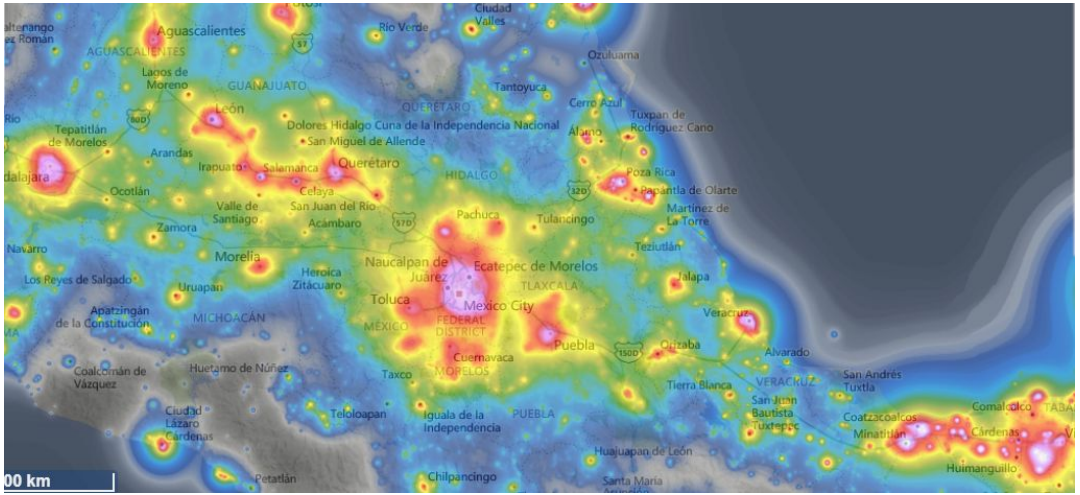
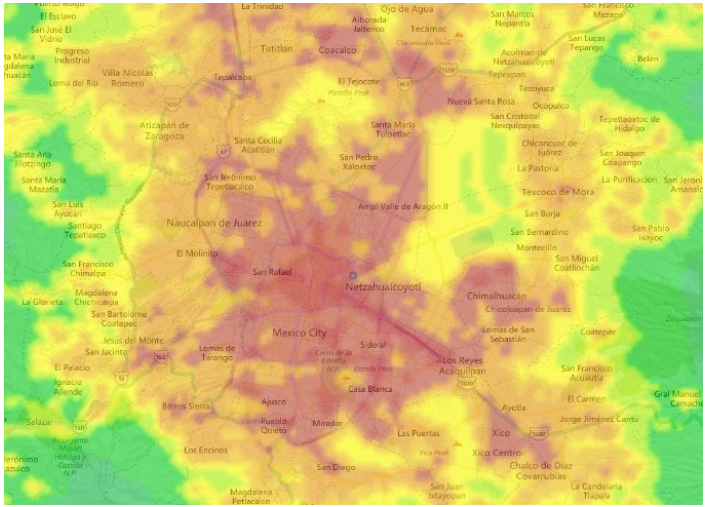
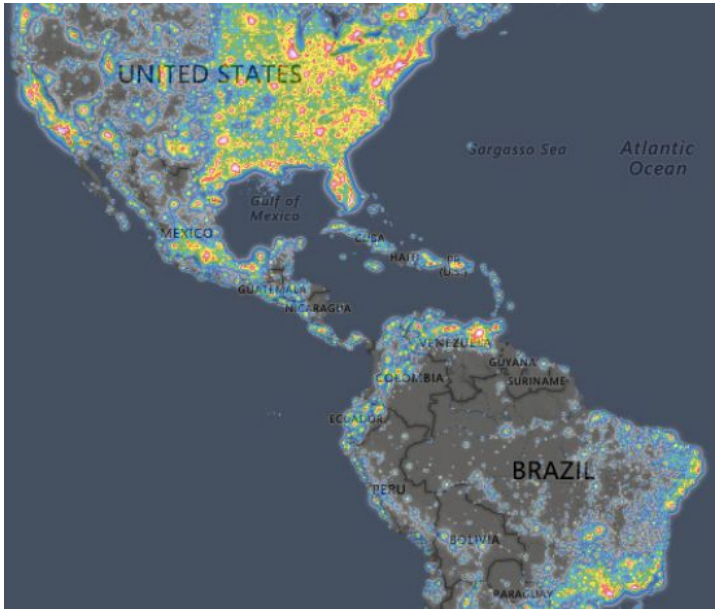
Light pollution

Extraction and Manipulation

Light pollution map







Why light pollution?

When doing a census and as a measure of wealth, the amount of light bulbs in a house is registered. More light bulbs might mean more income, bigger and brighter houses.

Light pollution has consequences, the more polluted a place, the less propensity to watch a starry night. This is another way to measure light pollution, as the capacity to watch stars.

Problem

We want light pollution measure for every lat-lon value. In order to be efficient, we must reduce the problem to those points in Mexico country area. Otherwise we would be reading useless data.

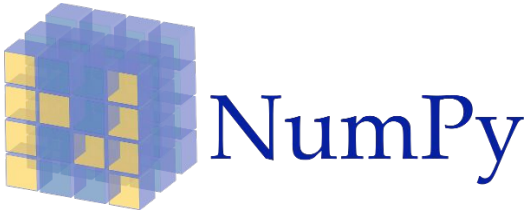
- First we need an accurate and reliable data source.
- Second, be able to read data from a GeoTIF (high resolution images), that is related with coordinates.
- Assign a light pollution score for every coordinate in the image.
- Assign a light pollution value to every lat-lon from targets.

Geopandas

Working with geospatial data easy in Python. It combines Pandas and shapely, providing geospatial data in pandas. Shapely allows us to manipulate and analyse planar geometric objects.

GDAL

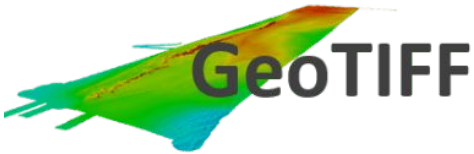
Geospatial Data Abstraction Library

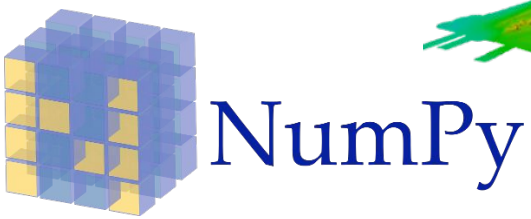


Programming and manipulating GDAL, manipulating raster data and OGR for geospatial vector data. Matching C++ classes aided with numpy for numeric approximation.

Rasterio

Library





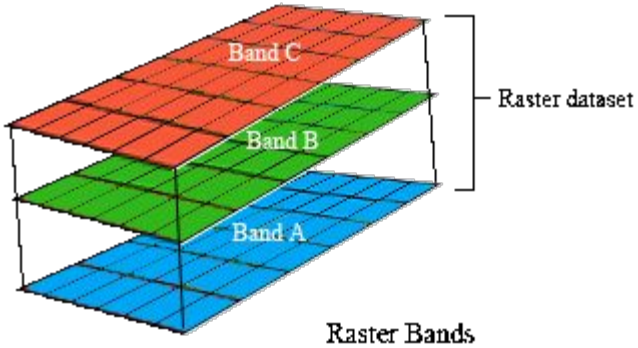
Geographic information systems use GeoTIFF and other formats to organize and store gridded raster datasets such as satellite imagery and terrain models. Rasterio reads and writes these formats and provides a Python API based on Numpy N-dimensional arrays and GeoJSON.

Global Radiance Calibrated Nighttime Lights

Data was extracted from: https://ngdc.noaa.gov/eog/dmsp/download_radcal.html. Though, last update was on 2014, still can use VIIRS information, with first four months of 2019.

NATIONAL CENTERS FOR ENVIRONMENTAL INFORMATION

Understanding GeoTIFF




```
import rasterio
dataset = rasterio.open("F16_20100111-20110731_rad_v4.avg_vis.tif")
dataset.name
#amount of bands in data
dataset.count #1
dataset.width #43201
dataset.height #16801
grayscale = dataset.read(1)
grayscale.shape #(16801, 43201)
```

The read method returns a numpy N-D array.

```
24  import gdal as gd
25  import numpy as np
26
27  def coord(file):
28      padfTransform = file.GetGeoTransform()
29      indices = np.indices(file.ReadAsArray().shape)
30      xp = padfTransform[0] + indices[1]*padfTransform[1] + indices[1]*padfTransform[2]
31      yp = padfTransform[3] + indices[0]*padfTransform[4] + indices[0]*padfTransform[5]
32      return xp,yp
33  file = gd.Open('Yourfile.tif') # A GeoTiff file
34  xp,yp = coord(file)
```

```
In [23]: padfTransform
Out[23]: (-180.00416666665, 0.0083333333, 0.0, 75.00416666665, 0.0, -0.0083333333)
```

```
In [25]: indices
Out[25]:
array([[[    0,     0,     0, ...,     0,     0,     0],
        [    1,     1,     1, ...,     1,     1,     1],
        [    2,     2,     2, ...,     2,     2,     2],
        ...,
        [16798, 16798, 16798, ..., 16798, 16798, 16798],
        [16799, 16799, 16799, ..., 16799, 16799, 16799],
        [16800, 16800, 16800, ..., 16800, 16800, 16800]],

       [[    0,     1,     2, ..., 43198, 43199, 43200],
        [    0,     1,     2, ..., 43198, 43199, 43200],
        [    0,     1,     2, ..., 43198, 43199, 43200],
        ...,
        [    0,     1,     2, ..., 43198, 43199, 43200],
        [    0,     1,     2, ..., 43198, 43199, 43200],
        [    0,     1,     2, ..., 43198, 43199, 43200]]])
```

Import gdal & numpy

Asignar un valor lat-lon a cada punto de la imagen. Por lo que se crea xp y yp con la información de latitud y longitud.

Output:

```
In [27]: xp
Out[27]:
array([[-180.00416667, -179.99583333, -179.9875    , ...,  179.97916523,
         179.98749856,  179.99583189],
       [-180.00416667, -179.99583333, -179.9875    , ...,  179.97916523,
         179.98749856,  179.99583189],
       [-180.00416667, -179.99583333, -179.9875    , ...,  179.97916523,
         179.98749856,  179.99583189],
       ...,
       [-180.00416667, -179.99583333, -179.9875    , ...,  179.97916523,
         179.98749856,  179.99583189],
       [-180.00416667, -179.99583333, -179.9875    , ...,  179.97916523,
         179.98749856,  179.99583189],
       [-180.00416667, -179.99583333, -179.9875    , ...,  179.97916523,
         179.98749856,  179.99583189]])
```

```
In [39]: yp
Out[39]:
array([[ 75.00416667,  75.00416667,  75.00416667, ...,  75.00416667,
         75.00416667,  75.00416667],
       [ 74.99583333,  74.99583333,  74.99583333, ...,  74.99583333,
         74.99583333,  74.99583333],
       [ 74.9875    ,  74.9875    ,  74.9875    , ...,  74.9875    ,
         74.9875    ,  74.9875    ],
       ...,
       [-64.97916611, -64.97916611, -64.97916611, ..., -64.97916611,
        -64.97916611, -64.97916611],
       [-64.98749944, -64.98749944, -64.98749944, ..., -64.98749944,
        -64.98749944, -64.98749944],
       [-64.99583277, -64.99583277, -64.99583277, ..., -64.99583277,
        -64.99583277, -64.99583277]])
```

```
In [28]: yp = padfTransform[3] + indices[0]*padfTransform[4] +
indices[0]*padfTransform[5]
Traceback (most recent call last):

  File "<ipython-input-28-a99664496806>", line 1, in <module>
    yp = padfTransform[3] + indices[0]*padfTransform[4] + indices[0]*padfTransform[5]

MemoryError: Unable to allocate array with shape (16801, 43201) and data type float64
```

Error of memory might occur. Try running again, otherwise manipulate on the cloud. Do not need to have all the matrix.

```
latitud = []
position = 0
for i in yp:
  if i[0] >= 12.211404 and i[0] <= 32.621072:
    latitud.append(position)
    position += 1
  else:
    position += 1
```

```
longitud = []
position = 0
for i in xp[0]:
  if i >= -119.850663 and i <= -80.355462:
    longitud.append(position)
    position += 1
  else:
    position += 1
```

```
lat_lon = []
for i in latitud:
  lat = yp[i][0]
  for e in longitud:
    lon = xp[0][e]
    position = [i, e]
    lat_lon.append([lat, lon, position])
```

```
index = 0
for i in lat_lon:
  i.append(lights[index])
  index += 1
```

Extract only the latitude and longitude space we want. Append latitude and longitude and assign its light value.

```
light_pollution = pd.DataFrame(lat_lon, columns = ['latitud', 'longitud',
                                                   'position_tif', 'light'])
```

```
In [11]: greyscale
Out[11]:
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

```
In [47]: light_pollution
Out[47]:
           latitud    longitud   position_tif  light
0         32.620834 -119.845834   [5086, 7219]    0.0
1         32.620834 -119.837500   [5086, 7220]    0.0
2         32.620834 -119.829167   [5086, 7221]    0.0
3         32.620834 -119.820834   [5086, 7222]    0.0
4         32.620834 -119.812500   [5086, 7223]    0.0
...             ...         ...            ...    ...
11610545  12.212500  -80.395834  [7535, 11953]    0.0
11610546  12.212500  -80.387500  [7535, 11954]    0.0
11610547  12.212500  -80.379167  [7535, 11955]    0.0
11610548  12.212500  -80.370834  [7535, 11956]    0.0
11610549  12.212500  -80.362500  [7535, 11957]    0.0

[11610550 rows x 4 columns]
```

```
[32.62083350284999, -111.65416694005, [5086, 8202], 0.0],
[32.62083350284999, -111.64583360675, [5086, 8203], 0.0],
[32.62083350284999, -111.63750027345, [5086, 8204], 0.0],
[32.62083350284999, -111.62916694015, [5086, 8205], 0.0],
[32.62083350284999, -111.62083360685, [5086, 8206], 0.0],
[32.62083350284999, -111.61250027355, [5086, 8207], 0.0],
[32.62083350284999, -111.60416694025, [5086, 8208], 0.0],
[32.62083350284999, -111.59583360695001, [5086, 8209], 0.0],
[32.62083350284999, -111.58750027365001, [5086, 8210], 0.0],
[32.62083350284999, -111.57916694035, [5086, 8211], 0.0],
[32.62083350284999, -111.57083360705, [5086, 8212], 0.0],
[32.62083350284999, -111.56250027375, [5086, 8213], 0.0],
[32.62083350284999, -111.55416694045, [5086, 8214], 0.0],
[32.62083350284999, -111.54583360715, [5086, 8215], 0.0],
[32.62083350284999, -111.53750027385, [5086, 8216], 0.0],
[32.62083350284999, -111.52916694055, [5086, 8217], 0.0],
[32.62083350284999, -111.52083360725, [5086, 8218], 0.0],
...]
```

BIG PROBLEM

When trying to join information based on lat-lon info to orders (for example) there are no coincidence. Therefore we have to use an intelligent imputation technique. Use light pollution infer to solve this.