Master 2 Data Science

Université Claude Bernard Lyon 1

Projet Data Mining

Speed Dating Experiment

réalisé par
Orhan Yazar, Florian Robinet, Hugo Duportet, Adam Benharrats

Table des matières

1 Introduction **2**

2 Analyse descriptive **3**
2.1 Qui participe ? . . . 3
2.1.1 Réussite en fonction de la différence d'âge . . . 4
2.1.2 Emploi et genre . . . 5
2.1.3 Ethnies et genres . . . 6
2.1.4 Activités des participants . . . 7
2.1.5 Attente des participants . . . 7
2.1.6 Salaire des participants . . . 8
2.1.7 Bilan . . . 8

3 Fouille de motifs **9**
3.1 Les données . . . 9
3.2 La méthode . . . 10
3.2.1 Règles d'association . . . 10
3.2.2 Indice de support et de confiance . . . 10
3.2.3 Indice lift . . . 10
3.2.4 Algorithme Apriori . . . 11
3.3 Code et résultats . . . 11

4 Clustering **16**
4.1 Kmeans . . . 16
4.2 Hiérarchique . . . 20

5 Classification **23**
5.1 Random Forest . . . 23
5.2 Gradient Boosting . . . 25
5.3 Comparaison . . . 28

6 Conclusion **28**

7 Annexe **29**
7.1 Code Python Analyse Descriptive . . . 29

1 Introduction

Le choix d'un partenaire conjugal est l'un des choix les plus importants que les gens font. Contrairement aux problèmes de calcul mathématique, où il pourrait y avoir un type de solution bien définie, le choix d'un partenaire est bien plus complexe. Dans les sociétés orientale et occidentale, ce choix est habituellement fait après une période d'apprentissage, au cours de laquelle les personnes engagées voient une relation de confiance croître entre eux et retrouve une part de leur envie chez l'autre. Lors de plusieurs sessions de speed-dating aux États-Unis, des scientifiques ont recueillis les avis et envies de nombreux participants de tous horizons (sexe, age, ethnie,...) dans le but au départ d'étudier l'influence de l'ethnie dans la décision ou non de revoir le partenaire rencontré. Le site de data science Kaggle a proposé de pousser plus loin cette analyse et a rendu l'accès à ce jeu de données libre afin d'élargir les champs d'interprétations de ces expérimentations. Nous pouvons en effet nous demander quels sont les attributs les plus significatifs dans la réussite d'une rencontre d'un partenaire de speed dating parmi toutes les informations disponibles dans ce vaste jeu de données. Dans cette optique, nous étudierons dans un premier temps les données en elles-mêmes et ce qu'elles représentent concrètement, avant de modéliser plusieurs modèles de fouilles et d'apprentissage afin d'en tirer le plus d'informations possibles.

2 Analyse descriptive

Nous nous intéressons dans cette première partie aux questions naturelles que posent cette étude, comment les participants se répartissent selon leur race et leur age, leur catégorie socio-professionnelle ainsi qu'un bref aperçu des relations entre ces attributs et les matches ou non lors d'une rencontre.

2.1 Qui participe ?

Un total de 552 participants a participé aux différentes sessions de speed dating. On note une moyenne d'age de 26.6 ans pour les hommes et de 26.1 pour les femmes.

FIGURE 1 – Proportion de participants selon leur âges et leur genre

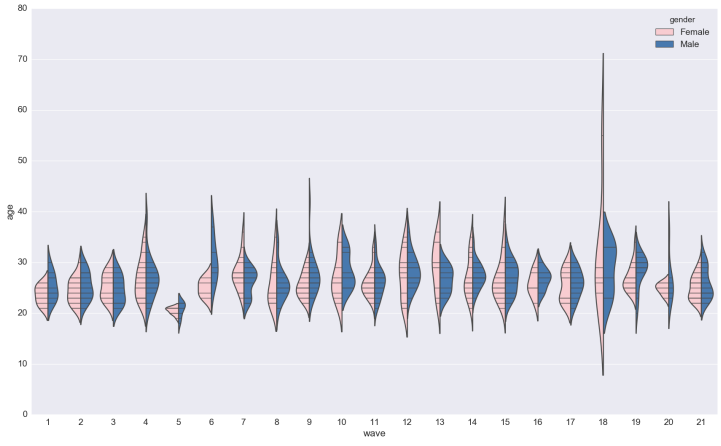


FIGURE 2 – Âge et genre des participants pour les différentes sessions

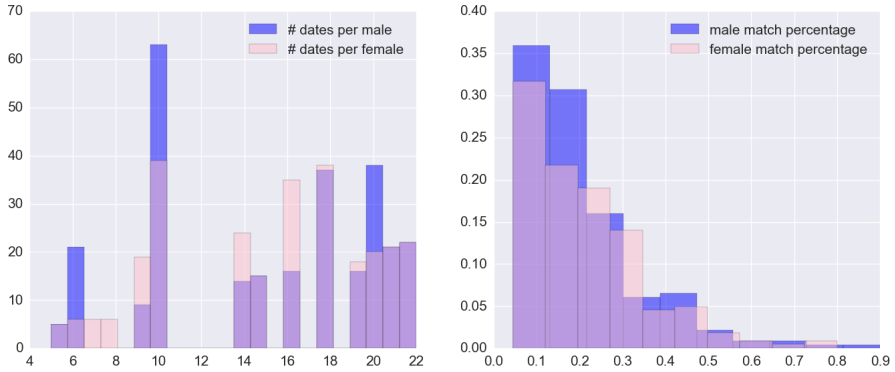


FIGURE 3 – Histogramme du nombres de rencontres par participants et genre — Distribution du nombre de 'matches' par genre

On peut simplement se demander qui des femmes ou des hommes ont le plus de chances de matcher. Les femmes ont une moyenne (0.21 %) de matches légèrement supérieures à celles des hommes (0.20 %). Cette différence n'est pas significative, mais le graphe montre que les femmes ont généralement tendance à avoir plus de matches que les hommes.

2.1.1 Réussite en fonction de la différence d'âge

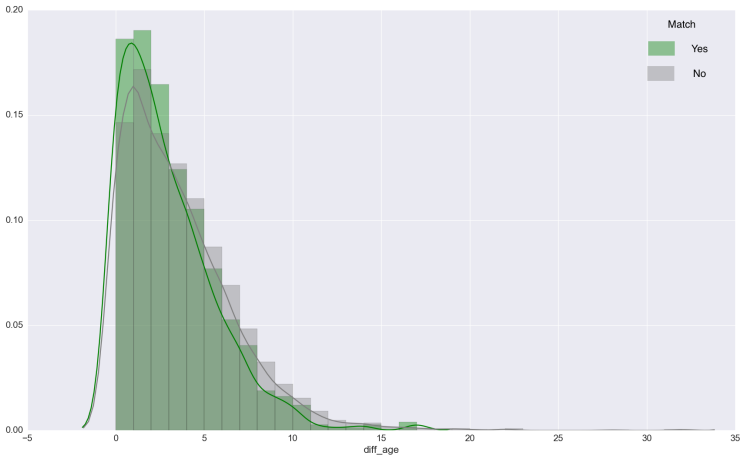


FIGURE 4 – Histogramme du nombres de matches selon la différence d'âge

Plus la différence d'âge est conqéquente entre deux participants, moins il y a de chances de match, cependant, une différence de moins de 3 ans n'influence pas significativement la décision des participants.

2.1.2 Emploi et genre

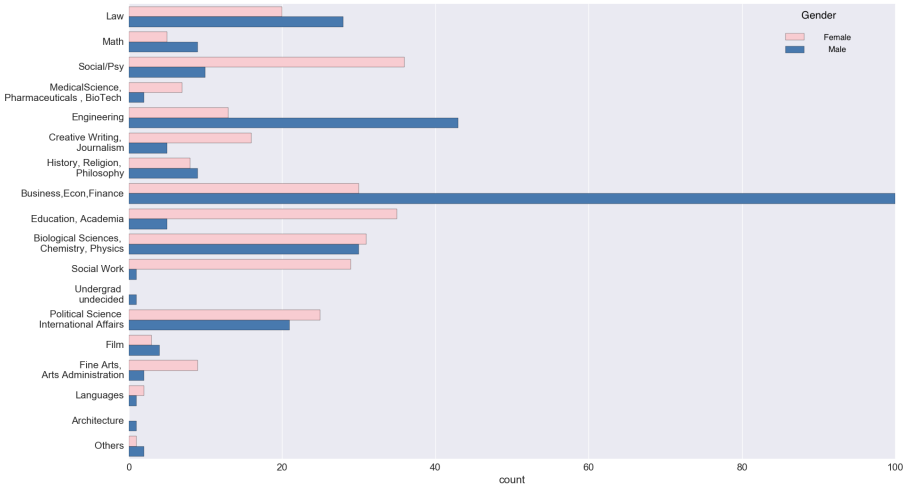


FIGURE 5 – Histogramme du nombre de participant selon leurs études

Sur l'ensemble des participants, les femmes privilégient les études dans les secteurs médicaux, des arts, de la création, du journalisme et de l'administration sociale alors que les hommes privilégient les secteurs de l'entreprise, de l'économie, de la finance ainsi que de l'ingénierie.

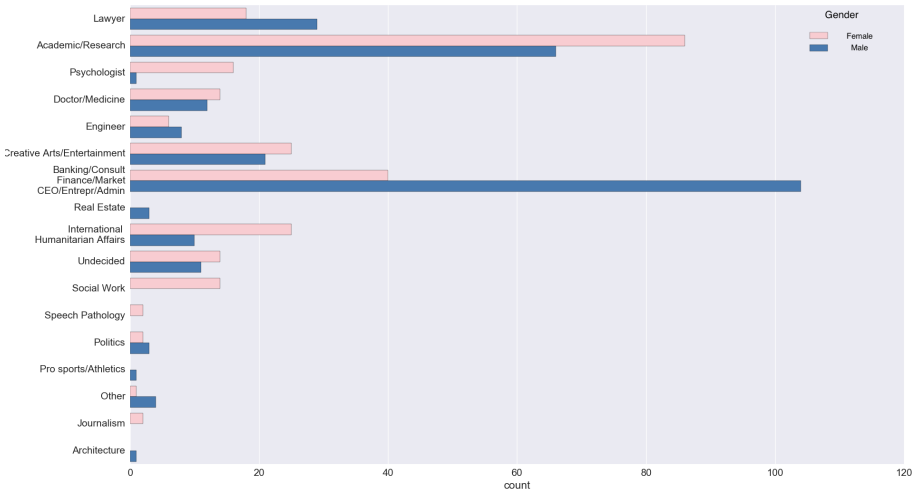


FIGURE 6 – Histogramme du nombre de participant selon leur carrière

2.1.3 Ethnies et genres

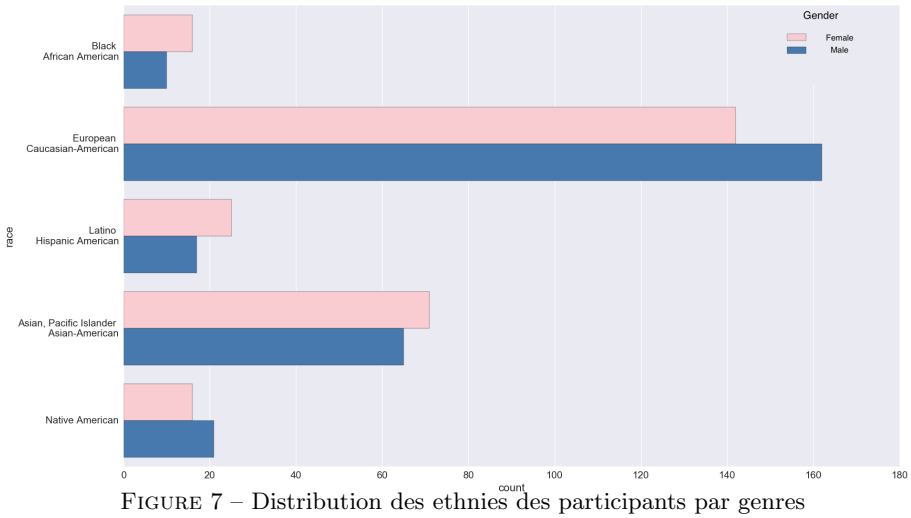


FIGURE 7 – Distribution des ethnies des participants par genres

Nous avons représenté les différentes ethnies des participants présents lors de l'ensemble des sessions.

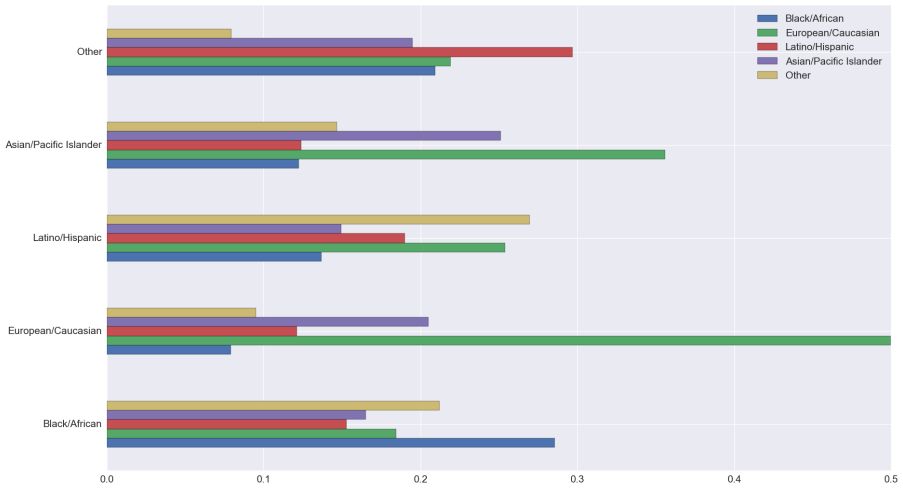


FIGURE 8 – Distribution des ethnies par matches normalisée

Ce barplot représente les ethnies des participants pour l'ensemble des rencontres où il y a eu match en fonction de l'ethnie de leur partenaire.

2.1.4 Activités des participants

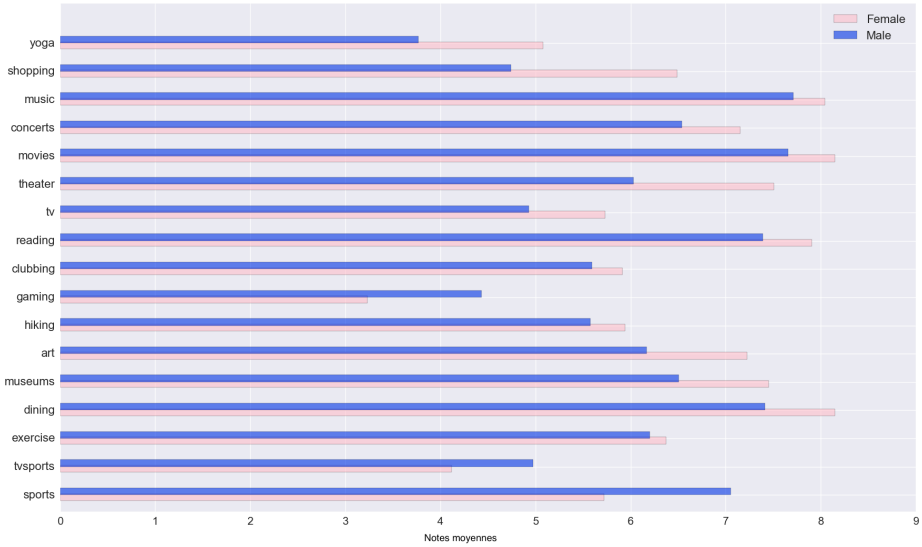


FIGURE 9 – Notes moyennes données à différentes activités par les participants

On peut remarquer que les hommes sont plus tournés vers le sport et le sport télévisé ainsi que les jeux vidéos que les femmes, tandis que ces dernières privilégient les activités culturelles, le shopping et le yoga par rapport aux hommes. Le reste des activités est noté de manière équivalente.

2.1.5 Attente des participants

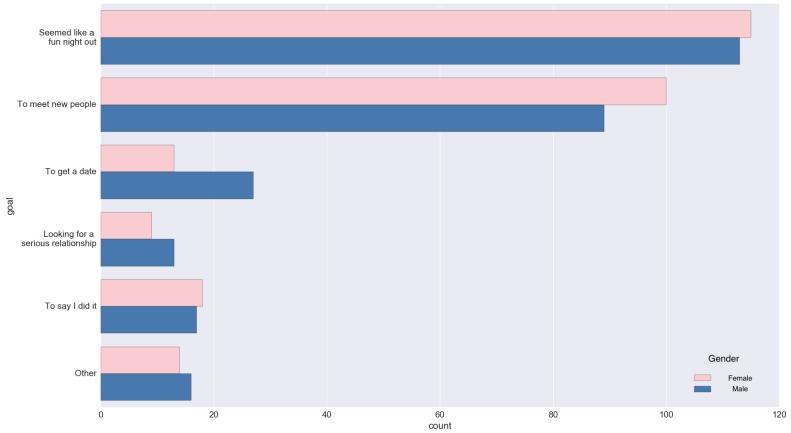


FIGURE 10 – Notes moyennes données à différentes activités par les participants

Les femmes s'attendent plus à de simples rencontres alors qu'elles ont effectivement un meilleur taux de réussite tandis que les hommes attendent plus que de simples rencontres et ont eux un moins bon taux de réussite.

2.1.6 Salaire des participants

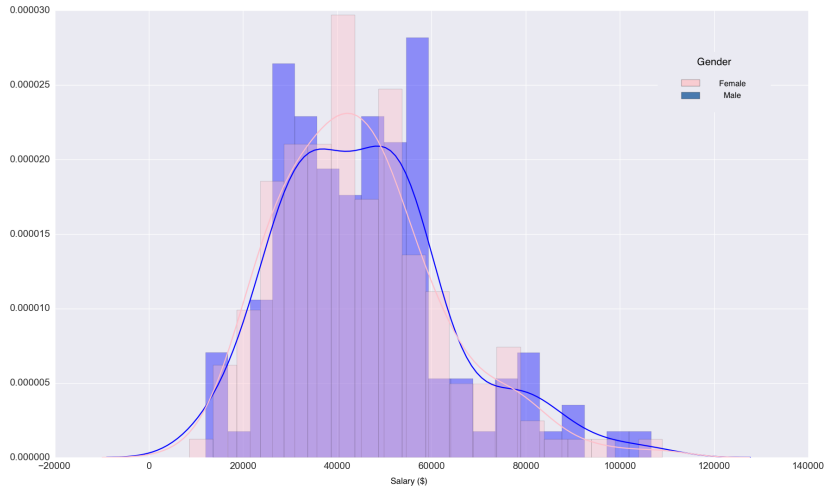


FIGURE 11 – Histogramme des participants en fonction de leur salaire et de leur genre

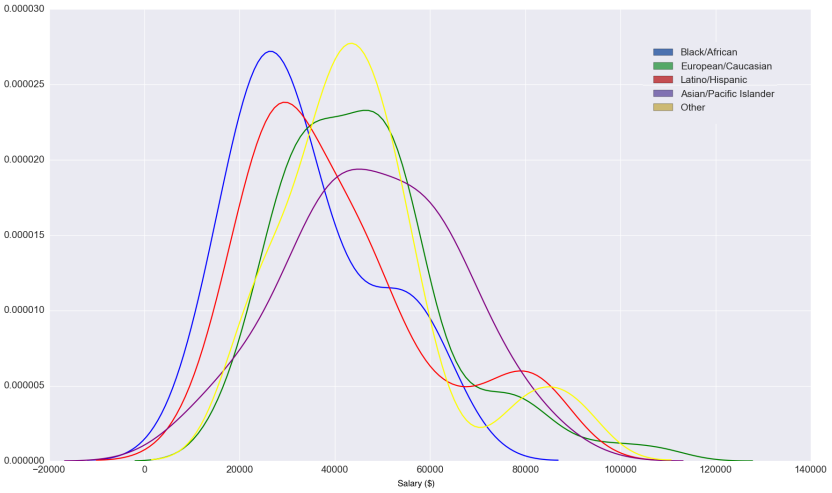


FIGURE 12 – Histogramme des participants en fonction de leur salaire et de leur ethnie

2.1.7 Bilan

Nous avons ainsi présenté les principales variables présentes dans le jeu de données que nous allons tenter d'analyser plus en profondeur à l'aide d'algorithmes de machine learning et de data mining.

3 Fouille de motifs

On va chercher dans cette partie à mettre en évidence des règles d'association. Il s'agira de définir dans un premier temps le sous jeux de données définit, c'est-à-dire les données conservées par rapport au dataset initial ainsi que les transformations apportées. Dans un deuxième temps, on décrira la méthodologie utilisée du point de vue théorique. Enfin, on explicitera la démarche adoptée sur notre jeu avec le logiciel R.

3.1 Les données

Pour la fouille de motif on ne conserve que certains attributs. L'objectif initial était de permettre la visualisation des règles d'association concernant le fait qu'il y ait eu match ou non entre les deux personnes. Nous avons donc structurés nos données en différents éléments.

La première variable est celle sur laquelle nous allons centrer notre analyse, il s'agit de la finalité du rendez-vous :

— Match : Match ou non Match

On l'associe dans un premier temps à des variables exogènes inhérentes au candidat :

— Sexe : Homme ou Femme
— Age : Jeune ou Agé

Ensuite on intègre des variables concernant la personnalité du candidat. Ces variables sont définies par la personne qu'il a rencontrée avec une note de 1 à 10. On discrétise chacune de ces variables en deux classes :

— Beauté : Beau ou Laid
— Sincérité : Sincère ou Menteur
— Intelligence : Intelligent ou Bête
— Caractère : Introverti ou Extraverti
— Ambition : Entreprenant ou Passif
— Sorties : Fêtard ou Casanier
— Communautarisé : Communautaire ou non communautaire

Alors, on intègre certains types de hobbies pour le candidat. Encore une fois chacune des variables est discrétisée en deux classes : Sports, TV, Musique, Musée, Randonnées, Films, Shopping, Restaurant , Jeux-vidéos, Lecture, Concerts, Yoga, Théâtre.

Enfin, on ajoute une variable concernant le fait que les deux candidats aient des points d'intérêts communs :

— Intérêts : Points commun ou absence de points commun

3.2 La méthode

On va chercher à expliciter ici la méthodologie mise en place pour procéder à la fouille de motif.

3.2.1 Règles d'association

Définissons dans un premier temps le concept de **règle d'association** :

— Soit $\mathrm{I} = \{i_1, i_2, ..., i_m\}$ un ensemble d'items.

— Soit $\mathrm{T} = \{t_1, t_2, ..., t_n\}$un ensemble de transactions, telles que t_i soit un sous-ensemble de I.

On définit la règle comme :

$$X \to Y \text{ , où } \mathrm{X} \in \mathrm{T},\ Y \in \mathrm{T} \text{ et } \mathrm{X} \cap Y = \emptyset$$

3.2.2 Indice de support et de confiance

La force d'une règle d'association est mesurée par son indice de support et son indice de confiance.

Le **recouvrement** de X dans T est noté $\mathrm{K_T(X)}$, il est définie par :

$$\{k = 1, 2, ..n | \mathrm{X} \subseteq t_k\}$$

Le **support** est alors la proportion des transactions de T contenant X, soit :

$$Supp(\mathrm{X}) = \frac{|\mathrm{K_T(X)}|}{Card(\mathrm{T})}.$$

L'**indice de support** d'une règle $\mathrm{X} \to Y$ est défini par la proportion de transactions de T qui contiennent $\mathrm{X} \cap Y$, soit $Supp(\mathrm{X} \cap Y)$. Il s'agit donc d'une estimation de la probabilité $\mathbb{P}(\mathrm{X} \cap Y)$.

3.2.3 Indice lift

Le **lift** d'une règle $\mathrm{X} \to Y$ mesure l'amélioration apportée par la règle d'association par rapport à un jeu de transactions aléatoire. Il est défini par :

$$\frac{Supp(\mathrm{X} \cap Y)}{Supp(\mathrm{X}).Supp(Y)}$$

Un lift supérieur à 1 traduit une corrélation positive de X et Y, et donc le caractère significatif de l'association.

3.2.4 Algorithme Apriori

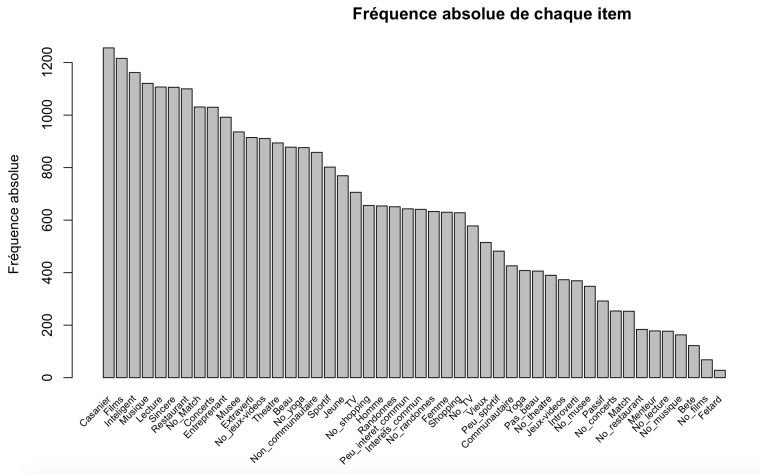
On utilisera l'algorithme Apriori pour réaliser la fouille de motif. Cet algorithme est divisé en deux grandes parties. La première concerne le comptage du support pour chaque item afin de déterminer si il est fréquent, il est déterminé par la fonction output_frequent(). La deuxième partie traite de la façons dont on explore le graphe combinatoire, il s'agit ici de la fonction children().

Input: $\mathcal{A}, \mathcal{D}$ as an array of subsets of $\mathcal{A}, \mu \in \mathbb{N}$
Output: $\{X \subseteq \mathcal{A} \mid f(X, \mathcal{D}) \geq \mu\}$
$\mathcal{P} \leftarrow \{\{a\} \mid a \in \mathcal{A}\}$
while $\mathcal{P} \neq \emptyset$ **do**
  $\mathcal{P} \leftarrow$ **output_frequent**$(\mathcal{P}, \mathcal{D}, \mu)$
  $\mathcal{P} \leftarrow$ **children**$(\mathcal{P})$
end while

3.3 Code et résultats

On utilisera dans notre étude le package "arules" sur R. Dans un premier temps on importe nos données sous forme de "tickets" ou "transactions". On réalise ensuite un histogramme des fréquence absolues de chaque item.

```
1 library(arules)
2 library(arulesViz)
3
4 #Importer les données sous forme de transaction et visualiser
  ↪  les fréquences absolues de chaque item#
5
6 table=read.transactions("/Users/hola/Desktop/TEST.csv", sep =
  ↪  ";", format="basket", rm.duplicates=TRUE)
7 itemFrequencyPlot(table, topN=60, type= "absolute", cex=0.7,
  ↪  ylab="Fréquence absolue", main="Fréquence absolue de chaque
  ↪  item")
```



Les règles d'association sont ensuite déterminées via la fonction apriori(). On explicite trois paramètres dans la fonction : un indice de support minimum de 0.4, un indice de confiance minimum de 0.8 et une longueur minimum de 4 pour les paterns. Cette selection est réalisée dans le but de limiter l'explosion combinatoire du nombre de règles et d'augmenter la significativité des règles.

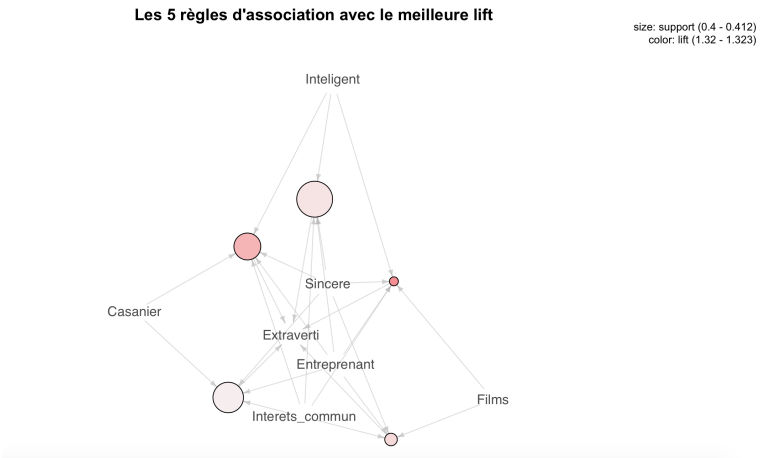
```
rules <- apriori(table,parameter = list(minlen=4, supp=0.4,
 ↪  conf=0.8))

Apriori

Parameter specification:
confidence minval smax arem  aval originalSupport maxtime
 ↪  support minlen maxlen target   ext
0.8    0.1    1 none FALSE            TRUE       5     0.4
 ↪  4     10  rules FALSE

Algorithmic control:
filter tree heap memopt load sort verbose
0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 513

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[48 item(s), 1284 transaction(s)] done
 ↪  [0.00s].
sorting and recoding items ... [30 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 7 8 9 done [0.04s].
```

```
writing ... [20370 rule(s)] done [1.06s].
creating S4 object  ... done [0.01s].
```

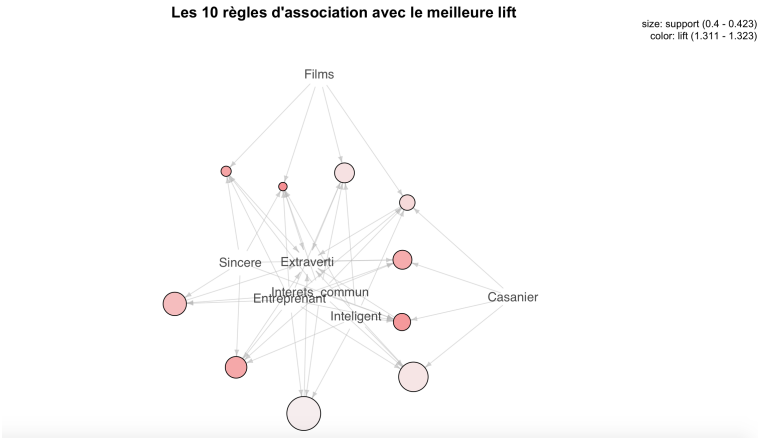
On obtient donc 20370 règles d'associations. Il s'agit par la suite de représenter les 5 puis les dix meilleures règles d'associations au vu de leur lift.

```
inspect(sort(rules, by="lift")[1:10])

lhs
 ↪  rhs          support   confidence lift
[1]  {Entreprenant,Films,Inteligent,Interets_commun,Sincere}
 ↪  => {Extraverti} 0.4003115 0.9431193  1.323459
[2]  {Casanier,Entreprenant,Inteligent,Interets_commun,Sincere}
 ↪  => {Extraverti} 0.4080997 0.9424460  1.322514
[3]  {Entreprenant,Films,Interets_commun,Sincere}
 ↪  => {Extraverti} 0.4018692 0.9416058  1.321335
[4]  {Entreprenant,Inteligent,Interets_commun,Sincere}
 ↪  => {Extraverti} 0.4119938 0.9412811  1.320880
[5]  {Casanier,Entreprenant,Interets_commun,Sincere}
 ↪  => {Extraverti} 0.4096573 0.9409660  1.320438
[6]  {Entreprenant,Interets_commun,Sincere}
 ↪  => {Extraverti} 0.4135514 0.9398230  1.318834
[7]  {Casanier,Entreprenant,Films,Inteligent,Interets_commun}
 ↪  => {Extraverti} 0.4065421 0.9371634  1.315101
[8]  {Entreprenant,Films,Inteligent,Interets_commun}
 ↪  => {Extraverti} 0.4104361 0.9360568  1.313549
[9]  {Casanier,Entreprenant,Inteligent,Interets_commun}
 ↪  => {Extraverti} 0.4190031 0.9356522  1.312981
[10] {Entreprenant,Inteligent,Interets_commun}
 ↪  => {Extraverti} 0.4228972 0.9345955  1.311498

plot(sort(rules, by="lift")[1:5], method="graph",
 ↪  control=list(type="items"),main="Les 5 règles d'association
 ↪  avec le meilleure lift")

plot(sort(rules, by="lift")[1:10], method="graph",
 ↪  control=list(type="items"),main="Les 10 règles d'association
 ↪  avec le meilleure lift")
```

On utilise la fonction plot avec l'option "graph" pour obtenir une visualisation plus lisible de nos règles.



On voit ici principalement des corrélations entre le trait de caractère "extraverti" et l'intelligence, la sincérité, le fait d'être entreprenant, de regarder des films, d'être casanier et d'avoir des intérêts en commun avec la personne rencontrée.

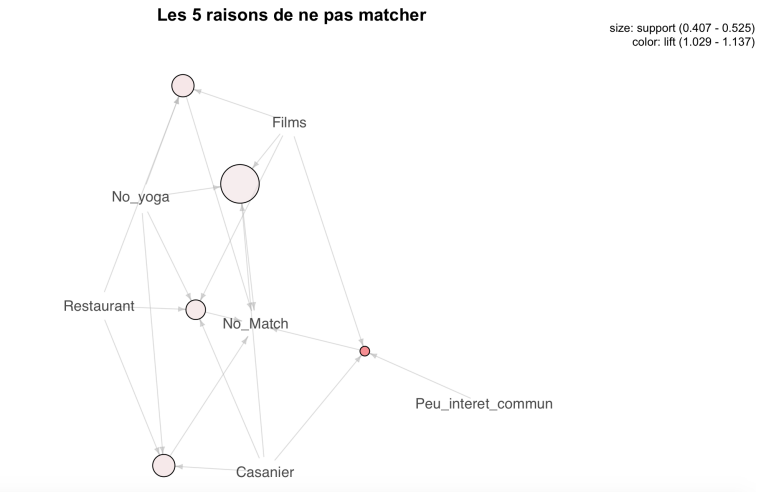


En représentant les dix meilleures règles on se rend compte que les éléments restent similaires ce qui confirme la significativité de ces éléments.

Il est aussi possible de réaliser une sélection des règles en fonction d'un élément pour voir quels attributs l'influence de manière significative.

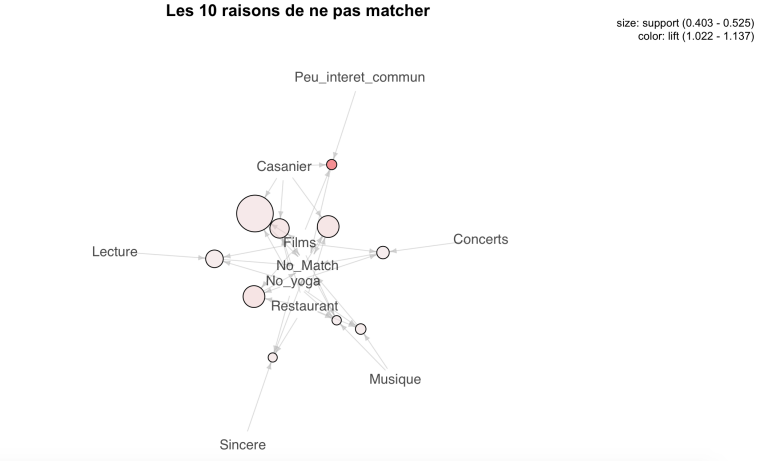
```
Celib=rules[inspect(rules)\$rhs=='{No_Match}']

plot(sort(Celib, by="lift")[1:5], method="graph",
 ↪  control=list(type="items"), main="Les 5 raisons de ne pas
 ↪  matcher")
```

On choisit ici de sélectionner les éléments influençant le plus le fait de ne pas "matcher" durant le rendez-vous.



On voit qu'il y a une corrélation entre le fait de ne pas "matcher" et le fait de regarder des films, de ne pas faire du yoga, d'être casanier, d'avoir peu d'intérêts communs avec la personne et de sortir au restaurant. Certains éléments paraissent cohérents comme le fait d'avoir peu d'intérêts communs avec la personne, d'autre sont moins évident comme le fait d'aimer aller au restaurant.

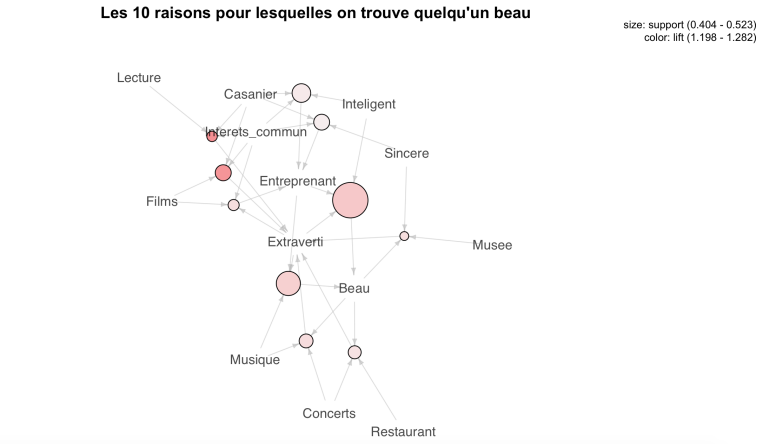
On élargit ensuite le nombre de règles considérées :



Les nouveau éléments mis en avant ne sont pas d'un abord évident. Le fait d'aimer la musique par exemple ne correspond pas forcément à quelque chose de

repoussant à priori.

On décide ensuite de travailler sur la perception de la beauté par un individu en fonction de la personnalité ou des hobbies d'une personne.



On retrouve un grand nombre d'éléments qui paraissent cohérents. On peut voir par exemple que le fait d'avoir des intérêts communs avec la personne nous fait la considérer "belle". Aussi le fait d'être extraverti va susciter l'attrait de la personne en face. Enfin le fait d'être entreprenant et intelligent incite aussi la personne à nous considérer comme "beau".

4 Clustering

Dans cette partie nous allons essayer de mettre en évidence différents clusters des rencontres. Parmis toutes les méthodes de clustering qui existent, nous allons utiliser la méthode k-means et la méthode hiérarchique. Le jeu de données sera un peu modifié ce ne sera pas le même que le jeu initial. En effet certaines colonnes peu informatives ont été supprimés ainsi que toute les lignes comprenant des NA.

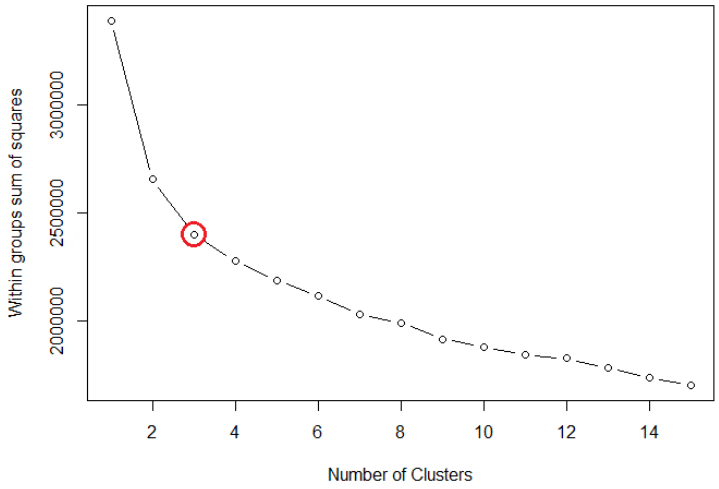
4.1 Kmeans

Tout d'abord, avant de commencer, il serait intéressant de savoir le nombre de clusters optimal qu'il faudrait pour la methode k-means. Pour cela nous allons tracer un graphe de la somme des carrés des colonnes par nombre de clusters extraits, et pour savoir le nombre optimal de cluster, on choisit le point qui se trouve sur le coude du graphe. Cette méthode est similaire à un scree test utilisé en analyse factorielle.

Listing 1 – Nombre optimal de clusters

```
wss <- (nrow(SDD)-1)*sum(apply(SDD,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(SDD,centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Nombre de clusters",ylab="Somme des carr s des groupes")
```

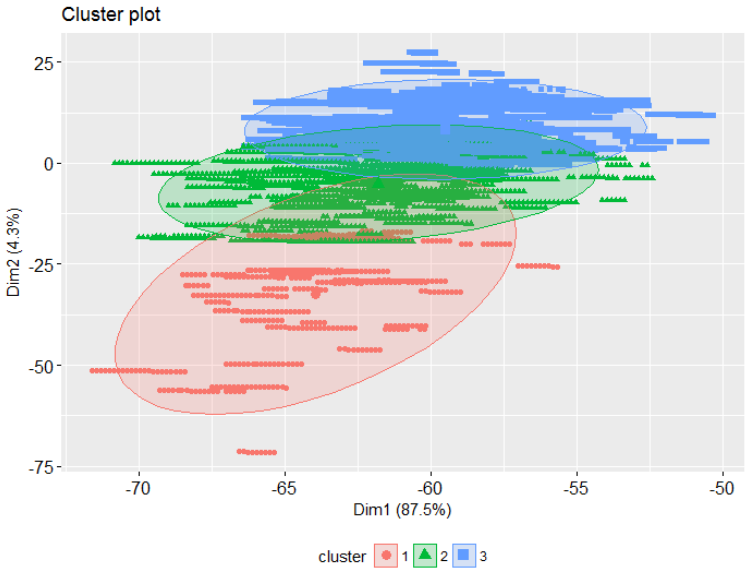
Nous faisons le test parmi 15 clusters car au delà il est clair que ce ne sera pas très intéressant.



Nous avons entourés en rouge le coude sur le graphe, qui correspond donc à 3 clusters. Nous allons donc faire un clustering avec la méthode k-means pour tenter d'extraire 3 clusters et de les visualiser.

Listing 2 – Nombre optimal de clusters

```
fit <- kmeans(SDD, 3) #kmeans    3 clusters
fviz_cluster(fit, data = SDD, geom = "point",stand = FALSE,frame.type = "norm") #trac  des clusters
fit$size #taille des clusters
```



On voit bien les 3 clusters apparaître. La commande fviz cluster réalise en fait une acp (analyse en composantes principales) et projette les points sur les axes de l'acp, d'où le fait que les clusters se croisent. Voici la taille des clusters :

cluster 1 : 557

cluster 2 : 2096

cluster 3 : 3694

Nous allons maintenant essayer de voir les 3 caractéristiques dominantes de chaque cluster. Petit rappel, nous avons fait un clustering sur les rencontres :

Listing 3 – Caractéristiques dominantes

```
1 #Remplacer NUMERO_DE_CLUSTER par 1, 2 ou 3
2 sort(table(SDD$match[SDD[40]==NUMERO_DE_CLUSTER]),↩
      decreasing=TRUE)[1:3]
3 sort(table(SDD$samerace[SDD[40]==NUMERO_DE_CLUSTER]),↩
      decreasing=TRUE)[1:3]
4 sort(table(SDD$age[SDD[40]==NUMERO_DE_CLUSTER]),decreasing=↩
      TRUE)[1:3]
5 sort(table(SDD$yoga[SDD[40]==NUMERO_DE_CLUSTER]),decreasing↩
      =TRUE)[1:3]
6 sort(table(SDD$shopping[SDD[40]==NUMERO_DE_CLUSTER]),↩
      decreasing=TRUE)[1:3]
7 sort(table(SDD$music[SDD[40]==NUMERO_DE_CLUSTER]),↩
      decreasing=TRUE)[1:3]
8 sort(table(SDD$concerts[SDD[40]==NUMERO_DE_CLUSTER]),↩
      decreasing=TRUE)[1:3]
```

```
sort(table(SDD$movies[SDD[40]==NUMERO_DE_CLUSTER]),decreasing=TRUE)[1:3]
sort(table(SDD$theater[SDD[40]==NUMERO_DE_CLUSTER]),decreasing=TRUE)[1:3]
sort(table(SDD$tv[SDD[40]==NUMERO_DE_CLUSTER]),decreasing=TRUE)[1:3]
sort(table(SDD$reading[SDD[40]==NUMERO_DE_CLUSTER]),decreasing=TRUE)[1:3]
sort(table(SDD$clubbing[SDD[40]==NUMERO_DE_CLUSTER]),decreasing=TRUE)[1:3]
sort(table(SDD$gaming[SDD[40]==NUMERO_DE_CLUSTER]),decreasing=TRUE)[1:3]
sort(table(SDD$art[SDD[40]==NUMERO_DE_CLUSTER]),decreasing=TRUE)[1:3]
sort(table(SDD$hiking[SDD[40]==NUMERO_DE_CLUSTER]),decreasing=TRUE)[1:3]
sort(table(SDD$museums[SDD[40]==NUMERO_DE_CLUSTER]),decreasing=TRUE)[1:3]
sort(table(SDD$dining[SDD[40]==NUMERO_DE_CLUSTER]),decreasing=TRUE)[1:3]
sort(table(SDD$exercise[SDD[40]==NUMERO_DE_CLUSTER]),decreasing=TRUE)[1:3]
sort(table(SDD$tvsports[SDD[40]==NUMERO_DE_CLUSTER]),decreasing=TRUE)[1:3]
sort(table(SDD$sports[SDD[40]==NUMERO_DE_CLUSTER]),decreasing=TRUE)[1:3]
```

Pour chaque discipline (sports, tvsports, exercise, dining, museums, arthiking, music, gaming, clubbing, reading, tv, theater, movies, concerts, shopping, yoga), nous avons pris les 3 où ont peu trouvé la note la plus grande sur 10 le plus souvent. La note représente l'intérêt porté par la personne lors de la rencontre.

cluster 1 :
non-match : 453 match : 104 (18,67%)
non-même race : 323 même race : 234
âge : 27 29 et 32 ans
sport : 167 personnes ont mis 10
dîner dehors : 144 personnes ont mis 10
musculation : 160 personnes ont mis 10

cluster 2 :
non-match : 1749 match : 347 (16,55%)
non-même race : 1237 même race : 859
âge : 27 23 et 28 ans
sport : 359 personnes ont mis 9
lecture : 607 personnes ont mis 9
art : 418 personnes ont mis 8

cluster 3 :
non-match : 3085 match : 609 (16,49%)
non-même race : 2266 même race : 1428
âge : 27 24 et 23 ans
lecture : 877 personnes ont mis 9
films : 829 personnes ont mis 9
musique : 895 personnes ont mis 10

Intéressons nous maintenant un peu plus au cluster 1. Pourquoi en particulier ce cluster ? Car c'est la qu'on a le plus grand pourcentage de match. En effet 18,67% des rencontres de ce cluster ont matchés. Voyons si les personnes qui ont matchés avaient la même race ou non :

Listing 4 – Match cluster 1

```
a=which(SDD[3][SDD[40]==1]==1)
SDD[5][a,]
>[1] 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 0 ↩
    0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 0 0 1 0 1 0 1 1 0 0 1 1↩
     1 1 0 0 1 1 1 1 1 0 1 1 0
 [69] 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 0 0 1 0 0 0 0↩
      0 1 1 1 1 1 0 0 0

 length(SDD[5][a,][SDD[5][a,]==1])
 >44
```

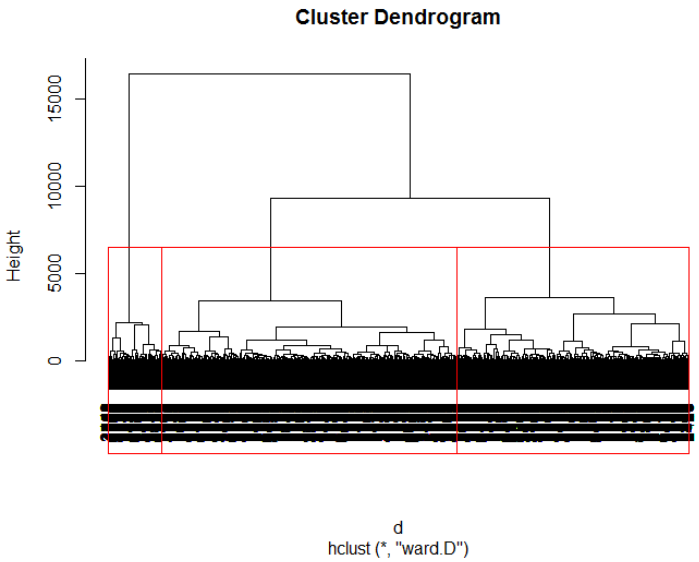
Ce qu'on apprends, c'est que sur les 104 match du premier cluster, les deux personnes étaient de la même race dans seulement 44 rencontres. Donc il apparaît que la race ici ne joue pas un rôle très important.

4.2 Hiérarchique

On peut faire exactement le même travail avec un clustering hiérarchique, et demander d'obtenir 3 clusters.

Listing 5 – Clustering hiérarchique

```
d <- dist(SDD, method = "euclidean")
fit <- hclust(d, method="ward")
plot(fit)
groups <- cutree(fit, k=3)
rect.hclust(fit, k=3, border="red")
```



Évidemment, la racine du dendogramme est totalement illisible vue le nombre de données que nous avons. Mais ce qu'on voit c'est que les tailles des clusters ont l'air semblable à celles des clusters du kmeans.

Voici la taille des clusters :

cluster 1 : 2537

cluster 2 : 3226

cluster 3 : 584

Nous allons essayer de faire la même descriptions des clusters que pour le kmeans.

cluster 1 :

non-match : 2099 match : 438 (17,26%)
non-même race : 1536 même race : 1001
âge : 24 23 et 25 ans
lectures : 653 personnes ont mis 9
musique : 520 personnes ont mis 9
films : 633 personnes ont mis 8

cluster 2 :

non-match : 2701 match : 525 (16,27%)
non-même race : 1976 même race : 1250
âge : 27 26 et 23 ans
lecture : 785 personnes ont mis 9
films : 857 personnes ont mis 9
musique : 779 personnes ont mis 10

cluster 3 :
non-match : 487 match : 97 (16,61%)
non-même race : 314 même race : 270
âge : 27 29 et 25 ans
sport : 163 personnes ont mis 10
dîner dehors : 144 personnes ont mis 10
musique : 131 personnes ont mis 10

Encore une fois, c'est dans le premier cluster que le ratio de match est le plus élevé. Voyons de la même façons si les personnes qui ont matché avaient la même race ou non. Sur les 438 match, seulement 166 avaient la même race.

En conclusion, les deux clusters donnent à peu près les mêmes résultats. La différence notable est que les caractéristiques dominantes ne sont pas les mêmes dans les clusters même si ce sont souvent les mêmes qui reviennent. On aurait pu fouiller un peu plus dans les cluster à savoir regarder les caractéristiques de chaque personnes de chaque rencontre. Nous pouvions également utiliser encore une ou deux autres méthodes de clustering pour avoir un plus grand panel de comparaison. Nous allons maintenant essayer de faire une approche un peu différente, à savoir plus statistique en tentant de faire une classification.

5 Classification

On se propose ici de prédire la variable binaire match en fonction des attributs suivants :

- sports
- tvsports
- exercise
- dining
- museums
- art
- hiking
- music
- gaming
- clubbing
- reading
- tv
- theater
- movies
- concerts
- shopping
- yoga

Nous mettrons en oeuvre dans cette partie l'algorithme Random Forest, GBM.

5.1 Random Forest

Notre base d'apprentissage se compose de 80% des données matchs, soit 5078, et notre échantillon test de 20%, c'est à dire de 1269 date.

Listing 6 – code R – Classification Random Forest.

```
1  library(ibr)
2  library(caret)
3  library(kernlab)
4  library(doParallel)
5  library(TTR)
6  library(foreach)
7  library(doSNOW)
8  library(randomForest)
9  library(pROC)
10
11 file="SDD.csv"
12 data=read.csv(file,header=T,sep=";")
13 data$match=as.factor(data$match)
14 data$id=NULL
15 data$match=mapvalues(data$match, from = c("0", "1"), to = c↩
       ("NO", "YES")) #Caret n'aime pas les 0 et 1 comme donne↩
```

```
     dans le csv, je change
16
17 trainIndex <- createDataPartition(data$match, p = .8,list =↩
     FALSE,times = 1)
18
19 Challenge_Train <- data[ trainIndex,]
20 Challenge_Test  <- data[-trainIndex,]
21
22 cl<-makeSOCKcluster(4) #change the 2 to your number of CPU ↩
    cores
23
24 registerDoParallel(cl)
25
26 fitControl <- trainControl(method = "repeatedcv",
27                            number = 10,
28                            repeats = 10,
29                            classProbs = TRUE,
30                            summaryFunction = ↩
                                twoClassSummary)
31
32 set.seed(825)
33
34 rfFit1 <- train(match ~ sports + tvsports+ exercise +  ↩
    dining + museums +art
35                + hiking+ music + gaming + clubbing + ↩
                      reading + tv +
36                  theater + movies + concerts + shopping + ↩
                       yoga, data = Challenge_Train,
37                method = "rf",
38                trControl = fitControl,
39                metric= "ROC")
40
41 stopCluster(cl)
42
43
44 Pred_rf =predict(rfFit1,newdata=Challenge_Test)
45 confusionMatrix(Pred_rf, Challenge_Test$match)
46
47 Pred_rf2 =predict(rfFit1,newdata=Challenge_Test,type="prob"↩
    ) # ici on cree  un autre objet de prediction mais qui ↩
    contient les noms des classes
48
49 rforestROC <- roc(Pred_rf2$YES,response = Challenge_Test$↩
    match,levels = rev(levels(Challenge_Test$match)))
50
51 plot(rforestROC, type = "S", print.thres = .5)
```

En sortie, nous avons les résultats suivants :

```
Confusion Matrix and Statistics

          Reference
Prediction   NO  YES
       NO  1029  203
       YES   28    9

               Accuracy : 0.818
                 95% CI : (0.7956, 0.8388)
    No Information Rate : 0.8329
    P-Value [Acc > NIR] : 0.9276

                  Kappa : 0.0238
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.97351
            Specificity : 0.04245
         Pos Pred Value : 0.83523
         Neg Pred Value : 0.24324
             Prevalence : 0.83294
         Detection Rate : 0.81087
   Detection Prevalence : 0.97084
      Balanced Accuracy : 0.50798

       'Positive' Class : NO
```

FIGURE 13 – Matrice de confusion et statistiques

Nous obtenons une accuracy d'environ 82%. Nous constatons que les *No* sont relativement bien classés. Sur les 1057 No de notre échantillons test, nous en prédisons 1029. Notre modèle fonctionne bien, de plus c'est assez logique, en effet, si trouver l'âme soeur était aussi facile, cela se saurait. Ceci explique en parti le fort taux de No. L'algorithme Random Forest s'avère être un classifieur relativement correct. De plus, le kappa se trouve dans la fourchette 0.0 — 0.20, il a donc un accord faible. A noté aussi qu'ici dans notre cas qui concerne des données speed-dating, si nous avions eu un fort taux de faux négatifs cela n'aurait pas été grave, cependant en avoir un grand nombre dans un contexte où l'on chercherait à prédire si une personne ou non à le cancer est dangereux. En effet, cela voudrait dire que l'on laisse s'échapper des personnes malades. Ceci est pour signaler l'importance que peut avoir la sensibilité (rappel) et la spécificité (précision) selon le contexte, les données, la nature de ce que l'on cherche à prédire.

5.2 Gradient Boosting

De manière analogue à la précédente, nous allons nous servir gradient boosting, conny aussi sous le nom de gradient boosting machine.

Listing 7 – code R – GRADIENT BOOSTING MACHINE.

```
1
2  file="SDD.csv"
3  data=read.csv(file,header=T,sep=";")
4  data$match=as.factor(data$match)
5  data$id=NULL
```

```
6  data$match=mapvalues(data$match, from = c("0", "1"), to = c("NO", "YES"))
7
8
9  trainIndex <- createDataPartition(data$match, p = .8,list = FALSE,times = 1)
10
11
12 Challenge_Train <- data[ trainIndex,]
13 Challenge_Test  <- data[-trainIndex,]
14
15 cl<-makeSOCKcluster(4) #change the 2 to your number of CPU cores
16
17 registerDoParallel(cl)
18
19 fitControl <- trainControl(method = "repeatedcv",
20                            number = 10,
21                            repeats = 10,
22                            classProbs = TRUE,
23                            summaryFunction = twoClassSummary)
24
25 set.seed(825)
26
27
28
29 gbmFit1 <- train(match ~ sports + tvsports+ exercise +  dining + museums +art
30                  + hiking+ music + gaming + clubbing + reading + tv +
31                    theater + movies + concerts + shopping + yoga, data = Challenge_Train,
32                  method = "gbm",
33                  trControl = fitControl,
34                  metric= "ROC")
35
36
37 stopCluster(cl)
38
39
40 Pred_gbm =predict(gbmFit1,newdata=Challenge_Test)
41 conf_gbm=confusionMatrix(Pred_gbm, Challenge_Test$match)
42
43 gbm.probs <- predict(gbmFit1,Challenge_Test,type="prob")
44
45 gbm.ROC <- roc(predictor=gbm.probs$YES,
46                response=Challenge_Test$match,
47                levels=rev(levels(Challenge_Test$match)))
48
```

```
49
50 plot(gbm.ROC,main="GBM ROC",type = "S", print.thres = .5)
```

En sortie, nous avons les résultats suivants :

```
Confusion Matrix and Statistics

          Reference
Prediction   NO  YES
       NO  1052  210
       YES    5    2

               Accuracy : 0.8306
                 95% CI : (0.8088, 0.8508)
    No Information Rate : 0.8329
    P-Value [Acc > NIR] : 0.6069

                  Kappa : 0.0077
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.995270
            Specificity : 0.009434
         Pos Pred Value : 0.833597
         Neg Pred Value : 0.285714
             Prevalence : 0.832939
         Detection Rate : 0.828999
   Detection Prevalence : 0.994484
      Balanced Accuracy : 0.502352

       'Positive' Class : NO
```

FIGURE 14 – Matrice de confusion et statistiques

On peut y lire que nous avons seulement 5 faux négatifs, le gradient boosting est plus performant que le random forest 83%. Comparons les deux algorithmes.

5.3 Comparaison

Le package caret peut être utilisé pour comparer nos deux modèles en utilisant les échantillons bootstrap qui ont été créés dans le processus de construction des deux modèles.

Listing 8 – code R – GBM VS RANDOM FOREST.

```
1
2 rValues <- resamples(list(rf=rfFit1 ,gbm=gbmFit1))
3 rValues$values
4 summary(rValues)
5 bwplot(rValues,metric="ROC",main="GBM vs Random Forest")↩
        # boxplot
```

On a :

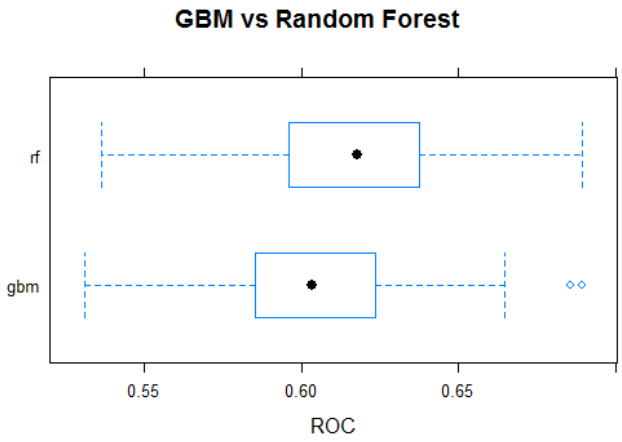


FIGURE 15 – GBM vs Random forest

6 Conclusion

Les gens ont tendance à aimer les membres du sexe opposé qu'ils perçoivent comme attrayants, mais l'un des facteurs qui contribue à l'évaluation de l'attractivité est la familiarité. Il existe plusieurs façons d'améliorer ce projet afin de générer des mesures plus précises et des préférences des partenaires de rencontres. Des extensions et la mise en œuvre de l'utilisation de l'analyse statistique théorique, ainsi que des techniques d'apprentissage automatique plus poussées, serait également utile pour une meilleure compréhension des marchés de rencontres. Rajoutons aussi la limite des données, en ce sens qu'elle ont été recueillies à partir d'une petite région (groupe spécifique de personnes), qui ne pouvait pas expliquer la généralité de la population globale.

7 Annexe

*

7.1 Code Python Analyse Descriptive

```
#Import libraries:
import pandas as pd
import numpy as np
import csv
import os
import re
import pandas as pd
import matplotlib as plt
import seaborn as sns
import pylab as plb
import matplotlib.figure as fig
import matplotlib.pyplot as plt
import matplotlib.axes as ax
import scipy as sp
from scipy.optimize import curve_fit

os.chdir("/Users/Dupi/Documents/Master_Data_Science/Data Mining/Projet/speed-dating-experiment/")

def remove_quotes(text):
  import re
  matches=re.findall(r'\"(.+?)\"',text)
  return ",".join(matches)


data = "Speed Dating Data.csv"
data_frame = pd.read_csv(data)
#data_frame = data_frame.applymap(lambda x:  x.replace(",","|") if isinstance(x,str) else x)

os.remove('sdd.csv')
data_frame.to_csv('sdd.csv')

'''
-------------------------------------------------------
Descriptive analysis
-------------------------------------------------------
'''

data = "Speed Dating Data.csv"
data_frame = pd.read_csv(data)

```

```
pd.set_option('display.max_columns', 1000)


data_frame.head()
data_frame.shape

'''
-----------------------------------------------------
Gender Age comparison
'''

unique_iid_frame = data_frame.groupby("iid").first()

gender_age = unique_iid_frame[['gender', 'age', 'wave']]
gender_age = gender_age.loc[~(np.isnan(gender_age.age))]
gender_age.loc[gender_age.gender == 1, 'gender'] = 'Male'
gender_age.loc[gender_age.gender == 0, 'gender'] = 'Female'

len(np.unique(data_frame.iid))
sns.set(rc={"figure.figsize": (8, 6)})
sns.set(font_scale=1.5)  # crazy big

colors = ["light rose", "windows blue"]
# Age | Gender global
from scipy.stats import norm
male = gender_age.loc[gender_age.gender =='Male']
female = gender_age.loc[gender_age.gender =='Female']
sns.distplot(male.age.astype(int), color='blue', hist=True,
     label='Male')      #, bins=len(np.unique(male.age))
sns.distplot(female.age.astype(int), color='pink', hist=
    True, label='Female')          #, bins=len(np.unique(
    female.age))
sns.plt.show()

# Age | Gender per wave
sns.violinplot(x='wave', y="age", hue="gender", data=gender
    _age,
                split=True, inner="stick", palette=sns.xkcd_
                    palette(colors));
sns.plt.show()

print('Mean Male: {0} ---- Mean Female: {1}\t'.format(
    gender_age.loc[gender_age.gender=='Male'].age.mean(),
    gender_age.loc[gender_age.gender=='Female'].age.mean())
    )

n_race=[]
for i in range(1,7):
    n_race.append(len(unique_iid_frame[unique_iid_frame.
        race == i].race))
```

```
n_race[4] = 1
'''
-------------------------------------------------------
Matches comparison
'''

# Forked from Kaggle

fig, axes = plt.subplots(1, 2, figsize=(18,5))

# The number of dates per person
num_dates_per_male = data_frame[data_frame.gender == 1].groupby('iid').apply(len)
num_dates_per_female = data_frame[data_frame.gender == 0].groupby('iid').apply(len)
axes[0].hist(num_dates_per_male, bins=22, alpha=0.5, label='# dates per male', color='blue')
axes[0].hist(num_dates_per_female, bins=22, alpha=0.5, label='# dates per female', color='pink')
axes[0].legend(loc='upper right')

# The number of matches per person
matches = data_frame[data_frame.match == 1]
matches_male = matches[matches.gender == 1].groupby('iid').apply(len)
matches_female = matches[matches.gender == 0].groupby('iid').apply(len)
weights1 = np.ones_like((matches_male / num_dates_per_male).dropna())/len((matches_male / num_dates_per_male).dropna())
weights2 = np.ones_like((matches_female / num_dates_per_female).dropna())/len((matches_female / num_dates_per_female).dropna())
axes[1].hist((matches_male / num_dates_per_male).dropna(), alpha=0.5, label='male match percentage',  color='blue',  weights=weights1)
axes[1].hist((matches_female / num_dates_per_female).dropna(), alpha=0.5, label='female match percentage', color='pink',  weights=weights2)
axes[1].legend(loc='upper right')

Male_percentage = np.mean((matches_male / num_dates_per_male).dropna())
Female_percentage = np.mean((matches_female / num_dates_per_female).dropna())
## Fit exponential to n matches

def func(x, a, b, c):
    return a*np.exp(-b*x) + c

```

```
116 # Fake data
117 # x = np.arange(0, 70., 2.)
118 # yl = 300 + 63*np.exp(-x/35.)
119 #
120 # popt, pcov = curve_fit(func, x, yl, p0=(40, 0.012, 250), ↩
      maxfev=20000)
121 # a, b, c = popt
122 # print 'a=', a, 'b=', b, 'c=', c
123 # print 'func=', func(x, a, b, c)
124 #
125 # popt2, pcov2 = curve_fit(func, x, yl, p0=None, maxfev↩
      =20000)
126 # a2, b2, c2 = popt2
127 # print 'a2=', a2, 'b2=', b2, 'c2=', c2
128 # print 'func=', func(x, a2, b2, c2)
129 #
130 # xf = np.linspace(0, 70, 100)
131 # yf = a*np.exp(-b*x) + c
132 #
133 # plt.clf()
134 # plt.plot(x, yf, 'r-', label="Fitted Curve")
135 # plt.plot(x, func(x, *popt))
136 # plt.plot(x, func(x, *popt2), 'b-', label='Fit w/o guess')
137 #
138 # plt.plot(x, yl, 'go', label='Lacquered')
139 #
140 # plt.legend()
141 # plt.ylabel("Temperature (K)")
142 # plt.xlabel("Time (min)")
143 # plt.show()
144
145 print('Avg. dates per male: {0:.1f}\t\tAvg. dates per ↩
      female: {1:.1f}\nAvg. male match percentage: {2:.2f}\↩
      tAvg. female match percentage: {3:.2f}'.format(
146         num_dates_per_male.mean(),
147         num_dates_per_female.mean(),
148         (matches_male / num_dates_per_male).mean() * 100.0,
149         (matches_female / num_dates_per_female).mean() * ↩
              100.0))
150
151
152 """
153 -------------------------------------------------------
154 Diff Age Plot
155 """
156
157 diff_age_frame = data_frame[['age', 'age_o', 'match']].copy↩
      ()
158 diff_age_frame['diff_age'] = (data_frame.age - data_frame.↩
      age_o)
```

```
diff_age_frame.dropna(inplace=True)
diff_age_frame.drop(['age', 'age_o'], inplace=True, axis=1)
diff_age_frame = diff_age_frame[diff_age_frame['diff_age']>=0]
n1 = len(np.unique(diff_age_frame[diff_age_frame.match == 1].diff_age))
n2 = len(np.unique(diff_age_frame[diff_age_frame.match == 0].diff_age))
sns.distplot(diff_age_frame[diff_age_frame.match == 1].diff_age,  color='green', hist=True, bins=n1+1)
sns.distplot(diff_age_frame[diff_age_frame.match == 0].diff_age,  color='grey', hist=True, bins=n2+8)


"""
-----------------------------------------------------
Race Plot
"""

race_frame = data_frame[['iid','race', 'race_o', 'match', 'gender', 'age']].groupby("iid").first()
race_list = ['Black \n African American','European \n Caucasian-American','Latino \n Hispanic American',
        'Asian, Pacific Islander \n Asian-American','Native American','Other']

h = sns.countplot(y="race",data=race_frame, hue='gender', palette=sns.xkcd_palette(colors))
h.set(yticklabels=race_list)


race_map = {1:"Black/African", 2:"European/Caucasian", 3:"Latino/Hispanic", \
            4:"Asian/Pacific Islander", 5:"Native American", 6:"Other"}


race_frame = data_frame[['race', 'race_o', 'match']].copy()
race_frame.loc[race_frame.match==1]
count_race = np.zeros(shape=[6, 6])
for i in range(1, 7):
    for j in range(1, 7):
        count_race[i-1, j-1] = len(race_frame.loc[np.logical_and(np.logical_and(race_frame.race == i, race_frame.race_o == j),
     race_frame.match == 1)])/float(n_race[i-1]+n_race[j-1])   #/float(n_race[i-1]+n_race[j-1])
    count_race[i-1,:] = count_race[i-1,:]/np.sum(count_race[i-1,:])

```

```
194 count_race_frame = pd.DataFrame(count_race, columns=race_↩
        map.values(), index=race_map.values())
195 count_race_frame.drop("Native American", inplace=True)
196 count_race_frame.drop("Native American", axis=1, inplace=↩
        True)
197 count_race_frame.plot.barh()
198
199
200 """
201 -----------------------------------------------------
202 Pair Plot
203 """
204 import seaborn as sns
205 sns.set(style="ticks")
206
207 # Load the example tips dataset
208 tips = sns.load_dataset("tips")
209
210 # Draw a nested boxplot to show bills by day and sex
211 sns.boxplot(x="day", y="total_bill", hue="match", data=tips↩
        , palette="PRGn")
212 sns.despine(offset=10, trim=True)
213
214
215 sns.set(style="ticks", color_codes=True)
216 iris = sns.load_dataset("iris")
217 g = sns.pairplot(iris)
218 g = sns.pairplot(iris, hue="species")
219
220 sns.set(style="ticks", color_codes=True)
221 pp_data = data_frame[['age', 'age_o', 'race', 'race_o', '↩
        samerace', 'income', 'match']]
222 pp_data = data_frame[['age', 'age_o', 'match']]
223 pp_data.sample(n=100)
224 pp_data.match.loc[pp_data.match==1] = 'Yes'
225 pp_data.match.loc[pp_data.match==0] = 'No'
226 pp_data.dropna(inplace=True)
227 g = sns.pairplot(pp_data, hue="match")
228
229 sns.jointplot(x, y, kind="hex", stat_func=kendalltau, color↩
        ="#4CB391")
230
231 '''
232 -----------------------------------------------------
233 Field of study
234 '''
235
236 study_list = ['Law','Math','Social/Psy','MedicalScience, \n↩
        Pharmaceuticals , BioTech ',
```

```
237             'Engineering','Creative Writing, \n Journalism'↩
                    ,'History, Religion, \n Philosophy',
238             'Business,Econ,Finance','Education, Academia','↩
                    Biological Sciences, \n Chemistry, Physics'↩
                    ,
239             'Social Work','Undergrad \nundecided','↩
                    Political Science \nInternational Affairs',
240             'Film','Fine Arts, \n Arts Administration','↩
                    Languages','Architecture','Others']
241
242 h = sns.countplot(y="field_cd", data=unique_iid_frame, hue=↩
        'gender', palette=sns.xkcd_palette(colors))
243 h.set(yticklabels=study_list)
244
245 '''
246 -----------------------------------------------------
247 Career
248 '''
249 career = ['Lawyer','Academic/Research','Psychologist','↩
        Doctor/Medicine','Engineer','Creative Arts/↩
        Entertainment',
250           'Banking/Consult\n Finance/Market\n CEO/Entrepr/↩
                  Admin','Real Estate',
251           'International \n Humanitarian Affairs','↩
                  Undecided','Social Work','Speech Pathology','↩
                  Politics',
252           'Pro sports/Athletics','Other','Journalism','↩
                  Architecture']
253 h = sns.countplot(y="career_c",data=unique_iid_frame, hue='↩
        gender', palette=sns.xkcd_palette(colors))
254 h.set(yticklabels=career)
255
256
257 '''
258 -----------------------------------------------------
259 Interest
260 '''
261
262 # activities_list = ['sports','tvsports','exercise','dining↩
        ','museums','art','hiking','gaming','clubbing','reading↩
        ',
263 #                        'tv','theater','movies','concerts↩
        ','music','shopping','yoga']
264 gender_interest = unique_iid_frame.groupby(['gender']).mean↩
        ()[activities_list]
265
266 h = gender_interest.T.plot.barh()
267
268 # gender_interest = unique_iid_frame.groupby(['gender']).↩
        mean()[activities_list].values
```

```
# h = plt.barh(np.arange(0,2*gender_interest.shape[1],2)-0.2,gender_interest[0,:], height=0.5,color='pink',alpha=0.6,label='Female')
# h = plt.barh(np.arange(0,2*gender_interest.shape[1],2)+0.2,gender_interest[1,:], height=0.5,color=[0,0.2,0.9],alpha=0.6,label='Male')
# h = plt.yticks(np.arange(0,2*gender_interest.shape[1],2)+0.2,activities_list,fontsize=16)
# h = plt.ylim(-1,2*gender_interest.shape[1]+1)
# h = plt.legend(loc=0,fontsize=16)

'''
-----------------------------------------------------
Attentes
'''

att = ['Seemed like a \n fun night out','To meet new people','To get a date','Looking for a \n serious relationship',
     'To say I did it','Other']
h = sns.countplot(y="goal", data=unique_iid_frame, hue='gender', palette=sns.xkcd_palette(colors))
h.set(yticklabels=att)

'''
-----------------------------------------------------
Salaire
'''
salary_male = unique_iid_frame[unique_iid_frame.gender==1].income
salary_male.dropna(inplace=True)
salary_female = unique_iid_frame[unique_iid_frame.gender==0].income
salary_female.dropna(inplace=True)
salary_male = map(lambda x: int(x[:-3].replace(",", "")), salary_male)
salary_female = map(lambda x: int(x[:-3].replace(",", "")),salary_female)
sns.distplot(salary_male, color='blue', hist=True, bins=20)
sns.distplot(salary_female, color='pink', hist=True,  bins=20)

colors = ['blue','green','red','purple','black','yellow']
race_list = ['Black \n African American','European \n Caucasian-American','Latino \n Hispanic American',
     'Asian, Pacific Islander \n Asian-American','Native American','Other']
for i in range(1,7):
    salary_temp = unique_iid_frame[unique_iid_frame.race==i].income
```

```
303     if len(salary_temp)==0:
304         continue
305     salary_temp.dropna(inplace=True)
306     salary_temp = map(lambda x: int(x[:-3].replace(",", "")↩
            ), salary_temp)
307     sns.distplot(salary_temp, color=colors[i-1], hist=False↩
            )
308     del salary_temp
```