Algorithmes d’apprentissage non supervisé

01-02

Partitionnement en K-moyennes

Au programme

- K-moyennes: intuition
- Algorithme des K-moyennes
- Initialisation de l’algorithme
- Choix du nombre de clusters
- Mise à l’échelle
- K-moyennes avec scikit-learn
- Ateliers
- Lectures et références

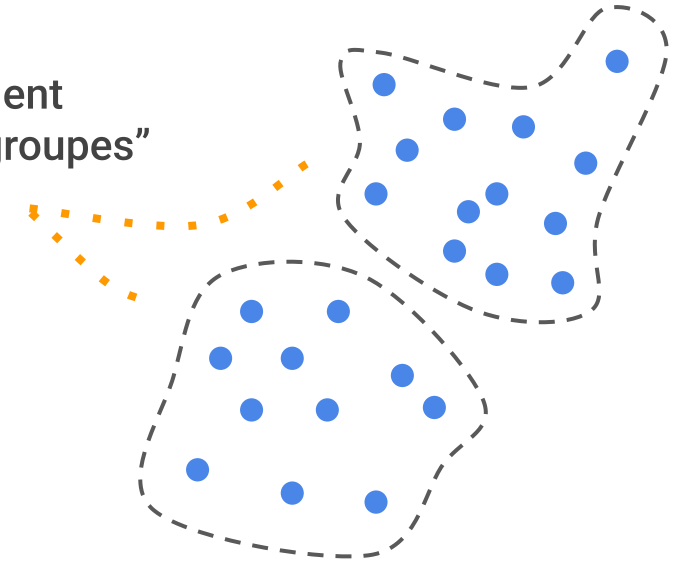
K-moyennes: intuition

Rappel des principaux types de partitionnement

- Partitionnement basé sur
    - les centroïdes (**K-moyennes**, CURE, …) 👈
    - la connectivité (hiérarchique, ...)
    - la distribution (BFR, …)
    - la densité (DBSCAN, OPTICS, …)
    - les grilles
- Et d’autres ….

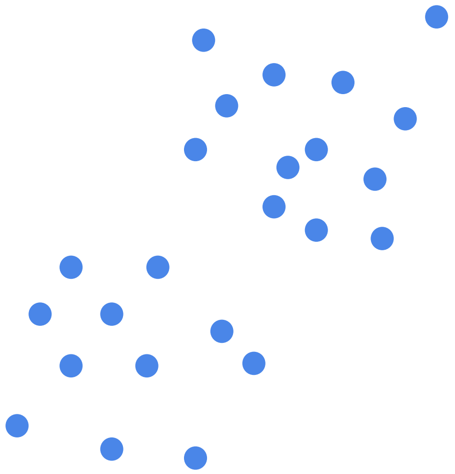
K-moyennes: intuition

Ces données semblent partitionnées en deux “groupes”



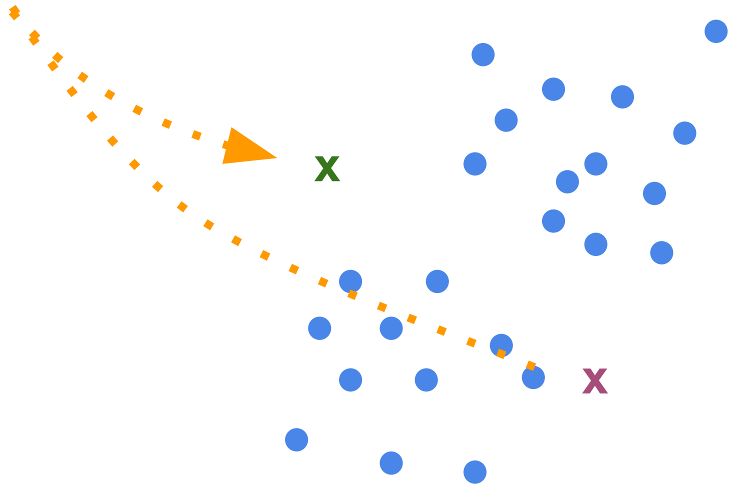
→ Comment découvrir ces partitions, ou clusters ?

K-moyennes: intuition



K-moyennes: intuition

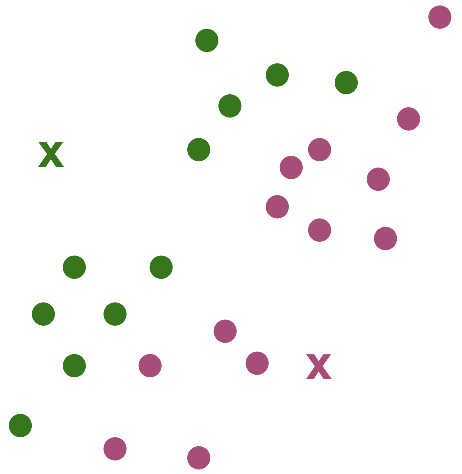
On place au hasard K **centroïdes** initiaux (ici K = 2)



K-moyennes: intuition

On place au hasard K centroïdes initiaux (ici $K = 2$)

On assigne les observations au centroïde le plus proche

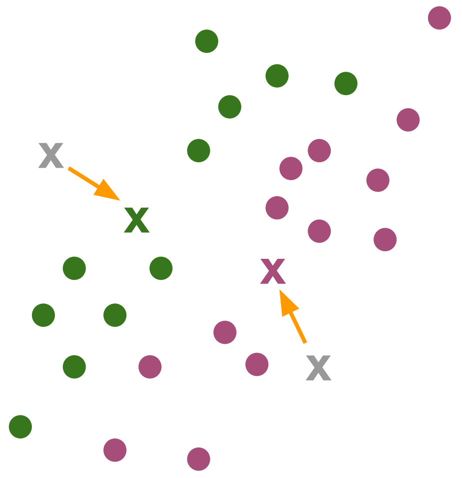


K-moyennes: intuition

On place au hasard K centroïdes initiaux (ici $K = 2$)

On assigne les observations au centroïde le plus proche

On déplace les centroïdes

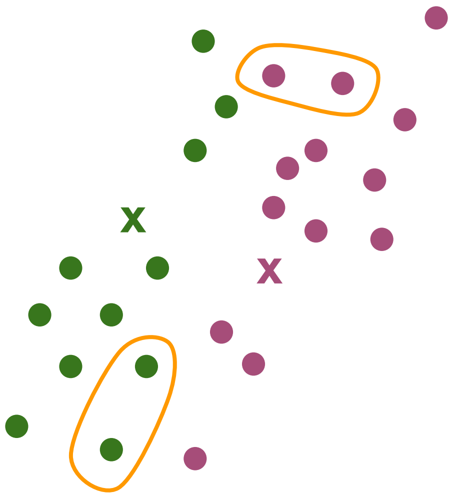


K-moyennes: intuition

On place au hasard K centroïdes initiaux (ici $K = 2$)

On assigne les observations au centroïde le plus proche

On déplace les centroïdes

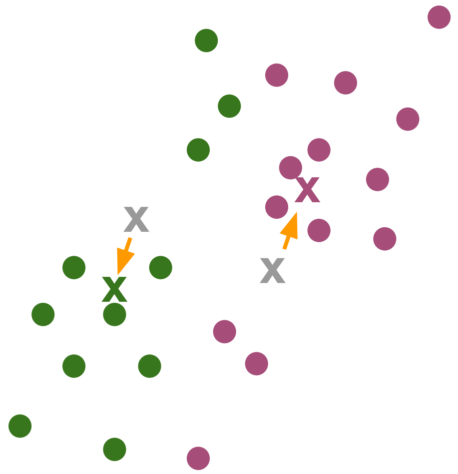


K-moyennes: intuition

On place au hasard K centroïdes initiaux (ici $K = 2$)

On assigne les observations au centroïde le plus proche

On déplace les centroïdes

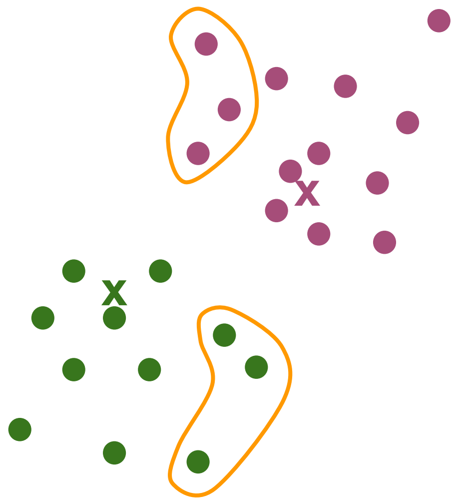


K-moyennes: intuition

On place au hasard K centroïdes initiaux (ici $K = 2$)

On assigne les observations au centroïde le plus proche

On déplace les centroïdes

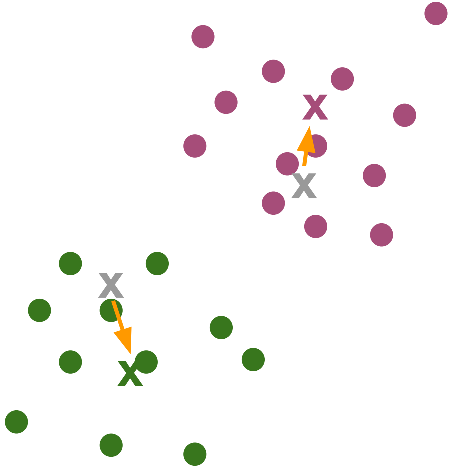


K-moyennes: intuition

On place au hasard K centroïdes initiaux (ici $K = 2$)

On assigne les observations au centroïde le plus proche

On déplace les centroïdes



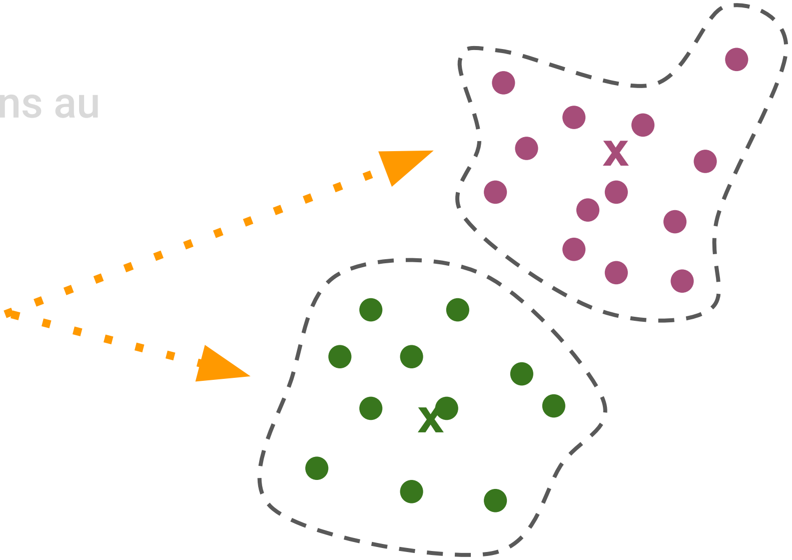
K-moyennes: intuition

On place au hasard K centroïdes initiaux (ici $K = 2$)

On assigne les observations au centroïde le plus proche

On déplace les centroïdes

On obtient deux clusters

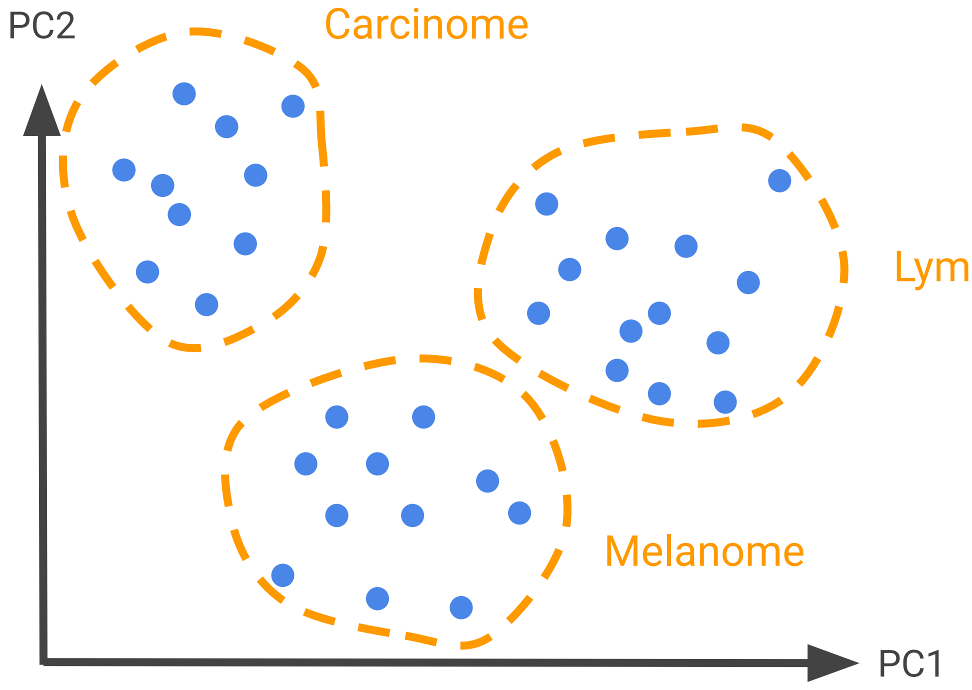


K-moyennes: intuition

La procédure précédente se réduit à un problème mathématique simple et intuitif: en considérant K clusters, si C_1, ..., C_K dénotent les ensembles contenant les indices des observations pour chaque cluster, alors les deux propriétés suivantes sont satisfaites:

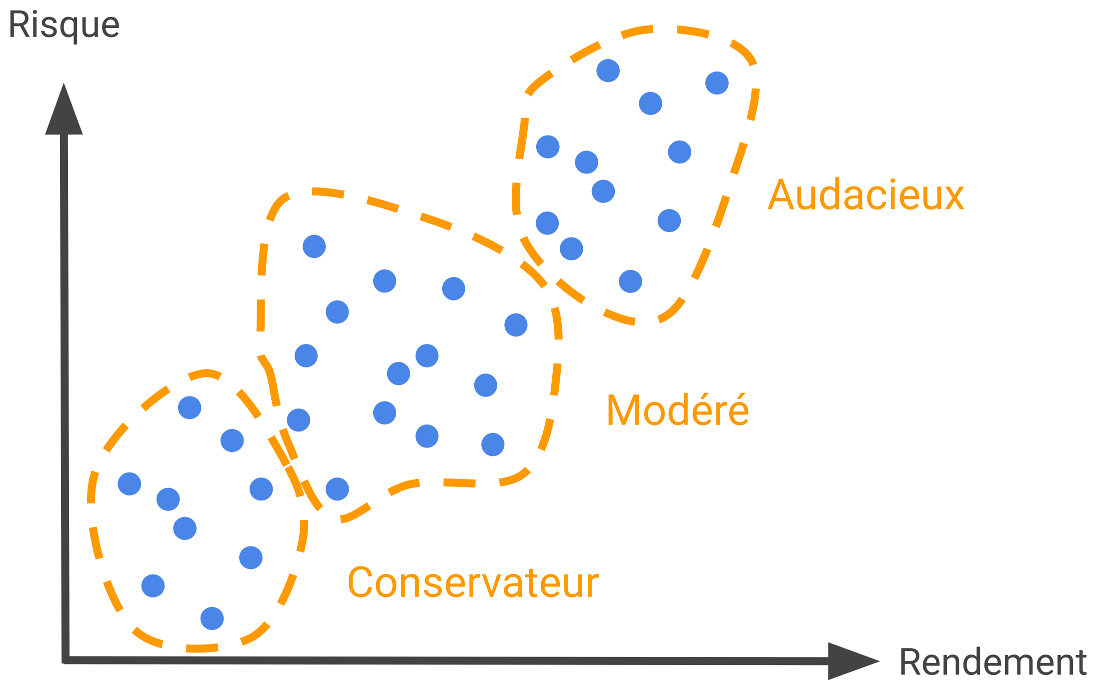
- $C_1 \cup C_2 \cup \ldots \cup C_K = \{ 1, \ldots, m \}$ → Chaque observation appartient à au moins un des K clusters
- $C_k \cap C_{k'} = \varnothing \ \forall \ k \neq k'$ → Les clusters ne se chevauchent pas, car aucune observation n'appartient à plus d'un cluster

K-moyennes pour clusters séparés



Cellules cancéreuses

K-moyennes pour clusters <u>non</u> séparés



Profiles d'investisseurs

Algorithme des K-moyennes

Ce que l'on cherche à optimiser

Soit les notations suivantes:

- $c^{(i)}$ index du cluster $(1, 2, \ldots, K)$ auquel l'observation $x^{(i)}$ est assignée
- μ_k centroïde du cluster k
- $\mu_{c^{(i)}}$ centroïde du cluster auquel l'observation $x^{(i)}$ est assignée

L'objectif est de minimiser la **fonction de coût** ci-dessous

Distance euclidienne

$$J(c^{(1)}, \cdots, c^{(m)}, \mu_1, \cdots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

En d'autres termes, la distance entre les observations d'un même cluster doit être petite, tandis que la distance entre les clusters doit être grande

Ce que l'on cherche à optimiser

Minimiser la fonction de coût

$$\min_{\substack{c^{(1)},\cdots,c^{(m)} \\ \mu_1,\cdots,\mu_K}} J(c^{(1)},\cdots,c^{(m)},\mu_1,\cdots,\mu_K)$$

Est un problème **NP-hard** !

Nécessité de trouver une solution approchée

Solution approchée

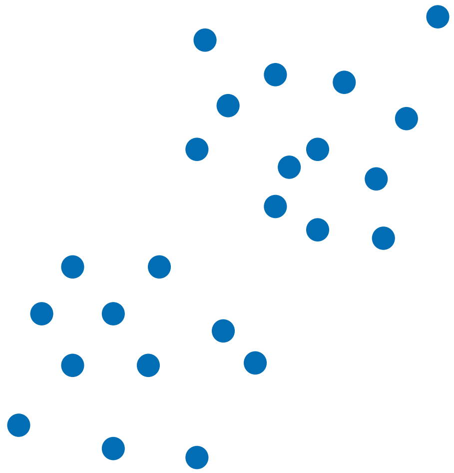
- **Algorithme de Lloyd (K-moyennes)**
- Aussi connu sous le nom d'itérations de Voronoï, il s'agit d'un algorithme itératif issu du génie électrique
- La fonction de coût

$$J(c^{(1)}, \cdots, c^{(m)}, \mu_1, \cdots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

est aussi appelée **distorsion**

Algorithme des K-moyennes: pseudo-code

```
Choix de K

Initialisation aléatoire de K centroïdes mu[1],
mu[2], …, mu[K]

Répéter {

    for i = 1 to m

        c[i] := index (de 1 à K) du centroïde le
        plus proche de x[i]

    for k = 1 to K

        mu[k] := moyenne des points assignés au
        cluster k

}
```



Algorithme des K-moyennes: pseudo-code

```
Choix de K

Initialisation aléatoire de K centroïdes mu[1],
mu[2], …, mu[K]

Répéter {

    for i = 1 to m

        c[i] := index (de 1 à K) du centroïde le
        plus proche de x[i]

    for k = 1 to K

        mu[k] := moyenne des points assignés au
        cluster k

}
```

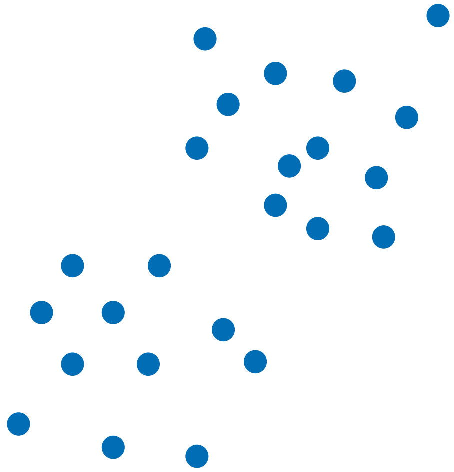
$$\min_{c^{(1)},\cdots,c^{(m)}} J(c^{(1)},\cdots,c^{(m)},\overbrace{\mu_1,\cdots,\mu_K}^{\text{constant}})$$

$$\min_{\mu_1,\cdots,\mu_K} J(\underbrace{c^{(1)},\cdots,c^{(m)}}_{\text{constant}},\mu_1,\cdots,\mu_K)$$

Initialisation de l’algorithme

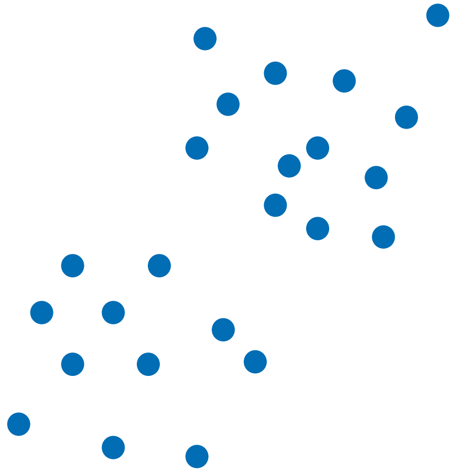
Initialisation de l’algorithme

```
Choix de K

Initialisation aléatoire de K centroïdes mu[1],
mu[2], …, mu[K]

Répéter {

    for i = 1 to m

        c[i] := index (de 1 à K) du centroïde le
        plus proche de x[i]

    for k = 1 to K

        mu[k] := moyenne des points assignés au
        cluster k

}
```



Initialisation de l'algorithme

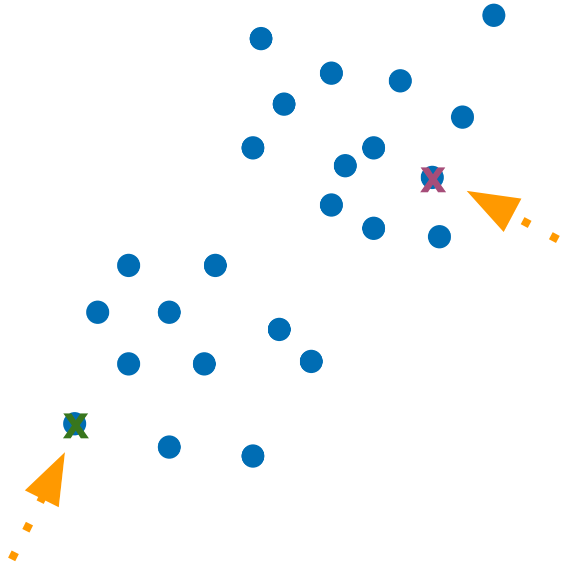
```
Choix de K

Initialisation aléatoire de K centroïdes mu[1],
mu[2], …, mu[K]

Répéter {

    for i = 1 to m

        c[i] := index (de 1 à K) du centroïde le
        plus proche de x[i]

    for k = 1 to K

        mu[k] := moyenne des points assignés au
        cluster k

}
```



Initialisation de l'algorithme

```
Choix de K

Initialisation aléatoire de K centroïdes mu[1],
mu[2], …, mu[K]

Répéter {

    for i = 1 to m
        c[i] := index (de 1 à K) du centroïde le
        plus proche de x[i]

    for k = 1 to K
        mu[k] := moyenne des points assignés au
        cluster k

}
```

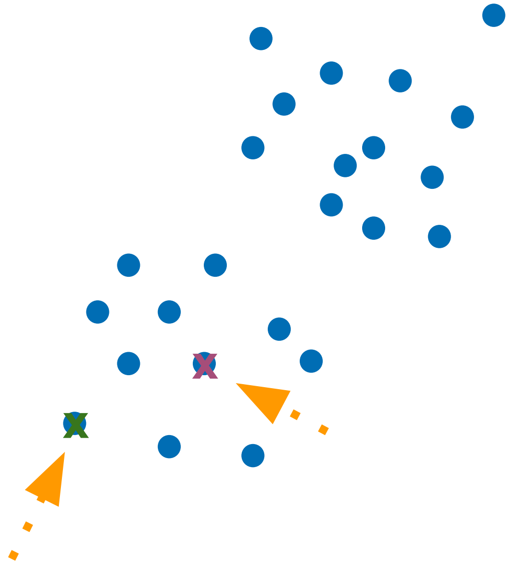
- K doit être inférieur à m (nombre d'observations)
- Choisir au hasard K observations
- Les K centroïdes $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$ sont égals à ces observations



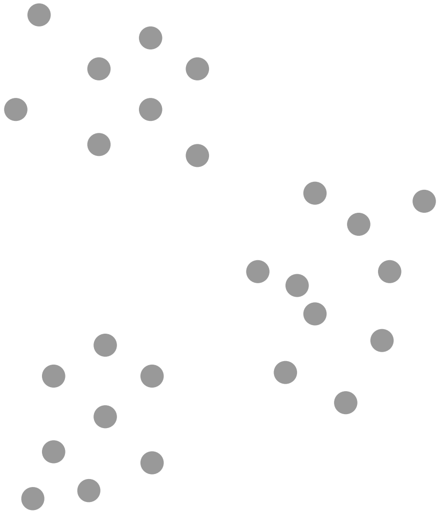
Initialisation de l'algorithme

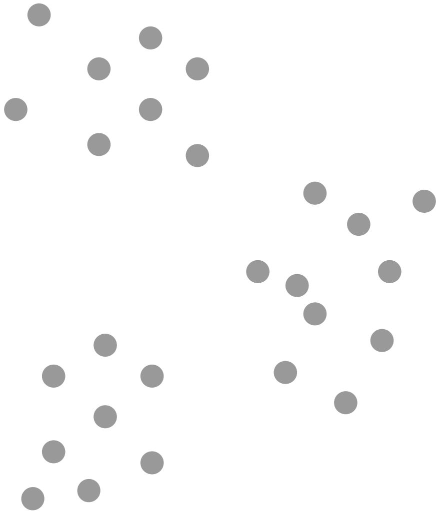
```
Choix de K

Initialisation aléatoire de K centroïdes mu[1],
mu[2], …, mu[K]

Répéter {

    for i = 1 to m

        c[i] := index (de 1 à K) du centroïde le
        plus proche de x[i]

    for k = 1 to K

        mu[k] := moyenne des points assignés au
        cluster k

}
```

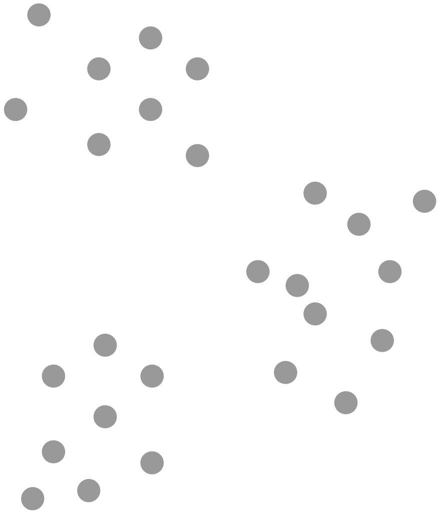
- K doit être inférieur à m (nombre d'observations)
- Choisir au hasard K observations
- Les K centroïdes $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$ sont égals à ces observations



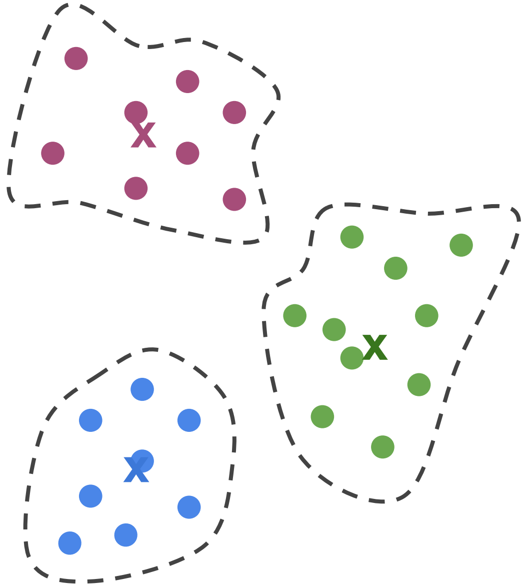
Algorithme des K-moyennes - optimum local

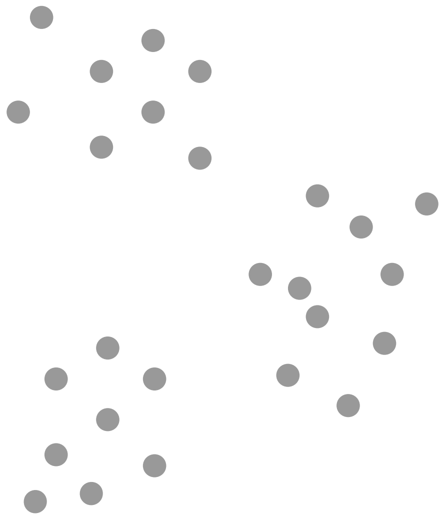


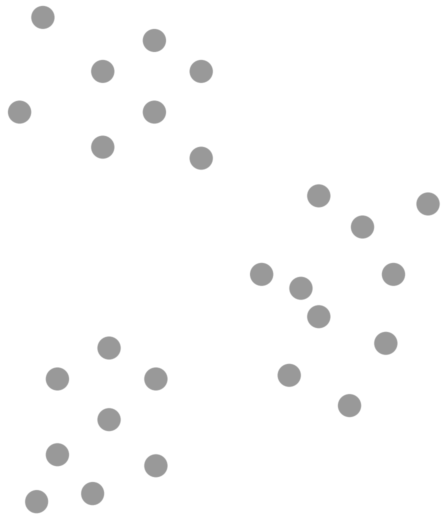




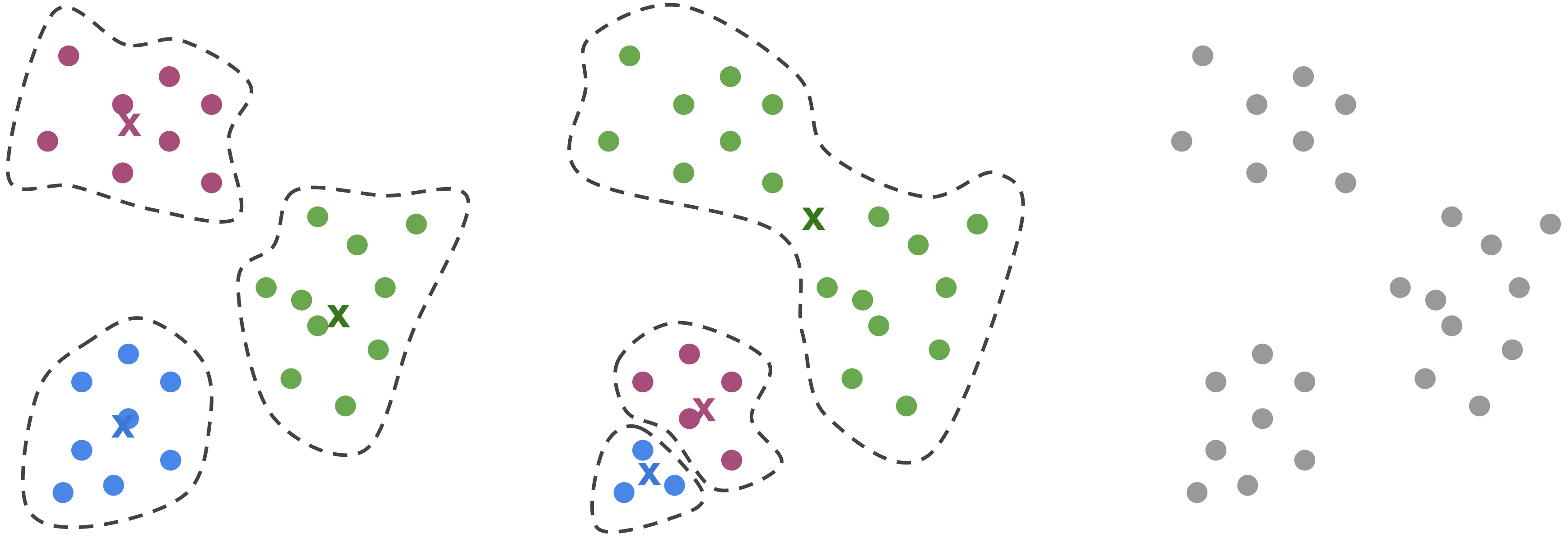
Algorithme des K-moyennes - optimum local



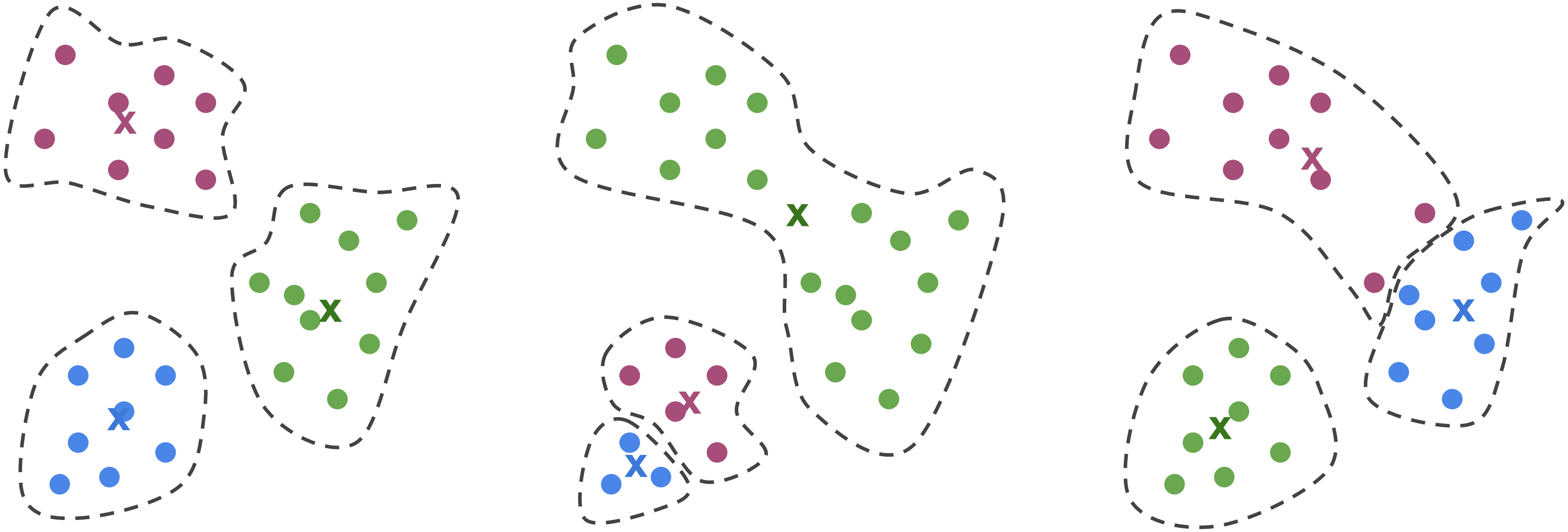




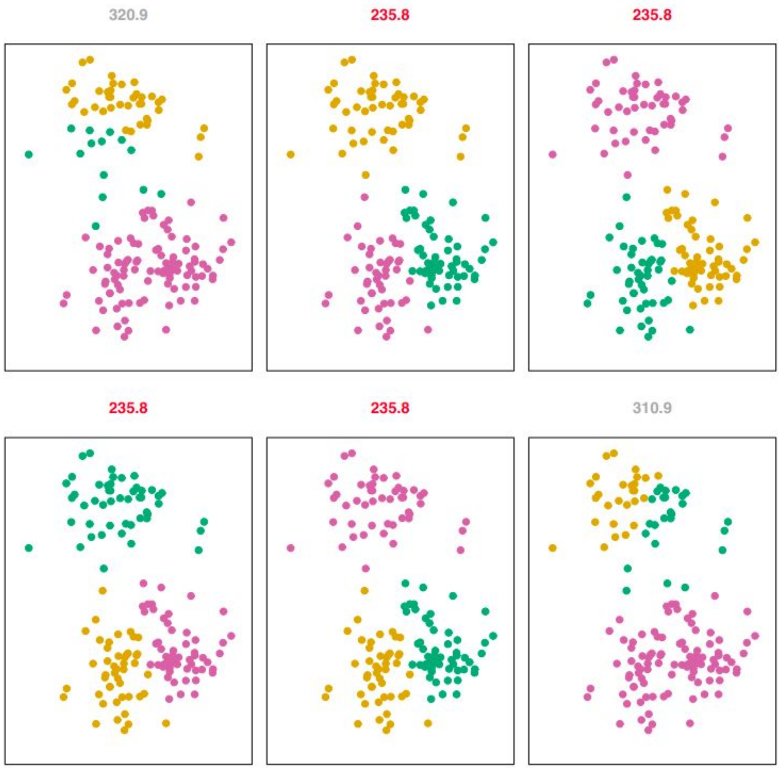
Algorithme des K-moyennes - optimum local



Algorithme des K-moyennes - optimum local



Algorithme des K-moyennes - optimum local



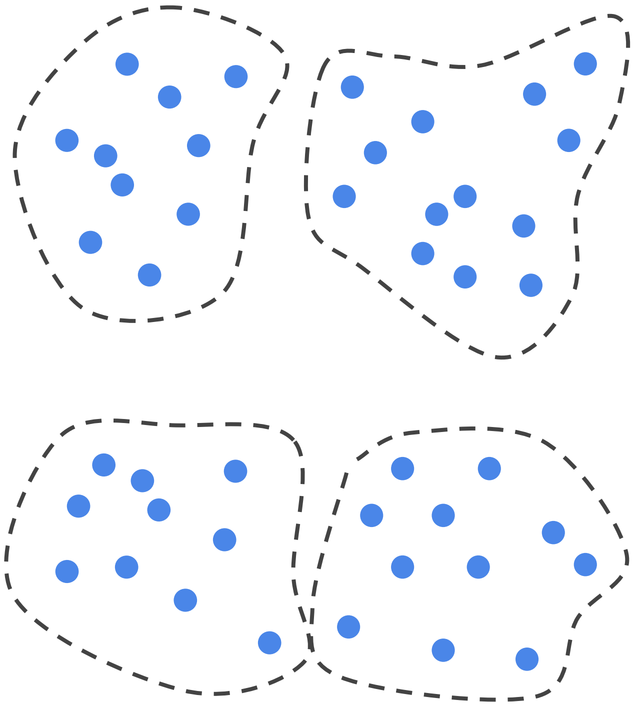
Extrait de [2] - Ici l'algorithme des K-moyennes est exécuté 6 fois avec $K = 3$

Algorithme des K-moyennes - exécutions multiples

```
for i = 1 to n_init {
    Initialisation aléatoire des K centroïdes mu[1], mu[2], …, mu[K]
    Exécution de l’algorithme des K-moyennes
    Sauvegarder dans un historique mu[1], mu[2], …, mu[K]
    Calculer la distorsion J
}
Choisir le résultat donnant la distorsion la plus faible
```

Choix du nombre de clusters

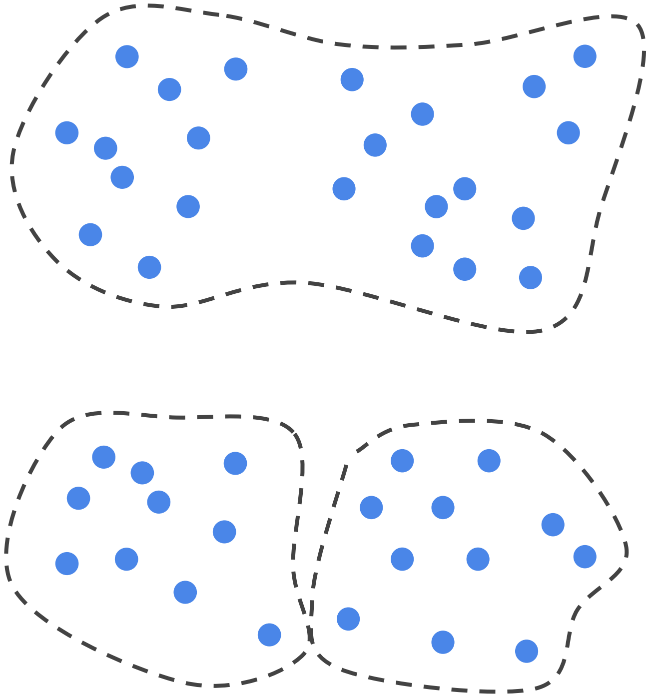
Le choix de K est souvent subjectif ...



Combien de clusters voyez vous ?

4 ?

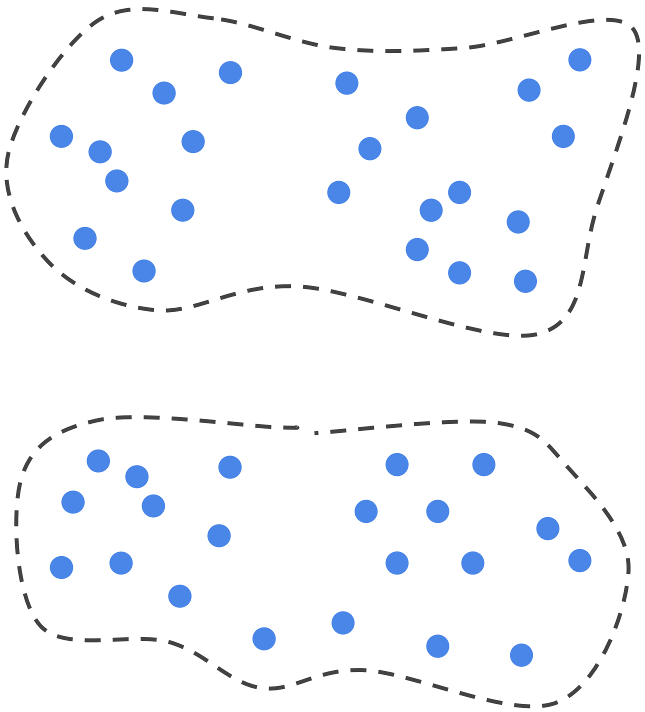
Le choix de K est souvent subjectif ...



Combien de clusters voyez vous ?

3 ?

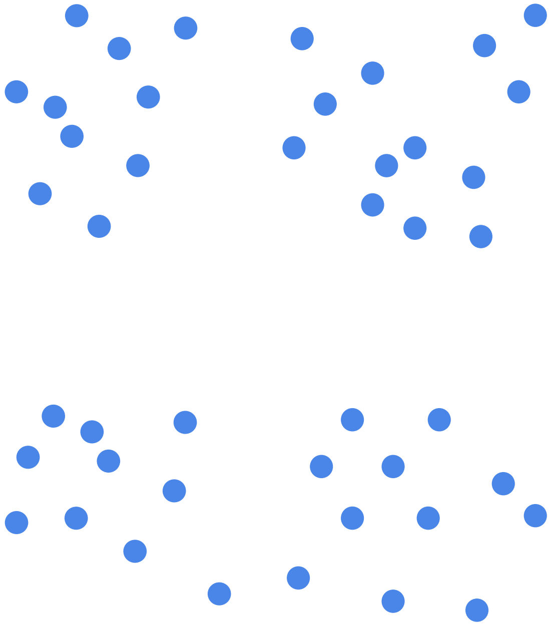
Le choix de K est souvent subjectif ...



Combien de clusters voyez vous ?

2 ?

Le choix de K est souvent subjectif ...



Combien de clusters voyez vous ?

Toutes les réponses peuvent être valables !

Le choix de K est souvent subjectif ...

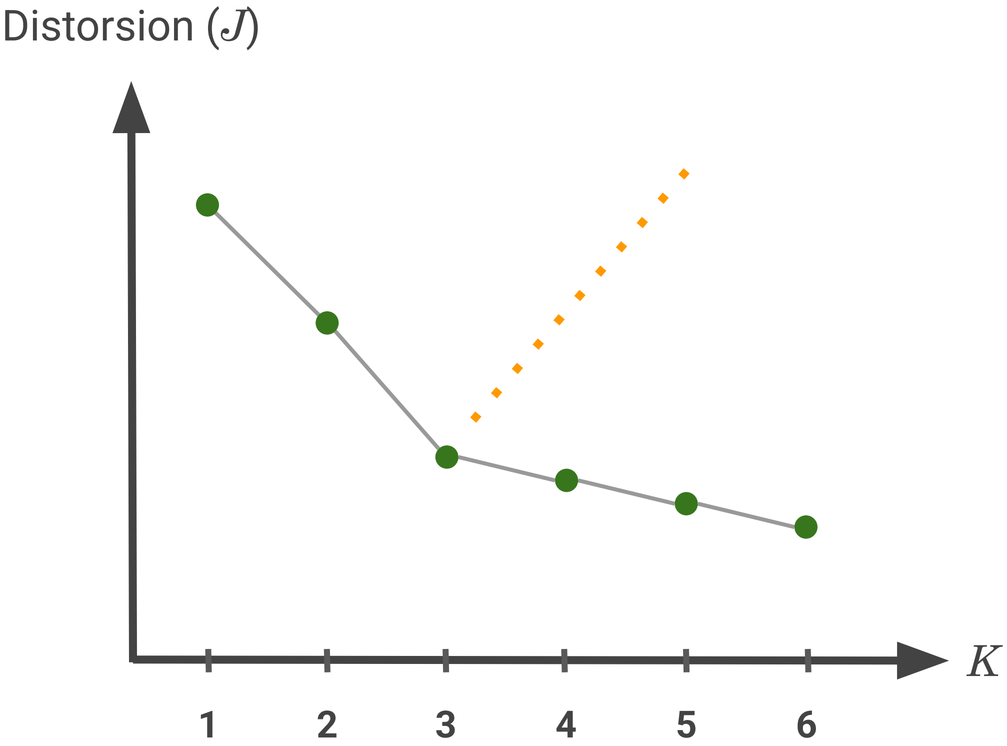
Il n’existe toujours pas de méthode “miracle” et la meilleure reste un choix… manuel !

Combien de clusters voyez vous ?

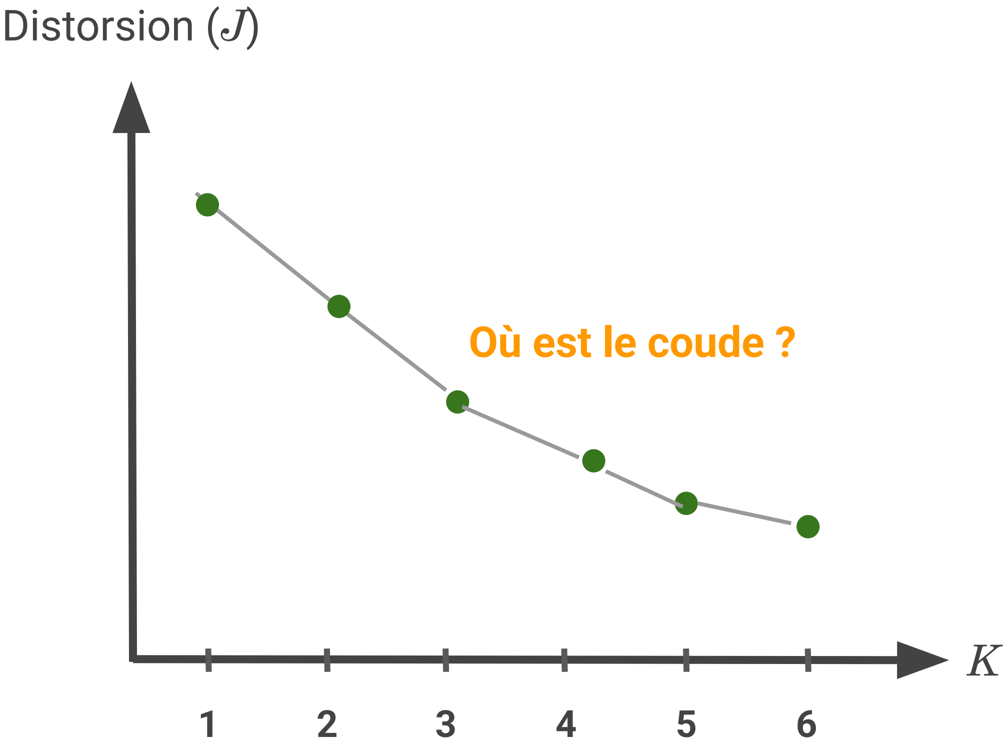
Il existe néanmoins quelques techniques pouvant fonctionner (parfois) ...

Toutes les réponses peuvent être valables !

La méthode du “coude” … en théorie



Et en pratique !



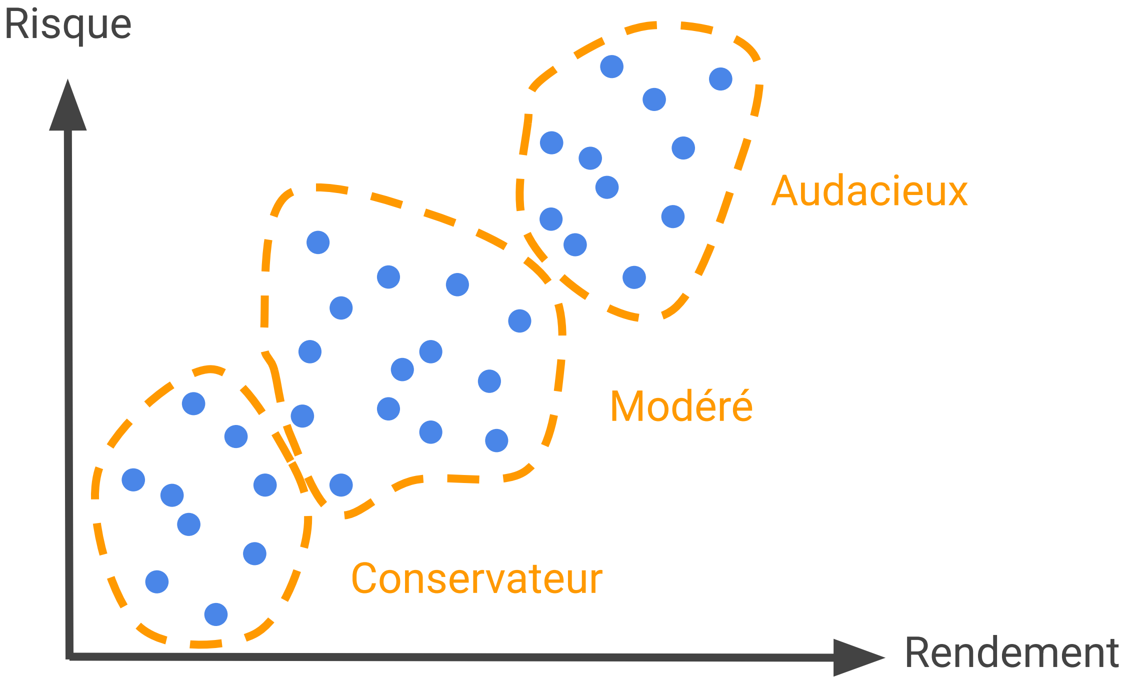
Algorithme des K-moyennes - “GridSearch”

```
for k = 1 to K  {
    for i = 1 to n_iter {
        Initialisation aléatoire des k centroïdes mu[1], mu[2], …, mu[K]
        Exécution de l’algorithme des K-moyennes
        Sauvegarder dans un historique mu[1], mu[2], …, mu[K]
        Calculer la distorsion J
    }
}
Choisir la valeur de K donnant la distorsion la plus faible
```

Exemple: 2 à 15

Exemple: 1 à 1000

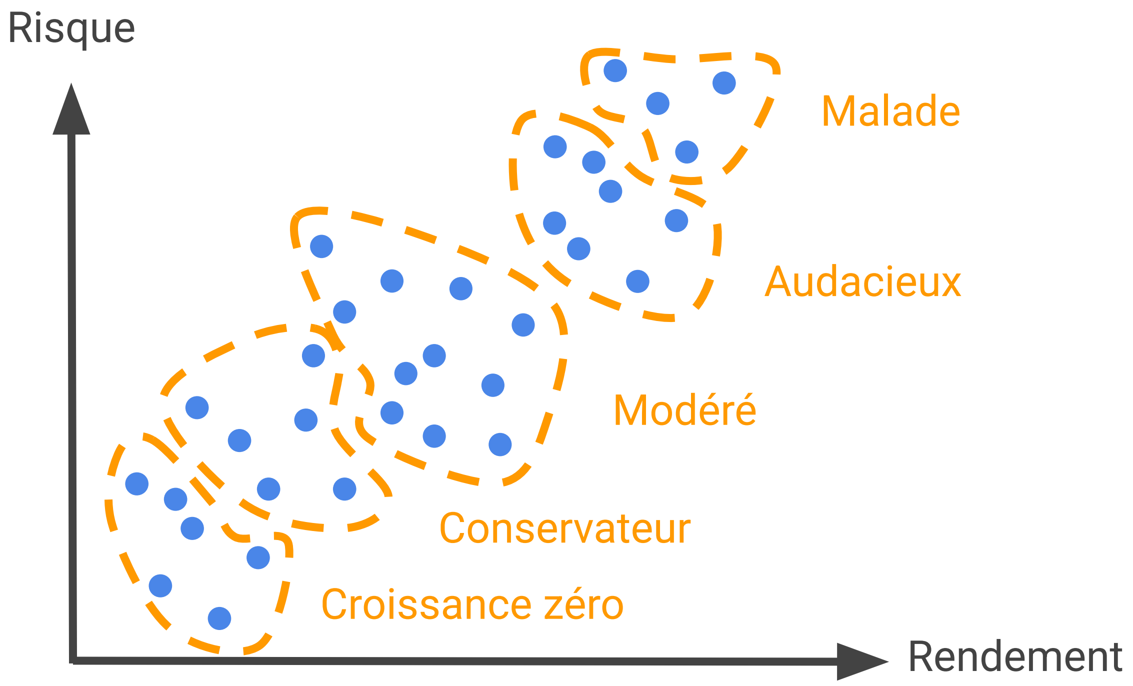
Choix de K à posteriori



$K = 3$

Profiles d'investisseurs

Choix de K à posteriori



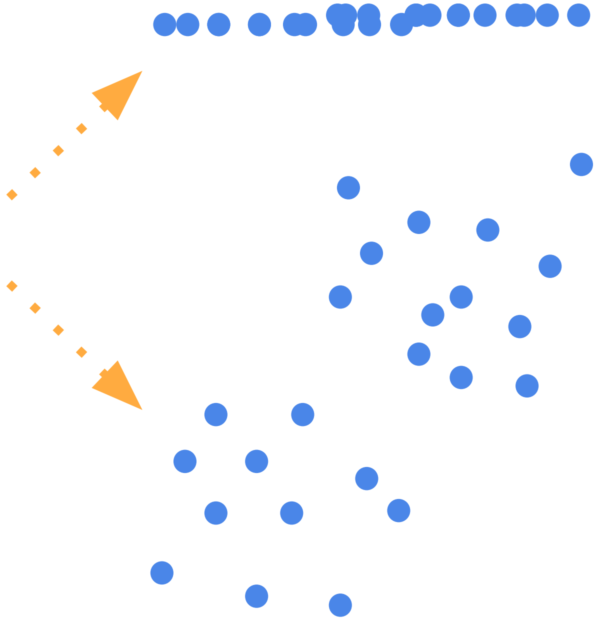
$K = 5$

Profiles d'investisseurs

Mise à l'échelle

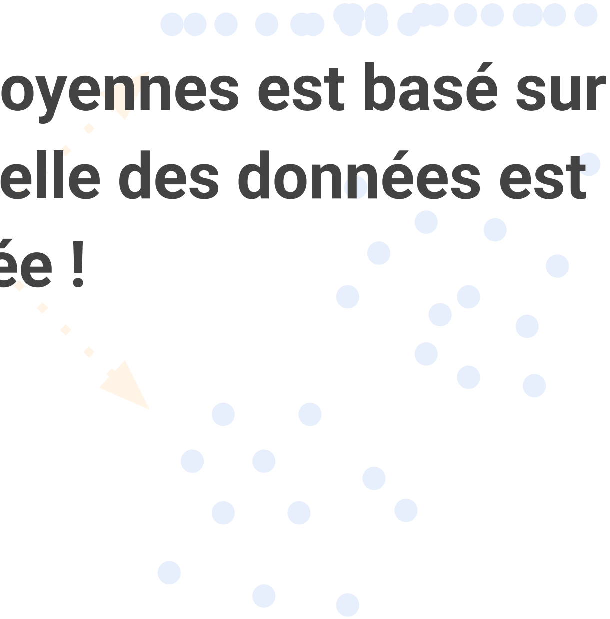
Mise à l'échelle

```
Choix de K

Initialisation aléatoire de K centroïdes mu[1],
mu[2], …, mu[K]

Répéter {

    for i = 1 to m

        c[i] := index (de 1 à K) du centroïde le
        plus proche de x[i]

    for k = 1 to K

        mu[k] := moyenne des points assignés au
        cluster k

}
```



Mise à l’échelle

```
Choix de K

Initialisation aléatoire de K centroïdes mu[1],
mu[2], …, mu[K]

Répéter {

    for i = 1 to m

        c[i] := index (de 1 à K) du centroïde le
        plus proche de x[i]

    for k = 1 to K

        mu[k] := moyenne des points assignés au
        cluster k

}
```



Étant donné que l’algorithme des K-moyennes est basé sur le calcul de distances, la mise à l’échelle des données est très recommandée !

K-moyennes avec scikit-learn

KMeans (scikit-learn 1.8)

Nombre de clusters (K)

Nombre d'itérations maximum

Nombre d'exécutions

*class sklearn.cluster.**KMeans**(**n_clusters=8**, *, **init='k-means++'**, **n_init='auto'**, **max_iter=300**, tol=0.0001, verbose=0, random_state=None, copy_x=True, **algorithm='lloyd'**)*

Méthode d'initialisation

Algorithme vu en cours

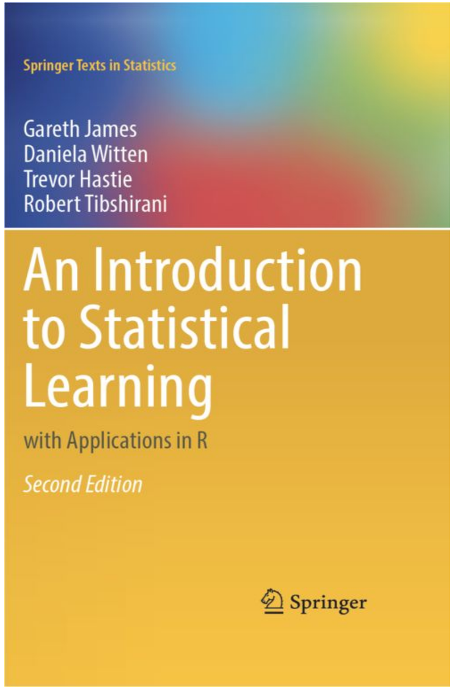
Ateliers

https://github.com/mswawola-cegep/420-a58-sf-gr-12022-hiver-2026.git

01-02-A1 et 01-02-A2

Lectures et références

Lectures



- Introduction to Statistical Learning with Applications in R
  Second edition (2021)

  → 12.4 Clustering Methods

Références

[1] CS229: Machine Learning - Stanford University

[2] Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, “Introduction to Statistical Learning with Applications in R - Second edition”