Case Study: Acquiring Image Data from ZTF, and Displaying and Working with Images in Python

For this exercise, we will access the IRSA database for ZTF directly and determine the time and location of a supernova in M101. A series of tutorials in Python will also be explored to enable you to open images, scale and work with the images and their headers, and access some of the advanced features of image analysis within IRAF through Python.

So let's get started! This exercise assumes that you have installed the Anaconda Python distribution, and also the Pyfits packages. If you have both of those you should be able to get everything to work; most of it will work with just the Stock Anaconda package, which includes numpy. You can get instructions for installing Anaconda at this site: http://phares.caltech.edu/iptf/iptf_SummerSchool_2014/handson-intro.html.

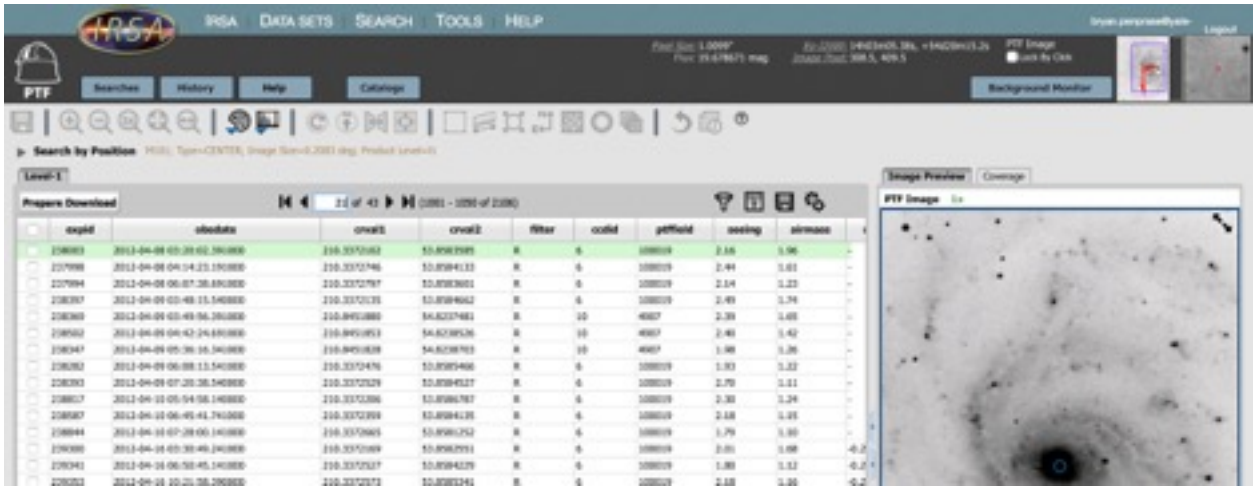
Part 1: M101 Supernova with PTF

To begin, let's use IRSA find the PTF images for the supernova in M101 which went off recently (in 2011).

You can access IRSA and its PTF data directly at http://irsa.ipac.caltech.edu/Missions/ptf.html

If you arrive at the main IRSA page, you can also see all the other missions that are available - IRSA has complete datasets for nearly every NASA infrared mission, and is a gold mine of data for your research. PTF is listed under "other missions" - and may require a login to get full access to everything. Jason Surace can enable you to have such access - just create an account from the IRSA page, and email Jason (jason@ipac.caltech.edu) to request this full access. For today you can use my login which is email *bryan.penprase@yale-nus.edu.sg* and password *ztfrocks*. You should see a page like the one below - this is the entry point!

Choose the PTF image service, and enter your target information below. For our search we are using the M101 galaxy; the IRSA server can resolve names and find coordinates - this include names for supernovae and other transient events. (HINT: For the M101 supernova, it is helpful to give the software a larger image size than the 500 arc second default; I used 750 and found the SN near the top of the image).



Scroll through the vast number of images and see if you can spot the supernova! Find out when the first image was taken with ZTF that showed a hint of the supernova. What are the coordinates of the supernova? How quickly does it rise? (the cursor gives flux per pixel and coordinates; you can use this to get an approximate light curve).

You can check boxes and stage the images for download that you might want to use. Try to see if you can get one of the images downloaded, and display it with your favorite image display program (I use ds9; it is available at http://ds9.si.edu/site/Download.html.

Take a moment and be awed by the vast amount of data at your finger tips! How could this database help you in your summer research? Try to see if there are ZTF holdings for objects that you are studying (quasars, variable stars, etc)!

Part 2: Basic Python and Image Display

There are a really amazing number of Python tutorials and resources available now. I have provided a summary of them at our web site at http://ztf.commons.yale-nus.edu.sg/python-and-resources/.

After trying many of the tutorials, I think I have found 2-3 that are most useful. You may fly through these, with everything working - or not! My experience is that it is really important to be sure you have installed Python (and perhaps some additional packages), and then start Python correctly. Finally you need to load packages into Python - most of my errors came from not pre-loading packages, and Python is very dumb about trying to find packages that are not explicitly loaded.

A very good first tutorial for Python is at the python4astronomers site at http://python4astronomers.github.io/plotting/matplotlib.html.

To run this, you will probably need to start up Python in a Terminal with

> ipython —matplotlib

For me this gives this set of lines:

```
bep04747-macair:~ bpenprase$ ipython --matplotlib
Python 2.7.9 |Anaconda 2.0.1 (x86_64)| (default, Dec 15 2014,
10:37:34)
Type "copyright", "credits" or "license" for more information.

IPython 3.0.0 -- An enhanced Interactive Python.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://binstar.org
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
Using matplotlib backend: MacOSX

In [1]:
```

The line is your input line, and you can paste commands into there. For the image display tutorial, I was able to get (nearly) everything to work - but only after pre-loading some routines.

You will also need to preload some images for the exercise - this would include for this first part the image known as stinkbug.png - it is on our web site.

Here are the four lines for loading in the packages you will need:

```
In [1]: import pyfits

In [2]: import numpy as np

In [3]: import pylab as py

In [4]: import img_scale
```

Then after that you should be able to run the full tutorial for image display - and changing image scales, and making nice plots at the site:

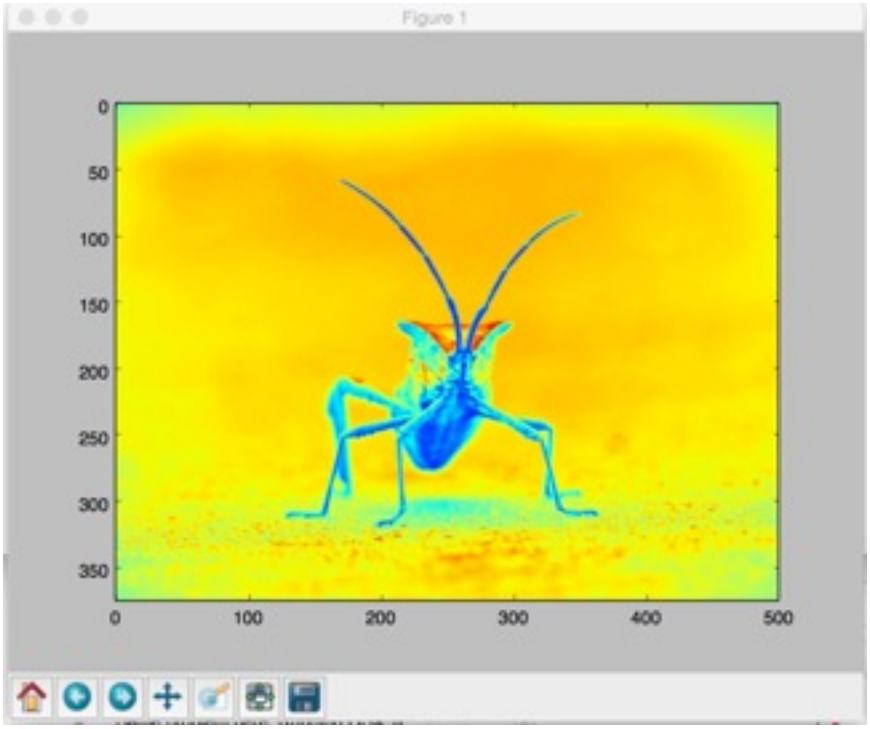
http://matplotlib.org/users/image_tutorial.html

Some of the first parts of that tutorial are below for convenience:

```
In [4]: img=mpimg.imread('stinkbug.png')
Out[4]:
In [5]: imgplot = plt.imshow(img)
(Source code, png, hires.png, pdf)
```

../_images/image_tutorial-1.png
You can also plot any numpy array - just remember that the datatype must be float32 (and range from 0.0 to 1.0) or uint8.

Before long - you should be able to produce these kind of cool looking bug images!

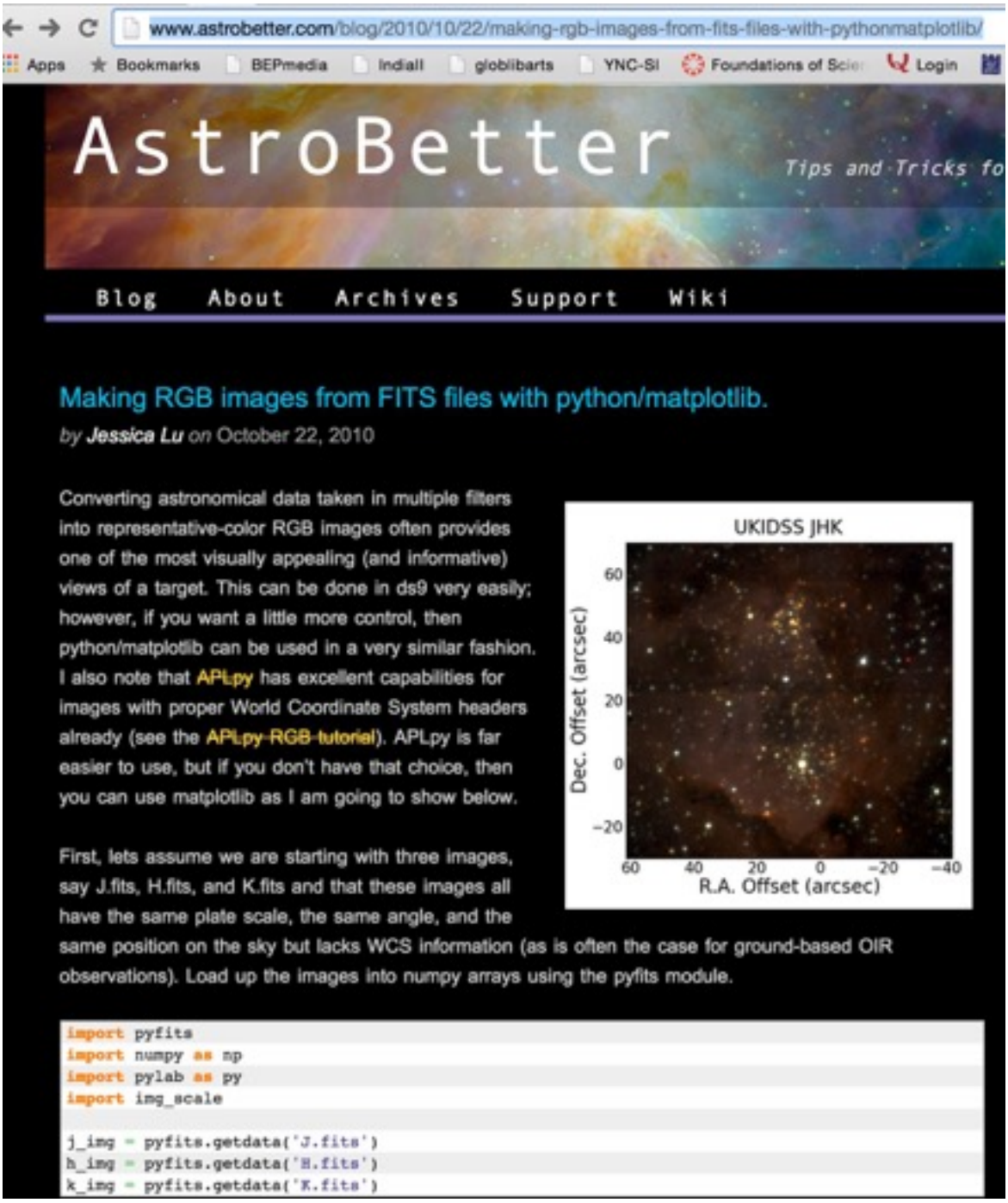


Part 3: More advanced Image Display

If you got here by our 40 minute session - I am amazed! However perhaps for later you can try these tutorials. This one leads you through an exercise of creating a tricolor image out of three monochrome filtered images. You can use the HST images on our site but will need to save them to the file names of “J.fits” “H.fits” and “K.fits” to paste in the commands. For convenience I used the three images at the site http://www.astro.yale.edu/mgeha/Singapore/Arp194/ and copied them to J.fits, H.fits and K.fits.

Start up the Python in the same way and work through the tutorial at:

http://www.astrobetter.com/blog/2010/10/22/making-rgb-images-from-fits-files-with-pythonmatplotlib/



Note that for a tricolor image with our HST images - image scaling commands need different values. These commands worked for me to produce a somewhat green tricolor image:

```
In [88]: img[:,:,0] = img_scale.sqrt(k_img, scale_min=0,
scale_max=0.01)
In [89]: img[:,:,1] = img_scale.sqrt(h_img, scale_min=0,
scale_max=0.01)
img_scale : sqrt
In [90]: img[:,:,2] = img_scale.sqrt(j_img, scale_min=0,
scale_max=0.01)
```

The rest of the exercise should work well – good luck! A very large reward to anyone who can find the images that go with the Tutorial (the J.fits, H.fits, and K.fits images).

Part 4: Even More advanced Image Display

As part of the PTF/ZTF boot camp, we have an exercise which is extremely complete. You can use the images from the above exercise to examine image headers, move around in the images, and produce a grey scale display image.
This exercise is at:

http://phares.caltech.edu/iptf/iptf_SummerSchool_2014/astropy-boot-camp.html

You can use the images from the above exercise in the part of the tutorial that covers image display. Just load in the requisite packages first:

```
import astropy.coordinates
import astropy.units as u
import astropy.io.fits
import astropy.stats
import astropy.table
import astropy.wcs
import astropy.cosmology
import scipy.optimize
import scipy.odr
```

Then you can use these command to open up the image, display image headers, and do lots of other things – ENJOY!

```
In [23]: import astropy.table

In [24]: import astropy.wcs

In [25]: import astropy.cosmology
```

```
In [26]: import scipy.optimize

In [27]: import scipy.odr

In [28]: fits = astropy.io.fits.open('J.fits')

In [29]: hdu = fits[0]

In [30]: hdu.header
Out[30]:
SIMPLE  =                    T / Fits standard
BITPIX  =                  -32 / Bits per pixel
NAXIS   =                    2 / Number of axes
NAXIS1  =                 4200 / Axis length
NAXIS2  =                 2800 / Axis length
```

The exercise leads through a useful image display and analysis exercise which should work with any image:

Now let's plot the image data. But let's use sigma-clipping to pick a nice scale for the image.

```
clipped_data = astropy.stats.sigma_clip(hdu.data.flatten())
mid = np.median(clipped_data)
std = np.std(clipped_data - mid)
```

```
plt.figure(figsize=(20, 10))
plt.imshow(hdu.data, vmin=mid-std, vmax=mid+3*std, cmap='binary')
plt.xlabel('pixel $x$')
plt.ylabel('pixel $y$')
```

```
<matplotlib.text.Text at 0x10e95fb90>
```