Statistical Learning project: Hotel booking demand

I.Caria, C.De Luca, M.Pernini

Libraries

```
library(ggplot2)
library(tidyverse)
library(leaps)
library(ggcorrplot)
library(regclass)
library(boot)
library(caret)
library(MASS)
library(knitr)
library(corrplot)
library(glmnet)
library(plotly)
library(pROC)
library(countrycode)
library(confintr)
```

Hotel booking demand dataset

We decided to analyze the *Hotel booking demand dataset* that we load from Kaggle. This dataset contains information about two different kinds of hotel: City Hotel and Resort Hotel. Each observation represents an hotel booking. Both hotels are located in Portugal: the resort hotel at the resort region of Algarve and the city hotel at the city of Lisbon.

The aim of the project is to assess if it is possible, starting from the informations provided by the booking dataset, to predict how likely is that a customer who booked a room will cancel the reservation. This can allow a hotel to plan how many staff are needed, how much food to buy, or, more in general, if it is worth (and to what extent) to engage in overselling in order to fill all available rooms. To achieve this result a prior explanatory analysis is required.

Most of the work deals with analysis and detailed description of the main features with respect to the variable "is_canceled", but we decided at the beginning to take a more general look on the whole set of data, in order to have a broader view of the main informations provided by the dataset.

```
# Load the dataset

hotel_bookings <- read.csv("/Users/matteopernini/Desktop/hotel_bookings.csv", na.strings="NULL")
View(hotel_bookings)
```

Dataset Pre-Processing

The dataset contains 32 variables describing 119390 observations.

In the following lines a detailed description of the variables in alphabetical order is provided:

ADR: the Average Daily Rate, which is the rate obtained by dividing the sum of all lodging transaction by the total number of staying nights;

Adults: the number of adults

Agent: ID of the travel agency that made the booking

ArrivalDateDayOfMonth : Day of the month of the arrival date

ArrivalDateMonth : Month of arrival date with 12 categories: "January" to "December"

ArrivalDateWeekNumber : Week number of the arrival date

ArrivalDateYear : Year of arrival date

AssignedRoomType : Code for the type of room assigned to the booking. Code is presented instead of designation for anonymity reasons

Babies : Number of babies

BookingChanges : Number of changes/amendments made to the booking from the moment the booking was entered on the PMS until the moment of check-in or cancellation

Children : Number of children

Company : ID of the company/entity that made the booking or responsible for paying the booking. ID is presented instead of designation for anonymity reasons

Country : Country of origin. Categories are represented in the ISO 3155–3:2013 format

CustomerType : Type of booking, assuming one of four categories: - Contract : when the booking has an allotment or other type of contract associated to it; - Group : when the booking is associated to a group; - Transient : when the booking is not part of a group or contract, and is not associated to other transient booking; - Transient-party : when the booking is transient, but is associated to at least other transient booking

DaysInWaitingList : Number of days the booking was in the waiting list before it was confirmed to the customer

DepositType : Indication on if the customer made a deposit to guarantee the booking. This variable can assume three categories: - No Deposit : no deposit was made; - Non Refund : a deposit was made in the value of the total stay cost; - Refundable : a deposit was made with a value under the total cost of stay

DistributionChannel : Booking distribution channel. The term "TA" means "Travel Agents" and "TO" means "Tour Operators"

IsCanceled : Value indicating if the booking was canceled (1) or not (0)

IsRepeatedGuest : Value indicating if the booking name was from a repeated guest (1) or not (0)

LeadTime : Number of days that elapsed between the entering date of the booking into the PMS and the arrival date

MarketSegment : Market segment designation. In categories, the term "TA" means "Travel Agents" and "TO" means "Tour Operators"

Meal : Type of meal booked. Categories are presented in standard hospitality meal packages: - Undefined/SC : no meal package; - BB : Bed & Breakfast; - HB : Half board (breakfast and one other meal – usually dinner); - FB : Full board (breakfast, lunch and dinner)

PreviousBookingsNotCanceled : Number of previous bookings not cancelled by the customer prior to the current booking

PreviousCancellations : Number of previous bookings that were cancelled by the customer prior to the current booking

RequiredCardParkingSpaces : Number of car parking spaces required by the customer

ReservationStatus : Reservation last status, assuming one of three categories: - Canceled : booking was canceled by the customer; - Check-Out : customer has checked in but already departed; - No-Show : customer did not check-in and did inform the hotel of the reason why

ReservationStatusDate : Date at which the last status was set

ReservedRoomType : Code of room type reserved. Code is presented instead of designation for anonymity reasons

StaysInWeekendNights : Number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay at the hotel

StaysInWeekNights : Number of week nights (Monday to Friday) the guest stayed or booked to stay at the hotel

TotalOfSpecialRequests : Number of special requests made by the customer (e.g. twin bed or high floor)

```
# First look to the dataset

glimpse(hotel_bookings)
```

```
## Rows: 119,390
## Columns: 32
## $ hotel                          <chr> "Resort Hotel", "Resort Hotel", "Resort~
## $ is_canceled                    <int> 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, ~
## $ lead_time                      <int> 342, 737, 7, 13, 14, 14, 0, 9, 85, 75, ~
## $ arrival_date_year              <int> 2015, 2015, 2015, 2015, 2015, 2015, 201~
## $ arrival_date_month             <chr> "July", "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, 1, 1, 1, ~
## $ stays_in_weekend_nights        <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ stays_in_week_nights           <int> 0, 0, 1, 1, 2, 2, 2, 2, 3, 3, 4, 4, 4, ~
## $ adults                         <int> 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, ~
## $ children                       <chr> "0", "0", "0", "0", "0", "0", "0", "0",~
## $ babies                         <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ meal                           <chr> "BB", "BB", "BB", "BB", "BB", "BB", "BB~
## $ country                        <chr> "PRT", "PRT", "GBR", "GBR", "GBR", "GBR~
## $ market_segment                 <chr> "Direct", "Direct", "Direct", "Corporat~
## $ distribution_channel           <chr> "Direct", "Direct", "Direct", "Corporat~
## $ is_repeated_guest              <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ previous_cancellations         <int> 0, 0, 0, 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, 0, 0, 0, ~
## $ reserved_room_type             <chr> "C", "C", "A", "A", "A", "A", "C", "C",~
## $ assigned_room_type             <chr> "C", "C", "C", "A", "A", "A", "C", "C",~
## $ booking_changes                <int> 3, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ deposit_type                   <chr> "No Deposit", "No Deposit", "No Deposit~
## $ agent                          <int> NA, NA, NA, 304, 240, 240, NA, 303, 240~
## $ company                        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ days_in_waiting_list           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ customer_type                  <chr> "Transient", "Transient", "Transient", ~
```

```
## $ adr                            <dbl> 0.00, 0.00, 75.00, 75.00, 98.00, 98.00,~
## $ required_car_parking_spaces    <int> 0, 0, 0, 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, 0, 0, 3, ~
## $ reservation_status             <chr> "Check-Out", "Check-Out", "Check-Out", ~
## $ reservation_status_date        <chr> "2015-07-01", "2015-07-01", "2015-07-02~
```

As we can see from the code above, there are many character variables that we converted into factors. Furthermore, we noticed that some categorical variables like *children* were numeric, so we converted them.

```
# Convert character columns into factors

hotel_bookings_new <- as.data.frame(unclass(hotel_bookings),
                       stringsAsFactors = TRUE)

# Convert binary columns "is_canceled" and "is_repeated_guest" into factor

hotel_bookings_new$is_canceled <- as.factor(hotel_bookings_new$is_canceled)
levels(hotel_bookings_new$is_canceled) <- c(0, 1)

hotel_bookings_new$is_repeated_guest <- as.factor(hotel_bookings_new$is_repeated_guest)
levels(hotel_bookings_new$is_repeated_guest) <- c(0, 1)

# Convert column "arrival_date_year" into factor

hotel_bookings_new$arrival_date_year <- as.factor(hotel_bookings_new$arrival_date_year)
levels(hotel_bookings_new$arrival_date_year) <- c("2015", "2016", "2017")

# Convert column "children" into numeric

hotel_bookings_new$children <- as.numeric(as.character(hotel_bookings_new$children))

# Convert column "reservation status date" into date

hotel_bookings_new$reservation_status_date <- as.Date(
  hotel_bookings_new$reservation_status_date, format = "%Y-%m-%d")
```

The dataset provides two different variables for the stay: *stays_in_weekend_nights* and *stays_in_week_nights*. We decided to add the sum of these two variables as a new variable *total_stays* for ease of analyses.

```
# New column for total stays

hotel_bookings_new=hotel_bookings_new%>%
  mutate(total_stays=(stays_in_week_nights + stays_in_weekend_nights) )
```

Missing values

```
colSums(is.na(hotel_bookings_new))[colSums(is.na(hotel_bookings_new))>0]
```

```
## children  country    agent  company
##        4      488    16340   112593
```

Since there are only 4 Nan values for the variable *children*, we decided to replace them with the value 0. The variables *agent* and *company* have too many Nan values, therefore we removed them. We left untouched the variable *country* because we did not use it in our models.

```
# Replacing missing values in children column from the corresponding babies column

n_children <- length(hotel_bookings_new$children)
for (i in 1:n_children) {
  if (is.na(hotel_bookings_new$children[i]))
    hotel_bookings_new$children[i] <- 0
}

# Remove columns "agent" and "company"

index_agent <- which(colnames(hotel_bookings_new)=="agent")
index_company <- which(colnames(hotel_bookings_new)=="company")
hotel_bookings_new = hotel_bookings_new[-c(index_agent, index_company)]
```

At the end of the pre-processing, we obtained the following dataset:

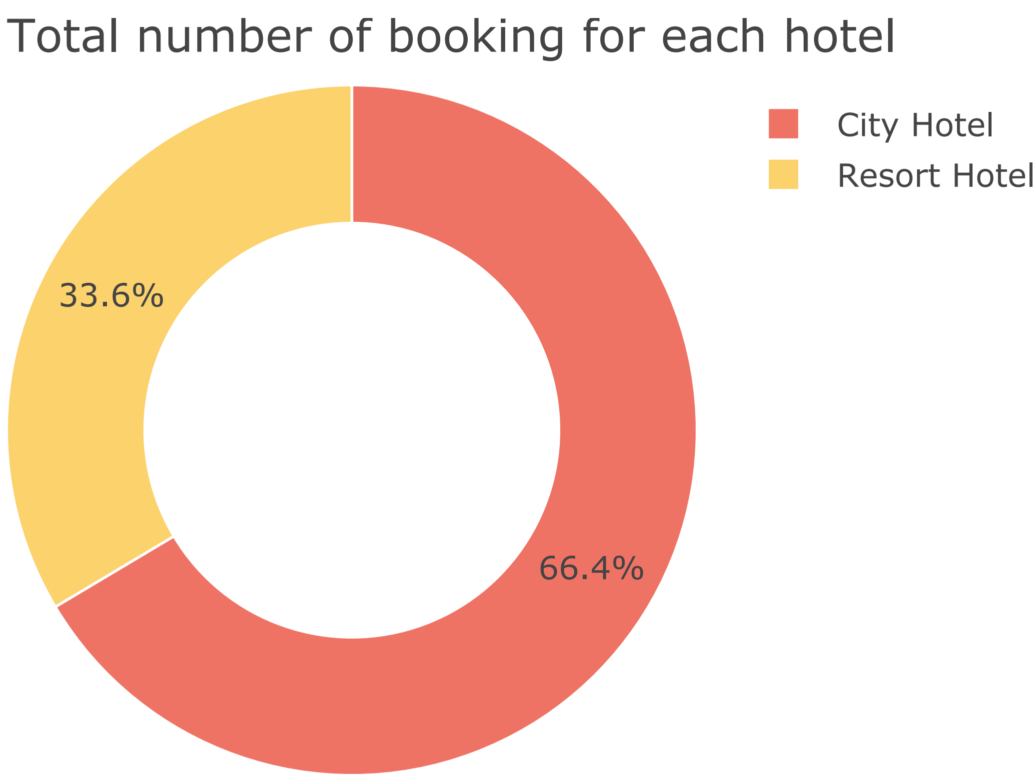
```
##           hotel       is_canceled   lead_time   arrival_date_year
##  City Hotel  :79330   0:75166     Min.   :  0   2015:21996
##  Resort Hotel:40060   1:44224     1st Qu.: 18   2016:56707
##                                   Median : 69   2017:40687
##                                   Mean   :104
##                                   3rd Qu.:160
##                                   Max.   :737
##
##  arrival_date_month arrival_date_week_number arrival_date_day_of_month
##  August :13877      Min.   : 1.00            Min.   : 1.0
##  July   :12661      1st Qu.:16.00            1st Qu.: 8.0
##  May    :11791      Median :28.00            Median :16.0
##  October:11160      Mean   :27.17            Mean   :15.8
##  April  :11089      3rd Qu.:38.00            3rd Qu.:23.0
##  June   :10939      Max.   :53.00            Max.   :31.0
##  (Other):47873
##  stays_in_weekend_nights stays_in_week_nights     adults
##  Min.   : 0.0000         Min.   : 0.0         Min.   : 0.000
##  1st Qu.: 0.0000         1st Qu.: 1.0         1st Qu.: 2.000
##  Median : 1.0000         Median : 2.0         Median : 2.000
##  Mean   : 0.9276         Mean   : 2.5         Mean   : 1.856
##  3rd Qu.: 2.0000         3rd Qu.: 3.0         3rd Qu.: 2.000
##  Max.   :19.0000         Max.   :50.0         Max.   :55.000
##
##     children           babies                 meal           country
##  Min.   : 0.0000   Min.   : 0.000000   BB       :92310   PRT    :48590
##  1st Qu.: 0.0000   1st Qu.: 0.000000   FB       :  798   GBR    :12129
##  Median : 0.0000   Median : 0.000000   HB       :14463   FRA    :10415
##  Mean   : 0.1039   Mean   : 0.007949   SC       :10650   ESP    : 8568
##  3rd Qu.: 0.0000   3rd Qu.: 0.000000   Undefined: 1169   DEU    : 7287
##  Max.   :10.0000   Max.   :10.000000                     (Other):31913
##                                                          NA's   :  488
##       market_segment  distribution_channel is_repeated_guest
##  Online TA    :56477   Corporate: 6677      0:115580
```

```
## Offline TA/TO:24219   Direct   :14645      1:  3810
## Groups       :19811   GDS      :  193
## Direct       :12606   TA/TO    :97870
## Corporate    : 5295   Undefined:    5
## Complementary:  743
## (Other)      :  239
## previous_cancellations previous_bookings_not_canceled reserved_room_type
## Min.   : 0.00000       Min.   : 0.0000                A      :85994
## 1st Qu.: 0.00000       1st Qu.: 0.0000                D      :19201
## Median : 0.00000       Median : 0.0000                E      : 6535
## Mean   : 0.08712       Mean   : 0.1371                F      : 2897
## 3rd Qu.: 0.00000       3rd Qu.: 0.0000                G      : 2094
## Max.   :26.00000       Max.   :72.0000                B      : 1118
##                                                       (Other): 1551
## assigned_room_type booking_changes      deposit_type    days_in_waiting_list
## A      :74053      Min.   : 0.0000   No Deposit:104641   Min.   :  0.000
## D      :25322      1st Qu.: 0.0000   Non Refund: 14587   1st Qu.:  0.000
## E      : 7806      Median : 0.0000   Refundable:   162   Median :  0.000
## F      : 3751      Mean   : 0.2211                       Mean   :  2.321
## G      : 2553      3rd Qu.: 0.0000                       3rd Qu.:  0.000
## C      : 2375      Max.   :21.0000                       Max.   :391.000
## (Other): 3530
##         customer_type        adr          required_car_parking_spaces
## Contract       : 4076   Min.   :  -6.38   Min.   :0.00000
## Group          :  577   1st Qu.:  69.29   1st Qu.:0.00000
## Transient      :89613   Median :  94.58   Median :0.00000
## Transient-Party:25124   Mean   : 101.83   Mean   :0.06252
##                         3rd Qu.: 126.00   3rd Qu.:0.00000
##                         Max.   :5400.00   Max.   :8.00000
##
## total_of_special_requests reservation_status reservation_status_date
## Min.   :0.0000            Canceled :43017    Min.   :2014-10-17
## 1st Qu.:0.0000            Check-Out:75166    1st Qu.:2016-02-01
## Median :0.0000            No-Show  : 1207    Median :2016-08-07
## Mean   :0.5714                               Mean   :2016-07-30
## 3rd Qu.:1.0000                               3rd Qu.:2017-02-08
## Max.   :5.0000                               Max.   :2017-09-14
##
##  total_stays
## Min.   : 0.000
## 1st Qu.: 2.000
## Median : 3.000
## Mean   : 3.428
## 3rd Qu.: 4.000
## Max.   :69.000
##
```

EDA

The dataset is made of two original datasets with hotel demand data. One of the hotels is a City Hotel and the other one is a Resort Hotel. The first thing to notice is that data are quite unbalanced and there are 79330 observations for the former and 40060 for the latter.

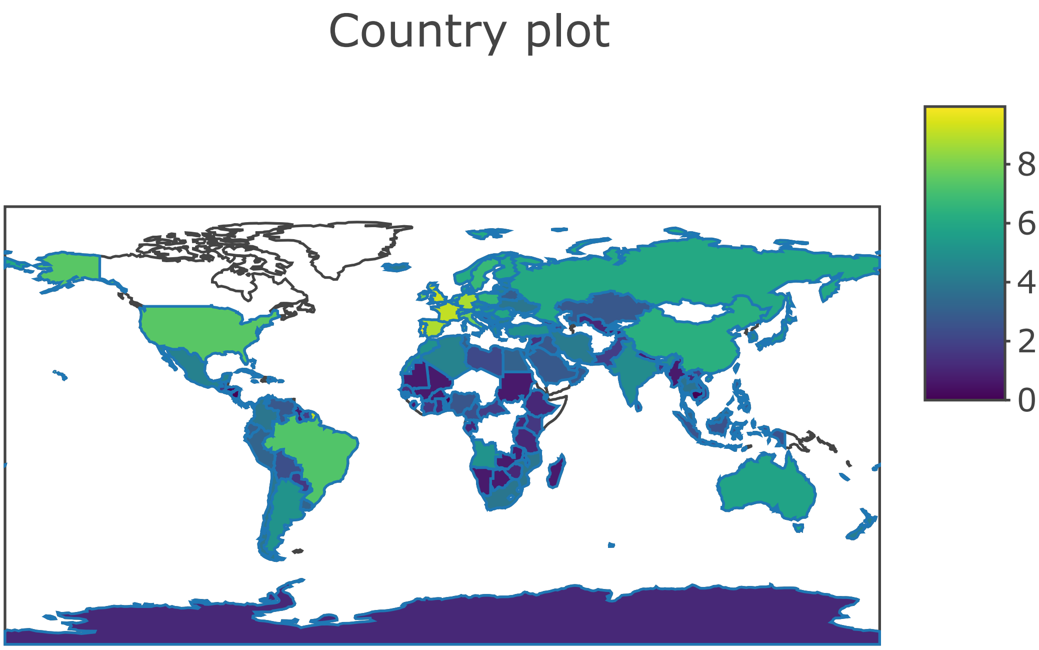
```r
# Hotel donut plot

df_hotel <- as.data.frame(hotel_bookings_new[, c("hotel")])
df_hotel <- as.data.frame(lapply(df_hotel, function(x) as.data.frame(table(x))))

colnames(df_hotel) <- c("hotel", "frequency")

colors_donut <- c('rgb((239,115,101))','rgb((251,210,108))')

fig_hotel <- df_hotel %>% plot_ly(labels = ~hotel, values = ~frequency,
                                  marker = list(colors = colors_donut,
                                  line = list(color = '#FFFFFF', width = 1)))
fig_hotel <- fig_hotel %>% add_pie(hole = 0.6)
fig_hotel <- fig_hotel %>% layout(title = "Total number of booking for each hotel",
             showlegend = T,
             xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
             yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE))
fig_hotel
```



Both hotels are located in Portugal; this is the reason why most of the guests come from Portugal, as we can see from the map plot below:

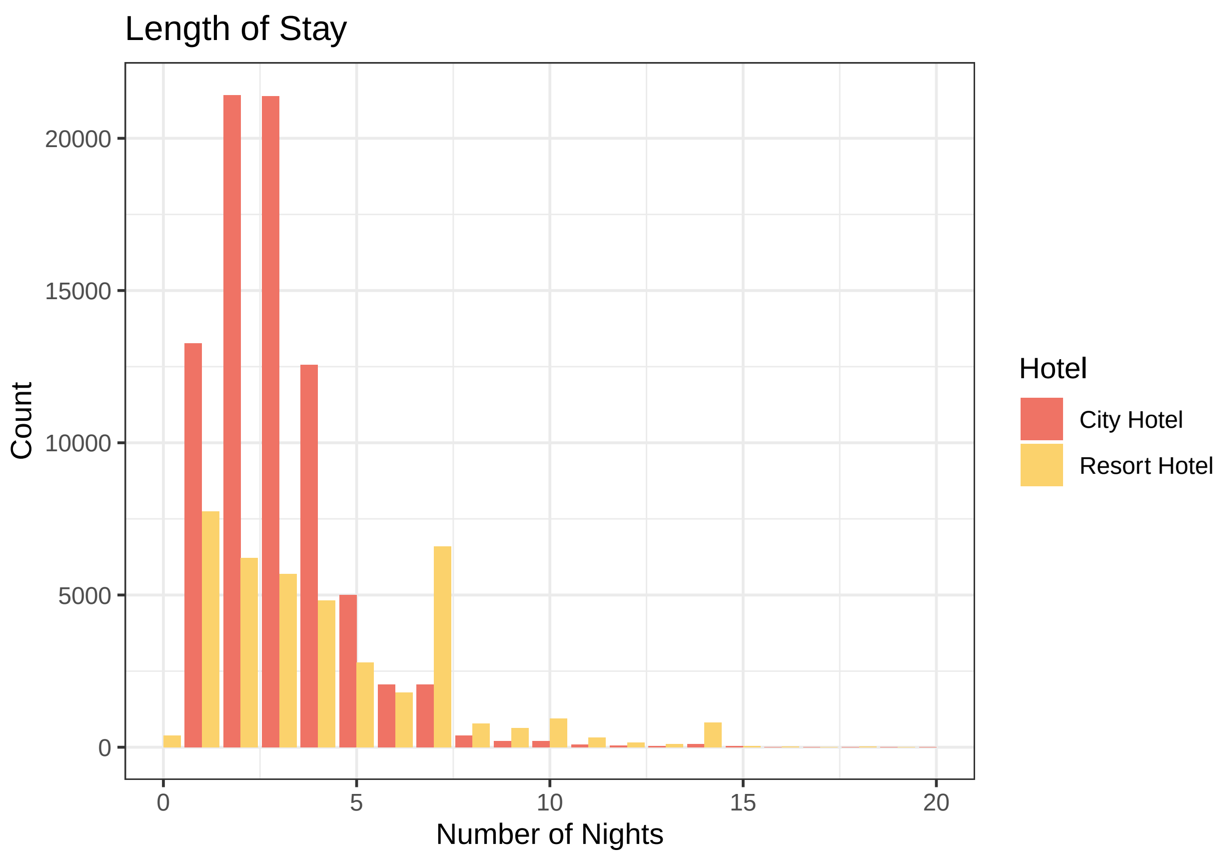
```r
# Country plot

df_country <- as.data.frame(
              hotel_bookings_new[hotel_bookings_new$is_canceled==0, c("country")])
```

```
df_country <- as.data.frame(lapply(df_country, function(x) as.data.frame(table(x))))
colnames(df_country) <- c('country', 'frequency')

# we used log-scale to see better the different number of guests for each country
fig_country <- plot_ly(df_country, type='choropleth', locations=df_country$country,
                       z=log(df_country$frequency+1), colorscale = 'Viridis' )
fig_country <- fig_country %>% layout(title = "Country plot")
fig_country
```



The following plot points out the difference between the two kind of hotels, since most of the reservations related to the City Hotel last approximately 2-3 nights, while in case of the Resort Hotel the same pattern is still observed, but in the meantime also 7 nights stand out as being a very popular choice among guests. More in general a long stay is very unusual in case of the City Hotel.

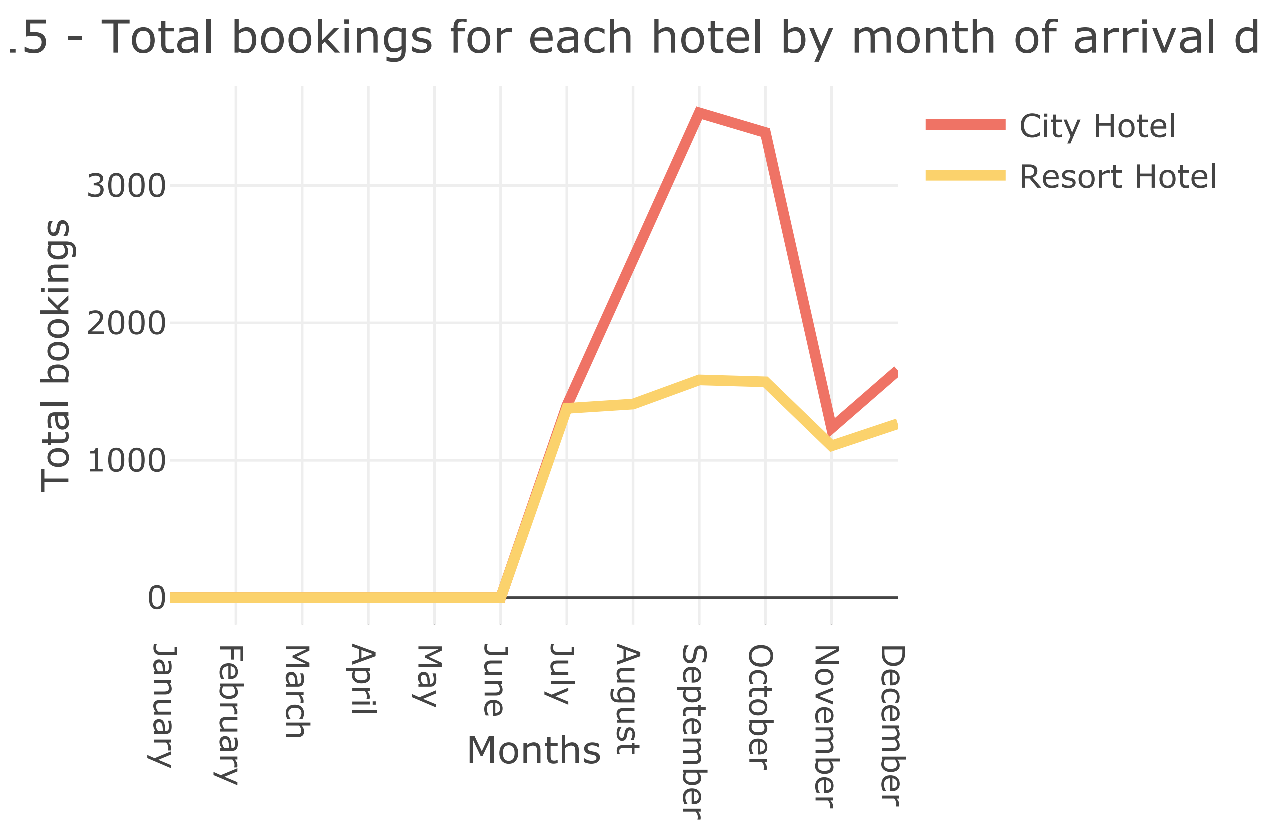
```
# Length of stays in night
ggplot(hotel_bookings_new, aes(x=total_stays, fill = hotel)) +
  geom_bar(stat = "count", position = position_dodge()) +
  scale_fill_manual(values=c("#EF7365", "#FBD26C"),
    name = "Hotel",
    breaks = c("City Hotel", "Resort Hotel"),
    labels = c("City Hotel", "Resort Hotel")
  ) +
  labs(title = "Length of Stay",
       x = "Number of Nights",
       y = "Count") + xlim(0,20) +
  theme_bw()
```



The dataset comprehends observations from three years, between the 1st of July of 2015 and the 31st of August 2017, including bookings that effectively arrived and bookings that were canceled. Another information provided by the dataset is the month of arrival, which allows us to take a look at the pattern of the booking curves month by month. We first made the plot, for each year, of total bookings by month of arrival date, separating the two types of hotels.

```
# Total bookings for each hotel by month (year: 2015)

df_months_City_2015 <- as.data.frame(
              hotel_bookings_new[hotel_bookings_new$hotel=='City Hotel'
              & hotel_bookings_new$arrival_date_year==2015, c( "arrival_date_month")])

df_months_Resort_2015 <- as.data.frame(
              hotel_bookings_new[hotel_bookings_new$hotel=='Resort Hotel'
              & hotel_bookings_new$arrival_date_year==2015, c( "arrival_date_month")])

df_months_City_2015 <- as.data.frame(lapply(df_months_City_2015, function(x)
  as.data.frame(table(x))))

df_months_Resort_2015 <- as.data.frame(lapply(df_months_Resort_2015, function(x)
  as.data.frame(table(x))))

colnames(df_months_City_2015) <- c("arrival_date_month_City", "frequency_City" )

colnames(df_months_Resort_2015) <- c("arrival_date_month_Resort", "frequency_Resort")
```

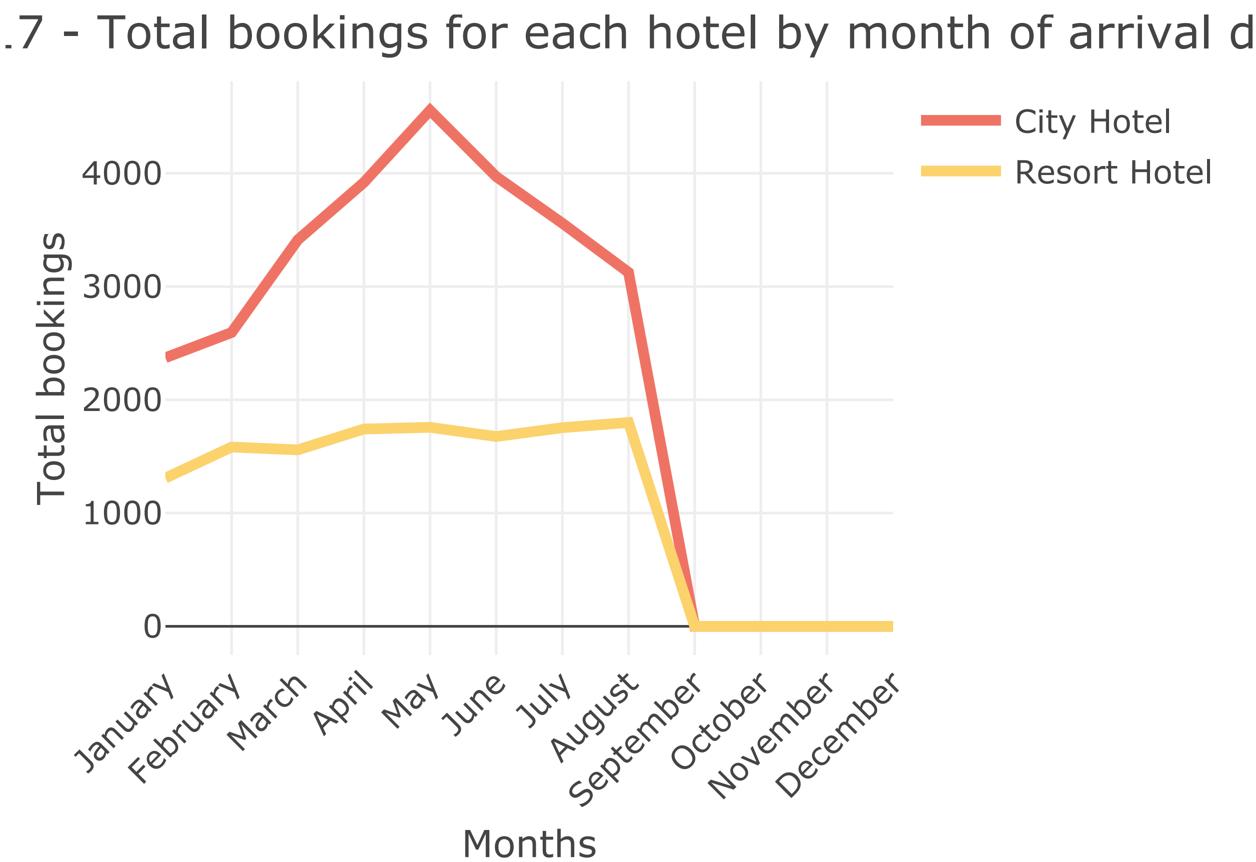
```
df_months_City_2015$arrival_date_month_City <- factor(
            df_months_City_2015$arrival_date_month_City,
            levels = c("January", "February", "March","April", "May", "June", "July",
                       "August", "September", "October", "November", "December"))

df_months_Resort_2015$arrival_date_month_Resort <- factor(
            df_months_Resort_2015$arrival_date_month_Resort,
            levels = c("January", "February", "March","April", "May", "June", "July",
                       "August", "September", "October", "November", "December"))


fig_months_2015 <- plot_ly()

fig_months_2015 <- fig_months_2015 %>% add_lines(data=df_months_City_2015,
                   x = ~arrival_date_month_City, y = ~frequency_City, name = 'City Hotel',
                   type = 'scatter', mode = 'lines',
                   line = list(color = 'rgb((239,115,101))', width = 4))

fig_months_2015 <- fig_months_2015 %>% add_lines(data=df_months_Resort_2015,
                   x = ~arrival_date_month_Resort, y = ~frequency_Resort,
                   name = 'Resort Hotel', type = 'scatter', mode = 'lines',
                   line = list(color = 'rgb((251,210,108))', width = 4))

fig_months_2015 <- fig_months_2015 %>% layout(
         title = "2015 - Total bookings for each hotel by month of arrival date",
         xaxis = list(title = "Months"),
         yaxis = list (title = "Total bookings"))

fig_months_2015
```




```r
# Total bookings for each hotel by month (year: 2016)

df_months_City_2016 <- as.data.frame(
                       hotel_bookings_new[hotel_bookings_new$hotel=='City Hotel'
                       & hotel_bookings_new$arrival_date_year==2016,
                       c( "arrival_date_month")])

df_months_Resort_2016 <- as.data.frame(
                         hotel_bookings_new[hotel_bookings_new$hotel=='Resort Hotel'
                         & hotel_bookings_new$arrival_date_year==2016,
                         c( "arrival_date_month")])

df_months_City_2016 <- as.data.frame(lapply(df_months_City_2016,
                                         function(x) as.data.frame(table(x))))

df_months_Resort_2016 <- as.data.frame(lapply(df_months_Resort_2016,
                                           function(x) as.data.frame(table(x))))

colnames(df_months_City_2016) <- c("arrival_date_month_City", "frequency_City" )

colnames(df_months_Resort_2016) <- c("arrival_date_month_Resort", "frequency_Resort")

df_months_City_2016$arrival_date_month_City <- factor(
         df_months_City_2016$arrival_date_month_City,
         levels = c("January", "February", "March","April", "May", "June", "July",
                    "August", "September", "October", "November", "December"))
```

```
df_months_Resort_2016$arrival_date_month_Resort <- factor(
         df_months_Resort_2016$arrival_date_month_Resort,
         levels = c("January", "February", "March","April", "May", "June", "July",
                    "August", "September", "October", "November", "December"))


fig_months_2016 <- plot_ly()

fig_months_2016 <- fig_months_2016 %>% add_lines(data=df_months_City_2016,
                       x = ~arrival_date_month_City, y = ~frequency_City,
                       name = 'City Hotel', type = 'scatter', mode = 'lines',
                       line = list(color = 'rgb((239,115,101))', width = 4))

fig_months_2016 <- fig_months_2016 %>% add_lines(data=df_months_Resort_2016,
                       x = ~arrival_date_month_Resort, y = ~frequency_Resort,
                       name = 'Resort Hotel', type = 'scatter', mode = 'lines',
                       line = list(color = 'rgb((251,210,108))', width = 4))

fig_months_2016 <- fig_months_2016 %>% layout(
         title = "2016 - Total bookings for each hotel by month of arrival date",
         xaxis = list(title = "Months",tickangle = -45),
         yaxis = list (title = "Total bookings"))

fig_months_2016
```

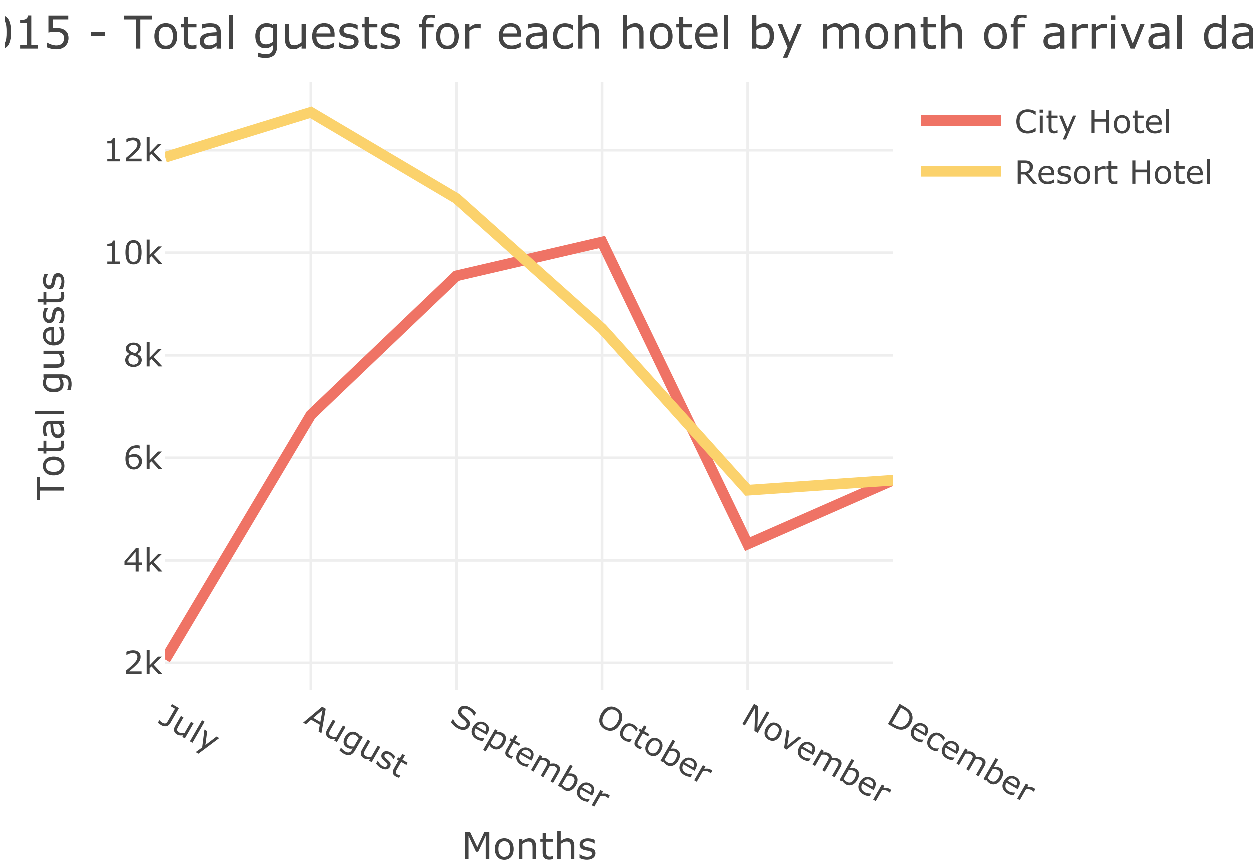
```r
# Total bookings for each hotel by month (year: 2017)

df_months_City_2017 <- as.data.frame(
                       hotel_bookings_new[hotel_bookings_new$hotel=='City Hotel'
                       & hotel_bookings_new$arrival_date_year==2017,
                       c( "arrival_date_month")])

df_months_Resort_2017 <- as.data.frame(
                         hotel_bookings_new[hotel_bookings_new$hotel=='Resort Hotel'
                         & hotel_bookings_new$arrival_date_year==2017,
                         c( "arrival_date_month")])

df_months_City_2017 <- as.data.frame(lapply(df_months_City_2017,
                                            function(x) as.data.frame(table(x))))

df_months_Resort_2017 <- as.data.frame(lapply(df_months_Resort_2017,
                                            function(x) as.data.frame(table(x))))

colnames(df_months_City_2017) <- c("arrival_date_month_City", "frequency_City" )

colnames(df_months_Resort_2017) <- c("arrival_date_month_Resort", "frequency_Resort")

df_months_City_2017$arrival_date_month_City <- factor(
         df_months_City_2017$arrival_date_month_City,
         levels = c("January", "February", "March","April", "May", "June", "July",
                    "August", "September", "October", "November", "December"))

df_months_Resort_2017$arrival_date_month_Resort <- factor(
         df_months_Resort_2017$arrival_date_month_Resort,
         levels = c("January", "February", "March","April", "May", "June", "July",
                    "August", "September", "October", "November", "December"))


fig_months_2017 <- plot_ly()

fig_months_2017 <- fig_months_2017 %>% add_lines(data=df_months_City_2017,
                      x = ~arrival_date_month_City, y = ~frequency_City,
                      name = 'City Hotel', type = 'scatter', mode = 'lines',
                      line = list(color = 'rgb((239,115,101))', width = 4))

fig_months_2017 <- fig_months_2017 %>% add_lines(data=df_months_Resort_2017,
                      x = ~arrival_date_month_Resort, y = ~frequency_Resort,
                      name = 'Resort Hotel', type = 'scatter', mode = 'lines',
                      line = list(color = 'rgb((251,210,108))', width = 4))

fig_months_2017 <- fig_months_2017 %>% layout(
         title = "2017 - Total bookings for each hotel by month of arrival date",
         xaxis = list(title = "Months",tickangle = -45),
         yaxis = list (title = "Total bookings"))

fig_months_2017
```



In the plot of 2016 a strange behavior is displayed: in the summertime period we expected to find a high number of bookings, however from the graph we can see a slight decrease. Therefore we decided to make the same kind of plots as before, but analyzing the total number of guests in the hotel over the various months, rather than the total number of reservations. From the plots below, we can clearly see that in the summer period of 2016 there is the highest number of guests present compared to the rest of the year, thus showing in effect that the two hotels host more people between July and August.

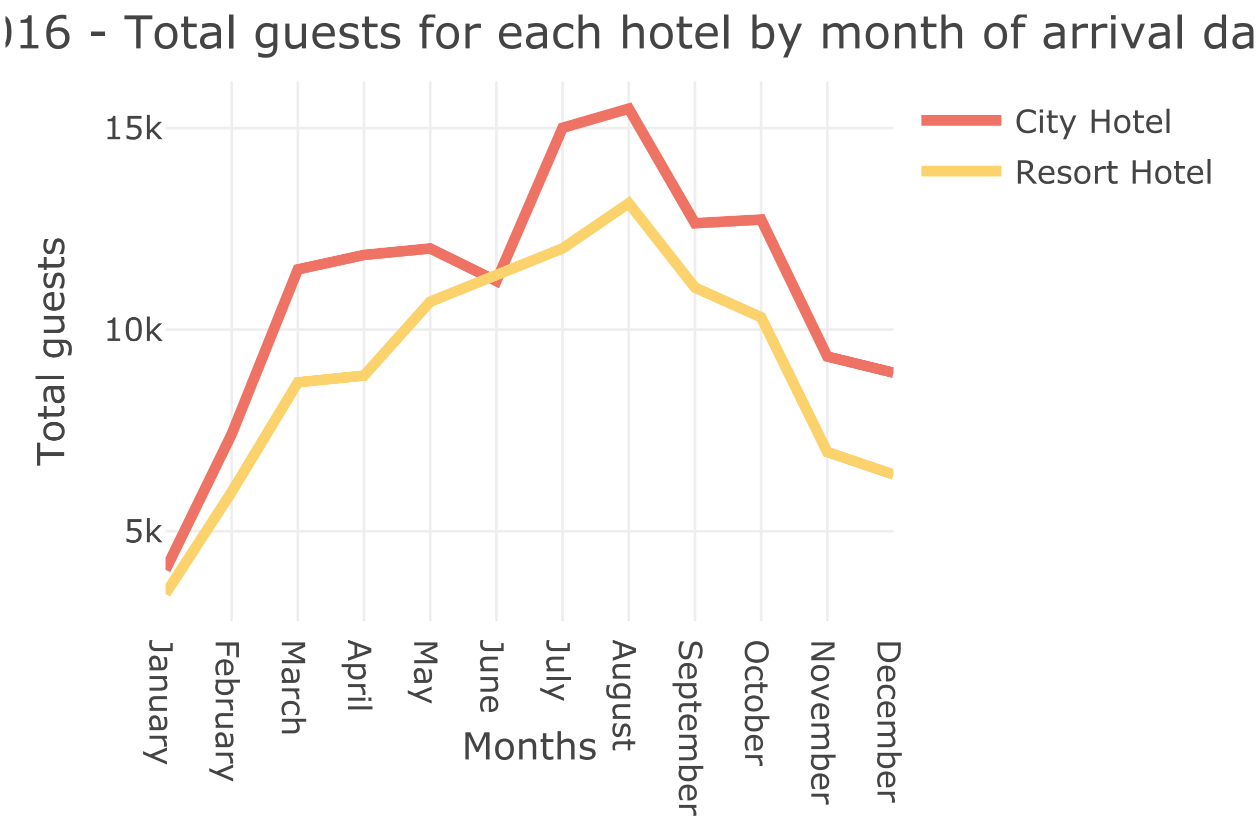
```
# Total guests for each hotel by month (year: 2015)

df_2015_City <- as.data.frame(hotel_bookings_new[hotel_bookings_new$is_canceled==0
                              & hotel_bookings_new$hotel=='City Hotel'
                              & hotel_bookings_new$arrival_date_year==2015,
                              c( "arrival_date_month", "total_stays", "adults",
                                 "children", "babies")])

df_guests_2015_City <- df_2015_City %>%
  group_by(arrival_date_month) %>%
  summarise(guests = sum(total_stays*(adults+children+babies))) %>%
  ungroup()

df_2015_Resort <- as.data.frame(hotel_bookings_new[hotel_bookings_new$is_canceled==0
                                & hotel_bookings_new$hotel=='Resort Hotel'
                                & hotel_bookings_new$arrival_date_year==2015,
                                c( "arrival_date_month", "total_stays", "adults",
                                   "children", "babies")])
```

```r
df_guests_2015_Resort <- df_2015_Resort %>%
  group_by(arrival_date_month) %>%
  summarise(guests = sum(total_stays*(adults+children+babies))) %>%
  ungroup()


df_guests_2015_City$arrival_date_month <- factor(
                                          df_guests_2015_City$arrival_date_month,
                                          levels = c("July", "August",
                                                     "September", "October",
                                                     "November", "December"))

df_guests_2015_Resort$arrival_date_month <- factor(
                                            df_guests_2015_Resort$arrival_date_month,
                                            levels = c("July", "August",
                                                       "September", "October",
                                                       "November", "December"))


fig_guests_2015 <- plot_ly()

fig_guests_2015 <- fig_guests_2015 %>% add_lines(data=df_guests_2015_City,
                                                 x = ~arrival_date_month,
                                                 y = ~guests, name = 'City Hotel',
                                                 type = 'scatter', mode = 'lines',
                                                 line = list(color = 'rgb((239,115,101))',
                                                             width = 4))

fig_guests_2015 <- fig_guests_2015 %>% add_lines(data=df_guests_2015_Resort,
                                                 x = ~arrival_date_month,
                                                 y = ~guests, name = 'Resort Hotel',
                                                 type = 'scatter', mode = 'lines',
                                                 line = list(color = 'rgb((251,210,108))',
                                                             width = 4))

fig_guests_2015 <- fig_guests_2015 %>% layout(
         title = "2015 - Total guests for each hotel by month of arrival date",
         xaxis = list(title = "Months"),
         yaxis = list (title = "Total guests"))

fig_guests_2015
```



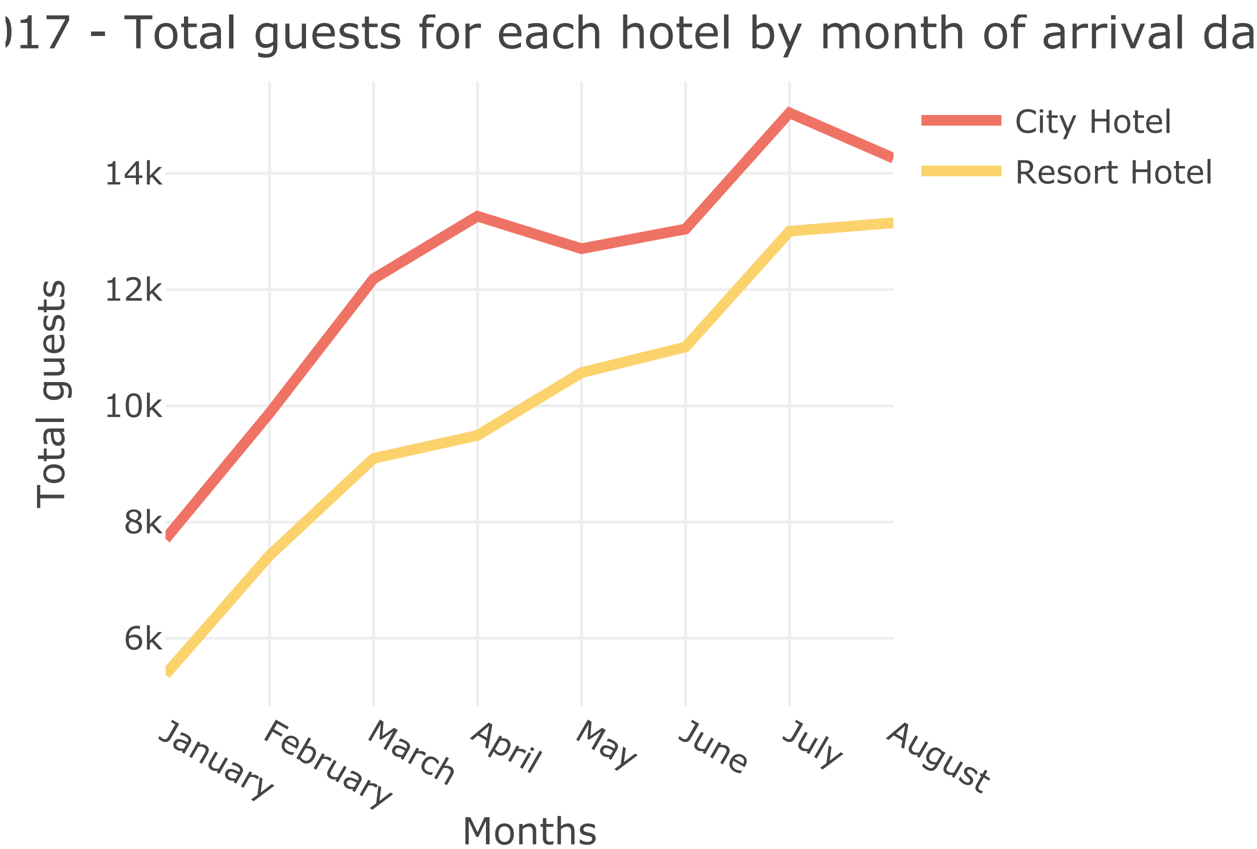

```
# Total guests for each hotel by month (year: 2016)

df_2016_City <- as.data.frame(hotel_bookings_new[hotel_bookings_new$is_canceled==0
                              & hotel_bookings_new$hotel=='City Hotel'
                              & hotel_bookings_new$arrival_date_year==2016,
                              c( "arrival_date_month", "total_stays", "adults",
                                 "children", "babies")])

df_guests_2016_City <- df_2016_City %>%
  group_by(arrival_date_month) %>%
  summarise(guests = sum(total_stays*(adults+children+babies))) %>%
  ungroup()

df_2016_Resort <- as.data.frame(hotel_bookings_new[hotel_bookings_new$is_canceled==0
                                & hotel_bookings_new$hotel=='Resort Hotel'
                                & hotel_bookings_new$arrival_date_year==2016,
                                c( "arrival_date_month", "total_stays", "adults",
                                   "children", "babies")])

df_guests_2016_Resort <- df_2016_Resort %>%
  group_by(arrival_date_month) %>%
  summarise(guests = sum(total_stays*(adults+children+babies))) %>%
  ungroup()


df_guests_2016_City$arrival_date_month <- factor(
```

```
                    df_guests_2016_City$arrival_date_month,
                    levels = c("January", "February", "March","April", "May",
                               "June", "July", "August","September", "October",
                               "November", "December"))

df_guests_2016_Resort$arrival_date_month <- factor(
                      df_guests_2016_Resort$arrival_date_month,
                      levels = c("January", "February", "March","April", "May",
                                 "June", "July", "August", "September", "October",
                                 "November", "December"))


fig_guests_2016 <- plot_ly()

fig_guests_2016 <- fig_guests_2016 %>% add_lines(data=df_guests_2016_City,
                   x = ~arrival_date_month, y = ~guests, name = 'City Hotel',
                   type = 'scatter', mode = 'lines',
                   line = list(color = 'rgb((239,115,101))', width = 4))

fig_guests_2016 <- fig_guests_2016 %>% add_lines(data=df_guests_2016_Resort,
                   x = ~arrival_date_month, y = ~guests, name = 'Resort Hotel',
                   type = 'scatter', mode = 'lines',
                   line = list(color = 'rgb((251,210,108))', width = 4))

fig_guests_2016 <- fig_guests_2016 %>% layout(
         title = "2016 - Total guests for each hotel by month of arrival date",
         xaxis = list(title = "Months"),
         yaxis = list (title = "Total guests"))

fig_guests_2016
```




```
# Total guests for each hotel by month (year: 2017)

df_2017_City <- as.data.frame(hotel_bookings_new[hotel_bookings_new$is_canceled==0
                              & hotel_bookings_new$hotel=='City Hotel'
                              & hotel_bookings_new$arrival_date_year==2017,
                              c( "arrival_date_month", "total_stays", "adults",
                                 "children", "babies")])

df_guests_2017_City <- df_2017_City %>%
  group_by(arrival_date_month) %>%
  summarise(guests = sum(total_stays*(adults+children+babies))) %>%
  ungroup()

df_2017_Resort <- as.data.frame(hotel_bookings_new[hotel_bookings_new$is_canceled==0
                                & hotel_bookings_new$hotel=='Resort Hotel'
                                & hotel_bookings_new$arrival_date_year==2017,
                                c( "arrival_date_month", "total_stays", "adults",
                                   "children", "babies")])

df_guests_2017_Resort <- df_2017_Resort %>%
  group_by(arrival_date_month) %>%
  summarise(guests = sum(total_stays*(adults+children+babies))) %>%
  ungroup()


df_guests_2017_City$arrival_date_month <- factor(df_guests_2017_City$arrival_date_month,
```

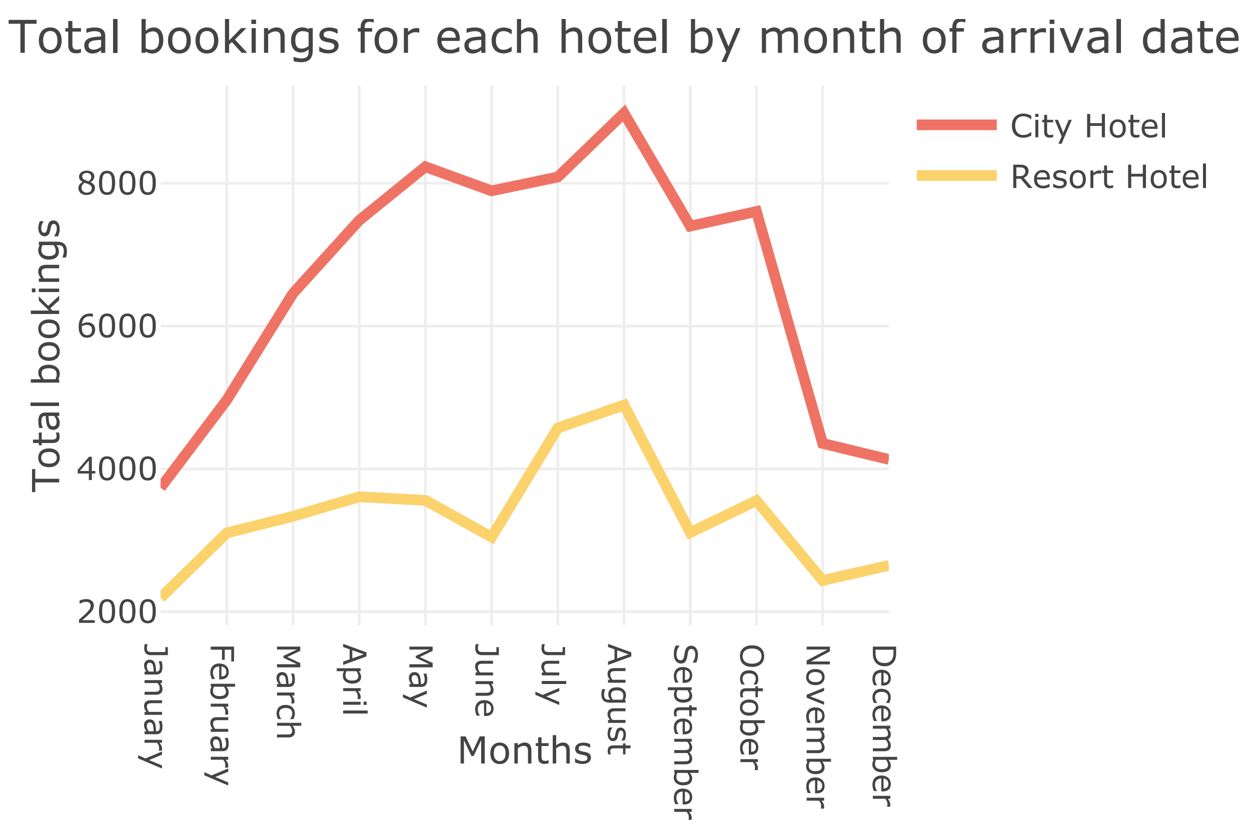
```
                    levels = c("January", "February", "March","April", "May",
                               "June", "July", "August"))

df_guests_2017_Resort$arrival_date_month <- factor(
                    df_guests_2017_Resort$arrival_date_month,
                    levels = c("January", "February", "March","April", "May",
                               "June", "July", "August"))


fig_guests_2017 <- plot_ly()

fig_guests_2017 <- fig_guests_2017 %>% add_lines(data=df_guests_2017_City,
                      x = ~arrival_date_month, y = ~guests, name = 'City Hotel',
                      type = 'scatter', mode = 'lines',
                      line = list(color = 'rgb((239,115,101))', width = 4))

fig_guests_2017 <- fig_guests_2017 %>% add_lines(data=df_guests_2017_Resort,
                      x = ~arrival_date_month, y = ~guests, name = 'Resort Hotel',
                      type = 'scatter', mode = 'lines',
                      line = list(color = 'rgb((251,210,108))', width = 4))

fig_guests_2017 <- fig_guests_2017 %>% layout(
         title = "2017 - Total guests for each hotel by month of arrival date",
         xaxis = list(title = "Months"),
         yaxis = list (title = "Total guests"))

fig_guests_2017
```



We finally compared also the total bookings and the total guests, considering all the observations. We can notice, looking at the following plots, that the behavior that we observed and analyzed above for the 2016 is not evident.

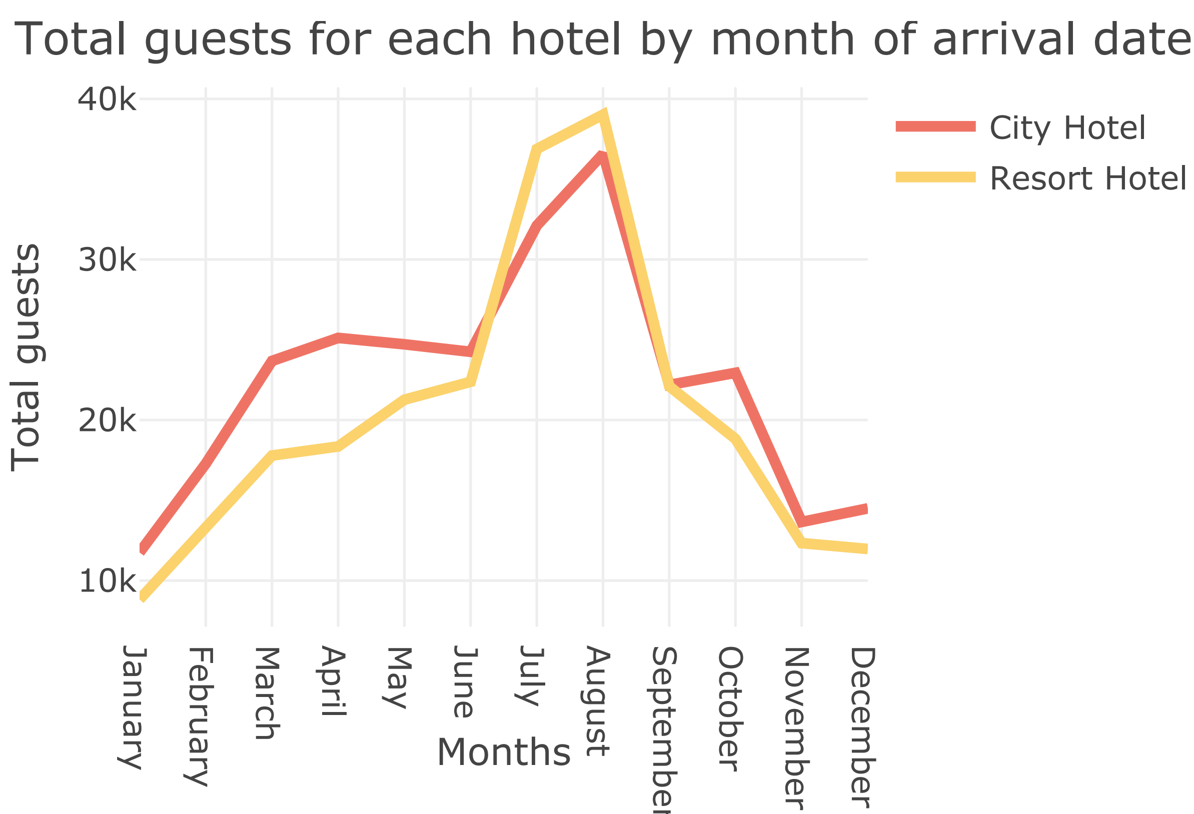
```
# Total bookings for each hotel by month of arrival date

df_months_City <- as.data.frame(
                  hotel_bookings_new[hotel_bookings_new$hotel=='City Hotel',
                  c( "arrival_date_month")])

df_months_Resort <- as.data.frame(
                    hotel_bookings_new[hotel_bookings_new$hotel=='Resort Hotel',
                    c( "arrival_date_month")])

df_months_City <- as.data.frame(
                  lapply(df_months_City, function(x) as.data.frame(table(x))))

df_months_Resort <- as.data.frame(
                    lapply(df_months_Resort, function(x) as.data.frame(table(x))))

colnames(df_months_City) <- c("arrival_date_month_City", "frequency_City" )

colnames(df_months_Resort) <- c("arrival_date_month_Resort", "frequency_Resort")

df_months_City$arrival_date_month_City <- factor(
          df_months_City$arrival_date_month_City,
```

```r
          levels = c("January", "February", "March","April", "May", "June", "July",
                     "August", "September", "October", "November", "December"))

df_months_Resort$arrival_date_month_Resort <- factor(
          df_months_Resort$arrival_date_month_Resort,
          levels = c("January", "February", "March","April", "May", "June", "July",
                     "August", "September", "October", "November", "December"))


fig_months <- plot_ly()

fig_months <- fig_months %>% add_lines(data=df_months_City,
                             x = ~arrival_date_month_City, y = ~frequency_City,
                             name = 'City Hotel', type = 'scatter', mode = 'lines',
                             line = list(color = 'rgb((239,115,101))', width = 4))

fig_months <- fig_months %>% add_lines(data=df_months_Resort,
                             x = ~arrival_date_month_Resort,
                             y = ~frequency_Resort, name = 'Resort Hotel',
                             type = 'scatter', mode = 'lines',
                             line = list(color = 'rgb((251,210,108))', width = 4))

fig_months <- fig_months %>%
layout(title = "Total bookings for each hotel by month of arrival date",
         xaxis = list(title = "Months"),
         yaxis = list (title = "Total bookings"))

fig_months
```




```
# Total guests for each hotel by month

df_City <- as.data.frame(hotel_bookings_new[hotel_bookings_new$is_canceled==0 &
                         hotel_bookings_new$hotel=='City Hotel',
                         c( "arrival_date_month", "total_stays", "adults",
                            "children", "babies")])

df_guests_City <- df_City %>%
  group_by(arrival_date_month) %>%
  summarise(guests = sum(total_stays*(adults+children+babies))) %>%
  ungroup()

df_Resort <- as.data.frame(hotel_bookings_new[hotel_bookings_new$is_canceled==0 &
                                              hotel_bookings_new$hotel=='Resort Hotel',
                                              c( "arrival_date_month", "total_stays",
                                                 "adults", "children", "babies")])

df_guests_Resort <- df_Resort %>%
  group_by(arrival_date_month) %>%
  summarise(guests = sum(total_stays*(adults+children+babies))) %>%
  ungroup()


df_guests_City$arrival_date_month <- factor(df_guests_City$arrival_date_month,
          levels = c("January", "February", "March","April", "May", "June", "July",
                     "August", "September", "October", "November", "December"))
```

```r
df_guests_Resort$arrival_date_month <- factor(df_guests_Resort$arrival_date_month,
          levels = c("January", "February", "March","April", "May", "June", "July",
                     "August", "September", "October", "November", "December"))


fig_guests <- plot_ly()

fig_guests <- fig_guests %>% add_lines(data=df_guests_City, x = ~arrival_date_month,
                                       y = ~guests, name = 'City Hotel',
                                       type = 'scatter', mode = 'lines',
                                       line = list(color = 'rgb((239,115,101))',
                                                   width = 4))

fig_guests <- fig_guests %>% add_lines(data=df_guests_Resort, x = ~arrival_date_month,
                                       y = ~guests, name = 'Resort Hotel',
                                       type = 'scatter', mode = 'lines',
                                       line = list(color = 'rgb((251,210,108))',
                                                   width = 4))

fig_guests <- fig_guests %>% layout(
         title = "Total guests for each hotel by month of arrival date",
         xaxis = list(title = "Months"),
         yaxis = list (title = "Total guests"))

fig_guests
```



Taking a look at the other variables we spent a few time in the analysis of the features *reserved_room_type* and *assigned_room_type* to understand what are the differences among them.

```
# Reserved room type table

table_hotel_reserved <- table(hotel_bookings_new$hotel,
                              hotel_bookings_new$reserved_room_type)
table_hotel_reserved
```

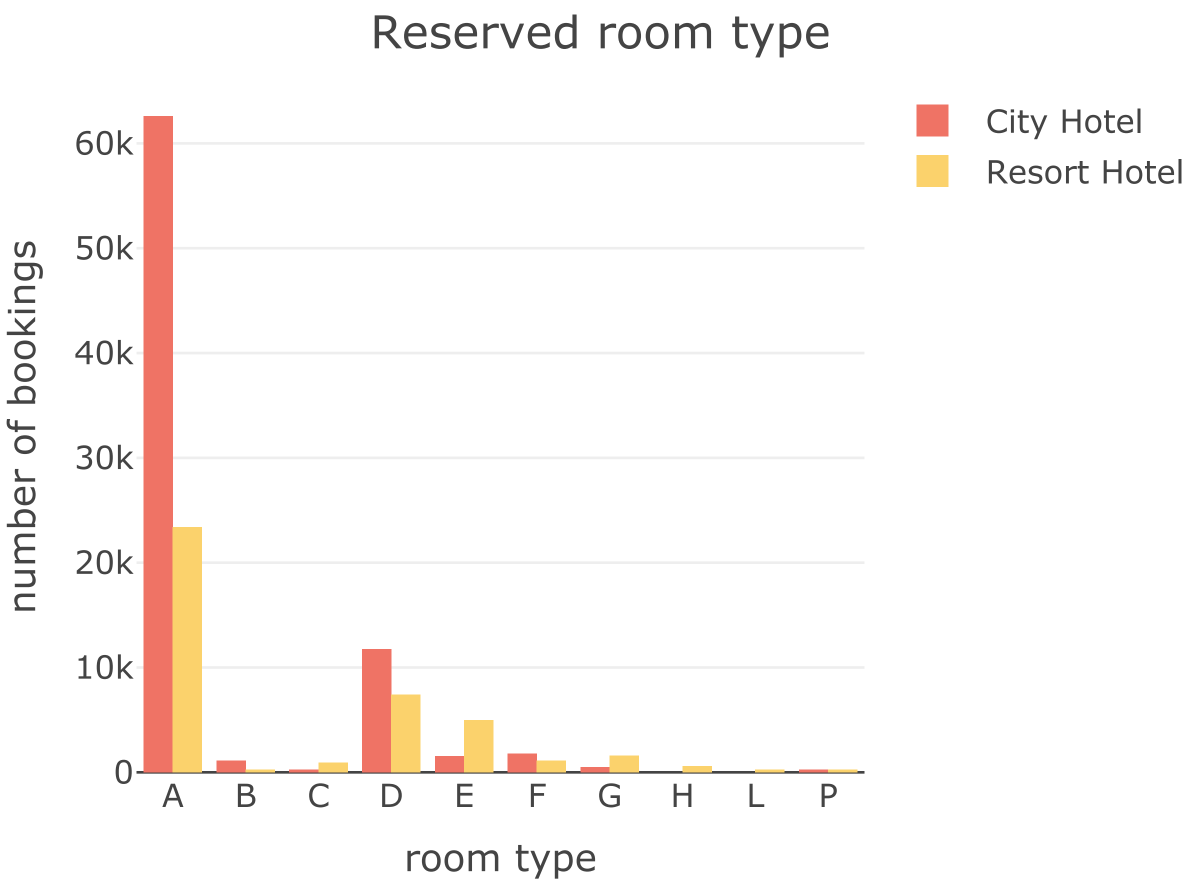
```
##
##                   A     B     C     D     E     F     G     H     L     P
##   City Hotel   62595  1115    14 11768  1553  1791   484     0     0    10
##   Resort Hotel 23399     3   918  7433  4982  1106  1610   601     6     2
```

```
# Assigned room type table

table_hotel_assigned <- table(hotel_bookings_new$hotel,
                              hotel_bookings_new$assigned_room_type)
table_hotel_assigned
```

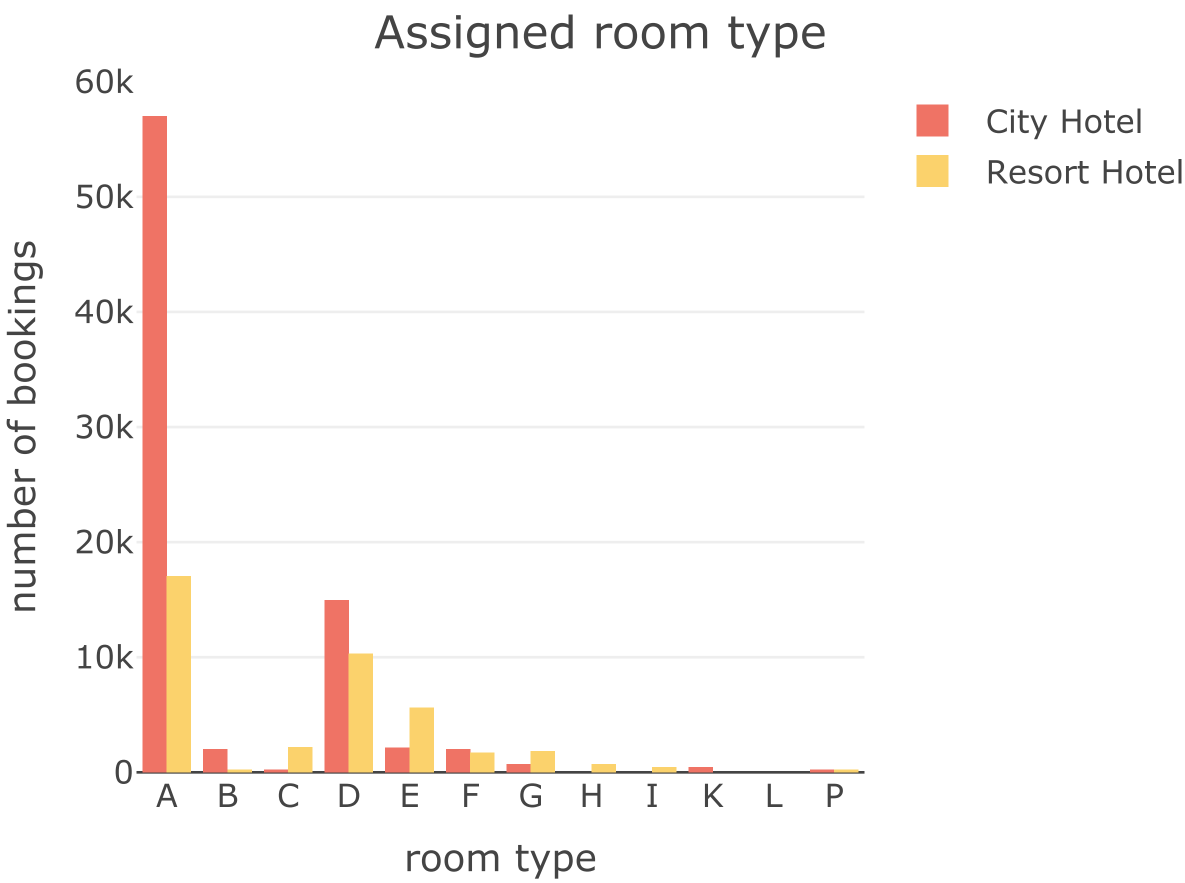
```
##
##                   A     B     C     D     E     F     G     H     I     K
##   City Hotel   57007  2004   161 14983  2168  2018   700     0     0   279
##   Resort Hotel 17046   159  2214 10339  5638  1733  1853   712   363     0
##
##                   L     P
##   City Hotel      0    10
##   Resort Hotel    1     2
```

As we can see from the tables above, there are no city hotel (reserved and assigned) room of type H and L. Furthermore, looking at the assigned room table, we noticed that there are two room types, I for the resort hotel and K for the city hotel, which were not present in the reserved room table.

```
# Reserved room type plot

city_reserved <- table_hotel_reserved[c('City Hotel'),]
resort_reserved <- table_hotel_reserved[c('Resort Hotel'),]
room_type_reserved <- colnames(table_hotel_reserved)

df_reserved <- data.frame(room_type_reserved, city_reserved, resort_reserved)

fig_reserved <- plot_ly(df_reserved, x = ~room_type_reserved, y = ~city_reserved,
                        type = 'bar', name = 'City Hotel',
                        marker = list(color = 'rgb(239,115,101)'))
fig_reserved <- fig_reserved %>% add_trace(y = ~resort_reserved, name = 'Resort Hotel',
                                           marker = list(color='rgb(251,210,108)'))
fig_reserved <- fig_reserved %>% layout(title = "Reserved room type",
                                        xaxis = list(title = 'room type'),
                                        yaxis =list(title = 'number of bookings'),
                                        barmode = 'group')
fig_reserved
```




```r
# Assigned room type plot

city_assigned <- table_hotel_assigned[c('City Hotel'),]
resort_assigned <- table_hotel_assigned[c('Resort Hotel'),]
room_type_assigned <- colnames(table_hotel_assigned)

df_assigned <- data.frame(room_type_assigned, city_assigned, resort_assigned)

fig_assigned <- plot_ly(df_assigned, x = ~room_type_assigned, y = ~city_assigned,
                        type = 'bar', name = 'City Hotel',
                        marker = list(color = 'rgb(239,115,101)'))
fig_assigned <- fig_assigned %>% add_trace(y = ~resort_assigned, name = 'Resort Hotel',
                                           marker = list(color='rgb(251,210,108)'))
fig_assigned <- fig_assigned %>% layout(title = "Assigned room type",
                                        xaxis = list(title = 'room type'),
                                        yaxis =list(title = 'number of bookings'),
                                        barmode = 'group')
fig_assigned
```



We want now to focus our attention on the variable *is_canceled*. The first thing to notice is that it it a binary variable, that can take values 0 (i.e. not canceled) or 1 (i.e. canceled). The following donut plot clearly shows how much the variable is unbalanced, since the number of non-cancelled reservations is larger than the number of cancelled ones (the proportion is about 2:1). This must be highlighted, since it may have an impact on the results of the model we are going to build.

```
table(hotel_bookings_new$is_canceled)
```

```
##
##     0     1
## 75166 44224
```

```
# Percentage of cancellations for hotel type
ggplot(data = hotel_bookings_new,
       aes(
         x = hotel,
         y = prop.table(stat(count)),
         fill = factor(is_canceled),
         label = scales::percent(prop.table(stat(count)))
       )) +
  geom_bar(position = position_dodge()) +
  geom_text(
    stat = "count",
    position = position_dodge(.9),
    vjust = -0.5,
    size = 3
```

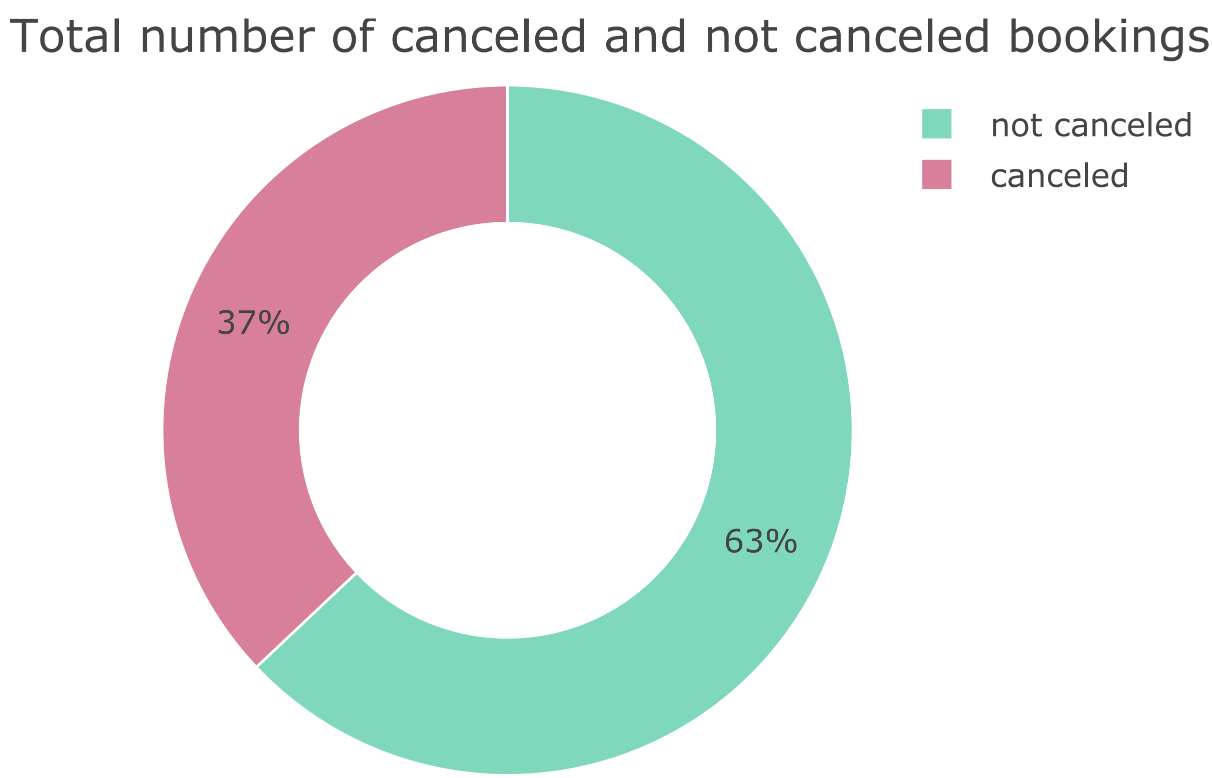
```r
  ) +
  scale_y_continuous(labels = scales::percent) +
  labs(title = "Cancellation Status by Hotel Type",
       x = "Hotel Type",
       y = "Count") +
  theme_classic() +
  scale_fill_manual(values=c("#7fd8be", "#d87f99"),
    name = "Booking Status",
    breaks = c("0", "1"),
    labels = c("Not Cancelled", "Cancelled")
  ) + theme_bw()
```

As stated before out of all the bookings, the majority of requests is for City Hotel (about 65% of the total bookings) and from the plot we can see that the percentage of confirmed status is higher than cancellations in both cases. In particular the ratio between canceled and not canceled is lower in case of Resort, which means that those who book a resort have a lower tendency to cancel their booking.

```r
# Cancellations donut plot
df_canc <- as.data.frame(hotel_bookings_new[, c("is_canceled")])
df_canc <- as.data.frame(lapply(df_canc, function(x) as.data.frame(table(x))))
colnames(df_canc) <- c("is_canceled", "frequency")
df_canc
```

```
##   is_canceled frequency
## 1           0     75166
## 2           1     44224
```

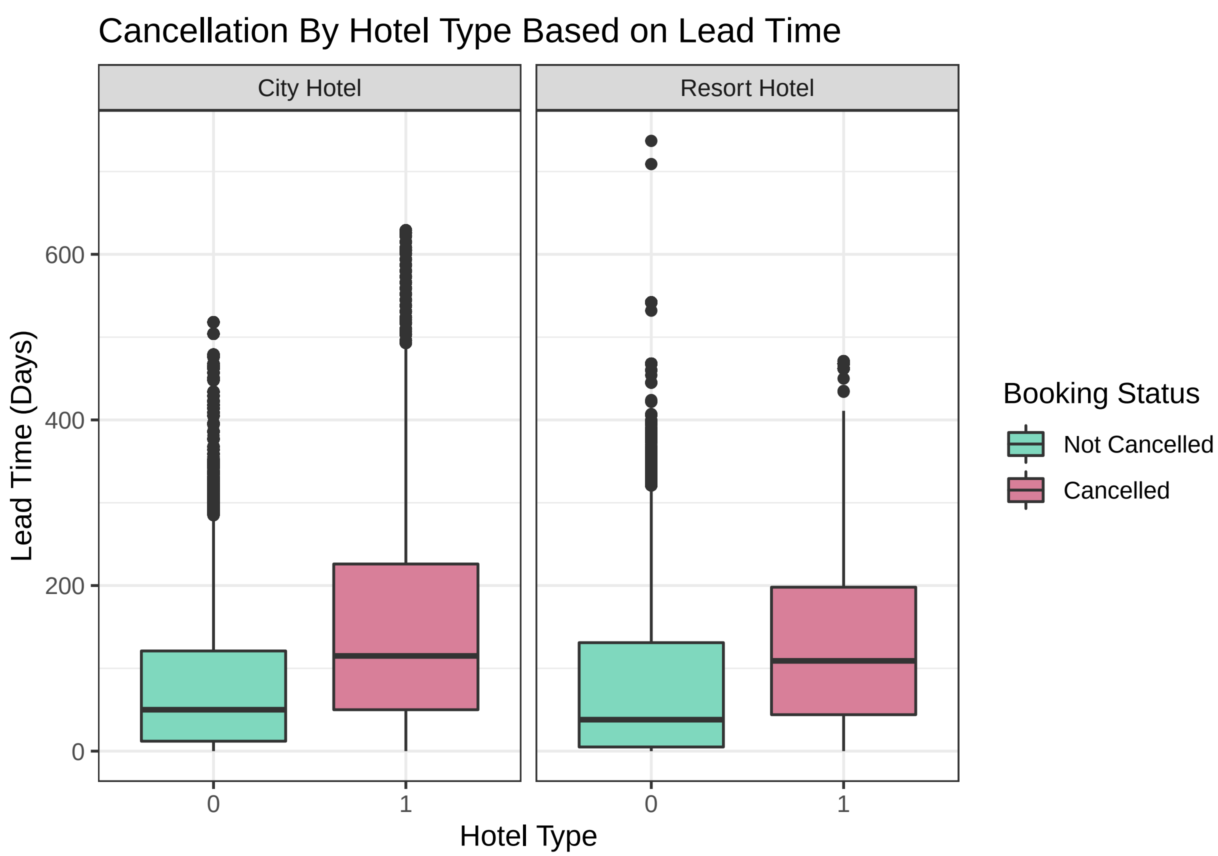
```r
colors_donut_canc <- c('rgb(127,216,190)','rgb(216,127,153)')
fig_canc <- df_canc %>% plot_ly(labels = c('not canceled','canceled' ),
                                values = ~frequency,
                                marker = list(colors = colors_donut_canc,
                                line = list(color = '#FFFFFF', width = 1)))
fig_canc <- fig_canc %>% add_pie(hole = 0.6)
fig_canc <- fig_canc %>% layout(
                        title = "Total number of canceled and not canceled bookings",
                        showlegend = T,
                        xaxis = list(showgrid = FALSE, zeroline = FALSE,
                                     showticklabels = FALSE),
                        yaxis = list(showgrid = FALSE, zeroline = FALSE,
                                     showticklabels = FALSE))
fig_canc
```



A first thing we want to study is the relation between cancellations and lead_time, which is the number of days that elapsed between the day of the booking and the arrival date or the eventual cancellation. The following boxplot shows the distribution of cancelled and not cancelled bookings for each hotel. We see at once that in both cases the median of cancelled is higher than for not cancelled: this means that among customers who do not cancel the booking, the majority prefer to book the room next to the day of arrival, while who books in advance (i.e. when lead time is larger) has a greater tendency to cancel the booking.

```r
# Boxplot to show cancellations based on lead_time
ggplot(data = hotel_bookings_new, aes(x=is_canceled, y=lead_time, fill=is_canceled))+
  geom_boxplot(position = position_dodge() ) +
  labs(
```

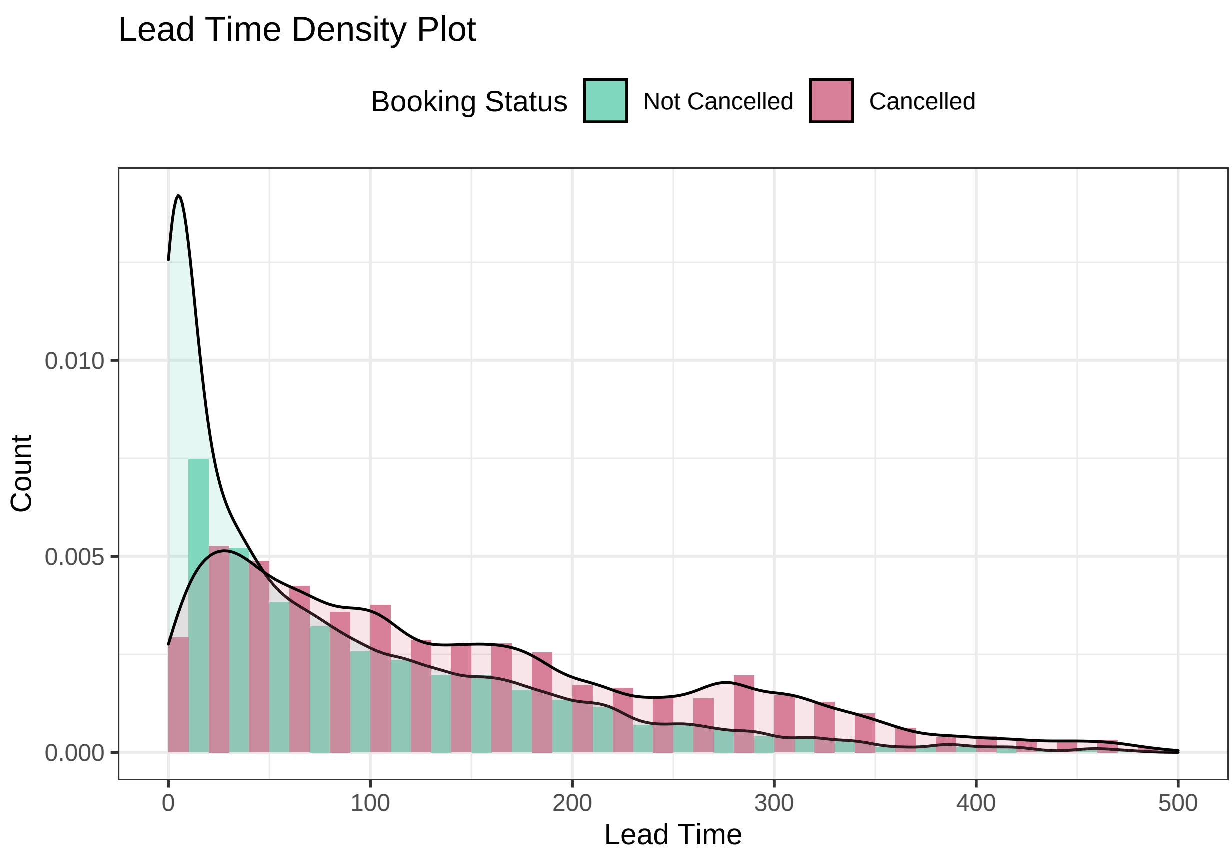
```
    title = "Cancellation By Hotel Type Based on Lead Time",
    x = "Hotel Type",
    y = "Lead Time (Days)"
  ) +
  scale_fill_manual(values=c("#7fd8be", "#d87f99"),
    name = "Booking Status",
    breaks = c("0", "1"),
    labels = c("Not Cancelled","Cancelled" )
  ) + theme_bw() +
  facet_wrap(~hotel)
```



This can be easily seen also by the density plot, where on the x-axis there is the lead_time and on y-axis the correspondent number of bookings for a specific lead_time. The plot shows both cancelled and not cancelled cases. The peak of the curve with respect to the not cancelled is higher e it occurs for a low value of lead_time, typically only 10 days. Both the distributions are clearly skewed, but the cancelled one stay higher than the other also for high values of lead_time, which means that the higher the time elapsed between the reservation and the arrival date the higher the risk of cancellation.

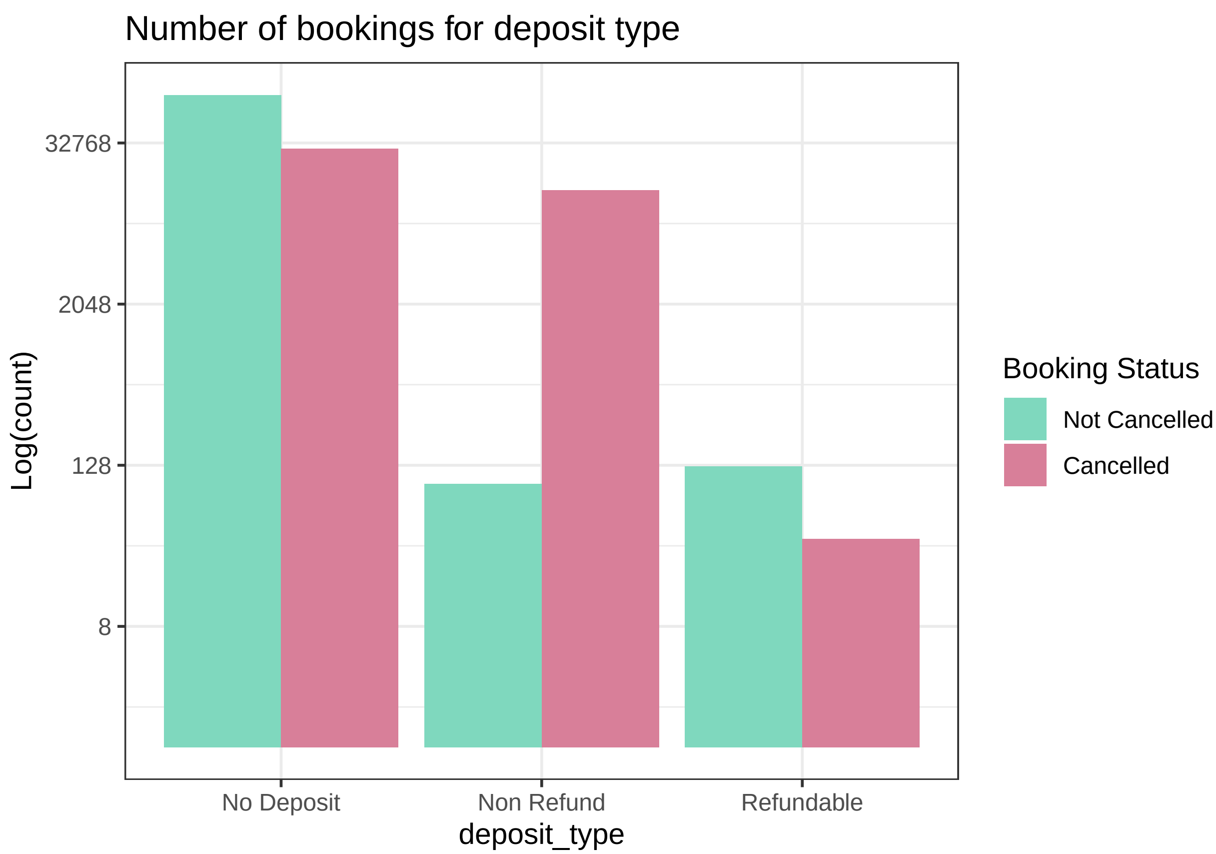
```
ggplot(hotel_bookings_new, aes(x = lead_time, fill = is_canceled)) +
  geom_histogram(aes(y = ..density..), position = position_dodge(), binwidth = 20 ) +
  geom_density(alpha = 0.2) +
  labs(title = "Lead Time Density Plot",
       x = "Lead Time",
       y = "Count") + scale_fill_manual(values=c("#7fd8be", "#d87f99"),
    name = "Booking Status",
    breaks = c("0", "1"),
```

```
    labels = c("Not Cancelled","Cancelled" )) +
  xlim(0,500) +
  theme_bw() + theme(legend.position = "top")
```



The following result is very interesting: the plot shows the number of cancellations for different types of deposit that a customer can made to guarantee the booking. Most of the time no deposit is required, but the main observation here is that if the deposit is not recoverable the number of cancellations becomes way higher than the number of confirmed bookings. In the plot a log scaling was applied on the y-axis to better show the difference among cancellations in the Non Refundable case (without the scaling the number of non cancelled reservations was so small that it could be barely seen).

```
# This plot shows the number of bookings per deposit_type
ggplot(hotel_bookings_new, aes(deposit_type, fill = is_canceled)) +
  geom_bar(stat = "count", position = position_dodge()) +
  labs(title = "Number of bookings for deposit type",
       x = "deposit_type",
       y = "Log(count)") +
  theme_bw() +
         scale_fill_manual(values=c("#7fd8be", "#d87f99"),
    name = "Booking Status",
    breaks = c("0", "1"),
    labels = c("Not Cancelled", "Cancelled") ) + scale_y_continuous(trans='log2')
```



This seems to be a very weird behavior and it raises the question if there is something wrong with the data. The following table shows mean values of the data, with respect to some specific features, grouped by *deposit_type*:

```
hotel_bookings_new %>% group_by(deposit_type)  %>%
  summarise("Lead Time" = mean(lead_time),
            "Prev Canc" = mean(as.numeric(previous_cancellations)),
            "Adults" = mean(as.numeric(adults)),
            "Children" = mean(as.numeric(children)),
            "Babies" = mean(as.numeric(babies)),
            "Special Req" = mean(as.numeric(total_of_special_requests)),
            .groups = 'drop')
```

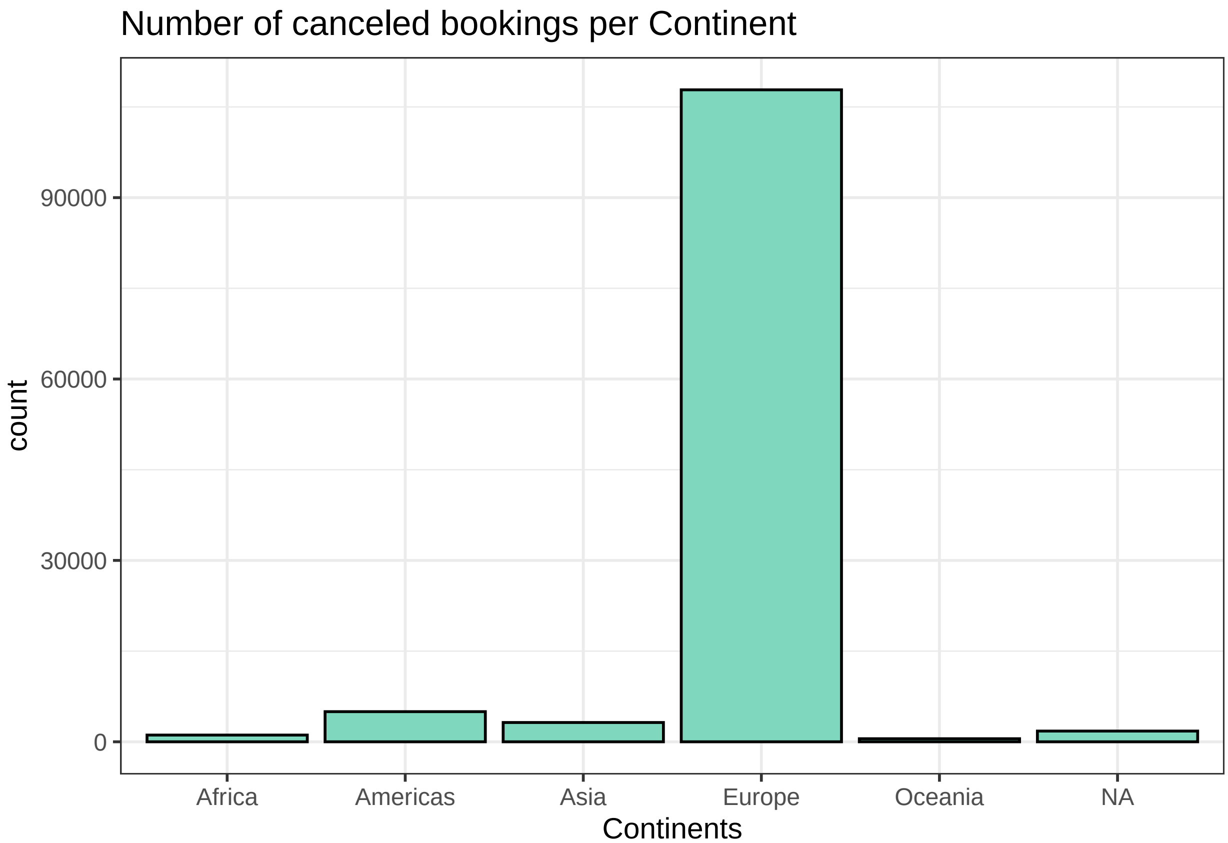
```
## # A tibble: 3 x 7
##   deposit_type `Lead Time` `Prev Canc` Adults Children  Babies `Special Req`
##   <fct>              <dbl>       <dbl>  <dbl>    <dbl>   <dbl>         <dbl>
## 1 No Deposit          88.8      0.0420   1.86 0.118    0.00907       0.651
## 2 Non Refund         213.       0.411    1.81 0.000617 0             0.00178
## 3 Refundable         152.       0        1.91 0.0309   0             0.142
```

We tried this kind of very superficial analysis in order to take a look at the differences between Non Refund and No Deposit, just to check if some specific pattern occurs among people who decide to cancel a non refundable reservation. The comparison shows the following: that Non Refund deposits are characterized by a *lead_time* which is twice as long as for No Deposit; that *previous_cancellations* is 10 times higher, that *children* and *babies* are very rare, that most of people are adults, and that *special_requests* are rare.

Based on these very superficial analysis it seems that people who cancelled non refundable reservations are especially adults, with very few special requests. The booking occurs on average long before the arrival and it's rare that a specific guest returns to the same hotel. Anyway, if it happens, he will repeatedly cancel the reservation, which is quite a strange behavior.

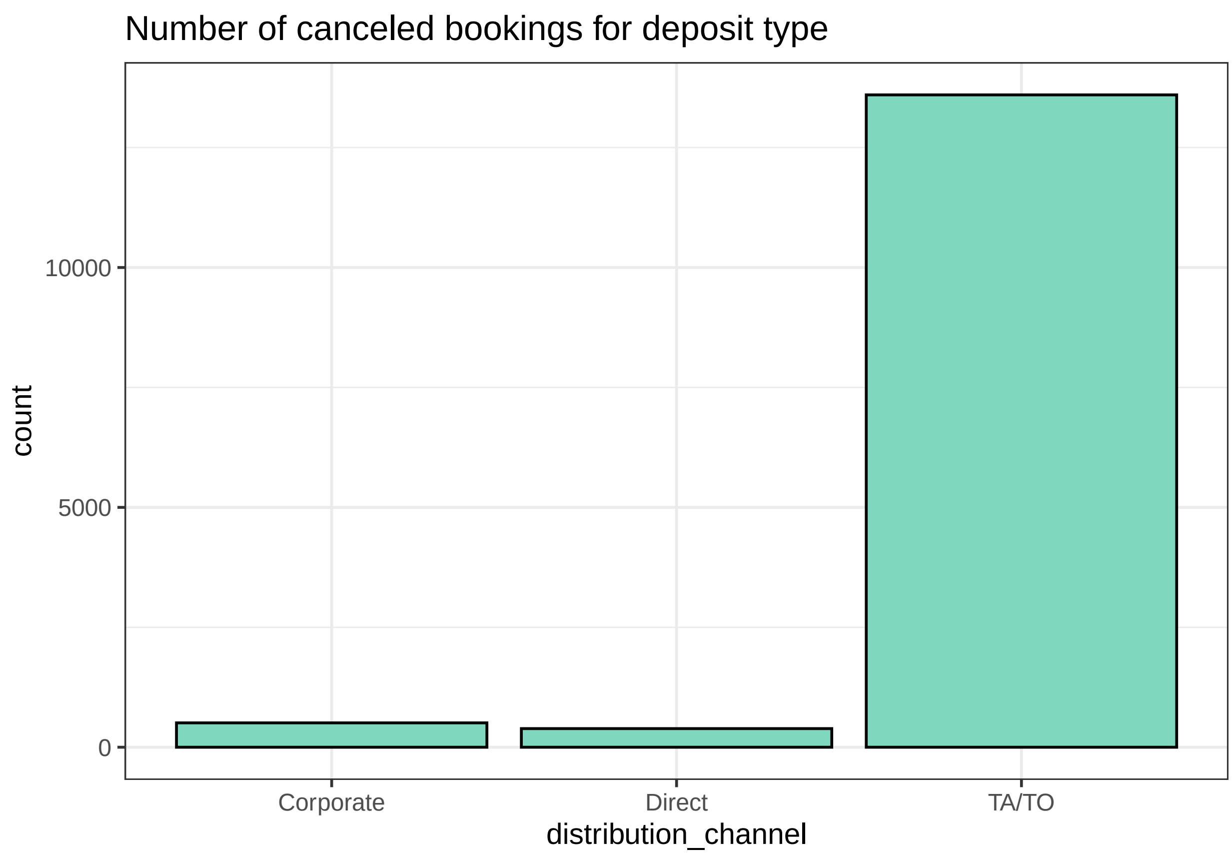
The aim of this analysis was to find some insight to justify the weird situation we discussed above. Unfortunately the data we worked on do not give a good explanation. By doing a lot of research we found the paper *Big Data in Hotel Revenue Management: Exploring Cancellation Drivers to Gain Insights Into Booking Cancellation Behavior*, by Nuno Antonio, Ana Maria De Almeida and Luìs Nunes, who are the same authors of the paper where the hotel booking dataset is presented. They wrote: "As an example, through analysis of the "Nonrefundable" canceled bookings in some Asiatic countries and from certain distribution channels, it is possible to understand why so many "Nonrefundable" bookings are canceled. These bookings are usually made through OTA using false or invalid credit card details. These bookings are issued as support for requests for visas to enter the country (a hotel booking is mandatory for applying for a Portuguese entry visa). After failing to charge the customer's credit card, the hotel identifies these bookings as "fake" and contacts the customer; however, during the time required to verify these bookings, they contribute negatively to demand forecast and demand-management decisions." Therefore we decided to see if this explanation can be confirmed by data at our disposal. We first created a new Dataframe, starting from the original one, that contains only observations regarding customers who canceled a non refundable reservation, then we plotted the number of observation with respect to the continent (information regarding continents is not provided by the original dataframe, so we had to add it thanks to the r function "countryside" provided by the homonymous library).

```
dftemp1 <- data.frame(hotel_bookings_new)
dftemp1$continent <- countrycode(hotel_bookings_new$country, "iso3c", "continent")

dftemp2 <- dftemp1[(dftemp1$deposit_type=="Non Refund" & dftemp1$is_canceled==1),]
ggplot(data = dftemp1,
       aes(
         x = continent,
         fill = factor(continent)
       )) + labs(title = "Number of canceled bookings per Continent",
       x = "Continents", y = "count") +
  geom_bar(col ="black",fill="#7fd8be") + theme_bw()
```



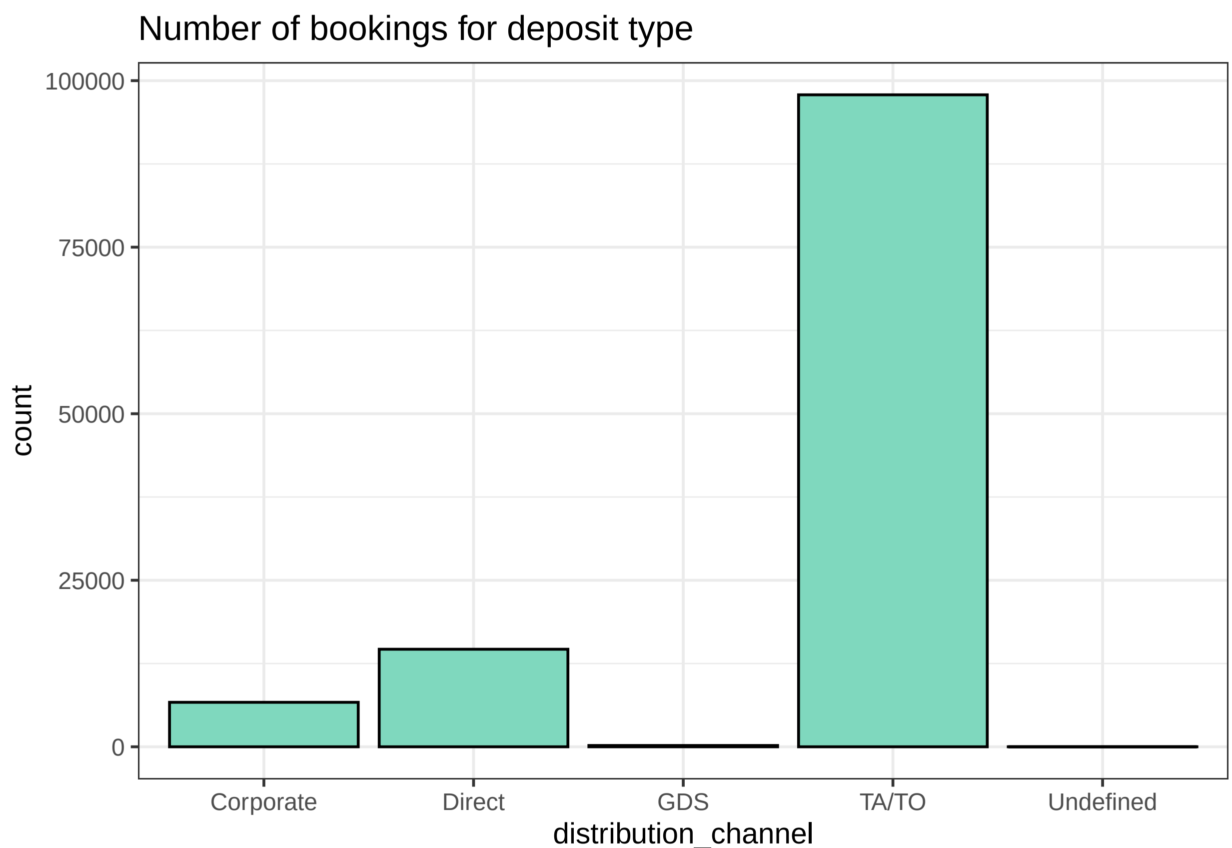
From the plot we can see that the majority of guests with the above characteristics come from Europe, which seems in contradiction with what the authors of the paper wrote. Pre also plotted a second graph to see what is the favorite distribution channel among the guests.

```
ggplot(dftemp2, aes(distribution_channel)) +
  geom_bar(stat = "count", position = position_dodge(), col ="black",fill="#7fd8be") +
  labs(title = "Number of canceled bookings for deposit type",
       x = "distribution_channel",
       y = "count") +
  theme_bw()
```



It turned out that Travel Agent/Tour Operator is the favorite distribution channel, which seems coherent with what the authors suggested. However it is a not conclusive result, since TA/TO remains the favorite distribution channel also on the whole dataset, as we can see from the plot below:

```
ggplot(hotel_bookings_new, aes(distribution_channel)) +
  geom_bar(stat = "count", position = position_dodge(), col ="black",fill="#7fd8be") +
  labs(title = "Number of bookings for deposit type",
       x = "distribution_channel",
       y = "count") +
  theme_bw()
```



In the end, data at our disposal are not sufficient to confirm what the authors of the paper argue.

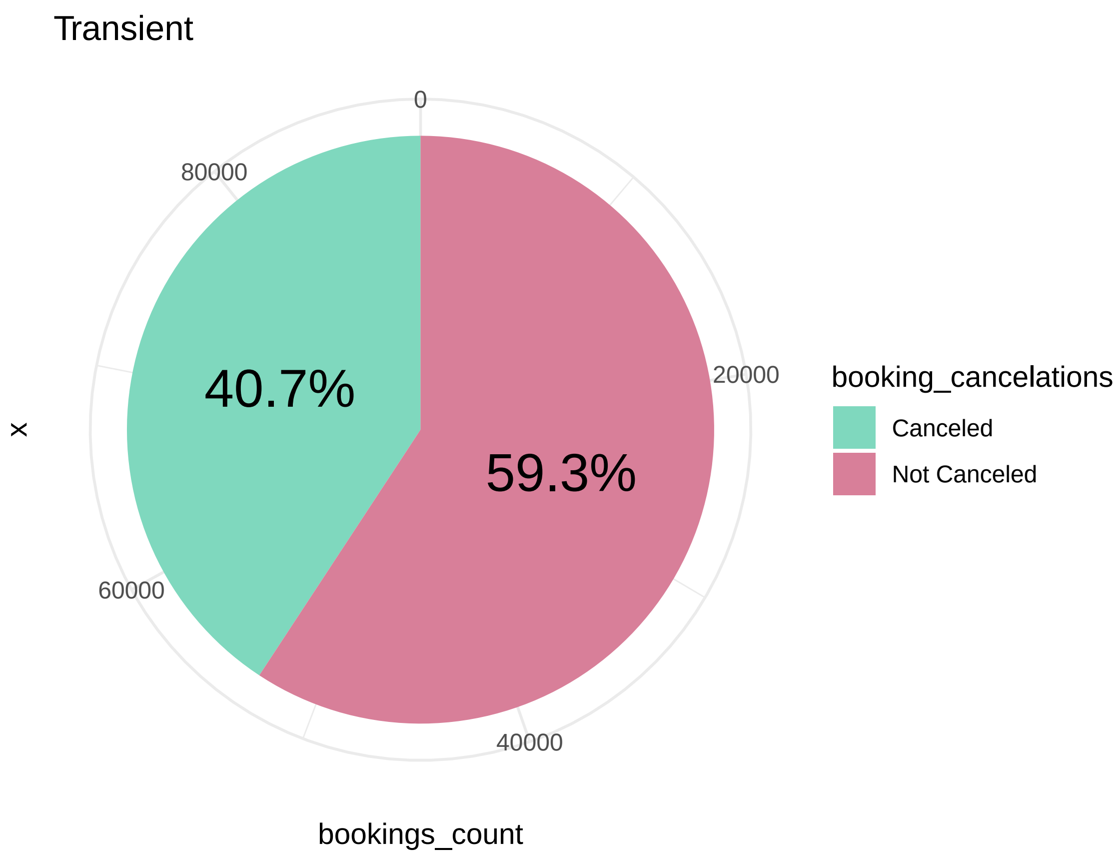
Another important thing to assess is the eventual relationship between cancellations and customer type. This analysis aims to answer to the questions: how many people, among the ones who cancelled the reservation, belong to the different customer classes? Is there a class of customers where the rate of cancellation is higher than the other? The following marplot shows the total number of booking cancellations for customer type.

```
#1. Plot 1 by total bookings
dfc.cancelation <- hotel_bookings_new %>%
  mutate(
    booking_cancelations = ifelse(is_canceled == 0, "Not Canceled", "Canceled")
  )

#2. Plotting
dfc.cancelation %>%
  group_by(booking_cancelations, customer_type) %>%
  summarize(bookings_count = n()) %>%
  ggplot(aes(x = reorder(customer_type, -bookings_count),y=bookings_count,
             fill = booking_cancelations)) +
  geom_bar(position = "dodge", stat = "identity")  +
  labs(title = "Canceled vs Non Canceled Bookings by Customer Type",
       x = "Customer Type", y = "count") +
  scale_fill_manual(values=c("#7fd8be", "#d87f99")) + theme_bw()
```

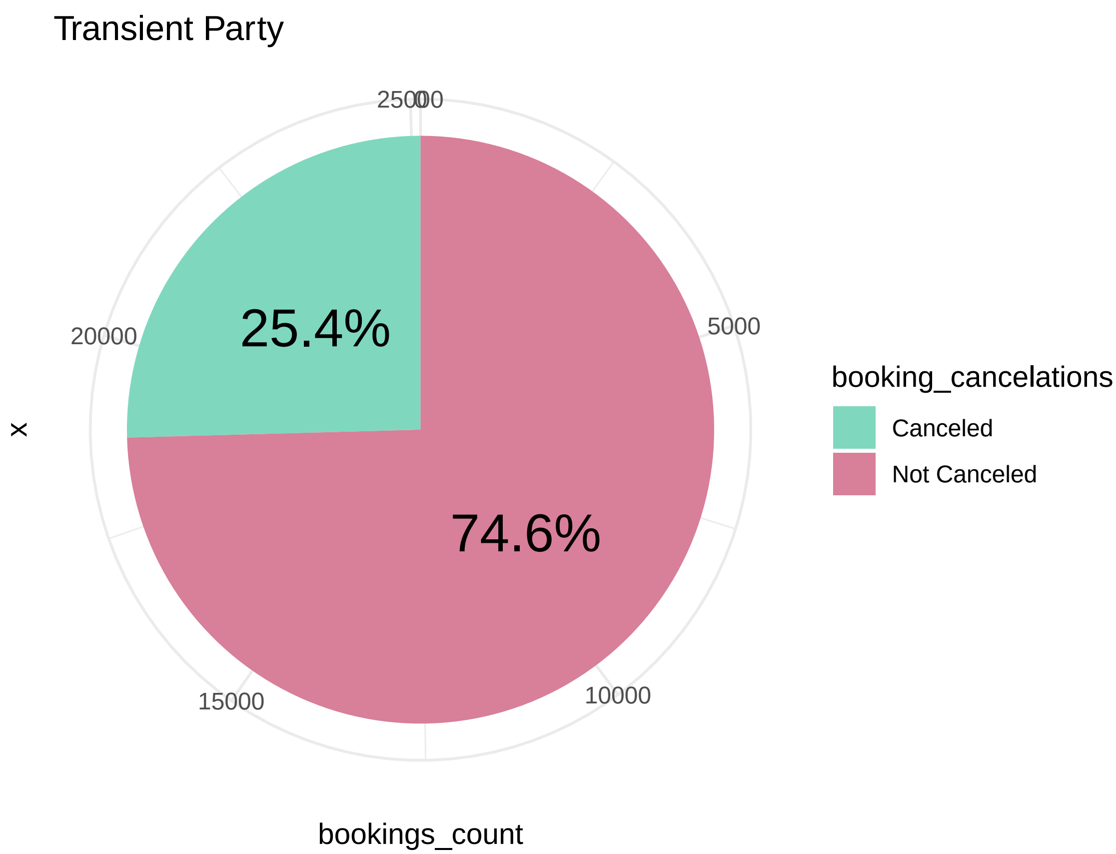
```
## `summarise()` has grouped output by 'booking_cancelations'. You can override
## using the `.groups` argument.
```

The graph shows that, as we could expect, most of the customers are Transient, which means that they are not associated to a group or other customers, but they are on they own. Transient-Party, maybe families that travel together are quite frequent, while travelers by contract or organized groups are more less frequent, but they are intersting to analyze. Therefore, in order to better show up the difference between canceled and not canceled in each of these classes, we decided to do the following graphs:

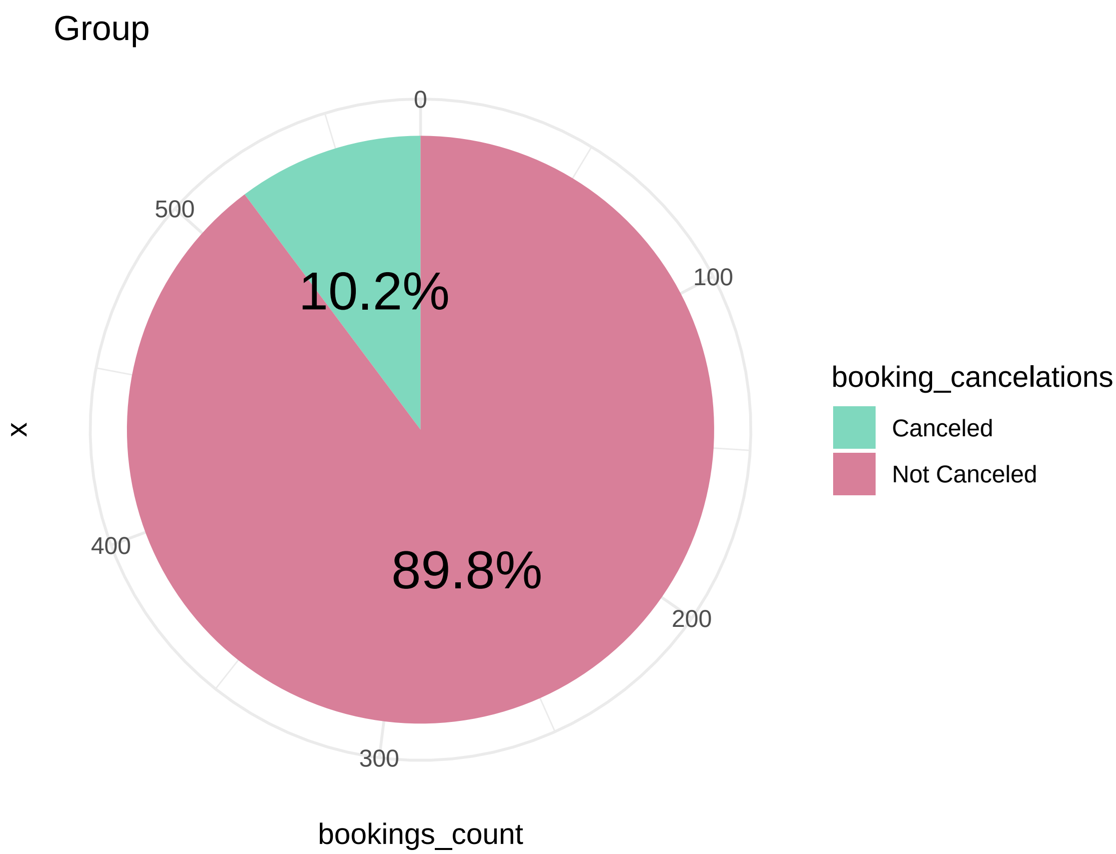
```
#1. Plot 1 by Transient customers
dfc.cancelation %>%
  filter(customer_type == "Transient") %>%
  group_by(booking_cancelations) %>%
  summarize(bookings_count = n()) %>%
  mutate(Percent = round(bookings_count / sum(bookings_count), 3)) %>%
  ggplot(aes(x = "", y = bookings_count, fill = booking_cancelations)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar("y", start=0) +
  geom_text(aes(label = paste0(Percent*100, "%")), position = position_stack(vjust=.5),
              size = 7)  +
  labs(title = "Transient") +
  theme_minimal() +
  scale_fill_manual(values=c("#7fd8be", "#d87f99"))
```



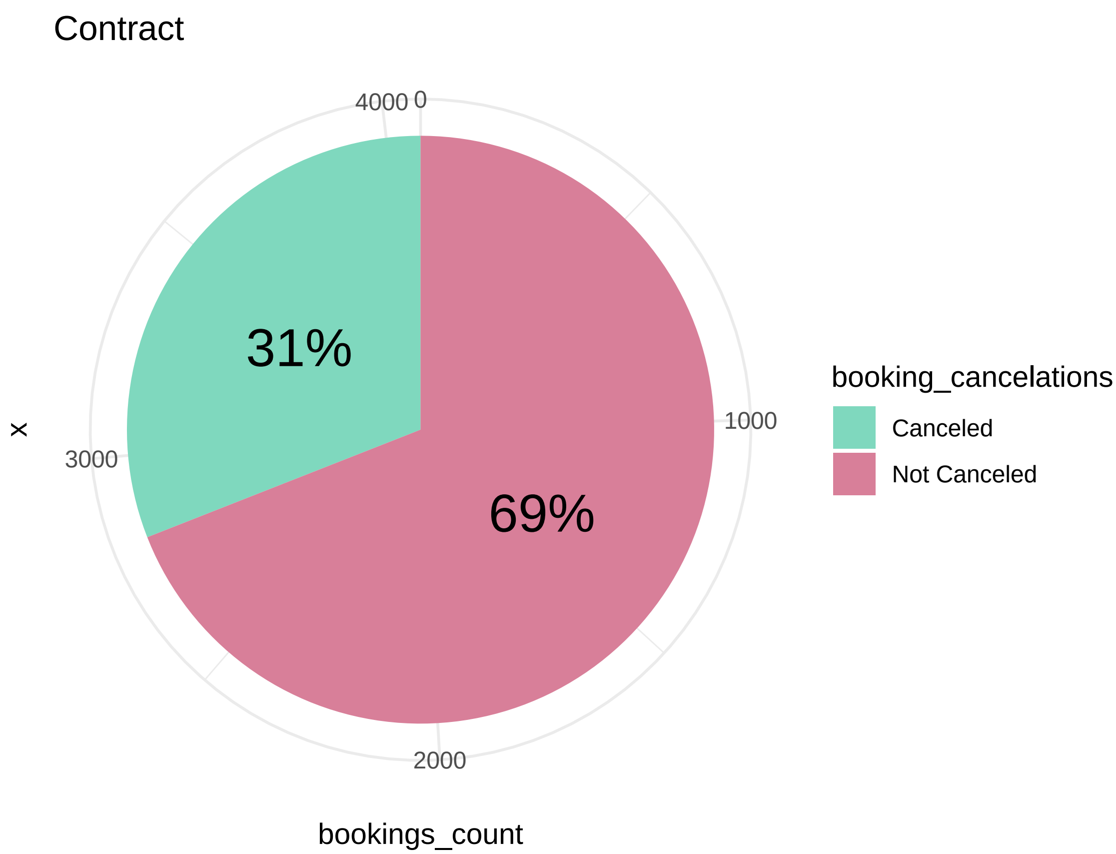

```
#2. Plot 2 by Transient Party customers
dfc.cancelation %>%
  filter(customer_type == "Transient-Party") %>%
  group_by(booking_cancelations) %>%
  summarize(bookings_count = n()) %>%
  mutate(Percent = round(bookings_count / sum(bookings_count), 3)) %>%
  ggplot(aes(x = "", y = bookings_count, fill = booking_cancelations)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar("y", start=0) +
  geom_text(aes(label = paste0(Percent*100, "%")), position = position_stack(vjust=.5),
              size = 7)  +
  labs(title = "Transient Party") +
  theme_minimal() +
  scale_fill_manual(values=c("#7fd8be", "#d87f99"))
```




```
#3. Plot 3 by Group customers
dfc.cancelation %>%
  filter(customer_type == "Group") %>%
  group_by(booking_cancelations) %>%
  summarize(bookings_count = n()) %>%
  mutate(Percent = round(bookings_count / sum(bookings_count), 3)) %>%
  ggplot(aes(x = "", y = bookings_count, fill = booking_cancelations)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar("y", start=0) +
  geom_text(aes(label = paste0(Percent*100, "%")), position = position_stack(vjust=.5),
              size = 7)  +
  labs(title = "Group") +
  theme_minimal() +
  scale_fill_manual(values=c("#7fd8be", "#d87f99"))
```



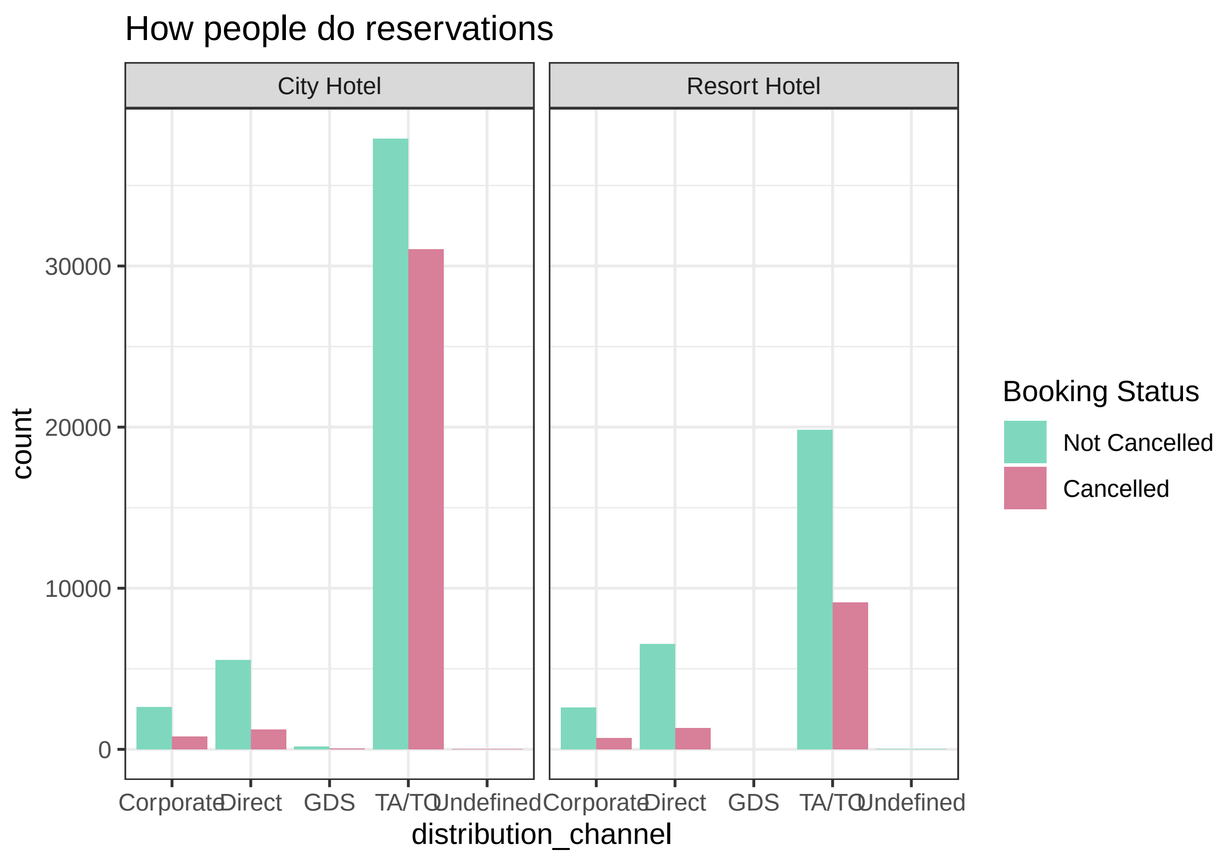

```
#4. Plot 4 by Contrcat customers
dfc.cancelation %>%
  filter(customer_type == "Contract") %>%
  group_by(booking_cancelations) %>%
  summarize(bookings_count = n()) %>%
  mutate(Percent = round(bookings_count / sum(bookings_count), 3)) %>%
  ggplot(aes(x = "", y = bookings_count, fill = booking_cancelations)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar("y", start=0) +
  geom_text(aes(label = paste0(Percent*100, "%")), position = position_stack(vjust=.5),
              size = 7)  +
  labs(title = "Contract") +
  theme_minimal() +
  scale_fill_manual(values=c("#7fd8be", "#d87f99"))
```



As we could expect the proportion of cancelations in groups is very low, since when people travel in group is unusual that the booking is canceled. We can think of organized travels to have an insight of this. Other cases are also not surprising and suggest that if people travel in group or with other people is less probable that they cancel the reservation, while if they are alone they can change their mind at the last minute. Since the number of groups or Travel-party is way lower than that of Travel class, it's not possible for a hotel to focus its efforts only on the two classes above, even if form the point of view of cancellations they are more reliable.

In conclusion we can show the favorite distribution channels among City Hotel and Resort. It turns out that both of them have TA/TO as main distribution channel. Also the behavior among other distribution channels seems to be the same, given the fact that we have less data for the Resort.

```
  # how people do reservations (distribution channel) and if the cancel it
ggplot(data = hotel_bookings_new, aes(distribution_channel,fill=is_canceled))+
  geom_bar(stat = "count", position = position_dodge()) +
  labs(
    title = "How people do reservations"
  ) +
  theme(axis.text.x=element_text(angle = 40 )) +
  scale_fill_manual(values=c("#7fd8be", "#d87f99"),
    name = "Booking Status",
    breaks = c("0", "1"),
    labels = c("Not Cancelled", "Cancelled")) +
  facet_wrap(~hotel) + theme_bw()
```



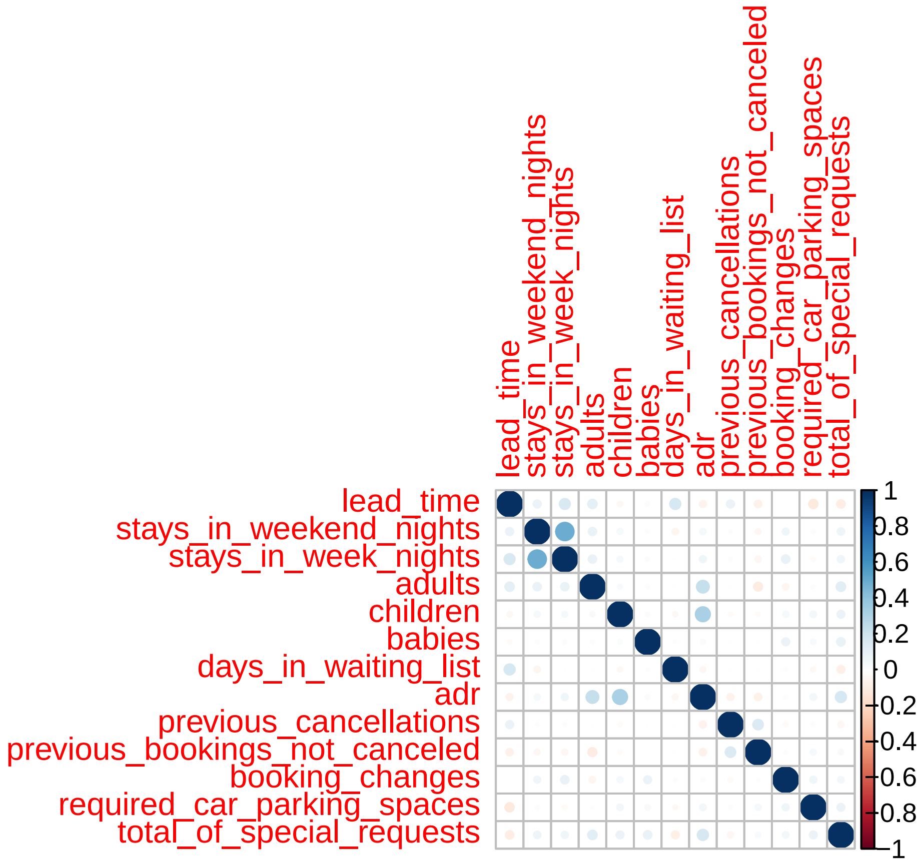
Measures of association between variables

Most of the variables in our dataset are categorical. Therefore, we decided to differentiate the association analysis between numerical variables and between categorical variables.

Association between numerical variables - Correlation

Usually, when we want to understand the relationship between two variables we immediately think of correlation. We know, however, that this is used to refer only to a linear relationship between two variables, and so we have to consider only numerical variables. Therefore, we made the following correlation matrix with all the numerical variables.

```
# Correlation matrix

hotel_numerical <- as.data.frame(hotel_bookings_new[,c("lead_time",
"stays_in_weekend_nights", "stays_in_week_nights", "adults", "children",
"babies", "days_in_waiting_list", "adr", "previous_cancellations",
"previous_bookings_not_canceled", "booking_changes",
"required_car_parking_spaces", "total_of_special_requests")])

cor_mat <- cor(hotel_numerical)

corrplot(cor_mat)
```



As we can see from the corrplot, the numerical variables are not correlated.

Association between categorical variables - Chi-square

Since there is not really a best way to describe the association between two different categorical variables, we decided to focus on associations between the variable *is_canceled* and the other categorical variables. In general, the most used method in this case is the Chi-square, and in particular the Cramer's V, which is a normalized version of the Chi-square statistics.

We start from the couple *is_canceled/market_segment*:

```
# Is_canceled - market segment

canceled_market <-table(
                  hotel_bookings_new$is_canceled, hotel_bookings_new$market_segment)
canceled_market
```

```
##
##     Aviation Complementary Corporate Direct Groups Offline TA/TO Online TA
##   0      185           646      4303  10672   7714         15908     35738
##   1       52            97       992   1934  12097          8311     20739
##
##     Undefined
##   0         0
##   1         2
```

```r
# Chi-square is_canceled - market segment

cs_market <-chisq.test(canceled_market, correct = FALSE)
cs_market
```

```
##
##  Pearson's Chi-squared test
##
## data:  canceled_market
## X-squared = 8497.2, df = 7, p-value < 2.2e-16
```

```r
# Cramer's V is_canceled - market segment
cramersv(cs_market)
```

```
## [1] 0.2667808
```

A second case is that of *is_canceled/deposite_type*:

```r
# Is_canceled - deposite_type

canceled_deposit <-table(
                   hotel_bookings_new$is_canceled, hotel_bookings_new$deposit_type)
canceled_deposit
```

```
##
##      No Deposit Non Refund Refundable
##   0       74947         93        126
##   1       29694      14494         36
```

```r
# Chi-square is_canceled - deposite_type

cs_deposit <-chisq.test(canceled_deposit, correct = FALSE)
cs_deposit
```

```
##
##  Pearson's Chi-squared test
##
## data:  canceled_deposit
## X-squared = 27677, df = 2, p-value < 2.2e-16
```

```r
# Cramer's V is_canceled - market segment
cramersv(cs_deposit)
```

```
## [1] 0.4814798
```

Furthermore a Chi-square test is conducted on the variable *arrival_date_month*, because it is a multilevel categorical variable that shows a large number of levels (12) with respect to the others. It is he only time variable we will use in the definition of the predictive model (since it is in our view the most relevant one according to the aim of this project, while informations about days of arrival are too specific and that about year of arrival too generic). For these reasons particular attention has been paid to it.

```r
# Is_canceled - arrival_date_month

canceled_month <-table(
                  hotel_bookings_new$is_canceled, hotel_bookings_new$arrival_date_month)
canceled_month
```

```
##
##     April August December February January July June March  May November
##   0  6565   8638     4409     5372    4122 7919 6404  6645 7114     4672
##   1  4524   5239     2371     2696    1807 4742 4535  3149 4677     2122
##
##     October September
##   0    6914      6392
##   1    4246      4116
```

```r
# Chi-square is_canceled - arrival_date_month

cs_month <-chisq.test(canceled_month, correct = FALSE)
cs_month
```

```
##
##  Pearson's Chi-squared test
##
## data:  canceled_month
## X-squared = 588.69, df = 11, p-value < 2.2e-16
```

```r
# Cramer's V is_canceled - arrival_date_month
cramersv(cs_month)
```

```
## [1] 0.07021987
```

In all these three cases considered above, the value is really close to zero, which means that our variables are very unlikely to be completely un-associated in some population. However, this does not mean the variables are strongly associated; a weak association in a large sample size may also result in a p-value close to zero. Furthermore, we know that when Cramer's V is 0, it indicates no association between the two variables, while Cramer's V equal to 1 means a strong association; in the intermediate cases it is difficult to interpret, because the value also depends on the size of table and many other things. For these reasons, since in the particular case of 2x2 tables there exist other association measures, we considered them.

Association between binary categorical variables - Relative risk, Odds ratio and Yule's Q

As already told, if we want to look at the association between two binary categorical variable we can use other measures, such as relative risk, Odds ratio and Yule's Q. We analyzed the following two couples of variables:

1. *is_canceled* and *hotel* :

```r
# Matrix is_canceled and hotel

canceled_hotel_matrix <- as.matrix(table(hotel_bookings_new$is_canceled,
hotel_bookings_new$hotel))
canceled_hotel_matrix
```

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

```r
# Not canceled (is_canceled == 0)

n1h <- 46228+28938
pnch.hat <- 46228/n1h


# Canceled (is_canceled == 1)

n2h <- 33102+11122
pch.hat <- 33102/n2h
```

```r
# Relative risk

pnch.hat / pch.hat
```

```
## [1] 0.8216511
```

```r
# Odds ratio

odds_ratio_h <- (46228 * 11122) / (28938*33102)
odds_ratio_h
```

```
## [1] 0.5367416
```

```r
# Yule's Q

Qh <- (odds_ratio_h-1) / (odds_ratio_h+1)
Qh
```

```
## [1] -0.301455
```

2. *is_canceled* and *is_repeated_guest* :

```r
# Matrix is_canceled and is_repeated_guest

canceled_guest_matrix <- as.matrix(table(
                          hotel_bookings_new$is_canceled,
                          hotel_bookings_new$is_repeated_guest))
rownames(canceled_guest_matrix) <- c("Not canceled", "Canceled")
colnames(canceled_guest_matrix) <- c("No Repeated guest", "Repeated guest")
canceled_guest_matrix
```

```
##
##                  No Repeated guest Repeated guest
##   Not canceled               71908           3258
##   Canceled                   43672            552
```

```r
# Not canceled (is_canceled == 0)

n1g <- 71908+3258
pncg.hat <- 71908/n1g


# Canceled (is_canceled == 1)

n2g <- 43672+552
pcg.hat <- 43672/n2g
```

```r
# Relative risk

pncg.hat / pcg.hat
```

```
## [1] 0.9687478
```

```r
# Odds ratio

odds_ratio_g <- (71908 * 552) / (3258*43672)
odds_ratio_g
```

```
## [1] 0.278973
```

```r
# Yule's Q

Qg <- (odds_ratio_g-1) / (odds_ratio_g+1)
Qg
```

```
## [1] -0.5637547
```

We know that relative risk equal to 1 means independence between the two variables; in our cases we obtained 0.8216511 for the is_canceled - hotel couple and 0.9687478 for is_canceled - is_repeated_guest. We then calculated the Odds ratio, for which we know that it is equal to 1 iff the variables are independent and finally we considered the Yule's Q, which is a way to rewrite the Odds ratio, in order to obtain values between -1 and 1, which makes it the most similar measure to the correlation one.

Models

Starting from the informations provided by the initial dataset, a first logistic regression model has been implemented, after removing variables with too many missing values (*agent* and *company*, as stated before). We already pointed out in the previous section that some other variables tend to be redundant; this is the case of *market_segment* and *reservation_status*: for this reason they are discarded from the further analysis. All the remaining variables are included in the initial model, which is called the *complete model*. At this point it's important to emphasize that a variable has been added artificially to the dataset: it is *total_stays*, which is the sum of two variables in the dataset, *stays_in_weekend_nights* and *stays_in_week_nights* and it represents the total duration of the client's stay.

Since most of the variables we are dealing with are categorical, the function *lm* (i.e. linear model) provided by *r* had to be replaced with the more general *glm* (i.e. generalized linear model). As argument, the hyper-parameter *family* was set to "binomial", because the response variable *is_canceled* is binary (values: 0,1).

```
complete_model<-glm(is_canceled~hotel+lead_time+reservation_status_date+arrival_date_month+
total_stays+adults+children+babies+meal+distribution_channel+is_repeated_guest+adr+
previous_cancellations+previous_bookings_not_canceled+booking_changes+deposit_type+
days_in_waiting_list+customer_type+required_car_parking_spaces+total_of_special_requests,
data = hotel_bookings_new,
family="binomial")
summary(complete_model)
```

```
##
## Call:
## glm(formula = is_canceled ~ hotel + lead_time + reservation_status_date +
##     arrival_date_month + total_stays + adults + children + babies +
##     meal + distribution_channel + is_repeated_guest + adr + previous_cancellations +
##     previous_bookings_not_canceled + booking_changes + deposit_type +
##     days_in_waiting_list + customer_type + required_car_parking_spaces +
##     total_of_special_requests, family = "binomial", data = hotel_bookings_new)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -0.7684  -0.4286   0.2310   5.3399
##
## Coefficients:
##                                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      2.060e+01  6.998e-01  29.435  < 2e-16 ***
## hotelResort Hotel                1.269e-01  1.878e-02   6.757 1.40e-11 ***
## lead_time                        5.048e-03  1.010e-04  49.988  < 2e-16 ***
## reservation_status_date         -1.409e-03  4.139e-05 -34.034  < 2e-16 ***
## arrival_date_monthAugust        -3.768e-01  3.403e-02 -11.073  < 2e-16 ***
## arrival_date_monthDecember       1.043e-01  4.123e-02   2.529  0.01143 *
## arrival_date_monthFebruary       2.051e-01  3.907e-02   5.249 1.53e-07 ***
## arrival_date_monthJanuary        1.159e-01  4.453e-02   2.603  0.00924 **
## arrival_date_monthJuly          -4.162e-01  3.400e-02 -12.241  < 2e-16 ***
## arrival_date_monthJune          -2.141e-01  3.536e-02  -6.054 1.41e-09 ***
## arrival_date_monthMarch         -5.742e-02  3.691e-02  -1.555  0.11984
## arrival_date_monthMay           -1.489e-01  3.434e-02  -4.335 1.46e-05 ***
## arrival_date_monthNovember      -3.994e-02  4.217e-02  -0.947  0.34355
## arrival_date_monthOctober       -2.936e-01  3.657e-02  -8.028 9.90e-16 ***
## arrival_date_monthSeptember     -6.451e-01  3.839e-02 -16.804  < 2e-16 ***
## total_stays                      3.729e-02  3.148e-03  11.847  < 2e-16 ***
## adults                           6.124e-02  1.358e-02   4.508 6.55e-06 ***
## children                         4.762e-02  1.908e-02   2.495  0.01259 *
## babies                           4.958e-02  8.315e-02   0.596  0.55098
## mealFB                           1.085e-01  1.072e-01   1.012  0.31169
## mealHB                          -4.705e-01  2.619e-02 -17.967  < 2e-16 ***
## mealSC                           5.153e-01  2.484e-02  20.745  < 2e-16 ***
## mealUndefined                   -9.353e-01  9.993e-02  -9.360  < 2e-16 ***
## distribution_channelDirect      -6.186e-01  4.822e-02 -12.828  < 2e-16 ***
```

```
## distribution_channelGDS        -7.997e-01  1.916e-01  -4.173 3.00e-05 ***
## distribution_channelTA/TO       2.172e-01  4.244e-02   5.117 3.10e-07 ***
## distribution_channelUndefined   1.532e+02  3.001e+07   0.000  1.00000
## is_repeated_guest1             -7.580e-01  8.476e-02  -8.944  < 2e-16 ***
## adr                             1.048e-02  2.282e-04  45.924  < 2e-16 ***
## previous_cancellations          2.572e+00  6.144e-02  41.865  < 2e-16 ***
## previous_bookings_not_canceled -4.201e-01  2.386e-02 -17.608  < 2e-16 ***
## booking_changes                -3.595e-01  1.525e-02 -23.571  < 2e-16 ***
## deposit_typeNon Refund          4.958e+00  1.109e-01  44.715  < 2e-16 ***
## deposit_typeRefundable          3.747e-01  2.035e-01   1.841  0.06555 .
## days_in_waiting_list           -2.278e-03  4.879e-04  -4.669 3.02e-06 ***
## customer_typeGroup              8.647e-02  1.658e-01   0.521  0.60202
## customer_typeTransient          1.361e+00  5.278e-02  25.796  < 2e-16 ***
## customer_typeTransient-Party    4.878e-01  5.527e-02   8.827  < 2e-16 ***
## required_car_parking_spaces    -3.778e+02  7.673e+05   0.000  0.99961
## total_of_special_requests      -5.700e-01  1.093e-02 -52.137  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 157398  on 119389  degrees of freedom
## Residual deviance: 103931  on 119350  degrees of freedom
## AIC: 104011
##
## Number of Fisher Scoring iterations: 12
```

The model output shows that most of the variables are significant, but some of them, like *children*, *babies* and *required_car_parking_space*, are surprisingly not significant, which means that the number of children or babies and the presence of a car parking are not relevant, in this linear model, in order to predict if a customer will cancel the reservation. We can compute the confusion matrix:

```r
logistic.prob_complete <- predict(complete_model, type="response")
logistic.pred_complete <- rep(0, length(logistic.prob_complete))
logistic.pred_complete[logistic.prob_complete > 0.5] <- 1
confusionMatrix(as.factor(logistic.pred_complete),
                hotel_bookings_new$is_canceled)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 71060 19375
##          1  4106 24849
##
##                Accuracy : 0.8033
##                  95% CI : (0.8011, 0.8056)
##     No Information Rate : 0.6296
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5461
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

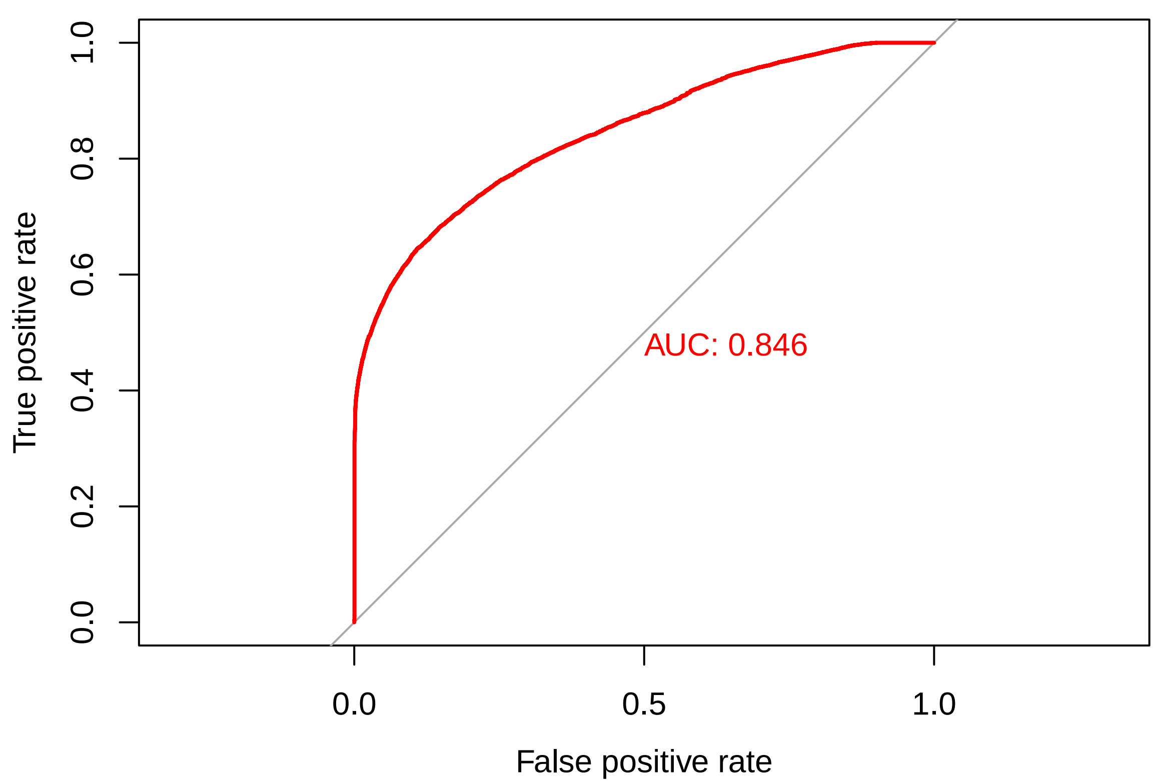
```
##
##             Sensitivity : 0.9454
##             Specificity : 0.5619
##          Pos Pred Value : 0.7858
##          Neg Pred Value : 0.8582
##              Prevalence : 0.6296
##          Detection Rate : 0.5952
##    Detection Prevalence : 0.7575
##       Balanced Accuracy : 0.7536
##
##        'Positive' Class : 0
##
```

And finally the ROC curve:

```r
# ROC Curve

roc.out_complete <- roc(hotel_bookings_new$is_canceled, logistic.prob_complete,
                        levels=c(0, 1))
```

```
## Setting direction: controls < cases
```

```r
plot(roc.out_complete,  print.auc=TRUE, legacy.axes=TRUE, xlab="False positive rate",
ylab="True positive rate", col="red")
```



At the beginning the idea was to use the function *regsubset* to perform model selection, but we soon find a serious problem: this function splits categorical variables in their levels and treats each of them as a dummy variable, which means that one of the levels could be found significant and then kept, while other levels of the same variable could be discarded. This is a serious issue when working with categorical variables. A good solution could be to use methods such as *force.in* or *force.out* to force some variables to be entirely in the model or entirely discarded. This strategy was not really helpful in this case, where almost all the variables are categorical and the use of such a method would drove us to hold them all, since at least some of levels for each variable turned out to be significant. It was also not clear how to perform the selection manually, since *glm* function does not return rates like *R* or the adjusted *R^2*.

So we decided to implement the function *stepAIC*, that takes as input the complete model and adds or removes progressively variables according to the hyper-parameter *direction* (typical values are *backward*, *forward* or *both*) so to try different combinations of predictors. The *direction* value was set to *both* (i.e. both *forward* and *backward*). The *stepAIC* function computes for each submodes the AIC, which is an estimator of prediction error and finally returns the model with the lowest AIC at each iteration and the correspondent AIC value.

```
model_AIC<-stepAIC(complete_model, direction = "both")
```

```
## Start:  AIC=104011.4
## is_canceled ~ hotel + lead_time + reservation_status_date + arrival_date_month +
##     total_stays + adults + children + babies + meal + distribution_channel +
##     is_repeated_guest + adr + previous_cancellations + previous_bookings_not_canceled +
##     booking_changes + deposit_type + days_in_waiting_list + customer_type +
##     required_car_parking_spaces + total_of_special_requests
##
##                                  Df Deviance    AIC
## - babies                          1   103932 104010
## <none>                                103931 104011
## - children                        1   103938 104016
## - adults                          1   103953 104031
## - days_in_waiting_list            1   103954 104032
## - hotel                           1   103977 104055
## - is_repeated_guest               1   104019 104097
## - total_stays                     1   104071 104149
## - arrival_date_month             11   104602 104660
## - booking_changes                 1   104593 104671
## - meal                            4   104769 104841
## - previous_bookings_not_canceled  1   104868 104946
## - distribution_channel            4   104994 105066
## - reservation_status_date         1   105115 105193
## - customer_type                   3   106086 106160
## - adr                             1   106094 106172
## - lead_time                       1   106476 106554
## - previous_cancellations          1   106605 106683
## - total_of_special_requests       1   106940 107018
## - required_car_parking_spaces     1   108331 108409
## - deposit_type                    2   113901 113977
##
## Step:  AIC=104009.8
## is_canceled ~ hotel + lead_time + reservation_status_date + arrival_date_month +
##     total_stays + adults + children + meal + distribution_channel +
##     is_repeated_guest + adr + previous_cancellations + previous_bookings_not_canceled +
##     booking_changes + deposit_type + days_in_waiting_list + customer_type +
```

```
##     required_car_parking_spaces + total_of_special_requests
##
##                                Df Deviance    AIC
## <none>                               103932 104010
## + babies                         1   103931 104011
## - children                       1   103938 104014
## - adults                         1   103953 104029
## - days_in_waiting_list           1   103954 104030
## - hotel                          1   103978 104054
## - is_repeated_guest              1   104019 104095
## - total_stays                    1   104072 104148
## - arrival_date_month            11   104603 104659
## - booking_changes                1   104594 104670
## - meal                           4   104769 104839
## - previous_bookings_not_canceled  1   104868 104944
## - distribution_channel           4   104994 105064
## - reservation_status_date        1   105116 105192
## - customer_type                  3   106087 106159
## - adr                            1   106094 106170
## - lead_time                      1   106476 106552
## - previous_cancellations         1   106605 106681
## - total_of_special_requests      1   106954 107030
## - required_car_parking_spaces    1   108332 108408
## - deposit_type                   2   113902 113976
```

The best model in our analysis turned out to be the complete model without *babies*, which fits in perfectly with the fact that *babies* has been recognized as non-significative in the first regression we performed on the complete model. We can see from the results that the difference in terms of AIC among the complete model and the one retrieved by the *stepAIC* function is minimal:

AIC_complete = 104011.4 AIC_not-babies = 104009.8

For an assessment of the models the AUC parameter has been chosen. This is an acronym for Area Under Curve and it provides a view on the overall performance of a classifier, summarized over all possible thresholds. The curve we refer to in this description is the ROC, that allows us to display the True Positive Rates (also called *Sensitivity*) and the False Positive Rates for all possible thresholds.

```r
logistic.prob_AIC=predict(model_AIC, type="response")
logistic.pred_AIC <- rep(0, length(logistic.prob_AIC))
logistic.pred_AIC[logistic.prob_complete > 0.5] <- 1
confusionMatrix(as.factor(logistic.pred_AIC), hotel_bookings_new$is_canceled)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 71060 19375
##          1  4106 24849
##
##                Accuracy : 0.8033
##                  95% CI : (0.8011, 0.8056)
##     No Information Rate : 0.6296
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

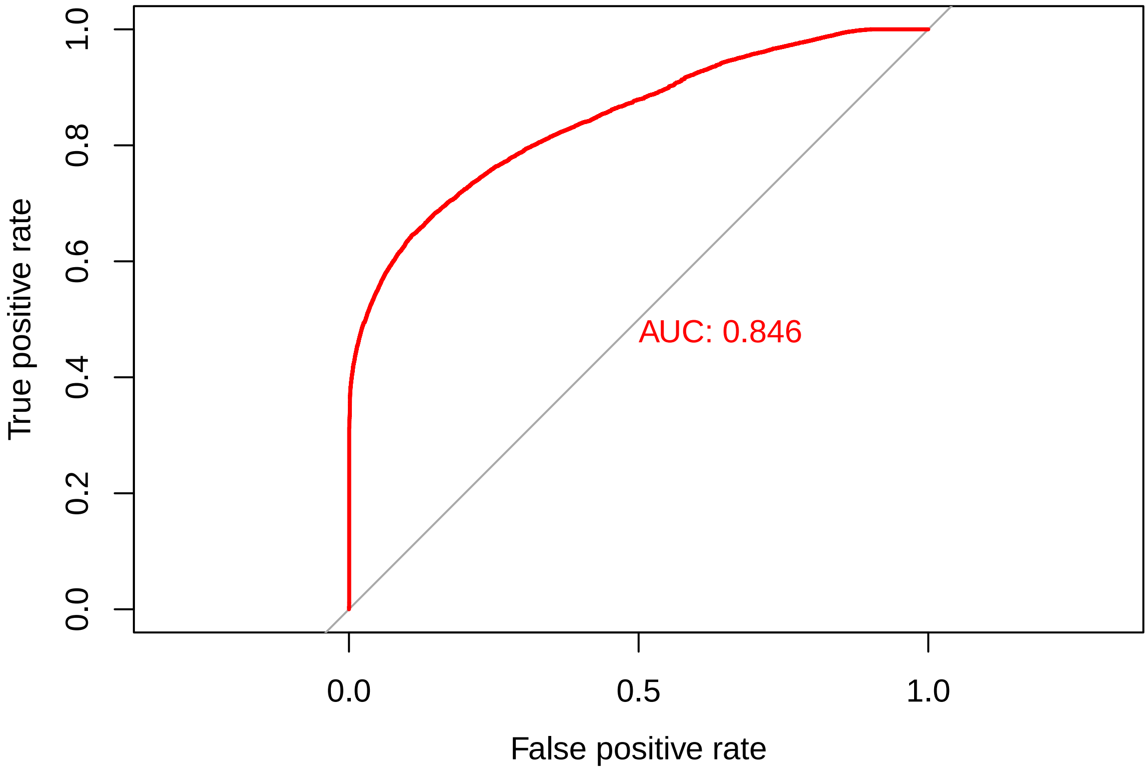
```
##                  Kappa : 0.5461
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##            Sensitivity : 0.9454
##            Specificity : 0.5619
##         Pos Pred Value : 0.7858
##         Neg Pred Value : 0.8582
##             Prevalence : 0.6296
##         Detection Rate : 0.5952
##   Detection Prevalence : 0.7575
##      Balanced Accuracy : 0.7536
##
##       'Positive' Class : 0
##
```

As can be seen from the plot, the ROC curve is almost exactly the same for our best models, which are the complete model and the complete one without babies. All the other curves corresponding to different models checked by the *stepAIC* function turn out to have a smaller AUC value, which implies a worse overall performance. An important remark is that AIC rate generally favors models with less variables. For this reason, even if in the end the two models are equivalent, the one without *babies* is to be preferred according to AIC rate.

```r
# ROC Curve

logistic.prob_AIC=predict(model_AIC, type="response")
roc.out_AIC <- roc(hotel_bookings_new$is_canceled, logistic.prob_AIC, levels=c(0, 1))
```

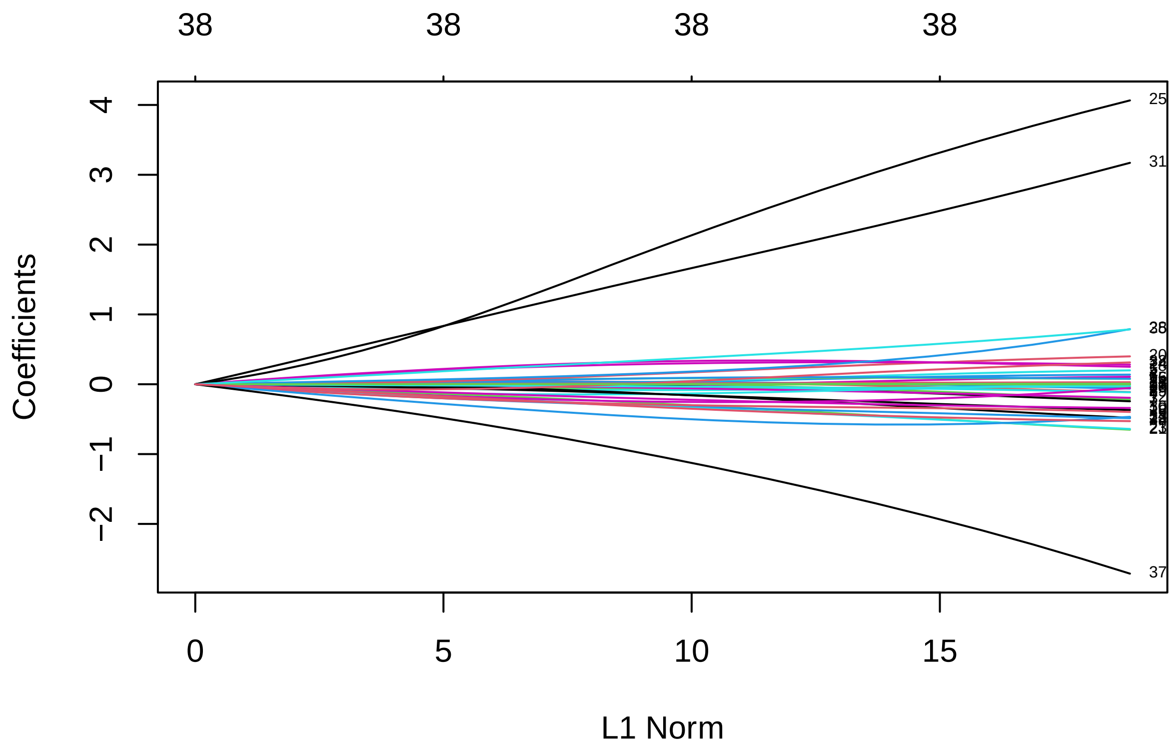
```
## Setting direction: controls < cases
```

```r
plot(roc.out_AIC,  print.auc=TRUE, legacy.axes=TRUE, xlab="False positive rate",
     ylab="True positive rate", col="red")
```

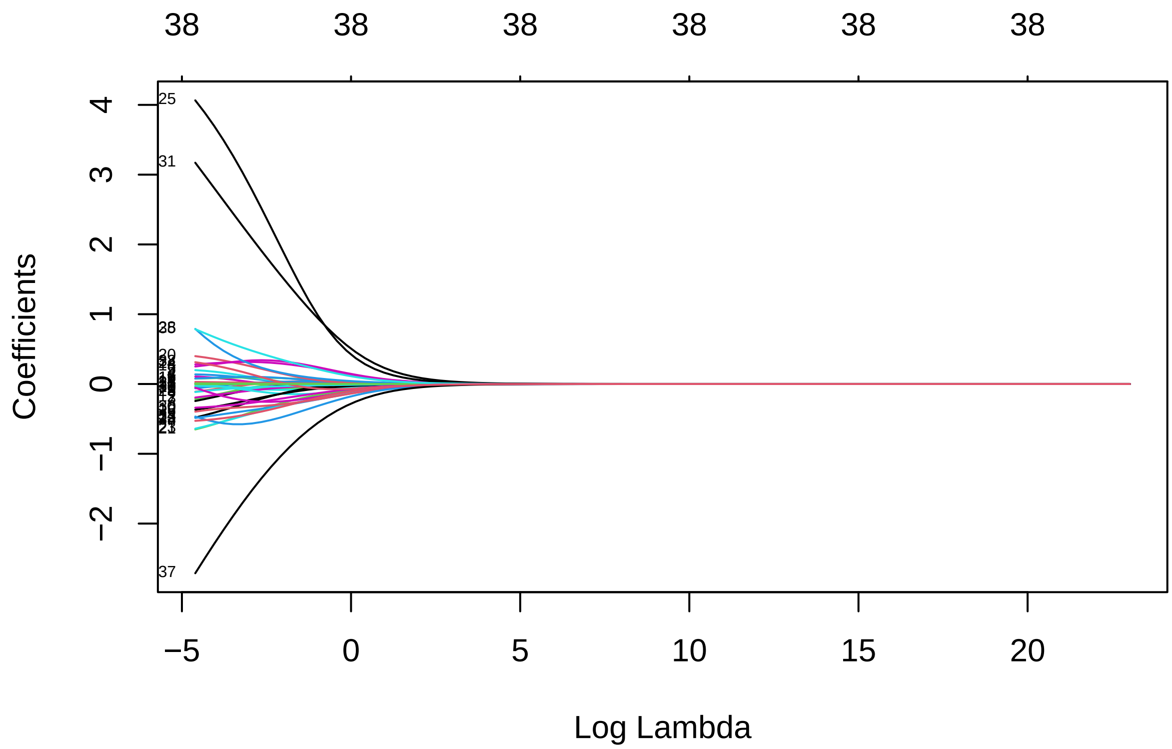


RIDGE

```
x <- model.matrix(is_canceled~lead_time+reservation_status_date+arrival_date_month+
total_stays+adults+children+babies+meal+distribution_channel+is_repeated_guest+adr+
previous_cancellations+previous_bookings_not_canceled+booking_changes+deposit_type+
days_in_waiting_list+customer_type+required_car_parking_spaces+
total_of_special_requests, hotel_bookings_new)[,-1]
y <- hotel_bookings_new$is_canceled

#Ridge plots
grid <- 10^seq(10, -2, length=100)
ridge.mod <- glmnet(x, y, alpha=0, family="binomial", lambda=grid)
plot(ridge.mod, label=TRUE)
```




```
plot(ridge.mod, xvar="lambda", label=TRUE)
```




```r
grid[65:75]
```

```
##  [1] 174.75284 132.19411 100.00000  75.64633  57.22368  43.28761  32.74549
##  [8]  24.77076  18.73817  14.17474  10.72267
```

```r
predict(ridge.mod, s=50, type="coefficients")
```

```
## 39 x 1 sparse Matrix of class "dgCMatrix"
##                                         s1
## (Intercept)                  -4.205117e-01
## lead_time                     2.674068e-05
## reservation_status_date      -7.020565e-06
## arrival_date_monthAugust      1.653373e-04
## arrival_date_monthDecember   -4.385458e-04
## arrival_date_monthFebruary   -7.769821e-04
## arrival_date_monthJanuary    -1.385730e-03
## arrival_date_monthJuly        9.255354e-05
## arrival_date_monthJune        9.788339e-04
## arrival_date_monthMarch      -1.072628e-03
## arrival_date_monthMay         5.872919e-04
## arrival_date_monthNovember   -1.243119e-03
## arrival_date_monthOctober     2.162196e-04
## arrival_date_monthSeptember   4.556363e-04
## total_stays                   6.790432e-05
```

```
## adults                            1.008852e-03
## children                          1.323124e-04
## babies                           -3.235136e-03
## mealFB                            4.639641e-03
## mealHB                           -6.003486e-04
## mealSC                            5.196657e-05
## mealUndefined                    -2.561198e-03
## distribution_channelDirect       -4.489128e-03
## distribution_channelGDS          -3.605817e-03
## distribution_channelTA/TO         4.446851e-03
## distribution_channelUndefined     8.769943e-03
## is_repeated_guest1               -4.682389e-03
## adr                               9.284747e-06
## previous_cancellations            1.271342e-03
## previous_bookings_not_canceled -3.717100e-04
## booking_changes                  -2.157136e-03
## deposit_typeNon Refund            1.435728e-02
## deposit_typeRefundable           -2.969693e-03
## days_in_waiting_list              2.982401e-05
## customer_typeGroup               -5.433403e-03
## customer_typeTransient            3.009255e-03
## customer_typeTransient-Party     -2.975662e-03
## required_car_parking_spaces      -7.781081e-03
## total_of_special_requests        -2.891373e-03
```

```
#Ridge on train set
train <- sample(1:nrow(x), nrow(x)/2)
test <- (-train)
y.test <- y[test]

ridge.mod <- glmnet(x[train, ], y[train], alpha = 0, family="binomial",
                    lambda = grid, thresh = 1e-12)

ridge.pred <- predict(ridge.mod, s = 4, newx = x[test, ], type="response")
predicted.classes <- ifelse(ridge.pred > 0.5, "1", "0")
confusionMatrix(y.test, as.factor(predicted.classes))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 37471     0
##          1 22176    48
##
##                Accuracy : 0.6285
##                  95% CI : (0.6246, 0.6324)
##     No Information Rate : 0.9992
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0027
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.62821
```

```
##            Specificity : 1.00000
##         Pos Pred Value : 1.00000
##         Neg Pred Value : 0.00216
##             Prevalence : 0.99920
##         Detection Rate : 0.62771
##   Detection Prevalence : 0.62771
##      Balanced Accuracy : 0.81411
##
##       'Positive' Class : 0
##
```

```
#Cross-validation to choose lambda
set.seed(1)
cv.out <- cv.glmnet(x[train, ], y[train], alpha = 0, family="binomial", nfold=10)
cv.out$lambda[1:10]
```

```
##  [1] 233.7576 212.9912 194.0697 176.8291 161.1200 146.8066 133.7647 121.8814
##  [9] 111.0538 101.1881
```

```
summary(cv.out$lambda)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##   0.02338   0.23394   2.34011  26.31062  23.39503 233.75760
```

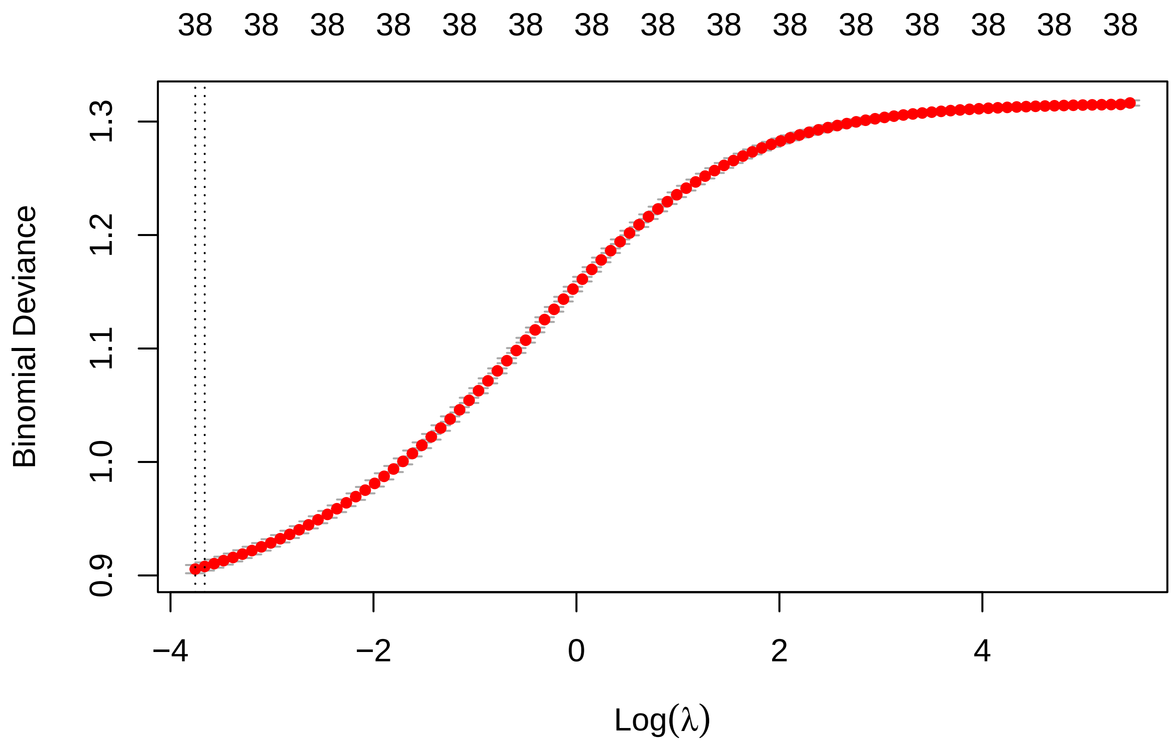
```
cv.out$cvm[1:10]
```

```
##  [1] 1.316380 1.315119 1.314986 1.314850 1.314700 1.314536 1.314356 1.314159
##  [9] 1.313943 1.313706
```

```
cv.out$cvsd[1:10]
```

```
##  [1] 0.002318557 0.002311099 0.002310559 0.002310000 0.002309387 0.002308716
##  [7] 0.002307981 0.002307175 0.002306293 0.002305327
```

```
plot(cv.out)
```




```
i.bestlam <- which.min(cv.out$cvm)
#i.bestlam
bestlam <- cv.out$lambda[i.bestlam]
#bestlam
cv.out$cvm[i.bestlam]
```

```
## [1] 0.9055696
```

```
#min(cv.out$cvm)

bestlam <- cv.out$lambda.min
cat("Best lambda is:", bestlam)
```

```
## Best lambda is: 0.02337576
```

```
#Ridge with best lambda
ridge.pred <- predict(ridge.mod, s = bestlam,
                      newx = x[test, ])
#Evaluation
predicted.classes <- ifelse(ridge.pred > 0.5, "1", "0")
confusionMatrix(y.test, as.factor(predicted.classes))
```

```
## Confusion Matrix and Statistics
##
```

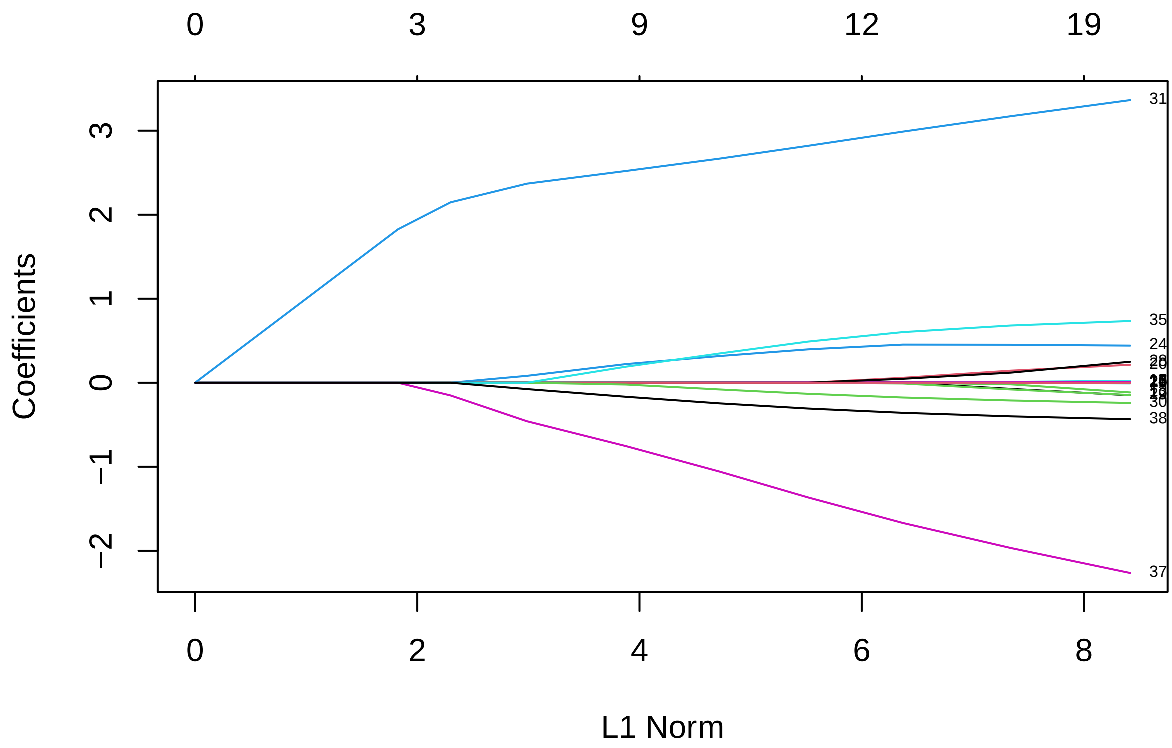
```
##           Reference
## Prediction     0     1
##          0 37246   225
##          1 13759  8465
##
##                Accuracy : 0.7657
##                  95% CI : (0.7623, 0.7691)
##     No Information Rate : 0.8544
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.4279
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7302
##             Specificity : 0.9741
##          Pos Pred Value : 0.9940
##          Neg Pred Value : 0.3809
##              Prevalence : 0.8544
##          Detection Rate : 0.6239
##    Detection Prevalence : 0.6277
##       Balanced Accuracy : 0.8522
##
##        'Positive' Class : 0
##
```

Ridge Regression has a discrete performance on our data, with an accuracy of 0.7719. According to the respective Confusion Matrix, it is possible to notice that the Specificity is higher than Sensitivity with a very high score of 0.9808.

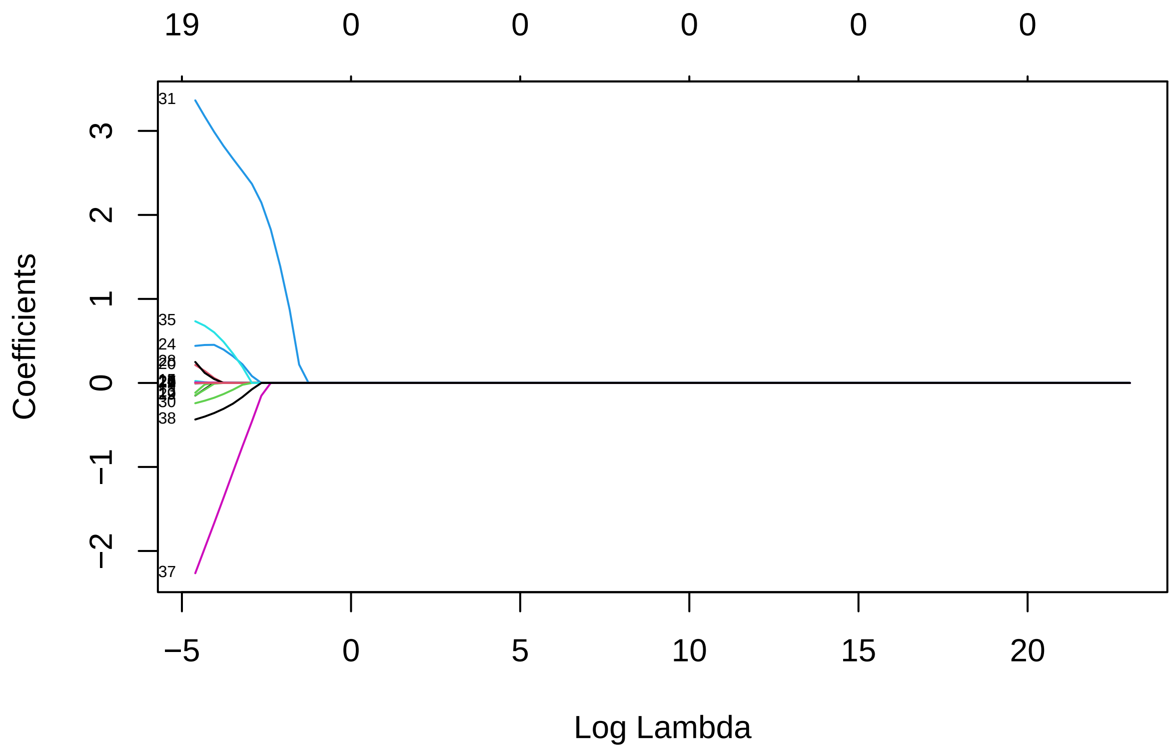
LASSO

```
lasso.mod <- glmnet(x,y,alpha=1, family="binomial", lambda=grid)
plot(lasso.mod, label=TRUE)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm): si
## riduce a valori unici di 'x'
```




```r
plot(lasso.mod, xvar="lambda", label=TRUE)
```

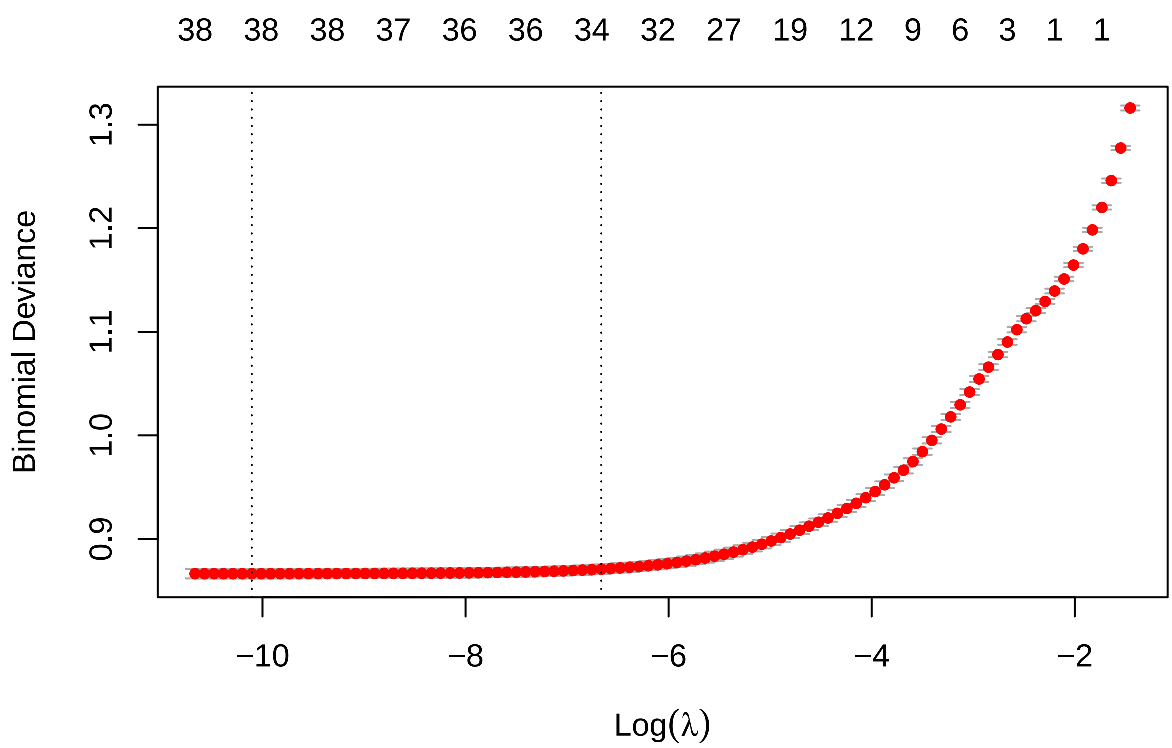



```r
set.seed(1)
cv.out <- cv.glmnet(x[train,], y[train], family="binomial", alpha=1)
```

```
## Warning: from glmnet C++ code (error code -92); Convergence for 92th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned

## Warning: from glmnet C++ code (error code -92); Convergence for 92th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned

## Warning: from glmnet C++ code (error code -92); Convergence for 92th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned

## Warning: from glmnet C++ code (error code -92); Convergence for 92th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned

## Warning: from glmnet C++ code (error code -92); Convergence for 92th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned

## Warning: from glmnet C++ code (error code -92); Convergence for 92th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
```

```
## returned

## Warning: from glmnet C++ code (error code -92); Convergence for 92th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

```
plot(cv.out)
```




```
bestlam <- cv.out$lambda.min
cat("Best lambda for Lasso Regression is:", bestlam)
```

```
## Best lambda for Lasso Regression is: 4.08498e-05
```

```
lasso.pred <- predict(lasso.mod, s=bestlam, newx=x[test,])
#Evaluation
predicted.classes <- ifelse(lasso.pred > 0.5, "1", "0")
confusionMatrix(y.test, as.factor(predicted.classes))
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction     0     1
##          0 37319   152
##          1 14508  7716
```

```
##
##                Accuracy : 0.7544
##                  95% CI : (0.7509, 0.7579)
##     No Information Rate : 0.8682
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3951
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7201
##             Specificity : 0.9807
##          Pos Pred Value : 0.9959
##          Neg Pred Value : 0.3472
##              Prevalence : 0.8682
##          Detection Rate : 0.6252
##    Detection Prevalence : 0.6277
##       Balanced Accuracy : 0.8504
##
##        'Positive' Class : 0
##
```

Lasso Regression shows similar results to Ridge. However, there is a little difference in accuracy, with a score of 0.7567. In conclusion, Ridge ends up to be the best models.

Logistic VS LDA VS QDA

```
#Train & Test
set.seed(321)
trainIndex <- createDataPartition(hotel_bookings_new$is_canceled,
                                  p=0.75,list=FALSE)
hotel_data_train=hotel_bookings_new[trainIndex,]
hotel_data_test=hotel_bookings_new[-trainIndex,]
is_canc_test=hotel_data_test$is_canceled
```

```
#Logistic Regression
glm.fits <- glm(is_canceled~lead_time+adr+total_of_special_requests,
                data=hotel_data_train,family=binomial)
glm.probs <- predict(glm.fits,hotel_data_test,type="response")
glm.pred <- rep(0,29847)
glm.pred[glm.probs>.5] <- 1

#Evaluation
confusionMatrix(as.factor(glm.pred),is_canc_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 16265  6501
##          1  2526  4555
```

```
##
##                Accuracy : 0.6976
##                  95% CI : (0.6923, 0.7028)
##     No Information Rate : 0.6296
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2997
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8656
##             Specificity : 0.4120
##          Pos Pred Value : 0.7144
##          Neg Pred Value : 0.6433
##              Prevalence : 0.6296
##          Detection Rate : 0.5449
##    Detection Prevalence : 0.7628
##       Balanced Accuracy : 0.6388
##
##        'Positive' Class : 0
##
```

```r
cat("Mean error is:", mean(glm.pred!=is_canc_test))
```

```
## Mean error is: 0.3024425
```

```r
#Polinomial Regression
glm.fits <- glm(is_canceled~lead_time+adr+total_of_special_requests+I(adr^3)+
                I(lead_time^2)+I(total_of_special_requests^2),
                data=hotel_data_train,family=binomial)
glm.probs <- predict(glm.fits,hotel_data_test,type="response")
glm.pred <- rep(0,29847)
glm.pred[glm.probs>.5] <- 1

#Evaluation
confusionMatrix(as.factor(glm.pred),is_canc_test)
```

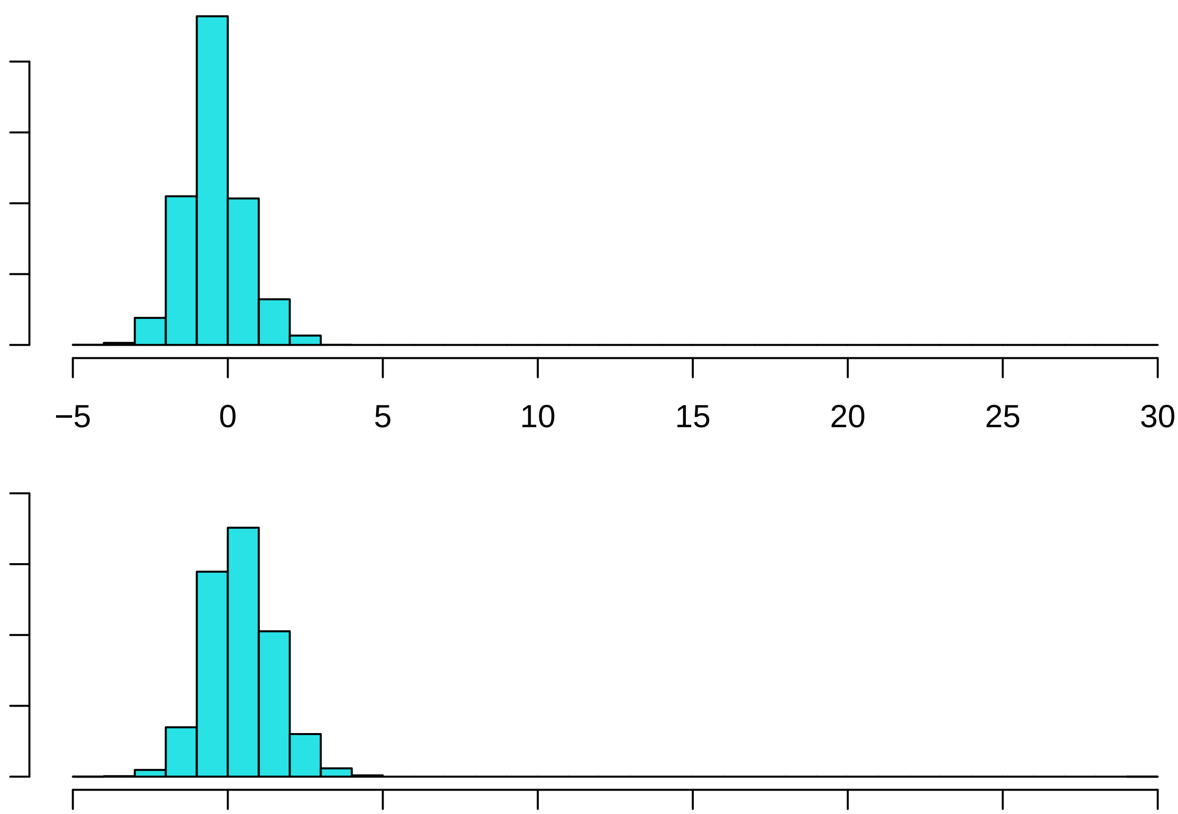
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 15808  5946
##          1  2983  5110
##
##                Accuracy : 0.7008
##                  95% CI : (0.6956, 0.706)
##     No Information Rate : 0.6296
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3212
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
##             Sensitivity : 0.8413
##             Specificity : 0.4622
##          Pos Pred Value : 0.7267
##          Neg Pred Value : 0.6314
##              Prevalence : 0.6296
##          Detection Rate : 0.5296
##    Detection Prevalence : 0.7289
##       Balanced Accuracy : 0.6517
##
##        'Positive' Class : 0
##
```

```
cat("Mean error is:", mean(glm.pred!=is_canc_test))
```

```
## Mean error is: 0.299159
```

```
#LDA
lda.fit <- lda(is_canceled~lead_time+adr+total_of_special_requests,
               data=hotel_data_train)
par(mar=c(1,1,1,1))
plot(lda.fit)
```




```
lda.pred <- predict(lda.fit, hotel_data_test)
lda.class <- lda.pred$class
```

```r
#Evaluation
confusionMatrix(as.factor(lda.class),is_canc_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 16345  6707
##          1  2446  4349
##
##                Accuracy : 0.6933
##                  95% CI : (0.6881, 0.6986)
##     No Information Rate : 0.6296
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2859
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8698
##             Specificity : 0.3934
##          Pos Pred Value : 0.7090
##          Neg Pred Value : 0.6400
##              Prevalence : 0.6296
##          Detection Rate : 0.5476
##    Detection Prevalence : 0.7723
##       Balanced Accuracy : 0.6316
##
##        'Positive' Class : 0
##
```

```r
cat("Mean error is:", mean(glm.pred!=is_canc_test))
```

```
## Mean error is: 0.299159
```

```r
#QDA
qda.fit <- qda(is_canceled~lead_time+adr+total_of_special_requests,
               data=hotel_data_train)
qda.pred <- predict(qda.fit, hotel_data_test)
qda.class <- qda.pred$class


# Evaluation
confusionMatrix(as.factor(qda.class),is_canc_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 16020  6601
##          1  2771  4455
```

```
##
##                Accuracy : 0.686
##                  95% CI : (0.6807, 0.6913)
##     No Information Rate : 0.6296
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2751
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8525
##             Specificity : 0.4029
##          Pos Pred Value : 0.7082
##          Neg Pred Value : 0.6165
##              Prevalence : 0.6296
##          Detection Rate : 0.5367
##    Detection Prevalence : 0.7579
##       Balanced Accuracy : 0.6277
##
##        'Positive' Class : 0
##
```

```r
cat("Mean error is:", mean(glm.pred!=is_canc_test))
```

```
## Mean error is: 0.299159
```

Conclusion

The main purpose of this work was to find, given the booking dataset, if it were possible to predict the event of booking cancellation through statistical models. However, working on a dataset where most of variables are categorical turned out to be far more complex than we expected for models like Logistic Regression, Ridge, Lasso and the others. Values like adjusted $R\hat{}2$ that are typically used to assess the quality of a model cannot be used in such a case. We then made use of the AIC value during the analysis.

Better results would have been achieved by using more complex models. For example a Decision Tree seems to be a good choice when you deal with a lot of categorical variables.

Despite all these limits our final model achieves good results on the test set and we can conclude that Logistic Regression, Ridge and Lasso are good models. In particular, with an accuracy of 0.8033, Logistic Regression is the best implemented model. According to accuracy scores, Lasso and Ridge show a slightly lower performance, respectively of 0.7577 and 0.7719. A larger difference in scores can be seen in Specificity and Sensitivity, where Logistic Regression scores are quite different from Ridge and Lasso scores (lower in Specificity, higher in Sensitivity).

A final remark is that the dataset lended itself very well to exploratory analysis and all the relevant informations we have been able to extract from it can be of great help for a hotel in order to plan the future work.