The Astropy Project

Adrian Price-Whelan

adrn adrianprw

Kelle Cruz

kelle kellecruz

Adrian Price-Whelan

Postdoctoral fellow
Princeton University
(visiting CCA this year, starting postdoc 2019)

adrn adrianprw

I'm mostly an astrophysicist
(part-time software developer)

I use the dynamics of stars throughout the Milky Way to study Dark Matter
(see October issue of Science, "Sky Rivers")

I develop specialized software for
Galactic dynamics

Dominant computational costs:
numerical integration + probabilistic inference
(sampling)

Adrian Price-Whelan

Postdoctoral fellow
Princeton University
(visiting CCA this year, starting postdoc 2019)

Astropy lead developer

adrn adrianprw

2011

How do I represent and transform astronomical coordinates in Python?

the year 2011

the year 2011

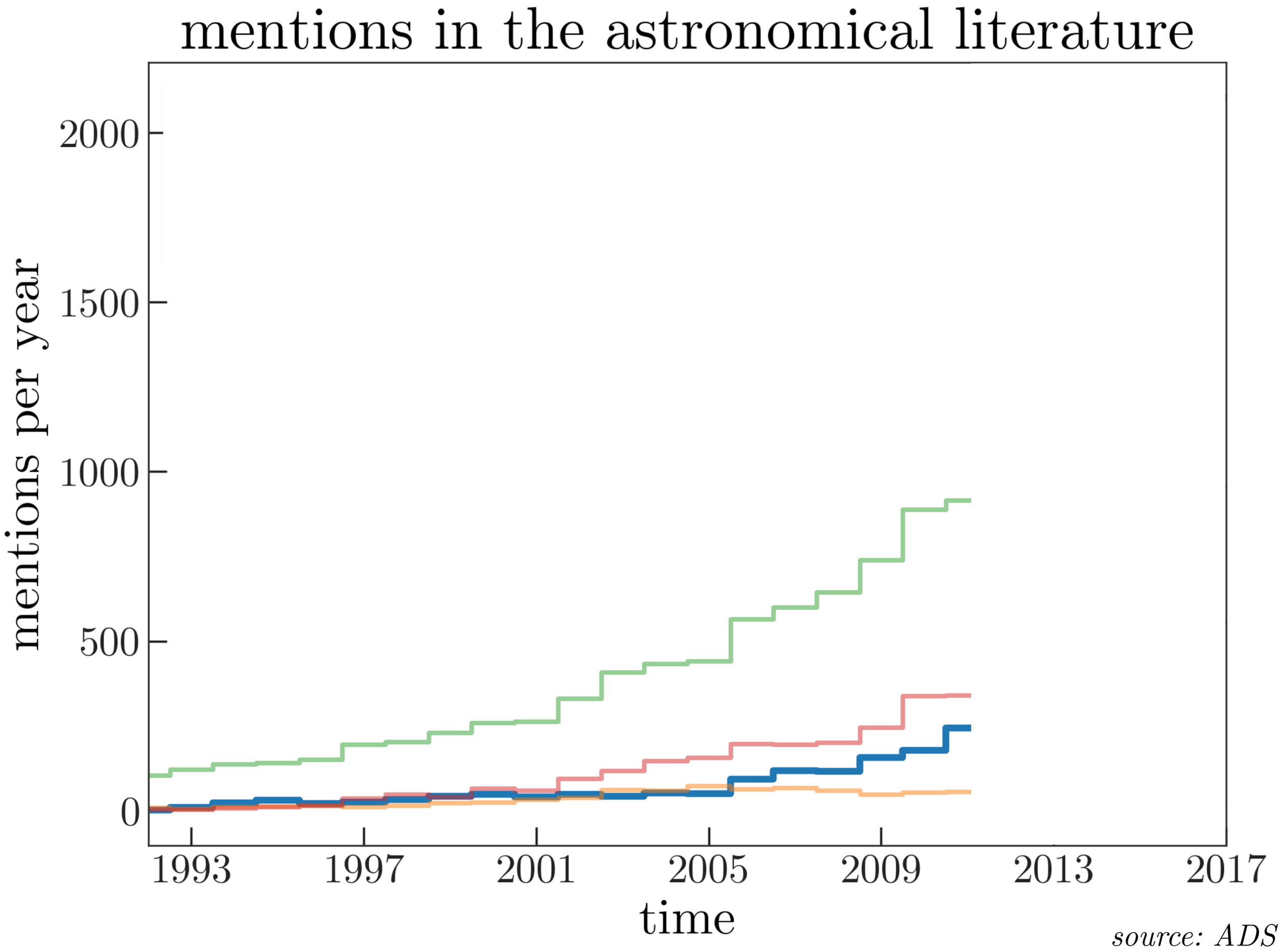
2011

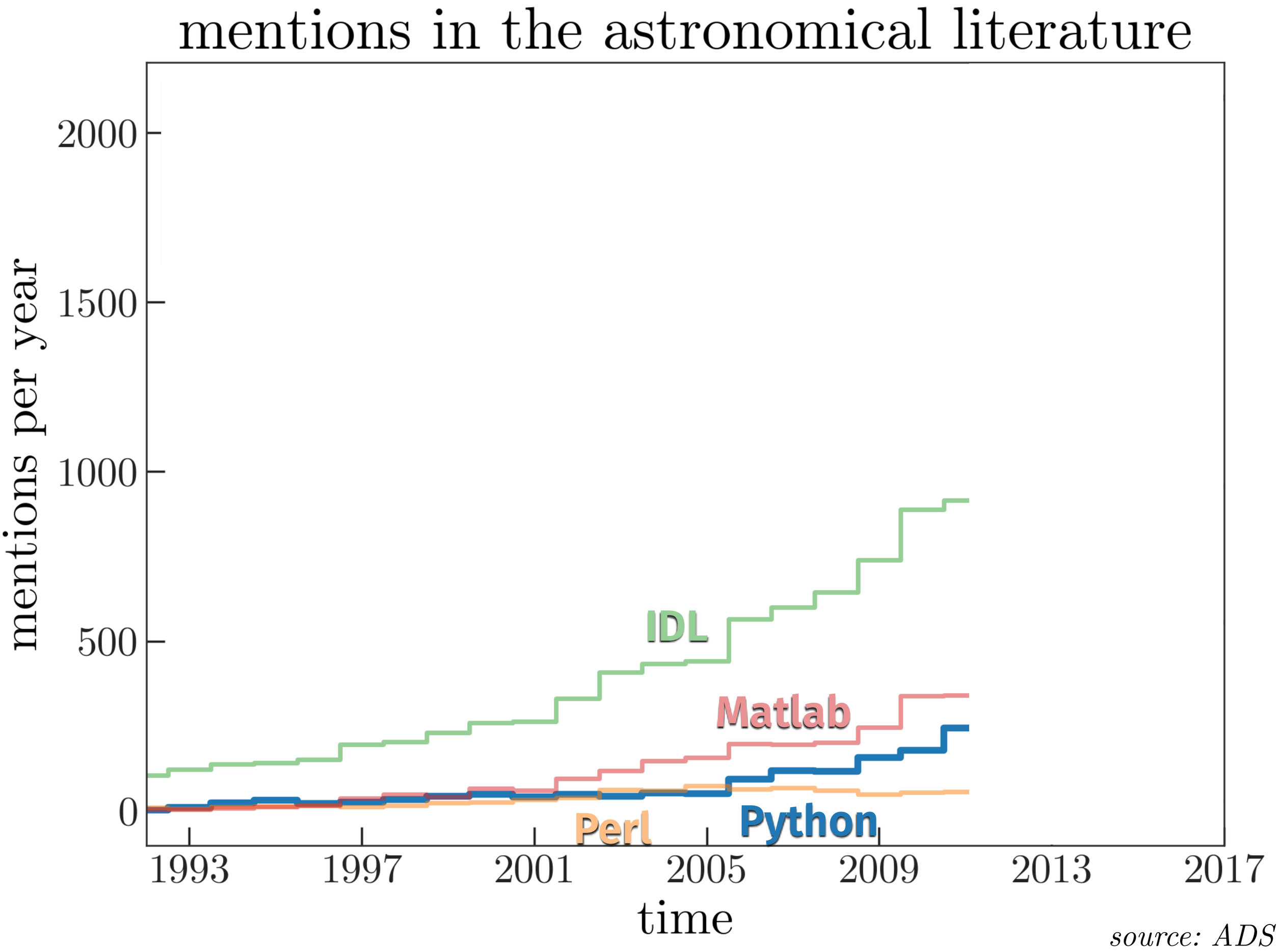
How do I represent and transform astronomical coordinates in Python?

2011

How do I represent and transform astronomical coordinates in Python?

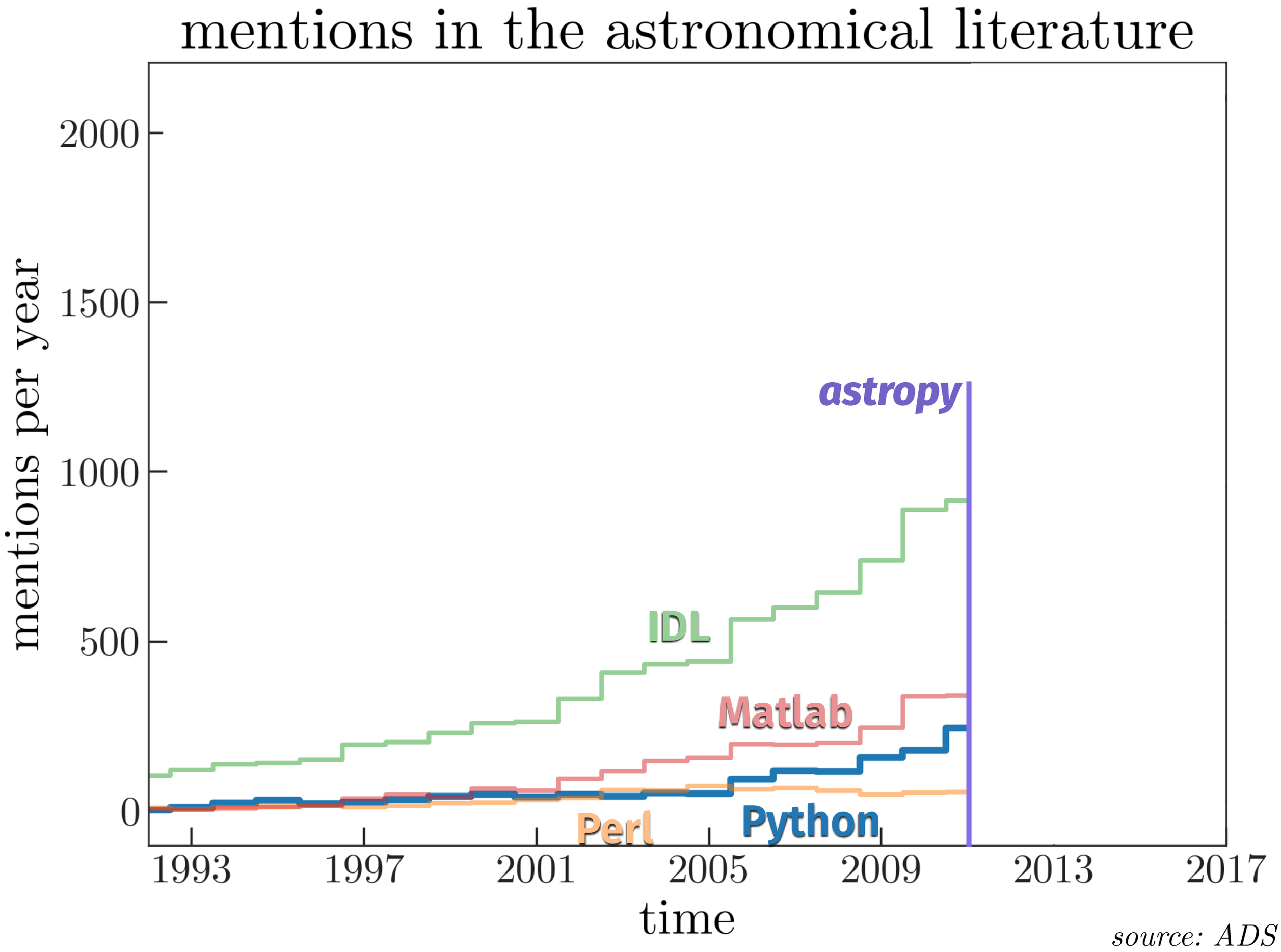
astrolib
astropysics
coordlib
ephempy
kapteyn
pyastro
pyast
+ more

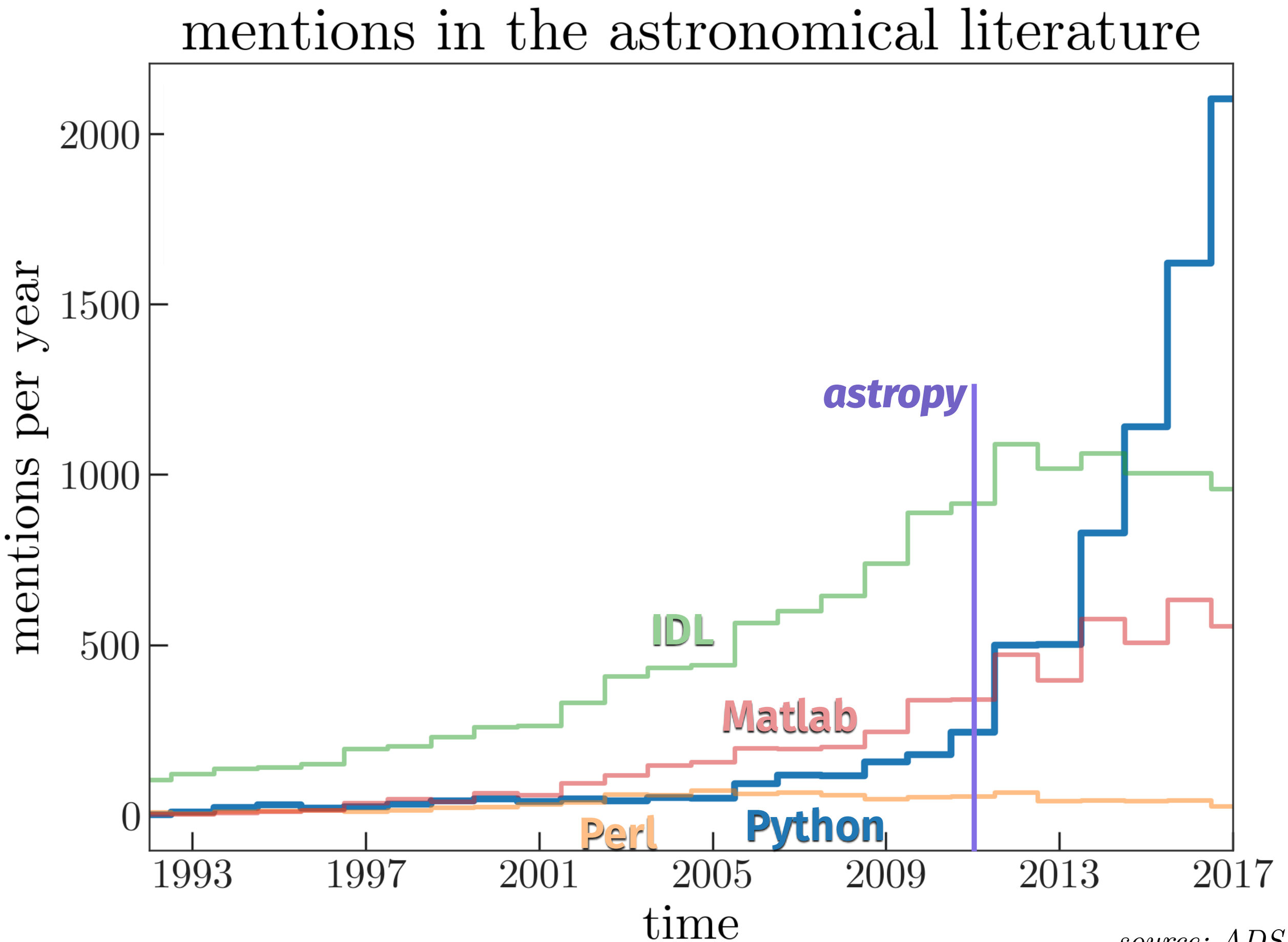

mentions in the astronomical literature
mentions per year
2000
1500
1000
500
0
1993
1997
2001
2005
2009
2013
2017
time
source: ADS


mentions in the astronomical literature
mentions per year
2000
1500
1000
500
0
IDL
Matlab
Perl
Python
1993
1997
2001
2005
2009
2013
2017
time
source: ADS

Idea: create a Python package with generic functionality that most astronomers need


SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.


mentions in the astronomical literature
mentions per year
2000
1500
1000
500
0
1993
1997
2001
2005
2009
2013
2017
time
astropy
IDL
Matlab
Perl
Python
source: ADS


mentions in the astronomical literature
mentions per year
2000
1500
1000
500
0
1993
1997
2001
2005
2009
2013
2017
time
astropy
IDL
Matlab
Perl
Python
source: ADS

The Astropy Library

The Astropy

Library **Ecosystem** **Community**

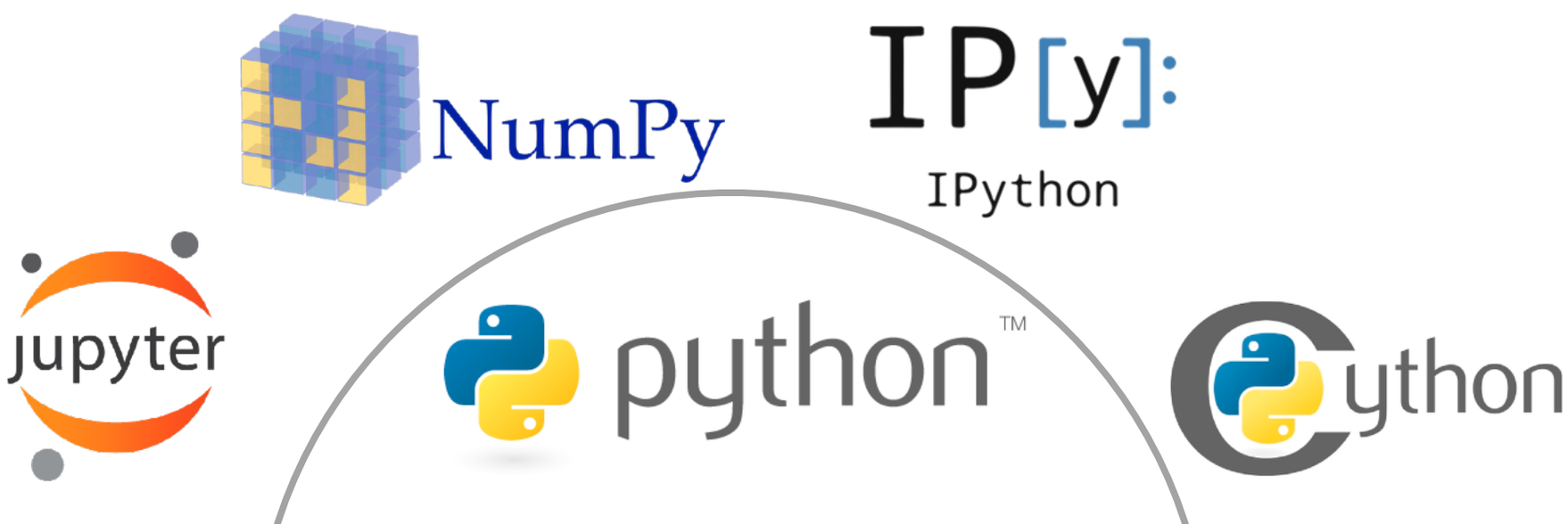
The Astropy library

“The ultimate goal that we seek is a package that would contain much of the core functionality and some common tools required across Astronomy, but not everything Astronomers will ever need.”

extending the scientific Python "stack"

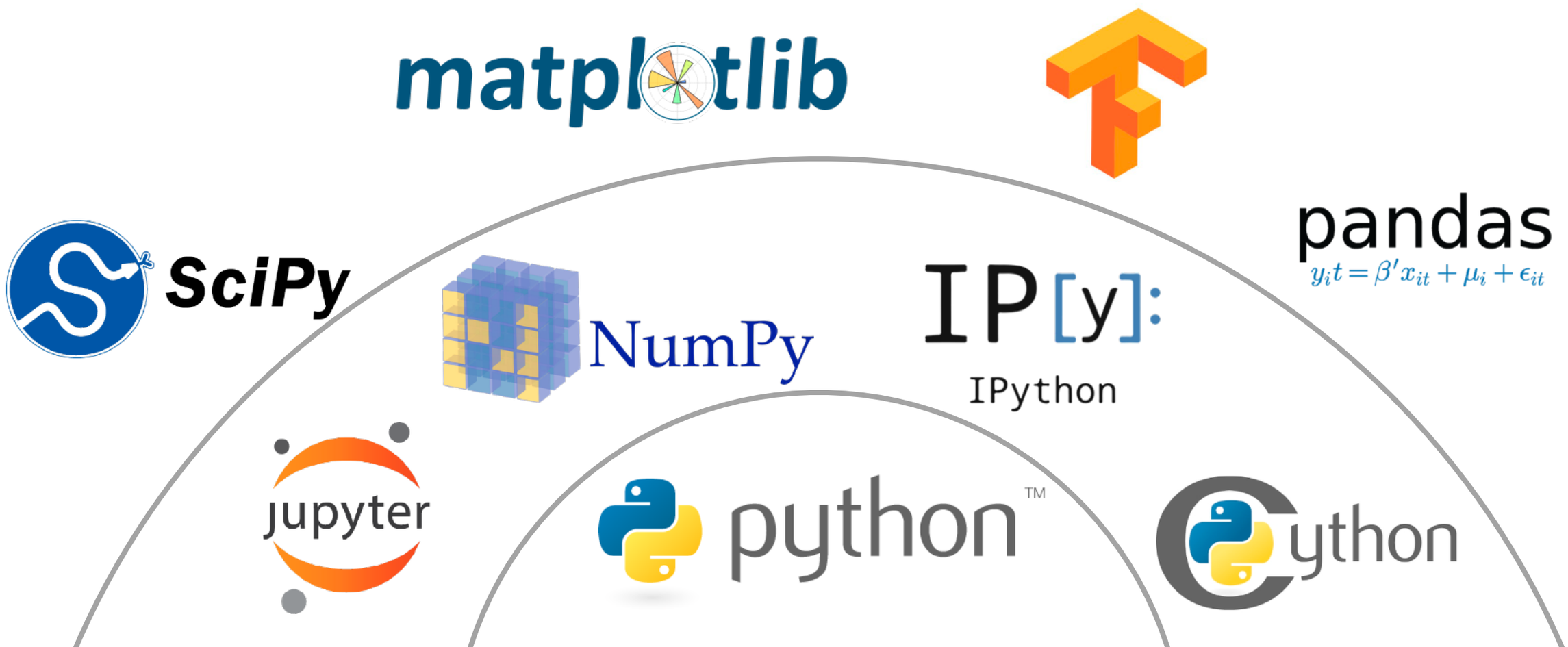
python™

extending the scientific Python "stack"



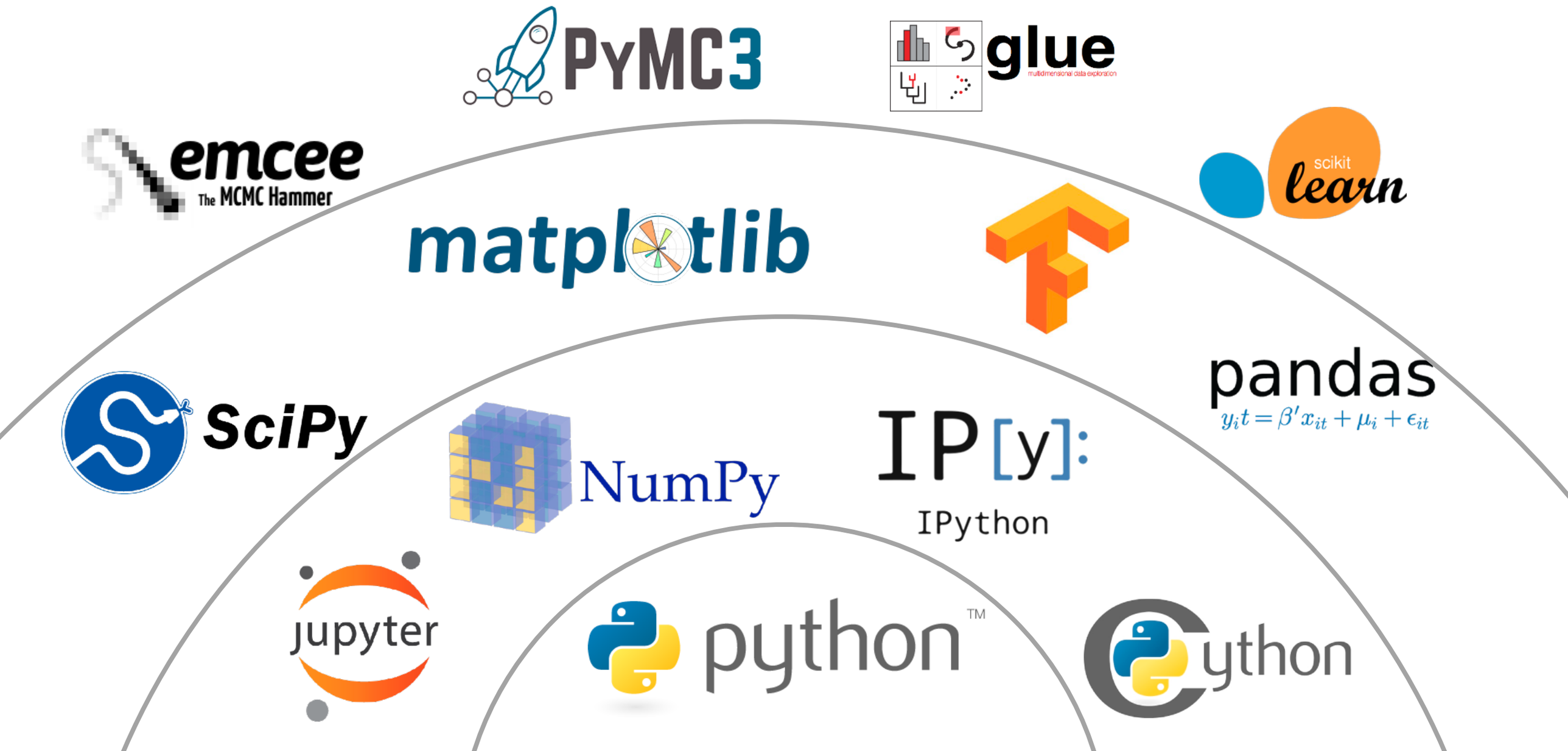
adapted from Jake Vanderplas

extending the scientific Python "stack"



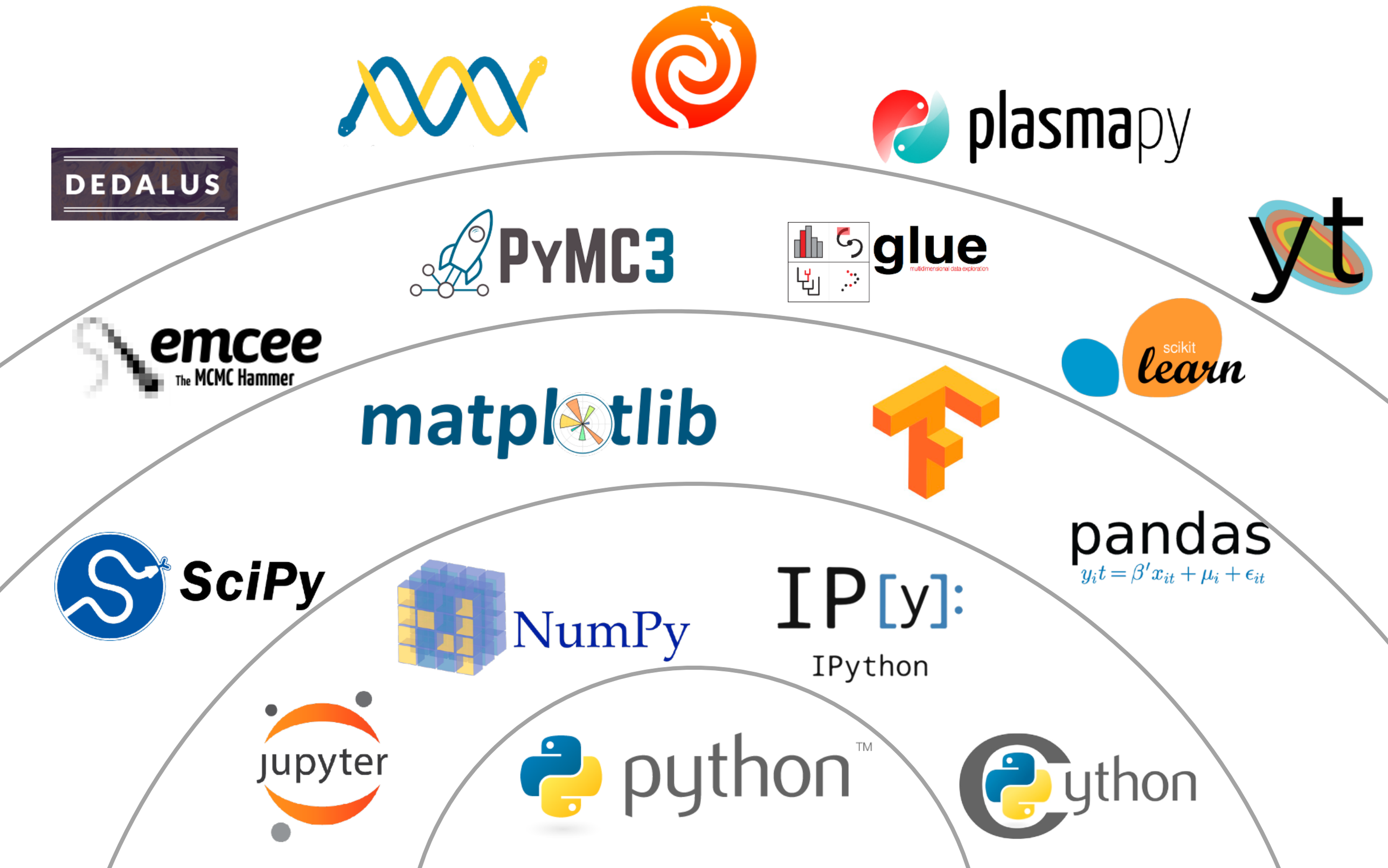
adapted from Jake Vanderplas

extending the scientific Python "stack"

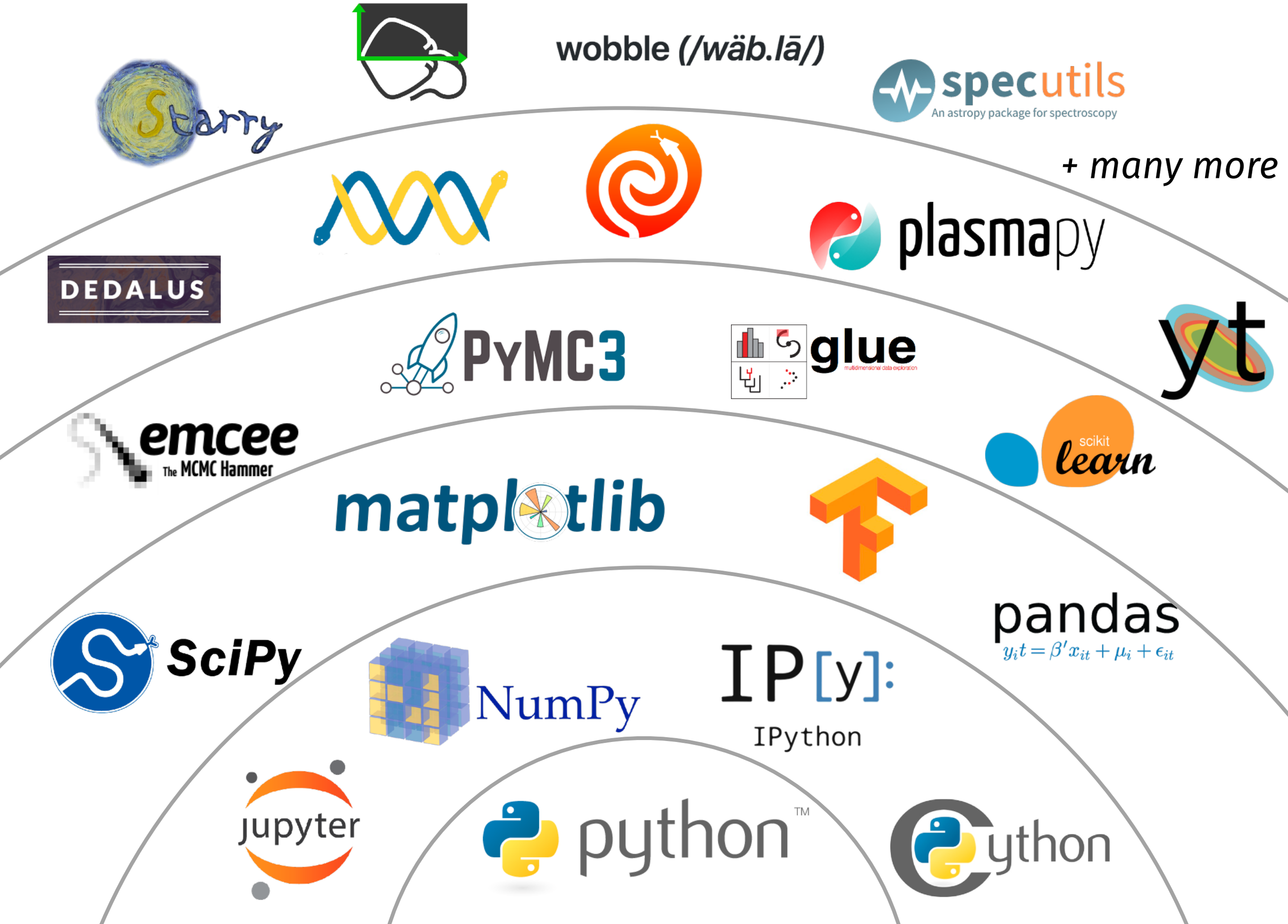


adapted from Jake Vanderplas

extending the scientific Python "stack"



adapted from Jake Vanderplas



Why?

Why need a library for astronomy?

Why need a library for astronomy?

Custom binary file formats (e.g., FITS)

Why need a library for astronomy?

Custom binary file formats (e.g., FITS)

Represent units & quantities in code

In [1]:
```
import astropy.units as u
```

In [1]:
```
import astropy.units as u
```

In [2]:
```
speed = 17. * u.km / u.s # speed of Voyager 1
speed
```

Out[2]: $17\ \frac{\mathrm{km}}{\mathrm{s}}$

In [1]:
```
import astropy.units as u
```

In [2]:
```
speed = 17. * u.km / u.s # speed of Voyager 1
speed
```

Out[2]: $17\ \frac{\mathrm{km}}{\mathrm{s}}$

In [3]:
```
speed.to(u.imperial.mi / u.hour)
```

Out[3]: $38027.917\ \frac{\mathrm{mi}}{\mathrm{h}}$

In [1]:
```
import astropy.units as u
```

In [2]:
```
speed = 17. * u.km / u.s # speed of Voyager 1
speed
```

Out[2]: $17\ \frac{\mathrm{km}}{\mathrm{s}}$

In [3]:
```
speed.to(u.imperial.mi / u.hour)
```

Out[3]: $38027.917\ \frac{\mathrm{mi}}{\mathrm{h}}$

In [4]:
```
distance = 4.37 * u.lightyear # distance to alpha Centauri
distance
```

Out[4]: 4.37 lyr

In [1]:
```
import astropy.units as u
```

In [2]:
```
speed = 17. * u.km / u.s # speed of Voyager 1
speed
```

Out[2]: $17\ \frac{\mathrm{km}}{\mathrm{s}}$

In [3]:
```
speed.to(u.imperial.mi / u.hour)
```

Out[3]: $38027.917\ \frac{\mathrm{mi}}{\mathrm{h}}$

In [4]:
```
distance = 4.37 * u.lightyear # distance to alpha Centauri
distance
```

Out[4]: $4.37\ \mathrm{lyr}$

In [5]:
```
distance / speed
```

Out[5]: $0.25705882\ \frac{\mathrm{lyr\ s}}{\mathrm{km}}$

In [1]:
```
import astropy.units as u
```

In [2]:
```
speed = 17. * u.km / u.s # speed of Voyager 1
speed
```

Out[2]: $17\ \frac{\mathrm{km}}{\mathrm{s}}$

In [3]:
```
speed.to(u.imperial.mi / u.hour)
```

Out[3]: $38027.917\ \frac{\mathrm{mi}}{\mathrm{h}}$

In [4]:
```
distance = 4.37 * u.lightyear # distance to alpha Centauri
distance
```

Out[4]: 4.37 lyr

In [5]:
```
distance / speed
```

Out[5]: $0.25705882\ \frac{\mathrm{lyr\ s}}{\mathrm{km}}$

In [6]:
```
(distance / speed).to(u.yr)
```

Out[6]: 77064.297 yr

Why need a library for astronomy?

Why need a library for astronomy?

Custom binary file formats (e.g., FITS)

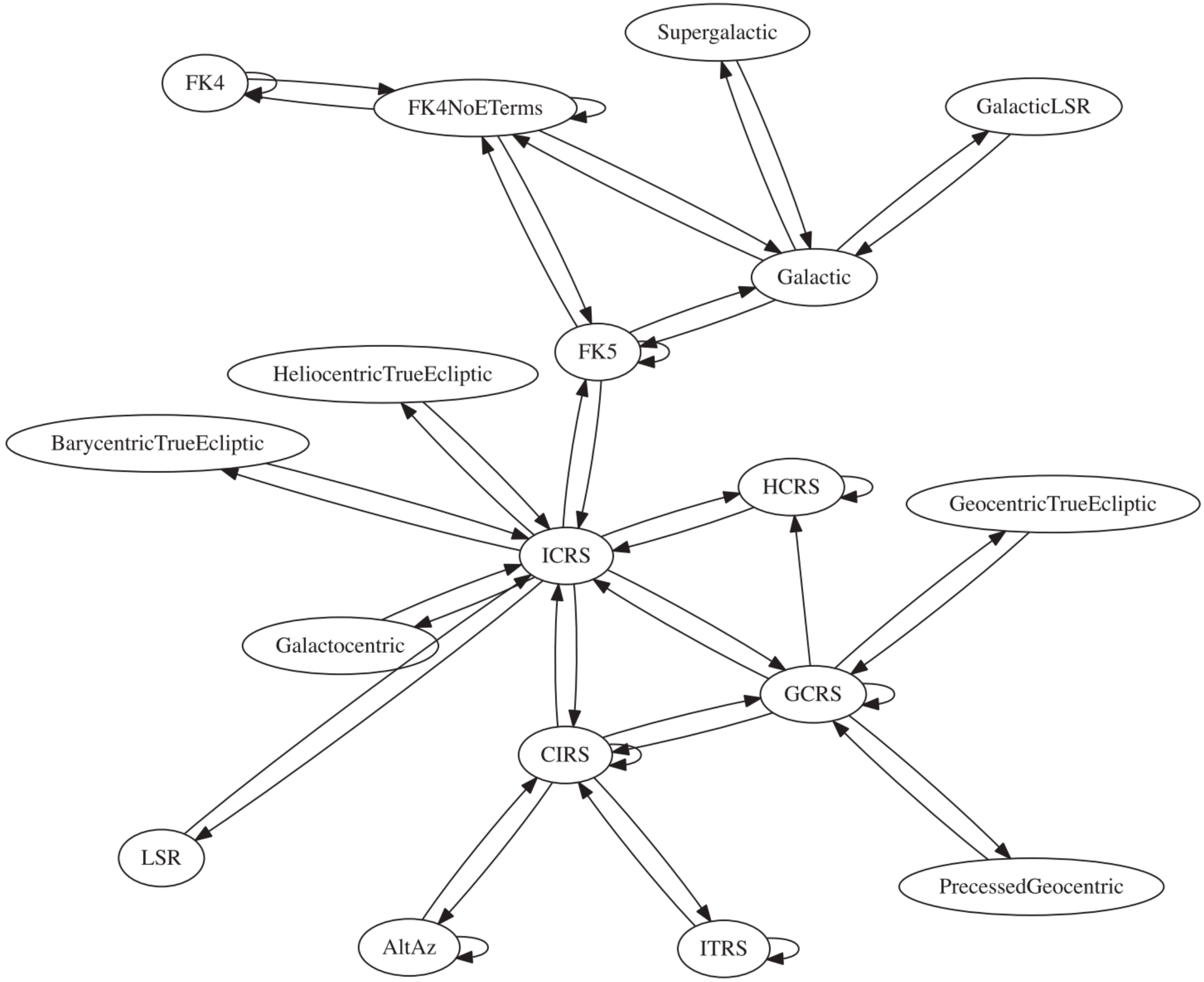
Represent units & quantities in code

Why need a library for astronomy?

Custom binary file formats (e.g., FITS)

Represent units & quantities in code

Coordinate systems & transformations


Supergalactic
FK4
FK4NoETerms
GalacticLSR
Galactic
FK5
HeliocentricTrueEcliptic
BarycentricTrueEcliptic
HCRS
GeocentricTrueEcliptic
ICRS
Galactocentric
GCRS
CIRS
LSR
PrecessedGeocentric
AltAz
ITRS

Why need a library for astronomy?

Why need a library for astronomy?

Custom binary file formats (e.g., FITS)

Represent units & quantities in code

Coordinate systems & transformations

Commonly-used but niche statistics (not in scipy.stats)

Why need a library for astronomy?

Custom binary file formats (e.g., FITS)

Represent units & quantities in code

Coordinate systems & transformations

Commonly-used but niche statistics (not in scipy.stats)

OMG ASCII tables

```
#K MERGERAD   = INDEF                    scaleunit  %-23.7g
#K IRAF = NOAO/IRAFV2.10EXPORT version %-23s
#K USER = davis name %-23s
#K HOST = tucana computer %-23s
#
#N ID    XCENTER   YCENTER   MAG         MERR          MSKY          NITER    \
#U ##    pixels    pixels    magnitudes  magnitudes    counts        ##       \
#F %-9d  %-10.3f   %-10.3f   %-12.3f     %-14.3f       %-15.7g       %-6d
#
#N          SHARPNESS   CHI          PIER  PERROR                              \
#U          ##          ##           ##    perrors                             \
#F          %-23.3f     %-12.3f      %-6d  %-13s
#
14        138.538     INDEF   15.461      0.003         34.85955      4        \
             -0.032      0.802       0      No_error
```

```
Table: Table name here
= ================================================================================
Catalog reference paper
    Bibliography info here
================================================================================
ADC_Keywords: Keyword ; Another keyword ; etc

Description:
    Catalog description here.
================================================================================
Byte-by-byte Description of file: datafile3.txt
--------------------------------------------------------------------------------
   Bytes Format Units  Label  Explanations
--------------------------------------------------------------------------------
   1-  3 I3     ---    Index  Running identification number
   5-  6 I2     h      RAh    Hour of Right Ascension (J2000)
   8-  9 I2     min    RAm    Minute of Right Ascension (J2000)
  11- 15 F5.2   s      RAs    Second of Right Ascension (J2000)
--------------------------------------------------------------------------------
Note (1): A CDS file can contain sections with various metadata.
          Notes can be multiple lines.
Note (2): Another note.
--------------------------------------------------------------------------------
  1 03 28 39.09
  2 04 18 24.11
```

```
\\name=value
\\ Comment
|  column1 |   column2 | column3 | column4  |    column5    |
|  double  |   double  |   int   |  double  |     char      |
|  unit    |   unit    |   unit  |   unit   |     unit      |
|  null    |   null    |   null  |   null   |     null      |
 2.0978     29.09056    73765     2.06000    B8IVpMnHg
```

```
|-----ra---|----dec---|---sao---|------v---|----sptype--------|
  2.09708    29.09056     73765    2.06000     B8IVpMnHg
```

```
\begin{table}
\begin{tabular}{cc}
x & y \\
1 & 1 \\
2 & 4 \\
3 & 9 \\
\end{tabular}
\end{table}
```

Why need a library for astronomy?

Custom binary file formats (e.g., FITS)

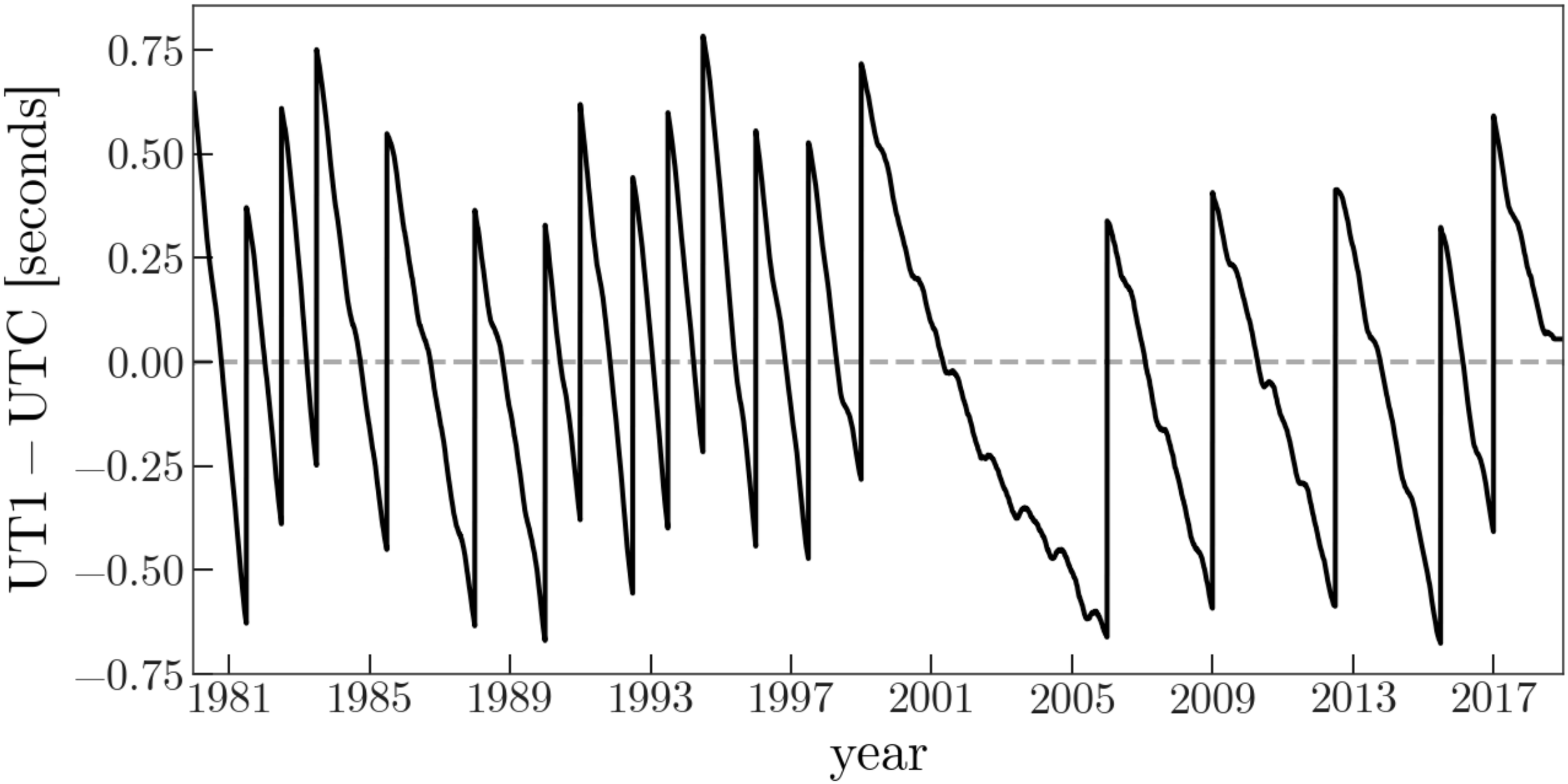
Represent units & quantities in code

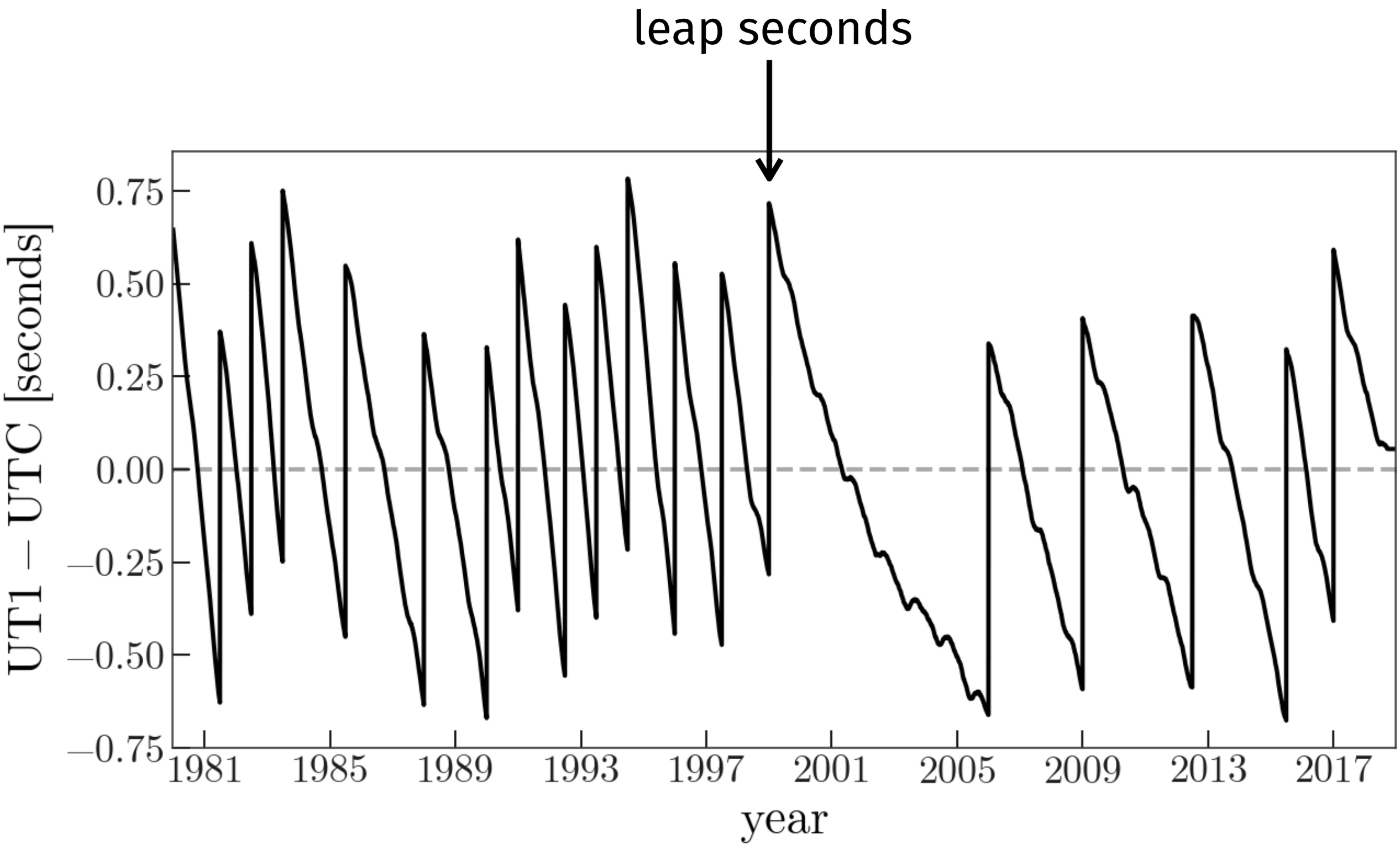
Coordinate systems & transformations

Commonly-used but niche statistics (not in scipy.stats)

OMG ASCII tables

Ultra-precise timing (pulsars!)


0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
UT1 − UTC [seconds]
1981
1985
1989
1993
1997
2001
2005
2009
2013
2017
year


leap seconds
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
UT1 − UTC [seconds]
1981
1985
1989
1993
1997
2001
2005
2009
2013
2017
year

Why need a library for astronomy?

Custom binary file formats (e.g., FITS)

Represent units & quantities in code

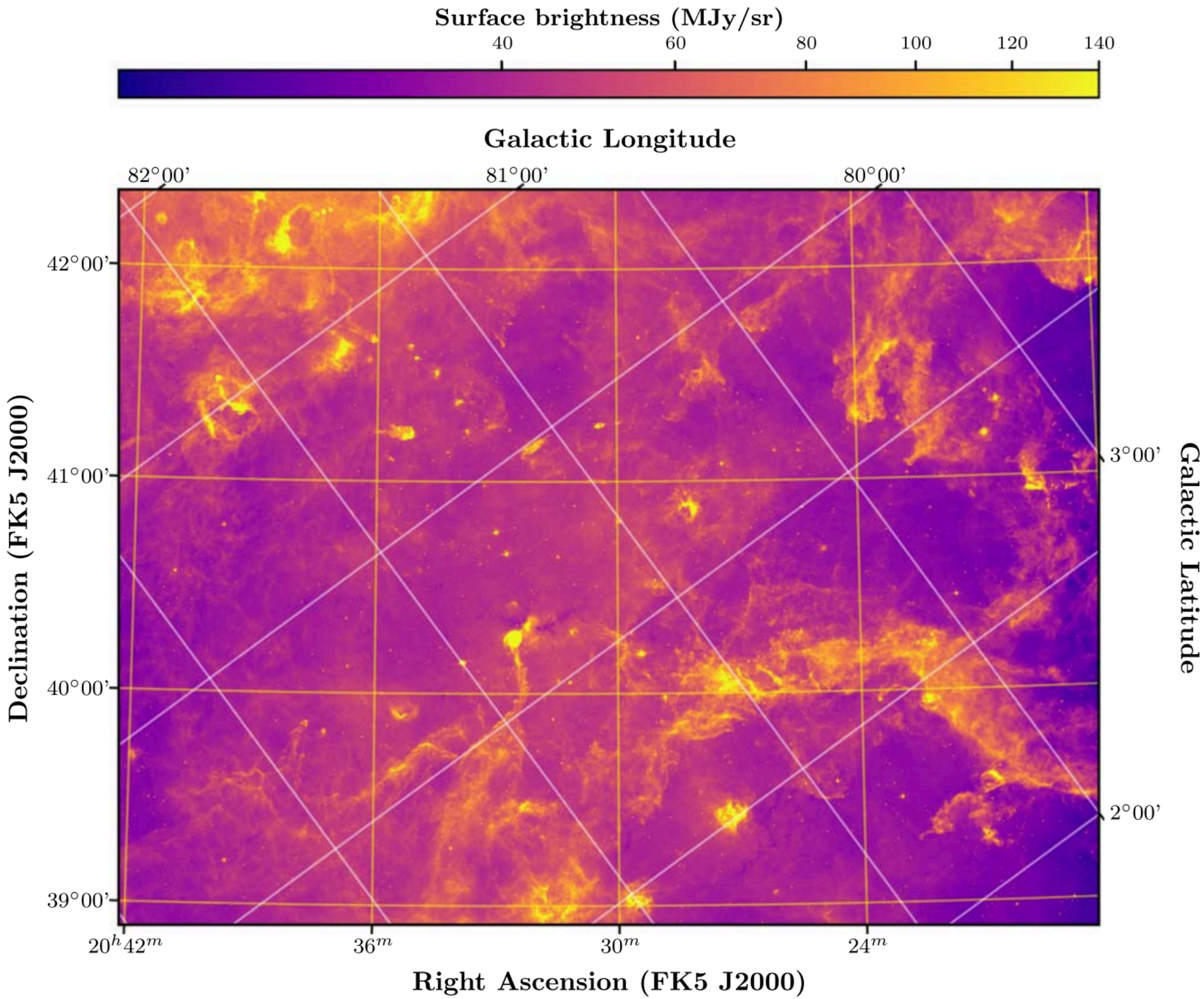
Coordinate systems & transformations

Commonly-used but niche statistics (not in scipy.stats)

OMG ASCII tables

Super-precise timing (pulsars!)

Plotting images of the sky (a sphere! projections...)


Surface brightness (MJy/sr)
40
60
80
100
120
140
Galactic Longitude
82°00'
81°00'
80°00'
42°00'
41°00'
40°00'
39°00'
3°00'
2°00'
Declination (FK5 J2000)
Galactic Latitude
20^h42^m
36^m
30^m
24^m
Right Ascension (FK5 J2000)

What?

Core functionality and common tools for astronomers

Core functionality and ***common tools*** *for astronomers*

Focus on user interface design

Example: astronomical coordinates

$$(\alpha, \delta, D) = (86.7^\circ, 53.09^\circ, 27 \text{ pc})$$

$$(\alpha, \delta, D) = (05\text{h}46\text{m}48\text{s}, +53\text{d}05\text{m}24\text{s}, 27 \text{ pc})$$

$$(x, y, z) = (0.9, 16.2, 21.6) \text{ pc}$$

$$(l, b, D) = (159.14^\circ, 12.46^\circ, 27 \text{ pc})$$

J05464800 + 5305240

Example: astronomical coordinates

$$(\alpha, \delta, D) = (86.7^\circ, 53.09^\circ, 27 \text{ pc})$$
$$(\alpha, \delta, D) = (05\text{h}46\text{m}48\text{s}, +53\text{d}05\text{m}24\text{s}, 27 \text{ pc})$$
$$(x, y, z) = (0.9, 16.2, 21.6) \text{ pc}$$
$$(l, b, D) = (159.14^\circ, 12.46^\circ, 27 \text{ pc})$$
$$\text{J}05464800 + 5305240$$

Reference frame
Coordinate system
Coordinate representation
Component formatting

Example: astronomical coordinates

$$(\alpha, \delta, D) = (86.7^\circ, 53.09^\circ, 27\ \text{pc})$$
$$(\alpha, \delta, D) = (05\text{h}46\text{m}48\text{s}, +53\text{d}05\text{m}24\text{s}, 27\ \text{pc})$$
$$(x, y, z) = (0.9, 16.2, 21.6)\ \text{pc}$$
$$(l, b, D) = (159.14^\circ, 12.46^\circ, 27\ \text{pc})$$
$$\text{J}05464800 + 5305240$$

In [1]:
```
import astropy.units as u
from astropy.coordinates import (SkyCoord, Galactic,
                                 CartesianRepresentation, ICRS)
```

In [2]:
```
SkyCoord(ra=86.7*u.deg, dec=53.09*u.deg, distance=27*u.pc)

SkyCoord(ra='05h46m48s', dec='+53d05m24s', distance=27*u.pc)

SkyCoord(x=0.9, y=16.2, z=21.6, unit=u.pc,
         representation_type=CartesianRepresentation)

SkyCoord(l=159.1*u.deg, b=12.46482*u.deg, distance=27*u.pc,
         frame=Galactic)
```

Example: astronomical coordinates

```
import astropy.units as u
from astropy.coordinates import (SkyCoord, Galactic,
                                 CartesianRepresentation, ICRS)
```

high-level interface

```
SkyCoord(ra=86.7*u.deg, dec=53.09*u.deg, distance=27*u.pc)

SkyCoord(ra='05h46m48s', dec='+53d05m24s', distance=27*u.pc)

SkyCoord(x=0.9, y=16.2, z=21.6, unit=u.pc,
         representation_type=CartesianRepresentation)

SkyCoord(l=159.1*u.deg, b=12.46482*u.deg, distance=27*u.pc,
         frame=Galactic)
```

```
<SkyCoord (Galactic): (l, b, distance) in (deg, deg, pc)
    ( 159.1,  12.46482,  27.)>
```

Example: astronomical coordinates

```
import astropy.units as u
from astropy.coordinates import (SkyCoord, Galactic,
                                 CartesianRepresentation, ICRS)
```

high-level interface

```
SkyCoord(ra=86.7*u.deg, dec=53.09*u.deg, distance=27*u.pc)

SkyCoord(ra='05h46m48s', dec='+53d05m24s', distance=27*u.pc)

SkyCoord(x=0.9, y=16.2, z=21.6, unit=u.pc,
         representation_type=CartesianRepresentation)

SkyCoord(l=159.1*u.deg, b=12.46482*u.deg, distance=27*u.pc,
         frame=Galactic)
```

```
<SkyCoord (Galactic): (l, b, distance) in (deg, deg, pc)
    ( 159.1,  12.46482,  27.)>
```

reference frame

```
Galactic(l=159.1*u.deg, b=12.46482*u.deg,
         distance=27*u.pc)
```

```
<Galactic Coordinate: (l, b, distance) in (deg, deg, pc)
    ( 159.1,  12.46482,  27.)>
```

Example: astronomical coordinates

```
import astropy.units as u
from astropy.coordinates import (SkyCoord, Galactic,
                                 CartesianRepresentation, ICRS)
```

high-level interface

```
SkyCoord(ra=86.7*u.deg, dec=53.09*u.deg, distance=27*u.pc)

SkyCoord(ra='05h46m48s', dec='+53d05m24s', distance=27*u.pc)

SkyCoord(x=0.9, y=16.2, z=21.6, unit=u.pc,
         representation_type=CartesianRepresentation)

SkyCoord(l=159.1*u.deg, b=12.46482*u.deg, distance=27*u.pc,
         frame=Galactic)
```

```
<SkyCoord (Galactic): (l, b, distance) in (deg, deg, pc)
    ( 159.1,  12.46482,  27.)>
```

reference frame

```
Galactic(l=159.1*u.deg, b=12.46482*u.deg,
         distance=27*u.pc)
```

```
<Galactic Coordinate: (l, b, distance) in (deg, deg, pc)
    ( 159.1,  12.46482,  27.)>
```

coordinate representation

```
data = CartesianRepresentation(x=0.9, y=16.2, z=21.6,
                               unit=u.pc)
frame = ICRS(data)
coords = SkyCoord(frame)
```

astropy – v3.0.5

Python + C/Cython extensions
Python >= 3.5

165,595 lines of Python
+ 87,957 lines of tests
+ 48,734 lines of documentation

>1,000 users
~100 downloads/day

astropy

Data structures and transformations

- Constants (`astropy.constants`)
- Units and Quantities (`astropy.units`)
- N-dimensional datasets (`astropy.nddata`)
- Data Tables (`astropy.table`)
- Time and Dates (`astropy.time`)
- Astronomical Coordinate Systems (`astropy.coordinates`)
- World Coordinate System (`astropy.wcs`)
- Models and Fitting (`astropy.modeling`)

Files, I/O, and Communication

- Unified file read/write interface
- FITS File handling (`astropy.io.fits`)
- ASCII Tables (`astropy.io.ascii`)
- VOTable XML handling (`astropy.io.votable`)
- Miscellaneous: HDF5, YAML, ASDF, pickle (`astropy.io.misc`)
- SAMP (Simple Application Messaging Protocol (`astropy.samp`)

Computations and utilities

- Cosmological Calculations (`astropy.cosmology`)
- Convolution and filtering (`astropy.convolution`)
- Data Visualization (`astropy.visualization`)
- Astrostatistics Tools (`astropy.stats`)

docs.astropy.org

How?

How?

Community driven development

Making community-driven development work

Making community-driven development work

1. Use GitHub to host code, track issues, contributions
   - Code review
   - Feature requests
   - Feature planning
   - Manage releases

Making community-driven development work

1. Use GitHub to host code, track issues, contributions
   - Code review
   - Feature requests
   - Feature planning
   - Manage releases

2. Use continuous integration to run tests
   - Travis CI, CircleCI, AppVeyor
   - Enforce good test coverage in new code
   - Test on multiple architectures, dependency versions

Making community-driven development work

1. Use GitHub to host code, track issues, contributions
   - Code review
   - Feature requests
   - Feature planning
   - Manage releases

2. Use continuous integration to run tests
   - Travis CI, CircleCI, AppVeyor
   - Enforce good test coverage in new code
   - Test on multiple architectures, dependency versions

3. Use readthedocs to serve documentation
   - Documentation generated from code with Sphinx
   - New contributions require documentation

Guiding principles

Open source, open development

Open source

astropy / astropy

Unwatch 178 | Star 1,896 | Fork 970

Code | Issues 873 | Pull requests 61 | Projects 1 | Wiki | Insights

Branch: master | astropy / LICENSE.rst | Find file | Copy path

astropy/astropy is licensed under the

BSD 3-Clause "New" or "Revised" License

A permissive license similar to the BSD 2-Clause License, but with a 3rd clause that prohibits others from using the name of the project or its contributors to promote derived products without written consent.

Permissions	Limitations	Conditions
✓ Commercial use	✗ Liability	License and copyright notice
✓ Modification	✗ Warranty	
✓ Distribution		
✓ Private use		

This is not legal advice. Learn more about repository licenses.

github.com/astropy/astropy

Open development

ENH: Adding Transit Periodogram to astropy.stats #7391

Edit

Open **dfm** wants to merge 55 commits into `astropy:master` from `dfm:transit-periodogram`

Conversation 21 | Commits 55 | Files changed 12 | +1,844 −0

dfm commented 12 days ago — First-time contributor

Hi all,

This pull request adds a new feature that I've been working on for a while now with contributions from **@mirca** and **@joeldhartman**. This feature is another popular astronomy-specific periodogram like Lomb-Scargle, but this one is used to find transiting exoplanets and eclipsing binary systems by using a top-hat basis instead of sines. For historical reasons, this algorithm is often referred to as "box least squares", but I think that the more descriptive "transit periodogram" name is better.

This algorithm was proposed by Kovács et al. (2002) and the method has since become the standard method of detecting transiting planets. In many cases, this is achieved by wrapping the Fortran code released by those authors (many people use f2py bindings that I wrote almost 5 years ago). With K2 data continuing to roll in and the launch of TESS (hopefully today! 🤞) it seems like it would be timely to have an implementation of this algorithm within AstroPy.

With this in mind, my collaborators and I have written an efficient implementation of this algorithm in C with Cython bindings that expose an interface that will be familiar for anyone who uses the LombScargle class in astropy.stats.

Reviewers

crawfordsm

larrybradley

At least 1 approving review is required to merge this pull request.

Assignees

No one—assign yourself

Labels

stats

whatsnew-needed

Projects

None yet

Open development

pllim commented 12 days ago Member

@dfm did you do a `git submodule update`?

pllim commented 12 days ago Member

Also saw this error in 32-bit build:

```
astropy/stats/transit_periodogram/transit_periodogram.c: In function ‘run_transit_periodogr
astropy/stats/transit_periodogram/transit_periodogram.c:169:30: error: ‘INFINITY’ undeclare
         best_objective[p] = -INFINITY;
                             ^
astropy/stats/transit_periodogram/transit_periodogram.c:169:30: note: each undeclared ident
error: command 'gcc' failed with exit status 1
```

dfm commented 12 days ago

I hadn't done that, but I did previously rebase. When I do a `git submodule update` now, it doesn't seem to do anything... submodules have always confused me :-\ sorry!

Open development

Timeseries object for Astropy (APE9) #12

Open **Cadair** wants to merge 4 commits into `astropy:master` from `Cadair:master`

Conversation 19 | Commits 4 | Files changed 1

Cadair commented on Mar 24, 2016 — Member

This is still in very early draft.

abigailStev commented on Mar 24, 2016

Pinging myself so I can follow development and discussion.

abigailStev referenced this pull request in **StingraySoftware/stingray** on Mar 24, 2016

Use `pandas` or `astropy.tables` in Lightcurve? #91 — Open

ehsteve commented on Mar 30, 2016

I am very interested in this. Thanks for putting this together **@Cadair**.

Guiding principles

Open source, open development

Provide tested, documented code to users

Software testing & continuous integration

astropy / astropy build passing

Current Branches Build History Pull Requests > Build #19488 More options

master Merge pull request #7411 from nden/format-output-in-evaluate

Tabular outputs shape

Commit 3198acd

Compare d3bb55d..3198acd

Branch master

P. L. Lim authored GitHub committed

#19488 passed

Ran for 28 min 44 sec

Total time 1 hr 26 min 28 sec

2 days ago

Restart build

Software testing & continuous integration

Initial tests — 13 min 17 sec

✓	# 19488.1	Compiler: gcc C	PYTHON_VERSION=3.5 SETUP_CMD='egg_info'	1 min 51 sec
✓	# 19488.2	Compiler: gcc C	PYTHON_VERSION=3.6 SETUP_CMD='egg_info'	1 min 50 sec
✓	# 19488.3	Compiler: clang C	CONDA_DEPENDENCIES=$CONDA_ALL_DEPE…	13 min 17 sec
✓	# 19488.4	Compiler: gcc C	MAIN_CMD="flake8 astropy --count $FLAKE8…	2 min 42 sec

Comprehensive tests — 15 min 24 sec

✓	# 19488.5	Compiler: gcc C	SETUP_CMD='build_docs -w' CONDA_DEPEN…	12 min 20 sec
✓	# 19488.6	Compiler: gcc C	PYTHON_VERSION=3.5 NUMPY_VERSION=1.1…	8 min 7 sec
✓	# 19488.7	Compiler: gcc C	PYTHON_VERSION=3.5 SETUP_CMD='test --r…	10 min 48 sec
✓	# 19488.8	Compiler: gcc C	SETUP_CMD='test --coverage --remote-data=…	15 min 24 sec
✓	# 19488.9	Compiler: gcc C	NUMPY_VERSION=prerelease EVENT_TYPE='…	2 min 15 sec
✓	# 19488.10	Compiler: gcc C	NUMPY_VERSION=1.12 EVENT_TYPE='push p…	6 min 56 sec
✓	# 19488.11	Compiler: gcc C	NUMPY_VERSION=dev SETUP_CMD='test --re…	10 min 58 sec

Documentation

astropy:docs astropy Index Modules Search

Astropy v3.0.4 » next »

Page Contents

Astropy Documentation
- Getting Started
- User Documentation
  - Data structures and transformations
  - Files, I/O, and Communication
  - Computations and utilities
  - Nuts and bolts
- Project details
- Index

astropy

A Community Python Library for Astronomy

The `astropy` package contains key functionality and common tools needed for performing astronomy and astrophysics with Python. It is at the core of the Astropy Project, which aims to enable the community to develop a robust ecosystem of Affiliated Packages covering a broad range of needs for astronomical research, data processing, and data analysis.

Getting Started

- Installation
- What's New in Astropy 3.0?
- Importing astropy and subpackages
- Getting started with subpackages
- Example Gallery
- Tutorials
- Get Help
- Contribute and Report Problems
- About the Astropy Project

User Documentation

Data structures and transformations

- Constants (**`astropy.constants`**)
- Units and Quantities (**`astropy.units`**)
- N-dimensional datasets (**`astropy.nddata`**)

v: stable

Documentation

astropy:docs astropy Index Modules Search

Astropy v3.0.4 » next »

Page Contents

Astropy Documentation
- Getting Started
- User Documentation
  - Data structures and transformations
  - Files, I/O, and Communication
  - Computations and utilities
  - Nuts and bolts
- Project details
- Index

astropy

A Community Python Library for Astronomy

The `astropy` package contains key functionality and common tools needed for performing astronomy and astrophysics with Python. It is at the core of the Astropy Project, which aims to enable the community to develop a robust ecosystem of Affiliated Packages covering a broad range of needs for astronomical research, data processing, and data analysis.

Getting Started

- Installation
- What's New in Astropy 3.0?
- Importing astropy and subpackages
- Getting started with subpackages
- Example Gallery
- Tutorials
- Get Help
- Contribute and Report Problems
- About the Astropy Project

User Documentation

Data structures and transformations

v: stable

- Constants (**`astropy.constants`**)
- Units and Quantities (**`astropy.units`**)
- N-dimensional datasets (**`astropy.nddata`**)

Documentation

astropy:docs astropy Index Modules Search

Astropy v3.0.4 » next »

Page Contents

Astropy Documentation
- Getting Started
- User Documentation
  - Data structures and transformations
  - Files, I/O, and Communication
  - Computations and utilities
  - Nuts and bolts
- Project details
- Index

astropy

A Community Python Library for Astronomy

The `astropy` package contains key functionality and common tools needed for physics with Python. It is at the core of the Astropy Project, which aims to enable th ecosystem of Affiliated Packages covering a broad range of needs for astronomical analysis.

Getting Started

- Installation
- What's New in Astropy 3.0?
- Importing astropy and subpackages
- Getting started with subpackages
- Example Gallery
- Tutorials
- Get Help
- Contribute and Report Problems
- About the Astropy Project

User Documentation

Data structures and transformations

- Constants (**`astropy.constants`**)
- Units and Quantities (**`astropy.units`**)
- N-dimensional datasets (**`astropy.nddata`**)

v: stable

Versions

latest stable v3.0.5 v3.0.4 v2.0.8

v1.3.3 v1.2.2 v1.1.2 v1.0.13 v0.4.6

v0.3.2 v0.2.5 v3.0.x v2.0.x v0.1 lts_a

Downloads

HTML

On Read the Docs

Project Home Builds Downloads

On GitHub

View Edit

Search

Search docs

Hosted by Read the Docs · Privacy Policy

Guiding principles

Guiding principles

Open source, open development

Provide tested, documented code to users

Guiding principles

Open source, open development

Provide tested, documented code to users

Encourage contributions from users

Guiding principles

Open source, open development

Provide tested, documented code to users

Encourage contributions from users

Don’t re-invent tools, but minimize dependencies

Guiding principles

Open source, open development

Provide tested, documented code to users

Encourage contributions from users

Don’t re-invent tools, but minimize dependencies

Feed features and functionality back upstream

Guiding principles

Open source, open development

Provide tested, documented code to users

Encourage contributions from users

Don’t re-invent tools, but minimize dependencies

Feed features and functionality back upstream

Allow & encourage extending core functionality

Who?

Who develops the Astropy library?

Astronomers!

~270 contributors

~20 package leads & maintainers

but...

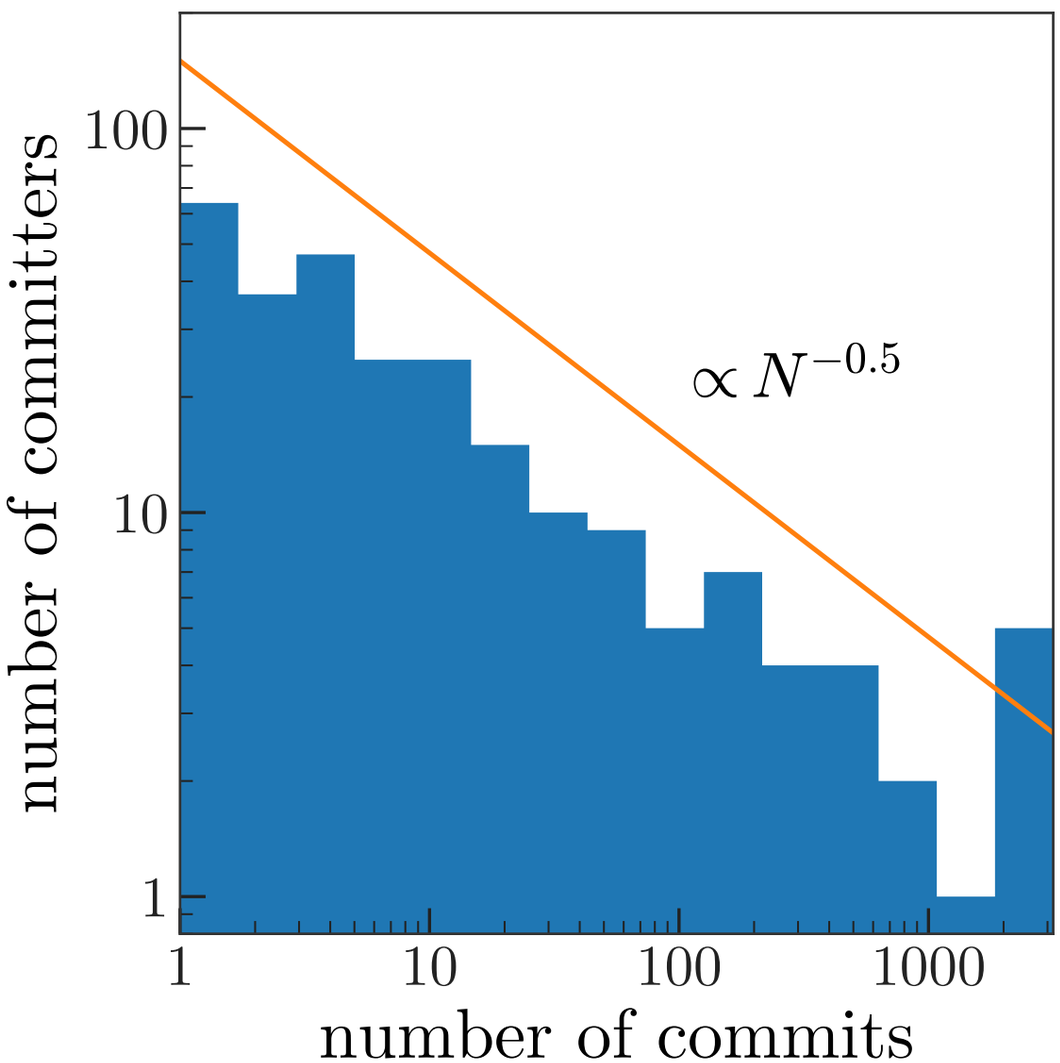
Who develops the Astropy library?

Astronomers!

~270 contributors

~20 package leads & maintainers

but...



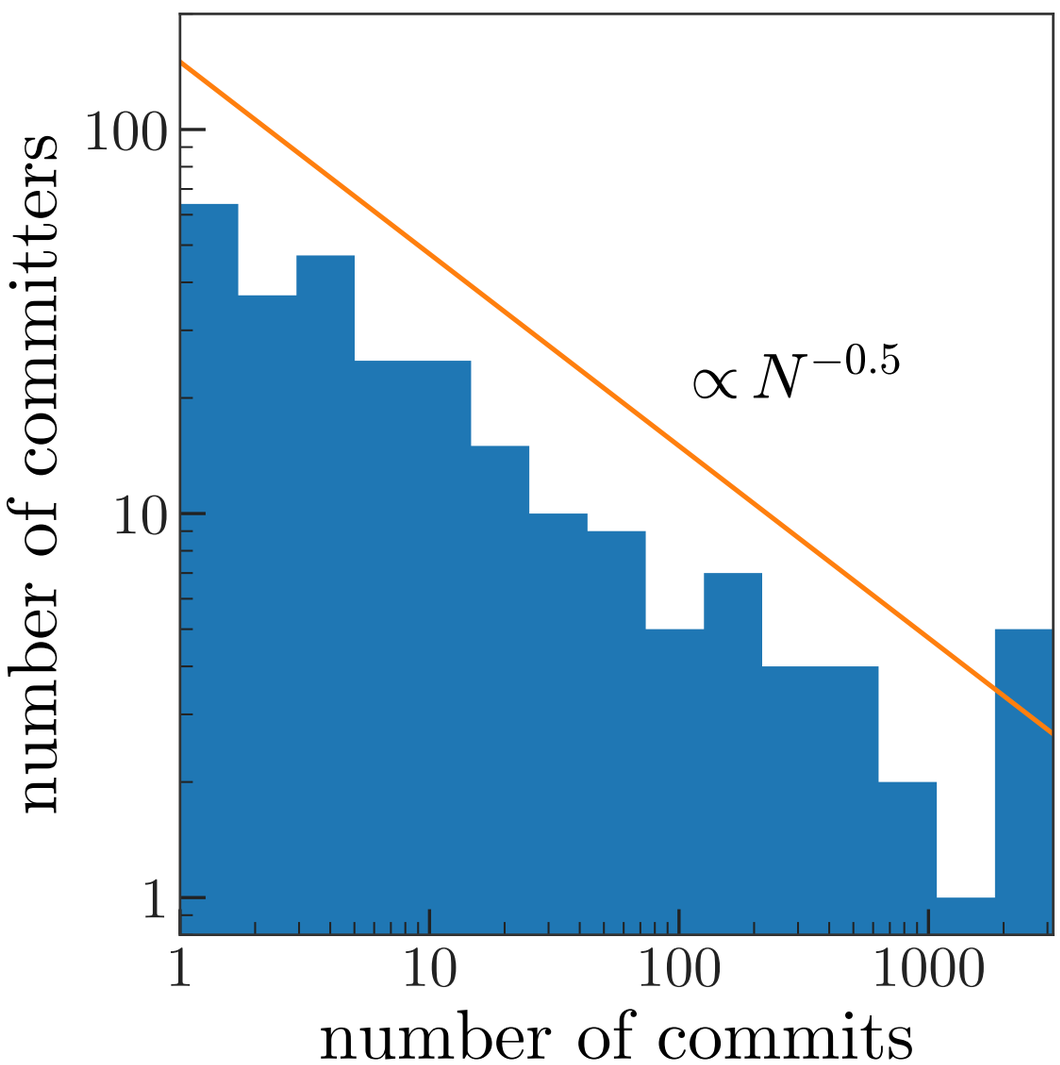
Who develops the Astropy library?

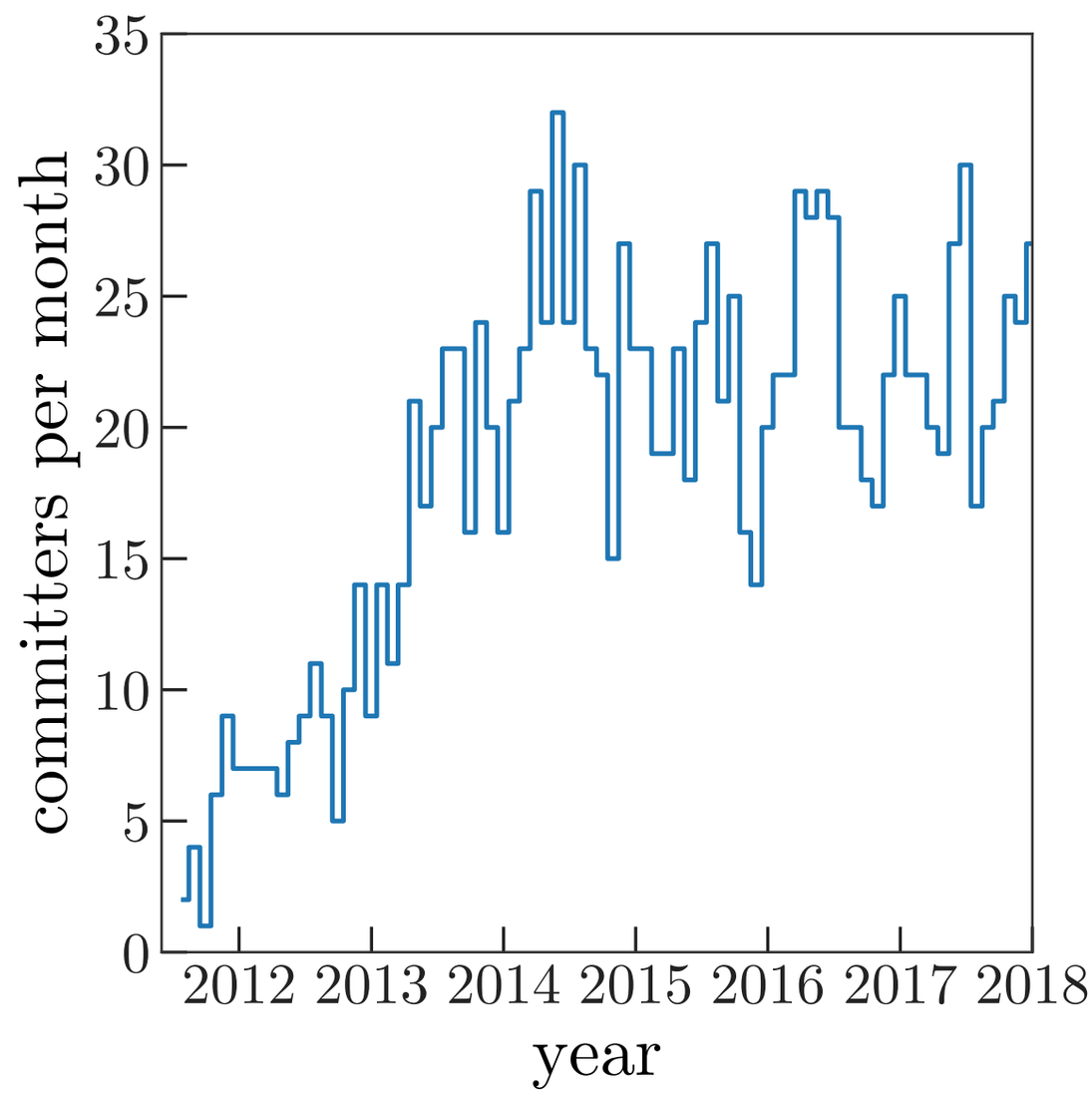
Astronomers!

~270 contributors

~20 package leads & maintainers

but...





		Maintainers	Deputies
Core package release coordinator		Brigitta Sipocz	Tom Robitaille, Erik Tollerud
Sub-package maintainer	astropy.constants	David Shupe	Marten van Kerkwijk
	astropy.convolution	Adam Ginsburg	Axel Donath, Larry Bradley
	astropy.coordinates	Erik Tollerud	Stuart Littlefair, Adrian Price-Whelan
	astropy.cosmology	Alex Conley	*Unfilled*
	astropy.io.ascii	Tom Aldcroft	Hans Moritz Günther
	astropy.io.fits	*Unfilled*	Simon Conseil, Michael Seifert, Dan D'Avella
	astropy.io.misc	Tom Robitaille[1]	Matteo Bachetti
	astropy.io.votable	*Unfilled*	Pey Lian Lim
	astropy.modeling	Nadia Dencheva	Pey Lian Lim
	astropy.nddata	Matt Craig	Steve Crawford, Michael Seifert
	astropy.samp	Tom Robitaille[1]	*Unfilled*
	astropy.stats	Steve Crawford	Larry Bradley
	astropy.table	Tom Aldcroft	Marten van Kerkwijk
	astropy.time	Tom Aldcroft	Marten van Kerkwijk
	astropy.units	Marten van Kerkwijk	Adrian Price-Whelan
	astropy.utils	Pey Lian Lim	Brigitta Sipocz, Erik Tollerud
	astropy.visualization	Larry Bradley	Tom Robitaille
	astropy.wcs	*Unfilled*	Nadia Dencheva

[1] Would prefer deputy role

		Maintainers	Deputies
Core package release coordinator		Brigitta Sipocz	Tom Robitaille, Erik Tollerud
Sub-package maintainer	astropy.constants	David Shupe	Marten van Kerkwijk
	astropy.convolution	Adam Ginsburg	Axel Donath, Larry Bradley
	astropy.coordinates	Erik Tollerud	Stuart Littlefair, Adrian Price-Whelan
	astropy.cosmology	Alex Conley	*Unfilled*
	astropy.io.ascii	Tom Aldcroft	Hans Moritz Günther
	astropy.io.fits	*Unfilled*	Simon Conseil, Michael Seifert, Dan D'Avella
	astropy.io.misc	Tom Robitaille[1]	Matteo Bachetti
	astropy.io.votable	*Unfilled*	Pey Lian Lim
	astropy.modeling	Nadia Dencheva	Pey Lian Lim
	astropy.nddata	Matt Craig	Steve Crawford, Michael Seifert
	astropy.samp	Tom Robitaille[1]	*Unfilled*
	astropy.stats	Steve Crawford	Larry Bradley
	astropy.table	Tom Aldcroft	Marten van Kerkwijk
	astropy.time	Tom Aldcroft	Marten van Kerkwijk
	astropy.units	Marten van Kerkwijk	Adrian Price-Whelan
	astropy.utils	Pey Lian Lim	Brigitta Sipocz, Erik Tollerud
	astropy.visualization	Larry Bradley	Tom Robitaille
	astropy.wcs	*Unfilled*	Nadia Dencheva

[1] Would prefer deputy role

- Evaluating & merging new pull requests by sub-package
- Feature development & issue tracking

The Astropy Coordination Committee

Erik Tollerud

Kelle Cruz

Tom Aldcroft

Tom Robitaille

- Overall coordination and management of the Astropy project
- Evaluating new affiliated packages
- Arbitrating disagreements in the core package
- Managing finances for the project

NUMFOCUS

OPEN CODE = BETTER SCIENCE

"...promote sustainable high-level programming languages, open code development, and reproducible scientific research"

numfocus.org

NUMFOCUS

OPEN CODE = BETTER SCIENCE

SPONSORED PROJECTS

NumPy
jupyter
Open Journals
SymPy
julia
Matplotlib
IPy[y]: IPython
RopenSci
FENICS PROJECT
PyTables
pandas
nteract
QuantEcon
Econ-ARK
AstroPy
Stan
将軍 sho gun
software carpentry
yt
sunpy
PyMC3
DATA CARPENTRY

numfocus.org

Engaging and supporting developers

How to contribute

Important to guide users to and through their first few PRs
Explain expectations: code + tests + docs

Coding guidelines

Follow PEP8 and general style of subpackage you're working in
Avoid multiple inheritance
When to include C code
etc.

docs.astropy.org/en/latest/development/codeguide.html

Engaging and supporting developers

Documentation guidelines

Docstring styles and content (for users)
Comments within code (for developers)

Testing guidelines

Practical issues: where to put tests, how to name them, etc.

Explain concepts and expectations: unit, regression, functional
(i.e. unit tests not enough, even if 100% coverage)

docs.astropy.org/en/latest/development/docguide.html
docs.astropy.org/en/latest/development/testguide.html

Engaging and supporting developers

Coordination meetings
(Astropy project developers)

Developer sprints
(usually during or around other meetings)

Regular telecons
(deadlines are good!)

Google Summer of Code
(2014—present)

Python in Astronomy

Engaging and supporting developers

hacking the scientific attribution system

2013A&A...558A..33A 2013/10 cited: 1521

Astropy: A community Python package for astronomy

Astropy Collaboration; Robitaille, Thomas P.; Tollerud, Erik J. *and 42 more*

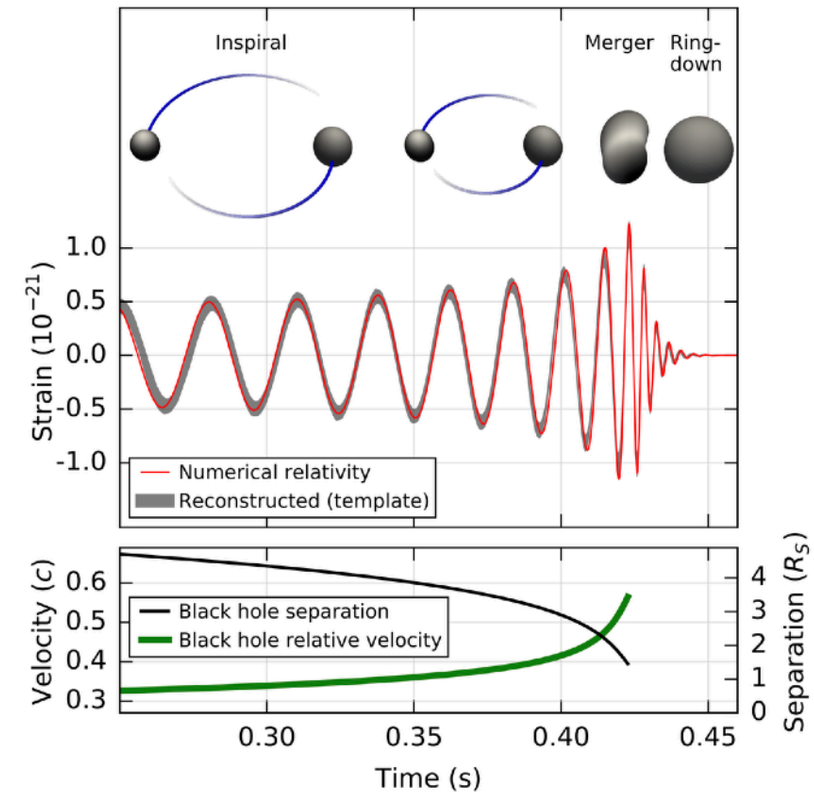
2018AJ....156..123A 2018/09 cited: 142

The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package

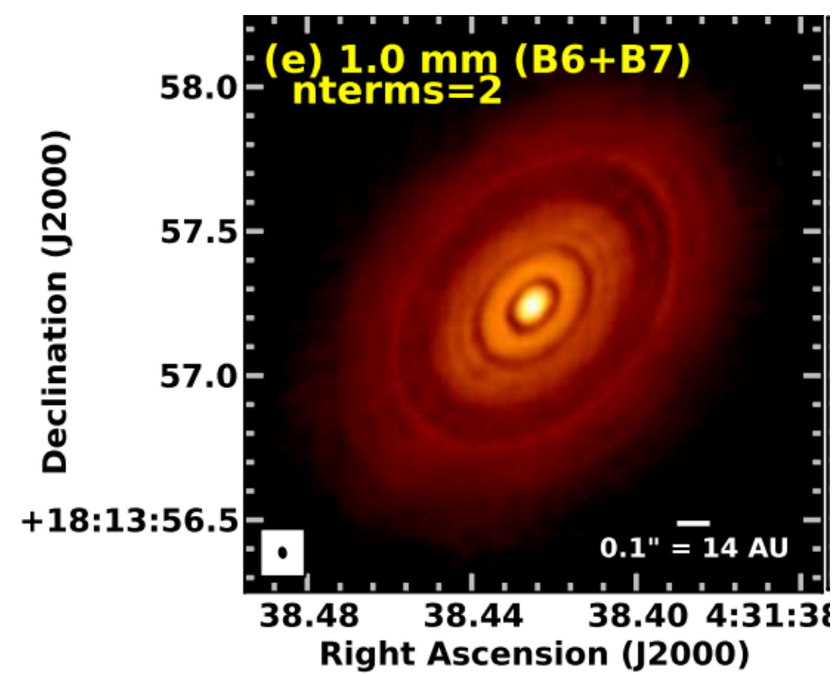
Astropy Collaboration; Price-Whelan, A. M.; Sipőcz, B. M. *and 137 more*

Who uses the Astropy library?

LIGO (detection of gravitational waves)
LIGO (neutron star merger follow-up)



ALMA (first observations)



+ many large observatories, surveys, and regular astronomers!

The Astropy ecosystem

“A common package should not preclude any other Astronomy package from existing, because there will always be more complex and/or specialized tools required.”

Astropy affiliated packages

Domain-specific astronomy Python packages that request to be part of the Astropy community

Commit to Astropy goals: improving reuse, interoperability, interface standards

http://affiliated.astropy.org/

Astropy affiliated packages

Core library:
General tools, long-term stable, longer release schedule

Affiliated packages:
Specialized tools, faster development and release cycle
~40 exist to date

http://affiliated.astropy.org/

Custom tools we use, develop, and release

4 pytest plugins
(all publicly released,
see here https://github.com/astropy/pytest-astropy)

Sphinx extensions
(see https://github.com/astropy/sphinx-automodapi)

Python package template
(see https://github.com/astropy/package-template)

Benchmarking tool: airspeed velocity
(see https://asv.readthedocs.io/en/stable/)

Astropy package template

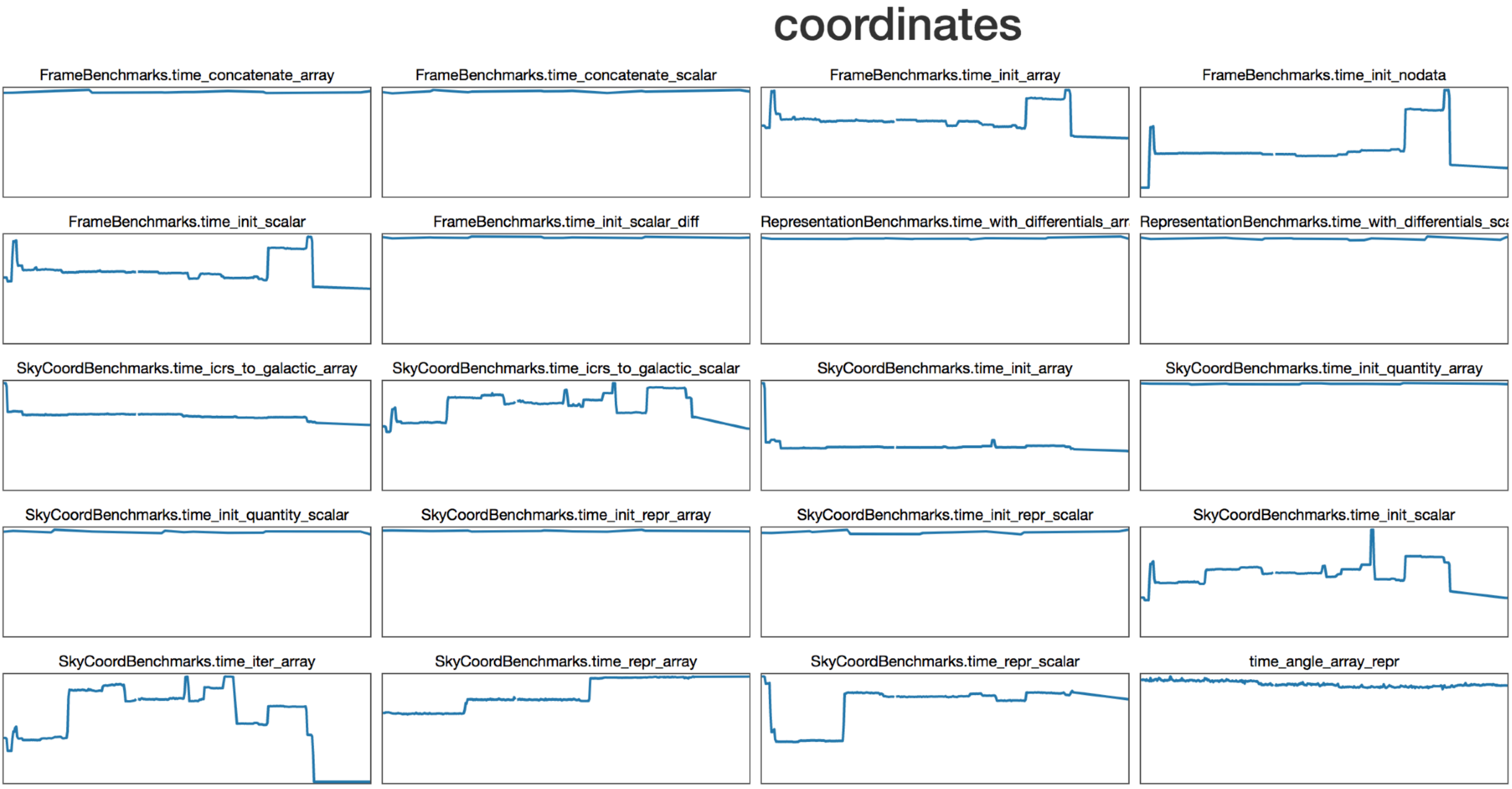
Goal: simplify startup and maintenance of packaged, released Python software

Sets up:
documentation template (Sphinx)
testing infrastructure (py.test)
CI configuration
setup.py script (support for Cython exts.)

(Uses cookiecutter)

https://github.com/astropy/package-template

Custom benchmarking



www.astropy.org/astropy-benchmarks

The Astropy community

“Our goal is to keep ours a positive, inclusive, successful, and growing community”

What is the Astropy community?

Superset of users & developers

Many entry points:
- Mailing lists: users, developers
- Slack
- Regular telecons
- New: Astropy event calendar

Conferences:
- Workshops
- Python in Astronomy
- .Astronomy

Engaging and supporting users

Documentation is *not enough*

Engaging and supporting users

Guides

Guides are comprehensive, conceptually-focused documents providing stand-alone introductions to core packages in addition to the underlying astronomical concepts.

Tutorials

Tutorials are step-by-step cookbooks for common activities that incorporate several packages. They are more specific and less conceptual than Guides but more extended than Examples.

Documentation

Documentation is the complete description of a package with all requisite details, including usage, dependencies, and examples.

Examples

Examples are stand-alone code snippets that live in the astropy documentation that demonstrate a specific functionality within a package.

Tutorials

Jupyter notebooks

Rendered to static HTML
OR
Open as live notebook with
Google Colaboratory

colab.research.google.com

Challenges & the future

Many lead / maintainer roles unfilled

How do we prevent burnout and support devs?

User -> Contributor -> Maintainer?

A significant development bottleneck is code review

How can we incentivize this effort?

Our goal is to enable *all* astronomy, not solve specific science questions

How do we fund "infrastructure" software like Astropy?

Challenges & the future

Tradeoff between API stability & betterizing / updating
(example: astropy.units is very general and used outside of astronomy, but is it worth the headache [to users & devs] of splitting it out?)

Incentivize performance enhancements

Improve educational materials