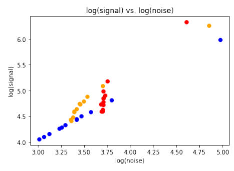
Kennedy Robinson

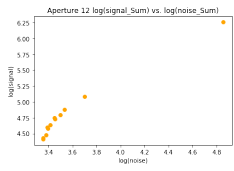
```
In [1]: %pylab nbagg
        %pylab nbagg
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import array
        from matplotlib.pyplot import *
        %matplotlib inline

        Populating the interactive namespace from numpy and matplotlib
        Populating the interactive namespace from numpy and matplotlib
```

```
In [2]: meansignal_sum6 = [14237.225,18259.944375, 11346.766781,12437.151875,18981.198438,973918.71563,38672.12,28002.898125,2
        6925.4925,31458.519688,64302.200313,21200.245313]

        sigma_sum6 = [1323.9876353,1704.8505011,1036.0358449,1163.236844,1810.0083419,93877.922605,3714.435517,2628.0533828,26
        14.5970534,2956.4096675,6240.2201715,1981.7102169,]

        meansignal_sum12 = np.array([30309.077813,38317.936563,25857.120625,27230.224688,39779.210313,1813989.9688,76105.50187
        5,56027.665938,53907.312188,62334.325938,122499.14688,43584.384063])

        sigma_sum12 = np.array([2402.4656658,2480.0498959,2279.0372509,2262.2192534,2451.6150603,71550.124238,3399.0839217,282
        9.1542824,2860.0239785,3136.5192019,5006.5843389,2604.8019448])

        meansignal_sum15 = [36683.72125,45696.143125,32510.195625,33349.475625,47226.316875,2010412.875,87600.19,65180.715,628
        94.6175,72211.529063,138629.06563,51740.77625]

        sigma_sum15 = [3530.7127093,3574.3418731,3514.979962,3413.5694601,3450.6277625,54708.325154,3894.2418978,3719.984326,3
        729.6820887,3952.6294976,4850.2823373,3636.0136737]

        meansignal_sum18 = [42929.10125,52591.13625,39176.395625,39439.311563,54062.520938,2133897.5313,97227.083438,73177.932
        813,70906.1975,80705.614063,151019.77188,59407.938125]

        sigma_sum18 = [5024.0190788,5059.5356173,5043.0863788,4887.347919,4832.4252067,40795.624416,5108.9485567,5154.35407,51
        30.7586996,5331.2258265,5618.410005,5100.2833238]
```

```
In [3]: plot(log10(sigma_sum6),log10(meansignal_sum6),'o',color='blue')
        xlabel('log(noise)')
        ylabel('log(signal)')
        plot(log10(sigma_sum12),log10(meansignal_sum12),'o',color='orange')
        plot(log10(sigma_sum18),log10(meansignal_sum18),'o',color='red')
        title('log(signal) vs. log(noise)')

Out[3]: Text(0.5,1,'log(signal) vs. log(noise)')
```



```
In [4]: xlabel('log(noise)')
        ylabel('log(signal)')
        #plot(log10(meansignal_sum12),log10(sigma_sum12),'o',color='orange')
        title('Aperture 12 log(signal_Sum) vs. log(noise_Sum)')
        plot(log10(sigma_sum12),log10(meansignal_sum12),'o',color='orange')

Out[4]: [<matplotlib.lines.Line2D at 0x7fe50b83e128>]
```



```
In [5]: #Scaled sky signal to the aperature area
        sky_signal= [6811.208,6794.661,6919.194,6832.245,6531.822,7727.604,6960.984,7176.612,6840.84,7183.706,7159.43,7051.075
        ]
        sky_area_noise = [2202.597,2188.092,2197.733,2146.434,2104.073,2245.764,2215.867,2230.032,2222.699,2235.775,2226.949,2
        239.398]
        Area = [452.63705,452.6338156,452.6292469,452.6558938,452.6306094,452.6929375,452.6124031,452.6372563,452.7025313,452.
        5569406,452.6799125,452.6447875]
        sky_noise = []
```

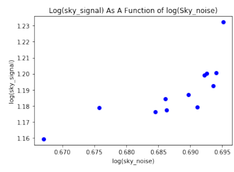
pairing individual measurements of O+S and S, subtracting the two, and calculating σ for these numbers. Or just calculate σ of (O+S)-S over the number of individual measurements for each star.

Looking at star 2, the flux signal = 31523.27656 and the flux σ = 1308.940778.

Now when we take the Sum of star 2 and subtract the Mksy*area, ((Object+sky)-sky), we get the the mean = 31523.28 and σ = 1288.326

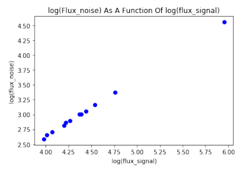
Therefore, approximetly the same σ.

```
In [6]: msky = [15.0466244,15.0117606,15.2864994,15.0934313,14.4303997,17.0697806,15.3793803,15.8551184,15.1112006,15.8740031,
        15.8160219,15.5764997]
        print("the sky signal in DU/pix is", msky)
        scale_sky_sigma_DU = np.array([25.66206,25.55962,25.65346,25.04702,24.54859,26.19321,25.86418,26.0384,25.95094,26.1274
        6,26.01312,26.10026])
        msky_noise = scale_sky_sigma_DU*0.189225
        print("The sky noise in DU/pix is:",msky_noise)
        plt.plot(log10(msky_noise),log10(msky),'o',color='blue')
        title('Log(sky_signal) As A Function of log(Sky_noise)')
        xlabel('log(sky_noise)')
        ylabel('log(sky_signal)')

        the sky signal in DU/pix is [15.0466244, 15.0117606, 15.2864994, 15.0934313, 14.4303997, 17.0697806, 15.3793803, 15.8
        551184, 15.1112006, 15.8740031, 15.8160219, 15.5764997]
        The sky noise in DU/pix is: [4.8559033  4.83651909 4.85427597 4.73952236 4.64520694 4.95641016
         4.89414946 4.92711624 4.91056662 4.94396862 4.92233263 4.9388217 ]

Out[6]: Text(0,0.5,'log(sky_signal)')
```



```
In [7]: flux_signal = np.array([23497.86969,31523.27656,18937.92719,20397.97969,33247.38844,1806262.25,69144.51813,48851.05375
        ,47066.47313,55150.62,115339.7281,36533.30938])
        flux_signal_div2 = flux_signal/2
        flux_noise = np.array([1012.656767,1308.940778,773.1262321,909.5965475,1458.691654,71843.47919,2910.03176,2010.339623,
        2031.431518,2266.368145,4684.620777,1562.233958])
        flux_noise_div2 = flux_noise/2

        print('The flux signal per second is:',flux_signal_div2)
        print('The flux noise per second is:' ,flux_noise_div2 )

        s____n = flux_signal_div2/flux_noise_div2
        print('The signal to noise is"', s____n )

        The flux signal per second is: [ 11748.934845  15761.63828    9468.963595  10198.989845  16623.69422
         903131.125     34572.259065  24425.526875  23533.236565  27575.31
          57669.86405   18266.65469 ]
        The flux noise per second is: [  506.3283835    654.470389     386.56311605   454.79827375
           729.345827   35921.739595    1455.01588     1005.1698115
          1015.715759    1133.1840725   2342.3103885    781.116979  ]
        The signal to noise is" [23.20417979 24.08304263 24.49525886 22.42530465 22.79260894 25.14163109
         23.76074347 24.29990097 23.16911632 24.33436074 24.62093168 23.38529975]
```

```
In [8]: plt.plot(log10(flux_signal_div2),log10(flux_noise_div2),'o',color='blue')
        xlabel('log(flux_signal)')
        ylabel('log(flux_noise)')
        title('log(Flux_noise) As A Function Of log(flux_signal)')

Out[8]: Text(0.5,1,'log(Flux_noise) As A Function Of log(flux_signal)')
```



What power law should relate σ and S?How do your observations compare with this expectation? Are there discrepancies at the low S end, explain.

The noise should be related to signal by N = √S. The signal versus noise plots for the Sum and flux correspond to the expected relation, with minor discrepancies at low signal, appearing to be in the region of flat field noise. The sky signal versus noise plot however varies greatly from the expected N = √S, demonstrating a fluctuations in the background noise.

Note: S = f t, where f is “flux” (counts per second). Although we haven't demonstrated it, it's fairly obvious that σ should depend on S, but not on f or t individually. Can you think of detectors and/or observing circumstances where this elementary assumption might be violated?)

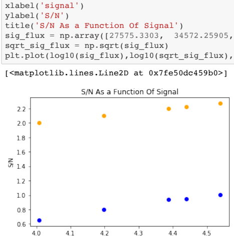
This assumption would be violated if the

```
In [9]: #sigma_flux^2 = sigma_sum^2 + sigma_msky*area^2
        f = np.array([55150.6606,69144.5181,48851.0538,20397.9797,31523.2766])
        sig_flux = f/2
        print("The flux signal is:",sig_flux )
        noise_sum = np.array([3136.519202,3399.083922,2829.154282,2262.219253,2480.049886])
        noise_msky = np.array([5.023076635,4.972461072,5.005955173,4.815358822,4.913907651])
        AREA_ID = np.array([452.5569406,452.6124031,452.6372563,452.6558938,452.6338156])
        noise_sum_squared = noise_sum**2
        noise_sky_squared = noise_msky**2
        noise_sky_sq_area = noise_sky_squared*AREA_ID
        sigma_flux_squared = noise_sum_squared + noise_sky_sq_area
        noise_flux = np.sqrt(sigma_flux_squared)
        print("The noise of the flux is:", noise_flux)
        Sig_to_noise = sig_flux/noise_flux
        print("The signal to noise ratio is S/N:", Sig_to_noise)

        The flux signal is: [27575.3303  34572.25905 24425.5269  10198.98985 15761.6383 ]
        The noise of the flux is: [3138.33894026 3400.72970369 2831.15821779 2264.53791952 2482.25239535]
        The signal to noise ratio is S/N: [ 8.78660044 10.16612964  8.62739735  4.50378409  6.34973234]
```

Calculate the signal-to-noise ratio for your final (sky subtracted) counts for each star. Plot both log(S/N) vs. log (S), and log (N) vs. log (S). (Here, the noise N is the same as σ, and S is sky-subtracted.) What relation between S/N and S would be expected to hold for observations limited by photon statistics in the star light? Plot this relation on your graphs, and comment on how well it fits. Most importantly, comment on, and explain, any discrepancies that (may) exist at the bright and faint ends of your plot.}

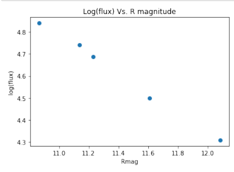
The S/N should be proportional to √S. The graphs have identical shapes, however, the S/N vs. S has half the S/N of the √S vs. S plot. There is a kink at higher signals in the S/N vs. S plot. This is likely due to scintilation noise.

```
In [10]: plt.plot(log10(sig_flux),log10(Sig_to_noise),'o',color='blue') #S/N vs. S
         xlabel('signal')
         ylabel('S/N')
         title('S/N As a Function Of Signal')
         sig_flux = np.array([27575.3303,  34572.25905,24425.5269,  10198.98985 ,15761.6383])
         sqrt_sig_flux = np.sqrt(sig_flux)
         plt.plot(log10(sig_flux),log10(sqrt_sig_flux),'o',color='orange')   #√S vs. S

Out[10]: [<matplotlib.lines.Line2D at 0x7fe50dc459b0>]
```



```
In [11]: #This Rmag was calculated using the IDs : V-(V-R)
         Rmag = [11.136,10.8672,11.231,12.084,11.61]
         flux_ap12_IDS = [55150.6606,69144.5181,48851.0538,20397.9797,31523.2766]
         log_flux = (log10(flux_ap12_IDS))
         logflux = [4.74155072,4.83975775,4.68887394,4.30958716,4.49863135]
```

```
In [12]: plt.plot(Rmag,log10(flux_ap12_IDS),'o')
         xlabel('Rmag')
         ylabel('log(flux)')
         title('Log(flux) Vs. R magnitude')
         plt.show()
```



```
In [13]: from statistics import mean
         import numpy as np
         xs = np.array([11.136,10.8672,11.231,12.084,11.61], dtype=np.float64)
         ys = np.array([4.74155072,4.83975775,4.68887394,4.30958716,4.49863135], dtype=np.float64)
         def best_fit_slope_and_intercept(xs,ys):
             m = (((mean(xs)*mean(ys)) - mean(xs*ys)) /
                  ((mean(xs)*mean(xs)) - mean(xs*xs)))
             b = mean(ys) - m*mean(xs)
             return m, b
         m , b = best_fit_slope_and_intercept(xs,ys)
         print("The slope of the graph is:",m)
         print("The y-intercerpt is:", b)

         m = -2.5*(log10(flux_ap12_IDS))
         print('The magnitudes are:', m)
         R = -0.44*m + b
         print('The brightness is:',R, '[mag arcsec-2]')

         The slope of the graph is: -0.4467079068818384
         The y-intercerpt is: 9.701735596910137
         The magnitudes are: [-11.8538768  -12.09939439 -11.72218484 -10.77396789 -11.24657838]
         The brightness is: [14.91744139 15.02546913 14.85949693 14.44228147 14.65023008] [mag arcsec-2]
```

Calculate the sky brightness [mag arcsec-2] during your observations. Compare with the sky brightness at a “good” site (~ 22.5 mag arcsec-2 in V). Explain the difference. How does this limit photometric observations of faint stars in Victoria?

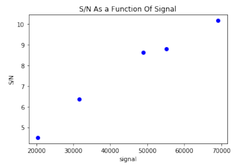
If the sky is too bright, fainter stars will be more difficult to observe as the signal from the sky would dominate the signal from the faint star. Therefore, the difference between the sky brightness and a good site is 22.5 - 11.8 = 10.7. Therefore the star is brighter by 2.5^10.7 magnitudes.

```
In [14]: sky = np.array([79.517105, 79.33286 , 80.784775 ,79.764465 ,76.26054 , 90.20891  ,81.27563,83.789764 ,79.85837 , 83.88
         9565, 83.58315  ,82.317345])

         b= 9.701735596910137
         mag = -2.5*log10(sky)
         R = -0.44*mag + b
         print(m)
         print("The sky brightness is",R, 'Mag Arcsec-2')

         [-11.8538768  -12.09939439 -11.72218484 -10.77396789 -11.24657838]
         The sky brightness is [11.79224221 11.79113402 11.79979807 11.793726   11.77226546 11.85250998
          11.80269198 11.81724566 11.79428808 11.81781433 11.8160662  11.80877609] Mag Arcsec-2
```

```
In [15]: import array
         #scaling the sky signal to correspond to the area of star aperature
         msky = [15.0466244,15.0117606,15.2864994,15.0934313,14.4303997,17.0697806,15.3793803,15.8551184,15.1112006,15.8740031,
         15.8160219,15.5764997]
         mksy_array = array.array('f',msky)
         scale = float(0.189225)
         ms= (mksy_array[:12])
         sca= np.divide(ms,scale)
         print("The scaled msky values are:", sca)
         scale_sky_sigma_DU = [25.66206,25.55962,25.65346,25.04702,24.54859,26.19321,25.86418,26.0384,25.95094,26.12746,26.0131
         2,26.10026]
         object_sigma = [2341.378,2419.296,2219.008,2203.653,2393.252,70426.26,3331.845,2766.179,2796.649,3069.102,4918.332,254
         2.863]

         The scaled msky values are: [79.517105 79.33286  80.784775 79.764465 76.26054  90.20891  81.27563
          83.789764 79.85837  83.889565 83.58315  82.317345]
```

```
In [16]: #sigma_flux^2 = sigma_sum^2 + sigma_msky*area^2
         f = np.array([55150.6606,69144.5181,48851.0538,20397.9797,31523.2766])
         sig_flux = f/2
         print("The flux signal is:",sig_flux )
         noise_sum = np.array([3136.519202,3399.083922,2829.154282,2262.219253,2480.049886])
         noise_msky = np.array([5.023076635,4.972461072,5.005955173,4.815358822,4.913907651])
         AREA_ID = np.array([452.5569406,452.6124031,452.6372563,452.6558938,452.6338156])
         noise_sum_squared = noise_sum**2
         noise_sky_squared = noise_msky**2
         noise_sky_sq_area = noise_sky_squared*AREA_ID
         sigma_flux_squared = noise_sum_squared + noise_sky_sq_area
         noise_flux = np.sqrt(sigma_flux_squared)
         print("The noise of the flux is:", noise_flux)
         Sig_to_noise = sig_flux/noise_flux
         print("The signal to noise ratio is S/N:", Sig_to_noise)

         The flux signal is: [27575.3303  34572.25905 24425.5269  10198.98985 15761.6383 ]
         The noise of the flux is: [3138.33894026 3400.72970369 2831.15821779 2264.53791952 2482.25239535]
         The signal to noise ratio is S/N: [ 8.78660044 10.16612964  8.62739735  4.50378409  6.34973234]
```

```
In [17]: plt.plot(f,Sig_to_noise,'o',color='blue')
         xlabel('signal')
         ylabel('S/N')
         title('S/N As a Function Of Signal')

Out[17]: Text(0.5,1,'S/N As a Function Of Signal')
```



```
In [18]: # S/N = [((flux/2 * gain)*t) / √((flux/2 * gain)*t + (msky/2 * gain * area)*t + ((sigma_singleRO * gain)^2 * area))]
         # a =
         Msky_signal_Ids = np.array([15.01176063,15.37938031,15.85511844,15.09343125,15.01176063])
         msky_signal_div2 = Msky_signal_Ids/2
         single_readout = float(1.3535111648496958)
```

```
In [19]: AREA_ID = np.array([452.5569406,452.6124031,452.6372563,452.6558938,452.6338156])
```

```
In [20]: sigma_fluxes_ids= np.array([2266.368145,2910.03176,2010.339623,909.5965475,1308.940778])
         sigma_fluxes_div2 = sigma_fluxes_ids/2
         print(sigma_fluxes_div2)
         print(noise_flux)

         [1133.1840725  1455.01588    1005.1698115   454.79827375  654.470389  ]
         [3138.33894026 3400.72970369 2831.15821779 2264.53791952 2482.25239535]
```

```
In [21]: # When V = +20, flux = 10^20/-2.5 = 0. 00000001 so flux/2 = 0.000000005
         # t = f(S/N)
         Given_S_N = [10,20,50,100]
         msky_DU_Pix = np.array([83.889565,81.27563,83.789764,79.764465,79.33286])
         AREA_ID = np.array([452.5569406,452.6124031,452.6372563,452.6558938,452.6338156])
         g = 3
         #area = pi*(0.435)**2
         fl = float(0.000000005)

         single_readout = float(1.3535111648496958)
         sky_div2 = msky_DU_Pix/2
         a = fl*g
         b = sky_div2*g*AREA_ID
         c = (((single_readout*g)**2)*AREA_ID)

         A = a**2
         B = -((100)*(a + b))
         C = -(100)*c

         # For S/N = 10
         t10_pos = (B + np.sqrt((B**2)-(4*A*C)))/(2*A)
         print(t10_pos)

         # For S/N = 20
         fl = float(0.000000005)
         single_readout = float(1.3535111648496958)
         sky_div2 = msky_DU_Pix/2
         a = fl*g
         b = sky_div2*g*AREA_ID
         c = (((single_readout*g)**2)*AREA_ID)
         A = a**2
         B = -((400)*(a + b))
         C = -(400)*c
         t20_pos = (B + np.sqrt((B**2)-(4*A*C)))/(2*A)
         print(t20_pos)

         # For S/N = 50
         fl = float(0.000000005)
         single_readout = float(1.3535111648496958)
         sky_div2 = msky_DU_Pix/2
         a = fl*g
         b = sky_div2*g*AREA_ID
         c = (((single_readout*g)**2)*AREA_ID)
         A = a**2
         B = -((2500)*(a + b))
         C = -(2500)*c
         t50_pos = (B + np.sqrt((B**2)-(4*A*C)))/(2*A)
         print(t50_pos)
         # For S/N = 100
         fl = float(0.000000005)
         single_readout = float(1.3535111648496958)
         sky_div2 = msky_DU_Pix/2
         a = fl*g
         b = sky_div2*g*AREA_ID
         c = (((single_readout*g)**2)*AREA_ID)

         A = a**2
         B = -((10000)*(a + b))
         C = -(10000)*c
         t100_pos = (B + np.sqrt((B**2)-(4*A*C)))/(2*A)
         print(t100_pos)

         [0. 0. 0. 0. 0.]
         [0. 0. 0. 0. 0.]
         [0. 0. 0. 0. 0.]
         [0. 0. 0. 0. 0.]
```

```
In [22]: #faintest star that can be observed in 10 minutes with S/N = 100
         sn = 100
         t = 600
         f = t/sn

         m = -2.5*log10(6)
         print(m)

         b= 9.701735596910137

         R = -0.44*m + b
         print(R)

         -1.9453781259591092
         10.557701972332145
```

What is the faintest star that can be observed in 10 minutes of total observing time for both Victoria and a dark site with at least 1% accuracy (S/N ≥ 100)? The above calculations depend on both the star aperture and sky annulus sizes. Explain why.

The faintest star that can be observed in 10 minutes has a brightness of 10.56. The above calculations depend on the star aperature as the aperature of an optical system is what determines the angle of the incoming rays in the image plane. The size of the aperature alters the amount of overall light that will reach the lens, ultimetly affecting the brightness of the image. A larger aperature will allow more photons to pass and hence create a brighter image. Therefore the above calculations depend on aperature size as the exposure time needed to reach a certain brightness would be smaller if the aperature is larger. The above calculations depend on sky annulus as its area is used to compute the background data, hence, allowing the noise from the background to be eliminated from the incoming photons during the exposure.

From first principles, show that the observed count rate for one of your stars “agrees” with what would have been predicted. Hint: V = 0 corresponds to ~1000 photons cm-2 s-1 A-1 incident on the top of the atmosphere at 5500A, to an accuracy of ±5%

SO R = 0 cooresponds to 1000 photons/2secs^-1/pi*(0.435)^2cm^-2* 1500Å^-1

=1000 photons / 1783.40361 cm-^2 s-^1 A-^1 = 0.560725566 ph/cm-^2 s-^1 A-^1

For very bright stars and high signal-to-noise ratios, another source of noise appears: “scintillation noise” -- a form of noise due to short timescale atmospheric fluctuations. Explain what this noise would do to your S/N vs. S plot.

Scintillation noise would create a kink in the S/N vs. S plot at higher signal levels as scintillation noise only appears for brighter stars.

Can you see any evidence of this in your data? If not, why not?

There appears to be scintillation noise present roughly around a signal of 5500 and a S/N between 8 and 9.

Why is scintillation noise not observed for fainter stars?

Scintillation noise is not observed for fainter stars as there would not be enough photons absorbed by the detector to create speckles.

Explain why errors due to not flat fielding, or to non-photometric observing conditions, appear in the S/N--S plot in a similar fashion to scintillation noise.

Flat fielding removes the noise created by pixel to pixel variations throughout the given array as well as the noise created by dust and scratches. Therefore, if flat fielding is not done the image will appear grainy. Non-photometric conditions would consist of cloudy skies with a transparency over 2%. This would mean the incoming photons would be blocked by the elements present in the unclear sky, resulting in a blotchy image. Scintillation noise is a second order effect caused by higher altitude turbulence. Scintillation noise creates a propagation in the curvature of the wavefront resulting in speckles. Therefore, all 3 errors mentioned created “speckle” like images and will therefore appear in the S/N vs. S plot in a similar fashion.

In the case where many stars exist on a single CCD frame, and where there is at least one bright, non-saturated, uncrowded star on the frame, it is better to use the brighter star(s) as ``template(s)'' for the point spread function, and then use least squares (or maximum likelihood) fitting to obtain the brightness of other stars relative to this star. Even for uncrowded fields, this results in significantly better photometry, especially for faint stars. Why?

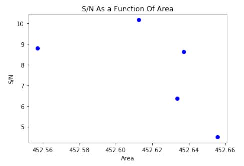
In non-coherent imaging systems such as a telescope, the image intensity formation is linear meaning that if two objects are observed at the same time, the resulting image will be equal to the sum of the two independent objects. It is easier to determine the magnitude of a brighter star, therefore, taking the brighter star as the template for the point spread function would mean that the remaining intensity would correspond to the fainter objects in the image. The method of least squares approximates the solution by minimizing the sum of the squares of the remainders left over in the results of every calculation. Therefore this method would be ideal for finding the brightness of faint stars as the intensity would be determined through the remainders.

Assume that measurements of the brightness of a star are completely dominated by background noise (sky or readout). Further assume that the sky level is well measured. For a 2D Gaussian star profile, how does the S/N of a brightness measurement depend on star aperture size? (Here you will take account of the fact that apertures of different sizes collect different amounts of light from a star.) Plot S/N vs. aperture size and comment. Explain why there is a maximum S/N. Derive the value of this maximum S/N, and the aperture radius at which it is achieved. Compare this “optimal” radius to the star aperture radius that you ended up using, and comment

In a background dominated system S/N = (S*t)/(√pi*r^2*t). Therefore if sky or readout noise dominates, the signal to noise of a brightness measurement will decrease with larger aperture radii.

There will be a maximum signal to noise as there will be a maximum aperature.

```
In [23]: AREA_ID = np.array([452.5569406,452.6124031,452.6372563,452.6558938,452.6338156])
         Sig_to_noise = np.array([8.78660044, 10.16612964 , 8.62739735 , 4.50378409  ,6.34973234])

         plt.plot(AREA_ID,Sig_to_noise,'o',color='blue')
         xlabel('Area')
         ylabel('S/N')
         title('S/N As a Function Of Area')

Out[23]: Text(0.5,1,'S/N As a Function Of Area')
```



In the case of photoelectric photometry (or, for that matter, any kind of photometry using a single element detector), one must alternate between measurements of star and sky. Consider a fixed total amount of time available to observe a star (and its associated sky). Suppose that α is the ratio of star flux to sky flux in the observing aperture. What is the optimum manner in which to divide the observing time between object and sky so as to achieve maximum S/N. (Hints: here S/N is the signal-to-noise ratio of the final object brightness – i.e., the S/N of (O+S)-S, where O is object brightness and S is sky brightness. Write down a general expression for S/N. Maximize.)

You should observe Star first, then sky near it. If the sky is bright the sky can be observed less frequently.