Práctica 2: Booking Data Cleaning

Gerard Alcalde and Guillem Rochina

2023-01-07

1. DESCRIPCIÓN DEL DATASET

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

1.1. Descripción general

El conjunto de datos a 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?

Asimismo, la fijación de precios o "pricing" puede ser una tardea ardua en los últimos años, pues la aparición de apartamentos vacacionales a través de plataformas como AirBnB ha incrementado de manera significativa la competencia a la que se deben enfrentar los propietarios de los hoteles, lo que influye en última instancia los precios a establecer por sus servicios. Dicho esto, ¿Qué precios se deben fijar dadas las características de un hotel? ¿Añadir un servicio adicional permitiría a un hotel elevar los precios de sus habitaciones?

Ante las preguntas presentadas en los párrafos anteriores, el objetivo de este proyecto es dual. Por un lado, se busca obtener las características que hacen que un hotel esté mejor posicionado que otro en el metabuscador de Booking.com y, por el otro, se persigue desarrollar un modelo que indique qué precios se deberían establecer dadas las características de una habitación y qué elementos pueden incrementar el precio de forma relevante.

En conclusión, el presente proyecto pretende dar continuidad a la Práctica 1, explotando los datos que se obtuvieron mediante web scrapping para ayudar a superar dos de los desafíos más relevantes para el sector hotelero.

2. LIMPIEZA DE LOS DATOS

Dado que muchas de las variables que desecharemos en la sección de selección e integración las vamos a utilizar a continuación, procedemos a realizar previamente la limpieza de los datos. A fin de iniciar el proceso, 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.

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 la limpieza de estas variables.

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

var	clase
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. En esta sección nos encargamos de tratar dichas columnas a fin de obtener nuevas variables con un formato adecuado, eliminando a su vez cualquier información que no nos sea de utilidad.

La columna hotel_coordinates incluye tanto los valores de latitud como los de longitud, separados por una coma. Dado que ambos dos valores nos indican 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.

```
booking <- booking %>% separate(hotel_coordinates, c("latitude", "longitude"), sep = ",", remove = FALSE)
```

Por su parte, hotel_scores alberga un diccionario en cada uno de los registros. Cada diccionario consta del nombre de cada dimensión (característica 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 en base a las comas como separador y, una vez obtenidas todas las columnas resultantes, extraer los datos numéricos de cada registro para quedarnos solo con los scores de cada dimensión con la función “parse_numbers”. Este proceso puede consultarse en el primer subapartado del Anexo.

Siguiendo adelante con la limpieza, dado que de las fechas tan sólo nos interesa el mes en el que se realiza la reserva (un alojamiento con unas características determinadas podría estar mejor posicionado en un determinado mes que en otro o tener un precio distinto) procedemos a extraer dicha información de las variables check-in y check-out. De hecho, dado que los datos que hemos extraído corresponden al mismo mes en todos los registros, tan sólo nos quedaremos con una columna de mes. Asimismo, no crearemos una columna is_june, pues la dejaremos como caso base, es decir, cuando no sea ninguna de las otras dos opciones (o Diciembre o Marzo).

```
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", NA), sep = "-", remove = FALSE) %>%
  mutate(is_december = ifelse(month == "December", 1, 0),
         is_march = ifelse(month == "March", 1, 0))
```

A continuación, 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.

```
# Extraemos los valores de codigo postal de la columna addres utilizando expresiones 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 exista) como servicios adicionales de hotel consideramos interesantes.

Por otro lado, nos encontramos con la columna que incluye la descripción del hotel, esta descripción es muy amplia y un análisis profundo de ella requeriría de técnicas de NLP, las cuales no son el objetivo de esta práctica. No obstante, consideramos que la longitud de la descripción del hotel sí que puede tener una relación con la posición en el buscador, por lo que transformaremos esta variable a una nueva variable que contenga el número total de palabras que contiene la descripción.

Finalmente, en la columna room_data, se incluye mucha información referente a los tipos de habitaciones disponibles, precios, características, etc. Esta se había extraído de esta forma con el objetivo de procesarla mediante un diccionario de Python. No obstante, al haber optado por un procesado con R este se complicará un poco más. En esta columna tenemos distintos datos que pueden ser muy relevantes, de los cuales extraeremos los siguientes:

- **min_price**: Precio de la habitación más económica.
- **max_price**: Precio de las habitaciones más caras del hotel.
- **suite_available**: Una de las habitaciones es de tipo suite.
- **is_apartment**: Una de las habitaciones es de tipo apartamento.
- **free_cancelation_available**: Alguna de las habitaciones tiene cancelación gratuita.

Este proceso de limpieza es el más largo de todos, por lo que también lo añadimos en el segundo subapartado del anexo.

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), min (45) y max_price (17) y el resto de las variables relacionadas con los scores (60) presentan valores faltantes, destacando free_wifi_score con 423 registros faltantes, consecuencia con toda seguridad de la ausencia de este servicio en dichos hoteles.

	Datos faltantes
postal_code	19
hotel_coordinates	0
latitude	16
longitude	16
hotel_score	8
hotel_scores	0
staff_score	60
facilities_score	60
cleanliness_score	60
comfort_score	60
value_for_money_score	60
location_score	60
free_wifi_score	423
hotel_description	0
min_price	45
max_price	17

Por lo que respecta a la variable postal_code encontramos que en todos aquellos registros donde se encuentan NAs, también hallamos datos faltantes en la mayoría del resto de columnas. Por tanto, dado que dichos registros no nos aportan información de valor, procedemos a eliminarlos del dataset.

	city	month	postal_code	longitude	hotel_score	staff_score	facilities_score
202	Barcelona	December	NA	NA	-1	NA	NA
203	Barcelona	December	NA	NA	-1	NA	NA
223	Barcelona	December	NA	NA	-1	NA	NA

Si comprobamos de nuevo qué columnas presentan todavía datos faltantes, observamos que en algunos casos (como longitude) hemos logrado eliminar todos los datos faltantes, mientras que en el resto se han reducido en 16 unidades el número de registros con NAs, pues como se había comentado todos aquellos registros con ausencia de postal_code, también carecía del resto de datos.

	Datos faltantes
postal_code	0
hotel_coordinates	0
latitude	0
longitude	0
hotel_score	5
hotel_scores	0
staff_score	44
facilities_score	44
cleanliness_score	44
comfort_score	44
value_for_money_score	44
location_score	44
free_wifi_score	407
hotel_description	0
min_price	29
max_price	1

A continuación, analizamos los datos faltantes de hotel_score. En este caso, encontramos que la ausencia de datos se registra de dos formas distintas, o bien con la introducción de NAs o bien con el valor -1. Por tanto, dado que el tratamiento en ambos casos va a ser el mismo, transformamos todos los valores -1 en NAs para poder llevar a cabo el siguiente proceso de inputación.

	city	month	hotel_score	staff_score	location_score	facilities_score	cleanliness_score
310	Barcelona	December	NA	9.4	9.4	8.6	8.9
844	Valencia	December	NA	7.7	9.5	7.2	7.5
958	Valencia	December	NA	9.2	9.3	9.1	9.3

	city	month	hotel_score	staff_score	free_wifi_score	free_wifi
3690	Madrid	June	-1	NA	NA	1
3736	Madrid	March	-1	NA	NA	0
3855	Madrid	March	-1	NA	NA	1

Dado que en este caso tan sólo encontramos datos faltantes en las columnas de score, no consideramos eliminar estos registros, sino llevar a cabo una imputación. Aunque se podría haber utilizado una metología más simple como el uso de la media o la mediana, hemos optado por hacer uso de un método más complejo, basado en la similitud entre registros, es decir, una imputación mediante el algoritmo de K vecinos más próximos. En este caso, la

función se encarga de devolver los mismos valores en caso de que ya existan en el dataset y el valor de los registros más “similares” en el caso de un registro con NAs.

```
# Imputamos los datos con K Neirest Neighbours
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$location_score <- kNN(booking)$location_score
booking$free_wifi_score <- kNN(booking)$free_wifi_score
```

En el caso de los precios, encontramos NAs en aquellos registros que presentaban valores iguales a 0 (no son valores lógicos). En este caso también encontramos que los únicos valores faltantes en estos registros corresponden a la columna de precio, por lo que procedemos de la misma forma que con los scores.

	city	month	hotel_score	max_price	min_price
3620	Madrid	June	9.5	903	NA
3698	Madrid	June	8.2	964	NA
3846	Madrid	March	8.8	981	NA
3847	Madrid	March	8.6	960	NA

Una última comprobación nos revela que el dataset ya no cuenta con datos faltantes, con independencia del formato que pudieran presenta (strings vacías, valores de -1 en integers, o simplemente NAs).

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 rápido y común se basa en el uso de boxplots. 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 **los scores del hotel**, **length_description**, **min_price** y **max_price**.

Empezamos analizando el número de habitaciones y observamos que en las 3 ciudades el número oscila entre 2 y 3 habitaciones para las búsquedas realizadas, lo cual son números razonables, aunque en Valencia esta aparezca como un outlier. Por lo tanto, aceptamos los valores outliers observados en la ciudad de Valencia como valores correctos.

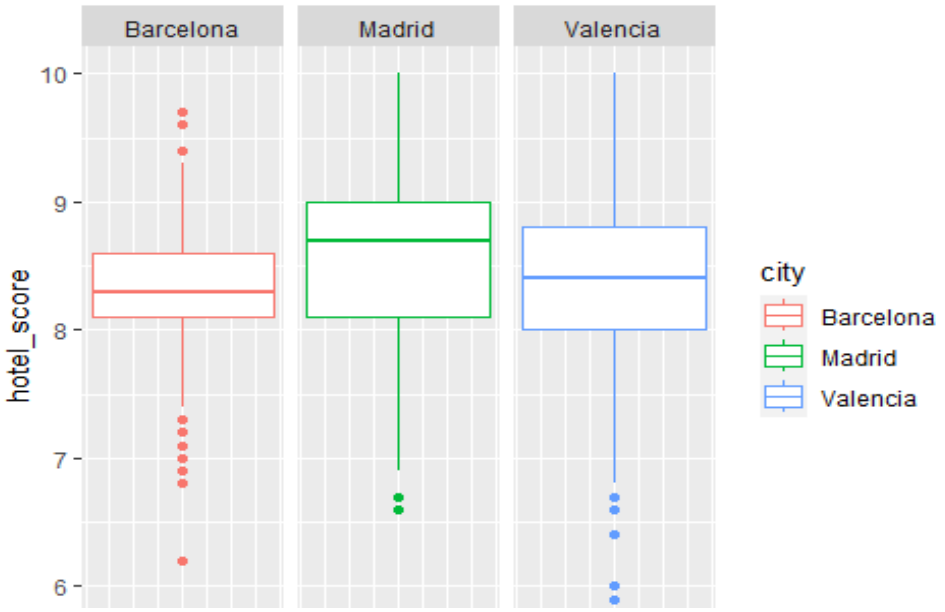
A continuación, se analiza la ubicación de los hoteles mediante su longitud y latitud. En el caso de Valencia vemos una mayor dispersión y tenemos la presencia de algunos outliers, esto es debido a que probablemente, al ser una ciudad más pequeña que Madrid o Barcelona tenga una menor disponibilidad hotelera y algunos de los hoteles hallados no pertenezcan a Valencia ciudad sino a los alrededores de esta. Para la longitud obtenemos unos resultados parecidos, pues hay una menor dispersión en los outliers de Valencia. Dado que estas variaciones tan pequeñas en los resultados son razonables, aceptamos los resultados como válidos.

En cuanto hotel_score, por un lado observamos que la puntuación general del hotel oscila entre el 8 y el 9, siendo mayor en Madrid que en Barcelona o Valencia. También nos encontramos que hay outliers en las 3 ciudades, no obstante, todos se hayan en una puntuación entre el 5 y el 10, cuando el rango factible es entre 0 y 10, por lo que pueden ser valores anómalos de hoteles que han obtenido una puntuación más baja de lo normal, pero

no por ello deben ser descartados. A continuación se presenta a modo de demostración general como se ha realizado el análisis de esta variable:

```
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
[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
[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
[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
```

Analizando el resto de scores, se observa que los hoteles con una mala puntuación lo hacen en la mayoría de sus características, lo que corrobora los resultados obtenidos.

Los hoteles con una puntuación de 10 destacan por obtener esta puntuación en todos los aspectos. Dado que solo corresponde a 3 registros y que los resultados son razonables, puede corresponder a un hotel nuevo con muy pocas valoraciones, lo que permitiría una nota excepcionalmente alta otorgada por unos pocos clientes.

hotel_score	facilities_score	cleanliness_score	location_score
6.9	7.0	7.4	8.0
7.0	6.8	7.3	8.8
5.9	5.9	5.9	7.6

El mismo comportamiento lo observamos en el resto de scores, excepto en algunos casos particulares. De los resultados obtenidos sorprende uno en Valencia que tiene unas puntuaciones muy elevadas excepto en **staff_score** la cual es bastante baja. Si nos fijamos estas puntuaciones son muy exactas (10.0, 9.0, 7.5 o 5.0) lo cual puede venir dado por un número de puntuaciones muy reducidas en las que un cliente ha puntuado de forma muy negativa este aspecto. No por ello eliminaremos el registro o lo modificaremos ya que es razonable el resultado, y afectará en la posición que presenta el hotel en el buscador.

Por su parte, las descripciones tienen una longitud muy similar en las 3 ciudades, en torno a las 120-170 palabras, presentando algún outlier por la parte superior, pero siempre con descripciones inferiores a las 400 palabras. Aunque corresponda a una descripción un poco

extensa, no parece que corresponda a un outlier sino simplemente a una descripción más extensa de lo habitual.

A continuación, se evalúan los precios máximos. Observamos unos precios promedio mucho más elevados con un mayor rango intercuartílico en la ciudad de Valencia, lo que sorprende al ser lo contrario que con los precios mínimos. También se observa que los outliers en este caso aparecen por el lado inferior especialmente en la ciudad de Madrid, aunque con precios positivos, que podrían corresponder a hoteles muy económicos o albergues.También se observa que los outliers en este caso aparecen por el lado inferior especialmente en la ciudad de Madrid, aunque con precios positivos, que podrían corresponder a hoteles muy económicos o albergues. Sorprende ver que los precios máximos no superan el valor máximo observado en el precio mínimo, lo que se puede atribuir a que aquellos hoteles con precios muy elevados no ofrecían una variedad de habitaciones y precios.

3. INTEGRACIÓN Y SELECCIÓN

La sección de integración y selección la encontramos tras la de limpieza, pues muchas de las columnas que queríamos descartar han sido tratadas en la fase de limpieza para extraer información de utilidad. Hecho esto, en la presente sección procedemos a eliminar todas las columnas del dataset original que ya no vamos a necesitar para los siguientes apartados.

```
booking <- booking %>%
  dplyr::select(-c("name", "search_date", "hotel_coordinates", "hotel_scores", "check.in",
"check.out", "address", "features", "hotel_description", "room_data"))
```

Por otro lado, hemos optado 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 o los precios de dichos alojamientos.
- 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 y, a su vez, directamente en los precios de los hoteles.

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, proceso que puede observarse en el subapartado 3 del Anexo. Una vez obtenidos, tan solo debemos hacer un merge con nuestro dataset original.

4. ANÁLISIS DE LOS DATOS

A continuación, se evaluará la influencia de las distintas variables sobre la posición de un hotel en el buscador y sobre el precio que este fija para sus habitaciones. Para ello se empleará una matriz de correlaciones. Una vez obtenidas las variables con mayor correlación con la variable objetivo llevaremos a cabo una selección de los atributos a analizar, un grupo de variables distinto por cada problema a tratar.

Tras generar los distintos grupos procederemos a evaluar la normalidad y la homogeneidad de la varianza para así poder determinar en el siguiente punto que pruebas estadísticas se pueden aplicar.

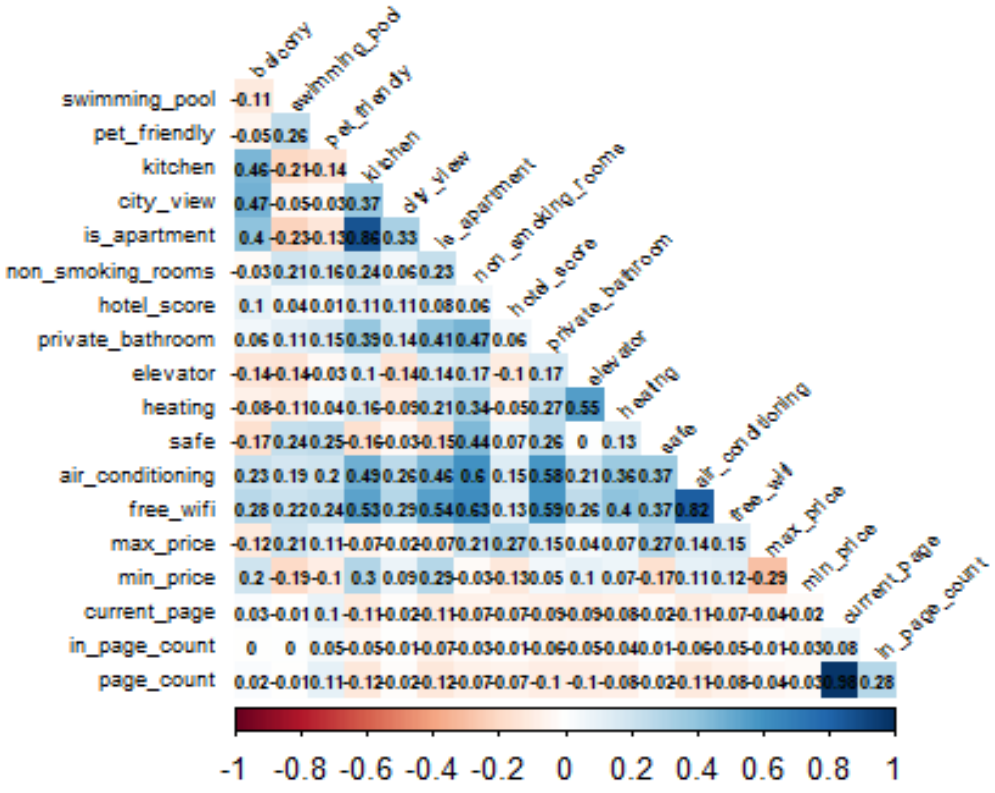
Finalmente aplicaremos distintos tests estadísticos adicionales a la correlación hallada inicialmente. Estas pruebas estadísticas incluirán contrastes de hipótesis y regresiones lineales (enfocados dese un punto de vista inferencial, no predictivo).

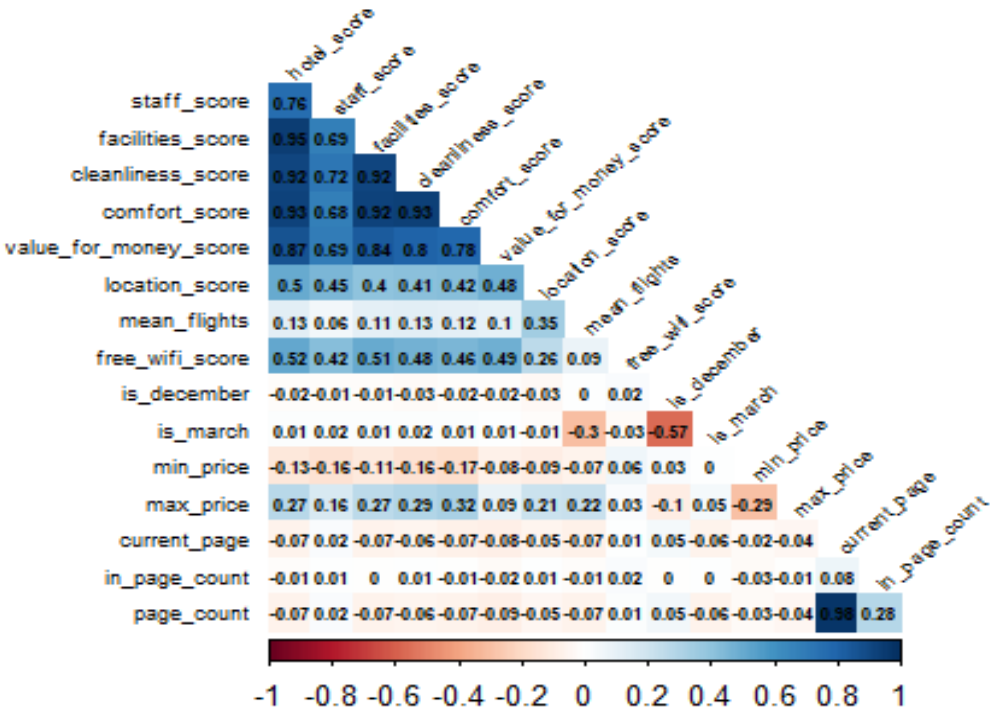
4.1 Correlación con la variable objetivo y selección de los grupos de datos que se quieren analizar

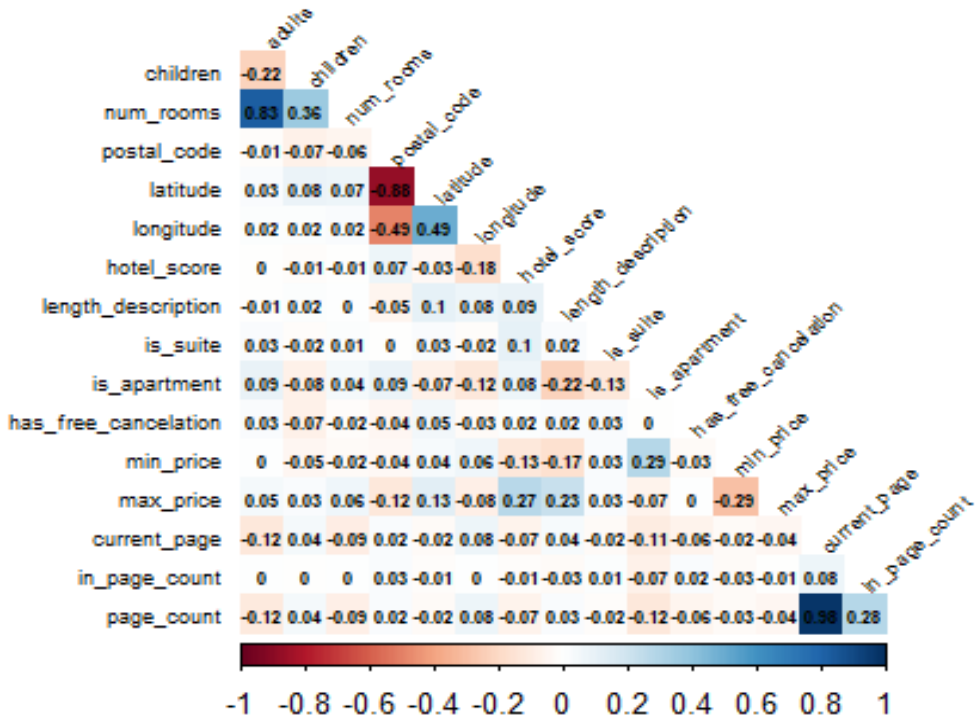
Comenzamos evaluando la correlación (de Spearman, dado que no podemos asumir normalidad y utilizamos variables categóricas también) entre las distintas variables con nuestra variable objetivo, para ello creamos varias matrices de correlación y evaluamos los resultados obtenidos.

```
cor_mat <- cor(mattmp, method = "spearman")
plot.new()
plot.window(xlim=c(-2,2), ylim=c(5,10))
corrplot(cor_mat, method="color", type = "lower", tl.col = "black", tl.srt = 45, d
iag = FALSE, tl.cex = 0.55, addCoef.col = 'black', number.cex = 0.5)
```

Como bien se puede observar en las matrices, las variables que recogen las distintas puntuaciones están correlacionadas entre ellas, por lo que un hotel que haya obtenido una puntuación elevada en un aspecto generalmente también lo obtendrá en el resto. Este factor es de especial interés si se seleccionan varias columnas de este tipo, pues corremos el riesgo de sufrir un problema de multicolinealidad. También encontramos que los hoteles con distintos atributos como aire acondicionado o baño privado suelen tener el resto de los servicios adicionales, aunque este factor puede deberse a que es más habitual encontrar este tipo de elementos en una habitación de hotel, por lo que es común que cuando exista una habitación con calefacción, por ejemplo, también tenga aire acondicionado o baño privado. Por último, aunque es evidente, cabe destacar que las mayores correlaciones las tenemos con variables directamente relacionadas o que presentan información de la misma fuente (current_page con page_count o max_price con min_price). No obstante, esto no es un problema, pues eliminaremos estas variables en los ajustes para no incurrir en “data leakage”.







Cuando evaluamos la posición en la página web observamos que no hay una correlación elevada con ninguna variable, encontrando valores por debajo de 0.12 (en valor absoluto) para todos los casos. No obstante, para el caso de page_count, hemos optado por someter a estudio aquellas variables que presentan mayor correlación de spearman con la variable dependiente, pues evaluando con distintas métricas que buscan determinar el nivel de dependencia no-lineal hemos obtenido resultados similares. Las variables seleccionadas en cuestión son: **kitchen, is_apartment**, **adults, air_conditioning, elevator, private_bathroom, num_rooms, value_for_money_score, heating, free_wifi, non_smoking_rooms, comfort_score, hotel_score, longitude, facilities_score pet_friendly**, mientras que la variable dependiente será **page_count.**

	page_count
kitchen	-0.1202309
is_apartment	-0.1201589
adults	-0.1158075
air_conditioning	-0.1140043
elevator	-0.0986088
private_bathroom	-0.0966138
num_rooms	-0.0866266
value_for_money_score	-0.0857426
heating	-0.0839920
free_wifi	-0.0752942
non_smoking_rooms	-0.0742490
mean_flights	-0.0695251
comfort_score	-0.0693636
hotel_score	-0.0676555
facilities_score	-0.0650869
longitude	0.0760104
pet_friendly	0.1053517

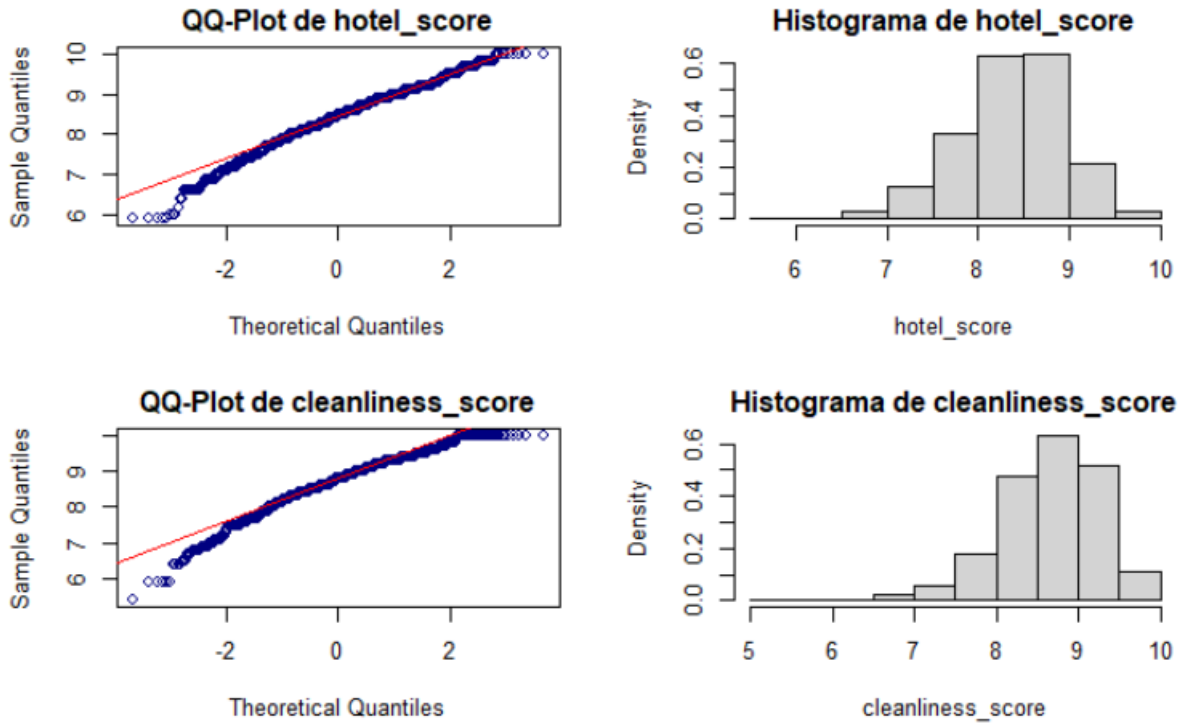
En el caso del problema a resolver relacionado con el precio, al analizar las correlaciones sí que encontramos valores entre 0.15 y 0.20. Aunque estas correlaciones no sean muy elevadas nos ofrecen un escenario más prometedor que el anteriormente presentado. En este caso, las variables más relevantes y las que seleccionaremos son: **postal_code, balcony, pet_friendly, latitude, air_conditioning, private_bathroom, free_wifi, staff_score, swimming_pool, location_score, non_smoking_rooms, mean_flights, length_description, facilities_score, hotel_score, safe, cleanliness_score, comfort_score.**

	max_price
postal_code	-0.1244939
balcony	-0.1185999
pet_friendly	0.1073362
latitude	0.1271236
air_conditioning	0.1408840
private_bathroom	0.1452074
free_wifi	0.1523803
staff_score	0.1640168
swimming_pool	0.2055779
non_smoking_rooms	0.2110602
location_score	0.2116932
mean_flights	0.2224695
length_description	0.2287604
facilities_score	0.2667483
hotel_score	0.2700575
safe	0.2719657
cleanliness_score	0.2919506
comfort_score	0.3206842

4.2 Comprobación de la normalidad y homogeneidad de la varianza

Para comprobar si los valores de las variables de nuestro dataset se distribuyen o se aproximan a una población distribuida normalmente haremos un análisis visual, con QQ-Plots e Histogramas, y un análisis mediante el test Shapiro-Wilk. Concretamente, solo comprobaremos la normalidad para las variables seleccionadas en el apartado anterior.

```
par(mfrow=c(2,2))
for(i in 1:ncol(booking_norm)){
  qqnorm(booking_norm[,i], main = paste("QQ-Plot de ",colnames(booking_norm)[i]),
col = "navy")
  qqline(booking_norm[,i], col = "red")
  hist(booking_norm[,i], main = paste("Histograma de ", colnames(booking_norm)[i])
, xlab=colnames(booking_norm)[i], freq = FALSE)
}
```



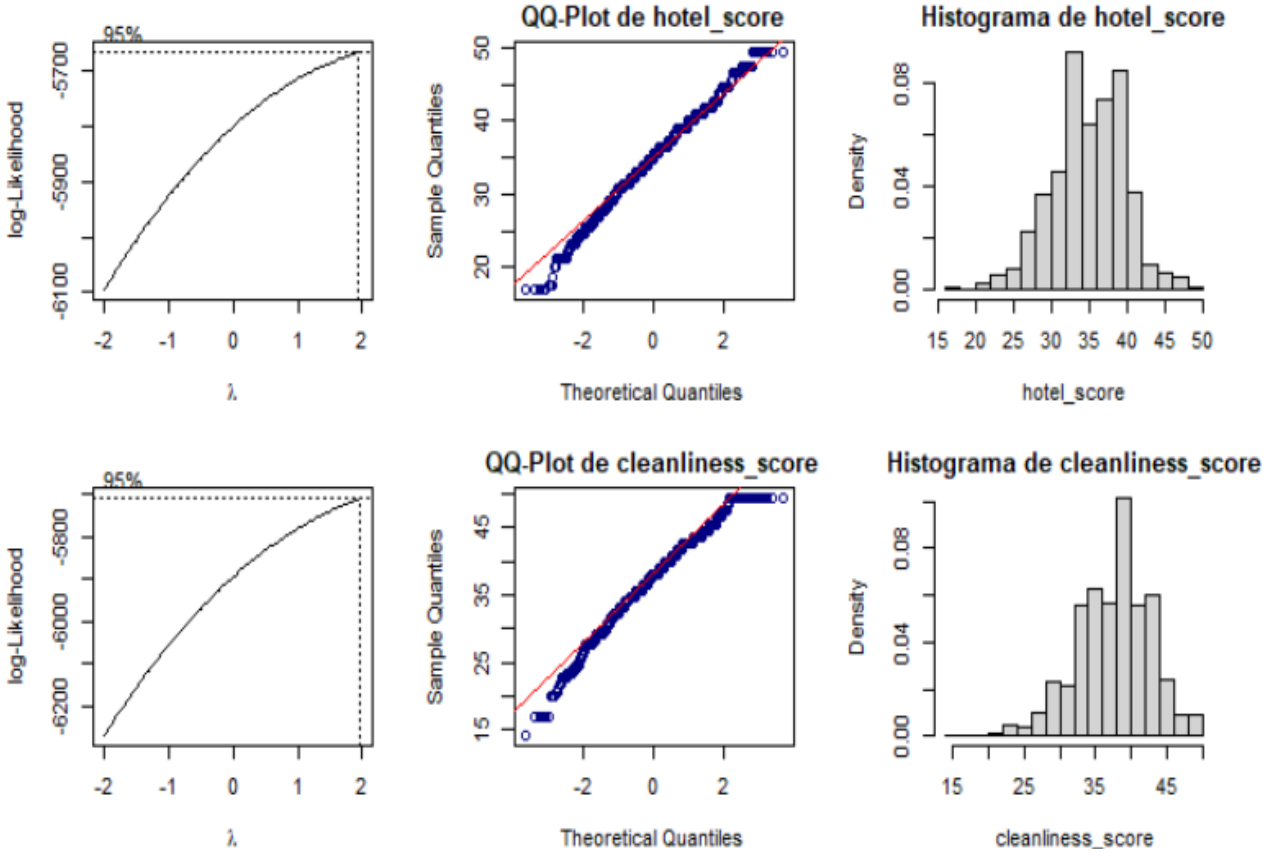
Los QQ-plots y los histogramas nos muestran unas distribuciones con una evidente asimetría negativa en la mayoría de los casos (excepto variables como length_description que muestran una asimetría más bien positiva, aunque podría ser influencia de un dato extremo que no ha sido considerado outlier), así como un evidente problema con las colas debido a esta, pues en la mayoría de los casos los puntos se alejan de la línea roja, la cual corresponde a la hipotética distribución normal. Por tanto, a priori podríamos rechazar la hipótesis de normalidad de las variables.

Es más, tras ejecutar el test Shapiro-Wilk confirmamos de forma más fiable nuestras primeras conclusiones, pues todos los p-valores muestran valores muy inferiores a 0.05, rechazando en todos los casos la hipótesis nula sobre la normalidad de los datos.

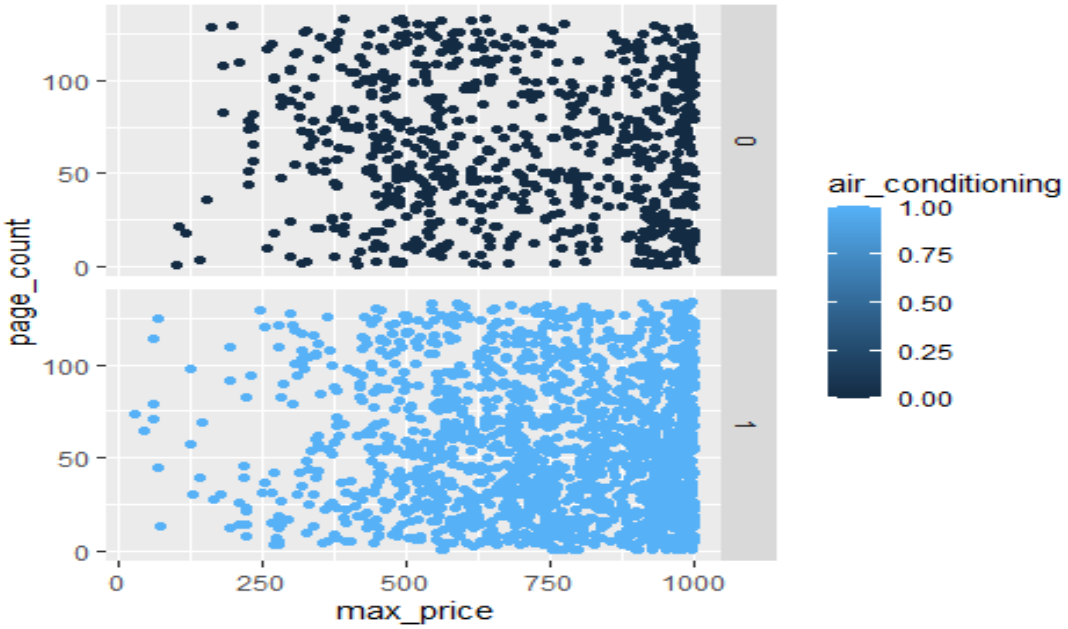
	p_values
value_for_money_score	1.23e-18
comfort_score	9.62e-22
hotel_score	8.63e-21
staff_score	8.08e-32
cleanliness_score	2.19e-26
location_score	8.67e-46

	p_values
facilities_score	1.54e-15
length_description	6.06e-33
min_price	6.87e-35
max_price	3.71e-40
page_count	2.73e-34

A fin de aproximar más los datos a una distribución normal, se propone hacer una transformación de box-cox y comprobar si los datos efectivamente se aproximan más a una normal. Tras llevar a cabo este proceso, si bien los histogramas muestran en algunos casos una distribución "visualmente más normales" y los QQ-plots parecen mostrar que la transformación ha aliviado ligeramente el problema con las colas, los resultados de la transformación no han sido suficientes como para poder considerar que los datos se distribuyen como una normal. No obstante, no debemos preocuparnos demasiado por este problema, pues tenemos suficientes registros como para poder apoyarnos en el Teorema del Límite Central. Por ejemplo, el t-test asume que las medias de las diferentes muestras se distribuyen normalmente (siendo una muestra de más de 30 registros suficiente, como rule of thumb, para que el t-test sea válido, aunque los datos no se distribuyan normalmente).



Seguidamente, analizaremos la homocedasticidad mediante scatterplots y varios test de homocedasticidad. En este caso, no podemos utilizar el test de Barlett ya que son muy sensibles ante datos no normales y, como acabamos de presentar, los datos analizados no se aproximan en ningún caso a una normal, por lo que utilizaremos el test no paramétrico de Fligner-Killeen.



Como todos los test de homogeneidad de varianzas, nos basamos en las siguientes hipótesis:

$$H_0: \sigma_i^2 = \cdots = \sigma_j^2$$

$H_0: \sigma_i^2 \neq \sigma_j^2$ para al menos un conjunto (i,j)

Donde rechazaremos la hipótesis nula cuando el pvalor del test sea inferior a un nivel de significación igual a 0.05. A continuación, presentamos código de ejemplo y el output resultante:

```
fligner.test(page_count ~ high_max_price, data = booking)

##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  page_count by high_max_price
## Fligner-Killeen:med chi-squared = 0.22846, df = 1, p-value = 0.63
```

Tras realizar tests a un gran número de variables de interés (principalmente las que van a ser sujeto de estudio para los tests estadísticos), se ha determinado que existe homogeneidad de varianzas, por lo que no nos encontraremos con problemas de heterogeneidad a la hora de realizar contrastes de hipótesis.

4.3 Aplicación de pruebas estadísticas para comparar los grupos de datos

En este apartado se aplicarán distintas pruebas estadísticas para comparar los distintos grupos de datos de los que disponemos y ver cual es su relación con la variable objetivo page_count. Esto nos permitirá evaluar que variables tienen un impacto significativo sobre la variable objetivo y cuales no. A continuación, se recogen unas pocas hipótesis, aunque sería posible evaluar otras que también pareciesen interesantes. Dado el carácter académico de este trabajo nos centraremos exclusivamente en estudiar la influencia de las variables:

- **air_conditioning**
- **has_free_cancelation**
- **hotel_score**
- **max_price**

Las hipótesis que trataremos de verificar serán las siguientes:

1. Aquellos hoteles con aire acondicionado tendrán una mejor posición (**page_count** inferior) a los que no dispongan de este.
2. Los hoteles con cancelación gratuita tendrán una posición diferente a los que no dispongan de esta.
3. Los hoteles con una puntuación global superior a un 9 tendrán un mejor posicionamiento.
4. Los hoteles con un precio mínimo inferior a la media tendrán un mejor posicionamiento.

A continuación, se estudiará cada una de estas hipótesis y el resultado obtenido.

Hipótesis 1

La primera hipótesis evalúa la influencia de que un hotel disponga de aire acondicionado en el posicionamiento del hotel, suponiendo que un hotel con aire acondicionado tendrá un posicionamiento mejor, es decir menor **page_count** que un hotel que no disponga de aire acondicionado. A continuación, se plantea la hipótesis nula (H_0) y la alternativa H_1).

- H_0: La posición global de un hotel con aire acondicionado y uno sin serán iguales.
- H_1: La posición global de un hotel con aire acondicionado será mejor (inferior) a la de uno sin.

En otras palabras:

$$\boldsymbol{H_0}: \boldsymbol{\mu_0 = \mu_1}$$
$$\boldsymbol{H_1}: \boldsymbol{\mu_0 \leq \mu_1}$$

En este caso se trata de un **contraste de hipótesis de dos muestras independientes sobre la media con varianzas no conocidas**. Aunque previamente no se ha cumplido la normalidad de los datos, por el Teorema del Límite Central podemos considerar la normalidad de los datos y aplicar un test paramétrico. En este caso aplicaremos un **test unilateral por la izquierda**, suponiendo (por lo que se ha visto en el apartado anterior) que las varianzas son iguales, por lo que el estadístico de contraste será:

$$T = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\left(\frac{1}{n_1} + \frac{1}{n_2}\right) \cdot \frac{(n_1 - 1) \cdot S_1^2 + (n_2 - 1) \cdot S_2^2}{n_1 + n_2 - 2}}} \sim t_{n_1 + n_2 - 2}$$

En el subapartado 4 se puede observar el procedimiento con el que obtenemos los siguientes resultados:

[1] "The p-value obtained is: 5.33600165433122e-13"
[1] "The mean of the group 1 is: 55.4309293146753"
[1] "The mean of the group 2 is: 64.871609403255"

Se obtiene un p-value muy cercano a 0, y muy inferior a 0.05, por lo que podemos rechazar la hipótesis nula y considerar que **la presencia de aire acondicionado en un hotel mejorará el posicionamiento de este en el buscador de Booking**. Esto se observa también al obtener una media de posicionamiento inferior en más de 9 posiciones en aquellos hoteles con aire acondicionado.

Hipótesis 2

Esta segunda hipótesis trata de evaluar la influencia de la cancelación gratuita, la cual para aportar una mayor diversidad en los análisis realizados, puede estar asociada a una posición distinta (inferior o superior) que la de un hotel sin cancelación gratuita. Definimos esta hipótesis mediante las hipótesis nula (H_0) y alternativa (H_1) siguientes:

- H_0: La posición global de un hotel con cancelación gratuita y uno sin serán iguales.
- H_1: La posición global de un hotel con cancelación gratuita será distinta a la de uno sin.

En otras palabras:

$$\boldsymbol{H_0: \mu_0 = \mu_1}$$
$$\boldsymbol{H_1: \mu_0 \neq \mu_1}$$

En este caso se trata de un **contraste de hipótesis de dos muestras independientes sobre la media con varianzas no conocidas**, al igual que en el caso anterior por el TLC podemos aceptar la normalidad de los datos y aplicar un test paramétrico, en concreto un **test bilateral**.

```
[1] "The p-value obtained is:  0.000376181455513198"
[1] "The mean of the group 1 is:  57.6696284329564"
[1] "The mean of the group 2 is:  67.3128491620112"
```

Observamos que se obtiene un p-value inferior a 0.05 por lo que debemos rechazar la hipótesis nula y, por lo tanto, obtenemos que **un hotel con cancelación gratuita obtendrá un posicionamiento distinto al de un hotel sin cancelación gratuita**. En concreto se ha observado que los hoteles con cancelación gratuita obtienen un mejor posicionamiento que aquellos sin.

Hipótesis 3

A continuación, evaluamos la influencia de la puntuación global del hotel. En este caso evaluaremos si un hotel con una puntuación global (**hotel_score**) superior o igual a un 9.0 obtiene un mejor posicionamiento que un hotel con una peor puntuación. A continuación, se plantea la hipótesis nula (H_0) y la alternativa (H_1)

Los hoteles con una puntuación global superior a un 9 tendrán un mejor posicionamiento.

- H_0: La posición global de un hotel con una puntuación global mayor o igual a 9.0 y uno con una puntuación inferior serán iguales.
- H_1: La posición global de un hotel con una puntuación global mayor o igual a 9.0 será mejor (menor) que uno con una puntuación inferior.

En otras palabras:

$$\boldsymbol{H_0: \mu_0 = \mu_1}$$
$$\boldsymbol{H_1: \mu_0 \leq \mu_1}$$

En este caso se trata de un **contraste de hipótesis de dos muestras independientes sobre la media con varianzas no conocidas**. Al igual que en los casos anteriores podemos asumir la normalidad de los datos por el TLC. En este caso se aplicará un **test bilateral por la izquierda**.

[1] "The p-value obtained is: 0.0483871639663181"
[1] "The mean of the group 1 is: 55.9346210995542"
[1] "The mean of the group 2 is: 58.5683229813665"

Se obtiene un valor del p-value de 0.048, el cual aun siendo muy cercano a 0.05 que utilizamos para el intérvalo de confianza es inferior, por lo que se debe rechazar la hipótesis nula y podemos decir que **un hotel con una puntuación superior a 9.0 obtendrá un mejor posicionamiento que el resto de hoteles**. No obstante, por el valor obtenido podemos determinar que esta variable no es tan influyente como el aire acondicionado o la cancelación gratuita.

Hipótesis 4

Finalmente evaluamos la influencia del precio máximo del hotel considerando que un hotel con un mayor precio máximo tendrá un peor posicionamiento. Para ello consideraremos las siguientes hipótesis nula (H_0) y alternativa (H_1)

Los hoteles con un precio máximo superior a la media tendrán un peor posicionamiento.

- H_0: La posición global de un hotel con precio máximo superior a la media y uno inferior serán iguales.
- H_1: La posición global de un hotel con precio máximo superior a la media será peor (superior) a la de uno con precio inferior.

O, en otras palabras:

$$\boldsymbol{H_0: \mu_0 = \mu_1}$$
$$\boldsymbol{H_1: \mu_0 \leq \mu_1}$$

En este caso también realizaremos un **contraste de hipótesis de dos muestras independientes sobre la media con varianzas no conocidas**. Como se ha visto en los casos anteriores, por el TLC podemos asumir la normalidad de los datos y aplicaremos un test paramétrico, en este caso emplearemos un **test unilateral por la derecha**.

[1] "The p-value obtained is: 0.979867959556137"
[1] "The mean of the group 1 is: 56.9824644549763"
[1] "The mean of the group 2 is: 59.4509254066181"

En este caso se ha obtenido un p-value de 0.98, por lo que se acepta la hipótesis nula y obtenemos que **los hoteles con un precio máximo superior a la media no obtienen un peor posicionamiento que el resto de los hoteles**. Es más, en este caso se ha observado que que la media de posicionamiento de los hoteles con un precio máximo superior a la media ha sido ligeramente mejor al resto.

4.4 Aplicación de funciones lineales para resolución del problema

Una vez analizados los distintos contrastes de hipótesis y hallada la correlación entre las distintas variables, tenemos un conocimiento del conjunto de datos lo suficientemente extenso como para analizar mediante una función lineal el posicionamiento de un hotel en el buscador de Booking.

Tras los resultados que se han obtenido observamos que hay distintas variables que pueden tener una mayor influencia sobre nuestra variable objetivo. Por un lado, si nos basamos en lo obtenido tras el análisis de correlación hallamos que variables como **min_price**, **comfort_score**, **cleanliness_score**, **safe** o **hotel_score** entre otros tienen una gran relevancia. Por otro lado, variables como **air_conditioning** o **has_free_cancelation** han mostrado una relevancia significativa durante el contraste de hipótesis.

Regresión Lineal Aplicada a page_count

Aproximación 1

En este apartado comenzaremos tratando de ajustar una regresión lineal simple aproximando la variable dependiente **page_count** con la variable explicativa que ha proporcionado una mayor correlación, que ha resultado ser **comfort_score**.

Obtenemos un valor de R-squared de 0.005, lo que es un valor muy bajo y representa una aproximación muy mala de la variable dependiente.

Aproximación 2

Con el objetivo de mejorar los resultados obtenidos se realizará una regresión lineal múltiple que considere todas las puntuaciones del hotel para predecir la posición de este. Para ello empleamos las variables: **hotel_score**, **staff_score**, **facilities_score**, **cleanliness_score**, **comfort_score**, **value_for_money_score**, **location_score** y **free_wifi_score**, para ver como la opinión de los clientes afecta en el posicionamiento de este en Booking.

Se observa que con la regresión lineal multiple R-squared aumenta hasta 0.019, algo mejor que empleando solo una variable, pero aun así los resultados obtenidos son muy malos y no aproximan de una forma correcta la variable dependiente.

De este primer análisis observamos que de todas las variables empleadas, las que han tenido una mayor influencia sobre la variable dependiente han sido **staff_score**, **value_for_money_score**, **location_store** y **free_wifi_score**.

Aproximación 3

A continuación, generamos una regresión lineal múltiple empleando tan solo las variables que han obtenido la mayor relación con la variable dependiente en los resultados de la regresión lineal múltiple anterior.

Observamos que, habiendo empleado tan solo 4 variables, de las 8 empleadas previamente, el R-squared tan solo se ha visto reducido en 0.0015.

Aproximación 4

A continuación, mantendremos estas variables que han mostrado los mejores resultados y añadiremos las que habíamos hallado que tenían una mayor correlación con la variable dependiente, que son: **min_price**, **balcony**, **pet_friendly**, **air_conditioning**, **private_bathroom**, **free_wifi**, **swimming_pool**, **non_smoking_rooms**, **mean_flights**, **length_description** y **safe.**

Con todas las variables la R-squared ha aumentado hasta 0.057, que sigue siendo un resultado muy bajo, pero dificil de mejorar con unos datos tan poco correlacionados con la variable objetivo.

De todas las variables empleadas se observa que las que tienen una mayor influencia son: **staff_score, free_wifi_score, balcony, pet_friendly** y **length_description** que influyen negativamente en el posicionamiento (aumentan las posiciones en las que aparece, es decir

más tarde) y, por otro lado, **air_conditioning, swimming_pool, mean_flights y value_for_money_score**, las cuales influyen positivamente (se aparece en posiciones más tempranas).

Aproximación 5

Finalmente, se aproxima la variable dependiente basándonos exclusivamente en los resultados del apartado previo.

De esta forma hemos conseguido mantener una R-squared similar a la obtenida previamente, pero empleando exclusivamente 9 variables en lugar de las más de 40 variables iniciales. El código de esta última regresión puede comprobarse en el subapartado 5.

Regresión Lineal Aplicada a max_price

Concluyendo esta última sección, concentramos nuestro análisis en torno a max_price. Anteriormente se ha observado una correlación entre el precio máximo de la habitación y sus características. Por tanto, seguidamente se tratará de predecir dicho precio mediante una regresión lineal múltiple empleando el resto de variables para ello:

Aproximación 1

Con ello obtenemos una R-squared de 0.2125, con lo que podemos observar que variables que tienen una mayor correlación con la variable objetivo nos proporcionarán unos resultados más apropiados. A partir de los resultados mostrados podemos mejorar la regresión empleando tan solo aquellas variables que han aportado más información en la regresión (aunque que son estadísticamente significativas) como han sido, por un lado **staff_score**, **location_score**, **free_wifi**, **swimming_pool**, **non_smoking_rooms**, **mean_flights**, **length_description** y **safe**, que influyen positivamente en el precio máximo y, por otro lado y aunque parezca sorprendente, **balcony** (esto podría ser debido a que es más común en apartamentos, y estos son más baratos que una habitación de hotel) y **air_conditioning**, que influyen negativamente en el mismo.

Aproximación 2

Para esta segunda aproximación tan solo se emplean las variables que han mostrado una mayor aportación a la regresión lineal. Se observa que, aun habiendo reducido el número de variables, mantenemos un coeficiente de determinación ligeramente superior 0.21. Hay que tener en cuenta que los resultados siguen sin ser satisfactorios, ya que una R-squared de 0.21 es bastante baja. No obstante, es mucho mejor que lo obtenido tratando de predecir la variable page_count. Esta mala predicción es producida como se había comentado previamente debido a la baja correlación de las variables con la variable objetivo.

5. REPRESENTACIÓN DE LOS RESULTADOS

A lo largo del documento se han presentado representaciones gráficas de los análisis realizados y de los resultados obtenidos.

6. RESOLUCIÓN DEL PROBLEMA Y CONCLUSIONES

Como se ha visto a lo largo del presente documento, se ha procedido a la limpieza del dataset obtenido a través del scrapeo de Booking.com, con la finalidad de intentar determinar qué variables influyen más en el posicionamiento y la determinación del precio de un hotel en dicha web.

En primer lugar, se ha llevado a cabo un profundo proceso de preprocesado de datos en el que se han creado un número significativo de variables, se han eliminado registros poco útiles o con muchos elementos vacíos o se han imputado valores en los casos oportunos y, finalmente, se ha analizado la existencia de outliers, los cuales no han sido eliminados dado que no se han considerado erróneos, pues los valores entraban dentro de la lógica de cada una de las variables.

Tras llevar a cabo distintas pruebas estadísticas, entre las que destacan el análisis de correlación y los contrates de hipótesis (incluyendo de normalidad y de homocedasticidad), y la culminación de la práctica con la aplicación de modelos de regresión lineal. Se obtienen distintas conclusiones:

Inicialmente, del análisis de correlación se ha observado que existen distintas variables correlacionadas entre ellas como, serían los servicios del hotel o las puntuaciones de las distintas categorías, pero ninguna de ellas parecía tener una correlación suficientemente fuerte con la variable objetivo.

Mediante el contraste de hipótesis se ha obtenido que determinadas variables como la presencia de aire acondicionado o la cancelación gratuita de la reserva sí que tenían un impacto sobre el posicionamiento de un hotel en el buscador, concretamente tener estos servicios implicaba, de media, un mejor posicionamiento en el buscador.

Finalmente, se ha tratado de obtener una regresión lineal múltiple que, a partir de distintas variables independientes, nos permitiese predecir la variable dependiente que indica la posición en la página del buscador. Este proceso ha sido complejo y ha resultado en unos resultados muy poco precisos, que aun habiendo conseguido mejorarse afinando el proceso, no servirían para una aproximación real. Esto es debido a una falta de correlación entre las variables dependientes y la objetivo, lo cual nos indica que hacer uso de modelos de regresión lineal limita la precisión de las predicciones que se puede alcanzar. No obstante, puesto que nuestra tarea era una aproximación inferencial, no predictiva, podemos considerar que nuestro proyecto resuelve, aunque de forma limitada, el problema planteado, pues de este proceso se ha obtenido que de las más de 40 variables iniciales que teníamos, con tan solo 9 de estas podemos desarrollar unas breves recomendaciones con respecto a cómo posicionar un hotel en Booking.

Como conclusión, tras el análisis realizado se ha obtenido que las variables que resultan ser más relevantes, influenciando de forma positiva o negativa en el posicionamiento de hotel son: **Staff_Score**, **Value_For_Money_Score, Free_Wifi_Score, Balcony, Pet_Friendly, Air_Conditioning, Swimming_Pool, Mean_Flights, Length_Description.**

Por otro lado, en cuanto al problema de pricing, hemos sido capaces de determinar que las variables que resultan más relevantes en la determinación del precio son: **Staff_Score**, **Location_Score**, **Free_Wifi**, **Swimming_Pool**, **Non_Smoking_Rooms**, **Mean_Flights**, **Length_Description**, **Safe, Balcony y Air_conditioning.** No obstante, nuestros resultados en cuanto a poder predictivo tienen un gran margen de mejoría.

Como sugerencias tras los resultados del análisis podemos decir que, dado un hotel con unas características determinadas, nuestra recomendación para mejorar su posicionamiento pasaría primero por mejorar la descripción del hotel haciéndola más corta y precisa a la vez que se enfoca en una estrategia de mejora de la calidad percibida con respecto al precio (value for money), ya que son dos medidas “gratuitas” que tendrán un gran impacto. En caso de querer invertir dinero se recomendaría hacer una instalación de una piscina o de aire acondicionado (aunque teniendo en cuenta su influencia en el precio se debería tener en consideración las prioridades del hotel).

Si, por otro lado, un hotel busca incrementar sus precios sería recomendable, aunque influenciando así negativamente en el posicionamiento, mejorar el trato al cliente por parte de los empleados e incluir otros servicios como Wifi gratuito o Piscina.

Contribuciones	Firma
Investigación previa	Gerard Alcalde, Guillem Rochina
Redacción de las respuestas	Gerard Alcalde, Guillem Rochina
Desarrollo del código	Gerard Alcalde, Guillem Rochina
Participación en el vídeo	Gerard Alcalde, Guillem Rochina

ANNEXO

Subapartado 1: Obtención de scores

```
# Definimos un vector con los nombres de las columnas a crear

columns <-  c("staff_score", "facilities_score", "cleanliness_score", "comfort_score", "val
ue_for_money_score", "location_score", "free_wifi_score")

# Separamos los strings (diccionarios) en función de las comas y creamos las columnas corre
spondientes. La primera columna creada es una NA porque todos los primeros registros en cad
a diccionario están vacíos (por un fallo en la extracción de datos). Estableciendo la colum
na como NA, indicamos a la función separate que ignore dichos datos y no cree columna algun
a para ellos.

booking <- booking %>%
  separate(hotel_scores, c(NA, "staff_score", "facilities_score", "cleanliness_score", "com
fort_score", "value_for_money_score", "location_score", "free_wifi_score"), sep = ",", remo
ve = FALSE)

## 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)
```

Subapartado 2: Obtención de información de room_data

```
# Función que encuentra el precio mínimo de la habitación dado un string con la información
de las habitaciones
find_min_price <- function(text) {
  if (text=="{}"){
    return(NA)
  }
  a <- gregexpr("([0-9]*', 'room_capacity)", text)

  min_price <- NA
  for (value in a[[1]]){
```

```
    substring <- substr(text, value, value+20)
    price <- as.numeric(strsplit(substring, "'")[[1]][1])
    if (is.na(min_price)) {
      min_price <- price
    }

    if (price < min_price) {
      min_price <- price
    }
  }
  if (min_price == 0){
    return(NA)
  }
  else {
    return(min_price)
  }
}

# Aplicamos la función a todos los valores
min_price_vector <- c()
for (i in seq(1, length(booking$room_data))) {
  min_price_vector <- append(min_price_vector, find_min_price(booking$room_data[i]))
}

# Guardamos los resultados en una nueva columna.
booking$min_price <- min_price_vector

# Función que encuentra el precio máximo de la habitación dado un string con la información
de las habitaciones
find_max_price <- function(text) {
  if (text=="{}"){
    return(NA)
  }
  a <- gregexpr("([0-9]*', 'room_capacity)", text)

  max_price <- NA
  for (value in a[[1]]){
    substring <- substr(text, value, value+20)
    price <- as.numeric(strsplit(substring, "'")[[1]][1])
    if (is.na(max_price)) {
      max_price <- price
    }

    if (price > max_price) {
      max_price <- price
    }
  }
  if (max_price == 0){
    return(NA)
  }
  else {
    return(max_price)
  }
}

# Aplicamos la función a todos los valores
max_price_vector <- c()
for (i in seq(1, length(booking$room_data))) {
  max_price_vector <- append(max_price_vector, find_max_price(booking$room_data[i]))
}

# Guardamos los resultados en una nueva columna.
booking$max_price <- max_price_vector

# Buscamos si hay una habitación en suite
is_suite_vector <- c()
for (i in seq(1, length(booking$room_data))) {
```

```
  is_suite <- gregexpr("(suite)", booking$room_data[i])[[1]][1]

  if (is_suite==-1) {
    is_suite_vector <- append(is_suite_vector, 0)
  } else {
    is_suite_vector <- append(is_suite_vector, 1)
  }
}

# Guardamos los resultados en una nueva columna.
booking$is_suite <- is_suite_vector

# Buscamos si hay opción de apartamento
is_apartment_vector <- c()
for (i in seq(1, length(booking$room_data))) {
  is_apartment <- gregexpr("(Apartment)", booking$room_data[i])[[1]][1]

  if (is_apartment==-1) {
    is_apartment_vector <- append(is_apartment_vector, 0)
  } else {
    is_apartment_vector <- append(is_apartment_vector, 1)
  }
}

# Guardamos los resultados en una nueva columna.
booking$is_apartment <- is_apartment_vector

# Buscamos si tiene cancelación gratuita
free_cancelation_vector <- c()
for (i in seq(1, length(booking$room_data))) {
  free_cancelation <- gregexpr("(Free cancellation)", booking$room_data[i])[[1]][1]

  if (free_cancelation==-1) {
    free_cancelation_vector <- append(free_cancelation_vector, 0)
  } else {
    free_cancelation_vector <- append(free_cancelation_vector, 1)
  }
}

# Guardamos los resultados en una nueva columna.
booking$has_free_cancelation <- free_cancelation_vector
```

Subapartado 3: Integración de datos de vuelos

```
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 cód
igo del aeropuerto por el nombre de la ciudad
        mutate(month_name = ifelse(MONTH == "03", "March", ifelse(MONTH == "06", "June", "D
ecember")),
               city_airp = ifelse(rep_airp == "ES_LEBL", "Barcelona", ifelse(rep_airp == "E
S_LEVC", "Valencia", "Madrid"))) %>%
        # Agrupamos por ciudad y mes y calculamos la media de vuelos.
        group_by(rep_airp, month_name) %>%
```

```
        mutate(mean_flights = mean(OBS_VALUE)) %>%
        ungroup() %>%
        dplyr::select(c(city_airp, month_name, mean_flights))

# 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"), by.y = c("city_airp", "month_name"))
```

Subapartado 4: Ejemplo de Contraste de Hipótesis

```
# Buscamos el conjunto de datos de estudio:
X1 <- booking %>%
  filter(has_free_cancelation==1) %>%
  dplyr::select("page_count")
X1 <- as.vector(unlist(X1),'numeric')

X2 <- booking %>%
  filter(has_free_cancelation==0) %>%
  dplyr::select("page_count")
X2 <- as.vector(unlist(X2),'numeric')

# Buscamos la varianza de ambas variables
var1 <- var(X1)
var2 <- var(X2)

# Definimos el nivel de confianza al 95%
alfa <- 0.05

# Buscamos las medias y la el número de registros en cada variable
mean1 <- mean(X1); n1 <- length(X1)
mean2 <- mean(X2); n2 <- length(X2)

# Calculamos el valor observado y el valor crítico
zobs <- (mean1-mean2)/sqrt(var1/n1 + var2/n2)
zcrit <- qnorm(1-(alfa/2))

# Finalmente calculamos el pvalue
pvalue <- pnorm(zobs, lower.tail=TRUE)
print(paste("The p-value obtained is: ", pvalue))

## [1] "The p-value obtained is:  0.000151290415318369"

# Mostramos las medias de las dos variables
print(paste("The mean of the group 1 is: ", mean(X1)))

## [1] "The mean of the group 1 is:  57.6696284329564"

print(paste("The mean of the group 2 is: ", mean(X2)))

## [1] "The mean of the group 2 is:  67.3128491620112"
```

Subapartado 5: Ejemplo de Modelo Lineal

```
# Creamos el modelo lineal
mrl_final <- lm(page_count~staff_score + value_for_money_score + free_wifi_score + balcony
+ pet_friendly + air_conditioning + swimming_pool + mean_flights + length_description, data
=booking)

#Evaluamos los resultados
summary(mrl_final)

##
## Call:
## lm(formula = page_count ~ staff_score + value_for_money_score +
```

```
##     free_wifi_score + balcony + pet_friendly + air_conditioning +
##     swimming_pool + mean_flights + length_description, data = booking)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -70.224 -30.654  -4.833  29.421  87.936
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             8.407e+01  1.021e+01   8.236 2.40e-16 ***
## staff_score             5.342e+00  1.351e+00   3.953 7.85e-05 ***
## value_for_money_score -1.159e+01  1.527e+00  -7.588 4.05e-14 ***
## free_wifi_score         3.085e+00  7.574e-01   4.073 4.73e-05 ***
## balcony                 6.608e+00  1.515e+00   4.361 1.33e-05 ***
## pet_friendly            1.171e+01  1.609e+00   7.276 4.14e-13 ***
## air_conditioning       -1.166e+01  1.399e+00  -8.332  < 2e-16 ***
## swimming_pool          -3.584e+00  1.867e+00  -1.919   0.0550 .
## mean_flights           -2.381e-06  4.876e-07  -4.884 1.08e-06 ***
## length_description      5.152e-02  2.046e-02   2.518   0.0118 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 36.42 on 3883 degrees of freedom
## Multiple R-squared:  0.05461,    Adjusted R-squared:  0.05242
## F-statistic: 24.92 on 9 and 3883 DF,  p-value: < 2.2e-16
```