ASTR 119: Session 18

Astropy Tutorial

Outline

1) Visualization of the Day
2) Final projects
3) Astropy / FITS Image Tutorial
4) Final project organization time
5) Save your work to GitHub

Final Projects

1) You will be asked to perform one of four final projects, working in a **group of up to 4 people of your choosing from your SECTION**. If you would like to be assigned to a group, please let me know immediately.

2) Please notify me of your group by **8:00am Thursday Nov 19**. Anyone who does not respond by **8:00am Thursday** letting me know their group or their desire to be assigned to a group will work on their own final project.

3) You will choose from one of the following 4 project topics, described in the following slides:
    1) Damped, driven pendulum
    2) Logistic map and chaos
    3) Astronomical source detection
    4) Monte Carlo Integration

4) Groups must select a final project topic by **Tues Nov 24**. Detailed instructions for each project will be distributed Thursday Nov 19.

5) Groups should be organized through GitHub, invite your group members and us (TAs) as collaborators on your project. Tag us (TAs) when your final project is ready to grade.

6) Each python module should indicate which student authored which part of the code.

7) Final projects are due (tags must happen by) **Tuesday, December 15, 2020 at 3pm.**

The Astropy Project

The Astropy Project is a community effort to develop a common core package for Astronomy in Python and foster an ecosystem of interoperable astronomy packages.

Please remember to acknowledge and cite the use of Astropy!

What's new in Astropy 3.0?

Current Version: 3.0.5

File Download

Download the session_18.tar file from the class Canvas site.

Open a terminal and use “tar” to expand the tar file into the separate files:

```
3. brant@eduroam-169-233-156-109.ucsc.edu: /Users/brant/Desktop/classes/astr_119/Session 18 (bash)
[17:55:57][brant@eduroam-169-233-156-109:~/Desktop/classes/astr_119/Session 18]$ tar -xvf session_18.tar
x ./._HorseHead.fits
x HorseHead.fits
x ./._M13_blue_0001.fits
x M13_blue_0001.fits
x ./._M13_blue_0002.fits
x M13_blue_0002.fits
x ./._M13_blue_0003.fits
x M13_blue_0003.fits
x ./._M13_blue_0004.fits
x M13_blue_0004.fits
x ./._M13_blue_0005.fits
x M13_blue_0005.fits
[17:56:07][brant@eduroam-169-233-156-109:~/Desktop/classes/astr_119/Session 18]$
```

Astropy Tutorial

Use Astropy to analyze FITS images

Based on a tutorial by Lia Corrales

In [1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from astropy.io import fits
```

Open the FITS file

In [3]:
```
fname = "HorseHead.fits"
hdu_list = fits.open(fname)
hdu_list.info()
```

```
Filename: HorseHead.fits
No.    Name         Ver    Type      Cards   Dimensions   Format
  0  PRIMARY          1 PrimaryHDU     161   (891, 893)   int16
  1  er.mask          1 TableHDU        25   1600R x 4C   [F6.2, F6.2, F6.2, F6.2]
```

Astropy Tutorial

Generally, the image information is located in the PRIMARY block. The blocks are numbered and can be accessed by indexing hdu_list.

In [4]:
```
image_data = hdu_list[0].data
```

Our data is now stored as a 2-D numpy array. But how do we know the dimensions of the image? We can simply look at the shape of the array.

In [6]:
```
print(type(image_data))
print(image_data.shape)
```

```
<class 'numpy.ndarray'>
(893, 891)
```

At this point, we can close the FITS file because we've stored everything we wanted to a variable.

In [7]:
```
hdu_list.close()
```

Astropy Tutorial

Shortcut: use "getdata()" to just read in the image data and close the file.

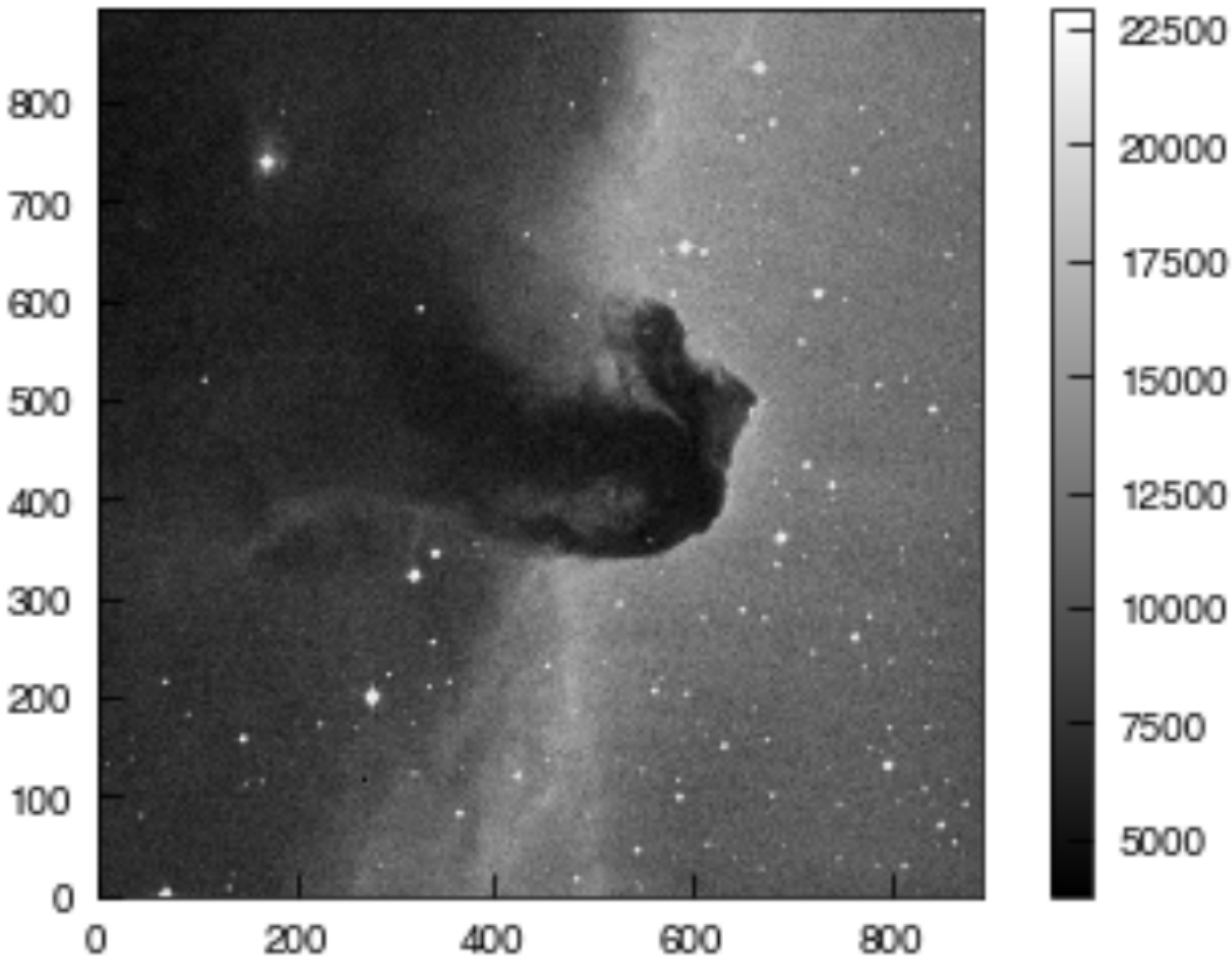
In [8]:
```
image_data = fits.getdata(fname)
print(type(image_data))
print(image_data.shape)
```

```
<class 'numpy.ndarray'>
(893, 891)
```

Let's show the data

In [9]:
```
plt.imshow(image_data, cmap='gray')
plt.colorbar()
```

Astropy Tutorial



Astropy Tutorial

Let's get some basic statistics about our image:

In [10]:
```
print('Min:', np.min(image_data))
print('Max:', np.max(image_data))
print('Mean:', np.mean(image_data))
print('Stdev:', np.std(image_data))
```

```
Min: 3759
Max: 22918
Mean: 9831.48167629
Stdev: 3032.3927542
```

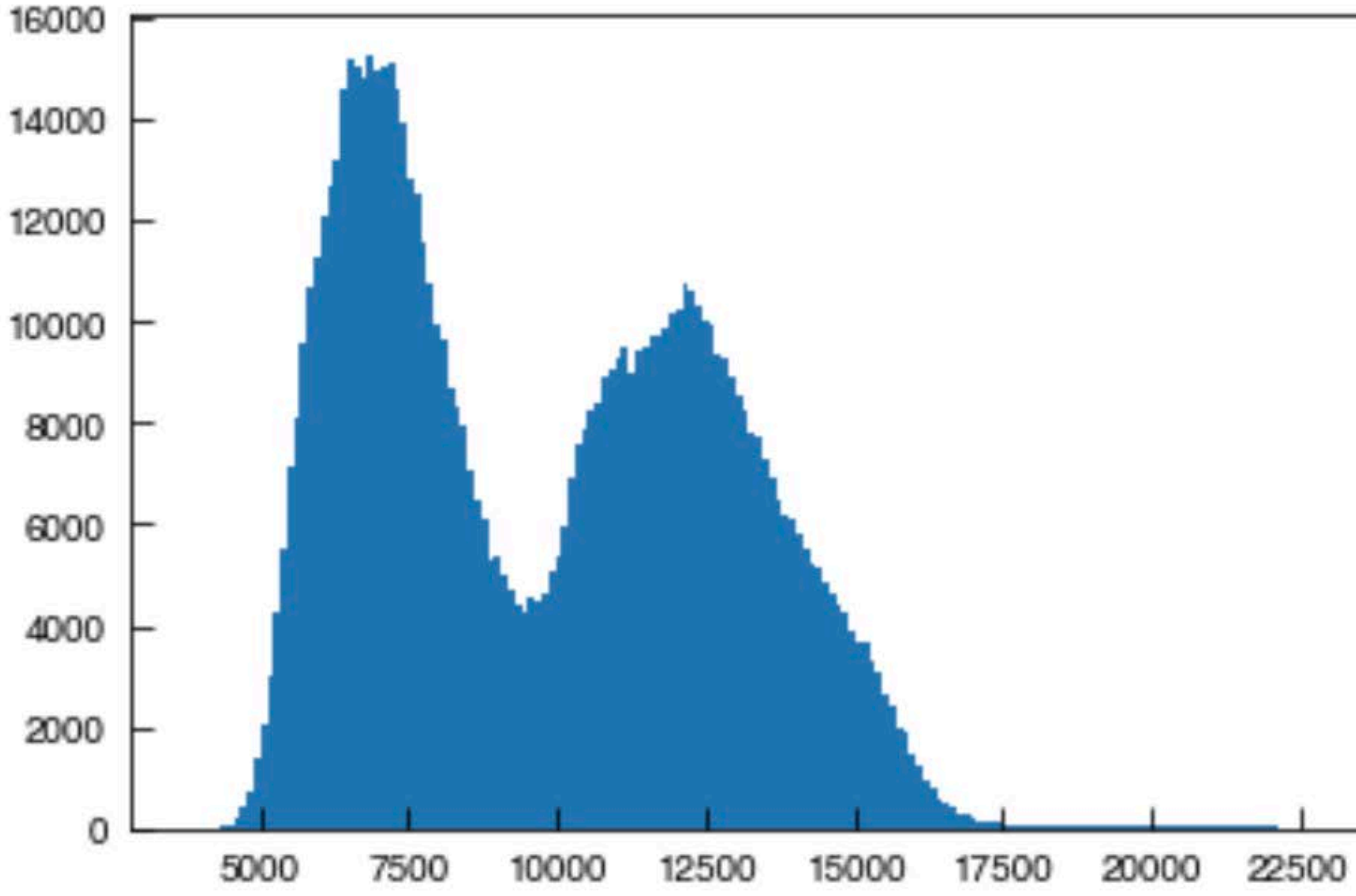
Plotting a histogram

To make a histogram with matplotlib.pyplot.hist(), we'll need to cast the data from a 2-D array to something one dimensional.

In this case, let's use the ndarray.flatten() to return a 1-D numpy array.

In [11]:
```
histogram = plt.hist(image_data.flatten(), bins='auto')
```

Astropy Tutorial



Astropy Tutorial

Displaying the image with a logarithmic scale

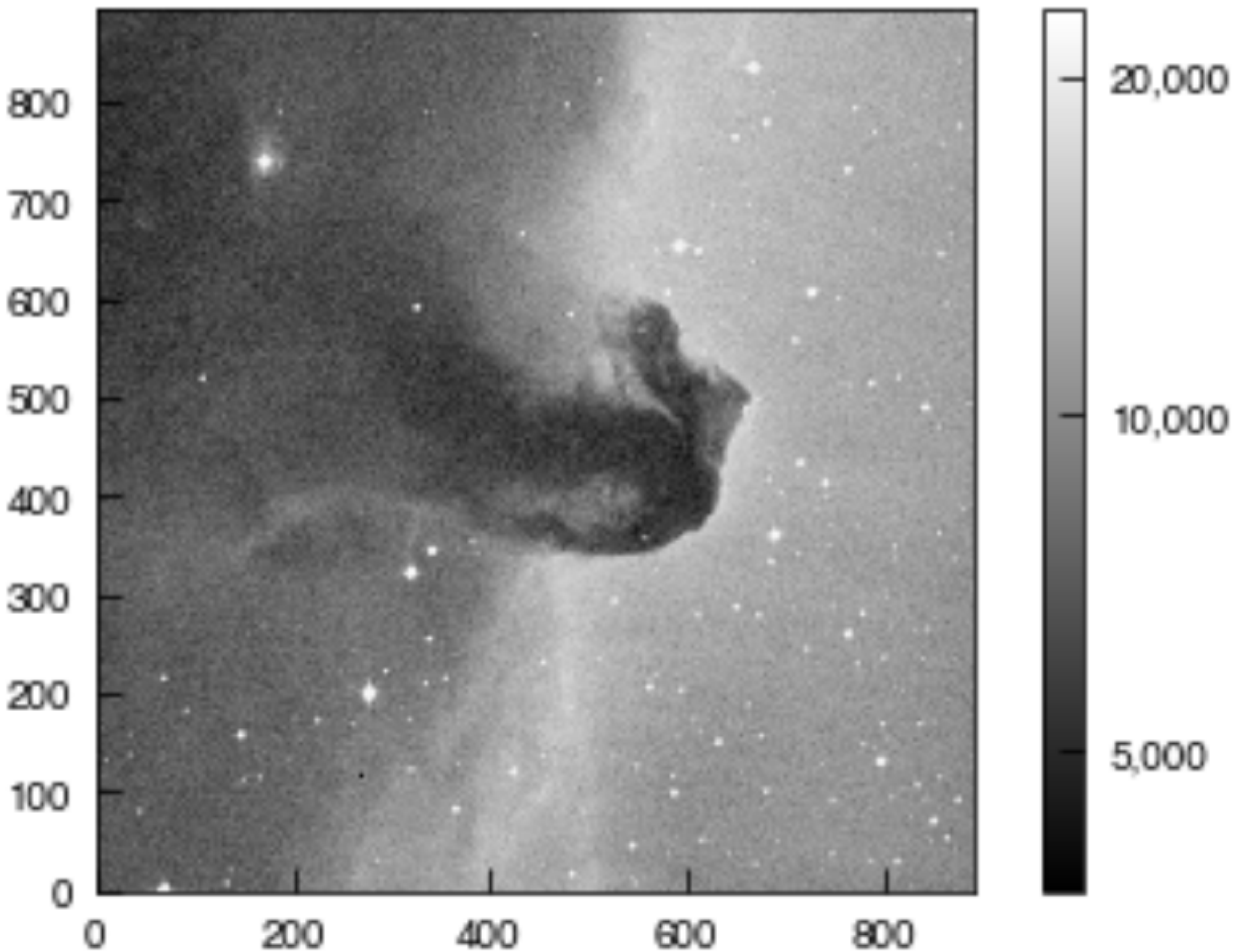
What if we want to use a logarithmic color scale? To do so, we can load the LogNorm object from matplotlib.

In [12]:
```
from matplotlib.colors import LogNorm
```

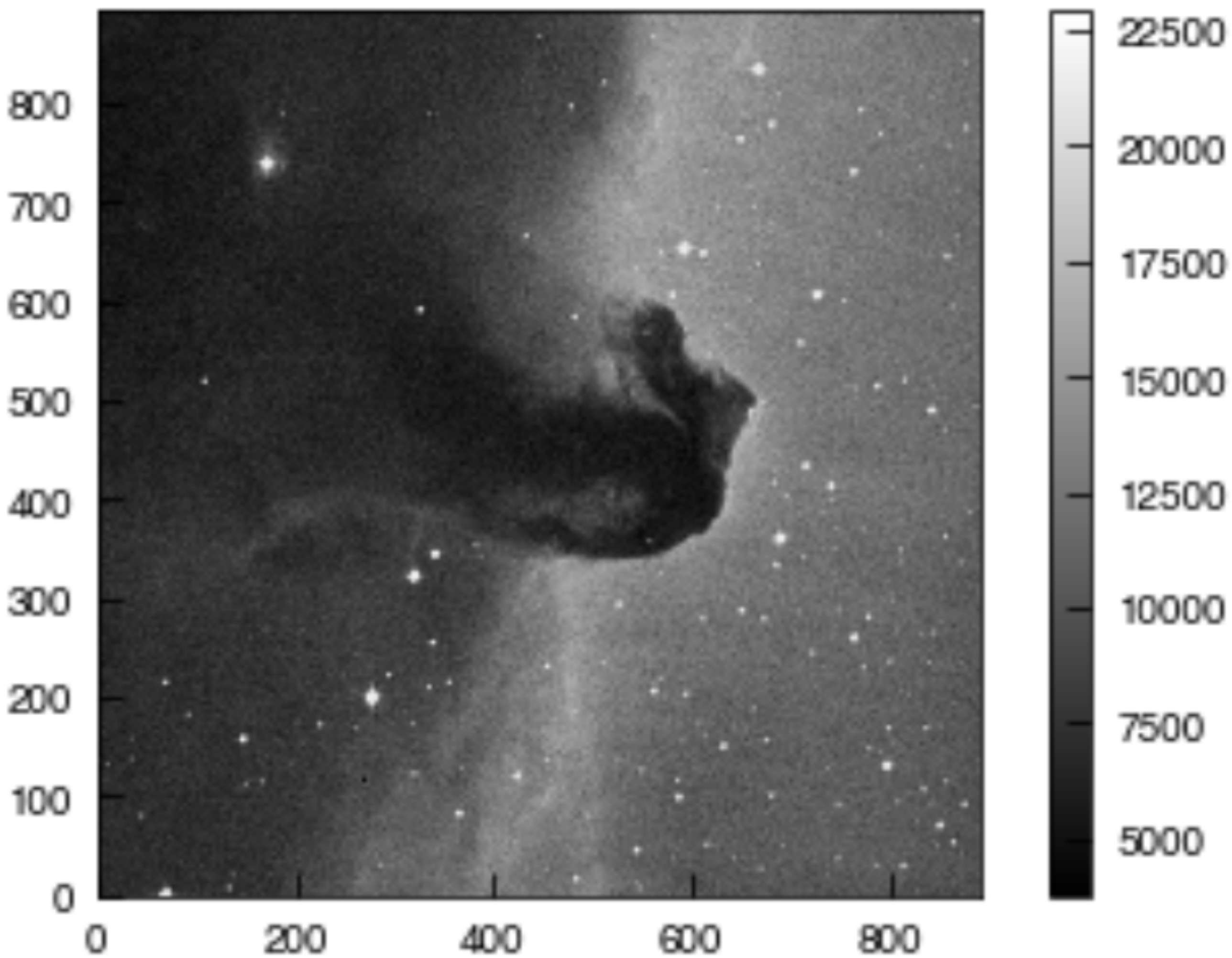
In [13]:
```
plt.imshow(image_data, cmap='gray', norm=LogNorm())

# Choose the tick marks based on the histogram above
cbar = plt.colorbar(ticks=[5.e3,1.e4,2.e4])
cbar.ax.set_yticklabels(['5,000','10,000','20,000'])
```

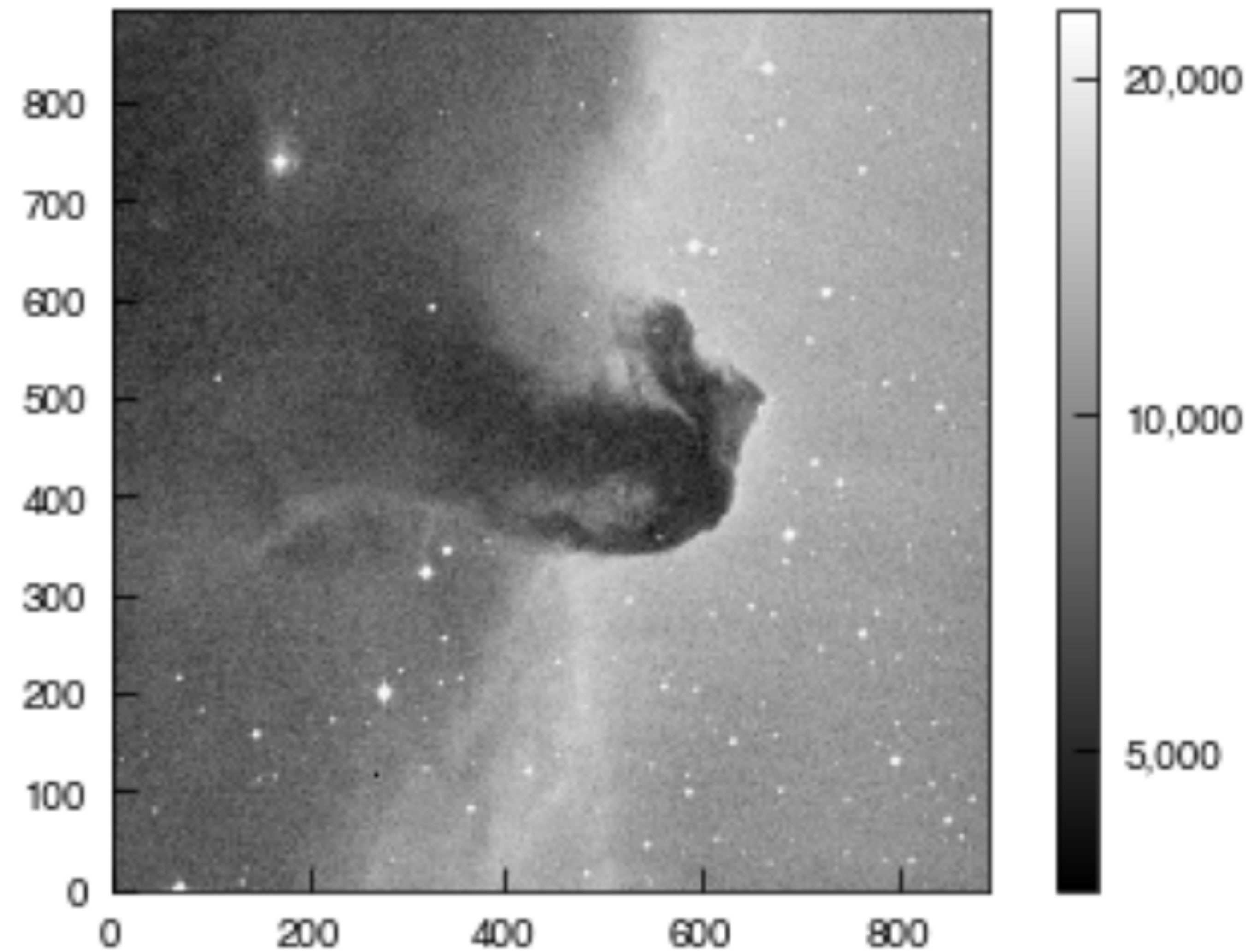
Astropy Tutorial



Astropy Tutorial



Linear



Logarithmic

UC SANTA CRUZ

Astropy Tutorial

Stacking Images

Since the noise in an image results from a random process, we use stacking of separate images to improve the signal to noise ratio of objects we observe. Here we are going to stack 5 images of M13 taken with a 10 inch telescope.

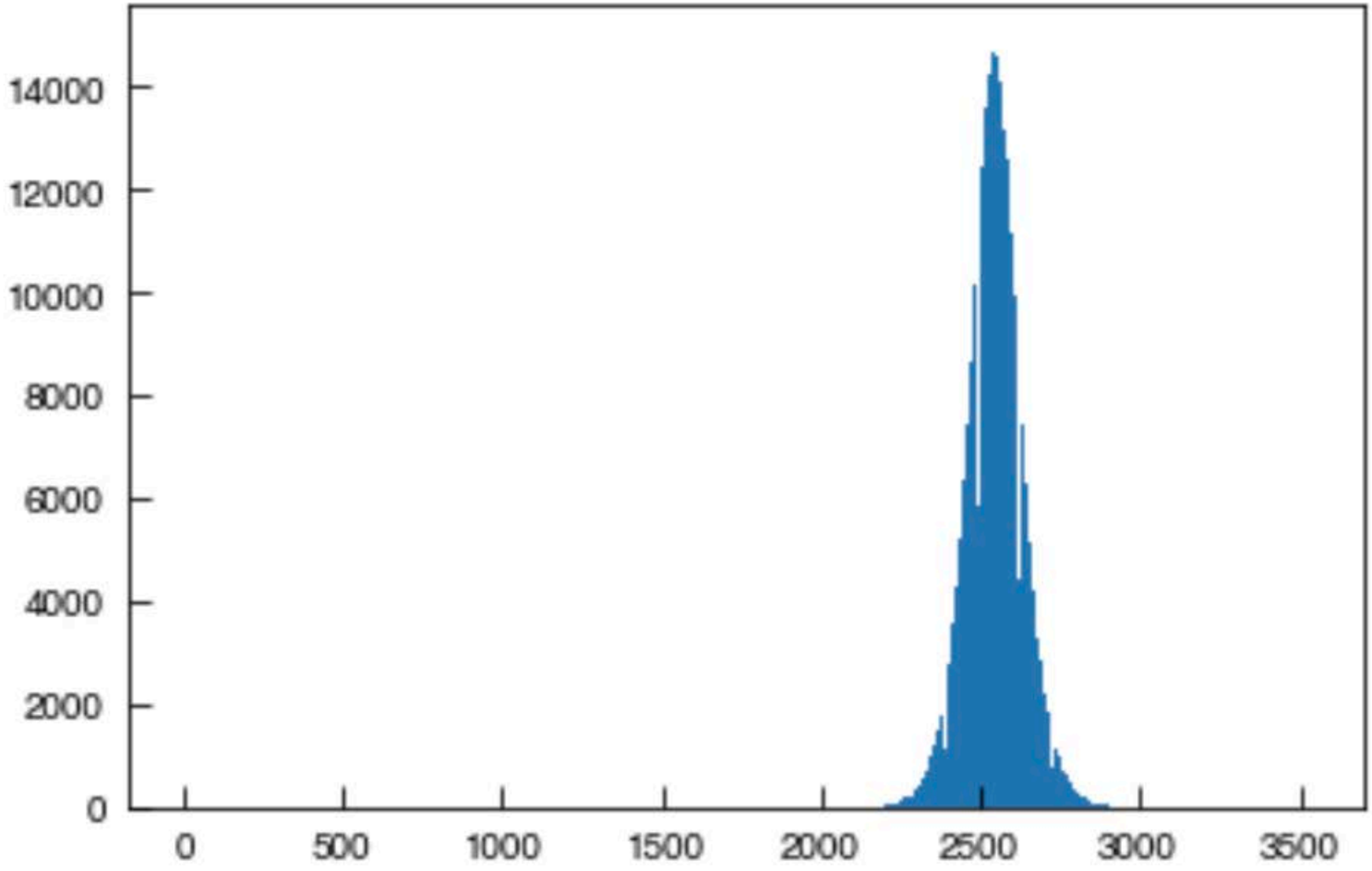
```
]:  # make a list of filenames
    image_list = ['M13_blue_0001.fits','M13_blue_0002.fits','M13_blue_0003.fits',\
                  'M13_blue_0004.fits','M13_blue_0005.fits']
```

```
]:  #make an array of images from the list of images
    image_concat = [fits.getdata(image) for image in image_list]
```

```
]:  #sum the images together
    final_image = np.sum(image_concat, axis=0)
```

```
]:  #plot a histogram of the image pixel values
    image_hist = plt.hist(final_image.flatten(), bins='auto')
```

Astropy Tutorial

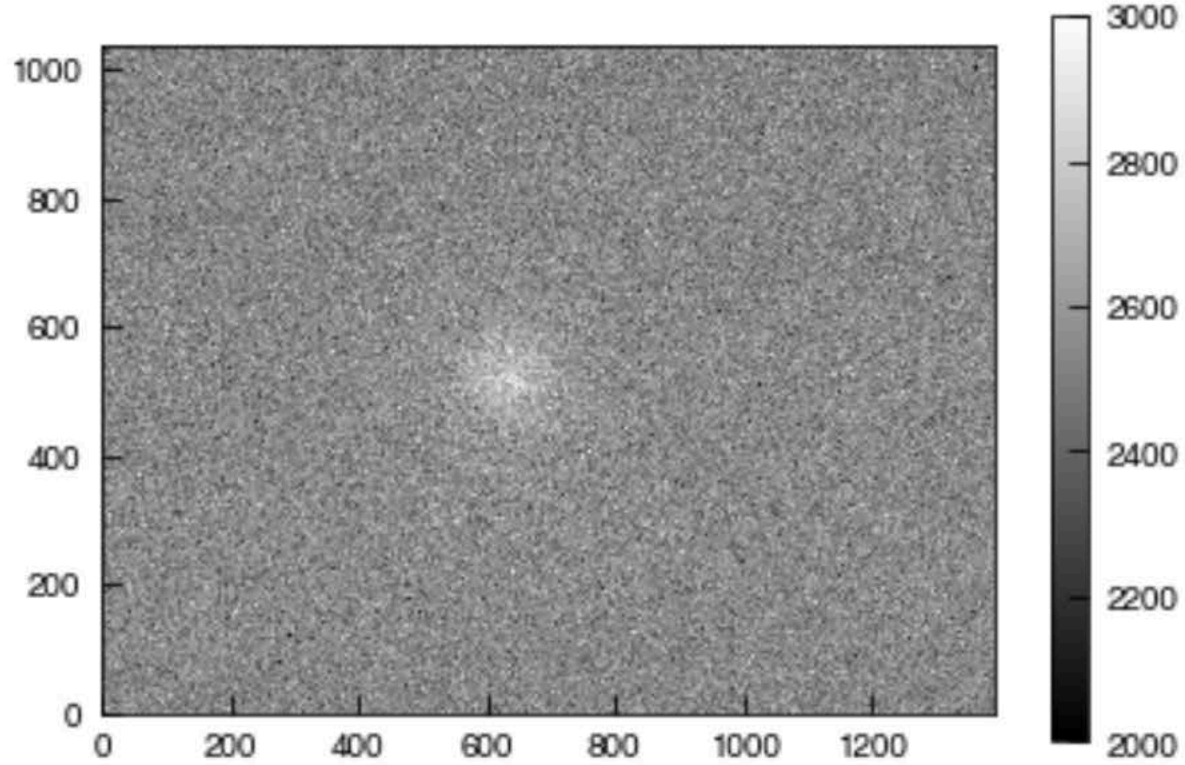


Astropy Tutorial

We'll use the keywords vmin and vmax to set limits on the color scaling for imshow.

In [18]:
```
plt.imshow(final_image, cmap='gray', vmin=2E3, vmax=3E3)
plt.colorbar()
```

Out[18]: <matplotlib.colorbar.Colorbar at 0x118685978>



Astropy Tutorial

Writing a new FITS file

We can easily do this with the writeto() method.

Warning: you'll receive an error if the file you are trying to write already exists. That's why we've set clobber=True.

```
outfile = 'stacked_M13_blue.fits'
hdu = fits.PrimaryHDU(final_image)
hdu.writeto(outfile, overwrite=True)
```

Final Projects

1) You have chosen from one of the following 4 project topics:

    1) Damped, driven pendulum
    2) Logistic map and chaos
    3) Astronomical source detection
    4) Monte Carlo Integration

2) Groups should be organized through GitHub, invite your group members and us (TA + prof) as collaborators on your project. Tag us (TA+prof when your final project is ready to grade).

3) Each python module should indicate which student authored which part of the code.

4) Final projects are due (tags must happen by) **Wednesday, December 12, 2018 at 3pm.**

Save Your Work

Make a GitHub project “astr-119-session-17”, and commit the programs you made today.