emcee

Dan Foreman-Mackey

CCA@Flatiron // dfm.io // @exoplaneteer // github.com/dfm

A modular ecosystem for probabilistic data analysis

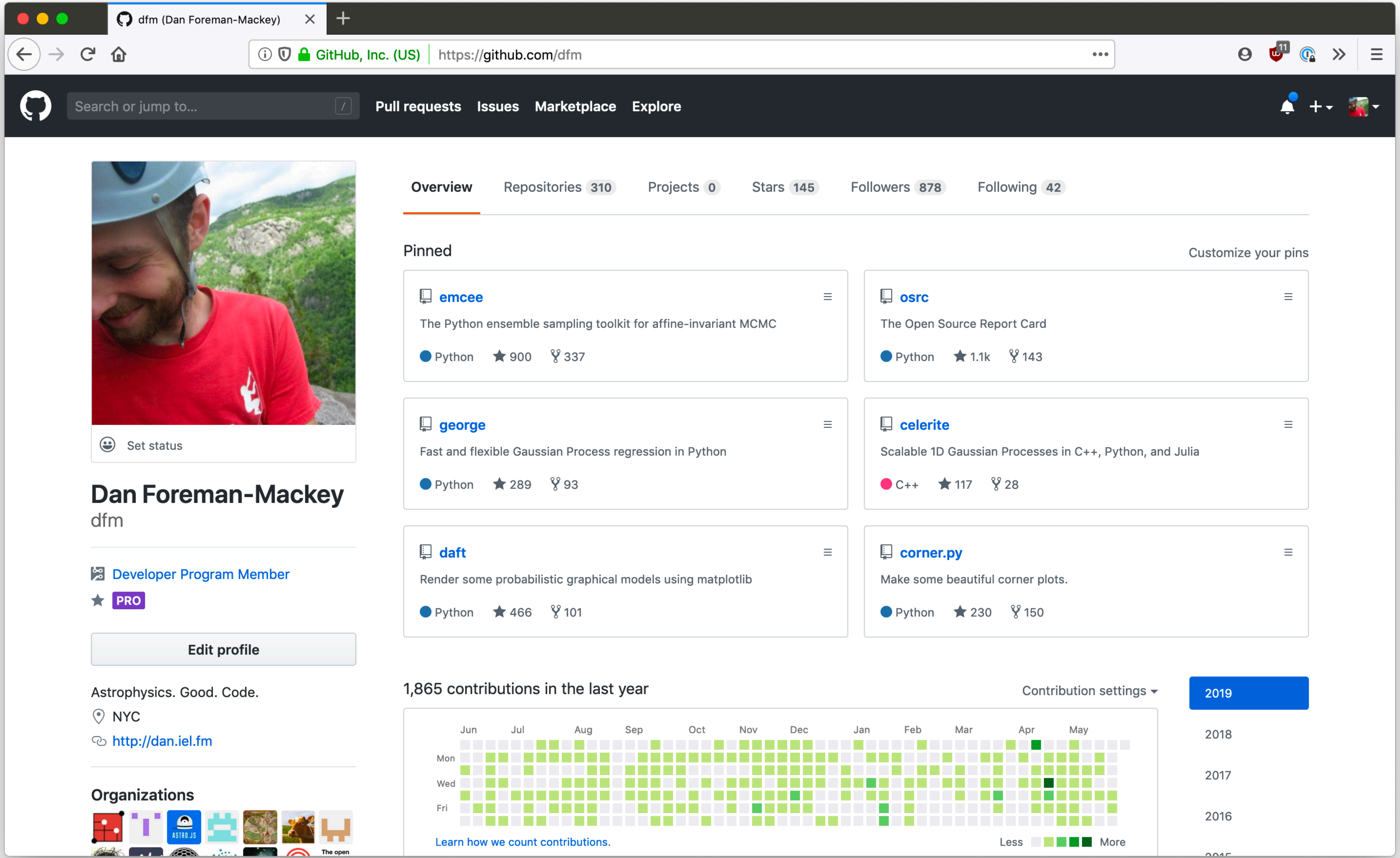
including *emcee*

Dan Foreman-Mackey

CCA@Flatiron // dfm.io // @exoplaneteer // github.com/dfm

Slides can be found at:

speakerdeck.com/dfm



dfm (Dan Foreman-Mackey)

GitHub, Inc. (US) https://github.com/dfm

Search or jump to...

Pull requests Issues Marketplace Explore

Set status

Dan Foreman-Mackey
dfm

Developer Program Member

PRO

Edit profile

Astrophysics. Good. Code.

NYC

http://dan.iel.fm

Organizations

Overview Repositories 310 Projects 0 Stars 145 Followers 878 Following 42

Pinned

Customize your pins

emcee
The Python ensemble sampling toolkit for affine-invariant MCMC
Python ★ 900 337

osrc
The Open Source Report Card
Python ★ 1.1k 143

george
Fast and flexible Gaussian Process regression in Python
Python ★ 289 93

celerite
Scalable 1D Gaussian Processes in C++, Python, and Julia
C++ ★ 117 28

daft
Render some probabilistic graphical models using matplotlib
Python ★ 466 101

corner.py
Make some beautiful corner plots.
Python ★ 230 150

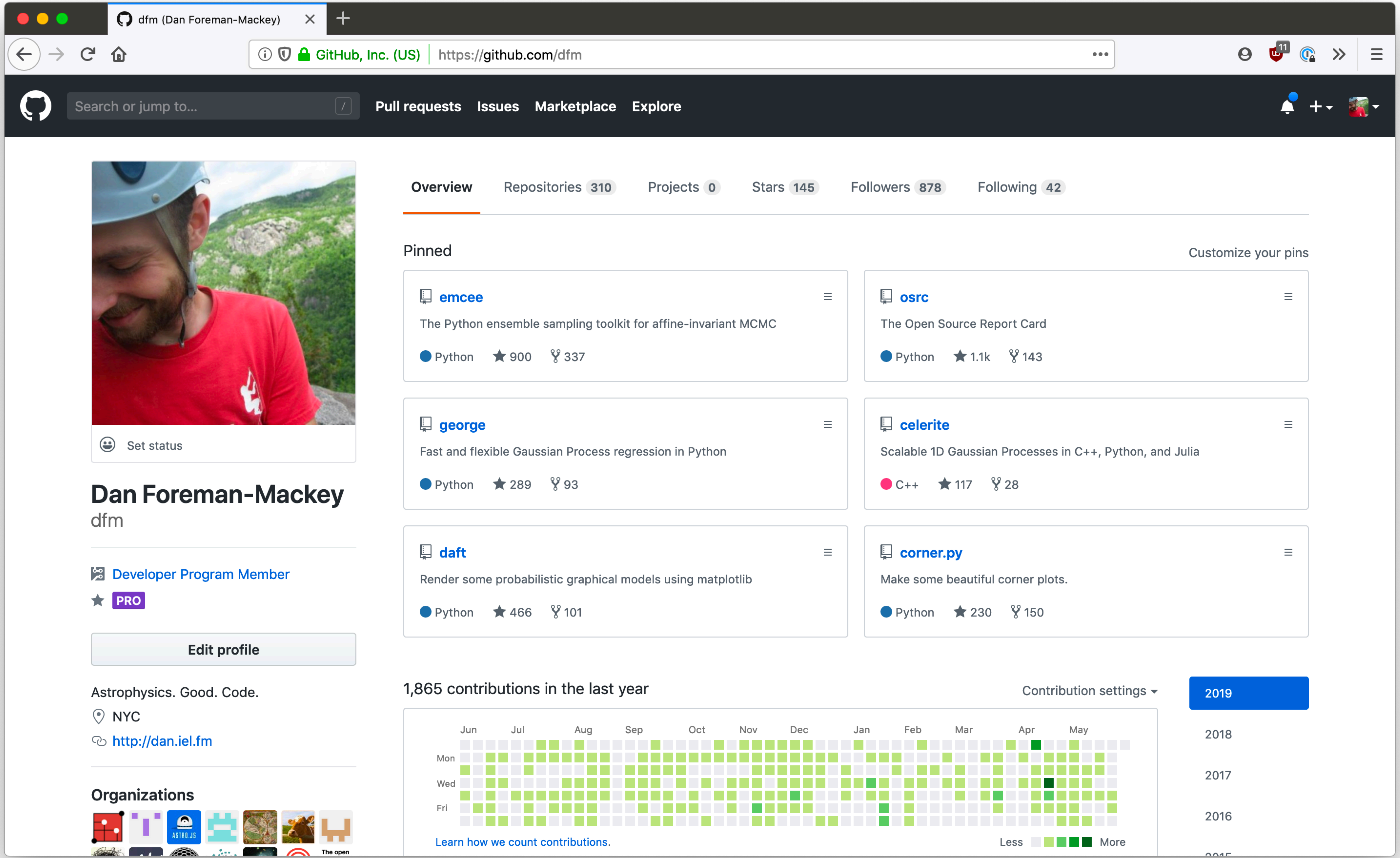
1,865 contributions in the last year

Contribution settings

Learn how we count contributions.

2019
2018
2017
2016

I have a GitHub **problem**.



Dan Foreman-Mackey
dfm

Developer Program Member
PRO

Astrophysics. Good. Code.
NYC
http://dan.iel.fm

Organizations

Pinned

emcee
The Python ensemble sampling toolkit for affine-invariant MCMC
Python ★ 900 337

osrc
The Open Source Report Card
Python ★ 1.1k 143

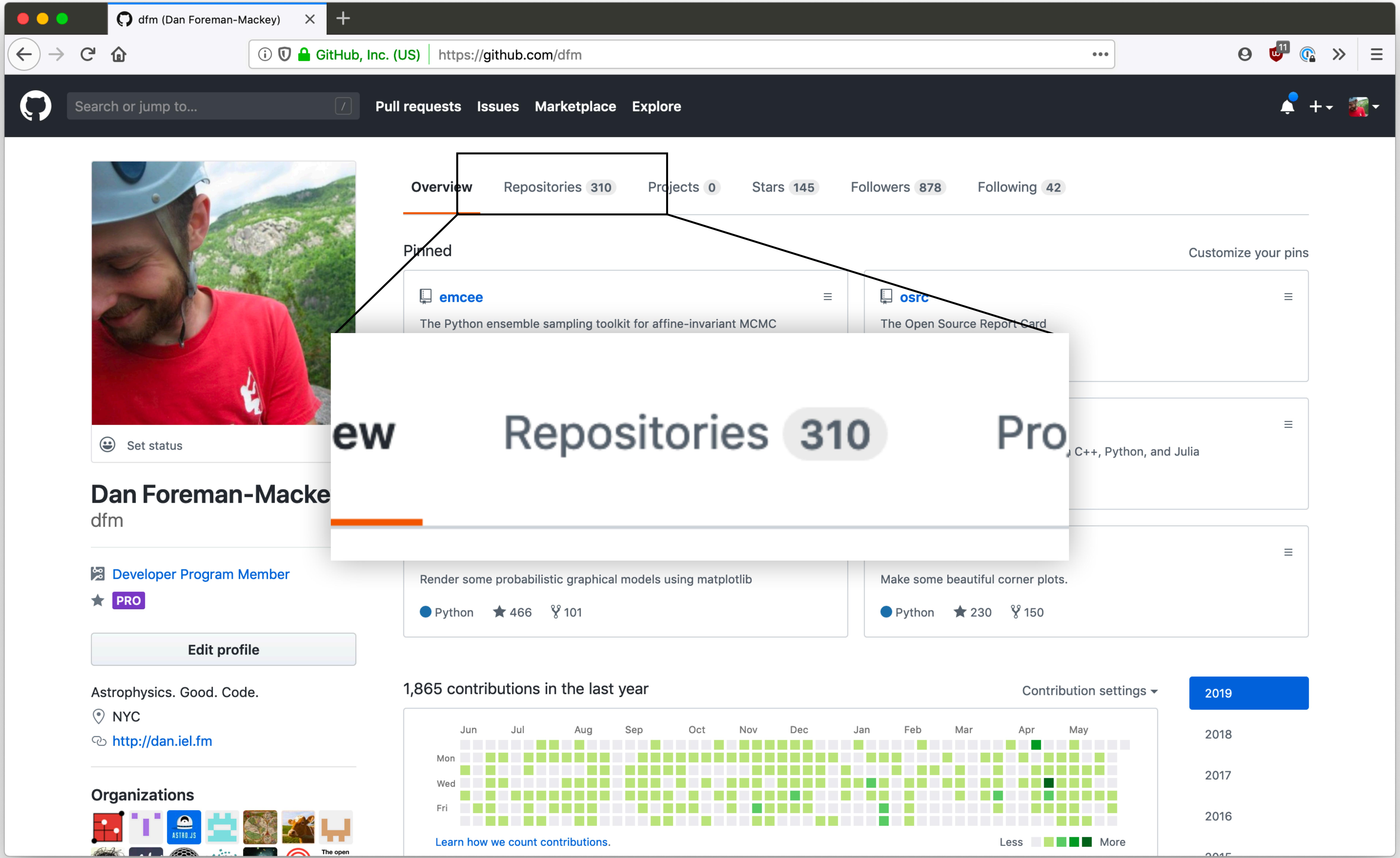
george
Fast and flexible Gaussian Process regression in Python
Python ★ 289 93

celerite
Scalable 1D Gaussian Processes in C++, Python, and Julia
C++ ★ 117 28

daft
Render some probabilistic graphical models using matplotlib
Python ★ 466 101

corner.py
Make some beautiful corner plots.
Python ★ 230 150

1,865 contributions in the last year



Dan Foreman-Macke

dfm

Astrophysics. Good. Code.

NYC

http://dan.iel.fm

emcee: The Python ensemble sampling toolkit for affine-invariant MCMC

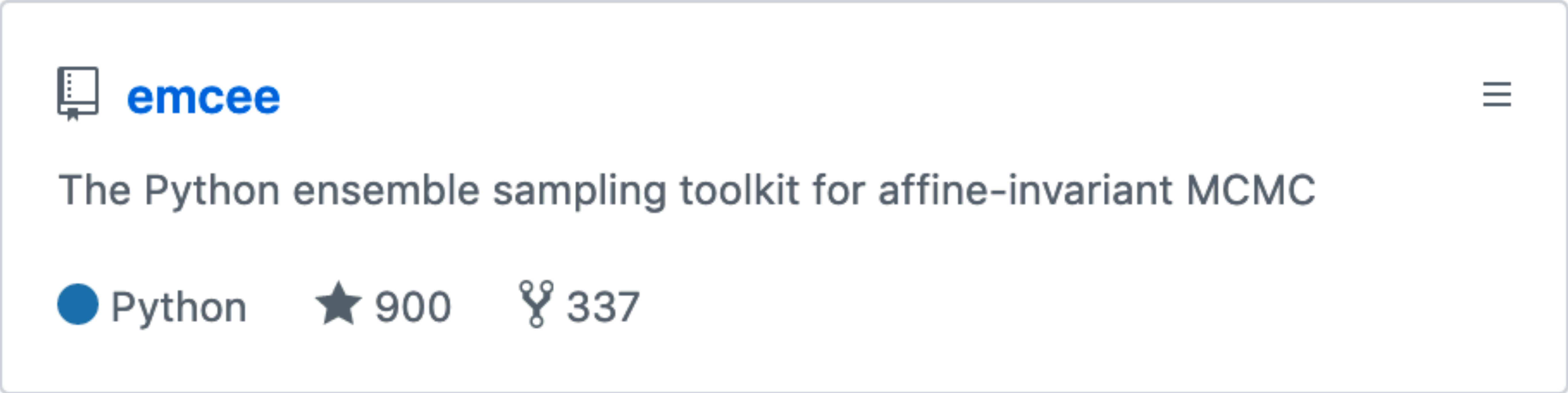
osrc: The Open Source Report Card

C++, Python, and Julia

Render some probabilistic graphical models using matplotlib

Make some beautiful corner plots.

1,865 contributions in the last year



emcee

The Python ensemble sampling toolkit for affine-invariant MCMC

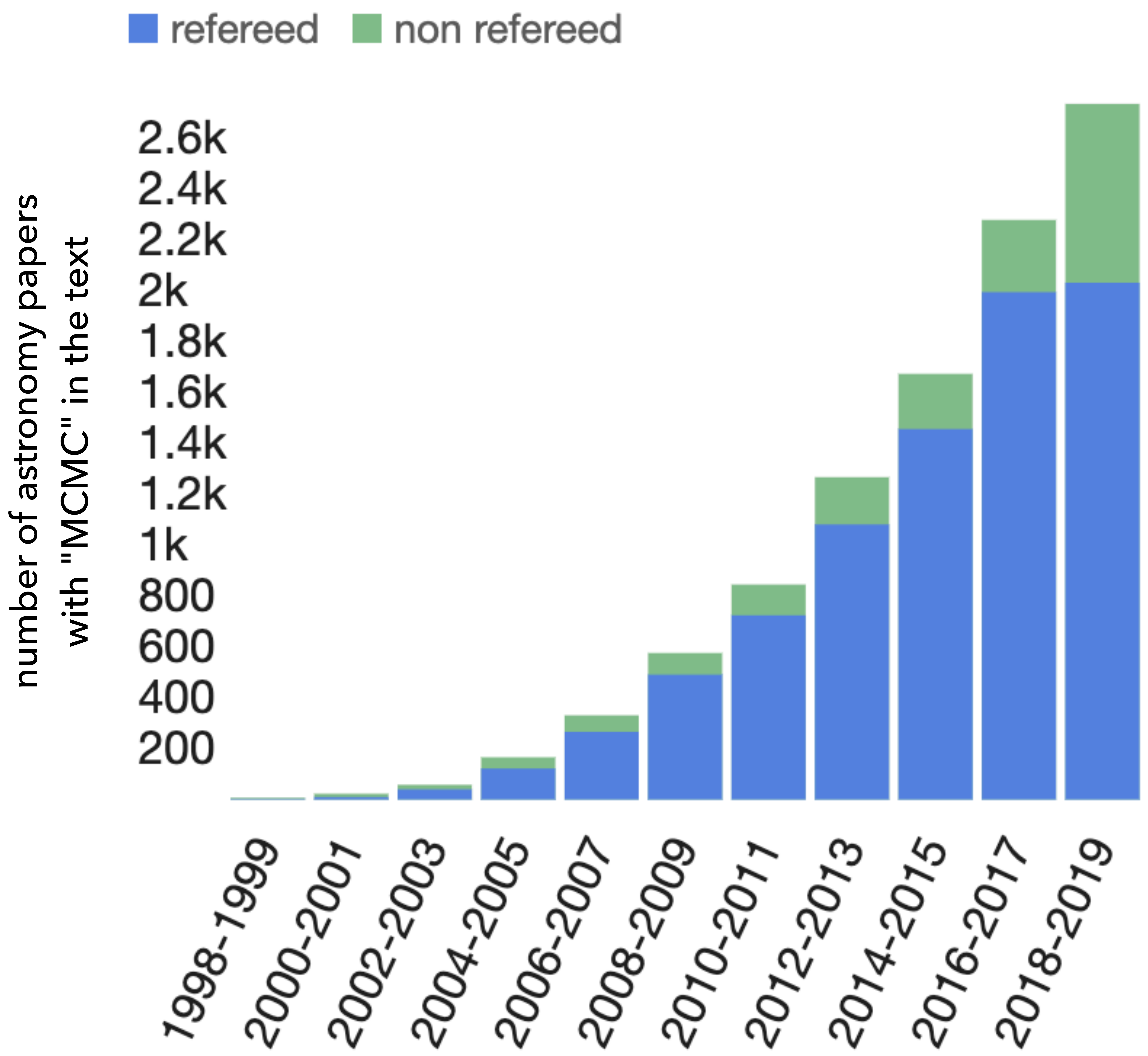
● Python ★ 900 ⑂ 337

1

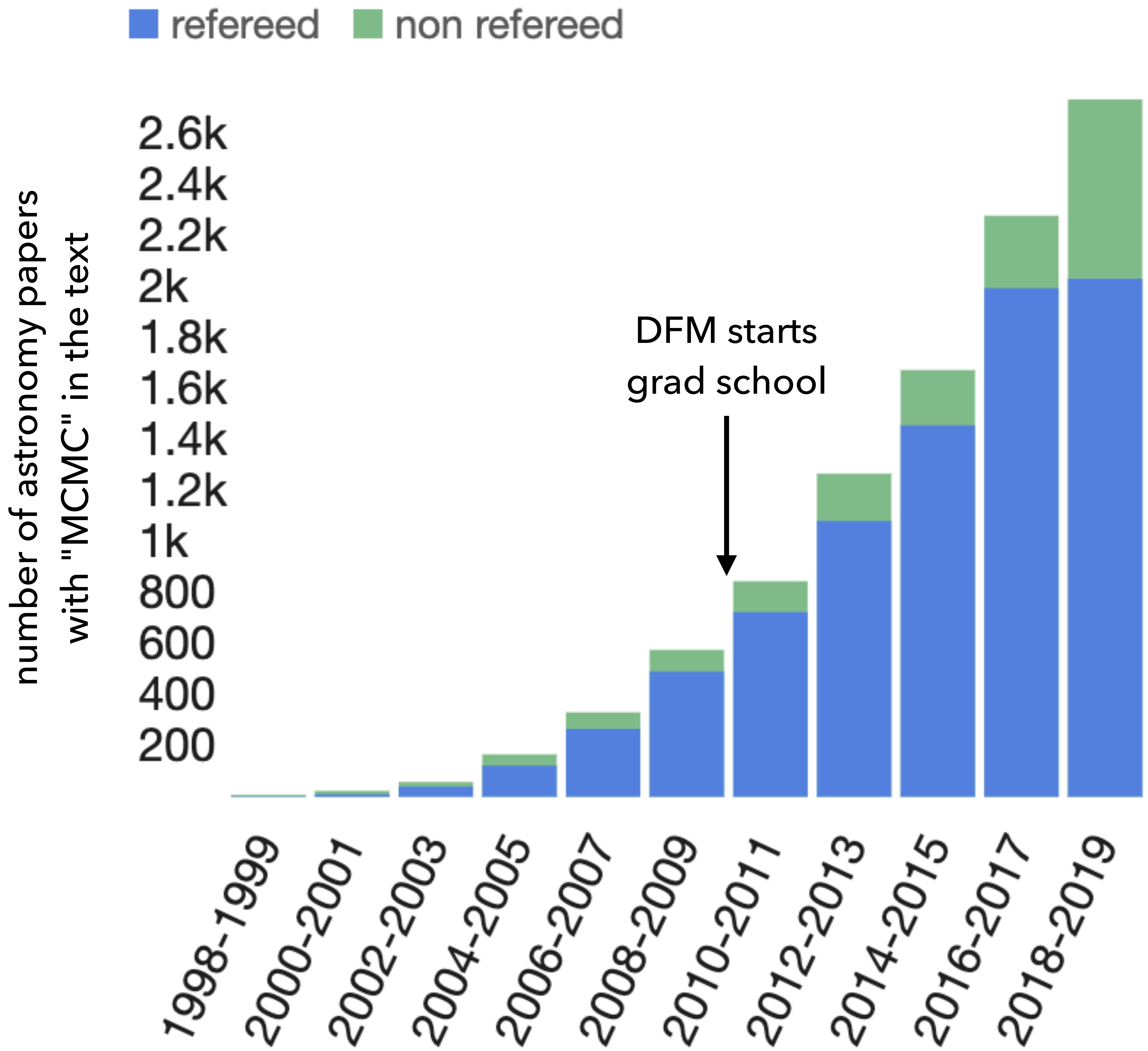
Context

p(physics | data)

Markov Chain Monte Carlo



ui.adsabs.harvard.edu



DFM starts
grad school

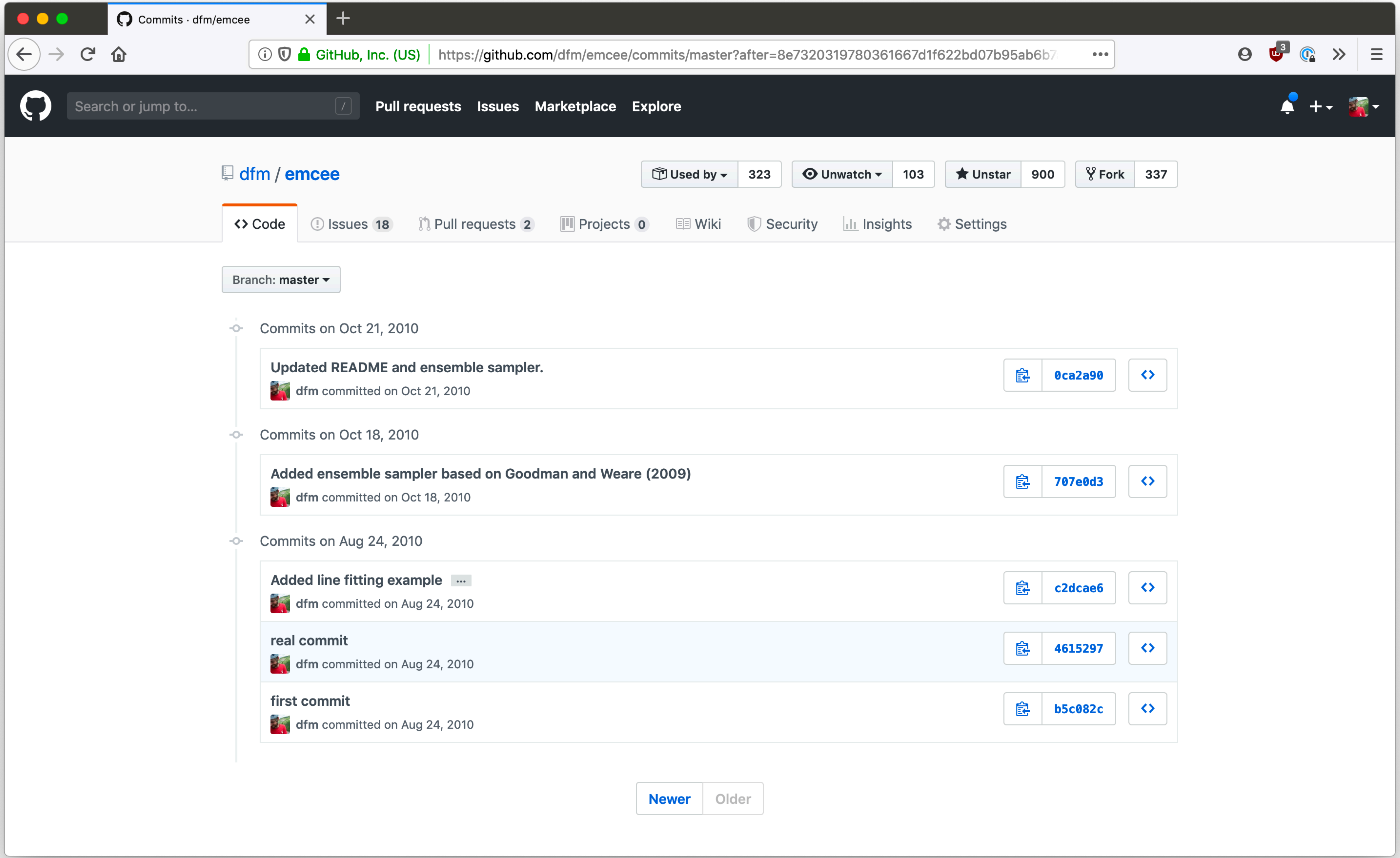
ui.adsabs.harvard.edu

In **2010**:

Everyone wrote their own MCMC sampler.

In **2010**:

So that's what I did too.



Commits on Oct 21, 2010

Updated README and ensemble sampler.
dfm committed on Oct 21, 2010
0ca2a90

Commits on Oct 18, 2010

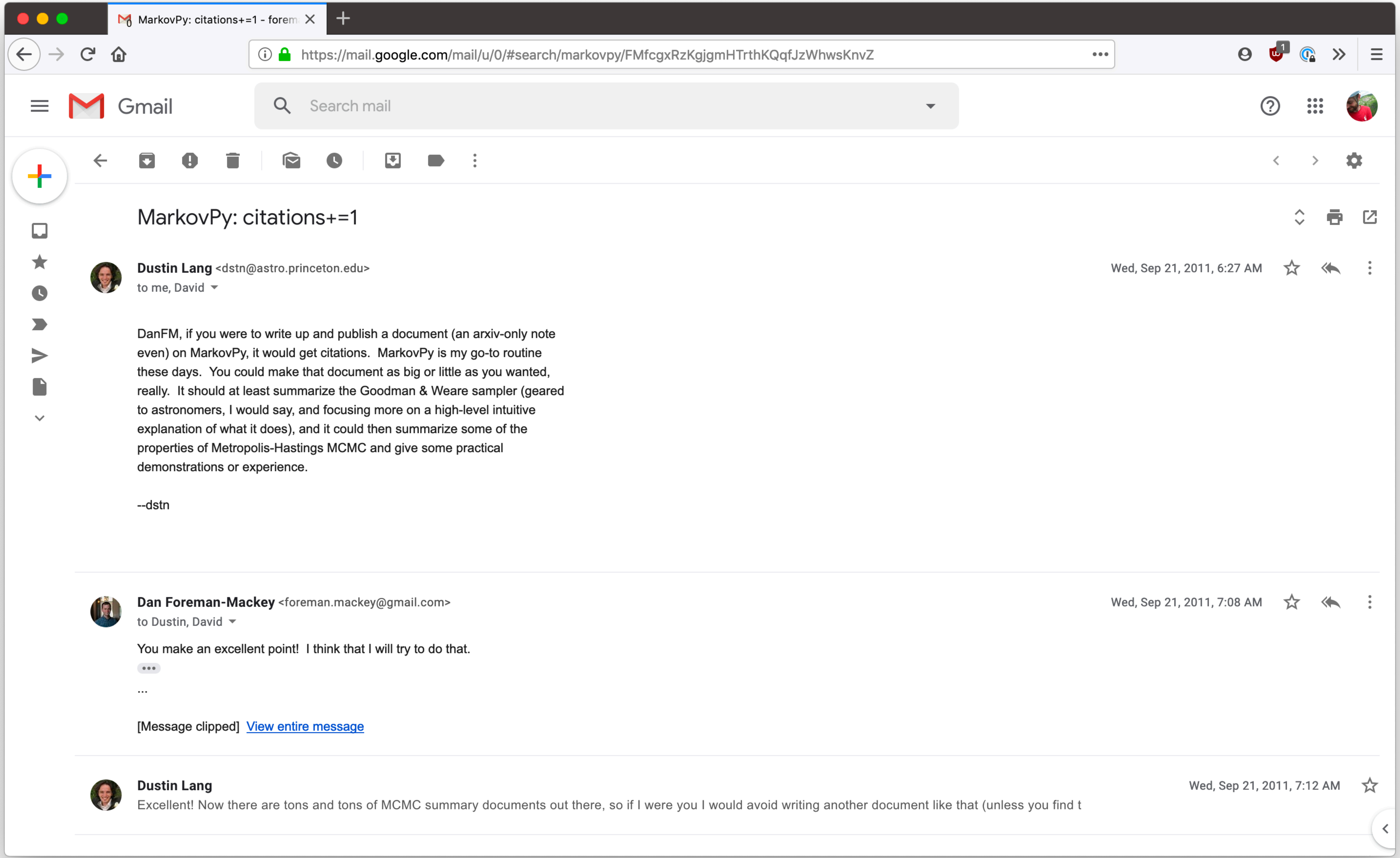
Added ensemble sampler based on Goodman and Weare (2009)
dfm committed on Oct 18, 2010
707e0d3

Commits on Aug 24, 2010

Added line fitting example …
dfm committed on Aug 24, 2010
c2dcae6

real commit
dfm committed on Aug 24, 2010
4615297

first commit
dfm committed on Aug 24, 2010
b5c082c


MarkovPy: citations+=1 - forem
https://mail.google.com/mail/u/0/#search/markovpy/FMfcgxRzKgjgmHTrthKQqfJzWhwsKnvZ
Gmail
Search mail

MarkovPy: citations+=1

Dustin Lang <dstn@astro.princeton.edu>
to me, David

Wed, Sep 21, 2011, 6:27 AM

DanFM, if you were to write up and publish a document (an arxiv-only note
even) on MarkovPy, it would get citations. MarkovPy is my go-to routine
these days. You could make that document as big or little as you wanted,
really. It should at least summarize the Goodman & Weare sampler (geared
to astronomers, I would say, and focusing more on a high-level intuitive
explanation of what it does), and it could then summarize some of the
properties of Metropolis-Hastings MCMC and give some practical
demonstrations or experience.

--dstn

Dan Foreman-Mackey <foreman.mackey@gmail.com>
to Dustin, David

Wed, Sep 21, 2011, 7:08 AM

You make an excellent point! I think that I will try to do that.

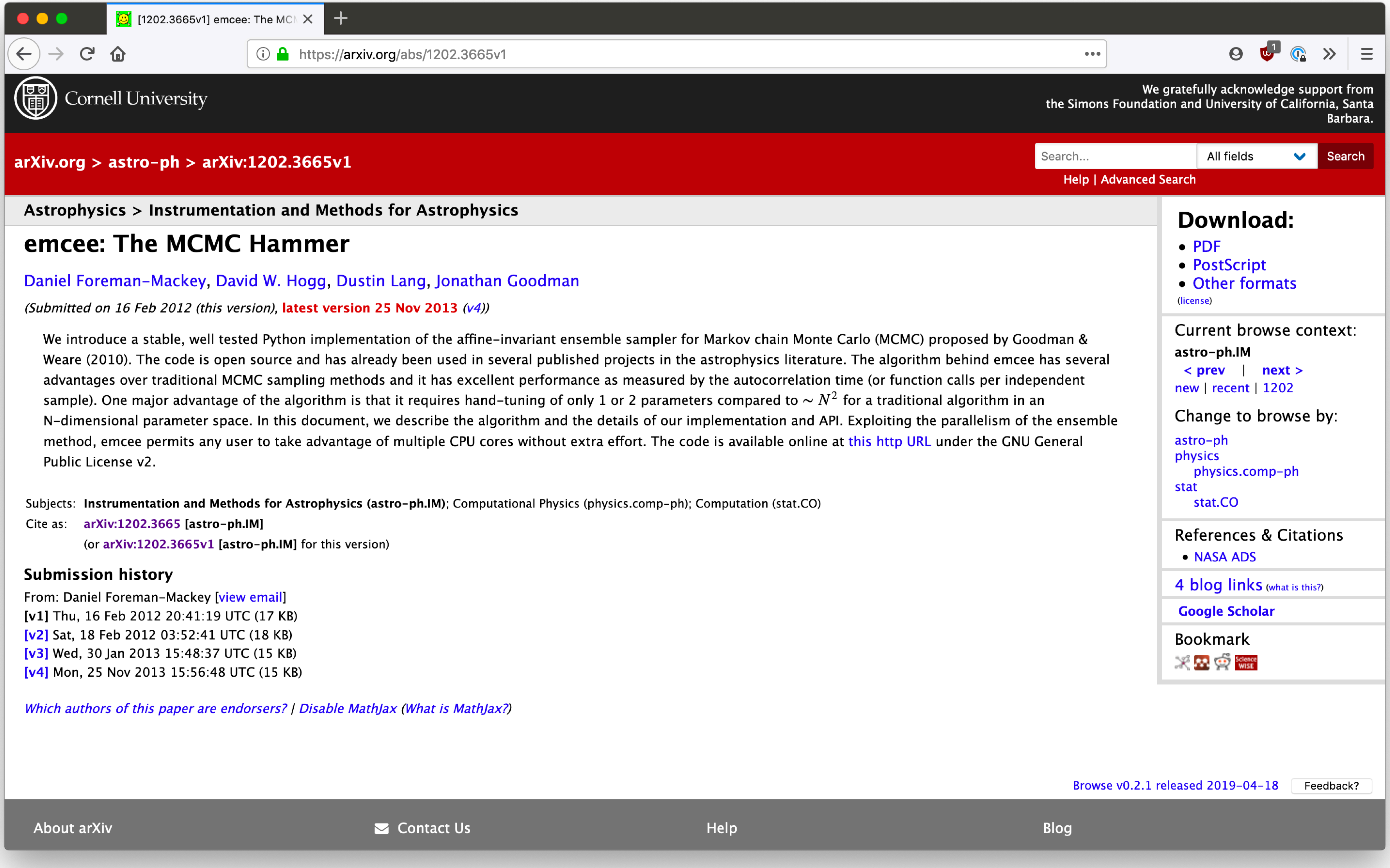
...

[Message clipped] View entire message

Dustin Lang

Wed, Sep 21, 2011, 7:12 AM

Excellent! Now there are tons and tons of MCMC summary documents out there, so if I were you I would avoid writing another document like that (unless you find t



[1202.3665v1] emcee: The MC
https://arxiv.org/abs/1202.3665v1

Cornell University

We gratefully acknowledge support from the Simons Foundation and University of California, Santa Barbara.

arXiv.org > astro-ph > arXiv:1202.3665v1

Search...
All fields
Search
Help | Advanced Search

Astrophysics > Instrumentation and Methods for Astrophysics

emcee: The MCMC Hammer

Daniel Foreman-Mackey, David W. Hogg, Dustin Lang, Jonathan Goodman

(Submitted on 16 Feb 2012 (this version), latest version 25 Nov 2013 (v4))

We introduce a stable, well tested Python implementation of the affine-invariant ensemble sampler for Markov chain Monte Carlo (MCMC) proposed by Goodman & Weare (2010). The code is open source and has already been used in several published projects in the astrophysics literature. The algorithm behind emcee has several advantages over traditional MCMC sampling methods and it has excellent performance as measured by the autocorrelation time (or function calls per independent sample). One major advantage of the algorithm is that it requires hand-tuning of only 1 or 2 parameters compared to $\sim N^2$ for a traditional algorithm in an N-dimensional parameter space. In this document, we describe the algorithm and the details of our implementation and API. Exploiting the parallelism of the ensemble method, emcee permits any user to take advantage of multiple CPU cores without extra effort. The code is available online at this http URL under the GNU General Public License v2.

Subjects: **Instrumentation and Methods for Astrophysics (astro-ph.IM)**; Computational Physics (physics.comp-ph); Computation (stat.CO)
Cite as: arXiv:1202.3665 **[astro-ph.IM]**
(or arXiv:1202.3665v1 **[astro-ph.IM]** for this version)

Submission history

From: Daniel Foreman-Mackey [view email]
[v1] Thu, 16 Feb 2012 20:41:19 UTC (17 KB)
[v2] Sat, 18 Feb 2012 03:52:41 UTC (18 KB)
[v3] Wed, 30 Jan 2013 15:48:37 UTC (15 KB)
[v4] Mon, 25 Nov 2013 15:56:48 UTC (15 KB)

Which authors of this paper are endorsers? | Disable MathJax (What is MathJax?)

Download:

- PDF
- PostScript
- Other formats

(license)

Current browse context:
astro-ph.IM
< prev | next >
new | recent | 1202

Change to browse by:
astro-ph
physics
physics.comp-ph
stat
stat.CO

References & Citations
- NASA ADS

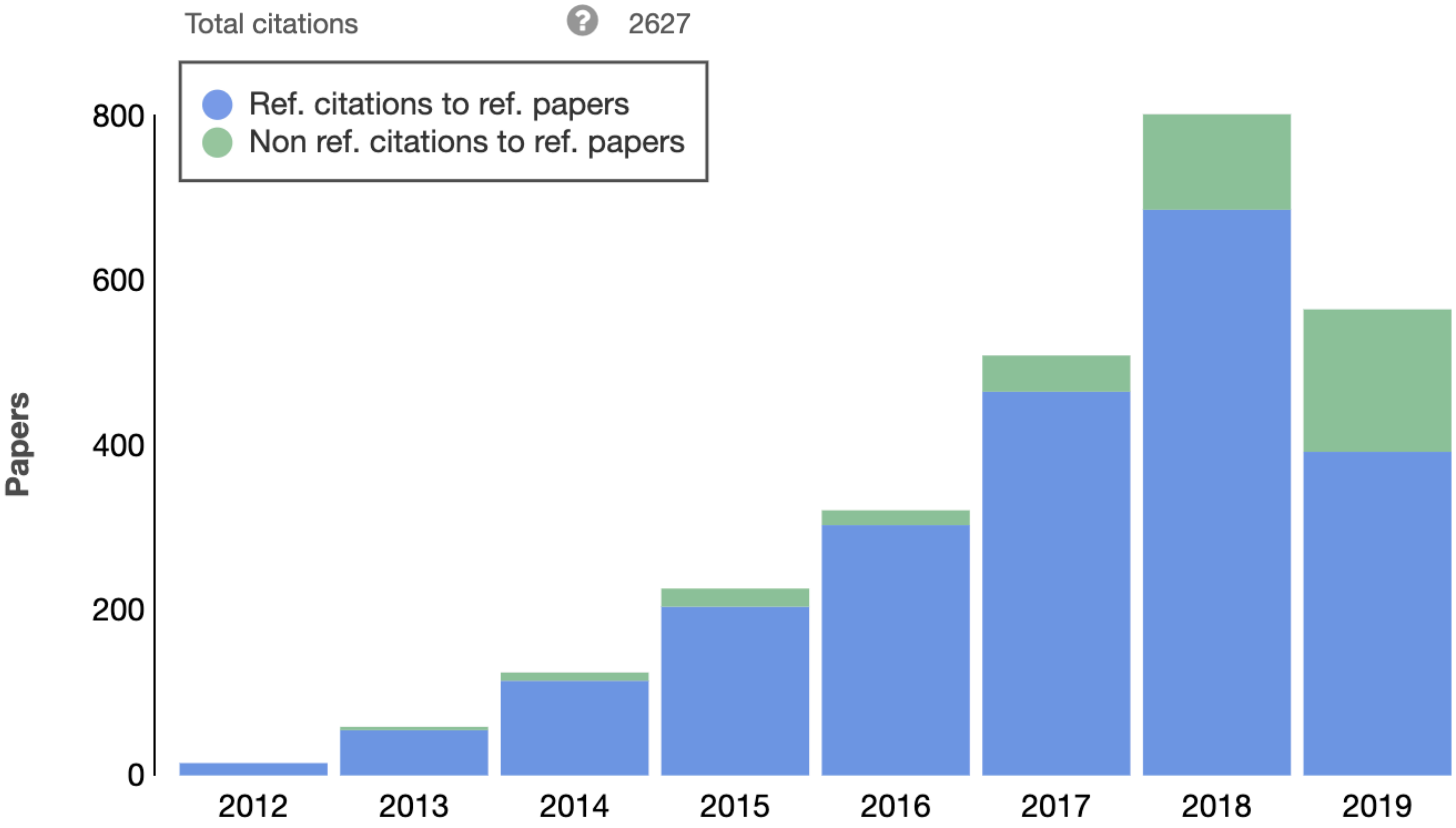
4 blog links (what is this?)

Google Scholar

Bookmark

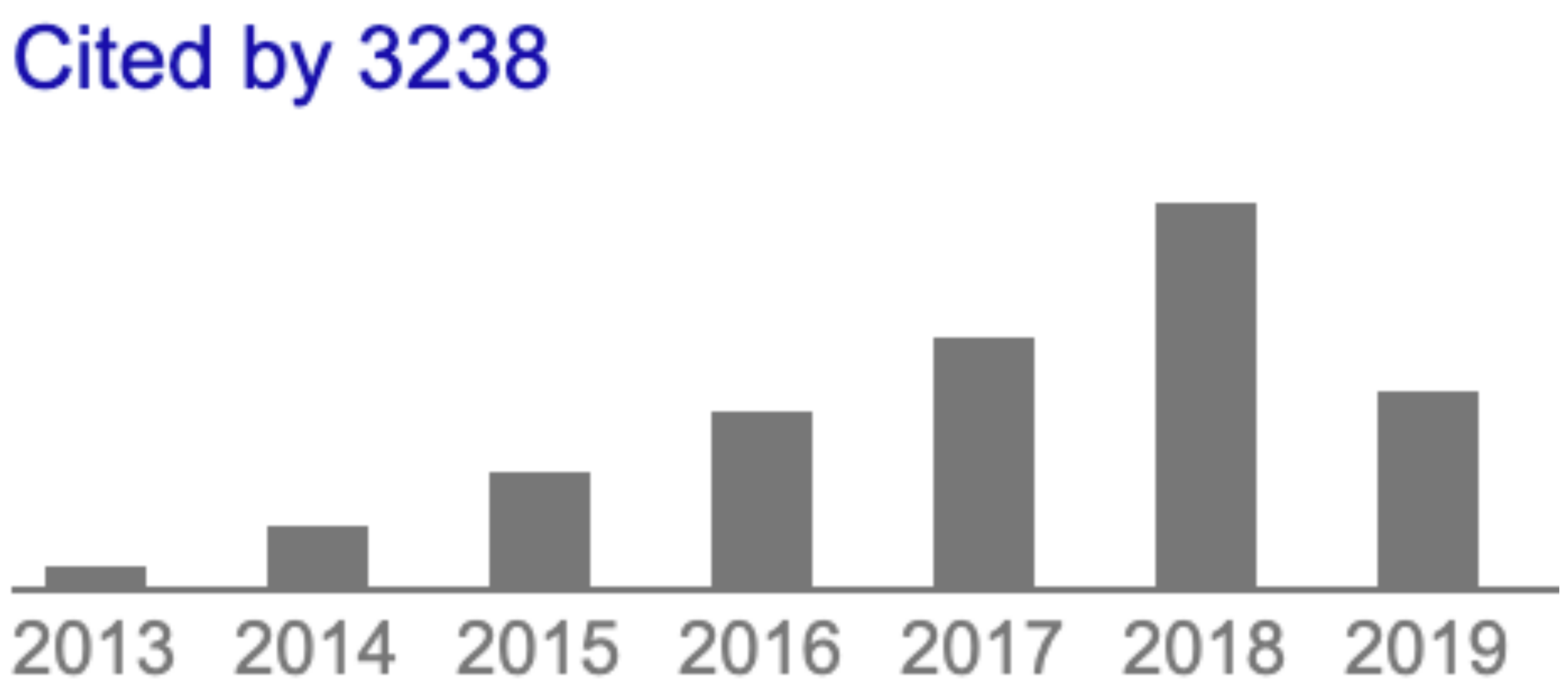
Browse v0.2.1 released 2019-04-18 Feedback?

About arXiv Contact Us Help Blog



Total citations 2627

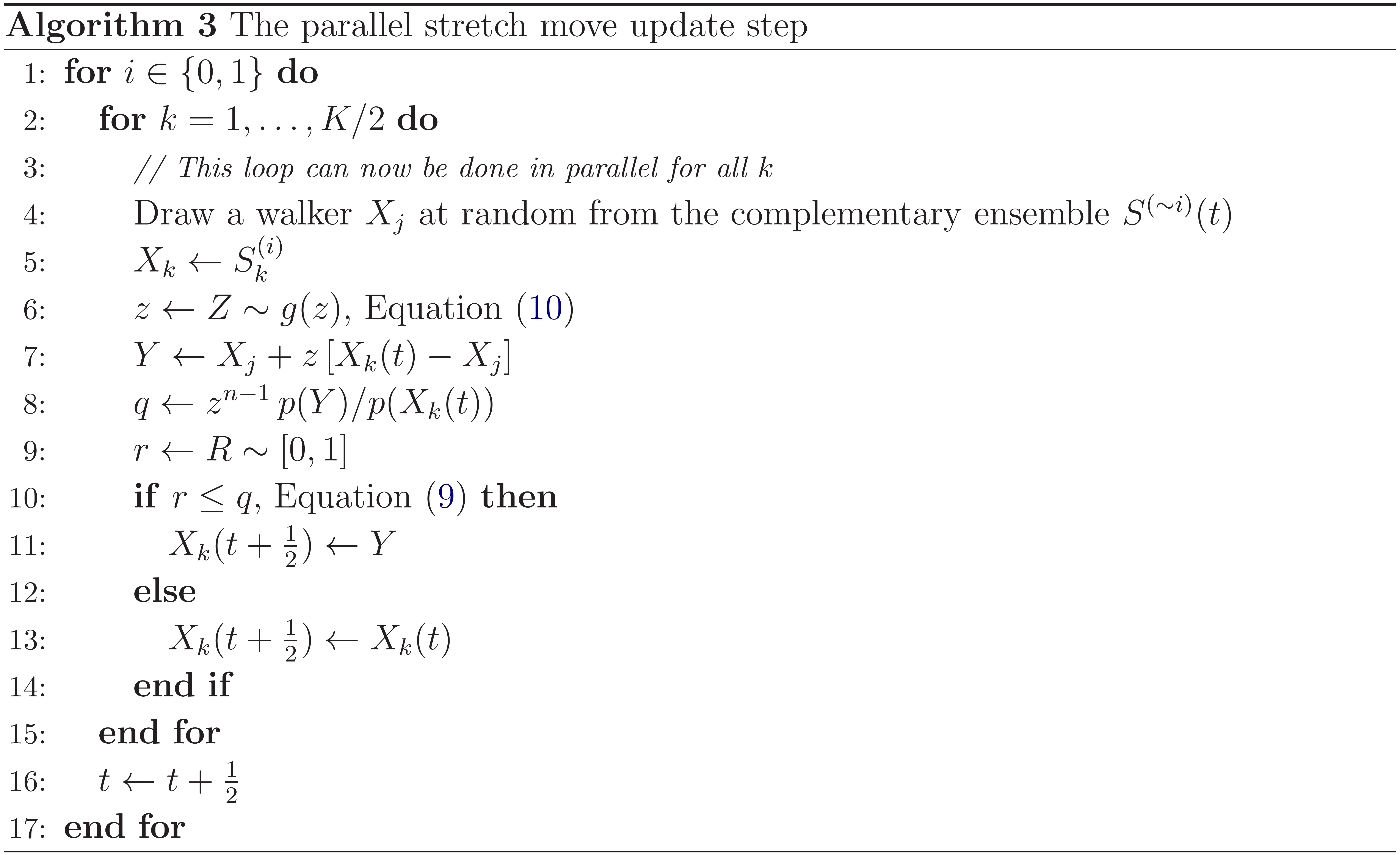
ui.adsabs.harvard.edu/abs/2013PASP..125..306F/metrics



Cited by 3238

scholar.google.com

The algorithm is nearly trivial.

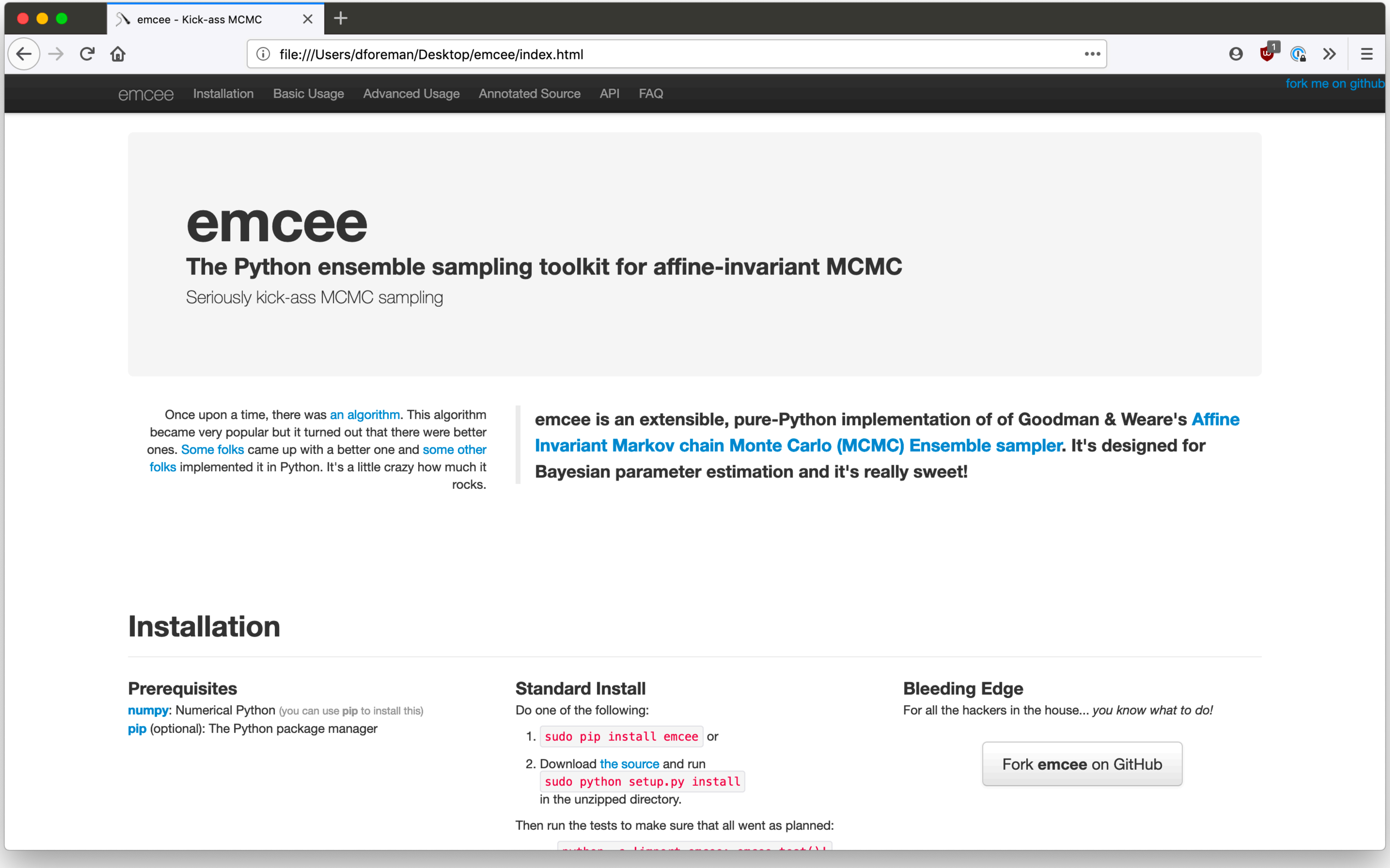


Algorithm 3 The parallel stretch move update step

```
for $i \in \{0, 1\}$ do
    for $k = 1, \ldots, K/2$ do
        // This loop can now be done in parallel for all $k$
        Draw a walker $X_j$ at random from the complementary ensemble $S^{(\sim i)}(t)$
        $X_k \leftarrow S_k^{(i)}$
        $z \leftarrow Z \sim g(z)$, Equation (10)
        $Y \leftarrow X_j + z\,[X_k(t) - X_j]$
        $q \leftarrow z^{n-1}\, p(Y)/p(X_k(t))$
        $r \leftarrow R \sim [0, 1]$
        if $r \leq q$, Equation (9) then
            $X_k(t + \frac{1}{2}) \leftarrow Y$
        else
            $X_k(t + \frac{1}{2}) \leftarrow X_k(t)$
        end if
    end for
    $t \leftarrow t + \frac{1}{2}$
end for
```

DFM+ (2013)

So why is it so **popular**?



emcee

The Python ensemble sampling toolkit for affine-invariant MCMC

Seriously kick-ass MCMC sampling

Once upon a time, there was an algorithm. This algorithm became very popular but it turned out that there were better ones. Some folks came up with a better one and some other folks implemented it in Python. It's a little crazy how much it rocks.

emcee is an extensible, pure-Python implementation of of Goodman & Weare's Affine Invariant Markov chain Monte Carlo (MCMC) Ensemble sampler. It's designed for Bayesian parameter estimation and it's really sweet!

Installation

Prerequisites

numpy: Numerical Python (you can use **pip** to install this)
pip (optional): The Python package manager

Standard Install

Do one of the following:

1. `sudo pip install emcee` or
2. Download the source and run `sudo python setup.py install` in the unzipped directory.

Then run the tests to make sure that all went as planned:

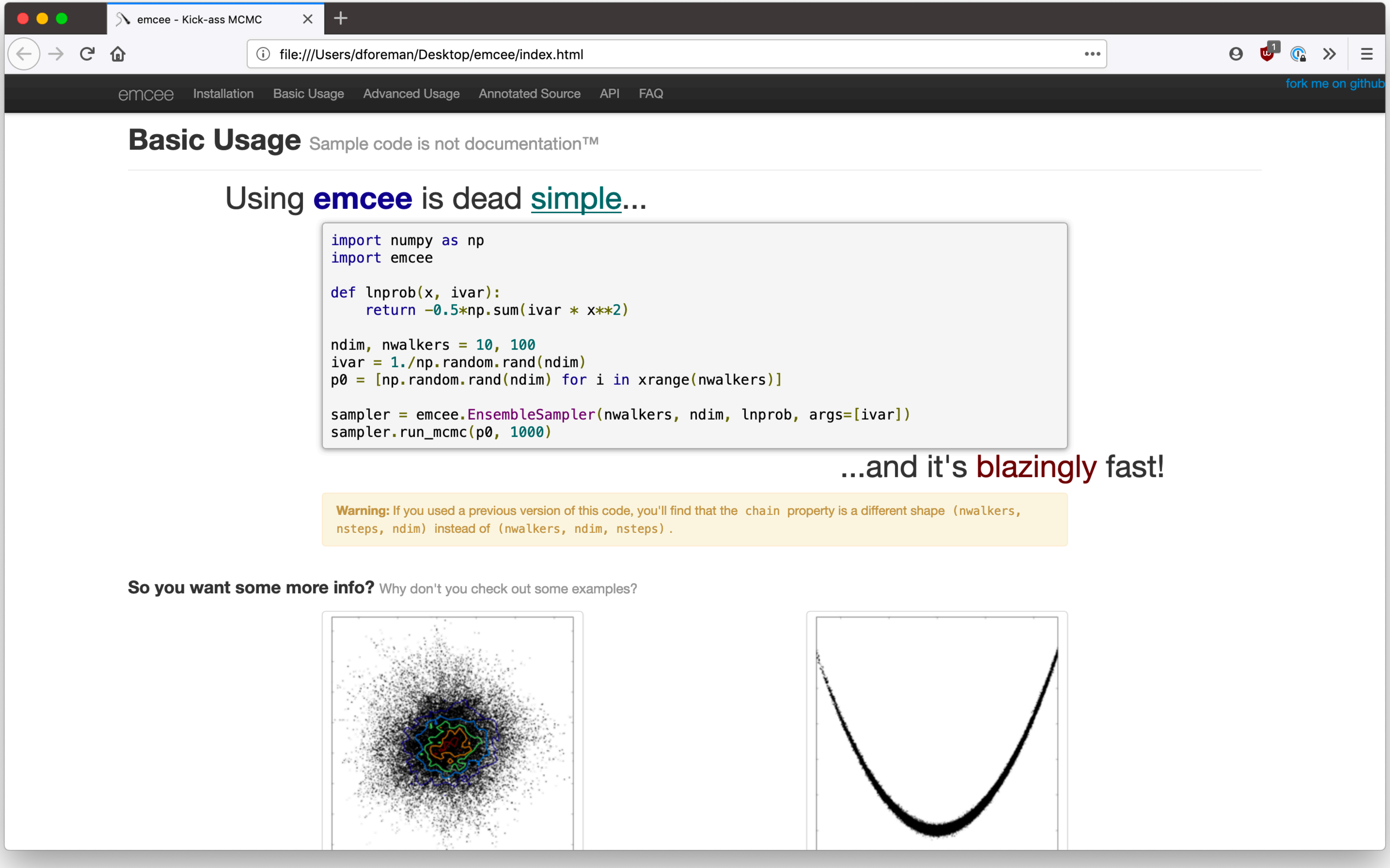
```
python -c 'import emcee; emcee.test()'
```

Bleeding Edge

For all the hackers in the house... *you know what to do!*

Fork **emcee** on GitHub

circa 2013



Basic Usage Sample code is not documentation™

Using **emcee** is dead simple...

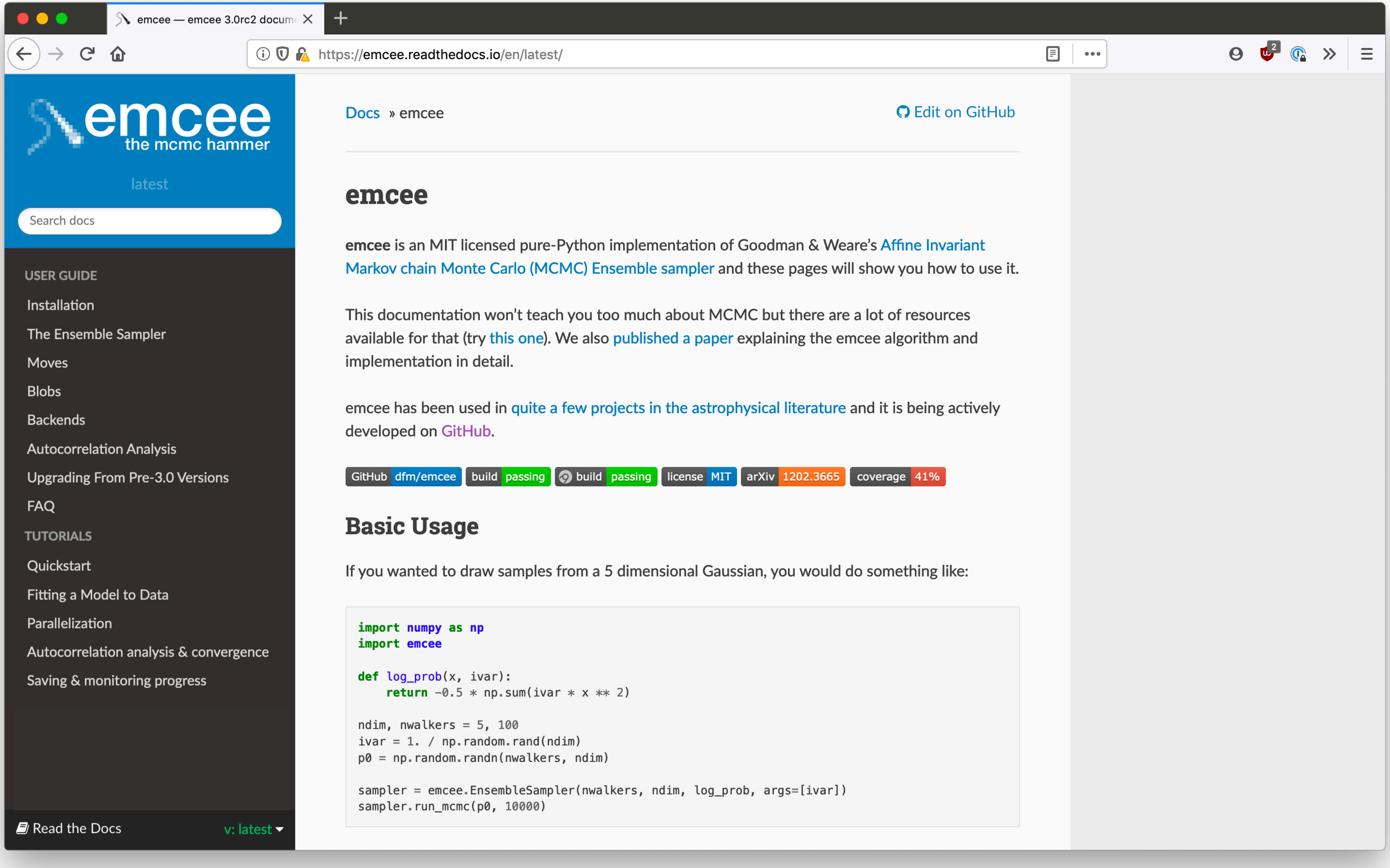
```
import numpy as np
import emcee

def lnprob(x, ivar):
    return -0.5*np.sum(ivar * x**2)

ndim, nwalkers = 10, 100
ivar = 1./np.random.rand(ndim)
p0 = [np.random.rand(ndim) for i in xrange(nwalkers)]

sampler = emcee.EnsembleSampler(nwalkers, ndim, lnprob, args=[ivar])
sampler.run_mcmc(p0, 1000)
```

...and it's blazingly fast!

Warning: If you used a previous version of this code, you'll find that the `chain` property is a different shape `(nwalkers, nsteps, ndim)` instead of `(nwalkers, ndim, nsteps)`.

So you want some more info? Why don't you check out some examples?

circa 2013



emcee

emcee is an MIT licensed pure-Python implementation of Goodman & Weare's Affine Invariant Markov chain Monte Carlo (MCMC) Ensemble sampler and these pages will show you how to use it.

This documentation won't teach you too much about MCMC but there are a lot of resources available for that (try this one). We also published a paper explaining the emcee algorithm and implementation in detail.

emcee has been used in quite a few projects in the astrophysical literature and it is being actively developed on GitHub.

Basic Usage

If you wanted to draw samples from a 5 dimensional Gaussian, you would do something like:

```
import numpy as np
import emcee

def log_prob(x, ivar):
    return -0.5 * np.sum(ivar * x ** 2)

ndim, nwalkers = 5, 100
ivar = 1. / np.random.rand(ndim)
p0 = np.random.randn(nwalkers, ndim)

sampler = emcee.EnsembleSampler(nwalkers, ndim, log_prob, args=[ivar])
sampler.run_mcmc(p0, 10000)
```

2

Lessons Learned

1

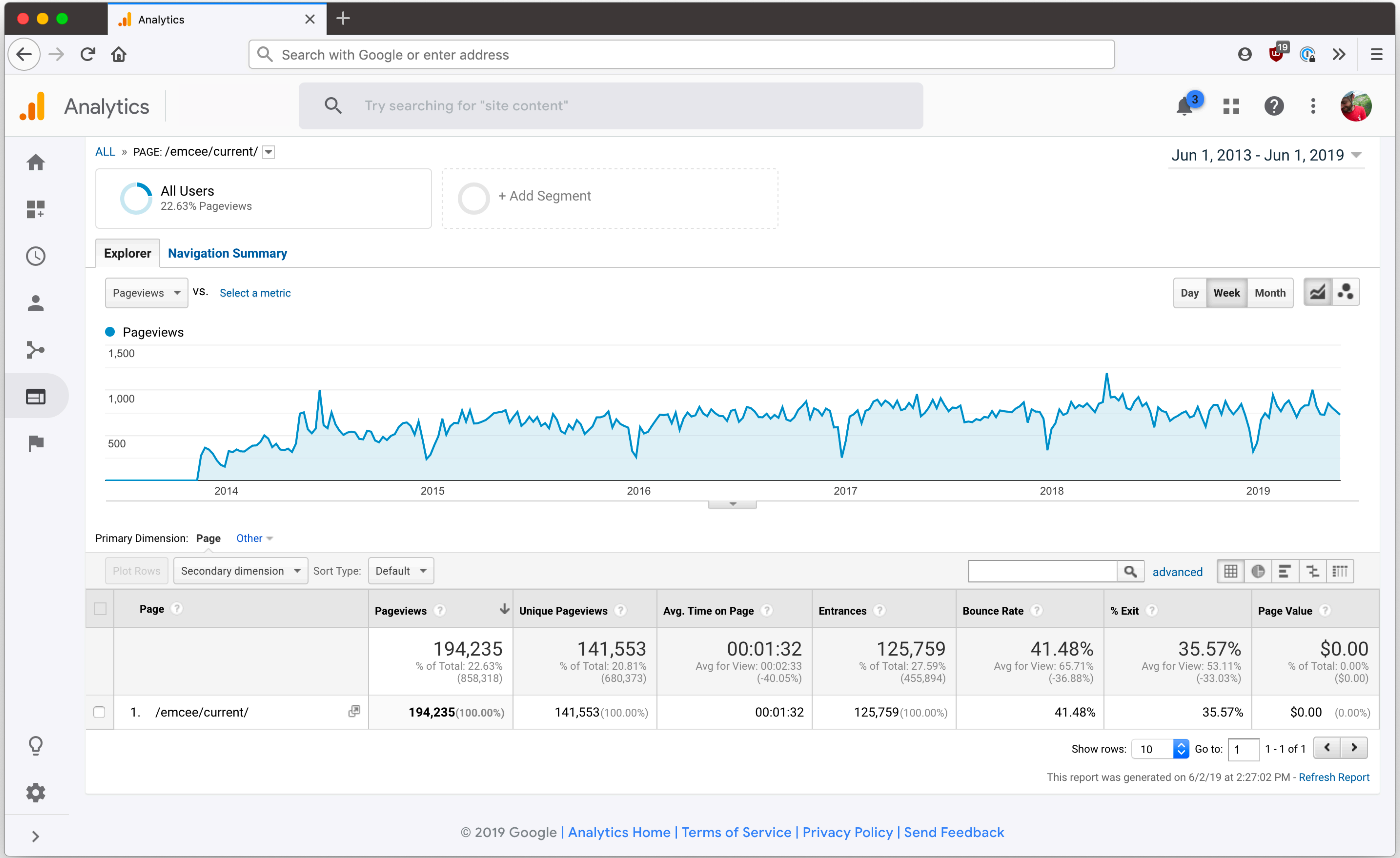
Releasing your code can be good for your career.

* Prior results do not guarantee a similar outcome.

2

Writing docs and tutorials is not a waste of time.

I use the documentation
that I've written every day.



	Page	Pageviews	Unique Pageviews	Avg. Time on Page	Entrances	Bounce Rate	% Exit	Page Value
		194,235 % of Total: 22.63% (858,318)	141,553 % of Total: 20.81% (680,373)	00:01:32 Avg for View: 00:02:33 (-40.05%)	125,759 % of Total: 27.59% (455,894)	41.48% Avg for View: 65.71% (-36.88%)	35.57% Avg for View: 53.11% (-33.03%)	$0.00 % of Total: 0.00% ($0.00)
1.	/emcee/current/	194,235(100.00%)	141,553(100.00%)	00:01:32	125,759(100.00%)	41.48%	35.57%	$0.00 (0.00%)

Show rows: 10 Go to: 1 1 - 1 of 1

This report was generated on 6/2/19 at 2:27:02 PM - Refresh Report

© 2019 Google | Analytics Home | Terms of Service | Privacy Policy | Send Feedback

Teaching is a good way to learn.

3

The extra email load isn't so bad.

I have been part of about **1700** email threads with the word "emcee".

That's only about

4.5 emails per week.

4

Beware of feature creep.

* Especially that first big pull request.

You will have to maintain the feature that you merge.

5

Keep it modular.

It's easier to write code
that does one thing well.

Package managers exist.

3 Ideas for a Successful Scientific Software Package

1

You should be the target audience.

2

Libraries, not scripts.

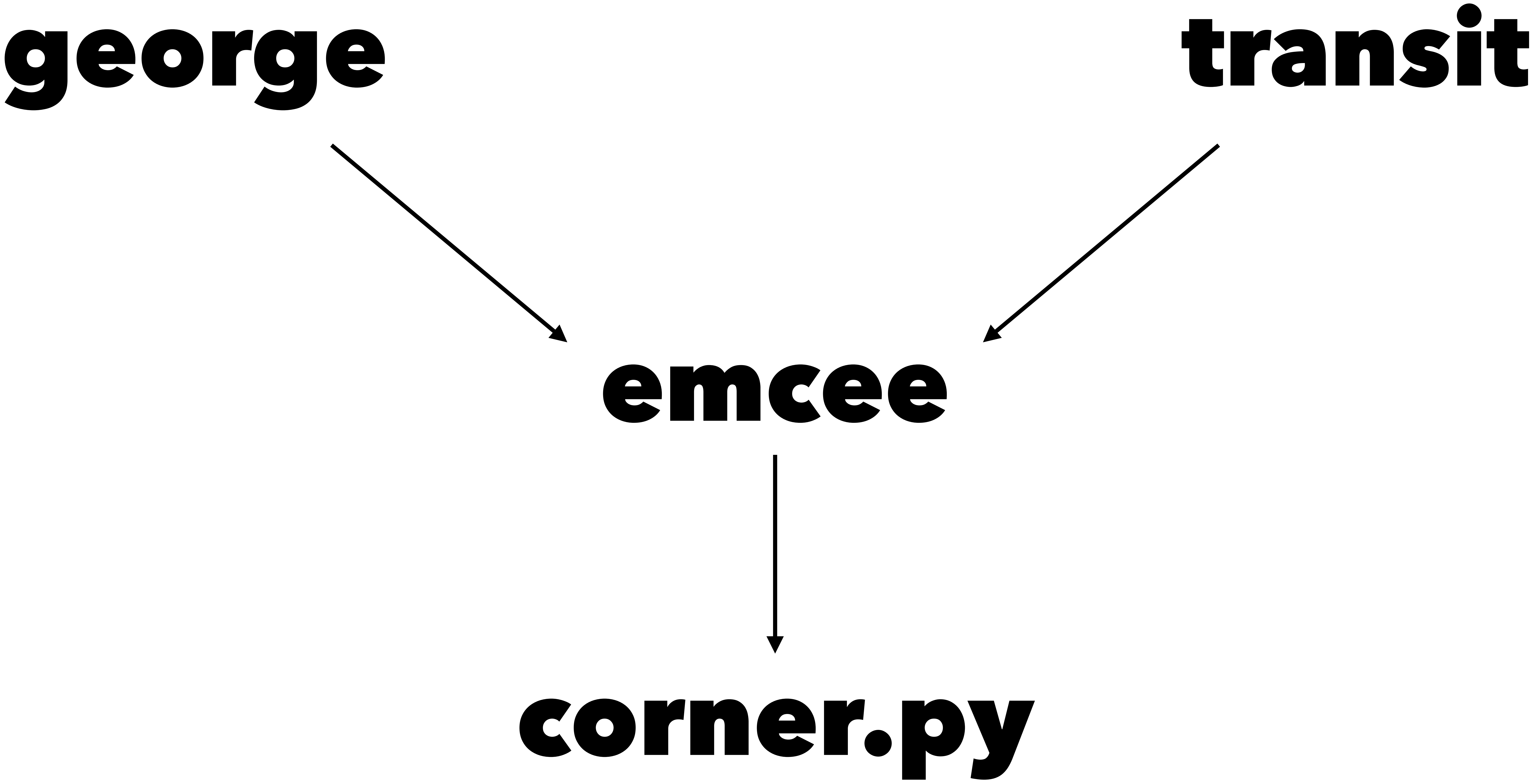
3

Tutorials, not (just) API docs.

4

Integrate with the **ecosystem**.

For example: fitting transiting exoplanet observations.



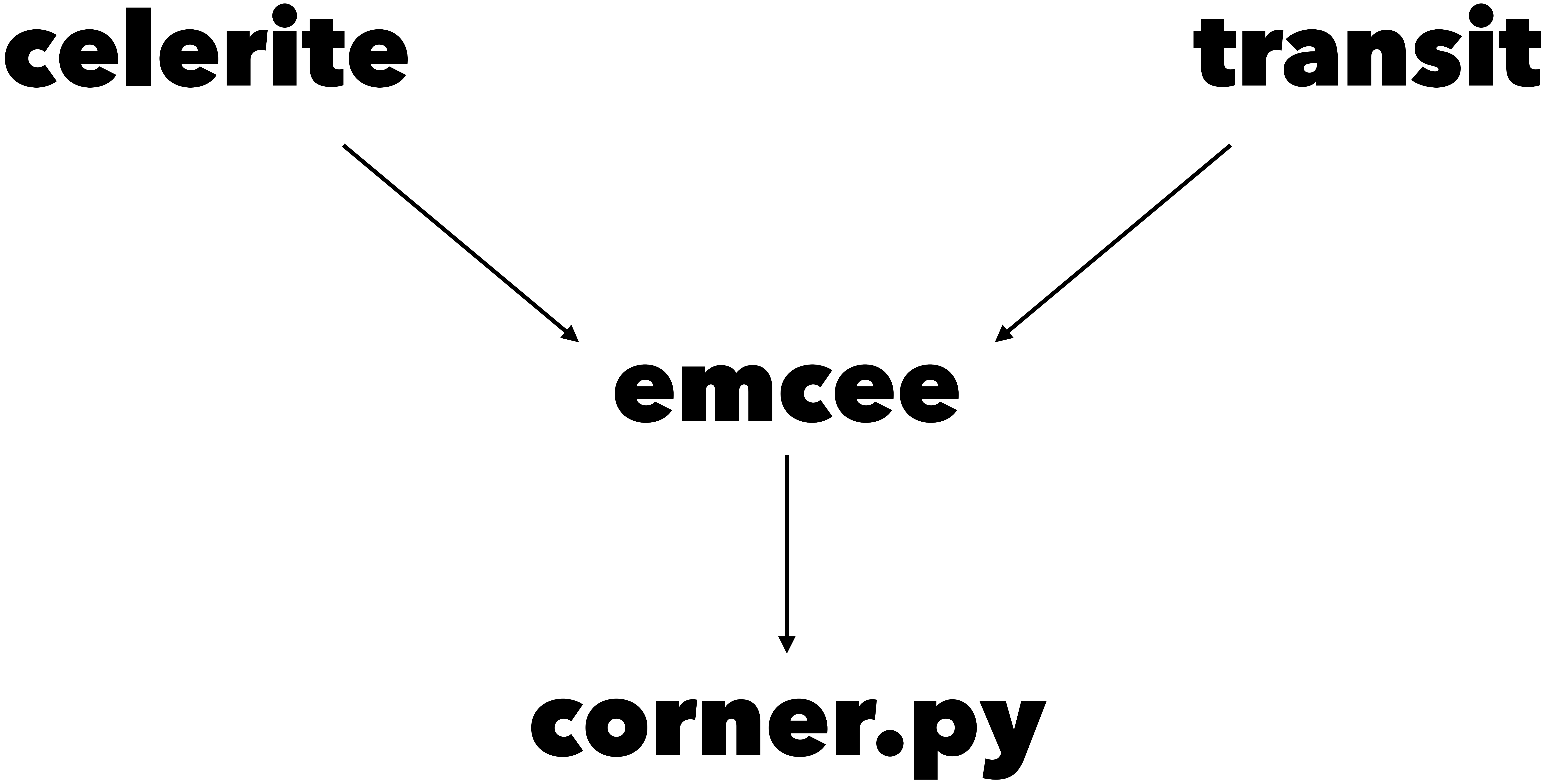
george

transit

emcee

corner.py

GitHub repositories; user: dfm



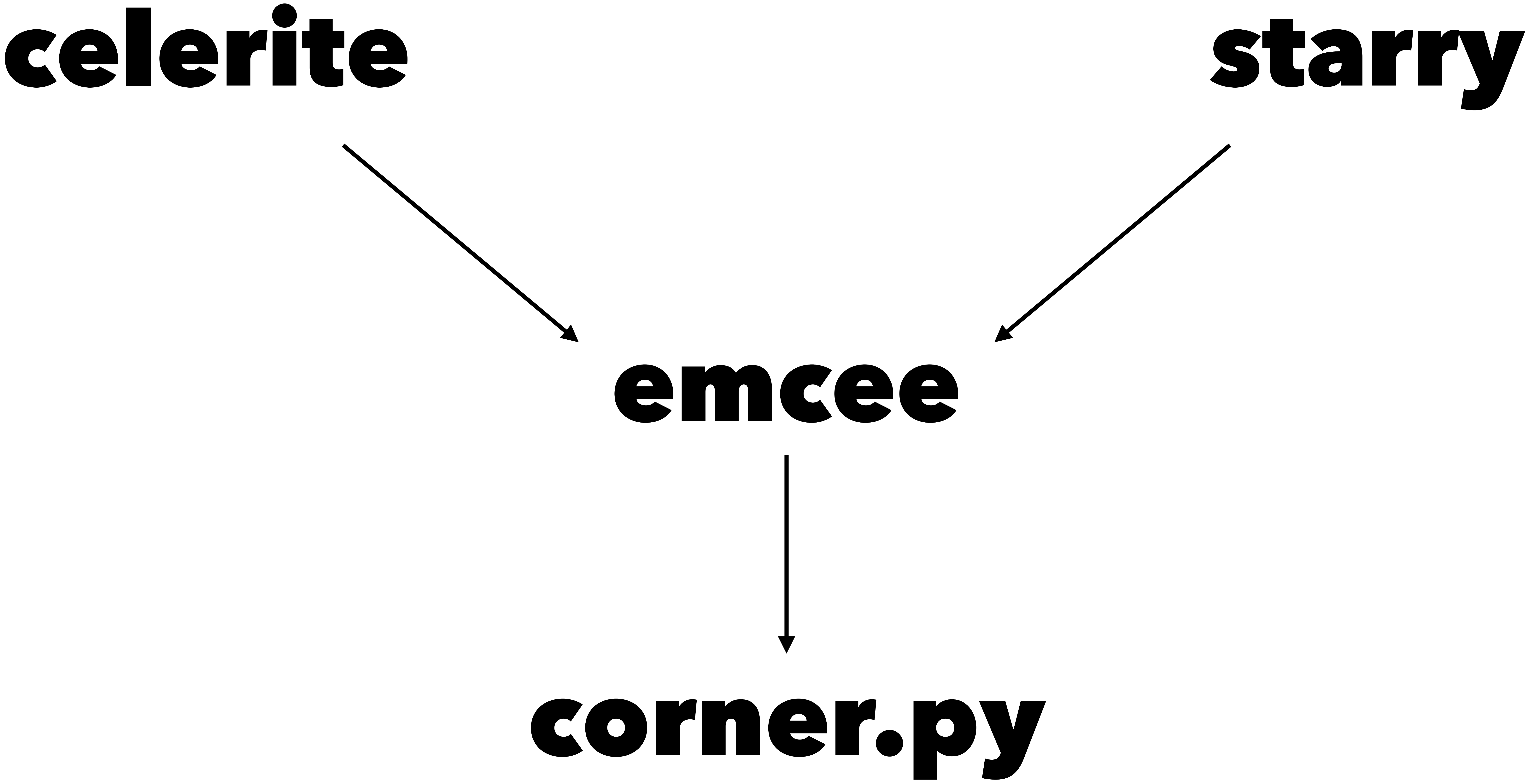
celerite

transit

emcee

corner.py

GitHub repositories; user: dfm



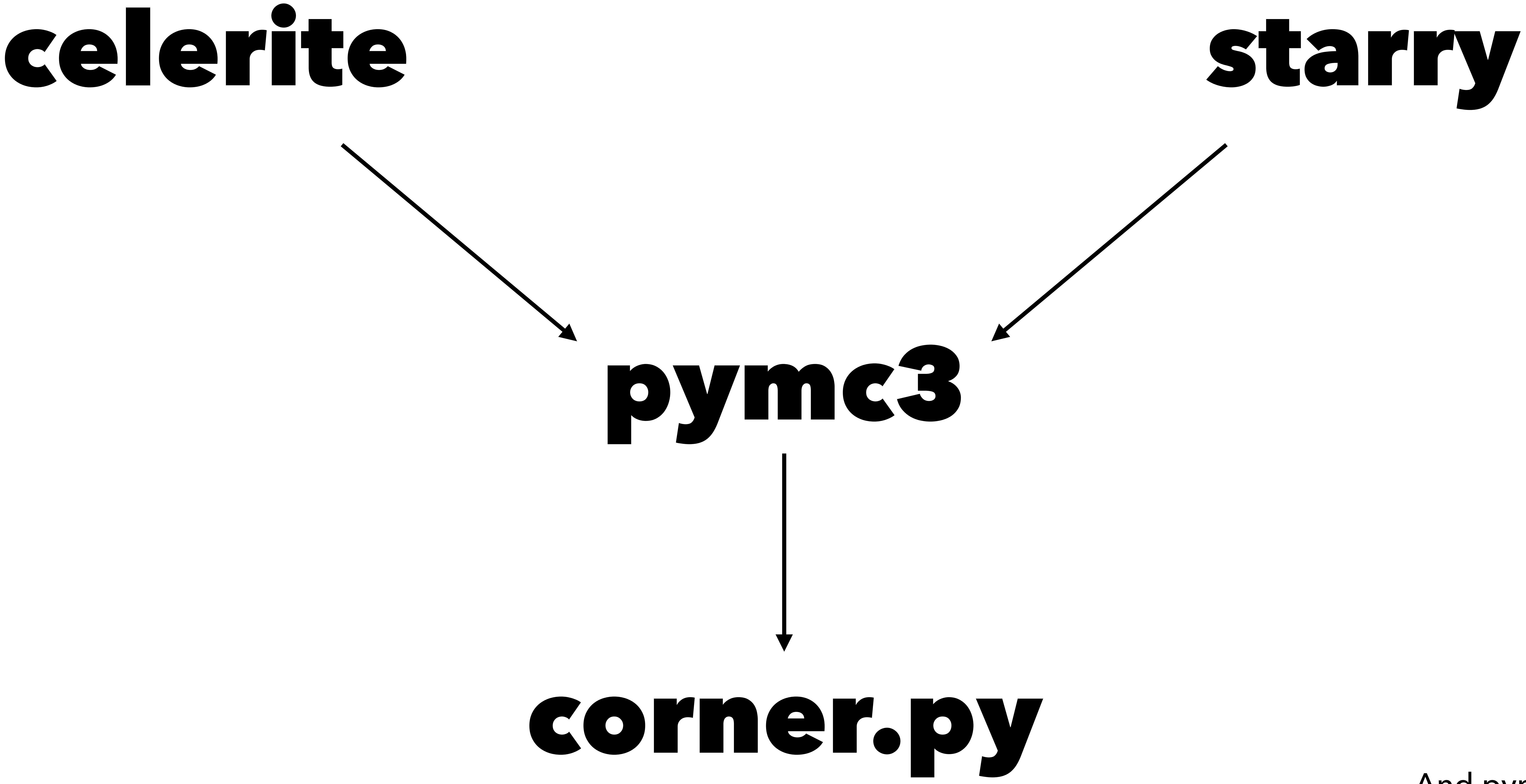
celerite

starry

emcee

corner.py

Except rodluger/starry by Rodrigo Luger
GitHub repositories; user: dfm



celerite

starry

pymc3

corner.py

And pymc-devs/pymc3
Except rodluger/starry by Rodrigo Luger
GitHub repositories; user: dfm

4

Open Questions

* A non-exhaustive list

1

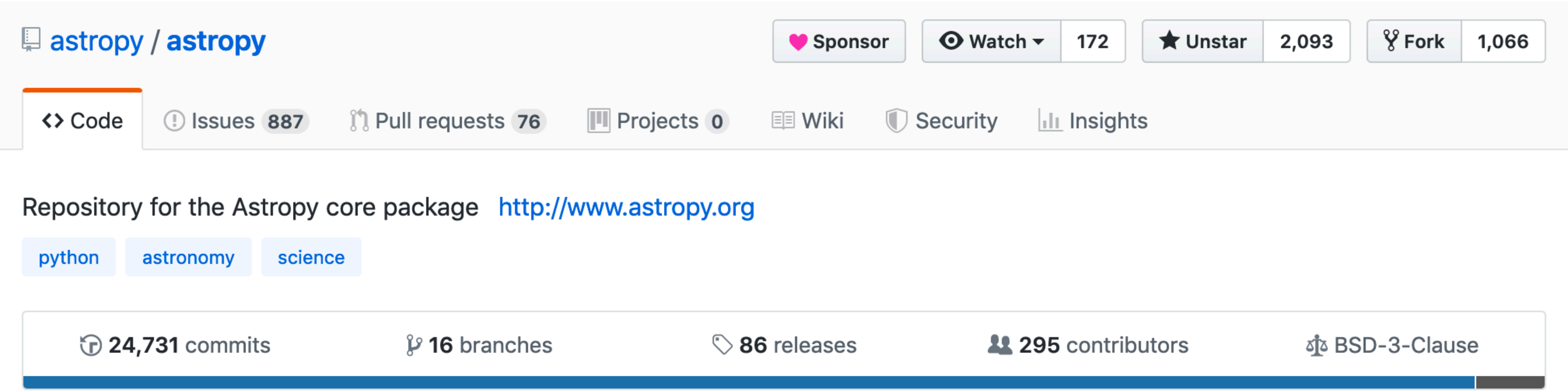
How do you build and maintain a sustainable developer community?

2

How do you balance community building and technical debt?

3

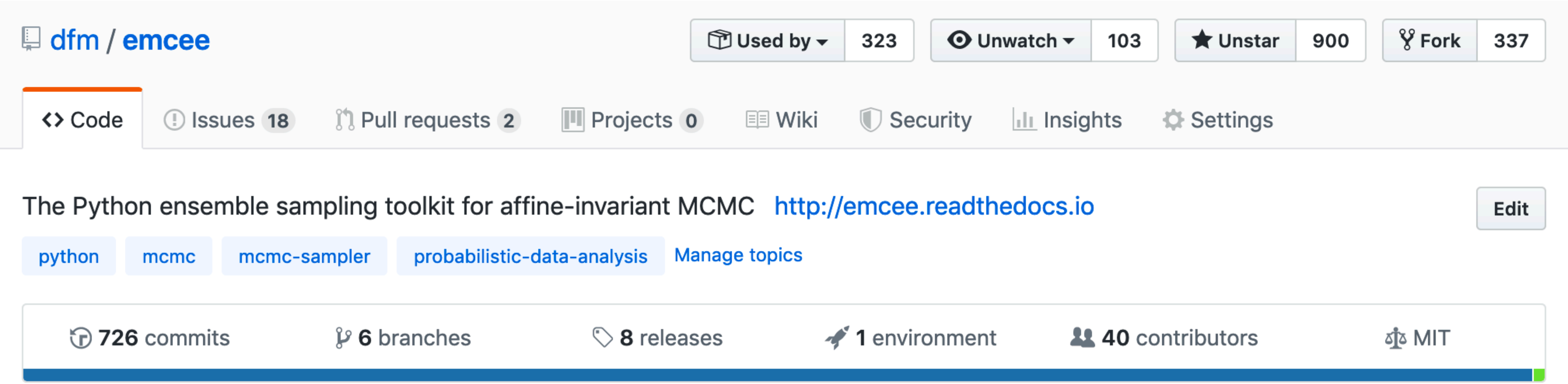
How do we give **credit** to developers of large projects?



astropy / astropy

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

24,731 commits · 16 branches · 86 releases · 295 contributors · BSD-3-Clause



dfm / emcee

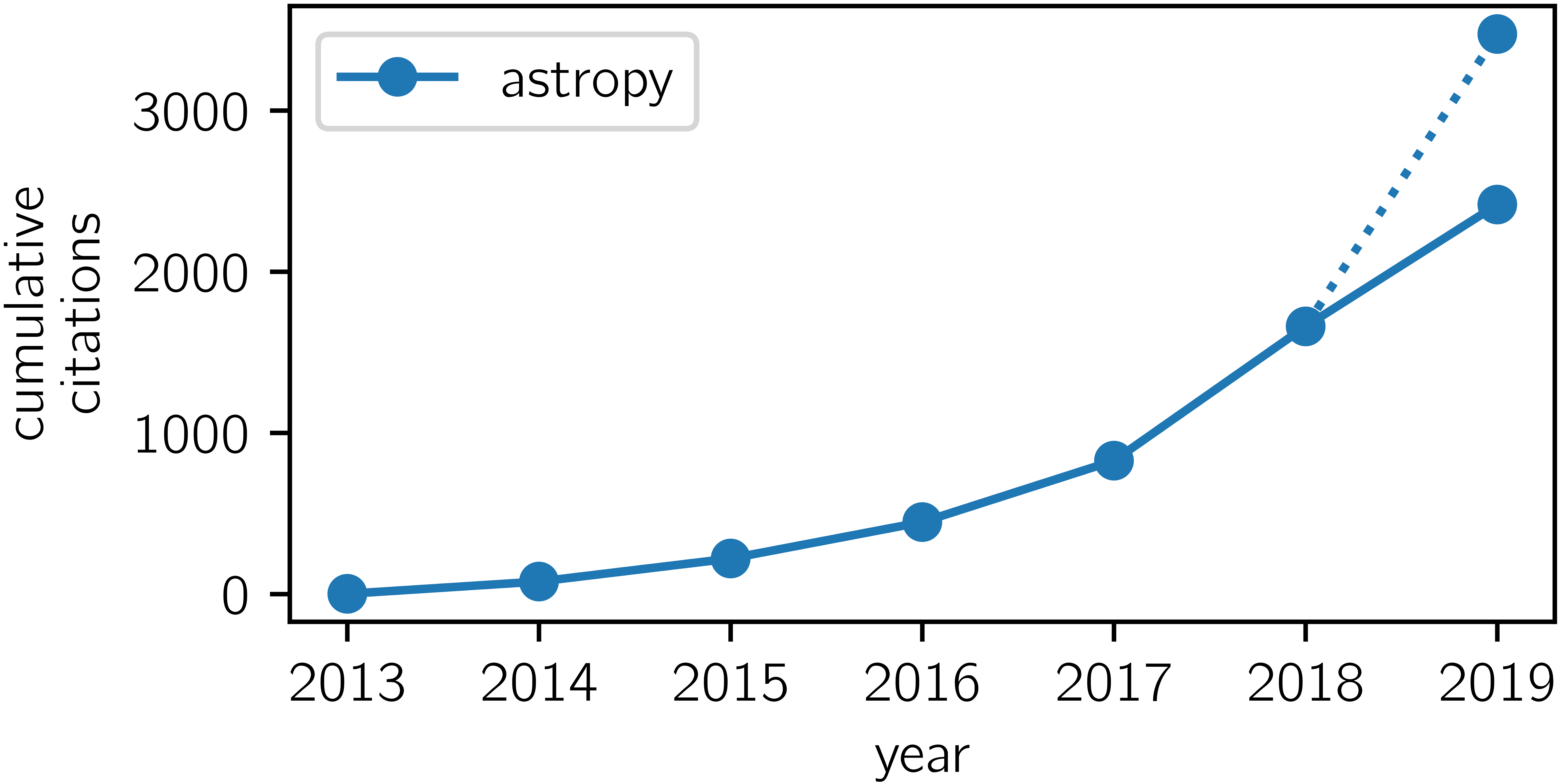
The Python ensemble sampling toolkit for affine-invariant MCMC http://emcee.readthedocs.io

726 commits · 6 branches · 8 releases · 1 environment · 40 contributors · MIT

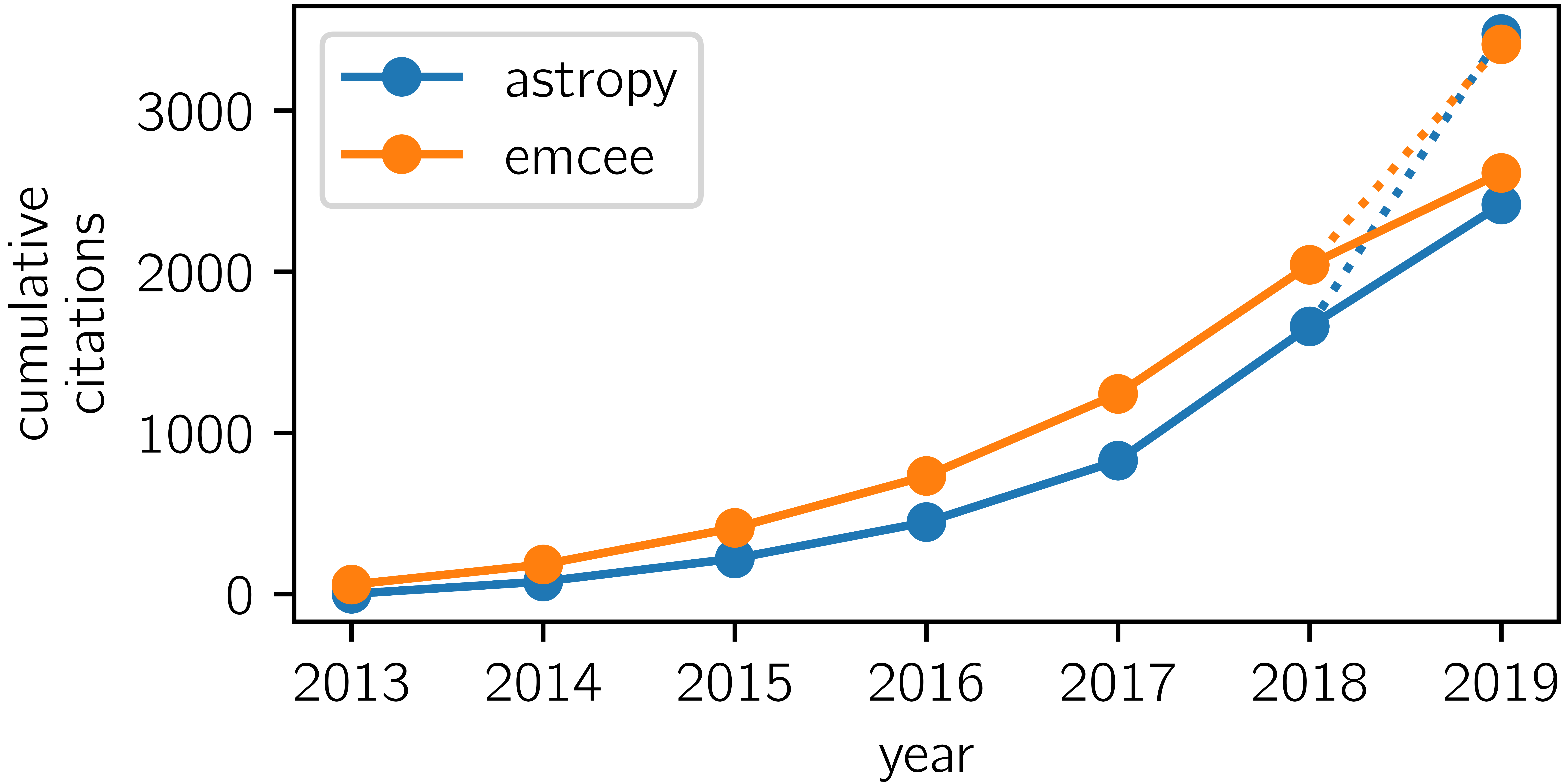
AstroPy is a much more successful open source project by all metrics.

AstroPy is a much more successful open source project by all metrics.

Except citation count.



data from: ui.adsabs.harvard.edu



data from: ui.adsabs.harvard.edu

Why?

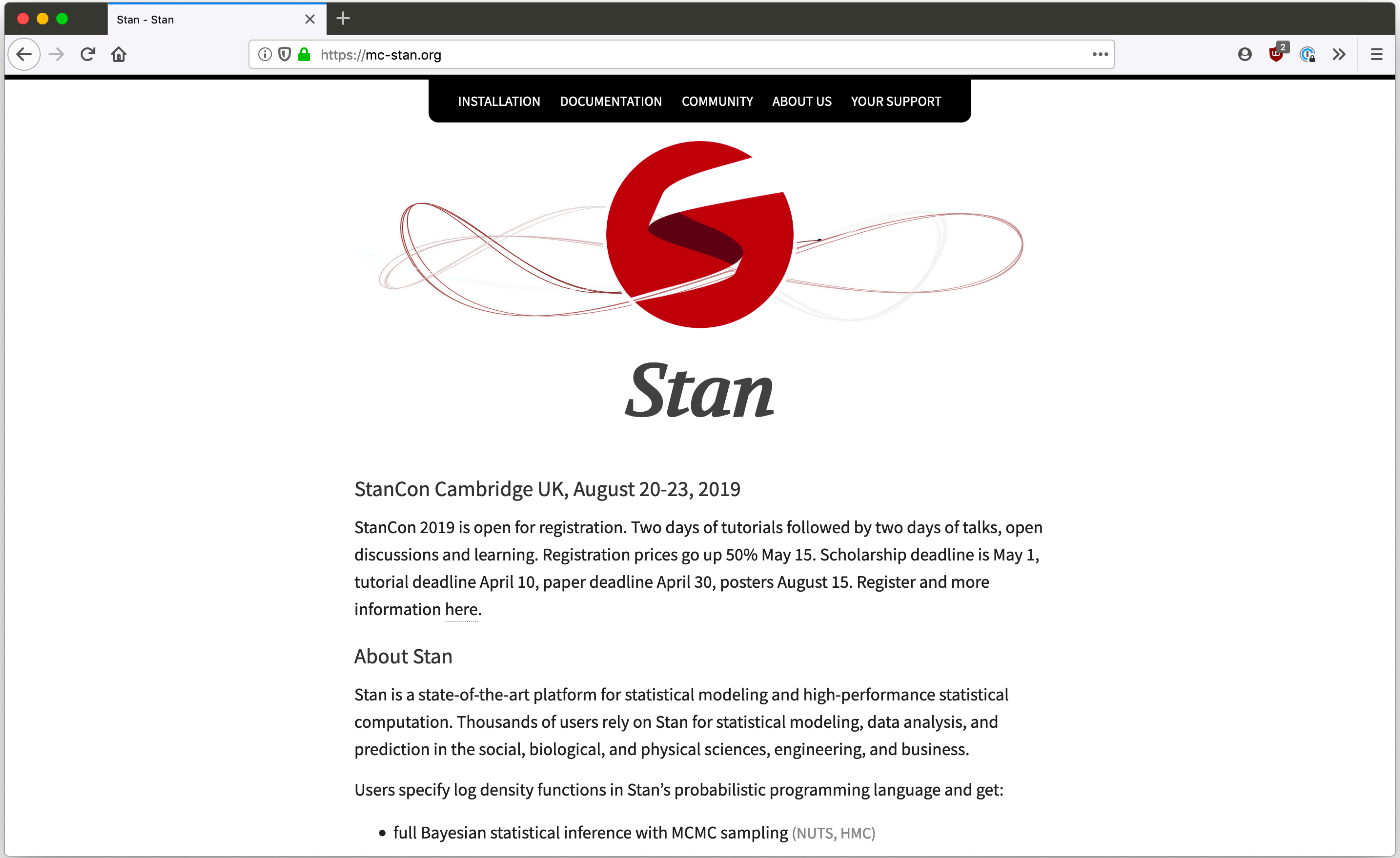
What should we do?

5

The Future

Will people still be using emcee in **10 years**?

I hope not!



StanCon Cambridge UK, August 20-23, 2019

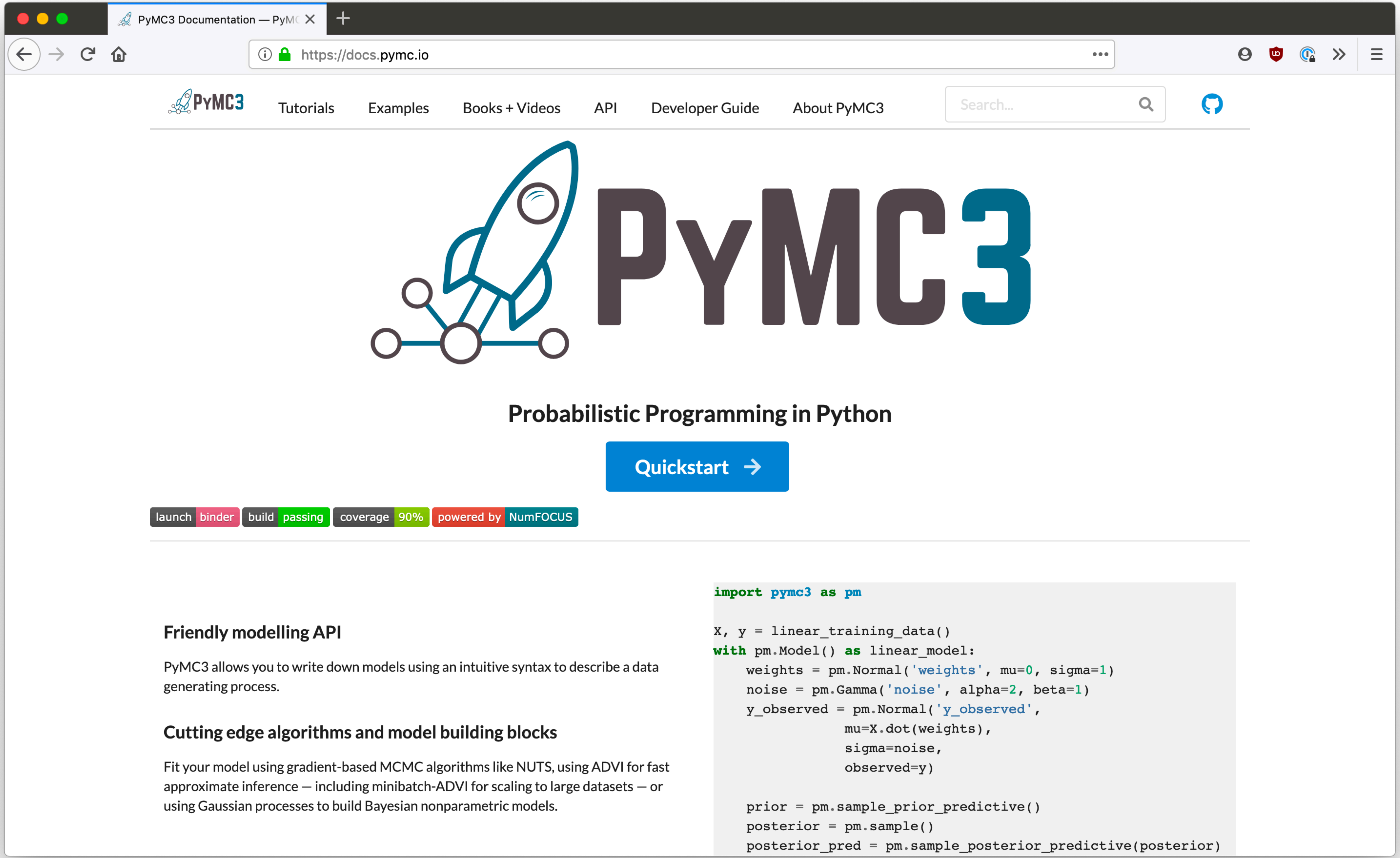
StanCon 2019 is open for registration. Two days of tutorials followed by two days of talks, open discussions and learning. Registration prices go up 50% May 15. Scholarship deadline is May 1, tutorial deadline April 10, paper deadline April 30, posters August 15. Register and more information here.

About Stan

Stan is a state-of-the-art platform for statistical modeling and high-performance statistical computation. Thousands of users rely on Stan for statistical modeling, data analysis, and prediction in the social, biological, and physical sciences, engineering, and business.

Users specify log density functions in Stan's probabilistic programming language and get:

- full Bayesian statistical inference with MCMC sampling (NUTS, HMC)



Probabilistic Programming in Python

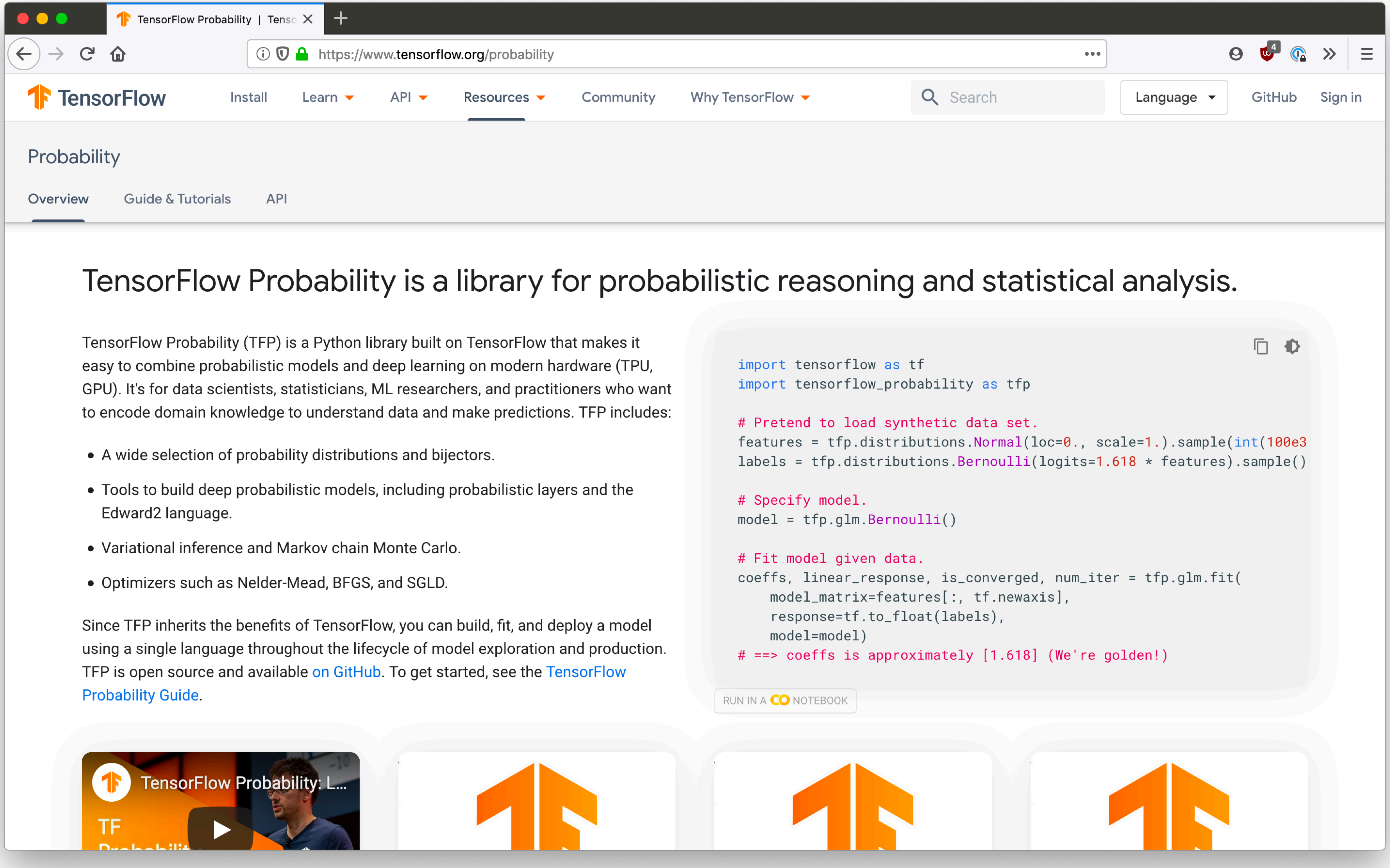
Friendly modelling API

PyMC3 allows you to write down models using an intuitive syntax to describe a data generating process.

Cutting edge algorithms and model building blocks

Fit your model using gradient-based MCMC algorithms like NUTS, using ADVI for fast approximate inference — including minibatch-ADVI for scaling to large datasets — or using Gaussian processes to build Bayesian nonparametric models.

```python
import pymc3 as pm

X, y = linear_training_data()
with pm.Model() as linear_model:
    weights = pm.Normal('weights', mu=0, sigma=1)
    noise = pm.Gamma('noise', alpha=2, beta=1)
    y_observed = pm.Normal('y_observed',
                mu=X.dot(weights),
                sigma=noise,
                observed=y)

    prior = pm.sample_prior_predictive()
    posterior = pm.sample()
    posterior_pred = pm.sample_posterior_predictive(posterior)
```



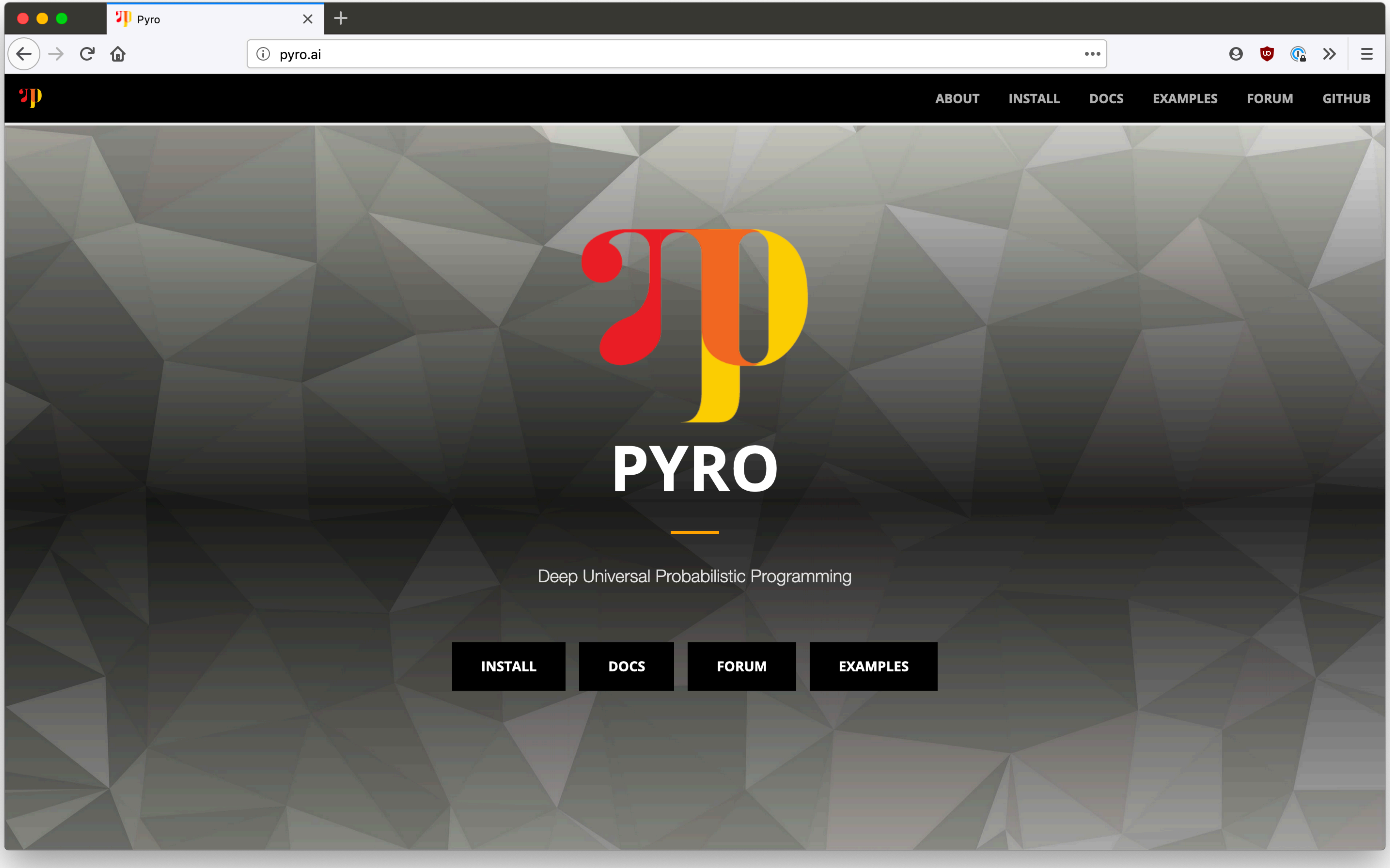
TensorFlow Probability is a library for probabilistic reasoning and statistical analysis.

TensorFlow Probability (TFP) is a Python library built on TensorFlow that makes it easy to combine probabilistic models and deep learning on modern hardware (TPU, GPU). It's for data scientists, statisticians, ML researchers, and practitioners who want to encode domain knowledge to understand data and make predictions. TFP includes:

- A wide selection of probability distributions and bijectors.
- Tools to build deep probabilistic models, including probabilistic layers and the Edward2 language.
- Variational inference and Markov chain Monte Carlo.
- Optimizers such as Nelder-Mead, BFGS, and SGLD.

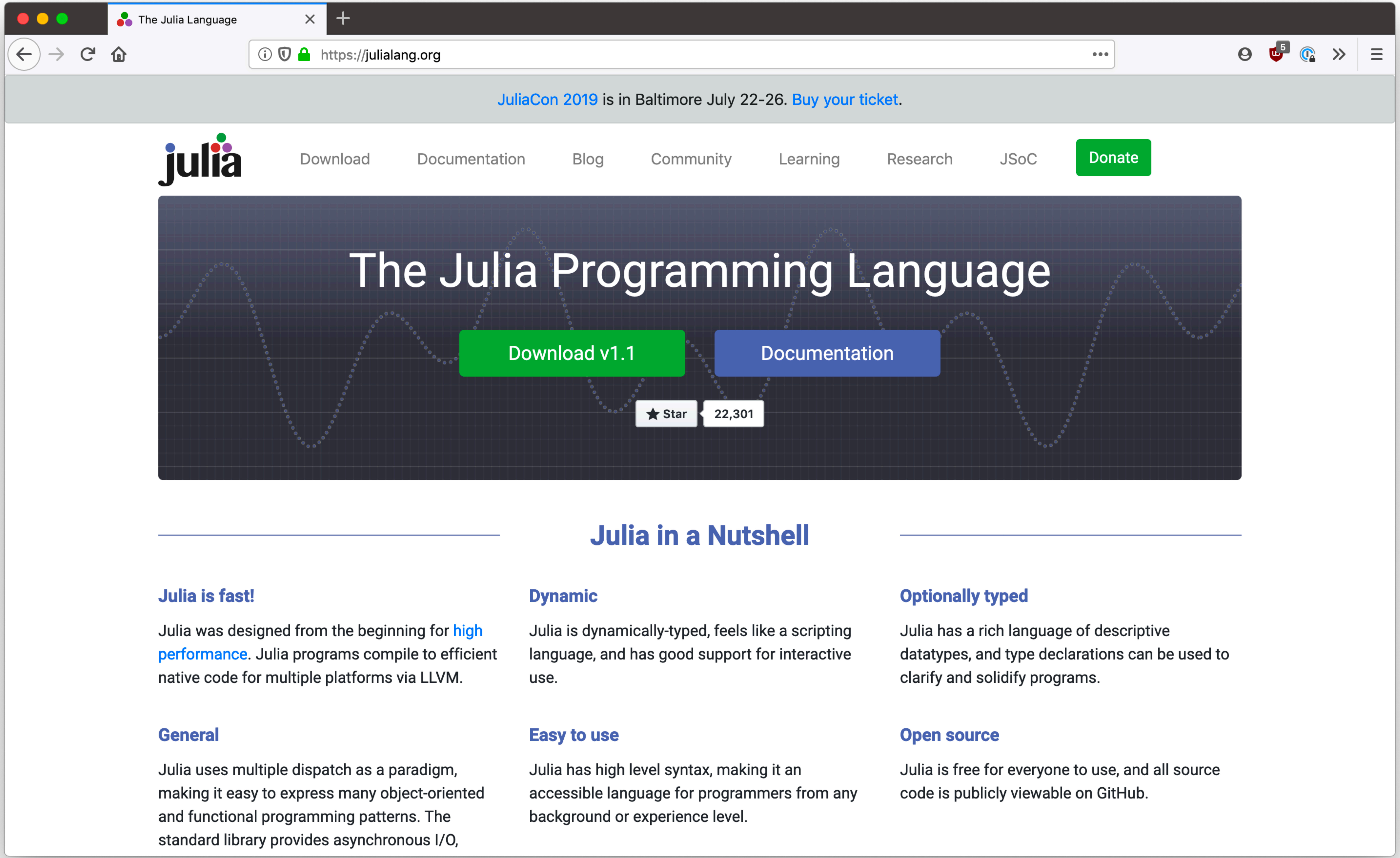
Since TFP inherits the benefits of TensorFlow, you can build, fit, and deploy a model using a single language throughout the lifecycle of model exploration and production. TFP is open source and available on GitHub. To get started, see the TensorFlow Probability Guide.

```
import tensorflow as tf
import tensorflow_probability as tfp

# Pretend to load synthetic data set.
features = tfp.distributions.Normal(loc=0., scale=1.).sample(int(100e3
labels = tfp.distributions.Bernoulli(logits=1.618 * features).sample()

# Specify model.
model = tfp.glm.Bernoulli()

# Fit model given data.
coeffs, linear_response, is_converged, num_iter = tfp.glm.fit(
    model_matrix=features[:, tf.newaxis],
    response=tf.to_float(labels),
    model=model)
# ==> coeffs is approximately [1.618] (We're golden!)
```



PYRO

Deep Universal Probabilistic Programming



JuliaCon 2019 is in Baltimore July 22-26. Buy your ticket.

The Julia Programming Language

Julia in a Nutshell

Julia is fast!

Julia was designed from the beginning for high performance. Julia programs compile to efficient native code for multiple platforms via LLVM.

Dynamic

Julia is dynamically-typed, feels like a scripting language, and has good support for interactive use.

Optionally typed

Julia has a rich language of descriptive datatypes, and type declarations can be used to clarify and solidify programs.

General

Julia uses multiple dispatch as a paradigm, making it easy to express many object-oriented and functional programming patterns. The standard library provides asynchronous I/O,

Easy to use

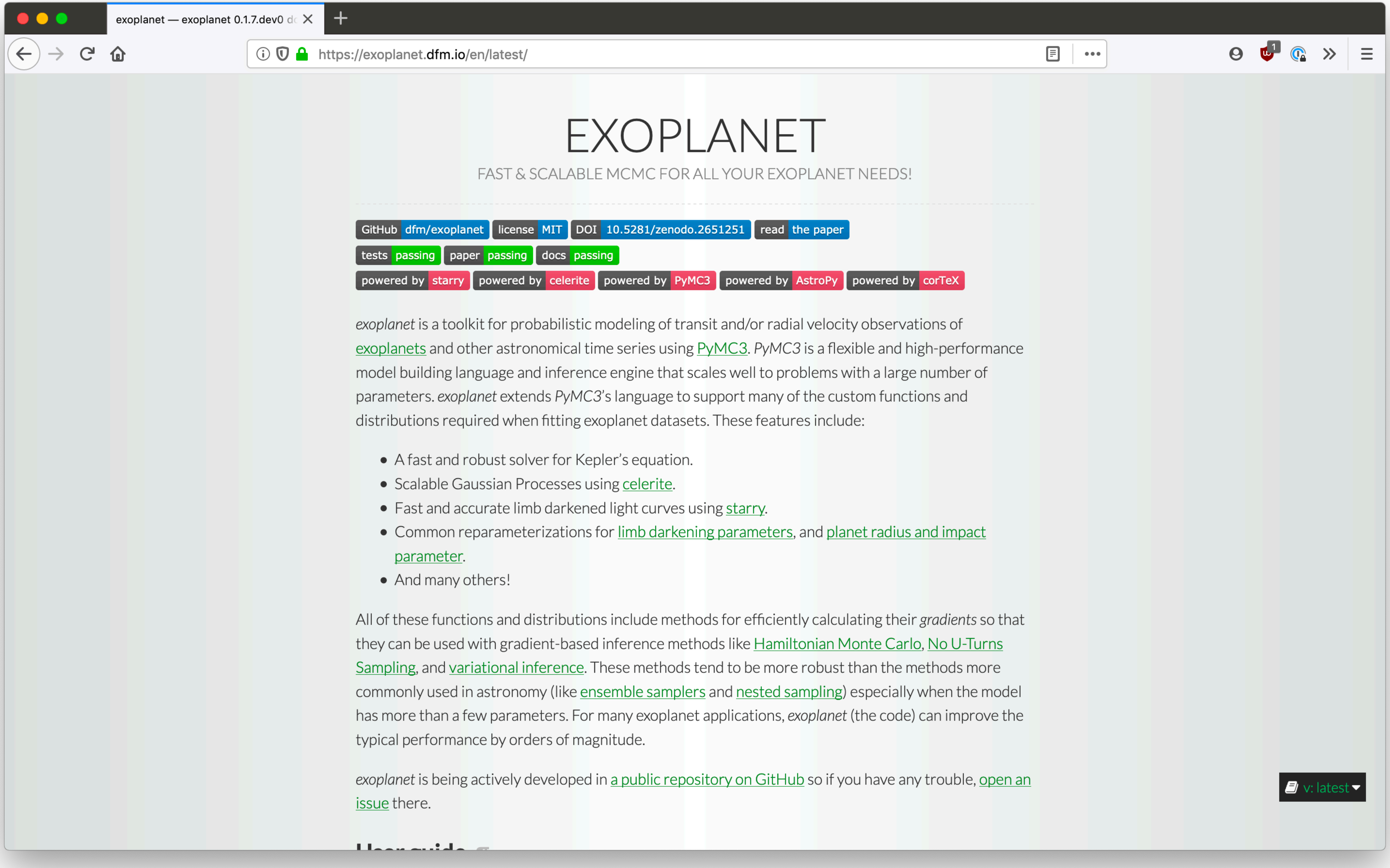
Julia has high level syntax, making it an accessible language for programmers from any background or experience level.

Open source

Julia is free for everyone to use, and all source code is publicly viewable on GitHub.

These all have strengths and weaknesses.

But these can have a steep **learning curve**.


exoplanet — exoplanet 0.1.7.dev0 d
https://exoplanet.dfm.io/en/latest/
v: latest

EXOPLANET

FAST & SCALABLE MCMC FOR ALL YOUR EXOPLANET NEEDS!

GitHub dfm/exoplanet | license MIT | DOI 10.5281/zenodo.2651251 | read the paper

tests passing | paper passing | docs passing

powered by starry | powered by celerite | powered by PyMC3 | powered by AstroPy | powered by corTeX

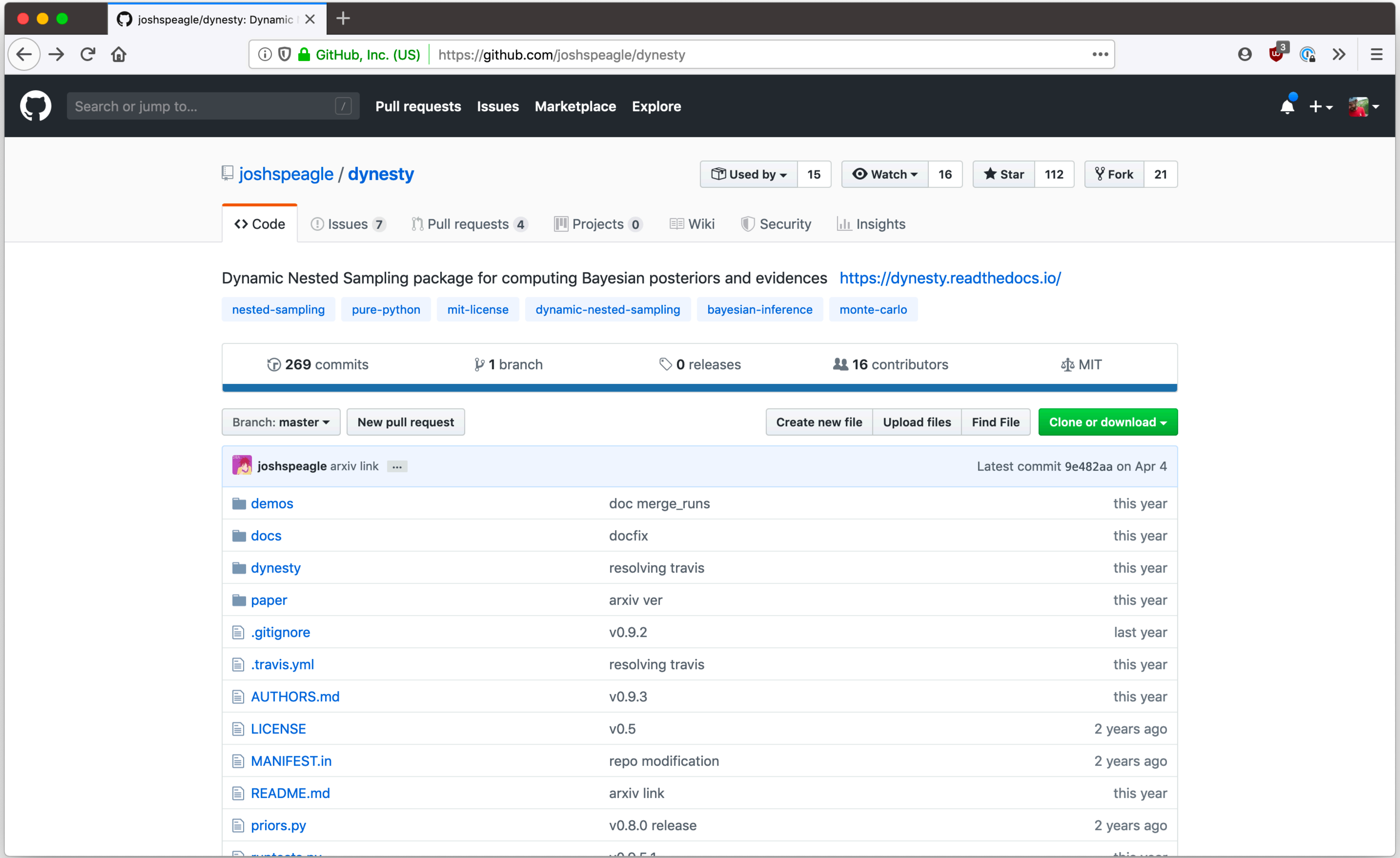
exoplanet is a toolkit for probabilistic modeling of transit and/or radial velocity observations of exoplanets and other astronomical time series using PyMC3. *PyMC3* is a flexible and high-performance model building language and inference engine that scales well to problems with a large number of parameters. *exoplanet* extends *PyMC3*'s language to support many of the custom functions and distributions required when fitting exoplanet datasets. These features include:

- A fast and robust solver for Kepler's equation.
- Scalable Gaussian Processes using celerite.
- Fast and accurate limb darkened light curves using starry.
- Common reparameterizations for limb darkening parameters, and planet radius and impact parameter.
- And many others!

All of these functions and distributions include methods for efficiently calculating their *gradients* so that they can be used with gradient-based inference methods like Hamiltonian Monte Carlo, No U-Turns Sampling, and variational inference. These methods tend to be more robust than the methods more commonly used in astronomy (like ensemble samplers and nested sampling) especially when the model has more than a few parameters. For many exoplanet applications, *exoplanet* (the code) can improve the typical performance by orders of magnitude.

exoplanet is being actively developed in a public repository on GitHub so if you have any trouble, open an issue there.

User guide



joshspeagle / dynesty

Dynamic Nested Sampling package for computing Bayesian posteriors and evidences https://dynesty.readthedocs.io/

nested-sampling pure-python mit-license dynamic-nested-sampling bayesian-inference monte-carlo

269 commits · 1 branch · 0 releases · 16 contributors · MIT

joshspeagle arxiv link … Latest commit 9e482aa on Apr 4

demos	doc merge_runs	this year
docs	docfix	this year
dynesty	resolving travis	this year
paper	arxiv ver	this year
.gitignore	v0.9.2	last year
.travis.yml	resolving travis	this year
AUTHORS.md	v0.9.3	this year
LICENSE	v0.5	2 years ago
MANIFEST.in	repo modification	2 years ago
README.md	arxiv link	this year
priors.py	v0.8.0 release	2 years ago

I plan on continuing to build tools in this **ecosystem** .

I want to learn how to continue to maintain this software and build a sustainable community.

6

Take Homes

Open source is good for business.

Tutorials are crucial.

Build libraries, not scripts.

Thanks!

Dan Foreman-Mackey

CCA@Flatiron // dfm.io // @exoplaneteer // github.com/dfm