SunPy (0.2): Python for Solar Physics Data Analysis

V. Keith Hughitt[1], S. Christe[2], J. Ireland[1], F. Mayer[3], D. Perez-Suarez[4], A. Shih[2], S. Mumford[6], R. Hewett[7], M. D. Earnshaw[5], C. Young[1], R. Schwartz[6]

[1]ADNET Systems, NASA GSFC, [2]NASA GSFC, [3]Technische Universitat Wien, Austria, [4]Trinity College Dublin, Ireland, [5]Blackett Laboratory, Imperial College, UK, [6]University of Sheffield, UK, MIT[7]

[6]The Catholic University of America, NASA GSFC

Overview

The SunPy project is an effort to create an open-source software library for solar physics using the Python programming language.

SunPy is an open source effort and all of our code is freely available at:

http://sunpy.org

SunPy is in active development and has just released version 0.2! To browse the functionality currently available checkout the documentation at

http://sunpy.readthedocs.org/

Why Python?

Each programming environment has its own advantages and disadvantages. Python was chosen because it excels above many other languages for scientific applications. Some benefits include:

- Free and open-source.
- *Many* different sources of help available – tutorials, code snippets, forums.
- Large following in the scientific community and access to numerous multi-disciplinary libraries.

Datatype Architecture

Although there are many different data sources in solar physics, there are many fewer *datatypes*. For example, AIA and XRT both produce images of the Sun. In SunPy, data from these instruments are examples of *maps* - spatially aware images of the Sun. Objects that handle data from AIA and XRT have the same core map functionality since they are the same *datatype*. However, the AIA and XRT objects also contain functionality specific to their instrument. The *datatype* architecture ensures a more integrated data analysis environment.

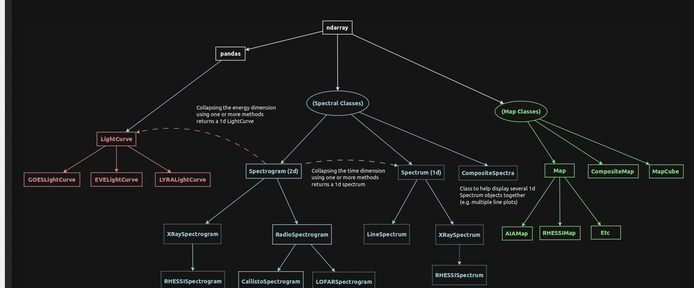

Figure 1: Architecture diagram for core SunPy datatypes showing relationships between them. Boxes with dashed outlines indicated planned classes.

Examples

Plotting a SunPy map

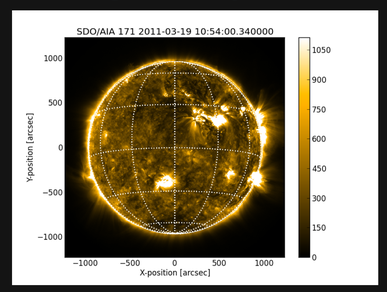

Figure 2: Venus transit as seen by SDO's AIA instrument.

One of the major focuses of development for SunPy has been to provide simple and powerful plotting capabilities. SunPy map objects are inspired by and work similarly to SSW map objects. By using Matplotlib for the underlying plotting, users gain access to the rich functionality provided by Matplotlib and are able to use any of the plot commands it supports. We provide an example fits file for the following commands but SunPy can read in fits files from any of th efollowing insttruments; SDO/AIA, Yohkoh/SXT, SOHO/EIT, SOHO/LASCO, SOHO/MDI, STEREO/EUVI, STEREO/COR PROBA2/SWAP, and Hinode/XRT.

```
import sunpy
aia = sunpy.make_map(sunpy.AIA_171_IMAGE)
aia.peek(draw_grid=30, draw_limb=True)
```

Plotting GOES light-curve data

SunPy is currently capable of working with several types of light-curve data including GOES, EVE, and LYRA. Files can be read in locally, or if dates are provided SunPy will first download the data corresponding to the specified date range and then load the data in.

```
# Example: plotting GOES data
from sunpy.lightcurve import GOESLightCurve
from sunpy.time import TimeRange
goes = GOESLightCurve.create(
    TimeRange('2012/08/07', '2012/08/08'))
goes.peek()
```

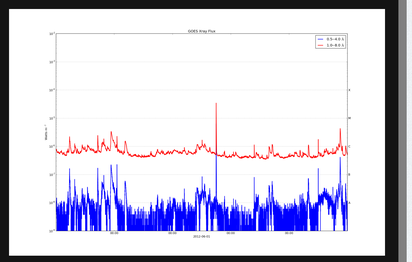

Figure 3: GOES light-curve from July 8, 2012

Plotting a Callisto spectrogram (beta)

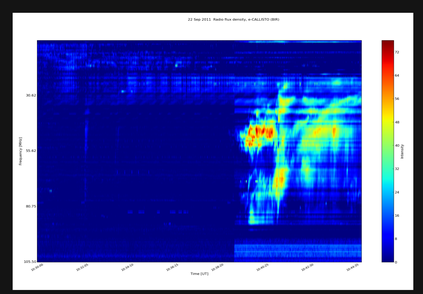

Figure 4: Callisto spectrogram with after background removal applied.

The most recent core data type to have support added in SunPy is spectral data. Initial support has been added for working with spectrum and spectrogram data.

```
# Example:Plotting a Callisto
# spectrogram with background
removed
import sunpy
from sunpy.spectra.sources.callisto
     import CallistoSpectrogram
im = CallistoSpectrogram.read(
    sunpy.CALLISTO_IMAGE
)
im.subtract_bg().show(min_=0)
```

Accessing feature and event metadata from the HEK

```
# Example: Querying HEK for flares
from sunpy.net import hek
client = hek.HEKClient()
date = hek.attrs.Time(
    (2012, 6, 1), (2012, 6,  6)
)
result = client.query(date,
hek.attrs.EventType('FL'),
  ( (hek.attrs.Event.Coord1 > 50) and
    (hek.attrs.FL.PeakFlux > 1000.0)    )
hek.attrs.FRM.Name == 'Flare Detective -
Trigger Module')
```

SunPy offers support for interacting with various online data services: users can query and download data from the Virtual Solar Observatory (VSO), create movies and screenshots using Helioviewer.org, or search for feature and event detections using the Heliophysics Event Knowledgebase (HEK).

SunPy can also perform complex queries not offered by the traditional interfaces. The code on the left is an example of an HEK query for flares detected between 2012/06/01 and 2012/06/06 using the “Flare Detective - Trigger Module” feature recognition method (FRM), with a peak flux greater than 1000.0, and west of 50 arcseconds.

Modules

SunPy includes a number of different modules, each with their own focus. Below is a brief overview of each of the major modules .

gui	Graphical interface components including a Plotman-like visualization tool.
image	Image processing routines.
inst	Instrument-specific functionality.
io	File input/output functionality.
lightcurve	Lightcurve functionality
map	N-dimensional matrix used to represent solar data.
net	Network functionality (VSO, HEK, Helioviewer, etc)
spectra	Spectrum and spectrogram functionality.
sun	Solar constants.
wcs	World Coordinate System transformations.

What's Next?

SunPy was started little over a year ago, but already much progress has been made. Initial efforts have focused on core functionality relating to acquiring and reading in data and plotting. Future efforts will continue to refine this work, and also add new features such as:

- A fully-featured graphical user interface capable of displaying and manipulating all of the main types of data supported by SunPy.
- Support for additional spectral data types such as RHESSI, SWAVES and LOFAR.
- Improved support for working with image cube data including animations.
- Performance improvements for working with large data sets.

Feedback?

Any comments or suggestions would be appreciated.

The author can be contacted at:

steven.d.christe@nasa.gov