Improving NumPy for Better Data Science Progress Report (III)

Stéfan J. van der Walt, Sebastian Berg, Warren Weckesser, Ross Barnowski, Jarrod Millman, Fernando Pérez

Berkeley Institute for Data Science (BIDS)
University of California, Berkeley

October 2019–October 2020

Alfred P. Sloan Foundation Grant G-2017-9960
https://sloan.org/grant-detail/8222

Gordon and Betty Moore Foundation Grant GBMF5447
https://www.moore.org/grant-detail?grantId=GBMF5447

Principal Investigator: Stéfan J. van der Walt

These grants support improvements to NumPy, a library for the Python scientific programming language. NumPy provides support for numerical programming in Python, which is critical for data-driven research.

Contents

Introduction

This report covers November 2019 through October 2020. We report on the completion of aims S1, S2, T1, T2, and T4; and discuss S3 and T4, along with plans for continued work.

Social Aims

S1	Improve Community Engagement
S2	Grow Core Team, Add Contributors
S3	Diversify Contributors

Technical Aims

T1	More Flexible & Sustainable Code
T2	Frequent & Consistent Releases
T3	Improve Data Type System
T4	New Array Protocol

Personnel

We appointed a new developer, Ross Barnowski, in December 2019. Ross has a background in developing educational material centered on reproducible and collaborative science with open-source software. After two years at BIDS, Matti Picus's appointment ended April 2020. He subsequently joined QuanSight Labs and remains active in the NumPy community.

Aims: Technical

We have completed T1, T2, and T4, and are making steady progress on T3.

T3: New Data Types

Work is underway on a new, extensible data type system for NumPy. In 2019, an early prototype[1] was developed which served as a springboard for further design discussions at the NumPy developer sprint hosted at BIDS in November 2019. The sprint resulted in simplified designs for array coercion and improved universal function (ufunc) dispatching[2].

[1] https://github.com/seberg/numpy/tree/dtypemeta

[2] https://github.com/numpy/archive/blob/master/sprints/2019-11-22.md#dtypes

The most significant challenge of the dtype system overhaul is to generate consensus within the NumPy community and among stakeholders in the wider scientific Python ecosystem. To aid in the conversation, four new NumPy Enhancement Proposals (NEPs) were written:

- NEP40: this informational NEP describes the current state of the dtype system and its shortcomings, and lays out the motivation for the new dtype system.
- NEP41 describes the architecture of the new design, how it maps to existing built-in data types, and how it allows users to design their own custom dtypes.
- NEP42 deals with implementation details of the new datatype classes and their hierarchy. In particular, it discusses implementation decisions relevant to array creation/coercion and casting.

- NEP43 plans how to restructure NumPy's universal functions (ufuncs) to incorporate the new dtype design.

This series of NEPs was published sequentially over the past year, and each has undergone independent review. Given the wide impact of the proposed changes, reviews from maintainers of other libraries that use or extend NumPy's dtype system—such as pandas and astropy—were actively sought out.[3]

[3] https://github.com/numpy/numpy/pull/15507

A new data-type architecture, the foundation of the dtype system overhaul, has been implemented and incorporated into NumPy[4]; so has the array creation and coercion functionality[5]. The implementation of the new casting system is underway[6].

[4] https://github.com/numpy/numpy/pull/15508

[5] https://github.com/numpy/numpy/pull/16200

[6] https://github.com/numpy/numpy/pull/17401

T4: Array Protocols

The `__array_function__` protocol had been enabled by default since NumPy version 1.17 and has seen adoption by popular array libraries such as dask[7] and cupy[8]. Feedback from these early adopters made us aware of use-cases not yet covered. Protocols were one of the main discussion topics at the NumPy developer sprint held at BIDS in November 2019[9], resulting in a NEP for a new `__array_module__` protocol[10]. A reference implementation[11] was developed to test the proposed approach. The new protocol has been adopted provisionally by JAX[12] for testing. The continuing development of protocols strengthens interoperability between NumPy and other array computation libraries.

[7] https://github.com/dask/dask/pull/4567

[8] https://github.com/cupy/cupy/pull/1650

[9] https://github.com/numpy/archive/blob/master/sprints/2019-11-22.md#duck-array-nep-discussion-11-12

[10] https://numpy.org/neps/nep-0037-array-module.html

[11] https://github.com/seberg/numpy-dispatch

[12] https://github.com/google/jax/pull/4076

Universal Intrinsics

Matti Picus, together with members of the community, wrote NEP 38[13], which outlines a framework for utilizing SIMD optimized instructions. An implementation was written by Sayed Adel[14,15] with guidance from the team and community.

[13] https://numpy.org/neps/nep-0038-SIMD-optimizations.html

[14] https://github.com/numpy/numpy/pull/13421

[15] https://github.com/numpy/numpy/pull/13516

New Text Reader

NumPy is able to read CSV files and similar text inputs, but much more slowly than, e.g., pandas. We wanted to improve performance, but in such a way that the work would be useful outside of NumPy too.

We therefore implemented a library in C[16] that can be adapted by other libraries who want to implement similar text reading functionality. The new reader is tested against the NumPy test suite to ensure full compatibility with existing functions, and matches the

[16] https://github.com/WarrenWeckesser/npreadtext

performance of pandas. The new reader is a drop in replacement for `numpy.loadtxt`, and our aim is for it to become the default.

NumPy Financial

Financial functions were removed[17] from NumPy[18] in v1.20 and have been re-packaged as `numpy-financial`.[19] This is likely the first time that NumPy's public-facing API has been reduced significantly.

[17] https://github.com/numpy/numpy/pull/17067

[18] https://numpy.org/neps/nep-0032-remove-financial-functions.html

[19] https://numpy.org/numpy-financial/

Polynomial Package

NumPy has multiple components for dealing with polynomials: the older polynomial *module* defined in `numpy.lib.polynomial` as well as the newer `numpy.polynomial` package. Users found it difficult to navigate the overlap in functionality, and in response we refactored, documented, and improved polynomial functionality:

- Added support for richer string representations of polynomial expressions with multiple formats in terminal environments[20].
- Added a *symbol* attribute to enhance expressing and operating on 1D polynomials[21].
- Improved the class-based interface for more robust input validation and error handling[22].
- Consolidated[23] and improved[24] documentation, providing guidance on how to transition from the old to the new system.

[20] https://github.com/numpy/numpy/pull/15666

[21] https://github.com/numpy/numpy/pull/16154

[22] https://github.com/numpy/numpy/pull/16108

[23] https://github.com/numpy/numpy/pull/15662

[24] https://github.com/numpy/numpy/pull/16164

General maintenance

The `numpydoc` Sphinx extension defines the NumPy docstring, which is widely used throughout the scientific Python ecosystem. We have improved its test suite and added automated cross-references[25] for commonly encountered NumPy and Python objects. We are also busy improving docstring validation.

[25] https://github.com/numpy/numpydoc/pull/295

The iterator API was cleaned up and improved.[26] We significantly reduced the amount of code related to ufunc reductions and fixed bugs related to broadcasting in ufunc reductions[27] and generalized ufuncs[28].

[26] https://github.com/numpy/numpy/pull/15162

[27] https://github.com/numpy/numpy/issues/15864

[28] https://github.com/numpy/numpy/issues/15139

Since the inclusion of the new random API in version 1.17, continual improvements have been made to that subpackage, including the addition of `permuted`[29]. This new function provides an alternative to the existing `shuffle` function that handles the `axis` argument in a more intuitive way. A new `multivariate_hypergeometric` distribution has also been added[30].

[29] https://github.com/numpy/numpy/pull/15121

[30] https://github.com/numpy/numpy/pull/13794

Process Improvements

We continued to clean up and improve automated testing, especially around documentation[31], and added features to make reviewing pull requests easier[32]. Several components that were previously included as git submodules have been removed and are now included via standard Python packaging. Weekly community calls were transitioned into alternating issue triage and community meetings.

[31] https://github.com/numpy/numpy/pull/15848

[32] https://github.com/numpy/numpy/pull/16337

Aims: Social

S3: Diversify Contributors

We have not made significant progress on this outcome. As became evident when we published the NumPy Nature paper (see below), our developer community and teams are not representative of NumPy's user base.

Ralf Gommers at QuanSight has made progress on this front through hiring. NumPy is also growing its variety of teams and roles. A town hall was held in October 2020 to discuss diversity and outreach with community members.

Mechanisms identified to help improve team diversity include paid roles, training, and formal mentorship. For the most part, paid summer internships have not been effective at growing the community and retaining developers. There is increased interest in the team to learn more about and improve diversity; the team is also committed to do the outreach necessary to involve those outside the sphere of existing self-selected volunteer developers.

NumPy Paper

A paper entitled *Array programming with NumPy*[33] was published in the September 16th, 2020 issue of *Nature*. The paper was written by members of the BIDS team in consultation with the community.

[33] https://www.nature.com/articles/s41586-020-2649-2

NumPy Community Survey

To learn from our community's experience using and developing NumPy, the Survey Team (under leadership of Inessa Pawson and assisted by Ross Barnowski) designed and administered the inaugural Community Survey. The survey team partnered with students and faculty from a master's course in survey methodology jointly hosted by the University of Michigan and the University of Maryland[34]. The survey was taken online from mid-July to mid-August 2020, and

[34] https://psm.isr.umich.edu/descriptions

gathered over 1200 responses. The survey is being analyzed, and a report is forthcoming[35].

[35] https://rossbar.github.io/numpy-survey-results/

Outreach

At a meeting of the University of Michigan EECS department in January 2020, Ross Barnowski delivered an interactive presentation[36], highlighting the role that NumPy played in recently, highly-publicized scientific breakthroughs. This was followed by panel discussions on reproducible scientific computing.

[36] https://github.com/BIDS-numpy/presentation-uofm-2020

In May 2020, he also hosted a half-day virtual workshop[37] for students of the African Institute of Mathematical Science's Masters in Machine Learning program[38]. Students from AIMS's campuses in Kigali, Rwanda and Accra, Ghana were in attendance.

[37] https://github.com/BIDS-numpy/presentation-AIMS-2020

[38] https://aimsammi.org/

Events

Mere weeks before the 2020 Tensor Developer Summit[39], the event had to be cancelled due to COVID-19. The purpose of the event was to bring together members from the various tensor computation libraries for two days of talks and discussions.

[39] https://xd-con.org/tensor-2020/

We were still able to host two sprints: an in-person developer sprint at BIDS in November 2019[40], and a virtual sprint for the SciPy conference in July 2020[41]. Sebastian Berg presented progress made toward the new data-type system[42] at the Dask summit.

[40] https://github.com/numpy/archive/blob/master/sprints/2019-11-22.md

[41] https://github.com/numpy/archive/blob/master/sprints/2020-07-11.md

[42] https://blog.dask.org/2020/04/28/dask-summit

Contribution statistics

Of the 1031 Pull Requests (PRs) opened and merged since October 2019, the team at BIDS created 228 and merged another 385. Of the remaining PRs, BIDS accounted for at least half the non-author comments on 96, and commented on another 72. In total, the team therefore was involved in about 781 of the 1031 PRs.