Exercice de manipulation d'images Sentinel

Pour cet exercice, on vous donne :
— des images SENTINEL 2A téléchargées à l'aide de Theia (répertoire) (voir `http://www.cesbio.ups-tlse.fr/multitemp/?page_id=10464` pour les détails des fichiers).
— un répertoire contenant un fichier SHP correspondant aux champs d'une exploitation agricole.
Les deux couches sont dans le même système de projection.

L'objectif est d'obtenir des **statistiques zonales** des réflectances pour chaque parcelle. On verra dans la seconde partie comment réaliser des statistiques zonales (utilisation de **rasterstats**) mais ce qu'il semble évident, c'est que la **tuile SENTINEL est totalement surdimensionnée** par rapport à la taille des quelques parcelles à analyser. Il est donc plus que nécessaire de commencer par réduire les données à traiter. Ce sera l'objet de la première partie du travail. La seconde partie traitera du calcul des statistiques zonales. Finalement, deux extensions seront proposées : la gestion des nuages et le traitement de plusieurs fichiers.

Il est demandé de s'appuyer sur le cours, *c.-à-d.* d'utiliser les modules Python de GDAL pour réaliser vos programmes. Tâcher de penser à un possible utilisation future de votre programme et donc à la faciliter de réutilisation :
— préférer l'utilisation de variable à l'utilisation de constante (notamment pour les noms de fichiers)
— utiliser des fonctions pour décomposer votre code

1 Détourage de l'image par l'emprise d'un Shapefile

L'objectif de cette section est de créer une nouvelle image qui sera une *stack* de toutes les bandes de SENTINEL à une résolution de $20m$, sur la seule région d'intérêt correspondant à l'image vectorielle.

Exercice 1 (Traitement d'un fichier unique)

Question a) Faire un programme qui créé un fichier TIF avec une bande correspondant à la région de l'image image de la bande B12 (Flat Reflectance), résolution $20m$, à partir de l'emprise du fichier shapefile.

Je propose la décomposition suivante que vous pourrez adapter si vous souhaitez prendre une autre approche :

1. *créer le programme pour ouvrir votre fichier shapefile et d'afficher les coordonnées de son emprise (notez qu'il ne sera pas nécessaire de parcourir les features).*
2. *ouvrir l'image raster de la bande B12 et faites en sorte de l'extraire la matrice de pixels correspondant uniquement à la région de l'image qui correspond à l'emprise spatiale de la couche shapefile*
3. *déterminer la taille de la nouvelle image (en nombre de pixels) à partir de la matrice créée (utiliser la fonction* ***np.shape****).*
4. *concrétiser cette matrice dans un nouveau fichier au format TIF qui ne contiendra qu'une seule bande (le type de données sera* ***gdal.GDT_Int16****)*

Vous pourrez alors visualiser votre nouveau fichier dans QGis pour savoir si tout c'est correctement passé (par exemple en sur-impression de l'image d'origine). Il est alors probable que vous ayez des problèmes

Question b) Ajuster le SRS et le **geotransform** de votre nouvelle couche de données

Question c) Que se passerait-il si les deux couches n'étaient pas dans le même SRS ?

Exercice 2 (Traitement d'un fichier unique : cas des images de résolution $10m$) *On cherche à faire la même chose pour la bande B2 ... mais celle-ci est d'une résolution de $10m$. Il faut donc baisser sa résolution pour la passer à $20m$ en ne prenant qu'un pixel sur 2.*

Si ***data*** *est une matrice de deux dimensions, alors il est possible de construire une matrice avec 2 fois moins de lignes et 2 fois moins de colonnes. L'exemple ci-dessous illustre une fonctionnalité du module* ***numpy*** *qui vous facilitera grandement cette question*[1].

```
data = numpy.reshape([i for i in range(20)], (5, 4) )
echdata = data[0::2,0::2]
print(echdata)
```

Question a) Testez ce code et expliquez comment il pourra être utilisé dans votre programme.

Question b) Réaliser un programme s'inspirant fortement du programme précédent et qui construit le fichier TIF de la bande B2 en résolution $20m$ pour l'emprise du fichier shapefile

Exercice 3 (Empilage des couches) *Dans cette partie, l'objectif est maintenant de créer une fichier unique contenant toutes les couches de l'image SENTINEL. L'ordre des couches n'a pas vraiment d'importance ... il faudra juste que vous vous y retrouviez.*

Question a) Reprendre votre code précédent et construire un programme Python qui fera tout d'abord une *stack* d'images au format TIF à partir des images de résolution de $20m$: B5, B6, B7, B11 et B12.

- *Vous ferez en sorte de réduire le nombre de ligne de code nécessaire pour mener cette opération en utilisant dans variables ! ! En particulier, je ne veux pas d'une solution qui serait faite de 5 copier/coller du code ! Et comme il y a une répétition ... il faut une boucle !*
- *Vous pourrez également faire appel à l'utilisation de fonctions pour rendre le programme concis, lisible et réutilisable.*

Question b) Modifier ensuite votre code pour qu'il intègre maintenant les images de résolution $10m$
Vous penserez à vérifier votre résultat à la fin à l'aide de QGis.

Exercice 4 (Ajout de nouvelles couches : NDVI et NDSI)

Question a) Compléter votre programme précédent pour qu'il ajoute une couche NDVI (calculer à partir des bandes B8 et B4)

Question b) Idem avec le $NDSI$ (bandes B3 et B11)

À ce niveau, vous devriez avoir une image telle que le propose l'illustration en haut du sujet, dans un fichier TIF, vous aurez toutes vos couches pour l'étendue spatiale minimal pour couvrir toutes les parcelles à étudier et spatialement bien positionnées.

2 Statistiques zonales

L'objet de cette section est de réaliser des statistiques zonales simples ... par exemple la moyenne des valeurs d'une bande raster pour une zone délimitée par un polygone. Bien évidemment, on peut avoir besoin de réaliser cette même opération pour une collection de polygones, voire de calculer d'autres statistiques que la moyenne.

Nous allons pour cela étudier en détail le problème du calcul des statistiques zonales. Dans un premier temps, on vous propose donc de réaliser ces méthodes directement en les programmant (de deux manières différentes), puis on utilisera le module **rasterstat**. Vous vous intéresserez aux temps de calcul pour appréhender les conséquences des choix des implémentations sur l'efficacité du code/de l'algorithme.

Dans la suite de cet exercice, vous pourrez utiliser directement l'image construite précédemment. En cas de besoin, on vous fournit également cette image au format TIF.

2.1 Version algorithmique

Dans cette partie, l'idée est de vous faire programmer l'algorithme naïf des statistiques zonales (calcul uniquement la moyenne), on vous donne pour cela un pseudo-code à implémenter.

Exercice 5 (Un algorithme de calcul d'une moyenne zonale)

Question a) Réaliser un programme qui :

1. *Ouvre l'image TIF construite dans l'exercice précédent,*
2. *Ouvre le shapefile des parcelles agricoles,*
3. *Réalise de calcul de la valeur moyenne des pixels d'une bande pour chaque parcelle comme illustré par le pseudo-algorithme ci-dessous.*

```
POUR TOUT polygone P de la couche shapefile FAIRE:
    Sum=0
    Nb=0
    POUR TOUT pixel (i,j) de l'image I FAIRE:
        SI P contient le pixel (i,j) ALORS:
            Nb += 1
            Sum += I[i,j]

    print( "moyenne pour P:", Sum/Nb )
```

NB : On ne demande ici que de calculer ces statistiques et de les afficher. Il serait néanmoins très pertinents de mettre ces informations dans une structure de données appropriées (par exemple une liste, ou comme on le verra plus tard, dans un data.frame ***pandas****).*

Question b) Évaluer le temps nécessaire à mener ces calculs

```
import time
start_time = time.time()
# votre code
elapsed_time = time.time() - start_time
print("Temps ecoule: "+str(elapsed_time))
```

2.2 Version numpy

On procède maintenant d'une manière a priori plus efficace algorithmiquement et aussi plus en adéquation avec les bonnes pratiques de Python.

On vous propose pour cela le code ci-dessous qu'il faudra ajuster à vos besoins (notamment les noms de fichier) :

```
dataset = gdal.Open("extract_sentinel.tif", 0 )
geotransform = dataset.GetGeoTransform()
projection = dataset.GetProjection()
data = dataset.GetRasterBand(1).ReadAsArray()

# operation un peu complexe :
driver = gdal.GetDriverByName('MEM') # In memory dataset
target_ds = driver.Create('', dataset.RasterXSize, dataset.RasterYSize, 1, gdal.
    GDT_UInt16)
target_ds.SetGeoTransform( geotransform )
target_ds.SetProjection( str(projection) )
gdal.RasterizeLayer( target_ds, [1], Layer, burn_values=[0], options=["ATTRIBUTE=IDN"])
labeled_pixels = target_ds.GetRasterBand(1).ReadAsArray()
#affichage du resultat
plt.imshow(labeled_pixels)

#test d'une fonctionnalite
plt.figure()
onep = (labeled_pixels==6).astype(int)
plt.imshow( onep )

#calcul des moyennes zonales
for i in range(Layer.GetFeatureCount()):
    onep = (labeled_pixels==i).astype(int)
    moy=np.sum(onep * data)/np.sum(onep)
    print("Feature "+str(i)+": "+str(moy))
```

Le programme peut être téléchargé ici.

Exercice 6

Question a) Reprendre le code, testez le et analysez-le pour le comprendre

Question b) Évaluer le temps nécessaire à mener ces calculs et comparer avec la solution précédente

2.3 Utilisation du module rasterstat

rasterstats est un module Python pour calculer des statistiques de zones de données raster à partir de géométries vectorielles (voir `https://pythonhosted.org/rasterstats/`).

La fonction de statistique zonales de ce module utilise directement les noms de fichiers. Par exemple, le code ci-dessous calcule la minimum et maximum de la bande 1 pour chaque parcelle :

```
from rasterstats import zonal_stats
from pprint import pprint

stats = zonal_stats("parcelles.shp", "image.tif", band=1, stats=["min","max"])
pprint(stats)
```

Exercice 7

Question a) Adapter le code ci-dessous à votre besoin pour calculer les moyennes d'une bande pour chaque parcelle agricole.

Question b) Évaluer le temps nécessaire à mener ces calculs et comparer avec les solutions précédentes. Conclure sur ces comparaisons.

3 Extensions

Exercice 8 (Traiter plusieurs images en *batch*)

Question a) Construire un programme complet avec toutes les fonctionnalités vue dans les sections précédentes : sélection de l'image puis calcul de statistiques zonales

Question b) Modifier votre code pour qu'il puisse traiter plusieurs images pour le même fichier shapefile (par exemple, pour faire du suivi temporel)

Question c) Modifier votre code pour qu'il puisse intégrer plusieurs fichiers shapefile différents (un fichier par exploitation agricole, par exemple)

Exercice 9 (Gestion des nuages)

Question a) Dans la première partie du code, implémenter une fonction qui réalise la détection de nuage selon la méthode de Sentinel (`https://earth.esa.int/web/sentinel/technical-guides/sentinel-2-msi/level-2a/algorithm`)

Vous supposerez qu'il n'y a pas de neige et vous limiterez ainsi aux grande étapes 1 à 6 (sans rentrer dans les détails).

Question b) Ajouter une nouvelle couche à votre mini-image qui pourra servir de masque de nuages

Question c) Dans le calcul des statistiques zonales, vous ferez en sorte que les pixels qui ont des nuages ne soient pas pris en compte. Vous pourrez adapter l'un des codes de calcul, ou voir comment utiliser **rasterstats** avec un masque de nuages.