# Sherpa: a mission-independent data analysis application

Peter E. Freeman, Stephen Doe, Aneta Siemiginowska

Harvard-Smithsonian Center for Astrophysics

## ABSTRACT

The ever-increasing quality and complexity of astronomical data underscores the need for new and powerful data analysis applications. This need has led to the development of *Sherpa*, a modeling and fitting program in the *CIAO* software package that enables the analysis of multi-dimensional, multi-wavelength data. In this paper, we present an overview of *Sherpa*'s features, which include: support for a wide variety of input and output data formats, including the new Model Descriptor List (MDL) format; a model language which permits the construction of arbitrarily complex model expressions, including ones representing instrument characteristics; a wide variety of fit statistics and methods of optimization, model comparison, and parameter estimation; multi-dimensional visualization, provided by *ChIPS*; and new interactive analysis capabilities provided by embedding the S–Lang interpreted scripting language. We conclude by showing example *Sherpa* analysis sessions.

**Keywords:** Chandra, CIAO, data analysis, fitting, modeling, Virtual Observatory

## 1. INTRODUCTION

In astronomical data analysis, one develops models of physical processes in the spectral, spatial, and/or temporal domains, then fits these models to observed data. These data may be of arbitrary dimension, and/or they may have been collected using an arbitrary number of telescopes that observe in different wavelength bands. *Sherpa*, the modeling and fitting application of the *Chandra Interactive Analysis of Observations (CIAO)* software package,[1] is designed to tackle such complex multi-dimensional, multi-wavelength analyses. Free of hard-wired instrument details, *Sherpa* is outfitted with a wide variety of models, fit statistics, and methods of optimization, model comparison, and parameter estimation, and it offers powerful embedded visualization, scripting, and data manipulation capabilities. It can thus be used to analyze energy- or wavelength-space data from, *e.g.*, ground-based telescopes, *ISO*, *Hubble*, *XMM*, and the *Chandra X-ray Observatory*, as well as from next-generation projects such as the Virtual Observatory.

In this paper we present a basic overview of *Sherpa*. In §2, we describe the application itself, while in §3 we describe *Sherpa*'s capabilities using a typical *Sherpa* session as a framework, from reading in data to determining the $1\sigma$ errors on the best-fit model parameters. In §4 we provide brief examples of how one may use *Sherpa*. For more information, the reader may consult the on-line *Sherpa* manual.[2]

## 2. IMPLEMENTATION OF SHERPA

The authors have developed *Sherpa* using the object-oriented C++ programming language.[3] In object-oriented programming, objects encapsulate related data and functions; thus, classes (from which objects are instantiated) can be written that more closely model the problem domain. For a modeling and fitting application, classes containing fitting methods, statistics, models, and data organize the code in a way that more intuitively mirrors the concepts underlying the fitting process.

*Sherpa* is an interactive application, using a lex/yacc-based parser to interpret commands. *Sherpa* accepts input via the command-line interface or ASCII script files.*

---

Send correspondence to: (pfreeman,sdoe,asiemiginowska)@head.cfa.harvard.edu; phone 1 617 496-7824; fax 1 617 495-7356; Harvard-Smithsonian Center for Astrophysics, MS 81, 60 Garden St., Cambridge, MA, 02138. 

*Application developers may also use *Sherpa*'s "wrapper" class, which provides an API for both C and C++ code.

The *CIAO* package also contains a number of other programs and libraries that enhance *Sherpa*'s capabilities. Data I/O is provided by the *CIAO* Data Model library.[4] The Data Model gives a higher-level, abstract view of astronomical data files and provides transparent access to the most common astronomical file formats (FITS, *IRAF* IMH, *IRAF* QPOE).[†] The Data Model provides a uniform interface so that the application need no longer use I/O functions specific to particular file formats. The Data Model also has a sophisticated filtering and binning syntax that allows the extraction of selected portions of data contained in a file.

For all visualization needs, *Sherpa* uses *ChIPS*, the *Chandra Imaging and Plotting System*.[5] As the name indicates, *ChIPS* provides an interface to plotting and imaging applications (currently *SM*,[6] *SAODS9*,[7] and *SAOTNG*[7]). *ChIPS* is both a stand-alone, interactive application and a C++ library (which is how it is used by *Sherpa*–it passes commands and data to *ChIPS*). *Sherpa* can access *ChIPS* functions through the *ChIPS* API, but it is also possible for *ChIPS* commands to be input directly at the *Sherpa* command-line.

As of *CIAO 2.1*, the S–Lang interpreted scripting language[8] has been embedded in *CIAO*. Its features include:

- global and local interpreted variables, and multidimensional arrays (up to seven dimensions);
- branching and looping, and programmability with user-defined functions;
- string and numeric data types, structures, and a limited form of pointers;
- built-in arithmetic and mathematical functions, which operate transparently on arrays; and
- extensibility–the ability to create new functionality for *CIAO* applications (*e.g.* GUIDE; see §4.3).

S–Lang is accessed through a supplementary library layer, dubbed Variables, Math, and Macros (VARMM),[9] which gives the user additional capability to define structured variables directly from disk files, as well as enabling existing *CIAO* applications to access S–Lang variables and other features with a bare minimum of new development effort.

## 3. SHERPA FUNCTIONALITY

The capabilities of *Sherpa* can be best described by following the steps of a typical analysis session. In such a session, the user would:

- read in (source and/or background) data (and set filters, *etc.*);
- build models that describe these data (as well as the telescope/detector combination);
- choose a statistic that quantifies how well these models describe the data;
- fit the models to the data, determining the minimum value of the chosen statistic;
- compare best-fit results achieved with different models to select one best-fit model; and
- estimate the errors for each parameter of the best-fit model.

Below, we discuss each of these items in turn. We note that a typical session would also include *ChIPS* visualization (of the source data, or of the background fit, *etc.*); while we do not discuss it specifically in this section, we provide examples of visualization in §4.

### 3.1. Sherpa Data Input (and Output)

Data input marks the beginning of a *Sherpa* analysis session. The data may be input from either a file on disk or an interpreted S–Lang variable. In Table 1, we list data types that may be input into *Sherpa*, while in Table 2 we list currently supported file formats. (Instrument characteristics, such as the point-spread function, or PSF, may also be contained in files that are read in when an instrument model is specified; see §3.2.1.)

One only needs to read in source data to start an analysis session; other data types (such as background) are not required. However, some types of data may be input automatically when source data are read; for instance, a PHA file can have columns that specify the statistical and systematic errors (`STAT_ERR` and `SYS_ERR`, respectively) and the data grouping (`GROUPING`), and it can have a keyword (`BACKFILE`) specifying the background dataset. Also, one can specify statistical and systematic errors, filters, and statistical weights via the command-line interface. If statistical errors are not input or specified, they are estimated by *Sherpa* during fitting; see §3.3.

An arbitrary number of datasets may be input into *Sherpa*, and arbitrary subsets of these data may be jointly analyzed. Since current standard processing of *Chandra* grating data includes the extraction of background spectra

[†]ASCII I/O is not provided by the Data Model but is provided elsewhere in *CIAO* library code.

**Table 1.** Input data types.

| | |
|---|---|
| `DATA` | Source data |
| `BACK` | Background data |
| `(B)ERRORS` | Source or background errors |
| `(B)ERRORS SYSTEM` | Source or background systematic errors |
| `FILTER` | Specification of which bins of a given dataset are to be analyzed |
| `WEIGHT` | Statistical weights for each datum |
| `GROUPS` | Specification of how data are to be binned |

**Table 2.** Supported file types.

| | |
|---|---|
| `ASCII` | ASCII data |
| `FITSBIN` | FITS binary table |
| `FITS` | FITS image |
| `IMH` | *IRAF* IMH image |
| `PHA` | Type I & II PHA data |
| `QP` | *IRAF* QPOE image |

from regions on either side of the source extraction region, one may also specify up to two background datasets per source dataset. These data may be fit simultaneously with the source data (see §4.2), or they may be subtracted from the source data on a channel-by-channel basis prior to a fit:

$$S_i' \;=\; S_i - \beta_S t_S \left[ \frac{\sum_{j=1}^{N} B_{i,j}}{\sum_{j=1}^{N} \beta_{B_j} t_{B_j}} \right] . \tag{1}$$

$S_i$ is the source datum in bin $i$, $B_{i,j}$ is the background datum in bin $i$ of background set $j$, $t$ is the observation time, and $\beta$ is the "backscale" (the `BACKSCAL` header keyword value in a PHA file), typically defined as the ratio of data extraction area to total detector area.

At any time during an analysis session, quantities like the background-subtracted data, convolved model amplitudes, or fit residuals may be output to files using the `WRITE` command. (The files may be saved in any of the formats listed in Table 2.) A user may also save (and later restore) the state of a *Sherpa* analysis session using a Model Descriptor List (MDL). The MDL is a record of all information relevant to a Sherpa fitting session: the names of all input data files and associated filters; all defined source and instrument models, with model parameter settings; the choices of optimization method and fit statistic; and the fluxes and identification of emission/absorption lines that the user may have identified (see §4.3). The MDL may be either saved to disk or instantiated as an S–Lang variable.

### 3.2. Building Model Expressions

Once source and/or background data are input, the next step is to create model expressions reflecting one's knowledge of the physical processes which gave rise to those data. (The user can also build model expressions that represent the observing instruments. See §3.2.1.) *Sherpa* currently provides nearly 40 of its own one- and two-dimensional models and 90 one-dimensional *XSPEC*[10] models that may be arbitrarily combined to build complex composite models, as we show below. Note that a user can also define a model[‡] and compile it into the `libascfitUser.so` shared-object library,[§] where it can be accessed by *Sherpa*.

[‡]The `usermodel`. One can also define an optimization method (`usermethod`) and statistic (`userstatistic`).
[§]In *CIAO* 2.2, *Sherpa* will also support user-model definition via S–Lang scripts.

**Sherpa model language.** The *Sherpa* model language resolves ambiguity by allowing the user to give a unique name or alias to each instance of a model. For example, if two datasets are entered, and each is to be fit with a Gaussian model, but with different parameters, one might type:¶

```
sherpa> gauss1d[g1]
sherpa> gauss1d[g2]
sherpa> source 1 = g1
sherpa> source 2 = g2
```

whereas if each is to be fit with the same Gaussian model, one might type:

```
sherpa> gauss1d[g1]
sherpa> source 1:2 = g1
```

**Linking parameters to other parameters or to models.** One can link an individual model parameter to another model parameter, so that their values are correlated. For instance, if a particular atomic line is observed by two different detectors, it could be modeled with two Gaussian functions whose centroids are linked:

```
sherpa> source 1 = gauss1d[g1]
sherpa> source 2 = gauss1d[g2]
sherpa> g1.pos => g2.pos
```

(Note that simple arithmetic relations are also possible, *e.g.* `g1.ampl => 2*g2.ampl`.) At this point, `g1.pos` is no longer a free parameter of the fit. It can be made free again with the command `UNLINK g1.pos`.

One can also link an individual model parameter to a model, to describe how a parameter's value will vary as a function of position in parameter space. For instance, one can model emission from an accretion disk using a blackbody function whose temperature is a function of radius:

```
sherpa> source = bb
sherpa> Temperature = POLY
sherpa> bb.kT => Temperature
```

**Nesting model expressions.** A model may be nested within another, *i.e.* one may specify a model expression of the form g(f(x)). In this example, the input data axis is transformed to log-space using *Sherpa*'s log model, and a blackbody model is evaluated in that space:

```
sherpa> logenergy = shlog
sherpa> source = bb{logenergy}
```

**Multi-dimensional model expressions.** One can specify completely different models that are to be evaluated along different axes of a multi-dimensional dataset, as in this example, where two-dimensional spectral-radius data are modeled with a combination of Lorentzian and power-law models:

```
sherpa> lorentz[Spatial]
sherpa> pow[Spec]
sherpa> source = Spatial{x1}*Spec{x2}
```

`x1` and `x2` represent the first and second axes of the input image, respectively.

¶The relationship between `gauss1d`, and `g1` and `g2` above is similar to the class-object relationship in object-oriented programming: `gauss1d` is the class, specifying a Gaussian function with parameters position, amplitude, and full-width at half-maximum, while `g1` and `g2` are instantiated objects of the class, each containing a specific set of parameter values.

### 3.2.1. Instrument models

Instrument models are used to quantify characteristics, such as effective area, a detector's energy response, or a mirror's point-spread function. They provide a mapping from photon space (where source and background models are evaluated) to counts space (where fit statistics are computed). The instrument model class is the key element which makes *Sherpa* a mission-independent application, permitting analysis of data observed by any telescope, regardless of whether it is ground-based or space-based.

Currently, *Sherpa* defines three instrument model classes: (1) `RSP`, in which an evaluated one-dimensional model is multiplied by an ancillary response (ARF, *i.e.* an effective area) on a bin-by-bin basis, then folded through a response matrix (RMF); (2) `PSFFromFile`, in which the evaluated one- or two-dimensional model is convolved with a numeric kernel; and (3) `PSFFromTCD`, in which the evaluated one- or two-dimensional model is convolved with an analytic kernel (*e.g.* Gaussian) defined within *CIAO*'s Transformation, Convolution, and Deconvolution (TCD) library. Future versions of *Sherpa* may include new classes to treat, *e.g.*, two-dimensional exposure maps.

## 3.3. Statistics

### 3.3.1. Statistics based on the $\chi^2$ distribution

The $\chi^2$ statistic is appropriate for the analysis of Gaussian-distributed data. It is defined as

$$\chi^2 \equiv \sum_i \frac{(D_i - M_i)^2}{\sigma_i^2} .$$

where $D_i$ is the (source or background) data in bin $i$, $M_i$ is the (convolved source or background) model predicted amplitude in bin $i$, and $\sigma_i$ is the estimated error for the $i^{\rm th}$ datum (the square root of the variance of the distribution from which that datum had been sampled). As noted in §3.1, one may specify the errors via a file or the command-line; if this is done, the $\chi^2$ statistic is used as shown above. Otherwise, the data are assumed to be Poisson-distributed,[||] with the errors for each datum estimated during analysis. The large array of error estimators that *Sherpa* provides is one of its key features; these are listed in Table 3. Note that the entries in this table are only correct if the background data have not been subtracted from the source data; otherwise errors are propagated in the standard manner ($\sigma_{S'}^2 = \sigma_S^2 + \sigma_B^2$). Also note that error estimates based on model amplitudes are inappropriate to use in the analysis of background-subtracted data, as they generally underestimate the true error.

Using a $\chi^2$-based statistic to analyze counts data is generally only valid in the Gaussian (high-counts) limit ($\gtrsim 5$ counts in *each* bin). This is because the approximations that must be made to derive the $\chi^2$ statistic from Poisson log-likelihood $\log\mathcal{L}$ break down otherwise. The `CHI GEHRELS`[11] and `CHI PRIMINI`[12] statistics are designed to work with low-count data; note that the former it is not generally sampled from the $\chi^2$ distribution and thus the derived best-fit statistic may appear to be "too good" ($\chi_G^2/N \ll 1$, where $N$ is the number of degrees of freedom in the fit), in the low-counts limit.

### 3.3.2. Statistics based on the Poisson likelihood

The Poisson likelihood function is

$$\mathcal{L} = \prod_i \frac{M_i^{D_i}}{D_i!} \exp(-M_i) . \qquad (2)$$

*Sherpa* features two statistics based on this function: `CASH` and `BAYES`.

The version of the `CASH` statistic[13] used by *Sherpa* is derived from $\mathcal{L}$ by (1) taking its logarithm, (2) dropping the factorial term (which remains constant during fits to given datasets), (3) multiplying by two, and (4) changing the sign (so that the statistic may be minimized, like $\chi^2$):

$$C \equiv 2\sum_i [M_i - D_i \log M_i] , \qquad (3)$$

In the high-counts limit, $\Delta C \sim \Delta\chi^2$, so that in principle one can use $\Delta C$ instead of $\Delta\chi^2$ in model comparison tests (see §3.5).

[||] The Poisson distribution tends asymptotically towards a Gaussian distribution as its expectation value approaches infinity.

**Table 3.** Statistics based on the $\chi^2$ distribution.

| Statistic | Variance $\sigma_i^2$ |
|---|---|
| `CHI DVAR` | $D_i$ |
| `CHI GEHRELS` (*Sherpa* default) | $\left[1+\sqrt{D_i+0.75}\right]^2$ |
| `CHI MVAR` | $M_i$ |
| `CHI PARENT` | $(\sum_{i=1}^{N} D_i)/N$ |
| `CHI PRIMINI` | $M_i$ from previous best-fit |

The `BAYES` statistic[14] is based on Bayesian statistical methodology** and is appropriate to use when a background is input and the rate of background accumulation may be taken as the same in both the background and source extraction regions. This statistic takes into account uncertainty in the (implicitly defined) background amplitudes via marginalization:

$$B \equiv -p(\vec{x}_S|D) = -\sum_i \int_{x_{B,i}} dx_{B,i}\, p(\vec{x}_S, x_{B,i}|D)\,,$$

where $\vec{x}_S$ represents the set of source model parameters and $x_{B,i}$ is the background amplitude in the $i^{\text{th}}$ bin. (Note that the above equation has an analytic solution that we do not reproduce here.)

Note that because the `CASH` and `BAYES` statistics are based on the likelihood function, they should not be applied to background-subtracted data. Also, there is no "goodness-of-fit" measure associated with `CASH` and `BAYES`, as there is of $\chi^2$-based statistics. Such a measure can, in principle, be computed by performing Monte Carlo simulations: one would repeatedly sample new datasets from the best-fit model, fit them, and note where the observed statistic lies within the derived distribution of statistics.

### 3.4. Optimization

Optimization is the act of minimizing $\chi^2$ or $-\log\mathcal{L}$ by varying the thawed parameters of the defined model. *Sherpa* provides a number of optimization methods, which can be classified in two broad categories: those which find a local minimum of the statistical surface in parameter space by moving along the local gradient of that surface, and those which examine large (hyper-)volumes of parameter space in a search for the global minimum (see Table 4††).

Below, we discuss the three optimization methods appropriate for finding local minima: `POWELL`, `SIMPLEX`, and `LEVENBERG-MARQUARDT`. Users should be acquainted with the (dis)advantages of each so as to make the best use of them. (For more information about *Sherpa's* other optimization methods, consult the *Sherpa* manual[2] and, *e.g.*, Press *et al.* 1992.[16] )

`POWELL`, a direction-set method in which the chosen statistic is minimized by varying each member of an (initially orthogonal) set of parameter-space vectors in turn, is *Sherpa's* default optimizer. Its advantages include the fact that no gradient calculations are required, and that it is a robust method, capable of finding minima even on complex statistical surfaces. (Also, unlike `LEVENBERG-MARQUARDT`, is can be used effectively with likelihood-based statistics.) Its primary disadvantage is that it is relatively slow.

In `SIMPLEX` optimization, the fit statistic is calculated at the $N+1$ vertices of a simplex in a $N$-dimensional parameter space, with the vertices being moved until the local minimum is bracketed. Its advantages include the fact that no gradient calculations are required, it can find minima of complex statistical surfaces, and it requires fewer model evaluations than `POWELL`. However, it is not as robust as `POWELL`. The `SIMPLEX` method is best-used when one starts the optimization close to the local minimum; for instance, it is a good optimizer to use in parameter estimation (see §3.6).

**Space does not permit us to provide details about Bayesian statistical methodology, which may be less familiar to some readers than the standard "frequentist" statistical paradigm. For an introduction to Bayesian statistics that is geared towards astrophysicists, see Loredo (1992).[14]

††Along with these *Sherpa* methods, a future release of *CIAO* will feature a stand-alone fitting application for low-counts data which uses Bayesian posterior sampling. See van Dyk et al. (2001).[15]

**Table 4.** Optimization methods in *Sherpa*.

| | |
|---|---|
| Local Minimum | `POWELL`, `SIMPLEX`, `LEVENBERG-MARQUARDT` |
| Global Minimum | `GRID(-POWELL)`, `MONTE(-POWELL)`, `SIMULATED ANNEALING` |

In `LEVENBERG-MARQUARDT` optimization, the local minimum is approached by taking steps in parameter space whose magnitudes $\delta\vec{x}$ are computed by solving the set of linear equations

$$\sum_{j=1}^{n} \alpha_{ij}(1+\lambda_{ij})\delta x_j = \beta_i \, ,$$

where

$$\alpha_{ij} \;=\; \sum_{k=1}^{n} \frac{1}{\sigma_k^2}\left[\frac{\partial M(\vec{x})}{\partial x_i}\frac{\partial M(\vec{x})}{\partial x_j}\right] \quad \text{and} \quad \beta_i \;=\; -\frac{1}{2}\frac{\partial \chi^2}{\partial x_i} \, ,$$

and $\lambda_{ij}$ is a matrix with non-zero diagonal elements whose magnitudes are inversely proportional to $\delta\vec{x}$. The primary advantage of `LEVENBERG-MARQUARDT` optimization is speed, while its disadvantages include the fact that a gradient computation is required, that it is appropriate for use with $\chi^2$-based statistics only, and that it is less robust when applied to optimization on a complex statistical surface. (To circumvent the third issue, we have introduced the option that the optimization method may be switched from `LEVENBERG-MARQUARDT` to `SIMPLEX` close to the minimum, where the disadvantages of `LEVENBERG-MARQUARDT` become more readily apparent.)

### 3.5. Model Comparison

Often, a user will fit more than one parametrized model to a given dataset, and will wish to compare the best-fit results of each. For instance, one may fit two continuum models to data, and need to decide whether the improvement in the fit statistic that is observed when using the more complex model is attributable to chance. To make this decision, one uses a model comparison test to yield either: (1) the frequentist test significance, $\alpha$, which is the probability of selecting the alternative (more complex) model $M_1$ when in fact the null hypothesis $M_0$ is correct; or (2) the Bayesian odds, the ratio of model posterior probabilities for $M_1$ and $M_0$. If the prior probability distribution for a model's parameter values is constant, then its posterior probability is proportional to the integral of the likelihood function $\mathcal{L}$ over parameter space.

The model comparison test that is currently available to the *Sherpa* user is the $\chi^2$ Goodness-of-Fit (GOF) test, an alternative-free test. The next version of *Sherpa* will also contain the Maximum Likelihood Ratio (MLR) test and the $F$-test. Methods of model comparison that may be included in future versions of *Sherpa* include: using simulations to determine model comparison test statistics numerically when the conditions for using an analytic test are not fulfilled; computing the Bayesian odds using the Laplace approximation[17]; and computing the Bayesian odds via numerical integration. We note that these new techniques, in addition to assisting the comparison of models, would also be useful for parameter estimation.

### 3.6. Parameter Estimation

Once one has selected a best-fit model, the next question is: what are the errors on the model parameters, *i.e.* what are the confidence intervals associated with each model parameter? In general, a frequentist statistician can determine possible intervals by repeatedly simulating data from the best-fit model, fitting these data, and determining the distribution of best-fit values for each model parameter.[‡‡] The central 68% of each distribution can then be deemed the $1\sigma$ confidence interval. However, simulations are computationally expensive, and if: (1) the $\chi^2$ or $\log\mathcal{L}$ surface is approximately shaped like a multi-dimensional paraboloid (*i.e.* contours of constant $\chi^2$ or $\log\mathcal{L}$ appear ellipsoidal in two-dimensional plots), and (2) the best-fit point is sufficiently far from parameter space boundaries, then confidence intervals may be estimated by examining the statistical surface itself.

[‡‡] A Bayesian would adapt methods mentioned in the previous section–using numerical integration or the Laplace approximation, *etc.*–to the problem of parameter estimation. Thus in the remainder of this subsection, we only discuss frequentist parameter estimation methods.

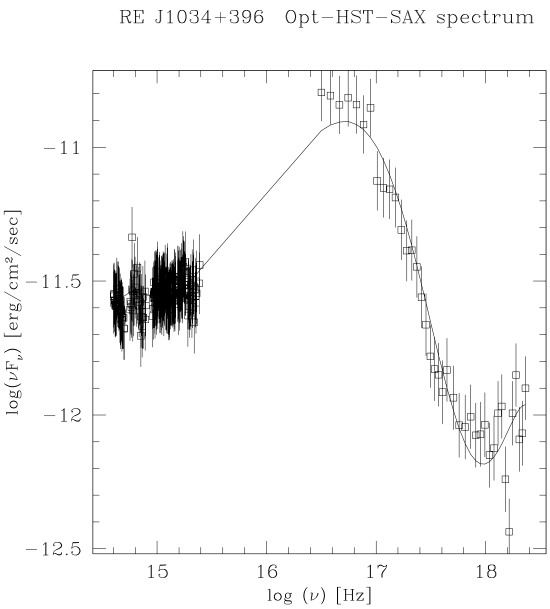


**Figure 1.** Best-fit of two polynomial functions to data of the narrow-line Seyfert 1 galaxy RE J1034+396, observed by the William Herschel 4.2m Telescope, *HST*, and *BeppoSAX*.

*Sherpa* currently features three parameter estimation methods appropriate for use when the statistical surface is "well-behaved": `UNCERTAINTY`, `PROJECTION`, and `COVARIANCE`. (In addition, one can make one- or two-dimensional plots showing the fit statistic value as a function of parameter value[s].) With `UNCERTAINTY`, the error for a particular thawed parameter is estimated by varying its value (while holding all other parameter values fixed to their best-fit values) until the fit statistic increases by a preset amount from its minimum value (*e.g.* $\Delta\chi^2 = 1$ for $1\sigma$). `PROJECTION` is similar to `UNCERTAINTY`, except that the values of all other parameters are allowed to float to new best-fit values. With `COVARIANCE`, errors are estimated by calculating the covariance matrix, the inverse of the matrix of statistical surface second derivatives at the best-fit point. Each of these methods has distinct (dis)advantages: for example, `UNCERTAINTY`, while fast, will generally underestimate an interval's size if the parameter is correlated with other parameters; and `PROJECTION` provides a means to visualize the surface and can be used even if the model parameters are correlated, but is in the strictest statistical sense no more accurate than the much faster `COVARIANCE` method (which is itself not useful for visualization).

## 4. EXAMPLES OF SHERPA ANALYSES

In this section, we present four examples of *Sherpa* analyses. We note that space limitations prevent us from showing all but a few commands that are used in these analyses; for full scripts, plus scripts showing other analyses, please consult the *Sherpa* analysis threads.[18]

### 4.1. Multi-wavelength analysis of spectra

In this example, we analyze flux data ($\log[\nu F_\nu]$) of RE J1034+396,[19] a low-redshift, narrow-line Seyfert 1 galaxy. The data were collected by the William Herschel 4.2m Telescope, *HST*, and *BeppoSAX*. Because the data are not sampled from a Poisson distribution, the errors must be input or specified; here, we assume that the error on the flux is 1%:

```
sherpa> errors = 0.01*data
```

The observed "blue bump" is modeled in *Sherpa* with two polynomial functions; the final fit is shown in Figure 1.

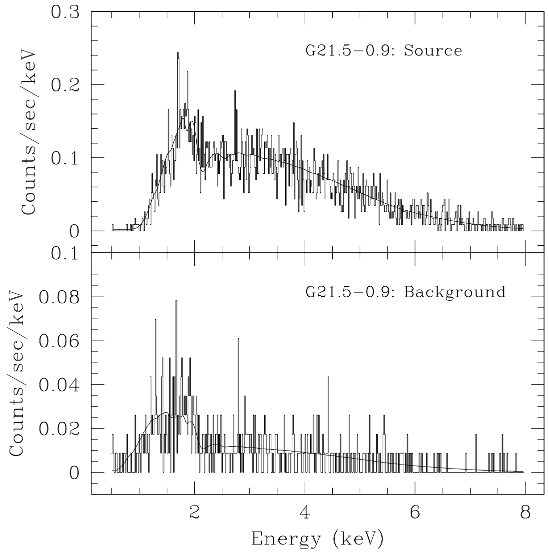


**Figure 2.** *Top:* Best-fit of a power-law times galactic absorption model to the source spectrum of supernova remnant G21.5–0.9. *Bottom:* Best-fit of a different power-law times galactic absorption model fit to a background spectrum extracted near G21.5–0.9.

### 4.2. Simultaneous analysis of source and background data

In this example, we analyze *Chandra* source and background spectra of the supernova remnant G21.5–0.9. In our analysis, we assume a power-law times galactic absorption model, with different model parameters for the source and background:

```
sherpa> source = xswabs[sabs]*pow[sp]   # uses the XSPEC wabs absorption model
sherpa> bg = xswabs[babs]*pow[bp]
```

We model the source and background data separately, rather than subtract the background data from the source data, because the low background count-rate. This low count-rate also motivates the use of the Cash statistic:

```
sherpa> statistic cash
```

The final fit is shown below, and in Figure 2.

```
sherpa> fit
 powll: v1.2
 powll:     converged to minimum =    -7.01375E+03 at iteration =     28
 powll:   final function value    =    -7.01375E+03
          sabs.nH  2.38646  10^22/cm^2
            sp.gamma  1.50622
            sp.ampl  0.00201939
          babs.nH  0.629181  10^22/cm^2
            bp.gamma  1.0345
            bp.ampl  0.000101356
```

### 4.3. Analysis of Chandra grating data

This example shows the analysis of *Chandra* grating spectra of the bright X-ray source Capella, which have been stored in one Type II PHA file. We concentrate on the first-order High Energy Grating (HEG) and Medium Energy Grating (MEG) spectra, which are input into *Sherpa* as datasets 3 (HEG -1), 4 (HEG +1), 9 (MEG -1), and 10 (MEG +1). Because the input Type II PHA data file contains columns defining the wavelengths for each bin, the analysis is assumed to be in wavelength-space. We examine only data between 6.7 and 6.8 Å:

```
sherpa> notice allsets wave 6.7:6.8
```

We then fit a normalized Gaussian function to the observed line:

```
sherpa> source 3,4  = ngauss[hg1] + const[co]
sherpa> source 9,10 = ngauss[mg1] + co
```

where the constant function represents the background. Because the line flux will be same in a contemporaneous MEG/HEG observation, the amplitudes are linked:

```
sherpa> mg1.ampl => hg1.ampl
```

Other parameters are not linked because of uncertainties in calibration. Note that we use only grating ARFs in this analysis; we could also model the line profile with a delta function and use both grating ARFs and RMFs. After the fit (Figure 3), we identify the most likely transition which gives rise to the observed line using GUIDE, a S–Lang-based extension to *Sherpa* which acts as an interface to the Atomic Plasma Emission Database (APED):[20]

```
sherpa> import("guide")
sherpa> identify(6.40)
Found 9 lines.
  Lambda  --    Ion     UpperLev  LowerLev Emis(ph cm^3/s) @ Peak Temp
   ...
   6.7403 --    Si XIII      2 ->      1,  3.548e-17 @ logT =  7.00
   ...
```

### 4.4. Analysis of two-dimensional data

In this last example, we demonstrate how one can model the spatial distribution of hot gas in the X-ray cluster MS 2137.3-2353, observed by *Chandra*. After the data are entered, we display them using *SAODS9*; we then load the three point source regions into *SAODS9* that we will use to interactively filter the data:

```
sherpa> ignore image
```

One could also use the regions to filter the data directly at the *Sherpa* command line:

```
sherpa> ignore filter ellipse(300.14946,299.8716,20.128119,16.76774,94.648547) + \
                      ellipse(431.96938,371.1944, 7.251325, 4.77655,11.890284) + \
                      ellipse(212.26666,145.8972, 4.744431, 4.33702,71.631001)
```

This command filters out the data *within* the defined regions. The remaining data, which represent only the intra-cluster gas, are then fit with a two-dimensional beta model profile. See Figure 4.

## ACKNOWLEDGMENTS

We would like to thank Mark Birkinshaw, William Joye, Malin Ljungberg, and Michael Noble for past and present contributions to *Sherpa*'s development. We would also like to thank Holly Jessop for her tireless work maintaining the *Sherpa* manuals and threads. The *Sherpa* project is supported by the *Chandra* X-ray Center under NASA contract NAS8-39073.

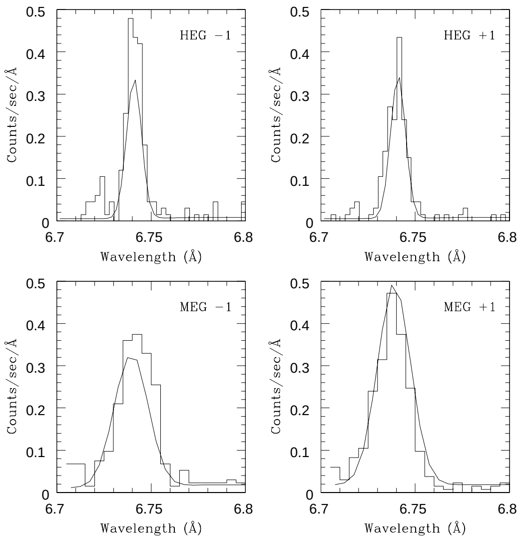


**Figure 3.** Best-fit of a normalized Gaussian function to an emission line (Si XIII 2→1 at 6.7403 Å) observed in four first-order HEG and MEG *Chandra* grating spectra of Capella.

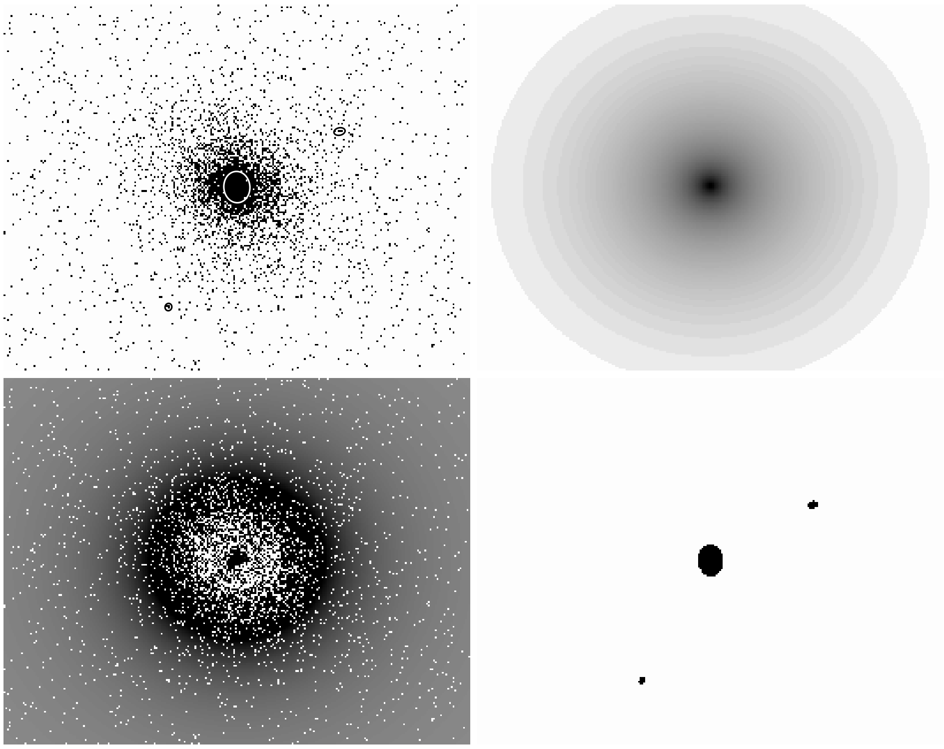

**Figure 4.** *Top Left: Chandra* ACIS-S data of X-ray cluster MS 2137.3-2353, with *SAODS9* source regions superimposed. *Top Right:* Best-fit of a two-dimensional beta model to the filtered data. *Bottom Left:* Residuals (in units of $\sigma$) of the best fit. *Bottom Right:* The applied filter; the data within the ovals were excluded from the fit.

## REFERENCES


1. `http://asc.harvard.edu/ciao/index.html`
2. `http://asc.harvard.edu/ciao/download/doc/sherpa_html_manual/index.html`
3. S. Doe, M. Ljungberg, A. Siemiginowska, & W. Joye, "Fitting and Modeling of AXAF Data with the ASC Fitting Application," in *Astronomical Data Analysis Software and Systems VII*, R. Albrecht, R. N. Hook, & H. A. Bushouse, ed., pp. 157–160, ASP, San Francisco, 1998.
4. J. McDowell, these proceedings; also `http://asc.harvard.edu/ciao/ahelp/dm.html`
5. `http://asc.harvard.edu/ciao/download/doc/chips_html_manual/index.html`
6. R. Lupton & P. Monger, `http://www.astro.princeton.edu/~rhl/sm/`
7. W. Joye & E. Mandel, `http://hea-www.harvard.edu/RD/index.html`
8. J. Davis, `http://www.s-lang.org/`
9. S. Doe, M. Noble, & R. Smith, "Interactive Analysis and Scripting in CIAO 2.0," in *Astronomical Data Analysis Software and Systems X*, F. R. Harnden, F. A. Primini, & H. E. Payne, ed., *in press*, ASP, San Francisco, 2001.
10. K. A. Arnaud, "XSPEC: The First Ten Years," in *Astronomical Data Analysis Software and Systems V*, G. H. Jacoby & J. Barnes, ed., pp. 17–20, ASP, San Francisco, 1996.
11. N. Gehrels, "Confidence limits for small numbers of events in astrophysical data," *Ap. J.*, **303**, pp. 336–346, 1986.
12. K. Kearns, F. Primini, & D. Alexander, "Bias-Free Parameter Estimation with Few Counts, by Iterative Chi-Squared Minimization," in *Astronomical Data Analysis Software and Systems IV*, R. A. Shaw, H. E. Payne, & J. J. E. Hayes, ed., pp. 331–334, ASP, San Francisco, 1995.
13. W. Cash, "Parameter estimation in astronomy through application of the likelihood ratio," *Ap. J.*, **228**, pp. 939–947, 1979.
14. T. J. Loredo, "The Promise of Bayesian Inference for Astrophysics," in *Statistical Challenges in Modern Astronomy*, E. D. Feigelson & G. J. Babu, ed., pp. 275–297, Springer-Verlag, New York, 1992.
15. D. A. van Dyk, A. Connors, V. L. Kashyap, & A. Siemiginowska, "Analysis of Energy Spectra with Low Photon Counts via Bayesian Posterior Simulation," *Ap. J.*, **548**, pp. 224–243, 2001.
16. W. H. Press, S. A. Teukolsky, W. T. Vetterling, & B. P. Flannery, *Numerical Recipes*, Cambridge University Press, Cambridge, 1992.
17. T. J. Loredo & D. Q. Lamb, "Establishing the existence of lines in gamma-ray bursts," in *Gamma-ray Bursts: Observations, Analyses, and Theories*, C. Ho, R. Epstein, & E. Fenimore, ed., pp. 414–415, Cambridge University Press, Cambridge, 1992.
18. `http://asc.harvard.edu/ciao/documents_threads.html`
19. E. M. Puchnarewicz, *et al.*, "Constraining the Black Hole Mass and Accretion Rate in the Narrow-Line Seyfert 1 Galaxy RE J1034+396," *Ap. J.*, **550**, pp. 644-654, 2001.
20. R. K. Smith, N. S. Brickhouse, D. A. Liedahl, & J. C. Raymond, "Collisional Plasma Models with APEC/APED: Emission Line Diagnostics of Hydrogen-like and Helium-like Ions," *Ap. J. Lett.*, *in press*, 2001.