Dans ce TP on découvrira les commandes principales d'analyse statistique que l'on trouve dans la bibliothèque **numpy** et les outils graphiques de matplotlib.

Calculs avec Numpy :

Tout d'abord importe le module **numpy** avec

```
import numpy as np
```

Les fonctions **np.min()** et **np.max()** renvoient le minimum et le maximum des éléments du tableau donné dans un axe qui peut être spécifié.

Exercice 1. Recopier et exécuter les scripts suivants

```
import numpy as np
tableau = np.array([[1,3,5],[7,9,0],[1,2,8]])
print('Notre tableau est le suivant :')
print(tableau)
print( 'Application de la fonction min():' )
print ("le min est :", np.min(tableau))
print( 'Application de la fonction max():')
print ("le max est :", np.max(tableau))
print( 'Application de la fonction min(): ' )
print(np.min(tableau,1))
print( 'Application de la fonction min(): ' )
print(np.min(tableau,0))
```

À retenir

Si **tableau** est un tableau de données déclaré avec la fonction **np.array**

- ⋆ **np.min** permet d'afficher le minimum des valeurs du tableau et **np.max** le maximum.
- ⋆ Lorsqu'on rajoute l'argument 0 ou 1, dans la commande **np.min** (ou **np.max**), sont retournés les minimums (ou les maximums) selon les lignes ou selon les colonnes sous forme d'un tableau.

La fonction **np.median()** renvoie la valeur médiane des éléments du tableau donné.

Exercice 2. Recopier et exécuter les scripts suivants

```
import numpy as np
tableau = np.array([[10,25,74], [30,45,12], [53,20,63]])
print('Notre tableau est le suivant :')
print(tableau)
print('Application de la fonction median():')
print(np.median(tableau))
print('Application de la fonction median() sur laxe 0:')
print (np.median(tableau,0))
print('Application de la fonction median() sur laxe 1:')
print( np.median(tableau,1))
```

À retenir

Si **tableau** est un tableau de données déclaré avec la fonction **np.array**

- ⋆ **np.median** permet d'afficher la valeur médiane des valeurs du tableau.
- ⋆ Lorsqu'on rajoute l'argument 0 ou 1, dans la commande **np.median** sont retournées les valeurs médianes selon les lignes ou selon les colonnes sous forme d'un tableau.

La fonction **np.mean()** renvoie la valeur moyenne des éléments du tableau donné.

Exercice 3. Recopier et exécuter les scripts suivants

```
import numpy as np
tableau = np.array([[10,25,74], [30,45,12], [53,20,63]])
print('Notre tableau est le suivant :')
print(tableau)
print('Application de la fonction mean():')
print(np.mean(tableau))
print('Application de la fonction mean() sur laxe 0:')
print (np.mean(tableau,0))
print('Application de la fonction mean() sur laxe 1:')
print( np.mean(tableau,1))
```

À retenir

Si **tableau** est un tableau de données déclaré avec la fonction **np.array**

- ⋆ **np.mean** permet d'afficher la valeur moyenne des valeurs du tableau.
- ⋆ Lorsqu'on rajoute l'argument 0 ou 1, dans la commande **np.mean** sont retournées les valeurs médianes selon les lignes ou selon les colonnes sous forme d'un tableau.

La bibliothèque **numpy** permet aussi de calculer la variance et l'écart type d'une série de données déclarées dans un tableau. La variance et l'écart type peuvent aussi être calculés selon un axe.

À retenir

Les commandes **np.var** et **np.std** calculent variance et écart type.

Exercice 4. Recopier et exécuter les scripts suivants

```
import numpy as np
tableau = np.array([[1,2,3], [4,5,6],[7,8,9]])
print('Notre tableau est le suivant :')
print(tableau)
print('Application de la fonction np.cumsum():')
print(np.cumsum(tableau))
```

À retenir

La commande **cumsum** calcul les sommes cumulées des valeurs du tableau. On peut aussi spécifier un axe.

```
import numpy as np
tableau = np.array([1,2,3,4,5,6,7,8,9])
print('Notre tableau est le suivant :')
print(tableau)
print('Application de la fonction np.quantile():')
print('la mediane est :',np.quantile(tableau, 0.5))
print('les quartiles sont:',np.quantile(tableau, [0.25, 0.5, 0.75]))
```

Exercice 5. Calculer la moyenne, la médiane, le maximum, le minimum, la variance et l'écart type de la série : 17, 18, 19, 22, 23, 20, 37, 29, 59, 33

Exercice 6. Calculer la moyenne, la médiane, le maximum, le minimum, la variance et l'écart type de la série des nombres pairs allant de 0 à 1000.

Il arrive que les séries statistiques soit regroupées dans un tableau valeurs/effectifs.

Exercice 7. On considère la série de valeurs : 1,2,2,2,3,3,3,3,4,5,5,5,6,7,7,7. On souhaite la regrouper dans un tableau valeurs/effectifs.

1. Créer la variable **serie** contenant les valeurs de la série avec la commande **np.array**
2. Compléter le script suivant afin qu'il crée une fonction qui renvoie un tableau valeurs effectifs.

```
def tableau_valeurs_effectifs(serie):
    n=len(serie)
    serie.sort()
    Valeurs=[]
    Effectifs=[]
    i=0
    while i<n:
        j=1
        x=serie[i]
        while i+j<n and serie[i+j]==x:
            j=...
        Valeurs=...
        Effectifs=...
        i=i+j
    return(np.array([Valeurs,Effectifs]))
```

3. Tester la fonction.

Exercice 8. On considère la série suivante :

Valeurs	1	2	3	4	5
Effectifs	2	4	1	3	6

1. Créer des scripts permettant de calculer la moyenne, la variance et l'écart type
2. Créer des scripts permettant de déterminer la médiane et les quartiles Q_1 et Q_3.

Exercice 9. On considère la série suivante des nombres pairs allant de 0 à 1000 sans faire appel aux fonctions de la bibliothèque **Numpy**.

1. Créer des scripts permettant de calculer la moyenne, la variance et l'écart type
2. Créer des scripts permettant de déterminer la médiane et les quartiles Q_1 et Q_3.

Représentations graphiques avec mathplotlib :

Tout d'abord importe le module **matplotlib.pyplot** avec

```
import matplotlib.pyplot as plt
```

On peut construire un histogramme simple à partir d'une série de valeurs.

Exercice 10. Créer le script :

```
import matplotlib.pyplot as plt
x =[1,2,2,3,4,4,4,4,4,5,5]
plt.hist(x,range=(0,5),bins=5,color='yellow',edgecolor='red')
plt.xlabel( ' valeurs ')
plt.ylabel( ' nombres ')
plt.title( 'Exemple d \'histogramme simple ')
plt.show()
```

Mais il est possible de représenter deux séries sur un même graphique, en vue de les comparer

Exercice 11. Créer le script :

```
import matplotlib.pyplot as plt
x1 = [1, 2, 2, 3, 4, 4, 4, 4, 4, 5, 5]
x2 = [1, 1, 1, 2, 2, 3, 3, 3, 3, 4, 5, 5, 5]
```

```
bins = [x + 0.5 for x in range(0, 6)]
plt.hist([x1, x2], bins = bins, color = ['yellow', 'green'],
            edgecolor = 'red', hatch = '/', label = ['x1', 'x2'],
            histtype = 'bar') # bar est le defaut
plt.ylabel('valeurs')
plt.xlabel('nombres')
plt.title('2 series')
plt.legend()
```

La boîte à moustache ou diagramme en boîte a pour limite de la boîte le premier et le dernier quartile, la barre du milieu donne la médiane, les moustaches vont jusqu'à la valeur la plus extrême dans la limite de 1.5 fois la hauteur de la boîte, et les points au-delà des moustaches sont représentés par des points isolés.

Exercice 12. Créer le script :

```
import numpy as np
series=np.array([[1, 2, 3, 4, 5, 13], [6, 7, 8, 10, 10, 11, 12], [1, 2, 3]])
import matplotlib.pyplot as plt
plt.subplot(121)
plt.boxplot(series)
plt.ylim(0, 14)
plt.title('Diagramme en boîte')
```

Le diagramme à barre

Exercice 13. Créer le script :

```
countries = ['USA', 'Brazil', 'Russia', 'Spain', 'UK', 'India']
totalDeaths = [112596, 37312, 5971, 27136, 40597, 7449]
plt.bar(countries, totalDeaths)
plt.show()
```

Le diagramme circulaire est aussi disponible

Exercice 14. Créer le script :

```
x = np.array([15, 25, 30, 40])
label = ["France", "Germany", "Uk", "US"]
plt.pie(x, labels=label)
plt.show()
```

À retenir

Les commandes **plt.hist** , **plt.boxplot** , **plt.bar** et **plt.pie** permettent de construire des histogrammes, diagrammes en boîte, diagramme à barres et circulaire.

Le lien suivant donne quelques compléments sur cette bibliothèque :
`http://www.python-simple.com/python-matplotlib/matplotlib-intro.php`