The Journal of Open Source Software

ndcube: Manipulating N-dimensional Astronomical Data in Python

Daniel F. Ryan [1,2¶], **Stuart Mumford** [*3], **Yash Sharma**[4], **Ankit Kumar Baruah**[5], **Adwait Bhope** [6], **Nabil Freij** [7,8], **Laura A. Hayes** [13], **Will T. Barnes** [12,1], **Baptiste Pellorce**[9,10], **Richard O'Steen** [11], **Derek Homeier** [3], **J. Marcus Hughes** [14], **David Stansby** [15], **Albert Y. Shih** [12], and **Matthew J. West** [14]

1 University of Applied Sciences Northwest Switzerland, Switzerland **2** American University, USA **3** Aperio Software Ltd, UK **4** Meta Platforms Inc., UK **5** Workato Gmbh, Germany **6** Uptycs India Pvt. Ltd., India **7** Lockheed Martin Solar and Astrophysics Laboratory, USA **8** Bay Area Environmental Research Institute, USA **9** Claude Bernard Lyon 1 University, France **10** Institute of Theoretical Astrophysics, Norway **11** Space Telescope Science Institute, USA **12** NASA Goddard Space Flight Center, USA **13** European Space Agency, ESTEC **14** Southwest Research Institute, USA **15** Advanced Research Computing Centre, University College London, UK ¶ Corresponding author

DOI: 10.21105/joss.05296

Software

- Review
- Repository
- Archive

Editor: Axel Donath

Reviewers:

- @maxnoe
- @timj

Submitted: 09 March 2023
Published: 25 July 2023

In partnership with

AMERICAN ASTRONOMICAL SOCIETY

This article and software are linked with research article DOI 10.3847/1538-4357/ace0bd, published in the Astrophysical Journal.

Summary

ndcube is a free, open-source, community-developed Python package for inspecting, manipulating, and visualizing n-dimensional coordinate-aware astronomical data. Its features are agnostic to the number of data dimensions and the physical coordinate types they represent. Its data classes link data and their coordinates and provide analysis methods to manipulate them self-consistently. These aim to provide simple and intuitive ways of handling coordinate-aware data, analogous to how users handle coordinate-agnostic data with arrays. ndcube requires that coordinate transformations be expressed via the World Coordinate System (WCS), a coordinate framework commonly used throughout astronomy. The WCS framework has multiple implementations, e.g. FITS-WCS, gWCS, etc., each with a different incompatible API, which makes workflows and derived tools non-transferable between implementations. ndcube overcomes this by leveraging AstroPy's WCS API [APE-14; Robitaille et al. (2018)] which can be wrapped around any underlying WCS implementation. This enables ndcube to use the same API to interact with any set of WCS transformations. ndcube's data-WCS coupling allows users to analyze their data more easily and reliably, thus helping to boost their scientific output.

Statement of Need

N-dimensional data sets are common in all areas of science and beyond. For example, a series of images taken sequentially with a CCD camera can be stored as a single 3-D array with two spatial axes and one temporal axis. In astronomy, the most commonly used framework for translating between array element indices and the location, time, etc. in the Universe being observed is the World Coordinate System (WCS). WCS's ability to handle many different physical types of coordinates (e.g. spatial, temporal, spectral, etc.) and their projections onto a data array (e.g. right ascension and declination, helioprojective latitude and longitude, etc.) make it a succinct, standardized and powerful way to relate array axes to the physical coordinate types they represent.

*co-first author

There are mature Python packages for handling N-D array operations – e.g. numpy (Harris et al., 2020), dask (Dask Development Team, 2016), etc. – and others for supporting WCS coordinate transformations – e.g. astropy (Astropy Collaboration et al., 2013, 2018, 2022), gWCS (Dencheva et al., 2022). However, none treat data and coordinates in a combined, self-consistent way. The closest alternative to ndcube is xarray (Hoyer & Hamman, 2017). However xarray has been developed for the requirements and conventions of the geosciences which, although similar to those of astronomy in concept, are sufficiently different in construction to cause significant friction. Crucially, xarray does not currently support WCS coordinate transformations. Tools that do support WCS-based coordinate-aware data analysis, such as the SunPy (Mumford et al., 2020) Map class for 2-D images of the Sun, tend to have APIs specific to particular combinations of dimensions, physical types, coordinate systems and WCS implementations. This limits their broader utility and makes the combined analysis of different types of data more difficult. It also inhibits collaboration by erecting technical barriers between sub-fields of astronomy.

ndcube overcomes these challenges via its design policy that all functionalities and APIs must be agnostic to the number of dimensions and coordinate types they represent. Moreover, ndcube's employment of the AstroPy WCS API makes it agnostic to the underlying WCS implementation, e.g. FITS-WCS, gWCS, etc.

The Role of ndcube and its Features

The ndcube package serves three specific purposes. First, it formalizes the NDCube 2 API in software via its abstract base classes (ABCs), `NDCubeABC`, `ExtraCoordsABC` and `GlobalCoordsABC`. The NDCube 2 API is a standardized framework for inspecting and manipulating coordinate-aware N-D astronomical data and is defined by 12th SunPy Enhancement Proposal [SEP-12; Mumford & Ryan (2020)]. A discussion of the philosophies underpinning the NDCube 2 API can be found in Ryan et al. (2023). Second, the ndcube package implements the NDCube 2 API in corresponding data and coordinate classes, `NDCubeBase`, `ExtraCoords` and `GlobalCoords`. These are viable off-the-shelf tools for end users and developers who do not want to implement the API themselves. Third, it provides additional support for coordinate-aware manipulation and visualization of N-D astronomical data via three high-level data classes: `NDCube`, `NDCubeSequence` and `NDCollection`. `NDCube` (note the different capitalization from the package name) inherits from `NDCubeBase` and so adheres to the NDCube 2 API but adds some additional features, such as a default visualization suite. The other classes are designed to handle multiple `NDCube` instances simultaneously.

The features in the ndcube package are designed to be practical tools for end users. But they are also powerful bases upon which to build tools for specific types of data. This might be a specific number and/or combination of physical types (spectrograms, image cubes, etc.), or data from specific instruments or simulations. Thus, ndcube can enhance the productivity of developers by centralizing the development and maintenance of the most useful and general functionalities. This leaves more time for developing a greater range of tools for the community and/or enables part-time developers to devote more effort to other aspects of their jobs, e.g. scientific analysis.

High-level Data Classes

The three high-level data classes provided by the ndcube package are `NDCube`, `NDCubeSequence` and `NDCollection`. `NDCube` requires that the data is stored in a single array object and described by a set of WCS transformations. The array can be any object that exposes `.dtype` and `.shape` attributes and can be sliced by the standard Python slicing API. Thus `NDCube` not only supports numpy arrays but also others such as dask for distributed computing (Dask Development Team, 2016), cupy for GPU operations (Okuta et al., 2017), etc. `NDCube` leverages the AstroPy WCS API for interacting with and manipulating the WCS transformations. This means `NDCube` can

support any WCS implementation, e.g. FITS-WCS, gWCS, etc., so long as it's supplied in an AstroPy-WCS-API-compliant object. The components of an `NDCube` are supplied by the following keyword arguments and accessed via attributes of the same name.

- `data`: The data array. (Required)
- `wcs`: The primary set of coordinate transformations. (Required)
- `uncertainty`: an `astropy.nddata.NDUncertainty` object giving the uncertainty of each element in the data array., optional
- `mask`: a boolean array denoting which elements of the data array are reliable., optional A `True` value implies the data is masked, i.e. unreliable., optional
- `meta`: an object for storing metadata, e.g. a Python dictionary., optional
- `unit`: the unit of the data., optional

`NDCube` also supports additional coordinate information. See the subsection on Coordinate Classes. `NDCube` provides several analysis methods such as slicing (by array indices), cropping (by real world coordinates), reprojecting to new WCS transformations, visualization, rebinning data, arithmetic operations, and more. All these methods manipulate the data, coordinates, and supporting data (e.g. uncertainties) simultaneously and self-consistently. This relieves users of well-defined, but tedious and error-prone tasks.

`NDCubeSequence` is designed to handle multiple `NDCube` instances that are arranged in some order. Cubes can be ordered along an additional axis to those represented by the cubes, e.g. a sequence of 2-D spatial images arranged along a 3rd time axis. In this case users can interact with the data as if it were a 3-D cube with a similar API to `NDCube`. Alternatively, the cubes can be ordered along one of the cubes' axes, e.g. a sequence of tiles in an image mosaic where each cube represents an adjacent region of the sky. `NDCubeSequence` provides APIs for both the (N+1)-D and extended N-D paradigms, that are simultanesouly available on each `NDCubeSequence` instance. This enables users to switch between the paradigms without reformatting or copying the underlying data. `NDCubeSequence` also provides various methods to help with data analysis. These APIs are similar to `NDCube` wherever possible, e.g. slicing, visualization, etc., to minimize friction between analyzing single and multiple cubes.

`NDCollection` allows unordered but related `NDCube` and `NDCubeSequence` objects to be linked, similar to how a Python dictionary is used. However, in addition to dictionary-like features, `NDCollection` allows axes of different cubes with the same lengths to be marked as 'aligned'. This enables these axes on all constituent cubes to be sliced at the `NDCollection` level. One application of this is linking derived data products, e.g. a spectral image cube and a Doppler map derived from one of its spectral lines. Marking both cubes' spatial axes as 'aligned' and slicing the `NDCollection` rather than the two cubes separately, simplifies the extraction of regions of interest and guarantees both cubes continue to represent the same field of view.

More detailed discussion on the roles of the above data classes' features and how to use them can be found in Ryan et al. (2023) and the ndcube documentation (ndcube Developers, 2023).

Coordinate Classes

The ndcube package provides two coordinates classes, `ExtraCoords` and `GlobalCoords`. `ExtraCoords` provides a mechanism for storing coordinate transformations that are supplemental to the primary WCS transformations. This can be very useful if, say, we have a spectral image cube whose images were taken at slightly different times but whose WCS does not include time. In this case `ExtraCoords` can be used to associate an `astropy.time.Time` object with the spectral axis without having to manually construct a new WCS, a potentially complicated task even for experienced users. `ExtraCoords` supports both functional and lookup-table-based transformations. It can therefore also be used as an alternative set of coordinate transformations to those in the primary WCS and used interchangeably.

By contrast, `GlobalCoords` supports scalar coordinates that apply to the whole `NDCube` rather than any of its axes, e.g. the timestamp of a 2-D image. Scalar coordinates are not supported

by WCS because it requires all coordinates to be associated with at least one array axis, hence the need for `GlobalCoords`. When an axis is dropped from an `NDCube` via slicing, the values of the dropped coordinates at the relevant location along the dropped axis are automatically added to the associated `GlobalCoords` object, e.g. the timestamp of a 2-D image sliced from a 3-D space-space-time cube. Thus coordinate information is never lost due to slicing.

`NDCube` objects are always instantiated with associated `ExtraCoords` and `GlobalCoords` objects, even if empty. Users can then add and remove coordinates subsequently. For a more in depth dicussion of `ExtraCoords` and `GlobalCoords`, see Ryan et al. (2023).

Community Applications of ndcube

The importance of the ndcube package is demonstrated by the fact that it is already a dependency of various software tools that support current ground-based and satellite observatories. These include the James Webb Space Telescope (JWST), Solar Orbiter, the Interface Region Imaging Spectrograph (IRIS), Hinode, and the Daniel K. Inouye Solar Telescope (DKIST) via the specutils (specutils Developers, 2023b, 2023a), jdaviz (JDADF-Developers et al., 2023), sunraster (Ryan & et al, in prep.), irispy-lmsal (irispy-lmsal Developers, 2023b, 2023a), EISPAC (EISPAC Developers, 2023b, 2023a) and DKIST user tools packages which all depend on ndcube. ndcube is also used in the data pipeline of the PUNCH mission [Polarimeter to UNify the Corona and Heliosphere; Deforest et al. (2022)], scheduled for launch in 2025. In addition, individual researchers are using the ndcube package in their own analysis workflows.

A network benefit of ndcube is that it standardizes the APIs for handling astronomical coordinate-aware N-D data. Adoption across astronomy and heliophysics helps scientists to more easily work with data from different missions and sub-communities. This can simplify multi-instrument data analysis, foster inter-field collaborations, and promote scientific innovation.

Acknowledgements

We acknowledge financial support for ndcube from NASA's Heliophysics Data Environment Enhancement program, the Daniel K. Inouye Solar Telescope, and Solar Orbiter/SPICE (grant 80NSSC19K1000). We also acknowledge the SunPy, Python in Heliophysics, and AstroPy communities for their contributions and support.

References

Astropy Collaboration, Price-Whelan, A. M., Lim, P. L., Earl, N., Starkman, N., Bradley, L., Shupe, D. L., Patil, A. A., Corrales, L., Brasseur, C. E., Nöthe, M., Donath, A., Tollerud, E., Morris, B. M., Ginsburg, A., Vaher, E., Weaver, B. A., Tocknell, J., Jamieson, W., … Astropy Project Contributors. (2022). The Astropy Project: Sustaining and Growing a Community-oriented Open-source Project and the Latest Major Release (v5.0) of the Core Package. *935*(2), 167. https://doi.org/10.3847/1538-4357/ac7c74

Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., Günther, H. M., Lim, P. L., Crawford, S. M., Conseil, S., Shupe, D. L., Craig, M. W., Dencheva, N., Ginsburg, A., VanderPlas, J. T., Bradley, L. D., Pérez-Suárez, D., de Val-Borro, M., Aldcroft, T. L., Cruz, K. L., Robitaille, T. P., Tollerud, E. J., … Astropy Contributors. (2018). The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package. *156*(3), 123. https://doi.org/10.3847/1538-3881/aabc4f

Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., Greenfield, P., Droettboom, M., Bray, E., Aldcroft, T., Davis, M., Ginsburg, A., Price-Whelan, A. M., Kerzendorf, W. E., Conley, A., Crighton, N., Barbary, K., Muna, D., Ferguson, H., Grollier, F., Parikh, M. M., Nair, P.

H., … Streicher, O. (2013). Astropy: A community Python package for astronomy. *558*, A33. https://doi.org/10.1051/0004-6361/201322068

Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. https://dask.org

Deforest, C., Killough, R., Gibson, S., Henry, A., Case, T., Beasley, M., Laurent, G., Colaninno, R., Waltham, N., & Punch Science Team. (2022). Polarimeter to UNify the Corona and Heliosphere (PUNCH): Science, Status, and Path to Flight. *2022 IEEE Aerospace Conference*, 1–11. https://doi.org/10.1109/AERO53065.2022.9843340

Dencheva, N., Mumford, S., Cara, M., Bradley, L., perrygreenfield, D'Avella, D., Sipőcz, B., Lim, P. L., Jamieson, W., Slavich, E., Shanahan, C., Davies, J., Earl, N., Burnett, Z., Simon, B., Tollerud, E., Deil, C., Streicher, O., Simpson, C., … Geiger, Z. (2022). *Spacetelescope/gwcs: GWCS v 0.18.3* (Version 0.18.3). Zenodo. https://doi.org/10.5281/zenodo.7478201

EISPAC Developers. (2023a). EISPAC code-base. In *GitHub Repository*. GitHub. https://github.com/USNavalResearchLaboratory/eispac

EISPAC Developers. (2023b). EISPAC documentation. In *ReadTheDocs*. ReadTheDocs. https://eispac.readthedocs.io/en/latest

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hoyer, S., & Hamman, J. (2017). Xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*, *5*(1). https://doi.org/10.5334/jors.148

irispy-lmsal Developers. (2023a). Irispy-lmsal code-base. In *GitHub Repository*. GitHub. https://github.com/LM-SAL/irispy-lmsal

irispy-lmsal Developers. (2023b). Irispy-lmsal documentation. In *ReadTheDocs*. ReadTheDocs. https://irispy-lmsal.readthedocs.io/en/v0.1.5/

JDADF-Developers, Averbukh, J., Bradley, L., Buikhuizen, M., Busko, I., Cherinka, B., Conroy, K., Earl, N., Fox, O., Geda, R., Jones, C., Karatay, H., Kotler, J., Lim, P. L., Morris, B., Nguyen, D., O'Steen, R., Ogaz, S., Ogle, P., … Volfman, S. (2023). *Jdaviz* (Version 3.3.0). Zenodo. https://doi.org/10.5281/zenodo.7625637

Mumford, S., Freij, N., Christe, S., Ireland, J., Mayer, F., Hughitt, V., Shih, A., Ryan, D., Liedtke, S., Pérez-Suárez, D., Chakraborty, P., K, V., Inglis, A., Pattnaik, P., Sipőcz, B., Sharma, R., Leonard, A., Stansby, D., Hewett, R., … Murray, S. (2020). SunPy: A Python package for Solar Physics. *The Journal of Open Source Software*, *5*(46), 1832. https://doi.org/10.21105/joss.01832

Mumford, S., & Ryan, D. F. (2020). SEP-0012: NDCube 2 API. In *GitHub repository*. GitHub. https://github.com/sunpy/sunpy-SEP/blob/master/SEP-0012.md

ndcube Developers. (2023). Ndcube documentation. In *ReadTheDocs*. ReadTheDocs. https://docs.sunpy.org/projects/ndcube/en/v2.1.1/

Okuta, R., Unno, Y., Nishino, D., Hido, S., & Loomis, C. (2017). CuPy: A NumPy-Compatible library for NVIDIA GPU calculations. *Proceedings of Workshop on Machine Learning Systems (LearningSys) in the Thirty-First Annual Conference on Neural Information Processing Systems (NIPS)*. http://learningsys.org/nips17/assets/papers/paper_16.pdf

Robitaille, T., Tollerud, E., Mumford, S., & Ginsburg, A. (2018). *Astropy Proposal for Enhancement 14: A shared Python interface for World Coordinate Systems (APE 14)*. https://doi.org/10.5281/zenodo.1188875

Ryan, D., & et al. (in prep.). sunraster: Manipulating Solar Slit-spectrograph Observations in Python. *Journal of Open Source Software*.

Ryan, D., Mumford, S., Barnes, W. T., Kumar Baruah, A., Bhope, A., Buchlin, E., Freij, N., Ginsburg, A., Hayes, L. A., Homeier, D., Hughes, J. M., Lowder, C., O'Steen, R., Pellorce, B., Robitaille, T., Sharma, Y., Shih, A. Y., Tollerud, E., & West, M. J. (2023). A Unified Framework for Manipulating N-dimensional Astronomical Data and Coordinate Transformations in Python: The NDCube 2 & Astropy APE-14 WCS APIs. *ApJ*. https://doi.org/10.3847/1538-4357/ace0bd

specutils Developers. (2023a). Specutils code. In *GitHub Repository*. GitHub. https://github.com/astropy/specutils

specutils Developers. (2023b). Specutils documentation. In *ReadTheDocs*. ReadTheDocs. https://specutils.readthedocs.io/en/stable/