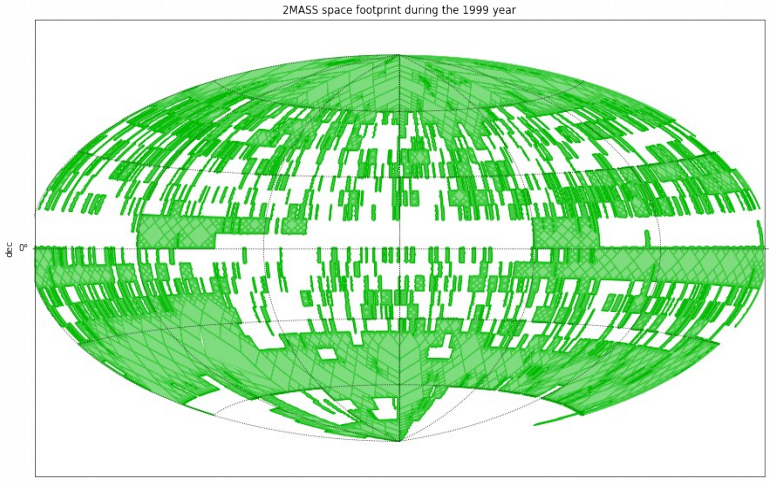
Recent STMOC developments at CDS : support in MOCPy, STMOC for VizieR tables

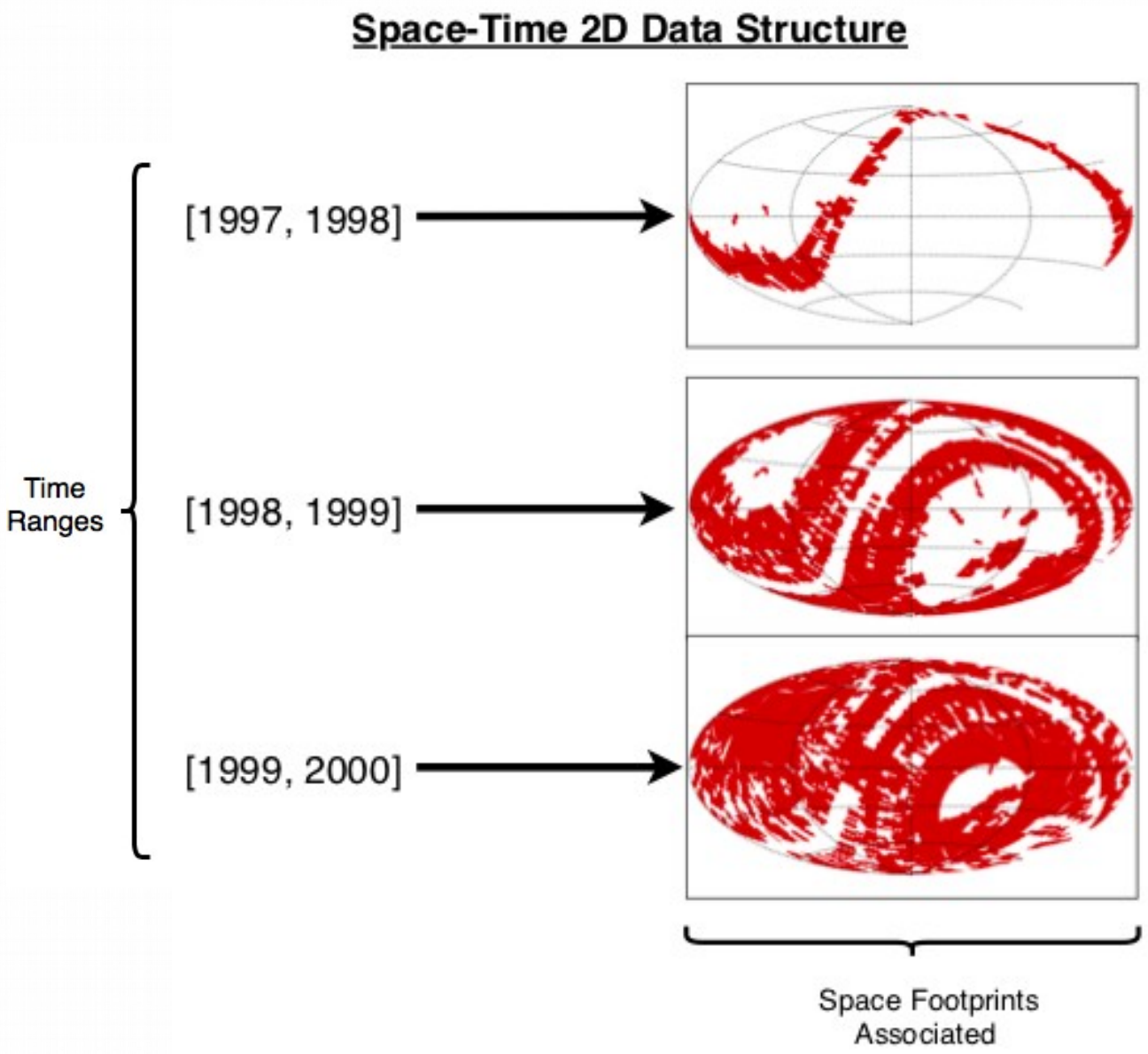
IVOA meeting Groningen, Netherlands
10/12/2019

M. Baumann
T. Boch
F.-X. Pineau
P. Fernique
A. Nebot



Space-Time Coverage

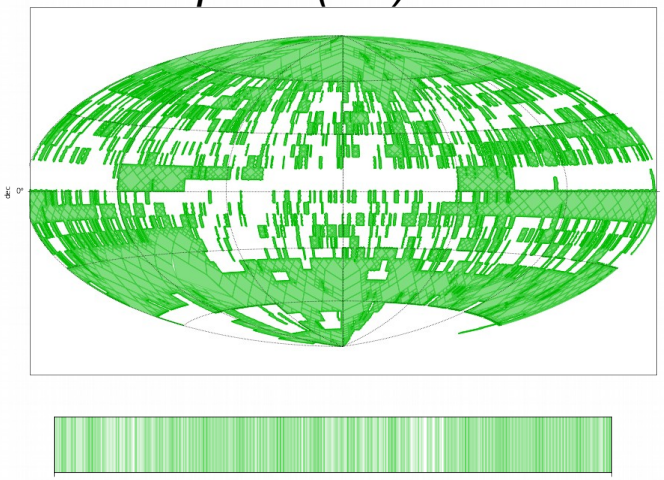
- A spatial coverage (i.e. MOC) with **time information.**
- 2D data-structure :
  - List of **time ranges**, each **linked to a spatial coverage**.
  - 1st axis: **Time**
  - 2nd axis: **Space** (HEALPix indices)
- Why ST-MOCs ?
  - Allows representing the **time evolution** of a **spatial footprint**, **query it by time**, **space**, **filter** a catalog.
  - Way of getting a **light footprint** of the catalogs (**10, 100x less data** depending on the space & time resolutions chosen) !



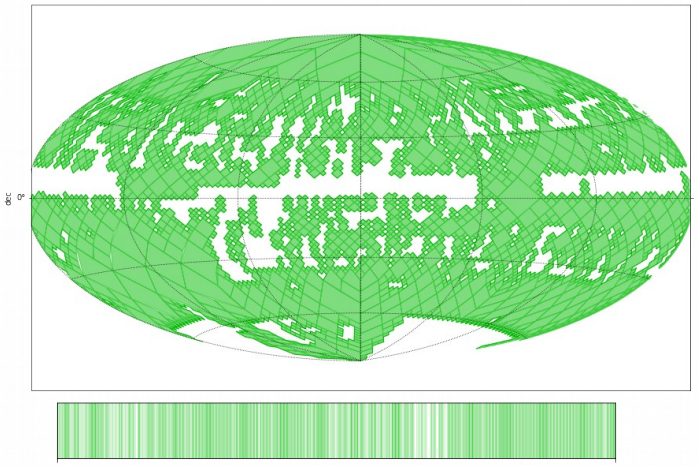
What's new in MOCPy ?

- ST-MOCs of course !
- API: new mocpy.STMOC class
  - **Creation**
    - astropy skycoords/ single times (range in the future)
    - spatial/temporal resolution
  - **Logical operations** (e.g. intersection of the XMM and Chandra ST-MOCs to find simultaneous observations).
  - Astropy table **filtering**

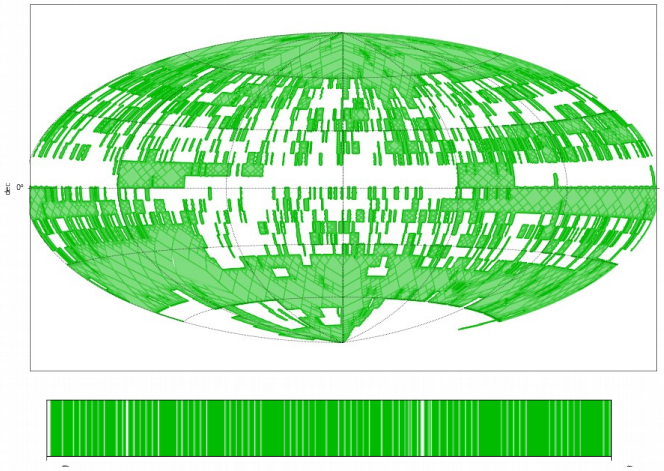
14 in *Time (~17 min)*,
7 in *Space (27')*



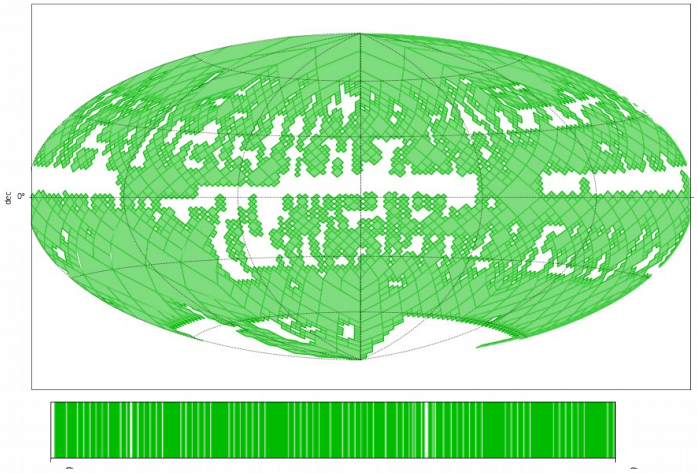
14 in *Time (~17 min)*,
5 in *Space (~1.8 deg)*



11 in *Time (~19 hours)*,
7 in *Space (~27')*

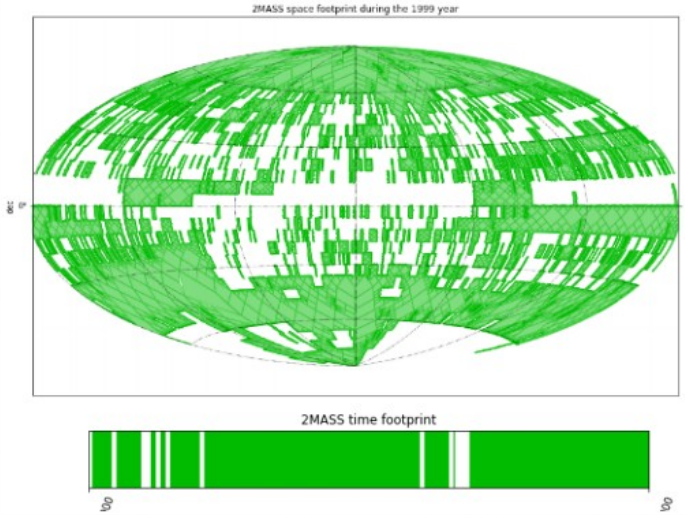


11 in *Time (~19 hours)*,
5 in *Space (~1.8 deg)*



What's new in MOCPy ?

- **Serialize** to FITS



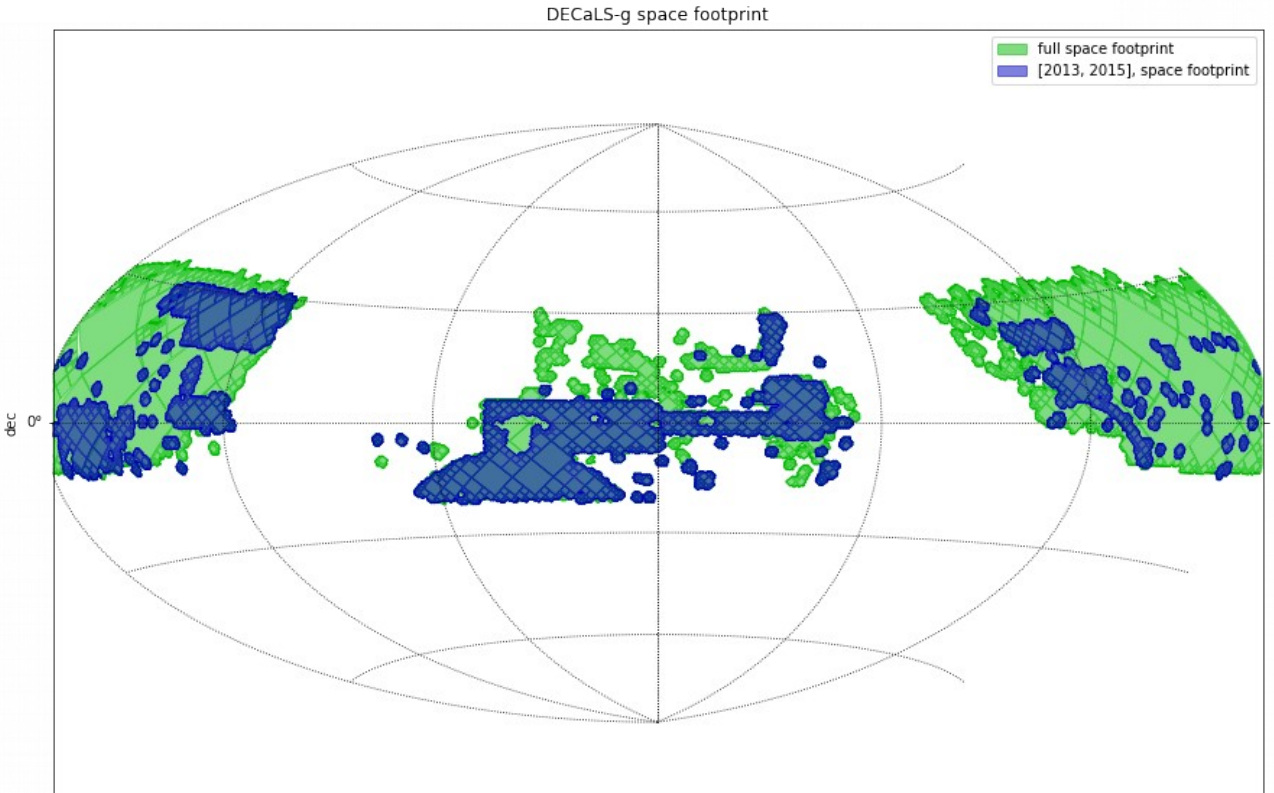
Serialization to a [int64]

[-134524, -247863, 78634, 722239, ..., 987834,
...,
-88765, -87736, 1247662, 2377875, ..., 87384]

Time ranges | HEALPix cell indices

- **Query** by **Time/Space**

DeCaLs full STMOC
queried by a **time range**



What's new in MOCPy ?

~160 ST-MOCs generated from VizieR catalogs having time and positions data (M.Baumann, T. Boch, A. Nebot)
http://alasky.u-strasbg.fr/footprints/STMOC/

Portal Simbad VizieR Aladin X-Match Other Help

STMOCs

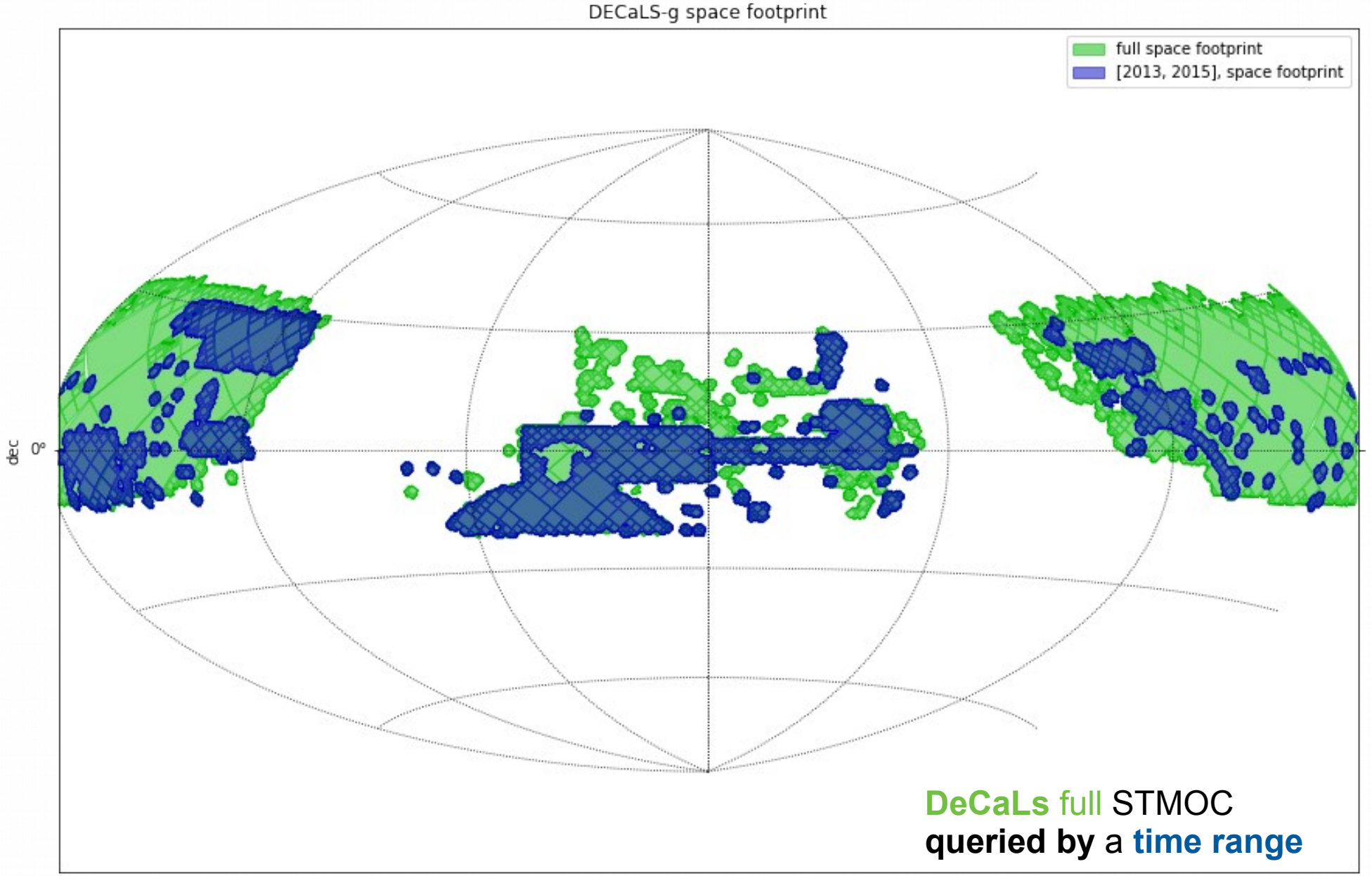
Show 10 entries Search:

Table ID	Catalogue name	Table name	#records	Time min	Time max
B/assocdata/obscore Get STMOC	Associated data in VizieR (G.Landais, 2016)	VizieR Spectra, images gathered in a table	8033403	1970-01-01 00:38:56.436	8579-01-20 08:37:44.003
B/chandra/chandra Get STMOC	The Chandra Archive Log (CXC, 1999-2014)	The Chandra Log (2019-09-29)	19501	1999-08-14 10:44:45.364	2021-02-24 00:01:06.023
B/gcvs/evs_cat Get STMOC	General Catalogue of Variable Stars (Samus+, 2007-2017)	Extragalactic Variable Stars. Catalogue (Vol. V)	10979	1885-08-21 11:57:55.572	1991-04-25 12:04:21.652
B/gcvs/gcvs_cat Get STMOC	General Catalogue of Variable Stars (Samus+, 2007-2017)	GCVS catalog (GCVS 5.1, version March, 2017)	53626	-4711-04-17 11:59:08.538	-4441-01-23 12:01:20.478
B/occ/moon Get STMOC	Occultation lights curves (Herald+ 2016)	table description	6358	1998-09-12 07:12:59.638	2019-07-27 20:18:05.104
B/swift/swiftlog Get STMOC	Swift Master Catalog (HEASARC, 2004-)	SWIFT logs	251242	2005-09-08 23:57:08.955	2019-09-27 00:02:48.620
B/vsx/vsx Get STMOC	AAVSO International Variable Star Index VSX (Watson+, 2006-2014)	Variable Star indeX, Version 2019-09-30	1390973	1585-01-31 11:59:30.378	2132-08-31 12:00:08.651
B/xmm/xmmlog Get STMOC	XMM-Newton Observation Log (XMM-Newton Science Operation Center, 2012)	The XMM-Newton Observation log (2019-09-30)	14408	2000-01-17 15:26:49.479	2019-09-20 03:19:54.698
I/337/cepheid Get STMOC	Gaia DR1 (Gaia Collaboration, 2016)	Cepheid stars identified in table VariableSummary as classification="CEP"	599	2014-05-03 20:42:41.343	2014-08-16 19:41:08.890
I/337/rrlyrae Get STMOC	Gaia DR1 (Gaia Collaboration, 2016)	RRLyrae stars identified in table VariableSummary as classification="RRLYR" (Rrlyrae)	2595	2014-07-23 05:43:48.533	2014-08-18 21:54:53.724

Showing 1 to 10 of 169 entries Previous 1 2 3 4 5 … 17 Next

Jupyter notebook

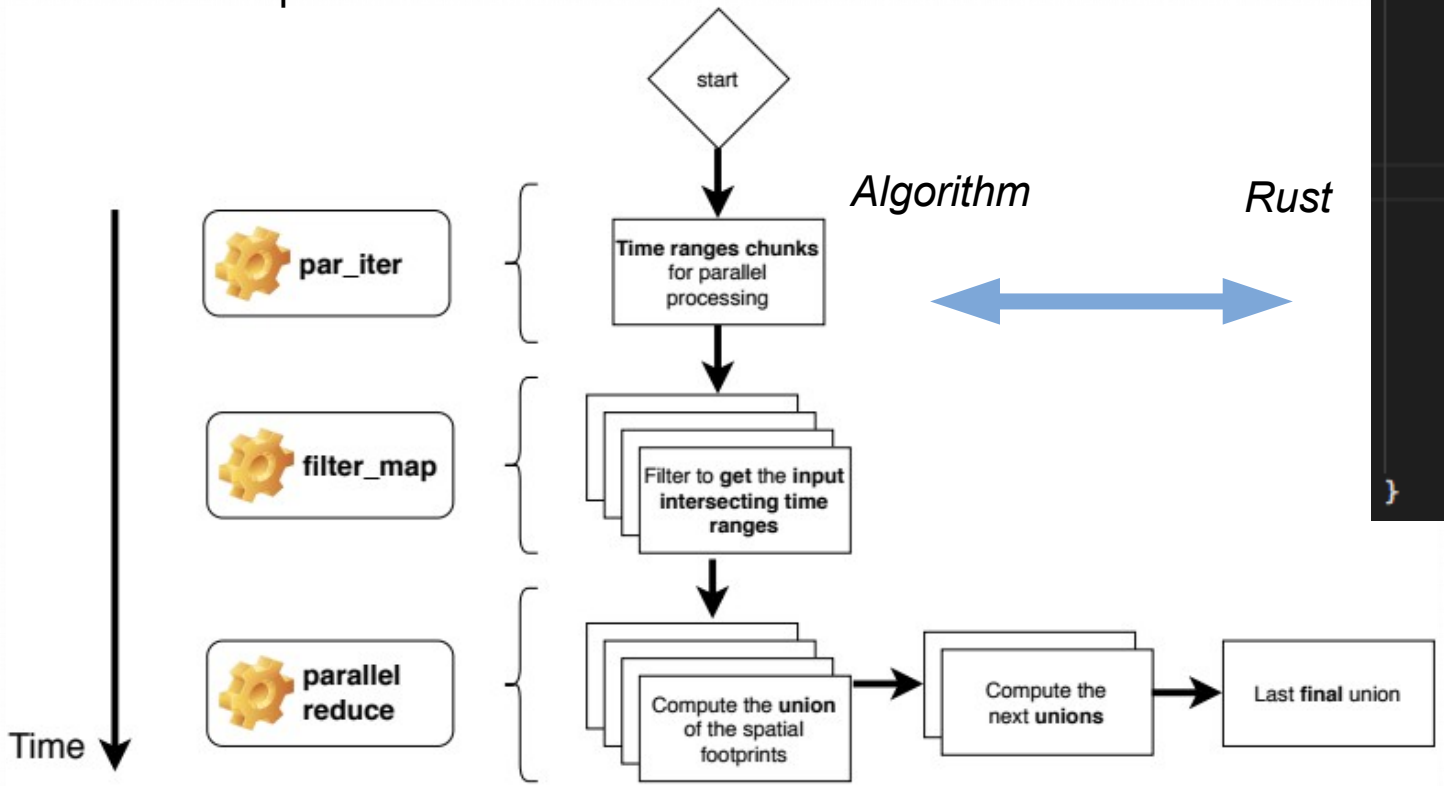
Demo (M. Baumann, A. Nebot)



Implementation details

- **All core code** written in **Rust** system programming langage
  * Compiler checks ensuring data race errors (RAII)
  * Easy parallelism thanks to **rayon** library

- Example of how Rust handles concurrency (**query_by_time** algorithm):
  (1) Parallel iterators : divide time ranges into chunks
  (2) For each chunk of ranges, **filter** those intersecting the input ranges
  (3) Parallel reduce: **union** of all the remaining spatial footprints.




```rust
pub fn project_on_second_dim(
    x: &NestedRanges<T>,
    coverage: &NestedRanges2D<T, S>,
) -> NestedRanges<S> {
    let coverage = &coverage.ranges;
    let ranges = coverage.x
        .par_iter()
        .zip_eq(coverage.y.par_iter())
        // Filter the time ranges to keep onl
        // that intersects with ``x``
        .filter_map(|(t, s)| {
            if x.intersects(t) {
                Some(s.clone())
            } else {
                None
            }
        })
        // Compute the union of all the 2nd
        // dim ranges that have been kept
        .reduce(
            || Ranges::<S>::new(vec![]),
            |s1, s2| s1.union(&s2),
        );

    ranges.into()
}
```

Rust/Python PyO3 binder

```rust
/// Compute the depth of a spatial coverage
///
/// # Arguments
///
/// * ``ranges`` - The input coverage.
#[pyfn(m, "coverage_depth")]
fn coverage_depth(_py: Python, ranges: &PyArray2<u64>) -> i8 {
    let ranges = ranges.as_array().to_owned();

    let coverage = coverage::create_nested_ranges_from_py(ranges);

    coverage::depth(&coverage)
}
```

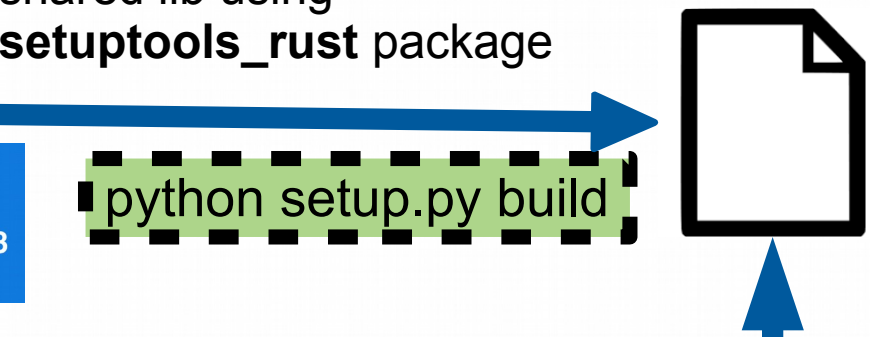
Rust core code

1. Easy generation of shared lib using **setuptools_rust** package

PyO3

python setup.py build

core.so



2. Call *core.so* from python side API code

Pros :

- **clean**: no third C langage
- keep the benefits of Rust (no data race errors) & Python GC
- **PyO3** good doc, reliable solution, can **raise Python exceptions from Rust** side code.
- **numpy wrapper available in Rust** for handling numpy arrays

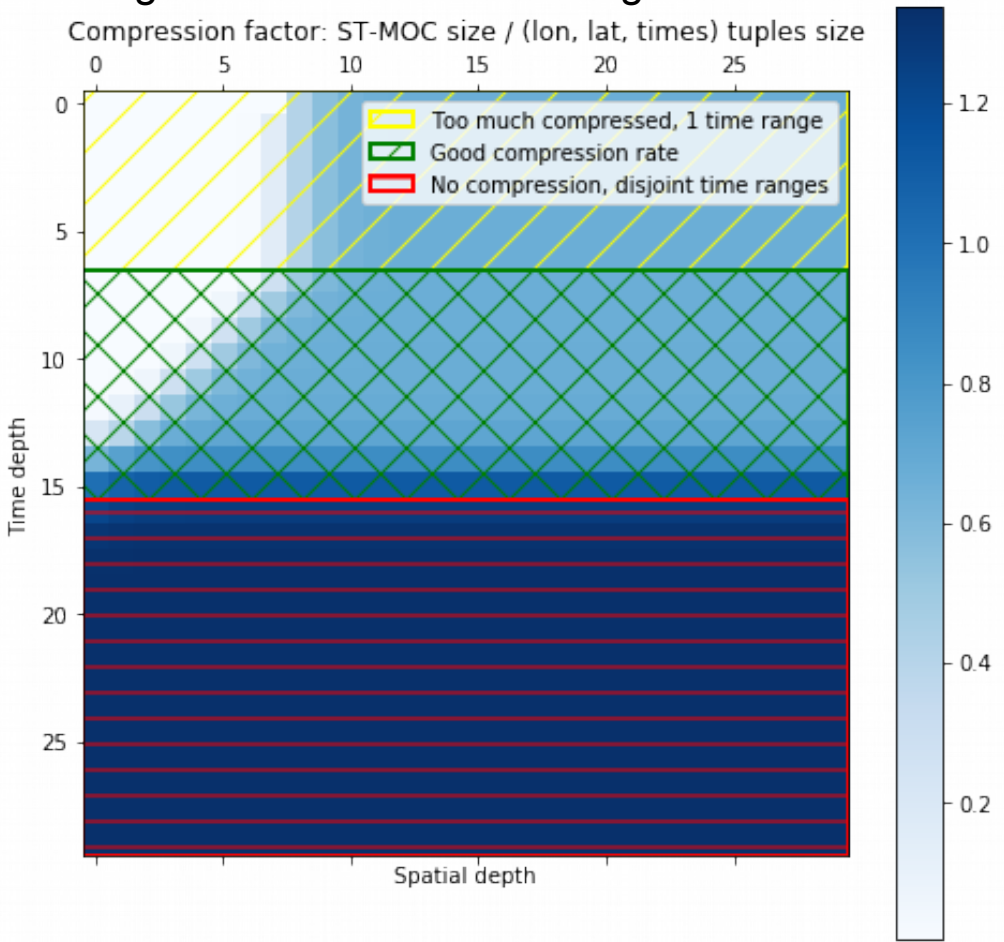
```python
@property
def max_order(self):
    """
    Depth of the smallest HEALPix cells found in the MOC instance.
    """
    depth = core.coverage_depth(self._interval_set._intervals)
    depth = np.uint8(depth)
    return depth
```

Python API code

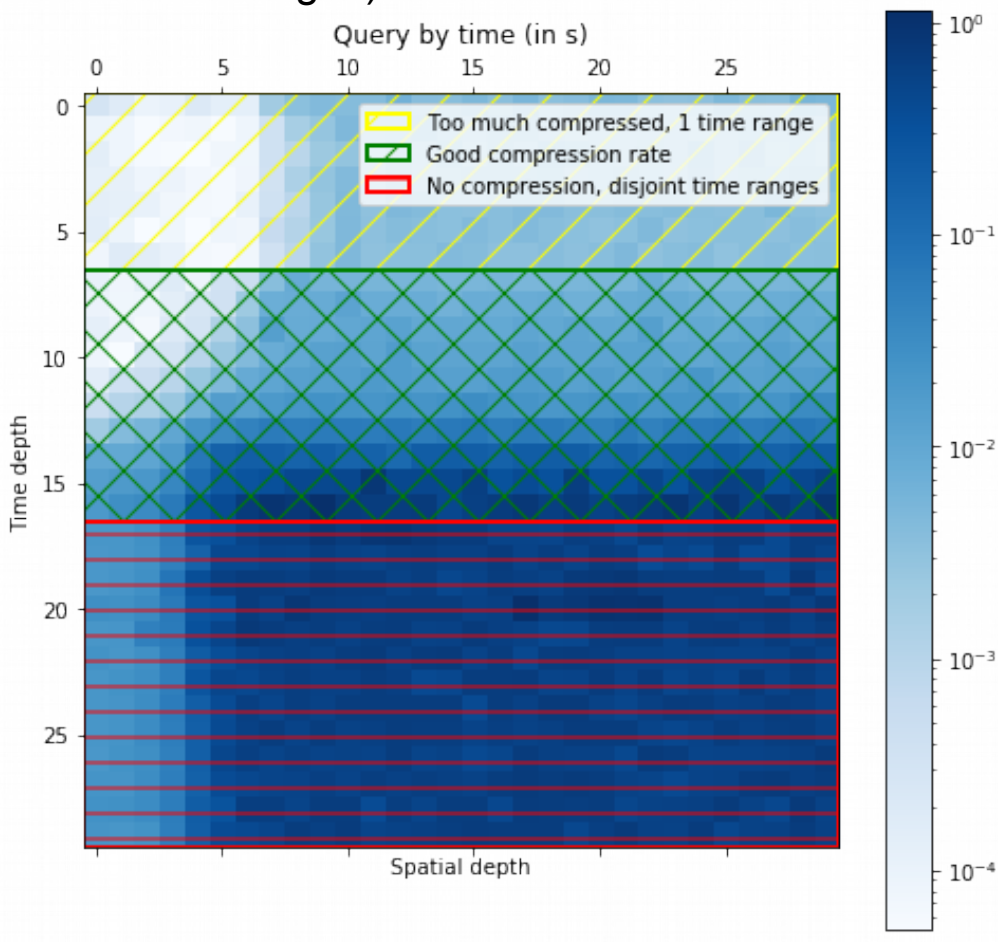
ST-MOC performance

- **Worst case study** :
  * 100000 (position, time) tuples at a random sky location and time (in a 1 year frame)
  * Time depth < 6 (~resolution 814 days), **time ranges all merged into one...** Loose of info
  * Time depth > 15 (~resolution 4min30s) **time ranges all disjoint**, no grouping… No compression

- **Compression factor gain** w.r.t the (space, time) resolution depths compared to the original data used for building the ST-MOC



- **Query by time performance** w.r.t the (space, time) resolution depths (union of all the time ranges)



Some Useful Links

- **GitHub**: https://github.com/cds-astro/mocpy
  * Links, Issue posting, Contributing instructions
- **New documentation**: https://cds-astro.github.io/mocpy/

- **Test it:** Space & Time coverage notebook:

https://mybinder.org/v2/gh/cds-astro/mocpy/master

- On **PyPI**: https://pypi.org/project/MOCPy/

Binary wheels for 32/64 Linux, Windows and MacOS

```
pip install --upgrade mocpy
```