SHERPA

CIAO's Modeling and Fitting Application

Aneta Siemiginowska

Modeling and Fitting Software

- **XSPEC** - analysis of 1D X-ray data (imaging + grating)
- **ISIS** and **Pint of Ale**- primarily for analysis of high-resolution (ie grating) X-ray data
- **Sherpa** - generalised multi-dimensional fitting package
- All programs use the technique of **forward fitting**:
  - a model is evaluated, compared to the actual data, and then the parameters are changed to improve the match. This is repeated until convergence occurs.

What can you do in Sherpa?

- Standard PHA based analysis.
- Model data in many spectral bands simultaneously, e.g., optical/ X-rays.
- Access ATOMDB and GUIDE/ISIS for grating data analysis.
- Fit radial profiles.
- Simulate 1D data.
- Model 2D image data, e.g., fit surface brightness of the extended source.
- Get normalization of your PSF, while fitting the data with 1D/2D PSF.
- Use the PSF as a convolution kernel in the 2D image analysis(FFT or sliding cell).
- Convolution using the TCD library kernel.
- Use of exposure maps in the image analysis.
- Joint-mode data: spatial-spectral, spatial-timing
- Use scripts based on Sherpa only commands.
- Use S-lang on command line and in S-lang based scripts.
- Use your own models with User Models and S-lang user models.

Standard PHA based analysis:

- Source data:
  - can be modeled in energy/wavelength space.
  - multiple data sets can be modeled with the same or different models in one Sherpa session.
  - data can be filtered on the command line, or from filter file.

- Instrument responses (RMF/ARF):
  - are entered independently from the source data.
  - one set of instrument responses can be read once and applied to multiple data sets.
  - several instrument responses used in analysis of one source model or multiple data sets.
  - multiple response files can be used in one source model expression.

- Background files:
  - are entered independently from the source data.
  - multiple background files can be used for one data set,e.g. grating analysis
  - the same background can be applied to multiple data sets.
  - background can be modeled independently of the source data, and have its separate instrument responses.
  - background can be modeled simultaneously with the source data.
  - background can be subtracted from the source data (subtract/unsubtract).

A set of Models

Combined to create a Source Expression

Model Parameters

Instrument Stack (ARF&RMF for PHA data Exposure Map and/or PSF for an image, NONE for radial profile)

What happens in Sherpa?

Add on the **Background Stack**

Input Data (with Errors)

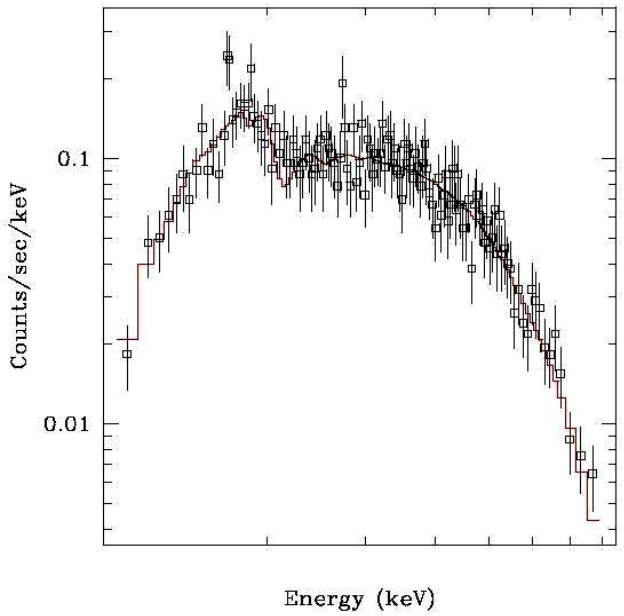
Predicted Data

Compare

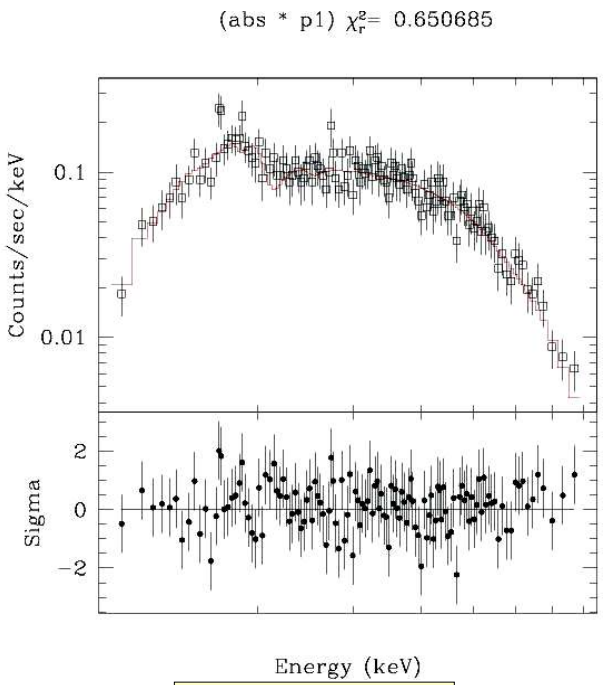
Fit Statistics

Optimization

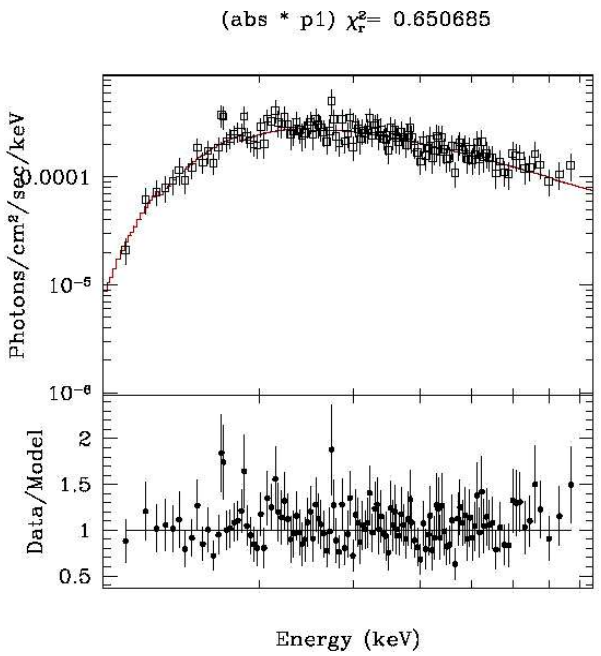
Displaying the Results



lp fit



lp 2 fit delchi



lp 2 ufit ratio

Main SHERPA Components

- Data Input/Output.
- Visualization through ChIPS and ds9
- Model library and model language.
- Statistics and Error Analysis.
- Optimization Methods.

Data Input/Output

- General use of data type and dimensionality.
- Supported types of files: ASCII, FITS binary tables and Images,PHA types I \& II, IRAF IMH and QPOE files
- Sherpa:
  - groups the data if appropriate;
  - treats integer, float or double precision data;
  - supports data of arbitrary dimensionality
- I/O interface through Data Model and Varmm
- Filtering while reading the data.
- Input data on the command line in two ways.

```
sherpa> data "image.fits(150:300,160:310)"
sherpa> show
Current Data Files:
Data 1: image.fits(150:300,160:310) fits.
Total Size: 22801 bins (or pixels)
Dimensions: 2
Size: 151 x 151
Total counts (or values): 20711 cts
```

or

```
sherpa> mydata=readfile("image.fits(150:300,160:310)")
        sherpa> print(mydata)
_filename       = image.fits
_path           = /data/
_filter         = (150:300,160:310)
_filetype       = 11
_header         = String_Type(268)
_transform      = TAN
_naxes          = 2
pixels          = Float_Type(151,151)
min             = Double_Type(2)
max             = Double_Type(2)
step            = Double_Type(2)
crval           = Double_Type(2)
crpix           = Double_Type(2)
crdelt          = Double_Type(2)
sherpa> print(mydata.crval(0))
278.386
sherpa> print(mydata.crval(1))
-10.5899
```

MODELS

- Three main type of models:
  - Source
  - Background
  - Instrument
- Model library consists of several models (plus XSPEC v.11) which can be used to define a source or background model
- There are different types of instrument models to support both 1D and 2D analysis.
- Instrument models are **convolved** with Source and Background models before the model predicted data is compared with the observed data.
- Instrument and Background models are **NOT** required. Source models have to be defined for fitting.

Instrument Models

FARF1D | FARF	**1-D file-based ARF**
FARF2D | FEXPMAP | FEXPMAP2D	**2-D file-based Exposure Map**
FPSF1D	**1-D file-based PSF**
FPSF2D | FPSF | PSFFROMFILE	**2-D file-based PSF**
FRMF1D | FRMF	**1-D file-based RMF**
RSP1D | RSP	**1-D instrument model (ARF/RMF)**
RSP2D	**2-D instrument model (ExpMap/PSF)**
TPSF1D	**1-D TCD-model-based PSF**
TPSF2D | TPSF | PSFFROMTCD	**2-D TCD-model-based PSF**

```
r sp1d[myfile]
    Param   Type      Value
 -----   ----      ----
  1    rmf  string:    "file.rmf"
  2    arf  string:    "file.arf"
tpsf2d[tpsf2d]
    Param   Type      Value        Min        Max
  -----   ----   -----    ---
 1  xsize     frozen      4          1       1024
 2  ysize     frozen      4          1       1024
 3 nsigma     frozen      2      1e-02       100
 4 funcTyp   frozen      1          0         7
 5    fft     frozen      1          0         1
 The Function Type is:  Gaussian.

fpsf2d[psf0]
    Param   Type      Value        Min        Max
  -----   ----   -----    ---       ---
 1   file     string:  "psf_image.fits"
 2  xsize    frozen         32          1       1024
 3  ysize    frozen         32          1       1024
 4   xoff    frozen          0       -512       512
 5   yoff    frozen          0       -512       512
 6    fft    frozen          1          0         1
```

Instrument Models Expressions

```
sherpa> farf1d[a](arf.fits)
sherpa> frmf1d[r](rmf.fits)
sherpa> instrument = a*r
```

Here, the photon spectrum y is multiplied by the ARF, then folded through the RMF. This instrument stack is equivalent to

```
sherpa> instrument = rsp[a](rmf.fits,arf.fits)
```

Sets of instrument models separated by the + operator each fold the same evaluated photon spectrum y, with the resulting group of counts spectra being summed.

```
sherpa> farf1d[a1](arf_order1.fits)
sherpa> farf1d[a2](arf_order2.fits)
sherpa> frmf1d[r1](rmf_order1.fits)
sherpa> frmf1d[r2](rmf_order2.fits)
sherpa> instrument = a1*r1 + a2*r2
```

Model Language

- All predefined in model library models can be used in model expression to build a source or background model
- Each library model can be given a unique name within Sherpa session.

```
sherpa> gauss1d(g1)
sherpa> source = ATTEN(att1)*BPL(b1)
att1.hcol parameter value (1e+20)
att1.heiRatio parameter value (0.1)
att1.heiiRatio parameter value (0.01)
b1.gamma1 parameter value (0)
b1.gamma2 parameter value (0)
b1.eb parameter value (100)
b1.ref parameter value (1)
b1.ampl parameter value (1)
```

- Model Parameters can be linked to other model parameters, arithmetic expression or other models.

```
sherpa> source = POLY(con)+gauss1d(g1)+gauss1d(g2)
sherpa> g1.ampl => 0.4*g2.ampl
```

or

```
sherpa> func = const1d(red)
sherpa> g1.pos => 0.568*func
```

? An argument of a model (e.g. energy) is defined as an expression in Nested Models.

Parameter Expression:

```
sherpa> Temperature = POLY
sherpa> BB.kT => Temperature
sherpa> show source
BB
bbody(BB)   (integrate: on)
        Param   Type        Value          Min          Max
        -----   ----        -----          ---          ---
          1       kT     link      varying      expression: Temperature
          2     ampl   thawed          0.3        1e-20        3.4028e+38
```

Argument Expression:

```
sherpa> xenergy = SHLOG(mod)
sherpa> source = BB{xenergy}
sherpa> show source
BB{xenergy}
bbody(BB)   (integrate: on)
        Param   Type        Value        Min          Max
        -----   ----        -----        ---          ---
          1       kT   thawed          0.3       0.1000         1000
          2     ampl   thawed        0.001        1e-20    3.4028e+38
shloge(mod)   (integrate: off)
        Param   Type        Value          Min          Max
        -----   ----        -----          ---          ---
          1   offset   frozen          0-3.4028e+38        3.4028e+38
          2    coeff   frozen          1-3.4028e+38        3.4028e+38
          3     ampl   frozen          1            0        3.4028e+38
```

- For Joint-Mode analysis one can apply models on each axis:

```
sherpa> DATA image.fits FITSIMAGE
sherpa> LORENTZ(SpatialAxis0)(98:5:200, 70:50:90,1:1:200)
sherpa> POWLAWID(SpecAxis1)
sherpa> SRC = SpatialAxis0{x1}*SpecAxis1{x2}
sherpa> show source
(SpatialAxis0{ 0 } * SpecAxis1{ 1 })
lorentzld(SpatialAxis0)  (integrate: on)
     Param   Type          Value          Min          Max
     -----   ----          -----          ---          ---
   1  fwhm   thawed             98            5          200
   2   pos   thawed             70           50           90
   3  ampl   thawed              1            1          200
powlawld(SpecAxis1)  (integrate: on)
     Param   Type          Value          Min          Max
     -----   ----          -----          ---          ---
   1 gamma   thawed            1.5          -10           10
   2   ref   frozen              1-3.4028e+38 3.4028e+38
   3  ampl   thawed              1        1e-20 3.4028e+38
```

Fit Statistics in Sherpa:

Sherpa has a large array of statistics appropriate for analyzing Poisson-distributed (*i.e.* counts) data.

- Statistics based on χ^2 :
  - CHI GEHRELS
  - CHI DVAR
  - CHI MVAR
  - CHI PARENT
  - CHI PRIMINI
- Statistics based on the Poisson likelihood:
  - CASH
  - BAYES

If the data are not Poisson-distributed (*i.e.* fluxes), then alternatives include:

? least-squares fitting: setting all variances to one

? providing errors in an input file.

Optimization in Sherpa

Optimization => minimizing the statistics (^{2} or $\log \mathcal{L}$) by varying the thawed parameters of the model.

- **Find a local minimum:**

  LEVENBERG-MARQUARDT
  POWELL
  SIMPLEX

  → Fast, but not appropriate for finding the global minimum of a complex statistical space when starting from a random point

- **Attempt to find the global minimum:**

  GRID
  GRID-POWELL
  MONTECARLO
  MONTE-LM
  MONTE-POWELL
  SMULATED ANNEALING

  → Computationaly intensive algorithms designed to search comlicated statistical surfaces.

- **Optimize/Reject/Filter:**

  SIGMA-REJECTION outliers are filtered from the data.

Confidence Intervals

- Vary a parameter's value, while holding the values of all the parameters to their best-fit values, until the fit statistic increases by some preset amount from its minimum value (^{2} = 1 for 1).
  - Uncertainty
  - Projection
- Calculate **Covariance** matrix:

1 confidence intervals are given by $\sqrt{C_{i,i}}$

where $C_{j,i} = I^{1}_{i,j}$

and $I_{i,j}$ - the information matrix computed at the best-fit point:

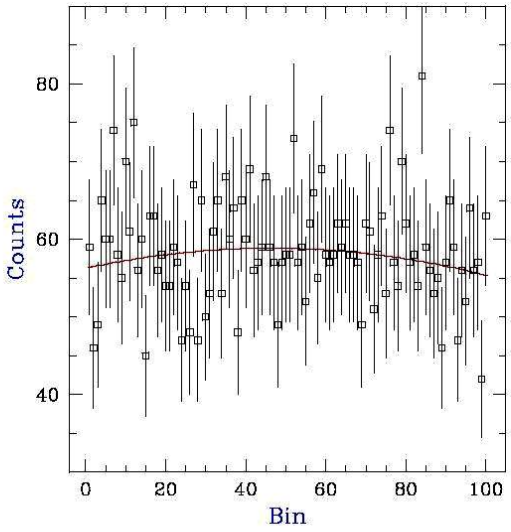
$$I_{i,j} = \frac{\partial^2 \ ^2}{\partial p_i \partial p_j}$$

or any other statistics

Visualize Confidence Levels

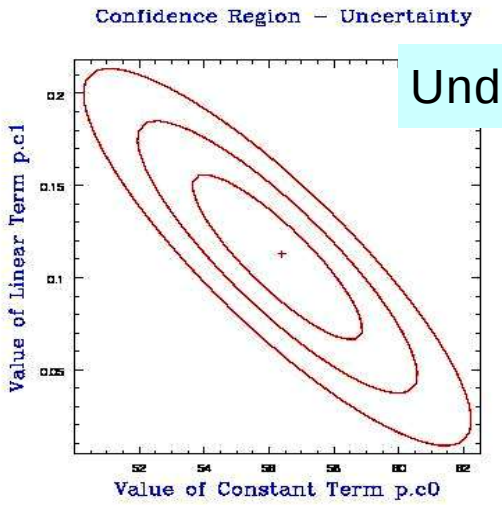
Uncertainty

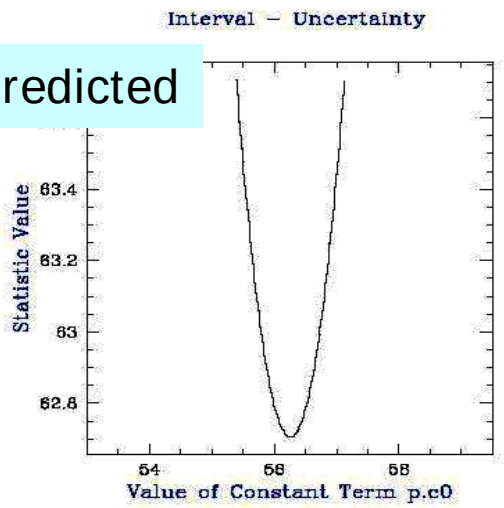
Data and the Best Fit Model

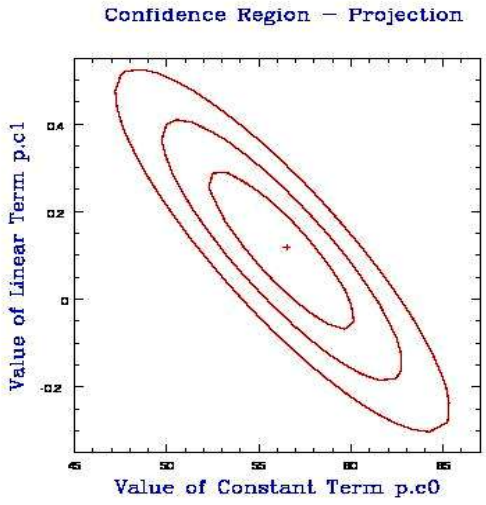


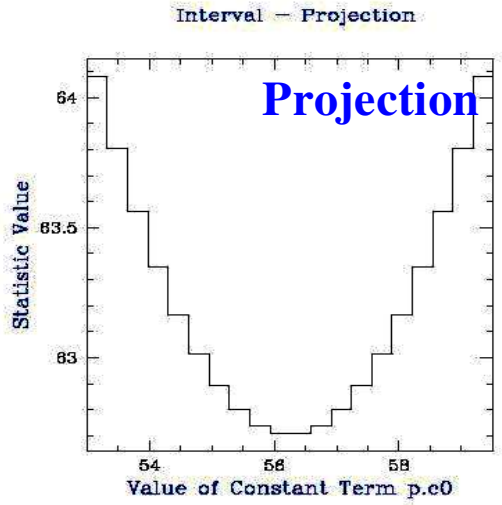
Well behave parameter space!

Underpredicted









Customize Sherpa

- **Sherpa State Object** (e.g. Configuration file) – S-lang variable initialized at the start of the Sherpa session:

```
sherpa> print(sherpa)
plot          = sherpa_Plot_State
dataplot      = sherpa_Plot_State
fitplot       = sherpa_FitPlot_State
resplot       = sherpa_Plot_State
multiplot     = sherpa_Draw_State
output        = sherpa_Output_State
regproj       = sherpa_VisParEst_State
regunc        = sherpa_VisParEst_State
intproj       = sherpa_VisParEst_State
intunc        = sherpa_VisParEst_State
proj          = sherpa_Proj_State
cov           = sherpa_Cov_State
unc           = sherpa_Unc_State
con_levs      = NULL
modeloverride = 0
multiback     = 0
deleteframes  = 1
clobber       = 0
```

Customize Plotting

Customize Confidence Levels

```
sherpa> print(sherpa.regproj)
fast          = 1
expfac        = 3
arange        = 1
min           = Double_Type[2]
max           = Double_Type[2]
log           = Integer_Type[2]
nloop         = Integer_Type[2]
sigma         = Double_Type[3]
```

Customize Sherpa

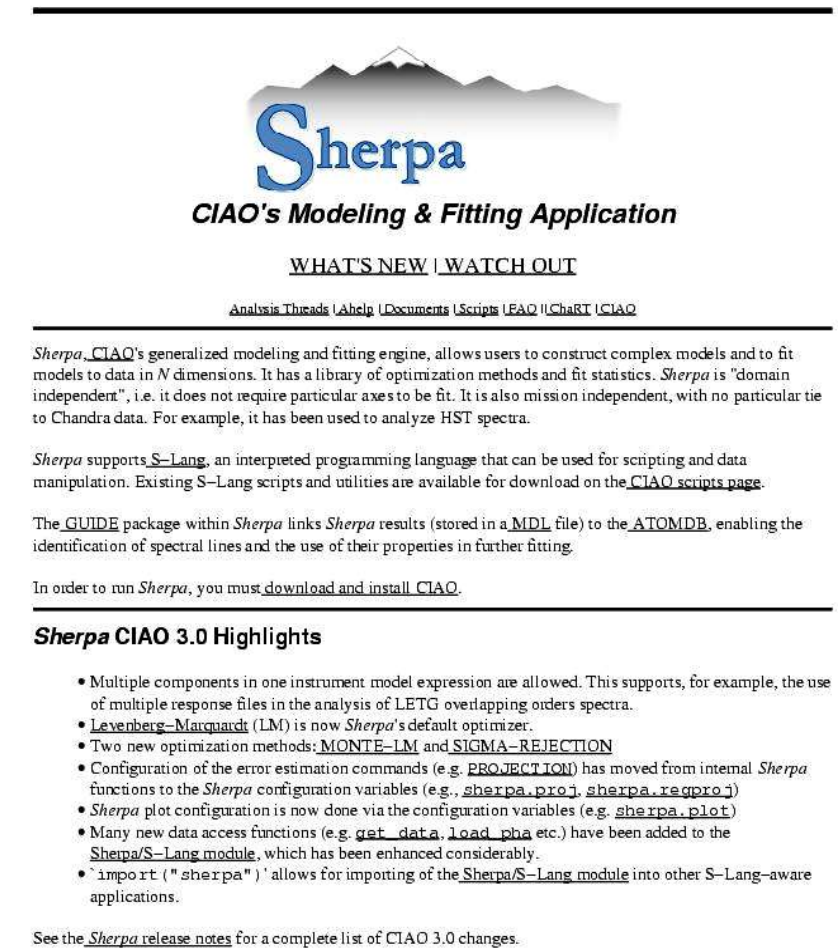
- Sherpa Resource File:
  - a text file with Sherpa/Chips/S-lang commands
- **Access:**
  - Environment variable SHERPARC
  - File **.sherparc** in current directory $PWD
  - File **.sherparc** in HOME directory $HOME
- Example:

```
unix% more .sherparc
# Example Sherpa resource file
message("Starting to process sherparc")
paramprompt off
method simplex
define q () { () = sherpa_eval("quit"); }
message("Finished processing .sherparc")
```

```
unix% sherpa
...
   Abundances set to Anders & Grevesse
Starting to process .sherparc
Model parameter prompting is off
Finished processing .sherparc
sherpa> show method
Optimization Method: Simplex
     Name      Value       Min        Max              Description
     ----      -----       ---        ---              -----------
 1   iters      2000         1      10000    Maximum number of iterations
 2   eps       1e-03      1e-04        100          Absolute accuracy
 3   alpha         1        0.1          2   Algorithm convergence factor
 4    beta       0.5       5e-02         1   Algorithm convergence factor
 5   gamma          2        1.1         20   Algorithm convergence factor
sherpa> q
Goodbye.
```

Learn More on Sherpa Web Page

http://cxc.harvard.edu/sherpa/



Sherpa

CIAO's Modeling & Fitting Application

WHAT'S NEW | WATCH OUT

Analysis Threads | Ahelp | Documents | Scripts | FAQ | ChaRT | CIAO

Sherpa, CIAO's generalized modeling and fitting engine, allows users to construct complex models and to fit models to data in *N* dimensions. It has a library of optimization methods and fit statistics. *Sherpa* is "domain independent", i.e. it does not require particular axes to be fit. It is also mission independent, with no particular tie to Chandra data. For example, it has been used to analyze HST spectra.

Sherpa supports S–Lang, an interpreted programming language that can be used for scripting and data manipulation. Existing S–Lang scripts and utilities are available for download on the CIAO scripts page.

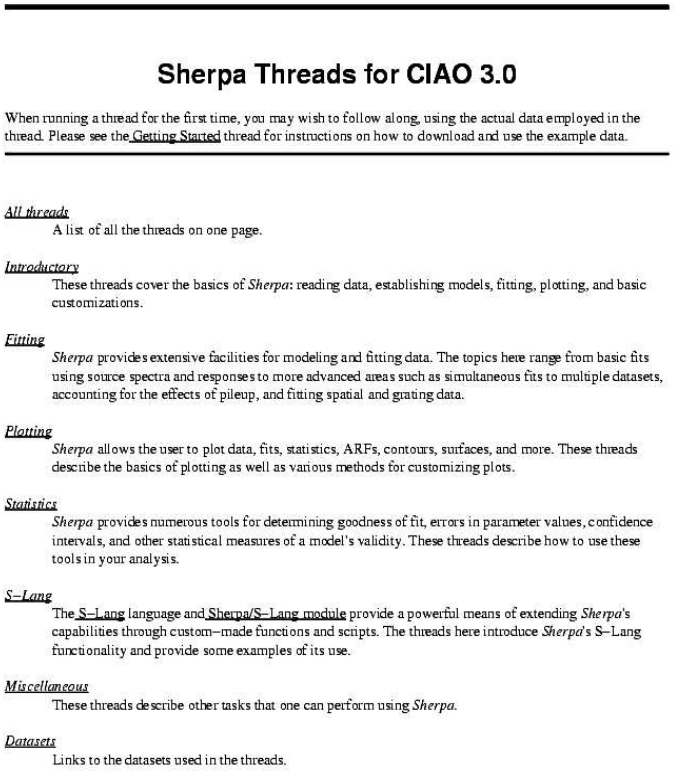
The GUIDE package within *Sherpa* links *Sherpa* results (stored in a MDL file) to the ATOMDB, enabling the identification of spectral lines and the use of their properties in further fitting.

In order to run *Sherpa*, you must download and install CIAO.

Sherpa CIAO 3.0 Highlights

- Multiple components in one instrument model expression are allowed. This supports, for example, the use of multiple response files in the analysis of LETG overlapping orders spectra.
- Levenberg–Marquardt (LM) is now *Sherpa*'s default optimizer.
- Two new optimization methods: MONTE–LM and SIGMA–REJECTION
- Configuration of the error estimation commands (e.g. PROJECTION) has moved from internal *Sherpa* functions to the *Sherpa* configuration variables (e.g., `sherpa.proj`, `sherpa.regproj`)
- *Sherpa* plot configuration is now done via the configuration variables (e.g. `sherpa.plot`)
- Many new data access functions (e.g. `get_data`, `load_pha` etc.) have been added to the Sherpa/S–Lang module, which has been enhanced considerably.
- `` `import("sherpa")' `` allows for importing of the Sherpa/S–Lang module into other S–Lang–aware applications.

See the *Sherpa* release notes for a complete list of CIAO 3.0 changes.



Sherpa Threads for CIAO 3.0

When running a thread for the first time, you may wish to follow along, using the actual data employed in the thread. Please see the Getting Started thread for instructions on how to download and use the example data.

All threads
A list of all the threads on one page.

Introductory
These threads cover the basics of *Sherpa*: reading data, establishing models, fitting, plotting, and basic customizations.

Fitting
Sherpa provides extensive facilities for modeling and fitting data. The topics here range from basic fits using source spectra and responses to more advanced areas such as simultaneous fits to multiple datasets, accounting for the effects of pileup, and fitting spatial and grating data.

Plotting
Sherpa allows the user to plot data, fits, statistics, ARFs, contours, surfaces, and more. These threads describe the basics of plotting as well as various methods for customizing plots.

Statistics
Sherpa provides numerous tools for determining goodness of fit, errors in parameter values, confidence intervals, and other statistical measures of a model's validity. These threads describe how to use these tools in your analysis.

S–Lang
The S–Lang language and Sherpa/S–Lang module provide a powerful means of extending *Sherpa*'s capabilities through custom–made functions and scripts. The threads here introduce *Sherpa*'s S–Lang functionality and provide some examples of its use.

Miscellaneous
These threads describe other tasks that one can perform using *Sherpa*.

Datasets
Links to the datasets used in the threads.