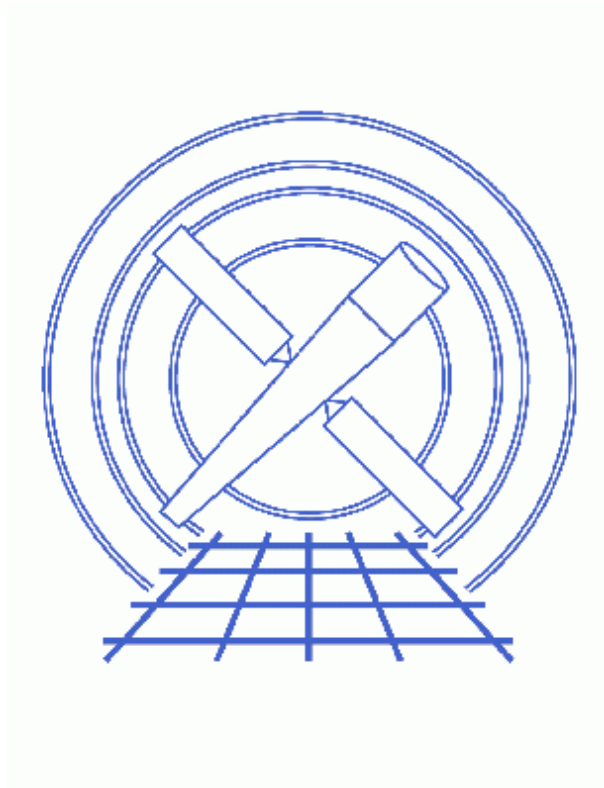


# Step-by-Step guide to changing the look of Sherpa plots



## Sherpa Threads (CIAO 3.4)

# Table of Contents

- *Getting Started*
- *Background Information*
- *Changing "DATA" plots*
- *Changing the "ARF" plot to match*
- *Changing the "FIT" plots*
- *Making global changes*
  - ◆ Existing Sherpa functions
  - ◆ Writing your own functions
- *Saving your changes*
- *Summary*
- *History*
- *Images*
  - ◆ The plot produced by LP DATA with the default settings
  - ◆ The plot produced by LP DATA after changing the settings
  - ◆ The plot produced by LP DATA ARF
  - ◆ Applying a similar customisation to ARF and DATA plots
  - ◆ LP MODEL is also changed
  - ◆ The plot produced by LP FIT

# Step-by-Step guide to changing the look 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 show how you can use these variables to change the default look of your plots.

### *Read this thread if:*

You want to change the look of a plot every time it is created.

### *Related Links:*

- The [Changing the look of Sherpa plots using setplot.sl](#) thread shows how you can make the same changes to your plots using a contributed script.
- The [Data Visualization](#) 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 [Advanced customization of Sherpa plots](#) thread shows how you can use the [pre- and post- hooks](#) in the configuration variables to provide almost total control over the look of the plot.
- The [save\\_state\(\)](#) command for saving any changes you have made to the *Sherpa* configuration variables.

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

---

## Getting Started

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

## Changing Sherpa Plots: Step-by-Step – Sherpa

For the various plots discussed below 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> source = xswabs[abs] * powlaw1d[p1]
abs.nH parameter value [0.1]
p1.gamma parameter value [1]
p1.ref parameter value [4]
p1.ampl parameter value [0.000149261]
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
```

---

## Background Information


Plots are created in *Sherpa* by the [L\\_PLOT](#) command.

Many aspects of the appearance of the plots can be changed by setting values in the *Sherpa* configuration variables (state objects). This is discussed in the [Configuration of Sherpa with Sherpa State Objects](#) section of `ahelp sherpa`, and the variables relevant to plotting are: [sherpa.plot](#), [sherpa.dataplot](#), [sherpa.fitplot](#), [sherpa.resplot](#), and [sherpa.multiplot](#).

By changing the values of the fields in these variables you change how new plots of the given type will be drawn. There are also several utility routines – such as [set log](#), [set lin](#), [set erron](#), and [set erroff](#) to change some of the common options.

---

## Changing "DATA" plots

The default settings for "DATA" plots produces [this plot](#)  for the dataset used in this example. The look of the plot is controlled by the fields of the [sherpa.dataplot](#) plotting variable. These can be viewed using the `print()` command:


```
sherpa> print(sherpa.dataplot)
x_errorbars      = 0
y_errorbars      = 1
errs_style       = bar
errs_type        = both
```

## Changing Sherpa Plots: Step-by-Step – Sherpa

```
x_log          = 0
y_log          = 0
curvestyle     = noline
curvecolor     = default
symbolstyle    = square
symbolcolor    = default
symbolsize     = 2
xlabel_size    = 1.5
ylabel_size    = 1.5
zlabel_size    = 1.5
title_size     = 1.5
tickvals_size  = 1.5
prefunc        = NULL
postfunc       = NULL
```


We decide we want the y axis to be drawn in a logarithmic scale and the symbols to be drawn in magenta whilst keeping the remaining options unchanged.

```
sherpa> sherpa.dataplot.y_log = 1
sherpa> sherpa.dataplot.symbolcolor = "magenta"
sherpa> print(sherpa.dataplot)
x_errorbars    = 0
y_errorbars    = 1
errs_style     = bar
errs_type      = both
x_log          = 0
y_log         = 1
curvestyle     = noline
curvecolor     = default
symbolstyle    = square
symbolcolor   = magenta
symbolsize     = 2
xlabel_size    = 1.5
ylabel_size    = 1.5
zlabel_size    = 1.5
title_size     = 1.5
tickvals_size  = 1.5
prefunc        = NULL
postfunc       = NULL
sherpa> lplot data
```

The new version of the plot looks [like this](#) . Note that the color is specified as a string, unlike the `chips.symbolcolor` field of the *ChIPS* state object which takes an integer.

We now decide that we want to make further changes to the plot:

```
sherpa> sherpa.dataplot.x_log = 1
sherpa> sherpa.dataplot.curvestyle = "step"
sherpa> sherpa.dataplot.curvecolor = "cyan"
sherpa> sherpa.dataplot.symbolstyle = "none"
sherpa> sherpa.dataplot.title_size = 2
sherpa> lplot 2 data arf
```

The resulting plot looks [like this](#) . The top plot has been changed to match the settings asked for, whilst the bottom plot – of the ARF – does not use any of these settings, because the ARF plot is controlled by the `sherpa.plot` plot variable.


Although the plot title was changed to a size of 2, from its default value of 1.5, the new plot does not have a title. This is because we draw two plots here and *Sherpa* only sets the title when an individual plot – i.e. "lplot data" – is created.

---

## Changing the "ARF" plot to match


In the previous section we changed *Sherpa* so that all "DATA" plots have a garish color scheme. However, other plots – such as of the ARF – do not share this scheme. We can set the fields in the `sherpa.plot` plot variable to change this:

```
sherpa> sherpa.plot.x_log = 1
sherpa> sherpa.plot.y_log = 1
sherpa> sherpa.plot.curvecolor = "cyan"
sherpa> sherpa.plot.title_size = 2
sherpa> lplot 2 data arf
```

which produces [this plot](#) .

Since the `sherpa.plot` plot object controls the look of many plots, not just those of the ARF, the following changes will also be seen in these other plots. For instance:

```
sherpa> lp model
```

The [resulting plot](#)  matches the style of the ARF plot. Since there is only one plot here we can see that the title has indeed been increased.

---

## Changing the "FIT" plots

Although there are several different plot variables:

- `sherpa.plot` for general plots, such as the ARF and model components
- `sherpa.dataplot` for plotting the source and background data
- `sherpa.resplot` for plotting "residual" style plots
- and `sherpa.fitplot` for plotting fits

they are all essentially identical – *except* for fits. This is because the `sherpa.fitplot` object contains extra variables to control how the fitted model is displayed.


Here we change the fit-style plots; although most of fields are the same as seen with previous plots, there are extra ones (beginning with "fit\_"):

```
sherpa> print(sherpa.fitplot)
x_errorbars      = 0
y_errorbars      = 1
errs_style       = bar
errs_type        = both
x_log            = 0
y_log            = 0
curvestyle       = noline
curvecolor       = default
symbolstyle      = square
symbolcolor      = default
symbolsize       = 2
fit_curvestyle   = step
```

```

fit_curvecolor = red
fit_symbolstyle = none
fit_symbolcolor = default
fit_symbolsize = 2
xlabel_size = 1.5
ylabel_size = 1.5
zlabel_size = 1.5
title_size = 1.5
tickvals_size = 1.5
prefunc = NULL
postfunc = NULL
sherpa> sherpa.fitplot.x_log = 1
sherpa> sherpa.fitplot.y_log = 1
sherpa> sherpa.fitplot.symbolstyle = "bigpoint"
sherpa> sherpa.fitplot.symbolcolor = "blue"
sherpa> sherpa.fitplot.symbolsize = 1
sherpa> sherpa.fitplot.fit_curvecolor = "green"
sherpa> sherpa.fitplot.tickvals_size = 2
sherpa> lp fit

```

The resulting plot looks [like this](#) .

## Making global changes

Whilst it is useful to be able to change the look of different plot types, there are a number of changes that you may wish to make to all plots. If you do this often then setting the same fields in the different plot variables is just too much typing!

*Sherpa* defines several functions to help out – and, as discussed below – you can write your own.

## Existing Sherpa functions

The pre-defined functions are:

- set log(), set xlog(), and set ylog()
- set lin(), set xlin(), and set ylin()
- set erron(), set xerron(), and set yerron()
- set erroff(), set xerroff(), and set verroff()

and they change the settings in all the plot variables when called. So,

```
sherpa> set_log
```

is equivalent to

```

sherpa> sherpa.plot.x_log = 1
sherpa> sherpa.plot.y_log = 1
sherpa> sherpa.dataplot.x_log = 1
sherpa> sherpa.dataplot.y_log = 1
sherpa> sherpa.fitplot.x_log = 1
sherpa> sherpa.fitplot.y_log = 1
sherpa> sherpa.resplot.x_log = 1
sherpa> sherpa.resplot.y_log = 1

```

and

```
sherpa> set_yerroff
```

is the same as

```
sherpa> sherpa.plot.y_errorbars = 0
sherpa> sherpa.dataplot.y_errorbars = 0
sherpa> sherpa.fitplot.y_errorbars = 0
sherpa> sherpa.resplot.y_errorbars = 0
```

## Writing your own functions

If you have a set of options you often want to use you can write a *S-Lang* function to set them in one go. For instance, the following small function

```
define set_myplot() {
  set_ylog;
  sherpa.resplot.y_log = 0;
}
```

will set all the Y axes to have a logarithmic scale apart from the residual plots. If this code is stored in a file – for example `myplot.sl` – then it can be loaded into *Sherpa* using

```
sherpa> evalfile("myplot.sl")
1
sherpa> set_myplot
```

and then the function called. The 1 returned by the `evalfile()` call indicates that the file was loaded successfully. More information on loading *S-Lang* scripts can be found in the [Sherpa scripts](#) thread.

An alternative to this technique is to save the settings to a file, using `save_state()`, and then load in that file with the `USE` command when these options are required. This is described further in the next section.

## Saving your changes

The `save_state()` command can be used to save your settings so that they can be used in other *Sherpa* sessions.

If called with no arguments, `save_state()` will write out the contents of the *Sherpa* configuration variables to the file `$HOME/.sherpa-state-rc`. This file will be over-written without warning (so you should not make any changes to it). When *Sherpa* starts it will automatically load in the settings from this file, so your plots will retain the look you have set for them.

If called with an argument, `save_state()` will write out the settings to the file name (its argument) instead of `$HOME/.sherpa-state-rc`. This file can then be read into a *Sherpa* session using the `USE` command; this is useful if you want to set up different plot styles for use in different situations.

The beginning of the file will look something like:

```
% Sherpa state for ciaouser, Wed Jul 30 17:45:13 2006
```



```
sherpa.plot.x_errorbars = 0
sherpa.plot.y_errorbars = 0
sherpa.plot.errs_style = "bar"
sherpa.plot.errs_type = "both"
sherpa.plot.x_log = 1
sherpa.plot.y_log = 1
sherpa.plot.curvestyle = "step"
sherpa.plot.curvecolor = "cyan"
sherpa.plot.symbolstyle = "none"
sherpa.plot.symbolcolor = "default"
sherpa.plot.symbolsize = 2
sherpa.plot.xlabel_size = 1.5
sherpa.plot.ylabel_size = 1.5
sherpa.plot.zlabel_size = 1.5
sherpa.plot.title_size = 2
sherpa.plot.tickvals_size = 1.5
sherpa.plot.prefunc = NULL
sherpa.plot.postfunc = NULL
...
```

(the values will depend on what changes you have made to the plotting variables).

---

## Summary

The `sherpa.*plot` plotting variables allow a user to change the default look of plots in *Sherpa*. It does not provide unlimited customization; for this you need to use the `prefunc` and `postfunc` variables as described in the [Advanced customization of \*Sherpa\* plots](#) thread.

The [Changing the look of \*Sherpa\* plots using `setplot.sl`](#) thread provides an alternative interface to changing the fields in the plotting variables. Note that both methods can be used at the same time.

---

## History

14 Jan 2005 reviewed for CIAO 3.2: no changes

21 Dec 2005 reviewed for CIAO 3.3: no changes

01 Dec 2006 reviewed for CIAO 3.4: no changes

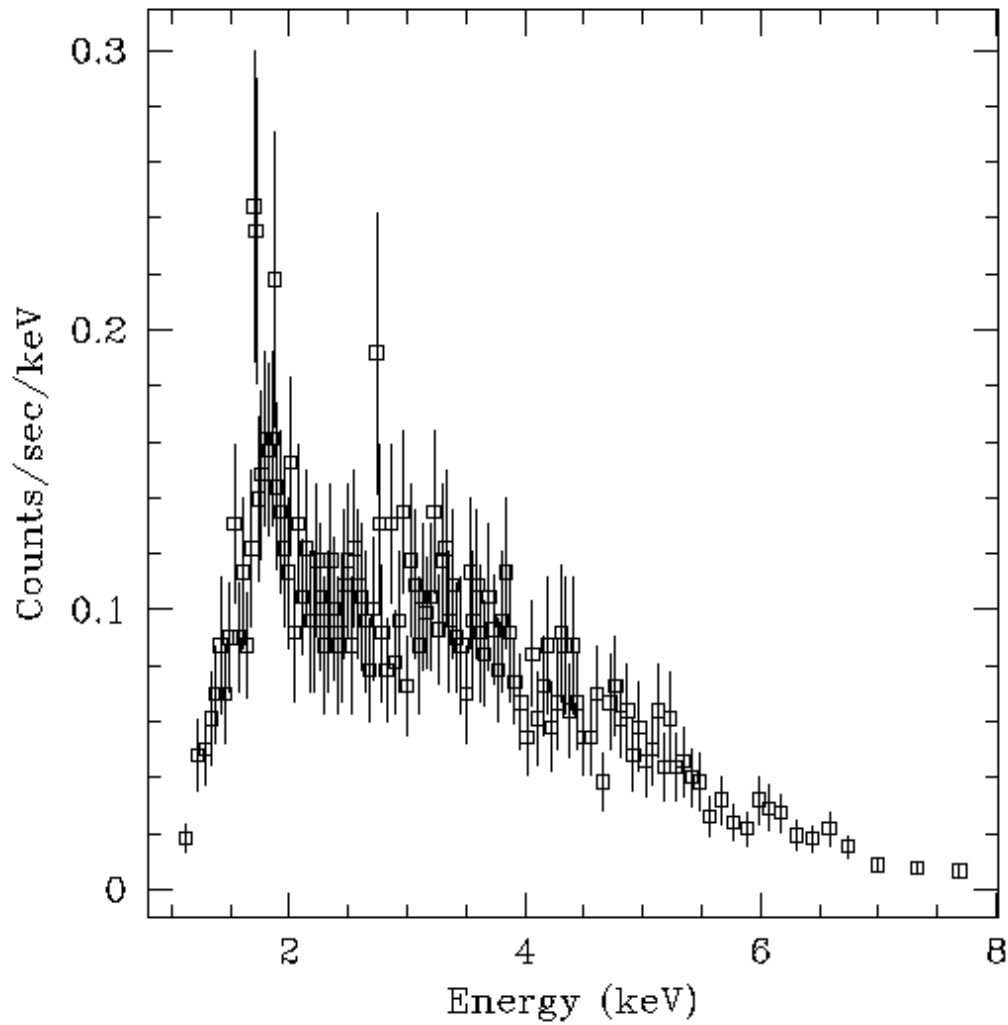
---

URL: [http://cxc.harvard.edu/sherpa/threads/setplot\\_manual/](http://cxc.harvard.edu/sherpa/threads/setplot_manual/)

Last modified: 1 Dec 2006

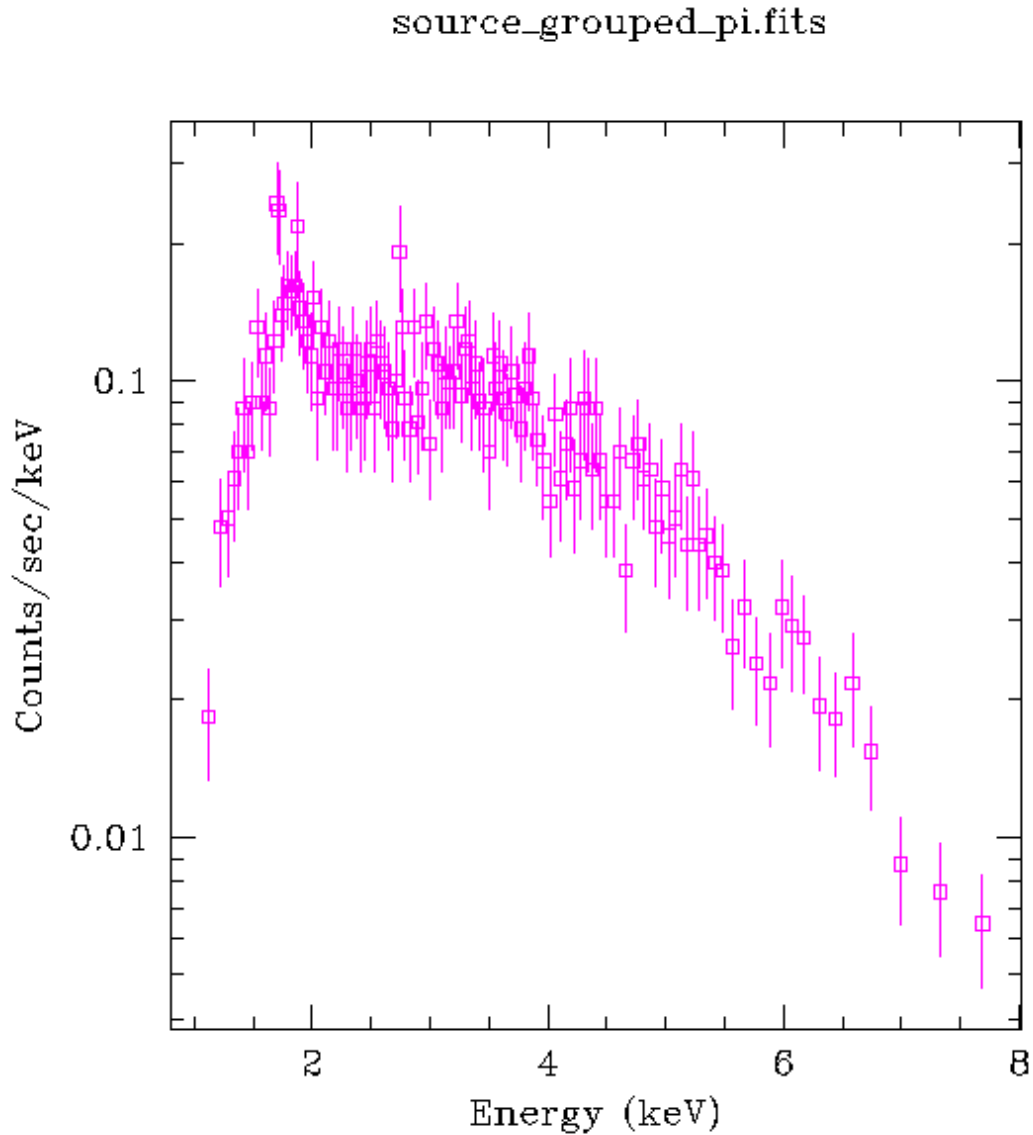
**Image 1: The plot produced by LP DATA with the default settings**

source\_grouped\_pi.fits



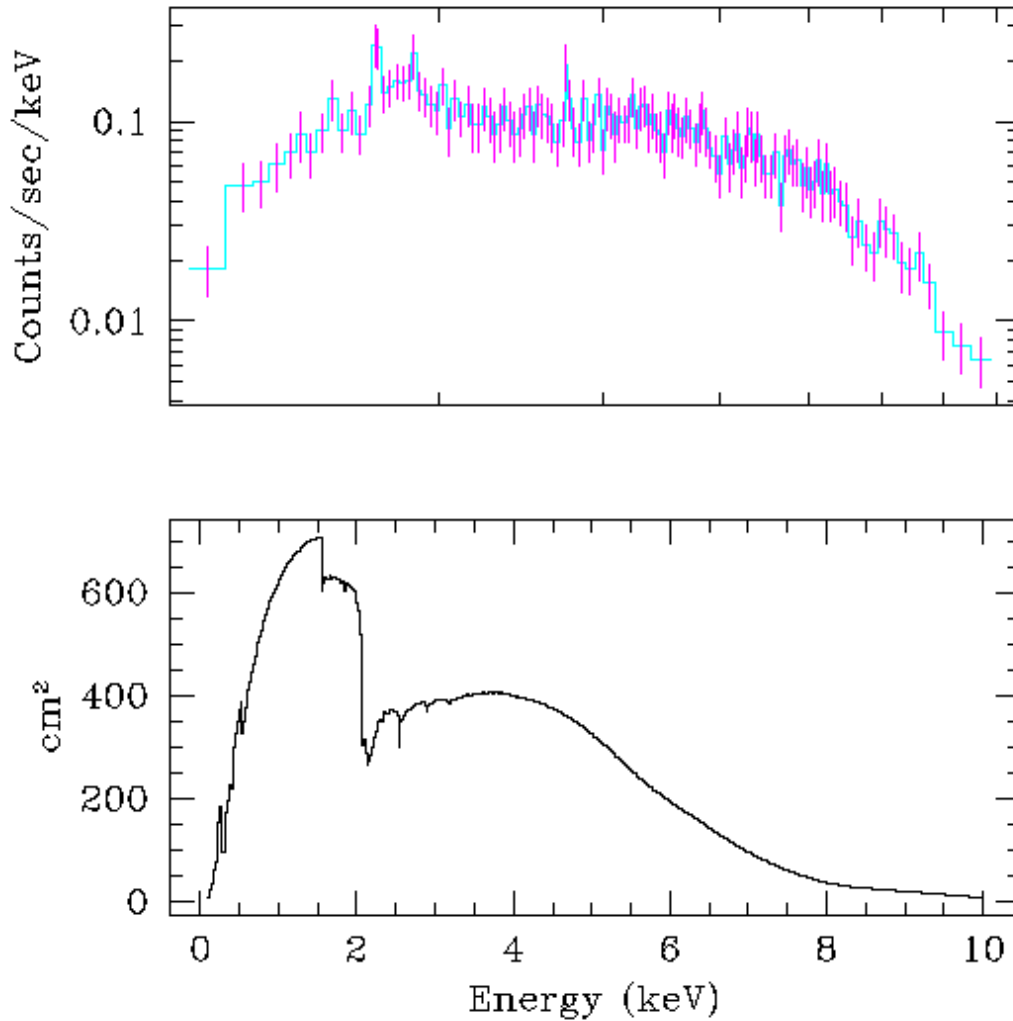
This shows the plot produced by the `LFPLOT DATA` command before changing any *Sherpa* plot variables.

**Image 2: The plot produced by LP DATA after changing the settings**



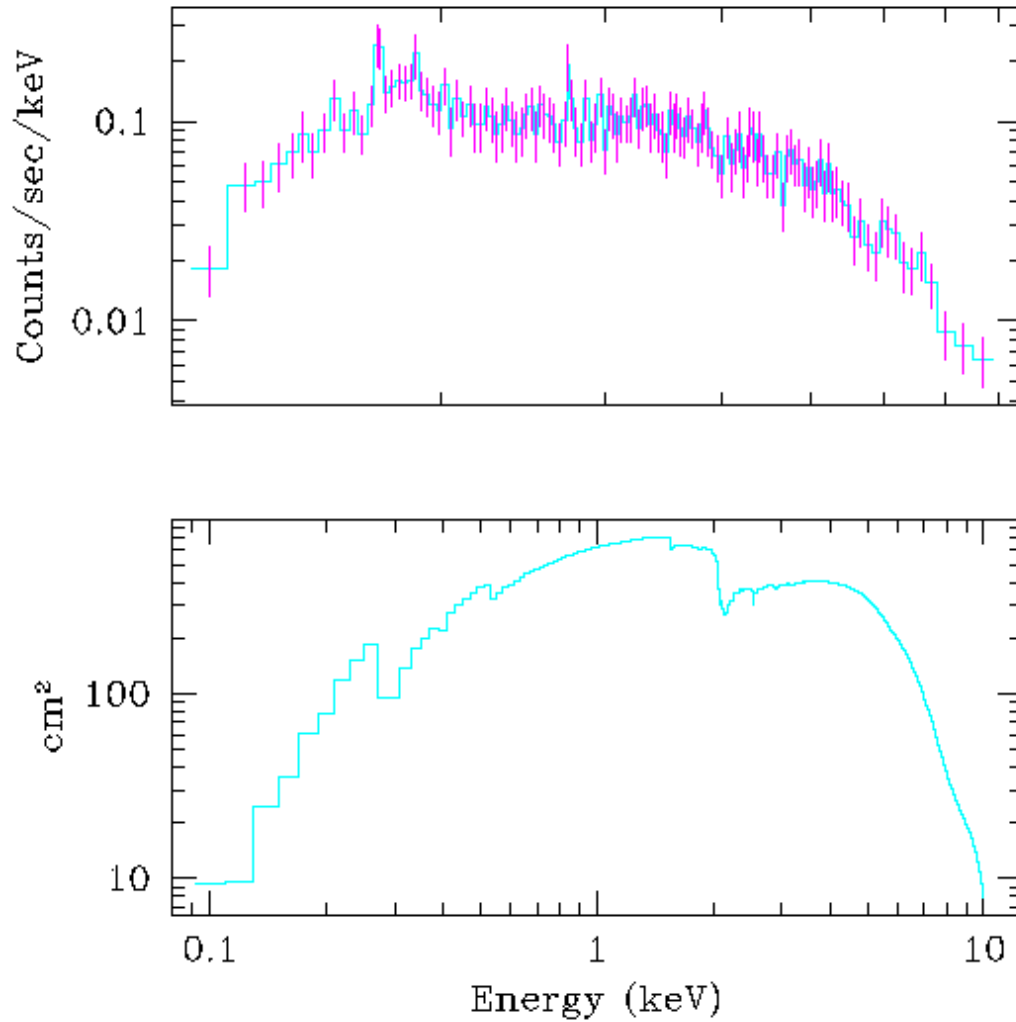
This shows the plot produced by the `LFPLOT DATA` command after changing the `sherpa.dataplot.y_log` and `sherpa.dataplot.symbolcolor` variables. The y axis is drawn with a logarithmic scale and the symbols (and error bars) are drawn in magenta.

**Image 3: The plot produced by LP DATA ARF**



This shows the plot produced by the `L PLOT DATA ARF`. The plot of the DATA section uses the new settings but the ARF plot does not, since it uses the settings defined in the `sherpa.plot` variable.

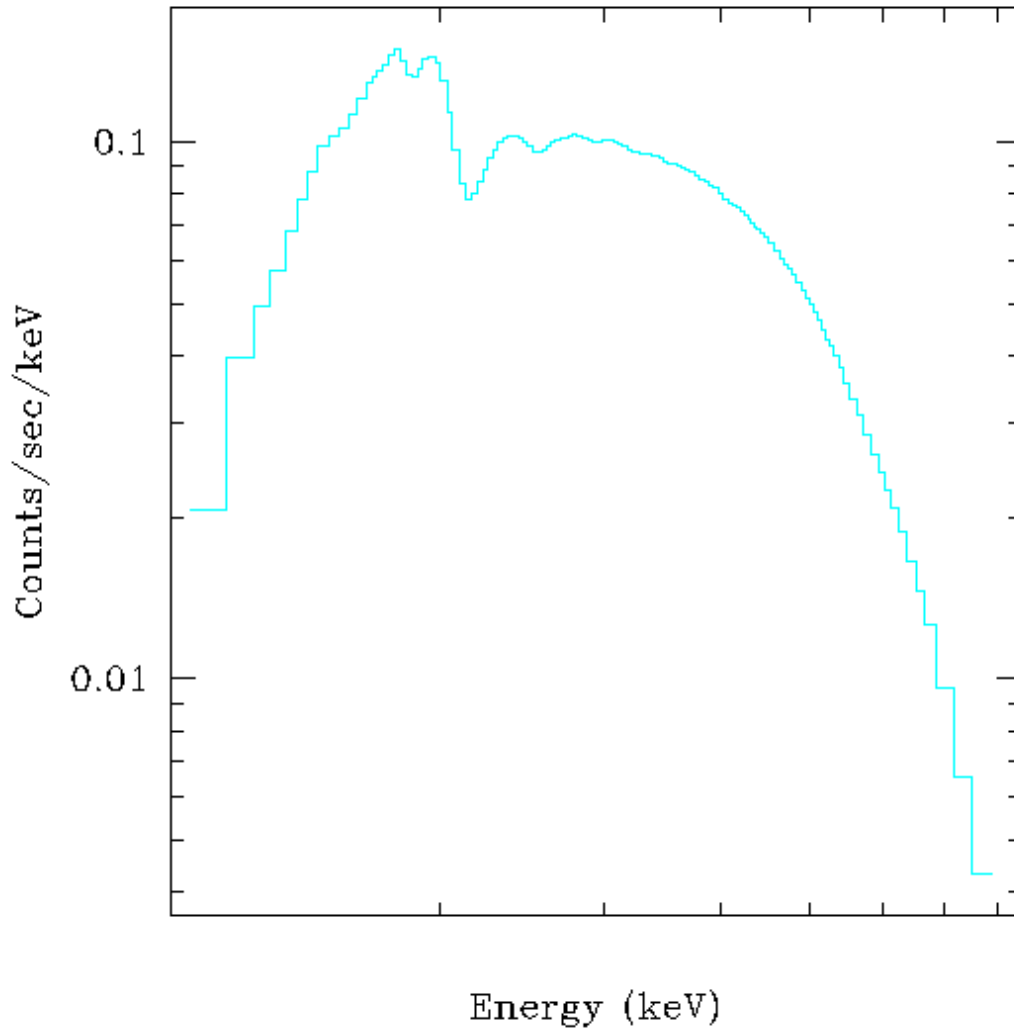
**Image 4: Applying a similar customisation to ARF and DATA plots**



Here we have changed the look of both the DATA and ARF plots to be similar (compare to the [previous version](#) of the plot). Note that the X-axes – though in a logarithmic scale for both plots – do not cover the same range.

**Image 5: LP MODEL is also changed**

Model Count Rate



Since the same plot variable is used to configure model and ARF plots – as well as many other plots – the model plot matches the style set up for the ARF plot. Since we now have a title we can see that the size of the title has been increased.

**Image 6: The plot produced by LP FIT**

