TD 2 : Manipulation de Données avec Pandas

Programmation pour le développement d'applications SIG

M2 GER
Année Universitaire 2025-2026
Université de Nantes -- IGARUN

PLAN

1. **Introduction à Pandas** : Le "Excel" de Python.
2. **Chargement de données** : Lecture de fichiers CSV et Excel.
3. **Exploration** : Inspecter la structure des données.
4. **Manipulation** : Sélection, Filtrage, Tri.
5. **Calculs** : Création de nouvelles colonnes.
6. **Agrégation** : Statistiques par groupe (groupby).
7. **Export** : Sauvegarder les résultats.
8. **Mini-Projet TD** : Analyse de données réelles.

OBJECTIFS DU TD

À la fin de ce TD, vous ne devriez plus jamais avoir besoin d'ouvrir Excel pour traiter un gros fichier CSV.

COMPÉTENCES VISÉES

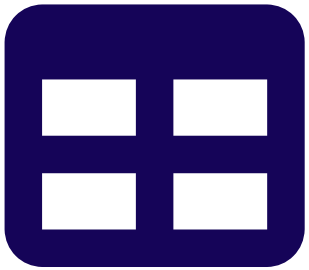
- Charger un jeu de données externe.
- Nettoyer des données (renommer, gérer les valeurs manquantes).
- Extraire des sous-ensembles de données (ex: "Toutes les villes > 10k hab").
- Produire des statistiques descriptives (Moyenne, Somme, Max).

1. La Librairie Pandas

QU'EST-CE QUE PANDAS ?

Pandas est la bibliothèque standard pour l'analyse de données en Python.
Son objet principal est le **DataFrame**.
Imaginez un DataFrame comme une feuille Excel programmable :

- **Lignes :** Observations (Indexées).
- **Colonnes :** Variables (Nommées).



DataFrame

IMPORTER PANDAS

Comme pour tout module, il faut l'importer avant de l'utiliser.
La convention universelle est de l'appeler `pd`.

```
import pandas as pd
```

> **ACTION**
>
> Ouvrez un nouveau Notebook Jupyter (`TD2_Nom.ipynb`) et exécutez cette ligne dans la première cellule.

2. Chargement & Exploration

LIRE UN FICHIER CSV

La fonction magique est `read_csv()`.

```
df = pd.read_csv("villes_france.csv")
```

ATTENTION AUX PIÈGES DU CSV

- **Séparateur :** Par défaut c'est la virgule `,`. En France, c'est souvent le point-virgule `;`.
- **Encodage :** Souvent `utf-8` ou `latin-1` (Windows).

```
df = pd.read_csv("fichier.csv", sep=";", encoding="latin-1")
```

INSPECTER LES DONNÉES

Une fois le fichier chargé dans la variable `df` (pour DataFrame), il faut regarder ce qu'il y a dedans.

- `df.head()` : Affiche les 5 premières lignes (Indispensable).
- `df.shape` : Affiche les dimensions (Lignes, Colonnes).
- `df.columns` : Liste les noms des colonnes.
- `df.dtypes` : Affiche le type de données de chaque colonne (int, float, object/texte).

EXERCICE 1 : CHARGEMENT

Pratique

DONNÉES RÉELLES (OPEN DATA)

Nous allons utiliser la **Base officielle des codes postaux** (data.gouv.fr).
Lien : https://www.data.gouv.fr/fr/datasets/r/dbe8a621... (CSV)

CONSIGNES

1. Copiez le lien du CSV ci-dessus.
2. Chargez-le directement dans pandas :
   ```
   df = pd.read_csv("20230823-communes-departement-region.csv", sep=";")
   ```
3. Affichez les 5 premières lignes.
4. Combien y a-t-il de communes ? (Utilisez `.shape`).

3. Manipulation (Sélection & Filtre)

SÉLECTIONNER UNE COLONNE

Pour extraire une seule colonne (ce qui renvoie une **Series**) :

```
noms = df["nom_commune"]
```

Pour extraire plusieurs colonnes (ce qui renvoie un nouveau **DataFrame**) :

```
subset = df[["nom_commune", "code_postal"]]
```

Notez les doubles crochets `[[...]]` pour la sélection multiple.

FILTRER LES LIGNES (QUERY)

C'est l'équivalent du "Select By Attribute" dans QGIS.
La syntaxe est : `df[ condition ]`.

```
# Garder uniquement les communes du code postal 44000
nantes_centre = df[ df["code_postal"] == 44000 ]

# Garder uniquement les communes dont le nom est "NANTES"
nantes = df[ df["nom_commune"] == "NANTES" ]
```

Astuce : On lit ça "DataFrame où la colonne Code Postal du DataFrame est 44000".

EXERCICE 2 : FILTRAGE

CONSIGNES

1. Créez un nouveau DataFrame `communes_44` contenant uniquement les communes dont le code postal commence par '44'.
   (Astuce: il faudra peut-être convertir le code postal en string ou utiliser mathématique : >= 44000 et < 45000).
2. Combien y en a-t-il ?
3. Cherchez la ligne correspondant à "SAINT-NAZAIRE".
4. Quel est son code INSEE ?

4. Calculs & Nettoyage

CRÉER DES COLONNES (CALCULS)

On peut faire des opérations vectorielles (sur toute la colonne d'un coup). C'est très rapide.

```
# Création d'une colonne fictive 'DENSITE'
# (Car notre CSV n'a pas la surface, on va simuler pour l'exemple)
df["SCORE"] = df["code_postal"] / 1000

# Créer une colonne majuscule
df["NOM_MAJ"] = df["nom_commune"].str.upper()
```

STATISTIQUES DESCRIPTIVES

Pandas offre des méthodes statistiques intégrées.

- `df["COL"].sum()` : Somme totale.
- `df["COL"].mean()` : Moyenne.
- `df["COL"].max()` / `.min()` : Maximum / Minimum.
- `df["COL"].std()` : Écart-type.
- `df.describe()` : Résumé statistique complet de toutes les colonnes numériques.

EXERCICE 3 : ANALYSE

CONSIGNES

1. Triez les communes de Loire-Atlantique (`communes_44`) par ordre alphabétique (`sort_values`).
2. Quelle est la première commune dans l'ordre alphabétique ?
3. Créez une colonne `DEPARTEMENT` qui contient les 2 premiers chiffres du code postal.
   Indice: `df["code_postal"].astype(str).str[0:2]`

5. Agrégation (Group By)

LE CONCEPT DE GROUPBY

Souvent, on veut des statistiques par catégorie (par département, par type de sol, etc.).
C'est le processus **Split-Apply-Combine** :

1. On sépare les données en groupes.
2. On applique une fonction (somme, moyenne) à chaque groupe.
3. On recolle les résultats.

```
resultat = df.groupby("DEPARTEMENT")
["nom_commune"].count()
```

Traduction : Grouper par département, et compter le nombre de communes.

EXERCICE 4 : SYNTHÈSE PAR DÉPARTEMENT

Pratique

CONSIGNES

En utilisant la colonne `DEPARTEMENT` créée précédemment sur le fichier France entière :

1. Comptez le nombre de communes par département.
2. Quel département (le code) a le plus de communes ?
3. (Bonus) Affichez ce résultat sous forme de graphique simple : `resultat.plot(kind="bar")`

6. Export & Conclusion

SAUVEGARDER SON TRAVAIL

Une fois les données nettoyées et calculées, il faut les exporter pour les utiliser ailleurs (QGIS, Excel, Rapport).

```
# Export en CSV (le plus courant)
df.to_csv("resultat_traite.csv", index=False, sep=";")

# Export en Excel (nécessite openpyxl)
df.to_excel("resultat.xlsx", index=False)
```

`index=False` évite de sauvegarder les numéros de ligne (0, 1, 2...) dans le fichier.

MINI-PROJET FINAL DU TD (AUTONOMIE)

ANALYSE DE LA POLLUTION (FICTIF)

Vous avez un fichier `polluants.csv` (Station, Date, Taux_NO2, Code_Zone).

1. Charger le fichier.
2. Supprimer les lignes où le Taux_NO2 est manquant (`df.dropna()`).
3. Filtrer pour ne garder que les taux > 50 (Seuil d'alerte).
4. Calculer la moyenne des taux par Code_Zone.
5. Exporter la liste des stations en alerte dans un nouveau CSV.

CE QU'IL FAUT RETENIR

- `pd.read_csv()` : La porte d'entrée.
- `df[condition]` : Le filtre magique.
- `df["new"] = ...` : Ajouter de l'info.
- `df.groupby()` : Synthétiser l'info.

PROCHAIN COURS (TD 3)

Nous ajouterons la dimension spatiale avec **GeoPandas**. Nous transformerons ces tableaux en cartes.