Check for updates

The SunPy Project: An interoperable ecosystem for solar data analysis

The SunPy Community, Will T. Barnes[1,2]*[†], Steven Christe[1†],
Nabil Freij[3,4†], Laura A. Hayes[5†], David Stansby[6†], Jack Ireland[1],
Stuart J. Mumford[7], Daniel F. Ryan[8] and Albert Y. Shih[1]

[1]NASA Goddard Space Flight Center, Greenbelt, MD, United States, [2]Department of Physics, American
University, Washington, DC, United States, [3]Lockheed Martin Solar and Astrophysics Laboratory, Palo
Alto, CA, United States, [4]Bay Area Environmental Research Institute, Moffett Field, CA, United States,
[5]ESTEC, European Space Agency, Noordwijk, Netherlands, [6]Advanced Research Computing Centre,
University College London, London, United Kingdom, [7]Aperio Software Ltd., Leeds, England,
[8]University of Applied Sciences and Arts Northwest Switzerland, Windisch, Switzerland

The SunPy Project is a community of scientists and software developers creating
an ecosystem of Python packages for solar physics. The project includes the
sunpy core package as well as a set of affiliated packages. The sunpy core
package provides general purpose tools to access data from different providers,
read image and time series data, and transform between commonly used
coordinate systems. Affiliated packages perform more specialized tasks that do
not fall within the more general scope of the sunpy core package. In this article,
we give a high-level overview of the SunPy Project, how it is broader than the
sunpy core package, and how the project curates and fosters the affiliated
package system. We demonstrate how components of the SunPy ecosystem,
including sunpy and several affiliated packages, work together to enable multi-
instrument data analysis workflows. We also describe members of the SunPy
Project and how the project interacts with the wider solar physics and scientific
Python communities. Finally, we discuss the future direction and priorities of the
SunPy Project.

1 Introduction

The SunPy Project is an organization whose mission is to develop and facilitate a high-
quality, easy-to-use, community-led, free and open-source solar data analysis ecosystem
based on the scientific Python environment. The vision of the project is to build a diverse
and inclusive solar physics and heliophysics community that supports scientific discovery
and enables reproducibility through the development of accessible, open-source software
(Bobra et al., 2020). To achieve this mission and to make this vision a reality, the SunPy
Project maintains and guides the development of a number of Python packages including
the sunpy core package, and organizes educational activities around the use of Python for
solar-physics research.

As the scientific Python environment matured in the early 2010s (Hunter, 2007;
Harris et al., 2020; Virtanen et al., 2020), the development of a Python package devoted
to solar physics became viable. This led to the founding of the SunPy Project in 2011 by

scientists at NASA Goddard Space Flight Center. The goal of the SunPy Project at that time was to develop a package that provided the core functionality needed for solar data analysis in Python. To distinguish the software package from the wider project, this original package is now known as the sunpy core package (The SunPy Community et al., 2020). As the SunPy Project and sunpy grew, an ecosystem of affiliated packages (Section 2.2) was developed to keep the sunpy core package from becoming too large and difficult to manage.

The SunPy Project is committed to the principles of open development. All code is hosted and openly-developed on GitHub[1] in order to enable anyone to contribute code or provide feedback. All packages within the SunPy Project must be under an Open Source Initiative (OSI)[2] approved license. Discussion is hosted on several open communication channels which include weekly community calls, mailing lists, a Discourse forum, and instant messaging *via* Matrix[3]. Additionally, the SunPy Project has a code of conduct[4] to ensure that communication within the project is open, considerate, and respectful.

The aim of this paper is to give a high level description of the SunPy Project, including its various components, and to describe the direction of the project in the coming years. Section 2 describes the various Python packages that form the project, including both the sunpy core package (Section 2.1) and the various affiliated packages (Section 2.2). Section 3 gives an overview of the roles within the project and describes how to become involved with the SunPy Project. Section 4 describes the various activities of the project within the broader solar physics community. Finally, Section 5 lays out a vision for the future of the SunPy Project.

2 Code

2.1 The sunpy core package

The sunpy package is the central pillar of the SunPy Project (The SunPy Community et al., 2020) and provides the fundamental tools for accessing, loading, and interacting with solar physics data in Python. As we will discuss in Section 2.2, sunpy functions as one part of a larger ecosystem of packages for doing solar physics research in Python. While other packages in the ecosystem may focus on particular analysis techniques or analyzing data from specific instruments, the sunpy "core" package is focused on providing general tools for working with solar physics data. As an example, coordinate transformations between common solar coordinate systems are provided by the sunpy core package because they are needed for the analysis of nearly all solar imaging data and are critical for performing multi-instrument studies. However, correcting an AIA image to account for instrument degradation would not belong in sunpy because it is specific to data from one instrument. This allows the sunpy core package to be

relatively small in size, thereby assuring its maintainability over time.

The primary components of the sunpy package are described briefly in the following paragraphs. For a more in-depth description of each of these components, see The SunPy Community et al. (2020, Section 4). The full documentation of the sunpy Application Programming Interface (API) is provided in the hosted online documentation[5].

2.1.1 Components of the core package

To search for and download data, sunpy provides the Fido interface for searching across a variety of data providers [e.g., the Virtual Solar Observatory (VSO)[6], or the Joint Science Operations Center (JSOC)[7]] maintained within the solar community. Internally, Fido is both an interface that defines the search API for creating data queries as well as a collection of client classes that provide a translation between this user-facing API and the search parameters accepted by individual data providers. A complete list of all supported data sources is provided in the documentation for using Fido[8]. Section 4.1.1 of (The SunPy Community et al., 2020) also provides a comprehensive discussion of the data sources that Fido searches by default. Additionally, Fido can also be extended to search additional data sources that may not be included in sunpy (e.g., the Solar Orbiter Archive, see Section 2.2.2). Attributes such as time, wavelength, and instrument name, among others, can be used to filter these search results. By providing a single interface to many disparate data sources, sunpy, *via* Fido, easily enables multi-instrument research workflows.

Once a user has downloaded data, the TimeSeries and Map objects can be used to load and visualize time series and two-dimensional image data, respectively. These objects hold the data alongside the associated metadata in order to perform metadata-aware operations such as concatenation for time series or cropping for image data. In the case of Map, a World Coordinate System (WCS, e.g., Greisen and Calabretta, 2002) is also constructed from the associated metadata to enable easy mapping between pixel and world coordinates *via* astropy. In nearly all cases, solar image data is stored in the FITS format (Wells et al., 1981) which has an accompanying well-defined metadata standard (Pence et al., 2010). The accompanying metadata for each Map object adheres to this standard. Solar time series data, however, do not have a standard metadata or file format and are stored in a variety of file formats, including FITS, netCDF, JSON, or plain text. As such, the metadata associated with each TimeSeries object is much more sparse compared to Map, but at minimum will include the time of each observation as well as some information about the associated instrument that made the observation.

Additionally, by extending the astropy coordinates framework (See Section 3.3 of The Astropy Collaboration et al., 2018 for more details), sunpy provides definitions of, and transformations between, common solar coordinate systems. Coordinates expressed

---

using these frames can be used to represent the positional information of solar features and events. sunpy implements both observer-dependent (e.g., helioprojective Cartesian) and observer-independent (e.g., Stonyhurst heliographic) coordinate frames (Thompson, 2006). Each Map object instance also carries with it the corresponding coordinate frame of that image and the coordinate of the observer as defined by the position of the observatory given in the associated metadata.

2.1.2 Testing infrastructure

Sunpy includes thousands of unit, regression and integration tests that are run using the pytest testing framework. This test suite is run on every pull request opened on the sunpy GitHub repository using GitHub Actions to ensure that contributions to the codebase do not lead to unexpected regressions. A full description of our testing practices can be found in our developer documentation[9].

2.1.3 Release schedule

There is a new release of the core package with feature enhancements approximately every 6 months. Every other release is designated a long term support (LTS) release and receives bug fixes for a year rather than for 6 months. Additionally, there are bug fix releases every month. For each release a digital object identifier (DOI) is automatically generated and a record is created on Zenodo.[10] By providing regularly scheduled, versioned releases of sunpy, the SunPy Project enables reproducibility. For example, if a researcher is attempting to reproduce a result from a paper that used sunpy v2.0.2, she can create a new virtual environment and install that exact version of sunpy, even if the current version is many versions ahead of v2.0.2.

This release process is completely automated through GitHub Actions.[11] When a release is tagged, an action is triggered that tests the package on all supported versions of Python and all supported operating systems. If the packages build successfully, they are automatically uploaded to Python Package Index (PyPi) and subsequently the release is updated on conda-forge.

2.2 Affiliated packages

As the sunpy package grew and the amount of domain- and instrument-specific code being developed in Python increased, it became increasingly challenging to store and maintain the functionality needed for all solar physics research in one package. As such, the affiliated package system was introduced (Mumford and Christe, 2014) so that the sunpy core package could be generic enough for other packages to build on. The goal of this system is to support and promote software packages outside the scope of the sunpy core package, and to provide guidance to developers in implementing and maintaining the specific functionality provided

by an affiliated package. This fosters code-ownership while ensuring the set of affiliated packages are interoperable and follow a set of common standards (Section 2.2.1). The SunPy Project provides development support through our community development efforts and by providing a package template as a foundation. In addition, affiliated packages are advertised at conferences and workshops where a SunPy poster, talk, or tutorial is given.

As a result of the creation of the affiliated package ecosystem, components of the sunpy core package that were tied directly to specific instruments or data analysis methods have recently been moved out into other affiliated packages. One example of this is aiapy (Barnes et al., 2020), a package for processing data from the Atmospheric Imaging Assembly (AIA, Lemen et al., 2012) on the *Solar Dynamics Observatory* (SDO, Pesnell et al., 2012). Prior to version 2.1, sunpy included functionality for calibrating level 1 AIA data. In 2019, in collaboration with the SunPy Project, the AIA instrument team began developing aiapy to provide a number of AIA-specific analysis routines in Python, including the aforementioned calibration software. aiapy became an affiliated package in 2020 and the AIA-specific functionality that previously lived in sunpy was deprecated and subsequently removed. This relocation of the code allows the AIA instrument team to have full autonomy over their calibration routines and release updates to their software on a more frequent timescale than that of the sunpy core package. At the same time, aiapy users and developers are able to take full advantage of the SunPy Project ecosystem.

Outside of the current list of affiliated packages, current and future NASA and ESA missions[12], as well as ground-based telescopes, such as the Daniel K. Inouye Solar Telescope (DKIST), have begun developing user tools for data analysis and/or pipelines for data calibration built on top of the SunPy ecosystem. While these packages are not yet affiliated, the SunPy Project has assisted in coordinating development efforts between these teams in order to foster a more interoperable ecosystem.

2.2.1 Application process

The affiliated package application process is completed in the open on GitHub and is open to all, both individuals and larger collaborations (e.g., instrument teams). To begin the process, an applicant opens an issue on the SunPy Project website GitHub repository[13] and provides details about the package, including the package name, the maintainers, a link to the code repository, and a link to the documentation. The Affiliated Package Liaison (Section 3.2) then selects a SunPy Project member to review the candidate affiliated package against the following criteria:

- *functionality*—is the package relevant to the solar physics community?
- *integration*—does the package make use of the existing ecosystem?

- *documentation*—is there hosted documentation, including examples and an API reference?
- *testing*—are there automatically run tests and is the coverage extensive?
- *duplication*—does the package duplicate existing functionality in the ecosystem?
- *community*—is there a code of conduct and do the developers engage the wider community?
- *development status*—is the project actively maintained, including versioned releases?

The assigned project member then scores the package in each category using a "stoplight" system (i.e., a package is scored green, orange, or red in each category). A detailed description of each criterion and the scoring for each is available on the affiliated package page of the SunPy Project website[14]. The submitting author of the affiliated package may also request an alternate reviewer, in which case the Affiliated Package Liaison will assign a new SunPy Project member to review the package. At the end of the review, the candidate package is either accepted, marked as provisional, or not accepted. If the package is accepted, the affiliated package is added to the list of affiliated packages on the SunPy Project website. If the package is marked as provisional or is not accepted, the reviewer and the Affiliated Package Liaison will work with the package authors to help them achieve provisional or accepted status. Accepted affiliated packages are reviewed once a year to ensure the interoperability of the ecosystem does not regress and that affiliated packages are actively maintained.

In all cases, the goal of the affiliated package review process is to broaden the ecosystem of tools for solar data analysis in Python. These criteria are not meant to be exclusionary, but rather to ensure interoperability and consistency across the ecosystem for the benefit of both users and developers. Interoperability in this context, means that affiliated packages should make use of the existing `sunpy` core data structures, (e.g., `Map` and `Timeseries`), *in lieu* of their own custom data structures. In the context of searching for and downloading data, affiliated packages should use the `Fido` interface and extend `Fido` for additional data sources as needed.

2.2.2 Current ecosystem

At the time of writing, the SunPy Project has a rich and growing ecosystem of affiliated packages. In addition to the `sunpy` core package, the affiliated package ecosystem includes:

- `aiapy` for functionality specific to the AIA instrument (Barnes et al., 2020)
- `ndcube` for generic handling of *N*-dimensional data sets with a world coordinate system (WCS) (Ryan et al., 2021).
- `pfsspy` for magnetic-field extrapolation (Stansby et al., 2020)
- `sunkit-instruments` for instrument-specific code that does not have a dedicated package (Ryan et al., 2022).
- `sunkit-image` for solar-specific image analysis or reduction techniques (Freij et al., 2022).

- `sunpy-soar`[15] for querying the Solar Orbiter Archive (SOAR)[16].

To demonstrate how several of the affiliated packages can be used together with `sunpy` in a scientific workflow, we show an example in **Figure 1** of how coronal loop structures can be analyzed using potential magnetic field extrapolations and multi-point extreme ultraviolet (EUV) observations. We have included a Jupyter notebook that illustrates each step of this workflow in the GitHub repository that accompanies this paper[17].

First, we use the `Fido` interface provided by `sunpy` to search for and download a synoptic magnetogram from the Helioseismic Magnetic Imager (HMI, Scherrer et al., 2012) on SDO for Carrington rotation 2255 which began on 2022-03-08. This is shown in the left panel in **Figure 1A**. Next, we identify active region NOAA 12976 which appeared near disk center, as seen from SDO, at 2022-03-29 21:04. The red box overlaid on the synoptic magnetogram is centered on the active region when it appeared at disk center at a Carrington longitude of 65°.

Since we are interested in the EUV observations of active region 12976, we also use `Fido` to query the VSO for data from AIA on SDO and the Extreme Ultraviolet Imager (EUVI) on the *Solar Terrestrial Relations Observatory* (STEREO, Howard et al., 2008). Additionally, we use the `sunpy-soar` package to allow `Fido` to search for and download data from the SOAR. Here, we query the SOAR for data from the Extreme Ultraviolet Imager (EUI, Rochus et al., 2020) on *Solar Orbiter* (Müller et al., 2020).

Figure 1B shows full disk EUV images from AIA (left), the full-sun imager (FSI) on EUI (middle), and EUVI on the STEREO-A spacecraft (right). We use the `aiapy` package to correct the AIA image (middle panel) for instrument degradation and update the pointing information. The red box in each panel is centered on active region 12976, as seen from the respective spacecraft, and has a width and height of 700 arcseconds. The top right panel of **Figure 1** shows the Stonyhurst heliographic longitude and radius (in AU) of SDO, STEREO-A and *Solar Orbiter* as derived from the observer location metadata of each image.

Viewing the active region from the vantage points of these three spacecraft (separated by >90°), we gain a better understanding of its three-dimensional structure. Additionally, we use the `pfsspy` package to compute a potential field extrapolation from the corresponding synoptic magnetogram as shown in **Figure 1A**. We trace field lines from areas of negative magnetic flux inside the red box corresponding to active region 12976. The resulting field lines are overlaid in green on top of the cutouts from each EUV image in **Figure 1C**. Each field line traced using `pfsspy` is an `astropy` coordinate object expressed in terms of a Carrington heliographic coordinate frame defined in `sunpy`. As such, it is straightforward to transform each field line to the observer-dependent coordinate frame of each image as defined by the corresponding observatory using the plotting functionality

---
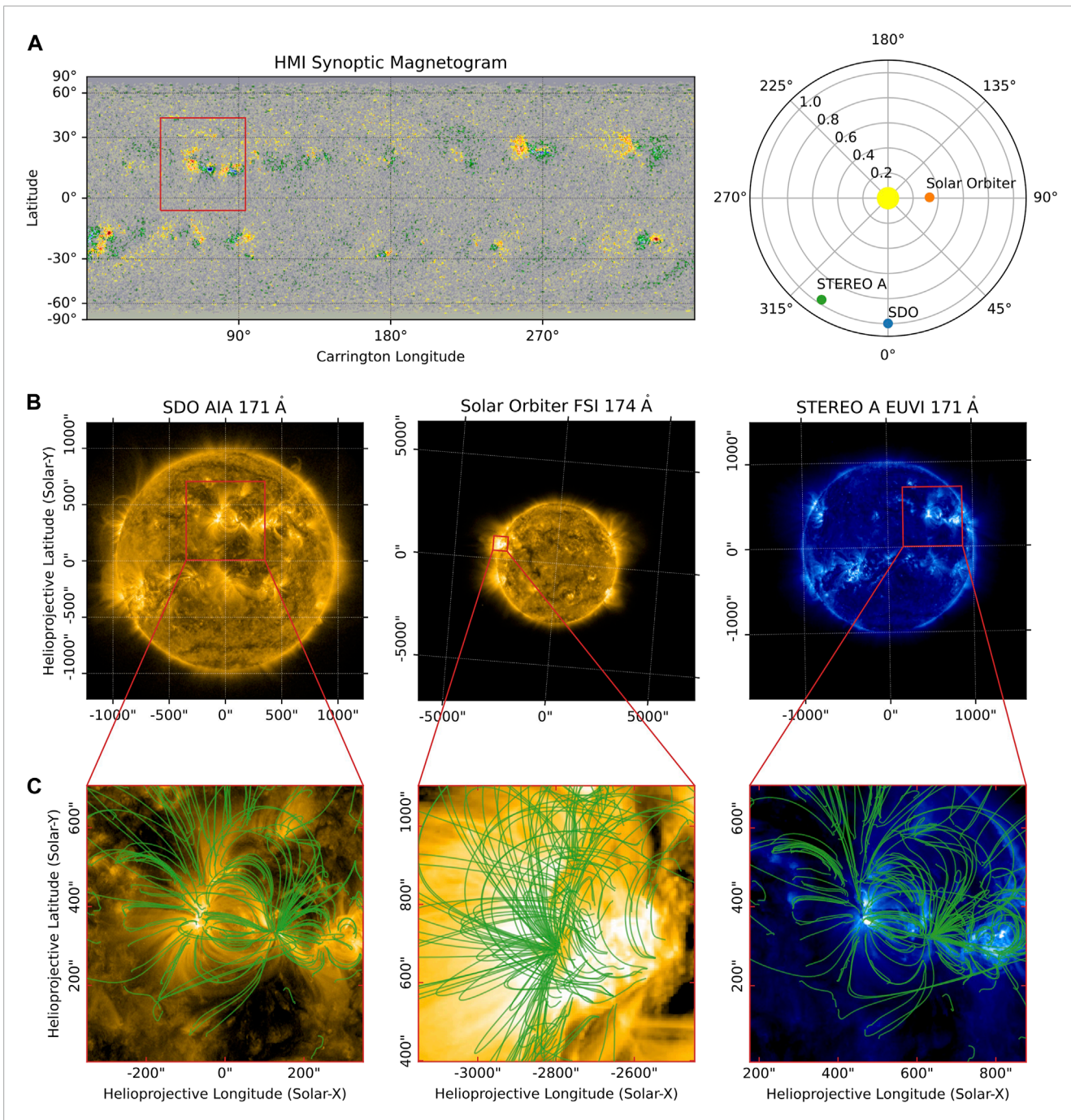
FIGURE 1
Illustration of multiple affiliated packages, including sunpy, sunpy_soar, aiapy, and pfsspy, working together. **(A)** The left panel shows the HMI synoptic magnetogram for Carrington rotation 2255. The red box is centered on the active region. The right panel shows the Stonyhurst heliographic longitude and radius (in AU) for SDO, STEREO A, and Solar Orbiter on 2022-03-29. **(B)** Full disk images from SDO AIA at 171 Å (left), SolO FSI at 174 Å (middle), and STEREO-A EUVI at 171 Å (right). All three images were downloaded using sunpy along with sunpy_soar to query the SOAR for the *Solar Orbiter* image. The AIA image was calibrated using aiapy. The red box in each panel is centered on the AR shown in the top panel. **(C)** Cutouts of the regions denoted in each image in **(B)**. pfsspy is used to compute a potential magnetic field solution from the magnetogram in **(A)** and trace field lines through the resulting volume. These field lines, shown in green, are transformed to the appropriate coordinate system of each instrument using sunpy.

provided in astropy. The interoperability between astropy, sunpy, sunpy-soar, aiapy, and pfsspy allows us to easily examine the three-dimensional magnetic structure of the active region and see to what extent the derived potential field corresponds to the EUV emission as observed by these three spacecraft.

3 People

3.1 Board and lead developers

The current structure of the SunPy Project is governed by the SunPy Project board (Christe, 2018). The board is a self-electing oversight board which delegates the majority of the day-to-day operations of the project to a lead developer, who in turn delegates it to members of the community. The lead developer has overall responsibility for the large scale organization of the sunpy core package, and ensures that pull requests comply with stated standards and align with the goals of the SunPy Project. The deputy lead developer supports the lead developer and fills in when the lead is absent. The board's role is to steer the overall direction of the SunPy Project and consists of scientists and researchers who are not necessarily involved directly with the day-to-day development of the sunpy core package.

3.2 Community roles

There are several specific community (or executive) roles within the SunPy Project that perform important duties related to the overall development and maintenance of the project. These roles encompass a range of responsibilities from the development of the core package and affiliated packages to project communication and liaison. The community roles are held by members of the wider solar community who are actively involved in the SunPy Project. Anyone interested in a community role is encouraged to apply.

At present, there are nine community roles within the SunPy Project. From the development side, these include the Lead Developer and the Deputy Lead Developer who are responsible for the development of the sunpy core package, support the development of affiliated packages, and lead the development community. To assist the Lead/Deputy Developers, there are several development community roles which include:

- Continuous Integration Maintainer
- Release Manager
- Webmaster
- Communication and Education Lead who is responsible for the overall engagement with the wider community
- Lead Newcomer and Summer of Code mentor who assists new contributors and oversees the Google Summer of Code project
- Affiliated Package Liaison who is responsible for overseeing the affiliated package review process (Section 2.2.1) and working with developers of current and candidate affiliated packages

3.3 Maintainers and contributors

The development of the sunpy core package depends principally on an established team of volunteers that support the Lead and Deputy Lead Developers. These volunteer *maintainers* are given commit access to the sunpy repository and are predominantly, though not exclusively, scientists from the solar community who use sunpy in their work. In addition to this group of core maintainers, there is a steady influx of new *contributors*,

averaging around 20–25 people per year. These contributors enable a wider range of features and code changes than would otherwise be normally possible due to the time constraints of the established team of volunteers. Within this subset of maintainers are members who maintain the specific sub-packages within sunpy like sunpy.map or sunpy.coordinates. These individuals are selected due to either their specific knowledge of the topic or their expertise with these sub-packages.

Contributing to the SunPy Project includes a wide range of activities, not all of them programming related. These include reporting bugs by raising issues on GitHub, requesting features, writing code and tests, providing feedback on pull requests, correcting or adding documentation, helping people who have problems or questions in communication channels and more. The SunPy Project is always looking for any new volunteers or people willing to contribute their time.

4 Community

4.1 Engagement with the solar physics community

In order for the SunPy Project to maintain and grow the sunpy core package and affiliated packages within the ecosystem, engagement with the wider solar physics community is critical. The mission of the SunPy Project is to be community-led, and the development is driven by the needs of the solar physics community. To facilitate this, the SunPy Project is building a community for which there is inclusive and open communication between those developing sunpy and those using sunpy in their scientific research. Active contributions from users in terms of bug reports, issues encountered with code or documentation, and feature requests are all vital to the sustainability and future of the SunPy Project. We emphasize that being part of the SunPy Project does not necessarily mean writing software. Contributions in the form of feedback and suggestions are equally important.

To foster communication, the SunPy Project supports several communication platforms (Section 4.1.1) through which users and developers can regularly interact. The SunPy Project posts on solar physics noticeboards about recent releases and regularly advertises sunpy and affiliated packages at scientific conferences, providing tutorials and support. We also ask that if sunpy is used for scientific work that it is cited in the literature[18], thereby increasing its visibility to the scientific community and ultimately contributing to the continued growth and development of the package. More recently, the SunPy Project has improved communications and established relationships with data providers such as VSO and the SOAR, and teams supporting both operating and developing instruments and missions. The SunPy Project is always looking for ways to improve the accessibility of the project and to grow the community.

---

18   See this page for a guide on how to cite sunpy in published works: https://sunpy.org/about#acknowledging-or-citing-sunpy

4.1.1 Communication channels

Over the years, the usage of sunpy and affiliated packages within the solar physics community has increased, and with that methods to communicate within the SunPy community have also increased. At the time of writing, several distinct communication channels are available. These include:

- Multiple GitHub repositories for bug reports and feature requests. These are listed under the SunPy Project GitHub organization[19].
- Real time messaging[20].
- Mailing lists[21].
- An online community forum[22].
- Weekly public calls that anyone can participate in[23].

Each has their own distinct purpose, and was created as a need arose for their existence. For example, the GitHub repository is used for the development of sunpy and issues and bugs can be raised there. However some scientists may not be familiar with GitHub and would like to ask a general question on how to do something within sunpy. For this, the mailing list, community forum, or real time Matrix chat may be the most appropriate. We actively encourage users and those interested in contributing to use any or all of these communication channels.

In addition to the main communication platforms specific to the SunPy Project, we maintain a presence within other communication channels used by the wider heliophysics community, including Helionauts[24] and communication channels used by the Python in Heliophysics Community.

4.2 Python in Heliophysics Community (PyHC)

The Python in Heliophysics Community (PyHC)[25] (Barnum et al., 2022) is a project with similar goals as the SunPy Project, but focuses on the wider Heliophysics community (Burrell et al., 2018). These include providing coding standards (Annex et al., 2018), curating a list of participating projects[26], hosting bi-monthly community meetings, and organizing an inaugural summer school for early career researchers. The SunPy Project is actively involved in PyHC, with sunpy being one of the core PyHC packages. SunPy Project members regularly attend community meetings and present updates. The SunPy Project also took part in the PyHC 2022 summer school. Moving forward, PyHC and the SunPy Project will continue to collaborate and build

upon efforts to use sunpy and affiliated packages within the larger heliophysics Python ecosystem.

4.3 Collaboration with the wider python ecosystem

The sunpy package forms part of the wider Python scientific ecosystem, requiring active collaboration with other scientific Python packages. Whenever possible, we aim to contribute to relevant open-source projects rather than duplicating functionality. As an example, large parts of sunpy depend on core functionality developed in the astropy package, including support for handling units, times, and coordinates.

The SunPy Project is sponsored by NumFOCUS, "a non-profit supporting open code for better science"[27]. NumFOCUS provides financial and organizational support for several important packages (e.g., numpy, pandas and xarray) and facilitates collaboration between packages throughout the scientific Python ecosystem. One example of this is the annual NumFOCUS summit that brings together the leaders of these packages to discuss interoperability, funding sources and other high-level topics that improve the Python ecosystem as a whole.

In addition, the SunPy Project is a member of the OpenAstronomy organization[28]. OpenAstronomy was created to collaborate on outreach, organize conferences such as Python in Astronomy, develop common tooling for infrastructure, and apply to internship programs such as Google Summer of Code (GSoC)[29] and Outreachy[30]. GSoC has been an invaluable source of programming effort for the SunPy Project over the past decade. The contributions from participants in this program have been crucial to sunpy. Examples of successful projects include the conversion to using astropy.time and creating a new Python API wrapper for Helioviewer.org [31]. As the focus of the OpenAstronomy organization is the broader astrophysics and astronomy community, the SunPy Project's participation has enabled closer ties within the rapidly growing Python-in-astronomy landscape.

5 The future of the SunPy Project

Development within the SunPy Project is driven by, and for, the solar physics community, responding to the needs of researchers for data analysis tools and techniques, and software for working with data from new missions. This means both the sunpy core package and other affiliated packages are continually changing and expanding. In September 2022 several members of the SunPy Project met at a coordination meeting to discuss the future of the project. Two key areas that emerged were updating the governance structure,

---

19 https://github.com/sunpy

20 Links to join the Matrix chat can be found at https://sunpy.org/help.html

21 https://groups.google.com/g/sunpy

22 https://community.openastronomy.org/

23 https://sunpy.org/jitsi

24 https://helionauts.org/

25 https://heliopython.org/

26 https://heliopython.org/projects/

27 https://numfocus.org/

28 https://openastronomy.org/

29 https://summerofcode.withgoogle.com/

30 https://www.outreachy.org/

31 https://hvpy.readthedocs.io/

and creating a roadmap for future development. The roadmap provides:

1. a set of priorities for developers to work on in the medium term.
2. a well scoped list of work items that funding can be sought for.
3. a mechanism to solicit input from the wider solar physics community on the medium term priorities for the SunPy Project.

At the time of writing, items in the draft roadmap include:

- Improving support and functionality for data with spectral coordinates (e.g., rastering spectrometers)
- Enabling multi-dimensional data sets (i.e., beyond 2D images).
- Improving support for running sunpy on cloud infrastructure.
- Creating a consistently structured set of documentation across all the SunPy Project packages.
- Adding functionality to rapidly visualize large data sets.
- Restructuring the project governance to, among other things, transform the lead developer positions into a multi-person steering committee and create an ombudsperson role.

The next step is consultation with the wider solar physics community. We invite feedback on this proposed roadmap *via* any of the aforementioned communication channels (**Section 4.1.1**) or by opening an issue on the repository used for tracking high-level, project-wide tasks and suggestions[32].

6 Conclusion

In this paper, we have summarized the SunPy Project and its various components, including the code developed and maintained by the project (**Section 2**), the people that comprise the project (**Section 3**) and the community that the project serves (**Section 4**). In particular, we have discussed how the sunpy package and the wider set of affiliated packages form a software ecosystem for solar physics research in Python and illustrated the types of analyses that such an ecosystem enables (**Figure 1**). Finally, we have summarized a tentative roadmap to steer the direction of the SunPy Project in the coming years. Importantly, we hope that such a high level description will provide a more clear understanding of the SunPy Project and the wider ecosystem and will encourage contributions of all forms from the global solar physics community.[33]

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found here: AIA and EUVI data are available through the VSO: https://sdac.virtualsolar.org/cgi/search. EUI data are available through the SOAR: https://soar.esac.esa.int/soar/. HMI

synoptic magnetogram data are available through the JSOC: http://jsoc.stanford.edu/.

Author contributions

WB contributed to writing the text and created **Figure 1**. SC, NF, LH, and DS contributed to writing the text. All authors contributed to manuscript planning and revision and have read and approved the submitted version.

Conflict of interest

Author SM was employed by Aperio Software Ltd.
The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

---

32   https://github.com/sunpy/sunpy-project

33   https://github.com/sunpy/sunpy-frontiers-paper

References

Annex, A., Alterman, B. L., Azari, A., Barnes, W., Bobra, M., Cecconi, B., et al. (2018). Python in heliophysics community (PyHC) standards. doi:10.5281/zenodo.2529131

Astropy Collaboration, Price-Whelan, A. M., Lim, P. L., Earl, N., Starkman, N., Bradley, L., et al. (2022). The astropy project: Sustaining and growing a community-oriented open-source project and the latest major release (v5.0) of the core package. *Astrophysical J.* 935, 167. doi:10.3847/1538-4357/ac7c74

Barnes, W., Cheung, M., Bobra, M., Boerner, P., Chintzoglou, G., Leonard, D., et al. (2020). Aiapy: A Python package for analyzing solar EUV image data from AIA. *J. Open Source Softw.* 5, 2801. doi:10.21105/joss.02801

Barnum, J., Masson, A., Friedel, R. H., Roberts, A., and Thomas, B. A. (2022). Python in heliophysics community (pyhc): Current status and future outlook. *Adv. Space Res.* doi:10.1016/j.asr.2022.10.006

Bobra, M., Bin, C., Christe, S., Hewett, R., Freij, N., Mumford, S., et al. (2020). SunPy proposal for enhancement 10: Vision and mission statement (SEP 0010). doi:10.5281/zenodo.7020094

Burrell, A. G., Halford, A., Klenzing, J., Stoneback, R. A., Morley, S. K., Annex, A. M., et al. (2018). Snakes on a spaceship—An overview of python in heliophysics. *J. Geophys. Res. Space Phys.* 123 (12), 402. doi:10.1029/2018JA025877

Christe, S. (2018). SunPy proposal for enhancement 2: SunPy organization definition (SEP 0002). doi:10.5281/zenodo.3261663

Freij, N., Ireland, J., Mumford, S., Barnes, W., Chaubey, V., Paul, J. A., et al. (2022). sunpy/sunkit-image: v0.4.2. doi:10.5281/zenodo.6578722

Greisen, E. W., and Calabretta, M. R. (2002). Representations of world coordinates in FITS. *Astronomy Astrophysics* 395, 1061–1075. doi:10.1051/0004-6361:20021326

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., et al. (2020). Array programming with NumPy. *Nature* 585, 357–362. doi:10.1038/s41586-020-2649-2

Howard, R. A., Moses, J. D., Vourlidas, A., Newmark, J. S., Socker, D. G., Plunkett, S. P., et al. (2008). Sun Earth connection coronal and heliospheric investigation (SECCHI). *Space Sci. Rev.* 136, 67–115. doi:10.1007/s11214-008-9341-4

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Comput. Sci. Eng.* 9, 90–95. doi:10.1109/MCSE.2007.55

Lemen, J. R., Title, A. M., Akin, D. J., Boerner, P. F., Chou, C., Drake, J. F., et al. (2012). The atmospheric imaging assembly (AIA) on the solar Dynamics observatory (SDO). *Sol. Phys.* 275, 17–40. doi:10.1007/s11207-011-9776-8

Müller, D., Cyr, O. C. S., Zouganelis, I., Gilbert, H. R., Marsden, R., Nieves-Chinchilla, T., et al. (2020). The solar orbiter mission - science overview. *Astronomy Astrophysics* 642, A1. doi:10.1051/0004-6361/202038467

Mumford, S., and Christe, S. (2014). SunPy proposal for enhancement 4: Packages affiliated with the SunPy project (SEP 0004). doi:10.5281/zenodo.3261752

Pence, W. D., Chiappetti, L., Page, C. G., Shaw, R. A., and Stobie, E. (2010). Definition of the flexible image transport system (fits), version 3.0. *Astronomy Astrophysics* 524, A42. doi:10.1051/0004-6361/201015362

Pesnell, W. D., Thompson, B. J., and Chamberlin, P. C. (2012). The solar Dynamics observatory (SDO). *Sol. Phys.* 275, 3–15. doi:10.1007/s11207-011-9841-3

Rochus, P., Auchère, F., Berghmans, D., Harra, L., Schmutz, W., Schühle, U., et al. (2020). The solar orbiter EUI instrument: The extreme ultraviolet imager. *Astronomy Astrophysics* 642, A8. doi:10.1051/0004-6361/201936663

Ryan, D., Mumford, S., Freij, N., Inglis, A., Leonard, D., Christe, S., et al. (2022). sunpy/sunkit-instruments: v0.3.1. doi:10.5281/zenodo.7190661

Ryan, D., Mumford, S., Sharma, Y., ankitBhope, A., Freij, N., et al. (2021). sunpy/ndcube: v2.0.1. doi:10.5281/zenodo.5715161

Scherrer, P. H., Schou, J., Bush, R. I., Kosovichev, A. G., Bogart, R. S., Hoeksema, J. T., et al. (2012). The helioseismic and magnetic imager (HMI) investigation for the solar Dynamics observatory (SDO). *Sol. Phys.* 275, 207–227. doi:10.1007/s11207-011-9834-2

Stansby, D., Yeates, A., and Badman, S. (2020). pfsspy: A Python package for potential field source surface modelling. *J. Open Source Softw.* 5, 2732. doi:10.21105/joss.02732

The Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., Günther, H. M., Lim, P. L., Crawford, S. M., et al. (2018). The astropy project: Building an open-science project and status of the v2.0 core package. *Astronomical J.* 156, 123. doi:10.3847/1538-3881/aabc4f

The SunPy Community, Barnes, W. T., Bobra, M. G., Christe, S. D., Freij, N., Hayes, L. A., et al. (2020). The sunpy project: Open source development and status of the version 1.0 core package. *Astrophysical J.* 890, 68. doi:10.3847/1538-4357/ab4f7a

Thompson, W. T. (2006). Coordinate systems for solar image data. *Astronomy Astrophysics* 449, 791–803. doi:10.1051/0004-6361:20054262

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nat. Methods* 17, 261–272. doi:10.1038/s41592-019-0686-2

Wells, D. C., Greisen, E. W., and Harten, R. H. (1981). Fits - a flexible image transport system. *Astronomy Astrophysics Suppl. Ser.* 44, 363.

Appendix

Here we provide a glossary of terms used throughout this paper:

- SunPy Project: The board and lead/deputy developers, the community roles, maintainers, and every package under its supervision.
- sunpy: The core package for using Python for scientific research in solar physics.
- SunPy ecosystem: The collection of packages that use or interface with sunpy and support scientific research in solar physics, including sunpy.
- Affiliated package(s): Solar physics related functionality outside the scope of the sunpy core package and that satisfies the standards enumerated in Section 2.2.1.