HOTEL BOOKING DEMAND

UNIVERSITY OF COLORADO DENVER

Anamika Singh, Anurag Lahon, Rubina Shaik

Appendix

Project Overview

In this project, we will try to predict the possibility of a booking cancellations for a hotel based on different factors and also try to predict if there is a likelihood of getting a special requests from customers based on different factors. The data set contains booking information for a city hotel and a resort hotel, and includes information such as when the booking was made, the number of adults, children, and/or babies, and the number of available parking spaces, among other things. From this, we can understand the customer's behavior and it might help us to take better decisions.

The process of our analysis will be: Understanding the Datasets, Data preparation and wrangling, Analyzing, and visualizing the data, Model building, comparing the model, and finally selecting the best model.

Our goal is to predict what type of customers need special request and to predict the possibility of booking cancellations by analyzing all the factors that can influence booking cancellations. We also performed time series analysis to forecast the number of bookings using Holtwinters and ARIMA model.

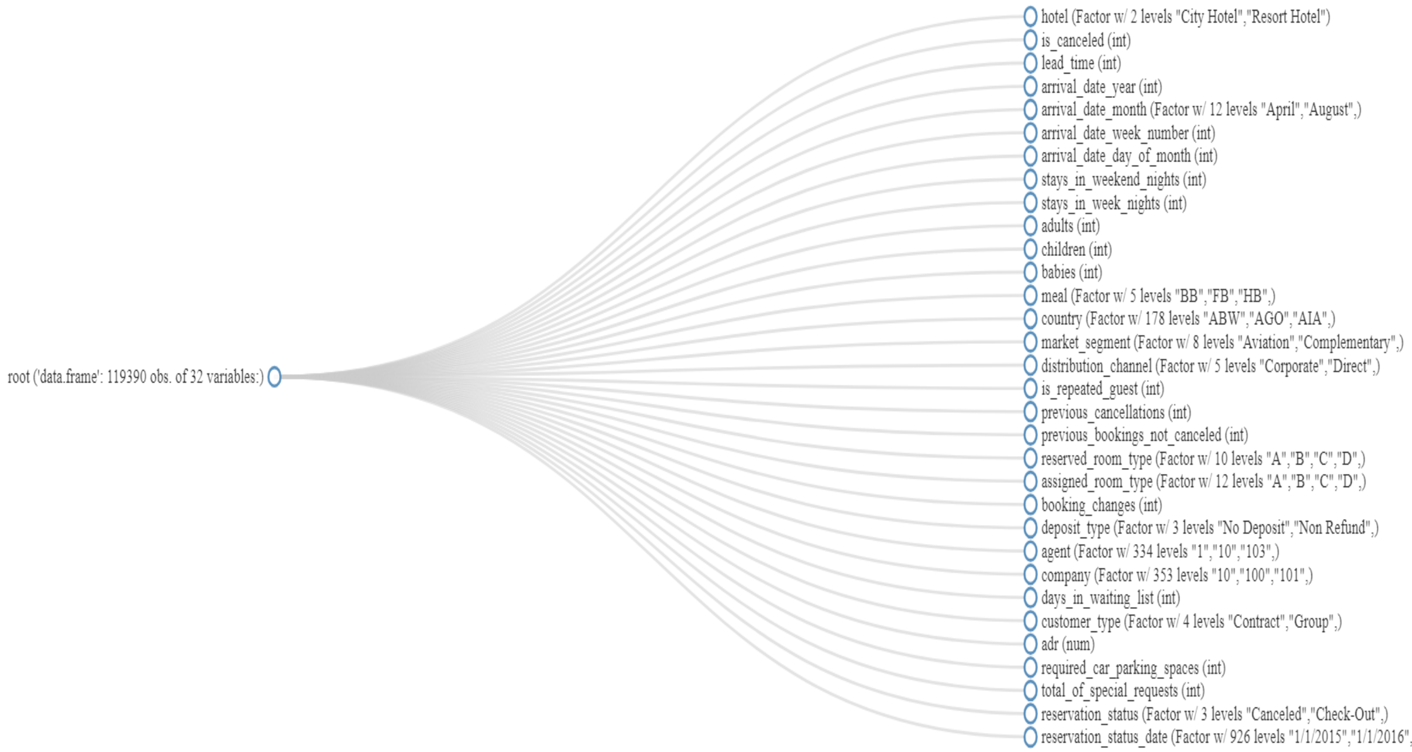
Data Source

Data was collected from the Kaggle website. The data is originally from the article Hotel Booking Demand Datasets, written by Nuno Antonio, Ana Almeida, and Luis Nunes for Data in Brief, Volume 22, February 2019. This data set contains booking information for a city hotel and a resort hotel, and includes information such as when the booking was made, length of stay, the number of adults, children, and/or babies, and the number of available parking spaces, among other things.

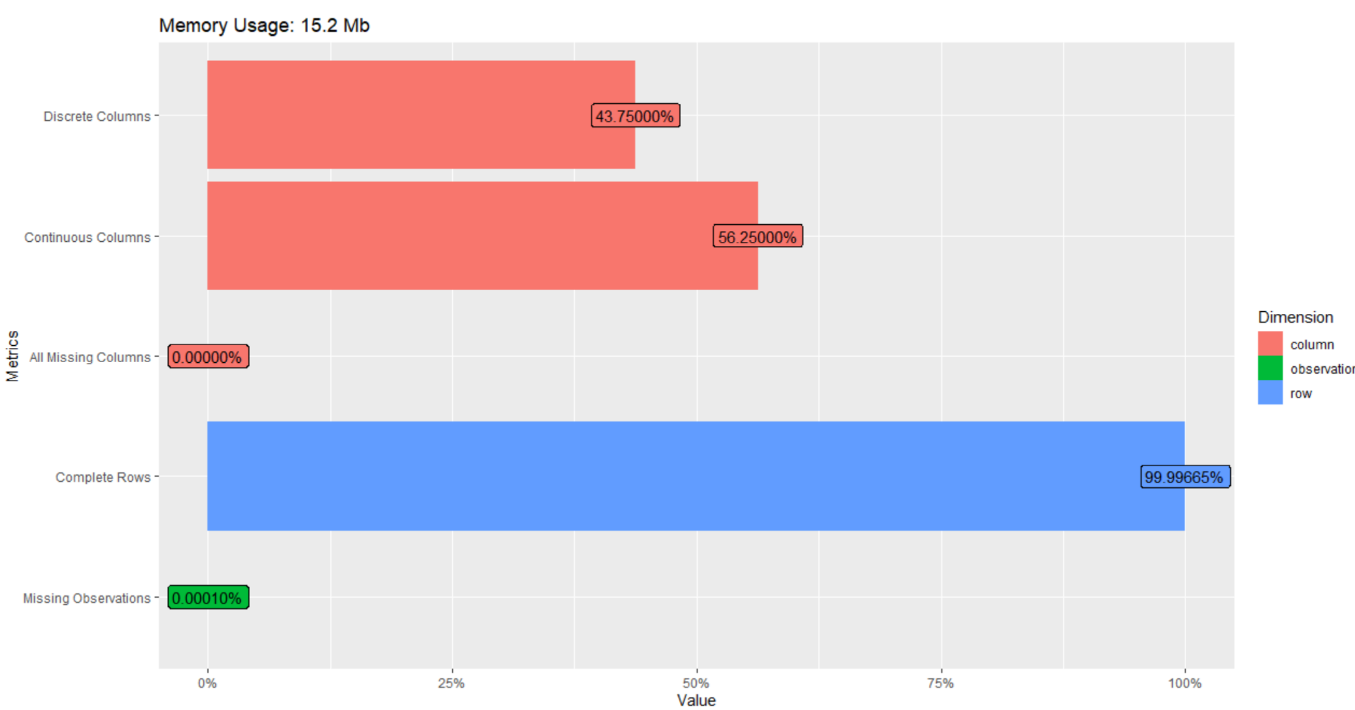
https://www.kaggle.com/jessemostipak/hotel-booking-demand

Data Understanding

The data set contains 119390 rows and 32 columns.



```
> #Structure of the hotel data set
> str(hotels)
'data.frame':	119390 obs. of  32 variables:
 $ hotel                         : Factor w/ 2 levels "City Hotel","Resort Hotel": 2 2 2 2 2 2 2 2 2 2 ...
 $ is_canceled                   : int  0 0 0 0 0 0 0 0 1 1 ...
 $ lead_time                     : int  342 737 7 13 14 14 0 9 85 75 ...
 $ arrival_date_year             : int  2015 2015 2015 2015 2015 2015 2015 2015 2015 2015 ...
 $ arrival_date_month            : Factor w/ 12 levels "April","August",..: 6 6 6 6 6 6 6 6 6 6 ...
 $ arrival_date_week_number      : int  27 27 27 27 27 27 27 27 27 27 ...
 $ arrival_date_day_of_month     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ stays_in_weekend_nights       : int  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 ...
 $ adults                        : int  2 2 1 1 2 2 2 2 2 2 ...
 $ children                      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ babies                        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ meal                          : Factor w/ 5 levels "BB","FB","HB",..: 1 1 1 1 1 1 1 2 1 3 ...
 $ country                       : Factor w/ 178 levels "ABW","AGO","AIA",..: 137 137 60 60 60 60 137 137 137 137 ...
 $ market_segment                : Factor w/ 8 levels "Aviation","Complementary",..: 4 4 4 3 7 7 4 4 7 6 ...
 $ distribution_channel          : Factor w/ 5 levels "Corporate","Direct",..: 2 2 2 1 4 4 2 2 4 4 ...
 $ 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            : Factor w/ 10 levels "A","B","C","D",..: 3 3 1 1 1 1 3 3 1 4 ...
 $ assigned_room_type            : Factor w/ 12 levels "A","B","C","D",..: 3 3 3 1 1 1 3 3 1 4 ...
 $ booking_changes               : int  3 4 0 0 0 0 0 0 0 0 ...
 $ deposit_type                  : Factor w/ 3 levels "No Deposit","Non Refund",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ agent                         : Factor w/ 334 levels "1","10","103",..: 334 334 334 157 103 103 334 156 103 40 ...
 $ company                       : Factor w/ 353 levels "10","100","101",..: 353 353 353 353 353 353 353 353 353 353 ...
 $ days_in_waiting_list          : int  0 0 0 0 0 0 0 0 0 0 ...
 $ customer_type                 : Factor w/ 4 levels "Contract","Group",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ adr                           : num  0 0 75 75 98 ...
 $ required_car_parking_spaces   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ total_of_special_requests     : int  0 0 0 0 1 1 0 1 1 0 ...
 $ reservation_status            : Factor w/ 3 levels "Canceled","Check-Out",..: 2 2 2 2 2 2 2 2 1 1 ...
 $ reservation_status_date       : Factor w/ 926 levels "2014-10-17","2014-11-18",..: 122 122 123 123 124 124 124 124 73 62 ...
>
```



The dataset is described below.

```
hotels

 32  Variables      119390  Observations
--------------------------------------------------------------------------------
hotel
       n  missing distinct 
  119390        0        2 

Value        City Hotel Resort Hotel
Frequency         79330        40060
Proportion        0.664        0.336
--------------------------------------------------------------------------------
is_canceled
       n  missing distinct     Info      Sum     Mean      Gmd 
  119390        0        2      0.7    44224   0.3704   0.4664 


--------------------------------------------------------------------------------


lead_time
       n  missing distinct     Info     Mean      Gmd      .05      .10      .25      .50      .75 
  119390        0      479        1      104    112.5        0        3       18       69      160 
     .90      .95 
     265      320 

lowest :   0   1   2   3   4, highest: 622 626 629 709 737
--------------------------------------------------------------------------------
arrival_date_year
       n  missing distinct     Info     Mean      Gmd 
  119390        0        3    0.847     2016   0.7499 

Value       2015  2016  2017
Frequency  21996 56707 40687
Proportion 0.184 0.475 0.341
--------------------------------------------------------------------------------
arrival_date_month
       n  missing distinct 
  119390        0       12 

lowest : April     August    December  February  January  , highest: March     May       November  October   Septemb
er

Value          April    August  December  February   January      July      June     March       May
Frequency      11089     13877      6780      8068      5929     12661     10939      9794     11791
Proportion     0.093     0.116     0.057     0.068     0.050     0.106     0.092     0.082     0.099

Value       November   October September
Frequency       6794     11160     10508
Proportion     0.057     0.093     0.088
--------------------------------------------------------------------------------
```

```
arrival_date_week_number
       n  missing distinct     Info     Mean      Gmd      .05      .10      .25      .50      .75
  119390        0       53        1    27.17    15.68        5        8       16       28       38
     .90      .95
      46       49

lowest :  1  2  3  4  5, highest: 49 50 51 52 53
---------------------------------------------------------------------------------------------------
arrival_date_day_of_month
       n  missing distinct     Info     Mean      Gmd      .05      .10      .25      .50      .75
  119390        0       31    0.999     15.8    10.13        2        4        8       16       23
     .90      .95
      28       30

lowest :  1  2  3  4  5, highest: 27 28 29 30 31
---------------------------------------------------------------------------------------------------



stays_in_weekend_nights
       n  missing distinct     Info     Mean      Gmd      .05      .10      .25      .50      .75
  119390        0       17    0.879   0.9276    1.026        0        0        0        1        2
     .90      .95
       2        2

lowest :  0  1  2  3  4, highest: 13 14 16 18 19

Value          0     1     2     3     4     5     6     7     8     9    10    12    13    14    16
Frequency  51998 30626 33308  1259  1855    79   153    19    60    11     7     5     3     2     3
Proportion 0.436 0.257 0.279 0.011 0.016 0.001 0.001 0.000 0.001 0.000 0.000 0.000 0.000 0.000 0.000

Value         18    19
Frequency      1     1
Proportion 0.000 0.000
---------------------------------------------------------------------------------------------------
stays_in_week_nights
       n  missing distinct     Info     Mean      Gmd      .05      .10      .25      .50      .75
  119390        0       35    0.953      2.5    1.865        0        1        1        2        3
     .90      .95
       5        5

lowest :  0  1  2  3  4, highest: 35 40 41 42 50
---------------------------------------------------------------------------------------------------



adults
       n  missing distinct     Info     Mean      Gmd      .05      .10      .25      .50      .75
  119390        0       14    0.569    1.856   0.4287        1        1        2        2        2
     .90      .95
       2        3

lowest :  0  1  2  3  4, highest: 26 27 40 50 55

Value          0     1     2     3     4     5     6    10    20    26    27    40    50    55
Frequency    403 23027 89680  6202    62     2     1     1     2     5     2     1     1     1
Proportion 0.003 0.193 0.751 0.052 0.001 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
```

---

children

n	missing	distinct	Info	Mean	Gmd
119386	4	5	0.201	0.1039	0.1955

lowest : 0 1 2 3 10, highest: 0 1 2 3 10

Value	0	1	2	3	10
Frequency	110796	4861	3652	76	1
Proportion	0.928	0.041	0.031	0.001	0.000

---

babies

n	missing	distinct	Info	Mean	Gmd
119390	0	5	0.023	0.007949	0.01578

lowest : 0 1 2 9 10, highest: 0 1 2 9 10

Value	0	1	2	9	10
Frequency	118473	900	15	1	1
Proportion	0.992	0.008	0.000	0.000	0.000

---

meal

n	missing	distinct
119390	0	5

lowest : BB FB HB SC Undefined, highest: BB FB HB SC Undefined

Value	BB	FB	HB	SC	Undefined
Frequency	92310	798	14463	10650	1169
Proportion	0.773	0.007	0.121	0.089	0.010

---

country

n	missing	distinct
119390	0	178

lowest : ABW AGO AIA ALB AND, highest: VGB VNM ZAF ZMB ZWE

---

market_segment

n	missing	distinct
119390	0	8

lowest : Aviation Complementary Corporate Direct Groups
highest: Direct Groups Offline TA/TO Online TA Undefined

Value	Aviation	Complementary	Corporate	Direct	Groups	Offline TA/TO
Frequency	237	743	5295	12606	19811	24219
Proportion	0.002	0.006	0.044	0.106	0.166	0.203

Value	Online TA	Undefined
Frequency	56477	2
Proportion	0.473	0.000

```
--------------------------------------------------------------------------------------------------------
distribution_channel
       n  missing distinct 
  119390        0        5 

lowest : Corporate Direct    GDS       TA/TO     Undefined, highest: Corporate Direct    GDS       TA/TO     Undefine
d

Value      Corporate    Direct       GDS     TA/TO Undefined
Frequency       6677     14645       193     97870         5
Proportion     0.056     0.123     0.002     0.820     0.000
--------------------------------------------------------------------------------------------------------
is_repeated_guest 
       n  missing distinct     Info      Sum     Mean      Gmd 
  119390        0        2    0.093     3810  0.03191  0.06179 
--------------------------------------------------------------------------------------------------------


previous_cancellations 
       n  missing distinct     Info     Mean      Gmd      .05      .10      .25      .50      .75 
  119390        0       15    0.154  0.08712   0.1682        0        0        0        0        0 
     .90      .95 
       0        1 

lowest :  0  1  2  3  4, highest: 19 21 24 25 26

Value           0      1      2      3      4      5      6     11     13     14     19     21     24
Frequency  112906   6051    116     65     31     19     22     35     12     14     19      1     48
Proportion  0.946  0.051  0.001  0.001  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000

Value          25     26
Frequency      25     26
Proportion  0.000  0.000
--------------------------------------------------------------------------------------------------------




previous_bookings_not_canceled 
       n  missing distinct     Info     Mean      Gmd      .05      .10      .25      .50      .75 
  119390        0       73    0.088   0.1371   0.2708        0        0        0        0        0 
     .90      .95 
       0        0 

lowest :  0  1  2  3  4, highest: 68 69 70 71 72
--------------------------------------------------------------------------------------------------------
reserved_room_type 
       n  missing distinct 
  119390        0       10 

lowest : A B C D E, highest: F G H L P

Value          A     B     C     D     E     F     G     H     L     P
Frequency  85994  1118   932 19201  6535  2897  2094   601     6    12
Proportion 0.720 0.009 0.008 0.161 0.055 0.024 0.018 0.005 0.000 0.000
--------------------------------------------------------------------------------------------------------
```

```
assigned_room_type
       n  missing distinct 
  119390        0       12 

lowest : A B C D E, highest: H I K L P

Value          A     B     C     D     E     F     G     H     I     K     L     P
Frequency  74053  2163  2375 25322  7806  3751  2553   712   363   279     1    12
Proportion 0.620 0.018 0.020 0.212 0.065 0.031 0.021 0.006 0.003 0.002 0.000 0.000
---------------------------------------------------------------------------------------------------------


booking_changes 
       n  missing distinct     Info     Mean      Gmd      .05      .10      .25      .50      .75 
  119390        0       21    0.388   0.2211   0.3919        0        0        0        0        0 
     .90      .95 
       1        1 

lowest :  0  1  2  3  4, highest: 16 17 18 20 21
---------------------------------------------------------------------------------------------------------
deposit_type 
       n  missing distinct 
  119390        0        3 

Value      No Deposit Non Refund Refundable
Frequency      104641      14587        162
Proportion      0.876      0.122      0.001
---------------------------------------------------------------------------------------------------------
agent 
       n  missing distinct 
  119390        0      334 

lowest : 1    10   103  104  105 , highest: 95   96   98   99   NULL
---------------------------------------------------------------------------------------------------------
company 
       n  missing distinct 
  119390        0      353 

lowest : 10   100  101  102  103 , highest: 93   94   96   99   NULL
---------------------------------------------------------------------------------------------------------
days_in_waiting_list 
       n  missing distinct     Info     Mean      Gmd      .05      .10      .25      .50      .75 
  119390        0      128     0.09    2.321    4.559        0        0        0        0        0 
     .90      .95 
       0        0 

lowest :   0   1   2   3   4, highest: 236 259 330 379 391
---------------------------------------------------------------------------------------------------------
customer_type 
       n  missing distinct 
  119390        0        4 

Value             Contract           Group       Transient Transient-Party
Frequency             4076             577           89613           25124
Proportion           0.034           0.005           0.751           0.210
---------------------------------------------------------------------------------------------------------
```

```
adr
      n  missing distinct     Info     Mean      Gmd      .05      .10      .25      .50      .75
  119390        0     8879        1    101.8    51.91    38.40    50.00    69.29    94.58   126.00
     .90      .95
  164.00   193.50

lowest :   -6.38    0.00    0.26    0.50    1.00, highest:  450.00  451.50  508.00  510.00 5400.00

Value         0    50   100   150   200   250   300   350   400   450   500  5400
Frequency   2437 35085 50975 21915  6234  2156   463    99    19     4     2     1
Proportion 0.020 0.294 0.427 0.184 0.052 0.018 0.004 0.001 0.000 0.000 0.000 0.000

For the frequency table, variable is rounded to the nearest 50
---------------------------------------------------------------------------------------------------
required_car_parking_spaces
      n  missing distinct     Info     Mean      Gmd
  119390        0        5    0.175  0.06252   0.1173

lowest : 0 1 2 3 8, highest: 0 1 2 3 8

Value           0      1      2      3      8
Frequency  111974   7383     28      3      2
Proportion  0.938  0.062  0.000  0.000  0.000
---------------------------------------------------------------------------------------------------

total_of_special_requests
      n  missing distinct     Info     Mean      Gmd
  119390        0        6    0.773   0.5714   0.7684

lowest : 0 1 2 3 4, highest: 1 2 3 4 5

Value          0     1     2     3     4     5
Frequency  70318 33226 12969  2497   340    40
Proportion 0.589 0.278 0.109 0.021 0.003 0.000
---------------------------------------------------------------------------------------------------
reservation_status
      n  missing distinct
  119390        0        3

Value       Canceled Check-Out   No-Show
Frequency      43017     75166      1207
Proportion      0.36      0.63      0.01
---------------------------------------------------------------------------------------------------
reservation_status_date
      n  missing distinct
  119390        0      926

lowest : 1/1/2015  1/1/2016  1/1/2017  1/10/2016 1/10/2017, highest: 9/8/2016  9/8/2017  9/9/2015  9/9/2016  9/9/20
17
```

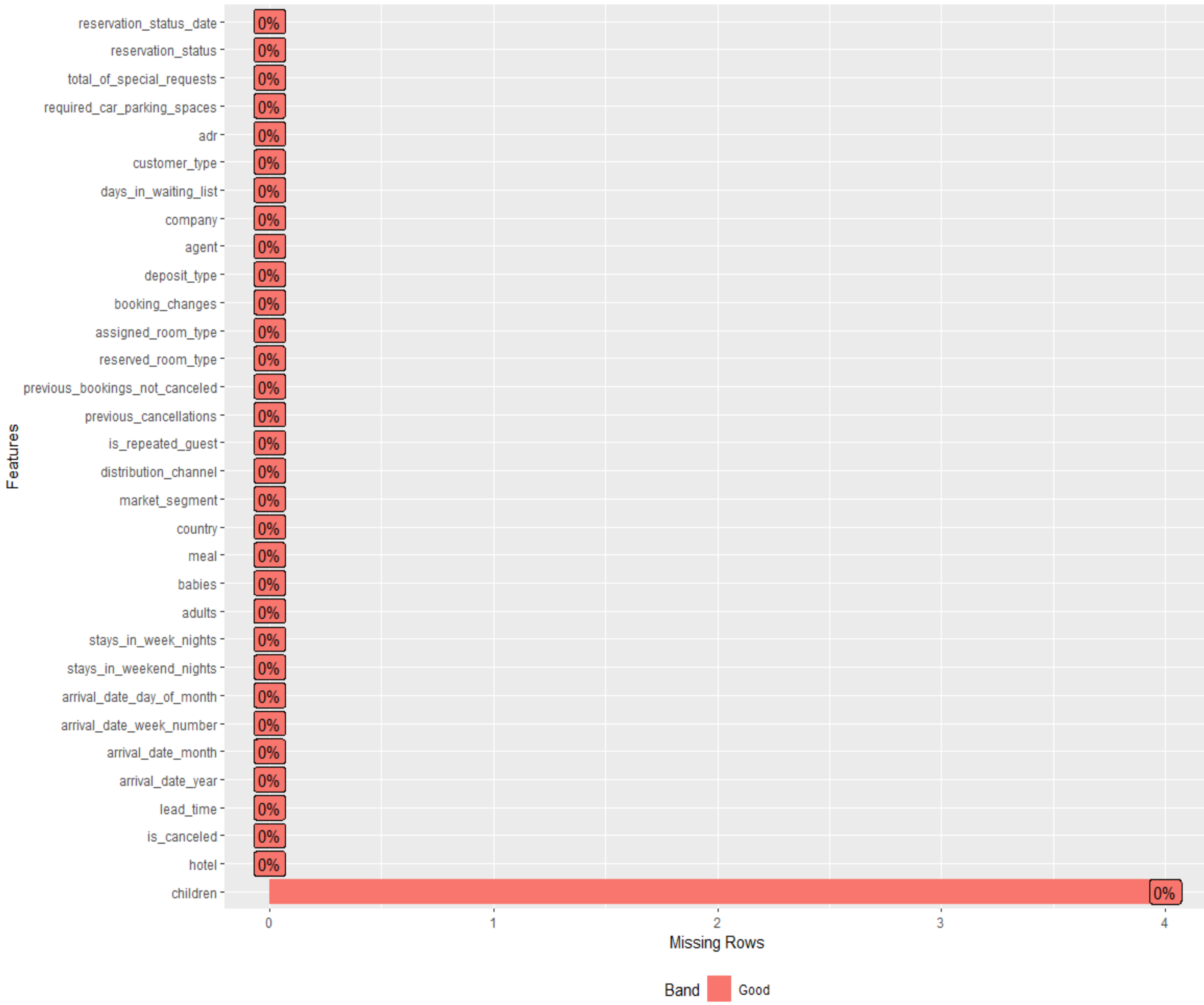
R library

We used different R library for this project like funModeling, tidyverse, Hmisc, DataExplorer, dplyr, caret, Mass, glmnet, randomforest, lattice, magrittr, ggplot2, scales, gridExtra, psych, plotly and many more.

DATA CLEANING

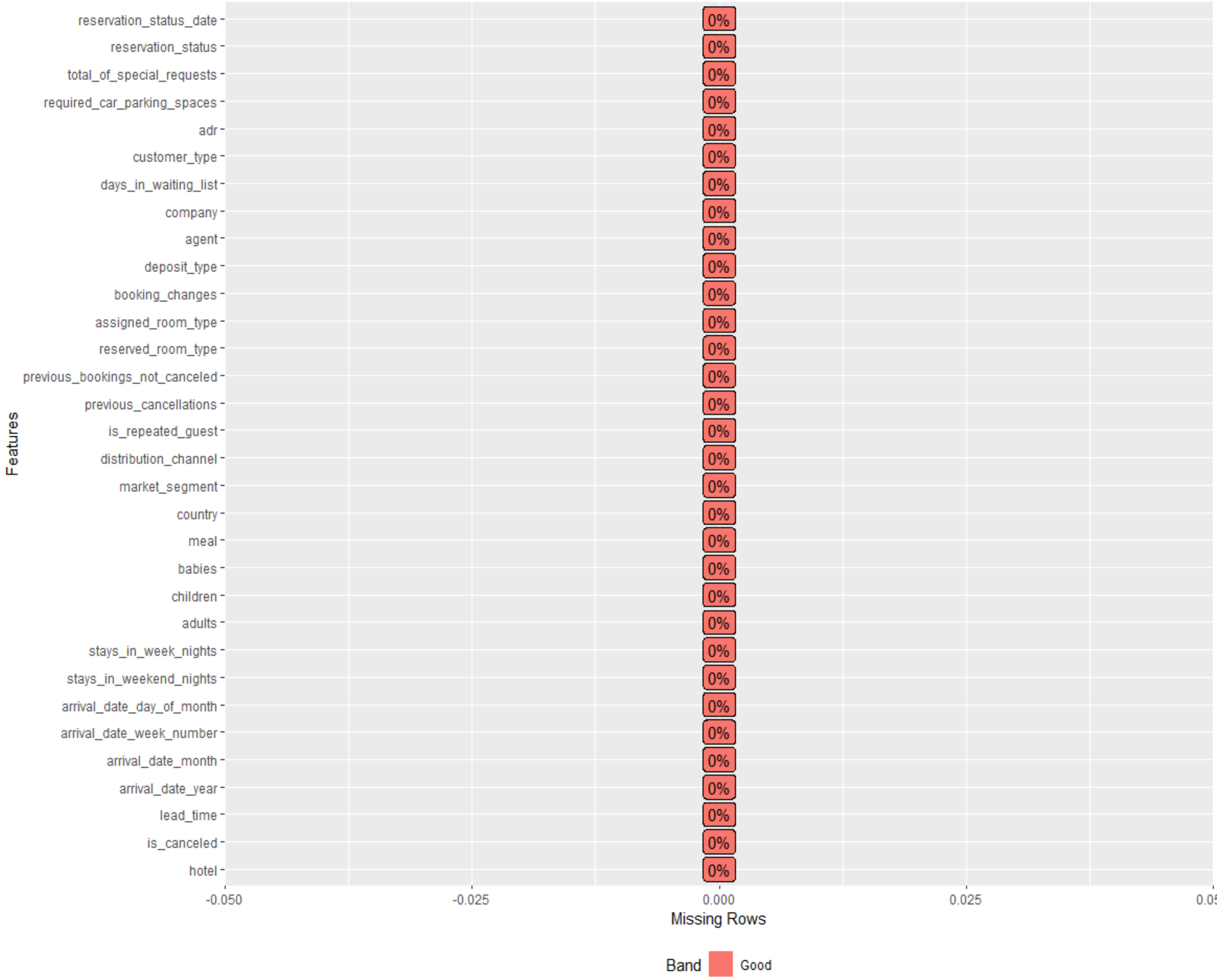
Missing Data

- There were only 4 missing values for children column in our dataset. We replaced those 4 missing rows in Children column with the corresponding Babies column.There were 489 null values in the country column. We removed those rows to get the final dataset.



Data preparation / Wrangling:

- For the meal column we replaced undefined rows with SC as both means no meal package.
- There were 8 undefined rows in the market segment column we replaced those Undefined columns with mode value of the market segment.
- There were 2 undefined rows in the distribution channel column, we replaced those Undefined rows with mode value of the distribution channel.

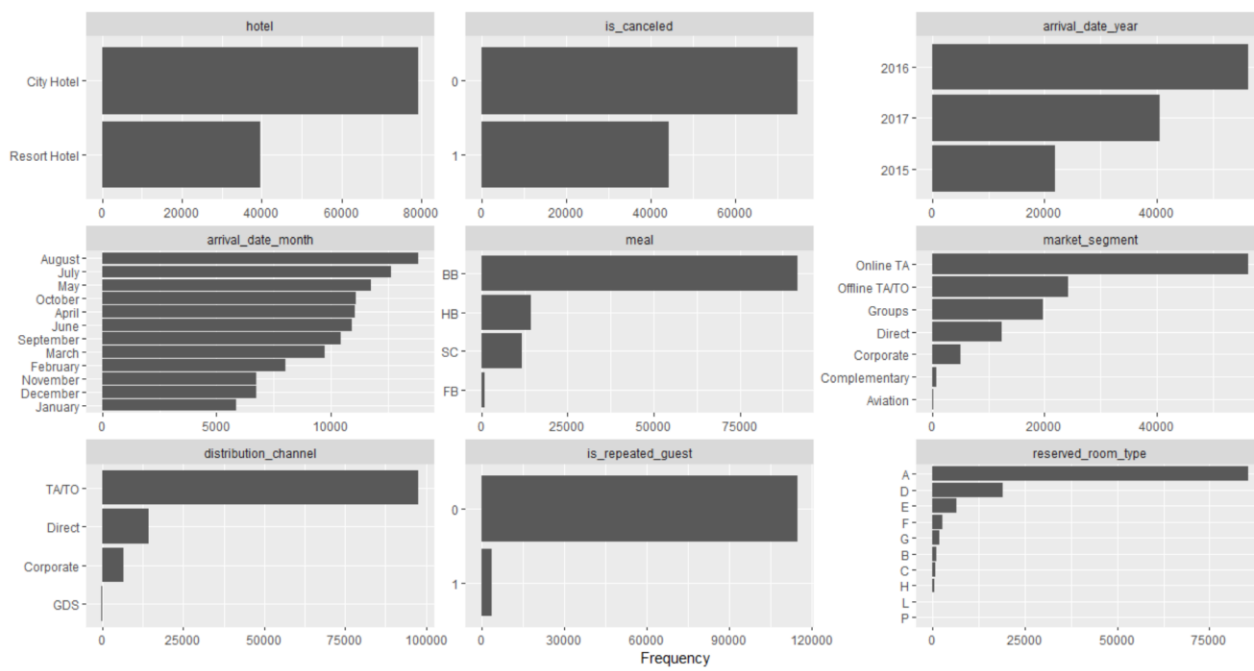


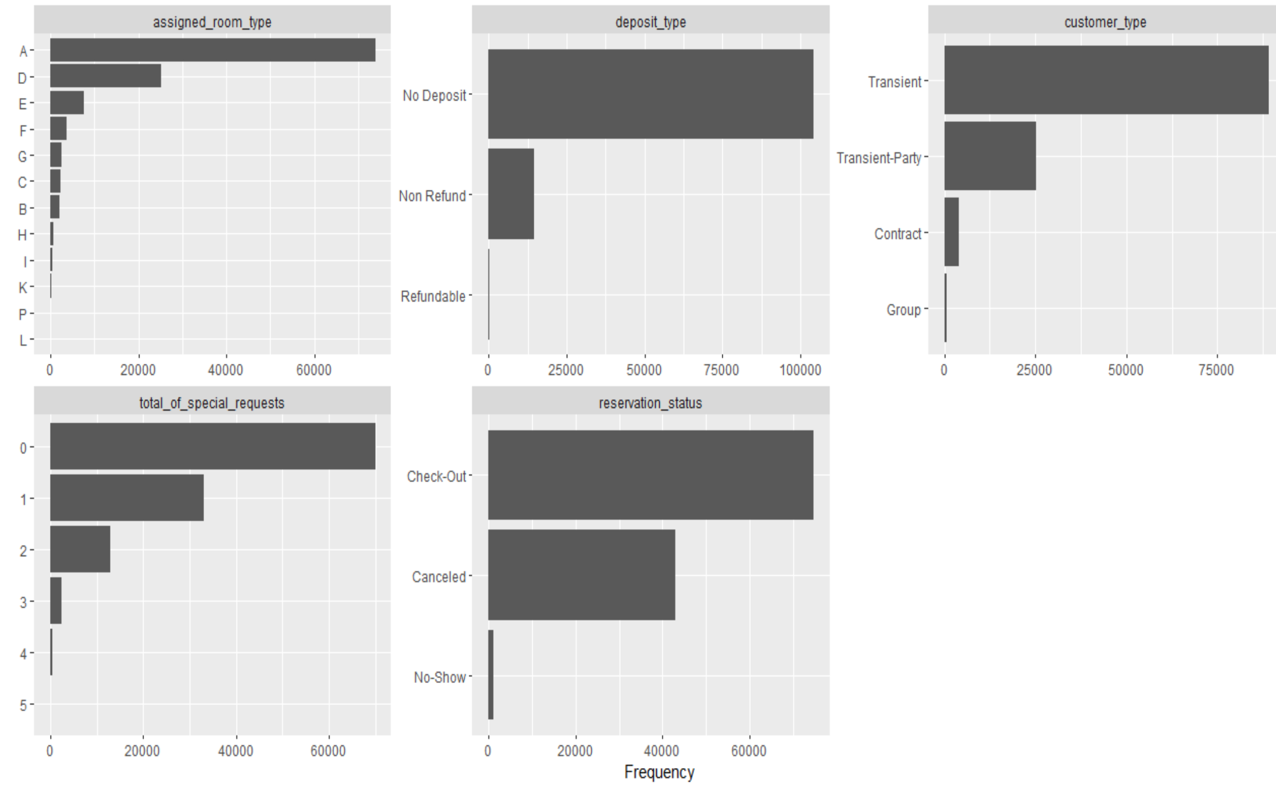
Histogram of all the numeric data

plot_num(hotels)

Barplots

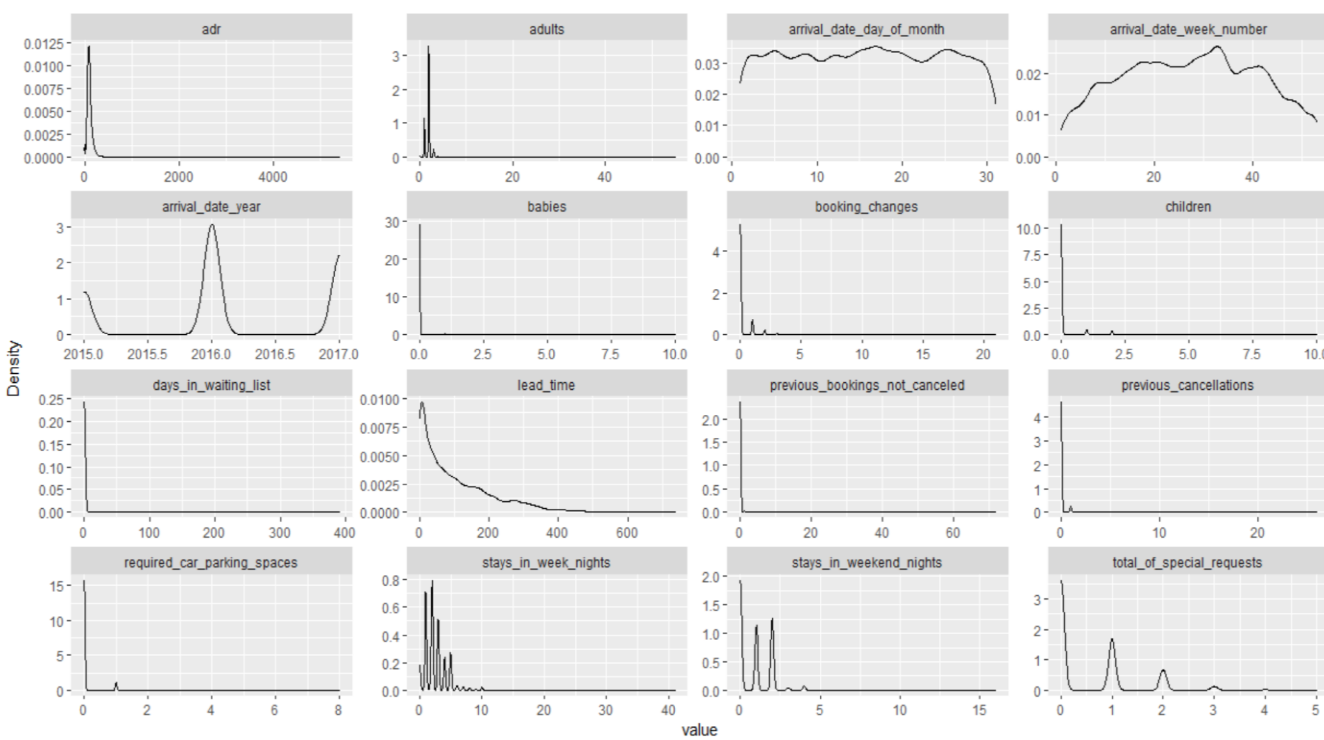
plot_bar(hotels)



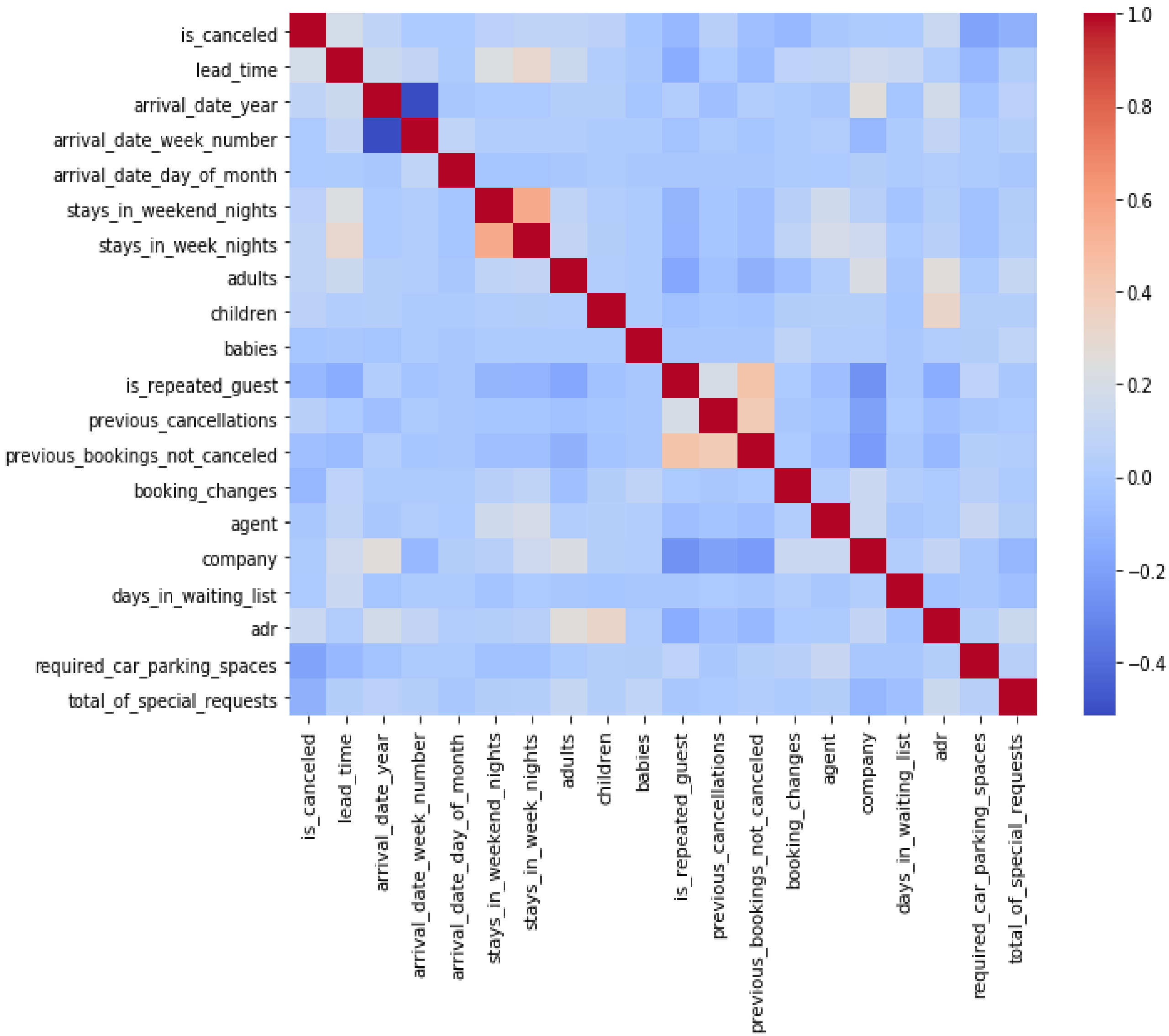


Density plot

plot_density(hotels)

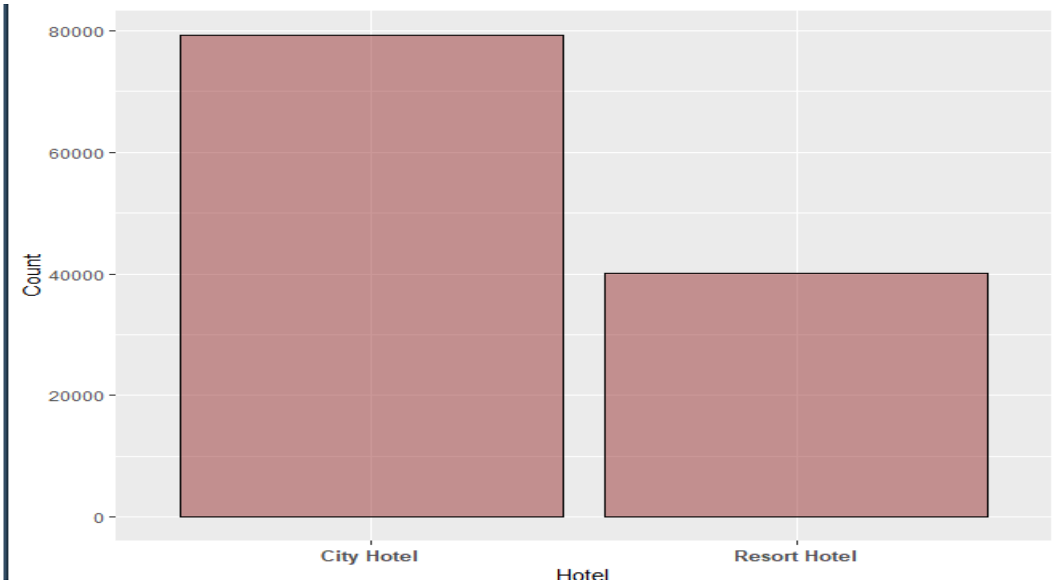


Correlation matrix



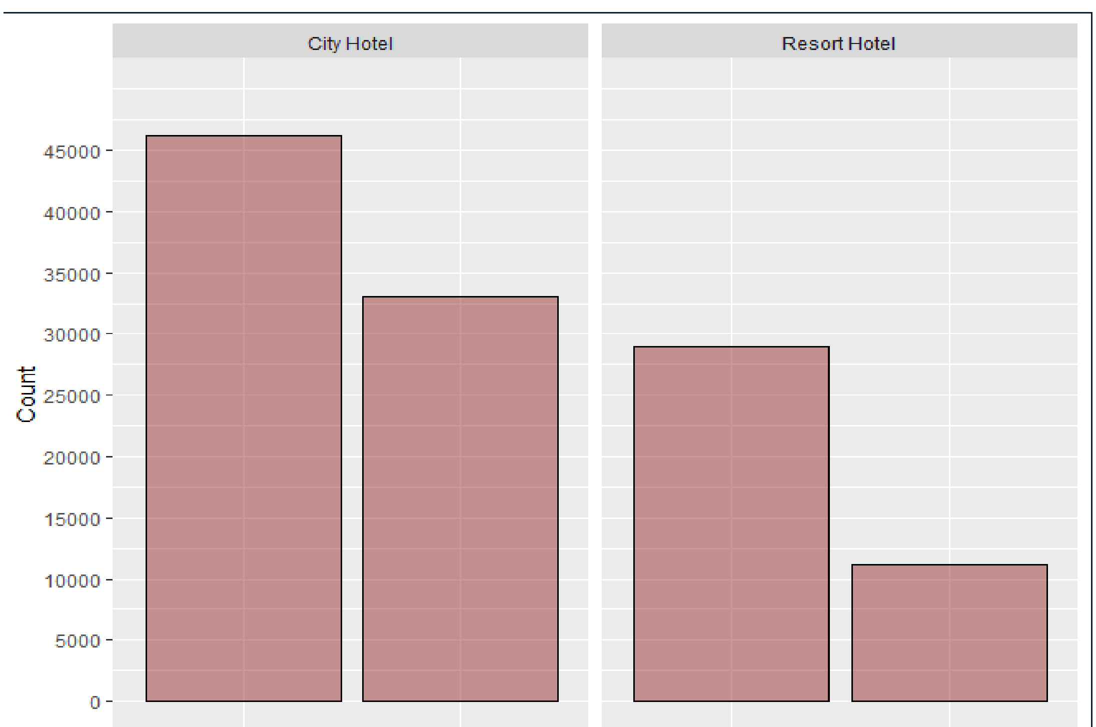
Hotels:

```
#Hotels
ggplot(hotels,aes(x=factor(hotel))) +
  geom_bar(col ="black",fill="#993333",alpha=0.5) +
  theme(axis.text.x = element_text(face="bold", size=10)) +
  scale_x_discrete("Hotel") +
  scale_y_continuous("Count")
```



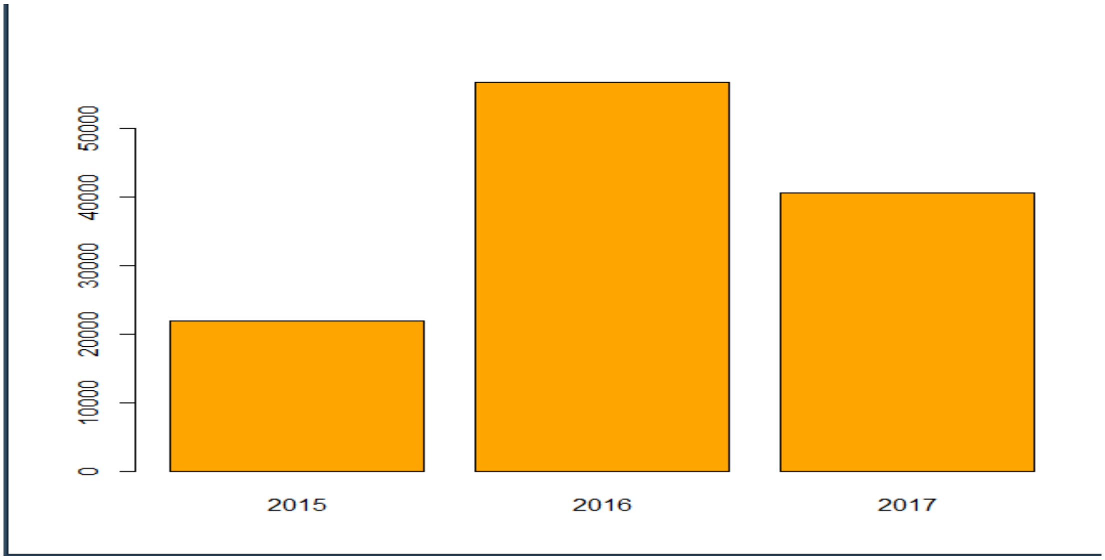
<u>Number of city hotel and Resort Hotel cancelled or not cancelled</u>:

```
#NUmber of city hotel and Resort Hotel cancelled or not cancelled
ggplot(data = hotels,aes(factor(is_canceled)))+
  geom_bar( col='black', fill="#993333", alpha = 0.5) +
  facet_wrap(~hotel) +
  scale_x_discrete("Canceled",labels = c("No","Yes")) +
  scale_y_continuous("Count",limits = c(0,50000),breaks=seq(0,47222,by=5000))  +
  theme(axis.text.x = element_text(face="bold", size=10))
```



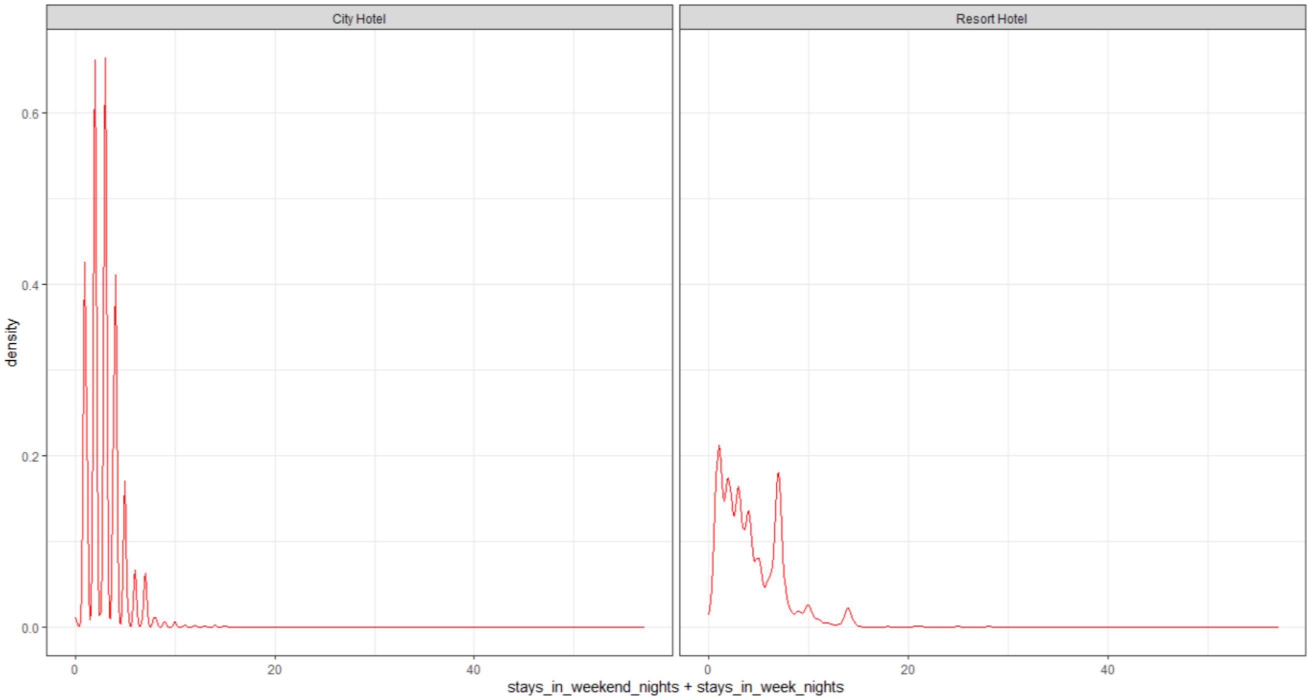
Year of arrival:

```
#Year of arrival
barplot(table(hotels$arrival_date_year), col='orange')
```



stay duration for both the hotels

ggplot(data=hotels, aes(stays_in_weekend_nights + stays_in_week_nights))+geom_density(col="red")+facet_wrap(~hotel)+theme_bw()

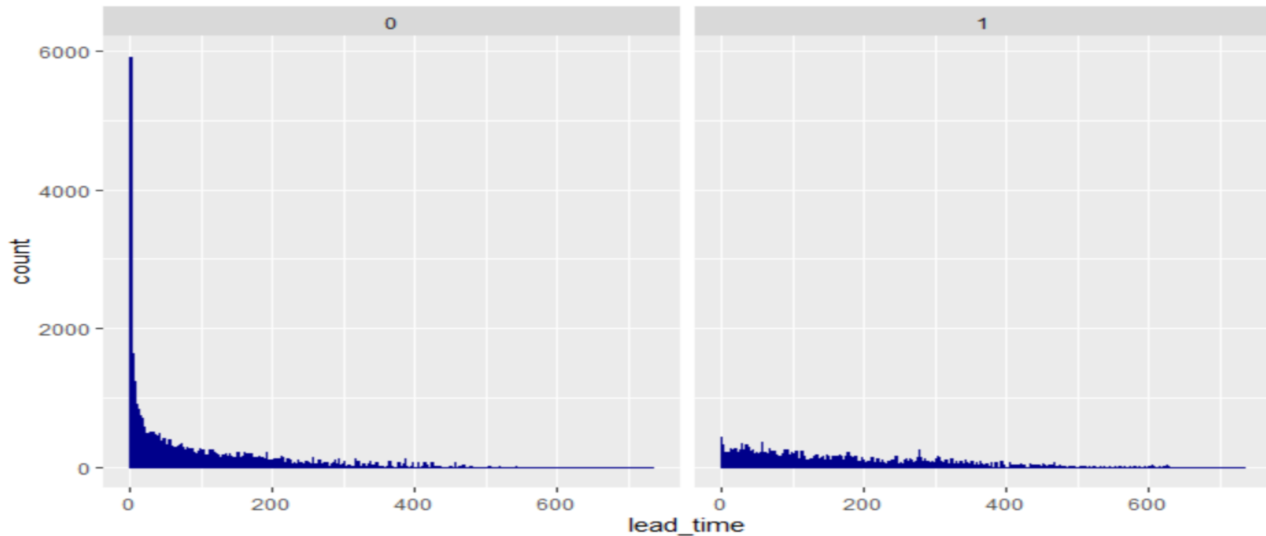


Cancellation Vs Booking Time

Number of days that elapsed between the entering date of the booking and the arrival date is less for the people who cancelled.

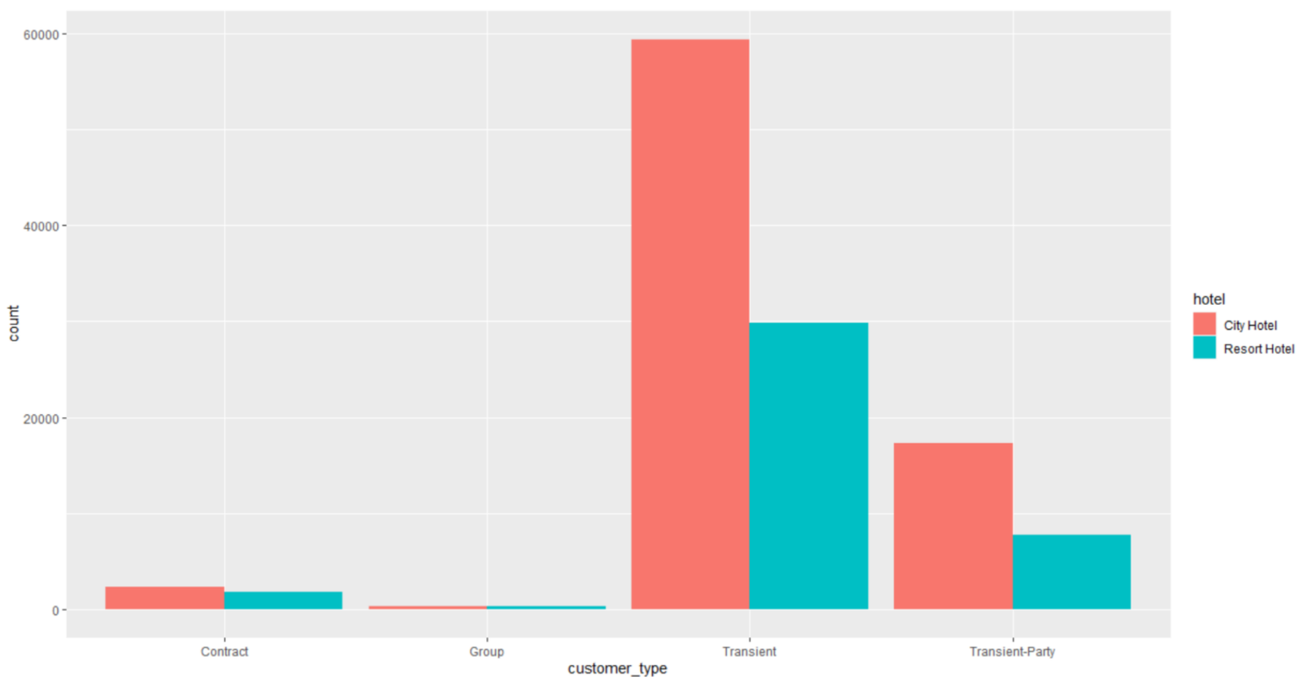
ggplot(data = hData , aes(lead_time)) + geom_histogram(binwidth = 0.8,col='darkblue'

)+ facet_wrap(~ is_canceled)



Hotel preference by customer type

ggplot(data=hotels,aes(customer_type,fill=hotel)) +geom_bar(stat="count",position = position_dodge())



Analysis by arrival month

Number of bookings made were highest in the month of July and August and lowest in January.

hData$arrival_date_month = factor(hData$arrival_date_month, levels = month.name)

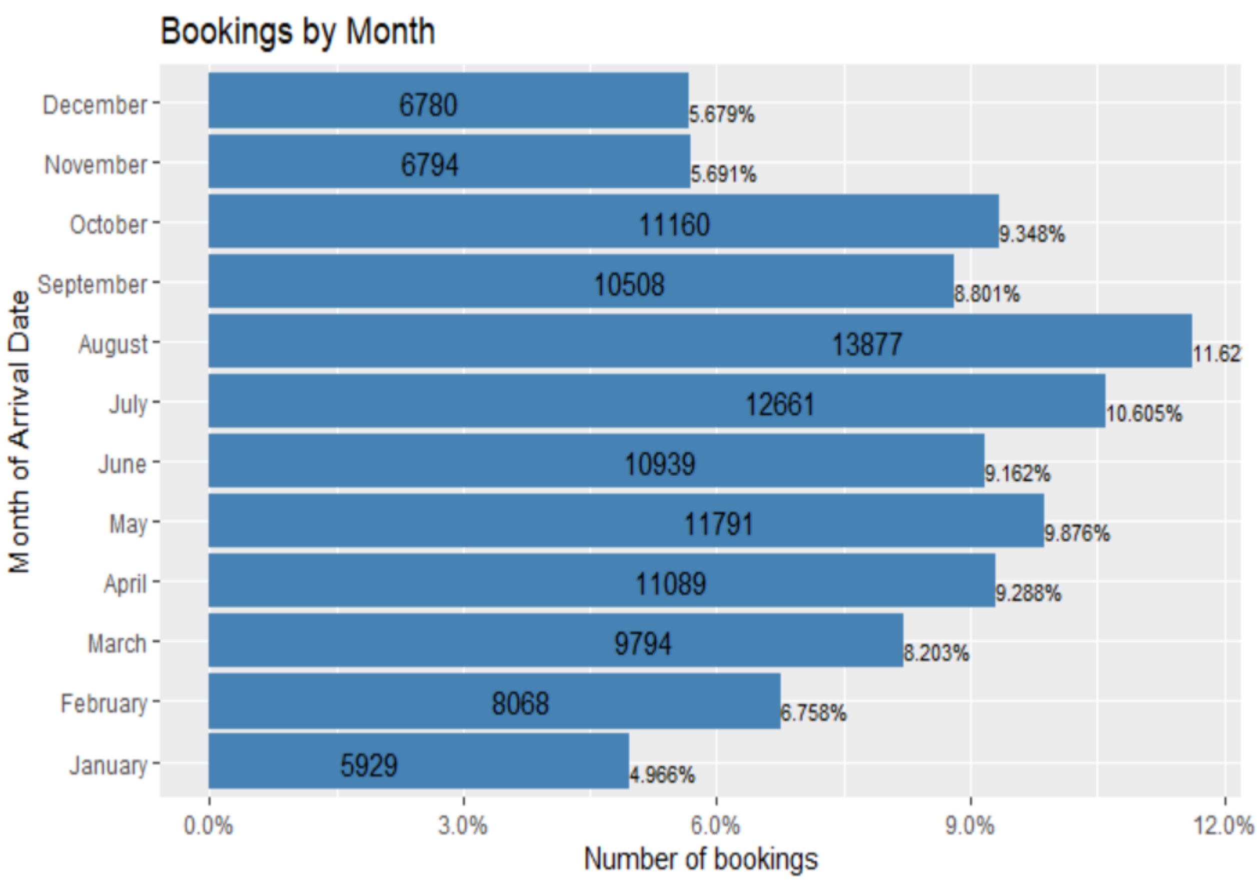
ggplot(data = hData,aes(x = arrival_date_month,y = prop.table(stat(count)),

label = scales::percent(prop.table(stat(count))))) + geom_bar(fill = 'steelblue') +

geom_text(stat = "count", position = position_dodge(1),vjust = 1, hjust=0,size = 3)+scale_y_continuous(labels = scales::percent) +

labs(title = "Bookings by Month", x = "Month of Arrival Date",y = "Number of bookings")+ coord_flip() +

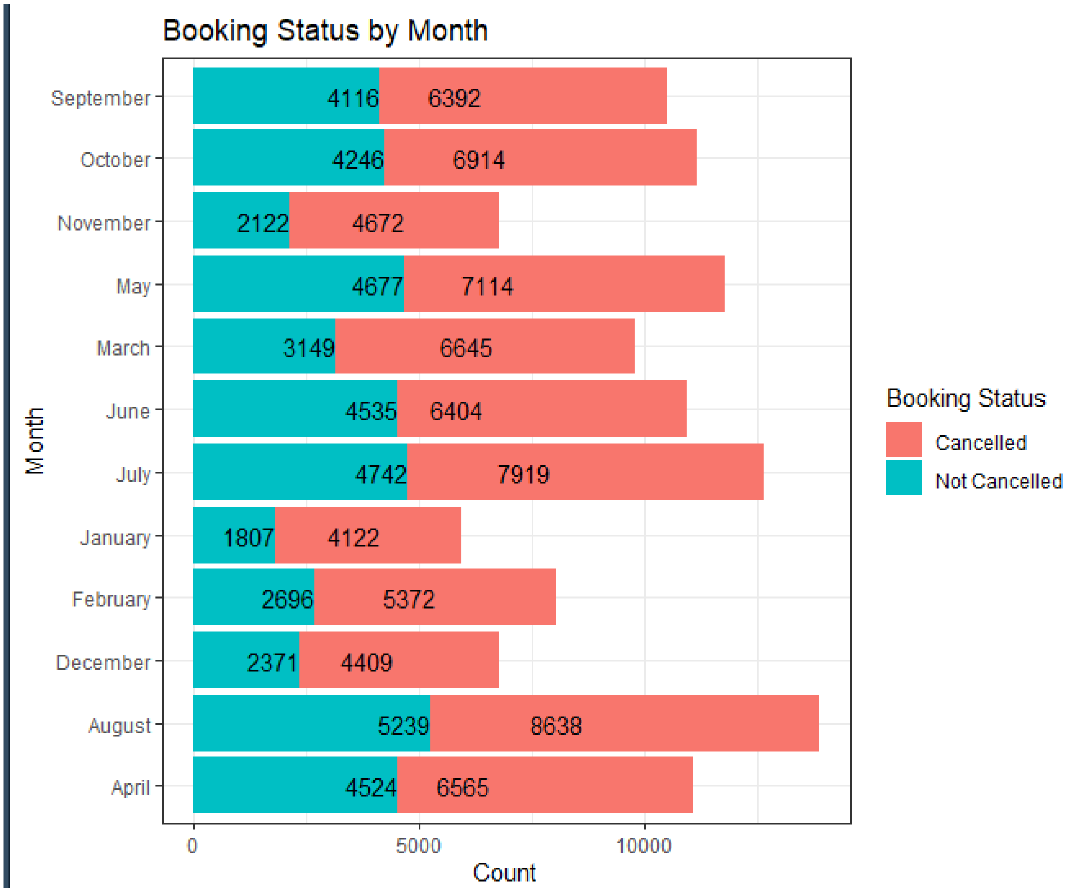
geom_text(stat = "count", aes(label = ..count..), hjust = 5)



<u>Booking made for month in different hotel:</u>

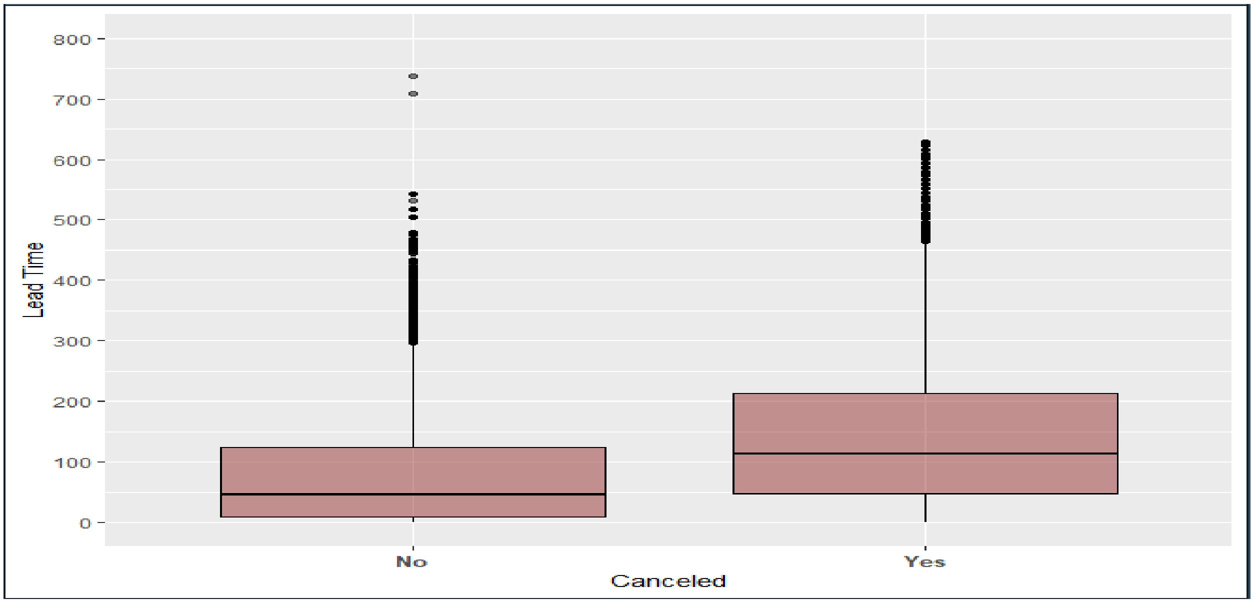
(data use the booking confirmation and not the check ins)

```
ggplot(hotels, aes(arrival_date_month, fill = factor(is_canceled))) +
  geom_bar() + geom_text(stat = "count", aes(label = ..count..), hjust = 1) +
  coord_flip() + scale_fill_discrete(
    name = "Booking Status",
    breaks = c("0", "1"),
    label = c("Cancelled", "Not Cancelled")
  ) +
  labs(title = "Booking Status by Month",
       x = "Month",
       y = "Count") + theme_bw()
```



Lead time and canceled:

```
#Canceled and Lead time
ggplot(data = hotels, aes(x = factor(is_canceled), y = lead_time  )) +
  geom_boxplot(col='black', fill="#993333", alpha = 0.5) +
  theme(axis.text.x = element_text(face="bold", size=10)) +
  scale_y_continuous("Lead Time",limits = c(0,800),breaks=seq(0,800,by=100)) +
  scale_x_discrete("Canceled",labels = c("No","Yes"))
```



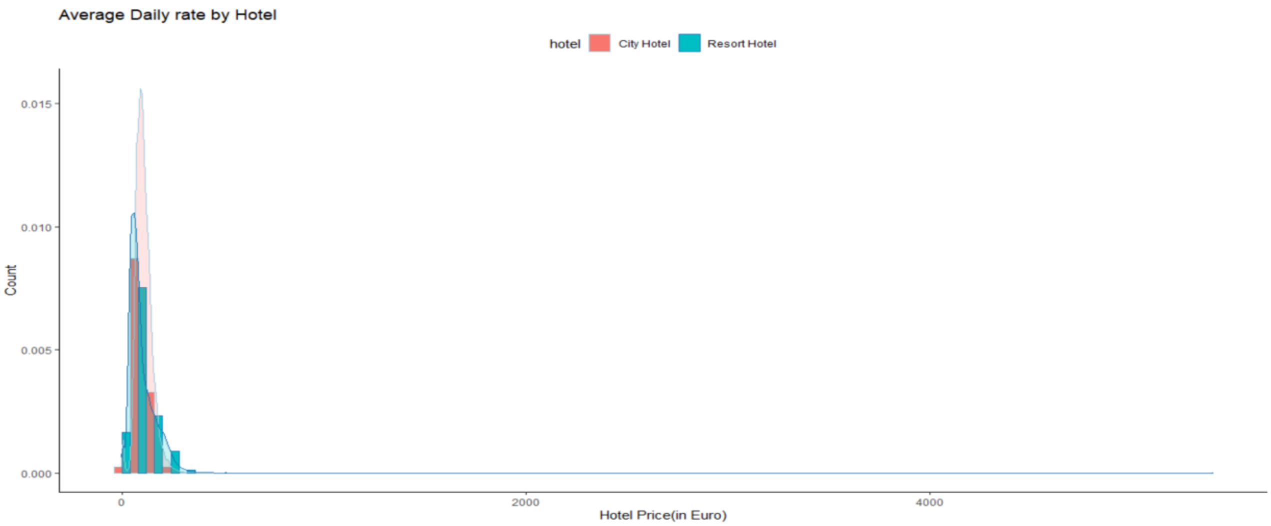
Average daily rate for both hotels

ggplot(data=hotels,aes(x=adr,fill=hotel,color=hotel))+geom_histogram(aes(y=..density..),

position = position_dodge(),binwidth=80)+geom_density(alpha=0.2)+

labs(title = "Average Daily rate by Hotel", x = "Hotel Price(in Euro)",y = "Count")+

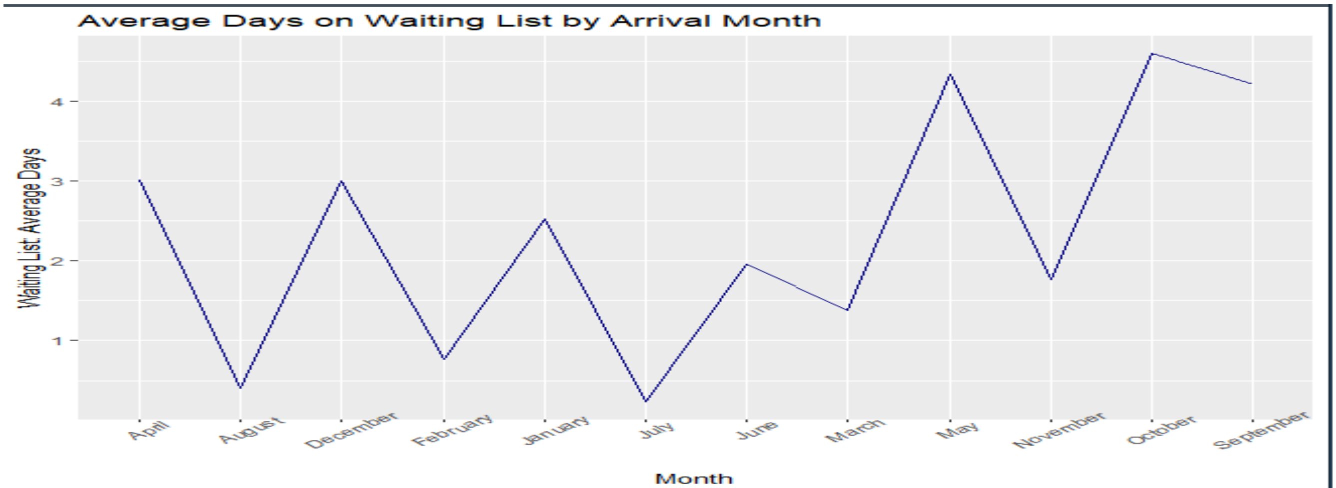
scale_color_brewer(palette = "Paired") + theme_classic() + theme(legend.position = "top")



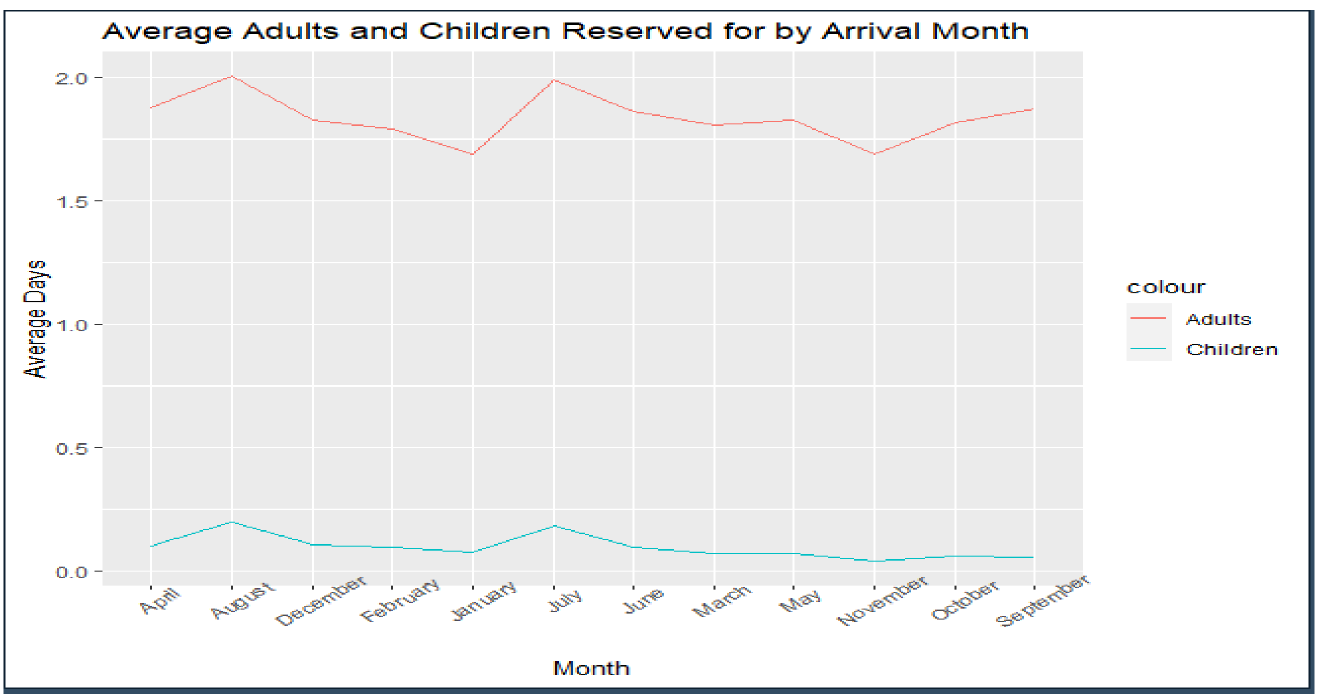
Average waiting list

May and October have the highest waiting times; these months represent the times right before and after peak reservation months (respectively).

```r
#May and October have  the highest waiting times; these months represent the times right before and after peak reservation months (respectively)
ggplot(hotels, aes(x=arrival_date_month, y=days_in_waiting_list, group=1)) + stat_summary(fun="mean", geom="line", col="navy") +
  ggtitle("Average Days on Waiting List by Arrival Month") + ylab("Waiting List: Average Days") + xlab("Month") +
  theme(axis.text.x=element_text(angle=40))
```




```r
  geom_line(aes(y=MEANADULTS, x=arrival_date_month, group=1, col="Adults")) +
  geom_line(aes(y=MEANCHILDREN, x=arrival_date_month, group=1, col="Children")) +
  ggtitle("Average Adults and Children Reserved for by Arrival Month") + xlab("Month") + ylab("Average Days") +
  theme(axis.text.x=element_text(angle=40))
```



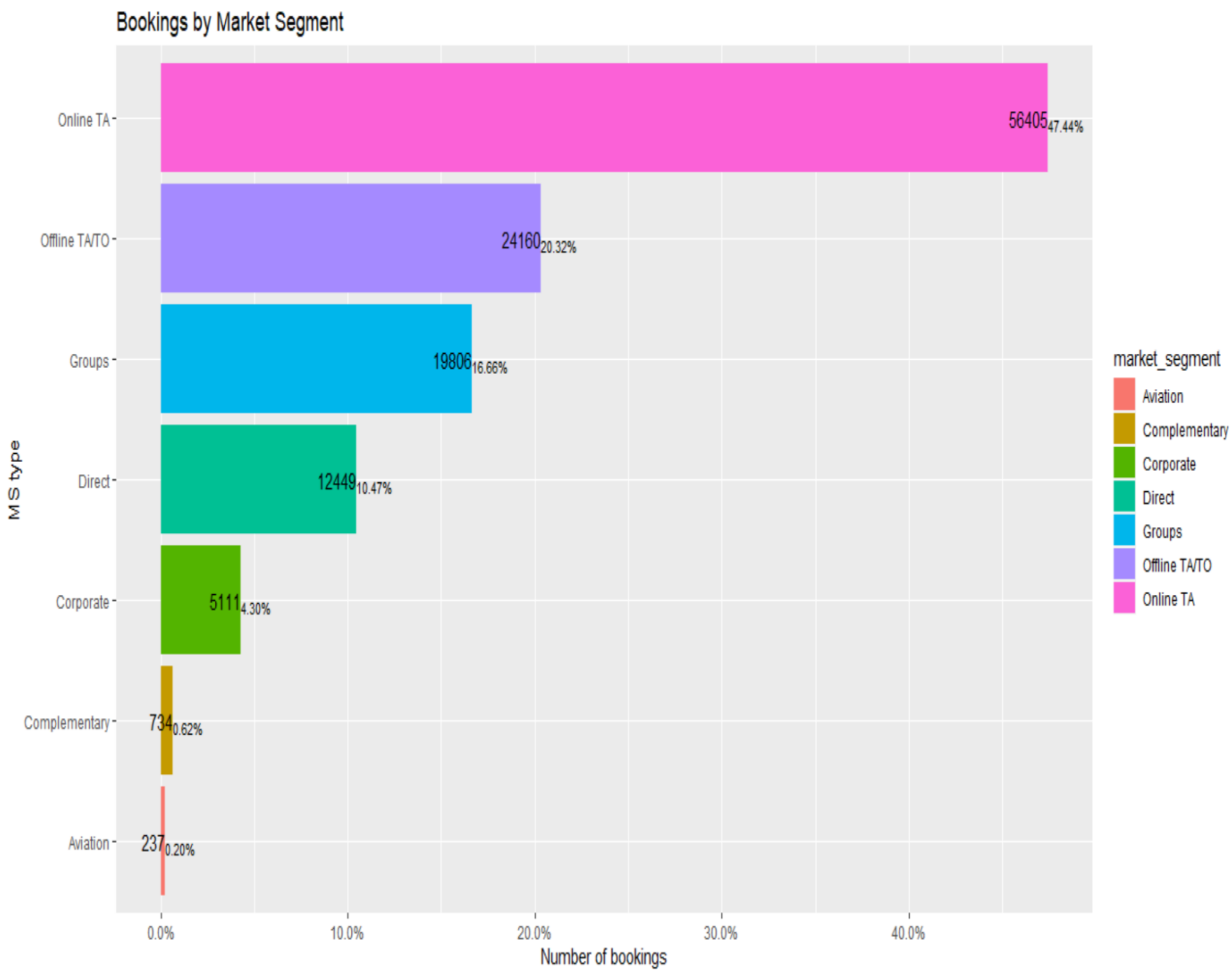
In Online market segment's cancellation is more.

ggplot(data = hotels,aes(x = market_segment,fill = market_segment,y = prop.table(stat(count)),

label = scales::percent(prop.table(stat(count))))) + geom_bar() +

geom_text(stat = "count", position = position_dodge(1),vjust = 1, hjust=0,size = 3)+scale_y_continuous(labels = scales::percent) + coord_flip() +labs(title = "Bookings by Market Segment", x = "MS type",y = "Number of bookings") +

geom_text(stat = "count", aes(label = ..count..), hjust = 1)



Couples booking cancellation is more:

```
#Couples booking cancellation is more
hotels %>% group_by(hotels$adults)  %>% summarise(length(is_canceled))
```

```
   `hotels$adults` `length(is_canceled)`
             <dbl>                 <int>
 1               0                   403
 2               1                 23027
 3               2                 89680
 4               3                  6202
 5               4                    62
 6               5                     2
 7               6                     1
 8              10                     1
 9              20                     2
10              26                     5
11              27                     2
12              40                     1
13              50                     1
14              55                     1
```

<u>Hotel Reserved Type cancellation</u>: 'A' type room cancellation is higher.

```
#'A' type room cancellation is higher
hotels %>% group_by(hotels$reserved_room_type)  %>% summarise(length(is_canceled))
```

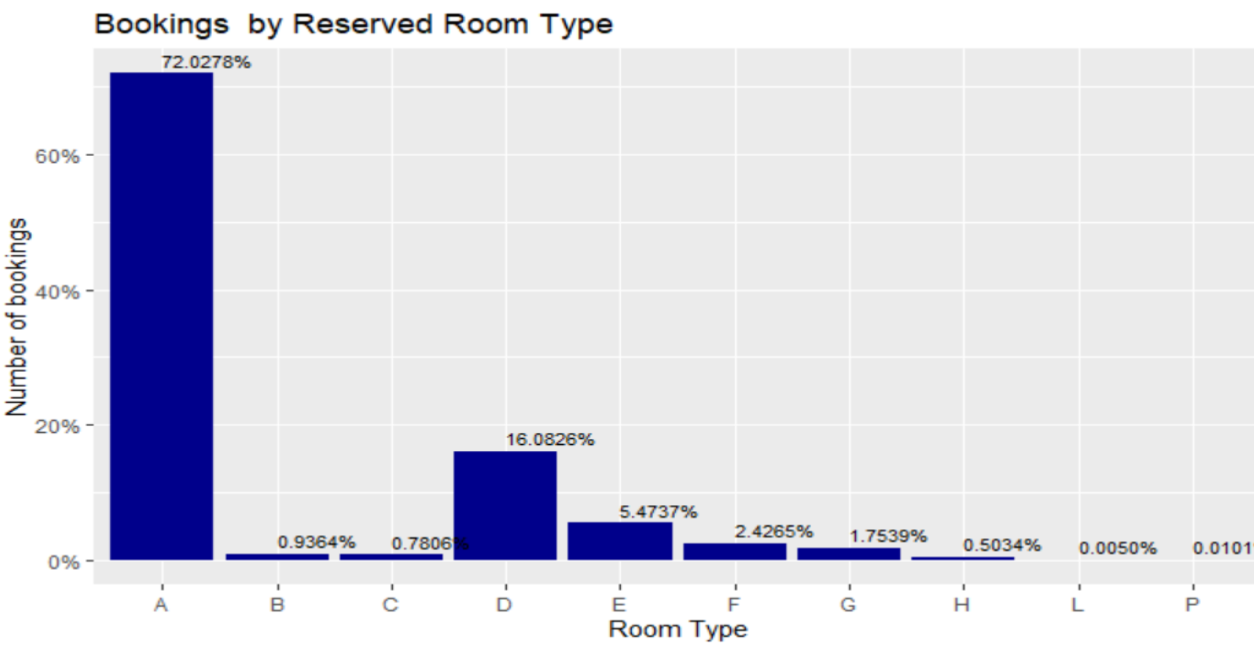
```
`hotels$reserved_room_type` `length(is_canceled)`
<fct>                                       <int>
A                                           85994
B                                            1118
C                                             932
D                                           19201
E                                            6535
F                                            2897
G                                            2094
H                                             601
L                                               6
P                                              12
```

<u>Analysis by Reserved Room Type</u>

ggplot(data = hData,aes(x = reserved_room_type ,y = prop.table(stat(count)),

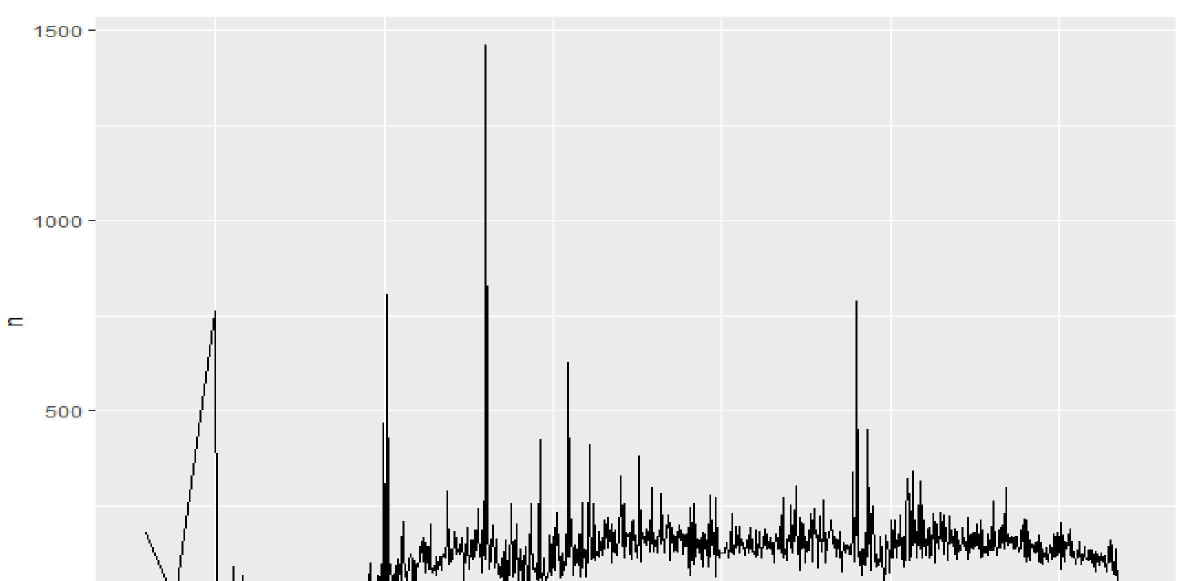
label = scales::percent(prop.table(stat(count))))) + geom_bar(fill = 'darkblue') +

geom_text(stat = "count", position = position_dodge(1),vjust = -0.5, hjust=0,size = 3)+scale_y_continuous(labels = scales::percent) +labs(title = "Bookings by Reserved Room Type", x = "Room Type",y = "Number of bookings")



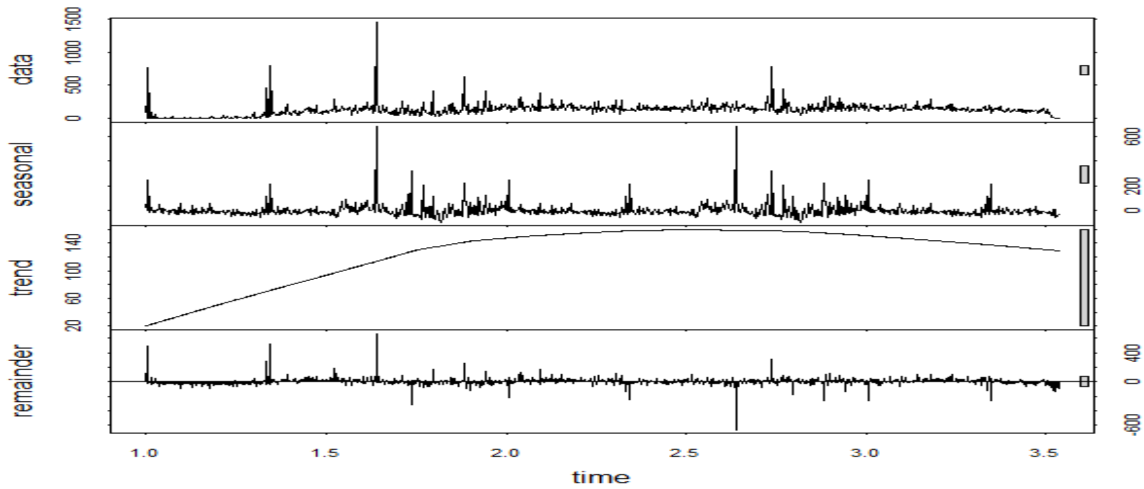
Reservation status date by year:

```
#'A' type room cancellation is higher
hotels %>% group_by(hotels$reserved_room_type)  %>% summarise(length(is_canceled))
```




```
#Time Series Analysis
ggplot(ts, aes(reservation_status_date, n)) + geom_line()

ts <- ts %>% filter(!is.na(n))
ts
# Frequency is set with 365 because it's daily
components <- stl(ts(ts$n, frequency=365), 'periodic')
# seasonal, trend, remainder
plot(components)
```



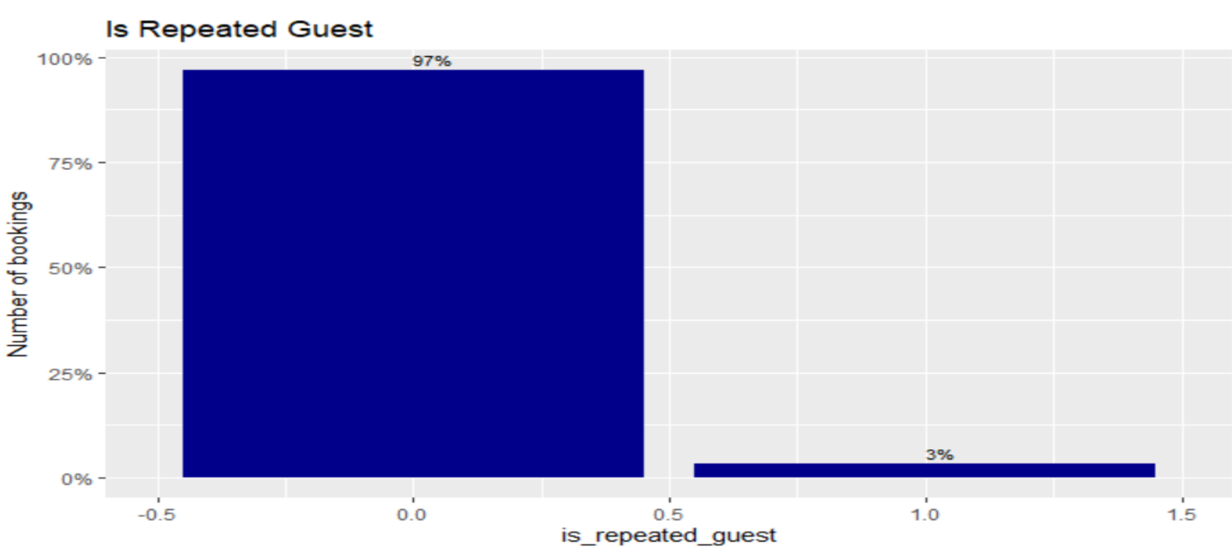
Analysis of repeated guests

Repeated guests are negligible with only 3%.

ggplot(data = hData,aes(x = is_repeated_guest ,y = prop.table(stat(count)),

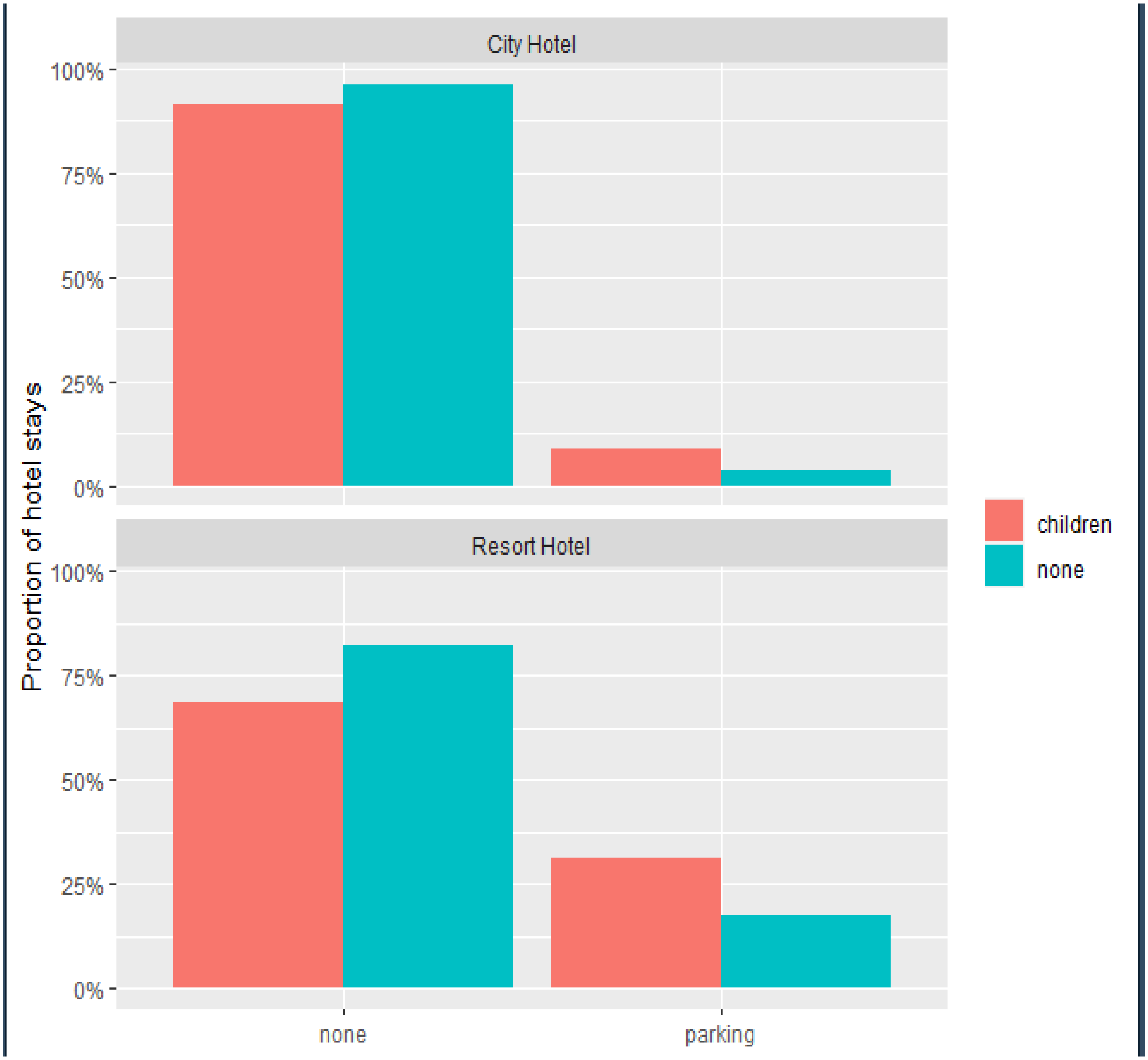
label = scales::percent(prop.table(stat(count))))) + geom_bar(fill = 'darkblue') +

geom_text(stat = "count", position = position_dodge(1),vjust = -0.5, hjust=0,size = 3)+scale_y_continuous(labels = scales::percent) +labs(title = "Is Repeated Guest", x = "is_repeated_guest",y = "Number of bookings")



Are the Guest with children need parking space then guest with no children? (Multivariate)

```
hotel_stays %>%
  count(hotel, required_car_parking_spaces, children) %>%
  group_by(hotel, children) %>%
  mutate(proportion = n / sum(n)) %>%
  ggplot(aes(required_car_parking_spaces, proportion, fill = children)) +
  geom_col(position = "dodge") +
  scale_y_continuous(labels = scales::percent_format()) +
  facet_wrap(~hotel, nrow = 2) +
  labs(
    x = NULL,
    y = "Proportion of hotel stays",
    fill = NULL
  )
```

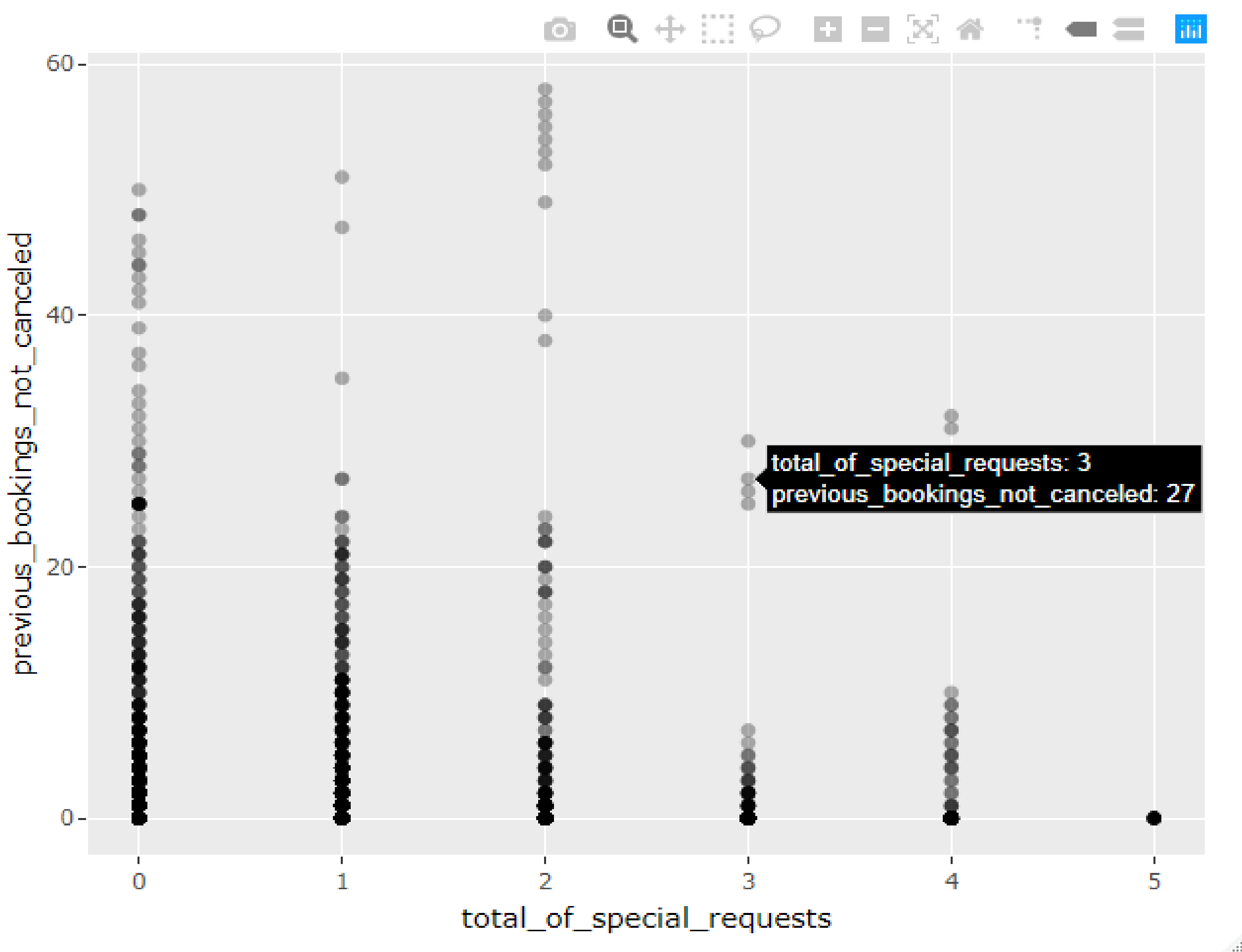


Scatterplot of total_of_special_request vs. previous_bookings_not_canceled in 2016:

```
# Load the plotly package
library(plotly)

# Store the scatterplot of total_of_special_request vs.previous_bookings_not_canceled in 2016
scatter <- hotels %>%
  filter(arrival_date_year == '2016') %>%
  ggplot(aes(x = total_of_special_requests, y = previous_bookings_not_canceled)) +
  geom_point(alpha = 0.3)

# Convert the scatterplot to a plotly graphic
ggplotly(scatter)
```

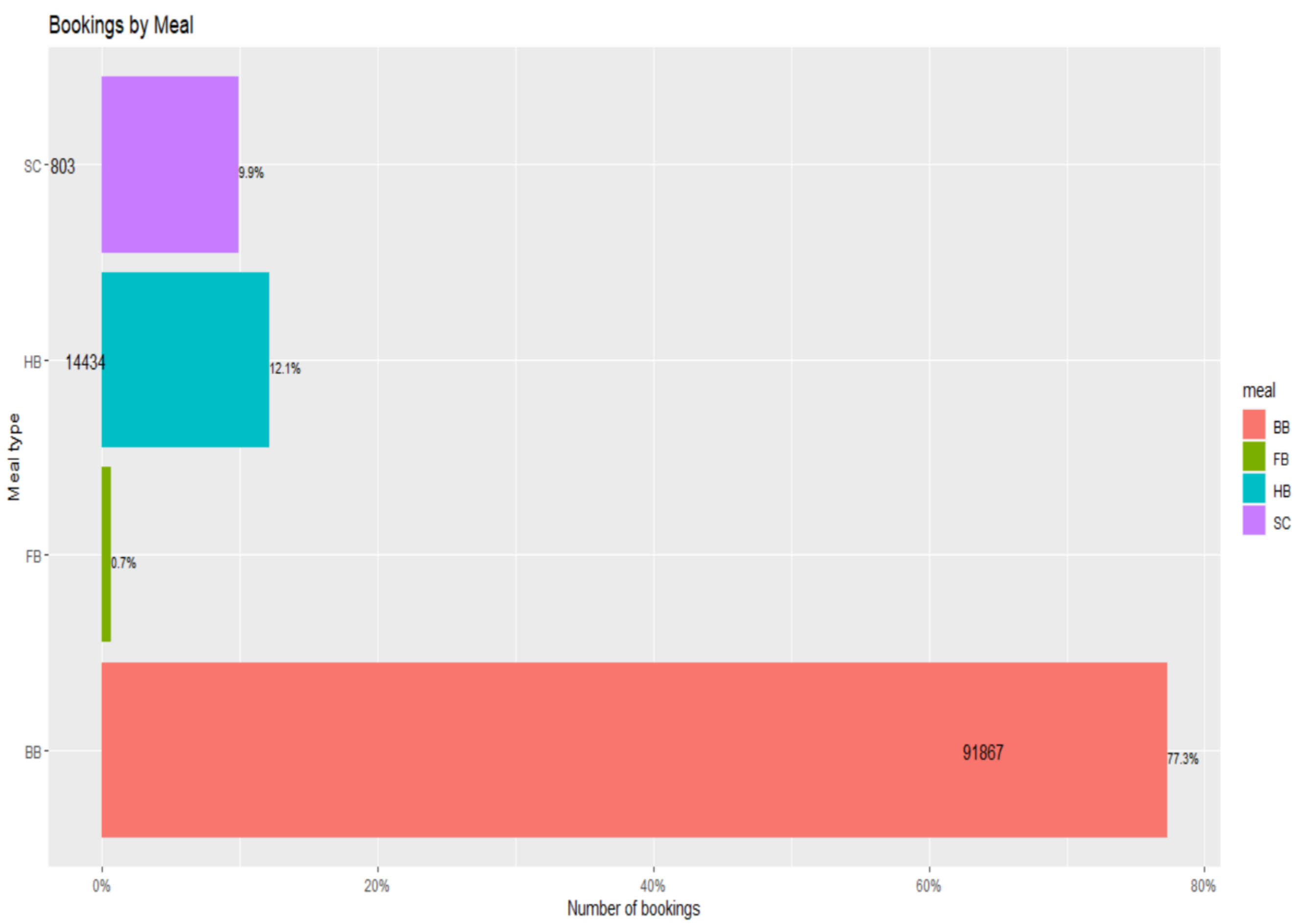


Analysis by Meal

Bookings made with Bed and Breakfast are 77%

Full Board meal bookings are low with 0.67%

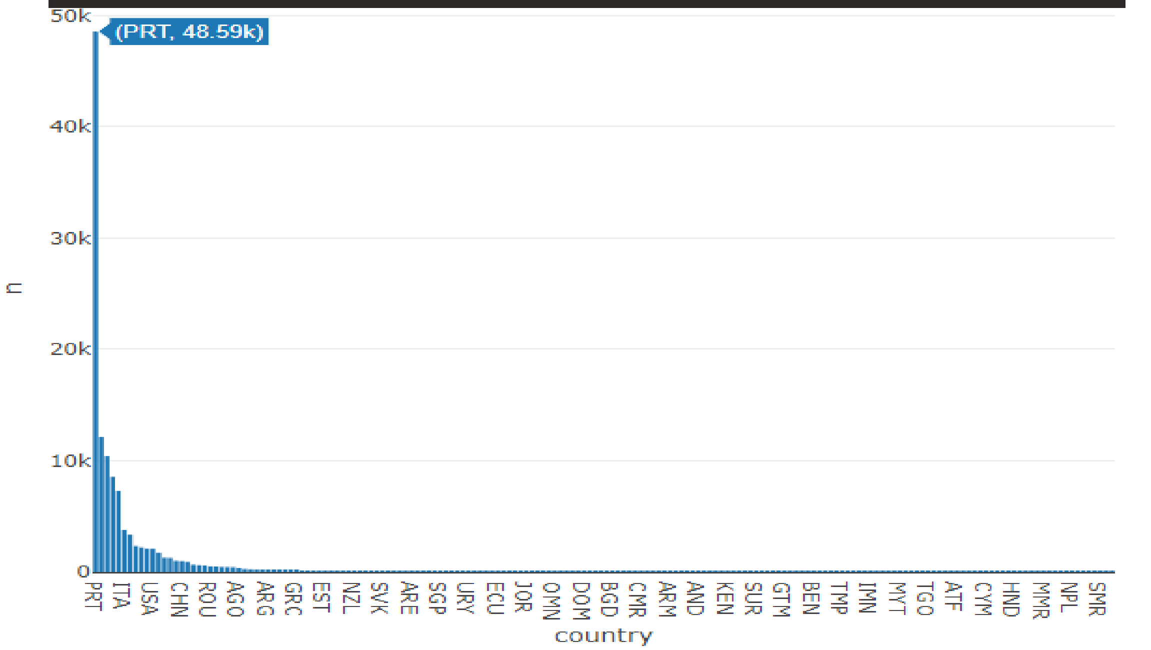
```
ggplot(data = hotels,aes( x = meal,fill = meal,y = prop.table(stat(count)),
                    label = scales::percent(prop.table(stat(count))))) +  geom_bar() +
 geom_text(stat = "count", position = position_dodge(1),vjust = 1, hjust=0,size = 3)+scale_y_continuous(labels = scales::percent) + coord_flip() +labs(title = "Bookings by Meal", x = "Meal type",y = "Number of bookings") +
 geom_text(stat = "count", aes(label = ..count..), hjust = 5)
```



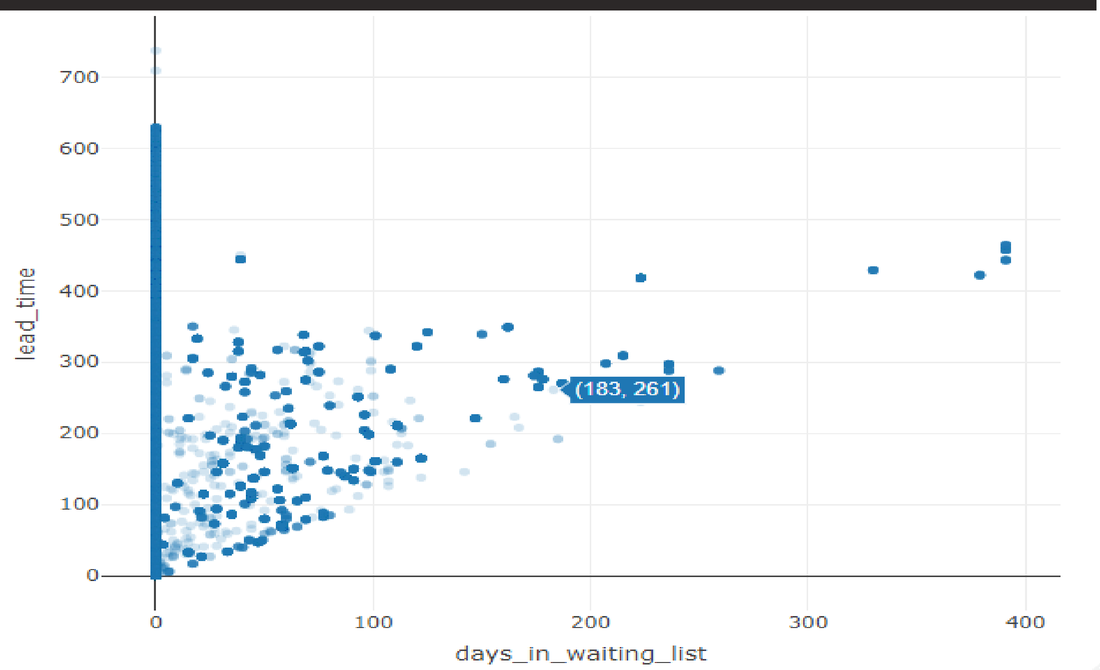
<u>Bar plot for country:</u>

Most travelers come from Portugal.

```
# Create a frequency for country
country_table <- hotels %>%
  count(country)
#Bar Plot for country
country_table %>%
  mutate(country = fct_reorder(country, n, .desc = TRUE)) %>%
  plot_ly(x = ~country, y = ~n) %>%
  add_bars()
country_table
```




```
# Create a scatter plot of days in waiting against lead time
hotels %>%
  plot_ly(x = ~days_in_waiting_list, y = ~lead_time) %>%
  add_markers(marker= list(opacity=0.2))
```



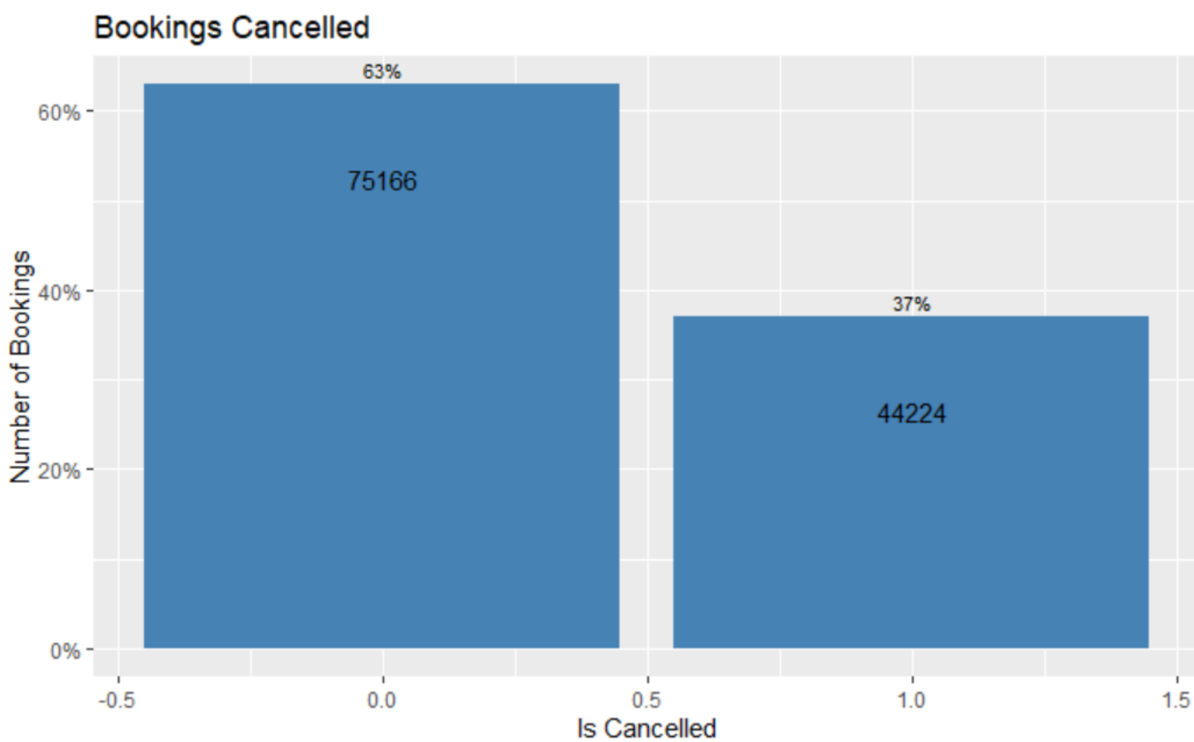
37% of the bookings were cancelled

ggplot(data = hData,aes(x = is_canceled,y = prop.table(stat(count)),

label = scales::percent(prop.table(stat(count))))) + geom_bar(fill = 'steelblue') +geom_text(stat = "count", aes(label = ..count..), vjust = 5) +

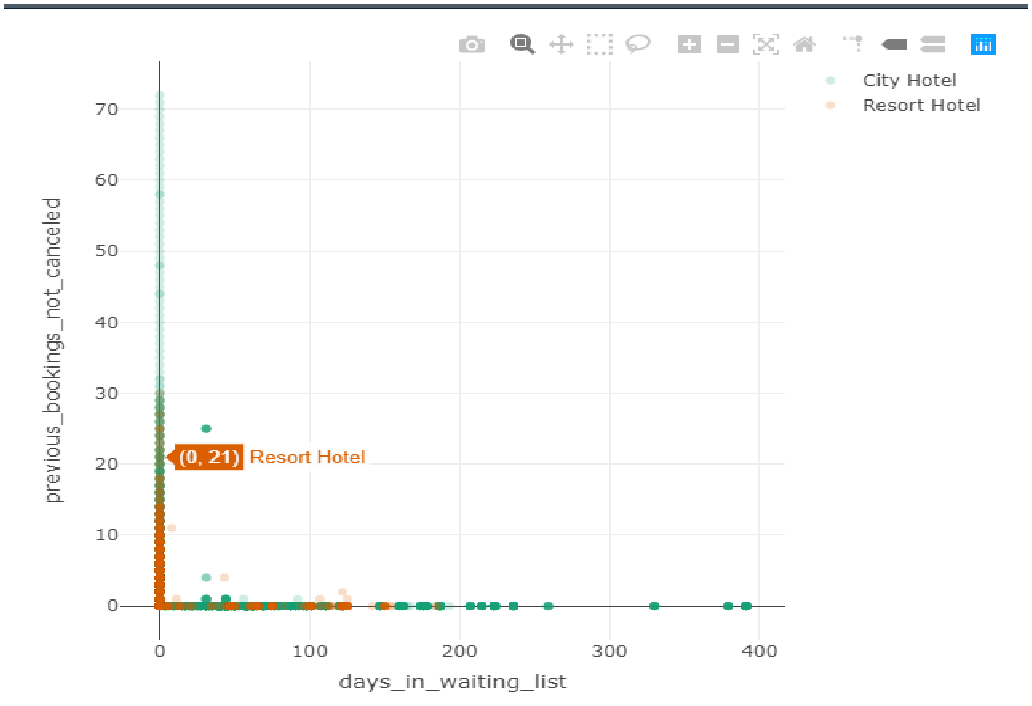
geom_text(stat = "count", position = position_dodge(.9),vjust = -0.5,size = 3)+

scale_y_continuous(labels = scales::percent) + labs(title = "Bookings Cancelled", x = "Is Cancelled",y = "Number of Bookings")



<u>Days in waiting and previous booking not canceled with third variable hotel:</u>

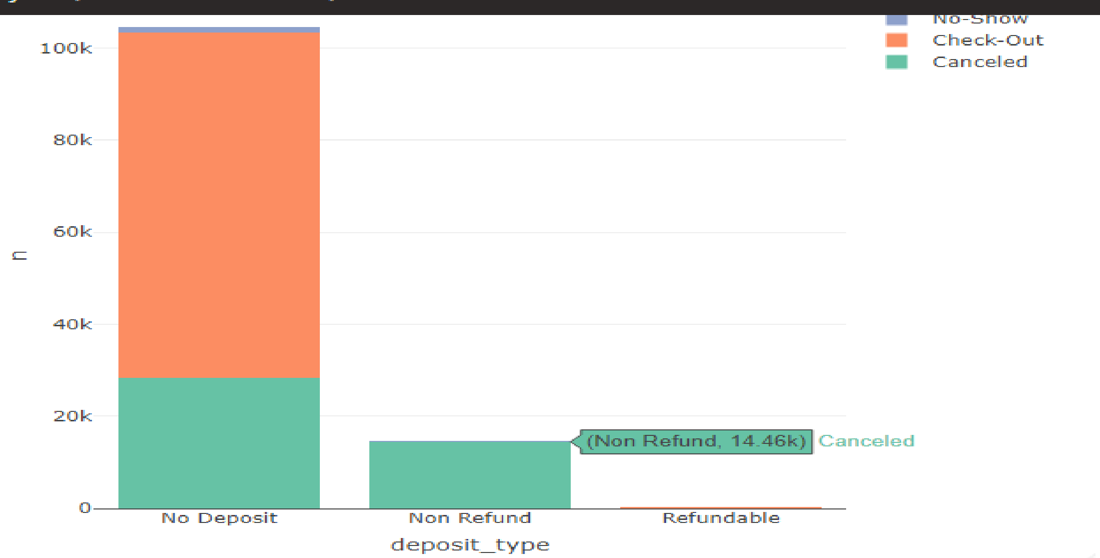
```
# Use color to add is_cancelled as a third variable
hotels%>%
  plot_ly(x = ~days_in_waiting_list, y = ~previous_bookings_not_canceled, color = ~hotel) %>%
  add_markers(colors = "Dark2",marker= list(opacity=0.2))
```



Stacked bar chart of Reservation type by Deposit type:

```
# Filter out the Reservation status and depodit type
Dep_Reserv<- hotels%>%
  filter(arrival_date_year == 2016 | arrival_date_year==2017 | arrival_date_year==2015)

#Create a stacked bar chart of Reservation type by Deposit type
Dep_Reserv %>%
  count(deposit_type, reservation_status) %>%
  plot_ly(x = ~deposit_type, y = ~n, color = ~reservation_status) %>%
  add_bars() %>%
  layout(barmode = "stack")
```



MODEL BUILDING

For this part, we predicted who is going to cancel the bookings using different model and we also try to predict whether there is the likelihood of getting request from customers. Since we are dealing with classification problem, we used accuracy to evaluate our models. We split our data set into 80% training and 20% testing and used cross validation to validate our model.

MODEL BUILDING - FOR BOOKING CANCELLATIONS

1) Multiple logistic regression

We build the model using some variables from the dataset as independent variables to predict booking cancellations. We got 80.2% accuracy with the training data and 80.4% accuracy with the testing data.

```
hotels$reservation_status_date=as.integer(hotels$reservation_status_date)

hotels$total_of_special_requests= as.factor(hotels$total_of_special_requests)

hotels$is_repeated_guest=as.factor(hotels$is_repeated_guest)

hotels$arrival_date_year=as.factor(hotels$arrival_date_year)

hotels$is_canceled=as.factor(hotels$is_canceled)

#Removing country column

hotels=hotels[-24]

hotels=hotels[-14]

dim(hotels)

#splitting dataset into training and testing data

set.seed(0)

n=nrow(hotels)

shuffled=hotels[sample(n),]

trainSet=shuffled[1:round(0.8 * n),]

testSet = shuffled[(round(0.8 * n) + 1):n,]
```

```
model1 <- glm(is_canceled ~ hotel + lead_time + arrival_date_month + children +

market_segment + is_repeated_guest + adults + babies +previous_cancellations +

deposit_type + booking_changes  +reserved_room_type + adr + days_in_waiting_list +
customer_type +                total_of_special_requests,              data = trainSet , family = "binomial")
```

```
Call:
glm(formula = is_canceled ~ hotel + lead_time + arrival_date_month +
    children + market_segment + is_repeated_guest + adults +
    babies + previous_cancellations + deposit_type + booking_changes +
    reserved_room_type + adr + days_in_waiting_list + customer_type +
    total_of_special_requests, family = "binomial", data = trainSet)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-6.1917  -0.7339  -0.4885   0.2488   3.5659

Coefficients:
                               Estimate Std. Error z value Pr(>|z|)
(Intercept)                  -2.1052709  0.1979664 -10.634  < 2e-16 ***
hotelResort Hotel            -0.1133319  0.0199076  -5.693 1.25e-08 ***
lead_time                     0.0048492  0.0001046  46.376  < 2e-16 ***
arrival_date_monthAugust     -0.0601457  0.0376634  -1.597 0.110282
arrival_date_monthDecember    0.1556932  0.0452561   3.440 0.000581 ***
arrival_date_monthFebruary    0.1677968  0.0433504   3.871 0.000109 ***
arrival_date_monthJanuary     0.0073861  0.0493215   0.150 0.880959
arrival_date_monthJuly       -0.1575822  0.0375571  -4.196 2.72e-05 ***
arrival_date_monthJune       -0.0949306  0.0395277  -2.402 0.016322 *
arrival_date_monthMarch      -0.0909921  0.0413815  -2.199 0.027888 *
arrival_date_monthMay        -0.0864177  0.0383575  -2.253 0.024262 *
arrival_date_monthNovember    0.0437475  0.0469341   0.932 0.351283
arrival_date_monthOctober    -0.0379481  0.0401504  -0.945 0.344583
arrival_date_monthSeptember  -0.2341996  0.0416251  -5.626 1.84e-08 ***
children                      0.1372670  0.0268352   5.115 3.13e-07 ***
market_segmentComplementary  -0.2249328  0.2400722  -0.937 0.348790
market_segmentCorporate      -0.4276003  0.1942914  -2.201 0.027749 *
market_segmentDirect         -0.4805377  0.1899246  -2.530 0.011401 *
market_segmentGroups          0.0181183  0.1915446   0.095 0.924640
market_segmentOffline TA/TO  -0.5674554  0.1894947  -2.995 0.002748 **
market_segmentOnline TA       0.7944610  0.1882023   4.221 2.43e-05 ***
is_repeated_guest1           -1.2262924  0.0798844 -15.351  < 2e-16 ***
adults                        0.1241837  0.0177483   6.997 2.62e-12 ***
babies                        0.1738451  0.0864878   2.010 0.044426 *
previous_cancellations        1.6797618  0.0499366  33.638  < 2e-16 ***
deposit_typeNon Refund        5.2701822  0.1167329  45.147  < 2e-16 ***
deposit_typeRefundable       -0.1156875  0.2231708  -0.518 0.604193
booking_changes              -0.4435113  0.0172807 -25.665  < 2e-16 ***
reserved_room_typeB           0.1341558  0.0839089   1.599 0.109859
reserved_room_typeC           0.1210993  0.0961643   1.259 0.207923
reserved_room_typeD          -0.0283371  0.0235383  -1.204 0.228640
reserved_room_typeE           0.0268426  0.0381860   0.703 0.482091
reserved_room_typeF          -0.4266999  0.0636914  -6.699 2.09e-11 ***
reserved_room_typeG          -0.2774592  0.0720097  -3.853 0.000117 ***
reserved_room_typeH          -0.3688352  0.1118942  -3.296 0.000980 ***
reserved_room_typeL           0.7636707  0.9404676   0.812 0.416785
reserved_room_typeP           9.3923877 43.9539860   0.214 0.830791
adr                           0.0042989  0.0002533  16.973  < 2e-16 ***
days_in_waiting_list         -0.0017368  0.0005340  -3.252 0.001145 **
customer_typeGroup           -0.3093576  0.1795966  -1.723 0.084977 .
customer_typeTransient        0.5443350  0.0556311   9.785  < 2e-16 ***
customer_typeTransient-Party  0.0527528  0.0591928   0.891 0.372820
total_of_special_requests1   -1.1777428  0.0207201 -56.840  < 2e-16 ***
total_of_special_requests2   -1.3544949  0.0291621 -46.447  < 2e-16 ***
total_of_special_requests3   -1.6674940  0.0644189 -25.885  < 2e-16 ***
total_of_special_requests4   -2.2855641  0.2078174 -10.998  < 2e-16 ***
total_of_special_requests5   -5.0224393  1.3591199  -3.695 0.000220 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)
```

Prediction with training data

train_pred <-predict(model1, trainSet,type = 'response')

library(knitr)

library(ROCR)

install.packages("verification")

library(verification)

pred <- prediction(train_pred,trainSet$is_canceled)

perform <- performance(pred,"acc")

max <- which.max(slot(perform,"y.values")[[1]])

prob <- slot(perform,"x.values")[[1]][max]

prob

train_pred1 <- ifelse(train_pred > prob, 1,0)

mean(trainSet$is_canceled == train_pred1)

tble <- table(Actual = trainSet$is_canceled,Predicted = train_pred1);tble

```
> train_pred1 <- ifelse(train_pred >  prob, 1,0)
> mean(trainSet$is_canceled == train_pred1)
[1] 0.8012657
> tble <- table(Actual = trainSet$is_canceled,Predicted = train_pred1 );tble
      Predicted
Actual     0     1
     0 55378  4333
     1 14571 20840
```

Prediction with testing data

test_pred <-predict(model1, testSet,type = 'response')

test_pred1 <- ifelse(test_pred > prob , 1,0)

#test accuracy 80.49%

mean(testSet$is_canceled == test_pred1)

tble1 <- table(Actual = testSet$is_canceled,Predicted = test_pred1);tble1

```
> mean(testSet$is_canceled == test_pred1)
[1] 0.8043734
> tble1 <- table(Actual = testSet$is_canceled,Predicted = test_pred1 );tble1
      Predicted
Actual     0     1
     0 14001  1033
     1  3619  5127
```

As we were working with a large dataset it became difficult to satisfy the assumptions of the logistic regression and we were not able to use all the variables because the execution time was also high.

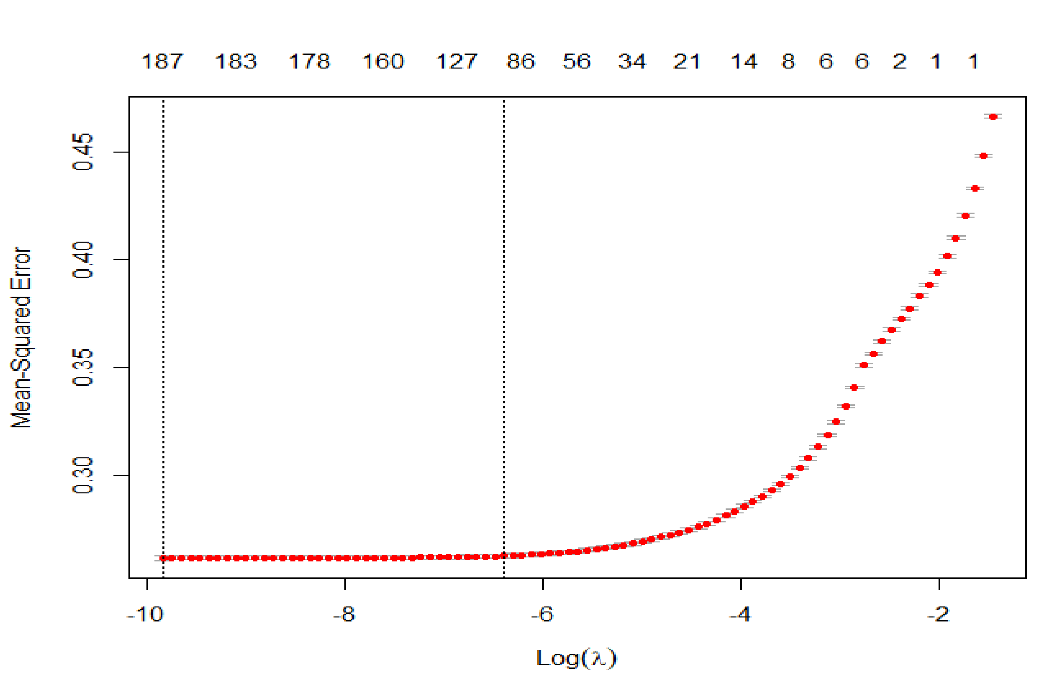
2) Lasso and Ridge regression

Model 1:

Our features will be:

lead_time , country , deposit type , adr ,arrival_date_day_of_month ,total_of_special_requests ,stays_in_weekend_nights ,stays_in_week_nights ,previous_cancelation,arrival_date_year,booking_changes ,required_car_parking_spaces and market_segment.

Using Lasso Regression:



Accuracy on training data set: 80.17%

```
#Predicting on training data set 80.17 %
#predict class, type="class"
lasso_prob <- predict(cv.out,newx = x_train1,s=lambda_1se,type="response")
#translate probabilities to predictions
lasso_predict <- rep("non_cancelled",nrow(trainSet))
lasso_predict[lasso_prob>.5] <- "canceled"
lasso_predict <- ifelse(lasso_predict=="canceled",1,0)

mean(lasso_predict==trainSet$is_canceled)
```

Accuracy on test set: 80.57 %

```
> #get test data
> #predict class, type="class"
> lasso_prob <- predict(cv.out,newx = x_test1,s=lambda_1se,type="response")
> #translate probabilities to predictions
> lasso_predict <- rep("non_cancelled",nrow(testSet))
> lasso_predict[lasso_prob>.5] <- "canceled"
> lasso_predict <- ifelse(lasso_predict=="canceled",1,0)
> mean(lasso_predict==testSet$is_canceled)
[1] 0.8057207
```

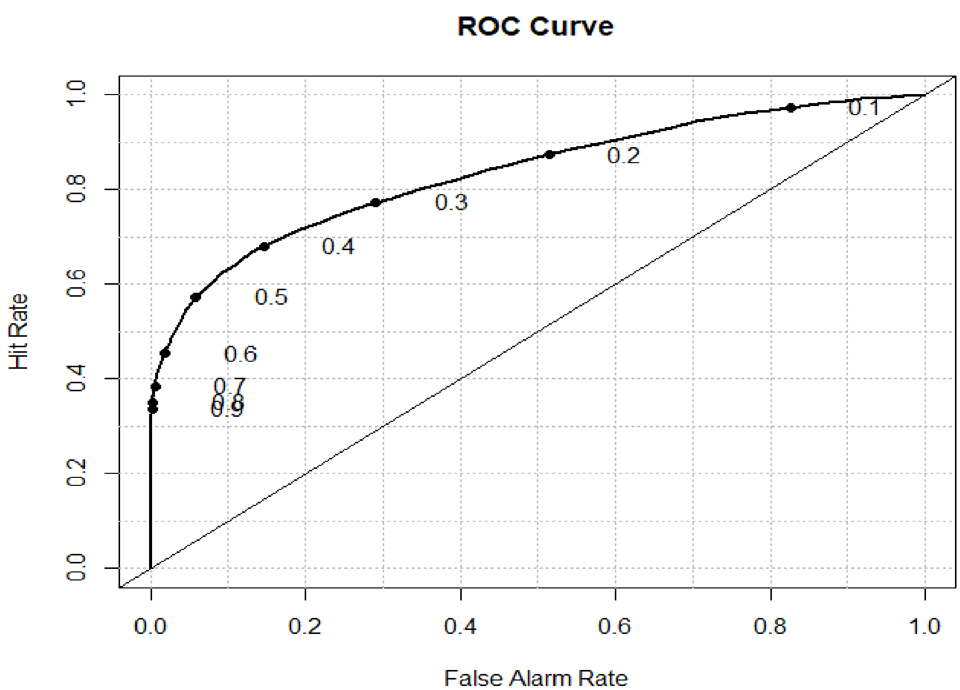
Confusion Matrix:

```
> #confusion matrix
> table(pred=lasso_predict,true=testSet$is_canceled)
    true
pred     0     1
   0 13774  3349
   1  1290  5465
```

Specificity and Sensitivity:

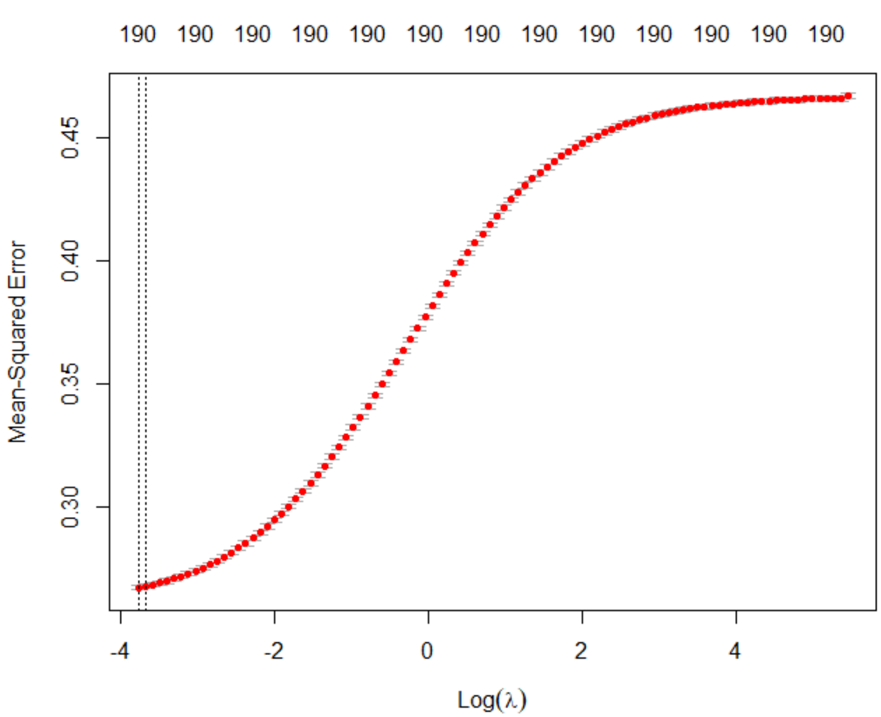
Specificity	Sensitivity
0.0214635	0.0909288

ROC:




```
> auc
[1] 0.7672008
```

Using ridge regression:



Accuracy on training data set:

```
#Predicting on training dataset 79.81%
ridge_prob <- predict(cv.out,newx = x_train1,s=lambda_1se,type="response")
#translate probabilities to predictions
ridge_predict <- rep("non_cancelled",nrow(trainSet))
ridge_predict[ridge_prob>.5] <- "canceled"
ridge_predict <- ifelse(ridge_predict=="canceled",1,0)

mean(ridge_predict==trainSet$is_canceled)
```

Accuracy on test data: 80.19%

```
> #get test data
> #predict class, type="class"
> ridge_prob <- predict(cv.out,newx = x_test1,s=lambda_1se,type="response")
> #translate probabilities to predictions
> ridge_predict <- rep("non_cancelled",nrow(testSet))
> ridge_predict[ridge_prob>.5] <- "canceled"
> ridge_predict <- ifelse(ridge_predict=="canceled",1,0)
> mean(ridge_predict==testSet$is_canceled)
[1] 0.8019097
```

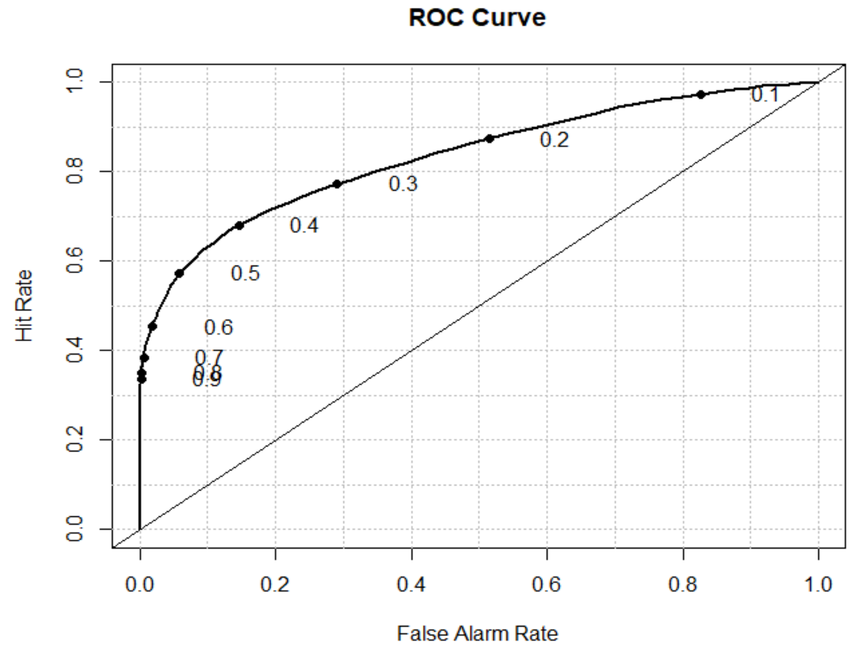
C

```
> tble1 <- table(Actual = testSet$is_canceled,Predicted = ridge_predict );tble1
      Predicted
Actual     0     1
     0 13945  1119
     1  3611  5203
```

Specificity and Sensitivity:

```
| Specificity| Sensitivity|
|-----------:|-----------:|
|   0.0186183|   0.0865695|
```

ROC Curve:



Auc:

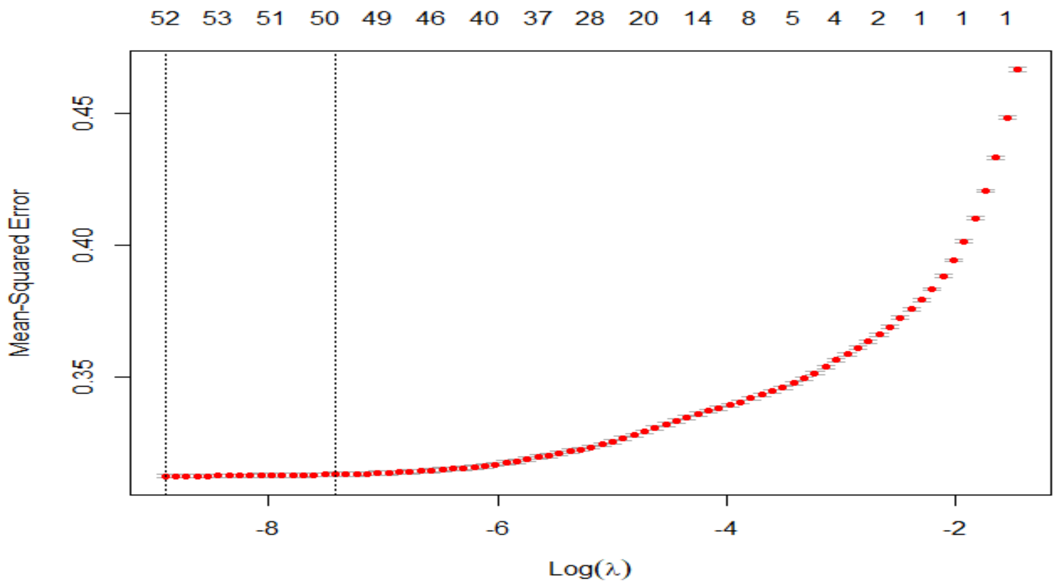
```
> auc
[1] 0.7672008
```

Model 2:

Regression with Lasso and Ridge Regression with positive coefficient and without reservation status.

Our features will be lead_time, arrival_date_year, arrival_date_month, arrival_date_week_number, arrival_date_day_of_month, stays_in_weekend_nights, stays_in_week_nights, adults, children, babies, meal, distribution_channel, is_repeated_guest , previous_cancellations, reserved_room_type, assigned_room_type, deposit_type, customer_type and adr.

Using Lasso Regression:



Accuracy on training data: 76.71 %

```
#Predict on training data set 76.71% accuracy
lasso_prob <- predict(cv.out,newx = x_train1,s=lambda_1se,type="response")
#translate probabilities to predictions
lasso_predict <- rep("non_cancelled",nrow(trainSet))
lasso_predict[lasso_prob>.5] <- "canceled"
lasso_predict <- ifelse(lasso_predict=="canceled",1,0)
```

Accuracy on test data: 76.91%

```
> #get test data
> #predict class, type="class"
> lasso_prob <- predict(cv.out,newx = x_test1,s=lambda_1se,type="response")
> #translate probabilities to predictions
> lasso_predict <- rep("non_cancelled",nrow(testSet))
> lasso_predict[lasso_prob>.5] <- "canceled"
> lasso_predict <- ifelse(lasso_predict=="canceled",1,0)
> mean(lasso_predict==testSet$is_canceled)
[1] 0.769118
```

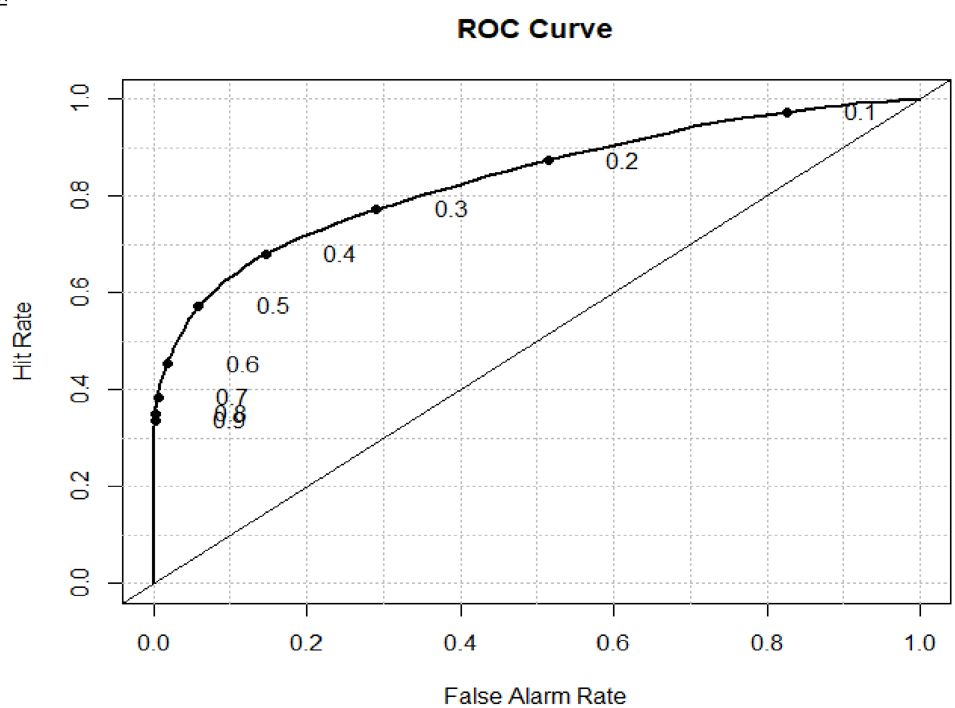
Confusion matrix:

```
> tble1 <- table(Actual = testSet$is_canceled,Predicted = lasso_predict );tble1
      Predicted
Actual     0     1
     0 14634   430
     1  5083  3731
>
```

Specificity and Sensitivity:

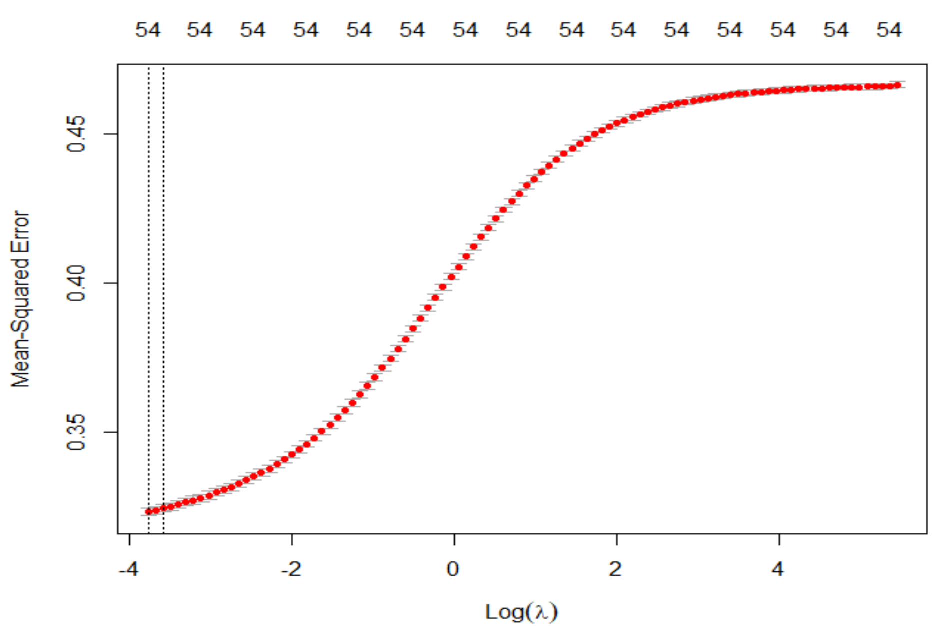
```
| specificity| sensitivity|
|-----------:|-----------:|
|   0.0071545|   0.0620778|
```

ROC curve:




```
> auc
[1] 0.6973795
```

Using Ridge Regression:



Accuracy on training data set: 75.81%

```
#Predicting on training data 75.81 %
#predict class, type="class"
ridge_prob <- predict(cv.out,newx = x_train1,s=lambda_1se,type="response")
#translate probabilities to predictions
ridge_predict <- rep("non_cancelled",nrow(trainSet))
ridge_predict[ridge_prob>.5] <- "canceled"
ridge_predict <- ifelse(ridge_predict=="canceled",1,0)

mean(ridge_predict==trainSet$is_canceled)
```

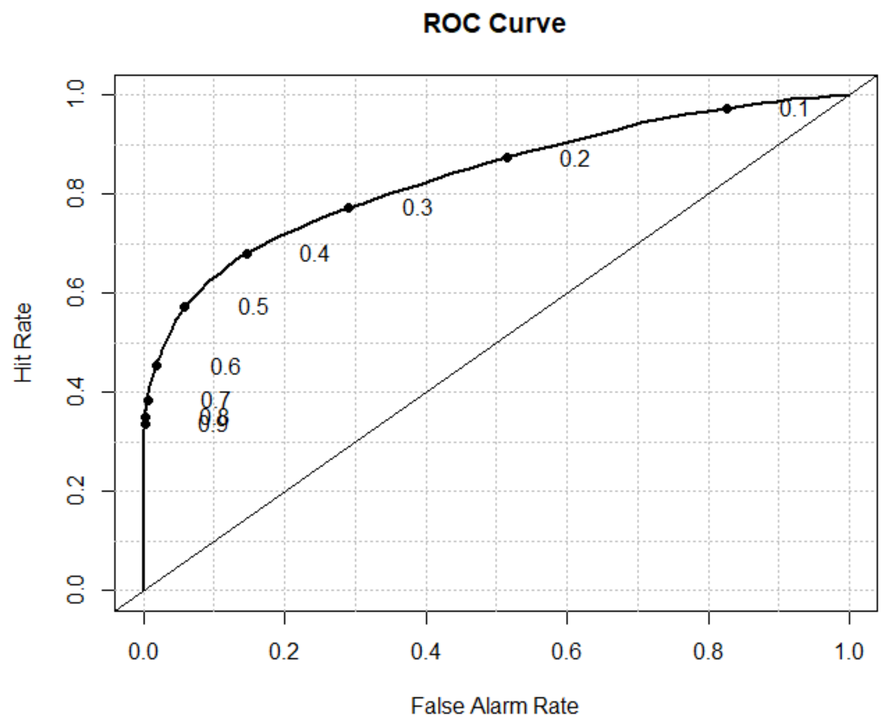
Accuracy on test data: 76.15 %

```
> #get test data
> #predict class, type="class"
> ridge_prob <- predict(cv.out,newx = x_test1,s=lambda_1se,type="response")
> #translate probabilities to predictions
> ridge_predict <- rep("non_cancelled",nrow(testSet))
> ridge_predict[ridge_prob>.5] <- "canceled"
> ridge_predict <- ifelse(ridge_predict=="canceled",1,0)
> mean(ridge_predict==testSet$is_canceled)
> tble1 <- table(Actual = testSet$is_canceled,Predicted = ridge_predict );tble1
      Predicted
Actual     0     1
     0 14741   323
     1  5370  3444
```

Specificity and Sensitivity:

```
| Specificity| Sensitivity|
|-----------:|-----------:|
|   0.0053742|   0.0573026|
```

<u>ROC curve:</u>



<u>Auc:</u>

```
> auc
[1] 0.6846501
```

3) Random forest

We also used advanced machine learning algorithm to predict the booking cancellations. Model was build using all the independent variables except for the reservation status and date column to predict the booking cancellations. 80% of the data as training data and 20% as testing.

Number of decision trees were 500 and the variable at each split was 3. We got 99% accuracy for the training dataset and 93.9% accuracy in predicting the model with testing dataset.

install.packages("randomForest")

library(randomForest)

sapply(hotels, class)

model1=randomForest(is_canceled~.-reservation_status -arrival_date_year -arrival_date_month,data=trainSet)

model1

```
Call:
 randomForest(formula = is_canceled ~ . - reservation_status
 -arrival_date_year - arrival_date_month, data = trainSet)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 5

        OOB estimate of  error rate: 6.19%
Confusion matrix:
      0     1 class.error
0 58938   773  0.01294569
1  5113 30298  0.14439016
```

<u>prediction and confusion matrix for training data</u>

modpredTrain=predict(model1,trainSet)

confusionMatrix(modpredTrain,trainSet$is_canceled)

```
          Reference
Prediction     0     1
         0 59686   341
         1    25 35070

               Accuracy : 0.9962
                 95% CI : (0.9957, 0.9965)
    No Information Rate : 0.6277
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9918

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9996
            Specificity : 0.9904
         Pos Pred Value : 0.9943
         Neg Pred Value : 0.9993
             Prevalence : 0.6277
         Detection Rate : 0.6275
   Detection Prevalence : 0.6311
      Balanced Accuracy : 0.9950

       'Positive' Class : 0
```

prediction and confusion matrix for testing data

modpredTest=predict(model1,testSet)

confusionMatrix(modpredTest,testSet$is_canceled)

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 14849  1266
         1   185  7480

               Accuracy : 0.939
                 95% CI : (0.9359, 0.942)
    No Information Rate : 0.6322
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8653

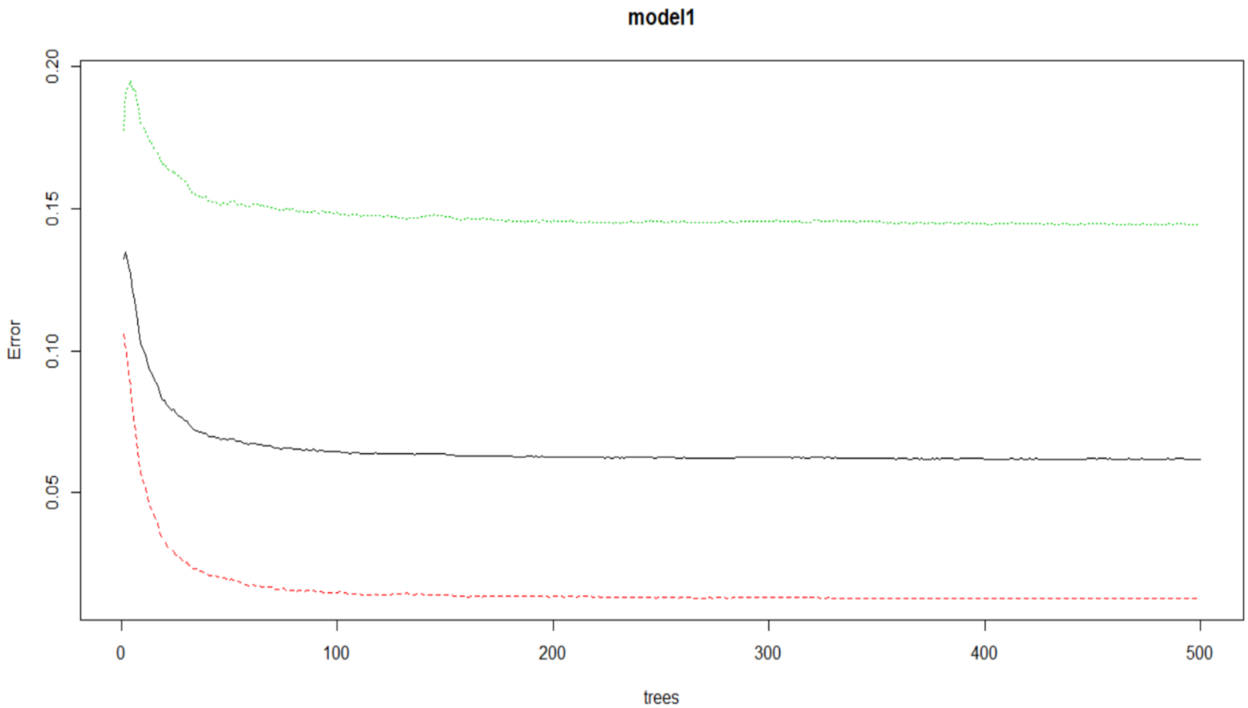
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9877
            Specificity : 0.8552
         Pos Pred Value : 0.9214
         Neg Pred Value : 0.9759
             Prevalence : 0.6322
         Detection Rate : 0.6244
   Detection Prevalence : 0.6777
      Balanced Accuracy : 0.9215

       'Positive' Class : 0
……
```

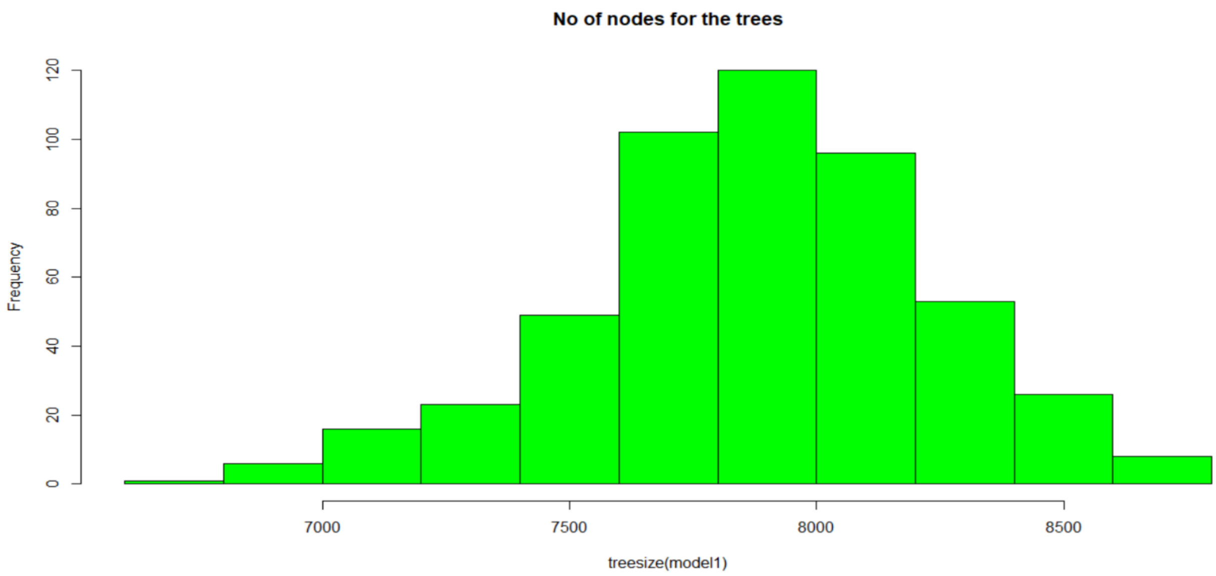
Error rate

plot(model1)



Number of nodes for the trees

hist(treesize(model1),main = "No of nodes for the trees",col = "green")



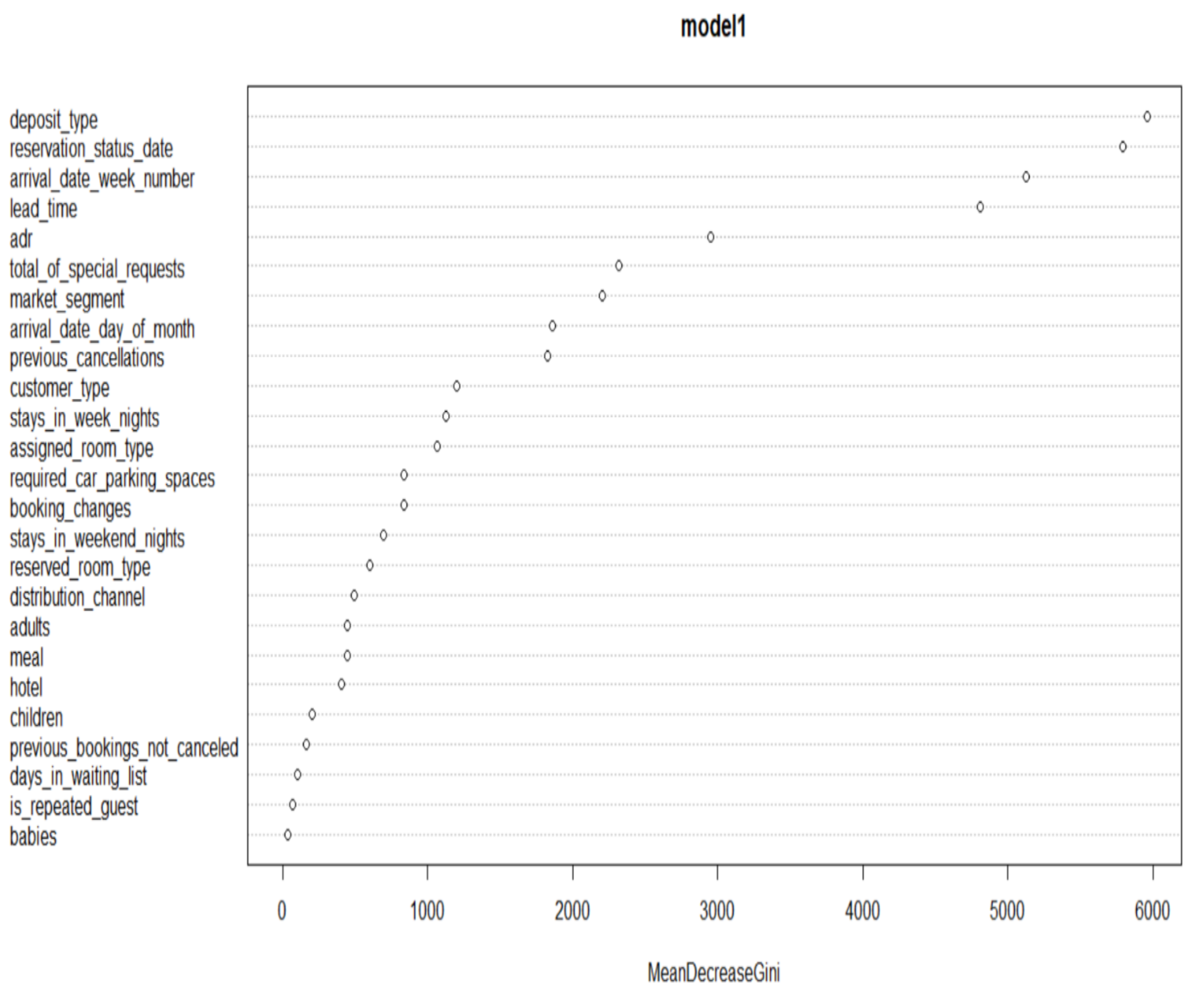
<u>variable importance</u>

varImpPlot(model1)

varUsed(model1)

[1] 71223 462910 449985 428817 211066 288381 123166 59846 11145 107109 74354 44110 11339 14506

[15] 17800 112886 141767 76621   7178   9005 52062 472371 19845 99143 572571



Random forest is the best model for predicting the booking cancellations among all other model as we got 93.9% accuracy with this model.

MODEL BUILDING - FOR SPECIAL REQUESTS

1. Multiple logistic regression

We build the model using some variables from the dataset as independent variables to predict special requests. We got 74% accuracy with the training data and 73% accuracy with the testing data.

```
hotels$total_of_special_requests=as.numeric(hotels$total_of_special_requests)

for (i in 1:118902) {

  if (hotels$total_of_special_requests[i]>1){

    hotels$total_of_special_requests[i]=(hotels$total_of_special_requests[i]= 1)

  }

}

hotels$total_of_special_requests= as.factor(hotels$total_of_special_requests)

hotels$total_of_special_requests

mod<- glm(total_of_special_requests ~ is_canceled + lead_time + stays_in_weekend_nights+stays_in_week_nights +is_repeated_guest + adults + babies + days_in_waiting_list +market_segment+ deposit_type+ customer_type, data = trainSet, family='binomial')
```

summary(mod)

```
Call:
glm(formula = total_of_special_requests ~ is_canceled + lead_time +
    stays_in_weekend_nights + stays_in_week_nights + is_repeated_guest +
    adults + babies + days_in_waiting_list + market_segment +
    deposit_type + customer_type, family = "binomial", data = trainSet)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-5.4566  -0.8721  -0.0668   0.8372   3.6781

Coefficients:
                               Estimate Std. Error z value Pr(>|z|)
(Intercept)                  -2.5803294  0.3052611  -8.453  < 2e-16 ***
is_canceled1                 -1.2797636  0.0190109 -67.317  < 2e-16 ***
lead_time                     0.0024889  0.0000978  25.449  < 2e-16 ***
stays_in_weekend_nights       0.0326657  0.0091684   3.563 0.000367 ***
stays_in_week_nights          0.0192225  0.0047845   4.018 5.88e-05 ***
is_repeated_guest1            0.6914810  0.0469052  14.742  < 2e-16 ***
adults                        0.2978544  0.0169886  17.533  < 2e-16 ***
babies                        1.7928183  0.1023103  17.523  < 2e-16 ***
days_in_waiting_list         -0.0057976  0.0007863  -7.373 1.67e-13 ***
market_segmentComplementary   2.8388618  0.3130580   9.068  < 2e-16 ***
market_segmentCorporate       1.1626729  0.3035647   3.830 0.000128 ***
market_segmentDirect          1.9968893  0.3019391   6.614 3.75e-11 ***
market_segmentGroups          0.6750343  0.3038173   2.222 0.026294 *
market_segmentOffline TA/TO   1.3286865  0.3020940   4.398 1.09e-05 ***
market_segmentOnline TA       3.2839159  0.3015898  10.889  < 2e-16 ***
deposit_typeNon Refund       -4.0343170  0.2154910 -18.722  < 2e-16 ***
deposit_typeRefundable       -0.8510032  0.2942588  -2.892 0.003828 **
customer_typeGroup           -0.5379469  0.1156366  -4.652 3.29e-06 ***
customer_typeTransient       -0.5442992  0.0444439 -12.247  < 2e-16 ***
customer_typeTransient-Party -0.5992910  0.0479859 -12.489  < 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: 128732  on 95121  degrees of freedom
Residual deviance:  96147  on 95102  degrees of freedom
AIC: 96187

Number of Fisher Scoring iterations: 8
```

install.packages("boot")

library(boot)

set.seed(0)

cv_results=cv.glm(na.omit(trainSet), mod, K=10)

cv_results$delta

Prediction and confusion matrix for training data

prediction=predict(mod,trainSet, type='response')

prediction1= ifelse(prediction>0.5,1,0)

tab1= table(prediction1,trainSet$total_of_special_requests)

tab1

1-sum(diag(tab1))/sum(tab1)

```
> tab1= table(prediction1,trainSet$total_of_special_requests)
> tab1

prediction1     0     1
          0 45323 14287
          1 10848 24664
> 1-sum(diag(tab1))/sum(tab1)
[1] 0.2642396
```

Pridiction and confusion matrix for test data

prediction2=predict(mod,testSet, type='response')

prediction3= ifelse(prediction2>0.5,1,0)

tab2= table(prediction3,testSet$total_of_special_requests)

1-sum(diag(tab2))/sum(tab2)

```
> tab2= table(prediction3,testSet$total_of_special_requests)
> tab2

prediction3     0     1
          0 11074  3672
          1  2746  6288
> 1-sum(diag(tab2))/sum(tab2)
[1] 0.2698907
```

As we were working with a large dataset it became difficult to satisfy the assumptions of the logistic regression and we were not able to use all the variables because the execution time was also high.

2 Lasso and Ridge regression

Model was build using all the independent variables except for the reservation status and date column to predict the special request. We also performed cross validation.

x_train1 = model.matrix(trainSet$total_of_special_requests~.-reservation_status -arrival_date_year -arrival_date_month, trainSet)[,-27]

y_train1 = trainSet$total_of_special_requests

x_test1=model.matrix(testSet$total_of_special_requests~.-reservation_status -arrival_date_year -arrival_date_month, testSet)[,-27]

y_test1=testSet$total_of_special_requests

Using Lasso regression:

We got 73.8% accuracy in predicting the model with training dataset and 73.3% with testing dataset.We also prepared confusion matrix to evaluate the performance of the model.

cv.out <- cv.glmnet(x_train1,y_train1,alpha=1,family="binomial",type.measure = "mse")

cv.out

```
> cv.out

Call:  cv.glmnet(x = x_train1, y = y_train1, type.measure = "mse", alpha = 1,      family = "binomial")

Measure: Mean-Squared Error

       Lambda Measure       SE Nonzero
min 0.0002156  0.3381 0.001246      49
1se 0.0024223  0.3393 0.001190      36
```

Minimum value of lambda

lambda_min <- cv.out$lambda.min

Best value of lambda

lambda_1se <- cv.out$lambda.1se

Regression coefficients

coef(cv.out,s=lambda_1se)

Prediction with training dataset

lasso_prob <- predict(cv.out,newx = x_train1,s=lambda_1se,type="response")

#translate probabilities to predictions

lasso_predict <- rep("No Request",nrow(trainSet))

lasso_predict[lasso_prob>.5] <- "Request"

lasso_predict <- ifelse(lasso_predict=="Request",1,0)

mean(lasso_predict==trainSet$total_of_special_requests)

tab= table(lasso_predict, trainSet$total_of_special_requests)

tab

1- sum(diag(tab))/sum(tab)

```
> mean(lasso_predict==trainSet$total_of_special_requests)
[1] 0.7383991
> tab= table(lasso_predict, trainSet$total_of_special_requests)
> tab

lasso_predict     0     1
            0 45616 14329
            1 10555 24622
```

Prediction with testing dataset

lasso_prob <- predict(cv.out,newx = x_test1,s=lambda_1se,type="response")

#translate probabilities to predictions

lasso_predict <- rep("No Request",nrow(testSet))

lasso_predict[lasso_prob>.5] <- "Request"

lasso_predict <- ifelse(lasso_predict=="Request",1,0)

mean(lasso_predict==testSet$total_of_special_requests)

confusion matrix

tab1= table(lasso_predict, testSet$total_of_special_requests)

tab

1- sum(diag(tab1))/sum(tab1)

```
> mean(lasso_predict==testSet$total_of_special_requests)
[1] 0.7337679
> # confusion matrix
> tab1= table(lasso_predict, testSet$total_of_special_requests)
> tab

lasso_predict     0     1
            0 45616 14329
            1 10555 24622
```

Using Ridge regression:

We got 73.9% accuracy in predicting the model with training dataset and 73.4% with testing dataset.We also prepared confusion matrix to evaluate the performance of the model.

cv.out <- cv.glmnet(x_train1,y_train1,alpha=0,family="binomial",type.measure = "mse")

cv.out

```
> cv.out

Call:  cv.glmnet(x = x_train1, y = y_train1, type.measure = "mse", alpha = 0,      family = "binomial")

Measure: Mean-Squared Error

     Lambda Measure        SE Nonzero
min 0.02107  0.3401 0.0007795      54
1se 0.02538  0.3406 0.0007669      54
```

Minimum value of lambda

lambda_min <- cv.out$lambda.min

Best value of lambda

lambda_1se <- cv.out$lambda.1se

Regression coefficients

coef(cv.out,s=lambda_1se)

Prediction with training dataset

ridge_prob <- predict(cv.out,newx = x_train1,s=lambda_1se,type="response")

#translate probabilities to predictions

ridge_predict <- rep("No Request",nrow(trainSet))

ridge_predict[ridge_prob>.5] <- "Request"

ridge_predict <- ifelse(ridge_predict=="Request",1,0)

mean(ridge_predict==trainSet$total_of_special_requests)

tab= table(ridge_predict, trainSet$total_of_special_requests)

tab

1- sum(diag(tab))/sum(tab)

```
> mean(ridge_predict==trainSet$total_of_special_requests)
[1] 0.7392822
> tab= table(ridge_predict, trainSet$total_of_special_requests)
> tab

ridge_predict     0     1
            0 45526 14155
            1 10645 24796
```

Prediction with testing dataset

ridge_prob <- predict(cv.out,newx = x_test1,s=lambda_1se,type="response")

#translate probabilities to predictions

ridge_predict <- rep("No Request",nrow(testSet))

ridge_predict[ridge_prob>.5] <- "Request"

ridge_predict <- ifelse(ridge_predict=="Request",1,0)

mean(ridge_predict==testSet$total_of_special_requests)

tab1= table(ridge_predict, testSet$total_of_special_requests)

tab1

1- sum(diag(tab1))/sum(tab1)

```
> mean(ridge_predict==testSet$total_of_special_requests)
[1] 0.7341463
> tab1= table(ridge_predict, testSet$total_of_special_requests)
> tab1

ridge_predict     0     1
            0 11169  3671
            1  2651  6289
```

3 Random forest

We also used advanced machine learning algorithm to predict the booking cancellations. Model was build using all the independent variables except for the reservation status and date column to predict the booking cancellations. 80% of the data as training data and 20% as testing.

Number of decision trees were 500 and the variable at each split was 3. We got 93% accuracy in predicting the model with testing dataset.

Here we also tried to tune the model by increasing the number of variables at each split.

model=randomForest(total_of_special_requests ~.,data = trainSet, importance = TRUE)

model

```
> model=randomForest(total_of_special_requests ~.,data = trainSet, importance = TRUE)
> model

Call:
 randomForest(formula = total_of_special_requests ~ ., data = trainSet,      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 5

        OOB estimate of  error rate: 16.84%
Confusion matrix:
      0     1 class.error
0 47373  8798   0.1566289
1  7220 31731   0.1853611
```

Prediction and confusion matrix for training data

install.packages("caret")

install.packages("e1071")

library(caret)

library(e1071)

p1=predict(model,trainSet)

confusionMatrix(p1,trainSet$total_of_special_requests)

```
> p1=predict(model,trainSet)
> confusionMatrix(p1,trainSet$total_of_special_requests)
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 55386   695
         1   785 38256

               Accuracy : 0.9844
                 95% CI : (0.9836, 0.9852)
    No Information Rate : 0.5905
    P-Value [Acc > NIR] : <2e-16
```

<u>Prediction and confusion matrix for testing data</u>

p2=predict(model,testSet)

confusionMatrix(p2,testSet$total_of_special_requests

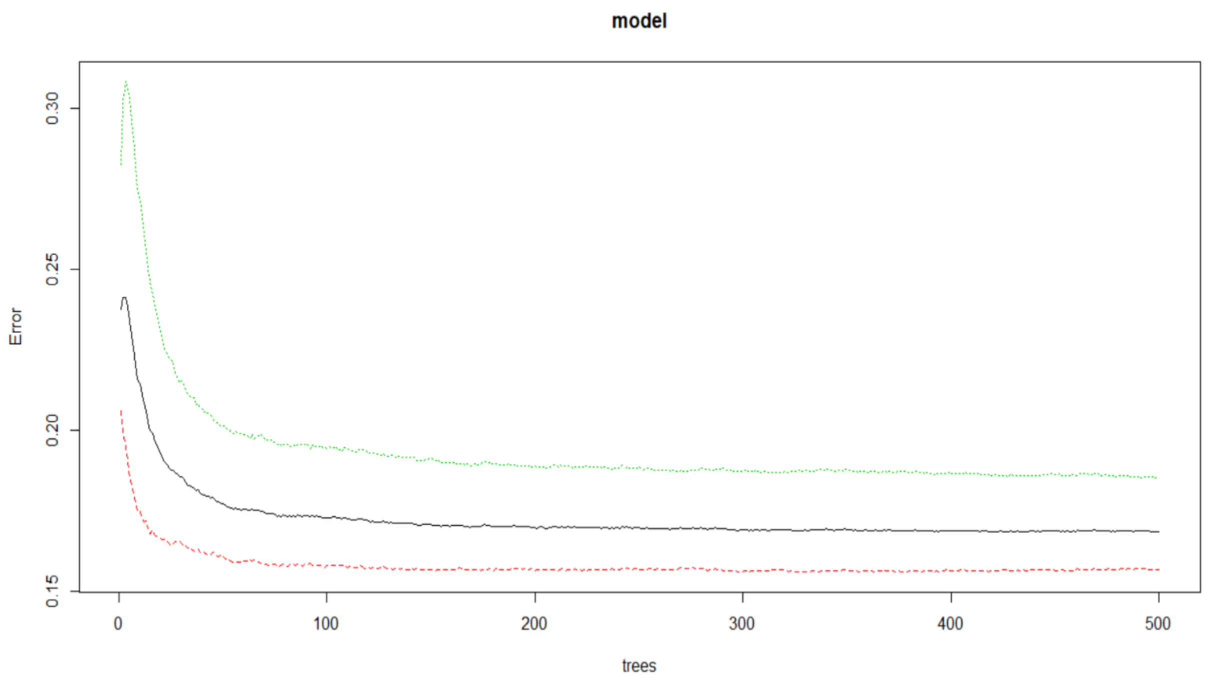
```
> #prediction and confusion matrix for testing data
> p2=predict(model,testSet)
> confusionMatrix(p2,testSet$total_of_special_requests)
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 11651  1860
         1  2169  8100

               Accuracy : 0.8306
                 95% CI : (0.8257, 0.8353)
    No Information Rate : 0.5812
    P-Value [Acc > NIR] : < 2.2e-16
```

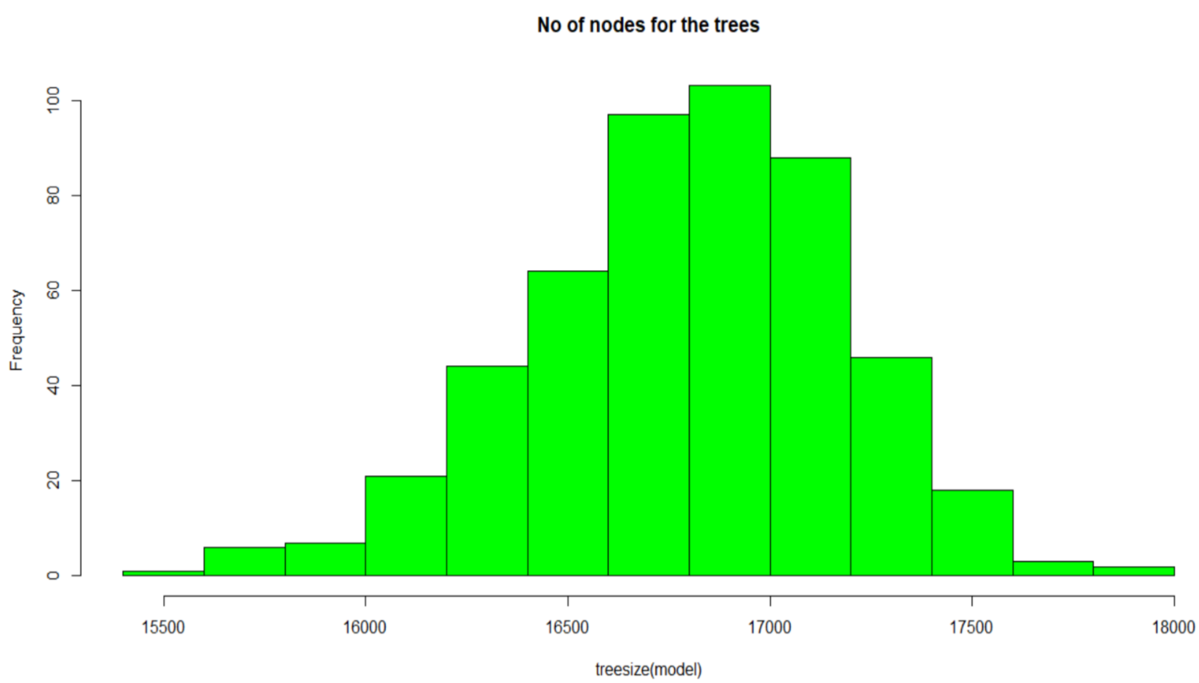
<u>Error rate</u>

plot(model)



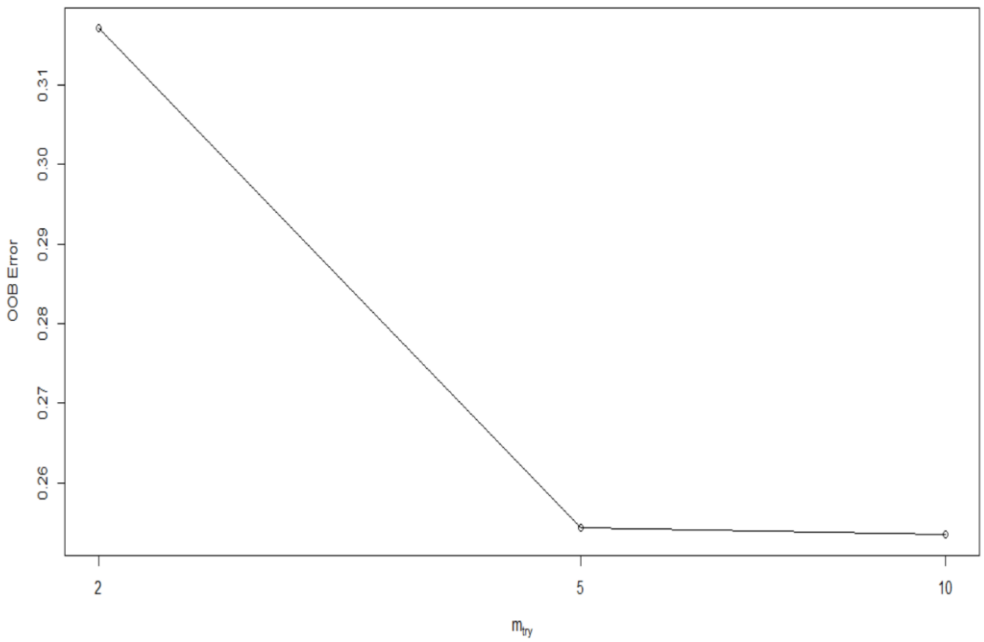
Number of nodes for the trees

hist(treesize(model),main = "No of nodes for the trees",col = "green")



Tuning model

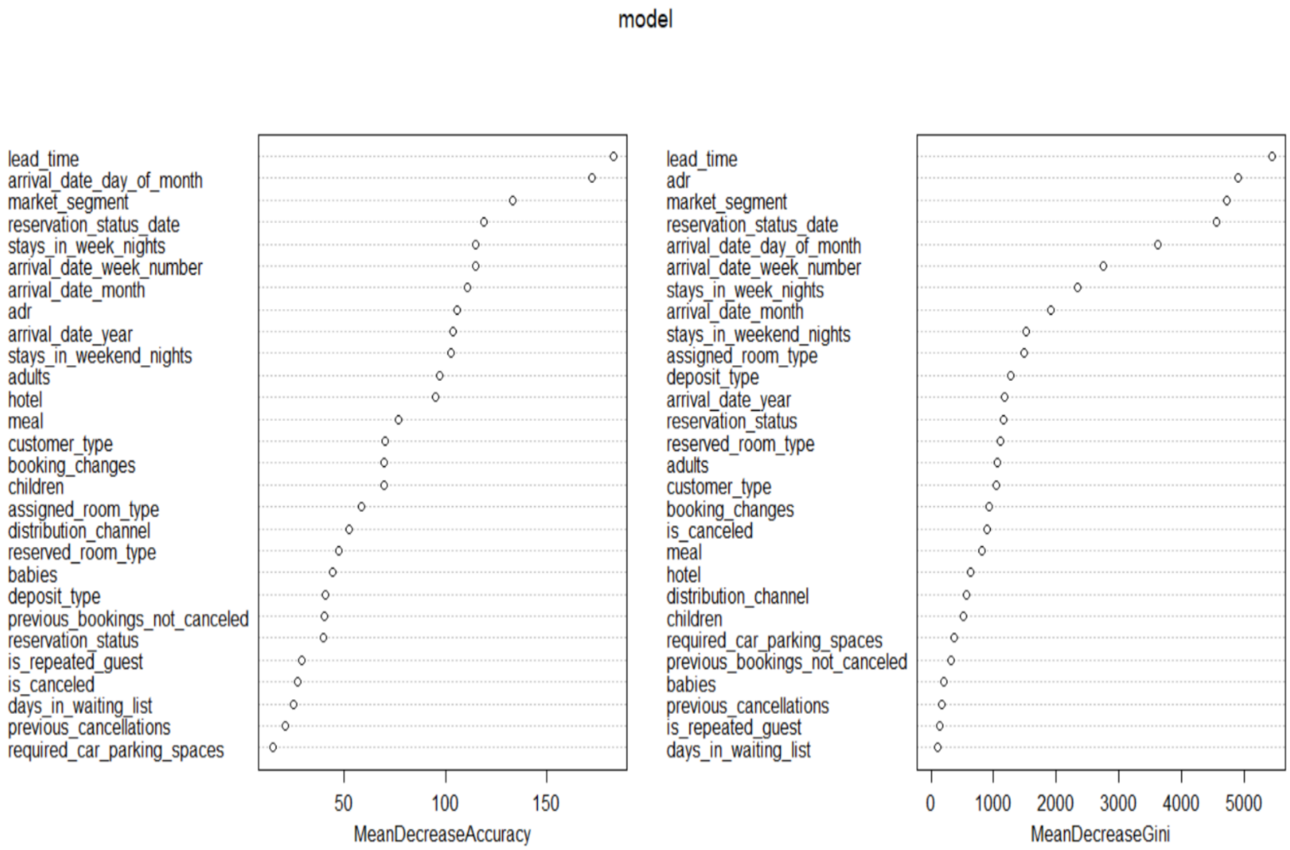
tuneRF(trainSet[,-27],trainSet[,27],stepFactor = 0.5,plot=TRUE,ntreeTry = 300,trace=TRUE,improve = 0.05)




```
> tuneRF(trainSet[,-27],trainSet[,27],stepFactor = 0.5,plot=TRUE,ntreeTry = 300,trace=TRUE,improve = 0.05)
mtry = 5  OOB error = 16.86%
Searching left ...
mtry = 10 	OOB error = 16.72%
0.008353594 0.05
Searching right ...
mtry = 2 	OOB error = 20.73%
-0.2293498 0.05
       mtry  OOBError
2.OOB     2 0.2073127
5.OOB     5 0.1686361
10.OOB   10 0.1672274
```

variable importance

varImpPlot(model)



Random forest is the best model for predicting the special requests among all other model as we got 83% accuracy with this model.

TIME SERIES ANALYSIS FOR BOOKINGS:

Given the data of bookings and cancellations from July 2015 to Sep 2017, time series analysis was performed for the number of bookings based on the column reservation_status_date.

Time series analysis is performed using MLR with a quadratic nonlinear model:

```
#Time series Analysis with Seasonal Component using LR

#Filter Bookings which are checkout and group by month
```{r}
hdd =  hotelData %>%
  filter(reservation_status=='Check-Out') %>%
  group_by(reservation_status_date,arrival_date_month )%>%
  summarise(n=n())

hdd$reservation_status_date = as.Date(hdd$reservation_status_date)

hd = hdd %>% group_by(Date=floor_date(reservation_status_date, "month")) %>%
  summarise(NumberOfBookings=sum(n)) %>%
 mutate(Month = month(Date)) %>%
  add_column(Timeperiod = 0 : 26)

hd$Month = as.factor(hd$Month)
hd = hd[-27,]
hd = hd[,c(1,4,3,2)]
hd
```
```

Date <date>	Timeperiod <int>	Month <fctr>	NumberOfBookings <int>
2015-07-01	0	7	1321
2015-08-01	1	8	2224
2015-09-01	2	9	2986
2015-10-01	3	10	3304
2015-11-01	4	11	1987
2015-12-01	5	12	1640
2016-01-01	6	1	1985
2016-02-01	7	2	2435
2016-03-01	8	3	3194
2016-04-01	9	4	3347

1-10 of 26 rows   Previous 1 2 3 Next

Different models were validated and found the below quadratic model to be the best satisfying assumptions of MLR , where equation is as follows:

NumberofBookings = 1289 + 150.27*Timeperiod – 4.348*Timeperiod2 + (Month as based)

````
#Model | : Quadratic Non Linear Regression
```{r}

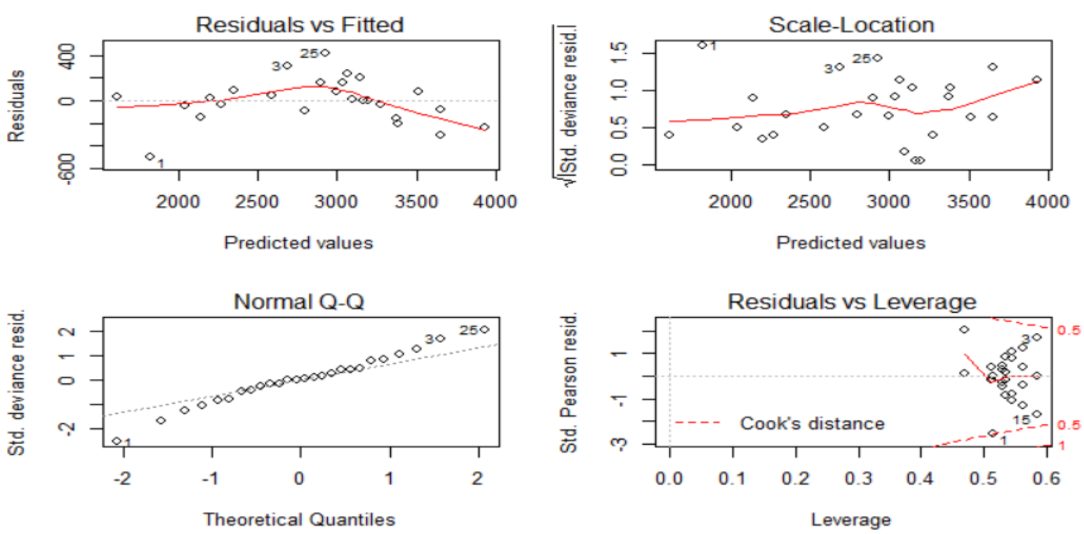
hd$Timeperiod2 = hd$Timeperiod*hd$Timeperiod
Qmodel2 = glm(NumberOfBookings ~ Timeperiod  + Timeperiod2 + Month , data=hd)
summary(Qmodel2)
layout(matrix(c(1,2,3,4),2,2))
plot(Qmodel2)

```
````


```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1289.602    263.545   4.893 0.000370 ***
Timeperiod   150.270     31.632   4.751 0.000472 ***
Timeperiod2   -4.349      1.227  -3.546 0.004027 ** 
Month2       218.458    279.761   0.781 0.450007    
Month3       820.615    280.095   2.930 0.012609 *  
Month4       855.970    280.719   3.049 0.010100 *  
Month5      1159.023    281.759   4.114 0.001437 ** 
Month6       752.275    283.401   2.654 0.021006 *  
Month7       529.949    265.683   1.995 0.069300 .  
Month8       763.074    265.797   2.871 0.014066 *  
Month9      1110.150    282.441   3.931 0.001997 ** 
Month10     1365.815    281.008   4.860 0.000391 ***
Month11      318.678    280.181   1.137 0.277572    
Month12     -323.260    279.772  -1.155 0.270398    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 78205.91)
```

Residual Plots for Quadratic Non Linear Model:



Validation of Quadratic Non linear Model and the test parameters:

```
library(caret)
install.packages("boot")
install.packages("carData")
library(boot)
library(carData)
library(car)
set.seed(4)
n=nrow(hd)
shuffled=hd[sample(n),]
train=shuffled[1:round(0.85 * n),]
test = shuffled[(round(0.85 * n) + 1):n,]

# Validation with Traning Data

Qmodel2 = glm(NumberOfBookings ~ Timeperiod  + Timeperiod2 + Month , data=hd)
summary(Qmodel2)
plot(Qmodel2)

#Prediction
prediction=predict.lm(Qmodel2,newdata=test)
prediction
test$NumberOfBookings

> prediction
1        2        3        4
3375.965 3651.104 3034.035 2264.035
> test$NumberOfBookings
[1] 3216 3348 3194 2233
>
```

```
#Compute metrics R2, RMSE, MAE

R2(prediction, test$NumberOfBookings)
RMSE(prediction, test$NumberOfBookings)
MAE(prediction, test$NumberOfBookings)

""""
> R2(prediction, test$NumberOfBookings)
[1] 0.9012691
> RMSE(prediction, test$NumberOfBookings)
[1] 189.7452
> MAE(prediction, test$NumberOfBookings)
[1] 163.5173
>
""""
```

TIME SERIES ANALYSIS USING FORECASTS MODEL:

ARIMA and HoltWinters forecasts are used to predict the future bookings .

````
#Create TimeSeries for seasonal data
#hs = hotel data seasonal
```{r}

n = length(hd$NumberOfBookings)
l = 2


hs = ts(hd$NumberOfBookings, start=c(2015, 7), end=c(2017,8), frequency= 12)

trainhs = ts(hd$NumberOfBookings[1: (n-l)], start=c(2015, 7) ,frequency= 12)
tesths  = ts(hd$NumberOfBookings[(n-l+1) : n], end=c(2017,8) ,frequency= 12)
hs

```
````

```
      Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
2015                                1321 2224 2986 3304 1987 1640
2016 1985 2435 3194 3347 3593 3168 3080 3240 3348 3694 3052 2233
2017 2635 2705 3216 3182 3573 3198 3336 3097
```

````
#Test for stationary time series
```{r}


adf.test(hs)
kpss.test(hs)

```
````

```
	Augmented Dickey-Fuller Test

data:  hs
Dickey-Fuller = -3.2251, Lag order = 2, p-value = 0.1057
alternative hypothesis: stationary

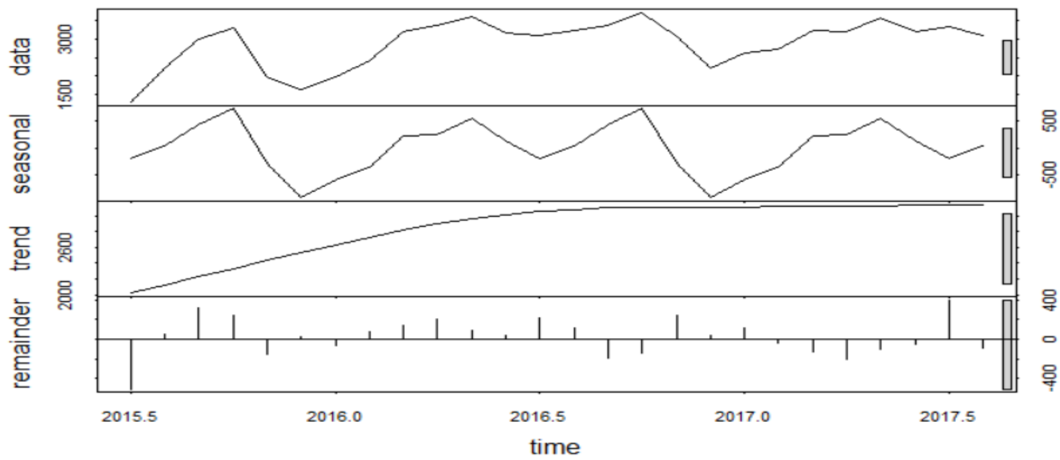

	KPSS Test for Level Stationarity

data:  hs
KPSS Level = 0.42882, Truncation lag parameter = 2, p-value = 0.06473
```

````
#See the components of time series
```{r}

components = stl(hs, 'periodic')
plot(components)

```
````



A clear increasing trend is seen for yearly data with a seasonal pattern observed.

1. MODEL USING ARIMA:

````
#Auto.arima
```{r}

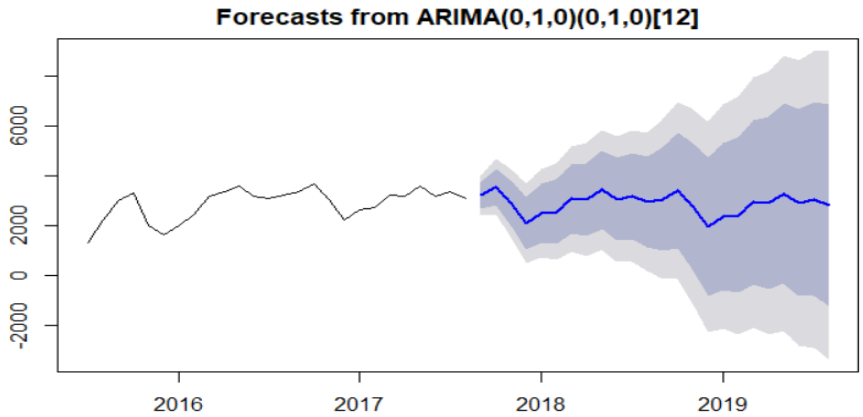
plot(forecast(auto.arima(hs)), sub = "Simple plot to forecast")

Afit = auto.arima(hs, trace=TRUE)

checkresiduals(Afit)
Aforecast = forecast(Afit)

accuracy(Aforecast)


```
````



Simple plot to forecast

```
 ARIMA(2,1,2)(0,1,0)[12]                    : Inf
 ARIMA(0,1,0)(0,1,0)[12]                    : 195.3849
 ARIMA(1,1,0)(0,1,0)[12]                    : 198.1213
 ARIMA(0,1,1)(0,1,0)[12]                    : 198.1003
 ARIMA(1,1,1)(0,1,0)[12]                    : 201.5643

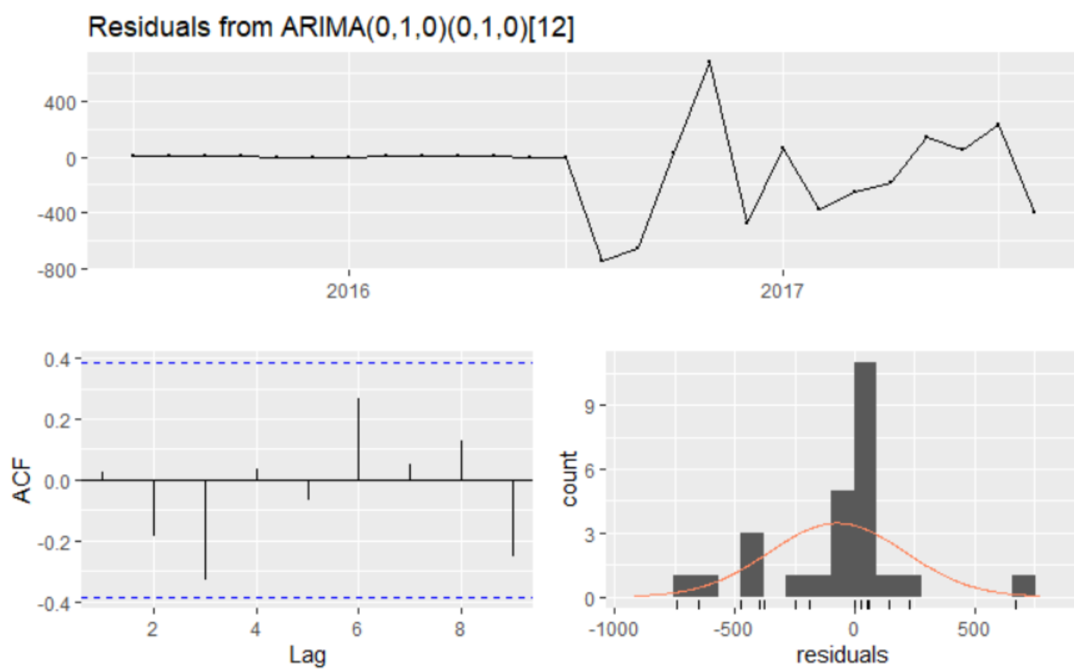
 Best model: ARIMA(0,1,0)(0,1,0)[12]


        Ljung-Box test

data:  Residuals from ARIMA(0,1,0)(0,1,0)[12]
Q* = 4.6398, df = 5, p-value = 0.4614

Model df: 0.   Total lags used: 5

                    ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
Training set -72.98426 286.6851 164.4508 -2.558678 5.461286 0.3415385 0.02412717
```



Future bookings forecast :

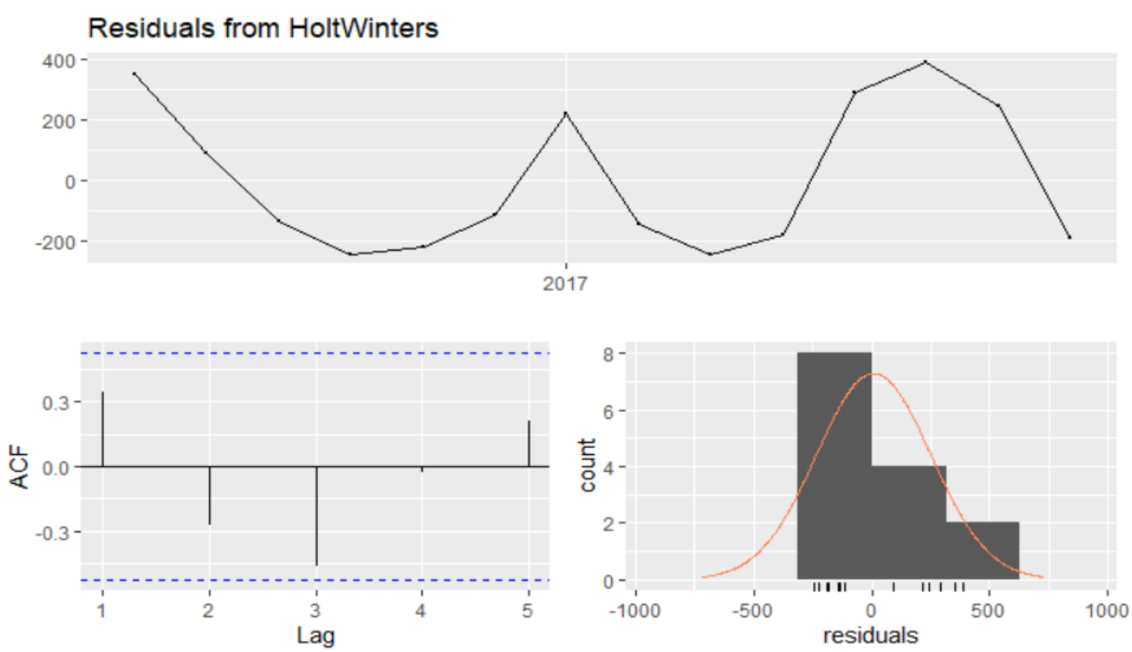
	Point Forecast <dbl>	Lo 80 <dbl>	Hi 80 <dbl>	Lo 95 <dbl>	Hi 95 <dbl>
Sep 2017	3205	2685.4155	3724.585	2410.3641	3999.636
Oct 2017	3551	2816.1965	4285.803	2427.2151	4674.785
Nov 2017	2909	2009.0532	3808.947	1532.6502	4285.350
Dec 2017	2090	1050.8310	3129.169	500.7281	3679.272
Jan 2018	2492	1330.1737	3653.826	715.1400	4268.860
Feb 2018	2562	1289.2831	3834.717	615.5474	4508.453
Mar 2018	3073	1698.3086	4447.691	970.5909	5175.409
Apr 2018	3039	1569.3931	4508.607	791.4302	5286.570
May 2018	3430	1871.2465	4988.754	1046.0922	5813.908
Jun 2018	3055	1411.9295	4698.070	542.1405	5567.859

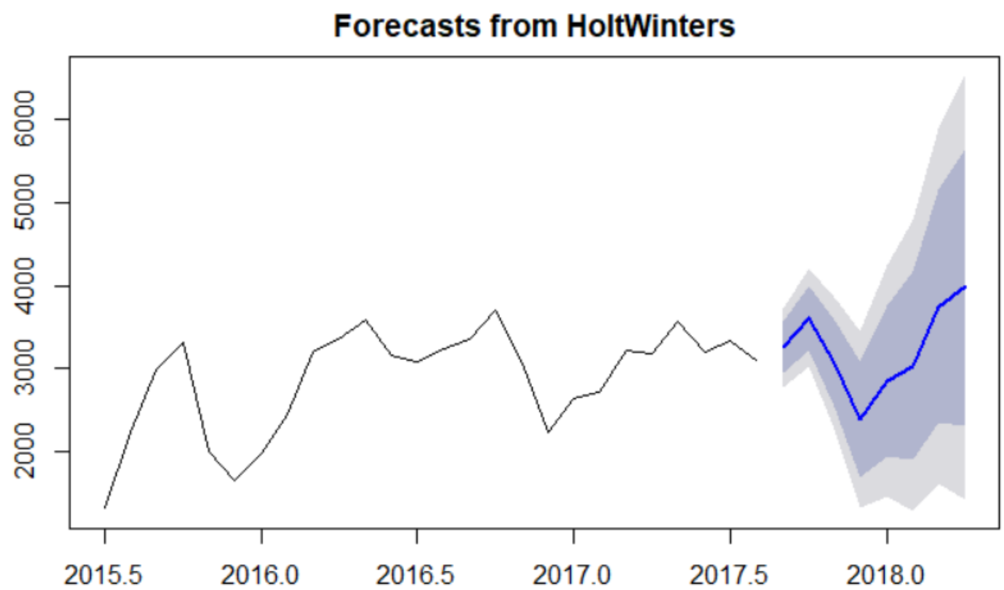
2. MODEL USING HOLTWINTERS:

```
#HoltWinters
```{r}
library("forecast")

Hfit = HoltWinters(hs ,beta=TRUE, gamma=TRUE)
Hfit$fitted
checkresiduals(Hfit)
Hforecast = forecast(Hfit, h=8)
accuracy(Hforecast)

plot(Hforecast)
Hforecast
```
```





Future forecasts of bookings:

	Point Forecast <dbl>	Lo 80 <dbl>	Hi 80 <dbl>	Lo 95 <dbl>	Hi 95 <dbl>
Sep 2017	3248.897	2938.885	3558.910	2774.775	3723.020
Oct 2017	3609.514	3222.398	3996.629	3017.471	4201.556
Nov 2017	3064.943	2544.560	3585.327	2269.086	3860.801
Dec 2017	2387.129	1690.133	3084.125	1321.165	3453.092
Jan 2018	2848.577	1942.073	3755.080	1462.199	4234.954
Feb 2018	3031.974	1889.385	4174.562	1284.535	4779.412
Mar 2018	3747.504	2346.090	5148.918	1604.226	5890.782
Apr 2018	3985.315	2304.844	5665.785	1415.257	6555.372

Validation of the two models:

```
```{r}
accuracy (Aforecast)
accuracy (Hforecast)

```
```

```
                    ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
Training set -72.98426 286.6851 164.4508 -2.558678 5.461286 0.3415385 0.02412717
                   ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
Training set 7.270737 233.2177 218.2107 0.1472528 6.987008 0.4531895 0.3437704
```

	ARIMA	HoltWinters
ME	-72.98426	7.270737
RMSE	286.6851	233.2177
MAE	164.4508	218.2107
MPE	-2.558678	0.1472528
MAPE	5.461286	6.987008
MASE	0.3415385	0.4531895
ACF1	0.02412717	0.3437704

Both models forecasts approximately same bookings in the future with minor differentiations. As auto.arima seasonality was not completely satisfied because of low cycles, Holtwinters can be used as a better model.

CONCLUSIONS

- Booking cancellation model will help to Identify the likelihood of bookings being cancelled and makes it possible for hotel managers to take measures to avoid these potential cancellations, such as offering services, discounts, or other perks.
- The prediction model enables hotel managers to mitigate revenue loss derived from booking cancellations and the risks associated with overbooking (reallocation costs, cash, or service compensations).
- Special request model will contribute to reduce uncertainty in the inventory allocation and pricing decision process by predicting the likelihood of getting a request from customers.
- The repeated guests are only 3%, which points a change in marketing and hospitality.