`aplpy`: Slicing it Right

Isaac Sherwood | May 18 2025

1 Introduction [Q. 1 - 4]

The Python package discussed in this report is the Astronomical Plotting Library in Python, better known as `aplpy`. The version used was the most recent version, version 2.2.0, released November 14, 2024. The package allows for efficient processing of astronomical data into publishable quality plots, with extensive tools available for visualizing flexible image transport system (FITS) files.

The package was selected as a way to explore plotting tools that expand upon tools and methods we discussed in class. I was also interested in exploring something specifically astronomy related, as opposed to the physics problems we focused on in the labs, as well as in PHYS276. The package was recommended by my peers as a useful method of analyzing astronomical data.

The `aplpy` package was created in 2009 by creators Thomas Robitaille and Eli Bressert. The package relies on the WCSaxes package of `astropy` to handle coordinate axes and grids, which in turn uses `matplotlib` to create plots. `aplpy` 2.0 and newer versions are compatible only with Python 3.5 and later. Comparable plots can be created using only `astropy` and/or `matplotlib`, however the process is much more cumbersome. The goal of `aplpy` is not to replace these packages, but rather to make it easier and more efficient to make high quality plots utilizing those tools.

The `aplpy` project is still maintained, with the most recent update made via GitHub on November 14, 2024. The project is currently maintained by creator Robitaille and a small group of other volunteer contributors. Users can contribute by reporting issues and new ideas in GitHub. The most recent updates include contributions from new community members who were not previously involved in the project.

2 Installation and source code [Q. 5 - 9]

The `aplpy` package can be easily installed using the `pip` package manager by running `!pip install aplpy` in the command line. The process is straightforward and requires no additional commands or other steps. Both `astropy` and `matplotlib` must also be installed in order for the package to be used.

All source code is available on the project GitHub. There, users can inspect and edit any of the original source code and get the most recently updated versions. Because `aplpy` is a relatively niche package, no other widely used Python packages rely on or use `aplpy`. The package is generally used in Python (.py) files or ipython notebook (.ipynb) files, but can also be used via the command line. The package can generally be used in any environment in which `astropy` and `matplotlib` are available.

3 Examples of code use [Q. 10 - 12]

The primary functionality of `aplpy` is for for the display of FITS files. Before analyzing a FITS file, it is recommended that the `get_pkg_data_filename` function from `astropy` be imported and used to retrieve the FITS file and assign it a local filename. This is not strictly necessary, but is recommended by the creators of `aplpy`.

Visualization of FITS files can then be carried out using the function `aplpy.FITSFigure`. This function takes the name of the FITS file as assigned by `get_pkg_data_filename` as its first and only positional argument. Several additional keyword arguments are available, most notably `dimensions` and `slice`. The `dimensions` keyword argument specifies the dimensions of the FITS file to be plotted against each other. The default is `[0, 1]`, plotting the zeroth and first dimensions. The `slices` keyword argument specifies the locations in dimensions other than those being plotted to be held constant during plotting. The default value is `[]`, corresponding to a two dimensional FITS file with no additional dimensions

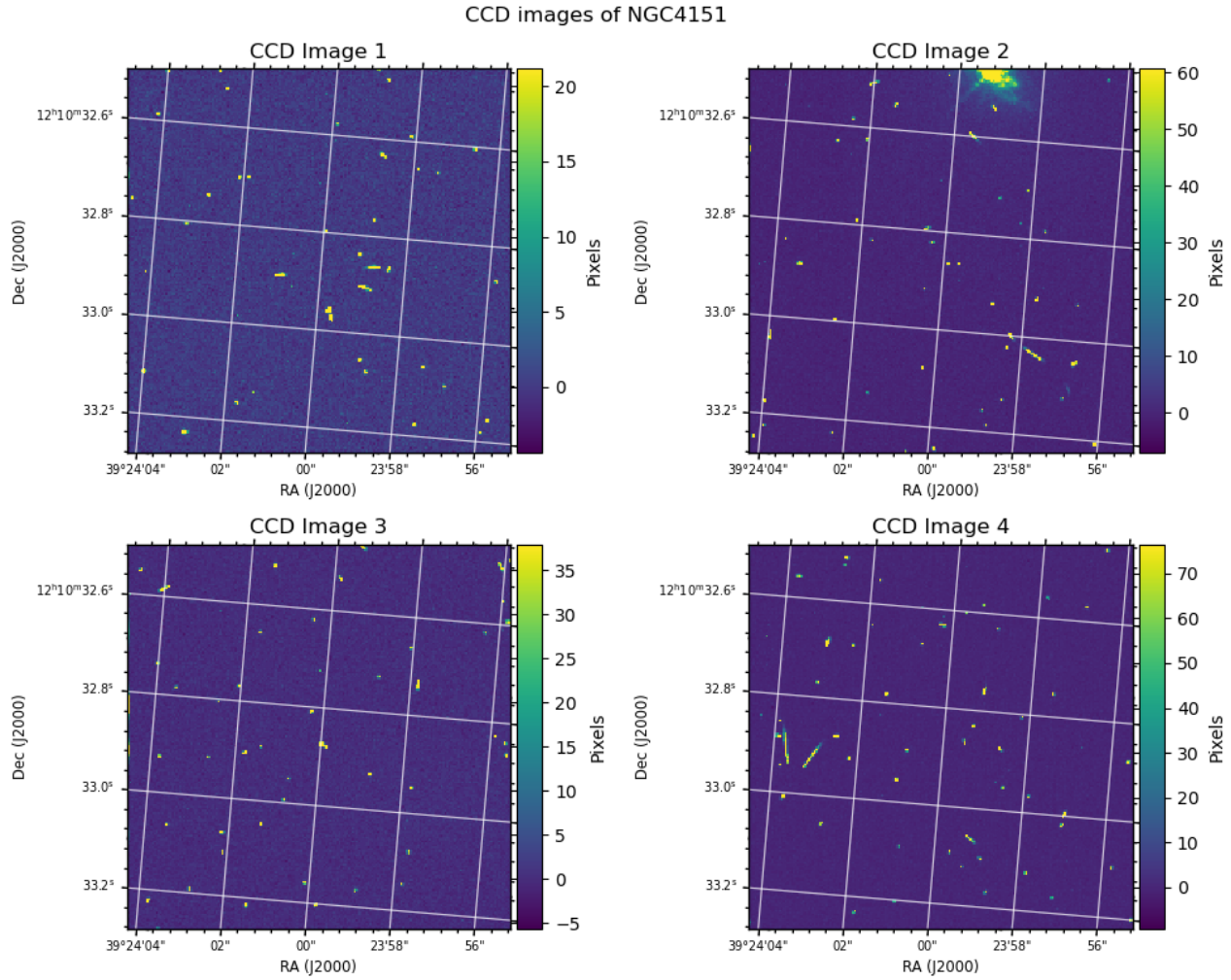

Figure 1: CCD images of NGC4151 taken by NASA. Each subplot shows a different image taken at a different time and under different conditions.

to slice.

Additionally, the `figure` argument allows the user to assign the generated plot to a previously created `matplotlib` figure, and the `subplot` arguments allows the user to generate the plot as a subplot of the figure specified with the `figure` argument. There are several other keyword arguments allowing the user to pass additional commands to `matplotlib`, as well as to modify the header data unit (HDU), rotate the image, or down sample. Plots created with `aplpy` are compatible with `matplotlib` figures and axes, allowing formatting and aesthetics to be modified using `matplotlib` commands.

Figure 1 shows an example plot of a FITS file containing four charge coupled device (CCD) camera images of NGC4151 taken at different times with different conditions. All of the images were provided by NASA. This figure was created using `FITSFigure`, plotting the default zeroth and first dimensions against each other. Because the FITS file was three dimensional, with the third dimension corresponding to different images, the `slices` argument was varied from 0 to 3 to plot a different image in each subplot. Subplots were created using the `figure` and `subplot` keyword arguments. The figure was formatted using commands in `matplotlib`.

4 Additional Comments [Q. 13 - 24]

The `aplpy` package is written entirely in Python. It does however rely heavily on `matplotlib` and `astropy` which both use C to handle certain com-

putations.

The only inputs into aplpy are FITS files, as well as the user specified parameters. If other file types or data formats are to be analyzed this must be done in using other tools, or be converted to FITS files using functions in other packages, such as astropy. The package outputs figures, by creating or modifying figure objects in matplotlib. These figures can then be downloaded or modified using commands in matplotlib. There is no other output type built into aplpy itself.

The aplpy package does not include any built in benchmarking tools, but other benchmarking modules, such as time can be used with aplpy for this purpose. The package does include both regression and unit tests, available in the GitHub directory. Given these tests, as well as the extensive testing and contributions from online users and professionals, I am confident that aplpy will give reliable and accurate results. The package has also been used extensively in publications with reproducible results.

As stated before, aplpy significantly relies on the matplotlib and astropy packages. This is clearly discussed in the documentation and examples, and can be seen when examining the source code in the project GitHub.

The documentation provided by the creators of the project is a series of web pages describing the different functions of the package. This documentation is detailed, and contained almost all information necessary to begin using the package. The site also includes a 'Beginner Tutorial' that was particularly useful when starting out with the package. There is also additional documentation and resources available on GitHub. Despite this, there were certain warnings and error messages encountered in the package that were not clearly discussed in the documentation. While I was still able to produce plots from the data, it was frustrating at times to encounter these errors that were not clearly discussed in the documentation.

When the code is used in publications, the authors of aplpy provide a preferred citation method. They specifically request the use of the following acknowledgement:

"This research made use of APLpy, an open-source plotting package for Python
(Robitaille and Bressert, 2012; Robitaille, 2019)"

The aplpy package has been cited in multiple published research papers before. Both of the references below are published astronomical papers from this year that used aplpy to visualize data.

This package was fairly simple to learn to use. Since it is based on matplotlib, no additional techniques or knowledge were required beyond what we learned in class. Combined with the well written and user friendly documentation, using this package was quite simple. The most difficult part was learning how FITS files work to store data. Once I figured this out using aplpy was effectively just an extension of matplotlib.

This project was completed with help from my peers Sophiya Mehra and Erika Falco. We collaborated on the coding and research portions. The coding example and this written report are entirely my own work.

5 References

1. Weak lensing analysis of A115, A2219 and A2261: Detection of galaxy groups and filaments around clusters:
https://www.researchgate.net/publication/390749559_Weak_lensing_analysis_of_A115_A2219_and_A2261_Detection_of_galaxy_groups_and_filaments_around_clusters

2. SIGNALS on the mixing of oxygen and nitrogen in the spiral galaxy NGC 6946
https://www.researchgate.net/publication/388657704_SIGNALS_on_the_mixing_of_oxygen_and_nitrogen_in_the_spiral_galaxy_NGC_6946