The CIAO Contributed Scripts
(and a brief introduction to CIAO)

Jonathan McDowell
& The CIAO Scripts Team

Part 1: a brief introduction to CIAO

What is CIAO?

CIAO is a free, open-source software package developed at the Chandra X-ray Center. CIAO stands for “Chandra Interactive Analysis of Observations”

- It is a collection of programs (*tools*, *applications, scripts, Python modules*).
- Generally run from the command line, although some pieces can be run from GUIs.

CIAO 4.3 is supported on

- Linux 32bit & 64bit
- Mac OS X 10.5 & 10.6 Intel 32bit & 64bit
- Solaris 10

Use “ciaover -v” within CIAO to report the platform information and which version of CIAO is installed:

```
unix% ciaover -v
The current environment is configured for:
  CIAO          : CIAO 4.3 Monday, April 18, 2011
  Tools         : Package release 2  Monday, April 18, 2011
  Sherpa        : Package release 1  Thursday, December 2, 2010
  Chips         : Package release 1  Thursday, December 2, 2010
  Prism         : Package release 1  Thursday, December 2, 2010
  Obsvis        : Package release 1  Thursday, December 2, 2010
  Core          : Package release 1  Thursday, December 2, 2010
  Graphics      : Package release 1  Thursday, December 2, 2010
  bindir        : /Users/user/Documents/ciao-4.3/bin
  Python path : CIAO

CIAO Installation: osx64
System information:
Darwin dhcp-131-142-152-156.cfa.harvard.edu 10.8.0 Darwin Kernel Version
10.8.0: Tue Jun  7 16:33:36 PDT 2011; root:xnu-1504.15.3~1/RELEASE_I386 i386
```

File Formats

- Chandra data is stored in FITS format. ASCII (text) files can now be handled by all tools and applications through the new software library known as the "ASCII kernel"
- When CIAO operates on data it stores processing state/information along with data (keywords, subspace).
- A single file can contain multiple "datasets" (e.g. data, GTI, weight map, regions) stored in "blocks".
- Blocks can contain image or table data. Table columns can be vectors.
- dmlist (a command line tool) or prism (a GUI) are available to view file contents.

Parameter Files (`ahelp parameter`)

The parameters for CIAO tools can be set on the command line or, as with IRAF and FTOOLS, using parameter files. Parameters files are ASCII format and are named `<tool>.par` (e.g. dmcopy.par).

Local copies of the parameter files are stored in $HOME/cxcds_param4. Always delete or rename the local files when upgrading CIAO to a new version.

```
unix% plist dmlist
Parameters for /home/user/cxcds_param4/dmlist.par
       infile = acisf04192N004_evt2.fits[cols ccd_id] Input dataset/block
specification
          opt = data           Option
     (outfile = )              Output file (optional)
        (rows = )              Range of table rows to print (min:max)
       (cells = )              Range of array indices to print (min:max)
     (verbose = 0)             Debug Level(0-5)
        (mode = ql)
```

There are number of CIAO tools (e.g. plist, pset, punlearn) that are used to read and write to the parameter files.

Filters, Regions, and GTIs

- filtering (removal of unwanted events) is an essential part of X-ray analysis - e.g. to remove periods of high background or poor aspect solution, exclude uninteresting sources from an image.
- the DataModel (DM) provides great filtering flexibility: e.g. dmstat “evt2.fits [EVENTS][energy>300][cols-grade]” (see `ahelp filtering,ahelp dmimgfiltering`)
- GTIs (Good Time Intervals) are used to define what times periods of the observation can be used (i.e. contain valid data). They are generally stored as a block in the event list (see `ahelp chandra times`)
- Regions are used to define the source and background areas of an image. They are text files that can be created manually or within ds9, and are used as a filter (e.g. “[sky=region (source.reg)]”). (see `ahelp dmregions`)
- Subspace records the filters applied to a file; dmlist can read this history using opt=subspace (see `ahelp subspace`)

Scripting Languages

- The main scripting (or interpreted = no compilation is necessary) language supported in CIAO via Sherpa and ChIPS is Python.
- Sherpa and ChIPS are importable modules for Python
- You **do not** need to know Python to use Sherpa and Chips, but if you do, you will be able to use its capabilities in your analysis (see Doug's talk on "Scripting in CIAO")

The second half of this talk focuses on the Python scripts that we have written to automate common analysis tasks. These scripts are available to users in the contributed scripts tarfile, which is part of a standard CIAO installation.

CIAO Overview

Data manipulation: copy, filter, extraction, stats, etc.

Data preparation (or Chandra-specific instrument tools): update calibration, correct for instrumental effects, find & extract grating data, create aspect histograms

Response tools: exposure map, PSF, RMF and ARF

Source Detection: celldetect, wavdetect, vtpdetect

Timing & Background tools: lightcurve, power spectrum, barycenter correction

Convolutions, Transforms, & Smoothing: csmooth, aconvolve, acrosscorr, apowerspectrum

Plotting: ChIPS (*)

Modeling/Fitting: Sherpa (*)

GUIs: DS9, prism,peg

(*) powerful data manipulation and scripting capabilities are now possible in these applications through the Python interpreted language.

The Data Model and the Data Manipulation Tools

- The CXC analysis and processing software is built on a common versatile interface library called the CXC Data Model (or just DM).
- The DM provides users with a powerful built-in data filtering and binning capability.
- The DM includes an "ASCII kernel" which gives the ability to operate on certain ASCII (text) file the same way as on FITS files (e.g. for filtering, plotting etc.)
- Any program that asks for a data file name as input accepts a *"virtual file"* string which allows the program to see a filtered version of the file in question.
- All columns of event lists are treated "equally": binning is allowed not only in spatial coordinates but also in time, energy coordinates, for example, giving the ability to create multidimensional images in space-energy, or space-time, etc.

Data Manipulation Tools

The four “core” DM tools are:

dmlist: list contents or structure of a file

dmcopy: filter and bin tables and images

dmextract: make a histogram table file (e.g. PHA file, lightcurve file) from a table column. Generate count histogram on supplied regions for a spatial table or image file.

dmgti: create custom Good Time Intervals (GTIs) from a constraint expression

There are over 40 “DM” tools in CIAO.

dmap dmpend dmellipse dmhedit dmimgfilt dmimgreproject dmmakereg dmreadpar dmtype2split dmarfadd dmextract dmhistory dmimghist dmimgthresh dmmaskbin dmregrid dmcontour dmfilth dmimg2jpg dmimglasso dmjoin dmmaskfill dmregrid2 dmcoords dmgroup dmimgadapt dmimgpick dmkeypar dmmerge dmsort dmcopy dmgroupreg dmimgblob dmimgpm dmlist dmnautilus dmstat dmdiff dmgti dmimgcalc dmimgproject dmmakepar dmpaste dmtcalc

Data Model Syntax (`ahelp dmsyntax`)

In the DM context, a "virtual file" in represented by a filename followed by a series of optional qualifiers in square brackets [ ]:

"filename[block][filter][columns/binning][options][rename]"

block - is the "section" of the file to use

filter- is the filter to be applied

columns/binning - specifies either the columns from a table to be included in an output table or the binning. When binning the data to generate an n- dimensional image, the range and binsize (min:max:bin) must be specified.

options - a sequence describing special options for the DM library

rename - specifies a name for the new block

Note that:

- the order of the qualifiers generally matters
- it isn't necessary to always specify all qualifiers

Simple examples of “virtual files”

Select the first three columns of the EVENTS block by number:

```
acisf01843N001_evt2.fits[EVENTS]
[time=84245787:84247000] [cols #1,#2,#3]
```

or by name:

```
acisf01843N001_evt2.fits[EVENTS][grade=0,2,3][cols
time,ccd_id,node_id]
```

after filtering in time or grade

Bin an events file to create a PI spectrum for a specified region
(input for dmextract): `acisf01843N001_evt2.fits[EVENTS]`
`[sky=region(mysrc.reg)][bin pi=1:1024:1]`

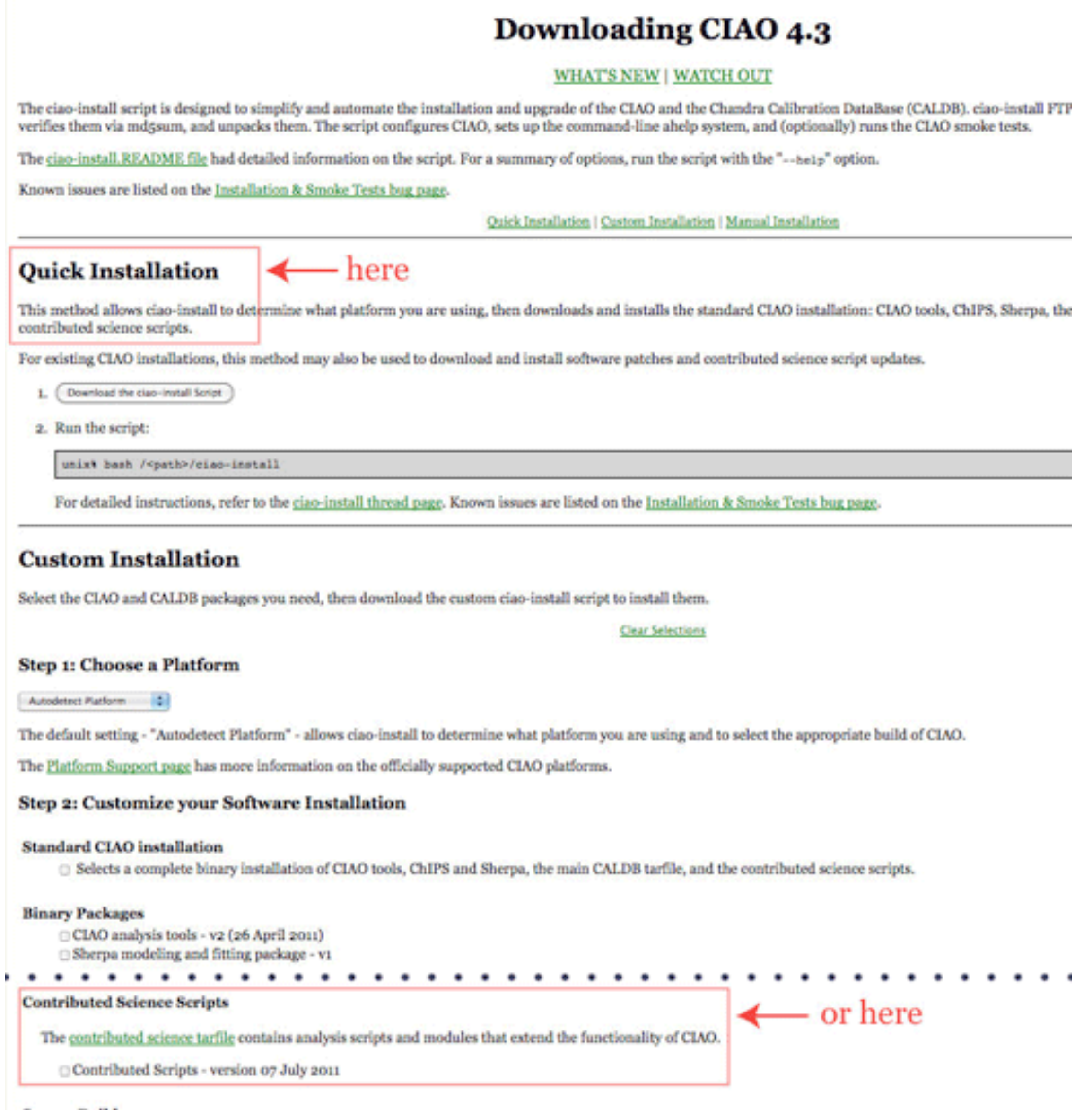
or an image (input for dmcopy): `acisf01843N001_evt2.fits`
`[x=3600:4000,y=3800:4200]" acis_center.fits`

Part 2: the CIAO Contributed Scripts

Installing the Contributed Tarfile

The contributed tarfile is included in the "Quick Installation" option on the CIAO download page. It may also be selected as part of the "Custom Installation" process.

The scripts depend on the components of a "standard" CIAO installation: tools, ChIPS, Sherpa, and the CALDB. If you have installed a customized version of CIAO, it is possible that some of the scripts and modules will fail due to a missing package.



Downloading CIAO 4.3

WHAT'S NEW | WATCH OUT

The ciao-install script is designed to simplify and automate the installation and upgrade of the CIAO and the Chandra Calibration DataBase (CALDB). ciao-install FTP
verifies them via md5sum, and unpacks them. The script configures CIAO, sets up the command-line ahelp system, and (optionally) runs the CIAO smoke tests.

The ciao-install.README file had detailed information on the script. For a summary of options, run the script with the "--help" option.

Known issues are listed on the Installation & Smoke Tests bug page.

Quick Installation | Custom Installation | Manual Installation

Quick Installation

This method allows ciao-install to determine what platform you are using, then downloads and installs the standard CIAO installation: CIAO tools, ChIPS, Sherpa, the
contributed science scripts.

For existing CIAO installations, this method may also be used to download and install software patches and contributed science script updates.

1. Download the ciao-install Script
2. Run the script:

```
unix% bash /<path>/ciao-install
```

For detailed instructions, refer to the ciao-install thread page. Known issues are listed on the Installation & Smoke Tests bug page.

Custom Installation

Select the CIAO and CALDB packages you need, then download the custom ciao-install script to install them.

Clear Selections

Step 1: Choose a Platform

Autodetect Platform

The default setting - "Autodetect Platform" - allows ciao-install to determine what platform you are using and to select the appropriate build of CIAO.

The Platform Support page has more information on the officially supported CIAO platforms.

Step 2: Customize your Software Installation

Standard CIAO installation

☐ Selects a complete binary installation of CIAO tools, ChIPS and Sherpa, the main CALDB tarfile, and the contributed science scripts.

Binary Packages

☐ CIAO analysis tools - v2 (26 April 2011)
☐ Sherpa modeling and fitting package - v1

Contributed Science Scripts

The contributed science tarfile contains analysis scripts and modules that extend the functionality of CIAO.

☐ Contributed Scripts - version 07 July 2011

How to reduce your Chandra data in four easy steps:

```
unix% download_chandra_obsid 198
```

```
unix% cd 198
unix% chandra_repro ./ out=repro
```

```
unix% fluximage repro_evt2.fits out=198 bin=2 bands=CSC
```

```
unix% specextract "repro_evt2.fits[sky=circle(4088,3982,10)]"
bkgfile="repro_evt2.fits[sky=circle(3880,3365,20)]" out=198
```

How to reduce your Chandra data in four easy steps:

1. download the data

```
unix% download_chandra_obsid 198
```

2. reprocess the dataset

```
unix% cd 198
unix% chandra_repro ./ out=repro
```

3. create a three-color fluxed image

```
unix% fluximage repro_evt2.fits out=198 bin=2 bands=CSC
```

4. extract the spectrum and response files for fitting

```
unix% specextract "repro_evt2.fits[sky=circle(4088,3982,10)]"
bkgfile="repro_evt2.fits[sky=circle(3880,3365,20)]" out=198
```

* The event file is actually named `acisf00198_001N003_repro_evt2.fits`

download_chandra_obsid

Download public data by ObsId from the Chandra archive:

```
unix% download_chandra_obsid 198
```

Specify multiple observations as a comma-separated list:

```
unix% download_chandra_obsid 198,1838,459
```

Each of these commands will download all the archived data files for the given ObsIDs. An optional list of filetypes can be included to limit the download:

```
unix% download_chandra_obsid 198 evt2,asol,bpix1
```

The supported filetypes are:

```
 aoff aqual asol bias bpix cntr_img dtf eph0 eph1 evt1 evt2
 flt fov full_img msk mtl oif osol pbk pha2 plt soff src2
 src_img stat sum vv
```

download_chandra_obsid

The status of the download is printed to the screen. The -q or --quiet flags can be used to turn off screen output.

```
unix% download_chandra_obsid 198 evt2,asol,bpix1
 Downloading files for ObsId 198, total size is 27 Mb.

   Type     Format      Size  0........H.........1  Download Time Average Rate
   ---------------------------------------------------------------------------
   evt2     fits       25 Mb  ####################            7 s  3480.5 kb/s
   bpix     fits       10 Kb  ####################          < 1 s  187.0 kb/s
   asol     fits        1 Mb  ####################          < 1 s  3962.5 kb/s

       Total download size for ObsId 198 = 27 Mb
       Total download time for ObsId 198 = 8 s


unix% ls -1 198/*
acisf00198_001N003_bpix1.fits.gz
acisf00198N003_evt2.fits.gz
pcadf075303382N003_asol1.fits.gz
```

download_chandra_obsid

This script doesn’t have a parameter file because of its basic functionality, but there are several command-line flags. Running without arguments provides use information:

```
Usage: download_chandra_obsid <obsid1>,..,<obsidN> [<type1>,..,<typeN>]

 Download public Chandra observations.  The observations to download are given
as a comma-separated list of ObsId numbers, and an optional comma-separated
list of file "types" can also be given, which will only download files that
contain the type strings. So an argument of '9123,9124' will download all data
for the obs ids 9123 and 9124, whereas '9123,9124 fov,vv,evt2' will only
download the V&V, fov, and evt2 files for these observations.  The data is
written to the current directory, with each obsid being saved to its own
directory (following the layout used by the Chandra archive).

Options:
  -h, --help        show this help message and exit
  -q, --quiet       Download the files without any screen output? [default:
                    False]
  -v, --version     List the version of the script and exit.
  -c, --copyright   List the copyright for the script and exit.
  -t, --filetypes   List the valid file types and exit.
  -d, --debug       Display diagnostic output? [default: False]
```

download_chandra_obsid

Limitations and Special Cases:

- the script cannot download proprietary data, since it is password-protected in the Archive.
- you have to know the ObsID for the dataset. This may be found via WebChaser, from the target search form, or in literature references.

chandra_repro

Reprocess any ACIS and HRC imaging or grating data to apply the newest calibration and the filtering steps recommended in the CIAO analysis threads.

The required input is the directory which contains the data files (e.g. created by download_chandra_obsid). In this example, the input files are in the working directory; the output directory is called “repro” (the default):

```
unix% chandra_repro ./ out=repro
```

The script reads data from the standard data distribution (e.g. primary and secondary directories) and creates a new bad pixel file, a new level=2 event file, and a new level=2 Type II PHA file (grating data only).

chandra_repro

```
 unix% chandra_repro 198 out=repro

Running chandra_repro
version: 28 April 2011
...
Running acis_process_events to reprocess the evt1...
Running destreak tool on evt1...
Filtering evt1 by grade and status...
Applying the good time intervals from the flt1 file...
The new evt2 file is: /198/repro/acisf00198_001N003_repro_evt2.fits

Updating event file header with chandra_repro HISTORY record

Setting user ardlib for observation-specific bad pixel file

Cleaning up intermediate files

WARNING: Observation-specific bad pixel file set for session use:
         /198/repro/acisf00198_001N003_bpix1.fits
         Run 'punlearn ardlib' when analysis of this dataset completed.

The data have been reprocessed.
Start your analysis with the new products in
/198/repro
```

chandra_repro

```
Parameters for /soft/ciao-4.3/contrib/param/chandra_repro.par

         indir = ./                 Input directory
        outdir = ./repro            Output directory
         (root = )                  Root for output filenames
     (badpixel = yes)               Create a new bad pixel file?
(process_events = yes)              Create a new level=2 event file?
     (destreak = yes)               Destreak the ACIS-8 chip?
   (set_ardlib = yes)               Set ardlib.par with the bad pixel
file?
  (check_vf_pha = yes)              Clean ACIS background in VFAINT
data?
      (pix_adj = default)           Pixel randomization: default|edser|
none|randomize
      (cleanup = yes)               Cleanup intermediate files on exit
      (clobber = no)                Clobber existing file
      (verbose = 1)                 Debug Level(0-5)
         (mode = ql)
```

chandra_repro

Limitations and Special Cases:

- special input cases: data taken in ACIS interleaved (alternating exposure) mode or that have multiple observation intervals (OBIs) will have more than one level=1 event file (`evt1.fits`) in the data distribution.

  The input data has to be separated into different input directories before running the script.

- continuous-clocking mode data: the script assumes that the source coordinates in the header are accurate (RA_TARG,DEC_TARG). They might need to be adjusted before reprocessing the data.

fluximage

Create exposure-corrected images of an ACIS or HRC observation, given an events file for one or more energy bands.

There are four required inputs: event file, output root name, binning, and energy bands.

```
unix% fluximage repro_evt2.fits out=198 bin=2 bands=CSC
```

The binning factor specifies how large the output images will be.

The script has predefined energy bands taken from the Chandra Source Catalog: ultrasoft, soft, medium, hard, broad, wide, CSC (soft, medium, & hard). Bands can also be specified as “low:high:eff” in keV, where eff is the effective energy for the instrument map.

fluximage

```
unix% fluximage repro/repro_evt2.fits out=198 bin=2 bands=CSC

Running fluximage...
Version: 01 July 2011

Using CSC ACIS science energy bands.
...
Creating aspect histogram(s).
Creating binned images.

Creating instrument maps in parallel.

Creating exposure maps in parallel.
Exposure map limits: 0.000000e+00, 1.481394e+06
Writing exposure map to 198_7_0.92_bin2.expmap
Exposure map limits: 0.000000e+00, 1.045427e+06
Writing exposure map to 198_7_3.8_bin2.expmap
Exposure map limits: 0.000000e+00, 1.622567e+06
Writing exposure map to 198_7_1.56_bin2.expmap

Exposure-correcting images in parallel.

Cleaning up intermediary files.
```

(continued)

fluximage

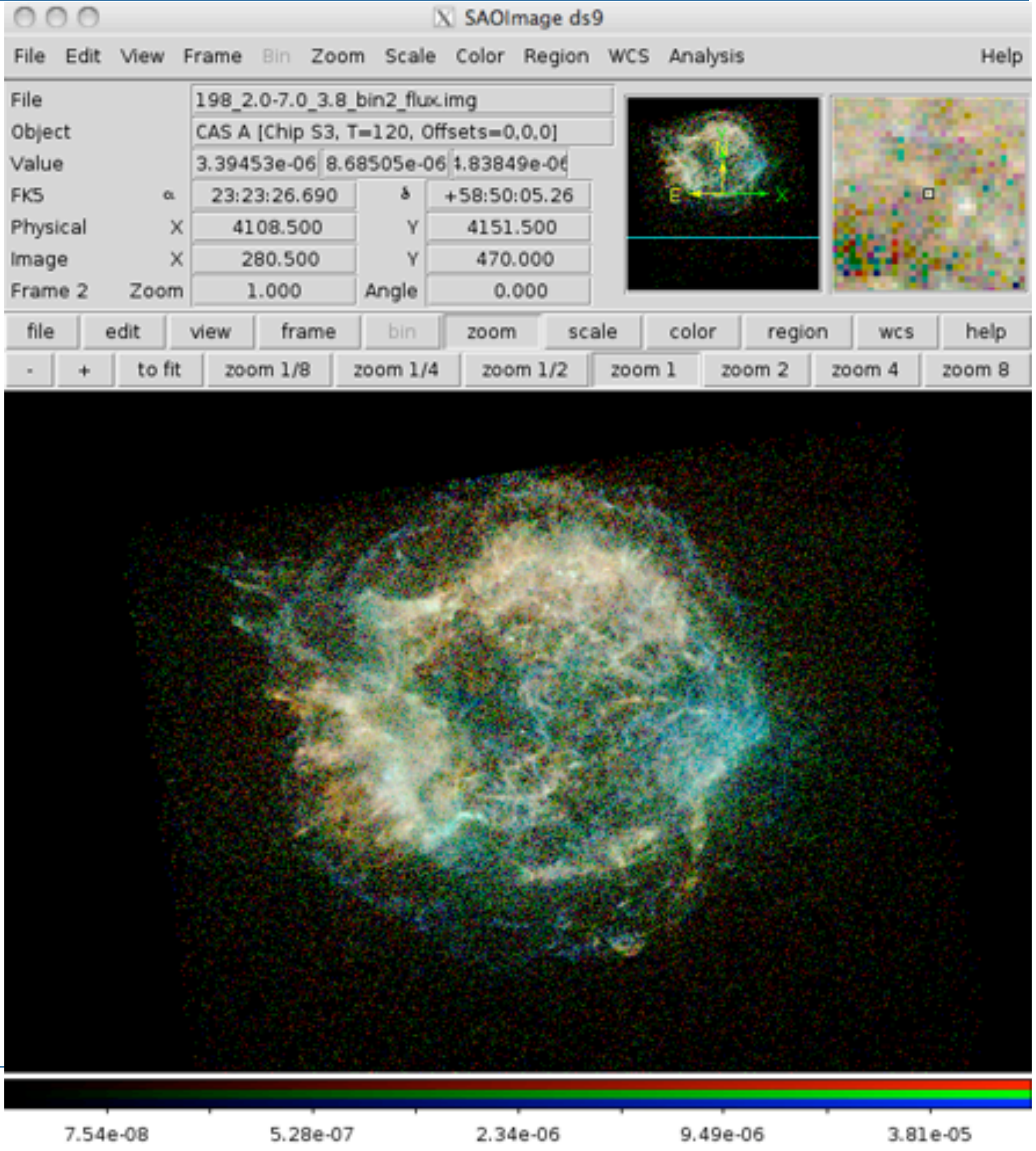
(output continued)

```
The following files were created:

 The binned counts images are:
     198_0.5-1.2_bin2.img
     198_1.2-2.0_bin2.img
     198_2.0-7.0_bin2.img

 The clipped counts images are:
     198_0.5-1.2_bin2_thresh.img
     198_1.2-2.0_bin2_thresh.img
     198_2.0-7.0_bin2_thresh.img

 The exposure maps are:
     198_0.92_bin2.expmap
     198_1.56_bin2.expmap
     198_3.8_bin2.expmap

 The exposure-corrected images are:
     198_0.5-1.2_0.92_bin2_flux.img
     198_1.2-2.0_1.56_bin2_flux.img
     198_2.0-7.0_3.8_bin2_flux.img
```

fluximage

The three-color image displayed in ds9.

The image display may be improved by smoothing the data in CIAO (csmooth, aconvolve) or interactively via the ds9 Analysis menu.



SAOImage ds9

File 198_2.0-7.0_3.8_bin2_flux.img
Object CAS A [Chip S3, T=120, Offsets=0,0,0]
Value 3.39453e-06 8.68505e-06 1.83849e-06
FK5 α 23:23:26.690 δ +58:50:05.26
Physical X 4108.500 Y 4151.500
Image X 280.500 Y 470.000
Frame 2 Zoom 1.000 Angle 0.000

fluximage

```
Parameters for /soft/ciao-4.3/contrib/param/fluximage.par

        infile =                  Input events file
       outroot =                  Output root of data products?
       binsize = 8                Image binning factor
         bands = broad            Energy bands, comma-separated list,
min:max:center in keV or ultrasoft, soft, medium, hard, broad, wide, CSC
     (asolfile = )                Input aspect solutions
   (badpixfile = )                Input bad pixel file
     (maskfile = )                Input mask file
      (pbkfile = )                Input pbk file for ACIS observations
      (dtffile = )                Input dtf file for HRC observations
    (normalize = no)              Normalize exposure map?
 (expmapthresh = 1.5%)            Remove low-exposure regions? '2%' excludes
pixels where exposure is < 2% of the maximum
     (parallel = yes)             Run processes in parallel?
       (tmpdir = ${ASCDS_WORK_PATH} -> /tmp) Directory for temporary files
      (cleanup = yes)             Delete intermediaray files?
      (clobber = no)              OK to overwrite existing output file?
      (verbose = 1)               Verbosity level?
         (mode = ql)
```

fluximage

Limitations and Special Cases:

- When processing an HRC-I observation the HRC background CALDB package must be installed, otherwise the script will fail during processing.
- There are a few HRC-I observations taken before 1999-12-06 for which a background file is not available. fluximage cannot yet process this case.
- Running the script on ACIS grating data only uses zeroth- and null-orders (i.e. the non-diffracted events), but HRC uses all orders.

specextract

Extract source and background spectra and the associated ARF and RMF files for point and extended sources, including the zero-order of grating data. Extended source responses are weighted appropriately. The point-source aperture correction may be applied in the unweighted-ARF case.

```
unix% specextract "repro_evt2.fits[sky=circle(4088,3982,10)]"
bkgfile="repro_evt2.fits[sky=circle(3880,3365,20)]" out=198
```

This is a actually a simplified version of the specextract command. The full command for a point source might look like:

```
unix% specextract "repro_evt2.fits[sky=circle(4088,3982,10)]"
bkgfile="repro_evt2.fits[sky=circle(3880,3365,20)]" out=198
weight=no correct=yes asp=repro/pcadf075303382N003_asol1.fits
pbkfile=repro/acisf075304025N003_pbk0.fits mskfile=repro/
acisf00198_001N003_msk1.fits badpixfile=repro/
acisf00198_001N003_bpix1.fits
```

specextract

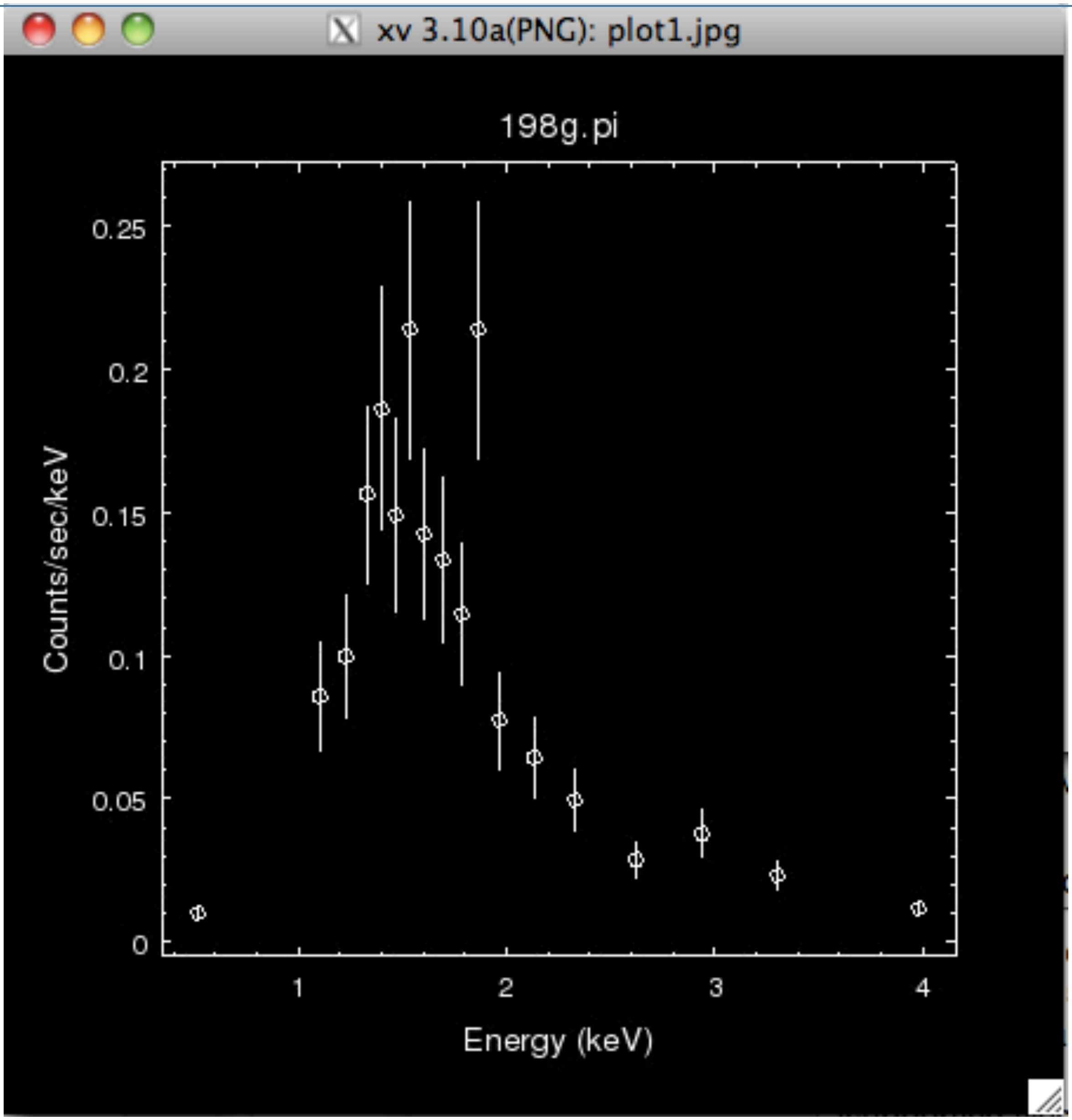
```
unix% specextract "repro_evt2.fits[sky=circle(4088,3982,10)]"
bkgfile="repro_evt2.fits[sky=circle(3880,3365,20)]" out=198

Running: specextract
  Version:    7 July 2011

Checking source input files for readability...
Checking background input files for readability...

Setting bad pixel file for item 1 of 1 in input list

Extracting src spectra for item 1 of 1 in input list
Creating src ARF for item 1 of 1 in input list
Creating src RMF for item 1 of 1 in input list
Using mkacisrmf...
Grouping src spectrum for item 1 of 1 in input list

Updating header of 198.pi with RESPFILE and ANCRFILE keywords.
Updating header of 198_grp.pi with RESPFILE and ANCRFILE keywords.
```

(continued)

specextract

(output continued)

```
Setting bad pixel file for item 1 of 1 in input list

Extracting bkg spectra for item 1 of 1 in input list
Creating bkg ARF for item 1 of 1 in input list
Creating bkg RMF for item 1 of 1 in input list
Using mkacisrmf...

Updating header of 198_bkg.pi with RESPFILE and ANCRFILE keywords.
Updating header of 198.pi with BACKFILE keyword.
Updating header of 198_grp.pi with BACKFILE keyword.


unix% ls
198.arf                 198_bkg.rmf
198_asphist7.fits       198.corr.arf
198_bkg.arf             198_grp.pi
198_bkg_asphist7.fits   198.pi
198_bkg.pi              198.rmf
```


xv 3.10a(PNG): plot1.jpg
198g.pi
Counts/sec/keV
Energy (keV)
0
0.05
0.1
0.15
0.2
0.25
1
2
3
4

specextract

```
Parameters for /soft/ciao/contrib/param/specextract.par

        infile =                      Source event file(s)
       outroot =                      Output directory path + root name for output files
        weight = yes                  Should response files be weighted?
       correct = no                   Apply point source aperture correction to ARF?
       bkgfile =                      Background event file(s)
       bkgresp = yes                  Create background ARF and RMF?
           asp =                      Source aspect solution or histogram file(s)
       combine = no                   Combine output spectra and responses?
       pbkfile =                      pbkfile input to mkwarf
       mskfile =                      mskfile input to mkwarf
      (rmffile = CALDB)               rmffile input for CALDB
        (ptype = PI)                  PI or PHA
    (grouptype = NUM_CTS)             Spectrum grouping type (same as grouptype in dmgroup)
      (binspec = 15)                  Spectrum grouping specification (NONE,1:1024:10,etc)
(bkg_grouptype = NONE)                Background spectrum grouping type (NONE, BIN, SNR,
NUM_BINS, NUM_CTS, or ADAPTIVE)
  (bkg_binspec = )                    Background spectrum grouping specification (NONE,10,etc)
       (energy = 0.3:11.0:0.01)       Energy grid
      (channel = 1:1024:1)            RMF binning attributes
  (energy_wmap = 300:2000)            Energy range for (dmextract) WMAP input to mkacisrmf
      (binwmap = tdet=8)              Binning factor for (dmextract) WMAP input to mkacisrmf
       (dafile = CALDB)               dafile input to mkwarf
    (badpixfile = )                   Bad pixel file for the observation
      (clobber = no)                  OK to overwrite existing output file?
      (verbose = 1)                   Debug Level(0-5)
         (mode = ql)
```

specextract

Limitations and Special Cases:

- The script assumes that the exposure does not vary significantly across a point or extended source region. If the region spans multiple chips, it is assumed that the GTIs of each are similar enough that a single aspect histogram file applies to each.
- The script considers the brightest pixel in a source region to represent the position of the source region. This won't work correctly for saturated sources (e.g. when pileup is an issue).
- Sufficiently large input source extraction regions can cause several of the tools to run very slowly, especially when creating weighted responses.

Bonus script: combine_spectra

Sums multiple imaging source PHA spectra, and optionally, associated background PHA spectra and source and background ARF and RMF instrument responses.

```
unix% combine_spectra src_spectra="10289.pi,10290.pi,
10291.pi,10292.pi,10293.pi" outroot=ngc6300
```

Given a list of source spectra, the script reads the background spectrum and response file names from the header, assuming the necessary keywords are set. Otherwise, all the files may be specified in the parameter file.

If the `combine` parameter in specextract is set to yes, the combine_spectra script is invoked to sum the newly-created spectra and responses.

combine_spectra

```
unix% combine_spectra src_spectra="10289.pi,10290.pi,10291.pi,10292.pi,10293.pi"
outroot=ngc6300

Running: combine_spectra
Version:    30 March 2011

Source PHA files to be combined:
10289.pi
10290.pi
10291.pi
10292.pi
10293.pi

Checking headers of source PHA files for ARF files to combine.
Found ARF file 10289.arf
Found ARF file 10290.arf
Found ARF file 10291.arf
Found ARF file 10292.arf
Found ARF file 10293.arf

Checking headers of source PHA files for RMF files to combine.
Found RMF file 10289.rmf
Found RMF file 10290.rmf
Found RMF file 10291.rmf
Found RMF file 10292.rmf
Found RMF file 10293.rmf
```

(continued)

combine_spectra

(output continued)

```
Checking headers of source PHA files for background files to combine.
Found background file 10289_bg.pi
Found background file 10290_bg.pi
Found background file 10291_bg.pi
Found background file 10292_bg.pi
Found background file 10293_bg.pi

Checking headers of background PHA files for ARF files to combine.
WARNING: The ANCRFILE header keyword value in 10289_bg.pi is 'none'; no ARF files to
combine.
WARNING: The ANCRFILE header keyword value in 10290_bg.pi is 'none'; no ARF files to
combine.
WARNING: The ANCRFILE header keyword value in 10291_bg.pi is 'none'; no ARF files to
combine.
WARNING: The ANCRFILE header keyword value in 10292_bg.pi is 'none'; no ARF files to
combine.
WARNING: The ANCRFILE header keyword value in 10293_bg.pi is 'none'; no ARF files to
combine.

Wrote file ngc6300_src.pi.
Wrote file ngc6300_bkg.pi.
Wrote files ngc6300_src.arf and ngc6300_src.rmf.
```

combine_spectra

```
Parameters for /soft/ciao-4.3/contrib/param/combine_spectra.par

    src_spectra =                 Source PHA files to combine; enter list or '@stack'
        outroot =                 Root name for output files
     (src_arfs = )                Source ARF files to combine; enter list or '@stack'
     (src_rmfs = )                Source RMF files to combine; enter list or '@stack'
  (bkg_spectra = )                Background PHA files to combine; enter list or '@stack'
     (bkg_arfs = )                Background ARF files to combine; enter list or '@stack'
     (bkg_rmfs = )                Background RMF files to combine; enter list or '@stack'
      (clobber = yes)             OK to overwrite existing output file?
      (verbose = 0)               Debug Level(0-5)
         (mode = ql)
```

combine_spectra

Limitations and Special Cases:

- Any grouping flags which may be present in input source or background PHA spectra will be ignored by the script.
- Combining background spectra with wildly varying spectral extraction region areas may yield misleading uncertainty estimates; i.e., some extractions will be over-represented while others will be under-represented.
- If the background rates contributing to a source are significantly different in the individual spectra to be combined, it is recommended that these spectra remain separate and be modeled simultaneously.

  Otherwise, the modeling results of the combined source spectrum could be biased towards the observation(s) with the highest background rate(s).

What next?

aprates

Let's perform aperture photometry on the source in the 0.5-7 keV energy range.

What chip does the source fall on?

```
unix% dmstat "198.evt[sky=region(198.reg),energy=500:7000]
[cols ccd_id]"
ccd_id
    min:          7              @:         1
    max:          7              @:         1
   mean:          7
  sigma:          0
    sum:          3689
   good:          527
   null:          0
```

The source is on chip 7.

aprates

What is the count rate?

```
unix% dmlist "198.evt[sky=region(198.reg),energy=500:7000]" counts
527
unix% dmlist "198.evt[sky=region(bg.reg),energy=500:7000]" counts
61
```

What are the region areas?

```
unix% dmlist "198.evt[sky=region(198.reg),ccd_id=7]" subspace | grep
-i area
   8 sky                   Real4                    Field area = 6.71089e
+07 Region area = 314.159

unix% dmlist "198.evt[sky=region(bg.reg),ccd_id=7]" subspace | grep -
i area
   8 sky                   Real4                    Field area = 6.71089e
+07 Region area = 1256.64
```

My region has radius of 10 pixels, which is ~ 5"

aprates

Where is it in MSC (off-axis-angle) coords? (The PSF size depends on this.)

```
unix%  dmcoords 198.evt 198.asol x=4088 y=3982 option=sky verbose=1 |
grep THETA
 THETA,PHI             56.5"         347.82 deg
```

What is the PSF fraction in my region? Note that 56.5" is 0.94'.

```
 ln -s $CALDB/data/chandra/default/reef/hrmaD1996-12-20reefN0001.fits
psf.data
 python
 > import psf
 > pdata = psf.psfInit("psf.data")
 > f = psf.psfFrac(pdata, 1.0, 0.94, 347.8, 5.0)
 > print(f)
0.97990472442
```

aprates

We can check it at a couple of other energies too

```
> f=psf.psfFrac(pdata,0.5,0.94,347.8,5.0)
> print(f)
0.986305015765

> f=psf.psfFrac(pdata,7.0,0.94,347.8,5.0)
> print(f)
0.926977752619
```

Since most of our counts are near 1 keV, we will adopt f = 0.98 even though that's a bit low for the higher energies

What is the exposure time?

```
unix% dmlist 198.evt keys | grep LIVETIME
0118 LIVETIME                     2483.1153432204 [s]       Real8
Livetime
```

aprates

We can now run aprates:

```
unix% aprates n=527 m=61 A_s=314.1 A_b=1256.6 alpha=0.98
beta=0.0 T_s=2483.11 T_b=2483.11 E_b=1 E_s=1 eng_s=1 eng_b=1
flux_s=1 flux_b=1 conf=0.90 verbose=0 outfile=out2.par

unix% plist out2.par | grep src_rate
(src_rate = 0.210299)
(src_rate_err_lo = 0.194772)
(src_rate_err_up = 0.225921)
(src_rate_conf = 0.900024)
(src_rate_status = 0)
(src_rate_signif = 22.1634)
(src_rate_mode = 0.210299)
```

The estimated count rate is 0.210 +/- 0.016 counts/s at a 90 percent confidence interval.

modelflux

This is a script which runs Sherpa without fitting, to estimate the flux corresponding to a given count rate.

Let's roughly estimate the flux our rate corresponds to, using a simple power law:

```
modelflux arf="198.corr.arf" rmf="198.rmf"
model="xsphabs.abs1*powlaw1d.p"
paramvals="abs1.nh=1.7;p.gamma=0.7" emin=0.5 emax=7.0
rate=0.21
Model fluxes:
Rate (0.5,7)= 0.21 count s^-1
Photon Flux (0.5,7)= 0.00061685 photon cm^-2 s^-1
Energy Flux (0.5,7)= 4.0753e-12 erg cm^-2 s^-1
```

Unfortunately this may be a factor of several off; for comparison a blackbody, which represents this data better.

modelflux

Using a blackbody:

```
modelflux arf="198.corr.arf" rmf="198.rmf"
model="xsphabs.abs1*xsbbody.b"
paramvals="abs1.nh=1.7;b.kT=0.5" emin=0.5 emax=7.0 rate=0.21

Model fluxes:
Rate (0.5,7)= 0.21 count s^-1
Photon Flux (0.5,7)= 0.00045321 photon cm^-2 s^-1
Energy Flux (0.5,7)= 1.7529e-12 erg cm^-2 s^-1
```

modelflux

And using a more detailed functional form obtained from a Sherpa fit, we get:

```
modelflux arf="198.corr.arf" rmf="198.rmf"
model="xsphabs.abs1*(xsvnei.v+xsbbody.b)"
paramvals="abs1.nh=1.7;v.kT=1.0;v.Tau=7.0e
+10;v.norm=0.0019;b.kT=0.5;b.norm=3.0e-5"
emin=0.3 emax=7.0 rate=0.21

Flux (0.5 -7 keV ) = (1.46+/-0.11) e-12 erg/cm**2/s
```

eff2evt

How about a model independent flux?

The eff2evt tool estimates a flux per photon for each event of approximate energy E, given the observation livetime T, as

$$F = E * (1 / A(E)) * (1 / T) \quad \text{erg / cm**2 / s}$$

By summing these over all detected photons in an energy range we get the estimated total flux.

eff2evt

```
unix% eff2evt 198.evt"[sky=region(198.reg)]" 198.eevt
unix% eff2evt 198.evt"[sky=region(bg.reg)]" bg.eevt

unix% dmstat 198.eevt"[energy=500:7000][cols flux]" | grep sum
    sum:            1.5567785755e-12

unix% dmstat bg.eevt"[energy=500:7000][cols flux]" | grep sum
    sum:            1.6160426087e-13
```

So we calculate the background contribution as

```
    bg = ( 314.1 / 1256.6 ) 1.62e-13 = 0.04e-12
```

and hence the net flux is

```
    net = 1.56e-12 - 0.04 e-12 = 1.52 e-12
```

eff2evt

Correcting for the PSF by x 1/0.98:

```
    net,corr = 1.55 e-12 erg/cm**2/s
```

Scaling with the count rate errors from aprates:

```
     F = 1.55 +/- 0.12 e-12 erg/cm**2/s
```

in good agreement with the Sherpa model estimate.

Part 3: Contributed Tarfile Updates

Keeping Current

The contributed tarfile is updated on a regular basis to release new features and bug fixes. The version file is updated with each release:

```
unix% cat $ASCDS_CONTRIB/VERSION.CIAO_scripts
07 Jul 2011
```

The release is announced in three places:

- CIAO News page (with RSS feed):
  http://cxc.harvard.edu/ciao/news.html

- chandra-users email list:
  http://cxc.harvard.edu/chandra-users/discussion_group.html

- Contributed tarfile webpage:
  http://cxc.harvard.edu/ciao/download/scripts/