Scientific Python Community & Communications Infrastructure

K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Tania Allard, & Melissa Mendonça

Scientific Python and Quansight Labs

July 6, 2022

Contents

1. Summary and scope

Projects in the scientific Python ecosystem have long built infrastructure in an ad-hoc fashion. Much of that infrastructure is for community building & communications: homepages are used to inform of news and releases, documentation & galleries are hosted online, web forums provide a medium for discussion, and various analytics (benchmarking, developer, pageviews) are published regularly. These resources are often constructed haphazardly and poorly maintained. In this proposal we address that problem by supporting existing efforts for improving shared infrastructure, by making such resources more accessible, and by relieving the maintenance burden on project maintainers unrelated to their primary areas of focus. Improvements in such infrastructure will, in addition, benefit global ecosystem outreach and allow us to expand our communities

to include those with skill-sets less represented.

2. *Overview of proposed work*

2.1 Expand and adopt common web themes

There are two web themes commonly deployed by community sites: the *Scientific Python Hugo Theme* and *pydata-sphinx-theme*. Improving them is an easy way to have a large impact on several websites simultaneously. By fostering their adoption, the ecosystem will present a more unified front to users, and reduce maintenance for developers. This grant will enable expanding adoption of the themes, as well as the development of new features such as responsive layout, blogging, and improved accessibility.

Outputs:

1. Add dark mode to Hugo Theme, improve dark mode for Sphinx Theme
2. Improved mobile support
3. Refactor and restructure code base
4. Improve blog functionality for Hugo Theme
5. Improve learn functionality for Hugo Theme
6. Improve code and functionality sharing between pydata-sphinx-theme and the Scientific Python Hugo Theme.
7. Add theme accessibility features identified by the access-centered work stream (see task 2.2)

Outcome 1:

The named themes will be widely usable by the community.

Indicators:

- The Hugo theme is adopted by at least ten ecosystem projects.

Outcome 2:

The ecosystem will feel better unified for users and contributors.

Indicators:

- Users report that ecosystem project websites are now easier to navigate, have a more unified look-and-feel, and are familiar across the board.

Estimated effort level: 1.55 FTE/year

2.2 Adopt and promote access-centered practices

Improving the accessibility of online resources can have a significant impact on their usability as well as broadening community participation and inclusion. The *Scientific Python Hugo Theme* and *pydata-sphinx-theme* are an excellent place to introduce accessibility standards and establish accessibility best practices for the broader ecosystem. We will accelerate work to develop such best practices for project maintainers and contributors and to propagate those across projects. We aim to achieve this by developing access-centered documentation and contribution guidelines, organizing online workshops, and working with other maintainers to improve the accessibility of their project's documentation and homepage (either through direct code contributions or by running accessibility clinics with them).

A set of access-centered practices will be written up as a SPEC document. For core projects to endorse and to provide guidance for those projects that we would not be able to support directly.

Outputs:

1. (year 1) Run accessibility audits for the Scientific Python Hugo theme and the pydata-sphinx themes.
2. (year 1) Ensure that the Scientific Python Hugo theme and the Pydata-Sphinx themes meet the applicable *WCAG accessibility guidelines*.
3. (year 1 - stretch goal) We may also consider the recommendations of the newly-founded W3C cognitive accessibility working group where relevant (this is a newly created initiative and very much in its infancy).
4. (year 1) Adopt an accessibility statement for both themes clearly outlining the to-date status of WCAG compliance.
5. (year 2) Generate a set of access-centered practices for the scientific community focused on generating *accessible content* (this might cover cases such as documentation, project homepages, interactive demos, and/or tutorials). This should include accessibility best practices for examples developed through section 2.5 in this proposal.
6. (year 2) Organize 3 or more accessibility workshops and/or clinics for the broader community.
7. (year 2 - stretch goal) Work with the disabled scientific community to identify and apply improvements as needed to improve theme usability instead of just working towards WCAG compliance.

Outcome 1:

The Scientific Python Hugo theme and the Pydata-Sphinx themes

are accessible to the following groups: users with low vision (including color blindness) and screen reader users.

Indicators:

- Applicable *WCAG success criteria for code* are met at an A or AA standard (this varies on the specific items and success criteria)
- Applicable *WCAG success criteria for design* are met at an A or AA standard (this varies on the specific items and success criteria)

Outcome 2:

Project maintainers, teams and contributors start adopting access-centered practices in their day-to-day maintenance, development, and content creation workflows.

Indicators:

- Number of projects that have adopted accessibility-related contributing guidelines or "good enough" practices for anyone without formal accessibility training to follow and implement.
- Significant increase in accessibility, access-centered, and disability inclusion conversations, issues opened and fixes across the ecosystem.

Estimated effort level: 1.35 FTE/year

2.3 Measure community involvement and project health

To better understand how developers and users engage with development, we propose building an analytics dashboard that summarizes activities, highlights important issues, and measures response times. This will help us better connect with developers, and gauge how approachable projects are to new volunteers. A *rough prototype* exists, but it requires focused effort to develop and to roll out to more projects.

Outputs:

1. Create a development analytics dashboard.
2. Establish a mechanism for generating weekly reports, and for storing the report data online.
3. The dashboard is hosted for at least five projects in the ecosystem.
4. Provide and document mechanisms for hosting the dashboard, and for modifying it to, e.g., include additional metrics and analyses.

Outcome 1:

We have a better understanding of how developers engage with projects; e.g., how long it takes to respond to issues, how often we fail to discuss or merge PRs, and how many developers we are able to engage longer term.

Indicators:

- In discussion with projects, we find that they changed or improved some aspect of development based on information provided by the dashboard.
- As a community, we are able to identify when specific projects experience difficulties such as rapid development slowdown.
- Projects contribute their own metrics & analyses to the dashboard.

Outcome 2:

Projects are better able to assess and provide evidence for their maintenance burden, which can be used to motivate funding and for reporting progress.

Indicators:

- One or more projects use information from their dashboards in a grant proposal or progress report.

Estimated effort level: 0.75 FTE/year

2.4 Translate content to involve wider audiences

Development takes place in English, which is reflected in project websites and documentation. While many contributors are comfortable with English as a first or second language, the language barrier excludes especially users that are very young, new to the community, non-English-native speakers with learning and cognitive disabilities, or from the Global South—all potential future contributors! Translation of all hosted content is infeasible, but landing pages, short tutorials, and contribution guides comprise little enough material that translations can be maintained by a small cross-project team. We will continue to provide theme infrastructure for translation, and increase efforts to translate the primary pages of at least eight core projects.

Outputs:

1. (year 1) provide well-documented translation workflows for project websites using the Scientific Python Hugo Theme.
2. (year 1) translations into 3 or more other widely used languages (e.g. Spanish, Chinese, Japanese) of the websites of at least 8 core projects launched.

3. (year 2) Provide well-documented translation workflows for narrative content in documentation sites using pydata-sphinx-theme.
4. (year 2) translations into 3 or more other widely used languages of the main beginner-focused tutorial(s) of at least 4 core projects launched.

Outcome 1:

A growing amount of content in multiple languages is available to end users of core Scientific Python projects.

Indicators:

- Number of project websites offering translations
- Number of languages content has been fully translated to, on average per project
- Number of documentation sites offering translations
- Total number of tutorials across all projects translated into >= 3 languages

Outcome 2:

Project maintainers and teams start seeing it as standard practice to offer translations of high-level content, and projects have separate translation teams.

Indicators:

- Number of projects that have established a translation team
- Number of languages that are maintained by a translation team, on average per project

Estimated effort level: 0.4 FTE total — 0.15 FTE/year for translation infrastructure development and maintenance; 0.25 FTE/year for translation coordination & actual translation work.

2.5 Improve utility of documentation through interactivity

Interactive project documentation has the potential to engage less experienced users, lower the barrier to entry for projects, and can be a valuable resource for schools to teach ecosystem libraries. The Binder project showed the value of this approach, but cannot sustain the loads placed upon it by ecosystem-wide documentation. Instead of using cloud infrastructure, it has recently become possible to host computation inside readers' web browsers through JupyterLite or PyScript. We will work with the developers of these tools to integrate

them seamlessly into project documentation, and to accelerate their adoption. We will make Pyodide, the underlying technology, a supported platform of Scientific Python core projects through build system & packaging improvements. We will also implement and propagate a best-practice pattern for serverless, low-maintenance demonstration widgets in project websites.

Outputs:

1. (year 1) add support for making documentation examples interactive in pydata-sphinx-theme
2. (year 1) make all the examples in the API reference guide and tutorials of at least 1 project interactive via JupyterLite or PyScript (available to end users)
3. (year 2) propose and develop a SPEC on how projects can make their documentation interactive via JupyterLite or PyScript,
4. (year 2) expand the CI support for cross-compiling to Pyodide/WebAssembly to at least 5 projects
5. (year 2) expand the availability of interactive documentation examples and tutorials to at least 5 projects

Outcome 1:

WebAssembly has become a widely supported target for Scientific Python projects. Most or all library functionality is now available through JupyterLite and PyScript, and remaining limitations are at least understood.

Indicators:

- Number of projects with CI support for Pyodide/WebAssembly
- Number of projects with an interactive demo widget integrated into their website

Outcome 2:

Scientific Python projects provide interactive html documentation, making it easier to learn about and try each project.

Indicators:

- Number of projects with interactive html documentation.
- Increased usage of interactive documentation by visitors to project websites.

Estimated effort level: 0.95 FTE/year

2.6 Assist with release management to relieve maintainer burden

To put all of the above into the hands of the community, and to aid with adoption, we will help make releases for several ecosystem packages.

Outputs:

1. Make one or more releases for at least five different ecosystem projects.
2. Improve release tooling (cibuildwheel, github-activity, etc.).
3. Better coordinate releases and releases of pre-requisites such as themes, numpydoc, etc.
4. Improve cross-project testing.

Outcome 1:

Projects release regularly; releases are stable, work well together, and users quickly gain access to new features.

Indicators:

- Releases break other packages less frequently and users report fewer breakages after upgrades.

Outcome 2:

Releases are easier to make for developers. They have the necessary tools and testing frameworks at their disposal.

Indicators:

- Releases take less time to make and occur more frequently.

Estimated effort level: 0.55 FTE/year

Additional elements

How does this work build on existing efforts from this group?

For the past year, staff from Quansight and the Scientific Python project have collaborated on refactoring and improving the numpy.org theme for wider use. It is currently deployed on, e.g., scipy.org, scientific-python.org, and blog.scientific-python.org. We have also developed additional functionality, which is now often requested by projects for their own sites.

Jarrod Millman is the release manager of numpydoc, which generates inline API reference documentation, and the pydata-sphinx-theme. He oversees development of the Scientific Python Hugo theme and the prototype community developer analytics dashboard.

Chris Holdgraf is the lead developer of the pydata-sphinx-theme.

Isabela Presedo-Floyd and Mars Lee have been championing accessibility efforts in the ecosystem, hosting several alt-text sprints and starting work on an ecosystem alt-text SPEC, with support from Jarrod Millman. This work is currently largely unfunded.

Ralf Gommers organized the initial development of numpy.org, presided over translation efforts, and is responsible for NumPy/SciPy build system & packaging improvements.

Stéfan van der Walt is the architect of the Scientific Python Hugo theme, and along with Jarrod has built community calendars, code highlighting, team galleries, web analytics, and YouTube videos with transcripts. He co-authored the first version of numpydoc, and led an early Google Summer of Code project to implement proof-of-concept interactive documentation for scikit-image.

Tania Allard implemented Gitpod support for multiple projects to lower the barrier to contribution. She has also guided the efforts to improve accessibility in JupyterLab, including formally establishing the Jupyter Accessibility Software subproject, and initial accessibility awareness and alt-text improvements for NumPy, and other core Scientific Python projects.

Melissa Mendonça leads a small team of Contributor Experience Leads across NumPy, Pandas, Matplotlib, and SciPy. She is also the NumPy Documentation Team coordinator.

Scientific Python: projects involved

The Scientific Python project exists to better coordinate the scientific Python ecosystem. The ecosystem, by nature, is a loose federation of projects that tend to have similar community and development practices. Projects are typically controlled by those who have invested a lot of time in them, and as such we cannot guarantee involvement of any specific project.

The goal of supporting the entire ecosystem is ambitious and won't be achieved immediately. In order to make things manageable, we started by creating a framework (SPEC—Scientific Python Ecosystem Coordination) for developing community guidelines that, over time, will be endorsed and adopted by a wide range of projects. In this framework, endorsement of ideas and mechanisms starts with the so-called *core projects* (well established, older projects, relied on by many others), the developers of which we have good relationships with. Once those projects are on board, it is likely that surrounding

projects will follow. The SPEC work is funded by the Moore grant.

Over the last year, we also did unfunded work around web infrastructure and accessibility. We've made some initial progress, but funding is needed to continue these efforts.

While we will continue working with well established, older projects that know us better, we hope to engage more projects over the next several years. We will use our social media platforms and in-person meetings, such as SciPy 2022, to publicize our initial efforts and larger vision and to recruit more projects and volunteers.

Impact and value to the biomedical community

Scientific Python has had a significant impact on scientific and research computing. For example, NumPy, Pandas, SciPy, Matplotlib, scikit-learn, scikit-image, and NetworkX are used in virtually every area of science and engineering. They are also used in biomedical tools and applications such as Napari, QIIME, Qiita, MNE, scikit-bio, and GALA. Several projects are used in the data analysis pipelines of large collaborations like the Human Cell Atlas and the International Brain Laboratory.

The estimated size of the user base, based on PyPI and Anaconda data, ranges from 15,000,000–25,000,000 for NumPy and Pandas to 1,000,000–5,000,000 for the two scikits and NetworkX. The citation count for the SciPy 1.0 paper (2020) is >11,500, and for the two NumPy papers (2011, 2020) >15,000. The scikit-learn paper (2011) tops even that with >59,000 citations.

An analysis of Bioconda, one of the largest collections of biomedical and bioinformatics software, shows that roughly 25% of the ~9,400 packages depend on Python, 10% on NumPy, 7.5% on Pandas, and 5% on SciPy. This is but one indicator of the significant impact scientific Python has on biomedical software and research.

The work targeted by this grant is broad and general, so while it does not target biomedical applications *per se*, it extends its benefits directly to that significant part of the biomedical research community that relies heavily on scientific Python. A unified, accessible, well documented ecosystem with regular releases becomes a better basis for open, reproducible biomedical research. A clear indicator of the utility of our work will be the adoption of SPECs, themes, and other outputs by biomedical packages.

Diversity, equity, & inclusion

The scientific Python community is working on improving opportunities for technical collaboration amongst a wider set of participants. One approach is to create project roles that are attractive to a more diverse pool of participants, such as the web community.

Accessibility and translation efforts will improve end user diversity, especially.

By improving communications and outreach, we will engage a larger audience of newcomers, necessary to improve community inclusion.

Risks and mitigation strategies

Because we are strengthening existing efforts, risks are relatively low. The biggest challenge is being able to find the right blend of contributors, who both have the necessary skills but also understand how our communities function. Because the web and devops communities are so large, we do not anticipate this to be a problem.

We will be touching infrastructure across the entire ecosystem. Since such changes have to be vetted by the community, we can never guarantee that they will be accepted, although our existing track record proves that projects are receptive to assistance. We can also not guarantee that projects will make use of shared infrastructure, but gains to developers are significant enough to make it likely. Moreover, since there are a large number of projects that could adopt our work, we are confident that some will embrace our efforts.

In terms of individual deliverables, interactive documentation carries more risk than others, because Pyodide, JupyterLite, and PyScript—the underlying technology—are not yet proven, and CPython itself has only recently added WebAssembly as a supported platform. That said, the Python community is invested in this effort, and we will work with both the JupyterLite and PyScript teams to address technical challenges.

As with any community project, additional or changed needs may be uncovered during the development process. Our team is agile in its response, and will re-prioritize needs to best serve the community.

Sustainability strategy

Most of the work done on themes will, once completed, require relatively little maintenance. Work on accessibility is aimed not only at providing projects with the required changes, but also educating

them on how to make those changes themselves. The improved communications and outreach will increase community participation, and should lead to volunteers stepping up to support our work. By publishing SPEC guidelines we make it easy to replicate our efforts, so that our team is not required for it to happen. Both the Scientific Python group and Quansight have a track record of long-term involvement in the community. We do not foresee changing our existing efforts to seek funding, to apply those funds for the improvement of the community, and to share what we learn along the way.