Hotel Reservation Prediction

Exploratory Data Analysis

Load dataset

```
df <- read.csv("hotel_reservation.csv")
```

Dimensionality

```
dim(df)
```

```
## [1] 17908    30
```

Structure

```
str(df)
```

```
## 'data.frame':    17908 obs. of  30 variables:
##  $ hotel                         : chr  "Resort Hotel" "City Hotel" "City Hotel" "City Hotel" ...
##  $ is_canceled                   : int  0 1 0 0 0 0 1 0 0 0 ...
##  $ lead_time                     : int  203 82 25 1 70 170 21 102 55 222 ...
##  $ arrival_date_year             : int  2016 2015 2016 2016 2017 2017 2016 2015 2016 2015 ...
##  $ arrival_date_month            : chr  "December" "July" "December" "March" ...
##  $ arrival_date_week_number      : int  49 29 53 11 16 17 10 42 47 38 ...
##  $ arrival_date_day_of_month     : int  2 16 27 9 16 27 4 16 19 14 ...
##  $ stays_in_weekend_nights       : int  2 0 0 0 2 0 0 0 2 1 ...
##  $ stays_in_week_nights          : int  5 3 3 1 2 3 1 2 5 1 ...
##  $ adults                        : int  2 2 3 1 2 2 1 2 2 2 ...
##  $ children                      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ babies                        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ meal                          : chr  "BB" "BB" "BB" "BB" ...
##  $ country                       : chr  "GBR" "PRT" "BRA" "SWE" ...
##  $ market_segment                : chr  "Direct" "Online TA" "Offline TA/TO" "Online TA" ...
##  $ distribution_channel          : chr  "Direct" "TA/TO" "TA/TO" "TA/TO" ...
##  $ is_repeated_guest             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ previous_cancellations        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ previous_bookings_not_canceled: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ reserved_room_type            : chr  "F" "A" "A" "A" ...
##  $ assigned_room_type            : chr  "F" "A" "K" "A" ...
##  $ booking_changes               : int  4 0 2 0 0 0 0 0 1 0 ...
```

```
##  $ deposit_type                   : chr  "No Deposit" "No Deposit" "No Deposit" "No Deposit" ...
##  $ agent                          : num  250 9 220 9 9 9 9 6 314 68 ...
##  $ company                        : num  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 ...
##  $ customer_type                  : chr  "Transient" "Transient" "Transient-Party" "Transient-Party"
##  $ adr                            : num  66.8 76.5 60 95 108 ...
##  $ required_car_parking_spaces    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ total_of_special_requests      : int  0 0 1 0 0 0 0 0 0 0 ...
```

Summary

```
summary(df)
```

```
##     hotel            is_canceled       lead_time     arrival_date_year
##  Length:17908       Min.   :0.0000   Min.   :  0.0   Min.   :2015     
##  Class :character   1st Qu.:0.0000   1st Qu.: 19.0   1st Qu.:2016     
##  Mode  :character   Median :0.0000   Median : 70.0   Median :2016     
##                     Mean   :0.3793   Mean   :104.7   Mean   :2016     
##                     3rd Qu.:1.0000   3rd Qu.:161.0   3rd Qu.:2017     
##                     Max.   :1.0000   Max.   :629.0   Max.   :2017     
##                                                                       
##  arrival_date_month arrival_date_week_number arrival_date_day_of_month
##  Length:17908       Min.   : 1.00            Min.   : 1.00            
##  Class :character   1st Qu.:16.00            1st Qu.: 8.00            
##  Mode  :character   Median :27.00            Median :16.00            
##                     Mean   :27.13            Mean   :15.88            
##                     3rd Qu.:38.00            3rd Qu.:24.00            
##                     Max.   :53.00            Max.   :31.00            
##                                                                       
##  stays_in_weekend_nights stays_in_week_nights     adults          children     
##  Min.   : 0.000          Min.   : 0.000       Min.   : 0.000   Min.   :0.0000  
##  1st Qu.: 0.000          1st Qu.: 1.000       1st Qu.: 2.000   1st Qu.:0.0000  
##  Median : 1.000          Median : 2.000       Median : 2.000   Median :0.0000  
##  Mean   : 0.936          Mean   : 2.509       Mean   : 1.858   Mean   :0.1043  
##  3rd Qu.: 2.000          3rd Qu.: 3.000       3rd Qu.: 2.000   3rd Qu.:0.0000  
##  Max.   :19.000          Max.   :50.000       Max.   :55.000   Max.   :3.0000  
##                                                                                
##      babies              meal             country          market_segment    
##  Min.   :0.000000   Length:17908       Length:17908       Length:17908      
##  1st Qu.:0.000000   Class :character   Class :character   Class :character  
##  Median :0.000000   Mode  :character   Mode  :character   Mode  :character  
##  Mean   :0.007483                                                           
##  3rd Qu.:0.000000                                                           
##  Max.   :9.000000                                                           
##                                                                             
##  distribution_channel is_repeated_guest previous_cancellations
##  Length:17908         Min.   :0.000     Min.   : 0.00000      
##  Class :character     1st Qu.:0.000     1st Qu.: 0.00000      
##  Mode  :character     Median :0.000     Median : 0.00000      
##                       Mean   :0.033     Mean   : 0.08722      
##                       3rd Qu.:0.000     3rd Qu.: 0.00000      
##                       Max.   :1.000     Max.   :26.00000      
```

```
##
##  previous_bookings_not_canceled reserved_room_type assigned_room_type
##  Min.   : 0.0000                Length:17908       Length:17908
##  1st Qu.: 0.0000                Class :character   Class :character
##  Median : 0.0000                Mode  :character   Mode  :character
##  Mean   : 0.1494
##  3rd Qu.: 0.0000
##  Max.   :56.0000
##
##  booking_changes   deposit_type          agent           company
##  Min.   : 0.0000   Length:17908       Min.   :  1.00   Min.   :  9.0
##  1st Qu.: 0.0000   Class :character   1st Qu.:  9.00   1st Qu.: 67.0
##  Median : 0.0000   Mode  :character   Median : 14.00   Median :174.0
##  Mean   : 0.2168                      Mean   : 87.32   Mean   :187.4
##  3rd Qu.: 0.0000                      3rd Qu.:229.00   3rd Qu.:266.5
##  Max.   :17.0000                      Max.   :531.00   Max.   :525.0
##                                       NA's   :2489     NA's   :16881
##  days_in_waiting_list customer_type           adr
##  Min.   :  0.000      Length:17908       Min.   :  0.0
##  1st Qu.:  0.000      Class :character   1st Qu.: 68.4
##  Median :  0.000      Mode  :character   Median : 94.5
##  Mean   :  2.212                         Mean   :101.5
##  3rd Qu.:  0.000                         3rd Qu.:125.1
##  Max.   :391.000                         Max.   :451.5
##
##  required_car_parking_spaces total_of_special_requests
##  Min.   :0.0000              Min.   :0.0000
##  1st Qu.:0.0000              1st Qu.:0.0000
##  Median :0.0000              Median :0.0000
##  Mean   :0.0588              Mean   :0.5687
##  3rd Qu.:0.0000              3rd Qu.:1.0000
##  Max.   :3.0000              Max.   :5.0000
##
```

Import library

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

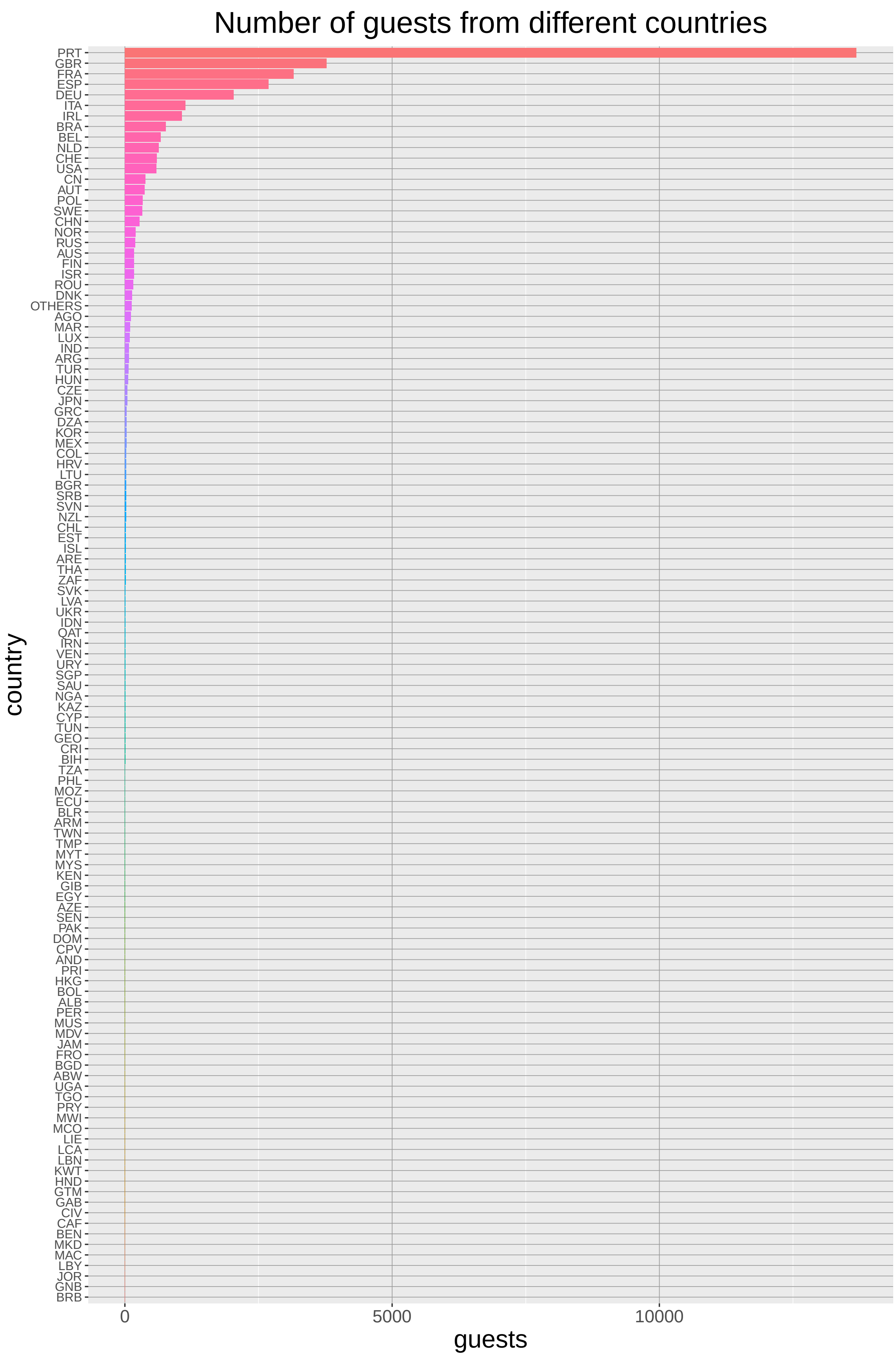
Calculate the number of guests from different countries

```
country_guests <- df%>%
  # Group the dataset by country
  group_by(country)%>%
  # Replace null as OTHERS for better understanding
  mutate(country=ifelse(is.na(country)|country=="","OTHERS",country))%>%
  # Calculate the sum of guests
  summarise(guests=sum(adults+children+babies),
            canceled=sum((adults+children+babies)*(is_canceled=="1")),
            not_canceled=sum((adults+children+babies)*(is_canceled=="0")))%>%
  # Show the result of countries in the order from the most to least guests
  arrange(desc(guests))
print(country_guests)
```

```
## # A tibble: 119 x 4
##    country guests canceled not_canceled
##    <chr>    <dbl>    <dbl>        <dbl>
##  1 PRT      13688     8201         5487
##  2 GBR       3775      856         2919
##  3 FRA       3159      604         2555
##  4 ESP       2691      780         1911
##  5 DEU       2033      360         1673
##  6 ITA       1135      413          722
##  7 IRL       1071      245          826
##  8 BRA        763      339          424
##  9 BEL        672      119          553
## 10 NLD        635      155          480
## # i 109 more rows
```

Visualize the number of guests from different countries

```
ggplot(data=country_guests,mapping=aes(x=guests,y=reorder(country,guests),fill=reorder(country,guests)))
  # Adjust the countries in order of guests
  geom_col()+
  # Set title
  labs(y="country",title="Number of guests from different countries")+
  theme(plot.title=element_text(size=24,hjust=0.5),
        # Don't show legend
        legend.position="none",
        # Set background
        panel.grid.major=element_line(color="grey60",linewidth=0.25),
        # Change text size
        axis.text.y=element_text(size=10),
        axis.text.x=element_text(size=14),
        axis.title.y=element_text(size=20),
        axis.title.x=element_text(size=20))
```



Number of guests from different countries
country
PRT
GBR
FRA
ESP
DEU
ITA
IRL
BRA
BEL
NLD
CHE
USA
CN
AUT
POL
SWE
CHN
NOR
RUS
AUS
FIN
ISR
ROU
DNK
OTHERS
AGO
MAR
LUX
IND
ARG
TUR
HUN
CZE
JPN
GRC
DZA
KOR
MEX
COL
HRV
LTU
BGR
SRB
SVN
NZL
CHL
EST
ISL
ARE
THA
ZAF
SVK
LVA
UKR
IDN
QAT
IRN
VEN
URY
SGP
SAU
NGA
KAZ
CYP
TUN
GEO
CRI
BIH
TZA
PHL
MOZ
ECU
BLR
ARM
TWN
TMP
MYT
MYS
KEN
GIB
EGY
AZE
SEN
PAK
DOM
CPV
AND
PRI
HKG
BOL
ALB
PER
MUS
MDV
JAM
FRO
BGD
ABW
UGA
TGO
PRY
MWI
MCO
LIE
LCA
LBN
KWT
HND
GTM
GAB
CIV
CAF
BEN
MKD
MAC
LBY
JOR
GNB
BRB
0
5000
10000
guests

Analysis: There are a total of 119 countries in the picture, and when drawing the picture, we found that some of them have empty values, so they are named after "OTHERS". From the graph, it can be observed that the top 10 countries with the highest number of residents are Portugal, the United Kingdom, France, Spain, Germany, Italy, Ireland, Brazil, Belgium, and the Netherlands. Among them, Portugal has the highest number of hotel guests, surpassing the second ranked country by about three times. And all the top 10 countries are from Europe. On the contrary, we can also observe from the visualization that only 1-2 people from 21 countries are staying in these two hotels, most of which come from West Africa and Central America.

```r
library(maps)
library(countrycode)

# Obtain the country border information
world_map <- map_data("world")
country_guests$country_name <- countrycode(country_guests$country, "iso3c", "country.name")
```

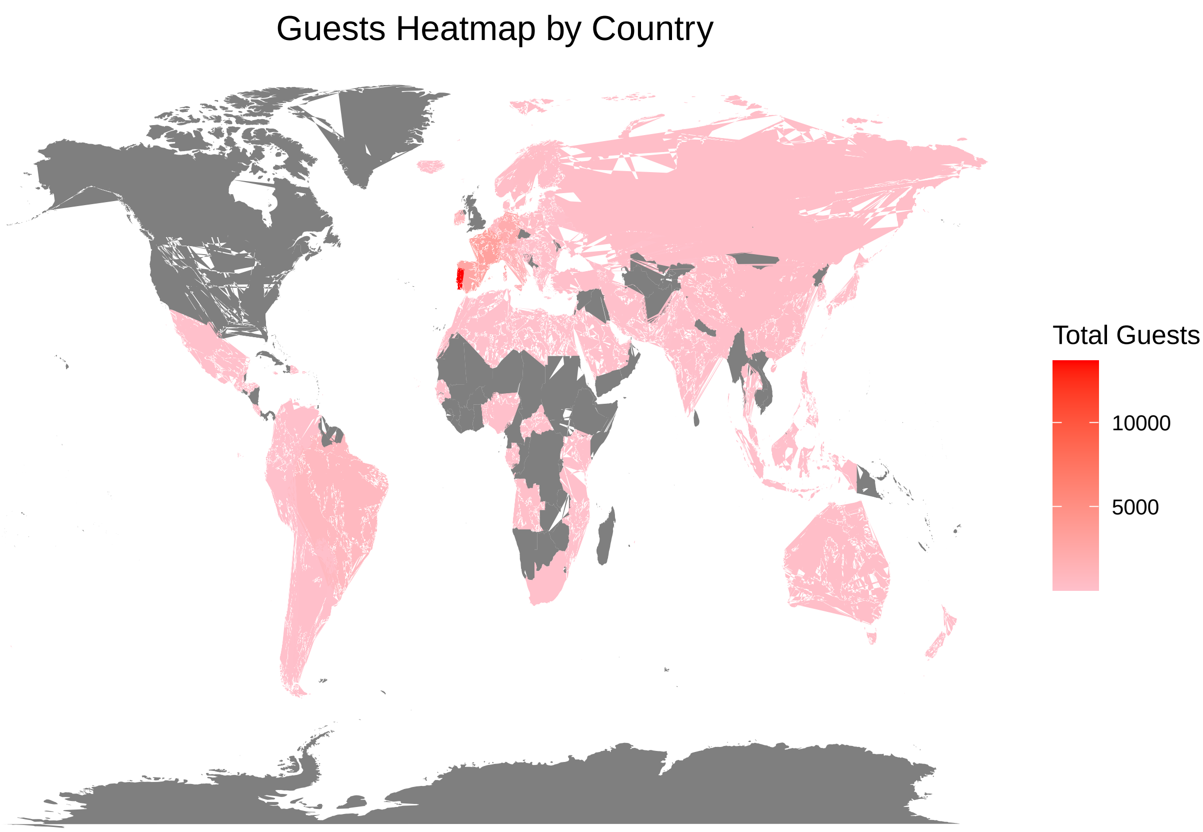
```
## Warning: Some values were not matched unambiguously: CN, OTHERS, TMP
```

```r
# Assigns "China"/"Timor-Leste" to the country_name attribute of a record
# whose country attribute is "CN"/"TMP" in the dataset
country_guests <- country_guests %>%
  mutate(country_name = ifelse(country == "CN", "China", country_name),
         country_name = ifelse(country == "TMP", "Timor-Leste", country_name))

# Merge data
merged_data <- merge(world_map, country_guests,
                     by.x="region", by.y="country_name",
                     all.x = TRUE)

# Create Heat Map
library(ggplot2)

ggplot(merged_data,
       aes(x=long, y=lat,
           group=group,
           fill=guests)) +
  geom_polygon() +
  scale_fill_gradient(low="pink", high="red", name="Total Guests") +
  theme_void() +
  labs(title="Guests Heatmap by Country")+
  theme(plot.title=element_text(hjust=0.5),
        legend.text=element_text(size=8),
        legend.title=element_text(size=10))
```

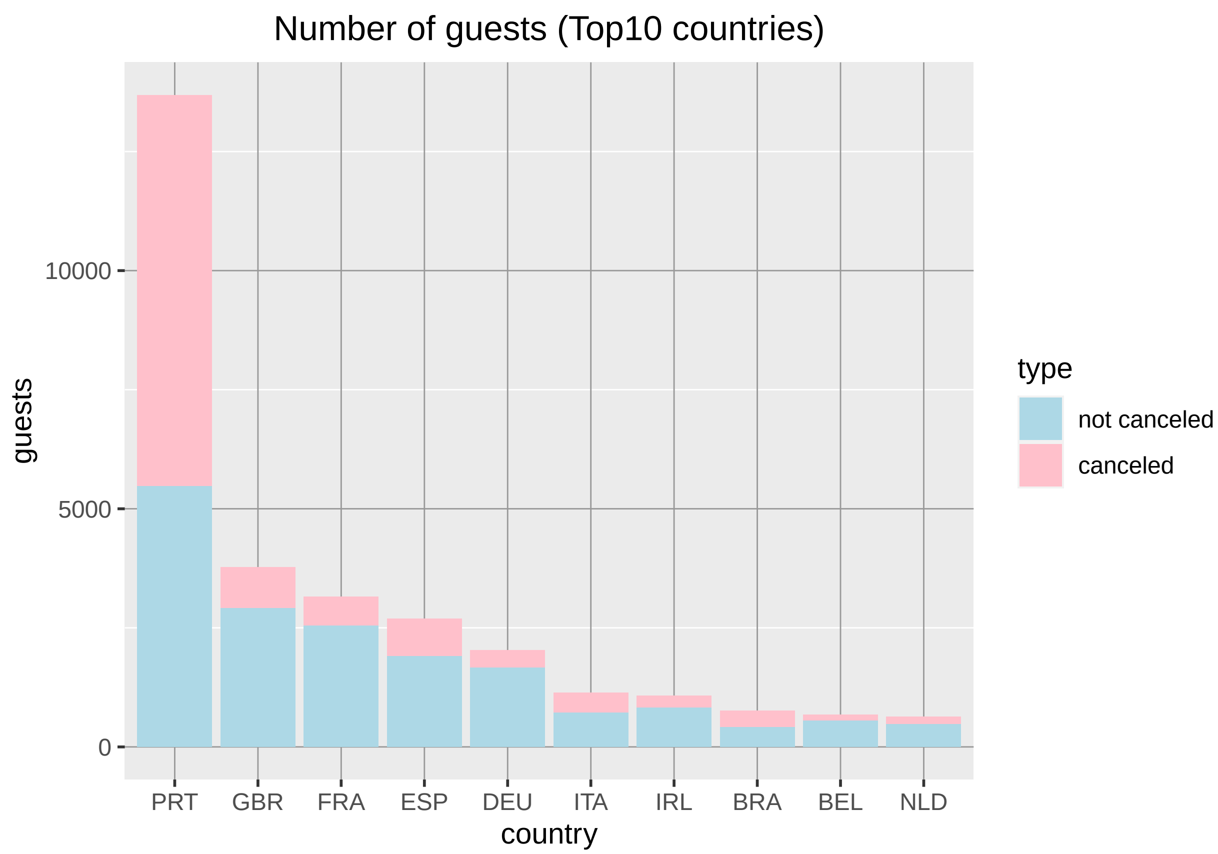


Analysis: In order to clearly show the geography distribution of the countries, we also draw the map, which showed that most of the customers are from Europe. Also, some of the countries that have high volume of customers are located in Africa.

Plot the top 10 countries with the largest number of guests

```
library(tidyr)
# Get data of top 10 countries
top10_country <- head(country_guests,10)%>%
  select(country,not_canceled,canceled)%>%
  gather("type","guests",-country)
# Plot
ggplot(data=top10_country,mapping=aes(x=reorder(country,-guests),y=guests,fill=type))+
  # Set color of bars
  geom_bar(stat="identity")+
  # Set title
  labs(x="country",title="Number of guests (Top10 countries)")+
  # Reset legend for better understanding
  scale_fill_manual(breaks=c("not_canceled","canceled"),
                    labels=c("not canceled","canceled"),
                    # Set color for bars
                    values=c("lightblue","pink"))+
  theme(plot.title=element_text(hjust=0.5),
```

```
    # Set background
    panel.grid.major=element_line(color="grey60",linewidth=0.25))
```



Analysis: As shown in the above figure, there are significant differences in the number of customers from different countries, making it difficult to display details in one image. Therefore, we selected the top 10 countries with the highest number of customers to observe customer distribution and cancellation numbers. Although Portugal has the largest number of customers, it also has a high cancellation rate, with a cancellation rate greater than 50%. Among the countries with the second to tenth highest number of customers, although there are also some cancellations, the proportion of cancellations is relatively small compared to Portugal.

Create an order for month, for latter visualization

```
month_in_order <- c("January", "February", "March", "April", "May", "June",
                    "July", "August", "September", "October", "November", "December")
```

Calculate the average number of nights the guests stayed per month

```
stay_nights_month <- df%>%
  # Group data by two attributes
```

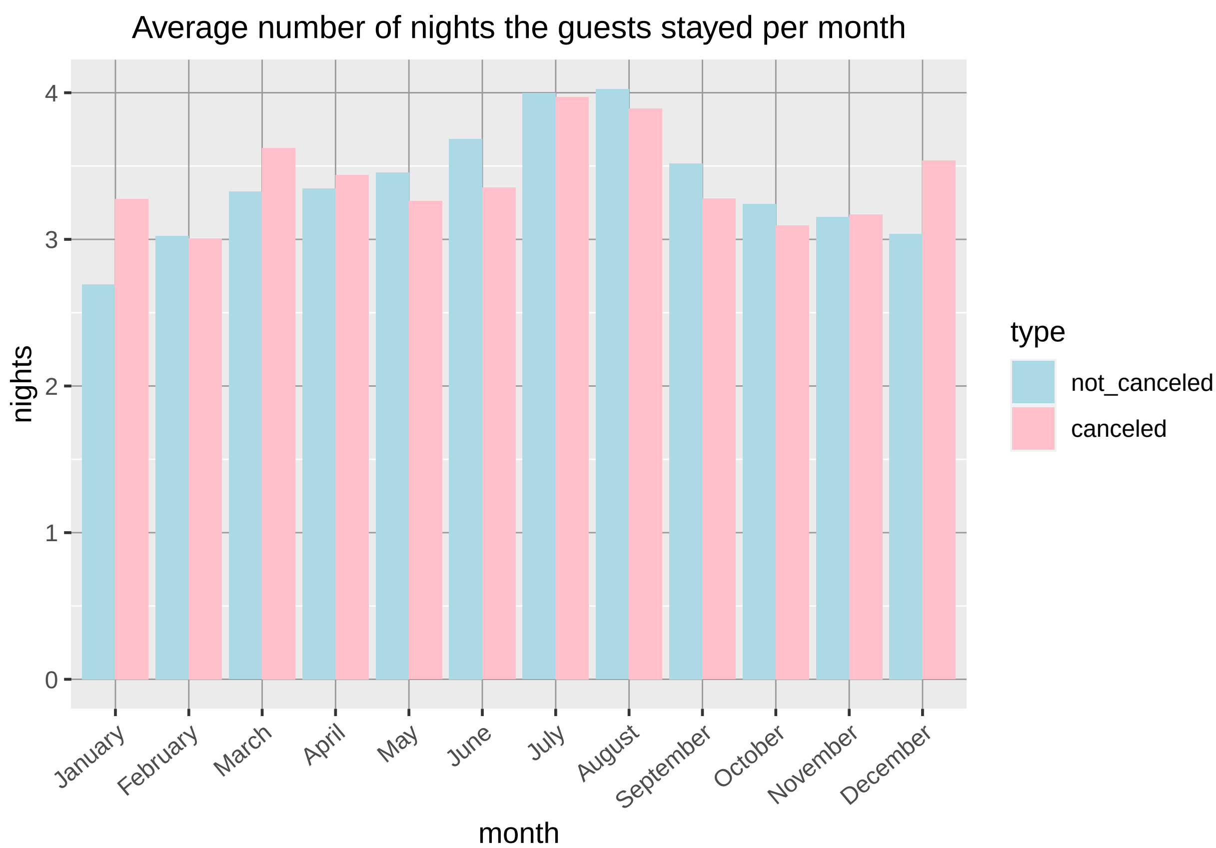
```r
  group_by(arrival_date_month,is_canceled)%>%
  # Calculate the result
  summarise(nights=sum(stays_in_weekend_nights+stays_in_week_nights)/n(),
            .groups="drop_last")%>%
  # Change the order of arrival_date_month
  mutate(arrival_date_month=factor(arrival_date_month,level=month_in_order))%>%
  # Let the result show in the order of month
  arrange(is_canceled,arrival_date_month)

# Print dataset
print(stay_nights_month)
```

```
## # A tibble: 24 x 3
## # Groups:   arrival_date_month [12]
##    arrival_date_month is_canceled nights
##    <fct>                    <int>  <dbl>
##  1 January                      0   2.69
##  2 February                     0   3.02
##  3 March                        0   3.33
##  4 April                        0   3.35
##  5 May                          0   3.46
##  6 June                         0   3.68
##  7 July                         0   4.00
##  8 August                       0   4.02
##  9 September                    0   3.52
## 10 October                      0   3.24
## # i 14 more rows
```

Visualize the average number of nights the guests stayed per month

```r
# Plot
ggplot(data=stay_nights_month,mapping=aes(x=arrival_date_month,y=nights,
                                          fill=as.factor(is_canceled)))+
  geom_bar(stat="identity",position="dodge")+
  # Set x label and title
  labs(x="month",title="Average number of nights the guests stayed per month")+
  # Reset legend for better understanding
  scale_fill_manual(name="type",
                    breaks=c("0","1"),
                    labels=c("not_canceled","canceled"),
                    # Set color for bars
                    values=c("lightblue","pink"))+
  # Change text size and angle
  theme(plot.title=element_text(size=12,hjust=0.5),
        axis.text.x=element_text(angle=40,hjust=1),
        # Set backgraound
        panel.grid.major=element_line(color="grey60",linewidth=0.25))
```



Analysis: We arranged the data months in order and calculated the average number of nights spent in hotels per month for customers who did not cancel their orders, and also for those who cancelled their orders.

From the graph, the average number of days a customer stays in a hotel per month fluctuates with each month, whether they have cancelled or not. Among them, the average number of nights for both canceled and not canceled from July to August is the highest, around 4 days. From the graph, it can be observed that the average length of stay in the hotel is about 3.5 days.

Customers who have not cancelled have a stable trend of first increasing and then decreasing in their stay days from January to August and August to December, while customers who have cancelled their stay have a higher average stay days in January, March, July, August, and December.

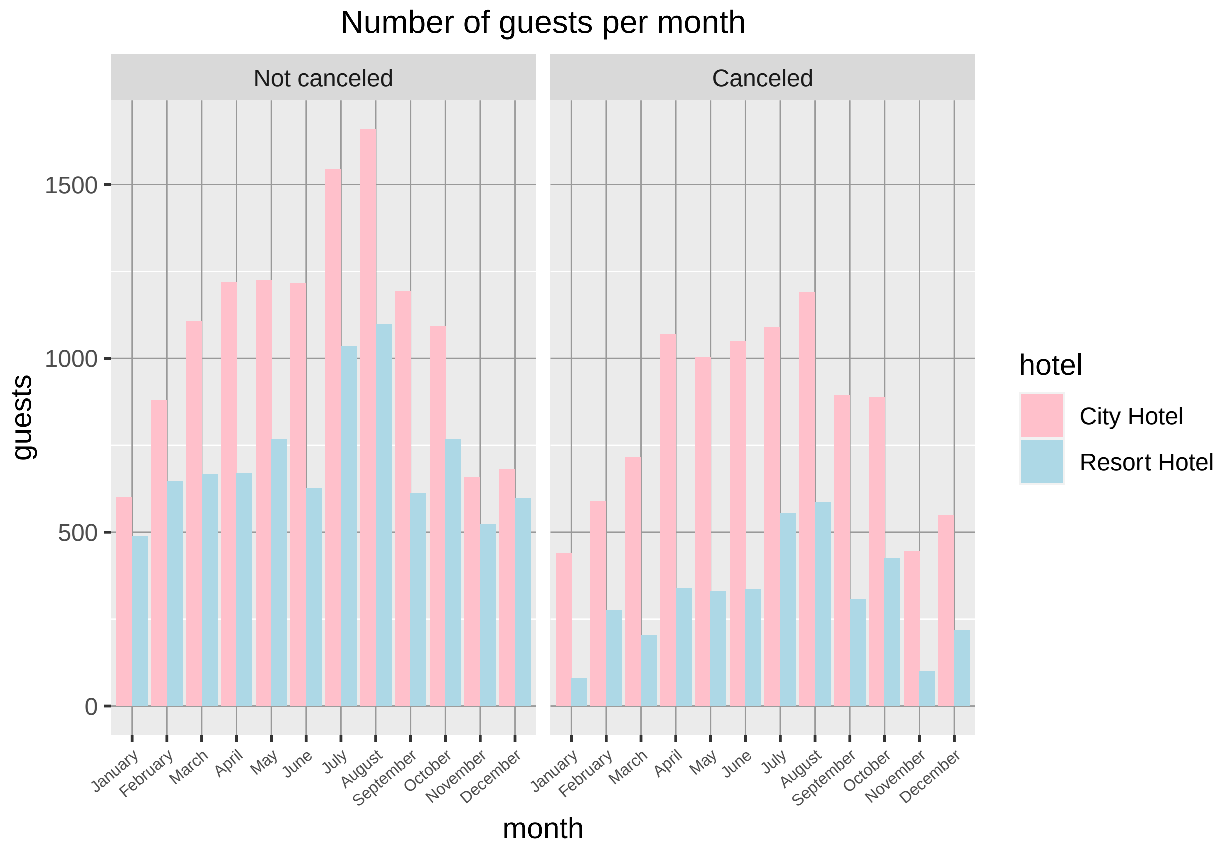
Calculate the number of guests per month for both Resort Hotel and City Hotel

```
guests_per_month <- df%>%
  # Group data by 3 attributes
  group_by(hotel,arrival_date_month,is_canceled)%>%
  # Calculate the number of guests
  summarise(guests=sum(adults+children+babies),.groups = "drop_last")%>%
  # Change the order of arrival_date_month
  mutate(arrival_date_month=factor(arrival_date_month,level=month_in_order))%>%
  # Arrange the result in an order
  arrange(is_canceled,arrival_date_month,hotel)
print(guests_per_month)
```

```
## # A tibble: 48 x 4
## # Groups:   hotel, arrival_date_month [24]
##    hotel        arrival_date_month is_canceled guests
##    <chr>        <fct>                    <int>  <dbl>
##  1 City Hotel   January                      0    600
##  2 Resort Hotel January                      0    490
##  3 City Hotel   February                     0    881
##  4 Resort Hotel February                     0    646
##  5 City Hotel   March                        0   1108
##  6 Resort Hotel March                        0    668
##  7 City Hotel   April                        0   1219
##  8 Resort Hotel April                        0    670
##  9 City Hotel   May                          0   1226
## 10 Resort Hotel May                          0    767
## # i 38 more rows
```

Visualize the number of guests per month for both Resort Hotel and City Hotel

```r
# Create labels for changing the sub titles
labels <- c("0"="Not canceled","1"="Canceled")
# Plot
ggplot(data=guests_per_month,mapping=aes(x=arrival_date_month,y=guests,fill=hotel))+
  geom_bar(stat="identity",position="dodge")+
  # Create sub plots
  facet_grid(~as.factor(is_canceled),labeller=as_labeller(labels))+
  # Set x label and title
  labs(x="month",title="Number of guests per month")+
  # Set color for bars
  scale_fill_manual(values=c("pink","lightblue"))+
  # Change text size and angle
  theme(plot.title=element_text(size=12,hjust=0.5),
        axis.text.x=element_text(angle=40,hjust=1,size=6),
        # Set background
        panel.grid.major=element_line(color="grey60",linewidth=0.25))
```



Analysis: In order to calculate the average number of guests per month, we divided customers into cancelled orders and non cancelled orders for observation. It can be observed that there is a similar trend in the number of customers who have not cancelled orders and who have cancelled orders every month, both showed an upward trend from January to August, while the number of customers checking in from August to December showed a downward trend.

By further dividing the images into City hotels and Resort hotels for observation, it can be observed that the average number of people staying at City hotels exceeds that of Resort hotels per month.

We speculate that the number of people staying at different hotels each month has the same trend, and there is a certain correlation with the month.

Calculate the average hotel price (adr) of each month for both Resort Hotel and City Hotel

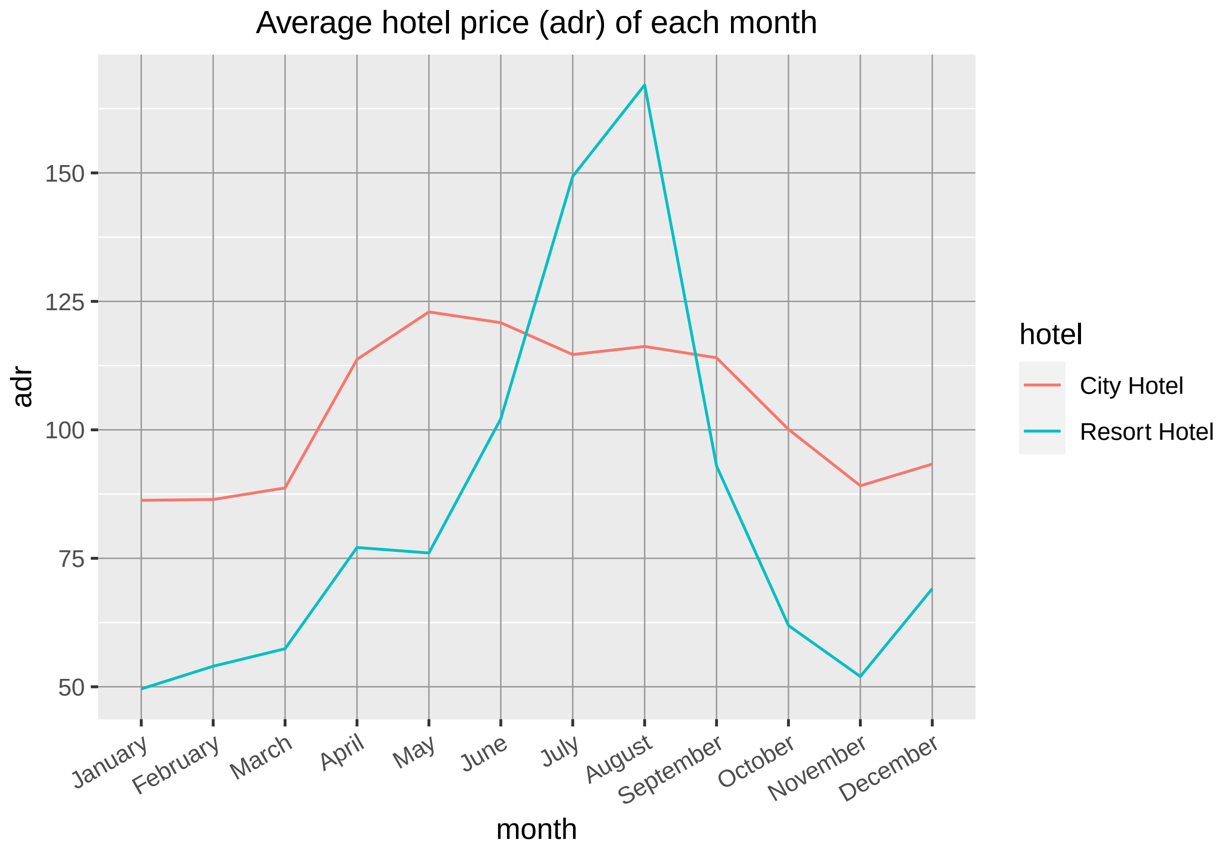
```
# Get data of all uncancelled orders
adr_per_month <- df[which(df$is_canceled=="0"),]%>%
  # Group data by hotel name and arrival month
  group_by(hotel,arrival_date_month)%>%
  # Calculate adr per month for each hotel
  summarise(price=sum(adr*(stays_in_weekend_nights+stays_in_week_nights))
            /sum(stays_in_weekend_nights+stays_in_week_nights),.groups="drop_last")%>%
  # Change the order of arrival_date_month
  mutate(arrival_date_month=factor(arrival_date_month,level=month_in_order))%>%
  # Let the result show in the order of month and hotel
```

```
  arrange(arrival_date_month,hotel)
print(adr_per_month)
```

```
## # A tibble: 24 x 3
## # Groups:   hotel [2]
##    hotel        arrival_date_month price
##    <chr>        <fct>              <dbl>
##  1 City Hotel   January             86.3
##  2 Resort Hotel January             49.6
##  3 City Hotel   February            86.4
##  4 Resort Hotel February            54.0
##  5 City Hotel   March               88.7
##  6 Resort Hotel March               57.4
##  7 City Hotel   April              114.
##  8 Resort Hotel April               77.1
##  9 City Hotel   May                123.
## 10 Resort Hotel May                 76.0
## # i 14 more rows
```

Visualize the average hotel price (adr) of each month for both Resort Hotel and City Hotel

```
# Plot
ggplot(data=adr_per_month,mapping=aes(x=arrival_date_month,y=price,color=hotel,group=hotel))+
  geom_line()+
  # Set x label and title
  labs(x="month",y="adr",title="Average hotel price (adr) of each month")+
  # Set color for bars
  #scale_color_manual(values=c("pink","lightblue"))+
  # Change text size and background
  theme(plot.title=element_text(size=12,hjust=0.5),
        axis.text.x=element_text(angle=30,hjust=1),
        panel.grid.major=element_line(color="grey60",linewidth=0.25))
```



Analysis: From the graph, it can be seen that the average monthly price of City hotel is relatively stable, with a steady upward trend from January to May. However, after June, although the price fluctuated slightly, the overall trend showed a downward trend. The average price reached its highest level in May, around 125 yuan per night.

The prices of the Resort Hotel showed a clear upward trend from January to August, especially with a significant increase in prices from July to August, and gradually decreasing after that.

Compared to City Hotel and Resort Hotel, it can be seen that except for July and August, the prices of Resort Hotel are significantly higher than those of City Hotel, while the prices of Resort are lower at other times.

From the graph, we speculate that there is a certain correlation between the average price of hotels and the month.

The detailed and descriptive results of analysis are shown beneath each graph, here are some general analysis conclusion.

1. The majority of customers come from European countries, and you can focus on observing customer profiles of European customers, such as behavioral habits, stay days, etc. This helps the hotel identify potential customers who may cancel orders.

2. Hotels can analyze customer behavior to understand why the countries with the highest proportion of customers have higher room cancellations, in order to develop appropriate strategies to retain potential customers.

3. From July to August, customers have the highest average number of nights staying in hotels, with the highest average hotel prices per month. It can also be observed that the number of customers who cancel orders during this time period is also the highest. If some means can be used to retain customers, the hotel will gain greater profits.
4. We consider July to August as the peak season for tourism, as the hotel has the highest number of guests. Due to the lower monthly average price of City Hotel compared to Resort Hotel, the number of people staying at City Hotel is much higher than that at Resort Hotel. However, the cancellation rate of City Hotel is also higher than that of Resort Hotel, so City Hotel can analyze the specific reasons for high occupancy and cancellation rates.

Data Pre-processing

Check the missing values

```
df[df == ""] <- NA
colSums(is.na(df))
```

```
##                          hotel                    is_canceled
##                              0                              0
##                      lead_time              arrival_date_year
##                              0                              0
##             arrival_date_month       arrival_date_week_number
##                              0                              0
##      arrival_date_day_of_month        stays_in_weekend_nights
##                              0                              0
##           stays_in_week_nights                         adults
##                              0                              0
##                       children                         babies
##                              0                              0
##                           meal                        country
##                              0                             92
##                 market_segment           distribution_channel
##                              0                              0
##              is_repeated_guest         previous_cancellations
##                              0                              0
## previous_bookings_not_canceled             reserved_room_type
##                              0                              0
##             assigned_room_type                booking_changes
##                              0                              0
##                   deposit_type                          agent
##                              0                           2489
##                        company           days_in_waiting_list
##                          16881                              0
##                  customer_type                            adr
##                              0                              0
##    required_car_parking_spaces      total_of_special_requests
##                              0                              0
```

Handle the missing values.

Firstly, it can be observed from the above statistics that there are only three attributes with missing values, namely country, agent, and company. Therefore, we have different processing methods for these different attributes.

Regarding country, we found that the proportion of null values in the dataset is very small, only 92/17908(0.51%). So, we have decided to directly delete these missing values. After removing the null value for country, there are still 17816 pieces of data in the dataset.

```r
# Remove countries with missing values
df <- subset(df, !is.na(country))
```

About agents and companies. In the dataset paper "Hotel Demand Dataset" [1], it was described that there are no NULL values for agent and company, as their NULL values represent a special category that does not come from agent and company reservations. So, we plan to fill in the empty values of these companies and agents into a new category of 0.

```r
# The missing values of agent and company are filled to 0
df$agent <- ifelse(is.na(df$agent), 0, df$agent)
df$company <- ifelse(is.na(df$company), 0, df$company)
```

Check duplicates and remove

Observing the dataset and according to the given variable description.txt file, we found that in the attribute 'meal', SC and Undefined represent the same category, both referring to 'no meal package'. Therefore, we decided to merge these two categories into one and check for redundant data.

```r
# Replace Undefined with SC
df$meal <- ifelse(df$meal == "Undefined", "SC", df$meal)
```

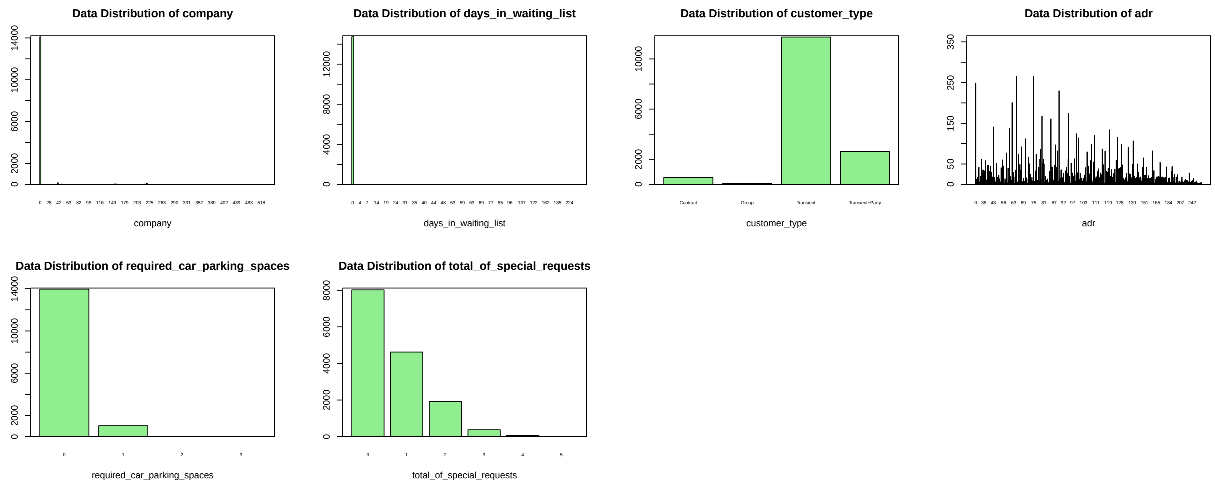
We found a total of 2822 redundant dayta in the dataset, with 14994 data remaining after deletion.

```r
# remove duplicates
df <- unique(df)
```

Data distribution

```r
par(mfrow = c(1, 4))
variables <- colnames(df)
for (variable in variables) {
  x <- df[[variable]]
  barplot(table(x),
          xlab = variable, col = "lightgreen",
          ylim = c(0, max(table(x)) + 100),
          main = paste("Data Distribution of", variable),
          cex.names = 0.5
  )
  box()
}
```


Data Distribution of hotel
hotel
Data Distribution of is_canceled
is_canceled
Data Distribution of lead_time
lead_time
Data Distribution of arrival_date_year
arrival_date_year
Data Distribution of arrival_date_month
arrival_date_month
Data Distribution of arrival_date_week_number
arrival_date_week_number
Data Distribution of arrival_date_day_of_month
arrival_date_day_of_month
Data Distribution of stays_in_weekend_nights
stays_in_weekend_nights
Data Distribution of stays_in_week_nights
stays_in_week_nights
Data Distribution of adults
adults
Data Distribution of children
children
Data Distribution of babies
babies
Data Distribution of meal
meal
Data Distribution of country
country
Data Distribution of market_segment
market_segment
Data Distribution of distribution_channel
distribution_channel
Data Distribution of is_repeated_guest
is_repeated_guest
Data Distribution of previous_cancellations
previous_cancellations
Data Distribution of previous_bookings_not_cancelec
previous_bookings_not_canceled
Data Distribution of reserved_room_type
reserved_room_type
Data Distribution of assigned_room_type
assigned_room_type
Data Distribution of booking_changes
booking_changes
Data Distribution of deposit_type
deposit_type
Data Distribution of agent
agent



Check and remove outliers

When dealing with outliers, we only observe and process numerical type attributes. After observation and consideration, we ultimately believe that the attributes that may have outliers are adults, children, babies, booking_changes and adr. The following will provide a detailed explanation of our judgment method and processing results.

Based on the data distribution graph drawn above, we can observe that the distribution of adults is mostly between 0-3 people. Based on the data distribution results and common sense, we believe that 0 adult hotel rooms with 5 or more people in one room are abnormal. So we decided to delete these situations. Based on the observation of the data distribution map of children and babies, we believe that although it is normal for the number of children and babies to be 0, it is abnormal for children or babies with more than 5 rooms to be occupied in one room. Also, we consider that for people who kept changing his/her reservation for more than 5 times are also abnormal according to the common sense. Therefore, such results are considered as outliers for deletion.

These commons sense for deletion also been checked by the data distribution plot, and those points that considered as ouliers are obviously away from most points.

The final amount of data to be deleted is 76, leaving 14918 rows of data remaining.

```r
# Remove outliers of adults, children, babies, booking changes
df <- df[!(df$adults == 0 | df$adults >= 5), ]
df <- df[!(df$children >= 5), ]
df <- df[!(df$babies >= 5), ]
df <- df[!(df$booking_changes > 5), ]
```

The following is an analysis of ADR outliers. Due to the fact that the price of the hotel is 0, which is not reasonable data based on common sense, we will delete it first.

```r
# Outliers of adr
# Remove inconsistent data
df <- df[df$adr != 0, ]
```

From T1-5, it can be seen that there is a certain price difference between City Hotel and Resort Hotel. In July August, the prices of Resort Hotel were significantly higher than those of City Hotel, while in other months they were the opposite. Therefore, we believe that the price difference between City Hotel and Resort

Hotel should also be considered when selecting outliers. So when analyzing the outliers of ADR, we decided to have a group discussion.

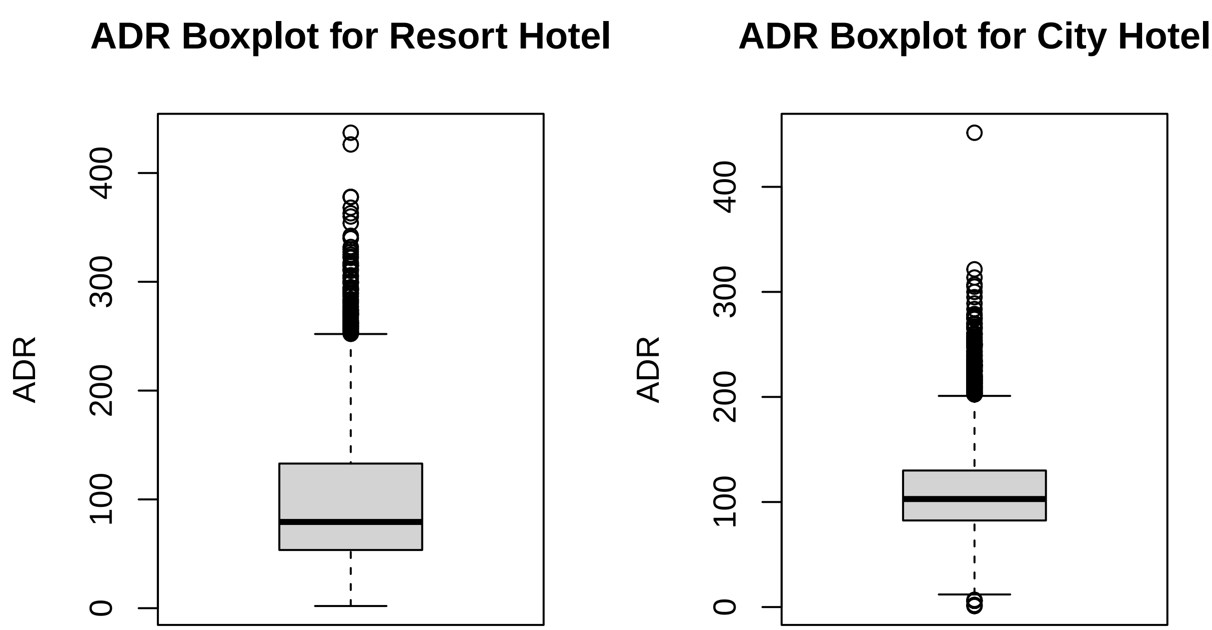
When detecting outliers, we considered the z-score method and the boxplot method. Due to the fact that the z-score detection of outliers requires a rough normal distribution of the data, and it was observed through the data distribution map that ADR does not meet this condition, we ultimately chose the box plot method. Then we draw the box diagram of ADR for two hotels separately. For data outside the upper and lower limits of the box plot, delete between them.

Ultimately, the Resort Hotel is considered to have 131 outliers, while the City Hotel is considered to have 222 outliers. After removing the outliers, there are still 14347 pieces of data left.

```
# Get Hotel type
hotel_types <- unique(df$hotel)

par(mfrow = c(1, 2))

# Box plot of different hotel types
for (hotel_type in hotel_types) {
  adr_df <- df$adr[df$hotel == hotel_type]
  boxplot(adr_df,
          ylab = "ADR",
          main = paste("ADR Boxplot for", hotel_type))
}
```



```r
# Create data frame
outliers_df <- data.frame()

# Calculate outliers for adr
for (hotel_type in hotel_types) {
  hotel_df <- df[df$hotel == hotel_type, ]

  Q1 <- quantile(hotel_df$adr, 0.25)
  Q3 <- quantile(hotel_df$adr, 0.75)
  IQR <- Q3 - Q1
  # The threshold is set to 1.5
  Minimum <- Q1 - 1.5 * IQR
  Maximum <- Q3 + 1.5 * IQR

  # Judge outliers
  outliers <- which(hotel_df$adr < Minimum | hotel_df$adr > Maximum)
  cat("The number of outliers for", hotel_type, ":", length(hotel_df$adr[outliers]), "\n")
  # Outliers are stored in the data frame
  outliers_df <- rbind(outliers_df, hotel_df[outliers, ])
}
```

```
## The number of outliers for Resort Hotel : 131
## The number of outliers for City Hotel : 222
```

```r
# Remove adr's outliers
df <- df[!(rownames(df) %in% rownames(outliers_df)), ]
```

data normalization

When we do data normalization, we only handle numerical data so we should firstly select the columns that need to be processed. That is to say, we need to delete data of Boolean type and char type.

It is worth noting that despite the company, agent, array_date_year, annual_date_ week_number, array_date_day_of_month is considered numerical data, but in reality they represent data of categorical or ordinal, so we do not consider normalizing this type of data.

```r
# Filter normalized data
# Filter data of numeric type
numeric_variables <- df[, sapply(df, is.numeric)]

# Filter binary data
non_binary_variables <- numeric_variables[, sapply(numeric_variables,
                                                   function(col) length(unique(col)) > 2)]

# Filter data that have no practical significance
insignificance_values <- c("arrival_date_year", "arrival_date_week_number",
                           "arrival_date_day_of_month","agent","company")
normalized_variables <- non_binary_variables[!(names(non_binary_variables) %in%
                                               insignificance_values)]
```

For data with a small range of values, we believe that they exist within a reasonable range that satisfies the model's scale range for input data, so we do not intend to standardize them. Based on our analysis of the dataset, we plan to set this value threshold at 10.

```r
# Filter data that is less than 10
max_values <- sapply(normalized_variables, max)
normalized_variables <- normalized_variables[, max_values > 10]
```

Having done the column filter process, we could check the the columns that are needed to do normalization.

```r
# check data that should do normalization
colnames(normalized_variables)
```

```
## [1] "lead_time"                      "stays_in_weekend_nights"
## [3] "stays_in_week_nights"           "previous_cancellations"
## [5] "previous_bookings_not_canceled" "days_in_waiting_list"
## [7] "adr"
```

We found that there are two commonly used data normalization methods, namely z-score and max min methods. We believe that each normalization method has some rationality, but here we choose to use the max-min normalization method for the following specific reasons.

1) Due to the handling of outliers in the previous steps, theoretically, there should be no obvious outliers or deviations in the data.
2) In addition, we did not find any dataset that satisfies a significant normal distribution or approximates a normal distribution.

So, we believe that using the max-min method here is more appropriate.

```r
# min-max
min_max <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

# min-max normalization
normalized_df <- as.data.frame(lapply(normalized_variables, min_max))
# Filter common column names and merge data
common_variables <- intersect(colnames(df), colnames(normalized_df))
df[, common_variables] <- normalized_df[, common_variables]
```

Encode categorical values

Firstly, we select the attributes that need to be encoded, which are char type data, years and those that represent for IDs. There are a total of 12 attributes that need to be processed. We classify and encode these attributes based on the size and order of different variable categories.

```r
# Gain variables of char type
character_variables <- which(sapply(df, is.character))

# Convert the char type to the factor type
for (variable in character_variables) {
  df[, variable] <- as.factor(df[, variable])
}
```

```
# The number of variables of char type
unique_counts <- data.frame(Column = character(), Count = numeric())
for (variable in character_variables) {
  variable_name <- names(df)[variable]
  unique_count <- length(unique(df[, variable]))
  unique_counts <- rbind(unique_counts, data.frame(Column = variable_name,
                                                   Count = unique_count))
}

print(unique_counts)
```

```
##                   Column Count
## 1                  hotel     2
## 2     arrival_date_month    12
## 3                   meal     4
## 4                country   116
## 5         market_segment     7
## 6   distribution_channel     4
## 7     reserved_room_type     8
## 8     assigned_room_type    10
## 9           deposit_type     3
## 10         customer_type     4
```

For attributes of char types with a certain order, we choose to encode them using ordinal encoding, which is array_date_month.

```
# Ordinal Encoding
factor_month <- factor(df$arrival_date_month, order = TRUE,
                       levels = c("January", "February", "March",
                                  "April", "May", "June","July",
                                  "August", "September", "October",
                                  "November", "December"))

encode_month <- as.numeric(factor_month)
df$arrival_date_month <- encode_month
```

For attributes without ordinal order, we classify them into label encoding and one hot encoding. First of all, country, reserved_room_type and assigned_room_type, each attribute has many categories, for example, country has 116 categories, reserved_room_type has 8 categories, etc. If they are subjected to one hot encoding, the dimensionality of the data may be high, which may have a certain impact on the prediction results, which also have high computation cost. Therefore, for this type of data, we choose to use label encoding to avoid generating high-dimensional data.

```
# Label Encoding
variables <- c("country","reserved_room_type", "assigned_room_type","agent","company")
for (encode_variable in variables) {
  df[[encode_variable]] <- as.integer(df[[encode_variable]])
}
```

For the remaining attributes, as their categories are not so many and there is no ordinal order, we have decided to directly adopt one hot encoding.

```r
# One-Hot Encoding
# Gain the variables' name of type factor
library(mltools)
```

```
##
## Attaching package: 'mltools'

## The following object is masked from 'package:tidyr':
##
##     replace_na
```

```r
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```r
factor_variables <- names(df[sapply(df, is.factor)])
print(factor_variables)
```

```
## [1] "hotel"                "meal"                 "market_segment"
## [4] "distribution_channel" "deposit_type"         "customer_type"
```

```r
for (encode_variable in factor_variables) {
  df <- one_hot(as.data.table(df), cols = encode_variable)
}
```

Since the company and agent here are IDs not numerical data, we believe that label encoding should also be performed.

```r
df$agent <- as.factor(df$agent)
df$company <- as.factor(df$company)
```

Finally, years should also be considered for encoding. We map different years onto a numerical label, and the final result is as follows.

```r
# encode year by mapping
df$arrival_date_year <- as.integer(factor(df$arrival_date_year,
                                          levels = unique(df$arrival_date_year)))
```

Store the preprocessed dataset

Finally, we store the processed data into a new_dataset.csv file, which contains 14374 rows and 48 columns.

```r
write.csv(df, file = "new_dataset.csv", row.names = FALSE)
```

References

[1] N. Antonio, A. Almeida, L. Nunes, Hotel booking demand datasets. Data in Brief. 2019;22(41-49):41-49. doi:10.1016/j.dib.2018.11.126