NASA ROSES 2020 Proposal - Reinforcing the Foundations of Scientific Python

Dharhas Pothina, Tyler Reddy, Andreas Müller,
Matt Haberland, Jeff Reback, Thomas Fan

January 2021

Contents

1 Scientific/Technical/Management (S/T/M)

Large parts of the scientific community use and rely on the scientific Python ecosystem. Array and dataframe data structures, provided by NumPy and pandas, and general scientific and machine learning algorithms, provided by SciPy and scikit-learn, lie at the heart of this ecosystem.

NumPy, pandas, SciPy and scikit-learn are mature community-driven open source projects. Together they form the foundation of many downstream technique specific, domain specific and application specific scientific software projects (See Fig. 1).

Although these are distinct projects with their own communities and governance, they are mutually dependent. The projects attempt to partner and work together to maintain a robust base upon which other projects can build. By submitting a joint proposal, the leadership of these projects aims to improve this integration and solidify this foundation.

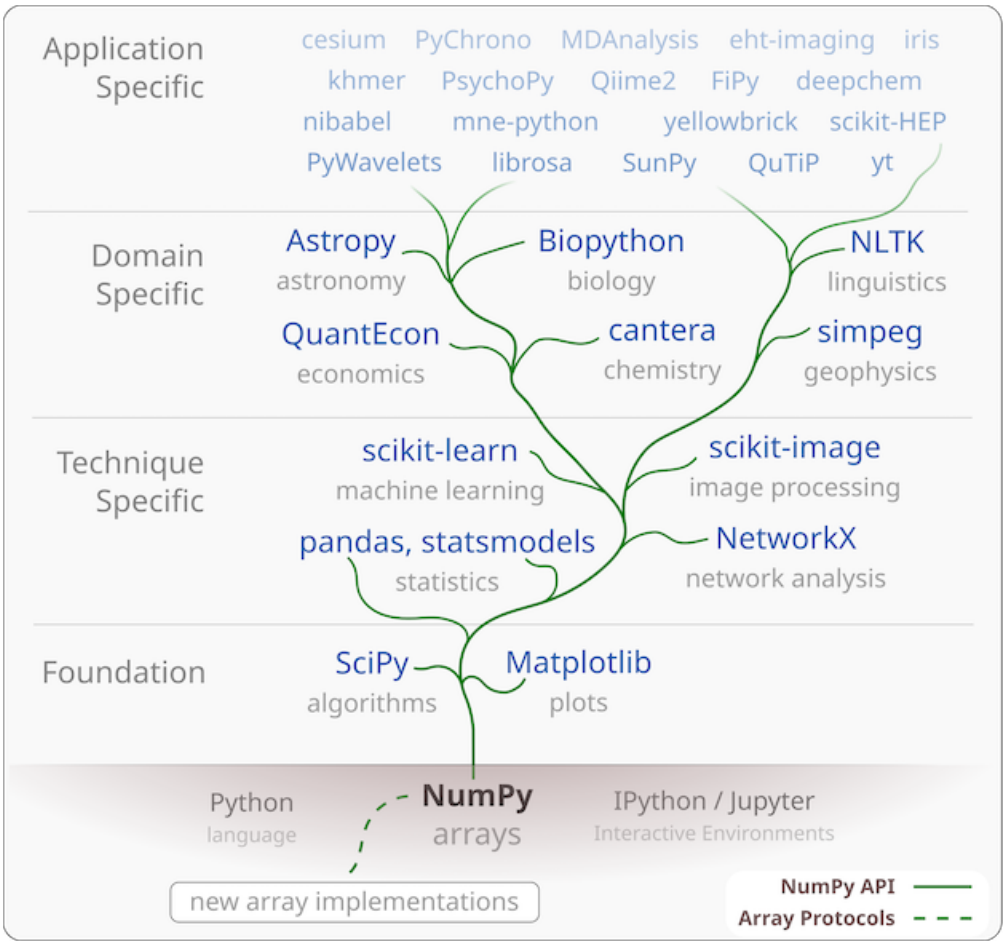

Figure 1: The scientific Python ecosystem

1.1 Objectives

Although these 4 projects are mature open source projects with massive userbases maintaining and improving them remains a challenge. Much of the work on these projects is through volunteer efforts. Through the work proposed, we will address key challenges in the maintenance of these four projects, as well as implement technical improvements with a focus on computational performance of and tighter integration between them.

From a maintenance perspective, there are two primary objectives: reducing the burden on project maintainers and encouraging participation in the community by new and existing volunteers.

Maintaining continuous integration (CI) and extending it to, e.g. new Python versions and more hardware platforms, is one of the most time-consuming activities for each project. We intend to have an infrastructure engineer work on this, sharing expertise between projects. NumPy, pandas, SciPy and Scikit-learn all make regular releases. Providing high-quality packages for Linux, Windows and macOS - and more recently also ARM64 and PowerPC - is critical for end users. Automating this process and integration testing between the different projects will save maintainers a lot of time, prevent regressions, and allow more frequent releases to be made easily.

The volume of both open issues and proposed contributions for each project is large - between 1300 (SciPy) and 3400 (pandas) issues, and between 180 (pandas) and 740 (scikit-learn) pull requests. The bulk of that work is volunteer effort. We will perform targeted issue triaging and code review to be able to prioritize the most important bug fixes and contributions, and get them integrated into the projects.

Computational speed improvements, lower memory usage and the ability to parallelize algorithms are beneficial to most use cases, and were at or near the top of desired enhancements in two recent end user surveys. Therefore performance will be a main theme of the technical work we propose. Our key objectives are:

- Better cross-project integration, in particular via data-type compatibility (e.g. reimplement pandas dtypes on top of NumPy's new dtype extension mechanism) and zero-copy protocols (e.g., enable scikit-learn to consume pandas dataframes in more situations).
- Adopt and implement a uniform API per project for parallel execution. This will build on the `n_jobs` and `workers` patterns in scikit-learn and SciPy, respectively.
- Extend the use of accelerator technologies. Pandas recently started using Numba as an optional dependency which we will extend to a larger part of the API; all projects have scope for significant performance improvements via heavier use of Cython.

More significant performance gains, and the ability to scale up to larger data sets, will be unlocked by enabling the use of GPU and distributed arrays in SciPy and scikit-learn. Initial experimental work has proven that this is feasible, via use of NumPy array protocols plus CuPy for GPU support and Dask for distributed support. We will solve the remaining technical hurdles to ensure that users are able to smoothly use CuPy and Dask throughout at least one SciPy submodule and one scikit-learn submodule.

1.2 Impact

Integration of CI and testing infrastructure between projects will reduce the maintenance burden on the core contributors and allow them to explore higher value algorithmic and performance aspects of the projects. Adapting to new hardware will become more straightforward.

Issues and contributions that are ignored have a tendency to drive potential volunteers away from the community. Regular triaging of issues and contributions will make the projects more welcoming to new contributors.

Stronger integration between the projects and the proposed performance improvements will allow these projects to be utilized for the massive datasets now being generated by the SMD communities.

The proposed work will increase both the health and the capabilities of these four projects, which is not only beneficial to the tens of millions of users they have, but also to the wider Python ecosystem - including space science specific libraries like AstroPy and SunPy - and the NASA missions that rely on scientific computing with Python.

1.3 Relevance to the SMD mission

Establishing the overall prevalence of open source software to the SMD mission is hard to quantify due to the software being freely distributed and not systematically cited. Examples of NumPy, SciPy, pandas and scikit-learn usage can be found in missions and projects across all four SMD divisions.

1.3.1 Selected Downstream Libraries

One way to look at the relevance to the SMD mission is to look at downstream libraries and communities that are based on the projects that form this proposal. Below is a small selection of high profile libraries that are important to the SMD.

- Matplotlib - Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
- AstroPy - The Astropy Project is a community effort to develop a common core package for Astronomy in Python and foster an ecosystem of interoperable astronomy packages.
- SunPy - The community-developed, free and open-source solar data analysis environment for Python.
- Pangeo - A community platform for Big Data geoscience.

Matplotlib is used by all 4 SMD communities, AstroPy is focussed Astrophysics, Sunpy on Heliophysics and Pangeo covers Earth Science and Planetary Science.

1.3.2 Usage in NASA Training Materials

An examination of publically available NASA training materials shows that NumPy, SciPy and pandas are key libraries that NASA scientists are encouraged to understand and use [1, 2] and that scikit-learn is considered an important important tool in the Machine Learning space [3]

1.3.3 Alignment with NASA Vision

NumPy and scikit-learn are mentioned explicitly in the 2018 National Academy of Sciences white paper titled *"Open Source Software Policy Options for NASA Earth and Space Sciences"* [4]. All four packages NumPy, SciPy, pandas, and scikit-learn packages are cited in a call to support the Python numerical core in a white paper co-authored by the Space Telescope Science Institute (STSI) in the 2020 Decadal Survey on Astronomy and Astrophysics [5].

Additionally, these 4 projects align with the following priorities and strategies outlined in the NASA 2020-2024 scientific vision [6].

- Priority 2 - Innovation
  - Strategy 2.2: Foster a culture that encourages collaboration in pursuit of common goals. *This proposal enhances the collaboration between 4 mature open source communities. Together these communities work toward the goal of improving the scientific Python ecosystem*
- Priority 3 - Interconnectivity and Partnerships
  - Strategy 3.4: Provide increasing opportunities for research institutions, including academia and non-profits, to contribute to SMD's mission. *The proposal includes partnerships between government and academic institutions and open source communities.*
  - Strategy 3.5: Pursue public-private partnerships in support of shared interests with industry. *The proposal is lead by Quansight an industry leader in open source in partnership with government, academic and open source entities.*
- Priority 4 - Inspiration
  - Strategy 4.1: Increase the diversity of thought and backgrounds represented across the entire SMD portfolio through a more inclusive environment. *We intend to place a strong focus on bringing in people from under-represented minority groups into the open source communities that surround the project.*

1.3.4 Metrics on usage by NASA software and citations

The NASA Public GitHub organization hosts NASA code approved for open source. This represents only a small fraction of all software developed at NASA, however it can still sketch a picture of how widespread usage of NumPy, pandas, SciPy and scikit-learn is at NASA SMD.

The 330 repositories under the NASA Public GitHub organization contain in total 889 imports of NumPy, 164 imports of SciPy, 55 imports of pandas and 20 imports of scikit-learn. Packages making significant use of NumPy, pandas, SciPy and/or scikit-learn include:

- Astrobee Robot Software, *performs vision-based localization, provides autonomous navigation, docking and perching, manages various sensors and actuators, and supports user interaction via screen-based displays, light signaling, and sound*,
- DELTA (Deep Earth Learning, Tools, and Analysis), *a framework for deep learning on satellite imagery*,
- Kamodo, *a CCMC tool for access, interpolation, and visualization of space weather models and data in Python*,
- Kepler-PyKE, *A suite of Python/PyRAF tools to analyze Kepler data*,
- RtRetrievalFramework, *software to retrieve a set of atmospheric/surface/instrument parameters from a simultaneous fit to spectra from multiple absorption bands*,
- SROMPy, *Stochastic Reduced Order Models with Python*,
- MXMCPy, *an open source package that implements many existing multi-model Monte Carlo methods (MLMC, MFMC, ACV) for estimating statistics from expensive, high-fidelity models by leveraging faster, low-fidelity models for speedup*,
- Lightkurve, *a community-developed, open-source Python package which offers a beautiful and user-friendly way to analyze astronomical flux time series data, in particular the pixels and lightcurves obtained by NASA's Kepler and TESS exoplanet missions*,

Citations in the NASA Astrophysics Data system provide another metric indicating widespread usage in fields relevant to NASA. NumPy [7, 8] has 1968 citations, SciPy [9] has 821 citations (note, within one year of publication), scikit-learn [10] has 141 citations. The canonical pandas citation [11] is a book which is not indexed in the NASA Astrophysics Data system; it has over 3200 citations in Google Scholar.

1.4 Technical approach and methodology

1.4.1 Maintenance activities

Each project has a large open issue and pull request count - from 1300 (SciPy) to 3400 (pandas) issues, and from 140 (pandas) to 750 (scikit-learn) pull requests. These represent the backlog of maintenance work for each project. Triaging this backlog, i.e. labelling issues with categories like "high priority", "good first issue", "needs review" and the topic or submodule they concern, is high value for the project. It allows volunteer contributors to find good places to contribute, and alerts maintainers to work needing their attention. Given that such triaging is currently not performed on a regular basis, we will execute this task throughout the grant period.

Continuous integration (CI), i.e. automatically testing each code change, is essential to maintaining each project. CI relies on multiple free-for-open-source services like Travis CI and GitHub Actions, and itself requires significant effort to keep running on these services. Maintaining and improving CI will be the second task ongoing throughout the grant period. In particular, we will:

1. Update CI configurations and tooling so they are the same for each project, thereby sharing maintenance cost,

2. Enable CI for new Python versions when those become available,
3. Improve hardware platform support, in particular for platforms that only recently became available like ARM64 Linux, ARM64 macOS, and PowerPC.
4. Update how releases get made via CI, so a single commit will build *all* release artifacts for a given release tag, and
5. Update build systems for the pending deprecation of the `distutils` module in the Python standard library.

Our other maintenance efforts will be driven by the outcome of issue triaging, and focus on code review (which is much more of a bottleneck than writing new code to fix bugs) and on identified structural issues in the code base that are too large or complex for a volunteer to tackle in their spare time. Overall we aim to spend one third of the engineering time in this grant on maintenance, and ensure that the issue and PR count is stable or decreasing over time - which is the best metric we know of for sustainability.

1.4.2 Performance improvements

Speed improvements, lower memory usage and the ability to parallelize algorithms are beneficial to most science domains and use cases - and in particular to data-intensive domains such as earth observation and astrophysics. They were also the top priority in the recent NumPy user survey with over 1200 respondents [12]. Therefore performance will be one of the main themes of the technical work we propose; we'll focus on this for the duration of the program. We identify a number of concrete work items here which will be augmented by user feedback:

Performance benchmarking suites based on Airspeed Velocity are present in each project, however they (a) cover only a small fraction of the code base, and (b) they do not get run automatically on dedicated hardware as part of CI. Therefore we will set up such dedicated hardware, make that connection to CI, and publish the results of each run to a website, with any performance regressions flagged for investigation. The benchmark coverage will be extended from the current coverage, estimated at 5-15% of functions, to $\geq 50\%$ for each project. Finally, we will add benchmarks for memory usage of functions. This benchmarking strategy will then be used to validate performance improvements we will implement.

SIMD usage in NumPy. Improving NumPy performance will have a positive impact on almost every use case. We will optimize the performance of key NumPy array operations (e.g. indexing) and core functionality like ufuncs via strategic use of newer CPU hardware acceleration features, i.e. SIMD instructions. This work will build on the universal SIMD architecture introduced recently in NumPy [13].

Pandas memory usage optimizations. Pandas was designed internally for optimal performance (zero copying) on 2D NumPy arrays. However, pandas dataframe construction from 1D arrays has become a widespread use-case. Furthermore, pandas often makes unnecessary copies. These two factors together result in excessive memory usage, which is often the limiting factor in performance. Based on the memory benchmarks and API usage data [14], we will optimize memory usage hotspots.

Parallelization interface in SciPy. The original SciPy code did not emphasize parallel execu-

tion. However, over the past decade, trends in microprocessor architecture indicate that clock speeds are stagnant but the numbers of CPU cores per chip are increasing [15]. The core SciPy routines therefore need to be updated to take advantage of modern hardware. SciPy has a "workers" API pattern, which is used in a few sets of functionality like FFTs and k-D trees. We will build on this pattern, and add support for it to all SciPy optimization functionality.

Cython performance optimizations. Cython 3.0 is about to be released, and it is the first release with full support for the new (post-1.7) NumPy C API. Scikit-learn makes extensive use of Cython; we will update its code base for Cython 3.0, which will bring both opportunities to improve performance for code using the NumPy C API and to switch to Cython memoryviews where those are more efficient.

1.4.3 GPU and distributed array support

Hardware for scientific computing is becoming more heterogeneous every year. Python users are often requesting better support for their particular hardware, e.g. for HPC systems [16] or deep learning applications [17]. Significant exploratory work by the NumPy community [18, 19] has established that it is possible to standardize a set of NumPy APIs and via a dispatch mechanism support use of CuPy, Dask, JAX and other NumPy-compatible packages in downstream libraries. Implementation requires the teams to coordinate on the exact APIs to support, formalize that set of APIs in NumPy, and then add support for the dispatching mechanism in SciPy and scikit-learn. GPU usage via CuPy will be the first target, because CuPy implements nearly 100% of the NumPy API.

1.4.4 Improved string dtypes

NumPy is coming towards the end of a large redesign of its dtype ("data type") system [20]. This gives the opportunity to add new string dtypes for encoded strings (e.g., UTF-8) and variable-length strings, an explicit goal on the NumPy roadmap [21]. We will add both of these dtypes. The variable-length string dtype will then be used in pandas, which as of today is forced to use the "object" dtype to store string data in dataframe columns. This will yield both usability improvements and easier to understand code within pandas.

1.4.5 Pandas dataframe support in scikit-learn

Scikit-learn was originally designed only with NumPy arrays in mind, however recently support for accepting pandas dataframes without conversion to NumPy arrays was added. One important feature still missing [22] is the ability for transformers to return dataframes rather than arrays when they receive a dataframe as input. We will add this feature. Furthermore we will improve memory usage by avoiding unnecessary data copies where possible. Such memory copies are often triggered because scikit-learn code was written for 2-D arrays while pandas dataframes are a collection of 1-D arrays that are not stored contiguously in memory. They can often be avoided by an explicit code path.

1.4.6 Optimization functionality in SciPy

SciPy's linear programming functionality is missing some oft-requested functionality; we will address this by adding interfaces to the best available open source libraries available. SciPy 1.6.0 made a start by adding the HiGHS Linear Programming solvers. We will extend that interface to include the HiGHS mixed-integer programming capabilities. Separate interfaces to CLP (linear programming), CBC (mixed-integer programming) and Ipopt (nonlinear programming). Each of these solvers is written in C++; this will be exposed to Python via Cython, and tests will be written in Python.

1.5 Sources of uncertainty & mitigation

NumPy, pandas, SciPy and scikit-learn are community-driven open source projects. Decision making is by consensus, with a Steering Council (a small group of senior core developers) stepping in only in case consensus cannot be reached. Because of the central position of these projects in the wider PyData ecosystem, new features get scrutinized heavily, and hence there is a risk of them being rejected. For maintenance work and performance optimizations that risk is minimal. The two deliverables with a non-negligible amount of uncertainty are the GPU and distributed array support, and the improved string dtypes. Each of these two deliverables is included prominently on the project roadmaps. The topics and benefits have already been discussed multiple times by the core development teams; inclusion on the roadmap means the features are desirable and the risk is limited to smaller decisions related to the precise implementation.

We mitigate the risk as follows:

1. The team for the proposed work includes senior maintainers of each project. Part of their role is to align with their co-maintainers and the wider community before the work starts, to ensure possible concerns are raised early and interested people can follow along with the design and implementation work.
2. Each deliverable will be proposed via an "enhancement proposal" to the project, in which topics like benefits to end users, maintainability and backwards compatibility are addressed explicitly.
3. If it turns out to be necessary after review of either the enhancement proposals or the implementation, the features may be implemented via "feature flags" that let users explicitly opt in to the new features, so the introduction can be gradual.

The estimated effort for the proposed work also contains uncertainty, as for any software project. We intend to mitigate this by applying best practices for agile development, and by tackling the complex tasks that are dependencies for other tasks (in particular the string dtypes and the array API standardization) early on so potential issues surface in time.

1.6 Team composition and responsibilities

This proposal brings together leading figures of each of the four participating projects. They are ideally positioned to make contributions with ecosystem-wide impact and coordinate efforts across packages. **Tyler Reddy** is a core developer of both NumPy and SciPy, and the SciPy release manager. He will oversee contributions to these projects, as well as perform SciPy releases and extend support for new hardware platforms and Python versions. **Andreas Mueller** has been a scikit-learn core developer for over 10 years, while **Jeff Reback** has co-lead pandas for the past 8 years. They will oversee the contributions to scikit-learn and pandas, respectively. **Matt Haberland** is a SciPy core developer. He will deliver the proposed new optimization features for SciPy. In addition, he will build on his previous efforts mentoring undergraduate research assistants at Cal Poly to contribute to SciPy. **Thomas Fan** has been a full-time scikit-learn core developer for the past two years. He will develop the GPU and distributed array support - that he already led the exploratory feasibility work on - and the zero-copy pandas dataframe support.

We combine this group of core developers with the experience of **Dharhas Pothina** (PI) leading teams delivering both open source and commercial projects related to the PyData stack as consulting lead at Quansight. Pothina has 15+ years of experience with these projects, and will act as the engineering manager for the project.

Past experience with funded work on community open source projects has taught us that the most effective strategy for team building is to combine experienced maintainers with hiring developers new to the community. This ensures the number of active maintainers grows over time - essential to the health of these projects which are still largely volunteer-driven - and that it brings in new skills and energy. Importantly, it is also an opportunity to increase the diversity of the projects. We intend to place a strong focus on bringing in people from under-represented minority groups.

We include two roles that focus on cross-project contributions, with one engineer focusing on continuous integration (CI) and packaging infrastructure, and the other on performance (speed and memory usage) improvements. Transferring best practices and sharing infrastructure tooling between the open source projects will increase the maintainability of the code bases. A third new software engineer will contribute to both NumPy and pandas, with a focus on string dtypes and user-defined functions. A fourth software engineer (half-time) will focus on GPU and distributed array support in SciPy, learning from the experience of Thomas Fan doing the same for scikit-learn.

These engineers will join Dharhas Pothina and Thomas Fan at Quansight Labs, which was created as a home for PyData developers by the creator of NumPy and SciPy (Travis Oliphant) and employs a significant numbers of founders and maintainers of PyData projects. Quansight Labs is already an Institutional Partner to both NumPy and SciPy, and provides an optimal environment for the engineers to learn about open source best practices and integrate into the PyData community.

Parts of this proposal will be executed by sub-awards to Los Alamos National Lab (for Tyler Reddy) and Cal Poly (for Matt Haberland).

1.7 Work plan

We will have two types of activities: *(a)* ongoing tasks like maintenance, issue triaging and code review, and *(b)* milestones for new feature and performance improvement deliverables.

As a team we will spend one third of our time on ongoing tasks, with each team member contributing to those tasks. We will track the progress towards milestones via agile project management, with daily checkins via Slack and weekly project meetings. Each deliverable has a target date for initial merge into the main branch of the relevant open source project, and a phase after that to address potential issues, write tutorials and developer documentation, and help downstream projects adopt the feature.

The key milestones for year 1 are:

- Set up benchmarking infrastructure and complete the benchmark suites,
- Implement string dtypes in NumPy,
- Array API standardization support in NumPy,
- Memory use optimizations in pandas,
- HiGHS interface for mixed-integer optimization in SciPy, and
- Pandas dataframe support in scikit-learn.

The key milestones for year 2 are:

- SIMD performance optimizations in NumPy,
- Adopt new string dtypes in pandas,
- GPU array suport via CuPy in scikit-learn and SciPy,
- CLP and CBC wrappers in SciPy,

The key milestones for year 3 are:

- Extend use of Numba in pandas user-defined functions,
- Distributed array support via Dask in scikit-learn and SciPy,
- Ipopt wrapper in SciPy,
- Complete Cython optimizations in scikit-learn,

The timeline for these milestones and dependencies between them are shown in figure 2. The management structure for the project is shown in figure 3.

1.7.1 Knowledge dissemination

Estimates of the size of the user base of NumPy, pandas, SciPy and scikit-learn range from five to forty million users. Disseminating knowledge at this scale is best done via documentation. We will produce three forms of documentation for each deliverable: tutorials aimed at discovering and learning the features, detailed API reference documentation, and design documentation aimed at future developers.

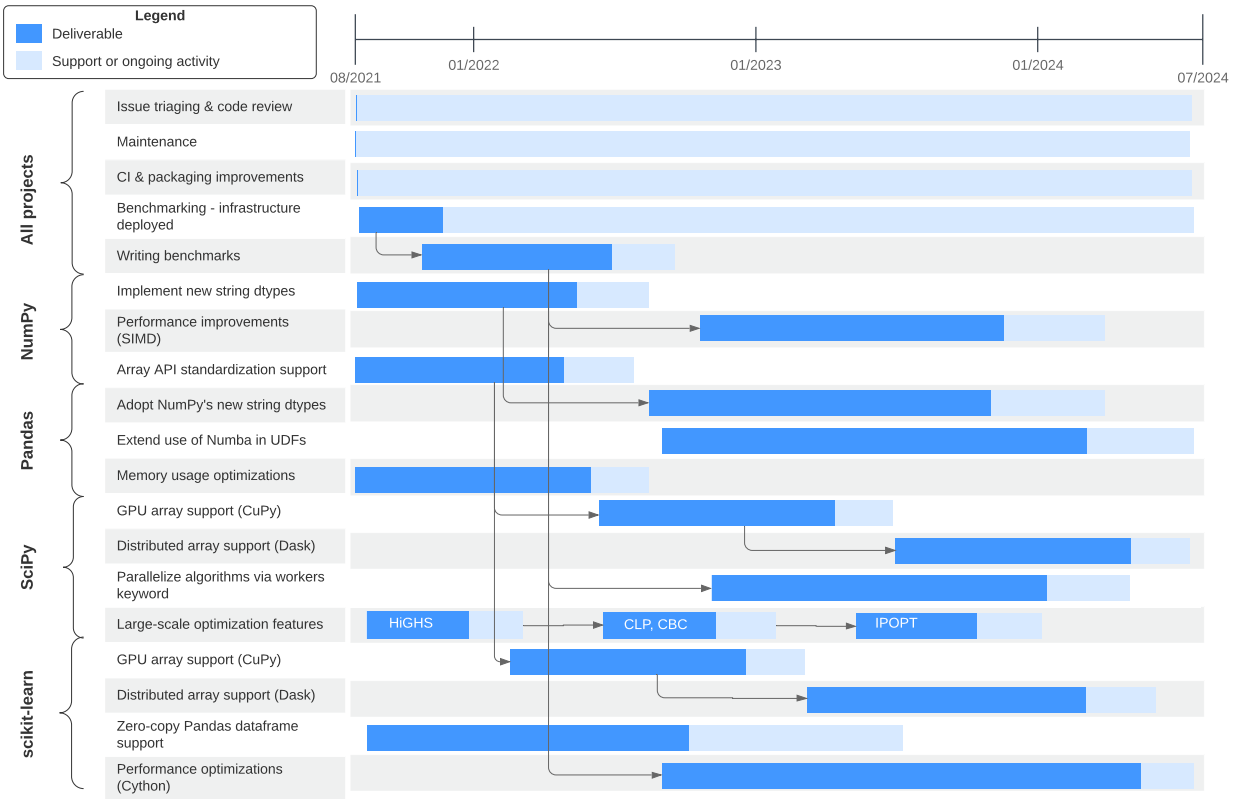

Figure 2: Gantt chart of deliverables and ongoing activities for common activities over projects, and for NumPy, pandas, SciPy and scikit-learn.

1.7.2 Governance and development model

Each of the open source projects we propose work on has a formal governance structure [23, 24, 25, 26], uses the permissive 3-clause BSD license, and has well-established and documented practices of collaborative development which takes place on GitHub. The engineers executing the proposed work will perform all work, including early prototyping and design discussions, in public and will follow the existing development guidelines. This includes:

1. For new features, propose the feature on the mailing list or (for larger features) via a formal "enhancement proposal",
2. For bug fixes and other maintenance work, submit a pull request on GitHub and address review comments that other community members may have on it,
3. For topics requiring high-bandwidth conversations, hold those in publicly announced community where possible - and at a minimum, give summaries of important offline conversations there.

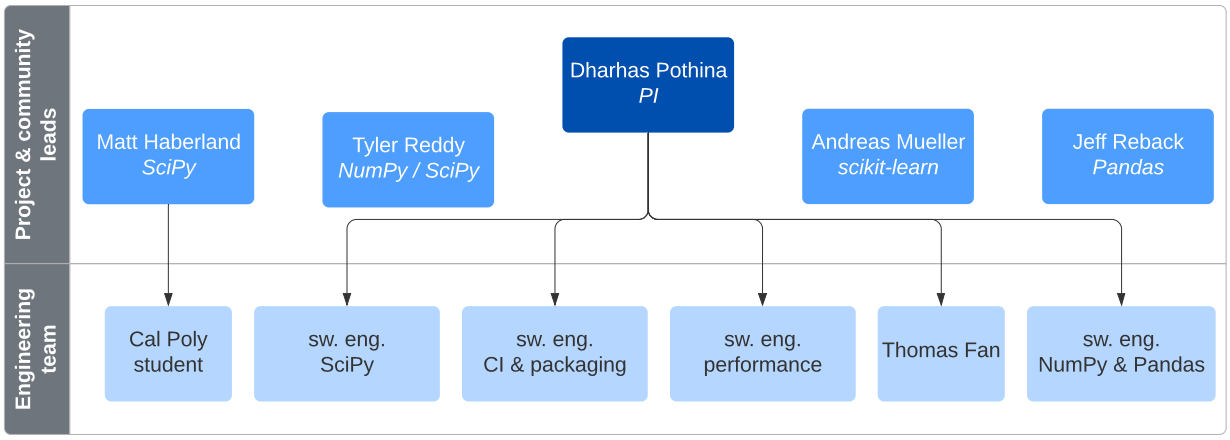

Figure 3: Management structure for the project.

1.7.3 Sustainability

We follow a few key *principles* to ensure that our proposed activities increase the long-term sustainability of the projects and the community around it:

1. Funded developers must make the life of volunteer contributors easier rather than harder. This implies they have to do more than their fair share of, e.g., code review and maintaining infrastructure and websites.
2. Do not increase the amount of technical debt. This means for example that meeting milestones may not be an excuse for taking shortcuts, and that existing technical debt should be addressed when it is encountered during the process of adding a new feature.
3. Pay received mentorship forward. This can be done for example by helping onboard new contributors and improving developer documentation.

We can *measure* sustainability, at least partially, through:

1. The number of active maintainers, and the number of unique contributors per release or time period.
2. Release frequency - each project does feature release at least twice a year, and bugfix releases more often.
3. The median time duration for first review of a pull request or response to an issue.
4. The balance between number of issues opened vs. closed, and pull requests opened vs. merged.

Furthermore, we would like to emphasize that this proposed work would be the first funded work that spans across these projects and shares engineers who focus on core needs of the projects like CI and packaging improvements. NumPy, pandas, SciPy and scikit-learn developers have been collaborating informally for a long time; this work will significantly strengthen those links, which enhances knowledge sharing and builds community - both important for sustainability.

1.7.4 Community & inclusivity

NumPy, pandas, SciPy and scikit-learn, as large community projects driven mostly by volunteers, all have a strong interest in community building. They each have a Code of Conduct [27, 28, 29, 30], and extensive contributing guidelines - from pull request templates to long-form tutorials on how to build the project and submit a pull request. Each project holds multiple in-person or virtual sprints coinciding with SciPy or PyData conferences where everyone is welcome and the focus is on onboarding newcomers. NumPy, pandas and scikit-learn each have also given talks to or held events for groups like Data Umbrella and Women in Machine Learning and Data Science that focus on under-represented groups in STEM or open source, and SciPy is currently planning an event with "Mentored Sprints" for diverse beginners.

We aim for this proposal team to actively participate in and help organize such community building activities. In addition, we'd like to emphasize that we will put a strong emphasis on creating a diverse team through focusing on attracting a diverse set of applicants for the roles that we need to hire for.

References

[1] Nasa advanced software technology group (astg) spring 2021 python classes. https://www.nccs.nasa.gov/nccs-users/user-events/python-classes.

[2] Nasa advanced software technology group (astg) python training opportunities. https://modelingguru.nasa.gov/docs/DOC-2775.

[3] Nasa high-end computing capability (hecc) knowledge base - machine learning. https://www.nas.nasa.gov/hecc/support/kb/machine-learning-173/.

[4] Engineering National Academies of Sciences and Medicine. *Open Source Software Policy Options for NASA Earth and Space Sciences*. The National Academies Press, Washington, DC, 2018.

[5] Joseph Harrington, Ralf Gommers, Chelle Gentemann, Derek Buzasi, Kevin Stevenson, Joshua Pepper, Perry Greenfield, Shubham Kanodia, Thomas Beatty, Ryan Challener, Joe Ninan, Jessie Christiansen, Arif Solmaz, Erik Tollerud, Nicholas Earl, Pey Lian Lim, Larry Bradley, Elisabeth Newton, Rachel Akeson, Megan Sosey, Philip Hodge, Paulo Miles-Páez, Kathleen Labrie, Henry Ngo, Sara Ogaz, Darren Williams, Michael Himes, Kathleen McIntyre, Adrienne Dove, Joshua Colwell, Joe Llama, Ryan T. Hamilton, Geert Barentsen, and Ryan Terrien. Support the Python Numerical Core. In *Bulletin of the American Astronomical Society*, volume 51, page 265, September 2019.

[6] Nasa explore: Vision 2020-2024, a vision for science excellence. https://science.nasa.gov/science-red/s3fs-public/atoms/files/2020-2024_Science-TAGGED.pdf.

[7] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with numpy. *Nature*, 585(7825):357–362, Sep 2020.

[8] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[9] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, Aditya Vijaykumar, Alessandro Pietro Bardelli, Alex Rothberg, Andreas Hilboll, Andreas Kloeckner, Anthony Scopatz, Antony Lee, Ariel Rokem, C. Nathan Woods, Chad Fulton, Charles Masson, Christian Häggström, Clark Fitzgerald, David A. Nicholson, David R. Hagen, Dmitrii V. Pasechnik, Emanuele Olivetti, Eric Martin, Eric Wieser, Fabrice Silva, Felix Lenders, Florian Wilhelm,

G. Young, Gavin A. Price, Gert-Ludwig Ingold, Gregory E. Allen, Gregory R. Lee, Hervé Audren, Irvin Probst, Jörg P. Dietrich, Jacob Silterra, James T. Webber, Janko Slavič, Joel Nothman, Johannes Buchner, Johannes Kulick, Johannes L. Schönberger, José Vinícius de Miranda Cardoso, Joscha Reimer, Joseph Harrington, Juan Luis Cano Rodríguez, Juan Nunez-Iglesias, Justin Kuczynski, Kevin Tritz, Martin Thoma, Matthew Newville, Matthias Kümmerer, Maximilian Bolingbroke, Michael Tartre, Mikhail Pak, Nathaniel J. Smith, Nikolai Nowaczyk, Nikolay Shebanov, Oleksandr Pavlyk, Per A. Brodtkorb, Perry Lee, Robert T. McGibbon, Roman Feldbauer, Sam Lewis, Sam Tygier, Scott Sievert, Sebastiano Vigna, Stefan Peterson, Surhud More, Tadeusz Pudlik, Takuya Oshima, Thomas J. Pingel, Thomas P. Robitaille, Thomas Spura, Thouis R. Jones, Tim Cera, Tim Leslie, Tiziano Zito, Tom Krauss, Utkarsh Upadhyay, Yaroslav O. Halchenko, Yoshiki Vázquez-Baeza, and SciPy 1.0 Contributors. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, Mar 2020.

[10] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011.

[11] Wes McKinney. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[12] Ralf Gommers. NumPy Steering Council, personal communication, 2021.

[13] NEP 38 — using SIMD optimization instructions for performance. `https://numpy.org/neps/nep-0038-SIMD-optimizations.html`, 2020.

[14] Python API inspect - statistics to better understand how python is used and written. `https://github.com/Quansight-Labs/python-api-inspect`, 2019.

[15] Karl Rupp and Siegfried Selberherr. The Economic Limit to Moore's Law. *IEEE Transactions on Semiconductor Manufacturing*, 24(1):1–4, 2011.

[16] Rollin Thomas. NERSC, personal communication, 2019.

[17] Sukwon Kim. Amazon, personal communication, 2019.

[18] NumPy Enhancement Proposals for dispatching - NEPs 18, 30, 31, 35 and 37. `https://numpy.org/neps`, 2020.

[19] Consortium for Python Data API Standards. `https://data-apis.org`, 2020.

[20] NumPy dtype redesign — NumPy Enhancement Proposals 40, 41, 42, 43. `https://numpy.org/neps/nep-0040-legacy-datatype-impl.html`, 2019.

[21] NumPy roadmap. `https://numpy.org/neps/roadmap.html`],.

[22] Scikit-learn roadmap. `https://scikit-learn.org/stable/roadmap.html`.

[23] NumPy governance document. `https://numpy.org/devdocs/dev/governance/index.html`.

[24] Pandas governance document. `https://github.com/pandas-dev/pandas-governance`.

[25] SciPy governance document. `http://scipy.github.io/devdocs/dev/governance/governance.html`.

[26] Scikit-learn governance document. https://scikit-learn.org/stable/governance.html.

[27] NumPy code of conduct. https://numpy.org/code-of-conduct/.

[28] Pandas code of conduct. https://github.com/pandas-dev/pandas-governance/blob/master/code-of-conduct.md.

[29] SciPy code of conduct. http://scipy.github.io/devdocs/dev/conduct/code_of_conduct.html.

[30] Scikit-learn code of conduct. https://github.com/scikit-learn/scikit-learn/blob/master/CODE_OF_CONDUCT.md.

2 Data Management Plan (DMP)

NumPy, pandas, SciPy and scikit-learn are software libraries and do not produce any scientific data as defined in E.1.2 that needs to be preserved.

All software code and documentation for these projects is stored in Git repositories on GitHub under these organizations:

- https://github.com/numpy/
- https://github.com/pandas-dev/
- https://github.com/scipy/
- https://github.com/scikit-learn/

Copies ("clones") of each repository are kept locally by developers on the project, and all released software versions are also archived on PyPI and in multiple other places (e.g., on anaconda.org and in Linux distributions). The proposed work will rely on these existing methods of managing and archiving software code and documentation.

The license for all produced code and documentation is the BSD 3-Clause License, one of the most common and permissive open source software licenses.

3 Biographical Sketches

Dharhas Pothina, Director of Consulting, Quansight
8656 W. Hwy 71, Bldg F200, Austin, Texas 78735, Ph. 512-740-5665

Relevant Experience

15+ years of experience in state and federal research labs leading large multi-disciplinary, multi-agency research projects. Track record of changing culture and pioneering open source technologies within the government. Expertise in computational modeling, big data/high performance computing, visualization and geospatial analysis. SciPy conference executive committee (2020, 2016, 2013, 2012). 2021 Dask Distributed Summit Program Co-Chair. Core developer of the "ulmo" ocean-met data retrieval library. Co-creator of "Qhub", an open source project which streamlines cloud and HPC deployments of JupyterHub and Dask.

Education

Ph.D., Civil Engineering, University of Texas at Austin (USA) 2009
Thesis title: "A multimodel approach to modeling bay circulation in shallow bay-ship channel systems"
M.S., Aerospace Engineering, University of Texas at Austin (USA) 2002
Thesis title: "A Coupled Discontinuous/Continuous Finite Element Method for Hydrodynamic Simulations Using the Shallow Water Equations"

Current Positions

Director of Consulting, Quansight 2019–Present
Senior solution architect for $5M-$8M/yr of client projects involving the open source scientific Python ecosystem. Projects included contributions to core PyData open source libraries, such as Numba, JupyterLab, JupyterHub, Dask, Holoviz and Pytables.

Past Positions

US Army Engineer R&D Center, Associate Technical Director 2016–2019
Technical Program Manager over $8M/yr portfolio of research in CFD, environmental simulation, big data visualization, web-based workflow automation and AI/ML
US Army Engineer R&D Center, Research Computer Engineer 2014–2016
Topics: Workflow Automation, Met-Ocean Modeling, Geospatial Analytics, Visualization of Massive Geospatial Datasets
Texas Water Development Board, Water Informatics Lead 2009–2014
Topics: Led Data, Modeling and GIS Teams, Image Processing, Workflow Automation, Data Publication
Texas Water Development Board, Coastal Modeler/Scientific Software Developer 2003–2009
Topics: FEM/CFD, Hydrodynamics, GIS, Coastal Ocean Studies

Selected Publications

h-index: 4, total citations: 207 (citation statistics: Google Scholar, January 18, 2021)

1. R.P. Signell, **D. Pothina** (2019), Analysis and visualization of coastal ocean model data in the cloud, *Journal of Marine Science and Engineering 7 (4), 110*
2. S.D. Christensen, **D. Pothina**, A. Valoroso, K. Winters (2018), Automated Data Discovery, Retrieval, Manipulation, and Publication using Python, Tethys, and HydroShare, *9th International Congress on Environmental Modelling and Software*
3. **D. Pothina**, P.J.F. Rudiger, J.A. Bednar, S. Christensen, K. Winters, K. Pevey, C. Ball, G. Brener (2018), EarthSim: Flexible Environmental Simulation Workflows Entirely Within Jupyter Notebooks, *Proc.. of the 17th Python in Science Conf. 48-55*
4. H. Huang, Y. Du, **D. Pothina**, J. Matsumoto, S. Negusse (2013), Corpus Christi Bay Three-Dimensional Hydrodynamics and Salinity Simulations Using Finite-Volume Coastal Ocean Model (FVCOM), *Estuarine and Coastal Modeling (2011), 46-65*
5. V. Aizinger, J. Proft, C. Dawson, **D. Pothina**, S. Negusse, A three-dimensional discontinuous Galerkin model applied to the baroclinic simulation of Corpus Christi Bay (2013), *Ocean Dynamics 63 (1), 89-113*
6. **D. Pothina**, A. Wilson (2011), Using Python, Partnerships, Standards and Web Services to provide Water Data for Texans, *Proc of 10th Python in Science Conference. 39-42*
7. C. Dawson, J.J. Westerink, J.C. Feyen, **D. Pothina** (2006), Continuous, discontinuous and coupled discontinuous–continuous Galerkin finite element methods for the shallow water equations, *International Journal for Numerical Methods in Fluids 52 (1), 63-88*

Andreas Christian Müller, Principle Research SDE, Microsoft
email: andreas.mueller.ml+cv@gmail.com

Professional Activities

Core developer and member of the Technical Committee for the machine learning package "scikit-learn". Creator of the "dabl" library for human-in-the-loop data science. Creator of the Python package "PyStruct" for structured prediction. Co-author of "CUV", a C++ and Python interface for CUDA, targeted at deep learning. Contributor to the Python computer vision package "scikit-image". Action Editor, Journal of Machine Learning Research, OSS Track

Awarded Grants:

- *Scikit-learn maintenance and enhancement to gradient boosting and search* (PI). Chan-Zuckerberg $150k. 2019-2020.
- *Extension & Maintenance of Scikit-learn* (PI). Alfred P. Sloan Foundation. $313k. 2017-2019.
- *Analysis and Extension of Scikit-learn* (PI). Bloomberg. $63k. 2017-2018.
- *SI2-SSE: Improving Scikit-learn usability and automation* (PI). NSF. $400k. 2017-2020.
- *Building Blocks and Search Improvements for Automated Machine Learning* (PI). DARPA. $351k. 2018.

Past Positions

Associate Research Scientist, Columbia University 2016 - 2020
Teaching in the Data Science Master program, scikit-learn development Research Engineer, NYU Center for Data Science 2014 - 2016
Development of open source tools for machine learning and data science Machine Learning Scientist, Amazon (Germany) 2013 - 2014
Design and implementation of large-scale machine learning and computer vision applications Ph.D., Computer Science, University of Bonn (Germany) 2010-2013

Selected Publications

h-index: 13, total citations: 4207 (citation statistics: Google Scholar, January 12, 2021)

1. D. Scherer, **A. Müller**, and S. Behnke (2010). Evaluation of pooling operations in convolutional architectures for object recognition. *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*. Springer, pp.92–101.
2. **A. Müller**, S. Nowozin, and C. Lampert (2012). Information Theoretic Clustering Using Minimum Spanning Trees. *Proceedings of DAGM / OAGM*, pp.205–215.
3. A. Abraham, F. Pedregosa, M. Eickenberg, P. Gervais, **A. Müller**, Müller, J. Kossaifi, A. Gramfort, B. Thirion, and G. Varoquaux (2014). Machine learning for neuroimaging with scikit-learn. *Frontiers in Neuroinformatics.*
4. **A. Müller**, and S. Behnke (2014). PyStruct: Structured Prediction in Python. *Journal of Machine Learning Research.*
5. G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, and **A. Müller** (2015). Scikit-learn: Machine Learning Without Learning the Machinery. *GetMobile: Mobile Computing and Communications* 19(1), 29–33.
6. **A. Müller**, and Guido, S. (2016). *Introduction to Machine Learning with Python.* O'Reilly.

Tyler Reddy, Staff Scientist Level 2, Los Alamos National Laboratory

Relevant Experience

Core developer of the SciPy, NumPy, and MDAnalysis libraries. Steering Council member and release manager for SciPy.

Past Positions

Los Alamos National Laboratory, Staff Scientist Level 2 2018 - Present
Topics: build system maintenance; computational geometry
Los Alamos National Laboratory, Director's Fellow 2017 - 2018
Topics: HIV-1 and viruses as bioenergetic containers
Oxford University (UK), Postdoctoral Research Associate 2011 - 2017
Topics: Influenza virion, Dengue Virion, Class III viral fusion proteins characterized with molecular dynamics simulations
Ph.D., Biochemistry & Molecular Biology, Dalhousie University (Canada) 2011
Thesis title: "Structure, flexibility, and overall motion of transmembrane peptides studied by NMR spectroscopy and molecular dynamics simulations"

Selected Publications

h-index: 18, total citations: 3855 (citation statistics: Google Scholar, January 12, 2021)

1. Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Férnandez del Río, Mark Wiebe, Pearu Peterson, Pierre G érard-Marchant, Kevin Sheppard, **Tyler Reddy**, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, Travis E. Oliphant (2020) Array Programming with NumPy. *Nature* 585: 357-362. [Citations: 286]
2. Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, **Tyler Reddy**, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 17: 261-272. [Citations: 2474]
3. **Reddy, T**. and Sansom, M.S.P. (2016) The Role of the Membrane in the Structure and Biophysical Robustness of the Dengue Virion Envelope. *Structure* 24: 375-382. [Citations: 56]
4. **Reddy, T.**, Shorthouse, D., Parton, D., Jefferys, E., Fowler, P.W., Chavent, M., Baaden, M., and Sansom, M.S.P. (2015) Nothing to sneeze at: a dynamic and integrative computational model of an influenza A virion. *Structure*. 23: 584-597. [Citations: 71]

Matt Haberland, Assistant Professor, BioResource and Agricultural Engineering,
California Polytechnic State University, San Luis Obispo

Relevant Experience

Core developer and Steering Council member of the SciPy library. Author of 374 commits, including SciPy's Python implementations of interior-point and revised simplex methods for linear programming and SciPy's interface to the HiGHS linear programming solvers.

Awarded Grants:

- *A Solid Foundation for Statistics in Python with SciPy* (Co-PI). Chan-Zuckerberg Initiative. $200k. 2019-2021.
- *Enhanced LAPACK Support in SciPy* (PI). NumFOCUS. $4.9k. 2019-2020.
- *SciPy Development Documentation Overhaul* (PI). NumFOCUS. $4.2k. 2019.
- *An Efficient, High-Level Implementation of Linear Programming* (PI). NumFOCUS. $2k. 2018-2019.

Past Positions

California Polytechnic State University, San Luis Obispo, Assistant Professor 2018–Present
Topics: Linear programming, numerical linear algebra, statistics computation
UCLA, Assistant Adjunct Professor in the Program in Computing 2014–2018
Topics: Robotic swarm control, body-worn video classification
Jet Propulsion Laboratory, Member of Technical Staff, Associate Level 2007–2009
Topics: created the contact sensor / stabilizer for Mars Science Laboratory rock drill
Ph.D., Mechanical Engineering, Massachusetts Institute of Technology (USA) 2014
Thesis title: "Extracting principles from biology for application to running robots"

Selected Publications

h-index: 9, total citations: 2803 (citation statistics: Google Scholar, January 17, 2021)

1. Virtanen, P., Gommers, R., Oliphant, T. E., **Haberland, M.**, Reddy, T., et al. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 17: 261–272. [Citations: 2474]
2. Li, H., Chen, H., **Haberland, M.**, Bertozzi, A.L., and Brantingham, J.P. (2021) PDES on Graphs for semi-supervised learning applied to first-person activity recognition in body-worn video. Accepted for publication in *Discrete and Continuous Dynamical Systems A*.
3. Qiao, Y., Shi, C., Wang, C., Li, H., **Haberland, M.**, Luo, X., Stuart, A. M., & Bertozzi, A. L. (2019). Uncertainty Quantification for Semi-Supervised Multi-Class Classification in Image Processing and Ego-Motion Analysis of Body Worn Videos. *Electronic Imaging 2019*. 11 (2019): 264-1–264-6. [Citations: 4]
4. Li, H., Feng, C., Ehrhard, H., Shen, Y., Cobos, B., Zhang, F., Elamvazhuthi, K., Berman, S., **Haberland, M.**, and Bertozzi, A.L. Decentralized stochastic control of robotic swarm density: Theory, simulation, and experiment. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017. [Citations: 26]

4 Current and Pending Support

Dharhas Pothina (PI): no current or pending support.

Matt Haberland (co-I) has the following current support:

1.
   - Title: A Solid Foundation for Statistics in Python with SciPy
   - PI: Matt Haberland
   - Program name: Essential Open Source Software for Science
   - Sponsoring organization: Chan Zuckerberg Initiative via NumFOCUS
   - Program officer: Dario Taborelli, (415) 990-0646, dario@chanzuckerberg.com,
   - Performance period: 2/1/2020 to 1/31/2021
   - Total amount received: $69,090
   - Time commitment: 9%
2.
   - Title: Robotic Control of Yellow Starthistle
   - PI: Matt Haberland
   - Program name: ARI Seed Grant
   - Sponsoring organization: CSU Agricultural Research Institute
   - Program officer: Sue Tonik, (805) 756-7241, stonik@calpoly.edu
   - Performance period: 7/1/2019 to 6/30/2021
   - Total amount received: $5,000
   - Time commitment: 2%

and no pending support.

Tyler Reddy (co-I): no current or pending support.

Thomas Fan has the following current support:

1.
   - Title: Improved Interoperability Between Scikit-learn Models and Data-Driven Discovery of Models
   - PI: Hod Lipson
   - Program name: Data-Driven Discovery of Models (D3M)
   - Sponsoring organization: California Institute of Technology Jet Propulsion Laboratory
   - Contact at sponsoring organization: Chris Mattmann, (818) 354-8810, chris.a.mattmann@jpl.nasa.gov
   - Performance period: 05/15/2020 to 03/31/2021
   - Total amount received: $174,74
   - Time commitment: 100%

5 Statements of Commitment

Andreas Müller and Jeff Reback participate in this proposal as part of their open source work on and leadership of scikit-learn and pandas, respectively. Microsoft is providing Müller work time for scikit-learn activities. Reback's contribution isn't directly connected to his employment, therefore he lists NumFOCUS - the 501(c)3 nonprofit with which pandas is affiliated - as participating organization.

Gmail Dharhas Pothina <dharhas@quansight.com>

statement of commitment for NASA proposal

1 message

Jeff Reback <jreback@yahoo.com> Mon, Jan 18, 2021 at 3:43 PM
Reply-To: Jeff Reback <jeff@reback.net>
To: "dharhas@quansight.com" <dharhas@quansight.com>
Cc: Ralf Gommers <rgommers@quansight.com>, Jeff Reback <jeffreback@gmail.com>

> I acknowledge that I am identified by name as Collaborator to the investigation, entitled "Reinforcing the Foundations of Scientific Python", that is submitted by Dharhas Pothina (Quansight) to the NASA funding announcement, and that I intend to carry out all responsibilities identified for me in this proposal. I understand that the extent and justification of my participation, as stated in this proposal, will be considered during peer review in determining in part the merits of this proposal. I have read the entire proposal, including the management plan and budget, and I agree that the proposal correctly describes my commitment to the proposed investigation. To conduct work for this investigation, my participating organization is **NumFOCUS**.

accepted & agreed

Jeff Reback
jreback - Overview

jreback - Overview

jreback has 47 repositories available. Follow their code on GitHub.

Gmail Dharhas Pothina <dharhas@quansight.com>

statement of commitment for NASA proposal

Andreas Mueller <andreas.mueller@microsoft.com> Tue, Jan 19, 2021 at 3:08 PM
To: "dharhas@quansight.com" <dharhas@quansight.com>
Cc: Ralf Gommers <rgommers@quansight.com>

I acknowledge that I am identified by name as Collaborator to the investigation, entitled "Reinforcing the Foundations of Scientific Python", that is submitted by Dharhas Pothina (Quansight) to the NASA funding announcement, and that I intend to carry out all responsibilities identified for me in this proposal. I understand that the extent and justification of my participation, as stated in this proposal, will be considered during peer review in determining in part the merits of this proposal. I have read the entire proposal, including the management plan and budget, and I agree that the proposal correctly describes my commitment to the proposed investigation. To conduct work for this investigation, my participating organization is Microsoft.

Best,
Andreas

6 Budget Justification

This proposal combines 4 established community driven projects: NumPy, SciPy, pandas and scikit-learn. By working together as a team we expect to be able to achieve tighter integration between the technologies and work more efficiently than if each project submitted individual proposals. Specifically, the proposed work involves coordination and cross project infrastructure development that would not be possible otherwise. Hence, the budget being requested is proportionally higher that would typically be requested in an individual proposal.

Additionally, regular contributors to these projects have established and distinct primary institutions. For this reason portions of the budget will need to be subawarded to the primary institutions of the key individuals who are uniquely qualified for this work. The team will be organized according to table 1.

	Effort		
Personnel	Year 1	Year 2	Year 3
Dharhas Pothina (PI)	20%	10%	5%
Tyler Reddy (co-I)	10%	10%	
Matt Haberland (co-I)	12%	9%	9%
Andreas Mueller (unfunded)	10%	10%	5%
Jeff Reback (unfunded)	10%	10%	5%
Thomas Fan	50%	75%	75%
Software engineer - NumPy & pandas	60%	60%	70%
Software engineer - SciPy	40%	40%	40%
Software engineer - CI & packaging	40%	40%	40%
Software engineer - performance	40%	40%	40%
Undergraduate research assistant	20%	15%	15%

Table 1: Table of Personnel and Work Effort. Percentages shown are in terms of full-time equivalent (FTE); 100% is 1840 hours.

6.1 Project and Community Leads

Salary support is requested for PI Dharhas Pothina. He will oversee the coordination of the proposed work between the 4 projects, supervise the Co-I's and manage the engineering team working across the projects.

Salary support is requested for Co-I Matt Haberland through a subaward with the California Polytechnic State University (Cal Poly). He is a SciPy core developer and will deliver the proposed new optimization features for SciPy.

Salary support is requested for Co-I Tyler Reddy. He is a core developer of both NumPy and SciPy, and the SciPy release manager. He will oversee contributions to these projects, as well as perform SciPy releases and extend support for new hardware platforms and Python versions.

No salary support is requested for Andreas Mueller and Jeff Reback, Andreas Mueller has been a scikit-learn core developer for over 10 years, while Jeff Reback has co-lead pandas for the past 8 years. They will oversee the contributions to scikit-learn and pandas, respectively. No time will be charged to this project as their salary is already covered by their primary institutions.

6.2 Engineering Team

Salary support is requested for Thomas Fan. Thomas Fan has been a full-time scikit-learn core developer for the past two years. He will develop the GPU and distributed array support and the zero-copy pandas dataframe support.

Support is also requested to fund undergraduate research assistant(s) as part of the subaward with Cal Poly. They will work on the SciPy project under the supervision of Matt Haberland.

Salary support is requested to bring in new developers to the project by hiring multiple software engineers. Two engineers focusing on cross project: one working on continuous integration (CI) and packaging infrastructure, and the other on performance (speed and memory usage) improvements. A third engineer will contribute to both NumPy and pandas, with a focus on string dtypes and user-defined functions. A fourth engineer will focus on GPU and distributed array support in SciPy.

6.3 Other Direct Costs

No capital equipment, travel or any other direct costs are requested.

7 Facilities and Equipment

All code is hosted on GitHub and tested on freely available continuous integration services. All software development can be executed on standard desktop machines, which are already provided to all engineers by their employer. The server for executing benchmarks is the only additional hardware needed - access to a suitable server for both the project team and other core developers of the open source projects will be provided by Quansight on its existing infrastructure.