Astronomical Data Analysis Software and Systems: XXIV
ASP Conference Series, Vol. 495
A. R. Taylor and E. Rosolowsky, eds.

Sherpa. On the Move to Open Collaborative, Development

Omar Laurino,[1] Aneta Siemiginowska,[1] Janet Evans,[1] Thomas Aldcroft,[1] Douglas J. Burke,[1] Jonathan McDowell,[1] Warren McLaughlin,[1] and Dan Nguyen[1]

[1]*Smithsonian Astrophysical Observatory. 60 Garden St, Cambridge MA 02474, USA.*

Abstract. Sherpa is the Chandra Interactive Analysis of Observations (CIAO) modeling and fitting application. Written in Python, with efficient C, C++, and Fortran extensions, Sherpa enables the user to construct complex models from simple definitions and fit those models to data, using a variety of statistics and optimization methods. Sherpa is a general-purpose fitting engine with advanced capabilities, and has been used as a backend for the development of new applications like Iris, the Virtual Astronomical Observatory spectral energy distribution builder and analyzer. However, building and installing Sherpa as a standalone Python package was problematic, and such a build would not maintain all of the Sherpa capabilities. For version 4.7 (December 2014) Sherpa's build scripts have been completely rewritten, standardized, and made independent of CIAO, so that Sherpa can now be built as a fully functional standalone Python package, and yet allow users the flexibility they need in order to build Sherpa in customized environments.

1. Sherpa

Sherpa [ascl:1107.005] allows users to fit 1-D and 2-D data sets with many advanced capabilities: one can arbitrarily combine any number of model components, simultaneously fit different datasets, link model parameters between different model components, import and use custom models as tables or Python functions, choose appropriate statistics for modeling Poisson or Gaussian data, import new statistics, including priors, visualize a parameter space with simulations or using 1-D/2-D cuts, calculate confidence levels on the best-fit model parameters, choose among robust optimization methods for the fit (Levenberg-Marquardt, Nelder-Mead Simplex or Monte Carlo/Differential Evolution), and perform Bayesian analysis with Poisson Likelihood and priors, using Metropolis or Metropolis-Hastings algorithm in the Markov-Chain Monte Carlo framework (Freeman et al. 2001; Refsdal et al. 2009; Siemiginowska et al. 2011).

Sherpa is available on Linux and OS X systems, and it has been used as a backend fitting engine for Iris [ascl:1205.007], a spectral energy distribution builder and analyser (Laurino et al. 2014).

2. Decoupling Sherpa from CIAO

Since version 4.5, Sherpa has been shipped both as part of CIAO (Fruscione et al. 2006), and as a standalone package. However, it was problematic to build Sherpa in a standalone fashion and to make it work alongside other Python tools not contained in the

CIAO package. For version 4.7 Sherpa's build scripts have been completely rewritten, standardized, and made independent of CIAO, so that Sherpa can now be easily built in many different environments, or installed as a binary. When building from source, users can customize the build configuration if necessary, and it is straightforward to integrate Sherpa with other Python packages and tools.

The next sections explore in detail the different options to build and install Sherpa.

3. Sherpa binary and source distributions

Source distributions allow users to build software on many different systems. For this reason we standardize the Sherpa source distribution so that it can be easily built on several platforms. Sherpa has been successfully built for Linux on ARM, x86 and x86_64 architectures, as well as different versions of OS X.

Sherpa is a Python package with extensions written in C, C++, and Fortran, leveraging the strengths of these programming languages especially in terms of computational efficiency. However, this also means that in order to build Sherpa one needs a robust development environment that may be out of reach for end users. For this reason, and in the tradition of CIAO, we also distribute Sherpa in a binary form.

The emergence of novel tools in the Python ecosystem makes it much easier to distribute Sherpa in a binary form that works *out of the box* but that can also allow users to install Sherpa alongside other software tools. In particular, Continuum Analytics[1] provides the Anaconda Python distribution, equipped with an advanced package manager, `conda`, that provides a platform-independent, flexible system for creating isolated virtual environments that comprise both Python and native packages. Moreover, the Miniconda distribution provides a lightweight installation of `conda` that can be used to create such environments without the large footprint of the entire Anaconda installation. We provide `conda` binary packages for Sherpa and some of its dependencies so that users can very easily install Sherpa in an existing Anaconda distribution. We also provide installers that set up a whole self-contained, extensible virtual environment for Sherpa in a few minutes.

For systems or users for which the standard binary installation is not practical or possible, as well as for users wanting to customize the Sherpa build, we also provide a source distribution.

4. How to install Sherpa

Sherpa can also be installed by Anaconda users by adding a special channel to their Anaconda installation[2], and then by using the `conda` command to install Sherpa and its dependencies:

```
$ conda config --add channels http://conda.binstar.org/sherpa
$ conda install sherpa
```

[1] http://www.continuum.io

[2] In the future Sherpa might become part of Anaconda itself or served by the default conda channel.

Users that do not have Anaconda can use the Sherpa standalone installer, a `bash` script that installs a self-contained Conda environment with Sherpa. Users can install other packages in this environment at will, using the `conda` package manager.

Sherpa can also be built from source. In this case users need to have a working development environment (compilers for C, C++, and Fortran, as well as tools like `make`).

As Sherpa is registered in the PyPI archive[3], building Sherpa from sources is as easy as:

```
$ pip install sherpa
```

Alternatively, users may download the Sherpa source code, unpack it, and run:

```
$ python setup.py install
```

These source build methods are rather general and allow Sherpa to be installed in any environment where any other Python package could be installed, including virtual environments or other Python distributions similar to Anaconda.

The next section explores some customization options available for building Sherpa.

5. Custom builds

There can be several reasons why a user might want to build Sherpa in a custom way. In order to make Sherpa builds self-consistent, Sherpa's source code ships with its own copy of the fast Fourier transform `fftw3` [ascl:1201.015] source code. However, in some cases users may want to link Sherpa to an existing installation of `fftw3`, in particular because this library can be built with platform-specific optimization options that make it much faster. Moreover, Sherpa can be interfaced to the X-ray models in the `xspec` package [ascl:9910.005] by enabling the relative Python extension.

In both cases, users can download the Sherpa source code from the Sherpa website, from PyPI, or from the Sherpa GitHub repository, unpack it, and edit the `setup.cfg` file to make Sherpa link its extensions to the `fftw3` or `xspec` libraries. The following command initiates the build and install Sherpa:

```
$ python setup.py install
```

6. Social development of Sherpa: GitHub

The work reported in this paper is aimed at opening the development of Sherpa to a broader community, so that scientists can extend the existing code to fit their requirements and also contribute to the development of Sherpa, using the social coding platform offered by GitHub[4].

Sherpa is extensible in many ways, and it allows users to employ their own Python functions as models, or to provide their own statistics or optimization methods. We are

[3] http://pypi.python.org/pypi/sherpa

[4] http://www.github.com/sherpa

planning to enhance the way users and other developers can extend Sherpa by providing a structured Software Development Kit that makes it easier to write, install, and test new code in Sherpa, as a result of the new development environment.

7. Conclusion

Starting with version 4.7, released in December 2014, Sherpa can be installed as a binary package or built from source in a straightforward way and independently of CIAO, and then seamlessly integrated with other Python tools and packages. For example, Sherpa can be used alongside `astropy` [ascl:1304.002] to perform fits in an `IPython`[5] Notebook.

Sherpa also provides immediate graphical feedback and supports both the CIAO plotting and imaging backends (`ChIPS`, `DS9` [ascl:0003.002]) as well as the popular `matplotlib`[6]. When using `matplotlib` in an `IPython` notebook, the plots can easily be embedded in the notebook itself.

For Input/Output, Sherpa supports the CIAO native backend (Crates) as well as PyFITS [ascl:1207.009].

This work is part of a larger framework that includes migrating the whole Sherpa codebase to GitHub, so that the wider astronomical community can be engaged in its development, as well as providing developers with a clean software framework for extending Sherpa.

More information can be found by visiting the Sherpa website[7] or the GitHub repository.

Acknowledgments. Support for the development of Sherpa is provided by the National Aeronautic and Space Administration through the Chandra X-ray Center, which is operated by the Smithsonian Astrophysical Observatory for and on behalf of NASA under contract NAS8-03060.

References

Freeman, P., Doe, S., & Siemiginowska, A. 2001, in Astronomical Data Analysis, vol. 4477 of Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, 76

Fruscione, A., et al. 2006, in Observatory Operations: Strategies, Processes, and Systems, vol. 6270 of Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series

Laurino, O., Budynkiewicz, J., DâĂŽÄôAbrusco, R., Bonaventura, N., Busko, I., Cresitello-Dittmar, M., Doe, S., Ebert, R., Evans, J., Norris, P., Pevunova, O., Refsdal, B., Thomas, B., & Thompson, R. 2014, Astronomy and Computing, . URL `http://www.sciencedirect.com/science/article/pii/S2213133714000328`

Refsdal, B. L., et al. 2009, in Proceedings of the 8th Python in Science Conference, Pasadena, CA, 2009, edited by G. Varoquaux, S. van der Walt and J. Millman, 51

Siemiginowska, A., Kashyap, V., Refsdal, B., van Dyk, D., Connors, A., & Park, T. 2011, in Astronomical Data Analysis Software and Systems XX, edited by I. N. Evans, A. Ac-

[5] http://www.ipython.org

[6] http://www.matplotlib.org

[7] http://cxc.cfa.harvard.edu/contrib/sherpa47

comazzi, D. J. Mink, & A. H. Rots, vol. 442 of Astronomical Society of the Pacific Conference Series, 439