The Astrophysical Journal, 890:68 (12pp), 2020 February 10 https://doi.org/10.3847/1538-4357/ab4f7a

The SunPy Project: Open Source Development and Status of the Version 1.0 Core Package

The SunPy Community,
Will T. Barnes[1,2], Monica G. Bobra[3,31], Steven D. Christe[4,31], Nabil Freij[28,32], Laura A. Hayes[4], Jack Ireland[4,31,33], Stuart Mumford[5,6,31,34], David Perez-Suarez[7,31,35], Daniel F. Ryan[4,29], Albert Y. Shih[4]
(Primary Paper Contributors),
and
Prateek Chanda[9], Kolja Glogowski[10,11], Russell Hewett[12,31], V. Keith Hughitt[13], Andrew Hill[14], Kaustubh Hiware[15], Andrew Inglis[4], Michael S. F. Kirk[4,8], Sudarshan Konge[16], James Paul Mason[4], Shane Anthony Maloney[17,18], Sophie A. Murray[17,18], Asish Panda[19,36], Jongyeob Park[20], Tiago M. D. Pereira[21,22], Kevin Reardon[23,31], Sabrina Savage[24,31], Brigitta M. Sipőcz[25], David Stansby[26], Yash Jain[15], Garrison Taylor[27], Tannmay Yadav[15], Rajul[15], and Trung Kien Dang[30]
(Sunpy Contributors)

[1] Lockheed Martin Solar and Astrophysics Laboratory, Palo Alto, CA 94304, USA
[2] Bay Area Environmental Research Institute, Moffett Field, CA 94952, USA
[3] W.W. Hansen Experimental Physics Laboratory, Stanford University, Stanford, CA 94305, USA
[4] NASA Goddard Space Flight Center, Greenbelt, MD 20771, USA; steven.christe@nasa.gov
[5] SP^{2}RC, School of Mathematics and Statistics, The University of Sheffield, Sheffield, UK
[6] Aperio Software, Leeds, LS6 3HN, UK
[7] University College London, Gower Street, London, UK
[8] The Catholic University of America, Washington, DC 20664, USA
[9] Indian Institute of Engineering Science and Technology Shibpur, India
[10] Kiepenheuer-Institut für Sonnenphysik, Freiburg, Germany
[11] eScience Department, Computing Center, University of Freiburg, Freiburg, Germany
[12] Department of Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0123, USA
[13] Center for Cancer Research, National Cancer Institute, Bethesda, MD 20892-9760, USA
[14] Oklahoma Baptist University, Shawnee, OK 74804, USA
[15] Indian Institute of Technology Kharagpur, India
[16] Microsoft India Development Center, India
[17] Astrophysics & Space Physics, School of Physics, Trinity College Dublin, Dublin, Ireland
[18] Astronomy and Astrophysics, Cosmic Physics, Dublin Institute for Advanced Studies, Dublin, Ireland
[19] Google, Mountain View, CA, USA
[20] Space Science Division, Korea Astronomy and Space Science Institute, Daejeon 34055, Republic of Korea
[21] Institute of Theoretical Astrophysics, University of Oslo, Oslo, Norway
[22] Rosseland Centre for Solar Physics, University of Oslo, Oslo, Norway
[23] National Solar Observatory, Boulder, CO 80303, USA
[24] NASA Marshall Space Flight Center, Huntsville, AL 35812, USA
[25] DIRAC Institute, Department of Astronomy, University of Washington, Seattle, WA 98195, USA
[26] Mullard Space Science Laboratory, University of England, Holmbury Hill Rd, Dorking RH5 6NT, UK
[27] Harvard-Smithsonian Center for Astrophysics, Cambridge, MA 02138, USA
[28] Institute for Environmental Analytics, University of Reading, Reading RG6 6BX, UK
[29] American University, Washington, DC 20016, USA
[30] Saw Swee Hock School of Public Health, National University Health System, National University of Singapore, Singapore; mail@kien.ai
Received 2019 August 23; accepted 2019 September 20; published 2020 February 12

Abstract

The goal of the SunPy project is to facilitate and promote the use and development of community-led, free, and open source data analysis software for solar physics based on the scientific `Python` environment. The project achieves this goal by developing and maintaining the `sunpy` core package and supporting an ecosystem of affiliated packages. This paper describes the first official stable release (version 1.0) of the core package, as well as the project organization and infrastructure. This paper concludes with a discussion of the future of the SunPy project.

Unified Astronomy Thesaurus concepts: The Sun (1693)
Software reviewed by the Journal of Open Source Software JOSS

[31] SunPy Board Member.
[32] SunPy Deputy Lead Developer and Release Manager.
[33] SunPy Communications Officer.
[34] SunPy Lead Developer.
[35] SunPy Summer of Code Administrator.
[36] Google Summer of Code 2014 Intern.

1. Introduction

Research astrophysicists rely on software to analyze increasingly large and complex data sets. Scientists that study our nearest star, the Sun, are no different. Solar physicists rely on remote sensing data from both space- and ground-based instruments to measure the properties of our star and deduce the physical mechanisms at work. In the past, most solar data analysis was performed using Fortran, a general purpose, compiled programming language designed

for scientific and engineering applications. Several reasons motivated the solar physics community to transition to the Interactive Data Language (IDL), a commercial and closed-source programming language, in the 1980s. For one, IDL is an interpreted programming language that enables faster code development compared to Fortran. Second, IDL includes many libraries that support scientific data visualization and analysis. Finally, the solar community benefitted from functionality developed by the astronomy community (specifically, the IDL Astronomy User's Library). Since then, significant functionality has been developed by the solar community and is freely distributed as the SolarSoftWare library (Freeland & Handy 1998).

Officially founded in 2014 March, the goal of the SunPy project is to provide the core functionality needed for solar data analysis in `Python`,[37] a high-level interpreted programming language, and facilitate a new transition to bring significant and new benefits to the solar community. The SunPy project develops and maintains a community-led, free, and open source[38] core `Python` package (`sunpy`), supports an ecosystem of affiliated packages (see Section 6) consistent with best practices (Wilson et al. 2014), and engages with the community through mailing lists, chat rooms, tutorials, summer programs, and mentorship. The SunPy project has similar goals to the Astropy project,[39] which develops the `astropy` core package (The Astropy Collaboration et al. 2018) for the astrophysics community.

The choice of `Python` was motivated by several different factors. The scientific `Python` ecosystem provides a rich and mature ecosystem of packages for performing scientific analysis and computation. It is supported by foundational packages for manipulating tabular (`pandas`, McKinney 2010) and multidimensional array (`numpy`, van der Walt et al. 2011) data, general purpose scientific computing (`scipy`, Jones et al. 2001), and publication-quality 2D plotting (`matplotlib`, Hunter 2007).[40] These core packages form the backbone of hundreds of additional scientific `Python` packages, such as `astropy` for functionality and tools specific to astronomy, `scikit-learn` for machine learning and data mining (Pedregosa et al. 2011), and `dask` for parallel and distributed computing (Rocklin 2015). Interoperability between all these packages enables interdisciplinary analysis across traditional fields of study including solar physics, space physics, and astrophysics.

The `Python` programming language is freely available, meaning users are not bound by restrictive and costly proprietary licenses. `Python` is one of the most widely used programming languages by professional software developers[41] and is also now used by most universities to teach computer science (Guo 2014). Therefore, early career members of the solar physics community will likely already know how to code in `Python`.

Finally, an important cultural factor motivated the adoption of `Python`. `Python` and many `Python` packages are open source and developed under open source licenses approved by the Open Source Initiative.[42] The `Python` developer community has embraced an inclusive and open development culture, which means that anyone is welcome to develop functionality to either enhance existing packages such as `sunpy` or create new ones (see Section 3). This means one given person or institution does not control the development of scientific `Python` software. This approach, combined with strict version control, allows scientists to create fully reproducible results.

For all these reasons, the scientific `Python` ecosystem played a key role in recent major scientific discoveries, such as the first detection of a gravitational wave (LIGO Scientific Collaboration and Virgo Collaboration et al. 2016) and the first image of a black hole (The Event Horizon Telescope Collaboration et al. 2019). The data and code used to generate these results are openly available, allowing anyone to reproduce them. Because the solar community is relatively small[43] in relation to other scientific communities, it will benefit immensely by leveraging the growing scientific `Python` ecosystem to solve increasingly difficult scientific challenges and produce world-class science.

This paper describes the first stable release (version 1.0) of the `sunpy` core package, which is publicly available in a GitHub repository (Mumford et al. 2020). A previous paper describes version 0.5 (The SunPy Community et al. 2015). This article is not meant to replace the `sunpy` documentation but provides an overview of the organization and highlights important functionality. The full text of the paper, including all of the code to produce the figures, is available in a GitHub repository.[44] A list of abbreviations used throughout this paper are provided in Table 1 for convenience.

2. Project Organization and Enhancement Proposals

The organization of the SunPy project is modeled on the structure of a board-only not-for-profit corporate entity. It consists of a self-selected board of up to 10 members. An executive director, elected by the board, leads the core development team and the development of the `sunpy` package and supports the development of affiliated packages. As such, the executive director is also referred to as the lead developer. A deputy lead developer, a release manager, a group of subpackage maintainers, as well as other volunteers from the developer community, support the lead developer. Board members serve two-year terms while the lead developer serves a one-year term. No positions have term limits.

The SunPy project is formally defined through SunPy Enhancement Proposals (SEPs), which are modeled after the Python Enhancement Proposal (PEP) process.[45] All SEPs are version-controlled, citable, and publicly available.[46] The first SEP (SEP-0001, Christe 2014a) defines the scope of an SEP (similar to Greenfield 2013) while the second (SEP-0002, Christe 2018) defines the SunPy project organization.

[37] https://www.python.org/

[38] https://opensource.org/osd

[39] https://www.astropy.org

[40] According to the The 2018 Python Developers Survey https://www.jetbrains.com/research/python-developers-survey-2018/, data analysis is the primary use for `Python` thanks to the broad functionality provided by these foundational packages.

[41] According to the 2019 Stack Overflow developer survey (https://insights.stackoverflow.com/survey/2019), `Python` is the fourth most popular language among professional developers.

[42] https://opensource.org/licenses

[43] For reference, out of the ~9500 members of the American Astronomy Society, approximately 500 are members of the society's Solar Physics Division.

[44] https://github.com/sunpy/sunpy-1.0-paper

[45] https://www.python.org/dev/peps/

[46] https://github.com/sunpy/sunpy-SEP

There are three types of SEPs:

1. Standard: introduces and describes a new feature, or changes to an existing feature to `sunpy`, and is meant to function as a high-level technical design document.
2. Process: describes a new process in the organization, or changes to an existing process.
3. Informational: provides information but does not introduce any new features, changes, or processes.

As of the time of writing, there are a total of nine SEPs that have been approved by the board. Some notable SEPs have led to the formal adoption of physical units (SEP-0003, Christe 2014b, see Section 4.4) and a specific scientific time class (SEP-0008, Mumford 2018, see Section 4.4) throughout the code base. SEPs have also been used to formally define the affiliated-package program (SEP-0004, Mumford & Christe 2014, see Section 6) as well as the release schedule and versioning system (SEP-0009, Mumford & Hewett 2019, see Section 4.5).

3. Development Model

To satisfy the mission of the Project, the SunPy community adopted an open development model, which is widely used within the scientific `Python` community. The `sunpy` package is hosted on GitHub and uses `git`[47] as its distributed version control software. The entire code base is publicly available and anyone can suggest changes through pull requests. Since the code base is licensed under a permissive two-clause BSD license,[48] anyone can redistribute, improve, repackage, or use it in a closed environment as long as they credit the SunPy developers and redistribute the license. In order to maintain a high-quality code, every contribution is reviewed and must satisfy the following requirements:

1. Code and documentation must follow widely used style guides (e.g., PEP 8[49]).
2. All new features must be accompanied with documentation, including code comments, formal documentation such as `Python` docstrings and a guide, if appropriate, as well as gallery examples.
3. Contributed code must provide unit tests[50] that cover the majority of the new functionality.
4. All code must be within scope, and must be approved by at least two members of the developer community before it is merged into the code base.[51]

These requirements are imposed on every pull request regardless of the contributor.

As of version 1.0, `sunpy` consists of 48,427 lines of code[52] contributed by 123 unique contributors with over 11,659 commits.[53] The leftmost panel of Figure 1 shows the steady growth of the code base since 2011 with, on average, approximately 16 lines added per day. A large reduction in the code base occurred after the release of version 0.9, primarily due to the deletion of unused code and the removal of `Python` 2 support.

The middle panel of Figure 1 shows the steady rise in the number of unique contributors with, on average, more than one new contributor added each month. The total number of contributors is large for a package of this size in the heliophysics community (Ware et al. 2019).

The rightmost panel of Figure 1 shows the distribution of the total number of commits per contributor as of 2019 June. The distribution is relatively steep with a log–log slope of −0.35, which indicates that relatively few contributors generate the majority of commits. The top 10 contributors are responsible for ∼80% of all commits. This compares poorly to other projects such as Astropy (The Astropy Collaboration et al. 2018). Unlike Astropy, the SunPy project was not formed to bring together and coordinate many existing developers and their `Python` packages. The development of the `sunpy` package was begun by a core group of scientists with no prior existing code base. It should be noted that the number of commits is only approximately related to contribution size since a single commit could include a substantial increase in functionality or be a simple typographical fix. Regardless, this distribution suggests that the core developer team has not grown substantially from that original core team. On the other hand, the total number of unique contributors is large, which means that there exists a pool of `sunpy` developers that are willing and have the knowledge to contribute. Converting these contributors into core developers is crucial to the long-term health of the community.

4. The SunPy Core Package

The aim of the `sunpy` package, the core package in the SunPy ecosystem, is to provide a set of Python tools for performing common tasks in the analysis of solar data. These tools include querying and downloading data, loading and visualizing time series and images, and performing coordinate transformations. In the following sections, we describe the primary capabilities of `sunpy` and highlight several improvements to the package that are included in version 1.0.

4.1. Data Search and Retrieval

One of the most important tasks that occurs before any analysis can take place is to search for and retrieve data. A particular science goal may require data from multiple data providers, each of which may have different methods for data search and retrieval. This heterogeneity increases the effort required by scientists to get the data they need. In order to address this issue, the `sunpy.net` subpackage provides interfaces to many commonly used data providers and catalogs in solar physics.

4.1.1. The `Fido` Interface

The most powerful component of `sunpy.net` is the `Fido` interface for data search and retrieval. `Fido` provides a unified interface that simplifies and homogenizes search and retrieval by allowing data to be queried and downloaded from multiple data sources simultaneously, irrespective of the underlying client. Currently, `Fido` supports the Virtual Solar Observatory (VSO; Hill et al. 2009), the Joint Science Operations Center (JSOC; thanks to the `drms` affiliated package, see Section 6.1), and a number of individual data providers that make their data available via web-accessible resources such as HTTP(S) websites (*RHESSI*, *SDO* EVE, *GOES* XRS, PROBA2 LYRA,

[47] https://git-scm.com/
[48] https://opensource.org/licenses/BSD-2-Clause
[49] https://www.python.org/dev/peps/pep-0008/
[50] A unit test checks for the correct operation of a small component (or "unit") of the code base.
[51] This requirement is relaxed for bug fixes or small documentation changes.
[52] This number includes documentation and comments. There are 30,906 lines of code if documentation and comments are not counted.
[53] A commit is a set of one or more modifications to the code base, with an associated description, and typically created by a single contributor.

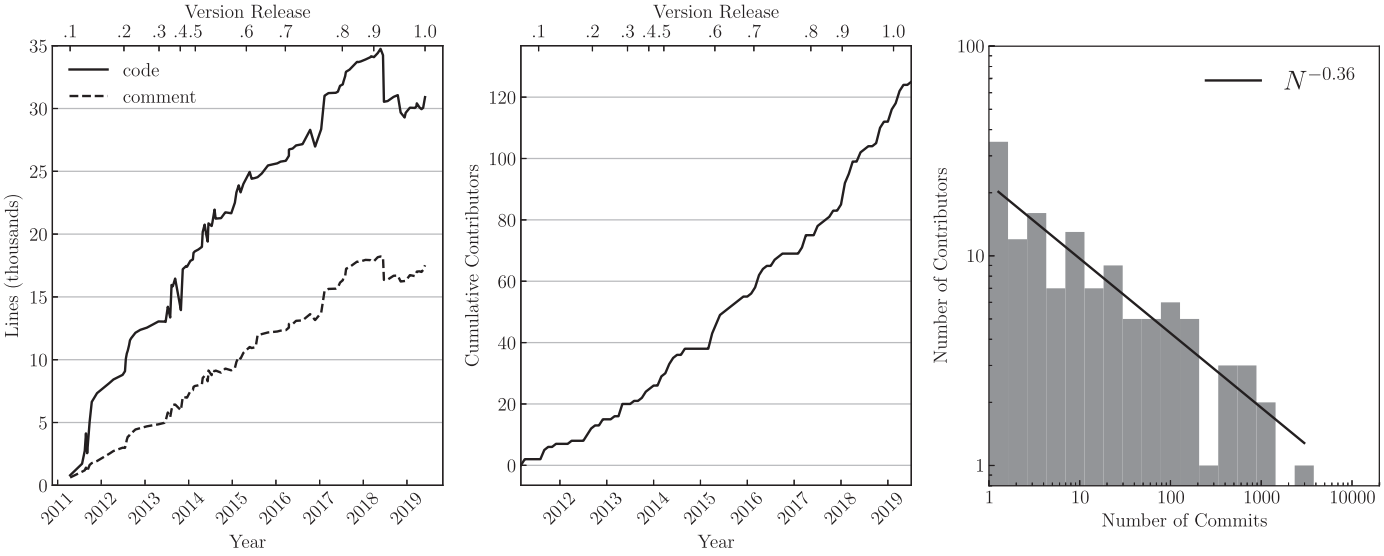


Figure 1. Left panel: a plot of the steady increase in the total number of lines of code (solid line) and lines of comments/documentation (dotted line) as a function of time. Major version releases are indicated along the top axis. A striking reduction in the code base occurred after version 0.9. This period saw a major code reorganization and deletion of obsolete features along with removing support for Python 2. Middle panel: the cumulative number of contributors to `sunpy` as a function of time shows a steady increase in the number of people involved in the development team. Right panel: a plot of the distribution of the number of commits per contributor. This distribution indicates that the majority of commits are undertaken by a small group of contributors. The average number of commits per contributor is less than 10 commits.

and NOAA sunspot number prediction) and FTP servers (NOAA sunspot number, NoRH).

A `Fido` search accesses multiple instruments and all available data providers in a single query. Search queries optionally include a variety of attributes, such as instrument, time range, and wavelength. The attributes can be joined using Boolean operators to enable complex queries. The result of a `Fido` query can be inspected and edited before retrieval and is then downloaded via asynchronous and parallel download streams. `Fido` also recognizes failed data downloads and allows for re-requesting files that were not retrieved.

4.1.2. HEK Client

In addition to data download, access to event catalogs are also an important aspect of solar physics research. One of the largest catalogs is the Heliophysics Event Knowledgebase (HEK, Hurlburt et al. 2012), which provides a searchable database of manually and automatically detected solar features and events (e.g., sunspots, flares, and coronal mass ejections). The `sunpy` package provides an HEK search client that is highly flexible allowing multiple event types and their properties to be queried simultaneously. For example, it is possible to search for active regions identified by the Spatial Possibilistic Clustering Algorithm (SPoCA, Verbeeck et al. 2014) above a user-specified size within a given time range.

4.1.3. Helioviewer Client

Finally, `sunpy.net` has a Helioviewer[54] client, which enables queries to the Helioviewer JPEG2000 image archive. This client accesses the data archive used by the powerful online solar image browsing tool provided by Helioviewer. It can query and download images as well as request constructed images of solar data from multiple sources.

[54] https://helioviewer.org/

Table 1
Common Abbreviations and Definitions

Abbreviation	Definition
AIA	Atmospheric Imaging Assembly
DRMS	Data Record Management System
GOES	*Geostationary Operational Environmental Satellite*
HAE	Heliocentric Aries Ecliptic
HCRS	Heliocentric Celestial Reference System
HEEQ	Heliocentric Earth Equatorial
HEK	Heliophysics Event Knowledgebase
HGC	Heliographic Carrington
HGS	Heliographic Stonyhurst
HPC	Helioprojective Cartesian
IDL	Interactive Data Language
IRIS	Interface Region Imaging Spectrograph
JSOC	Joint Science Operations Center
LTS	Long-term Support
MDI	Michelson Doppler Imager
NASEM	National Academies of Sciences, Engineering, and Medicine
NOAA	National Oceanic and Atmospheric Administration
PEP	Python Enhancement Proposal
SDO	*Solar Dynamics Observatory*
SJI	Slit Jaw Imager
SOHO	*Solar and Heliospheric Observatory*
SEP	SunPy Enhancement Proposal
UTC	Coordinated universal Time
VSO	Virtual Solar Observatory
WCS	World Coordinate System
XRS	X-Ray Sensor

4.2. Data Types

The `sunpy` package provides core data types that are designed to provide a general, standard, and consistent interface for loading and representing solar data across different instruments and missions. The two core data types currently supported in `sunpy` are handled by `TimeSeries` and `Map`, objects that support 1D temporal data and 2D image data,

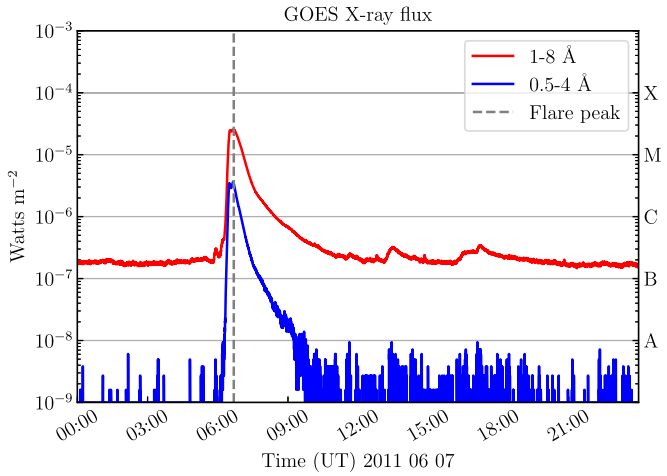


Figure 2. Example of a *GOES* XRS `TimeSeries` visualization over 24 hr. The two colors represent the two broadband channels; 1–8 Å (red) and 0.5–4 Å (blue). The sharp increase in flux at 06:33 UT is a solar flare and the gray dashed line indicates the time of the peak provided by the HEK.

respectively. The purpose of these core classes is to standardize data structures regardless of the data source (e.g., observational data from independent instruments). The classes maintain a consistent interface for accessing data attributes such as the data array itself as well as the metadata and relevant units. This arrangement allows for an easier workflow in the analysis of solar data observations. These core classes also include functionality for data manipulation and data visualization. This section provides an overview of the `TimeSeries` and `Map` data types.

4.2.1. `TimeSeries`

Many observations in the field of solar physics consist of spatially integrated measurements as a function of time. For example, the X-Ray Sensor (XRS) instrument on board the *Geostationary Operational Environmental Satellite* (*GOES*), which is used as the classification standard for solar flares, continuously measures the disk-integrated X-ray flux as a function of time in two broadband channels. The `TimeSeries` class aims to accommodate such solar time series data.

`TimeSeries` allows users to load time series data from a variety of solar instruments with appropriate units and timescales (see Section 4.4). A user can create a `TimeSeries` object either from data files stored locally (e.g., observational data sets acquired through `Fido`, see Section 4.1.1), or manually from custom time series data. The data array, metadata, and units data are all stored as attributes in the `TimeSeries` class.

Functionality is also provided for the manipulation of time series, including: adding, updating, truncating, resampling, and combining data within a `TimeSeries` or combining multiple `TimeSeries` together. `TimeSeries` also has built-in visualization methods to allow for easy inspection. An example of a `TimeSeries` created from *GOES* XRS observations of a solar flare is shown in Figure 2.

`TimeSeries` currently supports the data sources listed in Table 2 in addition to indices of solar activity from the National Oceanic and Atmospheric (NOAA) Space Weather Prediction Center that track the solar cycle and its predicted progression. Due to its flexible data structure, it is straightforward to add additional instruments and data sources to the `TimeSeries` object.

Table 2
Instruments Supported by the `Timeseries` and `Map` Objects as Described in Section 4.2

Supported by `Timeseries`	**Instrument reference**
Geostationary Operational Environmental Satellite (*GOES*) X-Ray Sensor (XRS)	Garcia (1994), Hanser & Sellers (1996)
Fermi Gamma-ray Burst Monitor (GBM)	Meegan et al. (2009)
Nobeyama Radioheliograph (NoRH)	Nakajima et al. (1994)
PRoject for Onboard Autonomy (PROBA2) Large Yield Radiometer (LYRA)	Dominique et al. (2013)
Solar Dynamics Observatory (*SDO*) EUV Variability Experiment (EVE)	Woods et al. (2010)
Reuven Ramaty High Energy Solar Spectroscopic Imager (*RHESSI*)	Lin et al. (2002)
Supported by `Map`	**Instrument reference**
COronal Solar Magnetism Observatory (COSMO) K-coronagraph (K-Cor)	de Wijn et al. (2012)
Hinode X-Ray Telescope (XRT)	Golub et al. (2008)
Interface Region Imaging Spectrograph (IRIS) Slit Jaw Imager (SJI)	De Pontieu et al. (2014)
PROBA2 Sun Watcher using Active Pixel System detector and Image Processing (SWAP)	Seaton et al. (2013)
RHESSI	Lin et al. (2002)
Solar and Heliospheric Observatory (*SOHO*) Extreme ultraviolet Imaging Telescope (EIT)	Delaboudiniere et al. (1995)
SOHO Large Angle Spectroscopic COronagraph (LASCO)	Brueckner et al. (1995)
SOHO Michelson Doppler Imager (MDI)	Scherrer et al. (1995)
SDO Atmospheric Imaging Assembly (AIA)	Lemen et al. (2012)
SDO Helioseismic and Magnetic Imager (HMI)	Schou et al. (2012)
Solar TErrestrial RElations Observatory (*STEREO*) Extreme Ultraviolet Imager (EUVI), COronagraph 1 and 2 (COR1/2) for both *STEREO A* and *B*	Howard et al. (2008)
Transition Region and Coronal Explorer (TRACE)	Handy et al. (1999)
Yohkoh Soft X-ray Telescope (SXT)	Tsuneta et al. (1991)

4.2.2. `Map`

A majority of solar data is in the form of images of the Sun. Images of the Sun are taken in multiple wavelengths from a wide range of both space- and ground-based instruments. For example, the Helioseismic and Magnetic Imager (HMI) instrument on board the *Solar Dynamics Observatory* (*SDO*) maps the magnetic field at the photosphere across the entire solar disk every 45 s. Images also require precise coordinate information in order to compare solar features observed across multiple wavelengths with different instruments.

The `Map` class in `sunpy` provides a framework to contain and analyze image data. A `Map` can be created from local files or using a URL to a remote data file. The `Map` class will automatically detect supported instruments and parse the

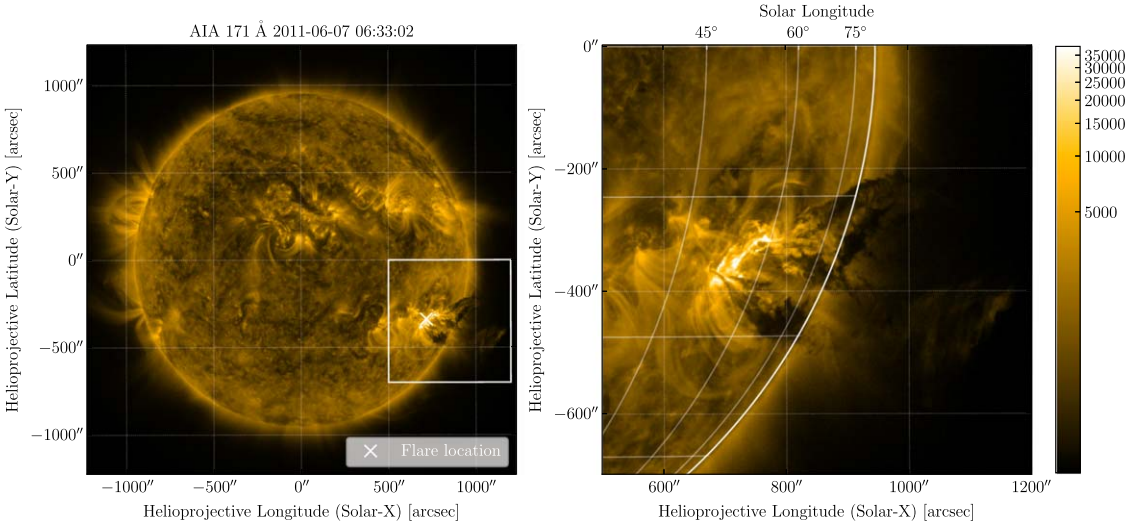


Figure 3. Example of a `Map` visualization from observations of the 171 Å wavelength channel of AIA on board *SDO*. The left panel shows an image of the entire Sun with the flare position as given by the HEK. The right panel shows the cropped view in the white box of the left-hand panel, focusing on an erupting flare (the same event as depicted in Figure 2).

metadata to infer the coordinate system from the appropriate Flexible Image Transport System (FITS) keywords (Thompson 2006; Pence et al. 2010). Other source-specific metadata is used to determine the appropriate color table and image normalization for visualization. It is also possible to create a custom `Map` by providing a 2D data array and metadata.

Similar to `TimeSeries`, the `Map` class contains the image data array, coordinate information, and relevant metadata as attributes. Visualization methods are also provided to inspect and plot solar image data. This visualization set includes the ability for `Map` to plot the image data in a way that represents the coordinates of the image accurately (see Section 4.3 for more on solar coordinates). This routine allows the user to plot coordinates of interest on a `Map` while correctly accounting for the coordinate frame. Other plotting components in `Map` include the ability to mask and clip the image data.

An example of a `Map` using data from the Atmospheric Imaging Assembly (AIA) on board *SDO* is shown in Figure 3. The left panel shows the full field of view of AIA using the appropriate color table and scaling for the observation, and the right panel shows a cropped view highlighting a solar flare whose *GOES* XRS light curve is shown in Figure 2.

When analyzing the dynamics of features on the solar disk, it is important to account for variations due to the apparent rotation of the Sun, which includes solar differential rotation, a latitudinally varying rotation rate due to the nonrigidity of the solar interior (see Beck 2000). The `sunpy.physics.differential_rotation` subpackage provides the ability for a `Map` to be transformed to earlier or future times including the effect of solar differential rotation (see Figure 4).

The `Map` class also provides functionality to analyze multiple images, such as overlaying images from different instruments with overlapping fields of view, or to combine multiple images together in a time-ordered sequence. This class includes the ability to coalign images from different instruments or images taken at different times to facilitate multi-instrument studies.

`Map` currently supports the data sources listed in Table 2 as well as the Helioviewer JPEG2000 image files of each of these data sources. Similar to `TimeSeries`, `Map` is architected to enable new data sources to be added easily.

4.3. Solar Coordinates

The `sunpy.coordinates` subpackage provides support for representing and transforming coordinates used in solar physics. These coordinates may represent events (e.g., flares), features on or above the Sun (e.g., magnetic loops), or the position of structures traveling throughout the heliosphere (e.g., coronal mass ejections). The package currently implements many of the most widely used Sun-centered coordinate frames including Helioprojective Cartesian (HPC), Heliographic Carrington (HGC), and Heliographic Stonyhurst (HGS), as well as Heliocentric Aries Ecliptic (HAE), Heliocentric Cartesian (HCC), and Heliocentric Earth Equatorial (HEEQ). See Thompson (2006) for more information about each of these coordinate frames. Additional coordinate frames will be available in future releases. The functionality provided in this package is built on top of and integrates with the `astropy.coordinates` framework (see Section 3.3 of The Astropy Collaboration et al. 2018).

An important feature of some of these solar coordinate frames is that they are observer-dependent, meaning that they are not fully defined without also specifying the location of the observer. The HPC frame, in particular, which is the most widely used coordinate frame for images of the Sun, places the origin of the frame at the observed center of the solar disk. This means that these observer-dependent frames have axes that change orientation depending on the location of the observer. The coordinates of the observer are stored in the same object as the other coordinate information. Since many observations of the Sun take place from or near the Earth, it is frequently an adequate approximation to use the center of the Earth as the observer position.

The observer-independent coordinate frames (HAE, HEEQ, HGC, and HGS) are useful for specifying the locations of features on the Sun or objects (e.g., spacecraft) in interplanetary space. The commonly used HGS frame is of particular note because it transforms in a straightforward manner to and from the Heliocentric Celestial Reference System (HCRS). This reference frame consists of the combination of two rotation angles: the time-

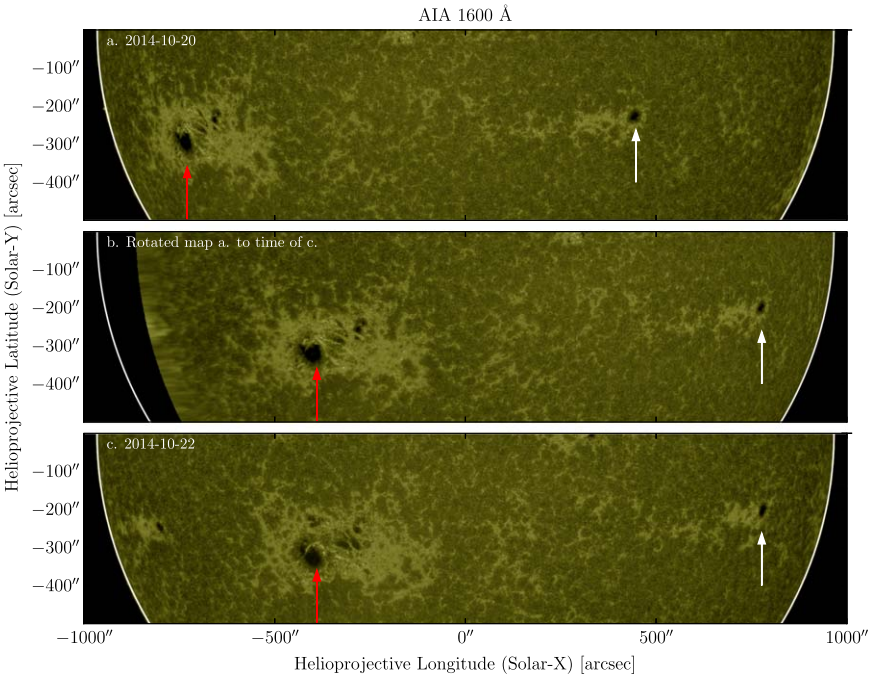


Figure 4. Example of transforming a `Map` to a different time, while accounting for differential rotation of the Sun. Panel (a) shows the Sun as observed by *SDO* AIA in 1600 Å on a particular day. A large and small sunspot group are highlighted by a red arrow and a white arrow, respectively. Panel (b) shows the observation after transforming forward in time by two days. Panel (c) shows the real observation at the time of panel (b), which compares well to the transformed image, disregarding the magnetic evolution of the sunspot groups.

independent angle between the Sun's rotation axis and the HCRS celestial pole (see Seidelmann et al. 2007) and the time-dependent angle of the Sun's central meridian (as seen from Earth) relative to the vernal equinox. This transformation provides the link from the frames defined in `sunpy.coordinates` with those defined in `astropy.coordinates`, allowing any solar frame to be transformed to and from celestial coordinate frames (see Figure 5). The functionality provided by the `sunpy.coordinates` subpackage has been extensively tested and agrees with published values in the *Astronomical Almanac*. For example, the apparent R.A. of the Sun agrees to a precision of 0.01 arcsecond.

A few example applications of the `sunpy.coordinates` subpackage are shown in Figure 6. The leftmost panel shows magnetic-field-line extrapolations projected onto a *SDO* AIA 171 Å image of an active region. The center and rightmost panels of Figure 6 make use of functions in `sunpy.coordinates` to obtain the apparent (light-travel time-corrected) locations of solar system bodies and overlay them on images of the Sun in solar coordinate frames.

This subpackage also provides the capability to query ephemeris information from a few different sources, including the active ephemeris in `astropy.coordinates`, JPL ephemerides, or JPL HORIZONS.[55]

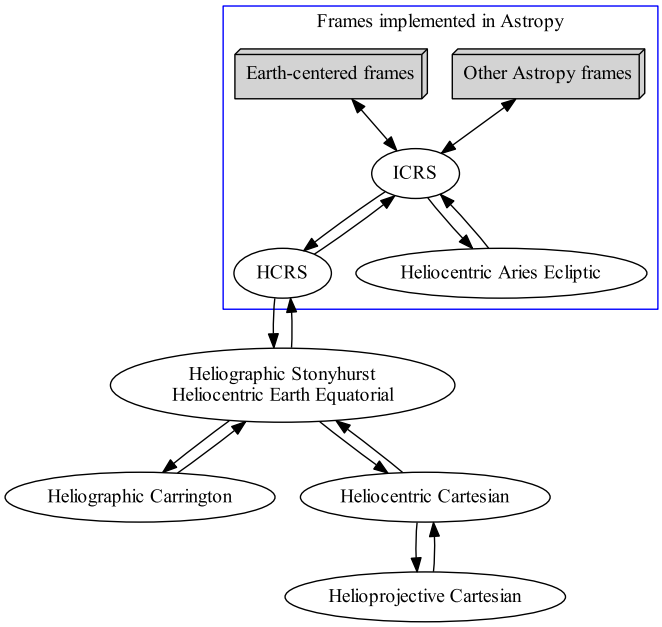


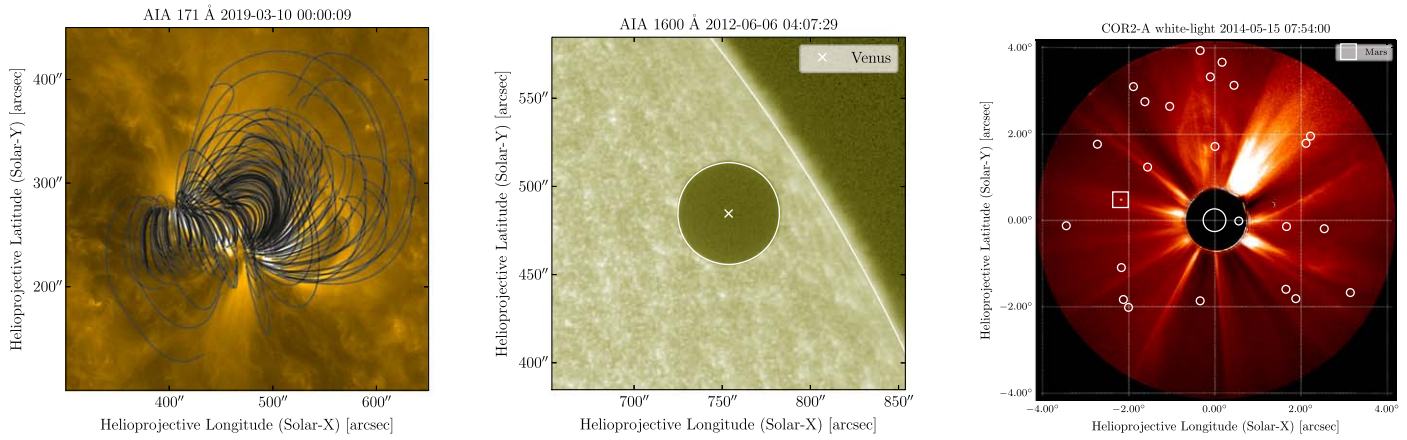
Figure 5. Diagram of the coordinate frames accessible through `sunpy.coordinates`, and how they transform between each other. The frames within the blue box are implemented in `astropy.coordinates`, but in the shared framework, any frame can be transformed to any other frame in this diagram.

4.4. Units and Time Scales

Solar observations include physical quantities measured at specific times. Specifying physical quantities and time accurately and precisely are therefore fundamental to any solar data analysis task. The SunPy project addresses this need through two SEPs that apply mandates to the `sunpy` core package.

Historically, calculations using physical quantities have been performed in software using simple numerical values with no associated units. At best, the unit information might be provided in a comment or encoded in a variable name.

[55] https://ssd.jpl.nasa.gov/?horizons



(a) (b) (c)

Figure 6. Several examples of using the coordinate machinery provided by the `sunpy.coordinates` subpackage. (a) Magnetic field lines traced from a potential field extrapolation overlaid on an *SDO* AIA 171 Å observation of an active region from 2019 March 10 00:00:09 UTC. The field extrapolation was computed with `pfsspy` (Stansby 2019). (b) The Venus transit as viewed by *SDO* AIA in 1600 Å. The predicted position of Venus is overplotted in the coordinate frame of the AIA image. (c) A coronagraph image of the solar corona as observed by *STEREO-A* COR-2 with the Sun represented by a white circle in the center. The predicted positions of stars from the Gaia (Gaia Collaboration et al. 2016) Data Release 2 catalog (Gaia Collaboration et al. 2018), marked by circles, as well as Mars, marked by a box, are overplotted in the coordinate frame of the image.

However, this separation between numbers and units can lead to errors in calculation, which in some cases can have severe consequences.[56] An effort to minimize the problem of dimensional errors is described in Damevski (2009), which advocates for the solution to be provided at the software architecture layer. Following this advice, `sunpy` utilizes the unit-aware functionality provided by the `astropy.units` subpackage throughout the entire code base. This package provides support for physical quantities through a `Quantity` class, which consists of a number and its associated unit(s). These quantities can be combined in expressions with unit conversions and cancellations automatically taken into account. Tests have shown that the performance overhead incurred by this functionality is typically minimal.

SEP-0003 (Christe 2014b) formally mandates that all user-facing functionality provided by `sunpy` make use of `astropy.units`. All functions and objects must have their input constrained to the appropriate type of unit (e.g., length, mass, or energy) and return an error if the input is not correct. Inputs can then be provided with any units that match the required type (e.g., millimeters and inches are both valid units for length) and conversions occur automatically without user intervention. The `sunpy.Sun.constants` subpackage contains many standard constants relevant to solar physics with additional information such as uncertainty and reference source.

As for time, the functionality in standard `Python` through the `datetime` subpackage is inadequate for the accurate representation of time necessary for scientific uses. Solar data analysis requires certain functionality to manipulate and represent times and dates. These requirements include support for leap seconds and the ability to represent, and convert between, specific scientific timescales and formats. Given that several different time formats and scales are used within solar physics, as well as in astrophysics, it is important to have such a consistent time object with the functionality to convert between different time representations. This is of particular importance in comparative studies with multi-instrument observations which may use different time measurement systems.

SEP-0008 (Mumford 2018) mandates the use of the `astropy.time` subpackage to represent a date or time throughout the `sunpy` code base. The `astropy.time` subpackage provides the necessary support for timescales other than Coordinated Universal Time (UTC; e.g., International Atomic Time or TAI), and provides functionality for representing common time formats used in solar physics (e.g., Julian day, Unix time). The further advantage of leveraging the functionality of `astropy.time` within `sunpy` is motivated by the support it provides for leap seconds, high-precision time representation, and custom time formats and scales. The use of the `astropy.time` also fosters cross-disciplinary studies between solar physics and astrophysics, such as performing detailed timing comparison studies between solar and extra-solar events, and to accurately make use of ephemeris calculations.

4.5. Release Cycle, Versioning, and Long-term Support

A formal release schedule for `sunpy` has been adopted with version 1.0. The primary goal is to provide clarity for support of releases and to improve predictability of changes to the code base for users of `sunpy`. Two releases are planned per year with 6 months between each. In order to align with the release cycle of `astropy`, a major dependency for `sunpy`, the plan is to release each May and November. The first release of the year will be a long-term support (LTS) release, which will be supported for 12 months or until the next LTS release. The second release will be non-LTS and will be supported for 6 months or until the next release. The `sunpy` package will follow an X.Y.z versioning system where the three components have the following meaning: "X" is the LTS version number (which will be incremented with every LTS); "Y" is the release counter (which will be 0 for LTS releases and incremented for each intermediate non-LTS release); and "z" is the bug-fix counter (which will be incremented for each bug-fix release).

[56] As an extreme example, the Mars Climate orbiter mission in 1988 failed due to a unit discrepancy (The Mars Climate Orbiter Mishap Investigation Board 1999).

5. Infrastructure

The SunPy project makes use of many tools and web services to develop high-quality code and documentation consistent with best practices. This section provides an overview of these tools which enable the SunPy project to achieve its goals without central institutional oversight.

5.1. Testing

`sunpy`'s test suite can be broken down into three broad categories: offline, online, and figure tests. Offline tests are used for checking the majority of the code base. Online tests specifically test codes that make use of online web services (e.g., VSO, JSOC). These tests depend on the availability of these online services. Finally, figure tests are used to ensure that code changes do not unintentionally change plots produced by `sunpy`.

While the test suite can be run manually, it is important to run the test suite in an automated fashion to maintain the integrity of the package and to make it easier for new contributors to understand the impact of their changes. The SunPy project makes extensive use of continuous-integration services, which provide automated testing and code-change inspection. All proposed code contributions trigger test suites to be run on a number of free services (including Microsoft's Azure Pipelines,[57] CircleCI,[58] and Codecov,[59]) which integrate into GitHub. These services provide the first review of any contribution by running the test suite on each operating system (Windows, Mac, and Linux), testing the documentation build, making comparison plots, and providing code-coverage metrics. Additionally, Travis CI[60] is used to run the entire test suite on a daily cadence to check for any changes in behavior due to changes in packages that `sunpy` depends on.

5.2. Documentation and Gallery

The SunPy project strives to provide up-to-date, approachable, and high-quality documentation. All documentation for `sunpy`, as well as all affiliated packages, uses the `Sphinx` documentation build system.[61] This system supports using plain text files with a markup language called `reStructuredText`. The build process converts these files, including documentation strings in `Python` files, into HTML, PDF, or LATEX documents. We make use of the `sphinx-gallery`[62] extension to build a gallery of analysis examples and the `sphinx-automodapi`[63] extension to generate documentation pages that list all of the available classes, functions, and attributes. Our online documentation[64] is automatically built and hosted on Read the Docs[65] for all releases and the latest code base.

6. Affiliated Packages

In order to foster collaboration, coordination, and code reuse, the SunPy project supports affiliated packages. These are `Python` packages that build upon the functionality of `sunpy` and/or provide general functionality useful to solar physics. Affiliated packages can also be used to develop and mature subpackage functionality outside of the `sunpy` core package for eventual inclusion. The following requirements must be met by affiliated packages:

1. The package must make use of all appropriate features in `sunpy`.
2. Documentation must be provided that explains the function and use of the package, and should be of comparable quality to `sunpy` documentation.
3. A test suite must be provided to verify the correct operation of the package.

For a package to become affiliated, developers can apply to the lead developer, and the SunPy board provides the final approval. Packages are re-reviewed on a yearly basis to ensure that they continue to meet the standards. To aid discoverability, all affiliated packages are listed on the SunPy website,[66] and they are advertised at national and international conferences and workshops. Finally, the development of these packages is provided informal support by the SunPy developer community, and a package template[67] has been developed that simplifies and standardizes packaging, testing, and documentation.

The following sections provide short descriptions of the existing affiliated packages.

6.1. drms

The `drms` (Glogowski et al. 2019) package provides functionality to access data hosted by the JSOC. Operated by Stanford University, JSOC is the primary data center for *SDO* HMI, *SDO* AIA, the Michelson Doppler Imager (MDI) on board the *Solar and Heliospheric Observatory* (*SOHO*) instrument, and the Interface Region Imaging Spectrograph (IRIS). JSOC's Data Record Management System (DRMS) is based on a PostgreSQL database that contains metadata, as well as pointers to image data, for every image contained in the archive. The `drms` package enables querying the image metadata in the JSOC DRMS. It can also be used to submit tailored data export requests (e.g., movies and images in various formats) and download data files. `Fido` depends on this package to enable it to access JSOC search results.

6.2. Ndcube

The `ndcube` package provides functionality for manipulating *N*-dimensional coordinate-aware data. Support is provided for any combination of axis types such as images (2 spatial axis), images over time (2 spatial and 1 time axis), spectrograms (wavelength and time), as well as more complex data sets (e.g., slit spectragraphs with wavelength, time, and spatial axes). The package provides the `NDCube` class, a subclass of Astropy's `NDData` data container that holds together the data array, data uncertainties, and (potentially) a data mask. `NDCube` enhances the functionality of `NDData` by adding support for handling coordinate transformations through the World Coordinate System (WCS) architecture commonly used in solar physics (Greisen & Calabretta 2002; Thompson 2006). Powerful and intuitive tools are included for slicing data sets

[57] https://azure.microsoft.com/en-us/services/devops/pipelines/
[58] https://circleci.com
[59] https://codecov.io
[60] https://travis-ci.org
[61] http://www.sphinx-doc.org/en/master/
[62] https://sphinx-gallery.github.io
[63] https://sphinx-automapi.readthedocs.io
[64] http://docs.sunpy.org/en/stable/
[65] https://readthedocs.org/
[66] https://sunpy.org/team#affiliated-packages
[67] https://github.com/sunpy/package-template

with a single command (using either array indices or coordinates) or slicing all components (including the data array and coordinates simultaneously). This functionality enables users to manipulate their data set quickly and accurately, allowing them to more efficiently and reliably achieve their science goals, and is meant to be used as a basis for more advanced and instrument-specific functionality (see Section 6.4). Support for the generalized WCS module (Dencheva et al. 2018) is planned for the next major release.

6.3. Radiospectra

The `radiospectra` package supports reading and analyzing dynamic radio spectra, as a function of time, primarily from e-Callisto, the International Network of Solar Radio Spectrometers.[68] It provides tools for downloading and reading data, handling metadata, homogenizing data, and defining and subtracting background. This package is planned to undergo major changes to use new tools being developed in `astropy/specutils`.[69]

6.4. IRISPy

The `IRISPy` package provides tools to read, manipulate, and visualize data from IRIS (De Pontieu et al. 2014). IRIS is a NASA Small Explorer mission with two instruments: the Slit Jaw Imager (SJI) and a rastering slit spectrograph. `IRISPy` is currently limited to reading level 2 data for either of these instruments. This package provides data classes which hold data from SJI and the slit spectrograph. Built on top of the functionality provided by `ndcube` (see Section 6.2), `IRISPy` links the main observations, metadata, uncertainties, data unit, mask, and WCS transformations and provides easy slicing of the data in any axis. Measurement uncertainties accounting for Poisson statistics and readout noise are automatically calculated, while a mask is used to ignore bad pixels when performing basic operations (e.g., mean, maximum, and visualizations).

7. Support and Sustainability

To date, the SunPy project relies largely on unpaid, volunteer efforts from early career scientists. The project has not received any significant direct financial support for its work facilitating and promoting open source and open development, including developing the `sunpy` package itself. This lack of financial support presents a number of challenges that the Astropy community also faces and have been described in Price-Whelan et al. (2018) and Muna et al. (2016).

The National Academies of Sciences, Engineering, and Medicine (NASEM) report on Open Source Software Policy Options for NASA Earth and Space Sciences (National Academies of Sciences, Engineering, and Medicine 2018) outlines several solutions to alleviate these problems—namely that the NASA Science Mission Directorate provide funding for new and existing open source software projects, promote scientists who spend time developing and improving open source software, and offer prizes for exemplary contributions to the open source software community. The NASEM report on Reproducibility and Replicability in Science (National Academies of Sciences, Engineering, and Medicine 2019) recommends that funding agencies invest in the research and development of open source software that support reproducibility. The SunPy project supports solutions like these for all relevant funding agencies (Christe et al. 2019) and furthermore has the ability to accept financial contributions from institutions or individuals through the NumFOCUS[70] organization. NumFOCUS is a 501 (c)(3) public charity that collects and manages tax-deductible contributions for many open source scientific software projects such as NumPy and Astropy.

The SunPy project participates in two summer-of-code programs, as part of OpenAstronomy[71], which offer stipends for students to contribute to open source projects. OpenAstronomy is a collaboration between open source astronomy, astrophysics, and heliophysics projects to share resources, ideas, and to improve code and support open development. Since 2013, 16 students have contributed to `sunpy` and affiliated packages through the Google Summer of Code[72] program. Through a similar program called the European Space Agency Summer of Code in Space[73] program, an additional 7 students have contributed since 2011. Many SunPy community members volunteered to serve as mentors.[74]

8. Community

As discussed in Bangerth & Heister (2013), a vital component to the long-term success of a project is building a healthy community of users, developers, and maintainers. The SunPy project has taken a number of steps to nurture and grow the SunPy community. We have adopted and enforce a formal code of conduct[75] that strives to foster an open, considerate, and respectful environment for all. Additionally, we maintain a number of active communication channels including mailing lists, real-time chats based on the open source Matrix protocol,[76] and weekly teleconferences. SunPy community members have also lead tutorials at national and international conferences, participated in summer programs, and provided mentorship to new members of the community. Developers and contributors are currently provided credit for their work by authorship on papers, posters, and release notes, as well as awards. For example, in 2018, Vishnunarayan K. I. was presented an award by NumFOCUS for exceptional contributions to the open source scientific software ecosystem by updating `sunpy` to use `astropy.time`.

9. Conclusion

Development of the `sunpy` core package has been ongoing for 8 yr, with the adoption of a formal project structure 5 yr ago. The core package has grown to provide significant and now mature functionality for a growing number of users. The release of version 1.0 is a significant milestone, and comes with a commitment to stability in future releases. Significant additional features are either being actively developed or are planned for future development. These planned udpates include support for generic spectra (one-dimensional or multidimensional), multidimensional data sets (e.g., slit spectrographs),

[68] http://www.e-callisto.org
[69] https://github.com/astropy/specutils
[70] https://numfocus.org/
[71] https://openastronomy.org/
[72] https://summerofcode.withgoogle.com/
[73] https://socis.esa.int/
[74] https://github.com/sunpy/sunpy/wiki/Wall-of-Fame
[75] https://docs.sunpy.org/en/stable/code_of_conduct.html
[76] https://matrix.org

and a standardized approach to metadata. The roadmap is maintained in a repository[77] to enable community discussion and suggestions.

The project formalization process, which defined a board structure, has succeeded in providing project stability as well as better recognition in the community. Through the board, significant decisions were able to be made such as joining NumFOCUS and adopting an official code of conduct,[78] the purpose of which is to ensure that the SunPy community is positive, inclusive, successful, and growing.

There are a number of obstacles to the continued growth and success of the project. The project does not yet have a significant and long-term funding stream. In addition to that challenge, the current core team is relatively small, which translates to an unhealthy dependence on key developers. A significant obstacle to the growth of the core developer team is the low familiarity of modern software engineering best practices in the solar community. Core developers of `sunpy` or affiliated packages must be knowledgeable of version control, code refactoring for public use, good user documentation, and unit testing. It takes considerable effort to provide new contributors with the appropriate guidance and training. The SunPy project is considering a number of ways to address this issue, including providing webinars, internship opportunities, and improving online documentation.

The SunPy project is now a member of the Python in Heliophysics Community,[79] whose members contribute to a collection of over 50 `Python` packages that span every subdiscipline within heliophysics. Burrell et al. (2018) provides an overview of the current state of `Python` packages in heliophysics and calls for a common framework to be developed. The development of `sunpy` is consistent with the standards established by the community (Annex et al. 2018) and many members of the SunPy project are signatories to these standards. The goal of this group is to coordinate `Python` development for heliophysics in order to improve interoperability and efficiency. In addition, some of the functionality provided by `sunpy` subpackages may be broadened to be generally useful for all of the heliophysics subdisciplines.

We thank the members of the community that have contributed to the SunPy project, that have opened issues and provided feedback, and that have supported the project in a number of other ways. We would like to acknowledge two new members of the SunPy board who joined while this paper was being prepared: Bin Chen and Tiago Pereira. Additionally, we thank Brian Dennis, Richard Schwartz, and William Thompson whose comments and discussion greatly improved this manuscript.

The following individuals recognize support for their personal contributions. B.M.S. is supported by the NSF grant AST-1715122 and acknowledges support from the DIRAC Institute in the Department of Astronomy at the University of Washington. The DIRAC Institute is supported through generous gifts from the Charles and Lisa Simonyi Fund for Arts and Sciences, and the Washington Research Foundation. D.S. was supported by STFC studentship ST/N504336/1 and STFC grant ST/N000692/1.

We acknowledge financial contributions from Google as part of the Google Summer of Code program and from the European Space Agency as part of the Summer of Code in Space program. We acknowledge financial contributions from NumFocus for improving the usability of SunPy's Data Downloader. Additionally, we acknowledge current and future funding from the Solar Physics Division of the American Astronomical Society for SunPy workshops and tutorials at annual meetings.

This work has made use of data from the European Space Agency (ESA) mission *Gaia*,[80] processed by the *Gaia* Data Processing and Analysis Consortium (DPAC).[81] Funding for the DPAC has been provided by national institutions, in particular, the institutions participating in the *Gaia* Multilateral Agreement.

This research made use of `astropy`,[82] a community-developed core Python package for Astronomy (Astropy Collaboration et al. 2013; The Astropy Collaboration et al. 2018).

Software: `astropy` (Collaboration 2018), `astroquery` (Ginsburg et al. 2019), `drms` (Glogowski et al. 2019), `GWCS` (Dencheva et al. 2018), `matplotlib` (Caswell et al. 2019), `ndcube`, `numpy` (v1.14.5 van der Walt et al. 2011), `pandas` (McKinney 2010), `pfsspy` (Stansby 2019), `scipy` (Jones et al. 2001), `sunpy` (Mumford et al. 2019).

ORCID iDs

Will T. Barnes https://orcid.org/0000-0001-9642-6089
Monica G. Bobra https://orcid.org/0000-0002-5662-9604
Steven D. Christe https://orcid.org/0000-0001-6127-795X
Nabil Freij https://orcid.org/0000-0002-6253-082X
Laura A. Hayes https://orcid.org/0000-0002-6835-2390
Jack Ireland https://orcid.org/0000-0002-2019-8881
Stuart Mumford https://orcid.org/0000-0003-4217-4642
David Perez-Suarez https://orcid.org/0000-0003-0784-6909
Daniel F. Ryan https://orcid.org/0000-0001-8661-3825
Albert Y. Shih https://orcid.org/0000-0001-6874-2594
Prateek Chanda https://orcid.org/0000-0002-7068-2866
Kolja Glogowski https://orcid.org/0000-0002-1361-5712
Russell Hewett https://orcid.org/0000-0001-8944-4705
V. Keith Hughitt https://orcid.org/0000-0003-0787-9559
Andrew Inglis https://orcid.org/0000-0003-0656-2437
Michael S. F. Kirk https://orcid.org/0000-0001-9874-1429
James Paul Mason https://orcid.org/0000-0002-3783-5509
Shane Anthony Maloney https://orcid.org/0000-0002-4715-1805
Sophie A. Murray https://orcid.org/0000-0002-9378-5315
Jongyeob Park https://orcid.org/0000-0002-1063-9129
Tiago M. D. Pereira https://orcid.org/0000-0003-4747-4329
Sabrina Savage https://orcid.org/0000-0002-6172-0517
Brigitta M. Sipőcz https://orcid.org/0000-0002-3713-6337
David Stansby https://orcid.org/0000-0002-1365-1908
Rajul https://orcid.org/0000-0003-3552-8799
Trung Kien Dang https://orcid.org/0000-0001-7562-6495

References

Annex, A., Alterman, B. L., Azari, A., et al. 2018, Python in Heliophysics Community (PyHC) Standards (v1.0), Zenodo, doi:10.5281/zenodo.2529131
Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33
Bangerth, W., & Heister, T. 2013, CS&D, 6, 015010

[77] https://github.com/sunpy/roadmap
[78] http://docs.sunpy.org/en/stable/coc.html
[79] heliopython.org
[80] https://www.cosmos.esa.int/gaia
[81] https://www.cosmos.esa.int/web/gaia/dpac/consortium
[82] https://www.astropy.org

Beck, J. G. 2000, SoPh, 191, 47
Brueckner, G., Howard, R., Koomen, M., et al. 1995, in The SOHO Mission, ed. B. Fleck, V. Domingo, & A. Poland (Berlin: Springer), 357
Burrell, A. G., Halford, A., Klenzing, J., et al. 2018, JGRA, 123, 384
Caswell, T. A., Droettboom, M., Hunter, J., et al. 2019, Matplotlib/matplotlib (v3.1.0), Zenodo, doi:10.5281/zenodo.2893252
Christe, S. 2014a, SunPy Proposal for Enhancement 1: SEP Purpose and Guidelines (SEP 0001) (v1), Zenodo, doi:10.5281/zenodo.3261403
Christe, S. 2014b, SunPy Proposal for Enhancement 3: Physical Units in SunPy (SEP 0003) (v1), Zenodo, doi:10.5281/zenodo.3261707
Christe, S. 2018, SunPy Proposal for Enhancement 2: SunPy Organization Definition (SEP 0002) (v1), Zenodo, doi:10.5281/zenodo.3261663
Christe, S., Ireland, J., & Ryan, D. 2019, A Recommendation for a Complete Open Source Policy, Zenodo, doi:10.5281/zenodo.3362439
Damevski, K. 2009, in Proc. 2009 Workshop on Component-Based High Performance Computing, CBHPC '09 (ACM) 13 (New York: ACM), 18
Delaboudiniere, J.-P., Artzner, G., Brunaud, J., et al. 1995, in The SOHO Mission, ed. B. Fleck, V. Domingo, & A. Poland (Berlin: Springer), 291
Dencheva, N., Mumford, S., Sipocz, B., et al. 2018, spacetelescope/gwcs: GWCS 0.11, Zenodo, doi:10.5281/zenodo.3352400
De Pontieu, B., Title, A. M., Lemen, J. R., et al. 2014, SoPh, 289, 2733
de Wijn, A. G., Burkepile, J. T., Tomczyk, S., et al. 2012, Proc. SPIE, 8444, 84443N
Dominique, M., Hochedez, J.-F., Schmutz, W., et al. 2013, SoPh, 286, 21
Freeland, S., & Handy, B. N. 1998, SoPh, 182, 497
Gaia Collaboration, Brown, A. G. A., Vallenari, A., et al. 2018, A&A, 616, A1
Gaia Collaboration, Prusti, T., de Bruijne, J. H. J., et al. 2016, A&A, 595, A1
Garcia, H. A. 1994, SoPh, 154, 275
Ginsburg, A., Sipőcz, B. M., Brasseur, C. E., et al. 2019, AJ, 157, 98
Glogowski, K., Bobra, M., Choudhary, N., Amezcua, A., & Mumford, S. 2019, JOSS, 4, 1614
Golub, L., Deluca, E., Austin, G., et al. 2008, in The Hinode Mission, ed. T. Sukari (New York: Springer), 27
Greenfield, P. 2013, Astropy Proposal for Enhancement 1: APE Purpose and Process (APE 1) (v1), Zenodo, doi:10.5281/zenodo.1043886
Greisen, E. W., & Calabretta, M. R. 2002, A&A, 395, 1061
Guo, P. 2014, Python Is Now the Most Popular Introductory Teaching Language at Top U.S. Universities, https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext
Handy, B. N., Acton, L. W., Kankelborg, C. C., et al. 1999, SoPh, 187, 229
Hanser, F. A., & Sellers, F. B. 1996, Proc. SPIE, 2812, 344
Hill, F., Martens, P., Yoshimura, K., et al. 2009, EM&P, 104, 315
Howard, R., Moses, J., Vourlidas, A., et al. 2008, SSRv, 136, 67
Hunter, J. D. 2007, CSE, 9, 90
Hurlburt, N., Cheung, M., Schrijver, C., et al. 2012, SoPh, 275, 67
Jones, E., Oliphant, T., Peterson, P., et al. 2001, SciPy: Open Source Scientific Tools for Python, http://www.scipy.org/
Lemen, J. R., Title, A. M., Akin, D. J., et al. 2012, SoPh, 275, 17
LIGO Scientific Collaboration and Virgo Collaboration, Abbott, B. P., Abbott, R., et al. 2016, PhRvL, 116, 061102
Lin, R. P., Dennis, B. R., Hurford, G. J., et al. 2002, SoPh, 210, 3
McKinney, W. 2010, Data Structures for Statistical Computing in Python, in Proc. 9th Python in Science Conf., ed. S. van der Walt. & J. Millman (Trieste: SISSA), 51
Meegan, C., Lichti, G., Bhat, P., et al. 2009, ApJ, 702, 791
Mumford, S. 2018, SunPy Proposal for Enhancement 8: Astropy Time (SEP 0008) (v1), Zenodo, doi:10.5281/zenodo.3261794
Mumford, S., & Christe, S. 2014, SunPy Proposal for Enhancement 4: Packages Affiliated with the SunPy Project (SEP 0004) (v1), Zenodo, doi:10.5281/zenodo.3261752
Mumford, S., & Hewett, R. 2019, SunPy Proposal for Enhancement 9: Release Pattern for SunPy Core (SEP 0009) (v1), Zenodo, doi:10.5281/zenodo.3261800
Mumford, S. J., Christe, S., Freij, N., et al. 2019, SunPy, Zenodo, doi:10.5281/zenodo.3257691
Mumford, S. J., Freij, N., Christie, S., et al. 2020, JOSS, doi:10.21105/joss.01832
Muna, D., Alexander, M., Allen, A., et al. 2016, arXiv:1610.03159
National Academies of Sciences, Engineering, and Medicine 2018, Open Source Software Policy Options for NASA Earth and Space Sciences (Washington, DC: The National Academies Press)
National Academies of Sciences, Engineering, and Medicine 2019, Reproducibility and Replicability in Science (Washington, DC: The National Academies Press)
Nakajima, H., Nishio, M., Enome, S., et al. 1994, IEEEP, 82, 705
Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, Journal of Machine Learning Research, 12, 2825
Pence, W. D., Chiappetti, L., Page, C. G., Shaw, R. A., & Stobie, E. 2010, A&A, 524, A42
Price-Whelan, A. M., Sipőcz, B. M., Günther, H. M., et al. 2018, AJ, 156, 18
Rocklin, M. 2015, Dask: Parallel Computation with Blocked Algorithms and Task Scheduling, in Proc. 14th Python in Science Conf., ed. K. Huff & J. Bergstra (Trieste: SISSA), 126
Scherrer, P. H., Bogart, R. S., Bush, R. I., et al. 1995, SoPh, 162, 129
Schou, J., Scherrer, P. H., Bush, R. I., et al. 2012, SoPh, 275, 229
Seaton, D., Berghmans, D., Nicula, B., et al. 2013, SoPh, 286, 43
Seidelmann, P. K., Archinal, B. A., A'Hearn, M. F., et al. 2007, CeMDA, 98, 155
Stansby, D. 2019, dstansby/pfsspy: pfsspy v0.2.1, Zenodo, doi:10.5281/zenodo.3237053
The Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, AJ, 156, 123
The Astropy Collaboration 2018, Astropy v3.0.5: A Core Python Package for Astronomy, Zenodo, doi:10.5281/zenodo.2556700
The Event Horizon Telescope Collaboration, Akiyama, K., Alberdi, A., et al. 2019, ApJL, 875, L1
The Mars Climate Orbiter Mishap Investigation Board 1999, Mars Climate Orbiter Mishap Investigation Board Phase I Report, NASA
The SunPy Community, Mumford, S. J., Christe, S. D., et al. 2015, CS&D, 8, 014009
Thompson, W. T. 2006, A&A, 449, 791
Tsuneta, S., Acton, L., Bruner, M., et al. 1991, SoPh, 136, 37
van der Walt, S., Colbert, S. C., & Varoquaux, G. 2011, CSE, 13, 22
Verbeeck, C., Delouille, V., Mampaey, B., & De Visscher, R. 2014, A&A, 561, A29
Ware, A., Barnum, J., Candey, R., et al. 2019, Python in Heliophysics Community Meeting (v1), Zenodo, doi:10.5281/zenodo.2537188
Wilson, G., Aruliah, D. A., Brown, C. T., et al. 2014, PLoS Biology, 12, e1001745
Woods, T., Eparvier, F., Hock, R., et al. 2010, The Solar Dynamics Observatory (Berlin: Springer), 115