Classification non supervisée

masedki.github.io

jeu. 28 nov. 2019

Introduction

Introduction

La classification non supervisée a pour objectif de créer une typologie des individus d'une population, à partir d'un ensemble de variables collectées chez ces individus. On cherche à regrouper les individus en sous-groupes ou en classes homogènes.

Introduction

La classification non supervisée a pour objectif de créer une typologie des individus d'une population, à partir d'un ensemble de variables collectées chez ces individus. On cherche à regrouper les individus en sous-groupes ou en classes homogènes.

On distingue deux grands types de méthodes :

- méthodes géométriques (k-means, CAH, ...)
- méthodes probabilistes (modèles de mélange)

Introduction

La classification non supervisée a pour objectif de créer une typologie des individus d'une population, à partir d'un ensemble de variables collectées chez ces individus. On cherche à regrouper les individus en sous-groupes ou en classes homogènes.

On distingue deux grands types de méthodes :

- méthodes géométriques (k-means, CAH, ...)
- méthodes probabilistes (modèles de mélange)

Pour ce cours, on suppose que les variables explicatives sont **quantitatives**.

Définitions

Les méthodes de classification non supervisée nécessitent une **dissimilarité** entre les individus. Une dissimilarité est une **distance** qui ne vérifie pas nécessairement l'inégalité triangulaire.

Définitions

Les méthodes de classification non supervisée nécessitent une **dissimilarité** entre les individus. Une dissimilarité est une **distance** qui ne vérifie pas nécessairement l'inégalité triangulaire.

Définition. Une fonction $d : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}^+$ est une **dissimilarité** sur l'espace $\mathcal{X}$ si et seulement si

1. d est symétrique : $\forall (x_1, x_2) \in \mathcal{X} \times \mathcal{X},\ d(x_1, x_2) = d(x_2, x_1)$
2. d est positive : $\forall (x_1, x_2) \in \mathcal{X} \times \mathcal{X},\ d(x_1, x_2) \geq 0$
3. $\forall (x_1, x_2) \in \mathcal{X} \times \mathcal{X},\ d(x_1, x_2) = 0 \Leftrightarrow x_1 = x_2$

Définitions

Les méthodes de classification non supervisée nécessitent une **dissimilarité** entre les individus. Une dissimilarité est une **distance** qui ne vérifie pas nécessairement l'inégalité triangulaire.

Définition. Une fonction $d : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}^+$ est une **dissimilarité** sur l'espace $\mathcal{X}$ si et seulement si

1. d est symétrique : $\forall (x_1, x_2) \in \mathcal{X} \times \mathcal{X},\ d(x_1, x_2) = d(x_2, x_1)$
2. d est positive : $\forall (x_1, x_2) \in \mathcal{X} \times \mathcal{X},\ d(x_1, x_2) \geq 0$
3. $\forall (x_1, x_2) \in \mathcal{X} \times \mathcal{X},\ d(x_1, x_2) = 0 \Leftrightarrow x_1 = x_2$

Définition. Une fonction $d : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}^+$ est une **distance** sur l'espace $\mathcal{X}$ si et seulement si

1. d est une dissimilarité sur l'espace $\mathcal{X}$
2. $\forall (x_1, x_2, x_3) \in \mathcal{X} \times \mathcal{X} \times \mathcal{X},\ d(x_1, x_2) \leq d(x_1, x_3) + d(x_3, x_2)$

Inertie

Les principaux algorithmes de clustering reposent sur la notion d'**inertie** du nuage de points (associée à une dissimilarité).

Définition. Soit $x_1, \ldots, x_n$ un ensemble de points de $\mathbb{R}^p$, l'inertie du nuage de points est définie par

$$I_{\text{Tot}} = \sum_{i=1}^{n} d^2(x_i, x_G).$$

où $x_G = \frac{1}{n} \sum_{i=1}^{n} x_i$ est le centre de gravité du nuage de points et d une dissimilarité.

Inertie

Supposons que le nuage de points soit composé de K classes notées $C_1, C_2, \ldots, C_K$ et formant une partition $\mathcal{P}$. L'inertie totale du nuage de points peut alors se décomposer comme

$$
\begin{aligned}
I_{\text{Tot}} &= \sum_{k=1}^{K} \sum_{x_i \in C_k} d^2(x_i, x_G) \\
&= \sum_{k=1}^{K} \sum_{x_i \in C_k} \left(d^2(x_i, x_{G_k}) + d^2(x_{G_k}, x_G) \right) \\
&= \underbrace{\sum_{k=1}^{K} \sum_{x_i \in C_k} d^2(x_i, x_{G_k})}_{I_W(\mathcal{P})} + \underbrace{\sum_{k=1}^{K} n_k d^2(x_{G_k}, x_G)}_{I_B(\mathcal{P})}
\end{aligned}
$$

où n_k est le nombre d'éléments dans la classe C_k et où on a appliqué le théorème de Huygens.

Décomposition de l'inertie totale (suite)

Le théorème de Huygens

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - a)^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 + (\bar{x} - a)^2$$

Inertie

Le terme $I_W(\mathcal{P}) = \sum_{k=1}^{K} \sum_{x_i \in C_k} d^2(x_i, x_{G_k})$ est appelé *inertie intra-classes*. Plus les classes sont homogènes, c'est à dire plus les points de chaque classes sont proches du centre de gravité de la classes, plus ce terme est petit.

Inertie

Le terme $I_W(\mathcal{P}) = \sum_{k=1}^{K} \sum_{x_i \in C_k} d^2(x_i, x_{G_k})$ est appelé *inertie intra-classes*. Plus les classes sont homogènes, c'est à dire plus les points de chaque classes sont proches du centre de gravité de la classes, plus ce terme est petit.

Le terme $I_B(\mathcal{P}) = \sum_{k=1}^{K} n_k d^2(x_{G_k}, x_G)$ est appelé *inertie inter-classes*. Il mesure la distance entre les classes : plus les classes sont éloignées les unes des autres, plus ce terme sera élevé.

Inertie

Le terme $I_W(\mathcal{P}) = \sum_{k=1}^{K} \sum_{x_i \in C_k} d^2(x_i, x_{G_k})$ est appelé *inertie intra-classes*. Plus les classes sont homogènes, c'est à dire plus les points de chaque classes sont proches du centre de gravité de la classes, plus ce terme est petit.

Le terme $I_B(\mathcal{P}) = \sum_{k=1}^{K} n_k d^2(x_{G_k}, x_G)$ est appelé *inertie inter-classes*. Il mesure la distance entre les classes : plus les classes sont éloignées les unes des autres, plus ce terme sera élevé.

Plus les points du nuage sont répartis en classes homogènes et bien distinctes les unes des autres, plus l'inertie intra-classes est faible (l'inertie totale étant constante, plus l'inertie inter-classes est élevée).

Inertie

Une première idée peut consister à chercher la partition minimisant l'inertie intra-classes. Cela donne la solution triviale d'une partition élémentaire à n classes (chaque classe est formée par une observation). Ainsi, comparer des partitions par l'inertie intra-classes, n'a de sens qu'à nombre de classes fixé.

Inertie

Une première idée peut consister à chercher la partition minimisant l'inertie intra-classes. Cela donne la solution triviale d'une partition élémentaire à n classes (chaque classe est formée par une observation). Ainsi, comparer des partitions par l'inertie intra-classes, n'a de sens qu'à nombre de classes fixé.

Une second idée peut alors consister, pour K fixé, à comparer toutes les partitions de taille K possibles, et de choisir la meilleure au sens de l'inertie inter-classes. Mais le nombre de partitions possibles de n éléments en K classes est le nombre de Stirling de seconde espèce :

$$p_{n,K} = \frac{1}{K!} \sum_{k=0}^{K} (-1)^{k-1} \binom{n}{k} k^n$$

Inertie

Une première idée peut consister à chercher la partition minimisant l'inertie intra-classes. Cela donne la solution triviale d'une partition élémentaire à n classes (chaque classe est formée par une observation). Ainsi, comparer des partitions par l'inertie intra-classes, n'a de sens qu'à nombre de classes fixé.

Une second idée peut alors consister, pour K fixé, à comparer toutes les partitions de taille K possibles, et de choisir la meilleure au sens de l'inertie inter-classes. Mais le nombre de partitions possibles de n éléments en K classes est le nombre de Stirling de seconde espèce :

$$p_{n,K} = \frac{1}{K!} \sum_{k=0}^{K} (-1)^{k-1} \binom{n}{k} k^n$$

En pratique une exploration exhaustive de toutes les partitions n'est donc pas faisable. Les algorithmes que nous présentons dans ce cours se contentent d'explorer un sous-ensemble des partitions possibles.

K-means

Description de l'algorithme

On considère que le nuage de points de $\mathbb{R}^p$ est muni de la distance euclidienne.

L'algorithme des K-means est un algorithme de partitionnement qui fournit une partition en K classes d'un ensemble de n points.

Description de l'algorithme

On considère que le nuage de points de $\mathbb{R}^p$ est muni de la distance euclidienne.

L'algorithme des K-means est un algorithme de partitionnement qui fournit une partition en K classes d'un ensemble de n points.

Idée générale En partant d'un premier ensemble de K points $\mu_1, \ldots, \mu_K$ que l'on désigne comme centres des classes (souvent choisis au hasard), on regroupe dans la classe C_k tous les points qui sont plus proches de μ_k que des autres centres.

Description de l'algorithme

On considère que le nuage de points de $\mathbb{R}^p$ est muni de la distance euclidienne.

L'algorithme des K-means est un algorithme de partitionnement qui fournit une partition en K classes d'un ensemble de n points.

Idée générale En partant d'un premier ensemble de K points $\mu_1, \ldots, \mu_K$ que l'on désigne comme centres des classes (souvent choisis au hasard), on regroupe dans la classe C_k tous les points qui sont plus proches de μ_k que des autres centres.

Une fois cette première partitions obtenue, on définit les nouveaux centres des classes en calculant les centres de gravités des classes précédemment obtenues.

Description de l'algorithme

On considère que le nuage de points de $\mathbb{R}^p$ est muni de la distance euclidienne.

L'algorithme des K-means est un algorithme de partitionnement qui fournit une partition en K classes d'un ensemble de n points.

Idée générale En partant d'un premier ensemble de K points $\mu_1, \ldots, \mu_K$ que l'on désigne comme centres des classes (souvent choisis au hasard), on regroupe dans la classe C_k tous les points qui sont plus proches de μ_k que des autres centres.

Une fois cette première partitions obtenue, on définit les nouveaux centres des classes en calculant les centres de gravités des classes précédemment obtenues.

Enfin, on réitère le procédé en affectant les points à la classe dont le centre est le plus proche, puis en recalculant les centres des classes, jusqu'à convergence vers une partition stable (inchangée d'une itération à l'autre).

Propriétés de l'algorithme

Pour une partition $\mathcal{P} = \{C_1, \ldots, C_K\}$ et un ensemble de points $\theta = \{\mu_1, \ldots, \mu_K\}$, on définit le critère suivant :

$$C(\mathcal{P}, \theta) = \sum_{k=1}^{K} \sum_{i=1}^{n} z_{ik} ||x_i - \mu_k||^2$$

avec $z_{ik} = 1$ si $x_i \in C_k$ et $z_{ik} = 0$ sinon.

En particulier, si θ contient les centres de gravité des classes de la partition $\mathcal{P}$, on retombe sur la somme des inerties intra-classes obtenues en considérant la distance euclidienne.

Algorithme des K-means

Données : Un n-échantillon $(x_1, \ldots, x_n)$
Résultat : Une partition $\mathcal{P}$
Initialisation : on choisit K points dans $\mathcal{X}$ que l'on définit comme les centres $\mu_1^{[0]}, \ldots, \mu_K^{[0]}$ des K classes
r = 0 ;

1 **tant que** *la partition n'est pas stable* **faire**

2 Définir la partition

$$z_{ik}^{[r]} = \begin{cases} 1 & \text{si } ||x_i - \mu_k^{[r]}|| \leq ||x_i - \mu_\ell^{[r]}||, \forall \ell = 1, \ldots, K \\ 0 & \text{sinon} \end{cases}$$

Mettre à jour les centres

$$\mu_k^{[r+1]} = \frac{\sum_{i=1}^n z_{ik}^{[r]} x_i}{\sum_{i=1}^n z_{ik}^{[r]}}$$

r = r + 1

3 **fin**

Algorithme 1 : Algorithme des K-means

Exemple : Iris de Fisher

On illustre le comportement de l'algorithme sur les données des iris de Fisher, contenant les longueurs et largeurs de sépales et des pétals de 150 fleurs apparenant à 3 espèces d'iris.

Voici les itérations de l'algorithme avec $K = 3$

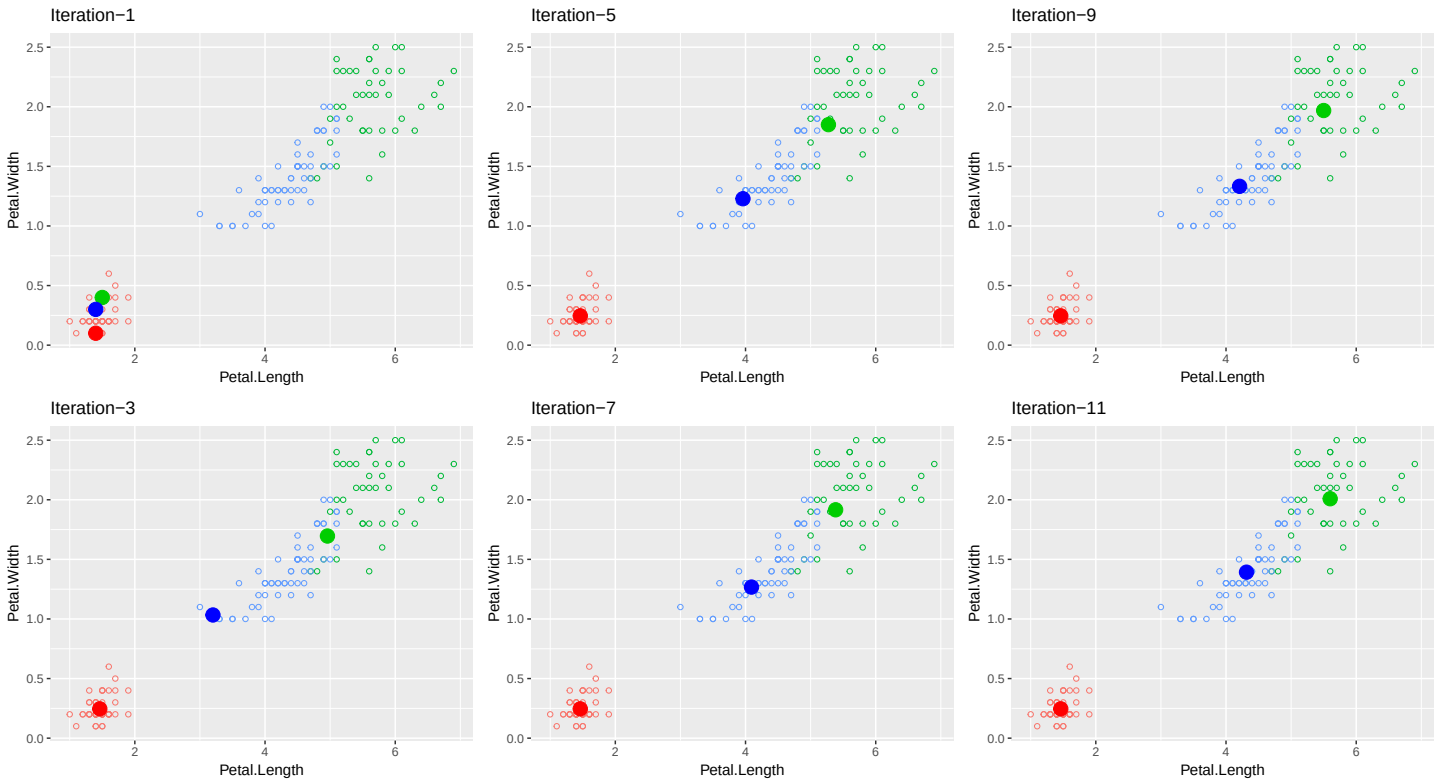


FIGURE 1 – Itération de l'algorithme des K-means sur les Iris de Fisher

Les K-means sous R

```r
data("iris")
cl <- kmeans(iris[,1:4], 3) # clustering
cluster.x <- cl$cluster     # partition
# cluster centers "fitted to each obs"
fitted.x <- fitted(cl)
head(fitted.x)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1        5.006       3.428        1.462       0.246
1        5.006       3.428        1.462       0.246
1        5.006       3.428        1.462       0.246
1        5.006       3.428        1.462       0.246
1        5.006       3.428        1.462       0.246
1        5.006       3.428        1.462       0.246
1        5.006       3.428        1.462       0.246
```

```r
resid.x <- iris[,1:4] - fitted(cl)
```

Algorithme des K-means

Proposition. À chaque itération de l'algorithme, le critère C est amélioré (*i.e.*, diminué), et on a

$$C(\mathcal{P}^{[r]}, \theta^{[r]}) \leq C(\mathcal{P}^{[r-1]}, \theta^{[r]})$$

$$C(\mathcal{P}^{[r]}, \theta^{[r+1]}) \leq C(\mathcal{P}^{[r]}, \theta^{[r]})$$

Algorithme des K-means

Montrons d'abord que

$$C(\mathcal{P}^{[r]}, \theta^{[r]}) \leq C(\mathcal{P}^{[r-1]}, \theta^{[r]})$$

Par définition, on a

$$z_{ik}^{[r]} = \begin{cases} 1 & \text{si } ||x_i - \mu_k^{[r]}|| \leq ||x_i - \mu_\ell^{[r]}||, \forall \ell = 1, \ldots, K \\ 0 & \text{sinon} \end{cases}$$

ainsi,

$$\sum_{k=1}^{K} z_{ik}^{[r]} ||x_i - \mu_k^{[r]}|| \leq \sum_{\ell=1}^{K} z_{ik}^{[r-1]} ||x_i - \mu_\ell^{[r]}||.$$

Algorithme des K-means

Montrons d'abord que

$$C(\mathcal{P}^{[r]}, \theta^{[r]}) \leq C(\mathcal{P}^{[r-1]}, \theta^{[r]})$$

Par définition, on a

$$z_{ik}^{[r]} = \begin{cases} 1 & \text{si } ||x_i - \mu_k^{[r]}|| \leq ||x_i - \mu_\ell^{[r]}||, \forall \ell = 1, \ldots, K \\ 0 & \text{sinon} \end{cases}$$

ainsi,

$$\sum_{k=1}^{K} z_{ik}^{[r]} ||x_i - \mu_k^{[r]}|| \leq \sum_{\ell=1}^{K} z_{ik}^{[r-1]} ||x_i - \mu_\ell^{[r]}||.$$

D'où en sommant sur les classes, et comme les centres restent inchangés

$$C(\mathcal{P}^{[r]}, \theta^{[r]}) \leq C(\mathcal{P}^{[r-1]}, \theta^{[r]})$$

Algorithme des K-means

Montrons maintenant que

$$C(\mathcal{P}^{[r]}, \theta^{[r+1]}) \leq C(\mathcal{P}^{[r]}, \theta^{[r]})$$

Par définition, on a

$$\mu_k^{[r+1]} = \frac{\sum_{i=1}^{n} z_{ik}^{[r]} x_i}{\sum_{i=1}^{n} z_{ik}^{[r]}}$$

$$C(\mathcal{P}^{[r]}, \theta^{[r+1]}) \leq C(\mathcal{P}^{[r]}, \theta^{[r]})$$

$$\begin{aligned}
C(\mathcal{P}^{[r]}, \theta^{[r]}) &= \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik}^{[r]} ||x_i - \mu_k^{[r]}||^2 \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik}^{[r]} ||x_i - \mu_k^{[r+1]} + \mu_k^{[r+1]} - \mu_k^{[r]}||^2 \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik}^{[r]} ||x_i - \mu_k^{[r+1]}||^2 + ||\mu_k^{[r+1]} - \mu_k^{[r]}||^2 \\
&- 2 \underbrace{< x_i - \mu_k^{[r+1]}, \mu_k^{[r+1]} - \mu_k^{[r]} >}_{=0} \\
&= C(\mathcal{P}^{[r]}, \theta^{[r+1]}) + \underbrace{||\mu_k^{[r+1]} - \mu_k^{[r]}||^2}_{\geq 0}
\end{aligned}$$

Algorithme des K-means

Corollaire. L'algorithme des K-means minimise l'inertie intra-classes.

Preuve. Immédiate en notant que $C(\mathcal{P}^{[r]}, \theta^{[r]}) = I_W(\mathcal{P}^{[r]})$.

Algorithme des K-means

Corollaire. L'algorithme des K-means minimise l'inertie intra-classes.

Preuve. Immédiate en notant que $C(\mathcal{P}^{[r]}, \theta^{[r]}) = I_W(\mathcal{P}^{[r]})$.

Corollaire. La suite numérique $C(\mathcal{P}^{[r]}, \theta^{[r]})$ est stationnaire.

Preuve. On a montré que la suite $C(\mathcal{P}^{[r]}, \theta^{[r]})$ est décroissante. De plus, elle ne peut prendre qu'un nombre fini de valeurs car il n'y a qu'un nombre fini de partitions $\mathcal{P}^{[r]}$ et donc de centres $\theta^{[r]}$.

Algorithme des K-means

Corollaire. L'algorithme des K-means minimise l'inertie intra-classes.

Preuve. Immédiate en notant que $C(\mathcal{P}^{[r]}, \theta^{[r]}) = I_W(\mathcal{P}^{[r]})$.

Corollaire. La suite numérique $C(\mathcal{P}^{[r]}, \theta^{[r]})$ est stationnaire.

Preuve. On a montré que la suite $C(\mathcal{P}^{[r]}, \theta^{[r]})$ est décroissante. De plus, elle ne peut prendre qu'un nombre fini de valeurs car il n'y a qu'un nombre fini de partitions $\mathcal{P}^{[r]}$ et donc de centres $\theta^{[r]}$.

Proposition. La suite $(\mathcal{P}^{[r]}, \theta^{[r]})$ est stationnaire.

Preuve. Notons que la stationnarité de $C(\mathcal{P}^{[r]}, \theta^{[r]})$ ne suffit pas à montrer la stationnarité de $(\mathcal{P}^{[r]}, \theta^{[r]})$ (plusieurs combinaisons $(\mathcal{P}^{[r]}, \theta^{[r]})$ pourraient donner la même valeur de C).

Algorithme des K-means

Proposition. La suite $(\mathcal{P}^{[r]}, \theta^{[r]})$ est stationnaire.

Preuve (suite). Comme la suite $C(\mathcal{P}^{[r]}, \theta^{[r]})$ est stationnaire, on a

$$\exists N,\ \forall r > N, C(\mathcal{P}^{[r]}, \theta^{[r]}) = C(\mathcal{P}^{[r+1]}, \theta^{[r+1]})$$

Donc, comme $C(\mathcal{P}^{[r]}, \theta^{[r]}) \leq C(\mathcal{P}^{[r]}, \theta^{[r]})$, on a également $C(\mathcal{P}^{[r]}, \theta^{[r]}) = C(\mathcal{P}^{[r]}, \theta^{[r]})$, où $\theta^{[r+1]}$ sont les centres de gravité des classes de la partition $\mathcal{P}^{[r+1]}$. Or pour la partition données, le critère C atteint son minimum en les centres de gravité des classes de la partition. Autrement dit, on a

$$C(\mathcal{P}^{[r]}, \theta^{[r]}) = \min_{\theta} C(\mathcal{P}^{[r]}, \theta)$$

De plus, d'après le théorème de Huygens, ce minimum est atteint de façon unique par les centres de gravité des classes de la partition. Donc, on a $\theta^{[r+1]} = \theta^{[r]}$. La partition étant définie de façon unique à partir des centres $\theta^{[r]}$, on a également $\mathcal{P}^{[r+1]} = \mathcal{P}^{[r]}$.

Algorithme des K-means

Remarque. Même si la suite $(\mathcal{P}^{[r]}, \theta^{[r]})$ est stationnaire, ce qui implique la convergence de l'algorithme), cette convergence dépend de l'état initial de l'algorithme. La convergence n'est donc que locale. Dans la pratique, il faut lancer plusieurs fois l'algorithme en considérant différentes initialisations et ne retenir que la partition correspondant à l'inertie intra-classes minimale.

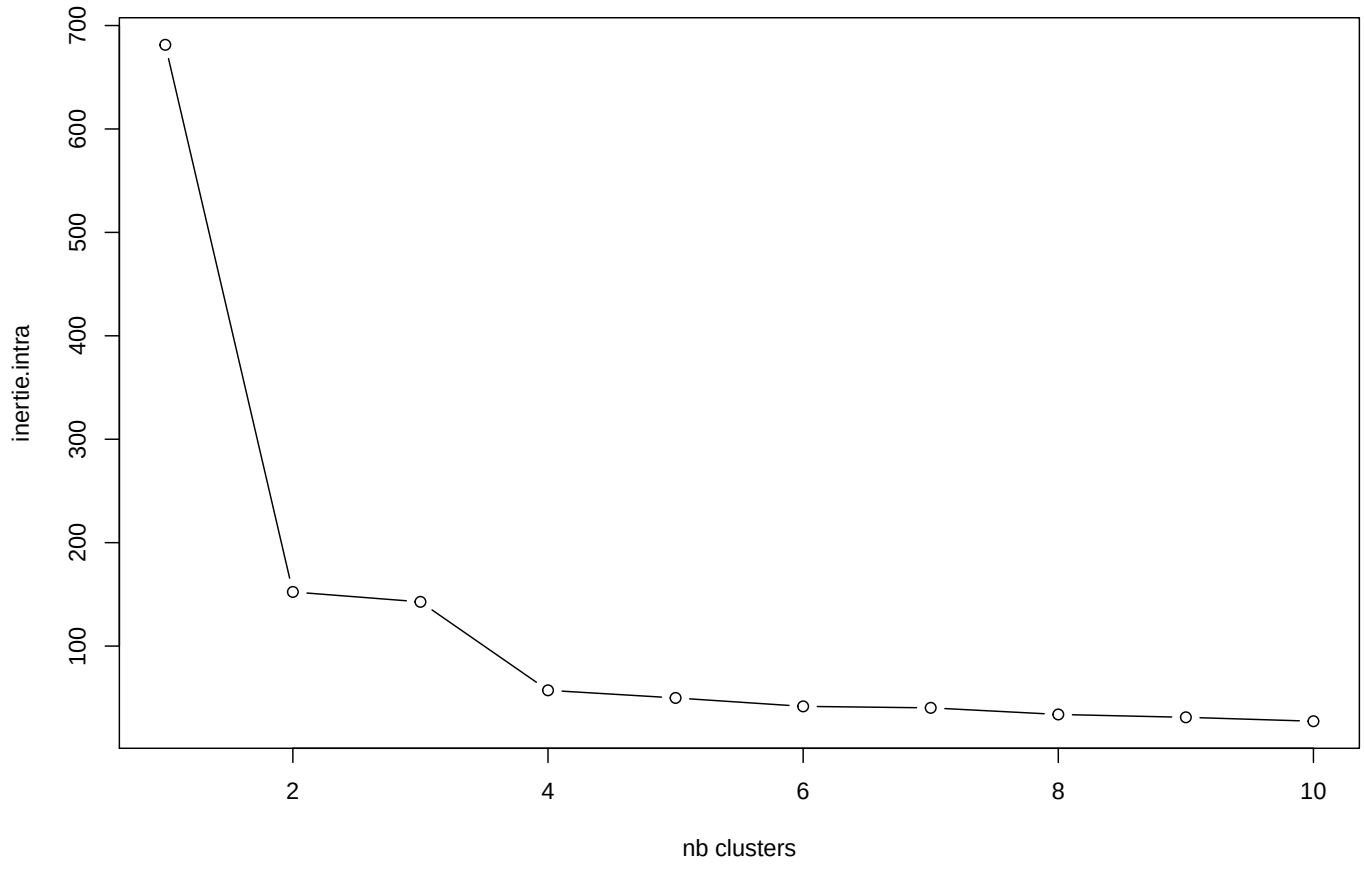
Algorithme des K-means

Remarque. Même si la suite $(\mathcal{P}^{[r]}, \theta^{[r]})$ est stationnaire, ce qui implique la convergence de l'algorithme), cette convergence dépend de l'état initial de l'algorithme. La convergence n'est donc que locale. Dans la pratique, il faut lancer plusieurs fois l'algorithme en considérant différentes initialisations et ne retenir que la partition correspondant à l'inertie intra-classes minimale.

Remarque. Utiliser la distance euclidienne implique que les classes obtenues sont sphériques. On peut généraliser en prenant la distance induite par le produit scalaire sur $\mathbb{R}^p$: $< x_1, x_2 >_M = x_1^\top M x_2$, la distance euclidienne étant obtenue en prenant $M = \mathbf{I}_p$. Certains cas ne seront pas bien traités par des classes elliptiques, ainsi une transformation préalable des données peut être effectuée (comme en ACP en normalisant les données).

Les K-means sous R ; choix du nombre de classes

```
cl <- lapply(1:10, function(k) kmeans(iris[,1:4], k))
inertie.intra <- sapply(cl, function(u) u$tot.withinss)
plot(1:10, inertie.intra, type="b", xlab="nb clusters")
```



Variantes

L'algorithme Partitionning Around Medoïds (PAM) permet de classifier des données de façon plus robuste (*i.e.*, moins sensible à des valeurs atypiques). Le noyau d'une classe est alors un médoïd (*i.e.*, l'observation d'une classe qui minimise la moyenne des distances aux autres observations de la classes).

Variantes

L'algorithme Partitionning Around Medoïds (PAM) permet de classifier des données de façon plus robuste (*i.e.*, moins sensible à des valeurs atypiques). Le noyau d'une classe est alors un médoïd (*i.e.*, l'observation d'une classe qui minimise la moyenne des distances aux autres observations de la classes).

Une différence majeur avec l'algorithme des K-means est qu'un médoïd fait partie des données et permet donc de partitionner des matrices de dissimilarités. En contre-partie, il est limité par le nombre d'observations (matrice de dissimilarités à stocker) et en temps de calcul (algorithme en $\mathcal{O}(n^2)$).

Variantes

L'algorithme Partitionning Around Medoïds (PAM) permet de classifier des données de façon plus robuste (*i.e.*, moins sensible à des valeurs atypiques). Le noyau d'une classe est alors un médoïd (*i.e.*, l'observation d'une classe qui minimise la moyenne des distances aux autres observations de la classes).

Une différence majeur avec l'algorithme des K-means est qu'un médoïd fait partie des données et permet donc de partitionner des matrices de dissimilarités. En contre-partie, il est limité par le nombre d'observations (matrice de dissimilarités à stocker) et en temps de calcul (algorithme en $\mathcal{O}(n^2)$).

D'autres algorithmes ont été proposés pour des types de données spécifiques : K-modes pour les variables qualitatives et K-prototypes pour des variables mixtes.

PAM sous R

```
require(cluster)
```

```
Loading required package: cluster
```

```
cl.pam <- pam(iris[,1:4], 3) # clustering
medoids.x <- cl.pam$medoids  # medoids
mediods.id <- cl.pam$id.med  # medoids ID
cl.x <- cl.pam$clustering    # partition
```

CAH

Description de l'algorithme

La classification ascendante hiérachique (CAH) a pour objectif de construire une suite de partitions emboîtées de l'ensemble des points $\{x_1, \ldots, x_n\}$.

Description de l'algorithme

La classification ascendante hiérachique (CAH) a pour objectif de construire une suite de partitions emboîtées de l'ensemble des points $\{x_1, \ldots, x_n\}$.

En partant de la partition initiale à n classes, l'étape k consiste à agréger deux classes entre elles pour former une partition à $n - k$ classes, jusqu'à ce que l'on ait regroupé tous les points dans la même classe. On l'appelle *ascendante* car elle part d'une partition initiale en n singletons pour aboutir à une partition en 1 classe.

Description de l'algorithme

La classification ascendante hiérachique (CAH) a pour objectif de construire une suite de partitions emboîtées de l'ensemble des points $\{x_1, \ldots, x_n\}$.

En partant de la partition initiale à n classes, l'étape k consiste à agréger deux classes entre elles pour former une partition à $n - k$ classes, jusqu'à ce que l'on ait regroupé tous les points dans la même classe. On l'appelle *ascendante* car elle part d'une partition initiale en n singletons pour aboutir à une partition en 1 classe.

Cet algorithme nécessite de définir une notion de distance/dissimilarité d entre les observations ainsi qu'un critère d'agrégation D qui mesure la distance entre les classes.

Description de l'algorithme

Données : Un n-échantillon $(x_1, \ldots, x_n)$
Résultat : Un ensemble de partitions $\mathcal{P}_0, \ldots, \mathcal{P}_{n-1}$
Initialisation : on choisit pour $\mathcal{P}_0$ la partition à n points où chaque point constitue une classe à part entière

1 **pour** *k=1,. . . ,n-1* **faire**
2 | Calculer les distances deux à deux entre les classes de la partition $\mathcal{P}_{k-1}$ à l'aide de la dissimilarité d;
3 | Former la partition $\mathcal{P}_k$ en regroupant les classes se trouvant à distance minimale;
4 **fin**

Algorithme 2 : Classification ascendante hiérarchique

Critères d'agrégation

Les critères d'agrégation D les plus utilisés entre deux parties A et B de centres de gravité respectifs x_A et x_B sont

- le critère de Ward :

$$D_W(A,B) = \frac{n_A n_B}{n_A + n_B} d(x_A, x_B)^2$$

- le critère du saut minimum (single linkage)

$$D_{\min}(A,B) = \min\{d(x,y) \mid x \in A, y \in B\}$$

- le critère du saut maximum (complete linkage)

$$D_{\max}(A,B) = \max\{d(x,y) \mid x \in A, y \in B\}$$

- le critère de la distance moyenne (average linkage)

$$D_{\text{moy}}(A,B) = \frac{1}{n_A n_B} \sum_{x \in A} \sum_{y \in B} d(x,y)$$

où n_A est le cardinal de A.

Critère de Ward

Proposition. L'utilisation du critère de Ward permet de regrouper à chaque étape de l'algorithme de CAH les classes dont la fusion permet le plus faible gain d'inertie intra-classes, ou de façon équivalente la plus faible perte d'inertie inter-classes.

Preuve. Soit Δ_k la perte d'inertie inter-classes entre les itérations $k-1$ et k de l'algorithme. Notons C_h et C_ℓ les deux classes ayant été regroupées lors de l'itération k. Le reste des classes restant inchangé, la différence d'inertie inter-classes entre les deux partitions provient uniquement de la différence d'inertie inter-classes entre les deux classes fusionnées et les deux classes séparées. Notons x_{C_h} le barycentre de la classe C_h, x_G le barycentre du nuage de points et x_C le barycentre de la classe fusionnée. On a alors

$$\Delta_k = n_h d^2(x_{C_h}, x_G) + n_\ell d^2(x_{C_\ell}, x_G) - (n_h + n_\ell) d^2(x_C, x_G).$$

Critère de Ward

$$\Delta_k = n_h||x_{C_h} - x_G||^2 + n_\ell||x_{C_j} - x_G||^2 - (n_h + n_\ell)||x_C - x_G||^2$$

Critère de Ward

$$\Delta_k = n_h||x_{C_h} - x_G||^2 + n_\ell||x_{C_j} - x_G||^2 - (n_h + n_\ell)||x_C - x_G||^2$$

Or, par définition du barycentre, on a

$$x_C = \frac{n_h}{n_h + n_\ell} x_{C_h} + \frac{n_\ell}{n_h + n_\ell} x_{C_\ell}$$

Critère de Ward

$$\Delta_k = n_h||x_{C_h} - x_G||^2 + n_\ell||x_{C_j} - x_G||^2 - (n_h + n_\ell)||x_C - x_G||^2$$

Or, par définition du barycentre, on a

$$x_C = \frac{n_h}{n_h + n_\ell} x_{C_h} + \frac{n_\ell}{n_h + n_\ell} x_{C_\ell}$$

On a alors

$$\begin{aligned}
(n_h + n_\ell)||x_C - x_G||^2 &= (n_h + n_\ell)||\frac{n_h}{n_h + n_\ell} x_{C_h} + \frac{n_\ell}{n_h + n_\ell} x_{C_\ell} - x_G||^2 \\
&= \frac{1}{n_h + n_\ell}||n_h x_{C_h} + n_\ell x_{C_\ell} - (n_h + n_\ell) x_G||^2 \\
&= \frac{n_h^2}{n_h + n_\ell}||x_{C_h} - x_G||^2 + \frac{n_\ell^2}{n_h + n_\ell}||x_{C_\ell} - x_G||^2 + \\
&\frac{2 n_h n_\ell}{n_h + n_\ell} < x_{C_h} - x_G, x_{C_\ell} - x_G >
\end{aligned}$$

Ainsi

Dendrogramme

Une façon usuelle de représenter les étapes successives de la CAH est celle du *dendrogramme*.

Dendrogramme

Une façon usuelle de représenter les étapes successives de la CAH est celle du *dendrogramme*.

Il s'agit d'une arborescence : les feuilles de l'arbre se trouvent en bas, et représentent les individus.

Dendrogramme

Une façon usuelle de représenter les étapes successives de la CAH est celle du *dendrogramme*.

Il s'agit d'une arborescence : les feuilles de l'arbre se trouvent en bas, et représentent les individus.

Chaque niveau de l'arborescence correspond au regroupement de deux classes du niveau précédent.

Dendrogramme

Une façon usuelle de représenter les étapes successives de la CAH est celle du *dendrogramme*.

Il s'agit d'une arborescence : les feuilles de l'arbre se trouvent en bas, et représentent les individus.

Chaque niveau de l'arborescence correspond au regroupement de deux classes du niveau précédent.

On représente ainsi les classes fusionnées à l'aide d'une branche les reliant, et la hauteur de la branche est proportionnelle à la distance entre ces classes.

Dendrogramme

Une façon usuelle de représenter les étapes successives de la CAH est celle du *dendrogramme*.

Il s'agit d'une arborescence : les feuilles de l'arbre se trouvent en bas, et représentent les individus.

Chaque niveau de l'arborescence correspond au regroupement de deux classes du niveau précédent.

On représente ainsi les classes fusionnées à l'aide d'une branche les reliant, et la hauteur de la branche est proportionnelle à la distance entre ces classes.

Au sommet de l'arbre, on retrouve la classe contenant tous les individus.

CAH sous R

```r
# CAH des iris avec critère de Ward
agn1 <- agnes(iris[,1:4],
              metric = "euclidean",
              method = "average")
# Dendrogramme
plot(agn1)
```

CAH sous R

```r
# CAH des iris avec critère du saut minimum
agn2 <- agnes(iris[,1:4],
              metric = "euclidean",
              method = "single")
# Dendrogramme
plot(agn2)
```

Partitionnement spectral

Introduction

Les algorithmes présentés dans les sections précédentes fonctionnent bien surtout lorsque les données sont réparties dans des classes convexes.

Introduction

Les algorithmes présentés dans les sections précédentes fonctionnent bien surtout lorsque les données sont réparties dans des classes convexes.

Les *méthodes de partitionnement spectrales*, quant à elles, fonctionnent également lorsque les classes ne sont pas convexes.

Introduction

Les algorithmes présentés dans les sections précédentes fonctionnent bien surtout lorsque les données sont réparties dans des classes convexes.

Les *méthodes de partitionnement spectrales*, quant à elles, fonctionnent également lorsque les classes ne sont pas convexes.

Les algorithmes que nous allons voir dans ce chapitre s'appuient tous sur une représentation du nuage de points sous forme de graphe, et reformulent la question de la classification en une question de partitionnement de graphe. Ils diffèrent ensuite sur le mode de construction du graphe et sur la méthode de partitionnement.

Notations et définitions

On dispose d'un ensemble de points $\{x_1, \ldots, x_n\}$ et d'une mesure de similarité entre chaque paire (x_i, x_j). À partir de ces données, on va construire un graphe pondéré et non orienté $G = (V, E)$, où V est l'ensemble des sommets du graphe, c'est-à-dire $V = \{x_1, \ldots, x_n\}$ et E l'ensemble des arêtes du graphe. On appelle G le *graphe de similarité*.

Notations et définitions

On dispose d'un ensemble de points $\{x_1, \ldots, x_n\}$ et d'une mesure de similarité entre chaque paire (x_i, x_j). À partir de ces données, on va construire un graphe pondéré et non orienté $G = (V, E)$, où V est l'ensemble des sommets du graphe, c'est-à-dire $V = \{x_1, \ldots, x_n\}$ et E l'ensemble des arêtes du graphe. On appelle G le *graphe de similarité*.

Matrice d'adjacence. Le fait que G soit pondéré signifie qu'à chaque arête entre deux sommets x_i et x_j, on associe un poids positif $w_{ij} \geq 0$. On note alors $W = (w_{ij})_{i=1,\ldots,n;j=1,\ldots,n}$ la *matrice d'adjacence* du graphe. En particulier, un poids nul entre deux sommets signifie qu'ils ne sont pas reliés par une arête. La matrice W est symétrique car le graphe G est non orienté, ce qui implique que $w_{ij} = w_{ji}$.

Notations et définitions

Degré d'un sommet. Dans un graphe non pondéré, le degré d'un sommet v est simplement le nombre d'arêtes dont l'une des extrémités est égale à v. Dans un graphe pondéré, on définit le degré du sommet x_i par

$$d_i = \sum_{j=1}^{n} w_{ij}$$

On note alors $D = \text{diag}\{d_1, \ldots, d_n\}$ la matrice diagonale contenant les degrés des sommets sur sa diagonale.

Graphe connexe. Un sous-ensemble $A \subset V$ est dit connexe s'il est possible de relier n'importe quelle paire de sommets de A à l'aide d'un chemin qui ne passe que par des sommets de A.

Notations et définitions (suite)

Composante connexe. Un sous-ensemble $A \subset V$ est une composante connexe du graphe G s'il est connexe, et si il n'existe aucune connexion entre A et sont complémentaire $V \setminus A$.

Partition. La suite $A_1, \ldots, A_K$ forme une partition de G si et seulement si quelques soient i et j, $A_i \cap A_j = \emptyset$ et $A_1 \cup \ldots \cup A_K = V$.

Mesure de similarité

Les mesures de similarité évoquées précédemment peuvent être utilisées. De plus, il existe d'autres mesure spécifiques aux algorithmes de partitionnement spectral comme :

- La fonction de similarité Gaussienne

$$s_{ij} = \exp(-\frac{d^2(x_i, x_j)}{2\sigma^2})$$

où $d^2(x_i, x_j)$ est une distance et σ un paramètre d'échelle à définir. Pus σ est grand, plus la similarité entre points diminue vite avec leur distance.

Mesure de similarité

Les mesures de similarité évoquées précédemment peuvent être utilisées. De plus, il existe d'autres mesure spécifiques aux algorithmes de partitionnement spectral comme :

- La fonction de similarité Gaussienne

$$s_{ij} = \exp(-\frac{d^2(x_i, x_j)}{2\sigma^2})$$

où $d^2(x_i, x_j)$ est une distance et σ un paramètre d'échelle à définir. Pus σ est grand, plus la similarité entre points diminue vite avec leur distance.

- La fonction de similarité cosinus

$$s_{ij} = \frac{< x_i, x_j >}{||x_i||||x_j||}.$$

Construction du graphe

Une fois que l'on dispose d'une mesure de similarité, il existe plusieurs façons de construire le graphe de similarité.

Construction du graphe

Une fois que l'on dispose d'une mesure de similarité, il existe plusieurs façons de construire le graphe de similarité.

Graphe entièrement connecté On relie entre eux tous les points dont la mesure de similarité est positive, et on pondère l'arête reliant x_i et x_j par la mesure de similarité s_{ij} entre ces points. En général, on utilise cette construction lorsque la mesure de similarité choisie reflète elle-même la similarité *locale* entre les points (*i.e.*, similarité Gaussienne).

Construction du graphe

Une fois que l'on dispose d'une mesure de similarité, il existe plusieurs façons de construire le graphe de similarité.

Graphe entièrement connecté On relie entre eux tous les points dont la mesure de similarité est positive, et on pondère l'arête reliant x_i et x_j par la mesure de similarité s_{ij} entre ces points. En général, on utilise cette construction lorsque la mesure de similarité choisie reflète elle-même la similarité *locale* entre les points (*i.e.*, similarité Gaussienne).

Construction du graphe (suite)

Graphe des k-plus proches voisins. On ne relie un sommet x_i qu'avec ses k plus proches voisins. La notion de k-voisinage n'étant pas symétrique, cette définition conduit à la construction d'un graphe orienté. On obtient alors un graphe non orienté en procédant de l'une des deux façons suivantes :

Construction du graphe (suite)

Graphe des k-plus proches voisins. On ne relie un sommet x_i qu'avec ses k plus proches voisins. La notion de k-voisinage n'étant pas symétrique, cette définition conduit à la construction d'un graphe orienté. On obtient alors un graphe non orienté en procédant de l'une des deux façons suivantes :

- on relie x_i et x_j si x_i appartient aux k plus proches voisins de x_j, ou si x_j appartient aux k plus proches voisins de x_i. On appelle ce graphe le graphe des k-plus proches voisins.
- on relie x_i et x_j s'ils appartiennent tous les deux aux k plus proches voisins de l'autre point. On appelle ce graphe le graphe des k plus proches voisins mutuels.

Construction du graphe (suite)

Graphe des k-plus proches voisins. On ne relie un sommet x_i qu'avec ses k plus proches voisins. La notion de k-voisinage n'étant pas symétrique, cette définition conduit à la construction d'un graphe orienté. On obtient alors un graphe non orienté en procédant de l'une des deux façons suivantes :

- on relie x_i et x_j si x_i appartient aux k plus proches voisins de x_j, ou si x_j appartient aux k plus proches voisins de x_i. On appelle ce graphe le graphe des k-plus proches voisins.
- on relie x_i et x_j s'ils appartiennent tous les deux aux k plus proches voisins de l'autre point. On appelle ce graphe le graphe des k plus proches voisins mutuels.

Graphe de ε-voisinage On relie entre eux tous les points dont la distance mutuelle est inférieure au seuil ε. On obtient alors un graphe binaire (les poids sont égaux à 1 si les sommets sont connectés et 0 sinon).

Laplacien du graphe.

Les méthodes de partitionnement spectrales s'appuient sur le spectre de la matrice de similarité du graphe, et plus précisément sur les valeurs et vecteurs propres de la matrice Laplacienne du graphe G. Il existe plusieurs définitions, dans la littérature, pour la matrice Laplacienne d'un graphe G pondéré, parmi lesquelles :

- Laplacien *non normalisé* : $L = D - W$.

Laplacien du graphe.

Les méthodes de partitionnement spectrales s'appuient sur le spectre de la matrice de similarité du graphe, et plus précisément sur les valeurs et vecteurs propres de la matrice Laplacienne du graphe G. Il existe plusieurs définitions, dans la littérature, pour la matrice Laplacienne d'un graphe G pondéré, parmi lesquelles :

- Laplacien *non normalisé* : $L = D - W$.

- Laplacien *normalisé* : $\tilde{L} = I - D^{-1}W$, ou la version symétrique $\tilde{L}_{\text{sym}} = I - D^{-1/2}WD^{-1/2}$.

Laplacien du graphe.

Les méthodes de partitionnement spectrales s'appuient sur le spectre de la matrice de similarité du graphe, et plus précisément sur les valeurs et vecteurs propres de la matrice Laplacienne du graphe G. Il existe plusieurs définitions, dans la littérature, pour la matrice Laplacienne d'un graphe G pondéré, parmi lesquelles :

- Laplacien *non normalisé* : $L = D - W$.

- Laplacien *normalisé* : $\tilde{L} = I - D^{-1}W$, ou la version symétrique $\tilde{L}_{\text{sym}} = I - D^{-1/2}WD^{-1/2}$.

Laplacien du graphe (suite)

Proposition. La matrice L vérifie les propriétés suivantes :

- $$q^\top L q = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} (q_i - q_j)^2, \ \forall q \in \mathbb{R}^n.$$
- L est symétrique et semi-définie positive.
- La plus petite valeur propre de L est 0, et le vecteur constant dont toutes les coordonnées sont égales à 1 est un vecteur propre associé à la valeur propre 0.
- L a n valeurs propres positives avec $0 = \lambda_1 \leq \ldots \leq \lambda_n$.

Propriété de la matrice L

Preuve. Montrons que

$$q^\top L q = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(q_i - q_j)^2,\ \forall q \in \mathbb{R}^n.$$

$$\begin{aligned}
q^\top L q &= q^\top D q - q^\top W q \\
&= \sum_{i=1}^{n} d_i q_i^2 - \sum_{i=1}^{n}\sum_{j=1}^{n} q_i q_j w_{ij} \\
&= \frac{1}{2}\left(\sum_{i=1}^{n} d_i q_i^2 - \sum_{j=1}^{n} d_j q_j^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n} q_i q_j w_{ij}\right) \\
&= \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} q_i^2 - \sum_{j=1}^{n}\sum_{i=1}^{n} w_{ji} q_j^2 - 2\sum_{i=1}^{n}\sum_{j=1}^{n} q_i q_j w_{ij}\right) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(q_i - q_j)^2
\end{aligned}$$

Propriété de la matrice L

Preuve. Montrons que L est symmétrique et semi-définie positive. D et W étant symétriques, L l'est aussi.

De plus, comme

$$q^\top Lq = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(q_i - q_j)^2,\ \forall q \in \mathbb{R}^n,$$

on a que L est également semi-définie positive car les w_{ij} sont positifs ou nuls.

Proposition. Soit G un graphe non orienté pondéré tel que $w_{ij} \geq 0$, $i = 1, \ldots, n; j = 1, \ldots, n$. Alors la multiplicité k de la valeur propre 0 pour la matrice L est égale au nombre de composantes connexes dans le graphe.

Preuve. Admis.

Algorithmes de partitionnement

Dans la pratique, il est rare que la graphe soit composé d'exactement K composantes connexes.

Algorithmes de partitionnement

Dans la pratique, il est rare que la graphe soit composé d'exactement K composantes connexes.

L'objectif des algorithmes de partitionnement est de "couper" des arêtes afin de créer de composantes connexe.

Algorithmes de partitionnement

Dans la pratique, il est rare que la graphe soit composé d'exactement K composantes connexes.

L'objectif des algorithmes de partitionnement est de "couper" des arêtes afin de créer de composantes connexe.

Si le graphe contient bien K composantes connexes, la matrice L est diagonale par blocs (en ordonnant les individus par classes), avec K blocs.

Algorithmes de partitionnement

Dans la pratique, il est rare que la graphe soit composé d'exactement K composantes connexes.

L'objectif des algorithmes de partitionnement est de "couper" des arêtes afin de créer de composantes connexe.

Si le graphe contient bien K composantes connexes, la matrice L est diagonale par blocs (en ordonnant les individus par classes), avec K blocs.

La matrice contenant les vecteurs propres est alors elle-même diagonale par blocs.

Algorithmes de partitionnement

Dans la pratique, il est rare que la graphe soit composé d'exactement K composantes connexes.

L'objectif des algorithmes de partitionnement est de "couper" des arêtes afin de créer de composantes connexe.

Si le graphe contient bien K composantes connexes, la matrice L est diagonale par blocs (en ordonnant les individus par classes), avec K blocs.

La matrice contenant les vecteurs propres est alors elle-même diagonale par blocs.

Il s'agit alors d'identifier ces blocs, en appliquant par exemple un algorithme des K-means à cette matrice.

Partitionnement spectral non normalisé

Données : Un n-échantillon $(x_1, \ldots, x_n)$ et une matrice de similarité associée S

Résultat : Une partitions $\mathcal{P}$ en K classes

Initialisation : Construire un graphe de similarité de matrice d'adjacence W

1 Calculer la matrice Laplacienne L ou $\tilde{L}$;

2 Calculer les vecteurs propres associés aux K plus petites valeurs propres de la matrice Laplacienne, et définir la matrice U contenant les k vecteurs propres en colonnes

$$U = (v_1 \mid v_2 \mid \ldots \mid v_K)$$

Réaliser une classification des lignes $u1, \ldots, u_n$ de la matrice U à l'aide d'un algorithme des K-means pour obtenir les classes $C_1, \ldots, C_K$;

3 Définir les classes $A_1, \ldots, A_K$ par

$$A_k = \{j \mid u_j \in C_k\}$$

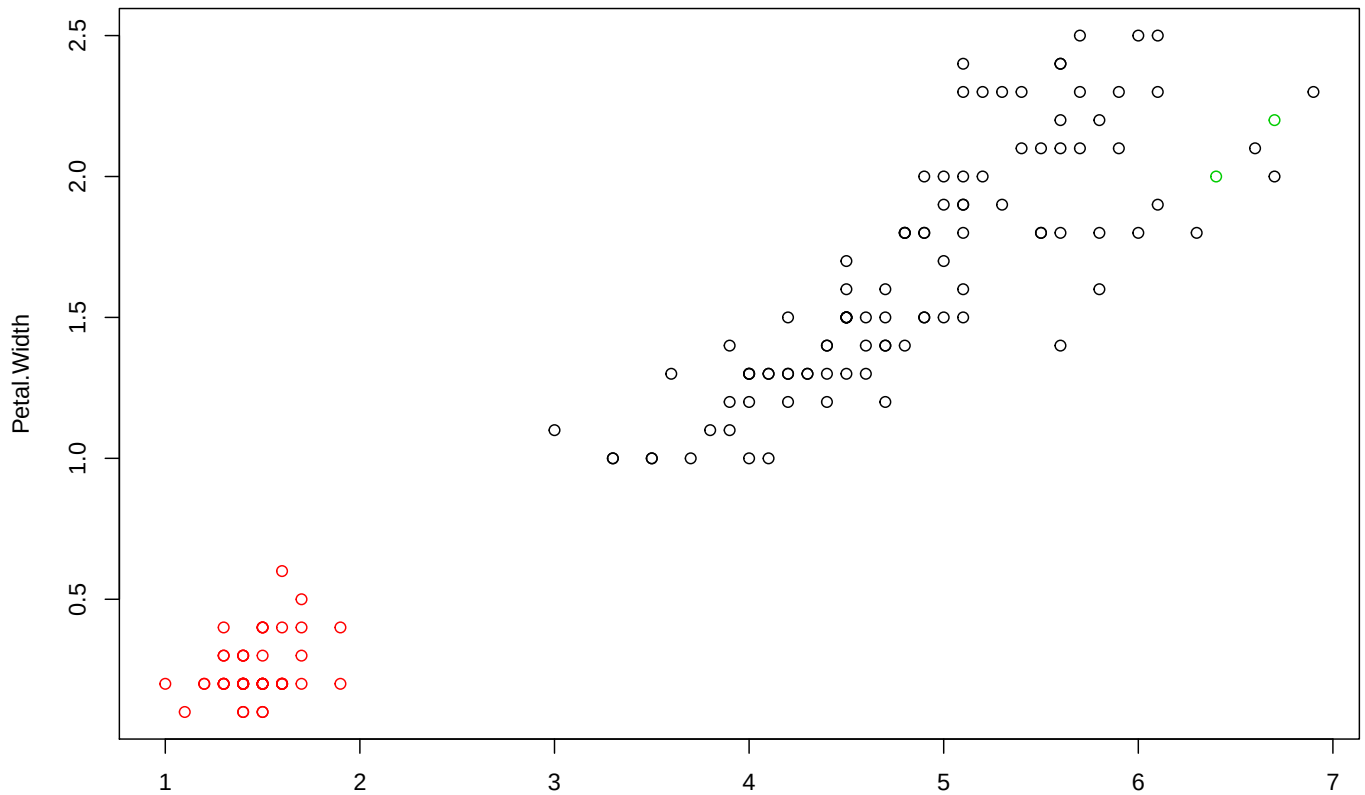
Algorithme 3 : Partitionnement spectral non normalisé

Clustering spectral sous R

```
require(kernlab)
```

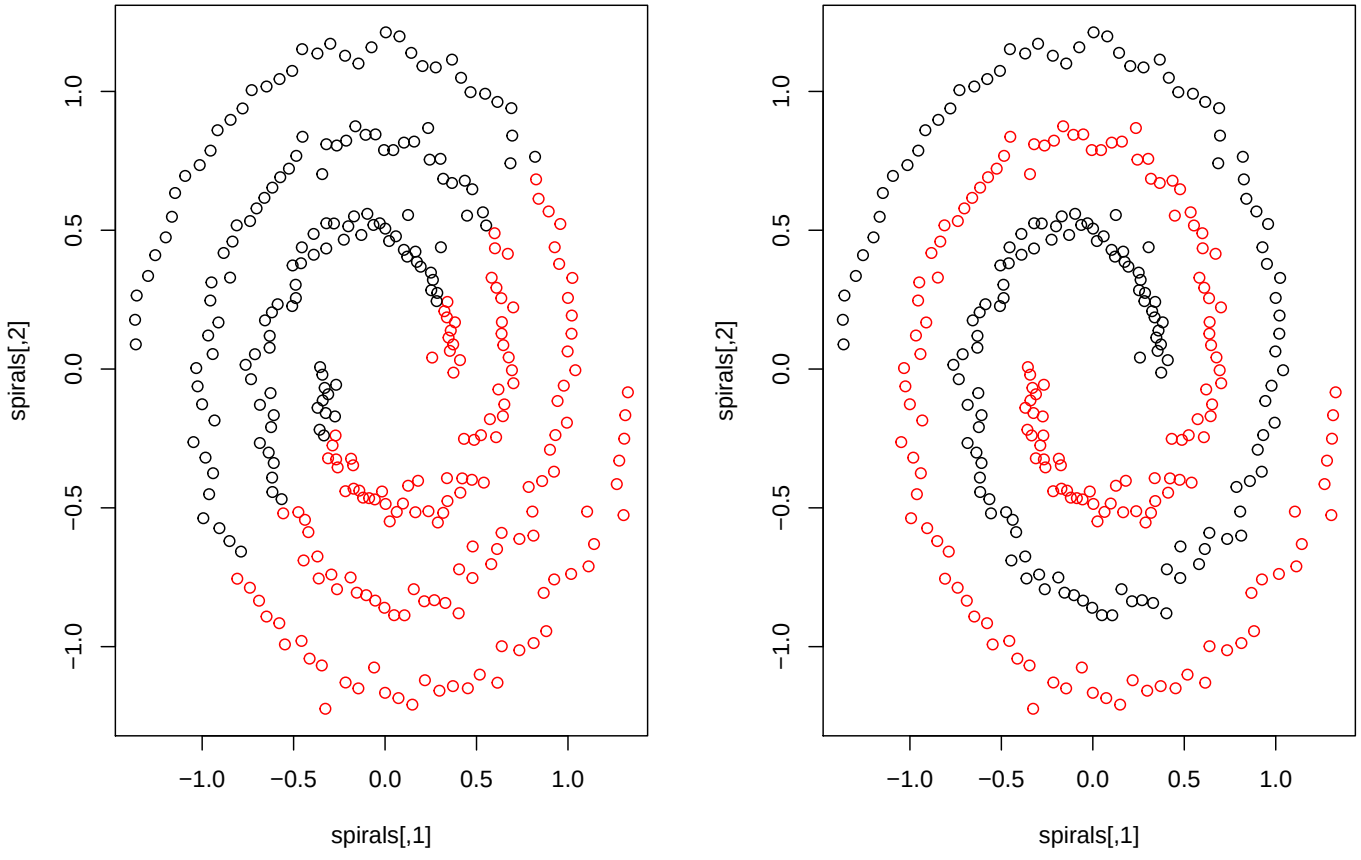
```
Loading required package: kernlab
```

```
clus.spec <- specc(as.matrix(iris[,1:4]), centers = 3)
plot(iris[,3:4], col=clus.spec)
```



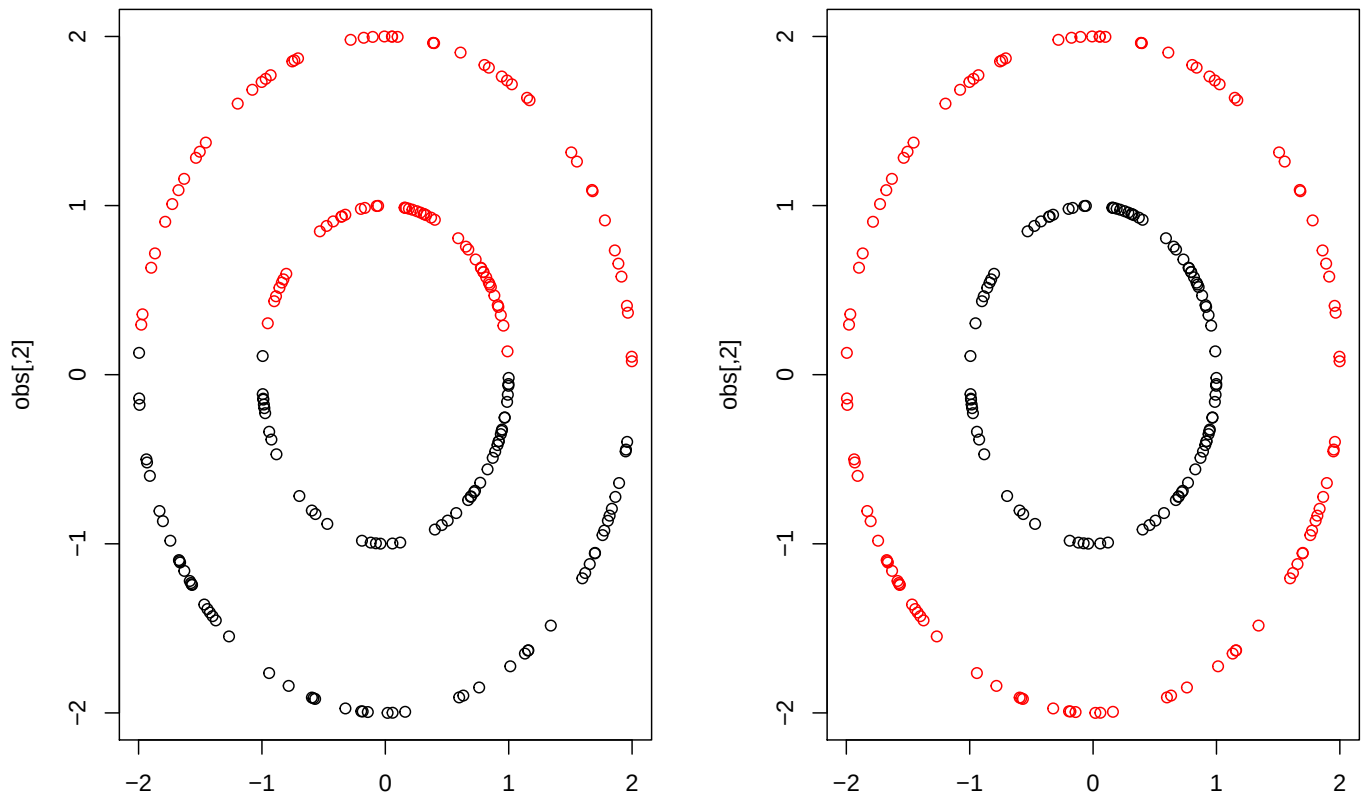
Clustering spectral sous R : spirals

```
data(spirals)
par(mfrow=c(1,2))
plot(spirals, col = kmeans(spirals, 2)$cluster)
plot(spirals, col = specc(spirals, 2))
```



Clustering spectral sous R : circles

```
theta <- runif(200, max = 2*pi)
rho <- 1 + sample(0:1, 200, replace = TRUE)
obs <- cbind(rho * cos(theta), rho*sin(theta))
par(mfrow=c(1,2))
plot(obs, col = kmeans(obs, 2)$cluster)
plot(obs, col = specc(obs, 2))
```



DBSCAN

Principe

Le principe de DBSCAN (Density-based spatial clustering of applications with noise) repose sur la notion de ε-voisinage d'un individu ou point défini comme l'ensemble des points appartenant à la boule de rayon ε centrée en ce point.

Principe

Le principe de DBSCAN (Density-based spatial clustering of applications with noise) repose sur la notion de ε-voisinage d'un individu ou point défini comme l'ensemble des points appartenant à la boule de rayon ε centrée en ce point.

En plus du rayon ε, un autre paramètre est considéré : *MinPts* qui précise un nombre minimum de points à prendre en compte dans cette boule.

Principe

L'ensemble des points se répartit en trois catégories :

- Un point x_i est un *coeur* si son ε-voisinage contient au moins *MinPts* points en l'incluant. Un point du ε-voisinage de x_i est dit *densément atteignable* depuis x_i. Par construction, aucun point ne peut être densément atteignable par un point qui n'est pas un coeur.

Principe

L'ensemble des points se répartit en trois catégories :

- Un point x_i est un *coeur* si son ε-voisinage contient au moins *MinPts* points en l'incluant. Un point du ε-voisinage de x_i est dit *densément atteignable* depuis x_i. Par construction, aucun point ne peut être densément atteignable par un point qui n'est pas un coeur.

- Un point x_j est dit *atteignable* depuis x_i s'il existe un chemin $x_i, x_{i_1}, x_{i_2}, ..., x_{i_\ell}, x_j$ de sorte que chaque $x_{i_{h+1}}$ est densément atteignable depuis x_{i_h}. Sous entendu, tous les points du chemin doivent être des coeurs à l'exception de x_j.

Principe

L'ensemble des points se répartit en trois catégories :

- Un point x_i est un *coeur* si son ε-voisinage contient au moins *MinPts* points en l'incluant. Un point du ε-voisinage de x_i est dit *densément atteignable* depuis x_i. Par construction, aucun point ne peut être densément atteignable par un point qui n'est pas un coeur.

- Un point x_j est dit *atteignable* depuis x_i s'il existe un chemin $x_i, x_{i_1}, x_{i_2}, ..., x_{i_\ell}, x_j$ de sorte que chaque $x_{i_{h+1}}$ est densément atteignable depuis x_{i_h}. Sous entendu, tous les points du chemin doivent être des coeurs à l'exception de x_j.

- Tous les points *non atteignables* de tout autre point, donc isolés, sont considérés comme atypiques (*outliers*).

Principe

Ainsi, si x_i est un coeur, il forme une classes avec tous les points (coeur ou pas) qui sont atteignables à partir de lui.

Principe

Ainsi, si x_i est un coeur, il forme une classes avec tous les points (coeur ou pas) qui sont atteignables à partir de lui.

Chaque classe contient au moins un coeur.

Principe

Ainsi, si x_i est un coeur, il forme une classes avec tous les points (coeur ou pas) qui sont atteignables à partir de lui.

Chaque classe contient au moins un coeur.

Des points qui ne sont pas de coeurs peuvent appartenir à une classe mais seulement en étant à sa frontière car ils ne peuvent servir à atteindre d'autres points.

Principe

Ainsi, si x_i est un coeur, il forme une classes avec tous les points (coeur ou pas) qui sont atteignables à partir de lui.

Chaque classe contient au moins un coeur.

Des points qui ne sont pas de coeurs peuvent appartenir à une classe mais seulement en étant à sa frontière car ils ne peuvent servir à atteindre d'autres points.

L'*atteignabilité* n'est pas une relation symétrique car un point qui n'est pas un coeur peut être atteint mais aucun point ne peut être atteint à partir de lui.

Principe

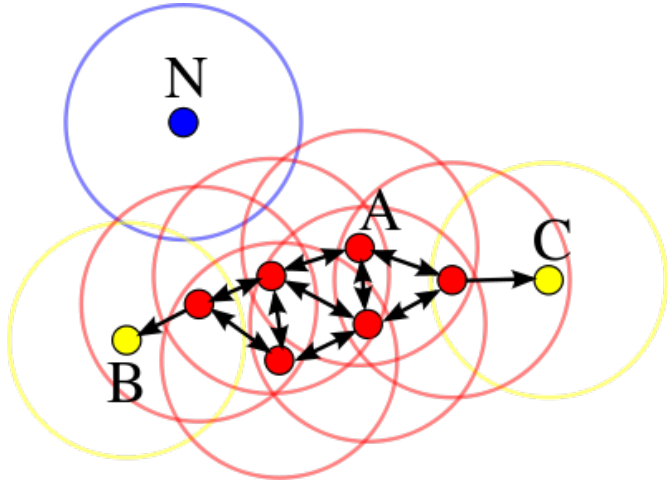


FIGURE 2 – Les points A sont les points déjà dans le cluster. Les points B et C sont atteignables depuis A et appartiennent donc au même cluster. Le point N est une donnée aberrante puisque son epsilon voisinage ne contient pas MinPts points ou plus.

Algorithme

DBSCAN nécessite de fixer les valeurs de deux paramètres : ε et *minPts*.

Algorithme

DBSCAN nécessite de fixer les valeurs de deux paramètres : ε et *minPts*.

L'algorithme démarre d'un point arbitrairement fixé qui n'a pas encore été visité. Si le ε-voisinage de ce point contient suffisamment de points, la construction d'une classe démarre. Sinon ce point est, au moins temporairement, étiqueté comme atypique mais peut être reconsidéré s'il apparaît par la suite dans le ε-voisinage du coeur d'une autre classe.

Algorithme

DBSCAN nécessite de fixer les valeurs de deux paramètres : ε et *minPts*.

L'algorithme démarre d'un point arbitrairement fixé qui n'a pas encore été visité. Si le ε-voisinage de ce point contient suffisamment de points, la construction d'une classe démarre. Sinon ce point est, au moins temporairement, étiqueté comme atypique mais peut être reconsidéré s'il apparaît par la suite dans le ε-voisinage du coeur d'une autre classe.

Si un point est un coeur, tous les points de son ε-voisinage sont affectés à sa classe puis chacun considéré par l'algorithme. Si l'un de ces points est un coeur son ε-voisinage vient compléter la classe et le processus continue jusqu'à la complétion de la classe des points atteignables.

Algorithme

DBSCAN nécessite de fixer les valeurs de deux paramètres : ε et *minPts*.

L'algorithme démarre d'un point arbitrairement fixé qui n'a pas encore été visité. Si le ε-voisinage de ce point contient suffisamment de points, la construction d'une classe démarre. Sinon ce point est, au moins temporairement, étiqueté comme atypique mais peut être reconsidéré s'il apparaît par la suite dans le ε-voisinage du coeur d'une autre classe.

Si un point est un coeur, tous les points de son ε-voisinage sont affectés à sa classe puis chacun considéré par l'algorithme. Si l'un de ces points est un coeur son ε-voisinage vient compléter la classe et le processus continue jusqu'à la complétion de la classe des points atteignables.

Puis l'algorithme considère un autre point pas encore visiter pour renouveler le procédé jusqu'à ce que tous les points soient étiquetés.

Algorithme

Données : Un n-échantillon $\mathbf{x} = (x_1, \ldots, x_n)$, ε et $MinPts$
Initialisation : $C = 0$

```
1 pour chaque point x_i non visité des données x marquer x_i
   comme visité;
2 V = epsilonVoisinage(x, x_i, ε);
3 si tailleDe(V) < MinPts alors
4 |   marquer x_i comme OUTLIERS ;
5 sinon
6 |   C++;
7 |   etendreCluster(x, x_i, V, C, ε, MinPts) ;
8 fin
```

Algorithme 4 : DBSCAN

Algorithme (suite)

Données : $\mathbf{x} = (x_1, \ldots, x_n)$, x_i, C, ε et *MinPts*

1 ajouter x_i au cluster C ;
2 pour chaque point x_j de V **si** x_j *n'a pas été visité* **alors**
3 | marquer x_j comme visité ;
4 | $V' =$ epsilonVoisinage($\mathbf{x}$, x_j, ε);
5 | **si** *tailleDe*(V') >= *MinPts* **alors**
6 | | $V = V \cup V'$;
7 **si** x_j *n'est membre d'aucun cluster* **alors**
8 | ajouter x_j au cluster C;

Algorithme 5 : etendreCluster

1 retourner tous les points de $\mathbf{x}$ qui sont à une distance inférieure à ε de x_i

Algorithme 6 : epsilonVoisinage

DBSCAN sous R

```
require(dbscan)
## find suitable eps parameter using a k-NN plot for k = di
## Look for the knee!
kNNdistplot(as.matrix(iris[,1:4]), k = 5)
abline(h=.5, col = "red", lty=2)
```

DBSCAN sous R

```
res <- dbscan(as.matrix(iris[,1:4]), eps = .5, minPts = 5)
res
```

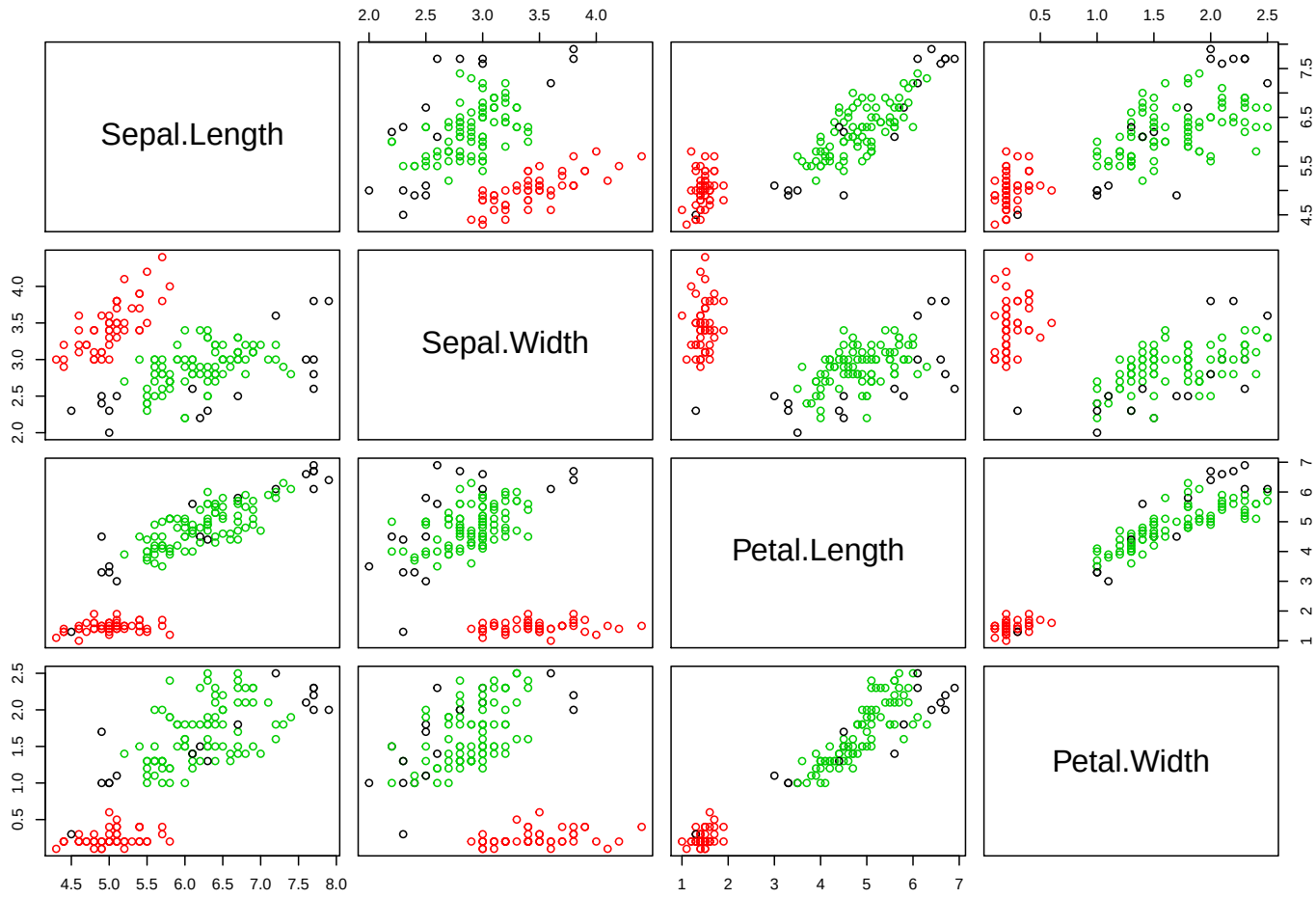
```
DBSCAN clustering for 150 objects.
Parameters: eps = 0.5, minPts = 5
The clustering contains 2 cluster(s) and 17 noise points.

 0  1  2
17 49 84

Available fields: cluster, eps, minPts
```

DBSCAN sous R

```
pairs(as.matrix(iris[,1:4]), col = res$cluster + 1L)
```



Propriétés

Avantages :

- DBSCAN n'est pas nécessaire de définir a priori le nombre de classes, celui-ci est une conséquence du choix des paramètres ε et *MinPts*

Propriétés

Avantages :

- DBSCAN n'est pas nécessaire de définir a priori le nombre de classes, celui-ci est une conséquence du choix des paramètres ε et *MinPts*

- DBSCAN peut distinguer des classes avec des formes pathologiques ou même imbriquées entre elles

Propriétés

Avantages :

- DBSCAN n'est pas nécessaire de définir a priori le nombre de classes, celui-ci est une conséquence du choix des paramètres ε et *MinPts*
- DBSCAN peut distinguer des classes avec des formes pathologiques ou même imbriquées entre elles
- DBSCAN est robuste aux observations atypiques et propose même de les détecter.

Propriétés

Avantages :

- DBSCAN n'est pas nécessaire de définir a priori le nombre de classes, celui-ci est une conséquence du choix des paramètres ε et *MinPts*
- DBSCAN peut distinguer des classes avec des formes pathologiques ou même imbriquées entre elles
- DBSCAN est robuste aux observations atypiques et propose même de les détecter.

Propriétés (suite)

Inconvénients :

- Le choix des valeurs de ε et *MinPts* doit être opéré de façon experte à partir de la compréhension des données et de leur environnement. *MinPts* apparaît comme une borne inférieure pour la taille des classes. Pour fixer ε, il est conseillé de tracer le graphe des distances de k plus proches voisins avec *minPts* pour valeur de k.

Propriétés (suite)

Inconvénients :

- Le choix des valeurs de ε et *MinPts* doit être opéré de façon experte à partir de la compréhension des données et de leur environnement. *MinPts* apparaît comme une borne inférieure pour la taille des classes. Pour fixer ε, il est conseillé de tracer le graphe des distances de k plus proches voisins avec *minPts* pour valeur de k.

- DBSCAN est mis en défaut si les classes présentent des densités locales hétérogènes et il ne met pas en évidence des classes si des groupes de points ne sont pas suffisamment distincts les uns des autres.

Conclusion

Conclusion

Une méthode de classification non supervisée produit une variable qualitative dont les modalités précisent la classe retenue pour chaque individu.

Conclusion

Une méthode de classification non supervisée produit une variable qualitative dont les modalités précisent la classe retenue pour chaque individu.

Chaque méthode, algorithme, chaque choix de critère, dont le nombre de classes conduit à des solutions différentes.

Conclusion

Une méthode de classification non supervisée produit une variable qualitative dont les modalités précisent la classe retenue pour chaque individu.

Chaque méthode, algorithme, chaque choix de critère, dont le nombre de classes conduit à des solutions différentes.

Quelques difficultés :

- Évaluation de la qualité de la partition
- Choix du nombre de classe
- Gestion de données complexes (catégorielles, mixtes, avec valeurs manquantes)
- Évaluation du risque d'erreur de classification
- Critère pour choisir la méthode optimale

Évaluation de la qualité de la partition

L'indice **silhouette** d'une classification permet est un graphe montrant comment chaque observation appartient plus ou moins à sa classe.

Supposons que n observations aient été réparties en K classes par un algorithme.

Soit $a(i)$ la moyenne des dissimilarités (ou distances) de l'observation i avec toutes les autres observations au sein d'une même classe. Plus $a(i)$ est petit meilleur est l'assignation de i à sa classe ; $a(i)$ est la dissimilarité moyenne de i à cette classe.

Soit $b(i)$ la plus faible moyenne des dissimilarités (ou distances) de l'observation i à chaque autre classe dont i ne fait pas partie. La classe avec cette plus faible dissimilarité moyenne est appelé classe *voisine* de i car c'est la meilleur classe suivante pour l'observation i.

La **silhouette**

Définition. La silhouette de la $i$ème observation est alors donnée par

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Plus ces valeurs sont proches de 1 et meilleure est la classification. La moyenne de ces valeurs est un indicateur global de qualité du clustering.

Évaluation de la qualité de la partition

```
require(cluster)
res <- kmeans(iris[,1:4], 3)
si <- silhouette(res$cluster,
                 daisy(iris[,1:4], metric = "euclidean"))
plot(si)
```

Choix du nombre de classes

Le choix du nombre de classes K est, comme le choix de la dimension en ACP, délicat à opérer.

Plusieurs heuristiques ont été proposées selon les critères précédents ou encore suivant le graphe de décroissance de la distance inter-classes qui est aussi la décroissance de la variance inter-classe dans le cas du saut de Ward. La recherche d'un "coude" dans ce graphe est une indication heuristique du choix de K.