Improving NumPy for Better Data Science
Progress Report

Stéfan van der Walt, Jarrod Millman, Matti Picus, Tyler Reddy, Fernando Pérez

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

January–October 2018

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: Jonathan Dugan

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

Specific Aims

On the **social** side, we aim to improve the process of engaging the community in NumPy development, to grow the core team of maintainers, to engage new contributors, and to establish a more diverse and inclusive contributor community.

On the **technical** side, we aim to make NumPy's codebase more flexible and accessible, and to lower the barrier to contribution and maintenance. Specifically, we proposed to improve the frequency and consistency of the release cycle, to improve the data type system, and to establish a new array protocol.

In this report, we describe our progress on these (and related) aims. Our day-to-day activities are coordinated via our public Trello board[1], and are also visible on the NumPy issue tracker[2].

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

[1] https://trello.com/b/Azg4fYZH/numpy-at-bids

[2] https://github.com/numpy/numpy

New Team

We hired Matti Picus (April 2018) and Tyler Reddy (June 2018). Matti is a core developer of PyPy (an alternative implementation of Python), where he contributed to the C-API compatibility layer[3] to allow it to import NumPy. As such, he has an in-depth knowledge of the internals of NumPy. He has been active in the open source community both as a contributor, teacher, and presenter at conferences. Tyler is a staff scientist and principal investigator at Los Alamos National Laboratory, Theoretical Division 6, on a two year sabbatical at BIDS for this project. He is a computational virologist by training, and a core developer and member of the steering committee for both the SciPy and MDAnalysis open source Python libraries.

[3] https://morepypy.blogspot.com/2018/09/inside-cpyext-why-emulating-cpython-c.html

Hiring was overseen by Stéfan van der Walt, Nathaniel Smith, Jarrod Millman, Nelle Varoquaux, Fernando Pérez, and Jonathan Dugan. Nathaniel left BIDS at the end of May and Stéfan took on the role of project lead and manager for Matti and Tyler. Stéfan is a core developer of both NumPy and SciPy, and the project leader for scikit-image. Jarrod, an early NumPy and SciPy community organizer and release manager and current NetworkX release manager, and Fernando, an assistant professor in Statistics and co-founder of the Jupyter project, advise Stéfan.

The NumPy steering committee gave Matti and Tyler commit rights to the NumPy repository, to facilitate triaging and merging of pull requests. A main concern of lead maintainer, Charles Harris, is

that there are not enough developers able to merge existing contributions. Our team effectively grows the core team by two, partially addressing that concern. In addition to reviewing code and bug reports, we are also contributing a significant amount of the new code changes (see Figure 1).

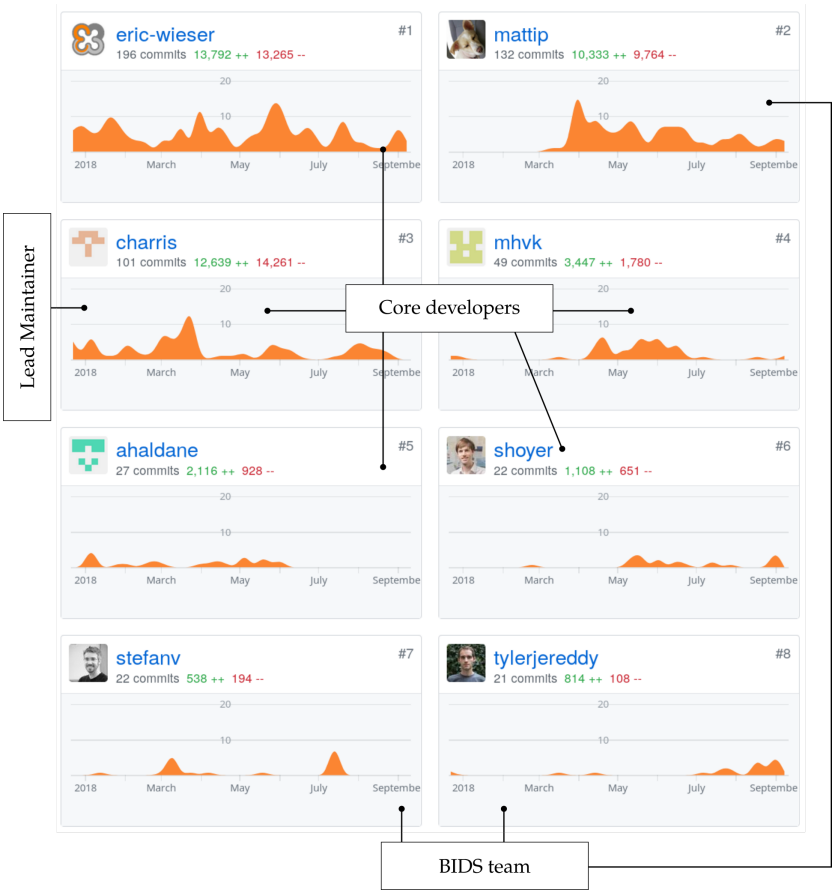

Figure 1: List of the most active contributors for the period January through September 2018, as well as their team affiliations. This only counts code commits and not code review, where we have also contributed significantly.

Social Impact

We have an interesting, and somewhat unique challenge: priorities for NumPy (like many community developed libraries) are determined by the community, so our team has to act in synergy, carefully considering any guidance, criticism, and concerns. As such, we make a significant effort to organize conversations—both within the community, and between ourselves and the community—and to bring developers together for high-bandwidth, face-to-face work sessions. Such "sprints" are commonplace in open source development, and help to dissolve issues that are hard to solve via online fora such as

a mailing list. They also provide an opportunity for developers to connect—socially and work-wise—building much needed trust and rapport in the collegial network that underpins distributed development.

In this context, we have three social aims. We have made significant progress on the first (**S1** Improve Community Engagement) and we are actively working on the second (**S2** Grow Core Team, Add Contributors) and third (**S3** Diversify Contributors).

Table 1: Current statuses of social aims.

Aim	Status
S1	Organized sprints and community meetings; developed roadmap; adopted Code of Conduct; formalized enhancement proposal process.
S2	Increased number of maintainers; developed new teaching material.
S3	Under active investigation; see Challenges & Opportunities below.

Sprints at BIDS

Sprints are technically focused meetings, typically limited in scope and attended by invitation. Through organizing several of these meetings, a community call, and visits by prominent members of the NumPy and SciPy communities, we are well positioned as a hub of scientific Python activity.

- **March:** NumPy Enhancement Proposal (NEP) sprint

  Early in 2018, Jarrod Millman helped to re-establish the use of NumPy Enhancement Proposals (NEPs) as the way of specifying and exploring bigger changes to the library. We talk more about this process in the NumPy Enhancement Proposal section below.

  With a new proposal structure in place, we needed the community to provide concrete suggestions for improvements. This meeting was a first occasion at which such ideas, some previously proposed on the mailing list, could be fleshed out and written down for discussion on the mailing list. The meeting was attending by core developers Stephan Hoyer, Nathaniel Smith, Robert Kern, and Stéfan van der Walt.

  NEPs developed at the sprint include NEP 17, NEP 19, and NEP 22.

http://www.numpy.org/neps/nep-0017-split-out-maskedarray.html
http://www.numpy.org/neps/nep-0019-rng-policy.html
http://www.numpy.org/neps/nep-0022-ndarray-duck-typing-overview.html

- **May:** NumPy developer sprint

  The goal of the May sprint was to write a draft roadmap for NumPy. Core members Charles Harris, Stephan Hoyer, Matti Picus, Jarrod Millman, Stéfan van der Walt, and Jaime Fernández del Río were present.

  The resulting document[4] was wider in scope than a roadmap, so while it became the skeleton for the current roadmap, it also served as a wider collection of ideas and concerns around NumPy development.

  Matt Rocklin, who was visiting BIDS at the time, joined us to discuss a new array protocol, leading to NEP 18.

- **June:** NumPy/SciPy sprint

  In June, we brought Ralf Gommers (chair of the SciPy steering committee) to BIDS to collaborate with us[5].

  NumPy and SciPy—the backbone of the SciPy "stack" or "ecosystem"—have shared roots, with NumPy even briefly called `scipy.core` early in its development. Their developer teams intersect, they follow similar practices, have jointly developed infrastructure, and SciPy remains the primary library consuming NumPy arrays. SciPy is also an excellent avenue for onboarding developers that will eventually work on NumPy itself.

  Because of this shared interest, we worked to **accelerate the release of a SciPy manuscript**[6], celebrating SciPy 1.0. A part of that work includes documenting the origins of NumPy, which we will expand in a follow-up paper. Having a clear citation target for SciPy will help potential contributors justify community time investment, and aid in the solicitation of grants.

  We **refined the initial draft NumPy roadmap**, conceptualized during the NumPy developer sprint in May, with the benefit of Ralf's experience in setting up a similar document for SciPy. This draft[7] was presented to the community at the SciPy2018 conference.

  We also drafted a **proposed restructuring of the scientific Python online landing page**[8]; this would, for example, disambiguate the SciPy library from the ecosystem, and provide new users with navigation instructions for the much larger ecosystem of scientific Python.

[4] https://github.com/BIDS-numpy/docs/blob/master/meetings/2018-05-25_numpy_roadmap_sprint.md

[5] Meeting agenda: https://github.com/BIDS-numpy/docs/blob/master/meetings/2018-06_week-of-scipy-numpy.md

[6] https://github.com/scipy/scipy-articles

[7] https://github.com/BIDS-numpy/docs/blob/master/roadmap/v1_without_SciPy2018_BoF_notes.md

[8] https://github.com/scientific-python/scientific-python.org/issues/1

Conferences

We engaged with the wider scientific Python community at two large meetings, namely SciPy2018 held in July in Austin, Texas, and EuroSciPy, held in August in Trento, Italy.

At **SciPy2018**, our goals were to get feedback on and solidify the community roadmap (drafted and refined during meetings at BIDS), to explore ideas for improving NumPy, and to generate interest in NumPy development.

Matti and Tyler led a discussion session around the proposed NumPy roadmap. Over 100 people attended, and around 20 contributed to the discussion, resulting in a revised roadmap[9].

[9] https://github.com/numpy/numpy/wiki/NumPy-roadmap-v2

On another occasion, we held a discussion session around the proposed improvements to data types, with around 25 people in attendance, many of whom are developers of packages, which consume NumPy arrays and exercise NumPy's full capabilities. These developers will be the eventual users of the new data type system, and as such their input[10] was critical.

[10] https://github.com/numpy/numpy/wiki/Dtype-Brainstorming

Finally, a sprint was held on the last two days of the conference, with the aim of helping new developers on board. One new contributor had her patch[11] accepted by the end of the session.

[11] https://github.com/numpy/numpy/pull/11567

At **EuroSciPy**, our goal was to inform the European community about the grant work we were doing, and to generate interest in contributing to NumPy. Matti delivered a talk[12,13] and a tutorial[14], and had many valuable informal conversations with NumPy users and library developers. At the sprint, five new contributors worked on NumPy, with one particularly useful documentation improvement accepted into the project[15].

[12] https://youtu.be/dGHIHCzabOM?t=1296

[13] https://github.com/BIDS-numpy/docs/blob/master/presentations/NumPyStatus.ip

[14] https://github.com/BIDS-numpy/docs/tree/master/presentations/c_from_python

[15] https://github.com/numpy/numpy/pull/11858

Code of Conduct

In July, we started a discussion[16] on the NumPy mailing list, proposing the adoption of the SciPy Code of Conduct for NumPy. While some preferred a shortened document, there was general agreement that this was a useful addition, and that re-using SciPy's existing Code of Conduct (which Tyler & Stéfan also helped to put in place[17]) would simplify matters. The Code of Conduct committee currently consists of Stéfan van der Walt, Nathaniel Smith, and Ralf Gommers.

[16] http://bit.ly/numpy-discuss-coc

[17] https://github.com/scipy/scipy/pull/7963

Teaching

As part of our community outreach efforts, Stéfan, Matti, and Tyler presented a one day workshop to summer students[18] at the Mathematical Sciences Research Institute (MSRI).

We put videos of our lectures[19,20,21] and materials[22] online, and gathered student feedback[23], which we are using to improve future lectures and materials.

[18] http://www.msri.org/summer_schools/827

[19] http://www.msri.org/summer_schools/827/schedules/24195

[20] http://www.msri.org/summer_schools/827/schedules/24216

[21] http://www.msri.org/summer_schools/827/schedules/24237

[22] https://github.com/BIDS-numpy/2018-07-19-seminar

[23] https://github.com/BIDS-numpy/docs/blob/master/presentations/numpy_msri_feedback.md

Technical Impact

We have four technical aims. **T1** (More Flexible & Sustainable Code) and **T2** (Frequent and Consistent Releases) require continuous investment, while **T3** (Improve Data Type System) and **T4** (New Array Protocol) involve more discrete work.

Aim	Status
T1	Code simplifications have been made, and we are looking for more opportunities to improve flexibility.
T2	Releases are being monitored to ensure consistency.
T3	Under active discussion with the community; experimental code is being written.
T4	A community proposal has been accepted, and is waiting to be implemented.

Table 2: Current statuses of technical aims.

While a lot of smaller tasks—code cleanup, bug triage, documentation improvements, testing—happen on a day-by-day basis, most of these changes are tracked and reflected in the NumPy GitHub issue tracker[24,25]. For work items of broader scope and impact, proposals are written which are subsequently vetted by the community. We outline that process below.

[24] https://github.com/numpy/numpy/issues

[25] https://github.com/numpy/numpy/pulls

NumPy Enhancement Proposals

In 2008, Jarrod, as release manager, labeled three documents related to feature implementations as NumPy Enhancement Proposals, and moved them into their own section of the documentation[26]. This gradually became the default place to describe major changes to the library, but unlike the Python Enhancement Proposals after which they were modeled, there was no formal structure in place. In 2018,

[26] https://github.com/numpy/numpy/commit/eae7e11f672e1c1d3bd66959005db1ff2c4a2c2c

we revisited this organizational aspect in NEP0—Purpose and Process[27], a modified version of Python's PEP process, which describes the process whereby Enhancement Proposals would traverse various states to eventually become implemented in NumPy. We also formatted the documentation[28] to describe the NEPs, organized them into categories, and improved navigation.

[27] https://www.numpy.org/neps/nep-0000.html

[28] https://www.numpy.org/neps

In the six months since the establishment of the new site and the renewed encouragement (at, for example, the NEP sprint) for developers to propose changes, nine NEPs have been written—a third of all NEPs in the history of the project.

We highlight four NEPs that are now finalized, and in which we were involved (in design, discussion, or implementation):

- **NEP 15—Merging two parts of the codebase**. Proposed by Nathaniel Smith and implemented by Matti, it makes the codebase more flexible and approachable. The major work was officially accepted into the code[29] and lays the foundation for simpler adjustments to NumPy internals and a more unencumbered reasoning about the operation of the library (Aim: **T1**).

[29] https://github.com/numpy/numpy/pull/10915

- **NEP 17—Split out Masked Arrays**. There are parts of the NumPy code, such as masked arrays, financial functions, and the `f2py` Fortran compiler, that do not fit within the narrower scope of NumPy as an array object, and exist inside NumPy mainly for historic reasons. This NEP proposed to refactor these—starting with masked arrays[30]—into separate projects under the NumPy umbrella. Its concern is that the additions, while useful in many contexts, clutter the codebase and increase maintenance burden, and that—with improved Python packaging—it is possible and advantageous to split them out. After discussion on the mailing list[31], it was decided not to continue along this path, given the long reliance on the components being bundled with NumPy. Rather, the emphasis would shift toward improving or rewriting the masked array class, with the existing implementation being preserved until a suitable and superior replacement becomes available. (Aim: **T1**)

[30] https://www.numpy.org/neps/nep-0017-split-out-maskedarray.html

[31] https://mail.python.org/pipermail/numpy-discussion/2018-May/078026.html

- **NEP 18—A dispatch mechanism for high level functions**. Initiated at a meeting of NumPy developers at BIDS in June, and shepherded through the approval process by interaction with the wider community. This proposal lets other projects reuse the NumPy API. The suggestion is complicated, and thus solicited many comments, but was recently approved[32] and is now in the implementation phase. (Aim: **T4**)

[32] https://mail.python.org/pipermail/numpy-discussion/2018-August/078493.html

- **NEP 20—Expansion of Ufunc signatures**. This grew out of work done by the BIDS team in reorganizing the `__matmul__` function. The proposal, which extends the flexibility of the `ufunc` machinery, was approved and is in final stages of implementation. Its completion will allow re-aligning `__matmul__` with the other `__dunder__` mathematical functions in numpy, simplifying the code and enabling further refactorings to the design of the critical `ufunc` flow. (Aim: **T1**)

Other experimental projects were initiated and later abandoned. For example, we explored whether it was possible to easily improve the speed of NumPy's CSV text file reader, since in practice users often have to import the `pandas` library for loading files only. Community feedback was that this was a tolerable state of affairs, and that we should consider more important challenges; our focus was hence directed elsewhere.

T1 More Flexible & Sustainable Code

NEP 15 is an example of a larger-scale change enabled by the support of this grant. It presented challenges since it required volunteer core developers to review modifications produced by full time employees. It is important to implement mechanisms that prevent the introduction of new problems in the code base, and that reduce the quality control burden placed on reviewers. To this end, the continuous integration testing infrastructure used to validate each NumPy code contribution now includes assessments of the code lines covered by unit tests in Python and C source files, with aggregate project source line coverage currently at 85%.

The community roadmap includes adding continuous integration testing for more exotic computer architectures, to prevent accidental breakage on these more rare and exotic platforms. To this end, we have added testing for the ARMv8 architecture[33].

We also contributed solutions for a Fortran compilation issue reported by NASA[34] that affected multiple projects; these changes should be included with the next major NumPy release (`1.16`).

[33] https://github.com/numpy/numpy/pull/11906

[34] https://github.com/numpy/numpy/issues/11824

T2 Frequent & Consistent Releases

NumPy has a fairly steady major release once every six months. For example, NumPy 1.15.0 was released near the end of July, about six months after the previous major release 1.14.0 in January. Matti released version 1.14.5.

2018 Releases

Jan 6	**1.14.0**
Feb 20	1.14.1
Mar 12	1.14.2
Apr 28	1.14.3
Jun 6	1.14.4
Jun 12	1.14.5
Jun 21	1.15.0rc1
Jul 9	1.15.0rc2
Jul 23	**1.15.0**
Aug 21	1.15.1
Sep 23	1.14.6
Sep 23	1.15.2

T3 Improve Data Type System

The data type (dtype) system overhaul is currently at an exploratory stage—we are actively experimenting with enhancing existing dtypes and creating new ones, while noting any obstacles and difficulties encountered.

For example, we have a prototype[35] of a dtype-related issue from the community roadmap[36], namely more natural combining of arrays of letters.

During the data type discussions at NumPy, the community asked that the new system should:

- Allow pure Python dtype implementations. Currently many parts of the dtype system require developers to write C code.
- Allow true subclassing dtypes, which would simplify implementation of things like a unit dtype on top of float dtypes.
- Allow storing extra data in a dtype, which would allow construction of a category dtype with textual labels.

The current data types might be coerced to provide some of these features, but the implementation is likely to be unclear and suboptimal; thus we are investing in a new, clean, and extendable implementation instead.

[35] https://github.com/numpy/numpy/pull/11931

[36] https://www.numpy.org/neps/roadmap.html

T4 New Array Protocol

NEP 18, a proposal for an `__array_function__ protocol`, was accepted[37] on September 20, 2018, after an extended discussion on the community mailing list[38]. The proposal provides a mechanism whereby NumPy can operate on objects from external software libraries, while those libraries dictate certain operational behavior. For example, it could allow addition that leverages multiple cores in a parallel library like dask. The NEP is now being implemented.

[37] https://www.numpy.org/neps/nep-0018-array-function-protocol.html

[38] http://numpy-discussion.10968.n7.nabble.com/Proposal-to-accept-NEP-18-array-function-protocol-td46000.html

Challenges & Opportunities

After forming a working team earlier this year, we turned to pressing and practically addressable matters: refactoring and cleaning up NumPy, reviewing code, triaging tickets, and improving infrastructure. This was followed by higher level, strategic initiatives, such as the organization of various meetings and the writing of a roadmap. In turn, these activities improved our understanding of our working relationship with the community, and helped us to set technical priorities.

With those priorities in place, we are now ready to approach the social aims of the grant—arguably harder to achieve than technical aims, but no less important for the long term benefit of the project.

What follows is an outline of challenges we face, but also of the opportunities—especially social—that we foresee for the coming year.

Reviewer Bandwidth

As explained in the earlier section on social impact, we are reliant on the community to provide feedback on proposed changes, and to review the code we write. As prodigiously productive as some community members are, it is hard to find the bandwidth to review code written by two or more full time developers.

Our team is cognizant of the matter, and spends time working on infrastructure, code review, and administrative matters to free up time for community members. Even so, this remains a fundamental challenge without an easy solution in sight.

The long term solution is to grow the core developer group, as discussed in the next section.

Expanding and Diversifying the Developer Team

NumPy is a complicated project, reducing the likelihood of "fly by" contributions. Instead, developers require a deep understanding of both low and high-level programming, of the underlying NumPy array principles, and of various technical details specific to NumPy. In other words, contributors need to invest a significant amount of time learning, and then even more contributing their new feature. This is a high barrier to entry, and there are not many people with enough technical skills and time available to participate.

We can lower the barrier to entry somewhat by, for example, improving developer and low-level documentation, or by simplifying the code structure. But, to significantly reduce the risk of having very few programmers take care of the entire library, we will need to solicit new talent and train the next generation of NumPy core developers.

In the coming year, we will address core developer scarcity and lack of diversity through both training and outreach.

For **training**, we will hold a five day **NumPy Developer Days** summit at BIDS. In broad, the various days will be split into:

- **Day 1**: Training for NumPy users (beginner & intermediate)
- **Day 2**: Training new NumPy developers (advanced)
- **Day 3**: Onboarding of new developers
- **Days 4, 5**: Hands-on NumPy development with core mentorship

Under-represented community members will be actively recruited for sponsored attendance.

For **outreach**, and to expand and diversify the core NumPy developer team, we will work with a program such as Outreachy[39] to provide funded internship opportunities to work on NumPy with the team at BIDS.

[39] https://www.outreachy.org

Related Activities

At BIDS, we frequently host sprints for other packages in the scientific Python ecosystem[40]. Talking to the developers of those packages often proves to be serendipitous, highlighting usage issues or desired features that would otherwise escape notice.

[40] https://scisprints.github.io/

For example, when Matt Rocklin, author of `dask`, visited BIDS for a *scikit-image/scikit-learn/dask* sprint, we asked him to participate in the NumPy array protocol discussion. With `dask` being a primary consumer of such an interface, he was able to make relevant and actionable recommendations, and became a co-author of the resulting NEP.

With BIDS as a central point of connection, we have had helpful discussions with Michael Droettboom (author of Air Speed Velocity, and more recently an implementation of NumPy on WebAssembly), Travis Oliphant (founder of Anaconda, now with QuanSight), Aric Hagberg (Los Alamos, NetworkX), Dan Schult (NetworkX), and Juan Nunez-Iglesias (scikit-image).

We have also had helpful conversations with those outside of the Python ecosystem, such as David Anthoff (Julia Tables), Karthik Ram (R OpenSci), and Oleksandr Pavlyk (Intel). These people, who do similar work but under entirely different settings, provide new perspectives that help calibrate our work against that of the larger open source and reproducible science ecosystem.