ukis$_{pysat}$

Jan 12, 2022

Contents:

1 data **3**

2 file **5**

3 raster **7**

4 Example **9**
4.1 Dimension order 10

5 STAC API Item IO Concept **11**
5.1 Environment variables 11

6 Indices and tables **27**

Python Module Index **29**

Index **31**

The UKIS-pysat package provides generic classes and functions to query, access and process multi-spectral and SAR satellite images.

CHAPTER 1

data

Download satellites data from different sources (currently Earth Explorer, SciHub, STAC), deal with and structure metadata.

CHAPTER 2

file

Work with you local satellite data files and read information out of file names and metadata files. Currently, focusing on Sentinel-1.

CHAPTER 3

raster

Reading satellite data and performing simple, but cumbersome tasks. This is just a layer on top of rasterio for stuff we often need. It can very well be that using *rasterio* directly is often the better choice.

CHAPTER 4

Example

Here's an example about some basic features:

```
from ukis_pysat.data import Source
from ukis_pysat.file import get_sentinel_scene_from_dir
from ukis_pysat.members import Datahub, Platform
from ukis_pysat.raster import Image

# connect to Copernicus Open Access Hub  and query metadata
src = Source(Datahub.Scihub)
meta = src.query_metadata(
    platform=Platform.Sentinel2,
    date=("20200101", "NOW"),
    aoi=(11.90, 51.46, 11.94, 51.50),
    cloud_cover=(0, 50),
)
for item in meta:  # item is now a PySTAC item
    print(item.id)
    uuid = item.properties["srcuuid"]

    # download geocoded quicklook and image
    src.download_quicklook(product_uuid=uuid, target_dir="/users/username/tmp")
    src.download_image(product_uuid=uuid, target_dir="/users/username/tmp")

    break

# get sentinel scene from directory
with get_sentinel_scene_from_dir("/users/username/tmp") as (full_path, ident):
    with Image(full_path.join("pre_nrcs.tif")) as img:
        # scale the image array, having one band
        img.arr = img.arr * 0.3
```

4.1 Dimension order

The dimension order can be set upon initialization and you have to choose between *first* (bands, rows, columns), being raster shape needed for rasterio, or *last* (rows, columns, bands), being image shape used by most image processing libraries. Default is *first*. Compare with the documentation of rasterio. UKIS-pysat uses rasterio internally and the dimension order is always reshaped to *first* (bands, rows, columns) if the dimension order has been set as *last*. If the image was initialized with dimension order *last*, the result will be reshaped to *last* (rows, columns, bands) when calling `img.arr`.

Altering the array replaces all bands. If it is intended to alter a particular band, the remaining bands should be copied.

CHAPTER 5

STAC API Item IO Concept

To enable reading from different types of file systems (similar to PySTAC), it is recommended that in the `__init__.py` of the client module, or at the beginning of the script using `ukis_pysat.data`, you overwrite the `STACAPI_IO.read_item_method` with function that read and write however you need. The following is an example for a on premise S3 environment:

```
from pystac import Item
from ukis_pysat.stacapi_io import STACAPI_IO

def on_premise_s3_url_method(feature, root_bucket="dem"):
    """the href is build like /collections/*collection_prefix*/items/*item_prefix*

    At some environments we will need to give back the href according to this method.
    """
    item = Item.from_dict(feature)
    href = item.get_self_href()
    stripped_href = href.replace(r"collections/", "").replace(r"items/", "")

    return Item.from_file(f"s3://{root_bucket}{stripped_href}/{item.id}.json")

STACAPI_IO.read_item_method = on_premise_s3_url_method
```

5.1 Environment variables

To use `ukis_pysat.data` and to download from the respective Datahub you need to set the credentials as environment variables.

For EarthExplorer that's:

```
EARTHEXPLORER_USER=your_username
EARTHEXPLORER_PW=your_password
```

For SciHub that's:

```
SCIHUB_USER=your_username
SCIHUB_PW=your_password
```

5.1.1 Installation

Installation with pip

Most users will want to do this:

```
pip install ukis-pysat[complete]  # install everything
```

There's also some lighter versions with less dependencies:

```
pip install ukis-pysat  # only install core dependencies (ukis_pysat.file can be used)

pip install ukis-pysat[data]  # also install dependencies for ukis_pysat.data

pip install ukis-pysat[raster]  # also install dependencies for ukis_pysat.raster
```

Some helper functions might need additional dependencies like *pandas*, *dask[array]* or *utm*. If this is the case you will receive an *ImportError*.

GDAL

If you're having troubles installing GDAL and Rasterio use conda and/or follow these instructions.

Tests

To run the tests set the Environment variables and then:

```
git clone https://github.com/dlr-eoc/ukis-pysat
cd ukis-pysat
pip install -e .[dev]
python -m unittest discover tests
```

If you set the environment variables with the credentials to the hubs, you can uncomment *@unittest.skip()* for these tests.

5.1.2 Python API Reference

ukis_pysat.data

class `ukis_pysat.data.`**`Source`**(*datahub*, *catalog=None*)

Bases: `object`

Provides methods to query data sources for metadata and download images and quicklooks (APIs only). Remote APIs and local data directories that hold metadata files are supported.

Parameters

- **`datahub`** – Data source (<enum 'Datahub'>).

- **catalog** – Only applicable if datahub is 'STAC_local'. Can be one of the following types: Path to STAC Catalog file catalog.json (String, Path). Pystac Catalog or Collection object (pystac.catalog.Catalog, pystac.collection.Collection). None initializes an empty catalog. (default: None)

add_items_from_directory(*item_dir*, *item_glob='*.json'*)

Adds STAC items from a directory to a STAC Catalog.

Parameters

- **item_dir** – Path to directory that holds the STAC items (String).
- **item_glob** – Optional glob pattern to identify STAC items in directory (String), (default: '*.json').

close()

Closes connection to or logs out of Datahub.

construct_metadata(*meta*, *platform*)

Constructs a STAC item that is harmonized across the different satellite image sources.

Parameters

- **meta** – Source metadata (GeoJSON-like mapping)
- **platform** – Image platform (<enum 'Platform'>).

Returns PySTAC item

download_image(*product_uuid*, *target_dir*)

Downloads satellite image data to a target directory for a specific product_id. Incomplete downloads are continued and complete files are skipped.

Parameters

- **product_uuid** – UUID of the satellite image product (String).
- **target_dir** – Target directory that holds the downloaded images (String, Path)

download_quicklook(*product_uuid*, *target_dir*)

Downloads a quicklook of the satellite image to a target directory for a specific product_id. It performs a very rough geocoding of the quicklooks by shifting the image to the location of the footprint.

Parameters

- **product_uuid** – UUID of the satellite image product (String).
- **target_dir** – Target directory that holds the downloaded images (String, Path)

query_metadata(*platform*, *date*, *aoi*, *cloud_cover=None*, *kwargs=None*)

Queries metadata from data source.

Parameters

- **platform** – Image platform (<enum 'Platform'>).
- **date** – Date from - to in format yyyyMMdd (String or Datetime tuple).
- **aoi** – Area of interest as GeoJson file or bounding box tuple with lat lon coordinates (String, Tuple).
- **cloud_cover** – Percent cloud cover scene from - to (Integer tuple) (default: None).
- **kwargs** – Dictionary of the additional requirements for the hub used (default: None).

Generates Metadata item of products that match query criteria (PySTAC item).

query_metadata_srcid (*platform*, *srcid*)
Queries metadata from data source by srcid.

Parameters

- **platform** – Image platform (<enum 'Platform'>).
- **srcid** – Srcid of a specific product (String).

Generates Metadata of product that matches srcid (PySTAC item).

stream_image (*product_uuid*)

Parameters **product_uuid** – UUID of the satellite image product (String).

Returns response

ukis_pysat.stacapi

ukis_pysat.file

ukis_pysat.file.**env_get** (*key: str*, *boolean: bool = False*) → Union[str, bool]
get an environment variable or fail with a meaningful error message

Parameters

- **key** – name of environment variable
- **boolean** – bool (default: False), optional. Returns true if key in ["true", "y", "yes", "1"].

Returns str or boolean

ukis_pysat.file.**get_footprint_from_manifest** (*xml_path: Union[str, pathlib.Path]*) → Any
Return a shapely polygon with footprint of scene, tested for Sentinel-1.

Parameters **xml_path** – path to manifest.safe

Returns shapely polygon

```
>>> get_footprint_from_manifest(Path(__file__).parents[1] / "tests/testfiles/
↪manifest.safe").wkt
'POLYGON ((149.766922 -24.439564, 153.728622 -23.51771, 154.075058 -24.737713,␣
↪150.077042 -25.668921, 149.766922 -24.439564))'
```

ukis_pysat.file.**get_ipf_from_manifest** (*xml_path: Union[str, pathlib.Path]*) → float
Get IPF version from manifest file, tested for Sentinel-1.

Parameters **xml_path** – path to manifest.safe

Returns ipf version (float)

```
>>> get_ipf_from_manifest(Path(__file__).parents[1] / "tests/testfiles/manifest.
↪safe")
2.82
```

ukis_pysat.file.**get_origin_from_manifest** (*xml_path: Union[str, pathlib.Path]*) → str
Get origin from manifest file, tested for Sentinel-1.

Parameters **xml_path** – path to manifest.safe

Returns country of origin

```
>>> get_origin_from_manifest(Path(__file__).parents[1] / "tests/testfiles/
↪manifest.safe")
'United Kingdom'
```

ukis_pysat.file.**get_pixel_spacing**(*scenedir: Union[str, pathlib.Path], polarization: str = 'HH'*) → Tuple[float, float]

Get pixel spacing, tested for Sentinel-1.

Parameters

- **scenedir** – path to unzipped SAFE-directory of scene
- **polarization** – str (default: 'HH')

Returns tuple with pixel spacing in meters and degrees as floats

```
>>> get_pixel_spacing(Path(__file__).parents[1] / "tests/testfiles")
(40.0, 0.0003593261136478086)
```

ukis_pysat.file.**get_polarization_from_s1_filename**(*filename: str, dual: bool = False*) → str

Get polarization from the filename of a Sentinel-1 scene. https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar/naming-conventions.

Parameters

- **filename** – top-level SENTINEL-1 product folder name
- **dual** – boolean (default: True), optional

Returns str

```
>>> get_polarization_from_s1_filename("MMM_BB_TTTR_1SDH_YYYYMMDDTHHMMSS_
↪YYYYMMDDTHHMMSS_OOOOOO_DDDDDD_CCCC.SAFE.zip")
'HH'
>>> get_polarization_from_s1_filename("MMM_BB_TTTR_1SSH_YYYYMMDDTHHMMSS_
↪YYYYMMDDTHHMMSS_OOOOOO_DDDDDD_CCCC.SAFE.zip")
'HH'
>>> get_polarization_from_s1_filename("MMM_BB_TTTR_2SSV_YYYYMMDDTHHMMSS_
↪YYYYMMDDTHHMMSS_OOOOOO_DDDDDD_CCCC.SAFE.zip")
'VV'
>>> get_polarization_from_s1_filename("MMM_BB_TTTR_1SDV_YYYYMMDDTHHMMSS_
↪YYYYMMDDTHHMMSS_OOOOOO_DDDDDD_CCCC.SAFE.zip", True)
'VV,VH'
```

ukis_pysat.file.**get_proj_string**(*footprint: Any*) → str

Get UTM projection string the centroid of footprint is located in. Footprint itself might cover multiple UTM zones.

Parameters **footprint** – shapely polygon

Returns string with information about projection

```
>>> get_proj_string(get_footprint_from_manifest(Path(__file__).parents[1] /
↪"tests/testfiles/manifest.safe"))
'+proj=utm +zone=56J, +ellps=WGS84 +datum=WGS84 +units=m +no_defs'
```

ukis_pysat.file.**get_sat_ts_from_datetime**(*dt: datetime.datetime, dformat: str = '%Y%m%dT%H%M%S'*) → str

Get ESA timestamp string (used in their filenames) from datetime object.

Parameters

- **dt** – datetime.datetime object
- **dformat** – : str, (default: %Y%m%dT%H%M%S)

Returns ESA timestamp as string

```
>>> get_sat_ts_from_datetime(datetime(2020, 1, 13, 7, 46, 19, tzinfo=timezone.utc)
'20200113T074619'
```

ukis_pysat.file.**get_sentinel_scene_from_dir**(*indir: Union[str, pathlib.Path]*) → Iterator[Tuple[pathlib.Path, str]]

Scan directory for s1 scenes, unzips them if necessary. Tested with Sentinel-1, -2 & -3.

Parameters **indir** – path to zipped S1 scene or directory with S1 scene

Yields full_path (directory with scene, str), ident (filename of scene, str)

```
>>> with get_sentinel_scene_from_dir(Path(__file__).parents[1] / "tests/testfiles
↪") as (fp, name):
...     print(name)
S1M_hello_from_inside
```

ukis_pysat.file.**get_ts_from_sentinel_filename**(*filename: str, start_date: bool = True, dformat: str = '%Y%m%dT%H%M%S'*) → datetime.datetime

Get timestamp from the filename of a Sentinel scene, according to naming conventions. Currently works for S1, S2 & S3.

Parameters

- **filename** – top-level SENTINEL product folder or file name
- **start_date** – boolean (default: True), False is Stop Date, optional
- **dformat** – str, (default: %Y%m%dT%H%M%S)

Returns datetime.datetime object with timezone information

```
>>> get_ts_from_sentinel_filename("S1M_BB_TTTR_LFPP_20200113T074619_
↪YYYYMMDDTHHMMSS_OOOOOO_DDDDDD_CCCC.SAFE.zip")
datetime.datetime(2020, 1, 13, 7, 46, 19, tzinfo=datetime.timezone.utc)
>>> get_ts_from_sentinel_filename("S1M_BB_TTTR_LFPP_YYYYMMDDTHHMMSS_
↪20200113T002219_OOOOOO_DDDDDD_CCCC.SAFE.zip", False)
datetime.datetime(2020, 1, 13, 0, 22, 19, tzinfo=datetime.timezone.utc)
>>> get_ts_from_sentinel_filename("S3M_OL_L_TTT____20200113T074619_
↪YYYYMMDDTHHMMSS_YYYYMMDDTHHMMSS_i_GGG_c.SEN3")
datetime.datetime(2020, 1, 13, 7, 46, 19, tzinfo=datetime.timezone.utc)
>>> get_ts_from_sentinel_filename("S3M_OL_L_TTTTTT_yyyymmddThhmmss_
↪20200113T074619_YYYYMMDDTHHMMSS_i_GGG_c.SEN3", False)
datetime.datetime(2020, 1, 13, 7, 46, 19, tzinfo=datetime.timezone.utc)
>>> get_ts_from_sentinel_filename("S2AM_MSIXXX_20200113T074619_Nxxyy_ROOO_Txxxxx_
↪<Product Discriminator>.SAFE")
datetime.datetime(2020, 1, 13, 7, 46, 19, tzinfo=datetime.timezone.utc)
```

ukis_pysat.raster

class ukis_pysat.raster.**Image**(*data, dimorder='first', crs=None, transform=None, nodata=None*)

Bases: object

Parameters

- **data** – rasterio.io.DatasetReader or path to raster or np.ndarray of shape (bands, rows, columns)
- **dimorder** – Order of channels or bands 'first' or 'last' (default: 'first')
- **crs** – Coordinate reference system used when creating form array. If 'data' is np.ndarray this is required (default: None)
- **transform** – Affine transformation mapping the pixel space to geographic space. If 'data' is np.ndarray this is required (default: None)
- **nodata** – nodata value only used when creating from np.ndarray, otherwise this has no effects, optional (default: None)

arr

array property

close()

closes Image

da_arr = None

dn2toa(*platform*, *mtl_file=None*, *wavelengths=None*)

This method converts digital numbers to top of atmosphere reflectance, like described here: https://www.usgs.gov/land-resources/nli/landsat/using-usgs-landsat-level-1-data-product

Parameters

- **platform** – image platform, possible Platform.Landsat[5, 7, 8] or Platform.Sentinel2 (<enum 'Platform'>).
- **mtl_file** – path to Landsat MTL file that holds the band specific rescale factors (str).
- **wavelengths** – like ["Blue", "Green", "Red", "NIR", "SWIR1", "TIRS", "SWIR2"] for Landsat-5 (list of str).

get_subset(*tile*, *band=0*)

Get slice of array.

Parameters

- **tile** – rasterio.windows.Window tile from get_tiles().
- **band** – Band number (default: 0).

Returns Sliced numpy array, bounding box of array slice.

get_tiles(*width=256*, *height=256*, *overlap=0*)

Calculates rasterio.windows.Window, idea from https://stackoverflow.com/a/54525931

Parameters

- **width** – int, optional (default: 256). Tile size in pixels.
- **height** – int, optional (default: 256). Tile size in pixels.
- **overlap** – int, optional (default: 0). Overlap in pixels.

Yields window of tile

get_valid_data_bbox(*nodata=0*)

bounding box covering the input array's valid data pixels.

Parameters **nodata** – nodata value, optional (default: 0)

Returns tuple with valid data bounds

mask (*bbox*, *crop=True*, *fill=False*, *mode='constant'*, *constant_values=0*)
Mask raster to bbox.

Parameters

- **bbox** – bounding box of type tuple or Shapely Polygon
- **crop** – bool, see rasterio.mask. Optional, (default: True)
- **fill** – enforce raster to cover bbox. if raster extent is smaller than bbox it will be filled according to mode and constant_values parameters. Optional (default: False)
- **mode** – str, how to fill, see rasterio.pad. Optional (default: 'constant')
- **constant_values** – nodata value, padding should be filled with, optional (default: 0)

pad (*pad_width*, *mode='constant'*, *constant_values=0*)
Pad raster in all directions.

Parameters

- **pad_width** – pad width in pixels, int.
- **mode** – str, how to pad, see rasterio.pad. Optional (default: 'constant')
- **constant_values** – nodata value, padding should be filled with, optional (default: 0)

Returns closed, buffered dataset in memory

to_dask_array (*chunk_size=(1, 6000, 6000)*)
transforms numpy to dask array

Parameters **chunk_size** – tuple, size of chunk, optional (default: (1, 6000, 6000))

Returns dask array

warp (*dst_crs*, *resampling_method=0*, *num_threads=4*, *resolution=None*, *nodata=None*, *target_align=None*)
Reproject a source raster to a destination raster.

Parameters

- **dst_crs** – CRS or dict, Target coordinate reference system.
- **resampling_method** – Resampling algorithm, int, defaults to 0 (Nearest) numbers: https://github.com/mapbox/rasterio/blob/master/rasterio/enums.py#L28
- **num_threads** – int, number of workers, optional (default: 4)
- **resolution** – tuple (x resolution, y resolution) or float, optional. Target resolution, in units of target coordinate reference system.
- **target_align** – raster to which to align resolution, extent and gridspacing, optional (Image).
- **nodata** – nodata value of source, int or float, optional.

write_to_file (*path_to_file*, *dtype*, *driver='GTiff'*, *nodata=None*, *compress=None*, *kwargs=None*)
Write a dataset to file. :param path_to_file: str, path to new file :param dtype: datatype, like np.uint16, 'float32' or 'min' to use the minimum type to represent values

Parameters

- **driver** – str, optional (default: 'GTiff')
- **nodata** – nodata value, e.g. 255 (default: None, means nodata value of dataset will be used)

- **compress** – compression, e.g. 'lzw' (default: None)
- **kwargs** – driver specific keyword arguments, e.g. {'nbits': 1, 'tiled': True} for GTiff (default: None) for more keyword arguments see gdal driver specifications, e.g. https://gdal.org/drivers/raster/gtiff.html

ukis_pysat.members

Possible members of enums Datahub and Platform.

class ukis_pysat.members.**Bands**

Bases: pydantic.main.BaseModel

Create a new model by parsing and validating input data from keyword arguments.

Raises ValidationError if the input data cannot be parsed to form a valid model.

class ukis_pysat.members.**Datahub**

Bases: enum.Enum

An enumeration.

EarthExplorer = 'EarthExplorer'

STAC_local = 'STAC Catalog/Collection'

Scihub = 'Scihub'

class ukis_pysat.members.**Platform**

Bases: enum.Enum

An enumeration.

Landsat5 = 'LANDSAT_TM_C1'

Landsat7 = 'LANDSAT_ETM_C1'

Landsat8 = 'LANDSAT_8_C1'

Sentinel1 = 'Sentinel-1'

Sentinel2 = 'Sentinel-2'

Sentinel3 = 'Sentinel-3'

5.1.3 Changelog

[] ()

Deleted

- removed own stac api client (pystac-client should be used from now on)
- deleted the StacApi reference in data.py

[1.3.4] (2022-01-11)

Added

- added deprecation warnings in stacapi.py and stacapi_io.py

Changed

- changed NotImplementedErrors in data.py

[1.3.3] (2021-10-11)

Added

- added DOI

[1.3.2] (2021-08-24)

Added

- `data`: allow *get* on item-search with geometry if *post* is not allowed

[1.3.1] (2021-08-02)

Fixed

- `data`: bugfix query local STAC

[1.3.0] (2021-07-27)

Changed

`-requirements`: Pystac version updated to 1.0.0, STAC version 1.0.0 #147 - `data`: Removed pylandsat dependency and added methods for downloading landsat products from GCS in data module #106

Fixed

- `data`: Consistency for Landsat & Sentinel #151

Added

- `data`: Streaming for Scihub #138

[1.2.1] (2021-05-05)

Fixed

- `data`: Copernicus Open Access Hub Url changed #142
- `data`: fixed download error from EarthExplorer #140

[1.2.0]] (2021-04-07)

Changed

- `data`: added option for additional parameter for query_metadata #27
- `data`: generator instead of catalog return for query_metadata and query_metadata_srcid #127
- `data`: platform removed from download_image and download_quicklook #130

[1.1.0] (2021-03-29)

Changed

- `data`: prep_aoi() is not private and supports __geo_interface__ #113
- `data`: change default scihup endpoint */dhus* to */apihub* #110
- `data`: mocked all interaction with Hubs for unit testing #116
- `data`: EarthExplorer changed to new M2M API due to dependency update #120
- `data`: removed hard dependency for *landsatxplore* & *pylandsat* #104
- `data`: *acquisitiondate* and *ingestiondate* have been removed #124
- split unit tests #110

Fixed

- `data`: *datetime* is now the searchable time of the assets, *start_datetime* and *end_datetime* are introduced #124

Added

- Github Action for black #111

[1.0.2] (2020-01-22)

Changed

- `raster`: documentation clarification for *Image* initialization with *nodata* #105

Fixed

- `data`: bugfix stacapi, bbox did not work because it has to be post like intersection

[1.0.1] (2020-01-20)

Changed

- `data`: make fiona and pyproj optional #104

- `data`: refactored stacapi_io for nicer imports

[1.0.0] (2020-01-20)

Added

- `data`: new stac api #101

Changed

- `data`: metadata as STAC items and collections #98
- `members`: Datahab.file refactored to Datahub.STAC_local, breaking! #99

[0.7.0] (2020-10-30)

Added

- `file`: read and return booleans from env #90
- `raster`: possibility to init with 2D array #88
- `data`: query_metadata_srcid() method and related test #96

Changed

- `get_polarization_from_s1_filename()`: return type is now only `str` instead of `Union[str, List[str]] #92`
- `data`: download_image check if file exists #95

[0.6.3] (2020-09-02)

Fixed

- `file`: imported Pattern of module typing

[0.6.2] (2020-09-02)

Added

- `file`: usage of type hints #76

Fixed

- `raster.get_valid_data_bbox()`: was using the wrong transform #84

[0.6.1] (2020-08-31)

Added

- `file`: added to_ESA_date() function #80
- `file.get_ts_from_sentinel_filename()`: possibility to choose date format
- `raster`: added nodata value upon dataset creation with numpy arrays #82

[0.6.0] (2020-08-28)

Added

- `raster.Image()`: With Statement Context Manager for Image #45
- `raster.Image()`: Alter image array #67
- `raster.Image()`: Target align option for *warp()* #60
- `raster.Image()`: Pass driver specific kwargs to *write_to_file()* #74

Fixed

- `data.Source()`: Fixed query metadata return for new EarthExplorer API #71
- `raster.Image()`: Consider all image bands in *pad()* #59
- `raster.Image()`: Memory leak caused by *__update_dataset()* #62

Changed

- `data.Metadata()`: Corrected field types #58
- `data.MetadataCollection()`: Improved plotting of MetadataCollection to_pandas method #56
- `data.MetadataCollection()`: Made filter method more flexible with list and fuzzy filter options #55
- `raster.Image()`: Split *_pad_to_bbox()* into *pad()* and *_get_pad_width()*, updated *mask()* #59
- replaced os.path with Pathlib #78

Removed

- `file`: removed *pack()* and *unpack()* #57

[0.5.0] (2020-07-03)

Added

- `raster.Image()`: optional nodata value for writing #32

Fixed

- `file.get_ts_from_sentinel_filename()`: Return datetime.datetime objects instead of timestamp strings #42
- `raster.Image()`: in-memory dataset could not be updated if not GTiff and other improvements #48 #52

Changed

- `raster.Image()`: renamed *mask_image()* to *mask()*
- `raster.Image()`: update of init signature to be less confusing #41 #50
- `raster.Image()`: in-memory dataset now always with "GTiff" driver #53

[0.4.0] (2020-06-05)

Added

- `raster.Image()`: expanded test_arr to test AttributeError #31
- `raster.Image()`: optional dimorder for arr and according test #31
- `dn2toa()` tests and testfiles #17
- `data.source()`: accept WKT string as AOI #26
- `data.source()`: check if an AOI string is a file or a WKT string #26

Fixed

- `raster.Image()`: bug in `dn2toa()` related to wrong array shape #17

Changed

- `raster.Image()`: changed `dn2toa(platform, metadata, wavelengths)` to `dn2toa(platform, mtl_file, wavelengths)` #17
- `raster.Image()`: `dn2toa` now raises an error (instead of logging a warning) if Platform is not supported.
- `raster.Image()`: explicit dtype when writing, optional compression #32
- `raster.Image()`: auto-update of in-memory *dataset* #35
- removed logger

[0.3.0] (2020-05-26)

Added

- `download.Source()`: support for local metadata queries #6

Changed

- split PyPI package into subsets to not require all dependencies for every installation #16
- `download.Source()`: removed `traceback` #6
- `download.Source()`: changed `Source(source, source_dir=None)` to `Source(datahub, datadir=None, datadir_substr=None)` #6
- `members.Datahub()`: changed `file` to `File` #6
- updated README #6 #16

[0.2.0] (2020-05-13)

Added

- `download.Source()`: Classes `Metadata` and `MetadataCollection` for metadata handling #13
- expanded metadata part in README #13 - requirements: pyfields
- `download.Source()`: `prep_aoi()` for on the fly preparation of aoi for queries #1
- `data.Image()`: method `get_subset()` to retrieve subset array and bounding box of image tile #12
- `download.Source()`: `query()` accepts now aoi in forms of geojson file with varying CRS or bounding box coordinates in Lat Lon #1
- requirements: pyproj #1
- `download.Source()`: added methods to filter and download metadata #4
- Sentinel3 test #10

Fixed

- `download.Source()`: Improved geocoding quicklooks #5
- fixed #7

Changed

- renamed `ukis_pysat.data` to `ukis_pysat.raster` and `ukis_pysat.download` to `ukis_pysat.data`, breaking compatibility with version 0.1.0 #18
- `download.Source()`: Moved `download_metadata()` and `filter_metadata()` to `Metadata` class #13
- `download.Source()`: Moved all metadata mapping from `query()` to `construct_metadata()` #1
- `download.Source()`: Changed `_construct_metadata()` to `construct_metadata()` and removed static #1
- `download.Source()`: Simplified api queries in `query()` #1
- `download.Source()`: removed `get_metadata()` #4
- requirements: Removed matplotlib, pandas and dask optional #9

[0.1.0] (2020-04-29)

- first release

5.1.4 About

Contributors

The UKIS team creates and adapts libraries which simplify the usage of satellite data. Our team includes (in alphabetical order):

- Boehnke, Christian
- Fichtner, Florian
- Mandery, Nico
- Martinis, Sandro
- Riedlinger, Torsten
- Wieland, Marc

German Aerospace Center (DLR)

Licenses

This software is licensed under the Apache 2.0 License.

Copyright (c) 2020 German Aerospace Center (DLR) * German Remote Sensing Data Center * Department: Geo-Risks and Civil Security

Contributing

The UKIS team welcomes contributions from the community. For more detailed information, see our guide on CONTRIBUTING in the repository if you're interested in getting involved.

What is UKIS?

The DLR project Environmental and Crisis Information System (the German abbreviation is UKIS, standing for Umwelt- und Kriseninformationssysteme) aims at harmonizing the development of information systems at the German Remote Sensing Data Center (DFD) and setting up a framework of modularized and generalized software components.

UKIS is intended to ease and standardize the process of setting up specific information systems and thus bridging the gap from EO product generation and information fusion to the delivery of products and information to end users.

Furthermore the intention is to save and broaden know-how that was and is invested and earned in the development of information systems and components in several ongoing and future DFD projects.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

u

ukis_pysat.data, 12
ukis_pysat.file, 14
ukis_pysat.members, 19
ukis_pysat.raster, 16

Index

A

add_items_from_directory() (*ukis_pysat.data.Source method*), 13
arr (*ukis_pysat.raster.Image attribute*), 17

B

Bands (*class in ukis_pysat.members*), 19

C

close() (*ukis_pysat.data.Source method*), 13
close() (*ukis_pysat.raster.Image method*), 17
construct_metadata() (*ukis_pysat.data.Source method*), 13

D

da_arr (*ukis_pysat.raster.Image attribute*), 17
Datahub (*class in ukis_pysat.members*), 19
dn2toa() (*ukis_pysat.raster.Image method*), 17
download_image() (*ukis_pysat.data.Source method*), 13
download_quicklook() (*ukis_pysat.data.Source method*), 13

E

EarthExplorer (*ukis_pysat.members.Datahub attribute*), 19
env_get() (*in module ukis_pysat.file*), 14

G

get_footprint_from_manifest() (*in module ukis_pysat.file*), 14
get_ipf_from_manifest() (*in module ukis_pysat.file*), 14
get_origin_from_manifest() (*in module ukis_pysat.file*), 14
get_pixel_spacing() (*in module ukis_pysat.file*), 15
get_polarization_from_s1_filename() (*in module ukis_pysat.file*), 15
get_proj_string() (*in module ukis_pysat.file*), 15
get_sat_ts_from_datetime() (*in module ukis_pysat.file*), 15
get_sentinel_scene_from_dir() (*in module ukis_pysat.file*), 16
get_subset() (*ukis_pysat.raster.Image method*), 17
get_tiles() (*ukis_pysat.raster.Image method*), 17
get_ts_from_sentinel_filename() (*in module ukis_pysat.file*), 16
get_valid_data_bbox() (*ukis_pysat.raster.Image method*), 17

I

Image (*class in ukis_pysat.raster*), 16

L

Landsat5 (*ukis_pysat.members.Platform attribute*), 19
Landsat7 (*ukis_pysat.members.Platform attribute*), 19
Landsat8 (*ukis_pysat.members.Platform attribute*), 19

M

mask() (*ukis_pysat.raster.Image method*), 17

P

pad() (*ukis_pysat.raster.Image method*), 18
Platform (*class in ukis_pysat.members*), 19

Q

query_metadata() (*ukis_pysat.data.Source method*), 13
query_metadata_srcid() (*ukis_pysat.data.Source method*), 13

S

Scihub (*ukis_pysat.members.Datahub attribute*), 19
Sentinel1 (*ukis_pysat.members.Platform attribute*), 19
Sentinel2 (*ukis_pysat.members.Platform attribute*), 19

`Sentinel3` (*ukis_pysat.members.Platform attribute*), 19
`Source` (*class in ukis_pysat.data*), 12
`STAC_local` (*ukis_pysat.members.Datahub attribute*), 19
`stream_image()` (*ukis_pysat.data.Source method*), 14

T

`to_dask_array()` (*ukis_pysat.raster.Image method*), 18

U

`ukis_pysat.data` (*module*), 12
`ukis_pysat.file` (*module*), 14
`ukis_pysat.members` (*module*), 19
`ukis_pysat.raster` (*module*), 16

W

`warp()` (*ukis_pysat.raster.Image method*), 18
`write_to_file()` (*ukis_pysat.raster.Image method*), 18