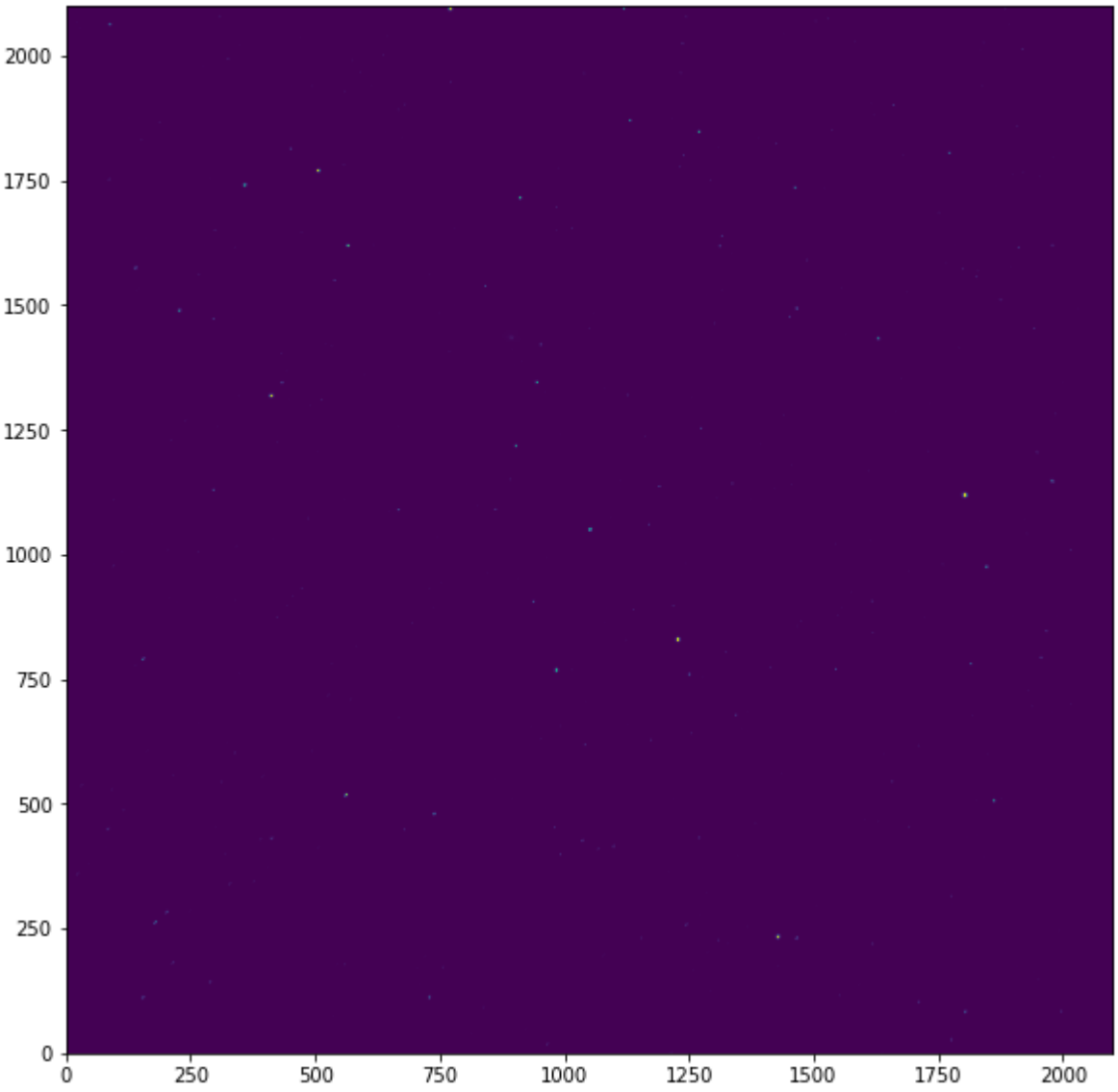
In [39]:
```
import os
import numpy as np
from astropy.io import fits
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
from matplotlib import cm
```

In [40]:
```
imagefile = os.path.join(os.sep, r'C:/Users/student/Downloads/object.fits')
image = fits.open(imagefile)[0].data
```

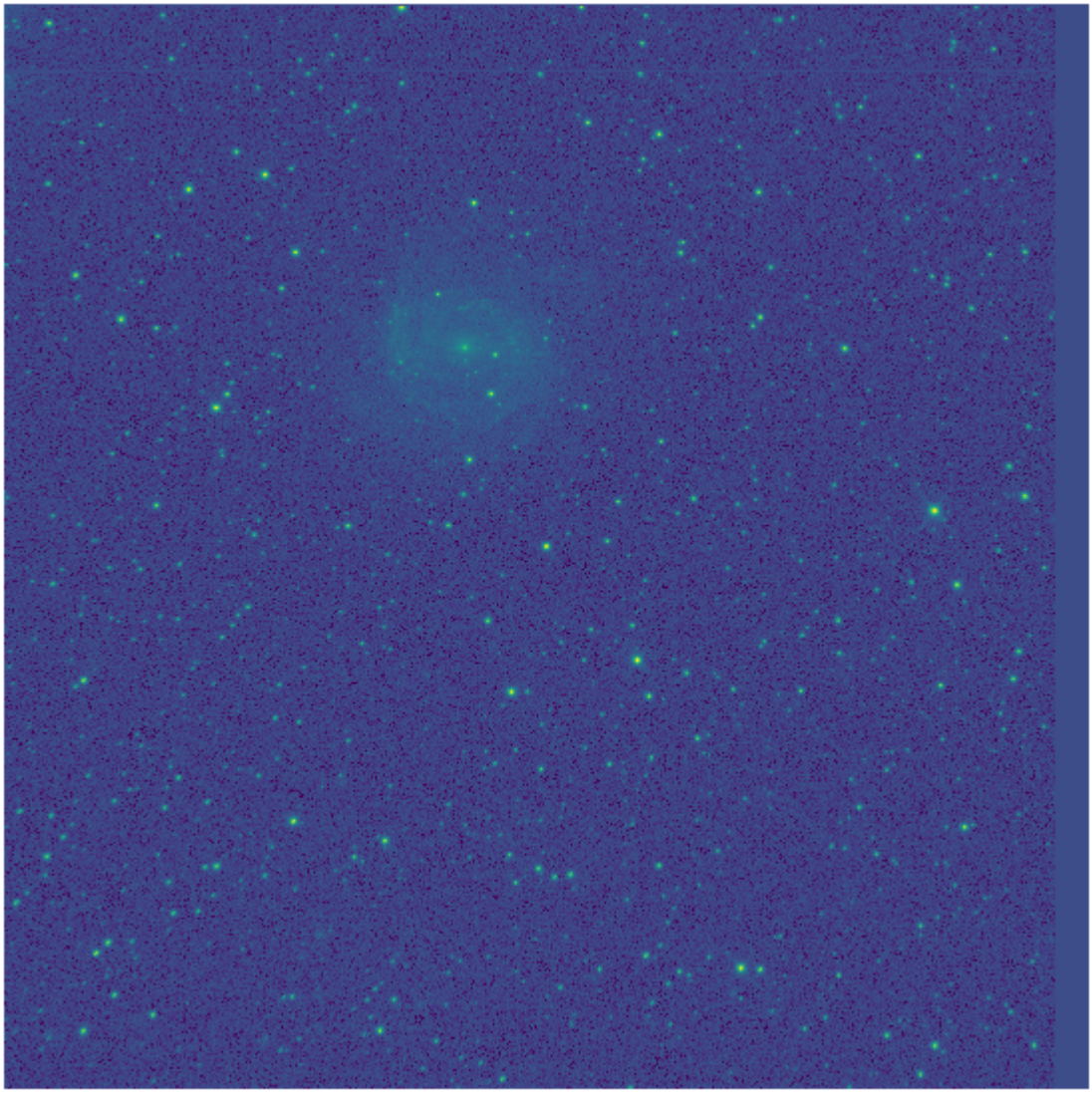
In [41]:
```
plt.figure(figsize = (10,10), facecolor = 'white')
plt.imshow(image, origin = 'lower')
plt.show
```

Out[41]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



In [43]:
```
plt.figure(figsize = (10,10), facecolor = 'white')
cmap = 'viridis'
gamma = (2.3)
norm = mpl.colors.LogNorm(gamma)
plt.imshow(np.sqrt(image), origin = 'lower', cmap = cmap, norm = norm)
plt.axis('off')
plt.show
```

Out[43]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



In [44]:
```
secfile = os.path.join(os.sep, r'C:\Users\student\Downloads\object.cat')
catalog = np.loadtxt(secfile)
```

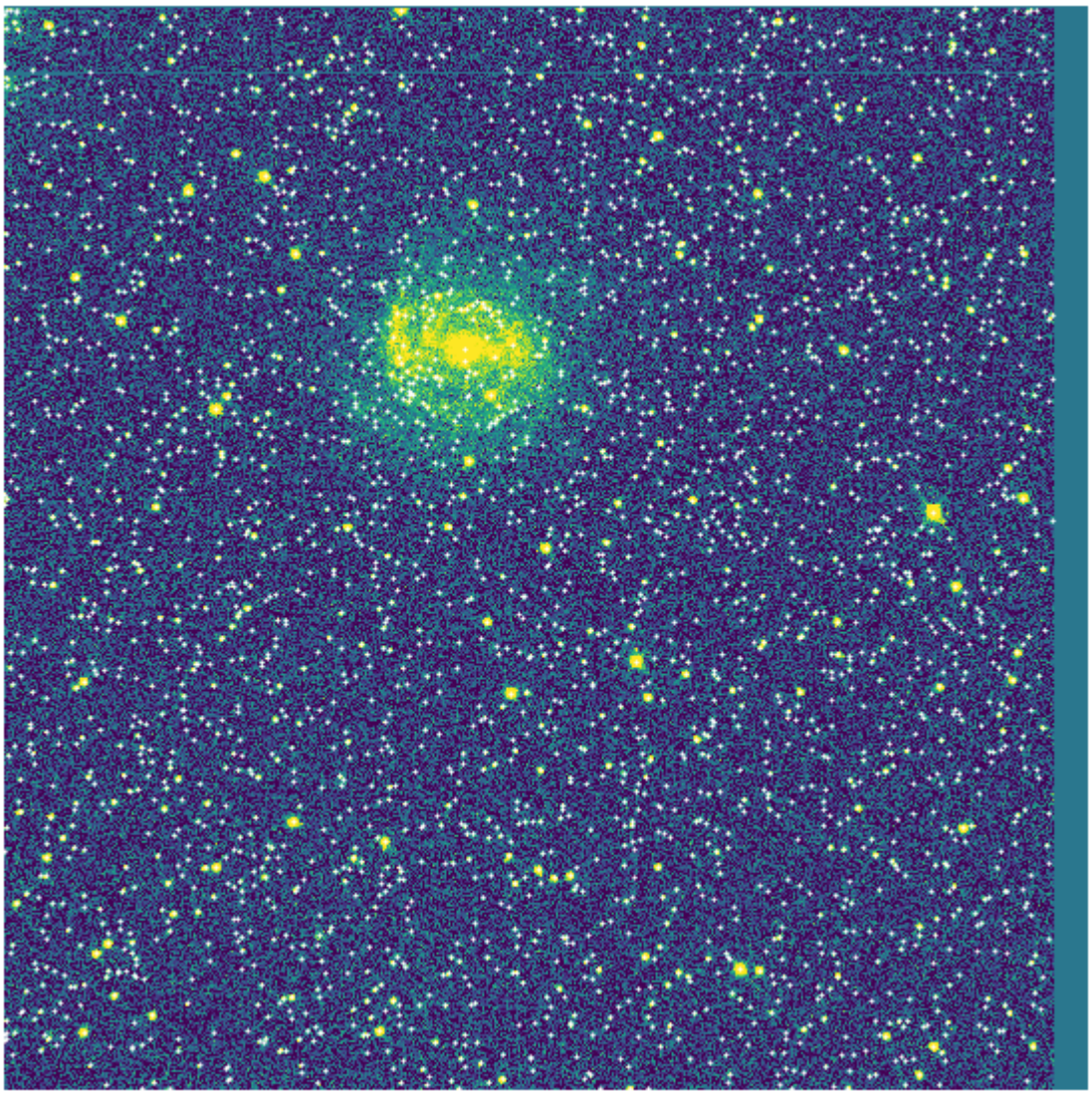
In [45]:
```
x = catalog[:, 5]
y = catalog[:, 6]
flags = catalog[:, 9]

print(x, y, flags)
```

```
[1774.1411 1677.0443  965.4194 ... 1183.3757 1687.2463  635.0281] [  29.3371
26.5926   20.1197 ... 1920.5032 1998.0814 2002.3348] [0. 0. 0. ... 0. 0. 2.]
```

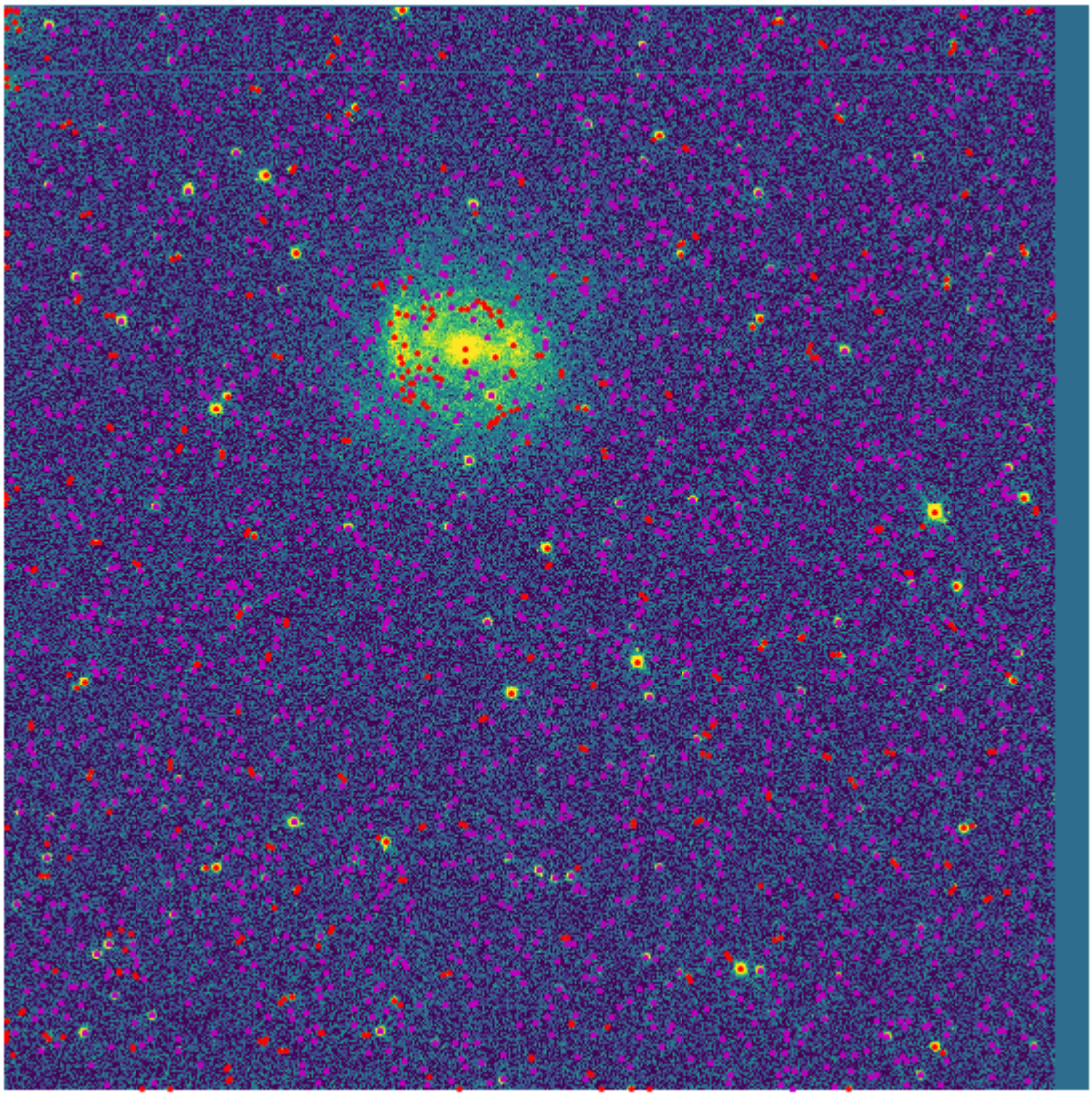
In [65]:
```
plt.figure(figsize = (10,10), facecolor = 'white')
cmap = 'viridis'
gamma = 4
norm = mpl.colors.LogNorm(gamma)
plt.imshow(np.sqrt(image), origin = 'lower', cmap = cmap, norm = norm, vmin = 5,
plt.axis('off')
plt.show
plt.plot(x,y, '*w', markersize = 1.5)
```

Out[65]:
```
[<matplotlib.lines.Line2D at 0x2d60c232dd8>]
```



In [70]:
```
plt.figure(figsize = (10,10), facecolor = 'white')
cmap = 'viridis'
gamma = 4
norm = mpl.colors.LogNorm(gamma)
plt.imshow(np.sqrt(image), origin = 'lower', cmap = cmap, norm = norm, vmin = 5,
plt.axis('off')
plt.show

for i, flagsInd in enumerate(flags):
    if flagsInd == 0:
        plt.plot(x[i],y[i], '*m', markersize = 3)
    else:
        plt.plot(x[i],y[i], '*r', markersize = 2.5)
```



Flagged stars (non-zero values) have data or image errors, i.e. distortion or extracting data error

In [72]:
```
flux = catalog[:, 1]
```

At x = 1563.94, y = 718.9; counts = 210.977 and flux = 2678.15 (Number 908)

At x = 722.6281, y = 1388.26; counts = 202.71 and flux = 1825.15 (Number 1435)

In [79]:
```
fluxStars = np.where((flux >= 1990) & (flux <= 2010))

fluxStars
```

Out[79]:
```
(array([ 263,  583, 1045, 1147, 1380, 1727, 2426, 2698], dtype=int64),)
```

In [97]:
```
coordsX = []
coordsY = []
for u in fluxStars:
    coordsX.extend((x[u]))
    coordsY.extend((y[u]))
coordsX = list(map(int, coordsX))
coordsY = list(map(int, coordsY))
print(coordsX)
print(coordsY)
```

```
[1141, 83, 119, 1838, 682, 98, 18, 176]
[216, 476, 828, 894, 1076, 1828, 1287, 2071]
```

In [86]:
```
image[1141, 216]
```

Out[86]:
```
71.40912
```

In [100]:
```
peakVals = []
for h in range(len(coordsX)):
    peakVals. append(image[coordsX[h], coordsY[h]])

print(peakVals)
```

```
[71.40912, 17.166082, 28.568373, 60.065742, 30.118006, 44.073376, 19.81029, 46.
838295]
```

In [101]:
```
stdDev = np.std(peakVals)
print(stdDev)
```

```
18.060295
```

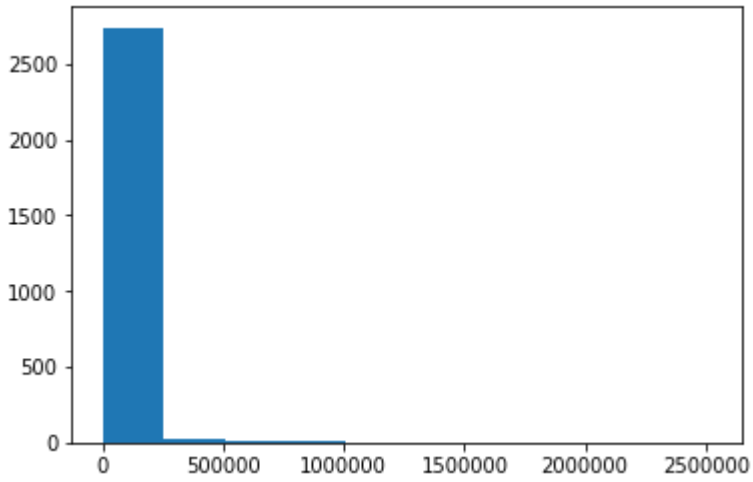
In [110]:
```
meanPeak = sum(peakVals) / len(peakVals)
print(np.sqrt(meanPeak))
```

```
6.305248627869232
```

Comparing the standard deviation of the pixel values to the square root of the mean, it is shown that the standard deviation is about 3 times the value of the square root of the mean.

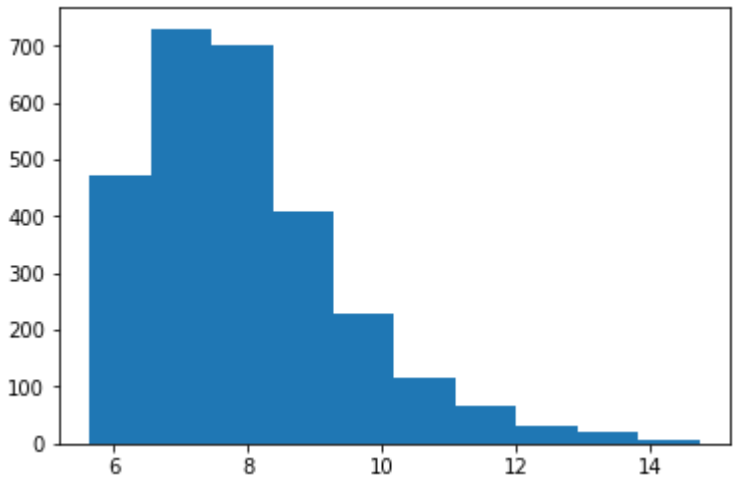
In [102]:
```
plt.hist(flux)
```

Out[102]:
```
(array([2.741e+03, 2.300e+01, 7.000e+00, 5.000e+00, 2.000e+00, 1.000e+00,
        2.000e+00, 0.000e+00, 0.000e+00, 1.000e+00]),
 array([2.81495200e+02, 2.52835446e+05, 5.05389396e+05, 7.57943347e+05,
        1.01049730e+06, 1.26305125e+06, 1.51560520e+06, 1.76815915e+06,
        2.02071310e+06, 2.27326705e+06, 2.52582100e+06]),
 <a list of 10 Patch objects>)
```



In [104]:
```
plt.hist(np.log(flux))
```

Out[104]:
```
(array([473., 730., 702., 409., 229., 117.,  65.,  32.,  19.,   6.]),
 array([ 5.6401154 ,  6.55031153,  7.46050766,  8.37070379,  9.28089992,
        10.19109606, 11.10129219, 12.01148832, 12.92168445, 13.83188058,
        14.74207672]),
 <a list of 10 Patch objects>)
```



The logarithmic histogram (above) looks more informative because it shows more of the data.

In [ ]:
```
```

Physics 134L, Fall 2021
Millar-Blanchaer

Lab 1 **Names:** Tim Marquez

Due date: Monday, October 4th, by 5:00pm, submitted through Gradescope.

This lab will introduce you to `ds9`, a venerable (if not fantastic) image viewer that is a common way astronomers still look at images; and `jupyter`, an interactive way of writing computer code in `python`. The command line used to be the standard way of interacting with a computer, but has been largely superseded (for most of you) by graphical user interfaces (GUIs) and window managers. The command line remains powerful, however, and computer experts and researchers live on the command line rather than in the window manager. While `ds9` will have limited usefulness beyond this class (unless you continue in astronomical research!), `python` and `jupyter` are very widely used across fields.

There will be several sections in this lab where you will write answers or results. These sections will be boxed. **Directions for you to do something will be bolded.** You will turn in a scanned (or photographed) printed lab with your written responses into Gradescope.

ds9 Fundamentals

The program ds9 allows you to display astronomical images, measure many of their properties, and much more. Unfortunately, the manual for ds9 is particularly terse and unhelpful. The only way to understand the program's capabilities is to try things, or to consult a guru (perhaps your own lab partner). Do the following few first steps.

Kashif — M, W } before 5 pm
T, Th, F } after 1 pm

Chen — M – W after 2 pm
F – after 1 pm

Tim — MW F – after 3 pm
T + Th – after 10:45 am
MW – before 12:30 pm
F – before 11 am

Download *object.fits* and object.cat from Gauchospace. You may right-click *object.fits* (if you associated fits files with ds9) and open with ds9, or you my load ds9 by clicking on the icon pinned to the taskbar and then using the open function (you can do this with both the GUI button and from the "File" menu) to find the file *object.fits*.
Once you have opened the image, About how many stars do you see?

~10 stars

Left click on "scale", on "more..." if necessary to make "zscale" visible, and on "zscale". Changing the scale parameters adjusts how the 2D image array in ds9 is display to the user, but doens't actually change the data itself. About how many stars now?

~1000 and one big bright object

Right click inside the image window, and drag the cursor around. What is the effect of moving it left-right? What is the effect of moving the (right-clicked) cursor up-down?

* Moving left-right changes noise/brightness
  → The bar at the bottom adjusts the black/white position
* Moving up-down adjusts contrast
  → The bar at the bottom changes how black/white blend

Experiment with different scale options (there are many!) to get a feel for how this functionality works. Try different colormaps (in the "color" menu). Describe the settings that you find best display the image and why.

squared with zscale works the best because it has the least noise

Astronomical images ordinarily come with "metadata" = (data about the data). In images in FITS format, the metadata appear in a "header" that can be displayed separately from the data. **In the ds9 window, click on "File/Display Fits Header..." Answer the following questions, based on the header.**

What are the x- and y- dimensions of the image (in pixels)?

$x = y = 2100$ px

What was the image exposure time (in s)?

60 s

What filter was used? (We will go into what this means later)

Filter = gp

What was the name of the object being observed?

SN 2013AA

What was the right ascension object being observed?

CRVAL1 = $2.1814 \cdot 10^2$

CRPIX1 = $1.0505 \cdot 10^3$

Google the object name you found above, and find an image of it on the web. Match this image to your ds9 display of `object.fits`. Adjust the scaling so you get a recognizable picture of the object (which is a supernova) and its host galaxy. (Try clicking "scale", then "min/max" and then various scalings. Then play with right-click-and-drag-the-cursor to get a good-looking image. This is as much art as science!)

What are the x,y coordinates of the supernova?

~~Position~~ = 1050.6 = x 1052.7 = y

What is the signal in the central pixel of the supernova? (units of counts)

~~Value~~ 49698.6

In the central pixel of the host galaxy?

~~888.9 = 1432.2 = y~~ ~~Value~~ 1476.29

Save an image of your screen (under the "File" menu of ds9). You may also use the Windows Snipping Tool. Save it as a jpeg and submit it with your completed the assignment.

using a program called `SExtractor` that you will use later in the course. Yes, the name is crass, and no, you should not choose a similar name when you write your own useful pieces of software.

The catalog of extracted stars is in a file called `object.cat`. We will visualize this catalog a few ways, getting a little practice with python arrays along the way. The first step is to load in the catalog. **You will define the catalog filename in exactly the same way you did with the image filename, using `os.path`.** You will then read it into python with the syntax

```
catalog = np.loadtxt(catalogfile)
```

Open the catalog in the text editor of your choice (notepad, wordpad, word, emacs, etc.) and take a look: the first few lines show what each column represents. For now, focus on three columns: x-position, y-position, and flags. **Set a new variable `x` equal to the array of x-positions from the catalog, a new variable `y` equal to the array of y-positions, and `flags` equal to the flag entry.** When you do this, remember that the first array element is zero. Play around to see what to do, use a search engine, and don't immediately ask for the answer. For example, what would the following lines give?

```
x = catalog[0, 0]
x = catalog[:, 0]
x = catalog[0]
x = catalog[0, :]
```

If you have never programmed before, this sort of indexing varies between programming languages but the basic idea is very standard. **Now plot the extracted stars atop your pretty image using `plt.plot(x, y)` (use a new cell). Use dots/markers rather than lines, and consult the manual for guidance if you need it:**

`https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html`

Next, change the plot a bit: plot those stars that have `flag == 0` in one color/marker style, and stars that have `flag != 0` in a different color/marker style. Hint: the easiest way to do this will involve either `numpy.where` or boolean indexing. You might want to go back to the indexing tutorial. Also, there is a very important difference between = and ==. The statement

```
x = 5
```

sets the variable `x` equal to the value 5. The statement

```
x == 5
```

will not change the value of `x`. It will evaluate to `True` if, indeed, `x` is the integer 5, and `False` otherwise. If `x` is an array, this statement will return an array of `True` and `False` values (a boolean array). The statement will return an error if `x` is not defined.

Once you have extracted arrays like `x_flagged` and `y_flagged` and have plotted them atop your image, comment on any differences that you notice between the flagged (`flag != 0`) and unflagged (`flag == 0`) stars. As usual, this comment should appear in a markdown cell below the figure. If you need insight or are curious, go search for what the flags represent in the `SExtractor` manual and use this to help inform your comments.

Finally, it is time for a few basic analyses on the stars. For this purpose, we will use one more column from the catalog: the flux for each star (second column). Set the variable `flux` equal to this array. This flux is the total number of counts within a fuzzy region around each star. **Choose a few representative stars, and compare the value of `flux` for that star with the peak value of the image in that area.** You may compute the peak value of the image by hand in ds9, or in python. Feel free to experiment and see if you can figure it out in python. If you do, please be aware that the first index is actually y, not x. **Determine the approximate relationship between the peak value of the image and the flux reported by `SExtractor`.**

The exact factor relating the peak intensity and total flux will vary from star. Part of the reason why is that the flux in any one pixel is subject to photon noise (also called shot noise or Poisson noise). If we have

x: 1563.94 y: 718.9
Value: 210.977 flux = 2626.15

a lot of photons, the scatter in the count rate scales roughly as the square root of the number of counts. This result is due to the central limit theorem and is a major result of statistics. **Choose the eight stars with fluxes between 1990 and 2010 as reported by `SExtractor`.** If you haven't already played with boolean indexing and/or `np.where`, now is the time. You probably don't want to do this by hand. Additional note: suppose I have two boolean values, `x` and `y`. The value `False` corresponds to zero and the value `True` corresponds to anything nonzero (one by default).

Consider the operations `*`, `/`, `+`, and `-`. Make a truth table for each of the following cases to prove your answer.
How could you write `x and y` (i.e. the logical **and** for the two variables `x` and `y`)?

x and y : * + / -
yes yes yes yes

How could you write `x or y` (the logical **or**)?

or → * + / -
y y y n

Using your newfound ability to select elements from arrays, find the x and y coordinates of these eight stars. Then compute the standard deviation of the eight peak pixel values. You may compute the peak pixel values either by hand in ds9 or directly in python. **Either way, be explicit about what you are doing in your jupyter notebook.** When you compute the standard deviation, you may also do that as you like, but again be clear. **What would you expect for the standard deviation according to the square root rule? Is this what you find? If your standard deviation was larger or smaller than this expectation, try to come up with possible explanations, and write these in a markdown cell in your jupyter notebook.**

We have one, last, easy task for this lab. **You will make histograms of the fluxes returned by `SExtractor`. Use the function `plt.hist()` for this purpose. Make two separate histograms: one of the fluxes, and one of the log of the fluxes (either base ten or natural log, your choice). Comment on which of these looks more informative and why.** As before, use your judgment and the manual pages to make a figure with labeled axes that looks nice. We will work in logarithmic units of flux called magnitudes for the rest of the course, and indeed, this is another column of the catalog file returned by `SExtractor`.

In this box put the name of your partner, and describe how you worked together to complete this lab.

Kashif

Upload this worksheet and the pdfs of your jupyter notebook(s) to the Gradescope assignment link on the "Lab 1" tab of the Gauchospace page. Each group member should upload their own version of this page, but it is ok if your code submissions are the same if you worked in partners the whole time.