SY09 Printemps 2023
TD/TP 04 — Analyse en composantes principales

L'objectif de cet exercice est de se familiariser avec la bibliothèque `scikit-learn` à travers le calcul de l'ACP.

1 Travaux pratiques

1.1 Introduction : ACP des données de notes

La bibliothèque `scikit-learn` permet facilement de calculer une ACP. Pour cela, il faut importer la classe suivante :

```
from sklearn.decomposition import PCA
```

Il faut ensuite instancier cette classe. Le seul paramètre qui nous intéresse est `n_components` qui fixe le nombre de composantes principales à retenir. On peut également fixer ce nombre de composantes principales en spécifiant une proportion d'inertie expliquée minimale sous la forme d'un nombre flottant entre 0 et 1.

Par exemple, pour calculer les deux premières composantes principales d'un tableau individu–variable `X`, on construit l'objet

```
cls = PCA(n_components=2)
```

Pour calculer la nouvelle représentation et renvoyer les composantes principales, il faut utiliser la méthode `fit_transform` en fournissant le jeu de données `X`. La méthode `fit_transform` apprend les deux premiers axes factoriels et calcule la transformation correspondante du jeu de données `X`.

```
pcs = cls.fit_transform(X)
```

La variable `pcs` contient alors les `n_components` composantes principales. De plus, les attributs suivants sont disponibles depuis l'objet `cls` :

- `components_` : les axes factoriels u_i rangés par lignes,
- `explained_variance_` : la variance expliquée par chacun des axes factoriels,
- `explained_variance_ratio_` : le pourcentage de variance expliquée par chacun des axes factoriels.

Pour appliquer la représentation apprise avec X sur un autre tableau individu–variable Y, on utilise

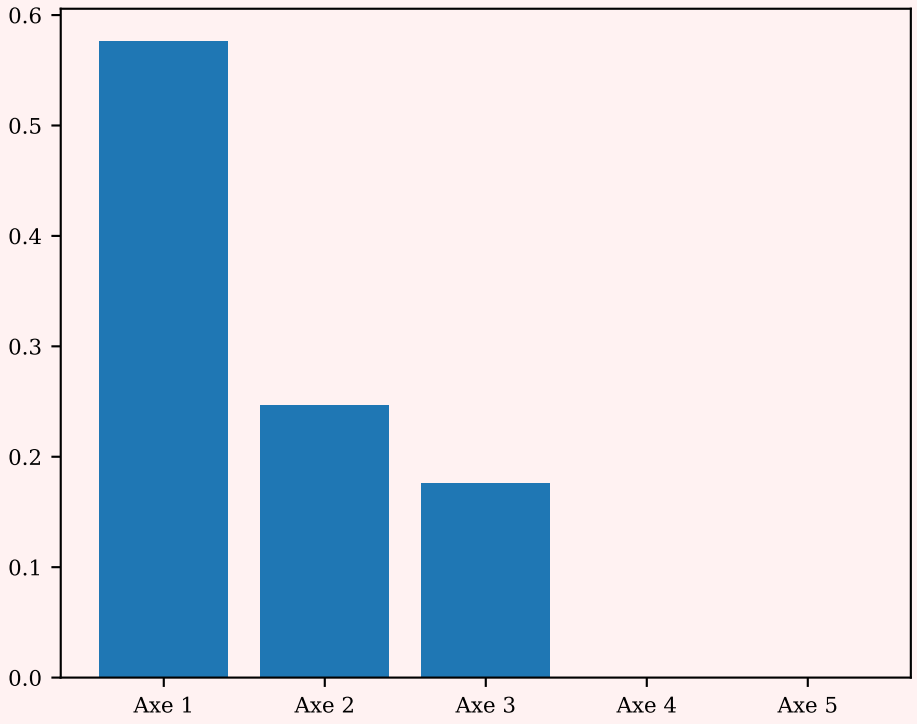
```
pcs_Y = cls.transform(Y)
```

1 Charger le jeu de données du TD précédent avec l'instruction

```
notes = pd.read_csv("data/notes.txt", sep="\s+")
```

Retrouver approximativement, avec les axes principaux, la base B_3 du TD précédent, ainsi que les variances expliquées correspondantes.

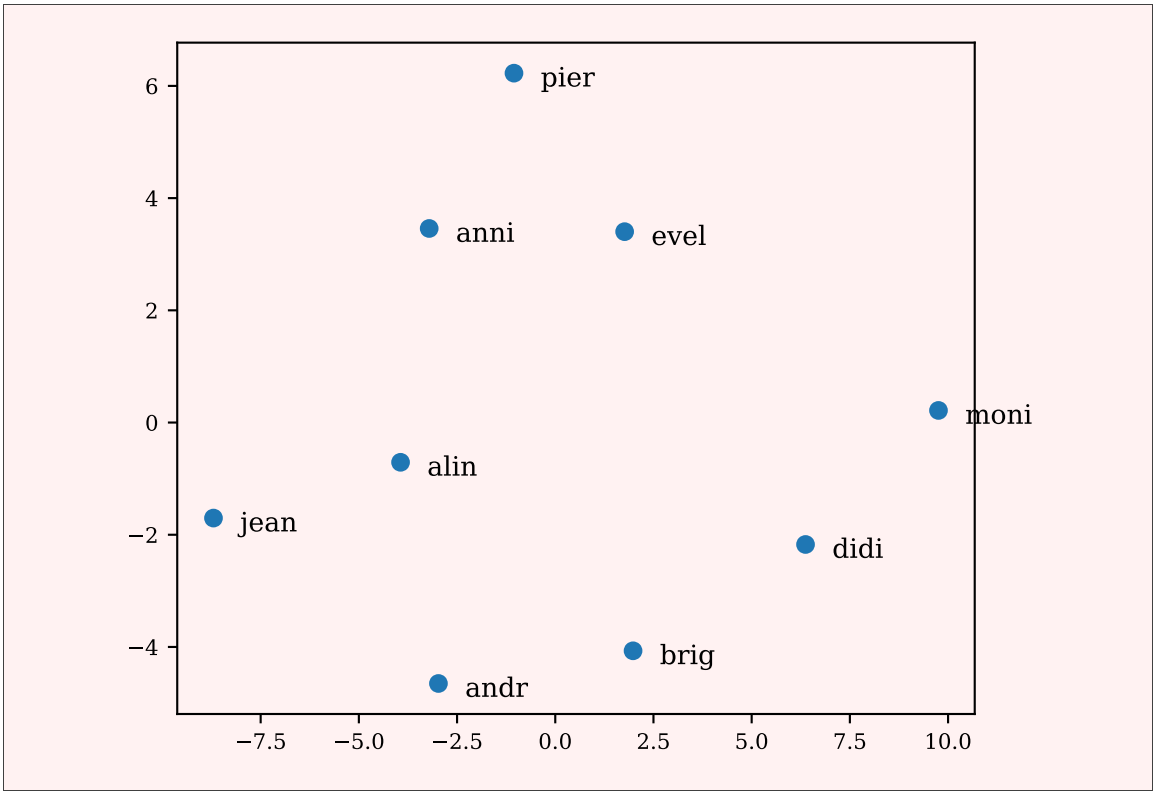
```
In  [1]: from sklearn.decomposition import PCA
         notes = pd.read_csv("data/notes.txt", sep="\s+")
         cls = PCA(n_components=5)
         pcs = cls.fit_transform(notes)
         cls.components_

Out [1]: array([[ 0.51453535,  0.50698853,  0.49235486,  0.48462835,
         ↪  0.03062778],
                [-0.56694916, -0.37199576,  0.65035356,  0.32323853,
                ↪  0.11289333],
                [ 0.05132308,  0.01445296, -0.10806565, -0.02254331,
                ↪  0.99245689],
                [-0.28874852,  0.55305647,  0.39373536, -0.67419539,
                ↪  0.03443659],
                [-0.57254891,  0.54635285, -0.40978192,  0.45343643,
                ↪  -0.01266839]])

In  [2]: plt.bar(["Axe 1", "Axe 2", "Axe 3", "Axe 4", "Axe 5"],
         ↪  cls.explained_variance_ratio_)
         plt.show()
```



On retrouve le premier axe factoriel qui correspond à la moyenne globale sans compter la note en arts.

2 Visualiser les individus dans le premier plan factoriel. On pourra utiliser la fonction `add_labels` pour ajouter les étiquettes.

```
In  [3]: plt.scatter(pcs[:, 0], pcs[:, 1])
         add_labels(pcs[:, 0], pcs[:, 1], notes.index)
         plt.show()
```

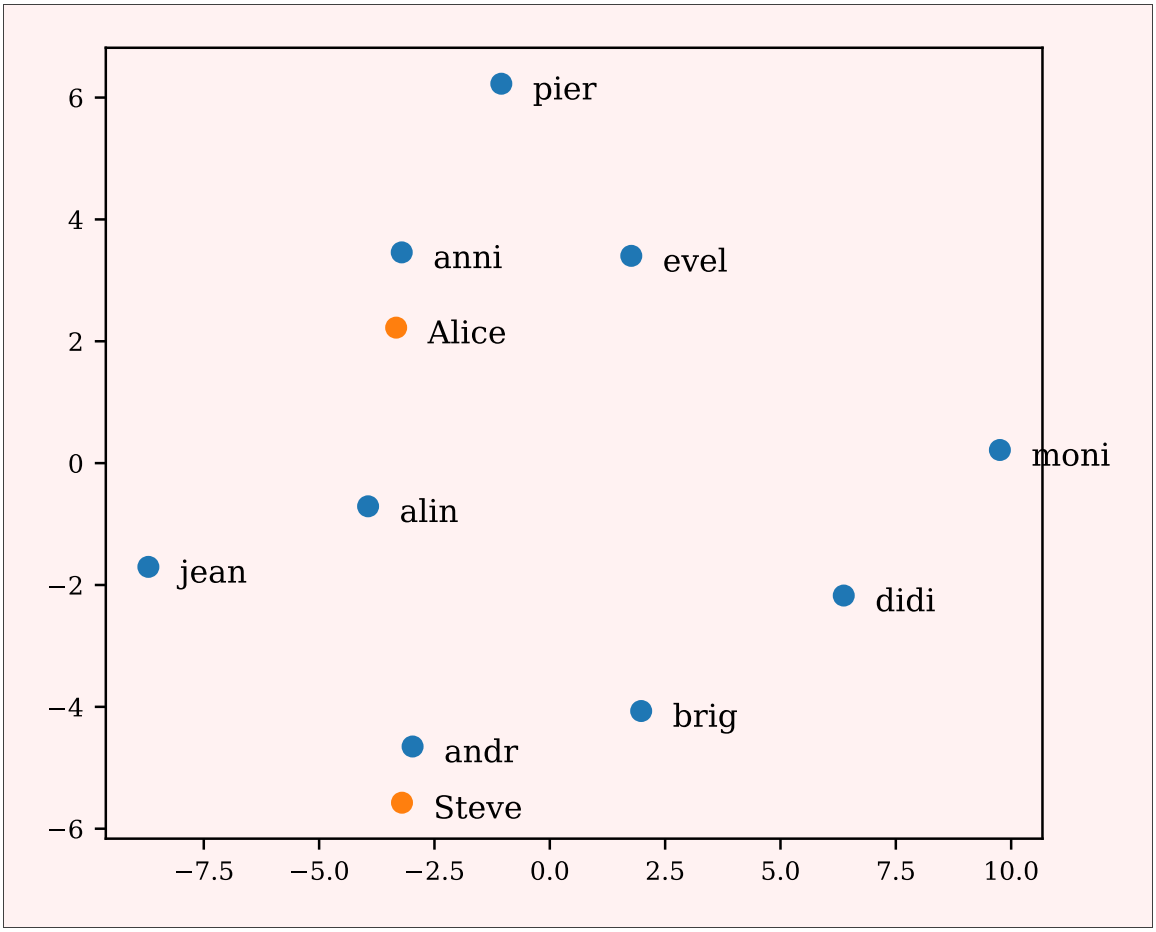


3 Deux nouveaux étudiants ont les notes suivantes :

Étudiant	math	scie	fran	lati	d-m
Alice	8.0	6.0	10.0	9	14
Steve	10	11	4.5	8.0	6

Représentez-les dans le premier plan factoriel.

```
In  [4]: plt.scatter(pcs[:, 0], pcs[:, 1])
         add_labels(pcs[:, 0], pcs[:, 1], notes.index)

         notes1 = pd.DataFrame(
             [[8.0, 6.0, 10.0, 9, 14], [10, 11, 4.5, 8.0, 6]],
             columns=notes.columns,
             index=["Alice", "Steve"],
         )
         pcs1 = cls.transform(notes1)
         plt.scatter(pcs1[:, 0], pcs1[:, 1])
         add_labels(pcs1[:, 0], pcs1[:, 1], notes1.index)
         plt.show()
```



1.2 ACP sur les données « Crabs »

On utilisera les données `Crabs` présentes dans le fichier `data/crabs.csv`. Ce jeu de données est constitué de 200 crabes décrits par huit variables (trois variables qualitatives, et cinq quantitatives).

4 Charger le jeu de données et sélectionner les variables quantitatives en utilisant le code Python suivant :

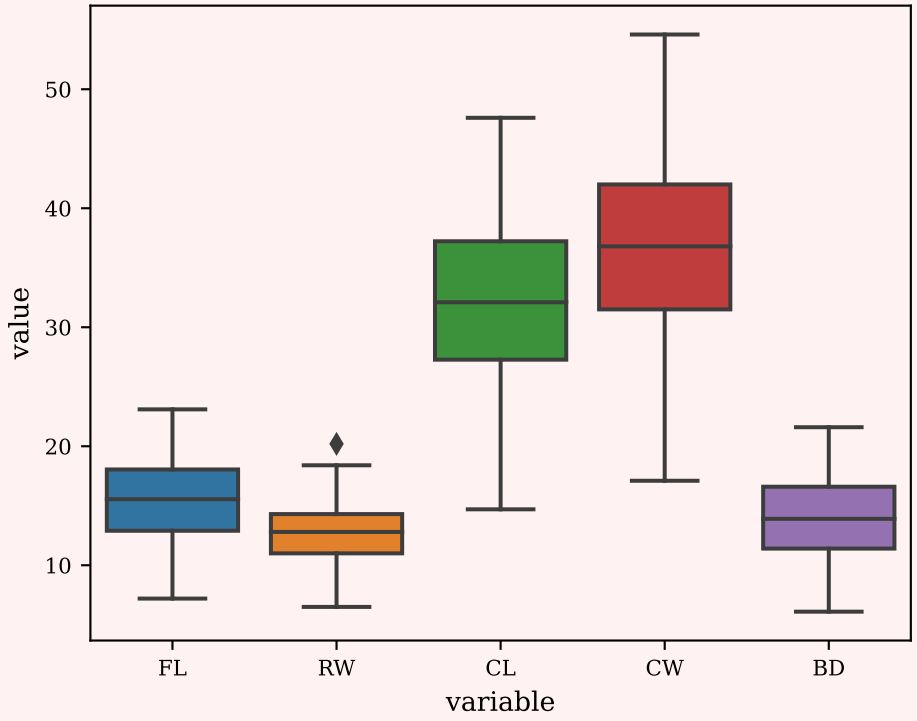
```
crabs = pd.read_csv("data/crabs.csv", sep="\s+")
crabsquant = crabs.iloc[:, 3:8]
```

```
In [5]: crabs = pd.read_csv("data/crabs.csv", sep="\s+")
        crabsquant = crabs.iloc[:, 3:8]
```

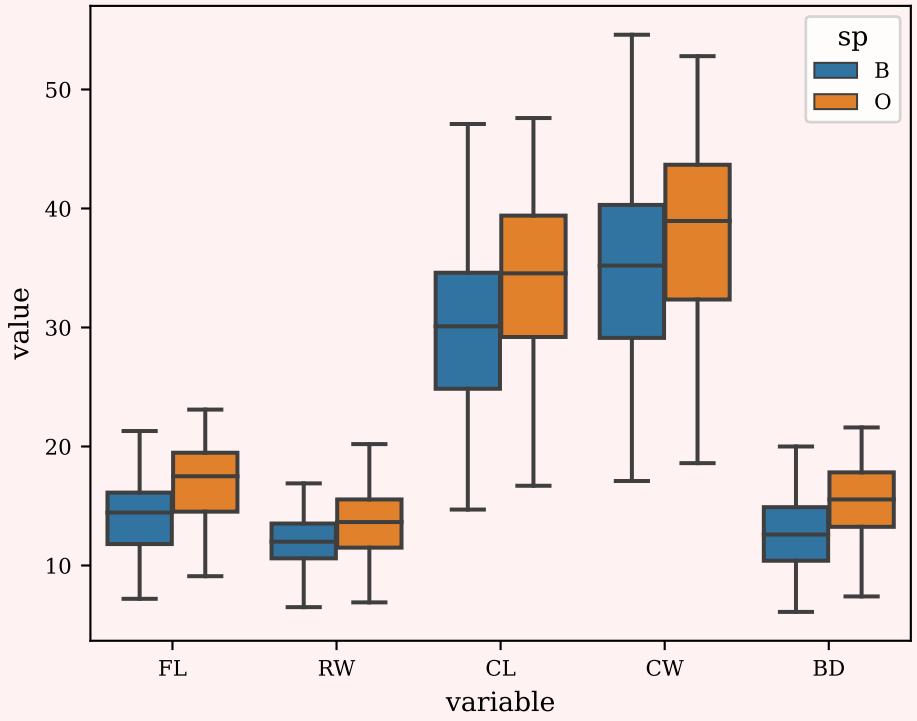
1.2.1 Analyse exploratoire

5 Effectuer dans un premier temps une analyse descriptive des données. On s'interrogera notamment sur les différences de caractéristiques morphologiques, en particulier selon l'espèce ou le sexe : semble-t-il possible d'identifier l'une ou l'autre à partir d'une ou plusieurs caractéristiques morphologiques ?

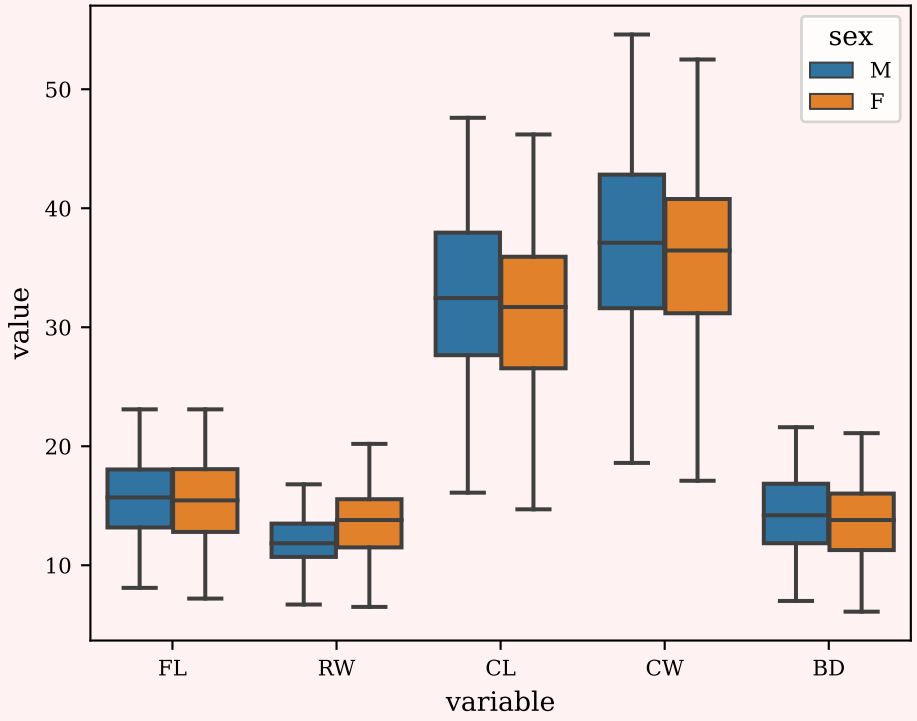
```
In [6]: crabs_long = crabs.melt(id_vars=["sp", "sex", "index"])
        sns.boxplot(x="variable", y="value", data=crabs_long)
        plt.show()
```



In [7]:
```
sns.boxplot(x="variable", y="value", hue="sp", data=crabs_long)
plt.show()
```

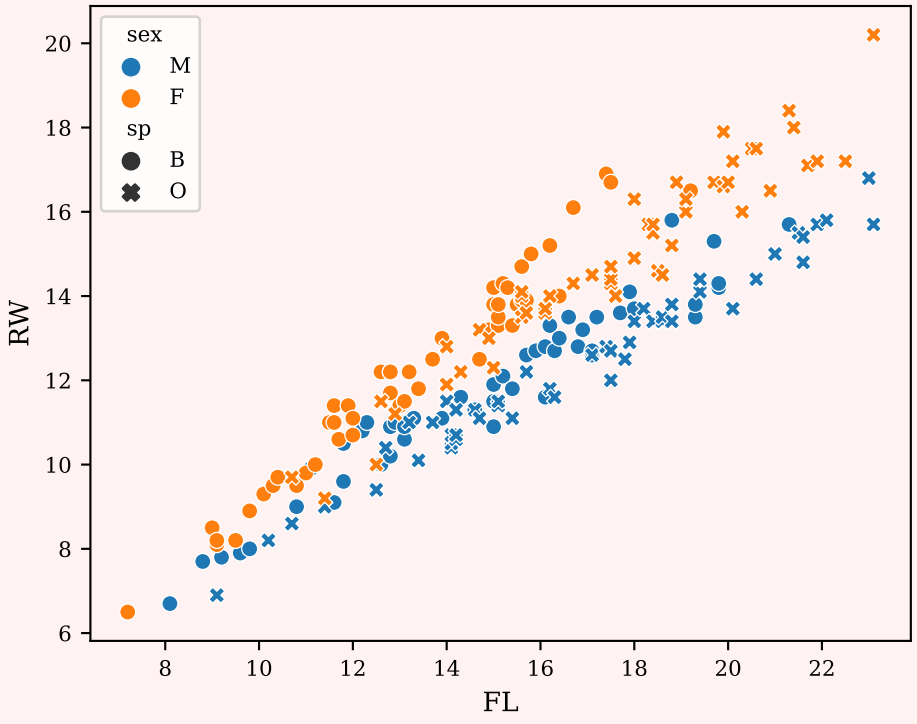


In [8]:
```
sns.boxplot(x="variable", y="value", hue="sex", data=crabs_long)
plt.show()
```

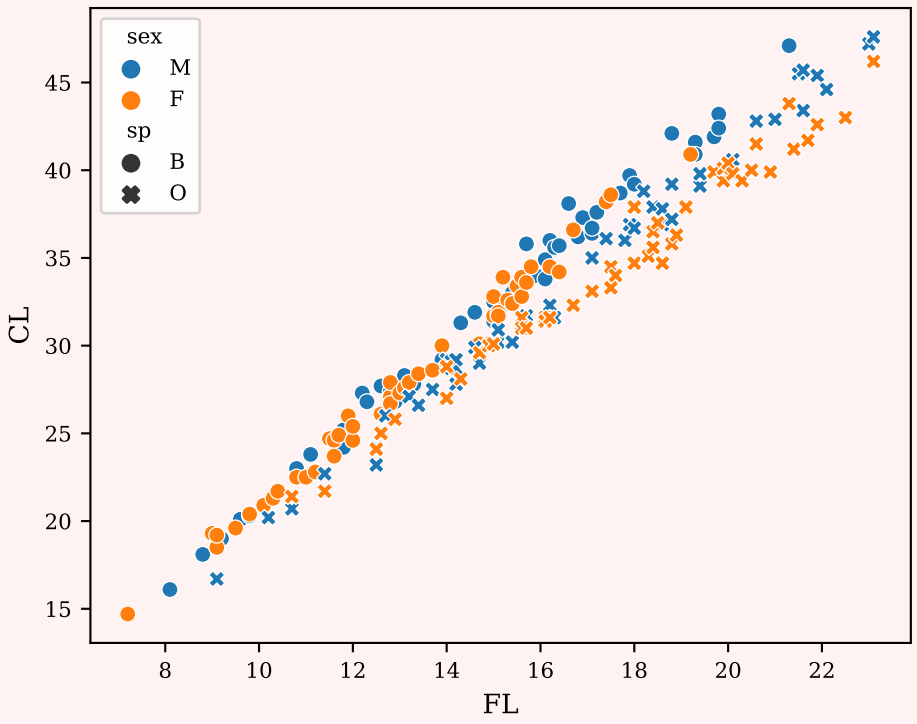


Le tracé des *boxplots* confirme que l'analyse monovariée permet de mettre en évidence des différences significatives au niveau des distributions (médianes significativement différentes) mais pas d'identifier l'espèce ou le sexe à partir d'une unique variable. Cela justifie une approche multivariée.

```
In [9]: sns.scatterplot(x="FL", y="RW", data=crabs, hue="sex", style="sp")
        plt.show()
```



```
In [10]: sns.scatterplot(x="FL", y="CL", data=crabs, hue="sex", style="sp")
         plt.show()
```



L'analyse des données brutes par un graphique de dispersion par paires met en évidence une très forte corrélation des différentes variables les unes aux autres. On est en présence d'un effet taille. Néanmoins, il est possible d'identifier des variables ou paires de variables pour lesquelles la distinction par espèce ou par sexe est visible.

Les couples CW/FL, et dans une moindre mesure BD/CW, semblent permettre de distinguer les deux espèces, qui restent néanmoins difficiles à séparer. Les combinaisons de RW avec chacune des autres variables semblent elles permettre de distinguer les crabes mâles des crabes femelles.

6 Dans un second temps, on étudiera la corrélation entre les différentes variables. Quelle en est vraisemblablement la cause ? Quel traitement est-il possible d'appliquer aux données pour s'affranchir de ce phénomène ?

Les graphiques de dispersion par paires comme le calcul des corrélations met en évidence les très grandes corrélations des variables les unes avec les autres. On est en présence d'un "effet taille", toutes ces variables représentant des mesures absolues (et non relatives à la taille globale du crabe).

L'idée pour s'affranchir de cet effet taille est donc de travailler sur des grandeurs relatives afin de mettre en évidence les différences morphologiques liées à l'espèce ou au sexe. Il s'agit donc de définir une variable que l'on assimilera à la taille globale du crabe, pour ensuite calculer les grandeurs relatives à cette variable. On pourra, au choix, définir la taille

— comme la somme des variables en présence :

```
In [11]: taille1 = crabs.FL + crabs.RW + crabs.CL + crabs.CW + crabs.BD
```

— comme la variable la plus corrélée aux autres :

```
In [12]: idx = np.argmax(crabsquant.corr().sum(axis=0).to_numpy())
         taille2 = crabsquant.iloc[:, idx]
```

On utilisera ensuite cette information de taille de manière à travailler sur des grandeurs relatives de la manière suivante :

```
In [13]: crabsquant2 = crabsquant.div(taille1, axis=0)
```

Si l'on a défini la taille comme étant la variable la plus corrélée aux autres, on prendra soin de la supprimer du tableau de données résultant :

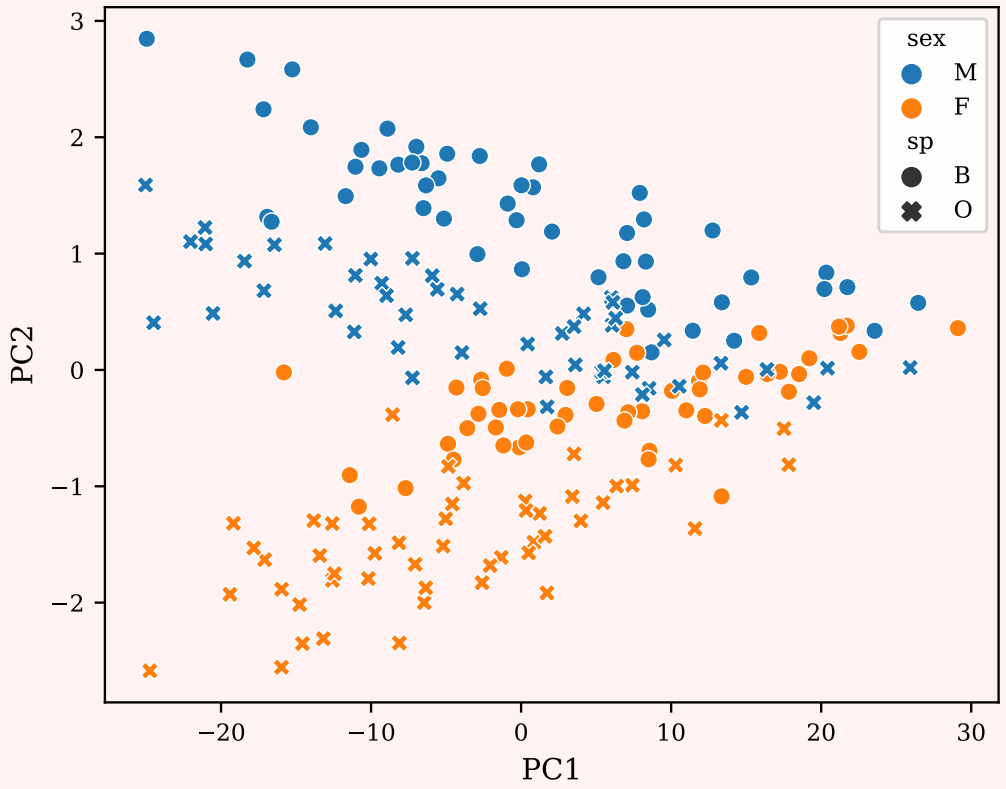
```
In [14]: crabsquant2 = crabsquant.div(taille2, axis=0)
         crabsquant2 = crabsquant2.drop(crabsquant2.columns[idx], axis=1)
```

1.2.2 ACP des données « Crabs »

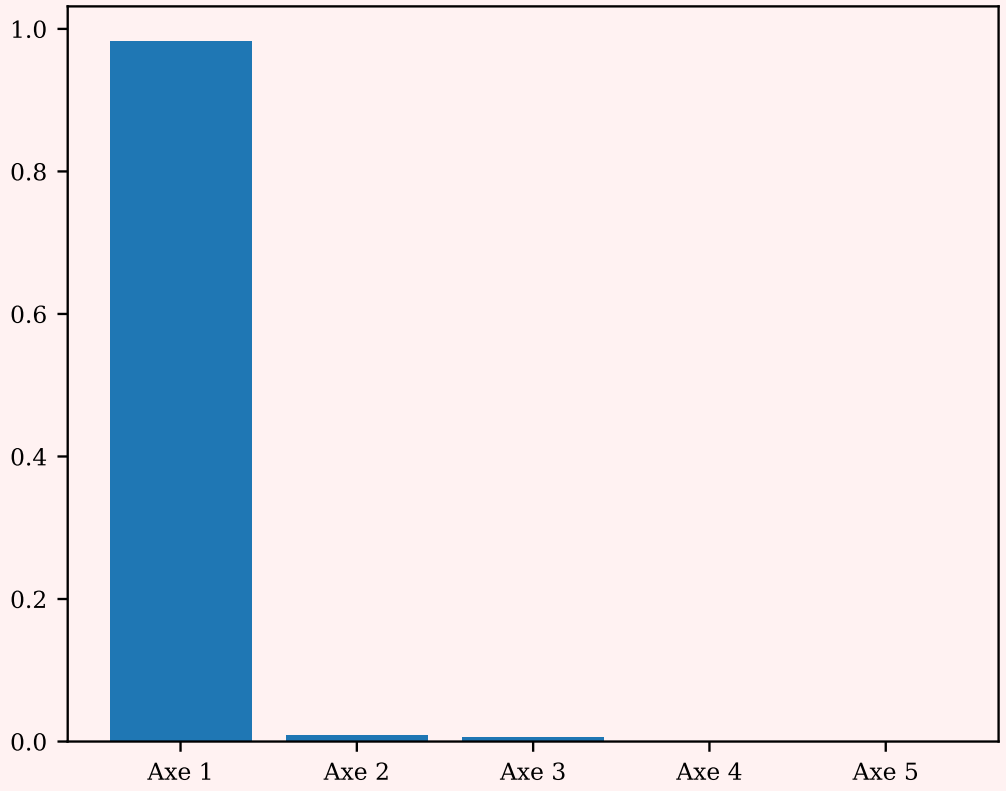
Cette étude vise à utiliser l'ACP pour trouver une représentation des crabes qui permettent de distinguer visuellement différents groupes, liés à l'espèce et au sexe.

1. Tester tout d'abord l'ACP sur `crabsquant` sans traitement préalable. Que constatez-vous ? Comment pouvez-vous expliquer ce phénomène à la lumière de l'analyse exploratoire de ces données menée préalablement ?

```
In [15]: from sklearn.decomposition import PCA
         cls = PCA(n_components=5)
         pcs_crabs = cls.fit_transform(crabsquant)

         df_crabs = pd.DataFrame(pcs_crabs, columns=[f"PC{i}" for i in
         ↪ range(1, 6)])
         sns.scatterplot(x="PC1", y="PC2", hue=crabs.sex, style=crabs.sp,
         ↪ data=df_crabs)
         plt.show()
```




```
In [16]: plt.bar(["Axe 1", "Axe 2", "Axe 3", "Axe 4", "Axe 5"],
         ↪ cls.explained_variance_ratio_)
         plt.show()
```



L'application de l'ACP sur les données `crabsquant` sans pré-traitement donne une représentation dans laquelle le premier axe porte une très grande partie de l'inertie. Ce phénomène est dû à l'effet taille évoqué plus haut.

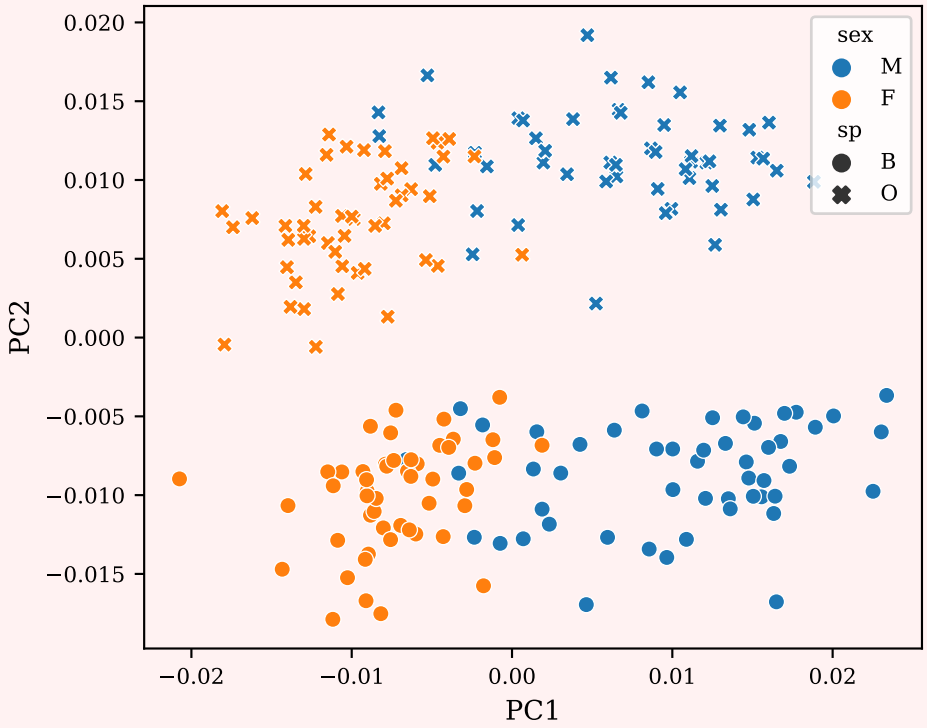
Le tableau de données contient en effet des mesures morphologiques relatives à 200 individus de toutes tailles : cette information est donc sous-jacente à chacune des grandeurs consignées dans le tableau. En conséquence, c'est principalement cette information de taille qui se trouve à l'origine de la variation des mesures morphologiques dans le jeu de données.

2. Trouver une solution pour améliorer la qualité de votre représentation en termes de visualisation des différents groupes.

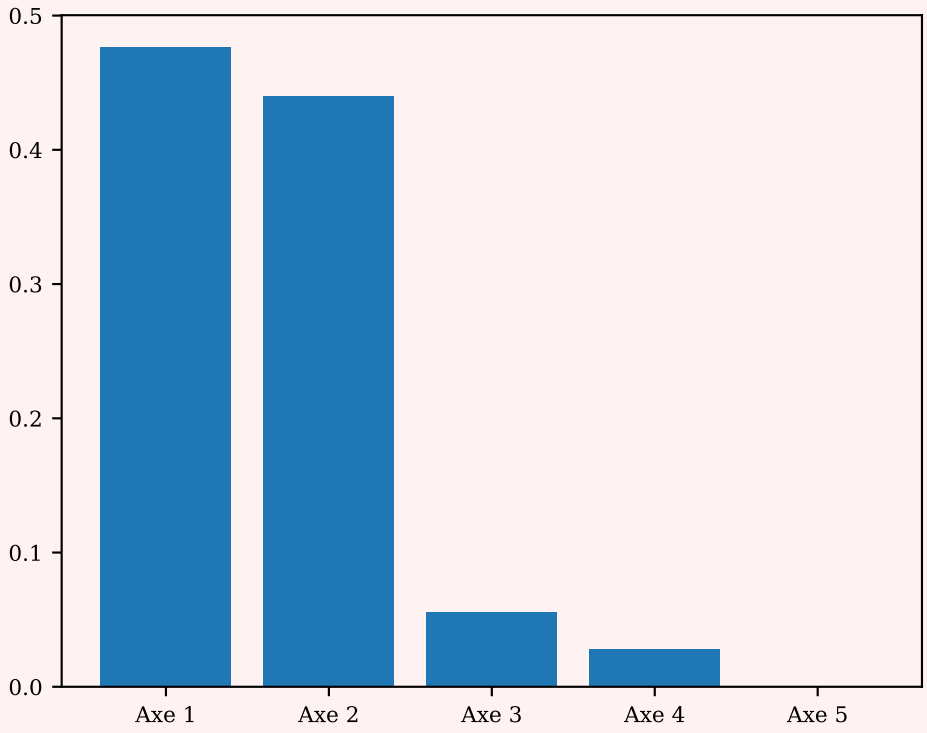
Il convient donc :

— soit de travailler sur les données pré-traitées comme suggéré dans la première partie du TP, la taille étant estimée par la somme des variables en présence,

```
In [17]: cls = PCA(n_components=5)
         pcs_crabs2 = cls.fit_transform(crabsquant2)

         df_crabs2 = pd.DataFrame(pcs_crabs2, columns=[f"PC{i}" for i
         ↪  in range(1, 6)])
         sns.scatterplot(x="PC1", y="PC2", hue=crabs.sex,
         ↪  style=crabs.sp, data=df_crabs2)
         plt.show()
```

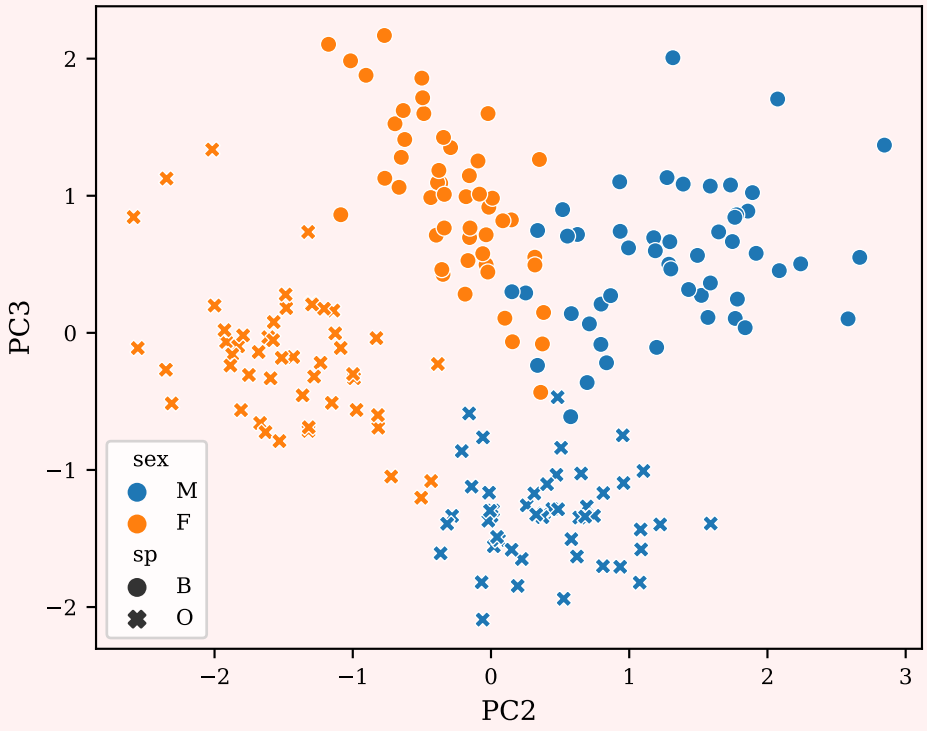



```
In [18]: plt.bar(["Axe 1", "Axe 2", "Axe 3", "Axe 4", "Axe 5"],
         ↪  cls.explained_variance_ratio_)
         plt.show()
```



— soit de supprimer simplement le premier axe factoriel et de représenter les données dans le plan formé par les second et troisième axes.

```
In [19]: sns.scatterplot(x="PC2", y="PC3", hue=crabs.sex,
             ↪ style=crabs.sp, data=df_crabs)
         plt.show()
```



Les deux stratégies permettent de distinguer les groupes de crabes correspondant aux différentes combinaisons espèce-sexe.

2 Exercices

2.1 Exercice pratico-théorique : ACP « à la main »

On associera dans cet exercice les mêmes pondérations à tous les individus, et on munira $\mathbb{R}^p$ de la métrique euclidienne.

7 Charger le jeu de données « notes » et définir les matrices M et D_p.

```
In [20]: df = pd.read_csv("data/notes.txt", sep="\s+")
         notes = df.to_numpy()
         n, p = notes.shape
         M = np.eye(p)
         Dp = 1/n * np.eye(n)
```

8 Centrer le jeu de donnée.

```
In [21]: notes_mean = notes.mean(axis=0)
         notes = notes - notes_mean
```

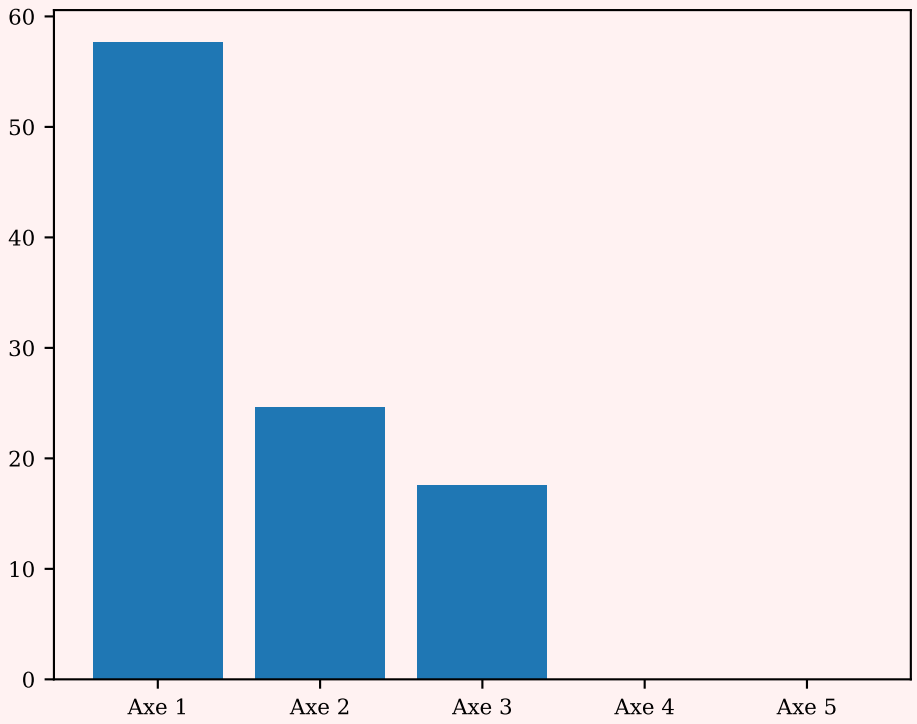
9 Calculer la matrice V telle que définie dans le cours et calculer les valeurs et vecteurs propres de la matrice VM en les ordonnant par ordre décroissant. On pourra utiliser la fonction `eigh` du module `linalg` :

```
import scipy.linalg as linalg
```

```
In [22]: V = notes.T @ Dp @ notes

         import scipy.linalg as linalg
         valp, vecp = linalg.eigh(V @ M)
         valp = valp[::-1]                                    # Ordre décroissant
         vecp = vecp[:, ::-1]
```

10 En déduire les axes factoriels de l'ACP du nuage de points défini par le jeu de données `notes`. Quels sont les pourcentages d'inertie expliquée par chacun de ces axes ?

```
In [23]: percents = 100 * valp / sum(valp)
         plt.bar(["Axe 1", "Axe 2", "Axe 3", "Axe 4", "Axe 5"], percents)
         plt.show()
```

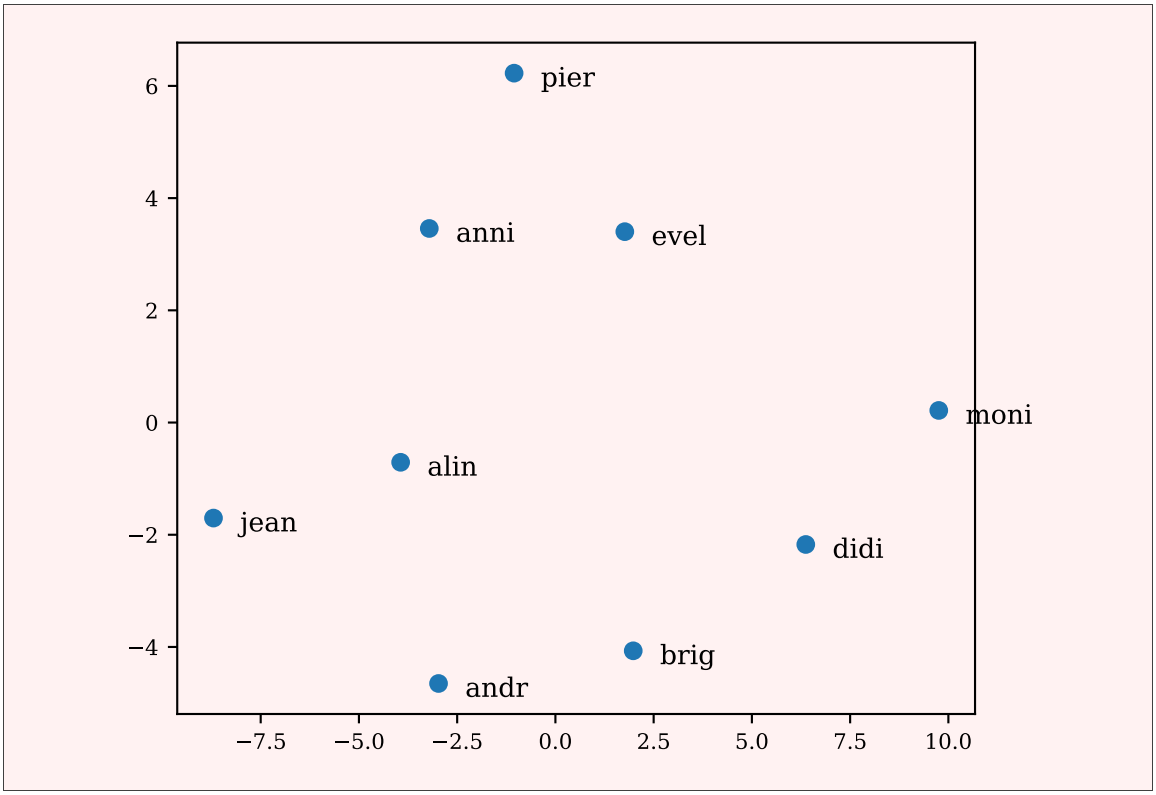


On retrouve les axes factoriels à quelques différences près :

- `scikit-learn` utilise une matrice de covariance légèrement différente (matrice de covariance empirique corrigée, où l'on divise par $n-1$ au lieu de n), d'où les différences ;
- les axes (qui sont des vecteurs propres) sont déterminés au signe près. Le signe dépend de l'algorithme de calcul des vecteurs propres utilisé.

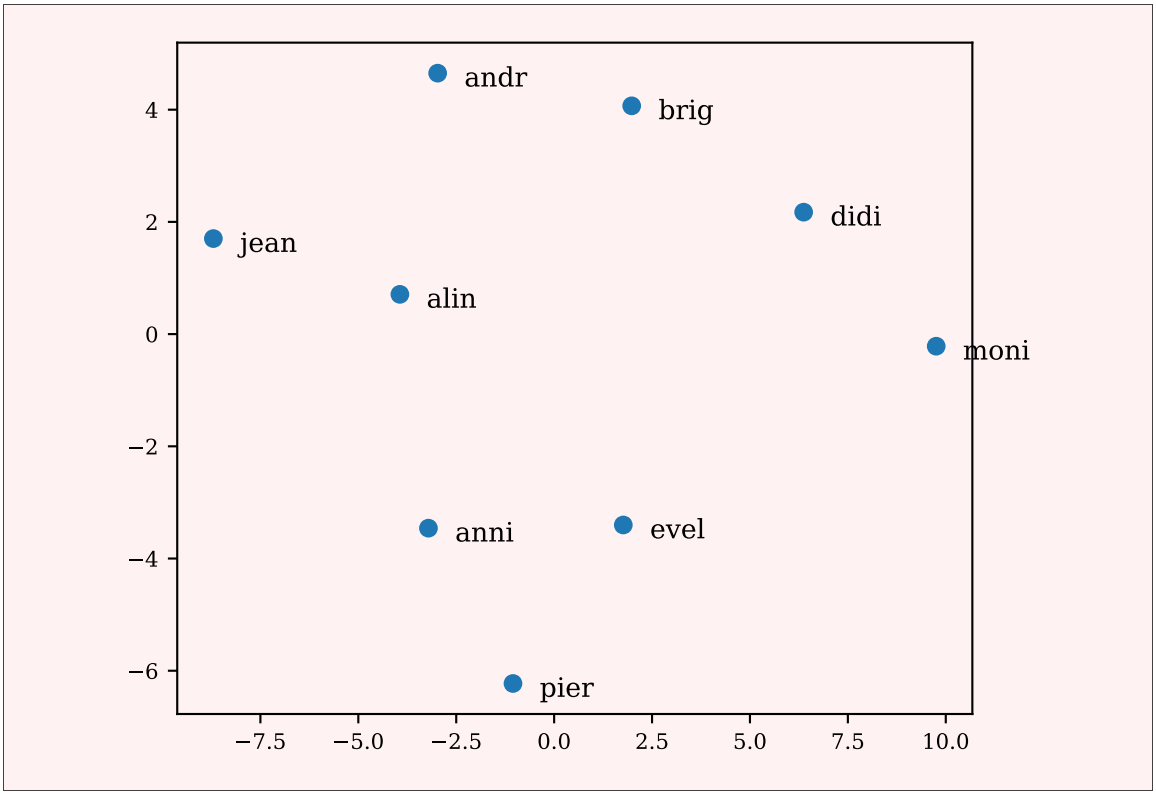
11 Calculer les composantes principales à l'aide des vecteurs propres calculés précédemment.

```
In [24]: U = vecp
         C = notes @ M @ U

         plt.scatter(-C[:, 0], -C[:, 1])
         add_labels(-C[:, 0], -C[:, 1], df.index)
         plt.show()
```



12 Retrouver les composantes principales à l'aide de la matrice W.

```
In [25]: W = notes @ M @ notes.T

         import scipy.linalg as linalg
         valp, vecp = linalg.eigh(W @ Dp)

         # Forçage à 0 (les valeurs propres doivent être positives)
         valp = np.maximum(valp, 0)

         valp = valp[::-1]                                    # Ordre décroissant
         vecp = vecp[:, ::-1]

         # Les composantes principales sont normées. On ajuste la variance pour
         # coincider avec les valeurs propres.
         C = vecp * np.sqrt((valp /  np.sum(Dp @ vecp**2, axis=0)))

         plt.scatter(C[:, 0], C[:, 1])
         add_labels(C[:, 0], C[:, 1], df.index)
         plt.show()
```



2.2 ACP et dualité

13 Soit $A \in \mathbb{R}^{n\times m}$ et $B \in \mathbb{R}^{m\times n}$ deux matrices. Montrer que AB et BA ont les mêmes valeurs propres non nulles.

Soit λ une valeur propre non nulle de la matrice AB. Il existe donc un vecteur non nul x tel que

$$ABx = \lambda x, \quad \Rightarrow \quad BABx = \lambda Bx,$$

en composant par B à gauche. En posant $y = Bx$, on obtient donc $BAy = \lambda y$. Remarquons enfin que le vecteur y est non nul : dans le cas contraire, on aurait $\lambda = 0$, ce qui est exclu par hypothèse. Ainsi, si λ est valeur propre (non nulle) de AB de vecteur propre associé x, alors λ est valeur propre (non nulle) de la matrice BA de vecteur propre associé $y = Bx$.

On peut appliquer le même raisonnement à la matrice BA : on en déduit ainsi que si λ est valeur propre (non nulle) de BA, de vecteur propre associé $y = Bx$, alors λ est valeur propre non nulle de BA, de vecteur propre associé $x' = Ay$.

En notant Λ_X l'ensemble des valeurs propres non nulles d'une matrice carrée X, on vient de montrer successivement que

$$\lambda \in \Lambda_{AB} \Rightarrow \lambda \in \Lambda_{BA} \quad \text{puis que} \quad \lambda \in \Lambda_{BA} \Rightarrow \lambda \in \Lambda_{AB};$$

En d'autres termes, $\Lambda_{AB} = \Lambda_{BA}$.

14 En déduire que les valeurs propres non nulles de VM et WD_p sont identiques.

On pose $A = X^T D_p$ et $B = XM$. On a alors $VM = AB$ et $WD_p = BA$. En appliquant le résultat précédent, les valeurs propres non nulles de VM et WD_p sont les mêmes.

15 Montrer que l'application $\phi_B : x \mapsto Bx$ envoie les vecteurs propres de AB associés à une valeur propre non nulle vers les vecteurs propres de BA associés à une valeur propre non nulle et que

l'application $\phi_A : x \mapsto Ax$ envoie les vecteurs propres de BA associés à une valeur propre non nulle vers les vecteurs propres de AB associés à une valeur propre non nulle.

En fait, on l'a déjà montré.

16 En déduire que si u est un vecteur propre associé à une valeur propre non nulle pour VM alors XMu est un vecteur propre pour WD_p associé à une valeur propre non nulle.

En déduire également que si v est un vecteur propre associé à une valeur propre non nulle pour WD_p alors $X^T D_p v$ est un vecteur propre pour VM associé à une valeur propre non nulle.

Application de la question précédente avec $A = X^T D_p$ et $B = XM$.

17 Montrer en plus que les vecteurs propres u de VM de norme 1 selon la métrique M sont envoyés vers des vecteurs propres v de WD_p de norme λ selon la métrique D_p lorsqu'on les multiplie par XM.

On pose $v = XMu$, on a alors

$$\begin{aligned} \|v\|_{D_p} &= v^T D_p v \\ &= u^T M X^T D_p X M u \\ &= u^T M V M u \\ &= \lambda u^T M u \\ &= \lambda \end{aligned}$$

En d'autres termes, si u est une direction factorielle de norme 1 (selon M), alors $v = XMu$ est la composante principale associée de norme λ (selon D_p).

18 Montrer en plus que les vecteurs propres v de WD_p de norme 1 selon la métrique D_p sont envoyés vers des vecteurs propres u de VM de norme λ selon la métrique M lorsqu'on multiplie par $X^T D_p$.

On pose $u = X^T D_p v$, on a alors

$$\begin{aligned} \|u\|_M &= u^T M u \\ &= v^T D_p X M X^T D_p v \\ &= v^T D_p W D_p v \\ &= \lambda v^T D_p v \\ &= \lambda \end{aligned}$$