The "SPectrogram Analysis and Cataloguing Environment" (SPACE) Labelling Tool

C. K. Louis [1,*], C. M. Jackman [1], S. W. Mangham [2], K. D. Smith [1], E O'Dwyer [1], A. Empey [1], B. Cecconi [3], P. Zarka [3], S. Maloney [1]

[1] *School of Cosmic Physics, DIAS Dunsink Observatory, Dublin Institute for Advanced Studies, Dublin 15, Ireland*
[2] *Department of Electronics & Computer Science, University of Southampton*
[3] *LESIA, Observatoire de Paris, CNRS, PSL Research University, Meudon, France*

Correspondence*:
Corresponding Author
corentin.louis@dias.ie

ABSTRACT

The SPectrogram Analysis and Cataloguing Environment (SPACE) tool is an interactive python tool designed to label radio emission features of interest in a time-frequency map (called "dynamic spectrum"). The program uses Matplotlib's Polygon Selector widget to allow a user to select and edit an undefined number of vertices on top of the dynamic spectrum before closing the shape (polygon). Multiple polygons may be drawn on any spectrum, and the feature name along with the coordinates for each polygon vertex are saved into a ".json" file as per the "Time-Frequency Catalogue" (TFCat) format along with other data such as the feature id, observer name, and data units. This paper describes the first official stable release (version 2.0) of the tool.

For full guidelines regarding your manuscript please refer to Author Guidelines.

Keywords: keyword, keyword, keyword, keyword, keyword, keyword, keyword

1 INTRODUCTION

Non-thermal planetary radio emissions are produced by out-of-equilibrium populations of charged particles in planetary magnetospheres, and are observed at almost all strongly magnetized planets in our solar system: the Earth, Jupiter, Saturn, Uranus and Neptune. The radio emissions can be divided into different classes, such as plasma waves, electromagnetic radio waves or electrostatic radio waves. It is highly desirable to select these distinct classes, which can often have characteristic frequency ranges, morphologies, or polarizations. Once catalogues of different emission types have been built up, that can enable large statistical studies unveiling both the average and extreme behaviour of planetary radio emissions.

There have been many long-running planetary spacecraft which have returned huge volumes of radio data and we have only scratched the surface of its analysis. For example, the Cassini mission at Saturn spent 13 years studying the kronian system, revealing several components to its radio spectrum (Lamy et al., 2008; Ye et al., 2011; Lamy, 2017; Taubenschuss et al., 2011). Furthermore, the Wind spacecraft has spent almost two decades observing terrestrial (and solar) radio emissions from a range of vantage points near Earth (Waters et al., 2021; Fogg et al., 2022; Bonnin et al., 2008).

Significant efforts have been made in recent years to classify radio emissions from Jupiter, where the non-thermal radio emission is composed of half a dozen components (Louis et al., 2021c). These components overlap themselves in time and frequency, making automatic detection non-trivial. Therefore, manually cataloguing them is mandatory to be able to study them independently. Previous catalogues have been using square boxes to define the time and frequency intervals containing the radio signal (such as Leblanc, 2020a,e,b,c,d), but to be able to automatically disentangle the emissions when using the catalogue, it is then needed to construct catalogues with polygon vertices and with distinct labels. This was done by a few authors, primarily using tools built in IDL to construct such catalogues of Jupiter radio emissions Marques et al. (2017); Zarka et al. (2021). Once catalogues are built they can comprise training sets which form the basis of supervised machine learning approaches to classify larger samples of unseen data.

Here we present a Python user interface tool to allow the drawing of polygons around features in dynamic spectra. Section 2 describes the package, and an example of the use of this package. Section 3 summarises the version history of the code. Section 4 gives examples of the application of the catalogues produced by the tool. Section 5 presents some future avenues to continue to improve the tool.

2 THE SPACE PACKAGE

The SPectrogram Analysis and Cataloguing Environment (SPACE) tool is an interactive python tool designed to label radio emission features of interest in a time-frequency map (called a “dynamic spectrum”). The program enables users to create and edit the vertices of a polygon on the dynamic spectrum plot, before naming and saving it as a ’feature’ in a catalogue for future analysis.

2.1 Installation

The code is available on an open-source GitHub repository (Louis et al., 2022), with full installation instructions present on the repository page, and packed with the tool. It can easily be downloaded and installed using `git` and `pip`, with all prerequisites included in the provided `requirements.txt` file.

2.2 Usage

Once installed using `pip`, the space labelling tool is available as a system-wide command `spacelabel`. It can then be used to view and label spacecraft observational data, by providing an input file in **HDF5** or **NASA CDF** format (see section 2.2.3 for specifics), and a time window to view. An example use is shown in Figure 1.

Users may select any number of the measurement types present in the file (e.g. polarisation, flux and/or power), and view them all tiled on the screen. For example, Figure 1a displays Cassini Flux and Polarization radio data, while Figure 1b only display Juno Flux radio data. Users can then click to select polygonal regions of the observation to label as features (top panel of Figure 1a). To close the polygon, users should click on the first drawn vertex of the polygon. Once done, a window pops up and ask to name the drawn feature appropriately (see top panel of Figure 1a). Features labelled in one view (e.g. intensity) appear simultaneously on the other views once they have been named (see bottom panel of Figure 1a), allowing users to easily see how a feature presents in multiple measurement types.

Once a region has been labelled, a user can pan their viewing window back and forth through the time range within the dataset by clicking on the `Prev` or `Next` buttons (see Figure 1), with an overlap applied between each view in order to facilitate labelling features that lie on the edge of a window. Once finished, the labelled regions can be saved as a **TFCat** (Time-Frequency Catalogue) formatted *JSON*[1] file (Cecconi et al., 2022, this issue) by clicking on the `Save` button, and used later. If a user re-opens the same data file, or another data file with the same naming structure (e.g. `observations_20180601_v02.cdf` and `observations_20180602_v02.cdf`) saved features from previous sessions will be pre-loaded (see Figure 1).

Full usage documentation is available on the GitHub repository for the code (**?**).

2.2.1 Procedure

When the code first opens a datafile, it compares the columns within to a selection of pre-made (and user-creatable) 'configuration' files for each type of input file (e.g. **CDF**, **HDF5**). Each describes a file in terms of the column names within it, and provides metadata for use in the tool - units and display names, and scaling factors that can be applied to change data stored in one unit into data viewed in another. If the data are in an unspecified format, the code will prompt the user to create a configuration file (see section 2.2.3). Alternatively, if their data fit multiple configuration files, they will be prompted to select which they want to use (see section 2.2.2).

Once a configuration has been determined, the code parses the observations and may re-bin the time into a coarser resolution in order to improve performance, taking the average of measurements in the new larger bins. It can also rescale the frequency data into evenly-spaced logarithmic bins between the minimum and maximum bins in the original data, as some data files have non-monotonic bin structures. When altering the frequency bins, measurements are logarithmically interpolated between the readings on the previous scale. Default adjustments can be defined in configuration files for file types, and may be over-ridden by command-line arguments to the tool. The parsed and adjusted data are then re-saved as a compressed **HDF5** file, reducing both the size of the data and time to access.

The pre-processed data are then displayed to the screen using MatPlotLib (Hunter, 2007), in a window the size of the user's initial time range. The dynamic range of the data is constrained to improve visibility of features; displaying, by default, the 5th-95th quantiles of the signal for each measurement (to prevent anomalously low or high values overly-compressing the ranges of interest, reducing the ability to discern features). GUI buttons allow users to pan between time windows of equivalent size - with each window will overlap the previous window by 25% in the direction of travel. Features can then be defined by drawing polygons on the time window using the MatPlotLib Polygon Selector widget.

2.2.2 Options

The user must specify the path to the file they want to visualise (or 'first' file in a collection, e.g. of **CDF** files), along with the start and end dates of their initial viewing window, in **ISO** *year-month-day hour:minute:second* format (e.g. `2018-06-12 18:00:01`). However, there are further options available:

- `-f FREQUENCY_BINS`: Rescales the data to this many evenly-spaced logarithmic bins. Overrides any default set in the configuration files.
- `-t MINIMUM_TIME_BIN`: Rebins the data to time bins of this size, if it is currently more finely binned. The bin size need to be given in second
- `-s SPACECRAFT`: Specifies the name of the spacecraft configuration file to use, if multiple describe the datafile the user has provided.

[1] `https://www.json.org/`

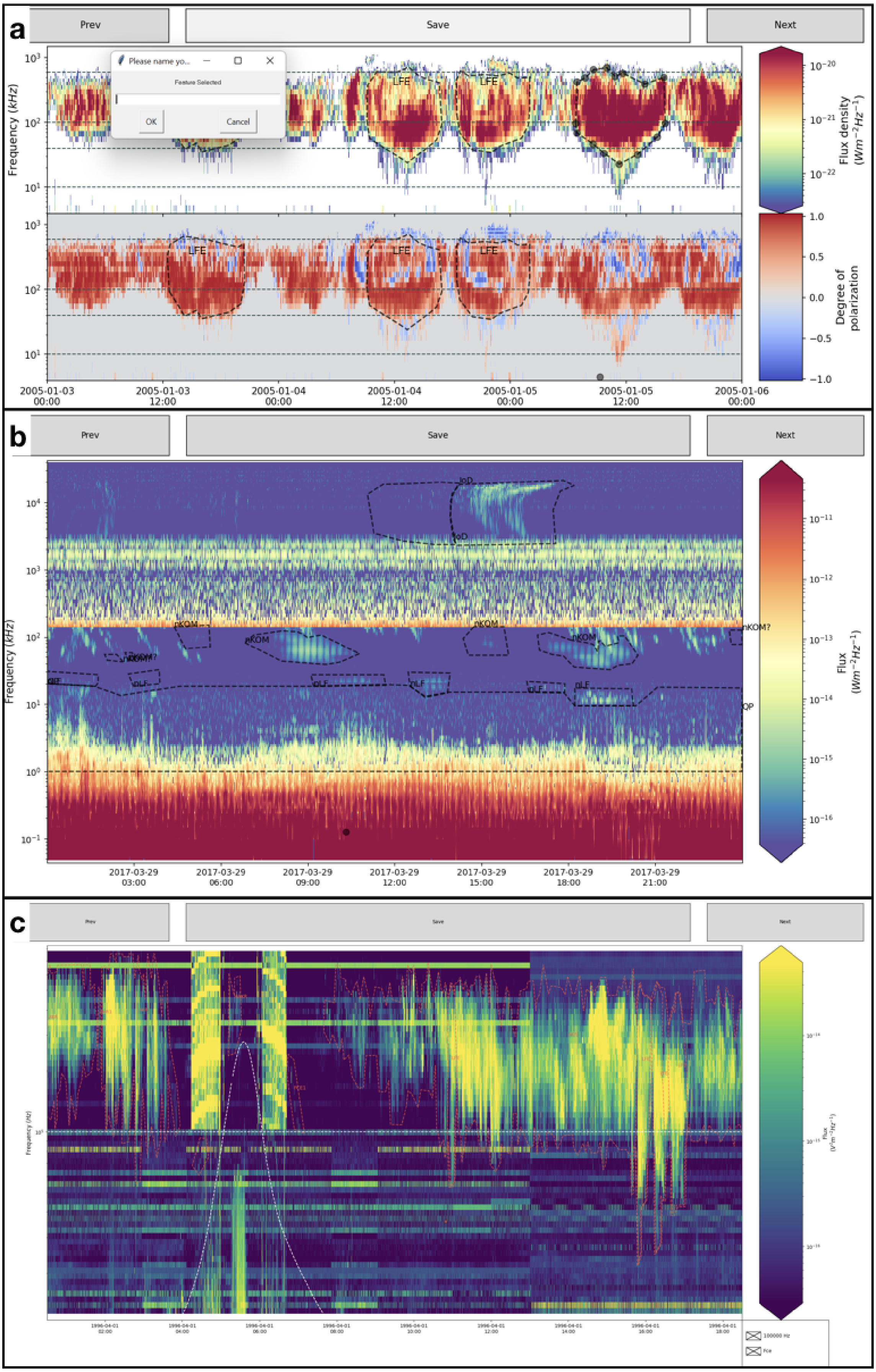

Figure 1: Caption on next page

Figure 1. Examples of plots from the SPACE labelling Tool.
Panel (a) displays Cassini/RPWS (Radio and Plasma Waves Science Gurnett et al., 2004) data (Lamy et al., 2008, 2009). The two panels show Intensity and Polarization data, respectively. At the top right of the top panel one can see a polygon that has just been drawn, with the window for naming the feature appearing at the top left of the graphics window. Other features have already been labelled, and appear in both intensity and polarisation views, with their names overlaid.
The data displays in panel (b) are the estimated flux density (Louis et al., 2021a,c) from by Juno/Waves measurements (Kurth et al., 2017), with the Louis et al. (2021b) catalogue overlaid.
Panel (c) displays observations of Polar/PWI instrument (Gurnett et al., 1995). The horizontal dashed-white line shows an example of the use of the `-horizontal_line` option. The variable dashed-white line show that the tool is also able to read a 1D table from the CDF file (provided that this has been specified in the configuration file)

- `-frac_dyn_range FRAC_MIN FRAC_MAX`: Defines the dynamic range of the colour bar in the visualisation, as a fraction of the distribution of values in the data file. This must be numbers between 0 and 1. Default values are 0.05 and 0.95 (the 5th-95th quantiles of the displayed signal)
- `-cmap CMAP`: The name of the color map that will be used for the intensity plot (by default: viridis)
- `-g VALUES_1 ... VALUES_N`: Draws horizontal line(s) on the visualisation at these specified frequencies to aid in interpretation of the plot. Values must be in the same units as the data. Lines can be toggled using check boxes.
- `--not_verbose`: If `not_verbose` is called, the debug log will not be printed. By default: verbose mode

2.2.3 Input Formats

The code is designed to cope with input files in a variety of file formats and column formats by use of configuration files, several of which are pre-provided. **HDF5** input files require at least three datasets, corresponding to **observation time** (floats, in MJD), **frequency range** (floats, in any arbitrary unit) and at least one measurement, stored with frequency as the rows and observation as the columns. The names and units for each measurement (in LaTeX form) must be provided in a configuration file, in easily-editable **JSON** format. The appropriate configuration files are automatically-selected by the code from those available - making it easy to work with **HDF5** files from a variety of collaborators with arbitrary naming schemes.

CDF files in NASA format are more structured, and can be read in either singly or as a collection, combining all files in the directory matching the naming scheme `[...]_YYYMMDD_[...].cdf` into a single pre-processed data file. As with the **HDF5** files, **CDF** files must contain a frequency attribute (floats, in any arbitrary unit) and a time attribute (either in `TT_2000` or `CDF_EPOCH` format, which is parsed using **Astropy**, Price-Whelan et al., 2018) and at least one measurement, stored with frequency as the rows and observation as the columns.

The code can easily be expanded to ingest other file formats (see section 2.3.2).

2.3 Structure

2.3.1 'Model-View-Presenter' Architecture

The code is designed using a standard object-oriented 'Model-View-Presenter' architecture, with strong separation between the data input and management, and the visualisation. This allows for easy development of both new input file-types (see section 2.3.2) and pre-processing options, and alternative GUI front-ends and settings (see section 5 for suggested development building on top of this flexibility). A generic 'Presenter' controls the logic of the program, and feeds data from the data models to the selected GUI view, and requests from the GUI for changes to the data models. Either the 'View' or 'Model' can be easily interchanged as long as they conform to the API expected by the 'Presenter'.

Full development documentation is available on the GitHub repository for the code (**?**).

2.3.2 Addition of new Input Formats

New input formats can be easily added by extending the base `DataSet` class included in the code. A developer only needs to define the routines for inputting the data from file; the code will then handle pre-processing and data access.

3 HISTORY OF THE CODE

The first version of the labelling code was developed in IDL by P. Zarka. It allowed users to read data from an IDL saveset (`sav` format), draw polygons around features of interest and label them. However, this IDL version had to be adapted to each new dataset. This code has been used to build many catalogues based on different observers (such as the Nançay Decameter Array (NDA) ground-based radio telescope (Marques et al., 2017), or the Cassini (Zarka et al., 2021) or Juno (Louis et al., 2021c,b) spacecraft).

The second version of the labelling tool was written in Python and was the first to be officially released (Empey et al., 2021). This version allowed to automatically read any dataset in `sav` or `cdf` format, based on the information requested from users from the terminal. The other main improvements compared to the previous version were the number of vertices in the polygons (unlimited) and the possibility to modify the vertices position during the polygon drawing (using the `Matplotlib's Polygon Selector widget`), as well as the production of the catalogue directly in TFCat format.

The current version (Louis et al., 2022) brings a large number of improvements, both in terms of architecture, usability and ergonomics, which are described in the previous sections.

4 APPLICATIONS

Once a catalogue has been produced, it can also be displayed using the SPACE labelling Tool (see Figure 1 or the Autoplot Software (Faden et al., 2010). For an example, the reader is invited to visit the web page `https://doi.org/10.25935/nhb2-wy29` where an autoplot template file is given to display the Juno data (Louis et al., 2021a) and the Louis et al. (2021b) catalogue overlaid in autoplot. See Cecconi et al. (2022, this issue) for more information about the display of a Catalogue using Autoplot. The catalogue can be used to study the different components of the radio emission spectrum, e.g. as done by Louis et al. (2021c), where the data can then be automatically selected using the catalogue via a mask or an inverse-mask. In the case presented in Figure 1, not every type of emission is labelled, but in each frequency range (kilometric or hectometric) only one radio component remains. We can then study the components one by one (e.g. their latitudinal distribution, as in Louis et al. (2021c), their distribution as a function of observer's or Sun's longitude, as in Zarka et al. (2021),or their distribution versus observer's longitude and satellite (Io) phase as in Marques et al. (2017)).

With the SPACE labelling Tool, we are also providing some useful routines to use the catalogue[2].

[2] `https://github.com/elodwyer1/Functions-for-SPACE-Labelling-Tool`

These catalogues can also then be used to train machine learning algorithm to detect automatically the radio emissions in past (Cassini, NDA) or future observation (such as Juno, JUICE, NDA).

5 LIMITATIONS & FUTURE WORK

The code is ready for distribution and use, but has some technical limitations. Potential works to address those limitations, and avenues of future development, are:

- **Performance:** The MatPlotLib-based front-end can struggle when provided with especially high resolutions of data, or over large time windows. Rebinning features exist to mitigate this, but ideally the front-end would be re-implemented in a more performant framework (e.g. Plotly Inc. (2015)).
- **Scalability:** The code loads all the data provided into memory at launch, limiting its applicability for large datasets. Whilst this can be mitigated by the feature to allow appending to **TFCat** files created by data files sharing filename formats, a 'deferred load' approach would be better. This would be best accomplished using the Dask and XArray libraries (Dask Development Team, 2016; Hoyer and Hamman, 2017).
- **Configurations:** The code depends heavily on pre-written configuration files, and can prompt users to create missing ones - but does not yet contain a 'wizard' or automatic walkthrough to aid users in creating them.
- **Catalogue integration:** The modular format of the code would make it possible to create 'dataset' types that access and download data directly from online catalogues, maintaining local caches.

CONFLICT OF INTEREST STATEMENT

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

AUTHOR CONTRIBUTIONS

C. Louis and C. Jackman led the development of the SPACE Labelling Tool. C. Louis wrote the paper and worked on the development of the code. P. Zarka developed the first IDL version of the code. A. Empey developed the first python version of the code. S. Maloney worked on the development of the second version of the code. S. W. Mangham developed the current version of the code. E. O'Dwyer tested the different versions of the code and give inputs to the developers of the code. K. Smith added features to the latest version of the code. B. Cecconi led the TFCat format of the catalogues. All the co-authors have contributed to the writing of the paper.

FUNDING

C.K. Louis', C.M. Jackman's, E. O'Dwyer's and A. Empey's work at the Dublin Institute for Advanced Studies was funded by Science Foundation Ireland Grant 18/FRL/6199. S. Mangham's work at the University of Southampton Research Software Group was funded by a grant via the Alan Turing Institute. K.D. Smith's work at the Dublin Institute for Advanced Studies was funded by a 2022 SCOSTEP/PRESTO Grant.

DATA AVAILABILITY STATEMENT

The code of the SPACE labelling Tool is open-source and freely available on github (Louis et al., 2022). The Cassini/RPWS dataset displayed in Figure 1a, produced by Lamy et al. (2008) is available at `https://doi.org/10.25935/ZKXB-6C84` (Lamy et al., 2009). The Juno/Waves dataset displayed in Figure 1b, produced by Louis et al. (2021c), is accessible at `https://doi.org/10.25935/6jg4-mk86` (Louis et al., 2021a), and the catalogue can be download at `https://doi.org/10.25935/nhb2-wy29` (Louis et al., 2021b) . The Polar/PWI dataset displayed in Figure 1c is accessible through the CDAWeb at `https://cdaweb.gsfc.nasa.gov/pub/data/polar/pwi/`.

REFERENCES

Bonnin, X., Hoang, S., and Maksimovic, M. (2008). The directivity of solar type III bursts at hectometer and kilometer wavelengths: Wind-Ulysses observations. Astronomy and Astrophysic 489, 419–427. doi:10.1051/0004-6361:200809777

Dask Development Team (2016). Dask: Library for dynamic task scheduling

[Dataset] Empey, A., Louis, C. K., and Jackman, C. M. (2021). SPACE Labelling Tool (version 1.1.0) [Code]. doi:10.5281/zenodo.5636922

Faden, J. B., Weigel, R. S., Merka, J., and W., F. R. H. (2010). Autoplot: a browser for scientific data on the web. Earth. Sci. Inform. 3, 41–49. doi:10.1007/s12145-010-0049-0

Fogg, A. R., Jackman, C. M., Waters, J. E., Bonnin, X., Lamy, L., Cecconi, B., et al. (2022). Wind/WAVES Observations of Auroral Kilometric Radiation: Automated Burst Detection and Terrestrial Solar Wind - Magnetosphere Coupling Effects. Journal of Geophysical Research (Space Physics) 127, e30209. doi:10.1029/2021JA030209

Gurnett, D. A., Kurth, W. S., Kirchner, D. L., Hospodarsky, G. B., Averkamp, T. F., Zarka, P., et al. (2004). The Cassini Radio and Plasma Wave Investigation. Space Science Reviews 114, 395–463. doi:10.1007/s11214-004-1434-0

Gurnett, D. A., Persoon, A. M., Randall, R. F., Odem, D. L., Remington, S. L., Averkamp, T. F., et al. (1995). The Polar Plasma Wave Instrument. Space Science Reviews 71, 597–622. doi:10.1007/BF00751343

Hoyer, S. and Hamman, J. (2017). xarray: N-D labeled arrays and datasets in Python. Journal of Open Research Software 5. doi:10.5334/jors.148

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. Computing in Science & Engineering 9, 90–95. doi:10.1109/MCSE.2007.55

[Dataset] Inc., P. T. (2015). Collaborative data science

Kurth, W. S., Hospodarsky, G. B., Kirchner, D. L., Mokrzycki, B. T., Averkamp, T. F., Robison, W. T., et al. (2017). The Juno Waves Investigation. Space Science Reviews 213, 347–392. doi:10.1007/s11214-017-0396-y

[Dataset] Lamy, L., , Cecconi, B., and Zarka, P. (2009). Cassini/RPWS/HFR LESIA/Kronos SKR Data Collection (Version 1.0) [Data set]. doi:10.25935/ZKXB-6C84

Lamy, L. (2017). The Saturnian kilometric radiation before the Cassini Grand Finale. In Planetary Radio Emissions VIII, eds. G. Fischer, G. Mann, M. Panchenko, and P. Zarka. 171–190. doi:10.1553/PRE8s171

Lamy, L., Zarka, P., Cecconi, B., Prangé, R., Kurth, W. S., and Gurnett, D. A. (2008). Saturn kilometric radiation: Average and statistical properties. Journal of Geophysical Research (Space Physics) 113, A07201. doi:10.1029/2007JA012900

[Dataset] Leblanc, Y. e. a. (2020a). A catalogue of Jovian decametric radio observations from January 1978 to December 1979 (Digitised version) [Data set]. doi:10.25935/GXZF-ZT33

[Dataset] Leblanc, Y. e. a. (2020b). A catalogue of Jovian decametric radio observations from January 1982 to December 1984 (Digitised version) [Data set]. doi:10.25935/403B-VA51

[Dataset] Leblanc, Y. e. a. (2020c). A catalogue of Jovian decametric radio observations from January 1985 to December 1987 (Digitised version) [Data set]. doi:10.25935/gh59-py87

[Dataset] Leblanc, Y. e. a. (2020d). A catalogue of Jovian decametric radio observations from January 1988 to December 1990 (Digitised version) [Data set]. doi:10.25935/cmqb-jj10

[Dataset] Leblanc, Y. e. a. (2020e). A catalogue of Jovian radio observations from January 1980 to December 1981 (Digitised version) [Data set]. doi:10.25935/RAS0-ER93

[Dataset] Louis, C. K., Jackman, C. M., Mangham, S. W., Smith, K. D., O'Dwyer, E., Empey, A., et al. (2022). SPACE Labelling Tool Version 2.0.0 (v2.0.0) [Code]. doi:10.5281/zenodo.6886528

[Dataset] Louis, C. K., Zarka, P., and Cecconi, B. (2021a). Juno/Waves estimated flux density Collection (Version 1.0). doi:10.25935/6jg4-mk86

[Dataset] Louis, C. K., Zarka, P., Cecconi, B., and Kurth, W. S. (2021b). Catalogue of Jupiter radio emissions identified in the Juno/Waves observations (Version 1.0). doi:10.25935/nhb2-wy29

Louis, C. K., Zarka, P., Dabidin, K., Lampson, P. A., Magalhães, F. P., Boudouma, A., et al. (2021c). Latitudinal Beaming of Jupiter's Radio Emissions From Juno/Waves Flux Density Measurements. Journal of Geophysical Research (Space Physics) 126, e29435. doi:10.1029/2021JA029435

Marques, M. S., Zarka, P., Echer, E., Ryabov, V. B., Alves, M. V., Denis, L., et al. (2017). Statistical analysis of 26 yr of observations of decametric radio emissions from Jupiter. Astronomy & Astrophysics 604, A17. doi:10.1051/0004-6361/201630025

Price-Whelan, A. M., Sipőcz, B., Günther, H., Lim, P., Crawford, S., Conseil, S., et al. (2018). The astropy project: Building an open-science project and status of the v2. 0 core package. The Astronomical Journal 156, 123

Taubenschuss, U., Leisner, J. S., Fischer, G., Gurnett, D. A., and Nemec, F. (2011). Saturnian Low-Frequency Drifting Radio Bursts: Statistical Properties and Polarization. In Planetary, Solar and Heliospheric Radio Emissions (PRE VII), eds. H. O. Rucker, W. S. Kurth, P. Louarn, and G. Fischer. 115–123

Waters, J. E., Jackman, C. M., Lamy, L., Cecconi, B., Whiter, D. K., Bonnin, X., et al. (2021). Empirical Selection of Auroral Kilometric Radiation During a Multipoint Remote Observation With Wind and Cassini. Journal of Geophysical Research (Space Physics) 126, e29425. doi:10.1029/2021JA029425

Ye, S. Y., Fischer, G., Menietti, J. D., Wang, Z., Gurnett, D. A., and Kurth, W. S. (2011). An Overview of Saturn Narrowband Radio Emissions Observed by Cassini RPWS. In Planetary, Solar and Heliospheric Radio Emissions (PRE VII), eds. H. O. Rucker, W. S. Kurth, P. Louarn, and G. Fischer. 99–113

Zarka, P., Magalhães, F. P., Marques, M. S., Louis, C. K., Echer, E., Lamy, L., et al. (2021). Jupiter's Auroral Radio Emissions Observed by Cassini: Rotational Versus Solar Wind Control, and Components Identification. Journal of Geophysical Research (Space Physics) 126, e29780. doi:10.1029/2021JA029780

FIGURE CAPTIONS