IRISreader
A Python Library for IRIS Data Processing

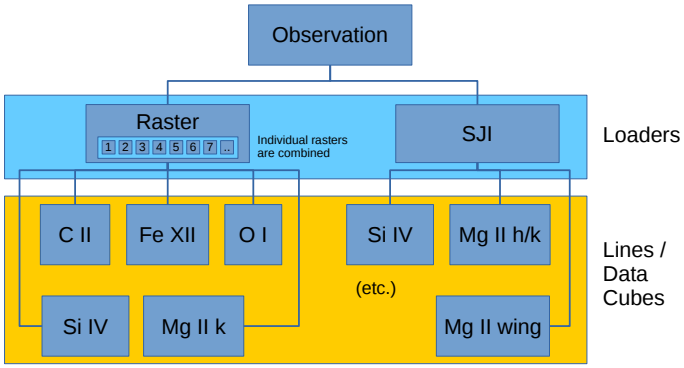
University of Applied Sciences Northwestern Switzerland
School of Engineering

UNIVERSITÉ DE GENÈVE

Cédric Huwyler[1] Lucia Kleint[1,3] Brandon Panos[1] Denis Ullmann[2] Martin Melchior[1] Säm Krucker[1,4] Sviatoslav Voloshynovskiy[2]

[1] University of Applied Sciences and Arts Northwestern Switzerland, Windisch, Switzerland
[2] University of Geneva, CUI-SIP, Geneva, Switzerland
[3] Kiepenheuer Institut für Sonnenphysik (KIS), Freiburg, Germany
[4] Space Sciences Laboratory, University of California, Berkeley, USA

How IRISreader mirrors the IRIS observation structure




```
from irisreader import observation
obs = observation("/iris/2014/09/10/20140910_112825_3860259453/")
```

```
print( obs.desc )
```

```
'Large sit-and-stare 0.3x120 1s Si IV Mg II h/k Deep x 8 FUV spec'
```

```
print( obs.sji.get_lines() )
```

	field	wavelength	description
0	FUV2	1400.0	Si IV 1400
1	NUV	2796.0	Mg II h/k 2796

```
print( obs.mode )
```

```
'sit-and-stare'
```

```
print( obs.raster.get_lines() )
```

	field	wavelength	description
0	FUV1	1335.7	C II 1336
1	FUV1	1343.3	1343
2	FUV1	1349.4	Fe XII 1349
3	FUV1	1355.6	O I 1356
4	FUV2	1402.8	Si IV 1403
5	NUV	2832.8	2832
6	NUV	2826.7	2826
7	NUV	2814.5	2814
8	NUV	2796.2	Mg II k 2796

```
obs.close()
```

The two ways of accessing lines:

Access **by index**: to iterate over all lines

```
print( obs.sji[0].n_steps )
```

```
for raster in obs.raster:
    print( raster.shape )
```

Access **by name**: to access specific lines

```
print( obs.raster("Mg").n_steps )
```

```
if obs.sji.has_line("Si IV"):
    obs.sji("Si IV").plot(0)
```

Accessing data and headers

Data cubes can be accessed directly via the index operator in the format *[time step, y pixel, x / lambda pixel]*, e.g.:

```
sji0_data = obs.sji[0][:,:,:]
mg_raster_slice = obs.raster("Mg")[10:20,400,400:500]
```

Images for single time steps can be loaded efficiently with *get_image_step()*, e.g.:

```
image = obs.raster("Mg").get_image_step( 400 )
```

Headers can be accessed through the (lazy loaded) *headers* list:

```
sat_rot = obs.raster("Mg").headers[0]['SAT_ROT']
```

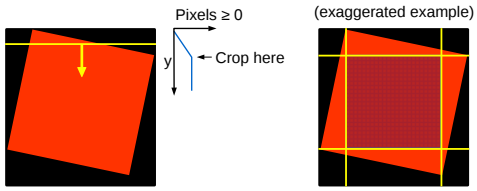
Lazy loading of data

A typical data cube in a FITS extension can take a few hundred megabytes in size. IRISreader only loads data and headers upon request and avoids time-consuming loading of unwanted data. This speeds up batch mode processing and reduces waiting times in interactive sessions.

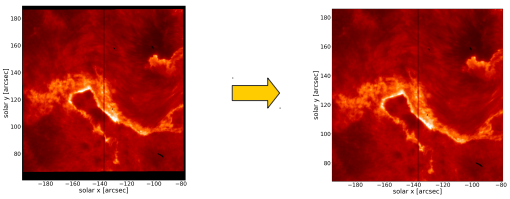
IRISreader is a Python library that allows for simple and efficient access to IRIS level 2 data in order to facilitate machine learning applications and to establish a big data supporting architecture for IRIS. The library is centered around the *OBS interface* that provides intuitive, structured access to IRIS observations. In addition, IRISreader offers functionality to efficiently iterate through years of IRIS data. We aim at creating an end-to-end machine learning research environment for IRIS data, and use this library to also provide functionality for downloading observations from the LMSAL repository and to do basic pre-processing on the data, such as efficient cropping of images and interpolating between different spectral resolutions; the library is particularly suited for interactive use in Jupyter notebooks. IRISreader is tailored specifically for big data processing and has no intention to compete with existing Python frameworks (such as *sunpy/IRISpy*) that are more directed towards science applications on single observations.

Image Cropping

Because IRIS level 2 images are rotated in order to correct for misalignment effects, one usually finds null patches (with value -200) near the image frame borders. We created an algorithm that efficiently crops out only the part of the image that is not null by moving in lines from all four sides toward the center and stopping once the number of non-negative pixels along the line stops increasing:




```
obs.sji("Mg II h/k").crop()
obs.sji("Mg II h/k").plot( 1190, units="coordinates" )
```



Iterating through years of observations

The following code reads the Mg spectra from all observations in a directory into a list, interpolating and cropping them to a range between 2794-2800 Å:

```
from irisreader import obs_iterator
spectra = []
lbd_min, lbd_max = 2794, 2800
for obs in obs_iterator("/iris/"):
   if obs.raster.has_line("Mg"):
      obs.raster("Mg").crop()
      for i in range( obs.raster("Mg").n_steps ):
         spectrum = obs.raster("Mg") \
            .get_interpolated_image_step(i, lbd_min, lbd_max, 100)
         spectra.append( spectrum )
```

Check IRISreader (alpha) out on Github: https://github.com/i4Ds/IRISreader

IRIS-9, Göttingen, 25-29 June 2018

Poster

5. Opportunities and challenges

irisreader - A Python Library for IRIS Data Processing

C. Huwyler[1] L. Kleint[1] B. Panos[1] D. Ullman[2] M. Melchior[1] S. Krucker[1] S. Voloshynovskyy[2]

[1] *University of Applied Sciences and Arts Northwestern Switzerland (FHNW), Institute for Data Science*
[2] *University of Geneva, Switzerland, Computer Vision and Multimedia Laboratory*

Irisreader is a Python library created at FHNW that facilitates machine learning applications on IRIS level 2 data. Machine learning libraries, such as scikit-learn, tensorflow, keras, etc. are readily available for Python, but IRIS data has very specific headers and keywords that are usually obtained using the IDL SolarSoft routine read_iris_l2.pro. Irisreader gives simple and efficient access to headers and data through a Python-based implementation of read_iris_l2.pro, thus enabling machine learning tools to directly access the data. In addition, irisreader currently allows for downloading single observations from the LMSAL repository, basic pre-processing of the data (such as cropping images and interpolating between different spectral resolutions) and basic plotting of slit-jaw images and spectra. Particularly useful features of irisreader are its ability to provide structured access to whole observations at once and to iterate efficiently through years of IRIS data. Irisreader will be released on Github before the 9th IRIS workshop.