Proposal Title

Faster Random Variate Sampling from SciPy Statistical Distributions

Two Sentence Summary of Proposal

Recently, `scipy.stats.sampling` added classes that efficiently sample random variates from arbitrary statistical distributions. We propose to use these features to speed up random variate sampling for many of SciPy's built-in statistical distributions.

Description of Proposal (<750 words, < 4500 chars)

Random variates (numbers) are needed in many domains of science and engineering for simulating physical processes and executing stochastic algorithms. Without specialized hardware, it is challenging for digital computers to generate truly unpredictable random numbers, so in practice, pseudorandom number generator (PRNG) algorithms return sequences of numbers that satisfy many properties of randomness. Most PRNG algorithms generate uniformly distributed numbers such that all values within a range are equally likely, but users frequently need numbers distributed according to other statistical distributions with prescribed, nonuniform probabilities of sampling each value. Consequently, `scipy.stats` includes over 100 statistical distributions objects, each with an `rvs` method for Random Variate Sampling from that distribution. But for many of these distributions, random variate sampling is slow, relying on a simple but inefficient default algorithm for converting the output of PRNGs into numbers drawn from the prescribed distribution, inverse transform sampling.

Recently, `scipy.stats.sampling` added several other general algorithms for sampling random variates from arbitrary statistical distributions [1]. After a modest set-up time that is independent of the required number of random variates, most of these algorithms are orders of magnitude faster at generating random variates than SciPy's default algorithm. However, there are currently two substantial impediments to their adoption: users must look beyond the familiar statistical distribution interface to discover these features, and users need to choose which of several algorithms to employ.

We propose to remove these obstacles by using the appropriate algorithm by default in each distribution's `rvs` method, dramatically increasing performance without any user effort. Specifically, we will override the `rvs` method of "frozen" distributions: when a user instantiates the frozen distribution with scalar arguments, the random variate sampling method will be initialized, and subsequent calls to the frozen distribution's `rvs` method will execute very quickly. We will also demonstrate the improvements in random sampling performance by adding tests to SciPy's benchmarking suite, and we will emphasize these improved capabilities in our statistical distribution API reference and tutorial.

References:

[1] Tirth Patel and Christoph Baumgarten. "ENH: stats: Integrate library UNU.RAN in scipy.stats · Issue #14215 · scipy/scipy". https://github.com/scipy/scipy/pull/14215

[2] The SciPy Commnity. "Detailed SciPy Roadmap". https://scipy.github.io/devdocs/dev/roadmap-detailed.html

[3] Matt Haberland. "ENH: stats: univariate distribution meta-issue · Issue #15928 · scipy/scipy". https://github.com/scipy/scipy/issues/15928

Benefit to Project/Community (<400 words, < 2500 chars)

Please explain the benefit of this proposal including:

- *Impact to the project*
- *Impact to the scientific ecosystem*
- *Impact to the community*

This work will impact the SciPy project by addressing one of the items on its roadmap: “Speed up random variate sampling (method `rvs`) of distributions, leveraging `scipy.stats.sampling` where appropriate” [2]. By completing this roadmap item now, all work that would have waited for its completion (e.g. future work by Christoph Baumgarten and Tirth Patel) will happen sooner.

This work will benefit the scientific Python ecosystem by seamlessly improving the performance of dependent libraries that rely on SciPy’s frozen distributions for random variate sampling. It will also encourage users to adopt the frozen distribution API for using SciPy distributions, which has many other advantages and paves the way for a much-needed overhaul of SciPy’s distribution infrastructure [3].

This work will strengthen the community by increasing the time that a core developer can allocate to reviewing the pull requests of others, which is essential to mentoring new contributors and retaining other repeat contributors.

Amount Requested

\$$x + w$

Brief Budget Justification - How will the money be spent?

\$$x$ (\$$y$ per hour $\times$ z hours) will compensate work by Christoph Baumgarten and is sufficient to fund this proposal.

Tirth Patel has agreed to review Christoph’s pull requests on a voluntary basis. Consequently, we request an additional \$$w$ to support Christoph Baumgarten for other `scipy.stats` maintenance and review of `scipy.stats` pull requests to free Tirth’s time for review of this work.

Timeline of Deliverables

July 2022 – decision notification

August 2022 – Prepare statistical distribution benchmarks to track the improvement in random variate sampling

September 2022 – edit SciPy stats distribution infrastructure (`scipy.stats.rv_continuous`) to facilitate the override of continuous univariate distribution `rvs` methods, and override `rvs` method of select distributions to demonstrate its efficacy

October-December 2022 – (individually) override the `rvs` method of remaining ~100 continuous univariate distributions as necessary

January 2022 – edit SciPy stats distribution infrastructure (`scipy.stats.rv_discrete`) to facilitate override of discrete univariate distribution `rvs` methods, and override `rvs` method of ~15 discrete univariate distributions

February 2022 – submit final report

Has someone been identified to carry out the work in the proposal?

Yes: Christoph Baumgarten and Tirth Patel. Christoph and Tirth are ideally suited to the work because they are both regular maintainers of `scipy.stats` with commit rights, and they are the original authors of `scipy.stats.sampling`.

Please list the name and email address of a project leader(s) who has approved this proposal. *

Matt Haberland <matt.haberland@gmail.com>

I agree to submit a grant report-back if my proposal is selected for funding.

I agree.