Introduction à GDAL en ligne de commande

GDAL est une librairie open-source pour le traitement de données géospatiales au format raster et au format vecteur. Durant ce TD, nous allons apprendre à utiliser les commandes GDAL pour effectuer différents traitements sur des MNT.

Création de votre environnement

Tout d'abord, commencez par créer un nouvel environnement GDAL :

```
conda create -n gdal
```

Activez le et installez gdal :

```
conda activate gdal
conda install -c conda-forge gdal
```

Vérifiez la version de GDAL que vous venez d'installer :

```
gdalinfo --version
```

Extraction d'informations de données rasters

Vous avez à votre disposition 4 tuiles SRTM (dossier */data/*) autour des Alpes, du Jura et de la Suisse. Pour les visualiser, importez les dans QGIS. Vous pouvez télécharger les données via ce lien : [https://enlive.live.unistra.fr/index.php/s/Wtgfwib6ONR3bbq]

Chaque fichier GeoTIFF possède un ensemble de métadonnées (projection, emprise, coordonnées ...). Pour les extraire, utilisez la commande suivante :

```
gdalinfo <nom_de_votre_raster>
gdalinfo -stats <nom_de_votre_raster>
```

Attention, mettez vous dans le répertoire des images (commande *cd* pour changer de répertoire, *dir* pour lister les fichiers/dossiers du répertoire courant).

Question : Quelle est l'altitude *min* et *max* de chaque raster à votre disposition ?

Création d'un raster virtuel

Un raster virtuel est un simple fichier text qui permet de regrouper plusieurs rasters pour en faire une mosaique. L'avantage de cette méthode est qu'elle ne prend aucune place sur le disque dur. Un *VRT* se construit de la façon suivante :

```
gdalbuildvrt -input_file_list <liste_des_fichiers_a_merger.txt> <
    nom_de_fichier_en_sortie.vrt>
```

On constate que la commande prend en entrée un fichier texte contenant l'ensemble de nos fichiers à mosaiquer. Deux possibilités s'offrent à nous, créer un fichier texte à la main en notant chaque fichier ligne par ligne ou automatiser la création du fichier.

Sur Windows, voici la commande pour créer un fichier avec le nom de nos rasters :

```
dir /b *.hgt > filelist.txt
```

Ce qui nous donne : "Liste moi tous les fichiers finissant par *.hgt* dans le répertoire courant (commande *dir /b *.hgt*) et mets moi le résultats/la sortie de cette commande dans un fichier nommé *filelist.txt*."

Vous pouvez ensuite exécuter la commande *gdalbuildvrt* avec les bons noms de fichier.

Question : quelle sont cette fois-ci les valeurs *min* et *max* de notre zone d'étude ?

Conversion des rasters

Avec GDAL, il est possible de convertir n'importe quel format de raster en un autre qui est supporté et reconnu par la librairie. Vous trouverez l'ensemble des formats disponibles sur le site officiel : [https://gdal.org/drivers/raster/index.html]

Nous allons convertir notre *Virtual Raster* mosaiqué en un fichier *GeoTIFF* :

```
gdal_translate -of GTiff <nom_de_fichier_en_entree.vrt> <
    nom_de_fichier_en_sortie.tif>
```

Question : Quelle est la taille de notre fichier *GeoTIFF* ?

Relançons cette commande en ajoutant un paramètre supplémentaire :

```
gdal_translate -of GTiff -co COMPRESS=LZW <nom_de_fichier_en_entree.vrt> <
    nom_de_fichier_en_sortie.tif>
```

Question : Quelle est maintenant la taille de notre fichier *GeoTIFF* ?

Comme vous venez de le constater, la taille a significativement réduit sans altération de la qualité de notre raster. Attention, ce format de compression (*LZW*) ne fonctionne que pour les raster de type **entier**.

NoData et COG

Pour cette partie, nous allons travailler sur le raster que nous venons de compresser.

Question : Quelle est la valeur *NoData* affectée à notre nouveau raster ? Sans utiliser QGIS :-)

Pour changer la valeur *NoData* du raster, vous pouvez utiliser la commande suivante :

```
gdal_translate -of GTiff <nom_de_fichier_en_entree.tif> <
    nom_de_fichier_en_sortie.tif> -co COMPRESS=LZW -co PREDICTOR=2 -
    a_nodata <valeur_nodata>
```

Question : Attribuez au raster une nouvelle valeur *NoData* de -9999.

Il existe un autre type de fichier de plus en plus utilisé par tous les logiciels/services de traitement de la donnée géospatiale, le format d'image *COG* (pour Cloud-Optimized GeoTIFF). Ce fichier stocke l'image dans un *cloud* local que vous pouvez accéder à travers un client tel que QGIS.

Question : Pour changer le type de l'image, utilisez non plus l'extension de fichier **GeoTIFF**, mais l'extension **COG** avec *gdal_ translate*.

Hillshade et carte d'élévation en couleurs

Pour traiter les MNT (ou DEM en anglais), il existe une commande spécialement implémentée dans GDAL : *gdaldem*.

```
gdaldem hillshade <nom_de_fichier_entree.tif> <nom_de_fichier_en_sortie.
    tif> -s 111120
```

Note : -s permet de définir l'échelle de notre MNT. Étant en mettre, la documentation [https://gdal.org/programs/gdaldem.html] nous informe que la valeur à mettre est 111120.

Afin d'avoir un meilleur rendu du MNT, il est possible d'en faire une image RGB avec une échelle de couleur très simplement. Pour cela, créez un fichier texte (*.txt*) de format *elevation, rouge, vert, bleu* :

```
500,101,146,82
1000,190,202,130
1500,241,225,145
2000,244,200,126
2500,197,147,117
3000,204,169,170
3500,251,238,253
4000,255,255,255
```

Note : vous pouvez extraire l'encodage RGB d'une couleur sur ce lien [https://www.w3schools.com/colors/colors_rgb.asp]

Vous n'aurez plus qu'à construire votre image avec échelle de couleurs :

```
gdaldem color-relief <nom_de_fichier_entree.tif> <nom_de_fichier_colormap.
    txt> <nom_de_fichier_sortie.tif>
```

Question : Combien l'image comporte-elle de bandes ? Où pouvez-vous trouver cette information ?

Question : Exportez l'image en PNG avec GDAL.

Introduction à GDAL en Python

GDAL possède également une surcouche Python où tous les traitements que vous venez de faire par ligne de commande peuvent également être exécutés par code. Dans cette partie, nous allons voir comment télécharger et utiliser un code sur Github, lire et écrire une image sur le disque et calculer des indices spectraux en utilisant **Numpy** et le calcul vectoriel.

Téléchargement automatique d'images Sentinel-2 L2A par le portail Théia

Le pôle Théia a été créé fin 2012 et a pour vocation de faciliter l'usage des données d'observation de la Terre. Ainsi, plusieurs chaînes de traitement ont été développé pour prétraiter les données et proposer aux utilisateurs des produits corrigés et directement utilisables à travers des SIG.

Olivier Hagolle, ingénieur de recherche au CESBIO, a développé de nombreux scripts qu'il met à disposition de la communauté pour faciliter le téléchargement et le prétraitement des données, la création de masques de nuages ou encore le calcul d'indices.

Dans cette partie, nous allons utiliser l'un de ses scripts que télécharge automatiquement des images Sentinel-2 prétraitées au niveau 2A du pôle Théia [https://github.com/olivierhagolle/theia_download].

Le script a déjà été téléchargé et se trouve dans le dossier *sentinel-download*. Vous aurez besoin de créer un compte sur la plateforme Théia : [https://sso.theia-land.fr/theia/register/register.xhtml]

Rentrez ensuite votre email et votre mot de passe Théia dans le fichier *config_theia.cfg* :

```
serveur = https://theia.cnes.fr/atdistrib
resto = resto2
token_type = text
login_theia = email
password_theia = password
```

Note : Installez *curl* à votre environnement conda

```
conda install -c conda-forge curl
```

Vous pourrez ensuite simplement lancer le script par ligne de commande (exactement comme GDAL) :

```
python ./theia_download.py -t T32ULU -c SENTINEL2 -a config_theia.cfg -d
    2016-09-01 -f 2016-10-01
```

qui téléchargera tous les produits Sentinel-2 de la tuile 32ULU entre le 01/09/2016 et le 01/10/2016.

Il est possible d'inclure différents paramètres comme la ville, le numéro de la tuile Sentinel-2 ou encore la couverture nuageuse maximum souhaitée :

```
python ./theia_download.py -l 'Toulouse' -c SENTINEL2 -a config_theia.cfg
    -d 2016-09-01 -f 2016-10-01 -m 50
```

Pour cette commande, on télécharge toutes les images Sentinel-2 sur Toulouse entre le 01/09/2016 et le 01/10/2016 avec moins de 50% de couverture nuageuse.

Vous pouvez accéder à la documentation en ajoutant *–help* :

```
python ./theia_download.py --help
```

Question : Lancez le téléchargement d'une image de la tuile T32ULU du capteur Sentinel-2 de niveau 2A acquise entre le 22/07/2019 et le 25/07/2019 avec moins de 10% de couverture nuageuse.

Lire et écrire une image

La lecture d'une image peut se faire avec plusieurs librairie telles que GDAL, tifffile, OpenCV, Rasterio ... mais toutes ne proposent pas l'extraction d'informations spatiales. Dans le cadre de ce TD, nous allons utiliser GDAL pour la lecture et l'écriture des images satellites.

Importez l'ensemble des librairies dont nous aurons besoin durant ce TP et les suivants :

```
import osgeo.gdal as gdal
import osgeo.ogr as ogr
import osgeo.osr as osr
from osgeo.gdalconst import *
```

Pour lire une image (mono-bande ou multi-bande) sous forme de *dataset* :

```
dataset = gdal.Open(<chemin_vers_image>)
```

Il est possible d'extraire le nombre de bandes spectrales de notre image facilement :

```
nb_bands = dataset.RasterCount
```

Puis d'en lire une sous forme de tableau :

```
band = dataset.GetRasterBand(<numero_de_la_bande>)
array = band.ReadAsArray().astype(float)
```

ATTENTION : contrairement au tableau, le numéro de la bande commence à 1 et non à 0.

Pour écrire une nouvelle image avec GDAL, cela nécessite quelques lignes de code. Il faut dans un premier temps, extraire la projection et la transformation géographique de notre image initale (dans le cas où les projections et transformations sont les mêmes !). Ensuite, vous pouvez créer votre dataset avec le driver correspondant au format de votre image de sortie (ici GeoTIFF) avec plusieurs paramètres : le chemin vers la nouvelle image, le nombre de colonnes et de lignes, le nombre de bandes et le type des données.

```
geo_transform = dataset.GetGeoTransform()
projection = dataset.GetProjection()

driver = gdal.GetDriverByName("GTiff")
output_dataset = driver.Create(<output_image>, <nb_colonnes>, <nb_lignes>,
    <nb_de_bandes>, gdal.GDT_Float32)
output_dataset.SetProjection(projection)
output_dataset.SetGeoTransform(geo_transform)

#Si on a plusieurs bandes, il suffit de faire une boucle sur ces deux
   fonctions
output_dataset.GetRasterBand(<numero_bande>).WriteArray(<tableau_python>)
output_dataset.GetRasterBand(<numero_bande>).SetNoDataValue(<valeur_nodata
   >)
```

Question : Vous avez à votre disposition une image Sentinel-2 L2A du 24/07/2019. Calculer le NDVI de l'image en lisant les bandes spectrales nécessaires (Figure 1). Utilisez les bandes **FRE**.

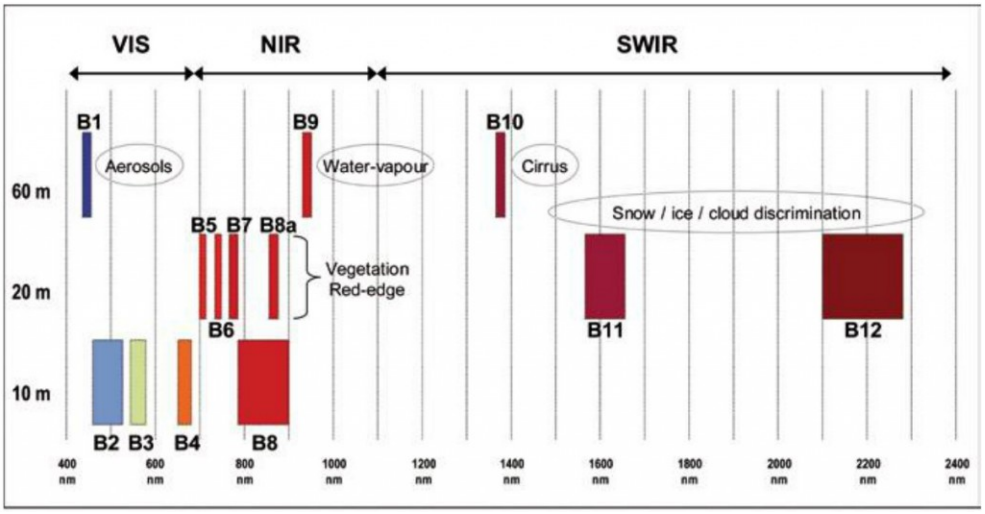

Figure 1: Bandes Sentinel-2 aux différentes résolutions spatiales.