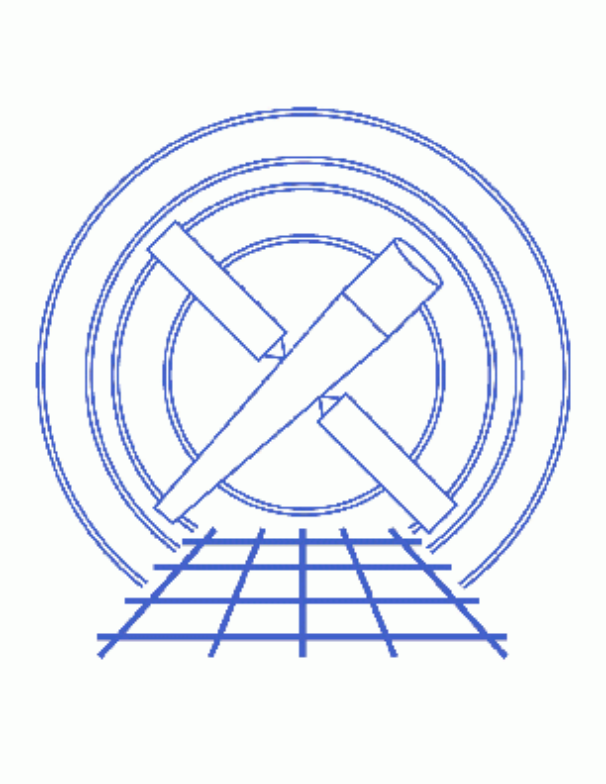
Sherpa Overview



Sherpa Threads (CIAO 3.4)

Table of Contents

URL: http://cxc.harvard.edu/sherpa/threads/intro/ Last modified: 1 Dec 2006

Sherpa Overview

Sherpa Threads

Overview

Last Update: 1 Dec 2006 − updated for CIAO 3.4: *Sherpa* version

Synopsis:

Sherpa is the generalized fitting engine of the CIAO software package (Chandra Interactive Analysis of Observations). *Sherpa* enables the user to fit models to data, particularly − but not exclusively − data that is being returned by NASA's Chandra X−ray Observatory (hereafter, CXO or Chandra).

Related Links:

- The other Introductory Sherpa threads.
- The Sherpa ahelp page.
- The overview of the S−Lang functions provided by *Sherpa*.

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

What Can Sherpa Do?

Sherpa is designed to allow complex model fitting to both spectral and spatial data, as well as derived data. Model expressions are defined by constructing a complex spectral shape from individual model components. These components can be labeled by the user so that more complex models may be built from simpler definitions. Model parameters may be linked using algebraic−style statements. *Sherpa* therefore establishes a model language for carrying out fitting.

Sherpa has additional powerful features, giving the user the capability to fit:

- data in 1−D, 2−D, or more;
- with arbitrary axes;
- with a number of built−in statistical tests;
- using a library of optimization methods.

Sherpa may be used:

- in the traditional X−ray astronomy fashion to fit pulse height spectra;
- to fit spatial images;

- to model simultaneously the spectral and spatial dimensions of a source.

Sherpa is structured to be flexible in other ways:

- it expects parameterized models of a source and background, then allowing for any of these parameters to be optimized.
- it contains an algebra that allows models to be combined in myriad ways, according to the user's needs.
- it also contains a library of built−in source models pertinent to CXO data (including the XSpec source model library).
- A user's own model can be readily included without recompiling *Sherpa*.
- The user's own statistical tests or optimization methods may be included, also without recompiling *Sherpa*.
- 2−D data temporal−spectral data is as easily analyzed as spatial data; moreover, the data may be derived data (e.g. temperatures and abundances) rather than original photons.
- Slang module allows easy access to the data and scripting.

Fitting data in Sherpa

Although *Sherpa* can fit models to a wide variety of data − be it X−ray spectra, one−dimensional radial profiles, or two−dimensional images − the same concepts and commands are used. Here we provide an overview of the concepts used in *Sherpa* when fitting data; please review the various sections of the *Sherpa* threads for detailed information on a specific application.

The first thing to know is that *Sherpa* uses a *forward−fitting* algorithm to fit a model to the data. This means that:

1. The ***source model*** is evaluated using the current set of ***parameters***.
2. The ***predicted data*** is created by taking the source model and then:
    A. If defined, the ***instrument model*** is used to convert the source model into something that the detector would produce (i.e. it accounts for the *effective area* and any *instrumental blurring* of the detector).
    B. The ***background model*** can be used to add in the predicted background signal. Note that the background can instead be *subtracted* from the observed data or ignored, depending on the quality of the data and how well understood the background is.
    C. If defined, the ***noise model*** is used to account for source components that should not be passed through the effective area of the instrument.
3. The predicted data is then compared to the ***observed data*** using the chosen ***statistic***. There are a number of statistics to chose from, depending on the quality of the data.
4. If the "best−fit location" has been found then the fit will stop, otherwise the parameter values are updated to try and improve the fit and the loop re−started. The choice of how the parameter values change is made by the ***optimization method***. *Sherpa* provides a large number of methods to choose from, depending on the complexity of the search space.

Not all the model components discussed above need be created − the choice depends on the quality of your data and what scientific question you are trying to answer by fitting a model to it.

Observed data

This is the data recorded by the instrument and can be stored in either ASCII or binary (FITS) form. Error values can also be read in for each element, calculated from the data (if one of the chi−square statistice is being used), or set to equal a mathematical expression. Systematic errors can also be included if necessary.

More than one file can be loaded into *Sherpa* at a time and these datasets can then be fit simultaneously or independently. For instance if you have LETGS data then loading the PHA−II file will load the different orders into separate datasets. Similarly, if you have XMM imaging data you can fit the same spatial model to the data from the MOS1, MOS2, and PN detectors.

When analysing X−ray spectra it is best to use PHA−I or PHA−II files since *Sherpa* can recognise them and set up things accordingly. For instance, *Sherpa* will recognise the grouping information added by `dmgroup` and will then used the grouped data for fitting.

Background data

While *Sherpa* does not require any background data, you can load in a data file using the same formats described for the source data file. Once loaded the background data can be subtracted from the source dataset or used to fit a background model. This latter step is necessary when using the statistics based on the Cash and Bayesian maximum−likelihood functions.

If using PHA−I or PHA−II data then the background will be automatically set to use the contents of the filename stored in the `BACKFILE` keyword in the header of your source file. Unlike X−Spec, the background will ***not*** be automatically subtracted from the data and so you will need to use the `SUBTRACT` command if you want this.

The instrument model

The instrument model describes the mapping between the physical quantities of the source model and those of the detector. Setting up the instrument model is often simple − in fact its automatic for PHA−I files if the `ANCRFILE` and `RESPFILE` keywords are correctly defined − although it is also possible to create combinations of instrument models if required.

When fitting spectral data the instrument model contains both the effective area (read in as the ARF) and − if not grating data − the spectral blurring of the detector (read in as the RMF). For radial profiles the PSF of the instrument can be used to convolve the model before fitting to the data. With imaging data the effective area can be included by using an exposure map and the model can also be convolved by the instrumental PSF. It is also possible to set up a separate instrument model for the background dataset.

It is also possible to fit a dataset ***without*** setting up an instrument model. This may be because one is not needed − such as fitting a lightcurve − or because the instrumental effects would not significantly change the results − e.g. when fitting a radial profile to a source that covers most of an ACIS chip, the 0.5 arcsecond resolution of the instrument can normally be ignored.

Filtering the observed data

It is often the case that you do not want to fit all the data you have loaded: you may want to exclude areas where the particle background dominates your X−ray spectrum or exclude point sources when fitting an image of an extended source. *Sherpa* allows you to define what ranges of your data you want included or excluded from the

fit. These regions can be changed at any time, which allows you to see how sensitive the fit results are to the chosen data range.

If an instrument model is supplied which contains a mapping between instrument and physical quantities − such as that between energy and the PI channel of an X−ray spectrum − then filters can also be defined in physical units.

The source model

Sherpa contains a large number of components that can be combined to define the source model that is to be fitted to the data. These range from one−dimensional and two−dimensional functions such as power law, polynomial, and gaussian to X−ray specific models (many of the X−Spec models are available). It is also possible to write your own models using either S−Lang or a compiled language like C and FORTRAN. If the background is being fit, rather than subtracted from the data or ignored, a separate source expression can be created to define the background model.

Each model component has a set of parameters that define its form, such as the FWHM, position, and ampliude of a one−dimensional gaussian. These values can be fixed (***frozen***) so they are not changed during fitting or freed (***thawed***) so they are. You can also change the value of a parameter and see how that changes the resulting source model.

When fitting multiple datasets simultaneously there are several ways that constraints can be applied. The same component (or components) can be used in the source expressions for the different datasets so that the same model is fit to all the datasets. Or, for a more fine−grained approach, individual parameters can be linked so that they are the same; for instance you can fit two power laws, with independent normalisations, but force their slopes to be the same.

For the more advanced cases it it also possible to create ***nested*** and ***joint−mode*** models.

Modelling detector noise

Both the source and background models are passed through the full instrument model when calculating the predicted data. At times it may be useful to be able to include a model that is not affected by the effective area (be it ARF or exposure map) of the instrument. An example would be when trying to include a model of the particle background in your fits to *Chandra* data. The noise model is used in such situations.

Predicted data

The predicted data is used when fitting the model but may also be of interest on its own, such as when simulating a source spectrum for a proposal or to estimate parameter errors. It is formed by combining the source and background models, after multiplication with the effective area (be it an ARF or an exposure map), together with the noise model. The resulting sum is then blurred by the instrument response (the RMF for spectral data and a PSF for imaging data) to create the predicted data.

Fit statistic

There are a number of statistics to use to compare the observed and predicted datasets: there is a set of choices based on the chi−square statistic with different ways for the errors to be calculated; maximum−likelihood functions based on the Cash statistic; a Bayesian maximum−likelihood function; and the ability to write your own

statistic.

Fit optimisation

The optimisation method controls how the fitted parameters are changed so as to improve the fit after each iteration. *Sherpa* provides a range of optimization methods − each with its own advantages and disadvantages − because there is no guarantee that, for a given situation, there will not be multiple local minima in the fit statistic (i.e. search space). There are two broad classes of optimiser available within *Sherpa*: "single−shot" and "scatter−shot" methods.

The single−shot methods start at a single position in the search space (the initial parameter values) and then vary them until a local minimum is found. Within this class there are trade−offs to be made between speed and robustness (i.e. how likely you are to find the local minimum).

The scatter−shot methods are used to allow the search to be made from a range of initial parameter values spread throughout the search space. They may therefore be more likely to find the global, rather than local, minimum of fit statistic. Some of these methods include the single−shot methods − so the fit is optimised starting at each grid point − which can provide the best of both worlds but can take a very long time to run.

It is possible to change the optimisation method and re−fit, to see how reliable the fit results are.

Estimating errors on best−fit parameters

Once the best−fit location has been found, *Sherpa* provides a number of routines for calculating the errors on the parameters. It is also possible to calculate a confidence map showing how two parameters are correlated.

Launching Sherpa

Once the CIAO software package has been installed, *Sherpa* is launched on the command line by typing `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>
```

From this point forward, you are working within the *Sherpa* environment.

Issuing Commands

Sherpa can be controlled in several ways, as described in the *Sherpa* ahelp page and below.

The Sherpa command language

Sherpa commands are entered on the command line:

```
sherpa> READ DATA data1.dat 1 2
sherpa> READ ERRORS data1.dat 1 3
sherpa> LPLOT DATA
sherpa> SOURCE = POLY[model1]
sherpa> THAW model1.c1
```

Note that within this document, as well as in the Sherpa Reference Manual, non−variable command elements are capitalized. However, *Sherpa* commands are ***not case sensitive*** and therefore may be issued in either upper or lower case.

Using S−Lang within Sherpa

The *Sherpa* command language is intended to allow a user to quickly load in a dataset, set up the appropriate source model, and fit it. It was *not* designed for users who want to script their fitting sessions, or want the ability to manipulate (read and write) the data and results in Sherpa.

As with *ChIPS*, S−Lang commands can be entered at the *Sherpa* prompt; unlike the *Sherpa* command language case *is* important for S−Lang commands. In the following example we call the S−Lang function `run_fit()`, which is part of the *Sherpa* S−Lang module:

```
sherpa> res = run_fit()
```

Note that we have been able to ignore both the trailing semi−colon and the need to declare `res` as a `variable` as *Sherpa* does this for us. When used from a script − executed either via the `evalfile()` command or via the `slsh` tool − you would have had to say something like:

```
variable res = run_fit();
```

Using scripts

Both sets of commands can also be used from scripts: the `use` command for *Sherpa* scripts and the `evalfile()` function to run S−Lang scripts. This is described in detail in the *Sherpa* and Scripts thread.

Getting Help

Help is available for all valid *Sherpa* command names, as well as for all model, method, and statistic names, by typing `ahelp`. For example, help for the command `FREEZE` is accessed by typing:

```
sherpa> ahelp freeze
```
This returns a brief synopsis of the command, syntax, and a few examples.

More extensive help is available by typing:

```
sherpa> ahelp −w freeze
```
Adding the `−w` argument to `ahelp` causes the full amount of available reference material to be displayed in the user's WWW browser.

To get the very latest version of the manual using `ahelp`, type:

```
sherpa> ahelp −i freeze
```
Adding the `−i` argument to `ahelp` causes the full amount of available reference material to be displayed in the user's WWW browser, taken from the current online CIAO web pages.

Help is also available for a set of *Sherpa* related concepts and keywords:

```
sherpa> ahelp input
sherpa> ahelp filter
```
Further information about the CIAO help system is available by typing:

```
sherpa> ahelp
```

Ending a Sherpa Session

To end a *Sherpa* session, use the `EXIT` command (or, equivalently, `BYE` or `QUIT`):

```
sherpa> EXIT
Goodbye.
```

Saving and Restoring Sherpa sessions

Sherpa allows you to save the current state − e.g. loaded data files, applied filters, current parameter values − and then restore them at a later time. This allows you to easily re−start an analysis session, *even after* exitting *Sherpa*, or to send such a session to a colleague for them to work on. *Sherpa* will also automatically create such a file if it crashes, allowing you to re−start the analysis.

Saving the current state

The `SAVE` command will create an ASCII file listing all commands necessary to re−create the current state of *Sherpa*. If you are in the middle of an analysis then

```
sherpa> SAVE ALL state.shp
```
creates the text file `state.shp`.

Restoring the state

The text file (here `state.shp`) can then be used from a new *Sherpa* to restore the analysis by saying:

```
sherpa> ERASE ALL
sherpa> USE state.shp
```

The call to `ERASE ALL` may not be necessary but ensures that any previous settings are cleared before the old session is restored. The data analysis can be resumed after the use of `USE`.

An alternative technique is to start *Sherpa* and give it the file name:

```
unix% sherpa state.shp
```

Recovering a Sherpa session

If for any reason, *Sherpa* exits with a nonzero status, a log file of the session is saved in the user's home directory to assist in recovering the *Sherpa* session. This log file contains contains all commands that were successfully executed up to the point that *Sherpa* failed. The name of the saved file is `$HOME/.sherpa-session-PID`, where `PID` is the process ID of the *Sherpa* session that failed.

To recover the session sherpa needs to be launched with the pointer to the saved file.For example, if the PID of a failed *Sherpa* session was 2048, then the command:

```
unix% sherpa ~/.sherpa-session-2048
```

will start *Sherpa* and execute all the commands from that failed session, up to but not including the command that caused a failure.

When *Sherpa* exits normally, with an exit status of zero, no such log file is left behind.

Obtaining Sherpa Threads Data Files

The *Sherpa* threads page contains a number of example data analysis sessions in *Sherpa*. The data files used in these threads are available as sherpa.tar.gz. After downloading and unpacking these files, the user should have a local directory named "sherpa" that has a subdirectory for each *Sherpa* thread. Each subdirectory contains the data for that thread, along with an ASCII file of the thread's *Sherpa* commands. You are now prepared to follow the example threads in order to gain familiarity with *Sherpa*.

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: *Sherpa* version

URL: http://cxc.harvard.edu/sherpa/threads/intro/ Last modified: 1 Dec 2006