lsru Documentation

Release 0.6.2

loicdtx

Aug 04, 2020

User guide

1 User Guide 3

2 API reference 5

3 Basic Usage 15

4 Order data using a polygon 17

5 Indices and tables 21

Index 23

Query, order and download Landsat surface reflectance data programmatically

`lsru` allows interaction with Usgs and Espa APIs programmatically from python. It has 3 main classes:

- `Usgs` is the interface to the USGS json API. It is mostly used to query the Landsat catalog for available scenes intersecting with a given spatio-temporal window.
- `Espa` is the interface to the ESPA API. Espa is a platform that proposes on demand pre-processing of Landsat data to surface reflectance. Orders can be placed directly from python using that class.
- `Order` is the interface to each individual orders placed to the espa platform; it allows retrieving order status and downloading corresponding scenes.

`lsru` also contains various utilities to smoothen workflows for various use cases of the module.

CHAPTER 1

User Guide

1.1 Installation

Activate a virtualenv (optional but preferable) and run:

```
pip install lsru
```

1.2 Setup

The package requires a configuration file in which usgs credentials are written. By default the file is called `~/.lsru` (this can be modified if you want to join this configuration with the configuration of another project) and has the following structure.

```
[usgs]
username=your_usgs_username
password=your_very_secure_password
```

CHAPTER 2

API reference

2.1 lsru

Interfaces to Usgs and Espa APIs

Usgs([version, conf])	Interface to the Usgs API
Usgs.login()	Login to the Usgs api
Usgs.search(collection, bbox[, begin, end, ...])	Perform a spatio temporal query on Landsat catalog
Usgs.get_collection_name(num)	Get Earth Explorer Landsat collection names
Espa([conf])	Interface to the Espa API
Espa.order(scene_list, products[, format, ...])	Place a pre-procesing order to espa
Espa.get_available_products(scene_list)	Get the list of available products for each elements of a list of scene ids
Order(orderid[, conf])	Class to deal with espa orders
Order.download_all_complete(path[, unpack, ...])	Download all completed scenes of the order to a folder
Order.cancel()	Cancel the order

2.1.1 lsru.Usgs

class lsru.**Usgs**(*version='stable'*, *conf='/home/docs/.lsru'*)

Interface to the Usgs API

See documentation of the `search` method for basic usage

Parameters

- **version** (*str*) – API version to use, defaults to `'stable'`
- **conf** (*str*) – Path of the configuration file containing usgs login credentials

USER

Usgs username

Type str

PASSWORD
Usgs password

Type str

endpoint
API endpoint

Type str

key
API key. Required to perform a search and obtained by running the `login()` method

Type str

key_dt
Time at which the key was generated

Type datetime.datetime

__init__(*version='stable'*, *conf='/home/docs/.lsru'*)
Initialize self. See help(type(self)) for accurate signature.

Methods

__init__([version, conf])	Initialize self.
get_collection_name(num)	Get Earth Explorer Landsat collection names
login()	Login to the Usgs api
search(collection, bbox[, begin, end, ...])	Perform a spatio temporal query on Landsat catalog

Attributes

key_age	Determines the age of API key

2.1.2 lsru.Usgs.login

Usgs.**login**()
Login to the Usgs api

This method is necessary to obtain an API key (automatically saved in the `key` attribute), and send other queries to the API

Returns True if query was successful, False otherwise

Return type bool

2.1.3 lsru.Usgs.search

Usgs.**search**(*collection*, *bbox*, *begin=None*, *end=None*, *max_cloud_cover=100*, *months=None*, *starting_number=1*, *max_results=50000*)
Perform a spatio temporal query on Landsat catalog

Parameters

- **collection** (*str*) – Landsat collection to query. Use LANDSAT_8_C1, LANDSAT_ETM_C1 and LANDSAT_TM_C1 for OLI, ETM+, and TM respectively
- **bbox** (*tuple*) – A bounding box in the form of a tuple (left, bottom, right, top)
- **begin** (*datetime.datetime*) – Optional begin date
- **end** (*datetime.datetime*) – Optional end date
- **max_cloud_cover** (*int*) – Cloud cover threshold to use for the query
- **months** (*list*) – List of month indices (1,12) for only limiting the query to these months
- **max_results** (*int*) – Maximum number of scenes to return
- **starting_number** (*int*) – Used to determine the result number to start returning from. Is meant to be used when the total number of hits is higher than `max_results`, to return results in a paginated fashion

Example

```
>>> from lsru import Usgs
>>> import datetime
>>> usgs = Usgs()
>>> usgs.login()
>>> scene_list = usgs.search(collection='LANDSAT_8_C1',
>>>                          bbox=(3.5, 43.4, 4, 44),
>>>                          begin=datetime.datetime(2012,1,1),
>>>                          end=datetime.datetime(2016,1,1))
>>> print(scene_list)
```

Returns List of scenes with complete metadata

Return type list

2.1.4 lsru.Usgs.get_collection_name

static Usgs.**get_collection_name**(*num*)

Get Earth Explorer Landsat collection names

Parameters **num** (*int*) – Landsat spacecraft number (4, 5, 7 or 8)

Returns

Earth Explorer collection name formated to i.e. pass to the search method

Return type str

2.1.5 lsru.Espa

class lsru.**Espa**(*conf='/home/docs/.lsru'*)

Interface to the Espa API

Espa is a platform providing on demand pre-processing of Landsat surface data. This class uses the API of the espa platform to query and place orders programatically

USER

Usgs username

Type str

PASSWORD
Usgs password

Type str

host
API host url

Type str

Parameters **conf** (*str*) – Path of the config file containing usgs credentials

__init__ (*conf='/home/docs/.lsru'*)
Initialize self. See help(type(self)) for accurate signature.

Methods

__init__([conf])	Initialize self.
get_available_products(scene_list)	Get the list of available products for each elements of a list of scene ids
order(scene_list, products[, format, note, ...])	Place a pre-procesing order to espa

Attributes

formats	Get a list of file formats supported by the platform
orders	Get a list of current orders
projections	Get a dictionary of projections supported by the platform
resampling_methods	Get a list of resamling methods suported by the platform
user	Get Usgs user details

2.1.6 lsru.Espa.order

Espa.**order**(*scene_list*, *products*, *format='gtiff'*, *note=None*, *resampling='nn'*, *resolution=None*, *projection=None*, *extent=None*, *extent_units='dd'*, *verbose=False*)
Place a pre-procesing order to espa

Parameters

- **scene_list** (*list*) – List of Landsat scene ids
- **products** (*list*) – List of products to order for pre-processing See Espa.get_available_products() to get information on available products
- **format** (*str*) – Pre-processing file format. See Espa.formats for information on available formats
- **note** (*str*) – Optional human readable message to pass to the order
- **resampling** (*str*) – Resamping method to be used when reprojecting or resizing ordered images. See Espa.resampling_methods for valid values.

- **resolution** (*float*) – Ouput resolution (optional). If specified, the pre-processing order will be resized to the specified resolution. If set to None (default), no resizing is performed and products are processed at their original resolution (usually 30m).
- **projection** (*dict*) – Optional dictionary with projection name and projection parameter values. Ordered products are re-projected to the specified projection when set. See `Espa.projections` for list and format of supported projections
- **extent** (*tuple*) – Bounding box to use to crop the pre-processed products bounding box is in the form of a (left, bottom, right, top) tuple. This is optional and requires a projection to be set.
- **extent_units** (*str*) – Units of the provided extent. `'dd'` (decimal degrees) is the default. If `` `meters' `` bounds are specified according to the coordinate reference system space.
- **verbose** (*bool*) – Prints the json body being sent. Useful for debugging purposes

Example

```
>>> from lsru import Espa, Usgs
>>> import datetime
>>> espa = Espa()
>>> usgs = Usgs()
>>> usgs.login()
>>> scene_list = usgs.search(collection='LANDSAT_8_C1',
...                          bbox=(3.5, 43.4, 4, 44),
...                          begin=datetime.datetime(2014,1,1),
...                          end=datetime.datetime(2018,1,1))
>>> scene_list = [x['displayId'] for x in scene_list]
>>> order = espa.order(scene_list, products=['sr', 'pixel_qa'])
```

Returns The method is mostly used for its side effect of placing a pre-processing order on the espa platform. It also returns a the `lsru.Order` instance corresponding to the order

Return type *lsru.Order*

2.1.7 lsru.Espa.get_available_products

`Espa.`**get_available_products**(*scene_list*)

Get the list of available products for each elements of a list of scene ids

Parameters **scene_list** (*list*) – List of scene ids

Example

```
>>> from lsru import Espa
>>> espa = Espa()
>>> print(espa.get_available_products([
...     'LE07_L1TP_029030_20170221_20170319_01_T1'
... ]))
```

Returns

Information on products available for each element of the input list provided

Return type dict

2.1.8 lsru.Order

class lsru.**Order**(*orderid*, *conf='/home/docs/.lsru'*)

Class to deal with espa orders

orderir

Espa order ID

Type str

Parameters

- **orderid** (*str*) – Espa order ID
- **conf** (*str*) – Path to file containing usgs credentials

__init__(*orderid*, *conf='/home/docs/.lsru'*)

Initialize self. See help(type(self)) for accurate signature.

Methods

__init__(orderid[, conf])	Initialize self.
cancel()	Cancel the order
download_all_complete(path[, unpack, ...])	Download all completed scenes of the order to a folder

Attributes

is_complete	Check if order has status complete
items_status	
status	Get the current status of the order
urls_completed	Get list of item's url whose status is complete

2.1.9 lsru.Order.download_all_complete

Order.**download_all_complete**(*path*, *unpack=False*, *overwrite=False*, *check_complete=True*)

Download all completed scenes of the order to a folder

Parameters

- **path** (*str*) – Directory where data are to be downloaded
- **unpack** (*bool*) – Unpack downloaded archives on the fly
- **overwrite** (*bool*) – Force overwriting existing files even when they already exist? Defaults to False
- **check_complete** (*bool*) – When local files exist and overwrite is set to False, check whether local and remote files sizes match? File is re-downloaded when sizes are different. Only makes sense if overwrite is set to False. Defaults to True. Also note that checking file size takes time (a few millisecons probably), so that you'll save time setting this argument to False in case you're sure previous downloads are complete Note that this option

does not work when `unpack` is set to True

Returns Used for its side effect of batch downloading data, no return

2.1.10 lsru.Order.cancel

`Order.`**`cancel`**`()`

Cancel the order

Orders are processed in the order they were placed. Cancelling an order may be useful when the order is blocking other orders

Returns The response of the API to the cancellation order

Return type dict

2.2 utils

utils.bounds(geom)	Return a bounding box from a geometry
utils.geom_from_metadata(meta)	Return a geometry from a Landsat scene metadata as returned by USGS api
utils.is_valid(id)	Landsat scene id validity checker
utils.url_retrieve(url, filename[, ...])	Generic file download function
utils.url_retrieve_and_unpack(url, path[, ...])	Generic function to combine download and unpacking of tar archives

2.2.1 lsru.utils.bounds

`lsru.utils.`**`bounds`**`(`*`geom`*`)`

Return a bounding box from a geometry

Adapted from https://gis.stackexchange.com/a/90554/17409

Parameters **`geom`** (*`dict`*) – Geojson like geometry

Example

```
>>> from lsru.utils import bounds
>>> geom = {'coordinates': [[[3.34481, 43.96708],
...                          [6.17992, 45.23413],
...                          [5.55366, 43.56654],
...                          [3.34481, 43.96708]]],
...         'type': 'Polygon'}
>>> print(bounds(geom))
```

Returns Bounding box (left, bottom, right, top)

Return type Tuple

2.2.2 lsru.utils.geom_from_metadata

lsru.utils.**geom_from_metadata**(*meta*)

Return a geometry from a Landsat scene metadata as returned by USGS api

Parameters **meta** (*dict*) – Landsat scene metadata as returned by Usgs Api

Example

```
>>> from lsru import Usgs
>>> from lsru.utils import geom_from_metadata
>>> import datetime
>>> from shapely.geometry import shape
>>> from pprint import pprint
```

```
>>> usgs = Usgs()
>>> usgs.login()
>>> scene_list = usgs.search(collection='LANDSAT_8_C1',
...                          bbox=(3.5, 43.4, 4, 44),
...                          begin=datetime.datetime(2012,1,1),
...                          end=datetime.datetime(2016,1,1))
>>> geom = geom_from_metadata(scene_list[0])
>>> s = shape(geom)
>>> pprint(geom)
>>> print(s.is_valid)
```

Returns GeoJson like geometry. CRS is always in longlat (EPSG 4326)

Return type dict

2.2.3 lsru.utils.is_valid

lsru.utils.**is_valid**(*id*)

Landsat scene id validity checker

Parameters **id** (*str*) – Landsat scene id

Example

```
>>> from lsru import Usgs
>>> from lsru.utils import is_valid
>>> import datetime
>>> import requests
>>> from pprint import pprint
>>> usgs = Usgs()
>>> usgs.login()
>>> scene_list = usgs.search(collection='LANDSAT_8_C1',
...                          bbox=(3.5, 43.4, 4, 44),
...                          begin=datetime.datetime(2012,1,1),
...                          end=datetime.datetime(2016,1,1))
>>> pprint(scene_list[2])
>>> print(is_valid(scene_list[2]['displayId']))
```

Returns Whether the provided scene id is valid or not

Return type bool

2.2.4 lsru.utils.url_retrieve

lsru.utils.**url_retrieve**(*url*, *filename*, *overwrite=False*, *check_complete=True*)

Generic file download function

Similar to url_retrieve from standard library with additional checks for already existing files and incomplete downloads

Parameters

- **url** (*str*) – Url pointing to file to retrieve
- **filename** (*str*) – Local file name under which the downloaded content is written
- **overwrite** (*bool*) – Force overwriting local file even when it already exists? Defaults to False
- **check_complete** (*bool*) – When local file exists and overwrite is set to False, check whether local and remote file sizes match? File is re-downloaded when sizes are different. Only makes sense if overwrite is set to False. Defaults to True

Returns The filename

Return type str

2.2.5 lsru.utils.url_retrieve_and_unpack

lsru.utils.**url_retrieve_and_unpack**(*url*, *path*, *overwrite=False*)

Generic function to combine download and unpacking of tar archives

Downloads the tar archive as a memory object and extracts its content to a new directory. Directory name is the remote file name with stripped extension

Parameters

- **url** (*str*) – Url pointing to tar file to retrieve
- **path** (*str*) – Path to directory under which a new directory containing the archive content will be created
- **overwrite** (*bool*) – Force overwriting local files even when the output directory already exist? Defaults to False

Returns The path containing extracted content

Return type str

CHAPTER 3

Basic Usage

This example shows how to query the data from a bounding box, order surface reflectance processing for full scenes without reprojection and download.

3.1 Define variables and query available scenes to Usgs

```
from lsru import Usgs
import datetime

# Define query extent
bbox = (3.5, 43.4, 4, 44)

# Instantiate Usgs class and login
usgs = Usgs()
usgs.login()

# Query the Usgs api to find scene intersecting with the spatio-temporal window
scene_list = usgs.search(collection='LANDSAT_8_C1',
                         bbox=bbox,
                         begin=datetime.datetime(2013,1,1),
                         end=datetime.datetime(2016,1,1),
                         max_results=10,
                         max_cloud_cover=40)

# Extract Landsat scene ids for each hit from the metadata
scene_list = [x['displayId'] for x in scene_list]
```

3.2 Place a processing order to Espa

The scene list can be used to send a processing order to Espa via the Espa API.

```
from lsru import Espa
from pprint import print

# Instantiate Espa class
espa = Espa()

# Place order (full scenes, no reprojection, sr and pixel_qa)
order = espa.order(scene_list=scene_list, products=['sr', 'pixel_qa'])
print(order.orderid)

# espa-loic.dutrieux@wur.nl-10212018-102816-245'
```

3.3 Check current orders status

```
for order in espa.orders:
    # Orders have their own class with attributes and methods
    print('%s: %s' % (order.orderid, order.status))

# espa-loic.dutrieux@wur.nl-10222018-062836-330: ordered
# espa-loic.dutrieux@wur.nl-10212018-174321-508: complete
# espa-loic.dutrieux@wur.nl-10212018-174430-792: complete
# espa-loic.dutrieux@wur.nl-10212018-102816-245: complete
# espa-loic.dutrieux@wur.nl-10182018-100137-786: complete
```

3.4 Download completed orders

When Espa finishes pre-processing an order, its status changes to `complete`, we can then download the processed scenes.

```
for order in espa.orders:
    if order.is_complete:
        order.download_all_complete('/media/landsat/download/dir')
```

CHAPTER 4

Order data using a polygon

The following example details the steps to place a pre-processing order of scenes intersecting with a polygon. Such task requires a little bit of boilerplate code since the Usgs API requires an extent and not a polygon, but thanks to `shapely` and some additional utilities provided by `lsru` it can be done with a few lines of code. The steps are roughly to:

- Read the feature (e.g. using `fiona` or directly from a geojson file with `json`)
- Compute the bounding box of the feature
- Query scenes intersecting with the bounding box to the Usgs API
- Filter out scenes that do not intersect with the geometry
- Place the pre-processing order to Espa

To run this script, we'll need shapely, and some utils to manipulate geojson geometries provided by `lsru`. You may need fiona as well in case you need to read a feature from a geospatial vector file (shapefile, geopackage, etc)

```
from datetime import datetime

from shapely.geometry import shape

from lsru import Usgs, Espa, Order
from lsru.utils import bounds, geom_from_metadata
```

For the present example, we'll use the following feature, which roughly corresponds to the contours of the state of Baja California Sur in Mexico.

```
feature = {'geometry': {'coordinates': [[[-114.192, 27.975],
                                         [-114.456, 27.8],
                                         [-115.093, 27.878],
                                         [-114.543, 27.313],
                                         [-113.928, 26.922],
                                         [-113.423, 26.844],
                                         [-112.72, 26.412],
                                         [-112.148, 25.741],
```

(continues on next page)

(continued from previous page)

```
                                          [-112.324, 24.827],
                                          [-111.687, 24.467],
                                          [-110.984, 24.147],
                                          [-110.72, 23.725],
                                          [-110.281, 23.504],
                                          [-110.061, 22.877],
                                          [-109.446, 23.06],
                                          [-109.424, 23.423],
                                          [-110.259, 24.347],
                                          [-110.347, 24.187],
                                          [-110.61, 24.327],
                                          [-110.588, 24.707],
                                          [-111.445, 26.215],
                                          [-112.654, 27.684],
                                          [-112.961, 28.053],
                                          [-114.192, 27.975]]],
           'type': 'Polygon'},
           'properties': {},
           'type': 'Feature'}

# Generate the bounding box of the geometry used for the spatial query
bbox = bounds(feature['geometry'])

# Instantiate Usgs class and login
usgs = Usgs()
usgs.login()
collection = usgs.get_collection_name(8)

# Query for records of Landsat 8 intersecting with the spatio-temporal window
meta_list = usgs.search(collection=collection, bbox=bbox,
                        begin=datetime(2018, 1, 1), end=datetime(2018, 2, 28),
                        max_cloud_cover=30)
print(len(meta_list))

# Printed:
# 52
```

Because we queried the data using the extent, it is highly probable that some scenes do not intersect with the initial geometry but only with its extent. We can therefore filter the list of scene metadata by testing for intersection between the scene bounds and the geometry. This is done using shapely

```
region_geom = shape(feature['geometry'])
meta_list = [x for x in meta_list if
             shape(geom_from_metadata(x)).intersects(region_geom)]

print(len(meta_list))

# printed:
# 27
```

The amount of element has reduced by half compared to the total API hits and we are now sure to have retained only scenes that actually intersect with the initial geometry.

We can now proceed to preparing the scene list for placing the order to espa

```
scene_list = [x['displayId'] for x in meta_list]
espa = Espa()
```

(continues on next page)

(continued from previous page)

```
order = espa.order(scene_list,
                   products=['pixel_qa', 'sr_ndmi'])
```

We can then track the status of the order and eventually download it once processing is completed

```
print(order.status)

# printed:
# ordered
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Index

Symbols

__init__() (*lsru.Espa method*), 8
__init__() (*lsru.Order method*), 10
__init__() (*lsru.Usgs method*), 6

B

bounds() (*in module lsru.utils*), 11

C

cancel() (*lsru.Order method*), 11

D

download_all_complete() (*lsru.Order method*), 10

E

endpoint (*lsru.Usgs attribute*), 6
Espa (*class in lsru*), 7

G

geom_from_metadata() (*in module lsru.utils*), 12
get_available_products() (*lsru.Espa method*), 9
get_collection_name() (*lsru.Usgs static method*), 7

H

host (*lsru.Espa attribute*), 8

I

is_valid() (*in module lsru.utils*), 12

K

key (*lsru.Usgs attribute*), 6
key_dt (*lsru.Usgs attribute*), 6

L

login() (*lsru.Usgs method*), 6

O

Order (*class in lsru*), 10
order() (*lsru.Espa method*), 8
orderir (*lsru.Order attribute*), 10

P

PASSWORD (*lsru.Espa attribute*), 8
PASSWORD (*lsru.Usgs attribute*), 6

S

search() (*lsru.Usgs method*), 6

U

url_retrieve() (*in module lsru.utils*), 13
url_retrieve_and_unpack() (*in module lsru.utils*), 13
USER (*lsru.Espa attribute*), 7
USER (*lsru.Usgs attribute*), 5
Usgs (*class in lsru*), 5