IST 687: Introduction to Data Science

(Spring 2021)

European Hotels Group

A thorough Data Analysis

Dixitha Kasturi
Jeeshitha Badireddi
Sahaj Khandelwal
Karan Bharadwaj

Overview

The European Hotels Group has always believed that the best way to maximize its revenue is to have a number of hotel properties, in a variety of settings, to accommodate a wide range of travellers. They currently have hotels located in cities and resort locations.

The new owners are not certain this is the correct business strategy for the future, and have hired us to analyze their data. They are interested in how understanding each property is performing individually over time, and to compare how each property is performing compared to the other over time.

We have been provided with 2 comprehensive data sets, one being a hotel dataset and another being a resort dataset and we, as data analysts, use various visualizations and models to generate meaningful insights from the data provided with a view to provide strategic recommendations to the European Hotels Group with a primary view to increase ADR.

The average daily rate (ADR) is a performance measure used in the hospitality industry to measure the strength of revenues generated. It is calculated by summing all of the revenues generated by all the occupied rooms and dividing that sum by the total number of occupied rooms over a given time period. It is a simple average that shows the revenues generated per occupied room.

TABLE OF CONTENTS

TABLE OF FIGURES

[1] Business Questions

[1.1] How does customer turnout vary with seasons?

We found seasonality to be an interesting factor that could enable us to derive some great insights from the datasets.

In order to do this, we created a variable called "Season" in both the datasets that contains the values; Summer, Winter, Spring or Fall based on the arrival date of each customer or supposed arrival date in terms on cancellations or no-shows.

We then proceeded to categorize the number of customers based on this Season variable
in order to study the frequency. In order to be more precise, we also performed an analysis on the exact month which had most observations or the least.

For the Hotel dataset, we noticed that there were a huge number of customers during the Summer season. In other words, customers reserved rooms in the Hotel mostly during the Summer. In terms of when there were the least number of customers, Winter showed a very low number when compared with the other seasons.
However, when looked at the microscopic level of "months", the month of November showed up with the lowest frequency and July had the highest frequency.

For the Resort dataset, a similar trend was observed in Summer having the highest number of customers visiting the resort. The contrast we noticed was that Fall had the lowest number of customers making bookings in the resort.
In the perspective on "months", July had a very number of customers and December had the lowest as expected from the prior seasonality analysis.

Hence, although the aggregate the number of customers in terms of season may give us a result indicating lower interactions in a particular season, the month with the least transactions may not even exist in that indicated season.

After performing the above analysis, we thought it would be interesting to do the same but only for customers who checked out, or in other words, excluding the customers who cancelled their reservation or did not show up.

For the Hotel dataset, what showed up to be different was not during the seasonality analysis but when looked at with the segregation of months, the month of August had the highest number of customers who stayed at the hotel and December had the least number of customer occupancy.

For the Resort dataset, both seasonality and monthly analysis different to an extent.
Winter was the season which had the lowest frequency of customer bookings.
In terms of months, August showed a difference by being the month with the most number of customers.

From this, we concluded that the analysis performed with transactions that had only Check_Out data made more sense as these were bookings that actually went through.
Based on the seasonality and monthly data analysis for check-out observations for both Hotels and Resorts, the company could target those particular months or that particular season to enhance the revenue brought in even further.

[1.2] How can we avoid cancellations?

Throughout the entire process of performing this data analysis, we realized the importance of focusing on cancellations and why these were happening. The best way for European Hotels Group to increase their ADR and average revenue brought in for both the Hotel and the Resort would be reduce the frequency of cancellations.

To offer further recommendations on how to do exactly that, we decided to first see what factors were causing these cancellations. There might have been another approach to this which is to see what factors can increase revenue or ADR but we used the inverse approach as we felt it would call for more awareness on “what not to do”.

The methods we used were Association Rules Mining and Support Vector Machine for prediction and seeing what factors caused cancellations. In order to do this, we used the IsCanceled variable which was present in the provided datasets and mapped other factors to them.

In terms of Association Rules Mining, we tried a huge number of combinations of certain attributes. When the attributes have a certain value, the model helped us check for likeliness of booking cancellation.
For the Hotel dataset, we obtained 116 rules where bookings are more likely to get cancelled. For example, an important insight we inferred was that when there was no deposit made, i.e., when DepositType = No Deposit, the bookings showed a high probability to get cancelled or for the customer to not show up. Another rule showed us that cancellations or no-shows were higher in winter which explains our earlier analysis using seasonality.
For the Resort dataset, we obtained 175 rules where bookings are more likely to get cancelled. Similar to the Hotel dataset, when DepositType = No Deposit, the bookings mostly lead to cancellations or no-shows. We also noted interesting combinations like when ReservedRoomType = A and Country = DEU, cancellations/no-shows were higher.

Coming to Support Vector Machine (SVM) models, we used the attributes PreviousCancellations, LeadTime, Meal, DaysInWaitingList, Season, ReservedRoomType and MarketSegment for both the Hotel and Resort dataset. Using these parameters, we received an accuracy of 76.11% for the Hotel dataset and an accuracy of 76.26% for the Resort dataset. From that, we made the inference that our models could be used to classify as they had a good fit with the datasets used.

[1.3] How does ADR differ with different attributes?

The first attribute we used to compare ADR was CustomerType. We attempted to see which type of customers generated the most or least ADR. For the Hotel dataset, we saw that Transient customers generated slightly higher ADR values when compared to the others. However, as we used box plots, there were a lot of outliers for Transient customers as well. Coming to the Resort dataset, Transient customers once again generated higher ADR values with some outliers.

We then used the Meal attribute which corresponds to the meal type chosen by that particular customer. Looking at the Hotel dataset, we saw that type Full Board generated little to no ADR whereas Half Board generated slighter higher ADR values. In the Resort dataset, Half Board generated slightly higher ADR values followed by Full Board and then Bed & Breakfast. Also, both dataset shad unidentified Meal/ unknown meal type entries which were classified separately.

Further, we used ReservedRoomType to compare the Average Daily Rate on. We felt that the ReservedRoomType was in fact the most relevant attribute as it affected ADR quite significantly. A customer is more likely to generate revenue for the Hotel or Resort based on the wanted room type. Performing analysis on the Hotel dataset, we saw that the room type “G” had customers that generated higher ADR values with B and C being one of the lowest. Coming to the Resort dataset, C and H seemed to be generating similar and higher ADR compared to the other room types with room type A generating the least.

When we tried to put our overall analysis into picture, we had to give weight to the fact that there were outliers, there was the matter of the frequency of customers contributing to this ADR in each category and other factors that could cause misinterpretation. For example, a particular room type may have generated the least ADR because it was the most expensive.

[1.4] Which countries do the majority of our customers come from?

The datasets we performed analysis on contained the attribute Country which indicated the origin countries of all the customers who made reservations.
As a company, recognizing the demographic of the current customers and using that to amplify or target future potential customers would be the right strategy to take. Taking this into perspective, we used the Country variable to check for frequency of customers.

To put that into a visualization for better understanding, we plotted the frequencies of origin countries as a word cloud and on a world map.
In the word cloud, we noticed that Portugal had the highest frequency for both datasets. This made sense taking into perspective that the Hotel and Resort are situated in Portugal. Hence, we disregarded Portugal for customer segmentation and considered the other countries.

For the Hotel dataset, France had the highest occurrence of customers followed by United Kingdom, Germany and Spain. These were all European countries which was another insight that made sense considering the background of the Hotel
For the Resort dataset, United Kingdom had the highest frequency of customers followed by Spain.
We can see that customers from Germany preferred our Hotel more than the Resort. We would like to offer the company a recommendation to further probe into why and how that preference of a Hotel could be amplified.

[1.5] How can we improve revenue taking cancellations into perspectives?

During our attempt to perform revenue analysis on the given data, we ended up finding very interesting factors. We tried to approach this business question with an unusual perspective that could help bring out good insights.

While classifying data, we saw that the datasets had certain inherent characteristics that would make it difficult for appropriate predictions or visualizations to be made. To be more specific, in observations that had bookings that were wither cancelled or was a no-show, the average revenue per stay attribute was still populated. This attribute was one that we developed from ADR and DaysStayed. We were perplexed as to how there could be an ADR or DaysStayed value when there was no customer that actually stayed at the hotel. The only logical solution is that these are supposed values if the reservation had actually followed through. Keeping this in mind, we performed further analysis.

For both datasets, we mapped Avg_Revenue_per_Stay with Seasons to check for patterns or a trend over time. It was natural that in Summer, Avg_Revenue_per_Stay was high for Hotels and Resorts.
However, the difference was observed when we separated a data frame for only transactions that had checked out, i.e., reservations that followed through. Furthering this perspective, we developed visualizations combining the above-mentioned mapping for the whole dataset and only for the data frame that had checked out customers in order to compare them.
We also combined the mapping for Hotels and Resorts specifically for checked out customers to compare their revenue turnaround. Our findings for this were that Hotels had a more even distribution of revenue being generated throughout the year whereas for Resorts, the peak of revenue generated was extremely high in the summer.
An interesting way to look at this is “what could have been” and “what is”. If there were no cancellations and no-shows, the Avg_Revenue_per_Stay over several seasons would be definitely higher than what it actually is.

[1.6] Does the time span between booking date and arrival/cancellation date follow a pattern? Also, how relevant is it?

In our analysis, we thought it would be useful to use the variable LeadTime which is the time duration in days between when the booking was made and when the

customer showed up or didn't show up or cancelled the booking. We mainly focused on the mean number of days in all the upcoming scenarios.

For the Hotel dataset, we found that the average number of days between booking and arrival was about 80 days, or in other words, customers reserved rooms in the Hotel approximately 80 days before. We also wanted to check the time elapsed when the customer ended up cancelling and saw that they took an average of 150 days from when the reservation was made to actually cancelling it. From our previous probing, we saw that customers who made the reservation and did not pay the deposit were most likely to cancel. Using this insight, we also mapped LeadTime for customers who cancelled and did not pay deposit in the first place. Here, we saw that customers took an average of 110 days to cancel after making the booking, provided they did not pay any deposit.

Applying the exact scenarios and analysis to the Resort dataset, we found that the average number of days between booking and arrival was about 78 days, which is very close to that of the Hotel data. Coming to the LeadTime for when customers cancelled their bookings, an average value of 128 days was seen which is quite less from the 150 days taken by customers in the Hotel dataset. Finally, when customers cancelled and no deposit was made in the first place, an average 119 days was taken to cancel the reservation.

We felt that this time span of when a customer would book and when they would most likely cancel can be used to drive marketing strategies for customers who made reservations such that it does not lead to cancellation. Doing this would drive up the ADR and Average Revenue.

[2] Data Acquisition, Data Cleaning and Data Transformation

[2.1] Data Acquisition

```
38  resort_hotel <- read_excel('H1-Resort.xlsx')
39  str(resort_hotel) #40060 observations with 28 attributes
40
41  city_hotel <- read_excel('H2-City.xlsx')
42  str(city_hotel) #79330 observations with 28 attributes
43
```

To load the data into R Studio from the Excel files provided to us, we have used the *read_excel* function to extract data from the excel files and load it into two dataframes – *resort_hotel* and *city_hotel* respectively. resort_hotel data frame contains 40060 observations with 28 attributes and city_hotel data frame contains 79330 observations with 28 attributes.

Descriptive Statistics:

```
> summary(city_hotel)
  IsCanceled       LeadTime      Arrival Date                ReservationStatusDate         ReservationStatus
 Min.   :0.0000   Min.   :  0.0   Min.   :2015-07-01 00:00:00   Min.   :2014-10-17 00:00:00   Length:79330
 1st Qu.:0.0000   1st Qu.: 23.0   1st Qu.:2015-10-22 00:00:00   1st Qu.:2016-02-05 00:00:00   Class :character
 Median :0.0000   Median : 74.0   Median :2016-07-02 00:00:00   Median :2016-08-10 00:00:00   Mode  :character
 Mean   :0.4173   Mean   :109.7   Mean   :2016-07-09 09:20:54   Mean   :2016-07-30 17:43:49
 3rd Qu.:1.0000   3rd Qu.:163.0   3rd Qu.:2017-02-20 00:00:00   3rd Qu.:2017-02-06 00:00:00
 Max.   :1.0000   Max.   :629.0   Max.   :2017-08-31 00:00:00   Max.   :2017-09-07 00:00:00
                                  NA's   :39270
 StaysInWeekendNights StaysInWeekNights     Adults        Children             Babies              Meal
 Min.   : 0.0000      Min.   : 0.000    Min.   :0.000   Length:79330       Min.   : 0.000000   Length:79330
 1st Qu.: 0.0000      1st Qu.: 1.000    1st Qu.:2.000   Class :character   1st Qu.: 0.000000   Class :character
 Median : 1.0000      Median : 2.000    Median :2.000   Mode  :character   Median : 0.000000   Mode  :character
 Mean   : 0.7952      Mean   : 2.183    Mean   :1.851                      Mean   : 0.004941
 3rd Qu.: 2.0000      3rd Qu.: 3.000    3rd Qu.:2.000                      3rd Qu.: 0.000000
 Max.   :16.0000      Max.   :41.000    Max.   :4.000                      Max.   :10.000000

   Country          MarketSegment      DistributionChannel IsRepeatedGuest   PreviousCancellations PreviousBookingsNotCanceled
 Length:79330       Length:79330       Length:79330        Min.   :0.00000   Min.   : 0.00000      Min.   : 0.0000
 Class :character   Class :character   Class :character    1st Qu.:0.00000   1st Qu.: 0.00000      1st Qu.: 0.0000
 Mode  :character   Mode  :character   Mode  :character    Median :0.00000   Median : 0.00000      Median : 0.0000
                                                           Mean   :0.02561   Mean   : 0.07974      Mean   : 0.1324
                                                           3rd Qu.:0.00000   3rd Qu.: 0.00000      3rd Qu.: 0.0000
                                                           Max.   :1.00000   Max.   :21.00000      Max.   :72.0000

 ReservedRoomType   AssignedRoomType   BookingChanges    DepositType          Agent            Company
 Length:79330       Length:79330      Min.   : 0.0000   Length:79330       Length:79330       Length:79330
 Class :character   Class :character  1st Qu.: 0.0000   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character  Median : 0.0000   Mode  :character   Mode  :character   Mode  :character
                                      Mean   : 0.1874
                                      3rd Qu.: 0.0000
                                      Max.   :21.0000

 DaysInWaitingList CustomerType           ADR         RequiredCarParkingSpaces TotalOfSpecialRequests
 Min.   :  0.000   Length:79330       Min.   :   0.0   Min.   :0.00000          Min.   :0.0000
 1st Qu.:  0.000   Class :character   1st Qu.:  79.2   1st Qu.:0.00000          1st Qu.:0.0000
 Median :  0.000   Mode  :character   Median :  99.9   Median :0.00000          Median :0.0000
 Mean   :  3.227                      Mean   : 105.3   Mean   :0.02437          Mean   :0.5469
 3rd Qu.:  0.000                      3rd Qu.: 126.0   3rd Qu.:0.00000          3rd Qu.:1.0000
 Max.   :391.000                      Max.   :5400.0   Max.   :3.00000          Max.   :5.0000
```

```
> summary(resort_hotel)
   IsCanceled        LeadTime       Arrival Date                ReservationStatusDate         ReservationStatus
 Min.   :0.0000   Min.   :  0.00   Min.   :2015-07-01 00:00:00   Min.   :2014-11-18 00:00:00   Length:40060
 1st Qu.:0.0000   1st Qu.: 10.00   1st Qu.:2016-02-14 00:00:00   1st Qu.:2016-01-26 00:00:00   Class :character
 Median :0.0000   Median : 57.00   Median :2016-08-19 00:00:00   Median :2016-07-31 00:00:00   Mode  :character
 Mean   :0.2776   Mean   : 92.68   Mean   :2016-08-15 15:40:38   Mean   :2016-07-28 14:07:14
 3rd Qu.:1.0000   3rd Qu.:155.00   3rd Qu.:2017-03-05 00:00:00   3rd Qu.:2017-02-11 00:00:00
 Max.   :1.0000   Max.   :737.00   Max.   :2017-08-31 00:00:00   Max.   :2017-09-14 00:00:00
 StaysInWeekendNights StaysInWeekNights     Adults          Children          Babies           Meal
 Min.   : 0.00        Min.   : 0.000    Min.   : 0.000   Min.   : 0.0000   Min.   :0.0000   Length:40060
 1st Qu.: 0.00        1st Qu.: 1.000    1st Qu.: 2.000   1st Qu.: 0.0000   1st Qu.:0.0000   Class :character
 Median : 1.00        Median : 3.000    Median : 2.000   Median : 0.0000   Median :0.0000   Mode  :character
 Mean   : 1.19        Mean   : 3.129    Mean   : 1.867   Mean   : 0.1287   Mean   :0.0139
 3rd Qu.: 2.00        3rd Qu.: 5.000    3rd Qu.: 2.000   3rd Qu.: 0.0000   3rd Qu.:0.0000
 Max.   :19.00        Max.   :50.000    Max.   :55.000   Max.   :10.0000   Max.   :2.0000
   Country          MarketSegment      DistributionChannel IsRepeatedGuest   PreviousCancellations PreviousBookingsNotCanceled
 Length:40060       Length:40060       Length:40060        Min.   :0.00000   Min.   : 0.0000       Min.   : 0.0000
 Class :character   Class :character   Class :character    1st Qu.:0.00000   1st Qu.: 0.0000       1st Qu.: 0.0000
 Mode  :character   Mode  :character   Mode  :character    Median :0.00000   Median : 0.0000       Median : 0.0000
                                                           Mean   :0.04438   Mean   : 0.1017       Mean   : 0.1465
                                                           3rd Qu.:0.00000   3rd Qu.: 0.0000       3rd Qu.: 0.0000
                                                           Max.   :1.00000   Max.   :26.0000       Max.   :30.0000
 ReservedRoomType   AssignedRoomType   BookingChanges   DepositType           Agent             Company
 Length:40060       Length:40060       Min.   : 0.000   Length:40060       Length:40060       Length:40060
 Class :character   Class :character   1st Qu.: 0.000   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Median : 0.000   Mode  :character   Mode  :character   Mode  :character
                                       Mean   : 0.288
                                       3rd Qu.: 0.000
                                       Max.   :17.000
 DaysInWaitingList  CustomerType           ADR         RequiredCarParkingSpaces TotalOfSpecialRequests
 Min.   :  0.0000   Length:40060       Min.   : -6.38   Min.   :0.0000           Min.   :0.0000
 1st Qu.:  0.0000   Class :character   1st Qu.: 50.00   1st Qu.:0.0000           1st Qu.:0.0000
 Median :  0.0000   Mode  :character   Median : 75.00   Median :0.0000           Median :0.0000
 Mean   :  0.5278                      Mean   : 94.95   Mean   :0.1381           Mean   :0.6198
 3rd Qu.:  0.0000                      3rd Qu.:125.00   3rd Qu.:0.0000           3rd Qu.:1.0000
 Max.   :185.0000                      Max.   :508.00   Max.   :8.0000           Max.   :5.0000
>
```

Figure 1 Descriptive StatisticsFigure

[2.2] Data cleaning

```
#Checking for NAs
cbind(
  lapply(
    lapply(city_hotel, is.na)
    , sum)
) #39270 NA values in the Arrival Date column

cbind(
  lapply(
    lapply(resort_hotel, is.na)
    , sum)
) #0 NA values
#This shows us that all the NAs are in the Arrival Date column
View(city_hotel)

# Verifying if any other column has na values:
x <- city_hotel[,-3] # generating a table without the arrivaldate column
na_x<- sum(is.na(x))
na_x #0, implying no other na values are present.
```

We initiate data cleaning by conducting a check on the presence and spread of NA values in the data sets. We realize that the resort hotel data set does not contain any NA values whereas the city hotel data set contains a total of 39270 NA values.

The natural next logical step would be to check the spread of NA values across the data set, i.e, identify the columns which hold NA values. We observe that all the NA

values in the *city_hotel* data frame are in the *Arrival Date* column. After manually looking at the data, the observations with NAs in the Arrival Date column have other relevant information under other columns.

Just to verify no other column has NA values, we generate a table without the *Arrival Date* column and then check for NAs, and conclude with the confirmation of the absence of NAs across other columns apart from the *Arrival Date* column.

```
#------ Data Cleaning ------

city_hotel$Agent <- as.factor(city_hotel$Agent)
city_hotel$AssignedRoomType <- as.factor(city_hotel$AssignedRoomType)
city_hotel$Company <- as.factor(city_hotel$Company)
city_hotel$Country <- as.factor(city_hotel$Country)
city_hotel$CustomerType <- as.factor(city_hotel$CustomerType)
city_hotel$DepositType <- as.factor(city_hotel$DepositType)
city_hotel$DistributionChannel <- as.factor(city_hotel$DistributionChannel)
city_hotel$IsRepeatedGuest <- as.factor(city_hotel$IsRepeatedGuest)
city_hotel$MarketSegment <- as.factor(city_hotel$MarketSegment)
city_hotel$Meal <- as.factor(city_hotel$Meal)
city_hotel$ReservationStatus <- as.factor(city_hotel$ReservationStatus)
city_hotel$ReservedRoomType <- as.factor(city_hotel$ReservedRoomType)
city_hotel$Children <- as.numeric(city_hotel$Children)
city_hotel <- city_hotel %>% drop_na(Children) #Dropping 4 rows which has NA values in the attribute Children
str(city_hotel) #Verifying the conversions
```

```
#Converting variables into appropriate datatypes
resort_hotel$Agent <- as.factor(resort_hotel$Agent)
resort_hotel$AssignedRoomType <- as.factor(resort_hotel$AssignedRoomType)
resort_hotel$Company <- as.factor(resort_hotel$Company)
resort_hotel$Country <- as.factor(resort_hotel$Country)
resort_hotel$CustomerType <- as.factor(resort_hotel$CustomerType)
resort_hotel$DepositType <- as.factor(resort_hotel$DepositType)
resort_hotel$DistributionChannel <- as.factor(resort_hotel$DistributionChannel)
resort_hotel$IsRepeatedGuest <- as.factor(resort_hotel$IsRepeatedGuest)
resort_hotel$MarketSegment <- as.factor(resort_hotel$MarketSegment)
resort_hotel$Meal <- as.factor(resort_hotel$Meal)
resort_hotel$ReservationStatus <- as.factor(resort_hotel$ReservationStatus)
resort_hotel$ReservedRoomType <- as.factor(resort_hotel$ReservedRoomType)
str(resort_hotel) #Verifying the conversions
```

Subsequently, we embark on the process of converting various variable data types as a categorical/factor variable. This would help us in later analysis and visualizations.

The *Children* column contains 4 NA values upon conversion to numeric data type and we decide to drop the corresponding 4 rows from the data frame.

```
100
101 #Creating Derived Variables
102
103 #Creating a variable to calculate the average revenue per stay
104 resort_hotel$DaysStayed <- resort_hotel$StaysInWeekendNights + resort_hotel$StaysInWeekNights
105 city_hotel$DaysStayed <- city_hotel$StaysInWeekendNights + city_hotel$StaysInWeekNights
106
107 resort_hotel$Avg_Revenue_per_Stay <- resort_hotel$DaysStayed * resort_hotel$ADR
108 city_hotel$Avg_Revenue_per_Stay <- city_hotel$DaysStayed * city_hotel$ADR
109
110 city_hotel$TotalPeople <-  city_hotel$Adults + city_hotel$Children + city_hotel$Babies
111 city_hotel$TotalChildren <- city_hotel$Babies + city_hotel$Children
112 resort_hotel$TotalPeople <-  resort_hotel$Adults + resort_hotel$Children + resort_hotel$Babies
113 resort_hotel$TotalChildren <- resort_hotel$Babies + resort_hotel$Children
114
```

[2.3] Data Transformation

Next, we create a few derived variables to make more sense of the data available to us.

First, we create a variable *Avg_Revenue_per_Stay to* effectively calculate the average revenue per stay. It's obtained by multiplying the ADR with the days stayed (which includes weekends and weekdays).

```
#Creating a newArrivalDate column
#In terms of city_hotel which has a lot of NA values, newArrivalDate is calculated based on reservation status updation date and number of days
city_hotel$newArrival <- as.Date(city_hotel$ReservationStatusDate - city_hotel$DaysStayed)
resort_hotel$newArrival <- as.Date(resort_hotel$ReservationStatusDate - resort_hotel$DaysStayed)
```

Since the *Arrival Date* column has a lot of NA values for city_hotel data frame, we have created a new column *newArrival* which is obtained from the *ReservationStatusDate* and *DaysStayed* columns. We have done the same for *resort_hotel* data frame as well, just to standardize procedures.

```
#Creating visitor type variable based on total number of people
city_hotel<- mutate(city_hotel, VisitorType = case_when( TotalPeople  == 1~ "Single",
                                                         TotalPeople == 2 & Adults == 2 ~ "Couple",
                                                         TotalPeople > 2 ~ "Family",
                                                         TotalPeople >= 2 & Adults >= 1 & TotalChildren >= 1 ~ "Family" ,
                                                         TotalPeople == 0 ~ "Family"
))

resort_hotel<- mutate(resort_hotel, VisitorType = case_when( TotalPeople  == 1~ "Single",
                                                         TotalPeople == 2 & Adults == 2 ~ "Couple",
                                                         TotalPeople > 2 ~ "Family",
                                                         TotalPeople >= 2 & Adults >= 1 & TotalChildren >= 1 ~ "Family" ,
                                                         TotalPeople == 0 ~ "Family"
))
```

Next, we create a Visitor Type variable based on the number of people involved in that stay. This would help us in effective analysis per nature of group in question.

```
resort_hotel <- mutate(resort_hotel, Season = case_when( format(resort_hotel$`Arrival Date`, "%m") == '03' ~ "Spring",
                                                         format(resort_hotel$`Arrival Date`, "%m") == '04' ~ "Spring",
                                                         format(resort_hotel$`Arrival Date`, "%m") == '05' ~ "Spring",
                                                         format(resort_hotel$`Arrival Date`, "%m") == '06' ~ "Summer",
                                                         format(resort_hotel$`Arrival Date`, "%m") == '07' ~ "Summer",
                                                         format(resort_hotel$`Arrival Date`, "%m") == '08' ~ "Summer",
                                                         format(resort_hotel$`Arrival Date`, "%m") == '09' ~ "Fall",
                                                         format(resort_hotel$`Arrival Date`, "%m") == '10' ~ "Fall",
                                                         format(resort_hotel$`Arrival Date`, "%m") == '11' ~ "Fall",
                                                         format(resort_hotel$`Arrival Date`, "%m") == '12' ~ "Winter",
                                                         format(resort_hotel$`Arrival Date`, "%m") == '01' ~ "Winter",
                                                         format(resort_hotel$`Arrival Date`, "%m") == '02' ~ "Winter"
))
city_hotel <- mutate(city_hotel, Season = case_when( format(city_hotel$newArrival, "%m") == '03' ~ "Spring",
                                                     format(city_hotel$newArrival, "%m") == '04' ~ "Spring",
                                                     format(city_hotel$newArrival, "%m") == '05' ~ "Spring",
                                                     format(city_hotel$newArrival, "%m") == '06' ~ "Summer",
                                                     format(city_hotel$newArrival, "%m") == '07' ~ "Summer",
                                                     format(city_hotel$newArrival, "%m") == '08' ~ "Summer",
                                                     format(city_hotel$newArrival, "%m") == '09' ~ "Fall",
                                                     format(city_hotel$newArrival, "%m") == '10' ~ "Fall",
                                                     format(city_hotel$newArrival, "%m") == '11' ~ "Fall",
                                                     format(city_hotel$newArrival, "%m") == '12' ~ "Winter",
                                                     format(city_hotel$newArrival, "%m") == '01' ~ "Winter",
                                                     format(city_hotel$newArrival, "%m") == '02' ~ "Winter"
))

```

Next, we create a seasonality variable based on when people visit the properties, be it a resort or a hotel. We have considered the standard convention of categorizing the months of March, April and May as Spring, June, July and August as Summers, September, October and November as Fall and December, January and February as Winter.

```
city_hotel<- mutate(city_hotel, DepositFactor = case_when( DepositType  == 'No Deposit'~ 1,
                                                           DepositType == 'Non Refund' ~ 2,
                                                           DepositType == 'Refundable' ~ 3
))
resort_hotel<- mutate(resort_hotel, DepositFactor = case_when( DepositType  == 'No Deposit'~ 1,
                                                               DepositType == 'Non Refund' ~ 2,
                                                               DepositType == 'Refundable' ~ 3
))

```

Next, we are creating a variable *DepositFactor* which gets assigned a numeric equivalent to *DepositType.* This would make it easier to handle the data.

```
city_hotel<- mutate(city_hotel, ReservationFactor = case_when( ReservationStatus == 'Canceled'~ 1,
                                                               ReservationStatus == 'Check-Out' ~ 2,
                                                               ReservationStatus == 'No-Show' ~ 3
))
resort_hotel<- mutate(resort_hotel, ReservationFactor = case_when( ReservationStatus == 'Canceled'~ 1,
                                                                   ReservationStatus == 'Check-Out' ~ 2,
                                                                   ReservationStatus == 'No-Show' ~ 3
))

```

Next, we create a variable *ReservationFactor* which gets assigned a numeric level based on *ReservationStatus.*

```
#Checking for NAs again
cbind(
  lapply(
    lapply(city_hotel, is.na)
    , sum)
) #39270 NA values in the Arrival Date column as before and 205 NA values in the VisitorType column

cbind(
  lapply(
    lapply(resort_hotel, is.na)
    , sum)
) #205 NA values in the VistorType variable

#Dropping the rows with NA values as they are insignificant
city_hotel <- city_hotel %>% drop_na(VisitorType)
resort_hotel <- resort_hotel %>% drop_na(VisitorType)
```

Next, we again inspect for NA values and additionally find 205 NA values for both the hotel and resort data sets in the *VisitorType* variable and we proceed to drop them.

```
#Removing ArrivalDate column due to the NA Values and also because we created newArrival
city_hotel <- city_hotel[,-3]
resort_hotel <- resort_hotel[,-3]

#Final Check for NAs
cbind(
  lapply(
    lapply(city_hotel, is.na)
    , sum)
)

cbind(
  lapply(
    lapply(resort_hotel, is.na)
    , sum)
)
#Both the datasets have no remaining NA Values
```

We proceeded to remove the ArrivalDate column for both datasets considering that we created a newArrival column that was not erroneous. To confirm that the final datasets did not have NA values, we performed a final check which showed that there were no remaining NA values.

```
207 city_hotel$Country <- countrycode(city_hotel$Country, origin='iso3c',destination='country.name')
208 city_hotel$Country[city_hotel$Country == 'CN'] <- 'Pitcairn'
209 city_hotel$Country[city_hotel$Country == 'TMP'] <- 'East Timor'
210 city_hotel <- city_hotel %>% drop_na(Country)
211 View(data.frame(city_hotel$Country))
212
213
214 resort_hotel$Country <- countrycode(resort_hotel$Country, origin='iso3c',destination='country.name')
215 resort_hotel$Country[resort_hotel$Country == 'CN'] <- 'Pitcairn'
216 resort_hotel <- resort_hotel %>% drop_na(Country)
217 View(data.frame(resort_hotel$Country))
218
```

Subsequently, we embark on changing country codes into country names.

```
221 city_country_table <- table(city_hotel$Country)
222 View(data.frame(city_country_table))
223 city_country_freq <- data.frame(city_country_table)
224 city_country_freq
225 names(city_country_freq)[names(city_country_freq)=="Var1"] <- "Country"
226 names(city_country_freq)[names(city_country_freq)=="Freq"] <- "Count"
227 View(city_country_freq)
228 View(city_hotel)
229
230 resort_country_table <- table(resort_hotel$Country)
231 View(data.frame(resort_country_table))
232 resort_country_freq <- data.frame(resort_country_table)
233 resort_country_freq
234 names(resort_country_freq)[names(resort_country_freq)=="Var1"] <- "Country"
235 names(resort_country_freq)[names(resort_country_freq)=="Freq"] <- "Count"
236 View(resort_country_freq)
237 View(resort_hotel)
238
```

We attempt to summarize frequencies of countries as is applicable to resort data frame as well as hotel data frame.

[3] Data Visualization

[3.1] Word cloud

```
242 wordcloud(words = city_country_freq$Country, freq = city_country_freq$Count, min.freq = 1,
243           max.words=200, random.order=FALSE, rot.per=0.35,
244           colors=brewer.pal(8, "Dark2")) #Portugal is the most occuring country of residence
245 wordcloud(words = resort_country_freq$Country, freq = resort_country_freq$Count, min.freq = 1,
246           max.words=200, random.order=FALSE, rot.per=0.35,
247           colors=brewer.pal(8, "Dark2")) #Portugal is the most occuring country of residence followed by the United Kingdom
248
```

Word cloud for city hotel data set:

Figure 2 Word Cloud for Hotel dataset

Word cloud for resort hotel data set:

Figure 3 Word Cloud for Resort Dataset

We generate a word cloud to visualize the frequency of occurrence of various nationalities. We found that for the hotel data set, Portugal is the most frequently occurring country of residence by a substantial margin. For the resort data set, we again found Portugal to be most frequently occurring country of residence followed by United Kingdom.

[3.2] Map plotting

```
world_map_city <- map_data("world")
world_map_city <- merge(world_map_city,city_country_freq, by.x ="region", by.y="Country")
str(world_map_city)
world_map_city <- world_map_city[order(world_map_city$group,world_map_city$order), ]
ggplot(world_map_city, aes(x=long, y=lat, group = group)) + geom_polygon(aes(fill = Count),color ="black")
+ coord_map(projection = "mercator") +scale_fill_gradient(high = "red", low ="blue")
+ scale_fill_continuous(low = "#9A7787", high = "RED",breaks =c(0,5000,10000,15000,20000,25000,30500))

world_map_resort <- map_data("world")
world_map_resort <- merge(world_map_resort,resort_country_freq, by.x ="region", by.y="Country")
str(world_map_resort)
world_map_resort <- world_map_resort[order(world_map_resort$group,world_map_resort$order), ]
ggplot(world_map_resort, aes(x=long, y=lat, group = group)) + geom_polygon(aes(fill = Count),color ="black")
+ coord_map(projection = "mercator") +scale_fill_gradient(high = "red", low ="blue")
+ scale_fill_continuous(low = "#9A7787", high = "RED",breaks =c(0,5000,10000,15000,20000,25000,30500))

```

Generating a map to study the frequency of customer nationalities – Hotel data set:

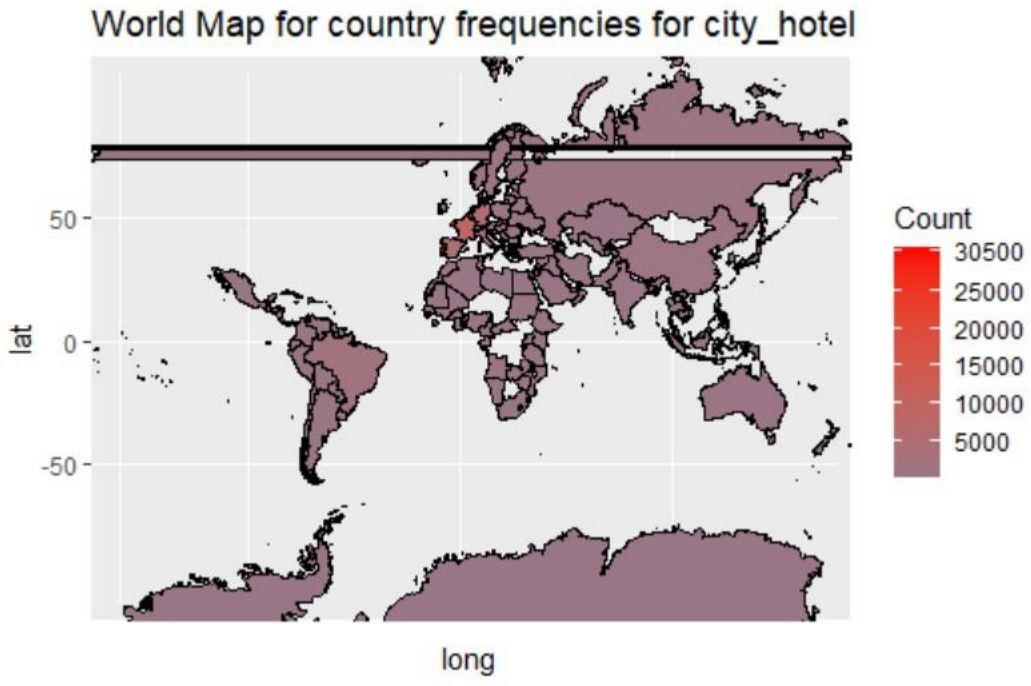

Figure 4 World Map for Hotel data

The results are consistent with the findings in the word cloud.

Generating a map to study the frequency of occurrence of nationalities – Resort data set:

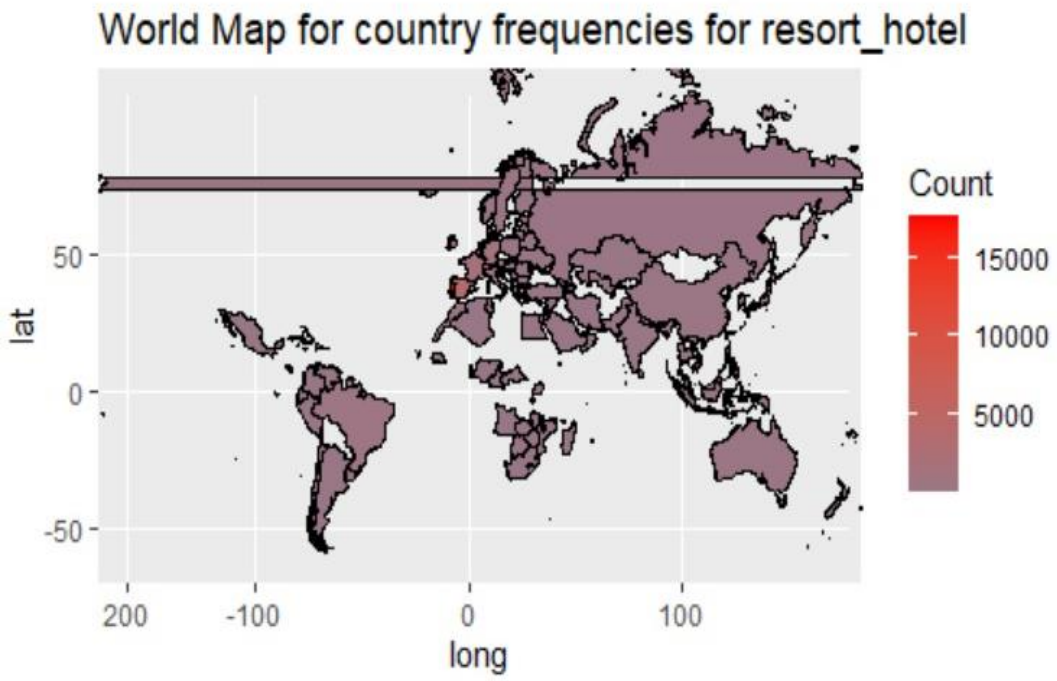

Figure 5 World Map for Resort Data

The results are again consistent with the results found in the word cloud.

[3.3] Line Plots

We are generating a table of months to get their total counts and converting it into a data frame for further analysis. This is to assist with studying patterns of occupancy relative to the month of the year, and helps with seasonality analysis.

```
#Generating a table of months to get their total counts and converting it into a dataframe for further analysis
tab1 <- format(city_hotel$newArrival, "%m") # Getting month of the arrival date
tab1 <- data.frame(table(tab1))
names(tab1)[names(tab1)=="tab1"] <- "Month" #Renaming the Column name in the table
View(tab1)

tab2 <- format(resort_hotel$newArrival, "%m") # Getting month of the arrival date
tab2 <- data.frame(table(tab2))
names(tab2)[names(tab2)=="tab2"] <- "Month" #Renaming the Column name in the table
View(tab2)
```

Plot to study customers by month – Hotel data set:

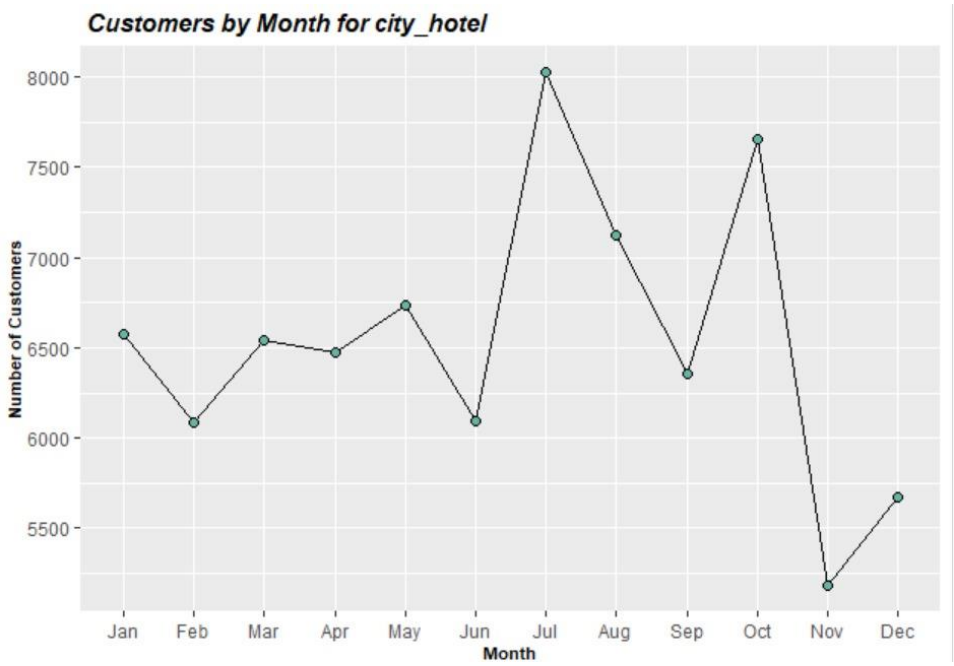

Figure 6 Line plot for customers by month of city data

The findings reveal that the number of customers booking the hotel is highest in the month of July, and closely followed by the month of October. November and December witness the least number of customers.

Plot to study customers by month – resort data set:

Figure 7 Customers by month for Resort data

The findings reveal that the number of customers booking the resort is highest in the month of July, and closely followed by the month of August. November and December witness the least number of customers.

Next, we are generating a table of seasons to get their total counts and converting it into a dataframe for further analysis. This with to assist with studying patterns of occupancy relative to the season.

```
294 tab_season_city <- data.frame(table(city_hotel$Season)) #Generating a table of seasons with total number of customers
295 names(tab_season_city)[names(tab_season_city)=="Var1"] <- "Season"  # renaming the tabular column
296 season_plot_city <- ggplot(data= tab_season_city,aes(x=Season,y=Freq, color = Season, group =1)) +  geom_line(color = "Grey")
297 + geom_point(shape = 21, color="black", aes(fill=Season), size=3 ) +theme(legend.position = "none")
298 season_plot_city <- season_plot_city + ggtitle("Trend by Season") + ylab(" Number of Customers")
299 season_plot_city <- season_plot_city + theme(
300   plot.title = element_text(color="Black", size=12, face="bold.italic"),
301   axis.title.x = element_text( size=8, face="bold"),
302   axis.title.y = element_text( size=8, face="bold"))
303 season_plot_city
304
305 tab_season_resort <- data.frame(table(resort_hotel$Season)) #Generating a table of seasons with total number of customers
306 names(tab_season_resort)[names(tab_season_resort)=="Var1"] <- "Season"  # renaming the tabular column
307 season_plot_resort <- ggplot(data= tab_season_resort,aes(x=Season,y=Freq, color = Season, group =1))
308 +  geom_line(color = "Grey")+ geom_point(shape = 21, color="black", aes(fill=Season), size=3 ) +theme(legend.position = "none")
309 season_plot_resort <- season_plot_resort + ggtitle("Trend by Season") + ylab(" Number of Customers")
310 season_plot_resort <- season_plot_resort + theme(
311   plot.title = element_text(color="Black", size=12, face="bold.italic"),
312   axis.title.x = element_text( size=8, face="bold"),
313   axis.title.y = element_text( size=8, face="bold"))
314 season_plot_resort
315
```

Plot to study customers by season – Hotel data set:

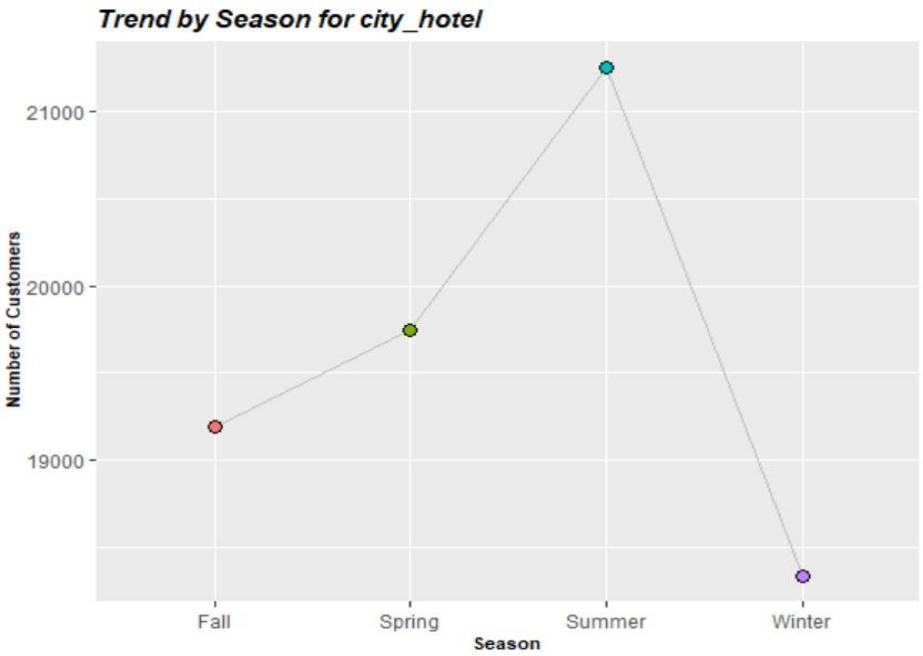

Figure 8 Customers by Seasons for City data

The plot reveals that the number of customers booking the hotel is highest in the Summer season with Spring coming second.

Plot to study customers by season – Resort data set:

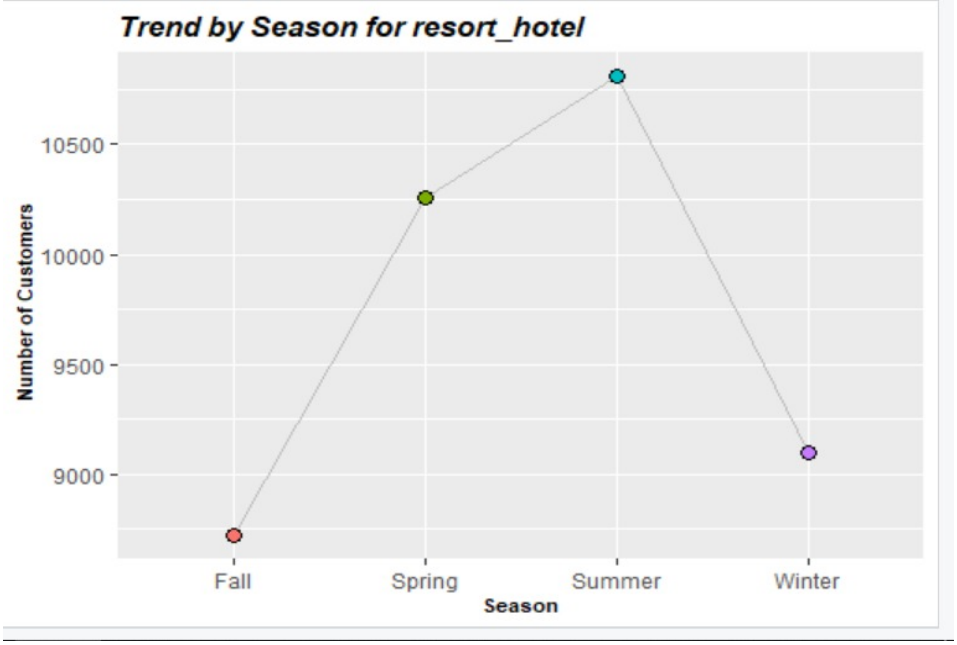

Figure 9 Customers by season for Resort data

The plot reveals that the number of customers booking the resort is highest in the Summer season with Spring coming second.
Next, we proceed to conduct monthly analysis based on the checkout dates for both the hotel and the resort data sets. This essentially eliminates no-shows and cancelled customers from the analysis.

```
#Checking seasonality for only checkout dates
sample_hotel_city <- data.frame(city_hotel$newArrival,city_hotel$ReservationStatus,city_hotel$Season)
View(sample_hotel_city)
sample_hotel_city<- sample_hotel_city[(sample_hotel_city$city_hotel.ReservationStatus != 'Canceled'
                                       & sample_hotel_city$city_hotel.ReservationStatus != 'No-Show'), ]
View(sample_hotel_city) # All check-outs
sample_hotel_resort <- data.frame(resort_hotel$newArrival,resort_hotel$ReservationStatus,resort_hotel$Season)
View(sample_hotel_resort)
sample_hotel_resort<- sample_hotel_resort[(sample_hotel_resort$resort_hotel.ReservationStatus != 'Canceled'
                                           & sample_hotel_resort$resort_hotel.ReservationStatus != 'No-Show'), ]
View(sample_hotel_resort) # All check-outs
```

```
#Extracting arrival months only for checked-out customers
tab8_city <- format(sample_hotel_city$city_hotel.newArrival, "%m") # Getting month of the arrival date
tab8_city <- data.frame(table(tab8_city)) # generating a table of months to get their total counts and converting it into a dataframe for furt
names(tab8_city)[names(tab8_city)== "tab8_city"] <- "Month" #Renaming the Column name in the table
View(tab8_city)
tab8_resort <- format(sample_hotel_resort$resort_hotel.newArrival, "%m") # Getting month of the arrival date
tab8_resort <- data.frame(table(tab8_resort)) # generating a table of months to get their total counts and converting it into a dataframe for
names(tab8_resort)[names(tab8_resort)== "tab8_resort"] <- "Month" #Renaming the Column name in the table
View(tab8_resort)
```

```
#Creating a plot for the month and number of customers that checked out.
p_city <- ggplot(data = tab8_city, aes( x= Month, y= Freq, group = 1)) + ggtitle("Customers who checked out in city_hotel")
+geom_line(data = tab8_city, aes( x= Month, y= Freq, group =1))
+geom_point(shape = 21, size = 2,color = "black", fill = "#BCD979",data = tab8_city, aes( x= Month, y= Freq, group =1))
+scale_x_discrete(labels=month.abb) + scale_y_continuous(breaks = seq(1000,6000, by = 500))
p_city
p_resort <- ggplot(data = tab8_resort, aes( x= Month, y= Freq, group = 1)) + ggtitle("Customers who checked out in resort_hotel")
+geom_line(data = tab8_resort, aes( x= Month, y= Freq, group =1))
+geom_point(shape = 21, size = 2,color = "black", fill = "#BCD979",data = tab8_resort, aes( x= Month, y= Freq, group =1))
+  scale_x_discrete(labels=month.abb) + scale_y_continuous(breaks = seq(1000,6000, by = 500))
p_resort
```

Plot to study customers by month based on check out date – Hotel data:

Figure 10 Customers by month who checked-out for Hotel data

The above plot reveals the highest number of checked out customers in August, with July following next. January and December have the lowest number of checked out customers.

Figure 11 Customers by month who checked out for Resort data

The above plot reveals the highest number of checked out customers in August, with July following next. December and June have the lowest number of checked out customers.

Next, we proceed to conduct seasonal analysis based on the checkout dates for both the hotel and the resort data sets. This essentially eliminates no-shows and cancelled customers from the analysis and gives us a seasonality perspective.

```
#Plotting according to seasons for check-out only
checkout_season_city <- data.frame(table(sample_hotel_city$city_hotel.Season)) #Generating a table of seasons with total number of customers
names(checkout_season_city)[names(checkout_season_city)=="Var1"] <- "Season"  # renaming the tabular column
checkout_season_plot_city <- ggplot(data= checkout_season_city,aes(x=Season,y=Freq, color = Season, group =1))
+  geom_line(color = "Grey")+ geom_point(shape = 21, color="black", aes(fill=Season), size=3 ) +theme(legend.position = "none")
checkout_season_plot_city <- checkout_season_plot_city + ggtitle("Trend by Season for Check-out only in city_hotel")
+ ylab(" Number of Customers")
checkout_season_plot_city <- checkout_season_plot_city + theme(
  plot.title = element_text(color="Black", size=12, face="bold.italic"),
  axis.title.x = element_text( size=8, face="bold"),
  axis.title.y = element_text( size=8, face="bold"))
checkout_season_plot_city

checkout_season_resort <- data.frame(table(sample_hotel_resort$resort_hotel.Season)) #Generating a table of seasons with total number of custc
names(checkout_season_resort)[names(checkout_season_resort)=="Var1"] <- "Season"  # renaming the tabular column
checkout_season_plot_resort <- ggplot(data= checkout_season_resort,aes(x=Season,y=Freq, color = Season, group =1))
+  geom_line(color = "Grey")+ geom_point(shape = 21, color="black", aes(fill=Season), size=3 ) +theme(legend.position = "none")
checkout_season_plot_resort <- checkout_season_plot_resort + ggtitle("Trend by Season for Check-out only in city_hotel")
+ ylab(" Number of Customers")
checkout_season_plot_resort <- checkout_season_plot_resort + theme(
  plot.title = element_text(color="Black", size=12, face="bold.italic"),
  axis.title.x = element_text( size=8, face="bold"),
  axis.title.y = element_text( size=8, face="bold"))
checkout_season_plot_resort
```

Plot to study customers by season based on check out date – Hotel data:

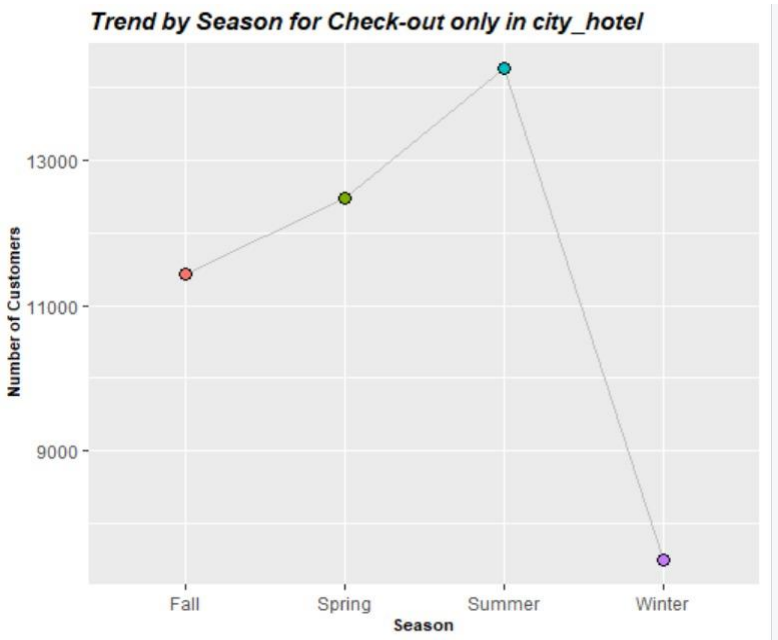

Figure 12 Customers by Season for City data who checked out

The plot reveals that the number of customers booking the hotel is highest in the Summer season with Spring coming second.

Plot to study customers by season based on check out date – Resort data:

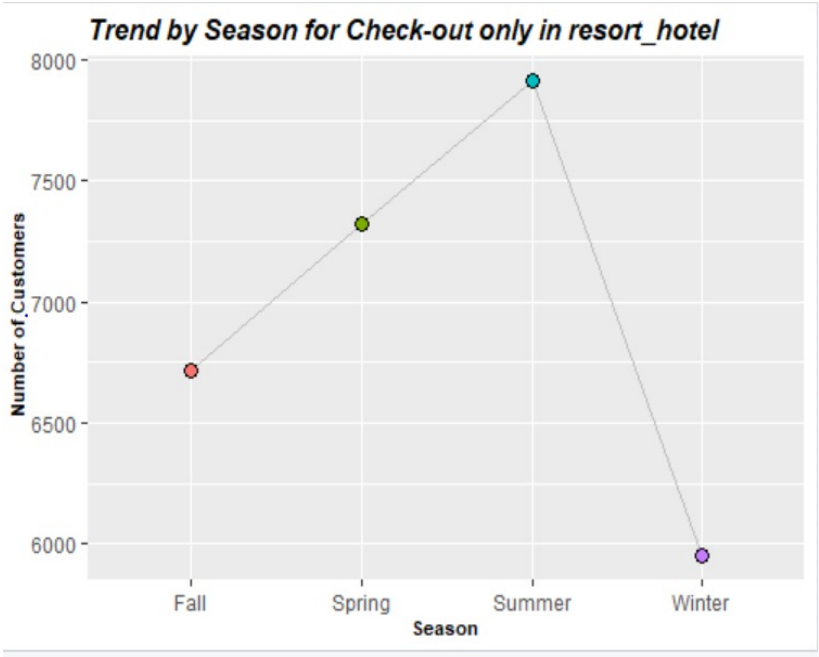

Figure 13 Customers by Season for Resort data who checked out

The plot reveals that the number of customers booking the resort is highest in the Summer season with Spring coming second.

[3.4] Bar Plots

Next, we proceed to produce a series of bar plots to estimate lead times.
We first generate these plots for the hotel data set.

```
#LeadTime plots
#For city_hotel
#LeadTime for Cancelled and No-Show data
Lead_CheckIn_city <- city_hotel[(city_hotel$ReservationStatus!= 'Canceled' & city_hotel$ReservationStatus!= 'No-Show'), ]
Lead_CheckIn_city_plot <- ggplot(data = Lead_CheckIn_city,aes(x =LeadTime)) + ggtitle("Number of Days Between Booking Date and Arrival f
  scale_y_continuous(limits = c(0,1000),breaks = seq(0,1000, by=100)) +
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"))
Lead_CheckIn_city_plot

#LeadTime for only Check-Out data
Lead_CheckOut_city <- city_hotel[(city_hotel$ReservationStatus!= 'Check-Out'), ]
Lead_CheckOut_city_plot <- ggplot(data = Lead_CheckOut_city,aes(x =LeadTime)) + ggtitle("Number of Days Between Booking Date and cancell
  scale_y_continuous(limits = c(0,1000),breaks = seq(0,1000, by=100)) +
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"))
Lead_CheckOut_city_plot

#LeadTime for Check-Out and No Deposit
Lead_CheckIn_Deposit_city <- city_hotel[(city_hotel$ReservationStatus!= 'Check-Out' & city_hotel$DepositType == 'No Deposit'), ]
Lead_CheckIn_Deposit_city_plot <- ggplot(data = Lead_CheckIn_Deposit_city,aes(x =LeadTime)) + ggtitle("Number of Days Between Booking Da
  scale_y_continuous(limits = c(0,500),breaks = seq(0,500, by=100)) +
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"))
Lead_CheckIn_Deposit_city_plot
```

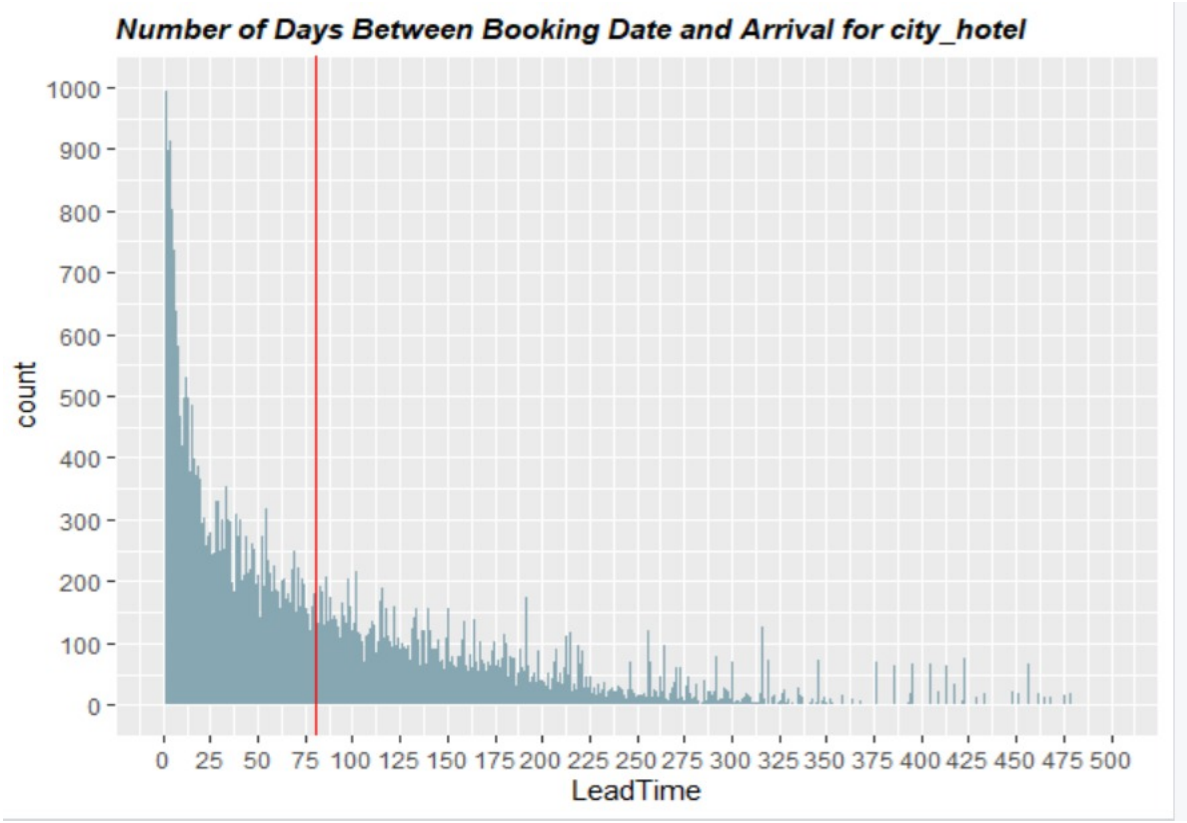

Figure 14 Lead Time plot for days between Booking and Arrival for City Data

In the above plot, we find the average lead time between the booking date and arrival date for the hotel data set to be approximately 80 days. It is also interesting to note that a large number of customers book it immediately before arriving or a day before.

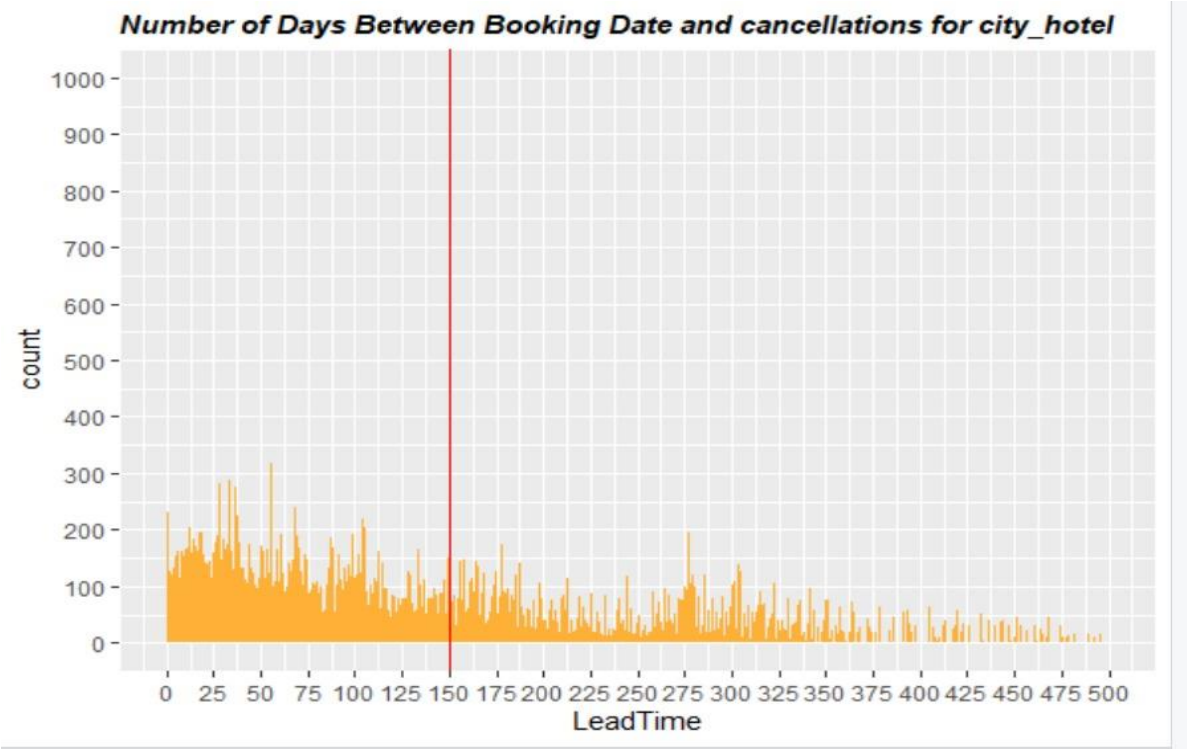

Figure 15 LeadTime plot for days between booking and Cancellations for City data

In the above plot, we find the average lead time between the booking date and the cancellation date to be about 150 days.

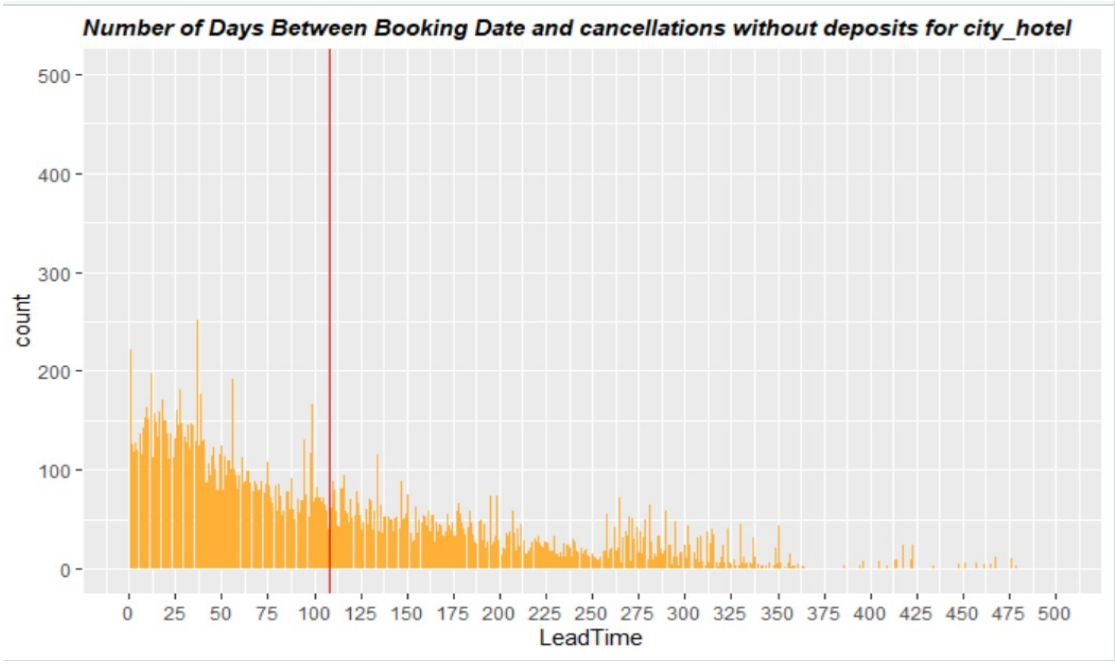

Figure 16 LeadTime plot for days between booking and cancellation with no deposit for City data

In the above plot, we find the average lead time between the booking date and cancellation date to be about 110 days. This plot is for the customers who haven't made deposits.
We now generate these plots for the resort data set.

```
#For resort_hotel
#LeadTime for Cancelled and No-Show data
Lead_CheckIn_resort <- resort_hotel[(resort_hotel$ReservationStatus!= 'Canceled' & resort_hotel$ReservationStatus!= 'No-Show'), ]
Lead_CheckIn_resort_plot <- ggplot(data = Lead_CheckIn_resort,aes(x =LeadTime)) + ggtitle("Number of Days Between Booking Date and Arriva
  scale_y_continuous(limits = c(0,1000),breaks = seq(0,1000, by=100)) +
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"))
Lead_CheckIn_resort_plot

#LeadTime for only Check-Out data
Lead_CheckOut_resort <- resort_hotel[(resort_hotel$ReservationStatus!= 'Check-Out'), ]
Lead_CheckOut_resort_plot <- ggplot(data = Lead_CheckOut_resort,aes(x =LeadTime)) + ggtitle("Number of Days Between Booking Date and can
  scale_y_continuous(limits = c(0,1000),breaks = seq(0,1000, by=100)) +
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"))
Lead_CheckOut_resort_plot

#LeadTime for Check-Out and No Deposit
Lead_CheckIn_Deposit_resort <- resort_hotel[(resort_hotel$ReservationStatus!= 'Check-Out' & city_hotel$DepositType == 'No Deposit'), ]
Lead_CheckIn_Deposit_resort_plot <- ggplot(data = Lead_CheckIn_Deposit_resort,aes(x =LeadTime)) + ggtitle("Number of Days Between Bookin
  scale_y_continuous(limits = c(0,500),breaks = seq(0,500, by=100)) +
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"))
Lead_CheckIn_Deposit_resort_plot
```

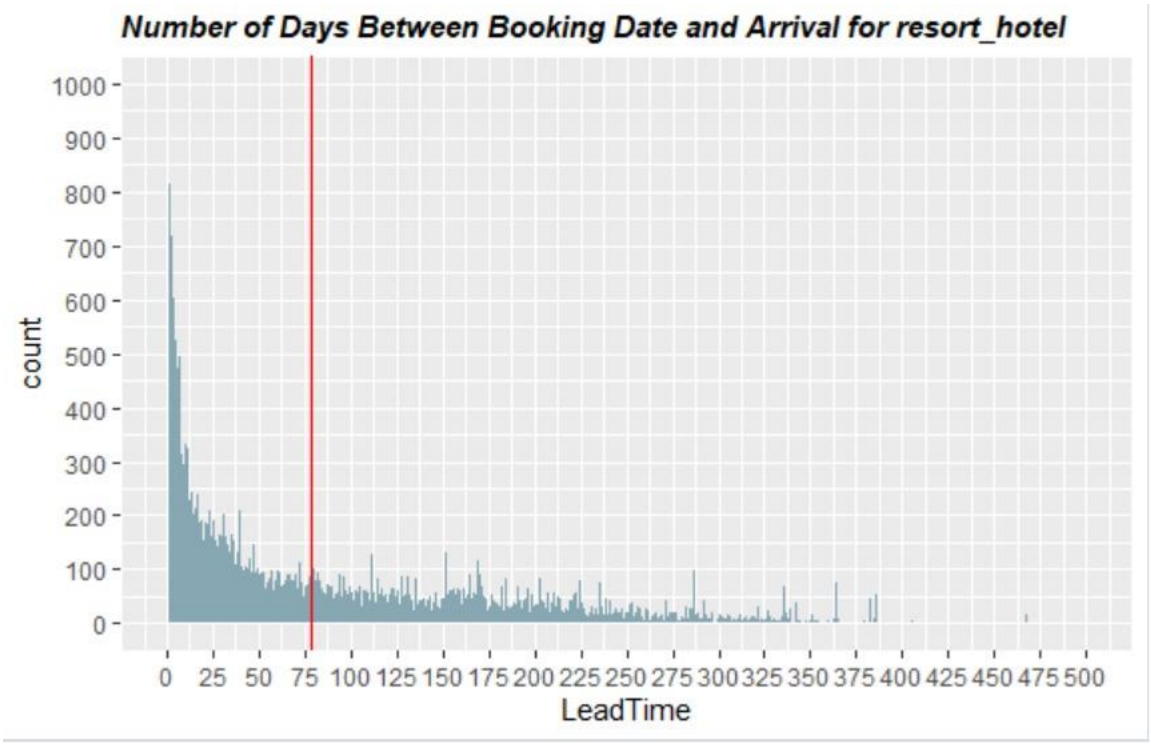

Figure 17 Lead Time plot for days between Booking and Arrival for Resort Data

In the above plot, we find the average lead time between the booking date and arrival date for the resort data set to be approximately 78 days. It is also interesting to note that a large number of customers book it immediately before arriving or a day before.

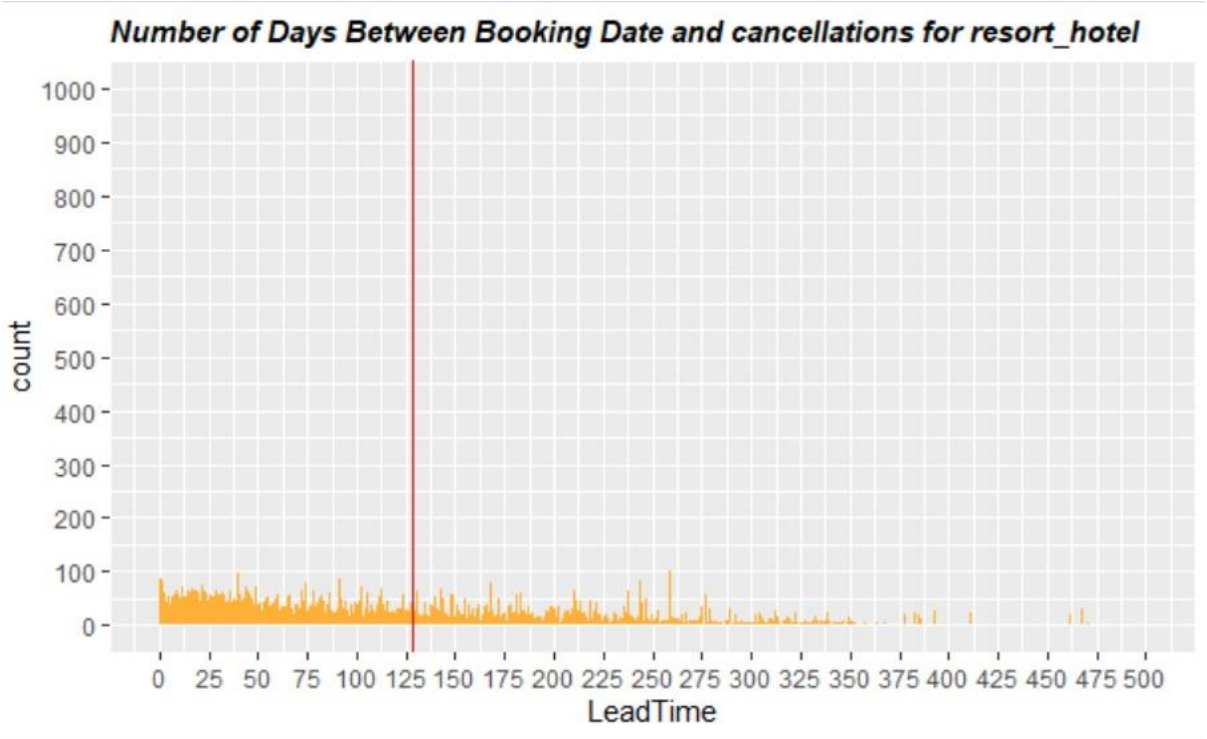

Figure 18 LeadTime plot for days between booking and Cancellations for Resort data

In the above plot, we find the average lead time between the booking date and the cancellation date to be about 128 days.

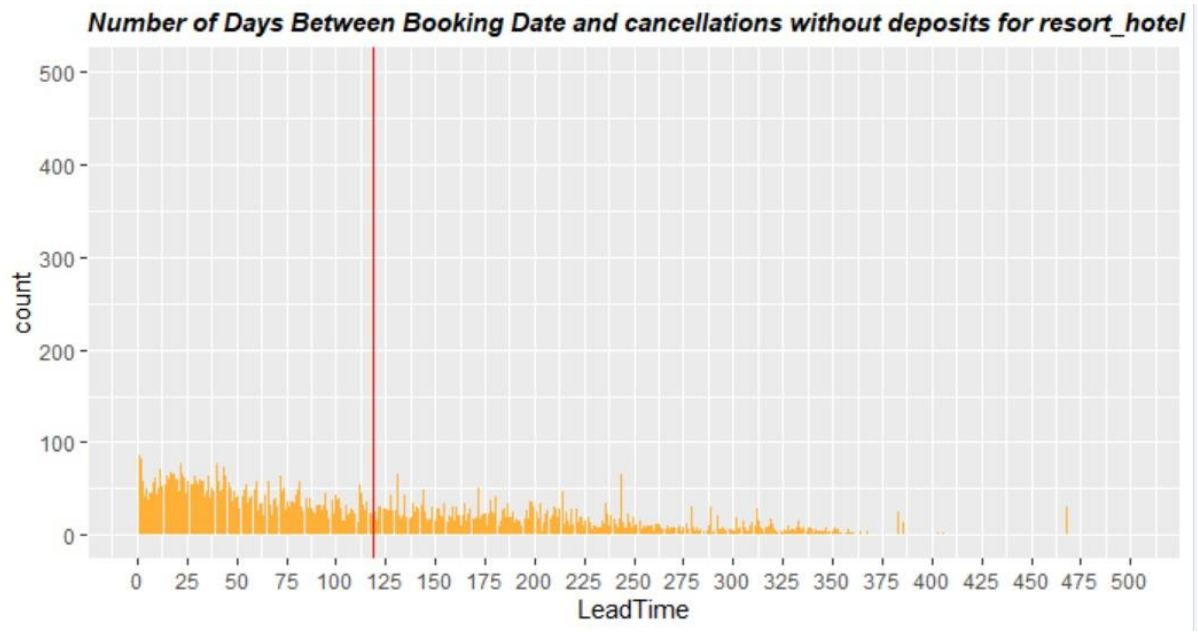

Figure 19 LeadTime plot for days between booking and cancellation with no deposit for Resort data

In the above plot, we find the average lead time between the booking date and cancellation date to be about 119 days. This plot is for the customers who haven't made deposits.

[3.5] Scatter plots

We now explore a series of visualizations using scatter plots.

First, we generate a scatter plot to check for the trend exhibiting customers who booked a certain room but getting a different room assigned ultimately.

```
#7
# How many times did people get a different room assigned
room_plot_city <- ggplot(data = city_hotel,aes(x= ReservedRoomType, y = AssignedRoomType)) + geom_jitter(shape = 21,size = 1,
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"), axis.title.x = element_text( size=9), axis.tit
room_plot_city

room_plot_resort <- ggplot(data = resort_hotel,aes(x= ReservedRoomType, y = AssignedRoomType)) + geom_jitter(shape = 21,size =
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"), axis.title.x = element_text( size=9), axis.tit
room_plot_resort
```

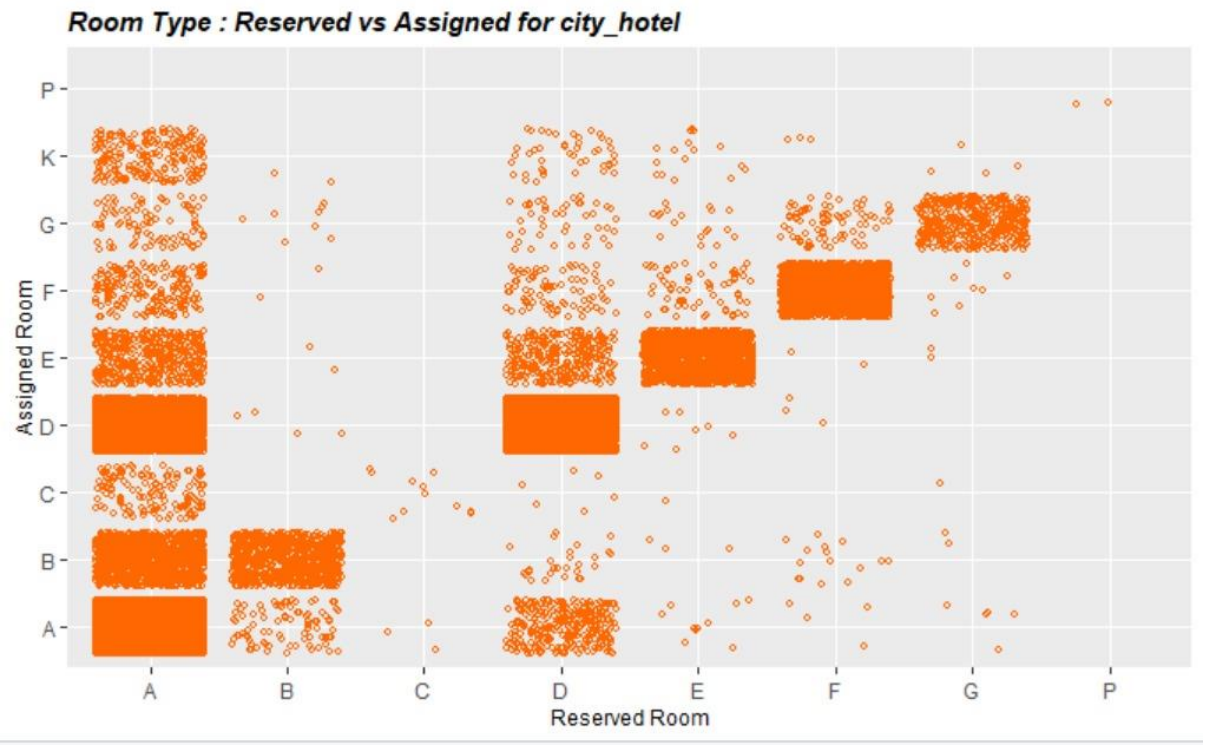

Figure 20 Reserved Room Type VS Assigned for City data

This above plot shows a lot of bookings for Room A followed by Room D. We also notice that due to overbookings or upgrades handed out, a lot of people booking A get a different category of room assigned, a similar trend is observed when the bookings for Room D are analyzed as well. Room P witnesses the least number of bookings followed by Room C.

Figure 21 Reserved Room Type VS Assigned for Resort data

This above plot shows a lot of bookings for Room A followed by Room D and E. We also notice that due to overbookings or upgrades handed out, a lot of people booking A get a different category of room assigned, a similar trend is observed when the bookings for Room D are analyzed as well. Room B witnesses the least number of bookings followed by Room L.

[3.6] Histograms

Next, we generate histograms to represent days spent in the waiting list for both the hotel and the resort data sets.

```
#8 - Histograms
hist_hotel <- ggplot(data= city_hotel, aes(x=DaysInWaitingList)) + geom_histogram(fill = "#7393B3" ) +g
hist_hotel
hist_resort <- ggplot(data = resort_hotel,aes(x=DaysInWaitingList)) + geom_histogram(fill = "#B99095")
hist_resort
```

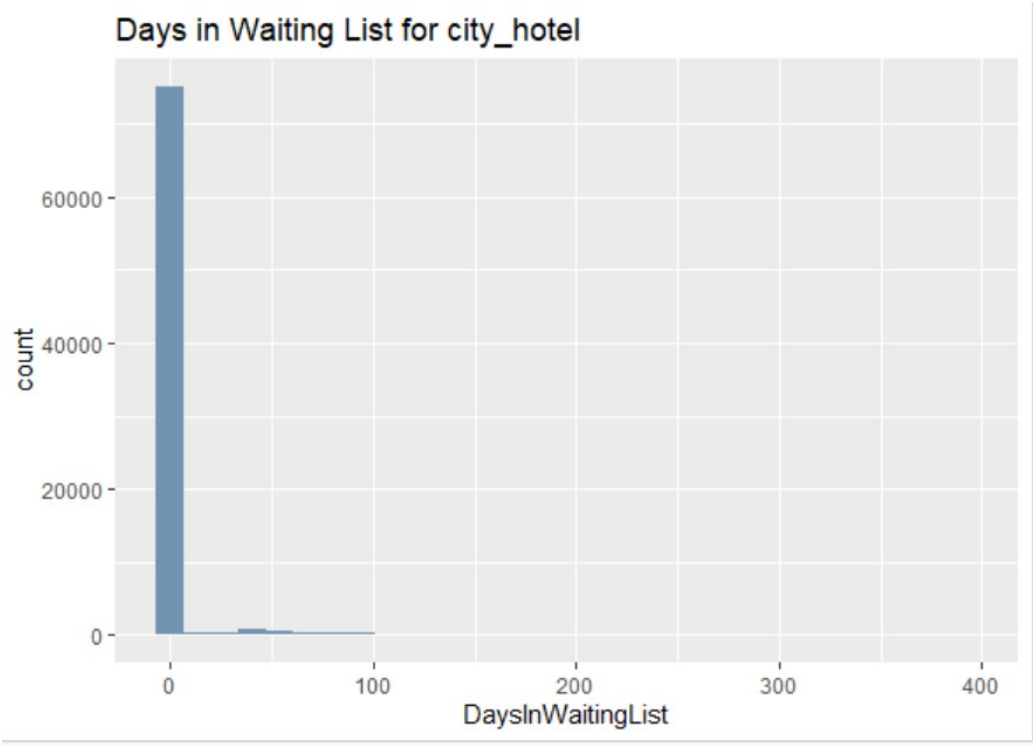

Figure 22 Histogram for Days in Waiting list of City data

The above plot shows a customer booking a hotel room gets the booking confirmed immediately almost all the time.

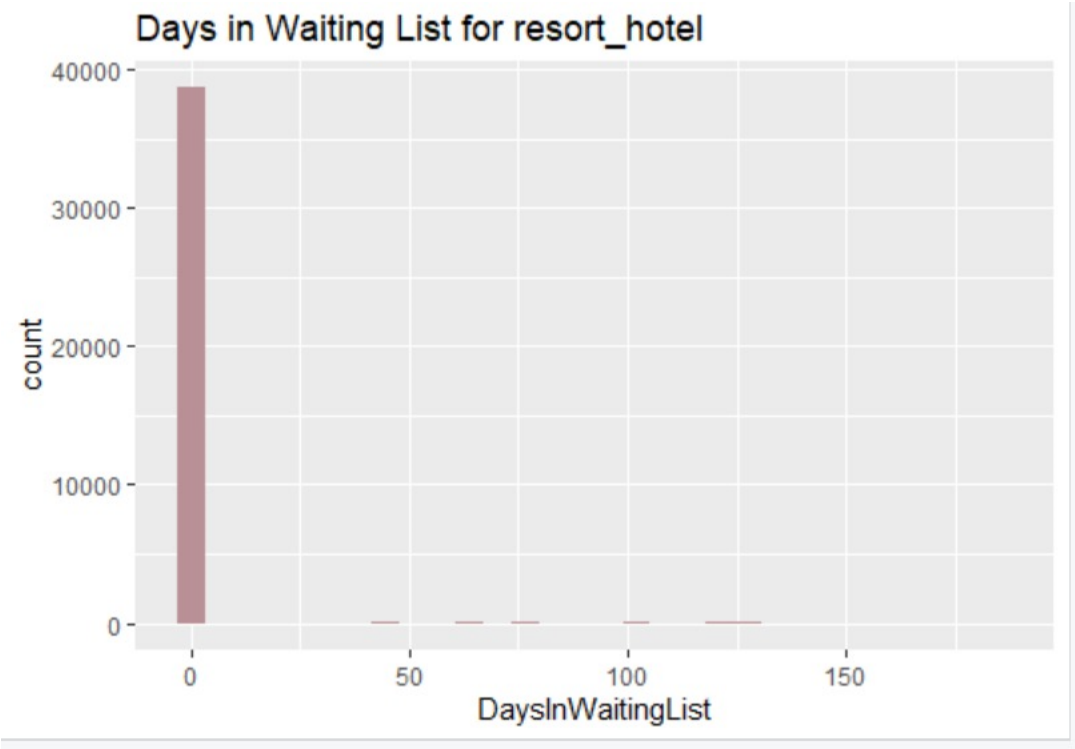

Figure 23 Histogram for Days in Waiting list of Resort data

The above plot shows a customer booking a hotel room gets the booking confirmed immediately almost all the time.

[3.7]Box Plots

Next, we generate a series of boxplots.

First, we generate a boxplot to compare ADR by customer type.

```
#8 - BoxPlots
#Generating ADR box plots by customer type
boxplot_adr_customertype_city <- ggplot(data = city_hotel, aes(x = CustomerType, y= ADR, fill = Custome
  ggtitle("Average Daily Rate by Customer Type") + scale_y_continuous(limits = c(0,500),breaks = seq(0,

boxplot_adr_customertype_city
boxplot_adr_customertype_resort <- ggplot(data = resort_hotel, aes(x = CustomerType, y= ADR, fill = Cus
  ggtitle("Average Daily Rate by Customer Type") + scale_y_continuous(limits = c(0,500),breaks = seq(0,

boxplot_adr_customertype_resort
```

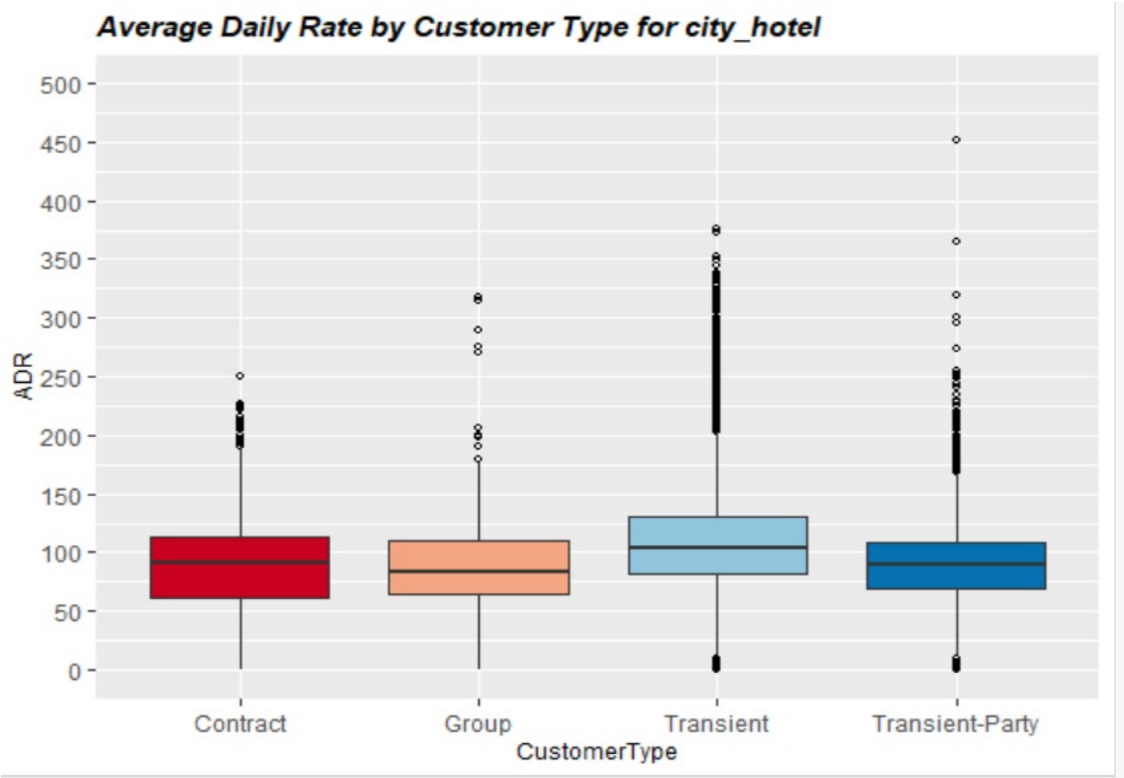

Figure 24 BoxPlots for Customer Type VS ADR of City data

For the hotel data set, we found that the median ADR for the Transcient customer type was the highest and the median ADR for the Group customer type was the least.

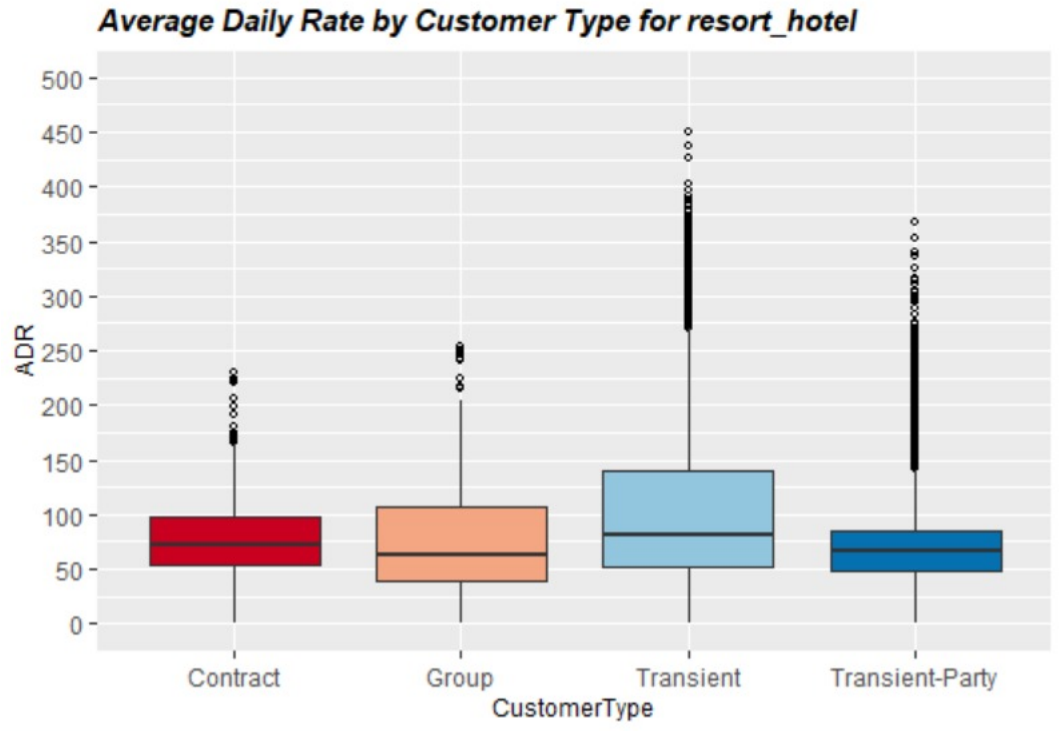

Figure 25 BoxPlot for Customer Type VS ADR for Resort data

For the resort data set, we found that the median ADR for the Transcient customer type was the highest and the median ADR for the Group customer type was the least.

Next, we generate a boxplot to compare ADR by room type.

```
#Generating ADR box plot by Room Type
boxplot_adr_roomtype_city <- ggplot(data = city_hotel, aes(x = ReservedRoomType, y= ADR, fill = Reserv
  ggtitle("Average Daily Rate by Room Type for city_hotel")+scale_y_continuous(limits = c(0,500),break

boxplot_adr_roomtype_city
boxplot_adr_roomtype_resort <- ggplot(data = resort_hotel, aes(x = ReservedRoomType, y= ADR, fill = Re
  ggtitle("Average Daily Rate by Room Type for resort_hotel")+scale_y_continuous(limits = c(0,500),bre

boxplot_adr_roomtype_resort
```

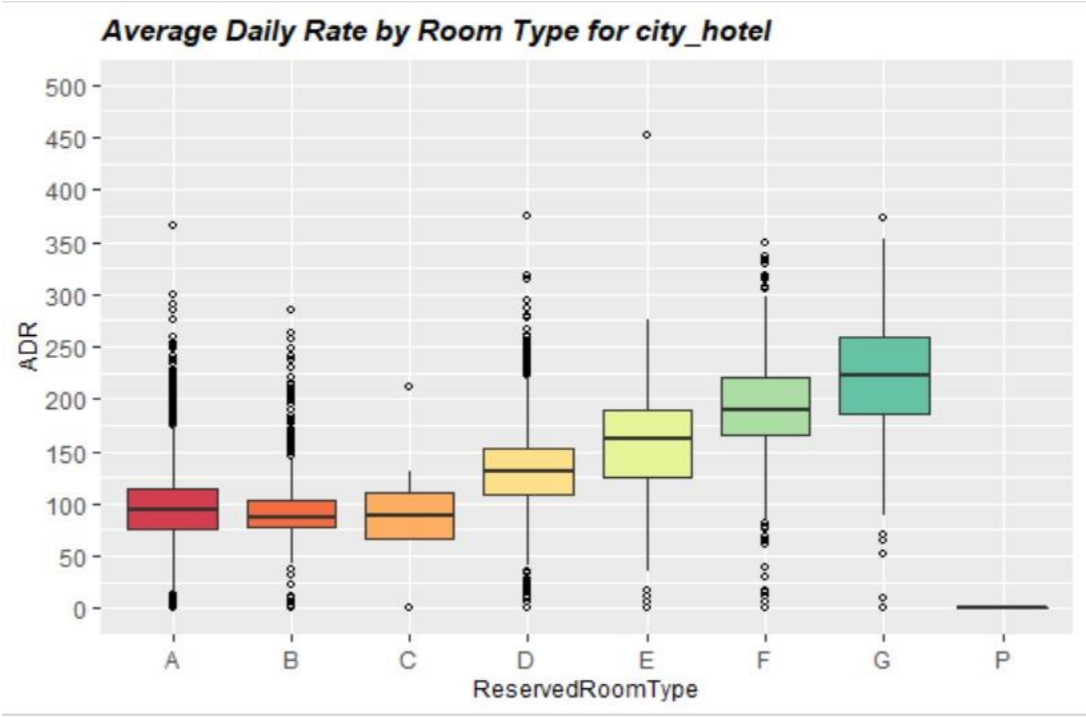

Figure 26 BoxPlot for Reserved Room Type VS ADR for City data

For the hotel data set, we found that the median ADR for the ReservedRoomType G was the highest and the median ADR for the ReservedRoomType P was the least.

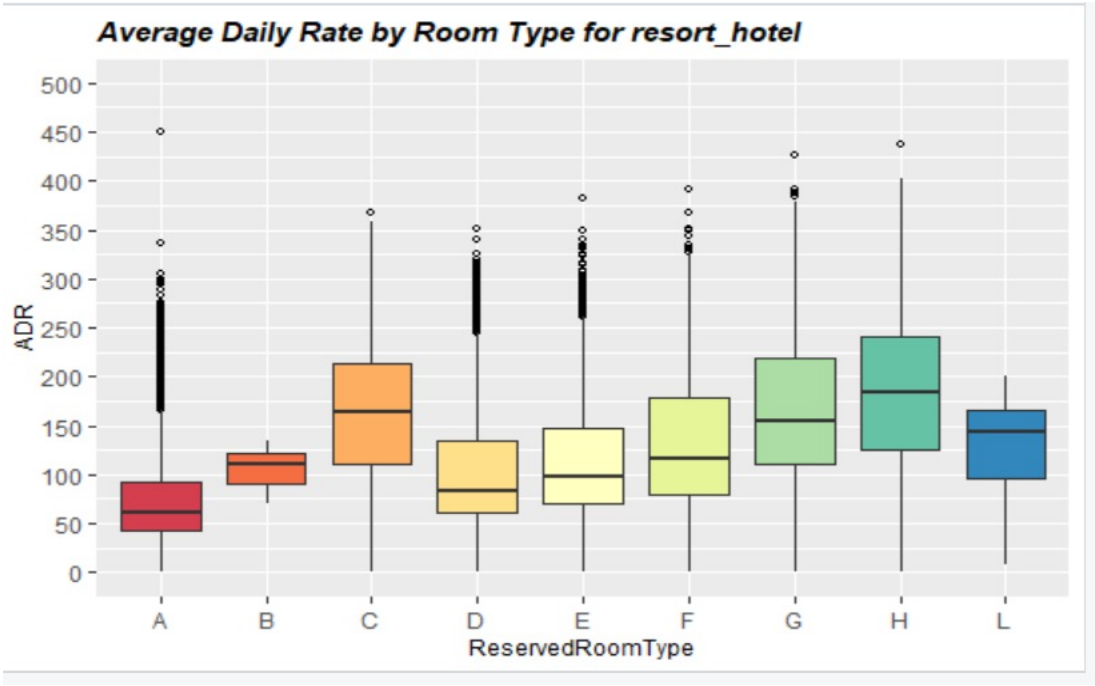

Figure 27 BoxPlot for Reserved Room Type VS ADR for Resort data

For the resort data set, we found that the median ADR for the ReservedRoomType H was the highest and the median ADR for the ReservedRoomType A was the least.

```
#Generating ADR box plot by meal plan
boxplot_adr_mealtype_city <- ggplot(data = city_hotel, aes(x = Meal, y= ADR, fill = Mea
  ggtitle("Average Daily Rate by Meal Type")+scale_y_continuous(limits = c(0,500),break

boxplot_adr_mealtype_city
boxplot_adr_mealtype_resort <- ggplot(data = resort_hotel, aes(x = Meal, y= ADR, fill =
  ggtitle("Average Daily Rate by Meal Type")+scale_y_continuous(limits = c(0,500),break

boxplot_adr_mealtype_resort
```

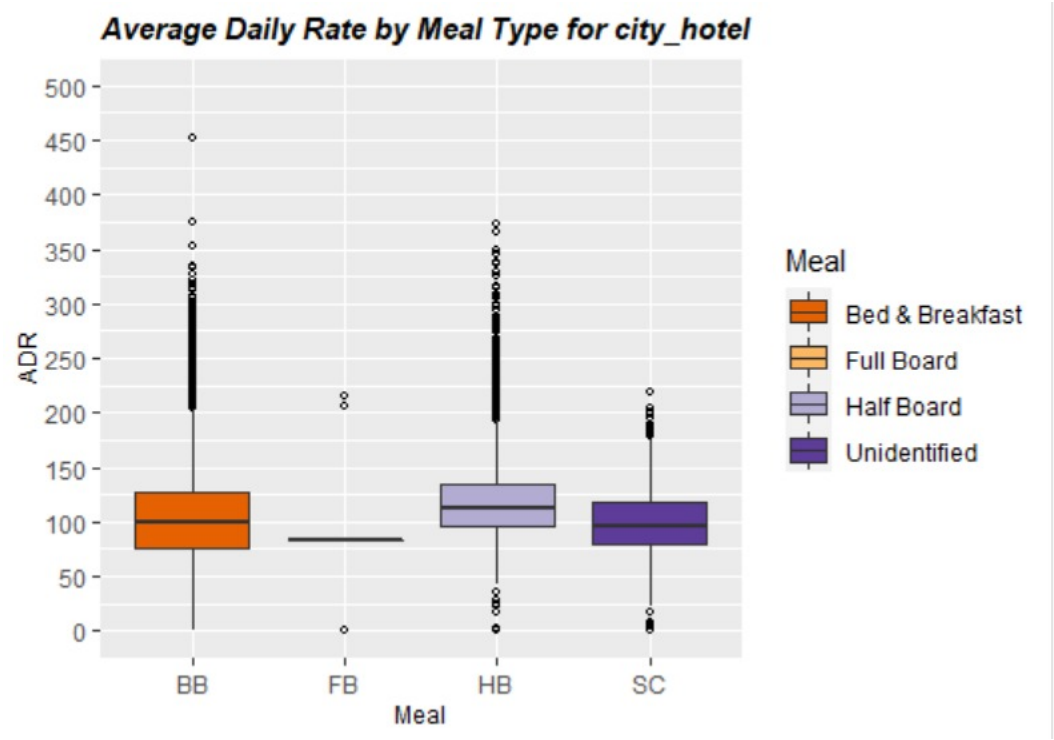

Figure 28 BoxPlot for Meal Type VS ADR for City data

For the hotel data set, we found that the median ADR for the meal type HB was the highest and the statistically significant median ADR for the meal type FB was the least.

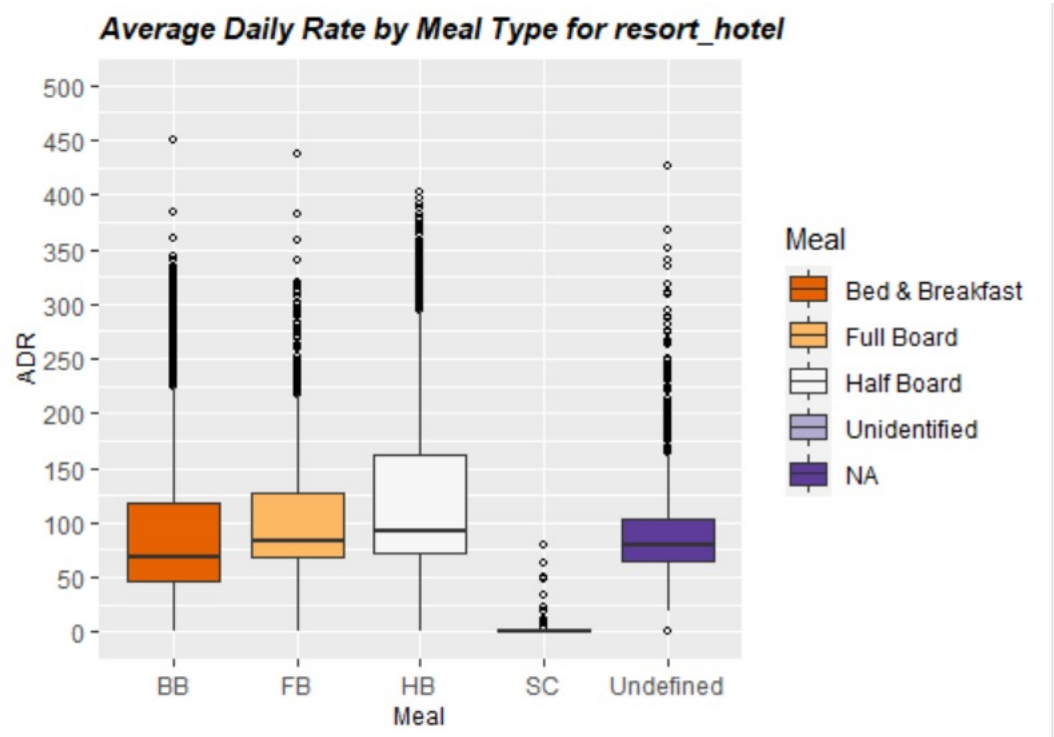

Figure 29 BoxPlot for Meal Type VS ADR for Resort data

For the resort data set, we found that the median ADR for the meal type HB was the highest and the statistically significant median ADR for the meal type BB was the least.

[3.8] Revenue Analysis

Upon examining the data set, we noticed that average revenue per days stayed and certain allied columns were also being populated for transactions which were cancelled or in the case of no-show. At this point, we realized that performing revenue analysis by taking those transactions into account would lead to erroneous inferences. However, we found it interesting that now we could compare the total transactions in both the data sets with those that actually followed through, i.e, the ones where customers actually went through with their stay. Hence, in our attempt to conduct revenue analysis, we will be comparing these two above situations and deducing insights based on “what could have been” and “what is”.

Revenue Analysis of the Hotel Data Set:

First, we conduct revenue analysis for the hotel data set.

```
#REVENUE ANALYSIS
#For city_hotel
#Dataframe for city_hotel to get revenue by season
sample_revenue_city <- data.frame(city_hotel$newArrivalDate,city_hotel$ReservationStatus,city_hotel$Avg_Revenue_per_St
#View(sample_revenue_hotel)

#Filtering data only for people who checked in
sample_revenue_checkin <- sample_revenue_city[(sample_revenue_city$city_hotel.ReservationStatus != 'Canceled' & sample
#View(sample_revenue_checkin) # All check-outs


#Creating a function to get sum of the column by condition.
sum_season <- function(j,m){
  j <- j[j$city_hotel.Season == m,]
  return(sum(j$city_hotel.Avg_Revenue_per_Stay))
}

#Creating a dataframe for average revenue by season. It has sum of all average revenues for those who checked in.
revenue_by_season <- data.frame(Season= c('Fall','Spring','Summer','Winter'))
revenue_by_season$Season <- as.factor(revenue_by_season$Season)
revenue_by_season <- mutate(revenue_by_season,Avg = case_when(revenue_by_season$Season == 'Fall' ~ sum_season(sample_r
                                                              revenue_by_season$Season == 'Spring' ~ sum_season(sample
                                                              revenue_by_season$Season == 'Summer' ~ sum_season(sample
                                                              revenue_by_season$Season == 'Winter' ~ sum_season(sample


))
str(revenue_by_season)
View(revenue_by_season)
```

```
#Creating a plot for average revenue of city_hotel by season for check-out data
plot_rev <- ggplot(data = revenue_by_season, aes( x= Season, y= Avg, group = 1)) +
  ggtitle("Check-Out in city_hotel")+
  geom_line(data = revenue_by_season, aes( x= Season, y= Avg, group = 1))+
  geom_point(shape = 21, size = 2,color = "black", fill = "#BCD979",data = revenue_by_season, aes( x= Season,
  ylab("Average Revenue (in millions)") +
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"), axis.title.x = element_text( si

plot_rev
```

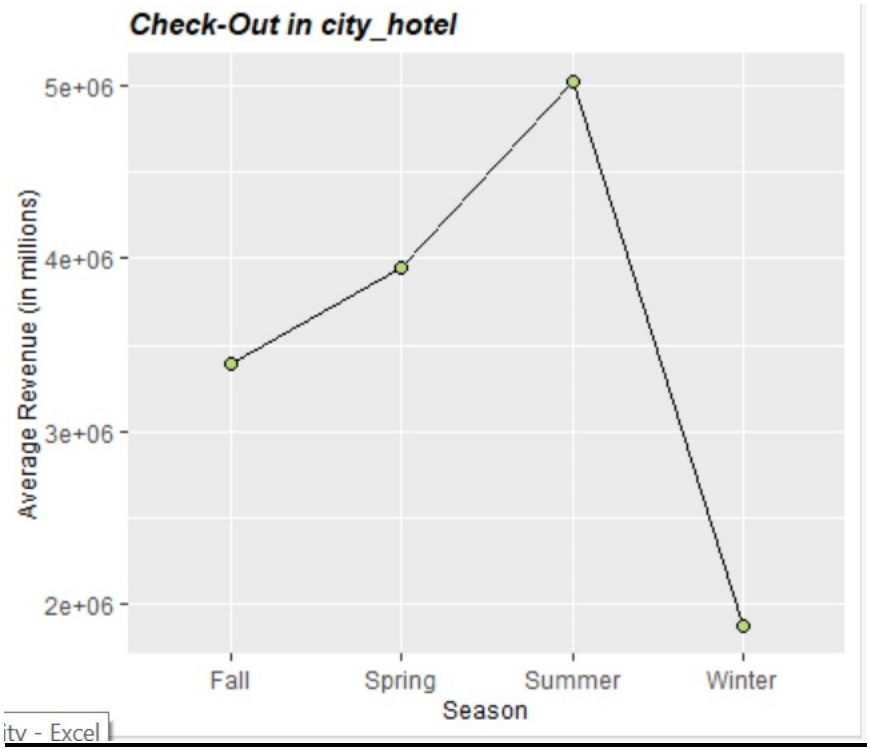

Figure 30 Average Revenue by Season for Check-outs in City Data

The above plot shows the distribution of average revenue (in millions), considering only checked-out customers, season wise. The results portray a peak in summer and a low in Winter.

```
#Filtering data only for people who canceled
sample_revenue_cancel <- sample_revenue_city
View(sample_revenue_cancel) # All check-outs

#Creating a function to get sum of the column by condition.
sum_season <- function(j,m){
  j <- j[j$city_hotel.Season == m,]
  return(sum(j$city_hotel.Avg_Revenue_per_Stay))
}

#Creating a dataframe for average revenue by season. It has  sum of all average revenues for those who checked in.
revenue_by_season_cancel <- data.frame(Season_cancel= c('Fall','Spring','Summer','Winter'))
revenue_by_season_cancel$Season_cancel <- as.factor(revenue_by_season_cancel$Season_cancel)
revenue_by_season_cancel <- mutate(revenue_by_season_cancel,Avg_cancel = case_when(revenue_by_season_cancel$Season == 'Fall' ~ sum_se
                                                                                   revenue_by_season_cancel$Season == 'Spring' ~ sum_
                                                                                   revenue_by_season_cancel$Season == 'Summer' ~ sum_
                                                                                   revenue_by_season_cancel$Season == 'Winter' ~ sum_


))
str(revenue_by_season_cancel)
View(revenue_by_season_cancel)
```

```
#Creating a plot for average revenue of hotels by season for checkin data
plot_rev_cancel <- ggplot(data = revenue_by_season_cancel, aes( x= Season_cancel, y= Avg_cancel, group = 1)) +
  ggtitle("If no cancellations in hotel")+
  geom_line(data = revenue_by_season_cancel, aes( x= Season_cancel, y= Avg_cancel, group = 1))+
  geom_point(shape = 21, size = 2,color = "black", fill = "#ec4646",data = revenue_by_season_cancel, aes( x= Season_cancel, y= Avg_ca
  ylab("Average Revenue (in millions)") +
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"), axis.title.x = element_text( size=9), axis.title.y = e

plot_rev_cancel
```

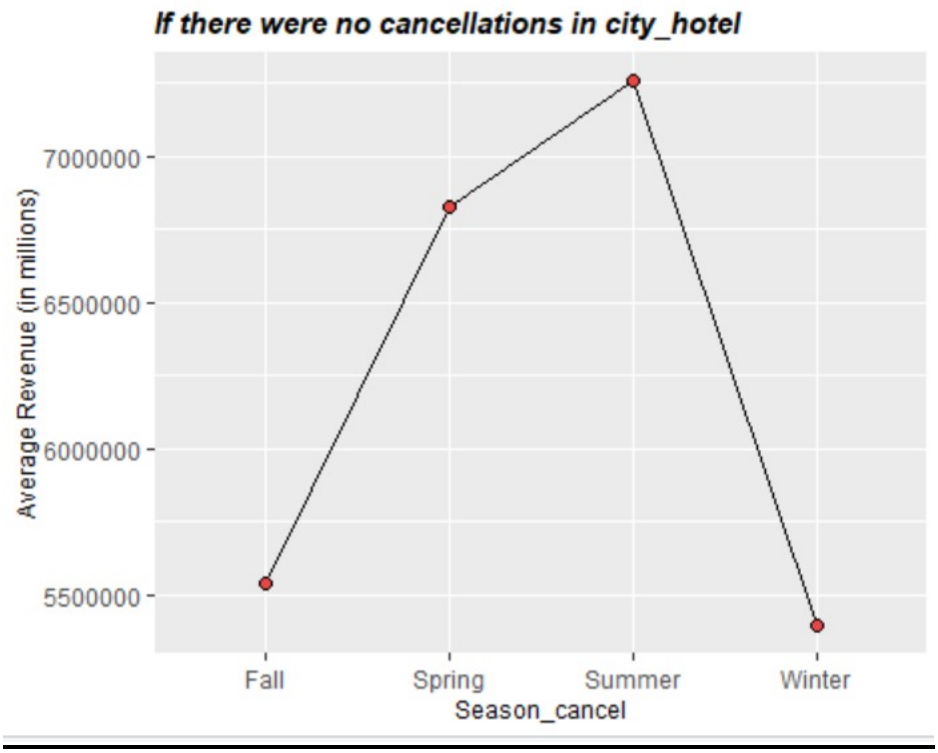

Figure 31 Average Revenue by Season for City Data

The above representation is a “what could have been” scenario. If the cancelled bookings had gone through, this is how the average revenue per season would be distributed. Again, the average revenue peaks in summer and bottoms out in Winter.

```
combined_df_city <- data.frame(revenue_by_season,revenue_by_season_cancel)
View(combined_df_city)
combined_plot_revenue_city <- ggplot(data = combined_df_city, aes ( x = Season, y = Avg)) + ggtitle("What
  geom_line(data = combined_df_city, aes( x= Season, y= Avg, group = 1))+
  geom_point(shape = 21, size = 2,color = "black", fill = "#BCD979",data = combined_df_city, aes( x= Seas
  geom_line(data = revenue_by_season_cancel, aes( x= Season_cancel, y= Avg_cancel, group = 1))+
  geom_point(shape = 21, size = 2,color = "black", fill = "#ec4646",data = combined_df_city, aes( x= Seas
  ylab("Average Revenue (in millions)") +
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"), axis.title.x = element_tex
combined_plot_revenue_city
```

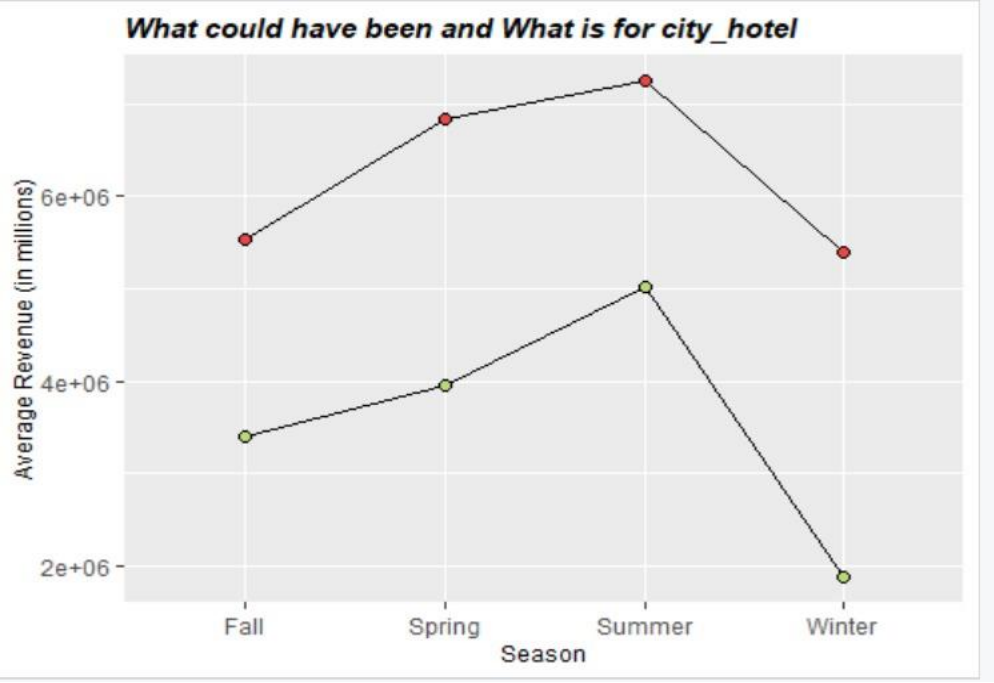

Figure 32 Combined Projection by Season for City Data

The above plot puts things in perspective and shows the amount of missed revenue due to cancellations. It combines the “what is” and “what could have been” scenarios and combines the previous two plots.

Revenue Analysis of the Resort Data Set:

```
#For resort_hotel
#Dataframe for hotels to get revenue by season
sample_revenue_resort <- data.frame(resort_hotel$newArrival,resort_hotel$ReservationStatus,resort_hotel$Avg_Revenue_
#View(sample_revenue_hotel)

#Filtering data only for people who checked in
sample_revenue_checkin_resort <- sample_revenue_resort[(sample_revenue_resort$resort_hotel.ReservationStatus != 'Can
#View(sample_revenue_checkin) # All check-outs


#creating a function to get sum of the column by condition.
sum_season_resort <- function(j,m){
  j <- j[j$resort_hotel.Season == m,]
  return(sum(j$resort_hotel.Avg_Revenue_per_Stay))
}

#Creating a dataframe for average revenue by season. It has sum of all average revenues for those who checked in.
revenue_by_season_resort <- data.frame(Season= c('Fall','Spring','Summer','Winter'))
revenue_by_season_resort$Season <- as.factor(revenue_by_season_resort$Season)
revenue_by_season_resort <- mutate(revenue_by_season_resort,Avg = case_when(revenue_by_season_resort$Season == 'Fall
                                                                            revenue_by_season_resort$Season == 'Spri
                                                                            revenue_by_season_resort$Season == 'Summ
                                                                            revenue_by_season_resort$Season == 'Wint


))
```

```
#creating a plot for average revenue of hotels by season for checkin data
plot_rev_resort <- ggplot(data = revenue_by_season_resort, aes( x= Season, y= Avg, group = 1)) +
  ggtitle("Revenue for Check-Out in Resorts by season")+
  geom_line(data = revenue_by_season_resort, aes( x= Season, y= Avg, group = 1))+
  geom_point(shape = 21, size = 2,color = "black", fill = "blue",data = revenue_by_season_resort, aes( x= Season, y=
  ylab("Average Revenue (in millions)") +
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"), axis.title.x = element_text( size=9),

plot_rev_resort
```

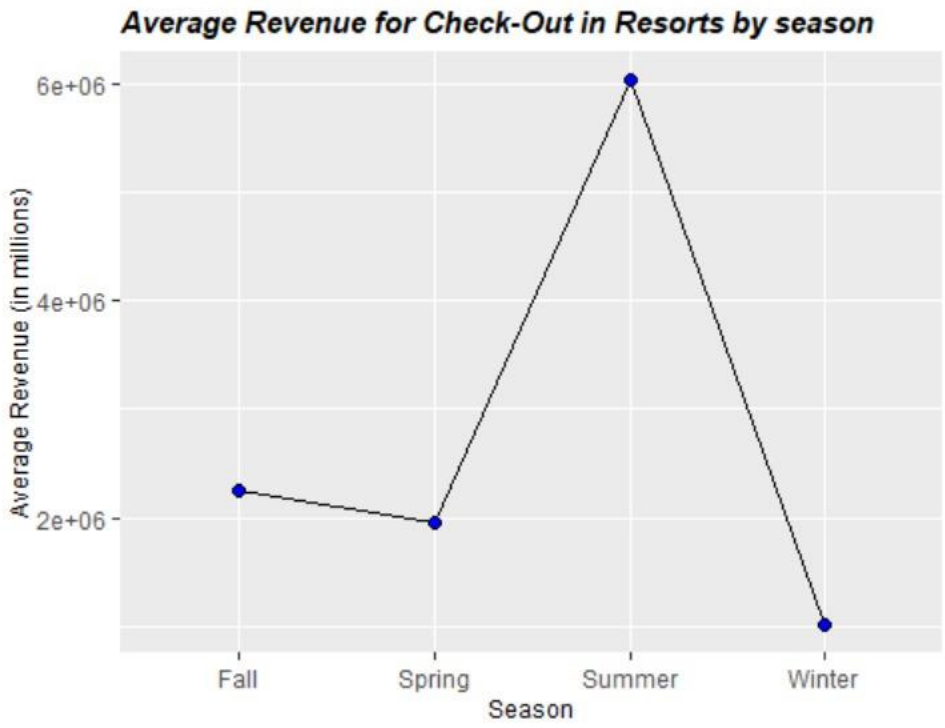

Figure 33 Average Revenue by Season for Check-outs in Resort Data

The above plot shows the distribution of average revenue (in millions), considering only checked-out customers, season wise. The results portray a peak in summer and a low in Winter.

```
#Filtering data only for people who canceled
sample_revenue_cancel_resort <- sample_revenue_resort
View(sample_revenue_cancel_resort) # All check-outs

#creating a function to get sum of the column by condition.
sum_season_resort <- function(j,m){
  j <- j[j$resort_hotel.Season == m,]
  return(sum(j$resort_hotel.Avg_Revenue_per_Stay))
}

#Creating a dataframe for average revenue by season. It has sum of all average revenues for those who checked
revenue_by_season_cancel_resort <- data.frame(Season_cancel_resort= c('Fall','Spring','Summer','Winter'))
revenue_by_season_cancel_resort$Season_cancel_resort <- as.factor(revenue_by_season_cancel_resort$Season_canc
revenue_by_season_cancel_resort <- mutate(revenue_by_season_cancel_resort,Avg_cancel = case_when(revenue_by_s
                                                                                                 revenue_by_s
                                                                                                 revenue_by_s
                                                                                                 revenue_by_s


))
str(revenue_by_season_cancel_resort)
View(revenue_by_season_cancel_resort)
```

```
#creating a plot for average revenue of hotels by season for checkin data
plot_rev_cancel_resort <- ggplot(data = revenue_by_season_cancel_resort, aes( x= Season_cancel_resort, y= Avg_canc
  ggtitle("If no cancellations in resort_hotel")+
  geom_line(data = revenue_by_season_cancel_resort, aes( x= Season_cancel_resort, y= Avg_cancel, group = 1))+
  geom_point(shape = 21, size = 2,color = "black", fill = "orange",data = revenue_by_season_cancel_resort, aes( x=
  ylab("Average Revenue (in millions)") +
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"), axis.title.x = element_text( size=9

plot_rev_cancel_resort
```

Figure 34 Average Revenue by Season for Resort Data

The above representation is a “what could have been” scenario. If the cancelled bookings had gone through, this is how the average revenue per season would be distributed. Again, the average revenue peaks in summer and bottoms out in Winter.

```
combined_df_resort <- data.frame(revenue_by_season_resort,revenue_by_season_cancel_resort)
View(combined_df_resort)
combined_plot_revenue_resort <- ggplot(data = combined_df_resort, aes ( x = Season, y = Avg)) + ggtitle("
  geom_line(data = combined_df_resort, aes( x= Season, y= Avg, group = 1))+
  geom_point(shape = 21, size = 2,color = "black", fill = "blue",data = combined_df_resort, aes( x= Seaso
  geom_line(data = combined_df_resort, aes( x= Season, y= Avg_cancel, group = 1))+
  geom_point(shape = 21, size = 2,color = "black", fill = "orange",data = combined_df_resort, aes( x= Sea
  ylab("Average Revenue (in millions)") +
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"), axis.title.x = element_tex
combined_plot_revenue_resort
```

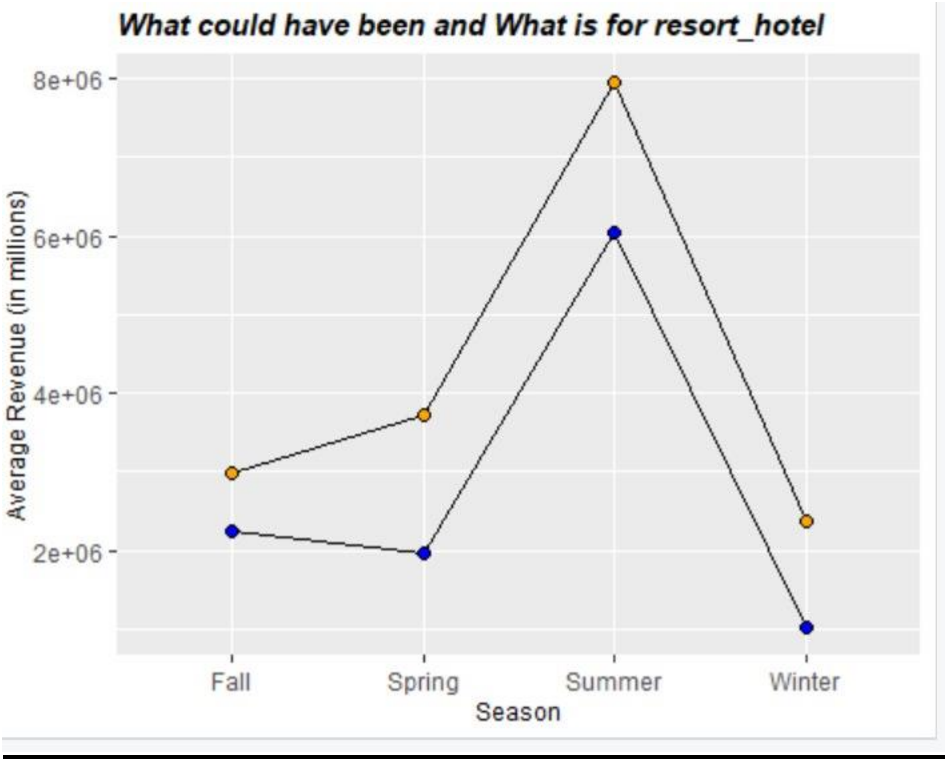

Figure 35 Combined Projection by Season for Resort Data

The above plot puts things in perspective and shows the amount of missed revenue due to cancellations. It combines the “what is” and “what could have been” scenarios and combines the previous two plots.

Next, we proceed to combine and contrast the revenue analysis findings for both the hotel and the resort data sets.

```
#Combining city_hotel and resort_hotel
#Plot for average revenue of hotel and resort by season for checkin
combined_df_checkin <- data.frame(revenue_by_season,revenue_by_season_resort)
View(combined_df_checkin)
str(combined_df_checkin)
combined_plot_checkin <- ggplot(data = combined_df_checkin, aes ( x = Season, y = Avg)) + ggtitle("Average Re
  geom_line(data = combined_df_checkin, aes( x= Season, y= Avg, group = 1))+
  geom_point(shape = 21, size = 2,color = "black", fill = "#BCD979",data = combined_df_checkin, aes( x= Seasc
  geom_line(data = combined_df_checkin, aes( x= Season, y= Avg.1, group = 1))+
  geom_point(shape = 21, size = 2,color = "black", fill = "blue",data = combined_df_checkin, aes( x= Season,
  ylab("Average Revenue (in millions)") +
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"), axis.title.x = element_text( s
combined_plot_checkin #blue is for resort, green for hotel
```

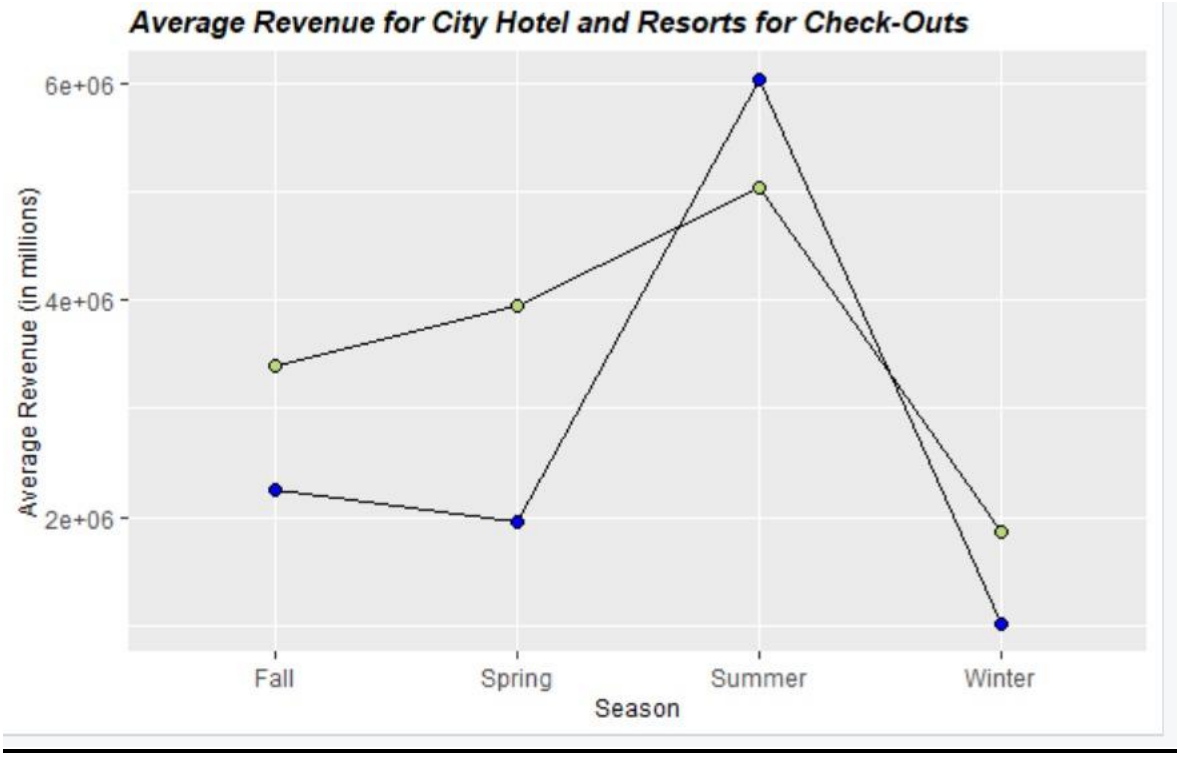

Figure 36 Combined Projection by Season for City Data and Resort Data for Check-outs

The above plot combines the distribution of average revenue season wise for both the hotel and the resort data set, considering only checked out customers in both the cases. Both curves peak in the summer and bottom out in the winter.

```
#Plot for average revenue of hotel and resort by season for cancellations
combined_df_cancel <- data.frame(revenue_by_season_cancel,revenue_by_season_cancel_resort)
view(combined_df_cancel)
str(combined_df_cancel)
combined_plot_cancel <- ggplot(data = combined_df_cancel, aes ( x = Season_cancel, y = Avg_cancel)) + ggtit
  geom_line(data = combined_df_cancel, aes( x= Season_cancel, y= Avg_cancel, group = 1))+
  geom_point(shape = 21, size = 2,color = "black", fill = "#ec4646",data = combined_df_cancel, aes( x= Seas
  geom_line(data = combined_df_cancel, aes( x= Season_cancel, y= Avg_cancel.1, group = 1))+
  geom_point(shape = 21, size = 2,color = "black", fill = "orange",data = combined_df_cancel, aes( x= Seasc
  ylab("Average Revenue (in millions)") +
  theme(plot.title = element_text(color="Black", size=11, face="bold.italic"), axis.title.x = element_text(
combined_plot_cancel #orange is for resort, red for hotel
```

Figure 37 Combined Projection by Season for City Data and Resort Data (What Could have been)

The above plot combines the distribution of average revenue season wise for both the hotel and the resort data set, considering all bookings, including the cancelled ones went through. Both curves peak in the summer and have least values during winter.

One striking observation from the above combined plot is that hotels have a more even distribution of revenue across seasons as compared to resorts. Resorts seem to generate disproportionate revenue during summers as compared to the other seasons, whereas hotels have a more even spread of revenue across seasons.

[4] Regression Modelling

For the regression modelling we used the output variable to be ADR and then tried a number of variables to find out their variability in likelihood to recommend. We are observing the changes in the values of ADR and how it is affected by the other variables. Step-wise linear regression analysis was used. First Top-down approach was used. Sample of it is included below. Based on the Pr/Probability values, only significant attributes were chosen and step-wise regression on them was performed. By doing this we excluded out the insignificant attributes to get accurate and reliable predictions. The same approach was used for both datasets.

```
Call:
lm(formula = ADR ~ ., data = city_hotel)

Residuals:
   Min     1Q Median     3Q    Max
-443.9   -8.1   -0.9    6.5 4392.4

Coefficients: (7 not defined because of singularities)
                                                 Estimate Std. Error  t value Pr(>|t|)
(Intercept)                                    -4.088e+02  2.992e+01  -13.662  < 2e-16 ***
IsCanceled1                                     1.661e+00  2.433e-01    6.827 8.73e-12 ***
LeadTime                                       -3.584e-02  1.099e-03  -32.599  < 2e-16 ***
ReservationStatusDate                           1.209e-03  1.711e-05   70.655  < 2e-16 ***
ReservationStatusCheck-Out                             NA         NA       NA       NA
ReservationStatusNo-Show                       -6.155e-01  8.284e-01   -0.743 0.457461
StaysInWeekendNights                           -1.807e+01  1.332e-01 -135.639  < 2e-16 ***
StaysInWeekNights                              -1.801e+01  1.050e-01 -171.468  < 2e-16 ***
Adults                                          5.156e+00  5.665e-01    9.102  < 2e-16 ***
Children                                        3.233e+00  5.505e-01    5.874 4.27e-09 ***
Babies                                         -3.289e+00  1.072e+00   -3.067 0.002162 **
MealFB                                          3.999e+00  3.647e+00    1.096 0.272965
MealHB                                          1.063e+01  3.930e-01   27.055  < 2e-16 ***
MealSC                                         -4.731e+00  2.975e-01  -15.906  < 2e-16 ***
CountryAlgeria                                  8.147e-01  8.323e+00    0.098 0.922026
CountryAmerican Samoa                          -1.362e+01  2.499e+01   -0.545 0.585729
CountryAndorra                                 -4.101e+00  1.854e+01   -0.221 0.824932
CountryAngola                                  -5.425e+00  8.036e+00   -0.675 0.499621
CountryAnguilla                                 3.224e+00  2.503e+01    0.129 0.897508
CountryAntarctica                               1.773e+01  1.854e+01    0.956 0.339034
CountryArgentina                                5.070e+00  8.149e+00    0.622 0.533825
CountryArmenia                                  1.406e+01  1.252e+01    1.123 0.261584
CountryAruba                                    3.462e-02  1.857e+01    0.002 0.998512
CountryAustralia                                7.034e+00  8.026e+00    0.876 0.380840
```

For *H1-Resort* Data set:

```
lm(formula = ADR ~ AssignedRoomType, data = resort_hotel)

Residuals:
    Min      1Q  Median      3Q     Max
-178.16  -39.25  -16.75   27.61  393.25

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       81.5459     0.4462 182.739  < 2e-16 ***
AssignedRoomTypeB 20.5850     4.6496   4.427 9.57e-06 ***
AssignedRoomTypeC 33.2025     1.3144  25.260  < 2e-16 ***
AssignedRoomTypeD  5.6009     0.7261   7.713 1.25e-14 ***
AssignedRoomTypeE 26.3163     0.8982  29.300  < 2e-16 ***
AssignedRoomTypeF 39.2172     1.4730  26.625  < 2e-16 ***
AssignedRoomTypeG 78.6330     1.4243  55.208  < 2e-16 ***
AssignedRoomTypeH 90.2306     2.2177  40.686  < 2e-16 ***
AssignedRoomTypeI -40.7621    3.1066 -13.121  < 2e-16 ***
AssignedRoomTypeL -73.5459   57.4355  -1.280      0.2
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 57.43 on 38875 degrees of freedom
Multiple R-squared:  0.1308,    Adjusted R-squared:  0.1306
F-statistic: 649.7 on 9 and 38875 DF,  p-value: < 2.2e-16
```

The above model was used for *H1-Resort* data set and it shows that *AssignedRoomType* accounts for about 13% of the variability in likelihood to recommend. The coefficient of AssignedRoomTypeI and AssignedRoomTypeL are negative which means that it is negatively contributing to the ADR.

```
Call:
lm(formula = ADR ~ Meal, data = resort_hotel)

Residuals:
    Min      1Q  Median      3Q     Max
-120.48  -42.22  -20.22   30.78  419.78

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   88.2195     0.3529 250.016   <2e-16 ***
MealFB        22.6621     2.2182  10.216   <2e-16 ***
MealHB        32.2563     0.7611  42.382   <2e-16 ***
MealSC       -81.7071     6.7235 -12.152   <2e-16 ***
MealUndefined  3.7760     1.8012   2.096   0.0361 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 60.05 on 38880 degrees of freedom
Multiple R-squared:  0.04951,   Adjusted R-squared:  0.04941
F-statistic: 506.3 on 4 and 38880 DF,  p-value: < 2.2e-16
```

The above model shows that for the *H1-Resort* dataset the type of *Meal* accounts for about 4% of the variability in likelihood to recommend. MealSc is negatively contributing to the ADR.

```
Call:
lm(formula = ADR ~ TotalPeople, data = resort_hotel)

Residuals:
     Min       1Q   Median       3Q      Max
-1401.82   -36.80   -15.85    25.99   413.20

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  45.4761     0.7378   61.64   <2e-16 ***
TotalPeople  24.6609     0.3360   73.40   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 57.73 on 38883 degrees of freedom
Multiple R-squared:  0.1217,    Adjusted R-squared:  0.1217
F-statistic:  5387 on 1 and 38883 DF,  p-value: < 2.2e-16
```

The above model shows that for the *H1-Resort* dataset the *TotalPeople* accounts for about 12% of the variability in likelihood to recommend.

```
Call:
lm(formula = ADR ~ ReservedRoomType + TotalPeople, data = resort_hotel)

Residuals:
    Min      1Q  Median      3Q     Max
-813.56  -33.74  -14.14   24.26  429.06

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       51.2176     0.7541  67.916   <2e-16 ***
ReservedRoomTypeB 25.7274    31.2654   0.823   0.4106
ReservedRoomTypeC 62.8575     1.9230  32.688   <2e-16 ***
ReservedRoomTypeD 24.4079     0.7330  33.298   <2e-16 ***
ReservedRoomTypeE 35.6905     0.8628  41.364   <2e-16 ***
ReservedRoomTypeF 52.6391     1.6926  31.099   <2e-16 ***
ReservedRoomTypeG 69.7609     1.5284  45.642   <2e-16 ***
ReservedRoomTypeH 85.6741     2.3566  36.355   <2e-16 ***
ReservedRoomTypeL 43.4172    22.1098   1.964   0.0496 *
TotalPeople       13.8609     0.3642  38.057   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 54.15 on 38875 degrees of freedom
Multiple R-squared:  0.2273,    Adjusted R-squared:  0.2271
F-statistic:  1271 on 9 and 38875 DF,  p-value: < 2.2e-16
```

The above mode shows that *ReservedRoomType* + *TotalPeople* account for 22% of the variability in likelihood to recommend.

```
Call:
lm(formula = ADR ~ LeadTime + DepositFactor + ReservedRoomType +
    Meal + VisitorType + Season, data = resort_hotel)

Residuals:
    Min      1Q  Median      3Q     Max
-257.35  -22.44   -4.96   16.19  381.17

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        70.787359   1.120968  63.148  < 2e-16 ***
LeadTime           -0.045181   0.002361 -19.136  < 2e-16 ***
DepositFactor      -7.850188   0.957809  -8.196 2.56e-16 ***
ReservedRoomTypeB  16.536071  24.340380   0.679    0.497
ReservedRoomTypeC  37.064722   1.557113  23.803  < 2e-16 ***
ReservedRoomTypeD  16.214793   0.582965  27.814  < 2e-16 ***
ReservedRoomTypeE  29.621034   0.683472  43.339  < 2e-16 ***
ReservedRoomTypeF  41.803971   1.327191  31.498  < 2e-16 ***
ReservedRoomTypeG  60.869532   1.255056  48.499  < 2e-16 ***
ReservedRoomTypeH  71.652285   1.890641  37.898  < 2e-16 ***
ReservedRoomTypeL  -3.893266  17.216297  -0.226    0.821
MealFB             30.961632   1.598164  19.373  < 2e-16 ***
MealHB             27.703594   0.547628  50.588  < 2e-16 ***
MealSC            -78.054180   4.720691 -16.534  < 2e-16 ***
MealUndefined      30.627074   1.285482  23.825  < 2e-16 ***
VisitorTypeFamily  29.307955   0.826877  35.444  < 2e-16 ***
VisitorTypeSingle -19.200882   0.605981 -31.686  < 2e-16 ***
SeasonSpring        6.385772   0.617814  10.336  < 2e-16 ***
SeasonSummer       63.938250   0.616931 103.639  < 2e-16 ***
SeasonWinter       -9.839862   0.646962 -15.209  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 42.15 on 38866 degrees of freedom
Multiple R-squared:  0.5319,    Adjusted R-squared:  0.5317
F-statistic:  2325 on 19 and 38866 DF,  p-value: < 2.2e-16
```

The above model takes into consideration a lot of variables and by looking at the result we can say that for H1-Resort dataset the variables account for 53.17% of the variability in likelihood to recommend.

For *H2-City* data set:

```
Call:
lm(formula = ADR ~ AssignedRoomType, data = city_hotel)

Residuals:
   Min     1Q Median     3Q    Max
-184.1  -21.6   -1.6   20.4 5303.4

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        96.6257     0.1658 582.897  < 2e-16 ***
AssignedRoomTypeB  -2.6724     0.9003  -2.968  0.00300 **
AssignedRoomTypeC   3.9421     3.1417   1.255  0.20957
AssignedRoomTypeD  24.9479     0.3634  68.645  < 2e-16 ***
AssignedRoomTypeE  47.0889     0.8679  54.259  < 2e-16 ***
AssignedRoomTypeF  81.9399     0.8957  91.477  < 2e-16 ***
AssignedRoomTypeG  87.4364     1.5072  58.013  < 2e-16 ***
AssignedRoomTypeK -42.9268     2.3668 -18.137  < 2e-16 ***
AssignedRoomTypeP -96.6257    27.8858  -3.465  0.00053 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 39.44 on 78721 degrees of freedom
Multiple R-squared:  0.1832,    Adjusted R-squared:  0.1831
F-statistic:  2207 on 8 and 78721 DF,  p-value: < 2.2e-16
```

The above model shows that the *AssignedRoomType* accounts for about 18% of the variability in likelihood to recommend. *AssignedRoomTypeK,P,B* are negatively contributing towards ADR.

```
Call:
lm(formula = ADR ~ Meal, data = city_hotel)

Residuals:
   Min     1Q Median     3Q    Max
-120.0  -25.5   -5.7   21.2 5295.2

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 104.7973     0.1743 601.222  < 2e-16 ***
MealFB      -28.3996     6.5397  -4.343 1.41e-05 ***
MealHB       15.2069     0.5694  26.707  < 2e-16 ***
MealSC       -5.7167     0.4597 -12.436  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 43.36 on 78726 degrees of freedom
Multiple R-squared:  0.01226,   Adjusted R-squared:  0.01223
F-statistic: 325.8 on 3 and 78726 DF,  p-value: < 2.2e-16
```

The above model shows that for the *H2-City* dataset the type of *Meal* accounts for about 1% of the variability in likelihood to recommend. This tells us that *Meal* Is not a significant factor to predict the ADR. Meal FB and SC are negatively contributing towards the ADR.

```
Call:
lm(formula = ADR ~ TotalPeople, data = city_hotel)

Residuals:
   Min     1Q Median     3Q    Max
-292.0  -21.9   -3.7   19.3 5293.3

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  52.7559     0.4598   114.7   <2e-16 ***
TotalPeople  26.9717     0.2245   120.2   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 40.11 on 78728 degrees of freedom
Multiple R-squared:  0.155,     Adjusted R-squared:  0.1549
F-statistic: 1.444e+04 on 1 and 78728 DF,  p-value: < 2.2e-16
```

The above model shows that for the *H2-City* dataset the TotalPeople accounts for about 15% of the variability in likelihood to recommend.

```
Call:
lm(formula = ADR ~ ReservedRoomType, data = city_hotel)

Residuals:
   Min     1Q Median     3Q    Max
-201.5  -21.2   -1.7   18.8 5303.8

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        96.2139     0.1537 626.127  < 2e-16 ***
ReservedRoomTypeB  -5.7272     1.1611  -4.933 8.13e-07 ***
ReservedRoomTypeC -10.7317    10.2397  -1.048 0.294619
ReservedRoomTypeD  35.2140     0.3866  91.080  < 2e-16 ***
ReservedRoomTypeE  60.3529     0.9895  60.994  < 2e-16 ***
ReservedRoomTypeF  92.9921     0.9207 101.004  < 2e-16 ***
ReservedRoomTypeG 105.2425     1.7608  59.770  < 2e-16 ***
ReservedRoomTypeP -96.2139    27.0891  -3.552 0.000383 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 38.31 on 78722 degrees of freedom
Multiple R-squared:  0.2292,    Adjusted R-squared:  0.2291
F-statistic:  3343 on 7 and 78722 DF,  p-value: < 2.2e-16
```

The above model shows that for the *H2-City* dataset the ReservedRoomType of the variability in likelihood to recommend. Reserved Room B,C,P are negatively impacting the ADR. It accounts for 22% of the variability in likelihood to recommend.

```
Call:
lm(formula = ADR ~ ReservedRoomType + TotalPeople + Meal, data = city_hotel)

Residuals:
   Min     1Q Median     3Q    Max
-226.0  -19.9   -2.3   18.1 5303.3

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        67.1619     0.4650 144.438  < 2e-16 ***
ReservedRoomTypeB  -8.6568     1.1248  -7.697 1.41e-14 ***
ReservedRoomTypeC  -8.0064     9.8920  -0.809 0.418302
ReservedRoomTypeD  30.5560     0.3902  78.302  < 2e-16 ***
ReservedRoomTypeE  52.8784     0.9611  55.021  < 2e-16 ***
ReservedRoomTypeF  67.0226     0.9911  67.626  < 2e-16 ***
ReservedRoomTypeG  84.9095     1.7295  49.095  < 2e-16 ***
ReservedRoomTypeP -70.8812    11.7170  -6.049 1.46e-09 ***
TotalPeople        14.7803     0.2396  61.679  < 2e-16 ***
MealFB            -19.9413     5.5874  -3.569 0.000359 ***
MealHB             17.4677     0.4871  35.864  < 2e-16 ***
MealSC              3.7193     0.3973   9.361  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 37.01 on 79314 degrees of freedom
Multiple R-squared:  0.2796,    Adjusted R-squared:  0.2795
F-statistic:  2799 on 11 and 79314 DF,  p-value: < 2.2e-16
```

The above model takes ReservedRoomType, TotalPeople, Meal for determining the ADR and by looking at the Adjusted R-squared we can say that the variables in above models accounts for 27.95% of the variability in likelihood to recommend. When multiple predictors are added, the adjusted R-squared value increased, which implies more data variability is justified.

```
Call:
lm(formula = ADR ~ LeadTime + DepositFactor + ReservedRoomType +
    Meal + VisitorType + Season, data = city_hotel)

Residuals:
   Min     1Q Median     3Q    Max
-217.0  -17.8   -1.7   17.9 5305.2

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)        8.889e+01  5.641e-01 157.572  < 2e-16 ***
LeadTime          -5.083e-02  1.355e-03 -37.519  < 2e-16 ***
DepositFactor      7.687e+00  4.125e-01  18.633  < 2e-16 ***
ReservedRoomTypeB -5.818e+00  1.217e+00  -4.781 1.75e-06 ***
ReservedRoomTypeC -1.083e+01  9.703e+00  -1.116    0.264
ReservedRoomTypeD  2.668e+01  4.017e-01  66.421  < 2e-16 ***
ReservedRoomTypeE  4.733e+01  9.590e-01  49.352  < 2e-16 ***
ReservedRoomTypeF  6.899e+01  9.545e-01  72.281  < 2e-16 ***
ReservedRoomTypeG  8.473e+01  1.701e+00  49.808  < 2e-16 ***
ReservedRoomTypeP -1.338e+02  2.567e+01  -5.212 1.88e-07 ***
MealFB            -2.276e+01  5.483e+00  -4.152 3.30e-05 ***
MealHB             1.878e+01  4.853e-01  38.690  < 2e-16 ***
MealSC             2.744e+00  4.082e-01   6.721 1.81e-11 ***
VisitorTypeFamily  2.603e+01  4.585e-01  56.773  < 2e-16 ***
VisitorTypeSingle -6.442e+00  3.426e-01 -18.805  < 2e-16 ***
SeasonSpring       7.040e+00  3.726e-01  18.892  < 2e-16 ***
SeasonSummer       8.450e+00  3.685e-01  22.930  < 2e-16 ***
SeasonWinter      -7.653e+00  3.799e-01 -20.146  < 2e-16 ***

Residual standard error: 36.3 on 78507 degrees of freedom
Multiple R-squared:  0.3083,    Adjusted R-squared:  0.3081
F-statistic:  2058 on 17 and 78507 DF,  p-value: < 2.2e-16
```

The above model shows that the variables account for 30.81% of the variability in likelihood to recommend. Every variable is a significant variable except the Roomtype C in the model.
ADR with respect to seasons:
For *H1-Resort* Dataset:

```
Call:
lm(formula = ADR ~ Season, data = resort_hotel)

Residuals:
    Min      1Q  Median      3Q     Max
-156.66  -24.70   -5.64   19.34  351.34

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   69.0361     0.4716 146.388  < 2e-16 ***
SeasonSpring   2.6648     0.6443   4.136 3.54e-05 ***
SeasonSummer  87.6217     0.6198 141.370  < 2e-16 ***
SeasonWinter -10.8363     0.6908 -15.687  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 44.99 on 40056 degrees of freedom
Multiple R-squared:  0.4639,    Adjusted R-squared:  0.4639
F-statistic: 1.156e+04 on 3 and 40056 DF,  p-value: < 2.2e-16
```

The variables in the above model accounts for 46.39% of the variability in likelihood to recommend. The above model shows that Winter season for the resorts is not a good time and as the coefficient of the Season Winter is negative and therefore it affects the ADR negatively.

For *H2-City* Dataset:

```
Call:
lm(formula = ADR ~ Season, data = city_hotel)

Coefficients:
 (Intercept)  SeasonSpring  SeasonSummer  SeasonWinter
      99.353        12.664        14.493        -5.006

> summary(lm_season_city)

Call:
lm(formula = ADR ~ Season, data = city_hotel)

Residuals:
   Min     1Q Median     3Q    Max
-113.8  -24.3   -5.0   19.2 5305.7

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   99.3533     0.3077  322.94   <2e-16 ***
SeasonSpring  12.6640     0.4318   29.32   <2e-16 ***
SeasonSummer  14.4931     0.4240   34.18   <2e-16 ***
SeasonWinter  -5.0058     0.4404  -11.37   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 42.82 on 79322 degrees of freedom
Multiple R-squared:  0.03563,   Adjusted R-squared:  0.0356
F-statistic:   977 on 3 and 79322 DF,  p-value: < 2.2e-16
```

The above model takes seasons and analysis how they affect the ADR. The predictor used is not a good predictor for predicting ADR because the variables account for only 3% of the variability in likelihood to recommend. This is a stark difference between the Resort and the hotels in cities. **This means that seasons impact the ADR for resorts much more than in the case with city hotels. This might because of the type of customers they have. City hotels could be used for business stays and other occasions and that why season matters less there. Whereas resorts are for vacationing and that’s why seasons matter.**

[5] Association Rules

Association Rule Mining Analysis was done to understand and find the frequent items in the dataset i.e the items that occur together frequently. With this analysis we tried to find if *IsCanceled* was dependent on other variables. We used the *IsCanceled* variable on the right hand side to find out the that what variables occurred frequently when the booking was cancelled. What were the reasons as to why the bookings were being canceled? Taking the *IsCanceled* variable for analysis makes sense because this variable can enable us to understand the factors that made the customers cancel the booking. We are using the apriori algorithm for the analysis.

For finding the initial rules we used the confidence of 0.5 and a support of 0.05. After that we got a lot of rules for both the datasets and we decided to pick the best rules out of the ones generated. We chose rules with confidence greater than 70% and high frequency.

For the left hand side we created a new dataset city_sample which included variables such as Deposit Type, Meal, Customer Type, Season, Reserved Room, Assigned Room.

For H1-*Resort* there were 175 rules that we found where cancelling could be linked with other factors. Some of those are:

Bookings are likely to get cancelled when …

Season type is Winter, Customer Type= transient, Country = Portugal, ReservedRoomType= A. This means that in winters the transient type customers who are coming from Portugal and reserve A type of room are more likely to cancel.

When Country is Portugal, season is winter. This means that in winters the customers coming from Portugal are more likely to cancel.

When meal is BB, season = winter, ReservedRoomType=A. This means that in room type A with meal plan of BB is more likely to get cancelled in winters.

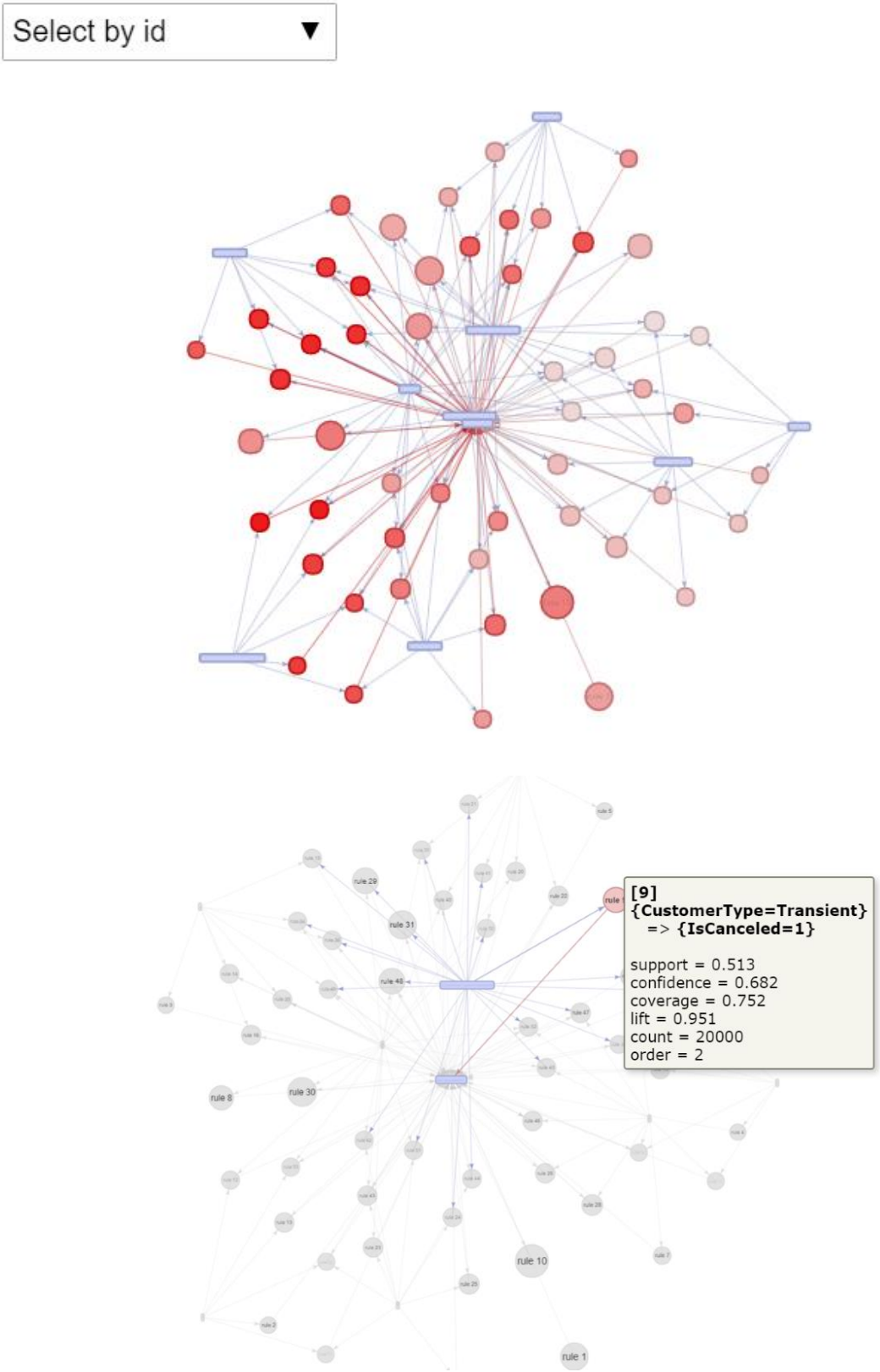

Figure 38 Graph for Association Rules of Resort Data

For *H2-City* there 116 rules that we found where cancelling could be linked with other factors. Some of those are:

Bookings are likely to get canceled …

When the ReservedRoomType is A and the Country is DEU. This shows that when the reserved room is A and country of origin is DEU the cancellations are higher.

When the deposit type= No deposit. This shows that when there is no deposit required for booking, people tend to cancel the bookings more.

When the ReservedRoomType is A, Meal is BB and deposit type is no deposit. Through fall, spring and summer people tend to cancel their reservation. In summer and spring the same category of people are more likely to cancel.

When the customer type= transient type and season is summer. The transient people cancel more bookings in the summer.

When Deposit type = No deposit, Meal type= BB, customer type=tranisent party and Reserved RoomType is A. This rule tells that when there is no deposit , the meal type is BB and the customers are transient and the reserved room type is A then more cancellations are likely to happen.

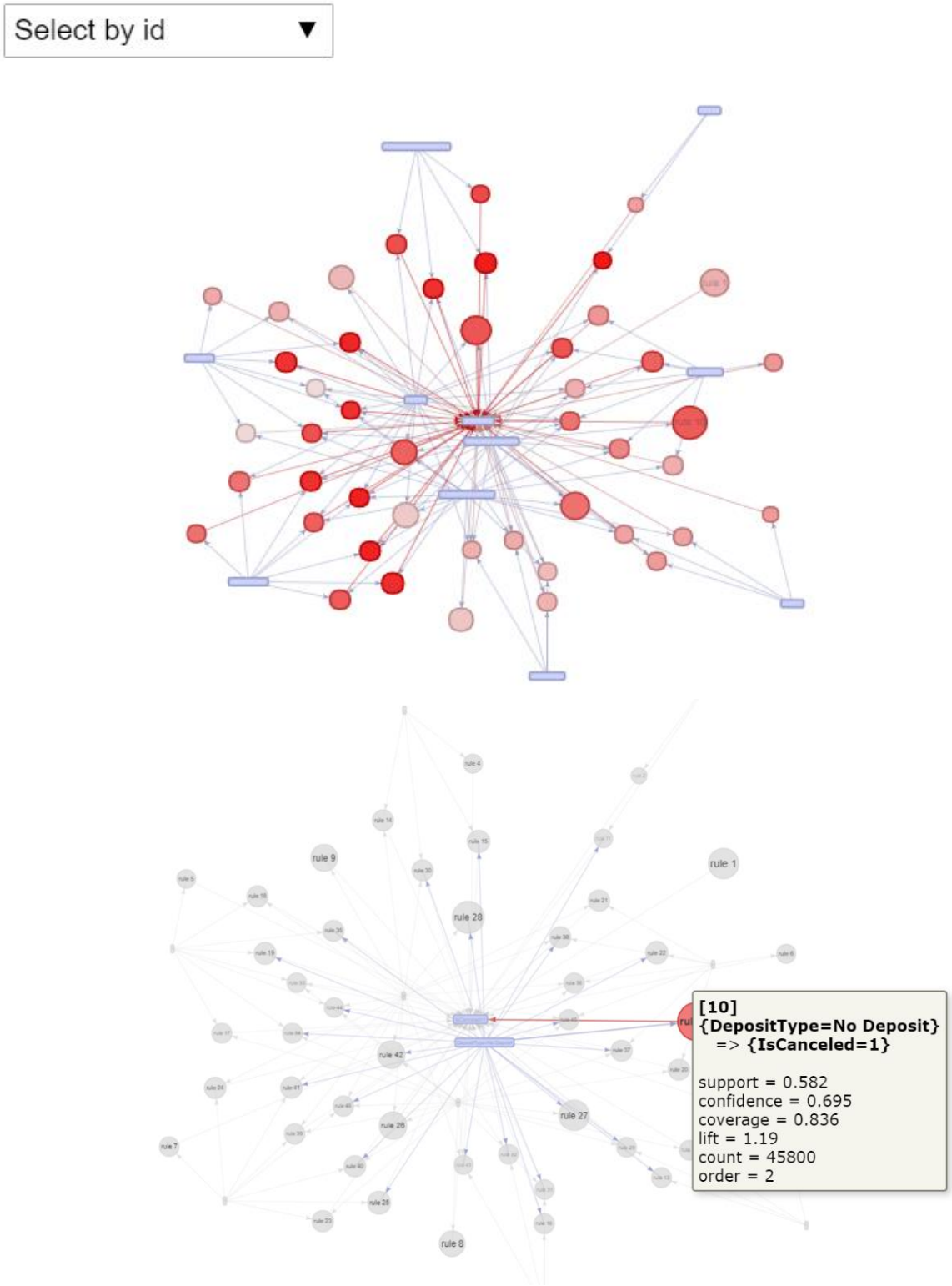

Figure 39 Graph for Association Rules of City Data

[6] Support Vector Models

For the support vector models we divided our data into training and test data sets. We divided the data in such a way that 40% was used for training and 60% for testing. Through stepwise analysis and trial and error we ended up using few attributes for the SVM classification, top-down approach was used where variables were excluded and accuracy was checked. We are using the *IsCanceled* data for analysis here.
We used the radial kernel for the models. Kpar = automatic i.e. all the parameters that are needed by the kernel function are automatically decided by the kernel function.

```
> svmOut_resort <- ksvm(IsCanceled ~ PreviousCancellations + LeadTime +Meal + DaysInWaitingList+ Season
 +ReservedRoomType+ MarketSegment,data = hotel_train_set_resort,kernel ="rbfdot", kpar="automatic" , C=
5, cross= 3,prob.model= T)
> svmOut_resort
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 5

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.263407702900038

Number of Support Vectors : 24015

Objective Function Value : -114770
Training error : 0.232702
Cross validation error : 0.238879
Probability model included.
> confusionMatrix(svm_pred_resort,hotel_test_set_resort$IsCanceled)
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 16469  5649
         1  1806  7485

               Accuracy : 0.7626
                 95% CI : (0.7579, 0.7673)
    No Information Rate : 0.5818
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4913

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

            Sensitivity : 0.9012
            Specificity : 0.5699
         Pos Pred Value : 0.7446
         Neg Pred Value : 0.8056
             Prevalence : 0.5818
         Detection Rate : 0.5243
   Detection Prevalence : 0.7042
      Balanced Accuracy : 0.7355
```

For the H1-Resort Dataset the predictors used were PreviousCancellations, LeadTime ,Meal ,DaysInWaitingList, Season ,ReservedRoomType and MarketSegment. The training error comes out to be 23.27%. The accuracy of the model is 76.26%. The cross-validation error is 23.88%. The accuracy tells us that the model could be used for classifications.

```
> confusionMatrix(svm_pred_city,hotel_test_set_city$IsCanceled)
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 16385  5613
         1  1890  7521

               Accuracy : 0.7611
                 95% CI : (0.7564, 0.7658)
    No Information Rate : 0.5818
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4887

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

            Sensitivity : 0.8966
            Specificity : 0.5726
         Pos Pred Value : 0.7448
         Neg Pred Value : 0.7992
             Prevalence : 0.5818
         Detection Rate : 0.5217
   Detection Prevalence : 0.7004
      Balanced Accuracy : 0.7346

       'Positive' Class : 0
```

For the H2-City Dataset the predictors used were PreviousCancellations, LeadTime, Meal, DaysInWaitingList, Season, ReservedRoomType and MarketSegment. The accuracy comes out to be 76.11%. The training error comes out to be 23.09%. The cross-validation error comes out to be 23.51%. By the accuracy we can say that the model fits our data and can be used to classify.

[7] Anomalies & Further Questions

Throughout our entire process of probing into this case study from European Hotels Group, we kept checking for validation, effectiveness and verification.

We wanted to be aware of an important factor that we noticed from the dataset that could change all the inferences we made. We noticed that there was data missing from certain time periods.
For example, in the Hotel dataset which had data from 2015. 2016 and 2017, only the year 2016 had data for all the months from January to December. This meant that there may or may not have been more transactions and these unknown transactions could change our entire analysis.
If we take the situation of our seasonality analysis, December was appearing to be month with a lower number of customers for both the datasets. When we looked into why this is the case and saw that the year 2017 did not contain any observations for December.

The question of whether these missing pieces of information could skew the data or trends is unknown and can be further looked into.