astropy.io

Oguz Tanatar

May 12, 2025

1. The name of the package is astropy.io. The basic aim of the package is to enable its users to read and write (or process outputs and inputs, respectively) data, specifically astronomical data. This data can be in many formats (VOTable, FITS file, ASCII Table, etc.). The package doesn't necessarily solve much by itself, but can be used in conjunction with other packages (namely matplotlib, pandas, scipy, and numpy) to fully generate data from otherwise hard-to-process data.
2. I picked this package, because I will need to use astropy.io's FITS reader capabilities in my research very frequently to read output FITS files from a program called GALPROP, which reconstructs cosmic ray spectra. The FITS reader I have right now is very bad, so I want to be able to improve it, and I thought using this project as an excuse to explore astropy would be very useful.
3. The package was initially released in 2012, and has underwent 7 versions since its initial release. It has a long genealogy from before it was released as well. In general, the story started with the Space Telescope Science Institute's work involving python in astronomy. In Fall 2011, a conversation around uniting astronomy python developers started, and that eventually lead to the release of the astropy in 2012, with astropy.io being shipped from the start. . Due to the history of astropy thoroughly involving the consolidation of relevant code, there are several other codes that can solve some of the same problems as astropy (e.g. pandas can also process data well), although astropy.io is a good one-stop-shop to handle everything.
4. astropy.io is still maintained. Matter of fact, the very day this was written, an update was shipped for astropy.io. The authors of astropy do not give a list of who worked on what and during what time periods. The package has grown exponentially over time, despite the developers' call for more developers. To contribute to the documentation, a developer must download the 'dev' version of astropy, report bugs, and join the dev-mailing list in order to contribute files. Consult the astropy contributor documentation for full detail.
5. How easy was it to install? What commands did you use?
    a. In the documentation, the user is instructed to use either a Windows, or a Unix-like system's Command Prompt. I am more comfortable with WSL (Linux Command Prompt on Windows), so I will use that for this installation.

b. The user is first instructed to create an isolated python environment. To achieve this purpose, I created a virtual environment using the following command.
    i. python3 -m venv ~/PHYS265_astropy_venv
        1. the -m flag tells your system to use the intended python version if your computer has multiple python versions.
    ii. You can replace ‘python3’ with ‘python’, depending on how your system is configured.

c. Then, I activate this virtual environment with:
    i. source ~/PHYS265_astropy_venv

d. from here, astropy can be installed with the command:
    i. python3 -m pip install ‘astropy[recommended]’

e. Although I chose to use a venv, it is not the only possible method of creating the isolated environment. It was simply my preference.

6. Does it install via the standard pip/conda?
    a. Astropy installs via the standard pip OR conda. My preferred method is pip, so that’s what I used in 5, but the documentation provides instructions for how to install using conda. astropy.io is installed by default.

7. The source code can be seen in two ways, namely. First, the code can be viewed in the astropy github. The astropy.io code, specifically, can be seen inspected in the ‘io’ subfolder. Otherwise, the same code can be inspected offline in a text editor with the following process.
    a. In the WSL virtual environment, run:
        i. cd ~/PHYS265_astropy_venv/lib/python-[version]/site-packages/astropy/io
        ii. Using vim, nano, or your favorite text editor, you can then inspect the code.

8. is the code used by other packages (if so, give one or two examples). ASCL codes have citations via their ADS link. See also 22.

    a. Astropy, and by relation, astropy.io, has several related packages. One (see below) is WCS, which is another astropy package commonly used with .io. Further, the specutils, cubeviz, reproject, SunPy, potutilsm and several other packages are related to astropy.

9. How is the code used. Is it commandline, python script, or a jupyter notebook, or even a web interface?

   a. The code is used via a python script, however an ipynb file would work too. Generally, command prompt's use ends at the installation of the package and running the written code. I used a notebook, purely so the individual outputs would be easier to read.

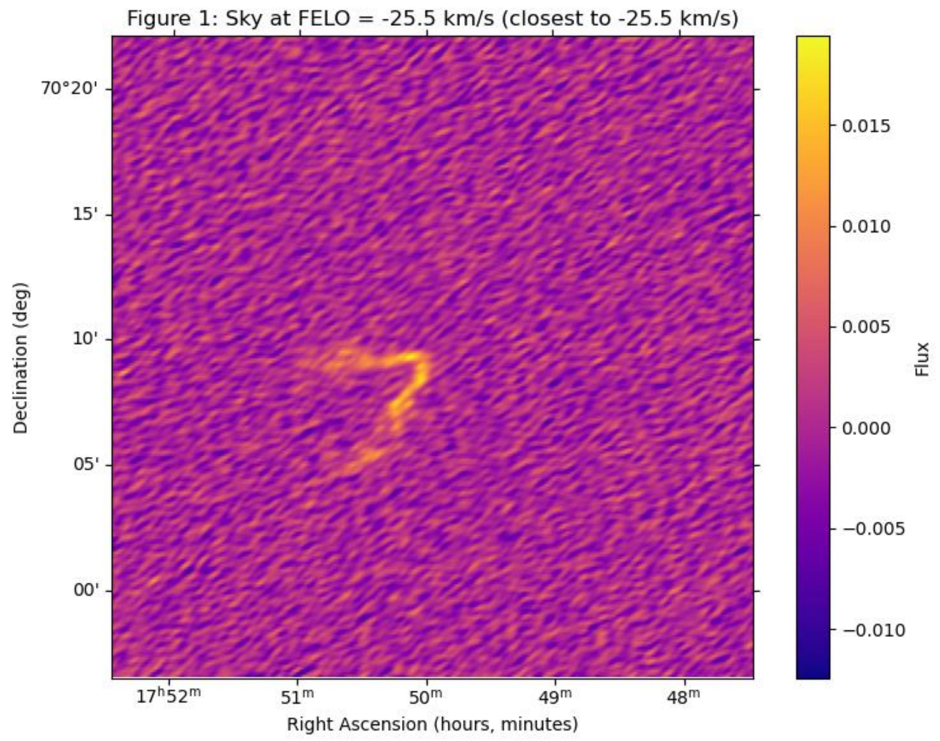
10. provide examples using the code. if you prefer to use a jupyter notebook instead of a python script, that's ok. See also 12.

    a. Opening FITS Files, overwriting certain header units, and plotting FITS data is shown in the notebook.

11. does the package produce figures, or are you on your own? Is matplotlib used?

    a. Astropy.io does not produce figures by itself. Instead, it can be used in conjunction with astropy.visualization and matplotlib to produce plots. This can be seen in the accompanying notebook.

12. your code and report should show at least one figure, and create a nice figure caption explaining what it shows. You notebook should show how the figure was made (i.e. be reproducable). Second figure is optional, but only use it when you need to illustrate something extra.

    a. 

    b. Figure 1: The HI Spectra plotted against sky coordinates at FELO-HEL = -25458. The code to produce the figure is in the notebook.

13. is the package pure python? or does it need accompanying C/C++/Fortran code?

    a. Astropy, generally, is written in python, but it has some components written in other languages like C to improve efficiency. The astropy.io library, specifically, is again mostly in python, with the primary logic and bread and butter written in python, while the FITS reader, for example, is written in C, so that it can use the library 'cfitsio'.

14. what is the input to the package? Just parameters, or dataset(s), or can they be generated from scratch?

    a. The input can be in the form of several types of data (FITS, ASCII, VOTable, HDF5, etc.). These are all datasets. These can be generated from scratch, although through entirely different avenues. There are certain softwares, like GALPROP, that use theory to generate the FITS. Otherwise, experimental data should be declared. Astropy.io also has capabilities to turn appropriate files into the FITS format.

15. what is the output of the package? Just parameters, or dataset(s)?, or just a screen output you would need to capture

    a. The output files are (again) data sets, just that instead of being in an unreadable format, is instead in our standard numerical format. Specifically for FITS, they are in the form of a Header + Data Unit (HDU). Elaborated on further in the notebook's markdown cells and comments ('We now have a readable…' , '#displaying astropy.io's …').

16. does the code provide any unit tests, regression or benchmarking?

    a. The astropy.io module does not specifically provide any tests, but it is compatible with unit, regression, and benchmark tests for general python packages. These can be in the form of pytest, unittest, BytesIO, or even efficiency packages like ASV.

17. how can you feel confident the code produce a reliable result? (see also previous question)

    a. We can check for reliability by checking for proper overlays with plots from known sources. One of these sources could be SAOImage ds9. Other options can be an overlay with published data from papers, where the experimental signal itself is salvaged and checked against the given signal. Otherwise, since FITS does provide numerical data as well, one can plot the residuals of data, and see if there is statistical agreement.

    b. Although these are not included, the two plots, when put side-by-side, do in fact show agreement.

18. what (main) python package(s) does it use or depend on (e.g. numpy, curve_fit, solve_ivp) - how did you find this out?

    a. It has several dependencies. The exhaustive list is as follows: yaml, contourpy, cycler, dateutil, erfa, fontTools, kiwisolver, matplotlib, mpl_toolkits, numpy, pillow, pip, pyerfa, pyparsing, scipy.

    b. In addition, although not a hard-coded dependency, it is generally good to use WCS, as many FITS files work in pixels, which do not provide direct information about location, time, and other quantities of importance. WCS can extract these quantities, allowing much more meaningful representation of the data.

19. what kind of documentation does the package provide? was it sufficient for you?

    a. The documentation can be found clearly on the astropy documentation page's user guide. This can be used in complement with community forums. I used the documentation extensively, and I thought it was largely sufficient for the pure coding part, when used in conjunction with forums and a couple google searches.

    b. However, the format of the FITS Cube changes wildly on a case-by-case basis in my experience (i.e. the FITS Cubes I am learning to work with in my research are very different from the one provided), and so significantly more research is required to be able to properly navigate a specific FITS Cube. For this, it's more about learning the context behind the data than reading the documentation, so it wasn't very helpful in this front.

20. if you use this code in a paper, do they give a preferred citation method?

    a. The authors instruct users to cite three papers: 'Astropy I (v0.2)', 'Astropy II (v2.0)', 'Astropy III (v5.0)' The specific bibtex can be found in the references section.

21. provide any other references you used in your report.

    a. Astropy preferred citation method: https://www.astropy.org/acknowledging.html

    b. Airspeed Velocity (asv) front page: https://asv.readthedocs.io/en/stable/

    c. Testing Documentation (pytest, BytesIO, unittest respectively): https://docs.pytest.org/en/stable/, https://docs.python.org/3/library/io.html#binary-i-o, https://docs.python.org/3/library/unittest.html

d. Astropy General User Guide (heavily used the FITS and WCS sections).: https://docs.astropy.org/en/stable/index_user_docs.html

e. Astropy Affiliated Packages: https://www.astropy.org/affiliated/

f. Astropy Contribution Guide: https://www.astropy.org/contribute.html

g. SAOImage DS9: https://sites.google.com/cfa.harvard.edu/saoimageds9/home?authuser=0

22. can you find two other papers that used this package. E.g. use ADS citations for ASCL based code. See also 8. Hee are the ADS Links

23. https://ui.adsabs.harvard.edu/abs/2013A%26A...558A..33A/abstract

24. https://ui.adsabs.harvard.edu/abs/2020A%26C....3200384B/abstract

25. did you have to learn new python methods to use this package? Or was the class good enough to get you through this project.

    a. I had to learn many new python methods to get through the project, primarily because in lecture, we have almost every function handed to us in very simple terms. However, when reading documentation with almost zero background, a lot of google searching is required to keep up. This resulted In me learning many new methods required in processing FITS, although I don't know if I learned any general methods with python.

26. Final Disclaimer: you need to state if you have prior experience in using the package or the data, or this is all new to you. In addition, if you collaborated in a group, as long as this is your work.

    a. I have very very little prior experience using astropy.io, as I explained in 2 (I probably have used it for ~6 hours in total before this project). My work with it has been very limited. I have not seen the data at all before. I am not working in a group.