Data Visualization

Sherpa Threads (CIAO 3.4)

Table of Contents

Data Visualization

Sherpa Threads

Overview

Last Update: 1 Dec 2006 – updated for CIAO 3.4: ChIPS version

Synopsis:

This thread provides a detailed introduction to data visualization in *Sherpa*.

Related Links:

- Sherpa Configuration: Using the State Objects thread: how the state objects (a.k.a. configuration variables) can be used to customize various features of *Sherpa*, including plots.
- The other threads on plotting in *Sherpa*.

Proceed to the HTML or hardcopy (PDF: A4 | letter) version of the thread.

Getting Started

Please follow the "Sherpa Threads: Getting Started" thread.

1−D Data

Introduction to 1−D Data Display in Sherpa

For 1−D data, the primary plotting command is LPLOT. In addition, a FITS Embedded Function (FEF) file that is read in via FEFFILE may be plotted using the command FEFPLOT. And, the OPLOT command may be used to plot multiple data curves into the same drawing area.

This thread demonstrates 1−D data visualization using LPLOT in detail.

Also, see the Sherpa Reference Manual for further information regarding data display capabilities within *Sherpa*. In particular, the *Sherpa* Display chapter of the Reference Manual contains information about 1−D data visualization.

1−D Plotting

A *Sherpa* session typically begins with the input of data. Here, we input a 1−D dataset and errors:

```
sherpa> READ DATA data1.dat 1 2
```

```
sherpa> READ ERRORS data1.dat 1 3
```

These data may then be plotted from within *Sherpa*:

```
sherpa> LPLOT DATA
```

Figure 1 📷 shows the resulting plot.

Next, we continue the *Sherpa* session by defining a source model, thawing model parameters, and fitting:

```
sherpa> PARAMPROMPT OFF
Model parameter prompting is off
sherpa> POLY[model1]
sherpa> model1 INTEGRATE OFF
sherpa> SOURCE = model1
sherpa> THAW model1.c1
sherpa> FIT
 LVMQT: V2.0
 LVMQT: initial statistic value = 2815.14
 LVMQT: final statistic value = 151.827 at iteration 5
        model1.c0  1.58227
        model1.c1  0.198455
```

Then, these data and fit may be plotted:

```
sherpa> LPLOT FIT
```

Next, we thaw another parameter, fit again, and plot this fit:

```
sherpa> THAW model1.c2
sherpa> FIT
 LVMQT: V2.0
 LVMQT: initial statistic value = 151.827
 LVMQT: final statistic value = 59.0027 at iteration 4
        model1.c0  1.30826
        model1.c1  0.347303
        model1.c2  -0.0135317

sherpa> LPLOT FIT
```

Figure 2 📷 shows the resulting plot.

Modifying Plots

Plotting in *Sherpa* works in concert with the *CIAO* plotting tool *ChIPS*. *ChIPS* commands may be issued from within *Sherpa* to modify plots. For example, the following *ChIPS* commands may be issued to add labels to the X and Y axes of the existing plot:

```
sherpa> XLABEL "X = Off-Axis (arcmin)"
sherpa> YLABEL "F(X) = SNR"
sherpa> REDRAW
```

Figure 3 📷 shows the resulting plot. Note that the command REDRAW must be issued in order for the Sherpa plot to be updated with previously issued *ChIPS* commands.

An important plotting concept is the numbering of curves. For example, in the current plot there are two curves; the data are curve number 1 and the red fit is curve number 2. To change the fit curve from histogram to a line, we issue the following command:

```
sherpa> C 2 SIMPLELINE
sherpa> REDRAW
```

Where, C 2 specifies that the SIMPLELINE command is to be applied to curve number 2. Similarly, we

make a modification to curve number 1 (change the symbol style to a cross):

```
sherpa> C 1 SYMBOL CROSS
sherpa> REDRAW
```

The *ChIPS* command INFO is useful for getting information on the number of existing curves, etc:

```
sherpa> INFO

Drawing Area  #1  << CURRENT DRAWING AREA
(Location: 0.15 0.9 0.1 0.9)
(Limits  : -7.45058e-09 11 1.37791 3.59135)
(Axes    : fouraxes   color: defaultcolor  width: 1)
        Curve #1
                SYMBOL: defaultcolor  cross
                CONNECTOR: noline  << CURRENT CURVE
        Curve #2
                SYMBOL: none
                CONNECTOR: red  simpleline
                LINE: solid
```

We next make another plot modification using the *ChIPS* command ERRS, to change the style of the errorbars:

```
sherpa> ERRS STANDARD
sherpa> REDRAW
```

Figure 4 📷 shows the resulting plot.

To add labels to the plot, we use the *ChIPS* command LABEL:

```
sherpa> LABEL 4.0 2.0 "F(X)=1.3083+0.3473X-0.0135X^2"
```

where the 4.0 2.0 coordinates specify the location on the plot for the placement of the label. To change this placement, the command is:

```
sherpa> L 1 4.0 1.5
sherpa> REDRAW
```

We may also add a second label:

```
sherpa> LABEL 1.0 3.3 "ACIS"
```

Now, to delete the first label:

```
sherpa> L 1 DEL
sherpa> REDRAW
```

To change the placement of the second label:

```
sherpa> L 2 4.0 2.0
sherpa> REDRAW
```

Note that, like curves, labels are also numbered. And, the L # and DEL commands must specify the label number.

Note that plots and their data may be saved: see the subsequent section Saving A *Sherpa* Session And Plot.

Plotting Into Multiple Windows

One may plot into multiple windows. For example, the following command plots the fit in one window and the fit residuals in another:

```
sherpa> LPLOT 2 FIT RESIDUALS
```

Figure 5 shows the resulting plot.

Various modifications may then be made to this plot using *ChIPS* commands:

```
sherpa>   # Change the data and fit plots in the 1st drawing
sherpa>   # area (D 1) to block symbols, and a line, respectively:
sherpa> D 1 C 1 NOLINE
sherpa> D 1 C 2 SIMPLELINE
sherpa> REDRAW
```

Note that drawing areas are also numbered, from top to bottom. In the above, we specify the drawing area number with D #. For example, D 1 C 1 NOLINE changes curve number 1 in drawing area number 1, and D 1 C 2 SIMPLELINE changes curve number 2 in drawing area number 1.

```
sherpa>   # Add labels to the X and Y Axes:
sherpa> D 2 XLABEL "X = Off-Axis (arcmin)"
sherpa> D 1 YLABEL "F(X) = SNR"
sherpa> REDRAW
```

Again, note that we specify the drawing area for the labels. For example, D 2 XLABEL adds an X axis label to drawing area number 2 (*i.e.* the bottom−most drawing area), and D 1 YLABEL adds a Y axis label to drawing area number 1.

```
sherpa>   # Add a title:
sherpa> TITLE "ACIS 25000 Counts Per Chip"
sherpa> REDRAW

sherpa>   # Remove the X Axis label from the 1st drawing area:
sherpa> D 1 XLABEL ""
sherpa> REDRAW

sherpa>   # Place a separation between the two drawing areas:
sherpa> SPLIT GAP y 0.04
sherpa> REDRAW

sherpa>   # Remove the X Axis tick marks from the 1st drawing area:
sherpa> D 1 TICKVALS X OFF
sherpa> REDRAW

sherpa>   # Change the format of the tick value labels on the Y Axes:
sherpa> D ALL TICKVALS y "%1.2f"
sherpa> REDRAW
```

Note that the "%1.2f" portion of the TICKVALS command specifies the format of the tick value labels using the standard C string format notation (see man printf).

```
sherpa>   # Add a label that contains the fit results:
sherpa> D 1 LABEL 2.0 1.7 "F(X)=1.3083+0.3473X-0.0135X^2"
sherpa> REDRAW
```

Note that text for labels may include *TeX* syntax for formatting text and creating mathematical expressions.

Figure 6 shows the resulting plot.

Printing Plots To PostScript

To create a hard copy of an existing plot, the command is PRINT:

```
sherpa> PRINT POSTFILE myplot1.ps
```

This command writes the plot to a PostScript file, which we here named myplot1.ps. This file may then be

sent to the printer, perhaps as follows:

```
sherpa> $lpr myplot1.ps
```

Saving A Sherpa Session And Plot

It is important to note that the data displayed in the current plot and the commands used to generate the plot are *not* written to disk by default. To save the current *Sherpa* plot, and its associated data:

- The plot and its data may be saved using the following *ChIPS* command:

```
sherpa> STORE myplot1.chp
```

This command specifies that a record of the commands used to generate the plot be saved in a file named myplot1.chp. The STORE command also creates a FITS file that contains the plotted data (e.g. myplot1.chp.fits).

We can confirm that the myplot1.chp file and the myplot1.chp.fits datafile were indeed written:

```
sherpa> ls myplot1*
myplot1.chp            myplot1.chp.fits
```

Examine the myplot1.chp using the *Unix* command more and note that this file contains all of the *ChIPS* commands necessary to create the plot; it may be edited to further modify the plot.

```
sherpa> $more myplot1.chp
#This is a ChIPS state file.
#ChIPS_Version: 2.20


redraw off


##############################
# Start of a new drawing area.
##############################
drawarea 0.15 0.9 0.52 0.9
label 2 1.7 "F(X)=1.3083+0.3473X-0.0135X^2"

#######################
# Start of a new curve.
skip 0
curve "/export/work/sherpa_threads/sherpa/plot/myplot1.chp.new.fits[curve5]" yup 3 ydn 4
symbol  square
symbol 4 0
symbol 2
errs x size 2
.
.
.
```

- In addition to saving the plot and the data used in the plot, we may also save the current *Sherpa* session:

```
sherpa> SAVE ALL mysession1.shp
```

Restoring A Sherpa Plot And Session

We may now exit *Sherpa*:

```
sherpa> EXIT
Goodbye.
```

And, later return to where we left off:

```
unix% sherpa
sherpa> RESTORE myplot1.chp
```

The above RESTORE command restores the plot. Note that the plot is restored using the commands recorded in the myplot1.chp file, and the myplot1.chp.fits datafile that was output and is referenced in the myplot1.chp file.

We are now able to make additional changes to the plot:

```
sherpa> TITLE BLUE
sherpa> REDRAW
```

It is important to note, however, that the *Sherpa* session has not yet been restored:

```
sherpa> SHOW

Optimization Method: Levenberg-Marquardt
Statistic:           Chi-Squared Gehrels
```

To restore the *Sherpa* session:

```
sherpa> USE mysession1.shp

sherpa> SHOW

Optimization Method: Levenberg-Marquardt
Statistic:           Chi-Squared Gehrels

-----------------
Input data files:
-----------------

Data 1: data1.dat ascii 1 2.
Total Size: 11 bins (or pixels)

(etc)
```

We are now able to continue with the fitting session:

```
sherpa> THAW model1.c3
sherpa> FIT
 LVMQT: V2.0
 LVMQT: initial statistic value = 59.0027
 LVMQT: final statistic value = 30.8491 at iteration 6
        model1.c0  1.49843
        model1.c1  0.1447
        model1.c2  0.0322936
        model1.c3  -0.00277729

sherpa> LPLOT 2 FIT RESIDUALS
```

Note that a plot saved from within *Sherpa* may also be restored into *ChIPS*:

```
unix% chips
```

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

chips> RESTORE myplot1.chp
```

It is sometimes more convenient to work within *ChIPS* itself when performing further plot modifications.

Saving Plotted Data To Output Files

As before, we can improve this plot using *ChIPS* commands:

```
sherpa> LPLOT 2 FIT RESIDUALS
sherpa> D 1 C 1 NOLINE
sherpa> D 1 C 2 SIMPLELINE
sherpa> D 2 XLABEL "X = Off-Axis (arcmin)"
sherpa> D 1 YLABEL "F(X) = SNR"
sherpa> TITLE "ACIS 25000 Counts Per Chip"
sherpa> D 1 XLABEL ""
sherpa> SPLIT GAP y 0.04
sherpa> D 1 TICKVALS X OFF
sherpa> D 1 TICKVALS y "%1.2f"
sherpa> D 2 TICKVALS y "%1.2f"
sherpa> D 1 LABEL 2.0 1.7 "F(X)=1.4984+0.1447X+0.0323X^2-0.0028X^3"
sherpa> REDRAW
```

And also as before, we can save the data being plotted, including this new fit, using the STORE command:

```
sherpa> STORE myplot2.chp
```

This command writes the plotted data to the current directory, and names the file myplot2.chp.fits. The commands used to generate the plot are saved to a file named myplot2.chp.

If we'd also like to save the *Sherpa* session, to possibly return to later, then we must issue the SAVE ALL command:

```
sherpa> SAVE ALL mysession2.shp
sherpa> EXIT
Goodbye.
```

Restoring Only A Plot

A plot may be restored either in *ChIPS*, or in *Sherpa*:

- To restore a plot in *ChIPS*:

```
unix% chips
chips> RESTORE myplot2.chp
```

One may then issue further *ChIPS* commands to modify the plot.

- To restore a plot in *Sherpa*:

```
unix% sherpa
sherpa> RESTORE myplot2.chp
```

Figure 7 shows the resulting plot. One may then issue further *ChIPS* commands to modify the plot.

In addition to restoring the plot, we may also restore the previously saved *Sherpa* session:

```
sherpa> USE mysession2.shp
.
```

```
.
sherpa> EXIT
Goodbye.
```

Overplotting

Two different datasets may be plotted separately in different drawing areas:

```
unix% sherpa
sherpa> DATA 1 data1.dat
sherpa> DATA 2 data2.dat
sherpa> LPLOT 2 DATA 1 DATA 2
sherpa> D 1 C 1 ERRS NONE
sherpa> D 1 C 1 SYMBOL NONE
sherpa> D 1 C 1 HISTO
sherpa> D 2 C 1 ERRS NONE
sherpa> D 2 C 1 SYMBOL NONE
sherpa> D 2 C 1 HISTO
sherpa> REDRAW
```

Figure 8 shows the resulting plot.

Or, one may overplot two different datasets in the same drawing area, using the *Sherpa* OPLOT command:

```
sherpa> CLEAR
sherpa> OPLOT DATA 1 DATA 2
```

Figure 9 shows the resulting plot.

Note that overplotting effectively works only for datasets having the same data space units (*i.e.* energy, wavelength, PHA bin, *etc.*).

Saving Spectral Data In Energy Space

Without an instrument model definition, input PHA data are in count space. For example,

```
sherpa> ERASE ALL
sherpa> DATA data3.pha
The inferred file type is PHA.  If this is not what you want, please
specify the type explicitly in the data command.
WARNING: using systematic errors specified in the PHA file.
sherpa> LPLOT DATA
```

Figure 10 shows the resulting plot. To save these data to an output file, one may either use the *Sherpa* WRITE command, or one may save the data from the plot. Here, we save the data from the plot:

```
sherpa> STORE myplot3_channel.chp
```

This will result in a myplot3_channel.chp file that contains the commands needed to regenerate the plot, and a myplot3_channel.chp.fits datafile that contains the data (in ASCII).

To convert the PHA data to energy space, one must define an instrument model. For example,

```
sherpa> RSP[inst3](data3_rmf.fits, data3_arf.fits)
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
sherpa> INSTRUMENT = inst3
sherpa> LPLOT DATA
```

Figure 11 shows the resulting plot. To save these data (which are now in energy space) to an output file, one must save the data from the plot:

```
sherpa> STORE myplot3_energy.chp
```

This will result in a `myplot3_energy.chp` file that contains the commands needed to regenerate the plot, and a `myplot3_energy.chp.fits` datafile that contains the data in energy space (in ASCII).

One may EXIT, and then later regenerate these plots:

```
sherpa> EXIT
Goodbye.

unix% sherpa
sherpa> RESTORE myplot3_channel.chp
sherpa> RESTORE myplot3_energy.chp
```

Note that these plots may be regenerated in either *Sherpa* or *ChIPS*, and that these plots make use of the data saved in the `myplot3_channel.chp.fits` and `myplot3_energy.chp.fits` datafiles.

We may now exit *Sherpa*:

```
sherpa> EXIT
Goodbye.
```

Overplotting Two Spectra In Energy Space

In order to overplot two spectra in energy space, one must first individually plot each spectrum in energy space. These plotted data are saved and are then overplotted using *ChIPS*. The procedure is as follows:

- First, input the two spectra:

```
unix% sherpa
sherpa> DATA 1 srcc.pi
The inferred file type is PHA.  If this is not what you want, please
specify the type explicitly in the data command.
WARNING: statistical errors specified in the PHA file.
        These are currently IGNORED.  To use them, type:
        READ ERRORS "<filename>[cols CHANNEL,STAT_ERR]" fitsbin
RMF is being input from:
  /export/work/sherpa_threads/sherpa/plot/srcc.rmf
ARF is being input from:
  /export/work/sherpa_threads/sherpa/plot/srcc.arf
Background data are being input from:
  /export/work/sherpa_threads/sherpa/plot/srcc_bg.pi
WARNING: statistical errors specified in the PHA file.
        These are currently IGNORED.  To use them, type:
        READ BERRORS "<filename>[cols CHANNEL,STAT_ERR]" fitsbin


sherpa> IGNORE 1 BAD


sherpa> DATA 2 srce.pi
The inferred file type is PHA.  If this is not what you want, please
specify the type explicitly in the data command.
WARNING: statistical errors specified in the PHA file.
        These are currently IGNORED.  To use them, type:
        READ ERRORS "<filename>[cols CHANNEL,STAT_ERR]" fitsbin
RMF is being input from:
  /export/work/sherpa_threads/sherpa/plot/srce.rmf
ARF is being input from:
  /export/work/sherpa_threads/sherpa/plot/srce.arf
Background data are being input from:
  /export/work/sherpa_threads/sherpa/plot/srce_bg.pi
WARNING: statistical errors specified in the PHA file.
        These are currently IGNORED.  To use them, type:
```

```
        READ BERRORS "<filename>[cols CHANNEL,STAT_ERR]" fitsbin

sherpa> IGNORE 2 BAD
```

Note that RMF and ARF files were automatically loaded since the input datafiles contained headers containing paths to these calibration files. As such, instrument models are automatically defined by *Sherpa* for these datasets. So when the data are plotted, they are plotted in energy space:

```
sherpa> LPLOT 2 DATA 1 DATA 2
```

- These energy space data may also be overplotted, using the *Sherpa* OPLOT command:

```
sherpa> OPLOT DATA 1 DATA 2
sherpa> C 1 NOLINE
sherpa> C 2 NOLINE
sherpa> C 1 ERRS BOTH
sherpa> C 2 ERRS BOTH
sherpa> C 1 SYMBOL SQUARE
sherpa> C 1 SYMBOL SIZE 2.0
sherpa> C 1 SYMBOL RED
sherpa> C 2 SYMBOL SQUARE
sherpa> C 2 SYMBOL SIZE 2.0
sherpa> REDRAW
```

Figure 12 shows the resulting plot.

Overplotting Model Components

The following is an example of fitting and then overplotting the model components:

```
sherpa> ERASE ALL
sherpa> DATA data5_pi.fits
The inferred file type is PHA.  If this is not what you want, please
specify the type explicitly in the data command.
WARNING: statistical errors specified in the PHA file.
        These are currently IGNORED.  To use them, type:
        READ ERRORS "<filename>[cols CHANNEL,STAT_ERR]" fitsbin
sherpa> RSP[detector1](data5_rmf.fits, data5_arf.fits)
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
sherpa> INSTRUMENT = detector1
sherpa> IGNORE BAD
sherpa> METHOD POWELL

sherpa> PARAMPROMPT OFF
Model parameter prompting is off
sherpa> SOURCE = XSPHABS[abs] * POWLAW1D[power]
sherpa> abs.nH.value = 0.0454
sherpa> FREEZE abs.nH
sherpa> FIT
 powll: v1.2
 powll:  initial statistic value =      2.38800E+03
 powll:     converged to minimum =      2.23713E+02 at iteration =        4
 powll:    final statistic value =      2.23713E+02
        power.gamma  1.87995
        power.ampl  0.000371676

sherpa> GAUSS1D[g]
sherpa> g.pos=6.0
sherpa> g.pos.min=5.0
sherpa> g.pos.max=7.0
sherpa> SOURCE = abs*power + g
sherpa> FIT
 powll: v1.2
 powll:  initial statistic value =      8.71340E+05
```

```
 powll:       converged to minimum =       2.22480E+02 at iteration =       21
 powll:     final statistic value =       2.22480E+02
          power.gamma  1.88101
          power.ampl  0.000371562
              g.fwhm  0.104683
              g.pos  6.82238
              g.ampl  3.28794e-05

sherpa> GAUSS1D[g2]
sherpa> SOURCE = abs*power + g + g2
sherpa> g2.pos=6.5
sherpa> g2.pos.min=6.0
sherpa> g2.pos.max=7.0
sherpa> FIT
 powll: v1.2
 powll:  initial statistic value =      6.51744E+05
 powll:       converged to minimum =       2.22464E+02 at iteration =       20
 powll:     final statistic value =       2.22464E+02
          power.gamma  1.881
          power.ampl  0.000371562
              g.fwhm  0.0386909
              g.pos  6.76242
              g.ampl  2.79281e-05
             g2.fwhm  0.0228727
             g2.pos  6.84466
             g2.ampl  0.000103375

WARNING:
   The value of g2.fwhm is equal to the g2.fwhm.min limit boundary.
   You may wish to consider changing min/max values and refitting.

sherpa> LPLOT FIT
sherpa> OPLOT SOURCE g g2
```

Figure 13 shows the resulting plot.

2−D Data

Introduction to 2−D Data Display in Sherpa

2−D data may be visualized with either a contour plot or a surface plot, using the plotting commands CPLOT and SPLOT respectively. For 2−D imaging data, the command is IMAGE. In addition, the REGION−PROJECTION command creates a contour plot of confidence regions, computed using the PROJECTION algorithm, and the REGION−UNCERTAINTY command creates a contour plot of confidence regions, computed using the UNCERTAINTY algorithm. Parts of this thread demonstrate 2−D data visualization using REGION−PROJECTION in detail.

2−D Plotting, Contour Plots

Here, 2−D data is input and then displayed using a contour plot:

```
sherpa> ERASE ALL
sherpa> READ DATA data6.dat ASCII 1 2 3
sherpa> CPLOT DATA
Contour Levels: 598 448.637 299.274 149.911 0.547731
Min: 0, Max: 598, Ave: 0.547731
```

Figure 14 shows the resulting plot.

Note that the plot levels may be modified using the *ChIPS* command <u>LEVELS</u>:

```
sherpa> LEVELS 3 5 20 100
sherpa> REDRAW
```

2–D Plotting, Surface Plots

These data may also be displayed using a surface plot:

```
sherpa> SPLOT DATA
```
Figure 15 [📷] shows the resulting plot.

Here, a second 2–D dataset is input, and then the two datasets are displayed into multiple windows, using surface plots:

```
sherpa> READ DATA 2 data7.dat ASCII 1 2 3
sherpa> SPLOT 2 DATA 1 DATA 2
```
Figure 16 [📷] shows the resulting plot.

2–D Plotting, REGION–PROJECTION

The current session is erased before inputting new data:

```
sherpa> ERASE ALL

sherpa> READ DATA data4_pi.fits
The inferred file type is PHA.  If this is not what you want, please
specify the type explicitly in the data command.
WARNING: statistical errors specified in the PHA file.
        These are currently IGNORED.  To use them, type:
        READ ERRORS "<filename>[cols CHANNEL,STAT_ERR]" fitsbin
```
These 1–D data may then be plotted from within *Sherpa*:

```
sherpa> LPLOT DATA
```
Next, we continue the *Sherpa* session by defining an instrument model, displaying the data in energy space, and filtering the dataset:

```
sherpa> PARAMPROMPT OFF
Model parameter prompting is off
sherpa> RSP[acis](data4_rmf.fits, data4_arf.fits)
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
sherpa> INSTRUMENT = acis
sherpa> LPLOT DATA
sherpa> IGNORE ENERGY :0.5, 8:
sherpa> LPLOT DATA
```
Next, model components and a source model are defined, the statistic is changed, and then the data are fit:

```
sherpa> POWLAW1D[p1]
sherpa> XSWABS[abs1]
sherpa> SOURCE = abs1*p1
sherpa> GUESS SOURCE
sherpa> STATISTIC CASH
sherpa> METHOD POWELL

sherpa> FIT
 powll: v1.2
```

```
 powll:   initial statistic value =      1.53063E+04
 powll:      converged to minimum =     -7.62300E+03 at iteration =      12
 powll:     final statistic value =     -7.62300E+03
           abs1.nH  1.48605  10^22/cm^2
            p1.gamma  0.815361
            p1.ampl  0.000711629


WARNING:
   The value of p1.ampl within 0.01% of the p1.ampl.max limit boundary.
   You may wish to consider changing min/max values and refitting.

sherpa> p1.ampl.max = 0.1
sherpa> FIT
 powll: v1.2
 powll:   initial statistic value =     -7.62300E+03
 powll:      converged to minimum =     -7.70549E+03 at iteration =       7
 powll:     final statistic value =     -7.70549E+03
           abs1.nH  2.37765  10^22/cm^2
            p1.gamma  1.50086
            p1.ampl  0.00200489


sherpa> LPLOT FIT
  ==> Error bars computed using Chi Gehrels.
```

Figure 17 shows the resulting plot of the fit.

Finally, we next use the REGION-PROJECTION command to create a 2–D contour plot of confidence regions, computed using the PROJECTION algorithm:

```
sherpa> REG-PROJ abs1.nH p1.gamma
Region-Projection: optimization reset to Simplex.
                  computing grid size with covariance...done.
                  outer grid loop 20% done...
                  outer grid loop 40% done...
                  outer grid loop 60% done...
                  outer grid loop 80% done...
Minimum: -7705.49
Levels are: -7703.2 -7699.31 -7693.66
----------------------------------------------------------------
NOTE: the limits on this plot cannot be altered.
      * To change the overall box size, you can:
         a) Set sherpa.regproj.expfac to a new value
            (if sherpa.regproj.arange = 1); or
         b) Set sherpa.regproj.min and max to new values
            (if sherpa.regproj.arange = 0).
      * To create a denser grid, set sherpa.regproj.nloop
        to new values.
      * To see current settings, type list_regproj().
      * For more information, do AHELP REGION-PROJECTION.
      This message will not be printed again during this session.
----------------------------------------------------------------
```

The REGION-PROJECTION parameter values are then modified as follows, and REGION-PROJECTION is then run again to produce another 2–D contour plot of the confidence regions:

```
sherpa> sherpa.regproj.arange = 0
sherpa> sherpa.regproj.min = [1.90,1.20]
sherpa> sherpa.regproj.max = [2.85,1.80]
sherpa> sherpa.regproj.nloop = [30,30]
sherpa> REG-PROJ abs1.nH p1.gamma
Region-Projection: optimization reset to Simplex.
                  grid size set by user.
                  outer grid loop 20% done...
                  outer grid loop 40% done...
                  outer grid loop 60% done...
                  outer grid loop 80% done...
```

```
Minimum: -7705.49
Levels are: -7703.2 -7699.31 -7693.66
```

Figure 18 📷 shows the resulting contour plot.

Modifying Plots

As with 1–D plotting, 2–D plotting in *Sherpa* also works in concert with the *CIAO* plotting tool *ChIPS*. So, *ChIPS* commands may be issued from within *Sherpa* to modify 2–D plots. For example, the following *ChIPS* commands may be issued to modify the existing contour plot:

```
sherpa> TWOAXES
sherpa> RED
sherpa> WIDTH 4.0
sherpa> REDRAW
```

Note that the command REDRAW must be issued in order for the plot to be updated with previously issued *ChIPS* commands.

Note, however, that axis limits on contour and surface plots may not be modified.

Printing Plots To PostScript

Just as with 1–D plots, to create a hard copy of an existing 2–D plot the command is PRINT:

```
sherpa> PRINT POSTFILE myplot4.ps
```

This command writes the plot to a PostScript file, which we here named myplot4.ps. This file may then be sent to the printer, perhaps as follows:

```
sherpa> $lpr myplot4.ps
```

Saving A Sherpa Session, Plot, and Plotted Data

Just as with 1–D plots, the data displayed in the current plot and the commands used to generate the plot are *not* written to disk by default. To save the current *Sherpa* plot, and its associated data:

- The plot may be saved using the following command:

  ```
  sherpa> STORE myplot4.chp
  ```

  This command specifies that a record of the commands used to generate the plot be saved in a file named myplot4.chp. The dataset used in the plot will be saved and named /tmp/<pid>sherpa_cont_params.ascii, where <pid> is a number that is related to the process ID of the current *Sherpa* session.

  We can confirm that the myplot4.chp file and the /tmp/<pid>sherpa_cont_params.ascii datafiles were indeed written:

  ```
  sherpa> ls myplot4*
  myplot4.chp
  sherpa> ls /tmp/*sherpa_cont_params.ascii
  /tmp/26978sherpa_cont_params.ascii
  ```

  We see that there is one datafile, as expected: /tmp/26978sherpa_cont_params.ascii contains the data being plotted. The contents of the myplot4.chp file are the commands needed to regenerate the plot, including commands to re–load this datafile:

```
sherpa> $more myplot4.chp
.
.
contour "/tmp/26978sherpa_cont_params.ascii"
```

- Since the datafile in /tmp/ will be deleted when the *Sherpa* session ends, it should be copied to the local directory:

```
sherpa> $cp /tmp/26978sherpa_cont_params.ascii myplot4.chp.dat
```

- In addition to saving the plot and the data used in the plot, we may also save the current *Sherpa* session:

```
sherpa> SAVE ALL mysession4.shp
```

- We may now exit *Sherpa*:

```
sherpa> EXIT
Goodbye.
```

And, later return to where we left off.
- The user will then need to edit the myplot4.chp file so that it points to the datafile myplot4.chp.dat, instead of to /tmp/26978sherpa_cont_params.ascii.

Restoring A Sherpa Plot And Session

```
unix% sherpa
sherpa> RESTORE myplot4.chp
```

The above RESTORE command restores the plot. Note that the plot is restored using the commands recorded in the myplot4.chp file, and the myplot4.chp.dat datafile that was output and is referenced in the myplot4.chp file.

We are now able to make additional changes to the plot:

```
sherpa> TITLE BLUE
sherpa> REDRAW
```

It is important to note, however, that the *Sherpa* session has not yet been restored.

To restore the *Sherpa* session and continue with further analyses:

```
sherpa> USE mysession4.shp
The inferred file type is ARF.  If this is not what you want, please
specify the type explicitly in the data command.
The inferred file type is PHA.  If this is not what you want, please
specify the type explicitly in the data command.
WARNING: statistical errors specified in the PHA file.
        These are currently IGNORED.  To use them, type:
        READ ERRORS "<filename>[cols CHANNEL,STAT_ERR]" fitsbin
sherpa> SHOW
.
.
.
```

History

14 Jan 2005 reviewed for CIAO 3.2: no changes

21 Dec 2005 reviewed for CIAO 3.3: no changes

01 Dec 2006 updated for CIAO 3.4: ChIPS version

Restoring A Sherpa Plot And Session

Image 1: Plotting 1D data in Sherpa

data1.dat
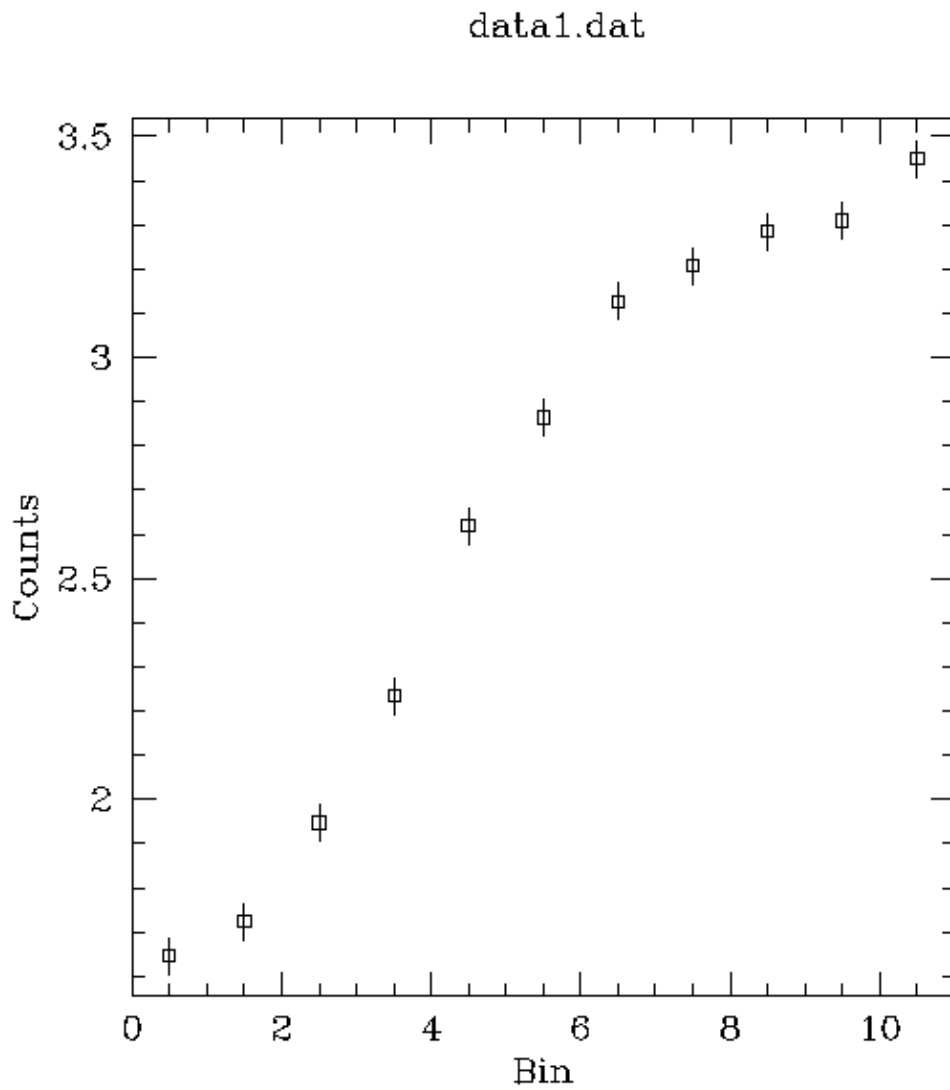
Image 2: Data and fit

data1.dat
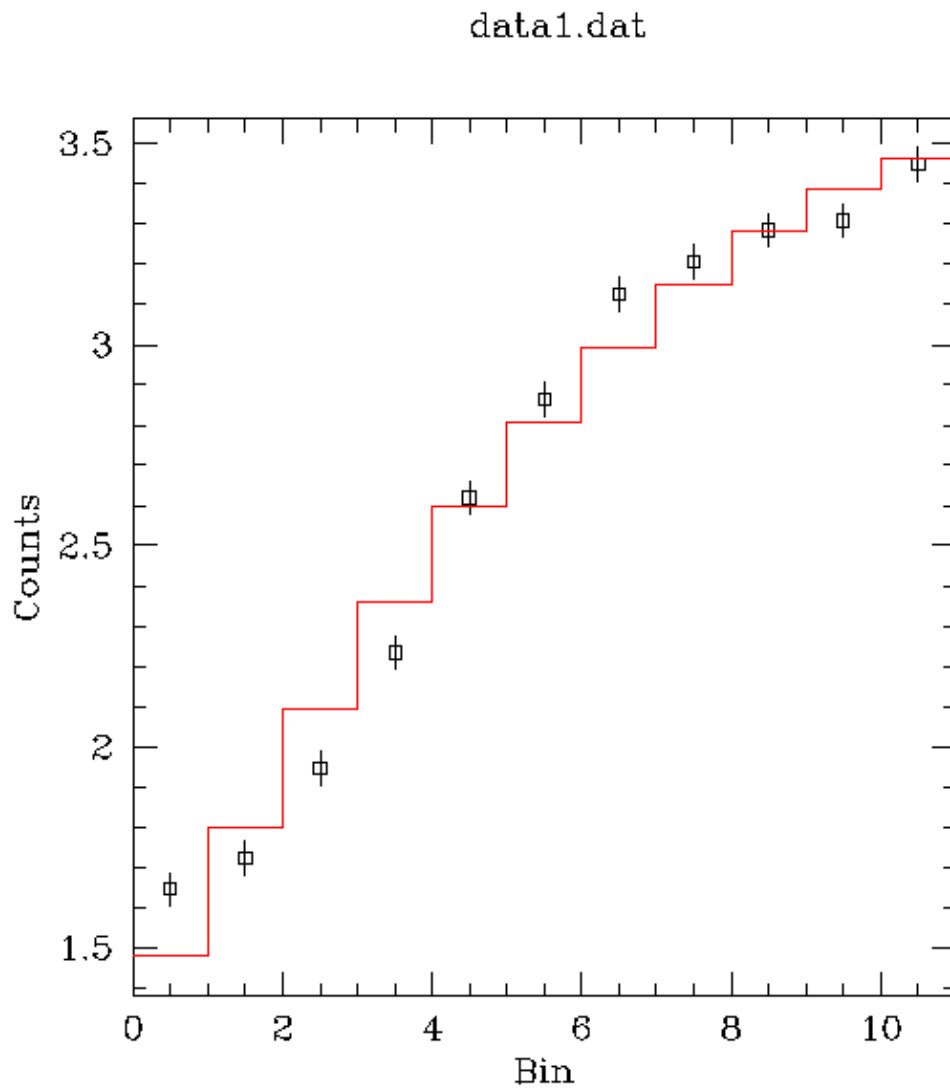
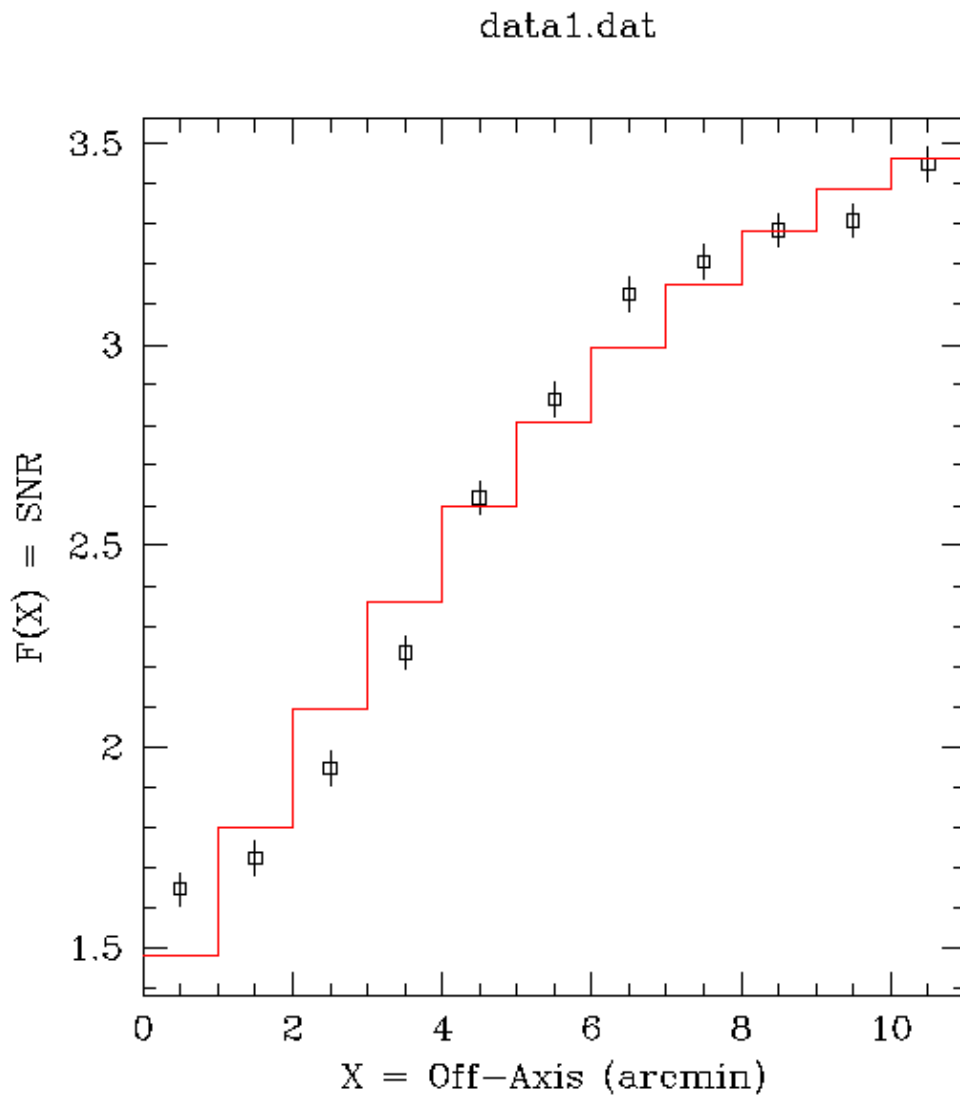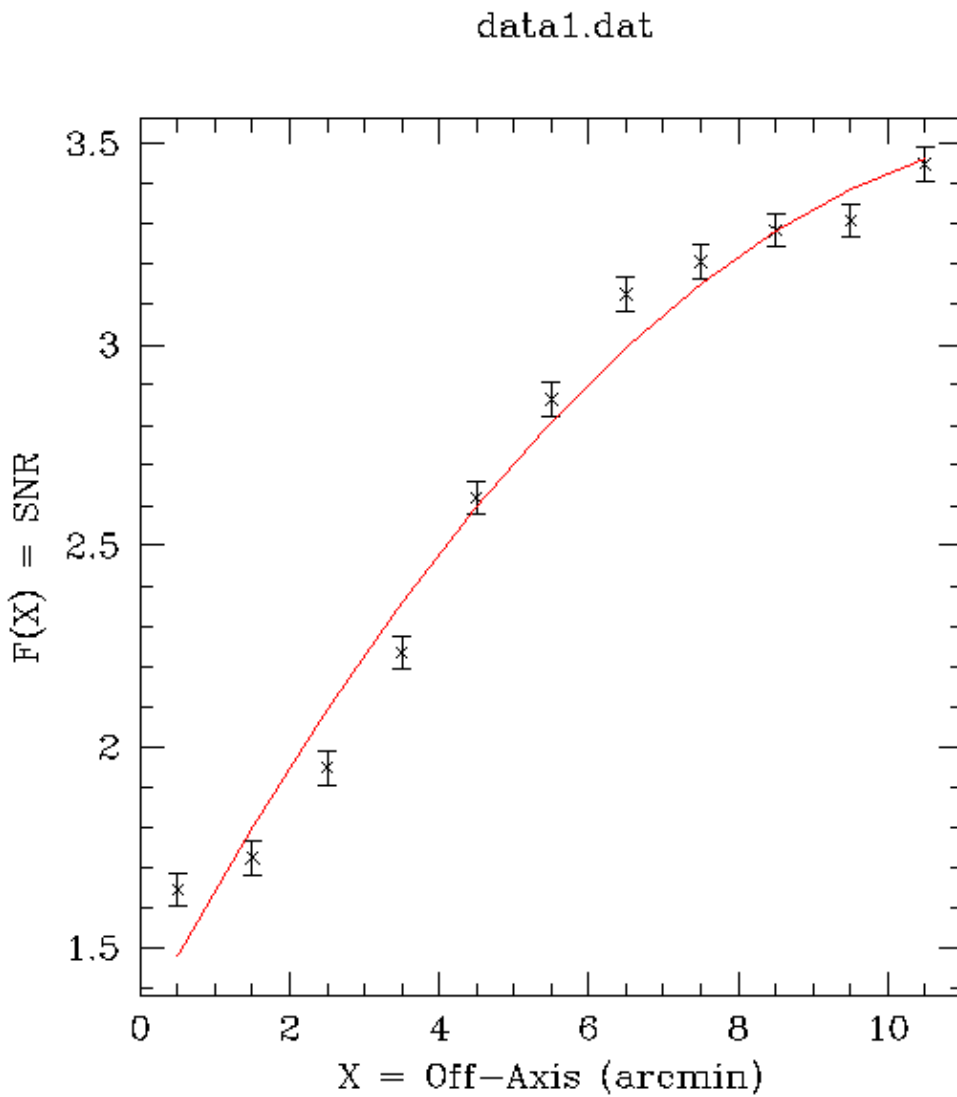## Image 3: Changing the axis labels of a plot

data1.dat

Image 4: Customizing lines and symbols

data1.dat

Image 5: Using multiple windows
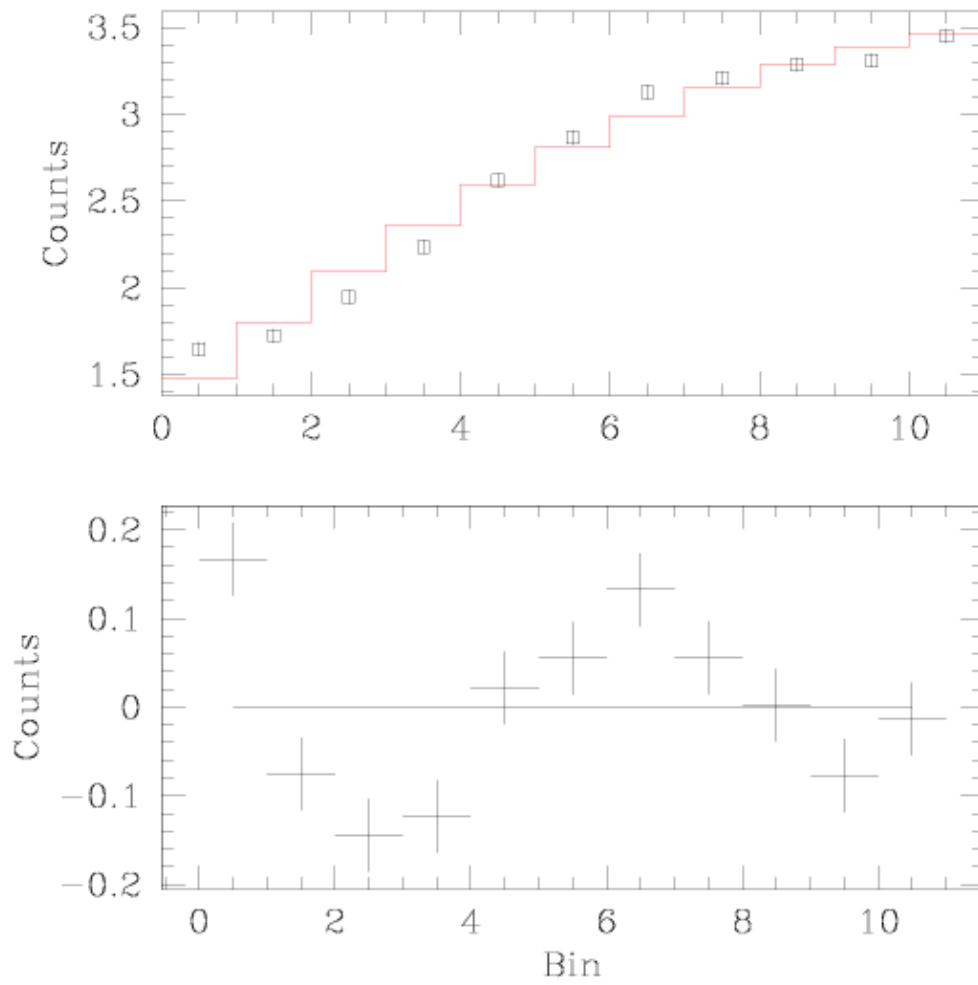
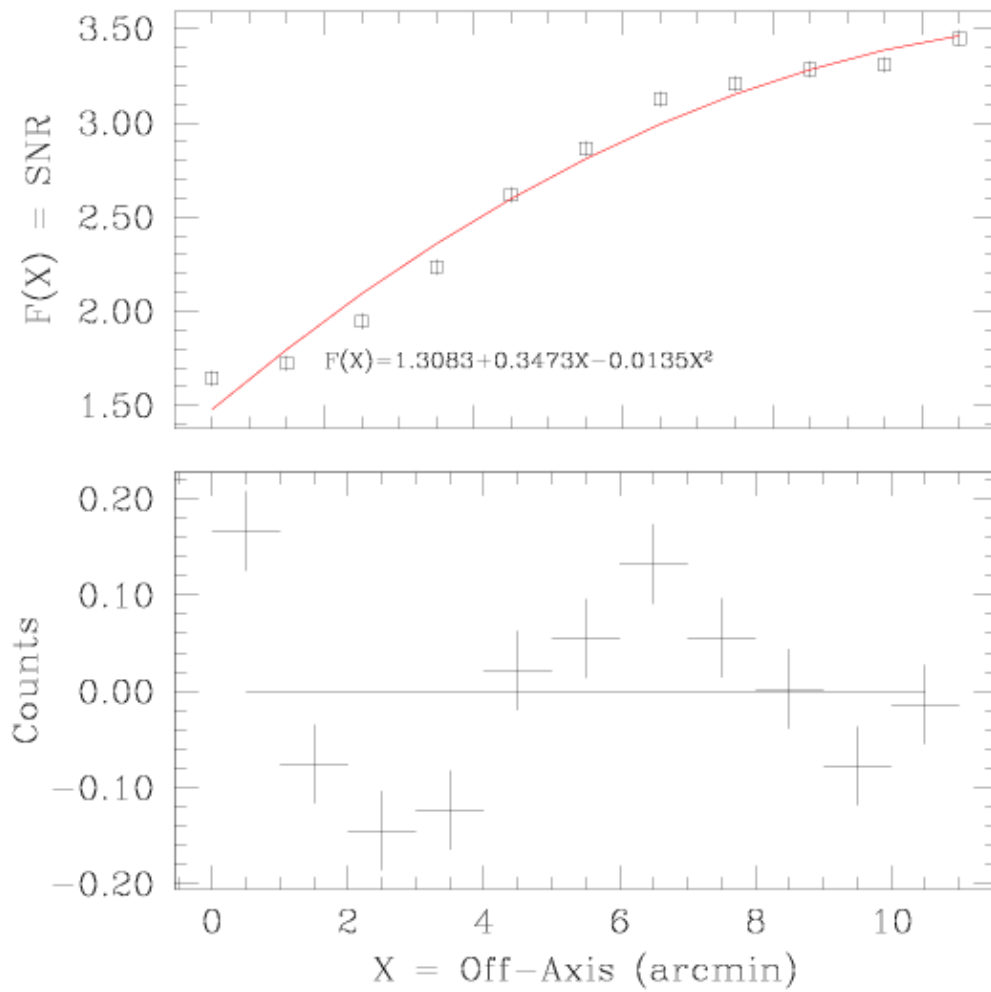# Image 6: Modifying plots in multiple windows

ACIS 25000 Counts Per Chip

$F(X)=1.3083+0.3473X-0.0135X^2$

Image 7: Restoring a plot from a previous Sherpa session

ACIS 25000 Counts Per Chip

$$F(X) = 1.4984 + 0.1447X + 0.0323X^2 - 0.0028X^3$$
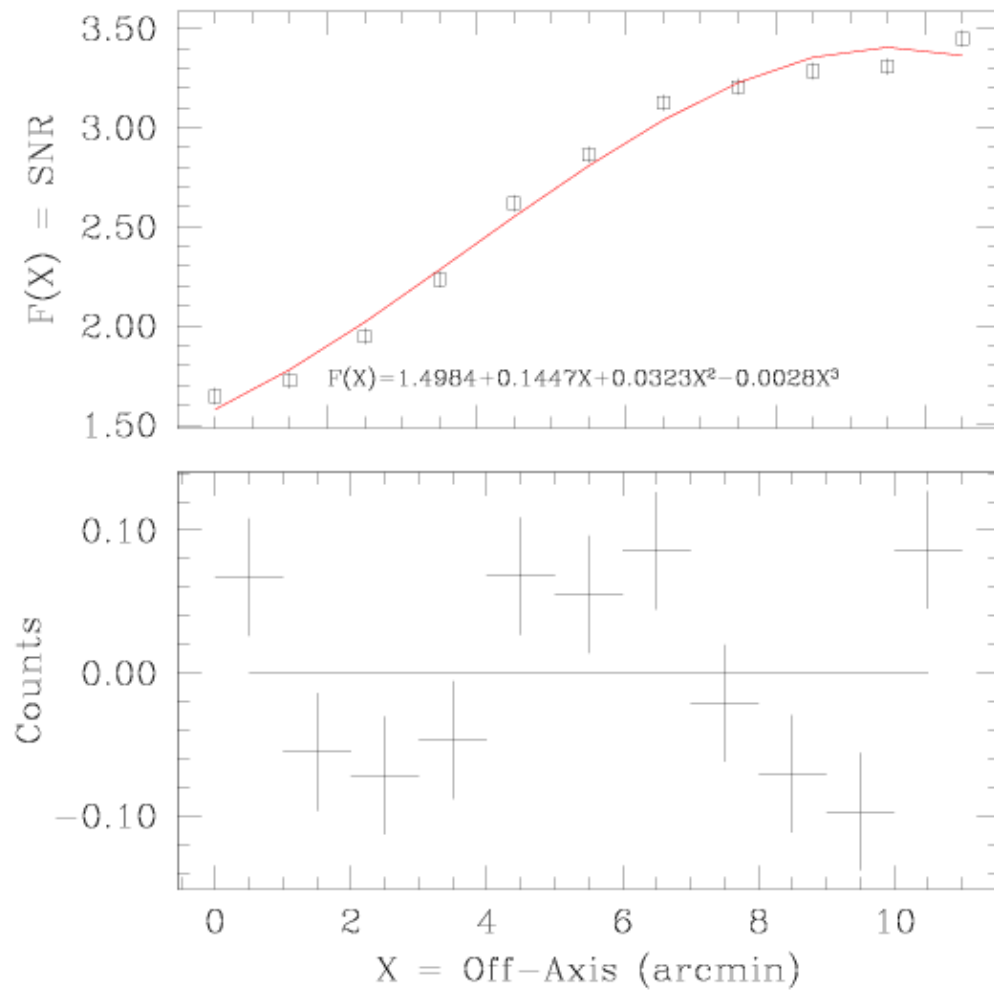
Image 8: Plotting 2 datasets in 2 windows
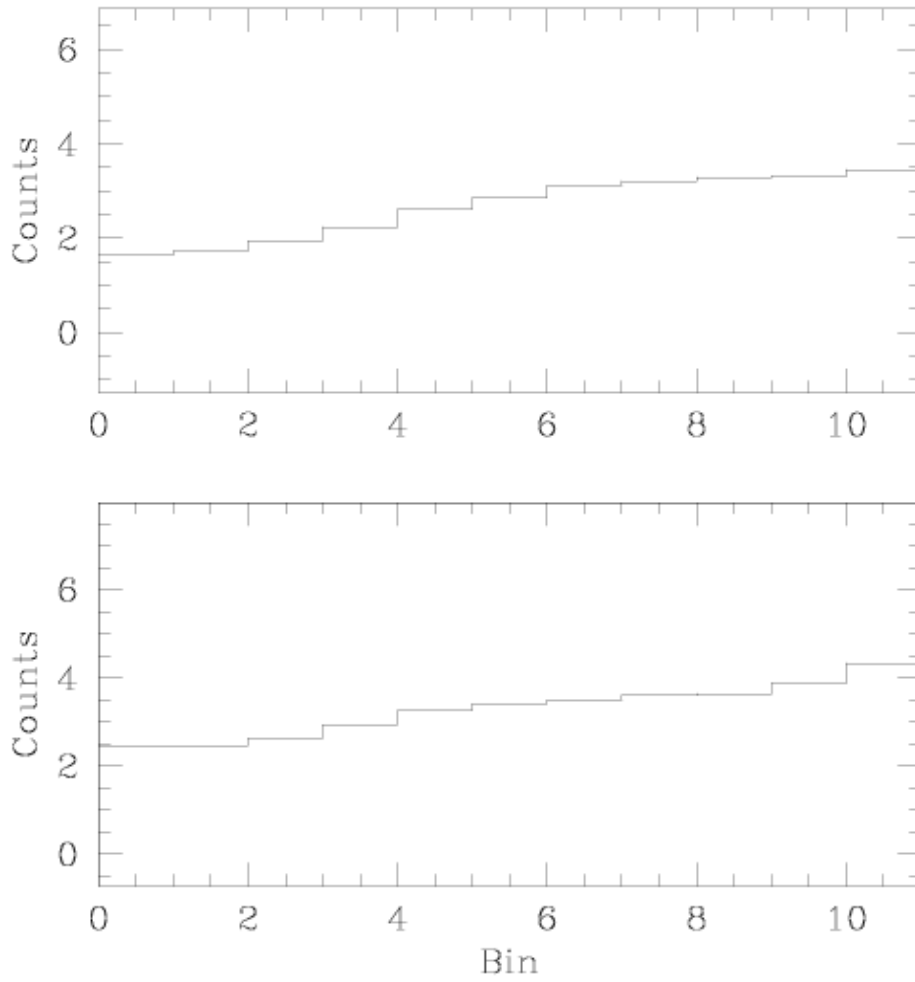
Image 9: Plotting 2 datasets in the same window
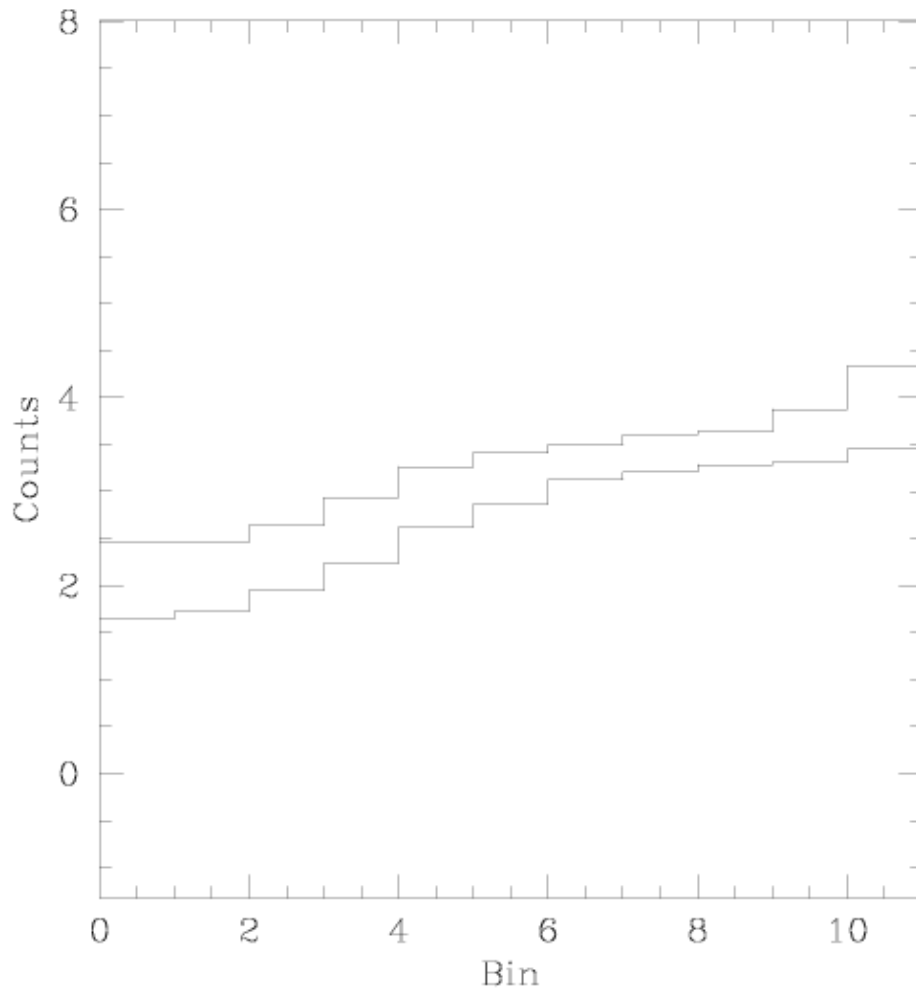
Image 10: Plotting a PHA spectrum in counts

data3.pha
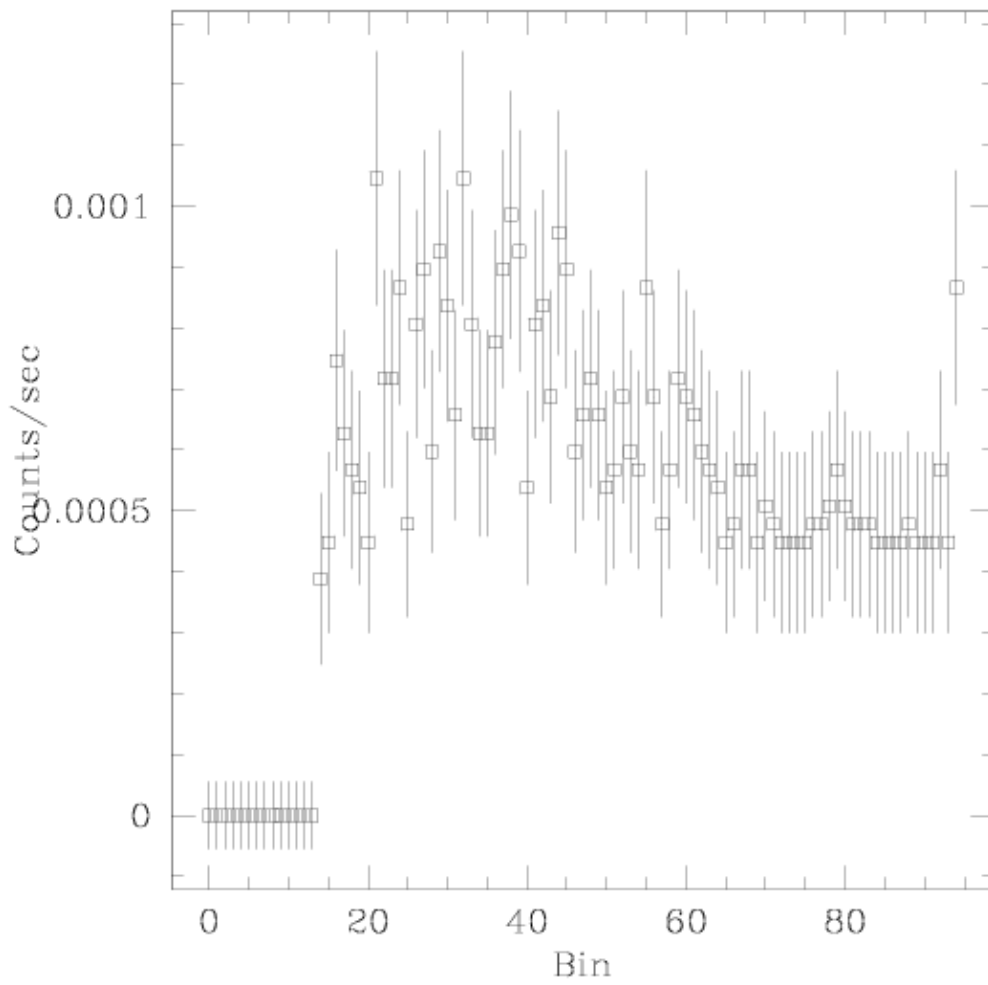
Image 11: Plotting a PHA spectrum in energy

data3.pha
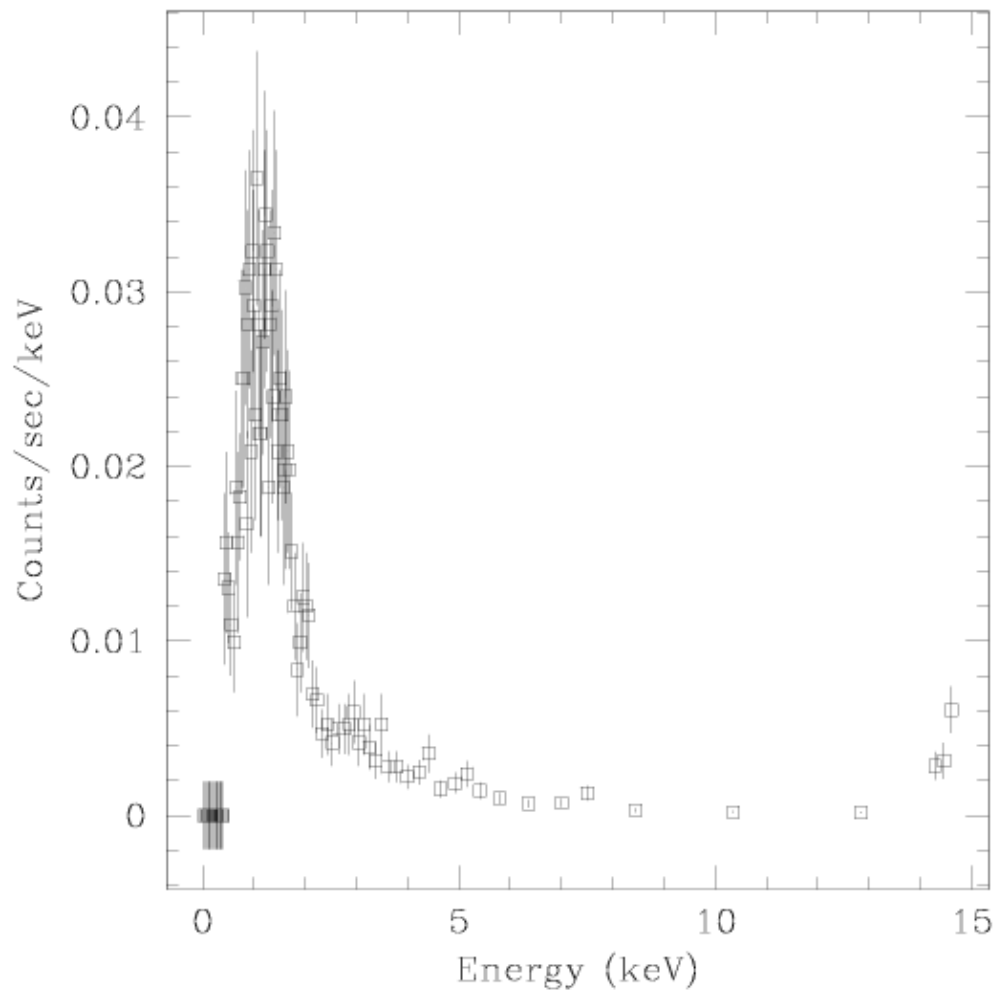
Image 12: Plotting 2 spectra in the same window
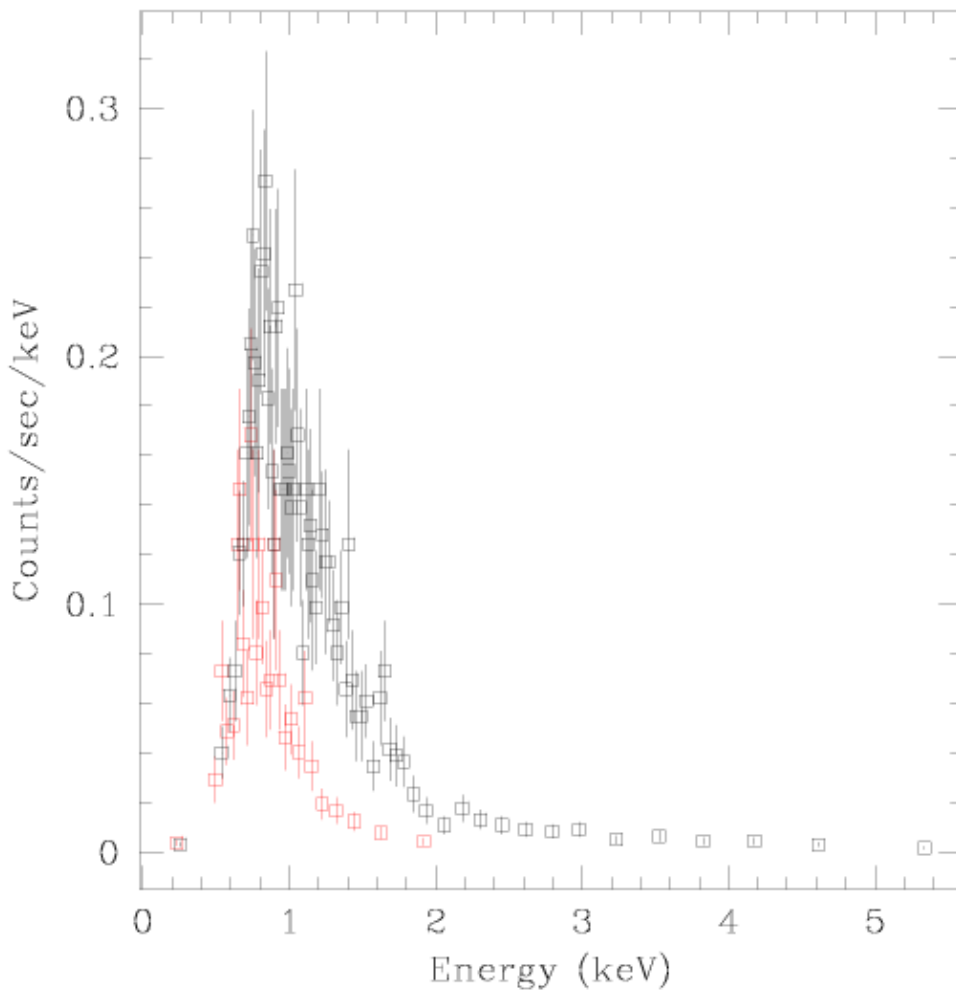
Image 13: Plotting 2 models in the same window
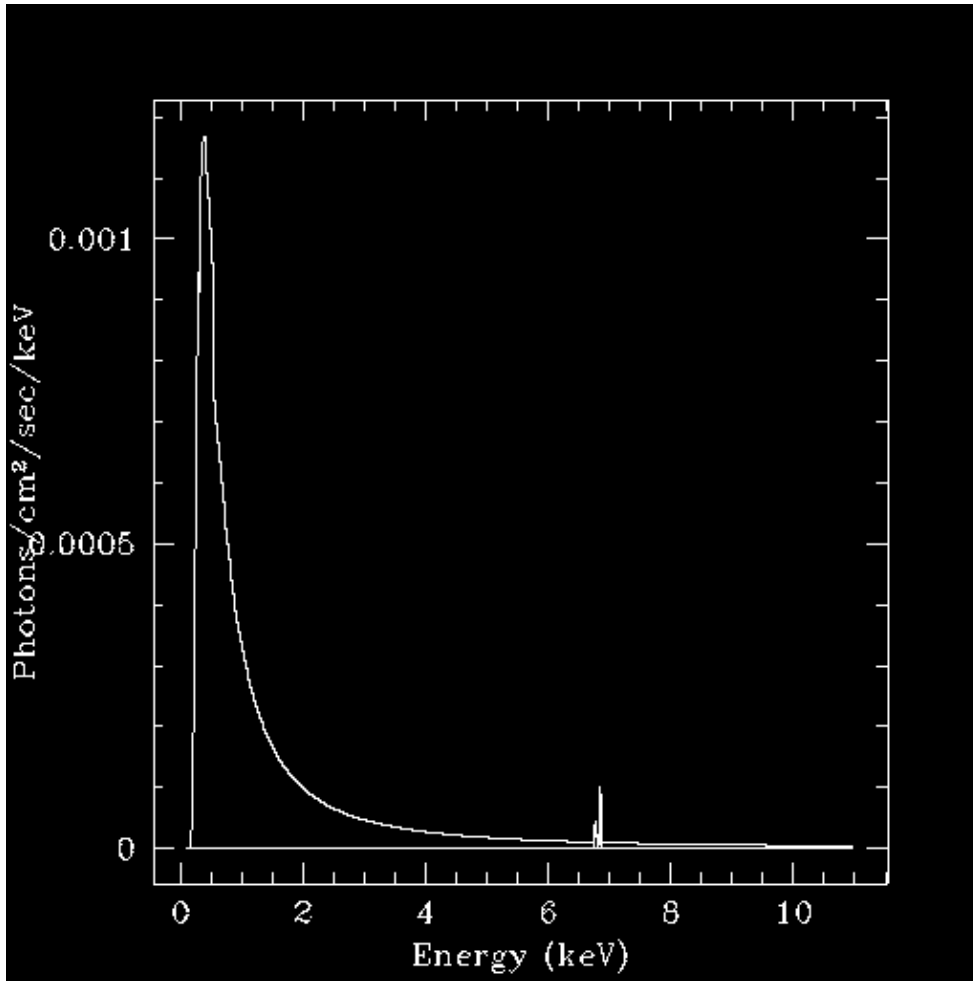
Image 14: Contour plot of 2D data

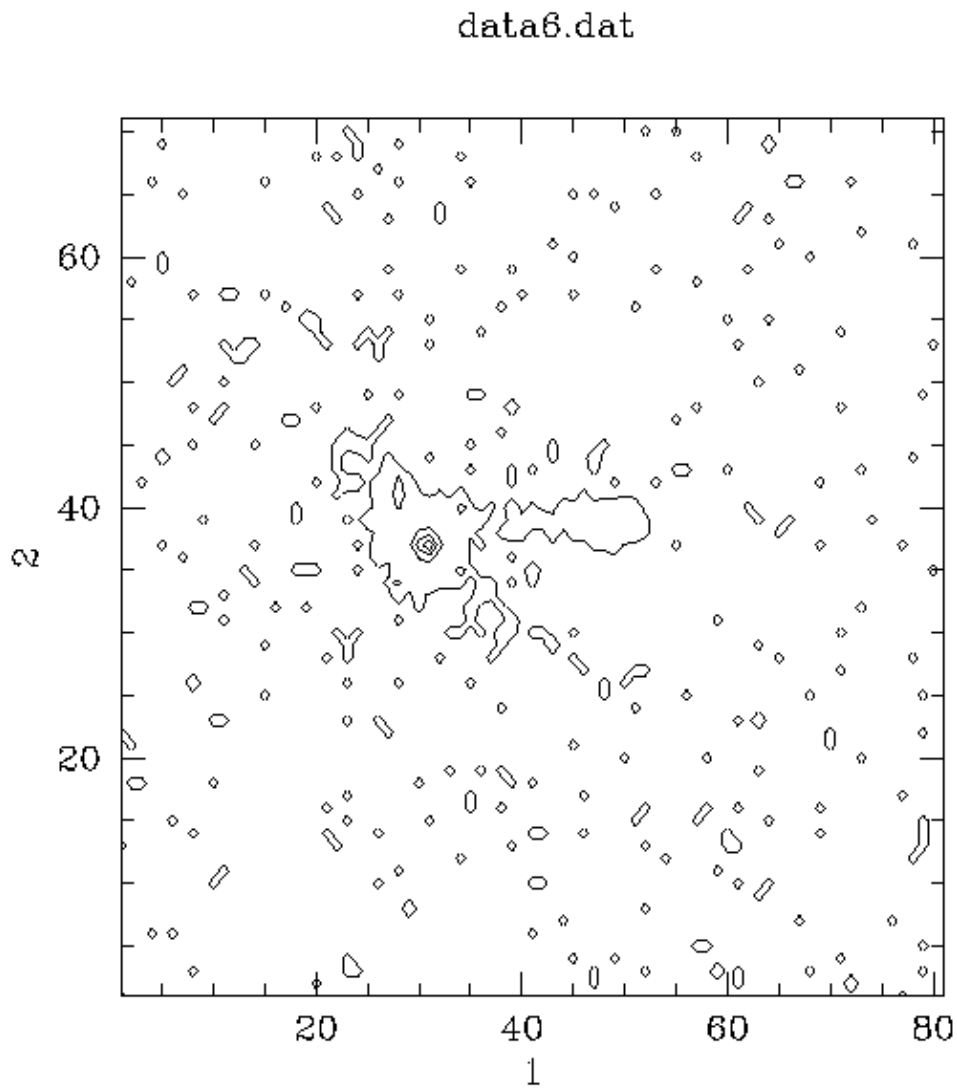data6.dat

Image 15: Surface plot of 2D data

data6.dat
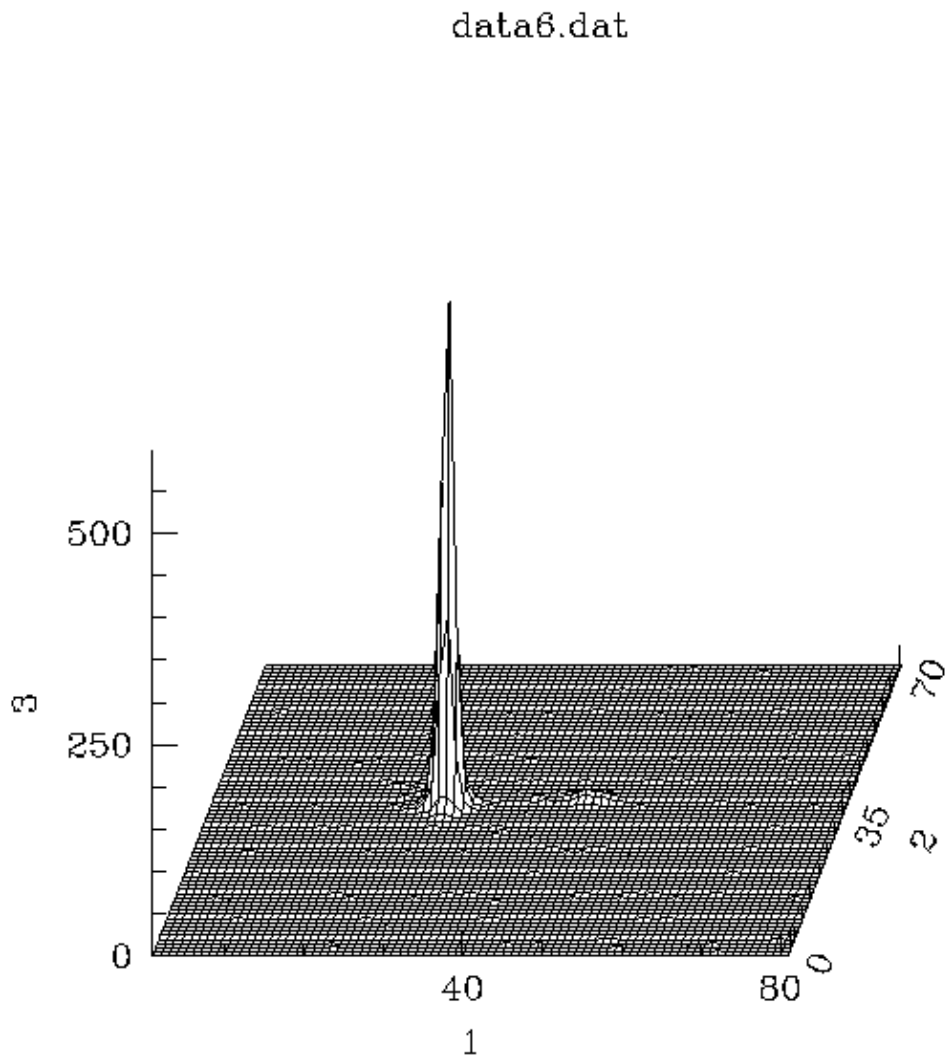
Image 16: Surface plots of two datasets

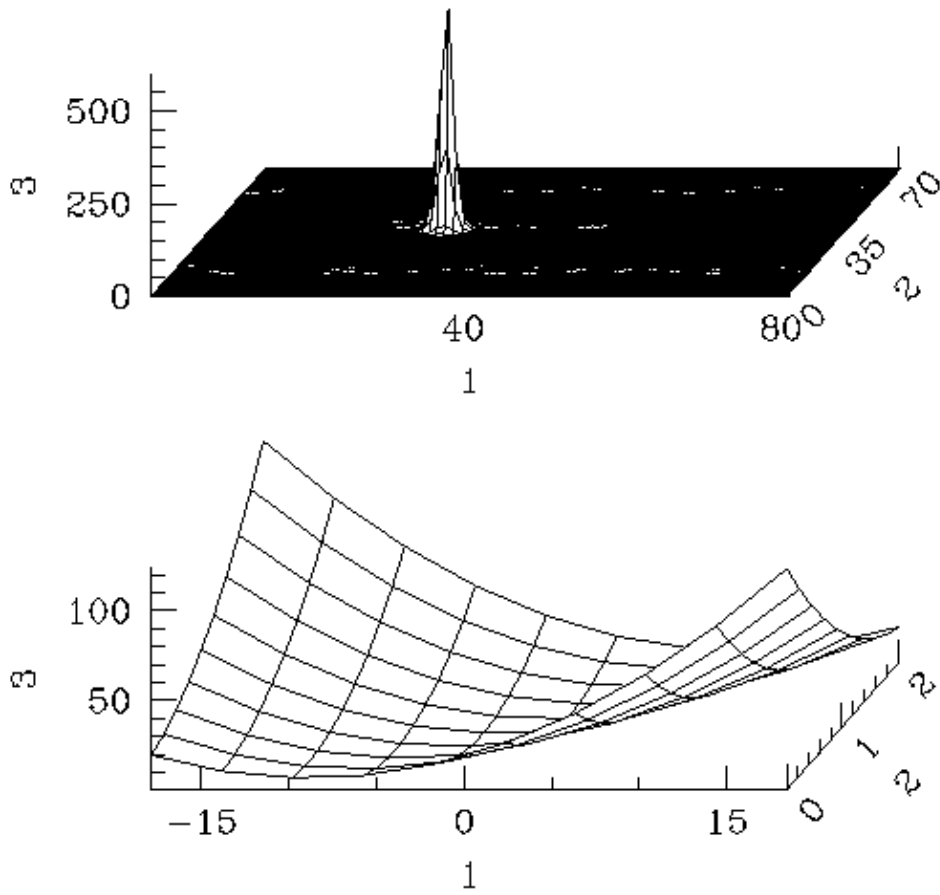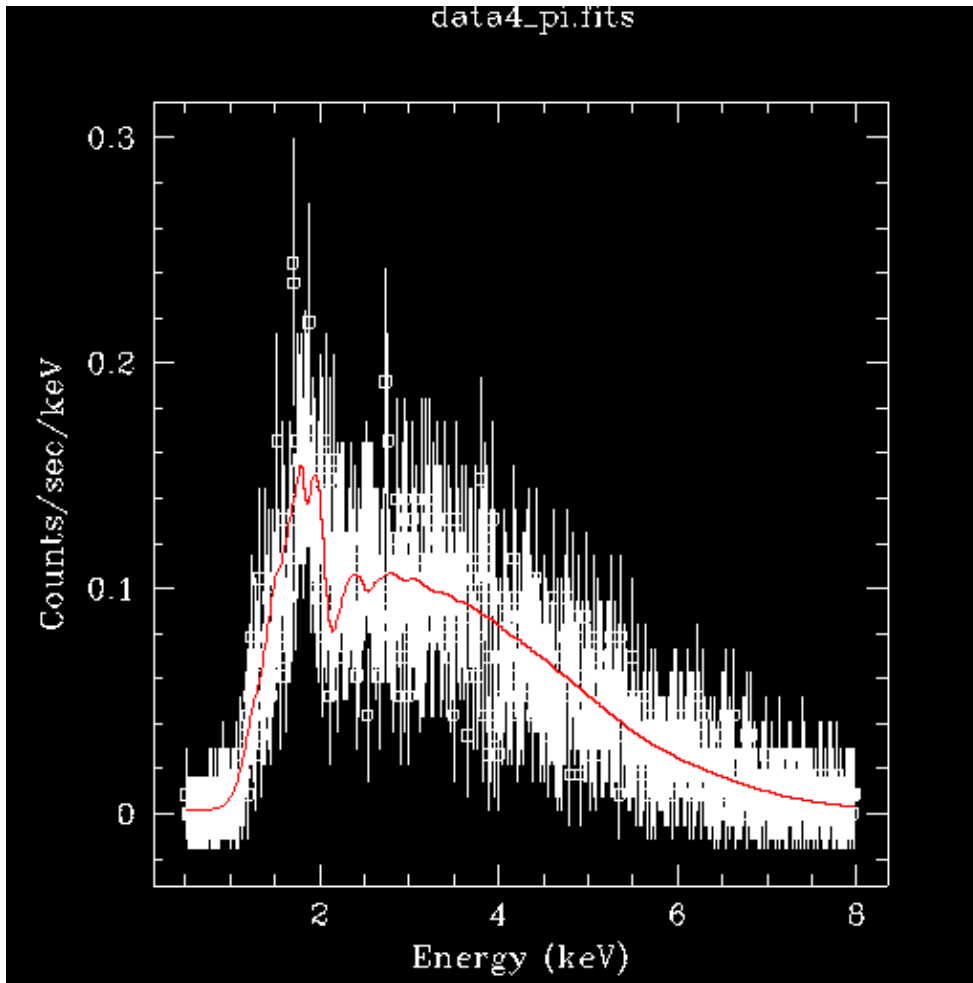**Image 17: 1D dataset with best fit**

Image 17: 1D dataset with best fit 35

Image 18: Confidence countours for the fit parameters

Image 18: Confidence countours for the fit parameters