

TD 6 — ACP : projection, reconstruction et inertie

Séance 6

Christophe Ambroise

2026-09-04

Objectifs

Démontrer la décomposition de Pythagore qui fonde l'ACP, construire l'ACP standardisée d'Iris sans utiliser les espèces, contrôler la reconstruction et ses pertes, puis établir le parallèle avec la décomposition intra-inter de k -means.

Les exercices portent sur Iris : 150 fleurs, quatre mesures en centimètres et une colonne **species**. Cette dernière reste séparée jusqu'à la dernière question. La convention adoptée dans tout le TD est

$$S = \frac{1}{n} X^T X,$$

où X désigne la matrice centrée, éventuellement standardisée.

Partie théorique

Exercice 1 — Projection orthogonale et Pythagore

Soit $X \in \mathbb{R}^{n \times p}$ une matrice centrée. Soit $F \subset \mathbb{R}^p$ un sous-espace de dimension q , de projecteur orthogonal P_F . La ligne x_i^T représente l'individu i .

1. Écrire x_i comme somme de sa projection $P_F x_i$ et de son résidu. Montrer l'orthogonalité des deux termes.
2. En déduire, pour chaque individu, puis pour tout le nuage,

$$I_T = I_{\text{proj}}(F) + I_{\text{res}}(F).$$

Identifier ces trois quantités comme des normes de Frobenius.

3. Montrer qu'à dimension q fixée, maximiser l'inertie projetée équivaut à minimiser l'inertie résiduelle.
4. Pour $q = 1$, écrire $F = \text{span}(u)$ avec $\|u\| = 1$. Montrer que

$$I_{\text{proj}}(F) = n u^T S u.$$

Résoudre le problème par multiplicateurs de Lagrange.

5. Généraliser à la dimension q et exprimer les deux inerties à partir des valeurs propres $\lambda_1 \geq \dots \geq \lambda_p$ de S .

Partie empirique

Exercice 2 — Spectre et choix de la dimension

Retirer `species`, puis standardiser chaque mesure avec l'écart-type population (diviseur n).

```
import numpy as np
import pandas as pd

iris = pd.read_csv("../data/iris.csv")
X_brut = iris.drop(columns="species")
X = (X_brut - X_brut.mean()) / X_brut.std(ddof=0)
n, p = X.shape

S = X.to_numpy().T @ X.to_numpy() / n
valeurs, axes = np.linalg.eigh(S)
ordre = np.argsort(valeurs)[::-1]
valeurs, axes = valeurs[ordre], axes[:, ordre]
```

```
iris_df <- read.csv("../data/iris.csv")
X_brut <- iris_df[, setdiff(names(iris_df), "species")]
n <- nrow(X_brut)
p <- ncol(X_brut)

moyennes <- colMeans(X_brut)
ecarts <- sqrt(colMeans(sweep(X_brut, 2, moyennes, "-")^2))
X <- as.matrix(sweep(sweep(X_brut, 2, moyennes, "-"), 2, ecarts, "/"))
S <- crossprod(X) / n
decomp <- eigen(S, symmetric = TRUE)
valeurs <- decomp$values
axes <- decomp$vectors
```

1. Vérifier numériquement les moyennes nulles et les variances unitaires.
2. Calculer S , ses valeurs propres et les taux d'inertie expliquée.
3. Vérifier les identités

$$\mathrm{tr}(S) = \sum_{j=1}^p \lambda_j = p, \quad I_T = n \mathrm{tr}(S) = np.$$

4. Compléter un tableau contenant, pour chaque axe, λ_j , le taux individuel et le taux cumulé.
5. Plus petite dimension atteignant 95 % d'inertie expliquée ? Que perd une règle fondée uniquement sur ce seuil ?

Exercice 3 — Reconstruction et qualité de représentation

Les scores complets sont $C = XU$. Pour $q \in \{1, 2, 3, 4\}$, la reconstruction est $\widehat{X}_q = C_q U_q^T$.

```
scores = X.to_numpy() @ axes

def reconstruction(scores, axes, q):
    return scores[:, :q] @ axes[:, :q].T

for q in range(1, p + 1):
```

```

Xq = reconstruction(scores, axes, q)
i_proj = np.sum(Xq ** 2)
i_res = np.sum((X.to_numpy() - Xq) ** 2)
# comparer à n * valeurs[q:].sum()

```

```

scores <- X %%% axes

reconstruction <- function(scores, axes, q) {
  scores[, seq_len(q), drop = FALSE] %%%
  t(axes[, seq_len(q), drop = FALSE])
}

for (q in seq_len(p)) {
  Xq <- reconstruction(scores, axes, q)
  I_proj <- sum(Xq^2)
  I_res <- sum((X - Xq)^2)
  # comparer à n * sum(valeurs[(q + 1):p]) si q < p
}

```

1. Calculer I_{proj} et I_{res} pour les quatre valeurs de q . Vérifier $I_T = I_{\text{proj}} + I_{\text{res}}$.
2. Vérifier numériquement

$$I_{\text{res}}(F_q) = n \sum_{j>q} \lambda_j.$$

3. Calculer, pour chaque individu, la qualité dans le premier plan :

$$\cos^2(i, F_2) = \frac{c_{i1}^2 + c_{i2}^2}{\|x_i\|^2}.$$

Identifier l'individu le moins bien représenté et la qualité médiane.

4. Calculer les corrélations variable–composante $\sqrt{\lambda_k} u_{jk}$. Interpréter les deux premiers axes sans consulter `species`.
5. Colorer ensuite le plan par espèce. Pourquoi cette coloration n'est-elle ni un apprentissage supervisé, ni une validation de l'ACP ?

Exercice 4 — Deux décompositions de l'inertie totale

Appliquer k -means à $K = 3$ sur les quatre variables standardisées, toujours sans la colonne `species`.

```

from sklearn.cluster import KMeans

groupes = KMeans(n_clusters=3, n_init=50, random_state=0).fit_predict(X)

def inerties_partition(X, groupes):
    X = np.asarray(X)
    centre = X.mean(axis=0)
    i_t = np.sum((X - centre) ** 2)
    i_w = i_b = 0.0
    for k in np.unique(groupes):
        bloc = X[groupes == k]
        centre_k = bloc.mean(axis=0)
        i_w += np.sum((bloc - centre_k) ** 2)
        i_b += len(bloc) * np.sum((centre_k - centre) ** 2)
    return i_t, i_w, i_b

```

```

set.seed(0)
groupes <- kmeans(X, centers = 3, nstart = 50)$cluster

inerties_partition <- function(X, groupes) {
  centre <- colMeans(X)
  I_T <- sum(sweep(X, 2, centre, "-")^2)
  I_W <- I_B <- 0
  for (k in sort(unique(groupes))) {
    bloc <- X[groupes == k, , drop = FALSE]
    centre_k <- colMeans(bloc)
    I_W <- I_W + sum(sweep(bloc, 2, centre_k, "-")^2)
    I_B <- I_B + nrow(bloc) * sum((centre_k - centre)^2)
  }
  c(I_T = I_T, I_W = I_W, I_B = I_B)
}

```

1. Pour $i \in G_k$, développer $\|x_i - \mu\|^2$ après insertion de μ_k . Montrer que le terme croisé disparaît après sommation dans G_k .
2. En déduire le théorème de Huygens :

$$I_T = \underbrace{\sum_k \sum_{i \in G_k} \|x_i - \mu_k\|^2}_{I_W} + \underbrace{\sum_k n_k \|\mu_k - \mu\|^2}_{I_B}.$$

3. Calculer I_W et I_B . Vérifier l'égalité à la précision machine.
4. Comparer cette décomposition à celle de l'ACP pour $q = 2$. Dans chaque cas, identifier l'approximation, la partie expliquée et le résidu.
5. Consulter enfin **species** par une table croisée. Cette table intervient-elle dans l'optimisation de k -means ?

Remarque finale

! À retenir

Deux théorèmes de la variance, deux approximations :

$$\text{ACP : } I_T = I_{\text{proj}} + I_{\text{res}}, \quad k\text{-means : } I_T = I_W + I_B.$$

Le premier choisit un sous-espace ; le second choisit une partition.

Scripts complets et exécutables :

- `python/td06_acp.py` ;
- `r/td06_acp.R`.