ndcube: a Python package for N-D astronomical coordinate-aware data analysis

Daniel F. Ryan
UCL/MSSL

Alliance, 21 October 2026



What is ndcube?



What is ndcube?

- A Python package for astronomical N-dimensional coordinate-aware data analysis.
- Free, open-source, community-developed, tested.
- Developed and maintained under the SunPy project, but available to all beyond SunPy too.



Core principle

ndcube is for handling an **data** whose coordinates are described by a **World Coordinate System (WCS) object**.

- Support for WCS is ndcube's USP and makes it uniquely suited to astronomy.



ndcube's data classes

NDCube

- One data array and one set of WCS coordinates. (Supporting/metadata data also supported).
- Flagship data class of package

NDCubeSequence

- A sequence of NDCube's ordered along a coordinate axis.
  - e.g. Images with different pointing over time.

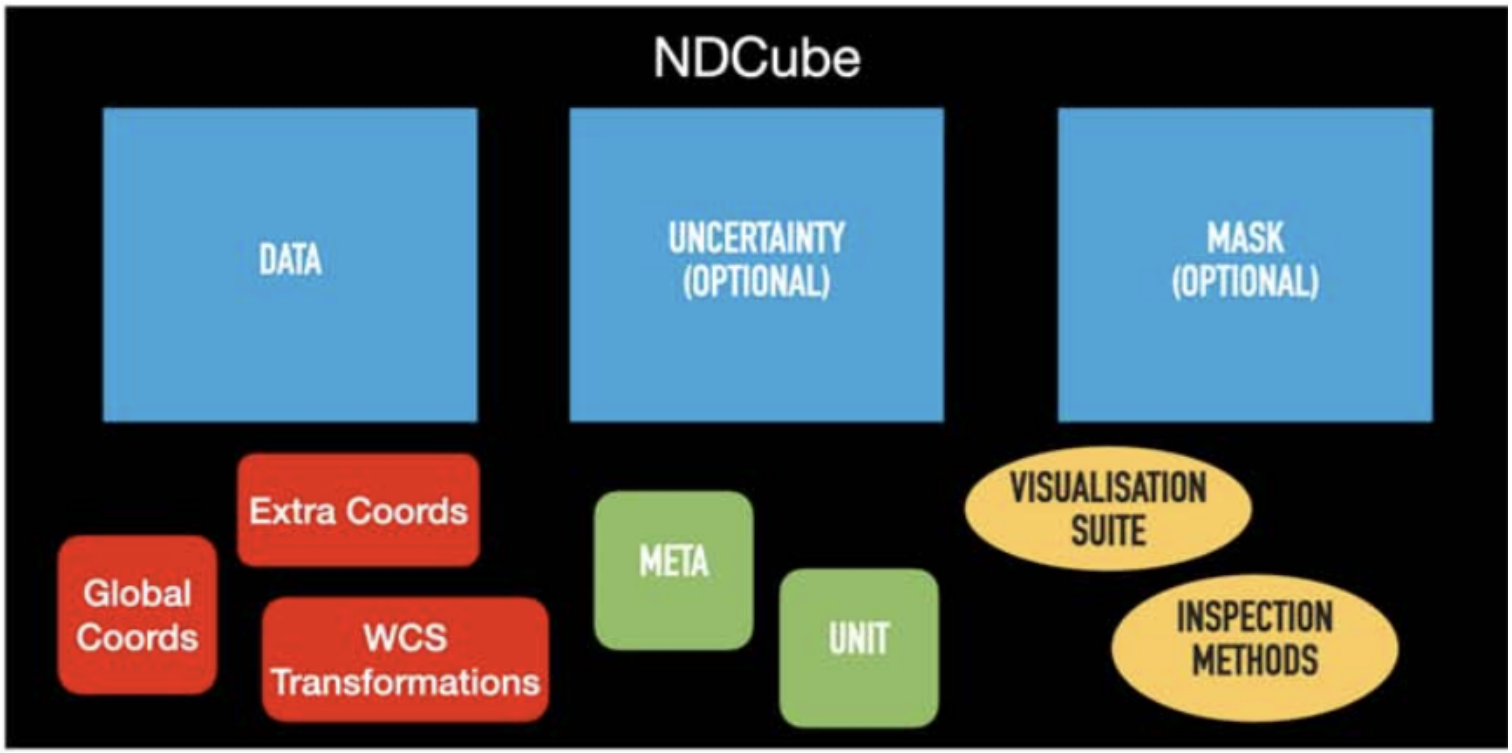
NDCollection

- A collection unordered of NDCubes/NDCubeSequences.
  - e.g. spectral-image cube and derived doppler map
- Functionalities for handling shared coordinate axes (not required).



Contents of an NDCube

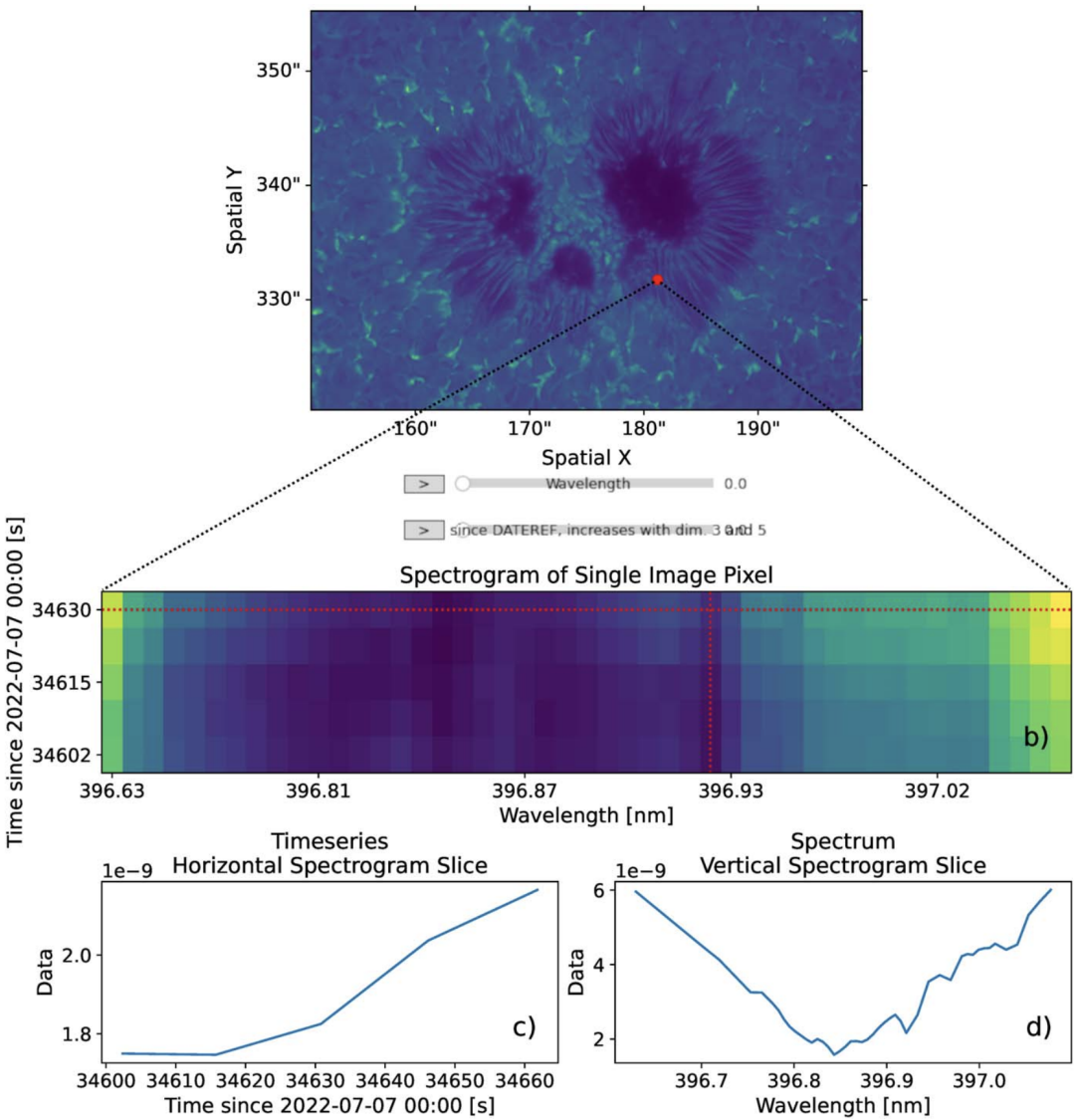
- Inherits from astropy.nddata.NDData
- 1 data array + 1 WCS object. Supporting metadata optional.



(a)



NDCube is ideal for multi-dimensional data analysis



cube.plot()

cube[:, :, y, x].plot()

cube[:, lam, y, x].plot()

cube[t, :, y, x].plot()



Functionalities of NDCube

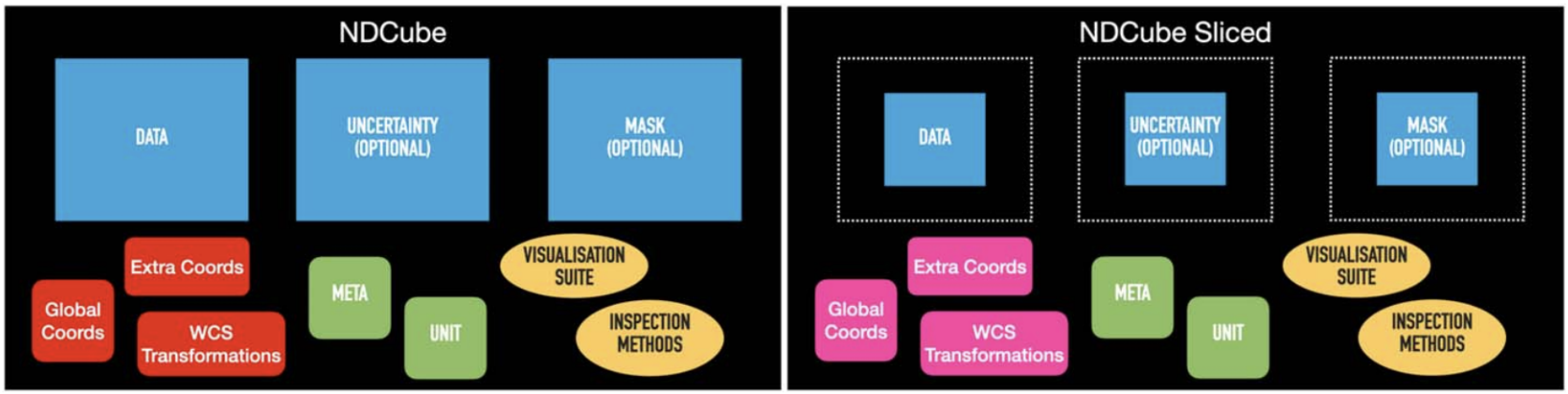
Key Point

All operations self-consistently alter data, coordinates and metadata as appropriate.



NDCube Slicing & Cropping

Slicing: Slicing by array indices



(a) (b)

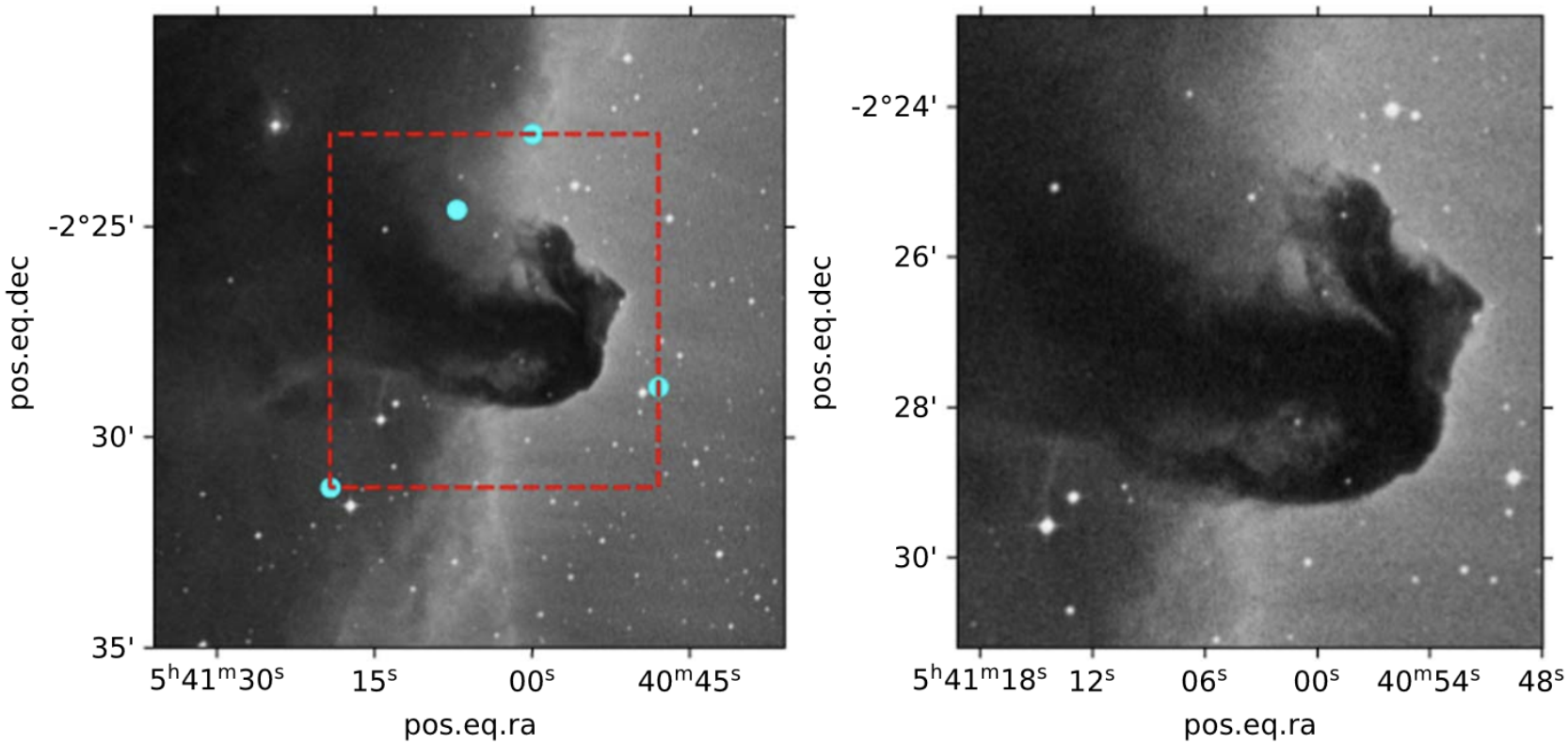
```
>>> cube[y1:y2, x1:x2]
# y1, y2, x1, x2 are ints
```



NDCube Slicing & Cropping

Slicing: Crop by array indices

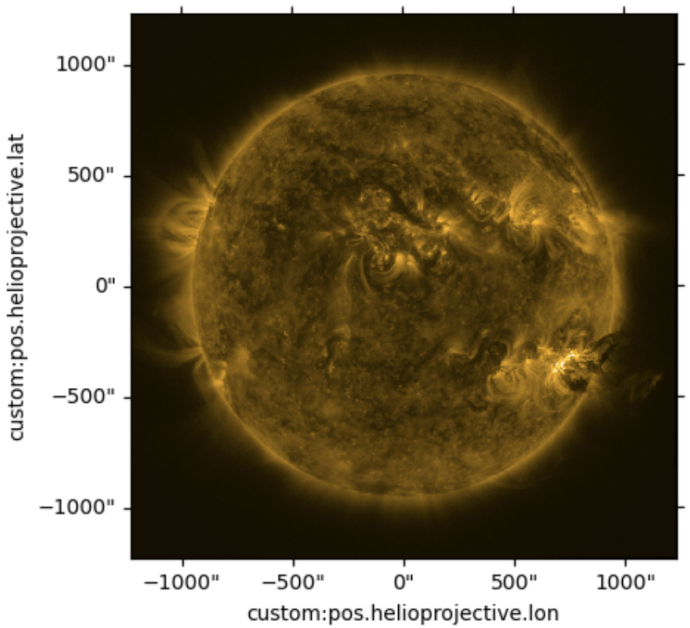
Cropping: Crop by coordinates

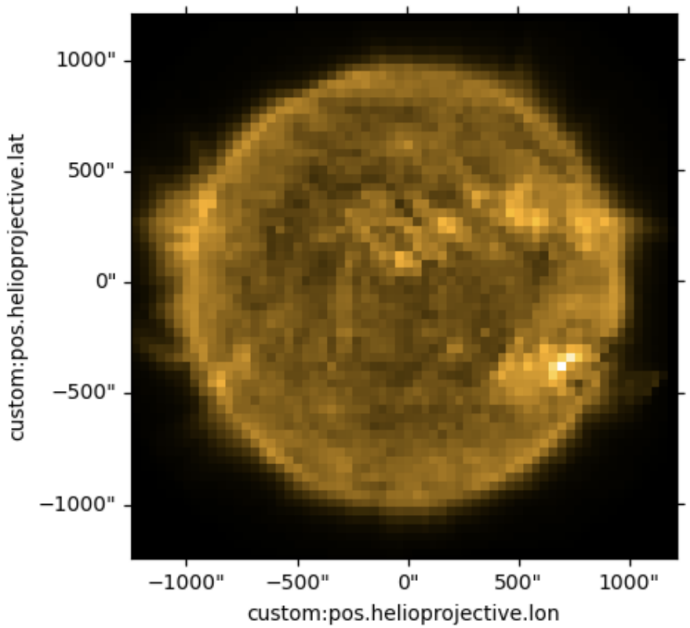



```
cube.crop((celestial1,), (celestial2,), (celestial3,), (celestial4,))
# celestials are astropy SkyCoords
```

NDCube Rebinning

```
rebinned_cube = cube.rebin((16, 16), operation=np.sum)
```

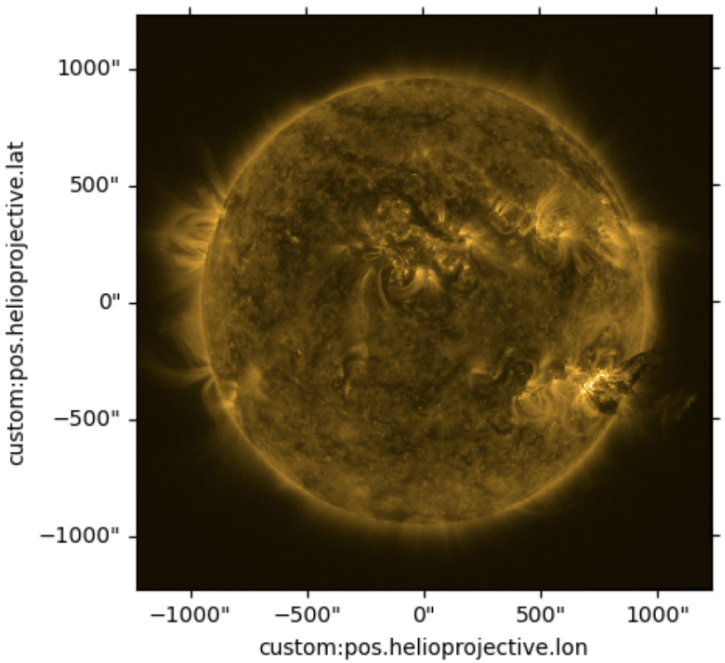


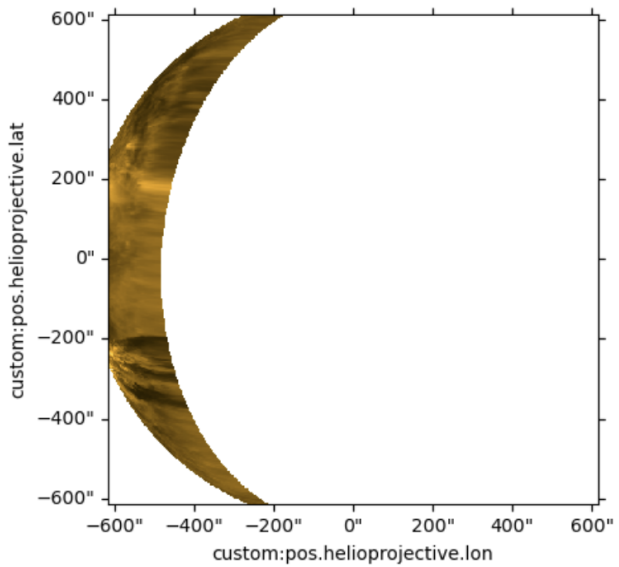




Reprojecting NDCube to different viewpoint

```
cube_from_mars = cube.reproject_to(wcs_from_mars)
```







Arithmetic operations

NDCube arithmetic operations consistently alter data, unit, and uncertainties, where appropriate.

Supported operands

- float, ints, etc.
- arrays
- astropy Quantities
  - Units are correctly propagated.

Supported operations

- addition, +
- subtraction, -
- multiplication, *
- division, /
- raise to power, **

Examples

```
>>> new_cube = cube * 10
>>> new_cube = cube + np.ones(cube.data.shape)
>>> new_cube = cube / (3 * u.s)  # new_cube.unit includes s^-1
>>> new_cube = cube**2
```



Recently Added Features



Arithmetic Operations with astropy NDData

More complex arithmetic operations between NDCube and astropy NDData now support:

- uncertainty propagation
- mask handling
- unit propagation

NDCube.fill_masked()

Deciding how masked values are treated can be done before operation via:

```
>>> filled_cube = cube.fill_masked(fill_value, uncertainty_fill_value=?
                                   unmask=False/True)
```

NDCube.to_nddata()

Arithmetic operations between coordinate-aware objects is ambiguous. Coordinates can be dropped from NDCube before operation via:

```
>>> cube_no_coords = cube.to_nddata(wcs=None)
```



Saving NDCubes to ASDF

```
>>> my_tree = {"mycube": cube}
>>> asdf.AsdfFile(tree=my_tree).write_to("somefile.asdf")
```



NDMeta: Axis-aligned metadata

NDMeta: a dict-like that allows metadata to be associated with axes of an NDCube $\implies$ sliceable.

Example

```
>>> cube.meta.add("exposure time", [1.9, 2.1, 5, 2, 2] * u.s, axes=0)
>>> cube[1:3].meta["exposure time"]
<Quantity [2.1, 5], s>
```

Metadata can be associated with multiple axes.



What's next for ndcube?



What's next for ndcube?

Funding:

- SunPy recently won a 3-year NASA Heliophysics Tools and Methods maintenance award, including for ndcube.

New Features:

- More user-friendly, coordinate-type-specific API for NDCube.



NDCube's future coordinate-type-specific API

Aim: Provide more user-friendly API wrappers for cases where WCS includes known physical types.

Planned supported physical types:

1. celestial
2. spectral
3. time
4. stokes

This will supplement, NOT replace, pre-existing API.



Possible inclusions in coordinate-type-specific API

Extracting coordinate grid

Current

```
>>> cube.axis_world_coords("lon")
```

Proposed

```
>>> cube.celestial.coords
```

Cropping

Current

```
>>> cube.crop((..., skycoord1, None,...),
              (..., skycoord2, None,...),...)
```

Proposed

```
>>> cube.celestial.crop(skycoord1,
                        skycoord2,...)
```

Well known aliases

Current

```
>>> cube.axis_world_coords("freq")[0].to(
        u.nm, equivalencies=u.spectral())
```

Proposed

```
>>> cube.spectral.wavelength
```

How might this help your use-cases?



Take-home messages

- ndcube provides generalised data analysis and visualisation tools for N-D coordinate-aware astronomical data analysis.
  - WCS-coordinate support is ndcube's USP.
- By acting as a dependency, ndcube reduces code bases and maintenance loads of packages throughout astronomy.
- Funding has been secured for maintaining ndcube for coming years, but more help is always welcome.
- A supplementary, user-friendly, coordinate-type-specific API is in development. Input welcome!

ndcube docs

ndcube repo

