TD 4 : Analyse Raster

Partie 3 : Télédétection & Indices Spectraux

M2 GER
Université de Nantes -- IGARUN

Contexte de la Partie 3

Jusqu'à présent, nous avons travaillé sur un MNT (1 seule bande = Altitude).
Les images satellites (comme Sentinel-2) sont **multi-spectrales**.

OBJECTIFS

- Manipuler un raster "Cube" (3 dimensions : Hauteur, Largeur, Bandes).
- Recomposer une image couleur (RGB).
- Calculer un indice de végétation (NDVI) pour cartographier la biomasse.

1. L'Image Multi-Spectrale

Structure d'une Image Satellite

Une image satellite n'est pas une simple "photo". C'est un empilement de matrices.

Notre fichier Sentinel-2 :

- **Bande 1 :** Bleu (490 nm)
- **Bande 2 :** Vert (560 nm)
- **Bande 3 :** Rouge (665 nm)
- **Bande 4 :** Proche Infrarouge (842 nm)

Array 3D : (4, Y, X)

Lecture du Fichier

Code

Ouvrons notre image Sentinel-2.

```
import rasterio
import numpy as np

# Ouvrir le fichier
src_s2 = rasterio.open("TD4_Data/Nantes_Sentinel2_4Bands.tif")

# Vérifier le nombre de bandes
print("Nombre de bandes :", src_s2.count)
print("Dimensions :", src_s2.width, src_s2.height)
```

Chargement des Bandes

Pour travailler, il faut extraire les bandes utiles dans des variables séparées.

```
# Rappel : Dans ce fichier spécifique (export GEE) :
# B1=Bleu, B2=Vert, B3=Rouge, B4=NIR

blue = src_s2.read(1)
green = src_s2.read(2)
red = src_s2.read(3)
nir = src_s2.read(4)
```

Chaque variable est maintenant une matrice 2D NumPy.

2. Visualisation en Vraies Couleurs (RGB)

Le Principe du RGB

Pour afficher une image "naturelle" à l'écran, il faut combiner 3 canaux :

- **R**ed (Rouge) $\rightarrow$ Canal Rouge de l'écran.
- **G**reen (Vert) $\rightarrow$ Canal Vert de l'écran.
- **B**lue (Bleu) $\rightarrow$ Canal Bleu de l'écran.

Nous allons empiler nos matrices `red`, `green`, `blue` pour créer une image couleur.

Normalisation des Valeurs

Les données Sentinel-2 brutes sont souvent sur 12 ou 16 bits (valeurs de 0 à 10000+).
Pour l'affichage, Matplotlib préfère des valeurs entre **0 et 1**.

```
# Fonction de normalisation simple
def normalize(array):
    array_min, array_max = array.min(), array.max()
    return (array - array_min) / (array_max - array_min)

# Normalisation des bandes RGB
red_n = normalize(red)
green_n = normalize(green)
blue_n = normalize(blue)
```

Empilement (Stack)

Nous utilisons `numpy.dstack` (Depth Stack) pour empiler les 3 matrices.

```
# Création de l'image RGB
rgb = np.dstack((red_n, green_n, blue_n))

# Affichage
plt.figure(figsize=(10, 10))
plt.imshow(rgb)
plt.title("Vue Vraies Couleurs (RGB)")
plt.show()
```

Si l'image est trop sombre, on peut multiplier les valeurs (ex: `rgb * 3`) pour éclaircir.

3. Calcul d'Indice : Le NDVI

Qu'est-ce que le NDVI ?

Normalized Difference Vegetation Index.

La végétation saine :

- Absorbe le **Rouge** (pour la photosynthèse).
- Reflète fortement le **Proche Infrarouge (NIR)**.

FORMULE

$$ NDVI = \frac{(NIR - Red)}{(NIR + Red)} $$

Interprétation du NDVI

Le résultat varie toujours entre **-1 et +1**.

- **< 0 :** Eau (l'eau absorbe le NIR).
- **0 - 0.2 :** Sol nu, béton, route.
- **0.2 - 0.5 :** Végétation clairsemée / Herbe.
- **> 0.6 :** Végétation dense / Forêt saine.

Exercice 1 : Calcul du NDVI

CONSIGNE

En utilisant les matrices `nir` et `red` chargées précédemment, calculez la matrice `ndvi`.

Attention : Pour éviter une division par zéro, assurez-vous que le dénominateur n'est pas nul (ou laissez NumPy gérer les avertissements).

```
# Rappel : les variables sont des floats (si normalisées) ou des entiers
# Il est préférable de travailler avec des floats
ndvi = (nir.astype(float) - red.astype(float)) / (nir + red)
```

Visualisation du NDVI


```
plt.figure(figsize=(10, 10))
# cmap 'RdYlGn' est parfaite (Rouge=Bas, Vert=Haut)
plt.imshow(ndvi, cmap="RdYlGn")
plt.colorbar(label="NDVI")
plt.title("Carte de Végétation (Nantes)")
plt.show()
```

Les parcs et les bords de l'Erdre devraient apparaître en vert foncé.

4. Extraction de la Végétation

Seuillage (Thresholding)

Nous voulons isoler uniquement la végétation "active".
Critère : NDVI > 0.4.

```
# Créer un masque binaire
vegetation_mask = (ndvi > 0.4)

plt.figure(figsize=(10, 10))
plt.imshow(vegetation_mask, cmap="Greens")
plt.title("Zones Végétalisées")
```

Calcul de Surface Végétalisée

Quelle surface cela représente-t-il en km^2 ?

MÉTHODE

1. Compter le nombre de pixels "Vrais" (True).
2. Multiplier par la surface d'un pixel.
3. Convertir en km^2.

Exercice 2 : Surface Totale


```python
# 1. Compter les pixels (somme d'un masque booléen = nb de True)
nb_pixels_veg = np.sum(vegetation_mask)

# 2. Récupérer la résolution (MNT ou Sentinel, ici 10m)
res_x = src_s2.transform[0]
res_y = -src_s2.transform[4] # Attention c'est en degrés ici si WGS84
# Supposons pour l'exercice que l'image est projetée en mètres (10m)
pixel_area = 10 * 10 # 100 m2

# 3. Calcul
surf_totale_km2 = (nb_pixels_veg * pixel_area) / 1e6
print(f"Surface de végétation : {surf_totale_km2:.2f} km²")
```

5. Export du NDVI

Sauvegarder le Résultat

Pour utiliser cette carte NDVI dans QGIS, exportons-la en GeoTIFF.

```
# Préparer les métadonnées
meta = src_s2.meta.copy()
meta.update({
    "count": 1,              # Une seule couche
    "dtype": "float32",      # Décimaux
    "driver": "GTiff"
})

# Écrire
with rasterio.open("NDVI_Nantes.tif", "w", **meta) as dst:
    dst.write(ndvi.astype("float32"), 1)
```

Résumé Partie 3

Nous avons :

- Ouvert une image multi-spectrale complexe.
- Reconstitué une image couleur naturelle.
- Appliqué une formule physique (NDVI) pixel par pixel.
- Transformé une information spectrale en information thématique (Végétation).

Prochaine étape (Partie 4) : Croiser ce raster avec nos communes (Vecteur) pour des statistiques zonales.