Pandas

1 Goals

We propose funding the continued maintenance and development of Pandas (McKinney (2010)), a Python library providing high-performance, easy-to-use data structures.

Pandas is a foundational library in the Scientific Python Ecosystem, providing the most widely-used data structure for heterogenous, tabular data, and is the most-used Python tag on StackOverflow (Robinson (2019)). Pandas is widely used in industry and research, with over 1,000 citations (Mueller et al. (2019)).

1.1 Library Maintenance

Pandas is a large library with many users. We have many open issues (about 3,000) and Pull Requests (about 100). Keeping up with the stream of updates is time consuming, but important to the project. We would like to fund time dedicated specifically to maintenance.

By dedicating time specifically to maintenance we hope to reduce the open issue backlog. This should also make pandas easier to contribute to, hopefully increasing the growth rate of new contributors to the project.

We expect to see the number of open issues and pull requests decline.

1.2 Native String Data Type

Currently, pandas stores text data in an `object`-dtype NumPy array. The current implementation has two primary drawbacks:

1. `object`-dtype is not specific to strings: any Python object can be stored in an `object`-dtype array, not just strings.
2. Storing an array of Python strings as an `object`-dtype array is not memory efficient. The NumPy memory model isn't especially well-suited to variable width text data.

To solve the first issue, we would like to implement a new extension type for string data. This will initially be opt-in, with users explicitly requesting `dtype="string"`. Over time, we'd like to provide a migration path to users so that the string extension type is used by default for text data.

To solve the second issue (memory efficiency), we'll explore alternative in-memory array libraries (for example, Apache Arrow). Using a library whose data model is better-suited to text data should result in lower memory usage when loaded into pandas. We also expect that operations (for example `Series.str.upper`) will be faster.

The second issue with pandas' current string implementation will be considered complete when a `string`-dtype array is no longer backed by a NumPy array of Python string objects.

1.3 Documentation Validation

To improve the quality and consistency of Pandas documentation, we've developed tooling to check docstrings in a variety of ways. Every docstring is checked for

1. Consistency: Ensuring that every docstring has the same components (parameters, return values, notes, examples, etc.). Ensuring that each section is formatted correctly.
2. Completeness: Ensuring that every function parameter is documented.
3. Correctness: Ensuring that the examples run correctly.

When a user submits a Pull Request to pandas, their changes to the documentation are automatically checked and informative error messages notify them of any issues.

Like many other projects, pandas uses the numpydoc standard for writing docstrings. With the collaboration of the numpydoc maintainers, we'd like to move the project-agnostic tooling we've written to a package outside of pandas, which any numpydoc-using project can benefit from.

If possible, we'd like this project to be undertaken by a member of an unrepresented minority, with mentorship provided by the pandas maintaiers. This project primarily requires experience with *using* pandas, NumPy, and related libraries, rather that deep knowledge of pandas' internals. Pandas has a diverse community, just not at the maintainer level (yet).

1.4 Performance Monitoring

Pandas uses airspeed velocity to monitor for performance regressions. ASV itself is a fabulous tool, but requires some additional work to be integrated into an open source project's workflow.

The asv-runner GitHub organization, currently made up of pandas maintainers, provides tools built on top of ASV. We have a physical machine for running a number of project's benchmarks and tools managing the benchmark runs and reporting on results.

We'd like to fund improvements and maintenance of these tools to

- Be more reliable. Currently, they're maintained on the nights and weekends when a maintainer has free time. This occasionally results in days or weeks of downtime.
- Tune the system to improve benchmark stability
- Report performance regressions to a project rather than relying on maintainers manually checking for regressions.
- Build a GitHub bot to request ASV runs *before* a Pull Request is merged. The benchmarks are too expensive to run as part of every commit. Running on-demand from a maintainer provides a nice balance.
- Pay for hosting a dedicated benchmark server. A maintainer's basement isn't the most stable environment for a machine.

This project has an impact beyond pandas. We run the benchmarks for many foundational libraries in the Scientific Python Ecosystem including xarray, Dask, scikit-image, scikit-learn, and PyMC3. Each of these projects would benefit from improvements made to the tooling.

2 Work plan

2.1 Library Maintenance

Maintainers funded by this grant would be expected to

1. Promptly triage newly opened issues.
2. Promptly review new pull requests.
3. Ensure conversations on open issues are progressing and not waiting on maintainer feedback.
4. Ensure that Pull Requests are not going stale waiting for maintainer feedback or contributor responses to feedback.
5. Periodically review the issue backlog to find and close issues that are duplicates or no longer relevant. We should ensure that open issues are clearly described and scoped.

2.2 Native String Data Type

The two aspects of this proposal (`string` extension type and an alternative in-memory array library) can largely proceed in parallel.

We expect the string extension type to be the easier of the two tasks. Pandas' current extension array interface shouldn't require any modifications. The bulk of the work will be updating the existing documentation.

Updating the string extension array to use an alternative in-memory array library will be a larger effort. First, we'll need to determine which array library to choose.

If Apache Arrow is chosen, many string algorithms will need to be implemented, preferable in the Apache Arrow C++ library so that other languages binding to the C++ library benefit as well.

3 Existing Support

Pandas currently has about two FTE.

- Anaconda Inc funds Tom Augspurger (50% FTE) and Brock Mendel (100% FTE)
- Ursa Labs funds Joris van den Bossche (50% FTE).
- NumFocus Small Development Grant for "Improving and modernizing the introductory *Getting Started* pages of the pandas documentation" ($5,000)

An up-to-date list of current members and institutional parters can be found in our governance documents.

References

McKinney, Wes. 2010. "Data Structures for Statistical Computing in Python." In *Proceedings of the 9th Python in Science Conference*, edited by Stéfan van der Walt and Jarrod Millman, 51–56.

Mueller, Andreas, Andy Terrel, Thomas Caswell, Ralf Gommers, Gina Helfrich, and Ryan Abernathey. 2019. "Mid-Scale Research Infrastructure - the Scientific Python Ecosystem," April. https://doi.org/10.6084/m9.figshare.8009441.v1.

Robinson, David. 2019. "Why Is Python Growing so Quickly?" *StackOverflow Blog*. StackOverflow. https://stackoverflow.blog/2017/09/14/python-growing-quickly/.