Volume Title
ASP Conference Series, Vol. **Volume Number**
Author

Astropy: Building Blocks for Astronomy Software

Erik M. Bray

Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, MD 21218, USA

Abstract. The Astropy Project is a community effort to develop an open source Python package of common data structures and routines for use by other, more specialized astronomy software in order to foster interoperability. The project encompasses the "core" `astropy` Python package, "affiliated packages" that strive to implement Astropy's coding standards and interoperability with other affiliated packages, and a broader community aimed at implementing Pythonic solutions to astronomy computing problems while minimizing duplication of effort. The project also provides a template for other projects that use Astropy to reuse much of Astropy's development framework without reinventing the wheel. Here we present an overview of the key features of the core package (existing and upcoming), current and planned affiliated packages, and how we manage a large open source project with a diverse community of contributors.

1. Core `astropy` Capabilities

The `astropy` package for Python (The Astropy Collaboration et al. 2013), referred to as the "core" package of the Astropy Project, contains classes and functions to address common needs of other astronomy software as well as interactive data analysis. It also includes a framework of sundry software development utilities such as a test framework, documentation helpers, a configuration system, and build and installation support.

Here we list some of the sub-packages that make up the functionality of the core `astropy` package as of version 0.2, as well as some new functionality that will debut in version 0.3.

1.1. Core Data Structures and Transformations

`astropy.units`. Handles defining and converting between physical units, and performing arithmetic with physical quantities (numerical scalars and arrays with associated units) using a powerful `Quantity` class.

`astropy.constants`. Contains a number of physical constants useful in astronomy and physics. Constants are represented as `Quantity` objects with additional metadata describing their provenance and uncertainties.

`astropy.nddata`. Provides an `NDData` class for managing n-dimensional array-based data with attached metadata, errors, and masking. Also provides new convolution routines that treat NaN values properly.

astropy.table. Provides a powerful `Table` class for storing and manipulating heterogeneous tables of data in a way that is familiar to users of NumPy (Oliphant 2006; Van Der Walt et al. 2011), though with more flexibility than the data structures built into NumPy, including support for metadata and masking.

astropy.time. Provides functionality for manipulating times and dates. Specific emphasis is placed on supporting time scales (e.g. UTC, TAI, UT1) and time representations (e.g. JD, MJD, ISO 8601) that are used in astronomy. Wraps the ERFA library of time and calendar routines, which is a fork of SOFA (Hohenkerk 2010) released under a more permissive license.

astropy.coordinates. Provides classes for representing celestial coordinates and transformation functions for converting between standard systems in a uniform way.

astropy.wcs. Contains utilities for managing WCS transformations in FITS files. These transformations map the pixel locations in an image to their real-world units. This sub-package is an import of the already widely used PyWCS package.

astropy.modeling. Provides a framework for representing models, performing model evaluation, and fitting. It supports 1D and 2D models and fitting with parameter constraints and an API for defining new models and fitting algorithms.

1.2. File I/O

One goal of Astropy is to minimize the effort of getting data into and out of different file formats. To that end it supports a registry of I/O "connectors" that allow data from different formats to be transparently read into and serialized from Astropy's generic `NDData` and `Table` classes. For example, a `Table` can be read from a FITS file (Pence et al. 2010), modified as needed, and written out to an HDF5 file. Support for lower-level operations on specific file formats is maintained as well.

astropy.io.fits. A port of the already widely used PyFITS (Barrett & Bridgman 1999) package, supporting reading and writing FITS files including support for many non-standard FITS conventions.

astropy.io.ascii. A port of the popular asciitables package with support for reading from and writing a wide range of plain text formats to and from Astropy `Table` objects.

astropy.io.votable. A port of the VOTable package for reading and writing the IVOA VOTable format (Oschenbein et al. 2009) to and from Astropy `Table` objects, as well as validating them.

astropy.io.misc. Includes optional support for tables stored in HD5 where h5py is available.

1.3. Astronomy Computations and Utilities

`astropy.cosmology`. Contains classes for representing cosmologies, and utility functions for calculating commonly used quantities that depend on a cosmological model. This includes distances, ages and lookback times corresponding to a measured redshift or the transverse separation corresponding to a measured angular separation.

`astropy.stats`. This sub-package holds statistical functions and algorithms used in astronomy, some of which are more specialized than one would find in `scipy.stats`.

`astropy.vo`. Handles simple access for Virtual Observatory (VO) services.

2. Affiliated Packages

The Astropy Project includes the concept of "affiliated packages". An affiliated package is an astronomy-related Python package that is not part of the core `astropy` package, but has requested to be included under the umbrella of the Astropy Project. Some affiliated packages may be considered for inclusion in the core package once they reach a reasonable level of maturity and usefulness. Others may be more specialized or have enough external dependencies (GUI libraries, for example) that they make more sense as separate packages. In general we hope that becoming an Astropy affiliated package will be seen as a good way for new and existing packages to gain exposure.

The standards for inclusion in the Astropy Project are not strict, but it does imply interoperability with other affiliated packages through use of the common classes and data structures provided by the core package. It also implies adoption of similar coding standards, as well as high standards of documentation, testing, ease of installation, and open access. To help affiliated package developers achieve these goals we provide our framework in the form of a package template that can be used to bootstrap new projects.

Some currently featured affiliated packages[1] include Montage-wrapper, Ginga, APLpy, astroML, and Astropysics. A few additional affiliated packages that are currently under development are photutils, astroquery, specutils, and kcorrect.

3. Community and Collaboration

The Astropy Project was launched as a response to an increasing proliferation of Python packages (often with "astro" in their names) that attempt to build core sets of routines ported from IDL and/or IRAF. Many of these efforts are maintained by one or two individuals or an individual institution, and have not always involved much community support or feedback. We became concerned that too much effort was being spent reinventing the wheel, rather than contributing to a common, open source code base.

A primary guiding principle behind Astropy has been that it is developed by and for the astronomy community, and that it strives to include contributions from users and developers from around the world and from all sub-disciplines. It also aims to encourage best practices from software engineering in the hope of creating a clean,

[1] `http://www.astropy.org/affiliated/index.html`

consistent, and stable API; thorough and well-maintained documentation;[2] and constant feedback through careful issue tracking and continuous integration testing. All of our development practices and guidelines are documented so that other projects—whether affiliated with Astropy or not—may adopt them without having to develop their own standards from scratch.

In order to better enable collaboration we have adopted the GitHub code hosting service which provides an issue tracker and a git source code repository. GitHub is designed around collaboration and enables a low barrier of entry to submitting new code and bug fixes to open source projects.

Astropy's development process ensures that any bug fixes or new features that are implemented are accompanied by tests and any relevant updates to the documentation. Free documentation building and hosting by Read the Docs ensures that the latest documentation builds correctly and is accessible online. All new pull requests are tested against multiple Python and NumPy versions using the continuous integration system Travis. Comprehensive testing against multiple OS platforms (Linux, MacOS X, and Windows) is made possible by Jenkins with hosting from Shining Panda. Multi-platform testing has made it possible for system integrators to include Astropy in many other software package bundles such as Enthought Canopy, Continuum Analytics' Anaconda, and OS packaging systems like Debian's APT.

Having an accessible development workflow has attracted contributions from many new developers (over 50 source code contributors at the time of writing). The ease of contribution has blurred the line between developers and users—in fact, we have even implemented a feature that allows anyone reading the documentation to submit edits and improvements with just a few clicks in their web browser and no prior knowledge of the git version control system. We strive for continuous feedback and rework and hope to avoid a "designed by committee" feel.

References

Barrett, P. E., & Bridgman, W. T. 1999, in Astronomical Data Analysis Software and Systems VIII, vol. 172 of Astronomical Society of the Pacific Conference Series, 483

Hohenkerk, C. 2010, Scholarpedia, 5, 11404. doi:10.4249/scholarpedia.11404

Oliphant, T. 2006, A Guide to NumPy, vol. 1 (Trelgol Publishing USA)

Oschenbein, F., Williams, R., Davenhall, C., Durand, D., Fernique, P., Giaretta, D., Hanisch, R., McGlynn, T., Szalay, A., Taylor, M. B., & Wicenec, A. 2009, VOTable Format Definition, Version 1.2, International Virtual Observatory Alliance (IVOA). Http://www.ivoa.net/Documents/VOTable/20091130/REC-VOTable-1.2.html

Pence, W. D., Chiappetti, L., Page, C. G., Shaw, R. A., & Stobie, E. 2010, A&A, 524, A42

The 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., Günther, H. M., Deil, C., Woillez, J., Conseil, S., Kramer, R., Turner, J. E. H., Singer, L., Fox, R., Weaver, B. A., Zabalza, V., Edwards, Z. I., Azalee Bostroem, K., Burke, D. J., Casey, A. R., Crawford, S. M., Dencheva, N., Ely, J., Jenness, T., Labrie, K., Lian Lim, P., Pierfederici, F., Pontzen, A., Ptak, A., Refsdal, B., Servillat, M., & Streicher, O. 2013, ArXiv e-prints. 1307.6212

Van Der Walt, S., Colbert, S., & Varoquaux, G. 2011, Computing in Science & Engineering, 13, 22

[2] `http://docs.astropy.org`