Projet analyse de données : Campagne marketing

Hamza RAIS, Tahar AMAIRI

2022-05-23

Contents

1 Introduction **2**

1.1 Mission . . . 2

1.2 Présentation du jeu de données . . . 2

1.2.1 Données personnelles . . . 2

1.2.2 Produits achetés . . . 2

1.2.3 Réductions promotionnelles . . . 3

1.2.4 Méthodes d'achat . . . 3

2 Nettoyage du dataset **3**

3 Statistique descriptive **7**

4 Régression linéaire et ANOVA ? **15**

5 Classification non-supervisée **18**

5.1 Composantes principales . . . 19

5.2 Clustering . . . 22

6 Classification supervisée **26**

6.1 Préparation . . . 26

6.2 AFD, LDA et QDA . . . 27

6.3 KNN . . . 28

6.4 CART et Random Forest . . . 29

6.5 Régression logistique . . . 32

6.6 Comparaison . . . 33

7 Conclusion **34**

1 Introduction

1.1 Mission

L'analyse de la personnalité des clients d'une entreprise aide à mieux comprendre ses clients et lui permet de modifier plus facilement ses produits en fonction des besoins, comportements et préoccupations spécifiques des différents types de clients.

En effet, les données récoltées permettent de dresser des portraits types. Ces données récoltées sur ce jeu de données permettent de déduire de manière sous-jacente des informations sur la perception de chaque portrait dressé de client c'est-à-dire qu'après cette analyse, le magasin saura modifier son produit en fonction de ses clients cibles issus de différents types de segments de clientèle. Au lieu de commercialiser un nouveau produit à chaque client de la base de données de l'entreprise à l'aveuglette, celle-ci peut analyser quel segment de clients est le plus susceptible d'acheter tel ou tel produit, puis commercialiser le produit uniquement sur ce segment particulier.

Également, elle peut revoir son inventaire et choisir de ramener dans un futur proche des proportions différentes de certains produits qui ont du succès avec les groupes à effectif majoritaire.

Finalement, il faudra aussi permettre à l'entreprise de pouvoir prédire la réponse d'un client à une offre de promotion.

1.2 Présentation du jeu de données

Le jeu de données comprend des variables concernant les clients directement mais aussi les produits achetés, le nombre de promotions utilisées pour acheter au magasin et finalement des données concernant la méthode d'un achat. C'est un jeu de données mixte, i.e, qu'il comprend des variables à la fois qualitatives et quantitatives.

1.2.1 Données personnelles

- ID : Identifiant unique du client
- Year_Birth : Année de naissance du client
- Education : Niveau d'éducation académique du client
- Marital_Status : Situation de famille du client
- Revenu : Revenu annuel du ménage du client
- Kidhome : Nombre d'enfants dans le foyer du client
- Teenhome : Nombre d'adolescents dans le foyer du client
- Dt_Customer : Date d'inscription du client dans l'entreprise
- Recency : Nombre de jours depuis le dernier achat du client
- Plainte : 1 si le client s'est plaint au cours des 2 dernières années, 0 sinon

1.2.2 Produits achetés

- MntWines : Montant dépensé en vin au cours des 2 dernières années
- MntFruits : Montant dépensé pour les fruits au cours des 2 dernières années
- MntMeatProducts : Montant dépensé pour la viande au cours des 2 dernières années
- MntFishProducts : Montant dépensé pour du poisson au cours des 2 dernières années
- MntSweetProducts : Montant dépensé en sucreries au cours des 2 dernières années
- MntGoldProds : Montant dépensé en or au cours des 2 dernières années

1.2.3 Réductions promotionnelles

- NumDealsPurchases : Nombre d'achats effectués avec une réduction
- AcceptedCmp1 : 1 si le client a accepté l'offre lors de la 1ère campagne, 0 sinon
- AcceptedCmp2 : 1 si le client a accepté l'offre lors de la 2ème campagne, 0 sinon
- AcceptedCmp3 : 1 si le client a accepté l'offre lors de la 3ème campagne, 0 sinon
- AcceptedCmp4 : 1 si le client a accepté l'offre lors de la 4ème campagne, 0 sinon
- AcceptedCmp5 : 1 si le client a accepté l'offre lors de la 5ème campagne, 0 sinon
- Response : 1 si le client a accepté l'offre lors de la dernière campagne, 0 sinon

1.2.4 Méthodes d'achat

- NumWebPurchases : Nombre d'achats effectués sur le site web de l'entreprise
- NumCatalogPurchases : Nombre d'achats effectués par le biais d'un catalogue
- NumStorePurchases : Nombre d'achats effectués directement en magasin
- NumWebVisitsMonth : Nombre de visites sur le site web de l'entreprise au cours du dernier mois

2 Nettoyage du dataset

Tout d'abord, nous allons récupérer le jeu de données :

```
#load data
data = read.csv2("marketing_campaign.csv", sep= "\t")
```

On commence par supprimer les observations où au moins une donnéee est manquante :

```
### Cleaning ###
data = na.omit(data) #drop missing values because few NA items
```

Nous allons maintenant travailler sur les différentes variables :

```
#check data type
str(data)
```

```
## 'data.frame':    2216 obs. of  29 variables:
##  $ ID                 : int  5524 2174 4141 6182 5324 7446 965 6177 4855 5899 ...
##  $ Year_Birth         : int  1957 1954 1965 1984 1981 1967 1971 1985 1974 1950 ...
##  $ Education          : chr  "Graduation" "Graduation" "Graduation" "Graduation" ...
##  $ Marital_Status     : chr  "Single" "Single" "Together" "Together" ...
##  $ Income             : int  58138 46344 71613 26646 58293 62513 55635 33454 30351 5648 ...
##  $ Kidhome            : int  0 1 0 1 1 0 0 1 1 1 ...
##  $ Teenhome           : int  0 1 0 0 0 1 1 0 0 1 ...
##  $ Dt_Customer        : chr  "04-09-2012" "08-03-2014" "21-08-2013" "10-02-2014" ...
##  $ Recency            : int  58 38 26 26 94 16 34 32 19 68 ...
##  $ MntWines           : int  635 11 426 11 173 520 235 76 14 28 ...
##  $ MntFruits          : int  88 1 49 4 43 42 65 10 0 0 ...
##  $ MntMeatProducts    : int  546 6 127 20 118 98 164 56 24 6 ...
##  $ MntFishProducts    : int  172 2 111 10 46 0 50 3 3 1 ...
##  $ MntSweetProducts   : int  88 1 21 3 27 42 49 1 3 1 ...
##  $ MntGoldProds       : int  88 6 42 5 15 14 27 23 2 13 ...
```

```
##  $ NumDealsPurchases  : int  3 2 1 2 5 2 4 2 1 1 ...
##  $ NumWebPurchases    : int  8 1 8 2 5 6 7 4 3 1 ...
##  $ NumCatalogPurchases: int  10 1 2 0 3 4 3 0 0 0 ...
##  $ NumStorePurchases  : int  4 2 10 4 6 10 7 4 2 0 ...
##  $ NumWebVisitsMonth  : int  7 5 4 6 5 6 6 8 9 20 ...
##  $ AcceptedCmp3       : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ AcceptedCmp4       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AcceptedCmp5       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AcceptedCmp1       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ AcceptedCmp2       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Complain           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Z_CostContact      : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ Z_Revenue          : int  11 11 11 11 11 11 11 11 11 11 ...
##  $ Response           : int  1 0 0 0 0 0 0 0 1 0 ...
##  - attr(*, "na.action")= 'omit' Named int [1:24] 11 28 44 49 59 72 91 92 93 129 ...
##   ..- attr(*, "names")= chr [1:24] "11" "28" "44" "49" ...
```

La colonne des identifiants n'est pas intéressante. Bien plus, on supprimera les variable **Z_Revenue** et **Z_CostContact** car elles ne contiennent qu'une valeur :

```
data$ID = NULL #ID is useless
print(unique(data$Z_CostContact)) #contains only one value
```

```
## [1] 3
```

```
print(unique(data$Z_Revenue)) #contains only one value
```

```
## [1] 11
```

```
data$Z_CostContact = NULL #useless
data$Z_Revenue = NULL #useless
```

On supprimera toutes les colonnes **AcceptedCmp** car la colonne **Response** résume à elle seule toutes ces colonnes :

```
#we kill keep response instead of all AcceptedCmp* columns
data$AcceptedCmp1 = NULL
data$AcceptedCmp2 = NULL
data$AcceptedCmp3 = NULL
data$AcceptedCmp4 = NULL
data$AcceptedCmp5 = NULL
```

On cherche à connaitre les différentes valeurs que peut prendre les variables contenant le statut marital et le niveau d'éducation et leurs fréquences d'apparition :

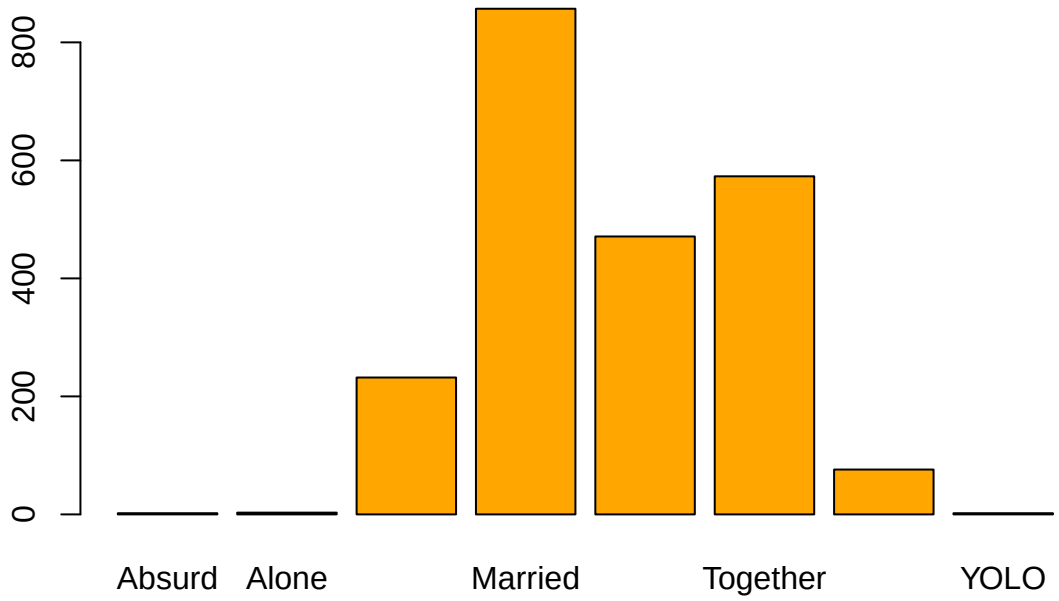
```
#get values from marital status and education
print(unique(data$Marital_Status))
```

```
## [1] "Single"   "Together" "Married"  "Divorced" "Widow"    "Alone"    "Absurd"
## [8] "YOLO"
```

```r
print(unique(data$Education))
```

```
## [1] "Graduation" "PhD"        "Master"     "Basic"      "2n Cycle"
```

```r
#check boxplot of marital status
barplot(table(data$Marital_Status), col='#ffa600')
```



Certains attributs représentent une partie infime de la population, il faudrait penser à les inclure dans un groupe ou à les supprimer. Par ailleurs, il y a peu d'observations avec un statut marital **YOLO** ou **Absurd**. On decide donc de les supprimer car la perte d'information est minime. De plus, on ne sait pas à quoi correspondent ces valeurs.

```r
#remove rows with YOLO and Absurd as marital status (few samples)
data = data[data$Marital_Status != "YOLO",]
data = data[data$Marital_Status != "Absurd",]
```

À l'inverse, pour le niveau d'éducation, on ne va pas supprimer certaines valeurs, on va préférer les englober dans deux sous-catégories représentant un niveau faible et élevé d'éducation : **postgraduate** et **undergraduate**.

```r
#to many Education lvl : we will summarize them in two categories :
#PG = postgraduate and UG = undergraduate
data$Education[data$Education == "2n Cycle"] = "UG"
data$Education[data$Education == "Basic"] = "UG"
```

```
data$Education[data$Education == "Graduation"] = "PG"
data$Education[data$Education == "Master"] = "PG"
data$Education[data$Education == "PhD"] = "PG"
```

Même raisonnement pour le statut marital : on englobe les clients mariés ou en relation en tant que personne en couple et le reste est pris pour célibataire :

```
#same thing for Marital_Status : single or couple
data$Marital_Status[data$Marital_Status == "Divorced"] = "Single"
data$Marital_Status[data$Marital_Status == "Widow"] = "Single"
data$Marital_Status[data$Marital_Status == "Together"] = "Couple"
data$Marital_Status[data$Marital_Status == "Married"] = "Couple"
data$Marital_Status[data$Marital_Status == "Alone"] = "Single"
```

Plutôt que de considérer de manière séparée les adolescents et les enfants, on les rassemble au sein d'une même variable **Child**. La logique derrière est que tous les deux représentent une charge supplémentaire au ménage du client.

```
#merge kid and teen variables into one column
data$Child = data$Kidhome + data$Teenhome
data$Kidhome = NULL
data$Teenhome = NULL
```

On souhaite connaitre l'âge des clients plutôt que leur année de naissance, ce qui sera beaucoup plus approprié pour les méthodes qu'on va implémenter ultérieurement.

```
#change date of birth into age
data$Age = 2022 - data$Year_Birth
data$Year_Birth = NULL
```

Même logique, on cherche à connaitre l'ancienneté du client que la date de son premier achat :

```
#change the date of customer's enrollment into seniority
data$Dt_Customer = substr(data$Dt_Customer,7,10)
data$Seniority  = 2022 - as.numeric(data$Dt_Customer)
data$Dt_Customer = NULL
```

Concernant les produits, on va créer deux catégories : les produits essentiels et les produits optionnels. De cette manière, on peut espérer que les ménages possédant le moins de revenus se concentreront davantage sur l'achat de produits essentiels.

```
#split products in two categories : necessary and optional products
data$MntOptional = data$MntWines + data$MntGoldProds + data$MntSweetProducts
data$MntNecessary = data$MntFishProducts + data$MntFruits + data$MntMeatProducts
data$MntWines = NULL
data$MntFishProducts = NULL
data$MntFruits = NULL
data$MntGoldProds = NULL
data$MntMeatProducts = NULL
data$MntSweetProducts = NULL
```

Finalement, toute dépense est regroupé dans une même variable **TotalPurchases** et on fixera les variables qualitatives en tant que facteur :

```
#merge all the purchases
data$TotalPurchases = data$NumWebPurchases + data$NumCatalogPurchases + data$NumStorePurchases
data$NumWebPurchases = NULL
data$NumCatalogPurchases = NULL
data$NumStorePurchases = NULL

isFactor <- function(x) #to set a column as factor
{
  if(length(unique(x)) == 2)
  {
    factor(x)
  }

  else x
}

#set factors
data[] = lapply(data,isFactor)
#order by column names
data = data[,order(names(data))]
attach(data)
```

3 Statistique descriptive

Tout d'abord, on regarde les statistiques générales sur l'ensemble des variables. Cela permet de mieux détecter les observations avec certaines données isolées.

```
summary(data)
```

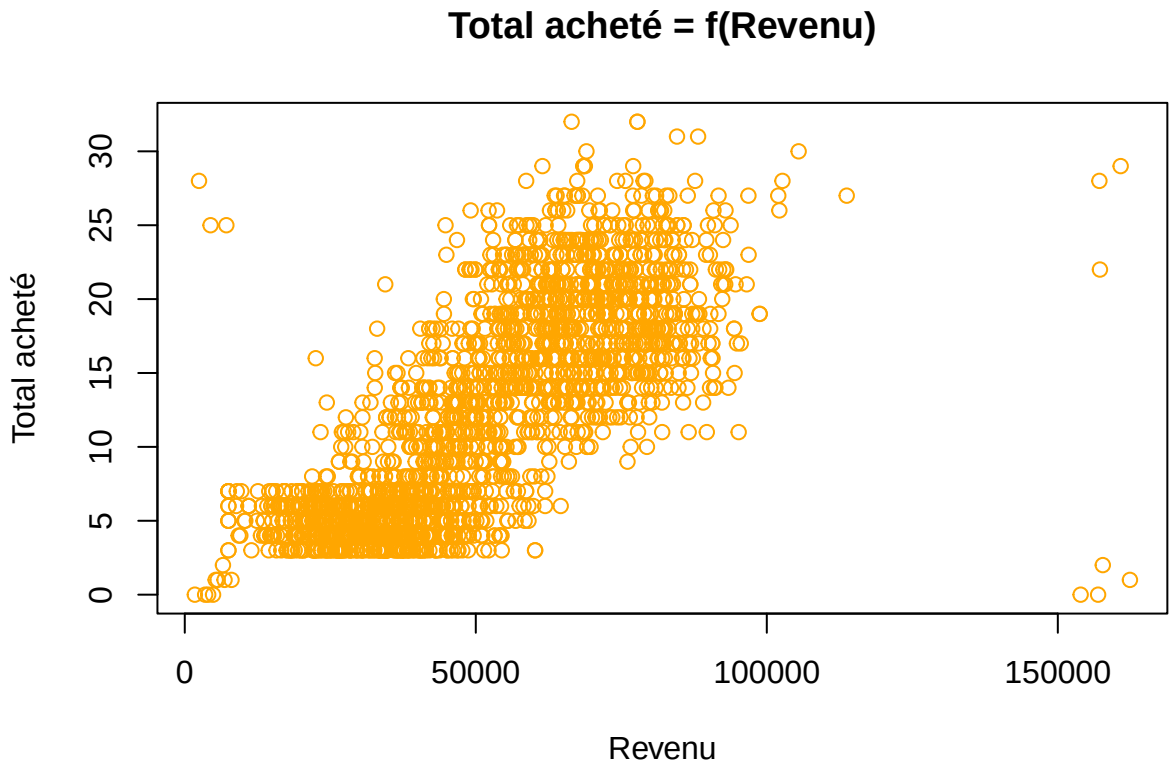
```
##       Age              Child       Complain Education     Income
##  Min.   : 26.00   Min.   :0.000   0:2191   PG:1958   Min.   :  1730
##  1st Qu.: 45.00   1st Qu.:0.000   1:  21   UG: 254   1st Qu.: 35234
##  Median : 52.00   Median :1.000                      Median : 51382
##  Mean   : 53.19   Mean   :0.948                      Mean   : 52232
##  3rd Qu.: 63.00   3rd Qu.:1.000                      3rd Qu.: 68522
##  Max.   :129.00   Max.   :3.000                      Max.   :666666
##  Marital_Status  MntNecessary     MntOptional     NumDealsPurchases
##  Couple:1430    Min.   :   1.0   Min.   :   3.0   Min.   : 0.000
##  Single: 782    1st Qu.:  25.0   1st Qu.:  41.0   1st Qu.: 1.000
##                 Median :  90.0   Median : 245.5   Median : 2.000
##                 Mean   : 230.8   Mean   : 375.9   Mean   : 2.321
##                 3rd Qu.: 357.8   3rd Qu.: 617.0   3rd Qu.: 3.000
##                 Max.   :1727.0   Max.   :1689.0   Max.   :15.000
##  NumWebVisitsMonth    Recency      Response   Seniority      TotalPurchases
##  Min.   : 0.00     Min.   : 0.00   0:1881   Min.   : 8.000   Min.   : 0.00
##  1st Qu.: 3.00     1st Qu.:24.00   1: 331   1st Qu.: 9.000   1st Qu.: 6.00
##  Median : 6.00     Median :49.00            Median : 9.000   Median :12.00
##  Mean   : 5.32     Mean   :49.05            Mean   : 8.971   Mean   :12.55
##  3rd Qu.: 7.00     3rd Qu.:74.00            3rd Qu.: 9.000   3rd Qu.:18.00
##  Max.   :20.00     Max.   :99.00            Max.   :10.000   Max.   :32.00
```

On s'aperçoit ici pour la variable **Income** qu'il y a un maximum bien trop grand par rapport aux autres résultats. On va donc enlever les observations où le revenu est supérieur à 600000 :

```
library(dplyr)
data = data %>% filter(Income < 6e+05)
```

On cherche maintenant à savoir s'il y a un lien intime entre le montant dépensé et le pouvoir d'achat des ménages :

```
plot(data$Income, data$Total, col='#ffa600', xlab='Revenu', ylab='Total acheté', main= 'Total acheté = f
```

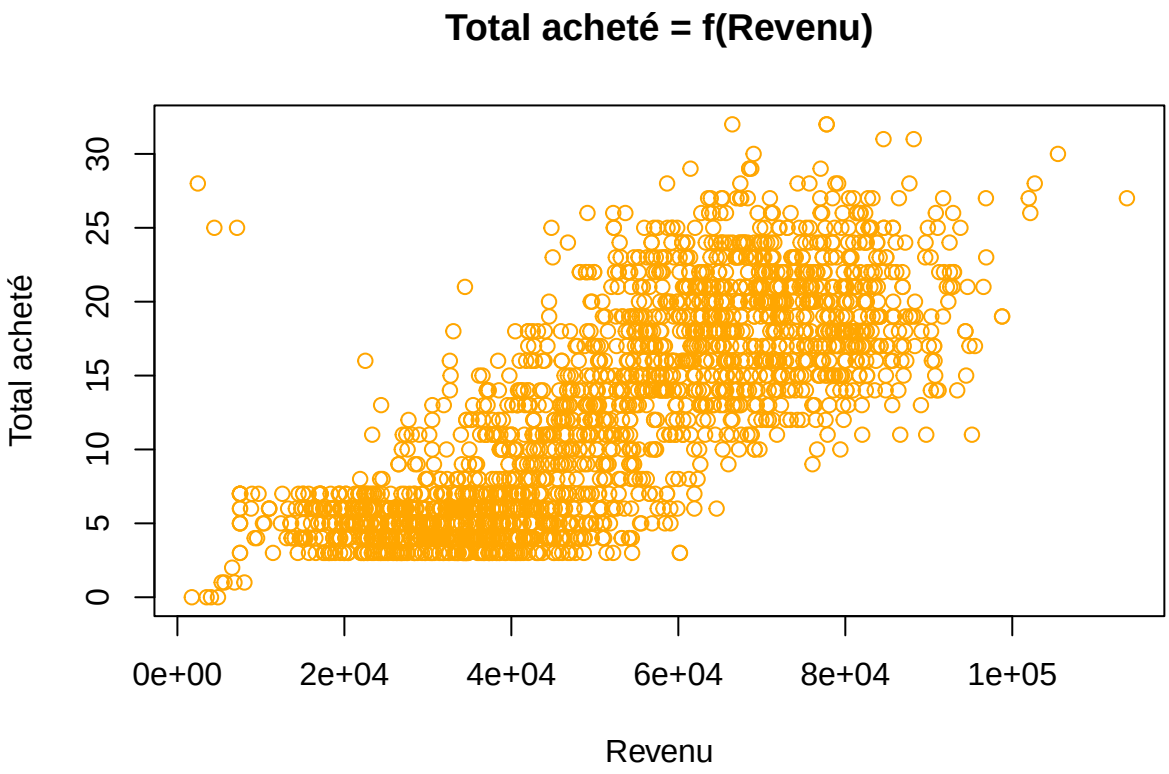


On remarque que c'est le cas : de manière assez générale, plus le revenu est élevé, plus le montant total dépensé sera important. En outre, on remarque 7 observations isolées avec des revenus au dessus de 150000, on va donc les supprimer :

```
data = filter(data, Income < 15e+04)
```

Voici le même graphique avec les observations isolées supprimées :

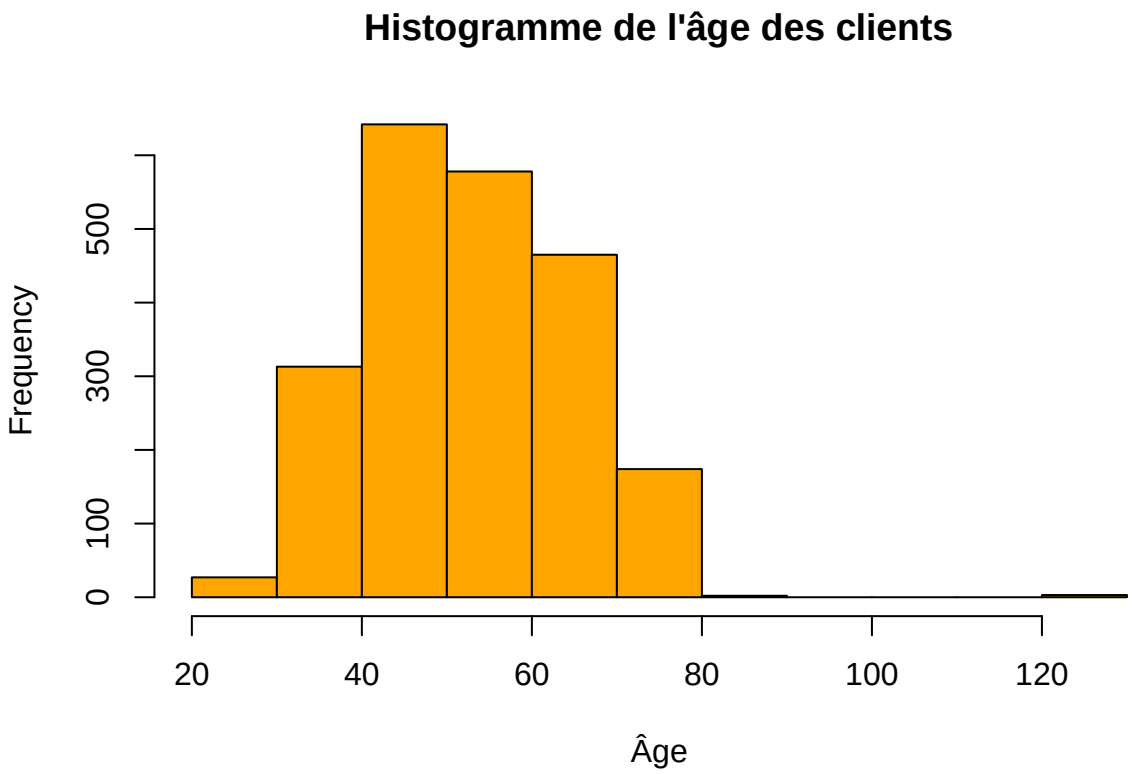
```
plot(data$Income, data$Total, col='#ffa600', xlab='Revenu', ylab='Total acheté', main= 'Total acheté = f
```



Pour la plupart de ces points, on retrouve une courbe linéairement croissante. Il ne serait pas anodin de penser que le revenu des clients a un impact sur leur pouvoir d'achat. Ainsi, on pourra notamment prédire le total acheté en fonction du revenu via une régression linéaire.

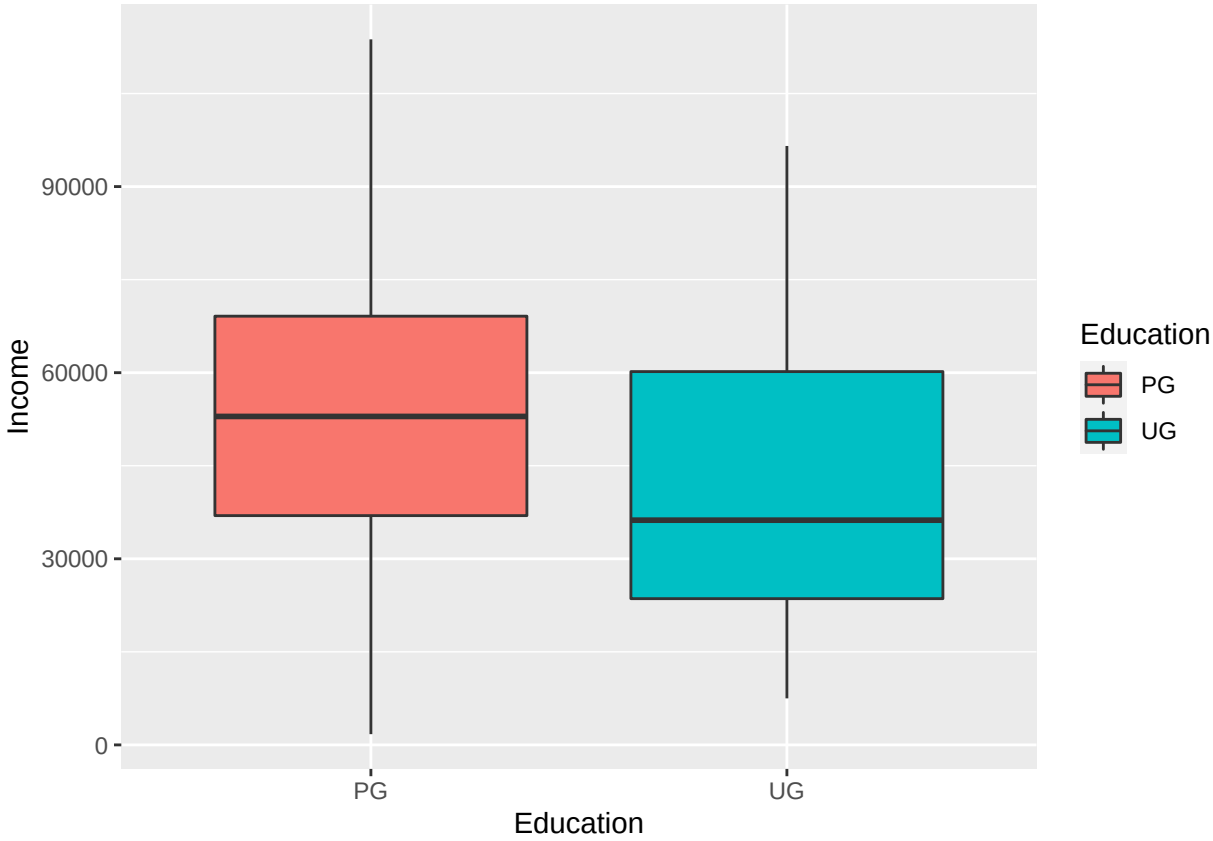
Étudions la répartition des âges des clients :

```
hist(data$Age, col='#ffa600', xlab="Âge", main= "Histogramme de l'âge des clients")
```



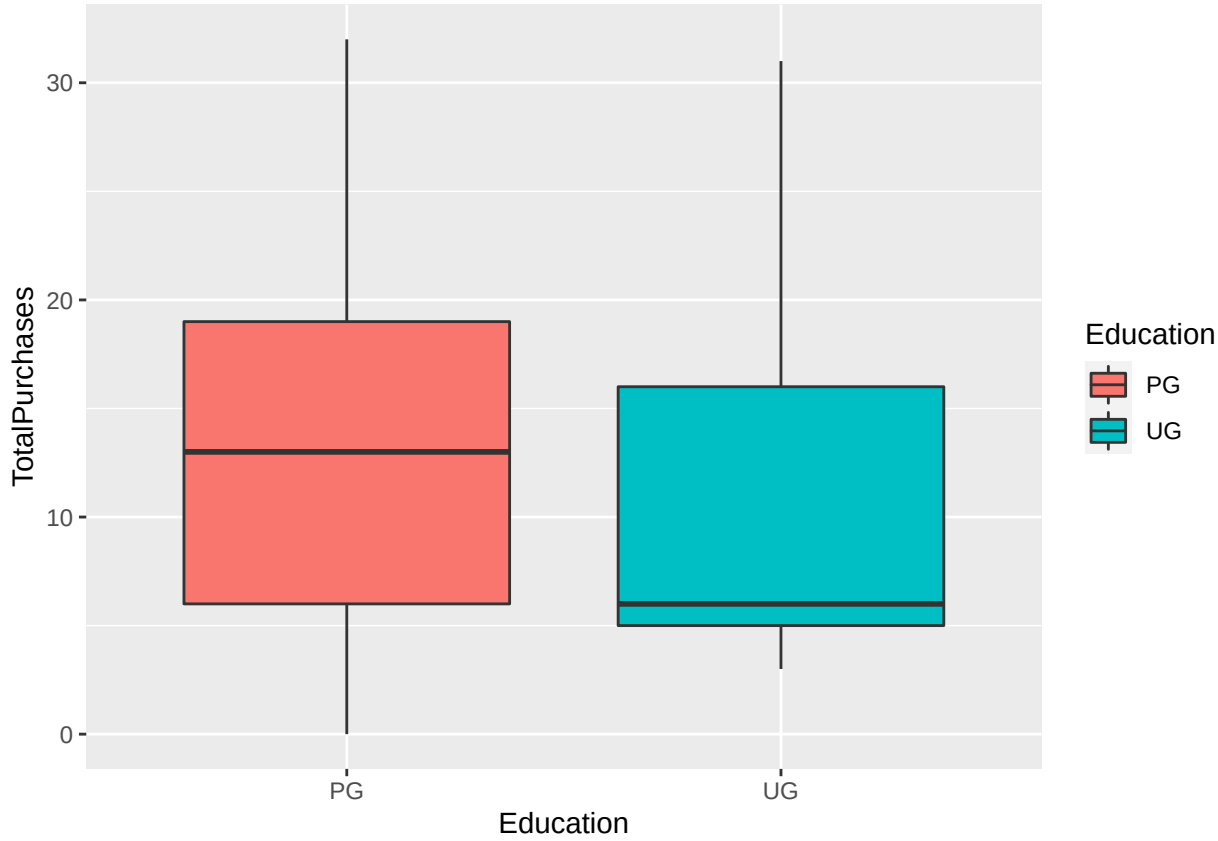
Une part importante des clients se retrouvent âgés de 45 à 55 ans. L'âge moyen est d'environ 50 ans. Comme nous connaissons pas la date du jeu de données, il se peut que la valeur 2022 utilisé pour obtenir l'âge des individus soit peut être élevée mais dans l'absolu cela ne changera absolument rien.

```
library(ggplot2)
ggplot(data, aes(x=Education,y=Income,fill=Education))+geom_boxplot(outlier.colour="black")
```



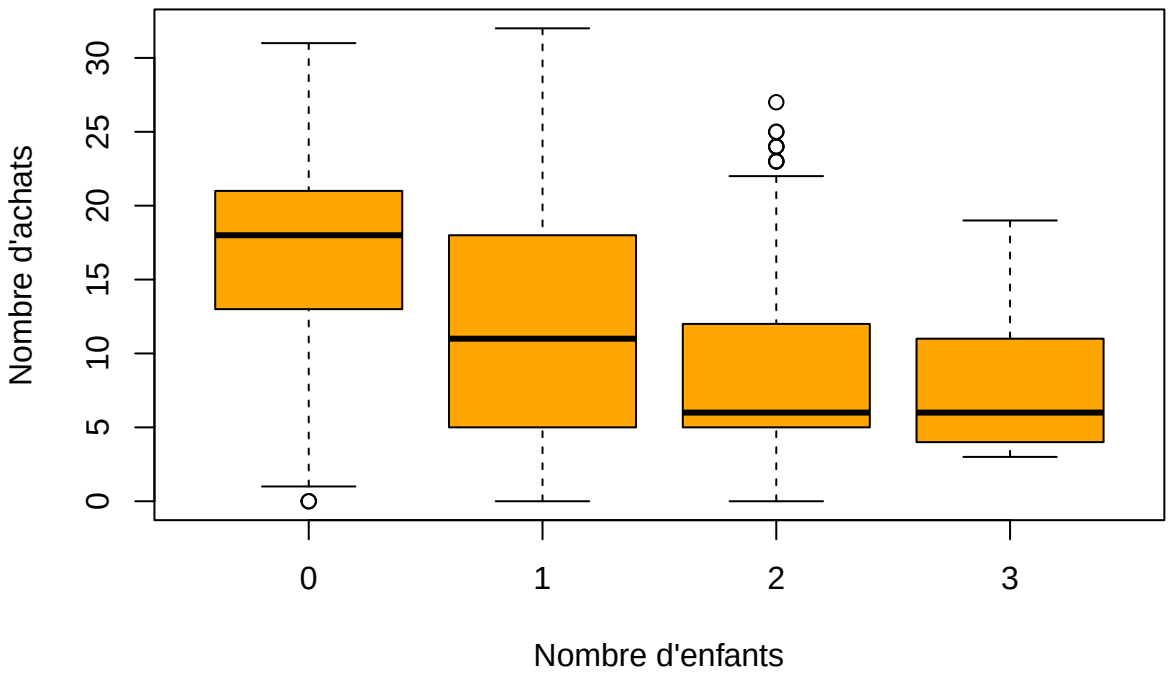
Plus le niveau d'éducation est élevé, plus le revenu du client est élevé. Cela se voit notamment avec la moyenne de revenu plus haute chez les clients diplômés.

```
ggplot(data, aes(x=Education,y=TotalPurchases,fill=Education))+geom_boxplot(outlier.colour="black")
```



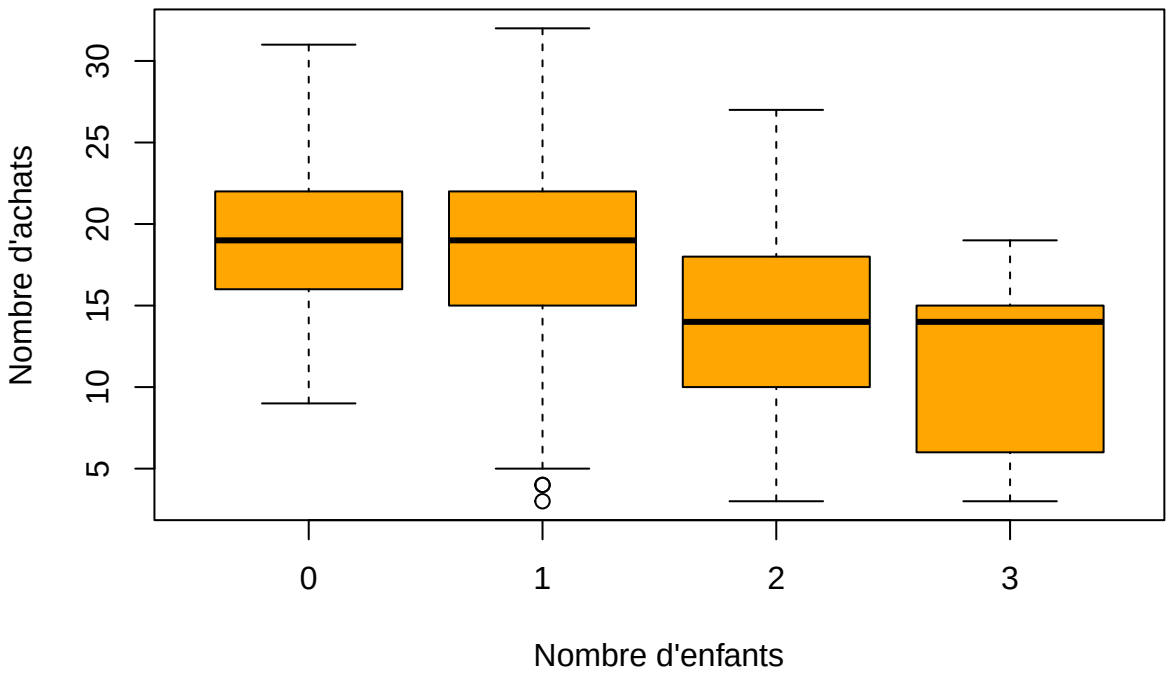
Et cela s'impacte par des achats beaucoup plus élevés chez les personnes hautement diplômées.

```
boxplot(TotalPurchases~Child,data=data, col="#ffa600", xlab="Nombre d'enfants", ylab="Nombre d'achats")
```



On remarque que plus les clients ont d'enfants et moins ils vont acheter fréquemment au magasin. Maintenant, on se demande si la boite à moustache change lorsque les ménages disposent d'un revenu supérieur à la moyenne.

```
temp_df= data
temp_df= filter(temp_df, Income>mean(Income))
boxplot(TotalPurchases~Child,data=temp_df, col="#ffa600", xlab="Nombre d'enfants", ylab="Nombre d'achats
```



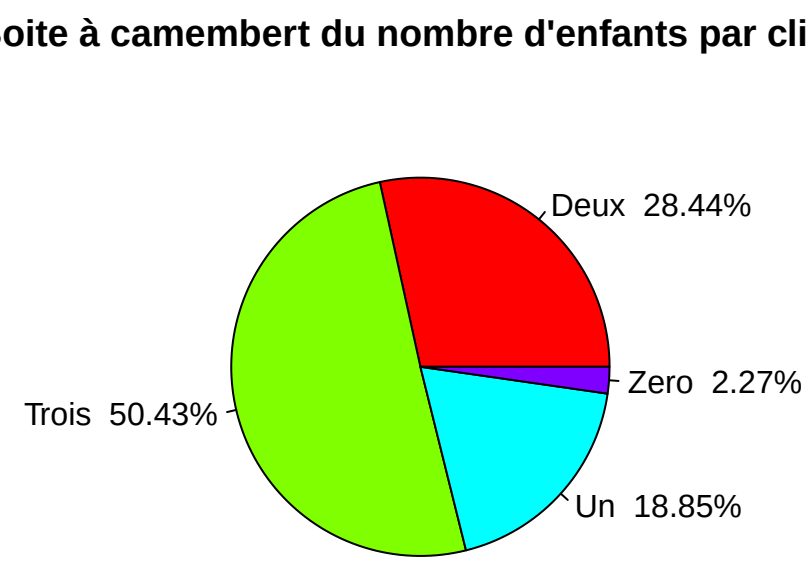
On remarque que l'écart se resserre lorsque le revenu est supérieur de la moyenne de revenus. Voyons maintenant le statut marital :

```
data %>% group_by(Marital_Status) %>% summarise(percent= 100 * n() / nrow(data) )
```

```
## # A tibble: 2 x 2
##   Marital_Status percent
##   <fct>            <dbl>
## 1 Couple            64.6
## 2 Single            35.4
```

64% du jeu de données comprend des clients en couple et le reste est célibataire.

```
temp_df= data
temp_df$Child[data$Child == 0] = "Zero"
temp_df$Child[data$Child == 1] = "Un"
temp_df$Child[data$Child == 2] = "Deux"
temp_df$Child[data$Child == 3] = "Trois"

values= c(28.44,50.43, 18.85, 2.27)
lbls=paste(names(table(temp_df$Child)), "", values)
lbls=paste(lbls,"%", sep="")
pie(values, labels = lbls, main="Boite à camembert du nombre d'enfants par client", col=rainbow(length(
```



Finalement, environ 79% des clients ont au moins 2 enfants et seulement 2% n'en ont pas, les clients sont donc généralement parents.

4 Régression linéaire et ANOVA ?

Comme on veut appliquer des modèles de régression, on doit utiliser des variables quantitatives uniquement. On va donc retirer les colonnes **Complain, Education, Marital_Status** et **Response**.

```
models_data = data
models_data = models_data[,-c(3, 4, 6, 12)]
```

On pose un modèle de régression multiple pour prédire le nombre d'achats à partir de toutes les autres variables présentes.

```
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

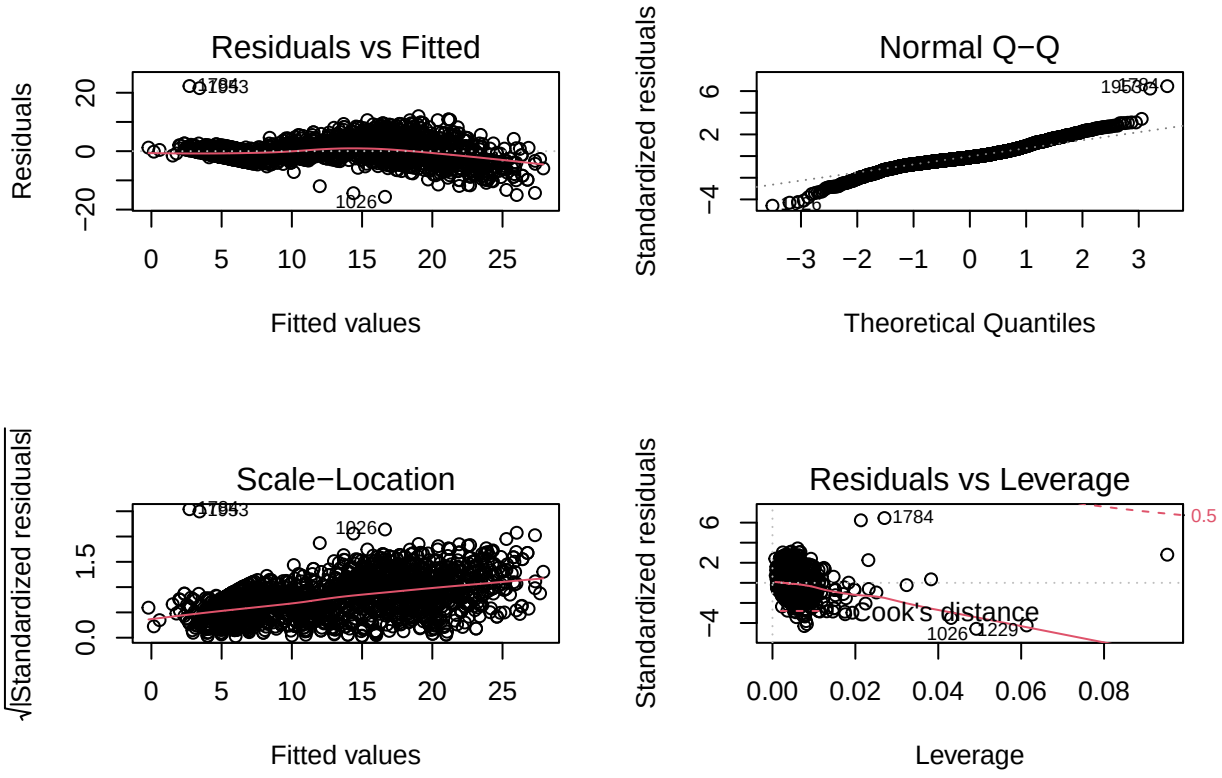
```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
res = lm(TotalPurchases~., models_data)
vif(res)
```

```
##                Age              Child             Income      MntNecessary
##           1.099280           1.913446           4.537734           2.889469
##       MntOptional NumDealsPurchases NumWebVisitsMonth            Recency
##           3.134294           1.466648           2.404926           1.004608
##         Seniority
##           1.216998
```

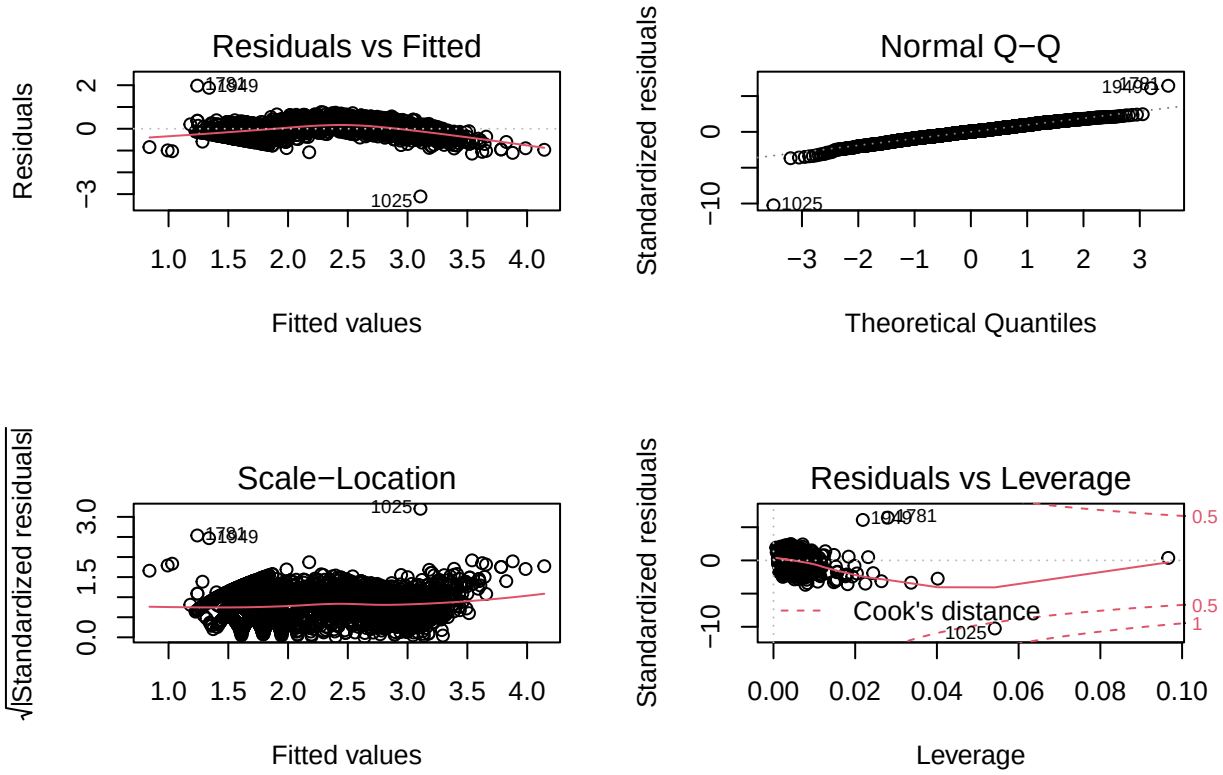
Si $VIF > 10$, on doit supprimer la variable du modèle, car il y a colinéarité. Cependant ici ce n'est pas le cas.

```
par(mfrow=c(2,2))
plot(res)
```



Essayons de voir avec le *log* si le **Q-Q Plot** est mieux réussi. Or, ce n'est pas possible d'appliquer le logarithme si les valeurs sont nulles. On va donc enlever les valeurs nulles pour TotalPurchases.

```
models_data = filter(models_data, TotalPurchases > 0)
res = lm(log(TotalPurchases)~., models_data)
par(mfrow=c(2,2))
plot(res)
```



Le modèle appliqué au log est meilleur car ici, le Q-Q plot montre que les données sont gaussiennes. On veut maintenant vérifier que les erreurs résiduelles sont gausssiennes avec un test de **shapiro**.

```
shapiro.test(res$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res$residuals
## W = 0.96891, p-value < 2.2e-16
```

La p-valeur est bien inférieure à 0.05. On va chercher à enlever certaines observations aberrantes et revérifier :

```
abs(rstudent(res))[abs(rstudent(res))>2]
```

```
##        10        46        65        70        73        99       140       228
##  2.755850  2.098775  2.378979  2.248164  2.211054  2.186042  2.058491  3.508209
##       259       290       322       326       346       417       419       493
##  2.043967  2.079505  2.413746  2.426669  2.001166  2.021735  3.145482  2.372368
##       498       555       630       647       651       667       680       687
##  2.312418  2.482112  2.917661  2.224610  2.317826  2.005826  2.268736  2.314982
##       691       699       718       736       758       801       841       891
##  2.140830  2.831291  2.573406  2.224610  3.370438  2.065690  2.057123  2.363893
##       964       973       978       984      1025      1043      1045      1051
```

```
##  2.026512  2.126247  2.268736  3.686041 10.502563  3.074745  2.072020  2.012022
##      1062      1066      1071      1082      1095      1098      1109      1141
##  2.109464  2.225934  2.154727  2.023991  2.158670  2.595982  2.034278  2.246139
##      1142      1162      1323      1344      1372      1404      1406      1480
##  2.091380  2.216367  3.074745  2.308790  2.023052  2.196880  2.118361  2.268156
##      1545      1548      1603      1610      1617      1621      1641      1721
##  3.295376  2.216367  2.062014  2.001166  2.400228  2.862181  2.467976  2.394479
##      1724      1763      1781      1814      1871      1949      1950      1977
##  3.421651  3.592494  6.497757  2.901958  2.493171  6.149192  2.031529  2.474356
##      2015      2045      2066      2078      2137      2156      2177      2194
##  2.116240  2.072020  2.412238  2.099046  2.007372  2.376070  3.208738  2.278276
```

Il y en a énormément, on va donc augmenter le seuil :

```
abs(rstudent(res))[abs(rstudent(res))>3]
```

```
##       228       419       758       984      1025      1043      1323      1545
##  3.508209  3.145482  3.370438  3.686041 10.502563  3.074745  3.074745  3.295376
##      1724      1763      1781      1949      2177
##  3.421651  3.592494  6.497757  6.149192  3.208738
```

On enlève les observations et on ré-entraine le modèle, on vérifie si le nombre d'observations aberrantes a baissé :

```
newdata = models_data[-c(416, 755, 981, 1022, 1040, 1320, 1542, 1721, 1760, 1778, 1946)]
res = lm(TotalPurchases~., models_data)
abs(rstudent(res))[abs(rstudent(res))>3]
```

```
##       63      399      622      630      699      718      984     1025
## 3.448556 3.098463 3.020317 3.823459 3.357423 3.295911 4.288380 5.062151
##     1098     1195     1234     1344     1617     1644     1724     1763
## 3.375435 3.081946 3.127829 3.069115 3.113261 3.127829 4.124962 3.041490
##     1781     1949     2066
## 6.744324 6.463383 3.240798
```

Ce n'est pas le cas, et on obtient toujours une p-valeur < 0.05.

```
shapiro.test(res$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res$residuals
## W = 0.96023, p-value < 2.2e-16
```

On ne pourra malheureusement pas effectuer de régression linéaire, ni d'anova donc.

5 Classification non-supervisée

L'objectif de la classification non-supervisée pour notre dataset est de trouver K classes permettant d'obtenir une segmentation précise de la clientèle. Celle-ci nous permettra donc de connaitre les spécificités de chaque classe et donc de regrouper dans des clusters des clients similaires.

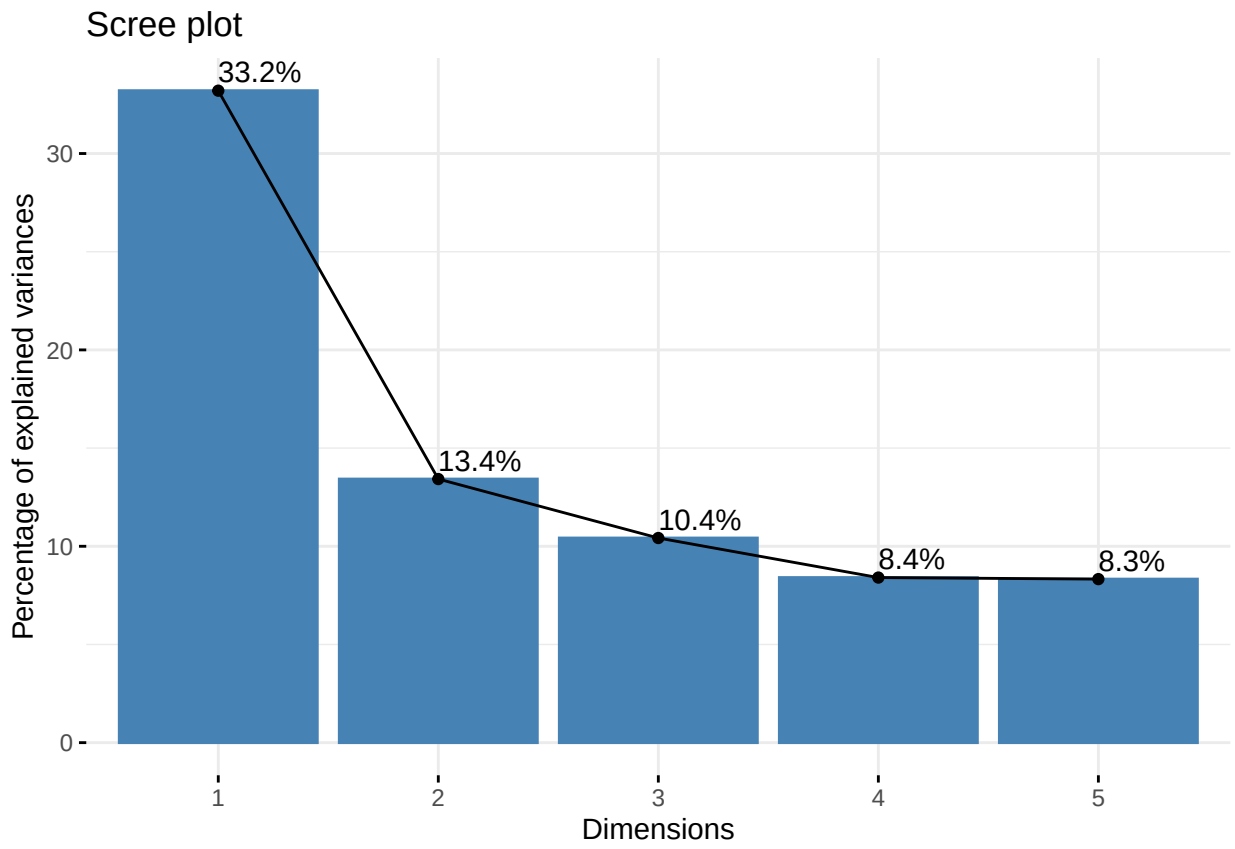
5.1 Composantes principales

Comme nous travaillons avec des variables à la fois quantitatives et qualitatives nous ne pouvons pas utiliser d'**ACP** ou d'**ACM**. Ainsi, nous utiliserons une **AFDM** avec uniquement les variables expliquant les caractéristiques **directes** d'un client : salaire, âge, nombre d'achat, comportement à l'achat etc... Nous exclurons donc les variables **Response** et **Complain** !

```
#load packages
library(FactoMineR)
library(factoextra)
options(ggrepel.max.overlaps = Inf)

#get the sub dataframe for the FAMD
varFAMD = names(data) %in% c("Response", "Complain")
dataFAMD = data[!varFAMD]

#FAMD
famd = FAMD(dataFAMD,graph = FALSE)
```

Regardons maintenant le pourcentage de variance expliqué par chaque dimension :

```
fviz_eig(famd,addlabels = TRUE) #% of variance for each dim
```

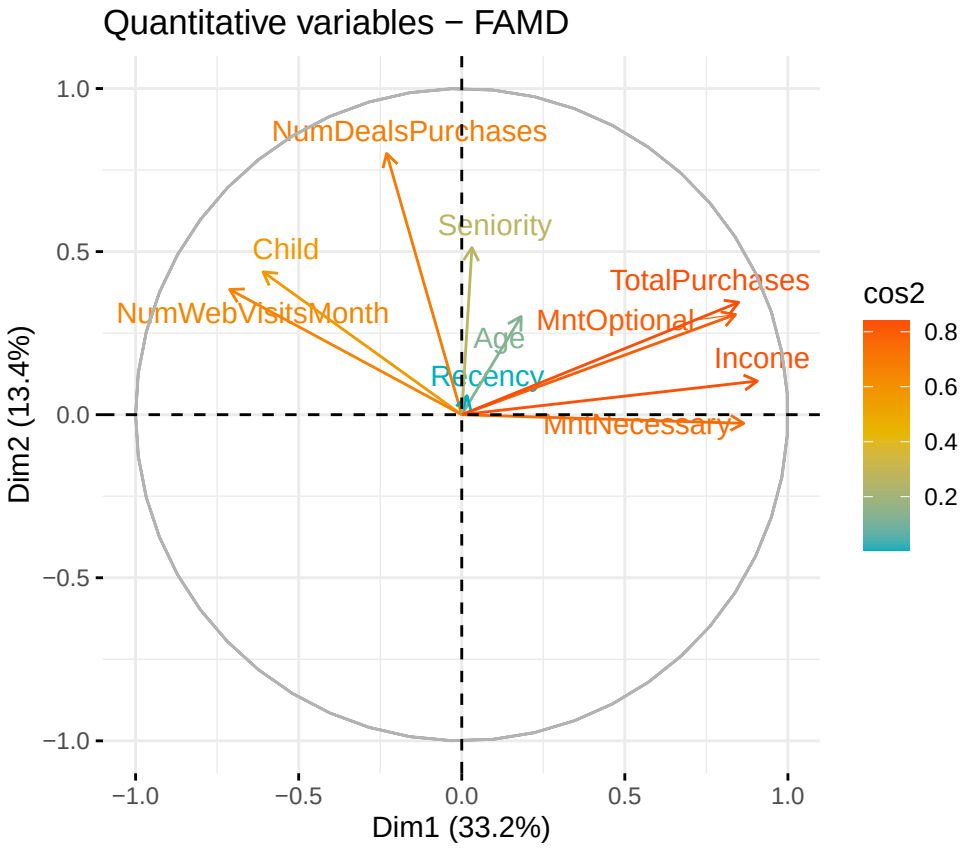


Avec deux dimensions nous expliquons près de la moitié de la variance. Par rapport aux TPs, c'est plutôt faible et nous verrons tout à l'heure que cela va s'expliquer par une qualité de représentation faible tant pour

les variables que pour les individus. Ainsi, il faudrait considérer au plus de 4 dimensions pour expliquer la majorité de la variance. Voyons maintenant la qualité de représentation fourni par l'AFDM :

- Variables quantitatives :

```
gradient = c("#00AFBB","#E7B800","#FC4E07") #for gradient colors
fviz_famd_var(famd,"quanti.var",col.var = "cos2",gradient.cols = gradient,repel = TRUE) #quanti cos2
```

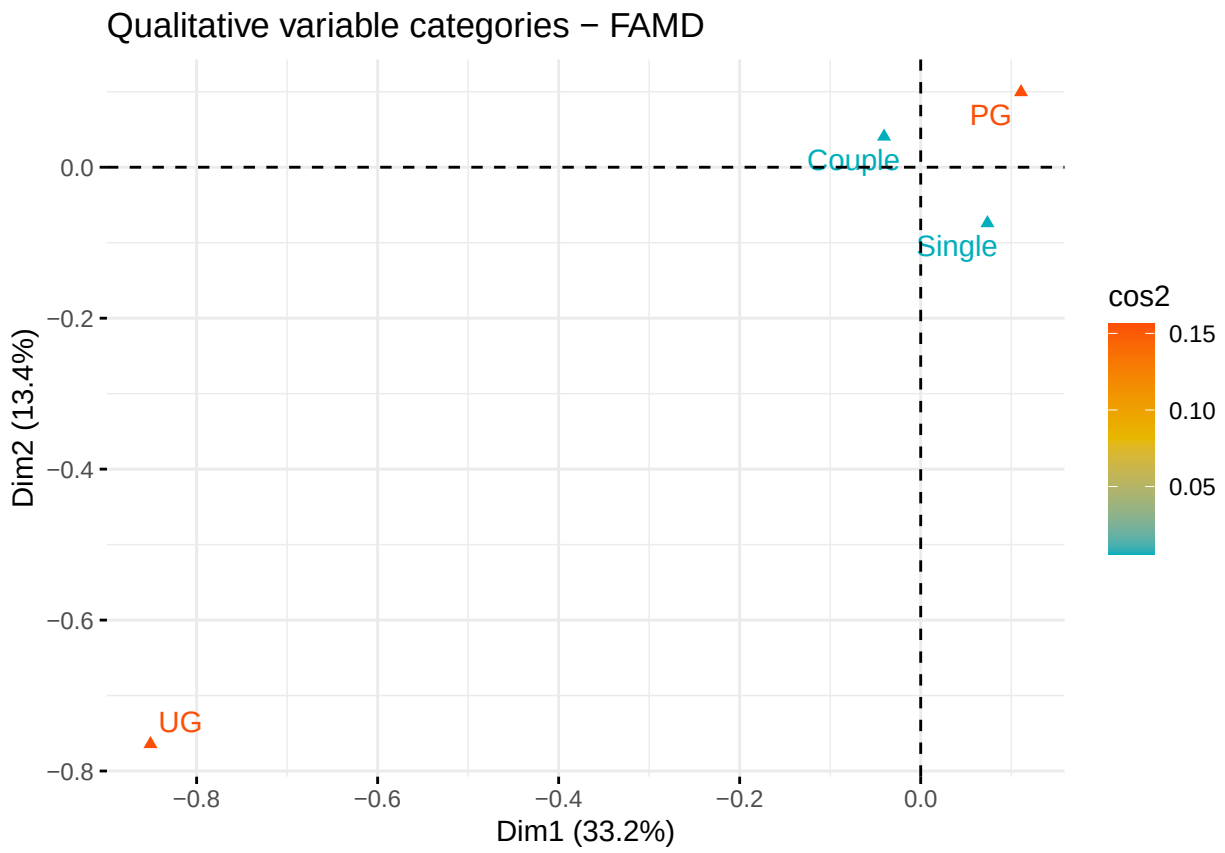


Avec le cercle de corrélation, on remarque que la majorité des variables quantitatives est bien représentée avec un $cos2 > 0.5$. Les variables mal représentées sont **Age, Recency** et **Seniority**. Par ailleurs, on remarque que le salaire et le nombre d'articles achetés sont représentés positivement par la première dimension (vers la droite) : ce qui est logique car avec un important salaire, un client peut se permettre d'effectuer plus d'achats. Bien plus, le nombre d'enfant et de visite web sont représentés de la même manière. Dans une autre mesure, on peut aussi ajouter le nombre d'achat via une promotion : cela peut s'expliquer par le fait qu'en ayant beaucoup d'enfant, on cherchera à acheter plus d'articles en promotion et pour les obtenir la visite régulière du site web est un bon moyen. Finalement, l'ancienneté, bien que moyennement bien représenté, est exprimée positivement par la deuxième dimension (vers le haut) et tend à suivre la représentation du nombre d'achat par promotion : il se peut donc que les clients les plus fidèles aient accès à plus de promotions.

Finalement, on s'attendait à avoir une corrélation entre le nombre d'enfant et les achats effectués, mais ce n'est pas le cas. Il y a aussi aucune différence entre les types de produits achetés (**Optional vs Necessary**) : par exemple, une famille nombreuse aura tendance à acheter plus de produits primaires que secondaires.

- Variables qualitatives :

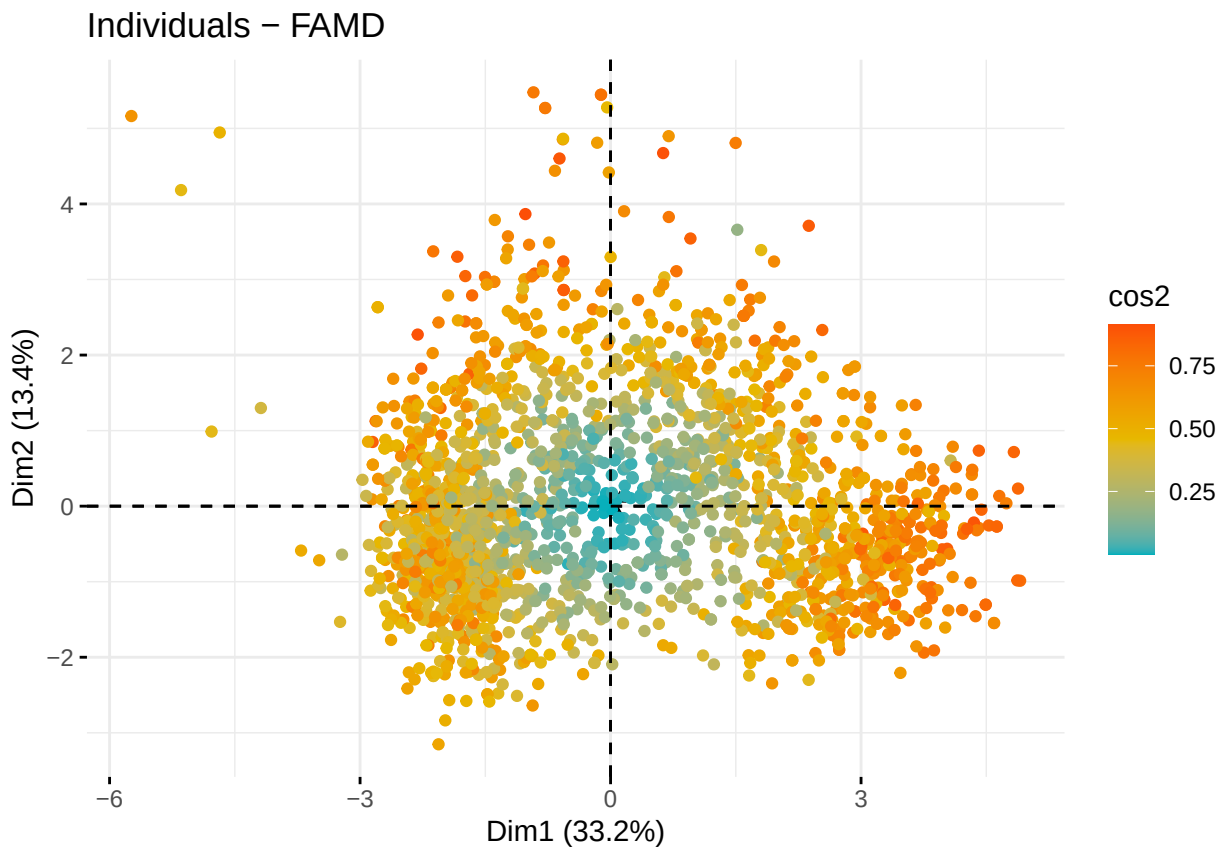
```
fviz_famd_var(famd,"quali.var",col.var = "cos2",gradient.cols = gradient,repel = TRUE) #quali cos2
```



Concernant les variables qualitatives, le statut marital est très mal représenté tout comme pour le niveau d'étude (ne pas se fier au gradient car il a un $cos2 = 0.15$). Cependant, l'AFDM permet d'obtenir une nette distinction entre les deux niveaux d'études.

- Individus :

```
fviz_famd_ind(famd, col.ind = "cos2",gradient.cols = gradient,labels=FALSE) #indiv cos2
```



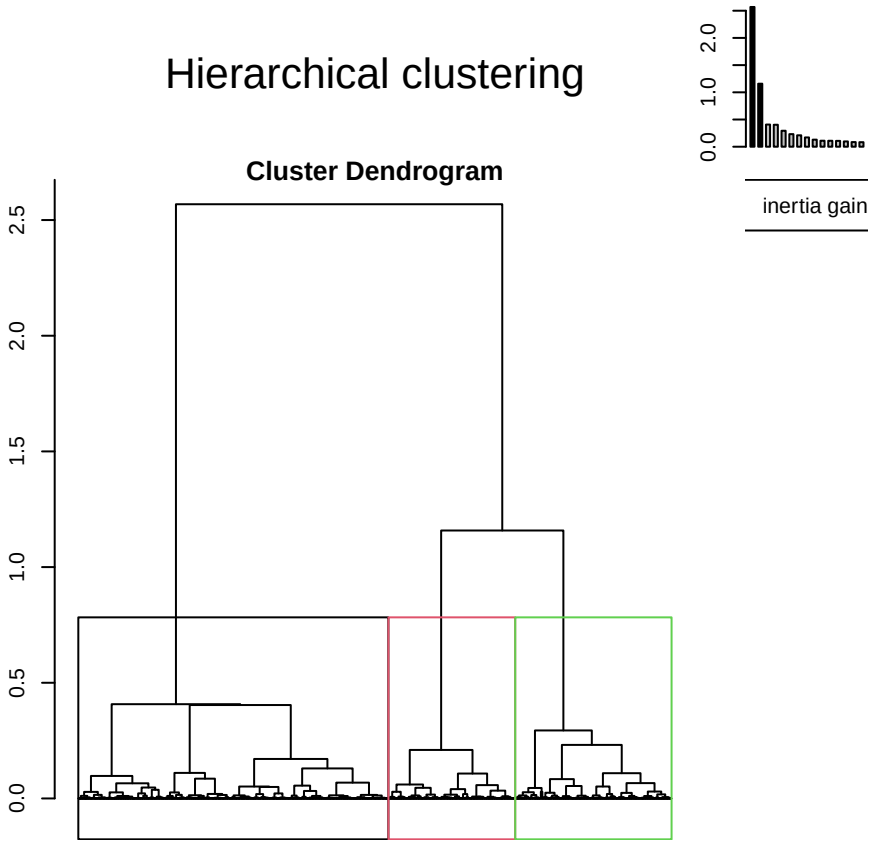
Finalement, pour les individus on remarque qu'il y a une forte concentration autour de l'origine du plan : les individus très proches de l'origine sont mal représentés, ce qui est tout à fait normal. Par conséquent, on observe un gradient du *cos*2, i.e, plus on s'éloigne de l'origine et mieux est la qualité de représentation des individus. Cependant, à vu d'oeil, il est très difficile de distinguer les clusters au vu du nombre important d'individus.

Avec ces premières observations, on peut déjà s'attendre à une division de la clientère au niveau du salaire et du nombre d'enfants. Bien plus, on sait que le statut marital et le niveau d'étude n'auront pas un important impact.

5.2 Clustering

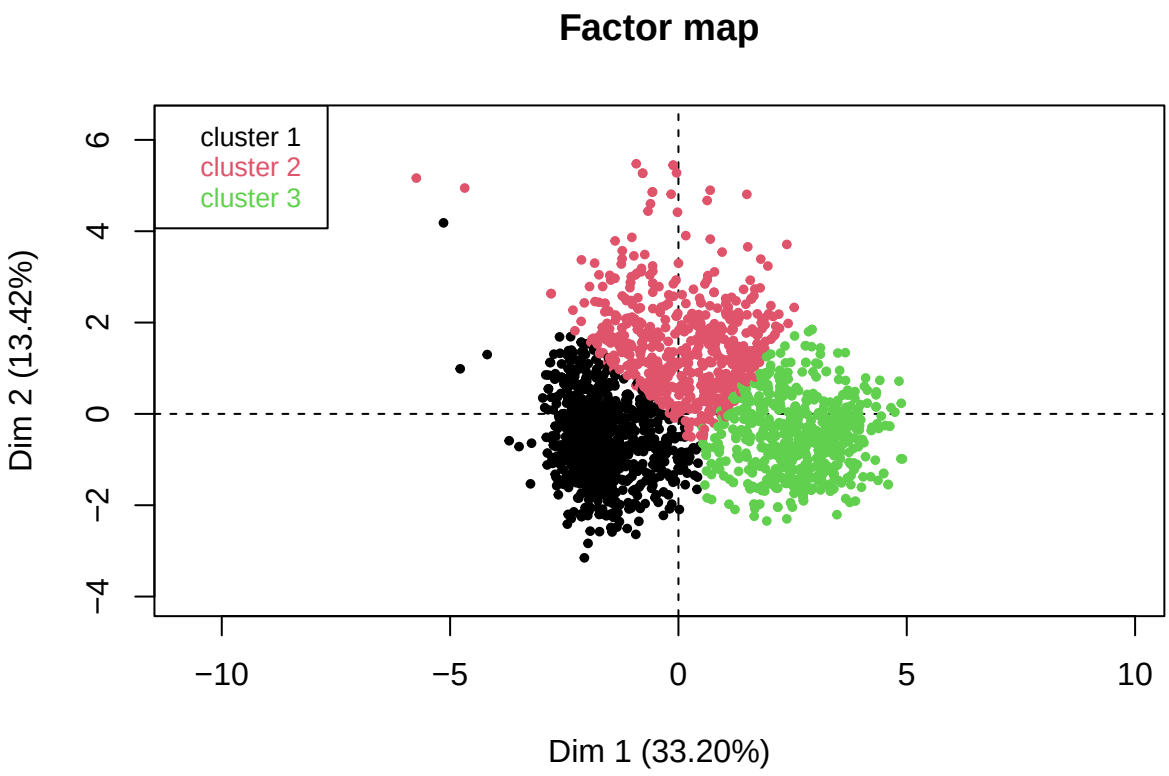
Pour effectuer le clustering, nous allons faire une **classification hiérarchique sur les composantes principales (HCPC)** obtenues à l'aide de l'AFDM. Comme nous avons des données mixtes avec pas mal de variables, cette méthode est très adaptée. Nous effectuerons aussi une consolidation **kmeans** en plus de la **CAH** et nous laisserons le choix du nombre de clusters à la fonction :

```
#HCPC
hcpc = HCPC(famd,consol = TRUE,iter.max = 1000,nb.clust = -1,graph = FALSE)
#plot dendrogram
plot(hcpc,choice = "tree",labels=FALSE)
```



On peut donc segmenter la clientèle en 3 clusters ! Affichons les :

```
#plot clusters
plot(hcpc,choice = "map",ind.names = FALSE,draw.tree = FALSE)
```



Les clusters obtenus sont très distincts sauf au niveau des frontières où il est très difficile de voir la différence car certains individus se chevauchent.

Passons maintenant à l'étude des caractériques de chaque cluster :

```
#get clusters spec
print(hcpc$desc.var$quanti)
```

```
## $`1`
##                       v.test Mean in category Overall mean sd in category
## NumWebVisitsMonth  18.469736         6.370775 5.335299e+00   1.917457e+00
## Child              14.164028         1.195825 9.491833e-01   6.939260e-01
## Seniority          -8.475317         8.835984 8.970962e+00   6.672824e-01
## NumDealsPurchases  -9.920280         1.879722 2.314428e+00   1.108635e+00
## Age               -10.296729        50.333002 5.320236e+01   1.137231e+01
## MntNecessary      -29.524071        36.804175 2.293598e+02   4.373997e+01
## MntOptional       -35.502918        63.440358 3.771506e+02   6.963323e+01
## Income            -35.615749     34462.887674 5.161773e+04   1.241867e+04
## TotalPurchases    -39.524449         5.956262 1.255672e+01   2.713283e+00
##                     Overall sd       p.value
## NumWebVisitsMonth 2.411333e+00  3.618486e-76
## Child             7.489583e-01  1.529686e-45
## Seniority         6.849900e-01  2.344404e-17
## NumDealsPurchases 1.884734e+00  3.398002e-23
## Age               1.198569e+01  7.289833e-25
## MntNecessary      2.805161e+02 1.413722e-191
```

```
## MntOptional       3.800516e+02 4.431234e-276
## Income            2.071679e+04 7.992566e-278
## TotalPurchases    7.182667e+00  0.000000e+00
##
## $`2`
##                       v.test Mean in category Overall mean sd in category
## NumDealsPurchases 29.323685         4.277873 2.314428e+00   2.318908e+00
## TotalPurchases    16.458276        16.756432 1.255672e+01   4.685552e+00
## Seniority         12.752812         9.281304 8.970962e+00   6.196693e-01
## Child             12.546569         1.283019 9.491833e-01   5.802671e-01
## MntOptional       11.278848       529.435678 3.771506e+02   3.050280e+02
## NumWebVisitsMonth 10.562424         6.240137 5.335299e+00   1.816817e+00
## Age                9.039530        57.051458 5.320236e+01   9.670245e+00
## Income             6.149017     56143.351630 5.161773e+04   1.152517e+04
## MntNecessary      -3.397032       195.506003 2.293598e+02   1.605984e+02
##                     Overall sd       p.value
## NumDealsPurchases 1.884734e+00 5.175734e-189
## TotalPurchases    7.182667e+00  7.315459e-61
## Seniority         6.849900e-01  3.006978e-37
## Child             7.489583e-01  4.151094e-36
## MntOptional       3.800516e+02  1.669174e-29
## NumWebVisitsMonth 2.411333e+00  4.450181e-26
## Age               1.198569e+01  1.573468e-19
## Income            2.071679e+04  7.796445e-10
## MntNecessary      2.805161e+02  6.812107e-04
##
## $`3`
##                       v.test Mean in category Overall mean sd in category
## MntNecessary       36.127796     5.764293e+02 2.293598e+02    268.7531544
## Income             33.505250     7.538901e+04 5.161773e+04   9696.3000317
## MntOptional        28.335319     7.459480e+02 3.771506e+02    320.1897363
## TotalPurchases     27.707921     1.937236e+01 1.255672e+01      4.1947955
## Age                 2.545369     5.424715e+01 5.320236e+01     13.6390420
## Seniority          -3.128961     8.897561e+00 8.970962e+00      0.6802036
## NumDealsPurchases -17.819943     1.164228e+00 2.314428e+00      0.5793366
## Child             -28.067676     2.292683e-01 9.491833e-01      0.4242127
## NumWebVisitsMonth -30.898077     2.783740e+00 5.335299e+00      1.6291239
##                     Overall sd       p.value
## MntNecessary      2.805161e+02 8.305296e-286
## Income            2.071679e+04 4.041735e-246
## MntOptional       3.800516e+02 1.269574e-176
## TotalPurchases    7.182667e+00 5.604791e-169
## Age               1.198569e+01  1.091622e-02
## Seniority         6.849900e-01  1.754258e-03
## NumDealsPurchases 1.884734e+00  4.948670e-71
## Child             7.489583e-01 2.431086e-173
## NumWebVisitsMonth 2.411333e+00 1.267673e-209
```

```
print(hcpc$desc.var$category)
```

```
## $`1`
##                Cla/Mod  Mod/Cla  Global     p.value   v.test
## Education=UG 67.71654 17.09742 11.5245 5.59318e-14  7.51727
## Education=PG 42.76923 82.90258 88.4755 5.59318e-14 -7.51727
```

```
##
## $`2`
##                          Cla/Mod   Mod/Cla   Global      p.value    v.test
## Education=PG           28.30769 94.682676 88.47550 5.452106e-09  5.832751
## Marital_Status=Couple  28.03935 68.439108 64.56443 2.210671e-02  2.288529
## Marital_Status=Single  23.55954 31.560892 35.43557 2.210671e-02 -2.288529
## Education=UG           12.20472  5.317324 11.52450 5.452106e-09 -5.832751
##
## $`3`
##                Cla/Mod   Mod/Cla  Global     p.value    v.test
## Education=PG  28.92308 91.707317 88.4755 0.002486484  3.024981
## Education=UG  20.07874  8.292683 11.5245 0.002486484 -3.024981
```

- **Cluster 1** : clientèle à faible revenu, avec beaucoup d'enfants et un niveau d'éducation de premier cycle. Elle effectue beaucoup de visite sur le site web et n'achète que très peu. C'est une clientèle récente et qui semble ne pas avoir accès énormément aux offres de promotions. Bien plus, elle favorise les achats primaires (i.e viande, poisson etc...).
- **Cluster 2** : ce cluster représente la clientèle de classe moyenne : elle dispose d'un revenu moyen, a une famille (en couple) et un niveau d'étude élevé. C'est une clientèle qui effectue beaucoup d'achats de produits secondaires (i.e bonbons, vins etc...) via des promotions. Elle visite aussi pas mal de fois le site web et dispose d'une importante ancienneté.
- **Cluster 3** : ce cluster représente la clientèle aisée avec un haut revenu et qui a un niveau d'étude important. Elle représente la majorité des achats effectués et semble ne pas avoir de préférence pour les types de produits. Bien plus, elle a très peu d'enfants voir pas du tout. Enfin, c'est aussi une clientèle récente.

Finalement, il est intéressant de noter le lien direct entre niveau d'étude et le salaire mais aussi celui avec le nombre d'enfants.

6 Classification supervisée

L'objectif dans cette section est de pouvoir prédire la colonne **Response** en se basant sur les caractérisitiques du client (âge, niveau d'étude, salaire) de manière à pouvoir cibler une certaine clientèle lors de campagne marketing.

6.1 Préparation

Nous allons préparer deux datasets : un pour l'entrainement et l'autre pour la prédiction. Nous utiliserons aussi un ratio de 80/20 :

```r
#get dataset for training and testing
set.seed(1)
n <- nrow(data)
p <- ncol(data)-1
test.ratio <- .2 # ratio of test/train samples
n.test <- round(n*test.ratio)
tr <- sample(1:n,n.test)
data.test <- data[tr,]
data.train <- data[-tr,]
```

Vérifions maintenant la distribution au sein de la colonne **Response** dans le dataset train :

```
#check the distribution of response
print(table(data.train$Response))
```

```
##
##    0    1
## 1495  268
```

On remarque que nous avons un dataset très déséquilibré, nous allons donc utiliser le package **SMOTE** pour l'équilibrer :

```
#balance the data.train set
library(DMwR)
data.train = SMOTE(Response ~., data.train)
```

6.2 AFD, LDA et QDA

Pour utiliser une **AFD**, il nous faut dataset avec des échantillons de taille équiprobable. Vérifions si c'est le cas :

```
print(table(data.train$Response))
```

```
##
##    0    1
## 1072  804
```

On ne peut pas donc utiliser une AFD. Vérifions maintenant la normalité de nos données en prenant une variable quantitative aléatoire pour voir si on peut utiliser une **LDA/QDA** :

```
#shapiro test
print(shapiro.test(data.train$Income))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data.train$Income
## W = 0.97211, p-value < 2.2e-16
```

```
#log transformation
print(shapiro.test(log(data.train$Income)))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  log(data.train$Income)
## W = 0.92265, p-value < 2.2e-16
```

Nous obtenons dans les deux cas (même lors d'une transformation log) une p-valeur inférieure à 5%, nos données ne sont donc pas gaussiennes. Cependant, on peut tout de même essayer ces deux algorithmes car nous avons notre dataset test pour vérifier.

Une LDA ou QDA fonctionne uniquement avec des variables quantitatives, or nos variables qualitatives ont deux modalités, on peut donc les tranformer en **dummies** :

```
#create data frame with dummies for categorial variables
matrix.train.dum = model.matrix(Response ~., data.train)[,-1]
matrix.test.dum = model.matrix(Response ~., data.test)[,-1]
data.train.dum = data.frame(matrix.train.dum)
data.test.dum = data.frame(matrix.test.dum)
data.train.dum$Response = data.train$Response
data.test.dum$Response = data.test$Response
```

Effectuons maintenant une LDA et QDA :

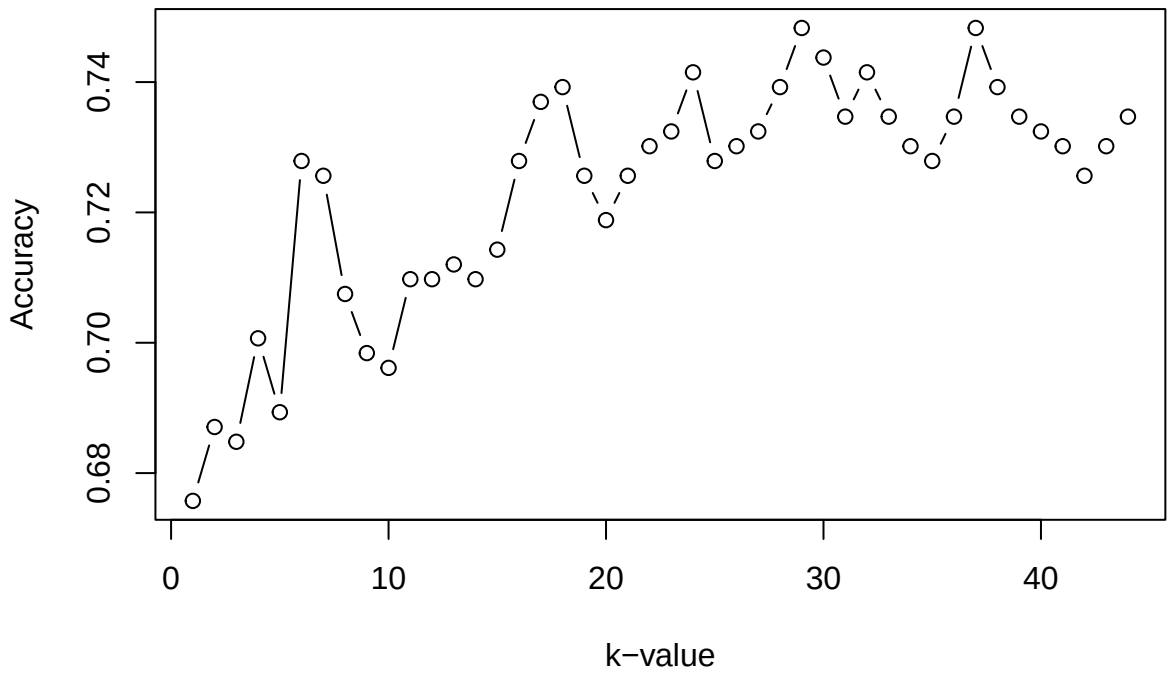
```
#lda/qda
library(MASS)
LDA = lda(Response~.,data=data.train.dum)
QDA = qda(Response~.,data=data.train.dum)

#predict
predict.LDA = predict(LDA,newdata=data.test.dum)$class
predict.QDA = predict(QDA,newdata=data.test.dum)$class

#get acc
LDA.acc = mean(predict.LDA  == data.test.dum$Response)
QDA.acc = mean(predict.QDA == data.test.dum$Response)

#get auc
library(pROC)
predict.LDA = predict(LDA,newdata=data.test.dum)$posterior[,2]
predict.QDA = predict(QDA,newdata=data.test.dum)$posterior[,2]
LDA.roc = invisible(roc(data.test.dum$Response,predict.LDA))
QDA.roc = invisible(roc(data.test.dum$Response,predict.QDA))
```

6.3 KNN

Dans cette partie nous allons mettre en oeuvre la **méthode des k plus proches voisins**. Elle nécessite des données normalisées et une valeur de K, représentant le nombre de plus proches voisins. Pour cela nous allons lancer l'algorithme avec plusieurs valeurs de K jusqu'à $K = \sqrt{N}$ avec N le nombre d'échantillon dans le data set train. Nous choisirons le K qui minimise le plus l'erreur de classe :

```
#KNN
suppressPackageStartupMessages(library(class))
#scale
data.train.dum.scale = data.frame(scale(matrix.train.dum,center=TRUE,scale=TRUE))
data.test.dum.scale = data.frame(scale(matrix.test.dum,center=TRUE,scale=TRUE))

i=1
knn.tmp=1
l = round(sqrt(nrow(data.train.dum.scale))) + 1
```

```r
for(i in 1:l) #test multiple k value to get the best accuracy
{
  knn.i = knn(train=data.train.dum.scale,test=data.test.dum.scale,cl=data.train$Response,k=i)
  knn.tmp[i] = sum(data.test$Response == knn.i)/nrow(data.test)
}

#get acc
knn.acc = max(knn.tmp)

#get best k value
knn.k = which(knn.tmp == knn.acc)[1]

#get auc
knn.opt = knn(train=data.train.dum.scale,test=data.test.dum.scale,cl=data.train$Response,k=knn.k,prob=T
knn.roc = roc(data.test.dum$Response,attributes(knn.opt)$prob)

#plot k-value
plot(knn.tmp,type="b",xlab="k-value",ylab="Accuracy")
```

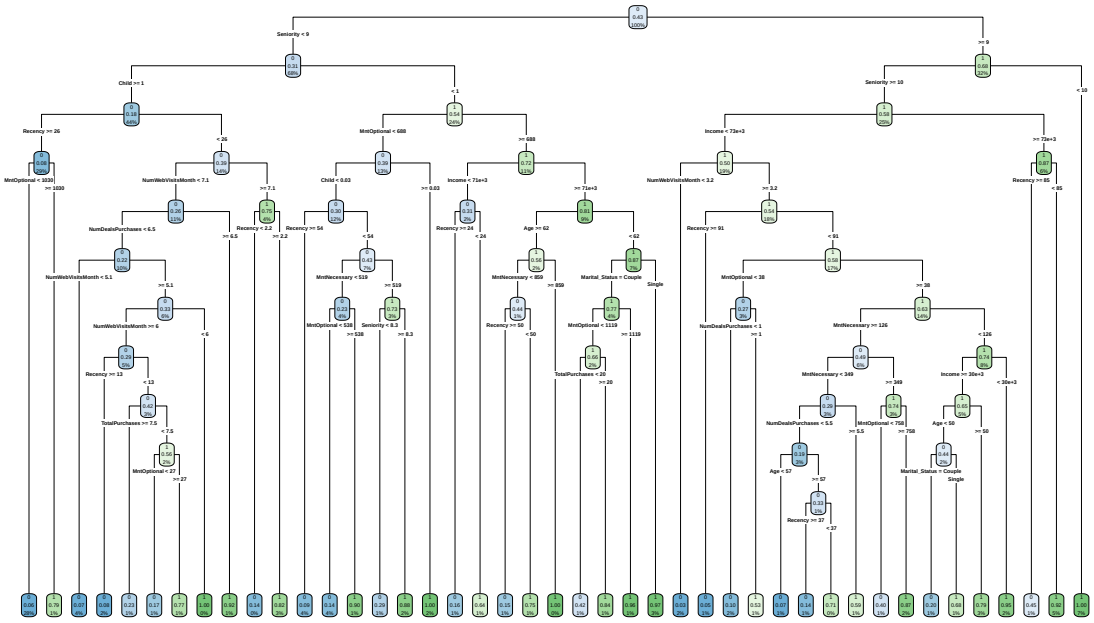


Ici, nous obtenons $K = 37$.

6.4 CART et Random Forest

L'implémentation de ces deux méthodes est très directe. On laissera les fonctions décidées pour la valeur de *cp* pour l'élagage de l'arbre avec CART :

```
#CART
library(rpart)
library(rpart.plot)
cart = rpart(Response~.,data.train,control=rpart.control(cp=0))

#get best cp
cp.opt = cart$cptable[which.min(cart$cptable[,"xerror"]),"CP"]

#get best tree
cart.opt = prune(cart,cp.opt)

#predict
predict.cart = predict(cart.opt, newdata=data.test, type="class")

#get acc
cart.acc = mean(predict.cart == data.test$Response)

#get auc
predict.cart = predict(cart.opt, data.test, type="prob")[,2]
cart.roc = roc(data.test$Response,predict.cart)

#plot tree
rpart.plot(cart.opt, type=4)
```

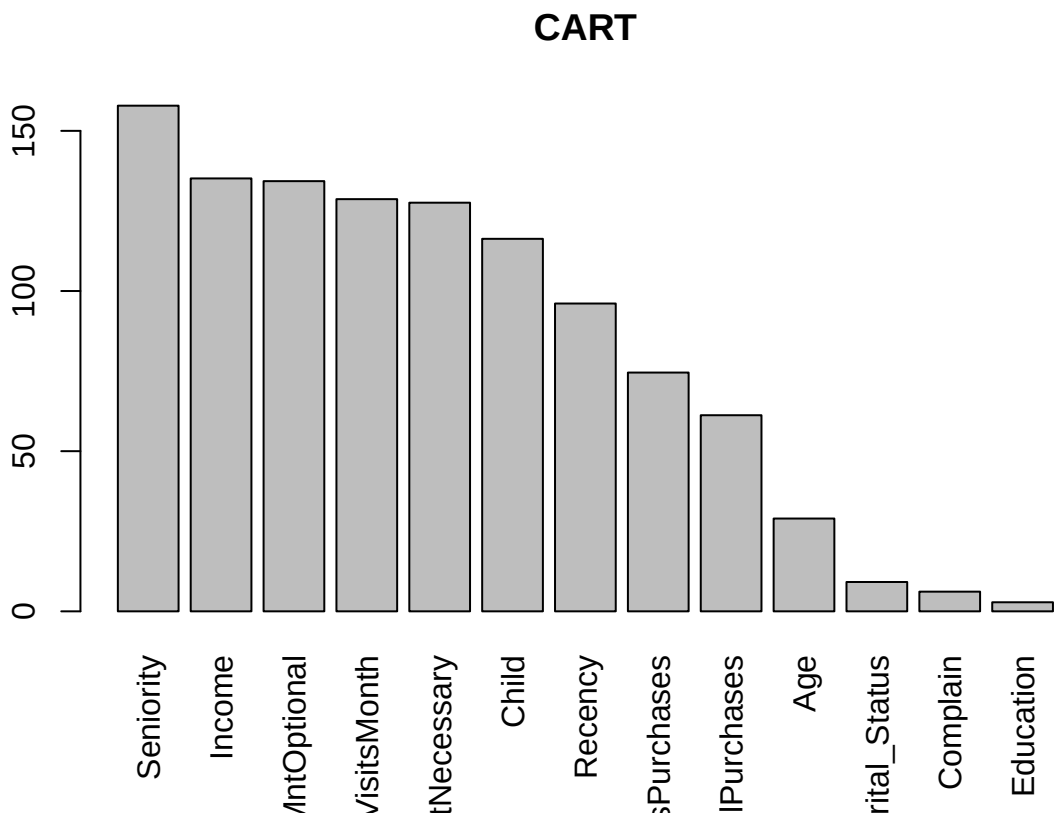


Il est difficile de lire l'arbre car il y a énormément de noeuds ! Cependant, il est possible de l'enregistrer en image avec une meilleure définition.

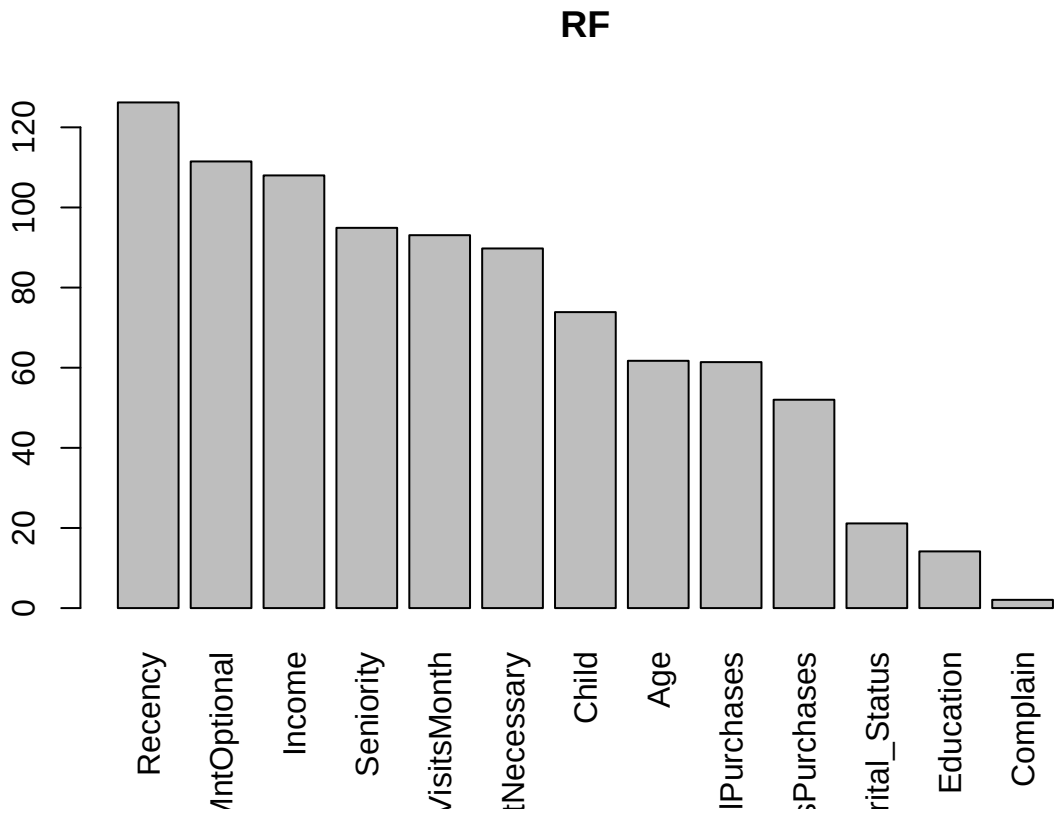
```
#RANDOM FOREST
library(randomForest)
RF = randomForest(Response~.,data.train)

#predict
predict.RF = predict(RF, newdata=data.test, type="class")

#get acc
RF.acc = mean(predict.RF == data.test$Response)

#get auc
predict.RF = predict(RF, data.test, type="prob")[,2]
RF.roc = roc(data.test$Response,predict.RF)
```

On peut maintenant obtenir l'importance de chaque variable :

```
#get importance (CART)
barplot(cart.opt$variable.importance, main = "CART", las=3)
```




```
#get importance (RF)
ord=order(RF$importance,decreasing = TRUE)
barplot(RF$importance[ord],names.arg=rownames(RF$importance)[ord], main = "RF",las=3)
```



On remarque qu'en général, pour les deux méthodes, les variables qui permettent d'expliquer le plus **Response** sont **Income**, **MntOptional**, **MntNecessary** et **Seniority**.

6.5 Régression logistique

Comme nous avons beaucoup d'échantillons dans le dataset train, nous allons choisir λ par cross-validation :

```
#LASSO
library(glmnet)
#v-fold because we have a lot of rows
lasso.cv = cv.glmnet(matrix.train.dum,data.train$Response,family="binomial",type.measure = "class")

#predict
predict.lasso = predict(lasso.cv,newx = matrix.test.dum,s = 'lambda.min',type = "class")

#get acc
lasso.cv.acc = mean(predict.lasso == data.test$Response)

#get auc
predict.lasso = predict(lasso.cv,newx = matrix.test.dum,s = 'lambda.min',type = "response")
lasso.cv.roc = roc(data.test$Response,predict.lasso)
```

On peut maintenant obtenir les **odds ratio** :

```r
#get odd ratio
opt.lamb = lasso.cv$lambda.min
print(exp(coef(lasso.cv,s = opt.lamb)))
```

```
## 14 x 1 Matrix of class "dgeMatrix"
##                                s1
## (Intercept)          0.00134875
## Age                  1.00000000
## Child                0.78524756
## Complain1            1.74546563
## EducationUG          2.14995149
## Income               1.00000000
## Marital_StatusSingle 2.03131181
## MntNecessary         1.00137416
## MntOptional          1.00111633
## NumDealsPurchases    1.00000000
## NumWebVisitsMonth    1.18859606
## Recency              0.97911104
## Seniority            1.75254013
## TotalPurchases       1.00000000
```

Pour expliquer **Response**, nous avons des résultats totalement différents par rapport à la section précédente. En effet, on remarque que les variables qualitatives ont le plus d'influence positivement : **Marital_Status** et **Education**. Ainsi, avec un client ayant un niveau d'étude faible et sans couple, on aura beaucoup plus de chance de lui faire accepter l'offre (on multiplie par $\approx 4-5$ nos chances\$). De même que pour CART et RF, l'ancienneté joue un important rôle : les clients les plus fidéles auront tendance à accepter les offres. Finalement, on voit que les clients qui visitent le plus souvent le site ont tendance à accepter les offres. Finalement, on remarqu'aussi que les clients qui ont effectué une plainte ont aussi tendance à accepter les offres (il se peut que l'entreprise compense l'inconvénience par une promotion).

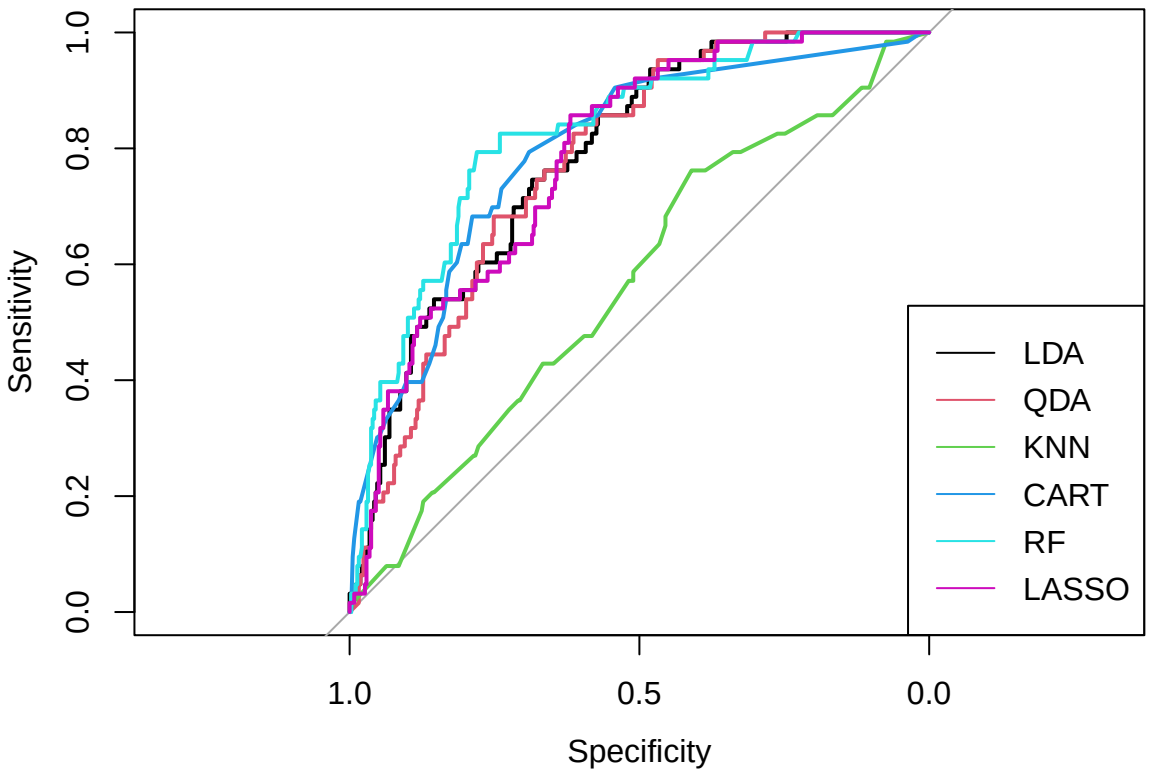
6.6 Comparaison

```r
#table
result=matrix(NA, ncol=6, nrow=2)
rownames(result)=c('accuracy', 'AUC')
colnames(result)=c('LDA','QDA','KNN','CART','RF','LASSO')
result[1,]= c(LDA.acc,QDA.acc,knn.acc,cart.acc,RF.acc,lasso.cv.acc)
result[2,]=c(LDA.roc$auc,QDA.roc$auc,knn.roc$auc,cart.roc$auc,RF.roc$auc,lasso.cv.roc$auc)
print(result)
```

```
##                LDA       QDA       KNN      CART        RF     LASSO
## accuracy 0.7687075 0.8049887 0.7482993 0.7845805 0.8321995 0.7732426
## AUC      0.7862182 0.7763500 0.5687831 0.7948686 0.8244520 0.7858823
```

```r
#plot
plot(LDA.roc, xlim=c(1,0))
plot(QDA.roc, add=TRUE, col=2)
plot(knn.roc, add=TRUE, col=3)
plot(cart.roc, add=TRUE, col=4)
plot(RF.roc,add=TRUE, col=5)
```

```r
plot(lasso.cv.roc, add=TRUE, col=6)
legend('bottomright', col=1:6, paste(c('LDA','QDA', 'KNN','CART','RF','LASSO')), lwd=1)
```



On remarque que l'ensemble des méthodes arrivent à prédire correctement la variable **Response**. Cependant, l'algorithme de KNN a une AUC très faible par rapport aux autres méthodes. Étonnement, LDA et QDA ont réussi à fonctionner malgrè la non-linéarité des données. Finalement, la meilleure méthode est la **Random Forest** tant pour l'accuracy que pour l'AUC.

Par ailleurs, il est intéressant de noter comment chaque méthode donne une interprétation différentes aux données (par exemple lasso vs random forest).

7 Conclusion

En conclusion, nous avons réussi à segmenter la clientèle de l'entreprise en 3 profils de clients distincts. Bien plus, on arrive à prédire la réponse d'un client suite à une offre de promotion à 83%.

Si nous devions conseiller une stratégie marketing à cette entreprise : elle doit premièrement essayer de fidéliser les deux clientèles qu'on a décrit auparavant (modestes et riches). Pour cela, elle peut s'appuyer sur le modèle de prédiction qu'on a implémenté pour cibler les clients susceptibles d'accepter une offre de promotion ou bien effectuer une étude plus poussée à l'aide des profils des clusters qu'on a dressé. Il faut aussi offrir plus de promotions pour les clients modestes même s'ils n'ont pas d'ancienneté et se focaliser sur la clientèle riche car elle représente la majorité des achats. Finalement, il ne faut pas non plus négliger la clientèle de la classe moyenne qui représente le socle important et fidèle des clients de l'entreprise.