# astropy

A Community Python Library for Astronomy

http://www.astropy.org


**Erik Tollerud**

**@eteq**

Yale University

Astropy Coordinating Committee Member

Hubble Fellow


**Scicoder 2015**

# Astropy's Origin Story

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

**A. Use any of:**

- pyast
- Astrolib
- Astropysics
- Kapteyn
- EphemPy
- 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).

The Result: astropy

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

# The astropy Project and package

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

Core package “astropy” ≠ “Astropy project”

Core package is what’s in github repo *astropy/astropy*

Project includes all the affiliated packages and associated community

Three-member coordination committee: currently me, Tom R, Perry

# What is the philosophy behind the code?

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

# What is the philosophy behind the code?

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

# What is the philosophy behind the code?

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

# What is the philosophy behind the code?

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

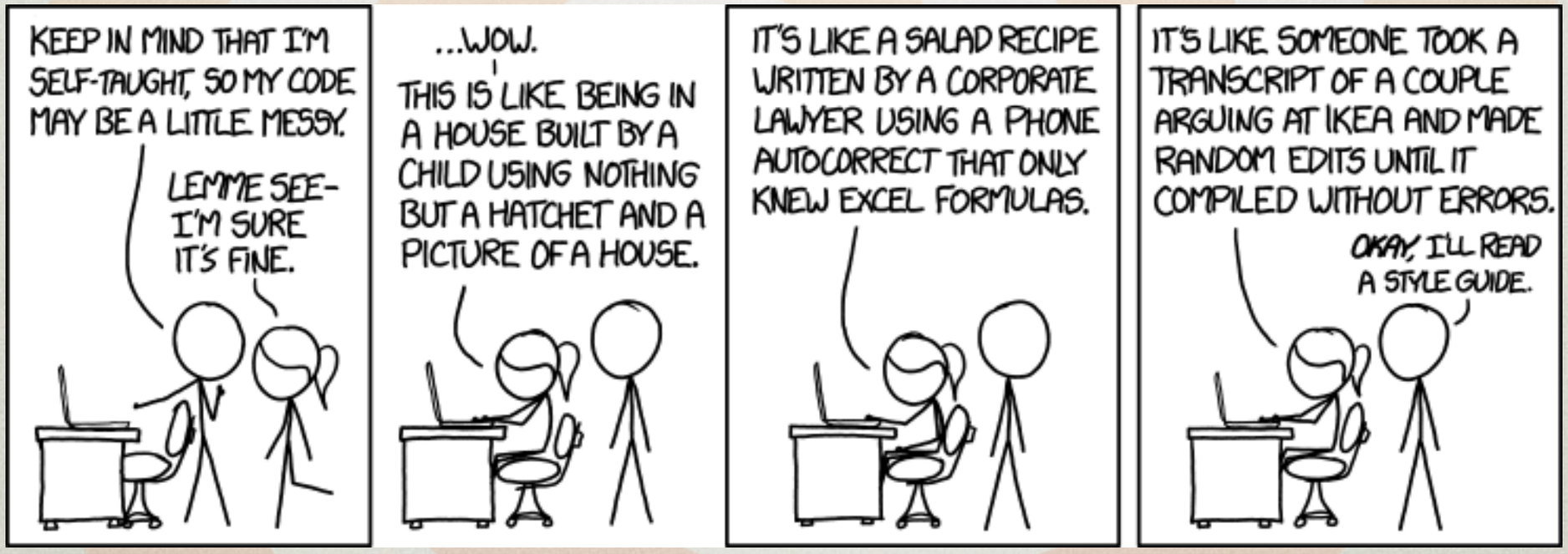
This means both *by* and *for* the community

(Professional) Astronomers help write it

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

# What is the philosophy behind the code?

But at the same time, it should address this problem:




By showing how things *can* be done!

# What's in the Astropy Core Package

Best place to look is always

http://docs.astropy.org

# What's in the Astropy Core Package

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 (and other astro) coordinates
- Image analysis and interoperability data structures.
- WCS
- Data download and caching tools
- FITS, VOTable, hdf5, extensible I/O
- Table manipulation, including many arcane astro formats
- Cosmology tools
- Astrostatistics convenience functions
- Data modeling and fitting
- Configuration and "plumbing" to make it all work

# Affiliated Packages

An affiliated package is an astronomy-related Python package that is not part of the astropy core package, but has requested to be included as part of the Astropy project's community.

# Affiliated Packages

Best place to look is always
http://affiliated.astropy.org

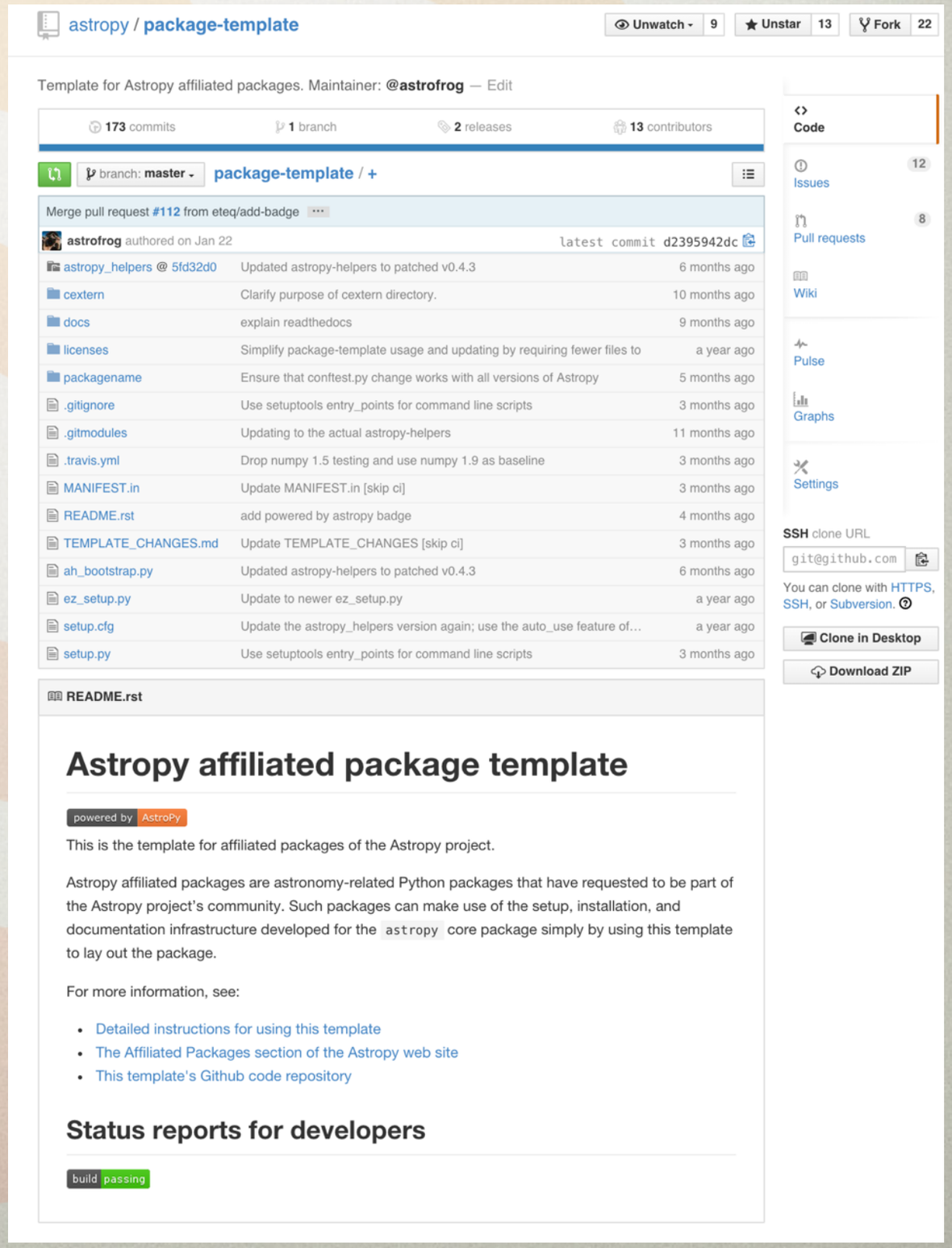
- APLpy: astronomical plotting
- astroML: astro machine learning (companion to a textbook)
- astroquery: access to internet resources
- ccdproc: ccd reductions
- gammapy: gamma-ray astronomy
- ginga: interactive image viz
- imexam: quick image analysis
- montage-wrapper: image mosaicing
- photutils: photometry
- pydl: simple IDL ports
- pyregion: ds9 region files
- PyVO: VO access
- sncosmo: supernova light curves fitting/typing/etc
- specutils: spectroscopy
- spherical_geometry: spherical polygons/regions
- WCSAxes: WCS-aware matplotlib plots

# What Unites them?

- A Common goal and vision: reducing duplication and embracing good coding practices (testing, docs)
- (For many) a package template

# Affiliated 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!



# Resources to Learn More about Astropy

Talk to me, Adrian, or local "Astropy ambassador"

http://www.astropy.org

Astropy mailing list

"Python users in astronomy" FB group

http://docs.astropy.org

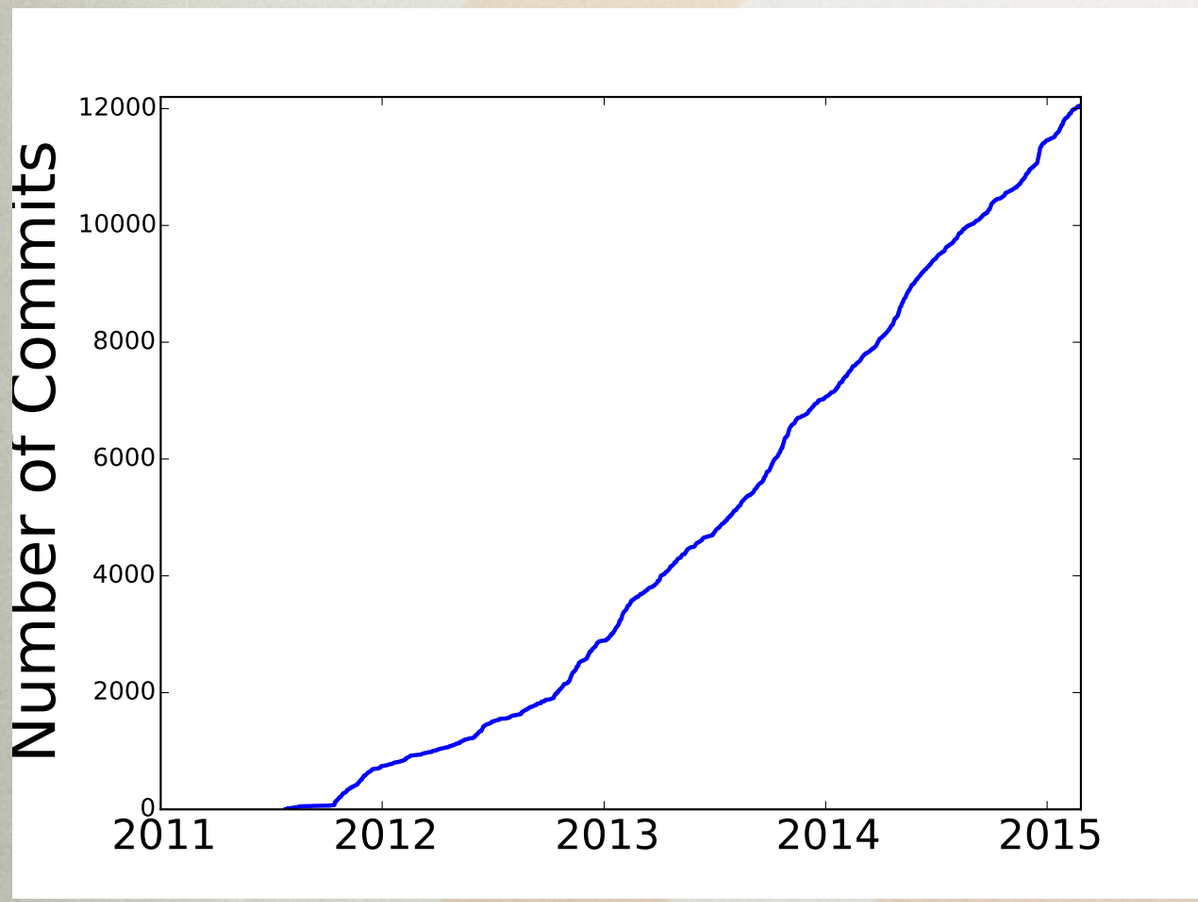
http://tutorials.astropy.org

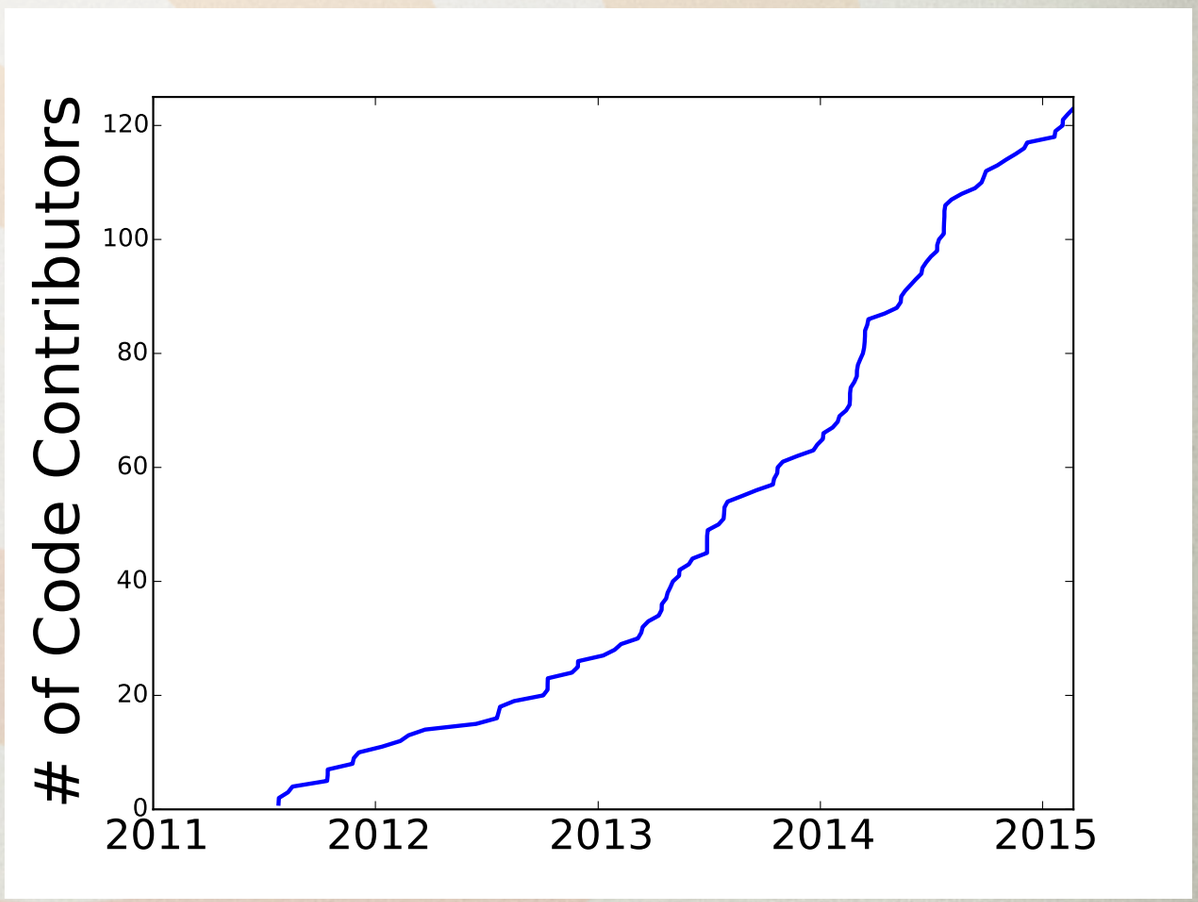
astropy-dev mailing list

# The “By” part

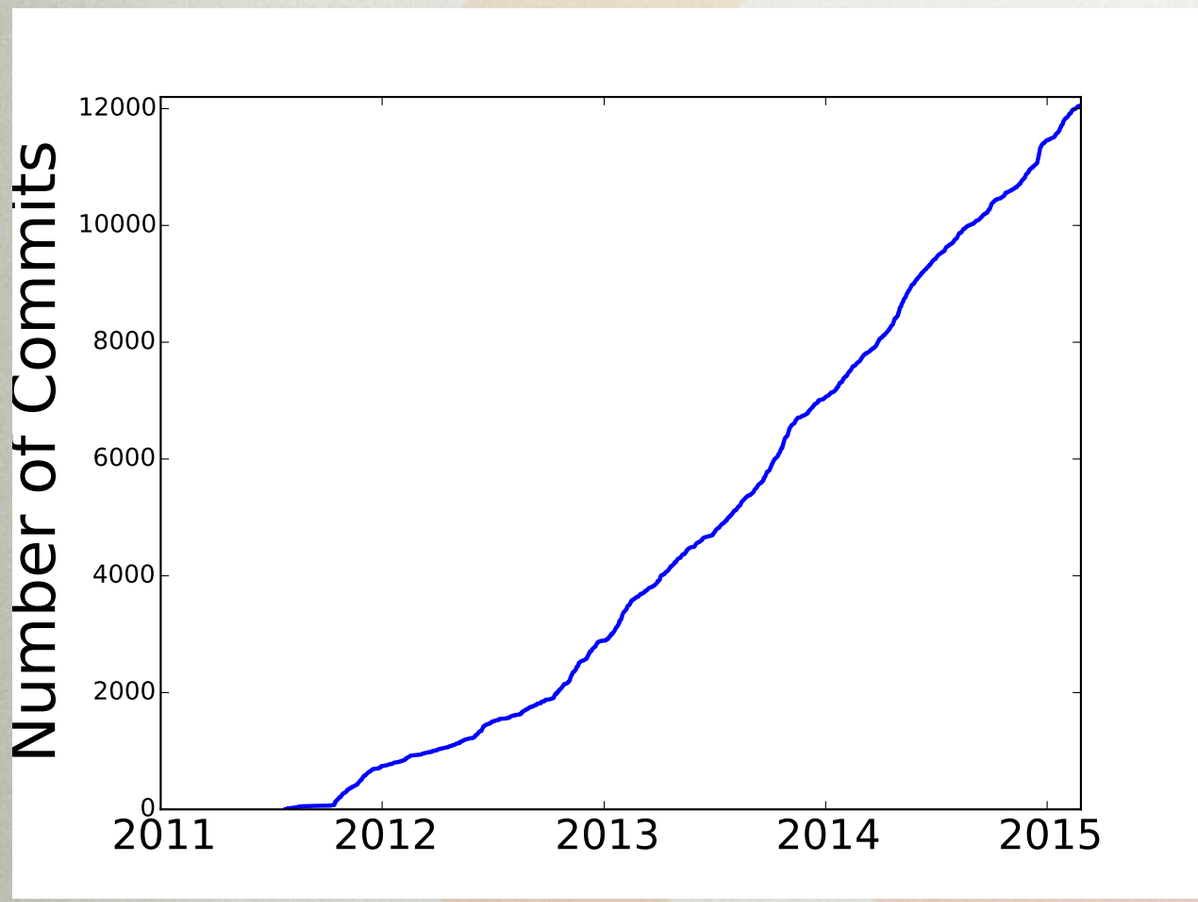
The Astropy Project is a community effort to develop a single core package for Astronomy in Python and foster interoperability between Python astronomy packages.

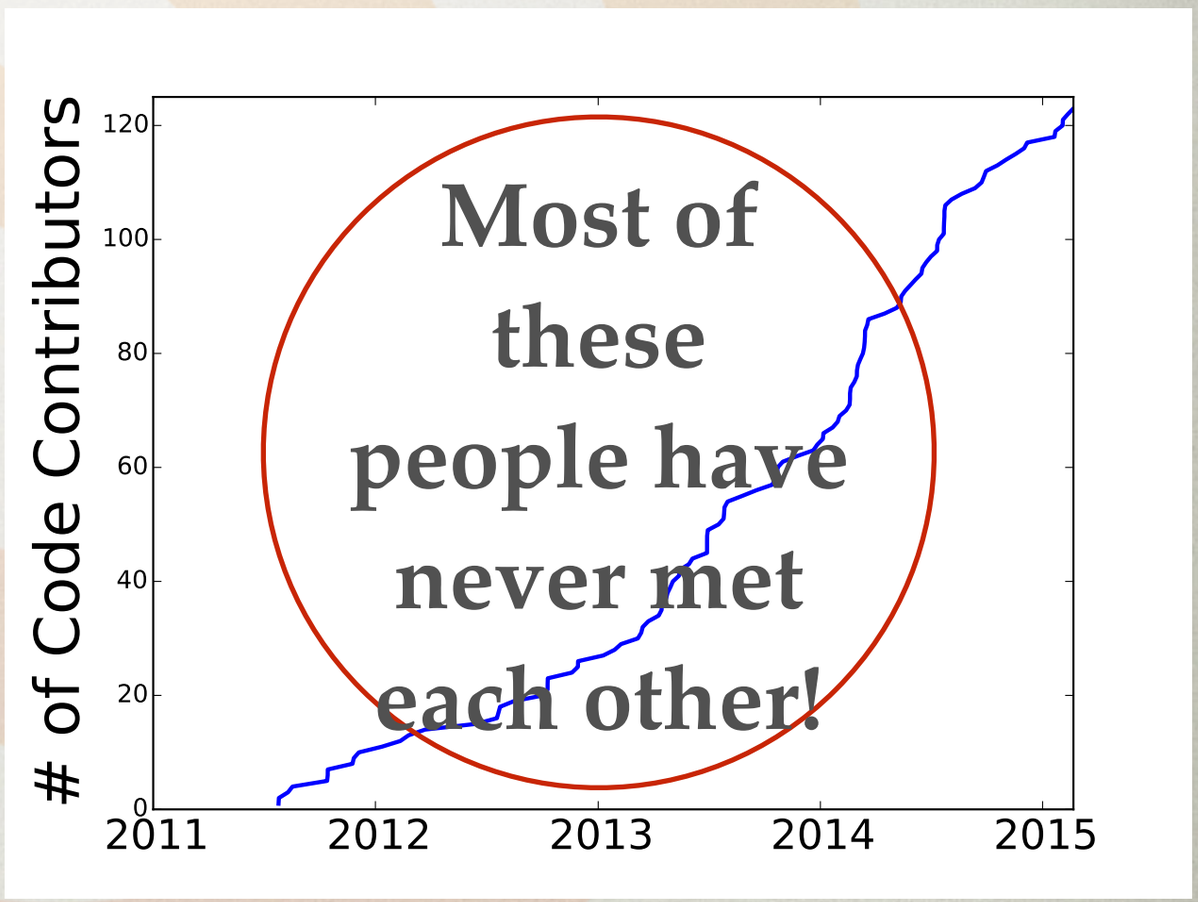
# Contributions are Growing



# Contributions are Growing



# What are the Key pieces for astropy's devlopment??

# What are the Key Elements?

github
SOCIAL CODING

astropy / astropy

Unwatch 86 | Unstar 530 | Fork 375

## Issue 3601 -- astropy.stats.funcs.bootstrap now accepts bootfuncs with multiple outputs #3628

Edit

Open **ezbc** wants to merge 7 commits into `astropy:master` from `ezbc:issue3601`

Conversation 8 | Commits 7 | Files changed 3 | +115 −6

**ezbc** commented 26 days ago

I have addressed Issue 3601. I implemented the option for users to supply a function with multiple outputs to `bootfunc`. They can control which `bootfunc` outputs to retain with `output_index`.

This function could be sped up if the `if` statements were moved outside of the loop.

**ezbc** added some commits 26 days ago

- `initial commit` 84afdce
- `fully functional, needs indices to be output_index` b69447d
- `bootstrap: reworked indices variable to be more pythonic` e7ac659
- `updated changes log` b7c1515
- `updated changes log for issue3601` 9e01308

**embray** added stats Affects-release labels 26 days ago

**embray** commented 26 days ago — Collaborator

@ezbc Your PR so far has the changelog entry and some tests which look good, but is missing the actual change to the function. Is it intentionally not implemented yet?

- `definite commit of funcs.py` 594ddd0

Labels: Affects-release, stats

Milestone: v1.1.0

Assignee: No one—assign yourself

Notifications: Unsubscribe

You're receiving notifications because you were mentioned.

5 participants

Lock pull request

# What are the Key Elements?

**embray** commented 26 days ago — Collaborator

**@ezbc** Your PR so far has the changelog entry and some tests which look good, but is missing the actual change to the function. Is it intentionally not implemented yet?

5 participants

Lock pull request

`definite commit of funcs.py` ✖ 594ddd0

**ezbc** commented 26 days ago

**@embray** This was not intentional, sorry about that. I must have committed funcs to a different branch.

I believe the changes are now pushed to the correct branch, issue3601, and are now ready to be merged.

**bsipocz** commented on an outdated diff 26 days ago — Show outdated diff

**bsipocz** commented on the diff 26 days ago

astropy/stats/tests/test_funcs.py — View full changes

```
@@ -236,6 +238,60 @@ def test_bootstrap():
236 238         bootresult = np.mean(funcs.bootstrap(bootarr, 10000, bootfunc=np.m
237 239         assert_allclose(np.mean(bootarr), bootresult, atol=0.01)
238 240
    241 +   # test a bootfunc with several output values
    242 +   # return just bootstrapping with one output from bootfunc
    243 +   with NumpyRNGContext(42):
    244 +       bootarr = np.array([[1, 2, 3, 4, 5, 6, 7, 8, 9, 0],
    245 +                           [4, 8, 8, 3, 6, 5, 2, 8, 6, 2]]).T
    246 +
    247 +       answer = np.array((0.19425, 0.02094))
    248 +
    249 +       bootresult = funcs.bootstrap(bootarr, 2,
    250 +                                    bootfunc=spearmanr)
```

**bsipocz** added a note 26 days ago — Collaborator

I think you may want to put this new part in a separate function as it depends on scipy. The old test can run without scipy, and this new function gets the decorator and runs only when scipy is available.

Add a line note

# What are the Key Elements?

**embray** commented 26 days ago — Collaborator

**@ezbc** Your PR so far has the changelog entry and some tests which look good, but is missing the actual change to the function. Is it intentionally not implemented yet?

`definite commit of funcs.py` ✗ 594ddd0

**ezbc** commented 26 days ago

**@embray** This was not intentional, sorry about that. I must have committed funcs to a different branch.

I believe the changes are now pushed to the correct branch, issue3601, and are now ready to be merged.

**bsipocz** commented on an outdated diff 26 days ago — Show outdated diff

**bsipocz** commented on the diff 26 days ago

astropy/stats/tests/test_funcs.py — View full changes

```diff
@@ -236,6 +238,60 @@ def test_bootstrap():
236 238         bootresult = np.mean(funcs.bootstrap(bootarr, 10000, bootfunc=np.m
237 239         assert_allclose(np.mean(bootarr), bootresult, atol=0.01)
238 240
    241 +   # test a bootfunc with several output values
    242 +   # return just bootstrapping with one output from bootfunc
    243 +   with NumpyRNGContext(42):
    244 +       bootarr = np.array([[1, 2, 3, 4, 5, 6, 7, 8, 9, 0],
    245 +                           [4, 8, 8, 3, 6, 5, 2, 8, 6, 2]]).T
    246 +
    247 +       answer = np.array((0.19425, 0.02094))
    248 +
    249 +       bootresult = funcs.bootstrap(bootarr, 2,
    250 +                                    bootfunc=spearmanr)
```

**bsipocz** added a note 26 days ago — Collaborator

I think you may want to put this new part in a separate function as it depends on scipy. The old test can run without scipy, and this new function gets the decorator and runs only when scipy is available.

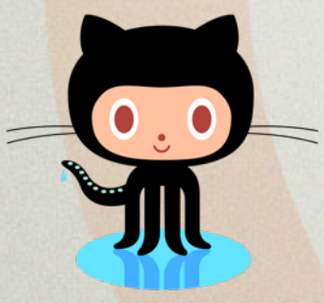
Add a line note

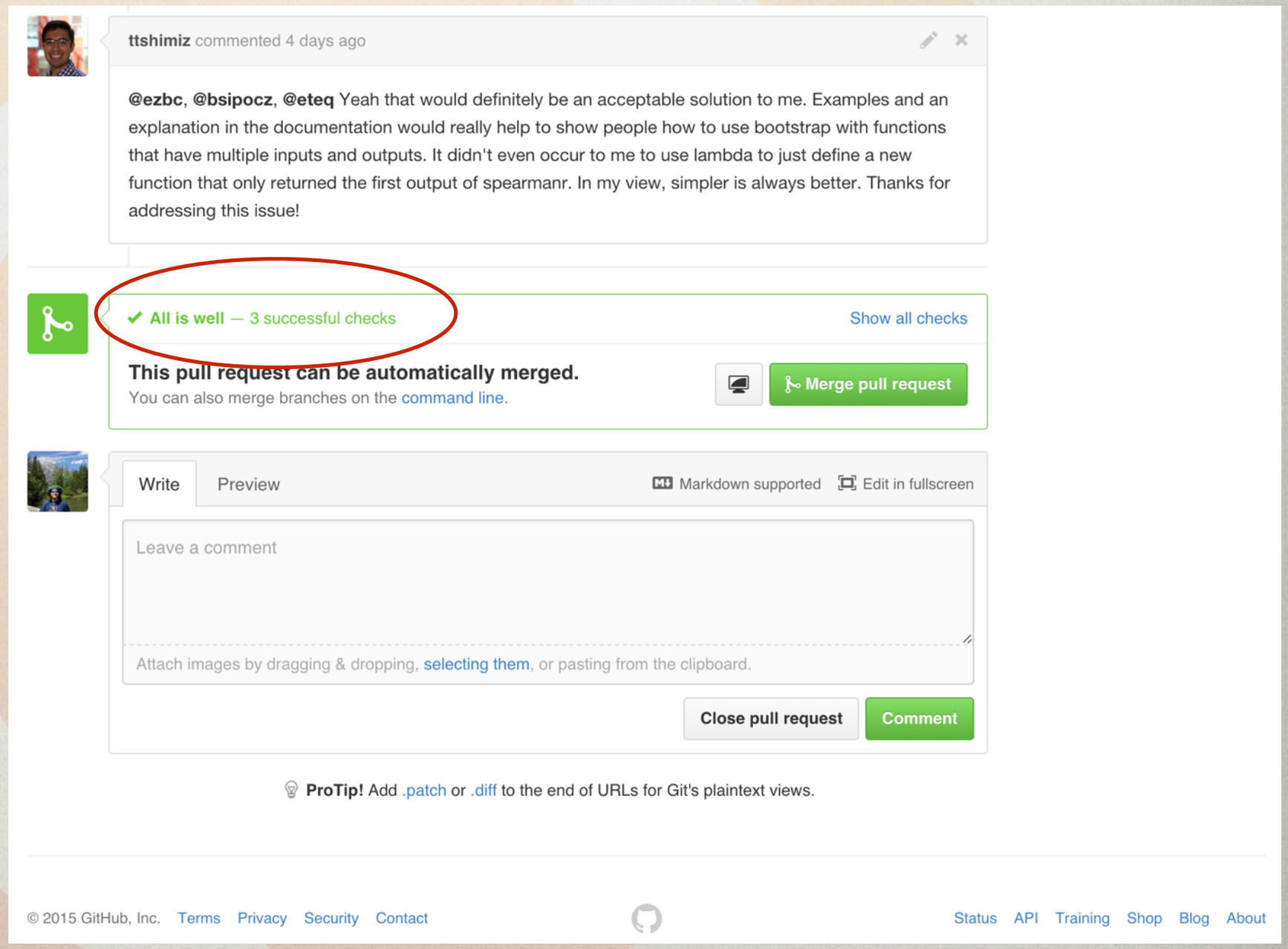
5 participants

Lock pull request

github SOCIAL CODING

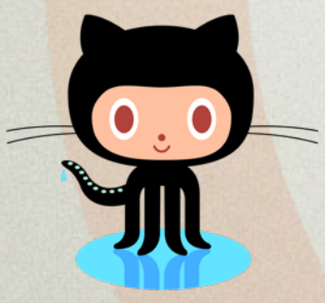
# What are the Key Elements?

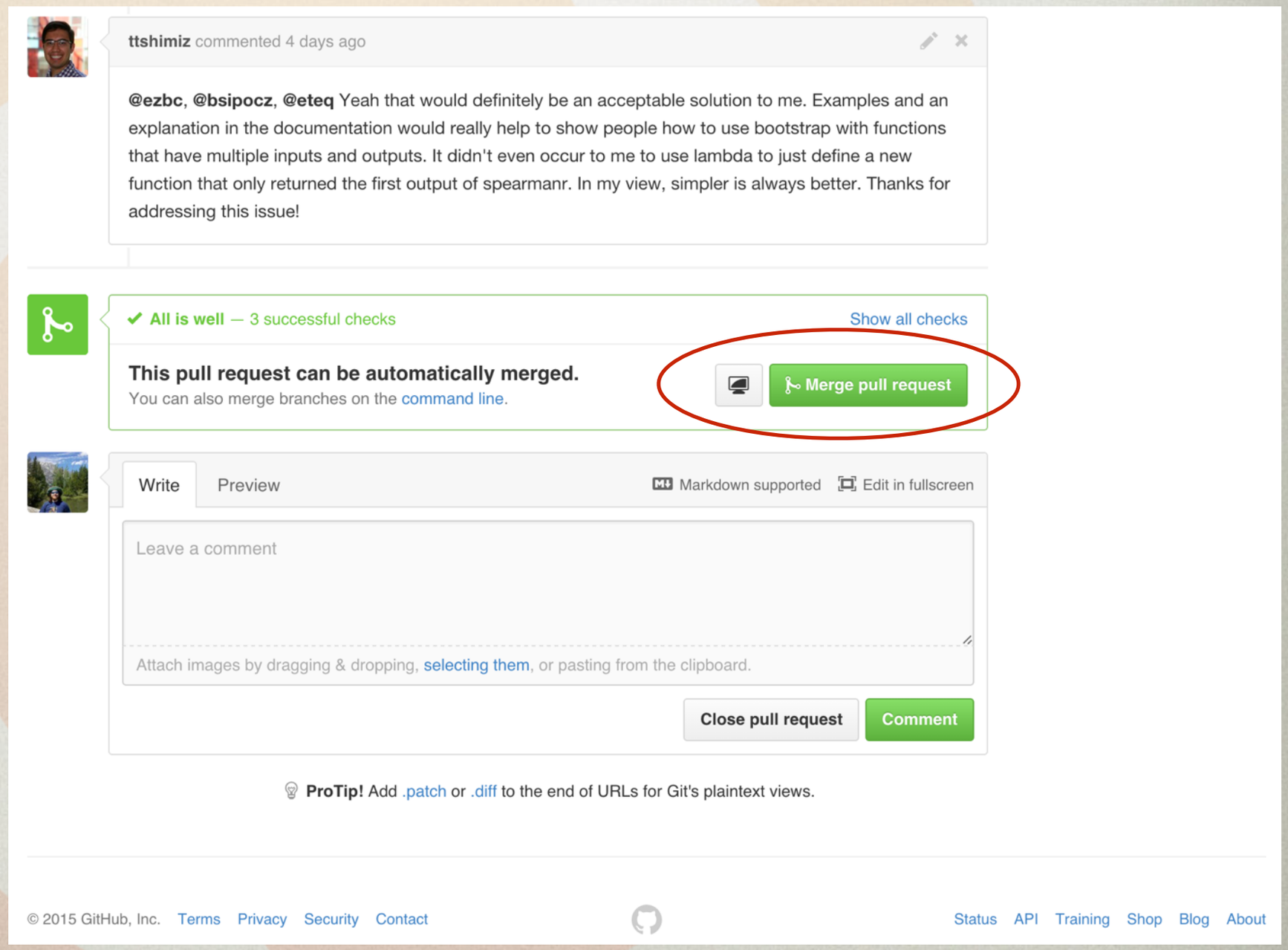





# What are the Key Elements?

github
SOCIAL CODING



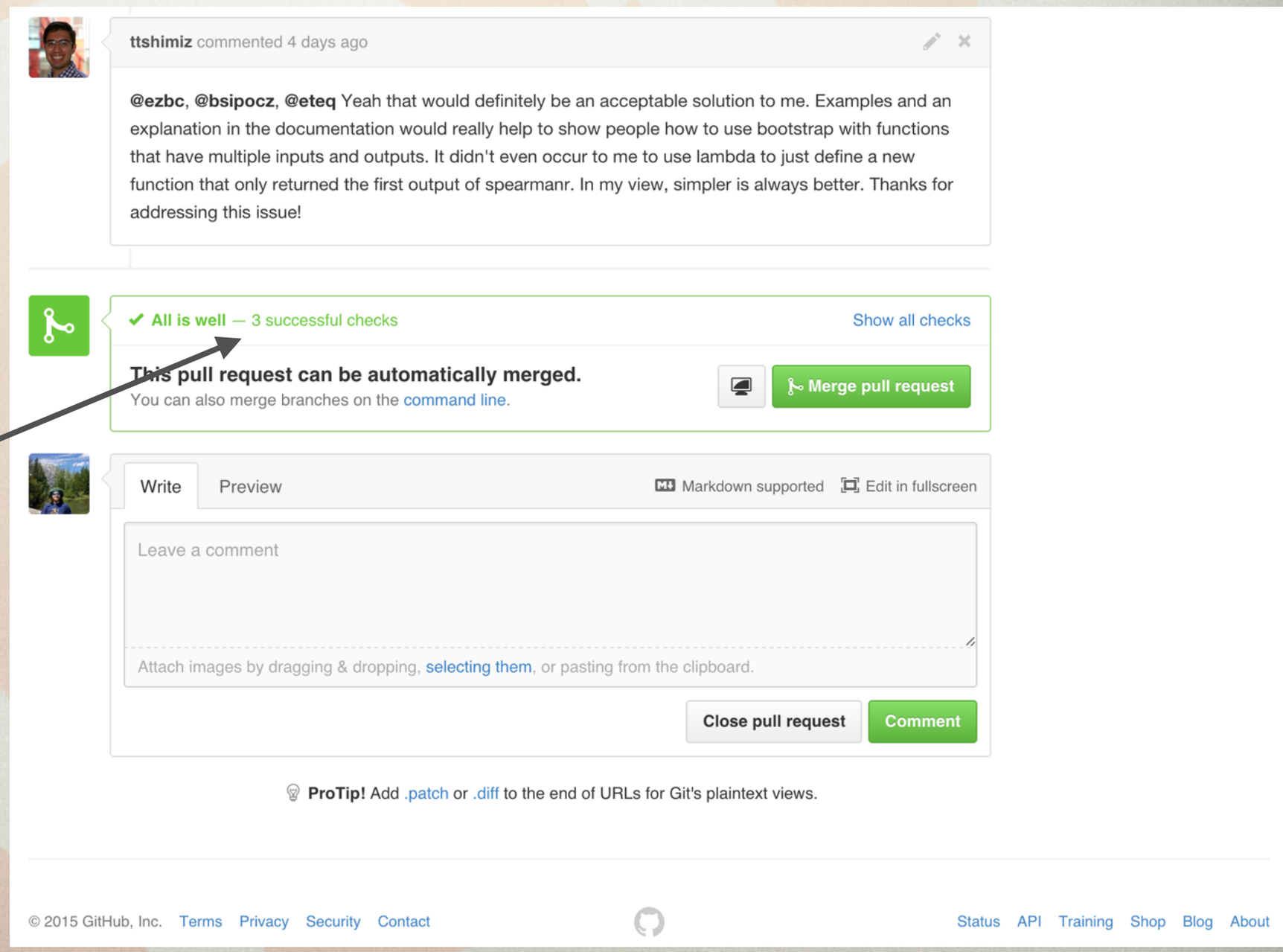


# What are the Key Elements?

py.test

Travis

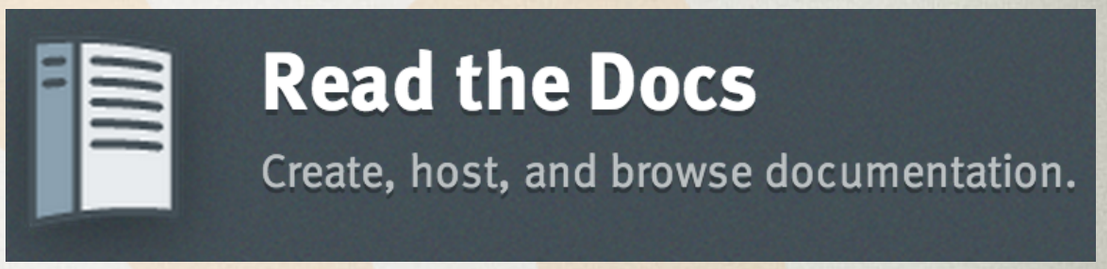
AppVeyor

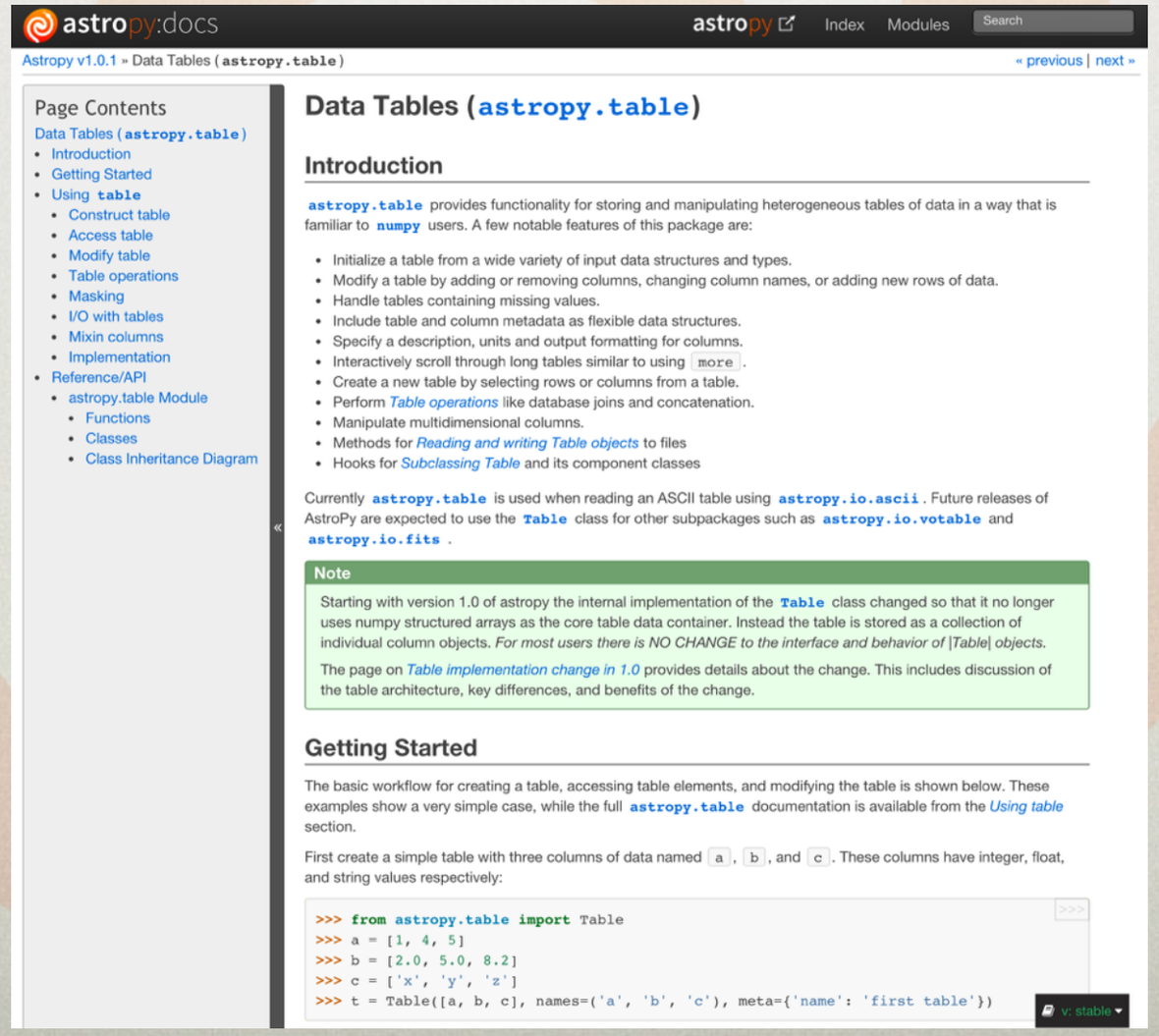


# What are the Key Elements?

Sphinx

Python documentat on generator



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

# What are the Key Elements?

python

Sphinx

Python documentat on 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**

v: stable

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

# What are the Key Elements?

- Github for sharing
- Test *everything* (automatically)
- *Easy* documentation (⇒ thorough)

# What are the Key Elements?

- Github for sharing
- Test *everything* (automatically)
- *Easy* documentation (⇒ thorough)

***Consider using some of this for "ordinary" research code/collaborations!***

# Now let's Do Something!

http://eteq.github.io/astropy-tutorials

astropy Tutorials

http://bit.ly/scicoder-astropy-tutorials

```
git clone https://github.com/eteq/astropy-tutorials.git
cd astropy-tutorials
git checkout scicoder
<OR: git checkout origin/scicoder>
```

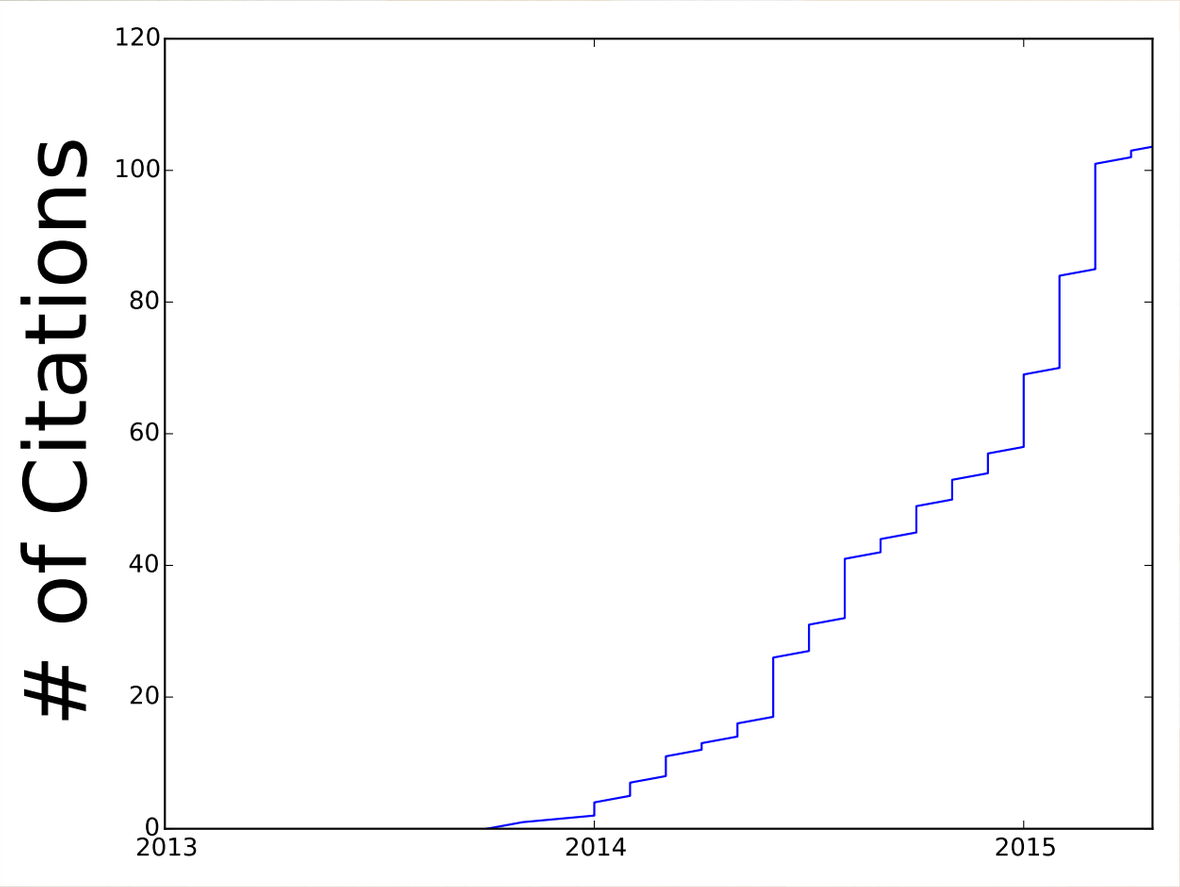
**Suggested Order:**

- Using Astropy Quantities for astrophysical calculations
- Read and plot catalog information from a text file
- Using astropy.coordinates to Match Catalogs and Plan Observations
- Any others you might think are interesting

# Astropy's Development

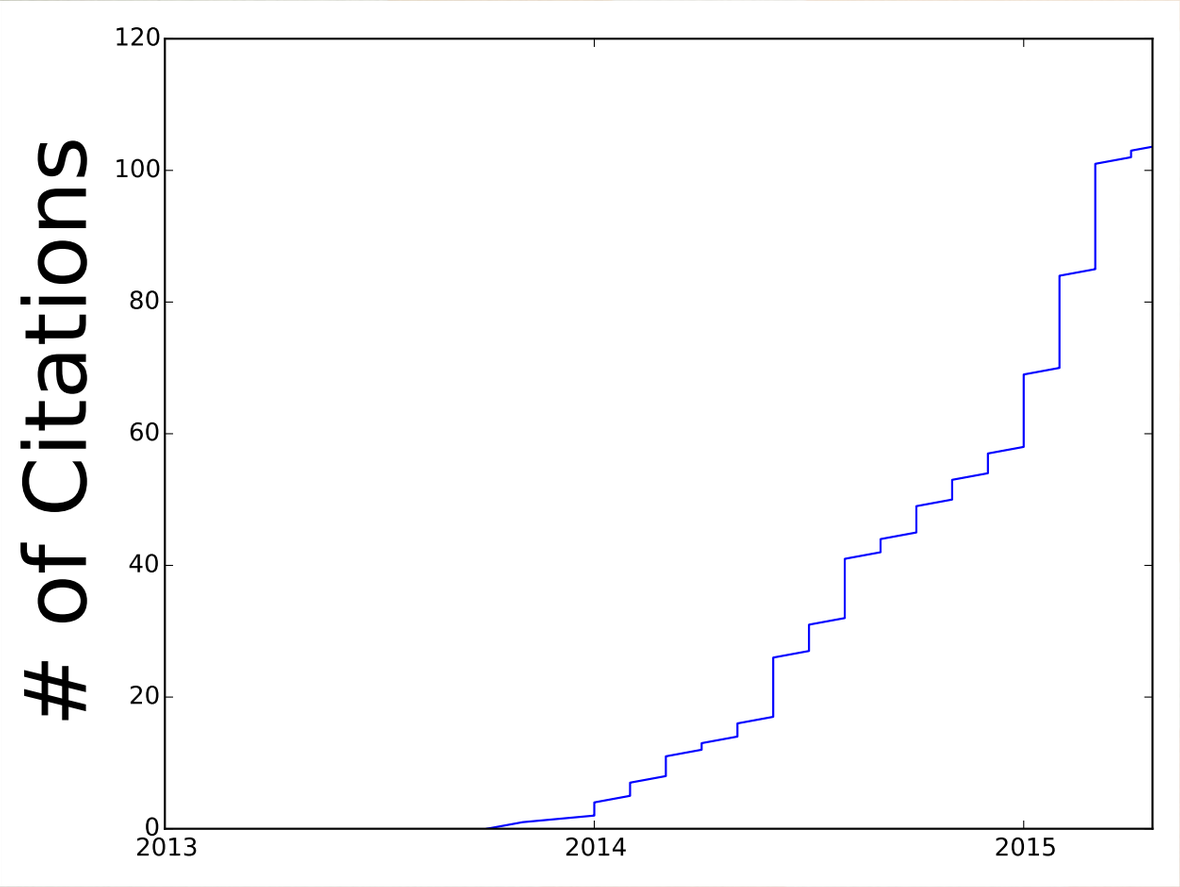
# Impact of Astropy

Astropy Collaboration et al. 2013
A&A 558 A33



# Impact of Astropy

Astropy Collaboration et al. 2013
A&A 558 A33

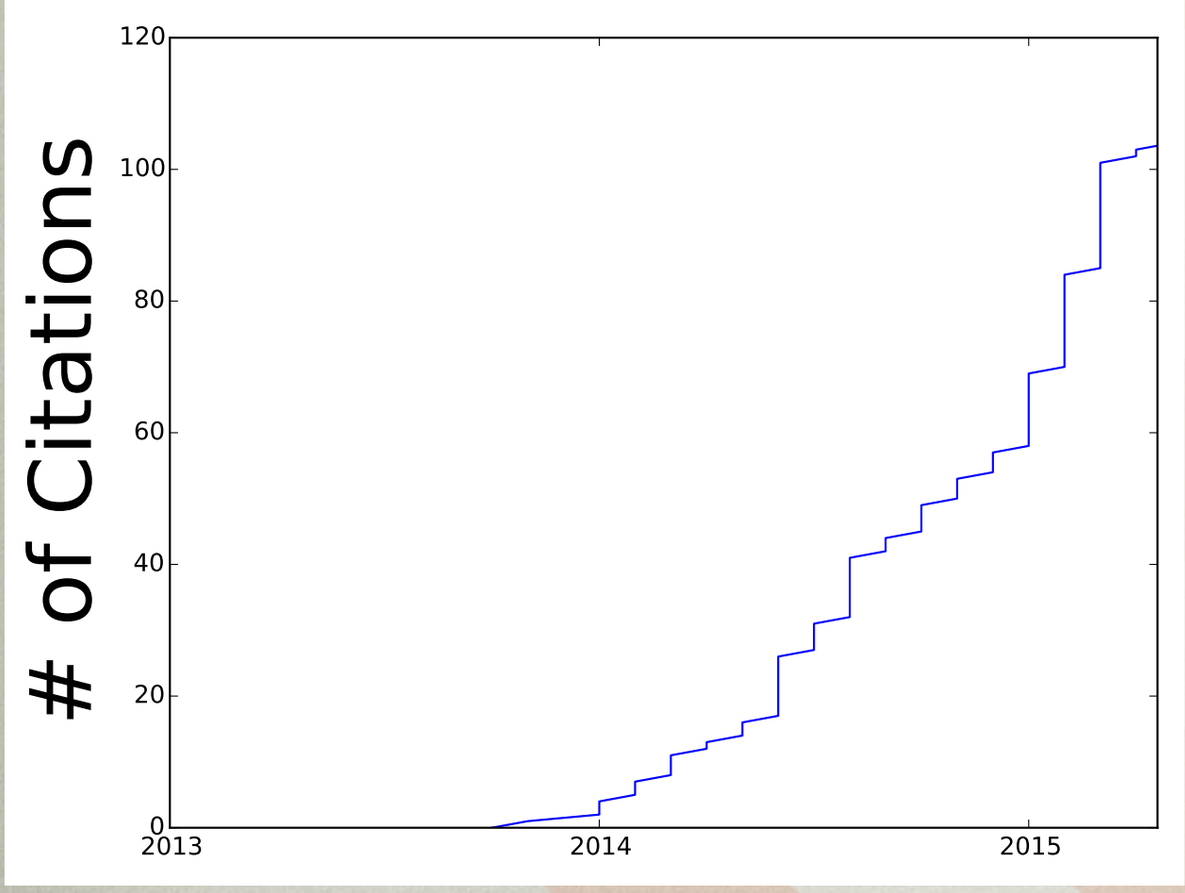


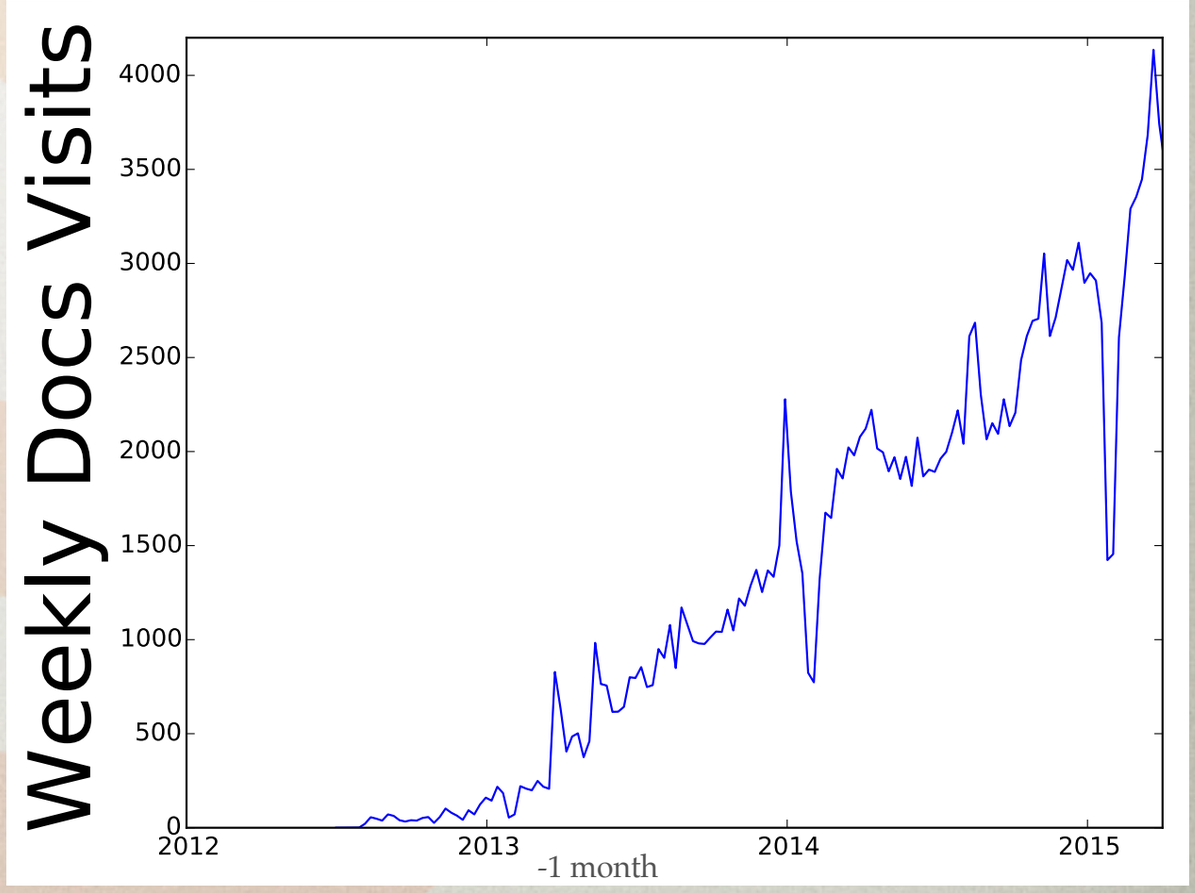

downloads 67.4k/month / 10-100 ?

# Impact of Astropy

Astropy Collaboration et al. 2013
A&A 558 A33

http://docs.astropy.org





/10-100 ?

# Giving Credit Where Credit Is Due

- Shailesh Ahuja
- Tom Aldcroft
- Kyle Barbary
- Geert Barentsen
- Paul Barrett
- Andreas Baumbach
- Chris Beaumont
- Daniel Bell
- Francesco Biscani
- Thompson Le Blanc
- Christopher Bonnett
- Joseph Jon Booker
- Médéric Boquien
- Azalee Bostroem
- Matthew Bourque
- Larry Bradley
- Gustavo Bragança
- Erik M. Bray
- Eli Bressert
- Hugo Buddelmeijer
- Mihai Cara
- Mabry Cervin
- Pritish Chakraborty
- Alex Conley
- Jean Connelly
- Simon Conseil
- Ryan Cooke
- Matthew Craig
- Steven Crawford
- Neil Crighton
- Kelle Cruz
- Daniel Datsev
- Matt Davis
- Christoph Deil
- Nadia Dencheva
- Jörg Dietrich
- Axel Donath
- Michael Droettboom
- Jonathan Eisenhamer
- Zach Edwards
- Thomas Erben
- Henry Ferguson
- Jonathan Foster
- Ryan Fox
- Lehman Garrison
- Simon Gibbons
- Adam Ginsburg
- Christoph Gohlke
- Perry Greenfield
- Dylan Gregersen
- Frédéric Grollier
- Karan Grover
- Kevin Gullikson
- Hans Moritz Günther
- Alex Hagen
- Michael Hoenig
- Emma Hogan
- Chris Hanley
- JC Hsu
- Anthony Horton
- Eric Jeschke
- Sarah Kendrew
- Marten van Kerkwijk
- Wolfgang Kerzendorf
- Lennard Kiehl
- Rashid Khan
- Dominik Klaes
- Kacper Kowalik
- Roban Hultman Kramer
- Arne de Laat
- Antony Lee
- Simon Liedtke
- Pey Lian Lim
- Joseph Long
- Aaron Meisner
- Serge Montagnac
- José Sabater Montes
- Michael Mueller
- Stuart Mumford
- Demitri Muna
- Prasanth Nair
- Bogdan Nicula
- Joe Philip Ninan
- Bryce Nordgren
- Miruna Oprescu
- Luigi Paioro
- Asish Panda
- Madhura Parikh
- Sergio Pascual
- Rohit Patil
- David Perez-Suarez
- Ray Plante
- Adrian Price-Whelan
- Xavier Prochaska
- David Pérez-Suárez
- QuanTakeuchi
- Tanuj Rastogi
- Thomas Robitaille
- Juan Luis Cano Rodríguez
- Evert Rol
- Alex Rudy
- Joseph Ryan
- Eloy Salinas
- Gerrit Schellenberger
- David Shiga
- David Shupe
- Jonathan Sick
- Leo Singer
- Brigitta Sipocz
- Shivan Sornarajah
- Shantanu Srivastava
- Ole Streicher
- Bernardo Sulzbach
- James Taylor
- Jeff Taylor
- Kirill Tchernyshyov
- Víctor Terrón
- Erik Tollerud
- James Turner
- Miguel de Val-Borro
- Jonathan Whitmore
- Lisa Walter
- Benjamin Alan Weaver
- Jonathan Whitmore
- Julien Woillez
- Víctor Zabalza