Comment trouver des structures dans vos données biologiques avec la classification non supervisée

Méthodes de distance, modèles de mélange ou réseaux de neurones.

Cathy Maugis-Rabusseau

INSA Toulouse / IMT
cathy.maugis@insa-toulouse.fr

23-05-2022

Plan

Objectif du clustering

- On observe n individus décrits par p variables

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

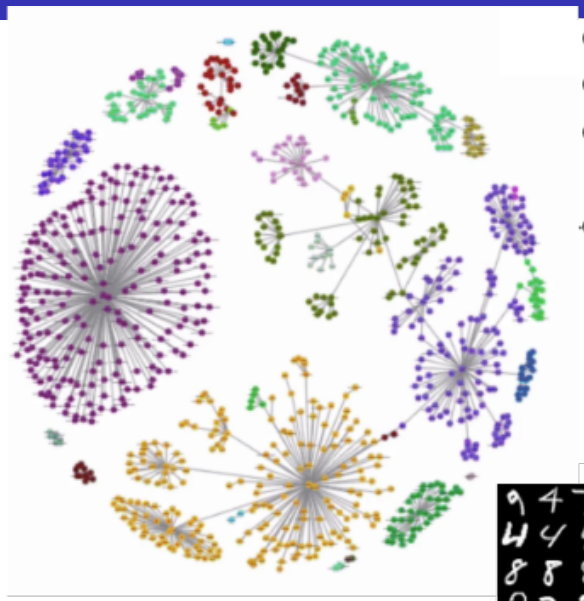
- **Classification** : organisation d'un ensemble d'individus hétérogènes en un ensemble de classes homogènes
- **Non supervisée** : on ne dispose d'aucune partition a priori des individus et on ne connait pas le nombre de classes K.

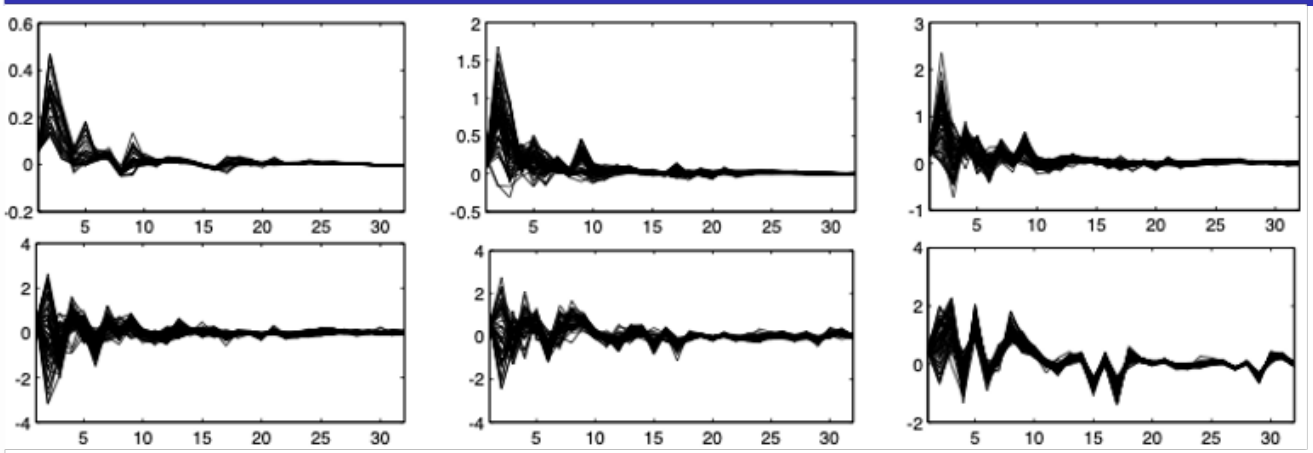
$$\Longleftrightarrow$$

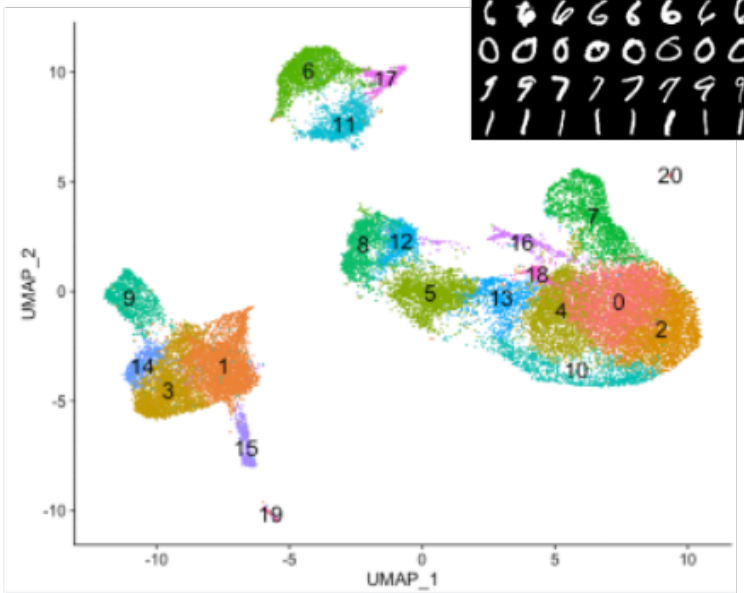
Déterminer K classes $\mathcal{P}_K = \{\mathcal{C}_1, \dots, \mathcal{C}_K\}$ des n individus à partir de X telles qu'une classe est une collection d'individus similaires entre eux et dissimilaires aux individus des autres classes (classes bien séparées).

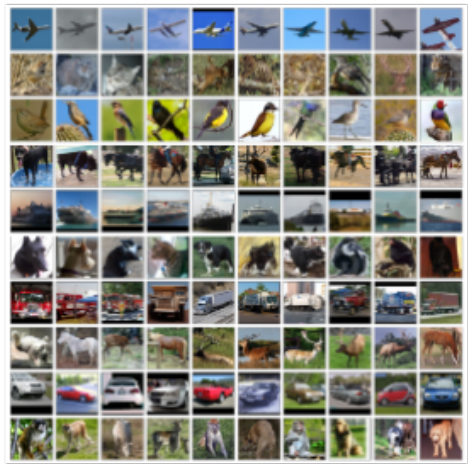
- Impossibilité d'une recherche exhaustive

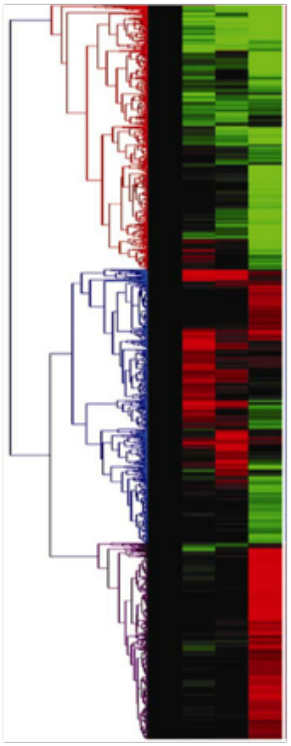
Exemples











Catégories de méthodes

- Les méthodes de clustering peuvent se différencier par
  - Type de "ressemblance" entre individus en terme de distance, de distribution de probabilité, ...
  - Type de "partitionnement" : hard ou fuzzy clustering
- Grandes catégories de méthodes :
  - Méthodes fondées sur une distance
    - Méthodes hiérarchiques
    - Méthodes par partitionnement
  - Méthodes basées sur la distribution probabiliste des données
  - Méthodes basées sur les réseaux de neurones
  - ...

Dans la suite

- Les données

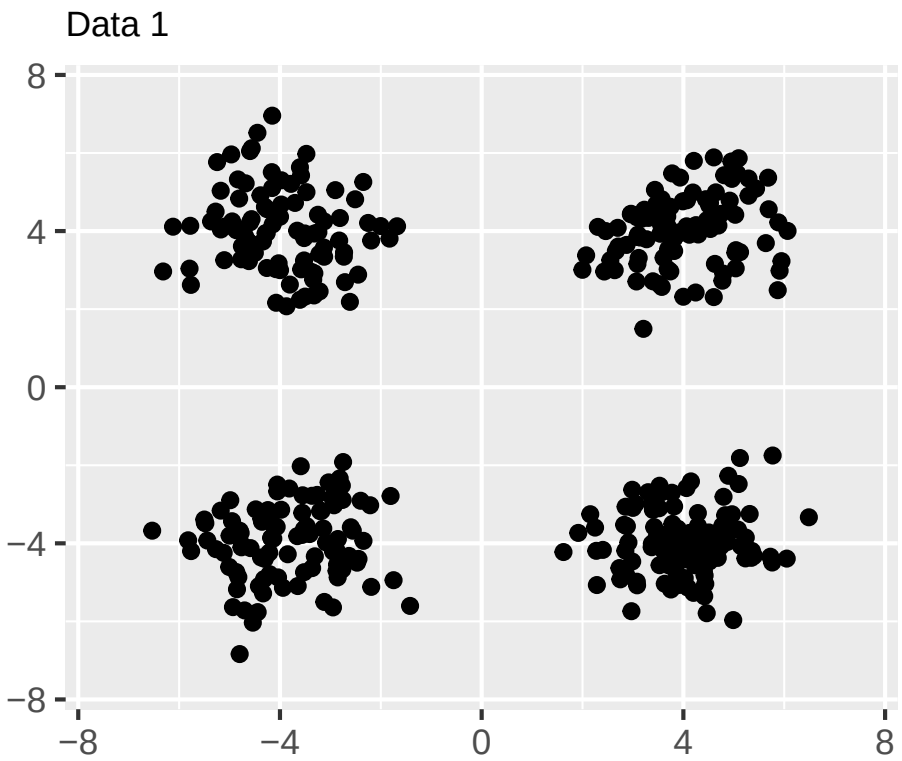
$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}$$

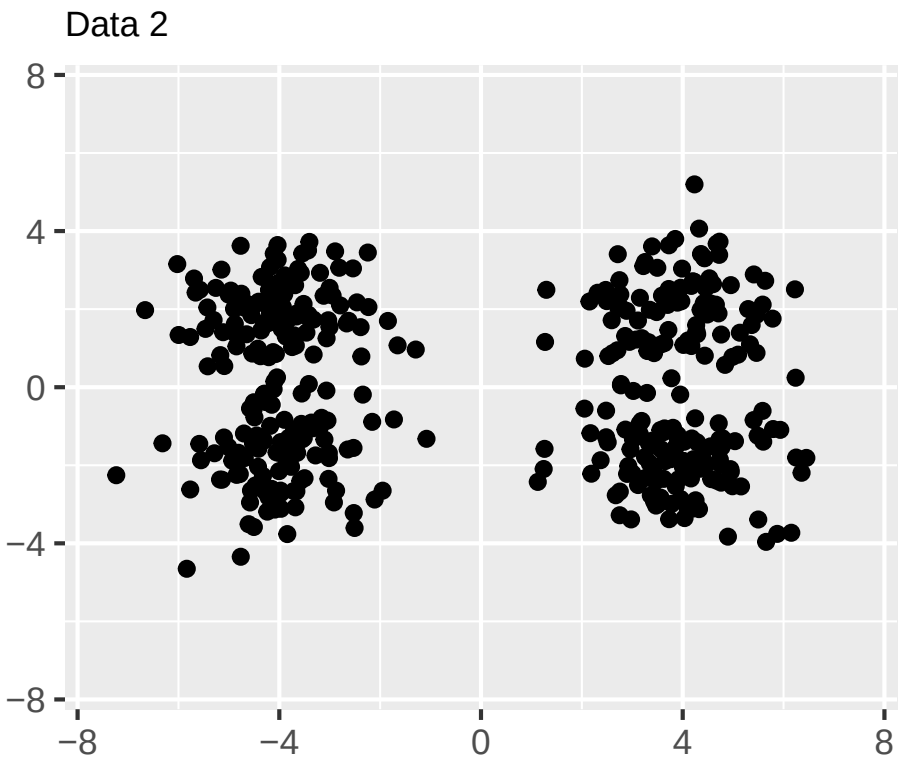
  avec $x_i \in \mathcal{X}$ (ex: $\mathcal{X} = \mathbb{R}^p$, $\{0,1\}^p$, $]-2\pi, 2\pi]^p$, ...)

- On peut partir du
  - Tableau initial des mesures
  - Tableau des mesures transformées
  - Tableau des coordonnées après une réduction de dimension (cf Cours 3)

Exemples simulés

- Jeux de données jouet ($n = 400$, $p = 2$)



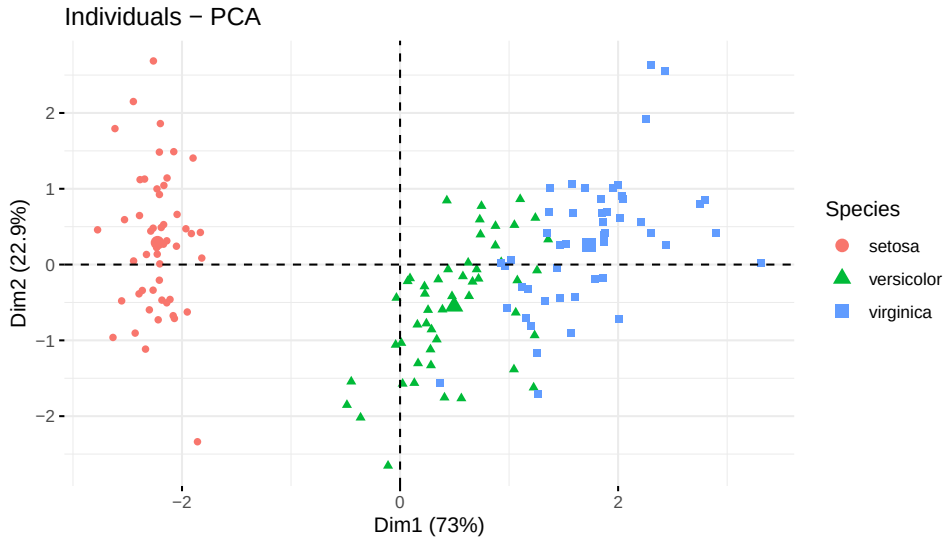


Exemple des iris

- 3 espèces d'iris : setosa (50), versicolor (50) et virginica (50)
- Mesures en centimètres de longueur du sépale, largeur du sépale, longueur du pétale et largeur du pétale

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

FIG. 2 – *I.setosa, I.versicolor, I.Virginica*



Plan

Plan

Définitions

- **Dissimilarité entre individus** : $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ telle que
  - $\forall (x_i, x_\ell) \in \mathcal{X}^2,\ d(x_i, x_\ell) = d(x_\ell, x_i)$ (symétrie)
  - $d(x_i, x_\ell) = 0 \Leftrightarrow x_i = x_\ell$
- **Similarité (normée) entre individus** : $s : \mathcal{X} \times \mathcal{X} \to [0, 1]$ telle que
  - $\forall (x_i, x_\ell) \in \mathcal{X}^2,\ s(x_i, x_\ell) = s(x_\ell, x_i)$ (symétrie)
  - $s(x_i, x_\ell) = 1 \Leftrightarrow x_i = x_\ell$
- **Distance entre individus** : dissimilarité + inégalité triangulaire

$$\forall (x_i, x_\ell, x_m) \in \mathcal{X}^3,\ d(x_i, x_m) \leq d(x_i, x_\ell) + d(x_\ell, x_m)$$

- **Distance ultramétrique** : distance + inégalité ultratriangulaire

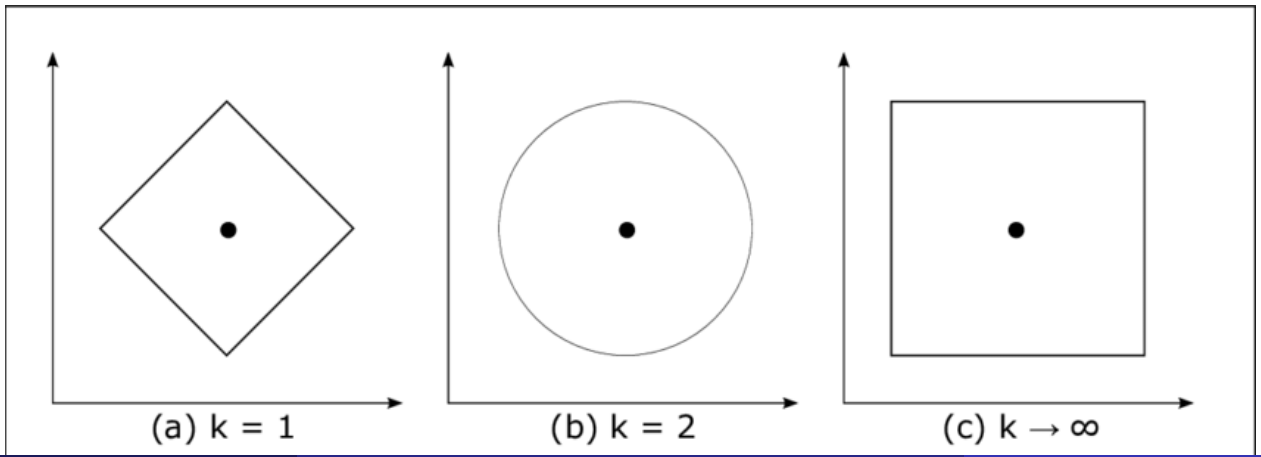
$$\forall (x_i, x_\ell, x_m) \in \mathcal{X}^3,\ d(x_i, x_m) \leq \max\left[d(x_i, x_\ell), d(x_\ell, x_m)\right]$$

Exemples pour variables quantitatives

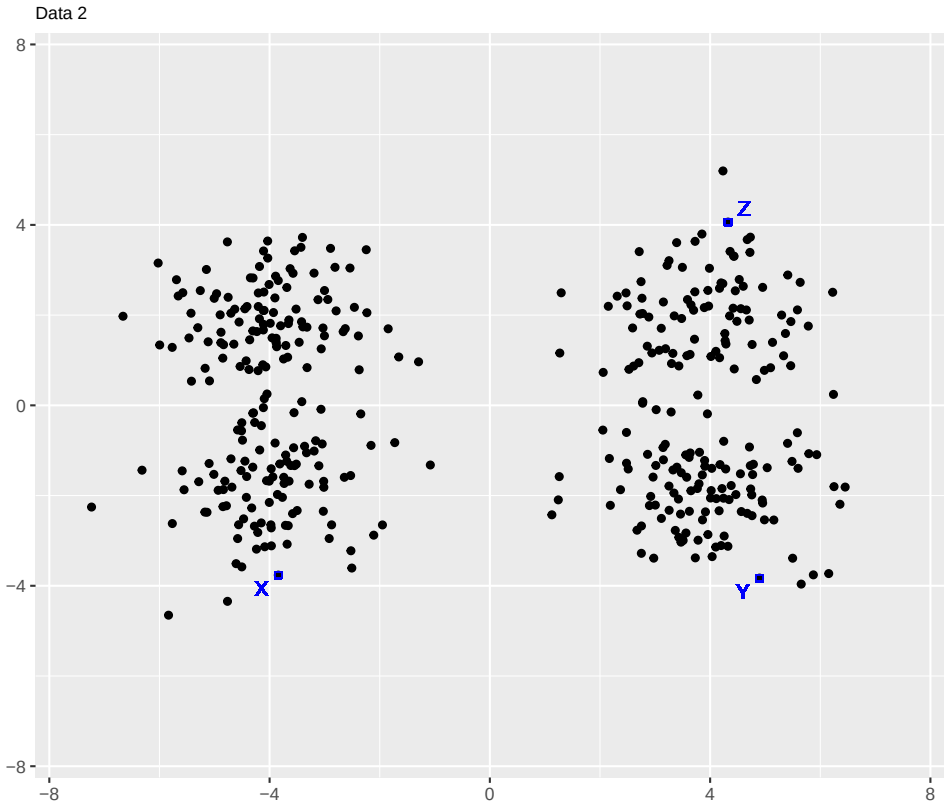
- $x_i \in \mathcal{X} = \mathbb{R}^p$ pour tout $i = 1, \ldots, n$
- Distance euclidienne usuelle $d(x_i, x_\ell) = \|x_i - x_\ell\|_2 = \sqrt{\sum_{j=1}^{p} (x_{ij} - x_{\ell j})^2}$
- Distance de Mahalanobis $d(x_i, x_\ell)^2 = (x_i - x_\ell)' \Sigma^{-1} (x_i - x_\ell)$ avec

Σ matrice de covariance

- Distance de Manhattan $d(x_i, x_\ell) = \|x_i - x_\ell\|_1 = \sum_{j=1}^{p} |x_{ij} - x_{\ell j}|$
- Norme infinie $d(x_i, x_\ell) = \max_{j=1,\ldots,p} |x_{ij} - x_{\ell j}|$



Exemples pour variables quantitatives



- Distance euclidienne $\|.\|_2$

	X	Y	Z
X	0.00	8.74	11.31
Y	8.74	0.00	7.91
Z	11.31	7.91	0.00

- Distance de Manhattan $\|.\|_1$

	X	Y	Z
X	0.00	8.80	15.99
Y	8.80	0.00	8.46
Z	15.99	8.46	0.00

- Distance de Mahalanobis $\|.\|_{2,\Sigma^{-1}}$

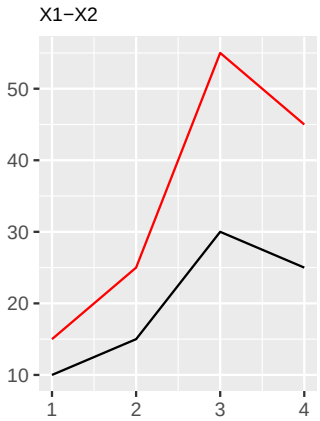
$$\Sigma = \begin{pmatrix} 16.71 & -0.53 \\ -0.53 & 4.6 \end{pmatrix}$$

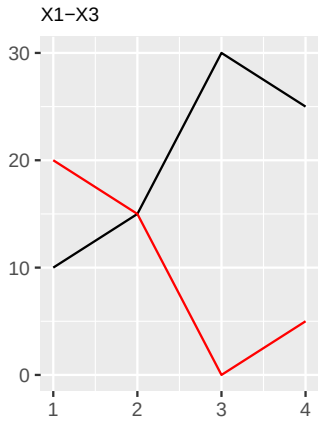
	X	Y	Z
X	0.00	2.14	4.28
Y	2.14	0.00	3.68
Z	4.28	3.68	0.00

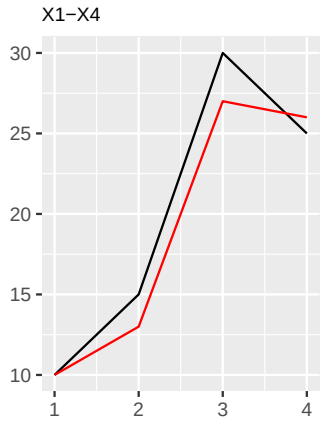
Exemples pour variables quantitatives

- Coefficient de corrélation de Pearson $\rho(x_i, x_\ell) \in [-1, 1]$
- Exemples de dissimilarités basées sur la corrélation

$$d(x_i, x_\ell) = 1 - \rho(x_i, x_\ell) \quad d(x_i, x_\ell) = 1 - |\rho(x_i, x_\ell)| \quad d(x_i, x_\ell) = 1 - \rho(x_i, x_\ell)^2$$







	X_2	X_3	X_4
$1 - \rho(X_1, .)$	0	2	0.02
$1 - \lvert\rho(X_1, .)\rvert$	0	0	0
$1 - \rho(X_1, .)^2$	0	0	0.04
$\lVert X_1 - . \rVert_2$	33,9	37,41	3,74

Exemples pour variables binaires

- Table de contingence entre 2 individus x_i et $x_\ell \in \{0,1\}^p$:

	1	0
1	m_{11}	m_{10}
0	m_{01}	m_{00}

- Variable binaire **symétrique**
  = pas d'influence sur le choix du codage $0-1$

Appariement simple $\quad s(x_i, x_\ell) = \frac{m_{11}+m_{00}}{m_{11}+m_{00}+m_{10}+m_{01}}$

Rogers et Tanimoto $\quad s(x_i, x_\ell) = \frac{m_{11}+m_{00}}{m_{11}+m_{00}+2(m_{10}+m_{01})}$

Sokal et Sneath $\quad s(x_i, x_\ell) = \frac{2(m_{11}+m_{00})}{2(m_{11}+m_{00})+m_{10}+m_{01}}$

- Variable binaire **asymétrique**
  = les valeurs 0-1 n'ont pas la même importance

Jaccard $\quad s(x_i, x_\ell) = \frac{m_{11}}{m_{11}+m_{10}+m_{01}}$

Dice $\quad s(x_i, x_\ell) = \frac{2m_{11}}{2m_{11}+m_{10}+m_{01}}$

Exemple

Nom	Sexe	Fièvre	Toux	Test1	Test2	Test3	Test4
Jules	M	Y	N	P	N	N	N
Marie	F	Y	N	P	N	P	N
Pierre	M	Y	P	N	N	N	N
Anna	F	N	P	N	P	N	N

- Jaccard :

	Jules	Marie	Pierre	Anna
Jules	1.0	0.5	0.50	0.00
Marie	0.5	1.0	0.20	0.00
Pierre	0.5	0.2	1.00	0.25
Anna	0.0	0.0	0.25	1.00

- Appariement simple :

	Jules	Marie	Pierre	Anna
Jules	1.00	0.71	0.71	0.29
Marie	0.71	1.00	0.43	0.29
Pierre	0.71	0.43	1.00	0.57
Anna	0.29	0.29	0.57	1.00

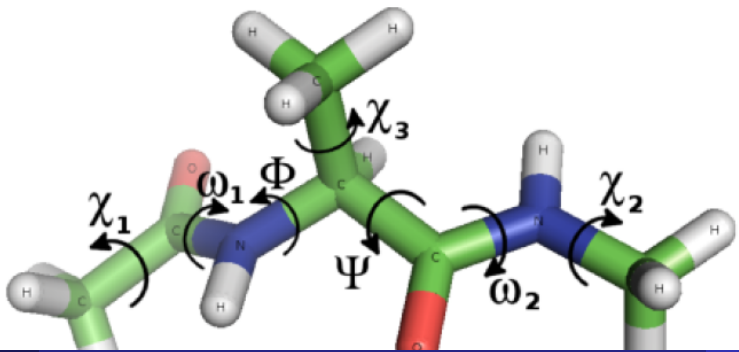
Et aussi . . .

- Données compositionnelles (ex : en microbiome)
  - $x_i = (x_{i1}, \ldots, x_{ip}) \in [0,1]^p$, $\sum_{j=1}^{p} x_{ij} = 1$
  - Distance d'Aitchison

$$d(x_i, x_\ell) = \|\mathrm{CLR}(x_i) - \mathrm{CLR}(x_\ell)\|_2$$

avec $\mathrm{CLR}(x_i) = (\ln(\frac{x_{ij}}{g(x_i)}))_j$ et $g(.)$ moyenne géométrique

- Données angulaires (ex : conformations de molécule)
  - $x_i = (x_{i1}, \ldots, x_{ip}) \in]-2\pi, 2\pi]^p$
  - $d(x_i, x_\ell) = \sum_{j=1}^{p} [\pi - ||x_{ij} - x_{\ell j}| - \pi|]$
  - $d(x_i, x_\ell) = \sum_{j=1}^{p} 2(1 - \cos(x_{ij} - x_{\ell j}))$
  - . . .



Variables mixtes

- Stratégie 1 : tout transformer en variables de même nature et choisir une distance appropriée
- Stratégie 2 : **Métrique de Gower** pour des variables quantitatives et binaires / nominales
- ...
- Attention : il est difficile de donner la même importance à des variables de natures différentes dans une distance

Quelques commandes

- Avec R 
  - `dist()` : method= “euclidian”, “manhattan”, “minkowski”, “maximum”, “canberra”, “binary”
  - `daisy()` [library(cluster)] : metric= “euclidian”, “manhattan”,“gower”
  - Dans `library(ade4)` : `dist.binary()`, `dist.quant()`, `dist.ktab()`
- Avec Python 
  - `dist.euclidean` [Numpy]
  - `scipy.spatial.distance` [SciPy]

Conclusion

- Bien adapter le choix de la distance (dissimilarité) à
  - la nature des données étudiées
  - la définition de ressemblance entre individus dans le contexte
  - selon la méthode de clustering choisie
- Attention au comportement de la distance en grande dimension (bcp de variables)

Plan

Définitions

- Soit d une **distance euclidienne** entre individus.

Soit $\mathcal{P}_K = \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ une partition des individus en K classes.

- **Inertie totale** : $I_T = \frac{1}{n} \sum_{i=1}^{n} d(x_i, c)^2$

  où $c = \frac{1}{n} \sum_{i=1}^{n} x_i$ est le centre de gravité du nuage de points

- **Inertie interclasse** : $I_{inter} = \frac{1}{n} \sum_{k=1}^{K} |\mathcal{C}_k| \times d(m_k, c)^2$

  où $m_k = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} x_i$ est le centre de gravité de la classe $\mathcal{C}_k$

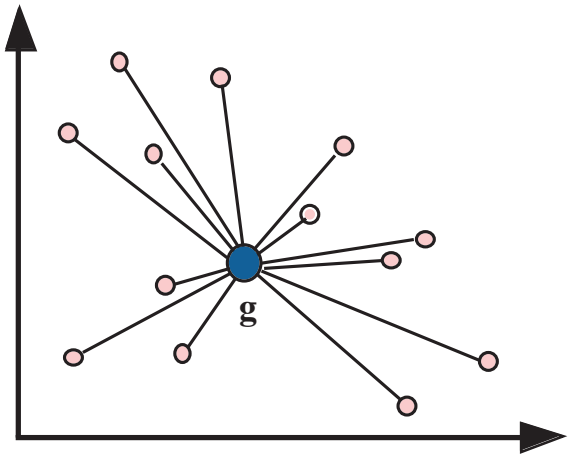
$\Rightarrow$ variance des centres des classes

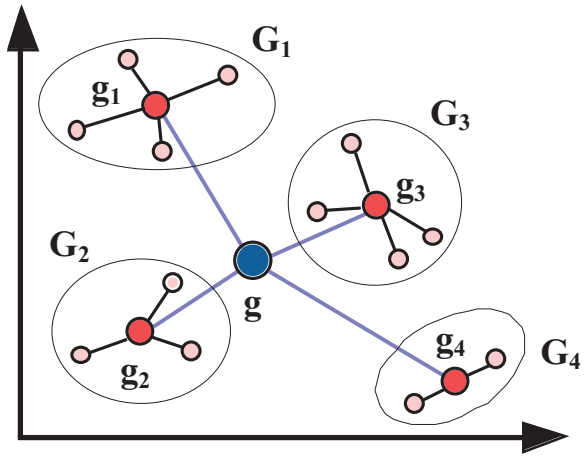
- **Inertie intra-classe** : $I_{intra} = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} d(x_i, m_k)^2$

$\Rightarrow$ variance des points d'une même classe

Propriété de Huygens

$$I_T = I_{inter} + I_{intra}$$

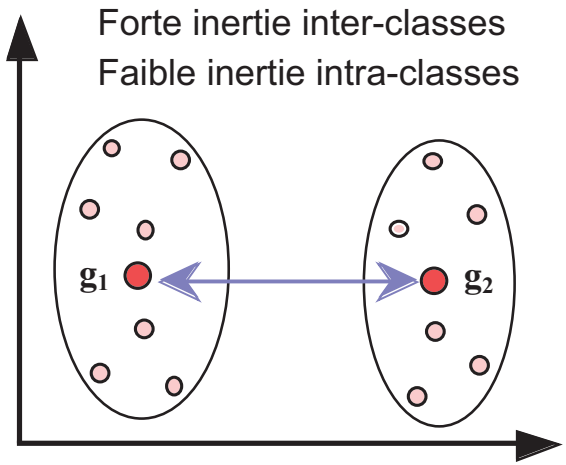


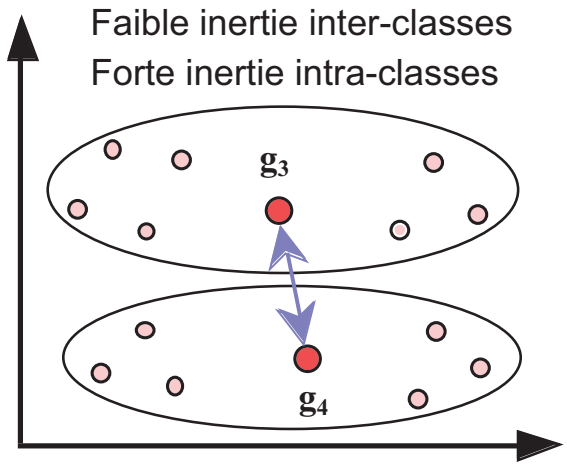


Bisson (2001)

Objectif

Minimiser l'inertie intra-classe $\Longleftrightarrow$ Maximiser l'inertie inter-classe





Bisson (2001)

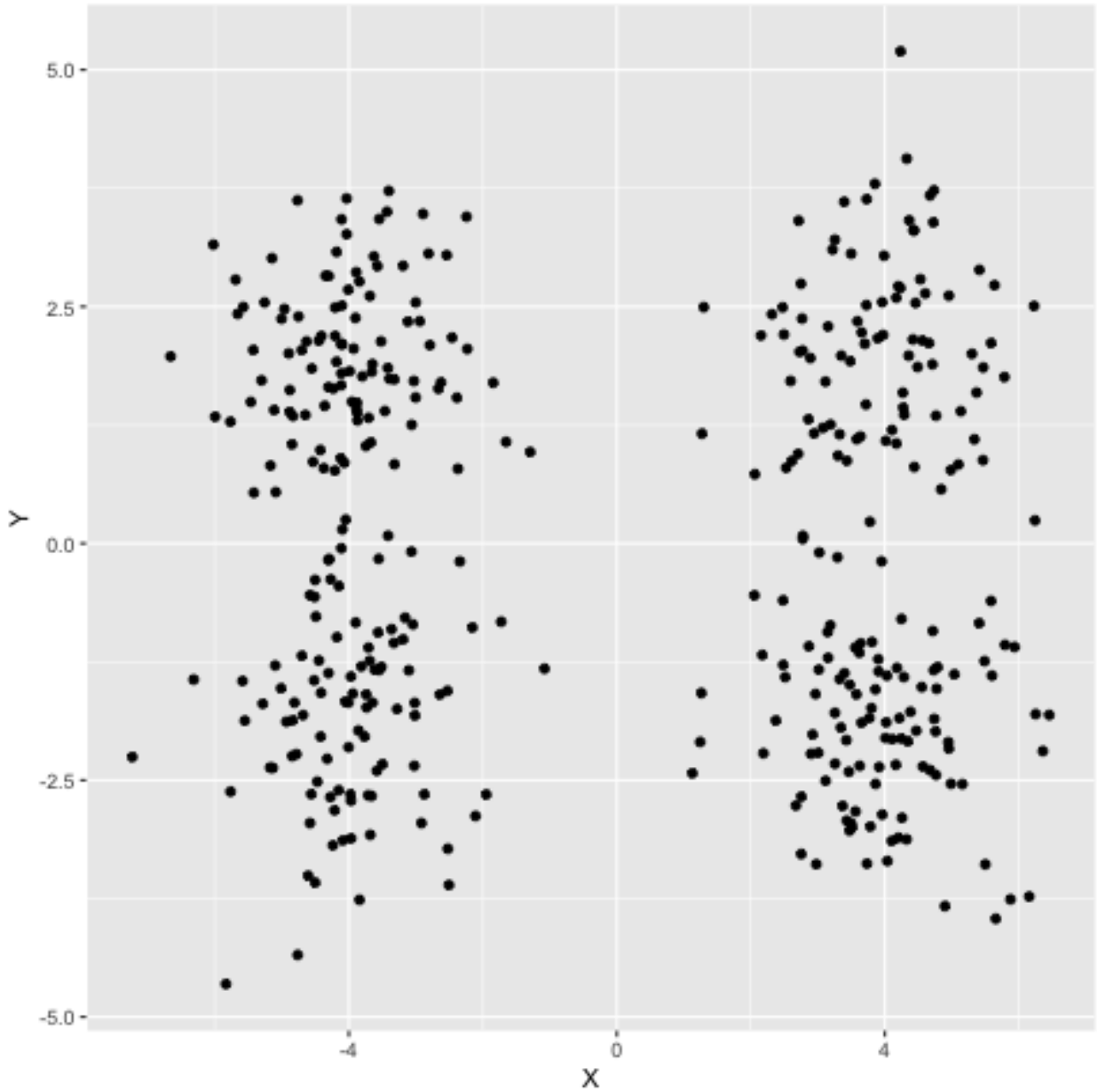
Plan

Plan

Principe

- On considère des **variables quantitatives** et d une distance (euclidienne)
- But : trouver une partition de l'ensemble des individus telle que **l'inertie intra-classe soit minimale**.
- Principe général de la méthode des **centres mobiles** (Forgy, 65)
  - Initialisation : choix de K et noyaux initiaux $c_1^{(0)}, \ldots, c_K^{(0)}$
  - A l'itération t
    - Affectation des individus à la classe la plus proche
    - Mise à jour des noyaux (centres de gravité)
  - Arrêt de l'algorithme quand la classification n'est plus modifiée
- Variantes :
  - **K-means** (MacQueen, 67)
  - **Nuées dynamiques** (Diday, 71)

Principe



Quelques commandes

- Avec 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), nuées dynamiques par défaut

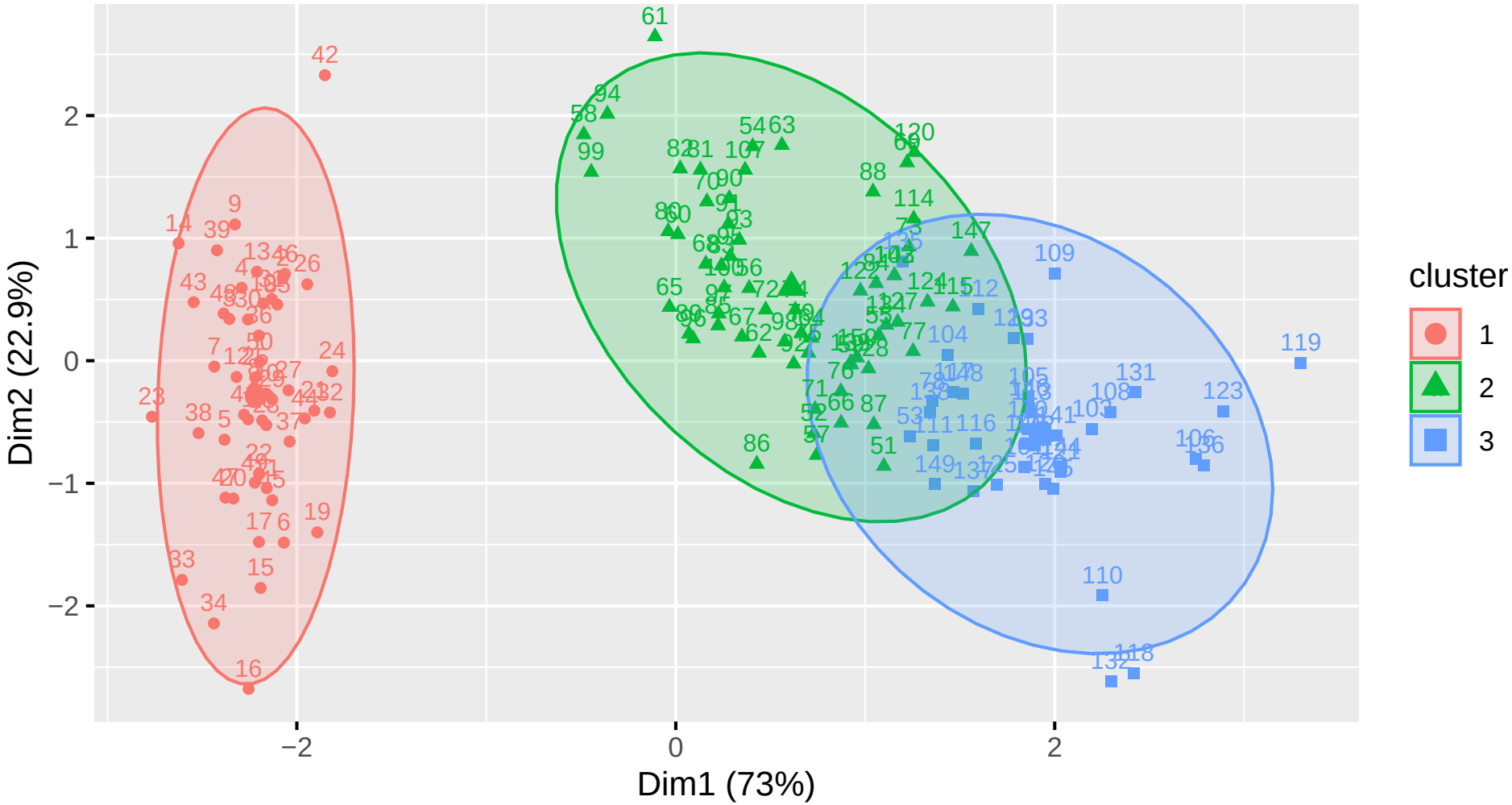
`km$cluster`, `km$centers`, `km$withinss`, `km$size`

- Avec Python 

  `kmeans=sklearn.cluster.KMeans(n_clusters=,...);`
  `kmeans.fit(data)` `kmeans.labels_`, `kmeans.cluster_centers_`,
  ...

Exemple

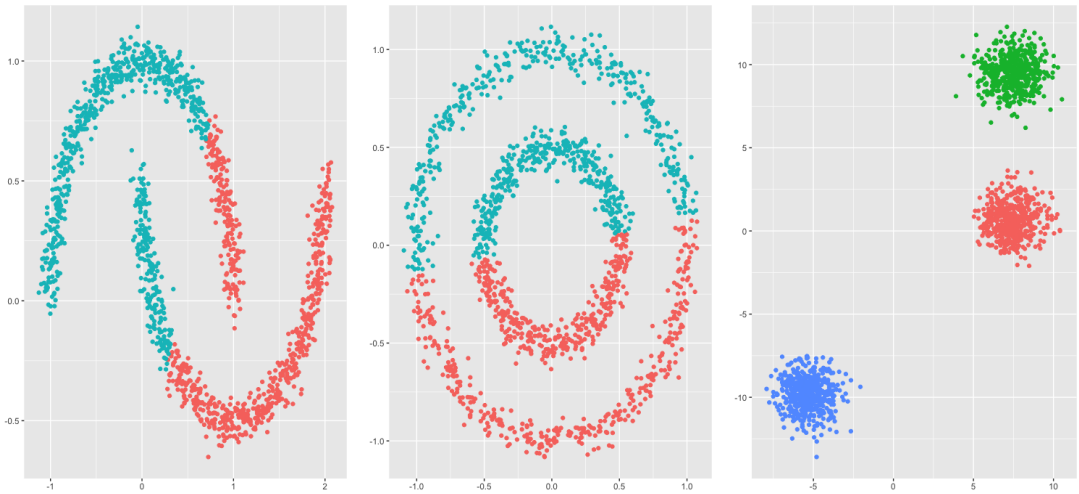
```
data(iris)
kmiris=kmeans(iris[,1:4],centers=3)
fviz_cluster(kmiris,data=iris[,1:4],ellipse.type="norm",labelsize=8)+ggtitle("")
```



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



Plan

Choix de K

- Pour chaque valeur de $K \in \{2, \ldots, K_{\max}\}$, 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
  Ex: silhouette, Gap statistique, ...
- Autres critères de type sélection de modèles (voir modèles de mélange)
- ...

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 $K \mapsto \text{RSQ}(K)$ forme un coude.

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

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

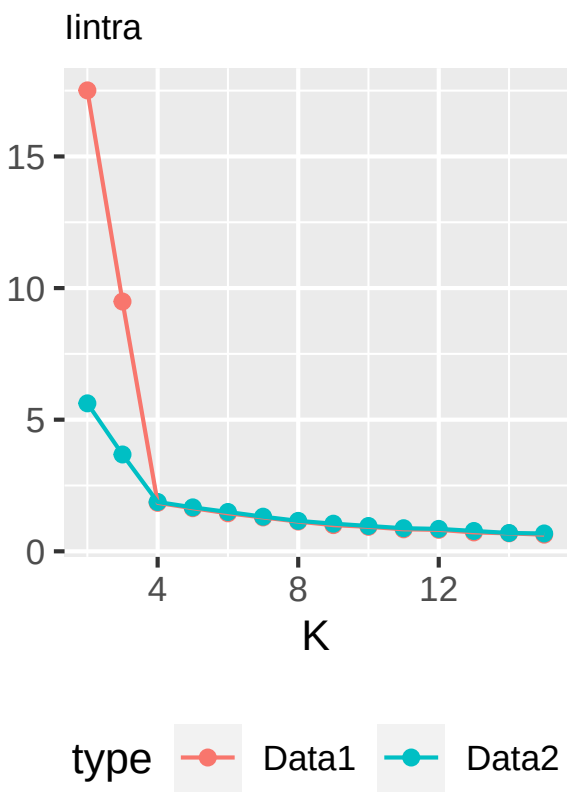
On retient l'endroit où on a la plus forte réduction du SPRSQ.

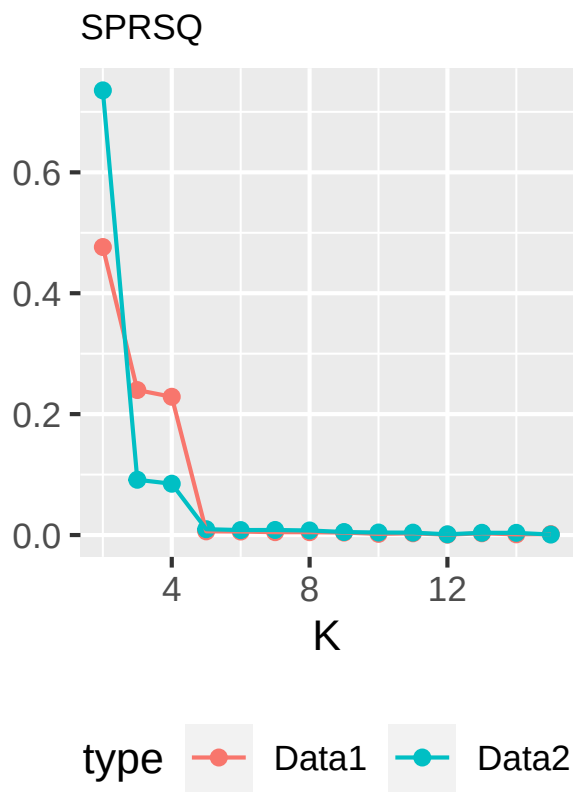
- **Calinski-Harabasz (CH)** :

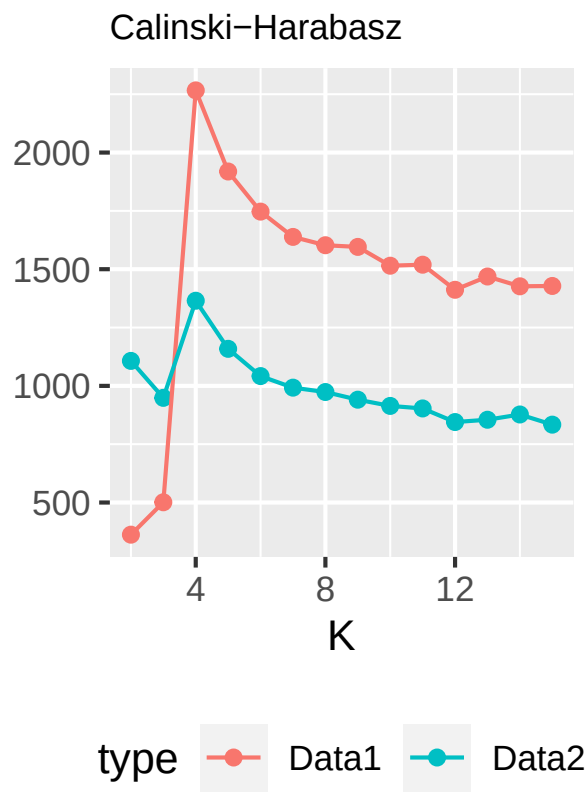
$$K \mapsto PseudoF(K) = \frac{I_{inter}(\mathcal{P}_K)/(K-1)}{I_{intra}(\mathcal{P}_K)/(n-K)}$$

On cherche un pic sur cette courbe

Exemple - Critères fondés sur les inerties

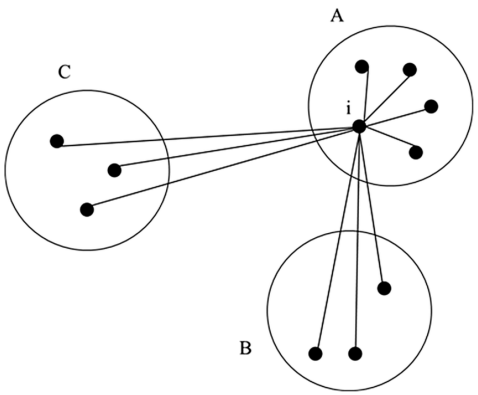






Silhouette

- Pour toute valeur $K \in \{1, \ldots, K_{\max}\}$:
  - $\{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ classification de $\{1, \ldots, n\}$
  - $\forall i \in \{1, \ldots, n\}, \exists! k \in \{1, \ldots, K\};\ i \in \mathcal{C}_k$
    - $a(i) = \frac{1}{|\mathcal{C}_k|-1} \sum_{\substack{\ell \in \mathcal{C}_k \\ \ell \neq i}} d(x_i, x_\ell)$ 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 : $\hat{K} = \underset{1 \leq K \leq K_{\max}}{argmax}\ S(K)$



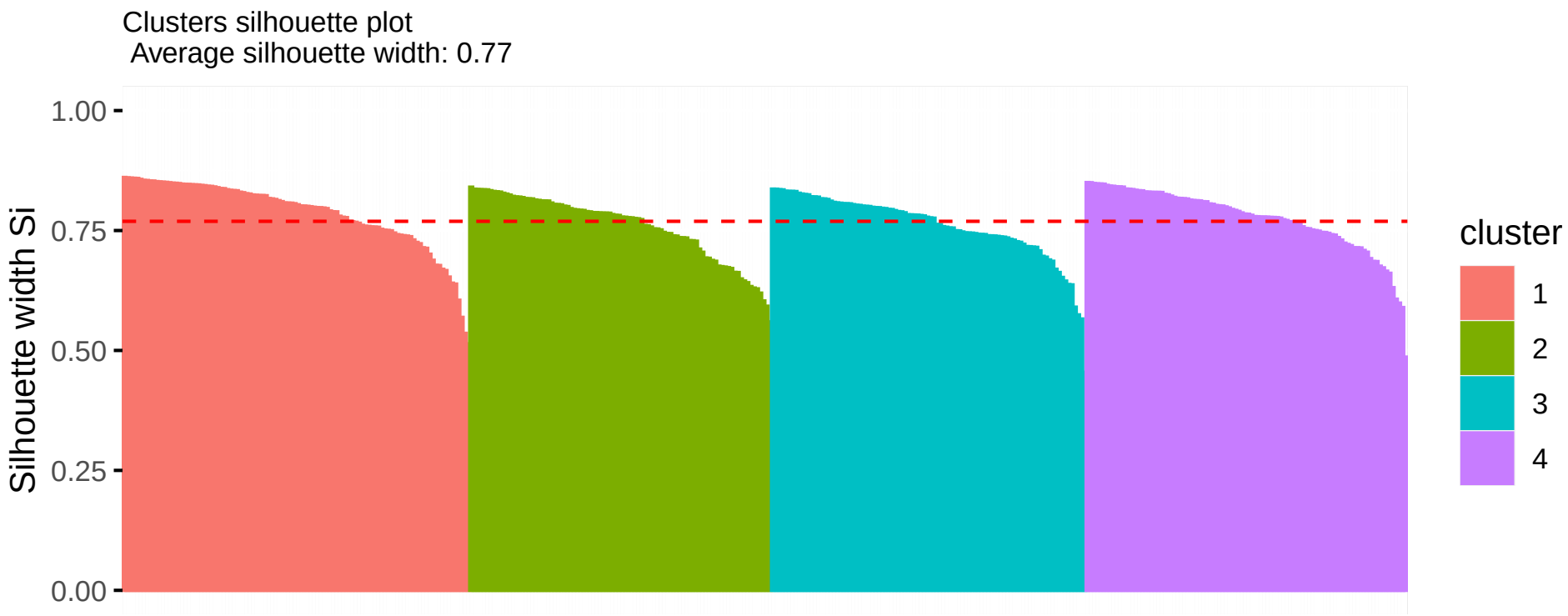
- avec `silhouette(.)` [library(cluster)]
- avec

`silhouette_samples(.)`, `silhouette_score(.)` [scikit-learn]

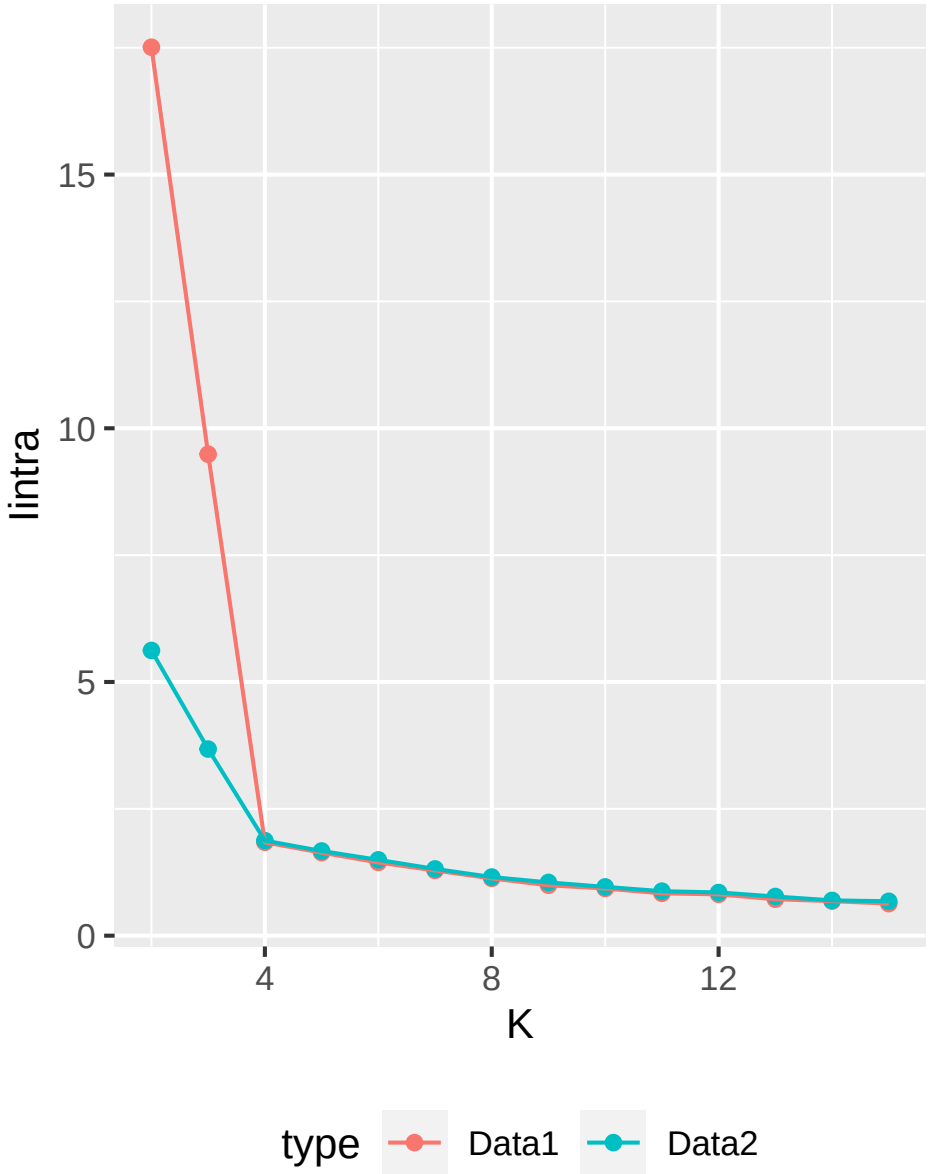
Exemple - Silhouette

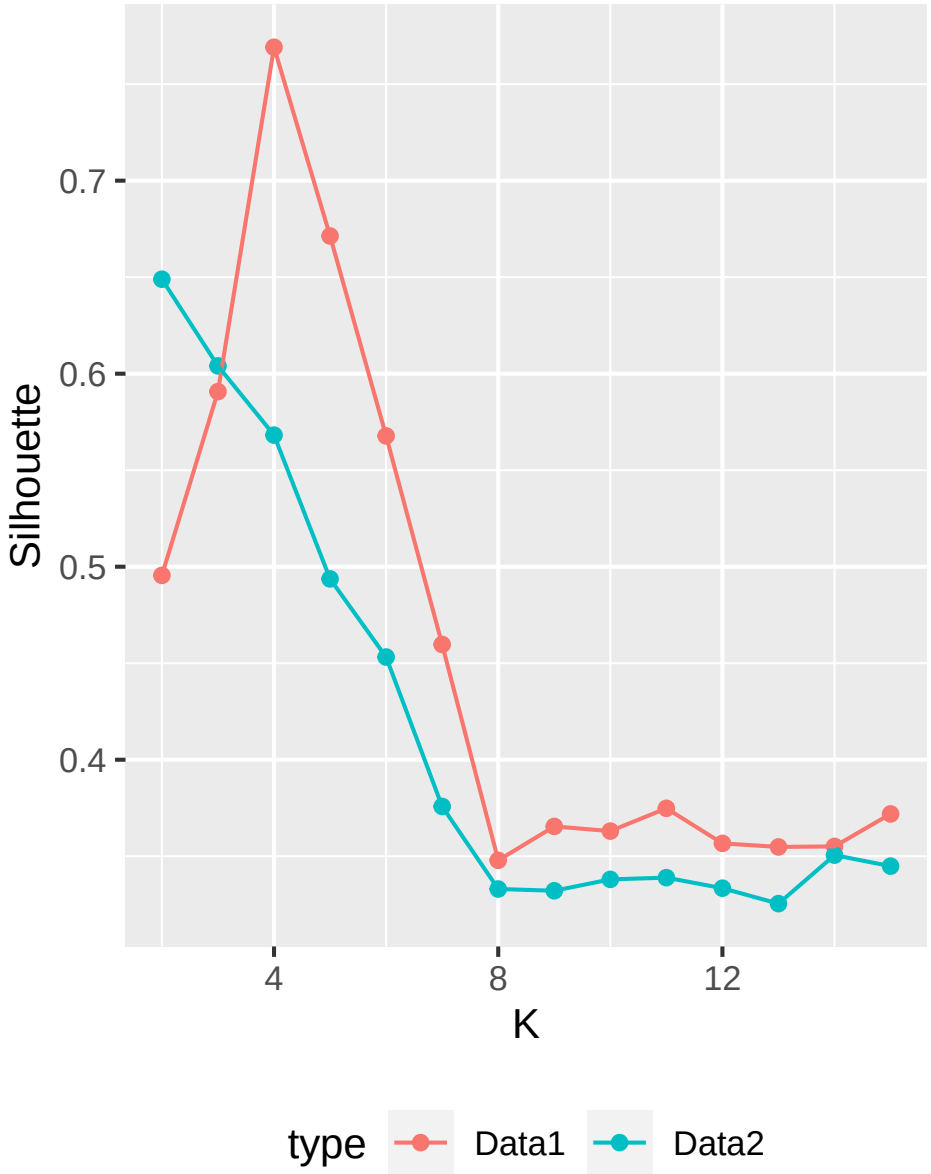
```
Classif<-kmeans(datasimu1[,1:2],4,nstart=10)
aux<-silhouette(Classif$cl, daisy(datasimu1[,1:2]))
fviz_silhouette(aux)+theme(plot.title = element_text(size =9))
```

```
  cluster size ave.sil.width
1       1  108          0.79
2       2   94          0.76
3       3   98          0.76
4       4  100          0.77
```



Exemple - Silhouette

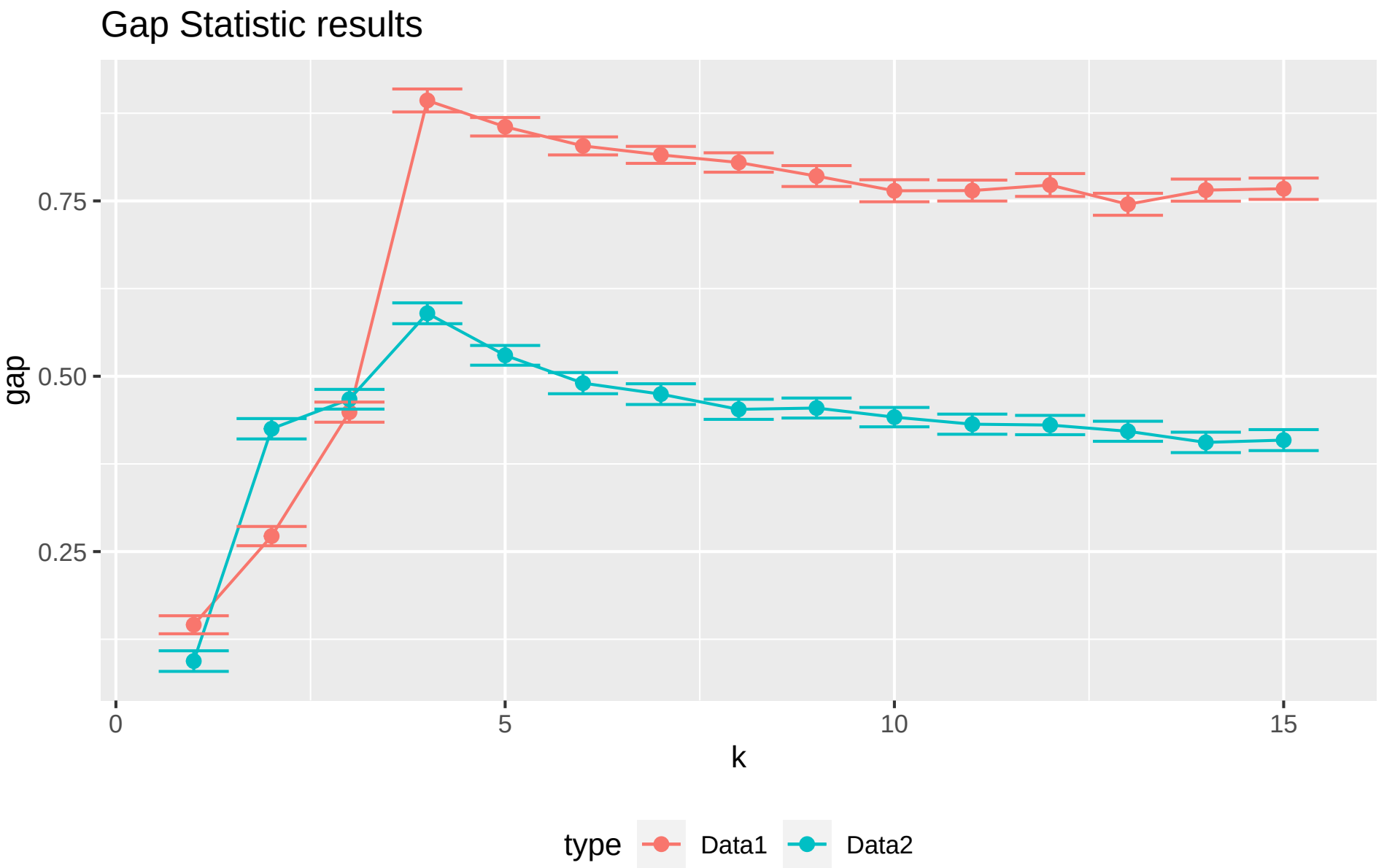




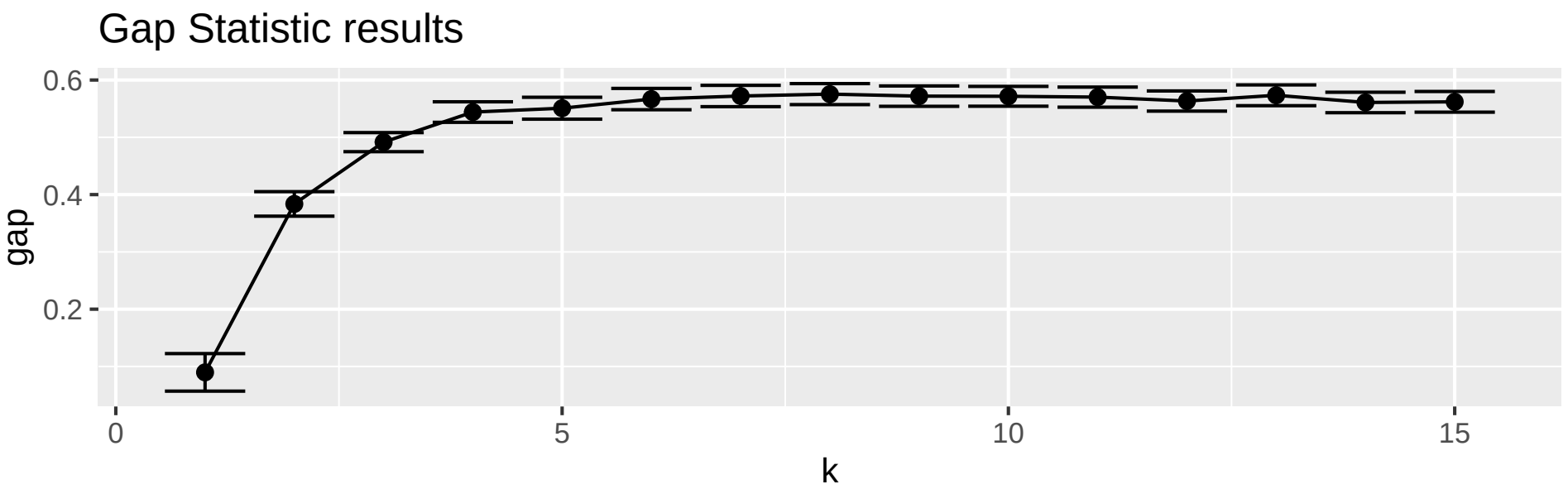
Gap statistique (Tibshirani et al.,01)

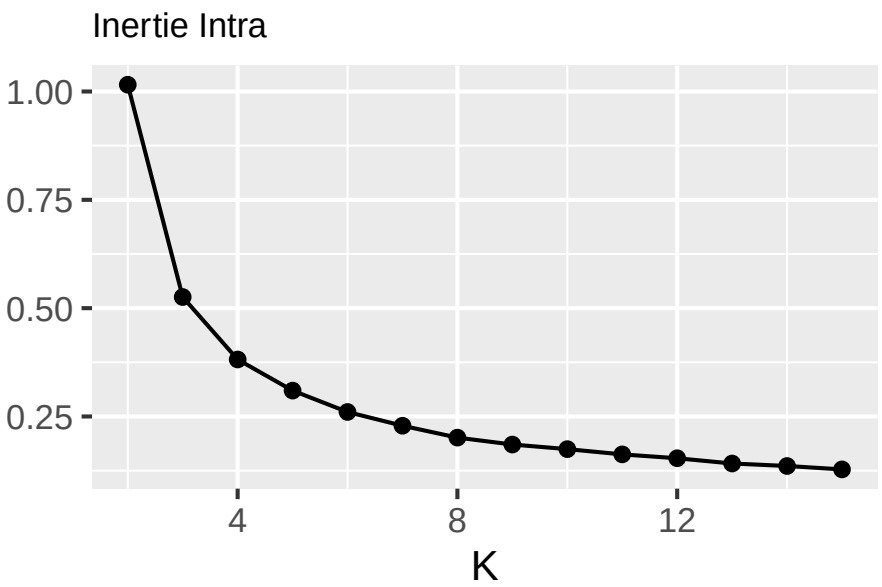
- Pour chaque K dans $\{1, \ldots, K_{\max}\}$:
  - Calculer la matrice de dispersion intra-classe
    $W_K = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in C_k} (x_i - m_k)'(x_i - m_k)$ et son déterminant $\det(W_K)$
  - Construire la courbe des $\ln(det(W_K))$ en fonction de K
  - Comparer cette courbe à celle obtenue à partir des données uniformément réparties
- L'estimation du nombre de classes à retenir est la valeur K correspondant au plus grand écart entre les deux courbes
- Avec  , `clusGap()` [library(cluster)]
- Avec  , github `milesgranger/gap_statistic`

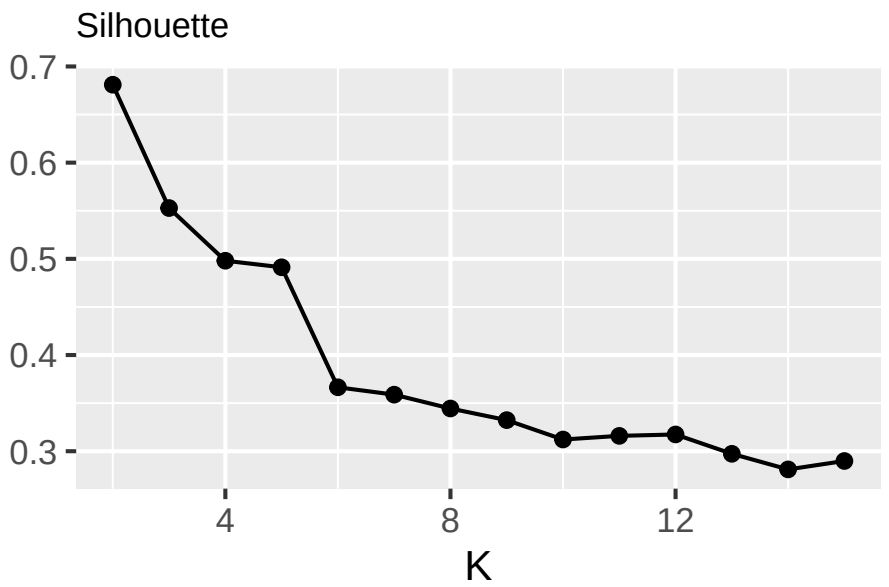
Exemple - Gap Statistic



Exemple des Iris







Plan

Extensions des K-means

- 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] : modif. de la dissimilarité et les modes remplacent les centres de gravité
- Pour variables **mixtes** : Algorithme K-**prototype** [Huang, 97]
- Fuzzy c-means

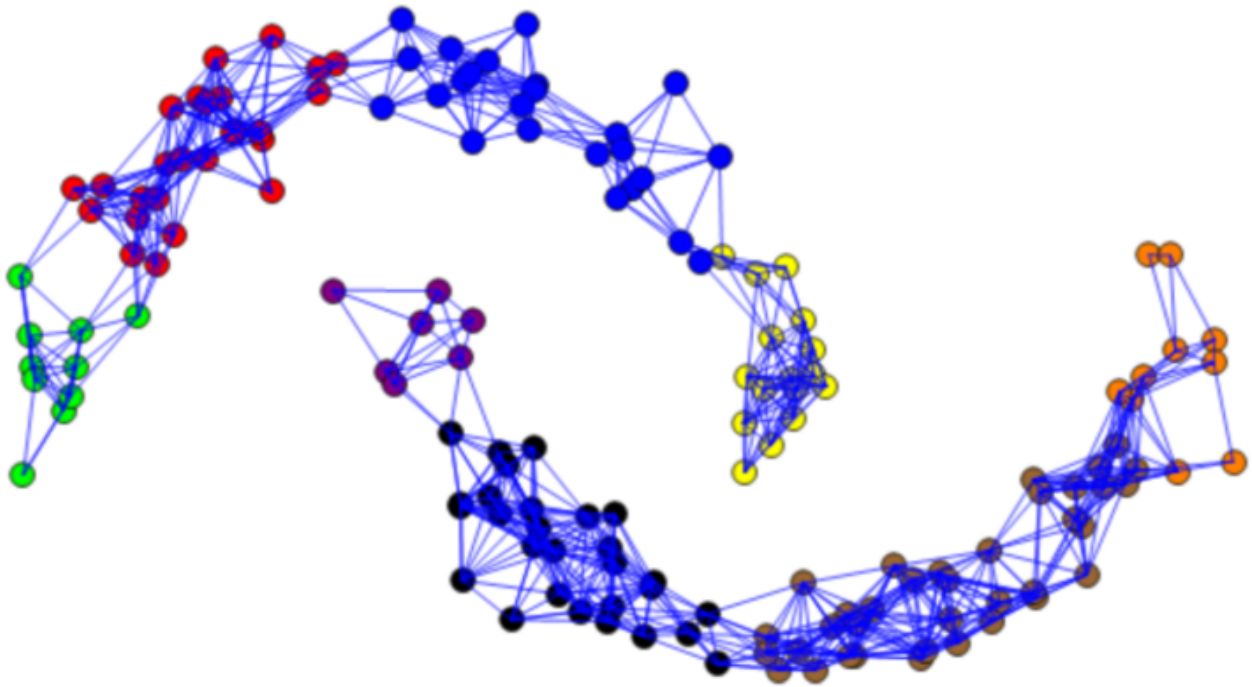
Méthodes de type K-médoides

- On remplace les centres de gravité par des médoïdes (des points de X)
- **PAM (Partitioning Around Medoids)** [Kaufman & Rousseeuw,90]
- 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 de bruit ou d'outliers (un médoïde est moins influencé par un outlier que la moyenne)

Plan

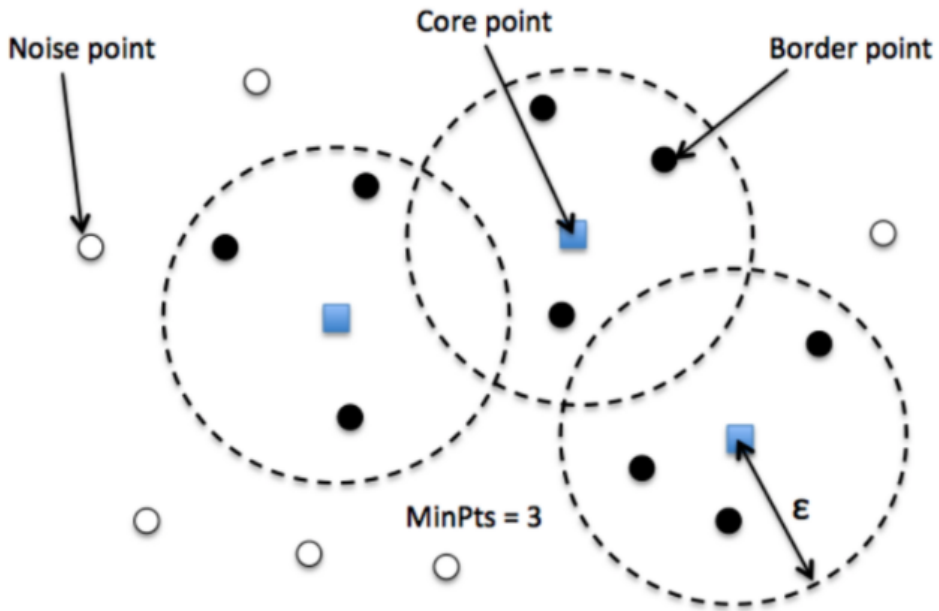
Spectral clustering

- Construction d'un graphe pondéré avec noeud=individu et S matrice de similarité
  ex: graphe de voisinage, KNN, graphe totalement connecté
  ex pour S : noyau gaussien, distance cosinus, ...
- A partir de S, décomposition spectrale d'une matrice Laplacienne L
- Partitionnement des données dans le sous-espace

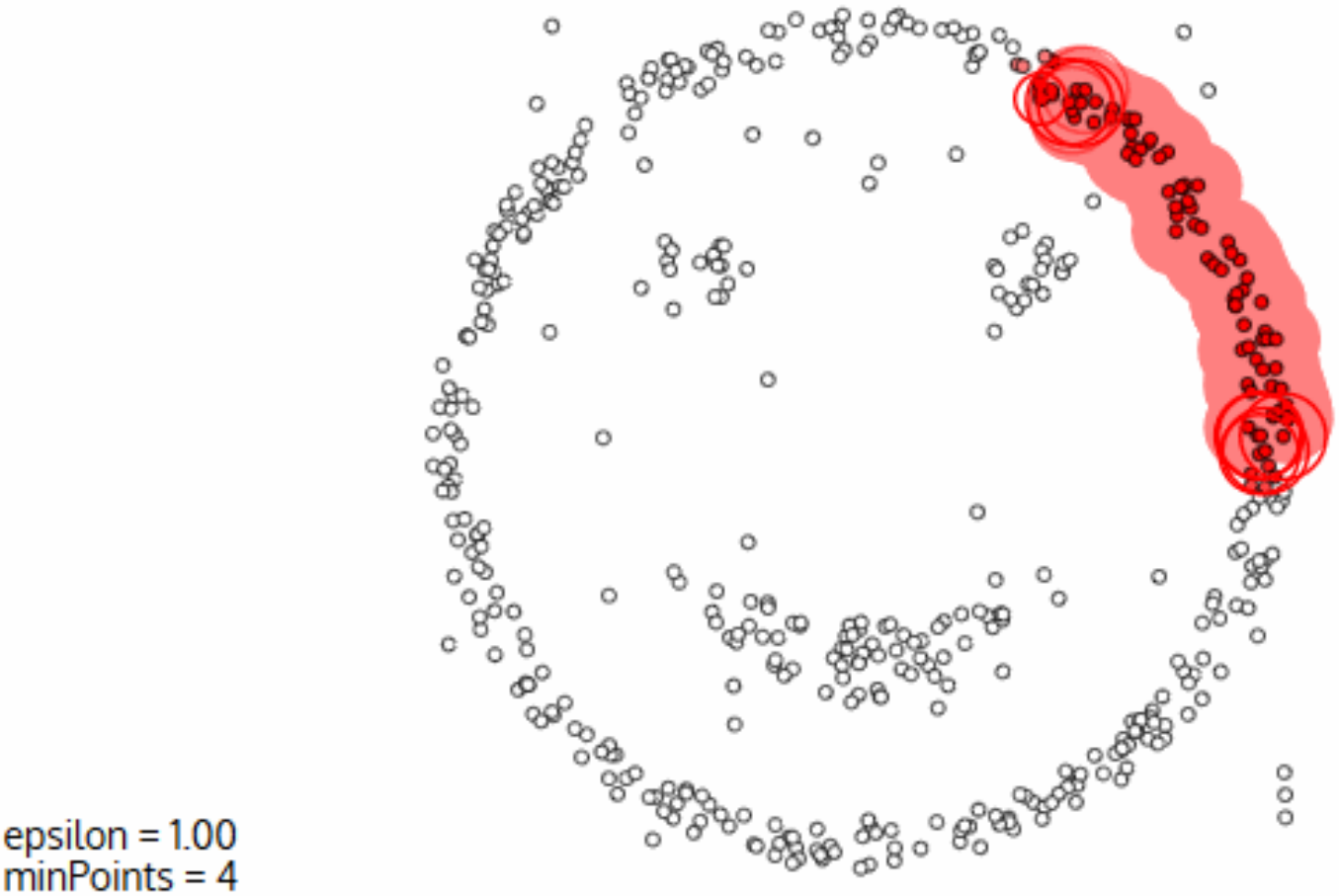


DBSCAN

- DBSCAN = Density-Based Spatial Clustering of Applications
- 2 paramètres :
  - la distance de voisinage ε
  - le nombre minimum de points minPts



DBSCAN



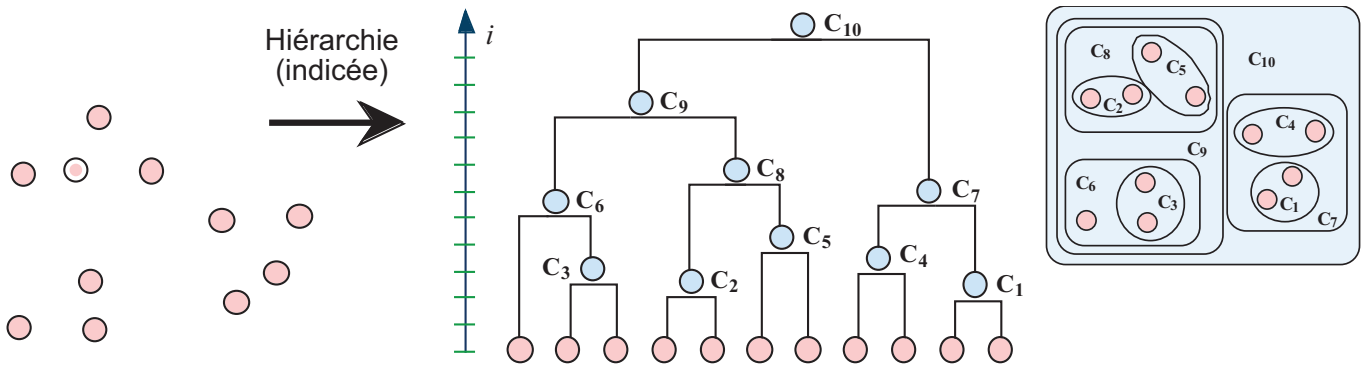
https://aaronscotthq.com/2020-05-28-scott_dbscan/

Plan

Plan

Algorithme général de CAH

- Initialisation : $\mathcal{P}_n = \{\{x_1\}, \ldots, \{x_n\}\}$
- Étapes agrégatives :
  - on part de la partition précédente $\mathcal{P}_K = \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ en K classes
  - on agrège les deux classes $\mathcal{C}_k$ et $\mathcal{C}_{k'}$ qui minimisent une **mesure d'agrégation** $D(\mathcal{C}_k, \mathcal{C}_{k'})$: $\mathcal{C}_{k \cup k'} = \mathcal{C}_k \cup \mathcal{C}_{k'}$ on obtient ainsi une partition en $K - 1$ classes
- On recommence l'étape d'agrégation jusqu'à obtenir une partition en une seule classe



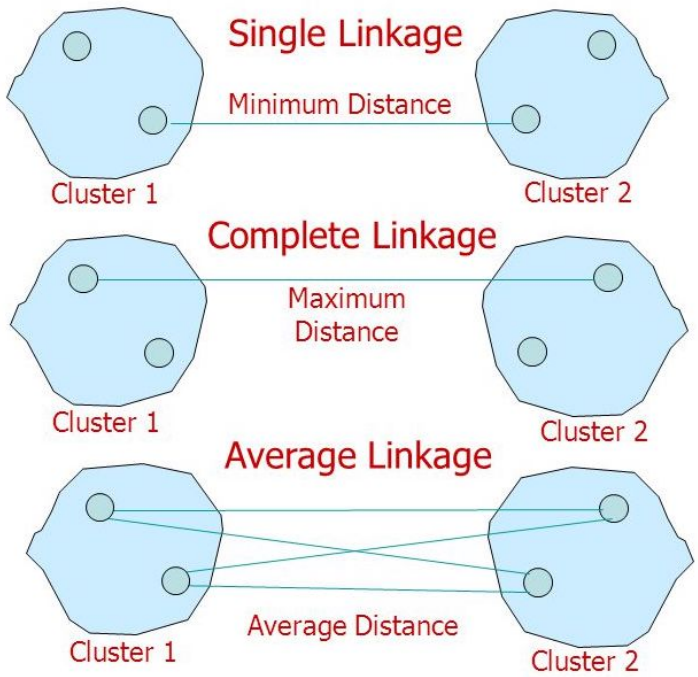
Les choix à faire

- Choix d'une **dissimilarité** d entre les points
- Choix d'une **mesure d'agrégation** D entre classes
- Construction d'un dendrogramme
- **Critère pour la coupure du dendrogramme** pour en déduire une classification des données

Plan

Single / Complete / Average linkage

- **Lien simple** (*Single linkage*)
  $D(\mathcal{C}_k, \mathcal{C}_{k'}) = \min_{i \in \mathcal{C}_k, \ell \in \mathcal{C}_{k'}} d(x_i, x_\ell)$
- **Lien complet** (*Complete linkage*)
  $D(\mathcal{C}_k, \mathcal{C}_{k'}) = \max_{i \in \mathcal{C}_k, \ell \in \mathcal{C}_{k'}} d(x_i, x_\ell)$
- **Lien moyen** (*Average linkage*)
  $D(\mathcal{C}_k, \mathcal{C}_{k'}) = \frac{1}{|\mathcal{C}_k||\mathcal{C}_{k'}|} \sum_{i \in \mathcal{C}_k} \sum_{\ell \in \mathcal{C}_{k'}} d(x_i, x_\ell)$



Mesures d'agrégation de Ward

- **Mesure d'agrégation de Ward**

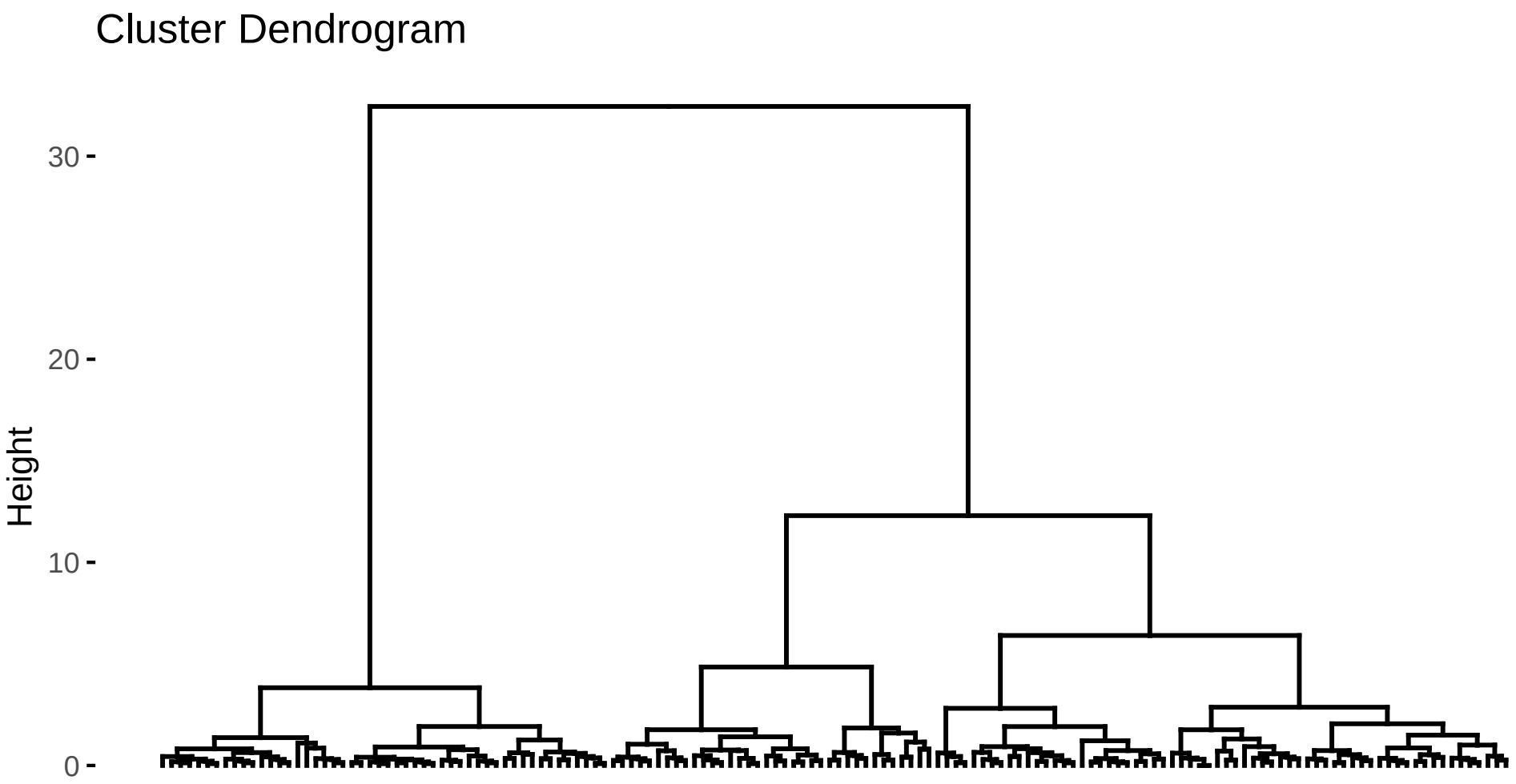
$$D(\mathcal{C}_k, \mathcal{C}_{k'}) = \frac{|\mathcal{C}_k||\mathcal{C}_{k'}|}{n(|\mathcal{C}_k| + |\mathcal{C}_{k'}|)} \, d(m_k, m_{k'})^2$$

  - m_k (resp. $m_{k'}$) centre de gravité de $\mathcal{C}_k$ (resp. $\mathcal{C}_{k'}$)
  - d distance euclidienne

- **Méthode de Ward** : Elle consiste à choisir à chaque étape les deux classes dont le regroupement implique une augmentation minimale de l'inertie intraclasse.

Exemple des iris

```
dx<-dist(iris[,-5],method="euclidian")
hward<-hclust(dx,method="ward.D2")
fviz_dend(hward,show_labels = FALSE)
```

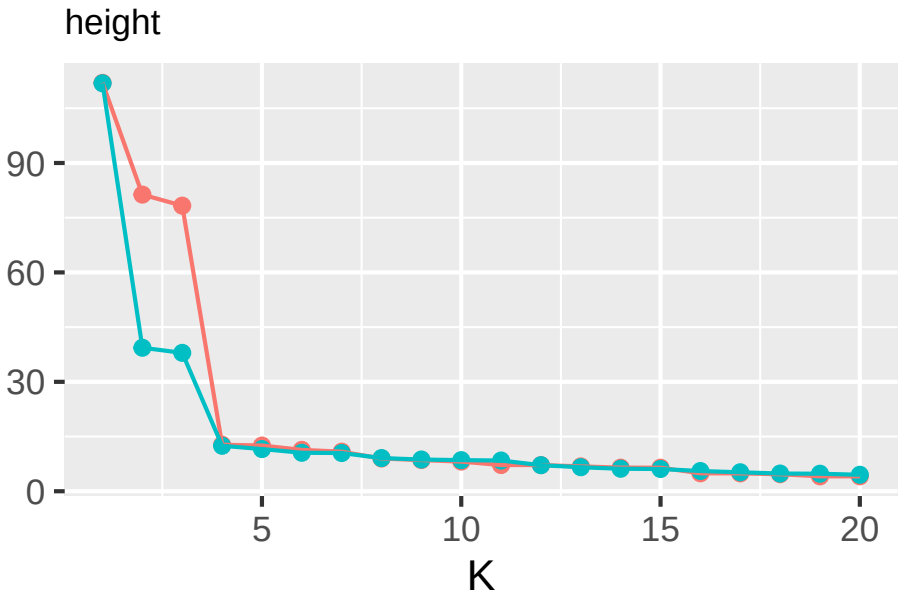


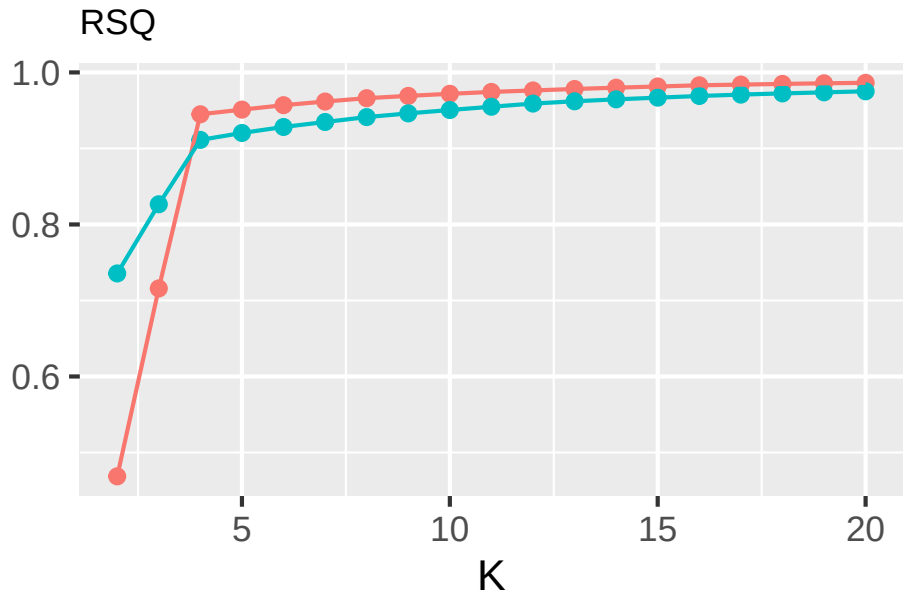
Plan

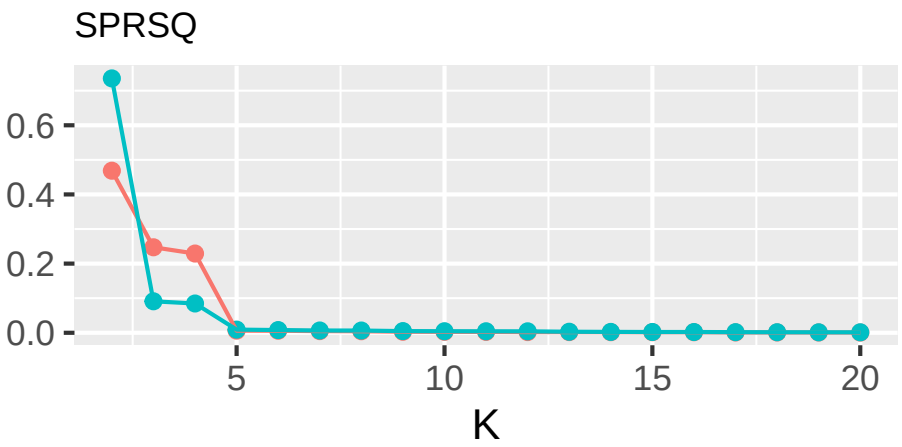
Comment faire ?

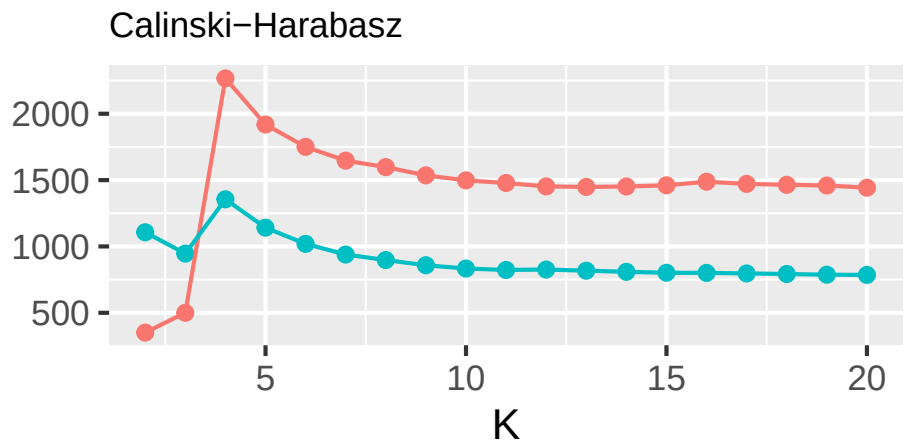
- Le choix du niveau de coupure du dendrogramme détermine le nombre de classes et ces classes sont alors uniques
- On peut définir la coupure du dendrogramme en déterminant à l'avance le nombre de classes dans lesquelles on désire répartir l'ensemble des données
- Le choix du niveau de coupure peut être facilité par l'examen des indices croissants de niveau de l'arbre hiérarchique
- On peut aussi faire ce choix en utilisant les indices tels que R^2, CH, Silhouette, Gap Statistic, ...

Exemple des données simulées

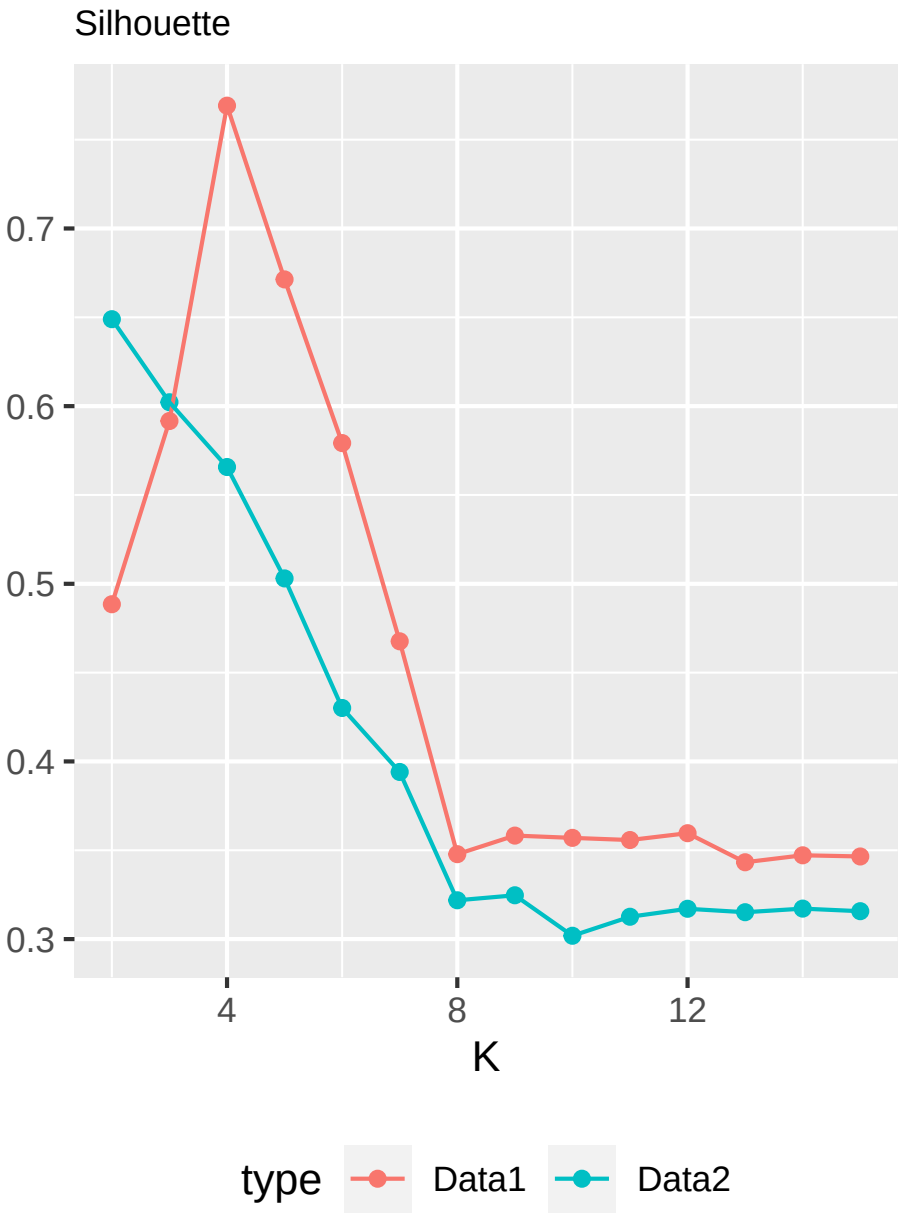


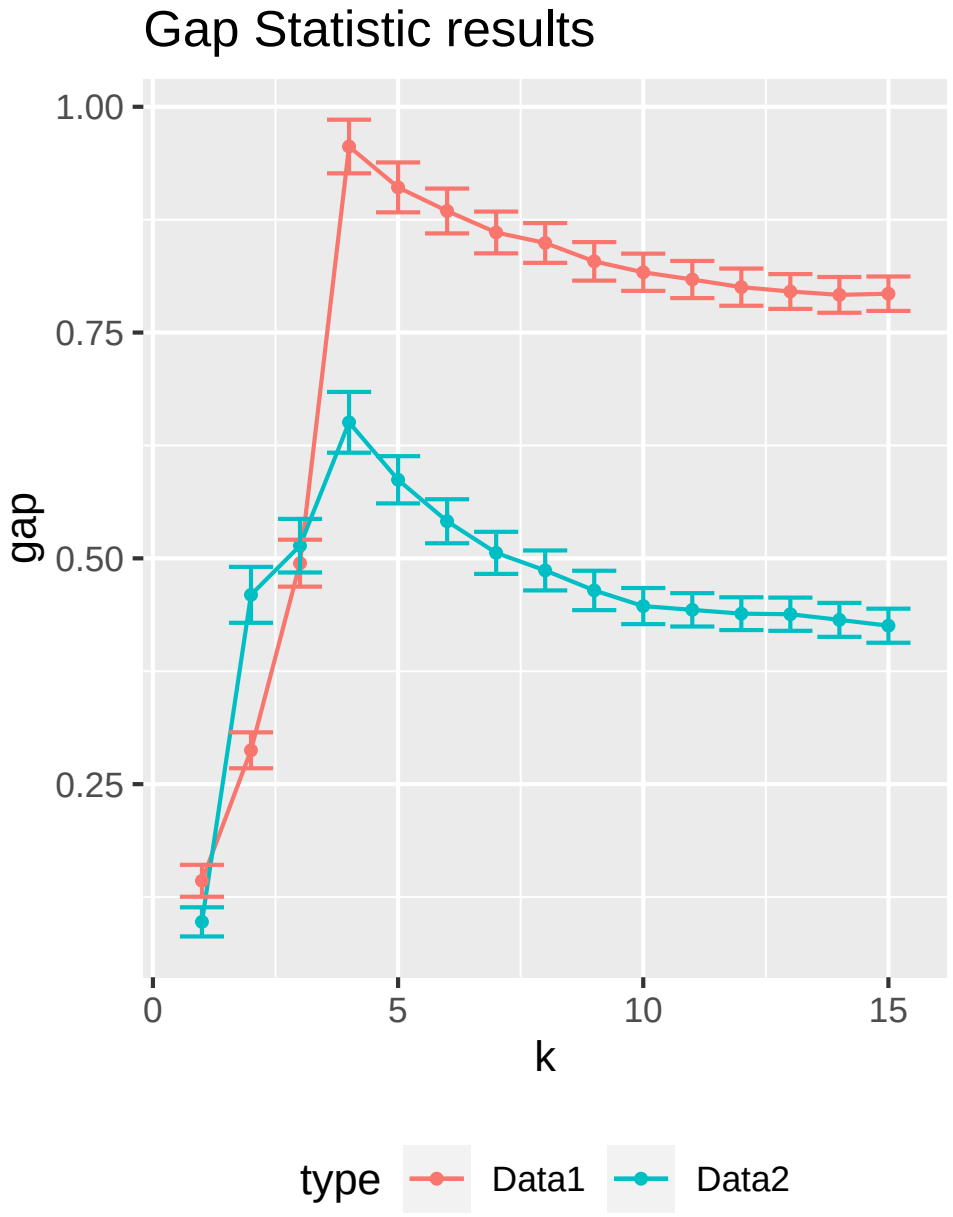




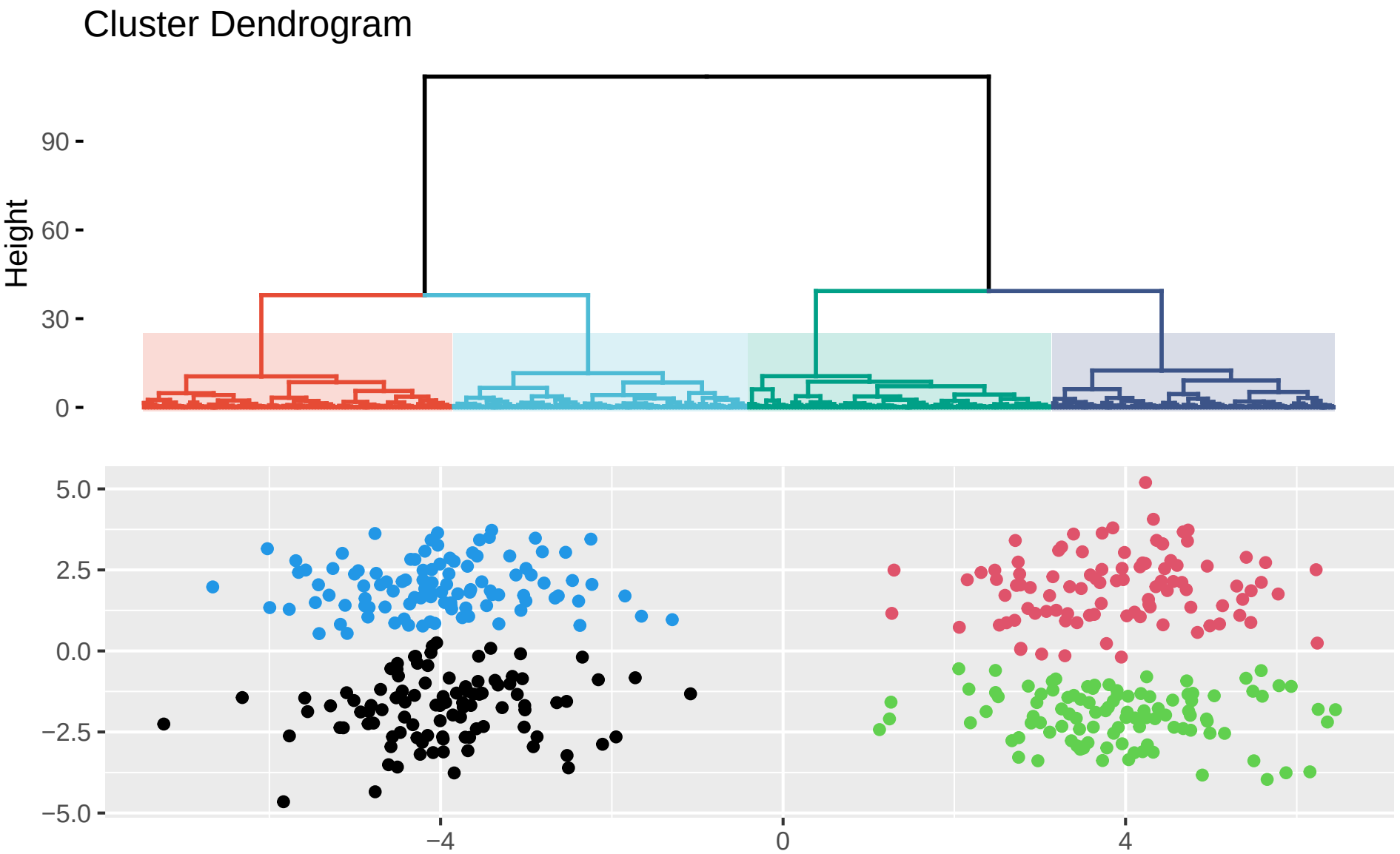


Exemple des données simulées





Exemple des données simulées



Plan

Avantages et inconvénients CAH

- Avantages :
  - Méthode flexible pour le niveau de finesse de la classification
  - Prise en compte facile de distances et d'indices de similarité de n'importe quel type
  - Rapidité d'execution et reproductible
- Inconvénients :
  - Choix de la coupure de l'arbre
  - La partition obtenue à une étape dépend de celle à l'étape précédente

Quelques commandes

- Avec R 
  - `hc=hclust(d,method=)`
    - *d* : tableau de distances comme produit par `dist()`
    - *method* : agrégation “ward.D2”,“single”,“complete”, “average”, . . .
  - `plot(hc,hang=,...)` ou `ggdendrogram(hc,...)` ou `fviz_dend()` pour tracer le dendrogramme
  - `cutree(hc,k=..)` pour obtenir la classification en *k* classes
- Avec Python 

scipy.cluster.hierarchy [scipy]

- `linkage(.)` : method=‘single’,‘complete’,‘average’,‘ward’,. . .
- `dendrogram(.)` : pour tracer le dendrogramme
- `fcluster(.)` : pour obtenir un clustering à partir du dendrogramme

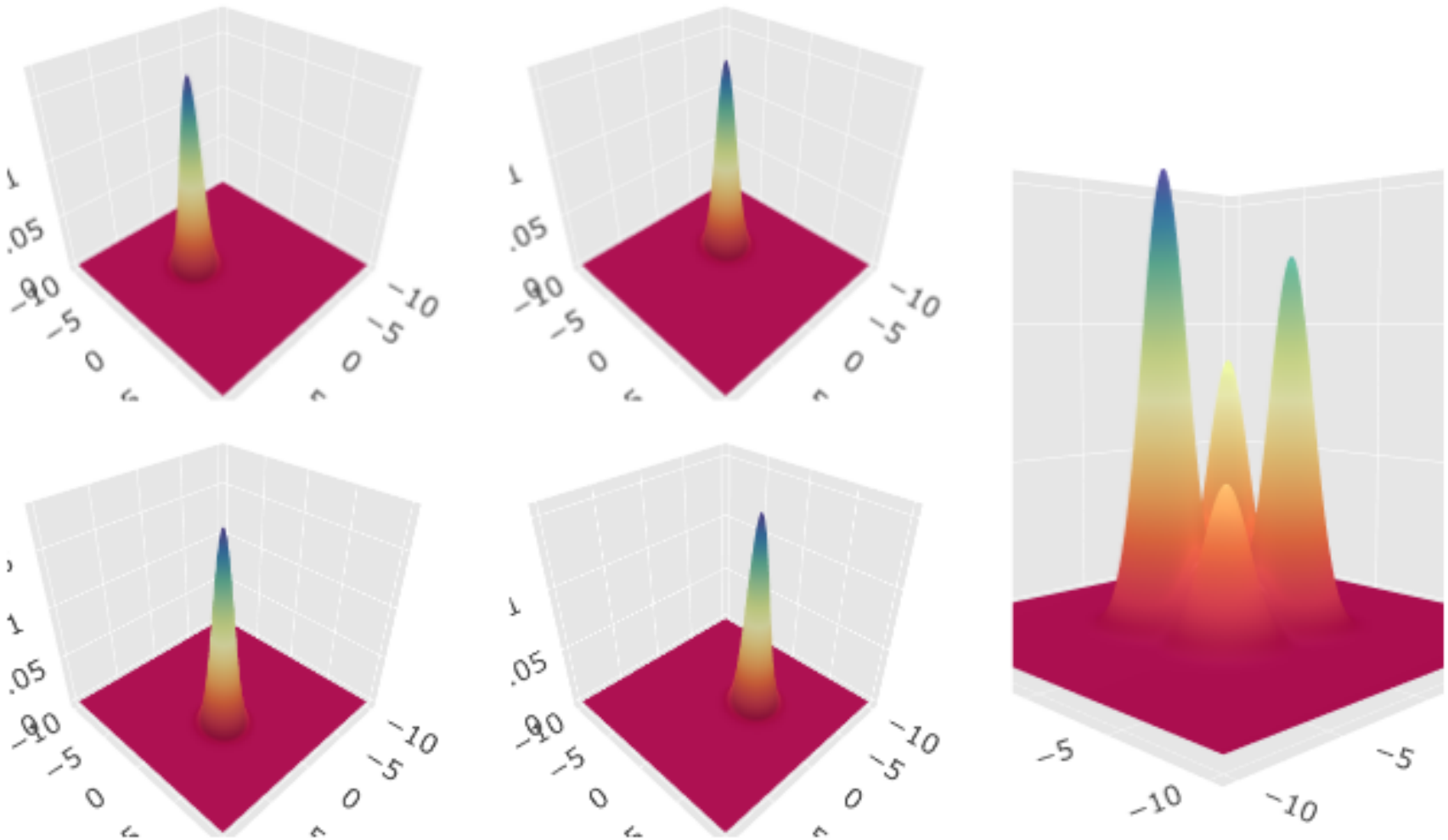
Plan

Plan

Hypothèses des mélanges finis

- On suppose que les données proviennent d'une population contenant plusieurs sous-populations
- Chaque sous-population est modélisée indépendamment des autres (**choix d'une loi de distribution pour chaque sous-population**).
- La population totale est alors vue comme un mélange de ces sous-populations. Le modèle résultant est un **modèle de mélange fini**.
- C'est une approche probabiliste !

Mélanges finis



Mélanges finis

- Un modèle de mélange à K composantes est de la forme

$$f(.|\theta_K) = \sum_{k=1}^{K} \pi_k f_k(.|\alpha_k)$$

  - $(\pi_1, \ldots, \pi_K)$ sont les proportions du mélange

$$\forall k \in \{1, \ldots, K\},\ \pi_k \in [0,1] \text{ et } \sum_{k=1}^{K} \pi_k = 1$$

  - $f_k(.|\alpha_k)$ est la densité de la $k$ème sous-population
  - $\theta_K = (\pi_1, \ldots, \pi_K, \alpha_1, \ldots, \alpha_K)$

- Le choix des f_k dépend de la nature des données

Les grandes étapes

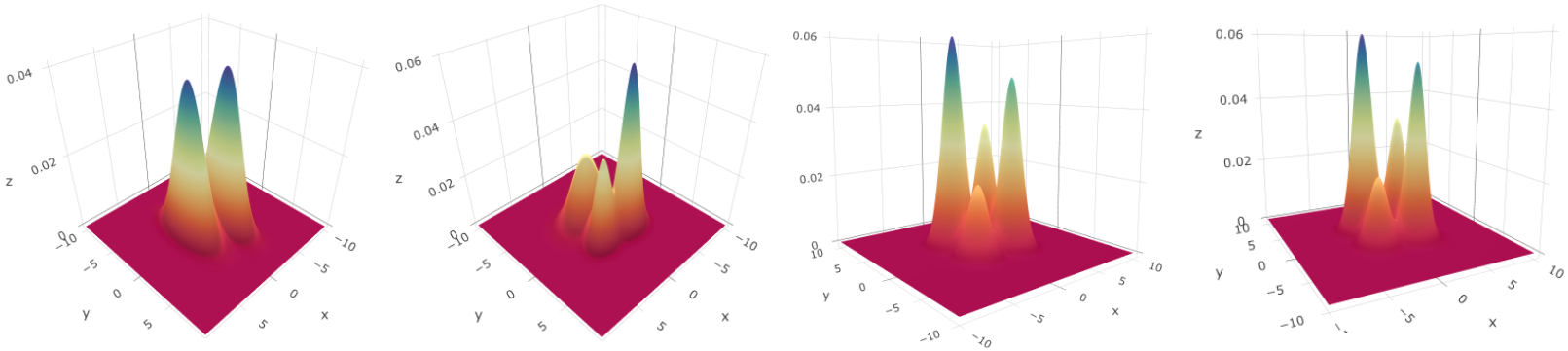
1. Collection de modèles :

$$\forall K \in \mathbb{N}^\star,\ \mathcal{S}_K = \left\{ x \in \mathbb{R}^p \mapsto f(x|\theta_K) = \sum_{k=1}^{K} \pi_k f_k(.|\alpha_k) \right\}$$

Mélanges à K=2 classes | **Mélanges à K=3 classes** | **Mélanges à K=4 classes** | **Mélanges à K=5 classes** | ...

$\Rightarrow$ Choix initial de la modélisation

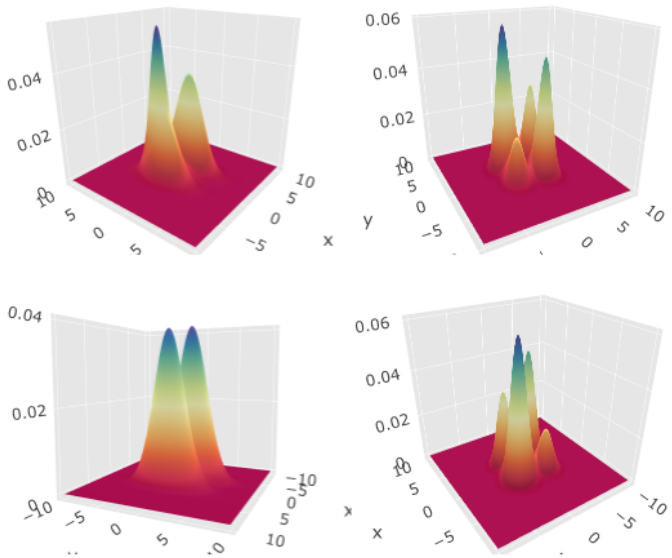
2. Dans chaque modèle $\mathcal{S}_K$: on détermine le mélange qui s'ajuste le mieux aux données: $f(.|\hat{\theta}_K)$

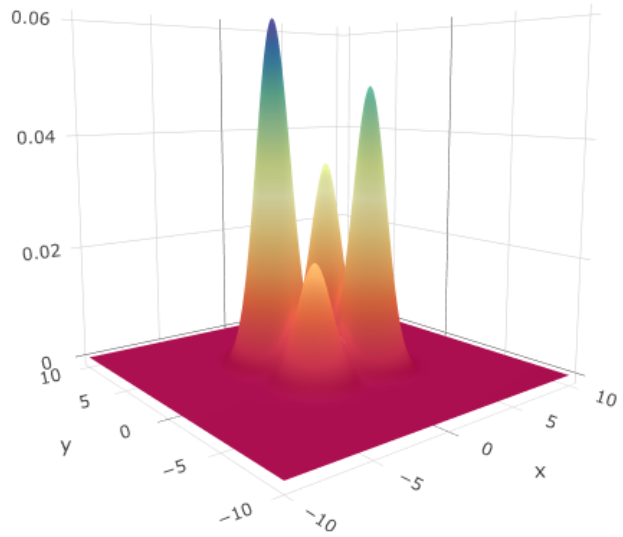


$\Rightarrow$ Besoin d'un algorithme d'estimation des paramètres $(\hat{\theta}_K)$

Les grandes étapes

3. Choisir le "meilleur" mélange parmi $f(.|\hat{\theta}_2), f(.|\hat{\theta}_3), \ldots, f(.|\hat{\theta}_{K_{\max}})$





$\Rightarrow$ Besoin d'un critère de sélection de modèles pour déterminer $\hat{K}$ et donc choisir $f(.|\hat{\theta}_{\hat{K}})$.

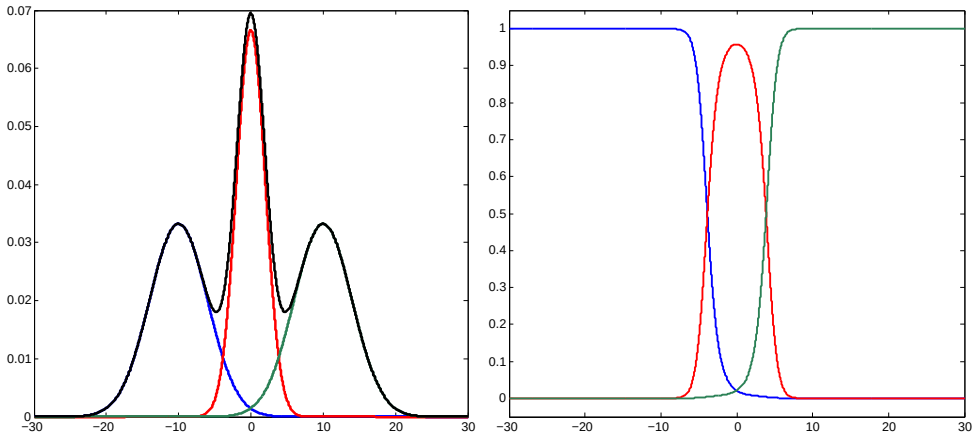
4. Règle du "MAP" pour en déduire une classification des données

Etape 4 : Règle du MAP

- Principe : chaque individu est affecté à la classe pour laquelle il a la plus forte probabilité d'appartenance conditionnellement à l'estimation des paramètres

- Probabilité conditionnelle d'appartenance de i à $\mathcal{C}_k$ avec le vecteur de paramètres θ

$$\begin{aligned} t_{ik}(\theta) &= P(z_{ik} = 1 | x_i, \theta) \\ &= \frac{\pi_k f_k(x_i|\alpha_k)}{\sum_{\ell=1}^{K} \pi_\ell f_\ell(x_i|\alpha_\ell)} \end{aligned}$$

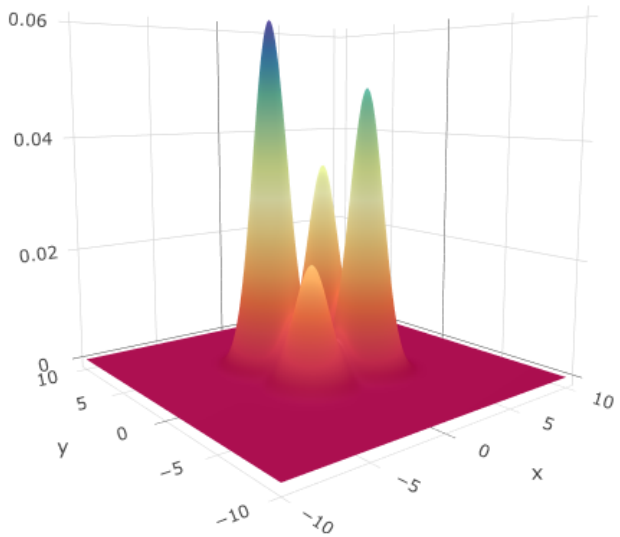


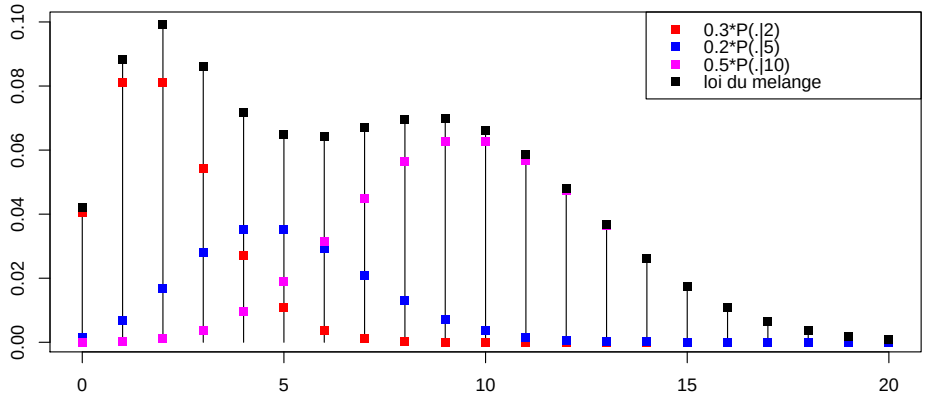
- Règle du maximum a posteriori (MAP) avec $\hat{\theta}_{\hat{K}}$:

$$i \in \mathcal{C}_k \text{ si } t_{ik}\left(\hat{\theta}_{\hat{K}}\right) > t_{i\ell}\left(\hat{\theta}_{\hat{K}}\right) \; \forall \ell \neq k$$

Etape 1 : Choix des distributions

- Données quantitatives : mélanges gaussiens, mélanges de Student, ...
- Données de comptages : mélanges de Poisson, de binomiales négatives, ...
- Données qualitatives : mélanges de multinomiales, ...
- Données compositionnelles : mélanges de Dirichlet, ...





Etape 2 : Estimation du maximum de vraisemblance

- On désire déterminer le vecteur des paramètres $\theta_K = (\pi_1, \ldots, \pi_K, \alpha_1, \ldots, \alpha_K)$ qui **maximise la logvraisemblance** :

$$\mathcal{L}(\underline{\mathbf{x}}|\theta_K) = \mathcal{L}(x_1, \ldots, x_n|\theta_K) = \sum_{i=1}^{n} \ln \left[\sum_{k=1}^{K} \pi_k f_k(x_i|\alpha_k) \right]$$

- Ce problème de maximisation ne possède généralement pas de solution analytique

- $\Rightarrow$ Algorithme d'optimisation pour approcher $\hat{\theta}_K$

Algorithmes itératifs de type EM

- Algorithme EM [Dempster et al.,77]

EM = Expectation Maximization

- Algorithme CEM [Celeux and Govaert, 92]

CEM = Classification EM

- Algorithme SEM [Celeux and Diebolt, 85]

SEM = Stochastique EM

- Algorithme SAEM [Delyon et al., 99]

SAEM = Stochastic Approximation EM

- Algorithme MCEM [Wei and Tanner, 90]

MCEM = Monte Carlo EM

Etape 3 : Critères asymptotiques de sélection

$$\hat{K} = \underset{K}{argmin}\ \text{crit}(K) = \underset{K}{argmin}\ -\mathcal{L}(\underline{\mathbf{x}}|\hat{\theta}_K) + \text{pen}(K)$$

- **AIC** (Akaike Information Criterion) [Akaike, 73]
$$\text{crit}(K) = -\mathcal{L}(\underline{\mathbf{x}}|\hat{\theta}_K) + \nu_K$$

- **BIC** (Bayesian Information Criterion) [Schwarz, 78]
$$\text{crit}(K) = -\mathcal{L}(\underline{\mathbf{x}}|\hat{\theta}_K) + \frac{\nu_K}{2}\ln(n)$$

- **ICL** (Integrated Completed Likelihood) [Biernacki et al., 00]
$$\text{crit}(K) = -\mathcal{L}(\underline{\mathbf{x}}|\hat{\theta}_K) + \frac{\nu_K}{2}\ln(n) + \text{Ent}(K)$$

où ν_K est nombre de paramètres libres des mélanges de $\mathcal{S}_K$

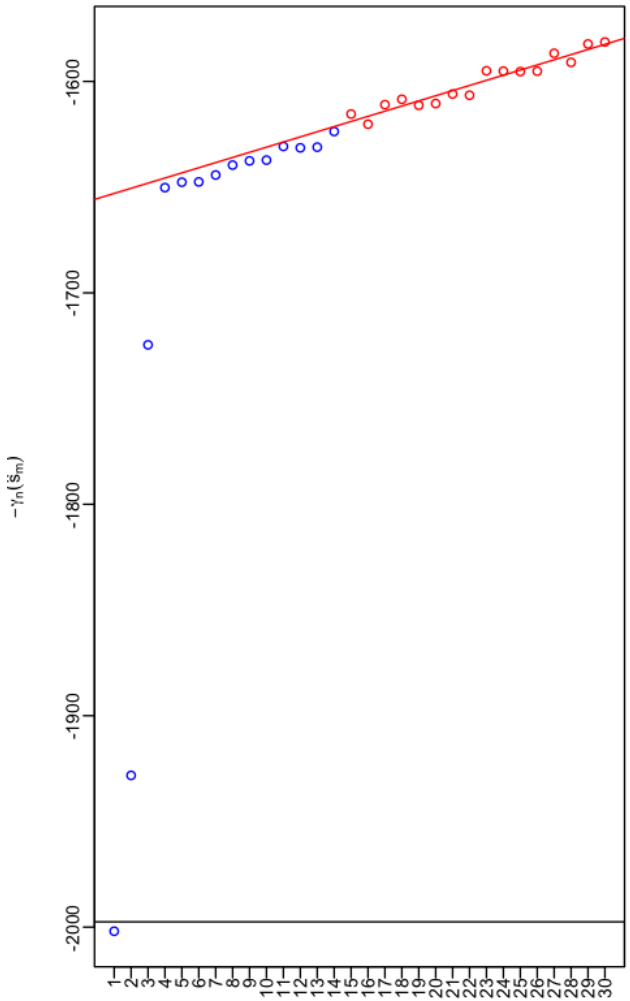
$\text{Ent}(K) = -\sum_{i=1}^{n}\sum_{k=1}^{K} \hat{z}_{ik}\ln[t_{ik}(\hat{\theta}_K)]$ est un terme d'entropie

Etape 3 : Alternative avec l'heuristique de pente

- Critère non-asymptotique

$$\hat{K} = \underset{K}{argmin}\ \text{crit}(K) = \underset{K}{argmin} -\mathcal{L}(\underline{\mathbf{x}}|\hat{\theta}_K) + 2\kappa D_K$$

- Nécessite de déterminer la quantité D_K qui représente la dimension du modèle $\mathcal{S}_K$
- Calibration de la pente inconnue κ



Plan

Mélanges gaussiens multivariés

- Données quantitatives : $x_i \in \mathbb{R}^p$
- Les observations sont supposées être un échantillon de loi

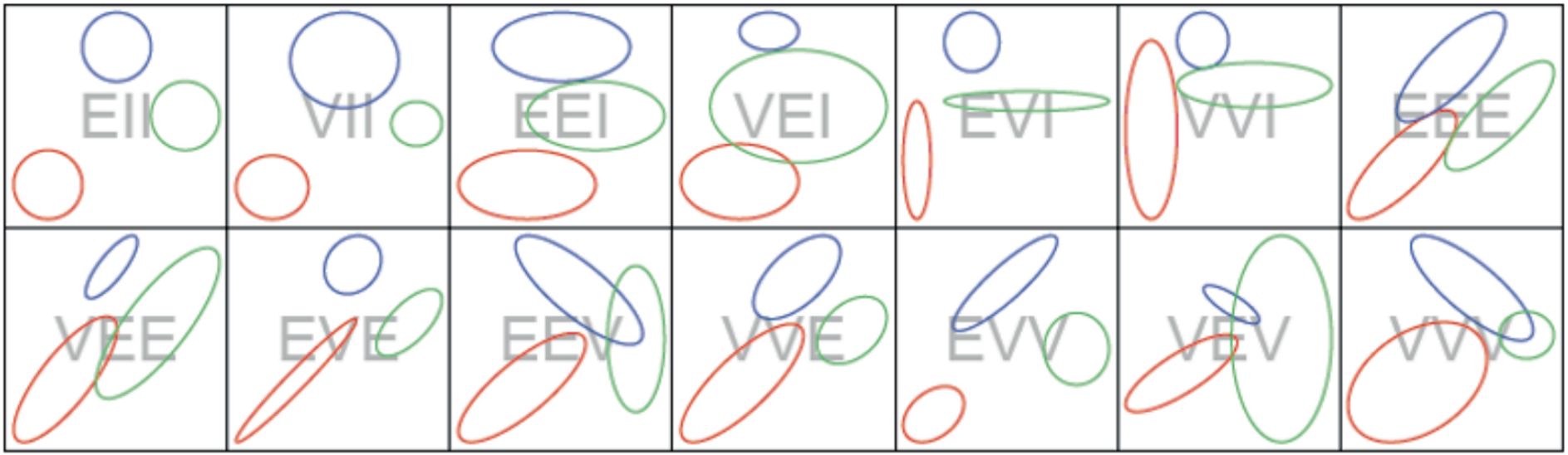
$$f(.|\theta_{K,m}) = \sum_{k=1}^{K} \pi_{k,m}\ \phi(.|\mu_k, \Sigma_{k,m})$$

  - $\phi(.|\mu_k, \Sigma_k)$ la densité d'une loi gaussienne multivariée $\mathcal{N}_p(\mu_k, \Sigma_k)$
  - $\theta_{K,m} = (\pi_{1,m}, \ldots, \pi_{K,m}, \mu_1, \ldots, \mu_K, \Sigma_{1,m}, \ldots, \Sigma_{K,m})$
- Collection de modèles: $\left(\mathcal{S}_{(K,m)}\right)_{(K,m)\in\mathbb{N}^\star\times\mathcal{M}}$ avec
  - $\mathcal{S}_{(K,m)} = \{x \in \mathbb{R}^p \mapsto f(x|\theta_{K,m}); \theta_{K,m} \in \Theta_{K,m}\}$
  - $\mathcal{M}$ = ensemble de formes de mélanges gaussiens

Formes m des mélanges gaussiens

- 14 formes (contraintes sur la décomposition en valeurs propres des Σ_k) réparties en 3 familles (sphérique, diagonale, générale)
  $\Rightarrow$ formes, volumes et orientations des ellipsoides
- Proportions supposées égales ou libres

$\Rightarrow$ 28 formes de mélanges gaussiens possibles



Bouveyron et al. [4]

Lien entre GMM et Kmeans

La méthode des Kmeans est un cas particulier des GMM :

- Collection de modèles : mélanges gaussiens sphériques

+

- Estimer les paramètres avec l'algorithme CEM

On peut alors utiliser les critères de sélection de modèles (Etape 3) pour choisir le nombre de classes.

Plan

Quelques librairies avec 

- Librairie `mclust` [Scrucca et al.]

mixtures of multivariate Gaussian

- Librairie `Rmixmod` [Biernacki et al.]

mixtures of multivariate Gaussian or multinomial components

- Librairie `mixture` [McNicholas P.D. et al.]

Gaussian, Student's t, generalized hyperbolic, variance-gamma or skew-t mixtures

- Librairie `movMF` [Horbik and Grün]

mixtures of von Mises-Fisher distributions

- et bien d'autres !

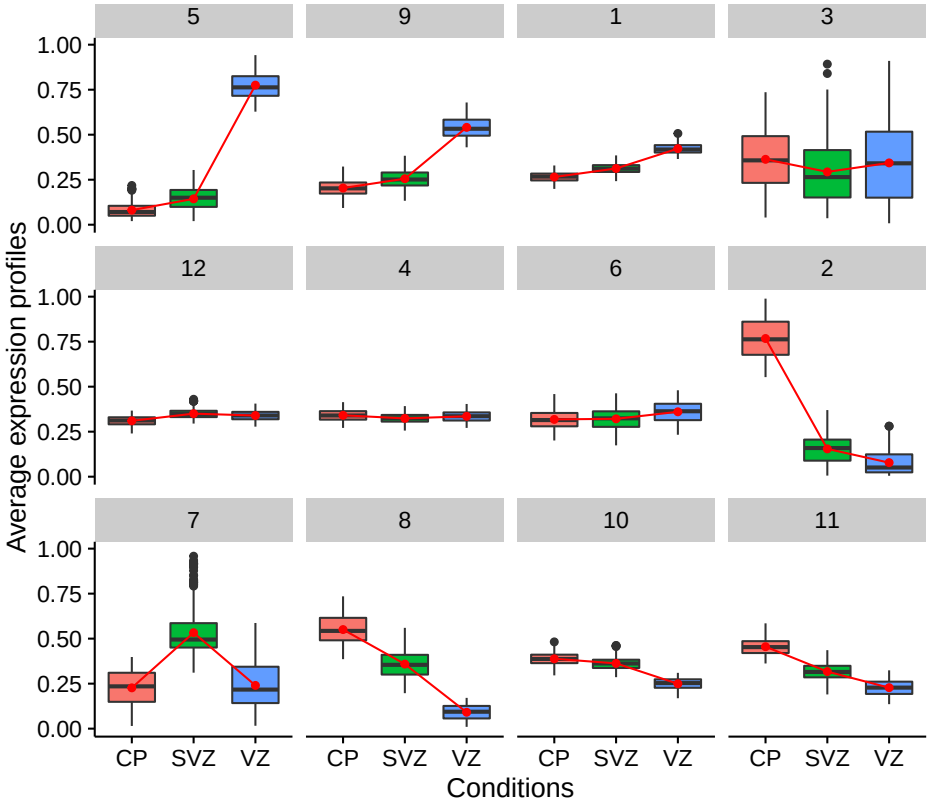
Et avec 

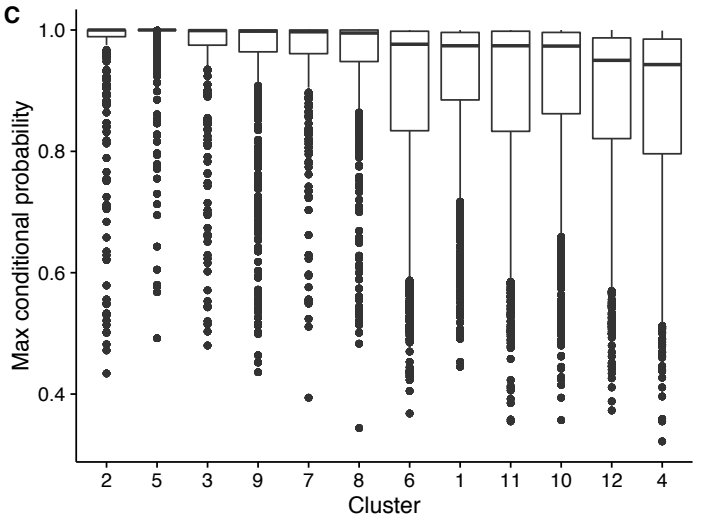
- `sklearn.mixture.GaussianMixture(.)`
- Package `PyMix` [Georgi et al., http://www.pymix.org/]
- Package `Pymixmod` (la version python de mixmod)
- `Multinomial_Mixture_Model` [https://github.com/diningphil/Multinomial_Mixture_Model]
- `studenttmixture` [https://github.com/jlparki/mix_T]
- ...

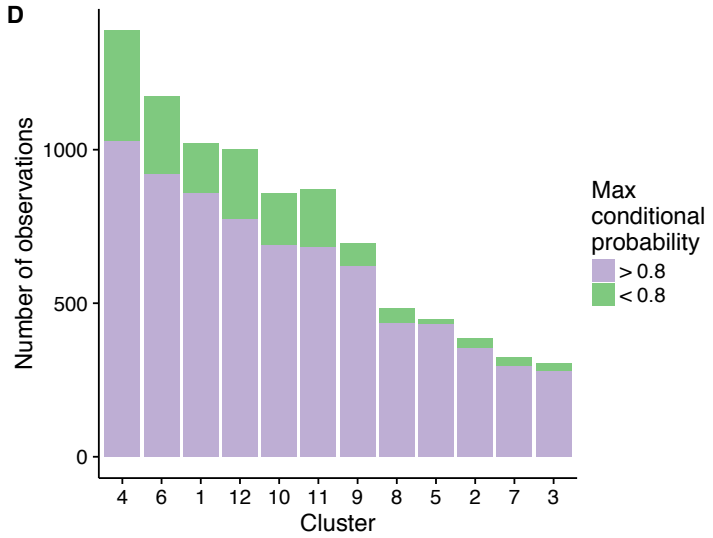
Exemple pour la co-expression

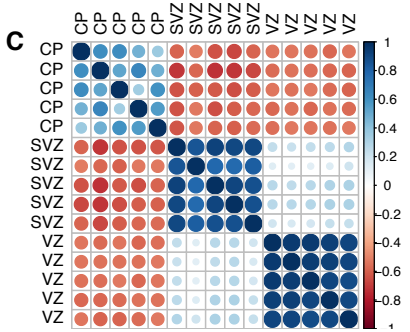
- Objectif : mettre en évidence des sous-groupes de gènes co-exprimés à partir de données transcriptomiques
- Mélanges gaussiens sur des données de puces à ADN
- Sur des données RNA-seq (comptages), mélanges de lois de Poisson [Rau et al., 15], de binomiales négatives [Li et al, 21], ...
- Reformulation du problème de données RNA-seq à des données compositionnelles
  - Transformations logit ou arcsin + mélanges gaussiens [Rau and Maugis-Rabusseau, 18]
  - Transformation de type CLR + Kmeans [Godichon-Baggioni et al, 19]

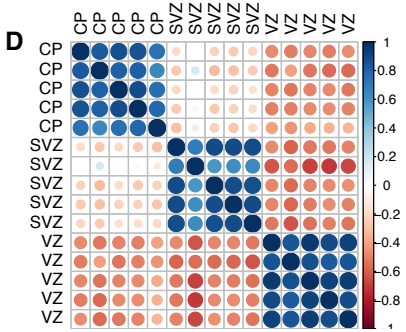
Exemple pour la co-expression











Plan

Principe

Méthodes basées sur 3 ingrédients

- Deep Neural Network (DNN)

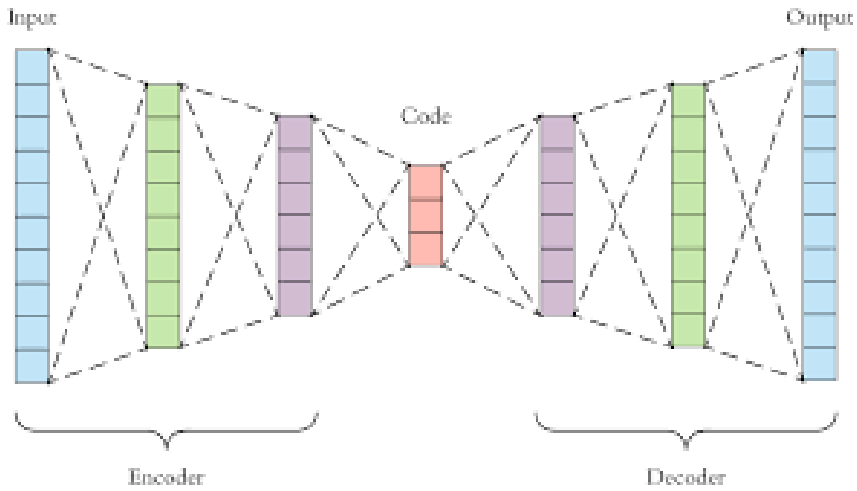
ex : autoencoder, VAE, GAN

- Network loss function L_R

ex : reconstruction loss of AE, ...

- Clustering loss L_C

ex : k-means loss, clustering assigment hardening (KL), ...



$$L = \alpha L_R + (1 - \alpha) L_C \text{ avec } \alpha \in [0, 1]$$

Exemples de méthodes

Articles de review : Aljalbout et al. (2018), Karim et al. (2021)

Method	Arch	Features for clust.	L_R	L_C	Combin.	Clust algo
JULE Yang et al. (16)	CNN	CNN output	-	Agglom. loss	-	Agglom. clust.
DEC Xie et al. (16)	MLP	Encoder output	AE reconst. loss	Cluster assign. hardening	Pretrain. fine-tuning	Network estim. centroids soft assign.
SCCNN Lukie et al. (16)	CNN	CNN output	Classif. loss	-	-	usual clust. methods
DCN Yang et al. (16)	MLP	Encoder output	AE reconst. loss	K-means loss	Altern. joint training and cluster updates	K-means
VaDE Zheng et al (16)	VAE	Encoder output	VAE loss	+ mixture model	-	Network estim. centroids
⋮	⋮	⋮	⋮	⋮	⋮	⋮

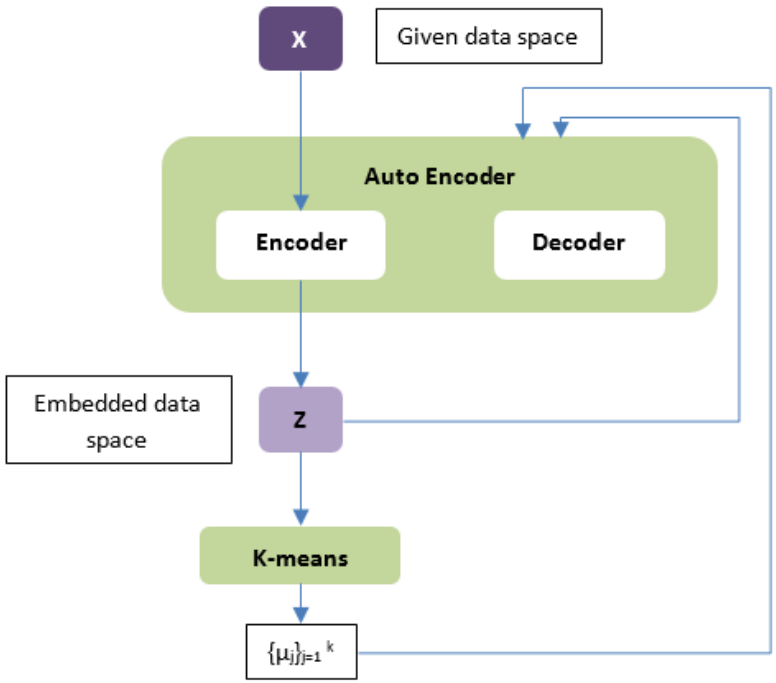
Deep Embedded Clustering (DEC)

[Xie, Girshick and Farhadi (2016)]

- Perceptron multicouche - Autoencoder (AE)
- Fonction de perte de reconstr. de l'AE $L_R = \sum_{i=1}^{n} \|x_i - f(x_i)\|^2$ avec $f(x_i)$ valeur de reconstr. de l'entrée x_i
- Fonction de perte du clustering

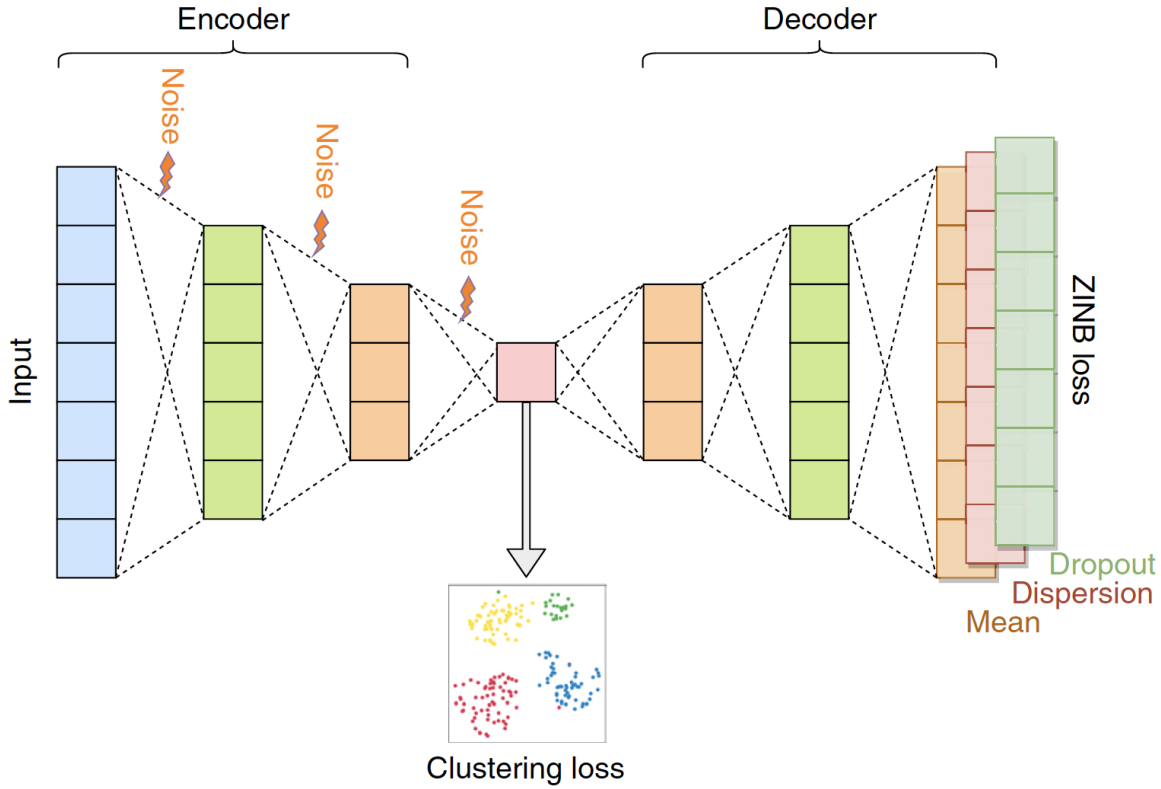
$$L_C = KL(P||Q) = \sum_i \sum_k p_{ik} \ln \left[\frac{p_{ik}}{q_{ik}} \right]$$

- $q_{ik} = \frac{(1+\|z_i - \mu_k\|^2/\alpha)^{-\frac{1+\alpha}{2}}}{\sum_{k'} \square_{k'}}$
- Distribution auxiliaire $p_{ik} = \frac{q_{ik}^2 / \sum_i q_{ik}}{\sum_{k'} \Delta_{k'}}$



scDeepCluster [Tian et al. (2019)]

- Clustering de cellules à partir de données single-cell RNAseq
- Denoising ZINB model-based autoencoder

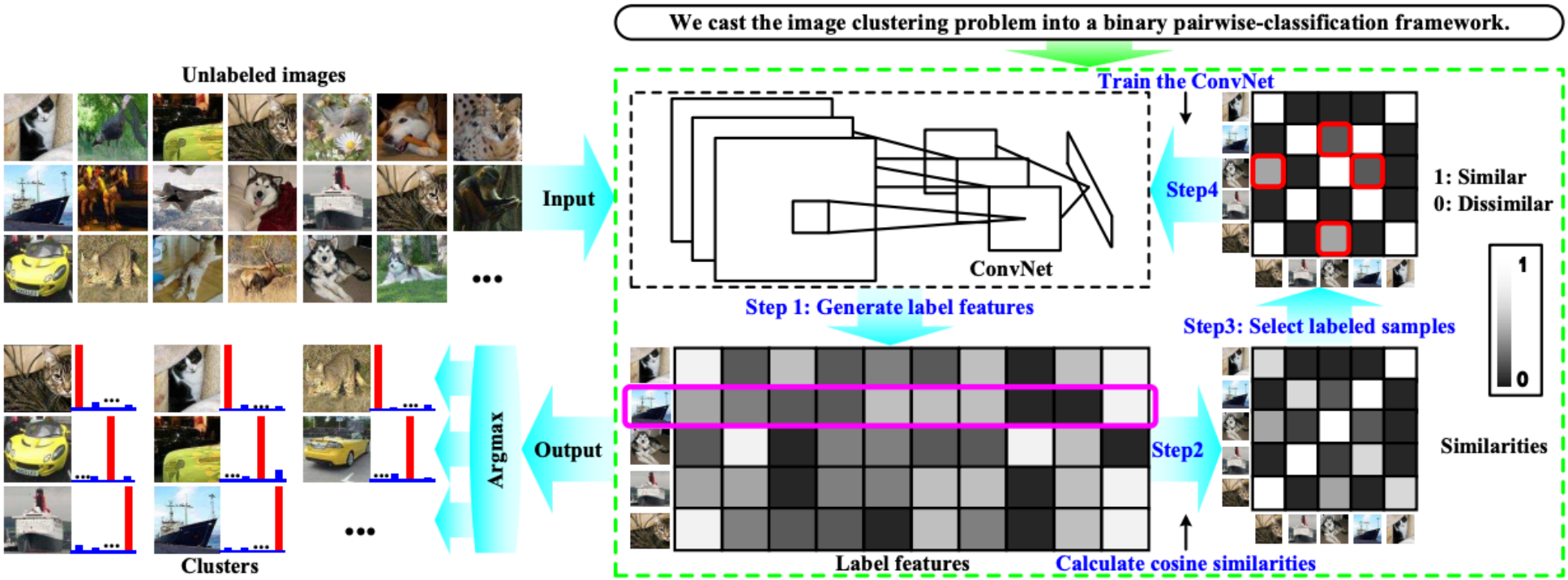


- Fonction de perte :

$$L(\theta, M) = L_{ZINB}(\theta) + \alpha\ L_{KL}(\theta, M)$$

Deep Adaptive Image Clustering (DAC)

[Chang et al. (2017)]



A retenir

- De nombreux choix :
  - forme du réseau de neurones, L_R, L_C
  - Choix de la méthode d'optimisation pour calibrer les paramètres
- Le temps calcul à K fixé
- Sélection du nombre de classes
- ...

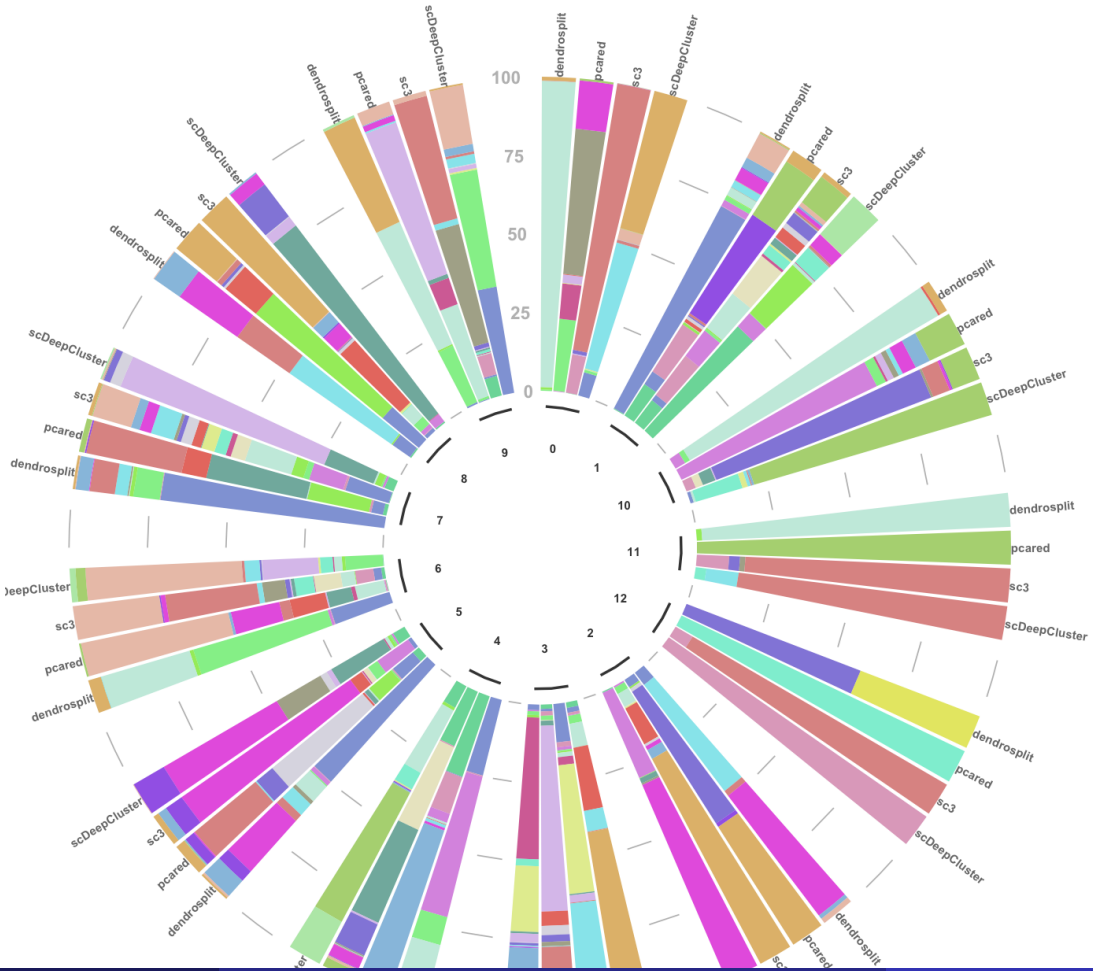
Plan

“Validation” d’un clustering

- Avec des connaissances externes
- Calculs d’indicateurs comme l’inertie intra par classe, les proba a posteriori en mélange, . . .
- Stabilité par perturbations
- . . .

Comparaison de plusieurs clustering

- L'indice ARI (Adjusted Rand Index)
- Stabilité de sous-groupes dans les différents clusterings
- Consensus de clusterings



Et on fait quoi . . .

- si on a des données manquantes
- pour tenir compte des réplicats, design des données, . . .
- pour exploiter des multiviews
- pour faire du biclustering
- . . .

Références

- Review général sur le clustering


Basel Abu-Jamous, Rui Fa, and Asoke K Nandi. "Integrative cluster analysis in bioinformatics". In: (2015).


Charu C Aggarwal and Chandan K Reddy. "Data clustering". In: *Algorithms and applications. Chapman&Hall/CRC Data mining and Knowledge Discovery series, Londra* (2014).


Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data clustering: theory, algorithms, and applications*. SIAM, 2020.

- Modèles de mélanges finis


Charles Bouveyron et al. *Model-based clustering and classification for data science: with applications in R*. Vol. 50. Cambridge University Press, 2019.


G.J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, 2000.

Références

- Deep Clustering


Elie Aljalbout et al. “Clustering with deep learning: Taxonomy and new methods”. In: *arXiv preprint arXiv:1801.07648* (2018).


Md Rezaul Karim et al. “Deep learning-based clustering approaches for bioinformatics”. In: *Briefings in Bioinformatics* 22.1 (2021), pp. 393–415.

- Multiviews clustering


Guoqing Chao, Shiliang Sun, and Jinbo Bi. “A survey on multi-view clustering”. In: *arXiv preprint arXiv:1712.06246* (2017).


Lele Fu et al. “An overview of recent multi-view clustering”. In: *Neurocomputing* 402 (2020), pp. 148–161.

Références

- Biclustering


Ziv Shkedy et al. *Applied biclustering methods for big and high dimensional data using R*. CRC Press Taylor & Francis Group, 2016.


Juan Xie et al. "It is time to apply biclustering: a comprehensive review of biclustering applications in biological and biomedical data". In: *Briefings in Bioinformatics* 20.4 (2018), pp. 1450–1465.

- Basel Abu-Jamous, Rui Fa, and Asoke K Nandi. "Integrative cluster analysis in bioinformatics". In: (2015).
- Charu C Aggarwal and Chandan K Reddy. "Data clustering". In: *Algorithms and applications. Chapman&Hall/CRC Data mining and Knowledge Discovery series, Londra* (2014).
- Elie Aljalbout et al. "Clustering with deep learning: Taxonomy and new methods". In: *arXiv preprint arXiv:1801.07648* (2018).
- Charles Bouveyron et al. *Model-based clustering and classification for data science: with applications in R*. Vol. 50. Cambridge University Press, 2019.
- Guoqing Chao, Shiliang Sun, and Jinbo Bi. "A survey on multi-view clustering". In: *arXiv preprint arXiv:1712.06246* (2017).
- Lele Fu et al. "An overview of recent multi-view clustering". In: *Neurocomputing* 402 (2020), pp. 148–161.
- Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data clustering: theory, algorithms, and applications*. SIAM, 2020.


Md Rezaul Karim et al. "Deep learning-based clustering approaches for bioinformatics". In: *Briefings in Bioinformatics* 22.1 (2021), pp. 393–415.


G.J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley, 2000.


Ziv Shkedy et al. *Applied biclustering methods for big and high dimensional data using R*. CRC Press Taylor & Francis Group, 2016.


Juan Xie et al. "It is time to apply biclustering: a comprehensive review of biclustering applications in biological and biomedical data". In: *Briefings in Bioinformatics* 20.4 (2018), pp. 1450–1465.