JSON and Python Implementation of Time-Frequency Radio Catalogues: TFCat

B. Cecconi[1], M. B. Taylor[2], X. Bonnin[1], A. Loh[1] and C. K. Louis[3]

[1]**LESIA, Observatoire de Paris-PSL, CNRS, Sorbonne Univ., Univ. Paris Cité, Meudon, France**

[2]**H H Wills Physics Laboratory, University of Bristol, Bristol, United Kingdom**

[3]**DIAS, Dublin, Ireland**

Corresponding author: baptiste.cecconi@observatoiredeparis.psl.eu

We present a catalogue format for events or features observed in the temporal-spectral domain (e.g., usually time-frequency, in the low frequency radio domain). This format is called TFCat (Time-Frequency catalogue), and is implemented in JSON, with a Python module to create and handle them. Based on GeoJSON, the corresponding python ecosystem for processing geometries is applicable.

In the low frequency domain, the electromagnetic signatures are tracers of energetic and unstable particle populations rather than atomic and molecular transitions. Such signatures are identified with their spectral-temporal shape in time varying spectrograms. Figure 1 shows a spectrogram including Solar radio bursts (related to energetic electron beams escaping from the Sun) and terrestrial auroral kilometric radiation (related to the magnetospheric activity), observed by the Cassini/RPWS instrument during its flyby of the Earth.The NDA was used as a ground support instrument for the Voyager mission, providing the science community with a monitoring of the Jovian Decametric emissions from 10 MHz to 40 MHz.

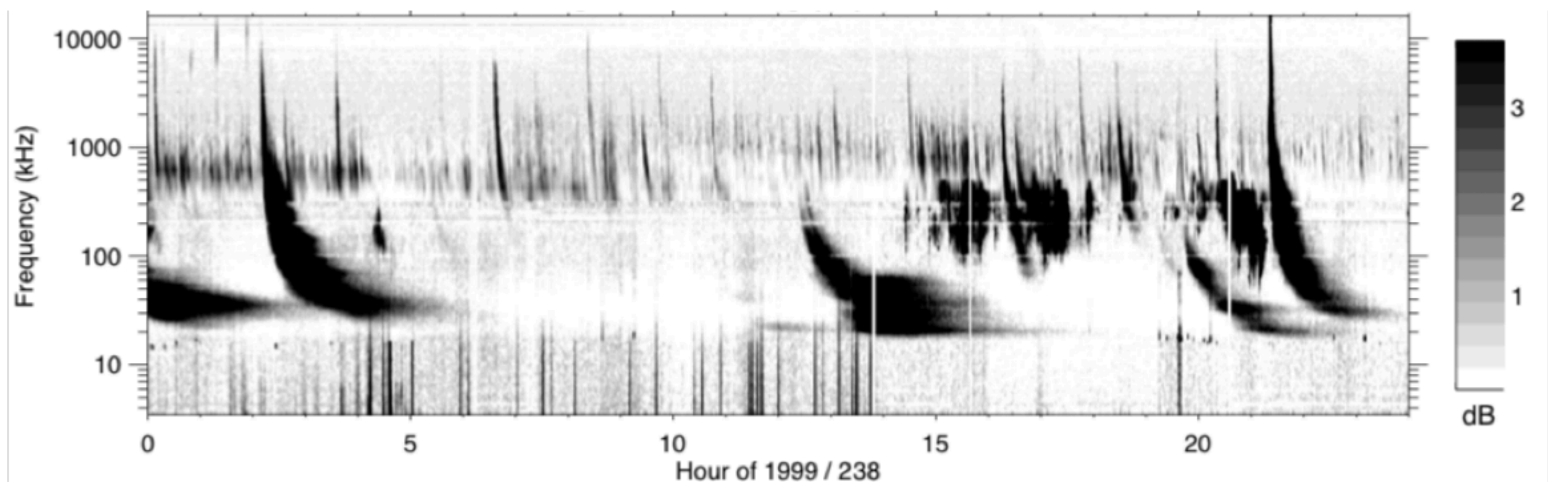


Figure 1: *Low frequency radio spectrogram (Wind/Waves) with solar and terrestrial natural radio emissions.*

A corpus of catalogues identifying and documenting such radio signatures have been published since the 70's, but there was no standard exchange format, leading to difficulties in sharing and reusing such catalogues.

TFCat (Time-Frequency Catalogue) is a data interchange format based on JSON. It defines several types of JSON objects and the manner in which they are combined to represent data about time-frequency features of a time spectrogram (a.k.a. dynamic spectrum), their properties, and their temporal and spectral extents. This implementation is inheriting from the GeoJSON file format [RFC7946].

In the past two years, several catalogues have been published using the TFCat format, and tools are now implementing interfaces on top of this specification, for labelling observations, displaying catalogues, or even using the catalogue shapes to select and process observational data.

Jupiter DAM emissions (Nançay/NDA)

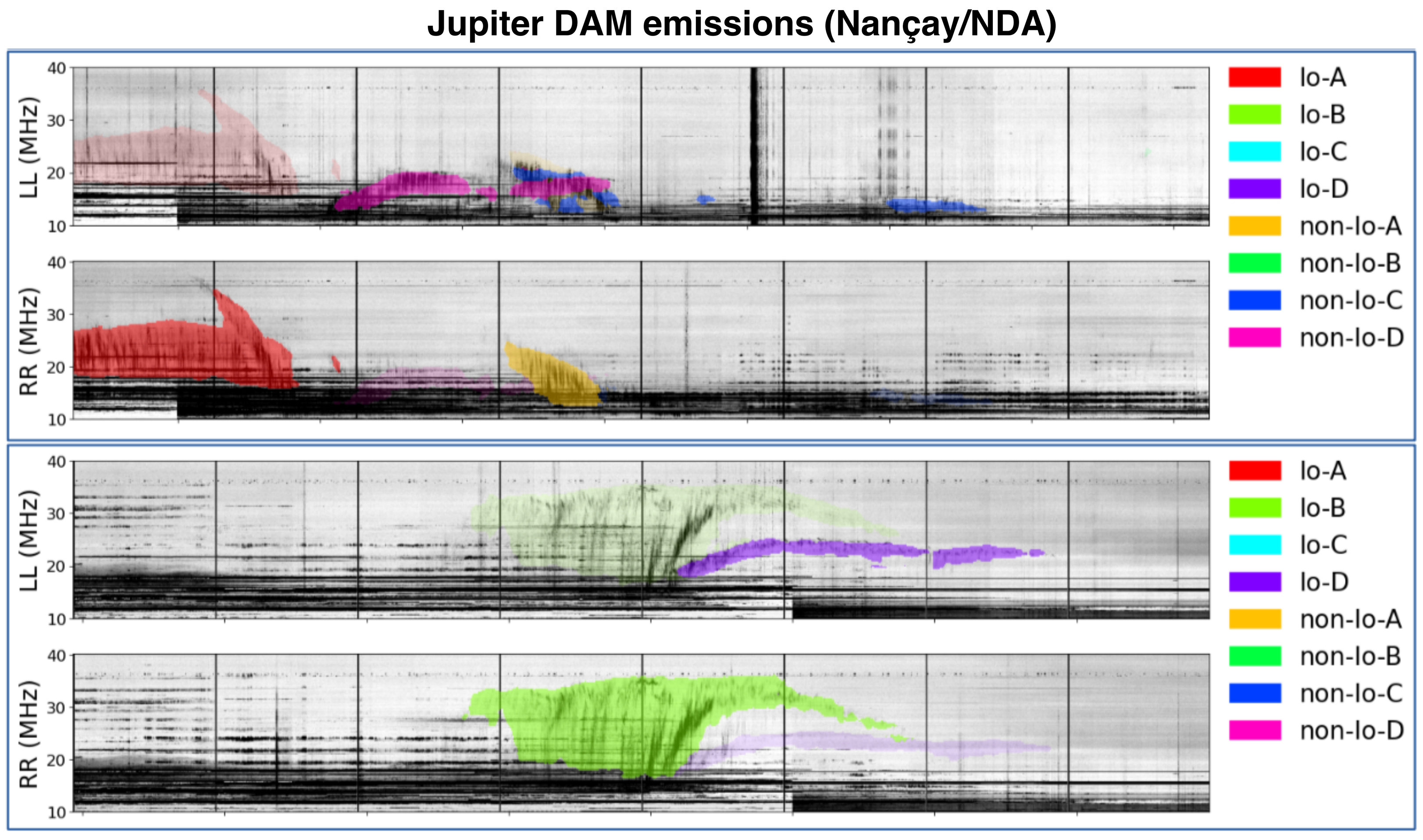
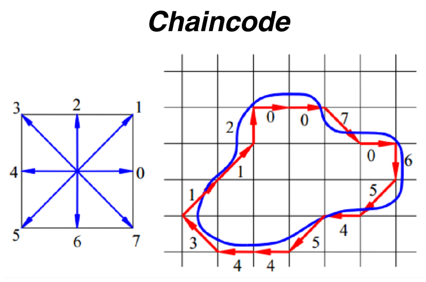


Figure 2. *Annotated NDA radio spectrogrammes using a machine-learning algorithm trained on a published catalogue (Marques et al. 2017). The upper pair of plots shows detected features within the published catalogue range, and on the bottom pair, new features detected on more recent observations.*

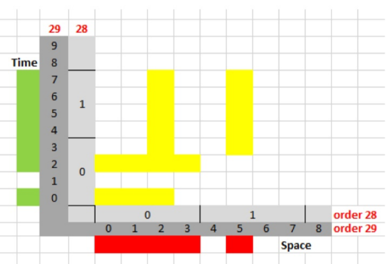
Events and Catalogues

- **IVOA**
  - VOTable => table (dates, spectral range…) + STC for geometry
  - VOEvent => content + location (centroid)
- **IHDEA**
  - HPEventList => table with "main time" of event and other parameters (arbitrary, but simple tabular data)
  - HPEventList serialisation = VOTable
- **Geometries** (set of coordinates, with type)
  - Images: *Chaincode* (starting point + step-by-steps directions in pixel coordinates)
  - *VO-STC* (spatial regions, but no spectro-temporal coordinates)
  - *VO-(ST)MOC* => Time-Spectral MOC ?
  - *OGC-GeoJSON* => geo-referenced spatial features

Chaincode



STMOC



GeoJSON

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

TFCat Model

- A **TF-geometry** is a geometry (point, line-string, polygon, multi-points, multi-line-strings, multi-polygon), with coordinates in time and frequency (or wavelength, or wavenumber, or energy).
- The **catalogue contains a list of features**, with a TF-geometry (or a set of geometries), and parameters.
- The **catalogue contains a definition of the features additional properties**, with data type, description, unit, UCD, etc.
- The **catalogue contains a set of global properties**, at catalogue level, with data type, description, unit, UCD, etc.
- The **catalogue contains a description of the CRS** (coordinate reference system) defining the temporal and spectral axes, as well as the frame reference position.

TFCat current implementation

- **Based on GeoJSON**, a catalogue model & format for geo-referenced shapes (on Earth, with lat-long coordinates).
- Using JSON for transport
- Implementation in Python reusing GeoJSON python library
- **With astronomy standards** when applicable:
  - Using IVOA vocabularies (UCD, VOUnits…) as much as possible
  - *STC1*: no support for *complex spectro-temporal geometries** / STC2: the same
  - *VOTable*: no support for *complex spectro-temporal geometries**
  - Use of TAP for distributing and searching?
  - Geometry in TAP and ADQL?
  - *Datalink* to link catalogue of feature to catalogue of observation?
  - Export to *Astropy* table: ok, possible for complex geometries (as objects)
- NB: *complex spectro-temporal geometries = something else than a point or a bounding box*

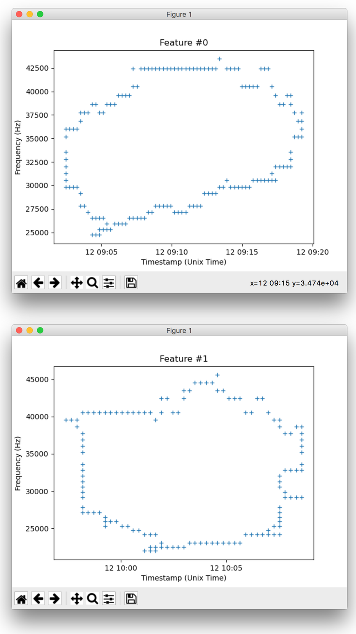
TFCat Model+Library

- TFCat Model Specification:
  https://gitlab.obspm.fr/maser/catalogues/catalogue-format

  How to cite:
  Cecconi, B., X. Bonnin, A. Loh. M. B. Taylor (**2022**), *JSON Implementation of Time-Frequency Radio Catalogues : TFCat* (Version 1.0). PADC. https://doi.org/10.25935/6068-8528

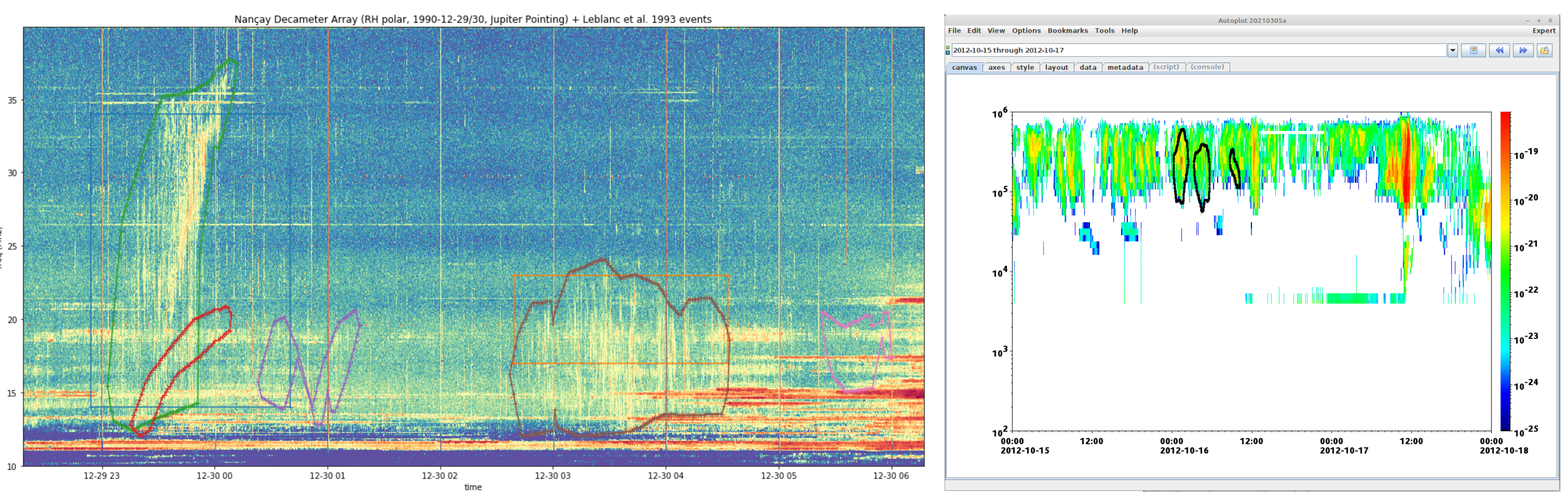
- TFCat python library (version 0.4.0):
  https://gitlab.obspm.fr/maser/catalogues/tfcat

```
(tfcat) macbookbc:taubenschuss_2021 baptiste$ python
Python 3.7.10 | packaged by conda-forge | (default, Feb 19 2021,
15:59:12)
[Clang 11.0.1 ] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> from tfcat import TFCat
>>> tf_file = 'cassini_faraday_patches_2006.json'
>>> cat = TFCat.from_file(tf_file)
>>> cat.crs
{"properties": {"name": "Time-Frequency", "ref_position_id":
"cassini-orbiter", "spectral_coords": {"type": "frequency", "unit":
"Hz"}, "time_coords_id": "unix"}, "type": "local"}
>>> cat.plot_feature(0)
>>> cat.plot_feature(1)
>>> cat.crs.time_converter(cat.feature(0).tmin).isot
'2006-09-12T09:02:26.750'
>>> cat.crs.spectral_converter(cat.feature(0).fmax)
<Quantity 43469. Hz>
```



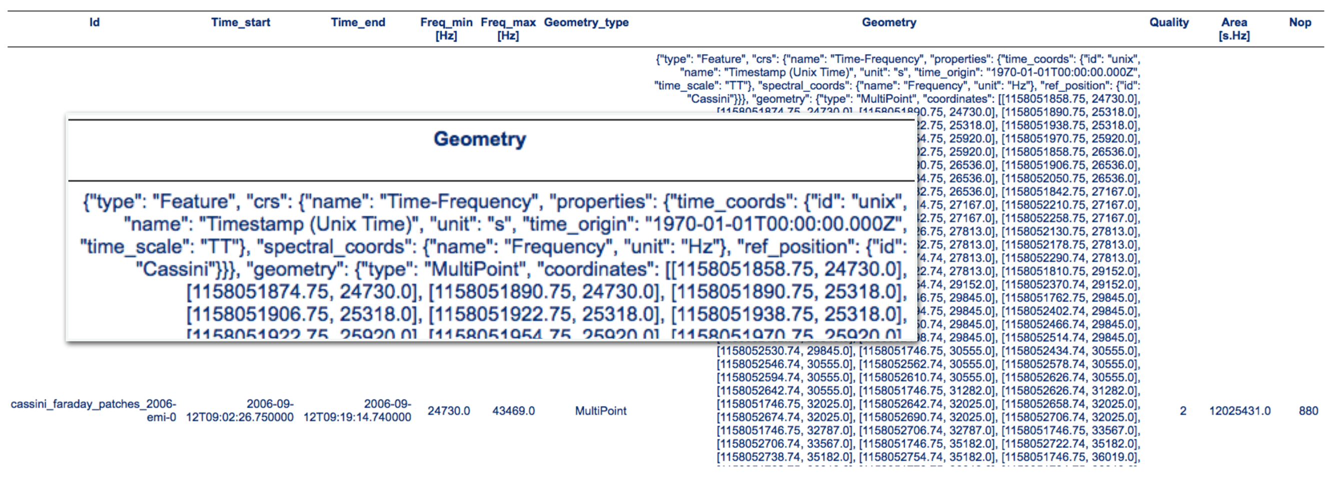
Example: TFCat + Das2 + (python or Autoplot)

- Data loaded from Nançay Das2* server (das2py), catalogue from 2 catalogues (Leblanc 1993 & Marques 2017)
  das2 = data-streaming API for time-series, with server-side resampling



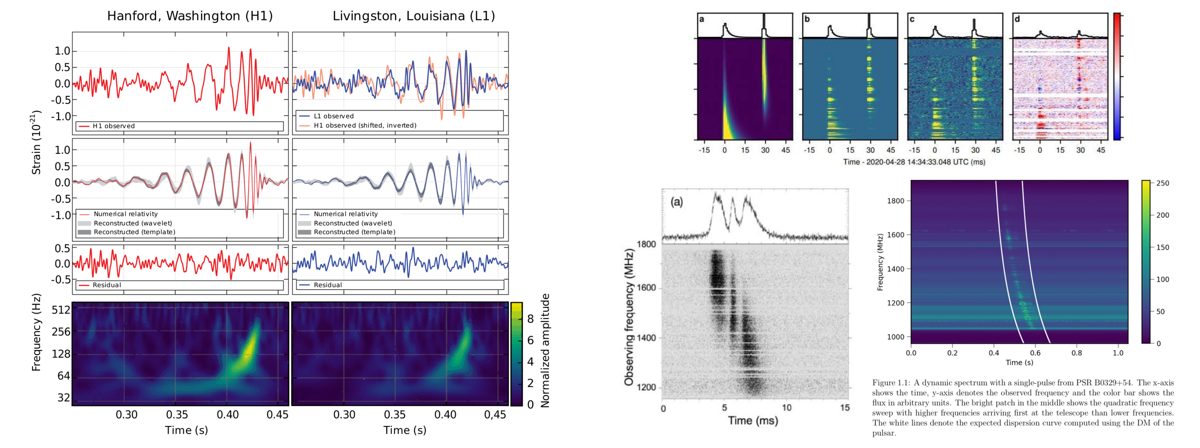
TFCat + TAP exploration

- 1 Feature per row, Geometry as JSON string,
  Feature mapped properties into columns
- JSON-string allows to keep all information (geometry type, CRS, coordinates) of Feature for future use.



Other use cases ?

- Astronomy transients: Pulsars, FRB, GW, CTA…
  for Machine Learning training sets?



The Europlanet 2024 Research Infrastructure project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871149. This work used the EGI Infrastructure with the dedicated support of IN2P3-IRES and CESNET-MCC. Additional funding was provided in France by the Centre National d'\'Etudes Spatiales (CNES), and Action Spécifique Observatoire Virtuel (ASOV).