TD 4 : Analyse Raster

Partie 2 : Manipulation & Algèbre de Carte

M2 GER
Université de Nantes -- IGARUN

État des lieux

Dans la Partie 1, nous avons :

- Ouvert le fichier `Nantes_MNT_IGN_10m.tif`.
- Chargé les données dans une variable `mnt` (tableau NumPy).
- Affiché l'image brute.

OBJECTIFS PARTIE 2

Extraire de l'information quantitative de ce tableau de chiffres (Statistiques, Seuillage, Zones à risque).

1. Statistiques Globales

La Puissance de NumPy

Notre variable `mnt` est un tableau NumPy (matrice).
NumPy possède des fonctions ultra-optimisées pour analyser ces millions de pixels.

```python
import numpy as np

# Calculer la moyenne de toute l'image
moyenne = np.mean(mnt)
print(f"Altitude moyenne : {moyenne:.2f} m")
```

Le Problème du NoData

Si votre image contient des bordures noires (valeurs 0 ou -9999), elles faussent la moyenne !

```
Exemple : [10, 12, 11, -9999] → Moyenne calculée = -2491 (FAUX)
```

Il faut dire à NumPy d'ignorer ces valeurs spécifiques.

Masquer les NoData

Nous utilisons `numpy.ma` (Masked Array) pour cacher les valeurs invalides.

```
# Créer un tableau masqué
# On masque tout ce qui est égal à la valeur NoData du fichier
if src.nodata is not None:
    mnt_valid = np.ma.masked_equal(mnt, src.nodata)
else:
    # Si pas de NoData défini, on suppose que c'est -9999
    mnt_valid = np.ma.masked_equal(mnt, -9999)
```

Désormais, les calculs ignoreront les pixels masqués.

Exercice 1 : Statistiques Réelles

CONSIGNES

Sur votre tableau masqué `mnt_valid`, calculez et affichez : 1. L'altitude minimale. 2. L'altitude maximale. 3. L'altitude moyenne. 4. L'écart-type (standard deviation).

Aide : `np.min()`, `np.max()`, `np.mean()`, `np.std()`.

Correction Exercice 1


```
print(f"Min : {np.min(mnt_valid)} m")
print(f"Max : {np.max(mnt_valid)} m")
print(f"Moyenne : {np.mean(mnt_valid):.1f} m")
print(f"Ecart-type : {np.std(mnt_valid):.1f} m")
```

Vous devriez trouver un Min proche de 0 (Loire) et un Max vers 90-100m (Sillon de Bretagne).

2. Analyse de Distribution (Histogramme)

L'Histogramme des Altitudes

Pour comprendre le relief, rien de tel qu'un histogramme.
Il montre la répartition des pixels : "Y a-t-il plus de plaines ou de collines ?"

```
# Aplatir le tableau (passer de 2D à 1D) pour l'histogramme
valeurs = mnt_valid.compressed() # .compressed() garde seulement les valeurs valides

plt.figure(figsize=(10, 6))
plt.hist(valeurs, bins=50, color='skyblue', edgecolor='black')
plt.title("Distribution des altitudes à Nantes")
plt.xlabel("Altitude (m)")
plt.ylabel("Nombre de pixels")
plt.show()
```

3. Algèbre de Carte (Reclassification)

Le Concept de Reclassification

C'est transformer une image **continue** (altitudes variées) en une image **discrète** (catégories).

Exemple : On veut isoler les zones basses potentiellement inondables.

- Si Altitude < 10m -> Classe 1 (Risque)
- Si Altitude >= 10m -> Classe 0 (Sûr)

Masques Binaires

En Python, c'est une simple comparaison logique.

```
# Créer un masque (Tableau de Vrai/Faux)
masque_inondation = (mnt_valid < 10)

# Afficher le résultat
plt.figure(figsize=(10, 10))
plt.imshow(masque_inondation, cmap="Blues")
plt.title("Zones < 10m (Potentiellement Inondables)")
plt.show()
```

Les zones en bleu foncé sont celles qui respectent la condition (True).

Exercice 2 : Seuillage Complexe

CONSIGNES

Créez une carte classifiée simple avec 3 classes : 1. **Basses terres :** < 15m (Valeur 1) 2. **Plateau :** entre 15m et 40m (Valeur 2) 3. **Collines :** > 40m (Valeur 3)

Astuce : Utilisez `np.where(condition, valeur_si_vrai, valeur_si_faux)` ou créez un tableau vide et remplissez-le.

Exercice 2 : Correction


```
# Créer une copie vide (remplie de zéros) de la même taille
classification = np.zeros(mnt.shape)

# Remplir les classes
classification[mnt < 15] = 1
classification[(mnt ⩾ 15) & (mnt ⩽ 40)] = 2
classification[mnt > 40] = 3

# Masquer les NoData pour le rendu
classif_valid = np.ma.masked_where(mnt_valid.mask, classification)

plt.imshow(classif_valid, cmap="viridis")
plt.colorbar(ticks=[1, 2, 3], label="Classe")
```

4. Export du Résultat

Sauvegarder un Raster

Code

Vous avez créé une nouvelle image (la classification). Il faut l'enregistrer en **GeoTIFF** pour l'ouvrir dans QGIS.
On réutilise les métadonnées du fichier source (`src.meta`), mais on doit les mettre à jour (car le type de données a changé, ce sont des entiers 1, 2, 3 maintenant).

```
# Copier les métadonnées d'origine
new_meta = src.meta.copy()

# Mettre à jour le type de données (entier) et le nombre de bandes
new_meta.update({
    "driver": "GTiff",
    "height": classification.shape[0],
    "width": classification.shape[1],
    "count": 1,
    "dtype": "int32"
})
```

Écriture du Fichier


```
# Écriture sur le disque
with rasterio.open("classification_nantes.tif", "w", **new_meta) as dest:
    dest.write(classification.astype("int32"), 1)
```

Le fichier `classification_nantes.tif` est maintenant créé dans votre dossier. Vous pouvez l'ouvrir dans QGIS !

Résumé Partie 2

Nous savons maintenant :

- Calculer des statistiques sur une image entière.
- Gérer les valeurs aberrantes (NoData).
- Effectuer des opérations conditionnelles (Reclassification).
- Exporter une nouvelle image géoréférencée.

Vers la Partie 3...

Jusqu'ici, nous avons travaillé sur **une seule bande** (Altitude).
Dans la prochaine partie, nous allons travailler avec **4 bandes** (Image Satellite Sentinel-2).
Objectif : Calculer un indice de végétation (NDVI) pour cartographier la santé des plantes.