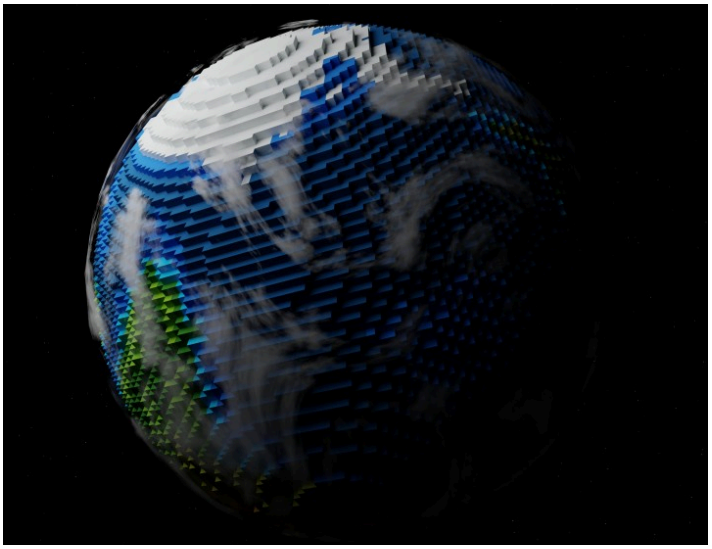

Photo by SIMON LEE on Unsplash

Merge Overlapping Rasters Using Python and GDAL VRT Pixel Functions

Chonghua Yin
Climate Scientist | Data Scientist | Developer

179 articles

January 29, 2024

Open Immersive Reader

Recently, I played with the global 30-m land-cover data of **GLC_FCS30-2020**. The data are grouped by 948 5X5 degrees regional tiles in the GeoTIFF format, which are named "GLCFCS30_E/W**N/S**.tif", where "E/W**N/S**" explains the longitude and latitude information of the upper left corner of each regional land-cover map. Further, each image contains a land-cover label band ranging from 0–255, and the invalid fill value is labeled as 0 and 250.

Issues

Four tiles were extracted to encompass the entire North Island in New Zealand. Nevertheless, the operation encountered an error upon attempting to read and merge them using xarray, a Python package. The simple debugging reveals the overlapping sections of these tiles (Figure 1 and Figure 2).

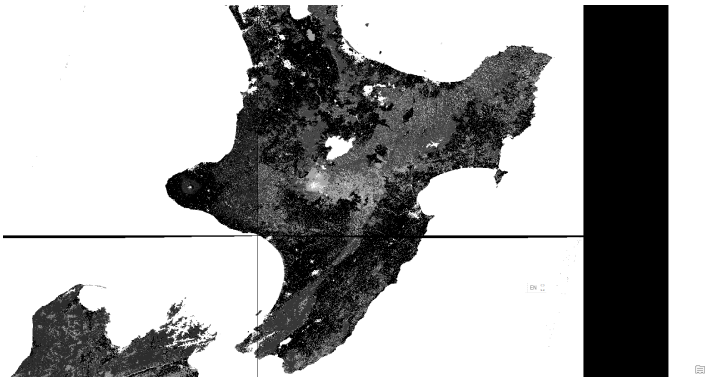

Figure 1

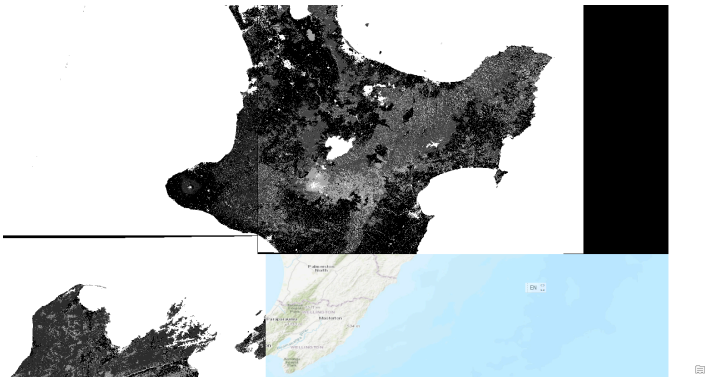

Figure 2

Depending on the data format and your chosen programming language, merging overlapping tiles or rasters can be performed through various software tools or programming libraries. This tutorial solves the issue using the ***GDAL VRT pixel functions***. One primary reason I chose the pixel function is that it supports embedded Python custom functions. In other words, we can use Python codes to perform more advanced and diverse calculations.

VRT -- GDAL Virtual Format

The **VRT driver** is a format driver for GDAL that allows a virtual GDAL dataset to be composed of other GDAL datasets with repositioning and algorithms potentially

applied and various kinds of metadata altered or added. VRT descriptions of datasets can be saved in an XML format normally given the extension. vrt.

In the VRT (Virtual Raster Table), a pixel function can be assigned to create a specialized band known as a 'derived' band. This band derives its pixel information from its source bands, and the pixel function generates the output raster. More information can be found at **https://gdal.org/drivers/raster/vrt.html**.

Solutions

In our scenario, merging the tiles could be accomplished using the numpy.max function. In other words, we selectively choose the maximum value within each overlapping area.

- ***Step 0***: define pixel functions

To dynamically add pixel functions into a vrt file, ***add_pixel_fn*** is defined following the guidance in a **similar** question.

```
import numpy as np
from osgeo import gdal

import warnings
warnings.filterwarnings('ignore')

def add_pixel_fn(filename: str) -> None:
    """inserts pixel-function into vrt file named
'filename'
    Args:
        filename (:obj:`string`): name of file, into
which the function will be inserted
    """

    header = """  <VRTRasterBand dataType="Byte"
band="1" subClass="VRTDerivedRasterBand">"""
    contents = """
    <PixelFunctionType>average</PixelFunctionType>

<PixelFunctionLanguage>Python</PixelFunctionLanguage
>
    <PixelFunctionCode><![CDATA[
from numba import jit
import numpy as np
@jit(nogil=True)
def max_jit(in_ar, out_ar, xoff, yoff, xsize, ysize,
raster_xsize, raster_ysize, buf_radius, gt):
```

```
    np.max(in_ar, axis = 0, out = out_ar)

def max(in_ar, out_ar, xoff, yoff, xsize, ysize,
raster_xsize,raster_ysize, buf_radius, gt):
    max_jit(in_ar, out_ar, xoff, yoff, xsize, ysize,
raster_xsize,raster_ysize, buf_radius, gt)
]]>
    </PixelFunctionCode>"""

    lines = open(filename, 'r').readlines()
    lines[3] = header
    lines.insert(4, contents)
    open(filename, 'w').write("".join(lines))
```

Before we merge tiles, we can check tile information using gdalinfo:

```
!gdalinfo data/GLC_FCS30_2020_E170S35.tif
```

The output looks like

```
...
Data axis to CRS axis mapping: 2,1
Origin = (169.899642213900421,-34.899285909892647)
Pixel Size = (0.000269494585236,-0.000269494585236)
Metadata:
  AREA_OR_POINT=Area
Image Structure Metadata:
  COMPRESSION=LZW
  INTERLEAVE=BAND
Corner Coordinates:
Upper Left  ( 169.8996422, -34.8992859)
(169d53'58.71"E, 34d53'57.43"S)
Lower Left  ( 169.8996422, -40.1296368)
(169d53'58.71"E, 40d 7'46.69"S)
Upper Right ( 175.0003662, -34.8992859) (175d 0'
1.32"E, 34d53'57.43"S)
Lower Right ( 175.0003662, -40.1296368) (175d 0'
1.32"E, 40d 7'46.69"S)
Center      ( 172.4500042, -37.5144614) (172d27'
0.02"E, 37d30'52.06"S)
Band 1 Block=256x256 Type=Byte, ColorInterp=Gray
...
```

- ***Step 1***: Create a vrt file to describe the selected four tiles.

```
out_name = "demo"
tifs = ['data/GLC_FCS30_2020_E170S35.tif',
```

```
data/GLC_FCS30_2020_E170S40.tif',
       "data/GLC_FCS30_2020_E175S35.tif",
"data/GLC_FCS30_2020_E175S40.tif"
       ]

vrt = gdal.BuildVRT(f'{out_name}.vrt', tifs)
```

- ***Step 2***: Add pixel functions

```
add_pixel_fn(f'{out_name}.vrt')
```

Now, part of the content of the created vrt file is presented as follows.

```
<VRTDataset rasterXSize="37480" rasterYSize="37965">
  <SRS dataAxisToSRSAxisMapping="2,1">GEOGCS["WGS
84",DATUM["WGS_1984",SPHEROID["WGS
84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],
AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHO
RITY["EPSG","8901"]],UNIT["degree",0.017453292519943
3,AUTHORITY["EPSG","9122"]],AXIS["Latitude",NORTH],A
XIS["Longitude",EAST],AUTHORITY["EPSG","4326"]]
</SRS>
  <GeoTransform>  1.6989964221390042e+02,
2.6949458523585642e-04,  0.0000000000000000e+00,
-3.4899285909892647e+01,  0.0000000000000000e+00,
-2.6949458523585642e-04</GeoTransform>
  <VRTRasterBand dataType="Byte" band="1"
subClass="VRTDerivedRasterBand">
    <PixelFunctionType>average</PixelFunctionType>

<PixelFunctionLanguage>Python</PixelFunctionLanguage
>
    <PixelFunctionCode><![CDATA[
from numba import jit
import numpy as np
@jit(nogil=True)
def max_jit(in_ar, out_ar, xoff, yoff, xsize, ysize,
raster_xsize, raster_ysize, buf_radius, gt):
    np.max(in_ar, axis = 0, out = out_ar)

def max(in_ar, out_ar, xoff, yoff, xsize, ysize,
raster_xsize,raster_ysize, buf_radius, gt):
    max_jit(in_ar, out_ar, xoff, yoff, xsize, ysize,
raster_xsize,raster_ysize, buf_radius, gt)
]]>
    </PixelFunctionCode>
<ColorInterp>Gray</ColorInterp>
    <SimpleSource>
      <SourceFilename
relativeToVRT="1">data/GLC_FCS30_2020_E170S35.tif</S
ourceFilename>
      <SourceBand>1</SourceBand>
```

```
      <SourceProperties RasterXSize="18927"
RasterYSize="19408" DataType="Byte" BlockXSize="256"
BlockYSize="256" />
      <SrcRect xOff="0" yOff="0" xSize="18927"
ySize="19408" />
      <DstRect xOff="0" yOff="0" xSize="18927"
ySize="19408" />
    </SimpleSource>
```

- ***Step 3***: Convert vrt file to GeoTIFF format

```
gdal.SetConfigOption('GDAL_VRT_ENABLE_PYTHON',
'YES')
ds = gdal.Open(f'{out_name}.vrt')
translateoptions =
gdal.TranslateOptions(gdal.ParseCommandLine("-ot
Byte -co COMPRESS=LZW -a_nodata 255 -co TILED=YES"
))
ds = gdal.Translate(f'{out_name}.tif', ds,
options=translateoptions)
```

Now the tiles have been merged, and the final output looks as follows. Isn't it neat and attractive?

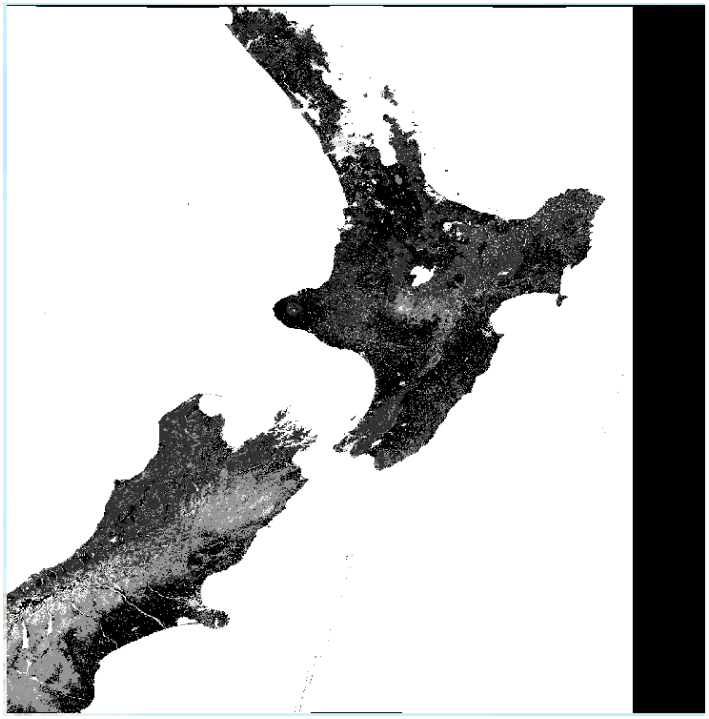

Figure 3

Summary and Discussions

When conducting spatial analysis, it is often necessary to use tiled data, and encountering overlapping tiles is a

common issue. As mentioned, there are many ways to merge overlapping tiles and rasters. However, pixel functions provide a flexible and powerful solution. It is worth trying.

References

https://gdal.org/drivers/raster/vrt.html

https://stackoverflow.com/questions/68760788/average-thousands-of-partially-overlapping-rasters-with-gdal

Liangyun, L., Xiao, Z., Xidong, C., Yuan, G., & Jun, M. (2020). GLC_FCS30-2020: Global Land Cover with Fine Classification System at 30m in 2020 (v1.2) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.4280923

Published by

Chonghua Yin
Climate Scientist | Data Scientist | Developer
Published • 2mo

179 articles

Just for fun during the weekend. Stay safe, healthy, and happy. A simple method to merge overlapping rasters or tiles using Python and GDAL VRT pixel functions.
#spatialanalysis #spatialdata #pythonprogramming