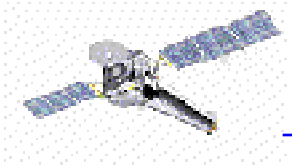


S-Lang in CIAO 2.3

- Enter at the sherpa and chips prompts (or included in scripts):

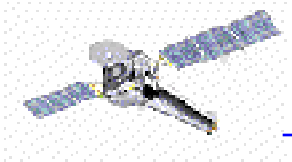
chips> apropos("read")

- If the optional OTS package is installed (as it is in /soft/ciao) then you can also use the slsh program to execute S-Lang scripts

unix% $ASCDS_INSTALL/ots/slang.v1.4.4/slsh/slsh foo.sl

- help

http://cxc.harvard.edu/ciao/threads/slang.html
ahelp slang
ahelp slang-tips [http://cxc.harvard.edu/ciao/download/scripts/]
apropos("foo")
help("command") [after import("isis");]



CIAO 2.3 importable modules:

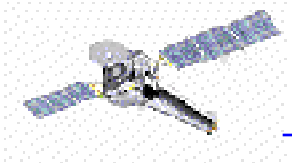
- chips - the "work horse", in that it also includes varmm
- varmmrl
- guide
- isis

Useful commands:
readfile(), curve(), chips_eval(), sherpa_eval(), apropos()

CIAO 3.0 importable modules:

varmm, varmmrl, chips, sherpa, guide, isis
caldb, group, paramio, pixlib, region, stackio, xpa

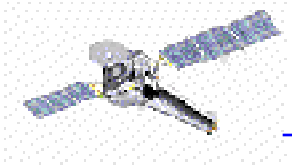
and slsh is available in $ASCDS_INSTALL/bin/



CALDB example

Find the FEF for a file (whose name is in the variable infile):

```
variable cal = calCreateInfo(infile);
if ( cal == NULL ) ... there's a problem ...
calSetData( cal, "FEF_PHA" );

variable status = fits_key_exists( infile, "CTI_CORR" );
variable expr = "cti_corr.eq.";
if ( andelse { status } { fits_read_key( infile, "CTI_CORR" ) } )
      expr += "yes";
else
      expr += "no";
calSetExpression( cal, expr );

variable feffile = calFindFile( cal );
if ( calGetError() != 0 ) ... there's a problem ...
```



Things to know in CIAO 2.3

- ChIPS and Sherpa can be used as calculators due to S-Lang:

```
chips> 23.0 * sin(PI/4.0)
16.2635
```

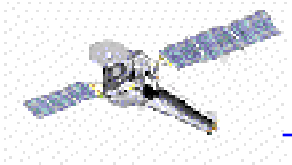
- S-Lang can save you typing:

```
chips> define c (x,y) { () = curve(x,y); }
chips> x = [0:10]; y = 2.3*x^3 - 4.9*x^2 + x;
chips> c(x,y)
```

- Customisations can be placed in one of three files

~/.varmmrc → S-Lang only

~/.chipsrc
~/.sherparc → One-line S-Lang commands and ChIPS/Sherpa



unix% cat ~/.chipsrc

```
% lazy
define q () { () = chips_eval("quit"); } % can be called without the ()
define mean (x) { return sum(x) * 1.0 / length(x); } % * 1.0 to make 'real'
```

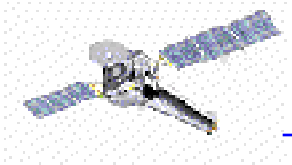
unix% cat ~/.sherparc

```
variable home = getenv("HOME");
variable _test = getenv("SHERPA_TESTING");
if ( _test == NULL ) { () = evalfile( home + "/.sherpasl" ); } else { message(
"** Testing Sherpa - >>NO<< files loaded by .sherparc" ); }
```

- How to load S-Lang scripts into ChIPS & Sherpa

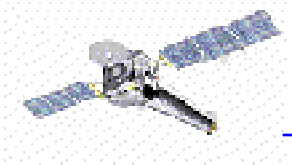
```
unix% chips --slscript foo.sl
chips> () = evalfile("foo.sl");
```

These treat the files as pure S-Lang, so you cannot include Sherpa/ ChIPS commands in them (cf `chips foo.chp').



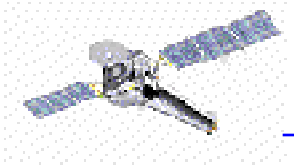
Displaying a column from a file:

```
define ce(str) { () = chips_eval(str); }
variable wgt = readfile( filename );
if ( wgt == NULL ) ... handle error ...
variable old_chips = @chips;
set_state_defaults( "chips" );
chips.symbolstyle = _chips->block;
ce( "redraw off" );
ce( "clear" );
() = curve( wgt.REGNUM, wgt.CONTRIB );
ce( "ylabel CONTRIB" );
ce( "split 2" );
cd( "d 1 tickvals x off" );
ce( "d 2" );
() = curve( wgt.REGNUM, wgt.FRACTION );
ce( "ylabel FRACTION" );
ce( "xlabel REGNUM" );
variable title;
( title, ) = strreplace( filename, "_", "\\_", strlen(filename) );
ce( "title 'weights file: " + title + "'" );
ce( "title size 1.2" );
ce( "redraw on" );
set_state( "chips", old_chips );
```

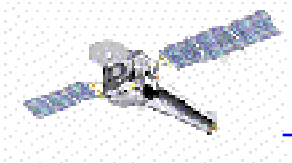


Redefine a Sherpa plot

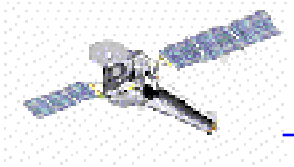
```
private define seval( command ) {
  variable retval = sherpa_eval( command );
  if ( retval != 0 ) error( "\nError: unable to execute the following Sherpa command\n" + command + "\n" );
} % seval()

private define ceval( command ) {
  variable retval = chips_eval( command );
  if ( retval != 0 ) error( "\nError: unable to execute the following ChIPS command\n" + command + "\n" );
} % ceval()

variable _mn;
static define _plot_fit_and () {
  variable usage_str = "Usage: _plot_fit_and( type, n )\n";
  if ( _NARGS != 2 ) { __pop_args( _NARGS ); message(usage_str); return 0; }
  variable type, n;
  ( type, n ) = ();

  _mn = NULL;
  if ( is_defined("get_modelname") == 2 ) {
    eval( "_mn = get_modelname(" + string(n) + ");" );
    if ( _mn == NULL ) { message( "\nError: no model defined for dataset #" + string(n) + "\n" ); return 0; }
  }

  seval( "lplot 2 fit " + string(n) + " " + type + " " + string(n) );
```



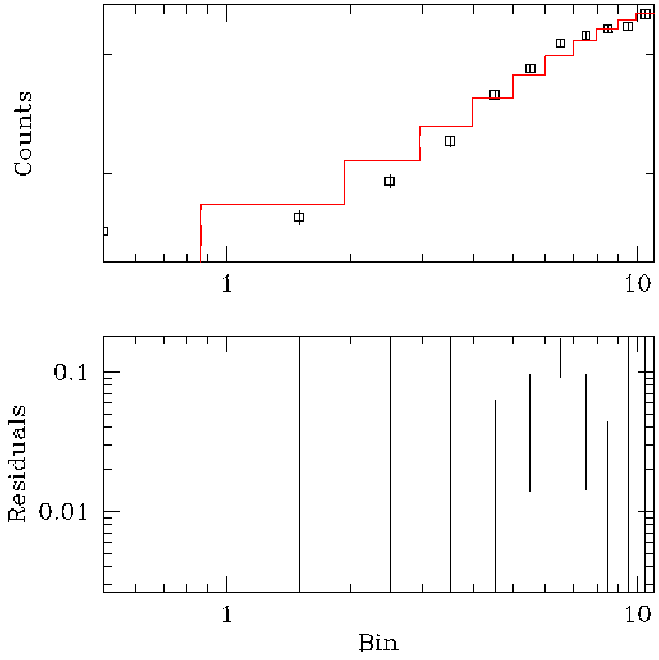

```
  ceval( "d 1 location 0.15 0.9 0.4 0.9" );
  ceval( "d 2 location 0.15 0.9 0.1 0.4" );
  ceval( "d 1 tickvals x off" );
  ceval( "d 2 linear y" );
  ceval( "c 1 symbol bigpoint" );
  ceval( "symbol size 1" );

  variable titlestring;
  if ( _mn != NULL ) { titlestring = "Model = " + _mn; } else { titlestring = time; }
  ceval( "title '" + titlestring + "'" );
  %%ceval( "redraw" );
  return 1;
} % _plot_fit_and()

static define _get_dataset_num () {
  variable n, funcname;
  switch ( _NARGS )
  { case 0: error("Internal error: _get_dataset_num() called with no arguments!!!"); }
  { case 1: n = 1; funcname = (); }
  { case 2: ( funcname, n ) = (); }
  { variable args = __pop_args( _NARGS ); funcname = args[0].value; n = 0; }

  variable usage_str = "Usage: " + funcname + "( [n] )\nIf supplied, n must be >= 1.";
  if ( _NARGS > 2 or n < 1 ) { message(usage_str); return NULL; }
  return n;
} % _get_dataset_num()
```



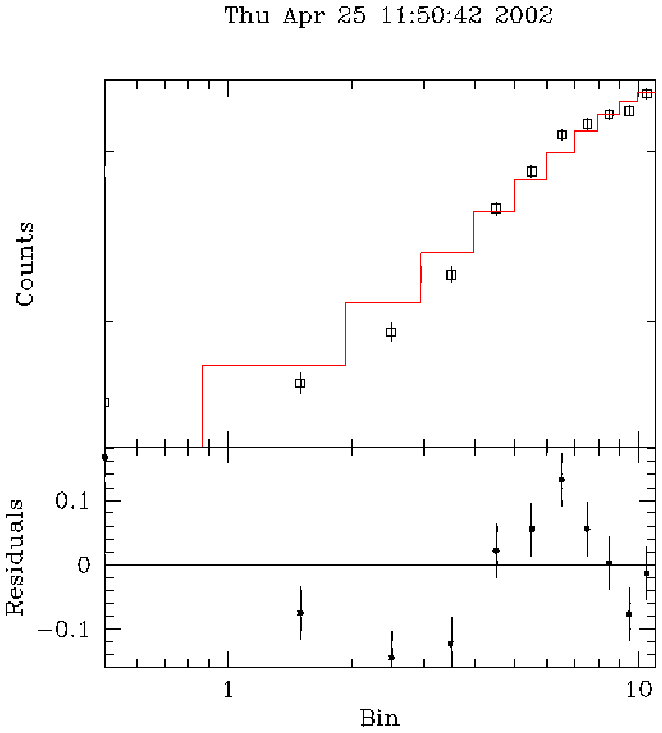

```
% public routines

public define lpfr () {
  variable args = __pop_args (_NARGS);
  variable n = _get_dataset_num( _function_name, __push_args(args) );
  if ( n == NULL ) return;
  variable retval = _plot_fit_and( "resid", n );
  if ( retval == 1 ) ceval( "redraw" );
} % lpfr()

public define lpfd () {
  variable args = __pop_args (_NARGS);
  variable n = _get_dataset_num( _function_name, __push_args(args) );
  if ( n == NULL ) return;
  variable retval = _plot_fit_and( "delchi", n );
  if ( retval == 1 ) ceval( "redraw" );
} % lpfd()
```

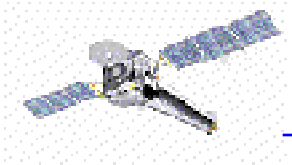


Before



After

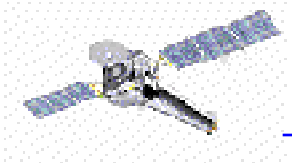




Reading a Sherpa model into S-Lang:

This will be obsolete in CIAO 3.0!

```
public define tmpname (id) {
  variable tmpdir = getenv( "ASCDS_TMP" );
  if ( tmpdir == NULL ) tmpdir = "/tmp";
  variable tmphead = tmpdir + "/" + id + "." + string(getpid()) + ".";
  variable limit = _tmpnum + 1000;
  while ( _tmpnum < limit ) {
    variable file = tmphead + string(_tmpnum) + ".tmp";
    _tmpnum++;
    if ( stat_file(file) == NULL ) return file;
  }
  print( "Oops, I did it again. Unable to generate a temporary file name." );
} % tmpname()

% get hold of the source
% - assume that the variable modelnum isthe source number to read
% - write out to a temp file and then read back in
% - source is stored as <E (keV)> <model (photon/cm^2/s)> where E is the mid-point of the bin
% - would be better to write out as FITS (smaller files) but varmm (CIAO 2.3) has a limit on the
%   number of FITS files that can be opened (128), so we use ascii instead
%
variable filename = tmpname("model");
```




```
% This is the important part
%
() = sherpa_eval( "write source " + string(modelnum) + " " + filename );
variable src = readascii( filename );
if ( src == NULL ) ... handle error ...
() = remove( filename );

% We could finish now, but let's massage the format a tiny bit
%
variable encol = src.col1;
variable othercol = src.col2;

% calculate the lower and upper limits of each bin
variable xlo, xhi;
variable dx = encol * 0.0;
dx[[0:-2]]  = encol[[1:-1]] - encol[[0:-2]];
dx[-1]      = encol[-1] - encol[-2];
dx *= 0.5;
xlo = encol - dx;
xhi = encol + dx;

% create the model
variable model = struct { elo, ehi, flux };
model.elo  = typecast( xlo, Float_Type );
model.ehi  = typecast( xhi, Float_Type );
model.flux = typecast( othercol, Float_Type );
```