Master 2 d'astrophysique - projet informatique Python 2023

G.Landais

October 20, 2023

Important : refer to the "Terms and Conditions" section at the end of the document for details on the delivery of the work (in particular for the deadline for submission of Friday, January 05th, 2024).

Note : Thank you to report immediately any misunderstanding or every technicals issues like library not installed or difficulties to install tools required for the project (refer to the section "Pre-requisite").

1 Project description

Build a Python MOC library that creates a MOC from list of HEALPix cells.

The project must provide a python code with its inline-documentation, a notebook and a **ReadMe** (formatted in Markdown (see https://en.wikipedia.org/wiki/Markdown)). The ReadMe file gives a brief abstract of the project with installation instructions.

Pre-requisite. For this project, you will use:

- astropy : https://docs.astropy.org/en/stable/
- astropy-healpix: https://astropy-healpix.readthedocs.io/
- mocpy: https://cds-astro.github.io/mocpy/stubs/mocpy.MOC.html
- Jupyter notebook

In options:

- ipyaladin: https://github.com/cds-astro/ipyaladin
- pyvo: https://pyvo.readthedocs.io/
- astroquery: https://astroquery.readthedocs.io/

2 Detail of the project

The mocpy (section 1) library is a python interface which exploits a RUST code. mocpy is a complete library providing MOC (see 2.2, p 2) serialization and functions to make operations (union, intesection, etc.).

We will build a (full) Python MOC code that allows to create a MOC serialization in text format from a list of HEALPix (see 2.1, p. 1). To do it we will use the astropy-healpix (section 1)library.
In the second part of the project , you will illustrate the MOC serialization using your MOC code and the mocpy library.

2.1 HEALPix

General documentation: https://healpix.jpl.nasa.gov/

The HEALPix pixelation covers the sky with pixels (HEALPix) of equal area. The subdivision of the sky can be done at different depth (or order). In the first level , the sky is divided into 12 pixels. Then the second level (order 2) is the result of dividing each pixel (order 1) into 4 pixels of the same area. The order of the pixelation will define the accuracy of the HEALPix pixelation.

Often we use the term nside: $nside = 2^{order}$

HEALPix numbering: each HEALPix cell has a number (=ipix). There are 2 numbering methods called 'NEST' and 'RING'. **In this project, we will use the NESTED numbering only**.

With nested numeration, the decrease/increase operations are simple **binary operations** (<< and >>):

- To go from order (n) to order (n+1), we divide the HEALPix cell (HEALPIx number = ipix) in 4 sub-cells : $ipix << 2, (ipix << 2) + 1, (ipix << 2) + 2, (ipix << 2) + 3$
- To go from order (n) to order (n-1): $ipix >> 2$

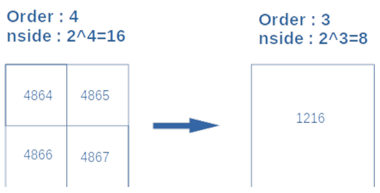

Figure 1: HEALpix decrease operation

2.2 MOC(Multi Order Coverage)

Official documentation: https://ivoa.net/documents/MOC/

The MOC(Multi Order Coverage) is a combination of HEALPix cells in different orders. MOC allows to minimize the number of cells. The idea consists to join adjacent cells in a given order into their HEALPix cell of the lower order.

Example: MOC serialization of a circle center=(0, 0), radius=0.5deg in order 8

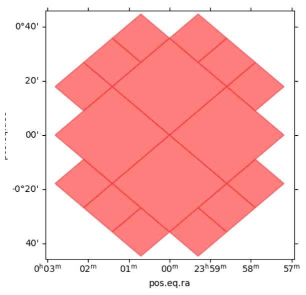

Figure 2: MOC of a circle

MOC serialization in text format:
syntax: order/ipix1 ipix2... order/ipixn ... (where 'order' is the order level)

```
7/69631 72362 75093 77824
8/278518-278519 278521 278523 289442 289454 300369 300381 311300 311302 311304-311305
```

There are different approaches (more or less efficient) do create a MOC from a list of HEALPix cell. For instance, we can have a recursive hierarchical approach and consider a MOC tree (see fig. p.3)
The figure is a extract of a MOC tree. Each node is an HEALPix in a given order (n). Each node can have between 0 to 3 sub nodes (HEALPix cells in the order (n+1)) .When the sub-node is completed (cells with border in the figure) the childs node are removed.

3 TODO

1. Extract table in VOTable format (see the examples in section 4, p.3)).
   Choose an API todo that (eg: astropy, astroquery or pyvo).
   Note: choose a table having position and query a reasonable number of records

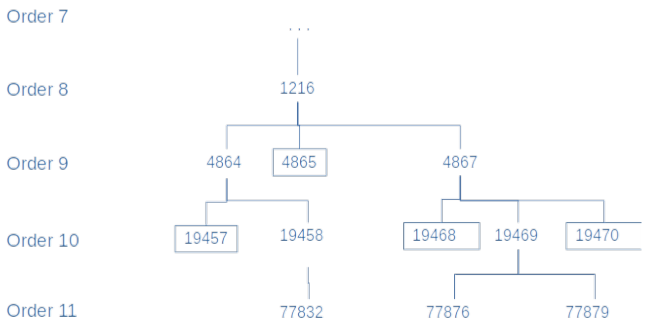

Figure 3: MOC tree

2. Build an astropy table with an added HEALPix column in the order that you choose (eg: order 10) You can use the astropy-healpix library to compute the HEALPix number from ra,dec.
3. Create a simple HEALPix pixelation from the HEALPix column (so a kind of MOC but in a unique order). Your code must contains a text serialization.
   Example of HEALPix pixelation serialization: 6/17407 18090 18773 19456
4. Extend the code to build a real MOC (with HEALPix in different order) and apply it to the HEALPix column. Your code must contains a text serialization (like example 2.2).

   Todo that, you can (but you can follow an other algorithm) have a recursive approach (see 2.2) that manages cells in a structure (made of dictionaries, lists, objects) and functions/methods that manages node operations (like to add or delete a node). Then you iterate the HEALPix values and feed one by one the MOC tree.
5. Illustrate the result in a Jupyter notebook with plots (matplotlib) or in Aladin (module ipyaladin).
   In the notebook, use your serialization and mocpy (especially for functions like plots or moc-operation only available in mocpy).

Note: to test, you can use the mocpy library and check if the results are similar. You can also plot the result using astropy (see the mocpy doc.). The mocpy library is able to check some inconsistencies like overlapping elements.
Note: for this project a debugger could help (for instance pdb)

4 Examples

Here are a list of VizieR example:

- table II/7A/catalog : UBVRIJKLMNH Photoelectric Catalogue (Morel+ 1978)
- table J/ApJ/831/67/table1: Galactic CHaMP. III. 12CO dense clump properties (Barnes+, 2016)
- table J/ApJ/804/L15: SDSS-DR7 broad-line QSOs (Sun+, 2015)

VizieR template to extract a VOTable: https://vizier.cds.unistra.fr/viz-bin/votable?-source=.....&-out.max=10000 where .... is the table name (the option -out.max limits the number of records)

5 Terms and Conditions

Please, acknowledge the receipt of the project to gilles.landais@unistra.fr.

The requested work is a personal project, the code must not be shared (even partially)
The deadline for submission is Friday, January 05th, 2024.

Submission modalities :

- Gather all documents including code and results (png, notebook, .py, setup, ReadMe) into a .zip file.
- Then send the archive to gilles.landais@unistra.fr. I will confirm the good reception after verification (readable files after extraction) of the contents of the archive.