

# TD 4 : Analyse Raster

Imagerie Satellitaire & MNT avec Rasterio

**M2 GER**

Université de Nantes -- IGARUN

# 1. Rappel : Vecteur vs Raster



## Vecteur (GeoPandas)

- Objets discrets (Points, Lignes).
- Attributs dans une table.
- Précision infinie (théorique).

## Raster (Rasterio)

- Grille continue de pixels.
- 1 pixel = 1 valeur (Altitude, Couleur).
- Résolution fixe (ex: 10m/pixel).

En Python, la référence absolue est **Rasterio** (développée par Mapbox).

## POURQUOI RASTERIO ?

Elle permet de lire et écrire des fichiers géospatiaux (GeoTIFF, JP2) en les transformant en tableaux **NumPy**.

*Rappel : NumPy est la librairie de calcul matriciel ultra-rapide de Python.*

## **2. Installation & Chargement**



# Installation

Setup

Comme pour GeoPandas, **ne jamais utiliser PIP** directement sur Windows.  
Si vous avez déjà installé GeoPandas via Conda au TD 3, vous avez probablement déjà Rasterio.

```
# Vérification dans Anaconda Prompt  
conda list rasterio  
  
# Si absent :  
conda install -c conda-forge rasterio
```

# Données pour ce TD (Fournies par l'enseignant)

Data

Nous allons travailler avec deux fichiers exportés de Google Earth Engine :

## 1. MODÈLE NUMÉRIQUE DE TERRAIN (MNT)

**Fichier** : Nantes\_MNT\_IGN\_10m.tif

**Source** : IGN RGE ALTI 1m.

**Contenu** : Altitude en mètres.

## 2. IMAGE SATELLITE SENTINEL-2

**Fichier** : Nantes\_Sentinel2\_4Bands.tif

**Contenu** : 4 Bandes (Bleu, Vert, Rouge, Proche Infrarouge).

**Action** : Copiez ces deux fichiers du Github.

# Ouvrir un Raster

Code

On n'utilise pas `read_file` mais `open`.

```
import rasterio
import matplotlib.pyplot as plt

# Ouvrir la connexion au fichier
src = rasterio.open("TD4_Data/Nantes_MNT_IGN_10m.tif")
```

*Attention : `src` n'est pas les données, c'est juste un curseur de lecture (comme ouvrir un livre sans le lire).*



# **3. Métadonnées & Géoréférencement**



# Lire les Métadonnées

Inspection

Avant de lire les pixels, il faut comprendre l'image.

```
# Dimensions (en pixels)  
print(src.width, src.height)  
  
# Nombre de bandes (couches)  
print(src.count) # 1 pour notre MNT  
  
# Système de coordonnées  
print(src.crs)
```

# La Matrice de Transformation

Complexe

C'est l'élément clé du géoréférencement. Elle lie les pixels (Ligne, Colonne) aux coordonnées (X, Y).

```
print(src.transform)
```

## INTERPRÉTATION

```
| 0.000089..., 0.0, -1.60... |  
| 0.0, -0.000089..., 47.25... |
```

- La valeur  $\sim 0.000089$  correspond à la résolution en degrés (environ 10m).
- Le CRS est EPSG:4326 (WGS84).

# Exercice 1 : Analyse

---

Pratique

## CONSIGNE

1. Ouvrez le fichier Nantes\_MNT\_IGN\_10m.tif.
2. Affichez sa résolution spatiale.
3. Affichez son système de coordonnées (CRS).
4. Vérifiez le nombre de bandes (Cela doit être 1).

# Exercise 1 : Correction

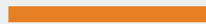
Solution

```
src = rasterio.open("Nantes_MNT_IGN_10m.tif")

res_x = src.transform[0]
res_y = src.transform[4]

print(f"Résolution : {res_x} x {res_y} degrés")
print(f"CRS : {src.crs}")
print(f"Bandes : {src.count}")
```

## **4. Lecture et Visualisation des Pixels**



# Lire les Données (Read)

Code

Pour manipuler les valeurs, on transfère les données du disque vers la mémoire (dans un tableau NumPy).

```
# Lire la première bande (L'altitude)
mnt = src.read(1)

print("Type :", type(mnt))
print("Dimensions :", mnt.shape)
```

*Rappel : En Python, on commence à compter les index à 0, mais en SIG/Rasterio, les bandes commencent souvent à 1.*

# Le Tableau NumPy

Structure

La variable `mnt` est maintenant une simple matrice de chiffres.

```
# Afficher un petit morceau (coin haut gauche)  
print(mnt[:5, :5])
```

Vous verrez des valeurs d'altitude en mètres.



Pour voir l'image, on utilise `imshow` (Image Show) de Matplotlib.

```
plt.figure(figsize=(10, 10))
plt.imshow(mnt, cmap="terrain")
plt.colorbar(label="Altitude (m)")
plt.title("MNT Nantes (IGN)")
plt.show()
```

*Essayez d'autres colormaps : 'viridis', 'magma', 'gray'.*

Les zones "vides" sont parfois remplies par une valeur spéciale (ex: -9999).

```
# Vérifier la valeur NoData
print(src.nodata)

# Masquer les valeurs NoData si besoin
import numpy.ma as ma
if src.nodata is not None:
    mnt_masque = ma.masked_values(mnt, src.nodata)
    plt.imshow(mnt_masque, cmap="terrain")
```

# Vers la Partie 2

---

Transition

Nous savons lire et voir un raster.

Dans la prochaine partie, nous allons manipuler les pixels :

- Calculer l'altitude moyenne de Nantes.
- Identifier les zones basses (inondables ?).
- Produire un histogramme des altitudes.