BE : Analyse en composantes principales

Par groupe de 4 étudiants, vous devez rédiger un rapport dans lequel vous répondrez aux questions du sujet et plus globalement au problème demandé. La qualité de rédaction ainsi que tous les compléments que vous apporterez (réflexions, analyses des réponses, tests sur d'autres données, etc...) seront fortement pris en compte. Toutes les fonctions programmées le seront en Python et seront abondamment commentées. Le rendu du projet se fera sous la forme d'un fichier .zip ou .rar contenant le rapport ainsi que l'ensemble des programmes.

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. Ce sujet comporte trois parties. **La première partie** concerne la mise en place théorique de l'ACP. Cette partie réutilise des éléments vus en cours et TDs. Lorsque vous réutilisez de tels éléments, il n'est pas nécessaire de les redémontrer. **La deuxième partie** est la programmation effective sous Python des résultats démontrés en partie 1. Le rendu attendu est l'utilisation des fonctions de bases des librairies Numpy et Matplotlib. Néanmoins l'utilisation de fonctions plus avancées qui améliorent les rendus graphiques par exemple ou/et la programmation en terme de classe (cf. cours de Python en Aéro2) seront bien évidement acceptées et seront valorisées. Cette partie comporte trois fonctions principales **ACP2D**, **ACP3D** et **ACP** qui sont des quasi-copier-coller l'une de l'autre, seule les représentations graphiques sont différentes (2D, 3D, dimension k quelconque respectivement). Enfin la **troisième partie** est une partie « bonus », i.e. hors barème, qui consiste en la programmation d'un algorithme décrit dans le sujet : les k-moyennes. Les compétences nécessaires pour programmer un tel algorithme sont d'un niveau Aéro1 (maniement des listes principalement). Cette partie « bonus » permet, lorsqu'elle est couplée à ACP, de produire un algorithme d'apprentissage non-supervisée : on crée des catégories dans les données sans connaissance a priori de ce qu'incarnent les données (de la science sans scientifique ?).

Partie 1 : Un peu de théorie

Soit $X := (X_1, X_2, \cdots, X_n)$ un vecteur aléatoire défini sur un espace probabilisé $(\Omega, \mathbb{A}, P)$ (les vecteurs aléatoires seront considérés comme des vecteurs lignes). On supposera que les variables aléatoires X_i sont **centrées** i.e. $E(X_i) = 0$, et **réduites** : $Var(X_i) = 1$. Attention cela ne signifie pas que les X_i suivent des 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 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 façon 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-linéaire comme l'analyse en composantes curvilignes ou l'ACP à noyau peuvent être très coûteuses en temps de calcul voire pas toujours réalisable.

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 de $\mathbb{R}^n$ de norme égal à 1 (i.e. appartenant à la sphère unité) et tel que :

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

On appelle v_1 la **première direction principale**. Par récurrence on définit la $(k+1)-$**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 :

$$v_{k+1} = \arg \max_{\substack{v \perp Vec(v_1, v_{2,\ldots}, v_k) \\ \|v\|_2 = 1,\ v \in \mathbb{R}^n}} Var\,(Xv)$$

avec $v \perp Vec\,(v_1, v_{2,\ldots}, v_k)$ 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. Montrer que :
$$Var\,(Xv) = v^T Cov(X) v$$
où $Cov(X)$ est la matrice de covariance du vecteur aléatoire X.

2. Justifier (en utilisant vos cours d'algèbres...) que la matrice $Cov(X)$ se diagonalise dans une base orthonormée i.e. qu'il existe V une matrice orthogonale et D une matrice diagonale telles que :
$$Cov(X) = VDV^T$$
Quitte à permuter les vecteurs de V, on supposera que les termes (diagonaux) de D sont rangés par ordre décroissant :
$$D := \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$
avec : $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.

3. Soit $k \in [\![1, n]\!]$. En déduire que la $k-$composante principale de X est donnée par :
$$\boxed{Y_k = Xv_k}$$
où v_k est le $k-$ième vecteur colonne de la matrice orthogonale V. En déduire que :
$$\boxed{Var(Y_k) = \lambda_k}$$
Ainsi on remarque au passage que les valeurs propres de la matrice de covariance $Cov(X)$ mesurent la dispersion du vecteur aléatoire X. On définit ainsi **le pourcentage de l'explication de la variance de la** $k-$**composante principale** par :
$$\boxed{\rho_k := \frac{\lambda_k}{Trace(Cov(X))}}$$
On parle également de ρ_k comme l**a part d'inertie expliquée** (l'inertie signifiant ici la somme des valeurs propres i.e. $Trace(Cov(X))$).

4. 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 donnée 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 $(Ind_1, Ind_2, \cdots, Ind_m)$ un $m-$échantillon du vecteur aléatoire X.[1]. 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}

1. On note Ind_j car ces variables aléatoires représentent dans la pratique les données acquisses sur des « individus » (les données/paramètres $X_1, \cdots$ suivant des lois de probabilités).

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 :

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

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

$$\overline{X_{mi}} := \frac{1}{m}\sum_{k=1}^{m} X_{ki} \qquad \text{et} \qquad \overline{S_{mi}} := \frac{1}{m-1}\sum_{k=1}^{m}\left(X_{ki} - \overline{X_{mi}}\right)^2$$

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

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

Attention on peut néanmoins montrer que ce dernier estimateur n'est pas sans biais (il est cependant asymptotiquement sans biais).

(a) On note E_{cr} la matrice « centrée-réduite » associée à E dont les coefficients sont données par les variables aléatoires :

$$(E_{cr})_{jk} := \frac{X_{jk} - \overline{X_{mk}}}{\overline{\sigma_{mk}}}$$

Démontrer que :

$$C := E_{cr}^T E_{cr}$$

est une approximation d'un estimateur de la matrice de covariance de X. On pourra supposer que pour tout i, j, k, p :

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

(b) Supposons donnée une réalisation R d'un $m-$échantillon $(Ind_1, Ind_2, \cdots, Ind_m)$ i.e. R est la donnée d'une matrice de taille (m, n) :

$$R := \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ldots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix} \in \mathcal{M}_{mn}(\mathbb{R})$$

et notons R_{cr} la matrice « centrée-réduite » associée à la réalisation R et qui forme une réalisation de E_{cr}. En utilisant les questions précédentes, montrer que si l'on connaît une décomposition en valeurs singulières de $R_{cr} = U\Sigma V^T$ on trouve les $k-$**directions principales** v_k où les v_k sont les $k-$ème vecteurs colonnes de V. De plus montrer que :

$$Var(Y_k) = \sigma_k^2$$

où σ_k est la $k-$ème valeur singulière de R_{cr}.

(c) Supposons que l'on ait calculé les k premières directions principales. Montrer que les données centrées-réduites R_{cr} peuvent être projetées (=approximées) dans l'espace vectoriel engendré par les vecteurs de directions principales : $Vec(v_1, \cdots, v_k)$ et que la matrice des composantes projetées sur cet espace vectoriel s'écrit par blocs colonnes sous la forme :

$$proj_k(R_{cr}) := \begin{pmatrix} \sigma_1^2 u_1 & \sigma_2^2 u_2 & \cdots & \sigma_k^2 u_k \end{pmatrix}$$

avec les u_j les $j-$ème vecteurs colonnes de U dans la décomposition en valeurs singulières de $R_{cr} = U\Sigma V^T$.À quel résultat d'algèbre vu en Ma313 ce résultat vous fait-il penser ?

5. 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 $m >> n$. D'un point de vu théorique justifier que :

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

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 :

$$\sigma_q^2 \geq 1 \text{ et } \sigma_{q+1}^2 < 1$$

où les σ_i sont les valeur singulière de R_{cr} . Dans la pratique on prendra donc q tel que :

$$\sigma_q^2 \geq 1-\epsilon \text{ et } \sigma_{q+1}^2 < 1-\epsilon$$

avec $\epsilon > 0$ « petit ».

6. Enfin et pour finir l'analyse, il est également intéressant de comparer les nouvelles variables trouvées Y_k par analyse en composantes principales et les anciennes X_i, $i \in [\![1,n]\!]$ en regardant leurs corrélations. On suppose construit les k premières composantes principales d'une matrice de donnée. Justifier que le vecteur :

$$Cor_i := \begin{pmatrix} Cor\,(Y_1,X_i) & Cor\,(Y_2,X_i) & \cdots & Cor\,(Y_k,X_i) \end{pmatrix} \in \mathbb{R}^k$$

est un point qui appartient à la boule unité $\mathbb{B}^k$ de $\mathbb{R}^k$. On peut alors penser à ce vecteur comme les coordonnées de la variable aléatoire X_i dans le repère des composantes principales $\{Y_1, Y_2, \cdots, Y_k\}$. On appelle ce type de représentation : le **cercle des corrélations**. Comment interpréter ce point (en fonction de la norme de Cor_i, de ses angles par rapports aux axes Y_j, etc...) ? Pour répondre à cette question vous pouvez vous aider des cas appliqués de la partie 2.

Partie 2 : Applications

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

- iris.csv : base de données sur les iris.
- 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-pizza-be-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.
- Au moins 2 autres bases de données que vous choisirez.

En utilisant l'analyse en composantes principales vous essaierez de décrire si des structures/liens apparaissent. Par exemple l'existence de catégories marquées, liens entre les variables mesurées et/ou les composantes principales, lorsque les données sont déjà « labellisées » peut-on retrouver ces « labels » à partir de l'analyse en composantes principales, etc...

Votre travail devra se présenter obligatoire sous la forme suivante :

- un fichier *ACP.py* qui contiendra toutes les fonctions demandées,
- un fichier *main.py* qui contiendra l'importation des données ainsi que les analyses en composantes principales réalisées sur celles-ci. Dans ce fichier vous importerez vos fonctions contenues dans *ACP.py* grâce à la commande suivante (placée en début de programme) :

```
import ACP as acp
```

et vous chargerez les données en utilisant la fonction np.genfromtxt. Par exemple sur *iris.csv* :

```
Ebrut=np.genfromtxt("iris.csv", dtype=str,  delimiter=',') #données brutes
labelscolonne=Ebrut[0,:-1]
labelsligne=Ebrut[1:,-1]
E=Ebrut[1:,:-1].astype('float')
```

Dans la suite nous allons illustrer le travail à faire par des exemples de sortie graphiques que l'on peut obtenir lorsque l'on réalise les fonctions demandées sur la base de données déjà rencontrée en Ma313 : **iris.csv.**

Dans votre fichier *ACP.py* vous devez :

- créer une fonction **centre_red(R)** : qui prend en entrée un tableau de données numérique R et qui renvoie la matrice R_{rc} centrée réduite associée.
- créer une fonction **approx(R,k)** : qui prend en entrée le tableau de données numériques R et k un entier non nul et qui renvoie la matrice (cf. Partie 1) :

$$proj_k(R_{cr}) := \begin{pmatrix} \sigma_1^2 u_1 & \sigma_2^2 u_2 & \cdots & \sigma_k^2 u_k \end{pmatrix}$$

- créer une fonction **correlationdirprinc(R,k)** : qui prend en entrée le tableau de données numériques R et k un entier non nul et qui renvoie la matrice (cf. Partie 1) de taille (k, n) dont les colonnes i sont données par :

$$Cor_i := \begin{pmatrix} Cor\,(Y_1, X_i) & Cor\,(Y_2, X_i) & \cdots & Cor\,(Y_k, X_i) \end{pmatrix}^T \in \mathbb{R}^k$$

- créer une fonction **ACP2D(R, labelsligne, labelscolonne)** qui prend en entrée :
  - R : le tableau de données numériques,
  - labelsligne : les labels des noms des lignes (s'ils existent, si non on prendra un vecteur d'entier de 1 à m),
  - labelscolonne : les labels des noms des colonnes (s'ils existent si non on prendra un vecteur d'entier de 1 à n),

  et qui produit en sortie :
  - le graphe qui représente les valeurs des variances σ_k^2 et le graphe qui représente le pourcentage de l'explication de la variance de chaque $k-$composante principale (cf. Partie 1).

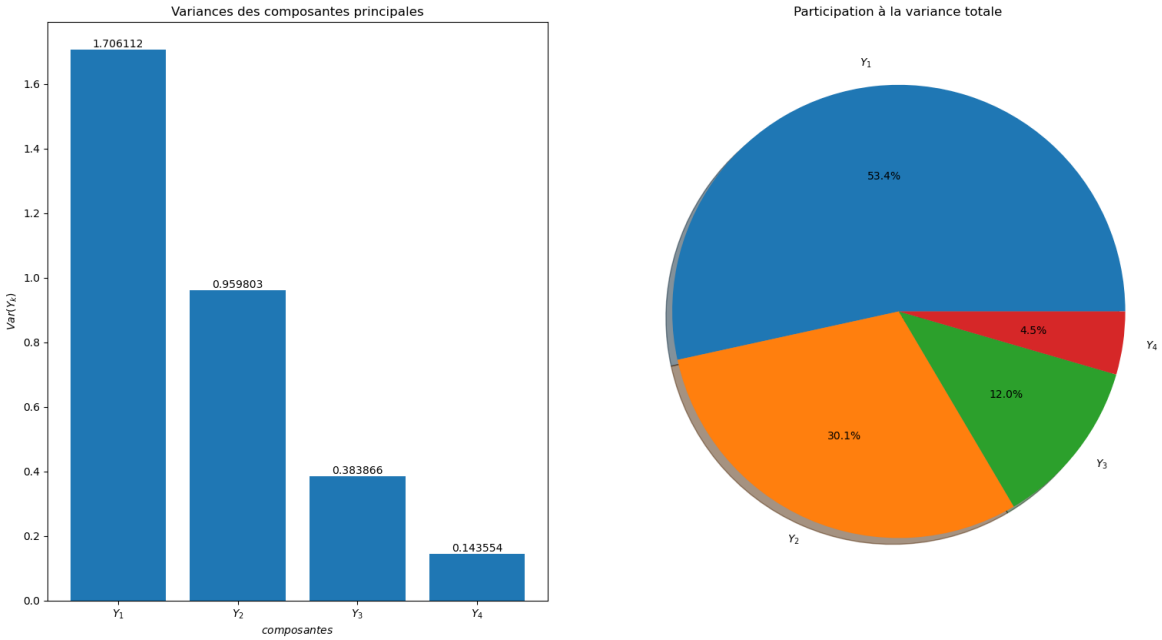

FIGURE 1 – Valeurs des variances et pourcentage explicatif de celles-ci pour les composantes principales

On pourra utiliser les commandes de matplotlib dont il est vivement conseillé d'aller voir les fichiers d'aide :
  - *fig1, (ax1,ax2) = plt.subplots(1,2)* : pour réaliser un tracer « double »,
  - *ax1.bar(à compléter)* : pour réaliser un diagramme en bâton,
  - *ax2.pie(à compléter)* : pour réaliser un diagramme en camembert.
- le graphe en 2D qui représente la matrice de sortie de approx(R,2) et le graphe de la matrice de sortie de la fonction **correlationdirprinc(R,2)**.

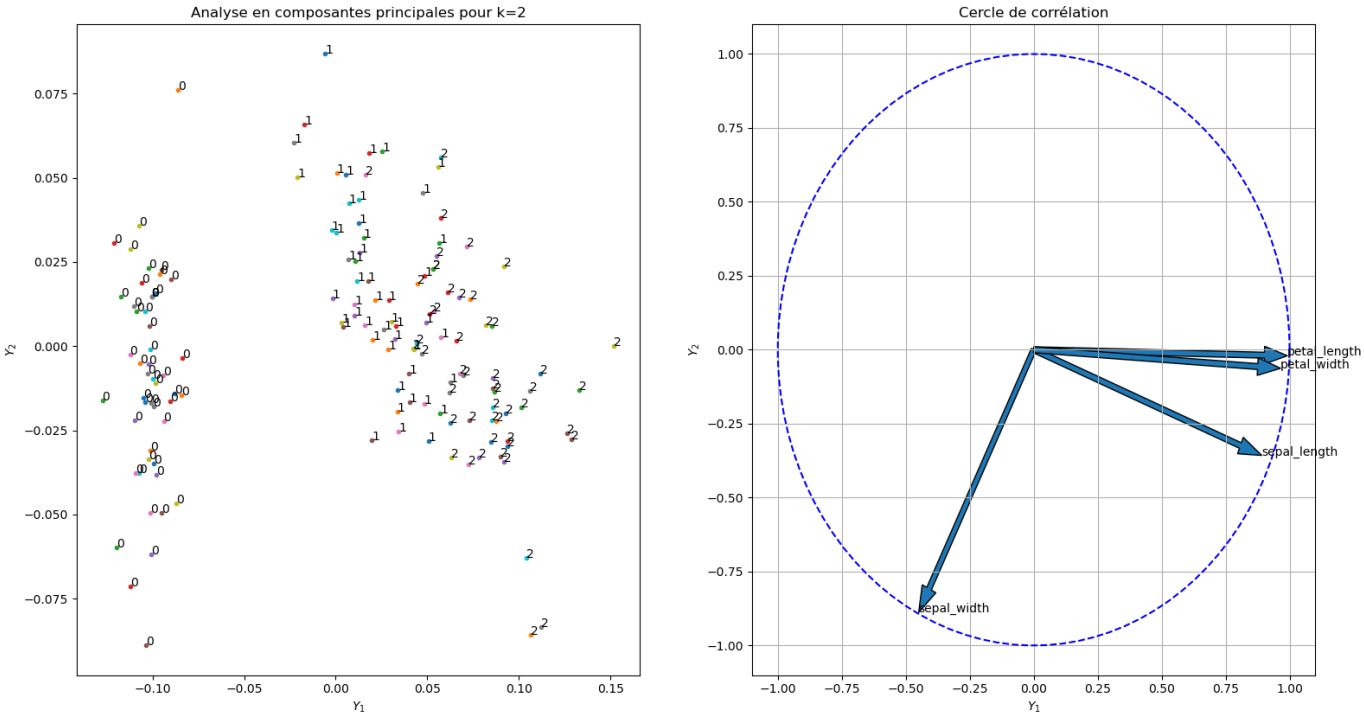

FIGURE 2 – Représentation des données dans le plan (Y_1, Y_2) et le cercle de corrélations dans ce même plan.

On pourra utiliser les commandes de matplotlib (dont il est vivement conseillé d'aller voir les fichiers d'aide encore une fois...) :

- *fig2, (ax3,ax4)= plt.subplots(1,2)* : pour réaliser un tracer « double »,
- *ax3.annotate(à compléter)* : pour annoter les points tracés (également pour ax4),
- *ax4.arrow(à compléter)* : pour tracer des flèches incarnant les corrélations entre les X_i et les Y_k.

- créer une fonction **ACP3D(R, labelsligne, labelscolonne)** qui réalise cette fois une analyse en composantes principales pour les 3 premières composantes principales et qui produira en sortie :
  - comme précédemment le graphe qui représentent les valeurs des variances σ_k^2 et le graphe qui représente le pourcentage de l'explication de la variance de chaque $k-$composante principale (cf. Partie 1).
  - le graphe en 3D ou les trois graphes (fois 2) des projections sur les plans $(Y_1, Y_2), (Y_1, Y_3), (Y_2, Y_3)$ qui représente(nt) la matrice de sortie de **approx(R,3)** et le graphe de la matrice de sortie de la fonction **correlation-dirprinc(R,3)**.

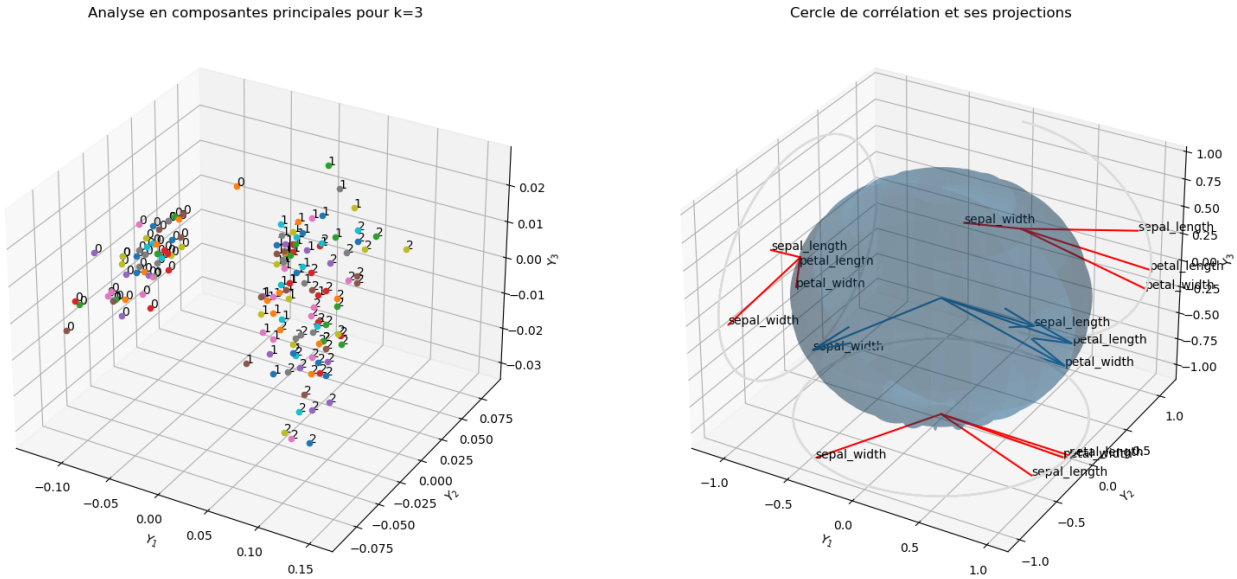

FIGURE 3 – Représentation des données dans l'espace (Y_1, Y_2, Y_3) et le cercle de corrélations dans ce même espace.

- créer une fonction **ACP(R, labelsligne, labelscolonne,k=0,epsilon=10**-1)** qui réalise une analyse en composantes principales pour les k premières composantes principales avec :
  - soit k est données par l'utilisateur,
  - soit si k n'est pas donné (k=0) : on prendra pour k, la valeur donnée par la règle dite **règle de Kaiser** (cf. Partie 1) pour une marge de epsilon (si non spécifiée par l'utilisateur on prendra $\epsilon = 10^{-1}$).

  Cette fonction produira en sortie :
  - comme précédemment le graphe qui représente les valeurs des variances σ_k^2 et le graphe qui représente le pourcentage de l'explication de la variance de chaque $k-$composante principale (cf. Partie 1),
  - la représentation graphique grâce à *plt.imshow* de la matrice de sortie de la fonction **correlationdirprinc(R,k)**. On pourra ainsi, en particuliers lorsque $k > 3$ (même si cela fait sens pour les dimensions plus petites), visualiser les corrélations entre les nouvelles variables (les directions principales) et les anciennes.

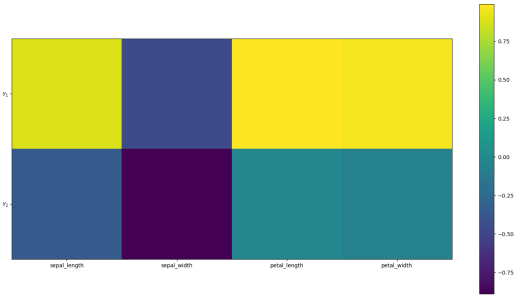

FIGURE 4 – Représentation de la matrice de corrélation entre les Y_i et les X_j sur Iris (k=2 d'après la règle de Kaiser avec $\epsilon = 10^{-1}$).

Partie 3 (Bonus) : Création de catégories par l'algorithme des k-moyennes

Si l'analyse en composantes principales permet de réduire la taille de représentation des données, il reste néanmoins à créer des catégories (=partitionnement) des données dans cette nouvelle représentation « débruitée ». En effet la réduction de dimension réalise une sorte de « réduction du bruit » sur les données au sens où les variations « marginale » sont ignorées/supprimées. Dans cette partie bonus nous allons voir un des algorithmes les plus connus « résolvant » le problème du partitionnement de données : l'**algorithme des k-moyennes** (**k-means** en anglais). Nous allons commencer par décrire formellement le problème du partitionnement de données.

Soient $(a_i)_{i\in[\![1,m]\!]}$, m vecteurs (vu comme vecteurs colonnes) de $\mathbb{R}^n$. Fixons k un entier et notons $D = \{a_1, \cdots, a_m\}$ l'ensemble de ces vecteurs vu comme un ensemble à m éléments. On rappelle que $\mathcal{P}(D)$ note l'ensemble des parties de D. Le problème du partitionnement de données consiste à trouver une partition $S = \{S_1, S_2, \cdots, S_k\}$ de l'ensemble D en k parties minimisant la quantité suivante :

$$\min_{S\in\mathcal{P}(D)} \sum_{j=1}^{k} \sum_{a_i\in S_j} \|a_i - \mu_j\|_2^2$$

où μ_j est le barycentre de l'ensemble S_j i.e. :

$$\mu_j = \frac{1}{Card(S_j)} \sum_{a_i\in S_j} a_i$$

avec $Card(S_j)$ le cardinal (=nombre d'éléments) de l'ensemble S_j. Nous allons voir un algorithme qui se veut une solution heuristique au problème de partitionnement des données : l'**algorithme des k-moyennes**.

1. **Algorithme des k-moyennes :** Soit $A \in \mathcal{M}_{mn}(\mathbb{R})$ une matrice que nous allons considérée par blocs de lignes :

$$A := \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix}$$

avec a_i le vecteur ligne : $a_i := (a_{i1}, a_{i2}, \cdots, a_{in})$.
Créer un programme/fonction $Kmoy(A, k, \epsilon)$ qui prend en entrée une matrice A, un entier $k \geq 2$ et un réel $\epsilon > 0$ et qui renvoie une liste de longueur k de matrices incarnant une partition de taille k de l'ensemble : $D = \{a_1, \cdots, a_m\}$ en suivant la procédure suivante :

(a) **Étape 1 :** choisir k vecteurs a_l distincts aux hasards parmi les lignes de A :

$$\{a_{l_1}, a_{l_2}, \cdots, a_{l_k}\}$$

et définir : $\mu_j^{(0)} := a_{l_j}$ pour tout $j \in [\![1, k]\!]$.

(b) **Étape 2 :** définir les partitions $S_j^{(1)}$ qui contiennent les a_i les plus proches de $\mu_j^{(0)}$ i.e. :

$$S_j^{(1)} := \left\{ a_i \mid \left\| a_i - \mu_j^{(0)} \right\|_2 \leq \left\| a_i - \mu_p^{(0)} \right\|_2, \forall p \in [\![1, k]\!] \setminus \{j\} \right\}$$

(c) **Étape 3 :** calculer les nouveaux barycentres des partitions $S_j^{(1)}$:

$$\mu_j^{(1)} := \frac{1}{Card(S_j^{(1)})} \sum_{a_i\in S_j^{(1)}} a_i$$

et définir la matrice (considérée par blocs lignes) :

$$\mu^{(1)} := \begin{pmatrix} \mu_1^{(1)} \\ \mu_2^{(1)} \\ \vdots \\ \mu_k^{(1)} \end{pmatrix}$$

(d) **Étape 4 :** Itérer les étapes 2 et 3 jusqu'à ce qu'il existe un entier f :

$$\left\|\mu^{(f-1)} - \mu^{(f)}\right\| \leq \epsilon$$

(e) **Étape 5 :** Renvoyer la liste de longueur k :

$$S = \{S_1, S_2, \cdots S_k\}$$

où les S_j sont les matrices dont les vecteurs lignes sont données par les vecteurs de la partition $S_j^{(f)}$.

2. Tester votre programme sur les matrices :

$$proj_p(R_{cr}) := \begin{pmatrix} \sigma_1^2 u_1 & \sigma_2^2 u_2 & \cdots & \sigma_p^2 u \end{pmatrix}$$

obtenue lors d'une ACP en p composantes principales (p peut être différent du k choisi comme nombre de catégories). On pourra également utiliser la règle de Kaiser pour choisir k le nombre de catégorie tout en réalisant une ACP pour $p = 2$ ou 3 afin de visualiser les données.

3. Calculer dans chaque cas testés :

$$Val(A, k) := \sum_{j=1}^{k} \sum_{a_i \in S_j^{(f)}} \left\|a_i - \mu_j^{(f)}\right\|_2^2$$

et analyser les résultats obtenues.

4. En dimension $k = 2$ ou 3 colorier sur la représentation de $proj_p(R_{cr}) := \begin{pmatrix} \sigma_1^2 u_1 & \sigma_2^2 u_2 & \cdots & \sigma_p^2 u \end{pmatrix}$ les catégories obtenues grâce à l'algorithme des k-moyennes i.e. on coloriera d'une même couleur les « individus » appartenant à une même catégorie.

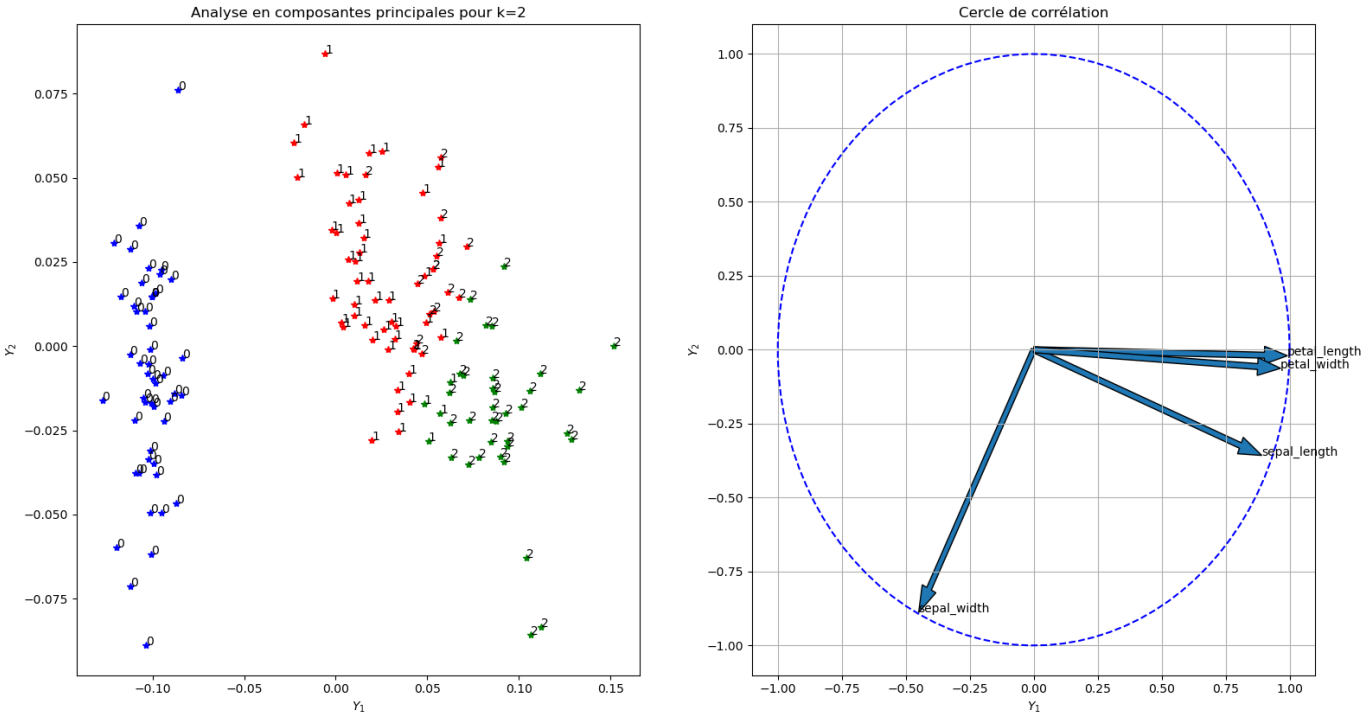

FIGURE 5 – Représentation des données dans le plan (Y_1, Y_2) en utilisant des k-moyennes (k=3) pour créer un partitionnement (catégories rouge, vert, bleu). On remarque que l'on trouve bien, à quelques erreurs près, les catégories données par les botanistes (=labels).