Course: Data Visualization

Ali ZAINOUL <ali.zainoul.az@gmail.com>

Crystal Clear Code
June 3, 2025

** Matplotlib **

Objectifs pédagogiques de la journée 1

- Installer un environnement Python dédié à la data visualisation
- Lire des fichiers `CSV` avec `pandas`
- Générer des visualisations simples avec `matplotlib`
- Se familiariser avec `JupyterLab`
- Comprendre les composants clés : `figure`, `axes`, `plot`

Installation de Python

- Télécharger la dernière version stable de Python :
  - `https://www.python.org/downloads/`
- Cocher **Add Python to PATH** lors de l'installation (Windows)
- Vérifier l'installation :

```
python --version
# ou selon l'OS
python3 --version
```

- Définir une variable d'environnement pour la version Python :

```
export PYTHON_VERSION=3.13
```

Prérequis techniques

- Avoir Python installé (`$PYTHON_VERSION` recommandé)
- Savoir utiliser un terminal ou une invite de commande
- Un éditeur de code (VSCode, JupyterLab, etc.)
- Connaissances de base en Python (variables, fonctions, listes)

Deux approches possibles

- **Approche 1 :** Installation complète avec **Anaconda**
  - Facile à mettre en place, convient aux débutants
- **Approche 2 :** Installation légère avec **Miniconda** ou **venv** + pip
  - Plus souple, recommandée pour les environnements professionnels

Installation via Anaconda (Windows, macOS, Linux)

- Télécharger Anaconda :
  `https://www.anaconda.com/products/distribution`
- Suivre les instructions selon votre système
- Ouvrir PowerShell (Windows) ou Terminal (macOS / Linux)
- Créer un environnement virtuel :

```
conda create -n dataviz python=$PYTHON_VERSION
conda activate dataviz
conda install matplotlib pandas jupyterlab
```

- Lancer l'environnement :

```
jupyter lab
```

Installation via Miniconda

- Télécharger Miniconda :
  `https://docs.conda.io/en/latest/miniconda.html`
- Installer selon votre système d'exploitation
- Vérifier l'installation :

```
conda --version
python --version
```

- Créer un environnement :

```
conda create -n dataviz python=$PYTHON_VERSION
conda activate dataviz
conda install matplotlib pandas jupyterlab
```

Utilisation de venv (tous systèmes)

- Créer un environnement virtuel :

```
python -m venv venv_dataviz
```

- Activer l'environnement :
  - macOS / Linux :

```
source venv_dataviz/bin/activate
```

  - Windows :

```
venv_dataviz\Scripts\activate.bat
```

- Installer les bibliothèques :

```
pip install --upgrade pip
pip install matplotlib pandas jupyterlab
```

Tester l'environnement de travail

- Lancer JupyterLab :

```
jupyter lab
```

- Créer un notebook Python et insérer ce code :

```
import pandas as pd
import matplotlib.pyplot as plt

print("Tout est prêt !")
```

Résumé : quelle méthode choisir ?

Méthode	Windows	macOS / Linux	Niveau recommandé
Anaconda	Oui	Oui	Débutant
Miniconda	Oui	Oui	Intermédiaire
venv + pip	Oui	Oui	Avancé

Éditeurs recommandés :

- Visual Studio Code (avec extension Python)
- JupyterLab (interface native)
- Jupyter Notebook (version classique)

Introduction à Matplotlib

- **Matplotlib** est une bibliothèque de visualisation très utilisée en Python.
- Elle permet de créer :
  - des courbes,
  - des histogrammes,
  - des nuages de points,
  - des barres, etc.
- L'interface `pyplot` (souvent importée sous le nom `plt`) est la plus utilisée.

Premier tracé avec `plot()`

- On commence par importer la bibliothèque :

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

- Puis on trace une courbe simple :

```
x = [1, 2, 3, 4]
y = [2, 4, 1, 3]
plt.plot(x, y)
plt.show()
```

- `plt.show()` est indispensable pour afficher la figure.

Personnaliser un graphique

- Matplotlib permet d'ajouter des titres, labels, couleurs :

```
plt.plot(x, y, color='red', linestyle='--', marker='o')
plt.title("Exemple de courbe")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

- On peut modifier :
  - la **couleur** (`color`),
  - le **style de ligne** (`linestyle`),
  - les **marqueurs** (`marker`).

Ajouter une légende

- Pour comparer plusieurs courbes, on utilise une légende :

```
plt.plot(x, y, label="Données A")
plt.plot(x, [i*2 for i in y], label="Données B")
plt.legend()
plt.title("Comparaison de courbes")
plt.show()
```

- `legend()` affiche une boîte avec les étiquettes données par `label`.

Tracer un graphique en barres

- Pour des données catégorielles, on utilise `bar()` :

```
jours = ["Lun", "Mar", "Mer", "Jeu", "Ven"]
temp = [14, 17, 15, 13, 16]
plt.bar(jours, temp, color='green')
plt.title("Températures de la semaine")
plt.show()
```

- On peut combiner `bar()` avec `xlabel()`, `ylabel()`, etc.

Plusieurs graphiques avec `subplot()`

- `subplot()` permet d'afficher plusieurs graphiques côte à côte :

```
plt.subplot(1, 2, 1)
plt.plot([1, 2, 3], [1, 4, 9])
plt.title("Courbe 1")

plt.subplot(1, 2, 2)
plt.plot([1, 2, 3], [9, 4, 1])
plt.title("Courbe 2")

plt.tight_layout()
plt.show()
```

- `subplot(1, 2, i)` = 1 ligne, 2 colonnes, i$^{?}$ graphique.

Mots-clés fondamentaux de Matplotlib

Mot-clé	Définition
`plt`	Alias de `matplotlib.pyplot`
`plot`	Trace une courbe linéaire simple
`show`	Affiche le graphique
`xlabel`	Nom de l'axe horizontal
`ylabel`	Nom de l'axe vertical
`title`	Titre du graphique
`figure`	Fenêtre de dessin globale
`axes`	Sous-zone de dessin dans une figure

TP 1 — Évolution des Notes avec Matplotlib

- Ouvrez un notebook Jupyter dans l'environnement virtuel.
- Créez un nouveau notebook Python.
- Importez la bibliothèque de visualisation :
  ```
  import matplotlib.pyplot as plt
  ```
- Créez une liste de notes, et de semaines :
  ```
  marks = [12, 14, 15, 13, 16]
  weeks = ["W1", "W2", "W3", "W4", "W5"]
  ```
- Tracez une courbe linéaire :
  ```
  plt.plot(weeks, marks)
  ```
- Ajoutez les étiquettes des axes et le titre :
  - `plt.xlabel("Weeks")`
  - `plt.ylabel("Marks")`
  - `plt.title("Évolution des notes")`
- Affichez le graphique : `plt.show()`

Explications

- **Importation:** `matplotlib.pyplot` est utilisée pour dessiner des graphiques.
- **Tracé:** `plt.plot()` trace une ligne reliant les points (x = semaines, y = notes).
- **Personnalisation:** Les étiquettes et le titre rendent le graphique plus lisible.
- **Affichage:** `plt.show()` permet de visualiser le résultat.

TP 2 — Analyse de chiffre d'affaires avec Pandas et Matplotlib

- Assurez-vous que le fichier `ventes.csv` est dans le dossier `data`.
- Lancez un nouveau notebook Jupyter.
- Importez Pandas et Matplotlib :
  - `import pandas as pd`
  - `import matplotlib.pyplot as plt`
- Lisez le fichier :

```
df = pd.read_csv("../../data/ventes.csv")
```

- Affichez les données :

```
df.head()
```

- Tracez une courbe avec les colonnes `mois` et `chiffre` :
  - `plt.plot(df['mois'], df['chiffre'])`
  - `plt.xlabel("Mois")`
  - `plt.ylabel("Chiffre d'affaires")`

Explications

- **Pandas:** Lit et structure les données du fichier CSV.
- **Matplotlib:** Visualise les données sous forme de courbe.
- **Titre et axes:** Fournissent un contexte au graphique.

TP 3 — Histogramme de températures journalières

- Assurez-vous que le fichier `temperatures.csv` est disponible.
- Démarrez un notebook Jupyter.
- Importez les bibliothèques nécessaires :
  - `import pandas as pd`
  - `import matplotlib.pyplot as plt`
- Lisez le fichier :

  ```
  df = pd.read_csv("../../data/temperatures.csv")
  ```
- Créez un histogramme :
  - `plt.bar(df['jour'], df['temperature'], color='green')`
  - `plt.xlabel("Jour")`
  - `plt.ylabel("Température (°C)")`
  - `plt.title("Températures hebdomadaires")`
  - `plt.xticks(rotation=45)`
  - `plt.show()`

Explications

- **plt.bar():** Représente les températures sous forme de barres.
- **Couleur:** Le paramètre `color='green'` personnalise l'apparence.
- **Rotation:** `xticks(rotation=45)` incline les noms de jours pour une meilleure lisibilité.

TP 4 — Comparaison de températures entre deux villes

- Assurez-vous d'avoir le fichier `villes.csv`.
- Créez un notebook Jupyter.
- Importez Pandas et Matplotlib :
  - `import pandas as pd`
  - `import matplotlib.pyplot as plt`
- Chargez le fichier :

```
df = pd.read_csv("../../data/villes.csv")
```

- Tracez les deux courbes :
  - `plt.plot(df['mois'], df['villeA'], label="Ville A")`
  - `plt.plot(df['mois'], df['villeB'], label="Ville B")`
  - `plt.xlabel("Mois")`
  - `plt.ylabel("Température (°C)")`
  - `plt.title("Températures mensuelles des deux villes")`
  - `plt.legend()`

Explications

- **Deux séries:** On trace deux courbes avec des couleurs différentes.
- **plt.legend():** Permet d'afficher une légende pour distinguer chaque ville.
- **Titre et axes:** Ajoutent du contexte à la comparaison visuelle.

TP 5 — Visualisation avec Matplotlib et Pandas

- Ouvrez Jupyter Notebook dans l'environnement virtuel.
- Créez un nouveau notebook.
- Importez Pandas: `import pandas as pd`.
- Importez Matplotlib: `import matplotlib.pyplot as plt`.
- Utilisez la commande pour lire le fichier CSV:
  `df = pd.read_csv("../../data/courses.csv")`
- Affichez les premières lignes du DataFrame:
  `df.head()`
- Set labels and title:
  - `plt.xlabel('Programming Language')`
  - `plt.ylabel('Days to Learn')`
  - `plt.title('Days to Learn Programming Languages')`
- Create and display a bar chart:
  - `plt.bar(df['Programming Language'], df['Learning Days'], color='blue')`

Explications

- **Import Libraries:**
  - `pandas` est importé pour la manipulation et l'analyse des données.
  - `matplotlib.pyplot` est importé pour créer des visualisations.
- **Read CSV File:**
  - `pd.read_csv()` lit le fichier CSV dans un DataFrame pandas.
- **Inspect Data:**
  - `head()` affiche les premières lignes du DataFrame pour inspecter les données.
- **Set Labels and Title:**
  - `xlabel()`, `ylabel()`, et `title()` définissent les étiquettes et le titre du graphique.
- **Create Bar Chart:**
  - `plt.bar()` crée un graphique à barres avec des données x et y spécifiées, et définit la couleur des barres à bleu.
- **Display the Plot:**
  - `plt.show()` affiche le graphique créé.

TP 6 - Visualisation de Données d'Étudiants - Étape 1

- Ouvrez Jupyter Notebook dans l'environnement virtuel.
- Créez un nouveau notebook.
- Importez Pandas: `import pandas as pd`.
- Importez Matplotlib: `import matplotlib.pyplot as plt`.
- Utilisez la commande pour lire le fichier CSV:
  `df = pd.read_csv("../../data/randomStudentData.csv")`
- Créez des sous-graphiques avec 1 ligne et 2 colonnes, ajustez la taille de la figure:
  `plt.figure(figsize=(12, 4))`
- Set the main title for the entire plot:
  `plt.suptitle('Student analytics')`

Visualisation de Données d'Étudiants avec Matplotlib et Pandas - Étape 2

- Create Subplot 1: Bar chart for "Marks":
  - `plt.subplot(1, 2, 1)`
  - `plt.bar(df['StudentID'], df['Marks'], color='blue')`
  - `plt.title('Student marks')`
  - `plt.xlabel('Student ids')`
  - `plt.ylabel('Student marks')`
  - `plt.xticks(rotation=45, ha='right')`

Visualisation de Données d'Étudiants avec Matplotlib et Pandas - Étape 3

- Create Subplot 2: Bar chart for "IQ":
  - `plt.subplot(1, 2, 2)`
  - `plt.bar(df['StudentID'], df['IQ'], color='orange')`
  - `plt.title('Student IQs')`
  - `plt.xlabel('Student ids')`
  - `plt.ylabel('Corresponding student IQ')`
  - `plt.xticks(rotation=45, ha='right')`
- Adjust subplot parameters for better layout:
  `plt.tight_layout()`
- Display the entire plot with subplots:
  `plt.show()`

Explications

- **Import Libraries:**
  - `pandas` est importé pour la manipulation et l'analyse des données.
  - `matplotlib.pyplot` est importé pour créer des visualisations.
- **Read CSV File:**
  - `pd.read_csv()` lit le fichier CSV dans un DataFrame pandas.
- **Create Subplots:**
  - `plt.figure(figsize=(12, 4))` crée une figure avec une taille spécifiée.
  - `plt.suptitle()` définit le titre principal pour l'ensemble du graphique.
  - `plt.subplot()` crée des sous-graphiques avec 1 ligne et 2 colonnes.

Explications - Suite

- **Create Bar Charts:**
  - `plt.bar()` crée des graphiques à barres avec des données spécifiées.
  - Les paramètres tels que la couleur, le titre, les étiquettes d'axe sont définis pour chaque sous-graphique.
- **Adjust Layout:**
  - `plt.tight_layout()` ajuste automatiquement les paramètres du sous-graphique pour une meilleure mise en page.
- **Display the Plot:**
  - `plt.show()` affiche l'ensemble du graphique avec les sous-graphiques.

TP 7 - Régression Linéaire: Marks vs IQ

- Ouvrez Jupyter Notebook dans l'environnement virtuel.
- Créez un nouveau notebook.
- Importez Pandas: `import pandas as pd`.
- Importez Matplotlib: `import matplotlib.pyplot as plt`.
- Importez la régression linéaire de Scikit-Learn: `from sklearn.linear_model import LinearRegression`.
- Utilisez la commande pour lire le fichier CSV:
  `df = pd.read_csv("../../data/randomStudentData.csv")`
- Extrait la variable indépendante (IQ) et la variable dépendante (Marks):
  - `X = df[['IQ']]`
  - `y = df['Marks']`

Régression Linéaire: Marks vs IQ (Suite)

- Créez un modèle de régression linéaire:
  `model = LinearRegression()`
- Adaptez le modèle aux données:
  `model.fit(X, y)`
- Faites des prédictions en utilisant le modèle:
  `y_pred = model.predict(X)`
- Plottez les points de données originaux:
  - `plt.scatter(X, y, color='blue', label='Original Data')`
- Plottez la ligne de régression:
  - `plt.plot(X, y_pred, color='red', linewidth=2, label='Linear Regression')`
- Set labels and title:
  - `plt.xlabel('IQ')`
  - `plt.ylabel('Marks')`
  - `plt.title('Linear Regression: Marks vs IQ')`

Explications

- **Import Libraries:**
  - `pandas` est importé pour la manipulation et l'analyse des données.
  - `matplotlib.pyplot` est importé pour créer des visualisations.
  - `LinearRegression` est importé de `sklearn.linear_model` pour la régression linéaire.
- **Read CSV File:**
  - `pd.read_csv()` lit le fichier CSV dans un DataFrame pandas.
- **Prepare Data:**
  - Les variables indépendantes (IQ) et dépendantes (Marks) sont extraites du DataFrame.
- **Create Linear Regression Model:**
  - `LinearRegression()` crée un modèle de régression linéaire.
- **Fit Model to Data:**
  - `fit(X, y)` ajuste le modèle aux données d'entraînement.

Explications - Suite

- **Make Predictions:**
  - `predict(X)` fait des prédictions en utilisant le modèle.
- **Plot Original Data Points:**
  - `plt.scatter()` crée un nuage de points pour les données originales.
- **Plot Regression Line:**
  - `plt.plot()` trace la ligne de régression à partir des prédictions.
- **Set Labels and Title:**
  - `xlabel()`, `ylabel()`, et `title()` définissent les étiquettes et le titre du graphique.
- **Show Legend:**
  - `plt.legend()` affiche la légende sur le graphique.
- **Display the Plot:**
  - `plt.show()` affiche le graphique résultant.

Lectures recommandées

- Documentation Pandas: `https://pandas.pydata.org`
- Documentation Matplotlib: `https://matplotlib.org/stable/index.html`

** Seaborn **

Introduction

- **Seaborn** est une bibliothèque Python basée sur **Matplotlib** qui facilite la création de visualisations statistiques attractives.
- Elle fournit une interface de haut niveau pour dessiner des graphiques informatifs à partir de données pandas.

Objectifs

- Simplifier la visualisation statistique des données
- Offrir des visualisations plus élégantes et automatiques que Matplotlib
- S'intégrer naturellement avec les DataFrames pandas

Comparaison rapide avec Matplotlib

- Matplotlib : plus bas niveau, très personnalisable
- Seaborn : plus haut niveau, plus automatique

Installation de Seaborn

Seaborn s'installe via `pip` : `pip install seaborn`

Importation des bibliothèques

- L'ordre compte:

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
```

Jeux de données intégrés

- Seaborn propose des jeux de données intégrés accessibles via `sns.load_dataset`.

```
df = sns.load_dataset("tips")
df.head()
```

Jeux de données courants

Quelques jeux courants :

- `"tips"` : pourboires dans un restaurant
- `"iris"` : fleurs d'iris
- `"penguins"` : caractéristiques de manchots
- `"titanic"` : passagers du Titanic

Histogrammes - Exemple

```
sns.histplot(data=df, x="total_bill", bins=20, kde=True)
plt.title("Distribution des additions")
plt.show()
```

Courbe de densité - Exemple

```
sns.kdeplot(data=df, x="total_bill", fill=True)
plt.title("Courbe de densité")
plt.show()
```

Boîtes à moustaches - Exemple

```
sns.boxplot(data=df, x="day", y="total_bill")
plt.title("Addition par jour")
plt.show()
```

Nuage de points - Exemple

```
sns.scatterplot(data=df, x="total_bill", y="tip", hue="sex")
plt.title("Pourboire en fonction de l'addition")
plt.show()
```

Régressions linéaires - Exemple

```
1 sns.lmplot(data=df, x="total_bill", y="tip", hue="sex")
```

Jointplot - Exemple

```
1 sns.jointplot(data=df, x="total_bill", y="tip", kind="hex")
```

Barplot - Exemple

```
1 sns.barplot(data=df, x="day", y="tip", ci="sd", hue="sex")
2 plt.title("Pourboires moyens par jour et par sexe")
3 plt.show()
```

Countplot - Exemple

```
sns.countplot(data=df, x="day", hue="sex")
plt.title("Nombre de repas par jour")
plt.show()
```

Violinplot - Exemple

```
sns.violinplot(data=df, x="day", y="total_bill", inner=None)
sns.stripplot(data=df, x="day", y="total_bill", color="k",
    alpha=0.3)
plt.title("Distribution par jour")
plt.show()
```

stripplot - Exemple

```
sns.violinplot(data=df, x="day", y="total_bill", inner=None)
sns.stripplot(data=df, x="day", y="total_bill", color="k",
    alpha=0.3)
plt.title("Distribution par jour")
plt.show()
```

Heatmaps et matrices de corrélation - Exemple

```
corr = df.corr(numeric_only=True)
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.title("Matrice de corrélation")
plt.show()
```

Clustermap - Exemple

```
sns.clustermap(corr, annot=True, cmap="vlag")
plt.title("Clustermap des corrélations")
plt.show()
```

Styles prédéfinis - Exemple

```
sns.set_style("whitegrid")  # autres : darkgrid, white, dark,
    ticks
```

Contextes d'affichage - Exemple

```
1 sns.set_context("talk")  # autres : paper, notebook, poster
```

Palettes de couleurs - Exemple

```
sns.set_palette("pastel")  # autres : deep, dark, muted,
    colorblind...
```

Interaction avec Matplotlib - Exemple

```
fig, ax = plt.subplots()
sns.histplot(data=df, x="total_bill", ax=ax)
ax.set_title("Titre avec Matplotlib")
plt.savefig("figure.png")
plt.show()
```

Analyse du jeu Titanic

```
df = sns.load_dataset("titanic")
sns.countplot(data=df, x="class", hue="survived")
plt.title("Survie par classe")
plt.show()
```

Analyse du jeu penguins

- **But** : Analyser le jeu de données `penguins`
- Charger le jeu de données
- Afficher les distributions des masses selon les espèces
- Étudier les corrélations entre caractéristiques

Résumé

- **Seaborn** est un outil puissant pour produire des visualisations statistiques élégantes en quelques lignes de code.
- Il est particulièrement adapté à l'analyse exploratoire de données, et s'intègre naturellement avec pandas et Matplotlib.
- Adapté à l'analyse exploratoire de données.
- S'intègre naturellement avec pandas.
- Idéal pour les visualisations rapides et informatives.
- Complémentaire à Matplotlib pour un contrôle fin.
- Moins adapté aux cas très personnalisés.

Conclusion

À retenir : Seaborn permet de gagner un temps précieux dans les phases d'analyse exploratoire tout en garantissant une qualité graphique élevée.

Lectures recommandées

- Documentation Pandas: `https://pandas.pydata.org`
- Documentation Scipy: `https://scipy.org/`
- Documentation Sklearn: `https://scikit-learn.org/stable/`
- Documentation Matplotlib: `https://matplotlib.org/stable/index.html`
- Documentation Seaborn: `https://seaborn.pydata.org/`