HW 3

Sarath Haridas, Abhinav Kumar, Tarun Newton, Pranav Pura Lingaraju, Kavya Puthuvaya, Pranav Srinivas Nagavelli

22 October, 2018

Contents

Business Problem

Michael Warnken (Senior Director of e-Commerce & General Manager) and Roselie Vaughn (Director of Customer Digital Experience) of Sun Country Airlines plan to develop more robust, data-driven customer insights in order to better structure their marketing efforts for the customers. To do so, they would like to better understand their customers.

Why Clustering?

We believe that by identifying segments of customers, the marketing efforts can be targeted to customer segments instead of tailoring the effort to individual customers. The segements could be identified in a way that ensures that the customers within the segment exhibit similar behaviour and customers across segments behave differently.

Given that there are no existing segments available, we use an unsupervised clustering alogrithm to identify the segments.

Data Cleaning

We import the data and explore the same to understand its structure, and clean the data to make it ready for analyses.

Data Import

Loading libraries

```
library(data.table)
library(lubridate)
library(dplyr)
library(stringr)
library(cluster)
library(ggplot2)
library(ggfortify)
library(stats)
library(factoextra)
library(naniar)
```

Importing data and removing rows that are completely duplicated.

Assumption

Since the data is at a trip-passenger level, duplicate records are data entry issues

```
# Data import
sun <- fread('SunCountry.csv', header = TRUE, stringsAsFactors = FALSE)

str(sun)
```

```
## Classes 'data.table' and 'data.frame':   3435388 obs. of  26 variables:
##  $ PNRLocatorID        : chr  "AAABJK" "AAABJK" "AAABMK" "AAABMK" ...
##  $ TicketNum           :integer64 3377365159634 3377365159634 3372107381942 3372107381942 33721074707
##  $ CouponSeqNbr        : int  2 1 2 1 1 1 1 1 1 1 ...
##  $ ServiceStartCity    : chr  "JFK" "MSP" "MSP" "SFO" ...
##  $ ServiceEndCity      : chr  "MSP" "JFK" "SFO" "MSP" ...
##  $ PNRCreateDate       : chr  "2013-11-23" "2013-11-23" "2014-02-04" "2014-02-04" ...
##  $ ServiceStartDate    : chr  "2013-12-13" "2013-12-08" "2014-02-23" "2014-02-20" ...
##  $ PaxName             : chr  "BRUMSA" "BRUMSA" "EILDRY" "EILDRY" ...
##  $ EncryptedName       : chr  "4252554D4241434B44696420493F7C2067657420746869732072696768745341414E4452
##  $ GenderCode          : chr  "F" "F" "M" "M" ...
##  $ birthdateid         : int  35331 35331 46161 46161 34377 39505 50874 34741 41690 38575 ...
##  $ Age                 : int  66 66 37 37 69 54 25 69 49 58 ...
##  $ PostalCode          : chr  "" "" "" "" ...
##  $ BkdClassOfService   : chr  "Coach" "Coach" "Coach" "Coach" ...
##  $ TrvldClassOfService : chr  "Coach" "First Class" "Discount First Class" "Discount First Class" ..
##  $ BookingChannel      : chr  "Outside Booking" "Outside Booking" "SCA Website Booking" "SCA Website
##  $ 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    : chr  "" "" "" "" ...
##  $ CardHolder          : logi  NA NA NA NA NA NA ...
##  $ BookedProduct       : chr  "CHEOPQ" "CHEOPQ" "" "" ...
##  $ EnrollDate          : chr  "" "" "" "" ...
##  $ MarketingFlightNbr  : chr  "244" "243" "397" "392" ...
##  $ MarketingAirlineCode: chr  "SY" "SY" "SY" "SY" ...
##  $ StopoverCode        : chr  "O" "" "O" "" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
# Removing row level duplicates
sun_undup <- sun[!duplicated(sun), ]
```

Analyzing the structure of the data

```r
summary(sun_undup)
```

```
##  PNRLocatorID         TicketNum              CouponSeqNbr
##  Length:3292499     Min.   :3372052115142   Min.   :1.000
##  Class :character   1st Qu.:3372107151902   1st Qu.:1.000
##  Mode  :character   Median :3372107829475   Median :1.000
##                     Mean   :3374375115251   Mean   :1.462
##                     3rd Qu.:3377300665844   3rd Qu.:2.000
##                     Max.   :3379578145804   Max.   :8.000
##
##  ServiceStartCity   ServiceEndCity     PNRCreateDate
##  Length:3292499     Length:3292499     Length:3292499
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  ServiceStartDate     PaxName          EncryptedName
##  Length:3292499     Length:3292499     Length:3292499
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   GenderCode         birthdateid           Age
##  Length:3292499     Min.   :-675290   Min.   :-2883.00
##  Class :character   1st Qu.:  39580   1st Qu.:   26.00
##  Mode  :character   Median :  45001   Median :   40.00
##                     Mean   :  44914   Mean   :   40.24
##                     3rd Qu.:  50134   3rd Qu.:   55.00
##                     Max.   :1112840   Max.   : 2012.00
##                     NA's   :29517     NA's   :29517
##   PostalCode        BkdClassOfService  TrvldClassOfService
##  Length:3292499     Length:3292499     Length:3292499
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
```

```
##
##   BookingChannel       BaseFareAmt       TotalDocAmt      UFlyRewardsNumber
##   Length:3292499     Min.   :   0.0   Min.   :    0.0   Min.   :100000191
##   Class :character   1st Qu.: 172.1   1st Qu.:  188.2   1st Qu.:200859411
##   Mode  :character   Median : 269.8   Median :  299.0   Median :202966820
##                      Mean   : 285.5   Mean   :  312.6   Mean   :204214760
##                      3rd Qu.: 366.5   3rd Qu.:  410.8   3rd Qu.:210380402
##                      Max.   :4342.0   Max.   :17572.0   Max.   :241086274
##                                                         NA's   :2614202
##   UflyMemberStatus   CardHolder      BookedProduct       EnrollDate
##   Length:3292499     Mode :logical   Length:3292499     Length:3292499
##   Class :character   FALSE:643578    Class :character   Class :character
##   Mode  :character   TRUE :34719     Mode  :character   Mode  :character
##                      NA's :2614202
##
##
##
##   MarketingFlightNbr MarketingAirlineCode StopoverCode
##   Length:3292499     Length:3292499       Length:3292499
##   Class :character   Class :character     Class :character
##   Mode  :character   Mode  :character     Mode  :character
##
##
##
##
```

```
dim(sun_undup)
```

```
## [1] 3292499      26
```

We have data for 3.3 million observations and 26 columns that captures travel information for each passenger for each trip. There are some irregularities that are clearly identifiable from certain columns

Findings

- Age and birthdate: We observe that the age has values that are missing. And, there are observations with negative age, which might have arised from a data entry issue at the birthdate column. We also have observations for age that are over 100 years old.
- GenderCode: We observe null values, and there are records where the gender is marked as U (Unknown)
- UflyRewardsNumber and CardHolder: We observe that there are multiple NAs for these two. This could either be because the customer did not have an SC credit card or that he didn't use this SC Credit card for the specific transaction. For the purpose of this analysis, we assume that the customer did not have an SC credit card.
- MarketingAirlineCode: There are airline codes capturing for other airlines as well. SY refers to Sun Country, and our analysis is going to focus only for Sun Country airlines
- BaseFareAmount and TotalDocAmount: We observe that there are records where the BaseFareAmount and the TotalFare amount are over USD 10,000. The maximum ticket price that Sun Country flies for Domestic in 2018 is USD 800. Therefore, we have to treat these extreme values before using them in any analysis

We subset the data for only Sun Country flights by filtering for records having the MarketingAirlineCode as SY. Because we are primarily interested in identifying customer patterns for Sun Country airlines only.

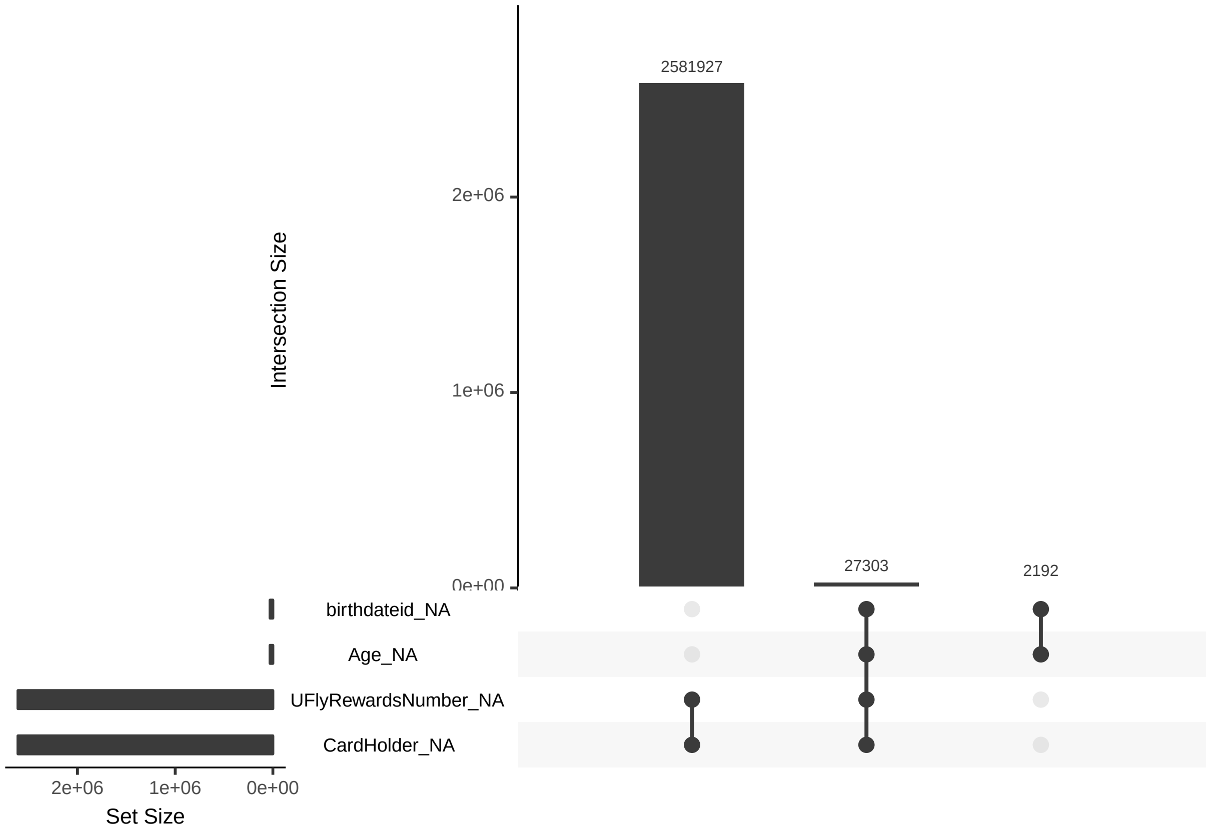
```
sun_undup <- sun_undup %>% filter(MarketingAirlineCode == 'SY')
```

```
## Warning: package 'bindrcpp' was built under R version 3.5.1
```

Missing Value Treatment

Firstly, we try to identify the sparsity of the missing values for the columns.

```r
gg_miss_upset(sun_undup, nsets = 4)
```



Description

The graph above captures the number of missing records in the data, and also captures the interaction between the missing values. The linkages at the bottom capture which variables are missing in common for the corresponding bar. For example, the first bar has around 258K missing values for both the UFlyRewardsNumber and the CardHolder variables.

Interpretation

We observe that the missing values are present for 4 columns only. Out of which, the missing values for UflyRewardsNumber and CardHolder are present across the same set of records, and the missing values for Age and BirthDateId are present across the same set of records.

Treating for UFlyRewardsNumber and CardHolder

The missing values for the UFlyRewardsNumber and CardHolder are present across the same set of records. Hence, the missing values can be replaced with dummy values for both the columns.

For the UFlyRewardsNumber, we can replace the missing values with 0 and CardHolder with "NA".

```r
sun_undup$UFlyRewardsNumber <- ifelse(is.na(sun_undup$UFlyRewardsNumber), 0,
                                      sun_undup$UFlyRewardsNumber)
sun_undup$CardHolder <- ifelse(is.na(sun_undup$CardHolder), "NA",
                               sun_undup$CardHolder)
```

Treating for Age and BirthDate

Before treating for the missing value of age, we must analyze whether the variables are missing at random or if there exists a pattern that would help us impute the missing values.

```r
# Comparing missingness of age with other variables
# Service Start Date
table("Age Missing" = is.na(sun_undup$Age),
      "Service Start Month" = lubridate::month(sun_undup$ServiceStartDate))
```

```
##            Service Start Month
## Age Missing      1      2      3      4      5      6      7      8      9
##       FALSE 236524 267668 347205 240451 236937 281157 325768 297233 204895
##       TRUE    3530   6249   5639   2981    990   1415   1532   1618    924
##            Service Start Month
## Age Missing     10     11     12
##       FALSE 242505 232498 345185
##       TRUE    1382   1241   1994
```

```r
# PNR Create Date
table("Age Missing" = is.na(sun_undup$Age),
      "PNR Create Month" = lubridate::month(sun_undup$PNRCreateDate))
```

```
##            PNR Create Month
## Age Missing      1      2      3      4      5      6      7      8      9
##       FALSE 306920 234645 242075 232024 277329 291784 300614 273871 288139
##       TRUE    1892   1369   1326   1174   2601   2217   2485   2714   4205
##            PNR Create Month
## Age Missing     10     11     12
##       FALSE 295906 279688 235031
##       TRUE    4224   3267   2021
```

```r
# Gender Code
table("Age Missing" = is.na(sun_undup$Age),
      "Gender Code" = sun_undup$GenderCode)
```

```
##            Gender Code
## Age Missing               F       M       U
##       FALSE       0 1697969 1560019      38
##       TRUE    29495       0       0       0
```

```r
# Booking Channel
table("Age Missing" = is.na(sun_undup$Age),
      "Booking Channel" = sun_undup$BookingChannel)
```

```
##            Booking Channel
## Age Missing     ANC     BOS     DCA     DFW     FCM     GJT     HRL
##       FALSE       9       1      24     503    3513       1      18
##       TRUE        0       0       0       0      15       0       0
##            Booking Channel
## Age Missing     JFK     LAN     LAS     LAX     MCO     MDW     MIA
##       FALSE     254     252     177     412      17     202       1
##       TRUE        0       3       1       0       0       0       0
##            Booking Channel
## Age Missing     MKE     MSN     MSP Outside Booking     PHX     PSP
##       FALSE     257       1    4788         1444752      21      28
##       TRUE        0       0      23            3075       0       0
##            Booking Channel
```

```
## Age Missing Reservations Booking     RSW SCA Website Booking     SEA
##       FALSE                161321      89             1426937      42
##        TRUE                 22828       0                2567       0
##            Booking Channel
## Age Missing     SFO SY Vacation Tour Operator Portal     UFO     XTM
##       FALSE     141       87278               126365     147     475
##        TRUE       0         677                  306       0       0
```

Interpretation

Similarly, comparing across all the other variables we are unable to observe any pattern with respect to any variable. Therefore, we infer that the values for age are missing at random, and can be removed. Also, there are only 38 records for which the Gender is Unknown ("U"). Therefore, these records with unknown Gender code can also be removed.

```
sun_undup_age_rm <- sun_undup %>% filter(!is.na(Age) & GenderCode != 'U')
```

Conclusion

- The missing values for the UFlyRewardsNumber and CardHolder have been replaced with dummy values.
- The missing values for the Age column were removed as they were identified to be missing at random

Outlier Treatment

We had identified outliers in the Age, TotalDocAmount and the BaseFareAmount.

Treating for Age

For age, there are only 4 records where the age is negative. Also, there are only a few records where age is > 100. There cannot be any person who has a negative age. Also, there are hardly people over the age of 100 who travel. Hence, these have to be data entry errors. These also have to be treated akin to the missing age approach. Also, since there are only a few records, we remove these records where age is beyond these limits instead of replacing them with median. *Assumption*

The data for which ages are out of the threshold are also missing at random, and therefore, removing them would not impact any inferences drawn from the data.

```
sun_undup_age_rm <- sun_undup %>%
  filter(Age > 0 & Age <= 100)
```

Description

Treating for amounts

We identify the univariates for the variables TotalDocAmount and the BaseFareAmount. As mentioned above, the current highest ticket price in the Sun Country website is around USD 800. Therefore, we can cap the prices at the 99.9th percentile for those records with higher prices.

The 99.9th percentiles for these 2 variables are USD 1,364 and USD 1,258.

```
quantile(sun_undup_age_rm$BaseFareAmt, probs = c(0, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75,
                                        0.9, 0.95, 0.99, 0.995, 0.999, 1))
```

```
##      0%      1%      5%     10%     25%     50%     75%     90%     95%
##    0.00    0.00    0.00   81.86  171.16  269.76  366.52  506.05  613.72
##     99%   99.5%   99.9%    100%
##  858.00  978.00 1258.00 4342.00
```

1258
```r
# 1258
quantile(sun_undup_age_rm$TotalDocAmt, probs = c(0, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75,
                                                  0.9, 0.95, 0.99, 0.995, 0.999, 1))
```

```
##        0%        1%        5%       10%       25%       50%       75%
##     0.000     0.000     0.000     0.000   187.800   298.000   409.800
##       90%       95%       99%     99.5%     99.9%      100%
##   569.240   683.900   942.250  1067.986  1364.511 17572.000
```

```r
# 1364

sun_fin <- sun_undup_age_rm %>%
  mutate(BaseFareAmt = ifelse(BaseFareAmt > 1258, 1258, BaseFareAmt),
         TotalDocAmt = ifelse(TotalDocAmt > 1364, 1364, TotalDocAmt))
```

Description

We also want to make sure we capture those customers who have had their first leg of journey from Sun Country.

Assumption

It is possible that certain customers flew another airline and used Sun Country as a connector. We next filter our data to include only those PNR IDs for which Sun Country was the first flight. We identify the leg of journey using the Coupon Sequence Number and filter out PNR IDs which don't start with coupon sequence number of 1.

```r
pnr <- sun_fin %>%
  group_by(PNRLocatorID) %>%
  summarise(min_cpn = min(CouponSeqNbr)) %>%
  filter(min_cpn == 1)
```

Conclusion

- The values for BaseFareAmount and TotalDocAmount were capped at their respective 99.9th percentile values
- Records where Age was negative or over 100 were assumed as data entry errors and treated as missing at Random and removed as per the Missing Value Treatment approach

Identifying Unique Customer

To identify customers, we need a unique key to identify customers. We could use the UFlyRewardsNumber to identify unique members. But we have no unique identifier for a non-member. Therefore, we have to identify a unique member identifier across all customers.

While there was no single column or combination of columns that was able to identify a unique member for the whole dataset, we observed that the combination of EncryptedName, BirthDateId and GenderCode was mostly unique. Hence, we assume that the unique identifier for a member is this combination.

We create a new column pkey which is a concatenation of the values in these columns. We use the pkey for the analysis going forward.

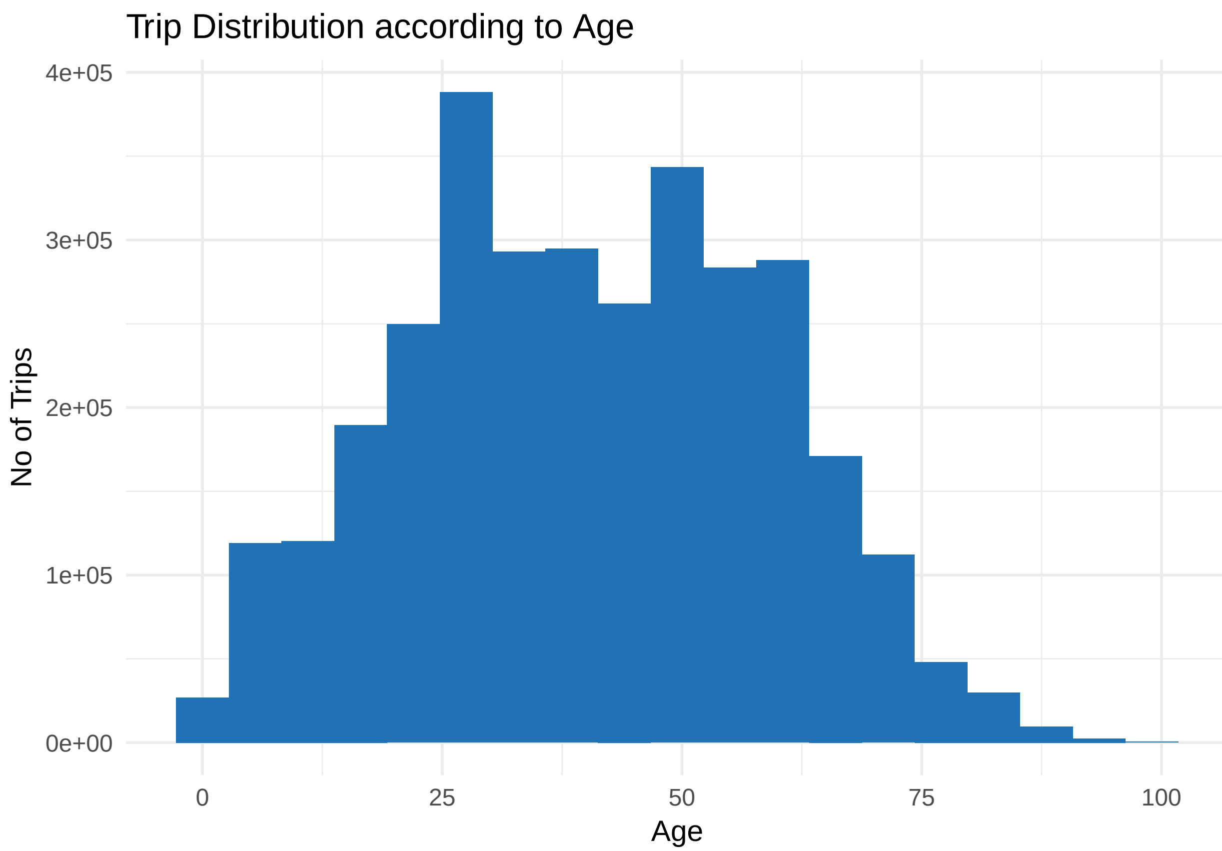
```r
sun_fin_out <- sun_fin %>%
  filter(PNRLocatorID %in% pnr$PNRLocatorID) %>%
  mutate(pkey = paste(EncryptedName, birthdateid, GenderCode, sep = "-"))
```

Data Exploration

We explore the data to understand common patterns or trends that would help us in identifying customer groups.

Age distribution

```r
ggplot(data = sun_fin_out) +
  aes(x = Age) +
  geom_histogram(bins = 19, fill = "#2171b5") +
  labs(title = "Trip Distribution according to Age",
       x = "Age",
       y = "No of Trips") +
  theme_minimal()
```



Trip count for top 10 destinations

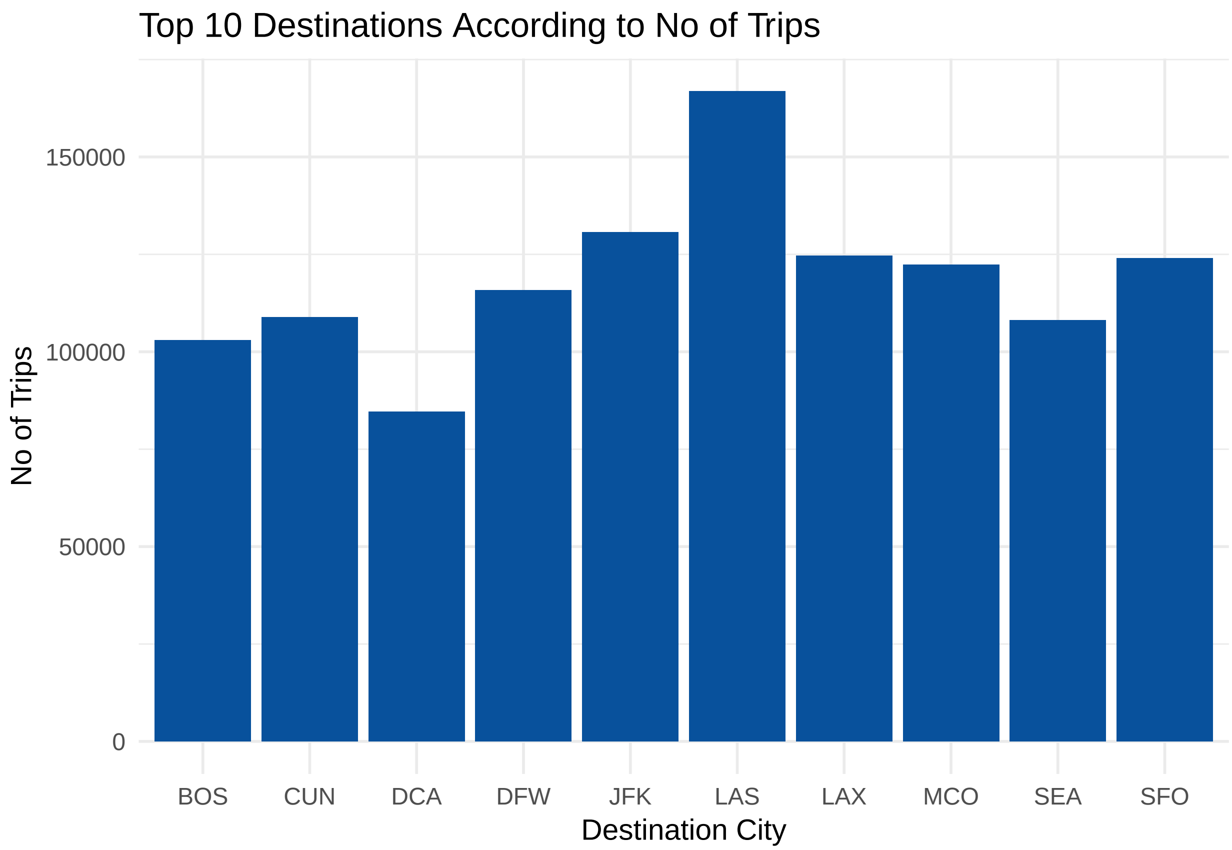
```r
sun_end_city <- sun_fin_out %>%
  group_by(ServiceEndCity) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>% head(11)

sub <- sun_end_city$ServiceEndCity[c(2:11)]

sun_end_city_top <- sun_fin_out  %>%
```

```r
  filter(ServiceEndCity %in% sub)

ggplot(data = sun_end_city_top) +
  aes(x = ServiceEndCity) +
  geom_bar(fill = "#08519c") +
  labs(title = "Top 10 Destinations According to No of Trips",
       x = "Destination City",
       y = "No of Trips") +
  theme_minimal()
```



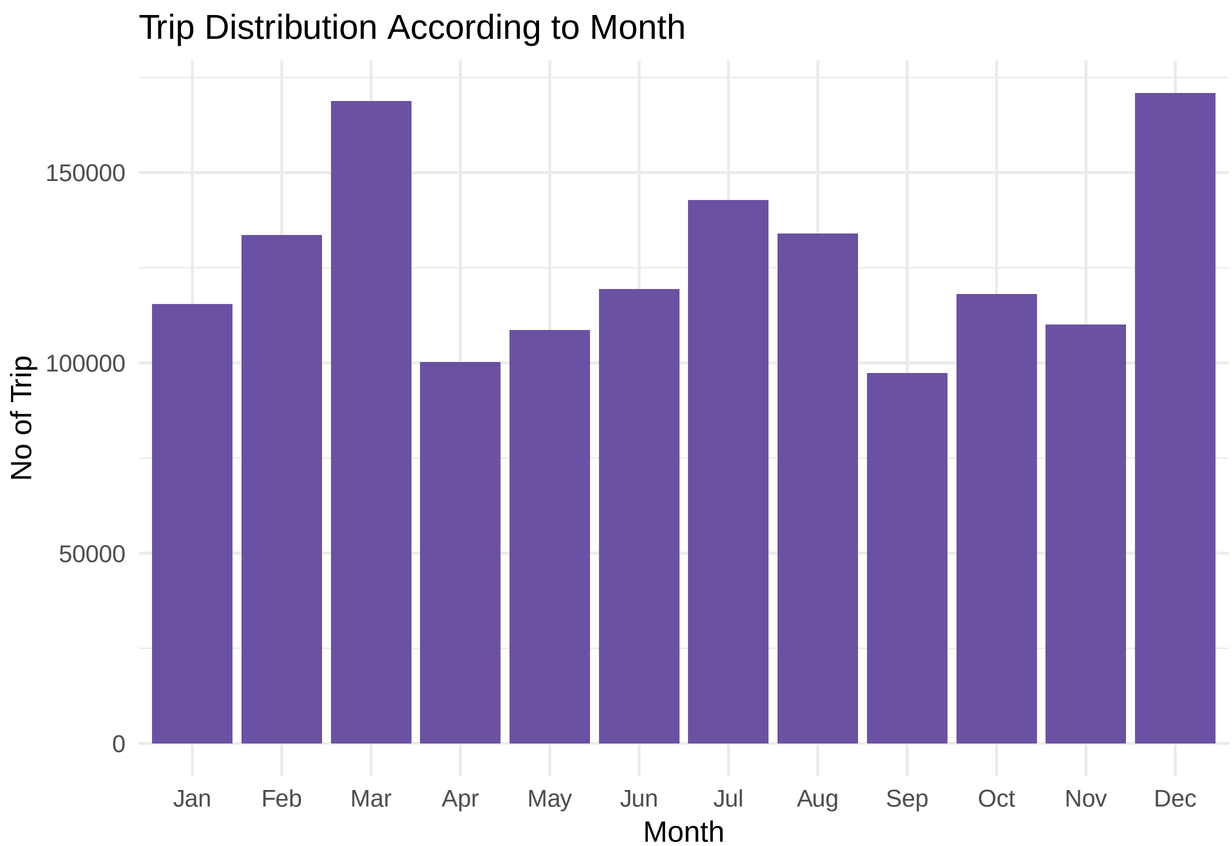
Trip distribution according to month

```r
sun_fin_out_qtr <-sun_fin_out
sun_fin_out_qtr$qtr <- lubridate::quarter(sun_fin_out_qtr$ServiceStartDate)
sun_fin_out_qtr <- sun_fin_out_qtr %>%
  filter(ServiceStartCity %in% "MSP")

date_dataset <- sun_fin_out_qtr %>%
  group_by(month=lubridate::month(ServiceStartDate)) %>%
  summarise(count = n())
date_dataset$month_abb <- month.abb[date_dataset$month]
date_dataset$month_abb_fac = factor(date_dataset$month_abb, levels = month.abb)

ggplot(data = date_dataset) +
  aes(x = month_abb_fac, weight = `count`) +
```

```
geom_bar(fill = "#6a51a3") +
labs(title = "Trip Distribution According to Month",
     x = "Month",
     y = "No of Trip") +
theme_minimal()
```



Data Transformation

To identify groups of similar customers, we need to create customer level clusters. To do so, we identify attributes that pertain to every individual customer. We aggregate from the trip level data to a customer level to arrive at our final dataset.

Analytical Dataset Structure

The table below details the list of columns that are captured in the analtyical dataset.

Column Name	Description	Comments
pkey	Primary Key for the customer	
GenderCode	Gender	1 for Male
age	Age	
amount	TotalDocAmount Paid by the customer	
bkng_chnl_out	Booking Channel - Outside Booking	
bkng_chnl_reserv	Booking Channel - Reservation Booking	
bkng_chnl_syvac	Booking Channel - SY Vacation Booking	
bkng_chnl_tour	Booking Channel - Tour Operator	

Column Name	Description	Comments
bkng_chnl_web	Booking Channel - SY Website Booking	
booked_coach_travel	Booked Class - Coach	
booked_fc_travel	Booked Class - First Class	
card_holder	SY Card Holder	
city_per_trip	Number of cities visited	
no_booking	Number of tickets booked	Distinct PNR Count
travel_coach_travel	Travelled Class - Coach	
travel_fc_travel	Travelled Class - First Class	
avg_min_dbd	Min Days Before Departure Tickets Booked	
avg_max_dbd	Max Days Before Departure Tickets Booked	
avg_len_stay	Avg Length of Stay if Round Trip	
min_len_stay	Min Length of Stay if Round Trip	
max_len_stay	Max Length of Stay if Round Trip	

Creating the Analytical Dataset

Description

Since the amount of data involved is huge, instead of aggregating the data at once, we aggregate individual columns and later merge them together. In this process, we remove the redundant data frames as well to conserve space. At the end of this approach, all the intermediate tables are removed, and only the required tables remain, which include the raw data and the customer level analytical dataset.

Post the join, we convert the GenderCode column to a numeric by replacing the Male with 1 and Female with 0. This is to ensure that we are able to run our algorithms like k-Means or Hierarchical clustering.

Assumption

- The customer behaviour is the same across the 2 years under consideration
- The customer behaviour will remain stable for the foreseeable future and the results of the analysis can be generalized for the current calendar year as well

```
rm(sun)
rm(sun_undup)
rm(sun_undup_age_rm)
rm(pnr)
rm(sun_fin)
rm(date_dataset)
rm(sun_end_city)
rm(sun_end_city_top)
rm(sun_fin_out_qtr)
rm(sub)
```

```
# Getting Age
cust_raw_age <- sun_fin_out %>% group_by(pkey, GenderCode) %>%
  summarise(age = max(Age))

# Getting Number of Bookings
cust_raw_no_booking <- sun_fin_out %>% group_by(pkey, GenderCode) %>%
  summarise(no_booking = n_distinct(PNRLocatorID))

# Getting Number of Bookings with Coach
cust_raw_booking_coach_travel <- sun_fin_out %>% group_by(pkey, GenderCode) %>%
  summarise(booked_coach_travel = sum(BkdClassOfService == 'Coach'))
```

```r
# Getting Number of Bookings with First Class
cust_raw_booking_fc_travel <- sun_fin_out %>% group_by(pkey, GenderCode) %>%
  summarise(booked_fc_travel = sum(BkdClassOfService == 'First Class'))

# Getting Number of Travels in Coach
cust_raw_travel_coach_travel <- sun_fin_out %>% group_by(pkey, GenderCode) %>%
  summarise(travel_coach_travel = sum(BkdClassOfService == 'Coach'))

# Getting Number of Travels in First Class
cust_raw_travel_fc_travel <- sun_fin_out %>% group_by(pkey, GenderCode) %>%
  summarise(travel_fc_travel = sum(BkdClassOfService == 'First Class'))

# Getting TotalDocAmount Paid by the Traveller
cust_raw_amount <- sun_fin_out %>% group_by(pkey, GenderCode) %>%
  summarise(amount = sum(TotalDocAmt))

# Getting the Number of Cities the Visited
cust_raw_city <- sun_fin_out %>% group_by(pkey, GenderCode) %>%
  summarise(city_per_trip = n_distinct(ServiceEndCity))

# Getting the Number of Instances of Outside Booking
cust_raw_bkng_chnl_out <- sun_fin_out %>% group_by(pkey, GenderCode) %>%
  summarise(bkng_chnl_out = sum(BookingChannel == 'Outside Booking'))

# Getting the Number of Instances of Reservations Booking
cust_raw_bkng_chnl_reserv <- sun_fin_out %>% group_by(pkey, GenderCode) %>%
  summarise(bkng_chnl_reserv = sum(BookingChannel == 'Reservations Booking'))

# Getting the Number of Instances of Tour Operator Booking
cust_raw_bkng_chnl_tour <- sun_fin_out %>% group_by(pkey, GenderCode) %>%
  summarise(bkng_chnl_tour = sum(BookingChannel == 'Tour Operator Portal'))

# Getting the Number of Instances of SY Vacation Booking
cust_raw_bkng_chnl_syvac <- sun_fin_out %>% group_by(pkey, GenderCode) %>%
  summarise(bkng_chnl_syvac = sum(BookingChannel == 'SY Vacation'))

# Getting the Number of Instances of SCA Website Booking
cust_raw_bkng_chnl_web <- sun_fin_out %>% group_by(pkey, GenderCode) %>%
  summarise(bkng_chnl_web = sum(BookingChannel == 'SCA Website Booking'))

# Identifying whether the Customer is a Card Holder
cust_raw_card_holder <- sun_fin_out %>% group_by(pkey, GenderCode) %>%
  summarise(card_holder = sum(!is.na(CardHolder)))

# Identifying the Days Before Departure and Length of Stay Metrics
cust_raw_datediff <- sun_fin_out %>% group_by(pkey, GenderCode, PNRLocatorID) %>%
  summarise(min_dbd = min(date_diff), max_dbd = max(date_diff)) %>%
  mutate(len_stay = max_dbd - min_dbd) %>%
  group_by(pkey, GenderCode) %>%
  summarize(avg_min_dbd = mean(min_dbd), avg_max_dbd = mean(max_dbd),
            avg_len_stay = mean(len_stay), min_len_stay = min(len_stay),
            max_len_stay = max(len_stay))
```

```r
# Merging Individual Datasets and Dropping Redundnat datasets
cust_raw_1 <- merge(cust_raw_age, cust_raw_amount)

rm(cust_raw_age)
rm(cust_raw_amount)

cust_raw_2 <- merge(cust_raw_1, cust_raw_bkng_chnl_out)

rm(cust_raw_1)
rm(cust_raw_bkng_chnl_out)

cust_raw_3 <- merge(cust_raw_2, cust_raw_bkng_chnl_reserv)

rm(cust_raw_2)
rm(cust_raw_bkng_chnl_reserv)

cust_raw_4 <- merge(cust_raw_3, cust_raw_bkng_chnl_syvac)

rm(cust_raw_3)
rm(cust_raw_bkng_chnl_syvac)

cust_raw_5 <- merge(cust_raw_4, cust_raw_bkng_chnl_tour)

rm(cust_raw_4)
rm(cust_raw_bkng_chnl_tour)

cust_raw_6 <- merge(cust_raw_5, cust_raw_bkng_chnl_web)

rm(cust_raw_5)
rm(cust_raw_bkng_chnl_web)

cust_raw_7 <- merge(cust_raw_6, cust_raw_booking_coach_travel)

rm(cust_raw_6)
rm(cust_raw_booking_coach_travel)

cust_raw_8 <- merge(cust_raw_7, cust_raw_booking_fc_travel)

rm(cust_raw_7)
rm(cust_raw_booking_fc_travel)

cust_raw_9 <- merge(cust_raw_8, cust_raw_card_holder)

rm(cust_raw_8)
rm(cust_raw_card_holder)

cust_raw_10 <- merge(cust_raw_9, cust_raw_city)

rm(cust_raw_9)
rm(cust_raw_city)

cust_raw_11 <- merge(cust_raw_10, cust_raw_no_booking)
```

```r
rm(cust_raw_10)
rm(cust_raw_no_booking)

cust_raw_12 <- merge(cust_raw_11, cust_raw_travel_coach_travel)

rm(cust_raw_11)
rm(cust_raw_travel_coach_travel)

cust_raw_13 <- merge(cust_raw_12, cust_raw_travel_fc_travel)

rm(cust_raw_12)
rm(cust_raw_travel_fc_travel)

cust_raw_fin_1 <- merge(cust_raw_13, cust_raw_datediff)

rm(cust_raw_13)
rm(cust_raw_datediff)

# One-Hot Encoding of GenderCode variable
cust_raw_fin_1$GenderCode <- ifelse(cust_raw_fin_1$GenderCode == 'M', 1, 0)
```

This is the anlaytical datataset that will be used for the clustering algorithm. The summary of it is below.

```r
# Top records for the Analytical Dataset
head(cust_raw_fin_1[, c(2:21)])
```

```
##   GenderCode age amount bkng_chnl_out bkng_chnl_reserv bkng_chnl_syvac
## 1          1  33  174.0             0                0               0
## 2          1  24  231.9             1                0               0
## 3          0  54  294.9             0                0               0
## 4          1  52    0.0             0                0               0
## 5          1  29  973.6             2                0               0
## 6          1  50  294.9             0                0               0
##   bkng_chnl_tour bkng_chnl_web booked_coach_travel booked_fc_travel
## 1              0             1                   1                0
## 2              0             0                   1                0
## 3              0             0                   1                0
## 4              2             0                   2                0
## 5              0             0                   2                0
## 6              0             0                   1                0
##   card_holder city_per_trip no_booking travel_coach_travel
## 1           0             1          1                   1
## 2           0             1          1                   1
## 3           0             1          1                   1
## 4           0             2          1                   2
## 5           2             2          1                   2
## 6           0             1          1                   1
##   travel_fc_travel avg_min_dbd avg_max_dbd avg_len_stay min_len_stay
## 1                0           7           7            0            0
## 2                0          50          50            0            0
## 3                0           0           0            0            0
## 4                0           9          16            7            7
## 5                0           9          11            2            2
## 6                0           0           0            0            0
##   max_len_stay
```

```
## 1                0
## 2                0
## 3                0
## 4                7
## 5                2
## 6                0
```

Identifying Customer Segments

Post the creation of the analytical dataset, we have to segment the users into multiple groups. However, there are multiple approaches that can be leveraged to form clusters. There can be Hierarchical or Partitioning methods that can be used to create clusters.

Hierarchical methods

Hierarchical clustering algorithms actually belong to 2 categories:

- Bottom-up
- Top-down

Bottom-up algorithms treat each data point as a single cluster at the beginning and then successively merge pairs of clusters until all clusters have been merged into a single cluster that contains all data points. Bottom-up hierarchical clustering is therefore called Agglomerative Clustering. This hierarchy is represented as a tree or dendrogram.

Top-down algorithms flow in the opposite direction of the bottom-up algorithm. They start with all the data points and successively split the cluster into pairs until the end nodes are all the individual points.

conclusion Since each customer is unique and there exists no sub-segment level(s), we can assume that there exists no inherent hierarchical order in our data. Therefore, we can infer that the hierarchical methods are not suitable for cluster identification. We also substantiated this by performing hierarchical clustering and not observing satisfactory results.

Partitioning methods Partitioning clustering is used to classify observations into multiple groups based on their similarity. The paritioning algorithm works by iteratively re-allocating observations between clusters until a stable partition is reached. However, the number of clusters need to be specified by the user. Similarity is calculated based on distance calculations.

We go ahead with the partitioning clustering method, with k-means as the first algorithm.

Rescaling data

Description

To apply distance based clustering, the first step is to rescale the numeric data columns ie., all numeric columns should have the same range of values. This process called normalization, will help us in handling columns that have varying scales. By normalizing, we rescale the data to a standardized scale, making the distance measures comparable.

There can be two ways in which the data can be rescaled:

- Min-Max Normalization – The data is rescaled to a 0-1 scale
- Standardization – The data is assumed to be normal and scaled to have a mean of 0 and a standard deviation of 1

The normalization method we chose is min-max normalization.

Min_Max Normalization

In this normalization approach we bring all numeric columns to the range of 0 and 1 with 0 being the lowest value in the column and 1 being the highest value in the column. All other values are normalized based on the following formula:

```
Yi = [Xi - min(X)]/[max(X) - min(X)]
```

```
normalize <- function(x){
  return ((x - min(x))/(max(x) - min(x)))}

idx <- sapply(cust_raw_fin_1, class) == "numeric"

cust_raw_fin_1[,idx] <- sapply(cust_raw_fin_1[,idx], normalize)
```

Post normalization, we have to choose a clustering method and evaluate its performance. There are various partitioning clustering methods available, and we start with the k-means algorithm.

k-means Assumptions and Limitations

- Can create clusters with a specific shape only – Since we have no idea of how the actual clusters will look like, we can assume that the clusters we obtain out of the algorithm are spherical in shape as we use the Euclidian distance measure
- Can work with numerical data only – Our dataset has only numerical clusters, and hence, there is no problem
- The number of clusters (k) needs to be specified before clustering – We will evaluate the clustering performance and choose the clusters based on the results
- Highly sensitive to outliers – Our data has been treated for outliers. Therefore, there would be no impact of outliers
- Cannot capture hierarchical structure – Since, we have not observed any significant results out of hierarchical clustering, we can infer that there is no hierarchical structure
- Hard Clustering – The customers are clustered into one group and one group only. It may be possible that a customer might belong to two different groups when his travel habits differ. But, given our original assumption that the behaviour is stable for the period under consideration, we can neglect this for the scope of this analysis. This assumption could be re-evaluated and restested in the next phase of the segmentation
- Convergence to local minima - k-means could converge to local minima instead of the global minima. The convergence should be evaluated by running multiple instances to identify whether similar results are being obtained across runs

The first step in the k-means algorithm is in choosing the value of k. To identify the value of k, we evaluate the clustering algorithm for different values of k and choose a k depending on the cluster performance.

Evaluating Clustering Performance

The clustering performance depends on the number of clusters we choose. The clusters formed should be such that there is high similarity within a cluster and low similarity between the clusters.

We are looking at the two metrics to evaluate that the clustering performance:

- SSE (Sum of Squared Errors) – SSE captures the sum of squared distance between each point and its centroid. Therefore, lower the SSE, higher the similarity between the point and its cluster
- Silhoutte Coefficient – SC is an alternative metric for cluster performance evaluation that is calculated based on the distance of a point to its cluster centroid and the nearest point outside of the cluster

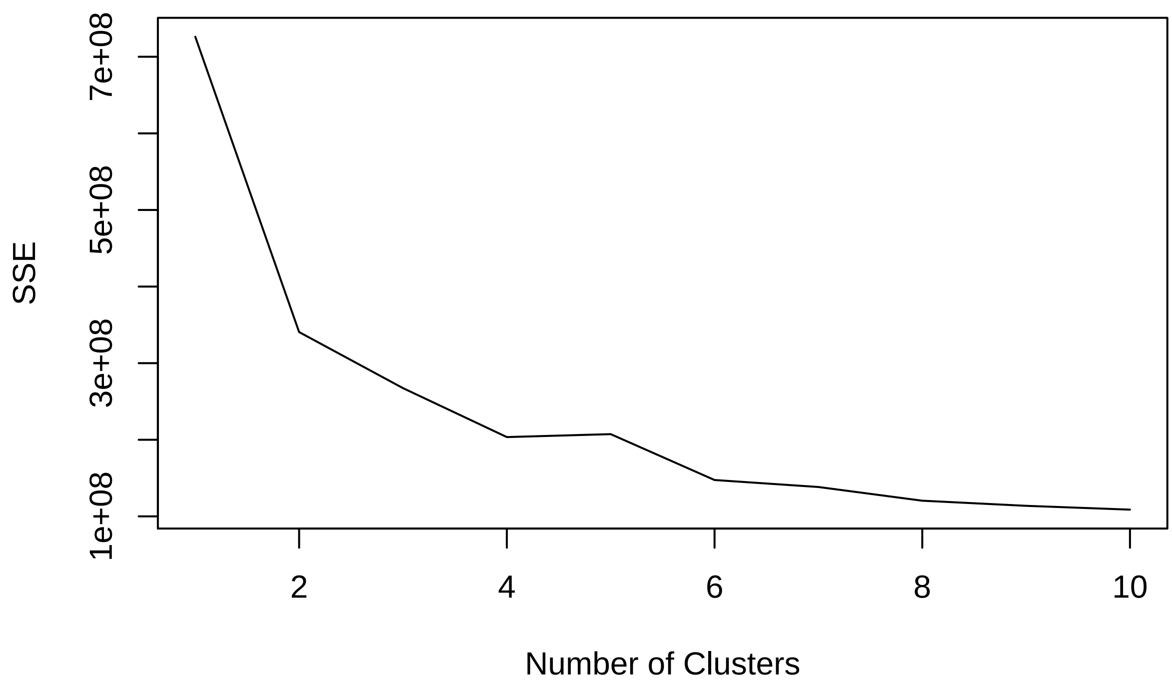
The performance we observed from the Silhouette coefficient was similar to the SSE.

Method 1: Elbow curve

```
SSE_curve <- c()
for (n in 1:10) {
  k_cluster <- kmeans(cust_raw_fin_1[2:21], n)
  print(k_cluster$withinss)
  sse <- sum(k_cluster$withinss)
  SSE_curve[n] <- sse
}
```

```
## [1] 726146699
## [1] 202466278 138119103
## [1]  87906272  42420049 137097988
## [1] 38840168 59910606 41390304 63357585
## [1]  20337324 104490865  39683491  22836914  19977230
## [1] 20335254 20621561 38015928 29748864 19587743 19158371
## [1] 16305698  9746408 18604826 14938401 37699958 27350464 13674272
## [1] 16320672 16617494 14523403 13580773 12405235 13803291 18276989 14899781
## [1] 11876259 15712178 14162057 17868446 14773340 10841919 11916497 10453065
## [9]  6163722
##  [1] 17368562 11476497 11670475  6457741 11911972  7150631 15239721
##  [8]  9119590  8165079 10224346
```

```
plot(1:10, SSE_curve, type="l", xlab="Number of Clusters", ylab="SSE")
```



From the plot, we can see that the for K = 4 the SSE drop is steep and after K = 4 the SSE is almost constant.

Method 2: Silhoutte Coefficient

For the calculation of the Silhouette Coefficient, we need to sample the dataset because computing the Silhoutte coefficient on the entire dataset is computationally tough. Therefore, we sampled 10000 records from the data, and based on the Silhoutte coefficient, generalized the K value for the entire dataset.

We observed that the k value recommended by the Silhouette Coefficient is the same as SSE.

Creating Clusters

Applying the K-means algorithm on the transformed and normalized data with the number of clusters as 4.

```
set.seed(123)
k_cluster <- kmeans(cust_raw_fin_1[2:21], 4, nstart=25, iter.max=1000)
```

Cluster Profiling

Customer segments provide clear information with respect to which customers fall under which segment. This understanding is crucial and will be leveraged for decision making. The output of a clustering algorithm doesn't explain what each cluster comprises of. If and only if the cluster composition is explained do the mathematically-derived clusters become business-consumable customer segments.

After obtaining the clusters, it is imperative that we understand what observations fall under each cluster. This helps us in understanding the patterns that make up the cluster. Cluster profiling is the method by which we try to explain the similarity within clusters and identify patterns that make up the cluster.

Mapping clusters and raw data

The clusters identified are first mapped to the original dataset to identify what set of customers make up each cluster.

```
cust_raw_fin_1$cluster_no <- k_cluster$cluster

# Merging the original dataset to get the cluster details
cluster_data <- merge(sun_fin_out, cust_raw_fin_1[,c("pkey","cluster_no")])
```

Cluster wise seasonality

Rationale

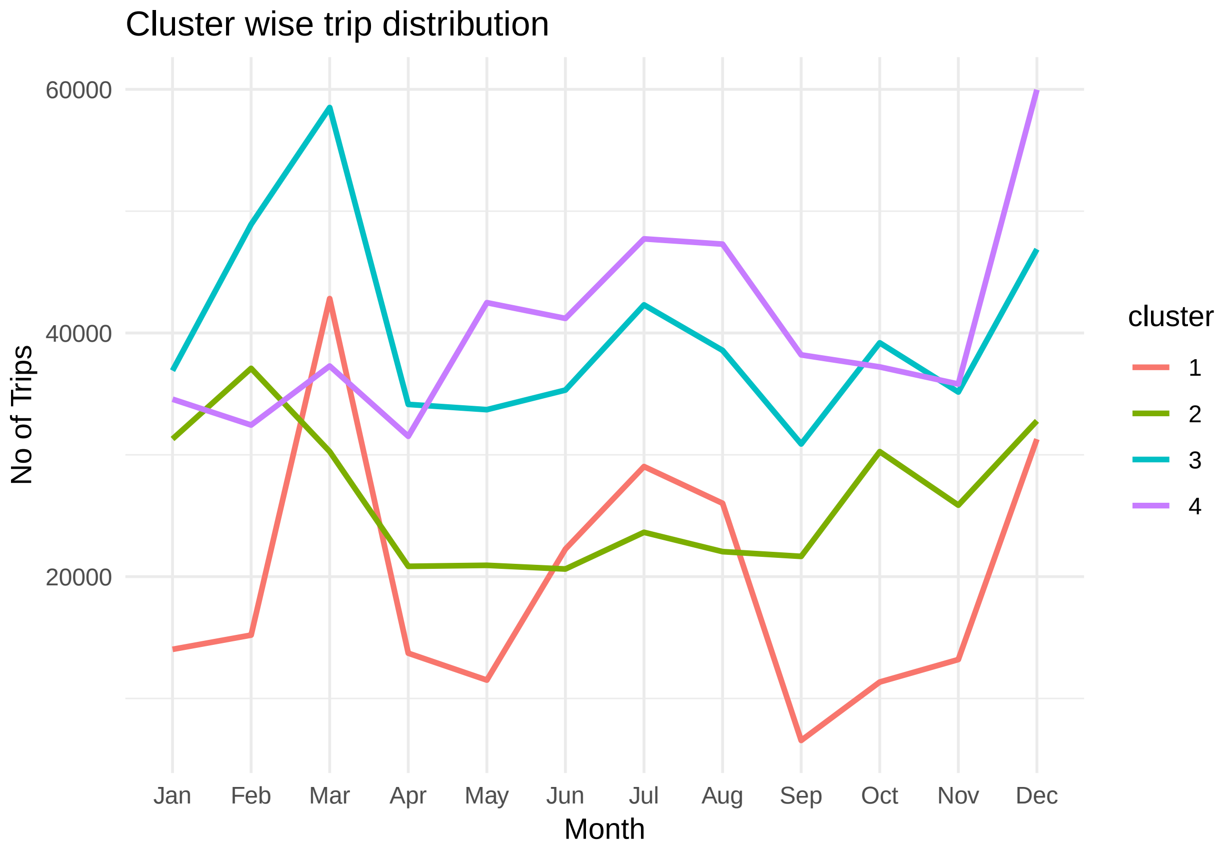
we tried to understand if clusters travel differently through out the year. Whether individual clusters have peak trave seasons and do clusters vary between each other

```
#month wise line graph cluster wise
cluster_data$ServiceStartDate <- lubridate::date(cluster_data$ServiceStartDate)
cluster_data_msp <- cluster_data %>% filter(ServiceStartCity %in% "MSP")

date_cluster_dataset <- cluster_data_msp %>%
  group_by(month=lubridate::month(ServiceStartDate),cluster_no) %>%
  summarise(count = n())
date_cluster_dataset$month_abb <- month.abb[date_cluster_dataset$month]
date_cluster_dataset$month_abb_fac = factor(date_cluster_dataset$month_abb,
                                            levels = month.abb)
date_cluster_dataset$cluster <- factor(date_cluster_dataset$cluster_no)

plotC <- ggplot(date_cluster_dataset,
                aes(x=month_abb_fac, y=count, group=factor(cluster_no)))
```

```r
plotC + geom_line(data=date_cluster_dataset,
                  aes(x=month_abb_fac, y=count, group=cluster, color= cluster),size=1) +
  labs(title = "Cluster wise trip distribution", x = "Month",y = "No of Trips")  +
  theme_minimal()
```



Interpretation

The above chart shows the number of trips for each cluster. + Cluster 1, kids and cluster 3, familes have peaks in March and June-August which are majorly the school holiday season + Cluster 4 seems to not have any peak travel time except for the peaks observed along with other clusters + Cluster 2 seems to have a peak during quarter 3

Assumptions and Conclusion

The idea that the groups travel differentl led us to investigate each cluster seperately to understand their specific characteristics and that is what we would explore in the couple of steps We assumed that the travel patterns observed in this dataset can be generalized to our customers

Loyal elderly vacationers

Elderly vacationers book on an average 67 days in advance

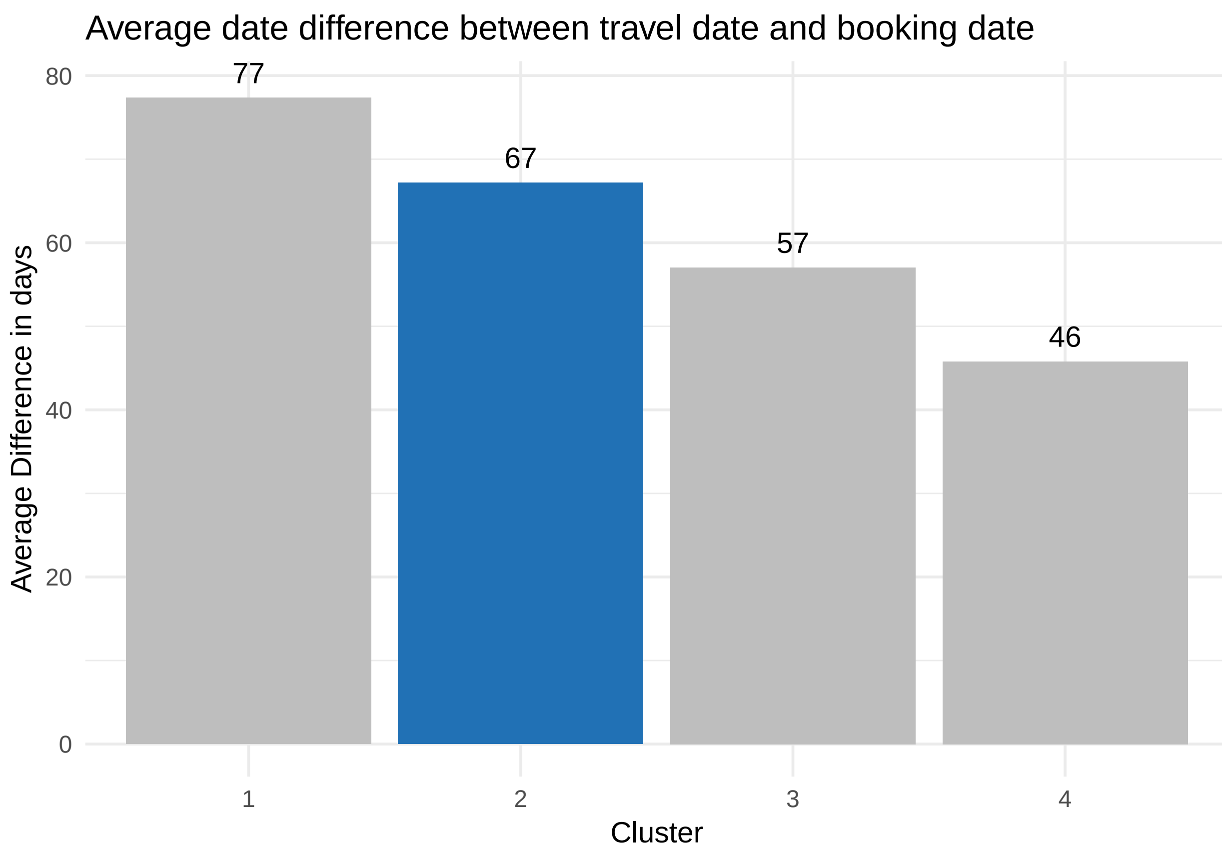
```r
cluster_data$diff_in_days =
  as.numeric(lubridate::date(cluster_data$ServiceStartDate) -
             lubridate::date(cluster_data$PNRCreateDate))

outpt <- cluster_data %>%
  group_by(cluster_no) %>%
```

```r
  summarise(average = mean(diff_in_days))
outpt$average<-as.numeric(outpt$average)
outpt$cluster_no <- factor(outpt$cluster_no)
outpt <- outpt %>%
  mutate(ToHighlight = ifelse(cluster_no %in% c(2), "yes", "no"))



ggplot(data = outpt,aes(x = cluster_no, y = average,fill = ToHighlight)) +
  geom_bar(stat = "identity") +
  scale_fill_manual( values = c( "yes"="#2171b5", "no"="gray" ), guide = FALSE ) +
  geom_text(aes(y = average,label = round(average, 0)), nudge_y = 0.5,vjust = -0.5) +
  labs(title = "Average date difference between travel date and booking date",
       x = "Cluster",
       y = "Average Difference in days") +
  theme_minimal()
```



Interpretation Cluster 2 who are mostly seniors with median age of 75 tend to book early as highlighted by the blue bar in the above graph

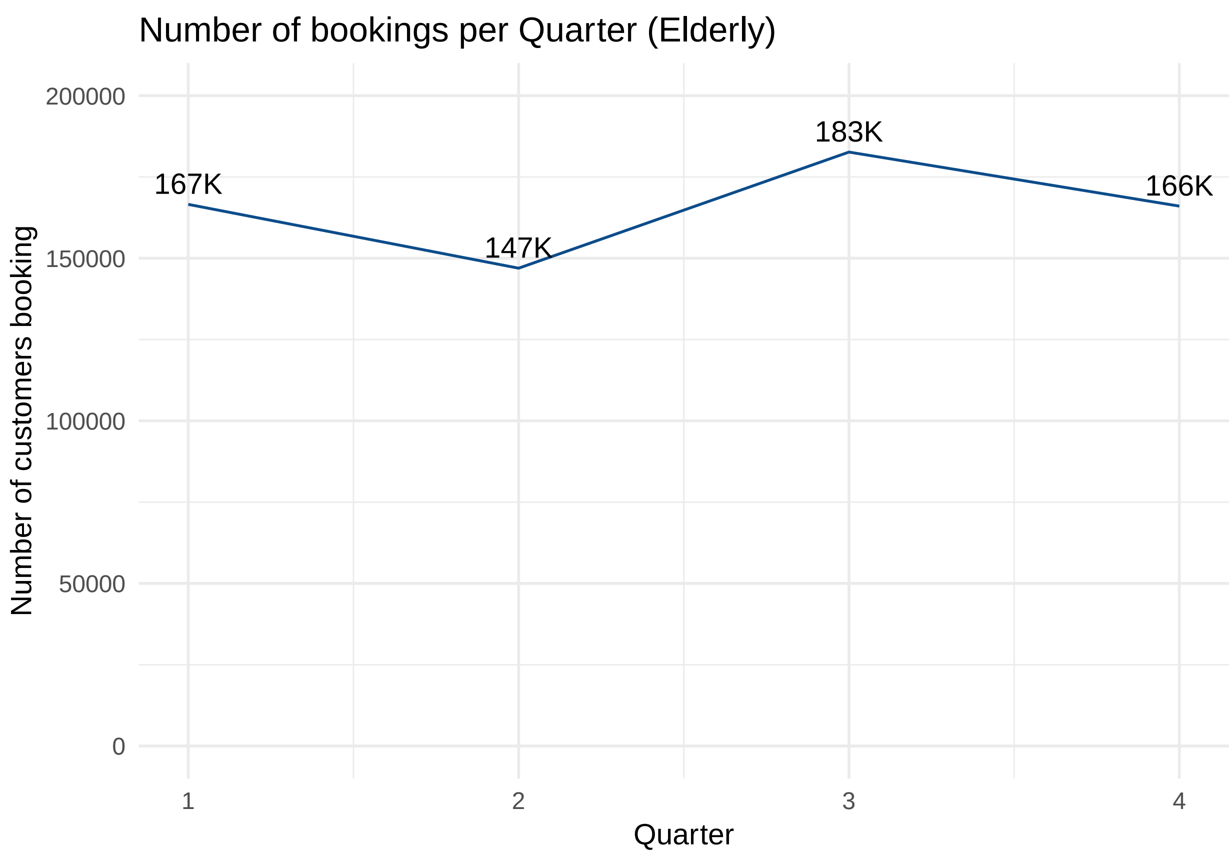
```r
#number of elderly customers booking quarter
##Elderly vacationers book in the third quarter for travel in holiday season
cluster_data$PNRCreateDate <- lubridate::date(cluster_data$PNRCreateDate)
cluster_data_qtr <-cluster_data
cluster_data_qtr$qtr <- quarter(cluster_data_qtr$PNRCreateDate)
cluster_data_qtr <- cluster_data_qtr %>% filter(cluster_no == 2)
```

```r
cluster_data_qtr$qtr <- as.numeric(cluster_data_qtr$qtr)
cluster_data_qtr <- cluster_data_qtr %>%
  group_by(qtr) %>%
  summarise(count = n())


library(ggrepel)
```

```
## Warning: package 'ggrepel' was built under R version 3.5.1
```

```r
ggplot(data = cluster_data_qtr) +
  aes(x = qtr, y = count) +
  geom_line(color = "#0c4c8a") +
  geom_text_repel(aes(y = count,
                      label = paste(round(count/1000), "K", sep = '')),
                  nudge_y = 0.5,vjust = -0.5) +
  ylim(0,200000) +
    labs(title = "Number of bookings per Quarter (Elderly)",
       x = "Quarter",
       y = "Number of customers booking") +
  theme_minimal()
```



Interpretation

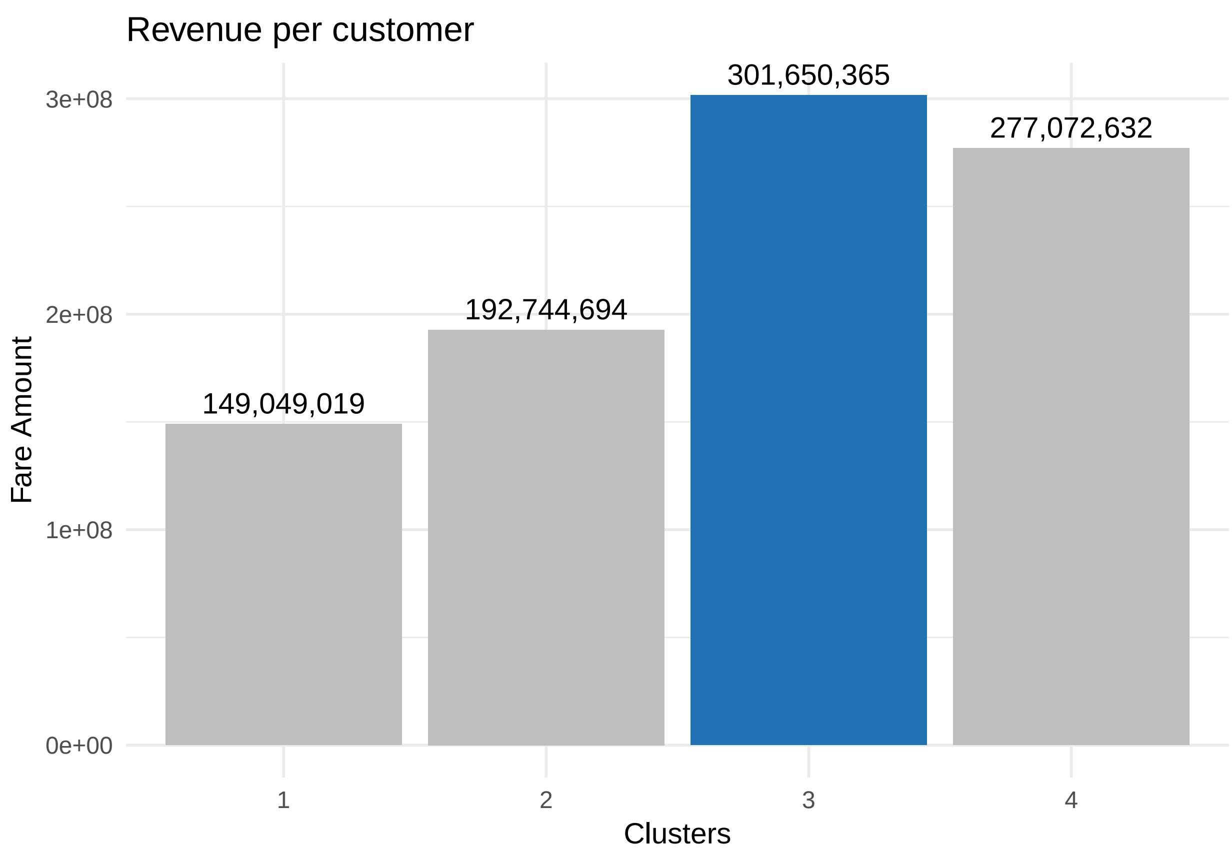
we also observe that peak booking time for the cluster 3, who are elderly is in the third quarter

Loyal family travelers

```r
#BaseFareAmt cluster wise
cluster_data_clust_grp <- cluster_data %>%
  group_by(cluster_no) %>%
  summarise(fare_sum = sum(BaseFareAmt))
cluster_data_clust_grp$cluster_no <- factor(cluster_data_clust_grp$cluster_no)

cluster_data_clust_grp <- cluster_data_clust_grp %>%
  mutate(ToHighlight = ifelse(cluster_no %in% c(3), "yes", "no"))

ggplot(data = cluster_data_clust_grp) +
  aes(x = cluster_no, y = fare_sum,fill = ToHighlight) +
  geom_bar(stat = "identity") +
  scale_fill_manual( values = c( "yes"="#2171b5", "no"="gray" ), guide = FALSE ) +
  geom_text(aes(y = fare_sum,label = prettyNum(fare_sum, big.mark = ',',
                                               scientific = FALSE)),
            nudge_y = 0.5,vjust = -0.5) +
  labs(title = "Revenue per customer",
       x = "Clusters",
       y = "Fare Amount") +
  theme_minimal()
```



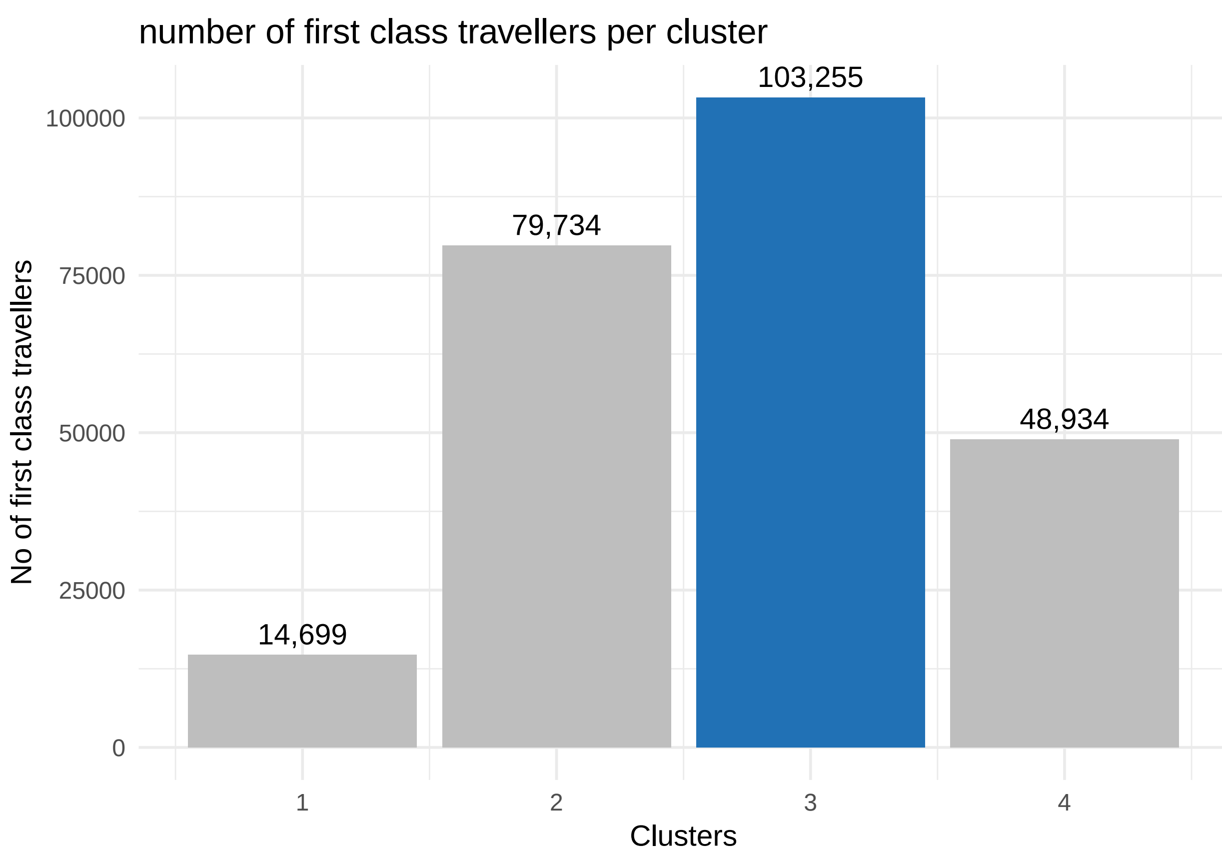
Interpretation It can be seen from the above graph that the cluster 3 generates most revenue.

First Class travellers

```
firstclass <- cluster_data %>%
  filter(TrvldClassOfService %in% c("First Class","Discount First Class")) %>%
  group_by(cluster_no) %>%
  summarise(count = n())

firstclass <- firstclass %>%
  mutate(ToHighlight = ifelse(cluster_no %in% c(3), "yes", "no"))


ggplot(data = firstclass) +
  aes(x = cluster_no, y = count,fill = ToHighlight) +
  geom_bar(stat = "identity") +
  scale_fill_manual( values = c( "yes"="#2171b5", "no"="gray" ), guide = FALSE ) +
  geom_text(aes(y = count,label = prettyNum(count, big.mark = ',',
                                            scientific = FALSE)),
            nudge_y = 0.5,vjust = -0.5) +
  labs(title = "number of first class travellers per cluster",
       x = "Clusters",
       y = "No of first class travellers") +
  theme_minimal()
```



Interpretation

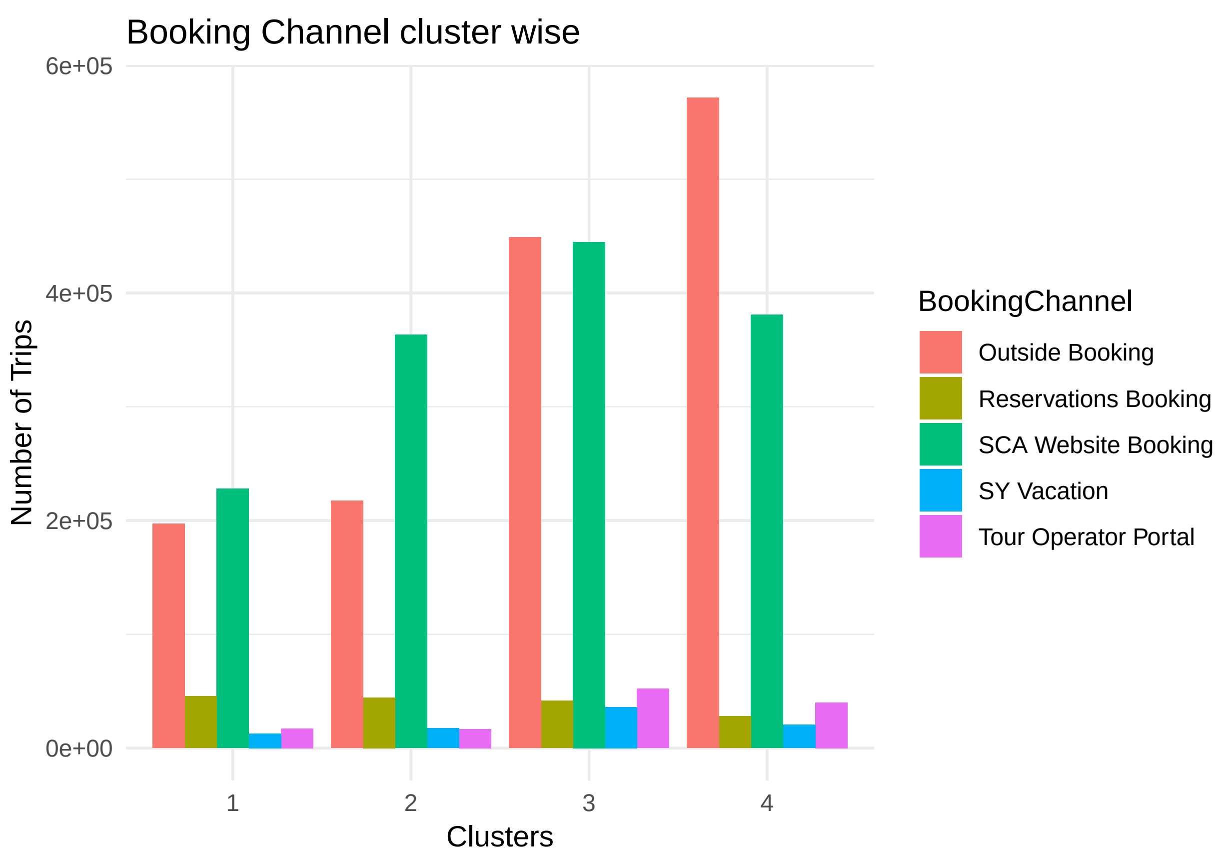
We also see that cluster 3 travels most in first class when compared to other segments which may be the reason for their contribution to revenue

```r
bkchnl<-cluster_data %>%
  group_by(cluster_no,BookingChannel) %>%
  summarise(count = n()) %>%
  arrange(desc(count))

bkchnl <- bkchnl %>%
  filter(BookingChannel %in%
           c("Outside Booking","SCA Website Booking",
             "Reservations Booking","Tour Operator Portal",
             "SY Vacation"))


bkchnl$cluster_no <- as.factor(bkchnl$cluster_no)


ggplot(data = bkchnl) +
  aes(x = cluster_no, fill = BookingChannel, weight = count) +
  geom_bar(position = "dodge") +
  labs(title = "Booking Channel cluster wise",
       x = "Clusters",
       y = "Number of Trips") +
  theme_minimal()
```

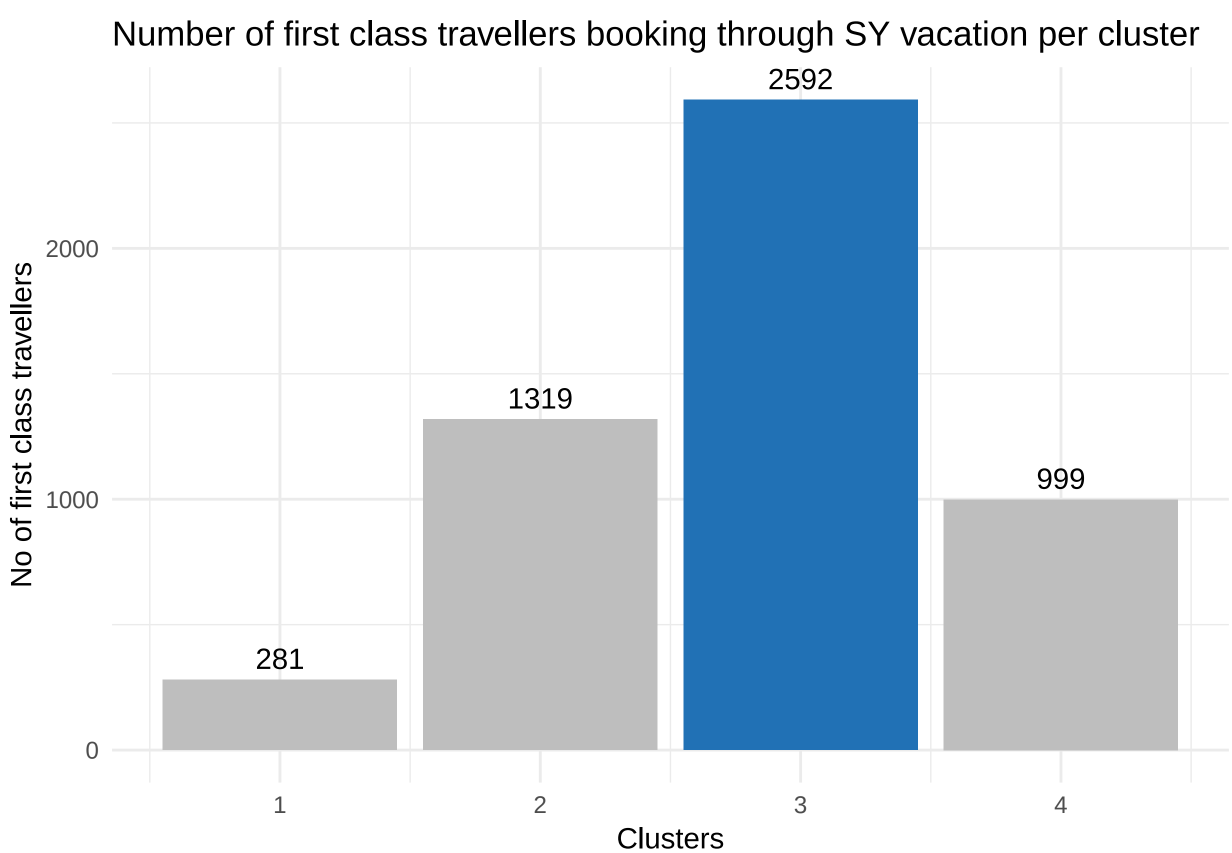


Interpretation Cluster 4 has very high number of outside bookings (indicated in red) when comapred to other clusters which supports our intuition that they might be business travelers.

```r
bkchnl<-cluster_data %>%
  filter(TrvldClassOfService %in% c("First Class","Discount First Class")) %>%
  filter(BookingChannel == 'SY Vacation') %>%
  group_by(cluster_no) %>%
  summarise(count = n())

bkchnl <- bkchnl %>%
  mutate( ToHighlight = ifelse(cluster_no %in% c(3), "yes", "no"))

ggplot(data = bkchnl) +
  aes(x = cluster_no, y = count,fill = ToHighlight) +
  geom_bar(stat = "identity") +
  scale_fill_manual( values = c( "yes"="#2171b5", "no"="gray" ), guide = FALSE ) +
  geom_text(aes(y = count,label = count), nudge_y = 0.5,vjust = -0.5) +
  labs(title = "Number of first class travellers booking through SY vacation per cluster",
       x = "Clusters",
       y = "No of first class travellers") +
  theme_minimal()
```



Interpretation

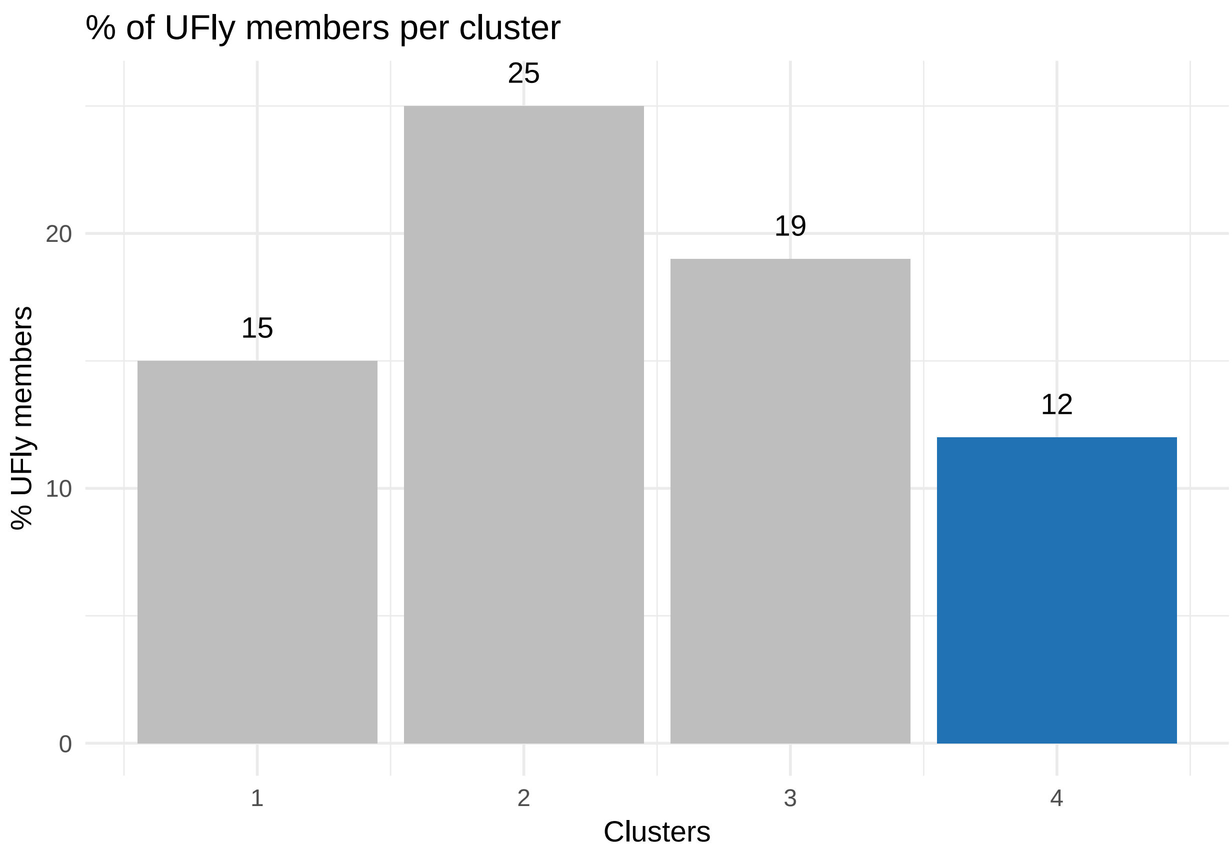
Cluster 3 have opted for highest number of SY vacation packages and we feel that since this cluster is price in-sensitive we are recommending Sun Country to target cluster 3 customers with SY vacation packages

```r
flymbr <- cluster_data %>%
  group_by(cluster_no,UflyMemberStatus) %>%
  summarise(count = n())
```

```r
out1 <- sqldf::sqldf('select cluster_no,
                     cast(sum(case when uflymemberstatus like "%Elite%"
                     or uflymemberstatus like "%Standard%"
                     then count else 0 end)*10000/sum(count) as float)/100
                     as percentage
                     from flymbr
                     group by cluster_no')
```

```
## Warning: package 'RSQLite' was built under R version 3.5.1
```

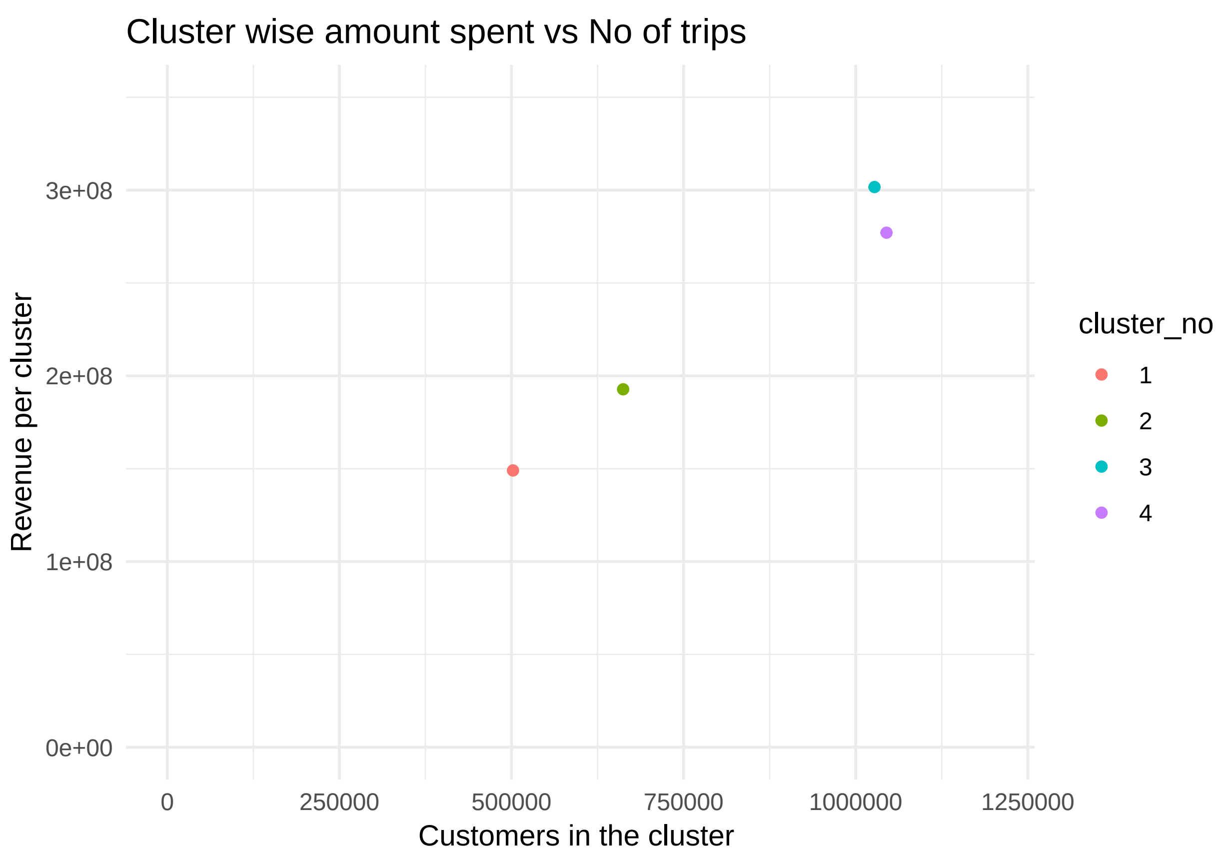
```r
ggplot(data = out1) +
  aes(x = cluster_no, y = percentage,fill = ToHighlight) +
  geom_bar(stat = "identity") +
  scale_fill_manual( values = c( "yes"="#2171b5", "no"="gray"), guide = FALSE) +
  geom_text(aes(y = percentage,label = percentage), nudge_y = 0.5,vjust = -0.5) +
  labs(title = "% of UFly members per cluster",
       x = "Clusters",
       y = "% UFly members") +
  theme_minimal()
```



Interpretation

In this we can see that the cluster 4 has lowest number of UFly enrollment when compared to other segments which is the reason Sun Country should focus more on this segment and try to increase enrollments for the same.

```r
tottrvl <- cluster_data %>% group_by(cluster_no) %>% summarise(trvlno = n())
totval <- cluster_data %>% group_by(cluster_no) %>% summarise(trvlfare = sum(BaseFareAmt))
mergedtrvlfare <- merge(tottrvl,totval,by='cluster_no')
mergedtrvlfare$cluster_no <- as.factor(mergedtrvlfare$cluster_no)
ggplot(data = mergedtrvlfare) +
  aes(x = trvlno, y = trvlfare, color = cluster_no) +
  geom_point() + xlim(0,1200000) + ylim(0,350000000) +
  labs(title = "Cluster wise amount spent vs No of trips ",
       x = "Customers in the cluster ",
       y = "Revenue per cluster") +
  theme_minimal()
```



Interpretation

In the above plot we can observe that cluster 4 who are young business travelers contribute to the highest number of trips.

Conclusion

Four behavioural segments exist in Sun Country's user base - Kids, Loyal elderly vacationers, Loyal family vacationers and Young business travelers. Characteristic of these user segments can be leveraged to offer products and services that suit their needs.

- Families with kids should be targeted with holiday packages customized for kids during the spring and summer breaks
- Target elderly vacationers at least 2 months before the holiday season

- Loyal family travelers present an opportunity to upsell ancillary services bundled with SY Vacation packages
- Turn young business travelers loyal through UFly membership enrollments