Astronomical Data Analysis Software and Systems XIII
ASP Conference Series, Vol. 314, 2004
F. Ochsenbein, M. Allen, and D. Egret, eds.

Extending Sherpa Data Analysis with S-Lang

S. Doe, P. Freeman, R. Smith, D. Burke, A. Siemiginowska

Harvard–Smithsonian Center for Astrophysics, 60 Garden Street, Cambridge, MA 02138

Abstract. *Sherpa* is the fitting application of the Chandra Interactive Analysis of Observations (CIAO) package. In this paper, we discuss how we have extended the capabilities of *Sherpa* in CIAO3.0 with the S–Lang programming language. Users are now able to load *Sherpa* data sets into S–Lang variables, and vice versa. Users also now have access to many new S–Lang functions which are the equivalents of *Sherpa* commands. Thus, it is now possible for users to write their own S–Lang scripts that load data into *Sherpa*, perform fits, and then copy the fit results out to S–Lang variables for further analysis in their scripts. Such scripts can be run from within *Sherpa*; it is also possible to load a *Sherpa* runtime module into another S–Lang application, and then run the same scripts in that application. We have also made it possible for users to run S–Lang scripts that modify the appearance of plots generated by *Sherpa* plotting commands. Finally, settings that affect the execution of many *Sherpa* commands can be changed via S–Lang variables. These settings can be stored and read back in at the start of a *Sherpa* session. These extensions to *Sherpa* follow from our efforts to embed S–Lang in CIAO, first described in Doe et al. (2001).

1. Introduction—What Are Sherpa And S–Lang?

Sherpa is the modeling and fitting tool of CIAO software package. We have developed it with the primary goal that a user should be able to take full advantage of Chandra's unprecedented observational capabilities and be able to analyze data in up to four dimensions (energy E or wavelength λ, time t, and spatial location [x,y]) with a wide variety of models, optimization methods, and fit statistics. We have also made it as general as possible, so that users can analyze data from other X–ray missions (*e.g., ROSAT, XMM*) and from other wavebands.

S–Lang is an interpreted language created by John Davis of the Center for Space Research at MIT. S–Lang is a popular language, with several hundred users, and several applications that use it as an extension language. Embedding such a language into an application can enhance its flexibility and power by allowing users to more easily extend its capabilities. We have added S–Lang to CIAO, where it is currently used most heavily in ChIPS (Chandra Imaging and Plotting Software) and *Sherpa*.

© Copyright 2004 Astronomical Society of the Pacific. All rights reserved.

2. S–Lang Access To Sherpa Data

Embedding S–Lang in *Sherpa* was an important first step. This allows users to call their own S–Lang functions from *Sherpa*. But simply calling new S–Lang functions is of limited use unless such functions can analyze data that have already been read into *Sherpa*.

Thus, we have added S–Lang functions that provide interfaces to *Sherpa* data structures. Among them are functions that copy *Sherpa* data from C++ to S–Lang variables. S–Lang scripts can now obtain copies of *Sherpa* data sets, and their associated errors and weights, as well as model values calculated from models fit to the data.

Data residing in S–Lang variables can also be copied back into *Sherpa*. S–Lang scripts can now copy data from a *Sherpa* data set, perform some analysis on the data, and then copy the data back into *Sherpa*. A *Sherpa* model can then be fit to this modified data set. We also provide other S–Lang functions to access other *Sherpa* information (*e.g.,* values and ranges or model parameters).

The *Sherpa* documentation[1] has a complete list of the S–Lang functions we provide, including examples using these functions.

These functions are available from the *Sherpa* command line; they are also available from a S–Lang module we distribute along with *Sherpa* itself. S–Lang modules are shared objects which can be dynamically linked to other S–Lang applications. *Sherpa* S–Lang functions, and thus, a *Sherpa* fitting session, can be run from any other S–Lang application (*e.g.,* ChIPS, slsh).

3. Interface to ChIPS

ChIPS is the Chandra Imaging and Plotting Software. S–Lang has also been embedded in ChIPS. For the plots generated from *Sherpa*, we now use S–Lang as the interface between *Sherpa* and ChIPS.

Sherpa has a number of S–Lang scripts that create certain types of plots (*e.g.,* plots of data with error bars, plots with data and model values overplotted, plots of residuals, and so on). These scripts copy the appropriate data, errors and model values from *Sherpa* and send them to ChIPS; the scripts then call various S–Lang functions which are equivalent to ChIPS plotting commands.

The plots *Sherpa* generates all have different default appearances. To allow users to change these defaults, we provide configuration variables that govern the attributes of the plots. Examples of such attributes include line style, point style, presence of x- and y-error bars, colors for line and points, *etc.*

The *Sherpa* documentation includes a list of all configuration variables and examples illustrating their use.

4. User Models in S–Lang

Sherpa provides a number of mathematical and physical models to which data can be fit. But *Sherpa* cannot possibly provide every model that every user

[1] `http://cxc.harvard.edu/sherpa/`

might want. Therefore, we provide an interface to models written in S–Lang. Users can write their own models in S–Lang and then load the model into *Sherpa*. Once the model is in *Sherpa*, it can be used just as a native *Sherpa* model would be—the syntax is identical.

5. Extending Sherpa

The enhancements we have added to *Sherpa*—S–Lang access to *Sherpa* data sets, a S–Lang interface to ChIPS, and a S–Lang interface to user models—help users to more easily write their own extensions to *Sherpa*. Such extensions could be functions to prepare *Sherpa* data sets for analysis, to modify plots produced by *Sherpa* plotting scripts, or to analyze data and fit results in ways that cannot be done with native *Sherpa* functions.

The CIAO Web pages include pointers to a number of "threads"[2] that (among many other things) show how to use S–Lang to extend *Sherpa*. These topics include:

- Introduction to the *Sherpa* S-Lang Module.
- Accessing fit results using S–Lang.
- Advanced customization of *Sherpa* plots.

6. Calculating K-corrections Using S-Lang and Sherpa

We conclude with a condensed version of another thread from our Web page—a thread that shows how to use *Sherpa*, with S–Lang extensions, to calculate k–corrections for any of the spectral models available from *Sherpa*. As an example we calculate the corrections necessary for the flux measured in the 0.5–2.0 keV energy range from a thermal plasma (as described by the Mewe–Kaastra–Liedahl—a.k.a. MEKAL—model) for a range of redshifts and gas temperatures. The aim is to emulate the figure presented in Appendix B of Jones et al. (1998).

The first step is to create an instance of the `xsmekal` model, and set the plasma temperature and metal abundance to be 7 keV and 0.3 times solar respectively.

```
sherpa> paramprompt off
Model parameter prompting is off
sherpa> source = xsmekal[plasma]
sherpa> plasma.kt = 7
sherpa> plasma.abund = 0.3
```

For the rest-frame energy range (0.5-2.0 keV) and redshift range (0-2) considered in this example, we need to ensure that the model is evaluated over at least 0.5 to 4.0 keV. It does not matter if the energy range is larger than that, so we chose 0.01 to 10 keV, with a grid spacing of 0.01 keV.

```
sherpa> dataspace (0.01:10:0.01) histogram
```

[2] `http://cxc.harvard.edu/sherpa/threads/slang.html`

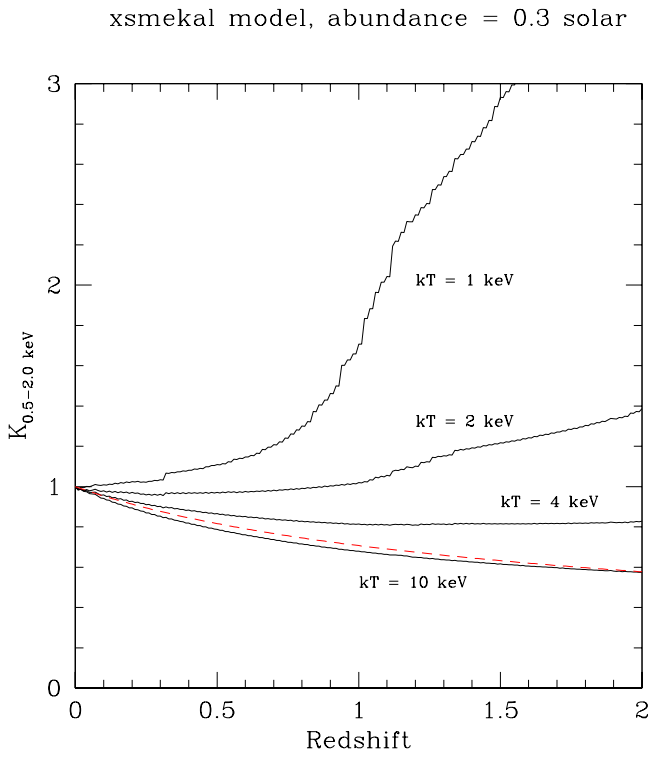

Figure 1. K–Correction, at temperatures of 1, 2, 4, and 10 keV.

We can then use the `calc_kcorr` function from the *Sherpa* utility package (`sherpa_utils.sl`, available for download from the *Sherpa* Web page). This function can return the k–correction for a single redshift, or a range of redshifts. Here we calculate the correction values for the redshift range 0 to 2.

```
sherpa> z = [0.0:2.1:0.1]
sherpa> kc = calc_kcorr( 1, z, 0.5, 2.0 )
```

This method can then be used to calculate the k–correction at plasma temperatures of 1, 2, 4, and 10 keV. The result is shown in Figure 1, which resembles that of Jones.

Acknowledgments. This project is supported by the Chandra X-ray Center under NASA contract NAS8-39073.

References

Doe, S., Noble, M., & Smith, R. 2001, in ASP Conf. Ser., Vol. 238, ADASS X, ed. F. R. Harnden, Jr., F. A. Primini, & H. E. Payne (San Francisco: ASP), 310

Jones, L. R., Scharf, C., Ebeling, H., Perlman, E., Wegner, G., Malkan, M., & Horner, D. 1998, ApJ, 495, 100