

Les méthodes d'ordination (ACP, AFC, etc) fournissent, comme leur nom l'indique, une ordination des individus, elles résument les données par un (ou plusieurs) score(s) numérique(s). Les méthodes de classification résument les données par une variable qualitative.

Calcul distance

```
D = dist( X, method = c( "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski" ) )
```

distance de Manhattan	1-distance	$\sum_{i=1}^n x_i - y_i $
distance euclidienne	2-distance	$\sqrt{\sum_{i=1}^n (x_i - y_i)^2}$
distance de Minkowski	p -distance	$\sqrt[p]{\sum_{i=1}^n x_i - y_i ^p}$
distance de Tchebychev	∞ -distance	$\lim_{p \rightarrow \infty} \sqrt[p]{\sum_{i=1}^n x_i - y_i ^p} = \sup_{1 \leq i \leq n} x_i - y_i $

NB : Tchebychev = “maximum”

CAH

Les méthodes ascendantes créent une partie en regroupant deux parties existantes.

```
cah = hclust(d = D, method = c("single", "complete", "average", "ward.D2"))
```

```
plot( cah, main = "Dendogramme", xlab = "")
```

Partition : `cutree(cah, h=hauteur ou k=nb de classe)`

La hauteur des nœuds = indice de dissimilarité entre les observations (+ la hauteur est importante, - la similarité des observations est importante)

Choix du nombre de classes

Critère du coude : on s'arrête au 1^{er} point d'infexion

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

R² ou coefficient de corrélation multiple : valeurs R² pour les différentes partitions issues du découpage de la cah

```
Rsq <- rep(0, nrow(X))
sum(scale(X, scale=FALSE)^2) -> SQTot
for(i in 1:nrow(X)) {
  Cla <- as.factor(cutree(cah,i))
  sum(t((t(sapply(1:ncol(X), function(i) tapply(X[,i], Cla, mean)))) - apply(X, 2, mean))^2) *
  as.vector(table(Cla))/SQTot -> RSQ[i]
}
Rsq
plot(Rsq, main="Valeurs de R2", xlab="nombre de classes", type='o', las=1)
```

Cette grandeur devrait être idéalement proche de 1. Puisque l'on cherche une grande inertie interclasse et une petite inertie intraclassée. $R^2 = \text{inertie interclasse} / \text{inertie totale}$

Pseudo F = inertie interclasse / inertie intraclassée

n = nrow(X)

```
pseudoF <- (Rsq[2:(n-1)]/((2:(n-1))-1)/((1-Rsq[2:(n-1)])/(n-(2:(n-1)))))
```

```
plot(pseudoF, main="Valeurs du pseudo F", xlab="nombre de classes", type='o')
```

Mesure de la complexité de la fonction hclust

```
x <- seq(from = 1000, to = 15000, by = 1000)
```

M <- lm(T[3,] ~ I(x) + I(x^2)) Approximation de la fonction T par un polynôme de degré 2

```
tempus <- function(n) M$coefficients[1] + M$coefficients[2]*n + M$coefficients[3]*n^2
```

tempus(n) résultats en secondes de temps nécessaire pour construire un CAH de n lignes de ce fichier

tempus(n)/86400 en jours

La méthode des Kmeans

```
K = kmeans(x, centers, iter.max = 10, nstart = 1, algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"))  
Partition = K$cluster  
plot(K,asp=1, pch=19, col=rainbow(max(K$cluster))[K$cluster])  
points(K$centers, col=rainbow(max(K$cluster))[K$cluster])  
K$betwents/K$totss
```

Les arguments de la fonction kmeans :

- 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. L'argument **iter.max** de la fonction kmeans permet de déterminer le nombre d'itérations souhaité. Par défaut **iter.max=10**, or il peut qu'il n'y ait pas de convergence en 10 itérations.
- 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.
- La fonction kmeans propose 4 versions classiques de la méthode des K-means : les versions de Lloyd, Forgy, Mac Queen et Hartigan-Wong. Cependant seules les versions de Mac Queen et de Hartigan-Wong permettent de garantir la production de K classes. Pour notre étude, nous avons choisi d'utiliser la version d'Hartigan-Wong (qui est la version par défaut).

K\$cluster	Vecteur qui donne à quelle classe appartient chaque élément
K\$centers	Matrice des centres de classes
K\$totss	Somme des carrés totale
K\$withinss	Vecteur de la somme des carrés à l'intérieur d'un groupe
K\$tot.withinss	Egal à sum(K\$withinss)
K\$betweenss	Egal à K\$totss - K\$tot.withinss
K\$size	nombre d'éléments dans chaque classe
K\$iter	nombre d'itérations
K\$ifault	indicateur d'un possible problème algorithmique