Fiche TD avec le logiciel R : tdr1110

Clustering ou classification avancée

D. Clot & A.B. Dufour

Cette fiche approfondit la connaissance des différentes stratégies d'agrégation utilisées en classification ascendante hiérarchique et fournit quelques critères usuels pour l'aide à l'identification du nombre de classes. Elle présente également en détail la fonction `kmeans` de R.

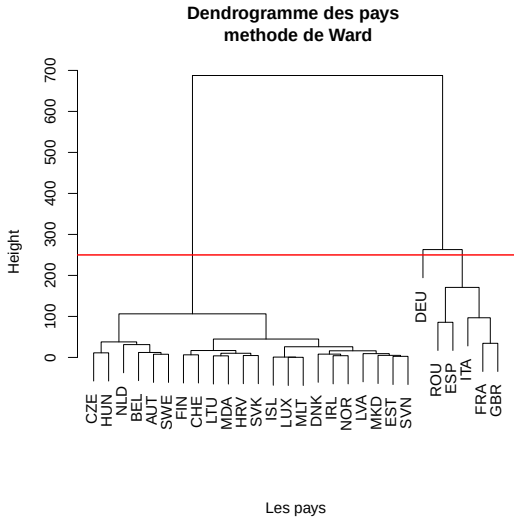
Table des matières

1 Critères de mesure de la qualité d'une partition

La fiche tdr1109 présente une introduction à la classification autour de données sur la mortalité dans 28 pays européens (données OMS, 2015). En utilisant la distance euclidienne sur ces données et en adoptant la stratégie de Ward, on obtient la représentation graphique associée : le dendrogramme. Puis dans la partie 5, dans une optique de partition, on coupe le dendrogramme à la hauteur de 250. Les groupes formés sont facilement obtenus grâce à la fonction `cutree`.

```
library(ade4)
library(adegraphics)
me <- read.table("http://pbil.univ-lyon1.fr/R/donnees/mortality_Europe.txt", h=TRUE)
mame <- me[,c(5:9,11:24)]
rownames(mame) <- me$Code
dmame <- dist(mame)
classif <- hclust(d = dmame, method = "ward.D2")
plot(classif, main="Dendrogramme des pays \n methode de Ward", xlab = "Les pays",
        sub = "")
abline(h = 250, col = "red", lwd = 1.5)
parti <- cutree(classif, h = 250)
parti
```

```
AUT BEL HRV CZE DNK EST FIN FRA DEU HUN ISL IRL ITA LVA LTU LUX MLT NLD NOR MDA ROU
  1   1   1   1   1   1   1   2   3   1   1   1   2   1   1   1   1   1   1   1   2
SVK SVN ESP SWE CHE MKD GBR
  1   1   2   1   1   1   2
```



Sans expertise sur le contexte, la question de la qualité d'un tel choix se pose ici et dans le cadre général. Comme cela est le cas pour le problème du choix de la dimension du sous-espace de projection lors d'une analyse en composantes principales, on dispose de différents critères pour aider au choix du nombre de classes. Ce paragraphe en présente quelques-uns.

1.1 Critère du coude

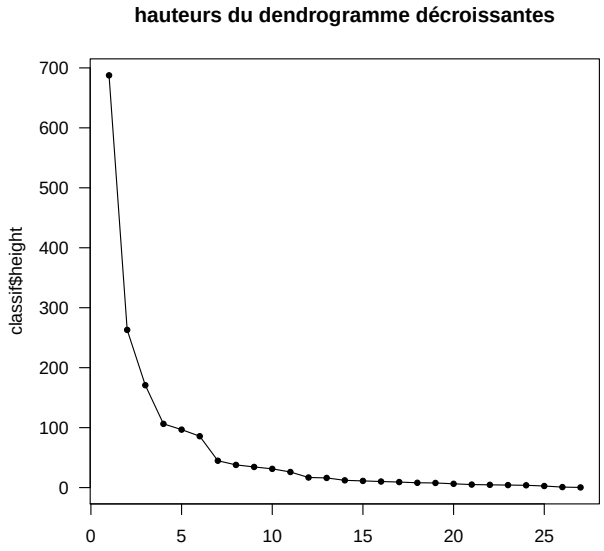
Une approche simple consiste à appliquer le critère du coude aux hauteurs découlant du dendrogramme et servant à construire l'ultramétrique. Considérant ces valeurs énumérées en ordre décroissant, on recherche un coude sur une représentation graphique de ces valeurs.

L'objet `classif` contient les hauteurs qui nous intéressent pour rechercher les différents coudes :

```
names(classif)
```

```
[1] "merge"       "height"      "order"       "labels"      "method"
[6] "call"        "dist.method"
```

```
plot(rev(classif$height), type = 'l', main = "hauteurs du dendrogramme décroissantes",
         ylab = "classif$height", xlab = "", las = 1)
points(1:length(classif$height), rev(classif$height), pch = 20)
```



Exercice.
Déterminez les différents coudes et motivez le choix de l'un d'eux. La hauteur de 250 proposée dans le tdr1109 est-elle validée par le critère ?

1.2 Observation du coefficient de corrélation multiple R^2

Il est fréquent de calculer le ratio de l'inertie interclasse sur l'inertie totale pour juger de la qualité d'une partition. Cette grandeur devrait être idéalement proche de 1 tout en correspondant à une partition moins fine que la partition discrète. Dans le cas d'une hiérarchie découlant d'un dendrogramme, il est possible de calculer le R^2 pour chaque partition et de s'aider du graphique de ces valeurs pour faire un choix équilibré entre la part d'inertie expliquée par les

classes et un petit [1] nombre de classes.

Le R^2, souvent noté RSQ, peut être estimé, pour une partition donnée P de K classes, par

$$R^2 = 1 - \frac{\sum_k \sum_{x \in C_k} \|x - g_k\|^2}{\sum_{x \in E} \|x - g_E\|^2}$$

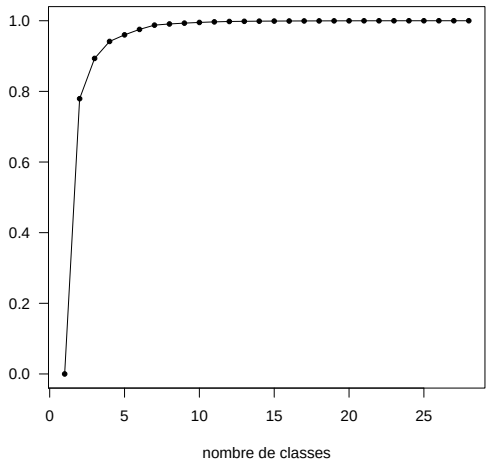
Tenant compte du fait que l'inertie totale des points de E est la somme de l'inertie intragroupe et de l'inertie intergroupe, il est clair que cette formule correspond bien au ratio attendu.

La formule précédente ne conduit pas à une implémentation très efficace pour le calcul du R^2. Mieux vaut essayer de calculer l'inertie des centres de classes pondérés. Le code ci-dessous permet de calculer les valeurs de R^2 pour les différentes partitions issues du découpage de l'objet `hexo2` :

```
RSQ <- rep(0, nrow(mame))
sum(scale(mame, scale = FALSE)^2) -> SQTot
for (i in 1:nrow(mame)) {
 Cla <- as.factor(cutree(hexo2, i))
 sum(t((t(sapply(1:ncol(mame), function(i) tapply(mame[,i], Cla, mean)))-
         apply(mame, 2, mean))^2) * as.vector(table(Cla)))/SQTot -> RSQ[i]
}
```

Ce code calcule, pour chaque partition, le centre de gravité de chaque classe, le centre de gravité global et les utilise pour calculer l'inertie interclasse, ramenée à l'inertie totale. La fonction `tapply` est utilisée pour calculer les centres de gravité des classes par variable et la fonction `sapply` permet de faire ce calcul sur l'ensemble des variables.

Reprenez ce code afin de produire le graphique des valeurs de R^2 ci-dessous.



1. petit relativement au nombre d'objets à classer

1.3 Pseudo F

Le pseudo F est un autre indicateur, une adaptation du F-ratio rencontré en analyse de la variance[2]. Il sert à mesurer la séparation des classes. Une fois le R^2 calculé, son évaluation est triviale :

$$\text{pseudo F} = \frac{\frac{R^2}{K-1}}{\frac{1-R^2}{n-K}}$$

où K est le nombre de classes de la partition considérée et n le nombre d'objets de l'ensemble E.

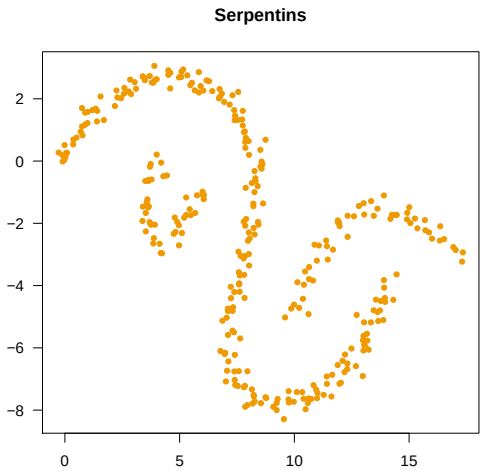
Exercice.
Calculez le pseudo F et représentez ses valeurs pour les différentes partitions précédemment considérées.

2 Adéquation et limites des stratégies d'agrégation

Toutes les stratégies d'agrégation ont des propriétés qui les distinguent. Certaines sont plus adaptées que d'autres pour détecter certains types de structures. Cette partie vise à illustrer l'adéquation de certaines stratégies dans certains cas et leurs limitations dans d'autres configurations.

2.1 Des serpentins

On considère tout d'abord les points de la figure suivante. Ils forment des amas aux formes serpentines.



Exercice 1

Téléchargez les points à l'url `http://195.220.111.226/R/serp.csv`.

1. Réalisez une classification ascendante hiérarchique reposant sur la distance euclidienne et la stratégie de Ward.
2. Testez les différents critères présentés en 1 pour essayer d'identifier un nombre de classes adapté (la valeur 3 semble-t-elle possible selon ces critères ?).
3. Pour les différentes valeurs de nombre de classes que vous aurez retenues, observez ces classes en utilisant un jeu de couleurs avec une couleur par classe.

La stratégie de Ward semble-t-elle proposer des classes correspondant à la structure des amas ?

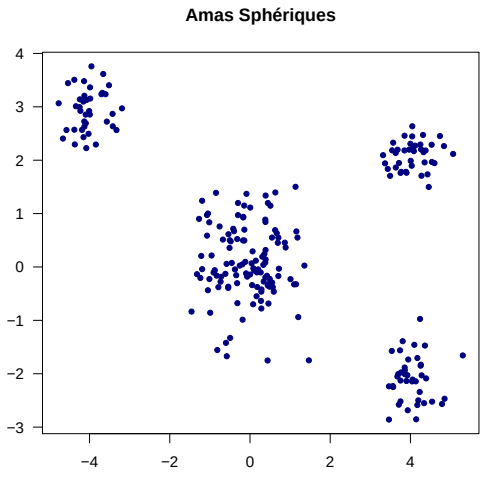
Exercice 2

Idem avec la stratégie du saut minimum, du saut max et de la dissimilarité moyenne.

2.2 Des amas sphériques

On considère à présent des ensembles de points formant des amas sphériques bien différenciés. A l'aide du code ci-dessous, construire une telle structure.

```
set.seed(11101)
library(MASS)
C1 <- mvrnorm(120, c(0,0), matrix(c(.5,0,0,.5),2,2))
C2 <- mvrnorm(40, c(4,2), matrix(c(.15,0,0,.15),2,2))
C3 <- mvrnorm(40, c(4,-2), matrix(c(.15,0,0,.15),2,2))
C4 <- mvrnorm(40, c(-4,3), matrix(c(.15,0,0,.15),2,2))
P <- rbind(C1, C2, C3, C4)
plot(P, pch = 19, cex = 0.75, col = 'navy', main = 'Amas Sphériques',
       xlab = "", ylab = "", las = 1)
```



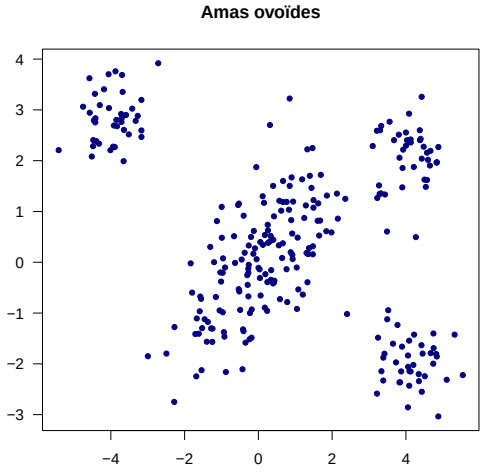
Exercice

Comme précédemment, essayez les différentes stratégies, calculez les différents indicateurs pour identifier les nombres de classes raisonnables et comparez enfin les capacités des stratégies à retrouver la structure en 4 groupes.

2.3 Des amas ovoïdes

On considère pour finir des ensembles de points formant des amas ovoïdes pas très bien différenciés. A l'aide du code ci-dessous, construire une telle structure.

```
set.seed(11102)
t = pi/6
C1 <- mvrnorm(150, c(0,0), 1.4*matrix(c(cos(t),sin(t),-sin(t),cos(t)),2,2))
C2 <- mvrnorm(40, c(4,2), matrix(c(.25,0,0,.25),2,2));
C3 <- mvrnorm(40, c(4,-2), matrix(c(.25,0,0,.25),2,2))
C4 <- mvrnorm(40, c(-4,3), matrix(c(.25,0,0,.25),2,2))
P <- rbind(C1, C2, C3, C4)
plot(P, pch = 19, cex = 0.75, col = 'navy', main = 'Amas ovoïdes',
       xlab = "", ylab = "", las = 1)
```



Exercice

Comme précédemment, essayez les différentes stratégies, calculez les différents indicateurs pour identifier les nombres de classes raisonnables et comparez enfin les capacités des stratégies à retrouver la structure en 4 groupes. Quelle est la stratégie la plus performante ?

3 Mesure de la complexité de la fonction `hclust`

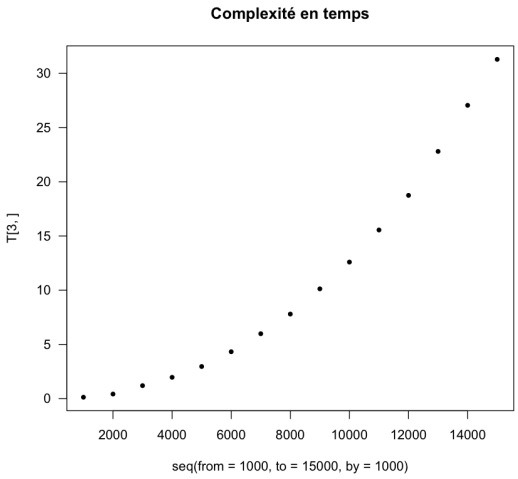
La classification ascendante hiérarchique fait intervenir des calculs qui peuvent nécessiter, dans leurs implémentations naïves, jusqu'à un facteur cubique du nombre d'observations. La complexité en temps calculatoire de cette méthode n'est pas très bonne et la fonction `hclust` n'est pas adaptée pour traiter des

volumes de données importants.

On va essayer ici de donner un ordre de grandeur de ce qu'il semble raisonnable de traiter avec la fonction `hclust`. Pour cela, on travaille sur un extrait d'un fichier [2] que l'on peut télécharger à l'adresse suivante `http://195.220.111.226/R/P200901_50000.csv`.

Téléchargez les données et placez-les dans une variable `big`. Le code ci-dessous mesure le temps d'exécution de la fonction `hclust` lorsque le nombre de lignes traitées par `hclust` augmente par tranche de 1000 :

```
T <- sapply(seq(from = 1000, to = 15000, by = 1000), function(n) system.time({
 cat(n,'\n');
 ind <- sort(sample(1:nrow(big), n));
 hclust(dist(big[ind,-1]), 'ward.D2');
}))
plot(seq(from = 1000, to = 15000, by = 1000), T[3,], pch = 20, las = 1,
       main = "Complexité en temps" )
```



La figure illustre que la complexité n'est pas en $o(n)$, mais plus vraisemblablement en $o(n^2)$ ou en $o(n^3)$. Il est possible d'utiliser R pour ajuster un polynome de degré 3 sur les points qu'on vient de déterminer. Pour calculer cette régression, on peut utiliser la fonction `lm` :

```
x <- seq(from = 1000, to = 15000, by = 1000)
M <- lm(T[3,] ~ I(x)+I(x^2)+I(x^3))
```

Exercice
À partir des valeurs renvoyées par la fonction `lm`, estimez le temps nécessaire - sous l'hypothèse que les ressources en mémoire ne seront pas épuisées au cours des calculs - pour construire une CAH pour 1 million de lignes de ce fichier.
Si votre ordinateur ne permet pas de tels calculs, on donne ci-dessous une combinaison linéaire obtenue sur une simulation :

2. version intégrale sur `http://open-data-assurance-maladie.ameli.fr/fic/depenses/Open_DAMIR/P200901.csv.gz`

coefficients	estimations
`intercept`	`-1.196e-01`
$I(x)$	`1.471e-04`
$I(x^2)$	`7.786e-08`
$I(x^3)$	`3.510e-12`

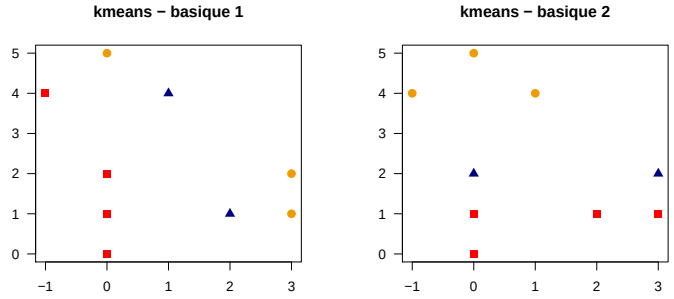
4 Introduction à la fonction `kmeans`

La fonction `kmeans` est une routine permettant d'aborder un autre type de problème de clustering : le nombre de classes à former étant fixé à K, on cherche à créer K groupes d'individus aussi homogènes que possible.

Etant donné une partition de départ en K groupes, on détermine pour chaque groupe son centre de gravité puis on reforme les groupes en associant ensemble les points qui sont les plus proches d'un centre de gravité. La procédure est itérée jusqu'à satisfaction d'un critère d'arrêt (généralement la stabilisation des groupes). Il faut bien comprendre qu'à l'issue des itérations, on espère avoir une partition de bonne qualité, mais il n'y a aucune garantie d'optimalité globale (par rapport à l'inertie intraclasse).

Exercice 1

La figure ci-dessous présente les mêmes points mais associés initialement différemment. En itérant la procédure de formation des groupes, déterminez pour chaque cas la configuration finale des groupes formés.



Notez qu'il est classique de remplacer les groupes de départ par des points qui joueront le rôle tenu par les centres de gravités - on parle de centres de classe. Ces points peuvent être choisis parmi les points à classer ou complétement aléatoirement.

La fonction `kmeans` implémente le schéma des méthodes des centres mobiles et en propose plusieurs déclinaisons. Elle prend en entrée les points et les centres de classes initiaux.

```
D <- matrix(c(0,0,0,1,0,2,2,1,3,1,3,2,0,5,1,4,-1,4), ncol = 2, byrow = TRUE)
C <- matrix(c(0,4,3/2,2.4,1,4/5), ncol = 2, byrow = TRUE)
kmeans(D,C)
```

```
K-means clustering with 3 clusters of sizes 3, 3, 3
Cluster means:
      [,1]     [,2]
1 0.000000 4.333333
2 2.666667 1.333333
3 0.000000 1.000000

Clustering vector:
[1] 3 3 3 2 2 2 1 1 1

Within cluster sum of squares by cluster:
[1] 2.666667 1.333333 2.000000
 (between_SS / total_SS =  85.2 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

La fonction renvoie les centres de classes, l'affectation des points aux classes numérotées, le R^2 et d'autres éléments.

Dans l'exemple, il est aisé d'observer les classes formées en jouant sur les couleurs des points :

```
KM <- kmeans(D,C)
couleur <- c("blue", "red", "orange2")
plot(D, asp = 1, pch = 19, col = couleur[KM$cluster])
points(KM$centers, pch = 8, col = couleur, cex = 2)
```

Parmi les objets renvoyés par la fonction, l'object `cluster` contient l'affectation des points aux classes et l'objet `centers` fournit les coordonnées des centres de classes. Il est également possible de calculer facilement le R^2 à partir de `totss` et `betweenss`.

Exercice 2

L'argument des centres de classes initiaux peut être remplacé par un simple entier, le nombre de classes à former. Dans ce cas, les centres de classes initiaux sont choisis aléatoirement. Il est évident que le résultat est fortement lié aux points initiaux. Afin d'observer cela, reprenez les données utilisées en 2.3 et faites plusieurs essais pour observer les variations du R^2.

Afin de maximiser les chances d'arriver à un résultat aussi bon que possible, il est possible de réaliser plusieurs essais et de garder celui correspondant au plus grand R^2. L'argument `nstart` permet de fixer le nombre d'essais - sa valeur implicite est fixée à 1. Donnez une grande valeur à ce paramètre et faites plusieurs essais en observant le R^2.

5 Les variantes des K-means à travers la fonction `kmeans`

On rappelle que la méthode des K-means consiste à faire évoluer la constitution des K groupes d'une partition des individus en les associant au centre de groupe le plus proche et en mettant à jour ces centres de classes en fonction des nouveaux groupes constitués.
La fonction `kmeans` propose 4 versions classiques de la méthode des K-means : la version de Lloyd puis celle de Forgy qui diffèrent sur le choix des centres

initiaux, puis les versions de Mac Queen et de Hartigan & Wong qui permettent de garantir la production de K classes.

5.1 Le risque des centres initiaux mal choisis

On prend les données générées par les lignes ci-dessous ainsi que les coordonnées des centres de classes `CK` utilisées pour initialiser notre partitionnement :

```
set.seed(11103)
C1 <- mvrnorm(120, c(0,0), matrix(c(0.5,0,0,0.5),2,2))
C2 <- mvrnorm(40, c(4,2), matrix(c(0.15,0,0,0.15),2,2))
C3 <- mvrnorm(40, c(4,-2), matrix(c(0.15,0,0,0.15),2,2))
C4 <- mvrnorm(40, c(-4,3), matrix(c(0.15,0,0,0.15),2,2))
P <- rbind(C1, C2, C3, C4)
CK <- matrix(c(-6,0,2,0,4,4,10,-1), ncol = 2, byrow = TRUE)
```

Exercice 1

Appliquez la fonction `kmeans` :

1) sur les points contenus dans P,

2) avec les centres indiqués dans CK,

3) avec la variante de Lloyd.

Combien de classes obtenez-vous en sortie ? L'examen du l'objet `cluster` permet de répondre à cette question, mais il peut être fastidieux. Il est aussi possible d'observer dans l'objet `centers` le nombre de centres pour lesquels aucune coordonnée n'a pu être calculée (faute d'individus dans la classe correspondante), ou plus simplement de tester la nullité de la taille des clusters (avec l'objet `size`).

Transformez votre commande afin d'obtenir les groupes formés au bout d'une seule itération et représentez-les en associant à chaque groupe une couleur distincte. Quelle est, selon vous, la raison de la diminution du nombre de classes ?

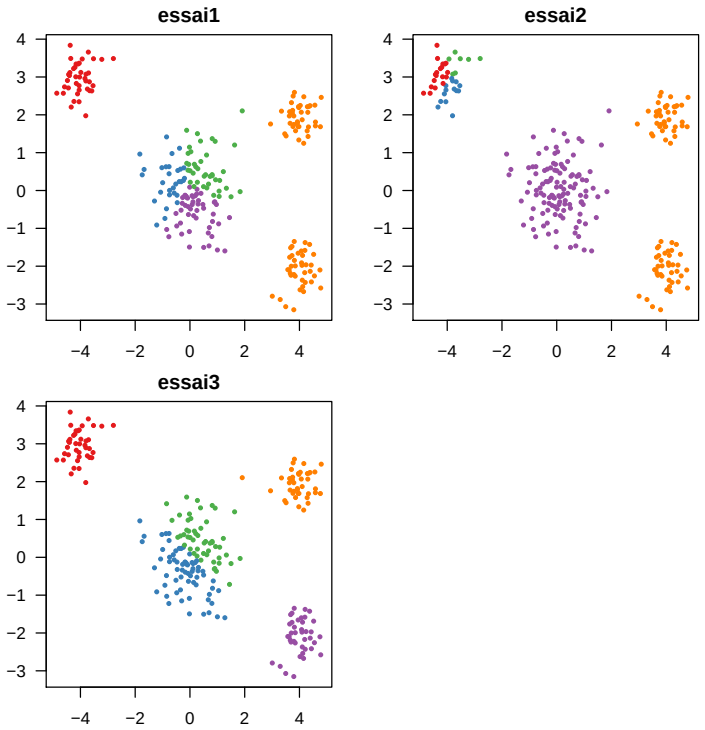
La version de Forgy propose une amélioration en choisissant les centres de classes aléatoirement parmi les individus. Ceci limite le risque de vidage de classes. La meilleure garantie reste implémentée dans les variantes de Mac Queen et de Hartigan-Wong qui mettent à jour les centres de classes dès qu'un point change de groupe.

Exercice 2

Faites un essai avec la variante de Forgy. Vous remplacerez l'argument donnant les coordonnées des centres par le nombre de centres à initialiser, 4 ici. Quel est le nombre de classes construites ?

5.2 Stabilité des classes construites

Reprenez vos précédentes commandes et réalisez plusieurs essais, avec les différentes variantes de Forgy, Hartigan et Mac Queen pour mesurer la stabilité des résultats. Il est assez facile d'observer une certaine diversité comme illustré sur la figure ci-dessous (variante de Forgy) :



Exercice

Afin de maximiser les chances d'identifier un partitionnement présentant la variance intragroupe la plus faible, il est possible de réaliser un grand nombre d'essais et de conserver la partition d'inertie intraclasse minimale. C'est l'objet de l'argument `nstart` de la fonction `kmeans`.

Réalisez de nouveaux essais en utilisant l'argument `nstart` avec une grande valeur (e.g. 100 ou 1000). Quelle est la variabilité dans les sorties produites ?

5.3 Performances des variantes des kmeans

La méthode de Lloyd présente le risque de construire des classes vides. De ce point de vue, celle de Forgy, Mac Queen et Hartigan & Wong sont meilleures. Ces deux dernières présentent la particularité de recalculer le centre de classes dès qu'une classe est modifiée. Quid de leurs performances ? La méthode de Hartigan & Wong est réputée être la meilleure...

Comme dans le sujet précédent, on va mesurer le temps d'exécution des variantes de la fonction utilisée en faisant varier le volume de données fournies en entrée. On travaille avec un fichier, téléchargeable à l'adresse `http://195.220.111.226/R/P200901_500000.csv`, comportant cette fois 500 000 lignes.

Exercice

Placez les données du fichier dans une variable `bigger` (sans omettre de contrôler que les données sont correctement lues) et adaptez les lignes ci-dessous, vues

lors du précédent TP, pour déterminer les temps d'éxecution pour les différentes variantes de l'algorithme utilisées par `kmeans` en fournissant en entrée un nombre de lignes augmentant par tranche de 10000 :

```
T <- sapply(seq(from=1000,to=1500,by=1000),function(n) system.time({
 cat(n,'\n');
 ind <- sort(sample(1:nrow(big),n));
 hclust(dist(big[ind,-1]),'ward.D2');
}))
plot(seq(from=100,to=1500,by=100),T[3,])
```

Représentez graphiquement les temps d'exécution pour chaque variante. L'hypothèse d'une complexité linéaire semble-t-elle vraisemblable ?

6 Parallélisation du code

R permet de tirer profit des machines dotées d'une architecture multicœurs. Le fait de trouver un intérêt à la multiplication des essais avec la fonction `kmeans` nous fournit un terrain d'expérimentation des bienfaits de la parallélisation de code dans le cadre relativement simple d'une machine multicœurs.

Exercice
On va utiliser le package `parallel`. Ce package est intégré à R depuis la version 2.14 et ne nécessite donc aucune installation (la version 3.3.2 est installée sur `tseetu.univ-lyon1.fr`).

Exécutez les lignes ci-dessous afin de déterminer le nombre de cœurs présents et utilisables par R.

```
library(parallel)
detectCores()
```

Mesurez le temps d'exécution de la fonction `kmeans` lorsque le nombre d'essai est 128.

```
X<-bigger[1:15000,]
T2 <- system.time({ KM <- kmeans(X,50,iter.max=300,nstart=128)})
```

Comparez le temps obtenu par les lignes ci-dessous :

```
system.time(mclapply(rep(128/detectCores(), detectCores()), function(ess)
        kmeans(X, 50, iter.max = 300, nstart = ess), mc.cores = detectCores())) -> T3
```

La fonction `mclapply` est une adaptation de la fonction `lapply` pour exploiter les architectures multicœurs. Elle répartit les appels de la fonction, donnée en deuxième argument, entre les cœurs en découpant les éléments de la liste fournie en premier argument. Ici, rien n'est fait pour récupérer la meilleure partition construite, l'objet étant seulement d'observer un gain de temps dans l'exécution. Plus de détails sur les moyens de paralléliser les calculs avec R dans [3].

Références

[1] A.D Gordon. *Classification*. Champman & Hall, 2nd edition, 1999

[2] S. Tufféry. *Data Mining et statistique décisionnelle - L'intelligence des données*. Éditions Technip, 2010

[3] V. Miele., V. Louvet. *Calcul parallèle avec R* . EDP Sciences, 2016