# sunpy

A Community Python Library for Solar Physics

**S. Christe**[1], S. Mumford[2], SunPy Board, SunPy Development Team

[1]NASA Goddard Space Flight Center, Greenbelt, MD

[2]University of Sheffield, UK

sunpy.org

github.com/sunpy/sunpy

Twitter: ehsteve

Github: ehsteve

# Background IDL

- SolarSoft (SSW) is a set of integrated software libraries, databases, and system utilities which provide a common programming and data-analysis environment for solar physics.
- SSW is modular, core package provides core tools and utilities users can add other packages which contain specialised software for different instruments or tools.
- SSW relies upon IDL (89% IDL code, 5.4% Perl)
- According to SLOCCount gen SSW is composed of 289,724 lines of code with a total approximate value of $10M (COCOMO model).
  - SDO Package 11k lines of code ($360k)
  - SOHO package 100k lines of code ($3M)
- Hosted and distributed by Lockheed Martin Solar and Astrophysics Laboratory

# What problem is SunPy trying to solve

| IDL + SolarSoft | Corollaries |
|---|---|
| Platform (IDL) is not free or open source | • Limits access |
| Solarsoft development is not open | • No easy or clear way for most users to contribute |
| Solarsoft is not version controlled | • Random code can change whenever you update.<br>• Not easy to reproduce past results.<br>• No history and can't easily revert |
| Solarsoft development is not coordinated or organized | • Much duplicated code (File conflicts: 1879 Conflicts where the code differs: 713)<br>• No standard way of doing things.<br>• No testing. |
| IDL is NOT used by other fields | • Students are not taught how to code in IDL<br>• Not marketable skill outside our community<br>• Cannot leverage work by other communities |
| Documentation is not standardized or discoverable | • People often waste time re-writing functions that already exists |

# What is SunPy

- **Goal**: To provide the fundamental software to make Python a free and viable platform for solar data analysis.
- Scientific Python stack is very powerful and maturing.
- New start means we can reorganize and rethink how we develop and share software.
  - Open source movement is now very mature.
  - Version control systems are very powerful (git and github) and social.

# Where is SunPy now

- Project began March 28, 2011 – five years ago!
- 25,000 lines of code (6500 commits)
- 83 contributors
- COCOMO cost: $350k
- 209 Issues
- 43 outstanding pull requests
- Released 0.6.2 (still beta, API in flux)
- Refereed Paper https://iopscience.iop.org/article/10.1088/1749-4699/8/1/014009

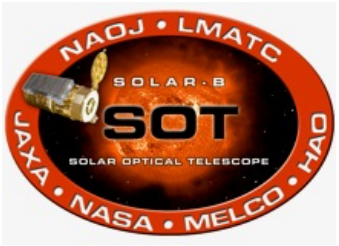
# Governance

- The primary role of the organization is to *facilitate and promote the use and development of a community-led, free and open-source solar data-analysis software based on the scientific Python environment*.
- Accomplishes this by voting on SunPy Enhancement Proposals (which can be proposed by anyone)
- Elect a lead developer
- Founding document (https://github.com/sunpy/sunpy-SEP/blob/master/SEP-0002.md)
- Membership (10 members)
  - Steven Christe (chair) – NASA GSFC
  - Russell Hewett - self
  - Andrew Inglis (secretary) – NASA GSFC
  - Jack Ireland – NASA GSFC
  - Stuart Mumford – U. of Sheffield
  - Juan Carlos Martínez Oliveros – UCB SSL
  - David Perez-Suarez (vice-chair) - U. College London
  - Kevin Reardon - DKIST
  - Thomas Robitaille - self
  - Albert Shih – NASA GSFC

# Mile-High Overview

- Focusing on calibrated (high level) data
- Focus on high level objects
- Provide a standardized and coordinated interface across data types and instruments
- Going after low hanging fruit to provide most amount of functionality with the least effort
  - Retrieving data
  - Reading data (little to no standardization in data)

# Supported Missions/Instruments

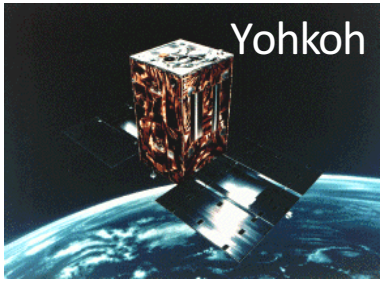





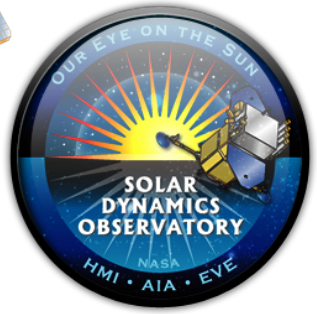



TRACE

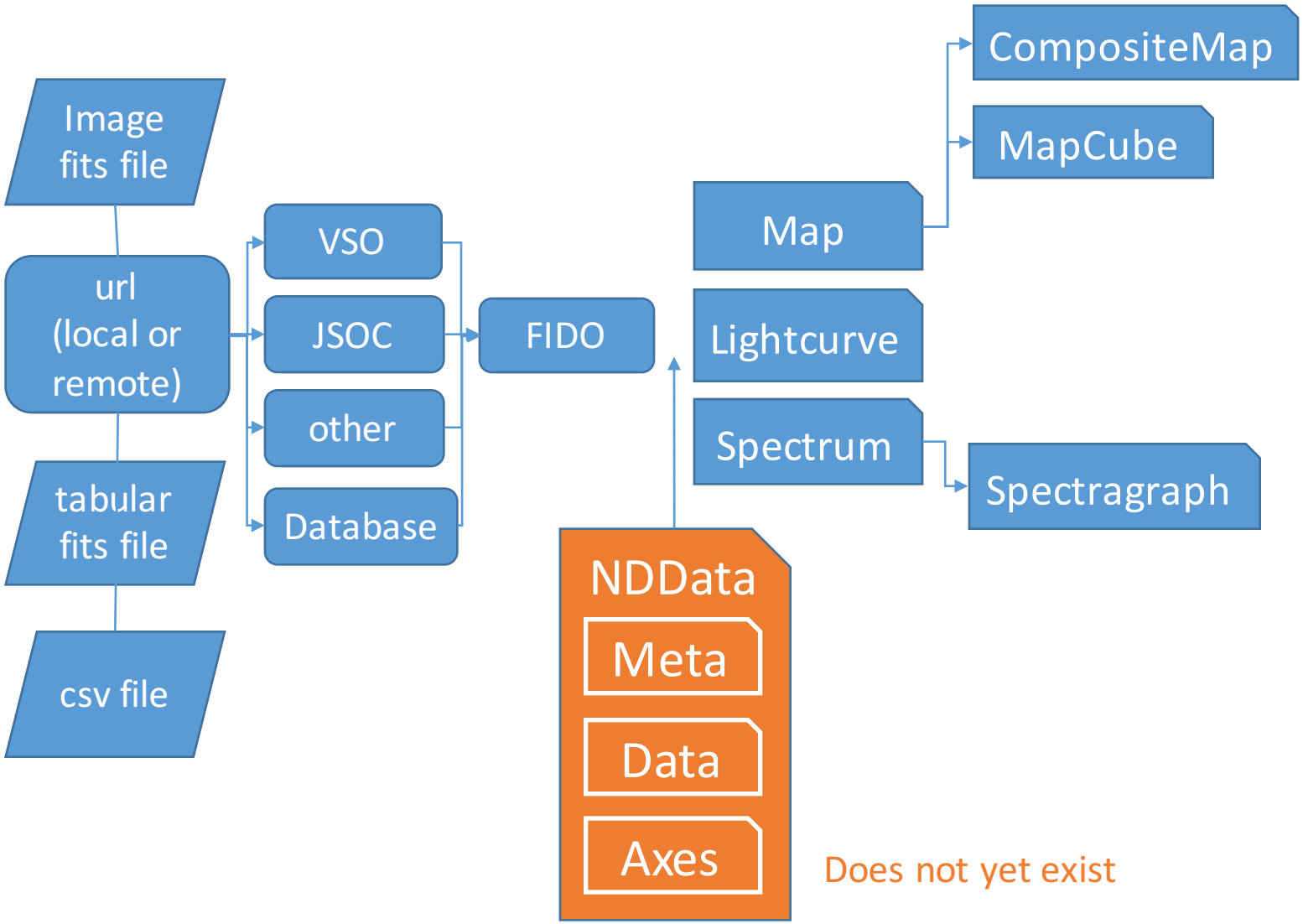

SWAP





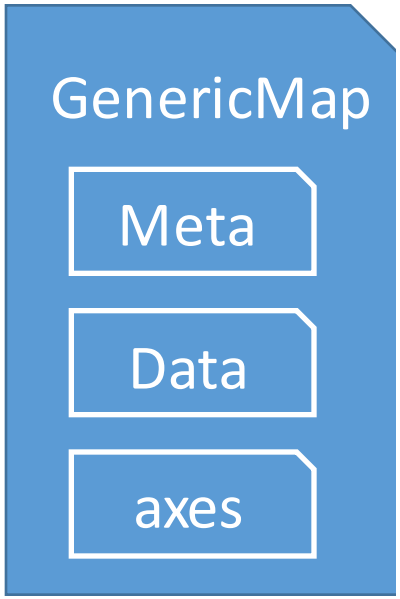


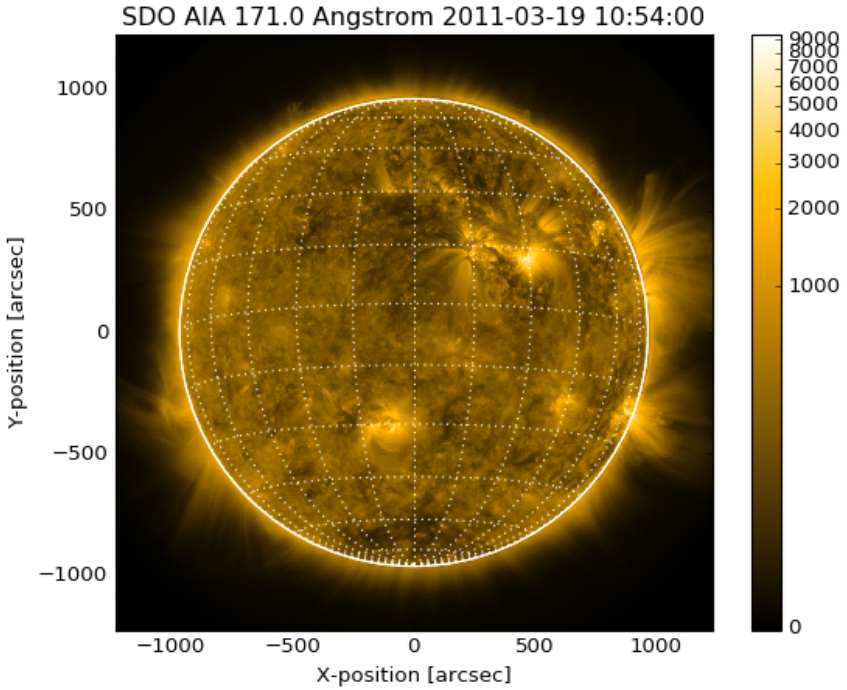

# SunPy Structure



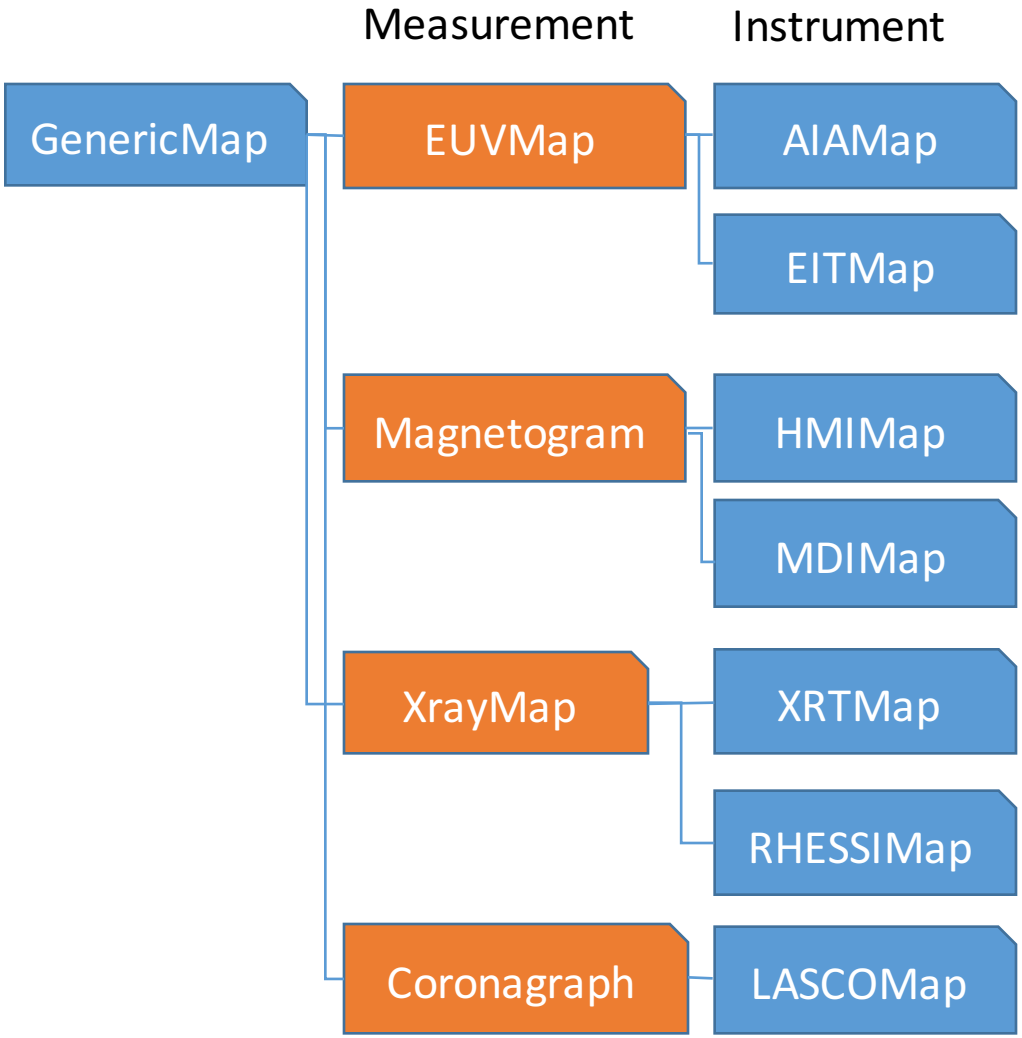
# GenericMap

- Currently most mature data structure, as it is most often used.
- Standard interface to coordinate-aware images
  - Meta data
  - Data (NDData)
  - Axes
- Plotting
- Functions
  - resample
  - rotate
  - save
  - Submap
  - superpixel
  - data_to_pixel
  - pixel_to_data
  - Max/min
- Subclassing
- Shortcuts to meta parameters



# Subclassing GenericMap




Instrument classes provide

- Placeholder for instrument specific tasks like data prepping
- Provides proper data scaling and colormaps for plotting
- Relevant documentation
- Backend uses it to standardize the data input (makes it easier for new devs to add new instrument support)

# MapCube & CompositeMap

- MapCube
  - An ordered list of same subclass of maps. By default, the maps are ordered by their observation date, from earlier maps to later maps.
  - Plotting provides animation
  - Provides alignment capabilities
- CompositeMap
  - An unordered list of any map (usually from the same time)
  - Provides tools to overplot data

# Utilities

- Solar constants (using astropy units)
- GUI
- Database (for local files)
- Solar Event information search (HEK)
- Coordinate transformations

# The Future

- More needed on other data objects
- *Daniel K. Inouye Solar Telescope (DKIST, formerly the Advanced Technology Solar Telescope, ATST)* support likely
- Google Summer of Code 2016!
- Conference Ideas
  - NDData structure discussion
  - WCS update discussion
  - Astropy models discussion
    - Add emission mechanisms (black-body, free-free bremsstrahlung, etc.)
  - Spectrum Sprint or Project
  - Meta object Sprint or Project