

Chapitre 5-3) Méthodes des K-means et DBSCAN

Maxime El Masri

3 MIC / INSA Toulouse

2023-2024

Introduction

- Données : On observe n individus décrits par p variables

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix} \text{ avec } x_i = (x_{i1}, \dots, x_{ip}) \in \mathcal{X}$$

- Soit d une distance euclidienne
- But : trouver une partition de l'ensemble des individus telle que l'inertie intra-classe soit minimale
- Rappel : Inertie intra-classe ($K \in \mathbb{N}$ fixé) :

$$I_{intra} = \sum_{k=1}^K \sum_{i \in \mathcal{C}_k} d(x_i, m_k)^2$$

Partie 3

1 Méthodes de type K-means

2 Extensions des K-means

3 Choix des noyaux initiaux

4 Choix du nombre de classes

5 DBSCAN

Principe général des méthodes de type K-means

- Initialisation :
 - ▶ choix du nombre de classes K
 - ▶ choix de K noyaux initiaux $c_1^{(0)}, \dots, c_K^{(0)}$
- On itère les deux étapes suivantes : à l'itération t
 - ▶ Affectation des individus à la classe la plus proche :
$$i \in \mathcal{C}_k^{(t)} \text{ si } d\left(x_i, c_k^{(t-1)}\right) = \min_{1 \leq k' \leq K} d\left(x_i, c_{k'}^{(t-1)}\right)$$
 - ▶ Mise à jour des noyaux : $c_1^{(t)}, \dots, c_K^{(t)}$
- Arrêt de l'algorithme quand la classification n'est plus modifiée

Méthode des centres mobiles [@forgy1965]

- $c_1^{(0)}, \dots, c_K^{(0)}$ sont choisis au hasard (sans remise) dans \mathcal{X}
- A l'itération t :
 - ▶ $i \in \mathcal{C}_k^{(t)}$ si $d\left(x_i, c_k^{(t-1)}\right) = \min_{1 \leq k' \leq K} d\left(x_i, c_{k'}^{(t-1)}\right)$
 - ▶ Mise à jour des noyaux :
- Arrêt de l'algorithme quand 2 itérations successives fournissent la même classification

$$\forall k \in \{1, \dots, K\}, \quad c_k^{(t)} = \frac{1}{|\mathcal{C}_k^{(t)}|} \sum_{i \in \mathcal{C}_k^{(t)}} x_i$$

Méthode des centres mobiles [@forgy1965]

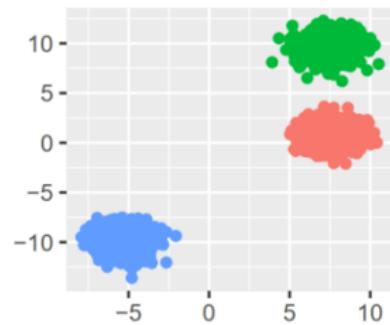
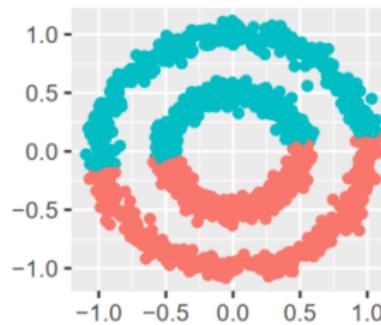
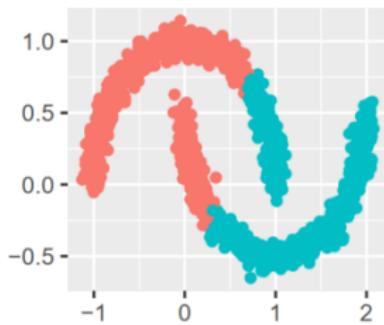
Points forts / faibles

- Avantages :

- ▶ Relativement efficace (rapide)
- ▶ Tend à réduire l'inertie intra-classe à chaque itération
- ▶ Forme des classes compactes, bien séparées

- Inconvénients :

- ▶ Influence du choix des noyaux initiaux
- ▶ Convergence vers un minimum local
- ▶ Nécessite la notion de centre de gravité
- ▶ Influence des outliers
- ▶ Non adapté pour des classes non convexes



Propriétés

Proposition:

L'inertie intraclasse $I_{\text{intra}}(\mathcal{P}_K^{(t)})$ décroît à chaque étape.

⇒ convergence vers un minimum local de l'inertie intra-classe.

Preuve:

Les deux points clés sont :

- si i passe de $\mathcal{C}_k^{(t-1)}$ à $\mathcal{C}_{k'}^{(t)}$ alors

$$d(x_i, c_{k'}^{(t-1)}) \leq d(x_i, c_k^{(t-1)})$$

- $c_k^{(t)}$ étant le centre de $\mathcal{C}_k^{(t)}$

$$\sum_{i \in \mathcal{C}_k^{(t)}} d(x_i, c_k^{(t)})^2 \leq \sum_{i \in \mathcal{C}_k^{(t-1)}} d(x_i, c_k^{(t-1)})^2$$

Variante : K-means [@mcqueen1967]

- Initialisation :
 - ▶ choix de K
 - ▶ $c_1^{(0)}, \dots, c_K^{(0)}$ sont tirés aléatoirement parmi les n points observés
- A chaque itération t
 - ▶ tirage au hasard d'un nouveau point $x_i \in \mathcal{X}$
 - ▶ détermination du noyau $c_k^{(t-1)}$ le plus proche de x_i
 - ▶ mise à jour du noyau :

$$c_k^{(t)} = \frac{x_i + |\mathcal{C}_k^{(t-1)}| c_k^{(t-1)}}{|\mathcal{C}_k^{(t-1)}| + 1}$$

Nuées dynamiques [@diday1971]

- Initialisation : choix de K sous-ensembles disjoints $N_1^{(0)}, \dots, N_K^{(0)}$ de cardinal q chacun (q fixé)
- A l'itération t :
 - On affecte les points de \mathcal{X}

$$\mathcal{C}_k^{(t)} = \left\{ x \in \mathcal{X}; D\left(x, N_k^{(t-1)}\right) \leq D\left(x, N_r^{(t-1)}\right) \forall r \neq k \right\}$$

avec $D(x, A) = \sum_{y \in A} d(x, y)$, $\forall A \subset \mathcal{X}$, $\forall x \in \mathcal{X}$

- Mise à jour des noyaux $(N_k^{(t)})_{1 \leq k \leq K}$: $N_k^{(t)}$ est le sous-ensemble de cardinal q qui minimise

$$\sum_{x \in N_k^{(t)}} D\left(x, \mathcal{C}_k^{(t)}\right).$$

Commandes R

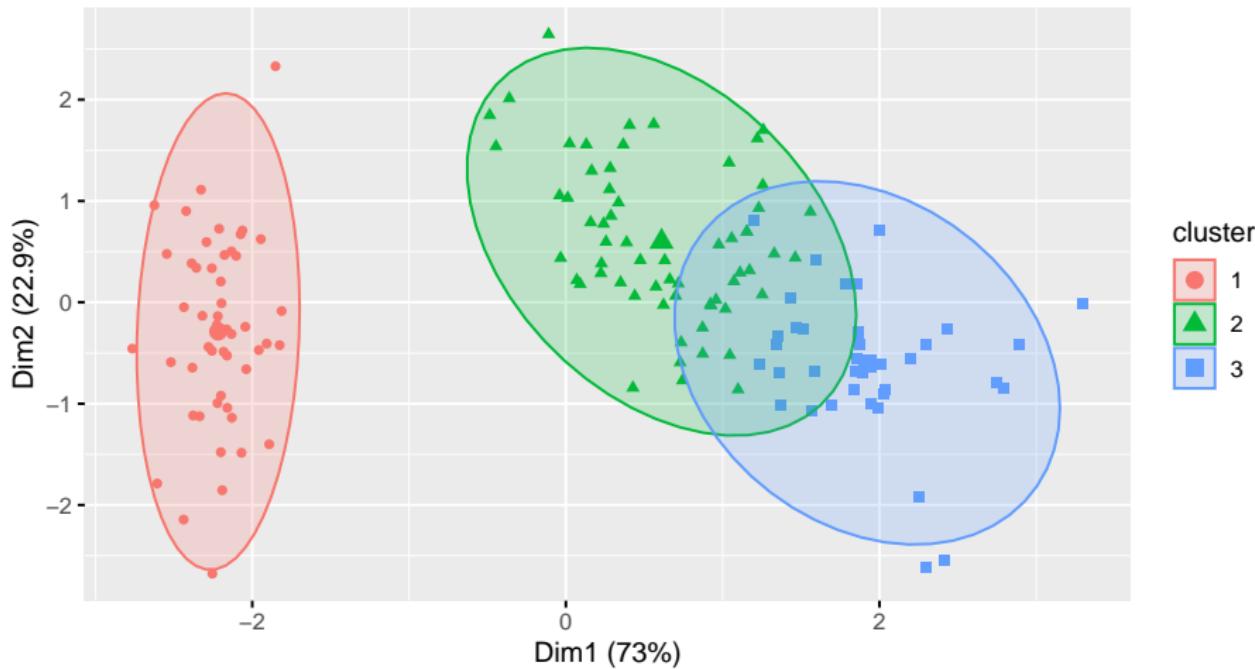
```
km=kmeans(x, centers=, iter.max=, algorithm=)
```

- *centers*: vecteur des noyaux initiaux ou nombre de classes
- *iter.max*: nombre d'itérations maximale
- *algorithm*: “Forgy” (centres mobiles), “MacQueen” (K -means), “Hartigan-Wong” (nuées dynamiques, par défaut)

```
km$cluster, km$centers, km$withinss, km$size
```

Exemple - Données iris

```
data(iris)
kmiris=kmeans(iris[,1:4],centers=3)
fviz_cluster(kmiris,data=iris[,1:4],ellipse.type="norm",labelsize=8,geom=c("point"))+ggtitle("")
```



Partie 3

1 Méthodes de type K-means

2 Extensions des K-means

3 Choix des noyaux initiaux

4 Choix du nombre de classes

5 DBSCAN

Variables qualitatives/mixtes

- Pour variables **qualitatives** :

- ▶ Dans les méthodes de type *K*-means, chaque classe est représentée par son centre de gravité, non adapté pour des variables qualitatives
- ▶ Algorithme ***K*-modes** [Huang, 98] : même principe que l'algorithme des centres mobiles en modifiant la dissimilarité

$$d(x_i, x_\ell) = \sum_{j=1}^p \frac{n_{ij} + n_{\ell j}}{n_{ij} \times n_{\ell j}} \mathbf{1}_{x_{ij} \neq x_{\ell j}}$$

où $n_{ij} = \#\{v \in \{1, \dots, n\}; x_{ij} = x_{vj}\}$. Utilisation des modes à la place des centres de gravité.

- Pour variables **mixtes** : Algorithme ***K*-prototype** [Huang, 98] (mélange des *K*-means et *K*-modes)
- Classification “floue” ou “souple”, pour tout type de variables : **Fuzzy c-means** (chaque individu a une probabilité d’appartenir à chaque classe)

Méthode des K-médoïdes

- Idée : On remplace les centres de gravité par des médoïdes (des points de \mathbf{X} , “représentatifs” des classes)
- Le **médoïde** est l’individu pour lequel la dissimilarité moyenne par rapport aux autres objets de la classe est la plus faible.
- On considère une fonction “coût”, donnée par la moyenne des distances de chaque classe, que l’on cherche à minimiser.

PAM (Partitioning Around Medoids) [@Kaufman]

- Initialisation : choix aléatoire de K médoïdes, $c_1^{(0)}, \dots, c_K^{(0)}$, parmi les n points de \mathcal{X}
- A chaque itération t
 - ▶ on associe chaque point x_i à son plus proche médoïde
 - ▶ pour chaque médoïde $c_k^{(t-1)} \in \mathcal{C}_k^{(t-1)}$: pour chaque point $x_i \in \mathcal{X}$ qui n'est pas un médoïde, on échange x_i avec $c_k^{(t-1)}$ et on calcule le coût $\sum_{j \in \mathcal{C}_k} d(x_j, x_i)$ de la configuration
 - ▶ on sélectionne la configuration minimisant le coût.
 - ▶ on répète ces étapes jusqu'à stabilité de la configuration.

Méthode des K-médoïdes

- Chaque classe est représentée par un individu de la classe donc pas de limitation sur le type de variables prises en compte
- Algorithme efficace pour de petits jeux de données
- PAM est plus robuste que K -means en présence d'outliers (un médoïde est moins influencé par un outlier que le barycentre)

Commandes R

- PAM : `pam(x, k, diss = , metric = , ...)` [library(cluster)]
metric = "euclidian" ou "manhattan", sinon x matrice de dissimilarité
- CLARA (Clustering LARge Application) :
`clara(x, k, metric = ,...)` [library(cluster)]
metric = "euclidian" ou "manhattan", sinon x matrice de dissimilarité

→ fonction à privilégier si grand nombre d'individus

Partie 3

1 Méthodes de type K-means

2 Extensions des K-means

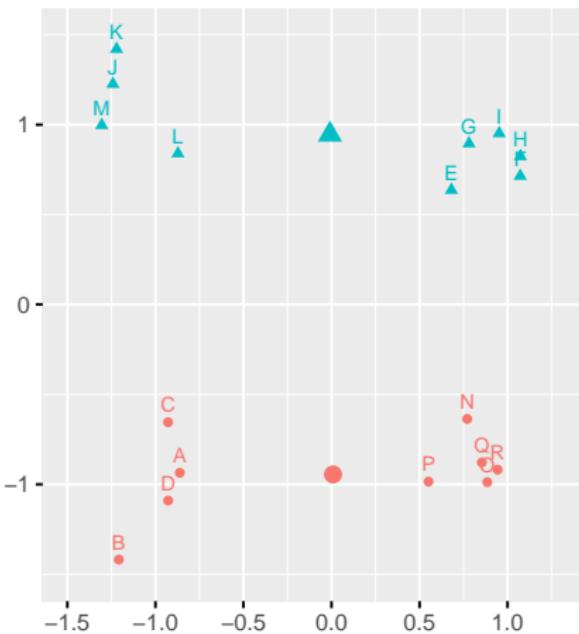
3 Choix des noyaux initiaux

4 Choix du nombre de classes

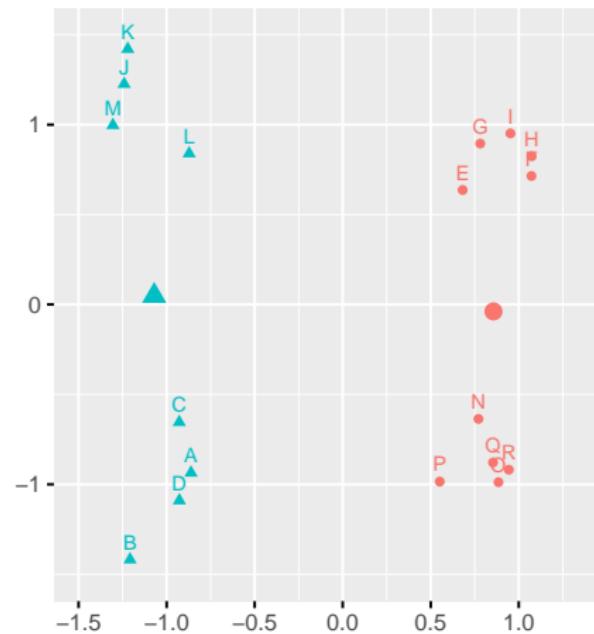
5 DBSCAN

Initialisation des noyaux

Kmeans Init. avec D et L



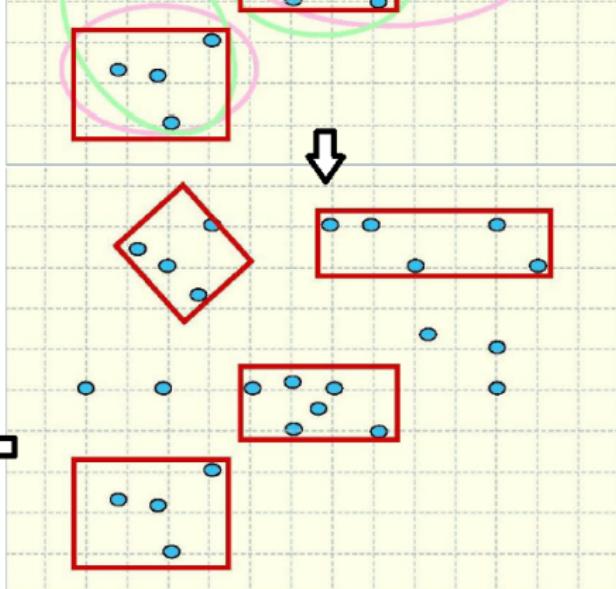
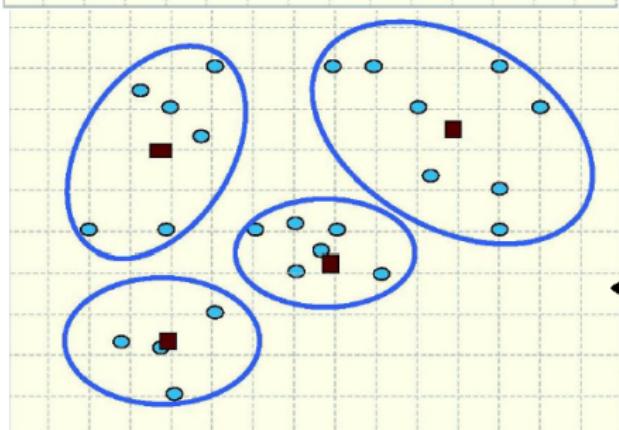
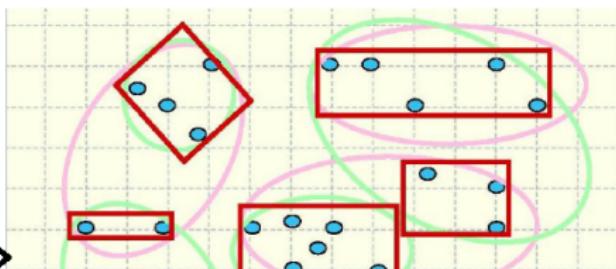
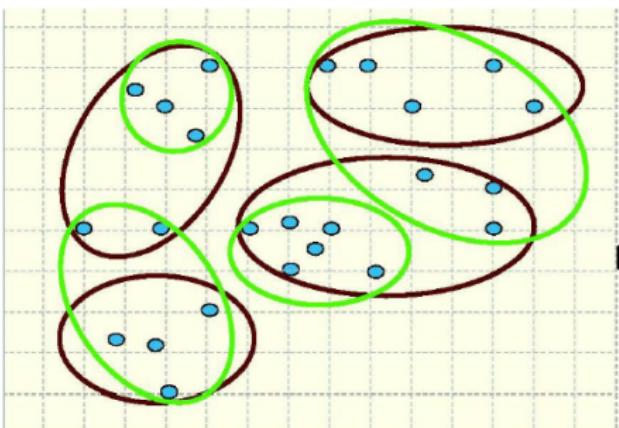
Kmeans Init. avec E et L



Choix des noyaux initiaux

- Sélection fondée sur des connaissances complémentaires
- Etude préliminaire des données
- Répétition de la méthode N fois $\Rightarrow N$ partitions et sélection de la partition ayant la plus faible inertie intra-classe
- Avec les “formes fortes” :
 - ▶ Répétition de la méthode N fois
 - ▶ On regroupe les individus appartenant toujours aux mêmes classes (formes fortes)
 - ▶ Initialisation des noyaux = centres de gravité des meilleures formes fortes

Formes fortes



Partie 3

1 Méthodes de type K-means

2 Extensions des K-means

3 Choix des noyaux initiaux

4 Choix du nombre de classes

5 DBSCAN

Choix de K

- Pour chaque valeur de $K \in \{2, \dots, K_{\max}\}$, on obtient une classification et on sélectionne finalement celle où on observe un saut important de l'inertie intra-classe ("coude").
- Choix par d'autres indices basés sur les inerties intra- et inter-.
- Choix par maximisation d'un indice (silhouette, Gap statistique, ...)

Critères fondés sur les inerties

- **R-Square :**

$$K \mapsto RSQ(K) = 1 - \frac{I_{intra}(\mathcal{P}_K)}{I_{totale}} = \frac{I_{inter}(\mathcal{P}_K)}{I_{totale}}$$

On retient l'endroit où la courbe $K \mapsto RSQ(K)$ forme un “coude”.

- **Semi-Partial R-Square :**

$$K \mapsto SPRSQ(K) = \frac{I_{inter}(\mathcal{P}_K) - I_{inter}(\mathcal{P}_{K-1})}{I_{totale}}$$

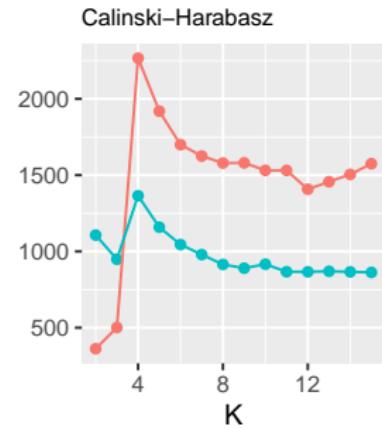
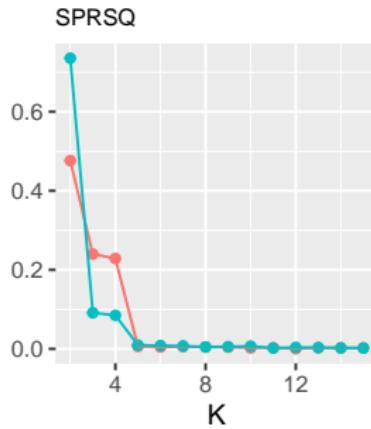
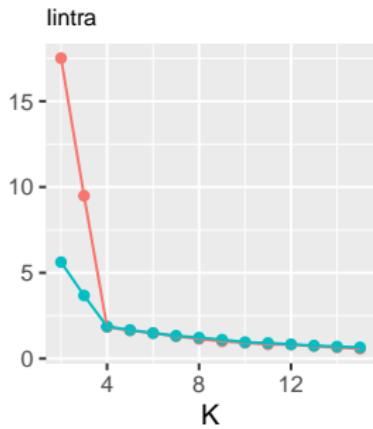
On retient l'endroit où on a la plus forte réduction du SPRSQ.

- **Calinski-Harabasz (CH) :**

$$K \mapsto CH(K) = \frac{I_{inter}(\mathcal{P}_K)/(K-1)}{I_{intra}(\mathcal{P}_K)/(n-K)}$$

On cherche un pic sur cette courbe.

Exemple sur les données simulées



type Data1 Data2

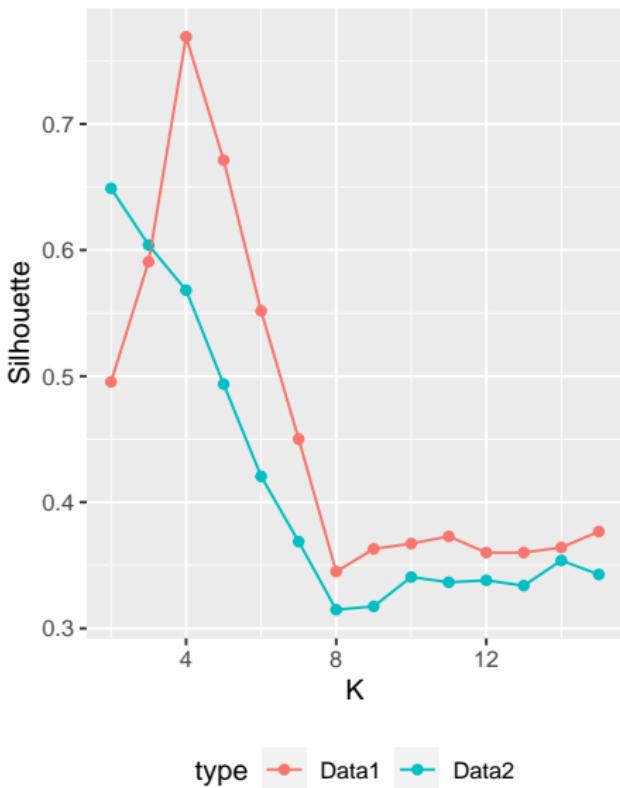
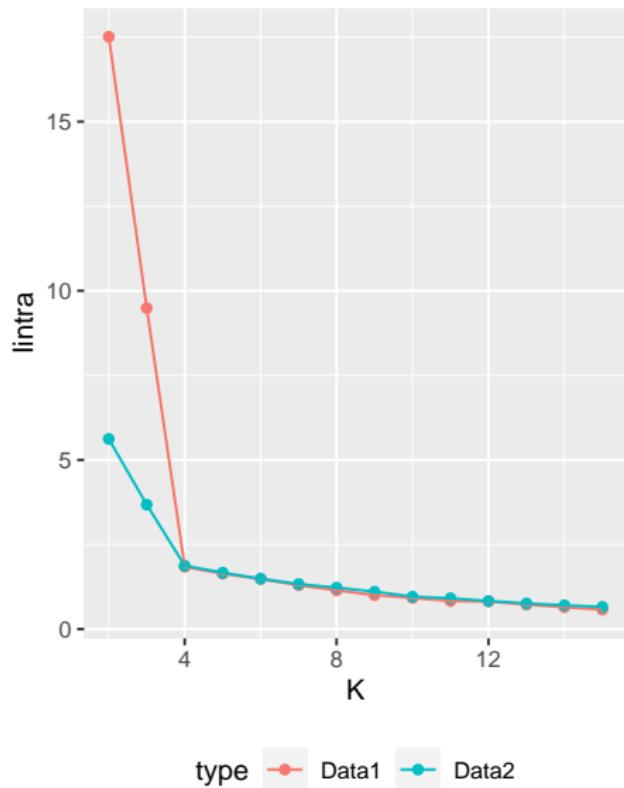
type Data1 Data2

type Data1 Data2

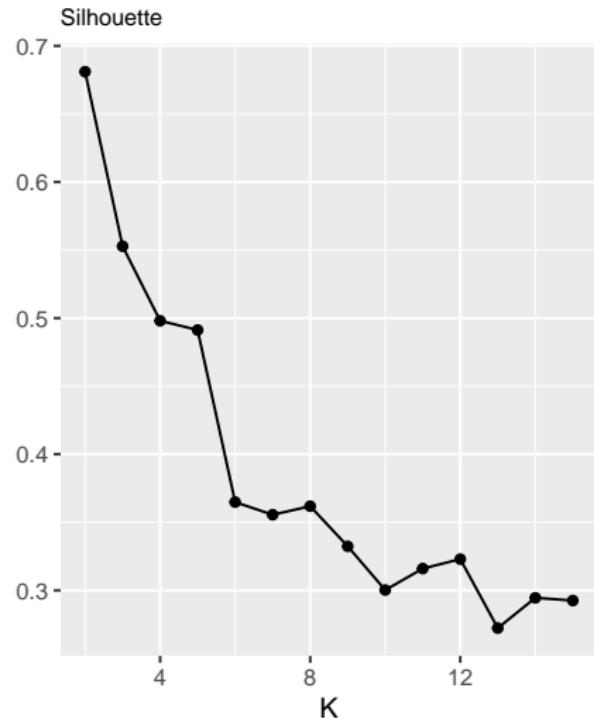
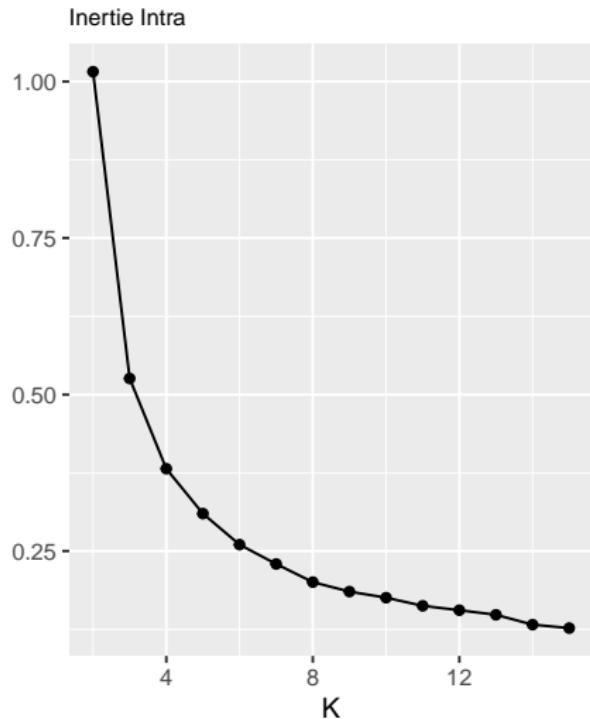
Silhouette

- Pour toute valeur $K \in \{1, \dots, K_{\max}\}$:
 - ▶ on considère $\{\mathcal{C}_1, \dots, \mathcal{C}_K\}$ classification de $\{1, \dots, n\}$
 - ▶ $\forall i \in \{1, \dots, n\}, \exists ! k \in \{1, \dots, K\}$ tq $i \in \mathcal{C}_k$. On calcule :
 - ★ $a(i) = \frac{1}{|\mathcal{C}_k|-1} \sum_{\substack{\ell \in \mathcal{C}_k \\ \ell \neq i}} d(x_i, x_\ell)$ et $b(i) = \min_{k' \neq k} \frac{1}{|\mathcal{C}_{k'}|} \sum_{\ell \in \mathcal{C}_{k'}} d(x_i, x_\ell)$
 - ★ $s(i) = \frac{b(i)-a(i)}{\max(b(i), a(i))} \in [-1, 1]$
 - ▶ $S(K) = \frac{1}{n} \sum_{i=1}^n s(i)$
- Le nombre de classes retenu : $\hat{K} = \underset{1 \leq K \leq K_{\max}}{\operatorname{argmax}} S(K)$
- Commande R : `silhouette(.) [library(cluster)]`

Exemple - Silhouette



Exemple des Iris



Partie 3

1 Méthodes de type K-means

2 Extensions des K-means

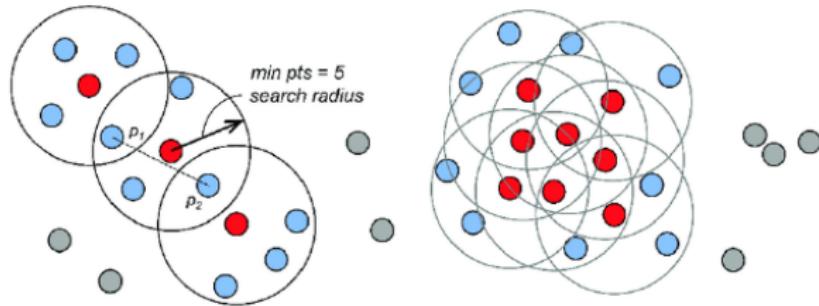
3 Choix des noyaux initiaux

4 Choix du nombre de classes

5 DBSCAN

Principe [@DBSCAN]

- DBSCAN = Density-Based Spatial Clustering of Applications
- 2 paramètres :
 - ▶ la distance de voisinage ε
 - ▶ le nombre minimum de points $minPts$
- Idée : pour un individu donné, vérifier si son ε -voisinage contient au moins $minPts$ points. Si oui, ce point fait partie d'une classe. On parcourt ensuite le ε -voisinage de proche en proche pour trouver l'ensemble des points de la classe.
- Usuellement, la distance euclidienne est utilisée.



Animation

https://aaronscotthq.com/2020-05-28-scott_dbSCAN/

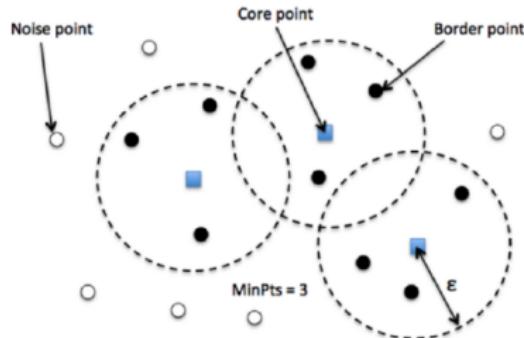
Catégories d'individus

A la fin de l'algorithme, un individu x_i appartient à l'une des 3 catégories :

- Point central : son voisinage est dense

$$\#\{\ell \in \{1, \dots, n\}; d(x_i, x_\ell) < \varepsilon\} \geq \text{minPts}$$

- Point frontière : appartient au ε -voisinage d'un point central mais n'est pas un point central
- Point aberrant : n'est ni un point central, ni un point frontière (donc non classé)



Choix des paramètres

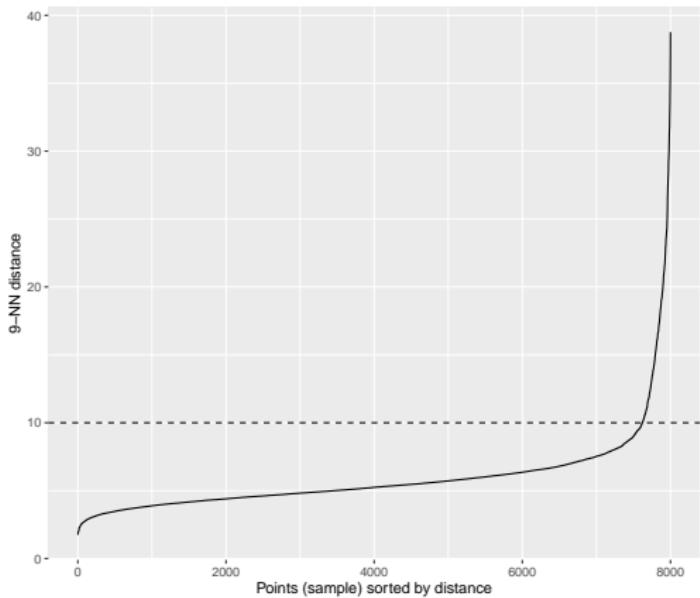
Choix de minPts :

- Il est préconisé de choisir au moins $\text{minPts} \geq p + 1$.
- Pour des données avec du bruit (outliers), $\text{minPts} = 2p$ peut être utilisé mais il peut être nécessaire de choisir des valeurs plus grandes pour des données de grande dimension.
- Dans d'autres sources, $\ln(n)$ est préconisé.

Choix de paramètres

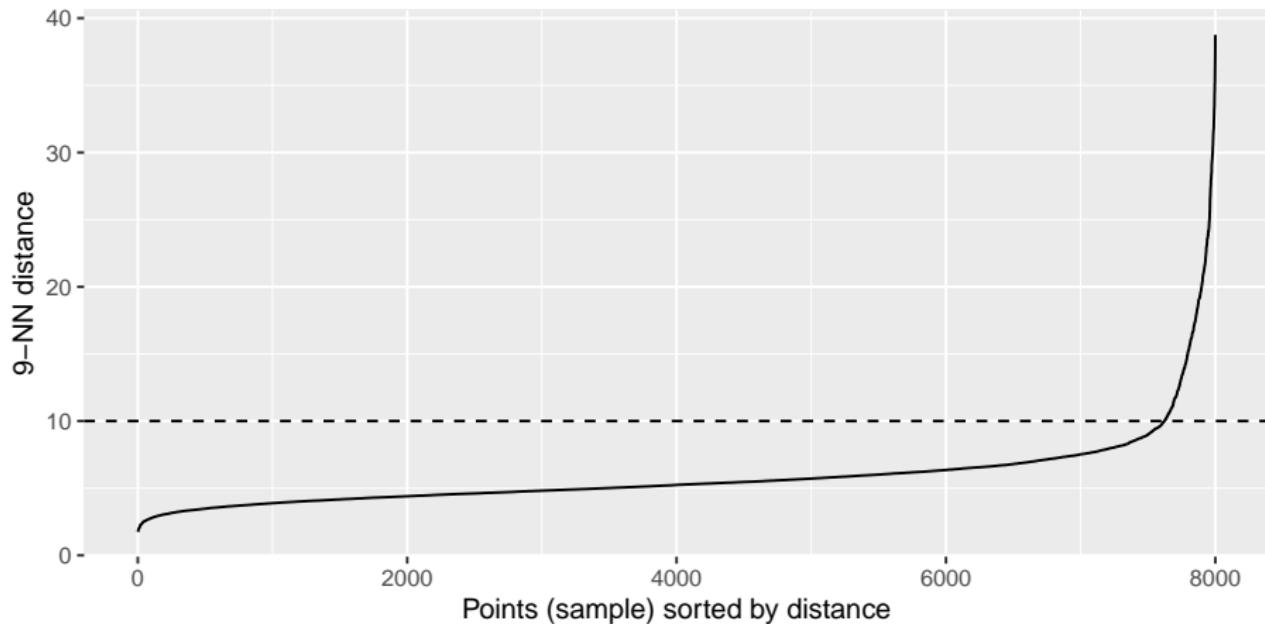
Choix de ε :

- On trace le graphe de distance kNN:
 - Pour chaque point, on calcule sa distance moyenne à ses $k = \min\{n, minPts\} - 1$ plus proches voisins
 - On trace les distances obtenues par ordre croissant
 - On choisit ε comme la valeur de la distance kNN où on observe un “coude”.



Commandes R et exemple

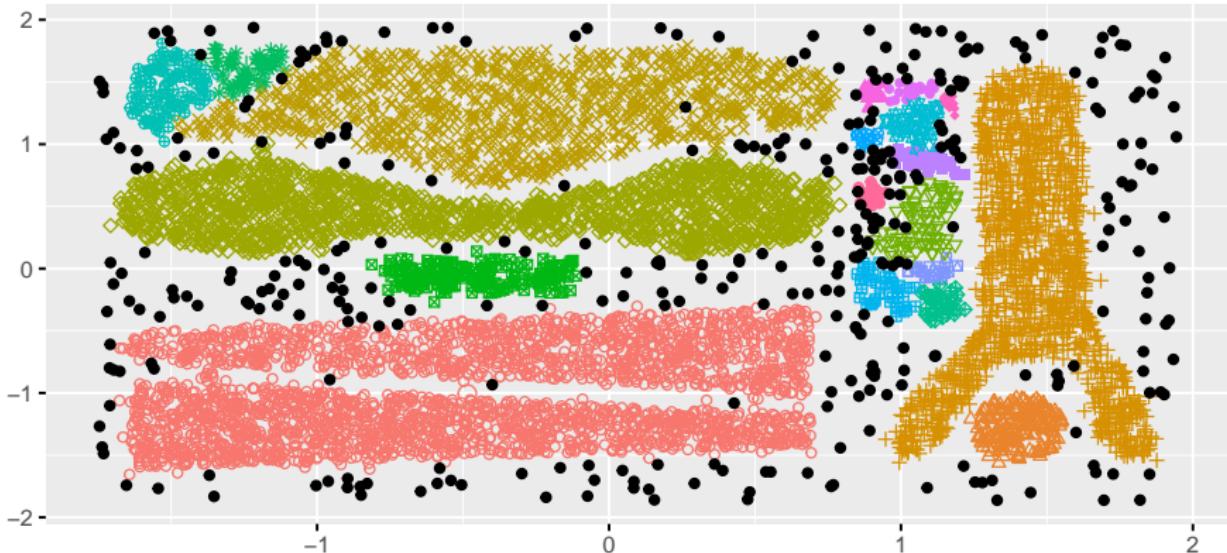
- Fonction `dbSCAN()` de `library(fpc)`
- Dans la `library(dbSCAN)`, les fonctions `dbSCAN()` et `kNNdistplot()`



Exemple

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
357	2588	182	1391	1268	1576	79	177	55	31	120	31	42	9	16	30
16	17	18	19												
15	13	7	13												

DBSCAN



Exemple

Kmeans

