Astropy and the Python in Astronomy Ecosystem

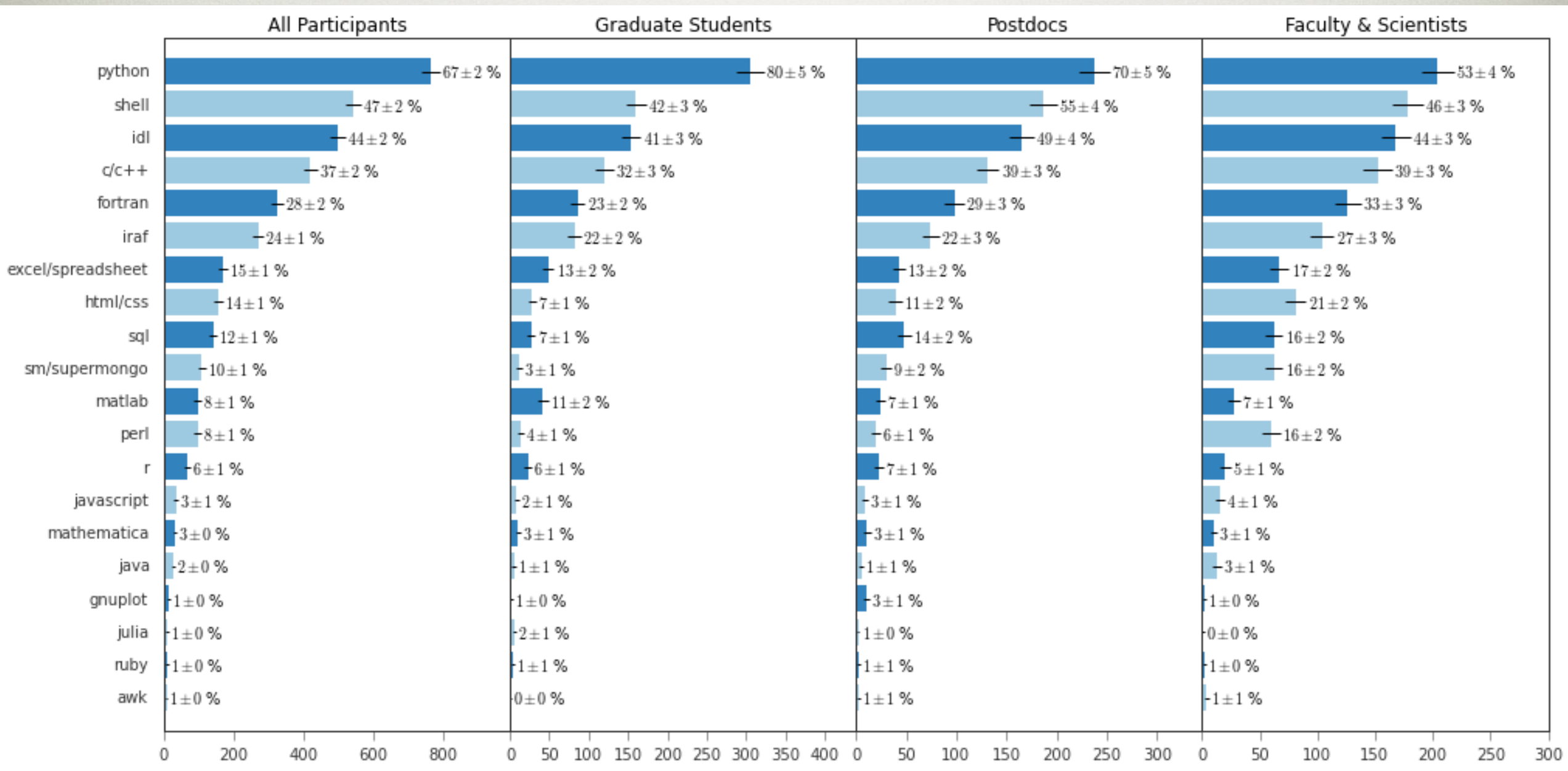
Erik Tollerud

@eteq

STScI | SPACE TELESCOPE SCIENCE INSTITUTE

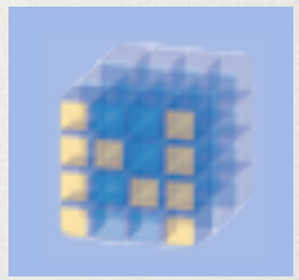
Astropy Coordination Committee Member
STScI Data Analysis Tools Branch, Project Scientist

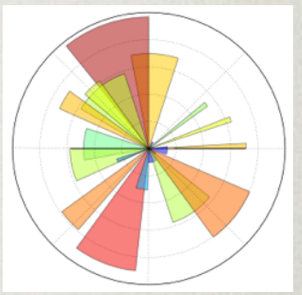
Python is now the dominant Language in Astro



Momcheva & Tollerud 2015

Python (esp. in Science) Embraces Open Development





Plenty of Open Source Astro Software is out there... But

Open Source

≠

Open Development

Astropy's Origin Story

Q. How do I use python to convert from Equatorial J2000 RA/Dec to Ecliptic coordinates (as of 2011)**?**

A. Use any of:

- pyast
- Astrolib
- Astropysics
- Kapteyn
- PyEphem
- PyAst
- PyAstro
- Probably more...

Lots of wasted effort!

Mutually incompatible!

Astropy's Origin Story

Everyone agreed this was bad.

> Do we as a community really need yet another separate python library for astronomy and yet another attempt at building a core set of routines ported from the IDL library?
>
> Marshall Perrin on "astropy" mailing list, June 2011

Astropy's Origin Story

Everyone agreed this was bad.

(Agreement ends up *crucial* to shared development.)

A grassroots discussion started in June 2011, followed by a series of votes (~100 astronomers), and some initial dev work (by a mix of astronomers and engineers).

The Result: astropy

Check out http://bit.ly/astropyvision for the original "vision"

What is the philosophy behind Astropy?

The Astropy Project is a community effort to develop a common core package for Astronomy in Python and foster an ecosystem of interoperable astronomy packages.

What is the philosophy behind Astropy?

The Astropy Project is a community effort to develop a common core package for Astronomy in Python and foster an ecosystem of interoperable astronomy packages.

This means both ***by*** and ***for*** the community

What is the philosophy behind Astropy?

The Astropy Project is a community effort to develop a common core package for Astronomy in Python and foster an ecosystem of interoperable astronomy packages.

This means both *by* and *for* the community

(Professional) Astronomers help write it

What is the philosophy behind Astropy?

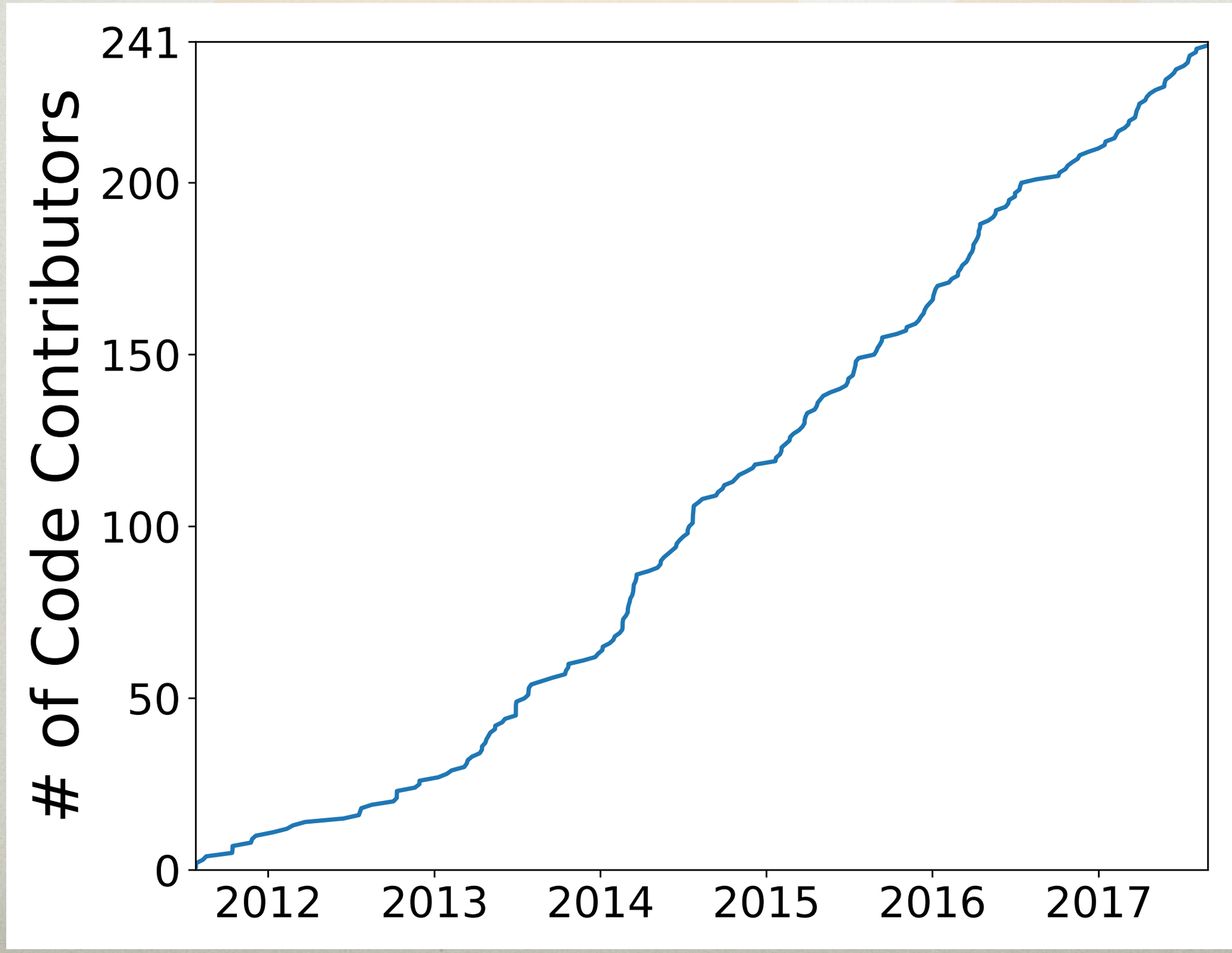
The Astropy Project is a community effort to develop a common core package for Astronomy in Python and foster an ecosystem of interoperable astronomy packages.

This means both *by* and *for* the community

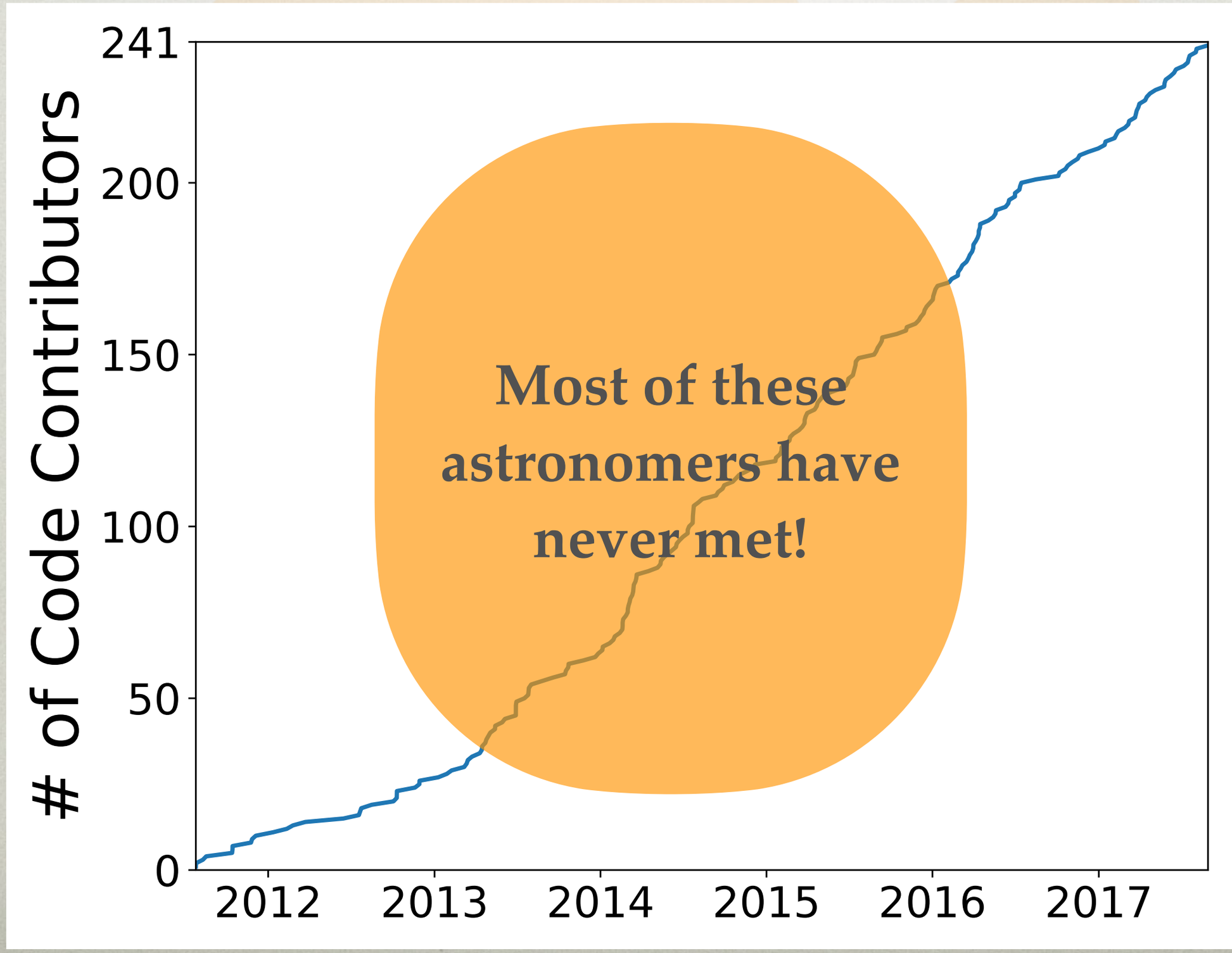
(Professional) Astronomers help write it

It should be useful for them as part of their day-to-day work

Contributions To Astropy are ≳200 and rising



Contributions To Astropy are ≳200 and rising



Getting Scientists to work together is like Herding Cats...

Getting Scientists to work together is like Herding Cats...

Getting Scientists to work together is like Herding Cats...

You just need to give them a structure in which to work and they'll do it themselves

Community Development: Github

This repository Search Pull requests Issues Marketplace Gist

astropy / astropy

Unwatch 146 Unstar 1,269 Fork 703

Code Issues 699 Pull requests 102 Projects 1 Wiki Settings Insights

Repository for the Astropy core package http://www.astropy.org

Edit

python astronomy science Manage topics

18,422 commits 10 branches 55 releases 206 contributors BSD-3-Clause

Branch: master New pull request Create new file Upload files Find file Clone or download

adrn committed on GitHub Merge pull request #5653 from adrn/units/optional-quantity-input ... Latest commit 5ed5985 3 days ago

astropy	Merge pull request #5653 from adrn/units/optional-quantity-input	3 days ago
astropy_helpers @ 70b09af	Updating astropy-helpers to v1.3.1	2 months ago
cextern	Update bundled erfa to 1.3.0.	7 months ago
docs	Merge pull request #5653 from adrn/units/optional-quantity-input	3 days ago
examples	Addressing review comment	14 days ago
licenses	Update README.rst	2 months ago
static	Fixed support on Python 3, and got rid of .astropy-root per astropy/a...	3 years ago
.astropy-root	Don't rely on .git to enable auto-build when importing from source tr...	2 years ago
.gitattributes	Use union merge for changelog	3 years ago
.gitignore	Ignore generated NGC6976 images [skip ci]	2 months ago

Community Development: Github PRs ... for All!

github
SOCIAL CODING

This repository Search Pull requests Issues Marketplace Explore

astropy / astropy

Unwatch 163 Unstar 1,436 Fork 769

Code Issues 709 Pull requests 84 Projects 1 Wiki Insights Settings

Allow FITS tables with time columns (not written by Astropy) to be read by io.fits #6442

Edit

Open AustereCuriosity wants to merge 14 commits into `astropy:master` from `AustereCuriosity:Time_read_fits`

Conversation 87 Commits 14 Files changed 6 +595 −92

AustereCuriosity commented on Aug 14 • edited — Contributor

This works for Chandra files and XMM files. I have added a test for reading Chandra files. The test needs to be more detailed and will be modified soon.

NOTE: The chandra file is huge, so I'll probably use another file or reduce it.

astropy-bot bot commented on Aug 14 • edited

Hi there **@AustereCuriosity** 👋 - thanks for the pull request! I'm just a friendly 🤖 that checks for issues related to the changelog and making sure that this pull request is milestoned and labelled correctly. This is mainly intended for the maintainers, so if you are not a maintainer you can ignore this, and a maintainer will let you know if any action is required on your part 😃 .

Everything looks good from my point of view! 👍

If there are any issues with this message, please report them here

bsipocz added this to the **v3.0.0** milestone on Aug 14

bsipocz added io.fits Work in progress labels on Aug 14

bsipocz commented on Aug 14 — Member

@AustereCuriosity - If you need this file, it can go into the astropy-data repository and then can be accessed as remote-data from here.

👍 1

Reviewers
taldcroft ✗
mhvk ✓

Assignees
No one—assign yourself

Labels
Affects-dev
Affects-release
Enhancement
io.fits

Projects
None yet

Milestone
v3.0.0

Notifications
Unsubscribe
You're receiving notifications because you're subscribed to this repository.

Community Development: Github PRs ... for All!

github
SOCIAL CODING

This repository Search Pull requests Issues Marketplace Explore

astropy / astropy

Unwatch 163 Unstar 1,436 Fork 769

Code Issues 709 Pull requests 84 Projects 1 Wiki Insights Settings

Allow FITS tables with time columns (not written by Astropy) to be read by io.fits #6442

Edit

Open AustereCuriosity wants to merge 14 commits into `astropy:master` from `AustereCuriosity:Time_read_fits`

Conversation 87 Commits 14 Files changed 6 +595 −92

AustereCuriosity commented on Aug 14 • edited — Contributor

This works for Chandra files and XMM files. I have added a test for reading Chandra files. The test needs to be more detailed and will be modified soon.

NOTE: The chandra file is huge, so I'll probably use another file or reduce it.

astropy-bot bot commented on Aug 14 • edited

Hi there **@AustereCuriosity** 👋 - thanks for the pull request! I'm just a friendly 🤖 that checks for issues related to the changelog and making sure that this pull request is milestoned and labelled correctly. This is mainly intended for the maintainers, so if you are not a maintainer you can ignore this, and a maintainer will let you know if any action is required on your part 😃 .

Everything looks good from my point of view! 👍

If there are any issues with this message, please report them here

bsipocz added this to the **v3.0.0** milestone on Aug 14

bsipocz added io.fits Work in progress labels on Aug 14

bsipocz commented on Aug 14 — Member

@AustereCuriosity - If you need this file, it can go into the astropy-data repository and then can be accessed as remote-data from here.

👍 1

Reviewers
taldcroft ✗
mhvk ✓

Assignees
No one—assign yourself

Labels
Affects-dev
Affects-release
Enhancement
io.fits

Projects
None yet

Milestone
v3.0.0

Notifications
Unsubscribe
You're receiving notifications because you're subscribed to this repository.

Community Development: Code Review

github
SOCIAL CODING

astropy/io/fits/fitstime.py

```
257 +
258 +def _get_info_if_time_column(col, global_info):
259 +    """
260 +    Check if a column without corresponding time column keywords i
```

mhvk on Aug 16 **Member**

Can you make the docstring explicit that this is only to special-case a column with the name 'TIME' and has units of time?

bsipocz commented 22 days ago **Member**

@AustereCuriosity - Could you please rebase this and address the review comments?

AustereCuriosity added some commits on Aug 6

Read time ...	658c86f
Test for geodetic locations and a little clean up	da67948
Corrections to handle numpy unicode strings in Python3 and Addition o...	d5e760a
Adding tests for GPS and LOCAL scales and for location warnings. Also...	c9a7208
Changelog entry added	6ded8c4
Documentation for reading time	ce51bae
Catch exception and warn user. Also change col.* to col.info.*	1430aef
Changes to documentation in order to incorporate the various aspects ...	ae2b087

Community Development: Code Review

github
SOCIAL CODING

astropy/io/fits/fitstime.py

```
257  +
258  +def _get_info_if_time_column(col, global_info):
259  +    """
260  +    Check if a column without corresponding time column keywords i
```

mhvk on Aug 16 **Member**

Can you make the docstring explicit that this is only to special-case a column with the name 'TIME' and has units of time?

bsipocz commented 22 days ago **Member**

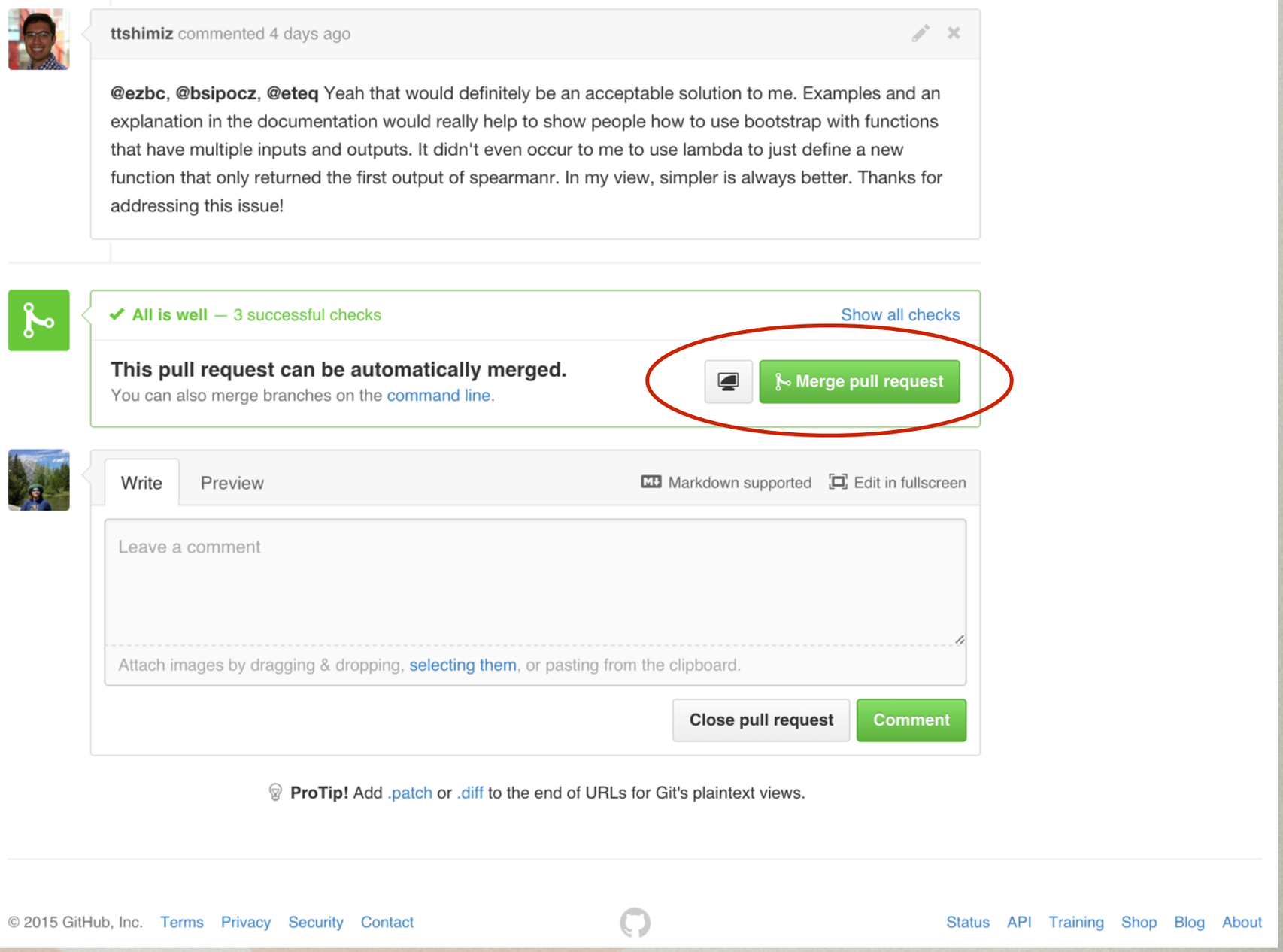
@AustereCuriosity - Could you please rebase this and address the review comments?

AustereCuriosity added some commits on Aug 6

Read time …	658c86f
Test for geodetic locations and a little clean up	da67948
Corrections to handle numpy unicode strings in Python3 and Addition o…	d5e760a
Adding tests for GPS and LOCAL scales and for location warnings. Also…	c9a7208
Changelog entry added	6ded8c4
Documentation for reading time	ce51bae
Catch exception and warn user. Also change col.* to col.info.*	1430aef
Changes to documentation in order to incorporate the various aspects …	ae2b087

Community Development: Consensus!

github
SOCIAL CODING



Community Development

github
SOCIAL CODING

Keeping this process for *all* code is what makes it a community project - a

Do-ocracy

The work that gets done is the work that someone does…

This means *anyone* can be a key player in the project.
This empowers and encourages contributions!

This makes Astropy Responsive to Users' Needs

The Astropy Project is a community effort to develop a single core package for Astronomy in Python and foster interoperability between Python astronomy packages.

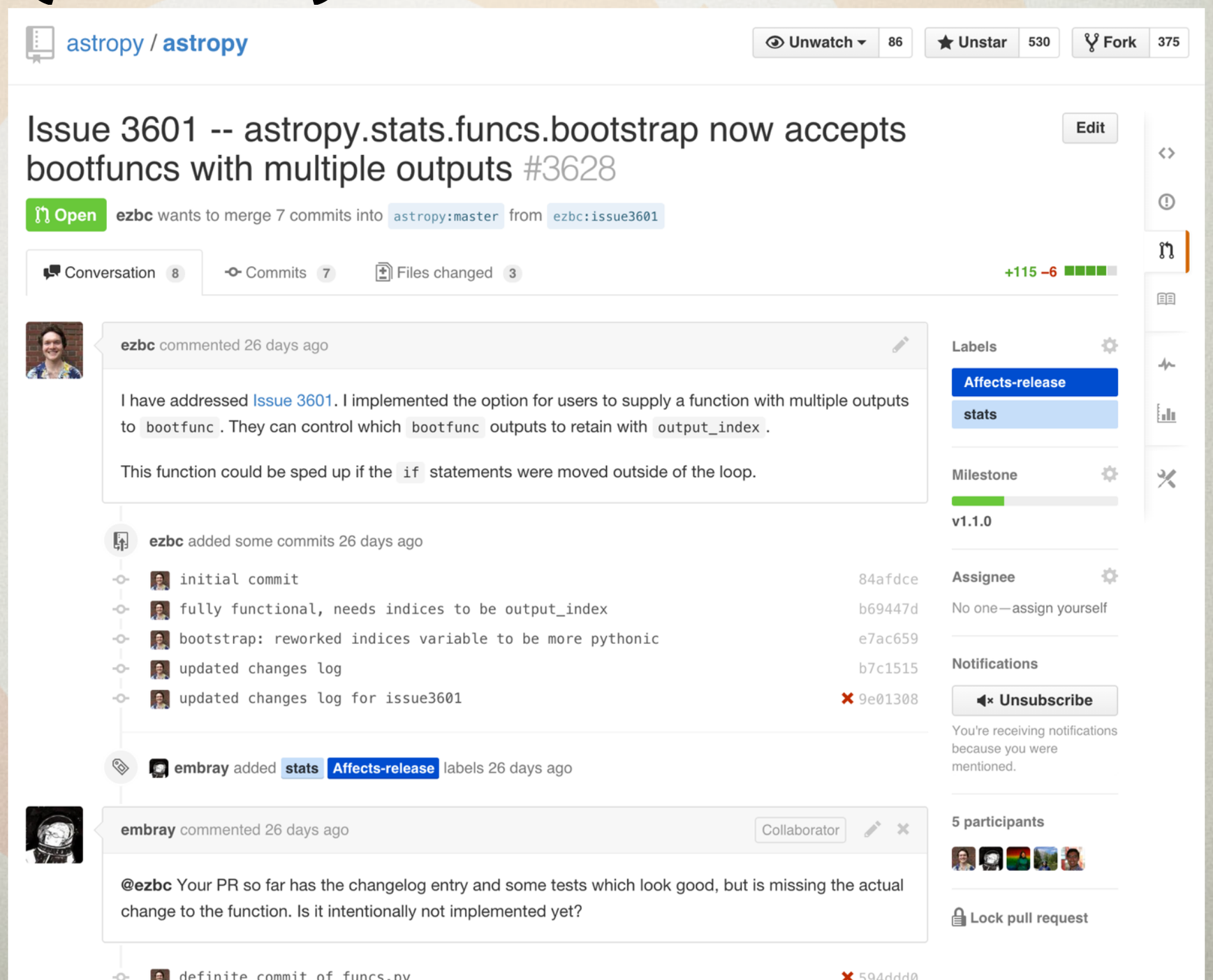
This means both *by* and *for* the community

(Professional) Astronomers help write it

It should be useful for them as part of their day-to-day work

How can this possibly Stay Stable?

Keys To Distributed Development: (Unit) Testing



Keys To Distributed Development: Tests are part of "Code"

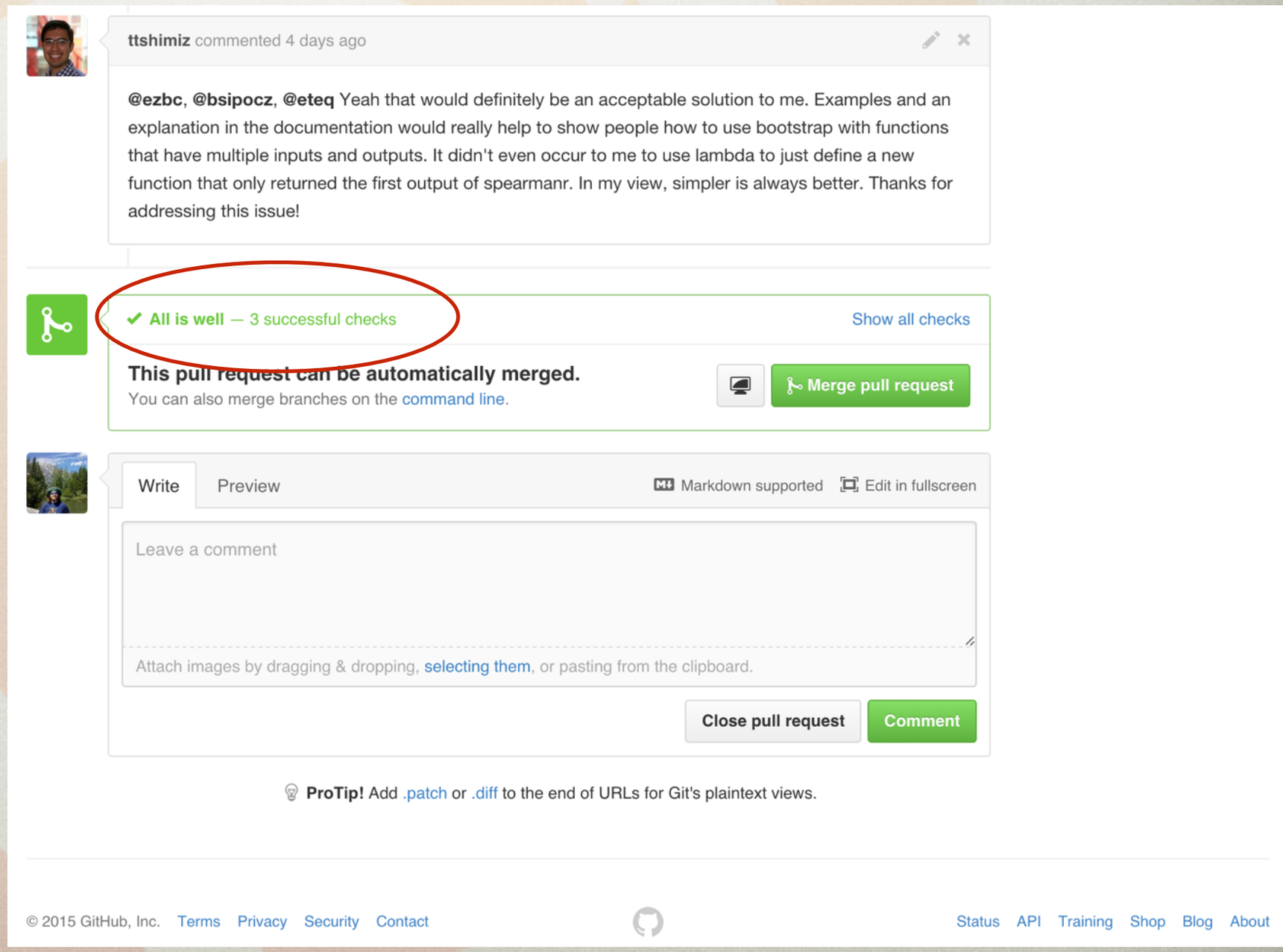
```
264  def test_bootstrap():
265      bootarr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 0])
266      # test general bootstrapping
267      answer = np.array([[7, 4, 8, 5, 7, 0, 3, 7, 8, 5],
268                         [4, 8, 8, 3, 6, 5, 2, 8, 6, 2]])
269      with NumpyRNGContext(42):
270          assert_equal(answer, funcs.bootstrap(bootarr, 2))
271
272      # test with a bootfunction
273      with NumpyRNGContext(42):
274          bootresult = np.mean(funcs.bootstrap(bootarr, 10000, bootfunc=np.mean))
275          assert_allclose(np.mean(bootarr), bootresult, atol=0.01)
276
```

Keys To Distributed Development: Continuous Integration

Travis CI

AppVeyor

circleci



Keys To Distributed Development: Docs In the Code

python

SPHINX
Python Documentation Generator

```python
class FLRW(Cosmology):
    """ A class describing an isotropic and homogeneous
    (Friedmann-Lemaitre-Robertson-Walker) cosmology.

    This is an abstract base class -- you can't instantiate
    examples of this class, but must work with one of its
    subclasses such as `LambdaCDM` or `wCDM`.

    Parameters
    ----------

    H0 : float or scalar `~astropy.units.Quantity`
        Hubble constant at z = 0.  If a float, must be in [km/sec/Mpc]

    Om0 : float
        Omega matter: density of non-relativistic matter in units of the
        critical density at z=0.  Note that this does not include
        massive neutrinos.

    Ode0 : float
        Omega dark energy: density of dark energy in units of the critical
        density at z=0.

    Tcmb0 : float or scalar `~astropy.units.Quantity`
        Temperature of the CMB z=0. If a float, must be in [K]. Default: 2.725.
        Setting this to zero will turn off both photons and neutrinos (even
        massive ones)

    Neff : float
        Effective number of Neutrino species. Default 3.04.

    m_nu : `~astropy.units.Quantity`
        Mass of each neutrino species. If this is a scalar Quantity, then all
        neutrino species are assumed to have that mass. Otherwise, the mass of
        each species. The actual number of neutrino species (and hence the
        number of elements of m_nu if it is not scalar) must be the floor of
        Neff. Usually this means you must provide three neutrino masses unless
        you are considering something like a sterile neutrino.

    name : str
        Optional name for this cosmological object.

    Ob0 : float
        Omega baryons: density of baryonic matter in units of the critical
        density at z=0.

    Notes
    -----
    Class instances are static -- you can't change the values
    of the parameters.  That is, all of the attributes above are
    read only.
    """
    def __init__(self, H0, Om0, Ode0, Tcmb0=2.725, Neff=3.04,
                 m_nu=u.Quantity(0.0, u.eV), name=None, Ob0=None):

        # all densities are in units of the critical density
        self._Om0 = float(Om0)
        if self._Om0 < 0.0:
            raise ValueError("Matter density can not be negative")
        self._Ode0 = float(Ode0)
        if Ob0 is not None:
            self._Ob0 = float(Ob0)
            if self._Ob0 < 0.0:
```

class astropy.cosmology. **FLRW**(*H0*, *Om0*, *Ode0*, *Tcmb0=2.725*, *Neff=3.04*, *m_nu=<Quantity 0.0 eV>*, *name=None*, *Ob0=None*) [edit on github][source]

Bases: **astropy.cosmology.core.Cosmology**

A class describing an isotropic and homogeneous (Friedmann-Lemaitre-Robertson-Walker) cosmology.

This is an abstract base class – you can't instantiate examples of this class, but must work with one of its subclasses such as **LambdaCDM** or **wCDM**.

Parameters:

H0 : float or scalar **Quantity**

Hubble constant at z = 0. If a float, must be in [km/sec/Mpc]

Om0 : float

Omega matter: density of non-relativistic matter in units of the critical density at z=0.

Ode0 : float

Omega dark energy: density of dark energy in units of the critical density at z=0.

Tcmb0 : float or scalar **Quantity**

Temperature of the CMB z=0. If a float, must be in [K]. Default: 2.725. Setting this to zero will turn off both photons and neutrinos (even massive ones)

Neff : float

Effective number of Neutrino species. Default 3.04.

m_nu : **Quantity**

Mass of each neutrino species. If this is a scalar Quantity, then all neutrino species are assumed to have that mass. Otherwise, the mass of each species. The actual number of neutrino species (and hence the number of elements of m_nu if it is not scalar) must be the floor of Neff. Usually this means you must provide three neutrino masses unless you are considering something like a sterile neutrino.

name : str

Optional name for this cosmological object.

Ob0 : float

Omega baryons: density of baryonic matter in units of the critical density at z=0.

Notes

Class instances are static – you can't change the values of the parameters. That is, all of the attributes above are read only.

Attributes Summary

H0	Return the Hubble constant as an **Quantity** at z=0

v: stable

Keys To Distributed Development: Docs on the Web

SPHINX
Python Documentation Generator

Read the Docs
Create, host, and browse documentation.

astropy About Get Help Contribute Affiliated Packages Search Documentation

astropy
A Community Python Library for Astronomy

The Astropy Project is a community effort to develop a single core package for Astronomy in Python and foster interoperability between Python astronomy packages.

Current Documentation | Other Docs ▼

Current Version: 1.1
Please remember to acknowledge the use of Astropy!

Install Astropy

OS X | Linux | Windows | </> Source | Developer

There are a number of options for installing the astropy package on MacOS X. Astropy can be installed using the MacPorts or Fink package managers, and is also included by default in the Anaconda Python Distribution (more details here), Enthought Canopy, and Ureka, which provide an easy way to get set up with a scientific Python distribution. MacPorts usually includes new releases almost immediately, but Anaconda and Canopy may not always include the latest version.

You can also install the latest version of Astropy using pip or by downloading the source code and installing it manually - see the Source tab above for more details.

Do none of the above instructions work for your system, or do you need more detailed instructions? Check out the installation instructions in our documentation.

astropy:docs astropy Index Modules Search

Astropy v1.0.1 » Data Tables (`astropy.table`) « previous | next »

Page Contents

Data Tables (`astropy.table`)

Introduction

`astropy.table` provides functionality for storing and manipulating heterogeneous tables of data in a way that is familiar to `numpy` users. A few notable features of this package are:

- Initialize a table from a wide variety of input data structures and types.
- Modify a table by adding or removing columns, changing column names, or adding new rows of data.
- Handle tables containing missing values.
- Include table and column metadata as flexible data structures.
- Specify a description, units and output formatting for columns.
- Interactively scroll through long tables similar to using `more`.
- Create a new table by selecting rows or columns from a table.
- Perform *Table operations* like database joins and concatenation.
- Manipulate multidimensional columns.
- Methods for *Reading and writing Table objects* to files
- Hooks for *Subclassing Table* and its component classes

Currently `astropy.table` is used when reading an ASCII table using `astropy.io.ascii`. Future releases of AstroPy are expected to use the `Table` class for other subpackages such as `astropy.io.votable` and `astropy.io.fits`.

Note

Starting with version 1.0 of astropy the internal implementation of the `Table` class changed so that it no longer uses numpy structured arrays as the core table data container. Instead the table is stored as a collection of individual column objects. *For most users there is NO CHANGE to the interface and behavior of |Table| objects.*

The page on *Table implementation change in 1.0* provides details about the change. This includes discussion of the table architecture, key differences, and benefits of the change.

Getting Started

The basic workflow for creating a table, accessing table elements, and modifying the table is shown below. These examples show a very simple case, while the full `astropy.table` documentation is available from the *Using table* section.

First create a simple table with three columns of data named `a`, `b`, and `c`. These columns have integer, float, and string values respectively:

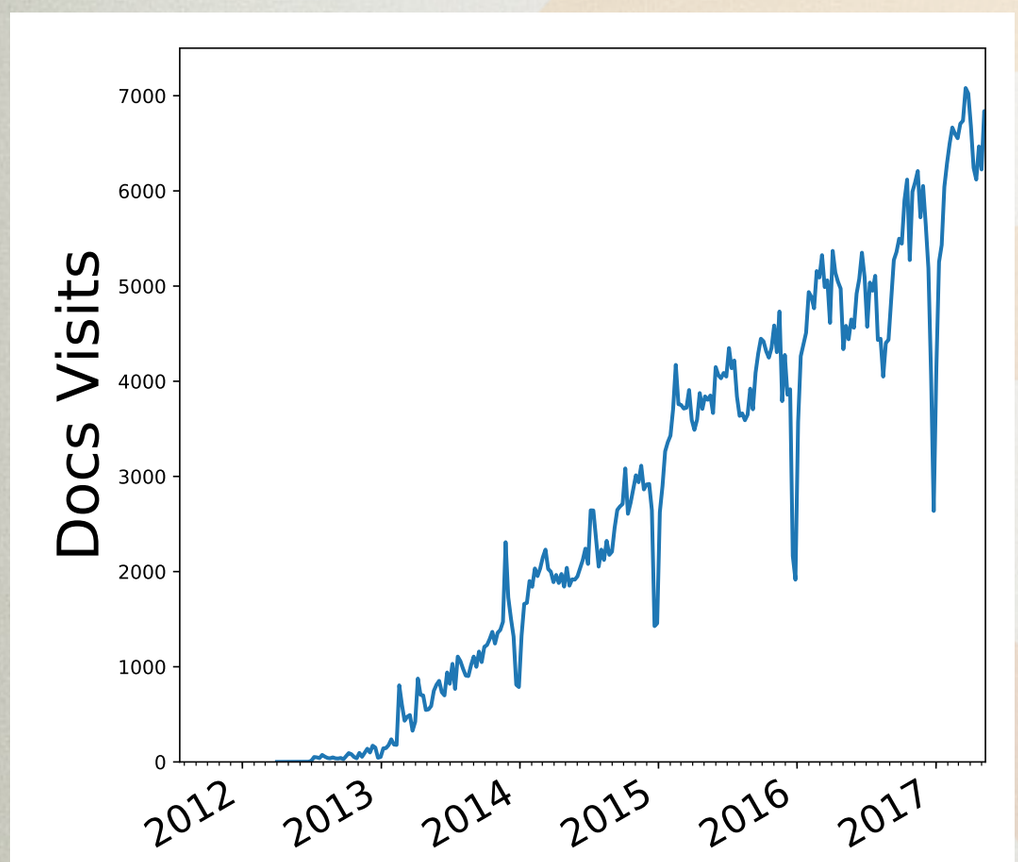
```
>>> from astropy.table import Table
>>> a = [1, 4, 5]
>>> b = [2.0, 5.0, 8.2]
>>> c = ['x', 'y', 'z']
>>> t = Table([a, b, c], names=('a', 'b', 'c'), meta={'name': 'first table'})
```

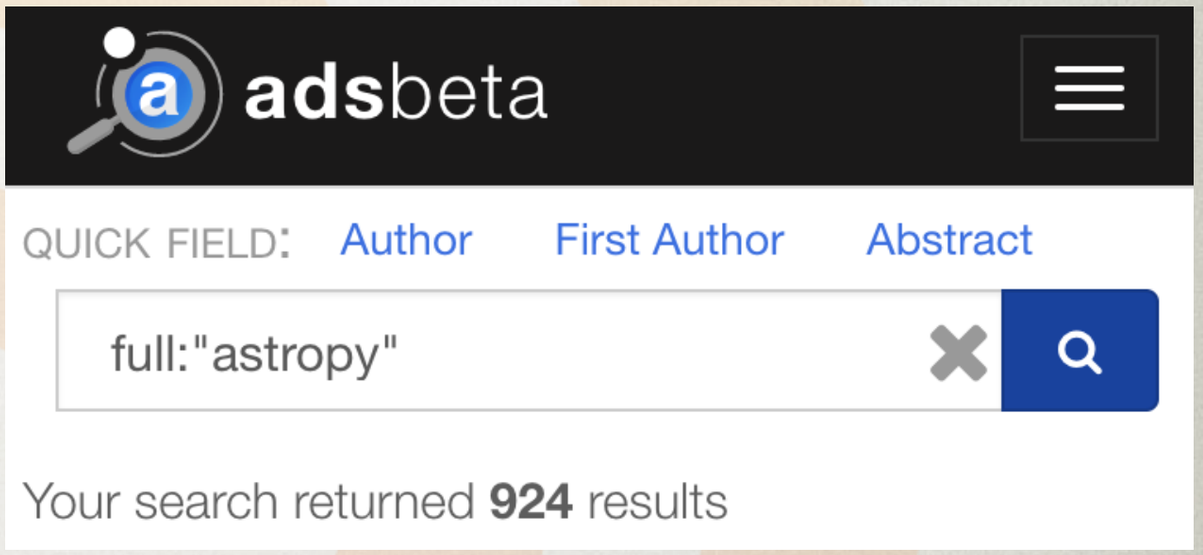
v: stable

So If You Want to do open Development:

- Must eat your own dog (cat?) food: if you require the same process of everyone, developers ≡ users.
- Unit tests and Continuous Integration ensure new contributions are fine. Tests and code are inseparable!
- Docs must be written **with** the code and continuously published

And It Actually DOES enable science





Reused by 900 projects

100 percentile impact overall

Compared to all research software on PyPI, based on relative downloads, software reuse, and citation.

And Large Projects

- James Webb Space Telescope (JWST)
- Hubble Space Telescope (HST)
- LSST Science Pipelines
- Gran Telescopio Canarias (GTC) pipeline for EMIR (Infrared MOS) and MEGARA (Optical MOS and IFU)
- INAF - DISCOS project
- Cherenkov Telescope Array Pipeline framework
- SimCADO - the MICADO+E-ELT instrument simulator
- Code Investigating GALaxy Emission (CIGALE)
- The Herschel Extragalactic Legacy Project
- Tuna at LAM, pipeline for reduction of Fabry-Perot interferograms
- Gemini data reduction software, and Science and Data quality pipeline
- Dark Energy Spectroscopic Instrument (DESI)
- MUSE - MPDAF (MUSE Python Data Analysis Framework)
- AAO - SAMI Galaxy Survey
- Hubble Frontier Fields
- Hubble RELICS Project
- UTFSM - Universidad Técnica Federico Santa María
- Daniel K. Inouye Solar Telescope (DKIST) data management software

The "For" part

The Astropy Project is a community effort to develop a single core package for Astronomy in Python and foster interoperability between Python astronomy packages.

This means both ***by*** and ***for*** the community

It should be useful for them as part of their day-to-day work

Astropy Core Package (currently v2.0.x)

Best place to look is always

http://docs.astropy.org

- Units and "Quantities" (arrays with units that act the way you'd expect). Integrated with comprehensive astro-appropriate physical constants
- Date/time good to nanoseconds over a Hubble time
- Celestial coordinates and their transformations (**now includes velocities**)
- Table manipulation, including many arcane astro formats (**now works with pandas**)
- *nddata*: Image analysis and interoperability data structures (**now includes CCDData**)
- Astro-appropriate convolution
- WCS (pixel ↔ sky mapping)
- Extensible I/O: FITS, VOTable, hdf5, custom
- Data modeling and fitting (**now integrate with units**)
- Common Astrostatistics tools
- Cosmology tools

How do you learn More about Using Astropy?

astropy Tutorials http://tutorials.astropy.org

http://docs.astropy.org

Python in Astronomy Facebook group

astropy Slack team: http://joinslack.astropy.org

Astropy mailing list

astropy-dev mailing list

Talk to me!

Some things for Next Major Version (v3.0)

- Feature freeze mid-Dec
- Only Python 3.x. Means some Py 3-only features can now be used.
- *"%"* operator for composite models (?)
- Support for statistical distributions in Quantity / uncertainties (?)
- 3.1 will be particularly stability and performance-focused

How does this connect back to the Broadest sense of "Community" Software?

The astropy Project and package

The Astropy Project is a community effort to develop a common core package for Astronomy in Python and foster an ecosystem of interoperable astronomy packages.

Core package "astropy" ≠ "Astropy Project"

The core package is what's in github repo *astropy/astropy*. I.e., what "`pip install astropy`" or "`conda install astropy`" gets you.

"Astropy Project" includes all the coordinated and affiliated packages and community.

Affiliated Packages

Canonical list is at: http://affiliated.astropy.org

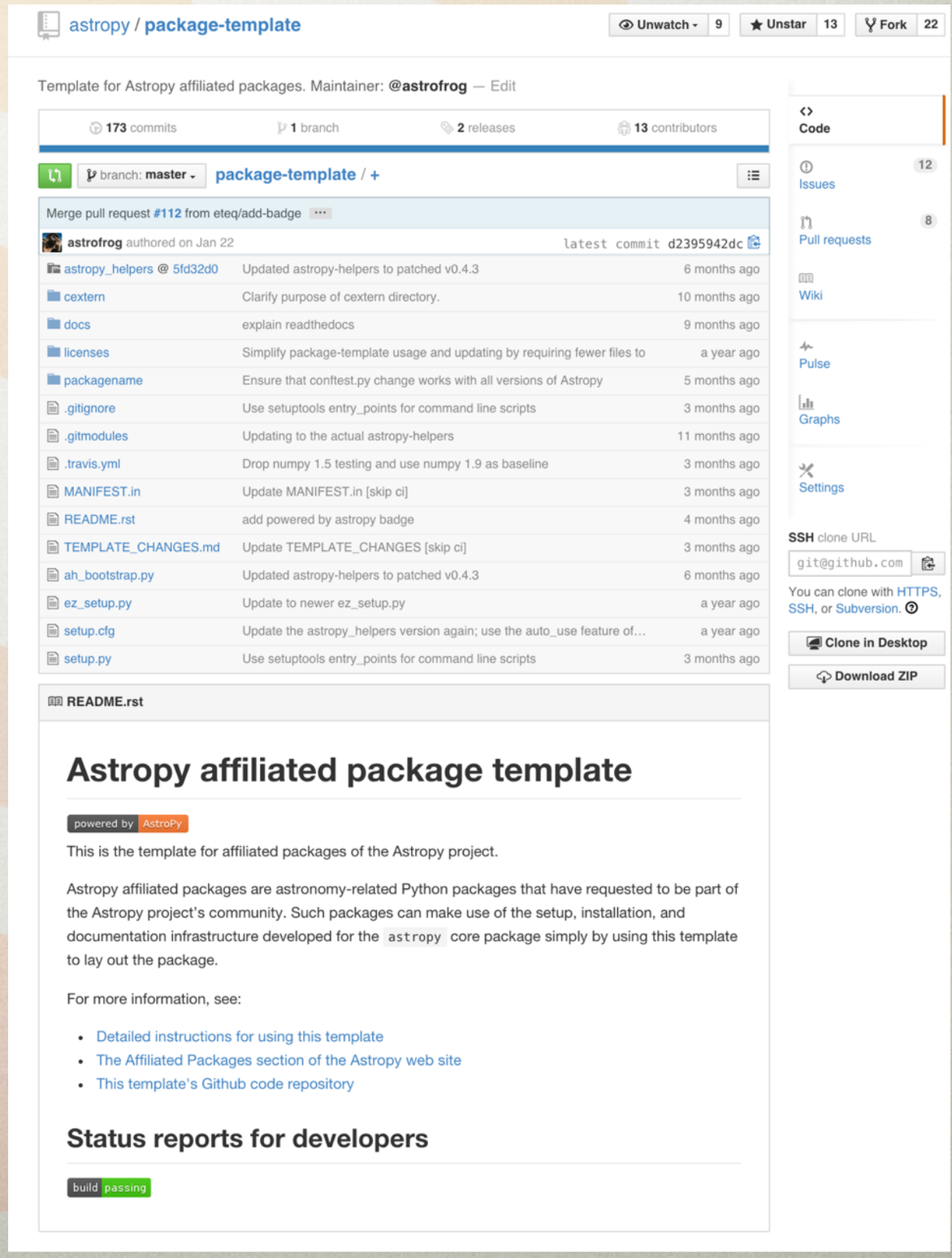
- *APLpy*: astronomical plotting
- *astroquery*: access to internet-accessible astronomy resources
- *ginga*: interactive image viz
- *imexam*: quick image analysis
- *pydl*: simple IDL ports
- *PyVO*: VO access
- *WCSAxes*: WCS-aware matplotlib plots
- *pyregion*: ds9 region files
- *montage-wrapper*: image mosaicing
- *ccdproc*: ccd reductions
- *photutils*: photometry
- *specutils*: spectroscopy
- *gammapy*: gamma-ray astronomy
- *sncosmo*: supernova light curves fitting/typing/etc
- *halotools*: high-performance tools for using n-body simulations to model galaxy formation
- *galpy*: tools for Galactic dynamics

What Unites Affiliated Packages?

- A common goal and vision: reducing duplication and embracing good coding practices (testing, docs), open development
- Listing on http://affiliated.astropy.org (curated and reviewed by the Astropy coordination committee - more on that later)
- (For many) a package template

Astropy Package Template

- Contains a ready-to-go "copy" of the astropy package layout. (Leans heavily on *astropy-helpers*)
- Provides documentation tools, testing framework, cython, configuration, etc.
- Docs on how to actually make it all work!



Changes to This Scheme that are In Progress

Affiliated packages -> “Coordinated” + affiliated

More “a la carte” helpers

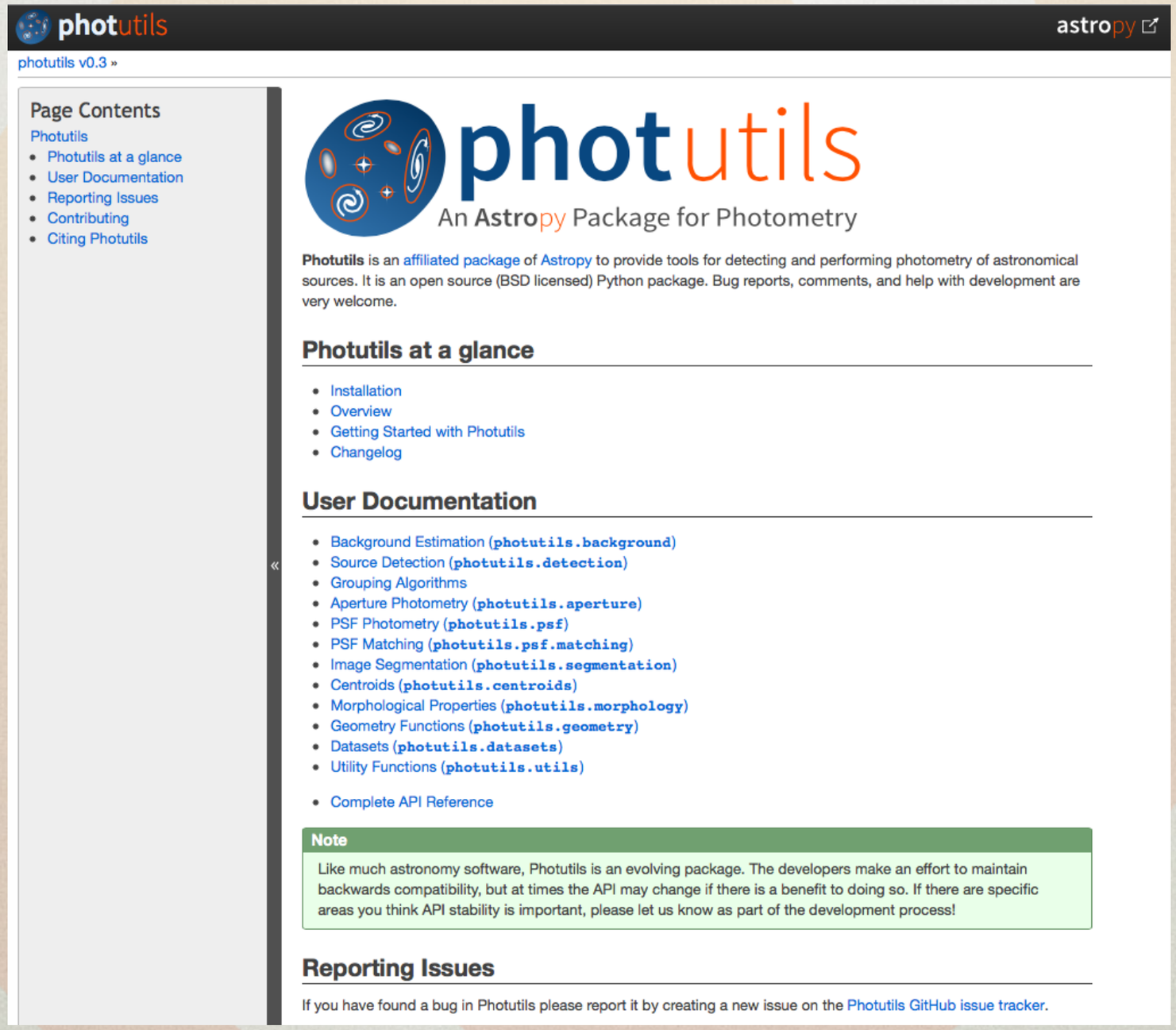
Affiliated package reviews that are more peer-review instead of “smoke-filled room” model

Affiliated Packages

Package	Functionality	Integration	Documentation	Devstatus	Python 3 Support
APLpy	🟢	🟢	🟢	🟡	🔴
astroML	🟢	🟢	🟡	🟡	🟢
astroquery	🟢	🟢	🟡	🟡	🟢
ccdproc	🟢	🔴	🟢	🟡	🔴
gammapy	🟢	🟢	🟢	🟡	🔴
ginga	🟢	🟢	🟡	🟡	🔴
glueviz	🟢	🟢	🟡	🟡	🔴
gwcs	🟢	🟢	🔴	🟢	🟢
imexam	🟢	🟢	🟡	🔴	🔴
montage-wrapper	🟡	🟢	🟡	🟢	🔴
photutils	🟢	🟢	🟡	🟡	🟢
pydl	🟢	🟢	🟡	🟢	🔴
pyregion	🔴	🟢	🟡	🟡	🟢
pyvo	🟡	🟢	🟢	🟡	🔴
reproject	🔴	🟢	🟡	🟡	🟢
sncosmo	🟢	🟢	🟡	🟢	🔴
spectral-cube	🟢	🟢	🟡	🔴	🔴
specutils	🟢	🟢	🟡	🟡	🔴
spherical-geometry	🟢	🔴	🟡	🟡	🔴
wcsaxes	🟢	🟡	🔴	🟡	🔴

Lets Dive into a few Affiliated packages that might be of particular interest here

Photutils: Community Photometry Tools



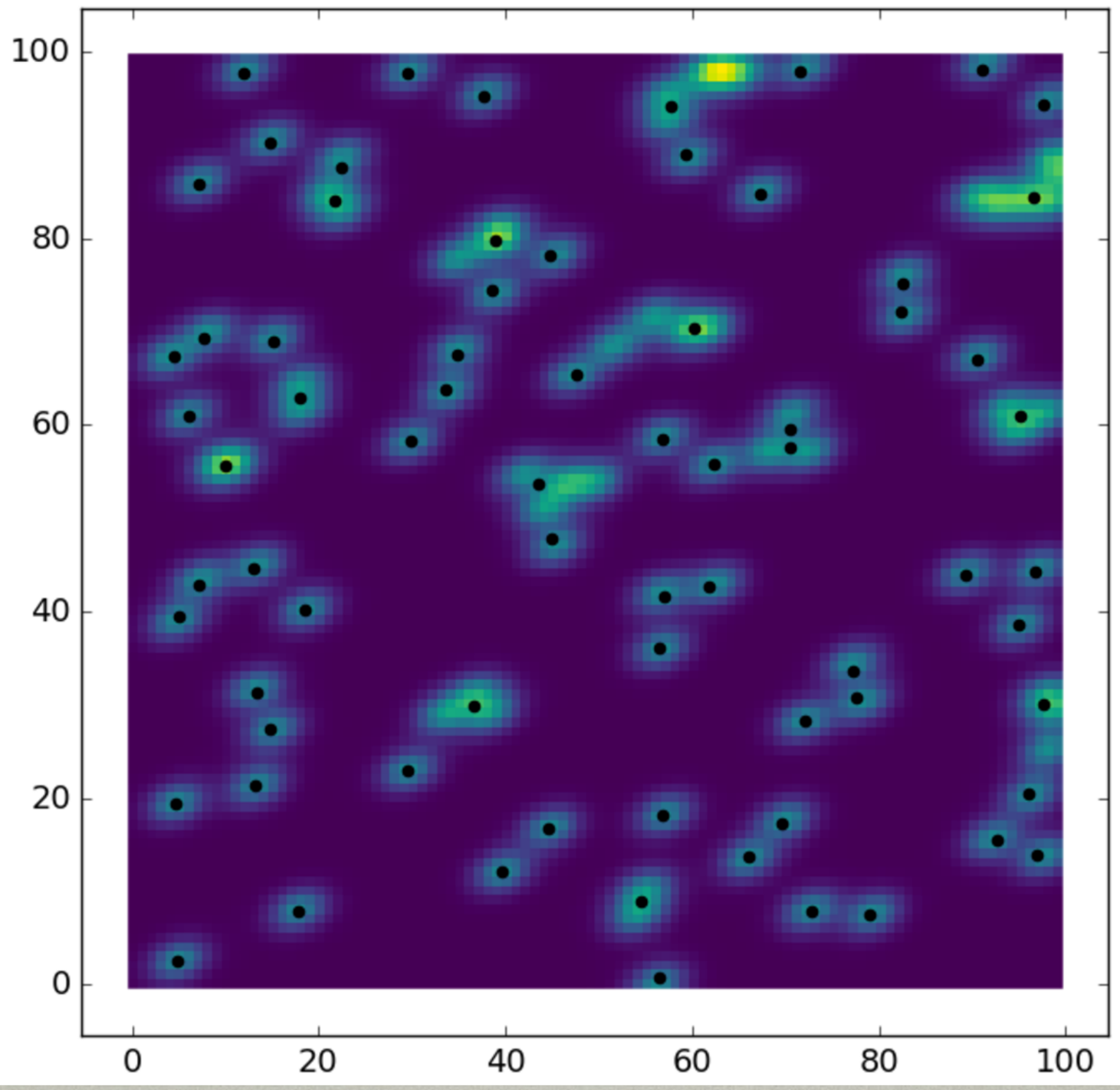
Feel free to check out

http://bit.ly/jwst-phot-tutorials

v0.3, http://photutils.readthedocs.io

Photutils: Object-Finding

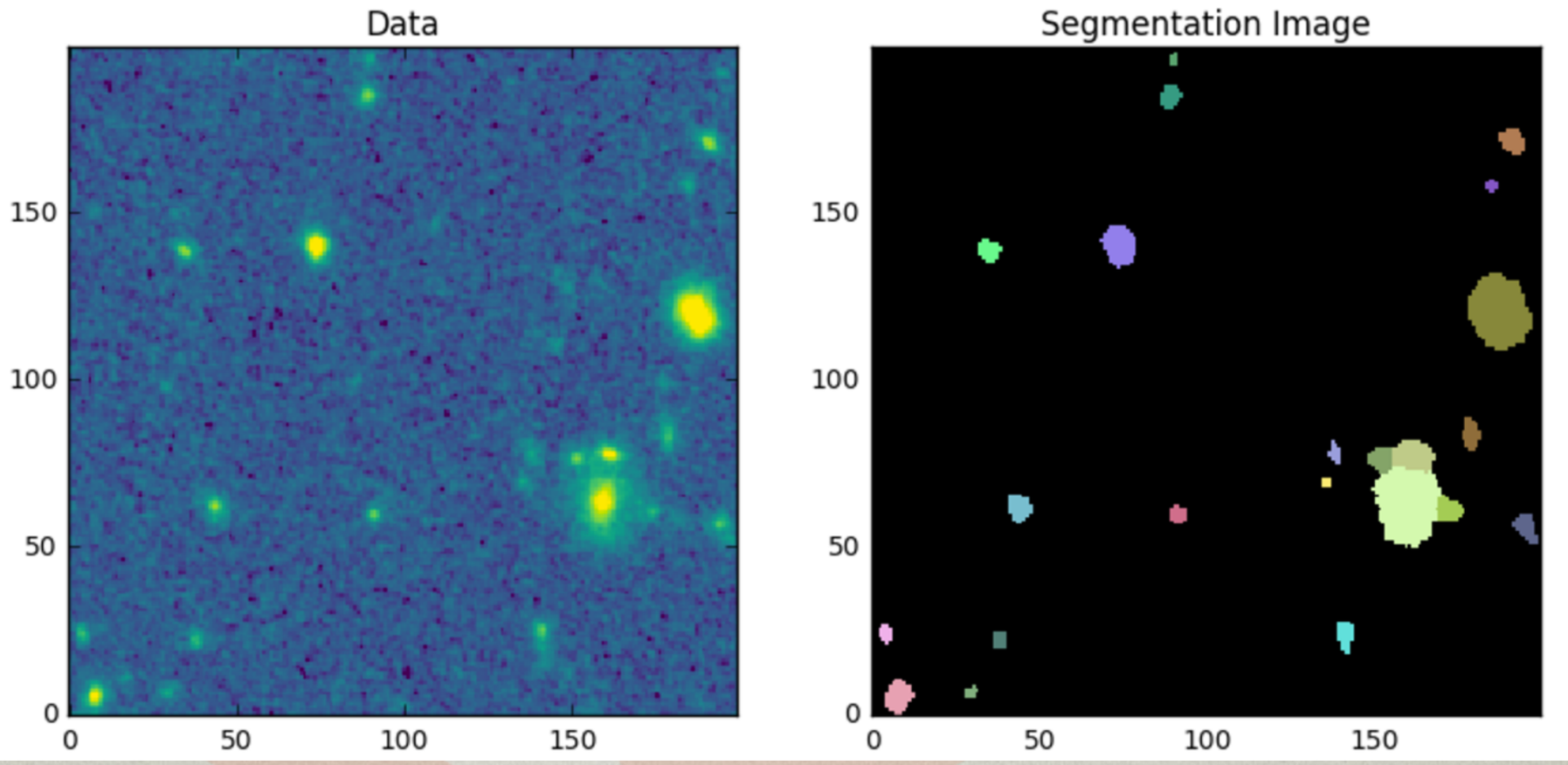
In [21]:
```
star_finder = photutils.findstars.DAOStarFinder(threshold=bkg_var/2,  fwhm=5)
found_stars = star_finder(im)

plt.imshow(im)
plt.scatter(found_stars['xcentroid'], found_stars['ycentroid'], color='k')
None
```



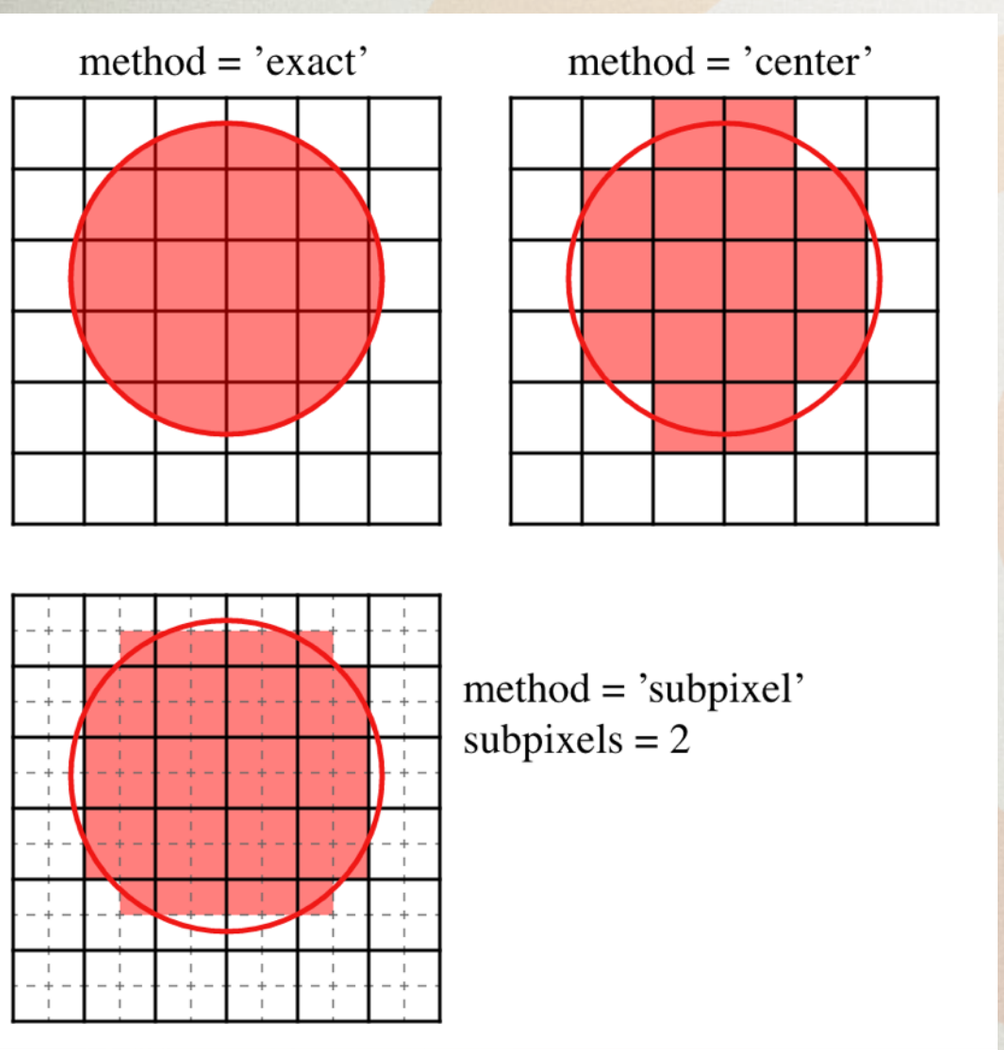
Photutils: Object-Finding

In [61]:
```
from photutils import deblend_sources

segm2 = deblend_sources(data, segm, npixels, filter_kernel=kernel,
                        contrast=0.001, nlevels=32)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 8))
ax1.imshow(scale_image(data, scale='sqrt', percent=99.5))
ax1.set_title('Data')
ax2.imshow(segm2, cmap=rand_cmap)
ax2.set_title('Segmentation Image')

print('Found {0} sources'.format(segm2.max))
```

```
Found 22 sources
```



Photutils: Aperture Photometry



In [41]:
```
from photutils import CircularAnnulus

positions = [(90.73, 59.43), (73.63, 139.41), (43.62, 61.63)]

aper = CircularAperture(positions, r=3)
bkg_aper = CircularAnnulus(positions, r_in=10., r_out=15.)
apers = [aper, bkg_aper]
```

Now, perform the photometry.

In [42]:
```
phot = aperture_photometry(data, apers)
phot.rename_column('aperture_sum_0', 'aperture_sum')
phot.rename_column('aperture_sum_1', 'annulus_sum')
phot
```

Out[42]:

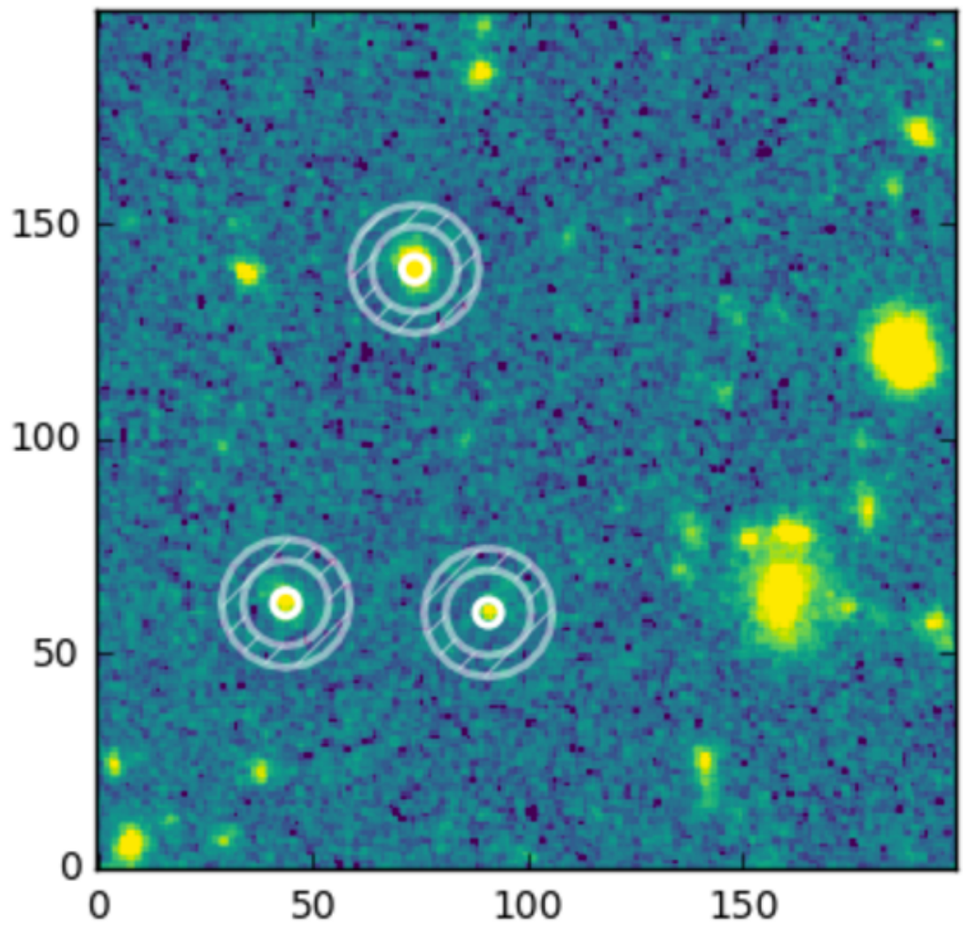
<QTable length=3>

id	xcenter	ycenter	aperture_sum	annulus_sum
	pix	pix		
int64	float64	float64	float64	float64
1	90.73	59.43	0.0866436609693	0.0199107563833
2	73.63	139.41	0.393646538117	0.0358905305285
3	43.62	61.63	0.130109734904	0.0166684757391

Photutils: Aperture Photometry

Apertures can plot themselves

In [46]:
```
plt.imshow(scale_image(data, scale='sqrt', percent=98.))

aper.plot(color='white', lw=2)
bkg_aper.plot(color='white', lw=2, hatch='//', alpha=0.5)
```



Photutils: PSF Photometry

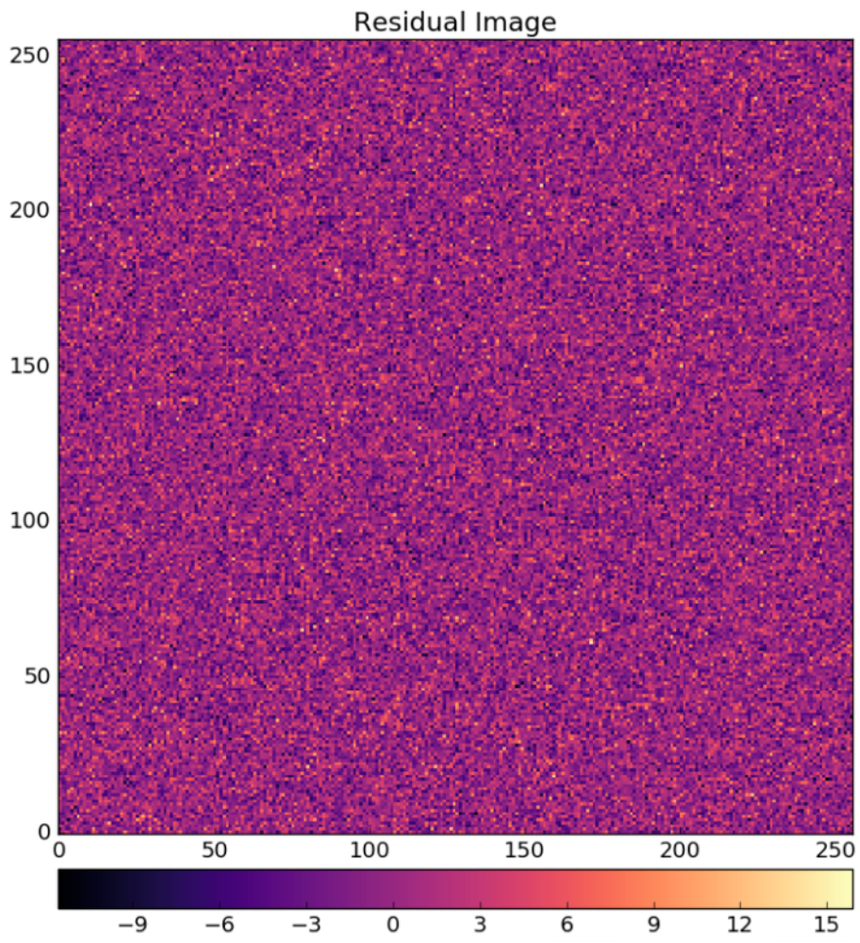
In [20]:
```
from astropy.table import Table
positions = Table(names=['x_0', 'y_0'], data=[starlist['x_mean'], starlist['y_mean']])
```

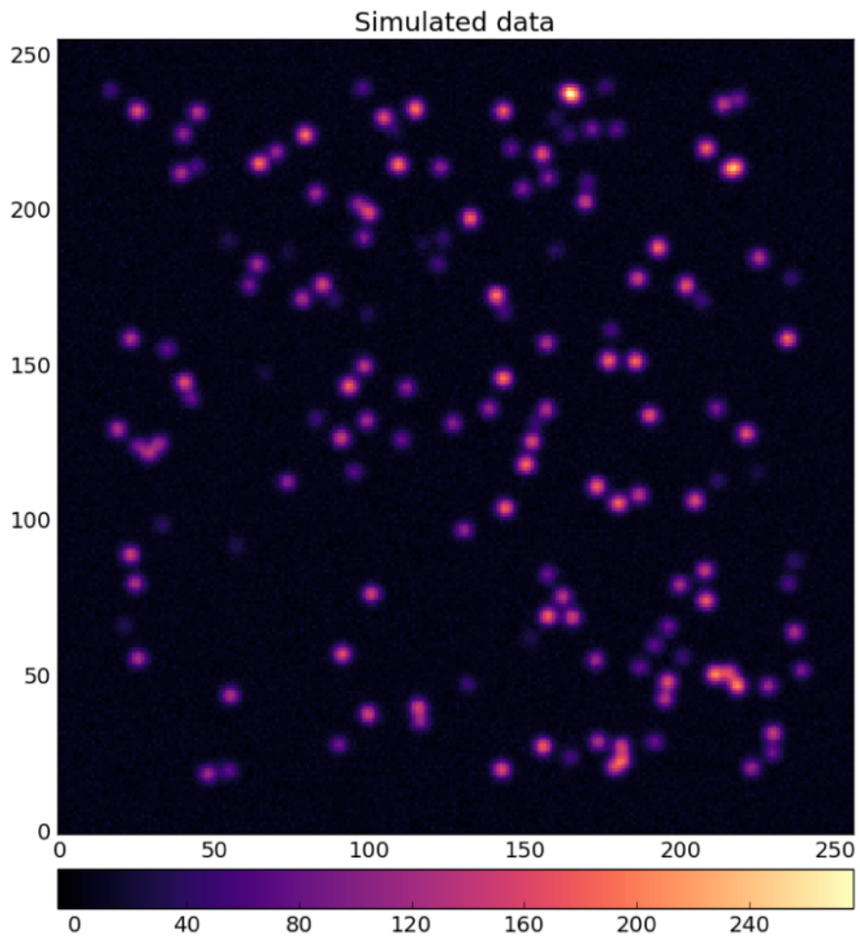
In [21]:
```
photometry_results = basic_photometry(image=image, positions=positions)
```

```
WARNING: Both positions and finder are different than None, which is ambiguous. finder is going to be ignored. [photu
tils.psf.photometry]
```

In [22]:
```
plt.subplot(1,2,1)
plt.imshow(basic_photometry.get_residual_image())
plt.title('Residual Image')
plt.colorbar(orientation='horizontal', fraction=0.046, pad=0.04)

plt.subplot(1,2,2)
plt.imshow(image)
plt.title('Simulated data')
plt.colorbar(orientation='horizontal', fraction=0.046, pad=0.04)
```

Out[22]: <matplotlib.colorbar.Colorbar at 0x11488be48>





Specutils

Specutils

Specutils is an `Astropy`_ affiliated package with the goal of providing a shared set of Python representations of astronomical spectra and basic tools to operate on these spectra. The effort is also meant to be a "hub", helping to unite the Python astronomical spectroscopy community around shared effort, much as `Astropy`_ is meant to for the wider astronomy Python ecosystem.

First Steps

The Spectrum1D class is one of the core classes of the specutils package. You can import it like this:

```
>>> from specutils import Spectrum1D
```

To instantiate it you can define a wave and a flux:

```
>>> wave = np.arange(6000, 9000) * u.Angstrom
>>> flux = np.random.random(3000) * u.Unit('W m-2 angstrom-1 sr-1')
```

and then call the **from_array** method:

```
>>> spec1d = Spectrum1D.from_array(wave, flux)
>>> spec1d.wavelength
<Quantity [ 6000., 6001., 6002.,...,  8997., 8998., 8999.] Angstrom>
>>> spec1d.flux
<Quantity [ 0.75639906, 0.23677036, 0.08408417,...,  0.82740303, 0.38345114,
            0.77815595] W / (Angstrom m2 sr)>
```

Or you can read a Spectrum from a .fits file with the **read_fits** method:

```
>>> from specutils.io import read_fits
>>> myspec = read_fits.read_fits_spectrum1d('myfile.fits')
```

Specutils

Specutils

Specutils is an `Astropy`_ affiliated package with the goal of providing a shared set of Python representations of astronomical spectra and basic tools to operate on these spectra. The effort is also meant to be a “hub”, helping to unite the Python astronomical spectroscopy community around shared effort, much as `Astropy`_ is meant to for the wider astronomy Python ecosystem.

First Steps

The Spectrum1D class is one of the core classes of the specutils package. You can import it like this:

```
>>> from specutils import Spectrum1D
```

To instantiate it you can define a wave and a flux:

```
>>> wave = np.arange(6000, 9000) * u.Angstro
>>> flux = np.random.random(3000) * u.Unit('
```

and then call the **from_array** method:

```
>>> spec1d = Spectrum1D.from_array(wave, flu
>>> spec1d.wavelength
<Quantity [ 6000., 6001., 6002.,...,  8997.,
>>> spec1d.flux
<Quantity [ 0.75639906, 0.23677036, 0.084084
            0.77815595] W / (Angstrom m2 sr)
```

Or you can read a Spectrum from a .fits file with the **read_fits**

```
>>> from specutils.io import read_fits
>>> myspec = read_fits.read_fits_spectrum1d(
```

Specutils APE #28

Edit

Open **crawfordsm** wants to merge 20 commits into `astropy:master` from `crawfordsm:specutils_take_2`

Conversation 17 · Commits 20 · Files changed 1 · +487 −0

crawfordsm commented on Oct 2 · Member

This is an updated version of the APE for `specutils`. We have started a new pull request for a fresh discussion after incorporating suggestions from the specutils coordination meeting and comments from the previous pull request, #23. We believe we have addressed many of the comments that were made, and incorporated many of the ideas shared.

Hopefully this meets many of the requirements that people were interested in, but we would be very happy to receive further comments on this version.

If you have commented, participated in the meeting, or would like to co-sign this APE, please feel free to leave your name in the comments or add it via a pull request.

Thank you to everyone for your comments and your patience as we created the updated document.

👍 1

Reviewers
Cadair

Assignees
No one—assign yourself

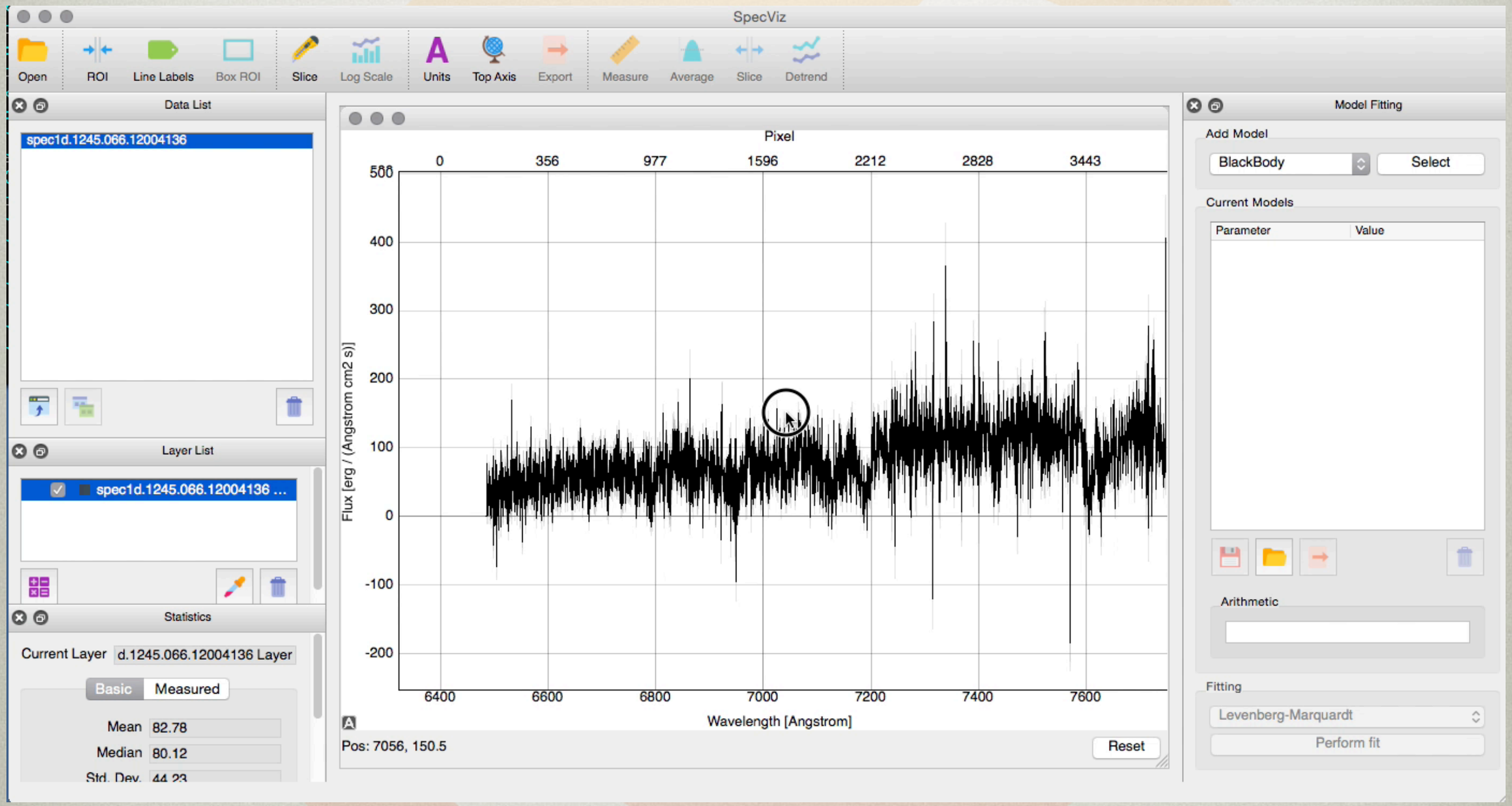
Labels
None yet

Projects
None yet

Milestone
No milestone

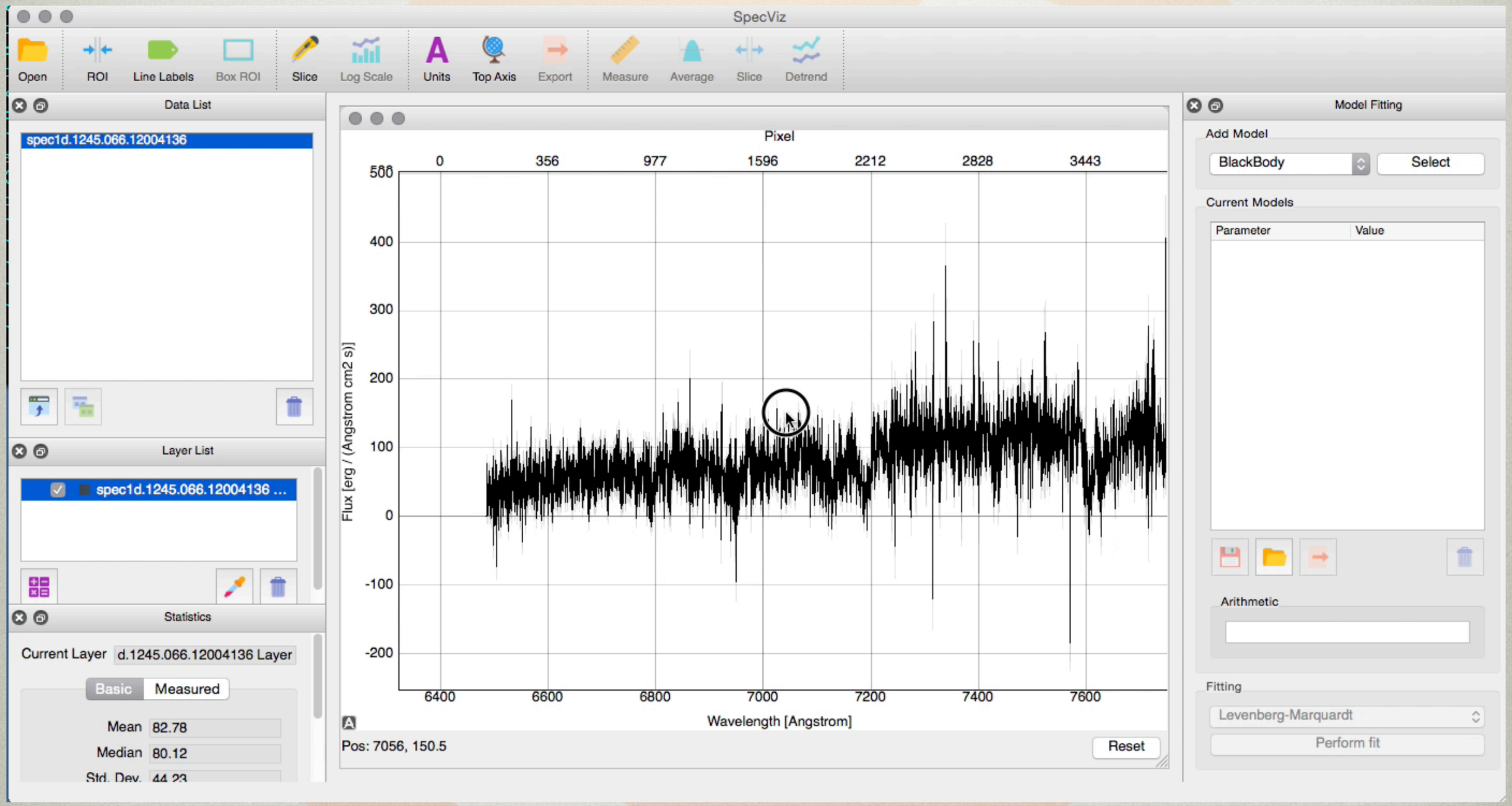
https://github.com/astropy/astropy-APEs/pull/28

SpecViz

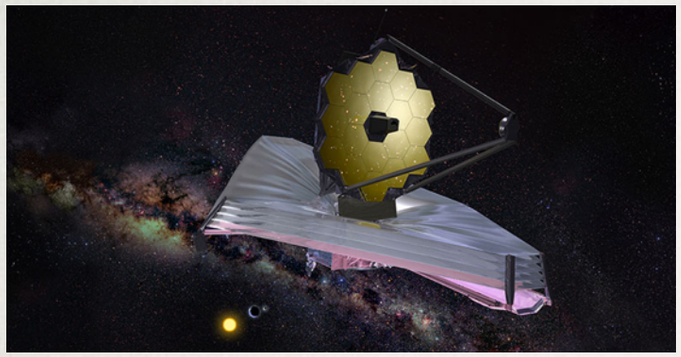


v1.0, http://specviz.readthedocs.io

SpecViz



v1.0, http://specviz.readthedocs.io

Affiliated Packages for  JWST

Most JWST data analysis tools will either be actual affiliated packages, or in the same framework

- *photutils*: Aperture and PSF photometry for NIRCam, MIRI
- *specviz, mosviz, cubeviz:* Spectroscopic visualization for NIRSpec
- *specutils*: "Gateway" to connect the above to other spectra
- *gwcs:* Sky <-> Pixel supporting distortions, IFUs, etc.
- *webbpsf & poppy*: Optical modeling of the JWST PSF (or other ST's!)
- (future) *jwst_psftools, jwst_autoreduce,* etc.

These All Aim To Be Open Developed

astropy / **photutils**

Unwatch 19 | Star 54 | Fork 65

Code | Issues 30 | Pull requests 7 | Projects 0 | Wiki | Settings | Insights

Affiliated package for image photometry utilities. Maintainers: **@larrybradley** and **@bsipocz**
http://photutils.readthedocs.io

Edit

astronomy | astropy | astropy-affiliated | photometry | python | source-detection | Manage topics

2,561 commits | **3** branches | **7** releases | **26** contributors | BSD-3-Clause

astropy / **specutils**

Unwatch 21 | Unstar 37 | Fork 54

Code | Issues 25 | Pull requests 6 | Projects 0 | Wiki | Settings | Insights

Affiliated package for 1D spectral operations. Maintainer: **@crawfordsm @keflavich @nmearl**

Edit

astropy | Manage topics

644 commits | **3** branches | **4** releases | **27** contributors

spacetelescope / **specviz**

Watch 6 | Star 5 | Fork 11

Code | Issues 37 | Pull requests 1 | Projects 1 | Wiki | Settings | Insights

An interactive astronomical 1D spectra analysis tool. http://specviz.readthedocs.io

Edit

astronomy | spectroscopy | viewer | data-analysis | python | Manage topics

740 commits | **4** branches | **12** releases | **10** contributors

Be welcoming and respectful, and build an open and understandable process that everyone shares. If your problem is shared by others, the cats might just herd themselves.

Astropy's Development

(The spheres are files, and the avatars are contributors.)