SY09 Printemps 2022
TD/TP 03 — Représentation euclidienne des données

1 Représentation euclidienne

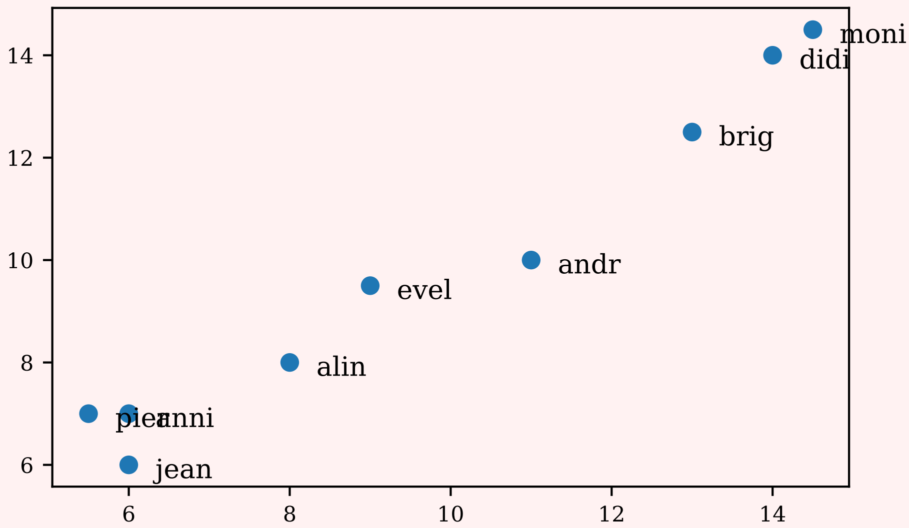
1.1 Représentation des données

Dans cette partie, on s'intéresse à un jeu de données contenant les notes de $n = 9$ individus pour $p = 5$ matières : mathématiques, sciences « naturelles » (physique-chimie), français, latin, « arts » (dessin-musique).

On souhaite étudier ces données et les représenter de manière à caractériser les individus en fonction de leur niveau scolaire. Plus particulièrement, on cherchera à déterminer une *base de représentation* permettant de visualiser « au mieux » les données.

$\boxed{1}$ Charger le jeu de données présent dans le fichier `data/notes.txt`. Visualiser la dispersion des notes. On pourra utiliser la fonction fournie `add_labels` pour ajouter les étiquettes.

```
In  [1]: notes = pd.read_csv("data/notes.txt", sep="\s+")
         plt.scatter(notes.math, notes.scie)
         add_labels(notes.math, notes.scie, notes.index)
         plt.show()
```

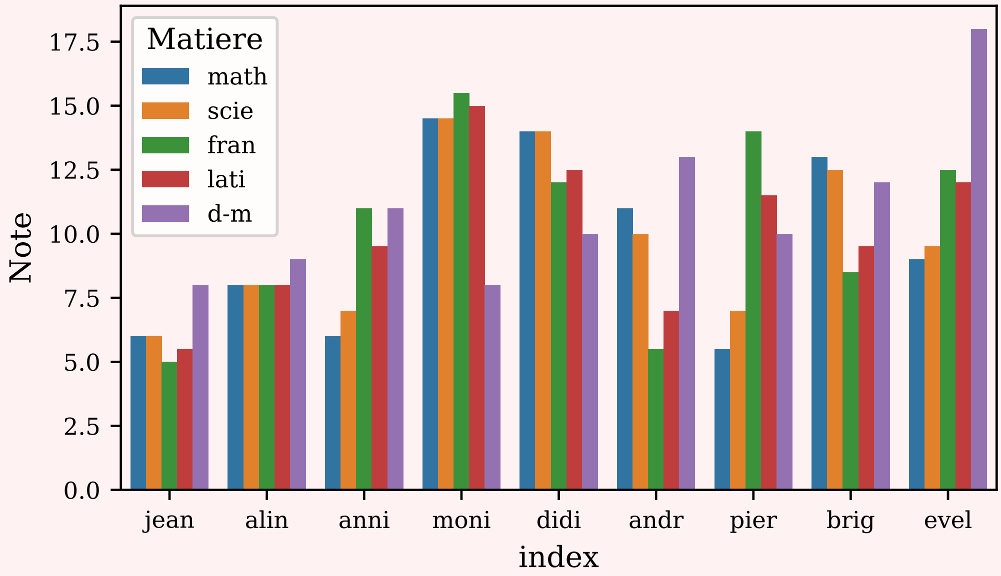


Analyse et représentation succinctes

$\boxed{2}$ Faire une brève analyse des données ; en particulier :

1. comment sont réparties les notes, dans chaque matière ?

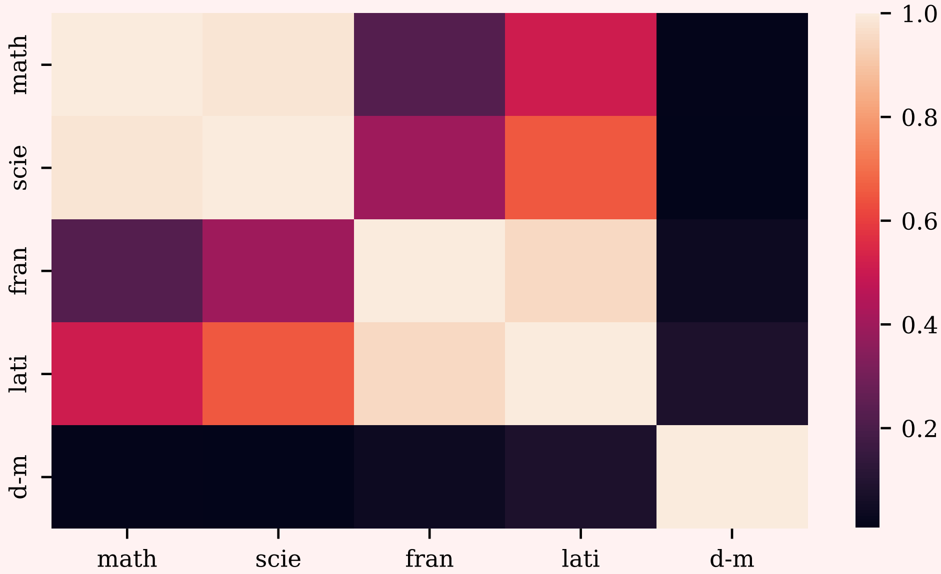
```
In [2]: notes0 = notes.reset_index()
        notes_melt = notes0.melt(id_vars=["index"], var_name="Matiere",
        ↪  value_name="Note")
        sns.barplot(x="index", y="Note", hue="Matiere", data=notes_melt)
        plt.show()
```



Pas de notes très basses (rien sous 5) ni très hautes (à l'exception d'un 18 en dessin-musique), avec des notes bonnes, moyennes et mauvaises dans chacune des matières.

2. Peut-on rapprocher certaines matières les unes des autres ?

```
In [3]: corr = notes.corr()
        sns.heatmap(corr)
        plt.show()
```

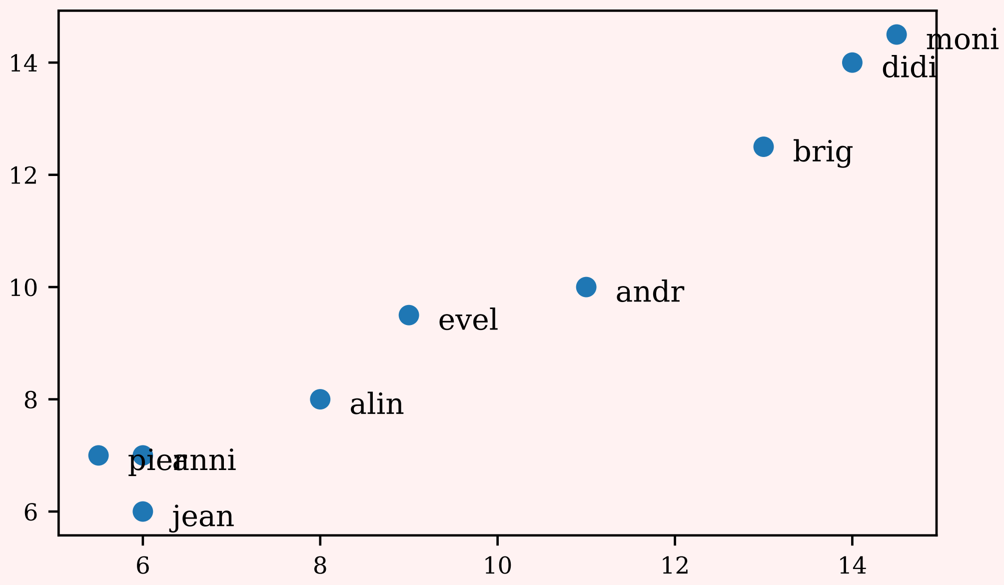


La matrice de corrélations indique que les matières scientifiques (maths-sciences) ainsi que les matières littéraires (français-latin) sont très corrélées entre elles. Les « arts » sont quant à eux décorrélés des autres matières.

3 On cherche à identifier des groupes d'élèves en fonction des résultats scolaires.

1. Représenter les élèves en fonction de leurs résultats dans les matières scientifiques ; quels sont les groupes d'individus qui semblent se détacher ?

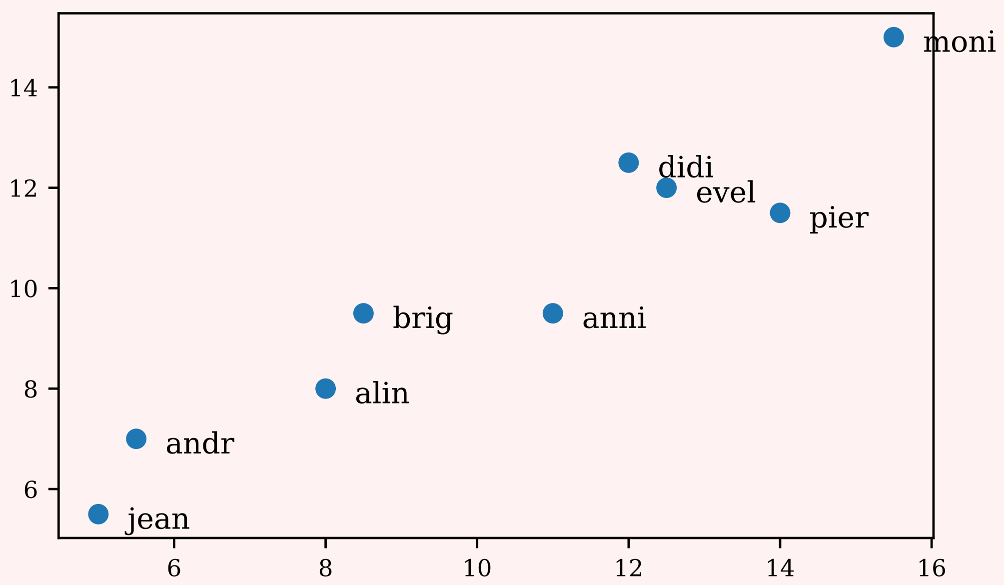
```
In [4]: plt.scatter(notes.math, notes.scie)
        add_labels(notes.math, notes.scie, notes.index)
        plt.show()
```



Les individus « brig », « didi » et « moni » se détachent.

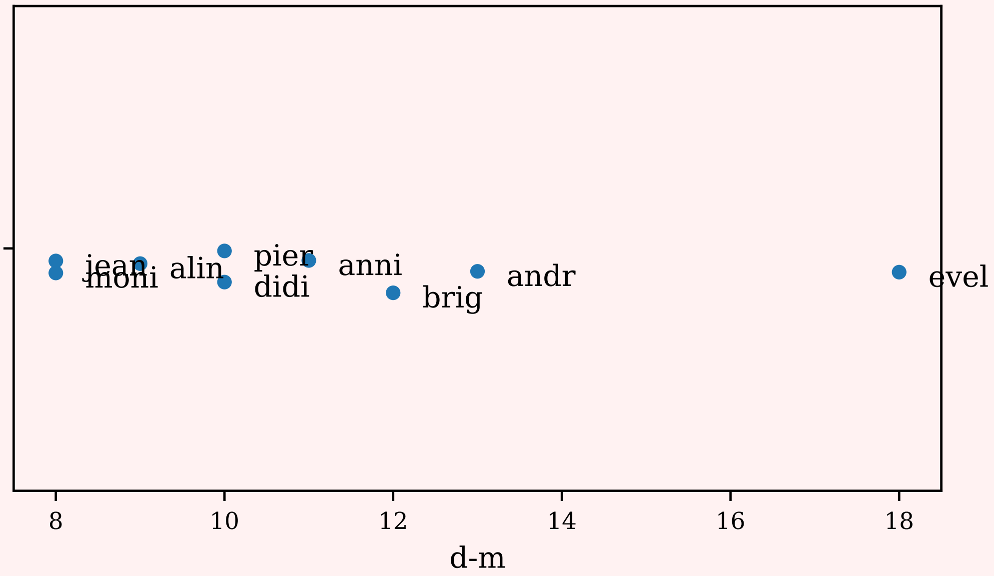
2. Faire de même avec les matières littéraires, puis avec les arts.

```
In [5]: plt.scatter(notes.fran, notes.lati)
        add_labels(notes.fran, notes.lati, notes.index)
        plt.show()
```



Les individus « didi », « evel », « pier » et « moni » se détachent.

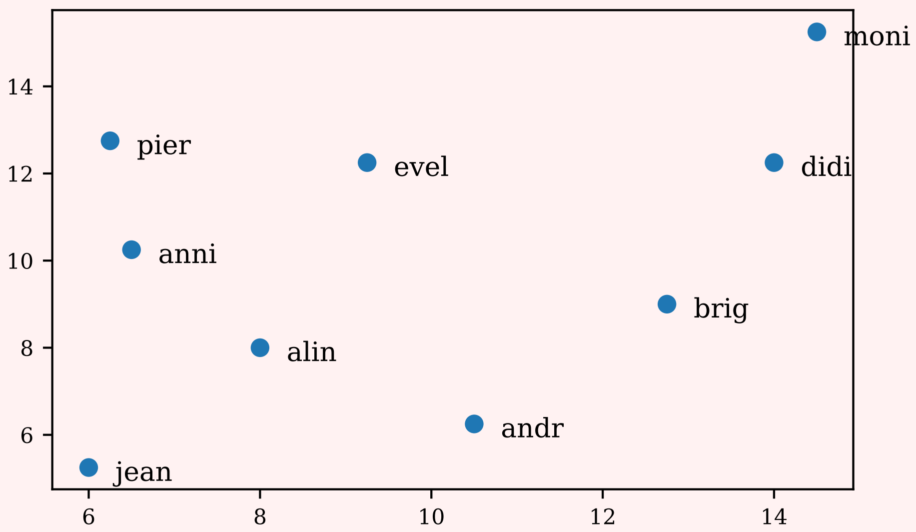
```
In [6]: ax = sns.stripplot(x=notes["d-m"])
        loc = ax.get_children()[0].get_offsets().data
        add_labels(*loc.T, notes.index)
        plt.show()
```



Seule « evel » se détache.

4 Représenter les élèves par deux informations : leur moyenne dans les matières scientifiques, et leur moyenne dans les matières littéraires. Interpréter les résultats obtenus.

```
In [7]: plt.scatter((notes.math + notes.scie) / 2, (notes.fran + notes.lati) /
        ↪ 2)
        add_labels((notes.math + notes.scie) / 2, (notes.fran + notes.lati) / 2,
        ↪ notes.index)
        plt.show()
```



Grosso modo, 1er quadrant : bons en sciences et en lettres, 2e quadrant : bons en lettres mais pas en sciences, 3e quadrant : mauvais en sciences et en lettres, 4e quadrant : bons en sciences mais pas en lettres.

Projection et qualité de représentation

5 On considère la matrice suivante :

$$A_1 = \begin{pmatrix} 1/2 & 0 & 1/2 & 0 & 0 \\ 1/2 & 0 & -1/2 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & -1/2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

1. Définit-elle une base ?

   Elle définit une base orthogonale mais pas orthonormale : les vecteurs la composant ne sont pas normés.

2. Comment exprimer les notes du tableau X dans la nouvelle base A_1, et comment revenir dans la base canonique de $\mathbb{R}^5$?

   On rappelle que si x_1 désigne un vecteur dans une base $\mathcal{B}_1$ et qu'on définit une base $\mathcal{B}_2$ à travers la matrice de passage P qui rassemble les coordonnées (en colonnes) des éléments de $\mathcal{B}_2$ dans la base $\mathcal{B}_1$ alors les nouvelles coordonnées de x_1 dans la base $\mathcal{B}_2$ notées x_2 vérifient $Px_2 = x_1$.

   Rappelons que les vecteurs sont rangés en ligne dans la matrice X et dans ses diverses représentations dans des bases de $\mathbb{R}^5$. En notant X_{A_1} les coordonnées dans la base A_1, la matrice de passage étant ici A_1, on trouve :

   $$A_1(X_{A_1})^T = X^T,$$

   d'où

   $$X_{A_1} = X\left(A_1^{-1}\right)^T. \tag{1}$$

3. Comment peut-on interpréter les coordonnées des individus exprimées dans cette nouvelle base ? Que permet de visualiser la représentation selon les composantes (X^1, X^2), ou (X^1, X^3), ou encore (X^2, X^4) ?

   La 1re composante représente la moyenne dans les matières scientifiques, la 2e la moyenne dans les matières littéraires. L'affichage dans le plan (X^1, X^2) permettra donc de distinguer les élèves selon leur niveau dans les matières scientifiques ou littéraires.

   La 3e représente la différence entre le niveau de maths et le niveau de sciences naturelles — couplée avec la 1re, elle permettra de distinguer, parmi les élèves moyens en sciences, ceux moyens partout de ceux bons dans une matière et mauvais dans l'autre. La 4e composante peut être interprétée comme la 3e (mais pour les matières littéraires : il conviendra donc de l'utiliser conjointement avec la seconde).

   La 5e composante, enfin, représente les élèves bons en arts, comme dans la base initiale (et ne communique aucune information concernant le niveau dans d'autres matières).

6 On considère à présent la matrice B_1 suivante :

$$B_1 = \begin{pmatrix} \sqrt{2}/2 & 0 & \sqrt{2}/2 & 0 & 0 \\ \sqrt{2}/2 & 0 & -\sqrt{2}/2 & 0 & 0 \\ 0 & \sqrt{2}/2 & 0 & \sqrt{2}/2 & 0 \\ 0 & \sqrt{2}/2 & 0 & -\sqrt{2}/2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

1. Définit-elle une base ?

La matrice B_1 correspond évidemment à la matrice A_1 après normalisation des vecteurs la composant. Elle définit donc une base orthonormée.

2. Quelles sont les coordonnées des individus de X dans la nouvelle base B_1 ? Comment peut-on interpréter les coordonnées des individus exprimés dans cette nouvelle base ? Que permet de visualiser la représentation selon les composantes (X^1, X^2), ou (X^1, X^3), ou encore (X^2, X^4) ?

On peut reprendre l'équation (1), en remplaçant A_1 par B_1 : en utilisant le fait que la matrice B_1 est orthogonale (c'est-à-dire $B_1^{-1} = B_1^\top$), on obtient

$$X_{B_1} = XB_1. \tag{2}$$

On peut interpréter les projections des individus de la même manière que dans la base A_1, si ce n'est que les vecteurs définissant B_1 étant normés, l'éloignement des points sera différent. Par exemple, dans le plan (X^1, X^5), le même élève sera désormais plus éloigné de l'origine qu'auparavant.

$\boxed{7}$ On considère à présent la matrice B_2 suivante :

$$B_2 = \begin{pmatrix} \sqrt{2}/2 & 0 & \sqrt{2}/2 & 0 & 0 \\ 0 & \sqrt{2}/2 & 0 & \sqrt{2}/2 & 0 \\ \sqrt{2}/2 & 0 & -\sqrt{2}/2 & 0 & 0 \\ 0 & \sqrt{2}/2 & 0 & -\sqrt{2}/2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

1. Définit-elle une base ?

La matrice B_2 définit une base orthonormée (on vérifie facilement que $u_i^T u_j = 0$ pour tout $i \neq j$ et que $u_i^T u_i = 1$ pour tout i).

2. Comment peut-on interpréter les coordonnées des individus exprimés dans cette nouvelle base ? Que permet de visualiser la représentation selon les composantes (X^1, X^2), ou (X^1, X^3), ou encore (X^2, X^4) ?

1re composante : moyenne de maths et de français, 2e : moyenne de sciences naturelles et latin, 3e : différence entre maths et français, 4e : différence entre sciences naturelles et latin, 5e : arts.

1.2 Choix d'une représentation

On définit la qualité de la représentation selon un axe comme étant la quantité d'inertie expliquée par cet axe.

$\boxed{8}$ Quelle est la quantité d'inertie du nuage de points expliquée par chacun des axes, si l'on considère la base canonique de $\mathbb{R}^5$? Quelle est la quantité d'inertie totale du nuage de points ? On pourra utiliser `np.cov` avec les bons arguments.

```
In  [8]: Xc = notes[['math', 'scie', 'fran', 'lati', 'd-m']].to_numpy()
         C = np.cov(Xc, bias=True, rowvar=False)
         np.diag(C).sum()

Out [8]: 48.97530864197531
```

La quantité totale d'inertie expliquée par les axes est égale à 48.97531.

$\boxed{9}$ Quelle est la quantité d'inertie expliquée par les axes si l'on considère la base A_1 ? Pour inverser une matrice, on pourra utiliser le sous-module `linalg` avec l'instruction

```
import numpy.linalg as linalg
```

```
In  [9]: A1 = np.array(
             [
                 [0.5, 0, 0.5, 0, 0],
                 [0.5, 0, -0.5, 0, 0],
                 [0, 0.5, 0, 0.5, 0],
                 [0, 0.5, 0, -0.5, 0],
                 [0, 0, 0, 0, 1],
             ]
         )
         import numpy.linalg as linalg
         C1 = np.cov(Xc @ linalg.inv(A1).T, bias=True, rowvar=False)
         np.diag(C1).sum()
Out [9]: 89.28395061728395
```

L'inertie totale n'est pas conservée lors du changement de base codé par `A1`.

$\boxed{10}$ Qu'en est-il pour la base B_1, pour la base B_2 ?

```
In  [10]: from math import sqrt
          r = sqrt(2)/2
          B1 = np.array(
              [
                  [r, 0, r, 0, 0],
                  [r, 0, -r, 0, 0],
                  [0, r, 0, r, 0],
                  [0, r, 0, -r, 0],
                  [0, 0, 0, 0, 1],
              ]
          )
          C1 = np.cov(Xc @ B1, bias=True, rowvar=False)
          np.diag(C1).sum()
Out [10]: 48.97530864197531

In  [11]: B2 = np.array(
              [
                  [r, 0, r, 0, 0],
                  [0, r, 0, r, 0],
                  [r, 0, -r, 0, 0],
                  [0, r, 0, -r, 0],
                  [0, 0, 0, 0, 1],
              ]
          )
          C2 = np.cov(Xc @ B2, bias=True, rowvar=False)
          np.diag(C2).sum()
Out [11]: 48.97530864197531
```

On utilise le fait que les matrices de passage sont orthogonales : $\left(B_1^{-1}\right)^T = B_1$ et $\left(B_2^{-1}\right)^T = B_2$. Lorsque le changement de base se fait vers une base orthonormale également, l'inertie est conservée.

$\boxed{11}$ On cherche à représenter le nuage de points dans un plan, au prix d'une perte d'information. Quels axes choisirait-on, parmi ceux définis par la base canonique, par B_1, ou par B_2 ? Pourquoi ? Interpréter.

On a les inerties expliquées suivantes :

	Axe 1	Axe 2	Axe 3	Axe 4	Axe 5
Canonique	11.388889	8.944444	12.061728	7.913580	8.666667
B_1	20.083333	19.280864	0.250000	0.694444	8.666667
B_2	14.382716	13.910494	9.067901	2.947531	8.666667

Et en pourcentage

	Axe 1	Axe 2	Axe 3	Axe 4	Axe 5
Canonique	23.254348	18.263171	24.628183	16.158306	17.695992
B_1	41.007058	39.368540	0.510461	1.417948	17.695992
B_2	29.367280	28.403075	18.515251	6.018402	17.695992

On choisirait les axes définis par les deux premiers vecteurs de B_1 : ce sont ceux qui expliquent la plus grande partie de l'inertie du jeu de données avec respectivement 41.0070582% et 39.3685405%.

En comparaison, la base B_2 explique 29.3672801% et 28.4030754% et la base de départ 23.2543484% et 18.2631712%.

On notera qu'avec cette stratégie (représentation dans un plan choisi en fonction de l'inertie), on perd complètement l'information portée par le 5ᵉ axe (c'est-à-dire le résultat en arts). Il convient donc de garder à l'esprit que le critère de « qualité » utilisé ici est quantitatif. Pour donner davantage de poids à cette matière, il aurait été possible d'utiliser une métrique donnant une pondération plus importante à la 5ᵉ variable.

$\boxed{12}$ On considère à présent la matrice B_3 suivante :

$$B_3 = \begin{pmatrix} 1/2 & 1/2 & 1/2 & 1/2 & 0 \\ 1/2 & 1/2 & -1/2 & -1/2 & 0 \\ 1/2 & -1/2 & -1/2 & 1/2 & 0 \\ 1/2 & -1/2 & 1/2 & -1/2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

La matrice B_3 est elle aussi orthogonale. Est-elle meilleure que B_1 en terme d'inertie expliquée ?

```
In  [12]: B3 = 1/2 * np.array(
              [
                  [1, 1, 1, 1, 0],
                  [1, 1, -1, -1, 0],
                  [1, -1, -1, 1, 0],
                  [1, -1, 1, -1, 0],
                  [0, 0, 0, 0, 2],
              ]
          )
          C3 = np.cov(Xc @ B3, bias=True, rowvar=False)
          np.diag(C3).sum()
Out [12]: 48.9753086419753
In  [13]: np.diag(C3)[:2].sum()
Out [13]: 39.364197530864196
In  [14]: np.diag(C1)[:2].sum()
Out [14]: 39.3641975308642
In  [15]: np.diag(C3)[0]
Out [15]: 28.223765432098766
In  [16]: np.diag(C1)[0]
Out [16]: 20.083333333333336
```

L'inertie totale est la même. L'inertie expliquée par les deux axes est la même. En revanche l'inertie expliquée par le premier axe est plus importante dans la base B_3.

2 Inertie cumulée enveloppante

L'idée de cette partie est de caractériser une bonne base orthogonale en terme d'inertie cumulée. Pour ce faire, on va avoir besoin de générer des bases orthogonales aléatoirement.

On admettra que le code suivant fournit une matrice orthogonale générée aléatoirement.

```
import numpy.linalg as linalg
U, _ = linalg.qr(np.random.randn(5, 5))
```

13 Créer une fonction qui renvoie les inerties cumulées associées au changement de base d'une matrice orthogonale choisie aléatoirement.

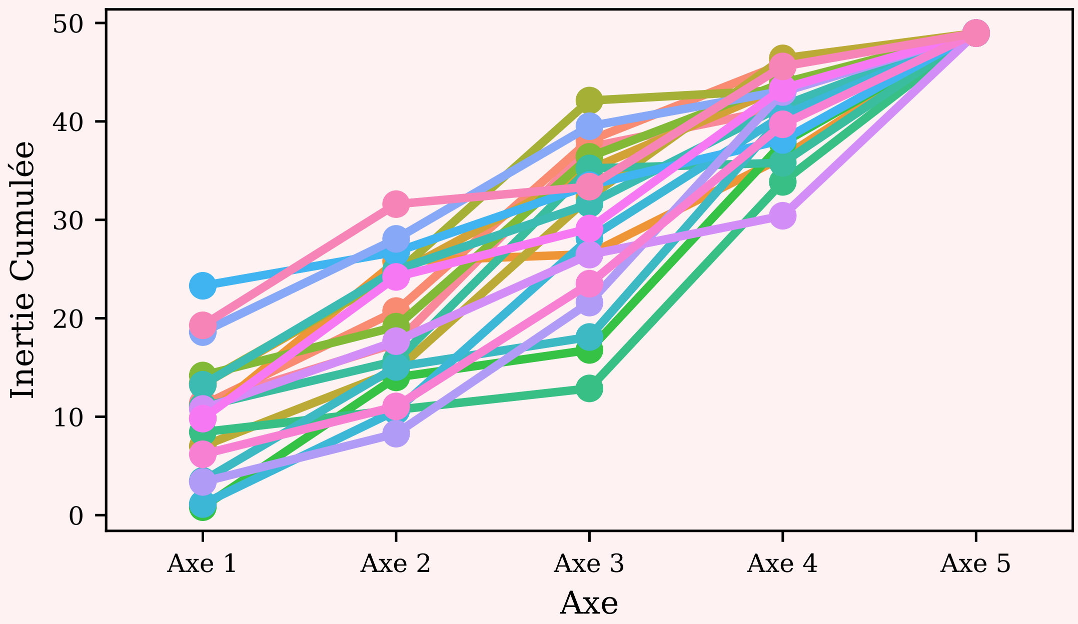
```
In  [17]: def random_cumsums():
              U, _ = linalg.qr(np.random.randn(5, 5))
              C = np.cov(Xc @ U, bias=True, rowvar=False)
              return np.cumsum(np.diag(C))
```

14 Afin de visualiser plusieurs courbes d'inertie cumulée pour différentes bases choisies aléatoirement, on veut former le jeu de données comportant les trois descripteurs suivants :
- l'identifiant de la base considérée,
- l'axe jusqu'auquel effectuer le cumul des inerties,
- l'inertie cumulée correspondante.

```
In  [18]: data = [random_cumsums() for i in range(20)]
          X0 = pd.DataFrame(data, columns=[f"Axe {i+1}" for i in range(5)])
          X0.index.name = "Base"
          X0 = X0.reset_index()
          X0 = X0.melt(id_vars="Base", var_name="Axe", value_name="Inertie
          ↪  Cumulée")
          X0

Out [18]:     Base    Axe  Inertie Cumulée
          0      0  Axe 1        11.370906
          1      1  Axe 1        11.154344
          2      2  Axe 1         9.844677
          3      3  Axe 1        13.256503
          4      4  Axe 1         7.009255
          ..   ...    ...              ...
          95    15  Axe 5        48.975309
          96    16  Axe 5        48.975309
          97    17  Axe 5        48.975309
          98    18  Axe 5        48.975309
          99    19  Axe 5        48.975309

          [100 rows x 3 columns]
```

15 Visualiser à l'aide de la fonction `sns.pointplot` les courbes pour chaque base considérée avec en abscisse le nombre d'axes et en ordonnée l'inertie cumulée. Quelle devrait être la propriété d'une base optimale en terme d'inertie expliquée ? Semble-t-il y en avoir une parmi les bases générées aléatoirement ?

```
In [19]: ax = sns.pointplot(x="Axe", y="Inertie Cumulée", hue="Base", data=X0,
         ↪   legend=False)
         ax.get_legend().remove()
         plt.show()
```



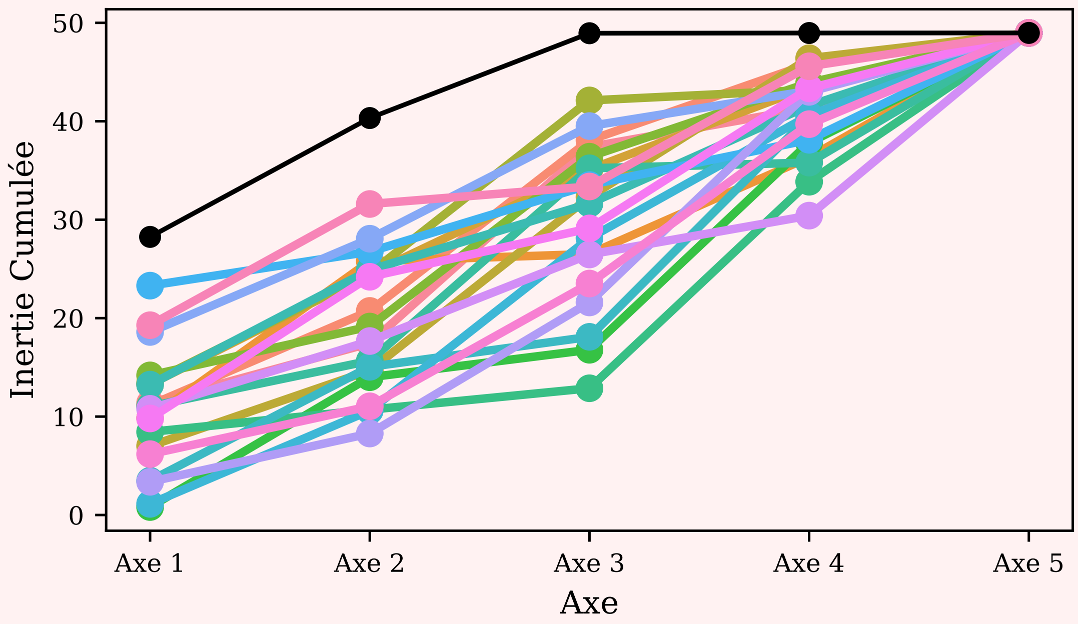
La courbe d'une base orthogonale optimale devrait dominer toute les autres. Il semble très peu probable de générer aléatoirement une base optimale.

$\boxed{16}$ Quel est la particularité de la base orthogonale définie par l'instruction suivante :

```
Bx = array([[0.515, -0.567, 0.051, -0.289, -0.573],
            [0.507, -0.372, 0.014, 0.553, 0.546],
            [0.492, 0.65, -0.108, 0.394, -0.41],
            [0.485, 0.323, -0.023, -0.674, 0.453],
            [0.031, 0.113, 0.992, 0.034, -0.013]])
```

→ il s'agit de la base du cours

```
In [20]: ax = sns.pointplot(x="Axe", y="Inertie Cumulée", hue="Base", data=X0,
         ↪   legend=False)
         ax.get_legend().remove()

         C = np.cov(Xc @ Bx, bias=True, rowvar=False)
         csx = np.cumsum(np.diag(C))

         ax.plot(range(5), csx, 'k-o', zorder=100)
         plt.show()
```



La courbe domine toutes les autres. Elle correspond à un changement de base optimal tel que calculé par l'ACP.

$\boxed{17}$ Que donne la matrice orthogonale suivante

$$B_y = B_x \begin{pmatrix} A & 0 & 0 \\ 0 & A & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

? Pas compris ?

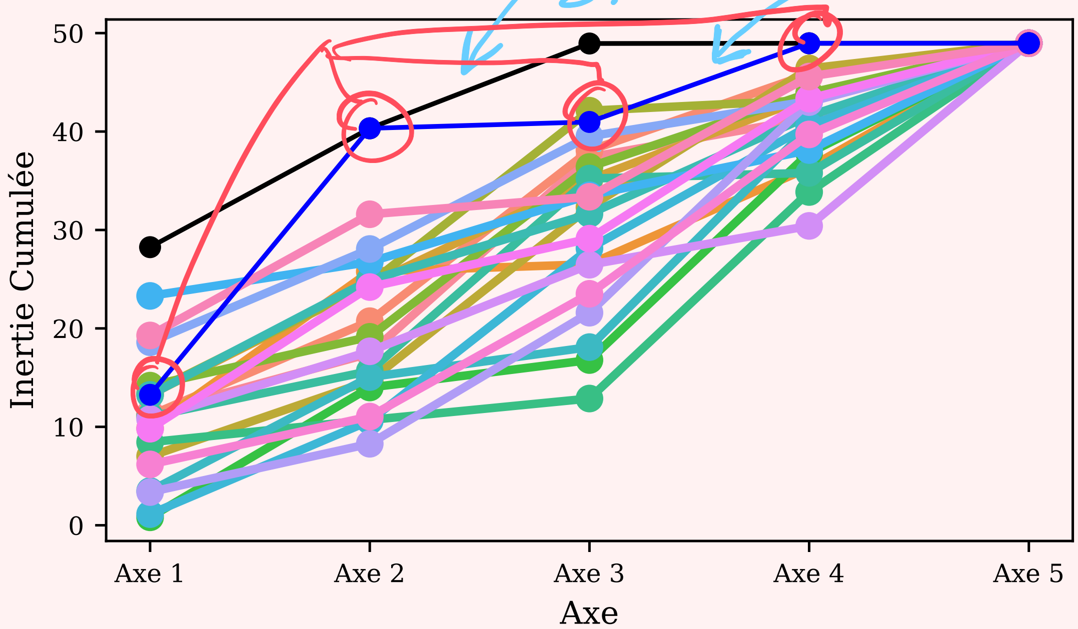
avec A une matrice orthogonale de taille 2 quelconque. Pourquoi ?

La matrice B_y est bien orthogonale. Elle mélange (orthogonalement) les deux premiers vecteurs de la base entre eux et les troisième et quatrième vecteurs de la base entre eux. Le premier axe n'est donc plus optimal. En revanche, le premier plan principal est le même pour les deux bases B_x et B_y. L'inertie expliquée par ce plan est donc la même. De même pour les axes 3 et 4. L'inertie cumulée est donc maximale si l'on utilise un sous-espace vectoriel de dimension 2 ou de dimension 4 pour représenter les données, mais la représentation dans un sous-espace de dimension 1 ou 3 est sous-optimale.

```
In  [21]: from scipy.linalg import block_diag
          A, _ = linalg.qr(np.random.randn(2, 2))

          By = Bx @ block_diag(A, A, 1)

          ax = sns.pointplot(x="Axe", y="Inertie Cumulée", hue="Base", data=X0,
          ↪  legend=False)
          ax.get_legend().remove()

          C = np.cov(Xc @ Bx, bias=True, rowvar=False)
          csx = np.cumsum(np.diag(C))

          ax.plot(range(5), csx, 'k-o', zorder=100)
          C = np.cov(Xc @ By, bias=True, rowvar=False)
          csy = np.cumsum(np.diag(C))

          ax.plot(range(5), csy, 'b-o', zorder=100) # noouput
Out [21]: [<matplotlib.lines.Line2D object at 0x12fd30dc0>]
In  [22]: plt.show()
```

By

Bx



3 Inertie par rapport à un axe

On se propose de vérifier expérimentalement les formules de Huygens. Pour cela, on utilise le code suivant

```
n, p = 100, 2
X = np.random.randn(n, p)

feat_metric = np.eye(p)
sample_metric = 1 / n * np.eye(n)
```

Im

DP

$\boxed{18}$ Compléter la fonction `inertia_factory` et définir deux fonctions `inertia_point`, `inertia_axe` qui calculent des inerties par rapport à un point ou à une droite.

Voir le fichier src/inertia.py.

```
In [23]: n, p = 100, 2
         X = np.random.randn(n, p)
         feat_metric = np.eye(p)
         sample_metric = 1 / n * np.eye(n)
         feat_inner, inertia_point, inertia_axe = inertia_factory(feat_metric,
          ↪   sample_metric)
```

19 Vérifier les fonctions avec le code suivant

```
X0 = np.array([[1, 0], [0, 1]])
n0, p0 = X0.shape
feat_metric0 = np.eye(p0)
sample_metric0 = 1 / n0 * np.eye(n0)
feat_inner0, inertia_point0, inertia_axe0 = inertia_factory(feat_metric0,
 ↪   sample_metric0)

import math
p0 = np.zeros(2)
v = np.array([0, 1])
p1 = np.array([-1, 0])
assert(math.isclose(inertia_point0(X0), .5))
assert(math.isclose(inertia_point0(X0, p0), 1.))
assert(math.isclose(inertia_axe0(X0, v), .5))
assert(math.isclose(inertia_axe0(X0, v, p1), 2.5))
```

```
In [24]: X0 = np.array([[1, 0], [0, 1]])
         n0, p0 = X0.shape
         feat_metric0 = np.eye(p0)
         sample_metric0 = 1 / n0 * np.eye(n0)
         feat_inner0, inertia_point0, inertia_axe0 =
          ↪   inertia_factory(feat_metric0, sample_metric0)

         import math
         p0 = np.zeros(2)
         v = np.array([0, 1])
         p1 = np.array([-1, 0])
         assert(math.isclose(inertia_point0(X0), .5))
         assert(math.isclose(inertia_point0(X0, p0), 1.))
         assert(math.isclose(inertia_axe0(X0, v), .5))
         assert(math.isclose(inertia_axe0(X0, v, p1), 2.5))
```

20 Calculer l'inertie de X par rapport à un point quelconque.

```
In [25]: p0 = np.array([2, 3])
         inertia_point(X, p0)

Out [25]: 15.744409410962476
```

21 Vérifier la première formule de Huygens en recalculant l'inertie d'une autre manière.

```
In  [26]: g = np.mean(X, axis=0)
          d = p0 - g

          mu = sum(np.diag(sample_metric))
          inertia_point(X) + mu * feat_inner(d, d)
Out [26]: 15.744409410962488
```

22 Calculer l'inertie de `X` par rapport à un axe quelconque.

```
In  [27]: p0 = np.array([2, 3])
          v = np.array([2, 1])
          inertia_axe(X, v, p0)
Out [27]: 4.719800062315761
```

23 Vérifier la formule de Huygens de l'inertie par rapport à une droite.

```
In  [28]: g = np.mean(X, axis=0)
          mu = sum(np.diag(sample_metric))
          vn = v / math.sqrt(feat_inner(v, v))
          k = feat_inner(p0 - g, vn)
          n = (p0 - g) - k * vn

          inertia_axe(X, v, g) + mu * feat_inner(n, n)
Out [28]: 4.719800062315765
```

⇒ pas compris :(