Projet Ma315 - BE

Institut Polytechnique des Sciences Avancées (IPSA)

Une première approche de l'analyse en composantes principales

Réalisé par :

Romain Lefol, Killian Belloc, Jean Tixeront, Willen Arab, Guilhem Perret-bardou, Hugo Dubouchet

Encadrants :

M. G. Couffignal, M. M. El-Mahbouby

Année académique : 2024-2025

30 Janvier 2025

Table des matières

Table des Illustrations

Introduction à l'ACP

Dans ce BE nous allons définir et programmer la méthode dite d'**analyse en composantes principales (ACP)** ou **principal component analysis (PCA)** en anglais. Cette méthode a été conçue par l'un des fondateurs de la statistique moderne Karl Pearson dans les années 1900. Depuis l'avènement de l'ordinateur, augmentant exponentiellement les capacités de calculs que demande la réalisation de cette méthode, on retrouve l'analyse en composantes principales dans tous les domaines des sciences pour la recherche de structures dans des données, le débruitage, la réduction en dimension de problème, etc. L'ACP est aujourd'hui un élément de base en analyse (linéaire) des données. C'est une méthode de réduction de dimension qui cherche à fournir la meilleure projection linéaire des données initiales vers un espace de dimension plus faible : l'idée est d'identifier des relations cachées entre des variables en transformant les données dans un espace plus simple, tout en préservant au maximum leur structure globale. En pratique, elle permet de condenser l'information la plus riche (du point de vue de la variance) sur un petit nombre de nouvelles variables appelées composantes principales. Lorsque l'on retient seulement les deux premières composantes, l'ACP offre ainsi une visualisation en deux dimensions qui préserve au maximum la dispersion originale des données, facilitant l'exploration et la détection d'éventuelles structures (groupes, tendances) au sein du jeu de données. Cette approche est souvent un incontournable dans de nombreux domaines : pour explorer rapidement la distribution de données d'images (par ex. reconnaissance faciale), pour identifier des regroupements naturels en génomique (analyses de SNP, d'expressions de gènes), pour détecter des profils de clients en marketing (clustering, identification de segments), ou simplement pour visualiser et expliquer la variabilité dominante au sein d'un large tableau de données. C'est donc un outil (statistique) exploratoire de premier plan, offrant un résumé visuel et compréhensible de données multidimensionnelles complexes. Ce BE sera une introduction très limitée aux possibilités de l'ACP, nous n'en verront que les prémices.

1 Partie 1 : Un peu de théorie

Soit $X := (X_1, X_2, \cdots, X_n)$ un vecteur aléatoire défini sur un espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$ **(les vecteurs aléatoires seront considérés comme des vecteurs lignes).** On supposera que les variables aléatoires X_i **sont centrées**, i.e. $\mathbb{E}(X_i) = 0$, et **réduites :** $\mathrm{Var}(X_i) = 1$. Attention cela ne signifie pas que les X_i suivent les mêmes lois.

Nous allons rechercher des variables aléatoires qui sont combinaisons linéaires des composantes du vecteur aléatoire X :

- qui rendent compte au mieux de la dispersion de X (hypothèse de maximisation de la variance) et,
- décorrélées entre elles (covariances nulles entre elles).

L'idée derrière ces deux hypothèses est de rechercher des structures/structurations (linéaires) cachées sur le système décrit par le vecteur aléatoire X. En effet, pour un système étudié décrit par un vecteur aléatoire X, dont les composantes sont des variables aléatoires X_i, ces variables aléatoires X_i ne sont pas nécessairement les meilleures façons de considérer ce système.

L'analyse en composantes principales (ACP) formalise le fait de rechercher des nouvelles variables aléatoires plus adaptées pour décrire notre système. Dans l'analyse en composantes principales, on impose que ces nouvelles variables aléatoires soient des combinaisons linéaires des X_i. Ce critère est purement pragmatique : empiriquement, on constate que cela donne de bons résultats, et des généralisations non-linéaires comme l'analyse en composantes curvilignes ou l'ACP à noyau peuvent être très coûteuses en temps de calcul voire pas toujours réalisables. Les concepts présentés dans cette première partie seront directement appliqués lors de l'implémentation en Python dans la suite.

Formalisons le problème précédent. On définit la première composante principale comme le vecteur aléatoire :

$$Y_1 := Xv_1 = \sum_{i=1}^{n} v_{1i} X_i$$

où v_1 est un vecteur colonne de $\mathbb{R}^n$ de norme égale à 1 (i.e. appartenant à la sphère unité) et tel que :

$$v_1 = \arg \max_{\|v\|_2=1,\, v\in\mathbb{R}^n} \mathrm{Var}(Xv)$$

On appelle v_1 **la première direction principale.** Par récurrence, on définit la $(k+1)$-ème **composante principale de** X de la manière suivante. Supposons que les k-premières composantes principales $Y_1, Y_2, \cdots, Y_k$ sont connues (construites). On cherche alors un vecteur aléatoire :

$$Y_{k+1} := Xv_{k+1} = \sum_{i=1}^{n} v_{(k+1)i} X_i$$

où v_{k+1} est un vecteur de $\mathbb{R}^n$ de norme égale à 1 (i.e. appartient à la sphère unité) et tel que :

$$v_{k+1} = \arg \max_{v\perp \mathrm{Vect}(v_1, v_2, \cdots, v_k),\, \|v\|_2=1,\, v\in\mathbb{R}^n} \mathrm{Var}(Xv)$$

avec $v \perp \text{Vect}(v_1, v_2, \cdots, v_k)$ signifiant que v appartient à l'orthogonal de l'espace vectoriel engendré par les vecteurs $v_1, v_2, \cdots, v_k$.

Nous allons montrer que ces vecteurs existent bien, i.e. que les problèmes énoncés précédemment ont bien une solution.

1.1 Matrice de covariance de X

Montrons que :

$$\text{Var}(Xv) = v^T \text{cov}(X) v$$

on note Y = Xv et :

$$\text{Var}(Y) = E(Y^2) - (E(Y))^2$$

Comme X est centré, on a $E(X) = 0$.
On a donc :

$$\text{Var}(Xv) = E[(Xv)(Xv)^T]$$

$$= E(v^T X^T X v)$$

Par linéarité :

$$\text{Var}(Xv) = v^T E(X^T X) v$$

Or, par définition, on sait que :

$$\text{Cov}(X) = E(X^T X)$$

Donc :

$$\boxed{\text{Var}(Xv) = v^T \text{Cov}(X) v}$$

où Cov(X) est la matrice de covariance du vecteur aléatoire X. —

1.2 Calcul de la première composante principale

On veut résoudre :

$$(\mathcal{P}) : \underset{||v||_2=1, v\in\mathbb{R}^n}{\arg\max} \; v^T \text{Cov}(X) v$$

sous la contrainte :

$$\|v\|_2 = 1$$

On a :

$$\mathcal{L}(v, \lambda) = v^T \text{Cov}(X) v - \lambda(\|v\|_2^2 - 1)$$

La fonction de Lagrange associée à $(\mathcal{P})$.
On cherche les points stationnaires de $\mathcal{L}(v, \lambda)$:

$$\nabla_v \mathcal{L} = 2\text{Cov}(X)v - 2\lambda v = 0$$

donc :

$$\boxed{\text{Cov}(X)v = \lambda v}$$

v est donc un vecteur propre de $\text{Cov}(X)$ associé à la valeur propre λ.

1.3 Diagonalisation dans une base orthonormée

$\mathrm{Cov}(X)$ est une matrice symétrique.

Définition :

- Une matrice $A \in M_n(\mathbb{R})$ est symétrique si $A = A^T$.
- Toute matrice symétrique est diagonalisable dans une base orthonormée (Théorème Spectral).

$$\mathrm{Cov}(X) = E\left[(X - E(X))(X - E(X))^T\right]$$

Elle est symétrique car :

$$\mathrm{Cov}(X)^T = \mathrm{Cov}(X)$$

D'après le théorème spectral :

$$\mathrm{Cov}(X) = VDV^T$$

où :

- V est une matrice orthogonale contenant les vecteurs propres de $\mathrm{Cov}(X)$,
- D est une matrice diagonale contenant les valeurs propres de $\mathrm{Cov}(X)$.

$$D = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix}$$

1.4 Calcul de la k-composante principale de X

D'après la question précédente :

$$\mathrm{Cov}(X) = VDV^T$$

Par définition :

$$Y_k = Xv_k$$

Car l'ACP consiste à projeter les données de X sur les directions principales. Ainsi, chaque Y est une combinaison linéaire de X pondérée par le vecteur propre associé. On a ainsi :

$$\mathrm{Var}(Y_k) = \mathrm{Var}(Xv_k)$$

$$\Leftrightarrow \mathrm{Var}(Y_k) = v_k^T \mathrm{Cov}(X) v_k = v_k^T \lambda_k v_k = \lambda_k$$

1.5 Estimateur de la matrice de covariance

Les questions précédentes supposent la connaissance du vecteur aléatoire X et par suite la connaissance des variables aléatoires X_i. Dans la pratique, il se peut que ces lois ne soient pas connues, mais que nous ayons accès à des échantillons du vecteur aléatoire X. Nous allons voir comment il est possible de donner un estimateur de la matrice de covariance de X à partir d'échantillons de X. On ne supposera pas dans les questions suivantes que les composantes X_i de X sont des variables aléatoires réduites et centrées. Soit $(\mathrm{Ind}_1, \mathrm{Ind}_2, \dots, \mathrm{Ind}_m)$ un m-échantillon du vecteur aléatoire X.

Un échantillon d'un vecteur aléatoire se représente sous la forme d'un tableau/matrice :

	X_1	X_2	$\cdots$	X_n
Ind_1	X_{11}	X_{12}	$\cdots$	X_{1n}
Ind_2	X_{21}	X_{22}	$\cdots$	X_{2n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Ind_m	X_{m1}	X_{m2}	$\cdots$	X_{mn}

Figure 1: Échantillon d'un vecteur aléatoire

On notera la matrice E associée à ces données. Cette matrice est de taille (m, n) et est à coefficient des fonctions $X_{ij} : \Omega \to \mathbb{R}$. Ainsi un vecteur colonne de E :

$$E_i := (X_{1i}, X_{2i}, \dots, X_{mi})^T$$

est la donnée d'un m-échantillon de la variable aléatoire X_i. On notera classiquement :

$$\overline{X_{mi}} := \frac{1}{m}\sum_{k=1}^{m} X_{ki} \quad \text{et} \quad \overline{S}_{mi} := \frac{1}{m-1}\sum_{k=1}^{m}(X_{ki} - \overline{X_{mi}})^2$$

les estimateurs sans biais et convergent de l'espérance $E(X_i)$ et de la variance $\mathrm{Var}(X_i)$ respectivement. Nous prendrons comme estimateur de l'écart-type :

$$\overline{\sigma_{mi}} := \sqrt{\overline{S_{mi}}}$$

1.5.1 Approximation d'un estimateur de la matrice de covariance de X

Définition de la matrice de covariance

$$\mathrm{Cov}(X) = \mathbb{E}\left[(X - \mathbb{E}(X))(X - \mathbb{E}(X))^\top\right]$$

Cependant, nous ne connaissons pas la loi de X. Estimons cette matrice à l'aide d'un échantillon :

La matrice centrée de E est donnée par :

$$(E_c) = X_{ij} - \overline{X_j}$$

où $\overline{X_j}$ est la moyenne de la $j$ème colonne :

$$\overline{X_j} = \frac{1}{m}\sum_{i=1}^{m} X_{ij}$$

On en déduit l'estimateur empirique de la matrice de covariance :

$$\mathrm{Cov}(X) = \frac{1}{m-1}E_c^T E_c$$

Effectuons une réduction pour rendre les données comparables entre elles, Normalisons chaque colonne et nous aurons la moyenne égale à 0 et la variance égale à 1.

Par définition, nous avons E_{cr} :

$$E_{cr} = \frac{X_{jk} - \overline{X_j}}{\sqrt{m-1}\sqrt{\overline{S_j}}}$$

$$\overline{S_j} = \frac{1}{m-1}\sum_{i=1}^{m}(X_{ij} - \overline{X_j})^2$$

et

$$\overline{\sigma_j} := \sqrt{\overline{S_j}}$$

Ainsi, E_{cr} est la division de chaque élément de E_c par l'écart-type estimé de sa colonne.
On trouve donc :

$$C = \frac{1}{m-1}E_{cr}^T E_{cr}$$

De plus, l'hypothèse donnée :

$$E\left(\left(\frac{X_{ik} - \overline{X}_{mk}}{\overline{\sigma}_{mk}}\right)\left(\frac{X_{jp} - \overline{X}_{mp}}{\overline{\sigma}_{mp}}\right)\right) \approx \frac{E\left((X_{ik} - \overline{X}_{mk})(X_{jp} - \overline{X}_{mp})\right)}{E(\overline{\sigma}_{mk})E(\overline{\sigma}_{mp})}$$

Nous indique que pour un échantillon ($m \gg n$), la réduction ne modifie pas les relations entre les variables. En effet, les covariances données par E_{cr} sont asymptotiquement équivalentes à celles de E_c.

1.5.2 Centrage et réduction de la matrice R

i. Centrage :
L'idée du centrage des données est de transformer une matrice $A \in M_{m,n}(\mathbb{R})$ en une matrice $A^{(c)} \in M_{m,n}(\mathbb{R})$ dont la moyenne des vecteurs lignes est nulle. Notons l_g le vecteur ligne moyen de A (c'est l'individu « moyen ») :

$$\boxed{l_g := \frac{1}{m}\sum_{i=1}^{m} l_i} \in M_{1,n}(\mathbb{R})$$

où l_i représentent les vecteurs lignes de A.

A. D'après l'énoncé, on nous donne l_g, le vecteur ligne moyen de $A \in M_{m,n}(\mathbb{R})$. Ici, le nombre de lignes est m et le nombre de colonnes est n. De plus, $l_g \in M_{1,n}$, car il a le même nombre de colonnes que A.

On pose :

$$l_g = \frac{1}{m}\sum_{i=1}^{m} l_i, \quad l_i \in M_{1,n}(\mathbb{R})$$

L'intérêt de cette équation est de sommer les lignes de la matrice A avant de les diviser par le nombre de lignes m, obtenant ainsi la moyenne de chaque composante de la somme des lignes de A.

Nous pouvons aussi écrire :

$$l_g = \frac{1}{m}\sum_{i=1}^{m} A_{i,j}, \forall j \in [1,n]$$

Nous reconnaissons que le produit matriciel entre un vecteur ligne unitaire et une matrice $A \in M_{m,n}(\mathbb{R})$ donne la somme des lignes de A. Dans notre bureau d'étude, le vecteur colonne $\mathbf{u}$ est le vecteur colonne unitaire $\mathbf{1}_m$ (ne contenant que des 1) que l'on transposera.

Ainsi, nous avons d'après les propriétés du calcul matriciel que :

$$\mathbf{1}_m^T A$$

nous donne la somme des lignes de A et $\mathbf{1}_m$ agit comme opérateur de sommation des lignes de A. Nous en déduisons que :

$$\mathbf{1}_m^T A = \sum_{i=1}^{m} l_i$$

$$\Leftrightarrow \frac{1}{m}\mathbf{1}_m^T A = \frac{1}{m}\sum_{i=1}^{m} l_i = l_g$$

B. D'après l'énoncé, l'idée du centrage est de transformer une matrice $A \in M_{m,n}(\mathbb{R})$ en une matrice $A^{(c)} \in M_{n,m}(\mathbb{R})$ dont la moyenne des vecteurs lignes est nulle.

Nous avons donc :

$$\frac{1}{m}\sum_{i=1}^{m} {l_i}^{(c)} = 0$$

Centrer cette matrice revient donc à soustraire la moyenne des lignes pour recentrer les données autour de 0.

Nous avons montré que :

$$l_g = \frac{1}{m}\mathbf{1}_m^T A$$

Cependant, cela nous donne uniquement un vecteur ligne. Afin de répéter l_g sur m lignes et de créer une matrice de taille $m \times n$, Effectuons un produit extérieur avec le vecteur colonne unitaire $\mathbf{1}_m$.
D'après l'intérêt du centrage, cela nous donne :

$$A^{(c)} = A - \mathbf{1}_m l_g$$

Donc :

$$A^{(c)} = A - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^T A$$

où $\frac{1}{m}\mathbf{1}_m\mathbf{1}_m^T A$ est la matrice dont chaque ligne est identique à l_g.

Par conséquent, nous avons soustrait la moyenne des lignes de A représentée par I_g à chaque ligne de A.

C. Nous donnons la formule :

$$A^{(c)} = A - \frac{1}{m}uu^T A$$

et d'après les propriétés de calcul matriciel, nous obtenons :

$$(A^{(c)})^{(c)} = A^{(c)} - \frac{1}{m}uu^T A^{(c)}$$

$$= A - \frac{1}{m}uu^T A - \frac{1}{m}uu^T(A - \frac{1}{m}uu^T A)$$

$$= A - \frac{1}{m}\mathbf{u}\mathbf{u}^T A - \frac{1}{m}\mathbf{u}\mathbf{u}^T A + \frac{1}{m^2}\mathbf{u}\mathbf{u}^T\mathbf{u}\mathbf{u}^T A$$

Sachant que :

$$\mathbf{u}\mathbf{u}^T = m$$

Nous obtenons alors :

$$(A^{(c)})^{(c)} = A - \frac{2}{m}\mathbf{u}\mathbf{u}^T A + \frac{1}{m}\mathbf{u}\mathbf{u}^T A$$

D'où :

$$(A^{(c)})^{(c)} = A - \frac{1}{m}\mathbf{u}\mathbf{u}^T A = A^{(c)}$$

ii. Réduction :

Soit une matrice $A \in M_{m,n}(\mathbb{R})$ que l'on pense comme un tableau de données. Les variables/paramètres mesurés représentés par les vecteurs colonnes, notons les colonnes de A : c_i, possèdent en général une dimension (des longueurs, vitesses, températures, etc.) et peuvent appartenir à des plages de valeurs difficilement comparables entre elles, i.e. les valeurs d'une colonne c_i peuvent être « très grandes » alors qu'une autre colonne c_j n'aura que des valeurs « petites ».

L'opération de réduction consiste à adimensionner les valeurs des colonnes et à les normaliser entre -1 et 1. Concrètement cela revient à remplacer les colonnes c_i par :

$$c_i \mapsto \frac{1}{\|c_i\|_2}c_i$$

Montrons qu'il existe une matrice $D \in M_{n,n}(\mathbb{R})$ diagonale telle que :

$$A^{(r)} := AD$$

avec $A^{(r)}$ la matrice réduite i.e. une matrice dont la norme des vecteurs colonnes est 1.

$$c_i^{(r)} = \frac{c_i}{\|c_i\|_2},$$

avec

$$\|C_i\|_2 = \sqrt{\sum_{i=1}^{m}(a_{ij})^2}$$

$$\text{Soit } D_{ii} = \frac{1}{\|C_i\|_2} \quad \text{pour } \forall i \in \{1, 2, \ldots, n\}.$$

Nous avons :

$$D = \begin{pmatrix} \frac{1}{\|C_1\|_2} & 0 & \cdots & 0 \\ 0 & \frac{1}{\|C_2\|_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\|C_n\|_2} \end{pmatrix}$$

et

$$A = (C_1, C_2, \ldots, C_n)$$

Donc

$$A^{(r)} = AD = (C_1, C_2, \ldots, C_n) \begin{pmatrix} \frac{1}{\|C_1\|_2} & 0 & \cdots & 0 \\ 0 & \frac{1}{\|C_2\|_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\|C_n\|_2} \end{pmatrix}$$

$$A^{(r)} = \begin{pmatrix} \frac{C_1}{\|C_1\|_2} & 0 & \cdots & 0 \\ 0 & \frac{C_2}{\|C_2\|_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{C_n}{\|C_n\|_2} \end{pmatrix}$$

iii.Données centrées-réduites:
A . *Cf : Code Python*

B. Soit $A \in M_{m,n}(\mathbb{R})$ une matrice centrée réduite. Montrons que :

$$\|A\|_F = \sqrt{n}.$$

Nous avons d'après les questions précédentes :

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |A_{ij}|^2} = \sqrt{\sum_{j=1}^{n} \|C_j\|_2^2}$$

ou pour une matrice centrée réduite nous avons :

$$\|C_j\|_2 = 1.$$

Donc :

$$\|A\|_F = \sqrt{\sum_{j=1}^{n} \|C_j\|_2^2} = \sqrt{\sum_{j=1}^{n} 1} = \sqrt{n}.$$

Déduisons que :

$$\sum_{i=1}^{\mathrm{rg}(A)} \lambda_i = n$$

où les λ_i sont les valeurs propres de $A^T A$. On a d'après le théorème spectral :

$$\sum_{i=1}^{\mathrm{rg}(A)} \lambda_i = \|A\|_F^2.$$

Avec A une matrice centrée réduite on en déduit donc :

$$\boxed{\sum_{i=1}^{\mathrm{rg}(A)} \lambda_i = \sqrt{n}^2 = n}$$

C . Notons R_{cr} la matrice « centrée-réduite » associée à la réalisation R et qui forme une réalisation de E_{cr}.
Nous avons la diagonalisation :

$$\Sigma = \frac{1}{m-1} R_{cr}^T R_{cr} = \frac{1}{m-1} V D V^T,$$

où D est une matrice diagonale dont les coefficients sont rangés par ordre décroissant :

$$D := \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

avec : $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$
Nous savons d'après la question 5.a que la matrice de covariance empirique d'un échantillon de X est :

$$\Sigma = \frac{1}{m-1} R_{cr}^T R_{cr}$$

Avec R_{cr} la matrice centrée-réduite associée à la réalisation R et qui forme une réalisation de E_{cr}. Par conséquent Σ est symétrique et semi-définie positive. Le théorème spectral nous donne :

$$R_{cr}^T R_{cr} = VDV^T \iff \Sigma = \frac{1}{m-1} VDV^T.$$

Où $\lambda_1, \ldots, \lambda_m$ sont les valeurs propres de V.On note $v_1, \ldots, v_m$ sont les vecteurs propres de V et k-directions principales. Montrons que $\mathrm{Var}(Y_i) = \lambda_k$. On a :

$$\mathrm{Var}(Y) = V^T \Sigma V$$

Utilisons la propriété $V^T V = I$:

$$= V^T \left(VDV^T\right) V$$

$$= (V^T V) D (V^T V)$$

$$= D = \mathrm{diag}(\lambda_1, \ldots, \lambda_m).$$

Nous en déduisons donc que,

$$\boxed{\mathrm{Var}(Y_i) = \lambda_k.}$$

1.5.3 Projection dans les directions principales

Supposons maintenant que l'on ait calculé toutes les directions principales i.e. que l'on ait la matrice V précédente. Justifions que l'expression de R_{cr} dans la base des directions principales $(v_i)_{i\in[1,n]}$ est :

$$P := R_{cr} V$$

Pour obtenir la matrice P, on projette la matrice R_{cr} dans la base des vecteurs propres associée à V, en la multipliant par cette dernière. En effet, V est une matrice dont les colonnes constituent une base orthonormée de l'espace vectoriel engendré par les vecteurs propres $(v_1, \ldots, v_k)$.

Ainsi supposons un entier $k \in [\![1, n]\!]$ fixé. Les données centrées-réduites R_{cr} peuvent donc être projetées dans l'espace vectoriel engendré par les vecteurs de directions principales : $Vec(v_1, \ldots, v_k)$ en considérant les k-premières colonnes de la matrice P précédente. On notera cette matrice $proj_k(R_{cr})$.

Il nous reste à donner un critère sur le nombre de composantes principales pour « bien analyser » les données. On suppose (cela est toujours le cas dans la pratique) que $m \gg n$. D'un point de vue théorique nous allons justifions que :

$$\frac{1}{n} \sum_{i=1}^{n} \mathrm{Var}(Y_i) = 1$$

Nous savons, d'après les questions précédentes :

$$\mathrm{Var}(Y_k) = \lambda_k$$

Ainsi que,

$$\sum_{i=1}^{\mathrm{rg}(A)} \lambda_i = n$$

Nous en déduisons avec rg(A) = n :

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^{n} \mathrm{Var}(Y_i) &= \frac{1}{n} \sum_{i=1}^{n} \lambda_i \\ &\iff \frac{n}{n} = 1 \\ &\iff \boxed{\frac{1}{n} \sum_{i=1}^{n} \mathrm{Var}(Y_i) = 1} \end{aligned}$$

Un critère pour choisir le nombre de composantes principales afin de projeter les données sur un espace « suffisamment représentatif » est donné par la règle dite **règle de Kaiser** : on choisit q le nombre de composantes principales comme l'indice q tel que :

$$\lambda_q \geq 1 \quad \text{et} \quad \lambda_{q+1} < 1$$

Dans la pratique on prendra donc q tel que :

$$\lambda_q \geq 1 - \epsilon \quad \text{et} \quad \lambda_{q+1} < 1 - \epsilon$$

avec $\epsilon > 0$ « petit ».

2 Partie 2 : Applications

Dans cette partie nous allons implémenter sous Python les résultats théoriques précédents. Nous appliquerons les fonctions demandées dans la suite sur les bases de données :

- **iris.csv** : base de données sur les iris. La dernière colonne ce tableau représente les labels des espèces et ne devra pas être considérée lors de l'ACP (en effet, la question que l'on peut se poser est de savoir si l'ACP « retrouve » les différentes espèces).
- **Pizzamod.csv** : cette base de donnée est une version modifiée (suppression de colonnes inutiles) de la base de données Pizza.csv. On peut trouver la description de cette base sur Kaggle : `https://www.kaggle.com/shishir349/can-pizzabe-healthy`.
- **Howellmod.csv** : version modifiée de Howell.csv (suppression de colonnes). Cette « grosse » base de données contient des mesures crâniennes (en mm) d'individus de populations diverses. Vous pouvez trouver un descriptif sur le site : `http://volweb.utk.edu/~auerbach/HOWL.htm`.
- Nous choisirons deux bases de données

En utilisant l'analyse en composantes principales nous essaierons de décrire si des structures/liens apparaissent comme par exemple l'existence de catégories marquées et lorsque les données sont déjà « labellisées » peut-on retrouver ces « labels » à partir de l'analyse en composantes principales, etc... Lorsque les tableaux de données sont grands nous choisirons de nous restreindre à certains individus (lignes) et certaines colonnes et de ne pas considérer le tableau de données dans son entièreté. Lorsque nous analyserons les résultats, nous expliciterons ce que signifient les directions principales dans le contexte des données.

2.1 Les fonctions pythons :

Pour obtenir certains graphiques, nous devons construire plusieurs fonctions en python :

2.1.1 La fonction centre_red(DATA) :

```
def centre_red(DATA):
    """
    Parametre
    ---------
    DATA : tableau de donnees

    Renvoie
    ---------
    La matrice centree reduite associee
    """
    return reduction(centrage(DATA))
```

Avec les fonctions centrage(R) et reduction(R) suivantes :

```
def centrage(DATA):
    """
    Parametre
    ---------
    DATA : tableau de donnees

    Renvoie
    ---------
    La matrice centree associee
    """
    if DATA.size == len(DATA[:]):
        m = DATA.size
        n = 1
    else:
        m, n = DATA.shape
    u = np.ones(m)
    return DATA - (1/m) * (u.T @ DATA)
```

```python
def reduction(DATA):
    """
    Parametre
    ---------
    DATA : tableau de donnees

    Renvoie
    ---------
    La matrice reduite associee
    """
    # Calcul des normes des colonnes
    norms = np.linalg.norm(DATA, axis=0)  # Norme L2 des colonnes

    # Construction de la matrice diagonale D
    D = np.diag(1 / norms)  # Inverser les normes pour remplir la diagonale

    # Calcul de la matrice reduite
    A_reduced = DATA @ D

    return A_reduced
```

2.1.2 La fonction Approx(R,k)

```
def approx(DATA, k):
    """
    Parameters
    ----------
    R : tableau de donnees numeriques.
    k : entier non nul.

    Returns
    -------
    une approximation de R.
    """

    # Dimensions de la matrice
    if DATA.size == len(DATA[:]):
        m = DATA.size
        n = 1
    else:
        m, n = DATA.shape

    DATA = centre_red(DATA)

    Sigma = (1 / (m - 1)) * (DATA).T @ (DATA)

    eigenvalues, eigenvectors = np.linalg.eigh(Sigma)

    sorted_indices = np.argsort(eigenvalues)[::-1]
    eigenvalues = eigenvalues[sorted_indices]
    eigenvectors = eigenvectors[:, sorted_indices]

    Vk = eigenvectors[:, :k]  # Matrice des k premiers vecteurs propres (taille n x k)

    projk_Rcr = (DATA) @ Vk
    projk_Rcr = -projk_Rcr

    return projk_Rcr, Vk, eigenvalues
```

2.1.3 La fonction ACP2D(R,labelsligne,labelscolonne)

```
def ACP2D(R, labelsLigne, labelsColonne):
    """
    Parametres
    ----------
    R : array(n,m)
        tableau des donnees numeriques
    labelsLigne : array contenant le nom des lignes
    labelsColonne : array contenant le nom des colonnes

    Renvoie
    -------
    Un diagramme en camembert
    Un graphique de la projection de notre tableau selon les directions principales
    """

    m, n = R.shape

    if type(labelsLigne[1]) == int:
        L = [0]
        for i in range(1, len(labelsLigne)):
            if labelsLigne[i] == labelsLigne[i-1]:
                L.append(L[i-1])
            else:
                L.append(L[i-1] + 1)
        labelsLigne = L

    P, V, lambda_i = approx(R, 2)

    var_i = np.zeros(n)
    for i in range(n):
        var_i[i] = lambda_i[i] / n

    plt.figure()
    plt.pie(var_i,
            labels=labelsColonne,
            autopct="%.2f%%",
            startangle=90)
    plt.title(label="")
    plt.show()

    plt.figure()

    couleurs = ['blue', 'orange', 'green', 'purple', 'cyan', 'magenta', 'yellow',
                'brown', 'beige', 'red', 'teal', 'lime']

    if len(couleurs) < len(labelsLigne):
        for i in range(0, len(labelsLigne) - len(couleurs)):
            couleurs.append(randcolour())

    for i in range(len(labelsLigne)):
        plt.scatter(P[i, 0], P[i, 1], color=couleurs[(labelsLigne[i])])

    plt.xlabel('v1')
    plt.ylabel('v2')
    plt.grid()
    plt.show()

    return
```

2.1.4 La fonction Cercle_corr2D :

```
p)
    hl = 0.05

    # Trace des fleches et annotations des points
    for i in range(R.shape[1]):
        # On recupere le vecteur de correlation pour chaque i
        x_corr = R[0, i]
        y_corr = R[1, i]

        # On normalise le vecteur de correlation pour qu'il atterrisse sur le cercle
        norm = np.sqrt(x_corr**2 + y_corr**2)
        x_corr_norm = x_corr / norm
        y_corr_norm = y_corr / norm

        # On trace la fleche associee partant du centre vers la tete du vecteur normalise
        ax.arrow(0, 0, (1-hl)*x_corr_norm, (1-hl)*y_corr_norm, head_width=0.05,
                 head_length=hl, fc='r', ec='r')

        # On ajoute le label de chaque point
        ax.annotate(labelscolonne[i], (x_corr_norm, y_corr_norm), textcoords="offset points",
                    xytext=(10, 5), ha='center', fontsize=10)

    # Parametrage visuel accessoire
    ax.axhline(0, color='black', linewidth=0.5)
    ax.axvline(0, color='black', linewidth=0.5)
    ax.set_title('Cercle des Correlations 2D')
    plt.grid(True)
    plt.show()
```

2.2 Résultats et interprétations :

2.2.1 Base de données iris.csv

Ces graphiques nous montrent que la longueur des sépales, la longueur et la largeur des pétales sont corrélées positivement mais que la largeur des sépales est quasiment indépendante.

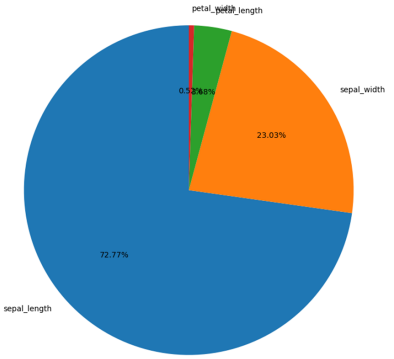

Figure 2: Camembert des variances (iris)

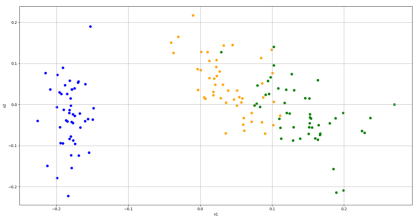

Figure 3: Représentation des données projetées (iris)

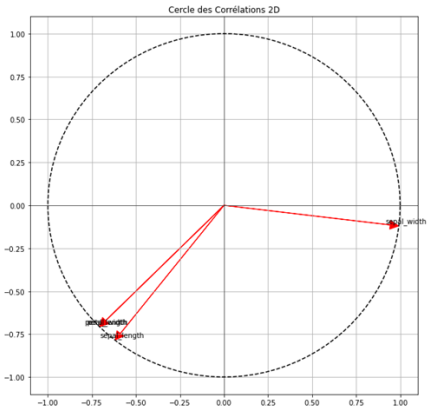

Figure 4: code des corrélations 2D (iris)

2.2.2 Base de donnée HowellMod.csv

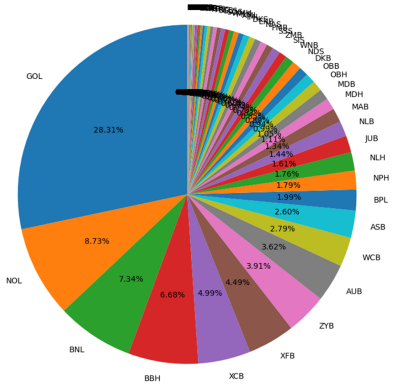

Figure 5: Camembert des variances (HowellMod)

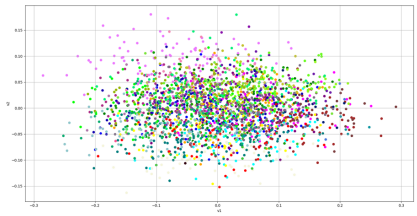

Figure 6: Représentation des données projetées (HowellMod)

L'éparpillement des points, et surtout la façon dont ils sont tous assez centrés, implique qu'il n'y a que les dimensions crâniennes de chaque population sont toutes assez similaires. Cela est tout à fait cohérent puisque les populations en question sont des populations composées d'humain, donc tous de la même espèce. Les différences présentes ne sont dues qu'à la dérive génétique qu'ont pu subir chaque populations. Ainsi, si l'on avait ajouté les mesures crâniennes de différentes espèces dans un même tableau, on aurait pu apercevoir différentes « taches », similaires à celle ci-dessus, et toutes plus ou moins séparées. On peut toutefois remarquer que la variance sur l'axe verticale de chaque population est plus petite que la variance sur l'axe horizontale.

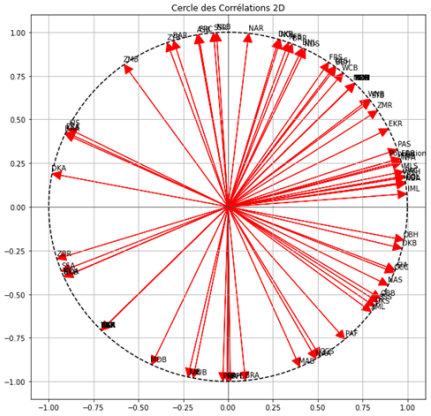

Figure 7: code des corrélations 2D (HowellMod)

Les flèches sont bien réparties, ce qui signifie que les données sur les axes décrivent bien les données de manière générale.

2.2.3 Base de données Pizzamod.csv

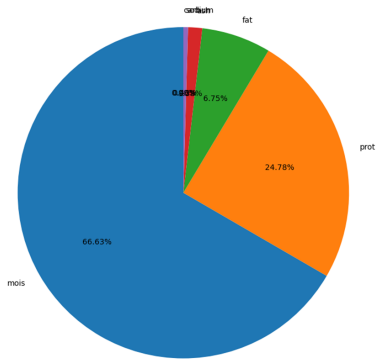

Figure 8: Camembert des variances (Pizzamod)

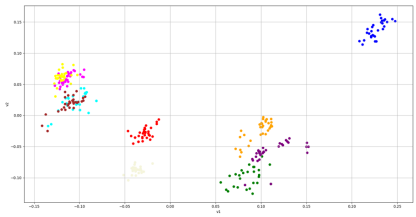

Figure 9: Représentation des données projetées (Pizzamod)

Le fait que des taches apparaissent sur la projection nous indique que toutes ces pizzas ont des valeurs nutritionnelles assez similaire, malgré le fait qu'elles aient différents ingrédients.

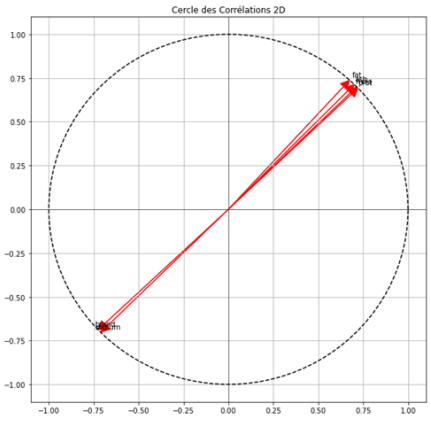

Figure 10: code des corrélations 2D (Pizzamod)

Ce graphique nous informe que toutes les données sont corrélées directement. Par exemple, si le gras (fat) est plus élevé, alors la cendre (ash), les protéines (prot), les glucides (carbs), les calories (cal) et les mois aussi, tandis que la marque sera plus basse (plus proche de A dans l'alphabet) et le sodium moins élevé.

2.2.4 Base de données Villepop.csv

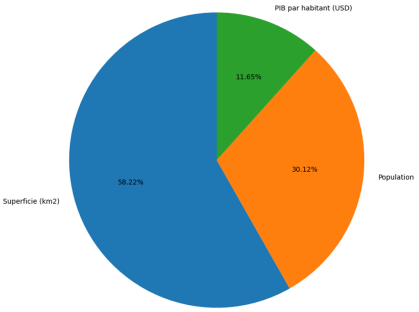

Figure 11: Camembert des variances (Villepop)

Sur le camembert on observe que la superficie est effectivement ce qui contribue le plus à la variance. La taille d'une capitale étant très changeant selon le niveau de développement du pays ou de la densité de population dans celle-ci. On pourrait aussi le conclure pour la population mais la quantité de personnes par nation est plus limité ainsi la variance est moins élevée. De plus la population suit la superficie car plus le pays est grand plus il y a d'habitant (hors cas particuliers comme Singapour ou Hong Kong qui sont des territoires ultra urbanisés). La variance du PIB est la plus faible aussi car si c'était le cas certains pays se serait déjà effondrés car ils ne pourraient pas échanger avec le monde.

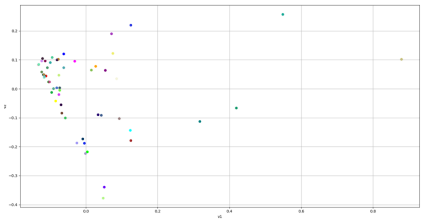

Figure 12: Représentation des données projetées (Villepop)

Cette différence entre d'ordre de grandeur entre les différents paramètres se voit particulièrement bien sur cette projection ; la plupart des points se trouvent plutôt proches du centre (0,0), mais il y a quelques points dont la variance est bien plus grande et qui sont bien plus éloignés du centre.

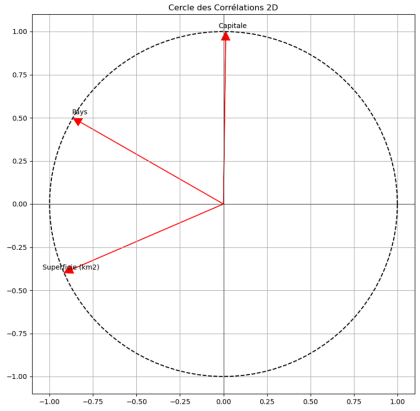

Figure 13: code des corrélations 2D (Villepop)

Ce graphique nous informe que la superficie des capitales est positivement corrélée au PIB par habitant par capitale.

2.2.5 Base de donnée InseePop.csv

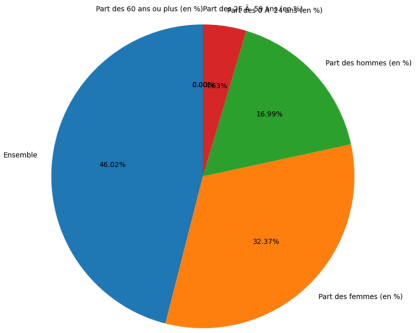

Figure 14: Camembert des variances (Inseepop)

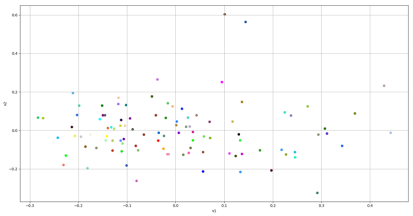

Figure 15: Représentation des données projetées (Inseepop)

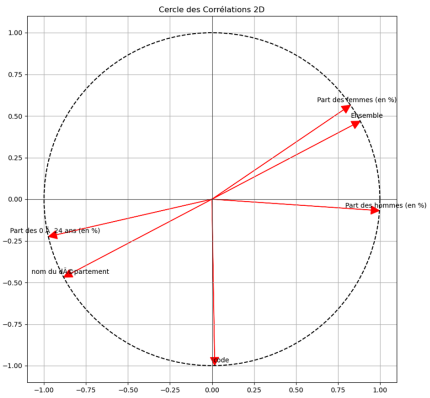

Figure 16: code des corrélations 2D (Inseepop)

Conclusion :

L'analyse en composantes principales (ACP) est une méthode puissante permettant facilement l'analyse et l'interprétation d'ensembles de données à plusieurs dimensions. En effet, elle est capable de réaliser des réductions dimensionnelles sans grande perte d'information, et est un outil important pour repérer et comprendre des mécanismes et des relations invisibles à l'œil nu entre des données. La projection dans des espaces en deux dimensions par exemple apporte à notre portée une visualisation des données intuitives sur des diagrammes. Seulement cette technique connaît ses limites, puisqu'il ne s'agit que d'une méthode linéaire. Elle n'est donc pas tout à fait capable d'extraire certaines relations plus complexes entre des données.