**INTRODUCTION A LA BIBLIOTHEQUE MATPLOTLIB**

**Introduction**

Matplotlib est une bibliothèque Python open source permettant de créer des visualisations de données.

La [visualisation de données](https://datascientest.com/quest-ce-la-dataviz-le-point-avec-nos-data-experts) est une étape clé de l’analyse de données. Après avoir collecté, stocké et analysé des données, il est essentiel de transformer les résultats de ces analyses en rapports et en visualisations graphiques.  
  
Pour cause, le cerveau humain comprend plus rapidement un diagramme qu’une succession de statistiques sous forme de tableau. La  Datavisualisation ou encore DataViz permet donc de partager les résultats d’une analyse avec les équipes non-techniques d’une entreprise, et notamment ses dirigeants.

**1 - Définition**

Matplotlib est une bibliothèque open source du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques. Elle est souvent combinée avec les bibliothèques python de calcul scientifique NumPy et SciPy.

**2 - Historique**

**Matplotlib** est une bibliothèque Python open source, initialement développée par le neurobiologiste John Hunter en 2002. L’objectif était de visualiser les signaux électriques du cerveau de personnes épileptiques. Pour y parvenir, il souhaitait répliquer les fonctionnalités de création graphique de MATLAB avec Python.

Suite au décès de John Hunter en 2012, Matplotlib a été **améliorée au fil du temps par de nombreux contributeurs** de la communauté open source. Elle est utilisée pour créer des graphiques, et des diagrammes de haute qualité. C’est une alternative open source à MATLAB.

Il est par exemple possible de créer **des tracés, des histogrammes**, des diagrammes à barre et tous types de graphiques à l’aide de quelques lignes de code. Il s’agit d’un outil très complet, permettant de générer des visualisations de données très détaillées.

**3 - Utilités**

* **Création de Graphiques** : Matplotlib permet de créer une grande variété de graphiques, y compris des graphiques en ligne, des histogrammes, des diagrammes à barres, des nuages de points, des graphiques en secteurs, et bien plus encore.
* **Personnalisation des Graphiques** : Vous pouvez personnaliser presque tous les aspects de vos graphiques, y compris les couleurs, les styles de ligne, les polices, les annotations, et les légendes.
* **Visualisation Interactive** : Avec des outils comme matplotlib.pyplot, vous pouvez créer des graphiques interactifs qui permettent de zoomer, de faire défiler et de mettre en évidence des parties spécifiques des données.
* **Intégration avec d’autres Bibliothèques** : Matplotlib s’intègre bien avec d’autres bibliothèques Python comme NumPy, Pandas, et SciPy, ce qui facilite l’analyse et la visualisation des données.
* **Publication de Qualité** : Les graphiques créés avec Matplotlib sont de haute qualité et peuvent être utilisés dans des publications académiques, des rapports professionnels, et des présentations.
* **Support pour Divers Formats de Sortie** : Matplotlib permet d’enregistrer les graphiques dans plusieurs formats, y compris PNG, PDF, SVG, et EPS, ce qui est utile pour différentes applications et plateformes.
* **Animation** : Vous pouvez créer des animations pour visualiser des données dynamiques ou des simulations, ce qui est particulièrement utile dans les domaines scientifiques et d’ingénierie.

**4 – Le module pyplot de matplotlib**

Le module pyplot de matplotlib est l’un de ses principaux modules. Il regroupe un grand nombre de fonctions qui servent à créer des graphiques et les personnaliser (travailler sur les axes, le type de graphique, sa forme et même rajouter du texte)

A – Le traçage de courbe

Matplotlib est avant tout une librairie qui permet de tracer des fonctions et d’afficher leurs courbes dans des graphiques.

Les principales fonctions utilisées sont :

plot() : pour tracer une courbe. Documentation : [matplotlib.pyplot.plot — Matplotlib 3.9.2 documentation](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html).

legend() : pour afficher la légende,

figure() : pour afficher la figure,

show() : ,

xlabel() : pour nommer l’axe x,

ylabel() : pour nommer l’axe y.

exemples de code :

import matplotlib.pyplot as plt

import numpy as np

plt.figure(0)

x = np.arange(0,15,0.1)   # format (start,stop,step)

y = np.sin(x)

z = np.cos(x)

plt.plot(x,y,'r',x,z,'y') #Les paramètres 'r','y' indique la couleur de fonctions, les paramètres (x,y) et (x,z) sont les coordonnées des points à placer pour tracer la fonction

plt.xlabel('abcisse')

plt.ylabel('ordonnee')

plt.legend(['sin(x)', 'cos(x)'])

plt.show()

**B- Le traçage de graphique**

Une autre fonctionnalité de Matplotlib est la réalisation de graphiques en 2D, ceci étant très utile pour un Data Scientist dans l’étape de visualisation des données. En effet, il est possible d’afficher des histogrammes, des diagrammes circulaires, des diagrammes en moustache, des scatter plots, des stackplots (pour visualiser des données de manière empilées comme), etc.

Ces graphiques peuvent servir par exemple pour afficher la distribution des données, leurs indicateurs statistiques, leur tendance sur une certaine période si l’on traite des données temporelles, etc.

Les principales fonctions utilisées sont

* hist()pour tracer un histogramme. Documentation : [matplotlib.pyplot.hist — Matplotlib 3.9.2 documentation](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html).
* bar()pour tracer un graphique en barres. Documentation : [matplotlib.pyplot.bar — Matplotlib 3.9.2 documentation](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.bar.html).
* pie()pour tracer un graphique en camembert. Documentation : [matplotlib.pyplot.pie — Matplotlib 3.9.2 documentation](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.pie.html)
* scatter()pour tracer un nuage de points. Documentation : [matplotlib.pyplot.scatter — Matplotlib 3.9.2 documentation](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.scatter.html).

Exemple de code :

import matplotlib.pyplot as plt

names = ['group a', 'group b', 'group c']

values = [1, 10, 100]

plt.figure( figsize=(9, 3)) #Cette ligne crée une nouvelle figure avec une taille de 9 pouces de large et 3 pouces de haut.

plt.subplot(131) #plt.subplot(131) crée un sous-graphique dans une grille de 1 ligne par 3 colonnes, et le place dans la première position.

plt.bar(names, values) #plt.bar(names, values) crée un diagramme à barres avec les noms des groupes sur l’axe des x et les valeurs sur l’axe des y.

plt.subplot(132)

plt.scatter(names, values)

plt.subplot(133)

plt.plot(names, values)

plt.suptitle('Categorical Plotting') #plt.suptitle('Categorical Plotting') ajoute un titre global à la figure.

plt.show() #plt.show() affiche la figure avec les trois sous-graphiques.

Exemple avec un diagrammes circulaires :

import matplotlib.pyplot as plt

plt.figure(1)

plt.pie(x = [25285,14255,15225,12670,13905,16500], labels = ['Jan', 'Feb', 'Mar', 'April', 'May', 'June'],

           colors = ['#5830f3', '#7644e3', '#16369b', '#16e5cb', '#7ecff1','#c7d8f0'],

           autopct = lambda x: str(round(x, 2)) + '%')

plt.legend()

plt.show()

**C – Autre**

Matplotlib.pyplot peut également être utilisé pour interagir avec des images à travers des fonctions comme imshow(), hist(), etc.

**5 – Autres modules de matplotlib**

1. matplotlib.animation : Pour créer des animations.
2. matplotlib.axes : Pour manipuler les axes des graphiques.
3. matplotlib.figure : Pour gérer les figures (fenêtres de dessin).
4. matplotlib.backends : Pour gérer les différents backends (interfaces de rendu).
5. matplotlib.colors : Pour manipuler les couleurs.
6. matplotlib.dates : Pour travailler avec des données temporelles.
7. matplotlib.gridspec : Pour organiser les sous-graphiques dans une grille.
8. matplotlib.image : Pour manipuler les images.
9. matplotlib.patches : Pour dessiner des formes géométriques.
10. matplotlib.widgets : Pour ajouter des widgets interactifs aux graphiques.

**6 – Conclusion**

Matplotlib permet de créer et d’afficher des visualisations diverses et variées, il existe d’autres bibliothèques permettant de faire des visuels tout aussi sympas. On peut citer entre autres les bibliothèques Seaborn, Bokeh ou encore Ggplot.

La visualisation de données est au centre de toutes les problématiques abordées par les Data Scientists et les Data Analysts.