Chapitre 5 : Statistiques univariées

Dans tout ce chapitre, on utilisera : **import numpy as np**
import pandas as pd
import matplotlib.pyplot as plt

Vocabulaire des statistiques :
Soit Ω un ensemble appelé **population**. Ces éléments sont appelés **individus**.
Le cardinal de Ω est l'**effectif** de la population.
Un **caractère** est une application X définie sur Ω.
Lorsque X est à valeurs réelles, il s'agit d'un **caractère quantitatif**.
L'ensemble des images des éléments de Ω forme une **série statistique** représentée par un vecteur $X = [x_1, \ldots, x_n]$.

1. Acquisition des données

Les données statistiques sont souvent disponibles sous forme de fichiers CSV (Comma-Separate-Value).
Elles sont par exemple disponibles sur le site de l'INSEE ou sur le site data.gouv

Pour lire un fichier .csv en Python, on peut utiliser : **pd.read_csv**
Exemple : `df = pd.read_csv("Fichier.csv")` stocke dans `df` la table des données contenues dans `Fichier.csv`

La commande **df.head()** permet alors de visualiser les 5 premières lignes de `df`.
La fonction **df.shape** renvoie le nombre de lignes et de colonnes de `df`.
La fonction **df.count** renvoie le nombre de valeurs de chaque colonne numérique.

Dans la suite, on étudiera soit :
_ une série statistique stockée dans le vecteur `X`, qu'on étudiera avec la bibliothèque `np`.
_ un tableau statistique, stockée dans la table de données `df`, qu'on étudiera avec la bibliothèque `pd`.

2. Indicateurs de position

Moyenne empirique :

La moyenne de la série $[x_1, \ldots, x_n]$ est le réel : $\overline{\mathbf{x}} = \mathbf{\frac{1}{n} \sum_{i=1}^{n} x_i}$

En Python : **np.mean(X)** **df.mean()**

Remarque : En probabilité : si $X \longrightarrow U(\{x_1, \ldots, x_n\})$, alors $E(X) = \sum_{i=1}^{n} x_i P(X = x_i) = \sum_{i=1}^{n} x_i \times \frac{1}{n} = \frac{1}{n} \sum_{i=1}^{n} x_i$

On remarquera donc l'analogie entre une série statistique et une variable aléatoire qui suit une loi uniforme.

Minimum : min(X) ou **np.min(X)** **Maximum : max(X)** ou **np.max(X)**

Médiane :

Principe : La médiane d'une série est un nombre m qui sépare la série en deux parties égales : il y a autant d'éléments inférieurs que d'éléments supérieurs.

En pratique : si $[x_1, \ldots, x_n]$ est une série ordonnée :

_ si n est impair : $n = 2N+1$, on pose $m = x_{N+1}$

_ si n est pair : $n = 2N$, on pose $m = \frac{x_N + x_{N+1}}{2}$

En Python : **np.median(X)** ou **df.median()**

Ex :

X = (1, 3, 4, 6, 12)

Y = (5, 7, 2, 3, 6, 10)

3. Indicateurs de dispersion

Etendue : C'est l'écart entre le maximum et le minimum.
En Python : max(X) – min(X)

Définition : **Variance empirique / Ecart-type empirique :**

Si $x = (x_1, \ldots, x_n)$, on appelle variance de x le nombre : $\mathbf{V(x) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})^2}$

et écart-type de x le nombre : $\boldsymbol{\sigma(x) = \sqrt{V(x)}}$

Remarque : On trouve aussi comme notation s_x pour l'écart-type, et s_x^2 pour la variance.

Propriété : **Formule de Koenig**

Soit $x = (x_i)_{1 \le i \le n}$ une série statistique. Alors $V(x) = \frac{1}{n} \sum_{i=1}^{n} x_i^2 - (\overline{x})^2$

Remarque :
En probabilité, si $X \longrightarrow U(\{x_1, \ldots, x_n\})$, alors

$V(X) = E((X - E(X))^2 = \sum_{i=1}^{n} (x_i - E(X))^2 P(X = x_i) = \frac{1}{n} \sum_{i=1}^{n} (x_i - E(X))^2$

On remarquera aussi l'analogie avec la formule de Huygens : $V(X) = E(X^2) - E(X)^2$.

Propriété :

Soit $x = (x_i)_{1 \le i \le n}$ une série statistique de moyenne $\overline{x}$, de variance V(x) et d'écart-type σ_x.
Soit a et b deux réels.
Soit y la série statistique définie par : $\forall\, i \in \{1, \ldots, n\}$, $y_i = a.x_i + b$.
Alors : $\mathbf{\overline{y} = a.\overline{x} + b}$ $\quad \mathbf{V(y) = a^2 V(x)}$ $\quad \mathbf{\sigma_y = |a| \sigma_x}$

On remarquera encore une fois l'analogie avec les probabilités.

En Python :
Variance de x **: np.var(X)** ou **df.var()**
Ecart-type de x : **np.std(X)** ou **df.std()**

Quartiles :
_ le 1^{er} quartile, noté q_1 est la première valeur de la série pour laquelle au moins 25% des valeurs de la série sont inférieures ou égales à q_1.
_ le $2^{ème}$ quartile, noté q_2 est la médiane
_ le $3^{ème}$ quartile, noté q_3 est la première valeur de la série pour laquelle au moins 75% des valeurs de la série sont inférieures ou égales à q_3.

Pour n grand, on a donc environ 25% entre le minimum et q_1, 25% entre q_1 et la médiane, 25% entre la médiane et q_3, 25% entre q_3 et le maximum.

Quantiles :
_ pour $k \in \{1, \ldots, 9\}$, le $k^{ème}$ décile, noté d_k, est la première valeur de la série pour laquelle au moins k dixièmes des valeurs de la série sont inférieures ou égales à d_k.
_ pour $k \in \{1, \ldots, 99\}$, le $k^{ème}$ centile, noté c_k, est la première valeur de la série pour laquelle au moins k% des valeurs de la série sont inférieures ou égales à c_k.

Ecart inter-quartile : $q_3 - q_1$
Ecart inter-décile : $d_9 - d_1$
Ecart inter-centile : $c_{99} - c_1$

En pratique, ces écarts sont plus utiles que l'étendue, car ils sont moins dépendants de quelques valeurs anormales qui peuvent apparaitre dans la série statistique.

En Python : **df.describe()** renvoie le nombre d'éléments, la moyenne et l'écart-type, le minimum, le maximum et les 3 quartiles.

4. Etude de tableaux de données

Considérons une matrice A de valeurs numériques.
Les opérateurs np.sum, np.cumsum, np.min, np.max, np.mean, np.median, np.var, np.std, peuvent se commander ligne par ligne ou colonne par colonne, en remplaçant (A) par **(A,0) (résultats colonnes par colonnes)** ou par **(A,1) (résultats lignes par lignes)**.

Ex : Avec A = $\begin{pmatrix} 0 & 6 \\ 2 & 4 \end{pmatrix}$, c'est-à-dire A=np.array([[0,6],[2,4]])

np.mean(A,0) renvoie : (1 5) np.mean(A,1) renvoie : $\begin{pmatrix} 3 \\ 3 \end{pmatrix}$

Considérons un tableau de données à plusieurs colonnes, appelées 'Colonne1', … , 'ColonneP'.

Il est alors possible d'avoir les indicateurs d'une seule colonne, à l'aide de :

df['NomColonne'].describe()

On peut également commander par ordre croissante ou décroissante des valeurs d'une colonne :

df.sort_values('NomColonne',ascending==True)
df.sort_values('NomColonne',ascending==False)

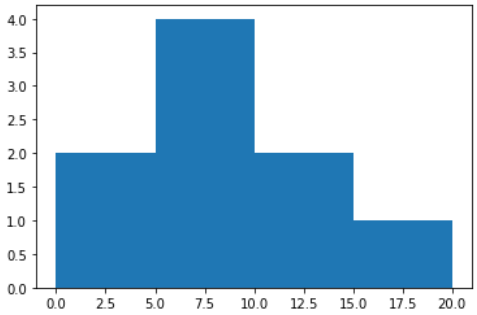
5. Histogramme et boite à moustache

Histogramme :
Soit $X = (x_1, \ldots, x_n)$ une série statistique.
_ si m est un entier, **plt.hist(X,m)** commande le tracé d'un histogramme représentant la série statistique x, constitué de m classes régulières, entre le minimum et le maximum. (par défaut, m vaut 10)
_ si $C = (c_1, \ldots, c_m)$ est un vecteur, **plt.hist(X, C)** commande le tracé de l'histogramme représentant la série statistique X, pour les classes $[c_1, c_2[$, $[c_2, c_3[\ldots[c_{m-1}, c_m]$.

Remarques :
_ L'intérêt de choisir ses classes est d'avoir des bornes entières, ou des classes de longueur différentes.
_ Attention, pour une représentation en histogramme, c'est **l'aire du rectangle** qui est proportionnelle à l'effectif (et non pas la hauteur !).
_ l'option density='True' permet d'avoir une surface totale égale à 1. (l'aire de chaque rectangle est égale à la fréquence de la classe)

Exemple :

```
X=[7,1,5,9,14,19,22,14,7,2]
Y=[0,5,10,15,20]
plt.hist(X, Y)
plt.show()
```



Boite à moustache (ou boite de Tukey) :
Elle représente une série statistique par :
_ la "boite" entre Q_1 et Q_3 (avec indication de la médiane)
_ les "moustaches" entre $Q_1 - 1.5(Q_3 - Q_1)$ et Q_1, et entre Q_3 et $Q_3 + 1{,}5(Q_3 - Q_1)$
(Ou entre d_1 et Q_1, et entre Q_3 et d_9, par exemple).
_ en dehors, les points restants sont marqués à part.

En Python : **plt.boxplot(X)**

6. Séries avec effectifs

Si la série statistique prend un nombre fini de valeurs, et prend plusieurs fois la même valeur, il est utile de la représenter sous la forme suivante :
$(x_1, \ldots, x_p)$ sont les valeurs prises (appelées **modalités**), et $(n_1, \ldots, n_p)$ les effectifs de chaque modalité.

Mode : c'est la modalité dont l'effectif est le plus grand.

Fréquence : si N est l'effectif global, $(N = n_1 + \ldots + n_p)$, alors la fréquence de la modalité x_i est $\mathbf{f_i = \frac{n_i}{N}}$.

Remarque :
Une fréquence est toujours comprise entre 0 et 1. Elle s'exprime souvent en %.
La somme des fréquences est égale à 1 (à comparer évidemment avec une probabilité)

Effectifs cumulés : Pour chaque x_i, on calcule l'effectif des résultats inférieurs ou égaux à x_i.

En Python, on utilise **np.cumsum** :
Si $X = [x_1, \ldots,, x_p]$, np.cumsum(X) $= [x_1, x_1 + x_2, \ldots, x_1 + \ldots + x_p]$

Fréquences cumulées : Pour chaque x_i, on calcule la fréquence des résultats inférieurs ou égaux à x_i., (à comparer à la fonction de répartition)

Si la série $(x_1, \ldots, x_p)$ est ordonnée dans l'ordre croissant :

Modalités	x_1	x_2	x_3	…	x_p
Effectifs	n_1	n_2	n_3	…	n_p
Effectifs cumulés	n_1	$n_1 + n_2$	$n_1 + n_2 + n_3$	…	N
Fréquences	f_1	f_2	f_3	…	f_p
Fréquences cumulées	f_1	$f_1 + f_2$	$f_1 + f_2 + f_3$	…	1

Exemple :
Considérons une table, de colonnes 'Valeurs' et 'Effectifs'.
On suppose que la table est ordonnée dans l'ordre croissant pour la colonne 'Valeurs'.
Par exemple : M = Valeurs Effectifs

1	2
2	1
3	2
4	1
5	4

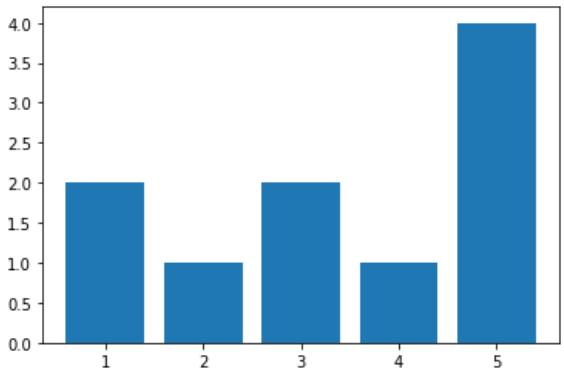
```
Eff=M['Effectifs'];print(Eff)
Effcum=np.cumsum(Eff);print(Effcum)
Freq=Eff/10;print(Freq)
Freqcum=np.cumsum(Freq);print(Freqcum)
```

renvoie :

effectifs =	effectifs cumulés =	fréquences =	fréqu cumulées =
2.	2.	0.2	0.2
1.	3.	0.1	0.3
2.	5.	0.2	0.5
1.	6.	0.1	0.6
4.	10.	0.4	1.

Diagramme en bâtons :
Si X = [x_1, …, x_p] et Y = [y_1, …., y_p] sont des vecteurs :
plt.bar(X, Y) trace le diagramme en barres des modalités x_i avec les effectifs (ou fréquences) y_i.

Exemple :

```
plt.bar(M['Valeurs'],M['Effectifs'])
plt.show()
```

renvoie :