August 2014

Introduction to Astronomical Data Methods

The work below is to be completed before the course start. Please contact me (marla.geha@yale.edu) with any questions.

How to Find an Asteroid: Introduction to Astronomical Data Methods: The goal of this course is to introduce you to observational methods in astrophysics, focusing on skills needed to analyze data from ground-based optical telescopes. Lectures will cover basics of optical data acquisition including telescope design and CCD cameras, data calibration and observing techniques, and the computational manipulation of digitized images. As part of class, we search for asteroids in a series of images taken with the QUEST survey telescope.

The pre-course exercises (each described in detail below) are aimed at setting up your computing environment for astronomical research:

1. Pre-Course Survey
2. Set up your computer environment
3. Exercise #1: Image puzzles in python
4. Exercise #2: Create color Hubble Space Telescope Images
5. Video introduction on the origin of asteroids

The links, required files and optional resources are available on the class website:

http://www.astro.yale.edu/mgeha/Singapore

----

1. **Pre-Course Survey:** To provide a better understanding of your background in research and programming, please fill out the brief survey linked on the course home page: http://www.astro.yale.edu/mgeha/Singapore/problemsets.html

2. **Introduction to Programming:** The goal of this section is to get you familiar with the command line and set up your programming environment. Prior programming experience in Python is not required, but I expected that you have some familiarity with general programming. We will start with an introduction to the command line and Unix commands (which you can skip if its already familiar).

The Command Line: The command line is the starting point for most astrophysics research computing. From the command line, you can move/copy files or startup other software. To open a command line terminal:

If you are using a Mac computer: Go to the Applications -> Utilities -> Terminal. Once Terminal opens use the pull down menu to open a window Shell -> New Window -> Basic, or type cmd^N.

If you are using a Windows machine: We suggest installing “winbash” to access the terminal: http://sourceforge.net/projects/win-bash/files/shell-complete/latest/
If windbash does not work, an alternative is cygwin.

Basic Unix Commands: The command line accepts commands in a language called "bash" for UNIX. We will denote UNIX commands with '>' (further below, python commands are denoted as '>>>'). A full UNIX tutorial can be found under 'Computer resources' on the class website. Read though this tutorial and execute the following tasks:

> List the contents of your home directory (> `ls`). Create a new directory under your home directory called 'Astronomy' (> `mkdir Astronomy`). From the class website, downloaded the file named example.py into your new directory. View the contents of this file using the 'more' command (> `more example.py`). This is a simple python script which we will use below.

Text Editing: When writing programs, we need to edit files without the forced formatting of programs such as Word or Powerpoint. There are a number of options for text editing. Most astronomers usually use *Sublime*, *Emacs or vi* for Mac. You can download Emacs for free (http://ftp.gnu.org/pub/gnu/emacs/) and *vi* comes preinstalled on Macs. I have recently started using Sublime for editing python code and recommend it highly:

http://www.sublimetext.com/

Sublime is available for Mac or Window users. Windows users can also try Notepad++. Inside your Astronomy directory, edit the file example.py by replacing my initials and date (MG 6/14) with you information and edit date. Exit the file and use the UNIX command > `more example.py` to verify you have created the file correctly.

Installing Python: Python programming will be our primary research tool throughout class. Python is a widely used language, and has been recently adopted by astronomers. Basic python is already installed on Macs machines, but you will need additional libraries for our work. Windows machine will need to install both python and additional libraries. There are multiple installations for scientific python libraries. I suggest the Anaconda distribution available at:

http://store.continuum.io/cshop/anaconda/

The distribution is free, but you will need to input an valid email address. If you do not know which version of this code to download, please email me to discuss. Download and follow instructions for installing this software. To test if this is successful, open a terminal and type 'ipython'. You should see the command: `In [1]:` To quit out of python type 'exit' or type control-D twice.

Basic Python: The first thing every computer programmer writes when learning a new language is a program to print 'hello world'. Follow the instructions below to write such a program (note for students new to programming, Exercise 0 will also help you through bringing up the terminal and text editing):

http://learnpythonthehardway.org/book/

I suggest working through Exercise 4, but there is useful information through #33.

Learning Python: There are many online resources for learning python. I've linked to videos and online courses on the Computer Resources. If you have no prior programming experience, I suggest working through the Google Python Class lectures (~2 hours of video).

3. Pre-Course Exercise #1: ***Image puzzles:*** Download the images puzzle1.png and puzzle2.png from the class website. Open ipython in the same directory and display each image. I suggest you read and display the images using the following commands:

```
import matplotlib.pyplot as plt
from matplotlib.image import imread

# Read in image
img = imread('puzzle1.png')

# Display image
imgplot = plt.imshow(img)
pylab.show()
```

You can type these commands directly into the python command, or run the example.py code which reads/displays puzzle1 (>>> run example.py). The image has been read into an array called img. This array has three columns of data corresponding to red, green and blue colors. See http://matplotlib.org/1.3.1/users/image_tutorial.html for more details on the structure of this array.

These files contains an image of something famous, however the image has been distorted. For puzzle1, the famous object is in the red values, however the red values have all been divided by 10. The blue and green values are all meaningless random values ("noise") added to obscure the real image. You must undo these distortions to reveal the real image. First, set all the blue and green values to 0 to get them out of the way. Look at the result. If you look very carefully, you may see the real image, although it is very dark (way down towards 0). Then multiply each red value by 10, scaling it back up to approximately its proper value. What is the famous object?

For puzzle2, the famous object is in the green and blue values, however, these have been divided by a factor of 20. The red values are meaningless random values. Solve this puzzle similar to above. What is this famous object?

4. Pre-Course Exercise #2: Making Color Images with Hubble Space Telescope Data

The Hubble Site (http://hubblesite.org/gallery/album/) has created a remarkable inventory of color images. In this exercize, we will create our own color images directly from Hubble Space Telescope data.

a. *Download HST images*: Data is available of the Problem Set page of several objects taken with the HST WFPC2 camera. For each object, there are multiple images taken in different filters (similar to the red/green/blue channels in the image puzzles above).

These images have already been processed to remove camera features and cosmic rays. You are asked to work with one of these datasets, but are welcome to work with any/all of them.

b. *Displaying fits images with ds9:* Unfortunately python doesn't yet have a good interactive tool to view images. We will use a stand alone program to display astronomical image called DS9 (stands for Deep Space 9, and yes, that's a Star Trek reference). First download the appropriate version of the code from:

http://hea-www.harvard.edu/RD/ds9/site/Download.html

For Mac users, click on the downloaded disk image (dmg) and move SAOImage DS9 file into your Applications folder. For Windows users, run the executable file to install. Opening the application will bring up a new window. Click on file -> open and import the file image.fits which you downloaded above.

c. *Display images in ds9*: First display your images in ds9. Open the ds9 and load a file (file -> open). Play with the zoom and scale until you are able to see your object. You can change the contrast with your mouse (Mac users, press command while dragging your mouse on the image). Play with the scaling of the image by clicking scale -> linear/log/etc. Move your mouse over the image and notice the numbers in the upper left panel. Click on zoom and zoom into the image so that you can see individual pixels. Click on color and change the color scale. Note that the actual numbers in this image (the numerical value of each pixel) don't change. Roughly estimate the average pixel value in the images.

d. *Read image into python*: Now read your images into python using the pyfits library. First `import pyfits` and read your image as:

```
im1 = pyfits.getdata('myimage.fits.gz')
```

The data are loaded into an array. To see the pixel values just type: `im1`. To see various math options you can do to this array, type "im1." and then press tab. For example, to determine the size of the array/image:

```
im1.size()
```

e. *Basic image statistics:* What is the minimum and maximum pixel value in your image? What is the mean value of the full image? To determine the mean of the image array: >>> `im1.mean`. Does this agree with your estimates from ds9?

f. *Display image in python*: Display your image in python:

```
plt.close()
fig=plt.figure()
plt.imshow(im1)
```

This probably doesn't look very good. Note that, unlike ds9, you cannot dynamically change the contrast or zoom in/out. How annoying! Someone should write a python tool to do this. You can manually set the min/max value displayed to improve how this looks, try typing `plt.imshow?` to list various options. Again, save a screen grab of this display.

g. You will combine three of your images to create a color image. The website below will guide you through this process:

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

This code (img_scale.py) is useful to to scale and save your images:
http://dept.astro.lsa.umich.edu/~msshin/science/code/Python_fits_image/

It is recommended that you try one filter's image at a time.

h. The Hubble Site (http://hubblesite.org/gallery/album/) has created a remarkable inventory of color images and contains color versions for each of your datasets. Look up your image and compare your work to the HLA. Do not be discouraged if your work isn't quite as pretty-- this is a difficult artform! This is not a skill that a professional astronomer uses for science, but can be very useful. Your professor only makes color images when giving public talks or creating graphics for press releases.

See also this optional paper to improve your color images:
http://arxiv.org/pdf/astro-ph/0312483v1.pdf

5. Pre-Course Video: An Introduction to Asteroids: The goal of this course is to search for asteroids and other Solar System objects in real astronomical data. The data will be be using was taken last year as part of the La Silla-QUEST survey. As an introduction to asteroids and their origin, I suggest watching these introductory videos below:

David Trilling: Assessing the Threat of Asteroid Impacts (20mins)
http://vimeo.com/channels/kfosastronomy/80004667
Kevin Walsh: Where Do Asteroids Come From (20 min)
http://vimeo.com/channels/kfosastronomy/80010139