TD 3 : GeoPandas

Partie 3 : Analyse Spatiale Multi-Couches

M2 GER
Université de Nantes -- IGARUN

État des Lieux (Entrées)

Nous disposons de deux GeoDataFrames propres et projetés (Lambert-93) :

1. QUARTIERS (POLYGONES)

- Géométrie : Limites administratives.
- Attributs : Nom, ID.
- **Calculé :** Surface du quartier en km².

2. PARCS (BUFFERS)

- Géométrie : Disques de 50m.
- Attributs : Nom, Type.
- **Calculé :** Surface du parc en m².

Objectif de l'Analyse

> **Quel est le pourcentage de surface couverte par des espaces verts dans chaque quartier ?**

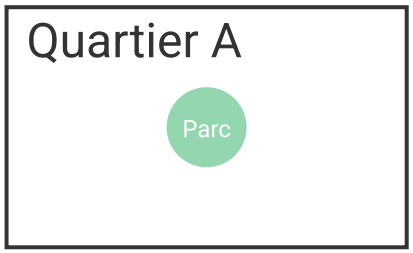
Méthodologie :

1. Associer chaque parc à son quartier (**Jointure Spatiale**).
2. Sommer les surfaces des parcs par quartier (**Agrégation**).
3. Calculer le ratio (**Calcul de Champs**).
4. Visualiser le résultat (**Cartographie**).

1. La Jointure Spatiale (Spatial Join)

Le Concept de Jointure Spatiale

Une jointure classique lie deux tables via une clé commune (ID).
Une **jointure spatiale** lie deux tables via leur **position géographique**.
Exemple : "Donne-moi le nom du quartier qui contient ce parc".



Le Parc hérite de "Quarti

La Fonction sjoin()

La syntaxe GeoPandas est : `gpd.sjoin(left_df, right_df, how, predicate)`

- **Left :** Ce qu'on veut conserver (les Parcs).
- **Right :** Ce qui donne l'information (les Quartiers).
- **Predicate :** La règle géométrique (intersects, within, contains).

```
# Jointure Spatiale
parcs_avec_quartier = gpd.sjoin(
    parcs_buffer,
    quartiers_L93,
    how="inner",
    predicate="intersects"
)
```

Exercice 3 : Exécution

> **ACTION**
>
> Exécutez la jointure spatiale dans votre notebook.

Vérifiez le résultat :

```
# Afficher les colonnes résultantes
print(parcs_avec_quartier.columns)
```

Vous devriez voir les attributs des deux couches (ex: `nom_left` pour le parc, `nom_right` pour le quartier).

Nettoyage des Colonnes

La jointure crée souvent des suffixes (`_left`, `_right`). Clarifions cela.

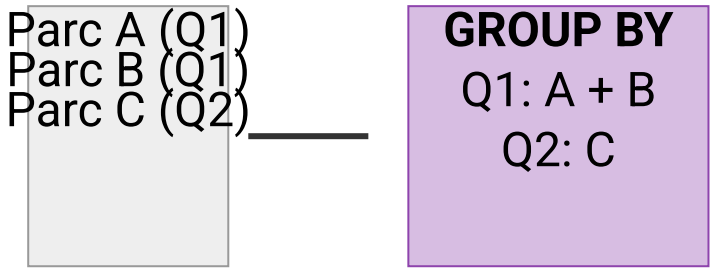
```
# Renommer pour la clarté
parcs_avec_quartier = parcs_avec_quartier.rename(columns={
    "nom_left": "nom_parc",
    "nom_right": "nom_quartier"
})

# Aperçu
parcs_avec_quartier[["nom_parc", "nom_quartier", "surf_m2"]].head()
```

2. Agrégation des Données (GroupBy)

Le Principe d'Agrégation

Nous avons une liste de parcs, chacun associé à un quartier.

Nous voulons **sommer** la surface des parcs pour chaque quartier.



Exercice 4 : Somme par Quartier

Utilisez la méthode `groupby()` de Pandas.

```
# Grouper par nom de quartier et sommer la surface
synthese_parcs = parcs_avec_quartier.groupby("nom_quartier")["surf_m2"].sum()

# Convertir le résultat (Série) en DataFrame
df_synthese = synthese_parcs.reset_index()
df_synthese.columns = ["nom_quartier", "total_parc_m2"]
```

Vérification des Résultats


```
df_synthese.head()
```

nom_quartier	total_parc_m2
Centre-Ville	45000.0
Doulon - Bottière	125000.0
...	...

Attention : Ce tableau ne contient plus de géométrie. C'est un simple DataFrame Pandas.

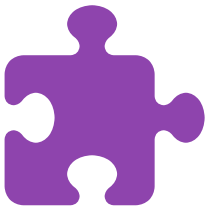
3. Reconstruction de la Table (Merge)

Pourquoi une Jointure Attributaire ?

Nous avons :

1. `quartiers_L93` : Géométrie des quartiers, mais pas les stats des parcs.
2. `df_synthese` : Stats des parcs, mais pas de géométrie.

Il faut les fusionner pour faire une carte.



La Méthode Merge

Nous allons joindre les données statistiques **vers** la couche géographique.

```
# Jointure Gauche (Left Join) pour garder tous les quartiers
# même ceux sans parcs.

carte_finale = quartiers_L93.merge(
    df_synthese,
    left_on="nom",       # Colonne dans quartiers_L93
    right_on="nom_quartier", # Colonne dans df_synthese
    how="left"
)
```

Gestion des Données Manquantes

Si un quartier n'a aucun parc, la jointure produira une valeur `NaN` (Not a Number). Pour le calcul, il faut remplacer ces `NaN` par 0.

```
# Remplacer NaN par 0
carte_finale["total_parc_m2"] = carte_finale["total_parc_m2"].fillna(0)
```

4. Calcul du Ratio & Cartographie

Préparation au Calcul

Nous voulons un pourcentage.

- Numérateur : Surface des parcs en m^2.
- Dénominateur : Surface du quartier en m^2.

Rappel : Nous avions calculé la surface du quartier en km^2. Il faut être cohérent avec les unités.

Exercice 5 : Calcul du Ratio


```
# 1. Convertir la surface quartier en m2
surf_quartier_m2 = carte_finale["surf_km2"] * 1000000

# 2. Calculer le ratio (%)
carte_finale["ratio_vert"] = (carte_finale["total_parc_m2"] /
surf_quartier_m2) * 100

# 3. Vérifier les résultats
carte_finale[["nom", "ratio_vert"]].sort_values(by="ratio_vert",
ascending=False)
```

Cartographie Statique

Créons une carte choroplèthe (couleurs graduées).

```
fig, ax = plt.subplots(figsize=(12, 10))

carte_finale.plot(
    column="ratio_vert",
    cmap="Greens",
    legend=True,
    edgecolor="black",
    legend_kwds={'label': "% de surface verte"},
    ax=ax
)

ax.set_title("Densité d'Espaces Verts à Nantes")
ax.set_axis_off()
```

5. Export & Conclusion

Formats d'Export

GeoPackage (.gpkg)

- Moderne, performant.
- Un seul fichier.
- Recommandé pour QGIS.

Shapefile (.shp)

- Obsolète mais universel.
- Multiples fichiers.
- Noms de colonnes tronqués (10 chars max).

Exporter les Résultats


```
# Export propre en GeoPackage
carte_finale.to_file("analyse_espaces_verts.gpkg", driver="GPKG")

# Export en GeoJSON (pour le web)
carte_finale.to_file("analyse_espaces_verts.geojson", driver="GeoJSON")
```

Bonus : Carte Interactive

Générer une carte web interactive avec `.explore()`.

```
m = carte_finale.explore(
    column="ratio_vert",
    cmap="viridis",
    tooltip=["nom", "ratio_vert"]
)

# Sauvegarder en HTML
m.save("carte_interactive.html")
```

Ouvrez le fichier HTML généré dans votre navigateur.

Synthèse de la Méthode

1. **Chargement :** Lecture des fichiers bruts.
2. **Harmonisation :** Reprojection en Lambert-93.
3. **Modélisation :** Transformation Points -> Surfaces (Buffers).
4. **Croisement :** Jointure Spatiale.
5. **Analyse :** Agrégation et calcul de ratio.
6. **Valorisation :** Carte et Export.

Conclusion du Module TD 3