TD 3 : GeoPandas

Partie 2 : Géométrie & Calculs

M2 GER
Université de Nantes -- IGARUN

État des lieux

Nous disposons de deux jeux de données chargés (Partie 1) :

QUARTIERS

Type : Polygones
CRS : WGS84 (Degrés)

PARCS

Type : Points
CRS : WGS84 (Degrés)

1. Systèmes de Coordonnées (CRS)

Inspection des données brutes

Regardons les coordonnées actuelles de nos quartiers.

```
quartiers.geometry.head(1)
```

```
0 POLYGON ((-1.551 47.215, -1.552 47.216 ...))
```

Observation : Les valeurs sont petites (47, -1). Ce sont des latitudes et longitudes (Degrés). C'est inadapté aux calculs de surface.

Pourquoi changer de CRS ?

WGS84 (EPSG:4326)

- Unité : **Degrés**
- La distance d'un degré change selon la latitude.
- Calcul d'aire $\approx$ Impossible.

Lambert-93 (EPSG:2154)

- Unité : **Mètres**
- Projection plane officielle France.
- Calcul d'aire = m^2.

La Reprojection

Nous utilisons la méthode `to_crs()`.

```
# Conversion vers le système métrique français
quartiers_L93 = quartiers.to_crs(epsg=2154)
parcs_L93 = parcs.to_crs(epsg=2154)

# Vérification
print(quartiers_L93.crs)
```

Rappel : Toujours reprojeter les deux couches dans le même système.

Vérification des Coordonnées

Vérifions que la transformation a fonctionné.

```
quartiers_L93.geometry.head(1)
```

```
0 POLYGON ((355120.5 6689001.2, ...))
```

Succès : Nous avons des coordonnées en centaines de milliers (X) et millions (Y). Ce sont bien des mètres.

Contrôle Visuel


```
fig, ax = plt.subplots(figsize=(8, 8))
quartiers_L93.plot(ax=ax, edgecolor="black", facecolor="none")
parcs_L93.plot(ax=ax, color="green", markersize=2)
plt.show()
```

Les points doivent se superposer parfaitement aux polygones.

2. Ingénierie Géométrique

Le Problème des Parcs

Calculons la surface actuelle de nos parcs :

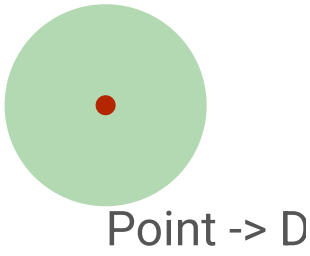
```
# Essai de calcul de surface
parcs_L93.geometry.area.head()
```

> **Résultat :** `0.0` partout.

Un **Point** n'a pas de dimension physique (longueur x largeur = 0). Pour faire de l'analyse spatiale, il nous faut des surfaces.

Solution : Le Buffer

Nous allons appliquer une zone tampon (**Buffer**) autour de chaque point.
Hypothèse : Faute de donnée précise, on attribue un rayon moyen de **50m** à chaque parc.



Application du Buffer


```
# 1. Copier le GeoDataFrame (Sécurité)
parcs_buffer = parcs_L93.copy()

# 2. Transformer la géométrie
# Le rayon est en METRES (car nous sommes en L93)
parcs_buffer["geometry"] = parcs_L93.geometry.buffer(50)
```

La colonne geometry contient désormais des **POLYGON** (des cercles).

Comparaison Avant / Après


```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 7))

parcs_L93.plot(ax=ax1, color="blue", markersize=5)
ax1.set_title("Avant : Points")

parcs_buffer.plot(ax=ax2, color="green", alpha=0.5)
ax2.set_title("Après : Surfaces (50m)")
```

3. Calculs d'Indicateurs

Surface des Parcs (Réelle)

Maintenant que ce sont des polygones, `.area` renvoie une valeur.

```
# Stocker la surface en m2
parcs_buffer["surf_m2"] = parcs_buffer.geometry.area

# Validation mathématique (Pi * r^2)
# 3.14 * 50^2 = 7850 m2
parcs_buffer[["nom_usuel", "surf_m2"]].head()
```

Surface des Quartiers

Calculons la surface de référence (les quartiers).

```
# Calcul de l'aire des quartiers
quartiers_L93["surf_m2"] = quartiers_L93.geometry.area
```

Conversion : Pour des quartiers, le km² est plus lisible.

```
# 1 km2 = 1 000 000 m2
quartiers_L93["surf_km2"] = quartiers_L93["surf_m2"] / 1e6
```

Exercice 3 : Le Plus Grand Quartier

CONSIGNE

Utilisez `sort_values()` pour trouver quel quartier a la plus grande superficie.

```
quartiers_L93.sort_values(
    by="surf_km2",
    ascending=False
).head(1)
```

4. Opérations Avancées (Bonus)

Centroïdes

L'opération inverse : transformer un Polygone en Point (Centre de gravité).

```
# Extraire le centre des quartiers
centres = quartiers_L93.geometry.centroid

# Affichage combiné
base = quartiers_L93.plot(color='white', edgecolor='black')
centres.plot(ax=base, color='red', markersize=5)
```

Utile pour placer des étiquettes (labels) sur une carte.

Enveloppe (Bounding Box)

Le rectangle minimal englobant une forme.

```
enveloppes =
quartiers_L93.envelope
enveloppes.plot(alpha=0.5,
edgecolor='blue')
```

Simplification extrême de la géométrie.

Distance

Calculer la distance entre deux géométries (en mètres).

```
# Distance entre le premier parc et le premier quartier
p1 = parcs_L93.geometry.iloc[0]
q1 = quartiers_L93.geometry.iloc[0]

dist = p1.distance(q1)
print(f"Distance : {dist:.2f} mètres")
```

Si la distance est 0, c'est que le parc est DANS le quartier.

Conclusion Partie 2

Bilan Technique

DONNÉES PRÊTES

- **Quartiers :** Reprojetés en Lambert-93. Surface calculée (km^2).
- **Parcs :** Reprojetés. Transformés en Surfaces (Buffers). Surface calculée (m^2).

Nous avons résolu le problème d'incompatibilité géométrique.

Vers la Partie 3

Nous avons des surfaces de parcs et des surfaces de quartiers.

Question : Comment savoir quelle surface de parc appartient à quel quartier ?

Il manque le lien entre les deux couches.

Prochaine étape : La Jointure Spatiale (Spatial Join).