s2lx Documentation

Release 0.1.0

Contributors

Jan 03, 2024

CONTENTS

s2lx is a python toolbox to simplify explorative work with Sentinel-2 multispectral imagery. It allows quickly extract data from region of interest, easily manipulate a test processing and visualiztion strategies. In addition, it could be used for quick merging and reprojecting of data.

CHAPTER

ONE

INSTALLATION

Easiest way to install **s2lx** is to use conda (or mamba) package management system. For convinience you can create separate conda environment using included *environment.yml* file:

```
$ conda env create -f environment.yml
```

Then activate the new environment:

```
$ conda activate s2lx
```

and install s2lx using pip:

```
(s2lx)$ pip install https://github.com/ondrolexa/s2lx/archive/master.
↪zip
```

To upgrade an already installed **s2lx** to the latest release:

```
(s2lx)$ pip install --upgrade https://github.com/ondrolexa/s2lx/
↪archive/master.zip
```

In any other scenario, the *s2lx* requires following dependencies:

- *python>=3.9*
- *numpy*
- *matplotlib*
- *scipy*
- *gdal*
- *shapely>=2*
- *pyproj*
- *scikit-image*
- *scikit-learn*

CHAPTER

TWO

USAGE

Start by downloading Sentinel-2 Level-2A or Level-1C product in SAFE format (e.g. from and unzip. Open Python prompt in *s2lx* environment and import classes from *s2lx*:

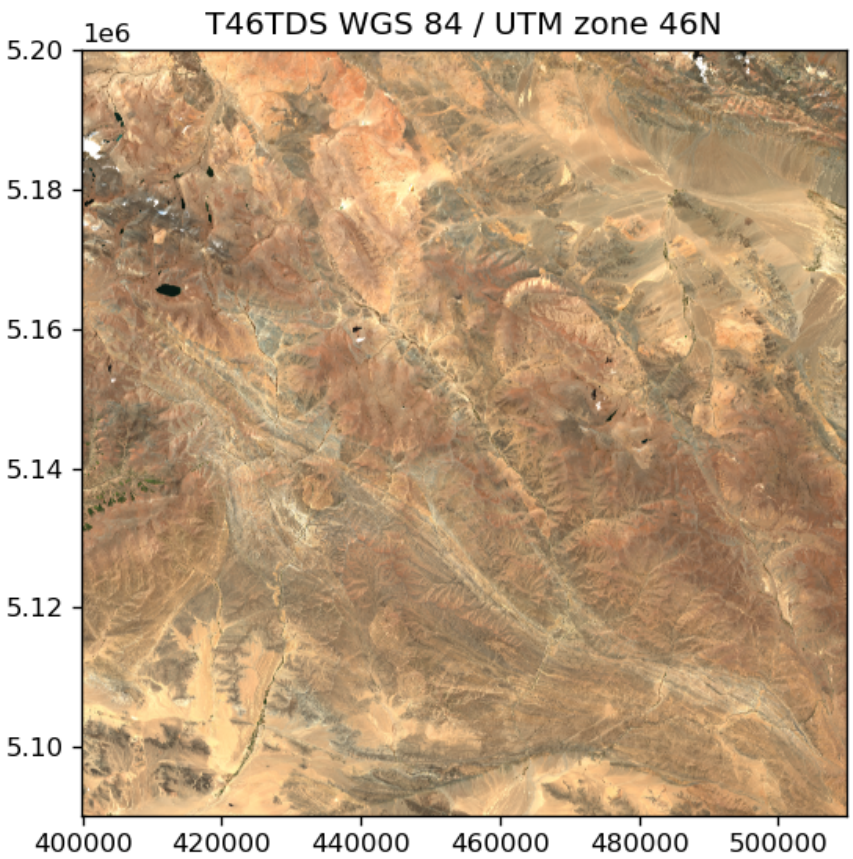
```
>>> from s2lx import *
```

Open SAFE data:

```
>>> s = SAFE('/path/to/safename.SAFE/MTD_MSIL2A.xml')
```

You can preview whole scene:

```
>>> s.preview()
```



Clip region of interest (note that bounds are defined in coordinate system of scene) and store in *S2* collection:

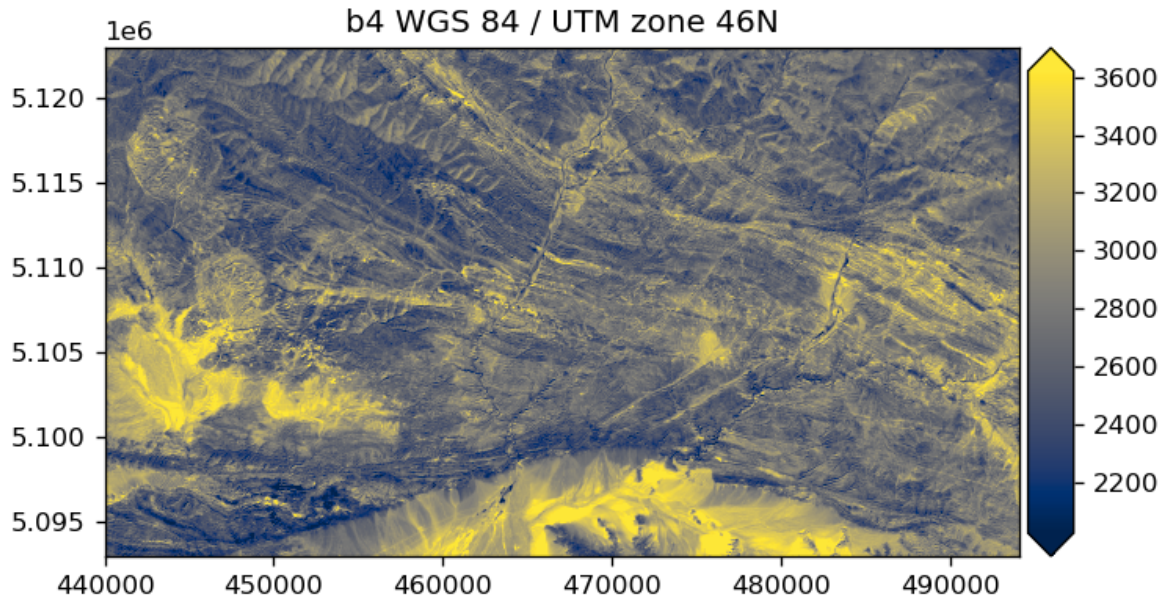
```
>>> bounds = (440000, 5093000, 494000, 5123000)  # (minx, miny, maxx,␣
↪maxy)
>>> d = s.clip(bounds, name='My Region')
```

To see the list of bands:

```
>>> d.bands
['b11', 'b12', 'b2', 'b3', 'b4', 'b5', 'b6', 'b7', 'b8']
```

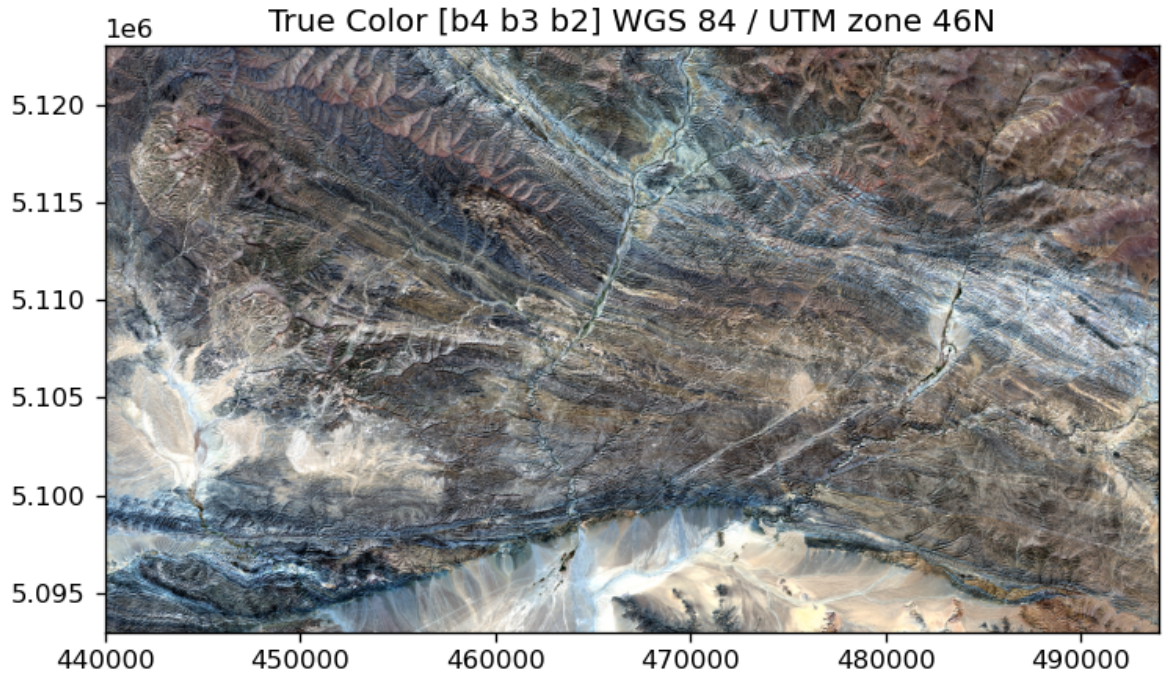
Individual bands could ba accessed as properties:

```
>>> d.b4.show()
```



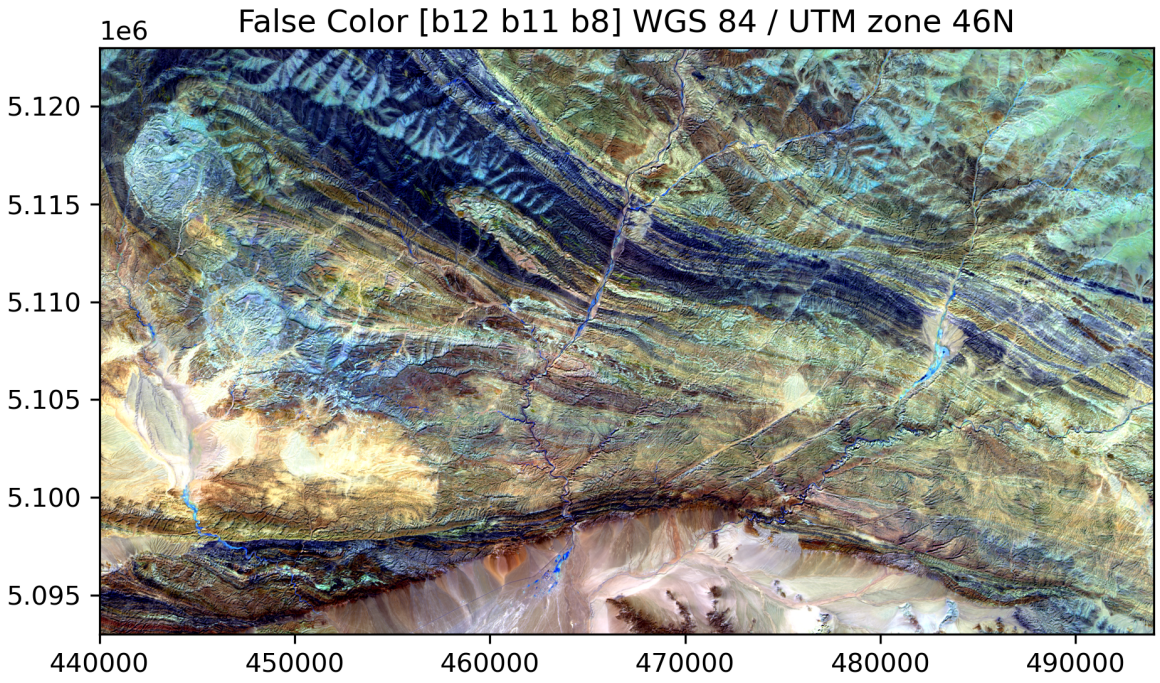
You can use *Composite* class to create RGB composite:

```
>>> rgb = Composite(d.b4, d.b3, d.b2, name='True Color')
>>> rgb.show()
```



or:

```
>>> rgb = Composite(d.b12, d.b11, d.b8, name='False Color')
>>> rgb.show()
```

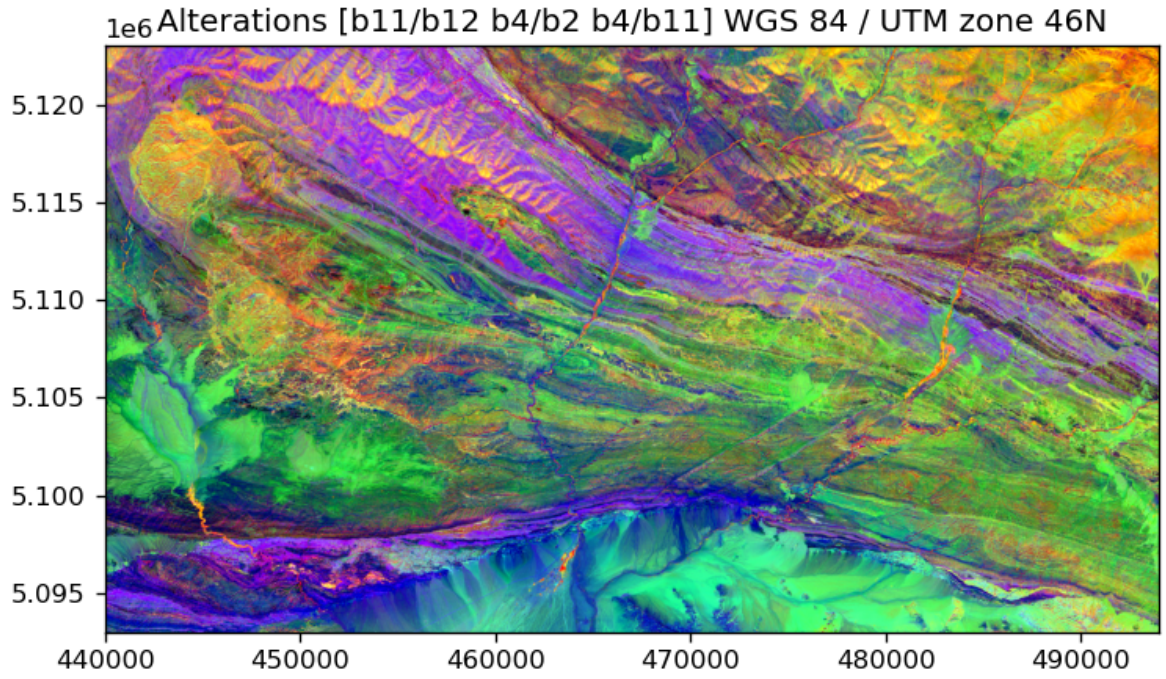


Bands and composites could be saved to GeoTIFF with *save* method:

```
>>> d.b4.save('b4.tif')
>>> rgb.save('truecolor.tif')
```

Bands support simple mathematical operations (addition, subtraction, division, multiplication)

```
>>> alt = Composite(d.b11/d.b12, d.b4/d.b2, d.b4/d.b11, name=
↪'Alterations')
>>> alt.show()
```



Bands could be filtered (check *s2lx.s2filters* for possible filters):

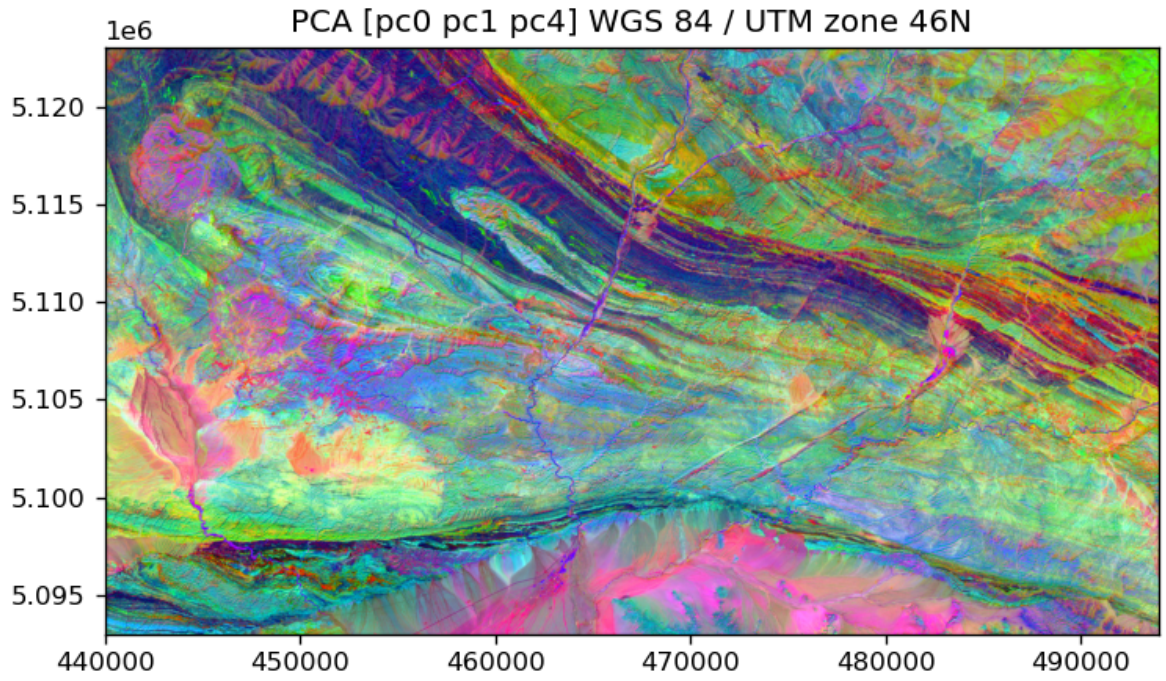
```
>>> medfilter = median_filter(radius=4)
>>> b12f = d.b12.apply(medfilter)
```

You can do PCA analysis using *S2.pca* method:

```
>>> p = d.pca()
```

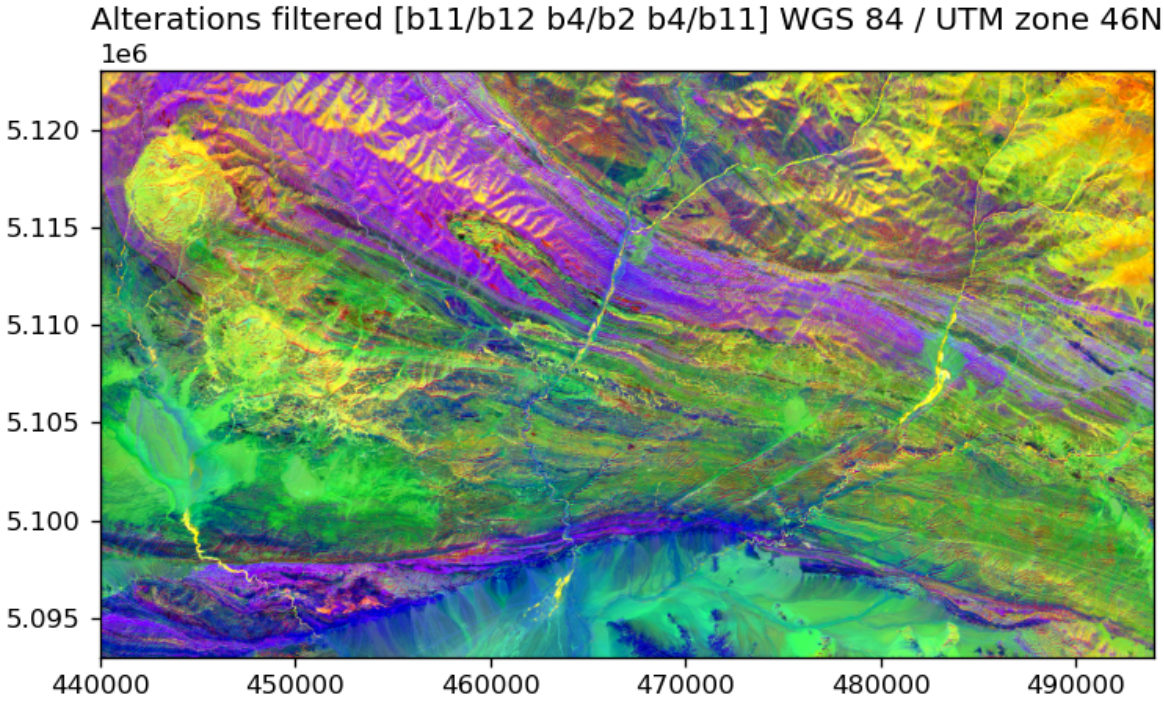
To create RGB composite from first three principal components:

```
>>> pca = Composite(p.pc0, p.pc1, p.pc4, name='PCA')
>>> pca.show()
```



You can use also PCA to filter your data, i.,e. use only few PC to reconstruct dataset. Here we remove last four (from 9) components with lowest explained variance and reconstruct all bands:

```
>>> r = d.restored_pca(remove=(5,6,7,8))
>>> altr = Composite(r.b11/r.b12, r.b4/r.b2, r.b4/r.b11, name=
↪'Alterations filtered')
>>> altr.show()
```


Alterations filtered [b11/b12 b4/b2 b4/b11] WGS 84 / UTM zone 46N
1e6
5.120
5.115
5.110
5.105
5.100
5.095
440000
450000
460000
470000
480000
490000

CHAPTER

THREE

API

s2lx provides following classes:

class s2lx.s2classes.**Band**(*name*, *array*, *transform*, *projection*, ***kwargs*)

Bases: object

Class to store band data

Raster band, internally stored as masked array, to support ROI operations. Bands support basic mathematical operations.

Parameters

- **name** (*str*) – name of the band. Name must start with a letter or the underscore character (not a number) and can only contain alpha-numeric characters and underscores.
- **array** (*numpy.ma.array*) – band data as 2d numpy masked array
- **transform** (*tuple*) – 6 coefficients geotransform. Rotation not supported
- **projection** (*str*) – CRS in WKT

Keyword Arguments

- **vmin** (*float*) – minimum value for color normalization. Default 2 percentile
- **vmax** (*float*) – maximum value for color normalization. Default 98 percentile

shape

shape of raster array

Type
tuple

vmin

minimum value used for colorscale. Default 2% percentile

Type
float

vmax

maximum value used for colorscale. Default 98% percentile

Type
float

apply(*fun*)

Apply function to raster data

You can use pre-defined filters from *s2lx.s2filters*, but any other function accepting 2D numpy array could be used.

Parameters
fun – function

Returns
s2lx.Band raster band with new values

astype(*dtype*)

Convert band to other dtype

Parameters
dtype – numpy dtype

Returns
s2lx.Band raster band

copy(***kwargs*)

Create copy of band

Transform and projection is not changed.

Keyword Arguments

- **name** (*str*) – New name. Default is original one
- **array** (*numpy.ma.array*) – New data. De3fault is original one

property dtype

type of data in band

Type
numpy.dtype

intersect(*other*)

Intersect bands

Returns tuple of two bands with all values masked except intersecting region. Both bands must have same transform and projection.

Parameters
other – *s2lx.Band* raster

Returns
tuple of two *s2lx.Band* raster bands

property max

Returns maximum of data

property min

Returns minimum of data

property norm

Returns matplotlib.colors.Normalize using vmin, vmax properties

property normalized

Returns normalized raster values as numpy.array

patch(*other*, *fit=False*)

Patch bands

All masked data are patched from other band. Both bands must have same transform and projection.

Parameters
other – *s2lx.Band* raster

Keyword Arguments
fit (*bool*) – If True, the patching dataset bands are linearly scaled to fit in overlapping region. Default False

Returns
s2lx.Band patched raster band

save(*filename*, *overviews=False*)

Save band as GeoTIFF

Parameters
filename (*str*) – GeoTIFF filename

Keyword Arguments
overviews (*bool*) – build overviews when True. Default False

setnorm(***kwargs*)

Set color normalization minimum a maximum

Keyword Arguments

- **vmin** (*float*) – minimum value for color normalization. Default 2 percentile
- **vmax** (*float*) – maximum value for color normalization. Default 98 percentile

show(***kwargs*)

Show band

Create matplotlib figure with raster band data and colorbar.

Keyword Arguments

- **figsize** (*tuple*) – figure size. Default is matplotlib defaults

- **filename** (*str*) – if not None, the plot is save to file. Default None
- **dpi** (*int*) – DPI for save image. Default 300
- **cmap** – matplotlib color map. Default 'cividis'
- **under** (*color*) – color used for values under vmin. Default cmap(0)
- **over** (*color*) – color used for values above vmax. Default cmap(1)
- **masked** (*color*) – color used for masked values. Default 'white'

property values

Return 1D array of non-masked values from band

Type
numpy.array

class s2lx.s2classes.**Composite**(*r*, *g*, *b*, *name='RGB composite'*)

Bases: object

Class to store RGB composite

Parameters

- **r** (Band) – band used for red channel
- **g** (Band) – band used for green channel
- **b** (Band) – band used for blue channel

Keyword Arguments
name (*str*) – name of the RGB composite

shape

shape of raster array

Type
tuple

transform

6 coefficients geotransform. Rotation not supported

Type
tuple

projection

CRS in WKT

Type
str

save(*filename*, *overviews=False*)

Save RGB composite as RGBA GeoTIFF

Alpha channel is generated from mask of bands.

Parameters
filename (*str*) – GeoTIFF filename

Keyword Arguments
overviews (*bool*) – build overviews when True. Default False

show(***kwargs*)

Show RGB composite

Create matplotlib figure with RGB composite.

Keyword Arguments

- **figsize** (*tuple*) – figure size. Default is matplotlib defaults
- **filename** (*str*) – if not None, the plot is save to file. Default None
- **dpi** (*int*) – DPI for save image. Default 300

class s2lx.s2classes.**S2**(**rasters*, ***kwargs*)

Bases: object

Class to store homogeneous collection of bands

All bands in collection share geographic reference, size and resolution. Individual bands in collection could be accessed by dot notation.

Parameters
***args** – any number of *Band* instances

Keyword Arguments

- **name** (*str*) – mame of collection. Default is 'S2'
- **meta** (*dict*) – Dictionary with metadata. Default is {}

transform

6 coefficients geotransform. Rotation not supported

Type
tuple

projection

CRS in WKT

Type
str

Examples

To acceess band, just use band name

```
>>> d.bands
['b11', 'b12', 'b2', 'b3', 'b4', 'b5', 'b6', 'b7', 'b8']
>>> d.b12 / d.b8
Band b12/b8 (904, 1041) float64
Min:0.500583 Max:1.86126 Vmin:0.962256 Vmax:1.40753
```

property bands

Sorted list of band names

Type
list

patch(*other*, *fit=False*)

Patch collection

All masked regions are patched from other

Parameters
other – *s2lx.S2* collection

Keyword Arguments
fit (*bool*) – If True, the patching dataset bands are linearly scaled to fit in overlapping region. Default False

Returns
s2lx.S2 collection

pca(***kwargs*)

PCA analysis

Calculate principal components from all bands in collection

Keyword Arguments

- **centered** (*bool*) – If True values are centered
- **n** (*int*) – Number of pricipal components calculated. Default is number of bands.

Returns

***s2lx.S2* collection. Components, explained variance and** explained variance ratio are available in metadata

restored_pca(***kwargs*)

PCA based filtering

Use PCA components with cumulative explained variance given by treshold. Aternatively, PCA components to be excluded from reconstruction could be defined.

Keyword Arguments

- **remove** (*int or list*) – PCA components to be removed
- **threshold** (*float*) – Threshold of explained variance in percents to be reconstructed. Default 98.

Returns

***s2lx.S2* collection. Components, explained variance and** explained variance ratio are available in metadata

class s2lx.s2classes.**SAFE**(*xml*)

Bases: object

Class to manipulate Sentinel-2 product

The SAFE format has been designed to act as a common format for archiving and conveying data within ESA Earth Observation archiving facilities. The SAFE format wraps a folder containing image data in a binary data format and product metadata in XML.

The Level-2A prototype product is an orthorectified product providing Bottom-Of-Atmosphere (BOA) reflectances, and basic pixel classification (including classes for different types of cloud). The generation of this prototype product is carried out by the User from Level-1C products.

The Level-2A image data product uses the same tiling, encoding and filing structure as Level-1C.

datasets

Dictionary containing information abou available sub- datasets. Commonly "10m", "20m", "60m" and "TCI"

Type
dict

meta

Dictionary with SAFE format metadata

Type
dict

clip(*bounds*, *res=20*, *name='Clip'*, *include8a=False*)

Clip scene by extent coordinates

Clip all bands in scene by rectangular bound (minX, minY, maxX, maxY). All bands in resulting collection have same resolution. For res=20, the '10m' dataset bands are downsampled, while for res=10, the bands of '20m' dataset are upsampled.

Note: Coordinates of the extent of the output are automatically aligned to multiples of resolution. This method also assume that coordinates of subwindow are given in CRS of scene

---

Parameters
bounds (*tuple*) – output bounds as (minX, minY, maxX, maxY) in target CRS

Keyword Arguments

- **res** (*int*) – resolution of clipped bands. Default 20
- **name** (*str*) – name of collection. Default is 'Clip'
- **include8a** (*bool*) – whether to include B8A band. Dafault False

Returns
s2lx.S2 collection

`clip_features`(*filename*, *driverName='GeoJSON'*, *res=20*, *name='Clip'*, *include8a=False*)

Clip scene to features extent in vector file

Clip all bands in scene by rectangular extent of features stored in vector file. All bands in resulting collection have same resolution. For res=20, the '10m' dataset bands are downsampled, while for res=10, the bands of '20m' dataset are upsampled.

Note: This method assume that CRS of vector file and scene is identical

Parameters
filename (*str*) – filename of vector file

Keyword Arguments

- **driverName** (*str*) – format of vector file. Default is 'GeoJSON' For available options see *ogr2ogr –formats*
- **res** (*int*) – resolution of clipped bands. Default 20
- **name** (*str*) – name of collection. Default is 'Clip'
- **include8a** (*bool*) – whether to include B8A band. Dafault False

Returns
s2lx.S2 collection

`clip_shape`(*shape*, *srccrs=None*, *res=20*, *name='Clip'*, *include8a=False*)

Clip scene to features extent in vector file

Clip all bands in scene by bounding box of shapely polygon. All bands in resulting collection have same resolution. For res=20, the '10m' dataset bands are downsampled, while for res=10, the bands of '20m' dataset are upsampled.

Parameters
shape – shapely polygon

Keyword Arguments

- **srccrs** (*str or pyproj.CRS*) – shape coordinate system. Default is CRS of scene
- **res** (*int*) – resolution of clipped bands. Default 20
- **name** (*str*) – name of collection. Default is 'Clip'
- **include8a** (*bool*) – whether to include B8A band. Dafault False

Returns
s2lx.S2 collection

coverage(*filename*, *driverName='GeoJSON'*)

Return area fraction of features in filename covered by scene

Parameters
filename (*str*) – filename of vector file

Keyword Arguments
driverName (*str*) – format of vector file. Default is 'GeoJSON' For available options see *ogr2ogr –formats*

Returns
float value of coverage (between 0 and 1)

property crs

coordinate reference system of scene

Type
pyproj.CRS

footprint(*dstcrs=None*)

Get scene footprint

Keyword Arguments
dstcrs (*str or pyproj.CRS*) – coordinate system of footprint. Default is scene CRS.

Returns
Shapely polygon feature

gclip(*name='Clip'*, *include8a=False*, *band='B12'*)

Quick clip by rectangular selection

Clip all bands in scene by rectangular selection drawn by mouse. All bands in resulting collection have 20m resolution and '10m' dataset bands are downsampled.

Note: Draw and modify selection by mouse. Clip by keypress enter.

Keyword Arguments

- **name** (*str*) – name of collection. Default is 'Clip'
- **include8a** (*bool*) – whether to include B8A band. Dafault False

Returns
s2lx.S2 collection

get(*dataset*, *band*)

Get band as numpy array

Parameters

- **dataset** (*str*) – matplotlib Figure size
- **band** (*str*) – matplotlib Figure size

Returns
Tuple of band numpy array and metadata dict

`intersection`(*other*, *dstcrs=None*)

Return intersections of scene footprints as shapely polygon

Parameters
other – *s2lx.SAFE* scene

Keyword Arguments
dstcrs (*str or pyproj.CRS*) – coordinate system for intersection. Default is scene CRS.

Returns
shapely polygon

`overlap`(*other*)

Check if scene footprints overlap

Parameters
other – *s2lx.SAFE* scene

Returns
True is footprints overlap, otherwise False

`preview`(***kwargs*)

Show the scene TCI image

Keyword Arguments

- **figsize** (*tuple*) – matplotlib Figure size
- **filename** (*str*) – if not None, the plot is save to file. Default None
- **dpi** (*int*) – DPI for save image. Default 300

`warp`(*bounds*, *dstcrs*, *res=20*, *name='Clip'*, *include8a=False*, *cutlineLayer=None*, *driverName='GeoJSON'*)

Reproject and clip scene by by coordinates

Reproject all bands in scene to target CRS and clip to rectangular window defined by coordinates. All bands in resulting collection have same resolution. For res=20, the '10m' dataset bands are downsampled, while for res=10, the bands of '20m' dataset are upsampled.

Note: Coordinates of the extent of the output are automatically aligned to multiples of resolution. This method also assume that bound coordinates are given in target CRS.

Parameters

- **bounds** (*tuple*) – output bounds as (minX, minY, maxX, maxY) in target CRS

- **dstcrs** (*str or pyproj.CRS*) – target coordinate system

Keyword Arguments

- **res** (*int*) – resolution of clipped bands. Default 20
- **name** (*str*) – name of collection. Default is 'Clip'
- **include8a** (*bool*) – whether to include B8A band. Dafault False
- **cutlineLayer** (*str*) – filename of vector file. Default None All pixels out of cutlineLayer are masked
- **driverName** (*str*) – format of vector file. Default is 'GeoJSON' For available options see *ogr2ogr –formats*

Returns
s2lx.S2 collection

warp_features(*filename*, *dstcrs=None*, *driverName='GeoJSON'*, *res=20*, *name='Clip'*, *include8a=False*, *crop=True*)

Reproject and clip scene to extent of features in vector file

Reproject all bands in scene to target CRS and clip to rectangular region defined by extent of features in vector file. If *crop* is True, the bands are cropped to outline of features. All bands in resulting collection have same resolution. For res=20, the '10m' dataset bands are downsampled, while for res=10, the bands of '20m' dataset are upsampled.

Note: For vector formats without stored CRS information, the method assume that coordinates coincide with scene CRS

Parameters
filename (*str*) – filename of vector file

Keyword Arguments

- **driverName** (*str*) – format of vector file. Default is 'GeoJSON' For available options see *ogr2ogr –formats*
- **dstcrs** (*str or pyproj.CRS*) – target coordinate system. Default is CRS of vector file
- **res** (*int*) – resolution of clipped bands. Default 20
- **name** (*str*) – name of collection. Default is 'Clip'
- **include8a** (*bool*) – whether to include B8A band. Default False
- **crop** (*bool*) – whether to crop to polygon outline. Default False

Returns
s2lx.S2 collection

warp_shape(*shape*, *srccrs=None*, *dstcrs=None*, *res=20*, *name='Clip'*, *include8a=False*)

Reproject and clip scene to extent of shapely polygon

Reproject all bands in scene to target CRS and clip to bounding box of shapely polygon. All bands in resulting collection have same resolution. For res=20, the '10m' dataset bands are downsampled, while for res=10, the bands of '20m' dataset are upsampled.

Parameters
shape (*str*) – shapely polygon

Keyword Arguments

- **srccrs** (*str or pyproj.CRS*) – shape coordinate system. Default is CRS of scene
- **dstcrs** (*str or pyproj.CRS*) – target coordinate system. Default is srccrs
- **res** (*int*) – resolution of clipped bands. Default 20
- **name** (*str*) – name of collection. Default is 'Clip'
- **include8a** (*bool*) – whether to include B8A band. Default False

Returns
s2lx.S2 collection

class s2lx.s2classes.**SAFEStore**(*path*)

Bases: object

Class to manipulate Sentinel-2 product store

The SAFEStore is a folder where all SAFE products ares stored.

path

Path to directory

Type
Path

SAFES

List of all SAFE directories in store

Type
list

tiles

List of all tile numbers in store

Type
list

s2lx.filters provides processing functions, which could be used for *s2lx.Band.apply* method:

CHAPTER

FOUR

CHANGELOG

All notable s2lx changes.

4.1 [0.1.0] - master

- initial version
- coordinates on figures are in band projection

PYTHON MODULE INDEX

S

INDEX

T

V

W