T.P. VI - Matrices & Statistiques

Code Capytale : `2c2d-893913`

I - Calculs sur des vecteurs

Le module `numpy` contient les fonctions suivantes qui prennent en argument un vecteur $X = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix}$:

* `sum` : renvoie la somme des éléments.
* `min` : renvoie l'élément minimal.
* `max` : renvoie l'élément maximal.
* `median` : renvoie la valeur médiane `med` (le nombre de valeurs inférieures à `med` est égal au nombre de valeurs qui lui sont inférieures).
* `mean` : renvoie la moyenne des éléments : $m = \frac{x_1+\cdots+x_n}{n}$.
* `var` : renvoie la variance des éléments : $\frac{(x_1-m)^2+\cdots+(x_n-m)^2}{n}$.
* `std` : renvoie l'écart-type des éléments : $\sqrt{\frac{(x_1-m)^2+\cdots+(x_n-m)^2}{n}}$.
* `cumsum` : renvoie la somme cumulée des éléments : $\begin{pmatrix} x_1 & (x_1+x_2) & (x_1+x_2+x_3) & \cdots & (x_1+\cdots+x_n) \end{pmatrix}$.

Exercice 1. Les notes d'un étudiant sont stockées dans un vecteur `notes` qui contient les valeurs $\begin{pmatrix} 18 & 12 & 15 & 10 & 14 \end{pmatrix}$. Modifier le code suivant pour qu'il affiche la moyenne et l'écart-type des notes de cet étudiant.

```
import numpy as np

notes = np.array(...)

moyenne = ...(notes)
print("moyenne", moyenne)

ecarttype = ...(notes)
print("écart-type", ecarttype)
```

Exercice 2. On définit la suite $(u_n)_{n\in\mathbb{N}}$ par

$$u_0 = 1 \text{ et, pour tout } n \geqslant 1,\ u_n = \frac{3 \times u_{n-1}}{n}.$$

1. Modifier la fonction suivante pour qu'elle prenne en argument un entier `n` et renvoie la valeur de u_n.

```
import numpy as np

def u(n):
    u = ...
    for i in range(1, ...):
        u = u * ...
    return u
```

2. À l'aide de la fonction précédente, afficher les valeurs de u_5 et u_{20}.

```
print("u5", ...)
print("u20", ...)
```

On s'intéresse maintenant à la suite $(s_n)_{n\in\mathbb{N}}$ définie par

$$\forall n \geqslant 0,\ s_n = \sum_{k=0}^{n} u_k.$$

3. Modifier le code suivant pour que la variable `S` contienne le vecteur $\begin{pmatrix} s_0 & \cdots & s_{20} \end{pmatrix}$.

```
U = np.ones((21, 1))
for i in range(1, ...):
    U[i] = ...
S = np.cumsum(...)
```

4. Modifier le code suivant pour qu'il représente graphiquement les termes de la suite $(S_n)_{0\leqslant n\leqslant 20}$. Que pouvez-vous conjecturer ?

```python
import matplotlib.pyplot as plt
plt.figure()
X = np.arange(0, 21, 1)
plt.plot(..., ...., 'o')
...
```

II - Statistiques bivariées

La fonction `np.shape(M)` renvoie la dimension de la matrice `M`.
Les fonctions vues sur les vecteurs peuvent être appliquées sur des matrices avec la modification suivante. Nous prenons l'exemple de la fonction `sum` mais ceci est valable pour toutes les fonctions :

- `sum(matrice)` : renvoie la somme de tous les éléments.
- `sum(matrice, 0)` : renvoie la somme des éléments par colonne.
- `sum(matrice, 1)` : renvoie la somme des éléments par ligne.

Exercice 3. Évaluer le code suivant pour bien comprendre le fonctionnement de ces fonctions.

```python
M = np.array([[12, 10, 4], [1, 18, 20]])

(n, p) = np.shape(M)
print("Nombre de lignes", n)
print("Nombre de colonnes", p)

print("matrice", M)

print("somme tous les éléments", np.sum(M))

print("somme les éléments par colonne", np.sum(M, 0))

print("somme les éléments par ligne", np.sum(M, 1))
```

Exercice 4. (Régression linéaire) Dans la matrice `M` suivante, nous avons enregistré les caractéritiques mesurées sur différents arbres. Chaque ligne de la matrice correspond à un arbre différent. À la ligne `i`, la quantité `M[i,0]` est la circonférence de l'arbre `i`, `M[i,1]` est sa hauteur et `M[i,2]` est son volume.

```python
import numpy as np
import matplotlib.pyplot as plt

M = np.array([
[8.3, 70, 10.3], [8.6, 65, 10.3],
[8.8, 63, 10.2], [10.5, 72, 16.4],
[10.7, 81, 18.8], [10.8, 83, 19.7],
[11, 66, 15.6], [11, 75, 18.2],
[11.1, 80, 22.6], [11.2, 75, 19.9],
[11.3, 79, 24.2], [11.4, 76, 21],
[11.4, 76, 21.4], [11.7, 69, 21.3],
[12, 75, 19.1], [12.9, 74, 22.2],
[12.9, 85, 33.8], [13.3, 86, 27.4],
[13.7, 71, 25.7], [13.8, 64, 24.9],
[14, 78, 34.5], [14.2, 80, 31.7],
[14.5, 74, 36.3], [16, 72, 38.3],
[16.3, 77, 42.6], [17.3, 81, 55.4],
[17.5, 82, 55.7], [17.9, 80, 58.3],
[18, 80, 51.5], [18, 80, 51],
[20.6, 87, 77] ])
```

1. À l'aide d'une fonction Python, déterminer le nombre d'arbres étudiés.

```python
n, p = ...(M)
print("Nombre d'arbres : ", n)
```

2. La fonction `plt.scatter(x, y)` trace le nuage des points dont les abscisses sont stockées dans le vecteur `x` et les ordonnées dans le vecteur `y`. Tracer le volume des arbres en fonction de leur circonférence.

```python
plt.figure()
plt.scatter(M[:,0], M[:,..])
...
```

3. À l'aide d'un unique appel de fonction, calculer les circonférence, hauteur et volume moyens.

```python
print("Caractéristiques moyennes", ...)
```

4. Modifier le code suivant pour qu'il affiche le point de coordonnées $(\overline{x}, \overline{y})$ où $\overline{x}$ est la circonférence moyenne et $\overline{y}$ est le volume moyen des arbres.

```
plt.figure()
plt.scatter(M[:,0], M[:,2])
plt.plot(np.mean(...), ..., "xr")
plt.show()
```

5. Dans le module `numpy`, la fonction `corrcoef(x, y)` renvoie la matrice $\begin{pmatrix} 1 & \rho(x,y) \\ \rho(y,x) & 1 \end{pmatrix}$. Calculer le coefficient de corrélation empirique entre la circonférence et le volume des arbres.

```
print("Coefficient de corrélation",\
    np.corrcoef(..., ...)[..., ...])
```

6. Dans le module `numpy`, la fonction `polyfit(X, Y, 1)` renvoie un couple `(a, b)` où `y = a x + b` est l'équation de la droite de régression linéaire entre `X` et `Y`. Modifier le code suivant pour calculer ces coefficients lorsque `X` est l'ensemble des circonférences et `Y` est l'ensemble des volumes ; puis tracer la droite correspondante.

```
(a, b) = np.polyfit(..., ..., 1)
T = np.arange(np.min(M[:,0])-1, np.max(M[:,0])+1)

def droite(x):
    return (a * x + b)

plt.figure()
plt.scatter(M[:,0], M[:,2])
plt.plot(np.mean(M[:,0]), np.mean(M[:,2]), "xr")
plt.plot(..., droite(...), 'g')
plt.show()
```