CUSTOMER ANALYTICS - POKEMON GO CASE

SUMMARY REPORT

Full code is available in the Pokemon_GO_Case_Final_Script.R script

A team group project from the MSc of Data Analytics & AI

Table of Contents :

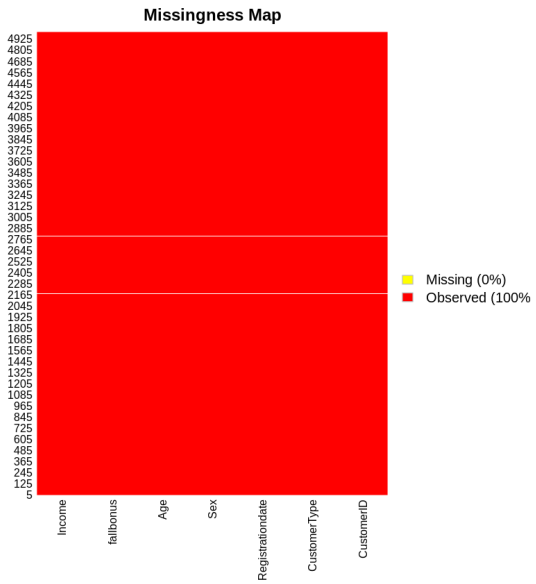
Question 1: Creation of a basetable + profiling

Loading data files and inspecting the customer database:

Code:

```
customerdata = read.csv("customerdata.csv")
fallfintrx = read.csv("fallfintrx.csv")
fallsesstrx = read.csv("fallsesstrx.csv")
summerfintrx = read.csv("summerfintrx.csv")
summersesstrx = read.csv("summersesstrx.csv")
library(Amelia)
#inspect customer database
missmap(customerdata,col=c("yellow","red"))
#no missing data
```

Comments:



We see no missing data in the customer database.

Getting active customers ID during the summer 2018 and retrieving all info associated to them from customer database (demographics and other)

Code:

```
library(dplyr)
#get active customers ID during summer
```

```
active_customer = merge(customerdata,summersesstrx,by="CustomerID",all=FALSE)
active_customer = active_customer %>% distinct(CustomerID)

#retrieve all info of the customerdatabase on them
active_customer = merge(active_customer,customerdata,by="CustomerID",all=FALSE)
head(active_customer,5)
```

Comments:
4,703 players were considered as active ones during summer 2018.

Defining Financial variables

Code:

```
## VARIABLES BASED ON SUMMER FINANCIAL DATASET ##
#some data prep for the summer financial database
library(Amelia)
#inspect summer financial transaction database
missmap(summerfintrx,col=c("yellow","red"))  #no missing data

endtime = as.Date('31/08/2018',"%d/%m/%Y")
summerfintrx$Date = as.Date(summerfintrx$Date,"%Y-%m-%d") #convert data tyoe
#defining function to convert productid bought by customer to the amount spent by
customer
to_amount = function(x) {
 if (x == 1) y=2.99 else {
   if (x ==2 )y = 4.99 else {
     if (x==3) y = 9.99 else {
       if(x==4) y=25 else{
         y=99
       }
     }
   }
 }
 y
}
#creating the column Amount
summerfintrx$Amount  = sapply(summerfintrx$ProductID, to_amount)

#compute RFM metrics from summer financial database
RFM_finance = summerfintrx%>%
 group_by(CustomerID)%>%
 summarise(frequency_fin=n(),#create frequency and recency for them
           recency_fin=as.numeric(endtime-max(Date)),
           monetaryvalue=sum(Amount)) %>%
 ungroup()
head(RFM_finance,5)
```

Comments : we computed 3 variables based on the summer financial database :

- frequency_fin ==> number of transactions by customer
- recency_fin ==> number of days since last customer transaction
- monetaryvalue ==> total amount spent by customer

Defining Gaming variables

Code:

```
## VARIABLES BASED ON THE SUMMER PLAY SESSION DATABASE ##

#some data prep for the summer financial database
library(Amelia)
#inspect summer financial transaction database
missmap(summersesstrx,col=c("yellow","red"))  #no missing data

endtime = as.Date('31/08/2018',"%d/%m/%Y")
summersesstrx$Date = as.Date(summersesstrx$Date,"%Y-%m-%d") #convert data type

#compute play metrics from summer playing sessions database
metrics_play = summersesstrx%>%
 group_by(CustomerID)%>%
 summarise(frequency_play=n(),#create frequency and recency for them
           recency_play=as.numeric(endtime-max(Date)),
           Cum_Distance=sum(Distance),
           Cum_Gaming_time = sum(Duration),
           Cum_Social = sum(Social),
           Average_Distance=mean(Distance),
           Average_Social = mean(Social),
           Average_Pokemon = mean(Pokemons),
           Average_Gaming_Time = mean(Duration))%>%
 ungroup()

head(metrics_play,5)
```

Comments: we computed 8 variables based on the summer play sessions database

We can add some more for determining the general profile of active customers

- frequency_play: number of playing sessions by customers
- recency_play : number of days since last playing session for a customer
- Cum_Distance : total distances by customer during the summer (in kms)
- Cum_Gaming_time : total gaming time by customer
- Cum_Social : total number of social interactions by customer
- Average_Distance : average distance by customer during a gaming session (in kms)
- Average_Gaming_time : average playing time by customer during a gaming session
- Average_Social : average number of social interactions by customer during a playing sessions

Get the Basetable containing the active players described by the financial variables and gaming variables

Code:

```
## GET BASETABLE ##
#get all the metrics computed for our summer active customers
basetable  = merge(active_customer,RFM_finance,by="CustomerID",all.x=TRUE)
basetable = merge(basetable, metrics_play, by = "CustomerID", all.x=TRUE)
## computing and transforming other variables of the basetable (before analysis)
#function to convert the income variable into a categorical variable
income_categories = function (x) {
 if (x==1) y="Low Income"
 else {
   if (x==2) y="Medium Income"
   else {
     y="High Income"
   }
 }
}
basetable$Income_Categories = sapply(basetable$Income, income_categories)

# computing seniority variable : nb of days since registration date
endtime = as.Date('31/08/2018',"%d/%m/%Y")
basetable$Registrationdate = as.Date(basetable$Registrationdate,"%Y-%m-%d")
#convert data type of registrationdate
basetable$Seniority = as.numeric(endtime-basetable$Registrationdate)
head(basetable,5)
```

Comments:

We got the basetable containing all the active players during the summer, described by RFM (financial) variables, and gaming variables

We also did some data preparation for commenting the basetable, by adding two new variables

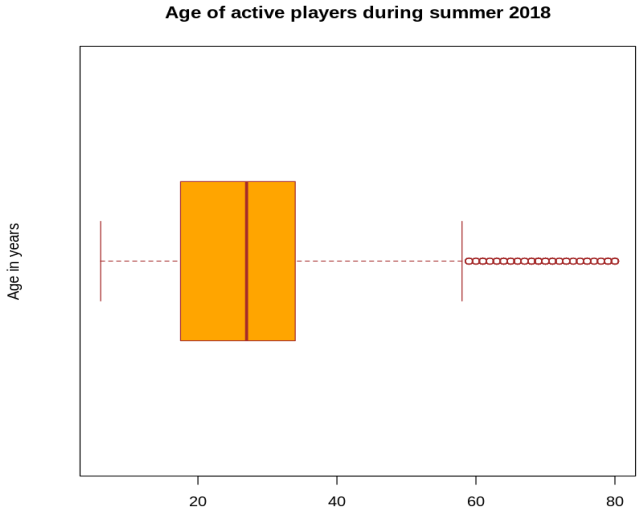
- Seniority : number of days since the active player joined/registered in the game
- Income_Categories : translating the Income variable into a categorical variable

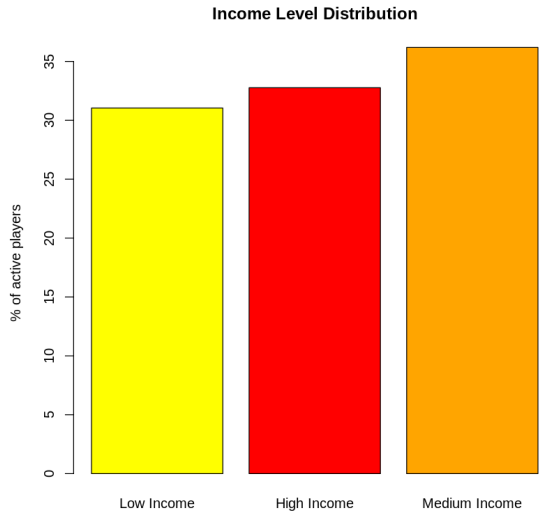
Describing the basetable using demographics variables

Code:

```
### DESCRIBING THE BASETABLE ###

## DEMOGRAPHICS TRENDS ##
dim(basetable)[1]  #4 703 active customers during summer
(sum(basetable$Sex)/dim(basetable)[1]) * 100  # 39.5 % of them are female
median(basetable$Age)  #=27,  50% of active customers are less than 27 years old
```

```
quantile(basetable$Age, 0.75) #=34, 75% of active customers are less than 34
years old

boxplot(basetable$Age,
main = "Age of active players during summer 2018",
ylab = "Age in years",
col = "orange",
border = "brown",
horizontal = TRUE
)
#income level analysis
counts_income = sort(table(basetable$Income_Categories)/dim(basetable)[1]*100)
#(Low=31%, High=33%, Medium=36%)
barplot(counts_income, main="Income Level Distribution", ylab= "% of active
players", col= c("yellow","red","orange"))
#there is no majority income level class represented in our active players :
players with lowest income are as many as players with highest income
```





Comments:

Majority of active players are males(60%), the median age is at 27, but 75% of the players are less than 34 years old.

Regarding the income level distribution, no majority class. Income does not seem to play a big role for describing active players.

Describing the basetable using financial variables

Code:

```
##FINANCIAL TRENDS##
library(Amelia)
```

```
#inspect missing value on the basetable
missmap(basetable,col=c("yellow","red"))

# removing the NA in the basetable : keeping active players who made at least one
transaction == the pruchasers
basetable_fin = basetable[!is.na(basetable$frequency_fin),]

(dim(basetable_fin)[1] / dim(basetable)[1])*100 #only 36 % of active players
have bought something on the app
mean(basetable_fin$monetaryvalue)  # in average,  players spent 11 euros during
the summer
(sum(basetable_fin$Sex)/dim(basetable_fin)[1]) * 100  #39% of purchasers are
female
median(basetable_fin$Age)  # median age of purchasers is the same as active
players in general

mean(basetable_fin$frequency_fin) # in average, a purchaser made 1.3 transaction
==> does not really make sense, better take a quantile
quantile(basetable_fin$frequency_fin, 0.95) # 95% of purchasers made at most  2
transactions during the summer

mean(basetable_fin$recency_fin) # in average, last transaction for a purchaser
was 56 days ago (if today is the 31/8/2018)
quantile(basetable_fin$recency_fin, 0.05)   #only 5% of the purchasers made a
transaction less than 6 days ago  (if today is the 31/8/2018)
```

Comments:

Only 1 active player out of 3 bought something in the game.

Those purchasers spent 11 euros on average this summer, 39% of them were females.

Although, purchasers are not frequent buyers : 95% of them made at most 2 transactions and only 5% of them made a transaction in last 6 days (supposing it is the 31/8/2018)

Describing the basetable using gaming variables

Code:

```
##GAMING TRENDS##
#Global gaming trends #
quantile(basetable$frequency_play, 0.75)   #=6, 3 out of 4 active players played
at most 6 times during the summer
mean(basetable$recency_play)  #=30, in average, an active  player last connected
a month ago  (distrib is right skewed )
quantile(basetable$Seniority, 0.55)  # = 336,  more than half of the active
players joined the game / registered less than a  year ago (55%)
mean(basetable$Cum_Distance) #=15km an active player walked 15km in average
during the summer while playing the game
```

```
mean(basetable$Cum_Gaming_time) #=338 mins=> 5.6 hours, an active player played PokemonGo 5.6 hours on average this summer
mean(basetable$Cum_Social)  #=4.97,  an active player had a total of 5 social interactions on average while playing this summer
# Gaming session trends #
mean(basetable$Average_Distance) #=3.8km in average, an active gamer can walk 4 km per session
mean(basetable$Average_Gaming_Time) #=59mins , an active player plays on average 1 hour during a gaming session
mean(basetable$Average_Social)  # in average, an active player has 1 social interaction per gaming session
mean(basetable$Average_Pokemon)  # an active player can catch up to 19 pokemons during a gaming session
```

Comments:

We can get a lot of descriptive statistics on a larger scale (gaming trends during the whole summer) and on a smaller scale (gaming trends during a gaming session).

Interesting trends can be that 3 active players out of 4 played at most 6 times during the summer. The average last connection was a month ago (supposing it is the 31/8/2018).

During a gaming session, an average player can walk 4kms, get 1 social interaction and can play up to 1 hour (in average).

Calculating the CLV value for our active customers

Code:

```
basetable$monetaryvalue[is.na(basetable$monetaryvalue)]<-0
active_paying_customer =
merge(summerfintrx,active_customer,by="CustomerID",all = FALSE)
active_paying_customer=active_paying_customer %>% distinct(CustomerID)
#active & paying customers during the summer

paying_fall = merge(customerdata,fallfintrx,by="CustomerID",all=FALSE)
#paying fall customers
paying_fall=paying_fall%>% distinct(CustomerID)
active_paying_customer$still_paying_fall = 0

active_paying_customer$still_paying_fall[active_paying_customer$CustomerID %in% paying_fall$CustomerID]=1 #active & paying customers who are also paying in the fall

avg_retention_rate <- mean(active_paying_customer$still_paying_fall)
```

```
calc_clv<-function(margin,r,d,acquisition,t)
{
  clv<--acquisition
  for(i in 0:t)#attention: start in year 0
  {
    clv<-clv+((r^i)*margin/(1+d)^i)
  }
  return (clv)
}


r=avg_retention_rate;r
d=0;
acquisition=0
# we worked with a block period equal to the summer period (4 months),
so we need to take each year as 3 blocks
# so for x years, t= x*3, assuming a period of 2 years, then t=3*2
t=3*2

clv<-
apply(basetable[,c("monetaryvalue","frequency_fin")],1,function(x)
calc_clv(x[1],avg_retention_rate,d,acquisition,t))
basetable$clv<-clv
mean(clv)
```

Comments:

- We set the monetary value equal to 0 for all non-paying customers.
- Then, we defined a global retention rate as the ratio of the number of paying and active players in both summer and fall over the number of active and paying customers during the summer period (34.65%)

Retention rate= # active & paying players (common in both summer & fall) / # active & paying players(in summer)

- Before the launch of the game, Pokemon was very popular around the globe from trading card games, video games, TV shows, comic books, etc. For that, Niantic relied on the nostalgic aspect of the game and the word-of-mouth, hence, Niantic spent virtually nothing on user acquisition. That is why we assumed an acquisition cost of 0.

(Source: https://blog.amplitude.com/pokemon-go-lost-players-won-game)

- For each customer, we calculated the sum of transactions amount in the summer, so as we are working with a block period equal to the summer period (4 months), we need to take each year as 3 of those periods. Therefore, for x years, the time period t = x*3. We also assumed a period of 2 years to be a reasonable period, then t=3*2.

- We assumed a discount rate of 0.
- The mean CLV obtained is 6.068 : on average, an active player will bring 6 euros to Niantic over its overall gaming experience on PokemonGO.

Question 2 - Lifecycle grids

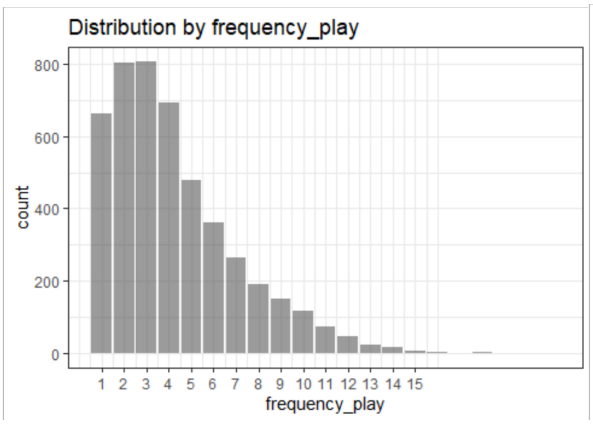
Checking the distribution of "recency" and "frequency", and breaking down the data into segments and groups.

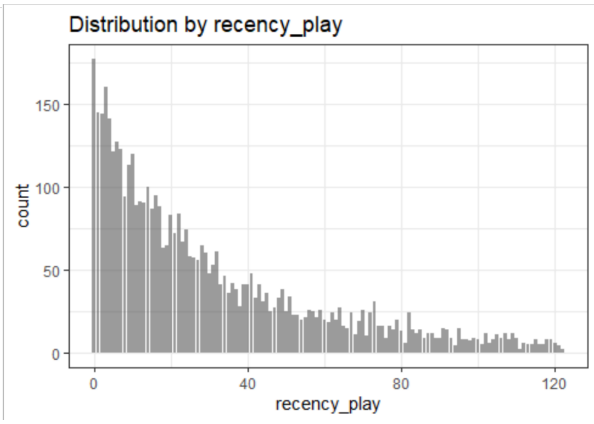
Code:

```
##GAMING TRENDS##
#Exploratory analysis to check distribution of recency and frequency

############# FOR PLAYING DATA ##################
library(ggplot2)
ggplot(basetable, aes(x=frequency_play)) +
  theme_bw() +
  scale_x_continuous(breaks=c(1:10)) +
  geom_bar(alpha=0.6) +
  ggtitle("Distribution by frequency_play")

ggplot(basetable, aes(x=recency_play)) +
  theme_bw() +
  scale_x_continuous(breaks=c(1:10)) +
  geom_bar(alpha=0.6) +
  ggtitle("Distribution by recency_play")

############# FOR FINANCIAL DATA ##################
ggplot(RFM_finance, aes(x=frequency_fin)) +
  theme_bw() +
  scale_x_continuous(breaks=c(1:10)) +
  geom_bar(alpha=0.6) +
  ggtitle("Distribution by frequency_fin")

ggplot(RFM_finance, aes(x=recency_fin)) +
  theme_bw() +
  scale_x_continuous(breaks=c(1:10)) +
  geom_bar(alpha=0.6) +
  ggtitle("Distribution by recency_fin")
#Distributing players into segments
customer.segm <- basetable %>%
  mutate(segm.freq.play=ifelse(between(frequency_play, 1, 1), '1',
                           ifelse(between(frequency_play, 2, 2), '2',
                                  ifelse(between(frequency_play, 3, 3), '3',
                                         ifelse(between(frequency_play, 4, 4),
'4',
                                                ifelse(between(frequency_play, 5,
5), '5', '>5')))))) %>%
  mutate(segm.rec.play=ifelse(between(recency_play, 0, 6), '0-6 days',
```

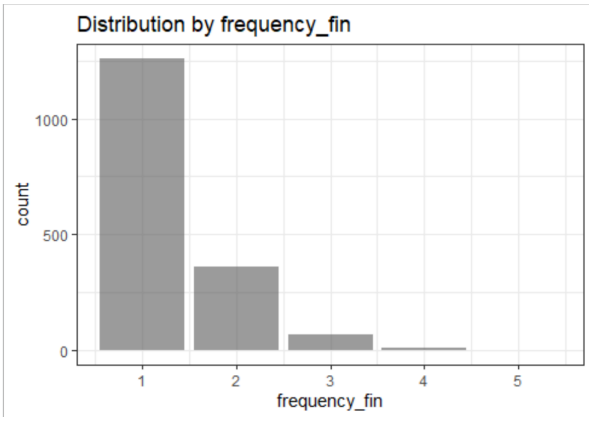
```
                        ifelse(between(recency_play, 7, 13), '7-13 days',
                               ifelse(between(recency_play, 14, 19), '14-19
days',
                                      ifelse(between(recency_play, 20, 45), '20-
45 days',
                                             ifelse(between(recency_play, 46,
80), '46-80 days', '>80 days'))))))

# defining order of boundaries
customer.segm$segm.freq.play <- factor(customer.segm$segm.freq.play,
levels=c('>5', '5', '4', '3', '2', '1'))
customer.segm$segm.rec.play <- factor(customer.segm$segm.rec.play, levels=c('>80
days', '46-80 days', '20-45 days', '14-19 days', '7-13 days', '0-6 days'))
```

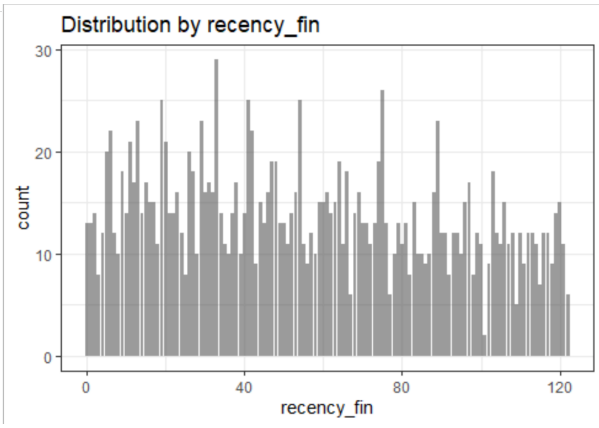
Playing (non-financial) data:





Financial data:





Comments:

This code visualises the distribution of recency and frequency for each of the Playing Sessions (Non-Financial) and Financial data.

After deciding how to segment the basetable, customer.segm is created by using the basetable data and dividing those data to their corresponding segments.

Generating non-financial lifecycle grids

```
#1. Quantity (Number of playing sessions) life cycle plot
lcg <- customer.segm %>%
  group_by(segm.rec.play, segm.freq.play) %>%
  summarise(quantity=n()) %>%
  mutate(player='player') %>%
  ungroup()

ggplot(lcg, aes(x=player, y=quantity, fill=quantity))+
  theme_bw() +
  theme(panel.grid = element_blank())+
  geom_bar(stat='identity', alpha=0.6) +
  geom_text(aes(y=max(quantity)/2, label=quantity), size=4) +
  facet_grid(segm.freq.play ~ segm.rec.play)+
  ggtitle("RF quantity grid")
#2. Gender (Sex) life cycle plot
customer.segm$Sex<-as.factor(customer.segm$Sex)
lcg.sub <- customer.segm %>%
  group_by(Sex, segm.rec.play, segm.freq.play) %>%
  summarise(quantity=n()) %>%
  mutate(player='player') %>%
  ungroup()

ggplot(lcg.sub, aes(x=player, y=quantity, fill=Sex)) +
  theme_bw() +
  scale_fill_brewer(palette='Set1') +
  theme(panel.grid = element_blank())+
  geom_bar(stat='identity', position='fill' , alpha=0.6) +
  facet_grid(segm.freq.play ~ segm.rec.play) +
  ggtitle("LifeCycle Grids by gender (proportion)")

#3. CustomerType life cycle plot
# where 1=walker,2=miscellaneous,3=social raider,4=catcher
customer.segm$CustomerType<-as.factor(customer.segm$CustomerType)
lcg.sub <- customer.segm %>%
  group_by(CustomerType, segm.rec.play, segm.freq.play) %>%
  summarise(quantity=n()) %>%
  mutate(player='player') %>%
  ungroup()

ggplot(lcg.sub, aes(x=player, y=quantity, fill=CustomerType)) +
  theme_bw() +
  theme(panel.grid = element_blank())+
  geom_bar(stat='identity', position='fill' , alpha=0.6) +
  facet_grid(segm.freq.play ~ segm.rec.play) +
  ggtitle("LifeCycle Grids by gender (proportion)")
```

```
#4. Income life cycle plot
lcg.sub <- customer.segm %>%
  group_by(Income_Categories, segm.rec.play, segm.freq.play) %>%
  summarise(quantity=n()) %>%
  mutate(player='player') %>%
  ungroup()

ggplot(lcg.sub, aes(x=player, y=quantity, fill=Income_Categories)) +
  theme_bw() +
  theme(panel.grid = element_blank())+
  geom_bar(stat='identity', position='fill' , alpha=0.6) +
  facet_grid(segm.freq.play ~ segm.rec.play) +
  ggtitle("LifeCycle Grids by Income Categories (proportion)")

#5. Month Cohorts life cycle plot
lcg.sub <- customer.segm %>%
mutate(cohort=as.numeric(factor(format(Registrationdate,format="%Y-%m")))) %>%
#creates a cohort for each month
  group_by(segm.rec.play, segm.freq.play,cohort) %>%
  summarise(quantity=n())

ggplot(lcg.sub, aes(x=cohort, y=quantity,fill=factor(cohort))) +
  theme_bw() +
  theme(panel.grid = element_blank())+
  geom_bar(stat='identity' , alpha=0.6) +
  facet_grid(segm.freq.play ~ segm.rec.play) +
  ggtitle("LifeCycle Grids  by Cohorts")
#6. Age group life cycle grids
lcg.sub <- customer.segm %>%
#Creates Age groups
  mutate(age_group=ifelse(between(Age, 6,12), 'Child',
                          ifelse(between(Age, 13, 18), 'Teen',
                                 ifelse(between(Age, 19, 39), 'Young
Adult',
                                        ifelse(between(Age, 40, 59),
'Middle-Age Adult', 'Senior Adult')))))

lcg.sub$age_group <- factor(lcg.sub$age_group , levels=c('Child',
'Teen', 'Young Adult', 'Middle-Age Adult', 'Senior Adult'))

lcg.sub <- lcg.sub %>%
  group_by(age_group, segm.rec.play, segm.freq.play) %>%
  summarise(quantity=n()) %>%
  mutate(player='player') %>%
  ungroup()
```

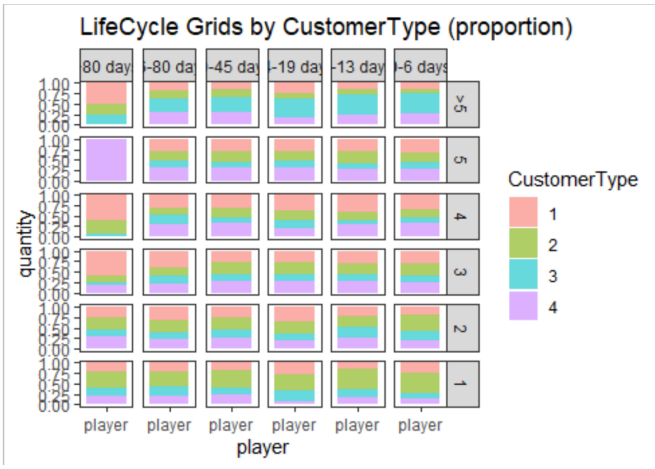
```
ggplot(lcg.sub, aes(x=player, y=quantity, fill=age_group)) +
  theme_bw() +
  scale_fill_brewer(palette='Set1') +
  theme(panel.grid = element_blank())+
  geom_bar(stat='identity', position='fill' , alpha=0.6) +
  facet_grid(segm.freq.play ~ segm.rec.play) +
  ggtitle("LifeCycle Grids by age group (proportion)")

ggplot(lcg.sub, aes(x=age_group, y=quantity, fill=age_group)) +
  theme_bw() +
  scale_fill_brewer(palette='Set1') +
  theme(panel.grid = element_blank(),axis.text.x=element_blank())+
  geom_bar(stat='identity' , alpha=0.6) +
  facet_grid(segm.freq.play ~ segm.rec.play) +
  ggtitle("LifeCycle Grids by age group ")
```

Comments:

The lifecycle grids are being generated using the customer.segm dataframe.

We generated several Life Cycle Grids and selected the most relevant ones:

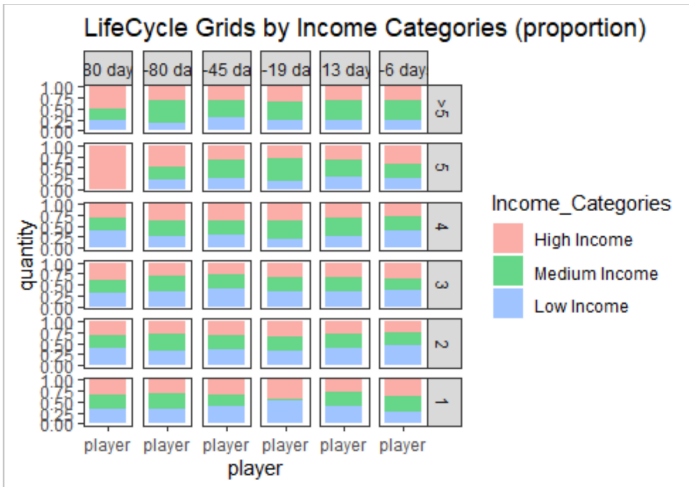
Life Cycle Grids per Customer Type



We built this grid based on the customer segmentation provided in the raw data (1 = Walker, 2 = Miscellaneous, 3 = Social raider, 4 = Catcher). The main observation is that players who play the most tend to use Pokemon Go for a specific utilization (to exercise and to have social interactions). Most of them belonged to the walker and social raiders segments. On the opposite, the new customers (and the one-time customers) are using the app without showing any preferences. This shows that in order to help customers to move in the right-

upper zone (where the best customers stand), they should use the app in a more specific way.

Life Cycle Grids by Income Categories

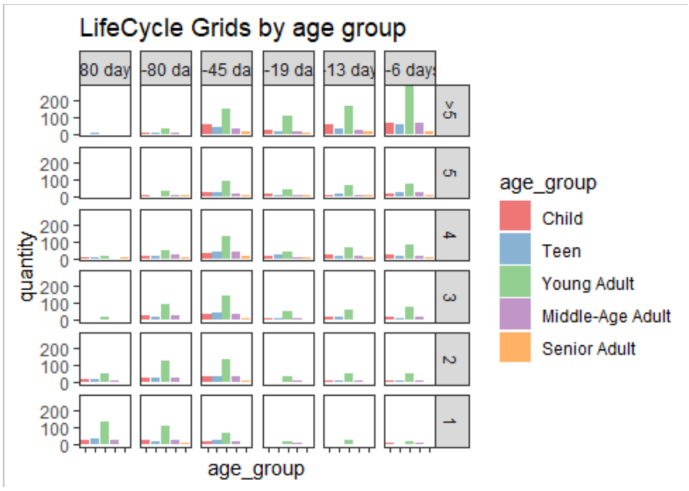


In order to adjust the Marketing strategy and to target the right consumer profile, we built a Life Cycle grids by Income categories. Even if the repartition seems homogenous, we see a change in the customer profile. Most of the former best consumers (upper-left zone) had high income, whereas the current best customers belong to the medium income group. Besides, more than half of the new customers have low income.

Even if this repartition is not obvious, it needs to be taken into account when defining the new marketing strategy.

Life Cycle Grids by Age Group

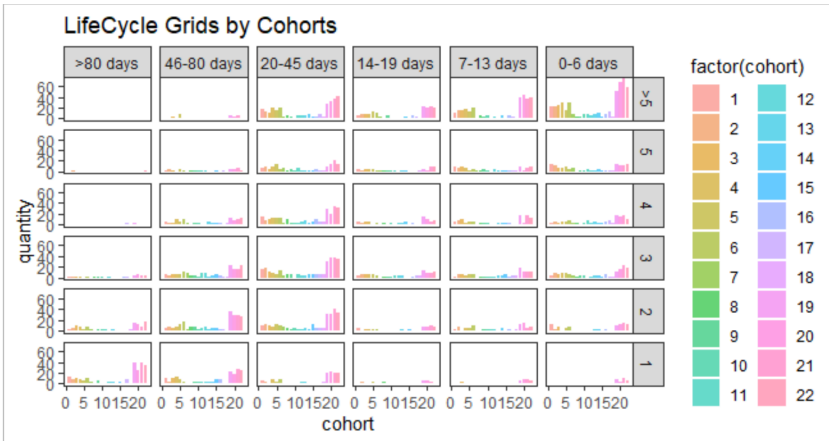
We split the players into 5 age groups (6-12 years: Child | 13-18 years: Teen | 19-39 years: Young Adult | 40-59: Middle-Age Adult | 60+: Senior Adult).



At first view, the app is mostly used by young adults, followed (by far) by children and teenagers. However, the Middle-Age Adult group should not be forgotten as a significant proportion of new customers belongs to that category. The challenge will be to target both

young and older customers, as they do not have the same reactions to marketing campaigns.

Life Cycle Grids by Cohorts



Here we splitted the customer into cohorts, where each cohort represents the month-year. For example, someone who joined the game on 2017-02-30, will belong to the cohort of February-2017.

The registration dates of the active summer customers in our basetable span from 2016-07-01 (as 1st of July of 2017) to 2018-04-30 (as 30 April of 2018). Hence we have 22 cohorts (Cohort1: July-2017 → Cohort22: April-2018).

By looking at the lifecycle grids, we can see an overall trend where not a lot of people joined the game in the middle periods. Also, we can see that the oldest and newest cohorts are more present, and the newest ones significantly exist in all 4 quadrants. This can be used in our strategy in marketing and targeting strategies to try to convert customers from the 1st, 3rd and 4th quadrants into the 2nd one to make them our best current customers.

Generating financial lifecycle grids

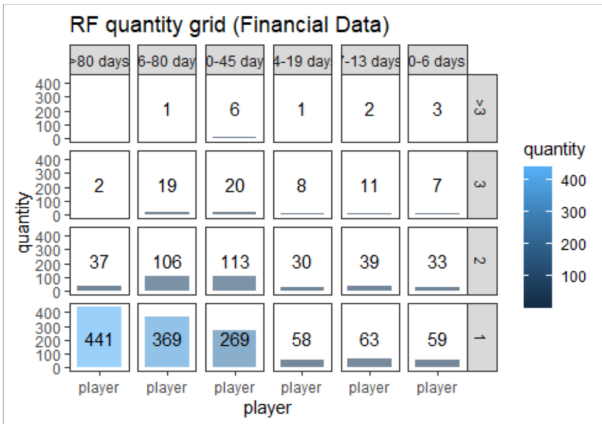
Code:

```
#Distributing players into segments by financial parameters
customer.segm.fin <- basetable %>%
  mutate(segm.freq.fin=ifelse(between(frequency_fin, 1, 1), '1',
                              ifelse(between(frequency_fin, 2, 2),
'2',
                                     ifelse(between(frequency_fin, 3,
3), '3','>3')))) %>%
  mutate(segm.rec.fin=ifelse(between(recency_fin, 0, 6), '0-6 days',
                             ifelse(between(recency_fin, 7, 13), '7-
13 days',
                                    ifelse(between(recency_fin, 14,
```

```
19), '14-19 days',

ifelse(between(recency_fin, 20, 45), '20-45 days',

ifelse(between(recency_fin, 46, 80), '46-80 days', '>80 days'))))))
customer.segm.fin<-
customer.segm.fin[!(is.na(customer.segm.fin$frequency_fin) &
is.na(customer.segm.fin$recency_fin)) ,]

# defining order of boundaries
customer.segm.fin$segm.freq.fin <-
factor(customer.segm.fin$segm.freq.fin, levels=c('>3', '3', '2', '1'))
customer.segm.fin$segm.rec.fin <-
factor(customer.segm.fin$segm.rec.fin, levels=c('>80 days', '46-80
days', '20-45 days', '14-19 days', '7-13 days', '0-6 days'))

lcg <- customer.segm.fin %>%
  group_by(segm.rec.fin, segm.freq.fin) %>%
  summarise(quantity=n()) %>%
  mutate(player='player') %>%
  ungroup()

ggplot(lcg, aes(x=player, y=quantity, fill=quantity))+
  theme_bw() +
  theme(panel.grid = element_blank())+
  geom_bar(stat='identity', alpha=0.6) +
  geom_text(aes(y=max(quantity)/2, label=quantity), size=4) +
  facet_grid(segm.freq.fin ~ segm.rec.fin)+
  ggtitle("RF quantity grid (Financial Data)")
```

Comments:

Here lifecycle grids are being generated using the customer.segm.fin dataframe.



This grid shows the frequency and recency of transactions made by the customers. The main observation is that a first transaction is not followed by a second one in the following days and weeks in most of the cases. Creating offers and bonuses might help in raising the number of transactions and stabilizing it in the long term.

Question 3 – Churn analysis + marketing strategy

Calculate which customers have churned

Code:

```
## GET customer id of churned players : gamers active this summer, and who did
not make any transaction in fall ##

# we already having summer active players in the basetable dataframe object

active_players_fall_transactions =
merge(basetable,fallfintrx,by="CustomerID",all.x=TRUE) #left join fall
transactions from active customers on CustomerID : retrieve transactions(s) of a
summer active player in fall or GET "na" if no transactions
churn_players =
active_players_fall_transactions[is.na(active_players_fall_transactions$TransID),
]  #only keep active players where transID is equal to NA (did not purchase
anything in fall)
churn_players$is_churn = rep(1, dim(churn_players)[1]) #labelizing them as
churners (==1)
churn_players = churn_players[,c("CustomerID", "is_churn")] # keep the customerID
and the churn label
dim(churn_players)[1]
```

Comments:

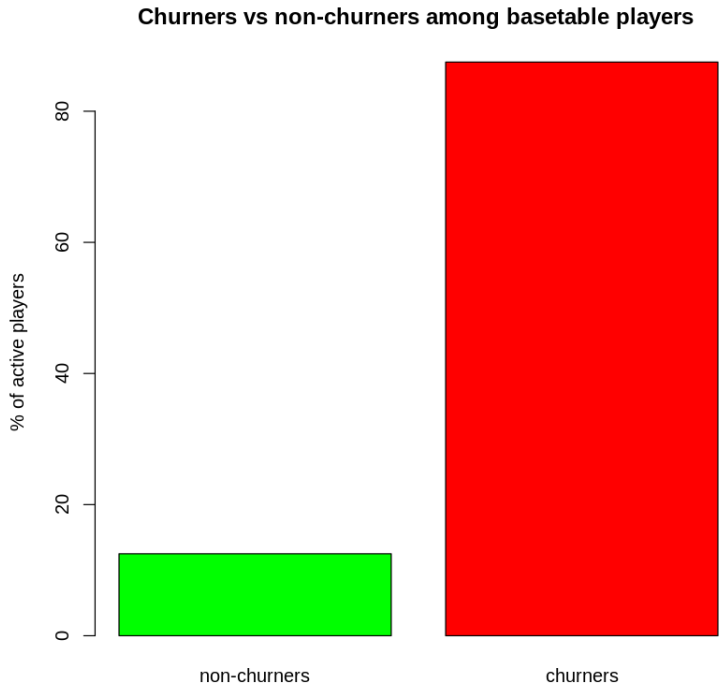
4,115 of the summer active players did not make any transactions in fall. Those players are considered as churners.

Checking the average churn rate in the basetable

Code:

```
## WARNING:  assuming that  the 73 summer active players that paid but never
played in fall as NON churners  (waiting for Professor answer)
basetable_with_churn = merge(basetable, churn_players, by="CustomerID",
all.x=TRUE) #updating the basetable from Q1 with churners
basetable_with_churn["is_churn"][is.na(basetable_with_churn["is_churn"])] = 0
#labelizing non churners as "0" instead of "NA"

churn_rate = (table(basetable_with_churn$is_churn)[2] /
dim(basetable_with_churn)[1]) *100 #87.5% of the  summer active players  churned
in fall (did not  pay anything) (=4,115 players) The basetable is highly
imbalanced.
churn_rate
barplot((table(basetable_with_churn$is_churn)/dim(basetable_with_churn)[1] )*100,
main="Fall churners vs non-churners among summer active players ", ylab= "% of
active players", col= c("green","red"), names.arg =c("non-churners", "churners")
)
```

Comments:



The churn ratio is pretty high : 87.5% of the players from our basetable are churners. Almost 9 summer active players out of 10 did not make any transactions in fall.
The label class is then considered as highly imbalanced.

Apply logistic regression to predict and detect churning players

1) Data Preparation on the basetable

Splitting the data set

Code:

```
### SPLITTING DATA ###
df = basetable_with_churn
library(caTools)
set.seed(1234)
split = sample.split(df$is_churn, SplitRatio = 0.70)    #stratified split
train = subset(df, split ==TRUE)
test = subset(df, split == FALSE)
#checking if split was well stratified
table(train$is_churn)[2]/dim(train)[1] #87% are churners
table(test$is_churn)[2]/dim(test)[1]  #87% are churners
```

Comments :

In this section, we prepared the data for training the model, by first Splitting the basetable (with churn) into a training set (70%) and a test set (30%), while keeping the same balance of churning vs non-churning players into both sets.
All the following data preparation steps are applied on train and test set separately to avoid any data leakage (ensured that model is only trained on train set).

Dealing with data types

Code:

```
# Dealing with data types ##
# checking data types
sapply(df, class)  #registrationData is a date variable, so really noisy and need
to be removed , Income variable appears as an integer variable while it describes
income level categories, we shall remove it and encode  the Income_Categories we
previously computed, CustomerType shall be converted into a categorical variable
, Sex and fallbonus  variables shall also be converted as categorical variables
#removing registrationDate ( we already used it for creating the feature
Seniority )
train$Registrationdate = NULL
test$Registrationdate = NULL
#removing income
train$Income = NULL
test$Income = NULL
#converting  Sex variable as a categorical variable
train$Sex = as.factor(train$Sex)
test$Sex =  as.factor(test$Sex)
#converting  faLLbonus variable as a categorical variable
```

```
train$fallbonus = as.factor(train$fallbonus)
test$fallbonus =  as.factor(test$fallbonus)
#converting  Income_Categories variable as a categorical variable
train$Income_Categories = as.factor(train$Income_Categories)
test$Income_Categories =  as.factor(test$Income_Categories)
#converting CustomerType as a categorical variable
customer_types = function (x) {  #naming the categories for analysis purpose
 if (x==1) y="walker"
 else {
   if (x==2) y="miscellaneous"
   else {
     if (x==3) y="social_raider"
     else {
       y="catcher"
     }
   }
 }
}
train$CustomerType = sapply(train$CustomerType, customer_types)
test$CustomerType = sapply(test$CustomerType, customer_types)
train$CustomerType = as.factor(train$CustomerType)
test$CustomerType = as.factor(test$CustomerType)
```

Comments :

Dealing with data types : some features are integer types and needed to be converted into categorical types (the Logistic Regression will ensure the encoding), other variables can be really noisy like RegistrationDate and needed to be removed. We also converted the CustomerType variable as a categorical one by creating its categories.

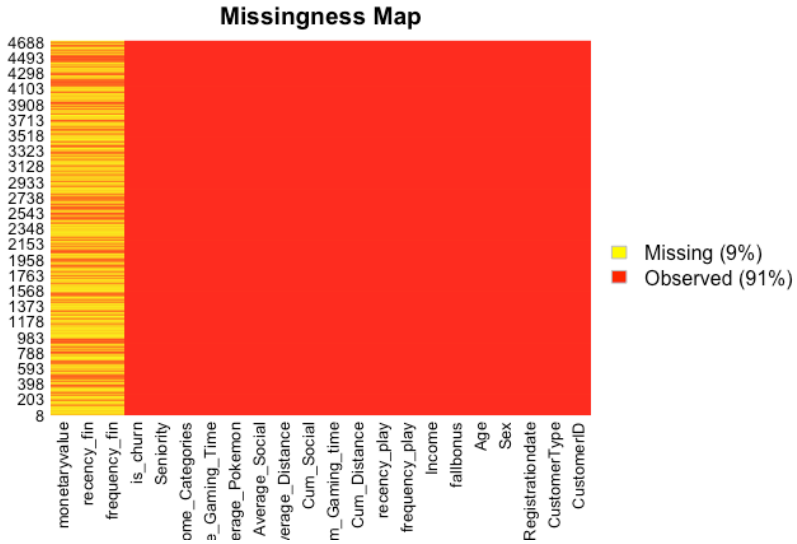
Dealing with missing values

Code:

```
## Dealing with missing values ##
# checking missing values  on whole dataframe
library(Amelia)
missmap(df,col=c("yellow","red"))  # only missing values on RFM financial
variables (active players who did not pay during summer ==> we will need to treat
them)
#computing mean values of reccency_fin, frequency_fin and monataryvalue variables
from train dataset
mean_f = mean(train$frequency_fin,na.rm = TRUE)
mean_r = mean(train$recency_fin,na.rm = TRUE)
mean_m = mean(train$monetaryvalue,na.rm = TRUE)
#replacing missing values on recency_fin, monataryvalue, frequency_fin by their
mean value in train dataset
train$frequency_fin[is.na(train$frequency_fin)]<-mean_f
train$recency_fin[is.na(train$recency_fin)]<-mean_r
```

```
train$monetaryvalue[is.na(train$monetaryvalue)]<-mean_m
#replacing missing values on recency_fin, monataryvalue, frequency_fin by their
mean value from traindataset  in test dataset == avoid any data leakage from the
test into the train set
test$frequency_fin[is.na(test$frequency_fin)]<-mean_f
test$recency_fin[is.na(test$recency_fin)]<-mean_r
test$monetaryvalue[is.na(test$monetaryvalue)]<-mean_m
```

Comments:



Dealing with Missing values on financial variables (active players of the basetable who never purchased something in the game) : we chose to replace NA values on a variable by its mean value, for training purpose.

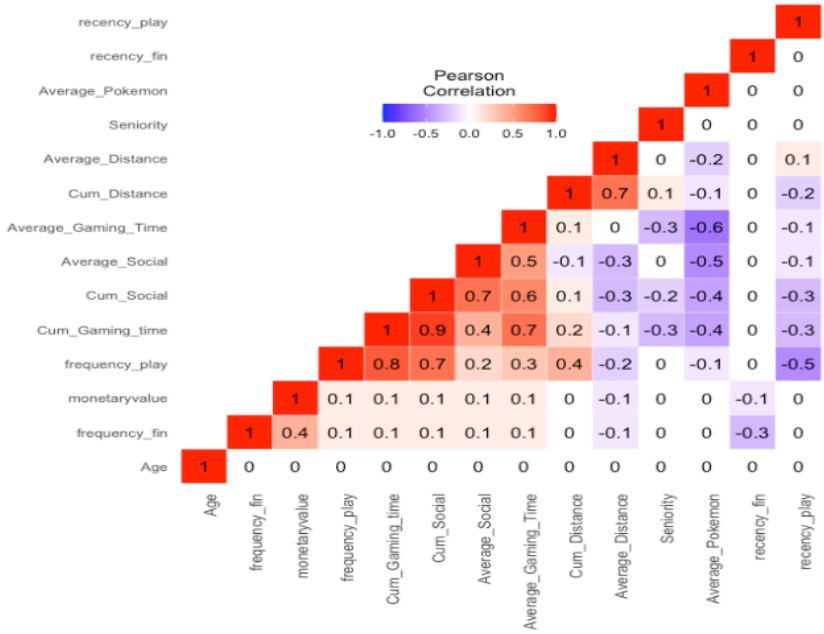
Dealing with numeric correlated variables

Code:

```
## Dealing with correlated variables ##
# checking correlation between numerical variables
library(ggplot2)
checking_corr  = train %>% select(Age, frequency_fin
,recency_fin,monetaryvalue,frequency_play,recency_play ,Cum_Distance
,Cum_Gaming_time, Cum_Social,Average_Distance ,Average_Social ,Average_Pokemon
,Average_Gaming_Time,Seniority)  # build dataframe only containing numerical
variables
cormat = round(cor(checking_corr),1)  #creating the correlation matrix
#creating heatmap
 # getting superior triangle of corr matrix
get_upper_tri <- function(cormat){
cormat[lower.tri(cormat)]<- NA
return(cormat)
 }
```

```r
#function to order the heatmap : display more correlated variables first
reorder_cormat <- function(cormat){
dd <- as.dist((1-cormat)/2)
hc <- hclust(dd)
cormat <-cormat[hc$order, hc$order]
}
library(reshape2)
# plotting Heatmap
cormat <- reorder_cormat(cormat)
upper_tri <- get_upper_tri(cormat)
melted_cormat <- melt(upper_tri, na.rm = TRUE)

ggheatmap <- ggplot(melted_cormat, aes(Var2, Var1, fill = value))+
geom_tile(color = "white")+
scale_fill_gradient2(low = "blue", high = "red", mid = "white",
  midpoint = 0, limit = c(-1,1), space = "Lab",
  name="Pearson\nCorrelation") +
 theme_minimal()+ # minimal theme
theme(axis.text.x = element_text(angle = 90, vjust = 1,
   size = 10, hjust = 1))+
coord_fixed()
#print corr coefficents on the heatmap
ggheatmap +
geom_text(aes(Var2, Var1, label = value), color = "black", size = 4) +
theme(
 axis.title.x = element_blank(),
 axis.title.y = element_blank(),
 panel.grid.major = element_blank(),
 panel.border = element_blank(),
 panel.background = element_blank(),
 axis.ticks = element_blank(),
 legend.justification = c(1, 0),
 legend.position = c(0.6, 0.7),
 legend.direction = "horizontal")+
 guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
               title.position = "top", title.hjust = 0.5))
#analysis : Cum_Social is highly correlate with Cum_gaming_time and
frequency_play  ; Cum_Gaming_time is highly correlated with frequency_play  ;
Cum_Distance is highly correlated with Average_Distance ==> we remove them
# removing high correlated variables
train$Cum_Social= NULL
test$Cum_Social = NULL
train$Cum_Gaming_time = NULL
test$Cum_Gaming_time = NULL
train$Cum_Distance = NULL
test$Cum_Distance = NULL
```

Comments:



Dealing with correlated variables : plotting a heatmap and removing high correlated variables, in order to avoid data repetition during the training process (most of the removed variables were those we computed when building the base table, in order to describe the general profile of the customers)

2) Modeling and Feature Selection

Code:

```
## MODELING & FEATURE SELECTION##
#training model using all variables
model <- glm(is_churn ~ . - CustomerID, family=binomial(link='logit'),data=train)
summary(model)

#dropping all insignificant variables, taking the 5% significant level , for the Income_Categories variable , none of the groups are significant
train$CustomerType = relevel(train$CustomerType, ref = "miscellaneous")# taking the Miscellaneous customer type as the base category for the CustomerType variables
model <- glm(is_churn ~ . - CustomerID - Average_Distance - Average_Pokemon - Average_Social - Average_Gaming_Time -Seniority -Age - frequency_fin -recency_fin -recency_play -Income_Categories - Sex, family=binomial(link='logit'),data=train)
summary(model)
```

Comments:

```
Call:
glm(formula = is_churn ~ . - CustomerID - Average_Distance -
    Average_Pokemon - Average_Social - Average_Gaming_Time -
    Seniority - Age - frequency_fin - recency_fin - recency_play -
    Income_Categories - Sex, family = binomial(link = "logit"),
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5255   0.3461   0.4134   0.5114   1.5319

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)               3.027976   0.155306  19.497  < 2e-16 ***
CustomerTypecatcher      -0.232069   0.152734  -1.519   0.1287
CustomerTypesocial_raider  0.285522   0.170413   1.675   0.0938 .
CustomerTypewalker         0.092514   0.161398   0.573   0.5665
fallbonus1                -1.287814   0.113125 -11.384  < 2e-16 ***
monetaryvalue             -0.010147   0.005214  -1.946   0.0517 .
frequency_play            -0.136094   0.019343  -7.036 1.98e-12 ***
```

We trained a logistic regression model on the training set, to predict the probability if a player will churn, i.e if an active player will stop paying.

By training the model on all the features we had from the basetable, and following the data preparation part, we removed insignificant variables.

The significant ones (or close to be significant) are the following :

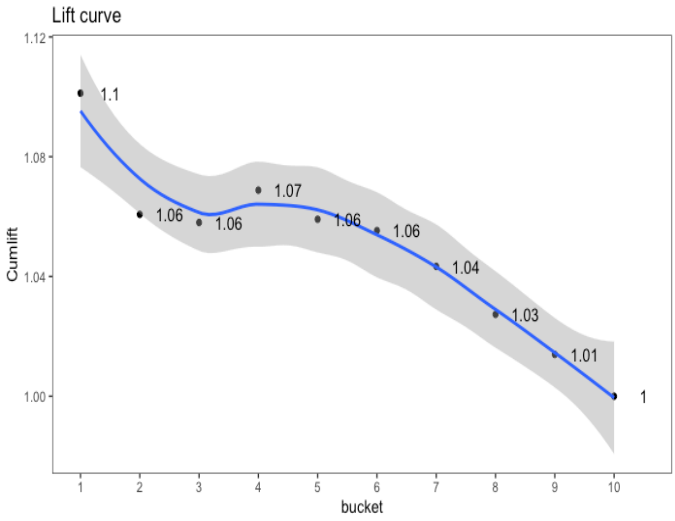
- frequency_play is significant : the more often a player would play, the less likely it tends to be a churner
- monetaryvalue is almost significant, we choose to keep it : the more money a player would spend, the less likely to churn
- **Fallbonus has a significant impact on the churn rate** : having received the fallbonus strongly decreases the odds of churning (coefficient is highly negative, odds of being a churner while having received fall bonus is exp(-1.28) = 0.25 times the odds of being a churner without having received the fall bonus, controlling for all other variables.The odds of churning are divided by 4 when a player has received the fall bonus) (beta coeff is the odds ratio between the two groups)
- Regarding the CustomerType variable : none of the groups are significant but the significance level and coefficient estimates can allow segment targeting :
    - the social raider group is close to be significant and being a social raider can increase the odds of churning comparing to being a miscellaneous player, and controlling for other variables (by exp(0.28)=1.32)
    - Additionally, the catcher group is kind of close to be significant and being a walker decreases the odds of churning by exp(-0.23)=0.76 comparing to miscellaneous players (and controlling for other variables)

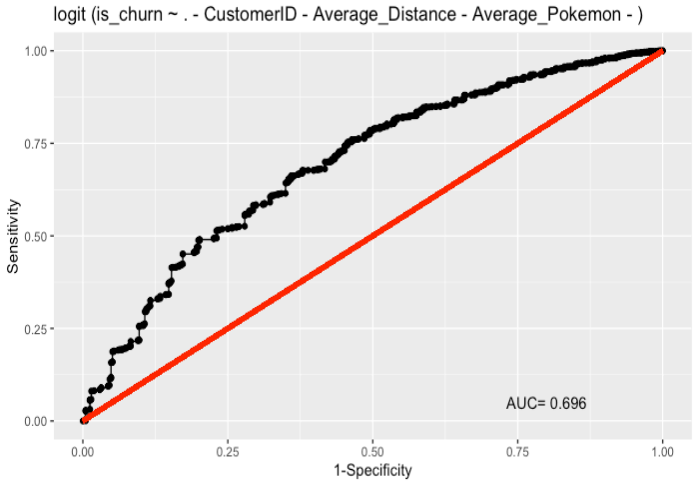
3) Evaluating model with overall metrics:

Code:

```
#PLOTTING THE ROC CURVE and predicting on the test set #
test$predictions = predict(model,test, type = 'response')
test$is_churn = as.factor(test$is_churn)
train$predictions  = predict(model,train, type = 'response')
train$is_churn  = as.factor(train$is_churn)
library(Deducer)
rocplot(model)

# PLOTTING THE LIFT Chart # (code from Professor Verbeeck)
lift <- function(depvar, predcol, groups=10)
{
 #making sure everything is numeric
 if(is.factor(depvar)) depvar <- as.integer(as.character(depvar))
 if(is.factor(predcol)) predcol <- as.integer(as.character(predcol))
 helper = data.frame(cbind(depvar, predcol))
 #sort df by churn prob from high to low and create deciles:
 helper[,"bucket"] = ntile(-helper[,"predcol"], groups)
 gaintable = helper %>% group_by(bucket)  %>%#in each decile
   summarise_at(vars(depvar), funs(total = n(),#count churners
                                    totalresp=sum(., na.rm = TRUE))) %>%#sum total
number of churners
   mutate(Cumresp = cumsum(totalresp),#cumulative churners
          Gain=Cumresp/sum(totalresp)*100,#churners found in decile versus total
amount of churners
          Cumlift=Gain/(bucket*(100/groups)))#gain obtained in comparison to
random number of churners found (10%-->20%)
 return(gaintable)

}
#construct the decile table
decilechart = lift(test$is_churn , test$predictions, groups = 10)

#attempt at a better lift chart
ggplot(decilechart, aes(x = bucket, y = Cumlift))+
 theme_bw() +
 theme(panel.grid = element_blank())+#removes colored box behind bar chart
 scale_x_continuous(breaks=c(1:10)) +
 geom_point()+
 geom_smooth()+
 geom_text(aes(x=bucket+0.5,y=Cumlift, label=round(Cumlift,2))) +
 ggtitle("Lift curve")
```

Comments:





We obtained general metrics to judge model performance such as :

- The AUC thanks to the ROC curve : AUC is quite good and close to 0.7 , we can notice that as long as we increase the cut off threshold (starting at 0), the True Positive Rate will decrease less quickly than the False Positive Rate ==> we can choose a big threshold, we will still have a more robust TPR than the FPR (ex: point on the ROC curve where TPR = 0.50 and FPR = 0.25 must correspond to a threshold close to 0.9) The greater the AUC, the better. The model is quite performant.

- The TopDecile Lift thanks to the Lift chart : the lift for the first decile is pretty low and is at 1.1. If we select the top 10% of players likely to churn, we can actually find 11% of true churners. The greater the Top Decile Lift, the better. Here it is quite low, but it is because of the imbalanced data set, in which we have much more churners than non churners. This metric is thus not relevant for the problem.

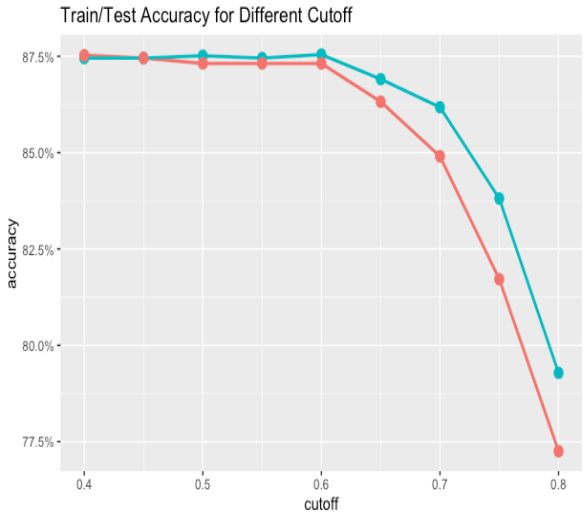
4) Finding optimal cut-off threshold

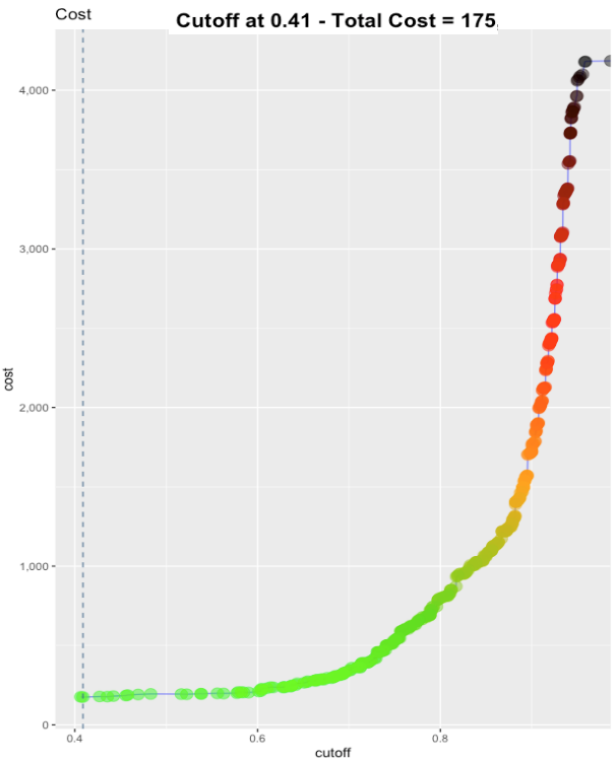
Code:

```
#FINDING OPTIMAL THRESHOLD#
source("unbalanced_functions.R")
accuracy_info <- AccuracyCutoffInfo( train = train, test = test,predict =
"predictions", actual = "is_churn" )
accuracy_info$plot
#checking with cut off of 0.5 : few FN, many FP
cm_info <- ConfusionMatrixInfo( data = test, predict = "predictions",
                                actual = "is_churn", cutoff = 0.5 )
cm_info
#checking the metrics
library(InformationValue)
sensitivity(test$is_churn,test$predictions, threshold = 0.5) #really high
specificity(test$is_churn,test$predictions, threshold = 0.5)#really low
1-misClassError(test$is_churn,test$predictions, threshold = 0.5)#accuracy, pcc
quit good
precision(test$is_churn,test$predictions, threshold = 0.5) # really high
```

```r
# 0.5 seems to be a good threshold if we really aim at detecting churners without caring that much about non- churners we may miss-classified (False Positives)
#let's then find optimal threshold to get a balance between FP and FN :
#defining  a "cost" function that computes  total  costs for the firm  when having a certain number of false positives (player predicted as churners but will finally keep paying) and a certain number of false negatives (player predicted as a non churner but will finally stop paying )
#minimize this cost function  by finding the best cut-off threshold that will classify probabilities of churning & gives us the best model metrics

#first  : compute the cost of not detecting a churner ==> we need to get the average CLV of the churning players
churners = basetable_with_churn[basetable_with_churn$is_churn==1,]
churners$monetaryvalue[is.na(churners$monetaryvalue)]=0 #players that never paid receive a monetaryvalue of 0
retention_rate = 1-0.87 # retention rate = 1-churn rate found above
acquisition_costs = 0 #assuming it is 0
cost_of_capital=  0 #assuming it is 0
t = 10  #let see on 10 years
calc_clv=function(margin,r,d,acquisition,t)
{
 clv<- -acquisition
 for(i in 0:t)#attention: start in year 0
 {
   clv<-clv+((r^i)*margin/(1+d)^i)
 }
 return (clv)
}
clv_churners=apply(churners[,c("monetaryvalue","frequency_fin")],1,function(x)
calc_clv(x[1],retention_rate,cost_of_capital,acquisition_costs,t))
cost_fn = mean(clv_churners)  #3.4 euros

#second  : compute the cost of classifying a non- churner as a churner ==> we need to assume a cost per consumer for a churn marketing campaign, let take 1 euros per player (it can be like sending emails  to those players predicted as churners)
cost_fp = 1

roc_info <- ROCInfo( data = cm_info$data, predict = "predict",
                     actual = "actual", cost.fp = cost_fp, cost.fn = cost_fn )
grid.draw(roc_info$plot)  #best cut off value is 0.41

# recomputing model metrics with this cut off value
library(InformationValue)
confusionMatrix(test$is_churn,test$predictions, threshold = 0.41)
sensitivity(test$is_churn,test$predictions, threshold = 0.41)#0.99, pretty good
specificity(test$is_churn,test$predictions, threshold = 0.41) #=0.005 still low, but at least not 0
1-misClassError(test$is_churn,test$predictions, threshold = 0.41)#0.87, still good
precision(test$is_churn,test$predictions, threshold = 0.41) #=0.87, still good
```

Comments:





In this section we aimed at retrieving the optimal cut off threshold to classify players as churners or non-churners given the predicted probability assigned by model.

To do so, we define a "cost" function for the Niantic which computed the cost of failing to detect a churner + cost of miss-classifying a non-churner. Therefore it computes the total cost for the firm of having false positives and false negatives.

We therefore have to estimate the cost of a False Positive and the cost of a False Negative:

- We assumed that the cost of having a False positive is 1 euro.(i.e the cost for a churn marketing campaign per customer, which can be sending an email to the players we think they will churn)
- By computing the average CLV of churning customers from the basetable , we estimate the **cost of a False Negative** at **3.4 euros** (i.e the amount of money Niantic can lose by customer if the latter actually churns)

Optimal threshold that minimizes the cost function is found at 0.41.

Given the business need for Niantic, we recommend to adjust the cut off threshold in order to detect churning players : *(see graph with cost in function of cutoff)*

- keeping a threshold at 0.41 to minimize the costs of failing to detect a churner AND miss-classifying a non-churners at the same time. When predicting on 1411 players for instance, it can minimize the total cost at 175 euros (assuming a churn marketing campaign = 1 euro /customer and a churning customer brings in average 3.4 euros to the company). 99% of churners were detected by the model, 87% of the churn predictions delivered by model were right.

- increasing the threshold above 0.41 to avoid miss-classification of non churners, but at the same time detecting less churners : the accuracy will plummet (see train/test accuracy plot)
- decreasing the threshold below 0.41 to reinforce the detection of churners while increasing the error of predicting a non-churner as a churner.

Recommendations following churn analysis

Here is a set of recommendations we can provide to Niantic to prevent churn following Q3 insights (please see the PowerPoint for the full marketing strategy recommended to Niantic):

- Keeping the segmentation of users in the customer database i.e label a new customer as a walker, or Raider, or Miscellaneous player or a catcher whenever we have enough information on him. As we saw when training the logistic regression model to predict churning players, the segmentation of customers (CustomerType variable) was close to be significant and thus important when estimating the probability of churning. For instance, based on model output we see that being a social raider player increases the odds of churning while being a catcher player decreases the odds of churning (comparing to a miscellaneous player)

- Using the predictive model (trained on 3,292 players and tested on 1,411 players from 2018) to prevent churning at the end of a summer period for instance. We saw that in 2018, 87% of summer active players did not pay anything in fall 2018. Besides, failing to detect a churning customer can cost 3.4 euros per customer on average to Niantic.

- If we assume the cost of a preventive e-mailing campaign to avoid churn is 1 euro per customer, Niantic should classify a player as a churner when its estimating probability of churning is above 0.41. Doing this, Niantic will ensure minimizing its total costs when failing to detect churners and miss-classifying non-churners (costs linked to model failures).

- However, setting a cut-off threshold above 0.41 will allow the firm to avoid more non-churner miss-classifications.

- Also, keep giving a bonus at the end of an active gaming season like summer, as we saw that whether or not a player had received a bonus was highly significant for predicting its probability of churning (it actually highly decreases the odds that a customer churn)