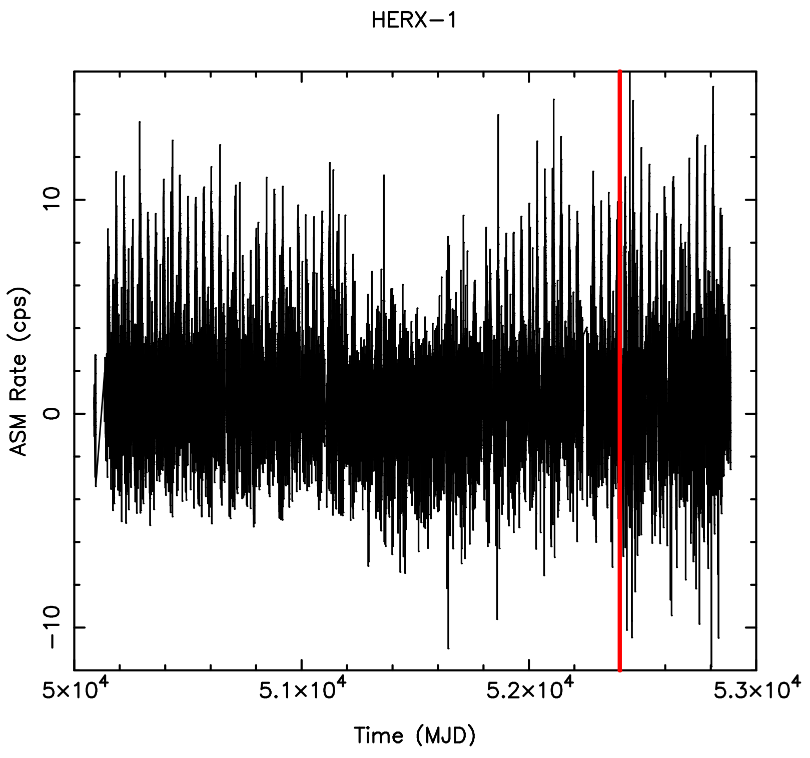
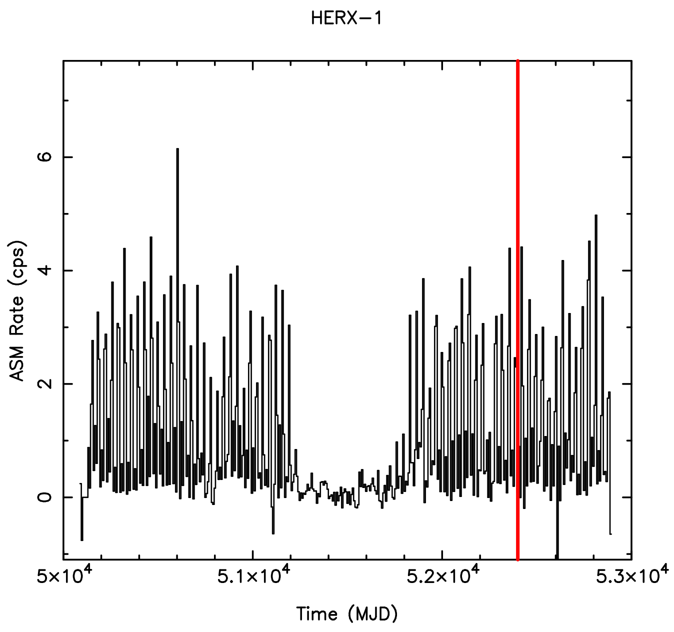
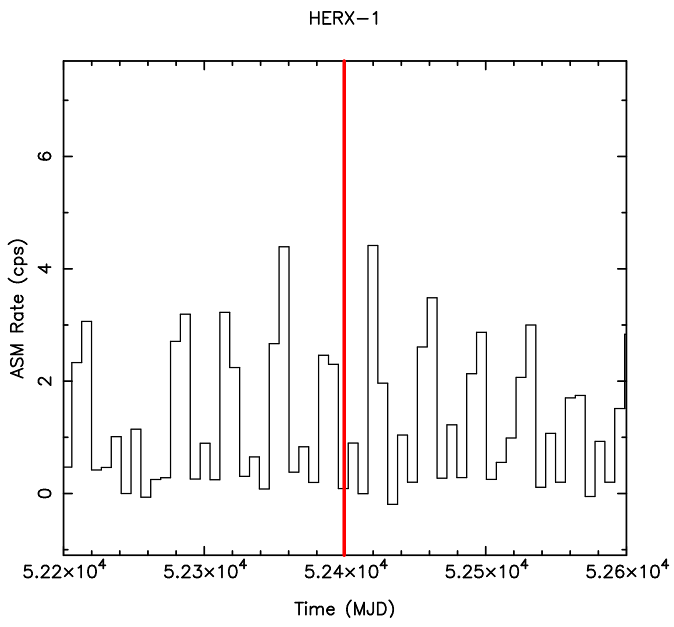
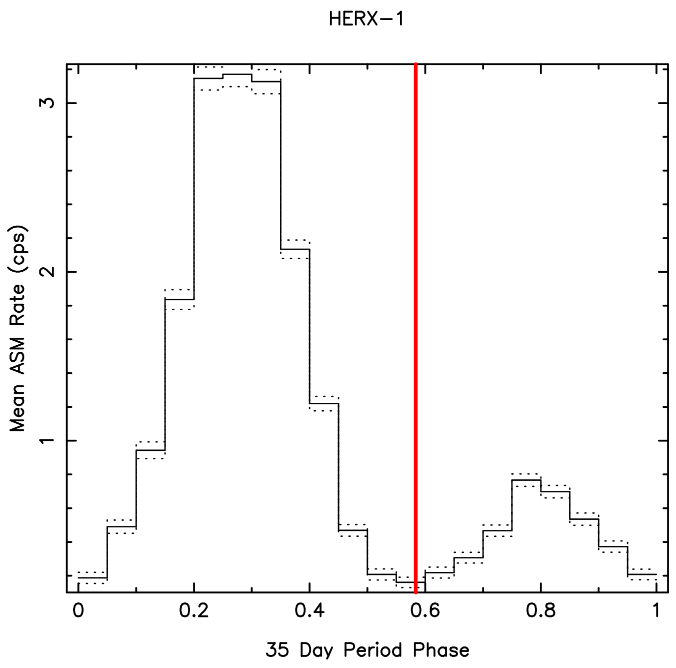


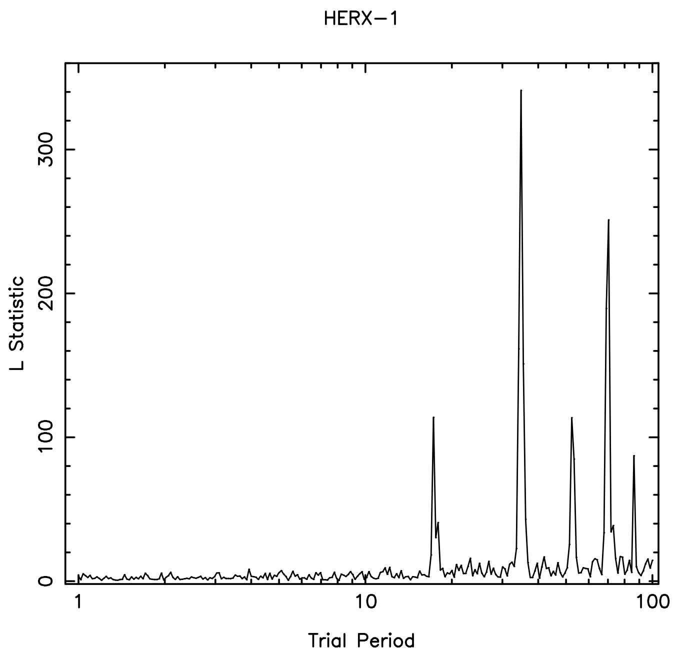
```
-----------------------------------------------------
Welcome to Sherpa: CXC's Modeling and Fitting Program
-----------------------------------------------------
Version: 3.0.2 (15 Oct 2003)

sherpa> () = evalfile("cuc_1");
Imported ISIS module version 1.1.6

%>>> Without leaving the fitting program, we can read the
%>>> relevant information from the Chandra header file ...


   mjdref = fits_read_key( "herx1.fits", "MJDREF" );
   obstrt = fits_read_key( "herx1.fits", "TSTART" );


%>>> ... which we can compare to the source behavior, as
%>>> revealed by the RXTE-ASM.  (We first read in a suite
%>>> of timing analysis routines, written in s-lang.)

   () = evalfile("sitar");

   ev = sitar_readasm( "xa_herx1_d1",,, 1.1,, 1 );

   plot( ev.time, ev.rate );
```



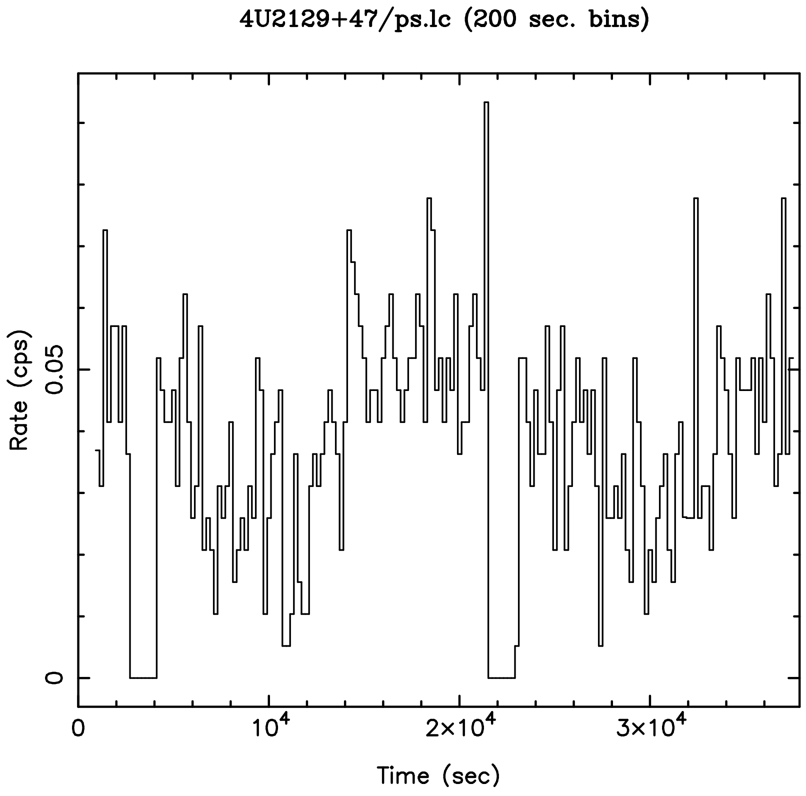


```
%>>> Rebin this rate lightcurve (or any rate lightcurve),
%>>> to make the plots more readable


   rb = sitar_rebin_rate( ev.time, ev.rate,, 7. );

   hplot( rb.bin_lo, rb.bin_hi, rb.rate );
```

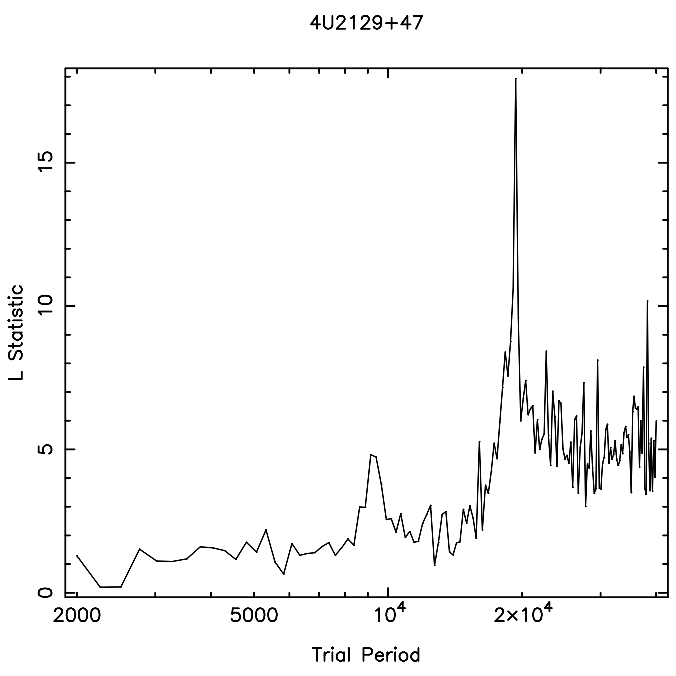


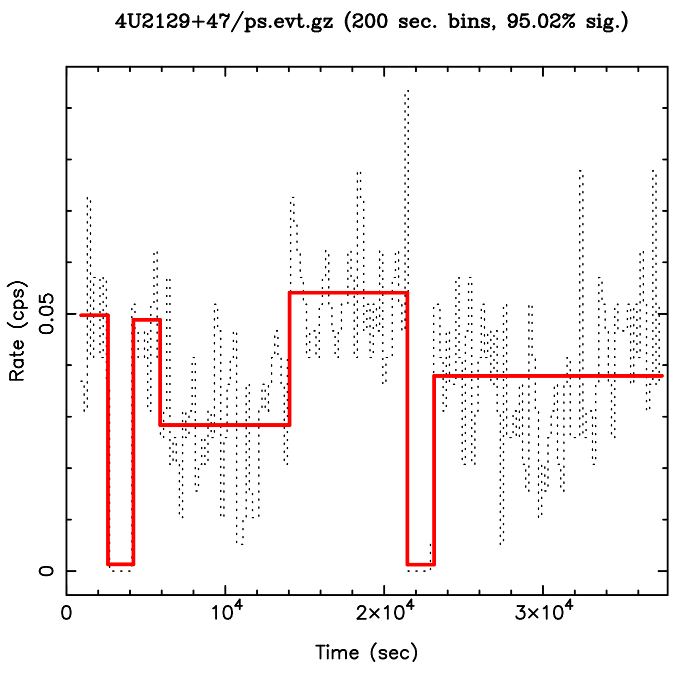


```
%>>> If we know the period, we can fold on it, and compare to
%>>> the phase of our Chandra observation


   pf = sitar_pfold_rate( ev.time, ev.rate, 35.,,,,, 51889. );

   hplot( pf.bin_lo, pf.bin_hi, pf.mean );


%>>> If we don't know the period, we can epoch fold the data
%>>> to find the best candidate period.


   ef = sitar_epfold_rate( ev.time, ev.rate, 1., 100., 10, -250. );

   plot( ef.prd, ef.lstat );
```

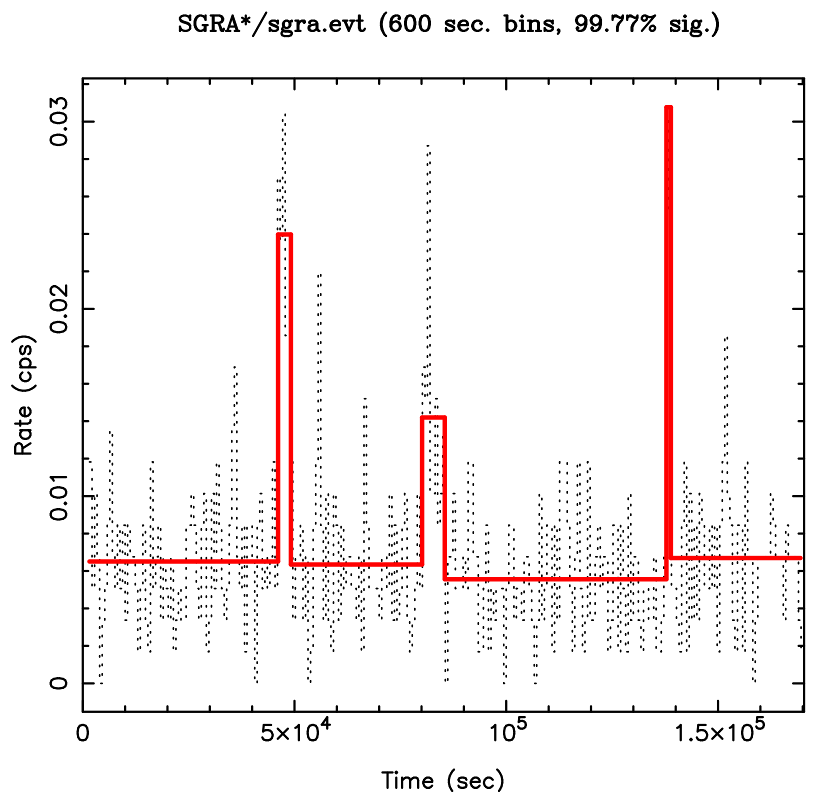


```
%>>> dmextract in CIAO 3 now creates lightcurves, with proper GTI,
%>>> exposure, and dead time correction information.  Called as:

    () = system( "dmextract \"ps.evt.gz[bin time="
                 + string(tstart) + ":"
                 + string(tstop)  + ":"
                 + string(plot_step) + "]\"
                 outfile=ps.lc clobber=yes opt=ltc1" );

%>>> or equivalently as:

    () = system( sprintf("dmextract '%s[bin time=%S:%S:%S]'
                         outfile='%S' %S", "ps.evt.gz",
                         tstart, tstop, plot_step, "ps.lc",
                         "clobber=yes opt=ltc1 verbose=0") );

    (ev.lo_t, ev.hi_t, ev.rate) =
      fits_read_col("ps.lc","TIME_MIN","TIME_MAX","COUNT_RATE");

   hplot( ev.lo_t, ev.hi_t, ev.rate );

%>>> Note that system calls can be combined with s-lang variables,
%>>> which allows powerful scripted analyses.
```

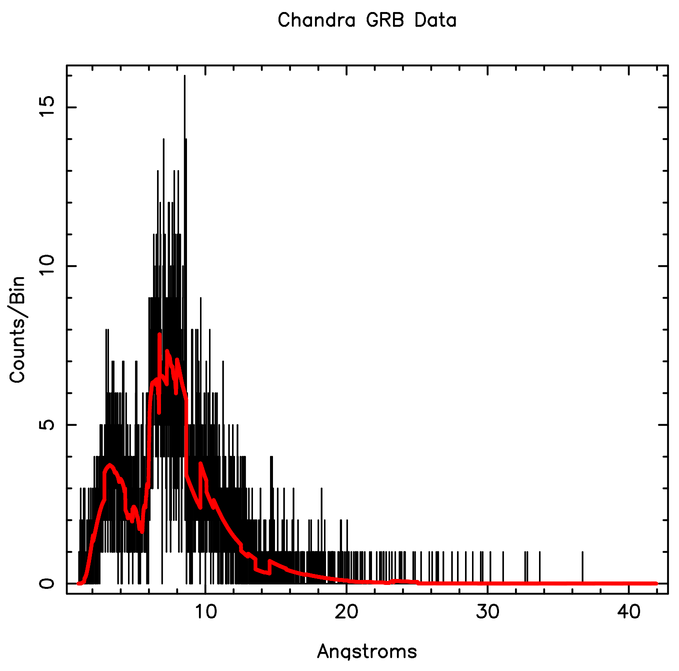


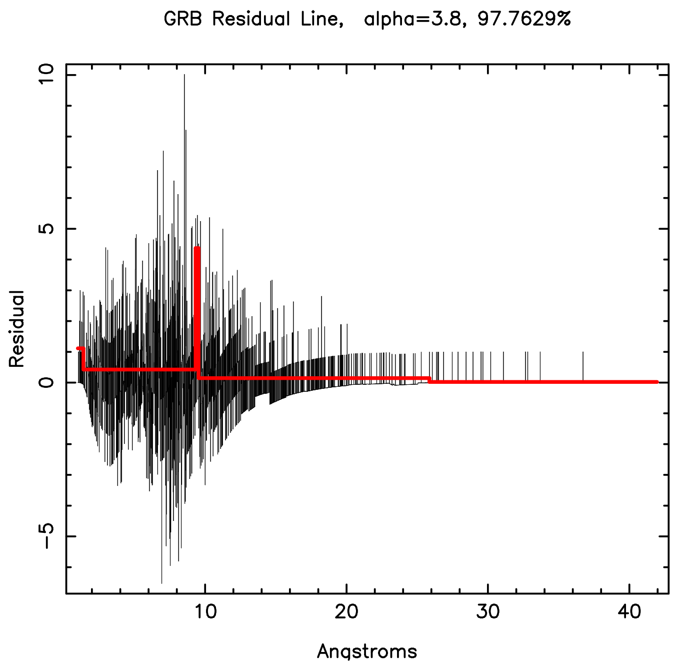


```
%>>> We can perform standard analyses on Chandra lightcurves ...


   ef = sitar_epfold_rate((ev.lo_t+ev.hi_t)/2.,ev.rate,2000,40000,10,150);

   plot( ef.prd, ef.lstat );



%>>> ... or more sophisticated analyses that take advantage of
%>>> Chandra's unique attributes (i.e., no background -> single
%>>> photons matter!)


   event_times = fits_read_col( "ps.evt.gz", "TIME" );

   cell = sitar_make_data_cells(event_times,1,clump,frame,tstart,tstop);

   ncp_prior = 3.

   results = sitar_global_optimum( cell, ncp_prior, 1 );

   hplot( ev.lo_t, ev.hi_t, ev.rate );
   ohplot( results.lo_t, results.hi_t, results.rate );
```



```
   event_times = fits_read_col( "sgra.evt", "TIME" );

   cell = sitar_make_data_cells(event_times,1,clump,frame,tstart,tstop);

   ncp_prior = 6.1

   results = sitar_global_optimum( cell, ncp_prior, 1 );

   hplot( ev.lo_t, ev.hi_t, ev.rate );
   ohplot( results.lo_t, results.hi_t, results.rate );


%>>> Mike Muno (Chandra Fellow, UCLA), without prior s-lang knowledge,
%>>> was able to make useful modifications to the code (now adopted
%>>> for very sparse lightcurves) in only a few hours, and within a
%>>> few days had applied it to over 20,000 lightcurves from
%>>> approximately 2,300 Galactic Center sources.
```





```
%>>> ‘‘counts vs. time bin’’ is conceptually no different than
%>>> ‘‘data residual vs. wavelength bin’’.  Without any code
%>>> modification, the algorithm can be applied to (absorbtion
%>>> or emission) line searches in gratings residuals.


   cell.pops = grb.data;
   cell.size = grb.model;

   ncp_prior = 3.8;

   results = sitar_global_optimum( cell, ncp_prior, 3 );
```

All the preceeding functions, complete with documentation, worked examples, sample data files, and driver functions, can be found at:

```
http://space.mit.edu/CXC/analysis/SITAR
```

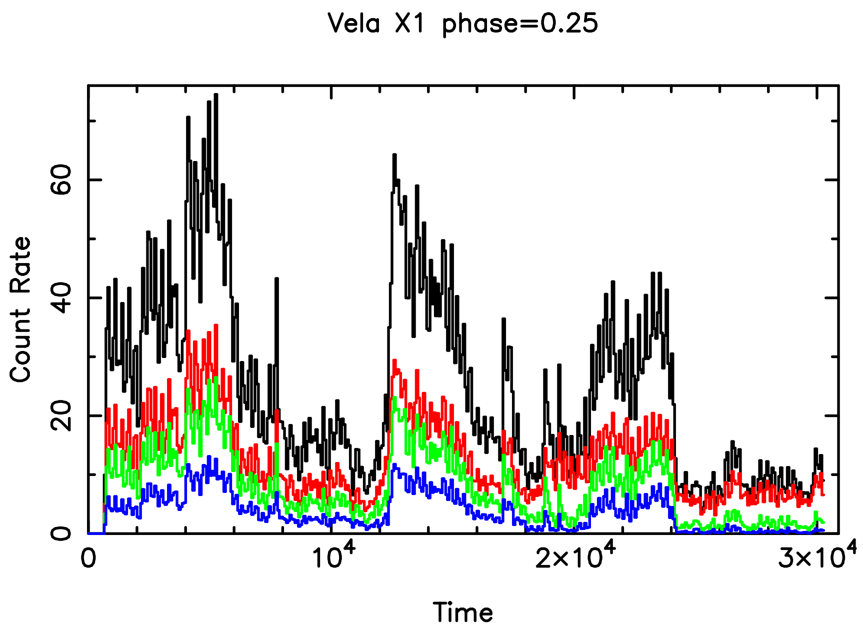
And see also:

```
http://asc.harvard.edu/ciao/threads/timing.html
```

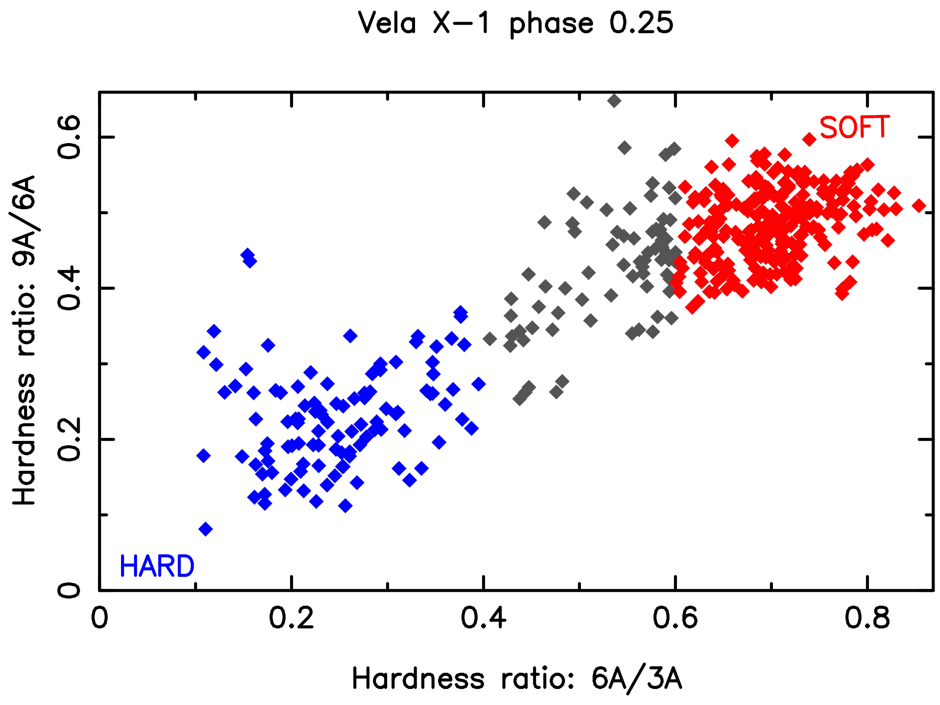
The scriptability of CIAO 3 means that SDS can prototype new tools very quickly. Advanced users, with no more effort than required to run an IDL code, can use the prototypes as soon as available. Command line versions (i.e., with parameter and ahelp files) of the most useful tools may be made available in future versions of CIAO, depending upon user interest and availability of SDS resources.

Here we present a demonstration of a prototype gratings lightcurve tool, developed as part of the SDS internal review of analysis threads/scenarios. A documented script-version of the tools will be available on the web within a few months. (Further demonstrations available from David Huenemoeder, `dph@space.mit.edu`.)

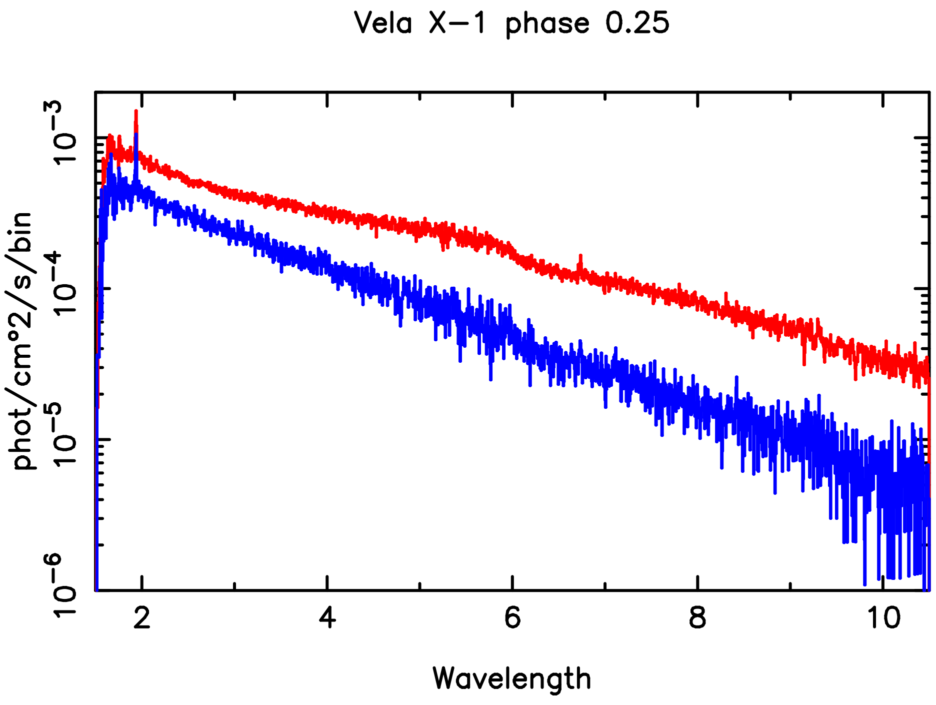
```
sherpa> () = evalfile("cuc_2")

% >>>  Read event file columns into a structure.
% >>>  Two files are required: one unfiltered "evt0" to determine exposure.
% >>>  The other to obtain the grating coordinates.


s = aglc_read_events("evt1a.tg", "evt0.exp");

% Reading event files evt1a.tg, evt0.exp ...
% Read 1190671 events.
% Read 1893566 events.

% >>>  Make light curves in bands, for HEG+MEG, +/-1st orders.
% >>>  Data to be binned is in the structure; arguments specify binning,
% >>>  wavelength regions, gratings, and orders.


c = aglc( s, 71, 1.5, 10.5, ["H","M"], [-1,1]);
% Frames per tbin = 42
% time_step = 7.312e+01
% ontime per bin = 7.140e+01
% Computing rates for ccd_id = 5 6 7 8 9
% min time = 5.950000 sec
% max time = 30281.273680 sec

% >>>  Here are the light curves for 1.5-10.5A (black)
% >>>  hplot(  c.time_min,  c.time_max,  c.count_rate, 1 );
```



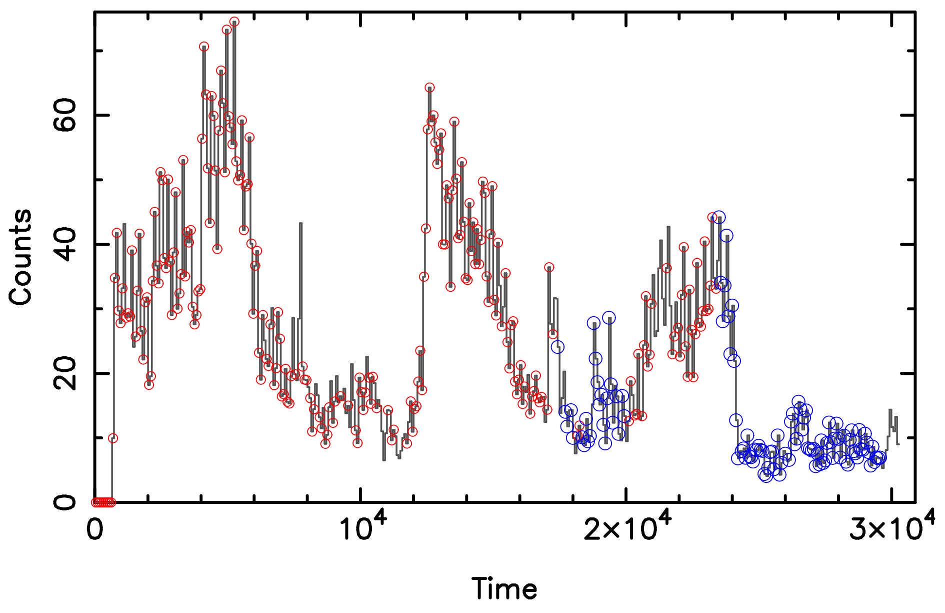
```
c1 = aglc( s, 71, 1.5, 4.5, ["H","M"], [-1,1]);
% Frames per tbin = 42
% time_step = 7.312e+01
% ontime per bin = 7.140e+01
% Computing rates for ccd_id = 6 7 8
% min time = 5.950000 sec
% max time = 30281.273680 sec

% >>>   ....... 1.5-4.5A (red)

c2 = aglc( s, 71, 4.5, 7.5, ["H","M"], [-1,1]);
% Frames per tbin = 42
% time_step = 7.312e+01
% ontime per bin = 7.140e+01
% Computing rates for ccd_id = 5 6 7 8
% min time = 5.950000 sec
% max time = 30281.273680 sec

% >>>  ........ 4.5-7.5A (green)

c3 = aglc( s, 71, 7.5, 10.5, ["H","M"], [-1,1]);
% Frames per tbin = 42
% time_step = 7.312e+01
% ontime per bin = 7.140e+01
% Computing rates for ccd_id = 5 6 7 8 9
% min time = 5.950000 sec
% max time = 30281.273680 sec

% >>>  ........ and 7.5-10.5A (blue).
```



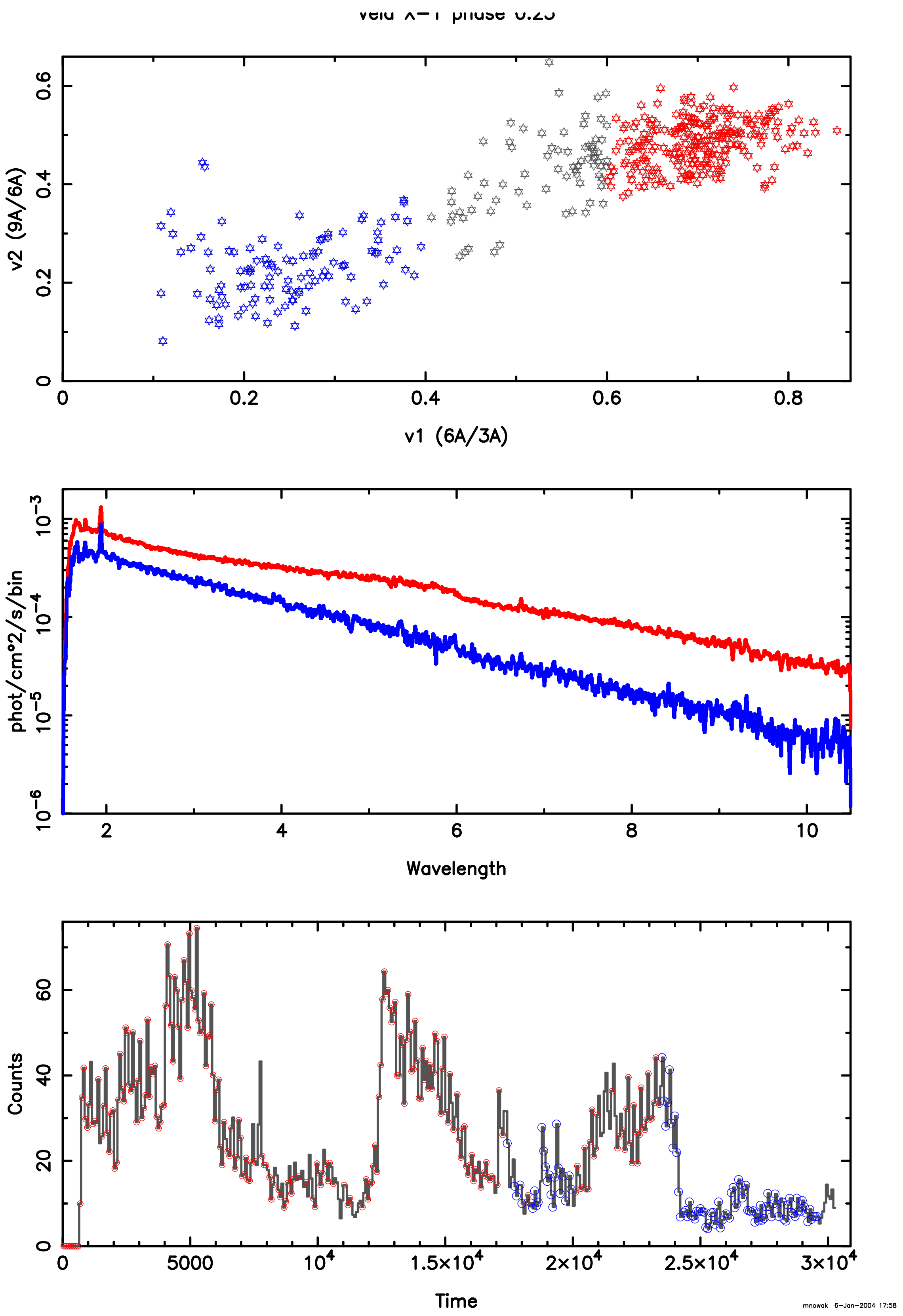
```
% >>>  Ratios of the curves in bands give us a "color-color" plot:


l = where(c1.count_rate>0 and c2.count_rate>0); % for non-zero denominator

v1 = c2.count_rate[l] / c1.count_rate[l];       % 6A/3A  lower=>harder
v2 = c3.count_rate[l] / c2.count_rate[l];       % 9A/6A  lower=>harder

plot(v1,v2);


% >>>  We can select via an expression,
% >>>  or use the gtk-module GUI, "vwhere()"
% >>>  See Mike Noble, mnoble@space.mit.edu,
% >>>  for a vwhere demo, and:
% >>>     http://space.mit.edu/~mnoble/slgtk/


lv_1 = where( v1 > 0.6 );
lv_2 = where( v1 <= 0.4 );
```



```
% >>>  The selection filters are then applied to the event structure, s,
% >>>  using the "reverse indices" from the light curve histogram
% >>>  stored in the light curve structure, c.



s_v1 = aglc_filter( c, s, lv_1 );   % filter s given c and lv_1.
s_v2 = aglc_filter( c, s, lv_2 );   % (large list --- wait for it...)


% >>>  Loading, interpolating,  and summing ARFs, binning spectra...


% >>>  Given the filtered data, we can now bin spectra from the events,
% >>>  and load and sum arfs to compute flux from the hard and soft states.

s1 = histogram( s_v1.wave, x1, x2 );  % bin counts
e1 = sum( c.exposure[lv_1] );         % compute exposure
f1 = s1 / atot / e1;                  % compute flux/bin from counts
```



```
% >>>  We can also look at where the hard and soft selections lie
% >>>  in the light curve itself.  We dynamically filter the
% >>>  light curve array with the selections made on the color-color ratios.


hplot(c.time_min, c.time_max, c.count_rate, 14 );  % gray line
oplot(c.time[lv_1], c.count_rate[lv_1], 2 );       % red dots
oplot(c.time_min[lv_2], c.count_rate[lv_2], 4 );   % blue dots


% >>>  Plot all together........


% >>>  Writing hardcopy to /tmp/VX1_demo_1.ps....
```


Vela X-1 phase 0.25
v2 (9A/6A)
v1 (6A/3A)
phot/cm^2/s/bin
Wavelength
Counts
Time
mnowak 6-Jan-2004 17:58