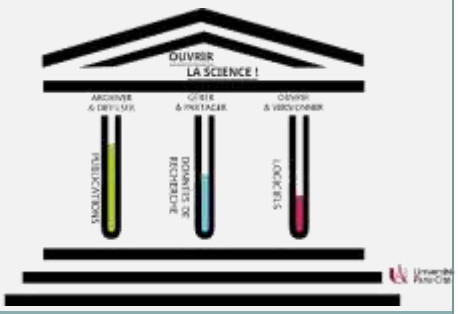
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Universitaire de Mila
Institut des Sciences de la Nature et la Vie
Département des science biologiques et agricoles
Master académique
Spécialité : Biochimie Appliquée

Intitulé de la matière : Logiciels libres et Open-Sources 2

CHAPITRE 02:

PROGRAMMATION AVANCÉES ET AUTOMATISATION



Préparé par M. Kamel Belmehboul

Année univ. 2025-2026

Introduction générale

Dans les sciences biologiques, on travaille souvent avec beaucoup de données :

– fichiers CSV.

– tableaux de mesures.

– résultats d'expériences.

– images, etc.

Le traitement manuel est lent et peut provoquer des erreurs.

C'est pourquoi nous utilisons :

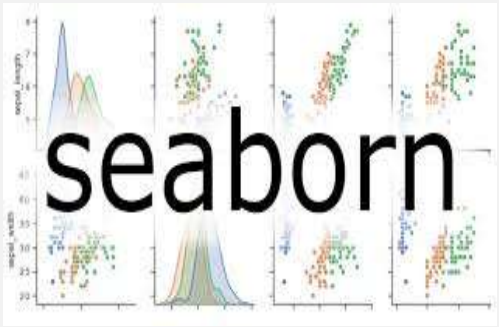
des scripts Bash pour automatiser les tâches répétitives .

Python avec NumPy, Pandas, Seaborn pour analyser et explorer les données .

des outils de visualisation avancée pour créer des graphiques et tableaux de bord.

L'objectif du cours

- L’objectif du cours est comprendre simplement comment ces outils aident les scientifiques à gagner du temps et à mieux analyser leurs résultats .



1. Scripts Bash avancés pour l'automatisation

Qu'est-ce que Bash ?

- **Bash** est un programme qui exécute des commandes Dans les systèmes Linux.
- Bash est un outil qui permet à l'utilisateur d'écrire une commande, de demander une action à l'ordinateur , et Bash exécute immédiatement cette action
- **Exemple 01:** Afficher la liste des fichiers

```
ls
```

Cette commande demande à Bash d'afficher les fichiers du dossier.

1. 1 des commandes Bash simples

- **Exemple 02:** Changer de dossier

  cd *Documents*

Permet d'entrer dans le dossier *Documents*.

- **Exemple 03:** Créer un fichier vide

  touch fichier.txt

- **Exemple 04:** Supprimer un fichier

  rm fichier.txt

- **Exemple 05:** Renommer ou déplacer un fichier

  mv fichier.txt ancien.txt

1. 2 Qu'est-ce qu'un Script ?

- **Un script =** un fichier texte avec plusieurs commandes**.**

Au lieu d'écrire manuellement les mêmes commandes chaque jour,

on peut les mettre dans un fichier et lancer ce fichier en une seule fois.

- **Script = une liste d'instructions écrite à l'avance**

Idée simple :
Un script = une feuille de consignes que vous donnez à votre assistant.

1. 2.1 Qu'est-ce qu'un Script Bash ?

- **Un Script Bash =** un script contenant des commandes Bash**.**

Il s'agit d'un fichier (souvent avec l'extension .sh)dans lequel on écrit des commandes **Bash**. Quand on lance le fichier, Bash exécute toutes les commandes dans l'ordre .

1.2.2 Exemple simplifié

- Exemple très simple de Script Bash:

```
#!/bin/bash

echo "Bonjour les étudiants"
echo "Liste des fichiers :"
ls
```

- Explication :

I. **echo** affiche un texte à l'écran .
II. **ls** liste les fichiers du dossier .
III. **Bash** exécute toutes ces lignes une par une.

1. 2.3 Relation entre Bash – Script – Script Bash

Concept	Définition	Rôle
Bash	Programme exécutant des commandes	Exécute vos instructions
Script	Fichier contenant plusieurs commandes	Automatisation
Script Bash	Script utilisant le langage **Bash**	Automatise des tâches

Image simple :

I. **Bash** = assistant.

II. **Script**= feuille d'instructions Script.

III. **Bash** = feuille écrite dans un langage que Bash comprend

1. 3 Pourquoi utiliser des scripts Bash ?

Pourquoi utiliser des scripts Bash ?

I. Gagner du temps.

II. Répéter une tâche facilement.

III. Éviter les erreurs manuelles.

IV. Lancer la même analyse sur plusieurs fichiers.

V. Automatiser le travail (automatisation).

1. 3.1 Comment un Bash Script permet l'automatisation

- Un **script Bash** permet l'automatisation parce qu'il exécute **une série de commandes automatiquement,** sans intervention humaine.

- **Voici le mécanisme :**

I. **Écriture des commandes une seule fois Dans un script Bash, on écrit toutes les étapes d'une tâche :**

1. 3.2 Comment un Bash Script permet l'automatisation

1. créer un dossier
2. copier des fichiers
3. analyser des données
4. supprimer des fichiers
5. répéter une action sur plusieurs éléments

Toutes ces commandes sont placées dans un fichier .sh

1. 3.3 Comment un Bash Script permet l'automatisation

II. Exécution automatique

Au lieu d'exécuter chaque commande manuellement, l'utilisateur lance

le script une seule fois

```bash
./mon_script.sh
```

Bash lit le fichier ligne par ligne et exécute toutes les commandes **dans l'ordre** sans pause.

1. 3.4 Comment un Bash Script permet l'automatisation

III. Répétition automatique (boucles)

Les scripts Bash utilisent des boucles comme :

bash

```
#!/bin/bash

for i in 1 2 3
do
    echo "Bonjour !"
done
```

Explication

- for i in 1 2 3 : la boucle va répéter l'action 3 fois.
- .echo "Bonjour !" : le script affiche un message.
- Le script exécute automatiquement les répétitions sans intervention manuelle → **c'est cela l'automatisation**

1. 4 Travail Pratique : Création d'un petit programme en Bash

- **Objectif :**

Créer un **script Bash** qui :

1. Demande le nom de l'utilisateur
2. Demande son âge
3. Affiche un message de bienvenue
4. Indique si l'utilisateur est majeur ou mineur

1. 4.1 Travail Pratique : Création d'un petit programme en Bash

- **Étape 1 :** Créer le fichier du script

Créer un fichier nommé :

pgsql

info.sh

1. 4.2 Travail Pratique : Création d'un petit programme en Bash

- **Étape 2 :** Écrire le code suivant dans le fichier

bash

```bash
#!/bin/bash

echo "Bienvenue dans le programme !"

# Demander le nom
read -p "Quel est votre nom ? " name

# Demander l'âge
read -p "Quel âge avez-vous ? " age

echo "--------------------------"
echo "Bonjour $name, vous avez $age ans."
```

1. 4.3 Travail Pratique : Création d'un petit programme en Bash


```bash
# Condition simple
if [ $age -ge 18 ]; then
    echo "Vous êtes majeur."
else
    echo "Vous êtes mineur."
fi

echo "Merci d'avoir utilisé ce programme !"
```

1. 4.4 Travail Pratique : Création d'un petit programme en Bash

- **Étape 3 :** Rendre le script exécutable

Dans le terminal, taper :

bash

```
chmod +x info.sh
```

- **Étape 4 :** Exécuter le script

bash

```
./info.sh
```

1. 4.5Travail Pratique : Création d'un petit programme en Bash

- **Exemple de résultat obtenu :**

markdown

```
Bienvenue dans le programme !
Quel est votre nom ? Kamel
Quel âge avez-vous ? 15
--------------------------
Bonjour Kamel, vous avez 15 ans.
Vous êtes mineur.
Merci d'avoir utilisé ce programme !
```

1. 4.6 Résumé du travail

Ce TP introduit la création d'un script **Bash** simple comprenant :

L'utilisation du **shebang**

L'affichage avec **echo**

La lecture d'entrées utilisateur avec **read**

L'utilisation de variables

La structure conditionnelle **if/else**

Les permissions d'exécution **chmod**

L'exécution d'un script dans le **terminal**

2. Utilisation de bibliothèques telles que NumPy , Pandas, Seabron pour explorer et modéliser des jeux de données .

I. Introduction aux bibliothèques:

- **NumPy :** C'est la bibliothèque de base pour les calculs numériques en Python. Elle permet de manipuler des tableaux (**arrays**) très efficacement, et de faire des opérations mathématiques très rapidement, même sur de grandes quantités de données.
- **Pandas :** Permet de structurer les données sous forme de « **DataFrame** », qui est un tableau à deux dimensions (**lignes et colonnes**). Grâce à **Pandas**, on peut lire des fichiers **CSV**, gérer les valeurs manquantes, trier, filtrer et agréger des données.
  Seaborn : Bibliothèque de visualisation statistique, construite au-dessus de **Matplotlib**, et conçue pour travailler très bien avec les DataFrame Pandas. Elle permet de faire des graphiques statistiques attrayants et informatifs (boîtes à moustaches, heatmap, nuages de points, etc.)

2. 1 Utilisation de bibliothèques telles que NumPy , Pandas, Seabron pour explorer et modéliser des jeux de données .

II. Pourquoi les utiliser pour explorer et modéliser des jeux de données ?

A. Grâce à **NumPy**, on peut manipuler des vecteurs, des matrices, et exécuter des calculs (moyenne, somme, écart-type…) très rapidement.

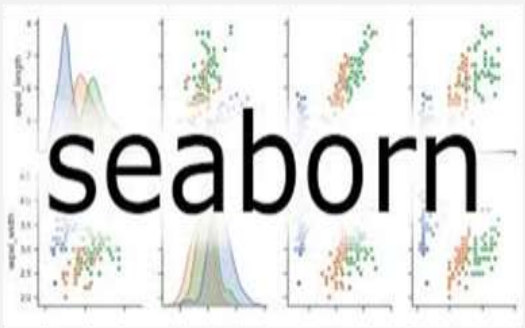
B. Avec **Pandas**, on peut importer un jeu de données réel (par exemple un fichier CSV), le nettoyer (par exemple supprimer ou remplir les valeurs manquantes), et le transformer (regrouper, filtrer, agréger).

C. Grâce à **Seaborn**, après avoir nettoyé les données, on peut visualiser des tendances statistiques, identifier des anomalies, des corrélations, ou des distributions

2. 2 Utilisation de bibliothèques telles que NumPy , Pandas, Seabron pour explorer et modéliser des jeux de données .

Ces trois bibliothèques ensemble fournissent un pipeline d'analyse très puissant :

- **NumPy** pour les calculs numériques.
- **Pandas** pour organiser et transformer les données.
- **Seaborn** pour visualiser les résultats.



2. 3 Exemples simples de code

- Voici quelques exemples de code **Python** montrant comment utiliser ces bibliothèques :

python

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Example data with NumPy
data = np.array([10, 20, 30, 40, 50])
mean_val = data.mean()
std_val = data.std()
print("Moyenne :", mean_val)
print("Écart type :", std_val)

# Example DataFrame with Pandas
df = pd.DataFrame({
    'Âge': [23, 45, 31, 22, 34],
    'Taille': [1.70, 1.85, 1.60, 1.75, 1.80]
})
    print(df)
    print(df.describe())
```

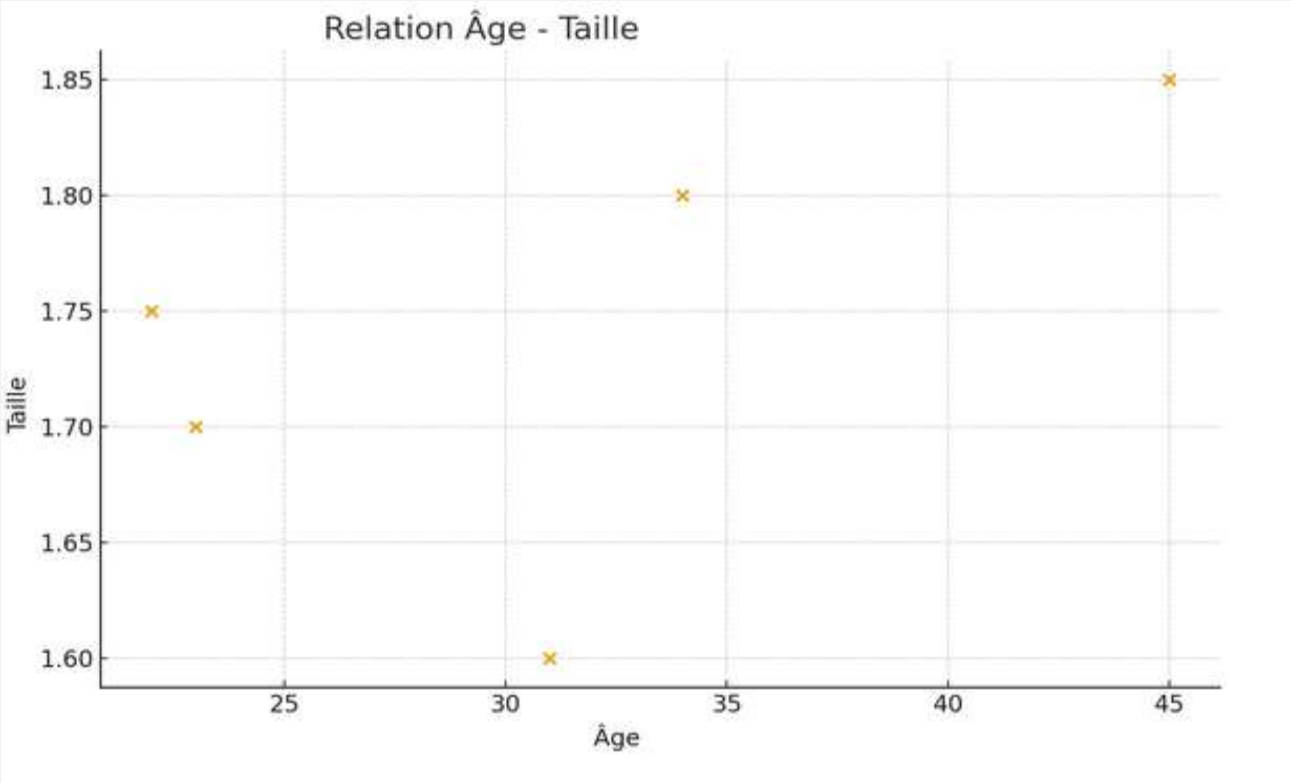
2. 4.1 Exemples simples de code


```
Moyenne : 30.0
Écart type : 14.142135623730951

DataFrame :
   Âge  Taille
0   23    1.70
1   45    1.85
2   31    1.60
3   22    1.75
4   34    1.80

Description :
              Âge    Taille
count    5.000000  5.000000
mean    31.000000  1.740000
std      9.354143  0.096177
min     22.000000  1.600000
25%     23.000000  1.700000
50%     31.000000  1.750000
75%     34.000000  1.800000
max     45.000000  1.850000
```

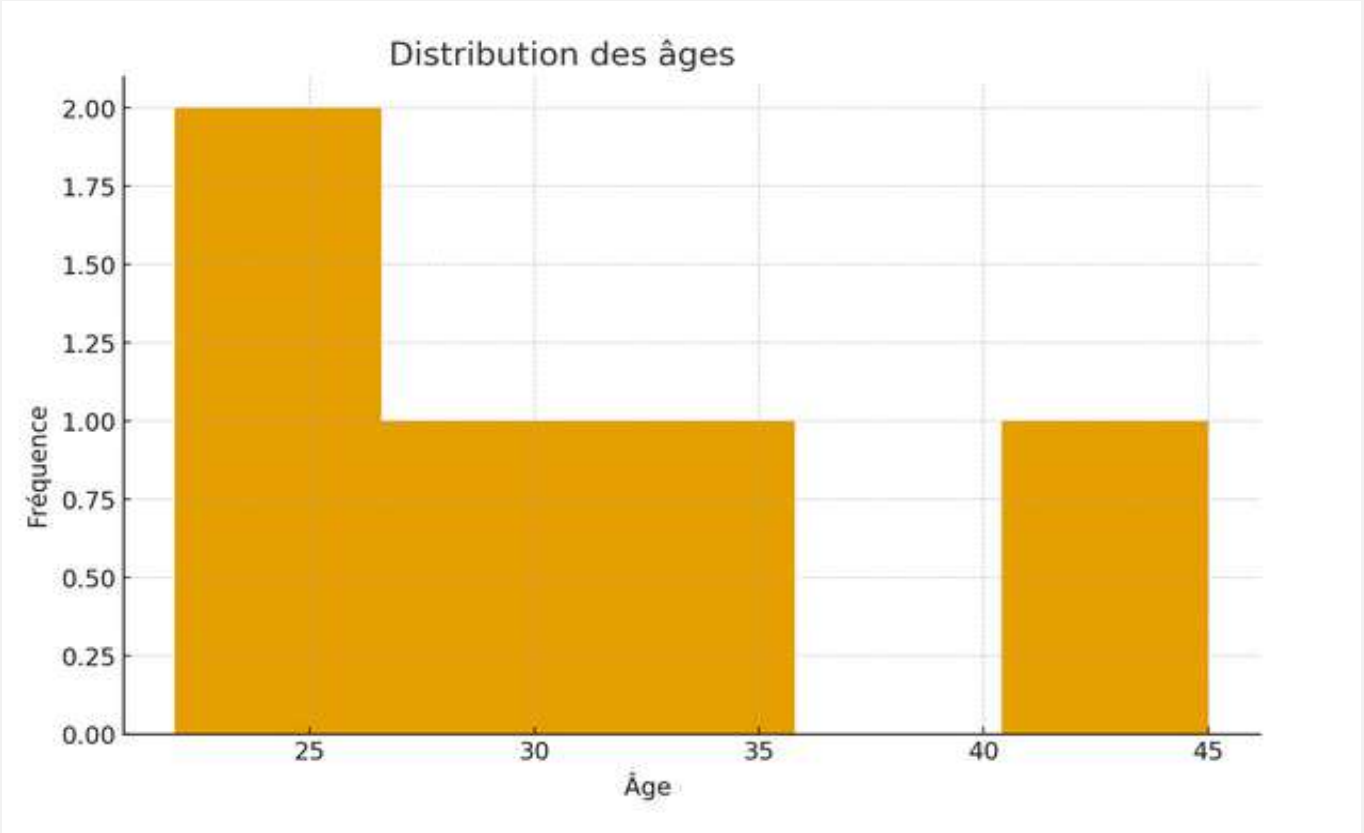
2. 4.2 Exemples simples de code


```
# رسم Scatter plot (علاقة Âge - Taille) باستخدام matplotlib
plt.figure()
plt.scatter(df['Âge'], df['Taille'])
plt.title("Relation Âge - Taille (العلاقة العمر - الطول)")
plt.xlabel("Âge (سنة)")
plt.ylabel("Taille (متر)")
plt.grid(True)
plt.show()

# رسم Histogram لتوزيع الأعمار باستخدام matplotlib
plt.figure()
plt.hist(df['Âge'], bins=5)
plt.title("Distribution des âges (توزيع الأعمار)")
plt.xlabel("Âge (سنة)")
plt.ylabel("Fréquence (التكرار)")
plt.grid(True)
plt.show()
```

2. 4.3 Exemples simples de code



2. 4.4 Exemples simples de code



3. Visualisation avancées des données

- La visualisation avancée consiste à aller au-delà des graphiques statiques traditionnels (comme les courbes, histogrammes, barres) pour produire des **visualisations interactives**, dynamiques, et intégrées dans des interfaces plus complètes appelées **tableaux de bord (dashboards).**
- Ces outils sont largement utilisés dans :

➢ L'analyse de données (Data Analysis).

➢ La Business Intelligence.

➢ Le pilotage de projets.

➢ Le suivi statistique en temps réel.

➢ L'apprentissage automatique (Machine Learning).

3. Visualisation avancées des données

- Les bibliothèques les plus populaires scientifiquement confirmées :

1. **Plotly** (Université de Washington, MIT, NASA l'utilisent).
2. **Dash** (développé par Plotly).
3. **Bokeh** (Université d'Oxford – Projet Anaconda).
4. **Altair** (Développé à l'Université de Washington).

3.1 Création de tableaux de bord interactifs (Dashboards)

- **Qu'est-ce qu'un tableau de bord interactif ?**

C'est une interface web qui rassemble plusieurs graphiques, indicateurs et filtres permettant à l'utilisateur de manipuler et explorer les données en temps réel.

- Un bon **Dashboard** permet:

1. Filtrer les données (année ‹région ‹catégorie).
2. Cliquer sur les éléments pour obtenir des informations supplémentaires.
3. Voir les données évoluer automatiquement.
4. Combiner plusieurs visualisations dans une même page.

3.1 Création de tableaux de bord interactifs (Dashboards)

- Les **Dashboard** sont essentiels dans les domaines suivants :

1. Analyse financière.
2. Suivi d'élèves (absences – notes – événements).
3. Gestion de projets urbains.
4. Suivi de performance industrielle.

3.1 Création de tableaux de bord interactifs (Dashboards)

- Bibliothèque recommandée : **Dash (by Plotly)**

C’est une bibliothèque Python qui permet de créer facilement des applications web interactives Sans avoir besoin de HTML ni de JavaScript

3.1 Création de tableaux de bord interactifs (Dashboards)

- **Exemple simple d'un tableau de bord Dash:**

python

```python
import dash
from dash import dcc, html
import plotly.express as px
import pandas as pd

# Exemple de données
df = pd.DataFrame({
    "Ville": ["Alger", "Oran", "Constantine"],
    "Population": [3500000, 2000000, 1000000]
})

fig = px.bar(df, x="Ville", y="Population", title="Population par ville")

# Création de l'application
app = dash.Dash(__name__)
```

3.1 Création de tableaux de bord interactifs (Dashboards)


```
app.layout = html.Div([
    html.H1("Dashboard démographique"),
    dcc.Graph(figure=fig)
])

if __name__ == "__main__":
    app.run_server(debug=True)
```

- Ce code crée automatiquement une application web accessible via le navigateur.
- L'utilisateur peut interagir avec le graphique (zoom, survol, sélection).

3.2 Création de graphiques interactifs

Les graphiques interactifs permettent :

- zoom / pan
- affichage d'informations au survol (hover)
- mise en surbrillance d'éléments
- filtres dynamiques

Bibliothèque utilisée scientifiquement **: Plotly Express**

3.2 Création de graphiques interactifs

- Exemple : **Scatter plot interactif**

python

```python
import plotly.express as px
import pandas as pd

df = pd.DataFrame({
    "Âge": [23, 45, 31, 26, 40],
    "Score": [88, 92, 75, 95, 85]
})

fig = px.scatter(df, x="Âge", y="Score", title="Âge vs Score")
fig.show()
```

3.2 Création de graphiques interactifs

- Exemple : **Histogramme interactif**

python

```
fig = px.histogram(df, x="Score", nbins=5, title="Distribution des scores")
fig.show()
```

Fin cours