Homework Assignment
Statistics, Source Detection, and Noise at High Energies
Due: 3:30 PM, Wednesday, October 13

For any of the questions involving calculations, you should show the details of these calculations. You will be graded not only on the answers that you provide, but on demonstration of the steps and reasoning involved in deriving those answers.

1. In Lab 5, you derived a number of statistical quantities from the background and source sub-regions of the *Chandra* Deep Field. Record these values in the table below.

	Number of pixels	Mean counts per pixel	Fraction of pixels with >0 counts	Fraction of pixels with >4 counts
Background Region	65536	0.102	0.0967 (0.0970)	0 (8.45e-8)
Source Region	262144	0.269	0.219 (0.236)	0.00114 (9.38e-6)

In the third and fourth columns, include the expected fractions based on the Poisson distribution in *parentheses*.

2. Estimate the mean background **B**, and its standard deviation σ_B, in IDL using the binned background FITS image, **back_img_binned.fits**. Because this region has an effective exposure time that is about half that in the source region, *multiply these numbers by a factor of two*. Choose an intensity threshold based on these values corresponding to **N**=10 standard deviations above the background. Record these values below.

Using IDL, construct a histogram (not normalized) for the binned source image **source_img_binned.fits.** Based the histogram, how many sources, $N_{sources}$, are there above the threshold? (You may need to examine the histogram multiple times, focusing on different ranges.) Record this value, as well.

	Background, B	Standard Deviation, σ_B	Threshold, $B+N\sigma_B$	$N_{sources}$ > Threshold
Detection parameters	12.95	5.35	66.5	22

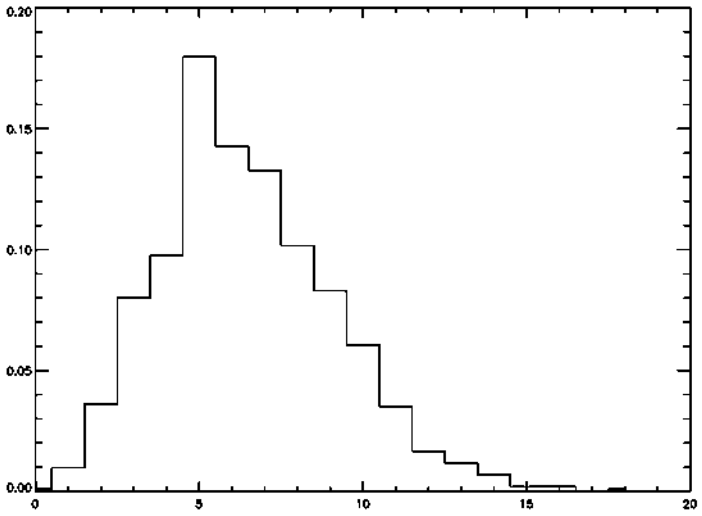
Calculate the signal-to-noise ratio for three sources – one well above the threshold, one close to the threshold, and one in-between.

	Total Counts	Background Counts	Source Counts	SNR
Bright Source	458	12.95	445	20.5
Faint Source	74	12.95	61	6.5
Intermediate Source	121	12.95	108	9.3

3. Write an IDL procedure that (a) reads a FITS image, (b) constructs a histogram and normalizes it to the number of array elements, and (c) creates a postscript file of the histogram. Set the range of pixel values in the plot (*xrange*) equal to the minimum and maximum values of the image, where these are determined *within* the procedure. Compile the program in IDL using .run, apply it to **cdfn_back_img_binned.fits**, and *attach hardcopies of the program and postscript file to your homework*.

A minimum procedure should, schematically, look something like this:

```
pro basic, fitsfile
image = readfits(fitsfile,hdr)
imhist=histogram(image, binsize=1, locations=hbins, /nan)
npixels_image=n_elements(image)
imhist_norm=imhist/float(npixels_image)
max_image=max(image)
min_image=min(image)
print, npixels_image, max_image, min_image
set_plot, 'ps'
device, filename='histogram.ps', /landscape
plot, hbins, imhist_norm, psym=10, xrange=[min_image,max_image]
device, /close
set_plot, 'x'
end
```



****

Comments

1. In this first question, some students mistakenly calculated the fraction of pixels with 4 or more counts, as opposed to more than 4 counts. As one might expect, the background region statistics are consistent with the Poisson distribution, while the source region has an excess of pixels with, e.g., >4 counts.
2. As is the case in several other places in this assignment, it is difficult and tedious to accurately count the number of sources above the threshold by examining the histogram. For that reason there was little, if any, penalty for being a little inaccurate. However, note that one can do all of the calculations quickly and precisely using IDL, with its power in manipulating arrays. The solutions above were generated with the following procedure:

```
pro solutions
 image_back = readfits("cdfn_back_img.fits",bkghdr)
 print, 'mean in background region', mean(image_back)
 print, '# of backgound pixels',n_elements(image_back)
 image_source = readfits("cdfn_source_img.fits",srchdr)
 print, '# of sources pixels', n_elements(image_source)
 print, 'mean in source region', mean(image_source)
 hist_back=histogram(image_back, binsize=1, locations=bins_back, /nan)
 hist_back_norm=hist_back/float(n_elements(image_back))
 hist_source=histogram(image_source, binsize=1, locations=bins_source, /nan)
 hist_source_norm=hist_source/float(n_elements(image_source))
; The fraction with >0 cts is 1 minus the number with 0 counts:
 print, 'fraction of bgd pixels with >0 cts', 1.0-hist_back_norm(0)
 print, 'fraction of src pixels with >0 cts', 1.0-hist_source_norm(0)
; The fraction with >4 cts is 1 minus the sum total of elements in the subarray constructed out of the
; first 5 entries in the histogram (that are the fractions with 0,1,2,3, and 4 counts):
 farr=indgen(5)
 print, 'fraction of bgd pixels with >4 cts', 1.0-total(hist_back_norm[farr])
 print, 'fraction of src pixels with >4 cts', 1.0-total(hist_source_norm[farr])
 image_back_binned = readfits("cdfn_back_binned_img.fits",bkghdr)
 true_back=2.0*mean(image_back_binned)
 print, 'bgd mean and stand. dev.', true_back,2.0*(stddev(image_back_binned))
 threshold=2.0*(mean(image_back_binned)+10.0*stddev(image_back_binned))
 print, 'threshold', threshold
 image_source_binned = readfits("cdfn_source_binned_img.fits",srchdr)
 hist_source=histogram(image_source_binned, binsize=1, locations=bins_source, /nan)
; identify the indices of the image array above the threshold:
 index=where(image_source_binned gt threshold)
; construct the subarry of pixels above the threshold:
 sources=image_source_binned(index)
 print, '# of sources above threshold', n_elements(sources)
; print out the sources above threshold, and identify a bright, a faint, and an intermediate source:
 print, sources
 index_bright=5
 index_faint=0
 index_interm=7
 total_bright=sources(index_bright)
 total_faint=sources(index_faint)
 total_interm=sources(index_interm)
 source_bright=total_bright-true_back
```

```
 source_faint=total_faint-true_back
 source_interm=total_interm-true_back
 print, 'background cts:', true_back
 print, 'total cts: bright, faint, intermediate', total_bright,total_faint,total_interm
 print,'source cts: bright, faint, intermediate', source_bright,source_faint,source_interm
; make three vectors for source, background, total to calculate all SNR at once:
 my_sources_source_array=[source_bright,source_faint,source_interm]
 my_sources_back_array=[true_back,true_back,true_back]
 my_sources_total_array=float([total_bright,total_faint,total_interm])
 my_sources_snr_array=my_sources_source_array/sqrt(my_sources_back_array+my_sources_total_array)
 print,'SNR: bright, faint, intermediate',my_sources_snr_array
end
```