# Revamping Matplotlib for Modern Data Structures

Thomas A. Caswell

## Contents

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

### 1.1 Matplotlib's role in scientific computing

Matplotlib[1] is a comprehensive Python library for creating static, animated, and interactive visualizations. It provides the visualization for projects across all SMD divisions, including flagship missions like the Hubble Space Telescope (HST) and the James Webb Space Telescope[2] (JWST), and instruments like the Curiosity Rover Mastcam[3].

Matplotlib is part of the Scientific Python Ecosystem (SPE), a loosely defined community of projects and programmers with the common goal of advancing science through the use of the Python programming language. This development is largely volunteer work or work that is sponsored implicitly by specific science projects. SPE, shown with a rough schematic in Figure 1, has a core of general purpose domain-agnostic tools, like NumPy[4] and SciPy[5]. Matplotlib is in the next ring of specialized, but still domain agnostic tools, and is the most prevalent data visualization library for the SPE. The outer rings are increasingly domain-specific tools, like AstroPy[6,7] for astrophysics, SunPy[8] for heliophysics, and MetPy[9] and cartopy[10] for earth sciences; all of these use Matplotlib for specialized plotting. This layered approach gives scientists and engineers convenient and powerful high-level tools while enabling direct access to the underlying libraries when needed.

Soon after its start by John D. Hunter in 2002, Matplotlib began growing as an open source plotting library with a BSD-compatible license. The first commits in the Matplotlib history date to early 2003. Over the last 17 years Matplotlib has been actively developed and maintained by a vibrant, primarily volunteer, community. Matplotlib has over 1,250 individual contributors to the code base, with many more individuals having contributed in ways not easily tracked, such as answering user questions on the mailing list and reporting issues.

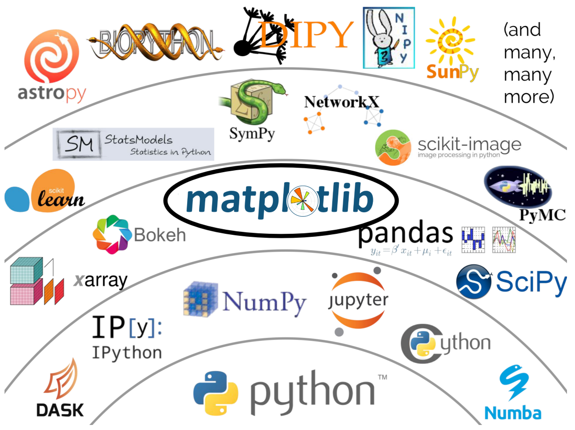


**Figure 1:** A schematic of the Scientific Python ecosytem. At the core we have the Python language itself with concentric rings of domain agnostic to domain specific libraries. Both AstroPy (top left) and SunPy (top right), used in the astrophysics and heliophysics divisions respectively, rely on Matplotlib (center, circled) Credit: Jake van der Plas, "The Unexpected Effectiveness of Python in Science", PyCon 2017

Matplotlib has a high-level API for quick plotting, such as in exploratory data analysis, plus a low-level API that gives full control for fine-tuned publication-quality plots, for animations, for writing both GUI-independent and GUI-specific tools for interactive data plotting, and for output to a variety of vector and raster formats including `svg`, `eps`, `pdf`, and `png`.

Matplotlib's generality and flexibility make it readily extendable to many data visualization domains. A rich ecosystem of downstream packages use Matplotlib as their base, providing domain-specific streamlining of Matplotlib's API: including yt[11], AstroPy[6,7], ArviZ[12], seaborn[13], xarray[14], cartopy[10], corner.py[15] and mpl-scatter-density[16].

Matplotlib is ubiquitous in scientific computing. Conservative estimates based on downloads and on usage of the documentation website indicate that Matplotlib has over a million users. We expect that a large fraction of NASA SMD-sponsored projects rely on this shared infrastructure to some degree, but this is hard to quantify. It is not usual to cite tool chains in scientific literature, so citation counts are likely to vastly under-represent the scientific use of Matplotlib. For example, a canonical reference for Matplotlib[1] has over 12,750 citations overall and over 4,500 citations in ADS[17], while a common reference of NumPy[18] has over 6,800 citations. These counts illustrate the problem in measuring usage via citations: Matplotlib depends on NumPy, yet Matplotlib has almost twice as many citations.

## 1.2 Objectives

Matplotlib is a primarily community-driven project, but we have grown to the point where we need supported developers with the time to organize, plan, and make decisions. Supported developers will allow us to take on more complex projects than can be implemented by part-time volunteer effort alone, while executing critical day-to-day maintenance tasks required to keep the project healthy. **We propose to split the effort on this grant equally between the two primary activities: overhauling our internal data representation, and library maintenance**.

### 1.2.1 Refactor the Internal Data Model

Matplotlib is based on visualizing data stored in Numpy arrays, which is a simple yet flexible data model that has been the underpinning of data visualization for the last few decades. However, new data types are being developed and standardized to carry structure and metadata along with the numerical values of the data. Physical units are a key example of such metadata that a plotting package should handle gracefully. Modern data structures can also describe massive data sets and allow performing operations without loading the whole data set into memory (i.e. dask, xarray); Matplotlib should leverage these packages to work efficiently with Big Data.

Eliding many of the implementation details, Matplotlib can be thought of as having three primary layers[19]: the user-facing plotting methods; the `Artist` layer; and the backends. The core of the architecture is the set of `Artist`s which can be understood as objects that describe something for a backend to render to the output device. Each individual `Artist` is responsible for holding the user data and style information for part of the final visualization, but the exact data held varies among `Artist` classes. For example, `matplotlib.text.Text` is responsible for drawing text and holds, among other things, the location of the text, the font family and size, the color, and the actual string. Internally Matplotlib represents a `Figure` as a tree of `Artist`s. The plotting methods in Matplotlib create these objects and add them to the tree. To render the final output Matplotlib walks the `Artist` tree depth-first and each each `Artist` is responsible for rendering itself via the backend. Thus, by changing the backend we can change the output format from raster to vector without making any changes to the `Artist` tree.

Currently, each plotting method and `Artist` independently handles sanitizing and storing data. Hence, each `Artist` subclass holds its data in a different way and some common functionality, such as handling data with attached units (e.g., degrees Celsius, dates), is repeated throughout the code base. This leads to inconsistencies across the library and makes it difficult for users to write code that updates interactive explorations or animations. Another consequence of the data storage is that Matplotlib also cannot exploit “Structured data”–combining multiple pieces of (possibly heterogeneous) data with labels and meta-data into single data structure; instead, users must extract data elements and reassemble them as arguments to Matplotlib plotting functions.

We have on-going theoretical and design work* to re-architect Matplotlib to better separate the data representation from the rest of the library. We are developing a consistent data access API such that each `Artist` will have a single `DataSource` object responsible for providing access to all of the data that the `Artist` needs. This design will allow us to overcome many of the issues with the current design, opening the door to new functionality. The current work is expected to produce design documents and a proof-of-principal implementation by the end of 2021. Based on this preliminary work **we propose refactoring all of Matplotlib to use the new data model**. The proposed design is consistent with the rest of the architecture; just as the `Artist`s are currently agnostic to the backend used, the `Artist`s will be agnostic to the internals of its `DataSource`.

An immediate motivation for this refactor is to improve Matplotlib’s handling of data with physical units attached. Units are fundamental to science and engineering, but most numerical software is unit-naive. It is the user’s responsibility to convert data to consistent units prior to any computation. Failing to do so leads to the most insidious of bugs: everything “works” but silently gives incorrect results! There are several libraries which provide unit-aware data structures in Python that can be used with Matplotlib for unit-aware plotting. **This capability is currently used to support spaceflight operations by Monte, JPL’s mission design and navigation software system**; NASA supported its initial development in Matplotlib.

The current support for physical units in Matplotib is based on inspecting the incoming data type. When the the data structure supplied by the user includes units, Matplotlib looks for a registered converter and sets default axis labels, limits, and functions for locating and formatting the ticks. Internally, Matplotlib uses the unit machinery to handle datetime and string-categorical types. Third-party libraries are able to register their own converters. With this machinery users can control the units in which the data is plotted, which may differ from the units in which it was supplied. Axis limits may be specified in physical units.

There are several problems with the current unit support in Matplotlib. It is under-documented, making it hard for users to understand and use, and hard for developers to extend. Test coverage is incomplete, and inconsistencies have accumulated in the code base. These range from subtle inconsistencies between plotting methods to data with units simply not working with some methods. **To address this we propose to use unit-handling as a motivating feature while doing the `DataSource` refactoring**. Unit logic will be consolidated in the `DataSource`.

There is a broad recognition across the SPE of the importance of supporting physical units as an intrinsic component of computation. There is on-going work in NumPy (NEP40-43), pandas, and xarray to add support for physical units to their foundational

*Supported by the Chan Zuckerberg Initiative, see current support.

data structures. The proposed work will ensure that Matplotlib can take advantage of such efforts, for seamless handling of data with units from computation to visualization.

In addition to fixing problems with the current support of physical units, this refactoring will lay the ground work for

- smart down-sampling of plotted data based on view limits,
- native consumption of structured data,
- seamless updating of the underlying data, either interactively or via streams,
- and use of alternative data sources such as out-of-core data structures, database queries or analytic functions.

### 1.2.2 General maintenance

While rewiring Matplotlib's data model, we need to sustain Matplotlib as a vibrant project and continue to support the hundred of thousands of users and hundreds of downstream packages. To maintain Matplotlib's health we must:

- fix critical bugs and regressions promptly as they are reported;
- make improvements while maintaining backward compatibility and documenting changes;
- categorize Issues and PRs in terms of topic, difficulty, and urgency;
- entrain new contributors to sustain and diversify the community developer team;
- maintain the continuous-integration infrastructure;
- manage the release process;
- and manage community discussions about proposed enhancements, features, and API changes.

These are tasks that are never "done"; users find new ways to use the library, they expose previously un-detected bugs, they request or propose new features, and the world continues to changes around us. **Dedicated full-time developers help ensure that things happen in a consistent and timely manner, and provide the crucial roadmaps that will steer the future of the library**.

Historically, pull requests (PRs) and issues have been submitted faster than they can be reviewed; Matplotlib has accumulated about 300 open PRs and 1300 open issues due to this imbalance. There are critical bug reports and insightful feature requests among the issue backlog, while among the PR backlog there are useful contributions or bug fixes that would improve the libraries for direct users and downstream packages. The large backlog is discouraging for new and occasional contributors and distracting for core developers.

In 2020, with a grant from CZI supporting a Research Software Engineer (RSE) working on Matplotlib for 10 month, we resolved 2,056 PRs and 999 issues. However, even at this rate we only reduced the PR backlog by 70 open PRs and held the issue backlog even. As part of reducing the backlog, we were able to fix several long-standing bugs. We have demonstrated that supporting maintenance work has a positive effect on the health of the project and **the additional resources requested here will further reduce, but not eliminate, the maintenance backlog**.

## 1.3 Impact and Relevance to Science Mission Directorate

As mentioned above, it is hard to estimate the prevalence of open source software due to it being freely distributed and not systematically cited. Matplotlib is in the top 100 most-downloaded packages from PyPI[20] and is packaged by every major Linux and scientific Python distribution. Another metric to measure impact is static analysis of publicly available code. According to github over 299k repositories and over 12k packages hosted on GitHub depend on Matplotlib[21]. A recent study of the almost 10M Jupyter notebooks on github found that over 3M of them have a direct Matplotlib import[22].

These measures show broad impact but cover many uses outside of the SMD community. If we restrict the static analysis to https://github.com/nasa, NASA's official organization on GitHub, we see that 53 of the 330 hosted repositories make reference to Matplotlib. Soliciting on social media for examples of Matplotlib in SMD research yielded a (non-exhaustive) list including:

- to study thunderstorms[23,24], seasonal ocean winds[25] and tropical storms[26];
- in the first science paper from the Parker Solar probe[27];
- in the Martian science program in both orbiter[28] and rover[3] contexts;
- as part of ground operations from the Mars Phoenix Lander;
- with data from Kepler and K2 missions to study Trojan asteroids[29] and Titan[30,31];
- on the New Horizons Kuiper belt extended mission[32];
- for visualization of scheduling, safety and constraint checks, and telemetry by the Swift science operations team[33,34];
- as part of the HST and JWST[2] data processing pipelines;
- in Monte, JPL's mission design and navigation software system to support spaceflight operations;
- in fundamental research on graphene[35];
- and for work on nuclear powered rockets[36].

This list shows the breadth of Matplotlib usage across the SMD divisions. Matplotlib falls under the umbrella of projects which the National Academies of Science recommended for funding in the report "Open Source Software Policy Options for NASA Earth and Space Sciences (2018)"[37] (Option B4). **Small investments in Matplotlib will have returns across the entire SMD portfolio**.

Maintaining Matplotlib's health and engineering for its future will have wide-ranging impacts for the scientific community, and for NASA in particular. As pointed out above, Matplotlib is the plotting layer for much of the Python ecosystem. It has been used in thousands of academic papers, plotting for news and sporting media, technical dash-boarding, and most modern data science applications. As of 2018Q2 over a third of papers published to astro-ph on the Arxiv contained a figure generated with Matplotlib[38], with a clear upward trend. Matplotlib has also been used in a number of high-profile astronomy projects including the Nobel prize-winning observation of black hole merger by the Laser Interferometer Gravitational-Wave Observatory (LIGO), the 2020 Nobel prize-winning discovery of a super massive compact object at the center of our galaxy and the recent observation of Sagittarius A* by the Event Horizon Telescope (EHT).

This proposal is another step towards Matplotlib's sustainability, and ability to move forward. Given the usage and the scale of the library it is not sustainable to continue to maintain Matplotlib as a volunteer-only effort. A core team of full-time developers and managers will better co-ordinate and nurture volunteer efforts, with the goal of growing and sustaining a diverse community of volunteer expert contributors.

Finally, with explicit support for their time, Matplotlib's core developers will better be able to plan the project's future. This requires vision, co-ordination with the rest of the scientific Python community, including downstream libraries, and getting out in the community to get feedback. It is very hard to provide that leadership and vision while working a "day job".

## 1.4 Risk Management

There is minimal risk in the maintenance work. There is a large volume of individually small tasks that need to be addressed: the issue and PR backlogs. Any dedicated effort devoted to clearing the backlog will have a positive impact on the project. We have demonstrated that supporting developers for maintenance work is an effective and efficient use of resources.

The data pipeline refactor carries more risk. Given the large user base it is imperative that changes are as backwards compatible as possible. While this large user base is on one hand a lever that multiplies the value any improvements, it also makes changes more expensive. We cannot simply change our public interface to track the internal refactoring, instead we must also write code to translate the old API to the new implementation. This engineering work makes the refactor more challenging and time consuming than starting from scratch. We will rely on, and expand, our extensive test suite to catch any unintentional changes early in the development process.

There is also the risk than we are underestimating the amount of work that will be required. There is always a possibility that during refactoring the scope of work will rapidly snowball due to unexpected internal dependencies or edge cases. However, even in the event that we cannot complete the implementation of a new data pipeline model within the performance period, throughout the process we will be working towards improving library internals and improving our documentation.

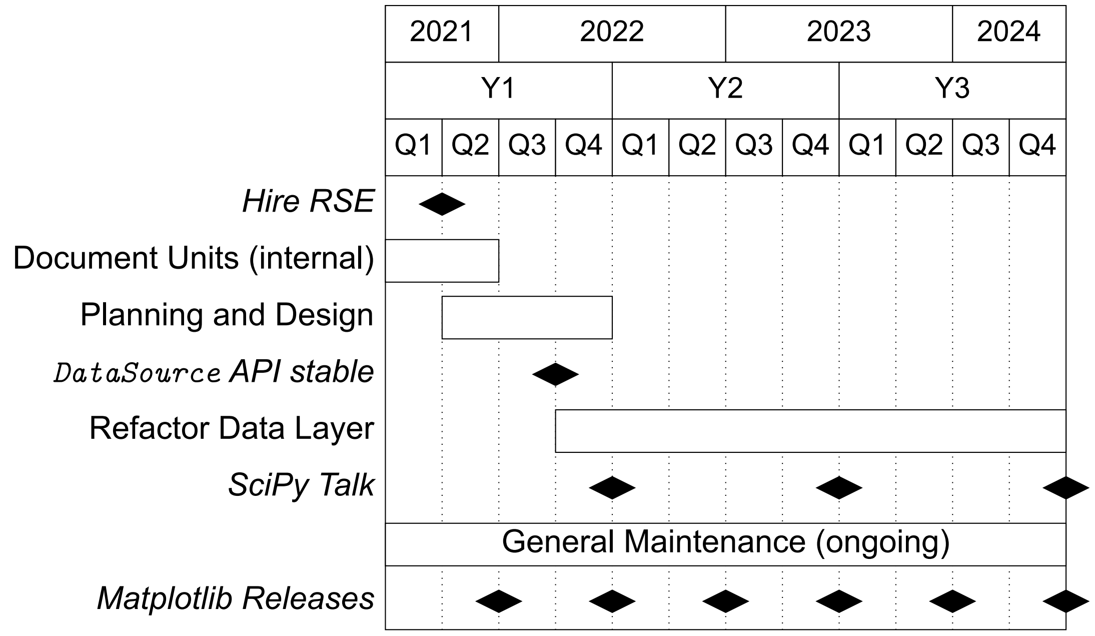
We anticipate that all of the work can be done incrementally and will be merged to the default branch of Matplotlib throughout the performance period. Frequently merging incremental work reduces the risk, and is consistent with our standard development process. Work done under this proposal will be reviewed in our normal fashion, and many small PRs are easier to review and merge than a single large PR. Large PRs also quickly drift out of sync with the master branch, and develop insurmountable rebase requirements. The improvements, both to the code and documentation, will be released to users as part of our standard semiannual release cadence.

There is personnel risk in that we have not yet identified the RSE who will be a key individual for carrying out this work. We are optimistic that we will be able to recruit a qualified individual in reasonable time. In early 2020 we ran a similar search for a 1 year position and had a number of qualified candidates, one of whom is currently employed via NumFOCUS as a RSE for Matplotlib.

## 1.5 Contributions of Principal Investigator and Key Personnel

Dr. Thomas A. Caswell has the sole individual responsibility for directing and supervising the execution of this work. He has extensive experience developing and maintaining Python libraries for scientists. He has been involved with Matplotlib from 2012 and had a leadership role in the project from 2014. He is also a core developer of h5py and has contributed to many of the other core projects of the SPE. At Brookhaven National Laboratory he is the lead architect and developer of the Bluesky Suite, an ecosystem of co-developed libraries and applications for data acquisition and management, which is being adopted for beamline operations at synchrotrons in the US and across the world.

## 1.6 Work Plan



### 1.6.1 General Maintenance

Throughout the project both Dr. Caswell and the RSE will work on Matplotlib general maintenance and community engagement. This includes, but is not limited to, fixing bugs, reviewing Pull Requests, answering user questions, welcoming new contributors to the project, and planning for the future. Dr. Caswell will particularly focus on the project roadmap and community governance.

### 1.6.2 Data Model Refactor

***Document Existing State of Data Pipeline*** We will document, both at the conceptual and API level, how the unit dispatch system currently works in theory and practice. Given the importance of backwards compatibility it is critical to understand what we are refactoring before beginning. This documentation will be aimed at Matplotlib developers and focus on the low-level details. Unfortunately, the original authors of the unit-handling code are no longer active with the project so we may have to reverse engineer some aspects of the code and motivation. This documentation will be used in further planning and as a reference for what is “correct” behavior when resolving inconsistencies.

***Refactor Planning and Design*** The data pipeline refactor is a critical part of the library that touches most artists, so we require a comprehensive plan. As part of developing this plan we will reach out to upstream data-structure libraries, such as pandas, numpy, xarray and the Consortium for Python Data API Standards [†], to downstream libraries, such as astropy, yt, and sunpy, and to users of the unit machinery, such as `unyt`[39], `pint`[40], `astropy.units`[6,7,41], and the Monte developers at JPL. This will be a unique chance to change the internal architecture of Matplotlib and we need to ensure that we engage with all stakeholders.

***API Design*** The core of the refactor will be a newly developed `DataSource` interface. This will provide a consistent interface between the `Artists` and their data. This interface, and the ability to provide alternate implementation of data storage and access, is what will allow the features discussed in section 1.2.1. This builds on our theoretical and proof-of-concept work that will be finished by the end of 2021. Active work on both projects will overlap and we will actively engage to ensure that the theoretical work is effectively translated. We will fully specify the `DataSource` interface and a write a reference implementation by the end of Y1Q3. Concurrently we will developed a detailed schedule to refactor the approximately 100 `Artist` classes in Matplotlib to use the new data model.

***Refactor Data Layer*** As mentioned above, it is critical that we maintain as much backwards compatibility as possible in the API changes needed for the refactoring. The variation among `Artist`s is both what we are trying to address and the biggest complicating factor. Each class well need to be handled on a case-by-case basis. We anticipate that there will be significant collateral work, such as writing additional tests to ensure that we do not accidentally break current behavior, and implementing back-compatibility shims. As we perform this refactor we will also add tests that each `Artist` and plotting method handles data with units correctly **This work will impact a majority of the library** and is the largest proposed change to the library in over a decade.

This whole process will be extensively documented, both inward facing, so that future Matplotlib developers understand why decisions were made, and outward facing so that users can make the most of the new and improved functionality.

A secondary goal is to extend the unit machinery to improve the user experience. Currently much of the mechanism of unit-handling is implicit and automatic based on type inference. We propose to extend the interface to allow optional direct control of the unit machinery by users and third-party library developers.

### 1.6.3 Work Process

The Research Software Engineer (RSE), yet to be identified, will devote 100% FTE to this proposal. They will split their time evenly between general maintenance and the data model refactoring. If the refactoring is completed faster than expected, the RSE will devote 100% of their effort to general maintenance. Dr. Caswell will devote 25% FTE to this proposal. This time will be split between general maintenance (10%), the data model refactor (10%), and management of the grant and supervision of the RSE (5%).

All work under this proposal will undergo the same review process as any other contribution. We encourage many small PRs to reduce risk. At least one maintainer not supported on this proposal will need to sign off on any changes before they are merged. This disseminates understanding of the proposed work and ensures community buy-in.

[†] https://data-apis.org/

Finally, by merging changes to the default branch continuously throughout the period of work the incremental improvements will be released as part of our semiannual release cadence.

The RSE will present the status of the work at two conferences yearly, expected to be SciPy in July and one of PyData regional conferences in the winter.

#### 1.6.4 Grant Management

Dr. Caswell of BNL and the Matplotlib Lead Developer is the PI of the proposed development. He is responsible for the quality and direction of the proposed work and the proper use of all awarded funds. He is also responsible for all management, and budget issues and is the final authority for this task.

## 1.7 Matplotlib Project Management

### 1.7.1 Governance and Finances

Matplotlib is a NumFOCUS Fiscally Sponsored Project.

Matplotlib is governed by a steering council, a Project Lead (PL), and several Deputy Project Leads (DPL)[42]. We are a community project and try to make all decisions, both strategic and day-to-day, by consensus. However, if consensus can not be reached, the responsibility falls back to the DPLs and ultimately to the PL. The current PL is Dr. Caswell.

As Matplotlib has matured we have moved from a single person, originally Dr. John Hunter, to a “BDFL” governance model. The leadership of the project passed to Dr. Micheal Droettboom in 2012. In 2014 Dr. Caswell and Dr. Droettboom were co-leads and in 2016 Dr. Caswell took over as the sole project lead. In 2020 we formally adopted the current governance model.

### 1.7.2 License

Matplotlib is licensed under the Matplotlib License[43] which is a derivative of the Python Software Foundation license[44] and is a permissive license compatible with the BSD license[45].

### 1.7.3 Sustainability Metrics

There are many possible ways to measure the sustainability of an open source project. We propose to focus on growing our developer community, reducing our Issue and PR backlogs, and maintaining a regular release cadence.

Matplotlib is a community driven project with a vast majority of the work done by volunteers. The thing we are shortest on is the time of maintainers who are able and willing to review Pull Requests. Increasing the number of regular contributors and maintainers improves the sustainability of the project in several ways; many hands make light work and it makes the project resilient to individuals leaving the project.

Quantitatively evaluating maintenance work can be tricky — some Issues or PRs take minutes to review while others can take days to weeks of effort — but we believe that there is value at looking at the total number of open Issues and PRs and the rate at which they are addressed. We aim to close Issues and PRs faster than they are opened until a reasonable equilibrium is reached. We will aim to hit the following metrics:

- Initial response to all issues / new PRs within a week
- Resolve majority of new issues / PRs within 1 month
- Reduce backlog of issues by 25 / quarter
- Reduce backlog of PRs by 25 / quarter

Finally, for the user community to benefit from our work we need to do regular releases. We will continue to maintain a bi-yearly release cycle for feature releases with 1-3 bug-fix releases as needed. Holding to a time-based release schedule is advantageous because it provides predictability to our users and reduce the lag between when a new feature is merged to the default branch and when it is generally available.

### 1.7.4 Collaboration with Related Projects

As noted above, we will be working closely with both upstream and downstream projects on our refactor. This work is timely as Numpy, Pandas, and xarray are all adding new data types with richer metadata and unit support to the Python ecosystem. Matplotlib must meet the challenge of supporting these new data types. This will require constant communication with those projects. Similarly, our changes will need to meet the needs of downstream domain-specific projects. Many of these projects already have specialized data pipelines where and are working around the limitations of Matplotlib's data model. Matplotlib will aggressively engage with these projects during the data pipeline refactor to ensure that we will actually meet their needs.

Much of the communication is through the standard communication channels, such as project issue trackers, mailing lists or discussions forums, and submitting PRs. The strongest relationships, both historical and current, are with projects where we have shared developers. For example Ryan May and Elliot sales de Andre are both core contributors to Matplotlib and Cartopy and David Stansby is a maintainer on both Matplotlib and `SunPy`. Micheal Droettboom, the previous Matplotlib Project lead, was a core developer on Astropy. Additionally, through NumFOCUS and domain-specific conferences there are regular in-person meetings.

We will work with down-stream projects to help them improve their test coverage of plotting code and to add jobs to their continuous integration that test development snapshots of Matplotlib. This is particularly useful during the planned refactor as we will be making extensive changes to Matplotlib that we believe are backwards compatible, but down-stream projects and users exercise the library in ways the test suite does not. While we hope to introduce any regressions, catching them before release is the next best thing.

As a NumFOCUS project, we recognize the importance of every project that is part of our open source scientific computing community. Though we would like for our work to be funded we are committed to supporting and collaborating with other NumFOCUS projects that receive funding regardless of our own outcome. We believe that this attitude is crucial for the success of our community and the sustainability of open source projects. It is our hope that this sentiment will be taken into consideration when evaluating our proposal.

### 1.7.5 Inclusive Community Development

Matplotlib strives to be an inclusive and open project, anyone who is willing and able to contribute to the project should feel welcome to do so. We have adopted the Contributor Covenant V2.0 as our Code of Conduct[46].

Matplotlib has an open development model, all work is done in the open and welcome contributions, in the from of bug reports, feature requests, or pull requests from anyone. We maintain an extensive contribution guide (https://matplotlib.org/devdocs/devel/index.html) as part of our documentation. To improve the development and retention of new contributors we have recently started two efforts: an "incubator" channel on gitter and a Triage Team.

The hardest part of getting started to contributing to open source projects is can be simply getting started. Open source communities, particularly big ones, can be intimidating for first time contributors. The incubator is a semi-closed chat room where new contributors can get support on any aspect of contributing to Matplotlib. This include the technical aspects of the code they are working on, help with git/github, our review process, or the social expectations and norms of the community. The goal is that by providing this support to first time contributors we will retain more of them as regular contributors and then maintainers.

The issue tracker is important to communication in the project because it serves as the centralized location for making feature requests, reporting bugs, identifying major projects to work on, and discussing priorities. For this reason, it is important to curate the issue list, adding labels to issues and closing issues that are resolved or unresolvable. Triaging issues does not require any particular expertise in the internals of Matplotlib but is extremely valuable to the project. To this end we have created a "Triage Team" in the organization who have power to tag, milestone, and close issues. In addition to the direct benefit of improving the issue triage and freeing the core-developers to spend more time reviewing PRs, this role will bring more people into the developer community and may provide a path way to becoming regular contributors and maintainers.

Ongoing work at NumFOCUS to develop metrics will help us evaluate the efficacy of these efforts at diversifying our contributor base.

### 1.7.6 Contribution Workflow

Matplotlib is an established community driven project in the "federation" model as defined by Nadia Eghbal[47]. We have a core group regular maintainers who take responsibility for reviewing and merging proposed changes to the library and welcome Pull Requests from anyone who is interesting in contributing. We strive for consensus and rely on the collective judgment of our maintainers to maintain the quality and functionality of the library.

Matplotlib uses a variation on the "git flow" process[48] to manage proposing and reviewing contributions to the library and documentation on GitHub. Changes are proposed by opening a "Pull Request" on GitHub; the process is the same for core maintainers, regular contributors, and a first time contributors. The proposed changes are reviewed by maintainers who either request changes, which starts a cycle of iteration with the contributor, or approve. In addition to human review we have an extensive test suite that is automatically run via cloud services and the results are reported to the PR. Once consensus is reached and the test pass the PR is merged and the changes will be released as part of the next release.

Matplotlib maintains an extensive suite of automated tests that exercises a large fraction of our code base. The full test suite, and a build of the documentation, is run against several versions of Python on OSX, Windows, and Linux hosts on every PR and merge to the default branch via continuous integration. For every new feature or bug fix that gets merged we also add tests that codify the expected behavior. This gives us the ability to make changes to the library with confidence that we will not break existing behavior.

The threshold for merging a PR depends on the reviewer judgment of the risk of the changes. PRs that only change documentation, which cannot introduce regressions or introduce new features, may be merged by the first reviewer whereas code changes need to be reviewed and approved by at least two maintainers (not including the author of the PR). However in either case a maintainer may wish to leave a positive review but not merge the PR to request additional feed back from other maintainers. If a maintainer objects to a PR, the PR will not be merged until their concerns have been addressed. If consensus cannot be reached, the final decision falls back to a Deputy Project Lead or the Project Lead.

Matplotlib is cautious about making backwards-incompatible change that intentionally break users existing code. While in an ideal world, future versions of the library would be 100% backwards compatible with previous versions, sometimes we do need to make incompatible changes. As part of the review process we check that any API changes are well documented and justified. When technically possible we provide user-visible warnings the version before we actually implement a breaking change. This provides a window for users to either adapt to the change or to communicate to us that they cannot adapt so we can reconsider the change. Given this high barrier to changing or removing behavior we are careful to make sure that any new API we add to the library is well thought out and complete because once we have released a version of the library with that code it is hard to take it back. These considerations together are important enough that we have a Deputy Project Lead responsible for API consistency.

This process works well for incremental contributions and bug fixes, new features or bigger changes are typically discussed before significant work is done. In many cases if the feature does not need to be in the core library we encourage contributors to create a new stand-alone project. This has several advantages including the giving the author more control, allows them to iterate faster than our 6 month release schedule, and gives them greater flexibility to change their APIs.

### 1.7.7 Information Dissemination

As a project Matplotlib maintains a range of communication channels aimed at several, overlapping, audiences. This includes a the source, issue tracker and PRs on GitHub, the published documentation (including historical and development versions), weekly developer call, mailing lists, a discourse forum, an active chat room on gitter, in-person presentations at conferences and pydata events, a blog, and several social media accounts.

The center of gravity of Matplotlib development takes place on GitHub (https://github.com/matplotlib/matplotlib) where the canonical repository for the source and documentation is hosted. Around this repository we use Issues and PRs to track bug reports, feature requests, and to discuss proposed changes. We also host our governance documents on GitHub (https://github.com/matplotlib/governance) and revise them via Pull Request.

We publish extensive prose, example, and API documentation (https://matplotlib.org) that is refreshed with each release. In addition to the top-level docs, which always refer to the most recent release, we also host historical (e.g. https://matplotlib.org/3.1.3

for the v3.1.3 documentation) and development versions of the documentation (https://dev.matplotlib.org). In 2020 we had between 700k-1M unique visitors a month to the documentation.

The weekly developer calls are typically attended by six to eight people and are used for high-bandwidth discussions about both the overall direction of the project and technical issues. The agenda and minutes are publicly available (https://hackmd.io/team/matplotlib) and the calls are open to all.

Matplotlib has an active gitter (https://gitter.im/matplotlib/matplotlib) chat room. Gitter is a real-time chat platform that we use for general coordination and resolving minor technical discussions. While the chat room's history is technically persistent, we treat it as transient. For more in-depth discussions or anything we want a record of, we move the conversation to github, the mailing list, or discourse.

For user support and general discussion we maintain two mailing lists, matplotlib-users (https://mail.python.org/mailman/listinfo/matplotlib-users) for user support and matplotlib-devel (https://mail.python.org/mailman/listinfo/matplotlib-devel) for developer discussion and announcements, and a discourse (https://discourse.matplotlib.org) instance. While users have to subscribe or register respectively to post, these forums are open to all. We also maintain a read-only mailing list for announcements (https://mail.python.org/mailman/listinfo/matplotlib-announce).

Matplotlib developers have frequently attended meetings including SciPy, PyData, and science conferences.

We have a blog (https://matplotlib.org/matplotblog/) that hosts user-submitted content highlighting work they have done using Matplotlib. We have project accounts on several social media platform including twitter and Instagram.

## 2 References

[1] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

[2] D. Swade, P. Greenfield, R. Jedrzejewski, , and J Valenti. Dms level 1 and 2 data product design. Technical report, 2010.

[3] J. F. Bell III, A. Godber, S. McNair, M. A. Caplinger, J. N. Maki, M. T. Lemmon, J. Van Beek, M. C. Malin, D. Wellington, K. M. Kinch, M. B. Madsen, C. Hardgrove, M. A. Ravine, E. Jensen, D. Harker, R. B. Anderson, K. E. Herkenhoff, R. V. Morris, E. Cisneros, and R. G. Deen. The mars science laboratory curiosity rover mastcam instruments: Preflight and in-flight calibration, validation, and data archiving. *Earth and Space Science*, 4(7):396–452, 2017. doi: https://doi.org/10.1002/2016EA000219. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2016EA000219.

[4] 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. ISSN 1476-4687. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[5] 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. ISSN 1548-7105. doi: 10.1038/s41592-019-0686-2. URL https://doi.org/10.1038/s41592-019-0686-2.

[6] Astropy Collaboration, T. P. Robitaille, E. J. Tollerud, P. Greenfield, M. Droettboom, E. Bray, T. Aldcroft, M. Davis, A. Ginsburg, A. M. Price-Whelan, W. E. Kerzendorf, A. Conley, N. Crighton, K. Barbary, D. Muna, H. Ferguson, F. Grollier, M. M. Parikh, P. H. Nair, H. M. Unther, C. Deil, J. Woillez, S. Conseil, R. Kramer, J. E. H. Turner, L. Singer, R. Fox, B. A. Weaver, V. Zabalza, Z. I. Edwards, K. Azalee Bostroem, D. J. Burke, A. R. Casey, S. M. Crawford, N. Dencheva, J. Ely, T. Jenness, K. Labrie, P. L. Lim, F. Pierfederici, A. Pontzen, A. Ptak, B. Refsdal, M. Servillat, and O. Streicher. Astropy: A community Python package for astronomy. A&A, 558:A33, October 2013. doi: 10.1051/0004-6361/201322068.

[7] Astropy Collaboration, A. M. Price-Whelan, B. M. SipHocz, H. M. G”unther, P. L. Lim, S. M. Crawford, S. Conseil, D. L. Shupe, M. W. Craig, N. Dencheva, A. Ginsburg, J. T. Vand erPlas, L. D. Bradley, D. P’erez-Su’arez, M. de Val-Borro, T. L. Aldcroft, K. L. Cruz, T. P. Robitaille, E. J. Tollerud, C. Ardelean, T. Babej, Y. P. Bach, M. Bachetti, A. V. Bakanov, S. P. Bamford, G. Barentsen, P. Barmby, A. Baumbach, K. L. Berry, F. Biscani, M. Boquien, K. A. Bostroem, L. G. Bouma, G. B. Brammer, E. M. Bray, H. Breytenbach, H. Buddelmeijer, D. J. Burke, G. Calderone, J. L. Cano Rodr’iguez, M. Cara, J. V. M. Cardoso, S. Cheedella, Y. Copin, L. Corrales, D. Crichton, D. D’Avella, C. Deil, ’E. Depagne, J. P. Dietrich, A. Donath, M. Droettboom, N. Earl, T. Erben, S. Fabbro, L. A. Ferreira, T. Finethy, R. T. Fox, L. H. Garrison, S. L. J. Gibbons, D. A. Goldstein, R. Gommers, J. P. Greco, P. Greenfield, A. M. Groener, F. Grollier, A. Hagen, P. Hirst, D. Homeier, A. J. Horton, G. Hosseinzadeh, L. Hu, J. S. Hunkeler, Z. Ivezi’c, A. Jain, T. Jenness, G. Kanarek, S. Kendrew, N. S. Kern, W. E. Kerzendorf, A. Khvalko, J. King, D. Kirkby, A. M. Kulkarni, A. Kumar, A. Lee, D. Lenz, S. P. Littlefair, Z. Ma, D. M. Macleod, M. Mastropietro, C. McCully, S. Montagnac, B. M. Morris, M. Mueller, S. J. Mumford, D. Muna, N. A. Murphy, S. Nelson, G. H. Nguyen, J. P. Ninan, M. N”othe, S. Ogaz, S. Oh, J. K. Parejko, N. Parley, S. Pascual, R. Patil, A. A. Patil, A. L. Plunkett, J. X. Prochaska, T. Rastogi, V. Reddy Janga, J. Sabater, P. Sakurikar, M. Seifert, L. E. Sherbert, H. Sherwood-Taylor, A. Y. Shih, J. Sick, M. T. Silbiger, S. Singanamalla, L. P. Singer, P. H. Sladen, K. A. Sooley, S. Sornarajah, O. Streicher, P. Teuben, S. W. Thomas, G. R. Tremblay, J. E. H. Turner, V. Terr’on, M. H. van Kerkwijk, A. de la Vega, L. L. Watkins, B. A. Weaver, J. B. Whitmore, J. Woillez, V. Zabalza, and Astropy Contributors. The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package. *aj*, 156(3):123, September 2018. doi: 10.3847/1538-3881/aabc4f.

[8] The SunPy Community, Will T. Barnes, Monica G. Bobra, Steven D. Christe, Nabil Freij, Laura A. Hayes, Jack Ireland, Stuart Mumford, David Perez-Suarez, Daniel F. Ryan, Albert Y. Shih, Prateek Chanda, Kolja Glogowski, Russell Hewett, V. Keith

Hughitt, Andrew Hill, Kaustubh Hiware, Andrew Inglis, Michael S. F. Kirk, Sudarshan Konge, James Paul Mason, Shane Anthony Maloney, Sophie A. Murray, Asish Panda, Jongyeob Park, Tiago M. D. Pereira, Kevin Reardon, Sabrina Savage, Brigitta M. Sipőcz, David Stansby, Yash Jain, Garrison Taylor, Tannmay Yadav, Rajul, and Trung Kien Dang. The sunpy project: Open source development and status of the version 1.0 core package. *The Astrophysical Journal*, 890:68–, 2020. doi: 10.3847/1538-4357/ab4f7a. URL https://iopscience.iop.org/article/10.3847/1538-4357/ab4f7a.

[9] Ryan M. May, Sean C. Arms, Patrick Marsh, Eric Bruning, John R. Leeman, Kevin Goebbert, Jonathan E. Thielen, and Zachary S. Bruick. Metpy: A Python package for meteorological data, 2020. urlhttps://github.com/Unidata/MetPy.

[10] Met Office. *Cartopy: a cartographic python library with a matplotlib interface*. Exeter, Devon, 2010 - 2015. URL http://scitools.org.uk/cartopy.

[11] M. J. Turk, B. D. Smith, J. S. Oishi, S. Skory, S. W. Skillman, T. Abel, and M. L. Norman. yt: A Multi-code Analysis Toolkit for Astrophysical Simulation Data. *The Astrophysical Journal Supplement Series*, 192:9, January 2011. doi: 10.1088/0067-0049/192/1/9.

[12] Ravin Kumar, Colin Carroll, Ari Hartikainen, and Osvaldo A. Martin. ArviZ a unified library for exploratory analysis of Bayesian models in Python. *The Journal of Open Source Software*, 2019. doi: 10.21105/joss.01143. URL http://joss.theoj.org/papers/10.21105/joss.01143.

[13] Michael Waskom and the seaborn development team. mwaskom/seaborn, September 2020. URL https://doi.org/10.5281/zenodo.592845.

[14] S. Hoyer and J. Hamman. xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*, 5(1), 2017. doi: 10.5334/jors.148. URL http://doi.org/10.5334/jors.148.

[15] Daniel Foreman-Mackey. corner.py: Scatterplot matrices in python. *The Journal of Open Source Software*, 1(2):24, jun 2016. doi: 10.21105/joss.00024. URL https://doi.org/10.21105/joss.00024.

[16] Thomas Robitaille. *mpl-scatter-density*, 2020 (accessed January 18, 2021). URL https://github.com/astrofrog/mpl-scatter-density.

[17] ADS. *NASA/ADS - Matplotlib: A 2D Graphics Enviroment*, 2021 (accessed January 18, 2021). URL https://ui.adsabs.harvard.edu/abs/2007CSE.....9...90H/abstract.

[18] 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.

[19] John Hunter and Micheal Droettboom. matplotlib. In Amy Brown and Greg Wilson, editors, *The Architecture of Open Source Applications: Volume II*, chapter 11. 2012.

[20] Hugo van Kemenade. *Top PyPI Packages*, 2021 (accessed January 11, 2021). URL https://hugovk.github.io/top-pypi-packages/.

[21] GitHub. *Matplotlib Dependancy Graph*, 2021 (accessed January 11, 2021). URL https://github.com/matplotlib/matplotlib/network/dependents.

[22] Alena Guzharina. *We Downloaded 10,000,000 Jupyter Notebooks From Github – This Is What We Learned*, 2020 (accessed January 11, 2021). URL https://blog.jetbrains.com/datalore/2020/12/17/we-downloaded-10-000-000-jupyter-notebooks-from-github-this-is-what-we-learned/.

[23] Timothy J. Lang, Walter A. Lyons, Steven A. Cummer, Brody R. Fuchs, Brenda Dolan, Steven A. Rutledge, Paul Krehbiel, William Rison, Mark Stanley, and Thomas Ashcraft. Observations of two sprite-producing storms in colorado. *Journal of Geophysical Research: Atmospheres*, 121(16):9675–9695, 2016. doi: https://doi.org/10.1002/2016JD025299. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2016JD025299.

[24] Eric C. Bruning, Clemens E. Tillier, Samantha F. Edgington, Scott D. Rudlosky, Joe Zajic, Chad Gravelle, Matt Foster, Kristin M. Calhoun, P. Adrian Campbell, Geoffrey T. Stano, Christopher J. Schultz, and Tiffany C. Meyer. Meteorological imagery for the geostationary lightning mapper. *Journal of Geophysical Research: Atmospheres*, 124(24):14285–14309, 2019. doi: https://doi.org/10.1029/2019JD030874. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019JD030874.

[25] Timothy J. Lang. Investigating the seasonal and diurnal cycles of ocean vector winds near the philippines using rapidscat and ccmp. *Journal of Geophysical Research: Atmospheres*, 122(18):9668–9684, 2017. doi: https://doi.org/10.1002/2017JD027516. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2017JD027516.

[26] Timothy J. Lang. Comparing winds near tropical oceanic precipitation systems with and without lightning. *Remote Sensing*, 12(23):3968, Dec 2020. ISSN 2072-4292. doi: 10.3390/rs12233968. URL http://dx.doi.org/10.3390/rs12233968.

[27] S. D. Bale, S. T. Badman, J. W. Bonnell, T. A. Bowen, D. Burgess, A. W. Case, C. A. Cattell, B. D. G. Chandran, C. C. Chaston, C. H. K. Chen, J. F. Drake, T. Dudok de Wit, J. P. Eastwood, R. E. Ergun, W. M. Farrell, C. Fong, K. Goetz, M. Goldstein, K. A. Goodrich, P. R. Harvey, T. S. Horbury, G. G. Howes, J. C. Kasper, P. J. Kellogg, J. A. Klimchuk, K. E. Korreck, V. V. Krasnoselskikh, S. Krucker, R. Laker, D. E. Larson, R. J. MacDowall, M. Maksimovic, D. M. Malaspina, J. Martinez-Oliveros, D. J. McComas, N. Meyer-Vernet, M. Moncuquet, F. S. Mozer, T. D. Phan, M. Pulupa, N. E. Raouafi, C. Salem, D. Stansby, M. Stevens, A. Szabo, M. Velli, T. Woolley, and J. R. Wygant. Highly structured slow solar wind emerging from an equatorial coronal hole. *Nature*, 576(7786):237–242, Dec 2019. ISSN 1476-4687. doi: 10.1038/s41586-019-1818-7. URL https://doi.org/10.1038/s41586-019-1818-7.

[28] Andrew M. Annex and Kevin W. Lewis. Regional correlations in the layered deposits of arabia terra, mars. *Journal of Geophysical Research: Planets*, 125(6):

e2019JE006188, 2020. doi: https://doi.org/10.1029/2019JE006188. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019JE006188. e2019JE006188 10.1029/2019JE006188.

[29] Conor A. Nixon, Todd M. Ansty, Nicholas A. Lombardo, Gordon L. Bjoraker, Richard K. Achterberg, Andrew M. Annex, Malena Rice, Paul N. Romani, Donald E. Jennings, Robert E. Samuelson, and et al. Cassini composite infrared spectrometer (cirs) observations of titan 2004–2017. *The Astrophysical Journal Supplement Series*, 244(1):14, Sep 2019. ISSN 1538-4365. doi: 10.3847/1538-4365/ab3799. URL http://dx.doi.org/10.3847/1538-4365/ab3799.

[30] Erin Lee Ryan, Benjamin N. L. Sharkey, and Charles E. Woodward. Trojan asteroids in theKeplerCampaign 6 field. *The Astronomical Journal*, 153(3):116, feb 2017. doi: 10.3847/1538-3881/153/3/116. URL https://doi.org/10.3847/1538-3881/153/3/116.

[31] Alex H. Parker, Sarah M. Hörst, Erin L. Ryan, and Carly J. A. Howett. k-means Aperture Optimization Applied to Kepler K2 Time Series Photometry of Titan. PASP, 131(1002):084505, August 2019. doi: 10.1088/1538-3873/ab28ad.

[32] Simon B. Porter, Marc W. Buie, Alex H. Parker, John R. Spencer, Susan Benecchi, Paolo Tanga, Anne Verbiscer, J. J. Kavelaars, Stephen D. J. Gwyn, Eliot F. Young, H. A. Weaver, Catherine B. Olkin, Joel W. Parker, and S. Alan Stern. High-precision orbit fitting and uncertainty analysis of (486958) 2014 MU69. *The Astronomical Journal*, 156(1):20, jun 2018. doi: 10.3847/1538-3881/aac2e1. URL https://doi.org/10.3847/1538-3881/aac2e1.

[33] Aaron Tohuvavohu. personal communication.

[34] Aaron Tohuvavohu, Jamie A. Kennea, James DeLaunay, David M. Palmer, S. Bradley Cenko, and Scott Barthelmy. Gamma-Ray Urgent Archiver for Novel Opportunities (GUANO): Swift/BAT Event Data Dumps on Demand to Enable Sensitive Subthreshold GRB Searches. ApJ, 900(1):35, September 2020. doi: 10.3847/1538-4357/aba94f.

[35] Sanghita Sengupta, Nathan S. Nichols, Adrian Del Maestro, and Valeri N. Kotov. Theory of liquid film growth and wetting instabilities on graphene. *Phys. Rev. Lett.*, 120:236802, Jun 2018. doi: 10.1103/PhysRevLett.120.236802. URL https://link.aps.org/doi/10.1103/PhysRevLett.120.236802.

[36] Michael J. Eades, Paolo F. Venneri, Vishal Patel, Jonathan K. Witter, Dudley A. Raine, and Claude Russell Joyner. Leu cermet ntp design space studies. Nuclear and Emerging Technolgies for Space, 2018. URL http://anstd.ans.org/nets-2018/wp-content/uploads/2018/02/nets2018-web-program-secured.pdf.

[37] National Academies of Sciences, Engineering, and Medicine. *Open Source Software Policy Options for NASA Earth and Space Sciences*.

The National Academies Press, Washington, DC, 2018. ISBN 978-0-309-48271-4. doi: 10.17226/25217. URL https://www.nap.edu/catalog/25217/open-source-software-policy-options-for-nasa-earth-and-space-sciences.

[38] Kacper Kowalik. Plotting software in arxiv, 2018 (accessed January 18, 2021). URL https://dxl.ncsa.illinois.edu/arxiv/index.html.

[39] Nathan J. Goldbaum, John A. ZuHone, Matthew J. Turk, Kacper Kowalik, and Anna L. Rosen. unyt: Handle, manipulate, and convert data with units in python. *Journal of Open Source Software*, 3(28):809, aug 2018. doi: 10.21105/joss.00809. URL https://doi.org/10.21105/joss.00809.

[40] Hernan E. Grecco. *Pint: makes units easy - pint 0.16.1*, 2020 (accessed January 17, 2021). URL https://pint.readthedocs.io/.

[41] The Astropy Developers. *Units and Quantities (astropy.units)*, 2020 (accessed January 17, 2021). URL https://docs.astropy.org/en/stable/units/.

[42] Matplotlib Developers. *Main Goverance Document*, 2020 (accessed January 17, 2021). URL https://github.com/matplotlib/governance/blob/master/governance.md.

[43] Matplotlib Contributors. *License*, 2020 (accessed January 17, 2021). URL https://matplotlib.org/users/license.html.

[44] Python Software Foundation. *History and License*, 2020 (accessed January 17, 2021). URL https://docs.python.org/3/license.html.

[45] John Hunter. *Licenses*, 2008 (accessed January 17, 2021). URL https://matplotlib.org/devel/license.html.

[46] Coraline Ada Ehmke. *Contributor Covenant Code of Conduct*, 2020 (accessed January 17, 2021). URL https://github.com/matplotlib/matplotlib/blob/master/CODE_OF_CONDUCT.md.

[47] Nadia Eghbal. *Working in public: the making and maintenance of open source software*. Stripe Press, 2020.

[48] GitHub Guides. *Understanding the GitHub flow*, 2020 (accessed January 17, 2021). URL https://guides.github.com/introduction/flow/.

## 3 Data Management Plan

Matplotlib is a software library and does not produce any scientific data as defined in E.1.2 that needs to be preserved. Matplotlib is a tool used to visualize data that is acquired through other means.

Matplotlib is a community library developed in the open on GitHub. It is released the Matplotlib license (https://matplotlib.org/users/license.html) is a permissive license that is a derivative of the PSF license (https://docs.python.org/3/license.html) and compatible with the BSD-3 license. All work done on Matplotlib as part of this grant will be done through the current workflow, will be publicly available, and released under the same license. Matplotlib uses git for version control, thus every developer has the full history on their computer which provides significant redundancy. Tagged releases of the software are published to PyPI (in both source and binary forms). In addition, Anaconada, macports, homebrew, and all major Linux distributions independently build, package, and distribute Matplotlib. Extensive User facing documentation, including install instructions and usage guides, is built and hosted at https://matplotlib.org.

Any new libraries created as part of the ROSES award will be developed in the open on GitHub and will released under a permissive license.

Any incidental work on other software packages, either upstream or downstream of Matplotlib, will follow the license and development of process of those projects.

## 4 Biographical Sketch

**Thomas A. Caswell**
Computational Scientist
Brookhaven National Laboratory
Building 741 P.O. Box 5000
Upton, NY 11973-5000
(631) 344-3146

### Relevant Experience

13+ years working in data intensive experimental science. 8 years of experience contributing to Scientific Python, 6 years in leadership roles in the open source community. Current Project lead on Matplotlib, past release manager for h5py, has contributed to NumPy, Pandas, SciPy, and IPython. Co-developed `trackpy`, now widely used in soft-matter physics community. Lead architect and developer of Bluesky Suite for data acquisition now being adopted at x-ray facilities around the world.

### Education

| | |
|---|---|
| Ph.D., Physics, University of Chicago | 2014 |
| M.S., Physics, University of Chicago | 2008 |
| B.A., Physics, Mathematics, Cornell University | 2007 |

### Professional Experience

| | |
|---|---|
| Brookhaven National Laboratory, Computational Scientist | 2020 - Present |
| Columbia University, Visiting Associate Research Scientist | 2019 - Present |
| Brookhaven National Laboratory, Associate Computational Scientist | 2017 - 2020 |
| Matplotlib Project Lead | 2016 - Present |
| Brookhaven National Laboratory, Assistant Computational Scientist | 2015 - 2017 |
| Matplotlib Project Co-Lead | 2014 - 2016 |
| Brookhaven National Laboratory, Research Associate | 2014 - 2015 |

### Honors/Awards (selected)

| | |
|---|---|
| Brookhaven National Laboratory Spotlight Award | 2018 |
| Brookhaven National Laboratory Lab Award | 2017 |
| Google Open Source Peer Bonus | 2017 |

### Refereed Publications (selected)

1. Maksim S Rakitin, Abigail Giles, Kaleb Swartz, Joshua Lynch, Paul Moeller, Robert Nagler, Daniel B Allan, **Thomas A Caswell**, Lutz Wiegart, Oleg Chubar, Yonghua Du. **Introduction of the Sirepo-Bluesky interface and its application to the optimization problems** Proc. SPIE 11493, Advances in Computational Methods for X-Ray Optics V, 1149311 (21 August 2020)

2. Lucas J Koerner, **Thomas A Caswell**, Daniel B Allan, Stuart I Campbell. **A Python Instrument Control and Data Acquisition Suite for Reproducible Research**, IEEE Transactions on Instrumentation and Measurement (2019)
3. Anya Tafliovich, Francisco Estrada, **Thomas A Caswell**. **Teaching Software Engineering with Free Open Source Software Development: An Experience Report**, Proceedings of the 52nd Hawaii International Conference on System Sciences (2019)
4. Ronald J Pandolfi, Daniel B Allan, Elke Arenholz, Luis Barroso-Luque, Stuart I Campbell, **Thomas A Caswell**, Austin Blair, Francesco De Carlo, Sean Fackler, Amanda P Fournier, Guillaume Freychet, Masafumi Fukuto, Dogˇa Gürsoy, Zhang Jiang, Harinarayan Krishnan, Dinesh Kumar, R Joseph Kline, Ruipeng Li, Christopher Liman, Stefano Marchesini, Apurva Mehta, Alpha T N'Diaye, Dilworth Y Parkinson, Holden Parks, Lenson A Pellouchoud, Talita Perciano, Fang Ren, Shreya Sahoo, Joseph Strzalka, Daniel Sunday, Christopher J Tassone, Daniela Ushizima, Singanallur Venkatakrishnan, Kevin G Yager, Peter Zwart, James A Sethian, Alexander Hexemer. **Xi-cam: a versatile interface for data visualization and analysis**, Journal of Synchrotron Radiation (2018). 25, 1261-1270
5. Abdul K Rumaiz, Anthony J Kuczewski, Joseph Mead, Emerson Vernon, Donald Pinelli, Eric Dooryhee, Sanjit Ghose, **Thomas A Caswell**, D Peter Siddons, Antonino Miceli, Jonathan Baldwin, Jonathan Almer, John Okasinski, Orlando Quaranta, Russell Woods, Thomas Krings, Stuart Stock. **Multi-element germanium detectors for synchrotron applications**, Journal of Instrumentation (2018) 13 (04), C04030
6. Simon JL Billinge, Christopher J Wright, Chia-Hao Liu, Michael Waddell, Pavol Juhas, Eric Dooryhee, Sanjit Ghose, Milinda Abeykoon, Arman Arkilic, Daniel Allan, **Thomas A Caswell**. **Robust Nanostructure from High Throughput Powder Diffraction Data** (2017) Microscopy and Microanalysis 23 (S1), 172-173
7. **Thomas A Caswell**. **Dynamics of the vapor layer below a Leidenfrost drop**, Phys Rev E 90, 013014 (2014)
8. **Thomas A Caswell**, Zexin Zhang, Margaret L Gardel, and Sidney R Nagel. **Observation and Characterization of the Vestige of the Jamming Transition in a Thermal 3D System**, Phys Rev E 87, 012303 (2013)
9. **Thomas A Caswell**, Peter Ercius, Mark W Tate, Alper Ercan, Sol M Gruner, David A Muller. **A High Speed Area Detector for Novel Imaging Techniques in a Scanning Transmission Electron Microscope** Ultramicroscopy 109, 304-311 (2009)
10. Xin Liu, Kyoung-Su Im, Yujie Wang, Jin Wang, David LS Hung, James R Winkelman, Mark W Tate, Alper Ercan, Lucas J Koerner, **Thomas A. Caswell**, Darol Chamberlain, Daniel R Schuette, Hugh Philipp, Detlef M Smilgies, Sol M Gruner. **Quantitative Characterization of Near-Field Fuel Sprays by Multi-Orifice Direct Injection Using Ultrafast X-ray Tomography Technique**. Society of Automotive Engineers (SAE) Technical Paper 2006-01-1041

## 5 Table of Personnel and Work Effort

| Work Efforts to be Funded by this Proposal | | | | | |
|---|---|---|---|---|---|
| | | Commitment (FTE) | | | |
| **Name** | **Role** | **Y1** | **Y2** | **Y3** | **Total** |
| Dr. Thomas A. Caswell | PI & Research Software Engineer | 0.25 | 0.25 | 0.25 | 0.75 |
| – | Research Software Engineer | 1.00 | 1.00 | 1.00 | 3.00 |
| **Total Funded Work Effort** | | **1.25** | **1.25** | **1.25** | **3.75** |
| **Work Efforts Proposed, but NOT to be Funded by this Proposal** | | | | | |
| | | **Commitment (FTE)** | | | |
| **Name** | **Role** | **Y1** | **Y2** | **Y3** | **Total** |
| Dr. Thomas A. Caswell | PI & Research Software Engineer | 0.00 | 0.00 | 0.00 | 0.00 |
| - | Research Software Engineer | 0.00 | 0.00 | 0.00 | 0.00 |
| **Total Unfunded Work Effort** | | **0.00** | **0.00** | **0.00** | **0.00** |
| **TOTAL Work Efforts Proposed (Funded + Unfunded)** | | | | | |
| | | **Commitment (FTE)** | | | |
| **Name** | **Role** | **Y1** | **Y2** | **Y3** | **Total** |
| Dr. Thomas A. Caswell | PI & Research Software Engineer | 0.25 | 0.25 | 0.25 | 0.75 |
| – | Research Software Engineer | 1.00 | 1.00 | 1.00 | 3.00 |
| **Grand Total of Work Efforts** | | **1.25** | **1.25** | **1.25** | **3.75** |

# 6 Current and Pending Support

## 6.1 Current Awards

Dr. Caswell is fully supported by NSLS-II operations. He has an agreement to reduce time allocated to NSLS-II operations by the amount of time that is required by this proposal and the award below.

| Name of Principal Investigator on Award | Award / Project Title | Program Name / Sponsoring Agency / Point of Contact telephone and email | Period of Performance | Total Amount received | Commitment (Person-Month per Year) |
|---|---|---|---|---|---|
| Thomas A. Caswell | Matplotlib - Foundation of Scientific Visualization (EOSS3) | Essential Open Source Software for Science (Cycle 3) Chan Zuckerberg Initiative Dario Taraborelli | 01/21 - 01/22 | $250k | 3.6 |
| Thomas A. Caswell | Hosted mac-mini for testing OSX-specific bugs | Small Development Grant NumFOCUS Nicole Foster | 05/20 - 05/23 | $3k | 0 |

## 6.2 Pending Awards

None

## 7 Budget Justification

### 7.1 Budget Narrative

Due to being an established community driven project most regular contributors have established and distinct primary institutions. For this reason significant portions of the budget will need to be subcontracted to the primary institutions of key individuals who are uniquely qualified for this work.

Salary support is requested for PI Dr. Thomas A. Caswell (0.25 FTE in years 1-3). He will participate in all aspects of the proposed work and supervise the RSE.

Salary support is requested for a Research Software Engineer (1 FTE in years 1-3). The RSE will be responsible for carrying out the data model refactoring and general maintenance tasks on Matplotlib. The RSE will be hired as a consultant through NumFOCUS.

Support is requested for the RSE to attend the SciPy conference in Austin, TX each year. Cost estimates are based on the following: Meal per diem from the GSA website, airfare from NYC to ATX from https://travel.google.com, 2019 registration and conference hotel.

| | Austin, TX scipy conference |
|---:|---:|
| airfare | $500 |
| lodging | $770 (5 nights at $154/night) |
| Registration | $700 |
| per diem | $396 ($61/day) |
| misc ground transport | $100 |
| total for one person | $2,466 |

NumFOCUS organizes 4-5 regional PyData conferences in the United States each year. Support is requested for the RSE to attend one of the PyData conference each year. Cost estimates are based on the following: Meal and hotel per diem from the GSA website for NYC in November (as a worst-case scenario), airfare from NYC to LAX from https://travel.google.com, and guidance from NumFOCUS as to the registration fee.

| | New York, NY PyDATA conference |
|---:|---:|
| airfare | $500 |
| lodging | $858 (3 nights at $286/night) |
| Registration | $200 |
| per diem | $342 ($76/day) |
| misc ground transport | $100 |
| total for one person | $2,000 |

Total requested for travel: $13,398

#### 7.1.1 Facilities and Equipment

Standard desktop workstations that Caswell has access to, either through BNL or personal hardware, is sufficient for this work.

We request $4,000 to purchase a computer, monitor, and accessories for the RSE.

## 7.2 Budget Details

### 7.2.1 Year 1

***Direct Labor*** No direct labor

***Other Direct Costs*** *Subcontract/Subawards*

- Dr. Thomas A. Caswell is the PI and will oversee all aspects of the proposed work. The time commitment is 0.25FTE. His primary appointment is at Brookhaven National Laboratory which will be subcontracted for his effort.

*Consultants*

- RSE who will work on both the unit machinery and general maintenance. The time commitment is 1.00 FTE.

*Equipment*

- $3k is requested to purchase a computer for the RSE.

*Services*

- No support for services is requested.

*Travel*

- Support is requested for the RSE to attend two conferences a year ($4,466). See the budget narrative for estimated cost breakdown.

### 7.2.2 Year 2

***Direct Labor*** There is no direct labor.

***Other Direct Costs*** *Subcontract/Subawards*

- Dr. Thomas A. Caswell is the PI and will oversee all aspects of the proposed work. The time commitment is 0.25FTE. His primary appointment is at Brookhaven National Laboratory which will be subcontracted for his effort.

*Consultants*

- RSE who will work on both the unit machinery and general maintenance. The time commitment is 1.00 FTE.

*Equipment*

- $500 is requested for any computer equipment required by the RSE.

*Services*

- No support for services is requested.

*Travel*

- Support is requested for the RSE to attend two conferences a year ($4,466). See the budget narrative for estimated cost breakdown.

### 7.2.3 Year 3

***Direct Labor*** There is no direct labor.

***Other Direct Costs*** *Subcontract/Subawards*

- Dr. Thomas A. Caswell is the PI and will oversee all aspects of the proposed work. The time commitment is 0.25FTE. His primary appointment is at Brookhaven National Laboratory which will be subcontracted for his effort.

*Consultants*

- RSE who will work on both the unit machinery and general maintenance. The time commitment is 1.00 FTE.

*Equipment*

- $500 is requested is requested for any computer equipment required by the RSE.

*Services*

- No support for services is requested.

*Travel*

- Support is requested for the RSE to attend two conferences a year ($4,466). See the budget narrative for estimated cost breakdown.

## 7.3 Tabular Form

**Budget Summary**

| **Budget Period:** | **1 (start and end date from cover page)** | **2** | **3** | **Total** |
|---|---|---|---|---|
| **Start Date:** | 8/1/2021 | 8/1/2022 | 8/1/2023 | |
| **End Date:** | 8/1/2022 | 8/1/2023 | 8/1/2024 | |
| | | | | |
| **A. PI (Caswell, subaward to BNL)** | 480.00 | 480.00 | 480.00 | 1,440.00 |
| **B. RSE (consultant through NumFOCUS)** | 1,920.00 | 1,920.00 | 1,920.00 | 5,760.00 |
| **Total Hours (A+B)** | **2,400.00** | **2,400.00** | **2,400.00** | **7,200.00** |
| | | | | |
| **C. Equipment Description** | $3,000.00 | $500.00 | $500.00 | $4,000.00 |
| **D. Travel** | $4,466.00 | $4,466.00 | $4,466.00 | $13,398.00 |
| **E. Participant/Trainee Support Costs** | $0.00 | $0.00 | $0.00 | $0.00 |
| **F. Other Direct Costs** | $0.00 | $0.00 | $0.00 | $0.00 |
| **G. Direct Costs (A through F)** | **$7,466.00** | **$4,966.00** | **$4,966.00** | **$17,398.00** |
| | | | | |
| **H. Indirect Costs** | $0.00 | $0.00 | $0.00 | $0.00 |
| **I. Total direct and Indirect Costs (G + H)** | **$7,466.00** | **$4,966.00** | **$4,966.00** | **$17,398.00** |
| | | | | |
| **J. Fee (Only applies to for profit entities)** | N/A | N/A | N/A | N/A |
| **K. Budget Total (I + J)** | **$7,466.00** | **$4,966.00** | **$4,966.00** | **$17,398.00** |