Partitionnement avec l'algorithme DBSCAN

STT-7735 - Méthodes d'analyse de données

Marc-Olivier Ricard

Université Laval

Hiver 2020

Plan de la présentation

Idée générale

- Algorithme de partitionnement proposé par Ester et collab. (1996)

Idée générale

- Algorithme de partitionnement proposé par Ester et collab. (1996)
- **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise

Idée générale

- Algorithme de partitionnement proposé par Ester et collab. (1996)
- **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise
- Classification basée sur la densité

Idée générale

- Algorithme de partitionnement proposé par Ester et collab. (1996)
- **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise
- Classification basée sur la densité
- Regrouper les points qui sont proches les uns des autres en utilisant une mesure de distance

Idée générale

- Algorithme de partitionnement proposé par Ester et collab. (1996)
- **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise
- Classification basée sur la densité
- Regrouper les points qui sont proches les uns des autres en utilisant une mesure de distance
- Identifier les régions les plus denses et les régions les moins denses

Idée générale

- Algorithme de partitionnement proposé par Ester et collab. (1996)
- **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise
- Classification basée sur la densité
- Regrouper les points qui sont proches les uns des autres en utilisant une mesure de distance
- Identifier les régions les plus denses et les régions les moins denses
- Identifier les données aberrantes

Motivations

Développer un algorithme qui combine quatre critères importants :

- Pouvoir choisir les paramètres d'entrée facilement

Motivations

Développer un algorithme qui combine quatre critères importants :

- Pouvoir choisir les paramètres d'entrée facilement
- Pouvoir découvrir des *clusters* de n'importe quelle forme

Motivations

Développer un algorithme qui combine quatre critères importants :

- Pouvoir choisir les paramètres d'entrée facilement
- Pouvoir découvrir des *clusters* de n'importe quelle forme
- Pouvoir être efficace avec beaucoup de données

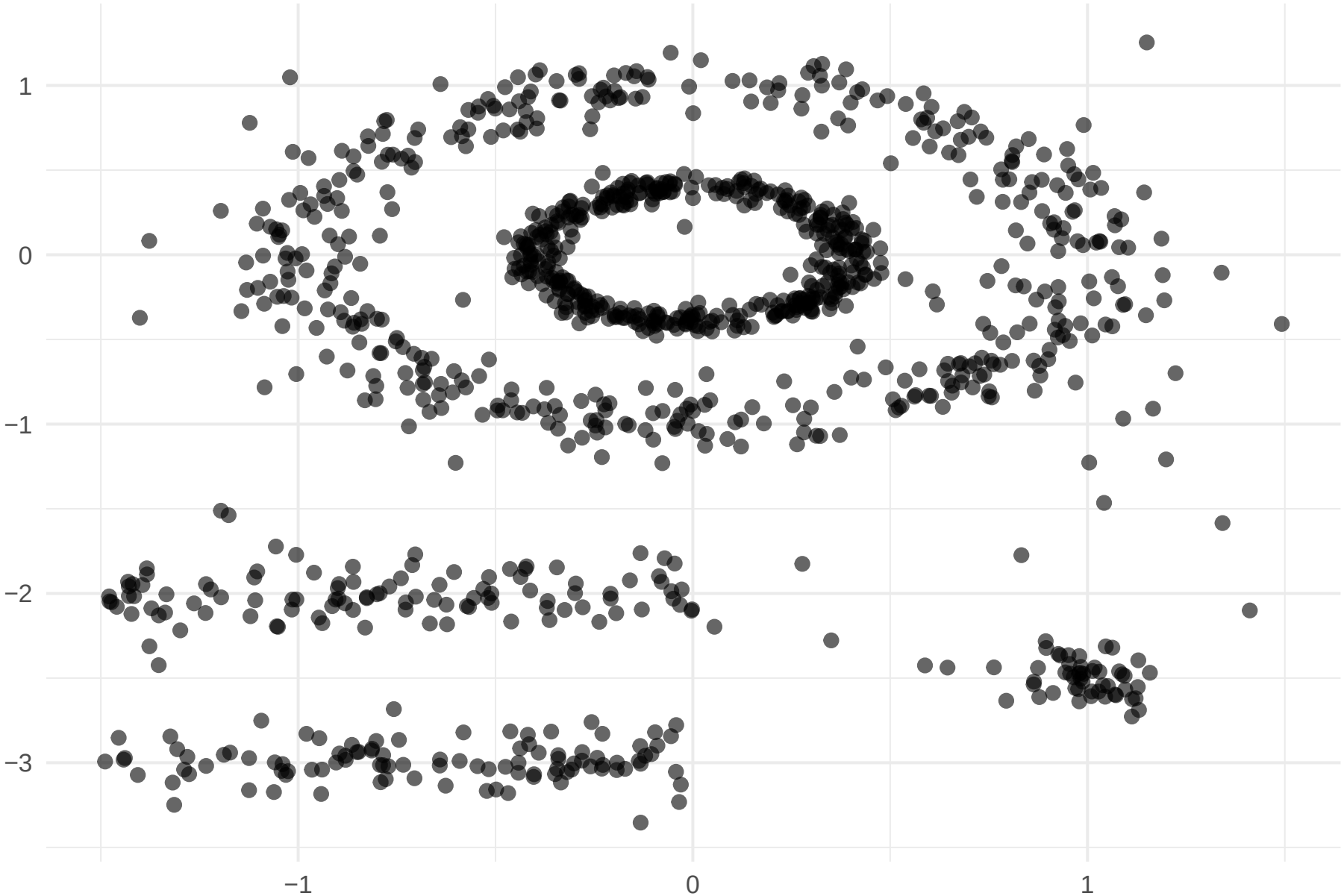
Motivations

Développer un algorithme qui combine quatre critères importants :

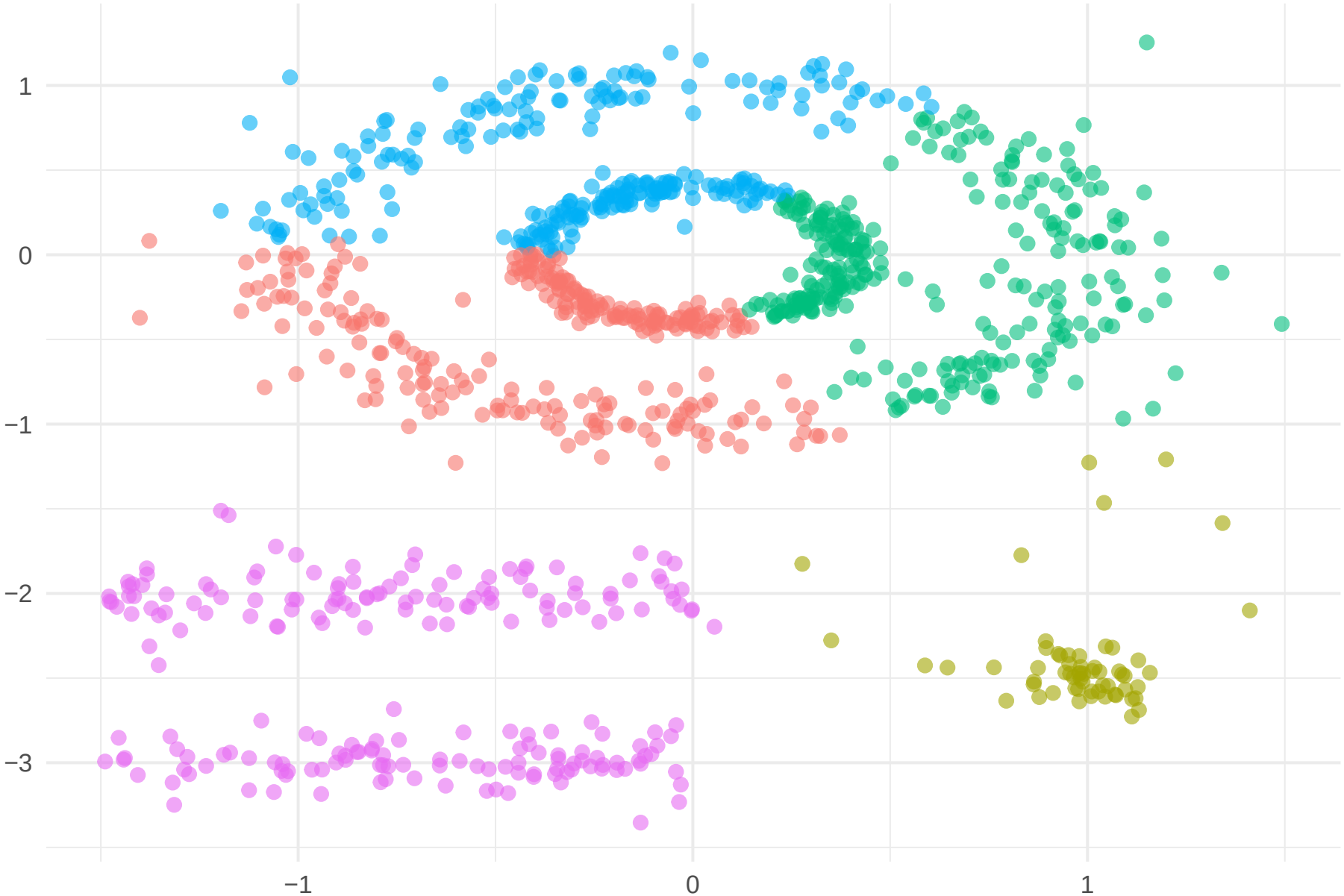
- Pouvoir choisir les paramètres d'entrée facilement
- Pouvoir découvrir des *clusters* de n'importe quelle forme
- Pouvoir être efficace avec beaucoup de données
- Pouvoir être efficace pour des données à plusieurs dimensions

Illustration rapide

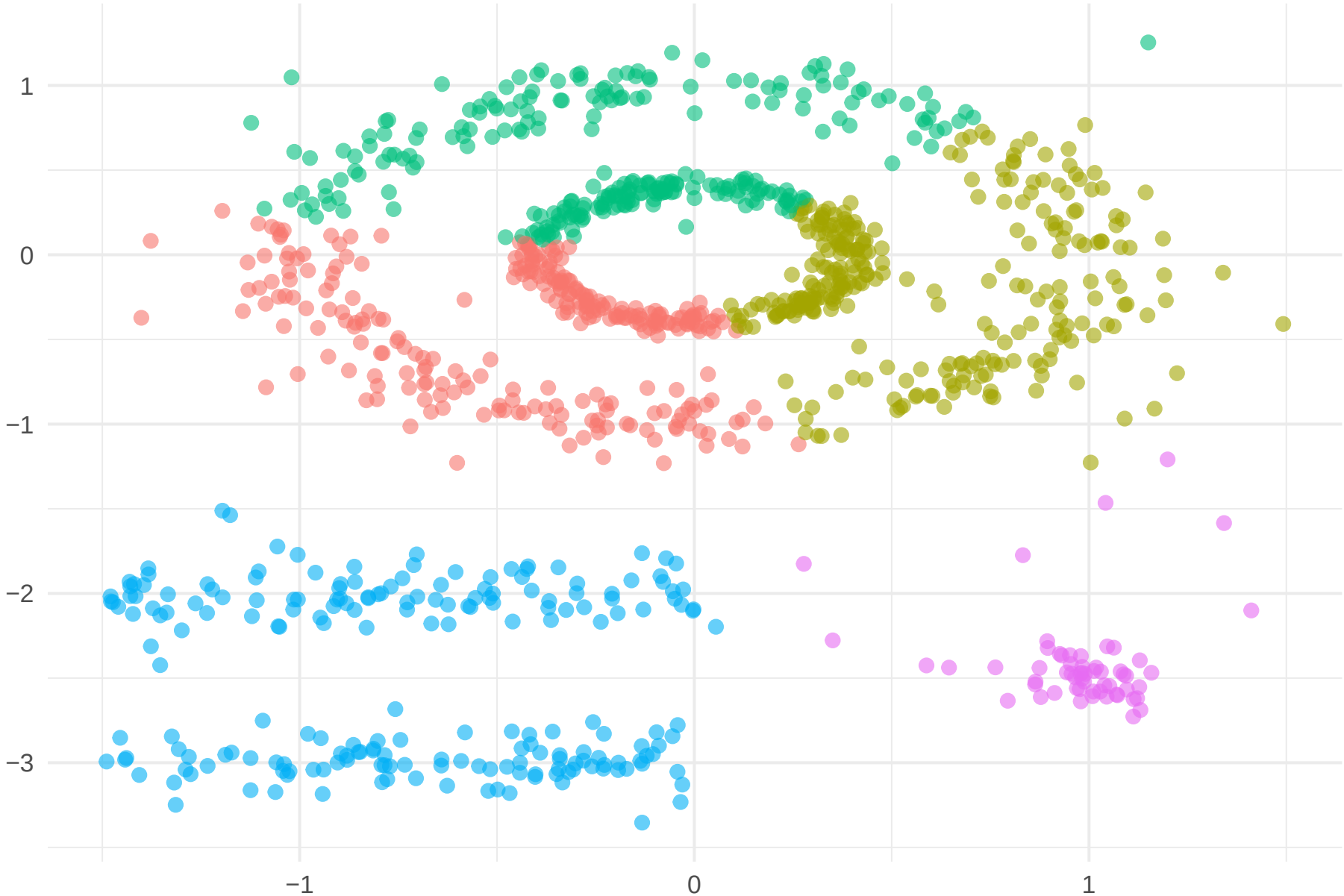
Données fictives



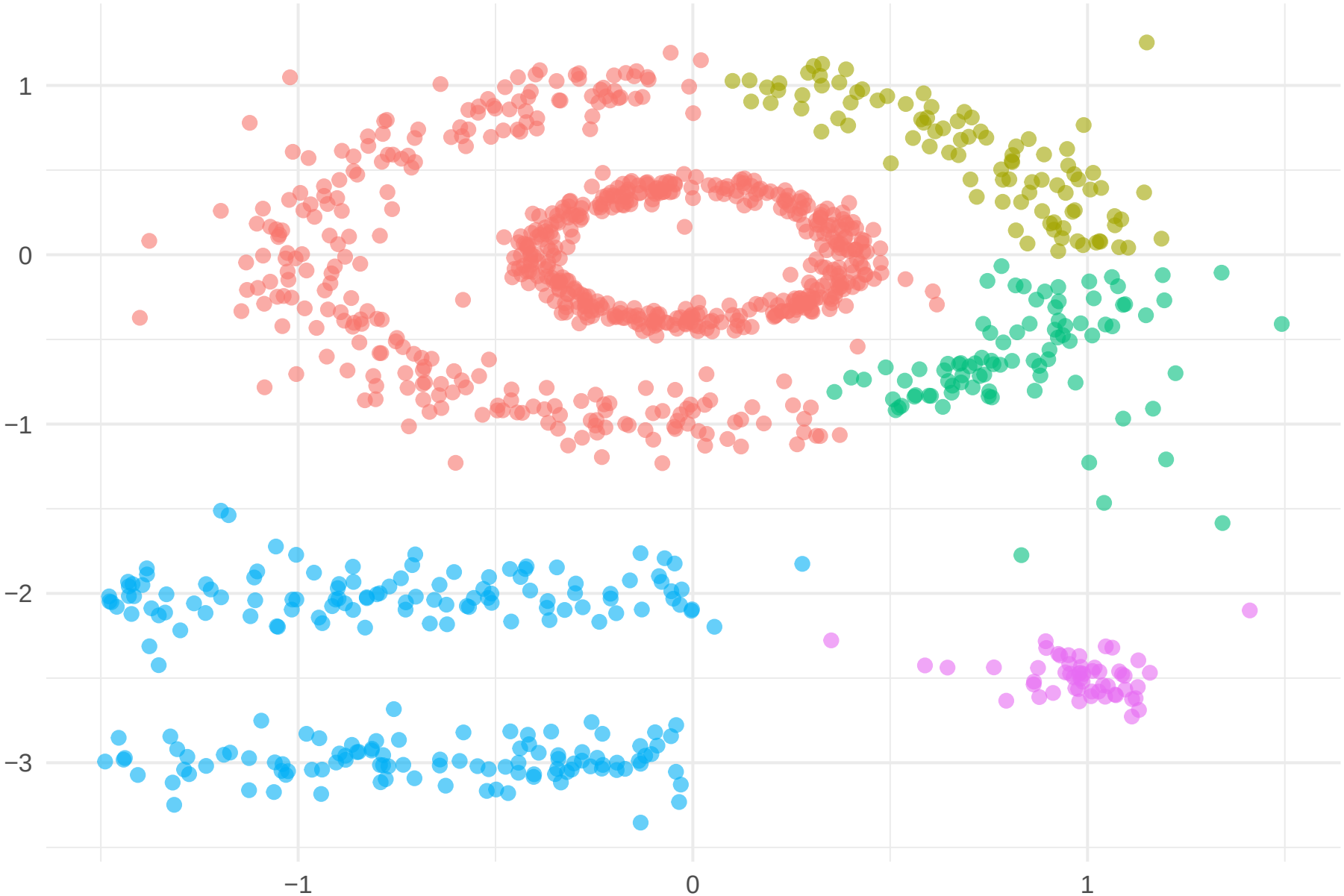
Regroupement algorithme k-moyennes



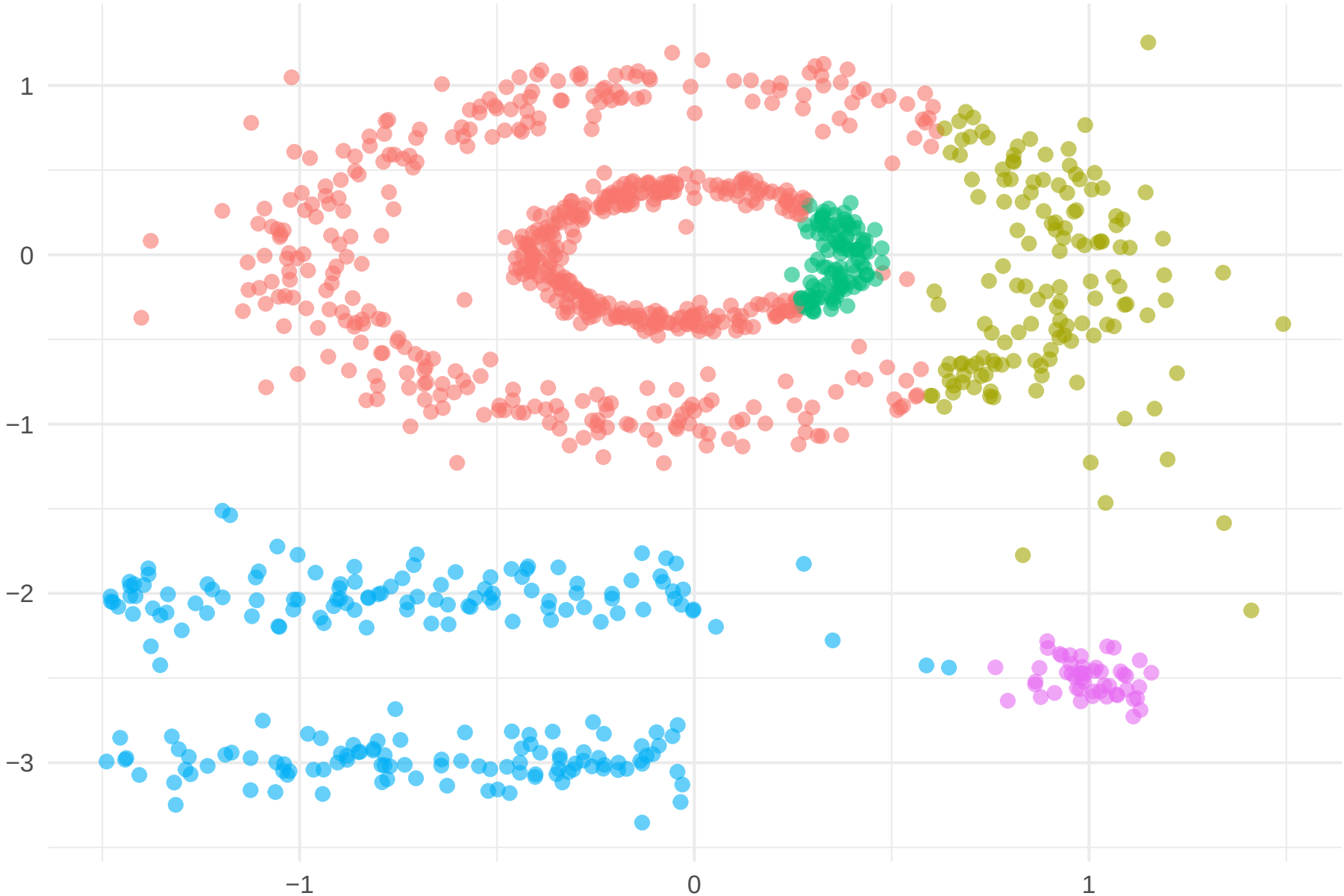
Regroupement algorithme k-médoïdes



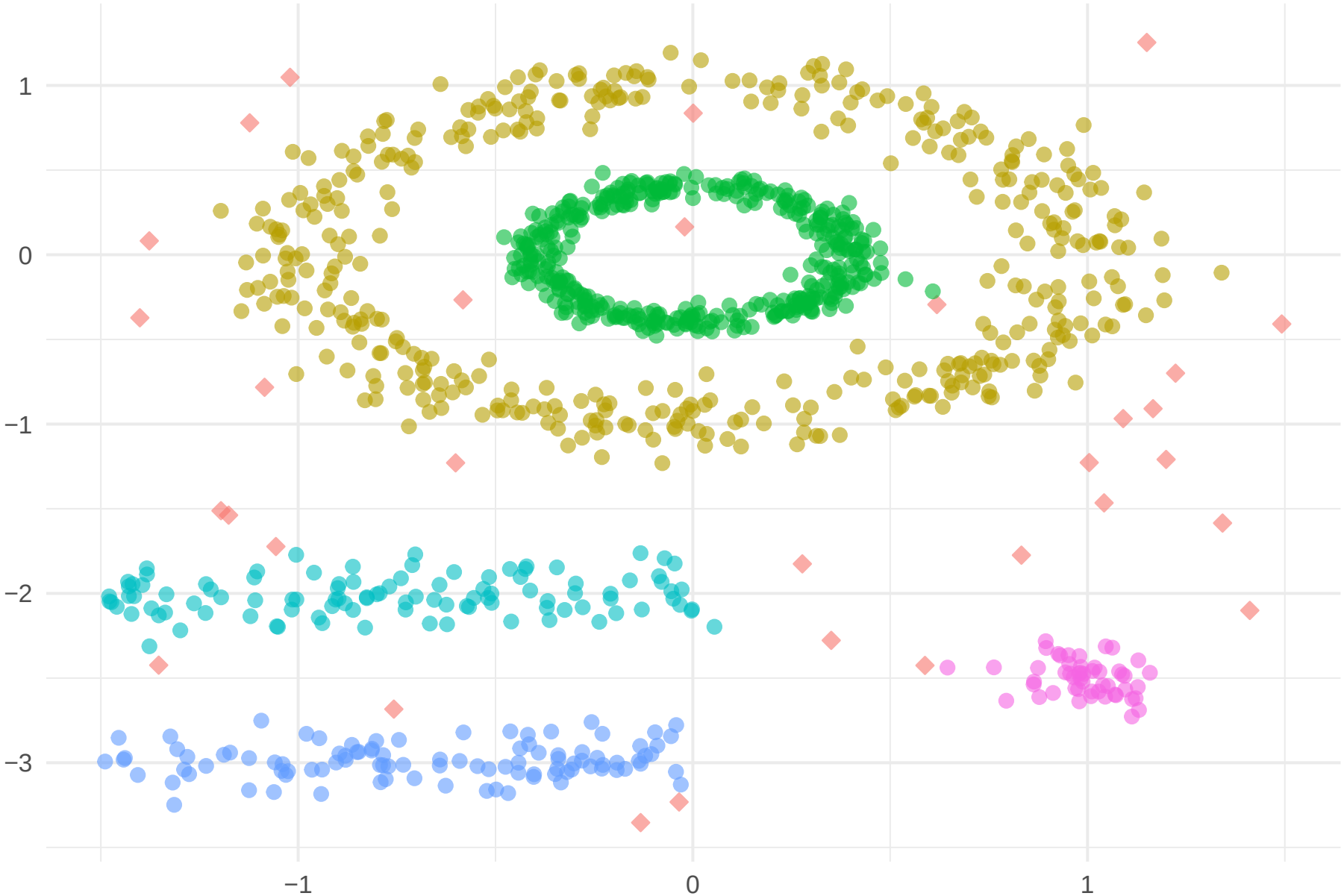
Regroupement algorithme hiérarchique ascendant



Regroupement algorithme modèle de mélange



Regroupement algorithme DBSCAN



Concepts importants utilisés par l'algorithme

Introduction aux paramètres

- Epsilon (ϵ) : si la distance entre deux points est plus petite que ϵ, ceux-ci sont considérés comme étant voisins.
- *Minimum number of points* (*MinPts*) : nombre de points minimal pour former une région dite dense.
- Le choix des valeurs de ces deux paramètres sera couvert dans une autre section.

Voisinage d'un point (*Neighborhood*)

Définition 1

L'ϵ-voisinage d'un point p des données D est défini par

$$N_\epsilon(p) = \{q \in D \mid d(p,q) \leq \epsilon\}$$

où $\epsilon \in \mathbb{R}^+$ et $d(p,q)$ est la distance entre les points p et q. La distance euclidienne est généralement utilisée.

À noter que $p \in N_\epsilon(p)$.

Type de points

Définition 2

- La densité à un point p est le nombre de points à l'intérieur du cercle de rayon ϵ formé autour de p, $|N_\epsilon(p)|$.
- Un point p est un point dit central (*core point*) si $|N_\epsilon(p)| \geq MinPts$, $MinPts \in \mathbb{N}^+$.
- Un point p est un point dit frontière (*border point*) si $|N_\epsilon(p)| < MinPts$, mais $p \in N_\epsilon(w)$ et w est un point central.
- Un point p est un point dit aberrant (*noise point*) si p n'est pas un point central ou un point frontière.

Type de points

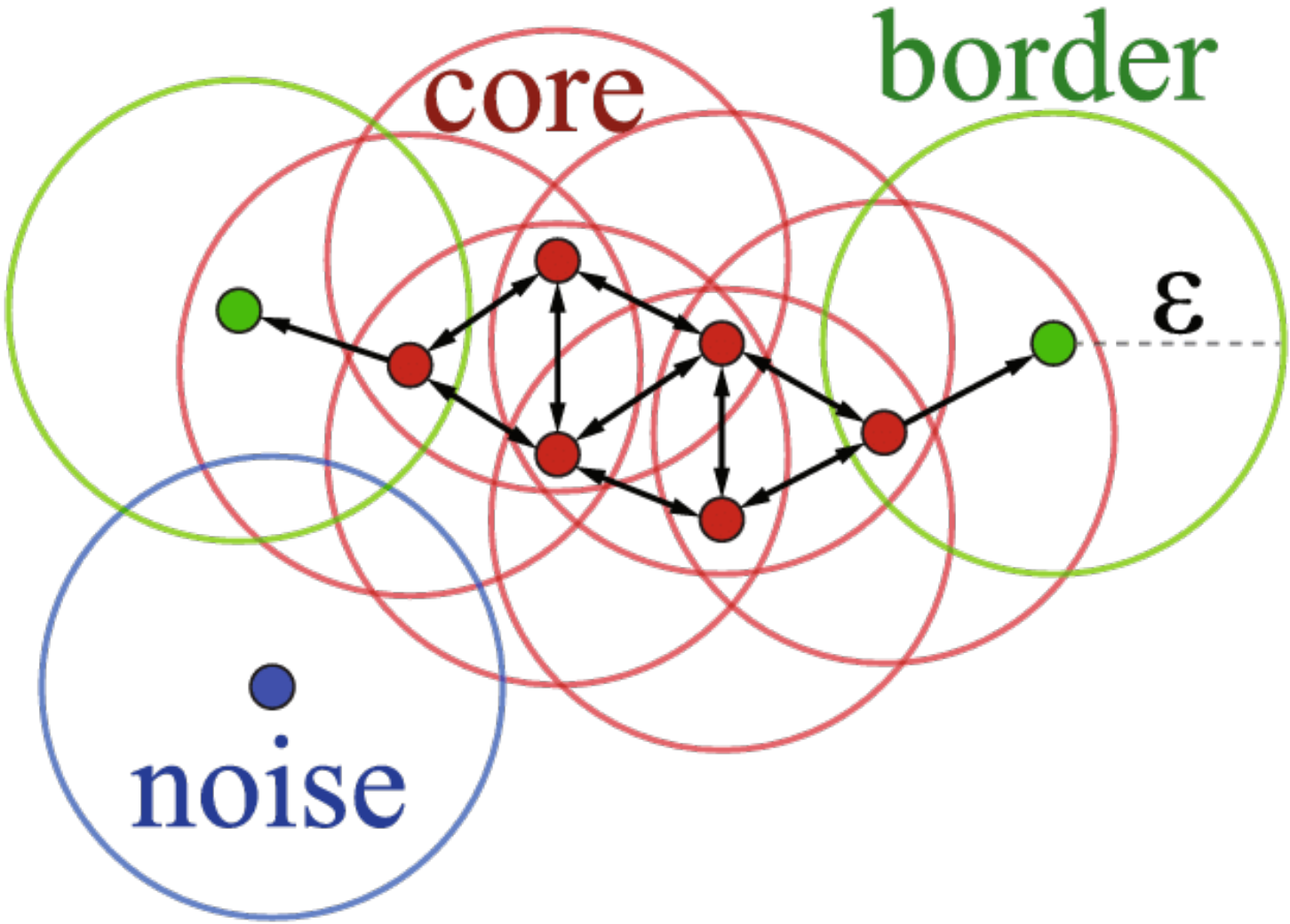


Figure 1 – Illustration tirée de Mehle et collab. (2017)

Directement accessible par la densité

Directly density-reachable

Définition 3

Un point p est directement accessible par la densité à partir du point q si

- $p \in N_\epsilon(q)$ et
- $|N_\epsilon(q)| \geq MinPts$.

À noter que cette relation est symétrique entre deux points centraux, mais ne l'est généralement pas entre un point central et un point frontière.

Accessible par la densité

Density-reachable

Définition 4

Un point p est accessible par la densité à partir du point q s'il existe une séquence de points $(p_1, p_2, \ldots, p_n)$ où $p_1 = q$, $p_n = p$ et p_{i+1} est directement accessible par la densité à partir de $p_i \ \forall \ i \in \{1, 2, \ldots, n-1\}$.

Cette relation est transitive, mais pas symétrique.

Connecté par la densité

Density-connected

Définition 5

Un point p est connecté par la densité à un point q s'il existe un point m tel que p et q sont accessibles par la densité à partir de m.

Cette relation est symétrique.

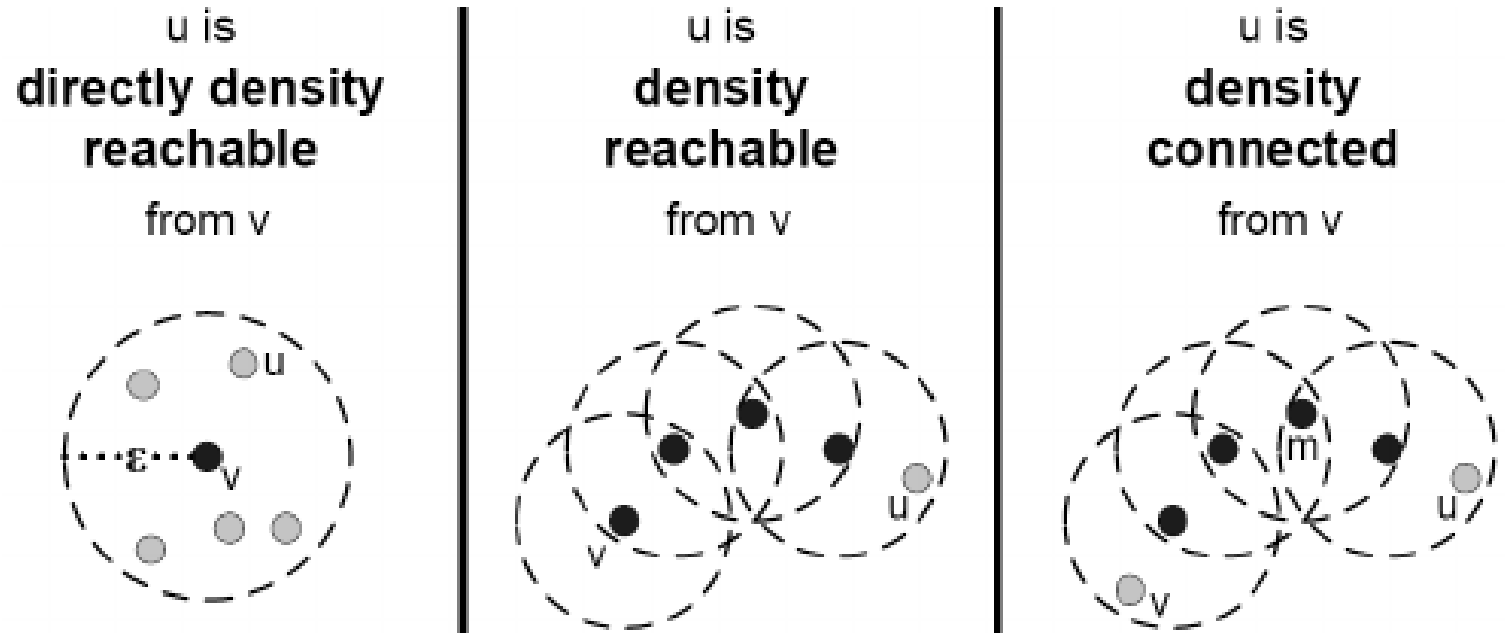


Figure 2 – Illustration tirée de Schlitter et collab. (2011)

Cluster

Définition 6

Un *cluster* C est un ensemble non vide de points appartenants à D qui respecte les conditions suivantes :

- $\forall\ (p, q)$, si $p \in C$ et q est accesible par la densité à partir de p, alors $q \in C$.
- $\forall\ (p, q) \in C$, p est connecté par la densité à q.

Cluster

Définition 6

Un *cluster* C est un ensemble non vide de points appartenants à D qui respecte les conditions suivantes :

- $\forall\ (p, q)$, si $p \in C$ et q est accesible par la densité à partir de p, alors $q \in C$.
- $\forall\ (p, q) \in C$, p est connecté par la densité à q.

Lemme 7

Soit le point $p \in D$ *et* $|N_\epsilon(p)| \geq MinPts$*. Alors,*
$O = \{o \mid o \in D, o$ *est accessible par la densité à partir de* $p\}$ *est un cluster.*

Cluster

Définition 6

Un *cluster* C est un ensemble non vide de points appartenants à D qui respecte les conditions suivantes :

- $\forall\ (p, q)$, si $p \in C$ et q est accesible par la densité à partir de p, alors $q \in C$.
- $\forall\ (p, q) \in C$, p est connecté par la densité à q.

Lemme 7

Soit le point $p \in D$ *et* $|N_\epsilon(p)| \geq MinPts$. *Alors,*
$O = \{o \mid o \in D, o$ *est accessible par la densité à partir de* $p\}$ *est un cluster.*

Lemme 8

Soit C *un cluster et* $p \in C$ *tel que* $|N_\epsilon(p)| \geq MinPts$. *Alors,*
$C = O = \{o \mid o$ *est accessible par la densité à partir de* $p\}$.

Bruit

Définition 9

Soit $C_1, C_2, \ldots, C_k$ les *clusters* de D. Le bruit B est défini ainsi :

$$B = \{p \in D \mid \forall\ i : p \notin C_i\}.$$

Algorithme

Étapes de l'algorithme

1. Commencer avec un point aléatoire p qui n'a toujours pas été visité, et calculer $N_\epsilon(p)$ et $|N_\epsilon(p)|$.

Étapes de l'algorithme

1. Commencer avec un point aléatoire p qui n'a toujours pas été visité, et calculer $N_\epsilon(p)$ et $|N_\epsilon(p)|$.
2. Si p est un point central, un nouveau *cluster* est créé avec les points appartenant à $N_\epsilon(p)$. Si un point $q \in N_\epsilon(p)$ est également un point central, $N_\epsilon(q)$ est ajouté au *cluster*. S'il n'y a plus de points centraux dans le voisinage étendu, le *cluster* est complet et finalisé.

Étapes de l'algorithme

1. Commencer avec un point aléatoire p qui n'a toujours pas été visité, et calculer $N_\epsilon(p)$ et $|N_\epsilon(p)|$.
2. Si p est un point central, un nouveau *cluster* est créé avec les points appartenant à $N_\epsilon(p)$. Si un point $q \in N_\epsilon(p)$ est également un point central, $N_\epsilon(q)$ est ajouté au *cluster*. S'il n'y a plus de points centraux dans le voisinage étendu, le *cluster* est complet et finalisé.
3. Si p n'est pas un point central, il est identifié comme du bruit pour l'instant.

Étapes de l'algorithme

1. Commencer avec un point aléatoire p qui n'a toujours pas été visité, et calculer $N_\epsilon(p)$ et $|N_\epsilon(p)|$.
2. Si p est un point central, un nouveau *cluster* est créé avec les points appartenant à $N_\epsilon(p)$. Si un point $q \in N_\epsilon(p)$ est également un point central, $N_\epsilon(q)$ est ajouté au *cluster*. S'il n'y a plus de points centraux dans le voisinage étendu, le *cluster* est complet et finalisé.
3. Si p n'est pas un point central, il est identifié comme du bruit pour l'instant.
4. Les étapes 1 à 3 sont répétées avec un nouveau point.

Étapes de l'algorithme

1. Commencer avec un point aléatoire p qui n'a toujours pas été visité, et calculer $N_\epsilon(p)$ et $|N_\epsilon(p)|$.
2. Si p est un point central, un nouveau *cluster* est créé avec les points appartenant à $N_\epsilon(p)$. Si un point $q \in N_\epsilon(p)$ est également un point central, $N_\epsilon(q)$ est ajouté au *cluster*. S'il n'y a plus de points centraux dans le voisinage étendu, le *cluster* est complet et finalisé.
3. Si p n'est pas un point central, il est identifié comme du bruit pour l'instant.
4. Les étapes 1 à 3 sont répétées avec un nouveau point.
5. Après avoir exploré tous les points, les points toujours seuls sont identifiés comme du bruit.

Paramètres

Choix des paramètres

- Choisir la valeur des deux paramètres peut s'avérer complexe et les résultats de l'algorithme sont généralement très sensibles aux valeurs choisies. De plus, les deux paramètres ne sont pas indépendants.

Choix des paramètres

- Choisir la valeur des deux paramètres peut s'avérer complexe et les résultats de l'algorithme sont généralement très sensibles aux valeurs choisies. De plus, les deux paramètres ne sont pas indépendants.
- D'un côté, si ϵ est trop petit, une grande partie des points ne sera pas dans un *cluster*. Par contre, si ϵ est trop grand, les différents *clusters* vont fusionner ensemble et la majorité des points sera dans le même *cluster*.

Choix des paramètres

- Choisir la valeur des deux paramètres peut s'avérer complexe et les résultats de l'algorithme sont généralement très sensibles aux valeurs choisies. De plus, les deux paramètres ne sont pas indépendants.
- D'un côté, si ϵ est trop petit, une grande partie des points ne sera pas dans un *cluster*. Par contre, si ϵ est trop grand, les différents *clusters* vont fusionner ensemble et la majorité des points sera dans le même *cluster*.
- Il existe une règle de pouce qui donne une valeur minimale pour *MinPts* : $MinPts \geq dim(D) + 1$. On choisira une valeur plus élevée lorsqu'on croit avoir beaucoup de bruit dans les données et pour avoir des *clusters* très significatifs.

Choix des paramètres - 2D

- Une partie intéressante de Ester et collab. (1996) est que l'auteur avance que son algorithme demande un seul paramètre d'entrée, ϵ.

Choix des paramètres - 2D

- Une partie intéressante de Ester et collab. (1996) est que l'auteur avance que son algorithme demande un seul paramètre d'entrée, ϵ.
- Soit $d_{NN}^{k}(p)$, la distance entre le point p et son $k$ième voisin le plus proche.

Choix des paramètres - 2D

- Une partie intéressante de Ester et collab. (1996) est que l'auteur avance que son algorithme demande un seul paramètre d'entrée, ϵ.
- Soit $d^k_{NN}(p)$, la distance entre le point p et son $k$ième voisin le plus proche.
- La méthode proposée demande de tracer le graphique des points $\{d^k_{NN}(p) \mid p \in D\}$ en ordre décroissant tel que $k = MinPts \geq dim(D) + 1$.

Choix des paramètres - 2D

- Une partie intéressante de Ester et collab. (1996) est que l'auteur avance que son algorithme demande un seul paramètre d'entrée, ϵ.
- Soit $d_{NN}^k(p)$, la distance entre le point p et son $k$ième voisin le plus proche.
- La méthode proposée demande de tracer le graphique des points $\{d_{NN}^k(p) \mid p \in D\}$ en ordre décroissant tel que $k = MinPts \geq dim(D) + 1$.
- On fixe ensuite ϵ en trouvant le pied de *l'éboulis* du graphique. La partie à gauche représente les $d_{NN}^k(p)$ tel que $p \in B$.

Choix des paramètres - 2D

- Une partie intéressante de Ester et collab. (1996) est que l'auteur avance que son algorithme demande un seul paramètre d'entrée, ϵ.
- Soit $d^k_{NN}(p)$, la distance entre le point p et son $k$ième voisin le plus proche.
- La méthode proposée demande de tracer le graphique des points $\{d^k_{NN}(p) \mid p \in D\}$ en ordre décroissant tel que $k = MinPts \geq dim(D) + 1$.
- On fixe ensuite ϵ en trouvant le pied de *l'éboulis* du graphique. La partie à gauche représente les $d^k_{NN}(p)$ tel que $p \in B$.
- Dans l'article original, l'auteur avance que les graphiques pour $k > 4$ ne sont pas significativement différents que pour $k = 4$ et fixe donc $MinPts = 4$. Il reste seulement à trouver ϵ avec $\{d^4_{NN}(p) \mid p \in D\}$.

Avantages et inconvénients

Avantages

- Pas besoin de spécifier le nombre de *cluster* à trouver

Avantages

- Pas besoin de spécifier le nombre de *cluster* à trouver
- Pouvoir trouver des *clusters* de toute forme

Avantages

- Pas besoin de spécifier le nombre de *cluster* à trouver
- Pouvoir trouver des *clusters* de toute forme
- L'algorithme gère très bien les données aberrantes en les identifiant

Avantages

- Pas besoin de spécifier le nombre de *cluster* à trouver
- Pouvoir trouver des *clusters* de toute forme
- L'algorithme gère très bien les données aberrantes en les identifiant
- Demande seulement deux paramètres (ou un)

Avantages

- Pas besoin de spécifier le nombre de *cluster* à trouver
- Pouvoir trouver des *clusters* de toute forme
- L'algorithme gère très bien les données aberrantes en les identifiant
- Demande seulement deux paramètres (ou un)
- Fonctionne très bien avec beaucoup de données

Inconvénients

- Le résultat n'est pas certain étant donné que les points frontières peuvent se retrouver dans différents *clusters* tout dépendant de l'ordre de visite des points. Cette situation est cependant rare.

Inconvénients

- Le résultat n'est pas certain étant donné que les points frontières peuvent se retrouver dans différents *clusters* tout dépendant de l'ordre de visite des points. Cette situation est cependant rare.
- Étant donné que les paramètres sont globaux, l'algorithme est beaucoup moins performant lorsque les *clusters* sont de densités variées.

Inconvénients

- Le résultat n'est pas certain étant donné que les points frontières peuvent se retrouver dans différents *clusters* tout dépendant de l'ordre de visite des points. Cette situation est cependant rare.
- Étant donné que les paramètres sont globaux, l'algorithme est beaucoup moins performant lorsque les *clusters* sont de densités variées.
- L'algorithme est très sensible aux valeurs des paramètres et ceux-ci sont parfois difficiles à choisir pour cette raison.

Inconvénients

- Le résultat n'est pas certain étant donné que les points frontières peuvent se retrouver dans différents *clusters* tout dépendant de l'ordre de visite des points. Cette situation est cependant rare.
- Étant donné que les paramètres sont globaux, l'algorithme est beaucoup moins performant lorsque les *clusters* sont de densités variées.
- L'algorithme est très sensible aux valeurs des paramètres et ceux-ci sont parfois difficiles à choisir pour cette raison.
- L'algorithme est techniquement limité aux données numériques ou ordinales.

En Pratique

Implémentation en R

Il existe deux paquetages intéressants pour utiliser l'algorithme :

- `fpc`, Hennig et Imports (2015), comporte plusieurs méthodes de partitionnement.
- `dbscan`, Hahsler et collab. (2019) comporte les algorithmes DBSCAN et OPTICS.[1]

J'ai personnellement utilisé `dbscan`.

1. Extenstion qui gère le problème des différentes densités, Ankerst et collab. (1999)

`dbscan::dbscan()`

Fonction pour effectuer l'algorithme.

Arguments :

- `x` : Jeu de données
- `eps` : ϵ
- `minPts` : *MinPts*, valeur par défaut de 5
- `weights` : Poids donnés à chaque donnée, valeur `NULL` par défaut

Résultats :

- `eps` : ϵ
- `minPts` : *MinPts*
- `cluster` : *Cluster* associé à chaque observation, valeur de 0 pour du bruit

`dbscan::kNNdistplot()`

Fonction pour tracer le graphique des points $\{d^k_{NN}(p) \mid p \in D\}$ pour déterminer la valeur de ϵ adéquate.

Arguments :

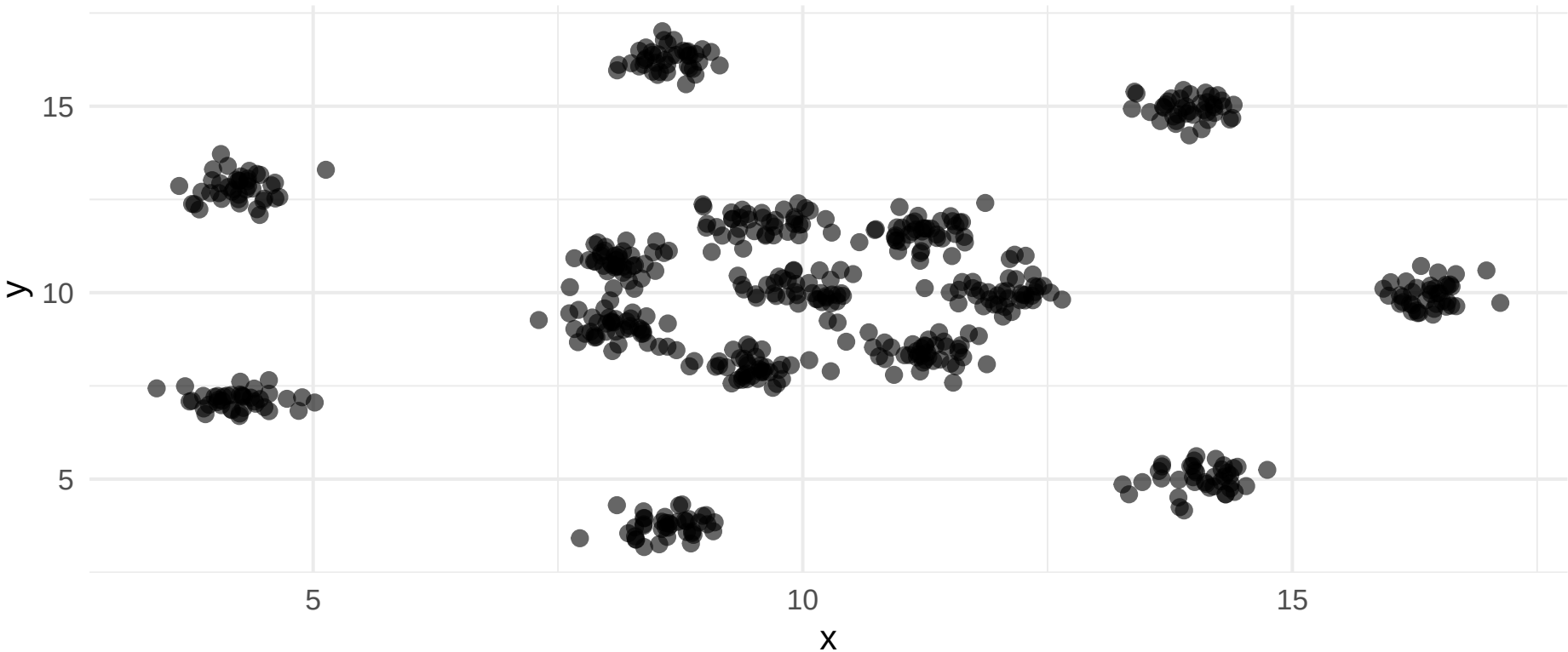
- `x` : Jeu de données
- `k` : k, valeur de 4 par défaut

La fonction trace la graphique ascendant.

Exemple

Données

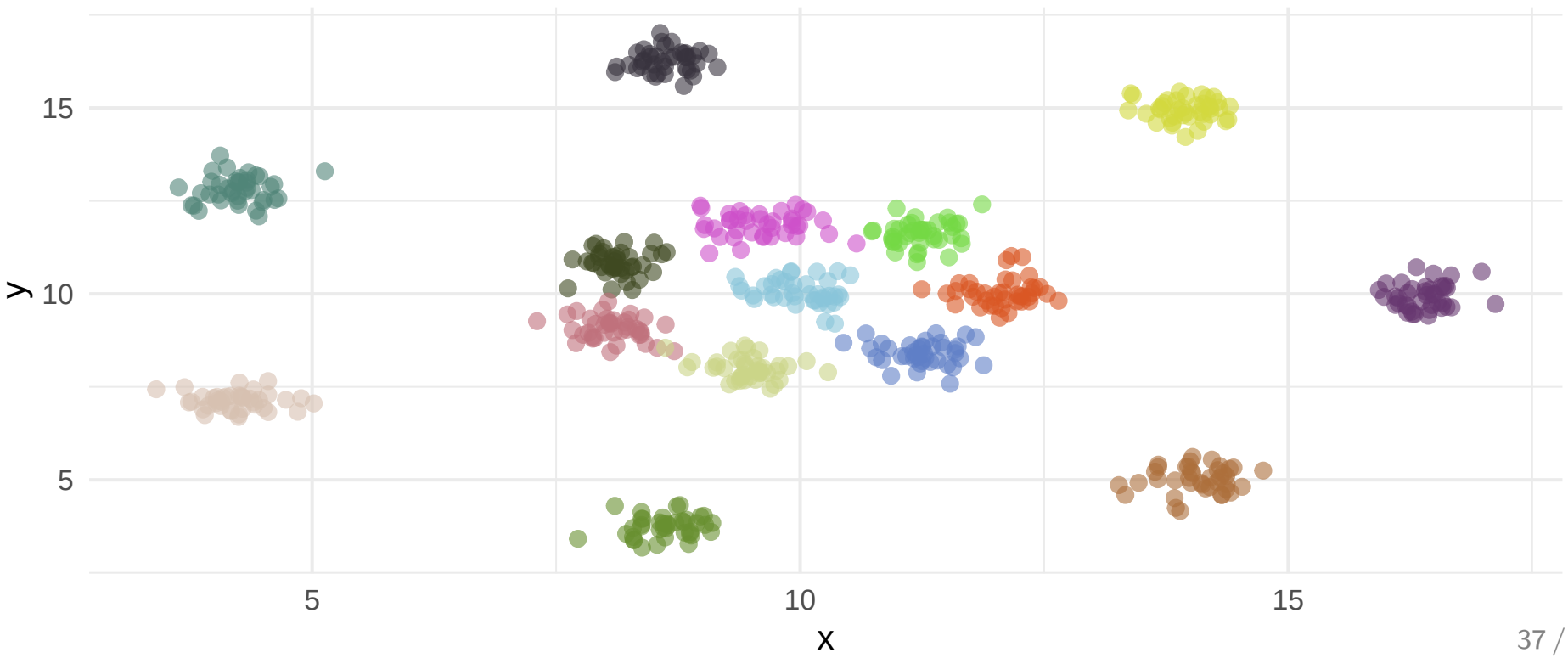
```
veenman %>%
  ggplot(aes(x,y)) + geom_point(alpha=.6, size=2) +
  theme_minimal()
```



[2]Données tirées de Veenman et collab. (2002)

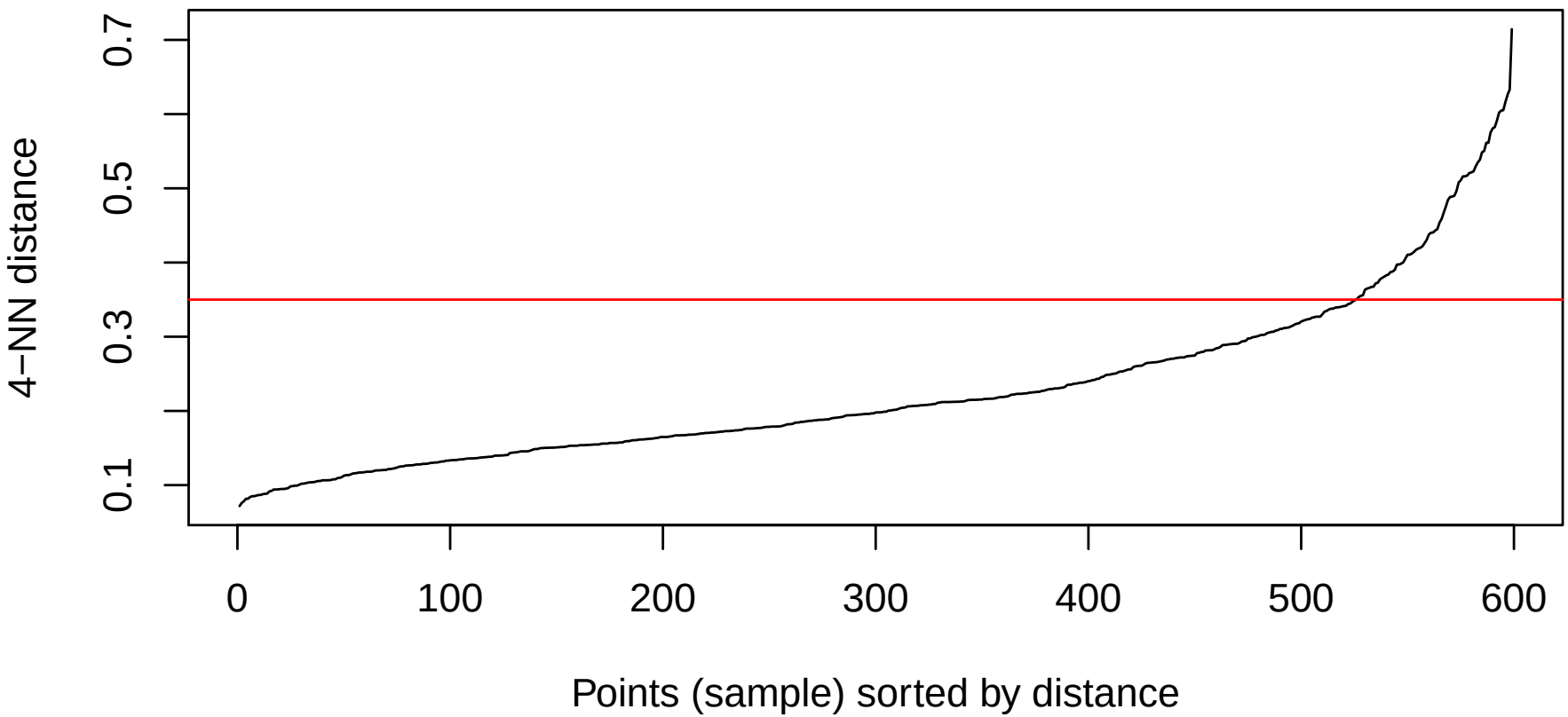
Clusters théoriques

```
veenman %>%
  ggplot(aes(x ,y, col=as.factor(th))) +
  geom_point(alpha=.6, size=2) +
  scale_color_manual(values=col_custom) +
  theme_minimal() + theme(legend.position = "")
```



Graphique des points $\{d_{NN}^k(p) \mid p \in D\}$

```
veenman.xy <- veenman %>% select(-th)
kNNdistplot(veenman.xy, k=4); abline(h=.35, col=2)
```



Algorithme

```
(db.res <- dbscan(veenman.xy, eps=.35, minPts=4))

## DBSCAN clustering for 599 objects.
## Parameters: eps = 0.35, minPts = 4
## The clustering contains 14 cluster(s) and 24 noise points.
##
##  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14
## 24 37 37 39 37 40 77 39 39 38 40 39 39 38 36
##
## Available fields: cluster, eps, minPts
```

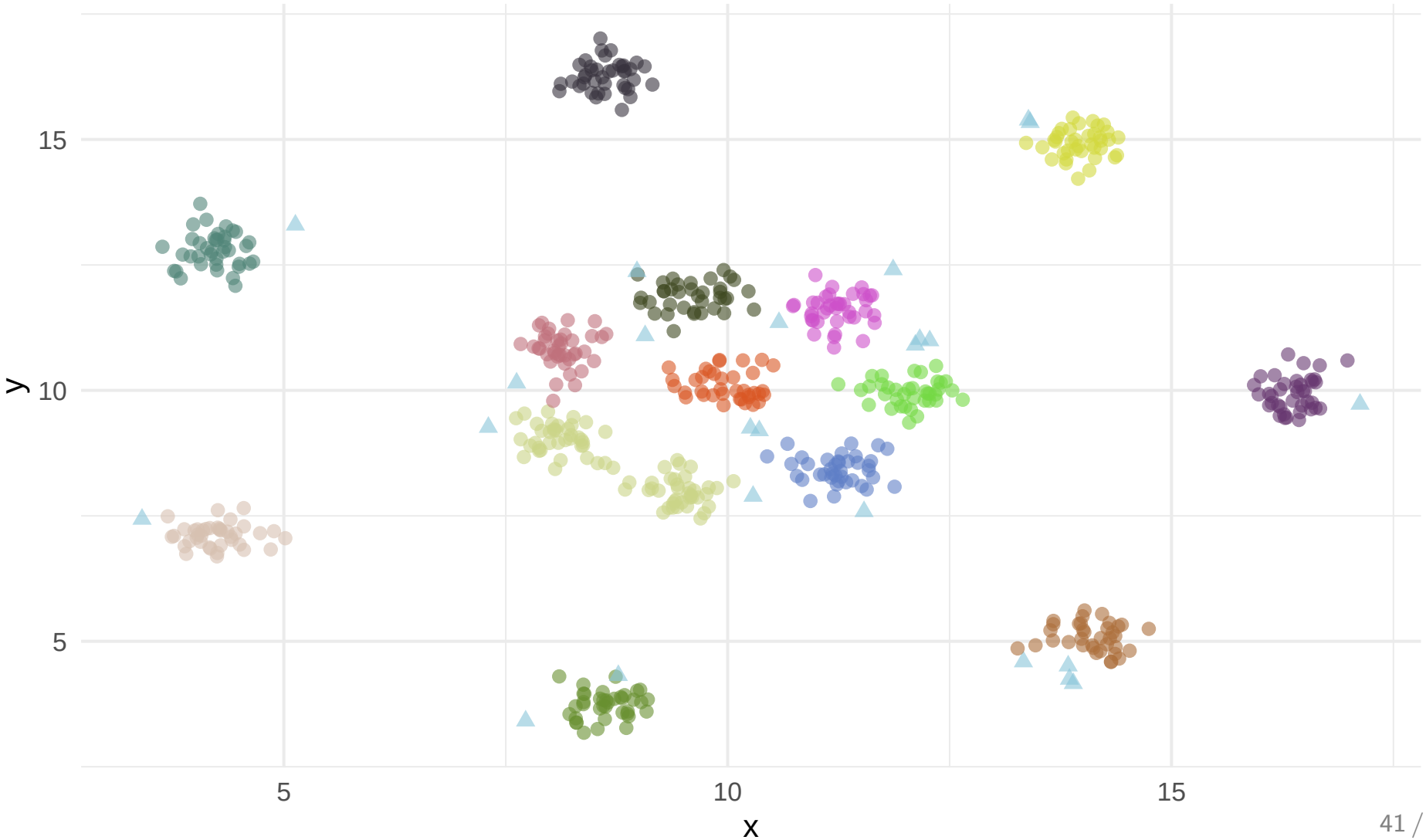
Résultats

```
veenman.db <- veenman.xy %>%
    mutate(clust_db = as.factor(db.res$cluster),
           outlier = as.factor(db.res$cluster==0))

veenman.gg <- veenman.db %>%
              ggplot(aes(x=x, y=y,
                         col=clust_db, shape=outlier)) +
              geom_point(size=2, alpha=.6) +
              scale_shape_manual(values=c(16,17)) +
              scale_color_manual(values=col_custom) +
              theme_minimal() +
              theme(legend.position = "")
```

Résultats

```
veenman.gg
```



Références I

Ankerst, M., M. M. Breunig, H.-P. Kriegel et J. Sander. 1999, «Optics : ordering points to identify the clustering structure», *ACM Sigmod record*, vol. 28, n° 2, p. 49–60.

Ester, M., H.-P. Kriegel, J. Sander, X. Xu et collab.. 1996, «A density-based algorithm for discovering clusters in large spatial databases with noise.», dans *Kdd*, vol. 96, p. 226–231.

Hahsler, M., M. Piekenbrock et D. Doran. 2019, «dbscan : Fast density-based clustering with r», *Journal of Statistical Software*, vol. 25, p. 409–416.

Hennig, C. et M. Imports. 2015, «Package 'fpc'», *URL : http ://cran. r-project. org/web/packages/fpc/fpc. pdf (available 08.07. 2017)*.

Références II

Mehle, A., B. Likar et D. Tomaževič. 2017, «In-line recognition of agglomerated pharmaceutical pellets with density-based clustering and convolutional neural network», *IPSJ Transactions on Computer Vision and Applications*, vol. 9, 10.1186/s41074-017-0019-2.

Schlitter, N., T. Falkowski et J. Laessig. 2011, «Dengraph-ho : Density-based hierarchical community detection for explorative visual network analysis», 10.1007/978-1-4471-2318-7_22.

Veenman, C. J., M. J. T. Reinders et E. Backer. 2002, «A maximum variance cluster algorithm», *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, n° 9, p. 1273–1280.