Examples of S-Lang in Data Analysis *

David Huenemoerder (dph@space.mit.edu)
MIT/CXC

27 January 2003

- **manipulating arrays with the "`where()`" function;**
- **making histograms, on arbitrary grids;**
- **extending the analysis system;**
- **regridding histograms;**
- **smoothing arrays, using complex arithmetic.**

*For a detailed version of these slides, see `http://space.mit.edu/CXC/docs.html#SLang`.

The `where` Function (Your New Best Friend)

```
 x = [ 5 : 10 : 1 ] ; % generate an array
 a = ( x <= 7 ) ;     % new array, value 1 if x <= 7, 0 otherwise.
 b = ( x >= 7 ) ;     % another, but equal 1 only if x >= 7
 print(a) ;           % let's look
1
1
1
0
0
0

 print( x[ where( a ) ] );   % print selected x elements for
                             % the first conditional
5
6
7
```

```
 l = where( a and b );  % define array elements for both conditionals true
 print( x[ l ] );       % print the value of x for both being true
7

 print( where(a) );     % Directly see what where does
0
1
2
```

A More Complicated Example

```
fevt1 = "acisf01451_000N002_evt1.fits";  % pick an event file
(x,y,s) = fits_read_col(fevt1, "x", "y", "status" ); % read some columns
x0 = 4060.86 ;   % set the source position (or read, compute, ...)
y0 = 4116.83 ;

% Find events within a given distance from the source position:
l = where( hypot( x-x0, y-y0 ) < 30 ) ;

% Define relative coordinates for the selected events:
dx = x[l] - x0 ;        dy = y[l] - y0 ;

%%% Now find certain status bits

ag_bits = 0xf shl 16;           % 16:19 are for ``afterglow''
lg = where ( s[l] & ag_bits );
```

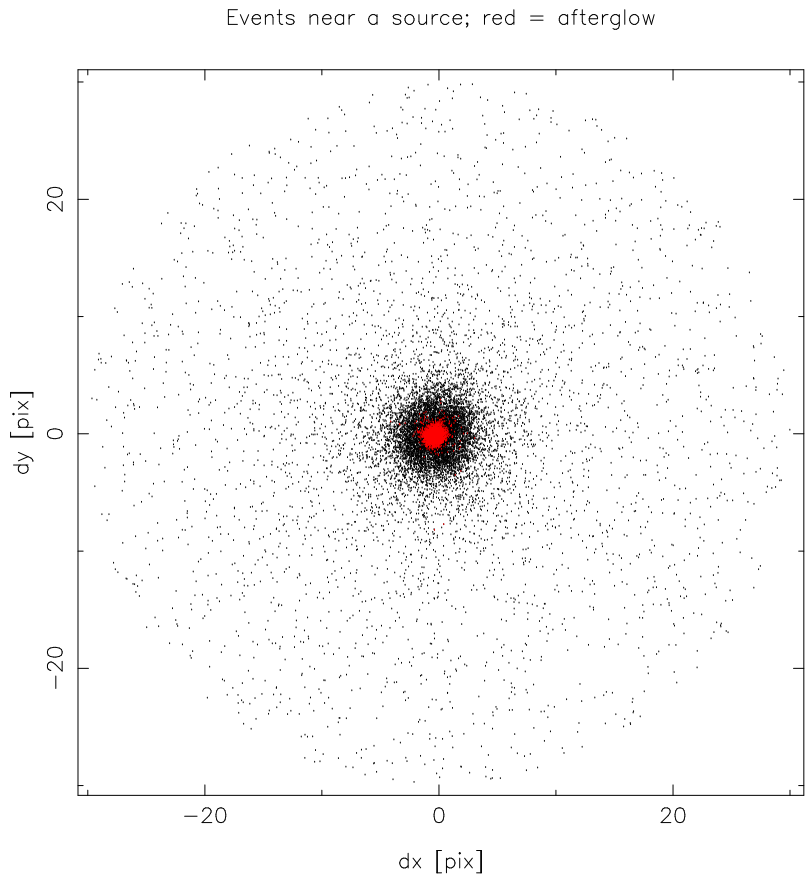

Figure 1: **Events selected by radius from a given location, with red events selected among those via afterglow bits.**

CIAO **tool equivalent operations:**

```
dmcopy acisf01451_000N002_evt1.fits"[col x,y,status,grade]"\
"[(x,y)=circle(4060.86,4116.83,30)]" src.fits

dmcopy src.fits"[exclude status=xxxxxxxxxxxx0000xxxxxxxxxxxxxxxx]"\
       srcglow.fits

ds9 -log -tile -zoom 2 -cmap bb -geometry 640x640 src.fits srcglow.fits &
```

The S-Lang example: 4 seconds (190 MB evt1 file)
The two dmcopy's take about 7 seconds, 2 output files.
Ten filters: S-Lang 14 seconds; `dmcopy` **90 seconds.**

Histograms (Your Second-Best Friend)

unix/CIAO: `dmcopy` **and** `dmextract`.
ISIS: `histogram` **and** `histogram2d`.

```
r = hypot( dx, dy );                          % create a radial distance variable
(rlo, rhi ) = linear_grid( 0.0, 30.0, 60 ); % define a grid
a =  PI*(rhi^2-rlo^2)  ;                      % area of each annulus
rprof = histogram( r, rlo, rhi ) ;            % bin into a radial profile

% look at grade 7 only...  (this is a piled source)

g = fits_read_col(fevt1, "grade" );   % read some columns
lg7 = where (g[1] == 7) ;             % sub-select a list within the list...
r7 = histogram( r[lg7], rlo, rhi ) ; % ... and bin this selection

% "on the fly" plot of histogram, for grade = 0...
ohplot(rlo, rhi,  histogram( r[where(g[1] == 0)], rlo, rhi ) / a, green  );
```

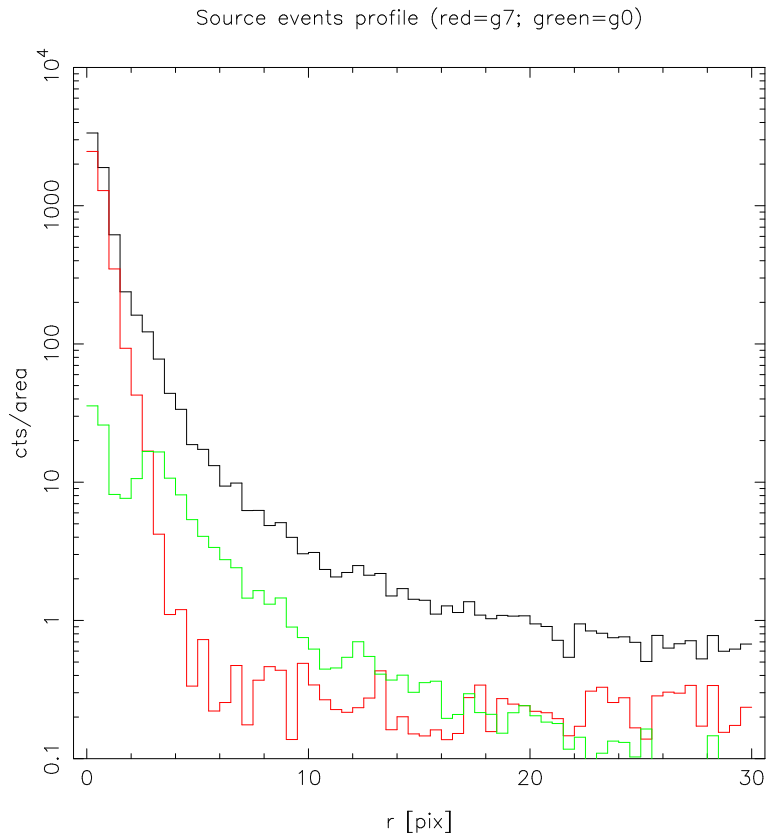

Figure 2: **Radial profiles for source events. Top curve is all grades, status values; green is grade=0, and red is grade=7.**

unix/CIAO equivalents:

```
dmextract src.fits"[bin sky=annulus(4060.86,4116.83,0:30:0.5)]" srcprof.fits
dmextract src.fits"[filter grade=7][bin sky=annulus(4060.86,4116.83,0:30:0.5)]"\
 srcprof_7.fits
dmextract src.fits"[filter grade=0][bin sky=annulus(4060.86,4116.83,0:30:0.5)]"\
 srcprof_0.fits
```

Variations: non-uniform grids:

```
(lrlo,lrhi) = linear_grid(log10(0.1), log10(30), 60);
lrlo = 10.^lrlo;
lrhi = 10.^lrhi;
la =  PI*(lrhi^2-lrlo^2)  ;                    % area of each annulus
lrprof = histogram( r, lrlo, lrhi ) ;          % bin into a radial profile
```

unix/CIAO equivalent: ? (An exercise for the reader.)

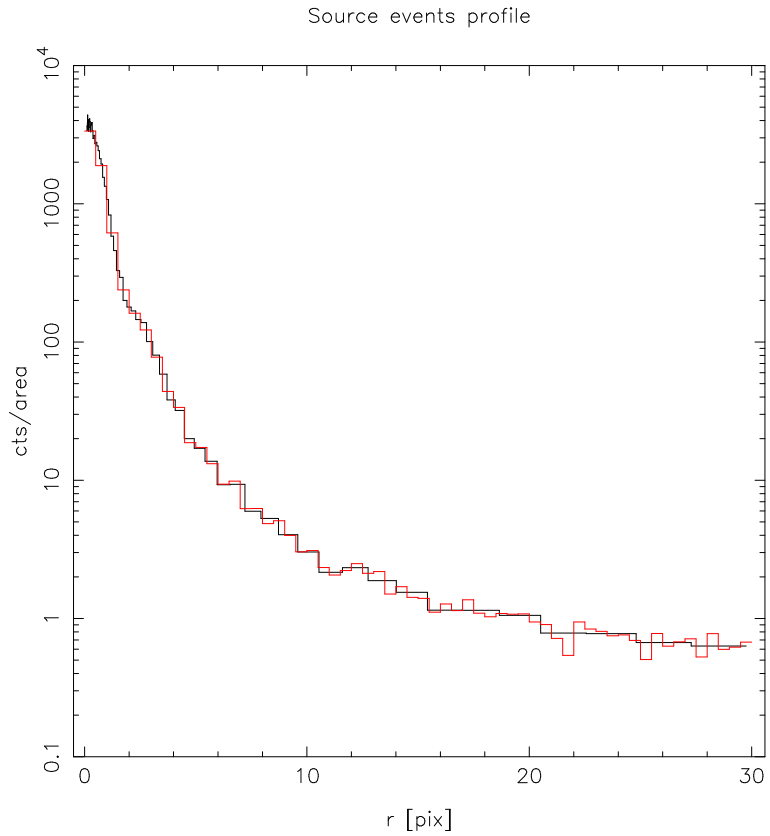

Figure 3: **Radial profile for source events on a logarithmic radial grid (black) compared to the prior linearly gridded profile (red)**

Extending the System

You want to do something similar to existing programs, but which doesn't exist, such as a cumulative distribution. Write a function!

```
define cumdist()
{
  variable h = NULL;
  if (_NARGS != 1)
  {
    message("Usage: h_cum = cumdist( h ); ");
    message("   Compute the cumulative histogram given histogram, h.");
    return h ;
  }
  variable h_in = () ;
  variable i,  len = length( h_in );
  h = @h_in;
  for (i=1; i<len; i++)    h[i] += h[i-1] ;
  return h ;
}
```

It is now a simple matter to compute the integrated radial profile:

```
evalfile("cumdist.sl");                % ``compile'' the source
crprof = cumdist( rprof );             % compute the cumulative distribution

%%% combine with filtering, histograms:
good = where( (g[1] < 7) and (g[1] != 1) and (g[1] != 5) and (not s[1]) );
crgood = cumdist( histogram( r[good], rlo, rhi ) );
```

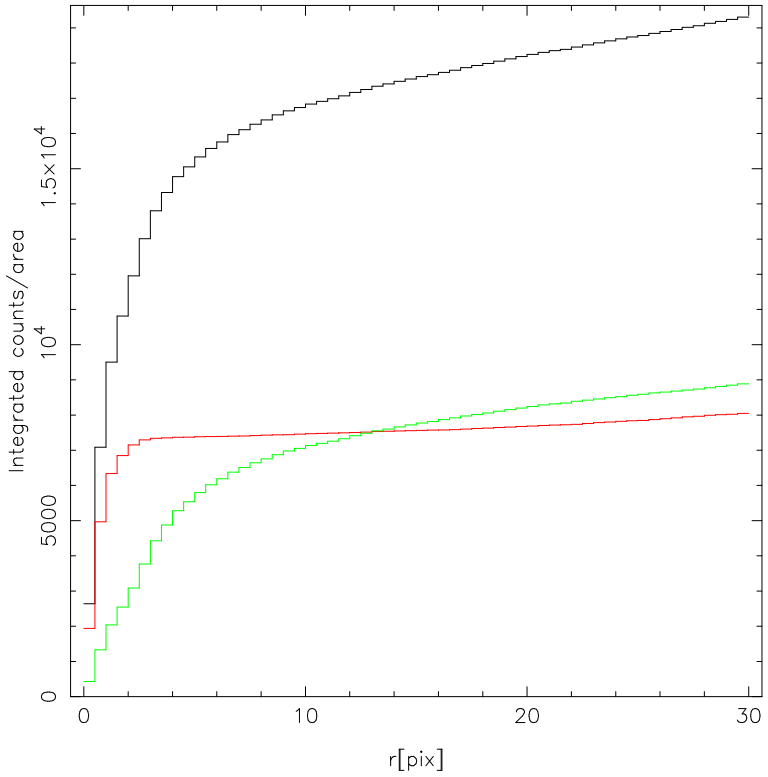

Figure 4: **Integrated radial profile for all source events (black) compared to that for grade=7 (red). green curve: good grades and status.**

Regridding

Combining existing histograms which are on different grids; e.g., summing MEG and HEG counts.

```
fpha = "acisf01451N002_pha2.fits";          % a binned spectrum file
() = load_data(fpha, [3,4,9,10]);           % load only +-1st orders' rows

d = get_data_counts(1);                     % copy to variable
xhlo = d.bin_lo;                            % define convenient variables
xhhi = d.bin_hi;
yh = d.value + get_data_counts(2).value;    % sum +-1st HEG (indices 1,2)

d = get_data_counts(3);                     % repeat for MEG (indices 3,4)
xmlo = d.bin_lo;  xmhi = d.bin_hi;
ym = d.value + get_data_counts(4).value;

yy = ym + rebin( xmlo, xmhi, xhlo, xhhi, yh );  % rebin and add HEG to MEG counts
```

Smoothing, and Complex Arithmetic

Smoothing can be accomplished by taking advantage of a low-level fast-fourier-transform function in ISIS and intrinsic complex arithmetic in S-Lang. The core use of the FFT and complex arithmetic can be seen in a few lines of S-Lang from a user-contributed Gaussian-smoothing program, gsmooth.sl:

```
...
 (ry, iy) = fft1d(y, y*0., -1);              % forward fft of data
 (rk, ik) = fft1d(karray, karray*0., -1);    % forward fft of kernel

 cy = ry + iy * 1i;          % convert to Slang complex types.
 ck = rk + ik * 1i;
 c = cy * ck ;               % convolve: fft product.
...
 (rsy, isy) = fft1d(Real(c), Imag(c), 1);     % inverse fft;
...
```

The user-contributed splot_data.sl uses the ISIS plotting and data manipulation infrastructure to "wrap" gsmooth() around the intrinsic structures to smooth the data array before display.

```
shplot(xmlo,xmhi,yy, 0.01);          % smoothed plot, summed HEG+MEG, on MEG grid
oshplot(xmlo,xmhi, ym, 0.01, 2);     % MEG +-1st order sum
oshplot(xhlo,xhhi, yh, 0.01, 3);     % HEG +-1st order sum, on HEG grid.

hplot(xmlo, xmhi, yy, 4);            % unsmoothed summed HEG+MEG
oshplot(xmlo, xmhi, yy, 0.01, 2 );   % smoothed
```

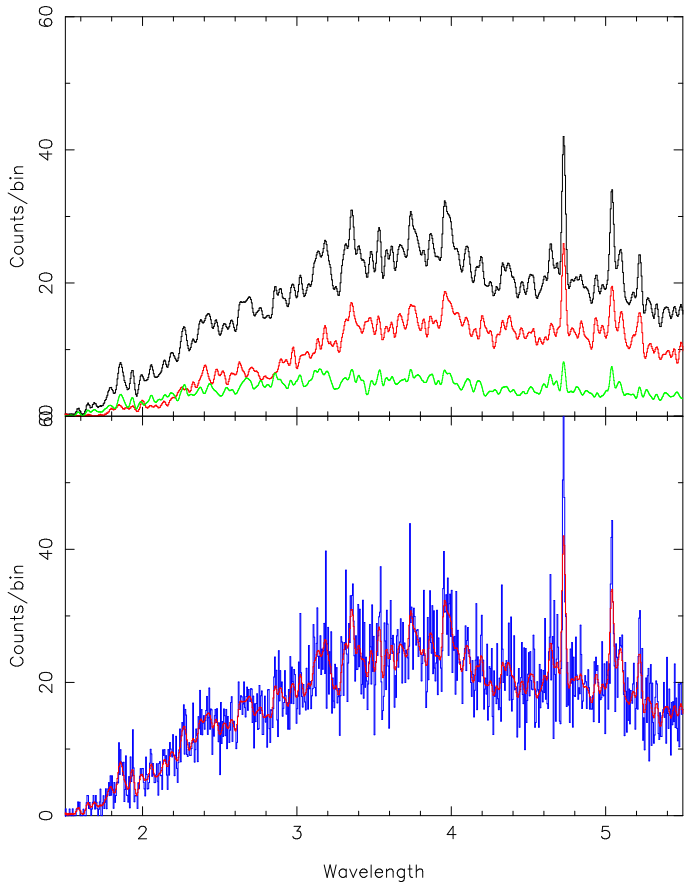

Figure 5: **The top shows smoothed plots of an HEG spectrum, MEG, and the sum. To form the sum, the HEG was regridded to MEG resolution. The bottom shows the unsmoothed summed spectrum, with the smoothed version overplotted.**

Summary

An S-Lang-based approach provides ability to

- **explore & experiment**
- **prototype new algorithms or programs**
- **apply personal customization**
- **extend an analysis system**
- **possibly achieve large performance gains**

(For a more detailed version of these slides, see the notes available at `http://space.mit.edu/CXC/docs.html#SLang`**, which includes more verbose examples, some notes regarding the** ISIS **implementation of S-Lang and availability, and a couple simple exercises to write convenient functions using "**`where`**.")**