S-Lang Histogram Module Reference

John E. Davis, davis@space.mit.edu Apr 16, 2004

Contents

Chapter 1

Introduction to the Histogram Module

The `histogram` module contains several functions for the efficient creation and manipulation of one and two dimensional histograms.

One of the most useful and distinguishing features of the histogram module is the notion of reverse-indices. Simply put, reverse-indices indicate what elements went into a specified histogram. The word "reverse" signifies that it is a mapping from the histogram back to the data that produced it.

Chapter 2

Using the Histogram Module

To use the `histogram` module in a **S-lang** script, it is first necessary to make the functions in the package known to the interpreter via

```
() = evalfile ("histogram");
```

or, if the application embedding the interpreter supports the `require` function,

```
require ("histogram");
```

may be used. If there is a namespace conflict between symbols in the script and those defined in the module, it may be necessary to load the histogram package into a namespace, e.g.,

```
() = evalfile ("histogram", "hist");
```

will place the histogram symbols into a namespace called `hist`.

Once the histogram module has been loaded, functions defined by the it may be used in the usual way, e.g.,

```
require ("histogram");
    .
    .
h = hist1d (points, [min(points):max(points):0.1]);
```

where `hist1d` is the 1-d histogram function.

Chapter 3

Examples

This section presents examples using the `hist1d` and `hist2d` functions. The examples are particularly relevant to X-ray astronomy. The **evt2img** demo program distributed with the `histogram` module may be regarded as a more complex combination of the examples below.

3.1 Example 1: Light Curves and Color-Color Diagrams

Each *event* measured by the *Chandra X-Ray Observatory* `http://cxc.harvard.edu` is characterized by many attributes including exposure-number and energy. Many objects observed by the observatory undergo flares causing a change in the observed count-rate. Suppose that the values of the variables `expno` and `energy` are 1-d arrays such that `expno[i]` and `energy[i]` represent the exposure number and energy of the ith event, respectively. Then an array representing the number of events per exposure number may be constructed using the `hist1d` function:

```
expno_grid = [min(expno):max(expno)];
count_rate = hist1d (expno, expno_grid);
```

Here, `expno_grid` represents the grid used for binning the exposure numbers into the histogram. Plotting `count_rate` as a function of `expno_grid` will show how the count-rate changes with exposure number. Such a plot is known as a light-curve.

For a weak source, there may be few events in each exposure causing the measured count-rate to look noisy. The signal to noise ratio can be increased by using larger bin sizes when constructing the histogram. For example,

```
bin_size = 50;   % Use 50 exposures per bin
expno_grid = [min(expno):max(expno):bin_size];
count_rate = hist1d (expno, expno_grid)/bin_size;
```

will compute a count-rate using 50 exposures per bin.

By studying the measured count-rate, one can ascertain whether or not the source had a flare. Another important question is whether during the flare the spectrum of the source changed. For example, the X-ray spectrum of some sources will change from a soft state (low energies) to a hard state (high energies) during a flare. The mean energy per exposure may be used to get a handle upon any spectral changes. The computation of this quantity involves computing the mean energy of each event within an exposure. Although one can use brute-force methods to compute this, the simplest and most efficient is to use a histogram, but keeping track of what events went into each bin of the histogram. As before

```
bin_size = 50;
expno_grid = [min(expno):max(expno):bin_size];
count_rate = hist1d (expno, expno_grid, &rev)/bin_size;
```

computes the count-rate. Note the use of `&rev` as an additional argument to `hist1d`. After `hist1d` returns, the value of `rev` will be an array-of-arrays of indices that indicate how the events were distributed into the histogram. That is, `rev[0]` represents the list of event indices that contributed to the first histogram bin. Hence, the expression

```
mean (energy[rev[0]]) / bin_size
```

gives the mean energy of the events in the first histogram bin. The mean energy as a function of exposure number may be computed by

```
num_bins = length (expno_grid);
mean_energy = Double_Type [num_bins];
for (i = 0; i < num_bins; i++)
  mean_energy[i] = mean (energy[rev[i]])/bin_size;
```

A plot of `mean_energy` versus `expno_grid` may give an indication of how the spectrum changed during the observation.

Finally, consider the construction of a so-called *color-color* diagram. This is simply a plot of the ratios of count-rates in various energy bands. Suppose that one is interested in three energy bands: 1-2 keV, 2-6 keV, and 6-12 keV. The event list may be filtered in these three bands as follows:

```
i_low = where ((energy >= 1) and (energy < 2));
i_med = where ((energy >= 2) and (energy < 6));
i_hi  = where ((energy >= 6) and (energy < 12));
```

These filters may be used to construct count-rates in the three bands:

```
rate_low = hist1d (energy[i_low], expno_grid)/bin_size;
rate_med = hist1d (energy[i_med], expno_grid)/bin_size;
rate_hi = hist1d (energy[i_hi], expno_grid)/bin_size;
```

The color-color plot is made by plotting `rate_hi/rate_med` versus `rate_med/rate_low`.

3.2 Example 2: A true-color image

This example shows how to use the `hist2d` function to create a color-coded image. In addition to the energy and exposure number, an event is also associated with an (X,Y) coordinate representing the position on the sky where the photon causing the event originated. The three energy bands of the previous example will be used. Events with an energy in the lowest band will be represented by the color red, the events in the middle band by green, and those in the highest energy band by blue.

A full resolution image generated by the Chandra Observatory's ACIS detector consists of 8192x8192 pixels. For economy, a lower resolution 1024x1024 image will be produced. The `hist2d` function will be used to produce the individual planes of the final image:

```
xgrid = [1:8192:8];
ygrid = [1:8192:8];
r_image = hist2d (X[i_low], Y[i_low], xgrid, ygrid);
g_image = hist2d (X[i_med], Y[i_med], xgrid, ygrid);
b_image = hist2d (X[i_hi], Y[i_hi], xgrid, ygrid);
```

Here `i_low`, `i_med`, and `i_hi` are defined as in the previous example.

Chapter 4

Histogram Module Function Reference

4.1 hist1d

Synopsis

Compute a 1-d histogram

Usage

```
h = hist1d (pnts, grid [, &rev)
```

Description

The `hist1d` function bins a set of points `pnts` into a 1-d histogram using bin-edges given by the grid argument. The optional third argument `&rev` is a reference to a variable whose value will be assigned the reverse-indices of the histogram.

The value of the ith bin in the histogram is given by the number of values in the `pnts` array that satisfy `grid[i]<=X<grid[i+1]` where `X` represents the value of the candidate point. The last bin of the histogram represents an overflow bin with the right bin edge at plus infinity. The grid is required to be sorted into ascending order.

The reverse-indices array is an array-of-arrays of indices such that `rev[i]` is an array of indices into the `pnts` array that have fallen into the ith bin.

See Also

`4.3 (hist1d_rebin), 4.2 (hist2d), 4.4 (hist_bsearch)`

4.2 hist2d

Synopsis

Compute a 2-d histogram

Usage

```
h = hist1d (xpnts, ypnts, xgrid, ygrid, [, &rev)
```

Description

The `hist2d` function bins a set of `(x,y)` pairs represented by the `xpnts` and `ypnts` arrays into a 2-d histogram using bin-edges given by the `xgrid` and `ygrid` arguments. The optional fifth argument `&rev` is a reference to a variable whose value will be assigned the reverse-indices of the histogram.

The value of the bin `[i,j]` of the histogram is given by the number of (X,Y) pairs that satisfy `xgrid[i]<=X<xgrid[i+1]` and `ygrid[i]<=Y<ygrid[i+1]`. The bins at the extreme edges of the histogram represent overflow bins with the upper bin edge at plus infinity. The grids are required to be sorted into ascending order.

The reverse-indices array is a 2-d array-of-arrays of indices such that `rev[i,j]` is an array of indices into the `xpnts` and `ypnts` arrays that have fallen into the bin `[i,j]`.

See Also

`4.1 (hist1d), 4.3 (hist1d_rebin), 4.4 (hist_bsearch)`

4.3 hist1d_rebin

Synopsis

Rebin a 1-d histogram

Usage

```
new_h = hist1d_rebin (new_grid, old_grid, old_h)
```

Description

The `hist1d_rebin` function rebins a histogram `old_h` with bin edges defined by `old_grid` into a histogram with bin edges given by `new_grid`.

Unlike the `hist1d` function which returns an integer array, the `hist1d_rebin` function returns an array of doubles since the new grid does not have to be commensurate with the old grid.

See Also

`4.1 (hist1d), 4.3 (hist1d_rebin), 4.4 (hist_bsearch)`

4.4 hist_bsearch

Synopsis

Perform a binary search

Usage

```
i = hist_bsearch (x, xbins)
```

Description

The `hist_bsearch` function performs a binary search for the bin `i` satisfying `xbins[i]<=x<xbins[i+1]`. If the value `x` is greater than or equal to the value of the last bin, the index of the last bin will be returned. If `x` is smaller than the value of the first bin (`xbins[0]`), then the function will return the index of the first bin, i.e., `0`. The grid is required to be sorted into ascending order.

If the value of `x` is an array, an array of indices will be returned.

Notes

As pointed out above, if the value of `x` is less than the value of the first bin, the index of the first bin will be returned even though `x` does not belong to the bin. If this behavior is not desired, then such points should be filtered out before the binary search is performed.

See Also

`4.1 (hist1d), 4.3 (hist1d_rebin)`