Examen Parcial

Profesora: Patricia Reyes Silva

Integrantes del grupo: Natalia Maury Castañeda (u201816996)
Oscar Flores Palermo (u201716498)
Carlos Eduardo Iparraguirre Marujo (u201810601)

Fecha de entrega: 11 de mayo del 2021
Ciclo: 2021-1

Índice

Contenido

Objetivo

El objetivo de esta tarea académica es realizar un análisis exploratorio de un conjunto de datos, creando visualizaciones y obteniendo inferencias básicas utilizando R/RStudio como herramienta de software. Para ello se utilizará el dataset de Hotel Booking Demand.

Caso de análisis

Sobre el Dataset

El dataset a analizar es el de Hotel Booking Demanad, la versión original es de kaggle.com pero se ha modificado para agregar ruido en los datos. No obstante, el origen de los datos es el mismo y tiene los mismos creadores. El dataset contiene la información de reserva de hoteles, ya sean en una ciudad o un resort, que hacen los clientes. Incluye los detalles de cuándo se hizo la reserva, por cuánto tiempo se quedó, cantidad de personas que reservaron y el tipo de persona (adulto, niño, bebé, etc.), etc. El Dataset fue creado por Nuno Antonio, Ana Almeida, y Luís Nunes. Lo publicaron en el artículo "*Hotel Booking Demand Dataset*" para *Data in Brief*, perteneciendo al volumen 22 de esta página en el 2019 (Mostipak; 2020). Debido a que todos los autores pertenecen a Portugal y han trabajado en algún momento en las mismas instituciones al mismo tiempo, se asume que el artículo se publicó en Portugal pero no se especifica dicha información.

Los datos fueron recolectados a través de la extracción de las bases de datos SQL del sistema de gestión de propiedades (SGM) de hoteles. Para ello, se utilizaron consultas TSQL directamente ejecutadas en la base de datos del SGM de los hoteles. Para el análisis de los datos, se utilizaron comandos en r. Ambos hoteles examinados se encuentran en Portugal, específicamente en H1 en la región turística de Algarve está el hotel resort y H2 en la ciudad de Lisboa está el hotel de ciudad. La administración de los hoteles mencionados dio su permiso para que la información sea pública.

Sobre los autores

Data in Brief es un *journal* que provee datasets. Esto les permite a los investigadores utilizar los datos de otros investigadores y también publicar sus propios datasets en artículos. El objetivo de la compañía es incrementar el tráfico en investigaciones de datos, facilitar la reproducción de datos al estar bien explicada en artículos, hacer que las investigaciones sean más fáciles de encontrar y accesibles a las personas (como un google de datasets), crear posibilidades para colaboraciones. El motivo de los objetivos de *Data in Brief* se basa en la premisa de que uno no sabe qué datos pueden ser útil para una persona, por lo que desean crear una plataforma que permita encontrar dichos datos (Elsevier B.V, ScienceDirect; S/F).

Ana Maria de Almeida es doctora (no se especifica en qué tiene su doctorado) e investigadora que actualmente pertenece al Instituto Universitario de Lisboa en el departamento de ciencias de la computación como profesora asistente. Tiene experiencia como investigadora senior en

la Universidad de Coimbra y en ISTAR-IUL en el departamento de ciencias de la computación donde hizo una colaboración como investigadora principal. Sus habilidades están enfocadas en matemática discreta, programación matemática, modelado y optimización de modelado, resolución de problemas, heurística, reconocimiento de patrones y matemática aplicada. Ha hecho 58 publicaciones relacionadas a datos utilizando matemática discreta, algoritmos, etc., ha realizado 3 proyectos relacionados a analítica predictiva y seguridad a lo largo de su carrera (ResearchGate, S/F).

Luís Nunes es un ingeniero de software que actualmente pertenece al Instituto Universitario de Lisboa en el departamento de ciencias de la computación como profesor asistente. En cuanto a habilidades y experiencia, sabe programación orientada a objetos, machine learning avanzado, reconocimiento de patrones, robótica, redes neuronales e inteligencia artificial, y visión computacional. Ha realizado 47 publicaciones relacionadas a los datos mediante machine learning, analítica predictiva, etc., ha realizado 4 proyectos relacionados al análisis, analítica predictiva, seguridad y *urban sensing* a lo largo de su carrera (ResearchGate, S/F).

Nuno Antonio es un ingeniero de datos que actualmente pertenece a la Universidade NOVA de Lisboa como profesor asistente. Tiene experiencia como profesor asistente en el Instituto Universitario de Lisboa y ha hecho un doctorado en el Instituto Universitario de Lisboa en el departamento de información científica y tecnológica. Sus habilidades se enfocan principalmente en la ciencia de datos[1], project management, Data mining, y machine learning avanzado. Ha realizado 36 publicaciones relacionadas a la analítica predictiva, machine learning, entre otros y 2 proyectos sobre analítica predictiva a lo largo de su carrera (ResearchGate, S/F).

Casos de uso

Este dataset tiene distintos casos de uso dependiendo de la persona o compañía que vaya a usar los datos.Un caso de uso sería para las cadenas de hoteles saber cuál es el tiempo promedio que la gente se queda en el hotel, quienes son ese tipo de personas (adultos, familias, etc) y en base a ello hacer ofertas. Otra de ellas es que se puede ver en qué fechas la gente se queda más en los hoteles para realizar paquetes, promociones, etc. Otro caso de uso puede ser para las tiendas que están cerca a los hoteles para saber cuándo vendrá más gente y saber cuándo pueden vender más, hacer ofertas, hacia quienes dirigir sus productos, etc. Dependiendo del sujeto puede variar el caso de uso de este dataset.

La importancia de este análisis del dataset se basa en el caso de uso que le de ya que tendrá valor para dichas personas. Un ejemplo serían los dueños de hoteles, vendedores, páginas web como Trivago.com, etc. Todos los sujetos mencionados anteriormente, los datos de este dataset tienen un valor para ellos por lo que este análisis es importante e interesante para ellos. Mientras que si este mismo análisis se le muestra a un veterinario no le será de importancia, todo depende del sujeto que vea el análisis y el caso de uso que le dará.

[1] En cuanto a la ciencia de datos, las habilidades específicas son: extracción de la información, inteligencia del negocio, analítica del negocio y la analítica predictiva

Conjunto de datos

Variable	Descripción	Tipo de dato	Posibles valores de los datos
#hotel	Nombre del hotel	Carácter	(H1 = Resort Hotel or +E5:E15H2 = City Hotel)
#is_canceled	Indica si la reservación es cancelada o no cancelada	Numérico	cancelada (1) no cancelada (0)
#lead_time	Número de días que transcurrieron entre la fecha de entrada de la reserva en el PMS y la fecha de llegada	Numérico	De 0 a 737 días
#arrival_date_year	Año (fecha) de llegada	Numérico	Del año 2015 al 2017
#arrval_date_month	Mes (fecha) de llegada	Carácter	Enero a Diciembre
#arrival_date_week_number	Número de la semana (fecha) de llegada en el año	Numérico	1 al 53
#arrival_date_day_of_month	El dia (fecha) de llegada	Numérico	1 al 31
#stays_in_weekend_nights	Número de noches que la persona se hospedó en dia de fin de semana (Sábado y Domingo)	Numérico	0 al 19
#stays_in_week_nights	Número de noches que la persona se hospedó en dia de semana (Lunes a Viernes)	Numérico	0 al 50
#adults	Número de adultos	Numérico	0 al 55
#children	Número de niños	Numérico	0 al 10

#babies	Numero de bebes	Numérico	0 al 10
#meal	Tipo de comida reservada. Se presenta por paquetes	Carácter	comida cama y desayuno = BB, media pensión = HB y otros
#country	País de origen. Representado en el formato ISO	Carácter	PRT, GBR, otros
#market_segment	Designación de segmento de mercado.	Carácter	Online TA, offline TA/TO, otros
#distribution_channel	Canal de distribución de reservas.	Carácter	TA/TO, direct, otros
#is_repeated_guest	Valor que indica si el nombre de la reserva fue de un huésped repetido (1) o no (0)	Numérico	0 al 1
#previous_cancellations	Número de reservas anteriores que fueron canceladas por el cliente antes de la reserva actual	Numérico	0 al 26
#previous_bookings_not_canceled	Número de reservas anteriores no canceladas por el cliente antes de la reserva actual	Numérico	0 al 72
#reserved_room_type	Código del tipo de habitación reservado.	Carácter	A, D, otros
#assigned_room_type	Código del tipo de habitación asignada a la reserva	Carácter	A, D, otros
#booking_changes	Número de cambios/modificaciones realizados a la reserva desde el momento en que se ingresó la reserva	Numérico	0 al 21
#deposit_type	indicación sobre si el cliente realizó un depósito para garantizar la reserva	Carácter	No hay depósito, no reembolso, otros
#agent	ID de la agencia de viajes de donde realizo la reserva	Carácter	ID o Null

#company	ID de la compañía o entidad donde realizo la reserva o el responsable pago la reserva.	Carácter	ID o Null
#days_in_waiting_list	Número de días que la reserva estuvo en lista de espera antes de que fuera confirmada al cliente	Numérico	0 al 391
#customer_type	Tipo de reserva de una de las cuatro categorías	Carácter	6.38 al 5.4K
#adr	Tarifa diaria promedio según se define dividiendo la suma de todas las transacciones de alojamientos	Numérico	0 al 8
#required_car_parking_spaces	Número de plazas de aparcamiento requeridas por el cliente	Numérico	0 al 8
#total_of_special_requests	Número de solicitudes especiales realizadas por el cliente	Numérico	0 al 5
#reservation_status	Último estado de la reserva	Carácter	check-out, canceled, otros
#reservation_status_date	Fecha en la que se estableció el último estado de la reserva.	Fecha	16 de octubre del 2014 al 13 septiembre del 2017

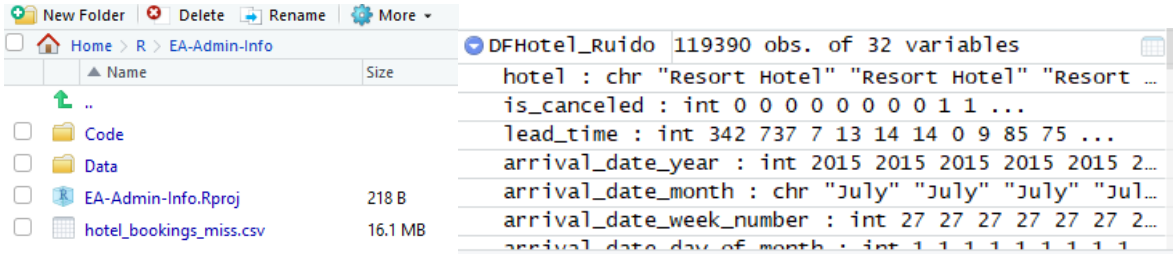
Análisis exploratorio de datos

Para poder analizar los datos, primero se deben de cargar, inspeccionar, pre procesar y luego visualizar

Cargar datos

Para cargar los datos, se descargó el archivo desde el link dado por el curso. Al descargar el archivo, este está en el formato xlsx, por lo que se utilizó la página web externa “Zamzar” para transformarlo a csv y facilitar la lectura de este archivo. Una vez transformado, se incluye el archivo en el proyecto y se usa el comando “read.csv” para leerlo. Finalmente, se usa el comando “View” para confirmar que se leyó con éxito los datos.

```
#Carga de Datos
DFHotel_Ruido <- read.csv("hotel_bookings_miss.csv", header = TRUE, stringsAsFactors = FALSE)
View(DFHotel_Ruido)
```



Una vez revisado, se usa la función “save” para guardar el archivo y ponerlo en la carpeta “Data”.

```
save(DFHotel_Ruido, file = "~/R/EA-Admin-Info/Data/DF_Ruido.RData")
```

Inspeccionar datos

Primero se carga el dataset que se obtuvo de la carga de datos. De ahí vemos el número de filas, columnas y el nombre de las columnas usando “nrow”, “ncol” y “colnames” respectivamente. Los resultados nos muestran que hay 119390 filas y 32 columnas, además del nombre de cada una de dichas columnas.

```
load("~/R/EA-Admin-Info/Data/DF_Ruido.RData")
nrow(DFHotel_Ruido)
ncol(DFHotel_Ruido)
colnames(DFHotel_Ruido)
```

```
> load("~/R/EA-Admin-Info/Data/DF_Ruido.RData")
> nrow(DFHotel_Ruido)
[1] 119390
> ncol(DFHotel_Ruido)
[1] 32
> colnames(DFHotel_Ruido)
 [1] "hotel"                      "is_canceled"                "lead_time"
 [4] "arrival_date_year"          "arrival_date_month"         "arrival_date_week_number"
 [7] "arrival_date_day_of_month"  "stays_in_weekend_nights"    "stays_in_week_nights"
[10] "adults"                     "children"                   "babies"
[13] "meal"                       "country"                    "market_segment"
[16] "distribution_channel"       "is_repeated_guest"          "previous_cancellations"
[19] "previous_bookings_not_canceled" "reserved_room_type"     "assigned_room_type"
[22] "booking_changes"            "deposit_type"               "agent"
[25] "company"                    "days_in_waiting_list"       "customer_type"
[28] "adr"                        "required_car_parking_spaces" "total_of_special_requests"
[31] "reservation_status"         "reservation_status_date"
```

Para conocer la estructura de cada columna y un pequeño ejemplo de que datos contiene, se usará la función “str”. La información más específica como media o clase de dato se ven con “summary”.

```
str(DFHotel_Ruido)
summary(DFHotel_Ruido)
```

```
> str(DFHotel_Ruido)
'data.frame':   119390 obs. of  32 variables:
 $ hotel                         : chr  "Resort Hotel" "Resort Hotel" "Resort Hotel" "Resort Hotel" ...
 $ is_canceled                   : int  0 0 0 0 0 0 0 0 1 1 ...
 $ lead_time                     : int  342 737 7 13 14 14 0 9 85 75 ...
 $ arrival_date_year             : int  2015 2015 2015 2015 2015 2015 2015 2015 2015 2015 ...
 $ arrival_date_month            : chr  "July" "July" "July" "July" ...
 $ arrival_date_week_number      : int  27 27 27 27 27 27 27 27 27 27 ...
 $ arrival_date_day_of_month     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ stays_in_weekend_nights       : int  NA 0 0 0 0 0 0 0 0 0 ...
 $ stays_in_week_nights          : int  0 0 1 1 2 2 2 2 3 3 ...
 $ adults                        : int  2 2 1 1 2 2 2 2 2 2 ...
 $ children                      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ babies                        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ meal                          : chr  "BB" "BB" "BB" "BB" ...
 $ country                       : chr  "PRT" "PRT" "GBR" "GBR" ...
 $ market_segment                : chr  "Direct" "Direct" "Direct" "Corporate" ...
 $ distribution_channel          : chr  "Direct" "Direct" "Direct" "Corporate" ...
 $ is_repeated_guest             : int  0 0 0 0 0 0 0 0 0 0 ...
 $ previous_cancellations        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ previous_bookings_not_canceled: int  0 0 0 0 0 0 0 0 0 0 ...
 $ reserved_room_type            : chr  "C" "C" "A" "A" ...
 $ assigned_room_type            : chr  "C" "C" "C" "A" ...
 $ booking_changes               : int  3 4 0 0 0 0 0 0 0 0 ...
 $ deposit_type                  : chr  "No Deposit" "No Deposit" "No Deposit" "No Deposit" ...
 $ agent                         : chr  "NULL" "NULL" "NULL" "304" ...
 $ company                       : chr  "NULL" "NULL" "NULL" "NULL" ...
 $ days_in_waiting_list          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ customer_type                 : chr  "Transient" "Transient" "Transient" "Transient" ...
 $ adr                           : num  0 0 75 75 98 ...
 $ required_car_parking_spaces   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ total_of_special_requests     : int  0 0 0 0 1 1 0 1 1 0 ...
 $ reservation_status            : chr  "Check-Out" "Check-Out" "Check-Out" "Check-Out" ...
 $ reservation_status_date       : chr  "7/1/2015" "7/1/2015" "7/2/2015" "7/2/2015" ...
> summary(DFHotel_Ruido)
    hotel            is_canceled       lead_time   arrival_date_year arrival_date_month
 Length:119390      Min.   :0.0000   Min.   :  0   Min.   :2015      Length:119390
 Class :character   1st Qu.:0.0000   1st Qu.: 18   1st Qu.:2016      Class :character
 Mode  :character   Median :0.0000   Median : 69   Median :2016      Mode  :character
                    Mean   :0.3704   Mean   :104   Mean   :2016
                    3rd Qu.:1.0000   3rd Qu.:160   3rd Qu.:2017
                    Max.   :1.0000   Max.   :737   Max.   :2017
                                     NA's   :21    NA's   :6
 arrival_date_week_number arrival_date_day_of_month stays_in_weekend_nights stays_in_week_nights
 Min.   : 1.00            Min.   : 1.0              Min.   : 0.0000         Min.   : 0.0
 1st Qu.:16.00            1st Qu.: 8.0              1st Qu.: 0.0000         1st Qu.: 1.0
 Median :28.00            Median :16.0              Median : 1.0000         Median : 2.0
 Mean   :27.16            Mean   :15.8              Mean   : 0.9275         Mean   : 2.5
 3rd Qu.:38.00            3rd Qu.:23.0              3rd Qu.: 2.0000         3rd Qu.: 3.0
 Max.   :53.00            Max.   :31.0              Max.   :19.0000         Max.   :50.0
 NA's   :25               NA's   :7                 NA's   :25              NA's   :12
     adults          children          babies             meal              country
 Min.   : 0.000   Min.   : 0.0000   Min.   : 0.00000   Length:119390      Length:119390
 1st Qu.: 2.000   1st Qu.: 0.0000   1st Qu.: 0.00000   Class :character   Class :character
 Median : 2.000   Median : 0.0000   Median : 0.00000   Mode  :character   Mode  :character
 Mean   : 1.856   Mean   : 0.1039   Mean   : 0.00795
 3rd Qu.: 2.000   3rd Qu.: 0.0000   3rd Qu.: 0.00000
 Max.   :55.000   Max.   :10.0000   Max.   :10.00000
 NA's   :12       NA's   :4         NA's   :32
 market_segment     distribution_channel is_repeated_guest previous_cancellations
 Length:119390      Length:119390        Min.   :0.00000   Min.   : 0.00000
 Class :character   Class :character     1st Qu.:0.00000   1st Qu.: 0.00000
 Mode  :character   Mode  :character     Median :0.00000   Median : 0.00000
                                         Mean   :0.03191   Mean   : 0.08712
                                         3rd Qu.:0.00000   3rd Qu.: 0.00000
                                         Max.   :1.00000   Max.   :26.00000
```

```
previous_bookings_not_canceled reserved_room_type assigned_room_type booking_changes
Min.    : 0.0000               Length:119390      Length:119390      Min.    : 0.0000
1st Qu.: 0.0000               Class :character   Class :character   1st Qu.: 0.0000
Median : 0.0000               Mode  :character   Mode  :character   Median : 0.0000
Mean   : 0.1371                                                     Mean   : 0.2211
3rd Qu.: 0.0000                                                     3rd Qu.: 0.0000
Max.   :72.0000                                                     Max.   :21.0000

deposit_type          agent             company          days_in_waiting_list customer_type
Length:119390      Length:119390      Length:119390      Min.   :  0.000      Length:119390
Class :character   Class :character   Class :character   1st Qu.:  0.000      Class :character
Mode  :character   Mode  :character   Mode  :character   Median :  0.000      Mode  :character
                                                         Mean   :  2.321
                                                         3rd Qu.:  0.000
                                                         Max.   :391.000
                                                         NA's   :7
     adr          required_car_parking_spaces total_of_special_requests reservation_status
Min.    :  -6.38   Min.    :0.00000           Min.    :0.0000           Length:119390
1st Qu.:  69.29   1st Qu.:0.00000           1st Qu.:0.0000           Class :character
Median :  94.58   Median :0.00000           Median :0.0000           Mode  :character
Mean   : 101.83   Mean   :0.06252           Mean   :0.5714
3rd Qu.: 126.00   3rd Qu.:0.00000           3rd Qu.:1.0000
Max.   :5400.00   Max.   :8.00000           Max.   :5.0000

reservation_status_date
Length:119390
Class :character
Mode  :character
```

Pre-procesamiento de datos

Para poder manipular los datos, se debe hacer un pre procesamiento para limpiar los datos.

Datos faltantes (NA)

Primero se deben identificar los valores nulos o NA y reemplazarlos. Para identificar los valores NA en el dataset, se usó la función is.na(). Luego, se creó otro dataframe llamado “ValoresVacios” para poder ver con la función summary() cuántas columnas tenían valores NA.

```
4  #Identificación de valores NA
5  ValoresVacios <- is.na(DFHotel_Ruido)
6  View(ValoresVacios)
7  summary(ValoresVacios)
```

Lo que se vé cuando se hace summary(ValoresVacios)

```
> summary(valoresvacios)
   hotel         is_canceled     lead_time       arrival_date_year arrival_date_month arrival_date_week_number
 Mode :logical   Mode :logical   Mode :logical   Mode :logical     Mode :logical       Mode :logical
 FALSE:119390    FALSE:119390    FALSE:119369    FALSE:119384      FALSE:119390        FALSE:119365
                                 TRUE :21        TRUE :6                               TRUE :25
 arrival_date_day_of_month stays_in_weekend_nights stays_in_week_nights   adults          children        babies
 Mode :logical             Mode :logical           Mode :logical        Mode :logical   Mode :logical   Mode :logical
 FALSE:119383              FALSE:119365            FALSE:119378         FALSE:119378    FALSE:119386    FALSE:119358
 TRUE :7                   TRUE :25                TRUE :12             TRUE :12        TRUE :4         TRUE :32
    meal          country          market_segment  distribution_channel is_repeated_guest previous_cancellations
 Mode :logical   Mode :logical   Mode :logical   Mode :logical         Mode :logical      Mode :logical
 FALSE:119390    FALSE:119390    FALSE:119390    FALSE:119390          FALSE:119390       FALSE:119390

 previous_bookings_not_canceled reserved_room_type assigned_room_type booking_changes deposit_type      agent
 Mode :logical                  Mode :logical      Mode :logical      Mode :logical    Mode :logical   Mode :logical
 FALSE:119390                   FALSE:119390       FALSE:119390       FALSE:119390     FALSE:119390    FALSE:119390

  company        days_in_waiting_list customer_type      adr          required_car_parking_spaces
 Mode :logical   Mode :logical        Mode :logical   Mode :logical   Mode :logical
 FALSE:119390    FALSE:119383         FALSE:119390    FALSE:119390    FALSE:119390
                 TRUE :7
 total_of_special_requests reservation_status reservation_status_date
 Mode :logical             Mode :logical      Mode :logical
 FALSE:119390              FALSE:119390       FALSE:119390
```

Con el summary() se pueden identificar las columnas que necesitan limpiar los datos vacíos. Estas columnas son lead_time, babies, arrival_date_year, arrival_date_week_number, arrival_date_day_of_month, stays_in_weekend_nights, stays_in_week_nights, adults, children, days_in_waiting_list. En total son 10 columnas que requieren reemplazar los valores NA.

Los valores que se encuentran en NA serán reemplazados con el promedio a excepción de las columnas children y babies que se les asignará el valor de 0. Para ello, se crea una función que le asigne los promedios de las columnas enviadas al dataframe. De esta manera, el proceso será más rápido y simple. Las columnas con los datos limpios se guardarán en un nuevo dataframe llamado DFHotel_Limpio.

Para hacer la función promedio, se hizo una función extra llamada mean.valor(x). Esta función recibe la variable que se desea reemplazar los valores NA con el promedio. La variable faltantes es un vector booleano que contiene los valores que son NA de la columna enviada, tot.faltantes es la suma de todos los datos que son NA del vector faltantes. El vector x.obs contiene todas los valores de la columna que no son NA, este valor se le asigna a la variable valorado. Finalmente, se reemplazan los valores que se encuentren en valorado[faltantes] con el promedio redondeado de x.obs y se retorna el valor de valorado.

```r
#Reemplazo por promedio:
mean.valor <- function(x){
  faltantes <- is.na(x)
  tot.faltantes <- sum(faltantes)
  x.obs <- x[!faltantes]
  valorado <- x
  valorado[faltantes] <- round(mean(x.obs))
  return (valorado)
}
```

Una vez que se tiene la función mean.valor, esta se utilizará para la función mean.df(). Esta función recibe el dataframe y las columnas que se va a reemplazar con el promedio. El vector

nombres contiene los nombres del dataframe enviado. En el for lo que se hace es un bucle en todas las columnas del dataframe, ahí la variable nombre contendrá el nuevo nombre de la columna del dataframe. Finalmente, se busca las columnas en el df[nombre] y se les asignará un nuevo valor llamando a la función mean.valor(df[,col]) donde se le enviará la columna.

```
mean.df <- function(df, cols){
  nombres <- names(df)
  for (col in cols) {
    nombre <- paste(nombres[col], sep = ".")
    df[nombre] <- mean.valor(df[,col])
  }
  df
}
```

Una vez que se tiene esta función, sólo se debe poner las columnas que se desean reemplazar con el promedio y el dataframe del cual sacar los valores. Finalmente, las columnas con los datos limpios serán asignados al nuevo dataframe DFHotel_Limpio que sólo contiene los valores limpios y ninguno que sea NA.

```
DFHotel_Limpio <- mean.df(DFHotel_Ruido, c(3,4,6,7,8,9,10,26))
```

Finalmente, se hizo otro dataframe Comprobacion, parecido al dataframe ValoresVacios, que contiene todos los valores que sean NA del dataframe DFHotel_Limpio con la función is.na(). Luego se le hace summary() a Comprobación para verificar que se cambiaron de forma efectiva con el promedio las columnas 3, 4, 7, 8, 9, 10 y 26.

```
Comprobacion <- is.na(DFHotel_Limpio)
summary(Comprobacion)
```

Como se ve en el summary(), todas las columnas seleccionadas ya no tienen ningún valor NA. Las únicas columnas con valores NA son la columna children y babies, pero estas se reemplazaran los valores nulos con un valor aleatorio.

```
> summary(Comprobacion)
   hotel         is_canceled     lead_time       arrival_date_year arrival_date_month arrival_date_week_number
 Mode :logical   Mode :logical   Mode :logical   Mode :logical     Mode :logical       Mode :logical
 FALSE:119390    FALSE:119390    FALSE:119390    FALSE:119390      FALSE:119390        FALSE:119390

 arrival_date_day_of_month stays_in_weekend_nights stays_in_week_nights   adults          children        babies
 Mode :logical             Mode :logical           Mode :logical        Mode :logical   Mode :logical   Mode :logical
 FALSE:119390              FALSE:119390            FALSE:119390         FALSE:119390    FALSE:119386    FALSE:119358
                                                                                       TRUE :4         TRUE :32
    meal          country         market_segment  distribution_channel is_repeated_guest previous_cancellations
 Mode :logical   Mode :logical   Mode :logical   Mode :logical        Mode :logical     Mode :logical
 FALSE:119390    FALSE:119390    FALSE:119390    FALSE:119390         FALSE:119390      FALSE:119390

 previous_bookings_not_canceled reserved_room_type assigned_room_type booking_changes deposit_type      agent
 Mode :logical                  Mode :logical      Mode :logical      Mode :logical   Mode :logical   Mode :logical
 FALSE:119390                   FALSE:119390       FALSE:119390       FALSE:119390    FALSE:119390    FALSE:119390

  company         days_in_waiting_list customer_type      adr           required_car_parking_spaces
 Mode :logical   Mode :logical         Mode :logical   Mode :logical   Mode :logical
 FALSE:119390    FALSE:119390          FALSE:119390    FALSE:119390    FALSE:119390

 total_of_special_requests reservation_status reservation_status_date
 Mode :logical             Mode :logical      Mode :logical
 FALSE:119390              FALSE:119390       FALSE:119390
```

La función para los valores aleatorios, sigue el mismo proceso que la de reemplazar con el promedio. Se crean dos funciones: una que saca el valor aleatorio de un valor rand.valor(x) y otra que saca el valor aleatorio de un dataframe random.df(df, cols). El procedimiento y la

funcionalidad de estas funciones es casi la misma que en la del promedio, la diferencia es que esta asigna un valor aleatorio. Para obtener el valor aleatorio se utiliza la función sample() que recibe la cantidad de valores a reemplazar (tot.faltantes) y de dónde sacará los valores aleatorios (x.obs).

```
#Reemplazo por valor aleatorio:
rand.valor <- function(x){
  faltantes <- is.na(x)
  tot.faltantes <- sum(faltantes)
  x.obs <- x[!faltantes]
  valorado <- x
  valorado[faltantes] <- sample(x.obs, tot.faltantes, replace = TRUE)
  return (valorado)
}

random.df <- function(df, cols){
  nombres <- names(df)
  for (col in cols) {
    nombre <- paste(nombres[col], sep = ".")
    df[nombre] <- rand.valor(df[,col])
  }
  df
}
```

Se utiliza esta función para las columnas de children (11) y babies (12).

```
DFHotel_Limpio <- random.df(DFHotel_Limpio, c(11,12))
```

Se hace el dataframe Comprobacion_Aleatorio, igual que el dataframe Comprobacion y ValoresVacios, para comprobar que ya no hay más valores NA. Con la función is.na() se le asignan los valores en un vector booleano del dataframe DFHotel_Limpio. Con la función summary() se verifica que ya no haya valores NA en ninguna fila ni columna del DF.

```
Comprobacion_Aleatorio <- is.na(DFHotel_Limpio)
summary(Comprobacion_Aleatorio)
```

Como se puede ver, ya no hay ningún valor NA y fueron reemplazados exitosamente.

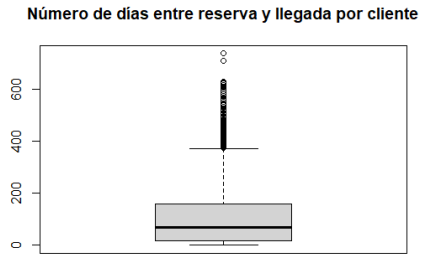
```
> summary(Comprobacion_Aleatorio)
   hotel         is_canceled     lead_time       arrival_date_year arrival_date_month arrival_date_week_number
 Mode :logical   Mode :logical   Mode :logical   Mode :logical     Mode :logical       Mode :logical
 FALSE:119390    FALSE:119390    FALSE:119390    FALSE:119390      FALSE:119390        FALSE:119390
 arrival_date_day_of_month stays_in_weekend_nights stays_in_week_nights   adults          children         babies
 Mode :logical             Mode :logical           Mode :logical        Mode :logical   Mode :logical   Mode :logical
 FALSE:119390              FALSE:119390            FALSE:119390         FALSE:119390    FALSE:119390    FALSE:119390
    meal          country         market_segment  distribution_channel is_repeated_guest previous_cancellations
 Mode :logical   Mode :logical   Mode :logical   Mode :logical        Mode :logical     Mode :logical
 FALSE:119390    FALSE:119390    FALSE:119390    FALSE:119390         FALSE:119390      FALSE:119390
 previous_bookings_not_canceled reserved_room_type assigned_room_type booking_changes deposit_type      agent
 Mode :logical                  Mode :logical      Mode :logical      Mode :logical   Mode :logical   Mode :logical
 FALSE:119390                   FALSE:119390       FALSE:119390       FALSE:119390    FALSE:119390    FALSE:119390
  company        days_in_waiting_list customer_type      adr          required_car_parking_spaces
 Mode :logical   Mode :logical        Mode :logical   Mode :logical   Mode :logical
 FALSE:119390    FALSE:119390         FALSE:119390    FALSE:119390    FALSE:119390
 total_of_special_requests reservation_status reservation_status_date
 Mode :logical             Mode :logical      Mode :logical
 FALSE:119390              FALSE:119390       FALSE:119390
```

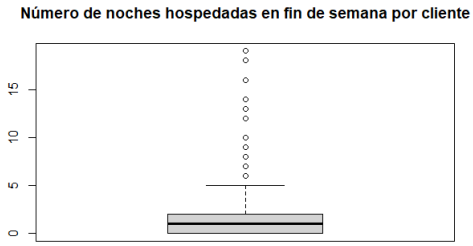
Datos atípicos (outliers)

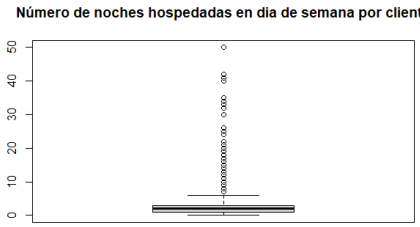
En el caso de los valores outliers, primero se debe detectar su presencia para saber qué columnas modificar. En este caso, se detectó valores outliers en cinco columnas distintas mediante el uso del gráfico boxplot. Este gráfico es muy efectivo para hallar valores que no

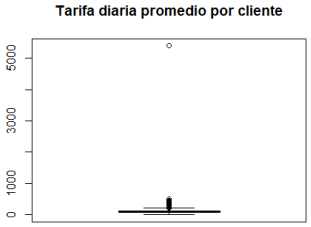
estén dentro del rango típico., ya que los muestra en sus resultados. Si se le añade “$out” al boxplot, se puede ver los valores específicos que quedan clasificados como outliers.

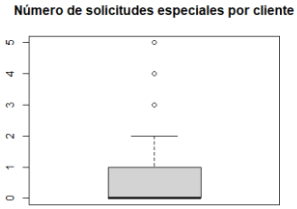
```
#Identificación de valores atípicos
boxplot(DFHotel_Limpio$lead_time, main = "Número de días entre reserva y llegada por cliente")
boxplot(DFHotel_Limpio$lead_time)$out
boxplot(DFHotel_Limpio$stays_in_weekend_nights, main = "Número de noches hospedadas en fin de semana por cliente")
boxplot(DFHotel_Limpio$stays_in_weekend_nights)$out
boxplot(DFHotel_Limpio$stays_in_week_nights, main = "Número de noches hospedadas en dia de semana por cliente")
boxplot(DFHotel_Limpio$stays_in_week_nights)$out
boxplot(DFHotel_Limpio$adr, main = "Tarifa diaria promedio por cliente")
boxplot(DFHotel_Limpio$adr)$out
boxplot(DFHotel_Limpio$total_of_special_requests, main = "Número de solicitudes especiales por cliente")
boxplot(DFHotel_Limpio$total_of_special_requests)$out
```











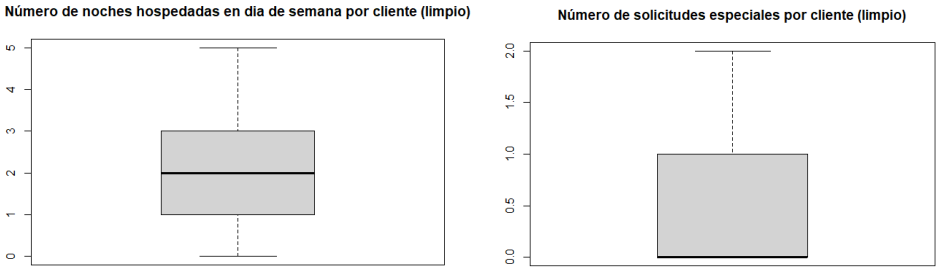
Ahora que ya se saben qué datos se deben de cambiar, solo falta crear las funciones que se encargará de la limpieza de datos atípicos. En este caso se usará la de cambio por mediana y promedio, y la de cambio por enmascarado o capping.

```
#Cambio por promedio y mediana
outliers.med <- function(x, removeNA = TRUE){
  quantiles <- quantile(x, c(0.05, 0.95), na.rm = removeNA)
  x[x<quantiles[1]] <- mean(x, na.rm = removeNA)
  x[x>quantiles[2]] <- median(x, na.rm = removeNA)
  x
}
#Cambio por enmascarado (capping)
outliers.cap <- function(x, removeNA = TRUE){
  qrts <- quantile(x, probs = c(0.25, 0.75), na.rm = removeNA)
  caps <- quantile(x, probs = c(.05, 0.95), na.rm = removeNA)
  iqr <- qrts[2]-qrts[1]
  altura <- 1.5*iqr
  x[x<qrts[1]-altura] <- caps[1]
  x[x>qrts[2]+altura] <- caps[2]
  x
}
```

En la primera función lo que se hace es reemplazar las variables dependiendo de su valor. Si está debajo del quinto percentil, se reemplaza con la media, mientras que si es mayor al percentil 95, se cambia por la mediana. En el caso de de segunda función, se sustituyen los valores por el percentil 0.05 o 0.95 dependiendo de si el valor supera el bigote superior o se encuentra debajo del bigote inferior.

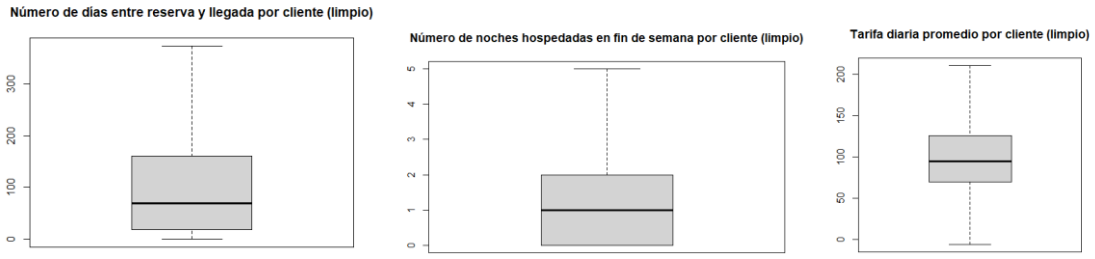
Estos fueron los boxplots resultantes de utilizar la primera función:

```r
boxplot(DFHotel_Limpio$stays_in_week_nights, main = "Número de noches hospedadas en dia de semana por cliente (outliers)")
boxplot(DFHotel_Limpio$stays_in_week_nights)$out
boxplot(outliers.med(DFHotel_Limpio$stays_in_week_nights), main = "Número de noches hospedadas en dia de semana por cliente (limpio)")
boxplot(outliers.med(DFHotel_Limpio$stays_in_week_nights))$out
summary(outliers.med(DFHotel_Limpio$stays_in_week_nights))
DFHotel_Limpio$stays_in_week_nights <- outliers.med(DFHotel_Limpio$stays_in_week_nights)
summary(DFHotel_Limpio$stays_in_week_nights)

boxplot(DFHotel_Limpio$total_of_special_requests, main = "Número de solicitudes especiales por cliente (outliers)")
boxplot(DFHotel_Limpio$total_of_special_requests)$out
boxplot(outliers.med(DFHotel_Limpio$total_of_special_requests), main = "Número de solicitudes especiales por cliente (limpio)")
boxplot(outliers.med(DFHotel_Limpio$total_of_special_requests))$out
summary(outliers.med(DFHotel_Limpio$total_of_special_requests))
DFHotel_Limpio$total_of_special_requests <- outliers.med(DFHotel_Limpio$total_of_special_requests)
summary(DFHotel_Limpio$total_of_special_requests)
```



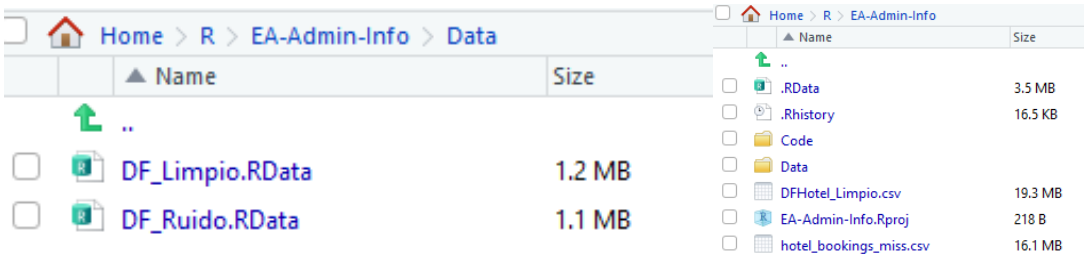
Estos fueron los boxplots luego de aplicar la segunda función:

```r
boxplot(DFHotel_Limpio$lead_time, main = "Número de días entre reserva y llegada por cliente (outliers)")
boxplot(DFHotel_Limpio$lead_time)$out
boxplot(outliers.cap(DFHotel_Limpio$lead_time), main = "Número de días entre reserva y llegada por cliente (limpio)")
boxplot(outliers.cap(DFHotel_Limpio$lead_time))$out
summary(outliers.cap(DFHotel_Limpio$lead_time))
DFHotel_Limpio$lead_time <- outliers.cap(DFHotel_Limpio$lead_time)
summary(DFHotel_Limpio$lead_time)

boxplot(DFHotel_Limpio$stays_in_weekend_nights, main = "Número de noches hospedadas en fin de semana por cliente (outliers)")
boxplot(DFHotel_Limpio$stays_in_weekend_nights)$out
boxplot(outliers.cap(DFHotel_Limpio$stays_in_weekend_nights), main = "Número de noches hospedadas en fin de semana por cliente (limpio)")
boxplot(outliers.cap(DFHotel_Limpio$stays_in_weekend_nights))$out
summary(outliers.cap(DFHotel_Limpio$stays_in_weekend_nights))
DFHotel_Limpio$stays_in_weekend_nights <- outliers.cap(DFHotel_Limpio$stays_in_weekend_nights)
summary(DFHotel_Limpio$stays_in_weekend_nights)

boxplot(DFHotel_Limpio$adr, main = "Tarifa diaria promedio por cliente (outliers)")
boxplot(DFHotel_Limpio$adr)$out
boxplot(outliers.cap(DFHotel_Limpio$adr), main = "Tarifa diaria promedio por cliente (limpio)")
boxplot(outliers.cap(DFHotel_Limpio$adr))$out
summary(outliers.cap(DFHotel_Limpio$adr))
DFHotel_Limpio$adr <- outliers.cap(DFHotel_Limpio$adr)
summary(DFHotel_Limpio$adr)
```



En el código anterior también se muestra como los nuevos valores reemplazan los antiguos y el uso de la función summary para confirmar que se reemplazan con éxito los datos. Por último, con los datos preprocesados, se guarda el data frame obtenido tanto en formato csv como en formato RData.

```r
#Guardado del dataframe preprocesado (Rdata y csv)
save(DFHotel_Limpio, file = "~/R/EA-Admin-Info/Data/DF_Limpio.RData")
write.csv(DFHotel_Limpio, "DFHotel_Limpio.csv", row.names = FALSE)
```

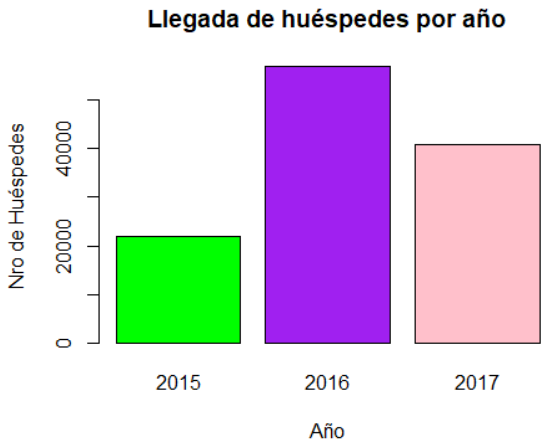


Visualizar datos

Relación demanda con tiempo

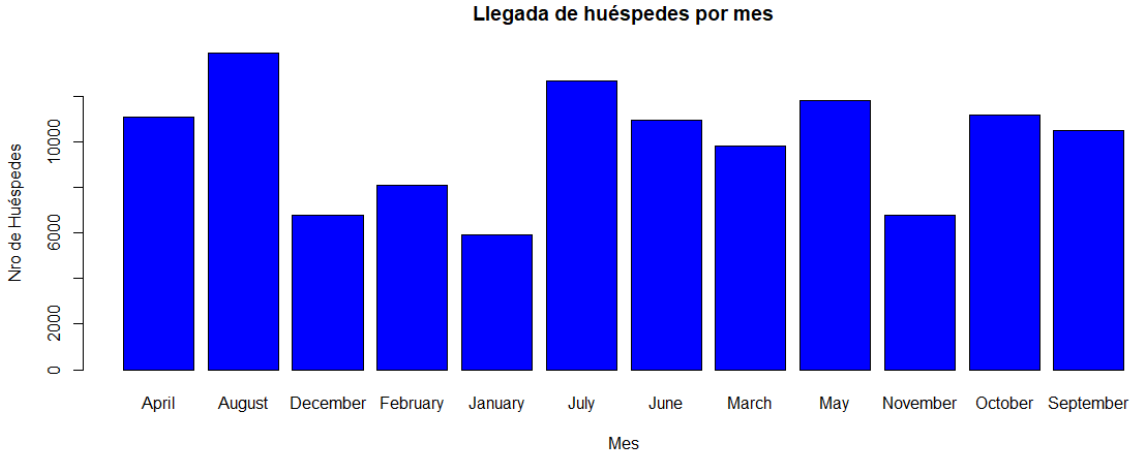
Si solo se considera la demanda por año, con el siguiente gráfico se puede ver que en general sí hubo un aumento en la demanda de 2015 a 2017, a pesar de que esta decreció entre 2016 y 2017. Este gráfico se obtuvo con la función barplot.

```
barplot(table(DFHotel_Limpio$arrival_date_year), main = "Llegada de huéspedes por año",
        xlab = "Año", ylab = "Nro de Huéspedes", col = c("green", "purple", "pink"))
```



Por otro lado, si se quiere ver la demanda en un formato mensual, se utiliza este gráfico que se obtuvo con la misma función.

```
barplot(table(DFHotel_Limpio$arrival_date_month), main = "Llegada de huéspedes por mes",
        xlab = "Mes", ylab = "Nro de Huéspedes", col = "blue")
```




```
table(DFHotel_Limpio$arrival_date_month)

  April    August  December  February   January      July      June     March       May  November
  11089     13877      6780      8068      5929     12661     10939      9794     11791      6794
October September
  11160     10508
```

Usando tanto el gráfico como la función table, se puede notar un aumento de demanda desde enero hasta mayo, una ligera caída entre mayo y junio, un aumento entre junio y agosto, otra caída menor en septiembre seguida de un alza en octubre, concluyendo con un gran decrecimiento en noviembre y diciembre. Debido a que hay más aumentos que caídas, se podría decir que la demanda también aumenta mensualmente.

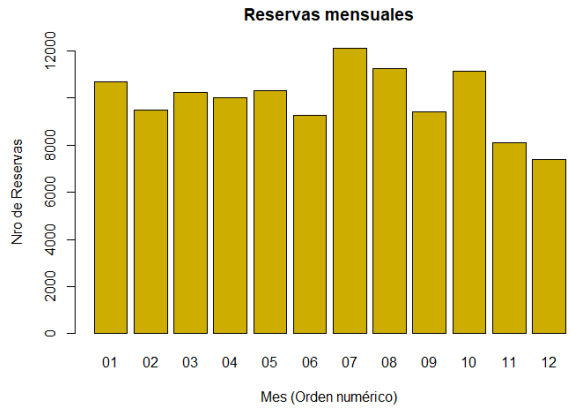
Temporada de reservas (alta, media baja)

Para hallar las temporadas de reservas, se va a dividir las reservas por mes. Esto se puede conseguir del valor “reservation_status_date”, pero primero se tendrá que pasar este dato a tipo “Date”.

```
table(DFHotel_Limpio$reservation_status_date)
fecha.reserva <- as.Date(DFHotel_Limpio$reservation_status_date, "%m/%d/%Y")
View(fecha.reserva)
fecha.reserva <- format(fecha.reserva,"%m")
```

Esto nos da una columna que contendrá específicamente los meses de cada reserva, que es justo lo necesario para generar el barplot que se necesita para conseguir la respuesta a las temporadas de reservas.

```
barplot(table(fecha.reserva), main = "Reservas mensuales",
        xlab = "Mes (Orden numérico)", ylab = "Nro de Reservas", col = "gold3")
```




```
> table(fecha.reserva)
fecha.reserva
   01    02    03    04    05    06    07    08    09    10    11    12
10681  9498 10230  9999 10304  9278 12106 11249  9403 11143  8099  7400
```

Del gráfico y la tabla resultantes se puede clasificar cada mes por su temporada de reserva. La clasificación va más al criterio o parámetros dados para cada tipo de clasificación, pero para este caso se diría que tanto Diciembre como Noviembre son meses con una baja temporada de reservas; Enero, Julio, Agosto, Octubre son meses con alta temporada de reservas, y los meses restantes quedarían en la sección media.

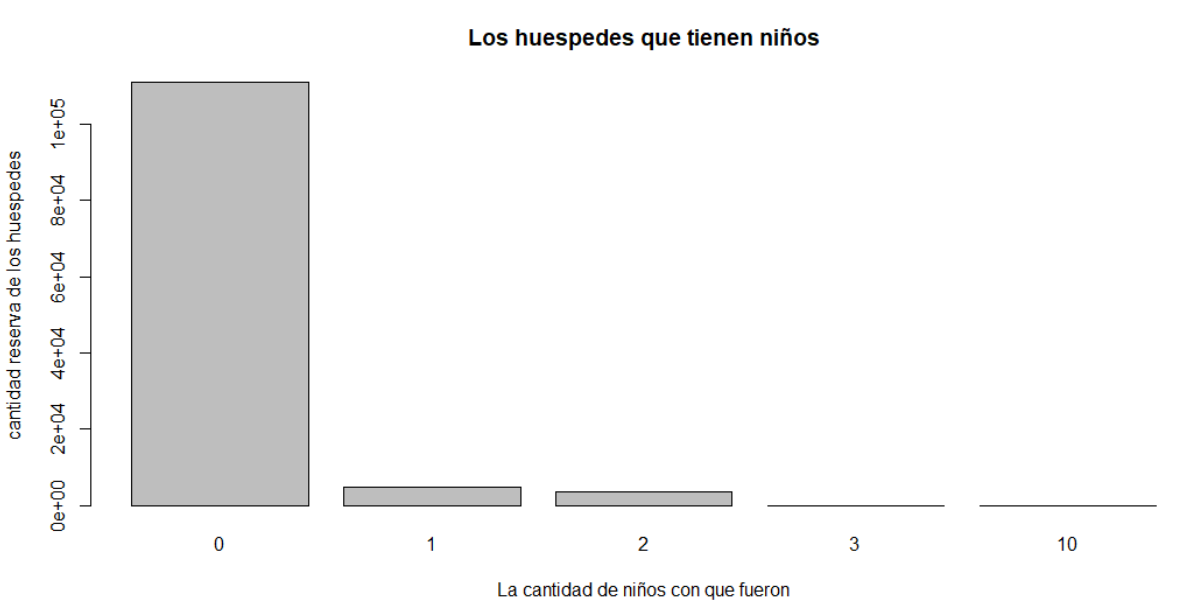
Reservas que incluyan niños

Para poder hallar la cantidad de niños y/o bebés que llevaron los huéspedes primero lo vemos por separado, viendo a los bebés que tienen los huéspedes por separado podremos saber cuántos de ellos incluyen bebés en su reserva.

```
barplot(table(DFHotel_Limpio$babies),
        main = "Los huespedes que tienen bebes",
        xlab = "La cantidad de bebes con que fueron",ylab ="cantidad reserva de los huespedes")
```

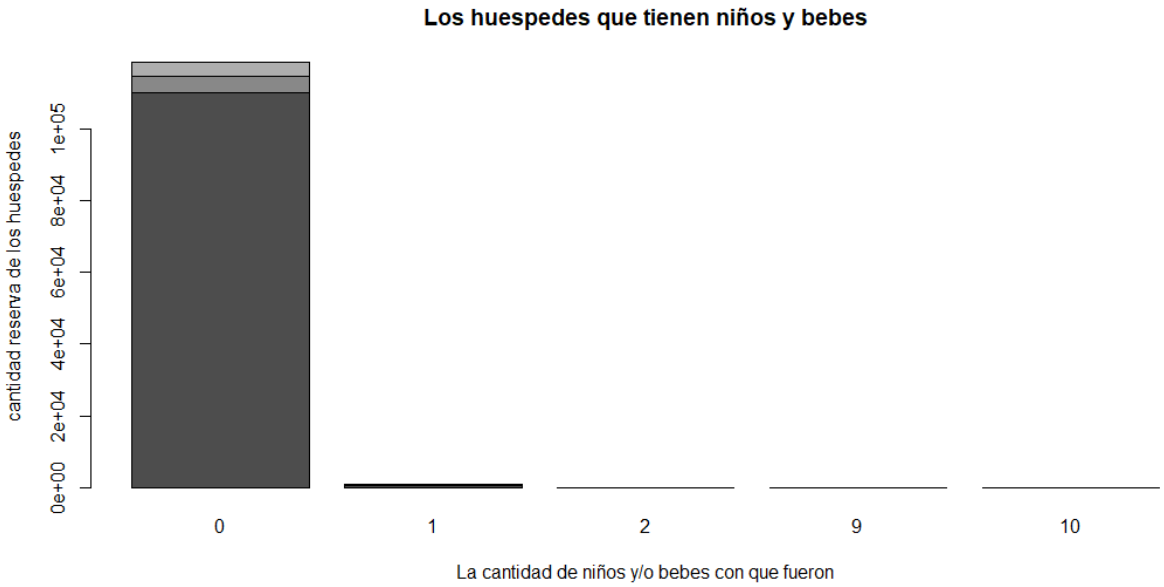
Vimos que los huéspedes no suelen ir con bebés a sus reservas de hoteles.
Ahora vamos a ver a los huéspedes que incluyen niños en su reserva.

```
barplot(table(DFHotel_Limpio$children, DFHotel_Limpio$babies),
        main = "Los huespedes que tienen niños",
        xlab = "La cantidad de niños con que fueron",ylab ="cantidad reserva de los huespedes")
```



También los huéspedes no suelen ir mucho a sus reservaciones con niños.
Ahora vemos a los huéspedes tener niños y bebés en su reservación.

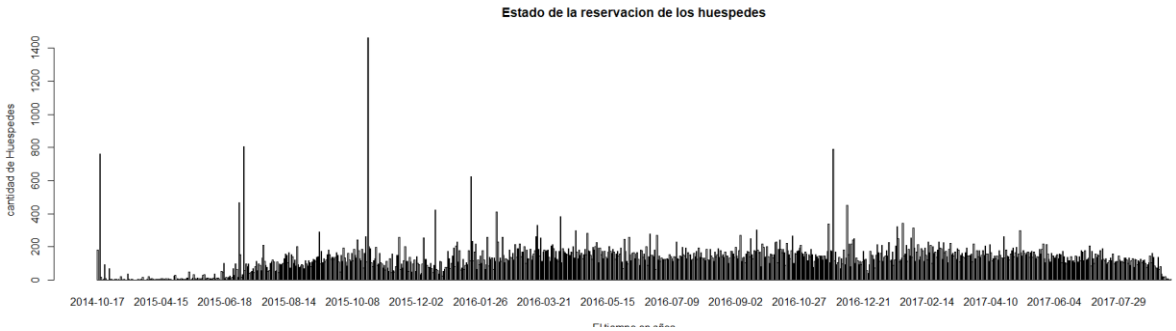
```
barplot(table(DFHotel_Limpio$children, DFHotel_Limpio$babies),
    main = "Los huespedes que tienen niños y bebes",
    xlab = "La cantidad de niños y/o bebes con que fueron",
    ylab ="cantidad reserva de los huespedes")
```



Época menor demanda de reservas

Considerando los estados de reserva que hicieron los huéspedes entre los años del 2014 al 2017 se puede hacer el siguiente gráfico en que se muestra

```
barplot(table(DFHotel_Limpio$reservation_status_date),
        main = "Estado de la reservacion de los huespedes",
        xlab = "El tiempo en años",ylab ="cantidad de Huespedes")
```



Del gráfico se puede ver que en el 17 de octubre del 2014 al 18 de Junio del 2015 tiene menor demanda en la reserva de los huéspedes

Espacios de estacionamiento

Esta visualización se hace con el propósito de saber si es importante contar estacionamiento en el hotel. Para ello se harán gráficos para a partir de ellos concluir la respuesta. Primero se hará una tabla llamada A que contiene el la cantidad de estacionamientos requeridos por tipo de hotel. Luego se harán tres gráficos de pie. El primero tendrá los estacionamientos requeridos considerando ambos hoteles. El segundo, tendrá los estacionamientos requeridos para el City Hotel. El tercero, tendrá los estacionamientos requeridos para el Resort Hotel.

Para hacer la tabla se usó la función table(). Los parámetros que recibe son las columnas required_car_parking_spaces y hotel del dataframe limpio (DFHotel_Limpio).

```
A <- table(DFHotel_Limpio$hotel,DFHotel_Limpio$required_car_parking_spaces)
```

```
A <- table(DFHotel_Limpio$hotel,DFHotel_Limpio$required_car_parking_spaces)
A
```

```
                0     1     2     3     8
City Hotel  77404  1921     3     2     0
Resort Hotel 34570  5462    25     1     2
```

Como se ve en la tabla, el total de espacios de estacionamiento requeridos en el City Hotel son 1 927 estacionamientos y no se requieren 77 404 espacios de estacionamiento. En el Resort Hotel se requieren 5 515 estacionamientos y no se requieren 34 570 estacionamientos.

Este gráfico se hizo con la función pie(). Los parámetros que recibió son la tabla A, el título del gráfico, el tamaño de las etiquetas (cex), el color para el gráfico (col) y sus bordes (border). La leyenda del gráfico se hizo con la función legend(). Los parámetros que recibe son la posición en la que se ubicará, las etiquetas y los colores a utilizar.

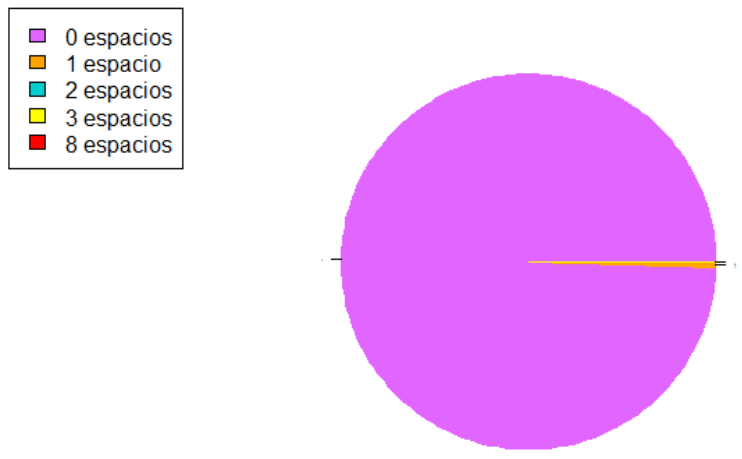
```
etiquetas <- c("0 espacios", "1 espacio", "2 espacios", "3 espacios", "8 espacios")

pie(table(DFHotel_Limpio$hotel,DFHotel_Limpio$required_car_parking_spaces),
    main = "Estacionamientos requeridos en ambos hoteles",
    cex = 0.000000000001,
    col = c("deeppink", "orange", "darkturquoise", "yellow", "red"),
    border = c("deeppink", "orange", "darkturquoise", "yellow", "red"))

legend("topleft", legend = etiquetas,
       fill =  c("deeppink", "orange", "darkturquoise", "yellow", "red"))
```

Para hacer el gráfico de pie que contiene los espacios requeridos para el City Hotel tiene casi el mismo procedimiento que para hacer el gráfico donde se consideran ambos hoteles. Se utilizará la función pie, la diferencia es que en vez de usar la tabla que contiene los estacionamientos de ambos hoteles, se usa el vector City que contiene los estacionamientos requeridos por el City Hotel.

```
City <- c(A[1,1], A[1,2], A[1,3], A[1,4], A[1,5]) #Contiene sólo los estacionamientos del City Hotel

pie(City,
    main = "Estacionamientos requeridos en el City Hotel",
    cex = 0.000000000001,
    col = c("aquamarine", "deepskyblue", "blue", "forestgreen", "khaki4"),
    border = c("aquamarine", "deepskyblue", "blue", "forestgreen", "khaki4"))

legend("topleft", legend = etiquetas,
       fill =  c("aquamarine", "deepskyblue", "blue", "forestgreen", "khaki4"))
```

Para hacer el gráfico de pie que contiene los espacios requeridos para el Resort Hotel tiene casi el mismo procedimiento que para hacer el gráfico donde se consideran ambos hoteles. Se utilizará la función pie, la diferencia es que en vez de usar la tabla que contiene los estacionamientos de ambos hoteles, se usa el vector Resort que contiene los estacionamientos requeridos por el Resort Hotel.

```
Resort <- c(A[2,2], A[2,3], A[2,4], A[2,5]) #Contiene sólo los estacionamientos del City Hotel

pie(Resort,
    main = "Estacionamientos requeridos en el Resort Hotel",
    cex = 0.000000000001,
    col = c("mediumorchid1", "orange", "darkturquoise", "yellow", "red"),
    border = c("mediumorchid1", "orange", "darkturquoise", "yellow", "red"))

legend("topleft", legend = etiquetas,
       fill =  c("mediumorchid1", "orange", "darkturquoise", "yellow", "red"))
```



A simple vista de las visualizaciones se puede inferir que no es necesario contar con espacios de estacionamiento para el Resort Hotel. En el caso del City Hotel, tampoco es necesario aunque hay algunos huéspedes que requieren 1 espacio de estacionamiento.

Meses en que se producen mayores cancelaciones

Para hallar en qué meses se producen la mayor cantidad de cancelaciones, se necesita separar primero solo los valores que son importantes, en este caso, las reservas canceladas. Esto se hará creando un data frame suplementario “solo.cancelado”, el cual cambia todos los valores en “reservation_status” que no sean igual a “Cancelled” a NA para proceder a remover las filas que no sean relevantes.

```
solo.cancelado <- DFHotel_Limpio
solo.cancelado$reservation_status[solo.cancelado$reservation_status != "Canceled"] <- NA
solo.cancelado <- na.omit(solo.cancelado)
```

Usando table se comprueba que solo hay reservas canceladas en este data frame.

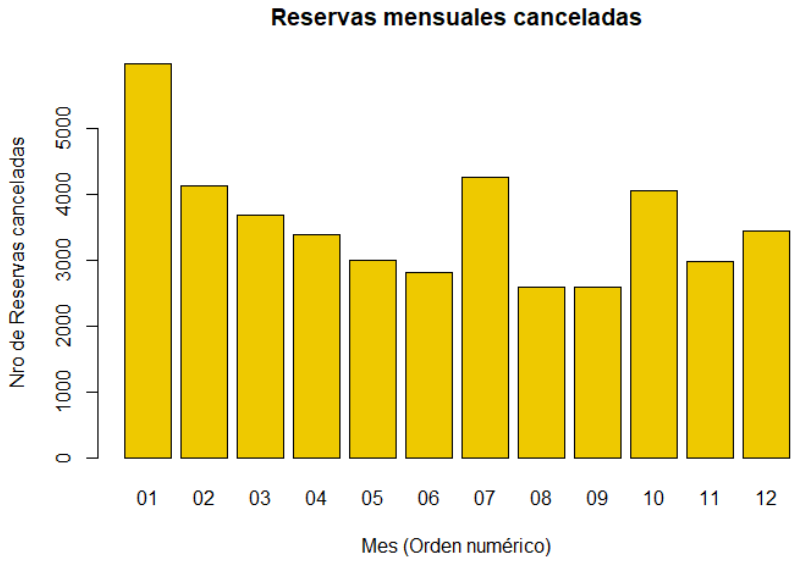
```
> table(solo.cancelado$reservation_status == "Canceled")

 TRUE
43017
```

De ahí se extrae los meses de cada reserva cancelada mediante la transformación a formato “Date”.

```
fecha.reserva.cancelado <- as.Date(solo.cancelado$reservation_status_date, "%m/%d/%Y")
View(fecha.reserva.cancelado)
fecha.reserva.cancelado <- format(fecha.reserva.cancelado,"%m")
```

Por último, se crea un gráfico barplot para obtener los resultados.

```
barplot(table(fecha.reserva.cancelado), main = "Reservas mensuales canceladas",
        xlab = "Mes (Orden numérico)", ylab = "Nro de Reservas canceladas", col = "gold2")
```




```
> table(fecha.reserva.cancelado)
fecha.reserva.cancelado
  01   02   03   04   05   06   07   08   09   10   11   12
5986 4129 3697 3393 3006 2828 4274 2596 2609 4072 2982 3445
```

Con los resultados obtenidos se puede decir con seguridad que los meses con mayor cantidad de reservas canceladas son Enero, Julio, Febrero y Octubre.

Cantidad de reservas por tipo de hotel

Con esta visualización se espera concluir qué hotel prefiere la gente. Para ello se harán tres gráficos. Primero, antes de hacer los gráficos se harán cinco tablas para saber las cantidades específicas de cada resultado. La primera tabla se llamará TiempoEspera que contiene la

cantidad de días que se estuvo en la lista de espera por tipo de hotel. La segunda tabla se llamará f_porc que contiene las frecuencias porcentuales de los días en la lista de espera de ambos hoteles. La tercera tabla se llamará F_Acum_City que contiene sólo las frecuencias de los días en la lista de espera del City Hotel. La cuarta tabla se llamará F_Acum_Resort que contiene sólo las frecuencias de los días en la lista de espera del Resort Hotel. La última tabla se llamará Cancelado que contiene la cantidad de reservas canceladas por tipo de hotel. El primer gráfico será un gráfico de barras que contiene las reservas canceladas de ambos hoteles. El segundo gráfico será uno de barras que muestre las frecuencias porcentuales de los días en la lista de espera del Resort Hotel. El tercer gráfico será uno de barras que muestre las frecuencias porcentuales de los días en la lista de espera del CityHotel.

Para hacer la tabla TiempoEspera se usó la función table(). Los parámetros que recibe son las columnas de hotel y days_in_waiting_list del dataframe limpio (DFHotel_Limpio). Como resultado se obtiene una tabla con 128 filas incluyendo la primera fila que contiene el tipo de hotel, y 3 columnas. El máximo valor en las filas es de 391 días.

```
TiempoEspera <- table(DFHotel_Limpio$days_in_waiting_list,DFHotel_Limpio$hotel)
TiempoEspera
```

	City Hotel	Resort Hotel
0	75887	39798
1	7	5
2	3	9
3	59	0
4	22	3
5	5	3
6	12	4
7	4	0
8	4	3
9	16	0
10	28	0
11	6	1
12	3	0
13	2	2
14	3	6
15	54	0
16	4	0
17	47	0

Para hacer la tabla f_porc se usó la función round(prop.table()). La función prop.table() se usa para saber la frecuencia relativa y esta recibe como parámetro la tabla TiempoEspera. La función round() recibe como parámetros la función prop.table() que es multiplicada por 100 para así tener la frecuencia porcentual y se le dice que redondee los valores de la tabla a 2 decimales. Esta tabla tiene 128 filas.

```
> f_porc <- round((prop.table(TiempoEspera)*100),2) #Frecuencia porcentual días lista de espera para ambos hoteles
> nrow(f_porc)
[1] 128
> f_porc

      City Hotel Resort Hotel
  0         63.56        33.33
  1          0.01         0.00
  2          0.00         0.01
  3          0.05         0.00
  4          0.02         0.00
  5          0.00         0.00
  6          0.01         0.00
  7          0.00         0.00
  8          0.00         0.00
  9          0.01         0.00
  10         0.02         0.00
  11         0.01         0.00
  12         0.00         0.00
  13         0.00         0.00
  14         0.00         0.01
  15         0.05         0.00
```

Para hacer la tabla F_Acum_City se utiliza la primera columna de la tabla f_porc, para así solo tener las frecuencias porcentuales del City Hotel.

```
> F_Acum_City <-f_porc[1:128, 1] #Frecuencia porcentual días lista de espera City Hotel
> F_Acum_City
    0     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19
63.56  0.01  0.00  0.05  0.02  0.00  0.01  0.00  0.00  0.01  0.02  0.01  0.00  0.00  0.00  0.05  0.00  0.04  0.00  0.03
   20    21    22    23    24    25    26    27    28    30    31    32    33    34    35    36    37    38    39    40
 0.02  0.03  0.05  0.00  0.02  0.02  0.00  0.02  0.04  0.00  0.11  0.02  0.03  0.02  0.08  0.00  0.00  0.06  0.19  0.02
   41    42    43    44    45    46    47    48    49    50    52    53    54    55    56    57    58    59    60    61
 0.05  0.03  0.01  0.12  0.05  0.08  0.00  0.04  0.02  0.06  0.00  0.01  0.00  0.01  0.04  0.02  0.14  0.01  0.03  0.01
   62    63    64    65    68    69    70    71    72    73    74    75    76    77    79    80    81    83    84    85
 0.05  0.07  0.00  0.01  0.04  0.07  0.02  0.01  0.00  0.00  0.00  0.02  0.00  0.05  0.02  0.02  0.00  0.00  0.00  0.01
   87    89    91    92    93    96    97    98    99   100   101   105   107   108   109   111   113   116   117   120
 0.07  0.00  0.04  0.00  0.03  0.04  0.00  0.05  0.01  0.00  0.04  0.00  0.00  0.02  0.00  0.06  0.00  0.00  0.00  0.02
  121   122   125   142   147   150   154   160   162   165   167   174   175   176   178   183   185   187   193   207
 0.00  0.00  0.00  0.00  0.03  0.00  0.00  0.02  0.03  0.00  0.00  0.02  0.00  0.04  0.03  0.00  0.00  0.04  0.00  0.01
  215   223   224   236   259   330   379   391
 0.02  0.05  0.01  0.03  0.01  0.01  0.01  0.04
```

Para hacer la tabla F_Acum_Resort se utiliza la segunda columna de la tabla f_porc, para así solo tener las frecuencias porcentuales del Resort Hotel.

```
> F_Acum_Resort <- f_porc[1:128, 2] #Frecuencia porcentual días lista de espera Resort Hotel
> F_Acum_Resort
    0     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19
33.33  0.00  0.01  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.01  0.00  0.00  0.00  0.00  0.00
   20    21    22    23    24    25    26    27    28    30    31    32    33    34    35    36    37    38    39    40
 0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
   41    42    43    44    45    46    47    48    49    50    52    53    54    55    56    57    58    59    60    61
 0.00  0.00  0.00  0.00  0.00  0.00  0.02  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
   62    63    64    65    68    69    70    71    72    73    74    75    76    77    79    80    81    83    84    85
 0.00  0.00  0.00  0.02  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.02  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
   87    89    91    92    93    96    97    98    99   100   101   105   107   108   109   111   113   116   117   120
 0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.02  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
  121   122   125   142   147   150   154   160   162   165   167   174   175   176   178   183   185   187   193   207
 0.00  0.05  0.01  0.00  0.00  0.00  0.01  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
  215   223   224   236   259   330   379   391
 0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
```

Para hacer la tabla Cancelado se usó la función table(). Los parámetros que recibe son las columnas de hotel y is_canceled del dataframe limpio (DFHotel_Limpio). La columna con nombre “0” significa las reservas que no fueron canceladas. La columna con nombre “1” significa las reservas que fueron canceladas.

```
Cancelado <- table(DFHotel_Limpio$hotel, DFHotel_Limpio$is_canceled)
Cancelado
```

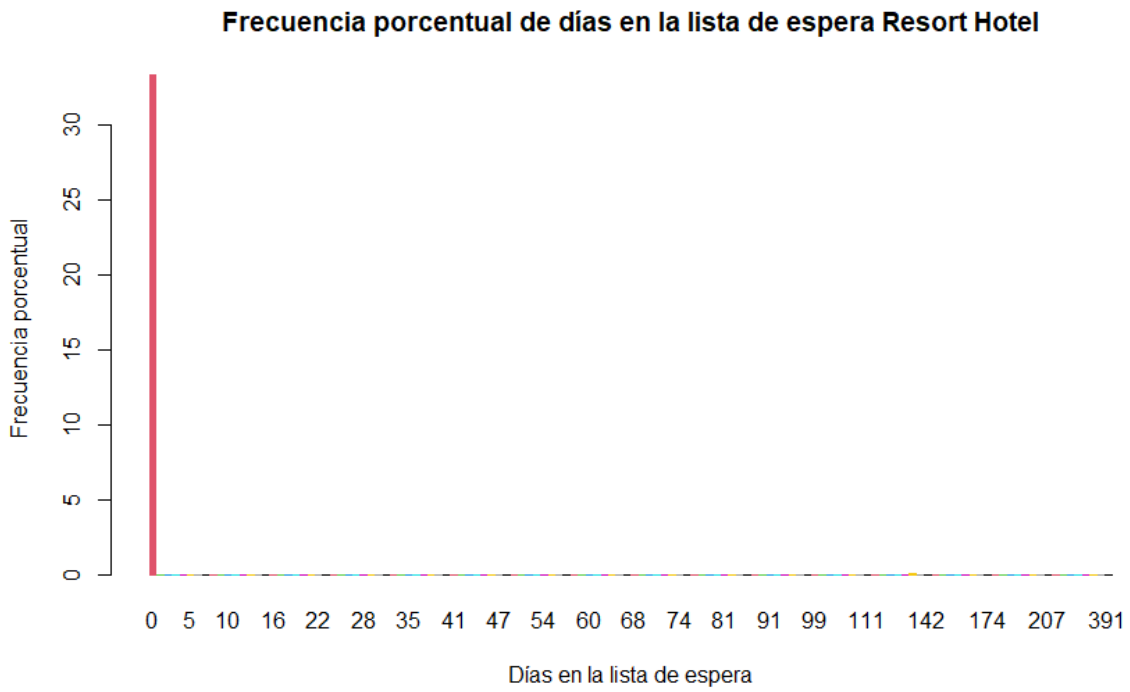
```
                0     1
City Hotel   46228 33102
Resort Hotel 28938 11122
```

Para hacer el gráfico de barras que considera las reservas canceladas de ambos hoteles se usó la función barplot(). Esta función recibe la tabla Can, el título del gráfico, los nombres de los ejes x e y, los colores que tendrá. Para hacer la tabla Can, se usan los valores de las variables CanCity y CanResort, que contienen sólo los valores cancelados de cada hotel.

```r
CanCity <- Cancelado[1,2] #Reservas canceladas en el City Hotel
CanResort <- Cancelado [2,2] #Reservas canceladas en el Resort Hotel
Can <- c(CanCity, CanResort) #Tabla con las reservas canceladas de ambos hoteles
hoteles <- c("CityHotel", "Resort Hotel")

barplot(Can,
     main = "Reservas canceladas en ambos hoteles",
     xlab = "Tipo de hotel", ylab = "Cantidad reservas canceladas",
     col = c("lightcoral", "hotpink"))

legend("topright", legend = hoteles,
       fill =  c("lightcoral", "hotpink"))
```

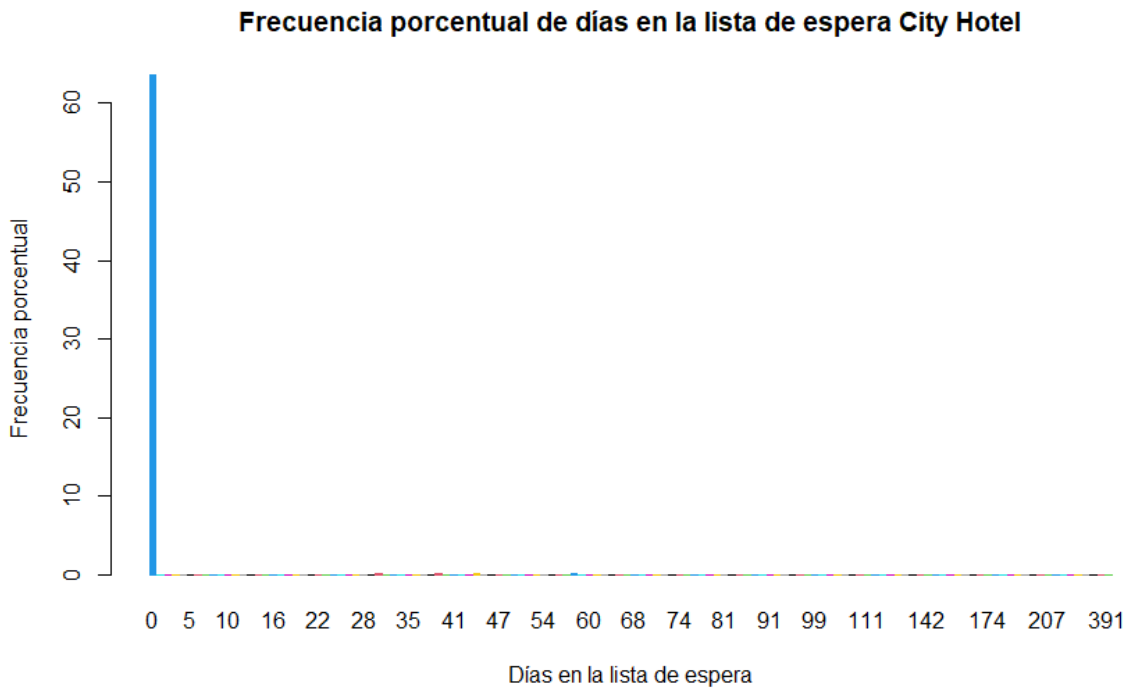
Para hacer el gráfico con las frecuencias porcentuales de los días de espera en el Resort Hotel, se hizo un barplot(). Esta función recibe como parámetros la tabla de las frecuencias porcentuales del Resort Hotel (F_Acum_Resort), el título, los títulos de los ejes, colores de las columnas y los bordes de las columnas. No se le hizo una leyenda a este gráfico ya que tiene demasiados elementos por lo que no sería legible ni factible.

```r
barplot(F_Acum_Resort,
        main = "Frecuencia porcentual de días en la lista de espera Resort Hotel",
        xlab = "Días en la lista de espera", ylab = "Frecuencia porcentual",
        col = c(2:129),
        border = c(2:129))
```



Para hacer el gráfico con las frecuencias porcentuales de los días de espera en el City Hotel, se hizo un barplot(). Esta función recibe como parámetros la tabla de las frecuencias porcentuales del City Hotel (F_Acum_City), el título, los títulos de los ejes, colores de las columnas y los bordes de las columnas. No se le hizo una leyenda a este gráfico ya que tiene demasiados elementos por lo que no sería legible ni factible.

```
barplot(F_Acum_City,
        main = "Frecuencia porcentual de días en la lista de espera City Hotel",
        xlab = "Días en la lista de espera", ylab = "Frecuencia porcentual",
        col = c(100: 228),
        border = c(100:228))
```



A simple vista de las visualizaciones se puede inferir que el Resort Hotel tiene menos cancelaciones y que se pasan menos días en la lista de espera.

Conclusiones preliminares

A partir de las conclusiones, se pueden obtener diversas conclusiones. La primera es que los estacionamientos no es importante contar con espacios de estacionamiento para el Resort Hotel. La tabla A muestra que 34570 no requieren estacionamiento y sólo 5490 requieren un estacionamiento. Por lo que la gran mayoría de clientes no requiere un estacionamiento, no es importante para la reserva. En el caso del City Hotel, tampoco es importante contar con espacios de estacionamiento. Según la tabla A, 77404 no requieren un estacionamiento y sólo 1926 huéspedes requieren mínimo 1 estacionamiento. Debido a que la gran mayoría no requiere un estacionamiento, se puede concluir que no son importantes para los huéspedes, por lo que no es un requisito para el hotel.

La segunda conclusión es que los huéspedes prefieren el Resort Hotel. Esto se infiere ya que el 72.2% no son canceladas versus el City Hotel que tiene un 58.3% de reservas no canceladas. Como se pueden ver en los gráficos de barras y de pie, el Resort Hotel tiene más reservas que no se cancelan. Otro elemento por el cual se concluyó que los clientes prefieren el Resort Hotel son el los días en la lista de espera. Para ello se hizo una tabla con la frecuencia porcentual de cada día en el que se está en la lista de espera en ambos hoteles. Con esa tabla se pudo determinar que se pasan menos días en la lista de espera del Resort Hotel (0.33%) que en el del City Hotel (0.66%) que puede ser un factor para que los clientes prefieran al Resort Hotel.

La tercera conclusión consiste en que la demanda de hoteles sí ha estado en aumento en los últimos tres años examinados por el data frame, teniendo un auge en el 2016. En cuanto a los resultados mensuales, es un poco más variado, pero hay más aumentos que disminuciones en cuanto a la cantidad de huéspedes, por lo que se puede decir que también hay un aumento general de la demanda.

La cuarta conclusión es que la mayoría de los meses tienen una temporada de reserva media, siendo muy balanceadas las reservas entre cada mes. Las excepciones notables son Noviembre y Diciembre con temporadas bajas y Enero, Julio, Agosto y Octubre, con temporadas altas.

La quinta conclusión es que los meses con mayor cantidad de reservas canceladas son Enero, Julio, Febrero y Octubre. De estos, tres de los cuatro meses también son meses con temporadas altas de reservas, por lo que se puede concluir que mientras más personas reserven en el hotel, es más probable que haya más reservas canceladas.

En la sexta conclusión, las personas usualmente prefieren realizar sus reservas de hotel sin niños y/o bebes. Debido a que, la mayoría de clientes prefieren salir solos o con otros adultos para no tener preocupaciones mientras que están fuera del hotel.

En la séptima conclusión se visualiza que la menor demanda de reserva es cuando empiezan a hacer estados de reservación para el hotel desde Octubre del 2014 hasta mediados del 2015, puesto que, se realiza menor cantidad de reservaciones por parte de los clientes.

Archivar y publicar

Todo este análisis realizado, se encuentra en un repositorio creado en la página web de Github. En este repositorio hay 2 carpetas, una que se llama data y otra que se llama code. La carpeta data contiene el dataset original y el dataset final resultante. La carpeta code contiene los scripts en R utilizados para el proceso de carga, inspección, pre-procesado y visualización del dataset.

A parte de las carpetas, el repositorio deberá tener un archivo .Readme. Este archivo contiene el objetivo, nombre de los integrantes, breve descripción del dataset y el link del pdf, conclusiones y la licencia utilizada para el proyecto.

Link del repositorio en github: https://github.com/OscarFloresP/ea-2021-1-cc51
Nombre del repositorio: ea-2021-1-cc51

Bibliografía:

Almeida. A, Antonio. N, Nunes. L (2019). *Hotel booking demand datasets*. Recuperado de https://www.sciencedirect.com/science/article/pii/S2352340918315191 [Consultado el 2 de mayo del 2021.]

Elsevier B.V, ScienceDirect (S/F). *Data in Brief*. Recuperado de https://www.sciencedirect.com/journal/data-in-brief [Consultado el 2 de mayo del 2021].

Mostipak. J (2020). *Hotel booking demand, From the paper: hotel booking demand datasets*. Recuperado de https://www.kaggle.com/jessemostipak/hotel-booking-demand?select=hotel_bookings.csv [Consultado el 2 de mayo del 2021].

ResearchGate (S/F). *Ana Maria De Almeida*. Recuperado de https://www.researchgate.net/profile/Ana-De-Almeida-6 [Consultado el 2 de mayo del 2021].

ResearchGate (S/F). *Nuno Antonio*. Recuperado de https://www.researchgate.net/profile/Nuno-Antonio [Consultado el 2 de mayo del 2021].

ResearchGate (S/F). *Luís Nunes*. Recuperado de https://www.researchgate.net/profile/Luis-Nunes-16 [Consultado el 2 de mayo del 2021].