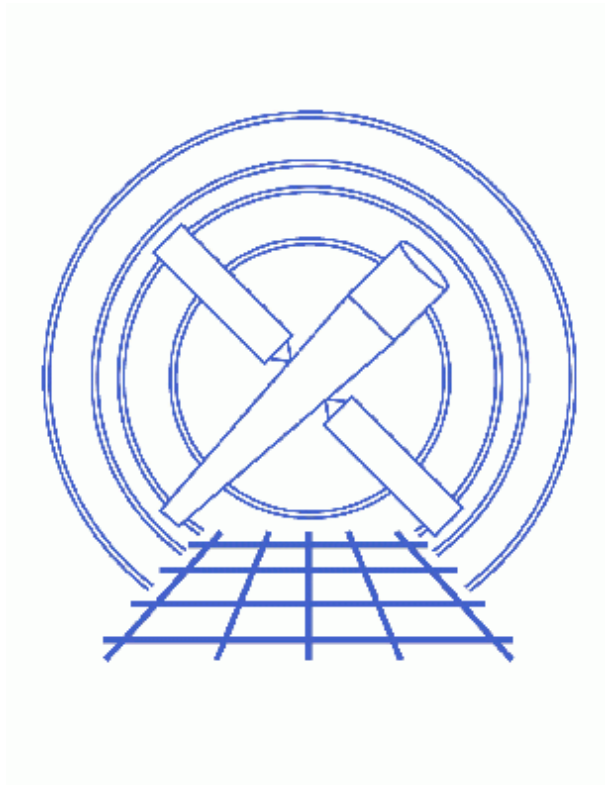


Advanced customization of Sherpa plots



Sherpa Threads (CIAO 3.4)

Table of Contents

- ***Getting Started***
 - ◆ Downloading the data
 - ◆ Downloading the sherpa_plotfns.sl script
 - ◆ Loading the sherpa_plotfns.sl script into Sherpa
- ***Using the new plots***
 - ◆ Fitting the data
 - ◆ Plotting the data
- ***How the customisations work***
 - ◆ Changing sherpa.resplot
 - ◆ Using the pre/postfunc hooks
- ***History***
- ***Images***
 - ◆ The plot produced by LP FIT
 - ◆ The plot produced by LP 2 FIT DELCHI
 - ◆ The plot produced by LP 2 UFIT RATIO
 - ◆ The plot produced by LP 2 ARF RESIDUALS
 - ◆ The plot produced by LP 2 FIT ARF

Advanced customization of Sherpa plots

Sherpa Threads

Overview

Last Update: 1 Dec 2006 – reviewed for CIAO 3.4: no changes

Synopsis:

Sherpa provides a number of plots suitable for data analysis, and the [Data Visualization](#) thread shows you how you can use *ChIPS* commands to modify the plots once they are created.

It is also possible to customise the appearances of these plots automatically using the [plot configuration variables](#) in *Sherpa*. In this thread we provide an example of this functionality by changing the look of "[lplot 2 fit resid](#)" style plots.

Read this thread if:

You want to change the look of a plot every time it is created but the options in the *Sherpa* configuration variables do not provide the control you need.

Related Links:

- The [Data Visualization](#) thread.
- The [Changing Sherpa plots using setplot.sl](#) thread.
- The help documents on the configuration variables (i.e. state objects) of *Sherpa* that control the plots: [sherpa.plot](#), [sherpa.dataplot](#), [sherpa.fitplot](#), [sherpa.resplot](#), and [sherpa.multiplot](#).
- The help document on [pre- and post- hooks](#).

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

Getting Started

Downloading the data

The data used in this thread is available in the [sherpa.tar.gz](#) file, as described in the "[Getting Started](#)" thread.

Downloading the sherpa_plotfns.sl script

The thread uses the `sherpa_plotfns.sl` script; for information about the script, consult the help file ("[ahelp sherpa_plotfns](#)"). The most recent version of `sherpa_plotfns.sl` is version 1.29 (02 Nov 2004):

```
unix% grep Id $ASCDS_CONTRIB/share/slsh/local-packages/sherpa_plotfns.sl
% $Id: sherpa_plotfns.sl,v 1.29 2004/11/02 15:59:14 dburke Exp $
```

Note that `$ASCDS_CONTRIB/share/slsh/local-packages/` is the default path in the standard CIAO scripts installation; see the [Scripts page](#) for more information. **Please check that you are using the most recent version before continuing.** If you do not have the script installed or need to update to a newer version, please refer to the [Scripts page](#).

Loading the sherpa_plotfns.sl script into Sherpa

The `sherpa_plotfns.sl` script is loaded into a *Sherpa* session with the `evalfile` function:

```
sherpa> () = evalfile("sherpa_plotfns.sl");
```

The version may also be found by viewing the `_sherpa_plotfns_version` or `_sherpa_plotfns_version_string` variables once the file is loaded:

```
sherpa> _sherpa_plotfns_version
129
sherpa> _sherpa_plotfns_version_string
1.29
```

If you wish the functions to always be available to *Sherpa* add the following line to your `~/ .sherparc` file:

```
() = evalfile("sherpa_plotfns.sl");
```

For more information on configuring *Sherpa*, see the [Customizing Sherpa with a Resource File](#) thread.

Using the new plots

Once the `sherpa_plotfns.sl` script is loaded then you do not need to do anything: the "fit + residual" style plots will automatically use the new style. In fact, any plot in which the second plot area contains a "residual" plot – so residuals, delchi, or ratio – will use the new settings.

Fitting the data

First we load in a dataset and fit it. For this example we use the same dataset and model as used in the [Estimating Errors and Confidence Levels](#) thread.

```
sherpa> data_source_grouped 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
RMF is being input from:
```

```

/data/ciao/rmf.fits
ARF is being input from:
/data/ciao/arf.fits
sherpa> ignore energy :0.5,8:
sherpa> paramprompt off
Model parameter prompting is off
sherpa> source = xswabs[abs] * powlaw1d[p1]
sherpa> fit
LVMQT: V2.0
LVMQT: initial statistic value = 4583.05
LVMQT: final statistic value = 83.2873 at iteration 8
      abs.nH  2.4061  10^22/cm^2
      p1.gamma  1.51851
      p1.ampl  0.000241434

```

Plotting the data

```

sherpa> set log
sherpa> lp fit

```

We call `set_log` first to ensure that the X and Y axes of the plot are drawn with logarithmic scales, as can be seen in the [plot created by `lplot fit`](#).

```

sherpa> lp 2 fit delchi

```

When we come to plot the [fit with residuals](#) the customisations made in `sherpa_plotfns.sl` can be seen. Note that the y axis of the residual plot is always drawn with a linear scale, even though `set_log` has been used (this is again due to the customisations and is [described below](#)).

The customised form of the plot will occur whenever a residual plot is draw in the second plot: for instance

```

sherpa> lp 2 ufit ratio

```

shows the "unfolded" fit together with the ratio of data to fit. Similarly

```

sherpa> lp 2 arf resid
Warning: negative and zero values ignored in log scale

```

shows the [ARF together with the residuals of the fit](#). These two examples also demonstrate another customisation: the X-axis of the two plots is set to cover the same range. Without this the plots could become confused as shown below:

```

sherpa> lp 2 fit arf
sherpa> d 1 location 0.15 0.9 0.4 0.9
sherpa> d 2 location 0.15 0.9 0.1 0.4
sherpa> redraw

```

which creates [this plot](#), where it is not easily possible to relate features in one plot to those in another because their X-axes do not match.

How the customisations work

There are two parts to the plot customisations:

Advanced customization of Sherpa plots – Sherpa

- changes to the `sherpa.resplot` configuration variable to change the look of the "residuals" plots
- the use of pre- and post- func hooks to further configure the plots

We discuss each part in turn below. Please also review the comments in the script itself.

Changing `sherpa.resplot`

The look of *Sherpa* plots is controlled by the settings of fields in the *Sherpa* state object. These fields are indicated by the **bold** text below:

```
sherpa> print(sherpa)
plot           = sherpa_Plot_State
dataplot      = sherpa_Plot_State
fitplot       = sherpa_FitPlot_State
resplot       = sherpa_Plot_State
multiplot    = sherpa_Draw_State
output         = sherpa_Output_State
regproj        = sherpa_VisParEst_State
regunc         = sherpa_VisParEst_State
intproj        = sherpa_VisParEst_State
intunc         = sherpa_VisParEst_State
proj           = sherpa_Proj_State
cov            = sherpa_Cov_State
unc           = sherpa_Unc_State
con_levs       = NULL
modeloverride  = 0
multiback      = 0
deleteframes   = 1
clobber        = 0
```

and are described in the [sherpa.plot](#), [sherpa.dataplot](#), [sherpa.fitplot](#), [sherpa.resplot](#), and [sherpa.multiplot](#) documentation.

The field to use depends on the plot type you want to change. Here we want to change residual plots so we use `sherpa.resplot` and change it so that there are no error bars drawn along the X axis, that symbols are drawn using the "bigpoint" style (this is a circle whose size can be scaled), and ensure that the symbols are not drawn too large. This is accomplished by the following three lines.

```
sherpa.resplot.x_errorbars = 0;
sherpa.resplot.symbolstyle = "bigpoint";
sherpa.resplot.symbolsize  = 1;
```

Using the pre/postfunc hooks

While the `sherpa.resplot` and related variables allow you to change the look of much of the plots, they do not allow us to make all the changes we want. In order to change the size of the two plots and change the title we use the pre- and post- hooks which allows us to write functions that will be called whenever a plot is about to be created or has just been created.

```
sherpa.resplot.prefunc = &_sherpa_resid_prefunc;
sherpa.resplot.postfunc = &_sherpa_resid_postfunc;
```

These two lines tell *Sherpa* to call the function `_sherpa_resid_prefunc` just before drawing a "residual" plot and the function `_sherpa_resid_postfunc` just after creating the plot. The two fields are normally set to NULL which means that no function will be called. If not NULL then they need to contain a S-Lang *reference*

Advanced customization of Sherpa plots – Sherpa

to a function which is what the syntax "&function_name" does. See the [Referencing and Dereferencing](#) section of the "Guide to the S-Lang Language" for more information.

It is in these two functions that we make the additional changes to create the desired plot. A simplified listing of the two functions follows; see the script for the full version.

The function called before the residual plot has been created is:

```
%%% A - define the function
%
static define _sherpa_resid_prefunc(data, label, dataset, type) {

  %%% B - check status
  %
  if ( 2 != chips_get_pane ) return;

  %%% C - set up the title
  %
  variable model = get_source_expr(dataset);
  variable fitres = get_goodness(dataset);
  variable title;
  if ( NULL != fitres.rstat )
    title = sprintf( "title '%s \\chi^2 r= %f'", model, fitres.rstat );
  else
    title = sprintf( "title '%s'", model );
  () = chips_eval( title );

  %%% D - change the plot locations
  %
  () = chips_eval( "d 1 location 0.15 0.9 0.4 0.9" );
  () = chips_eval( "d 2 location 0.15 0.9 0.1 0.4" );

  %%% E - change the X axis of the first plot
  %
  () = chips_set_pane(1);
  () = chips_eval( "tickvals_x_off" );

  %%% F - ensure we are back in the correct pane
  %
  () = chips_set_pane(2);

}
```

A – define the function

The function can have any name and we chose to mark it as `static` so that it is not easily accessible from outside this file (see the [Namespaces](#) section of the S-Lang Guide for more information).

B – check status

This function will only alter the plot if the residual plot is in the second "pane" (also called "drawing area") of *ChIPS*. This ensures that "`lp resid`" will still behave sensibly; however it does not catch the cases when you have more than two plots, such as "`lp 3 fit resid arf`".

C – set up the title

Here we change the title of the plot to be the source expression and – if the statistic used is based on chi square – the reduced chi-square value of the fit. See the [Accessing fit results using S-Lang](#) thread for more information.

Since there is currently no S-Lang function in *ChIPS* to set the plot title, we have to create a string

Advanced customization of Sherpa plots – Sherpa

containing the *ChIPS* command and evaluate it using `chips_eval()`.

D – change the plot locations

This section performs the main part of the customization: it changes the size and location of the two panes so that the first plot takes up a larger part of the window. Again we use `chips_eval()` to execute a set of *ChIPS* commands.

E – change the X axis of the first plot

Since the two plots now meet – i.e. have a common X axis – we want to remove the axis values from the first plot, otherwise they would appear in the top part of the residual plot. We do this with the `tickvals` command.

F – ensure we are back in the correct pane

Since this function is called *before* the plot is created we need to ensure that we leave the selected pane (drawing area) as we left it; in this case to number 2.

The function called after the residual plot has been created is:

```
%%% A - define the function
%
static define _sherpa_resid_postfunc() {

  %%% B - check status
  %
  if ( 2 != chips_get_pane ) return;

  %%% C - set the y axis of the residual plot
  %
  () = chips_set_yscale(1);

  %%% D - what are the X axes ranges of the first two plots?
  %
  () = chips_set_pane(1);
  variable xs1 = chips_get_xscale;
  variable xlo1, xhi1;
  ( xlo1, xhi1 ) = chips_get_xrange();
  if ( 0 == xs1 ) {
    xlo1 = 10^xlo1;
    xhi1 = 10^xhi1;
  }

  () = chips_set_pane(2);
  variable xs2 = chips_get_xscale;
  variable xlo2, xhi2;
  ( xlo2, xhi2 ) = chips_get_xrange();
  if ( 0 == xs2 ) {
    xlo2 = 10^xlo2;
    xhi2 = 10^xhi2;
  }

  %%% E - do the X axes ranges match?
  %
  variable xlo = max([xlo1,xlo2]);
  variable xhi = min([xhi1,xhi2]);
  () = chips_set_pane(1);
  () = chips_set_xrange(xlo,xhi);
  () = chips_set_pane(2);
  () = chips_set_xrange(xlo,xhi);
  () = chips_set_xscale(xs1);
}
}
```


A – define the function

This is similar to the "pre" function, except that this function is not sent any arguments.

B – check status

Again we only want to make any changes if the residual plot is the second plot in the series.


C – set the y axis of the residual plot

As it generally does not make sense to plot the residuals on a logarithmic scale we ensure that the Y axis is linear. This means that we are effectively ignoring the value of the `sherpa.resplot.y_log` field. This is done so that people can use the `set_log` or `set_ylog` command and still get a linear scale on the residuals plot.

D – what are the X axes ranges of the first two plots?

Here we find out what the limits are on the X axes of the first two plots. Note that this is slightly complicated because the `chips_get_xrange()` function returns the logarithm of the limits if the axis scale is logarithmic. Hence we have to check for this condition and change the values accordingly.

E – do the X axes ranges match?

Here we ensure that the two plots have a common X-axis range by setting both of them to the minimum overlap between `xlo1-xhi1` and `xlo2-xhi2`. This is all to avoid plots looking like [this](#) .

We also force the X axis of the residual plot to have the same scaling – i.e. logarithmic or linear – as the main plot. In most plots this is likely not to make a difference (since they will already match).

History

14 Dec 2004 updated for CIAO 3.2: script version and path

21 Dec 2005 reviewed for CIAO 3.3: no changes

01 Dec 2006 reviewed for CIAO 3.4: no changes

URL: <http://cxc.harvard.edu/sherpa/threads/slangplot/>

Last modified: 1 Dec 2006

Image 1: The plot produced by LP FIT

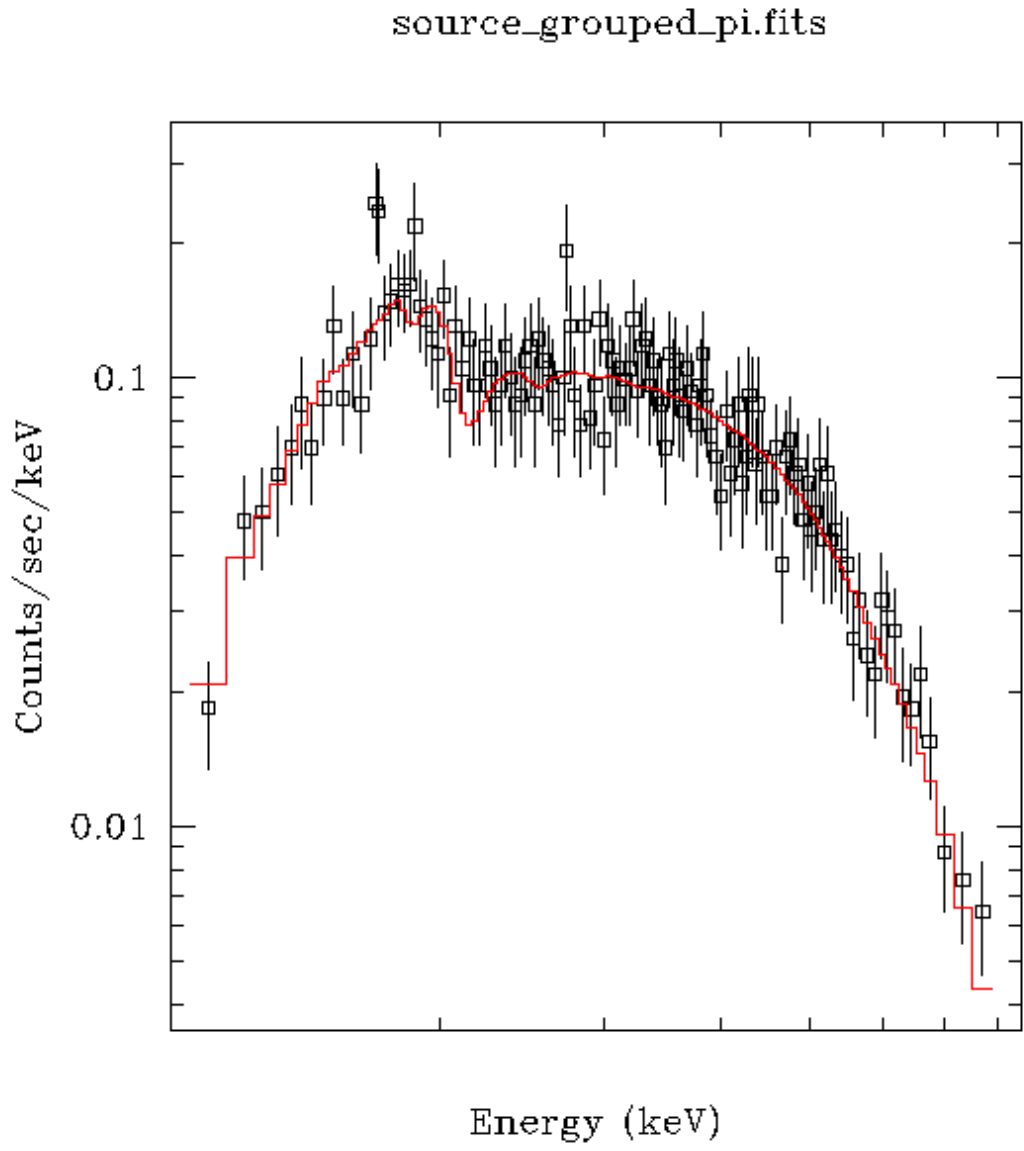
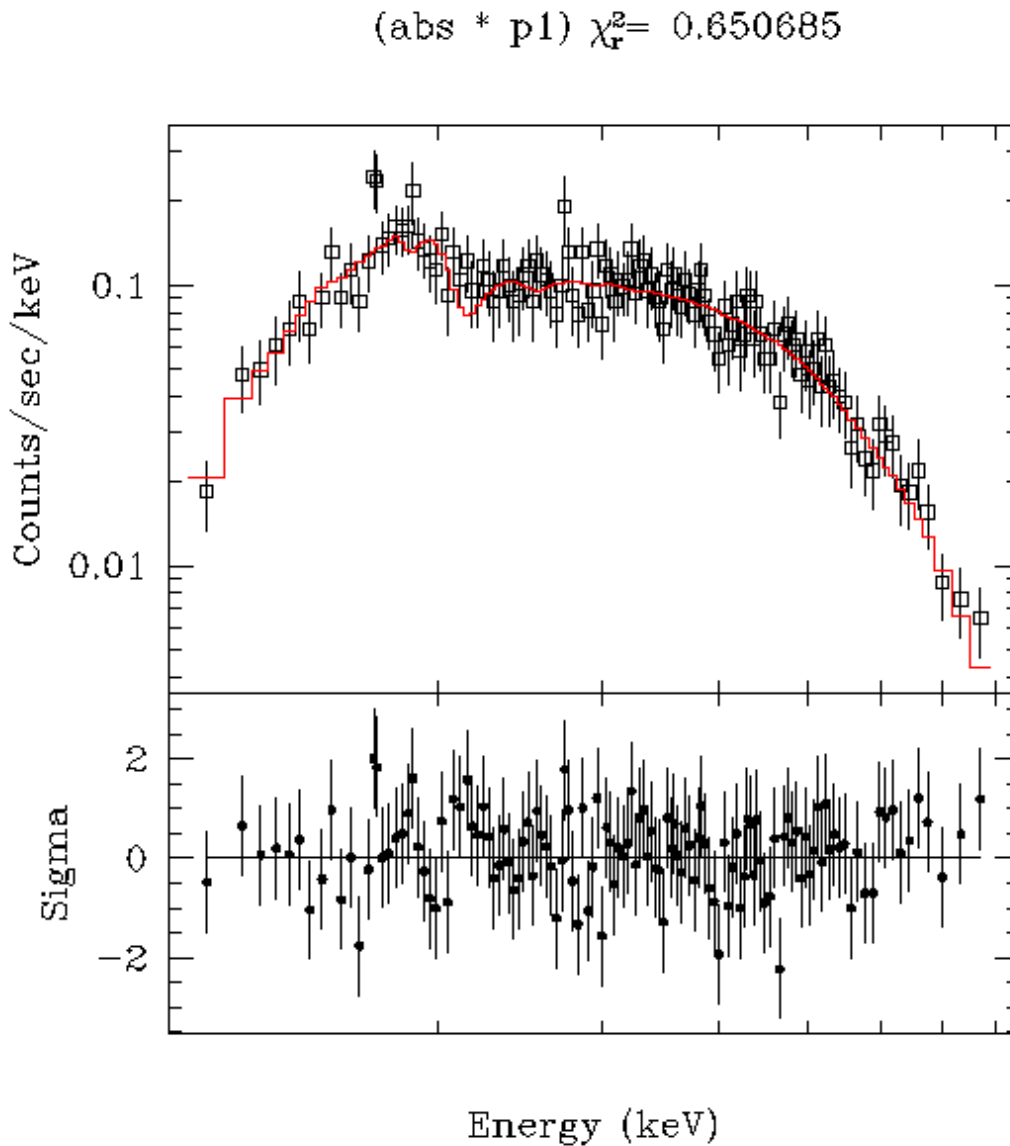
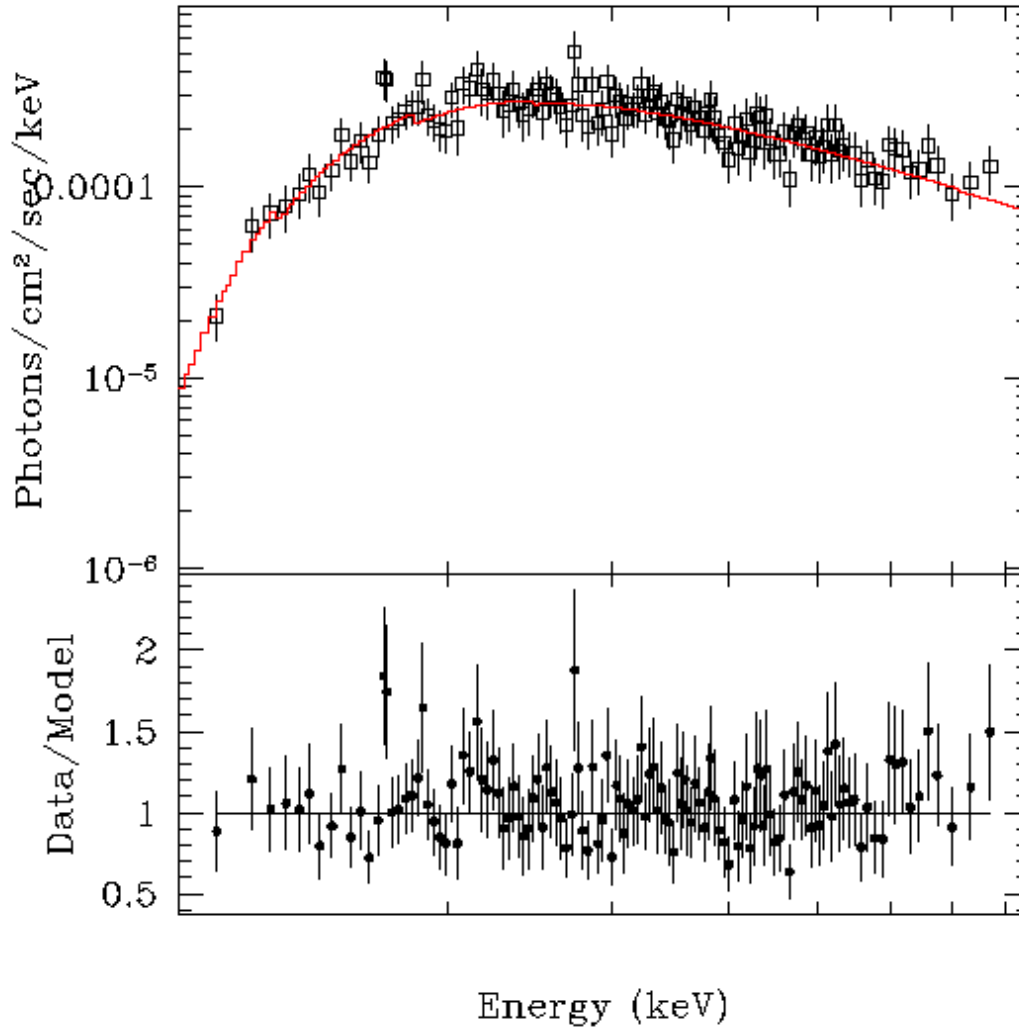


Image 2: The plot produced by LP 2 FIT DELCHI

The customisations made in `sherpa_plotfns.sl` have changed the plot title – it includes the source model and reduced chi-square statistic (when appropriate) – and has had the spacing and size of the two plots changed to give greater prominence to the fit results. Also notice that the y axis of the residual plot is in linear spacing even though the fit results have logarithmic axes.

Image 3: The plot produced by LP 2 UFIT RATIO

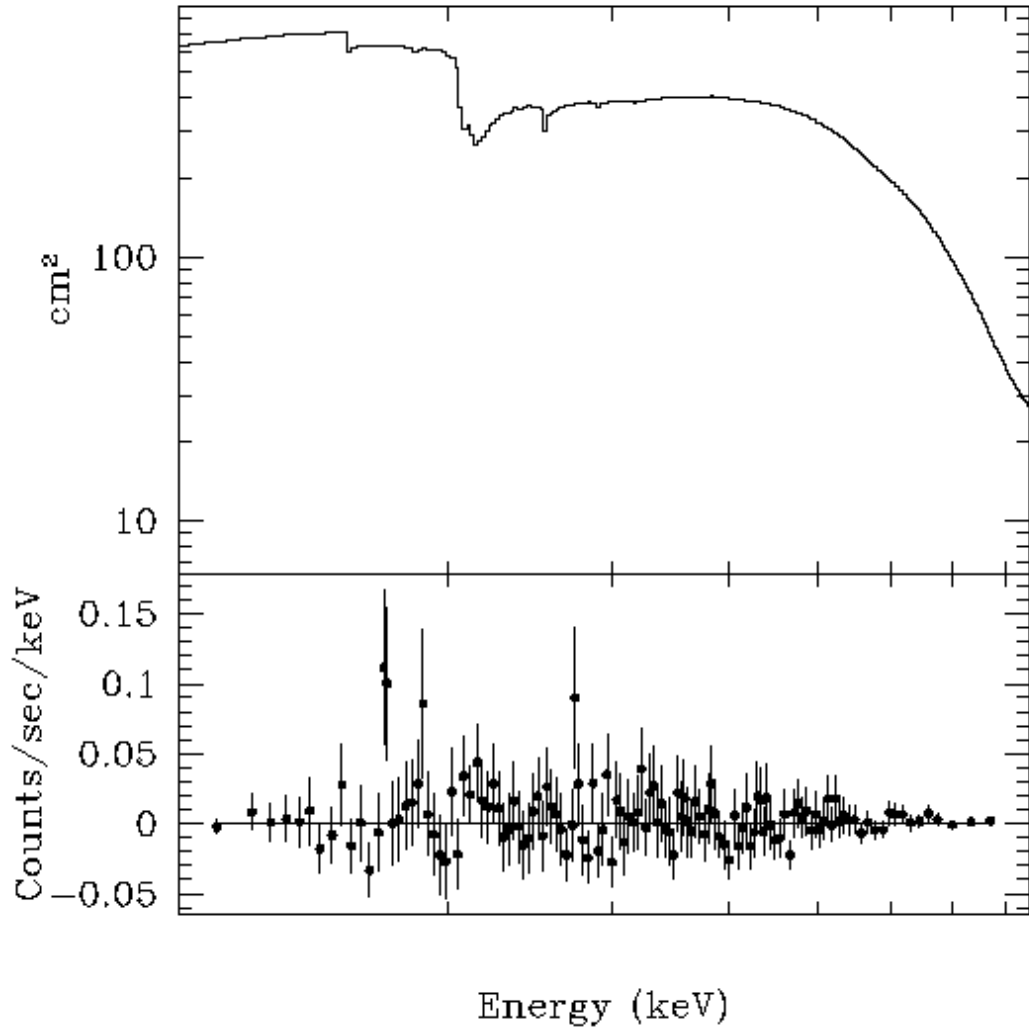
(abs * p1) $\chi^2_r = 0.650685$



Here we plot the "unfolded" fit and the ratio of the data to the fit. The two graphs have been adjusted to have a common X-axis range by the script.

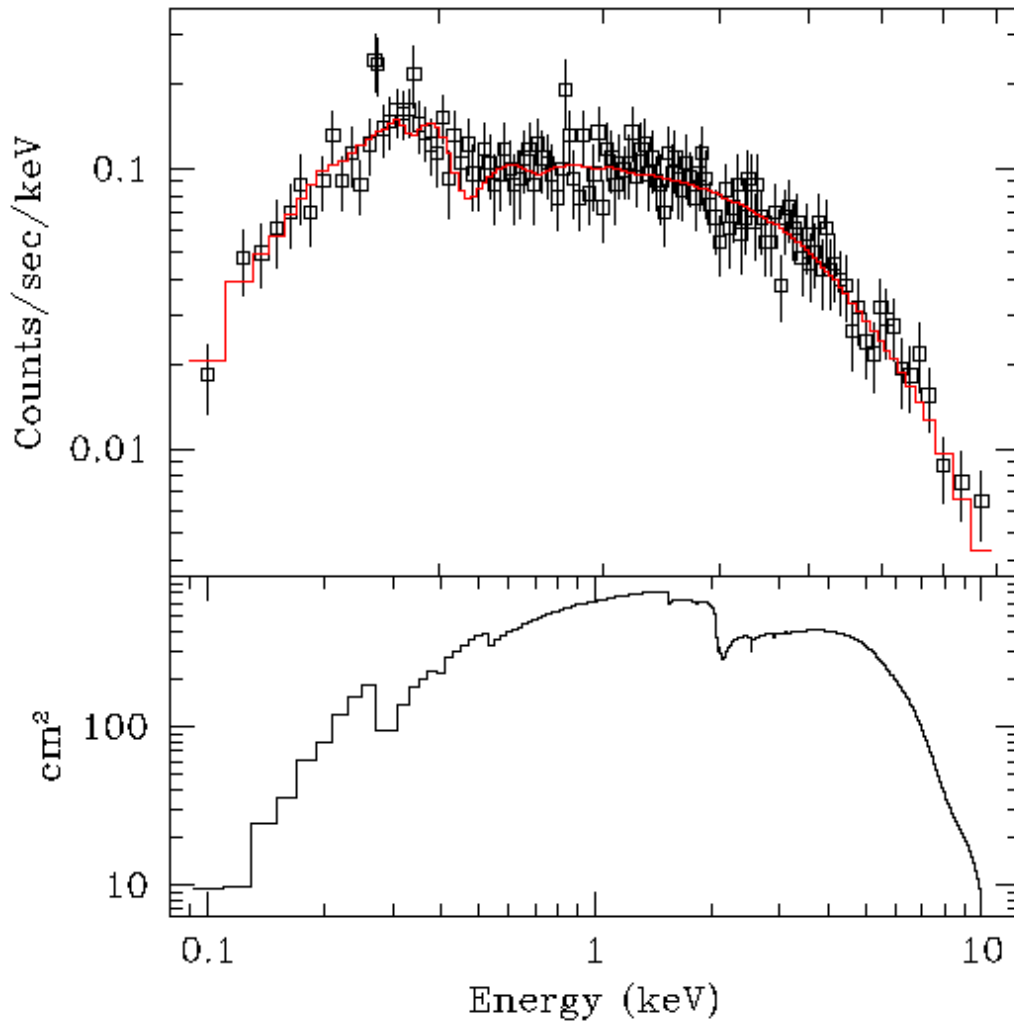
Image 4: The plot produced by LP 2 ARF RESIDUALS

(abs * p1) $\chi^2_r = 0.650685$



As with `lp 2 ufit ratio` the two graphs have been adjusted to have a common X-axis range.

Image 5: The plot produced by LP 2 FIT ARF



Here we show the confusion that can occur if the graphs are not adjusted to have a common X-axis range.