Customer Segmentation - A private airline

Venkatachalapathy Othisamy

10/18/2016

Introduction

This private airline is a unique player in the airline carrier industry has endured the threats of intense competition from large national brands. Started as a charter carrier, it has expanded its business to offer scheduled flight services to various destinations. By 2014, this airline had survived bankruptcies, multiple economic recessions, threats of mergers and was now stable and profitable.

This airline has data representing 1.52 million customers making 1.86 million trips between January 2013 and December 2014.

Understanding the data

The original this airline data is in .csv file format. We will use R to preprocess the data and analyze it. The data dictionary is shown below.

Field	Description
PNRLocatorID	PNR #. This could,be treated as a confirmation number. Multiple flights and segments all roll,up to one PNR #.
TicketNum	Ticket Number - An itinerary may have multiple ticket numbers.
CouponSeqNbr	Sabre assigned,sequence number of the ticket coupon. Values are 1-16; values 5 and greater,are a result of a conjunctive ticket.
ServiceStartCity	Airport code for,the city the flight is leaving from.
ServiceEndCity	Airport code for,the city the flight is landing at.
PNRCreateDate	Date the ticket was,booked
ServiceStartDate	When the flight,takes off
PaxName	First 4 digits of,the passenger last name and the first 2 digits of the passenger 1st name
EncryptedName	Passenger name in,encrypted format.
GenderCode	M for Male and F,for Female
Birthdateid	A unique identifier,that allows you to match birthdates across records without actually providing,the birth date.
Age	Age of the,passenger at the time of the flight.
PostalCode	Postal code of the,location from where the booking was made.
BkdClassOfService	What class of,service (e.g. coach, first class, etc.) the passenger booked
TrvldClassOfService	What class of,service (e.g. coach, first class, etc.) the passenger

Field	Description
	travelled. This could,be upgraded later in the flight.
BookingChannel	How the passenger,booked the flight. If this is showing a 3 letter code, it's most likely booked, at that airport. UFO is booked in Cancun.
BaseFareAmt	Amount of the base,fare (without taxes) of all segments on the ticket. Under certain,circumstances such as bulk, net, award tickets, this value will be blank.
TotalDocAmt	Total price of this,ticket document including base fare, taxes and fees stated in the,EquivBaseFareCurrCode. In the case of exchanges this amount may be zero or,may only represent the difference in price from the previous ticket amount
UflyRewardsNumber	The rewards number,that was provided when booked.
UflyMemberStatus	The Ufly member,status. It will be either Standard or Elite.
CardHolder	True or False value,if the member is also a credit card holder.
BookedProduct	Free form text,field that is used to put discount codes in
EnrollDate	When the member,enrolled in Ufly rewards
MarketingFlightNbr	Flight Number
MarketingAirlineCode	The Airlines with,which booking was made. We are only interested in "SY" which is,the code for the airline
StopoverCode	O' for Layovers ie,halt in a city for less than 24 hours.'X' for Stopovers that is halt in a,city for more than 24 hours.

Code to open the `.csv` data R

```
folder="C:/Users/venka/Documents/Fall Semester/Exploratory Data Analysis/airline_data"
setwd(folder)

data<-read.csv("airline_data.csv") #Data is stored in the dataframe called data

# data_orig <- data
# sample_rows <- sample.int(nrow(data),100000)
# data <- data[sample_rows,]
# write.csv(data,"data_small.csv")
```

The structure of the data is shown below

```
str(data)
```

```
## 'data.frame':    3435388 obs. of  26 variables:
##  $ PNRLocatorID        : Factor w/ 1179997 levels "AAAACD","AAAAMU",..: 5 5 7 7
8 10 13 13 18 19 ...
##  $ TicketNum           : num  3.38e+12 3.38e+12 3.37e+12 3.37e+12 3.37e+12 ...
##  $ CouponSeqNbr        : int  2 1 2 1 1 1 1 1 1 1 ...
##  $ ServiceStartCity    : Factor w/ 73 levels "ANC","AZA","BOS",..: 28 44 44 61 3
9 51 28 28 39 56 ...
##  $ ServiceEndCity      : Factor w/ 80 levels "ABE","ANC","AUS",..: 52 35 70 52 5
2 52 52 52 52 52 ...
##  $ PNRCreateDate       : Factor w/ 1146 levels "2011-06-22","2011-06-30",..: 742
742 815 815 852 329 1074 1074 935 787 ...
##  $ ServiceStartDate    : Factor w/ 730 levels "2013-01-01","2013-01-02",..: 347
342 419 416 478 43 683 683 521 406 ...
##  $ PaxName             : Factor w/ 506327 levels "A ALZI","AABEAL",..: 55636 55
636 118338 118338 415872 189535 154608 154594 267759 418857 ...
##  $ EncryptedName       : Factor w/ 1448740 levels "4120414C52484D414E44696420493
F7C20676574207468697320726967687 45A494144",..: 158061 158061 330656 330656 1196791
541391 424667 424652 758612 1203806 ...
##  $ GenderCode          : Factor w/ 4 levels "","F","M","U": 2 2 3 3 2 3 2 3 3 3
...
##  $ birthdateid         : int  35331 35331 46161 46161 34377 39505 50874 34741 4
1690 38575 ...
##  $ Age                 : int  66 66 37 37 69 54 25 69 49 58 ...
##  $ PostalCode          : Factor w/ 8132 levels "","00000","00022",..: 1 1 1 1 1
1 1 1 1 3840 ...
##  $ BkdClassOfService   : Factor w/ 3 levels "Coach","Discount First Class",..: 1
1 1 1 1 1 1 1 1 ...
##  $ TrvldClassOfService : Factor w/ 3 levels "Coach","Discount First Class",..: 1
3 2 2 1 1 1 1 1 2 ...
##  $ BookingChannel      : Factor w/ 29 levels "ANC  ","BOS  ",..: 18 18 23 23 21
23 23 23 21 23 ...
##  $ BaseFareAmt         : num  234 234 294 294 113 ...
##  $ TotalDocAmt         : num  0 0 338 338 132 ...
##  $ UFlyRewardsNumber   : int  NA NA NA NA NA NA NA NA NA 202369882 ...
##  $ UflyMemberStatus    : Factor w/ 3 levels "","Elite","Standard": 1 1 1 1 1 1
1 1 1 3 ...
##  $ CardHolder          : Factor w/ 3 levels "","false","true": 1 1 1 1 1 1 1 1
1 2 ...
##  $ BookedProduct       : Factor w/ 396 levels "","03RYPR","04NY4R",..: 50 50 1 1
1 1 289 289 258 1 ...
##  $ EnrollDate          : Factor w/ 192106 levels "","2007-07-17 00:00:00.000000
0",..: 1 1 1 1 1 1 1 1 1 54964 ...
##  $ MarketingFlightNbr  : Factor w/ 819 levels "0001","0003",..: 452 451 565 560
523 689 452 452 523 553 ...
##  $ MarketingAirlineCode: Factor w/ 5 levels "DE","F9","FI",..: 5 5 5 5 5 5 5 5
5 5 ...
##  $ StopoverCode        : Factor w/ 3 levels "","O","X": 2 1 2 1 1 1 1 1 1 1 ...
```

Summarizing it

```
summary(data)
```

```
##   PNRLocatorID       TicketNum          CouponSeqNbr   ServiceStartCity
##  OPKWDG :    558   Min.   :3.372e+12   Min.   :1.000   MSP    :1612636
##  DILBDW :    484   1st Qu.:3.372e+12   1st Qu.:1.000   LAS    : 176026
```

```
##  KOUWAC :    466   Median :3.372e+12   Median :1.000   JFK    : 139477
##  OHYPCP :    458   Mean   :3.374e+12   Mean   :1.464   MCO    : 135790
##  EJYFAM :    416   3rd Qu.:3.377e+12   3rd Qu.:2.000   LAX    : 131661
##  MQYFIJ :    406   Max.   :3.380e+12   Max.   :8.000   SFO    : 130449
##  (Other):3432600                                       (Other):1109349
##  ServiceEndCity       PNRCreateDate       ServiceStartDate
##  MSP    :1605351   2013-12-03:  15526   2014-11-30:   8919
##  LAS    : 175819   2013-12-02:  14914   2014-03-16:   8885
##  MCO    : 137147   2014-01-06:  11864   2014-02-16:   8855
##  JFK    : 137084   2012-11-26:  11209   2014-01-05:   8824
##  LAX    : 131552   2013-12-04:  11143   2014-12-28:   8814
##  SFO    : 128943   2014-09-09:  10751   2013-12-29:   8703
##  (Other):1119492   (Other)   :3359981   (Other)   :3382388
##     PaxName
##  JOHNMA :   2336
##  JOHNJA :   1590
##  JOHNBR :   1440
##  ANDEMA :   1394
##  JOHNJE :   1361
##  JOHNDA :   1306
##  (Other):3425961
##                                                                      Encr
yptedName
##  4A4F484E534F4E44696420493F7C2067657420746869732072696768744D49434841454C
:    132
##  4A4F484E534F4E44696420493F7C2067657420746869732072696768744348524953544F5048455
2:    113
##  42555244455454454696420493F7C2067657420746869732072696768744524F42455254
:    109
##  455249434B534F4E44696420493F7C2067657420746869732072696768744B494D4245524C59
:    105
##  4A4F5244414E44696420493F7C2067657420746869732072696768744B454C4C59
:    105
##  414E444552534F4E44696420493F7C2067657420746869732072696768744A414D4553
:    102
##  (Other)
:3434722
##  GenderCode   birthdateid           Age              PostalCode
##    :  43999   Min.   :-675290   Min.   :-2883.00          :2744995
##  F:1770581   1st Qu.:  39620   1st Qu.:   26.00   55416  :   8822
##  M:1620768   Median :  45088   Median :   40.00   55347  :   8803
##  U:     40   Mean   :  44982   Mean   :   40.05   55311  :   7888
##              3rd Qu.:  50251   3rd Qu.:   55.00   55044  :   7856
##              Max.   :1112840   Max.   : 2012.00   55124  :   7059
##              NA's   :43999     NA's   :43999      (Other): 649965
##             BkdClassOfService             TrvldClassOfService
##  Coach               :3341587   Coach               :3179137
##  Discount First Class:   1166   Discount First Class:  88686
##  First Class         :  92635   First Class         : 167565
##
##
##
##
##                BookingChannel     BaseFareAmt      TotalDocAmt
##  Outside Booking     :1476199   Min.   :   0.0   Min.   :    0.0
##  SCA Website Booking :1457143   1st Qu.: 174.9   1st Qu.:  189.8
##  Reservations Booking: 262329   Median : 273.2   Median :  302.2
```

```
##  Tour Operator Portal: 133201   Mean   : 287.7   Mean   :  314.9
##  SY Vacation         :  94601   3rd Qu.: 370.2   3rd Qu.:  414.6
##  MSP                 :   4969   Max.   :4342.0   Max.   :17572.0
##  (Other)             :   6946
##  UFlyRewardsNumber   UflyMemberStatus   CardHolder      BookedProduct
##  Min.   :100000191          :2740908        :2740908          :2209904
##  1st Qu.:200860074   Elite   :  14361   false: 659060   SSWMIR : 390853
##  Median :202968124   Standard: 680119   true :  35420   BSGTIX : 137082
##  Mean   :204242059                                      GRP    : 129515
##  3rd Qu.:210381861                                      EXP    :  99627
##  Max.   :241086274                                      UP     :  78106
##  NA's   :2740908                                        (Other): 390301
##                        EnrollDate      MarketingFlightNbr
##                             :2740908   341    :  94238
##  2007-07-17 00:00:00.0000000:  21348   342    :  88055
##  2007-12-02 15:27:19.0000000:    763   504    :  76411
##  2007-12-02 15:06:38.0000000:    721   503    :  73575
##  2013-01-23 15:53:24.0000000:    285   244    :  72449
##  2011-03-03 06:31:17.0000000:    145   243    :  67803
##  (Other)                     : 671218   (Other):2962857
##  MarketingAirlineCode StopoverCode
##  DE:      1            :1720454
##  F9:   3319           O:1601636
##  FI:     13           X: 113298
##  HA:   1705
##  SY:3430350
##
##
```

Data Preperation

Data Cleaning

The following are the attributes that need treatment.

GenderCode and Birthdateid	Removing rows with faulty Gendercode and BirthdateID
Age	Replacing faulty values with median value
UflyRewardsNumber	Replacing NAs with 0
UflyMemberStatus	Replacing Missing values with “non-member”
Duplicate PNRs	Removing rows with duplicate PNRs
BookingChannel	Removing rows with city codes as BookingChannel
Marketing Airline Code	Removing rows with airline code other than “SY”
Error PNRs	Removing error PNRs

We need to remove rows with faulty Gendercode and BirthdateID

```
#Filtering out records which have NA for BirthdateID
#same as data <- data %>%filter(!is.na(birthdateid))
data%<>%filter(!is.na(birthdateid))

data$GenderCode<-as.character(data$GenderCode)
data%<>%filter(GenderCode!="")

#Filtering out records which have data for GenderCode
data$GenderCode<-as.factor(data$GenderCode)
```

Replacing faulty values in Age with median value

```
#Replacing negative ages with median value
data$Age[data$Age < 0] <- median(data$Age)

#Replacing age values greater than 120 with median value
data$Age[data$Age > 120] <- median(data$Age)
```

Replacing NAs in UflyRewardsNumber with 0

```
#Replacing NAs with 0
data$UFlyRewardsNumber[is.na(data$UFlyRewardsNumber)] <- 0
```

Replacing Missing values in UflyMemberStatus with non-member

```
#Convert factor level data to string
data$UflyMemberStatus<-as.character(data$UflyMemberStatus)

#Replacing missing values with non-ufly
data$UflyMemberStatus[data$UflyMemberStatus==''] <-"non-ufly"
```

Retaining only those rows which have single occurrence of PNRLocatorID, CouponSeqNbr, PaxName, ServiceStartCity, ServiceEndCity, ServiceStartDate combination.

```
data%<>%
  group_by(PNRLocatorID,CouponSeqNbr,PaxName,ServiceStartCity,ServiceEndCity,Servic
eStartDate)%>%
  filter(n()==1)
```

Removing rows with faulty city codes as BookingChannel. Some rows have city names for Booking Channel. Replacing faulty data with Other

```
data$BookingChannel<-as.character(data$BookingChannel)
data$BookingChannel[data$BookingChannel!="Outside Booking" &
                      data$BookingChannel!="SCA Website Booking" &
                      data$BookingChannel!="Tour Operator Portal" &
                      data$BookingChannel!="Reservations Booking" &
                      data$BookingChannel!="SY Vacation"] <- "Other"
data$BookingChannel<-as.factor(data$BookingChannel)
```

Removing rows with MarketingAirlineCode code other than SY, the airline code for the airline.

```
data$MarketingAirlineCode<-as.character(data$MarketingAirlineCode)
data%<>%filter(MarketingAirlineCode=="SY")
data$MarketingAirlineCode<-as.factor(data$MarketingAirlineCode)
```

Creating a new column called error which contains 1 if the PNR is errored or 0 otehrwise. Error PNR refers to those which do not start with coupon sequence number 1.

```
data%<>%group_by(PNRLocatorID)%>%
  mutate(error= ifelse(min(CouponSeqNbr)!=1,1,0))
```

Retaining only the non errored rows and check how many rows are remaining.

```
data%<>%filter(error==0)
nrow(data)
```

```
## [1] 3073826
```

Data Sampling

Since the data, after transformation, has 3.2 million rows, we take a sample of the data to perform further analysis to facilitate R to handle the data with ease. Since the data is at the level of one row per flight, just taking a random sample of the rows will distort the trip details. So, we take a sample of the PNRLocatorIDs and retain all the records belonging to the sampled PNRs.

```
#Obtain Unique PNRs
uniquePNRs<-unique(data$PNRLocatorID)

#To produce the same samples every time the code is run
set.seed(1234567)


sample_PNRs<-sample(uniquePNRs,10000)

#Obtaining data related to the sampled 10,000 PNRs
sample_data<-data%>%filter(PNRLocatorID %in% sample_PNRs)
```

Data Transformation

For the purpose of analysis, attributes are created as a combination of other attributes.

1	**UID**	**Unique ID for every customer**
2	Age Bucket	Bin customer age into 5 age buckets
3	True Origin	The starting city of every trip
4	Final destination	The ending city of the trip
5	True Destination	The actual destination of the trip (City of longest stay)
6	Oneway-RoundTrip	1 if the trip was a round trip and 0 if one way
7	Group Size	Size of the group if the trip constituted of more than one passengers.

8	Group-Single	1 if the trip was flown by more than 1 customers and 0 if the trip was flown by a single customer.
1	**UID**	**Unique ID for every customer**
9	Seasonality	Q1 if travel was made in Jan-Mar Q2 if travel was made in Apr-June Q2 if travel was made in July-Sept Q2 if travel was made in Oct-Dec
10	Days Booked in Advance	Number of days between booking and travel

Creating a Unique ID for each customer by concatenating Encrypted name, GenderCode and birthdateid.

```
sample_data<-sample_data%>% mutate(uid=paste(EncryptedName,GenderCode,birthdateid,s
ep=""))
```

Binning the customers' age into 1 of 5 age buckets

```
sample_data%<>%mutate(age_group =
                        ifelse(Age>=0 & Age<18,"0-17",
                               ifelse(Age>=18 & Age < 25,"18-24",
                                      ifelse(Age>=25&Age<35,"25-34",
                                             ifelse(Age>=35 & Age<55,"35-54",
                                                    ifelse(Age>=55,"55+",0)
                                                    )
                                             )
                                      )
                               )
                      )
```

Determining the true Service Start City for each row in the data. It will be the First city from which the trip started

```
true_origins<-sample_data%>%
  arrange(PNRLocatorID,CouponSeqNbr)%>%
  group_by(PNRLocatorID,PaxName)%>%
  do(data.frame(true_origin=first(.$ServiceStartCity)))

sample_data<-merge(sample_data,true_origins,
                   by.x=c("PNRLocatorID","PaxName"),
                   by.y = c("PNRLocatorID","PaxName"))
```

Determining where the trip ended. If the trip is a round trip, the service end city (Final Destination) will be the same as the service start city (True Origin)

```
final_destination<-sample_data%>%
  arrange(PNRLocatorID,CouponSeqNbr)%>%
  group_by(PNRLocatorID,PaxName)%>%
  do(data.frame(final_destination=last(.$ServiceEndCity)))

sample_data<-merge(sample_data,final_destination,
                   by.x=c("PNRLocatorID","PaxName"),
                   by.y = c("PNRLocatorID","PaxName"))
```

Determining what was the trips true destination. We assume this was the place where most time was spent on the trip.

```
#Convert Service Start date to Date type
sample_data$ServiceStartDate<-as.Date(sample_data$ServiceStartDate)

#The place of maximum stay during the trip.
diff1<-sample_data%>%
  arrange(PNRLocatorID,CouponSeqNbr)%>%
  group_by(PNRLocatorID,PaxName)%>%
  mutate(stay=lead(ServiceStartDate)-ServiceStartDate,default=0)%>%
  select(PNRLocatorID,PaxName,ServiceStartCity,ServiceEndCity,ServiceStartDate,stay
)

diff1$stay[is.na(diff1$stay)]<-0
diff1$stay<-as.numeric(diff1$stay)

true_destination<-diff1%>%
  group_by(PNRLocatorID,PaxName)%>%
  do(data.frame(true_destination= first(as.character(.$ServiceEndCity)[.$stay==max
(.$stay)])))
```

```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character
```

```
sample_data<-merge(sample_data,true_destination,
                   by.x=c("PNRLocatorID","PaxName"),
                   by.y = c("PNRLocatorID","PaxName"))
```

Next, determining if the trip was a one-way or round-trip. The trip is considered a round trip if the service end city (Final Destination) will be the same as the service start city (True Origin).

```
sample_data%<>%
  mutate(round_trip = ifelse(as.character(true_origin)==as.character(final_destina
tion), 1, 0))
```

Next, determining the group size, the number of people who traveled together in each trip.

```
sample_data%<>%
  group_by(PNRLocatorID)%>%
  mutate(group_size= length(unique(uid)))
```

Next, we have a special inidcator if the group-size was 1,i.e., flown by a single customer

```
sample_data%<>%
  group_by(PNRLocatorID)%>%
  mutate(group= ifelse(group_size>1,1,0))
```

Handling seasonality in terms of quaters. Assign Q1 to Q4 based on the quarter of the year in which the trip was made

```
sample_data$ServiceStartDate<-as.Date(sample_data$ServiceStartDate)
#Convert ServiceStartDate from factor to Date format
sample_data%<>%
  group_by(PNRLocatorID,PaxName)%>%
  mutate(seasonality= ifelse(month(ServiceStartDate)>=1 & month(ServiceStartDate)<
=3,"Q1",
                             ifelse(month(ServiceStartDate)>=4 & month(ServiceStar
tDate)<=6,"Q2",
                                    ifelse(month(ServiceStartDate)>=7 & month(Serv
iceStartDate)<=9,"Q3",
                                           ifelse(month(ServiceStartDate)>=10 & m
onth(ServiceStartDate)<=12,"Q4",0)
                                           )
                                    )
                             )
         )
```

Finally, calculating the number of days the ticket was booked in advance. It is the difference between PNRCreateDate and ServiceStartDate

```
sample_data$PNRCreateDate <- as.Date(sample_data$PNRCreateDate)
sample_data$ServiceStartDate <- as.Date(sample_data$ServiceStartDate)
sample_data%<>%
  mutate(days_pre_booked=as.numeric(floor( difftime(ServiceStartDate,
                                                    PNRCreateDate,units=c("days")
))))
```

Customer Segmentation

We want to use the data to segment customers of the airline into general categories of people with similar flying patterns. The goal is to group the observations in the data into clusters such that every datum in a cluster is more similar to other datums in the same cluster than it is to datums in other clusters.

Changing data granularity

In order to run the segmentation algorithm, we need to first have the data at the right granularity. Since we are looking to segment customers, it is important to bring the data to the granularity of customers. Hence, transforming the data such that each row represents a unique customer-PNR combination.

```
sample_data%<>%
  select(PNRLocatorID, uid, PaxName, ServiceStartDate, BookingChannel, TotalDocAmt
,
         UFlyRewardsNumber,UflyMemberStatus, age_group,true_origin,true_destinatio
n,
         round_trip,group_size,group, seasonality,days_pre_booked)

customer_data <- sample_data %>%
  group_by(PNRLocatorID,uid,PaxName) %>%
  summarise(ServiceStartDate=first(ServiceStartDate),
            BookingChannel=first(BookingChannel),
            avg_amt=max(TotalDocAmt),
            UFlyRewards=first(UFlyRewardsNumber),
            UflyMemberStatus=first(UflyMemberStatus),
            age_group=last(age_group),
            true_origin=first(true_origin),
            true_destination=first(true_destination),
            round_trip=first(round_trip),
            group_size=first(group_size),
            group=first(group),
            seasonality=last(seasonality),
            days_pre_booked=max(days_pre_booked))

#Retaining only those attributes that are meaningful for clustering
customer_data%<>%
  select(-PNRLocatorID,-uid,-PaxName,-ServiceStartDate,-UFlyRewards)
```

```
## Adding missing grouping variables: `PNRLocatorID`, `uid`
```

```
  nrow(sample_data)
```

```
## [1] 26325
```

```
  #Granularity of data was reduced to customer level
  nrow(customer_data)
```

```
## [1] 15181
```

Units and Scaling

The initial understanding of the data has shown us that this contains attributes of different units. Units affect what clustering algorithms will discover. One way to try to make the clustering more coordinate-free is to transform all the columns to have a value between 0 and 1. This is called Normalization. There are multiple techniques of achieving normalization. We will be using the min-max normalization technique.

```
#Min-Max normalization: x= x-max/max-min
customer_data_t=read.csv("customer_data.csv")
customer_data=customer_data_t[,-1]
normalize <- function(x){return ((x - min(x))/(max(x) - min(x)))}

ungrouped <- ungroup(customer_data)

customer_data_km = mutate(ungrouped,
                      avg_amt = normalize(avg_amt),
                      days_pre_booked = normalize(days_pre_booked),
                      group_size=normalize(group_size))
```

```
## Warning: failed to assign NativeSymbolInfo for lhs since lhs is already
## defined in the 'lazyeval' namespace
```

```
## Warning: failed to assign NativeSymbolInfo for rhs since rhs is already
## defined in the 'lazyeval' namespace
```

```
write.csv(customer_data_km,"customer_data_km.csv")
```

Clustering algorithm

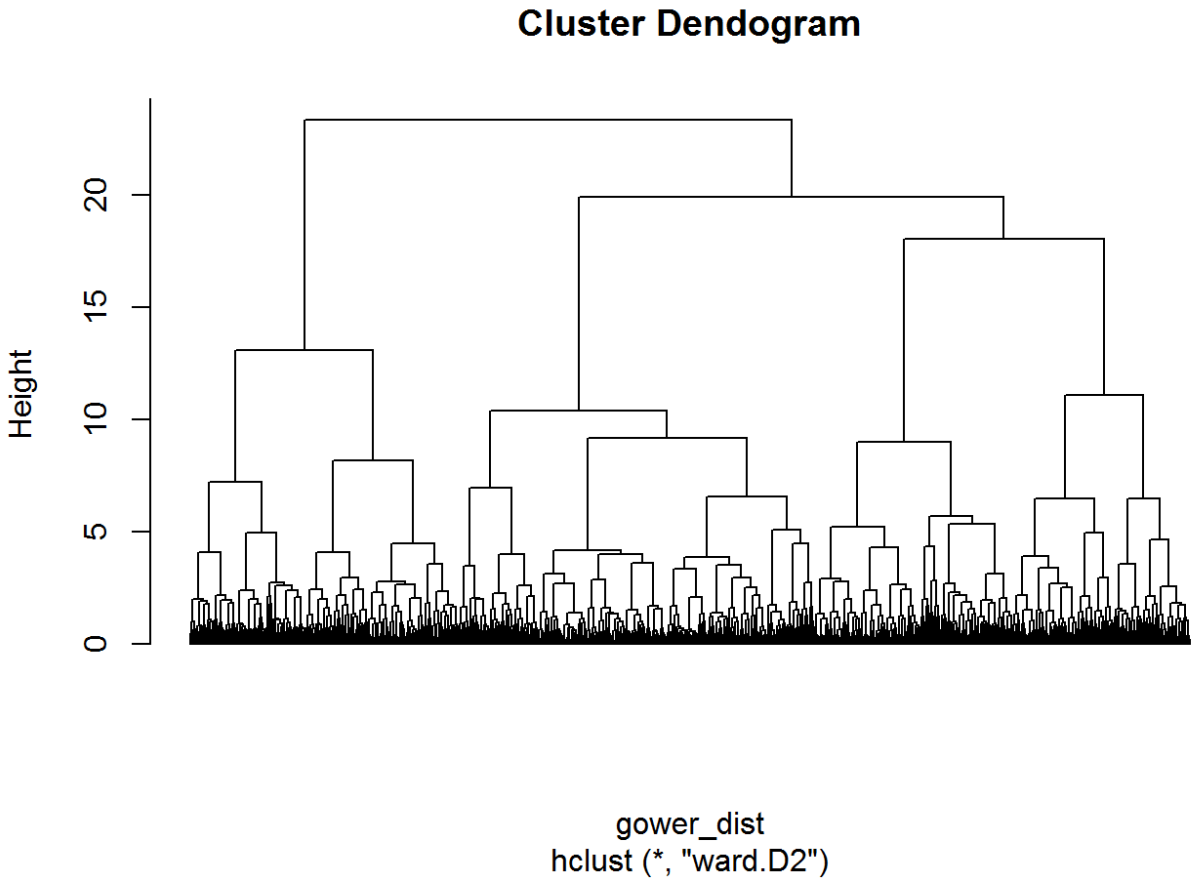
Various clustering algorithms can be used to achieve the goal of segmentation

```
customer_data_fc=data.frame(sapply(customer_data_km[,c("BookingChannel","UflyMe
mberStatus","age_group","true_origin","true_destination","seasonality")],as.fac
tor))
customer_data_tmp=data.frame(sapply(customer_data_fc,as.numeric))
customer_data_nm=cbind(customer_data_km[,c(4,9,10,11,13)],customer_data_tmp)
customer_data_norm=sapply(customer_data_nm[,-1],FUN = normalize)
```

```
#Calculating Gower distance

#Converting columns to Factor variables
customer_data_fc=data.frame(sapply(customer_data_km[,c("BookingChannel","UflyMe
mberStatus","age_group","true_origin","true_destination","seasonality","group",
"round_trip")],as.factor))

customer_data_nm=cbind(customer_data_km[,c(4,10,13)],customer_data_fc)
library(cluster)
memory.limit(size = 1000000)
```

```
## [1] 1e+06
```

```
#Performing hierarchial clustering
gower_dist=daisy(customer_data_nm, metric = "gower")
customer_hcl=hclust(gower_dist,method="ward.D2")

#plot
plot(customer_hcl,hang=0,label=F,main="Cluster Dendogram")
```




```
travel_groups = cutree(customer_hcl,5)
customer_num_agg = aggregate(customer_data_km[,c(4,10,13)],list(travel_groups),
median)

#Calculating mode
getmode <- function(x) {
     ux <- unique(x)
     ux[which.max(tabulate(match(x, ux)))]
 }
customer_data_ch= data.frame(sapply(customer_data_km[,c("BookingChannel","UflyM
emberStatus","age_group","true_origin","true_destination","seasonality","group"
,"round_trip")],as.character))

customer_cat_agg = aggregate(customer_data_ch,list(travel_groups),FUN=getmode)

travel_group_characteristcs=merge(customer_cat_agg,customer_num_agg)
travel_group_characteristcs
```

```
##   Group.1      BookingChannel UflyMemberStatus age_group true_origin
## 1       1 SCA Website Booking         Standard       55+         MSP
## 2       2     Outside Booking         non-ufly     35-54         MSP
## 3       3     Outside Booking         non-ufly     35-54         MSP
## 4       4     Outside Booking         non-ufly     35-54         MSP
## 5       5 SCA Website Booking         non-ufly     35-54         MSP
##   true_destination seasonality group round_trip   avg_amt group_size
## 1              MSP          Q4     1          1 0.1723060      0.125
## 2              LAS          Q1     1          1 0.1798907      0.125
## 3              MSP          Q3     0          0 0.1088492      0.000
## 4              MSP          Q4     0          1 0.1950601      0.000
## 5              MSP          Q3     1          0 0.1058385      0.125
##   days_pre_booked
## 1      0.09469697
## 2      0.10416667
## 3      0.03409091
## 4      0.07007576
## 5      0.06628788
```

```
sil_hclust=silhouette(travel_groups,gower_dist)
summary(sil_hclust)
```

```
## Silhouette of 15181 units in 5 clusters from silhouette.default(x = travel_g
roups, dist = gower_dist) :
##  Cluster sizes and average silhouette widths:
##       2693       5408       2320       3000       1760
## 0.04025151 0.20174107 0.22997532 0.13939941 0.18713157
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.3854  0.1188  0.1960  0.1634  0.2442  0.3137
```

```
#Performing PAM clustering
library(fpc)
```

```
##
## Attaching package: 'fpc'
```

```
## The following object is masked from 'package:dbscan':
##
##     dbscan
```

```
kmed<-pam(gower_dist,8)

sil_pam=silhouette(kmed,gower_dist)
summary(sil_pam)
```

```
## Silhouette of 15181 units in 8 clusters from pam(x = gower_dist, k = 8) :
##  Cluster sizes and average silhouette widths:
##       1418       3004       1559       1870       2315       1574
## 0.07438830 0.10732115 0.05612067 0.15778301 0.11812212 0.12312507
##       1696       1745
## 0.08096464 0.09564907
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.2113  0.0227  0.1040  0.1042  0.1830  0.3969
```

```
#Performing k-modes clustering
customer_data_modes=customer_data
#Creating bins to perform k-modes

get_amt_bins <- function(x) {
     if (x<164){'100-164'}
  else if (x>164 & x<278){'165-275'}
  else if (x>278 & x<389){'276-400'}
  else ('>400')
}

get_days_bins <- function(x) {
     if (x<20){'1-20'}
  else if (x>20 & x<50){'21-50'}
  else if (x>51 & x<80){'51-80'}
  else ('>80')
   }

customer_data_modes$avg_amt=t(data.frame(lapply(customer_data[,4],get_amt_bins)
))

customer_data_modes$days_pre_booked=t(data.frame(lapply(customer_data$days_pre_
booked,get_days_bins)))

customer_modes_factor=data.frame(sapply(customer_data_modes[,-c(1,2)],as.factor
))

customer_kmodes=kmodes(customer_modes_factor,8)

#Calculating Gower distance for the pruned dataset
gower_dist_modes=daisy(customer_modes_factor, metric = "gower")

sil_kmodes=silhouette(customer_kmodes$cluster,gower_dist_modes)
summary(sil_kmodes)
```

```
## Silhouette of 15181 units in 8 clusters from silhouette.default(x = customer
_kmodes$cluster, dist = gower_dist_modes) :
##  Cluster sizes and average silhouette widths:
##        2621        2194        2461        2384        1747        1362
##  0.01676744 -0.01894262 -0.01307225  0.03748312  0.18798753  0.14501531
##        1173        1239
##  0.10842952  0.07695442
## Individual silhouette widths:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.24760 -0.03347  0.05039  0.05323  0.13360  0.39010
```

```
kmed_binned<-pam(gower_dist_modes,8)

sil_pam_binned=silhouette(kmed_binned,gower_dist_modes)
summary(sil_pam_binned)
```

```
## Silhouette of 15181 units in 8 clusters from pam(x = gower_dist_modes, k = 8
) :
##  Cluster sizes and average silhouette widths:
##       1350       2446       2870       2273       1873       1312
## 0.09654652 0.07490533 0.02639310 0.07696787 0.10377795 0.14947673
##       1581       1476
## 0.04399394 0.11525762
## Individual silhouette widths:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.208100  0.003631  0.082610  0.078680  0.150500  0.421700
```

```
write.csv(customer_data_nm,"customer_data_nm.csv")
```

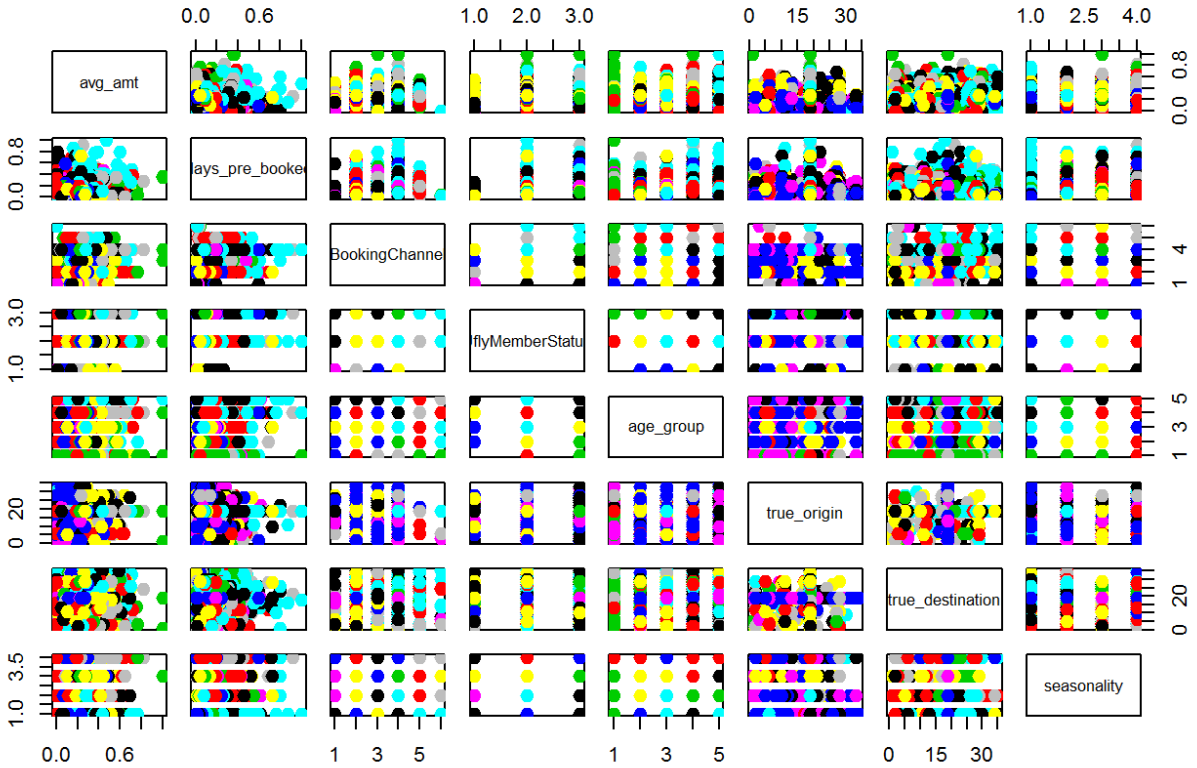
Visualizing the Clusters

Now that the data is divided into clusters, we can interpret the clusters by summarizing the data within each cluster. We can visualize differences between clusters in Ufly membership, one way vs round trips, group size, seasonality, booking channel, amount spent, days pre booked.

```
setwd("~/Fall Semester/Exploratory Data Analysis/airline_data")
customer_data_t=read.csv("customer_data_nm.csv")
customer_data_nm=customer_data_t[,-1]
```

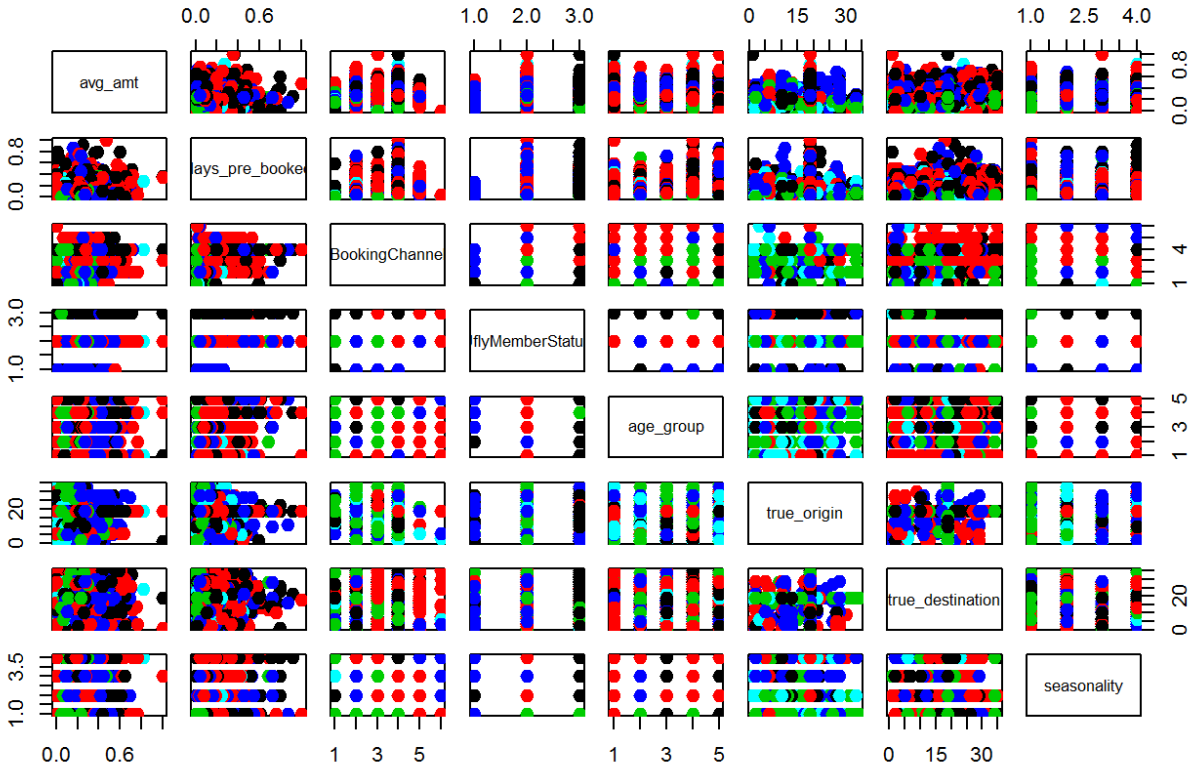
```
#k-modes clustering
plot(customer_data_nm[,c(1,3:9)], col = (kmed$clustering), main = "K medoids cl
ustering",pch = 20, cex = 2)
```

K medoids clustering

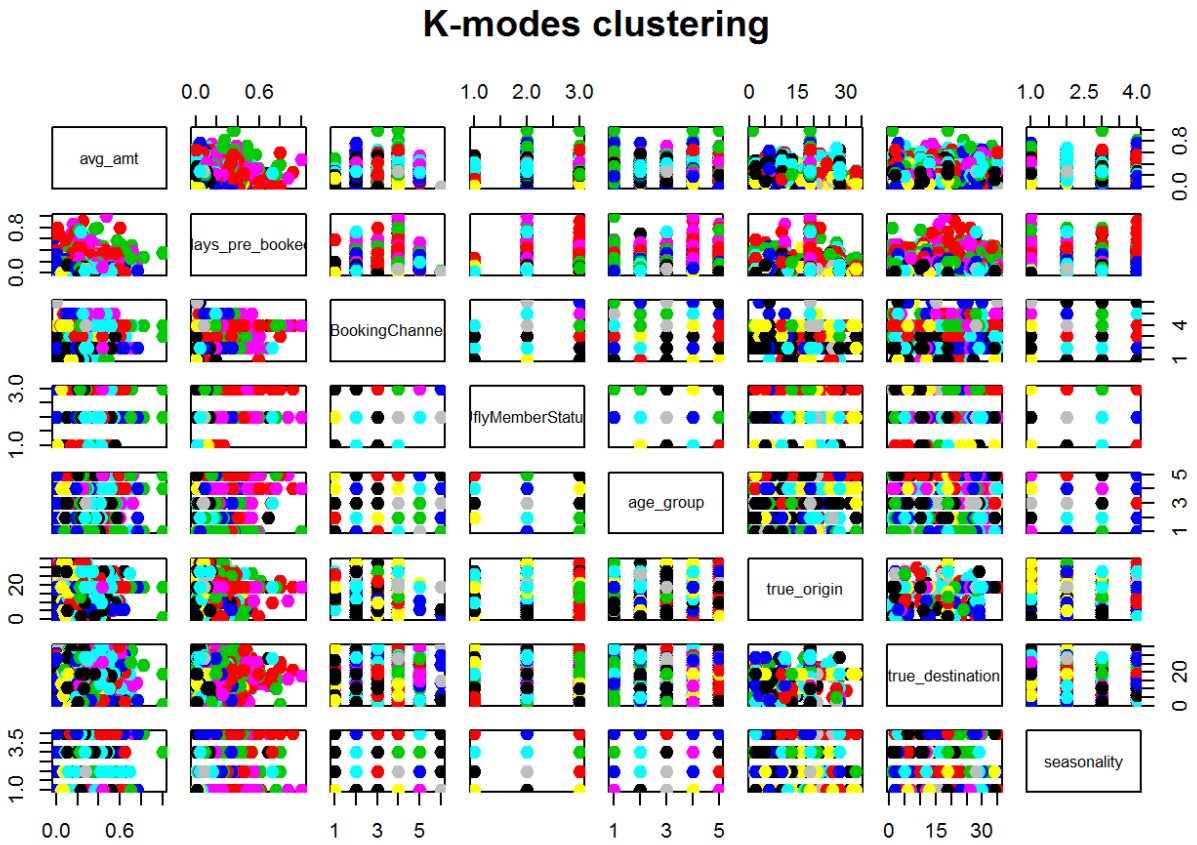



```
#hierarchical clustering
plot(customer_data_nm[,c(1,3:9)], col = (travel_groups), main = "Hierarchical c
lustering",pch = 20, cex = 2)
```

Hierarchical clustering



```
#k-medoids clustering
plot(customer_data_nm[,c(1,3:9)], col = (customer_kmodes$cluster), main = "K-m
odes clustering",pch = 20, cex = 2)
```



Overview

Data Engineering

The data obtained from the airline is not useful as such. But I made the data useful by doing the below steps. Also mentioned below are the reasons to do the same.

Filtering out data: The data we received from the airline is at flight level. The data also contains the trips in various flights other than the concernd airline. But we are interested in this concerned airline flight details. Hence we need to filter the data

Cleaning the data

Scenario to filter out the data: The data contains many missing values and data entry errors. We need to fix thohse errors. For example, the birth date and gender of passengers have many missing values. For this, filtering out the columns is the best method as we cannot randomnly put the values for the birth date and gender.

There are data entry errors in Custoemr Age. In order to make the segmentation useful, filtering out the instances which have unusual number as age is logical.

Scenario to make business assumptions. In order to make the dataset more useful, filtering out instances based on all the columns will result in losing too much data. Hence, we did make certain assumptions about the missing values of few attributes. For instance, for missing valaues in Ufly member status, we assumed that the passenger corresponding to that row is not a memeber of Ufly

program and assigned the status as ‘Non-member’.

In other case, if the rewards Number is 0, then we assumed that the passenger was not provided with rewards number and hence assigned the value of 0 to that passenger.

Data duplication: The data might contain duplicate entries of same customer making the same trip. Hence we Removingd the duplicate entries by retaining only the rows, which have unique PNR,sequence number, Customer Name and so on.

Transforming the level of data: In order to gain insights about customers and their travelling patterns, we need to segment them according to the trips they made with the airline. We can do it using unsupervised machihne learning. But for that we need data on Customer level. But we have data at flights level. For e.g. if a customer takes flight from Minneapolis to NewYork via Chicago, the data has one row for flights form Minneapolis to Chicago and one row for the filght from Chicago to NewYork. But ultimately, the goal is to segment the customers. Hence, we need to transform the data from flight level to Customer level. Also we would want to know whether each customer travelled in groups.

For that, we need to know where the customer flied to and from which place. This will give us more insights on where really the customers are heading to on which occasions. We also need tot know whether customer made round trip or a single trip ass this will give more information about each customer. Also, dates are too granular and it is impossible to comapare the flight patterns on each date in a year. Hence, we can group the dates into quarters.

Binning data: There are many attributes which will contian lots of continuous values such as age. But segmentation characterisitcs will give mean/median of each cluster, which might not give useful information about that cluster. In order to make the cluster more meaningful, we can bin the age into 4/5 groups.

Most of the segmentation algorithms use distance metrics to calculate similarity. But the data is in different units and scale. Hence we normalized the dataset.

Finally, we need data on customer level. For that, each row of the final data should contian a single row for each customer. This is acheived by agrregating each of the above discussed attributes on a customer-PNR level.

Segmentation

We can segment the customer using different clustering algorithms. The dataset contains few numerical and many categorical data.

Hence the k-means algorithm cannot be used as it is designed to handle numerrical data. I used k-medoids,kmodes and hierarchical clustering methods to segment the customers.

Kmedoids and hierarchcal clustering algorithm accepts similarity matrix. Getting the similarity matrix became challenge, as the distance needs to be calculated between numerical and categorical variables. I used Gower distance formula to accomplish this.

In Hierarchical clustering method, I used ward distance to calcualte the distance. Based ont he dendogram, I cut the tree to give five clusters. Then, I calculated median for numerical variables belonging to each cluster and mode for categorical variables.

In k-medoids algorithm, I initially used 5 clusters. But the silhouette co-efficient for 5 clusters did not come good. Hence I increased the cluster number to 8, which increased the silhouette co-efficient. But the resultant co-efficient is still less than the hierarchical clustering.

I tried k-modes algorithm. For k-modes algorithm, I created bins for average amount spent and number of days pre booked.Iagain used 8 clusters for k-modes algorithm as it gave the best result among k-modes. Then I tried k-modes without binning which did not improve the silhouette co-efficient much.

Then, I used the binned dataset in k-medoids algorithm, which did not improve the clustering performance much ass well.

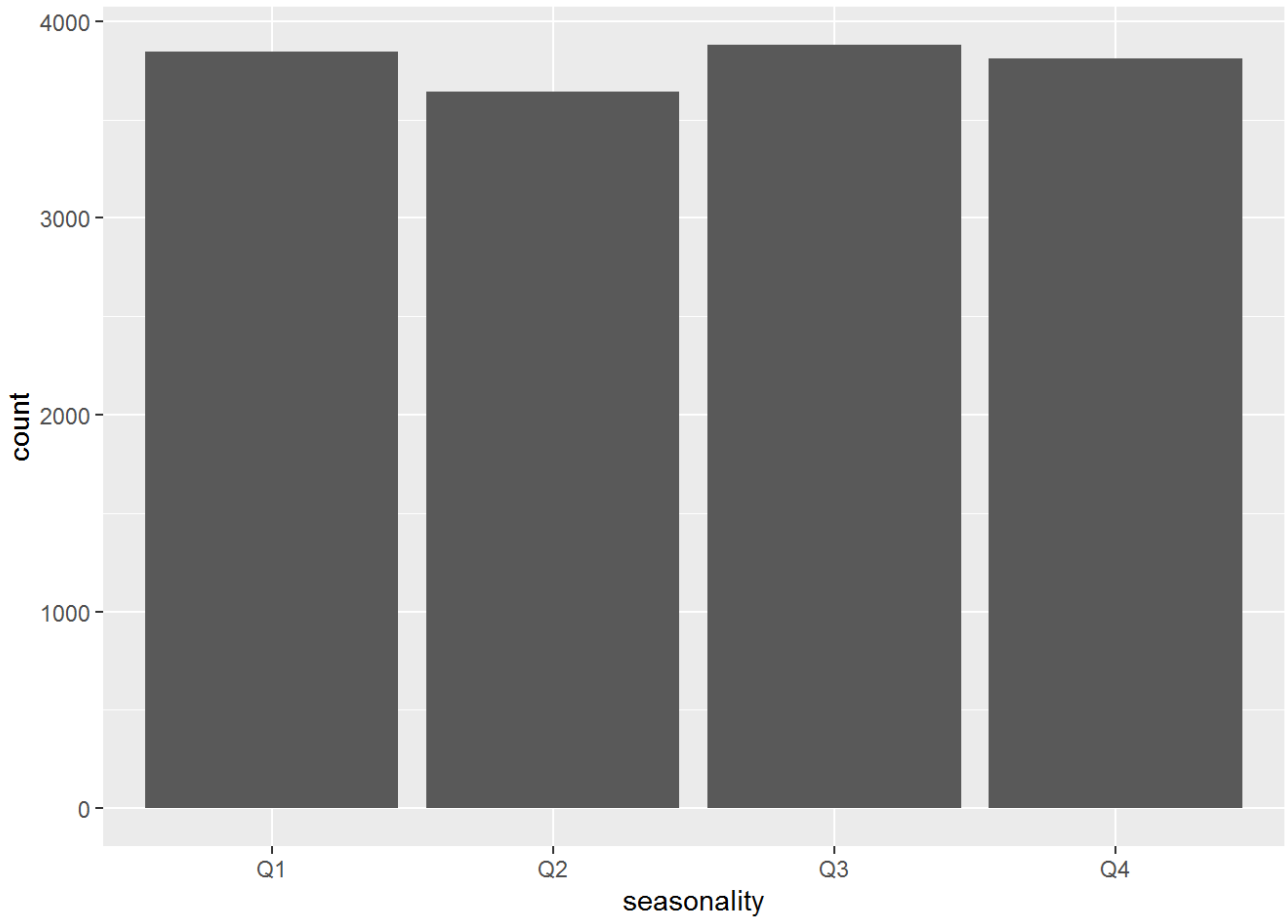
I did not use Gaussian mixture modelling as it accepts only the numerical variables. Converting categorical to numerical variables to calculate the distance did not make business sense as we do not know the weight of each categorical variables.

Visualization

```
by_booking_channel=customer_data%>%group_by(BookingChannel)%>%summarise(sum(avg
_amt))
by_booking_channel=setNames(by_booking_channel,c("Booking_channel","sum_of_AVG_
amount"))

#g1<-ggplot(customer_data)+aes(customer_data$days_pre_booked,customer_data$avg_
amt)+geom_point(pch=15,color=kmed_binned$clustering,size=1)
#g1
```

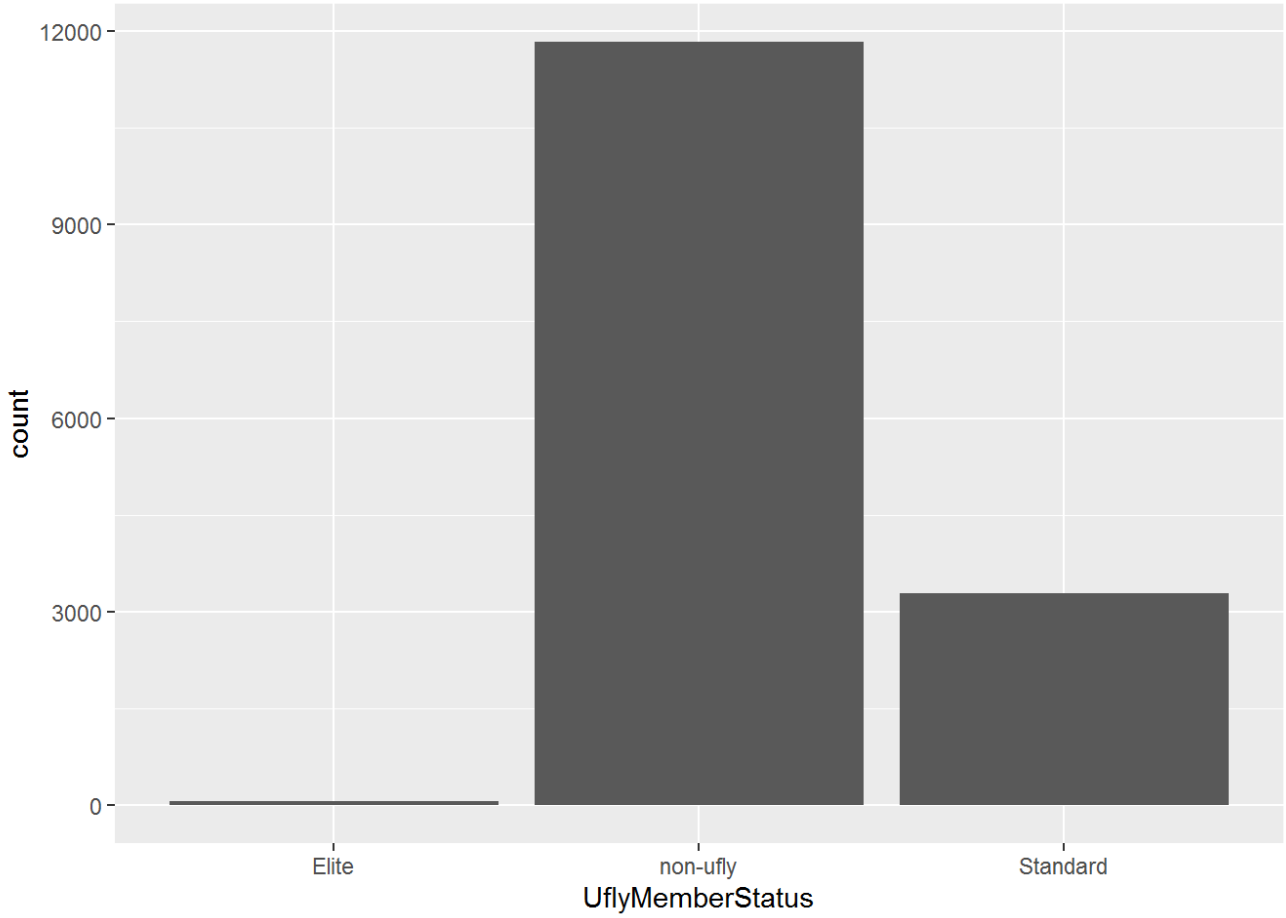
In the first plot, seasonality vs trip count is shown. This shows that, the number of trips from in the quarter 2 is less when compared to other quarters. Hence we need to take steps to increase the trips in Q2.

```
g1=ggplot(data=customer_data,aes(x=seasonality))+geom_bar()
g1
```



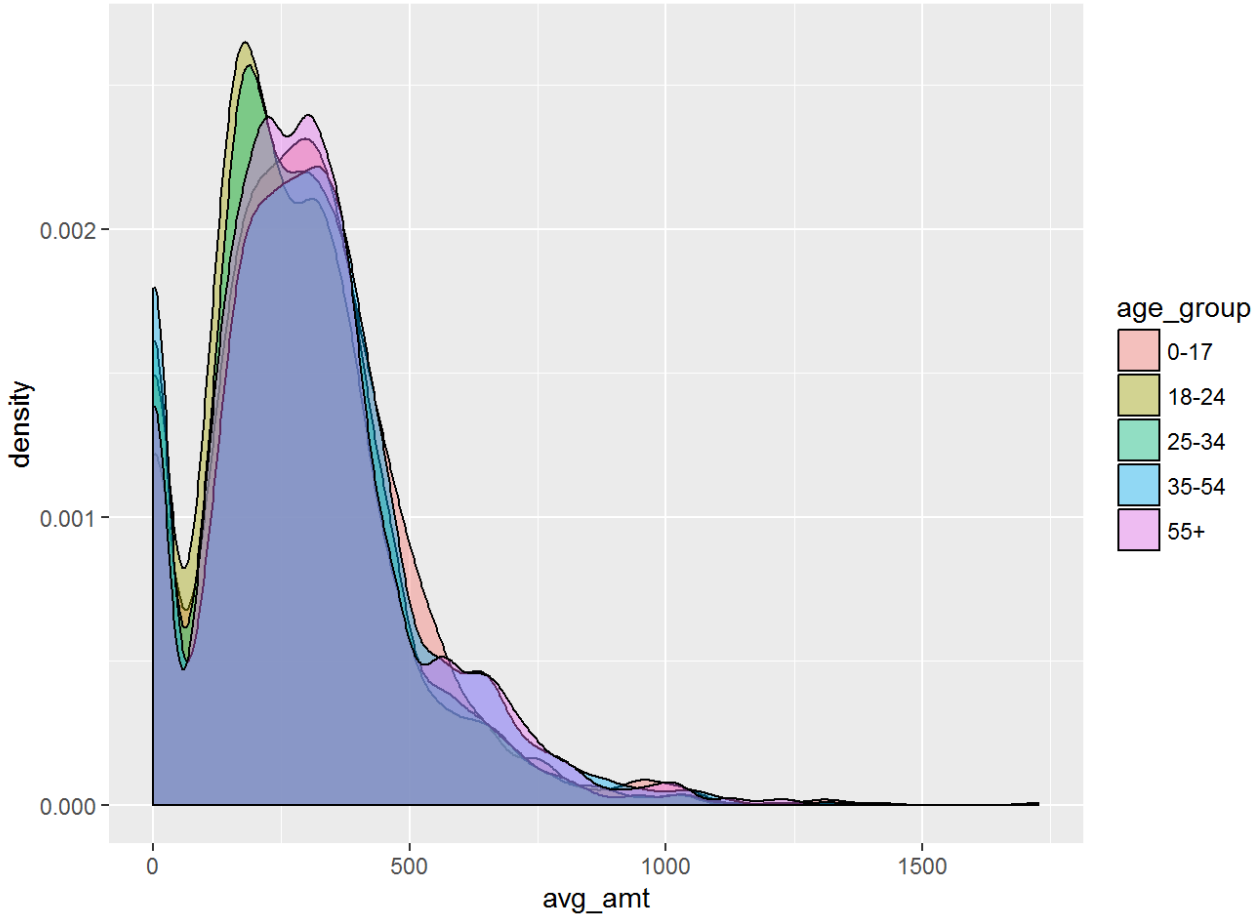
The below second plot shows that Elite members have very less number of trips when compared to standard class members. This is usual, but the elite memberrship count is very low. Hence, steps need to be taken to increase the elite membership.

```
g2=ggplot(data=customer_data,aes(x=UflyMemberStatus))+geom_bar()
g2
```



The below plot shows the distribution of amount spending based on the age group. Most of the people who belong to age-group of 18-24 spends around $200, whereas most of the people in both ends of the spectrum $300. We can see that, the number of people who spends more than $500 as average amount reduces drastically. Hencec steps need to be taken to increase customer engagement and customer spending.

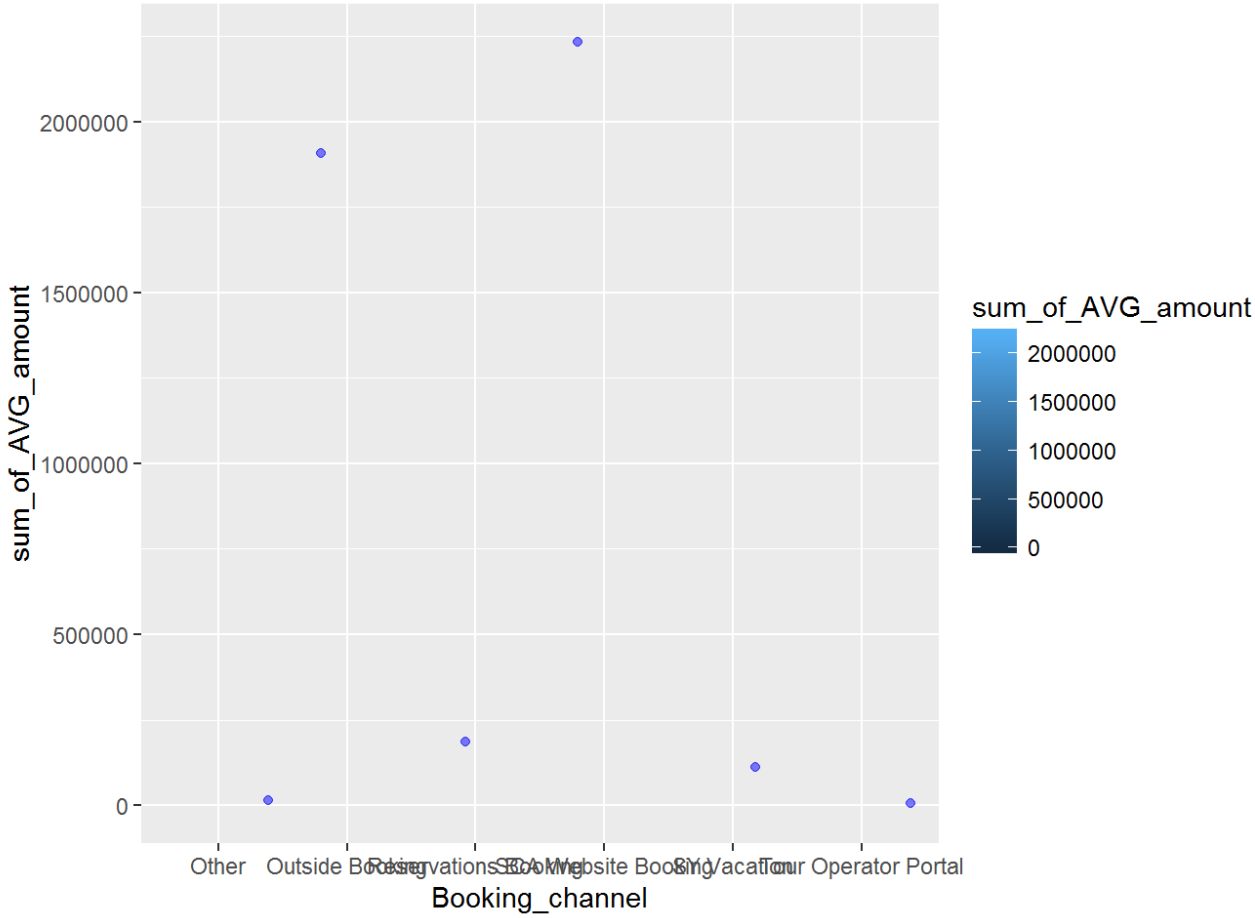
```
g3=ggplot(data=customer_data, aes(x=avg_amt, fill=age_group)) + geom_density(al
pha=.4)
g3
```



The below plot shows the total average amount per booking channel. Here, we can see that SCA weebsite booking contributes highest to the total amount of spending by customers. This means that the SCA website is very popular among the customers. We need to maintain this trend by giving importance to the compaany website.

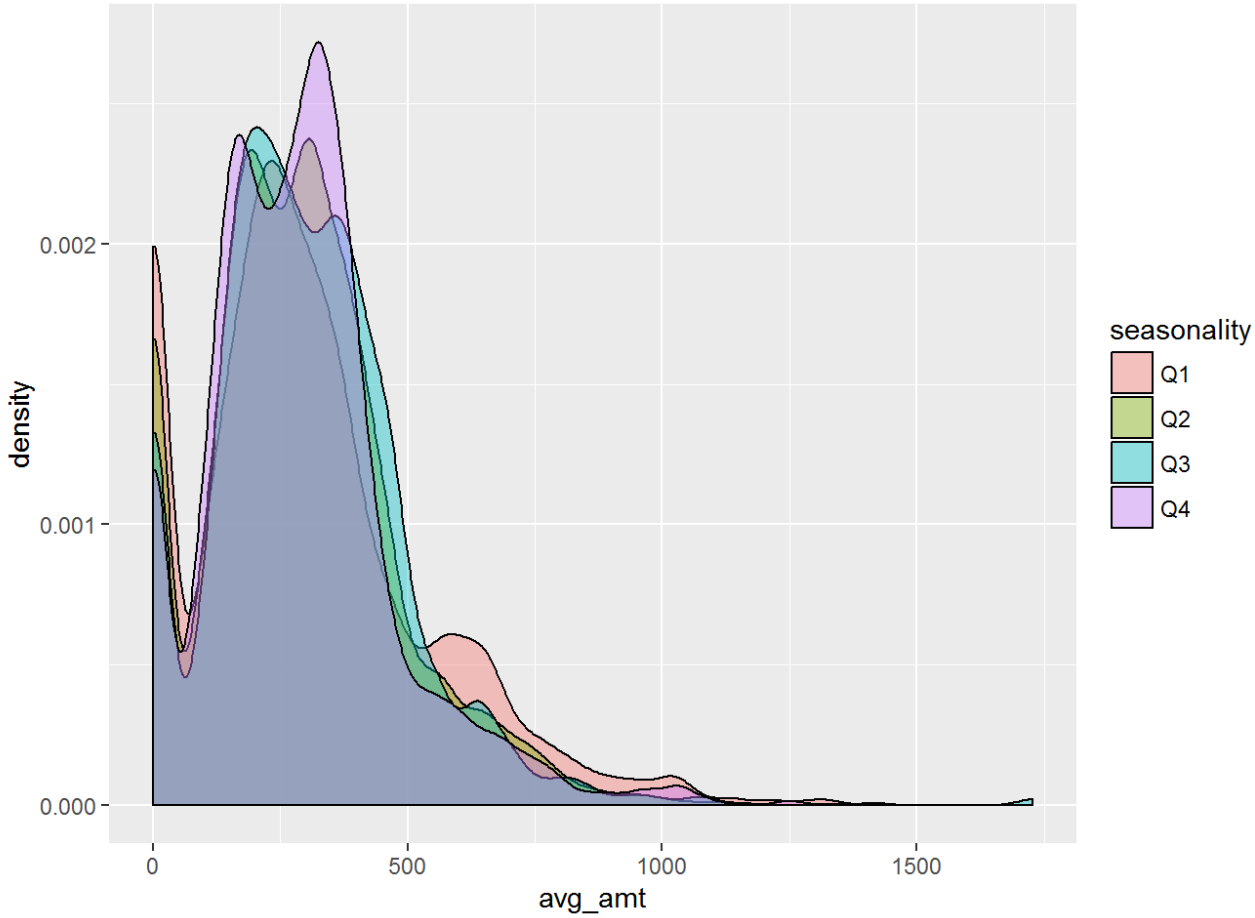
```
ggplot(by_booking_channel,aes(x=Booking_channel,y=sum_of_AVG_amount))+ geom_poi
nt(position = 'jitter',color='blue',alpha=.5) +geom_line(aes(colour = sum_of_AV
G_amount, group = sum_of_AVG_amount))
```

```
## geom_path: Each group consists of only one observation. Do you need to
## adjust the group aesthetic?
```



The below plot shows that people are spending more on Q4, which is expected due to holiday seasons. But whwn we look at the count, the number of trips is not the highest in Q4. This means that, less people travel by our airline but they spend more. This shows that the airline's ticket price is costlier when compared to other airlines. Hence less people choose our airline. We may need to dig deeper into ticket price during holidays in Q4.

```
g4=ggplot(data=customer_data, aes(x=avg_amt, fill=seasonality)) + geom_density(
alpha=.4)
g4
```



Power lies in simply understanding and exploring the data

Building correct models: Extracting business sense out of the data is the first step towrads building accurate models. Understanding the dataset means we can transform the data to a level, which will make complete business sense. For eg, in the current dataset, we wanted to segment the customers based on the trips they undertook. After we understood the dataset, we aggregated the data to customer level. If we do not study the data enough and its granularity, we might end up building completely worng models. Hence, understanding the data is very essential in building the correct models.

Attribute selection: Also, once we get a deeper understanding of data, we will get to know the features that will be important in modelling in the later stages. Also, we will know the attributes that doesn't make business sense. We can Removing those useless attributes. Hence after understanding the data, we will end up with a firm feature set which will help in reducing the modelling complexity.

Efficiency: By taking a deeper look into the data, we may come to know about the attributes which are not required for the models with its granular information. The best example for this that I could think of from the current dataset is travelling date. For segmentation and modelling, the models do not require the information of each date. We can either group the dates at week/month/quarterky level depending upon the need. This will make the model work more efficiently on the dataset without loosing too much information.

The downside of focusing too much on fancy models

One big advantages of fancy models is the underlying assumptions it amke while processing the data. Not caring about the underlying assumption to the minute detail will result in applying wrong models to the dataset which ultimately results in inaccurate results.

If results are not obtained correctly in one model, moving to other fancy models without analysing the underlying reasons will not help. The sample size of the data might not be good enough or it might be

due to some other reasons. In that case, applyng fancy models will not improve the results.