clase-4-operaciones-raster-1

April 30, 2024

EDUCACION.EARTH - CURSO TELEDETECCION CON PYTHON

CLASE 4: OPERACIONES CON DATOS RASTER

0.0.1 Introducción

En esta clase vamos a ver operaciones básicas a nivel ráster. Son variadas las posibilidades, vamos a concentrarnos en algunas que ilustran casos interesantes en lo relacionado al entrenamiento de modelos con redes neuronales.

Se sugiere ver la documentación de `gdal_translate` y de `gdal_warp` para ver mayor el panorama completo.

En el notebook de este link pueden encontrar un apéndice para mayor detalle de los parámetros utilizados.

Vamos a trabajar con una imagen Sentinel-2 con las bandas RGB para la provincia de Chaco (Argentina), donde se evidencia un proceso marcado de desmonte en los últimos años. La imagen es del año 2021.

[2]:
```
!mkdir -p ./data/
!gdown 1B2L0wW5or60jfrFJ2UHXI_tgP4lEFtat -O ./data/
!gdown 1gALtw3QspXwBZRZ1PPHmciJAbSJp4sAb -O ./data/
```

```
Downloading…
From: https://drive.google.com/uc?id=1B2L0wW5or60jfrFJ2UHXI_tgP4lEFtat
To: /content/data/s2_rgb.tif
100% 40.5M/40.5M [00:00<00:00, 150MB/s]
Downloading…
From: https://drive.google.com/uc?id=1gALtw3QspXwBZRZ1PPHmciJAbSJp4sAb
To: /content/data/aoi.geojson
100% 495/495 [00:00<00:00, 1.56MB/s]
```

[4]:
```
!ls ./data/
```

```
aoi.geojson  s2_rgb.tif
```

0.0.2 1. Reproyección del Raster:

La reproyección es una operación esencial en teledetección que permite cambiar el sistema de coordenadas de un raster para adecuarlo a un sistema de referencia espacial específico. En este ejemplo, utilizaremos las bibliotecas rasterio y GDAL para reproyectar un raster a un sistema de coordenadas diferente, como el EPSG:32721 para WGS 84.

Con rasterio

[5]:
```
!pip install rasterio
```

```
Collecting rasterio
  Downloading rasterio-1.3.10-cp310-cp310-manylinux2014_x86_64.whl (21.5 MB)
                           21.5/21.5 MB
43.9 MB/s eta 0:00:00
Collecting affine (from rasterio)
  Downloading affine-2.4.0-py3-none-any.whl (15 kB)
Requirement already satisfied: attrs in /usr/local/lib/python3.10/dist-packages
(from rasterio) (23.2.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-
packages (from rasterio) (2024.2.2)
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.10/dist-
packages (from rasterio) (8.1.7)
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.10/dist-
packages (from rasterio) (0.7.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from rasterio) (1.25.2)
Collecting snuggs>=1.4.1 (from rasterio)
  Downloading snuggs-1.4.7-py3-none-any.whl (5.4 kB)
Requirement already satisfied: click-plugins in /usr/local/lib/python3.10/dist-
packages (from rasterio) (1.1.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-
packages (from rasterio) (67.7.2)
Requirement already satisfied: pyparsing>=2.1.6 in
/usr/local/lib/python3.10/dist-packages (from snuggs>=1.4.1->rasterio) (3.1.2)
Installing collected packages: snuggs, affine, rasterio
Successfully installed affine-2.4.0 rasterio-1.3.10 snuggs-1.4.7
```

[6]:
```
import rasterio
from rasterio.warp import calculate_default_transform, reproject

# Abrir el raster original
with rasterio.open('./data/s2_rgb.tif') as src:
    # Definir el sistema de coordenadas deseado (EPSG:4326 para WGS 84)
    dst_crs = 'EPSG:32721'

    # Calcular la transformación y dimensiones del nuevo raster reproyectado
    transform, width, height = calculate_default_transform(src.crs, dst_crs,␣
 ↪src.width, src.height, *src.bounds)
```

```
    # Configurar las opciones para el nuevo raster reproyectado
    kwargs = src.meta.copy()
    kwargs.update({
        'crs': dst_crs,
        'transform': transform,
        'width': width,
        'height': height
    })

    # Crear un nuevo raster reproyectado
    with rasterio.open('./data/s2_rgb_reproj.tif', 'w', **kwargs) as dst:
        # Reproyectar la data al nuevo raster
        reproject(
            source=rasterio.band(src, 1),
            destination=rasterio.band(dst, 1),
            src_transform=src.transform,
            src_crs=src.crs,
            dst_transform=transform,
            dst_crs=dst_crs,
            resampling=rasterio.enums.Resampling.bilinear
        )
```

Con GDAL

[7]:
```
from osgeo import gdal, osr

# Abrir el raster original
src_ds = gdal.Open('./data/s2_rgb.tif')

# Definir el sistema de coordenadas deseado (EPSG:4326 para WGS 84)
dst_crs = 'EPSG:32721'

# Crear un objeto SpatialReference para el sistema de coordenadas de destino
dst_srs = osr.SpatialReference()
dst_srs.SetFromUserInput(dst_crs)

# Crear un nuevo dataset para el raster reproyectado
dst_ds = gdal.Warp('./data/s2_rgb_reproj_gdal.tif', src_ds, dstSRS=dst_srs,␣
 ↪resampleAlg=gdal.GRA_Bilinear)

# Cerrar los datasets
src_ds = None
dst_ds = None
```

0.0.3 2. Guardar en JPG en lugar de TIFF:

El formato en el que almacenamos nuestros datos raster puede tener un impacto significativo en el tamaño del archivo y la calidad de la imagen. En este caso, mostraremos cómo guardar un raster en formato JPEG en lugar de TIFF utilizando las bibliotecas rasterio y GDAL.

Con rasterio

[8]:
```
src.close()
dst.close()
```

[9]:
```
import rasterio

# Abrir el raster original
with rasterio.open('./data/s2_rgb.tif') as src:
    # Configurar opciones de compresión (ejemplo: compresión JPEG con calidad␣
 ↪del 90%)
    options = src.profile.copy()
    options.update({
        'driver': 'JPEG',
        'tiled': True,
        'quality': 90
    })

    # Crear un nuevo raster comprimido
    with rasterio.open('./data/s2_rgb.jpg', 'w', **options) as dst:
        # Copiar la data del raster original al nuevo raster comprimido
        dst.write(src.read())
```

```
WARNING:rasterio._env:CPLE_NotSupported in driver JPEG does not support creation
option BLOCKYSIZE
WARNING:rasterio._env:CPLE_NotSupported in driver JPEG does not support creation
option TILED
WARNING:rasterio._env:CPLE_NotSupported in driver JPEG does not support creation
option COMPRESS
WARNING:rasterio._env:CPLE_NotSupported in driver JPEG does not support creation
option INTERLEAVE
WARNING:rasterio._env:CPLE_AppDefined in 4-band JPEGs will be interpreted on
reading as in CMYK colorspace
```

Con GDAL

[10]:
```
from osgeo import gdal

# Abrir el raster original
src_ds = gdal.Open('./data/s2_rgb.tif')

# Configurar opciones de compresión para el nuevo formato (ejemplo: JPEG con␣
 ↪calidad del 90%)
```

```
options = gdal.TranslateOptions(format='JPEG', creationOptions=['QUALITY=90'])

# Crear un nuevo archivo en formato JPEG
gdal.Translate('./data/s2_rgb_gdal.tif', src_ds, options=options)

# Cerrar el dataset original
src_ds = None
```

0.0.4 3. Comprimir:

La compresión de rasters es una práctica común para reducir el tamaño de los archivos y ahorrar espacio de almacenamiento. A continuación, presentamos cómo aplicar compresión a un raster utilizando GDAL.

Con GDAL

[11]:
```
from osgeo import gdal

# Abrir el raster original
src_ds = gdal.Open('./data/s2_rgb.tif')

# Configurar opciones de compresión para el nuevo formato (ejemplo: JPEG con␣
 ↪calidad del 90%)
options = '-co COMPRESS=JPEG -co JPEG_QUALITY=90'

# Crear un nuevo archivo comprimido
gdal.Translate('./data/s2_rgb_compress_gdal.tif', src_ds, options=options)

# Cerrar el dataset original
src_ds = None
```

0.0.5 4. Subconjunto de bandas

Veamos como extraer bandas específicas de la imagen.

[12]:
```
import rasterio
from rasterio.enums import Resampling

banda_deseada = 2 #la verde

with rasterio.open('./data/s2_rgb.tif') as src:
    # Obtener las bandas del raster
    bandas = src.read()

    # Seleccionar la banda deseada (en este caso, la banda 1)
    banda_seleccionada = bandas[banda_deseada - 1]

    # Obtener el perfil del raster original
```

```
    perfil = src.profile

    # Actualizar el perfil para tener solo una banda
    perfil.update(count=1)

    # Guardar la banda seleccionada en un nuevo raster
    with rasterio.open('./data/s2_green.tif', 'w', **perfil) as dst:
        dst.write(banda_seleccionada, 1)
```

Inspeccionemos el archivo resultante con GDAL. El módulo `gdal.Info` es particularmente útil para explorar las propiedades geoespaciales de imágenes, como proyecciones, resoluciones, estadísticas de bandas y otros metadatos asociados a los datos raster.

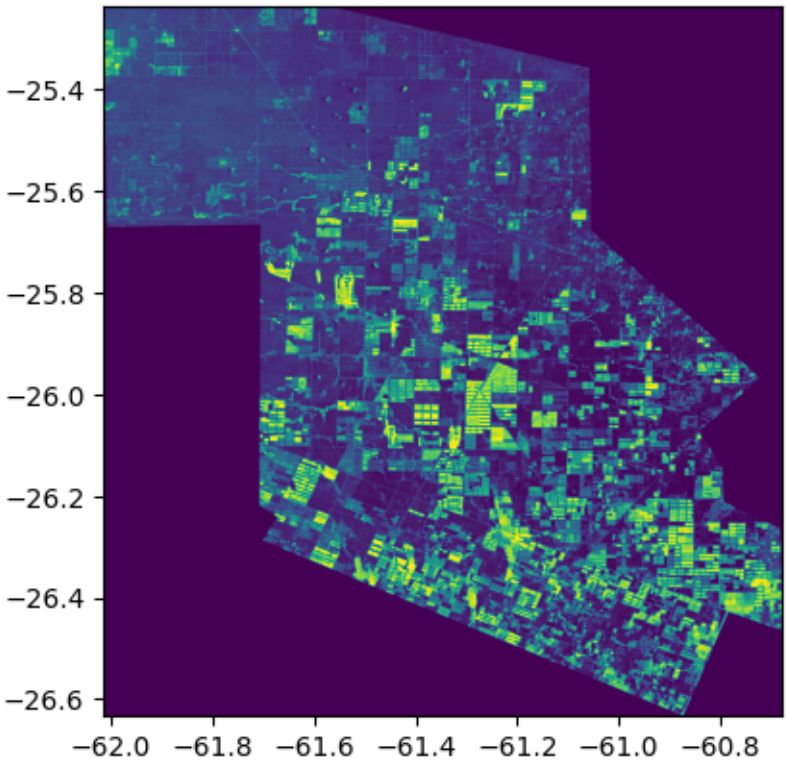
[13]:
```
gdal.Info('./data/s2_green.tif', format='json')
```

[13]:
```
{'description': './data/s2_green.tif',
 'driverShortName': 'GTiff',
 'driverLongName': 'GeoTIFF',
 'files': ['./data/s2_green.tif'],
 'size': [5000, 5000],
 'coordinateSystem': {'wkt': 'GEOGCRS["WGS 84",\n    ENSEMBLE["World Geodetic
System 1984 ensemble",\n        MEMBER["World Geodetic System 1984
(Transit)"],\n        MEMBER["World Geodetic System 1984 (G730)"],\n
MEMBER["World Geodetic System 1984 (G873)"],\n        MEMBER["World Geodetic
System 1984 (G1150)"],\n        MEMBER["World Geodetic System 1984 (G1674)"],\n
MEMBER["World Geodetic System 1984 (G1762)"],\n        MEMBER["World Geodetic
System 1984 (G2139)"],\n        ELLIPSOID["WGS 84",6378137,298.257223563,\n
LENGTHUNIT["metre",1]],\n        ENSEMBLEACCURACY[2.0]],\n
PRIMEM["Greenwich",0,\n        ANGLEUNIT["degree",0.0174532925199433]],\n
CS[ellipsoidal,2],\n        AXIS["geodetic latitude (Lat)",north,\n
ORDER[1],\n            ANGLEUNIT["degree",0.0174532925199433]],\n
AXIS["geodetic longitude (Lon)",east,\n            ORDER[2],\n
ANGLEUNIT["degree",0.0174532925199433]],\n    USAGE[\n        SCOPE["Horizontal
component of 3D system."],\n        AREA["World."],\n
BBOX[-90,-180,90,180]],\n    ID["EPSG",4326]]',
  'dataAxisToSRSAxisMapping': [2, 1]},
 'geoTransform': [-62.014387155,
  0.0002672308306,
  0.0,
  -25.234484815,
  0.0,
  -0.0002797533458],
 'metadata': {'': {'AREA_OR_POINT': 'Area'},
  'IMAGE_STRUCTURE': {'COMPRESSION': 'DEFLATE', 'INTERLEAVE': 'BAND'}},
 'cornerCoordinates': {'upperLeft': [-62.0143872, -25.2344848],
  'lowerLeft': [-62.0143872, -26.6332515],
  'lowerRight': [-60.678233, -26.6332515],
```

```
  'upperRight': [-60.678233, -25.2344848],
  'center': [-61.3463101, -25.9338682]},
 'wgs84Extent': {'type': 'Polygon',
  'coordinates': [[[-62.0143872, -25.2344848],
    [-62.0143872, -26.6332515],
    [-60.678233, -26.6332515],
    [-60.678233, -25.2344848],
    [-62.0143872, -25.2344848]]]},
 'bands': [{'band': 1,
   'block': [5000, 1],
   'type': 'Byte',
   'colorInterpretation': 'Gray',
   'metadata': {}}],
 'stac': {'proj:shape': [5000, 5000],
  'proj:wkt2': 'GEOGCRS["WGS 84",\n    ENSEMBLE["World Geodetic System 1984
ensemble",\n        MEMBER["World Geodetic System 1984 (Transit)"],\n
MEMBER["World Geodetic System 1984 (G730)"],\n        MEMBER["World Geodetic
System 1984 (G873)"],\n        MEMBER["World Geodetic System 1984 (G1150)"],\n
MEMBER["World Geodetic System 1984 (G1674)"],\n        MEMBER["World Geodetic
System 1984 (G1762)"],\n        MEMBER["World Geodetic System 1984 (G2139)"],\n
ELLIPSOID["WGS 84",6378137,298.257223563,\n            LENGTHUNIT["metre",1]],\n
ENSEMBLEACCURACY[2.0]],\n    PRIMEM["Greenwich",0,\n
ANGLEUNIT["degree",0.0174532925199433]],\n    CS[ellipsoidal,2],\n
AXIS["geodetic latitude (Lat)",north,\n            ORDER[1],\n
ANGLEUNIT["degree",0.0174532925199433]],\n        AXIS["geodetic longitude
(Lon)",east,\n            ORDER[2],\n
ANGLEUNIT["degree",0.0174532925199433]],\n    USAGE[\n        SCOPE["Horizontal
component of 3D system."],\n        AREA["World."],\n
BBOX[-90,-180,90,180]],\n    ID["EPSG",4326]]',
  'proj:epsg': 4326,
  'proj:projjson': {'$schema':
'https://proj.org/schemas/v0.5/projjson.schema.json',
   'type': 'GeographicCRS',
   'name': 'WGS 84',
   'datum_ensemble': {'name': 'World Geodetic System 1984 ensemble',
    'members': [{'name': 'World Geodetic System 1984 (Transit)',
      'id': {'authority': 'EPSG', 'code': 1166}},
     {'name': 'World Geodetic System 1984 (G730)',
      'id': {'authority': 'EPSG', 'code': 1152}},
     {'name': 'World Geodetic System 1984 (G873)',
      'id': {'authority': 'EPSG', 'code': 1153}},
     {'name': 'World Geodetic System 1984 (G1150)',
      'id': {'authority': 'EPSG', 'code': 1154}},
     {'name': 'World Geodetic System 1984 (G1674)',
      'id': {'authority': 'EPSG', 'code': 1155}},
     {'name': 'World Geodetic System 1984 (G1762)',
      'id': {'authority': 'EPSG', 'code': 1156}},
```

```
     {'name': 'World Geodetic System 1984 (G2139)',
      'id': {'authority': 'EPSG', 'code': 1309}}],
    'ellipsoid': {'name': 'WGS 84',
     'semi_major_axis': 6378137,
     'inverse_flattening': 298.257223563},
    'accuracy': '2.0',
    'id': {'authority': 'EPSG', 'code': 6326}},
   'coordinate_system': {'subtype': 'ellipsoidal',
    'axis': [{'name': 'Geodetic latitude',
      'abbreviation': 'Lat',
      'direction': 'north',
      'unit': 'degree'},
     {'name': 'Geodetic longitude',
      'abbreviation': 'Lon',
      'direction': 'east',
      'unit': 'degree'}]},
   'scope': 'Horizontal component of 3D system.',
   'area': 'World.',
   'bbox': {'south_latitude': -90,
    'west_longitude': -180,
    'north_latitude': 90,
    'east_longitude': 180},
   'id': {'authority': 'EPSG', 'code': 4326}},
  'proj:transform': [-62.014387155,
   0.0002672308306,
   0.0,
   -25.234484815,
   0.0,
   -0.0002797533458],
  'raster:bands': [{'data_type': 'uint8'}],
  'eo:bands': [{'name': 'b1', 'description': 'Gray'}]}}
```

[14]:
```
import matplotlib.pyplot as plt
from rasterio.plot import show
data = rasterio.open('./data/s2_green.tif')
show(data)
plt.show()
```



Con GDAL

GDAL no proporciona una función directa en Python para extraer y guardar una única banda utilizando gdal_translate. Sin embargo, aún puedes lograr esto con el uso de gdal_translate en línea de comandos desde Python utilizando el módulo subprocess.

[15]:
```
!apt-get install gdal-bin
```

```
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
The following additional packages will be installed:
  python3-gdal python3-numpy
Suggested packages:
  libgdal-grass python-numpy-doc python3-pytest
The following NEW packages will be installed:
  gdal-bin python3-gdal python3-numpy
0 upgraded, 3 newly installed, 0 to remove and 45 not upgraded.
Need to get 5,055 kB of archives.
After this operation, 25.1 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 python3-numpy
amd64 1:1.21.5-1ubuntu22.04.1 [3,467 kB]
Get:2 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy/main amd64
```

```
python3-gdal amd64 3.6.4+dfsg-1~jammy0 [1,027 kB]
Get:3 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy/main amd64
gdal-bin amd64 3.6.4+dfsg-1~jammy0 [561 kB]
Fetched 5,055 kB in 1s (5,009 kB/s)
Selecting previously unselected package python3-numpy.
(Reading database … 121752 files and directories currently installed.)
Preparing to unpack …/python3-numpy_1%3a1.21.5-1ubuntu22.04.1_amd64.deb …
Unpacking python3-numpy (1:1.21.5-1ubuntu22.04.1) …
Selecting previously unselected package python3-gdal.
Preparing to unpack …/python3-gdal_3.6.4+dfsg-1~jammy0_amd64.deb …
Unpacking python3-gdal (3.6.4+dfsg-1~jammy0) …
Selecting previously unselected package gdal-bin.
Preparing to unpack …/gdal-bin_3.6.4+dfsg-1~jammy0_amd64.deb …
Unpacking gdal-bin (3.6.4+dfsg-1~jammy0) …
Setting up python3-numpy (1:1.21.5-1ubuntu22.04.1) …
Setting up python3-gdal (3.6.4+dfsg-1~jammy0) …
Setting up gdal-bin (3.6.4+dfsg-1~jammy0) …
Processing triggers for man-db (2.10.2-1) …
```

[16]:
```python
import subprocess

def seleccionar_y_guardar_banda_gdal_translate(raster_path, banda_deseada,␣
 ↪salida_path):
    # Construir el comando gdal_translate
    comando = [
        'gdal_translate',
        '-b', str(banda_deseada),
        raster_path,
        salida_path
    ]

    # Ejecutar el comando
    subprocess.run(comando)

# Ejemplo de uso
seleccionar_y_guardar_banda_gdal_translate('./data/s2_rgb.tif', banda_deseada,␣
 ↪'./data/s2_green_gdal.tif')
```

[17]:
```python
gdal.Info('./data/s2_green_gdal.tif', format='json')
```

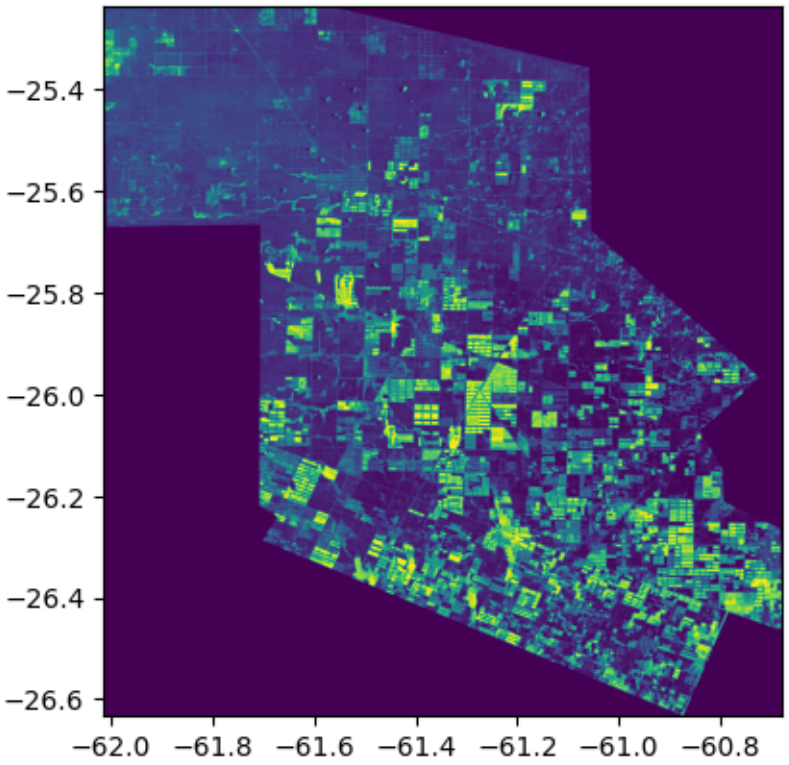
[17]:
```
{'description': './data/s2_green_gdal.tif',
 'driverShortName': 'GTiff',
 'driverLongName': 'GeoTIFF',
 'files': ['./data/s2_green_gdal.tif'],
 'size': [5000, 5000],
 'coordinateSystem': {'wkt': 'GEOGCRS["WGS 84",\n    ENSEMBLE["World Geodetic
System 1984 ensemble",\n        MEMBER["World Geodetic System 1984
```

```
(Transit)"],\n        MEMBER["World Geodetic System 1984 (G730)"],\n
MEMBER["World Geodetic System 1984 (G873)"],\n        MEMBER["World Geodetic
System 1984 (G1150)"],\n        MEMBER["World Geodetic System 1984 (G1674)"],\n
MEMBER["World Geodetic System 1984 (G1762)"],\n        MEMBER["World Geodetic
System 1984 (G2139)"],\n        ELLIPSOID["WGS 84",6378137,298.257223563,\n
LENGTHUNIT["metre",1]],\n        ENSEMBLEACCURACY[2.0]],\n
PRIMEM["Greenwich",0,\n        ANGLEUNIT["degree",0.0174532925199433]],\n
CS[ellipsoidal,2],\n        AXIS["geodetic latitude (Lat)",north,\n
ORDER[1],\n            ANGLEUNIT["degree",0.0174532925199433]],\n
AXIS["geodetic longitude (Lon)",east,\n            ORDER[2],\n
ANGLEUNIT["degree",0.0174532925199433]],\n    USAGE[\n        SCOPE["Horizontal
component of 3D system."],\n        AREA["World."],\n
BBOX[-90,-180,90,180]],\n    ID["EPSG",4326]]',
  'dataAxisToSRSAxisMapping': [2, 1]},
 'geoTransform': [-62.014387155,
  0.0002672308306,
  0.0,
  -25.234484815,
  0.0,
  -0.0002797533458],
 'metadata': {'': {'AREA_OR_POINT': 'Area'},
  'IMAGE_STRUCTURE': {'INTERLEAVE': 'BAND'}},
 'cornerCoordinates': {'upperLeft': [-62.0143872, -25.2344848],
  'lowerLeft': [-62.0143872, -26.6332515],
  'lowerRight': [-60.678233, -26.6332515],
  'upperRight': [-60.678233, -25.2344848],
  'center': [-61.3463101, -25.9338682]},
 'wgs84Extent': {'type': 'Polygon',
  'coordinates': [[[-62.0143872, -25.2344848],
    [-62.0143872, -26.6332515],
    [-60.678233, -26.6332515],
    [-60.678233, -25.2344848],
    [-62.0143872, -25.2344848]]]},
 'bands': [{'band': 1,
   'block': [5000, 1],
   'type': 'Byte',
   'colorInterpretation': 'Green',
   'metadata': {}}],
 'stac': {'proj:shape': [5000, 5000],
  'proj:wkt2': 'GEOGCRS["WGS 84",\n    ENSEMBLE["World Geodetic System 1984
ensemble",\n        MEMBER["World Geodetic System 1984 (Transit)"],\n
MEMBER["World Geodetic System 1984 (G730)"],\n        MEMBER["World Geodetic
System 1984 (G873)"],\n        MEMBER["World Geodetic System 1984 (G1150)"],\n
MEMBER["World Geodetic System 1984 (G1674)"],\n        MEMBER["World Geodetic
System 1984 (G1762)"],\n        MEMBER["World Geodetic System 1984 (G2139)"],\n
ELLIPSOID["WGS 84",6378137,298.257223563,\n            LENGTHUNIT["metre",1]],\n
ENSEMBLEACCURACY[2.0]],\n    PRIMEM["Greenwich",0,\n
```

```
ANGLEUNIT["degree",0.0174532925199433]],\n    CS[ellipsoidal,2],\n
AXIS["geodetic latitude (Lat)",north,\n            ORDER[1],\n
ANGLEUNIT["degree",0.0174532925199433]],\n        AXIS["geodetic longitude
(Lon)",east,\n            ORDER[2],\n
ANGLEUNIT["degree",0.0174532925199433]],\n    USAGE[\n        SCOPE["Horizontal
component of 3D system."],\n        AREA["World."],\n
BBOX[-90,-180,90,180]],\n    ID["EPSG",4326]]',
  'proj:epsg': 4326,
  'proj:projjson': {'$schema':
'https://proj.org/schemas/v0.5/projjson.schema.json',
   'type': 'GeographicCRS',
   'name': 'WGS 84',
   'datum_ensemble': {'name': 'World Geodetic System 1984 ensemble',
    'members': [{'name': 'World Geodetic System 1984 (Transit)',
      'id': {'authority': 'EPSG', 'code': 1166}},
     {'name': 'World Geodetic System 1984 (G730)',
      'id': {'authority': 'EPSG', 'code': 1152}},
     {'name': 'World Geodetic System 1984 (G873)',
      'id': {'authority': 'EPSG', 'code': 1153}},
     {'name': 'World Geodetic System 1984 (G1150)',
      'id': {'authority': 'EPSG', 'code': 1154}},
     {'name': 'World Geodetic System 1984 (G1674)',
      'id': {'authority': 'EPSG', 'code': 1155}},
     {'name': 'World Geodetic System 1984 (G1762)',
      'id': {'authority': 'EPSG', 'code': 1156}},
     {'name': 'World Geodetic System 1984 (G2139)',
      'id': {'authority': 'EPSG', 'code': 1309}}],
    'ellipsoid': {'name': 'WGS 84',
     'semi_major_axis': 6378137,
     'inverse_flattening': 298.257223563},
    'accuracy': '2.0',
    'id': {'authority': 'EPSG', 'code': 6326}},
   'coordinate_system': {'subtype': 'ellipsoidal',
    'axis': [{'name': 'Geodetic latitude',
      'abbreviation': 'Lat',
      'direction': 'north',
      'unit': 'degree'},
     {'name': 'Geodetic longitude',
      'abbreviation': 'Lon',
      'direction': 'east',
      'unit': 'degree'}]},
   'scope': 'Horizontal component of 3D system.',
   'area': 'World.',
   'bbox': {'south_latitude': -90,
    'west_longitude': -180,
    'north_latitude': 90,
    'east_longitude': 180},
```

```
 'id': {'authority': 'EPSG', 'code': 4326}},
'proj:transform': [-62.014387155,
 0.0002672308306,
 0.0,
 -25.234484815,
 0.0,
 -0.0002797533458],
'raster:bands': [{'data_type': 'uint8'}],
'eo:bands': [{'name': 'b1', 'description': 'Green'}]}}
```

[18]:
```
import matplotlib.pyplot as plt
from rasterio.plot import show
data = rasterio.open('./data/s2_green_gdal.tif')
show(data)
plt.show()
```



0.0.6 5. Recortar a partir de una Capa Vectorial:

El recorte basado en una capa vectorial es fundamental para enfocarse en áreas específicas de interés (AOI) en teledetección. A continuación, detallamos cómo recortar un raster utilizando tanto rasterio como GDAL, tomando como referencia una geometría definida en una capa vectorial (`aoi.geojson`).

Con rasterio

[19]:
```
import geopandas as gpd
from rasterio.mask import mask

# Abrir el raster original
with rasterio.open('./data/s2_rgb.tif') as src:
    # Abrir la capa vectorial AOI.geojson
    aoi = gpd.read_file('./data/aoi.geojson')

    # Recortar el raster basado en la geometría de la capa vectorial
    out_image, out_transform = mask(src, aoi.geometry, crop=True)

    # Actualizar el perfil del raster recortado
    out_meta = src.meta.copy()
    out_meta.update({
        'height': out_image.shape[1],
        'width': out_image.shape[2],
        'transform': out_transform
    })

    # Guardar el nuevo raster recortado
    with rasterio.open('./data/s2_rgb_clip.tif', 'w', **out_meta) as dst:
        dst.write(out_image)
```

[20]:
```
gdal.Info('./data/s2_rgb_clip.tif', format='json')
```

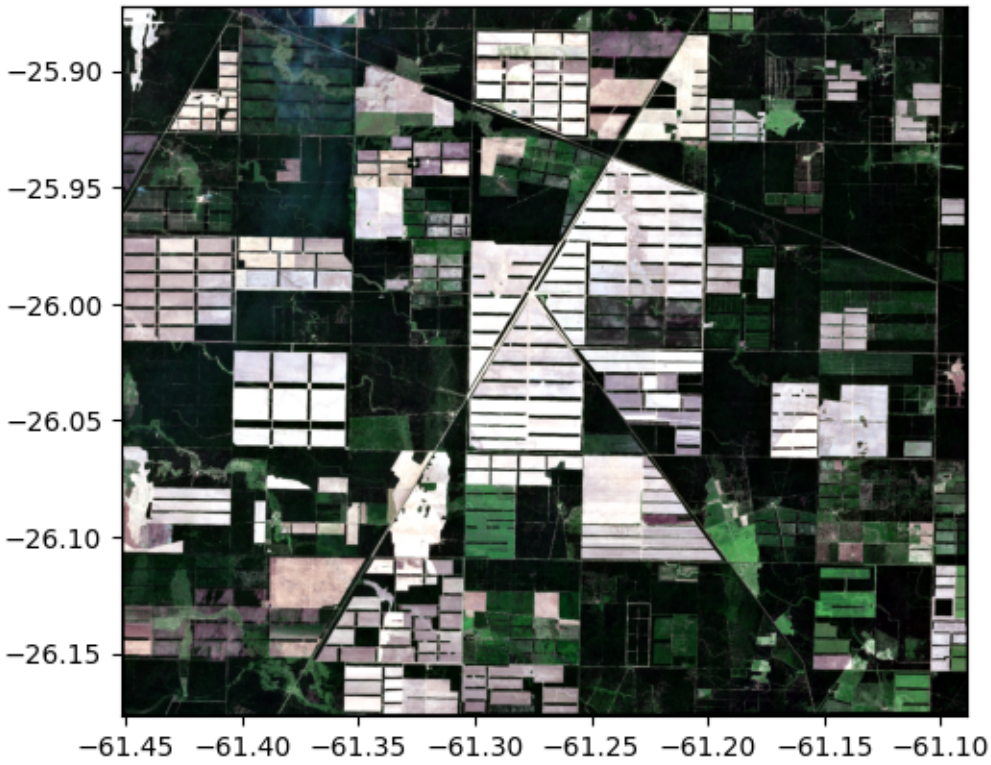
[20]:
```
{'description': './data/s2_rgb_clip.tif',
 'driverShortName': 'GTiff',
 'driverLongName': 'GeoTIFF',
 'files': ['./data/s2_rgb_clip.tif'],
 'size': [1358, 1090],
 'coordinateSystem': {'wkt': 'GEOGCRS["WGS 84",\n    ENSEMBLE["World Geodetic
System 1984 ensemble",\n        MEMBER["World Geodetic System 1984
(Transit)"],\n        MEMBER["World Geodetic System 1984 (G730)"],\n
MEMBER["World Geodetic System 1984 (G873)"],\n        MEMBER["World Geodetic
System 1984 (G1150)"],\n        MEMBER["World Geodetic System 1984 (G1674)"],\n
MEMBER["World Geodetic System 1984 (G1762)"],\n        MEMBER["World Geodetic
System 1984 (G2139)"],\n        ELLIPSOID["WGS 84",6378137,298.257223563,\n
LENGTHUNIT["metre",1]],\n        ENSEMBLEACCURACY[2.0]],\n
PRIMEM["Greenwich",0,\n        ANGLEUNIT["degree",0.0174532925199433]],\n
CS[ellipsoidal,2],\n        AXIS["geodetic latitude (Lat)",north,\n
ORDER[1],\n            ANGLEUNIT["degree",0.0174532925199433]],\n
AXIS["geodetic longitude (Lon)",east,\n            ORDER[2],\n
ANGLEUNIT["degree",0.0174532925199433]],\n    USAGE[\n        SCOPE["Horizontal
```

```
component of 3D system."],\n        AREA["World."],\n
BBOX[-90,-180,90,180]],\n    ID["EPSG",4326]]',
  'dataAxisToSRSAxisMapping': [2, 1]},
 'geoTransform': [-61.451866256587,
  0.0002672308306,
  0.0,
  -25.8720426900782,
  0.0,
  -0.0002797533458],
 'metadata': {'': {'AREA_OR_POINT': 'Area'},
  'IMAGE_STRUCTURE': {'INTERLEAVE': 'PIXEL'}},
 'cornerCoordinates': {'upperLeft': [-61.4518663, -25.8720427],
  'lowerLeft': [-61.4518663, -26.1769738],
  'lowerRight': [-61.0889668, -26.1769738],
  'upperRight': [-61.0889668, -25.8720427],
  'center': [-61.2704165, -26.0245083]},
 'wgs84Extent': {'type': 'Polygon',
  'coordinates': [[[-61.4518663, -25.8720427],
    [-61.4518663, -26.1769738],
    [-61.0889668, -26.1769738],
    [-61.0889668, -25.8720427],
    [-61.4518663, -25.8720427]]]},
 'bands': [{'band': 1,
   'block': [1358, 1],
   'type': 'Byte',
   'colorInterpretation': 'Red',
   'mask': {'flags': ['PER_DATASET', 'ALPHA'], 'overviews': []},
   'metadata': {}},
  {'band': 2,
   'block': [1358, 1],
   'type': 'Byte',
   'colorInterpretation': 'Green',
   'mask': {'flags': ['PER_DATASET', 'ALPHA'], 'overviews': []},
   'metadata': {}},
  {'band': 3,
   'block': [1358, 1],
   'type': 'Byte',
   'colorInterpretation': 'Blue',
   'mask': {'flags': ['PER_DATASET', 'ALPHA'], 'overviews': []},
   'metadata': {}},
  {'band': 4,
   'block': [1358, 1],
   'type': 'Byte',
   'colorInterpretation': 'Alpha',
   'metadata': {}}],
 'stac': {'proj:shape': [1358, 1090],
  'proj:wkt2': 'GEOGCRS["WGS 84",\n    ENSEMBLE["World Geodetic System 1984
```

```
ensemble",\n        MEMBER["World Geodetic System 1984 (Transit)"],\n
MEMBER["World Geodetic System 1984 (G730)"],\n        MEMBER["World Geodetic
System 1984 (G873)"],\n        MEMBER["World Geodetic System 1984 (G1150)"],\n
MEMBER["World Geodetic System 1984 (G1674)"],\n        MEMBER["World Geodetic
System 1984 (G1762)"],\n        MEMBER["World Geodetic System 1984 (G2139)"],\n
ELLIPSOID["WGS 84",6378137,298.257223563,\n            LENGTHUNIT["metre",1]],\n
ENSEMBLEACCURACY[2.0]],\n    PRIMEM["Greenwich",0,\n
ANGLEUNIT["degree",0.0174532925199433]],\n    CS[ellipsoidal,2],\n
AXIS["geodetic latitude (Lat)",north,\n            ORDER[1],\n
ANGLEUNIT["degree",0.0174532925199433]],\n        AXIS["geodetic longitude
(Lon)",east,\n            ORDER[2],\n
ANGLEUNIT["degree",0.0174532925199433]],\n    USAGE[\n        SCOPE["Horizontal
component of 3D system."],\n        AREA["World."],\n
BBOX[-90,-180,90,180]],\n    ID["EPSG",4326]]',
  'proj:epsg': 4326,
  'proj:projjson': {'$schema':
'https://proj.org/schemas/v0.5/projjson.schema.json',
   'type': 'GeographicCRS',
   'name': 'WGS 84',
   'datum_ensemble': {'name': 'World Geodetic System 1984 ensemble',
    'members': [{'name': 'World Geodetic System 1984 (Transit)',
      'id': {'authority': 'EPSG', 'code': 1166}},
     {'name': 'World Geodetic System 1984 (G730)',
      'id': {'authority': 'EPSG', 'code': 1152}},
     {'name': 'World Geodetic System 1984 (G873)',
      'id': {'authority': 'EPSG', 'code': 1153}},
     {'name': 'World Geodetic System 1984 (G1150)',
      'id': {'authority': 'EPSG', 'code': 1154}},
     {'name': 'World Geodetic System 1984 (G1674)',
      'id': {'authority': 'EPSG', 'code': 1155}},
     {'name': 'World Geodetic System 1984 (G1762)',
      'id': {'authority': 'EPSG', 'code': 1156}},
     {'name': 'World Geodetic System 1984 (G2139)',
      'id': {'authority': 'EPSG', 'code': 1309}}],
    'ellipsoid': {'name': 'WGS 84',
     'semi_major_axis': 6378137,
     'inverse_flattening': 298.257223563},
    'accuracy': '2.0',
    'id': {'authority': 'EPSG', 'code': 6326}},
   'coordinate_system': {'subtype': 'ellipsoidal',
    'axis': [{'name': 'Geodetic latitude',
      'abbreviation': 'Lat',
      'direction': 'north',
      'unit': 'degree'},
     {'name': 'Geodetic longitude',
      'abbreviation': 'Lon',
      'direction': 'east',
```

```
     'unit': 'degree'}]},
   'scope': 'Horizontal component of 3D system.',
   'area': 'World.',
   'bbox': {'south_latitude': -90,
    'west_longitude': -180,
    'north_latitude': 90,
    'east_longitude': 180},
   'id': {'authority': 'EPSG', 'code': 4326}},
  'proj:transform': [-61.451866256587,
   0.0002672308306,
   0.0,
   -25.8720426900782,
   0.0,
   -0.0002797533458],
  'raster:bands': [{'data_type': 'uint8'},
   {'data_type': 'uint8'},
   {'data_type': 'uint8'},
   {'data_type': 'uint8'}],
  'eo:bands': [{'name': 'b1', 'description': 'Red'},
   {'name': 'b2', 'description': 'Green'},
   {'name': 'b3', 'description': 'Blue'},
   {'name': 'b4', 'description': 'Alpha'}]}}
```

[21]:
```
import matplotlib.pyplot as plt
from rasterio.plot import show
data = rasterio.open('./data/s2_rgb_clip.tif')
show(data)
plt.show()
```



Con GDAL

[22]:
```
from osgeo import ogr, gdal

# Abrir el raster original
src_ds = gdal.Open('./data/s2_rgb.tif')

# Crear una máscara para el recorte basado en la geometría de la capa vectorial
options = gdal.WarpOptions(cutlineDSName='./data/aoi.geojson',␣
 ↪cropToCutline=True)
dst_ds = gdal.Warp('./data/s2_rgb_clip_gdal.tif', src_ds, options=options)

# Cerrar los datasets
src_ds = None
dst_ds = None
```

[23]:
```
gdal.Info('./data/s2_rgb_clip_gdal.tif', format='json')
```

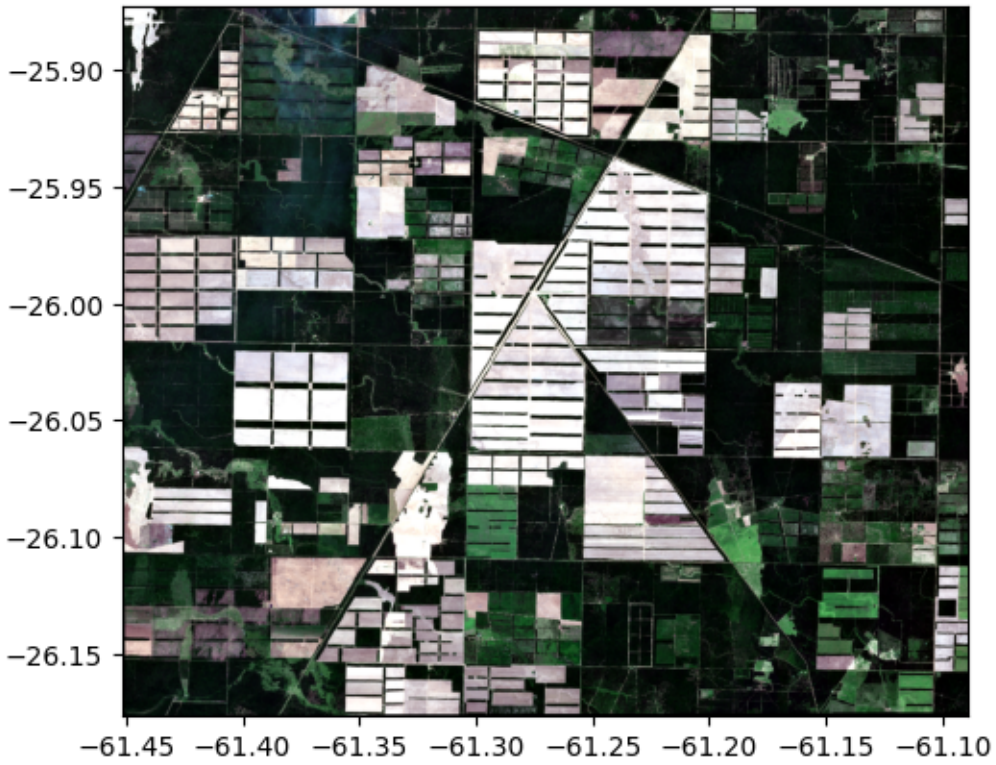
[23]:
```
{'description': './data/s2_rgb_clip_gdal.tif',
 'driverShortName': 'GTiff',
 'driverLongName': 'GeoTIFF',
 'files': ['./data/s2_rgb_clip_gdal.tif'],
```

```
 'size': [1325, 1113],
 'coordinateSystem': {'wkt': 'GEOGCRS["WGS 84",\n    ENSEMBLE["World Geodetic
System 1984 ensemble",\n        MEMBER["World Geodetic System 1984
(Transit)"],\n        MEMBER["World Geodetic System 1984 (G730)"],\n
MEMBER["World Geodetic System 1984 (G873)"],\n        MEMBER["World Geodetic
System 1984 (G1150)"],\n        MEMBER["World Geodetic System 1984 (G1674)"],\n
MEMBER["World Geodetic System 1984 (G1762)"],\n        MEMBER["World Geodetic
System 1984 (G2139)"],\n        ELLIPSOID["WGS 84",6378137,298.257223563,\n
LENGTHUNIT["metre",1]],\n        ENSEMBLEACCURACY[2.0]],\n
PRIMEM["Greenwich",0,\n        ANGLEUNIT["degree",0.0174532925199433]],\n
CS[ellipsoidal,2],\n        AXIS["geodetic latitude (Lat)",north,\n
ORDER[1],\n            ANGLEUNIT["degree",0.0174532925199433]],\n
AXIS["geodetic longitude (Lon)",east,\n            ORDER[2],\n
ANGLEUNIT["degree",0.0174532925199433]],\n    USAGE[\n        SCOPE["Horizontal
component of 3D system."],\n        AREA["World."],\n
BBOX[-90,-180,90,180]],\n    ID["EPSG",4326]]',
  'dataAxisToSRSAxisMapping': [2, 1]},
 'geoTransform': [-61.4515990257564,
  0.0002734830236178,
  0.0,
  -25.872322443424,
  0.0,
  -0.0002734695779249],
 'metadata': {'': {'AREA_OR_POINT': 'Area'},
  'IMAGE_STRUCTURE': {'INTERLEAVE': 'PIXEL'}},
 'cornerCoordinates': {'upperLeft': [-61.451599, -25.8723224],
  'lowerLeft': [-61.451599, -26.1766941],
  'lowerRight': [-61.089234, -26.1766941],
  'upperRight': [-61.089234, -25.8723224],
  'center': [-61.2704165, -26.0245083]},
 'wgs84Extent': {'type': 'Polygon',
  'coordinates': [[[-61.451599, -25.8723224],
    [-61.451599, -26.1766941],
    [-61.089234, -26.1766941],
    [-61.089234, -25.8723224],
    [-61.451599, -25.8723224]]]},
 'bands': [{'band': 1,
   'block': [1325, 1],
   'type': 'Byte',
   'colorInterpretation': 'Red',
   'mask': {'flags': ['PER_DATASET', 'ALPHA'], 'overviews': []},
   'metadata': {}},
  {'band': 2,
   'block': [1325, 1],
   'type': 'Byte',
   'colorInterpretation': 'Green',
   'mask': {'flags': ['PER_DATASET', 'ALPHA'], 'overviews': []},
```

```
   'metadata': {}},
  {'band': 3,
   'block': [1325, 1],
   'type': 'Byte',
   'colorInterpretation': 'Blue',
   'mask': {'flags': ['PER_DATASET', 'ALPHA'], 'overviews': []},
   'metadata': {}},
  {'band': 4,
   'block': [1325, 1],
   'type': 'Byte',
   'colorInterpretation': 'Alpha',
   'metadata': {}}],
 'stac': {'proj:shape': [1325, 1113],
  'proj:wkt2': 'GEOGCRS["WGS 84",\n    ENSEMBLE["World Geodetic System 1984
ensemble",\n        MEMBER["World Geodetic System 1984 (Transit)"],\n
MEMBER["World Geodetic System 1984 (G730)"],\n        MEMBER["World Geodetic
System 1984 (G873)"],\n        MEMBER["World Geodetic System 1984 (G1150)"],\n
MEMBER["World Geodetic System 1984 (G1674)"],\n        MEMBER["World Geodetic
System 1984 (G1762)"],\n        MEMBER["World Geodetic System 1984 (G2139)"],\n
ELLIPSOID["WGS 84",6378137,298.257223563,\n            LENGTHUNIT["metre",1]],\n
ENSEMBLEACCURACY[2.0]],\n    PRIMEM["Greenwich",0,\n
ANGLEUNIT["degree",0.0174532925199433]],\n    CS[ellipsoidal,2],\n
AXIS["geodetic latitude (Lat)",north,\n            ORDER[1],\n
ANGLEUNIT["degree",0.0174532925199433]],\n        AXIS["geodetic longitude
(Lon)",east,\n            ORDER[2],\n
ANGLEUNIT["degree",0.0174532925199433]],\n    USAGE[\n        SCOPE["Horizontal
component of 3D system."],\n        AREA["World."],\n
BBOX[-90,-180,90,180]],\n    ID["EPSG",4326]]',
  'proj:epsg': 4326,
  'proj:projjson': {'$schema':
'https://proj.org/schemas/v0.5/projjson.schema.json',
   'type': 'GeographicCRS',
   'name': 'WGS 84',
   'datum_ensemble': {'name': 'World Geodetic System 1984 ensemble',
    'members': [{'name': 'World Geodetic System 1984 (Transit)',
      'id': {'authority': 'EPSG', 'code': 1166}},
     {'name': 'World Geodetic System 1984 (G730)',
      'id': {'authority': 'EPSG', 'code': 1152}},
     {'name': 'World Geodetic System 1984 (G873)',
      'id': {'authority': 'EPSG', 'code': 1153}},
     {'name': 'World Geodetic System 1984 (G1150)',
      'id': {'authority': 'EPSG', 'code': 1154}},
     {'name': 'World Geodetic System 1984 (G1674)',
      'id': {'authority': 'EPSG', 'code': 1155}},
     {'name': 'World Geodetic System 1984 (G1762)',
      'id': {'authority': 'EPSG', 'code': 1156}},
     {'name': 'World Geodetic System 1984 (G2139)',
```

```
    'id': {'authority': 'EPSG', 'code': 1309}}],
   'ellipsoid': {'name': 'WGS 84',
    'semi_major_axis': 6378137,
    'inverse_flattening': 298.257223563},
   'accuracy': '2.0',
   'id': {'authority': 'EPSG', 'code': 6326}},
  'coordinate_system': {'subtype': 'ellipsoidal',
   'axis': [{'name': 'Geodetic latitude',
     'abbreviation': 'Lat',
     'direction': 'north',
     'unit': 'degree'},
    {'name': 'Geodetic longitude',
     'abbreviation': 'Lon',
     'direction': 'east',
     'unit': 'degree'}]},
  'scope': 'Horizontal component of 3D system.',
  'area': 'World.',
  'bbox': {'south_latitude': -90,
   'west_longitude': -180,
   'north_latitude': 90,
   'east_longitude': 180},
  'id': {'authority': 'EPSG', 'code': 4326}},
 'proj:transform': [-61.4515990257564,
  0.0002734830236178,
  0.0,
  -25.872322443424,
  0.0,
  -0.0002734695779249],
 'raster:bands': [{'data_type': 'uint8'},
  {'data_type': 'uint8'},
  {'data_type': 'uint8'},
  {'data_type': 'uint8'}],
 'eo:bands': [{'name': 'b1', 'description': 'Red'},
  {'name': 'b2', 'description': 'Green'},
  {'name': 'b3', 'description': 'Blue'},
  {'name': 'b4', 'description': 'Alpha'}]}}
```


[24]:
```
import matplotlib.pyplot as plt
from rasterio.plot import show
data = rasterio.open('./data/s2_rgb_clip_gdal.tif')
show(data)
plt.show()
```



0.0.7 6. Recorrido por Ventanas en rasterio:

El recorrido por ventanas es una técnica útil al trabajar con grandes conjuntos de datos raster, permitiendo procesar el raster en bloques más pequeños. En este ejemplo, presentamos cómo realizar un recorrido por ventanas utilizando rasterio para procesar datos raster de manera eficiente.

Veamos el caso donde queremos recorrer de a 256 pixeles (ventana) y guardar la primera banda de la imagen resultante (chip) como un nuevo archivo.

El nombre del archivo resultante deberá ser `chip_{posicion_pixelInicial}_{posicion_pixelFinal}.tif`, dentro del directorio `chips`.

[25]:
```
!mkdir -p chips
```

[30]:
```
from rasterio.windows import Window
ventana_size = 256
salida_path_template = './chips/chip_{i}_{j}.tif'
with rasterio.open('./data/s2_rgb.tif') as src:
        width = src.width
        height = src.height
        for i in range(0, width, ventana_size):
                    for j in range(0, height, ventana_size):
```

```
                        window = Window(i, j, min(ventana_size, width - i),␣
↪min(ventana_size, height - j))

                        # Leer solo la primera banda de la ventana
                        chip = src.read(window=window, indexes=1)
                        chip = chip.reshape(1,window.height,window.width)

                        # Configurar el perfil para la nueva imagen
                        profile = src.profile
                        profile.update(width=window.width,
                                       height=window.height,
                                       count=1,
                                       transform = rasterio.windows.
↪transform(window, src.transform))
                        # Construir el nombre de salida con coordenadas de la␣
↪ventana
                        salida_path = salida_path_template.format(i=i, j=j)

                        # Guardar el chip como un nuevo archivo raster
                        with rasterio.open(salida_path, 'w', **profile) as dst:
                            dst.write(chip)
```

Pasemos a una función para ordenar mejor el código.

[27]:
```
def recortar_y_guardar_chips(raster_path, ventana_size, salida_path_template):
    """
    Recorta el raster en chips de tamaño ventana_size y guarda cada chip como␣
↪un archivo raster individual.

    Parameters:
    - raster_path (str): Ruta al archivo raster original.
    - ventana_size (int): Tamaño de la ventana para recortar los chips.
    - salida_path_template (str): Plantilla para las rutas de salida de los␣
↪archivos raster.

    Returns:
    - None
    """
    with rasterio.open(raster_path) as src:
        width = src.width
        height = src.height

        for i in range(0, width, ventana_size):
            for j in range(0, height, ventana_size):
                # Crear una ventana para el chip
                window = Window(i, j, min(ventana_size, width - i),␣
↪min(ventana_size, height - j))
```

```
                # Leer solo la primera banda de la ventana
                chip = src.read(window=window, indexes=1)
                chip = chip.reshape(1, window.height, window.width)

                # Configurar el perfil para la nueva imagen
                profile = src.profile
                profile.update(width=window.width,
                               height=window.height,
                               count=1,
                               transform = rasterio.windows.transform(window,␣
↪src.transform))

                # Construir el nombre de salida con coordenadas de la ventana
                salida_path = salida_path_template.format(i=i, j=j)

                # Guardar el chip como un nuevo archivo raster
                with rasterio.open(salida_path, 'w', **profile) as dst:
                    dst.write(chip)

# Ejemplo de uso
recortar_y_guardar_chips('./data/s2_rgb.tif', 256, './chips/chip_{i}_{j}.tif')
```

Explicación paso a paso:

1. Importación de bibliotecas: Importamos las bibliotecas necesarias, en este caso, rasterio y la clase Window de rasterio.windows.
2. Definición de la función recortar_y_guardar_chips: Creamos una función que toma tres parámetros (raster_path, ventana_size, y salida_path_template) para recortar un raster en chips y guardar cada chip como un archivo raster individual.
3. Bloque con rasterio.open: Abrimos el conjunto de datos raster original como un bloque de contexto (with), lo que garantiza que el conjunto de datos se cierre correctamente después de su uso.
4. Iteración sobre ventanas: Utilizamos dos bucles for para iterar sobre las ventanas de tamaño ventana_size tanto en la dirección x como en la dirección y del raster original.
5. Creación de ventana con Window: Creamos una ventana (Window) para el chip en cada iteración. Ajustamos las dimensiones de la ventana si estamos cerca de los bordes del raster original.
6. Lectura de la primera banda y reshape: Leemos solo la primera banda de la ventana y luego ajustamos la forma (reshape) del chip para que tenga la forma (1, height, width).
7. Configuración del perfil del nuevo raster: Actualizamos el perfil del conjunto de datos original para que coincida con las dimensiones del chip.
8. Construcción de la ruta de salida: Construimos la ruta de salida utilizando la plantilla proporcionada y las coordenadas de la ventana.

9. Escritura del chip en un nuevo raster: Abrimos un nuevo conjunto de datos raster de salida (dst) y escribimos el chip en este conjunto de datos.

10. Ejemplo de uso: Llamamos a la función recortar_y_guardar_chips con una muestra de ruta al conjunto de datos raster original, un tamaño de ventana de 256 píxeles y una plantilla para las rutas de salida.

[28]:
```
import glob
glob.glob('./chips/*')
```

[28]:
```
['./chips/chip_1280_0.tif',
 './chips/chip_3584_3584.tif',
 './chips/chip_2816_512.tif',
 './chips/chip_4352_1024.tif',
 './chips/chip_4608_256.tif',
 './chips/chip_2816_3072.tif',
 './chips/chip_4608_1024.tif',
 './chips/chip_1536_4096.tif',
 './chips/chip_3328_2048.tif',
 './chips/chip_3584_3840.tif',
 './chips/chip_0_4352.tif',
 './chips/chip_1280_4096.tif',
 './chips/chip_3072_1792.tif',
 './chips/chip_512_1792.tif',
 './chips/chip_2816_1280.tif',
 './chips/chip_4352_4864.tif',
 './chips/chip_2048_1280.tif',
 './chips/chip_512_1024.tif',
 './chips/chip_3328_3072.tif',
 './chips/chip_3328_4864.tif',
 './chips/chip_4352_2304.tif',
 './chips/chip_3072_3072.tif',
 './chips/chip_4096_4864.tif',
 './chips/chip_2304_1536.tif',
 './chips/chip_4864_4864.tif',
 './chips/chip_256_1792.tif',
 './chips/chip_2816_2560.tif',
 './chips/chip_1792_3328.tif',
 './chips/chip_1536_2816.tif',
 './chips/chip_2816_2048.tif',
 './chips/chip_2560_1024.tif',
 './chips/chip_1536_1024.tif',
 './chips/chip_0_1280.tif',
 './chips/chip_3840_2304.tif',
 './chips/chip_1792_3072.tif',
 './chips/chip_512_4352.tif',
 './chips/chip_1280_1280.tif',
 './chips/chip_4864_1536.tif',
```

```
'./chips/chip_3584_2304.tif',
'./chips/chip_1024_4608.tif',
'./chips/chip_3584_4864.tif',
'./chips/chip_512_1536.tif',
'./chips/chip_1792_3840.tif',
'./chips/chip_768_2560.tif',
'./chips/chip_512_4608.tif',
'./chips/chip_512_768.tif',
'./chips/chip_768_256.tif',
'./chips/chip_1024_1536.tif',
'./chips/chip_768_2816.tif',
'./chips/chip_3840_3072.tif',
'./chips/chip_4608_768.tif',
'./chips/chip_3840_1280.tif',
'./chips/chip_4864_2560.tif',
'./chips/chip_1024_3584.tif',
'./chips/chip_3584_4352.tif',
'./chips/chip_256_1280.tif',
'./chips/chip_2304_512.tif',
'./chips/chip_3328_768.tif',
'./chips/chip_2048_3072.tif',
'./chips/chip_4608_3328.tif',
'./chips/chip_4864_4352.tif',
'./chips/chip_2560_0.tif',
'./chips/chip_3584_0.tif',
'./chips/chip_1536_1280.tif',
'./chips/chip_3584_1792.tif',
'./chips/chip_512_3328.tif',
'./chips/chip_3840_1024.tif',
'./chips/chip_256_768.tif',
'./chips/chip_3328_4352.tif',
'./chips/chip_3328_0.tif',
'./chips/chip_4096_1024.tif',
'./chips/chip_0_4608.tif',
'./chips/chip_3584_2560.tif',
'./chips/chip_2560_2816.tif',
'./chips/chip_2304_1792.tif',
'./chips/chip_0_1024.tif',
'./chips/chip_1536_3072.tif',
'./chips/chip_4608_2560.tif',
'./chips/chip_4608_3840.tif',
'./chips/chip_3072_1280.tif',
'./chips/chip_768_1792.tif',
'./chips/chip_768_1280.tif',
'./chips/chip_0_2048.tif',
'./chips/chip_512_2304.tif',
'./chips/chip_2560_3072.tif',
```

```
'./chips/chip_3328_3840.tif',
'./chips/chip_1024_1280.tif',
'./chips/chip_2048_768.tif',
'./chips/chip_0_4864.tif',
'./chips/chip_0_3584.tif',
'./chips/chip_1792_256.tif',
'./chips/chip_768_4352.tif',
'./chips/chip_2048_2816.tif',
'./chips/chip_2304_2816.tif',
'./chips/chip_2816_4352.tif',
'./chips/chip_1280_4352.tif',
'./chips/chip_2560_1280.tif',
'./chips/chip_256_3328.tif',
'./chips/chip_1280_2560.tif',
'./chips/chip_2048_1024.tif',
'./chips/chip_4608_1536.tif',
'./chips/chip_4352_2048.tif',
'./chips/chip_4096_3328.tif',
'./chips/chip_2816_3840.tif',
'./chips/chip_1792_1280.tif',
'./chips/chip_4608_4096.tif',
'./chips/chip_4864_3072.tif',
'./chips/chip_3328_4096.tif',
'./chips/chip_2816_768.tif',
'./chips/chip_3328_1792.tif',
'./chips/chip_2304_4608.tif',
'./chips/chip_3584_256.tif',
'./chips/chip_2048_4864.tif',
'./chips/chip_3584_4608.tif',
'./chips/chip_3584_1024.tif',
'./chips/chip_768_1536.tif',
'./chips/chip_1280_1792.tif',
'./chips/chip_4608_4864.tif',
'./chips/chip_4352_4352.tif',
'./chips/chip_4096_512.tif',
'./chips/chip_2304_1280.tif',
'./chips/chip_1792_4608.tif',
'./chips/chip_1792_4352.tif',
'./chips/chip_4096_3072.tif',
'./chips/chip_1280_2816.tif',
'./chips/chip_0_2816.tif',
'./chips/chip_2304_3840.tif',
'./chips/chip_4352_2560.tif',
'./chips/chip_3072_4864.tif',
'./chips/chip_3328_3328.tif',
'./chips/chip_2816_256.tif',
'./chips/chip_512_3584.tif',
```

```
'./chips/chip_0_3328.tif',
'./chips/chip_2048_4608.tif',
'./chips/chip_0_768.tif',
'./chips/chip_256_3072.tif',
'./chips/chip_3328_1536.tif',
'./chips/chip_1024_4096.tif',
'./chips/chip_4608_512.tif',
'./chips/chip_2560_3328.tif',
'./chips/chip_4608_2816.tif',
'./chips/chip_0_2560.tif',
'./chips/chip_3584_1280.tif',
'./chips/chip_2816_1536.tif',
'./chips/chip_3840_3328.tif',
'./chips/chip_4352_1792.tif',
'./chips/chip_3072_2816.tif',
'./chips/chip_3840_2048.tif',
'./chips/chip_0_4096.tif',
'./chips/chip_4608_3072.tif',
'./chips/chip_1536_3584.tif',
'./chips/chip_4096_2560.tif',
'./chips/chip_3584_3072.tif',
'./chips/chip_2304_3584.tif',
'./chips/chip_256_2816.tif',
'./chips/chip_3840_3584.tif',
'./chips/chip_3584_512.tif',
'./chips/chip_1792_3584.tif',
'./chips/chip_768_4096.tif',
'./chips/chip_4352_3840.tif',
'./chips/chip_4864_256.tif',
'./chips/chip_512_1280.tif',
'./chips/chip_2560_2304.tif',
'./chips/chip_3328_3584.tif',
'./chips/chip_1024_4352.tif',
'./chips/chip_4864_4096.tif',
'./chips/chip_1280_3072.tif',
'./chips/chip_3072_0.tif',
'./chips/chip_768_3584.tif',
'./chips/chip_1536_2304.tif',
'./chips/chip_256_4608.tif',
'./chips/chip_2560_1536.tif',
'./chips/chip_4608_0.tif',
'./chips/chip_1536_2048.tif',
'./chips/chip_3840_1792.tif',
'./chips/chip_4096_768.tif',
'./chips/chip_0_1792.tif',
'./chips/chip_4608_1280.tif',
'./chips/chip_256_3584.tif',
```

```
'./chips/chip_4864_1792.tif',
'./chips/chip_4352_512.tif',
'./chips/chip_3840_768.tif',
'./chips/chip_2048_1792.tif',
'./chips/chip_2560_768.tif',
'./chips/chip_3072_1536.tif',
'./chips/chip_3072_2304.tif',
'./chips/chip_3072_256.tif',
'./chips/chip_2560_4096.tif',
'./chips/chip_2816_2816.tif',
'./chips/chip_0_2304.tif',
'./chips/chip_1536_1536.tif',
'./chips/chip_1536_1792.tif',
'./chips/chip_2048_0.tif',
'./chips/chip_1792_768.tif',
'./chips/chip_3584_768.tif',
'./chips/chip_4352_4096.tif',
'./chips/chip_1536_2560.tif',
'./chips/chip_4352_0.tif',
'./chips/chip_3072_1024.tif',
'./chips/chip_4096_3584.tif',
'./chips/chip_2304_3328.tif',
'./chips/chip_2304_3072.tif',
'./chips/chip_1024_3840.tif',
'./chips/chip_256_4096.tif',
'./chips/chip_3072_512.tif',
'./chips/chip_4864_2048.tif',
'./chips/chip_256_1024.tif',
'./chips/chip_2048_2048.tif',
'./chips/chip_2816_4608.tif',
'./chips/chip_512_2560.tif',
'./chips/chip_3840_4864.tif',
'./chips/chip_2304_256.tif',
'./chips/chip_4608_4352.tif',
'./chips/chip_1280_768.tif',
'./chips/chip_2560_4864.tif',
'./chips/chip_256_4352.tif',
'./chips/chip_4096_2048.tif',
'./chips/chip_1536_3328.tif',
'./chips/chip_1792_4096.tif',
'./chips/chip_1024_512.tif',
'./chips/chip_0_256.tif',
'./chips/chip_2304_1024.tif',
'./chips/chip_2048_512.tif',
'./chips/chip_2816_4096.tif',
'./chips/chip_768_2304.tif',
'./chips/chip_0_3840.tif',
```

```
'./chips/chip_2816_4864.tif',
'./chips/chip_4096_2816.tif',
'./chips/chip_1536_0.tif',
'./chips/chip_1024_256.tif',
'./chips/chip_2560_3584.tif',
'./chips/chip_2816_1792.tif',
'./chips/chip_4352_3072.tif',
'./chips/chip_1280_1024.tif',
'./chips/chip_4864_0.tif',
'./chips/chip_1792_2304.tif',
'./chips/chip_2560_2560.tif',
'./chips/chip_1280_512.tif',
'./chips/chip_2816_0.tif',
'./chips/chip_256_512.tif',
'./chips/chip_2048_1536.tif',
'./chips/chip_2560_512.tif',
'./chips/chip_1024_2560.tif',
'./chips/chip_4352_3584.tif',
'./chips/chip_2816_3328.tif',
'./chips/chip_768_3328.tif',
'./chips/chip_3840_3840.tif',
'./chips/chip_0_3072.tif',
'./chips/chip_1280_256.tif',
'./chips/chip_2304_2560.tif',
'./chips/chip_256_2048.tif',
'./chips/chip_3328_4608.tif',
'./chips/chip_4864_1024.tif',
'./chips/chip_0_512.tif',
'./chips/chip_2304_0.tif',
'./chips/chip_3072_2048.tif',
'./chips/chip_1024_2816.tif',
'./chips/chip_256_4864.tif',
'./chips/chip_3072_768.tif',
'./chips/chip_4096_1792.tif',
'./chips/chip_3072_4096.tif',
'./chips/chip_3328_512.tif',
'./chips/chip_1792_1024.tif',
'./chips/chip_1280_3328.tif',
'./chips/chip_2304_2048.tif',
'./chips/chip_1792_2560.tif',
'./chips/chip_768_3072.tif',
'./chips/chip_2304_4864.tif',
'./chips/chip_2560_3840.tif',
'./chips/chip_3072_3840.tif',
'./chips/chip_3072_4608.tif',
'./chips/chip_2560_256.tif',
'./chips/chip_3328_1024.tif',
```

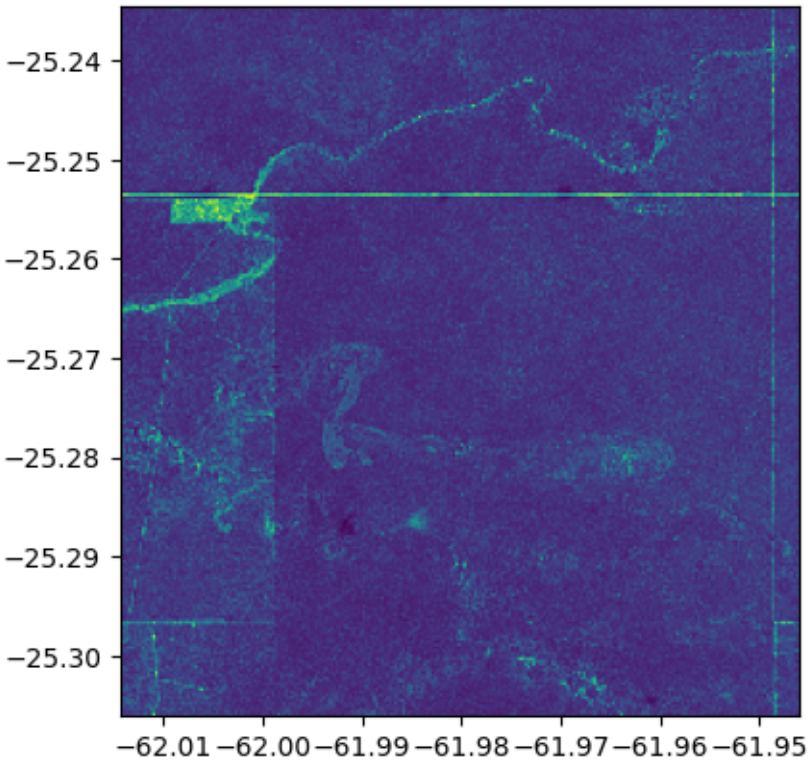
```
'./chips/chip_4096_0.tif',
'./chips/chip_4352_256.tif',
'./chips/chip_3840_512.tif',
'./chips/chip_2048_256.tif',
'./chips/chip_4608_2048.tif',
'./chips/chip_1024_3328.tif',
'./chips/chip_2048_4352.tif',
'./chips/chip_3840_256.tif',
'./chips/chip_2560_2048.tif',
'./chips/chip_4352_2816.tif',
'./chips/chip_3072_2560.tif',
'./chips/chip_768_4608.tif',
'./chips/chip_512_2048.tif',
'./chips/chip_2816_1024.tif',
'./chips/chip_1792_2048.tif',
'./chips/chip_4864_768.tif',
'./chips/chip_2560_4608.tif',
'./chips/chip_1024_1024.tif',
'./chips/chip_1024_768.tif',
'./chips/chip_768_512.tif',
'./chips/chip_4864_1280.tif',
'./chips/chip_4608_2304.tif',
'./chips/chip_768_768.tif',
'./chips/chip_4608_3584.tif',
'./chips/chip_4864_3584.tif',
'./chips/chip_768_1024.tif',
'./chips/chip_3072_3328.tif',
'./chips/chip_2048_4096.tif',
'./chips/chip_1536_3840.tif',
'./chips/chip_768_0.tif',
'./chips/chip_512_2816.tif',
'./chips/chip_256_2560.tif',
'./chips/chip_2048_2304.tif',
'./chips/chip_512_0.tif',
'./chips/chip_2304_4352.tif',
'./chips/chip_4096_4608.tif',
'./chips/chip_1280_2048.tif',
'./chips/chip_1536_4608.tif',
'./chips/chip_3840_2816.tif',
'./chips/chip_2816_2304.tif',
'./chips/chip_1024_1792.tif',
'./chips/chip_1280_3584.tif',
'./chips/chip_4352_1536.tif',
'./chips/chip_1792_1792.tif',
'./chips/chip_2560_1792.tif',
'./chips/chip_1280_1536.tif',
'./chips/chip_2048_2560.tif',
```

```
'./chips/chip_1024_2048.tif',
'./chips/chip_4352_768.tif',
'./chips/chip_1792_512.tif',
'./chips/chip_3840_4608.tif',
'./chips/chip_512_512.tif',
'./chips/chip_768_3840.tif',
'./chips/chip_0_1536.tif',
'./chips/chip_1536_512.tif',
'./chips/chip_256_0.tif',
'./chips/chip_1792_0.tif',
'./chips/chip_1024_4864.tif',
'./chips/chip_2304_768.tif',
'./chips/chip_1792_1536.tif',
'./chips/chip_1792_4864.tif',
'./chips/chip_512_4864.tif',
'./chips/chip_3328_2816.tif',
'./chips/chip_3584_2048.tif',
'./chips/chip_3584_3328.tif',
'./chips/chip_4864_2304.tif',
'./chips/chip_3584_4096.tif',
'./chips/chip_1536_256.tif',
'./chips/chip_4096_1536.tif',
'./chips/chip_3840_2560.tif',
'./chips/chip_1024_3072.tif',
'./chips/chip_768_2048.tif',
'./chips/chip_1536_768.tif',
'./chips/chip_4608_4608.tif',
'./chips/chip_1536_4352.tif',
'./chips/chip_1280_2304.tif',
'./chips/chip_256_1536.tif',
'./chips/chip_3328_1280.tif',
'./chips/chip_2048_3584.tif',
'./chips/chip_2304_4096.tif',
'./chips/chip_4608_1792.tif',
'./chips/chip_4864_3840.tif',
'./chips/chip_256_2304.tif',
'./chips/chip_4352_1280.tif',
'./chips/chip_512_3840.tif',
'./chips/chip_3328_2560.tif',
'./chips/chip_2560_4352.tif',
'./chips/chip_4096_4352.tif',
'./chips/chip_3840_0.tif',
'./chips/chip_4864_512.tif',
'./chips/chip_1536_4864.tif',
'./chips/chip_4864_2816.tif',
'./chips/chip_4864_3328.tif',
'./chips/chip_512_256.tif',
```

```
 './chips/chip_2048_3840.tif',
 './chips/chip_2304_2304.tif',
 './chips/chip_3328_2304.tif',
 './chips/chip_3328_256.tif',
 './chips/chip_512_3072.tif',
 './chips/chip_4096_256.tif',
 './chips/chip_256_3840.tif',
 './chips/chip_768_4864.tif',
 './chips/chip_2048_3328.tif',
 './chips/chip_4096_2304.tif',
 './chips/chip_4864_4608.tif',
 './chips/chip_3072_3584.tif',
 './chips/chip_3072_4352.tif',
 './chips/chip_3840_4096.tif',
 './chips/chip_4096_1280.tif',
 './chips/chip_256_256.tif',
 './chips/chip_4096_3840.tif',
 './chips/chip_3584_2816.tif',
 './chips/chip_4352_4608.tif',
 './chips/chip_3840_4352.tif',
 './chips/chip_2816_3584.tif',
 './chips/chip_1280_4608.tif',
 './chips/chip_1280_4864.tif',
 './chips/chip_4352_3328.tif',
 './chips/chip_1792_2816.tif',
 './chips/chip_1280_3840.tif',
 './chips/chip_3584_1536.tif',
 './chips/chip_4096_4096.tif',
 './chips/chip_512_4096.tif',
 './chips/chip_3840_1536.tif',
 './chips/chip_1024_2304.tif',
 './chips/chip_0_0.tif',
 './chips/chip_1024_0.tif']
```

[29]:
```
chips_filenames = glob.glob('./chips/*')
print(chips_filenames[45])
chip_data = rasterio.open(chips_filenames[45])
show(chip_data)
```

```
./chips/chip_512_768.tif
```


−25.24
−25.25
−25.26
−25.27
−25.28
−25.29
−25.30
−62.01
−62.00
−61.99
−61.98
−61.97
−61.96
−61.95

[29]: <Axes: >