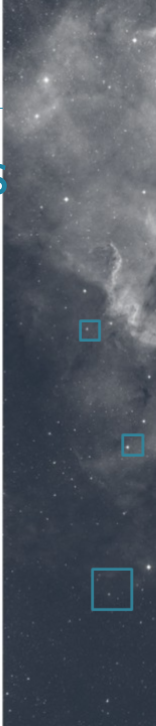
~~Feedback on VOTable and~~ Fast STC-S queries thanks to B-MOCs

F.-X. Pineau[1]

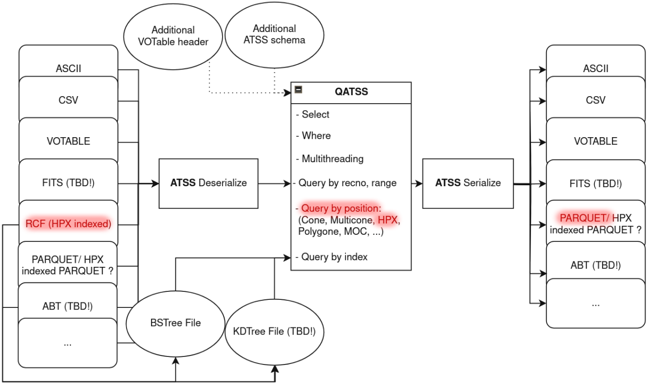
[1]Centre de Données astronomiques de Strasbourg

20th May 2023



Context: QATSS

- **QATSS** (or QAT2S): **Q**uery **A**stronomical **T**able **S**erialization **S**ystem: see **ADASS 2023** (Tucson) **poster**



Additional VOTable header

Additional ATSS schema

ASCII

CSV

VOTABLE

FITS (TBD!)

RCF (HPX indexed)

PARQUET/ HPX indexed PARQUET ?

ABT (TBD!)

...

ATSS Deserialize

QATSS

- Select
- Where
- Multithreading
- Query by recno, range
- Query by position: (Cone, Multicone, HPX, Polygone, MOC, ...)
- Query by index

ATSS Serialize

ASCII

CSV

VOTABLE

FITS (TBD!)

PARQUET/ HPX indexed PARQUET ?

ABT (TBD!)

...

BSTree File

KDTree File (TBD!)

- RCF input + HEALPix queries + Parquet output: could allow to build a layer compatible with V. Rubin “HiPSCat”

About V. Rubin “HiPSCat”

- Name clash with:
  - existing HiPS catalogues described in:
    - the IVOA standard
    - 2015A&A...578A.114F (Fernique et al.)
  - CDS HiPSCat service
  - CDS hipsgen-cat tool described here
- HiPS standard: **P** stand for **Progressive**
  - *“the more you zoom, the more information you get”*
  - rows even at order 0
  - rows at order $n + 1$ complementary to rows at order n
- V. Rubin “HiPSCat” more a Multi-Resolution HEALPix Maps (MRM) / Multi-Order Maps (MOM)
  - *value* associated to an HEALPix cell = list of rows
  - *xmatch* = specific operator when merging 2 MRCs (Multi Resolution Catalogues), which are specific MRMs
  - possible to *emulate* HiSP?

About VOTable
(slides kept, but not enough time to talk about this!)

VOTable needs in QATSS

In QATSS, we wanted to:

- support VOTable inputs
- support VOTable outputs
- enrich current CDS large catalogue files (RCF) metadata
    - store VizieR table metadata in a separate file
    - update the metadata
        - manually (`TOML` representation of a VOTable)
        - automatically when possible (remove/rename elements)
    - support VOTable metadata inputs

VOT Lib Rust / VOT Cli

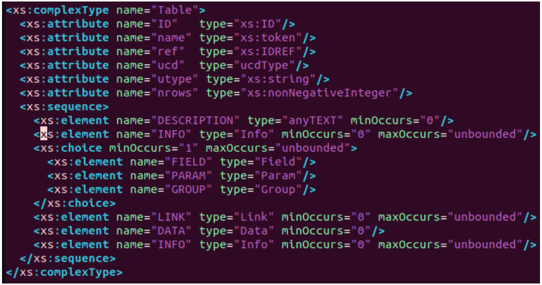
QATSS is **full Rust**, so we have been developping:

- a ***VOTable* librarie in Rust**: *VOT Lib Rust*
  - open source
    - MIT license
    - available on github and crates.io
  - ⇒ re-usability of the code in other projects
- a **standalone tool** to convert and edit VOTables: ***vot-cli***
  - open source
    - MIT license
    - available on github
  - pre-compile executable on github release and pypi
  - serves as documentation for *VOT Lib Rust*
  - serves as bench for *VOT Lib Rust*

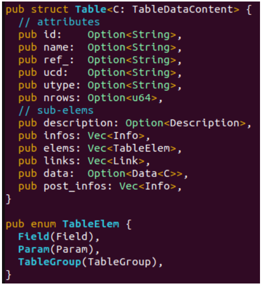
Feedbacks on VOTable 1/3

One goal was to allow **round-trip conversion of the VOTable from XML to** other formats (**JSON/TOML/YAML**)

- How?
  - internal representation is a mix of Rust Structures and Enums
  - rely on serde.rs crate to support JSON/TOML/YAML/. . .



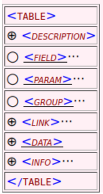
```xml
<xs:complexType name="Table">
  <xs:attribute name="ID"   type="xs:ID"/>
  <xs:attribute name="name" type="xs:token"/>
  <xs:attribute name="ref"  type="xs:IDREF"/>
  <xs:attribute name="ucd"  type="ucdType"/>
  <xs:attribute name="utype" type="xs:string"/>
  <xs:attribute name="nrows" type="xs:nonNegativeInteger"/>
  <xs:sequence>
    <xs:element name="DESCRIPTION" type="anyTEXT" minOccurs="0"/>
    <xs:element name="INFO" type="Info" minOccurs="0" maxOccurs="unbounded"/>
    <xs:choice minOccurs="1" maxOccurs="unbounded">
      <xs:element name="FIELD" type="Field"/>
      <xs:element name="PARAM" type="Param"/>
      <xs:element name="GROUP" type="Group"/>
    </xs:choice>
    <xs:element name="LINK" type="Link" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="DATA" type="Data" minOccurs="0"/>
    <xs:element name="INFO" type="Info" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```



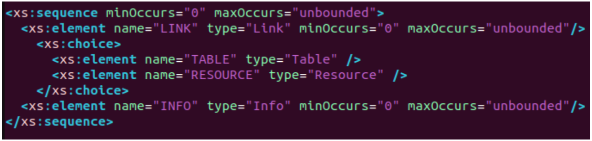
```rust
pub struct Table<C: TableDataContent> {
  // attributes
  pub id:    Option<String>,
  pub name:  Option<String>,
  pub ref_:  Option<String>,
  pub ucd:   Option<String>,
  pub utype: Option<String>,
  pub nrows: Option<u64>,
  // sub-elems
  pub description: Option<Description>,
  pub infos: Vec<Info>,
  pub elems: Vec<TableElem>,
  pub links: Vec<Link>,
  pub data:  Option<Data<C>>,
  pub post_infos: Vec<Info>,
}

pub enum TableElem {
  Field(Field),
  Param(Param),
  TableGroup(TableGroup),
}
```

Feedbacks on VOTable 1/3

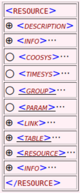
- Problems/difficulties:
  - INFO both before and after RESOURCES or DATA
    - post-INFO: “*post-operational diagnostics*”
    - solution: two attributes (*infos* / *post_infos*)
    - (missing pre-INFO in TABLE schema in §7.1)
  - FIELDRef only possible in GROUP in TABLE
    - “*FIELDRef element defined by referring to a FIELD element defined elsewhere in the parent TABLE*”
    - solution: 2 structures (Group and TableGroup)
  - XSD choices replaced by a Rust Enum, but we created a ResourceSubElem to support the following XSD:
    - (RESOURCE schema oversimplified in §7.1)



<TABLE>
⊕ <DESCRIPTION>
○ <FIELD>···
○ <PARAM>···
○ <GROUP>···
⊕ <LINK>···
⊕ <DATA>
⊕ <INFO>···
</TABLE>



```
<xs:sequence minOccurs="0" maxOccurs="unbounded">
  <xs:element name="LINK" type="Link" minOccurs="0" maxOccurs="unbounded"/>
    <xs:choice>
      <xs:element name="TABLE" type="Table" />
      <xs:element name="RESOURCE" type="Resource" />
    </xs:choice>
  <xs:element name="INFO" type="Info" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
```



<RESOURCE>
⊕ <DESCRIPTION>
⊕ <INFO>···
○ <COOSYS>···
○ <TIMESYS>···
○ <GROUP>···
○ <PARAM>···
⊕ <LINK>···
⊕ <TABLE>···
⊕ <RESOURCE>···
⊕ <INFO>···
</RESOURCE>

vot-cli perf

BINARY to TABLEDATA conversion test made by Renaud Savalle:

- Input: 19 GB, 49 202 126 rows, 33 columns
- Output: 32 GB
- Hardware: 2 distinct **SATA SSDs** to read/write, >16 threads
- Time: **3m**, i.e. **110 MB/s** read and **180 MB/s** write
- Single CPU read limitation (base64 + find row bounds)?

```
> time vot sconvert --parallel 16 \
>  -i input_xml-bin.vot -o output_xml-td.vot -f xml-td
real    2m57.473s
user    9m50.099s
sys     0m55.046s
```

vot-cli perf

`TABLEDATA` to `BINARY` conversion test:

- Input: 2.8 GB, 225 columns, 1 000 000 rows
- Output: 1.5 GB
- Hardware: **MVNe SSD raid**, lot of CPUs (we use 30 threads)
- Time: **4.43 s**, i.e. **900 MB/s read + 350 MB/s write**
  - Same perf (4.6s) as: `grep '</TR>' gaia_dr3.vot | wc -l`
- `BINARY` to `TABLEDATA`: *16s*, i.e. **100 MB/s read**
- I/O or single CPU read limitation (see *real* vs *user*)?

```
> time vot sconvert --in gaia_dr3.vot --out out.bin.vot \
>                   --out-fmt xml-bin --parallel 30
real    0m4,430s
user    1m5,025s
sys     0m4,518s
```

Feedbacks on VOTable 2/3

About multi-threading reading performances

- Multi-threading strategy (assuming streaming):
  - one thread read the input and identify raw rows
    - bytes between `<TR>` and `</TR>` for `TABLEDATA`
    - base64 decoded bytes for `BINARY` and `BINARY2`
  - the same thread create chunks of N raw rows
  - multiple threads convert chunks of input raw rows into chunks of output raw rows
    - one chunk per thread
  - one thread take output chunks and write the result
    - also performs the base64 encoding for `BINARY` and `BINARY2`

Feedbacks on VOTable 2/3

About multi-threading reading performances

- `TABLEDATA`
  - performances limited by collisions between `</TD>` and `</TR>`?
    - `time grep '</TR>' gaia_dr3.vot | wc -l` ⇝ 4.6 s
    - `time cat gaia_dr3.vot | wc -l` ⇝ 1.5 s
    - `time wc -l gaia_dr3.vot` ⇝ 0.5 s
- `BINARY` and `BINARY2`
  - current *vot-lib-rust* / *vot-cli* implementation:
    - slow code to clean base64 string (space, `\n`, ...)?
    - try a faster base64 decoding library?
  - we have to read variable size elements to know raw row length!
    - what about starting each row by its byte length?
    - what about making distinctive blocks of *xx* base64 rows?
    - or what about one bas64 encoded raw per line? (but padding pb)
- Ok, VOTable is verbose and not made for very high performances anyway...

Edition with vot-cli

How to edit a VOTable from the command line?

- How to easily point to any tag (to remove or edit it)?
  - Problem:
    - `ID` mandatory in `COOSYS` and `TIMESYS` **only**
    - `name` (unicity not guaranteed) mandatory in `FIELD`, `PARAM` and `INFO` **only**
  - Generic way: use XPATH
    - complexity due to genericity
    - performances?
  - `vot-cli`: use a Virtual ID defined for each tag, see
    - get virtual IDs with: `vot get -i vot.xml struct`
- `vot-cli edit`: use a part of the API from the command line

See vot-cli README file.

Feedbacks on VOTable 3/3

VOTable (in its current state) is not adapted as a universal astronomical table format:

- Supported types are limited:
  - `unsignedByte` but no `Byte`
  - `short/int/long` but no `unsigned short/int/long`
- Limited/optional formatting information:
  - No `sign`, no 0 padding, . . .
  - `width` and `precision` optional: formatting purpose or lossless decimal type?
- Data part:
  - too much verbose
    - `TABLEDATA`: `<TD></TD>` = 9 bytes per column
    - `BINARY/BINARY2`: 33% loss due to base64
  - lack a way to quickly identify row bounds (`\n` vs `</TR>`)
  - and/or lack a way to identify row blocks

Efficient STC-S queries thanks to B-MOC

Geometrical queries in QATSS

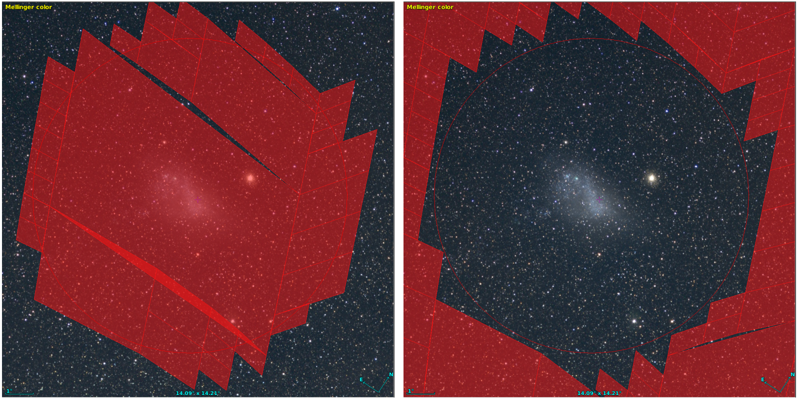
- Context: geometrical queries in QATSS
- Already supported:
  - **cone** / elliptical cone / multi-cone
  - **box** / **polygon**
  - **zone**
  - HEALPix cell / MOC
- Not supported:
  - complex operations (not, intersection, union, xor)
- What about supporting the **STC-S** note?
  - e.g. support query from ESO provided STC-S regions

STC-S to MOC

- Context: MOCPy users request
  - need for a *MOC from STC-S* method
  - Daniel Durand (among others)
  - TOPCAT, Aladin, Aladin Lite, . . . are STC-S aware since a long time
- 1st missing piece: STC-S parser in Rust
  - see STCSLibRust on github and crates.io
- 2nd missing piece: how to deal with complex geometries (NOT/DIFFERENCE/INTERSECTION)?
  - MOCLibRust
    - supports MOC from geometrical shapes (cone, polygon, . . .)
    - supports set operations on MOC (union, not, . . .)
    - **does not** support MOC from operations on geometrical shapes
  - CDS Healpix Rust
    - supports B-MOC

MOC

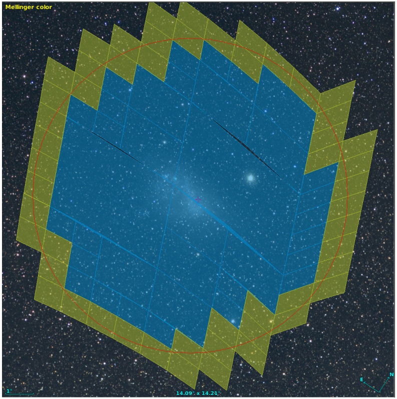
MOC(cone) and Not(MOC(cone))

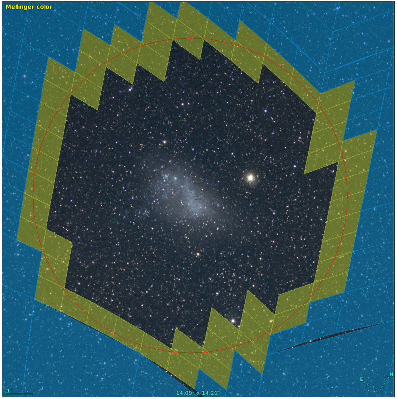


- all areas of interest **not covered** for *Not(cone)*

B-MOC

BMOC(cone) and Not(BMOC(cone))





- blue: cells fully covered by the cone
- green: cells partially covered by the cone
- all areas of interest **covered** for the *Not(cone)*

BMOC

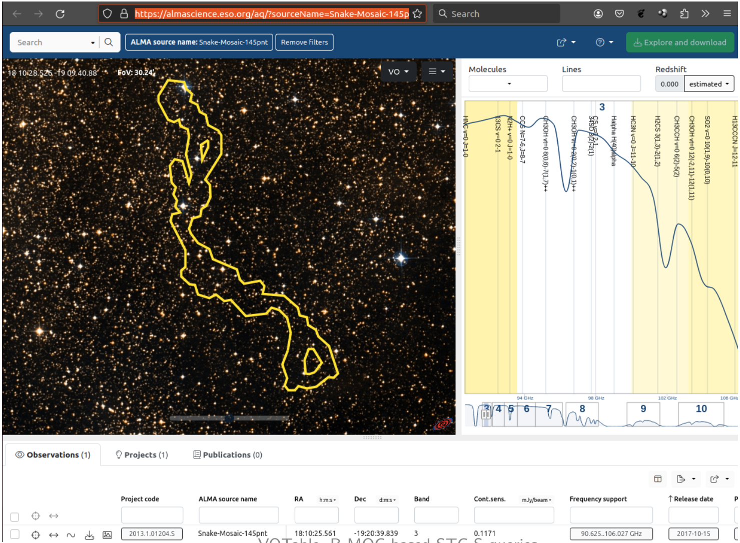
- BMOC: ***B*** stands for ***Boolean***-MOC
  - one boolean flag per HEALPix cell
  - *flag* = *true*: cell **fully covered** by the progenitor shape
  - *flag* = *false*: cell **partially covered** by the progenitor shape
- BMOC ≈ Multi-resolution Map with a boolean value
  - but: *order* + *cell index* + *value* encoded on a **single u64**
  - *order* + *cell index* encoded following ZUNIQ instead of UNIQ
    - ZUNIQ preserves the range representation order, thus. . .
    - . . . it allows for streaming operations
- Original motivation: **no post-filtering needed** for cells fully covered by a shape!
- The CDS HEALPix Rust library provides:
  - BMOC as output of shape coverage methods:
    - Cone, Elliptical cone, polygon, . . .
  - Set operations on BMOCs
    - Not, Union, Intersection, Minus, Xor

STC-S queries in QATSS

- 1 - parse STC-S
- 2 - convert STC-S into B-MOC (approximation)
- 3 - convert STC-S into a Spatial Filter (exact solution)
- 4 - retrieve rows from an HEALPix indexed format
  - if row is in a fully covered cell, return it
  - if row is on a border, apply Spatial Filtering

STC-S querie in QATSS

ALMA STC-S region provided by Felix Stoehr:



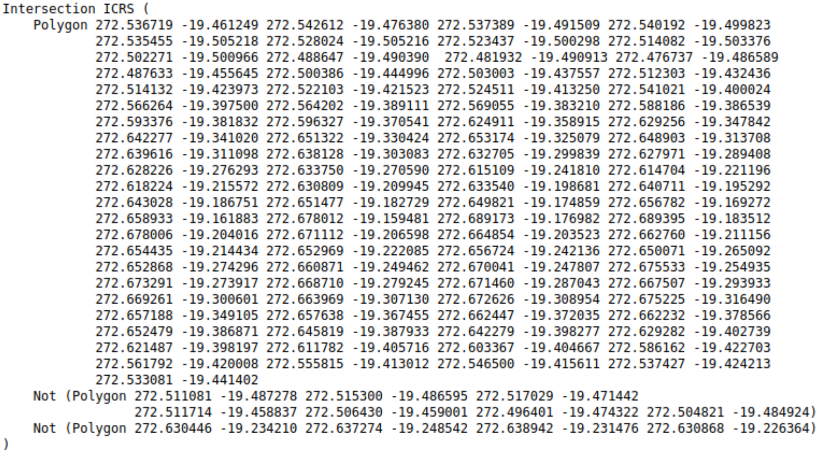
https://almascience.eso.org/aq/?sourceName=Snake-Mosaic-145p

ALMA source name: Snake-Mosaic-145pnt

Redshift: 0.000 estimated

	Project code	ALMA source name	RA h:m:s	Dec d:m:s	Band	Cont.sens. mJy/beam	Frequency support	↑ Release date
□	2013.1.01204.S	Snake-Mosaic-145pnt	18:10:25.561	-19:20:39.839	3	0.1171	90.625..106.027 GHz	2017-10-15

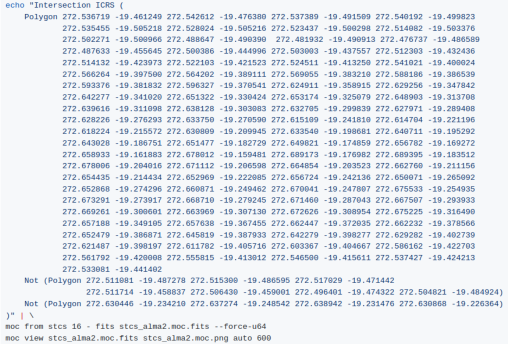
□ STC-S querie in QATSS



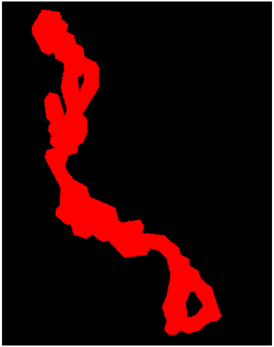
```
Intersection ICRS (
    Polygon 272.536719 -19.461249 272.542612 -19.476380 272.537389 -19.491509 272.540192 -19.499823
            272.535455 -19.505218 272.528024 -19.505216 272.523437 -19.500298 272.514082 -19.503376
            272.502271 -19.500966 272.488647 -19.490390  272.481932 -19.490913 272.476737 -19.486589
            272.487633 -19.455645 272.500386 -19.444996 272.503003 -19.437557 272.512303 -19.432436
            272.514132 -19.423973 272.522103 -19.421523 272.524511 -19.413250 272.541021 -19.400024
            272.566264 -19.397500 272.564202 -19.389111 272.569055 -19.383210 272.588186 -19.386539
            272.593376 -19.381832 272.596327 -19.370541 272.624911 -19.358915 272.629256 -19.347842
            272.642277 -19.341020 272.651322 -19.330424 272.653174 -19.325079 272.648903 -19.313708
            272.639616 -19.311098 272.638128 -19.303083 272.632705 -19.299839 272.627971 -19.289408
            272.628226 -19.276293 272.633750 -19.270590 272.615109 -19.241810 272.614704 -19.221196
            272.618224 -19.215572 272.630809 -19.209945 272.633540 -19.198681 272.640711 -19.195292
            272.643028 -19.186751 272.651477 -19.182729 272.649821 -19.174859 272.656782 -19.169272
            272.658933 -19.161883 272.678012 -19.159481 272.689173 -19.176982 272.689395 -19.183512
            272.678006 -19.204016 272.671112 -19.206598 272.664854 -19.203523 272.662760 -19.211156
            272.654435 -19.214434 272.652969 -19.222085 272.656724 -19.242136 272.650071 -19.265092
            272.652868 -19.274296 272.660871 -19.249462 272.670041 -19.247807 272.675533 -19.254935
            272.673291 -19.273917 272.668710 -19.279245 272.671460 -19.287043 272.667507 -19.293933
            272.669261 -19.300601 272.663969 -19.307130 272.672626 -19.308954 272.675225 -19.316490
            272.657188 -19.349105 272.657638 -19.367455 272.662447 -19.372035 272.662232 -19.378566
            272.652479 -19.386871 272.645819 -19.387933 272.642279 -19.398277 272.629282 -19.402739
            272.621487 -19.398197 272.611782 -19.405716 272.603367 -19.404667 272.586162 -19.422703
            272.561792 -19.420008 272.555815 -19.413012 272.546500 -19.415611 272.537427 -19.424213
            272.533081 -19.441402
    Not (Polygon 272.511081 -19.487278 272.515300 -19.486595 272.517029 -19.471442
                 272.511714 -19.458837 272.506430 -19.459001 272.496401 -19.474322 272.504821 -19.484924)
    Not (Polygon 272.630446 -19.234210 272.637274 -19.248542 272.638942 -19.231476 272.630868 -19.226364)
)
```

□ STC-S querie in QATSS

View MOC with moc-cli



```
echo "Intersection ICRS (
    Polygon 272.536719 -19.461249 272.542612 -19.476380 272.537389 -19.491509 272.540192 -19.499823
            272.535455 -19.505218 272.528024 -19.505216 272.523437 -19.500298 272.514082 -19.503376
            272.502271 -19.500966 272.488647 -19.490390  272.481932 -19.490913 272.476737 -19.486589
            272.487633 -19.455645 272.500386 -19.444996 272.503003 -19.437557 272.512303 -19.432436
            272.514132 -19.423973 272.522103 -19.421523 272.524511 -19.413250 272.541021 -19.400024
            272.566264 -19.397500 272.564202 -19.389111 272.569055 -19.383210 272.588186 -19.386539
            272.593376 -19.381832 272.596327 -19.370541 272.624911 -19.358915 272.629256 -19.347842
            272.642277 -19.341020 272.651322 -19.330424 272.653174 -19.325079 272.648903 -19.313708
            272.639616 -19.311098 272.638128 -19.303083 272.632705 -19.299839 272.627971 -19.289408
            272.628226 -19.276293 272.633750 -19.270590 272.615109 -19.241810 272.614704 -19.221196
            272.618224 -19.215572 272.630809 -19.209945 272.633540 -19.198681 272.640711 -19.195292
            272.643028 -19.186751 272.651477 -19.182729 272.649821 -19.174859 272.656782 -19.169272
            272.658933 -19.161883 272.678012 -19.159481 272.689173 -19.176982 272.689395 -19.183512
            272.678006 -19.204016 272.671112 -19.206598 272.664854 -19.203523 272.662760 -19.211156
            272.654435 -19.214434 272.652969 -19.222085 272.656724 -19.242136 272.650071 -19.265092
            272.652868 -19.274296 272.660871 -19.249462 272.670041 -19.247807 272.675533 -19.254935
            272.673291 -19.273917 272.668710 -19.279245 272.671460 -19.287043 272.667507 -19.293933
            272.669261 -19.300601 272.663969 -19.307130 272.672626 -19.308954 272.675225 -19.316490
            272.657188 -19.349105 272.657638 -19.367455 272.662447 -19.372035 272.662232 -19.378566
            272.652479 -19.386871 272.645819 -19.387933 272.642279 -19.398277 272.629282 -19.402739
            272.621487 -19.398197 272.611782 -19.405716 272.603367 -19.404667 272.586162 -19.422703
            272.561792 -19.420008 272.555815 -19.413012 272.546500 -19.415611 272.537427 -19.424213
            272.533081 -19.441402
    Not (Polygon 272.511081 -19.487278 272.515300 -19.486595 272.517029 -19.471442
                 272.511714 -19.458837 272.506430 -19.459001 272.496401 -19.474322 272.504821 -19.484924)
    Not (Polygon 272.630446 -19.234210 272.637274 -19.248542 272.638942 -19.231476 272.630868 -19.226364)
)" | \
moc from stcs 16 - fits stcs_alma2.moc.fits --force-u64
moc view stcs_alma2.moc.fits stcs_alma2.moc.png auto 600
```



```
echo "Intersection ICRS ( ... )" | \
  moc from stcs 16 - fits stcs_alma2.moc.fits --force-u64
moc view stcs_alma2.moc.fits stcs_alma2.moc.png auto 600
```

STC-S querie in QATSS

- Data: **1.5 TB Gaia DR3** RCF file
- Simple query (first row) takes **100 ms**:

```
> time qat2s --select 'DR3Name,RAdeg,DEdeg,Source' \
>             --print-header                      \
>             --limit 1                           \
>             gaia_dr3.rcf all
DR3Name,RAdeg,DEdeg,Source
Gaia DR3 34361129088,45.00432028915398,0.021047763781174733

real    0m0,101s
user    0m0,096s
sys     0m0,005s
```

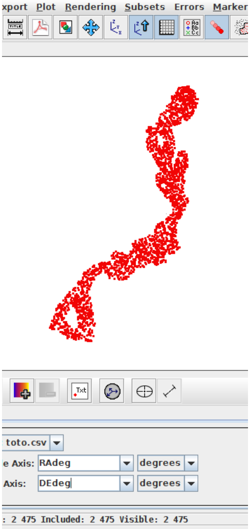
STC-S querie in QATSS

The ALMA STC-S query:

- takes **190 ms**
- 2475 rows returned

:

```
time qat2s --print-header --select 'DR3Name,RAdeg,DEdeg,Source' gaia_dr3.rcf pos DEFAULT DEFAULT stcs \
  "Intersection ICRS (
    Polygon 272.536719 -19.461249 272.542612 -19.476380 272.537389 -19.491509 272.540192 -19.499823
            272.535455 -19.505218 272.528024 -19.505216 272.523437 -19.500298 272.514082 -19.503376
            272.502271 -19.500966 272.488647 -19.490390  272.481932 -19.490913 272.476737 -19.486589
            272.487633 -19.455645 272.500386 -19.444996 272.503003 -19.437557 272.512303 -19.432436
            272.514132 -19.423973 272.522103 -19.421523 272.524511 -19.413250 272.541021 -19.400024
            272.566264 -19.397500 272.564202 -19.389111 272.569055 -19.383210 272.588186 -19.386539
            272.593376 -19.381832 272.596327 -19.370541 272.624911 -19.358915 272.629256 -19.347842
            272.642277 -19.341020 272.651322 -19.330424 272.653174 -19.325079 272.648903 -19.313708
            272.639616 -19.311098 272.638128 -19.303083 272.632705 -19.299839 272.627971 -19.289408
            272.628226 -19.276293 272.633750 -19.270590 272.615109 -19.241810 272.614704 -19.221196
            272.618224 -19.215572 272.630809 -19.209945 272.633540 -19.198681 272.640711 -19.195292
            272.643028 -19.186751 272.651477 -19.182729 272.649821 -19.174859 272.656782 -19.169272
            272.658933 -19.161883 272.678012 -19.159481 272.689173 -19.176982 272.689395 -19.183512
            272.678006 -19.204016 272.671112 -19.206598 272.664854 -19.203523 272.662760 -19.211156
            272.654435 -19.214434 272.652969 -19.222085 272.656724 -19.242136 272.650071 -19.265092
            272.652868 -19.274296 272.660871 -19.249462 272.670041 -19.247807 272.675533 -19.254935
            272.673291 -19.273917 272.668710 -19.279245 272.671460 -19.287043 272.667507 -19.293933
            272.669261 -19.300601 272.663969 -19.307130 272.672626 -19.308954 272.675225 -19.316490
            272.657188 -19.349105 272.657638 -19.367455 272.662447 -19.372035 272.662232 -19.378566
            272.652479 -19.386871 272.645819 -19.387933 272.642279 -19.398277 272.629282 -19.402739
            272.621487 -19.398197 272.611782 -19.405716 272.603367 -19.404667 272.586162 -19.422703
            272.561792 -19.420008 272.555815 -19.413012 272.546500 -19.415611 272.537427 -19.424213
            272.533081 -19.441402
    Not (Polygon 272.511081 -19.487278 272.515300 -19.486595 272.517029 -19.471442
                 272.511714 -19.458837 272.506430 -19.459001 272.496401 -19.474322 272.504821 -19.484924)
    Not (Polygon 272.630446 -19.234210 272.637274 -19.248542 272.638942 -19.231476 272.630868 -19.226364)
)" | wc -l
2476

real	0m0,188s
user	0m0,179s
sys	0m0,011s
```



toto.csv

e Axis: RAdeg degrees

Axis: DEdeg degrees

: 2 475 Included: 2 475 Visible: 2 475

Remarks on STC-S

- Discrepancies between the STC-S note EBNF and TAP 1.0 STC-S BNF
  - $<$ *coordsys* $>$ defined once in STC-S, many times in TAP
    - STC-S does not allow to mix various frames, TAP does
    - I tend to favor STC-S
  - $<$ *flavor* $>$: CART / CARTESIAN, SPHER / SPHERICAL
    - I tend to favor TAP
  - $<$ *frame* $>$: UNKNOWNFrame / UNKNOWNFRAME
    - I tend to favor STC-S (I like case sensitivity)
- STC-S DIFFERENCE is a MINUS, not a XOR
  - MINUS is simpler to emulate than XOR
    - *A MINUS B* = *A AND NOT(B)*
    - *A XOR B* = *(A OR B) AND NOT(A AND B)*
  - XOR more natural when expressing a boolean difference
    - *(A MINUS B).contains(p)* ⇔ *A.contans(p) && !B.contains(p)*
    - *(A XOR B).contains(p)* ⇔ *A.contains(p) != B.contains(p)*
  - Remove DIFFERENCE, replace by XOR (and add MINUS)?

□ STC-S vs MOC

- STC-S:
  - pro: **precise area**, **compact** for simple shapes (e.g. cone)
  - con: **complex** (various frames, fillfactor?, . . .), **no natural indexation**
- MOC:
  - pro: **easy to use** (same frame, . . .), **efficient** HEALPix based **indexation**
  - con: **approximated area**
  - may be more compact than STC-S (depending on the resolution) for complex regions
- STC-S regions and MOCs **are complementary**!
  - convert **STC-S to MOC**: **possible thanks to B-MOC**
    - (so far we miss frame conversions in our implementation)
  - convert **MOC to (original) STC-S**: **not possible**
  - keep STC-S for exact, compact, representation (depending on the complexity)
  - **rely on MOC (+post filtering) for efficient STC-S queries**