Lorentz center

Python in Astronomy

Workshop: 20 - 24 April 2015, Leiden, the Netherlands

Scientific Organizers

- Pauline Barmby, Western U
- Eric Jeschke, Subaru Telescope
- Sarah Kendrew, U Oxford
- Stuart Mumford, U Sheffield
- Magnus Persson, Leiden U
- Thomas Robitaille, MPIA Heidelberg

Keynote Speakers

- Kelle Cruz, Hunter College
- Perry Greenfield, STScI Baltimore
- Stuart Mumford, U Sheffield
- Erik Tollerud, Yale U
- Britton Smith, U Edinburgh

The Lorentz Center is an international center in the sciences. Its aim is to organize workshops for scientists in an atmosphere that fosters collaborative work, discussions and interactions. For registration see: www.lorentzcenter.nl

The large Hydra constellation is commonly represented by a snake, which is also the mascot of Python programming language, quickly becoming one of the largest and most useful languages for astronomy. Poster design: SuperNova Studios . NL

Universiteit Leiden The Netherlands FOM STW Num FOCUS LCOGT.net GitHub python NWO Netherlands Organisation for Scientific Research Physical Sciences

Lorentz center

www.lorentzcenter.nl

Introduction

The *Python in Astronomy* workshop took place from April 20th to 24th 2015 at the Lorentz Center in Leiden. This aim was to bring together Python developers, users, and educators in Astronomy to share information about state-of-the art Python Astronomy packages, as well as focus on improving interoperability between astronomical Python packages, providing training for new contributors, and developing a common set of educational materials for Python in Astronomy.

In total, 54 participants attended the meeting. The format of the meeting was designed to include presentations in the mornings, and free-form *unconference* sessions in the afternoons. The idea of the unconference time was to allow participants to propose and vote for sessions during the workshop itself. Individual unconference sessions were typically 1h long, and there were usually at least three parallel sessions.

The talks given in the morning sessions have been collected in a Github repository (DOI: http://dx.doi.org/10.5281/zenodo.17666), and the videos have been posted to the Python in Astronomy Youtube channel. The purpose of this document is to complement these resources by providing proceedings for the various unconference sessions. The tone is deliberately informal, and we have tried to include information that will be useful for future such events. The main categories of unconference sessions were tutorials and demos, discussions about future plans for development of packages, discussions on educational resources, discussions on community aspects, and finally coding sprints/hacks.

We hope that you find this document useful – please don't hesitate to get in touch with the organizers if any information is missing or unclear, or if you are interested in getting involved in some of the efforts described here!

Many thanks to the Lorentz center for making this meeting possible, and our sponsors (GitHub, LCOGT, NumFOCUS, and the Python Software Foundation) for providing generous travel support which allowed us to make sure that no one was unable to attend the meeting for financial reasons.

Table of Contents

Tutorials and Demos

Git tutorial (led by Pauline Barmby)

The tutorial went through the basics of using Git on a single computer; we had about 10 people actively typing along, plus a couple of expert helpers. We followed the Software Carpentry lesson (items 1-5) fairly closely and found that this fit well into a little-more-than-an-hour session. (Workshop attendees were much more familiar with the command line than the typical Software Carpentry workshop attendee and that certainly helped speed things up.) We ran into a few glitches with people having trouble setting their preferred editor with 'git config' and getting into 'detached HEAD' state when experimenting with rolling back changes, but neither was fatal; everyone seemed to be comfortable with the basic ideas by the end.

GitHub tutorial (led by Matt Craig)

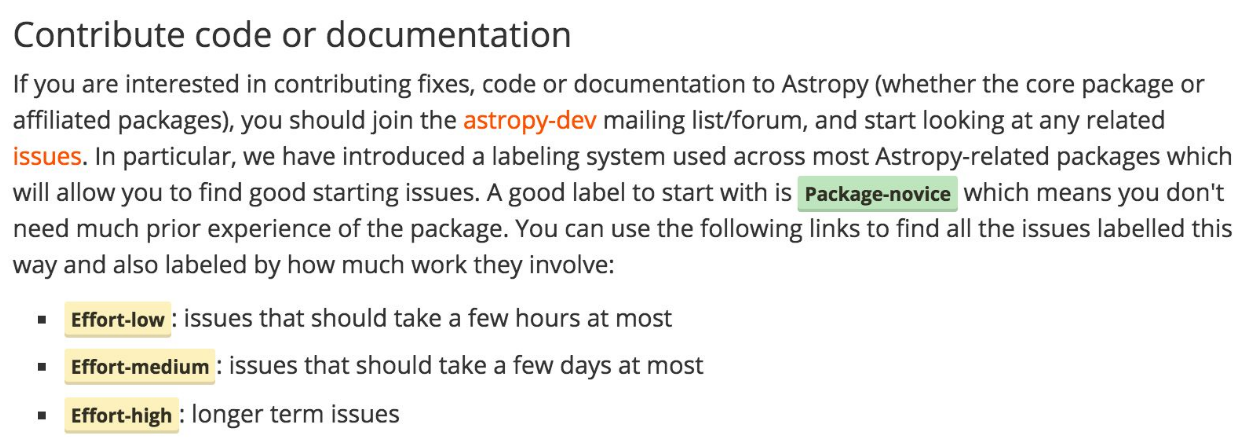
In this tutorial participants forked a "toy" repository on GitHub that was based on the example repository used in the Git tutorial preceding it. Each participant cloned their fork to their local machine, made a new branch, edited a file on the branch, pushed that branch to GitHub, and opened a pull request for their changes against the original repository. The distinction between branches and forks caused quite a bit of confusion; fundamentally a fork is really like a clone, not a branch. We looked at what a repository owner sees on GitHub when a pull request is made and discussed how to update a pull request.

"My first pull request" (led by Tom Robitaille)

The aim of this session was to put into practice the skills learned in the previous two tutorials on Git and GitHub. Experienced developers paired up attendees with existing open issues in various packages of the Astronomy/Python ecosystem. The following attendees opened pull requests as part of this session:

- Christine Koepferl: pull request #130 in Hyperion
- Ricarda Beckmann: pull request #36 in the acknowledgment generator
- Althea Moorhead: pull request #35 in the acknowledgment generator
- Alexa Villaume: pull requests #255 and #257 in APLpy
- Haley Gomez: pull request #256 in APLpy
- Pauline Barmby: pull request #158 in WCSAxes
- Jennifer Karr: pull request #206 in ccdproc
- Mathieu Servillat: pull request #157 in WCSAxes

The most difficult part of introducing people to working on a first pull request is finding 'easy' issues that can be solved relatively quickly, so as to be able to focus on the Git and GitHub workflow rather than worrying about debugging obscure issues. As part of Astropy, we have defined a set of labels for issues that can be used across packages, to indicate issues that are suitable for people not previously familiar with packages, but we had no easy way to search for these issues across many repositories. Following the workshop, we added links on the Astropy website that return a full list of issues that are suitable for beginners:


Contribute code or documentation

If you are interested in contributing fixes, code or documentation to Astropy (whether the core package or affiliated packages), you should join the astropy-dev mailing list/forum, and start looking at any related issues. In particular, we have introduced a labeling system used across most Astropy-related packages which will allow you to find good starting issues. A good label to start with is **Package-novice** which means you don't need much prior experience of the package. You can use the following links to find all the issues labelled this way and also labeled by how much work they involve:

- **Effort-low**: issues that should take a few hours at most
- **Effort-medium**: issues that should take a few days at most
- **Effort-high**: longer term issues

In future similar training sessions, we will point attendees to these lists to make it easier to identify good starting issues.

Intermediate git topics (led by Erik Bray)

Toward the end the introductory Git tutorial on Monday there was some discussion of more intermediate topics that some people wanted to learn more about. In particular, there were a number of people throughout the week who expressed an interest in learning more about the "git rebase" command, both what it does and when/why to use it. About ten of us took time for a more in-depth look at how git branches work, how merging works, and finally how rebase works and how it is different from normal merging. We used the Learn Git Branching tool by Peter Cottle to visualize a git repository and how merging and rebase change the structure of the repository. There were also some moments when even Erik was confused, but was able to correct himself – the important point to come out of that was that although git is not easy, having some idea of how a git repository works enables one to reason about the various git commands in order to dig one's self out of holes.

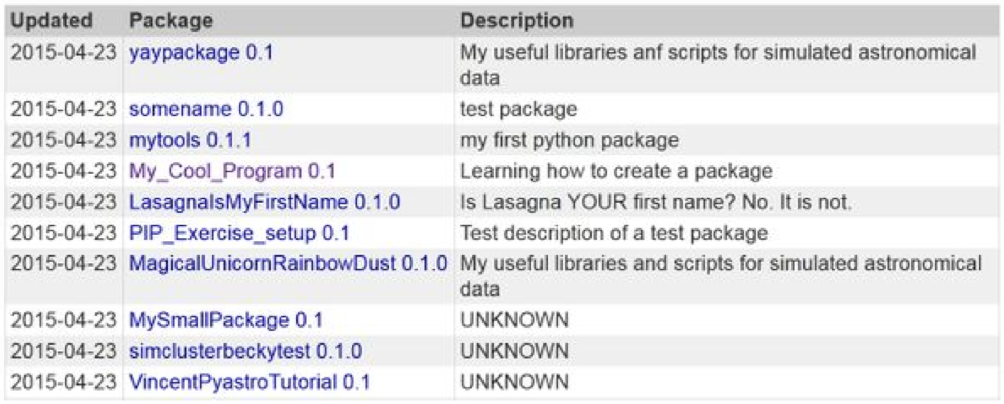
Intro to Python packaging (led by Erik Bray)

Packaging of Python projects is, in the most common cases, not too difficult. However, there remains a fair amount of confusion surrounding the topic, in large part for historic reasons – the best practices for Python packaging have changed in fits and spurts, and it's difficult to keep up and find current documentation. We looked at some of the history and the overloaded vocabulary that makes packaging a confusing subject, and tried to cut through it to narrow in on the current best practices. We also discussed why packaging of Python code can be useful even for personal use, and is not just for making public code releases. We finished by having everyone create their own setup.py script for a piece of sample code provided as part of the tutorial. The only major issue to come up was caused by a typo in one of the slides that lead people to mistype a parameter in their setup.py scripts. However, the distutils documentation contained the correct spelling and most people were able to find the fix on their own. Another issue for some people was not having permissions to install Python packages on their own machines. However, the topic of personal installation of Python packages (via virtualenv or the like) was beyond the scope of this tutorial. Because we only got through about half the slides, we had a follow up tutorial described next.

The slides for this and the next tutorial can be found here.

Python packaging and releasing (led by Erik Bray)

This was a followup to the intro to Python packaging. We went through most of the rest of the prepared slides, including some discussion about how to go about versioning one's software. We had to skip a few of the slides on version numbers for time. We then finished by going over how to create an account on PyPI and register and upload one's package. We used the TestPyPI site which is an exact replica of the real PyPI, but which is wiped clean once a week or so – this is useful both for practice, and for testing one's package before uploading it to the real PyPI. Although a few people had network issues or other problems getting the syntax of the PyPI configuration file just right, several others managed to upload some amusing practice packages:



Updated	Package	Description
2015-04-23	yaypackage 0.1	My useful libraries anf scripts for simulated astronomical data
2015-04-23	somename 0.1.0	test package
2015-04-23	mytools 0.1.1	my first python package
2015-04-23	My_Cool_Program 0.1	Learning how to create a package
2015-04-23	LasagnaIsMyFirstName 0.1.0	Is Lasagna YOUR first name? No. It is not.
2015-04-23	PIP_Exercise_setup 0.1	Test description of a test package
2015-04-23	MagicalUnicornRainbowDust 0.1.0	My useful libraries and scripts for simulated astronomical data
2015-04-23	MySmallPackage 0.1	UNKNOWN
2015-04-23	simclusterbeckytest 0.1.0	UNKNOWN
2015-04-23	VincentPyastroTutorial 0.1	UNKNOWN

This tutorial along with the previous one enabled at least a couple of workshop-goers to upload their real code to PyPI for the first time as well.

Performance tutorial (led by Mike Droettboom)

This tutorial covered:

- coverage.py, which is a tool for measuring code coverage of Python programs (essentially which lines get executed when tests are executed)
- Profiling, which is the process of determining how much resources different parts of a program use, such as CPU, memory, etc.. The tools covered include time, psrecord, cProfile, snakeviz, runsnakerun, kernprof/line_profiler
- Benchmarking over time using asv (or 'airspeed velocity')

Given the time constraints, this was really just a "survey" talk to show what is possible without too much hands-on following along. We did identify a hole in the Python toolset – a memory profiler that understands Numpy arrays.

The slides for this tutorial can be found here.

Getting started with Django (led by Edward Gomez and Mathieu Servillat)

In this tutorial, we first created a virtual environment into which to install Django and then followed the official online tutorial:

https://docs.djangoproject.com/en/1.8/intro/tutorial01/
https://docs.djangoproject.com/en/1.8/intro/tutorial02/

An adapted version of the files in the DjangoProject tutorial are available here:

https://github.com/zemogle/django-pyastro15

One of the main reasons for this session was showing that if you have a project which needs a simple, powerful way to interact with a database in Python, Django is a good framework. We had very little time to cover much of the tutorials but by the end of the session everyone had a virtual environment set up running Django, with a little bit of experience of the built in Admin site.

Like all powerful frameworks, it is highly complex and can be quite daunting for the novice. There is a large and active developer community who are present and responsive on StackOverflow.

Full notes on this session can be found here.

yt tutorial (led by Britton Smith)

This session was devoted to taking first steps with the yt analysis toolkit. All participants succeeded in installing yt, mostly with pip and conda with one person using the official yt install script. Participants tried out some simple analysis on one of the sample datasets available on http://yt-project.org/data/. We first made simple plots such as projections and slices. Most participants were analyzing a simulation of an isolated galaxy. We went through the steps of making a spherical data container, calculating the angular momentum vector, and using that to create a properly oriented disk data container. Finally, we made two-dimensional profiles of the gas mass in bins of density and temperature for the disk. We concluded with a discussion of how to use yt to do exploratory analysis of a simulation whose nature is unknown.

Pycharm tutorial (led by Matt Mechtley)

We went over the basics of PyCharm, which is an integrated development environment (IDE) for Python, including: creating a new project, setting up its Python interpreter, basics of autocomplete and code introspection (and how awesome and vital these features are), viewing function docstrings, git integration, and basics of using the debugger. Matt plans to make a tutorial podcast covering the same material.

Ginga visualization (led by Eric Jeschke)

Ginga is a Python project that provides a high-performance visualization widget for viewing scientific images contained in numpy arrays. This unconference session was primarily for people unfamiliar with the project, or who had specific interests in it. We showed demos of the basic widget as used in some example programs, a short demonstration of the reference viewer (an example “full featured” fits viewer that comes bundled with the package) and Q&A from participants interested in possibly using the widget in their projects or writing a plugin for the reference viewer.

Toyz demo (led by Fred Moolekamp)

We became the first group of users to attempt to install Toyz on their laptops and discovered a few bugs in the installation process. We then looked at a demo of different features of Toyz including features in the image viewer, how to connect to a remote data source and open multiple interactive plots. We then discussed possible use cases and how Toyz can be customized for individual analysis. The video demo from the lightning talk is here, and there are Github repos for Toyz and Astro-Toyz.

Monte Carlo sampling methods (led by Joe Zuntz)

In this session we discussed people’s experiences using a variety of MCMC (and similar) sampling methods. There’s been wide adoption of these methods across astronomy for constraining the parameters of models, and an explosion of different approaches and codes.

We started with the fundamentals of MCMC, looking at the Metropolis Hastings algorithm and the basics of proposals and convergence in single chain methods. I discussed how these samplers are all collected and given a common interface in the CosmoSIS package, which I also talked about later in the week.

We then moved on to more modern sampling methods which are more parallelizable than Metropolis, like the famed emcee, kombine, and nested sampling in (py)multinest and Kyle Barbary's Nestle code. We also learned about the limits of the multinest parallelization, which apparently can be quite wasteful if used with many samplers.

We ended on the related area of maximum likelihood methods, in which a best-fitting single parameter set is found instead of a distribution. There was a strong recommendation to use the Minuit sampler for such problems for its robustness when dealing with numerically difficult problems.

Full notes for this session can be found here and a list of links to packages is here.

PythonTex (led by Stuart Mumford)

PythonTex is a LaTeX package that allows Python code entered within a TeX document to be executed, and the output to be included in the original document. A quick session on PythonTeX expanded on a lightning talk demo that was given earlier during the workshop, but we ran into a lot of installation problems. After some research the conclusion reached was that the easiest way to install PythonTeX is an up-to-date TeXLive installation. The figure helpers Stuart has developed are available as a Python module here.

Development Discussions

Observatory planning and scheduling (led by Eric Jeschke and others)

This aim of this unconference session was to gather ideas for an observation planning and scheduling package to be created as an astropy affiliated package. There are two GSoC students scheduled to work on this and the mentors present (Eric J., Erik T. and Christoph D.) were interested in gathering ideas from the community. There was an interesting discussion about the different use cases for telescope scheduling (we heard from LCOGT, SALT and Subaru) including the need for good performance in calculations since these can make up a significant fraction of the time needed to schedule. We also created a summary of constraints that are used for different use cases. The observation planning side of the package should provide convenience classes that make it easy to produce airmass charts and other common plots and tables that astronomers commonly need when planning a night, as well as performing simple scheduling to optimize efficiency.

The full notes for this session can be found here.

Python packages for working with spectra (led by Wolfgang Kerzendorf)

We first discussed the packages that are out there and what issues they face when dealing with spectra. One of the most common issues was the pixel to physical coordinates transformation (commonly known as WCS, for world coordinate system). The second main issue was that there are currently no replacements for spectral reductions and that specutils does not currently offer much in that direction either.

In the end, we decided to split up efforts for spectral analysis and reduction into three branches: representation and data, reduction, and analysis. Steve Crawford is leading the effort called specreduce that will use specutils for internal objects and will try to recreate some of the IRAF onedspec and twodspec functionality. For this effort there was a desire for a formal release of specutils so that people could easily install this with pip (this is now done with the release of specutils v0.1). Specutils will continue to develop a Spectrum1D object that can hold the necessary metadata for all the different spectral applications (e.g. uncertainties with Neil Crighton, rest-frequency with Adam Ginsburg).

Discussion of astropy.stats (led by Alex Hagen)

We first discussed how the Astropy community wanted to proceed with astropy.stats, and then discussed some statistical techniques we'd like to see added to the package. This discussion led to multiple pull requests and more development.

Those at the session talked about a bit of the tension with astropy.stats, that is, that many statistical techniques used by astronomers are not unique to astronomy, and thus some live in astropy.stats, some in scipy.stats, pandas, and other places. The community consensus was that if there's a statistical technique someone in astronomy wants to add, it can be first added to astropy.stats, and then moved upstream (for example to SciPy) if needed. Astropy is also easier to commit to than SciPy, and so this approach is welcoming, and allows for the fastest and easiest way for folks to commit new statistics code. We also discussed how larger packages, such as emcee, are very useful for some statistical analysis, but anything that large should be an affiliated package and not in astropy.stats.

We next discussed some good resources and ideas for functions to add to astropy.stats:

Good resources:

- interactive lecture notes: https://github.com/dhuppenkothen/ClassicalStatsPython
- Tutorials from astro stats course: https://github.com/mwcraig/astropy-tutorials.git

Desired functionality:

- Circular/spherical distributions (+2, at least) (very basic available in scipy.stats.morestats) – issue in astropy
- Bayesian blocks (pull request #3756 opened in Astropy since the workshop)
- Dealing with correlated errors and covariances – issue #3709 in Astropy
- (Straight) Line fitting with all the different possible error inputs (x,y,correlated,etc) – issue #3710 in Astropy
- Tutorials
- information criteria (way to penalize your likelihood as you add more parameters) – pull request #3700 opened in Astropy
- Poisson error bars (beyond just taking square root) – pull request #3717 opened in Astropy
- Wrappers/translations from stats functions to functions astronomers understand
- Comparisons of N-th dimensional distributions – issue #3707 in Astropy
- Goodness-of-fit measures (overlaps with modeling in astropy, e.g. bootstrap/jackknife test) – issue #3708 in Astropy
- Tools for working with cash statistics – issue #3711 in Astropy
- More robust image statistics (or at least a tutorial/improved documentation)

This discussion prompted multiple astropy issues and pull requests, and thus astropy.stats should have significant additional functionality and techniques very soon.

Regions and shapes (led by Tom Aldcroft and Perry Greenfield)

There was discussion about the various facets of the issue of dealing with regions on the sky. In the end these facets appeared to fall in one of the following categories:

1) being able to read and write existing disk formats or serializations for region definitions.
2) representing all the necessary information about regions in python objects (to some extent, this is a union of all the use case needs)
3) determining what operations should be possible on regions.

One issue was determining whether all could be done by the same package. To alleviate stress from the suspense of waiting, the short answer is that it doesn't appear that 3) can be well handled by one package. The reason is essentially set by the somewhat different goals used for regions. Two major use cases are:

1. Aperture photometry or other similar extraction needs. Typically these involve comparatively small areas and often rely on simple shapes (circles, rectangles, ellipses and annuli thereof). Very often the location of the center is given in world coordinates (though it doesn't have to be), but the computation of what pixels are involved is almost always done in cartesian pixel coordinates. Speed is essential since this kind of masking or extraction may be done on thousands of areas.

2. Determination of the boundaries of larger areas on the sky, e.g., footprints and similar. In this case the drawbacks of using cartesian coordinate assumptions are often significant, and it often best to do such computations in an intrinsically spherical coordinate system that are not subject to the problems presented by the poles of the celestial coordinate system used. But unfortunately, doing this way is much slower and does not lend itself to the needs of aperture photometry.

The following covers in more detail some of the topics discussed under the first three categories:

1. Dealing with various formats. DS9 region files are probably the most common, but not the only format available. Other cases included VO region definitions (it's unclear how widely this used, but apparently Aladin uses it). CASA has its own region format as does CIAO. There is often associated metadata for regions such as labels, coordinate system, etc, that needs to be retained.

2. Generic python objects. The need for two different computational approaches suggests that two different kinds of objects are needed, but it is quite likely they can inherit from a base class that supports common operations for both (e.g., is a point inside a region is useful in both cases)

3. Supported operations. For aperture photometry, the need is mostly to determine which pixels are within the specified region. This is essentially a mask generation operation. The question was raised as to whether only binary masks were needed or was dealing with fractional pixels needed as well. Answer: fractional pixels perhaps eventually, but probably not initially. There may need to be a few different options for determining pixel

membership. Some did indicate that being able to perform union and intersection of regions for aperture photometry would be useful in some cases, but generally only very simple cases would be needed (and perhaps could be satisfied by a bit more complex shapes, i.e.., fractional circles and annuli).

For footprints, besides membership, the determination of more complex footprints from the combination of simpler ones is required. This can result in multiple disconnected regions so this case must be handled.

Existing software:

All of the following handle regions in some way:

- DS9
- Ginga *
- photutils *
- SDSS mangle
- funtools
- pyregion *
- IRAF X-ray package
- CASA
- CIAO
- shapely *
- spherical-geometry *

Those marked with * are easily used by Python. It was mentioned that pyregion parses DS9 region files quite slowly and this probably needs to be improved. It is quite possible that the best solution to this is to use the most appropriate parts of pyregion, ginga, shapely, photutils and spherical-geometry, but that needs some investigation. The work can be partitioned somewhat into the I/O aspect, base class design, and the specialized classes. There was hope that at least a tentative API could be defined by the end of the workshop. Some discussions continued after the workshop and resulted in the creation of a package to experiment with an API that covers all use cases.

FITS alternatives/replacements (led by Mike Droettboom)

The basic outline of the new file format being developed at STScI, Advanced Scientific Data Format (ASDF) was shared. Many of the questions from the audience involved whether certain uses cases were met by the new format, mostly in the positive. There was a core contingent of participants who are currently using HDF5 for reasons of performance whose needs may not be met by ASDF. It is unclear if these can be resolved, since the primary concerns of ASDF – transparency and expressiveness – may be at odds with those of HDF5 – flexibility and performance. A later offline investigation was performed to see if ASDF's structures could be represented directly in HDF5 which turned out to be not terribly promising.

The full notes for this session can be found here.

Generalized world coordinate systems (led by Nadia Dencheva and James Turner)

In this session we described the GWCS (Generalized World Coordinate System) package, developed at STScI. Nadia explained a bit how the transforms and co-ordinate frames are used and chained together and some recent changes in the modelling syntax etc. There was a discussion of use cases and a request to work out and add a couple of examples to the documentation:

1. Given two images, one with known WCS and one with incorrect or unknown WCS, create a valid WCS for the second one by fitting (x, y) positions in it to (ra, dec) in the first one.

2. Given a WCS with mixed axes, e.g. (RA, Lambda, DEC) make sure the inverse works correctly on WCS.invert(SkyCoord, Spectral) input.

There was also a discussion of how to use GWCS with FITS files and FITS WCS. We decided to use astropy.wcs to read the FITS WCS from the header and use the all_pix2world() and similar methods as forward and inverse transformations in GWCS. There's a PR for this here. Work on GWCS serialization is still in progress, as the intent is to build on the work done for ASDF but at the moment Nadia is still using a placeholder JSON serialization.

In a related session we discussed the coordinate frames in GWCS. We decided that

- gwcs.coordinate_frames will be renamed to gwcs.axes with CelestialAxes, SpectralAxes and so on. They will be informational containers. SpectralAxes will hold a reference to the associated CelestialAxes.reference_frame to facilitate changing reference position when the SpectralAxes is of type Velocity.
- All conversions between standard systems will be done in astropy.coordinates.

There was general enthusiasm to get started using GWCS, eg. in specutils, and Mike Droettboom already has a hack to integrate it with Matplotlib.

During the week James Turner additionally started hacking out a simple example of CCD mosaicking with GWCS and ndimage. Due to performance optimizations without which the code is unusable, it's currently a bit too convoluted to go in the main documentation. In the process it was noted that the order of the axis mappings between GWCS & ndimage can get a bit confusing.

The full notes for this session can be found here.

Photometry in python (led by Matt Craig)

There are currently (at least) two python-based photometry packages: photutils, an astropy-affiliated package which implements some of the photometry functionality of IRAF, and SEP, a python wrapper around the internals of the Source Extractor.

Photutils implements daofind-like source detection, aperture photometry, and image segmentation. Although rudimentary PSF photometry is included it needs additional work. SEP is does exactly what Source Extractor does: background generation, source detection and aperture photometry with some source deblending, all of it very fast.

Participants agreed that PSF photometry, local background subtraction in aperture photometry, source deblending and the addition of convenience functions (analogous to apphot) are the top priorities for future development. Development of SEP as a separate package will be frozen, with a slow integration of SEP into photutils as an option for performing the photometry at a low-level. A higher-level interface to SEP needs to be added to photutils; Matt Craig and Kyle Barbary agreed to work on this. Curtis McCully and Christoph Deil agreed to work on PSF photometry in photutils.

Existing comparisons between photutils, SEP and SExtractor need to be highlighted on the Web site, and additional comparisons, to, e.g. IRAF, should be added and perhaps eventually developed into a paper.

Photutils and SEP should be better advertised. Jennifer Karr and Brigitta Sipocz will work on tutorials for photutils for inclusion in astropy-tutorials.

Two other unconference sessions covered topics very relevant to this one: astropy modeling and MCMC galfits. See the latter for a link to the google group created to discuss galaxy fitting.

The full notes for this session can be found here, and the Twitter hashtag was #pyastrophot.

New reprojection module (led by Tom Robitaille)

The reproject package is a proposed Astropy affiliated package that aims to deal with many different kinds of image reprojection, where reprojection means that we want to transform data in a given world coordinate system and projection into a different world coordinate system (e.g. what Montage, SWARP, and drizzle do). We brainstormed on use cases that should be covered by this module, which are reprojection of:

- Celestial 2-d image
- Non-celestial 2-d image
- N-dimensional cube with celestial axes
- N-dimensional cubes with no celestial axes

- HEALPIX data
- X-ray event lists

In addition, we discussed the algorithms that are or should be available:

- Interpolation
- Flux-conserving Montage algorithm
- Drizzle

A few people started to sprint on various aspects, including testing out the package and identifying things that do not work correctly, Later in the workshop, there were discussions of how we can build a mosaicking engine around the reproject functionality (including determining optimal headers for mosaicking, and co-adding the images). A few weeks after the workshop, the first version (v0.1) was released on PyPI.

Astropy modeling (led by Erik Tollerud and Erik Bray)

There was both some confusion about what the astropy.modeling package is for (from those who haven't used it much) and interest by those who have used, or at least looked at it in discussing what directions it should go in. It was apparent shortly into the discussion that we should have had a prior tutorial (pun unintended) to explain and demonstrate the existing capabilities of the package. Those of us who work on it or have used it tried to summarize, by explaining that it is really two things: A framework for describing manipulating mathematical models in an object-oriented fashion in Python. The common interface between these models then enables writing tools that work on generic models. The other aspect of the astropy.modeling package is a collection of "fitters" for fitting models to data–demonstrating, in principle, how the model interface makes the process of defining models independent of individual fitting algorithms or other statistical analyses.

One concern that was immediately raised by Christoph Deil, and possibly others, is that is vague what is meant by "fitter" in this context. We explained that a fitter in astropy.modeling can consist of both an optimization algorithm and a statistic function to optimize. Christoph raised the point that the fitting process should include determination of goodness of fit, and that there should be a place for that in the fitting framework. This was uncontroversial, though the details were not hashed out.

Also of interest was that many of the attendees were interested in Bayesian analysis–in their words the current capabilities of the astropy.modeling fitting framework only allow "traditional" frequentist techniques–maximum likelihood estimation. We discussed what would be needed to enable integration of Bayesian fitting techniques, and agreed that the most critical missing feature was the ability to assign priors to all the parameters on a model. This wasn't discussed in detail, but presumably priors could be assigned either as some (normalized) array from an external source, or in principle as some distribution which would itself be described by astropy.modeling Model object with its own hyperparameters. There should be no huge technical barriers to integrating this into the modeling framework. The only major barrier is that no one currently active in development has expertise in Bayesian analysis, so some further community involvement would be desirable to get this work done.

The third major topic was how astropy.modeling should interact with (or if it should even be replaced by) the modeling and fitting package Sherpa, which was originally developed as part of Chandra's software suite, but has recently been made available under an open source license. In fact, much of astropy.modeling's early design was inspired by Sherpa. It was the opinion of Tom Aldcroft, who had the most experience in the room with Sherpa, that its model definition framework may be less flexible than Astropy's for integrating with other fitting code–that it is too tightly coupled to Sherpa's optimizers to be as "generically" useful as Astropy models. However, there are many useful models in Sherpa that should be made available in Astropy, and it is also possible to adapt arbitrary Astropy models to Sherpa via Sherpa's custom model interface. We decided found that there are many useful features in Sherpa's optimizers that could be made available through astropy.modeling, with Sherpa as an optional dependency.

We also discussed several other statistical fitting packages such as PyMC, PyMultinest, jbopt, and possibly others. The overall summary to this discussion was that astropy.modeling should best be viewed as a "glue" between various disparate optimization packages and routines, enabling a common interface between them (while still allowing access to the low-level details of individual codes). A few other desired features for astropy.modeling were raised, including end-to-end handling of errors, and support for units, but we did not have time to go into detail on the specifics of those features, except to say that Erik Bray is working on an initial prototype of units support.

Astropy: where next? (led by Tom Robitaille)

In this session, which included both users and developers, we brainstormed on future improvements and features for the core Astropy package. Ideas raised during this session include:

- An easier (one-line) way to match/merge catalogs. Matching two catalogs is already possible, but takes several lines of code, and given that this is a common use-case, several users and developers agreed that having a one-line way to do this would be much better. A working group consisting of Geert Barentsen, Becky Smethurst, and Christoph Deil was formed to work on a possible API and implementation.
- A way to register images - that is, if an image has WCS information and another has incorrect or missing WCS information, we should have a way to fit the correct WCS to the second image so as to then be able to align the images later. The reproject affiliated package is able to reproject two images with different WCS to a common WCS assuming the WCS is valid for both images, but we are missing the ability to fit WCS to images in the first place. This will be made much easier with the new generalized world coordinate system (GWCS) framework being developed by Nadia Dencheva and others at STScI, so we agreed

that getting GWCS in Astropy 1.1 would be ideal. Matt Mechtley and Tom Robitaille can look into this with help from STScI.

- Improvements to performance in several places, and in particular table I/O and manipulation for large files. Since every issue may be different, users who run into performance limitations should open dedicated issues on GitHub, outline their use cases, and provide example files and code.
- Several users also requested the ability to be able to interface to databases (e.g. SQL) from the Table class. There are two levels of interfacing here - one is to simply be able to read and write from databases (as was possible in ATpy, but this code has not been ported to Astropy), and the other is to actually make Table objects a 'live' view of the database. The latter will likely be complicated to implement, but the former should be feasible (either using code from ATpy or using pandas or sqlalchemy). A working group was formed, consisting of Geert Barentsen, Carolin Villforth, Joe Zuntz, Tom Aldcroft, and Tom Robitaille.

Python and galaxies (led by Steve Crawford)

Through the conference, a number of packages related to galaxies were presented including yt (interface for galaxy simulations), barak (fitting absorption line profiles), and starpy (determination of star formation histories of galaxies using MCMC techniques). During this session, we presented other packages that we are using, tools that we have developed, and brainstormed what further tools we would find useful.

Much of the discussion revolved around tools for performing analysis of galaxy profiles. This included different ways of fitting galaxies (see Matt Mechtley's discussion of MCMC fitting below), wrappers for existing tools (Alex Hagen's galfit python parser, Kyle Barbary's source extraction and photometry library SEP), and existing codes still in development (Benne Holwerda's morphometric software for quantitative morphologic measurements, Pauline Barmby's imagecube for matching images across different wavebands). One thing that generated the most interest from people were interactive tools to be used to improve galaxy photometric and kinematic fitting, where a user can create custom masks for the data (see Eric Jeschke 'interactive bad pixel masking' hack session).

In addition we also discussed ways to bridge the gap between observers and theorists using yt. For the most part, the observers were interested in ways to extract simulated observations from the models (HI or CO maps, realistic spectra, data cubes, or images). Additional feature requests included easy ways to find sources in the simulations that matched an observed object and possible integration into galaxy zoo projects.

The full notes for this session can be found here.

MCMC galfits (led by Matt Mechtley)

There are many disparate packages right now trying to do the same thing: 2D surface brightness modeling (like Galfit) but with MCMC sampling. We discussed how we might better combine efforts rather than everyone rolling their own. There are several parts to the problem (creating the SB models, PSF convolution, hooks for the various MCMC samplers, analysis after sampling is complete), but we discussed how we might be able to move much of this to astropy.modeling (SB models) and photutils (PSF/convolution). We created a Google group for further discussion and planning.

Educational Resources

Astro-Python educational resources (led by Kelle Cruz and Pauline Barmby)

There were two major threads in this discussion: how to format new educational material, and how to make existing material more discoverable and shareable. For new material, both Jupyter notebooks and GitHub repositories (Software Carpentry has a template) are both good options. For discovering existing materials, it would be ideal to have a database of links plus suggested curricula for different needs (few-day workshops, week-long summer schools, full-semester courses) and levels. The Open Astrophysics Bookshelf approach is one possible way to go with this.

Major outcomes:

- started a Google group for discussions
- also started compiling resources on an AstroBetter wiki

The full notes for this session can be found here.

Astropy tutorials (led by Kelle Cruz)

astropy-tutorials is a repository of tutorials which demonstrate the functionality of both Python and and the astro-specific packages. The rendered tutorials are visible at http://tutorials.astropy.org and the github repository is https://github.com/astropy/astropy-tutorials.

To suggest/request a new tutorials, please make a new issue or comment on an existing one: https://github.com/astropy/astropy-tutorials/issues

The discussion and work on new tutorials began in the unconference session, but work was done over several days. One new tutorial was contributed and two were started:

- Cosmology and plot with redshift and universe age axis
- Photutils
- ccdproc

Based on these interactions, more guidelines to authors were added to the Contributing document.

The full notes for this session can be found here.

astropython.org reboot (led by Tom Aldcroft)

The astropython.org site was launched 5 years ago with the goal of becoming the main community portal for Python in Astronomy. In one sense it has been successful because the site is one of the top two generic informational / resource sites about Python in astronomy. But in another way it has fallen short because there is little community involvement.

This site uses Google App Engine and is basically all custom code built around the bloggart engine. Issues with the site include:

- Guest posting is very difficult.
- Even for current owners, adding content is unpleasant.
- The comment system is lacking (no feedback to comment authors etc).
- The website code itself is convoluted and difficult to maintain / improve.

The plan is to start over with Django and modern web tools to bring fresh energy and community involvement into this project. This will be done by an Astropy Google Summer of Code student, with Tom Aldcroft as primary mentor. The most important difference will be that any authenticated user (google, facebook, twitter) will be able to post (with moderation as needed). A key goal will be retaining high signal to noise in terms of content.

During the unconference session we brainstormed ideas for reorganizing and renaming the top-level content and settled on:

- Forum (News, community, opinion)
- Packages and tools
- Teach and learn (tutorials, education resources, code snippets)

In addition to discovery via content tagging, we talked about dividing content (especially packages) by a few predefined categories.

We decided that a markdown based system for content input would be the baseline, with a preference for github-flavored markdown. Other suggestions in included having topic upvotes, auto-tweeting, and email group subscriptions.

Making documentation easy to write (led by Stuart Mumford)

The discussion focused on the challenges faced by new users when writing documentation, specifically with the sphinx package. We spent a long time explaining the architecture of sphinx and how it leads to some complex error messages, however, as we discussed sphinx is a very powerful package and is easy to use, when it works.

The main conclusion from the session was that the Astropy docs on writing documentation are very comprehensive, but not widely known. There should also be an effort to expand the documentation on common errors, or methods for debugging problems.

The full notes for this session can be found here.

Community Discussions

Running a local user group (led by Althea Moorhead and Adam Ginsburg)

We discussed issues faced by groups trying to arrange weekly (or otherwise regular) meetings of python / coding enthusiasts at science institutes. Many groups (CfA, ESO, NASA Marshall) started with initial enthusiasm but ran out of steam after a few meetings. By contrast, the hackNY group has weekly 3 hour-1 day "meetings" of grad students teaching and advising each other; this group apparently faces no problems of participation or interest.

One solution is the newly created e-mail list to provide fresh material for discussion each week. The updated http://astropython.org site is also expected to provide a partial solution by providing lists of new packages and important updates to major packages that can provide a starting point for discussion.

The full notes for this session can be found here.

Credit for software development (led by Erik Tollerud)

This discussion was about trying to determine what steps should be taken to try to improve how credit is given for software work in astronomy. The discussion was focused particularly for projects like Astropy or affiliated packages, where many of the contributors are on traditional academic research tracks where committees often don't consider software work as "real" work.

There was a fair amount of discussion and exploration of tools for making metrics to measure software contribution that might be useful for inclusion on a CV or even used by search committees. An example examined in some detail was OpenHub, which attempts to quantify the effort involved in open source packages (for example, it reports astropy as being worth approximately $1.5M in developer time, for 27 person-years of effort). Some of the attendees agreed to try to work on designing metrics that would follow an approach like this to quantify the impact of such projects on astronomy, as well as determining individuals' contributions. Other discussion centered around how many metrics provide ways to "game the system." Further discussion considered exactly what software contributions should be treated as - for example, a Pull Request could be thought of as a (small-scale) peer-reviewed publication.

Further discussion considered ways to co-opt the existing system to give credit to contributors. Providing awards through the RAS or AAS might make it clearer that these contributions have real value. Including better names that could go on CVs for important roles in larger projects like Sunpy and Astropy was also discussed. Tools like Zenodo or the ASCL provide ways to immediately use the existing citation system to provide more traditional forms of credit for software work. At that point we realized that a more "high-powered" group had had a meeting and published a report on the importance of mechanisms for citing software work. General consensus seemed to be that developing better metrics could be a valuable outcome, but more information is needed on what the “average” astronomer (i.e., those not plugged into the astro software community) would actually consider as a valid metric.

The full notes for this session can be found here, and the Twitter hashtag was #pyastrocred

Roles/titles org chart (led by Kelle Cruz)

As a spin-off from the discussion on credit, one of the ways of improving how we give credit to people in the community we identified is to define roles more clearly, and assign meaningful titles to them. There are many roles needed in a typical organization such as Astropy (lead developers of sub-packages or affiliated packages, being responsible for tutorials, documentation, outreach, interactions on social media, and so on), and by making these roles explicit, we can recognize that they are all are essential to a functional community and project.

When applying for jobs, it’s important in many cases to show examples of leadership, and there are many places in projects like Astropy where one can assume a leadership position, and this should be highlighted. For instance, saying ‘I was the lead developer of astropy.stats’ or ‘I am the coordinator for tutorial content for the Astropy project’ is more meaningful than ‘I am part of the Astropy project’ or ‘I contributed to Astropy’.

The bottom line from this discussion was that each project (for example Astropy, SunPy, yt, etc.) needs to come up with a list of all the different roles one can take in the project, and this will help identify not only places where no one is clearly responsible for something, but also places where some people might be over-committed and looking for someone else to take over.

Career tracks discussion (led by Erik Tollerud)

We discussed several aspects relating to career tracks and strategy for members of the Python Astronomy community, both relating to how people contributing to this community can maximize their chances in the traditional academic track (postdocs, tenure-track) and also in terms of defining their own career track if the traditional career track does not suit them. The main points and conclusions from this discussion session were:

- It is critically important for us all to recognize the existence and importance of multiple career tracks. While many in the community in traditional academic positions see that as the only road, there are other ways to apply the experiences from this community in non-traditional astronomy positions or positions outside the field entirely.
- One of the main benefits of being a member of our community is that it is an invaluable network to have. Networking is (for better or worse) a hugely important factor for success when applying for any academic or non-academic job, and through the open source

development community, you will get to know people at many different institutions across the world and therefore it's very likely you may meet people at an institution where you would like to go. Even if the person you know through this network does not work in your field of research, they can still serve as an entry point into their institution–for instance they could have you visit to work on Python development and then arrange a research talk so that you can then meet other members of the institution. In addition, knowing people outside your field of research is very important because most people on a hiring committee may not be in your field of research, and being part of the open-source development community, which is intrinsically multidisciplinary, is a great way to make valuable contacts.

- For non-traditional career tracks in or outside of Astronomy, it is also important to realize that people within the astronomy software community will be able to help you by writing reference letters, LinkedIn recommendations and endorsements, and so on. If you contribute to projects like Astropy, you should ask lead developers or coordination committee members whether they would be able to write letters for you.
- For career tracks in software, in and out of Astronomy, it is important to understand the different types of development that employers may be looking for - in some cases it might be important for you to have very detailed knowledge of certain technologies, while in other cases broader and shallower knowledge would be desirable. In any case, it is important to show potential employers that you do not know just Python, but also lower level languages (e.g. C), and that you are able to learn new languages/technologies efficiently.
- For Astronomy software careers, it is important to know when a job is more focused on the astronomy or algorithms, and when it is more "software-y". I.e., is it more about developing new approaches to solving scientific problems, more about developing solid software, or a mix of both? Different positions will have different expectations in this regard, and a candidate who understands which way a position goes will have a significant advantage.
- People interested in career tracks outside of Astronomy should make sure to set up a LinkedIn profile as soon as possible so as to grow their 'online CV' and network of connections.
- We need to ensure that we invite to conferences and workshops people who represent a diversity of career tracks. This will provide a diversity of role models, expand our network, and keep everyone aware of the career opportunities available to us. We also must ensure that members of our community following these tracks are not excluded from the community. They provide important opportunities both for us to learn and to provide context and alternative perspectives.

Code of Conduct (led by Kelle Cruz)

During the workshop, prompted by some discussions and interactions on social media, to create a code of conduct for the Python Astronomy developer and user community, in order to make our community as inclusive and welcoming as possible. This code of conduct is being adopted by the Astropy project, and the intent is that other Python projects in Astronomy, and other forums (such as the newly formed Python users in Astronomy Facebook group) can adopt this too. The writing of the code of conduct was a nice exercise in collaborative writing during the workshop. The code of conduct, as adopted by the Astropy project, can be found here.

Sprints/Hacks

A number of sessions described in previous sections including coding, but there were also dedicated sprinting/hacking sessions, a few of which are described below:

Astroquery hack session (led by Adam Ginsburg)

Many people were interested in getting new features into astroquery. Pull requests have been issued:

The online Astrometry.net
The LCOGT archive
Performance improvement in Vizier (not completed, but new options are available)
The HEASARC archive

In addition, some important fixes to the template module were made.

Ginga collaboration sprint (led by Eric Jeschke)

This was an unconference session to collaborate and sprint on some projects using ginga. Megan Sosey got her imexam package updated to work with the latest version of ginga. Stuart Mumford optimized the speed of single point coordinate conversions when tracking the mouse across the image in the ginga reference viewer and astropy is performing the WCS lookups. He also got sunpy coordinates handled correctly when solar images are being displayed. Mike Droettboom got a plugin working to browse ASDF files and load them into the reference viewer.

Interactive bad pixel masking (led by Eric Jeschke)

This unconference session was about designing a plugin for the ginga reference viewer to create bad pixel masks. This was driven by interest from Alexa Villaume and Peter Teuben. Alexa got up to speed on plugin development in ginga and got a basic plugin working that draws circles in images and then obtains information about the areas drawn.