Creating Curve Number Grid from Soil and Landuse

Prepared by
Sayan Dey and Venkatesh Merwade
Purdue University
FAIR Science in Water Resources

Initialize PyQGIS

This code is used for initializing PyQGIS. You only need to execute this once per session.

In [ ]:
```
import sys, os
os.environ['QT_QPA_PLATFORM']='offscreen'
sys.path.append('/apps/share64/debian7/anaconda/anaconda3-5.1/envs/qgis/')
from qgis.core import *
from qgis.analysis import QgsNativeAlgorithms
#from qgis.utils import *
import processing
from processing.core.Processing import Processing

qgs = QgsApplication([], False)
qgs.initQgis()
Processing.initialize()
QgsApplication.processingRegistry().addProvider(QgsNativeAlgorithms())

from PyQt5.QtCore import QVariant
```

Import libraries

For this exercise, you will need one additional library: pandas.
Note that os and sys were already imported in the previous cell

```
In [ ]: import pandas as pd
```

Define path and file names for input and output

The input files are made available to you in a public folder. The path of the public folder is:
/srv/projects/cybertrainingfair/files/public/FAIR_Data_Processing/CN_Grid
It has the following files:
1) Boundary shapefile - Boundary.shp
2) Soil data shapefile - Soil.shp
3) Landuse raster - LU.tif
4) Curve Number Lookup Table - LookUp.csv
5) Landuse reclassify table - NLCD_reclass.csv
Assign variables to these filenames (including the full path). You can use _os.path.join()_ or string operations.
Finally, define your output folder here. This is where you will store your intermediate files and final output. This should be located in your mygeohub storage space. Note that your mygeohub home location is /home/mygeohub/username

```
In [ ]: input_folder = '/srv/projects/cybertrainingfair/files/public/FAIR_Data_Processing/CN_Grid'
        boundary_sh = os.path.join(input_folder, "Boundary.shp")
        soil_sh = input_folder+'/Soil.shp'
        landuse_tif = input_folder+'/LU.tif'
        cn_lookup = input_folder+'/LookUp.csv'
        lu_reclass = input_folder+'/NLCD_reclass.csv'
        if not os.path.exists(os.getcwd()+'/output'):
            os.mkdir(os.getcwd()+'/output')
        output_folder = os.getcwd()+'/output'
```

```
In [ ]: # help(QgsVectorLayer)
        # help(QgsRasterLayer)
        # os.path.exists(landuse_tif)
        # os.listdir(input_folder)
        output_folder
```

```
In [ ]: os.path.isfile(landuse_tif)
        os.path.join(input_folder, "Boundary.shp")
```

Check Data

Check if all three geospatial input files are in the same coordinate system

In [ ]:
```
boundary_crs = QgsVectorLayer(boundary_sh, baseName='Boundary CRS', providerLib='ogr').crs().authid()
soil_crs = QgsVectorLayer(soil_sh, baseName='Soil CRS', providerLib='ogr').crs().authid()
# import rasterio as rio
# with rio.open(landuse_tif) as landuse: # not available in qgis?
#     landuse.bounds
# landuse.crs
landuse_crs = QgsRasterLayer(landuse_tif, baseName='Landuse CRS').crs().authid()
```

In [ ]:
```
print('boundary:', boundary_crs, 'soil:', soil_crs, 'landuse:', landuse_crs)
```

Preparing Landuse Raster

The NLCD landuse raster contains many classes ranging from 11 to 95 as per NLCD code. We will be reclassifying these into 4 categories: Water (1), Medium Residential (2), Forest (3) and Agricultural (4). Print the NLCD_reclass.csv file to see the reclassification criterion.

In [ ]:
```
lu_table = pd.read_csv(lu_reclass)
```

In [ ]:
```
lu_table.head()
```

Create table layer for reclassification criterion

The following code creates a layer containing the reclassification table using [QgsVectorLayer() (https://qgis.org/pyqgis/3.4/core/QgsVectorLayer.html#qgis.core.QgsVectorLayer)](https://qgis.org/pyqgis/3.4/core/QgsVectorLayer.html#qgis.core.QgsVectorLayer) from a delimited file. This layer is required for specifying the reclassification cirterion. The text file needs to be input as a URI (Universal Resource Identifier). The URI for NLCD_reclass.csv is provided below.

In [ ]:
```
table_uri = 'file:///srv/projects/cybertrainingfair/files/public/FAIR_Data_Processing/CN_Grid/NLCD_re
class.csv?type=csv&detectTypes=yes&geomType=none&subsetIndex=no&watchFile=no'
table_lyr = QgsVectorLayer(table_uri, "", "delimitedtext")
```

In [ ]:
```
# table_lyr
```

Reclassify landuse raster

We will use native:reclassifybylayer (https://docs.qgis.org/3.4/en/docs/user_manual/processing_algs/qgis/rasteranalysis.html#reclassify-by-layer) with _processing.run()_.
You will need to specify the following arguments:
'INPUT_RASTER': full path to LU.tif,
'RASTER_BAND': 1,
'INPUT_TABLE': table_lyr,
'MIN_FIELD': 'min',
'MAX_FIELD': 'max',
'VALUE_FIELD': 'value'
'NO_DATA': -9999,
'RANGE_BOUNDARIES': 0,
'NODATA_FOR_MISSING': False,
'DATA_TYPE': 5,
'OUTPUT': full path to output file example: output_folder/lu_reclass_raster.tif

In [ ]:
```
processing.run("native:reclassifybylayer", {
    'INPUT_RASTER': landuse_tif,'RASTER_BAND':1,
    'INPUT_TABLE': table_lyr,
    'MIN_FIELD':'min','MAX_FIELD':'max','VALUE_FIELD':'value',
    'NO_DATA':-9999,'RANGE_BOUNDARIES':0,'NODATA_FOR_MISSING':False,'DATA_TYPE':5,
    'OUTPUT': os.path.join(output_folder,"lu_reclass_raster.tif")})
print("Landuse raster is ready!")
```

Create Soil Raster

Soil data is available as polygon (vector) dataset. We convert it into raster dataset using _gdal:rasterize_. We need to ensure that the soil raster has the same extent and pixel size (resolution) as the landuse raster. This is done using the "EXTENT" argument of _gdal:rasterize_.

Creating requisite expression for EXTENT

The EXTENT argument of _gdal:rasterize_ needs an expression (string) stating the west, east, south and north bounds of the raster as well as its coordinate system. The string has the syntax:
"west bound, east bound, south bound, north bound [CRS Auth ID]"

For this case, it should look like
'224850.725009,296730.725009,3403431.244106,3516291.244106 [EPSG:26917]'.

Create a raster layer for the landuse raster. From the layer's extent, extract the coordinate system and bounds to create the string shown above.

In [ ]:
```
# extent
lu_lyr = QgsRasterLayer(landuse_tif)
# ext = QgsRasterLayer(landuse_tif, baseName='Landuse Extent').extent()
ext = lu_lyr.extent()
```

In [ ]:
```
ext_str= str(ext.xMinimum())+','+str(ext.xMaximum())+','+str(ext.yMinimum())+','+str(ext.yMaximum())+
' '+ '['+str(landuse_crs)+']'
```

In [ ]:
```
ext_str
```

In [ ]:
```
# help(QgsRasterLayer)
```

Extracting horizontal and vertical resolution of landuse

From the landuse layer, get its resolution in X and Y direction using rasterUnitsPerPixelX and rasterUnitsPerPixelY attribute.

In [ ]:
```
pixelSizeX = int(lu_lyr.rasterUnitsPerPixelX())
pixelSizeY = int(lu_lyr.rasterUnitsPerPixelY())
```

Converting Soil polygon to raster

Create a vector layer for soil data. Use _processing.run_ to execute _gdal:rasterize_. It has the following arguments:
'INPUT': soil layer,
'FIELD':'HSG_Index',
'BURN':None,
'UNITS':1,
'WIDTH':pixelSizeX,
'HEIGHT':pixelSizeY,
'EXTENT': expression,
'NODATA':0,
'OPTIONS':'',
'DATA_TYPE':5,
'INIT':None,
'INVERT':False,
'OUTPUT':full path to file where output is saved

Hint: See how the reclassifybylayer tool has been used above.

In [ ]:
```
soil_lyr = QgsVectorLayer(soil_sh)
```

In [ ]:
```
processing.run("gdal:rasterize",{'INPUT':soil_lyr, 'FIELD':'HSG_Index', 'BURN': None, 'UNITS':1, 'WIDTH': pixelSizeX, 'HEIGHT': pixelSizeY, 'EXTENT':ext_str, 'NODATA':0,
                                'OPTIONS':'', 'DATA_TYPE':5, 'INIT':None, 'INVERT':False, 'OUTPUT': os.path.join(output_folder, 'soil_raster.tif')})
print("Soil raster is ready!")
```

Calculating CN for each cell

We are going to use the _gdal:rastercalculator_ to calculate the CN value for each corresponding cell of soil and landuse raster. The look up table provides the CN value for each pair of soil and landuse value/category. The information in the look up table needs to be converted to a formula (string) that the raster calculator can use to create the CN raster.

The formula is as follows:
'100 *(A==1) + 57* logical_and(A==2, B==1) + 72 *logical_and(A==2,B==2) + 81* logical_and(A==2,B==3) + 86 *logical_and(A==2,B==4)* +
'*30* logical_and(A==3, B==1) + 58 *logical_and(A==3,B==2) + 71* logical_and(A==3,B==3) + 78 *logical_and(A==3,B==4)* +
'*67* logical_and(A==4, B==1) + *77 logical_and(A==4,B==2) + 83* logical_and(A==4,B==3) + 87* logical_and(A==4,B==4)'

Can you read the look up table from LookUp.csv file and parse the text to create the above string? Here A is landuse, B is soil.

In [ ]:
```
cn_table = pd.read_csv(cn_lookup)
```

In [ ]:
```
cn_table.head()
```

In [ ]:
```
cn_formula = ('100* (A==1) + 57* logical_and(A==2, B==1) + 72* logical_and(A==2,B==2) + 81* logical_and(A==2,B==3) + 86* logical_and(A==2,B==4) + 30* logical_and(A==3, B==1) + 58* logical_and(A==3,B==2) + 71* logical_and(A==3,B==3) + 78* logical_and(A==3,B==4) + 67* logical_and(A==4, B==1) + 77* logical_and(A==4,B==2) + 83* logical_and(A==4,B==3) + 87* logical_and(A==4,B==4)')
```

Finally, use the _gdal:rastercalculator_ to get the CN grid. It needs the following arguments (similar to reclassifybylayer tool)
'INPUT_A': full path to reclassified landuse raster,
'BAND_A':1,
'INPUT_B': full path to soil raster,
'BAND_B':1,
'FORMULA': CN formula defined above,
'NO_DATA':None,
'RTYPE':4,
'OPTIONS':'',
'OUTPUT': full path to output file, make sure the output file is a .tif file

In [ ]:
```
processing.run('gdal:rastercalculator',{'INPUT_A': os.path.join(output_folder,"lu_reclass_raster.tif"
), 'BAND_A':1, 'INPUT_B':os.path.join(output_folder, 'soil_raster.tif'),
                                       'BAND_B':1, 'FORMULA':cn_formula, 'NO_DATA':None, 'RTYPE':4,
'OPTIONS':'', 'OUTPUT': os.path.join(output_folder, 'cn_raster.tif')})
print('CN raster is ready!')
```

In [ ]: