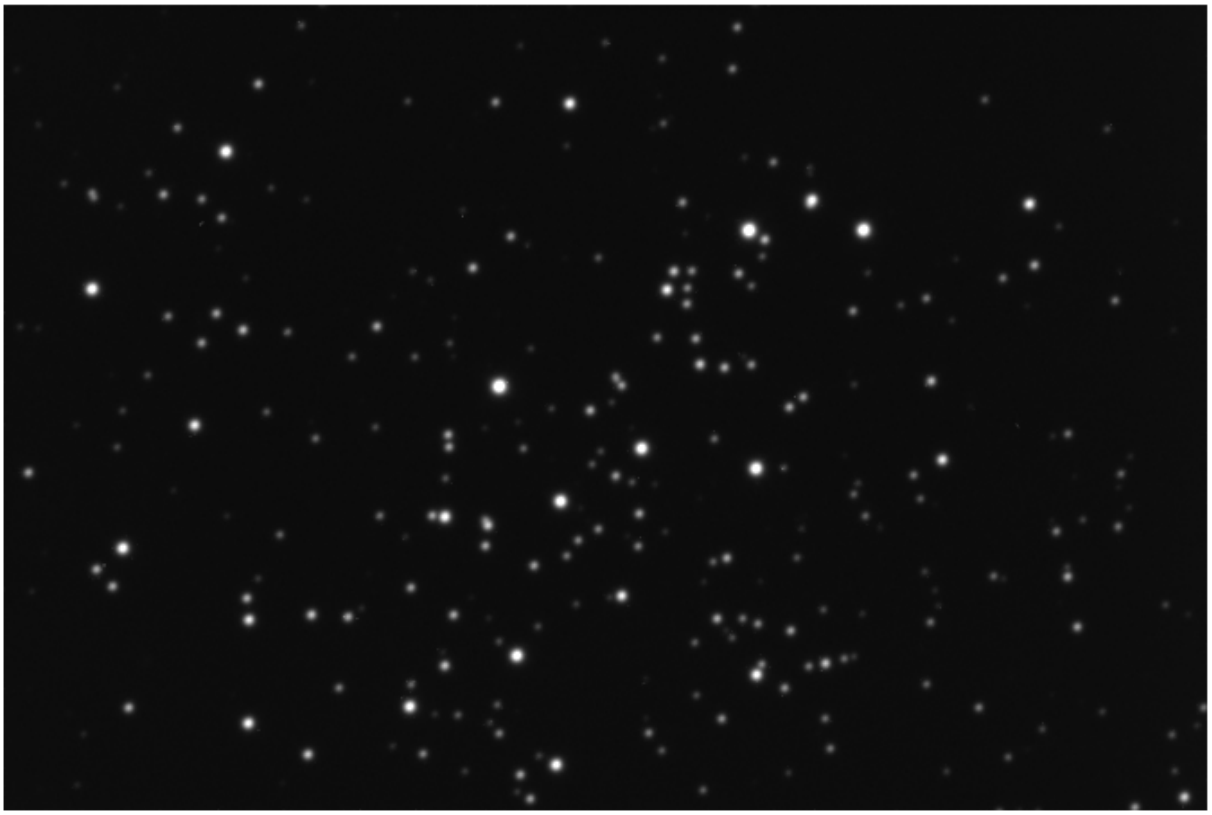
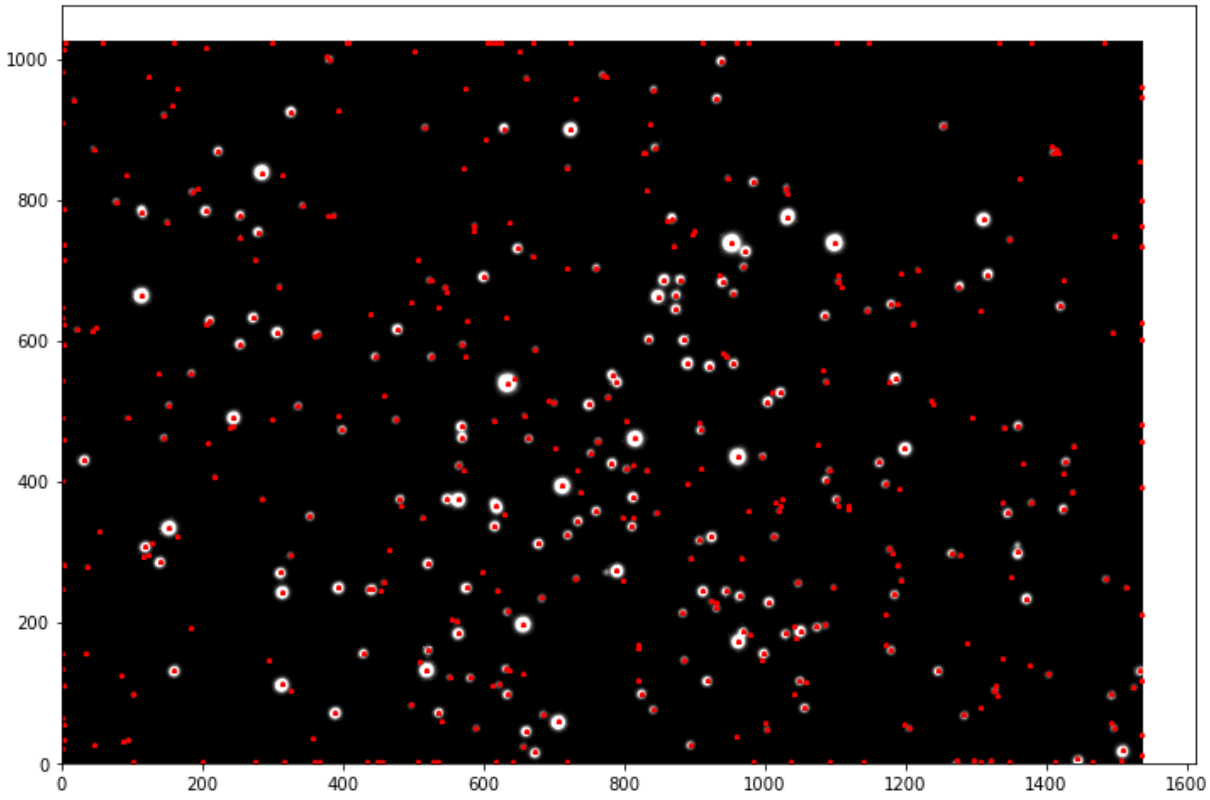
In [1]:
```
import os
import numpy as np
from astropy.io import fits
import matplotlib.pyplot as plt
import matplotlib as mpl
from matplotlib import cm
#importing necessary libraries
```

In [2]:
```
imagefile = os.path.join(os.sep, r'E:\Documents\134L\cluster1.fits')
image = fits.open(imagefile)[0].data
#importing the fits file
```

In [3]:
```
plt.figure(figsize = (12,10), facecolor = 'white')
cmap = 'gray'
plt.imshow(np.log(image), origin = 'lower', cmap = cmap, vmin = 6.2, vmax = 9)
plt.axis('off')
plt.show()
#displaying the cluster1 image
```

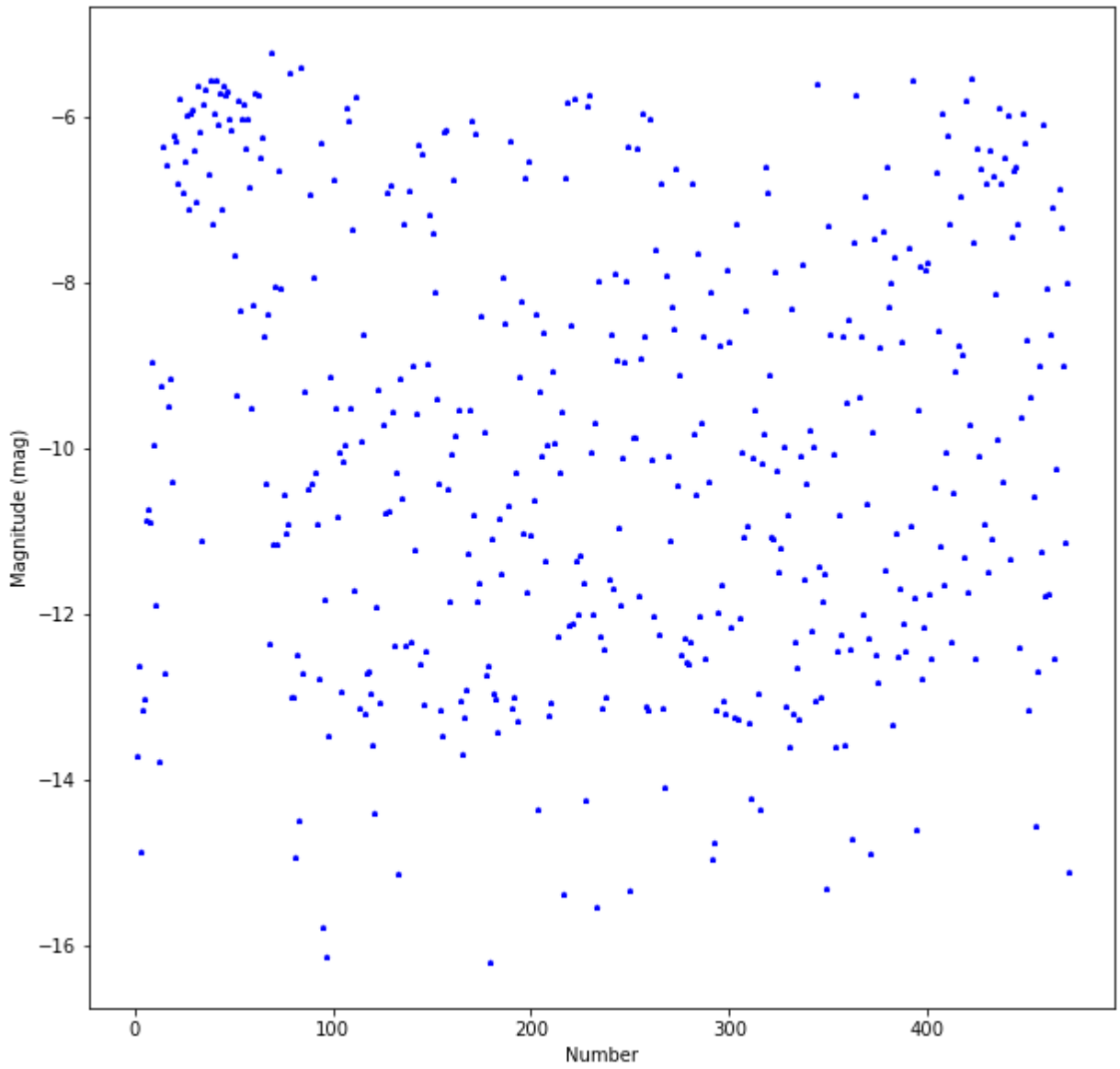


In [4]:
```
secfile = os.path.join(os.sep, r'E:\Documents\134L\cluster1.cat')
catalog = np.loadtxt(secfile)
x_image = catalog[:,1]
y_image = catalog[:,2]

plt.figure(figsize = (12,10), facecolor = 'white')
cmap = 'gray'
plt.imshow(np.log(image), origin = 'lower', cmap = cmap, vmin = 6.38, vmax = 7)
plt.plot(x_image, y_image, '*r', markersize = 3)
plt.show()
#plotting the positions of stars onto the image
```

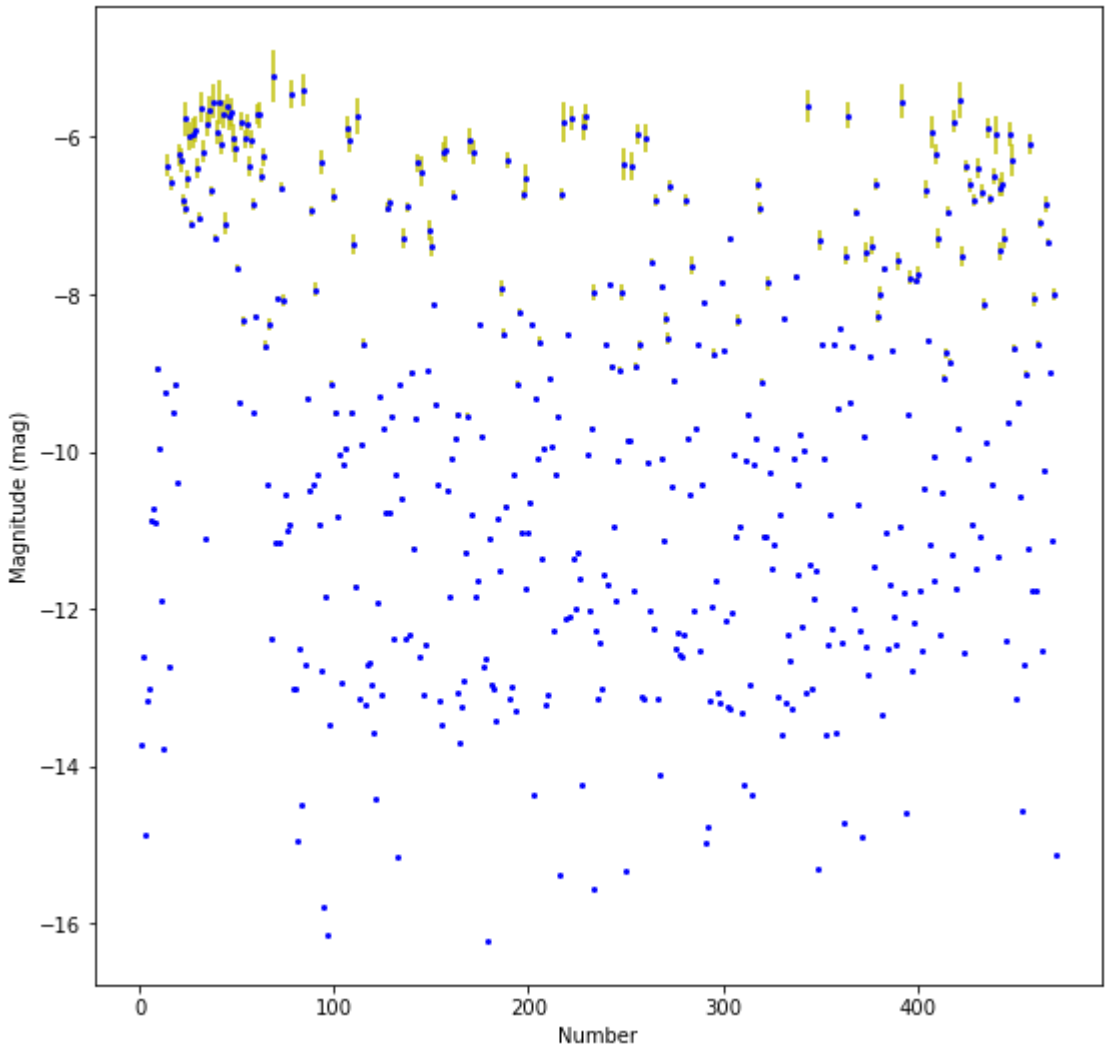


In [7]:
```
MAG_ISOCOR = catalog[:, 5]
MAGERR_ISOCOR = catalog[:, 6]
NUMBER = catalog[:, 0]

plt.figure(figsize = (10,10))
plt.xlabel('Number')
plt.ylabel('Magnitude (mag)')
plt.plot(NUMBER, MAG_ISOCOR, '*b', markersize = 3)
```

Out[7]: [<matplotlib.lines.Line2D at 0x18761471160>]

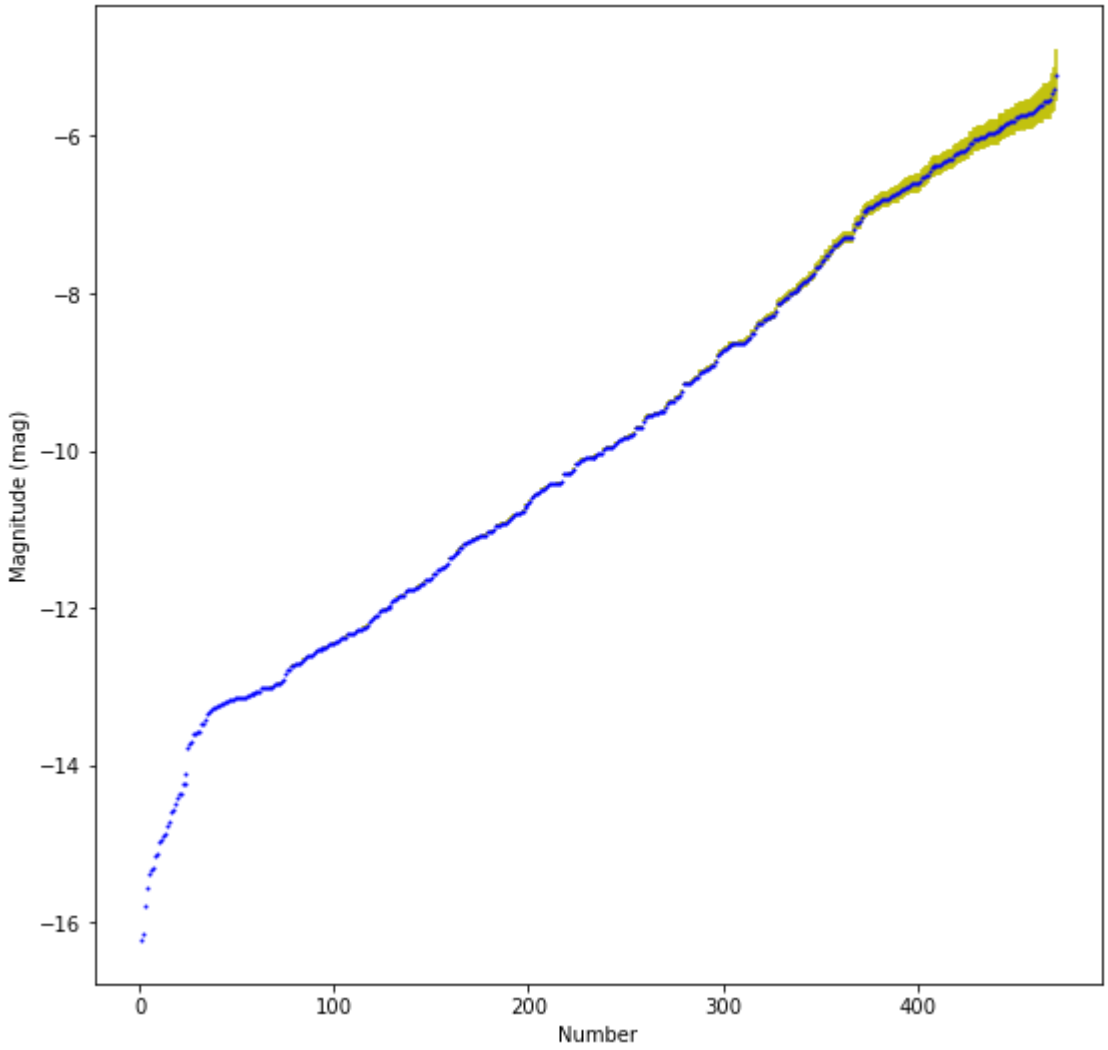


In [10]:
```
plt.figure(figsize =(9,9))
plt.xlabel('Number')
plt.ylabel('Magnitude (mag)')
plt.errorbar(NUMBER, MAG_ISOCOR, MAGERR_ISOCOR, ecolor = 'y', fmt = 'bo', markers
plt.show()
#plotting mag_isocor and magerr_isocor according to their number, not sorted
```

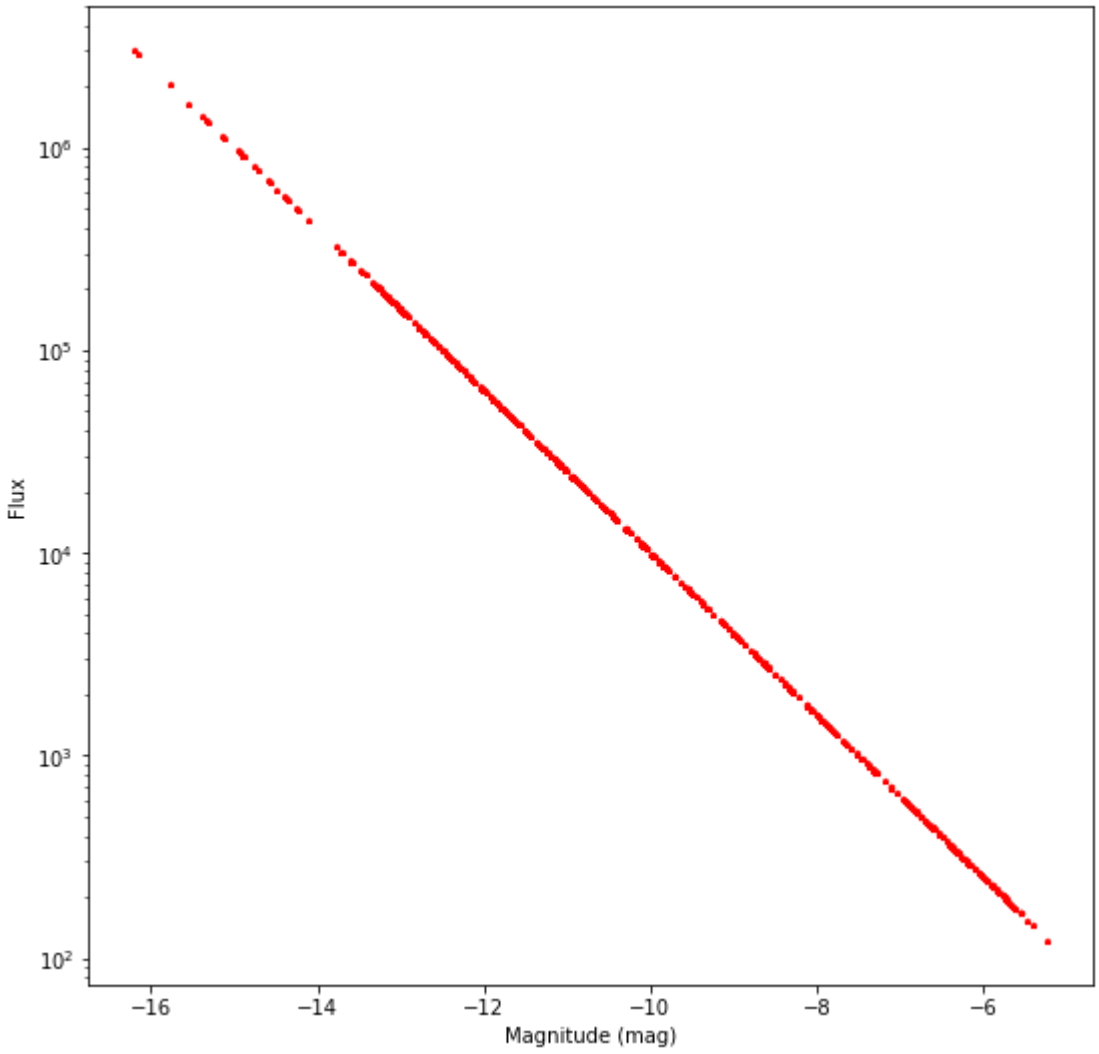


In [13]:
```
mags_sorted = np.sort(MAG_ISOCOR)
mag_errs_sorted = np.sort(MAGERR_ISOCOR)

magArgSort = np.argsort(MAG_ISOCOR)
magerrArgSort = np.argsort(MAGERR_ISOCOR)
#the goal is to organize the magnitudes in order and plot them
```

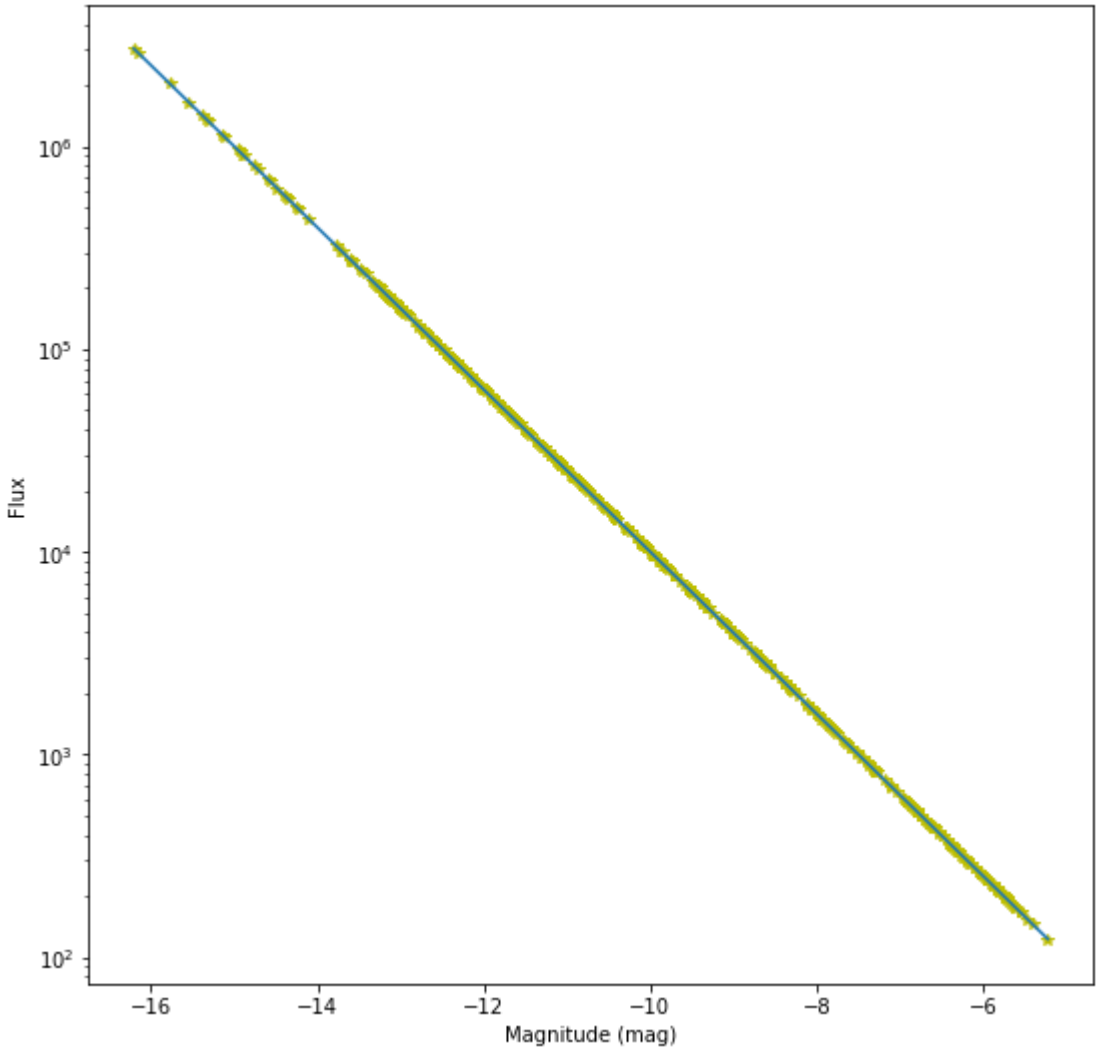
In [14]:
```
plt.figure(figsize = (9,9))
plt.xlabel('Number')
plt.ylabel('Magnitude (mag)')
plt.errorbar(NUMBER, mags_sorted, yerr = mag_errs_sorted, ecolor='y', fmt = 'bo',
plt.show()
#plotting the sorted mags and mag errs
```



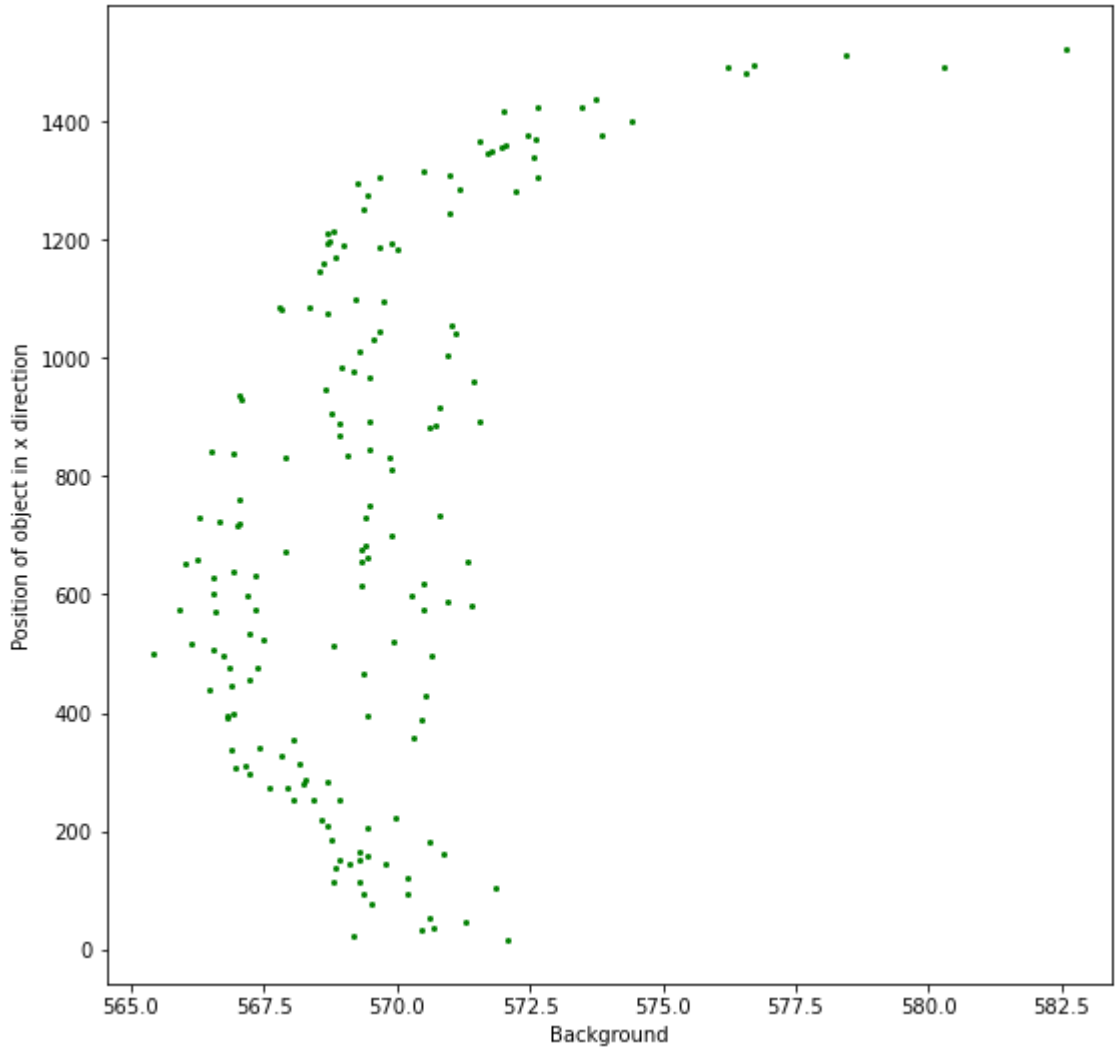
In [15]:
```
FLUX_ISOCOR = catalog[:,3]
FLUXERR_ISOCOR = catalog[:,4]

plt.figure(figsize = (9,9))
plt.ylabel('Flux')
plt.xlabel('Magnitude (mag)')
plt.plot(MAG_ISOCOR, FLUX_ISOCOR, '*r', markersize = 3)
plt.yscale('log')
#plotting the logarithmic relationship of Flux and Magnitude, using a log normali
```



In [127]:
```
#adding a line of best fit:
x = np.linspace(np.amin(MAG_ISOCOR), np.amax(MAG_ISOCOR), 10)
a = -0.4
b = 0
y = 10**(a*x + b)

plt.figure(figsize = (9,9))
plt.ylabel('Flux')
plt.xlabel('Magnitude (mag)')
plt.plot(MAG_ISOCOR, FLUX_ISOCOR, '*y', lw = 4, markersize = 6)
plt.plot(x, y) #Line of best fit
plt.yscale('log')
```

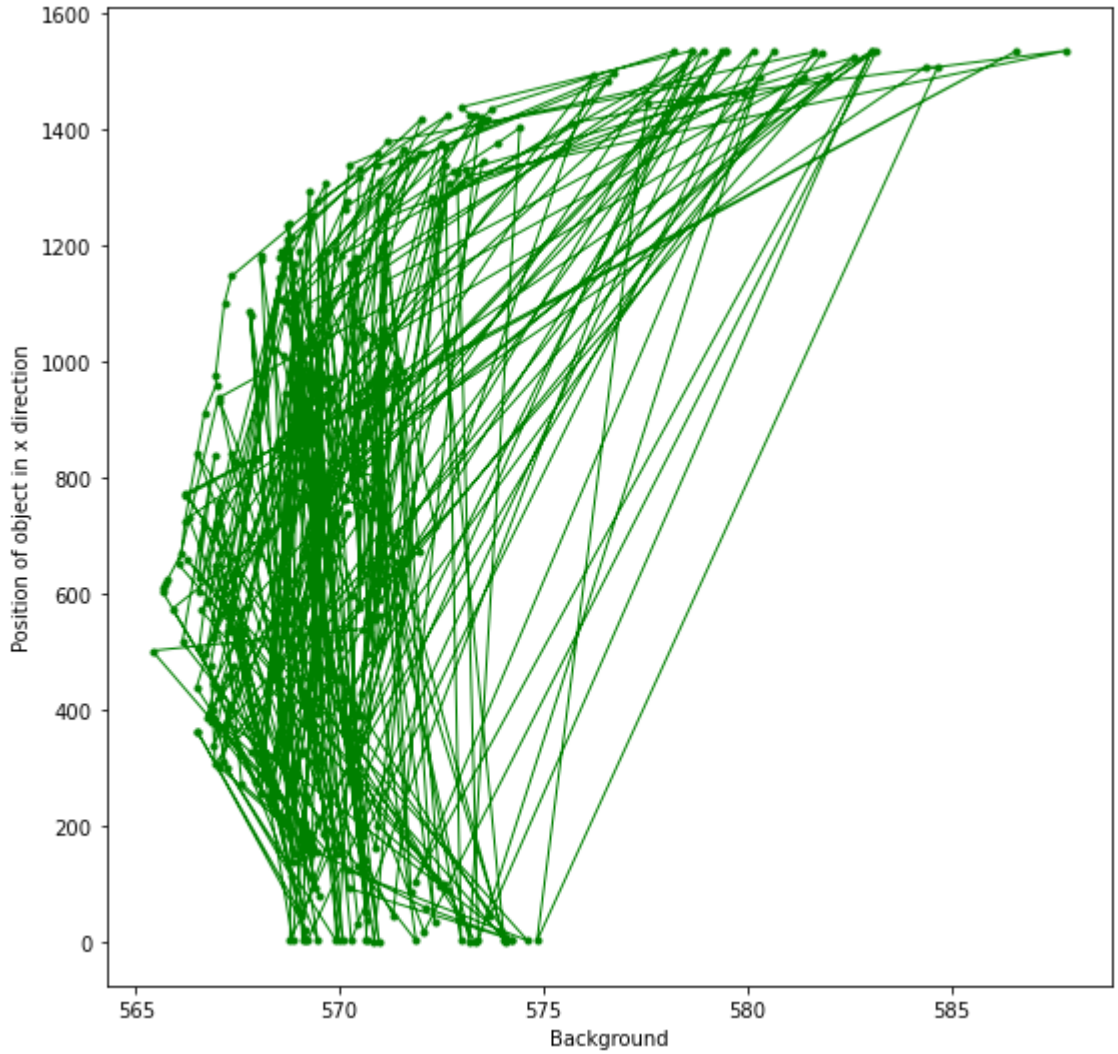


In [111]:
```
FWHM = catalog[:,12]
flags = catalog[:,14]
BG = catalog[:, 13]

plt.figure(figsize = (9,9))
plt.xlabel('Background')
plt.ylabel('Position of object in x direction')

#filter out bad flags
for i, flagsInd in enumerate(flags):
    if flagsInd == 0:
        realFWHM = (FWHM[i])
        realBG = (BG[i])
        realx_image = x_image[i]
        plt.scatter(realBG, realx_image, s=4, c = 'g')
#plotting the FWHM data without bad flags
```



In [86]:
```
#plotting scatter plot with lines (unusable)
plt.figure(figsize = (9,9))
plt.xlabel('Background')
plt.ylabel('Position of object in x direction')
plt.plot(BG, x_image,  'o-', lw = 1, markersize = 3, c = 'g')
```

Out[86]: [<matplotlib.lines.Line2D at 0x187618b4ee0>]



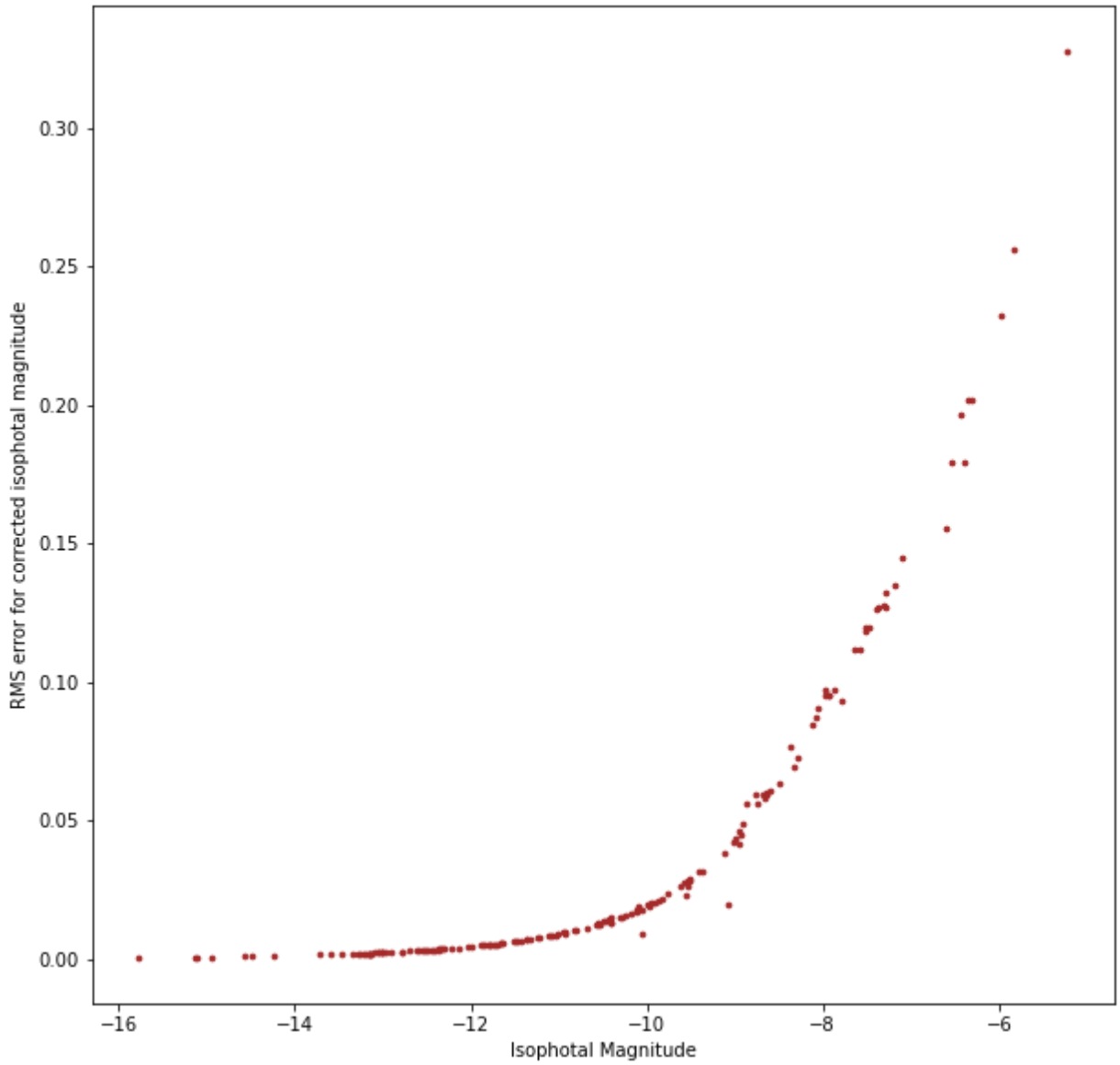
In [54]:
```
print(np.mean(BG))
#average background pixel value over all data points
```

```
570.3387658174098
```

In [23]:
```
print(np.std(BG))
#standard deviation of the background pixel value
```

```
3.2868164128536215
```

In [98]:
```
plt.figure(figsize = (10,10))
plt.xlabel('Isophotal Magnitude')
plt.ylabel('RMS error for corrected isophotal magnitude')

for i, flagsInd in enumerate(flags):
    if flagsInd == 0:
        realMAG = MAG_ISOCOR[i]
        realMAGERR = MAGERR_ISOCOR[i]
        plt.scatter(realMAG, realMAGERR, s=7, c = 'brown')
#plotting the relationship between MAG_ISOCOR and MAGERR_ISOCOR
```



In [118]:
```
#computing N_phot
N_phot = 2.36 * (np.mean(FLUX_ISOCOR) + 571 * (3.14159)*(9/2)**2)
#where gain = 2.36, pi = 3.14159, and FWHM estimate is about 9 px
```

In [119]:
```
print(N_phot)
```

```
310765.1343380808
```

In [120]:
```
571 * (3.14159)*(9/2)**2
```

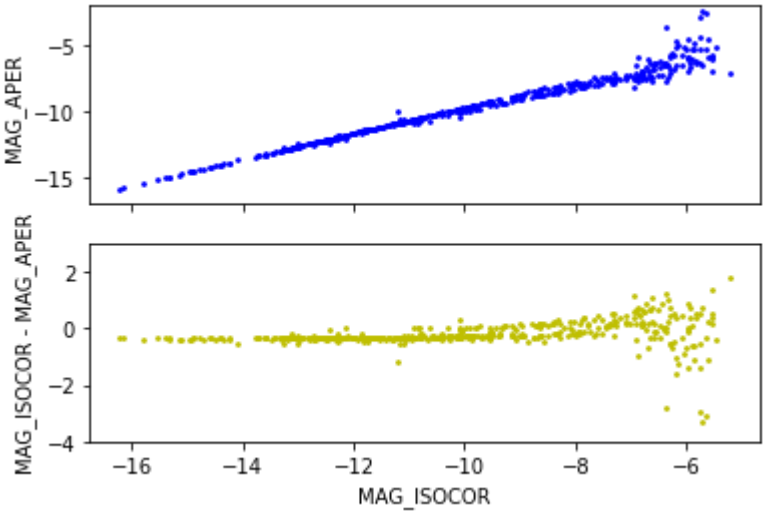
Out[120]:
```
36325.4197725
```

In [128]:
```
mag = np.arange(int(np.amin(MAG_ISOCOR)), int(np.amax(MAG_ISOCOR))+1)
```

In [175]:
```
#The relationship between MAG_ISOCOR, MAG_APER, and (MAG_APER - MAG_ISOCOR)

MAG_APER = catalog[:, 9]

plt.figure(figsize = (17,17))
fig, (ax0, ax1) = plt.subplots(2, 1, sharex = True)
plt.xlabel('MAG_ISOCOR')
ax0.scatter(MAG_ISOCOR, MAG_APER, c = 'b', s = 3)
ax1.scatter(MAG_ISOCOR, MAG_ISOCOR-MAG_APER, c = 'y', s = 3)
ax0.set_ylabel('MAG_APER')
ax0.set_ylim([-17,-2])
ax1.set_ylabel('MAG_ISOCOR - MAG_APER')
ax1.set_ylim([-4,3])
```

Out[175]:
```
(-4.0, 3.0)
```

```
<Figure size 1224x1224 with 0 Axes>
```



In [169]:
```
MAGERR_APER = catalog[:, 10]
print(MAGERR_ISOCOR)
print(MAGERR_APER)
#finding error values of ISOCOR and APER and comparing them to the subplots
```

```
 2.3600e-02 9.9000e+01 8.7000e-03 1.2027e+00 4.4000e-03 5.0000e-03
 4.3200e-02 1.4300e-02 2.6000e-03 2.8000e-03 5.0000e-04 8.3000e-03
 2.3000e-03 2.2000e-03 1.8000e-03 6.8000e-03 5.4000e-03 7.9800e-02
 5.4100e-02 9.4000e-03 4.0400e-01 2.1000e-03 2.3000e-03 1.0100e-02
 2.0000e-03 3.1900e-02 5.3400e-02 7.4000e-03 9.8300e-02 4.7000e-03
 1.4190e-01 7.3000e-03 7.6000e-03 4.0800e-02 1.1000e-03 2.2000e-02
 1.5100e-02 4.9500e-02 6.0000e-03 1.5900e-02 2.0000e-03 2.2000e-03
 2.8100e-02 1.6700e-02 3.4000e-03 1.3600e-02 2.5200e-02 7.0000e-04
 1.7060e-01 3.5960e-01 3.7000e-03 3.9700e-02 3.7000e-03 1.3907e+00
 6.0000e-03 4.0000e-03 6.2000e-03 5.0000e-03 1.2000e-03 6.6520e-01
 6.0012e+00 1.5600e-02 4.0000e-03 1.9900e-02 7.0000e-04 8.7100e-02
 3.1000e-03 2.1000e-03 3.1000e-03 2.3000e-03 5.2000e-03 3.4200e-02
 4.9000e-03 5.8800e-02 3.6900e-02 7.6000e-03 4.2000e-03 1.5000e-02
 3.5200e-02 7.7500e-02 1.2480e-01 7.0000e-04 1.9500e-02 1.7400e-02
 3.2800e-01 4.5000e-03 3.8900e-02 1.1883e+00 4.8200e-02 2.1000e-03
 2.1000e-03 6.4610e-01 1.4200e-02 4.0000e-03 6.6400e-02 3.5000e-03
 1.7770e-01 2.1000e-03 1.4000e-03 6.2200e-02 1.5200e-02 6.9000e-03
 6.8400e-02 6.3800e-02 8.6600e-02 1.1200e-02 2.5600e-02 3.0000e-03
 3.4000e-03 2.9000e-03 2.8000e-03 3.3000e-03 1.0970e-01 1.8100e-02
 1.0400e-02 1.0990e-01 3.9000e-03 2.1400e-02 3.3200e-02 2.9000e-03
```

In [185]:
```
#finding the 3 extreme outliers in the upper plot
for u,Ind in enumerate(MAG_APER):
    if Ind >= -3:
        print(MAG_APER[u])
```

```
-2.5644
99.0
99.0
99.0
99.0
99.0
99.0
99.0
99.0
-2.4571
99.0
99.0
99.0
-2.8083
99.0
99.0
```

In [ ]:
```

```

```
x = np.linspace(np.amin(mags), np.amax(mags), 10)
```

and a variable y by

```
y = a*x + b
```

where you determine the best-fit coefficients a and b by eye. Depending on how you have implemented your logarithmic vertical scale, you may need to plot either

```
y = np.exp(a*x + b)
```

or

Flux = $10^{(-0.4 * \text{mag})}$

```
y = 10**(a*x + b)
```

for it to appear as a line on your plot. **Please overplot your best-fit line on the scatter plot of magnitudes against logarithmic flux.**

Write down the definition of astronomical magnitudes in the space below.

Astronomical magnitudes is the brightness measurement of objects in space

Apparent mag: $m_p = -2.5 \log(F_p) + C_p$

How closely does your fitted constant a match expectations?

Approximately what magnitude would yield a flux of 1 given your fitted a and b? Please show your work.

Flux = $10^{(ax+b)}$ → $a = -0.4$, $b = 0$

Magnitude = 0

$\therefore$ flux = $10^{(-0.4x)}$

$1 = 10^{-0.4(\text{magnitude})}$

The Sizes of (Images of) Stars

The catalog file has a column entitled `FWHM_IMAGE` (meaning Full Width at Half Maximum of the star image). This parameter is a measure of the width of the star images in units of pixels – smaller numbers mean sharper images. You can also estimate these values yourself by looking at the image in ds9.

What happens if you do use lines to connect the data points? Explain what the program is doing to generate this unusable plot.

If there are lines to connect the data points, it looks like the lines are connecting to seemingly random points. I think the program is trying to draw a function out of it, which is why there are so many lines in the denser area and why each point is intersected once.

What is a typical value for the sky background? What is the approximate scatter in the reported values?

The typical value for the sky background is around 571 px. The scatter is around the 570 = x and 573 = x range.

Within Python you can calculate simple statistics (min, max, mean, standard deviation) for data sets using numpy functions. Ordinarily, your visual impression of the typical value of a bunch of plotted data points will closely approximate the mean value, and your visual estimate of the peak-to-peak scatter (ignoring rare outliers) will be roughly 4 times the standard deviation. How do mean and standard deviation computed with Python compare with the visual estimates you just made?

The mean calculated is 570.341 px, so it is very close to the estimated typical value.

Now, make a scatter plot MAGERR_ISOCOR as a function of MAG_ISOCOR. MAGERR_ISOCOR is an estimate of the error in MAG_ISOCOR, based on square-root-rule counting error for the number of detected photons in the object plus the underlying sky background (derived from the Poisson distribution). The total number of sky+object photons is

$N_{\text{phot}} = \text{GAIN} \cdot (\text{FLUX\_ISOCOR} + \text{BACKGROUND} \cdot N_{\text{pix}})$

Where GAIN is the number of detected photo-electrons per signal count (2.36 e^-/count for this image, determined from the EGAIN keyword in the FITS header). N_{pix} is the number of pixels occupied by the object image. With ISOCOR photometry this number is hard to know for sure, but a reasonable guess is $\pi(\text{FWHM}/2)^2$. **Use this guess, the GAIN value just given, and your estimate of BACKGROUND to write an expression for N_{phot} as a function of FLUX_ISOCOR.**

$N_{phot} = (2.36\, e^-/\text{count}) \cdot (\text{Flux\_Isocor} + \text{Background} \cdot \pi\left(\frac{\text{FWHM}}{2}\right)^2)$

$N_{phot} = (2.36\, e^-/\text{count})\left(\text{Flux\_Isocor} + 571\left(\pi\left(\frac{9}{2}\right)^2\right)\right)$

$N_{phot} = (2.36)(\text{Flux\_Isocor} + 76325.42)$

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

Katie Sheng

Upload this worksheet and the pdfs of your jupyter notebook(s) to the Gradescope assignment link on the "Lab 3" 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. Clearly label the different parts of the lab using the Markdown features of Jupyter notebook and using code comments. You want to make it easy for the TA to find your work.

Assume that you have read in the catalog file and set an array called mags equal to the column MAG_ISOCOR and an array called mag_errs equal to the column MAGERR_ISOCOR. Explain why the following code will produce nonsense if you use the array mag_errs_sorted for the error bars of mags_sorted:

```
mags_sorted = np.sort(mags)
mag_errs_sorted = np.sort(mag_errs)
```

It's because the sorted error bars and the sorted mags don't match up in the catalog file, i.e. they are out of order.

To sort the arrays in the same way, try `np.argsort`. Experiment yourself a bit to get it working, and read the documentation page.

Now remake your plot of magnitudes with error bars, but sorted by magnitude on the horizontal axis. Note: think for a minute about whether or not you want to also sort the column NUMBER.

Do the estimated errors vary with magnitude in a systematic way? Explain.

Yes, the greater the magnitudes got, the larger the errors get.

So far we have used Source Extractor to obtain only the most minimal information about the objects in the detector field of view–the positions and fluxes from each detected object. We will now look at Source Extractor output files for which we have asked the program to do a more thorough job. Typically we want more kinds of information about each object for one of three reasons: to represent the data in a more convenient way, to display the result of a different way to estimate a number (such as the flux) for which we already have a value, or to give information about the reliability of the results displayed in other columns.

We will continue to work with the catalog given by `cluster1.cat`, but now will load the columns corresponding to FLUX_ISOCOR and FLUXERR_ISOCOR. Open the catalog file in a text editor like notepad if you cannot find them. As usual, remember that Source Extractor counts from 1, while python indexes from 0.

Now we will check the relationship between flux and magnitude (which is also given in the textbook by Chromey). **Make a scatter plot with MAG_ISOCOR on the x-axis and FLUX_ISOCOR on the y-axis. Once you have done this, adjust the vertical axis to have a logarithmic scaling (check `pyplot.yscale`).**

Once you have this scatter plot, try fitting a line by eye. The mechanics of how to actually do this properly are beyond the scope of this class. **For now, please define a variable x by**

↙ Index this later. (i.e. take out "bad flag" data)

Open the image in ds9, set the "scale" option to "zscale," and estimate the size of the images of stars (in pixels). How do these sizes compare with the FWHM_IMAGE values in the catalog file?

~~38 px~~ (other estimates: 20 px, 11 px, 8 px)

~~the average FWHM value is 17.12 px, some others is really~~

~~about 20% of my estimate~~ , Max: 22.25, Min: 2.53

Now try setting the ds9 "scale" to "min/max" and "linear" and estimate the sizes of a few of the star images (necessarily the brightest ones, since those are all you can see this way). Use the cursor to read the pixel values. How do the sizes you estimate in this way compare to the FWHM_IMAGE values in the catalog file?

Big star size estimates: 13 px, 10 px, 10 px

Set the "Horizontal Graph" option in the "View" tab, and estimate the FWHM of a bright star from this graph. How does this estimate compare with the other two estimates you gave above?

My estimate is about 9 px

Finally, in your jupyter notebook, define integer magnitudes covering the range of magnitudes in `cluster1.cat`:

```
x = np.arange(int(np.amin(mags)), int(np.amax(mags)) + 1)
```

and evaluate your expression for magnitude uncertainties at these integer x values. Overplot this on your plot of `MAGERR_ISOCOR` vs. `MAG_ISOCOR` using a thick, red line.

Consistency and Systematic Errors

Now we know how to use Source Extractor output to estimate the fluxes and magnitudes of stars, and also the precision that simple physics says we should be achieving. But how well do we know our errors, really? Might effects other than photon counting statistics be dominant? And what about systematic errors, which give us consistent and repeatable wrong answers? How can we test for these?

SExtractor can estimate source fluxes in several ways – two of these are isophotal photometry (which deals well with objects having funny shapes) and aperture photometry (which works well for perfectly round objects, as star images are supposed to be). Check section 7.4 of *Source Extractor for Dummies* for an explanation of what these things mean. The picture on p. 41 is particularly helpful.

Since there are different ways to estimate what ought to be the same quantities, let's see if we get the same answers with aperture photometry as we did with isophotal photometry. If not, the differences tell us something about our errors. In your jupyter notebook, make a figure with two graphs, one on top of the other. For this task, you'll want to use subplots sharing the x-axis (read the documentation page for `pyplot.subplots`). **On the top axes, plot `MAG_ISOCOR` on the x-axis and `MAG_APER` on the y-axis.** You should expect a tight but not perfect correlation between the two measurements, with a few dramatic outliers. The wide range in plotted magnitudes makes it hard to see the errors. **On the lower graph plot `MAG_ISOCOR` on the x-axis and, on the y-axis, plot (`MAG_ISOCOR` – `MAG_APER`). Choose your y-axis plot range carefully, to show the most information. If necessary, sacrifice a few outliers to show the typical scatter better.**

What is the peak-to-peak scatter in the difference (`MAG_ISOCOR` - `MAG_APER`), if you ignore extreme outliers? How does this compare with the tabulated uncertainties `MAGERR_ISOCOR` and `MAGERR_APER`? What do you conclude from this?

The peak-to-peak scatter in Mag_Isocor - Mag_Aper is from about -4 to 2.5. A line of best fit would be near zero, slightly negative. The line would be similar to the ~~Mag~~ error values in the sense that they are almost 0.

Identify the stars corresponding to 3 of the most extreme outliers in your upper plot, and look at them using ds9. Can you describe the source(s) of these systematic errors?

Physics 134L, Fall 2021
Professor Max Millar-Blanchaer

Lab 3 **Names:** Tim Marquez, Katie Sherry

Due date: Monday, Nov. 1, by 5:00 pm, through Gradescope.

Read the entire lab before you begin, so that you know what to expect and how to budget your time. In this lab, we will look at the brightness of stars as measured in astronomical units called "magnitudes." Toward the end of this lab, and more seriously in the fourth lab, we will use the brightnesses of stars in clusters to infer important physical parameters of the clusters and of the stars. Throughout this lab we will use two main files: `cluster1.fits` and `cluster1.cat`. The `cluster1.cat` file (we've used them in the past), originally came from a program known as Source Extractor and contains various quantities about all the stars found by the program that we will explore. At the end of his lab, we will learn how to replicate similar functionality using modern python packages.

Astronomical Magnitudes

We'll begin by making a two-dimensional plot of `cluster1.fits` and overplotting the locations of the stars given in `cluster1.cat`. Refer to the first lab if you have forgotten how to do this. **As in that lab, please use a scale that shows the full dynamic range of the image and that looks nice: no garish color schemes, and few (if any) pixels beyond the limits of your color scale. Please do this in a new jupyter notebook.** The x and y pixel locations might be in different locations than they were in `object.cat` from Lab 1, so please check and make sure that you are reading the correct column. You may open `cluster1.cat` in a text editor (like notepad) to double-check the contents of each column.

The file `cluster1.cat` has many columns besides the x and y positions. We will now look at the information in the columns `MAG_ISOCOR` and `MAGERR_ISOCOR`. **First, make a scatter plot showing `MAG_ISOCOR` as a function of `NUMBER`, i.e., simply plot the magnitudes as a function of the order in which they appear in the catalog. Overplot the error bars given by `MAGERR_ISOCOR`. Check the documentation for `pyplot.errorbar` if you are unsure of how to get started.**

It's hard to interpret that previous plot; it looks like a bit of a mess. **To make it easier to see what is going on, let's sort the stars by magnitude.** You'll want to sort the magnitudes and their uncertainties in the same way. In other words, you want the same uncertainty to remain associated with each measured magnitude.

Now use the flux-vs-magnitude expression from page 2 to write an expression for N_{phot} as a function of MAG_ISOCOR.

Since $\text{Flux} = 10^{(-0.4\,\text{Mag})}$

$$\therefore \boxed{N_{phot} = 2.36\left(10^{-0.4\,\text{Mag\_Isocor}} + 36325.42\right)}$$

The expected counting error (measured in photo-electrons) is the square root of N_{phot}. To get the noise in units of counts, we must divide by the GAIN. Write an expression for this counting error as a function of MAG_ISOCOR.

$$\boxed{\frac{\sqrt{N_{phot}}}{\text{Gain}} = \left(10^{-0.4\,\text{Mag\_Isocor}} + 36325.42\right)^{1/2}}$$

Now the tricky part. What error (in magnitudes) is implied by this error in the measured value of N_{phot}? Assume that the error is small compared to N_{phot} itself (i.e., that $N_{\text{phot}} \ll 1$), and show that if

$\text{mag} = m_1 - 2.5\log_{10}(\text{flux})$

(where m_1 is the magnitude that yields flux $= 1$), then

$\delta\text{mag} \approx 1.086\frac{\delta\text{flux}}{\text{flux}}$.

Derive this expression below. You may use standard propagation of uncertainties and take the square root.

Now combine this expression and the one for counting error to write an expression for the expected error in magnitudes as a function of the object magnitude.