

# MANIPULACIÓN DE IMÁGENES SATELITALES LANDSAT-8 CON PYTHON

Sensoramiento remoto  
2021-II

Profesor  
Dr. Ever Menacho  
Casimiro

Grupo 11  
Campos Sánchez, Kevin

# INTRODUCCIÓN

El procedimiento para manipular imágenes espaciales es muy similar en diferentes softwares. El tratamiento básico es considerarlas un capa ráster, lo que abre un gran abanico de procesos en Sistemas de Información Geográfica (SIG) (Olaya V., 2020). Estos nos permiten controlar gestionar, visualizar y analizar datos espaciales para solucionar problemas alrededor del mundo (Geoinnova, 2021).



Según FAO (1992), La preferencia de los usuarios depende de sus necesidades al utilizar el SIG, el costo de este y las capacidades del usuario. En este sentido, los lenguajes de programación son el complemento perfecto dado que permite la automatización de procesos, son de licencia libre y existe documentación suficiente para los usuarios en diferentes niveles de dominio.

En el caso de Python, existen librerías dedicados a la manipulación de capas rásters, en el presente informe trataremos dos de estas librerías: GDAL y Rasterio, así como geoprocесamiento básico y el potencial de la automatización de procesos al trabajar con imágenes satelitales.



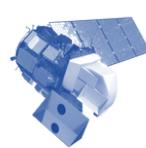
## OBJETIVOS

- 4 Automatizar procesamiento de imágenes Landsat-8

- 3 Recortar imágenes Landsat-8

- 2 Visualizar imágenes Landsat-8 con GDAL y Rasterio

- 1 Crear molde de recorte en QGIS



Landsat 8

# CREACIÓN DE MOLDE

## CUENCA LURÍN



Autoridad Nacional del Agua



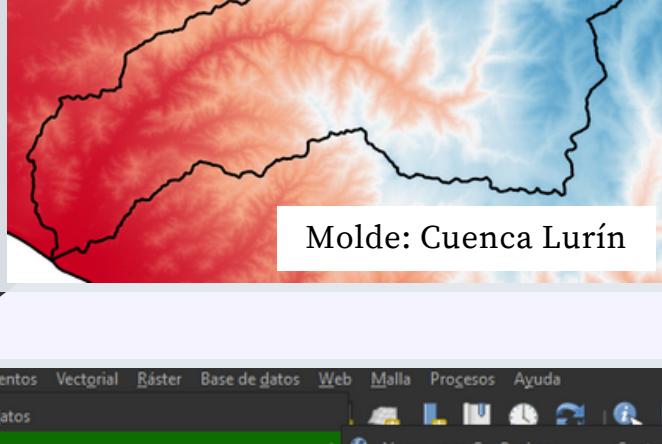
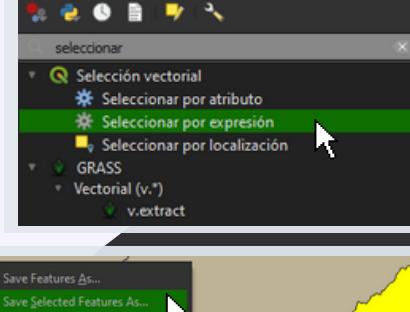
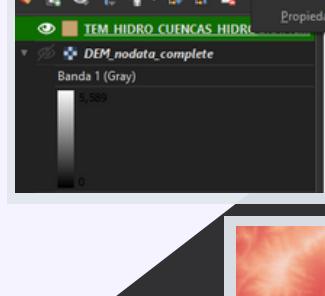
Cuencas hidrográficas del Perú

### Qgis 3.18

Es un Sistema de Información Geográfica (SIG) de código abierto, multiplataforma, con soporte de numerosos formatos y funcionalidades de datos vector, datos ráster y bases de datos

#### 1. Selección por expresión

"NOMBRE" = "Cuenca Lurín"

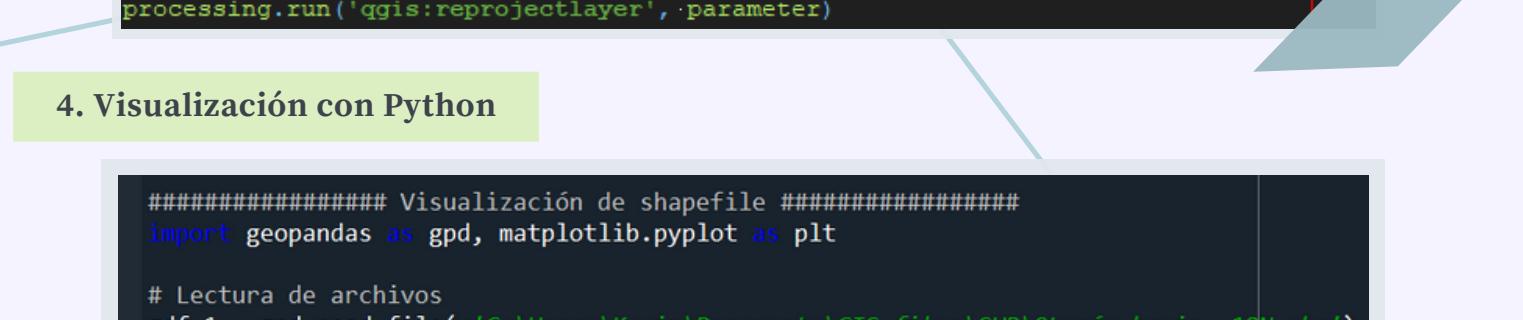


Molde: Cuenca Lurín

#### 2. Creación de área



#### 3. Reproyección a EPSG:32618



Recordar: proyección de las imágenes es EPSG:32618 (UTM 18N)

El shapefile de la cuenca está originalmente en EPSG:32718 (UTM 18S), por lo que se debe reproyectar.

```
import processing

parameter = {'INPUT': r'C:\Users\Kevin\Documents\GIS_files\SHP\Lurin_basin_18S.shp',
             'TARGET_CRS': 'EPSG:32618',
             'OUTPUT': r'C:\Users\Kevin\Documents\GIS_files\SHP\Lurin_basin_18N.shp'}

processing.run('qgis:reprojectlayer', parameter)
```

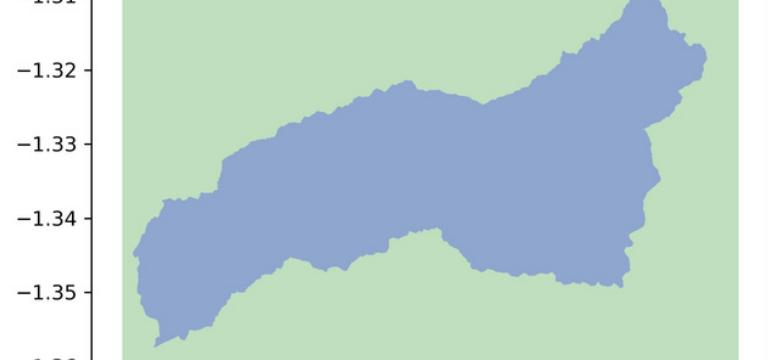
#### 4. Visualización con Python

```
#####
# Visualización de shapefile #####
import geopandas as gpd, matplotlib.pyplot as plt

# Lectura de archivos
gdf_1 = gpd.read_file(r'C:\Users\Kevin\Documents\GIS_files\SHP\QLurín_basin_18N.shp')
gdf_2 = gpd.read_file(r'C:\Users\Kevin\Documents\GIS_files\SHP\Lurin_basin_18S.shp')

# Se almacena como geodataframe
print(gdf_1)

fig, ax = plt.subplots(1,1, figsize = (6,6))
gdf_1.plot(ax = ax, color = 'green', alpha = 0.25)
gdf_2.plot(ax = ax, color = 'blue', alpha = 0.25)
ax.set_title('Shapefile - Cuenca Lurín', fontsize = 14, color = 'darkblue')
plt.savefig('Lab_3\shp_plot.jpg', dpi = 300)
```



Las coordenadas corresponden al sistema de referencia EPSG:32618 (UTM 18N)

#### Referencias recomendadas:

- Guía rápida de geopandas - visualización: <https://cutt.ly/7UFydpf>
- Documentación geopandas plot: <https://cutt.ly/wUFypvf>

Scripts disponibles en: <https://github.com/Kevcampos/UNALM-meteorology>



# VISUALIZACIÓN DE IMÁGENES LANDSAT-8

Segundo, visualizar la imagen Landsat 8 sin cortar. Mediante las librerías GDAL y Rasterio en el IDE Spyder 5.

## 1. Mediante librería GDAL

```
##### Visualización mediante GDAL #####
import gdal, os, matplotlib.pyplot as plt, numpy as np

os.chdir(r'C:\Users\Kevin\Desktop\UNALM\OctCiclo\Senso_rem')

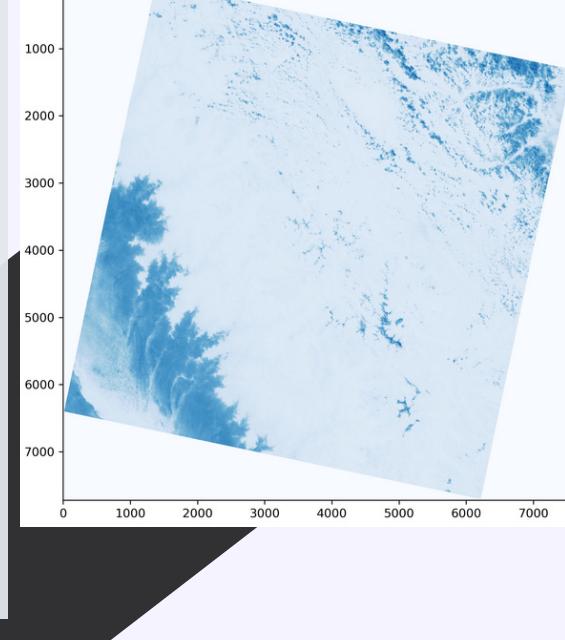
# Cargar imagen: Banda 2 (Blue)
img = 'Image\LC08_L1TP_007068_20170424_20170502_01_T1_B2.TIF'
open_gdal = gdal.Open(img)

img_open = open_gdal.GetRasterBand(1) # Cargar banda
img_L8 = img_open.ReadAsArray()      # Pasar a Array

# Normalización
def Normalize(a):
    a_2 = np.nanpercentile(a, 2); a_98 = np.nanpercentile(a, 98)
    return (a - a_2)/(a_98 - a_2)

img_L8 = Normalize(img_L8)

# Gráfica
plt.figure(figsize = (8,8))
plt.imshow(img_L8, cmap = 'Blues')
plt.title('Banda 2: Blues', fontsize = 16, color = 'darkblue')
plt.savefig('Lab_3\img_gdal.jpg', dpi = 300)
```



## 2. Mediante librería Rasterio

```
##### Visualización mediante Rasterio #####
import matplotlib.pyplot as plt, os, rasterio, numpy as np
os.chdir(r'C:\Users\Kevin\Desktop\UNALM\OctCiclo\Senso_rem')
images = glob.glob('Image\*[2-4].tif') # Carga de imágenes

# Abrir imágenes
R_B2 = rasterio.open(images[0])
R_B3 = rasterio.open(images[1]); R_B4 = rasterio.open(images[2])

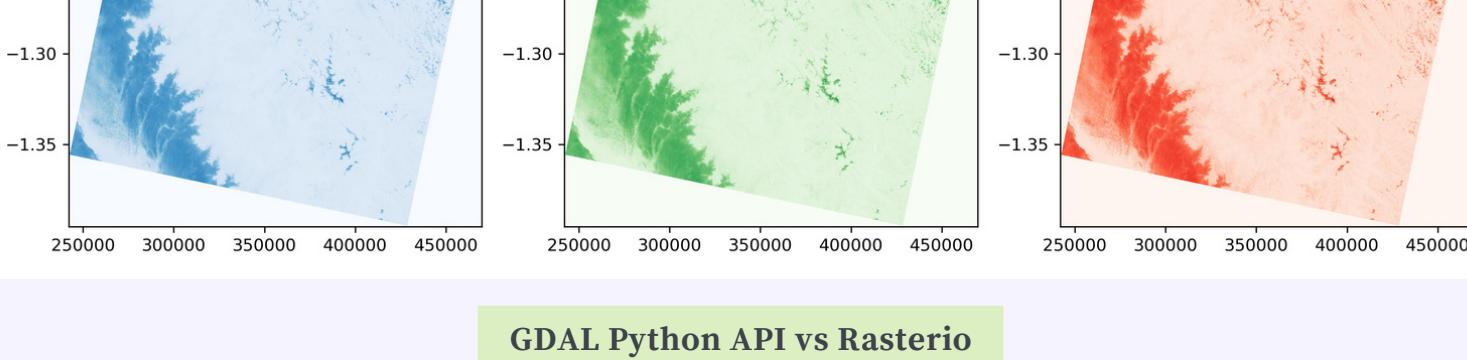
# Transformar a arrays
Array_R_B2 = R_B2.read(1); Array_R_B3 = R_B3.read(1); Array_R_B4 = R_B4.read(1)

# Normalización
def Normalize(a):
    a_2 = np.nanpercentile(a, 2); a_98 = np.nanpercentile(a, 98)
    return (a - a_2)/(a_98 - a_2)

Array_R_B2 = Normalize(Array_R_B2)
Array_R_B3 = Normalize(Array_R_B3)
Array_R_B4 = Normalize(Array_R_B4)

# Visualización
fig, axes = plt.subplots(1,3, figsize = (15,10))
bands = [Array_R_B2, Array_R_B3, Array_R_B4]
color = ['Blues', 'Greens', 'Reds']

from rasterio.plot import show
for i in range(3):
    show(bands[i], transform=R_B2.transform, cmap = color[i],
         title= f'Banda {i+2}: {color[i]} - Landsat 8', ax = axes[i])
plt.savefig('Lab_3\plot_rasterio.jpg', dpi = 300)
```



## GDAL Python API vs Rasterio

Ambos están basados en GDAL, el cual es una librería de código abierto basado en C++, enfocada al manejo de datos ráster y vectoriales en SIG.

GDAL Python API es la adaptación a Python de GDAL, estando constituido por dos librerías: GDAL y OSG. Debido a que es una adaptación de C++, algunos usuarios reportan problemas de incompatibilidad con librerías propias de Python.

Rasterio en cambio, se enfoca en utilizar una sintaxis "pythonista", en el que aprovecha las virtudes de Python para realizar los mismos procesos. Es por ello que para muchos, Rasterio es una opción más cómoda y accesible que GDAL Python API.

Sin embargo, tal como postula FAO (1992), cada usuario determina sus preferencias según sus necesidades, por lo que ambas son potentes librerías para el manejo de capas rásters.

Referencia recomendada:

- Documentación gdal Python API: <https://cutt.ly/oUZe0gN>
- Manejo básico con gdal Python API: <https://cutt.ly/rUZrrrw>
- Geoprocessing with GDAL/OGR Python API: <https://cutt.ly/NUZrktj>

-----

- Guía rápida de Rasterio: <https://cutt.ly/pUnvQL3>

- Visualización con rasterio: <https://cutt.ly/PUnY6w4>

Scripts disponibles en: <https://github.com/Kevcamposs/UNALM-meteorology>



# RECORTAR IMÁGENES

## LANDSAT-8

Cuarto, recortar la imagen Landsat 8 a partir del shapefile y sus coordenadas creadas inicialmente.

### 0. Cargar librerías

```
import os, numpy as np
from osgeo import gdal

path = r'C:\Users\Kevin\Desktop\UNALM\OctCiclo\Senso_rem'
os.chdir(path)

# Carga de imagen y transformación a array
img = 'Image\LC08_L1TP_007068_20170424_20170502_01_T1_B2.TIF'
img_open = gdal.Open(img)
```

### 1. Mediante coordenadas

#### Información del proveedor

Nombre	QLurín_basin_18N
Ruta	C:\Users\Kevin\Documents\GIS_files\SHP\QLurín_basin_18N.shp
Almacenamiento	ESRI Shapefile
Comentario	
Codificación	UTF-8
Geometría	Polygon (MultiPolygon)
SRC	EPSG:32618 - WGS 84 / UTM zone 18N - Proyectado
Extensión	288651.0196189094567671,-1360643.7017785683274269 : 371836.6587493440601975,-1307771.4735176989343017
Unidad	metros
Número de objetos	1

```
##### Clip with points - GDAL #####
# Carga de points
points = [288651.0196189094567671,      # X mínima
          -1307771.4735176989343017,      # Y mínima
          371836.6587493440601975,        # X máxima
          -1360643.7017785683274269]    # Y máxima

# Clip
img_clip = gdal.Translate('Lab_3\Clip_points.TIF', # Carpeta destino
                           img_open,                  # Array a cortar
                           projWin = points)         # Shapefile
```

### 2. Mediante shapefile

```
##### Clip with mask - GDAL #####
# Carga de Shapefile
shp = r'C:\Users\Kevin\Documents\GIS_files\SHP\Lurin_basin_18N.shp'

# Clip
img_clip = gdal.Warp('Lab_3\Clip_mask.TIF', # Carpeta destino
                      img_open,                  # Array a cortar
                      cutlineDSName = shp,       # Shapefile
                      cropToCutline = True,     # Activar clip
                      dstNodata= np.nan)        # Completar valores
```

### 3. Visualizar recorte

```
##### Visualización del recorte #####
import matplotlib.pyplot as plt, geopandas as gpd, rasterio as rio, numpy as np

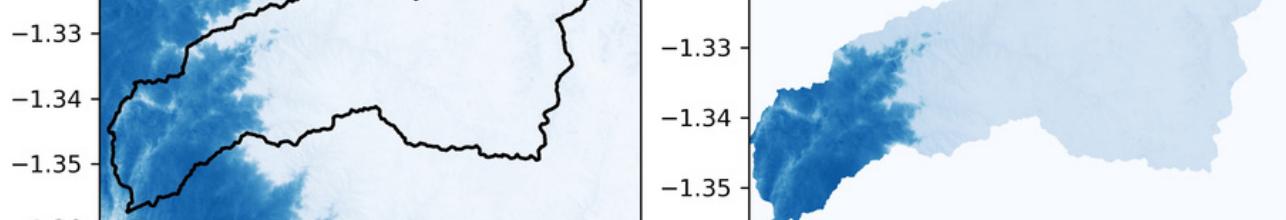
# Normalización
def Normalize(a):
    a_2 = np.nanpercentile(a, 2); a_98 = np.nanpercentile(a, 98)
    return (a - a_2)/(a_98 - a_2)

# Cargar ráster (imagen)
clip = rio.open('Lab_3\Clip_points.TIF'); clip2 = rio.open('Lab_3\Clip_mask.TIF')
clip_transform = clip.transform
clip = Normalize(clip.read(1)); clip2 = Normalize(clip2.read(1))

# Cargar shapefile - cuenca Lurín
basin = gpd.read_file(r'C:\Users\Kevin\Documents\GIS_files\SHP\Lurin_basin_18N.shp')

# Visualización
fig, (ax1, ax2) = plt.subplots(1,2, figsize = (9,6.5))
plt.suptitle('Cuenca Lurín: Banda 2 (Blues) - Landsat 8', x = .5, y = .75, fontsize = 18)
rio.plot.show(clip, transform = clip_transform, cmap = 'Blues',
              title= 'clip with points', ax = ax1)
basin.boundary.plot(ax = ax1, color = 'black', markersize=4)
rio.plot.show(clip2, transform = clip_transform, cmap = 'Blues',
              title= 'clip with mask', ax = ax2)
plt.savefig('Lab_3\img_clip.jpg', dpi = 300)
```

### Cuenca Lurín: Banda 2 (Blues) - Landsat 8



Referencia recomendada:

- Manejo de ráster con GDAL Python API: <https://cutt.ly/7UZtCQk>
- Guía rápida de geopandas - visualización: <https://cutt.ly/7UFydpf>

Scripts disponibles en: <https://github.com/Kevcampos/UNALM-meteorology>



# AUTOMARIZACIÓN DE PROCESOS

Por último, se crea dos módulos para el procesamiento de imágenes Landsat 8. El primero contiene los procesos de recorte de la imagen y el segundo realiza combinación de bandas.

## Módulo 1: img\_L8.py

```
class img_L8:  
    def __init__(self, img, normalize = True):  
        self.img = img  
        import rasterio as rio, numpy as np  
  
        self.img_ar = rio.open(self.img).read(1)  
        self.crs = rio.open(self.img).transform  
  
        if normalize == True:  
            a_2 = np.nanpercentile(self.img_ar, 2)  
            a_98 = np.nanpercentile(self.img_ar, 98)  
            self.img_ar = (self.img_ar - a_2)/(a_98 - a_2)  
  
        return print('La imagen se cargó correctamente.')  
  
    def plot(self, title = 'Landsat 8', transform = True, cmap = 'gray'):  
        from rasterio.plot import show  
        if transform == True:  
            show(self.img_ar, transform = self.crs, title = title, cmap = cmap)  
        else:  
            show(self.img_ar, title = title, cmap = cmap)  
  
    def clip_shp(self, shp, output):  
        from osgeo import gdal  
        import numpy as np  
        gdal.Warp(output, self.img, cutlineDSName = shp, cropToCutline = True, dstNodata= np.nan)  
        return print('Proceso finalizado.')  
  
    def clip_points(self, points, output):  
        from osgeo import gdal  
        gdal.Translate(output, self.img, projWin = points)  
        return print('Proceso finalizado.')
```

Lectura de la imagen

Gráfico básico

Recorte de imagen

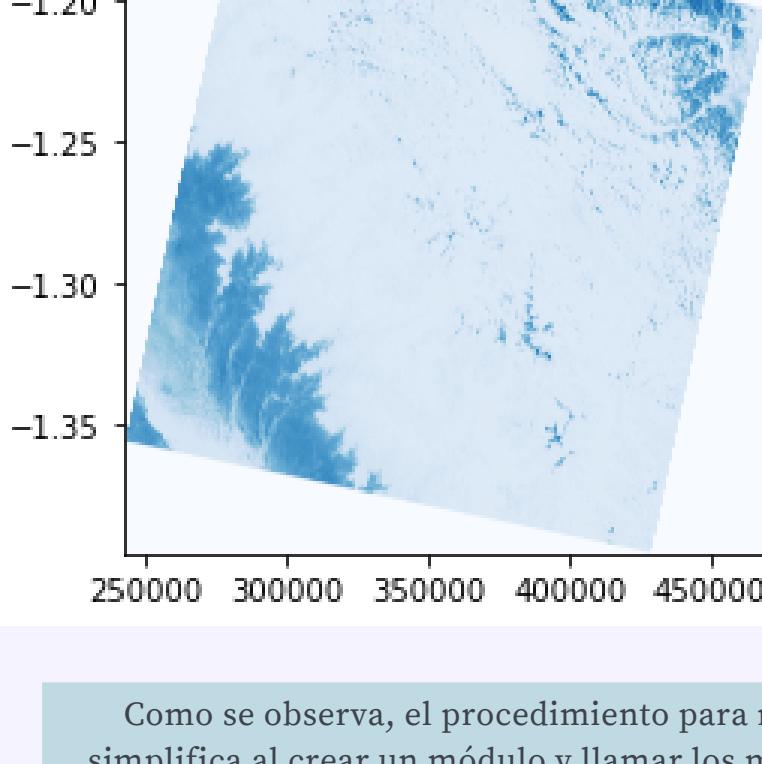
## Ejemplo 1: img\_18\_test.py

```
import os  
from img_L8 import img_L8  
  
# Definir carpeta de trabajo  
os.chdir(r'C:\Users\Kevin\Desktop\UNALM\OctCiclo\Senso_rem')  
  
# Imagen y creación de objeto  
image = r'Image\LC08_L1TP_007068_20170424_20170502_01_T1_B2.TIF'  
Banda_2 = img_L8(image)  
  
# Clip con máscara  
shp = r'C:\Users\Kevin\Documents\GIS_files\SHP\QLurín_basin__18N.shp'  
Banda_2.clip_shp(shp, 'test.TIF')  
  
# Clip con puntos  
points = [288651.0196189094567671, # X mínima  
          -1307771.4735176989343017, # Y mínima  
          371836.6587493440601975, # X máxima  
          -1360643.7017785683274269] # Y máxima  
  
Banda_2.clip_points(points, 'test2.TIF')  
  
# Plot básico  
Banda_2.plot(title = 'Landsat 8 - Banda 2 (Blue)',  
              cmap = 'Blues')
```

Cargamos el módulo

Creamos el objeto

Realizamos el corte



Se creó el objeto Banda\_2, el cual mediante los métodos clip\_points y plot, se realizó un recorte a base de coordenadas.

B2\_Lurin

B3\_Lurin

B4\_Lurin

B5\_Lurin

B6\_Lurin

Recortes creados al crear nuevos objetos con otras bandas

Como se observa, el procedimiento para recortar imágenes satelitales se simplifica al crear un módulo y llamar los métodos correspondientes. Esta es la ventaja que brinda los lenguajes como Python al automatizar los procesos.

Referencia recomendada:

- Guía rápida de Rasterio: <https://cutt.ly/pUnvQL3>

- Características de Jupyter notebook: <https://cutt.ly/MUnvSbe>

Scripts disponibles en: <https://github.com/Kevcampos/UNALM-meteorology>



## Módulo 2: composite\_L8.py

```
class composite_L8:
    def __init__(self, img, normalize = True):
        self.img = img
        import rasterio as rio, numpy as np

        self.img_ar1 = rio.open(self.img[0]).read(1)
        self.img_ar2 = rio.open(self.img[1]).read(1)
        self.img_ar3 = rio.open(self.img[2]).read(1) Lectura de la imagen

        self.crs = rio.open(self.img[0]).transform

    if normalize == True:
        def Normalize(a):
            a_2 = np.nanpercentile(a, 2); a_98 = np.nanpercentile(a, 98)
            return (a - a_2)/(a_98 - a_2)

        self.img_ar1 = Normalize(self.img_ar1)
        self.img_ar2 = Normalize(self.img_ar2)
        self.img_ar3 = Normalize(self.img_ar3) Normalización

    return print('Las imágenes se cargaron correctamente.')

def composite(self):
    import numpy as np
    self.com = np.stack((self.img_ar1, self.img_ar2, self.img_ar3)) Combinación de bandas

    return print('Composite completado.')

def plot(self, title = 'Landsat 8 - composite', save = True,
        output = 'com.TIF', shp = None):
    import matplotlib.pyplot as plt, rasterio as rio, geopandas as gpd

    fig, ax = plt.subplots(1,1, figsize = (10,10)) Gráfico

    rio.plot.show(self.com, transform = self.crs, title= title, ax = ax)
    if shp != None:
        shape = gpd.read_file(shp)
        shape.boundary.plot(ax = ax, color = 'black', markersize=4.5)
    if save == True:
        plt.savefig(output, dpi = 300)
```

## Ejemplo 2: composite\_L8\_test.py

```
import os
from composite_L8 import composite_L8

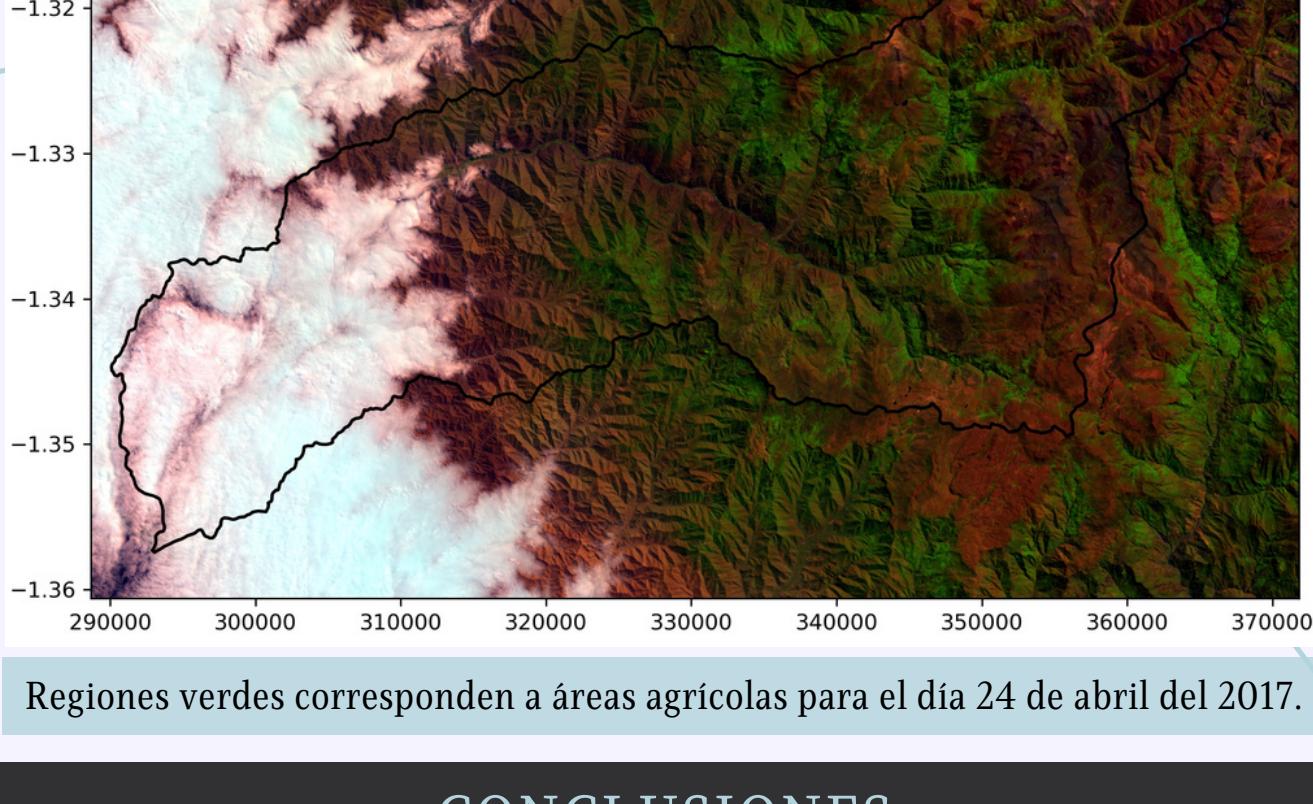
# Definir carpeta de trabajo
os.chdir(r'C:\Users\Kevin\Desktop\UNALM\OctCiclo\Senso_rem\Lab_3')

shp = r'C:\Users\Kevin\Documents\GIS_files\SHP\Lurin_basin_18N.shp'

# Carga de imágenes
b2 = r'B2_Lurin.TIF'; b5 = r'B5_Lurin.TIF'; b6 = r'B6_Lurin.TIF'

images = [b6, b5, b2] # Combinación 6-5-2: zonas agrícolas

Agricultura = composite_L8(images)
Agricultura.composite()
Agricultura.plot(title = 'Cuenca Lurín - Zonas agrícolas (6-5-2)',
                 output = 'C652_Lurín.jpg', shp = shp)
```



Regiones verdes corresponden a áreas agrícolas para el día 24 de abril del 2017.

## CONCLUSIONES

Python, a través de la librerías Rasterio y GDAL Python API, complementado con la Programación Orientada a Objetos para crear módulos que automatizan procesos, son un potente complemento y alternativa a los SIG al manipular imágenes satelitales Landsat-8.

## REFERENCIAS BIBLIOGRÁFICAS

FAO (1992). FUNCIONAMIENTO Y USOS DE LOS SISTEMAS DE INFORMACION GEOGRAFICA. En Food and Agriculture Organization. Los sistemas de información geográfica y la telepercepción en la pesca continental y la acuicultura. <https://www.fao.org/3/t0446s/T0446S08.htm>

Geoinnova (11 de agosto del 2021). ¿Qué es un SIG, GIS o Sistema de Información Geográfica?. Geoinnova. <https://geoinnova.org/blog-territorio/que-es-un-sig-gis-o-sistema-de-informacion-geografica/>

Olaya V. (2020). Procesado de imágenes. En Olaya V. Sistemas de Información Geográfica. Github. <https://olaya.github.io/libro-sig/chapters/Imagenes.html>