Astronomical Data Analysis Software and Systems XVI
ASP Conference Series, Vol. 376, 2007
R. A. Shaw, F. Hill and D. J. Bell, eds.

Developing Sherpa with Python

S. Doe, D. Nguyen, C. Stawarz, B. Refsdal, A. Siemiginowska, D. Burke, I. Evans, J. Evans, J. McDowell

Smithsonian Astrophysical Observatory, Cambridge, MA, USA

J. Houck, M. Nowak

MIT Kavli Institute for Astrophysics and Space Research, Cambridge, MA, USA

Abstract. *Sherpa* is the general purpose fitting and modeling application for CIAO, the Chandra Interactive Analysis of Observations system. We have modified the original design and implemented a new version in Python. This version will be part of the upcoming CIAO4.0 release. We have previously presented a modular, flexible design for CIAO4.0 with the goal of packaging many models, fitting methods and statistics for analysis of astronomical data. The new design promised to be more robust than the previous *Sherpa*, and more easily extensible with user-written scripts. (We already see some sign of this, in that there were 50,000 lines of code in the CIAO3.0 implementation; with our new, cleaner design, implemented in Python, only half that number of lines were required.) We present the latest updates to our design, and our progress developing *Sherpa*. A major feature of this work has been the use of Python to implement the data structures from our design. Each part of *Sherpa*—models, fitting methods, statistics, and so on—has been implemented as a Python module. We have also developed application code to bind together data, models, statistics, and fitting methods for performing fits to data, as well as a high-level UI that makes it simple for users to read in data, define models, and perform fits. Working in Python has been a great aid in speeding development of *Sherpa*. We expect that Python will also simplify extending and maintaining the *Sherpa* code base, as well as making it possible to interoperate with other Python-based astronomy packages. To make *Sherpa* fully accessible to S–Lang users, we use PySL, a new package that is an interface between Python and S–Lang. Users are now able to import other Python or S–Lang modules to extend *Sherpa*; in addition, users may write and use scripts of their own, written in either Python or S–Lang.

1. Design and Python Implementation

The design of *Sherpa* is in three layers, as shown in Figure 1. The base modules constitute the first layer; each module provides a set of functions (models, optimization functions, statistics, and also astronomy-specific functions) that are needed by the application. But each base module is independent, and can be used by itself in other programs. The C++ or Fortran functions they wrap can be directly linked to standalone programs (e.g., a C program that needs only to calculate *Sherpa* models), without having to link to Python or any other *Sherpa* modules. Any of the modules in this layer can also be loaded into Python. We

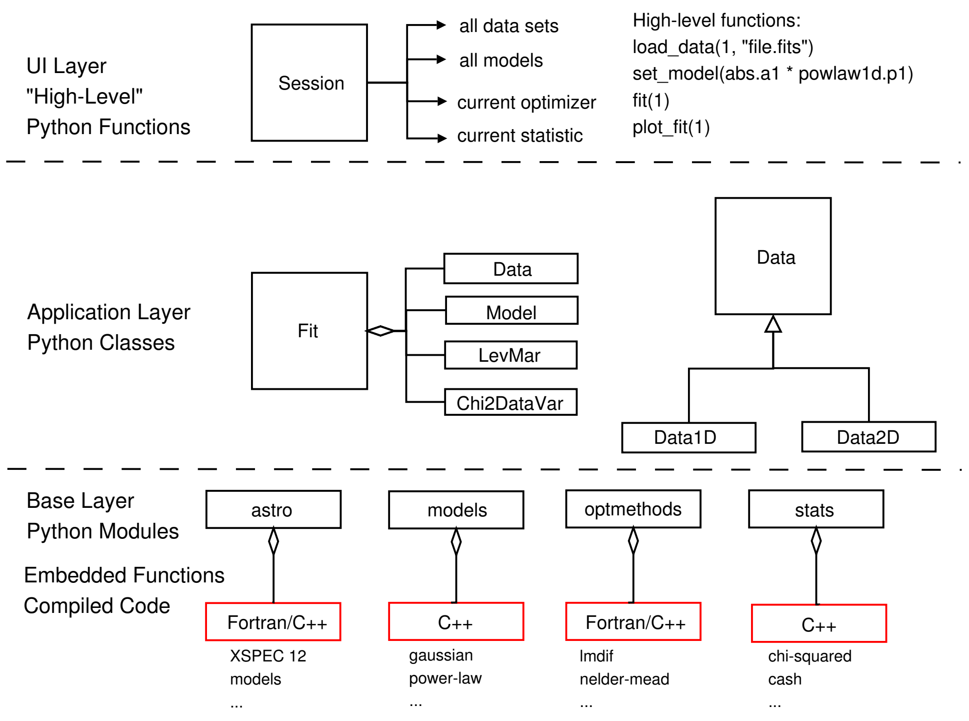

Figure 1. Base, application, and UI layers of the *Sherpa* design.

use NumPy[1] to provide support for numerical arrays. NumPy is the sole Python dependency for the base modules.

The second layer is the application layer. This layer contains the logic needed for the *Sherpa* application to fit models to data (along with other science functions). *Data* classes encapsulate data sets; *Fit* containers group data and models together, for the duration of a fit.

The third layer is the UI layer. This layer contains master lists of all the data sets loaded into *Sherpa*, all the models that are tied to data sets, and pointers to the currently used optimization method and fit statistic. It also provides "high-level" functions to make it easier for users to read in data, create and assign models to data sets, fit models to data, and visualize results.

2. Python and S–Lang

Any base module, or all of the *Sherpa* modules, can be imported into Python. For users interested in the whole package, we provide a module (sherpa.astro.ui) that automatically loads all the *Sherpa* modules, including the modules specific to astronomy and the high-level UI, into Python. As Figure 2 shows, we can load *Sherpa* into a variety of Python environments. *Sherpa* can be directly loaded

[1]http://numpy.scipy.org

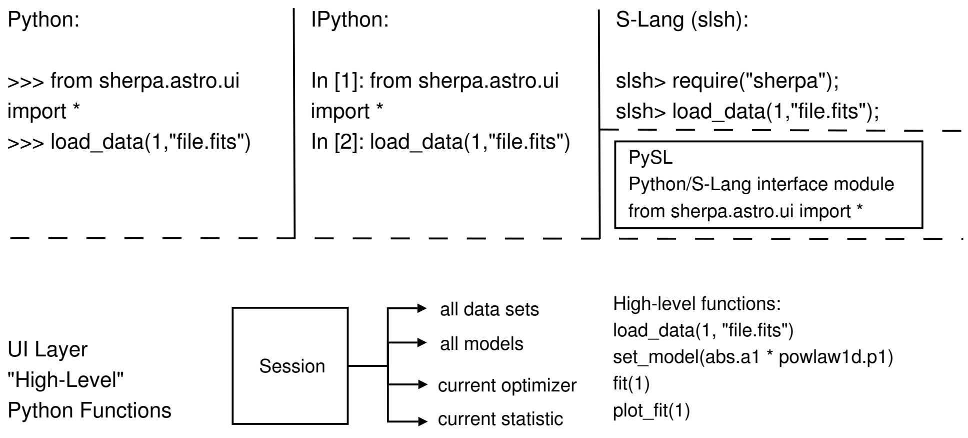
Python:
>>> from sherpa.astro.ui import *
>>> load_data(1,"file.fits")
IPython:
In [1]: from sherpa.astro.ui import *
In [2]: load_data(1,"file.fits")
S-Lang (slsh):
slsh> require("sherpa");
slsh> load_data(1,"file.fits");
PySL
Python/S-Lang interface module
from sherpa.astro.ui import *
UI Layer
"High-Level"
Python Functions
Session
all data sets
all models
current optimizer
current statistic
High-level functions:
load_data(1, "file.fits")
set_model(abs.a1 * powlaw1d.p1)
fit(1)
plot_fit(1)

Figure 2. *Sherpa* modules in a variety of environments.

into Python, or can be loaded into other Python environments that provide an interactive UI, such as IPython.

We also provide support for our existing S–Lang users. S–Lang[2] is an interpreted language that can be embedded into other applications, and has been used to extend CIAO 3.0. The new package PySL[3] is a general interface between Python and S–Lang. We use PySL to make the *Sherpa* package accessible from S–Lang. We also provide a S–Lang module that automatically loads PySL and *Sherpa*, and wraps all the high-level UI functions. Ideally, S–Lang users should not need to know that *Sherpa* is running Python under the hood.

3. An Example Sherpa Session

A short session illustrating *Sherpa* functions is shown here. A file containing an X-ray spectrum is read in, and a model consisting of an absorption model and a power-law is fit to the data. The results are shown in Figure 3.

```
>>> load_pha("3c273.pi")
>>> notice_id(1, 0.1, 6.0)
>>> subtract()
>>> set_model(xsphabs.abs1 * powlaw1d.p1)
>>> abs1.nH = 0.07
>>> freeze(abs1.nH)
>>> fit()
```

[2]http://www.s-lang.org

[3]http://software.pseudogreen.org/pysl

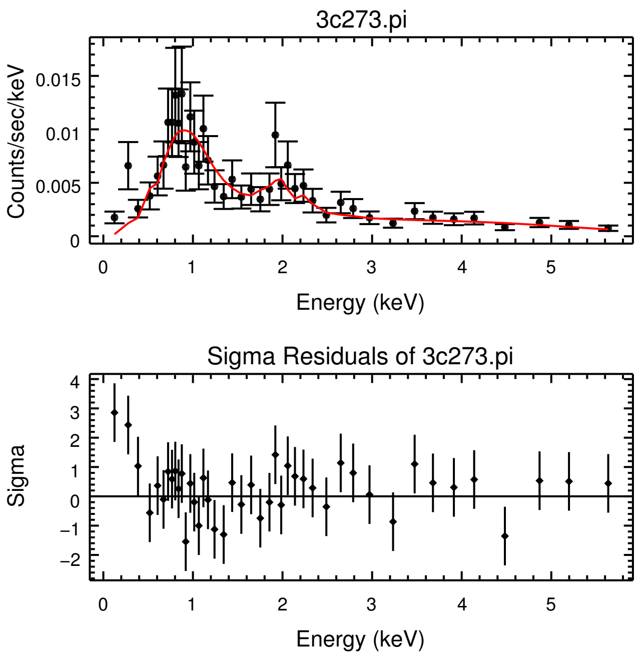

Figure 3. Plot of data, model and residuals created by a *Sherpa* fitting session.

4. Interfaces to non-CIAO Packages

Sherpa will be released as part of CIAO 4.0. When used within CIAO, *Sherpa* automatically imports Python modules that are interfaces to CRATES (the new CIAO Data Model interface), ChIPS (the Chandra Imaging and Plotting System), and DS9 (the SAO imager). However, we have designed *Sherpa* so that it can be built and used without other CIAO components. If *Sherpa* alone is present, then the user may substitute PyFITS for CRATES, and *matplotlib* for ChIPS. We include "back-end" interfaces to PyFITS and *matplotlib*. If these modules are present, they are detected at runtime. If the rest of CIAO is not present, PyFITS and *matplotlib* are automatically loaded.

Our new Python implementation has helped us design a version of *Sherpa* that will be easier to build and run as a stand-alone application.

Acknowledgments. Support of the development of *Sherpa* is provided by the National Space and Aeronautics Administration through the Chandra X-ray Center, which is operated by the Smithsonian Astrophysical Observatory for and on behalf of NASA through contract NAS–03060.