Introduction to Sherpa

Aneta Siemiginowska

Chandra X-ray Center

http://cxc.harvard.edu/sherpa

CIAO Workshop — AAS Meeting Seattle — January 2019
Observations and Data Collection

Astrophysical process

Detector collects photons, adds noise

Random number of photons reach the detector

? draw conclusion about the astrophysical source

x-rays
Scientific Experiment

Observations

Data

Instrument specific processing software such as CIAO

Data Analysis:
source detections, source properties, image analysis, features, spectra, physical properties of the source, *apply models* to understand the source nature

Conclusions and Final Decision
Sherpa

Data:
arrays, spectra, light curves, images

Models:
parameterized description of the data

Fit Statistics

Fit Methods
(optimization methods)
Sherpa

- Data Input/Output
  - Astropy.io
  - PyCrates

- Models Library
  - Sherpa, XSPEC models, user models, templates

- Fit Statistics: Poisson and Gaussian likelihood

- Fit Methods:
  - minimization and sampling

- Visualization:
  - ChIPS, ds9, matplotlib

- Final Evaluation & Conclusions
Data in Sherpa

- **X-ray Spectra**
  typically PHA files with the RMF/ARF calibration files

- **X-ray Images**
  FITS images, exposure maps, PSF files

- **Lightcurves**
  FITS tables, ASCII files

- **Derived** functional description of the source:
  - Radial profile
  - Temperatures of stars
  - Source fluxes

- **Concepts of Source and Background data**

- **Any data array** that needs to be fit with a model
Data in Sherpa

• Load functions to input data:

  data: load_data, load_pha, load_arrays, load_ascii
  calibration: load_arf, load_rmf, load_multi_arfs, load_multi_rmf
  background: load_bkg, load_bkg_arf, load_bkg_rmf
  2D image: load_image, load_psf
  General type: load_table, load_table_model, load_user_model

• Multiple Datasets - data id

  Default data id = 1
  load_data(2, “data2.dat”, ncols=3)

• Filtering the data

  load_data expressions
  notice/ignore commands in Sherpa

Help file:
load_data( [id=1], filename, [options] )
load_image( [id=1], filename|IMAGECrate,[coord=”logical”] )

Examples:
load_data("src", "data.txt", ncols=3)
load_data("rprofile_mid.fits[cols RMID,SUR_BRI,SUR_BRI_ERR]")
load_data("image.fits")
load_image("image.fits", coord="world")

Examples:
notice(0.3,8)
notice2d("circle(275,275,50)"
Models in Sherpa

- **Parameterized models:** \( f(x_i, p_k) \)
  - absorption - \( N_H \)
  - photon index of a power law function - \( \Gamma \)
  - blackbody temperature \( kT \)

- **Library of models**

- **User Models can be added**

- **Model language to build compound model expressions**

```python
sherpa> list_models()
['absorptionedge',
 'absorptiongaussian',
 'atten',
 'bbody',
 'bbodyfreq',
 'beta1d',
 'beta2d',
 'blackbody',
 ........
```
Building Models: Expressions

- Standard operations: + - * :
- Linking parameters: link()
- Convolution:
  - responses, arf & rmf files via standard I/O
  - PSF - an image file or a Sherpa model
  - load_conv() - a generic kernel from a file or defined by a Sherpa model
Building Models: Examples

• Building composite models:
  • models in the library: e.g. `powlaw1d`, `atten`
  • give a **name** for a model component in the expression:
    
    \[
    \text{set\_source}(1, \text{'atten.abs1*atten.abs2*powlaw1d.p1'})
    \]
    
    \[
    \text{set\_source}(2, \text{'abs1*abs2*powlaw1d.p2'})
    \]

• Building a model expression with **convolved and unconvolved** components:
  
  \[
  \text{set\_full\_model}(1, \text{'psf(gauss2d.g2)+const2d.c1'})
  \]
Building Models: Examples

• Source and Background models:

```
set_source(2, 'xsphabs.abs1*(powlaw1d.pl+gauss1d.g1)')
set_bkg_model(2, 'const1d.mybkg')
```
User Models: Python Function

- Adding a user model defined as a Python function is *Shockingly Simple!*

```python
def myline(pars, x):
    return pars[0] * x + pars[1]

load_user_model(myline, "myl")
add_user_pars("myl", ["m","b"])
set_source(myl)

myl.m=30
myl.b=20
```
User Models: Tables

- **load_table_model()**

  - The file may be 1D data from a FITS table, ASCII table data, or 2D data from a FITS image file.
    - modelname - the name for the table model
    - filename - the name of the file which contains the data
    - ncols (table input) - number of columns to read;
    - colkeys (table input) - list of column names;
    - dstype (table input) - dataset type: *Data1D, Data1DInt, Data2D, Data2DInt*;
    - coord (image input) - the coordinate system: *logical, image, physical, world, or wcs*;
    - method - interpolation method in sherpa.utils:
      - *linear_interp, nearest_interp, neville, neville2d*;

\[\texttt{sherpa}\>\texttt{load_table_model("emap", "expmap.fits")}\]
User Models: XSPEC Table Model

load_xstable_model("pul", "coplrefl.fits")
set_source(xsphabs.galabs * pul)
pul.xi = 1.5
pul.e_cut = 25

XSPEC table model coplrefl.fits is input as pul and used in the model expression.

show_source()
Model: 1
(xsphabs.galabs * xstablemodel.pul)

<table>
<thead>
<tr>
<th>Param</th>
<th>Type</th>
<th>Value</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>galabs.nH</td>
<td>thawed</td>
<td>1</td>
<td>0</td>
<td>100000</td>
<td>10^22 atoms / cm^2</td>
</tr>
<tr>
<td>pul.xi</td>
<td>thawed</td>
<td>1.5</td>
<td>1.477</td>
<td>3.977</td>
<td></td>
</tr>
<tr>
<td>pul.gamma</td>
<td>thawed</td>
<td>1</td>
<td>0.5</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>pul.e_cut</td>
<td>thawed</td>
<td>25</td>
<td>5</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>pul.e_fold</td>
<td>thawed</td>
<td>15</td>
<td>5</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>pul.redshift</td>
<td>frozen</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>pul.norm</td>
<td>thawed</td>
<td>1</td>
<td>0</td>
<td>1e+24</td>
<td></td>
</tr>
</tbody>
</table>
Fit Statistics in Sherpa

Fit statistics - math operation on data and model arrays

In [19]: list_stats()
Out[19]:
['cash',
'chi2',
'chi2constvar',
'chi2datavar',
'chi2gehrels',
'chi2modvar',
'chi2xspecvar',
'cstat',
'leastsq',
'userstat',
'wstat']
In [20]: set_stat('cash')

chi2 statistics as defined by different weights
and Poisson likelihood - cash/cstat/wstat
Fit Statistics in Sherpa

In [19]: list_stats()
Out[19]:
['cash',
 'chi2',
 'chi2constvar',
 'chi2datavar',
 'chi2gehrels',
 'chi2modvar',
 'chi2xspecvar',
 'cstat',
 'leastsq',
 'userstat',
 'wstat']
In [20]: set_stat('cash')

“Handbook of X-ray Astronomy “
(2011), Arnaud, Smith, Siemiginowska
Fitting: Search in the Parameter Space

**sherpa-28> fit()**

```markdown
Dataset       = 1
Method        = levmar
Statistic     = chi2datavar
Initial fit statistic = 644.136
Final fit statistic   = 632.106 at function evaluation 13
Data points   = 460
Degrees of freedom = 457
Probability [Q-value] = 9.71144e-08
Reduced statistic     = 1.38316
Change in statistic   = 12.0305
  zabs1.nh    0.0960949
  p1.gamma    1.29086
  p1.ampl     0.000707365
```

**sherpa-29> print get_fit_results()**

```markdown
datasets   = (1,)
methodname = levmar
statname   = chi2datavar
succeeded  = True
parnames   = ('zabs1.nh', 'p1.gamma', 'p1.ampl')
parvals    = (0.0960948525609, 1.29085977295, 0.000707365006941)
covarerr   = None
statval    = 632.10587995
istatval   = 644.136341045
dstatval   = 12.0304610958
numpoints  = 460
dof        = 457
qval       = 9.71144259004e-08
rstat      = 1.38316385109
message    = both actual and predicted relative reductions in the sum of squares are at most
ftol=1.19209e-07
nfev       = 13
```
Fitting: Sherpa Optimization Methods

- **Optimization** - a minimization of a function:

  "A general function $f(x,p)$ may have **many isolated local minima**, non-isolated minimum hypersurfaces, or even more complicated topologies. No finite minimization routine can guarantee to locate the unique, global, minimum of $f(x,p)$ without being fed intimate knowledge about the function by the user."

- Therefore:
  1. Never accept the result using a single optimization run; always test the minimum using a different method.
  2. Check that the result of the minimization does not have parameter values at the edges of the parameter space. If this happens, then the fit must be disregarded since the minimum lies outside the space that has been searched, or the minimization missed the minimum.
  3. Get a feel for the range of values of the fit statistic, and the stability of the solution, by starting the minimization from several different parameter values.
  4. Always check that the minimum "looks right" using a plotting tool.
Fitting: Optimization Methods in Sherpa

- **“Single - shot” routines: Simplex and Levenberg-Marquardt**
  start from a set of parameters, and then improve in a continuous fashion:
  - Very Quick
  - Depend critically on the initial parameter values
  - Investigate a local behaviour of the statistics near the initial parameters, and then make another guess at the best direction and distance to move to find a better minimum.
  - Continue until all directions result in increase of the statistics or a number of steps has been reached

- **“Scatter-shot” routines: moncar (differential evolution)**
  search over the entire permitted parameter space for a better minima than near the starting initial set of parameters.

- **Bayesian sampling methods: Markov-Chain Monte Carlo**
Optimization Methods: Comparison

Example: Spectral Fit with 3 methods

Data: high S/N simulated ACIS-S spectrum of the two temperature plasma

Model: photoelectric absorption plus two MEKAL components (correlated!)

Start fit from the same initial parameters
Figures and Table compares the efficiency and final results

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of Iterations</th>
<th>Final Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Levmar</td>
<td>31</td>
<td>1.55e5</td>
</tr>
<tr>
<td>Neldermead</td>
<td>1494</td>
<td>0.0542</td>
</tr>
<tr>
<td>Moncar</td>
<td>13045</td>
<td>0.0542</td>
</tr>
</tbody>
</table>

Figures and Table compares the efficiency and final results
Optimization Methods: Probing Parameter Space

Statistics vs. Temperature

Temperature

2D slice of Parameter Space probed by each method

levmar

simplex

moncar

minimum

minimum
Sherpa, MCMC and Bayesian Analysis

MCMC samplers in Sherpa:

- Metropolis and Metropolis-Hastings algorithms
- Support for the Bayesian analysis with priors.

- Explores parameter space and summarizes the full posterior or profile posterior distributions.

- Computed parameter uncertainties can include systematic or calibration errors.

- Simulates replicate data from the posterior predictive distributions.
Sherpa, MCMC and Bayesian Analysis

MCMC samplers:
- Metropolis and Metropolis-Hastings algorithms
- Support for Bayesian analysis with priors.

- Explores parameter space and summarizes the full posterior or profile posterior distributions.
- Computed parameter uncertainties can include systematic or calibration errors.
- Simulates replicate data from the posterior predictive distributions.
Visualization of the MCMC Results

Trace of a parameter during MCMC run

Parameter

Iteration

Probability Density

Cumulative Density

Parameter 1

Parameter 2
Final Analysis Steps

• How well are the model parameters constrained by the data?
• Is this a correct model?
• Is this the only model?
• Do we have definite results?
• What have we learned, discovered?
• How our source compares to the other sources?
• Do we need to obtain a new observation?
Confidence Limits

Essential issue = after the bets-fit parameters are found estimate the confidence limits for them. The region of confidence is given by (Avni 1976):

\[ \chi^2_\alpha = \chi^2_{\text{min}} + \Delta(\nu, \alpha) \]

\( \nu \) - degrees of freedom
\( \alpha \) - level
\( \chi^2_{\text{min}} \) - minimum

\( \Delta \) depends only on the number of parameters involved not on goodness of fit

<table>
<thead>
<tr>
<th>( \alpha ) (%)</th>
<th>( q ) (No. of Interesting Parameters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>68.00</td>
<td>1.00</td>
</tr>
<tr>
<td>90.00</td>
<td>2.71</td>
</tr>
<tr>
<td>99.00</td>
<td>6.63</td>
</tr>
</tbody>
</table>

TABLE 1
Constants for Calculating Confidence Regions
Calculating Confidence Limits means Exploring the Parameter Space - Statistical Surface

Example of a “well-behaved” statistical surface in parameter space, viewed as a multi-dimensional paraboloid ($\chi^2$, top), and as a multi-dimensional Gaussian ($\exp(-\chi^2/2) \approx L$, bottom).
Aneta Siemiginowska

Confidence Intervals

```
sherpa-40> covariance()
Dataset   = 1
Confidence Method = covariance
Fitting Method    = neldermead
Statistic          = chi2datavar

covariance 1-sigma (68.2689%) bounds:

<table>
<thead>
<tr>
<th>Param</th>
<th>Best-Fit</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs1.nH</td>
<td>1.1015</td>
<td>-0.00153623</td>
<td>0.00153623</td>
</tr>
<tr>
<td>mek1.kT</td>
<td>0.841024</td>
<td>-0.00115618</td>
<td>0.00115618</td>
</tr>
<tr>
<td>mek1.norm</td>
<td>0.699764</td>
<td>-0.00395776</td>
<td>0.00395776</td>
</tr>
<tr>
<td>mek2.kT</td>
<td>2.35844</td>
<td>-0.00371253</td>
<td>0.00371253</td>
</tr>
<tr>
<td>mek2.norm</td>
<td>1.03725</td>
<td>-0.00172503</td>
<td>0.00172503</td>
</tr>
</tbody>
</table>

```

```
sherpa-42> conf()

mek1.kT lower bound:   -0.00113811
mek1.kT upper bound:   0.0011439
mek2.kT lower bound:   -0.00365452
mek2.kT upper bound:   0.00364805
mek1.norm lower bound: -0.00377224
mek1.norm upper bound: 0.00376011
mek2.norm lower bound: -0.00164417
mek2.norm upper bound: 0.00164816
abs1.nH lower bound:   -0.00147622
abs1.nH upper bound:   0.00147268

```

```
sherpa-42> print get_conf_results()

----------> print(get_conf_results())

datasets   = (1,)
methodname = confidence
fname      = neldermead
statname   = chi2datavar
sigma      = 1
percent    = 68.2689492137

covariance 1-sigma (68.2689%) bounds:

<table>
<thead>
<tr>
<th>Param</th>
<th>Best-Fit</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs1.nH</td>
<td>1.1015</td>
<td>-0.00147622</td>
<td>0.00147268</td>
</tr>
<tr>
<td>mek1.kT</td>
<td>0.841024</td>
<td>-0.00113811</td>
<td>0.0011439</td>
</tr>
<tr>
<td>mek1.norm</td>
<td>0.699764</td>
<td>-0.00377224</td>
<td>0.00376011</td>
</tr>
<tr>
<td>mek2.kT</td>
<td>2.35844</td>
<td>-0.00365452</td>
<td>0.00364805</td>
</tr>
<tr>
<td>mek2.norm</td>
<td>1.03725</td>
<td>-0.00164417</td>
<td>0.00164816</td>
</tr>
</tbody>
</table>

```

```
```
Not well-behaved Surface

Non-Gaussian Shape
Confidence Regions

sherpa-61> reg_proj(p1.gamma,zabs1.nh,nloop=[20,20])
sherpa-62> print get_reg_proj()
min     = [ 1.2516146   0.07861824]
max     = [ 1.33010494  0.11357147]
nloop   = [20, 20]
fac     = 4
delv   = None
log    = [False False]
sigma  = (1, 2, 3)
parval0 = 1.29085977295
parval1 = 0.0960948525609
levels = [ 634.40162888  638.28595426  643.93503803]
MCMC Results: Probability Distributions

sherpa> plot_scatter(p1,p2)

sherpa> plot_pdf(p1)

sherpa> plot_cdf(p1)
Distributions of Flux and Parameters

Functions: `sample_energy_flux, sample_flux`

Monte Carlo Simulations of parameters assuming Gaussian distributions for all the parameters Characterized by the covariance matrix, includes correlations between parameters.

```python
sherpa-19> flux100=sample_energy_flux(0.5,2.,num=100)
sherpa-20> print flux100
---------> print(flux100)
[[ 2.88873592e-10   1.10331438e+00   8.40356670e-01   6.97503733e-01
  2.35411369e+00   1.03580042e+00]
 [ 2.90279483e-10   1.10243140e+00   8.41174148e-01   7.01009661e-01
 sherpa-26> plot_energy_flux(0.5,2,num=1000)
```

* Characterize distributions: plot PDF and CDF and obtain Quatiles of 68% and 95%.

```python
sherpa-30> fluxes=numpy.sort(flux1000[:,0])
sherpa-31> a95=fluxes(0.95*len(flux1000[:,0])-1)
sherpa-32> a68=fluxes(0.68*len(flux1000[:,0])-1)
```
**Sherpa - Summary**

- Modeling and fitting application for Python.
- User Interface and high level functions written in Python.
- Modeling 1D/2D (N-D) data: arrays, spectra, images.
- Powerful language for building complex expressions.
- Provides a variety of statistics and optimization methods (including Bayesian analysis).
- Support for wcs, responses, psf, convolution.
- Extensible to include user models, statistics and optimization methods.
- Included in several software packages.
- Source code on GitHub [https://github.com/sherpa/sherpa](https://github.com/sherpa/sherpa)
- Open development with continuous integration via Travis
Core Team:
Omar Laurino, Doug Burke, Warren McLaughlin, Dan Nguyen, Aneta Siemiginowska

Code contributions:
Tom Aldcroft, Jamie Budynkiewicz, Christoph Deil, Brigitta Sipocz

Sherpa is a modeling and fitting application for Python  [http://cxc.cfa.harvard.edu/contrib/sh...](http://cxc.cfa.harvard.edu/contrib/sh...)

1,844 commits  15 branches  20 releases  10 contributors

Branch: master  New pull request

- docs: Fix typo on quick.rst  12 days ago
- extern: add xspec and ds9 for macos  6 months ago
- helpers: remove default dependencies values from xspec_config  3 months ago
- notebooks: Docs: update notebook  2 months ago
- scripts: implement first 2D API test, and implement what’s missing  4 months ago
- sherpa-test-data @ 7aeb726: Can we create an XSPEC multiplicative model  2 months ago
- sherpa: Merge #527 (DougBurke) - Switch pileup test from unittest to pytest s...  12 days ago
- travis: Docs: Note graphviz as a requirement for building the documentation  2 months ago
Open Development

Code contribution

Travis
- continuous integration
Learn more on Sherpa Web Pages

http://cxc.harvard.edu/sherpa

Freeman, P., Doe, S., & Siemiginowska, A. 2001, SPIE 4477, 76
Refsdal et al. 2009 - Sherpa: 1D/2D modeling and fitting in Python in Proceedings of the 8th Python in Science conference (SciPy 2009), G Varoquaux, S van der Walt, J Millman (Eds.), pp. 31-37
Welcome to the Sherpa documentation. Sherpa is a Python package for modeling and fitting data. It was originally developed by the Chandra X-ray Center for use in analysing X-ray data (both spectral and imaging) from the Chandra X-ray telescope, but it is designed to be a general-purpose package, which can be enhanced with domain-specific tasks (such as X-ray Astronomy). Sherpa contains an expressive and powerful modeling language, coupled with a range of statistics and robust optimisers.

Sherpa is released under the GNU General Public License v3.0, and is compatible with Python versions 2.7, 3.5, and 3.6. Information on recent releases and citation information for Sherpa is available using the Digital Object Identifier (DOI) 10.5281/zenodo.593753.

**Introduction**

- Installation
  - Requirements
  - Releases and version numbers
  - Installing a pre-compiled version of Sherpa
  - Building from source
  - Testing the Sherpa installation
- A quick guide to modeling and fitting in Sherpa
  - Getting started
  - Fitting a one-dimensional data set
  - Including errors
  - Fitting two-dimensional data
Using Sherpa in Astronomy Software

• IRIS - GUI for data exploration and SED fitting  
  http://www.usvao.org/index.html%3Fpage_id=357.html

• BAX - Bayesian X-ray Analysis  
  https://github.com/JohannesBuchner/BXA

• XMM-Newton Source Catalog:  
  http://xmm-catalog.irap.omp.eu/docs/spectral-fitting
  • web interface to spectral fitting of the sources in 3XMM-DR6 catalog

• Astropy\textsuperscript{a} Affiliated packages:  
  http://www.astropy.org/affiliated/index.html
  • GammaPy  
    https://gammapy.readthedocs.io/en/latest/
  • Naima  

• Saba - Sherpa-Astropy Bridge  
  https://saba.readthedocs.io/en/latest/
  • Google funded a summer student (through GSOC program) to develop the code and documentation.
  • pending application for Astropy affiliated package.

\textsuperscript{a)} From astropy.org web page:
Astropy is a community-driven package intended to contain much of the core functionality and some common tools needed for performing astronomy and astrophysics with Python.
SABA: Sherpa - Astropy Bridge

- **astropy.modeling**
  - uses *scipy.optimize*, weighted least square, does not calculate uncertainties
  - bridge between the model definition language in *astropy.modeling* and the powerful fitting capabilities of Sherpa.
- **Sherpa**
  - has a selection of robust optimization algorithms coupled with configurable fit statistics.
  - can estimate parameter confidence intervals, with methods that allow for coupled non-gaussian errors.
  - has an MCMC sampler that can be used to explore the posterior probability distribution.
Spectral (SED) Fitting with Composite Templates

Fig. 6. Rest-frame SED of class B HLIRG and their best-fit models. Symbols as in Fig. 5. The long-dashed lines (blue in the colour version) are the best fits obtained using composite templates (see Sects. 4.1 and 5.2).
Fitting Spatial Profiles of the HST observations of Mrk 231

Leighly et al. (2016)
Surface Brightness Profiles (with & without PSF)

Chandra and XMM

O’Sullivan et al. (2011)

HST Images

Richings, Utley & Kording (2011)

Chandra

Wang et al. (2010)

Radio loudness and surface brightness profile

Figure 1. Galaxy images (top row) and radial brightness profiles (bottom row) for a confident Sérsic fit (NGC 7342, left), Core fit (NGC 3379, center) and Double-Sérsic fit (NGC 7217, right).
Image Analysis

Optical-X-ray offsets
Searches for Binary BH
and GW Recoils

Comerford et al. (2015)

Barrows et al. (2016)
Identifying Substructures in X-ray Clusters

Sanders & Fabian (2012)

Randall et al. (2015)
Summary

• Sherpa is a Python package.
• It provides models, fit statistics and optimization methods for variety of problems in astronomy.
• It is flexible and extensible as it accepts new models, statistics or optimization.
• Sherpa can also be included in a modeling software in Python.
Composite Models in BXA Bayesian X-ray Analysis

Figure 5: Observed (convolved) spectrum of object 179, binned for plotting to 10 counts per bin. Shown are analyses using various models and their individual components: powerlaw (upper left), wabs (upper right), torus+scattering (lower left) and wabs+pexmon+scattering (lower right). The posterior of the parameters are used to compute the median and 10%-quantiles of each model component.

Buchner et al. 2014
Spatial Fitting of the TeV emission in H.E.S.S. observations

**Fig. 3.** Profile of the VHE emission along the line between the peak of the point-like emission and the peak of the diffuse emission, as illustrated in the inset. Fits using a single and a double Gaussian function are shown in dashed and solid lines respectively. The positions of XMMU J101855.4–58564 and PSR J1016–5857 are marked with dashed and dotted vertical lines and red and yellow stars in the inset, in which the significance image obtained using an oversampling radius of 0.1° is shown.

**Fig. 4.** VHE photon spectrum of HESS J1018–589 for a point-like source at position A (in blue dots and dashed blue line) and derived from a region of size 0.30° comprising the point-like and diffuse emission (in black dots and solid black line). The residuals to the fit are shown in the bottom panel.

Abramowski et al. (2012)
There are also Sherpa Jupyter Notebooks available on Sherpa GitHub page:

https://github.com/sherpa/sherpa/wiki

Example of Image Fitting:

http://nbviewer.jupyter.org/github/anetasie/SherpaNotebooks/blob/master/ImageFitting.ipynb