Python in Astronomy 2016
(un)Proceedings

Tom Robitaille, Kelle Cruz, Perry Greenfield, Eric Jeschke, Mario Juric, Stuart Mumford, Chanda Prescod-Weinstein, Megan Sosey, Erik Tollerud, Jake VanderPlas, Jes Ford, Dan Foreman-Mackey, Tim Jenness, Tom Aldcroft, Mike Alexandersen, Michele Bannister, Kyle Barbary, Geert Barentsen, Samuel Bennett, Médéric Boquien, Jose Ivan Campos Rozo, Steven Christe, Lia Corrales, Matthew Craig, Christoph Deil, Nadia Dencheva, Axel Donath, Stephanie Douglas, Leonardo Ferreira, Adam Ginsburg, Nathan Goldbaum, Karl Gordon, Andrew Hearin, Cameron Hummels, Daniela Huppenkothen, Elise Jennings, Johannes King, Samantha Lawler, Andrew Leonard, Pey Lian Lim, Lisa McBride, Brett Morris, Carolina Nunez, Russell Owen, John Parejko, Ekta Patel, Adrian Price-Whelan, Rafael Ruggiero, Brigitta Sipocz, Abigail Stevens, James Turner, Sarah Tuttle, Petia Yanchulova Merica-Jones, Peter Yoachim

The Python in Astronomy 2016 workshop took place from March 21st to 25th 2016 at the University of Washington eScience Institute in Seattle. This aim was to bring together Python developers, users, and educators 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 as well as free-form unconference sessions, and two days dedicated to tutorials, sprints, and hacks. 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 one hour long, and there were usually at least three parallel sessions.

The talks given in the morning sessions have been collected on Zenodo, and the videos have been posted to the Python in Astronomy 2016 YouTube channel. The purpose of the present document is to complement these resources by providing proceedings for the various unconference sessions as well as summarize the different hacks. 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 University of Washington eScience Institute for hosting this meeting, and our sponsors (the Large Synoptic Survey Telescope, NumFOCUS, and the Python Software Foundation) for providing generous travel support.

Unconference Discussions

How to switch to Python 3 (led by Tom Robitaille)

We started off by discussing the main changes in Python 3 that will affect the typical user. These include: changes to the print statement which is now a function, division which now returns a floating point value when dividing two integers (rather than rounding to the nearest integer), strings which are now unicode by default (which allows non-English-centric characters and importantly emoji), and for some users, the fact that some functions now return generators instead of lists (e.g., zip, range, and map). While there was a lot of work for package developers (especially with C extensions) to support Python 3 due to other changes, the transition should be relatively straightforward for the typical user. At the very least, Python 2 users should already always include:

```
from __future__ import print_function, division
```

at the top of scripts which will allow the print function to already be used, making the transition a lot easier in future.

We then also discussed ways of transitioning to Python 3. For users that rely on Anaconda/conda to manage their Python distribution, it is possible to create a Python 3 environment and switch to it then switch back. In this way, the transition can be done progressively – new scripts can be written in Python 3, and users can switch back to Python 2 for older scripts. Alternatively, users can use the environments to make sure that scripts work in both Python 2 and 3. To translate Python 2 code into Python 3 code, users can make use of the 2to3 tool (installed by default with Python) – however this produces Python 3-only code. We also discussed the six and python-future packages, which can be used to make sure that functions behave the same in Python 2 and 3, and the modernize tool, which can be used to convert Python 2 code to Python 2 and 3-compatible code.

Later in the week, we decided to work on a page summarizing the nice new and useful features in Python 3 and providing a migration guide for users (see the **Hacks/Sprints** section)

Analyzing simulated LSST surveys (led by Peter Yoachim)

We looked at MAF, the tool developed by LSST to analyze the scientific performance of simulated and real surveys. There is a public repository where members of the scientific community can contribute new metrics to help ensure LSST can be optimized for their specific science case. We discussed the difficult problem of how well LSST can perform on Solar System science, and fielded general questions about the LSST and its data management pipeline.

Funding for community software development (led by Kelle Cruz)

This discussion was about identifying funding avenues for community software development. We specifically discussed strategies for obtaining grants from US government agencies such as NASA and NSF (e.g., SSI, SSE) and private sources such as the Sloan Foundation, the Moore Foundation, and the Simons Foundation. Participants shared their experiences and relevant information gathered from program officers and grant panels. The organizational needs for accepting grants and the role NumFOCUS plays for Astropy was also discussed

(they currently have an agreement to hold the finances for the Astropy project in trust). A specific action item that came out of this discussion is to form an OpenAstronomy committee to seek out funding sources for community software development.

Image regions (led by Adam Ginsburg)

We discussed the need for a new regions / shapes library within the Astropy project. We came up with a plan to replace pyregion as the region library for Astropy. While pyregion has some useful features, it has an API very different from the rest of Astropy packages and a code structure inconsistent with other python packages. We therefore began development of the new regions package, which will go through a development and beta testing phase as an affiliated package before being incorporated into the core Astropy package.

During the week, we implemented a basic API for region objects within Astropy and built a Ds9 region file parser that is more complete than pyregion. Thanks to help from the Ds9 development team (Bill Joye & Eric Mandel), we have a reasonably complete library of Ds9 regions against which we can test the parser. The hack team involved Adam Ginsburg, Tom Robitaille, and Johannes King. Meanwhile, Eric Jeschke, Tom Robitaille, and Pey Lian Lim.

PythonTex: Python inside LaTeX (led by Stuart Mumford)

Tools for embedding Python code inside LaTeX documents, primarily PythonTex, were discussed. The advantages of such software, such as the ability to have code which generates plots inside the document, and the ability to have parameters from the analysis automatically update in the text were presented. Some of the drawbacks such as long compile times, exporting plain LaTeX for journal submissions, and interoperability with other tools were put forward and ways to mitigate these were discussed.

The texfigure package was demoed, features that enable easier embedding of plots from various Python libraries such as Matplotlib, Mayavi and yt as well as integration with Astropy quantities and Table were highlighted. The need for more documentation for texfigure and PythonTeX was brought up and work on this is ongoing.

The future of astropy.modeling (led by Tom Robitaille and others)

We started off by discussing features that are missing in astropy.modeling, but then moved into a more general discussion about whether astropy.modeling is currently following the correct development direction. In particular, concerns were raised about the existing fitters being too restrictive and making too many assumptions about the particular type of fitting. While other parts of Astropy have generally tried to do things the ‘right’ way, astropy.modeling only implements the simplest kind of fitting and doesn’t encourage good statistical practices.

We decided that it would make sense to better separate the Astropy models (which are useful in any case) from the fitting part, and there were some suggestions to deprecate the fitting entirely and instead rely for example on sherpa and other packages to carry out the fitting. At the same time, the existing fitters provide an easy way to interface to the SciPy fitters, so there may be value in keeping them, but refactoring the fitting documentation so that the models and fitting documentation are separate, and that the fitting documentation shows both ways to interface with the SciPy fitters and ways of interfacing with sherpa and other packages (such as emcee).

Specific action items to come out of this were to investigate how easy it would be to interface Astropy models with sherpa and emcee (which was done later during the week), and that we should prepare an Astropy Proposal for Enhancements (APE) that discusses which path to follow moving forward. In addition, a Google Summer of Code student will be working on the Astropy/Sherpa bridge.

What is your software development workflow? (led by Adrian Price-Whelan)

We had a organic discussion of software tools that we use to facilitate our daily work flow. Several text editors (Sublime Text, TextMate) were demoed including of linters and git integration. Below are links to software that was demoed as part of this session:

- Collaborative manuscript preparation: Overleaf, Sharelatex, GitHub, and Authorea
- Reference management: Zotero and Papers
- Improved terminals: Iterm2 and Z shell
- Remote session persistence: Tmux
- Text editors: Sublime Text, TextMate, Atom, and emacs

The session was recognized as “surprisingly informative” and it reminded all of us of the value of learning about new tools from our colleagues. Specific action items was for everyone to ask colleagues about their software tools and better share innovations and new tools with colleagues.

Current state and future of NDData in Astropy (led by Matt Craig)

Discussion began with a description of what “NDData” currently is in Astropy, specifically:

- **NDDataBase**, an abstract class that defines the interface only, i.e. a minimal set of properties that **every** NDData-like object must have and what the meaning is. The point is to allow software to be written that can assume for example that the data can always be accessed by a “data” property, that metadata is stored as “meta”, and so on. This is essentially an agreement about what we will call the things needed to work with pixel data.
- **NDData**, an example implementation of a concrete class that implements the NDDataBase interface. Its implementation is minimal, providing setters for properties like mask without putting any restrictions on what those properties are (for example an array or a function).
- **NDDataArray**, an example implementation of an NDData object that has mixin classes for slicing/ indexing, arithmetic, and uncertainty propagation for a simple case.

There was some discussion of metadata and how to combine it; some work on that has been done in astropy.utils.metadata. Much of the discussion focused on the “mask” property and on flags. Projects like JWST and LSST need a way to provide information about mask planes and allow the user to select mask planes to apply in a particular situation. In the current framework that requires some interplay between mask (currently a boolean property) and flags (not included in the NDDataBase interface but are present in NDDataArray). Several options were discussed, and the preferred one was to allow the mask to be an object. In simple use cases the mask could simply be a boolean numpy array. In more complex cases mask could be an object so that one could do ndd.mask.apply_flags(...) and then ndd.mask returns an appropriate mask based on the user choices, and application code can assume that .mask is a boolean that says indicates whether a pixel should be ignored.

Astroplan: observation scheduling discussion (led by Brett Morris)

Observation scheduling with astroplan is the subject of an upcoming Google Summer of Code 2016 project, and the first attempt at building a scheduler with astroplan was made by Erik Tollerud in astropy/astroplan#137. Stephanie Douglas contributed an update to #137 in astropy/astroplan#151, which spawned a number of questions about terminology that needed to be resolved.

We held a discussion with two main purposes: (1) to propose a common lexicon for the constraints+scheduling API that astroplan will use during GSoC 2016, and (2) to assess the degree of compatibility between Erik and Stephanie's proposed API and the glossary of scheduling terms that we all understand. The notes linked above start to outline those terms.

Should Astropy become the new IRAF? (led by Michele Bannister)

This discussion started off since as times goes on, there was a feeling that Astropy risks becoming too large and functionality starts not becoming discoverable. In particular, participants felt that it is not always easy to know where to look (e.g., in the core package or in affiliated packages) for specific functionality. One idea that came out of the discussion would be for all Astropy core and affiliated packages to have a way to declare what functionality they support, then combine all this into a single index of functionality. It would also be nice to have a more up-to-date community driven table of correspondence between IRAF, IDL, and Python, optionally starting from the guide on AstroBetter.

We discussed some of the motivation behind the current model of having the core package and separate affiliated packages, and discussed the pros and cons of having a single monolithic package, many small packages, and something in between. In particular, we talked about the difficulty of installing IRAF, and that Astropy will want to avoid that in future. At the moment, Astropy is very much easier to install than IRAF, and we should make sure it stays that way.

Extending Python (ctypes, f2py, cython, numba, etc.) (led by Jake VanderPlas)

Python is useful for many computational applications, but tends to be slower than other options when it comes to looping over many similar operations. Vectorization through NumPy is a suitable solution in many cases, but has some disadvantages: namely vectorization often leads to a tradeoff between memory use and computational speed.

This discussion was driven by Jake VanderPlas sharing a notebook he developed for a lecture within a Python class. It covered some of the options for interfacing NumPy with compiled languages like C and Fortran, as well as incorporating compilable code within Python itself. In brief, the discussion covered:

- ctypes: Python's low-level interface to C objects. This works for quick-and-dirty applications, but is not easy to make work cross-platform.
- f2py: Python's Fortran interface generator. This can create powerful interfaces to Fortran or C libraries. Much of SciPy was originally built on F2Py.
- Cython: this is a superset of Python, which can be translated directly to C code and compiled. This lets one write "Pythonic" code that can still run very quickly.
- Numba: this is a package that can transparently just-in-time (JIT) compile Python functions into LLVM bytecode, which can target the CPU or GPU.

- cffi: The C Foreign Function Interface is a more recent option, developed in part to make more of a separation between wrapping of compiled code and the CPython implementation.

We had a lively discussion of the advantages and disadvantages of each. In particular, Cython seems to be a standardly-used tool in the community (see AstroPy in particular), and while people share some excitement on the prospects of Numba, most agreed that it's still somewhat brittle and not yet ready for use in production code.

A general time series/lightcurve class for Astropy (led by Stuart Mumford)

This discussion focused on the question: *What would a universal time series class in Astropy look like?* In particular, we discussed what kind of functionality would be general to include in such a class and that would satisfy all the participants' use cases. We also discussed details about the implementation, and whether such a class should for example be based on the Astropy Table class. There were some requests for making sure that the implementation has good performance with large datasets. The primary action item of this discussion was to write an Astropy Proposal for Enhancements, which was started here during the week.

Try out your data in Glue and discuss improvements (led by Tom Robitaille)

We started off by getting Glue up and running on participants' laptops. We then tried out some of the basic functionality with some canned test data, and then proceeded to use our own data. We tested out various 2D and 3D visualization methods. Overall response was positive, and some bug reports were filed as issues on GitHub.

What is yt? (led by Nathan Goldbaum and Cameron Hummels)

We provided a basic introduction to yt. This included covering some of the introductory material, with a focus on basic visualization and how to load data in a variety of formats.

Astropy docs enhancements (led by Adrian Price-Whelan and Adam Ginsburg)

The astropy-tutorials effort was started to create more immersive examples of inter-package functionality of the Astropy project and simultaneously teach fundamental Python concepts. Two key issues with the tutorials are that (1) the tutorials are written as IPython notebooks and stored in a separate GitHub repository so that they are not automatically tested with the core Astropy repository and (2) community contribution to the project has been slower than expected. We discussed whether, given this, it is worth spending more developer effort on creating new content or considering developing a new way to demonstrate and highlight Astropy functionality.

We decided to create a new "examples gallery" in the core Astropy documentation that contains shorter snippets or examples of features that are tested alongside the documentation. (See scikit-learn for implementation example.) During the sprints, we implemented this idea via sphinx-gallery and added content to an example gallery.

Specific action items are to: cultivate contributors to write tutorials and add examples to the gallery, identify examples already in documentation which could be migrated to the gallery, and advertise the existence of these resources to broader astronomy community.

[Note that since the workshop, the gallery of tutorials above has now been included in the Astropy documentation.]

Extreme openness: a debate (led by Jake Vanderplas and Sarah Tuttle)

We discussed the idea of doing open research from start to end – that is, all the data and code would be public from the start, and the paper would be written in the open. While participants in the discussion generally agreed that this was a noble goal, there were concerns by some that this is simply not possible for junior participants, since it greatly increases the chances of getting scooped. We therefore then discussed what intermediate steps could be taken towards open research.

The most obvious place where we could become more open would be to make sure that at publication time, we also provide all the data and code needed to reproduce a paper. This then provides the benefits of openness without the downsides of getting scooped (nevertheless, there were concerns that for PhD students who may be working on a series of related papers, this could still be an issue). One of the main barriers to full reproducibility is the investment of time needed to tidy up the code and data to make sure they are understandable by others – although arguably it would benefit the scientists who did the work themselves if they look back at the work in 10-20 years. We noted that there is very little reward in the current system for publishing reproducible science, and some ideas were put forward, including for example having journals provide incentives (e.g., free open access, reduced page charges, 'badges' on papers, featured articles) for reproducible papers. We also noted that *full* reproducibility may also be very difficult, short of providing an entire virtual machine which would contain everything.

Ginga sprint/refactoring (led by Eric Jeschke)

Eric Jeschke and Pey-Lian Lim sat down and attempted to resolve some outstanding issues. For instance, re-organization of plugin manual into sub-pages and improving Jupyter notebook functionality, among other things. There were discussions and coding. By the end, Eric merged two pull requests. Pey Lian opened a few issues based on the sprint findings. Stuart Mumford also made some improvements to Cuts plugin. Megan Sosey worked on improving Ginga backend for "imexam". While, Matt Craig tested Ginga on Jupyter notebook. Several others also tested Ginga and provided feedback and/or bug reports.

Astropy Roles (led by Kelle Cruz, Erik Tollerud, and Tom Robitaille)

We discussed officially assigning roles for tasks like maintaining sub-packages, documentation, release, and distribution. Each role would have at least one main person and at least one deputy. Some concerns were raised in assigning unpaid roles to an open-source project that depends on volunteer contributors. Jake Vanderplas provided an example from SciPy, where he is assigned to maintain the stats package but he only chimes in when no one else does after a while (e.g., when a pull request sitting there too long without review). In addition, he mentioned that he has a backup available when he is not. The key outcome is that the Astropy Project will create and maintain a web page on http://astropy.org which defines project roles and lists the lead and deputy for each role.

Plan for Python in Astronomy 2017 (led by Stuart Mumford and Kelle Cruz)

There was general agreement that Python in Astronomy 2017 (and beyond) should happen. There was general agreement that a size around 55-75 was best. We brainstormed a list of features that would be important for any location to have to be suitable for this conference series:

- Lecture room that can fit all participants
- Small meeting rooms (for groups of 3-20) each with projectors
- All rooms in same building and if possible same floor
- Tables, not just chairs
- Good internet
- Whiteboards/blackboards
- Central area for planning
- Ease of travel (financially and travel wise internationally)
- Accessibility
- Enough people to populate LOC

We also brainstormed on a list of possible locations and contact people, and it will be the responsibility of the next SOC to decide on a venue. We decided that the SOC chair should first be appointed, and then the SOC chair should appoint the SOC (with an ideal size of ~6 or so SOC members in total). Stuart Mumford has been named the SOC chair for the 2017 meeting.

We made a list of all the things that a SOC/LOC need to do, and an action item will be to tidy these up and make them into a formal document that future SOCs/LOCs can use.

Astropy paper v2.0 (led by Erik Tollerud)

All participants agreed that a second Astropy paper is now needed, to make sure that we continue to give credit to new contributors. We discussed what the content of the paper should be – it should include a description of all the significant new additions since the last paper, and could also include:

- An overview of astropy-helpers
- An overview of the package-template
- The long term support (LTS) releases

We proposed submitting the paper at the 1.3 feature freeze so that it is hopefully published shortly after the actual final 1.3 release. We also discussed the issue of author order. Since it's very difficult to find metrics everyone would agree on, we floated the idea of making sure 'The Astropy Collaboration' is the cited 'author', and that we then just list authors alphabetically, with an optional footnote saying that the relative contributions of people to the package can be seen for example on GitHub. Erik Tollerud and Kelle Cruz have volunteered to oversee this paper.

Hacks/Sprints

Fast Lomb-Scargle periodograms

People: Jake Vanderplas

The aim was to take four separate implementations of Lomb-Scargle periodograms and gave them a consistent API, with the intent to include this in Astropy. This was then merged into the core Astropy package following the workshop, and is now available in Astropy 1.2.

Demo link

Astropy examples gallery

People: Adrian Price-Whelan and Kelle Cruz

The aim was to add a gallery of Astropy examples to http://docs.astropy.org using sphinx-gallery. This was the outcome of an earlier unconference session. This was merged into the astropy documentation following the workshop, and is available in Astropy 1.2.

Source Code - Demo link

Distribution class for Astropy

People: Erik Tollerud

Continued to work on developing a Quantity + Monte-Carlo samples class for Astropy that allow easy distribution/error propagation.

Demo link

Reading large ASCII tables in smaller chunks

People: Tom Aldcroft

There is an outstanding feature request in Astropy to provide the ability to read very large ASCII tables in more manageable chunks. One might then filter or otherwise process each chunk individually and maintain only the desired outputs instead of the entire table. I developed a solution to this which leverages the existing functionality in the fast C-based reader to use memory mapping of the file to split the input into smaller chunks.

This appeared to be ready for a formal pull request, but then a nasty issue surfaced for input files greater than 2GB. Even though everything about the development system was 64-bits, for files bigger than 2**32 bytes the new routine starts gobbling all available memory until crashing out. In the process of this hack I learned a bit about the perils of mmap and will likely consider a more outside-in implementation. This will require a larger change footprint but should be generalizable to more formats.

Source code - Diff from master

Inquiry lesson Plan

People: Stephanie Douglas, Drew Leonard, Sam Lawler, Sam Bennett, Jes Ford, and Mike Alexandersen

We set out to create an inquiry activity where learners investigate the relationship between multi-band photometry and a star's visible color as well as good programming practices. We sketched out the astronomy and programming learning goals, as well as rubrics for both, and a very rough investigation prompt ("here's photometry - what can you learn"). We also tested some sample investigation paths; one is linked below. The future development plans are unclear; we likely need a test class before we invest too much more time in it.

Source Code (planning doc; includes brainstorming and discussion of teaching testing)

Demo Link (sample investigation path)

Astroplan scheduler

People: Stephanie Douglas, Brett Morris, Erik Tollerud, Eric Jeschke, and Geert Barentsen

We spent two unconference sessions on this, one coding and one on future development. Erik had previously coded the scheduler class and related classes. Stephanie added a scheduler that schedules targets at the best time in order of priority. We also had a planning meeting where we mostly defined various scheduling terms (notes here). Stephanie will take over the initial scheduler development from Erik; there may be a Google Summer of Code student working on it this summer.

Source code - Demo Link

Glue + pyspeckit

People: Adam Ginsburg and Tom Robitaille

We worked on making a data viewer in Glue based on pyspeckit, which allows users to show 1D spectra, as well as collapse 3D spectral cubes and subsets of 3D spectral cubes. Users can then select points from the spectrum to use for the fitting, and fit line profiles or polynomials.

Source Code - Demo Link

@astrofrog GitHub bot

People: Dan Foreman-Mackey

The goal was to try and use machine learning to generate GitHub comments, by learning from existing Astropy contributors. An example of a generated comment is: *If can still see you forther and because some real up in pos I want to do: systecls. the instrotative numbying to ```setup.py`` of an error is the user as *1.0 at the docs of but at help t-end of the people equinorm in any of the name of developming links rows cos.*

Background information on Recurrent Neural Networks:

- http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://github.com/dfm/pyastro16

Eblur/dust

People: Lia Corrales, Karl Gordon, and Médéric Boquien

We discussed the essential properties of a dust model and ways to organize the code to make it useful for the astrophysical community. During the code sprints, Lia reorganized and packaged the code using the Astropy package template.

Source Code - Demo Link

Astroplan web app

People: Geert Barentsen and Brett Morris

Exposing astroplan's functionality to the web!

Source code - Demo Link

Python imexam

People: Megan Sosey

I worked on adding ginga's new HTML5 widget to imexam as the primary viewer for ginga and removed the matplotlib and qt widgets which were slower to interact with and more cumbersome for users. Basic communication was enabled during the sprint and fully flushed out later. This will be part of the next stable imexam release.

Source Code

Time series object

People: Stuart Mumford

Following an excellent discussion about what features a time series object in Astropy would have, we put together the start of an APE proposing adding such an object to Astropy core.

Source Code - Demo Link

Open Astronomy conda channel

People: Matt Craig

The goal was to create a conda channel for any astronomy package.

Source Code - Conda channel

Astropy modeling and Sherpa

People: Daniela Huppenkothen

We developed a proof of concept of how to use astropy.modeling with the emcee package.

Notebook

stginga

People: Pey Lian Lim, Eric Jeschke, and Russell Owen

stginga is probably the first ever “affiliated package” of Ginga, in the sense that it provides additional Ginga plugins and functionality that are specific to STScI data analysis needs. When a feature (usually a plugin) in stginga is deemed appropriate for the “core” (Ginga), it is absorbed into Ginga via a pull request on GitHub.

At this unconference, Pey Lian Lim presented a demo of TVMark and TVMask plugins to Eric Jeschke and Russell Owen. The TVMark plugin was modeled after “tvmark” task in IRAF, while TVMask is like TVMark but for mask images instead of X-Y coordinates. Later, during a sprint, Pey Lian improved TVMark plugin based on feedback mainly from Russell. The improved plugin was presented to all participants as a lightning talk, which also covered additional stginga features as time permitted.

Ginga

People: Eric Jeschke, Pey Lian Lim, Stuart Mumford, Megan Sosey, Russell Owen, Erik Tollerud, Adam Ginsburg, Matt Craig, and others

This sprint included collaborations with several projects that are (or are interested in) using Ginga, several bug fixes and documentation enhancements, and feedback from users and troubleshooting opening problem image files. We implemented some requested features.

Generalized WCS examples

People: Nadia Dencheva

I wrote a function that takes pixel and world coordinates and creates a WCS.

Source Code

Cube reprojection

People: Adam Ginsburg and Axel Donath

We generalized cube reprojection in the reproject package to work on all 3 dimensions (including spectral resampling). We also opened a pull request to incorporate it into spectral-cube.

Python 3 for scientists

People: Stephanie Douglas, Adrian Price-Whelan, Stuart Mumford, Nathan Goldbaum, Tom Robitaille, and Erik Tollerud

We decided to work on a website that focuses on demonstrating new Python 3 features that would be useful for the average scientist, and does not include any negatives/shaming/etc of Python 2 :)

Souce code - Demo Link

Astrohackweek planning application form

People: Kyle Barbary and Daniela Huppenkothen (with consulting from Kelle Cruz)

We designed the application form for the next awesome week-long workshop: AstroHackWeek. One interesting outcome from discussions was the phrasing of application questions about gender and racial/ethnic background. Rather than enumerating all possible gender and racial/ethnic backgrounds in a multiple choice question which would inevitably be incomplete, we simply ask if the applicant self-identifies as an underrepresented minority in each category. This gives us just the information we plan to use in selection and avoids the enumeration problem. Meanwhile, having separate questions for gender and race/ethnicity preserves some information on intersectionality.

Source Code - Demo Link

Morphometry in photutils

People: Leonardo Ferreira

After a brief discussion with Erik Tollerud, I started to implement non-parametric morphology measurements into photutils based on those available in Morfometryka. As there many of these measurements available in the literature, I made an example case using Gini Coefficient with a plan for extending photutils.morphometry as a hub for this kind of measurements in the future.

Python 3 for packages

People: Brigitta Sipocz

The goal was to make more packages from the OpenAstronomy ecosystem be compatible with Python 3. The result is that the new releases, v0.2 of Halotools and v0.7 of SunPy, are now Python 3 compatible.