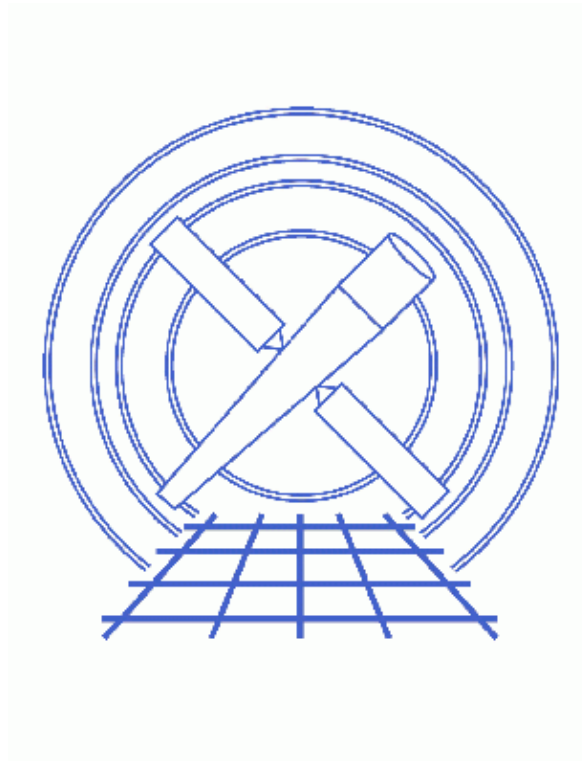


Create a Phase-binned Spectrum



CIAO 3.4 Science Threads

Table of Contents

- *Get Started*
- *Define a Source Region*
- *Create PHASE column*
 1. Convert Time to Phase
 2. “Fold” the Observation
- *Make a Lightcurve (optional)*
- *Extract the Spectrum*
 1. Run dmextract
 2. Plot the Spectrum
- *Caveats*
- *Parameter files:*
 - ◆ dmcalc
 - ◆ dmextract
 - ◆ dmextract
- *History*
- *Images*
 - ◆ Image of CC-mode observation
 - ◆ Source region overlaid on image
 - ◆ Phase-binned lightcurve
 - ◆ Phase-resolved spectrum

Create a Phase-binned Spectrum

CIAO 3.4 Science Threads

Overview

Last Update: 1 Dec 2006 – updated for CIAO 3.4: ChIPS and Sherpa versions

Synopsis:

Creating a phase-binned spectrum for a periodic source allows the user to compare the spectrum from different portions of the cycle and look for changes that may occur as the source rotates or orbits.

Purpose:

To create a phase-binned spectrum.

Read this thread if:

you are working with any observation that has a known (or predicted) period.

Proceed to the [HTML](#) or hardcopy (PDF: [A4](#) / [letter](#)) version of the thread.

Get Started

Sample ObsID used: 133 (ACIS-I, PSR B0540-69)

File types needed: evt2

The pulsar used here has a frequency of 19.794885 Hz (period of 50.5181 ms).

The standard event files have not been barycenter corrected. We have already run the [Apply Barycenter Correction](#) thread to generate barycenter-corrected times, accounting for the difference in photon arrival times as Chandra orbits the Earth and the Earth moves around the Sun. This barycenter correction should precede any phase calculation.

Hereafter we will call the level=2 event file:

```
acis_133_bary_evt2.fits
```

It is important to note that you should not fold a spectrum on a period that is close to or smaller than the temporal resolution of the data. The majority of ACIS observations are done in timed exposure (TE) mode, collecting photons and reading them out periodically. Each readout is called a "frame" and every event in a given frame has the same time assigned to it. The default resolution is 3.2 s, plus 0.04104 seconds frame transfer time; the exact value used, including the frame transfer time, is stored in the TIMEDEL header

keyword:

```
unix% dmkeypar acis_timed_evt2.fits TIMEDEL echo+
3.24104
```

Trying to analyze this TE-mode observation on the 50 ms timescale used in this thread would result in meaningless output.

The benefit of continuous clocking (CC) mode observations, such as the one used here, is that they have a much shorter time resolution since the data is continually read out:


```
unix% dmkeypar acis_133_bary_evt2.fits TIMEDEL echo+
0.00285
```


Define a Source Region

In subsequent steps of this thread, we are only interested in results from the source region, not the full field of view. Here we show how to define the source region.

Load the file into ds9:

```
unix% ds9 acis_133_bary_evt2.fits &
```

Since this is a CC-mode observation, there is only one spatial dimension (see the [Continuous Clocking Mode why topic](#) for details), so the image looks like [Figure 1](#) .

Outline the source on the image. [Figure 2](#)  shows the rotated box used in this example; for information on how to rotate shapes, see [this FAQ](#).

Then save the region from ds9:

- Region -> File Format-> Ciao
- Region -> File Coordinate System -> Physical
- Region -> Save Regions... -> Save As "source.reg"

The resulting region file will look something like this:

```
unix% more source.reg
# Region file format: CIAO version 1.0
rotbox(4095.125,4054.625,19.815731,5.6090815,333.70372)
```

If you are interested, you can use this region file as input to [dmcopy](#) to create a new, smaller event list. If you choose to do this, you will not need to specify a source region in the [Run dmextract step](#).

Create PHASE column

When working with large files, you can save some time by combining the two [dmtcalc](#) steps that are used in this section. Here we give the short-and-sweet example; for a full explanation of the steps, jump ahead to the [Convert Time to Phase](#) step.

```
unix% punlearn dmtcalc
unix% dmtcalc infile=acis_133_bary_evt2.fits outfile=acis_133_PHASE.fits \
expression=@dmtcalc.expr
```

where

```
unix% cat dmtcalc.expr
GPHASE=(time-TSTART)*19.794885
PHASE=GPHASE-(long)GPHASE
```

Note that this step does not take in account the frequency derivative; see the end of [the first section](#) for information on how to do so.

After completing this step, you can go right to the [Extract the Spectrum](#) section of this thread.

1. Convert Time to Phase

In order for these calculations to make sense, the event times in the evt2.fits file need to be recast relative to the start of the cycle. Since the epoch of phase zero is not known, we need to pick an arbitrary reference point for the phase. In this case, we use the start time of the observation, which is recorded in the header keyword TSTART. You can use a different point at which the phase is known to be zero, but keep in mind that you will need to figure out a way to convert it to the mission time used in the FITS files (seconds since JD 2450814.5).

Run [dmtcalc](#) on the file, creating a new column named GPHASE. The expression used below first subtracts the offset from the zero point and then multiplies by the frequency (here 19.794885 Hz). Since the tool can recognize Chandra header keywords, we can put TSTART directly into the equation:

```
unix% punlearn dmtcalc
unix% pset dmtcalc infile=acis_133_bary_evt2.fits
unix% pset dmtcalc outfile=acis_133_GPHASE.fits
unix% pset dmtcalc expression="GPHASE=(time-TSTART)*19.794885"
unix% dmtcalc
Input file (acis_133_bary_evt2.fits):
Output file (acis_133_GPHASE.fits):
expression(s) to evaluate (GPHASE=(time-TSTART)*19.794885):
```

Specify the frequency accurately enough to keep errors in phase small. This is particularly important when working with high frequencies.

If you are interested in taking the frequency derivative ("f dot") into account, you must modify the [expression](#) parameter accordingly.

*For reference, the generic form of the **expression** parameter is:*

```
unix% pset dmtcalc expression="GPHASE=(time-TSTART)*(f0 +(fdot * (time-TSTART)))"
```

or, if you are working with with the period instead of the frequency:

```
unix% pset dmtcalc expression="GPHASE=(time-TSTART)/(P0 +(Pdot * (time-TSTART)))"
```

2. ``Fold'' the Observation

``Folding'' the observation entails associating all the events that happen at a given point in the phase cycle with one another, with the phase traditionally being defined as a fraction of the orbital cycle < 1. To do this, the integer portion of the GPHASE must be subtracted off so that all values are greater than or equal to 0 and less than 1 (e.g. phases of 0.5, 1.5, and 2.5 all fall at identical points in the cycle):

```
unix% punlearn dmtcalc
unix% pset dmtcalc infile=acis_133_GPHASE.fits
unix% pset dmtcalc outfile=acis_133_PHASE.fits
unix% pset dmtcalc expression="PHASE=GPHASE-(long)GPHASE"
unix% dmtcalc
```

```
Input file (acis_133_GPHASE.fits):
Output file (acis_133_PHASE.fits):
expression(s) to evaluate (PHASE=GPHASE-(long)GPHASE):
```

The contents of the parameter file may be checked using [plist dmtcalc](#).

Make a Lightcurve (optional)

At this point, it is possible to make a lightcurve of the phase-folded data. Use [dmextract](#) to bin on the PHASE column while simultaneously filtering on the source region:

```
unix% dmextract "acis_133_PHASE.fits[sky=region(source.reg)][bin PHASE=0:1:0.1]" \
  acis_133_lightcurve.fits opt=generic
```


The resultant lightcurve can be displayed in [ChIPS](#):

```
unix% chips

Welcome to ChIPS, version CIAO 3.4
Copyright (C) 1999-2003, Smithsonian Astrophysical Observatory

chips> curve "acis_133_lightcurve.fits[cols phase,counts,stat_err]" x 1 y 2 yerr 3
chips> xlabel PHASE
chips> xlabel size 1.5
chips> ylabel COUNTS
chips> ylabel size 1.5

chips> quit
```

[Figure 3](#)  shows the plot, which clearly illustrates the counts as a function of phase.

IMPORTANT NOTE: make sure that an interesting feature that may appear in your lightcurve is not the result of a false period; for more information, see the [Timing Analysis with Lightcurves](#) why topic. An unfortunate choice in the width of the phase bin may lead to features that are not real. This is due in part to the fact that events need not be distributed uniformly in the phase histogram even if they *are* distributed uniformly in phase from the source.

Also, note that exposure times are *not* modified and that the counts histogram has no GTI information. This is the same limitation that is mentioned in the [Caveats section](#).

The contents of the parameter file may be checked using [plist dmextract](#).

Extract the Spectrum

1. Run dmextract

Now a source spectrum for any phase interval, in this case between 0.2 and 0.4, can be created.

Using the region we defined [previously](#):

```
unix% punlearn dmextract
unix% pset dmextract \
  infile="acis_133_PHASE.fits[sky=region(source.reg),phase=0.2:0.4][bin pi]"
unix% pset dmextract outfile=acis_133_phase_bin_spectrum.fits
```

```
unix% dmextract
Input event file (acis_133_PHASE.fits[sky=region(source.reg),phase=0.2:0.4][bin pi]):
Enter output file name (acis_133_phase_bin_spectrum.fits):
```

The contents of the parameter file may be checked using [plist dmextract](#).

2. Plot the Spectrum

The resulting spectrum may be plotted in *Sherpa*:

```
unix% sherpa

-----
Welcome to Sherpa: CXC's Modeling and Fitting Program
-----
Version: CIAO 3.4


Type AHELP SHERPA for overview.
Type EXIT, QUIT, or BYE to leave the program.

Notes:
  Temporary files for visualization will be written to the directory:
  /tmp
  To change this so that these files are not deleted when you exit Sherpa,
  edit $ASCDS_WORK_PATH in your 'ciao' setup script.

  Abundances set to Anders & Grevesse

sherpa> data acis_133_phase_bin_spectrum.fits
The inferred file type is PHA.  If this is not what you want, please
specify the type explicitly in the data command.

sherpa> ploty counts
sherpa> sherpa.dataplot.symbolstyle="none"
sherpa> sherpa.dataplot.curvestyle="simpleline"
sherpa> lplot data
```

which will give you a plot as shown in [Figure 4](#) . The value along the y-axis is the sum of all the counts that occurred in the chosen 20% of the observation's cycle. In this example, we *did not* include "f dot" in the [Convert Time to Phase](#) step.

Caveats

1. This thread does not correctly account for the data in the GTI tables when binning on phase. That is, the EXPOSURE keyword in the header of the new file **IS NOT** the sum of the new time bins. It follows the usual guidelines: filtering on one column doesn't update any information in the other columns. There are future plans to update this thread with a method that properly modifies the GTIs (through use of [dmgti](#)).
2. For fast timing of ACIS CC data, there are other corrections that may be important, such as read-out time and removing the dither and motion of the SIM . There are planned modifications to [acis_process_events](#) to take these calculations into account. Until the updated tool is available, please see [Zavlin, et al. 2000 \(ApJ, 540, L25\)](#) for a discussion of correcting the event arrival times.

Note that the Zavlin paper does not describe how to remove the readout time, only the dither and SIM motion. As a result, the events will still have some phase offset, but it will be the same for all of them. A comparison of the absolute phase of Chandra ACIS CC data to data from another mission can not be performed until this effect is also removed.

Create a Phase-binned Spectrum – CIAO 3.4

Parameters for /home/username/cxcds_param/dmtcalc.par

```
infile = acis_133_GPHASE.fits Input file
outfile = acis_133_PHASE.fits Output file
expression = PHASE=GPHASE-(long)GPHASE expression(s) to evaluate
(kernel = default)          Data Model creation/copy kernel
(clobber = no)              Clobber output file if it exists?
(verbose = 0)                Debug level
(mode = ql)
```

Parameters for /home/username/cxcds_param/dmextract.par

```
#-----
#
# DMEXTRACT -- extract columns or counts from an event list
#
#-----
infile = acis_133_PHASE.fits[sky=region(source.reg)][bin PHASE=0:1:0.1] Input event file
outfile = acis_133_lightcurve.fits Enter output file name
(bkg = )          Background region file or fixed background (counts/pixel/s) subtracti
(error = gaussian) Method for error determination(poisson|gaussian|<variance file>)
(bkgerror = gaussian) Method for background error determination(poisson|gaussian|<variance
(bkgnorm = 1.0)      Background normalization
(exp = )            Exposure map image file
(bkgexp = )        Background exposure map image file
(sys_err = 0)      Fixed systematic error value for SYS_ERR keyword
(opt = generic)    Output file type: phal
(defaults = ${ASCDS_CALIB}/cxo.mdb -> /soft/ciao/data/cxo.mdb) Instrument defaults file
(wmap = )          WMAP filter/binning (e.g. det=8 or default)
(clobber = no)     OK to overwrite existing output file(s)?
(verbose = 0)      Verbosity level
(mode = ql)
```

Parameters for /home/username/cxcds_param/dmextract.par

```
#-----
#
# DMEXTRACT -- extract columns or counts from an event list
#
#-----
infile = acis_133_PHASE.fits[sky=region(source.reg),phase=0.2:0.4][bin pi] Input event file
outfile = acis_133_phase_bin_spectrum.fits Enter output file name
(bkg = )          Background region file or fixed background (counts/pixel/s) subtracti
(error = gaussian) Method for error determination(poisson|gaussian|<variance file>)
(bkgerror = gaussian) Method for background error determination(poisson|gaussian|<variance
(bkgnorm = 1.0)      Background normalization
(exp = )            Exposure map image file
(bkgexp = )        Background exposure map image file
(sys_err = 0)      Fixed systematic error value for SYS_ERR keyword
(opt = phal)       Output file type: phal
(defaults = ${ASCDS_CALIB}/cxo.mdb -> /soft/ciao/data/cxo.mdb) Instrument defaults file
(wmap = )          WMAP filter/binning (e.g. det=8 or default)
(clobber = no)     OK to overwrite existing output file(s)?
(verbose = 0)      Verbosity level
```


(mode = ql)

History

03 Jan 2005 reviewed for CIAO 3.2: no changes

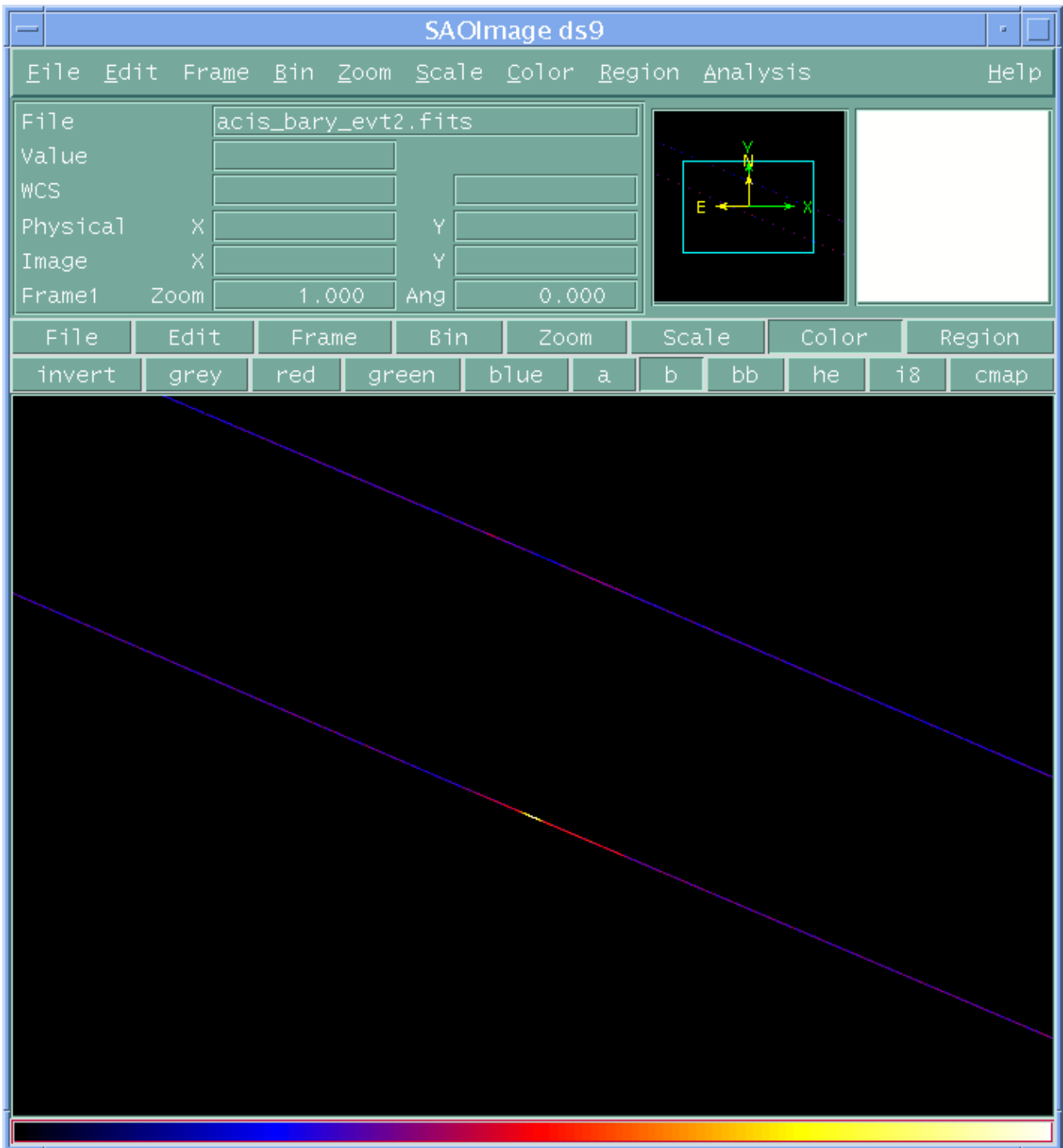
19 Dec 2005 updated for CIAO 3.3: default value of `dmextract` error and `bkgerror` parameters is "gaussian"; output filenames include ObsID

01 Dec 2006 updated for CIAO 3.4: ChIPS and Sherpa versions

URL: http://cxc.harvard.edu/ciao/threads/phase_bin/

Last modified: 1 Dec 2006

Image 1: Image of CC-mode observation



In CC-mode observations, there is only one spatial dimension. The other is sacrificed to increase the temporal resolution of the data

Image 2: Source region overlaid on image

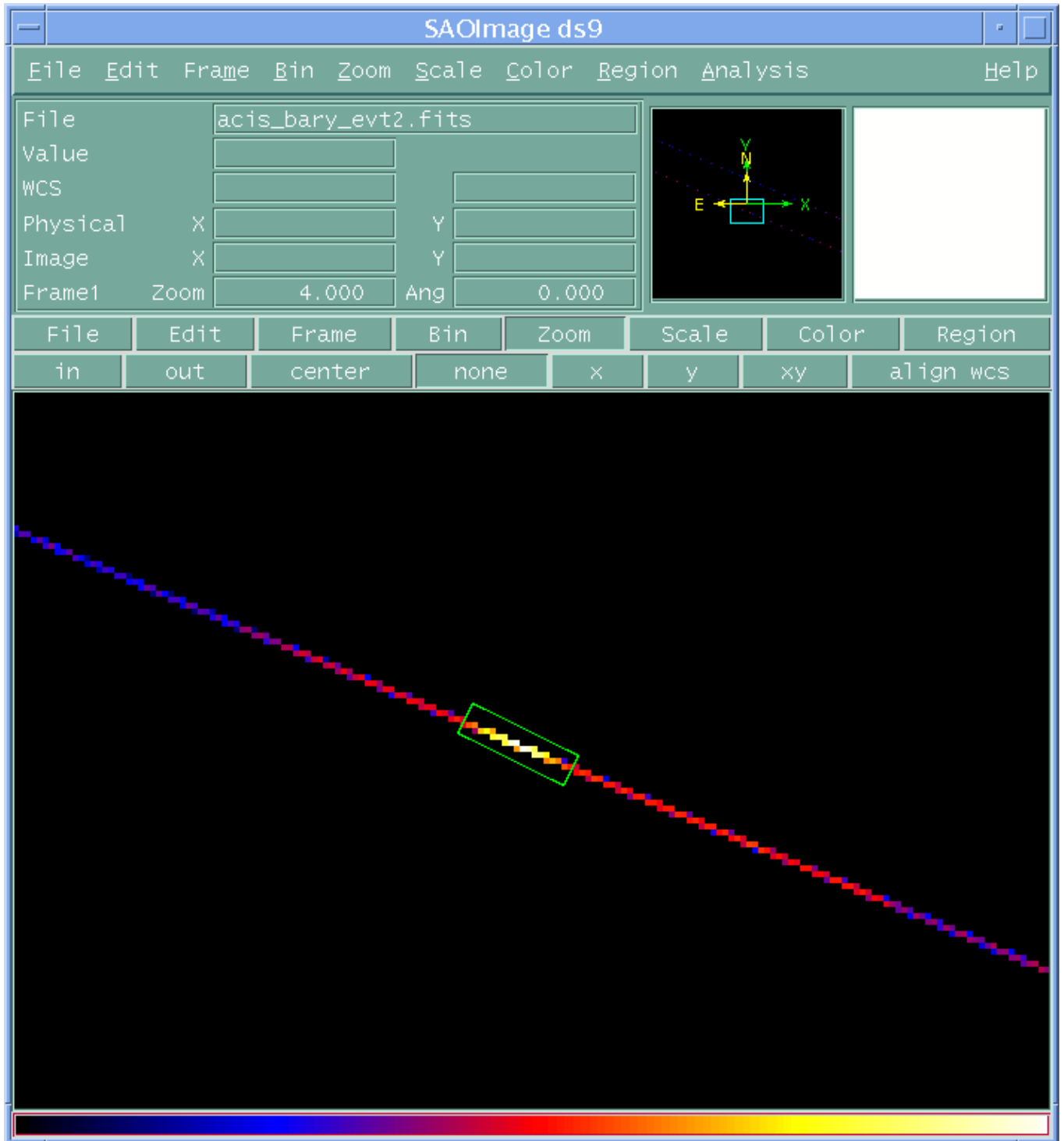


Image 3: Phase-binned lightcurve

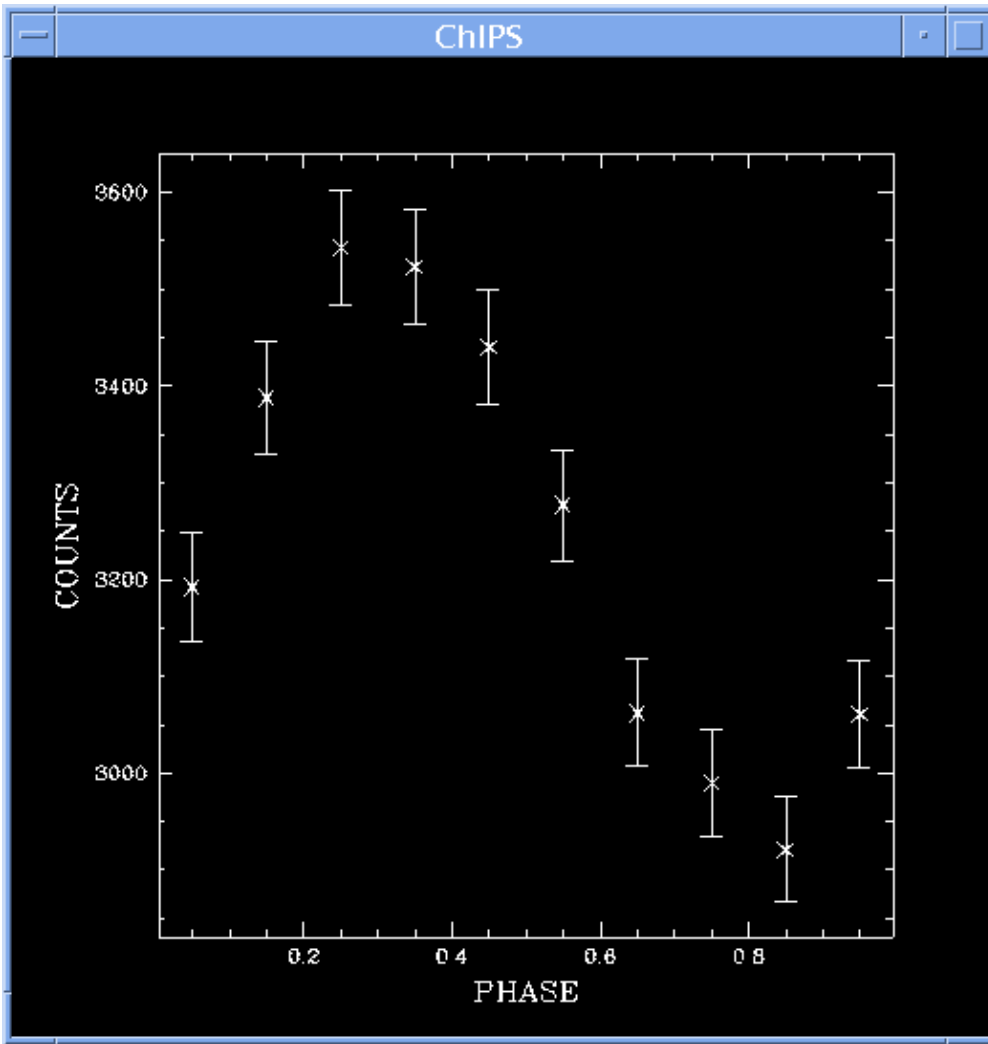
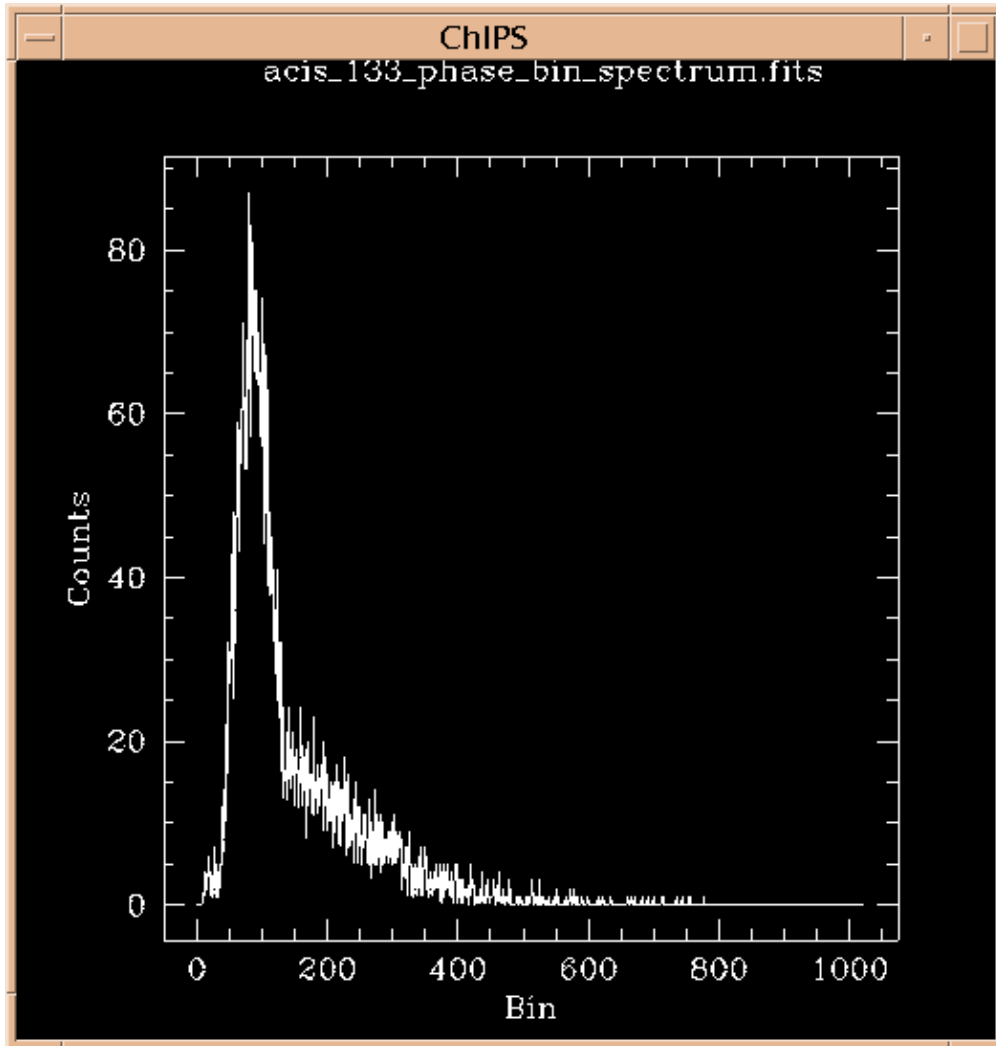


Image 4: Phase-resolved spectrum

This plot shows the spectrum of the source when it has a phase between 0.2 and 0.4.



Create a Phase-binned Spectrum – CIAO 3.4