



Introduction au Machine Learning

Séance 9 — Apprendre sans étiquettes : clustering et réduction de dimension

Dr. El Hadji Bassirou TOURÉ

Département de Mathématiques et Informatique, Faculté des Sciences et Techniques, Université Cheikh Anta Diop de Dakar

<https://ebtoure.sn/cours/introduction-machine-learning/chapitres/09-non-supervise.html>

Version du 11 octobre 2026



Version en ligne

Introduction au ML — Séance 9

Apprendre sans étiquettes : clustering et réduction de dimension

Dr. El Hadji Bassirou TOURÉ

Département de Mathématiques et Informatique, Faculté des Sciences et Techniques, Université Cheikh Anta Diop de Dakar

Objectifs de la séance

Contenu

Jusqu'ici, chaque exemple venait avec une *réponse* (l'étiquette à prédire) : c'était l'apprentissage **supervisé**. Cette séance explore l'autre grande famille — l'apprentissage **non-supervisé**, où l'on cherche une structure dans des données *sans* réponse connue.

- Le **non-supervisé** : ce qui change quand il n'y a plus d'étiquettes, et ce que l'on peut alors apprendre.
- Le **clustering** avec **k-means** : regrouper les exemples similaires, en réutilisant la *distance* de la Séance 4. Algorithme, exemple chiffré, choix du nombre de groupes.
- Les fondations mathématiques nécessaires : **variance**, **covariance**, **vecteurs propres** — chacune avec intuition, formalisme et chiffres.
- La **réduction de dimension** avec l'**ACP** (PCA) : compresser les données en conservant l'essentiel de l'information.
- **Application** à DataSANTÉ : des clusters de patients, une visualisation en 2D, et un test révélateur du lien avec le paludisme.

Apprendre sans étiquettes

Un changement de monde

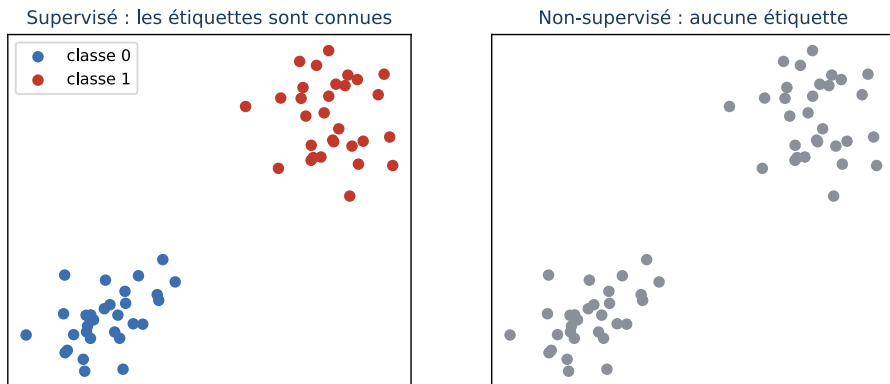
De la réponse connue à la structure cachée

Les Séances 3 à 8 partageaient un point commun : chaque exemple portait une *étiquette* — le diagnostic, le prix, la classe. Le modèle apprenait à reproduire cette réponse. Mais une immense partie des données du monde réel n'a **aucune étiquette** : des dossiers patients sans diagnostic posé, des transactions sans catégorie, des images sans description. Peut-on encore apprendre quelque chose ? Oui — c'est tout l'objet du non-supervisé.

Sans réponse à prédire, l'objectif change : non plus *reproduire* une étiquette, mais *découvrir* une structure que personne n'a annotée. Deux grandes tâches : regrouper les exemples qui se ressemblent (**clustering**), et simplifier des données complexes en gardant l'essentiel (**réduction de dimension**). Cette séance traite les deux, avec leurs outils phares : k-means et l'ACP.

Supervisé contre non-supervisé

Deux mondes : avec ou sans réponse connue



À gauche, le supervisé : chaque point a une couleur (son étiquette connue). À droite, le non-supervisé : les mêmes points *sans* couleur. La tâche n'est plus de classer selon une vérité donnée, mais de *trouver* seul des regroupements.

L'idée : deux grandes tâches sans étiquette

L'idée en une phrase

En non-supervisé, on cherche une **structure** dans les données seules. Deux tâches dominent : le *clustering* (regrouper les exemples similaires en « paquets ») et la *réduction de dimension* (décrire les données avec moins de variables, sans perdre l'essentiel).

Le **clustering** répond à « quels exemples se ressemblent ? » — utile pour segmenter une patientèle, détecter des profils, repérer des anomalies. La **réduction de dimension** répond à « comment résumer fidèlement des données à beaucoup de variables ? » — utile pour visualiser, compresser, ou nettoyer avant un autre modèle. Toutes deux reposent sur une notion géométrique simple, déjà rencontrée en Séance 4 : la *distance* entre exemples. Commençons par là.

Rappel — la distance entre deux exemples

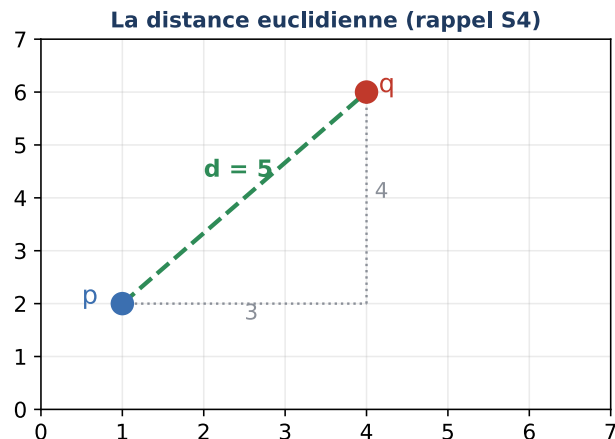
Définition

Deux exemples sont des points dans l'espace des variables. Leur **distance euclidienne** mesure à quel point ils se ressemblent : petite distance, exemples proches (similaires) ; grande distance, exemples éloignés.

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{j=1}^d (x_j - x'_j)^2}$$

C'est la distance de la Séance 4 (celle du k-NN), généralisation du théorème de Pythagore à d dimensions. Tout le non-supervisé s'appuie dessus : un cluster regroupe des points *proches* ; un axe principal capte la direction où les points *s'étalent* le plus. Exemple : entre $\mathbf{x} = (1; 2)$ et $\mathbf{x}' = (4; 6)$, $d = \sqrt{3^2 + 4^2} = \sqrt{25} = 5$.

La distance, en image



Entre les points $p = (1; 2)$ et $q = (4; 6)$: un déplacement horizontal de 3, vertical de 4, donc une distance $d = \sqrt{3^2 + 4^2} = 5$ (le triangle 3-4-5). Cette mesure de proximité est le seul ingrédient géométrique dont k-means a besoin.

Le clustering avec k-means

L'intuition : des paquets autour de centres

Regrouper ce qui se ressemble

Imaginons les dossiers de milliers de patients, sans diagnostic. Certains se ressemblent (jeunes, forte fièvre, saison des pluies), d'autres aussi (âgés, glycémie élevée). L'idée du clustering : former des **paquets** de patients similaires, chacun résumé par un patient « moyen » — son *centre*. C'est exactement ce que fait k-means.

Le « k » est le nombre de paquets (clusters) que l'on souhaite. Chaque cluster est représenté par son **centre** (la moyenne de ses membres). Un bon clustering place les centres de sorte que chaque point soit proche du sien et loin des autres. La question est : comment trouver ces centres automatiquement ? L'algorithme de k-means y répond par une boucle remarquablement simple.

L'idée : alterner deux étapes

L'idée en une phrase

k-means part de k centres au hasard, puis alterne deux étapes jusqu'à stabilité : **(1)** affecter chaque point au centre le plus proche ; **(2)** recalculer chaque centre comme la moyenne de ses points.

C'est tout. L'étape (1) utilise la distance ; l'étape (2) une simple moyenne. On répète : les centres bougent, les affectations changent, jusqu'à ce que plus rien ne bouge — la *convergence*. L'algorithme est garanti de converger (en un nombre fini d'étapes), car chaque étape ne peut que diminuer une certaine quantité d'erreur, l'*inertie*, que nous définirons. Voyons d'abord le mécanisme se dérouler sur un petit exemple, à la main.

k-means, formellement

Les deux étapes

Pour k centres $\mathbf{c}_1, \dots, \mathbf{c}_k$. **Affectation** : chaque point $\mathbf{x}_i$ rejoint le cluster du centre le plus proche. **Mise à jour** : chaque centre devient la moyenne des points de son cluster.

$$\text{affectation : } a_i = \arg \min_k \|\mathbf{x}_i - \mathbf{c}_k\|^2 \quad \text{mise à jour : } \mathbf{c}_k = \frac{1}{|C_k|} \sum_{i \in C_k} \mathbf{x}_i$$

La première équation attribue à chaque point l'indice a_i du centre le plus proche (au sens de la distance au carré). La seconde recalcule chaque centre $\mathbf{c}_k$ comme le barycentre (la moyenne) des points C_k qui lui sont affectés. On boucle entre les deux jusqu'à ce que les affectations ne changent plus. La répétition de ces deux gestes — proche, moyenne — est l'algorithme.

k-means à la main — 6 points, $k = 2$ (1/2)

Affectation puis mise à jour

Six points : $(1, 1)$, $(2, 1)$, $(1, 2)$ en bas à gauche, $(5, 6)$, $(6, 5)$, $(6, 6)$ en haut à droite. On vise $k = 2$ clusters. Initialisation *volontairement médiocre* : $c_1 = (1, 1)$, $c_2 = (2, 1)$ (les deux centres collés en bas).

- **Affectation 1** : chaque point au centre le plus proche $\rightarrow$ labels $[0, 1, 0, 1, 1, 1]$ (les trois points du haut rejoignent c_2 , plus proche). Inertie = 108.
- **Mise à jour 1** : $c_1 =$ moyenne de $\{(1, 1), (1, 2)\} = (1; 1,5)$; $c_2 =$ moyenne des quatre autres = $(4,75; 4,5)$.

Les centres ont bougé : on recommence.

k-means à la main — 6 points, $k = 2$ (2/2)

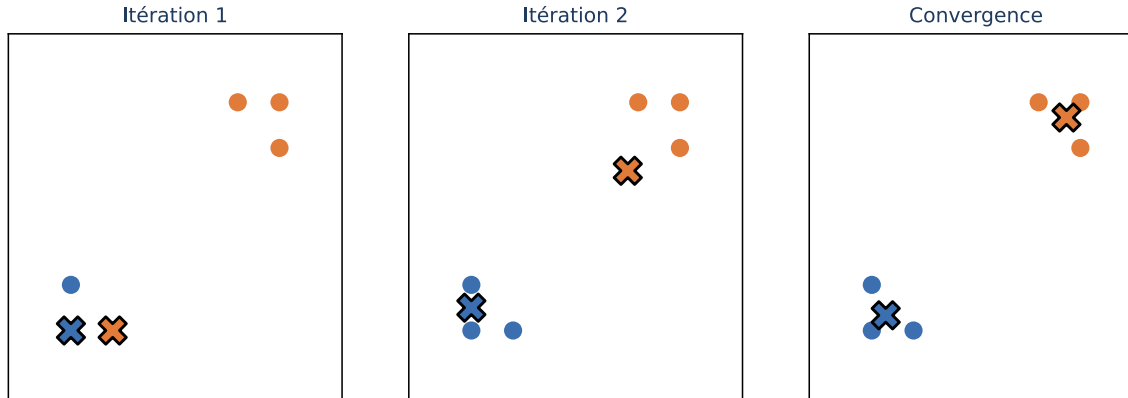
Convergence en deux itérations de plus

- **Affectation 2** : avec les nouveaux centres, labels = $[0, 0, 0, 1, 1, 1]$ — les trois points du bas ensemble, les trois du haut ensemble. Inertie = 9,69 (forte baisse).
- **Mise à jour 2** : $c_1 = (1,33; 1,33)$, $c_2 = (5,67; 5,67)$.
- **Affectation 3** : labels inchangés → *convergence*. Inertie finale = 2,67.

Partant d'une initialisation médiocre, k-means a trouvé les deux groupes naturels en trois itérations. [scikit-learn](#) donne exactement les mêmes centres et la même inertie (2,67). L'inertie a chuté de 108 à 2,67 : c'est la mesure que l'algorithme minimise.

Les itérations, en image

k-means : alterner affectation et mise à jour jusqu'à stabilité (X = centres)



De gauche à droite : l'initialisation médiocre (centres collés), puis le déplacement des centres (les X) au fil des mises à jour, jusqu'à la séparation nette des deux groupes. Chaque étape rapproche les centres du cœur de leur nuage — l'inertie diminue à chaque fois.

L'inertie : ce que k-means minimise

Définition

L'**inertie** (ou inertie intra-cluster) est la somme des distances au carré entre chaque point et le centre de son cluster. C'est la mesure de *compacité* des clusters : plus elle est faible, plus les points sont serrés autour de leur centre.

$$\mathcal{J} = \sum_{k=1}^K \sum_{i \in C_k} \|\mathbf{x}_i - \mathbf{c}_k\|^2$$

Chaque étape de k-means *diminue* l'inertie : l'affectation place chaque point près de son centre le plus proche (baisse) ; la mise à jour recentre chaque centre sur ses points (baisse encore). Comme l'inertie est positive et décroît à chaque étape, l'algorithme converge. *Attention* : il converge vers un **minimum local**, pas forcément le meilleur — d'où l'usage de plusieurs initialisations (option `n_init` de scikit-learn), dont on garde la meilleure.

Combien de clusters ? Le vrai problème

Choisir k n'est pas évident

k-means exige de fixer k à l'avance. Mais combien de groupes y a-t-il « vraiment » dans les données ? Sans étiquette, personne ne le sait. Choisir k est le problème central du clustering, et il n'a pas de réponse unique — seulement des *guides*.

Deux outils complémentaires aident à trancher. La **méthode du coude** regarde comment l'inertie baisse quand k augmente : au-delà d'un certain k , ajouter des clusters n'aide presque plus — c'est le « coude » de la courbe. Le **score de silhouette** mesure, pour chaque point, s'il est bien plus proche de son cluster que du voisin : on choisit le k qui *maximise* ce score. Aucun n'est infallible ; on les croise avec le bon sens métier.

Le score de silhouette, formellement

Définition

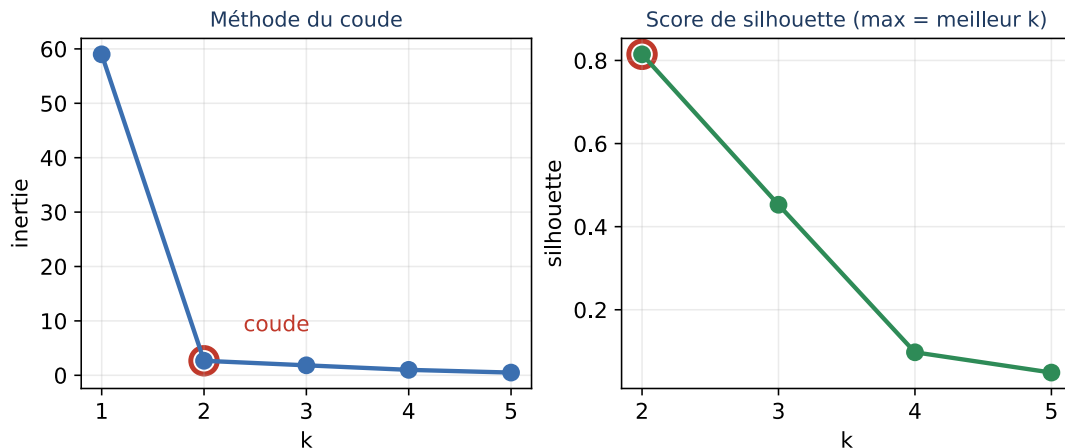
Pour un point, soit a sa distance moyenne aux points de son cluster, et b sa distance moyenne aux points du cluster *voisin le plus proche*. Le score de silhouette du point est :

$$s = \frac{b - a}{\max(a, b)} \in [-1, 1]$$

Lecture : s proche de 1 — le point est bien plus proche de son cluster que du voisin (bien classé) ; s proche de 0 — il est à la frontière ; s négatif — il serait mieux dans le cluster voisin (mal classé). On moyenne s sur tous les points : le k qui donne la *plus haute* silhouette moyenne est le meilleur candidat. Sur nos six points, la silhouette vaut 0,81 pour $k = 2$ et chute ensuite — elle confirme deux clusters.

Coude et silhouette, en image

Choisir k : le coude de l'inertie et le maximum de silhouette



À gauche, l'inertie chute fortement jusqu'à $k = 2$ puis ralentit nettement : le « coude » désigne $k = 2$. À droite, la silhouette est maximale à $k = 2$ (0,81) puis décroît. Les deux outils convergent : deux clusters. Quand ils divergent, c'est au jugement (et au métier) de trancher.

Piège — k-means est sensible à l'échelle

Standardiser avant de regrouper

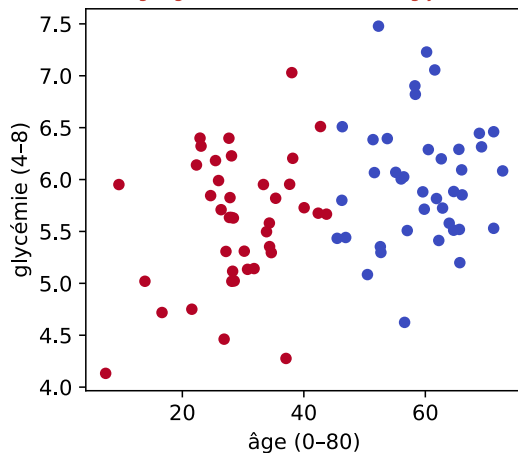
k-means repose sur la distance ; or une variable à grande échelle (l'âge, 0 à 80) domine une variable à petite échelle (la glycémie, 4 à 8) dans le calcul de distance. Sans **standardisation**, les clusters ne reflètent quasiment que la variable la plus étalée — un biais grossier.

C'est le même impératif qu'en Séances 4 et 9 pour tout modèle géométrique : ramener chaque variable à une échelle comparable (moyenne 0, écart-type 1) *avant* de lancer k-means. Sinon, regrouper des patients « par âge uniquement » sans le vouloir. La standardisation n'est pas optionnelle : c'est une étape de la méthode.

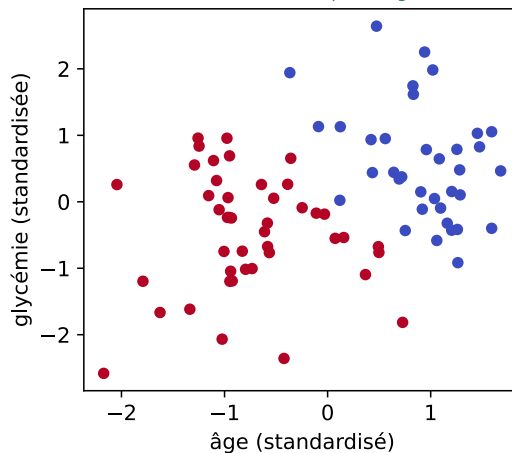
L'effet de l'échelle, en image

k-means est sensible à l'échelle : toujours standardiser

Sans standardisation :
l'âge (grande échelle) écrase la glycémie



Avec standardisation :
les deux variables comptent également



À gauche, sans standardisation : k-means sépare presque uniquement selon l'âge (axe horizontal, grande échelle), ignorant la glycémie. À droite, après standardisation : les deux variables pèsent également, et les clusters ont un sens. Toujours standardiser avant un clustering.

Piège — k-means suppose des clusters sphériques

Une forme imposée aux données

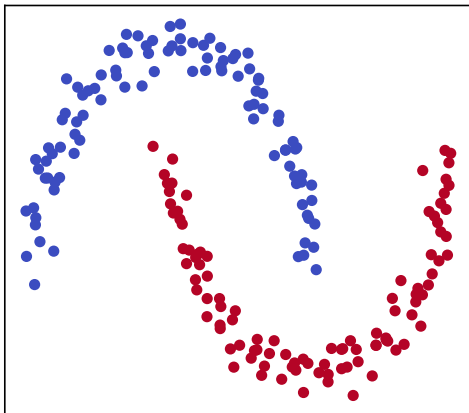
k-means cherche des clusters *compacts et ronds* (sphériques), de tailles comparables. Quand la structure réelle est allongée, imbriquée ou de densités très inégales, il échoue — il découpe l'espace en blocs ronds qui ne correspondent pas aux vrais groupes.

C'est une limite *de fond*, pas un réglage : l'algorithme ne peut pas représenter des formes arbitraires. Pour des structures complexes, d'autres méthodes existent (clustering hiérarchique, DBSCAN, mélanges gaussiens). Mais pour des groupes globalement ronds et bien séparés — le cas le plus fréquent — k-means reste l'outil de référence : simple, rapide, interprétable.

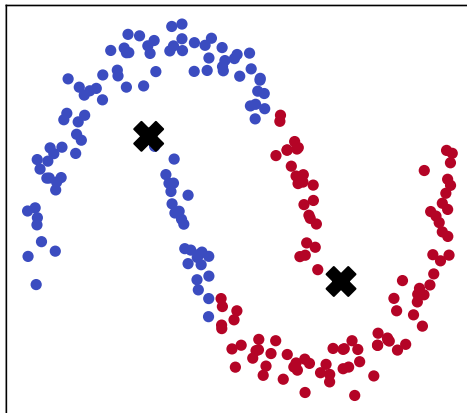
La limite de forme, en image

La limite de k-means : il suppose des clusters sphériques

Structure réelle : deux lunes imbriquées



k-means coupe en deux blocs ronds : ÉCHOUÉ



La structure réelle (à gauche) est deux lunes imbriquées. k-means (à droite) ne peut tracer que des frontières droites entre centres : il coupe les lunes en deux blocs ronds, à contresens de la vraie structure. Connaître cette limite, c'est savoir quand k-means *n'est pas* le bon outil.

En pratique : KMeans

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

X = StandardScaler().fit_transform(df[colonnes])    # OBLIGATOIRE
kmeans = KMeans(
    n_clusters=3,      # le nombre de clusters, a fixer
    n_init=10,         # 10 initialisations, on garde la meilleure
    random_state=0)
labels = kmeans.fit_predict(X)    # le cluster de chaque point
print(kmeans.inertia_)           # inertie finale
print(kmeans.cluster_centers_)   # les centres
```

`fit_predict` renvoie directement le cluster de chaque exemple. `n_init=10` relance dix fois avec des initialisations différentes et conserve la meilleure (inertie la plus basse) — c'est la parade au piège du minimum local. La standardisation préalable est, là encore, indispensable.

Exercice — une étape de k-means

Énoncé

Quatre points en 1D : $x_1 = 1$, $x_2 = 2$, $x_3 = 8$, $x_4 = 9$. On vise $k = 2$, avec centres initiaux $c_1 = 1$ et $c_2 = 2$. (a) Affecter chaque point au centre le plus proche. (b) Recalculer les deux centres. (c) Une nouvelle affectation changerait-elle quelque chose ?

Correction

(a) Distances : x_1 à c_1 (0) → cluster 1 ; x_2 à c_2 (0) → cluster 2 ; $x_3 = 8$ est à 7 de c_1 et 6 de c_2 → cluster 2 ; $x_4 = 9$ idem → cluster 2. Clusters : $\{1\}$ et $\{2, 8, 9\}$. **(b)** $c_1 = 1$, $c_2 = (2 + 8 + 9)/3 = 6,33$. **(c)** Nouvelle affectation : $x_2 = 2$ est maintenant à 1 de c_1 et 4,33 de c_2 → il rejoint le cluster 1. Les clusters deviennent $\{1, 2\}$ et $\{8, 9\}$ — oui, ça change : l'algorithme n'a pas encore convergé.

Fondations : variance, covariance, vecteurs propres

Pourquoi ces outils maintenant

Préparer la réduction de dimension

La seconde tâche du non-supervisé — réduire le nombre de variables — repose sur une idée géométrique : trouver les *directions* où les données s'étalent le plus. Formaliser cette idée demande trois notions : la **variance** (l'étalement d'une variable), la **covariance** (comment deux variables varient ensemble), et les **vecteurs propres** (les axes privilégiés d'une matrice). On les introduit avec intuition, formule et chiffres.

Ces outils ne sont pas des digressions : ils *sont* la mécanique de l'ACP. Les voir clairement maintenant rendra la PCA limpide ensuite — au lieu d'une formule magique. Chacune en une diapo : l'idée, l'écriture, un exemple numérique.

La variance — l'étalement d'une variable

Définition

La **variance** mesure à quel point les valeurs d'une variable s'écartent de leur moyenne. Petite variance : valeurs tassées autour de la moyenne ; grande variance : valeurs très dispersées.

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

On centre chaque valeur (soustraire la moyenne $\bar{x}$), on élève au carré (pour compter les écarts dans les deux sens), on moyenne. Exemple : pour $x = (2, 4, 6, 8)$, moyenne $\bar{x} = 5$, et $\text{Var}(x) = \frac{1}{4}(9 + 1 + 1 + 9) = 5$. La variance est la brique : la PCA cherchera les directions de *variance maximale*, car c'est là que se trouve l'information.

La covariance — deux variables ensemble

Définition

La **covariance** mesure si deux variables varient *ensemble* : positive si elles montent ensemble, négative si l'une monte quand l'autre baisse, nulle si elles sont indépendantes (linéairement).

$$\text{Cov}(x, y) = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$

Même esprit que la variance, mais en croisant deux variables. Exemple : $a = (1, 2, 3, 4)$ et $b = (2, 4, 6, 8) = 2a$ varient parfaitement ensemble $\rightarrow \text{Cov}(a, b) = 2,5 > 0$. Quand on a plusieurs variables, on range toutes les covariances dans une **matrice de covariance** (symétrique) : variances sur la diagonale, covariances ailleurs. C'est cette matrice que la PCA va analyser.

Les vecteurs propres — les axes d'une matrice

Définition

Un **vecteur propre** d'une matrice A est une direction qui, transformée par A , ne change pas d'orientation — seulement de longueur. Le facteur d'allongement est la **valeur propre** λ associée.

$$A \mathbf{v} = \lambda \mathbf{v}$$

$\mathbf{v}$ est un axe « privilégié » de A : la transformation se contente de l'étirer (ou contracter) d'un facteur λ , sans le faire tourner. Appliqué à une *matrice de covariance*, ce concept est l'or de la PCA : les vecteurs propres pointent les directions de variance des données, et les valeurs propres *mesurent* cette variance. La plus grande valeur propre désigne la direction où les données s'étalent le plus.

Vecteurs propres — exemple chiffré

Une matrice de covariance jouet

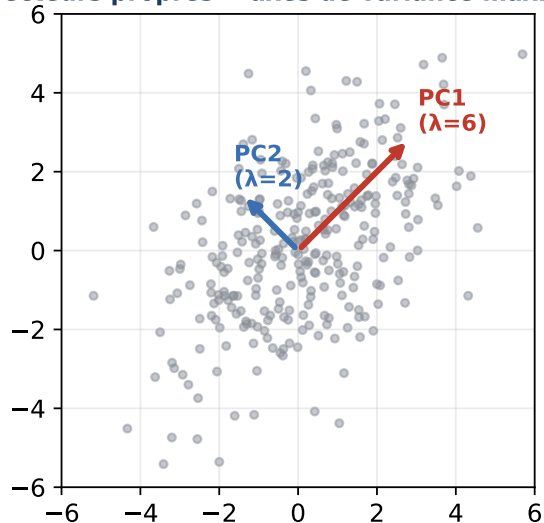
Soit la matrice de covariance $C = \begin{pmatrix} 4 & 2 \\ 2 & 4 \end{pmatrix}$ (deux variables de variance 4, covariance 2). Ses valeurs et vecteurs propres :

- valeurs propres : $\lambda_1 = 6$ et $\lambda_2 = 2$;
- vecteur propre de $\lambda_1 = 6$: $\mathbf{v}_1 = (0,707; 0,707)$, soit la *diagonale* $(1, 1)/\sqrt{2}$;
- vérification : $C \mathbf{v}_1 = (4,24; 4,24) = 6 \times (0,707; 0,707) = \lambda_1 \mathbf{v}_1$ ✓

La direction de plus grande variance est la diagonale (où les deux variables, corrélées, s'étalent de concert), et elle porte $\lambda_1/(\lambda_1 + \lambda_2) = 6/8 = 75$ % de la variance totale. C'est exactement ce que la PCA va exploiter.

Les vecteurs propres, en image

Vecteurs propres = axes de variance maximale



Un nuage de points de covariance $\begin{pmatrix} 4 & 2 \\ 2 & 4 \end{pmatrix}$. Le vecteur propre principal (rouge, $\lambda = 6$) pointe la direction où le nuage s'étire le plus — la diagonale. Le second (bleu, $\lambda = 2$), perpendiculaire, capte l'étalement résiduel. Les vecteurs propres de la covariance *sont* les axes naturels des données.

La réduction de dimension : l'ACP

L'intuition : compresser sans trop perdre

Beaucoup de variables, peu d'information ?

Un jeu de données peut avoir des dizaines de variables, souvent redondantes (taille et poids, âge et ancienneté...). Les visualiser ou les traiter devient lourd. L'idée de l'**analyse en composantes principales** (ACP, ou PCA) : remplacer les variables d'origine par un petit nombre de *nouvelles* variables — les composantes principales — qui captent l'essentiel de l'information.

« L'information », ici, c'est la **variance** : une direction où les données s'étalent beaucoup porte de l'information ; une direction où elles ne bougent presque pas est négligeable. La PCA trouve les directions de plus grande variance (les vecteurs propres de la covariance, qu'on vient de voir) et y projette les données. On peut alors garder seulement les premières — réduire la dimension — en ne perdant qu'un peu de variance.

L'idée : projeter sur les axes de variance maximale

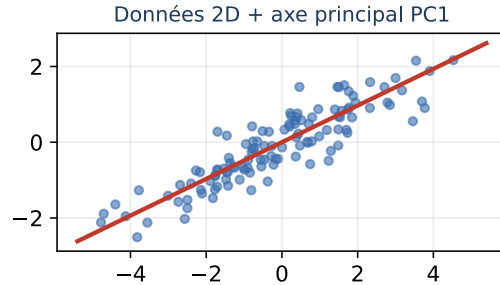
L'idée en une phrase

La **PCA** calcule les directions orthogonales de variance décroissante (les *composantes principales*, vecteurs propres de la matrice de covariance), puis projette les données dessus. Garder les premières composantes, c'est réduire la dimension en conservant le maximum de variance.

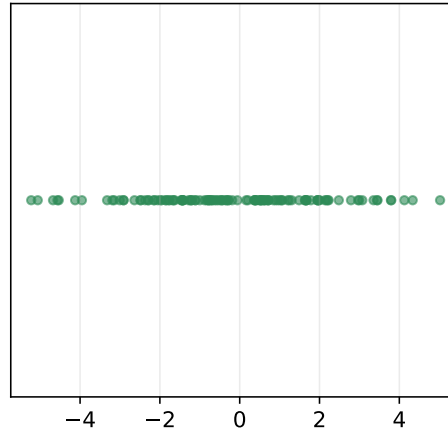
La première composante (PC1) est la direction de plus grande variance ; la deuxième (PC2), perpendiculaire, la plus grande variance restante ; et ainsi de suite. Chaque composante a une **valeur propre** qui dit *combien* de variance elle porte. Projeter sur les r premières composantes remplace d variables par $r < d$ nouvelles, en gardant la part de variance $\sum_{i \leq r} \lambda_i / \sum_i \lambda_i$. C'est la compression : moins de dimensions, presque toute l'information.

La projection, en image

La PCA projette sur les directions de plus grande variance



Projection sur PC1 : réduit à 1 dimension



À gauche, un nuage 2D et son axe principal PC1 (rouge), la direction de plus grande variance. À droite, les points *projetés* sur PC1 : on est passé de deux dimensions à une seule, en conservant l'essentiel de l'étalement. C'est la réduction de dimension : moins de variables, presque autant d'information.

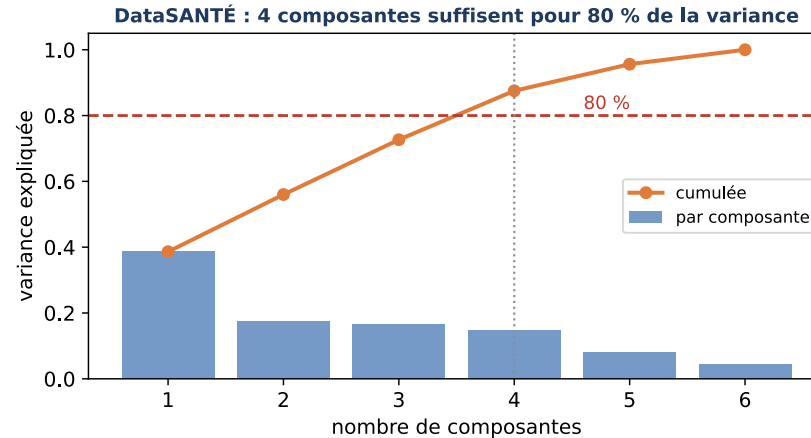
La variance expliquée — combien garder ?

Définition

La **variance expliquée** d'une composante est la part de la variance totale qu'elle capte : $\lambda_i / \sum_j \lambda_j$. En la cumulant sur les premières composantes, on sait combien de variance on conserve en n'en gardant qu'un certain nombre.

C'est le critère pour choisir le nombre de composantes : on garde assez de composantes pour atteindre un seuil de variance (souvent 80 ou 90 %). Le graphique de la variance cumulée (le « scree plot ») montre où s'arrêter : au point où ajouter une composante n'apporte plus grand-chose. Sur notre exemple jouet, PC1 portait déjà 75 % de la variance — une seule composante suffirait presque. Voyons sur DataSANTÉ.

PCA sur DataSANTÉ — la variance cumulée

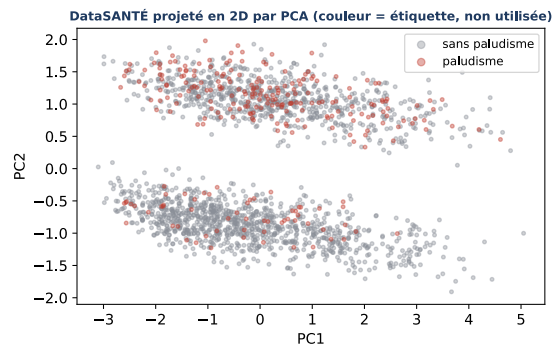


Six variables standardisées de DataSANTÉ. Les barres (bleu) donnent la variance de chaque composante ; la courbe (orange) la variance cumulée. Pour atteindre 80 % de variance, il faut 4 composantes (ligne rouge). On pourrait donc résumer six variables par quatre, en ne perdant que 20 % de l'information — ou par deux pour *visualiser*, au prix d'une perte plus forte.

Visualiser DataSANTÉ en deux dimensions

Projeter pour voir

Un usage majeur de la PCA : ramener des données à **deux** composantes pour les *visualiser* sur un plan — impossible autrement avec six variables. Sur DataSANTÉ, les deux premières composantes conservent 56 % de la variance : assez pour une vue d'ensemble.



Chaque point est un patient (PC1–PC2). La couleur (rouge = paludisme) *n'a pas servi* à la projection : on devine pourtant un gradient — la PCA, sans étiquette, organise déjà les patients d'une façon liée à la maladie.

Piège — la PCA exige la standardisation et reste linéaire

Deux limites à connaître

(1) Standardisation : la PCA maximise la variance ; sans standardisation, une variable à grande échelle accapare les premières composantes, comme pour k-means. Toujours standardiser avant. **(2) Linéarité** : la PCA ne capte que des directions *droites* de variance ; une structure courbe ou non-linéaire lui échappe.

Autre limite : les composantes principales sont des *combinaisons* des variables d'origine, souvent difficiles à interpréter (« $0,4 \times \text{âge} - 0,3 \times \text{glycémie} + \dots$ »). La PCA est excellente pour compresser, visualiser et débruiter, mais elle sacrifie l'interprétabilité directe des variables. Comme toujours, un outil puissant a son revers : ici, la lisibilité.

En pratique : PCA

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

X = StandardScaler().fit_transform(df[colonnes]) # OBLIGATOIRE
pca = PCA(n_components=2) # garder 2 composantes
X_reduit = pca.fit_transform(X) # les donnees en 2D

print(pca.explained_variance_ratio_) # variance par composante
print(pca.explained_variance_ratio_.sum()) # variance totale conservee
```

`n_components=2` demande deux composantes (pour visualiser) ; on peut aussi passer un seuil, par exemple `PCA(n_components=0.8)` pour garder automatiquement assez de composantes pour 80 % de variance.

`explained_variance_ratio_` donne la part de variance de chaque composante — le critère de choix. Standardiser avant reste impératif.

Application : la structure cachée de DataSANTÉ

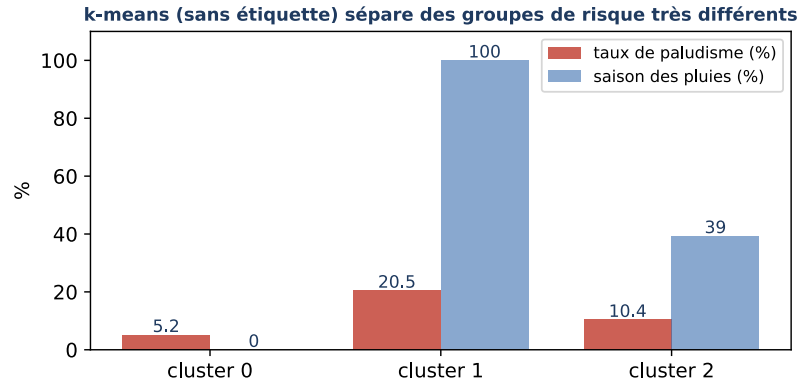
Le mini-défi : le clustering connaît-il le paludisme ?

Une question révélatrice

Test décisif du non-supervisé : on lance k-means sur les patients de DataSANTÉ *sans jamais lui montrer l'étiquette paludisme*. Question : les clusters qu'il forme ont-ils un lien avec la maladie ? Si oui, c'est la preuve que la structure non-supervisée capte quelque chose de réel — alors qu'aucune réponse ne lui a été donnée.

On standardise cinq variables (âge, glycémie, hémoglobine, fièvre, saison), on demande $k = 3$ clusters, puis — *seulement après* — on regarde le taux de paludisme dans chaque cluster. Le clustering n'a aucune information sur la maladie ; tout lien observé émerge de la seule structure des données.

Le verdict : des groupes de risque très différents



Sans voir l'étiquette, k-means forme trois groupes aux taux de paludisme *très* différents : cluster « pluies » (100 %) avec **20,5 %** de paludisme, contre cluster « sec » (0 %) à **5,2 %** (global 11,8 %). Le clustering a retrouvé, seul, un facteur de risque majeur.

La leçon de l'application

Le non-supervisé révèle du réel

Le fait marquant : un algorithme à qui l'on n'a donné *aucune* étiquette retrouve une structure liée au diagnostic. Cela montre que les regroupements ne sont pas arbitraires — ils reflètent des profils cliniques réels. C'est toute la valeur du non-supervisé : faire émerger des structures *avant* toute annotation, parfois là où on ne les attendait pas.

Usages concrets : segmenter une patientèle pour adapter la prise en charge, repérer des profils atypiques (détection d'anomalies), ou explorer un jeu de données neuf pour orienter les analyses suivantes. Le non-supervisé est souvent la *première* étape d'un projet — comprendre la structure — avant de passer, si besoin, au supervisé.

Synthèse

Clustering et réduction : deux usages distincts

Quand utiliser quoi

besoin	outil
regrouper des exemples similaires	clustering (k-means)
segmenter une patientèle, des profils	clustering (k-means)
visualiser des données à > 2 variables	PCA (2 composantes)
compresser / débruiter avant un modèle	PCA (seuil de variance)
détecter des structures inconnues	clustering puis PCA

Le protocole non-supervisé

De bout en bout

1. Standardiser les variables (impératif pour la distance comme pour la variance) → **2.** pour *regrouper* : k-means avec `n_init` élevé, choisir k par le coude et la silhouette → **3.** pour *réduire* : PCA, choisir le nombre de composantes par la variance cumulée (seuil 80–90 %) → **4.** *interpréter* les clusters ou les composantes au regard du métier → **5.** se rappeler les limites : k-means suppose des clusters ronds, la PCA est linéaire. En cas de structure complexe, envisager d'autres méthodes.

À retenir — Séance 9 (1/2)

Le clustering

- Le **non-supervisé** cherche une structure *sans étiquettes* : regrouper (clustering) ou simplifier (réduction de dimension).
- **k-means** alterne *affectation* (chaque point au centre le plus proche) et *mise à jour* (chaque centre = moyenne de ses points), jusqu'à convergence.
- Il minimise l'**inertie** (compacité des clusters) ; il converge vers un *minimum local* — d'où `n_init` élevé.
- Choisir k : la **méthode du coude** (inertie) et le **score de silhouette** (maximiser). Toujours **standardiser** ; attention aux clusters non sphériques.

À retenir — Séance 9 (2/2)

La réduction de dimension

- Fondations : **variance** (étalement), **covariance** (variation conjointe), **vecteurs propres** ($A\mathbf{v} = \lambda\mathbf{v}$, les axes d'une matrice).
- La **PCA** projette les données sur les directions de variance maximale (vecteurs propres de la covariance) ; les valeurs propres mesurent la variance captée.
- Choisir le nombre de composantes par la **variance cumulée** (seuil 80–90 %). Sur DataSANTÉ : 4 composantes pour 80 %.
- Limites : **standardiser** d'abord ; la PCA est *linéaire* et ses composantes sont peu interprétables.

Pièges fréquents

- Oublier de **standardiser** : les variables à grande échelle dominent la distance (k-means) et la variance (PCA).
- Lancer k-means une seule fois : risque de tomber sur un mauvais minimum local — utiliser `n_init`.
- Choisir k au hasard : croiser le coude, la silhouette et le bon sens métier.
- Appliquer k-means à des clusters non sphériques (lunes, anneaux) : il échouera.
- Garder trop peu de composantes en PCA : vérifier la variance cumulée avant de réduire.
- Interpréter les composantes principales comme des variables d'origine : ce sont des combinaisons, souvent opaques.

Prochaine séance

Séance 10 — Les réseaux de neurones

Retour au supervisé, mais avec le modèle le plus emblématique de l'IA moderne : le **réseau de neurones**. On verra que sa brique de base est la régression logistique de la Séance 6, comment empiler des couches, et — pour les curieux — comment un réseau apprend par rétropropagation. Et l'on posera la question honnête : sur nos données, bat-il vraiment les méthodes d'ensemble de la Séance 8 ?

Références

- Géron — *Hands-On Machine Learning*, 3^e éd., O'Reilly, 2022 (clustering, réduction de dimension).
- Müller, Guido — *Introduction to Machine Learning with Python*, O'Reilly, 2016.
- James, Witten, Hastie, Tibshirani — *An Introduction to Statistical Learning*, 2^e éd., Springer, 2021 (chap. 12).
- Hastie, Tibshirani, Friedman — *The Elements of Statistical Learning*, Springer, 2009.
- Cours : University of Washington CSE 446 ; Stanford CS229 (apprentissage non-supervisé).

Ressources de la séance

- **Notebook** · [TP 9 — Apprentissage non supervisé](#) (397 Ko)
- **TD** · [Fiche d'exercices 9](#) (216 Ko)