Bibliothèques Python

La présente Fiche explique, à l'aide d'exemple, comment faire des statistiques descriptives à l'aide de Python. Vous Pouvez à tous moment copier, ou réécrire, le code pour le tester sur le compilateur disponible sur le site du Cours ou dans le volet Annexe.

Utilisation de Python
Calcul des Paramètres Statistiques Univariés

1. Bibliothèques Python pour les Statistiques

Une bibliothèque en Python est un ensemble de modules qui contiennent des fonctions, des classes et des variables prédéfinies. Ces modules sont des fichiers Python (.py) contenant du code réutilisable qui facilite la réalisation de tâches spécifiques sans avoir à les coder de zéro. Les bibliothèques sont **réutilisables** et **organisées**.

Une Dans cette première séance du Deuxième Bloc (portant sur les statistiques descriptives univariées), on a recours à ces bibliothèques :

Pandas

Pandas offre des structures de données et des outils de manipulation de données puissants et faciles à utiliser pour Python, elle simplifie la manipulation et l'analyse des données, rendant les opérations complexes sur les données plus intuitives et rapides. Pandas permet de créer des structures de données appelées DataFrame et Series, facilitant le calcul des paramètres statistiques.

```
import pandas as pd
data = [10, 20, 30, 40, 50]
series = pd.Series(data)
```

NumPy

NumPy est une bibliothèque fondamentale pour le calcul scientifique en Python, offrant un support pour les tableaux de grandes dimensions et des fonctions mathématiques. elle est performante pour les opérations mathématiques sur de grandes quantités de données et est utilisé pour effectuer des calculs statistiques efficaces.

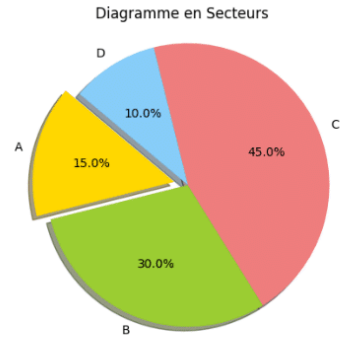
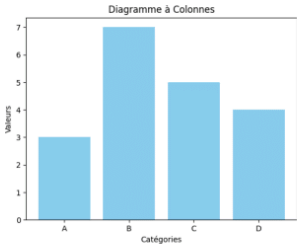
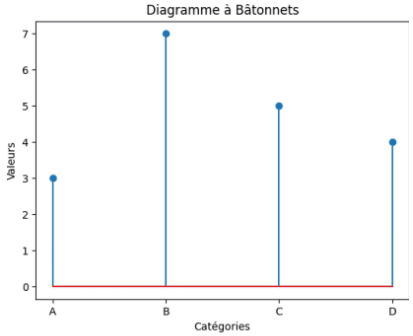
```
import numpy as np
data = np.array([10, 20, 30, 40, 50])
mean = np.mean(data)
```

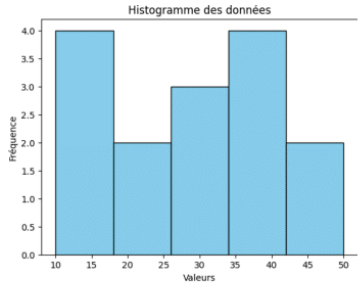
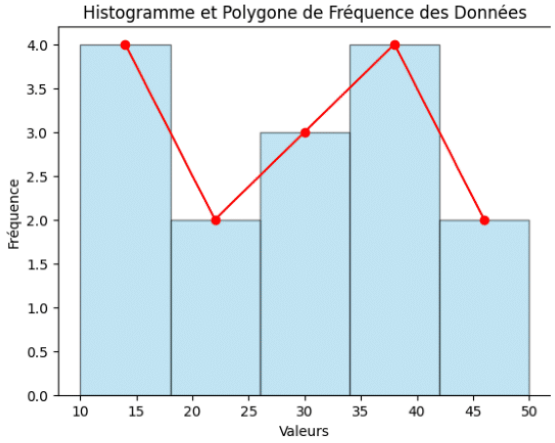
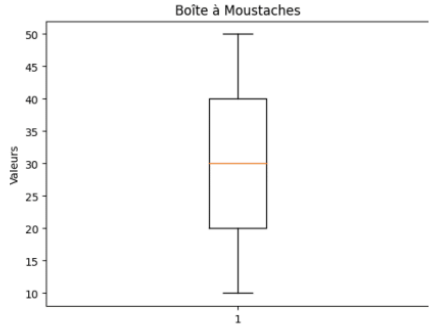
Matplotlib

Matplotlib est une bibliothèque de traçage pour Python qui permet de créer des graphiques statiques, animés et interactifs. Matplotlib est très flexible et largement utilisé pour la visualisation de données. Matplotlib est utilisé pour créer des graphiques des paramètres statistiques.

```
import matplotlib.pyplot as plt
plt.hist(data, color='skyblue', edgecolor='black')
plt.xlabel('Valeurs')
plt.ylabel('Fréquence')
plt.title('Histogramme des données')
plt.show()
```

2. Représentations Graphiques

Diagramme	Code Python	Résultat
Diagramme en Secteurs *Un graphique circulaire représentant les proportions de différentes catégories.*	import matplotlib.pyplot as plt labels = ['A', 'B', 'C', 'D'] sizes = [15, 30, 45, 10] colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue'] explode = (0.1, 0, 0, 0) plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140) plt.axis('equal') plt.title('Diagramme en Secteurs') plt.show()	
Diagramme à Colonnes *Un graphique représentant les valeurs par des colonnes verticales.*	import matplotlib.pyplot as plt categories = ['A', 'B', 'C', 'D'] values = [3, 7, 5, 4] plt.bar(categories, values, color='skyblue') plt.xlabel('Catégories') plt.ylabel('Valeurs') plt.title('Diagramme à Colonnes') plt.show()	
Diagramme à Bâtonnets *Un graphique représentant les valeurs par des bâtonnets.*	import matplotlib.pyplot as plt categories = ['A', 'B', 'C', 'D'] values = [3, 7, 5, 4] plt.stem(categories, values, use_line_collection=True) plt.xlabel('Catégories') plt.ylabel('Valeurs') plt.title('Diagramme à Bâtonnets') plt.show()	

Histogramme *Un graphique montrant la distribution des données à travers des intervalles.*	import matplotlib.pyplot as plt data = [10, 20, 20, 30, 30, 30, 40, 40, 40, 40, 10, 10, 10, 50, 50] plt.hist(data, bins=5, color='skyblue', edgecolor='black') plt.xlabel('Valeurs') plt.ylabel('Fréquence') plt.title('Histogramme des données') plt.show()	
Polygone de Fréquences *Un graphique montrant la fréquence des données à travers leurs quartiles.*	import matplotlib.pyplot as plt import numpy as np data = [10, 20, 20, 30, 30, 30, 40, 40, 40, 40, 10, 10, 10, 50, 50] counts, bin_edges = np.histogram(data, bins=5) bin_centers = (bin_edges[:-1] + bin_edges[1:]) / 2 plt.hist(data, bins=5, color='skyblue', edgecolor='black', alpha=0.5) plt.plot(bin_centers, counts, marker='o', color='red') plt.xlabel('Valeurs') plt.ylabel('Fréquence') plt.title('Histogramme et Polygone de Fréquence des Données') plt.show()	
Boîte à Moustaches (Boxplot) *Un graphique montrant la répartition des données à travers leurs quartiles.*	import matplotlib.pyplot as plt data = [10, 20, 30, 40, 50] plt.boxplot(data) plt.ylabel('Valeurs') plt.title('Boîte à Moustaches') plt.show()	

3. Paramètres Statistiques

Indicateur	Formule	Code dans Python	Exemple
Mode *La valeur la plus fréquente dans l'ensemble de données.*	Rechercher la valeur la plus fréquente.	import pandas as pd data = [10, 20, 20, 30, 30, 30, 40, 50] series = pd.Series(data) mode = series.mode()[0] print(mode)	Données: [10, 20, 20, 30, 30, 30, 40, 50] Mode: 30
Médiane *La valeur qui sépare la moitié supérieure et inférieure des données.*	Rechercher la valeur centrale après tri.	median = series.median() print(median)	Données: [10, 20, 20, 30, 30, 30, 40, 50] Médiane: 30
Moyenne Arithmétique *La somme des valeurs divisée par le nombre de valeurs.*	\(\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \)	mean = series.mean() print(mean)	Données: [10, 20, 20, 30, 30, 30, 40, 50] Moyenne: 28.75
Étendue *La différence entre la valeur maximale et minimale.*	\(\text{Étendue} = \max(x_i) - \min(x_i) \)	range_ = series.max() - series.min() print(range_)	Données: [10, 20, 20, 30, 30, 30, 40, 50] Étendue: 40
Variance *La mesure de dispersion des valeurs par rapport à la moyenne.*	\(\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n} \)	variance = series.var() print(variance)	Données: [10, 20, 20, 30, 30, 30, 40, 50] Variance: 164.64285714285714
Écart-Type *La racine carrée de la variance.*	\(\sigma = \sqrt{\sigma^2} \)	std_dev = series.std() print(std_dev)	Données: [10, 20, 20, 30, 30, 30, 40, 50] Écart-Type: 12.828172463606714
Quartiles *Les quartiles divisent les données en quatre parties égales.*	Calculer les 25e, 50e et 75e percentiles.	quartiles = series.quantile([0.25, 0.5, 0.75]) print(quartiles)	Données: [10, 20, 20, 30, 30, 30, 40, 50] Quartiles: 25.0, 30.0, 35.0
Déciles *Les déciles divisent les données en dix parties égales.*	Calculer les 10e, 20e, ..., 90e percentiles.	deciles = series.quantile([0.1 * i for i in range(1, 10)]) print(deciles)	Données: [10, 20, 20, 30, 30, 30, 40, 50] Déciles: 13.0, 20.0, 20.0, 30.0, 30.0, 30.0, 33.0, 40.0, 46.0
Centiles *Les centiles divisent les données en cent parties égales.*	Calculer les 1er, 2e, ..., 99e percentiles.	centiles = series.quantile([0.01 * i for i in range(1, 100)]) print(centiles)	Données: [10, 20, 20, 30, 30, 30, 40, 50] Centiles: [11.4, 12.8, 14.2, 15.6, ..., 45.4, 46.8, 48.2]