New Python Developments to Access CDS Services

Matthieu Baumann,[1] and Thomas Boch[2]
[1]*CNRS Observatoire de Strasbourg, Strasbourg, Alsace, France;*
matthieu.baumann@astro.unistra.fr

[2]*CNRS Observatoire de Strasbourg, Strasbourg, Alsace, France*

Abstract.
We will present recent developments made in the frame of the ASTERICS project and aimed at providing Python interface to CDS services and Virtual Observatory standards. Special care has been taken to integrate these developments into the existing astropy/astroquery environment.
A new astroquery.cds module allows one to retrieve image or catalogue datasets available in a given region of the sky described by a MOC (Multi Order Coverage map) object. Datasets can also be filtered through additional constraints on their metadata.
The MOCPy library has been upgraded: performance has been greatly improved, unit tests and continuous integration have been added, and the integration of the core code into the astropy.regions module is under way. We have also added an experimental support for creation and manipulation of T-MOCs which describe the temporal coverage of a data collection.

1. Python Packages Presentation

1.1. MOCPy (Boch & Baumann 2015): a Library Handling the Creation and Manipulation of MOCs

New features and improvements have been added to the library:

- MOCPy (Boch & Baumann 2015) has been optimized and tends to use numpy's broadcasting feature as much as possible. Creating a MOC from a list of astropy `SkyCoord` is a lot faster thanks to the vectorization involved when operations are directly done on numpy arrays.

  The following code shows the implementation of `from_lonlat` responsible for creating a MOC from lon and lat astropy quantities at a given `order`. This code:

  - Uses `astropy-healpix` to get the HEALPix cells where the (lon, lat) coordinates are located.
  - Build a $N \times 2$ numpy array storing the intervals of the HEALPix cells at a given order.

  No Python loops over the quantities are involved here as it is encouraged to perform operations directly on numpy arrays.

- Dependencies to healpy have been removed. We now use astropy-healpix and therefore have changed the licence of MOCPy (Boch & Baumann 2015) from GPL to BSD-3.

- A new `serialize` method has been added, taking an optional format argument that can be set to fits or json.

- New methods `fill` and `perimeter` have been implemented. These methods are responsible for plotting the MOC (resp. its perimeter) on a matplotlib axe using a projection defined by an `astropy.wcs.WCS` object.

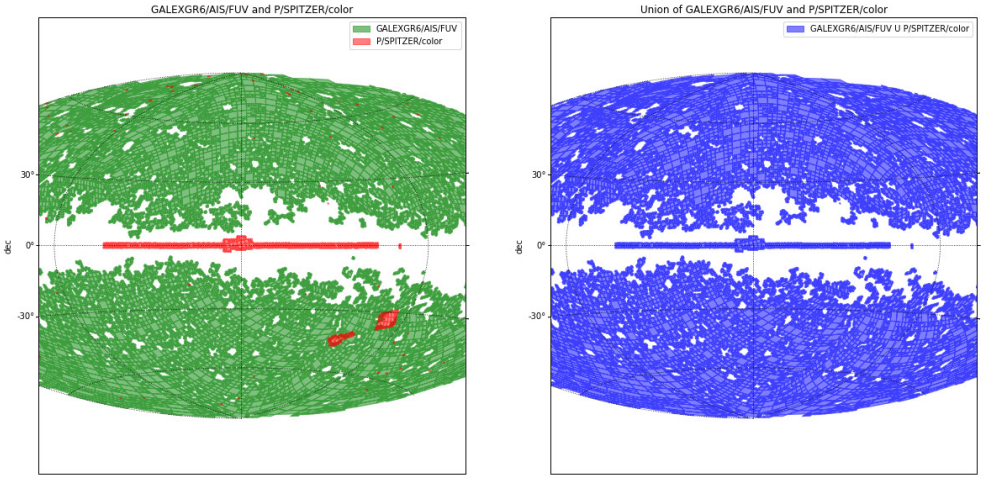

Figure 1. Union of the MOCs between GALEXGR6/AIS/FUV and SPITZER

- A new `TMOC` class handles the creation and manipulation of temporal MOCs. A `from_times` method creates a T-MOC object from an `astropy.time.Time` object. As for the spatial MOCs, it is possible to `serialize` a T-MOC, compute the intersection, union, difference between several T-MOCs as well as use them to filter an `astropy.time.Time` object.

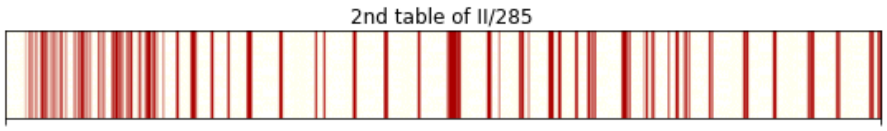

Figure 2. Example of a T-MOC created from II/285

```
First observation: 1978-05-10 20:09:28.672
Last observation: 2004-04-22 16:56:36.350
Total duration: 227.424 jd
Max order: 14
```

1.2. astroquery.cds (Baumann 2018a): a New Module for Retrieving Data Collections Based on Region and/or Meta-data Queries

astroquery.cds (Baumann 2018a) has been merged into the master branch of astroquery in July the 23th and will be available for its next release (v0.3.9). This module requests the CDS MOCServer, a server storing MOCs and meta-data of $\simeq$ 20000 data collections. This package offers two methods (see the module's documentation (Baumann 2018a) for more details):

- `query_region` retrieves the collections having their observations in a specific region. Regions can be expressed as `mocpy.MOC` objects, circle or polygon sky regions.
- `find_datasets` retrieves the collections based on a constraint on their meta-data.

These two methods return by default an `astropy.table.Table` containing the meta-data of one collection per row. An optional argument `return_moc=True` can be used to directly retrieve the MOC (a `mocpy.MOC` object) of the matching collections.

Below 1 is an example of an astropy table returned by `query_region` and filtered to select only the vizier tables having between 75000 and 100000 sources. The meta-data shown here are `obs_id`, `obs_title` and `dataproduct_type`. For a list of all the possible meta-data returned by the `cds` module, please refer to the page 18 of the HiPS IVOA paper (Fernique et al. 2017).

Table 1. Example of an astropy table returned by astroquery.cds

obs_id	**obs_title**	**dataproduct_type**
I/208/ppm3	The 90000 stars Supplement to the PPM Catalogue (Roeser+, 1994) (ppm3)	catalog
I/237/catalog	The Washington Visual Double Star Catalog, 1996.0 (Worley+, 1996) (catalog)	catalog
I/276/catalog	Tycho Double Star Catalogue (TDSC) (Fabricius+ 2002) (catalog)	catalog

2. State of the Art of the CDS Python Tools

The following image 3 results from a notebook (Baumann 2018b) combining different Python packages, most of them being developed by the CDS team through the past years. It is available on the cds-astro github repository as an example for astronomers. This script:

1. Retrieves two MOCs from the MOCServer (Baumann 2018a).
2. Computes their intersection (Boch & Baumann 2015) and shows the resulting MOC on an aladin-lite view (ipyaladin).

3. Searches for a vizier table in optical regime having some observations in this region (Baumann 2018a).
4. Retrieves the table using astroquery.vizier.
5. Filters the table to only keep the observations lying in the MOC (Boch & Baumann 2015) and adds the filtered table to the aladin view (ipyaladin).

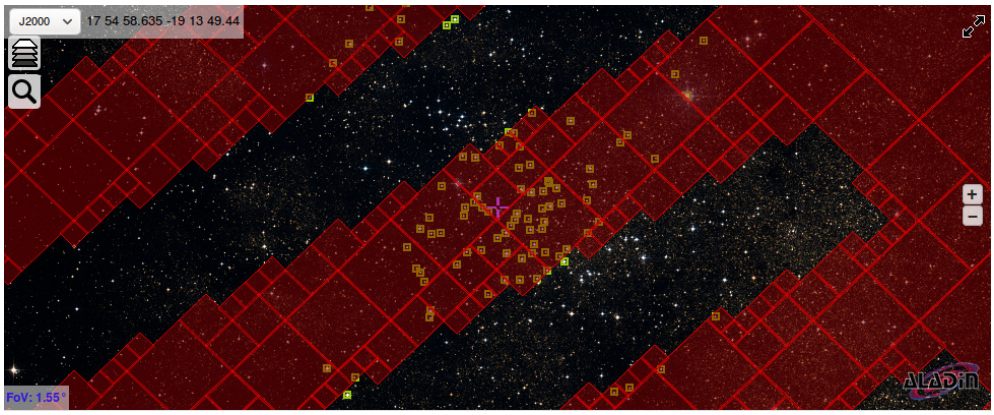

Figure 3. Aladin-lite view showing a Vizier table filtered by a MOC

3. Future Improvements

- MOCPy (Boch & Baumann 2015) is currently being integrated into astropy-regions. New classes, `MOCSkyRegion` and `MOCPixelRegion` will be implemented.
  `MOCSkyRegion` is the equivalent of the `mocpy.MOC` class, therefore it will contain all its features (serialization, intersection, ...). A `MOCPixelRegion` is a MOC sky region projeted using an astropy WCS object.
- `query_region` from astroquery.cds will be upgraded to accept `MOCSkyRegion` objects.
- The `query_region` methods of both astroquery Simbad and Vizier should accept `MOCSkyRegion` too so that Simbad and Vizier tables can be filtered by MOCs.

References

Baumann, M. 2018a, astroquery.cds documentation page,
`https://astroquery.readthedocs.io/en/latest/cds/cds.html`
— 2018b, Notebook example illustrating the state of the art of the CDS Python tools,
`https://github.com/cds-astro/ADASS-IVOA18`
Boch, T., & Baumann, M. 2015, Python library to easily create and manipulate MOCs (Multi-Order Coverage maps) , `https://github.com/cds-astro/mocpy`
Fernique, P., Allen, M., Boch, T., Donaldson, T., Durand, D., Ebisawa, K., Michel, L., Salgado, J., & Stoehr, F. 2017, HiPS - Hierarchical Progressive Survey Version 1.0, IVOA Recommendation 19 May 2017. `1708.09704`