Chapitre 2 — Analyse et Manipulation de Données Géologiques avec Python

---

2.1 Manipulation de données géologiques avec pandas

Objectifs

- Comprendre comment importer, structurer et nettoyer des données géologiques.
- Préparer les données pour l'analyse (statistiques, classification, tri...).
- Apprendre les bases de l'analyse temporelle et spatiale.

2.1.1 Chargement des données géologiques

Les données peuvent provenir de :

- Forages (profondeur, densité, lithologie...)
- Couches sédimentaires
- Fichiers CSV, Excel, TXT

Exemple de code

```
# تحميل البيانات الجيولوجية باستخدام pandas
import pandas as pd

# تعليق : تحميل بيانات الآبار من ملف CSV — قراءة ملف
forages = pd.read_csv("forages.csv")
print(forages.head())

# تعليق : تحميل طبقات جيولوجية من ملف Excel — قراءة ملف Excel
couches = pd.read_excel("couches_geologiques.xlsx")
```

2.1.2 Nettoyage des données

Les données géologiques contiennent souvent :

- valeurs manquantes
- unités non cohérentes
- erreurs de profondeur ou densité

Exemple de nettoyage

```
# التحقق من القيم المفقودة
forages.isnull().sum()

# تعويض القيم المفقودة بالمتوسط
forages['épaisseur_couche'] =
forages['épaisseur_couche'].fillna(forages['épaisseur_couche'].mean())

# حذف البيانات غير الصالحة
forages = forages.dropna()

# إزالة الأعماق السلبية
forages = forages[forages['profondeur'] >= 0]
```

2.1.3 Analyse descriptive

On peut analyser :

- profondeur moyenne
- classification des couches
- distribution des lithologies

```
# الحسابات الإحصائية
profondeur_moyenne = forages['profondeur'].mean()

# تصنيف حسب السمك
forages['type_couche'] = forages['épaisseur_couche'].apply(lambda e:
"épaisse" if e > 50 else "mince")

# حساب الليثولوجيا
forages['lithologie'].value_counts()
```

⏱️2.1.4 Analyse temporelle et spatiale

Séries temporelles

```
# تحويل التاريخ
forages['date'] = pd.to_datetime(forages['date'])
forages = forages.set_index('date')

# الإنتاج الشهري
prod_mensuelle = forages['production'].resample('M').sum()
```

Données spatiales

```
# حساب المسافات بين الإحداثيات
forages['distance'] = (((forages['x2'] - forages['x1'])**2) + ((forages['y2']
- forages['y1'])**2))**0.5
```

2.2 Visualisation des données géologiques

Les graphiques sont essentiels pour comprendre la structure et l'évolution géologique.

2.2.1 Histogrammes & courbes

```
# هيستوغرام الأعماق
plt.hist(forages['profondeur'], bins=8, edgecolor='black')
plt.title("Histogramme des profondeurs")
plt.show()
```

```
# منحنى الكثافة حسب العمق
plt.plot(forages['profondeur'], forages['densité'])
plt.gca().invert_xaxis()  # عكس العمق
plt.show()
```

2.2.2 Visualisations 2D

```
plt.scatter(forages['longitude'], forages['latitude'],
c=forages['profondeur'], cmap='viridis')
plt.colorbar(label="Profondeur (m)")
plt.show()
```

2.2.3 Visualisation 3D

```
from mpl_toolkits.mplot3d import Axes3D

ax.scatter(forages['longitude'], forages['latitude'], forages['profondeur'])
```

2.3 Modélisation géologique avec Python

2.3.1 Introduction

La modélisation permet de comprendre :

- l'évolution d'un bassin sédimentaire
- les dépôts successifs
- l'érosion
- la compaction

2.3.2 Simulations géologiques

Dépôt sédimentaire

```
epaisseur = taux_depot * temps
```

Érosion

```
erosion = 200 - 0.3 * temps
```

Profil final

```
profil = epaisseur - 0.3 * temps
```

2.3.3 Bibliothèques avancées

- **GemPy** : modèles 3D
- **PyGIMLi** : modélisation géophysique (résistivité, tomographie)
- **ObsPy** : traitement sismique
- **SciPy / NumPy** : équations géophysiques