eodms-api-client

Release 1.2.0

Mike Brady

Apr 23, 2022

CONTENTS:

1 How to Install **3**
- 1.1 Configuring .netrc . . . 3

2 Example Usage **5**
- 2.1 Simple Example Paramset . . . 5
- 2.2 Command-Line Interface (CLI) . . . 6
- 2.3 Interactive Python . . . 6

3 Source-Code Docstrings **9**
- 3.1 Main Client . . . 9
- 3.2 EODMS Collection Parameter Validation . . . 10
- 3.3 Geographic Functions . . . 10
- 3.4 Authentication Functions . . . 11

4 Indices and tables **13**

Python Module Index **15**

Index **17**

This package was created as a tool for myself for interacting with RCM data archived on the Earth Observation Data Management System (EODMS) operated by Natural Resources Canada.

It provides a command-line interface for querying, ordering, and downloading data from EODMS as well as a Python client class `EodmsAPI` for doing the same operations from within a Python session.

```
$ eodms -c RCM -g aoi_polygon.geojson --dump-results
```

```
>>> from eodms_api_client import EodmsAPI
>>> client = EodmsAPI(collection='RCM')
>>> client.query(geometry='aoi_polygon.geojson')
```

Vector geometry files for spatial subsetting can be anything supported by Fiona (GeoJSON, Esri Shapefile, OGC GeoPackage, KML/KMZ, etc.)

CHAPTER

ONE

HOW TO INSTALL

There are 3 main ways to install `eodms-api-client`

1. If you're using conda, download from conda-forge:

```
conda install -n <name-of-conda-env> eodms-api-client -c conda-forge
```

2. No access to conda or not a fan? It's also on PyPI:

```
pip install eodms-api-client
```

3. If you like playing with potentially-unstable code, install the source from github:

```
pip install https://github.com/m9brady/eodms-api-client/archive/main.zip
```

1.1 Configuring .netrc

In order to not be bugged to enter your username/password every time, you may wish to create an EODMS entry in `.netrc` file (`_netrc` on Windows) for your userprofile

For Linux/Mac users:

```
echo "machine data.eodms-sgdot.nrcan-rncan.gc.ca login <username> password <password>" >>
↪ ~/.netrc
# set permissions to user-readwrite-only
chmod 600 ~/.netrc
```

For Windows users:

```
cd %USERPROFILE%
type nul >> _netrc
notepad _netrc
# in notepad, paste the following and replace the chevroned text
machine data.eodms-sgdot.nrcan-rncan.gc.ca login <username> password <password>
# save and quit notepad
```

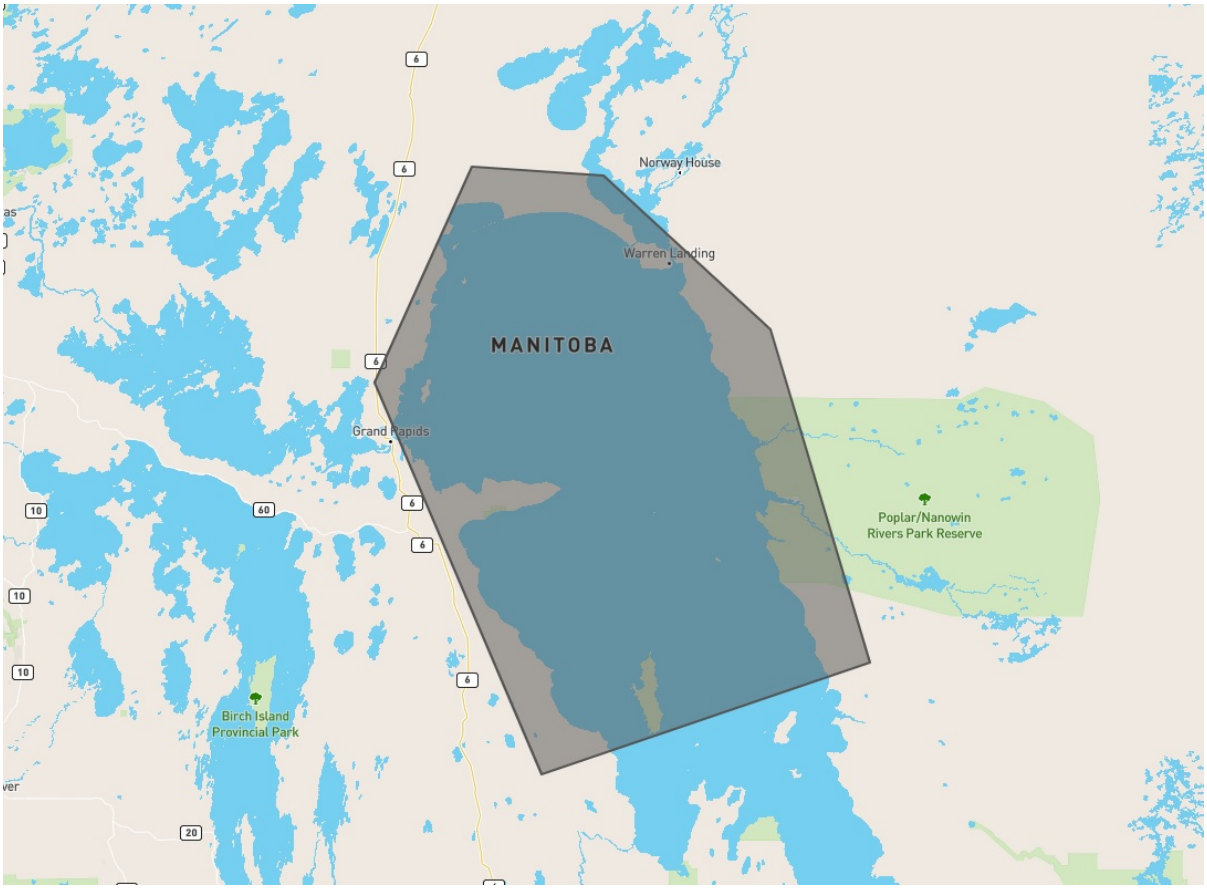
CHAPTER

TWO

EXAMPLE USAGE

2.1 Simple Example Paramset

Our study area is Northern Lake Winnipeg and we have constructed a geojson polygon using geojson.io (made possible by the folks at Mapbox)



The geojson file that the above image displays can be found here

We are also only interested in **RCM** imagery from **August 15 2020** to **August 22 2020** for our analysis.

2.2 Command-Line Interface (CLI)

First we run a query to see what's available

```
$ eodms -c RCM -s 2020-08-15 -e 2020-08-22 -g lakewinnipegnorth.geojson --dump-results
2020-12-23 10:34:09 | eodmsapi.cli | INFO | Querying EODMS API
Fetching result metadata: 100%|| 26/26 [00:04<00:00,  5.30item/s]
2020-12-23 10:34:18 | eodmsapi.cli | INFO | Finished query. 26 results
2020-12-23 10:34:18 | eodmsapi.cli | INFO | Saving query results to file: ./query_
↪results.geojson
```

We can then inspect the `query_results.geojson` in Python or GIS software (like QGIS) to see if it satisfies our needs. If there are no modifications to be made, we can submit the order as-is

```
$ eodms -c RCM -s 2020-08-15 -e 2020-08-22 -g lakewinnipegnorth.geojson --submit-order
```

After order submission, you will receive emails from EODMS on the status of your order. Once your order status has changed to "Complete", take note of the `Order ID` in the Delivery Notification email and supply it to the CLI

```
$ eodms -c RCM --order-id <order_id>
```

2.3 Interactive Python

Just like in the command-line example, we first run a query to see what's available

```
>>> from eodms_api_client import EodmsAPI
>>> client = EodmsAPI(collection='RCM')
>>> client.query(start='2020-08-15', end='2020-08-22', geometry='lakewinnipegnorth.
↪geojson')
Fetching result metadata: 100%|| 26/26 [00:09<00:00,  2.70item/s]
>>> len(client.results)
26
```

Since we are already in Python, we can do some exploration of the results in order to filter out any scenes we won't be needing

```
# let's see what kind of beam modes are available
>>> client.results.groupby('Beam Mode Type')['Granule'].agg('count')
Beam Mode Type
Medium Resolution 30m     2
Medium Resolution 50m    23
Quad-Polarization         1
Name: Granule, dtype: int64

# we like the sound of the 30m and quad-pol products. Let's subset to just those
>>> subset = client.results.loc[client.results['Beam Mode Type'] != 'Medium Resolution␣
↪50m']
>>> len(subset)
3

# let's make sure that the 3 scenes left have decent overlap with our area-of-interest␣
↪(AOI)
```

(continues on next page)

(continued from previous page)

```
>>> import geopandas as gpd

# project to meters (UTM 14N WGS84) for area calculations
>>> aoi = gpd.read_file('lakewinnipegnorth.geojson').to_crs('epsg:32614')
>>> subset = subset.to_crs(aoi.crs)
>>> subset['overlap_area'] = subset.intersection(aoi.unary_union).area
>>> subset['overlap_pct'] = subset['overlap_area'] / subset.area
>>> subset['overlap_pct']
3     0.887386
8     0.015402
18    0.962696
Name: overlap_pct, dtype: float64

# there is 1 scene that has less than 2% of its area overlapping with our AOI
# so let's remove it!
>>> subset = subset.loc[subset['overlap_pct'] > 0.1]

# now we extract the EODMS record Ids for our 2 scenes and submit our order
>>> record_ids = subset['EODMS RecordId'].tolist()
>>> order_ids = client.order(record_ids)
```

Same as with the CLI example, we wait for the "Order Complete" email and provide the `Order Id` to our client.

```
>>> client.download(order_id)
```

CHAPTER

THREE

SOURCE-CODE DOCSTRINGS

3.1 Main Client

class eodms_api_client.eodms.**EodmsAPI**(*collection*, *username=None*, *password=None*)

Entry-point for accessing the EODMS REST API

Inputs:

- collection: The EODMS Collection to which queries and orders will be sent
- username: EODMS account username, leave blank to use .netrc (if available)
- password: EODMS account password, leave blank to use .netrc (if available)

query(***kwargs*)

Submit a query to EODMS and save the results as a geodataframe in a class attribute

Inputs:

- kwargs: Any number of keyword arguments that will be validated based on the EODMS collection being queried

Outputs:

- self.results: A class attribute containing a geodataframe with the returned query results

order(*record_ids*, *priority='Medium'*)

Submit an order to EODMS using record ID numbers retrieved from a search query

Inputs:

- record_ids: list of record ID numbers to order
- priority: one of ['Low', 'Medium', 'High', 'Urgent'] Default: 'Medium'

Outputs:

- order_ids: list of EODMS ordering system ID numbers for later downloading

download(*order_ids*, *output_location='.'*)

Appears that the endpoint has a hard limit of 100 results, so need to be fancy if more than 100 items are given for an orderId

order_ids: list of integer order numbers output_location: where the downloaded products will be saved to (will be created if doesn't exist yet)

class eodms_api_client.eodms.**EODMSHTMLFilter**(*, *convert_charrefs=True*)

Custom HTML parser for EODMS API item status responses

Stolen from stackoverflow user FrBrGeorge: https://stackoverflow.com/a/55825140

3.2 EODMS Collection Parameter Validation

eodms_api_client.params.**validate_query_args**(*args*, *collection*)

Try to validate as many keyword arguments as possible for the given EODMS collection

Inputs:

- args: dictionary of keyword arguments provided by the user
- collection: EODMS collection to validate arguments against

Outputs:

- query: a properly-escaped query string ready to be sent to EODMS API

eodms_api_client.params.**generate_meta_keys**(*collection*)

For the given collection, return the various keys that will be used to pull image metadata from EODMS

3.3 Geographic Functions

eodms_api_client.geo.**transform_metadata_geometry**(*meta_geom*, *target_crs=None*)

Transform a given image footprint geometry from WGS84 to the desired crs

EODMS always returns geometries in WGS84, so we provide the option to transform into something more useful through the use of a CLI flag (–t_srs)

Inputs:

- meta_geom: the footprint geometry as a shapely.geometry object
- target_crs: either a pyproj.CRS or properly-formatted string for the desired projection

Outputs:

- if target_crs is not supplied, simply return the passed geometry
- if target_crs is supplied, return the transformed geometry

eodms_api_client.geo.**metadata_to_gdf**(*metadata*, *collection*, *target_crs=None*)

Instead of a jumbled JSON-looking dictionary, create a geopandas.GeoDataFrame from the EODMS search query response

Inputs:

- metadata: dictionary of EODMS search query and image metadata response
- collection: string of EODMS Collection ID - used for column name standardization
- target_crs: pyproj.CRS instance or properly-formatted string for the desired projection

Outputs:

- df: geopandas.GeoDataFrame containing <metadata> projected to <target_crs>

eodms_api_client.geo.**load_search_aoi**(*geofile*)

Given a file, attempt to parse it as a set of vector features to be sent as part of a request to the EODMS REST API. If the features are not already in WGS84, they are transformed by this function.

Inputs:

- geofile: A file containing vector data (SHP, GEOJSON, GPKG, etc.)

Outputs:

- wkt: The Well-Known Text representation of the features within <geofile>

3.4 Authentication Functions

eodms_api_client.auth.**create_session**(*username=None*, *password=None*)

Create a persistent session object for the EODMS REST API using the given username and password. If neither is provided, attempt to use the .netrc file and fallback to console entry as a last-resort

Inputs:

- username: EODMS username
- password: EODMS password

Outputs:

- session: requests.Session object for the EODMS REST API

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

e

eodms_api_client.auth, 11
eodms_api_client.eodms, 9
eodms_api_client.geo, 10
eodms_api_client.params, 10

INDEX

C

create_session() (*in module eodms_api_client.auth*), 11

D

download() (*eodms_api_client.eodms.EodmsAPI method*), 9

E

eodms_api_client.auth
    module, 11
eodms_api_client.eodms
    module, 9
eodms_api_client.geo
    module, 10
eodms_api_client.params
    module, 10
EodmsAPI (*class in eodms_api_client.eodms*), 9
EODMSHTMLFilter (*class in eodms_api_client.eodms*), 9

G

generate_meta_keys() (*in module eodms_api_client.params*), 10

L

load_search_aoi() (*in module eodms_api_client.geo*), 10

M

metadata_to_gdf() (*in module eodms_api_client.geo*), 10
module
    eodms_api_client.auth, 11
    eodms_api_client.eodms, 9
    eodms_api_client.geo, 10
    eodms_api_client.params, 10

O

order() (*eodms_api_client.eodms.EodmsAPI method*), 9

Q

query() (*eodms_api_client.eodms.EodmsAPI method*), 9

T

transform_metadata_geometry() (*in module eodms_api_client.geo*), 10

V

validate_query_args() (*in module eodms_api_client.params*), 10