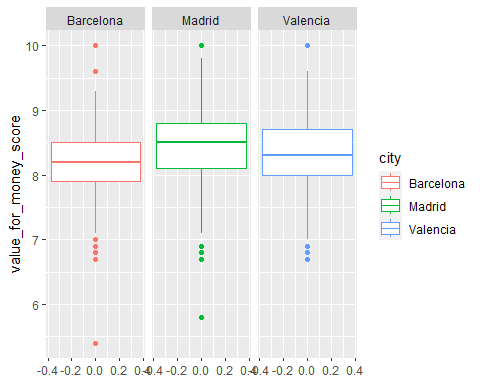
boxplot.stats(booking$comfort\_score)$out

## [1] 7.1 7.1 7.1 6.8 6.9 7.1 6.7 6.1 7.1 6.8 6.1 6.7 6.1 7.0 7.1 7.1 6.1 7.0 6.7  
## [20] 6.7 6.6 6.9 6.5 6.6 6.9 7.1 6.7 7.1 7.1 6.9 6.8 7.1 6.9 6.8 7.0 7.1 7.1 6.8  
## [39] 6.7 7.1 6.7 7.1 7.1 6.9 7.1 7.1 7.0 7.1 7.1 7.1 7.1 6.7 7.1 6.1 6.1 7.1 7.0  
## [58] 7.1 7.1 6.7 6.7 6.1 6.9 7.1 6.9 6.7 6.8 7.1 6.9 6.9

booking %>%  
 select(-c("hotel\_description", "room\_data")) %>%  
 filter(comfort\_score < 7.0) %>%  
 sample\_n(3)

## city check\_in month\_in adults children check\_out month\_out num\_rooms  
## 1 Madrid 2023-03-12 March 2 3 2023-03-16 March 2  
## 2 Valencia 2022-12-12 December 2 3 2022-12-16 December 2  
## 3 Madrid 2022-12-12 December 5 3 2022-12-16 December 3  
## address postal\_code  
## 1 Madrid City Center, 28005 Madrid, Spain 28005   
## 2 Calle Navardera, 2 , Poblats Maritims, 46024 Valencia, Spain 46024   
## 3 Madrid City Center, 28005 Madrid, Spain 28005   
## latitude longitude hotel\_score staff\_score facilities\_score  
## 1 40.40919 -3.7113892 6.6 7.5 7.2  
## 2 39.45995 -0.3338888 5.9 7.1 5.9  
## 3 40.40919 -3.7113892 6.6 7.5 7.2  
## cleanliness\_score comfort\_score value\_for\_money\_score location\_score  
## 1 6.9 6.9 7.5 9.1  
## 2 5.9 6.1 6.7 7.6  
## 3 6.9 6.9 7.5 9.1  
## free\_wifi\_score balcony swimming\_pool pet\_friendly kitchen city\_view  
## 1 0.0 1 0 0 1 0  
## 2 7.7 0 0 0 1 0  
## 3 0.0 1 0 0 1 0  
## aparments non\_smoking\_rooms private\_bathroom elevator heating safe  
## 1 0 1 1 0 0 0  
## 2 0 1 1 1 1 0  
## 3 0 1 1 0 0 0  
## air\_conditioning free\_wifi page\_count current\_page in\_page\_count  
## 1 1 0 78 4 1  
## 2 1 0 11 1 11  
## 3 1 0 53 3 2  
## as.integer(!is.na(NA))  
## 1 0  
## 2 0  
## 3 0

ggplot(booking, aes(y = value\_for\_money\_score, color = city))+  
 geom\_boxplot() +  
 facet\_grid(~city)



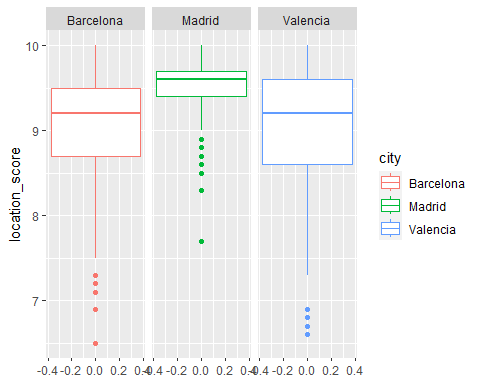
boxplot.stats(booking$value\_for\_money\_score)$out

## [1] 6.9 6.9 10.0 10.0 10.0 5.8 6.7 10.0 6.7 10.0 10.0 6.7 6.9 10.0 6.7  
## [16] 6.8 6.9 6.7 6.7 6.9 10.0 10.0 10.0 5.8 10.0 9.8 10.0 10.0 10.0 6.8  
## [31] 9.8 10.0 10.0 10.0 10.0 10.0 6.9 10.0 9.8 10.0 10.0 6.7 6.9 10.0 10.0  
## [46] 6.9 6.9 10.0 6.7 6.7 5.4 6.9 10.0 10.0 6.8 10.0 10.0 10.0 6.9 6.8  
## [61] 10.0 10.0 6.7 6.7 6.7 10.0 6.9 6.7 10.0 10.0 10.0 5.8 6.9 10.0 10.0  
## [76] 9.8 10.0 10.0 10.0

booking %>%  
 select(-c("hotel\_description", "room\_data")) %>%  
 filter(value\_for\_money\_score <= 7.0) %>%  
 sample\_n(3)

## city check\_in month\_in adults children check\_out month\_out num\_rooms  
## 1 Madrid 2022-12-12 December 4 1 2022-12-16 December 2  
## 2 Barcelona 2022-12-12 December 5 3 2022-12-16 December 3  
## 3 Barcelona 2023-03-12 March 2 3 2023-03-16 March 2  
## address  
## 1 Calle Jes\357\277\275s y Mar\357\277\275a, 17, Madrid City Center, 28012 Madrid, Spain  
## 2 Carrer Comandant Benitez 20 4\357\277\2751\357\277\275 BCN, Les Corts, 08028 Barcelona, Spain  
## 3 Carrer de Bail\357\277\275n, 22, Eixample, 08010 Barcelona, Spain  
## postal\_code latitude longitude hotel\_score staff\_score facilities\_score  
## 1 28012 40.41117 -3.703568 6.9 9.3 7.1  
## 2 08028 41.38019 2.126472 7.0 7.6 6.8  
## 3 08010 41.39279 2.176247 7.7 7.7 7.5  
## cleanliness\_score comfort\_score value\_for\_money\_score location\_score  
## 1 8.3 6.8 6.9 9.4  
## 2 7.2 7.1 6.8 7.6  
## 3 7.6 7.5 7.0 8.8  
## free\_wifi\_score balcony swimming\_pool pet\_friendly kitchen city\_view  
## 1 7.5 0 0 0 1 0  
## 2 8.8 0 0 0 1 0  
## 3 0.0 0 0 0 1 0  
## aparments non\_smoking\_rooms private\_bathroom elevator heating safe  
## 1 0 1 1 0 1 0  
## 2 0 0 1 1 0 0  
## 3 0 0 1 1 1 0  
## air\_conditioning free\_wifi page\_count current\_page in\_page\_count  
## 1 1 0 100 4 23  
## 2 1 0 132 5 27  
## 3 0 0 63 3 13  
## as.integer(!is.na(NA))  
## 1 0  
## 2 0  
## 3 0

ggplot(booking, aes(y = location\_score, color = city))+  
 geom\_boxplot() +  
 facet\_grid(~city)



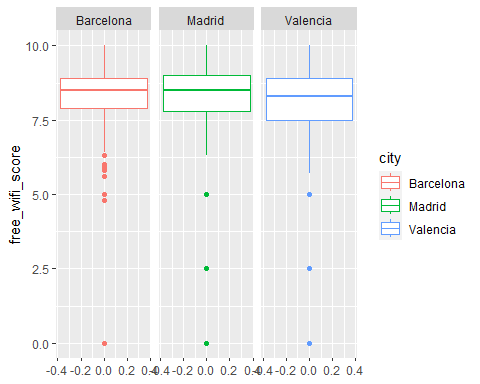
boxplot.stats(booking$location\_score)$out

## [1] 7.6 7.3 7.6 7.6 7.2 7.6 6.8 6.7 7.6 7.5 7.7 7.3 7.8 7.6 7.8 7.5 7.8 7.8  
## [19] 7.6 6.8 7.8 6.8 7.8 7.5 7.7 6.6 7.3 7.8 7.5 7.8 7.5 7.8 7.5 7.3 7.6 7.6  
## [37] 7.6 7.7 6.8 7.5 7.5 6.7 7.6 7.6 7.3 7.3 7.5 6.8 6.8 6.8 7.5 7.7 7.7 6.9  
## [55] 7.7 7.8 6.8 7.5 7.8 7.3 7.6 7.5 7.2 7.3 7.8 7.1 7.8 7.2 6.5 6.9 7.6 6.8  
## [73] 7.7 6.8 6.7 7.6 7.6 7.3 7.6 7.6 7.6 7.8 7.5 6.9 7.2 7.2 6.8 7.6 7.8 7.8  
## [91] 6.8 7.7 7.5 7.5 7.8 7.7 7.5 6.8 6.8 6.8 7.8 7.8 7.8 7.7 7.6 7.8 7.7

booking %>%  
 select(-c("hotel\_description", "room\_data")) %>%  
 filter(location\_score < 7.0) %>%  
 sample\_n(3)

## city check\_in month\_in adults children check\_out month\_out num\_rooms  
## 1 Valencia 2022-12-12 December 2 3 2022-12-16 December 2  
## 2 Valencia 2023-06-12 June 2 3 2023-06-16 June 2  
## 3 Valencia 2023-06-12 June 2 3 2023-06-16 June 2  
## address postal\_code  
## 1 78 Avinguda del Cardenal Benlloch, Algiros, 46021 Valencia, Spain 46021   
## 2 78 Avinguda del Cardenal Benlloch, Algiros, 46021 Valencia, Spain 46021   
## 3 78 Avinguda del Cardenal Benlloch, Algiros, 46021 Valencia, Spain 46021   
## latitude longitude hotel\_score staff\_score facilities\_score  
## 1 39.47228 -0.3536916 7 7.3 7.4  
## 2 39.47228 -0.3536916 7 7.3 7.4  
## 3 39.47228 -0.3536916 7 7.3 7.4  
## cleanliness\_score comfort\_score value\_for\_money\_score location\_score  
## 1 6.8 7.1 7.5 6.8  
## 2 6.8 7.1 7.5 6.8  
## 3 6.8 7.1 7.5 6.8  
## free\_wifi\_score balcony swimming\_pool pet\_friendly kitchen city\_view  
## 1 5 0 0 0 1 0  
## 2 5 0 0 0 1 0  
## 3 5 0 0 0 1 0  
## aparments non\_smoking\_rooms private\_bathroom elevator heating safe  
## 1 0 1 1 0 0 0  
## 2 0 1 1 0 0 0  
## 3 0 1 1 0 0 0  
## air\_conditioning free\_wifi page\_count current\_page in\_page\_count  
## 1 1 0 13 1 13  
## 2 1 0 55 3 4  
## 3 1 0 13 1 13  
## as.integer(!is.na(NA))  
## 1 0  
## 2 0  
## 3 0

ggplot(booking, aes(y = free\_wifi\_score, color = city))+  
 geom\_boxplot() +  
 facet\_grid(~city)



boxplot.stats(booking$free\_wifi\_score)$out

## [1] 0.0 5.6 0.0 0.0 0.0 0.0 5.6 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  
## [19] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  
## [37] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 5.0 0.0 0.0 0.0 5.0 0.0 0.0 0.0  
## [55] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 5.0 0.0 0.0 0.0  
## [73] 0.0 0.0 0.0 0.0 0.0 5.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  
## [91] 2.5 0.0 0.0 0.0 0.0 5.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  
## [109] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 4.8 5.6 0.0 0.0  
## [127] 5.6 0.0 4.8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 5.0 0.0 0.0 0.0 0.0 0.0  
## [145] 2.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  
## [163] 0.0 0.0 0.0 5.6 4.8 5.6 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  
## [181] 0.0 0.0 0.0 0.0 0.0 0.0 5.0 0.0 0.0 5.0 0.0 0.0 0.0 0.0 0.0 5.0 5.7 0.0  
## [199] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 5.0 5.0 0.0 0.0 0.0 0.0 0.0 0.0 5.0  
## [217] 0.0 0.0 0.0 0.0 5.6 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  
## [235] 5.0 0.0 0.0 5.6 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 5.8  
## [253] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 5.0 0.0 0.0 0.0  
## [271] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  
## [289] 0.0 0.0 0.0 5.0 0.0 0.0 0.0 5.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 5.6 0.0  
## [307] 4.8 5.6 0.0 5.6 5.6 4.8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 5.6 5.6  
## [325] 5.8 0.0 0.0 4.8 0.0 0.0 0.0 5.0 5.6 0.0 5.0 0.0 0.0 0.0 0.0 0.0 5.0 0.0  
## [343] 5.0 0.0 0.0 0.0 0.0 5.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  
## [361] 5.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 5.7 0.0 0.0 0.0 0.0 0.0  
## [379] 0.0 5.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  
## [397] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  
## [415] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 5.0 0.0 0.0 0.0  
## [433] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  
## [451] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

booking %>%  
 select(-c("hotel\_description", "room\_data")) %>%  
 filter(free\_wifi\_score < 5.0 & free\_wifi\_score > 0) %>%  
 sample\_n(3)

## city check\_in month\_in adults children check\_out month\_out num\_rooms  
## 1 Barcelona 2023-06-12 June 4 1 2023-06-16 June 2  
## 2 Barcelona 2023-06-12 June 5 3 2023-06-16 June 3  
## 3 Valencia 2022-12-12 December 5 3 2022-12-16 December 3  
## address postal\_code  
## 1 Carrer de Sant Pau, 34, Ciutat Vella, 08001 Barcelona, Spain 08001   
## 2 Carrer de Sant Pau, 34, Ciutat Vella, 08001 Barcelona, Spain 08001   
## 3 9 Calle Emilia Pardo Baz\357\277\275n, Olivereta, 46018 Valencia, Spain 46018   
## latitude longitude hotel\_score staff\_score facilities\_score  
## 1 41.37962 2.1724995 7.1 8.6 6.7  
## 2 41.37962 2.1724995 7.1 8.6 6.7  
## 3 39.46423 -0.3984764 8.3 9.0 8.7  
## cleanliness\_score comfort\_score value\_for\_money\_score location\_score  
## 1 7.5 7.1 7.7 9.1  
## 2 7.5 7.1 7.7 9.1  
## 3 9.0 8.8 8.6 8.0  
## free\_wifi\_score balcony swimming\_pool pet\_friendly kitchen city\_view  
## 1 4.8 0 0 0 0 0  
## 2 4.8 0 0 0 0 0  
## 3 2.5 0 0 0 1 0  
## aparments non\_smoking\_rooms private\_bathroom elevator heating safe  
## 1 0 0 0 0 0 0  
## 2 0 0 0 0 0 0  
## 3 0 1 1 0 1 0  
## air\_conditioning free\_wifi page\_count current\_page in\_page\_count  
## 1 0 0 42 2 17  
## 2 0 0 16 1 16  
## 3 1 0 75 3 24  
## as.integer(!is.na(NA))  
## 1 0  
## 2 0  
## 3 0

#### 2.4. Exportación de los datos postprocesados

#write.csv(booking, "hotel\_data\_processed.csv")

## 3. INTEGRACIÓN Y SELECCIÓN

En primer lugar, hemos obtado por añadir datos de vuelos por varias razones:

* Un mayor número de vuelos en un mes podría aumentar las búsquedas de hoteles cerca del aeropuerto.
* Un mayor número de vuelos en un mes indica más turismo internacional (en lugar de sólo turismo nacional). Un tipo de turismo que podría estar buscando hoteles con características diferentes a los turistas nacionales.
* Un mayor número de vuelos en un mes indica más turismo general, lo que puede influir en las estrategias de promoción y ofertas de determinados hoteles, influyendo en última instancia en su posición en el buscador Booking.

# Leemos los datos de vuelos  
flights <- read.csv("avia\_tf\_apal\_linear.csv.gz")  
summary(flights)

## DATAFLOW LAST.UPDATE freq unit   
## Length:1502955 Length:1502955 Length:1502955 Length:1502955   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
##   
##   
##   
## tra\_meas airline rep\_airp TIME\_PERIOD   
## Length:1502955 Length:1502955 Length:1502955 Length:1502955   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
##   
##   
##   
## OBS\_VALUE OBS\_FLAG   
## Min. : 0 Mode:logical   
## 1st Qu.: 0 NA's:1502955   
## Median : 264   
## Mean : 75333   
## 3rd Qu.: 3166   
## Max. :289204357

Para añadir dicha información hacemos un tratamiento de los datos obtenidos en Eurostat a fin de obtener una media de vuelos por mes para los últimos 5 años. Una vez obtenidos, tan solo debemos hacer un merge con nuestro dataset original.

proc\_flights <- flights %>%   
 # Filtramos para quedarnos solo con los aeropuertos de interés  
 # Código OACI/ICAO: Barcelona --> ES\_LEBL; Valencia --> ES\_LEVC; Madrid: ES\_LEMD  
 filter(rep\_airp %in% c("ES\_LEBL", "ES\_LEVC", "ES\_LEMD")) %>%   
 # Filtramos para quedarnos solo con los datos de carga de pasajeros  
 filter(tra\_meas == "PAS\_CRD") %>%  
 # Nos interesa solo los datos mensuales, así que filtramos por ellos  
 filter(freq == "M") %>%  
 separate(TIME\_PERIOD, c("YEAR", "MONTH"), sep = "-") %>%  
 # Nos quedamos solo con los últimos 5 años y los meses de Marzo, Junio y Diciembre  
 filter(YEAR %in% c("2022", "2021", "2020", "2019", "2018") & MONTH %in% c("03", "06", "12")) %>%  
 # No nos interesa los datos por aerolínea, solo los generales  
 filter(airline == "TOTAL") %>%  
 # Modificamos la columna de MONTH para tener los nombres del mes y cambiamos el código del aeropuerto por el nombre de la ciudad  
 mutate(month\_name = ifelse(MONTH == "03", "March", ifelse(MONTH == "06", "June", "December")),  
 city\_airp = ifelse(rep\_airp == "ES\_LEBL", "Barcelona", ifelse(rep\_airp == "ES\_LEVC", "Valencia", "Madrid"))) %>%  
 # Agrupamos por ciudad y mes y calculamos la media de vuelos.  
 group\_by(rep\_airp, month\_name) %>%  
 mutate(mean\_fligths = mean(OBS\_VALUE)) %>%  
 ungroup() %>%  
 select(c(city\_airp, month\_name, mean\_fligths))  
  
# Nos quedamos con los registros únicos  
final\_flights <- unique(proc\_flights)  
  
# Hacemos un merge con bookings  
booking <- booking %>%  
 merge(final\_flights, by.x = c("city", "month\_in"), by.y = c("city\_airp", "month\_name"))

## ANÁLISIS DE LOS DATOS

Añadir:

* Que scores influyen más (están más correlacionadas) con hotel\_score: Esto se puede hacer mediante una regresión lineal o una simple correlación de Pearson. Antes hay que comprobar normalidad y homocedasticidad. Podemos incluir un análisis del VIF para plantear que hay scores intercorrelacionadas entre ellas (un score como combinación lineal de otras)
* Comprobar si un feature influye en que se tenga una media de hotel\_score mayor o menor. Esto se puede hacer con un t-test, pero hay que comprobar antes la condición de normalidad.
* Para comprobar normalidad hacer uso de qqplots y el test kolmogorov-smirvo o el lilliefors.
* Finalizar con una regresión logística generalizada para determinar que variables influyen en la página donde aparece el hotel y un regresión lineal múltiple. Para la regresión logística podemos hacer uso de curvas ROC y para la regresión lineal múltiple podemos hacer un backward selection en base al p-value.

## RESOLUCIÓN DEL PROBLEMA