Introduction to Sherpa

Aneta Siemiginowska

Chandra X-ray Center

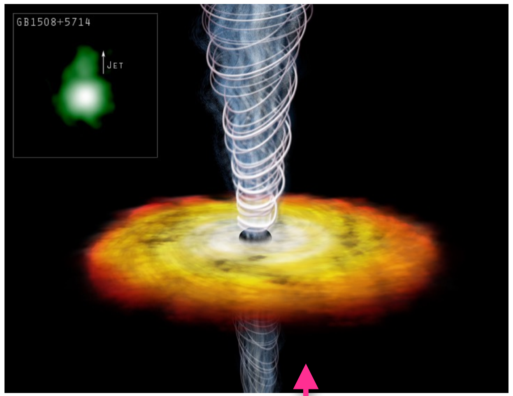
http://cxc.harvard.edu/sherpa

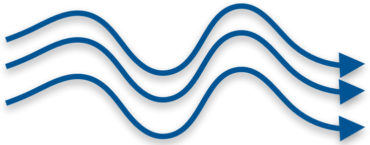
CIAO Workshop at ArAS SfA 5
Egypt and Virtually Everywhere - Oct 2020

Observations and Data Collection

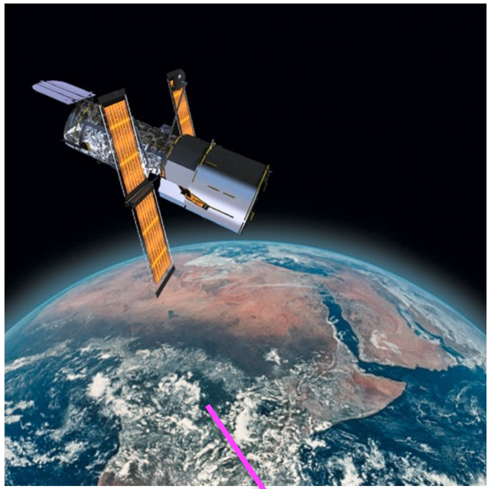
Astrophysical process

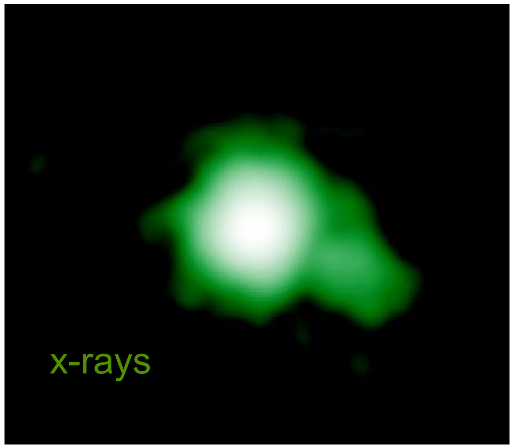
Detector collects photons, adds noise





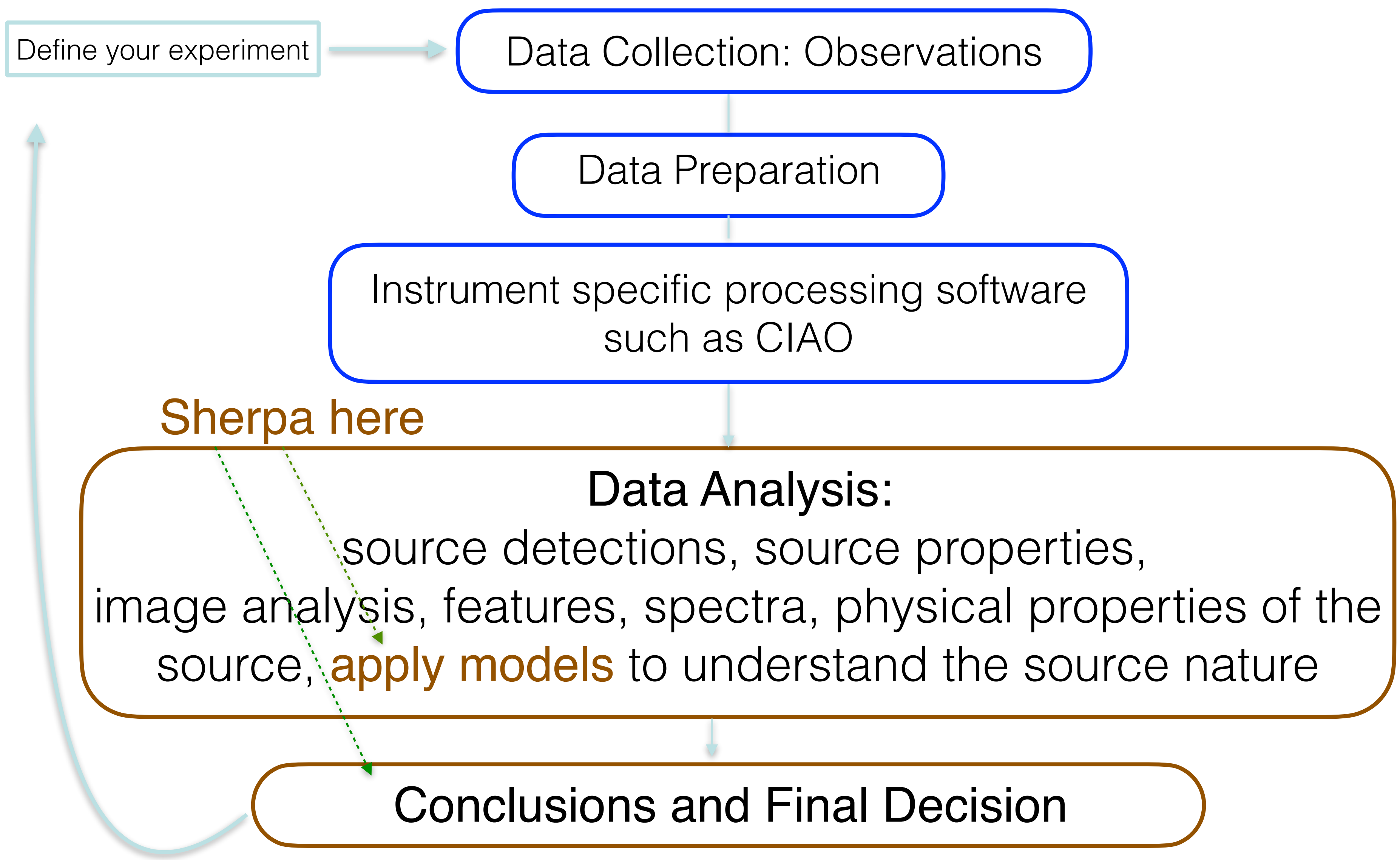
Random number of photons reach the detector





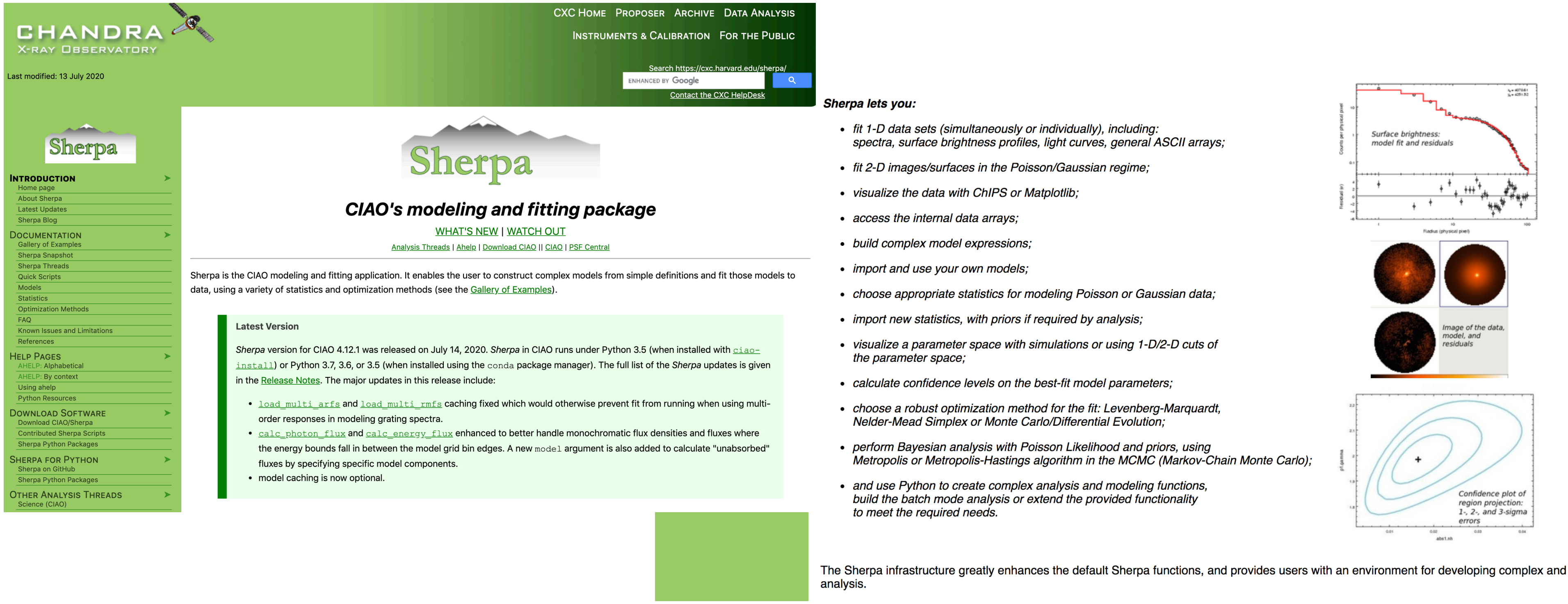
draw conclusion about the astrophysical source

Scientific Experiment


Define your experiment
Data Collection: Observations
Data Preparation
Instrument specific processing software such as CIAO
Sherpa here
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 CIAO Web Pages

http://cxc.harvard.edu/sherpa


CHANDRA
X-RAY OBSERVATORY
CXC Home Proposer Archive Data Analysis
Instruments & Calibration For the Public
Search https://cxc.harvard.edu/sherpa/
ENHANCED BY Google
Contact the CXC HelpDesk
Surface brightness: model fit and residuals
Image of the data, model, and residuals
Confidence plot of region projection: 1-, 2-, and 3-sigma errors

Last modified: 13 July 2020

Introduction
- Home page
- About Sherpa
- Latest Updates
- Sherpa Blog

Documentation
- Gallery of Examples
- Sherpa Snapshot
- Sherpa Threads
- Quick Scripts
- Models
- Statistics
- Optimization Methods
- FAQ
- Known Issues and Limitations
- References

Help Pages
- AHELP: Alphabetical
- AHELP: By context
- Using ahelp
- Python Resources

Download Software
- Download CIAO/Sherpa
- Contributed Sherpa Scripts
- Sherpa Python Packages

Sherpa for Python
- Sherpa on GitHub
- Sherpa Python Packages

Other Analysis Threads
- Science (CIAO)

Sherpa

CIAO's modeling and fitting package

WHAT'S NEW | WATCH OUT

Analysis Threads | Ahelp | Download CIAO || CIAO | PSF Central

Sherpa is the CIAO modeling and fitting application. It enables the user to construct complex models from simple definitions and fit those models to data, using a variety of statistics and optimization methods (see the Gallery of Examples).

Latest Version

Sherpa version for CIAO 4.12.1 was released on July 14, 2020. *Sherpa* in CIAO runs under Python 3.5 (when installed with `ciao-install`) or Python 3.7, 3.6, or 3.5 (when installed using the `conda` package manager). The full list of the *Sherpa* updates is given in the Release Notes. The major updates in this release include:

- `load_multi_arfs` and `load_multi_rmfs` caching fixed which would otherwise prevent fit from running when using multi-order responses in modeling grating spectra.
- `calc_photon_flux` and `calc_energy_flux` enhanced to better handle monochromatic flux densities and fluxes where the energy bounds fall in between the model grid bin edges. A new `model` argument is also added to calculate "unabsorbed" fluxes by specifying specific model components.
- model caching is now optional.

Sherpa lets you:

- *fit 1-D data sets (simultaneously or individually), including: spectra, surface brightness profiles, light curves, general ASCII arrays;*
- *fit 2-D images/surfaces in the Poisson/Gaussian regime;*
- *visualize the data with ChIPS or Matplotlib;*
- *access the internal data arrays;*
- *build complex model expressions;*
- *import and use your own models;*
- *choose appropriate statistics for modeling Poisson or Gaussian data;*
- *import new statistics, with priors if required by analysis;*
- *visualize a parameter space with simulations or using 1-D/2-D cuts of the parameter space;*
- *calculate confidence levels on the best-fit model parameters;*
- *choose a robust optimization method for the fit: Levenberg-Marquardt, Nelder-Mead Simplex or Monte Carlo/Differential Evolution;*
- *perform Bayesian analysis with Poisson Likelihood and priors, using Metropolis or Metropolis-Hastings algorithm in the MCMC (Markov-Chain Monte Carlo);*
- *and use Python to create complex analysis and modeling functions, build the batch mode analysis or extend the provided functionality to meet the required needs.*

The Sherpa infrastructure greatly enhances the default Sherpa functions, and provides users with an environment for developing complex and analysis.

References:

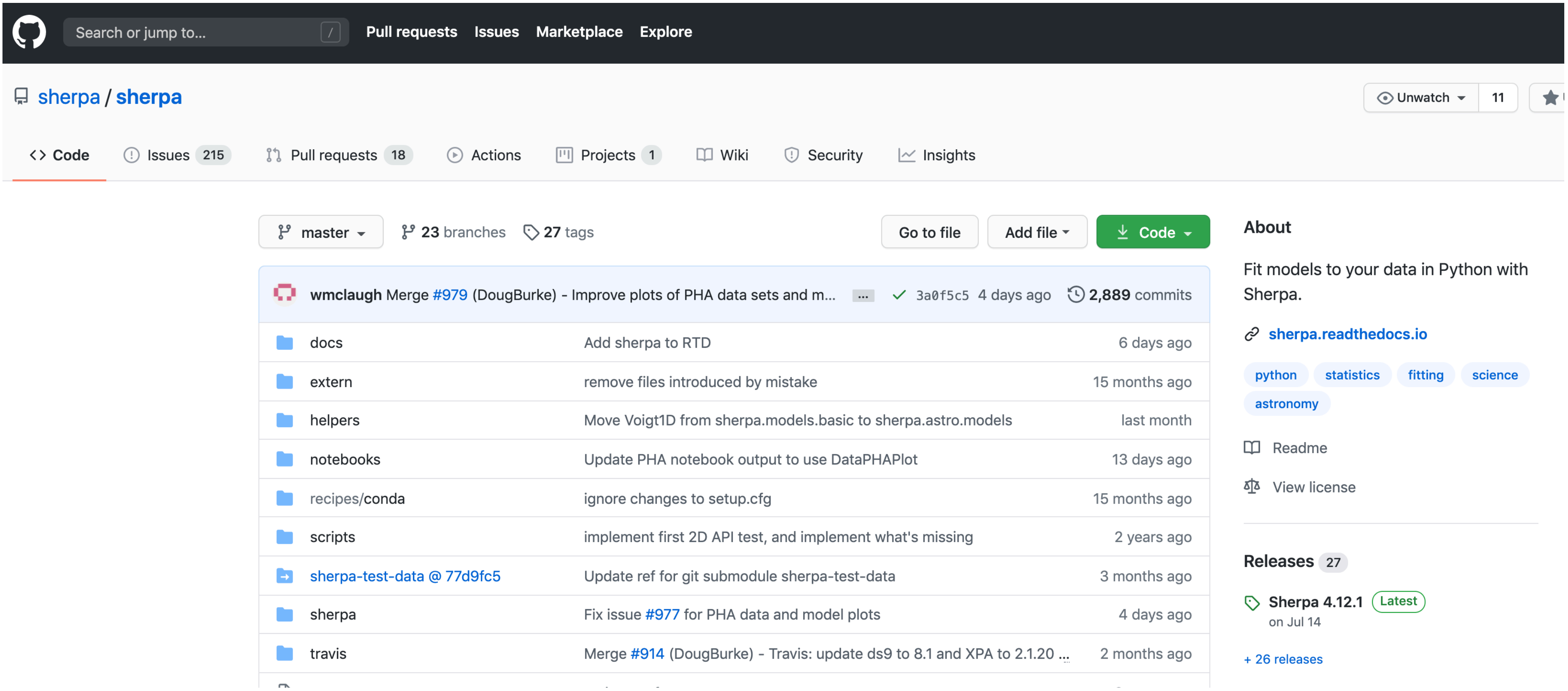
Freeman, P., Doe, S., & Siemiginowska, A. 2001 - *Sherpa: a mission-independent data analysis application* - SPIE 4477, 76

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. 51-57

Source Code and Development on GitHub https://github.com/sherpa/sherpa

Core Team:

Doug Burke, Warren McLaughlin, Dan Nguyen, Moritz Guenther, Aneta Siemiginowska
+ DS/SDS and other contributors



wmclaugh Merge #979 (DougBurke) - Improve plots of PHA data sets and m... 3a0f5c5 4 days ago 2,889 commits

docs	Add sherpa to RTD	6 days ago
extern	remove files introduced by mistake	15 months ago
helpers	Move Voigt1D from sherpa.models.basic to sherpa.astro.models	last month
notebooks	Update PHA notebook output to use DataPHAPlot	13 days ago
recipes/conda	ignore changes to setup.cfg	15 months ago
scripts	implement first 2D API test, and implement what's missing	2 years ago
sherpa-test-data @ 77d9fc5	Update ref for git submodule sherpa-test-data	3 months ago
sherpa	Fix issue #977 for PHA data and model plots	4 days ago
travis	Merge #914 (DougBurke) - Travis: update ds9 to 8.1 and XPA to 2.1.20 ...	2 months ago

About

Fit models to your data in Python with Sherpa.

sherpa.readthedocs.io

python statistics fitting science astronomy

Readme

View license

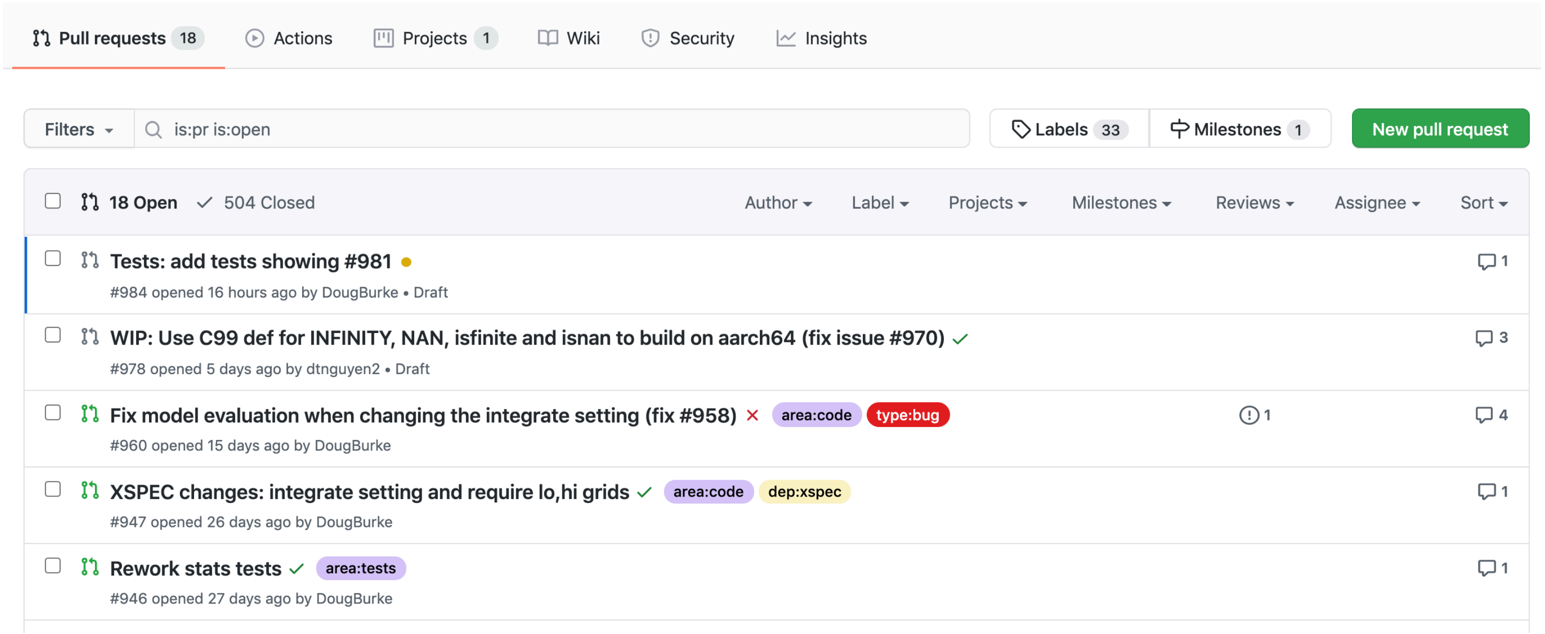
Releases 27

Sherpa 4.12.1 Latest
on Jul 14

+ 26 releases

Open Development on GitHub

Code contributions



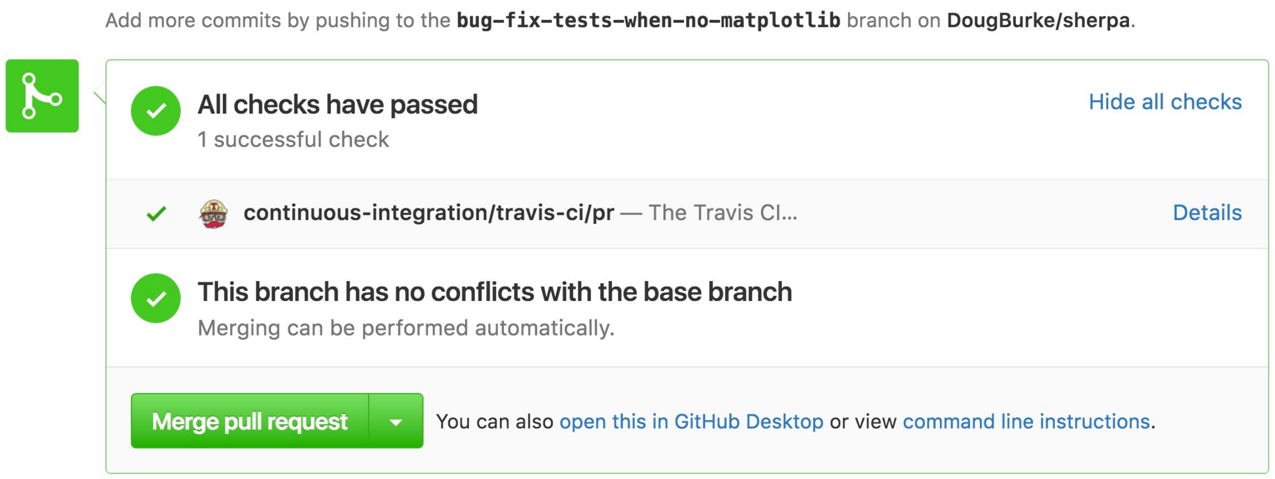
Pull requests 18 · Actions · Projects 1 · Wiki · Security · Insights

Filters · is:pr is:open · Labels 33 · Milestones 1 · New pull request

18 Open ✓ 504 Closed · Author · Label · Projects · Milestones · Reviews · Assignee · Sort

- Tests: add tests showing #981 ● — #984 opened 16 hours ago by DougBurke • Draft — 1
- WIP: Use C99 def for INFINITY, NAN, isfinite and isnan to build on aarch64 (fix issue #970) ✓ — #978 opened 5 days ago by dtnguyen2 • Draft — 3
- Fix model evaluation when changing the integrate setting (fix #958) ✗ area:code type:bug — #960 opened 15 days ago by DougBurke — 1 — 4
- XSPEC changes: integrate setting and require lo,hi grids ✓ area:code dep:xspec — #947 opened 26 days ago by DougBurke — 1
- Rework stats tests ✓ area:tests — #946 opened 27 days ago by DougBurke — 1

Travis continuous integration testing



Add more commits by pushing to the **bug-fix-tests-when-no-matplotlib** branch on **DougBurke/sherpa**.

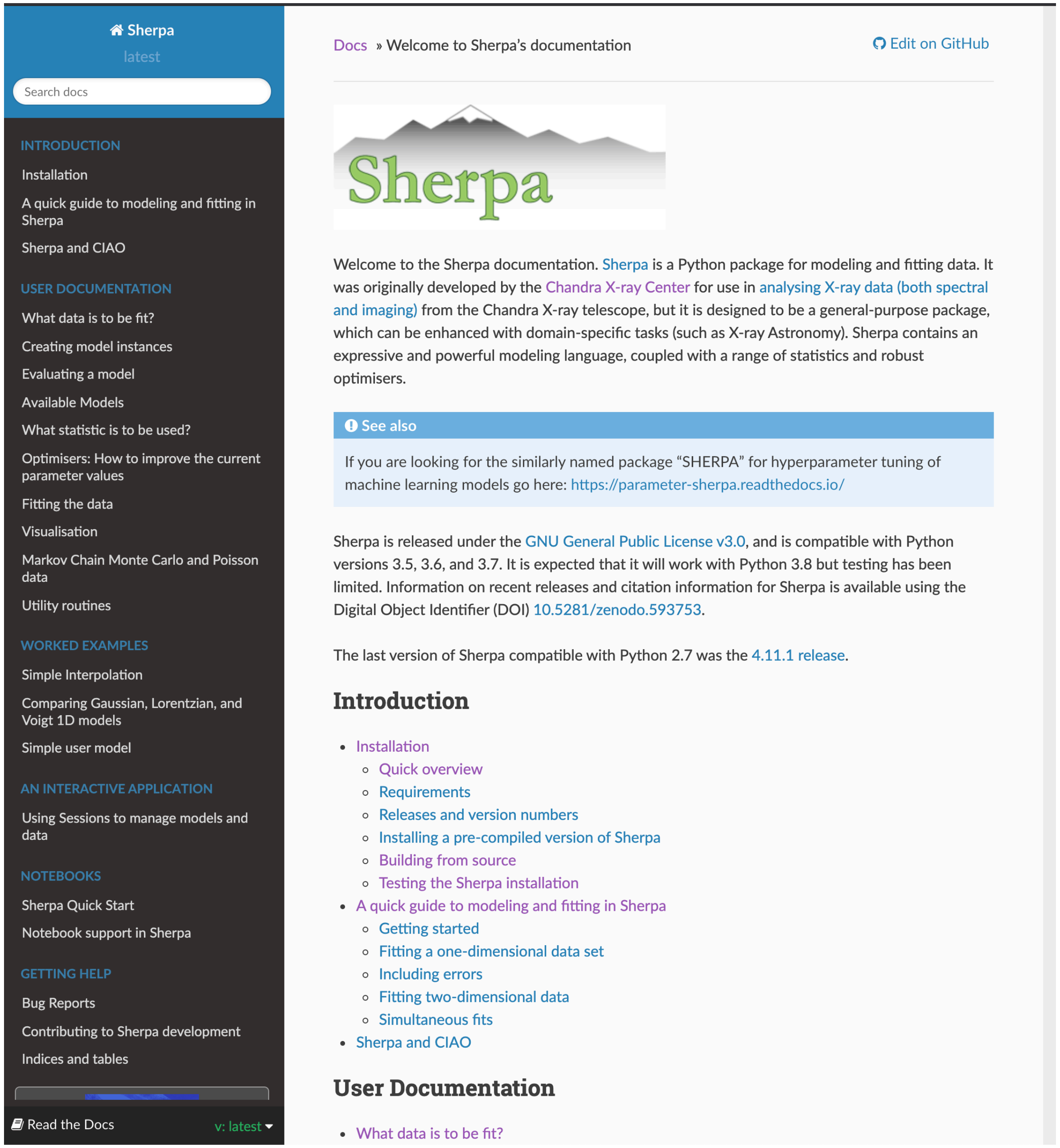
All checks have passed
1 successful check
Hide all checks

✓ **continuous-integration/travis-ci/pr** — The Travis CI… Details

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request · You can also open this in GitHub Desktop or view command line instructions.

Python Documentation



Sherpa
latest

Search docs

INTRODUCTION
Installation
A quick guide to modeling and fitting in Sherpa
Sherpa and CIAO

USER DOCUMENTATION
What data is to be fit?
Creating model instances
Evaluating a model
Available Models
What statistic is to be used?
Optimisers: How to improve the current parameter values
Fitting the data
Visualisation
Markov Chain Monte Carlo and Poisson data
Utility routines

WORKED EXAMPLES
Simple Interpolation
Comparing Gaussian, Lorentzian, and Voigt 1D models
Simple user model

AN INTERACTIVE APPLICATION
Using Sessions to manage models and data

NOTEBOOKS
Sherpa Quick Start
Notebook support in Sherpa

GETTING HELP
Bug Reports
Contributing to Sherpa development
Indices and tables

Read the Docs v: latest

Docs » Welcome to Sherpa's documentation   Edit on GitHub

Sherpa

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.

See also

If you are looking for the similarly named package "SHERPA" for hyperparameter tuning of machine learning models go here: https://parameter-sherpa.readthedocs.io/

Sherpa is released under the GNU General Public License v3.0, and is compatible with Python versions 3.5, 3.6, and 3.7. It is expected that it will work with Python 3.8 but testing has been limited. Information on recent releases and citation information for Sherpa is available using the Digital Object Identifier (DOI) 10.5281/zenodo.593753.

The last version of Sherpa compatible with Python 2.7 was the 4.11.1 release.

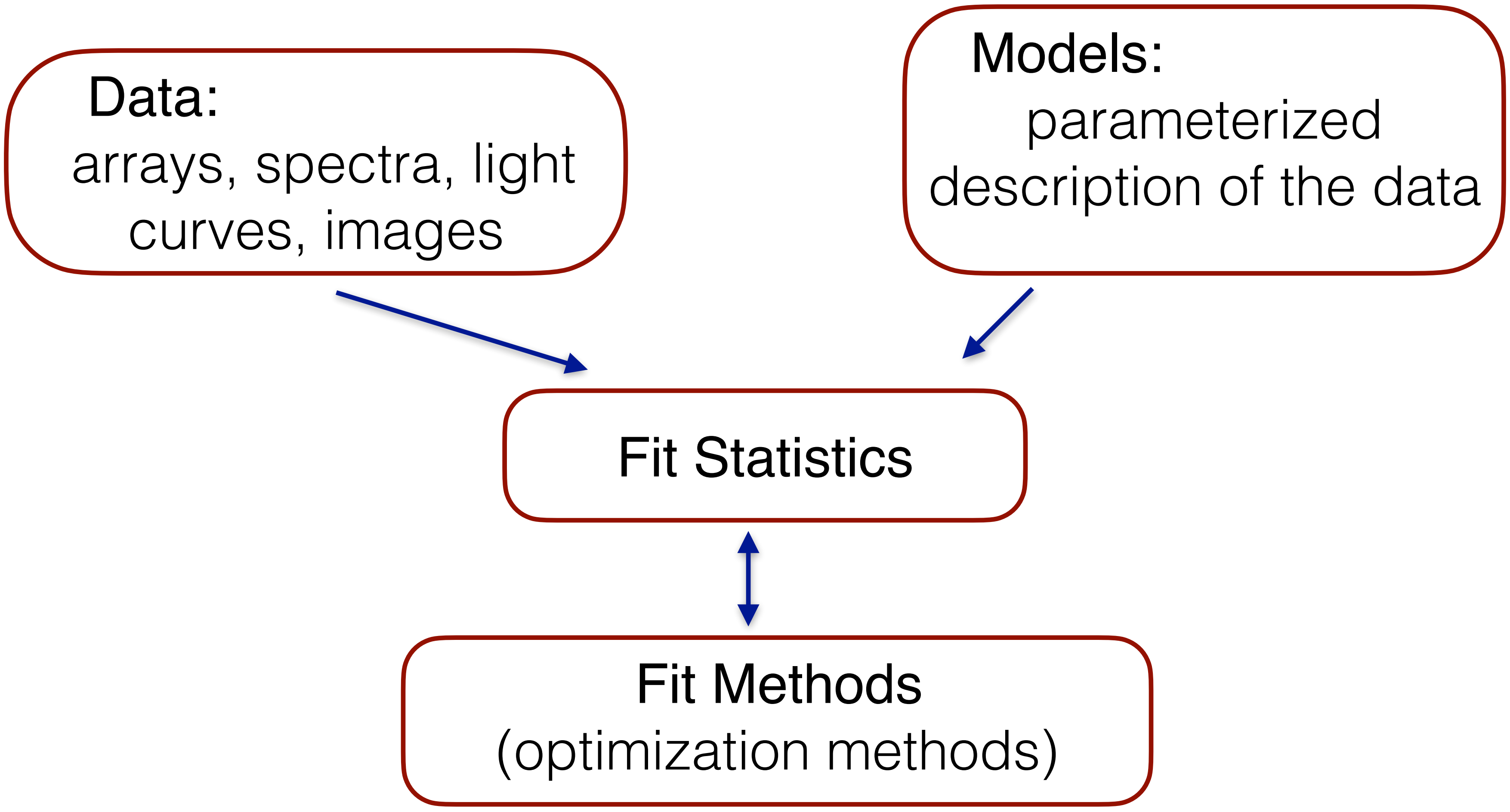
Introduction

- Installation
  - Quick overview
  - 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
  - Simultaneous fits
- Sherpa and CIAO

User Documentation

- What data is to be fit?

Sherpa Components


Data:
arrays, spectra, light curves, images
Models:
parameterized description of the data
Fit Statistics
Fit Methods
(optimization methods)

Sherpa Components

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:
matplotlib, ds9

Final Evaluation & Conclusions
statistical tests, model selection

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

- Input data:

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

- Multiple Datasets - data id

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

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”))

- Filtering the data
  load_data expressions
  notice/ignore commands in Sherpa

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

Models in Sherpa

- Parameterized models: $M(x_i, p_k)$

  x_i - independent grid, i.e. energy

  p_k - parameters,

  examples: absorption column - N_H

  photon index of a power law function - Γ

  blackbody temperature kT

- Library of models:

```
sherpa In [1]: list_models()
    Out[1]:
    ['absorptionedge',
    'absorptiongaussian',
     'absorptionlorentz',….
```

- Model language to build compound model expressions.
- Add user models.

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:

```
set_source(1,'atten.abs1*atten.abs2*powlaw1d.p1')
set_source(2,'abs1*abs2*powlaw1d.p2')
```

- Building a model expression with convolved and unconvolved components:

```
set_full_model(1,'psf(gauss2d.g2)+const2d.c1')
```

Building Models: Examples

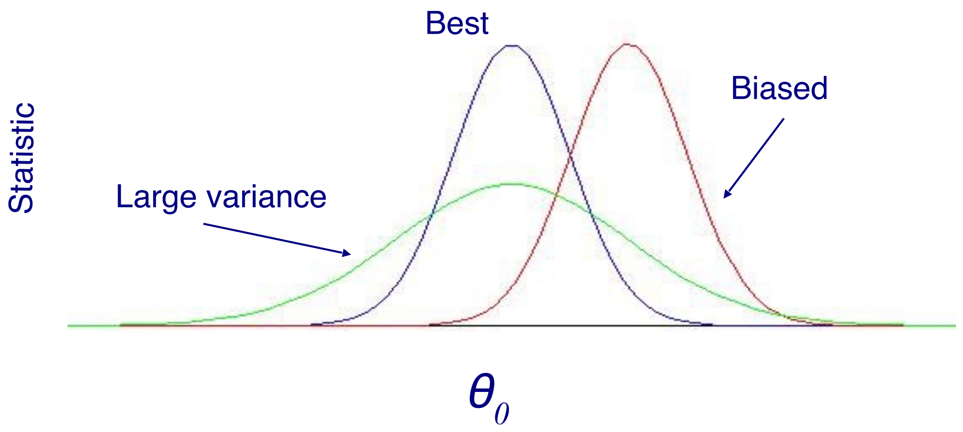
- Source and Background models:

```
set_source(2,’xsphabs.abs1*(powlaw1d.p1+gauss1d.g1)’)
set_bkg_model(2,’const1d.mybkg’)
```

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')
```

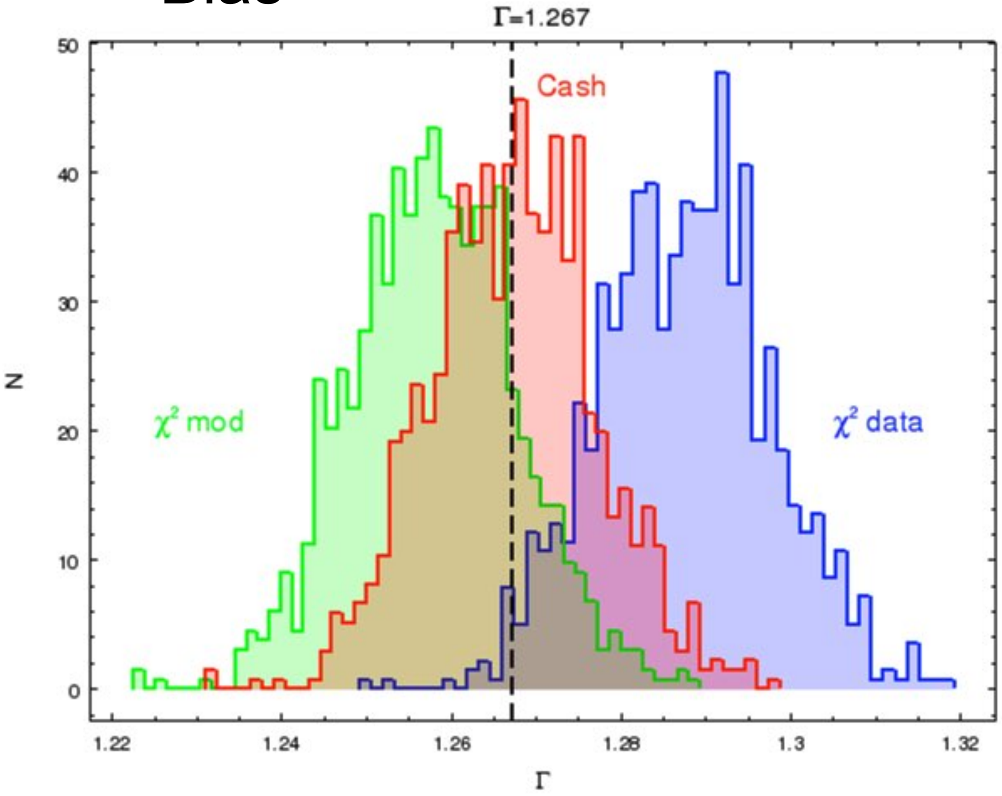


chi2 statistics - appropriate for Gaussian data
Poisson likelihood - cash/cstat/wstat

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')
```

Bias



"Handbook of X-ray Astronomy "
(2011), Arnaud, Smith, Siemiginowska

see the Notebook:
https://cxc.harvard.edu/ciao/workshop/oct20_egypt_virt/cstat_vs_chisq_SimsNotebook.ipynb

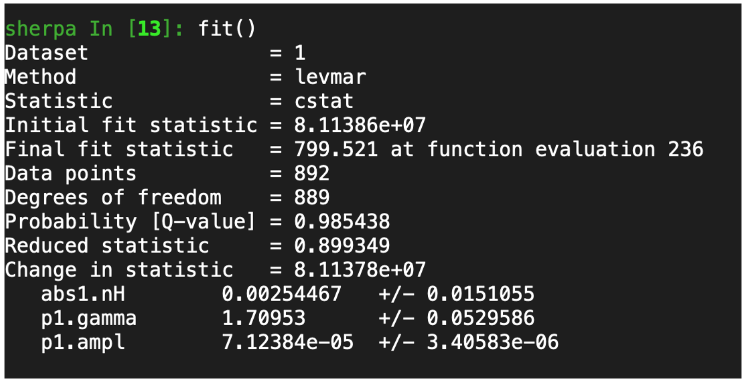
Model Fitting:

Search the Model Parameter Space for the Best Model Parameters

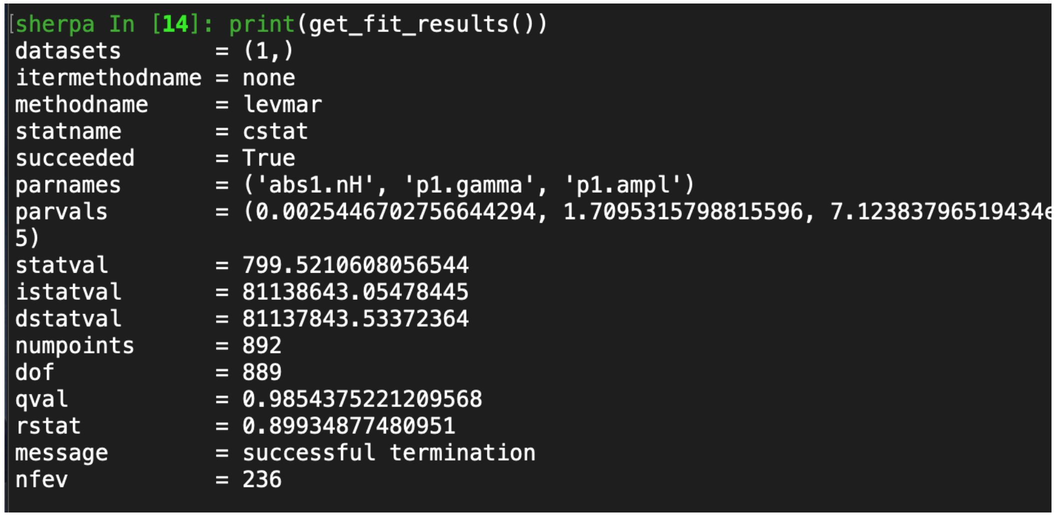
NOTE:

The fit result is as good (or as bad) as your model.

Model misspecification is often a result of bad fit!



```
sherpa In [13]: fit()
Dataset               = 1
Method                = levmar
Statistic             = cstat
Initial fit statistic = 8.11386e+07
Final fit statistic   = 799.521 at function evaluation 236
Data points           = 892
Degrees of freedom    = 889
Probability [Q-value] = 0.985438
Reduced statistic     = 0.899349
Change in statistic   = 8.11378e+07
   abs1.nH        0.00254467   +/- 0.0151055
   p1.gamma       1.70953      +/- 0.0529586
   p1.ampl        7.12384e-05  +/- 3.40583e-06
```



```
[sherpa In [14]: print(get_fit_results())
datasets       = (1,)
itermethodname = none
methodname     = levmar
statname       = cstat
succeeded      = True
parnames       = ('abs1.nH', 'p1.gamma', 'p1.ampl')
parvals        = (0.0025446702756644294, 1.7095315798815596, 7.12383796519434e
5)
statval        = 799.5210608056544
istatval       = 81138643.05478445
dstatval       = 81137843.53372364
numpoints      = 892
dof            = 889
qval           = 0.9854375221209568
rstat          = 0.89934877480951
message        = successful termination
nfev           = 236
```

Fitting: Sherpa Optimization Methods

- Optimization - a minimization of a function:

  ‘A general function *f(x,p)* may have <u>many isolated local minima</u>, 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

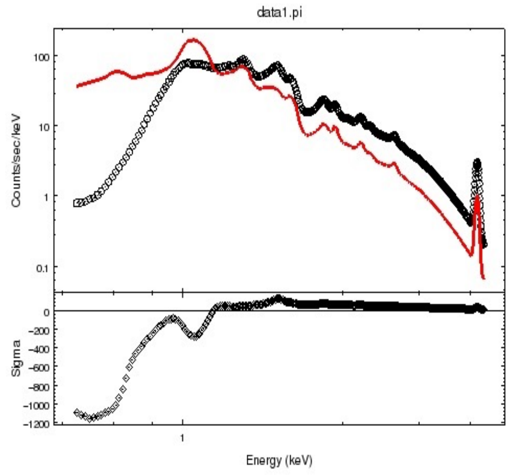
Optimization Methods: Comparison

Example: Spectral Fit with 3 methods

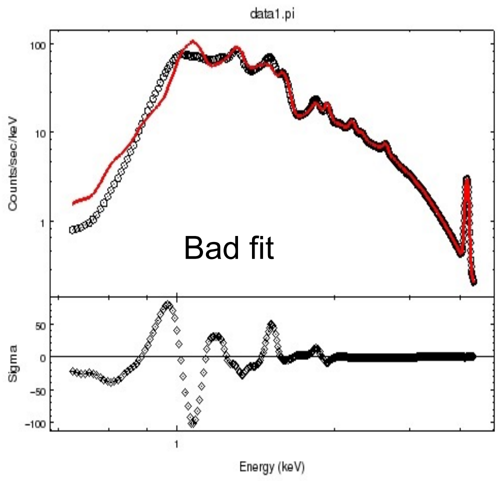
Data: high S/N simulated ACIS-S spectrum of the two temperature plasma
Model: photoelectric absorption plus two MEKAL components (correlated!)

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

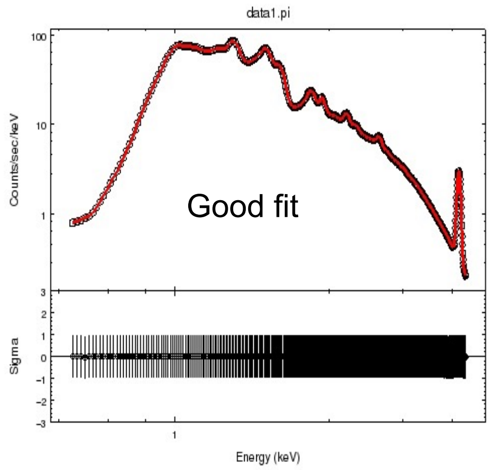
Method	Number of Iterations	Final Statistics
Levmar	31	1.55e5
Neldermead	1494	0.0542
Moncar	13045	0.0542



Data and Model with initial parameters



Levmar fit



Nelder-Mead and Moncar fit

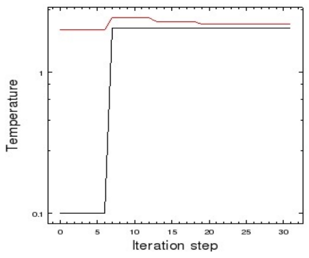
Optimization Methods: Probing Parameter Space

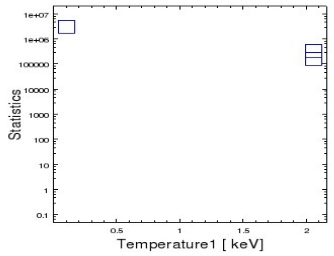
Temperature

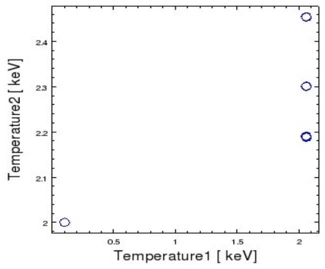
Statistics vs. Temperature

2D slice of Parameter Space probed by each method

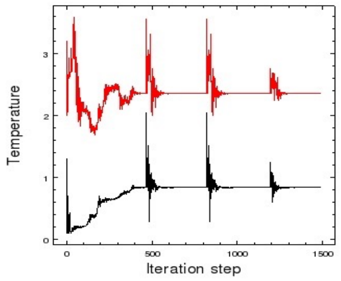
levmar

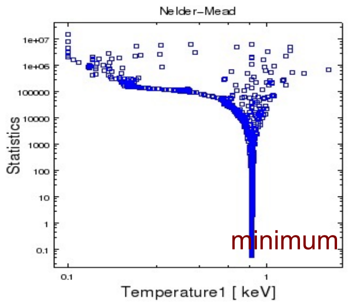


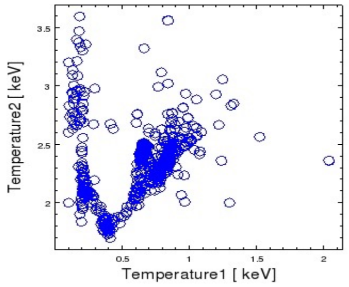




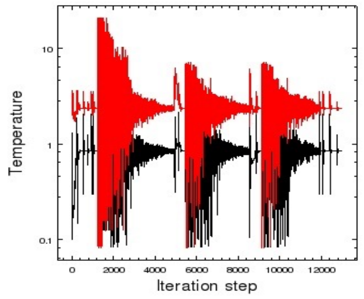
simplex

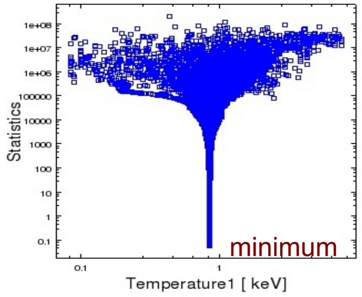


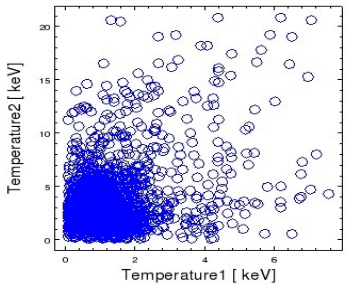




moncar



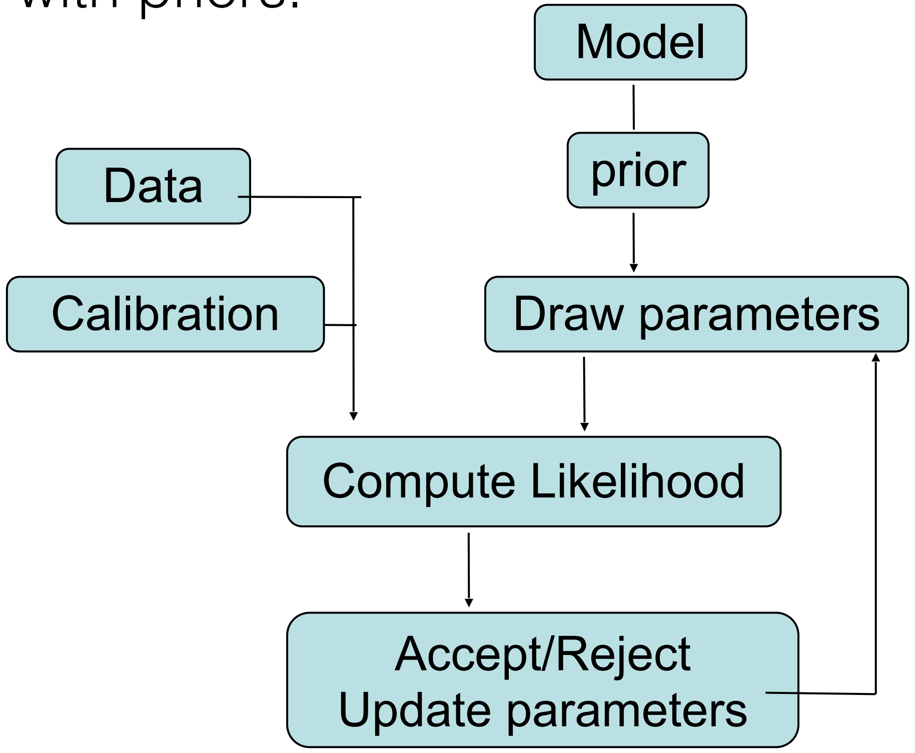




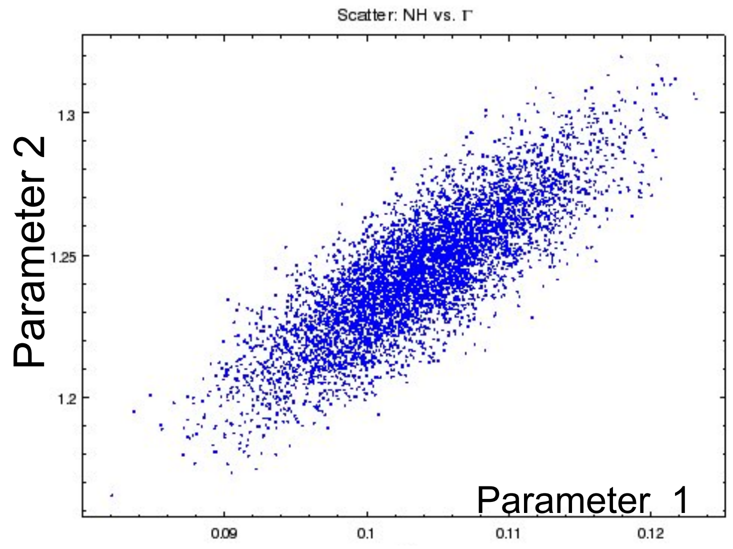
Sherpa, MCMC and Bayesian Analysis

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

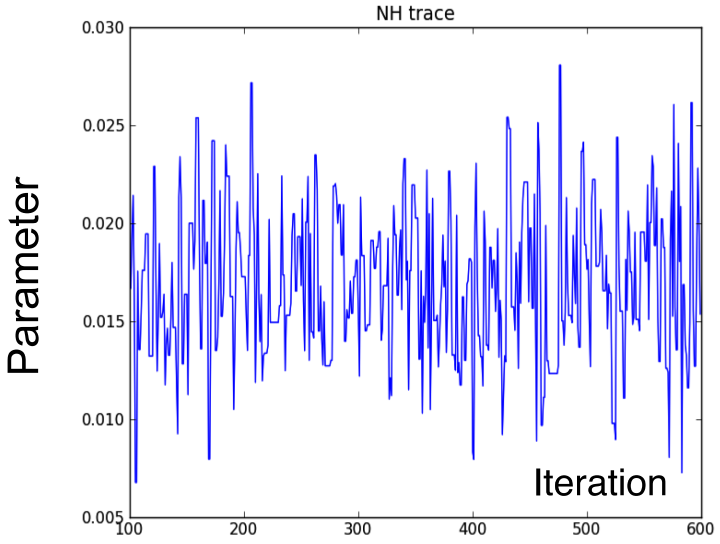
- Explores parameter space and summarizes the full posterior or profile posterior distributions.
- Computed parameter uncertainties can include systematic or calibration errors.
- Simulates replicate data from the posterior predictive distributions.

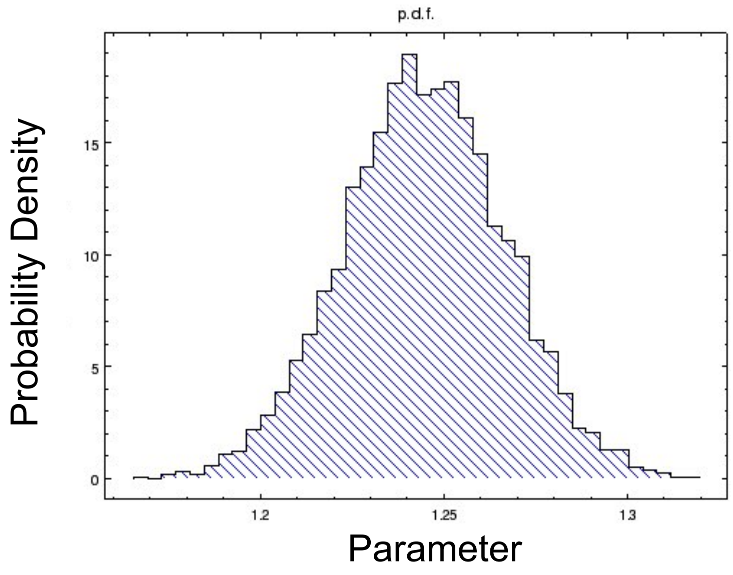


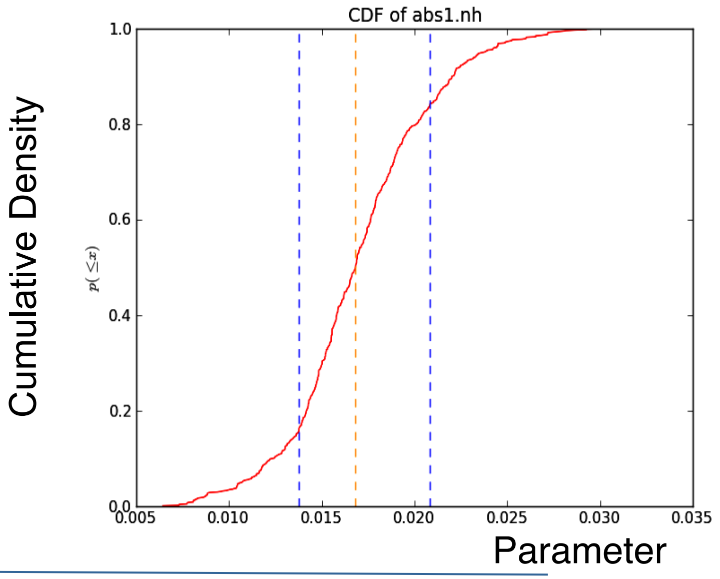
Visualization of the MCMC Results



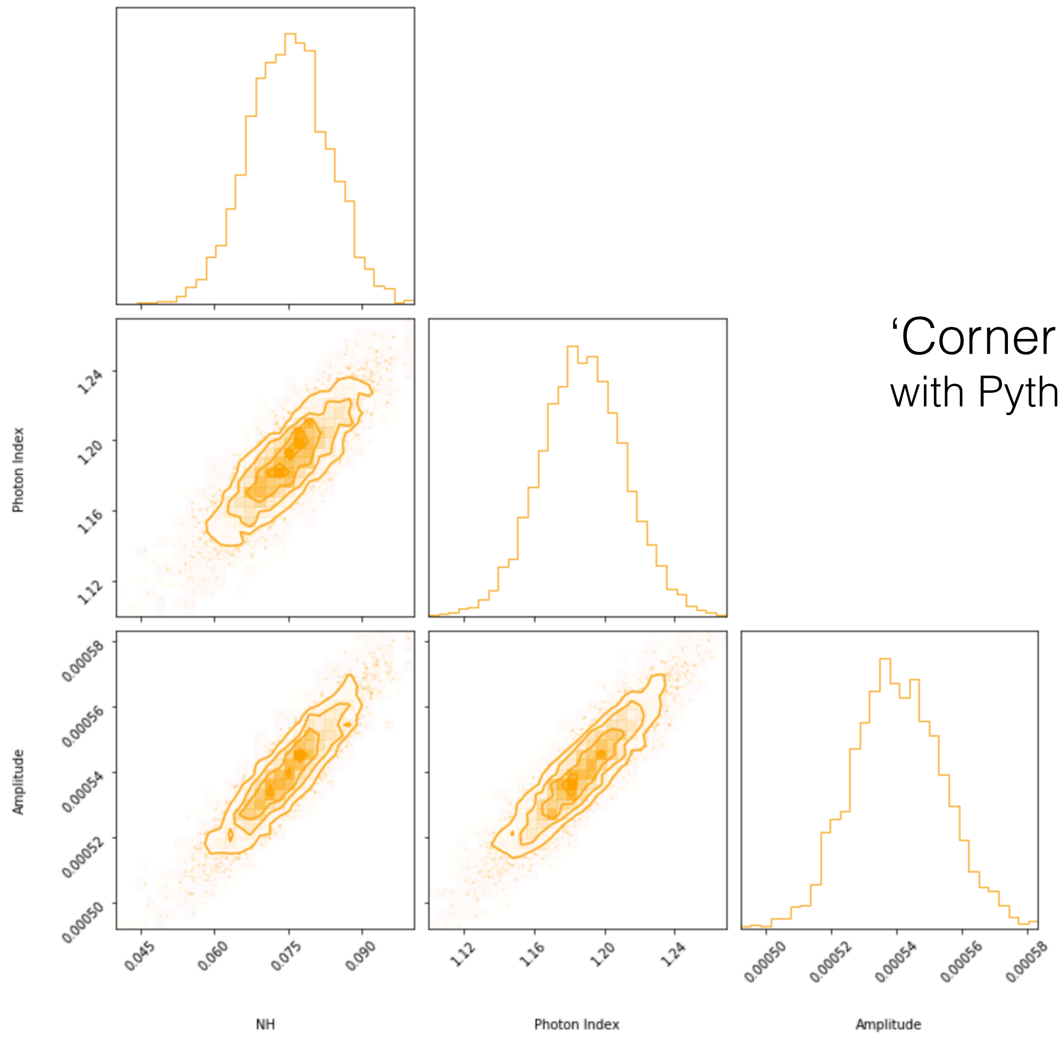
Trace of a parameter during MCMC run







Visualization of the MCMC Results



'Corner Plots'
with Python package `corner`

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):

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

ν - degrees of freedom
α - level
χ^2_{min} - minimum

Δ depends only on the number of parameters involved not on goodness of fit

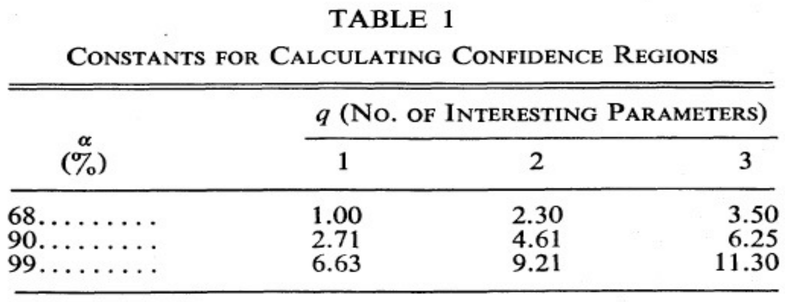
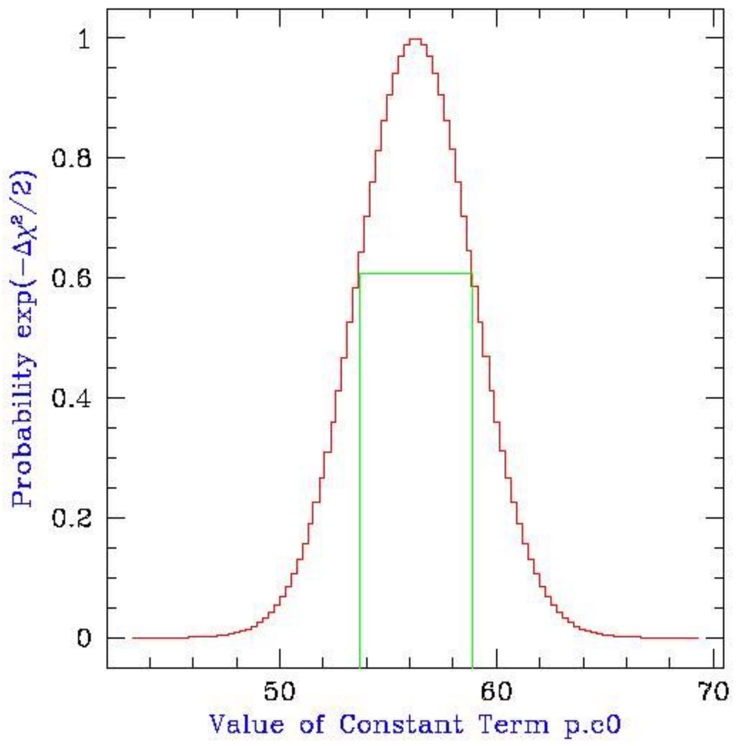
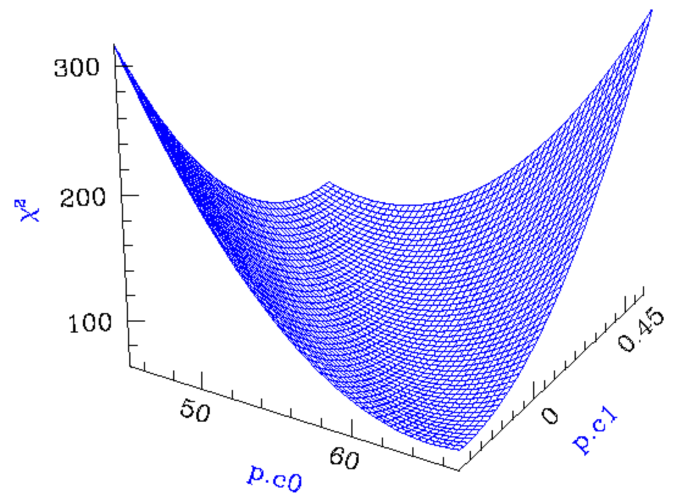


TABLE 1

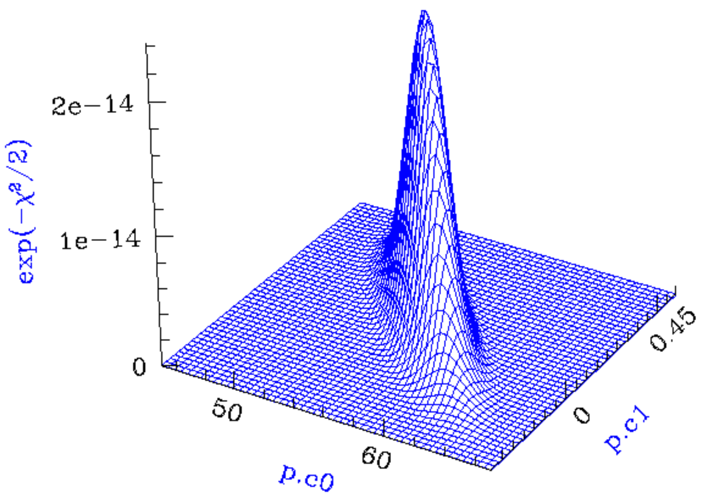
CONSTANTS FOR CALCULATING CONFIDENCE REGIONS

α (%)	q (No. of Interesting Parameters) 1	2	3
68.........	1.00	2.30	3.50
90.........	2.71	4.61	6.25
99.........	6.63	9.21	11.30



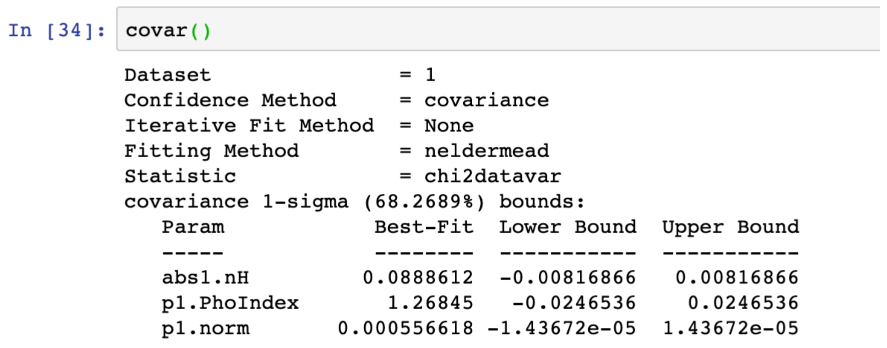


Calculating Confidence Limits means Exploring the Parameter Space - Statistical Surface

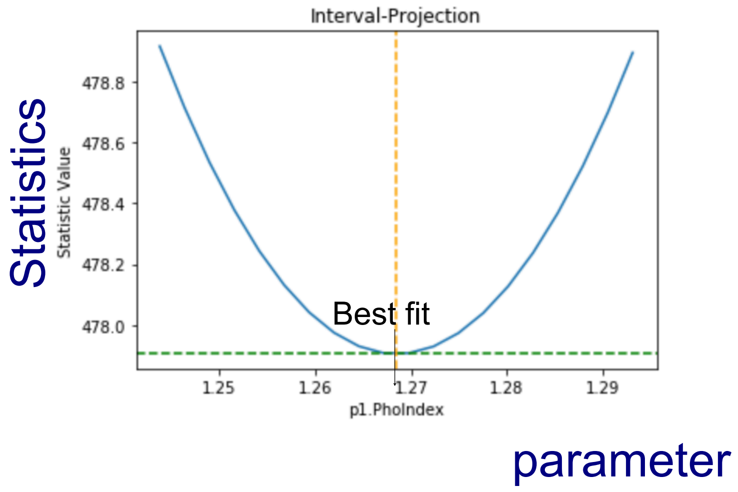


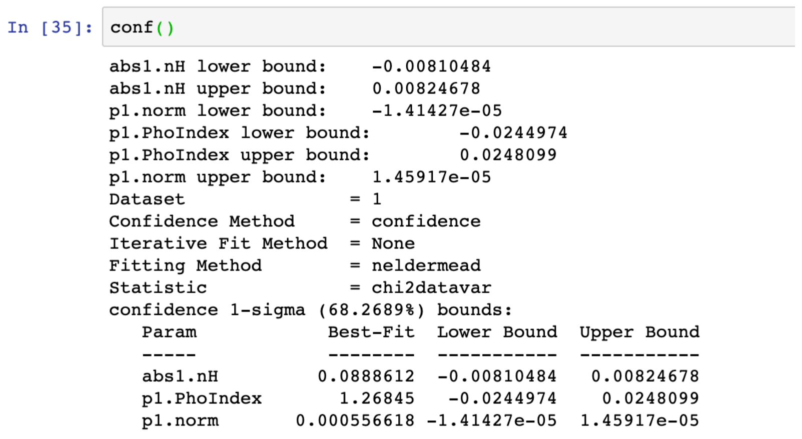
Example of a “well-behaved” statistical surface in parameter space, viewed as a multi-dimensional paraboloid (χ^2, top), and as a multi-dimensional Gaussian ($\exp(-\chi^2/2) \approx L$, bottom).

Confidence Intervals

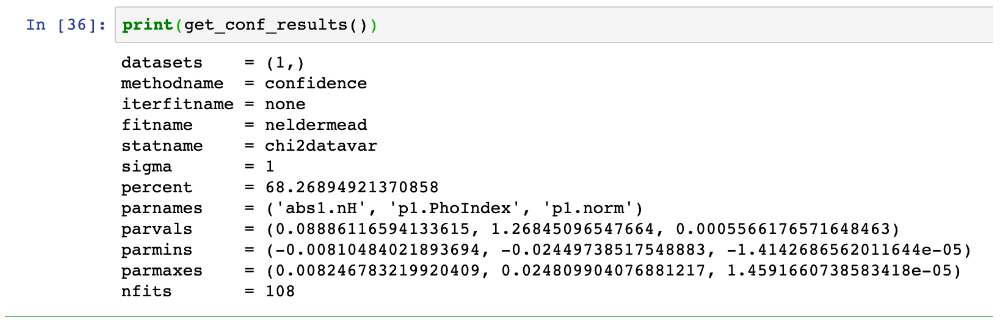


```
In [34]: covar()

         Dataset               = 1
         Confidence Method     = covariance
         Iterative Fit Method  = None
         Fitting Method        = neldermead
         Statistic             = chi2datavar
         covariance 1-sigma (68.2689%) bounds:
            Param            Best-Fit  Lower Bound  Upper Bound
            -----            --------  -----------  -----------
            abs1.nH         0.0888612  -0.00816866   0.00816866
            p1.PhoIndex       1.26845   -0.0246536    0.0246536
            p1.norm       0.000556618 -1.43672e-05  1.43672e-05
```



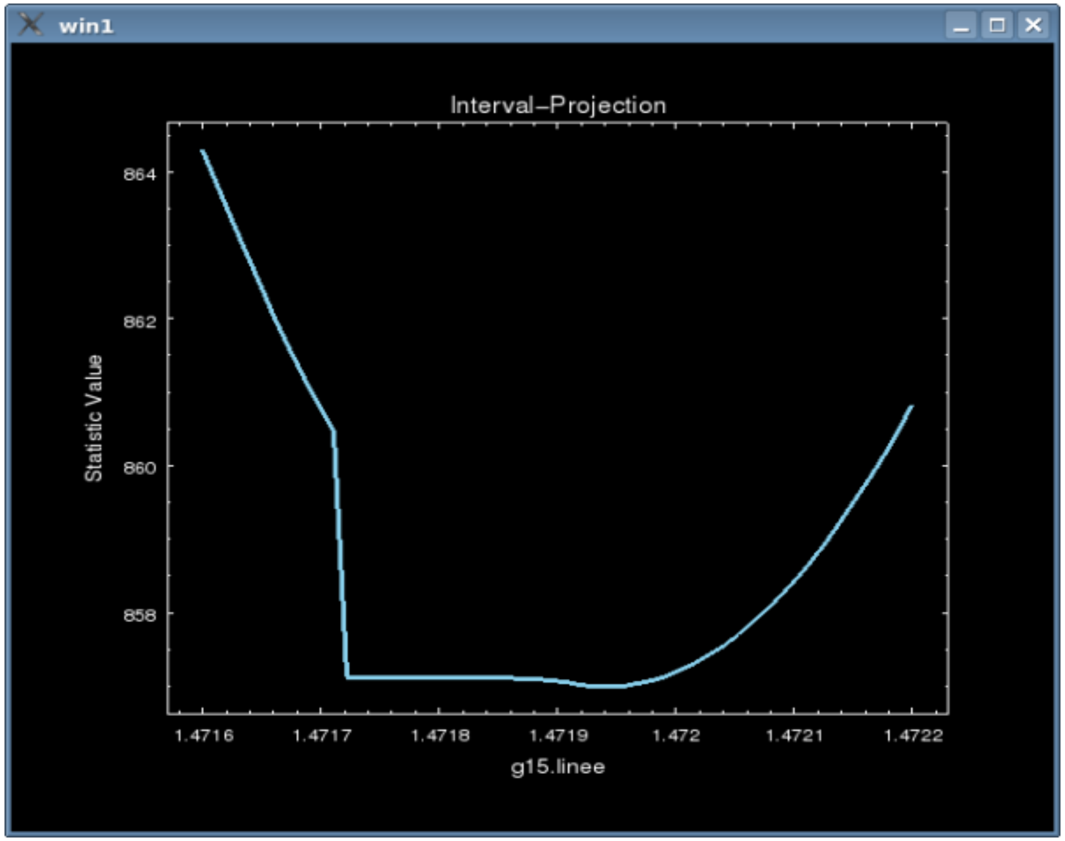


```
In [35]: conf()

         abs1.nH lower bound:    -0.00810484
         abs1.nH upper bound:    0.00824678
         p1.norm lower bound:    -1.41427e-05
         p1.PhoIndex lower bound:        -0.0244974
         p1.PhoIndex upper bound:        0.0248099
         p1.norm upper bound:    1.45917e-05
         Dataset               = 1
         Confidence Method     = confidence
         Iterative Fit Method  = None
         Fitting Method        = neldermead
         Statistic             = chi2datavar
         confidence 1-sigma (68.2689%) bounds:
            Param            Best-Fit  Lower Bound  Upper Bound
            -----            --------  -----------  -----------
            abs1.nH         0.0888612  -0.00810484   0.00824678
            p1.PhoIndex       1.26845   -0.0244974    0.0248099
            p1.norm       0.000556618 -1.41427e-05  1.45917e-05
```



```
In [36]: print(get_conf_results())

         datasets    = (1,)
         methodname  = confidence
         iterfitname = none
         fitname     = neldermead
         statname    = chi2datavar
         sigma       = 1
         percent     = 68.26894921370858
         parnames    = ('abs1.nH', 'p1.PhoIndex', 'p1.norm')
         parvals     = (0.08886116594133615, 1.26845096547664, 0.0005566176571648463)
         parmins     = (-0.00810484021893694, -0.02449738517548883, -1.4142686562011644e-05)
         parmaxes    = (0.008246783219920409, 0.024809904076881217, 1.4591660738583418e-05)
         nfits       = 108
```

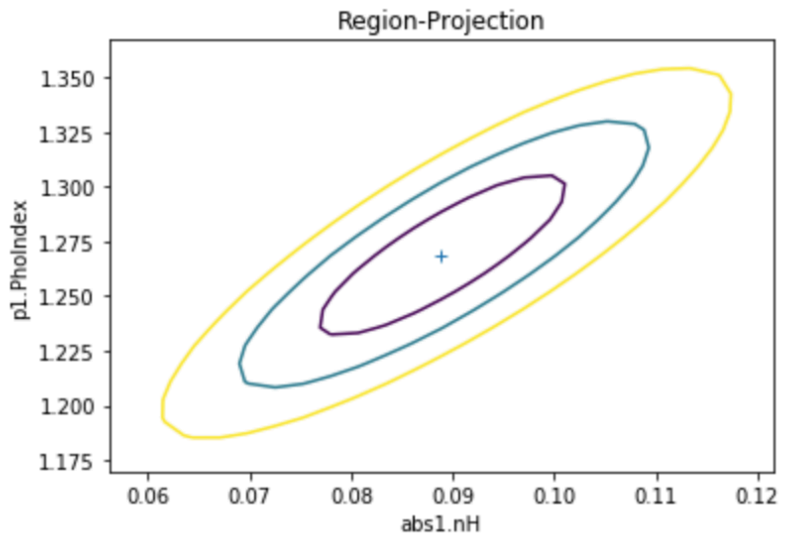
Not well-behaved Surface



Non-Gaussian Shape

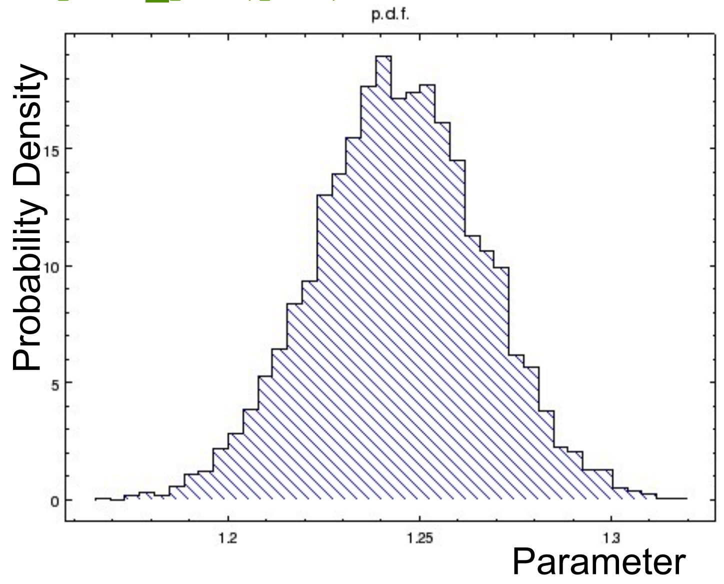
Confidence Regions

```
In [42]: reg_proj(abs1.nH, p1.PhoIndex, nloop=[25,25])

WARNING: Setting optimization to levmar for region projection plot
```

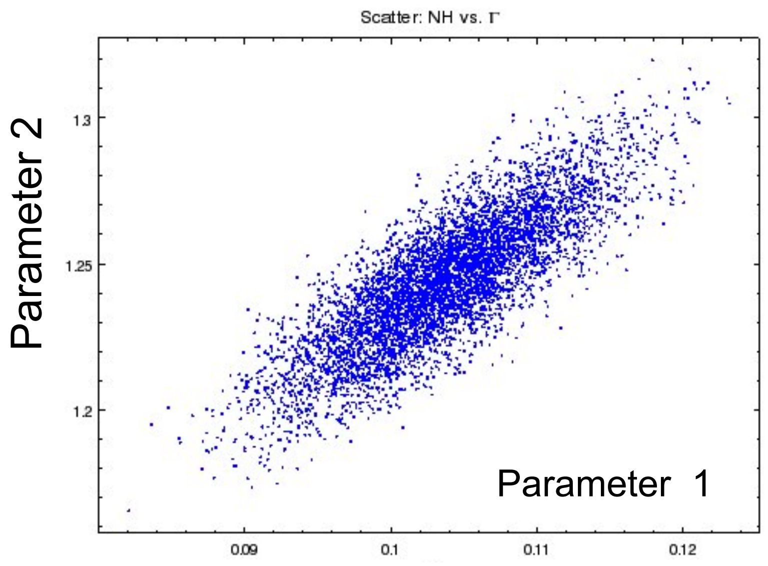


MCMC Results: Probability Distributions

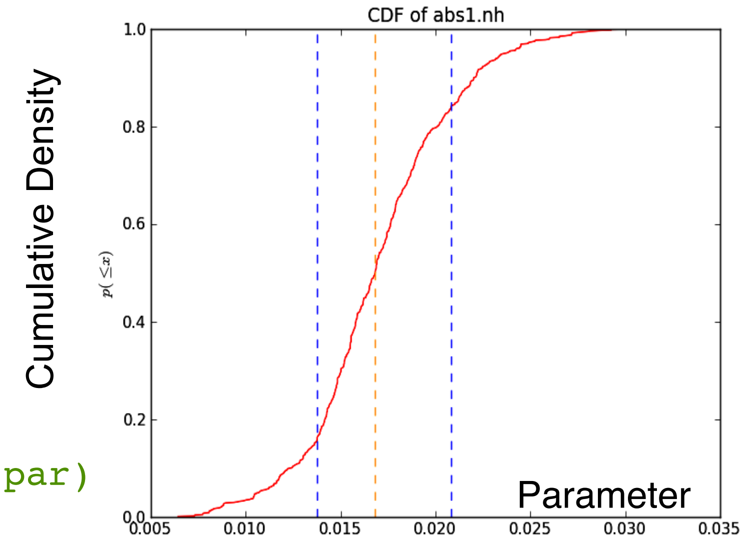
`plot_pdf(par)`



`plot_scatter(par1,par2)`



`plot_cdf(par)`

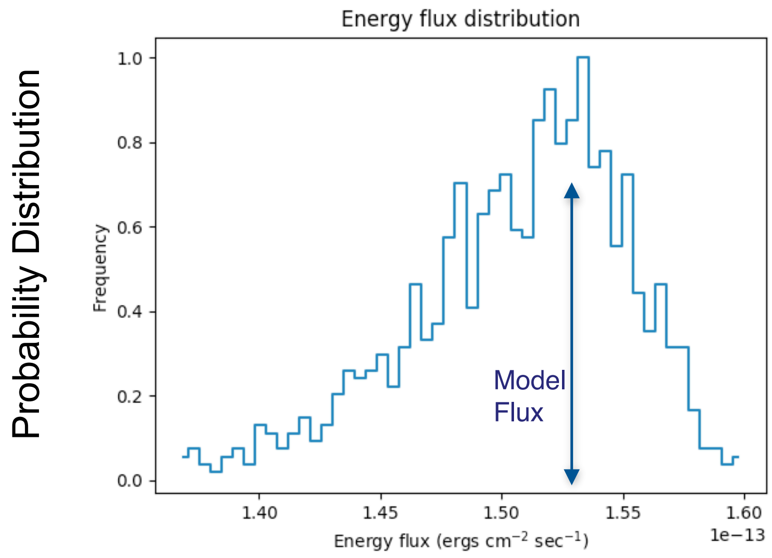


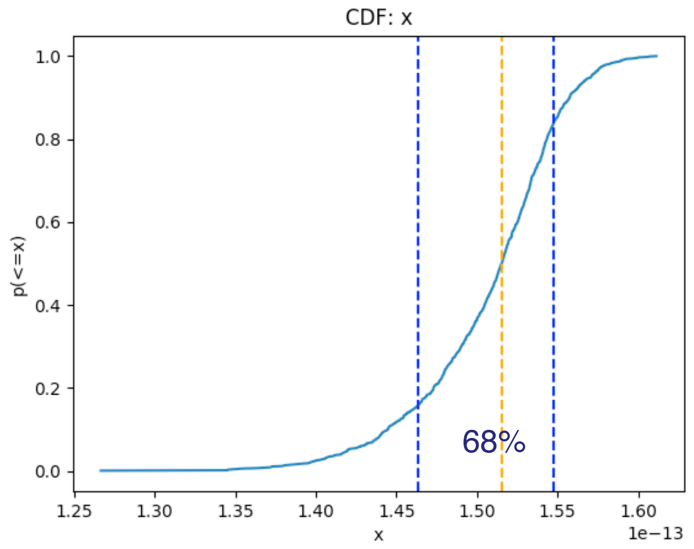
Flux Uncertainties

Functions: `sample_energy_flux, sample_flux`

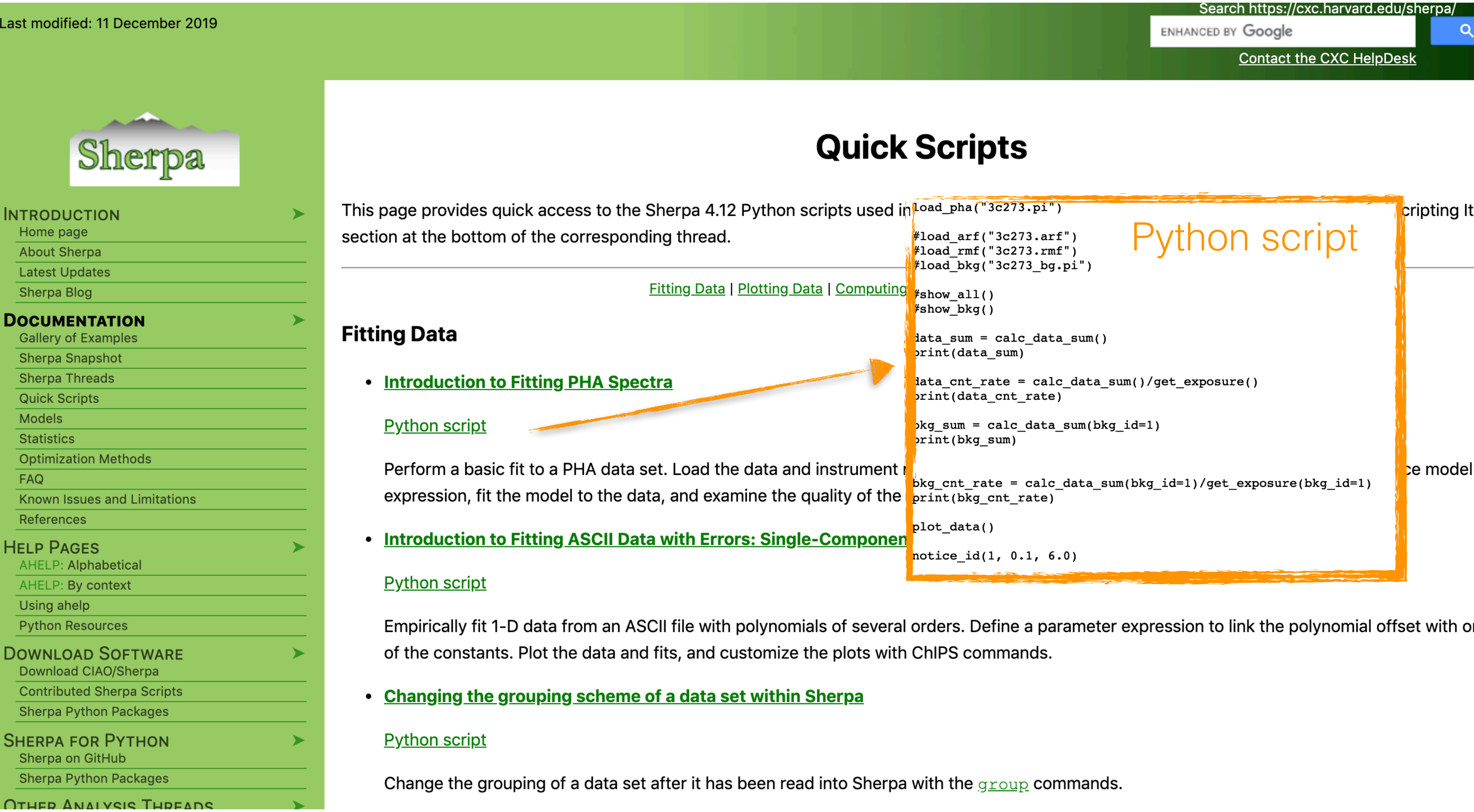
Monte Carlo Simulations of parameters assuming Gaussian distributions for all the parameters
Characterized by the covariance matrix, includes correlations between parameters.

```
sherpa In [6]: flux100=sample_energy_flux(0.5,2.,num=100)
sherpa In [7]: 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 In [8]: plot_energy_flux(0.5,2,num=1000)
```





Sherpa in CIAO



Last modified: 11 December 2019

Search https://cxc.harvard.edu/sherpa/

Contact the CXC HelpDesk

Introduction
- Home page
- About Sherpa
- Latest Updates
- Sherpa Blog

Documentation
- Gallery of Examples
- Sherpa Snapshot
- Sherpa Threads
- Quick Scripts
- Models
- Statistics
- Optimization Methods
- FAQ
- Known Issues and Limitations
- References

Help Pages
- AHELP: Alphabetical
- AHELP: By context
- Using ahelp
- Python Resources

Download Software
- Download CIAO/Sherpa
- Contributed Sherpa Scripts
- Sherpa Python Packages

Sherpa for Python
- Sherpa on GitHub
- Sherpa Python Packages

Other Analysis Threads

Quick Scripts

This page provides quick access to the Sherpa 4.12 Python scripts used in cripting It section at the bottom of the corresponding thread.

Fitting Data | Plotting Data | Computing

Fitting Data

- **Introduction to Fitting PHA Spectra**

  Python script

  Perform a basic fit to a PHA data set. Load the data and instrument ce model expression, fit the model to the data, and examine the quality of the

- **Introduction to Fitting ASCII Data with Errors: Single-Componen**

  Python script

  Empirically fit 1-D data from an ASCII file with polynomials of several orders. Define a parameter expression to link the polynomial offset with o of the constants. Plot the data and fits, and customize the plots with ChIPS commands.

- **Changing the grouping scheme of a data set within Sherpa**

  Python script

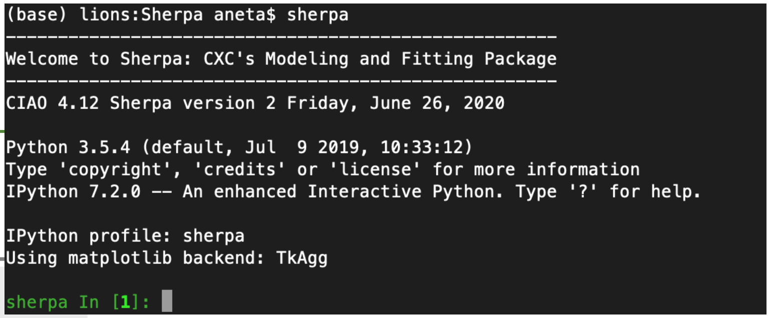
  Change the grouping of a data set after it has been read into Sherpa with the group commands.

Python script

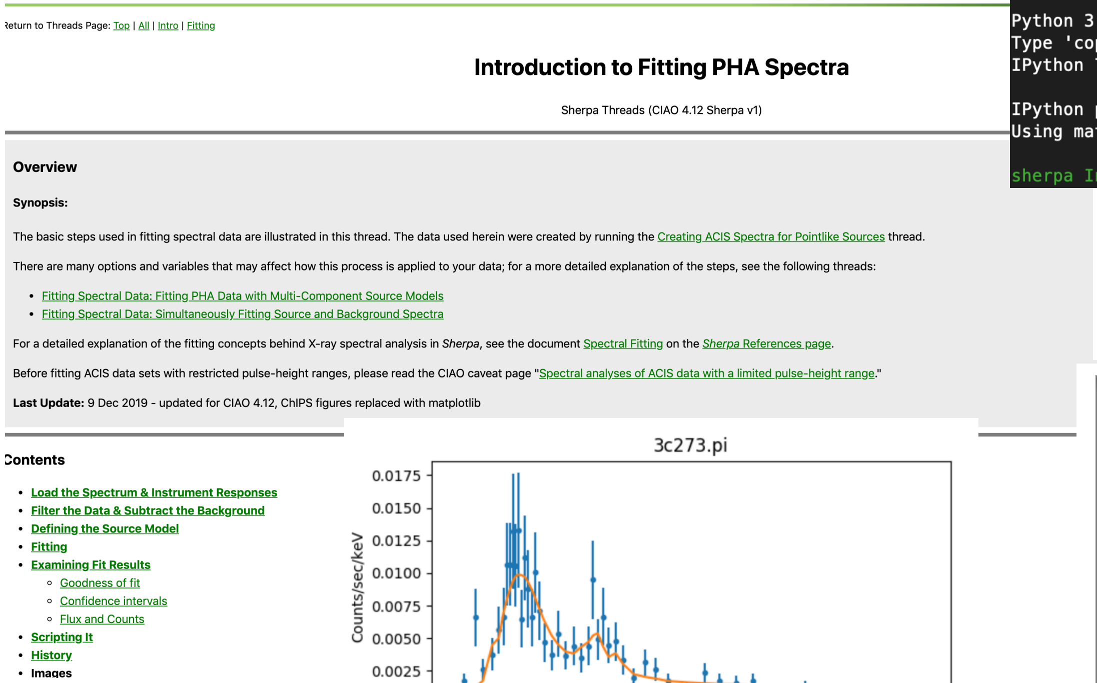
```
load_pha("3c273.pi")

#load_arf("3c273.arf")
#load_rmf("3c273.rmf")
#load_bkg("3c273_bg.pi")

#show_all()
#show_bkg()

data_sum = calc_data_sum()
print(data_sum)

data_cnt_rate = calc_data_sum()/get_exposure()
print(data_cnt_rate)

bkg_sum = calc_data_sum(bkg_id=1)
print(bkg_sum)

bkg_cnt_rate = calc_data_sum(bkg_id=1)/get_exposure(bkg_id=1)
print(bkg_cnt_rate)

plot_data()

notice_id(1, 0.1, 6.0)
```

Sherpa in CIAO

Start Sherpa



```
(base) lions:Sherpa aneta$ sherpa
-----------------------------------------------------
Welcome to Sherpa: CXC's Modeling and Fitting Package
-----------------------------------------------------
CIAO 4.12 Sherpa version 2 Friday, June 26, 2020

Python 3.5.4 (default, Jul  9 2019, 10:33:12)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.2.0 -- An enhanced Interactive Python. Type '?' for help.

IPython profile: sherpa
Using matplotlib backend: TkAgg

sherpa In [1]: 
```



Return to Threads Page: Top | All | Intro | Fitting

Introduction to Fitting PHA Spectra

Sherpa Threads (CIAO 4.12 Sherpa v1)

Overview

Synopsis:

The basic steps used in fitting spectral data are illustrated in this thread. The data used herein were created by running the Creating ACIS Spectra for Pointlike Sources thread.

There are many options and variables that may affect how this process is applied to your data; for a more detailed explanation of the steps, see the following threads:

- Fitting Spectral Data: Fitting PHA Data with Multi-Component Source Models
- Fitting Spectral Data: Simultaneously Fitting Source and Background Spectra

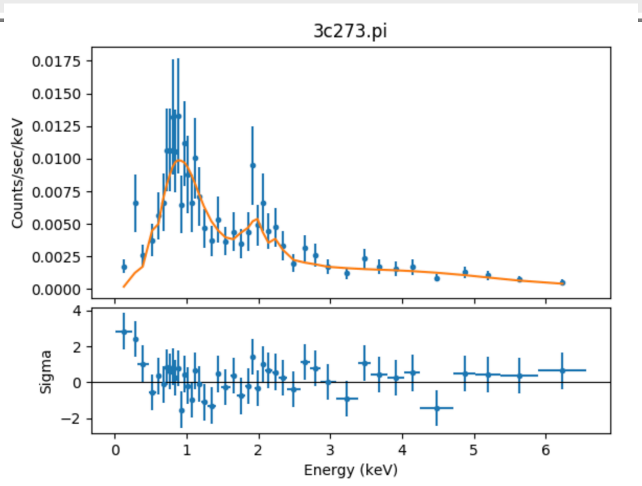
For a detailed explanation of the fitting concepts behind X-ray spectral analysis in *Sherpa*, see the document Spectral Fitting on the *Sherpa* References page.

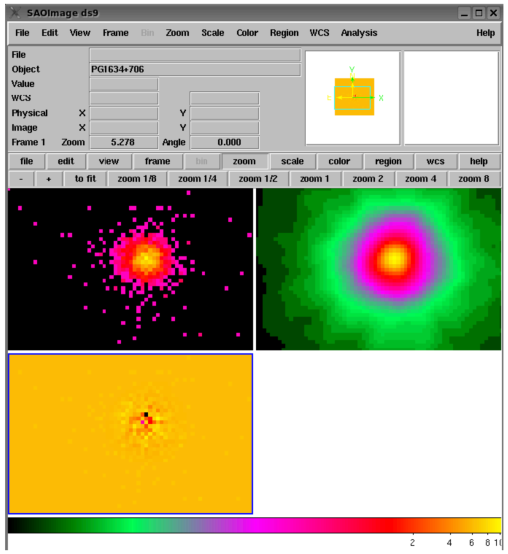
Before fitting ACIS data sets with restricted pulse-height ranges, please read the CIAO caveat page "Spectral analyses of ACIS data with a limited pulse-height range."

Last Update: 9 Dec 2019 - updated for CIAO 4.12, ChIPS figures replaced with matplotlib

Contents

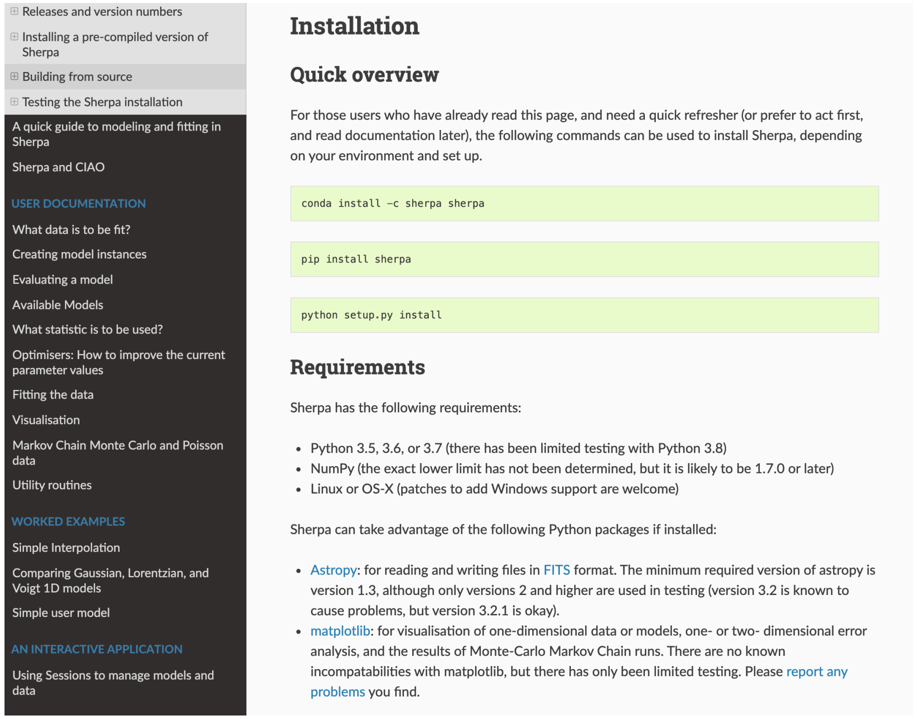
- **Load the Spectrum & Instrument Responses**
- **Filter the Data & Subtract the Background**
- **Defining the Source Model**
- **Fitting**
- **Examining Fit Results**
  - Goodness of fit
  - Confidence intervals
  - Flux and Counts
- **Scripting It**
- **History**
- **Images**




SAOImage ds9
File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
File
Object PG1634+706
Value
WCS
Physical X Y
Image X Y
Frame 1 Zoom 5.278 Angle 0.000
file edit view frame bin zoom scale color region wcs help
- + to fit zoom 1/8 zoom 1/4 zoom 1/2 zoom 1 zoom 2 zoom 4 zoom 8

Installing Sherpa

- Note - installed as part of CIAO with ciao-install or conda install
- Independent Python package:



⊞ 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
Sherpa and CIAO
USER DOCUMENTATION
What data is to be fit?
Creating model instances
Evaluating a model
Available Models
What statistic is to be used?
Optimisers: How to improve the current parameter values
Fitting the data
Visualisation
Markov Chain Monte Carlo and Poisson data
Utility routines
WORKED EXAMPLES
Simple Interpolation
Comparing Gaussian, Lorentzian, and Voigt 1D models
Simple user model
AN INTERACTIVE APPLICATION
Using Sessions to manage models and data

Installation

Quick overview

For those users who have already read this page, and need a quick refresher (or prefer to act first, and read documentation later), the following commands can be used to install Sherpa, depending on your environment and set up.

```
conda install -c sherpa sherpa
```

```
pip install sherpa
```

```
python setup.py install
```

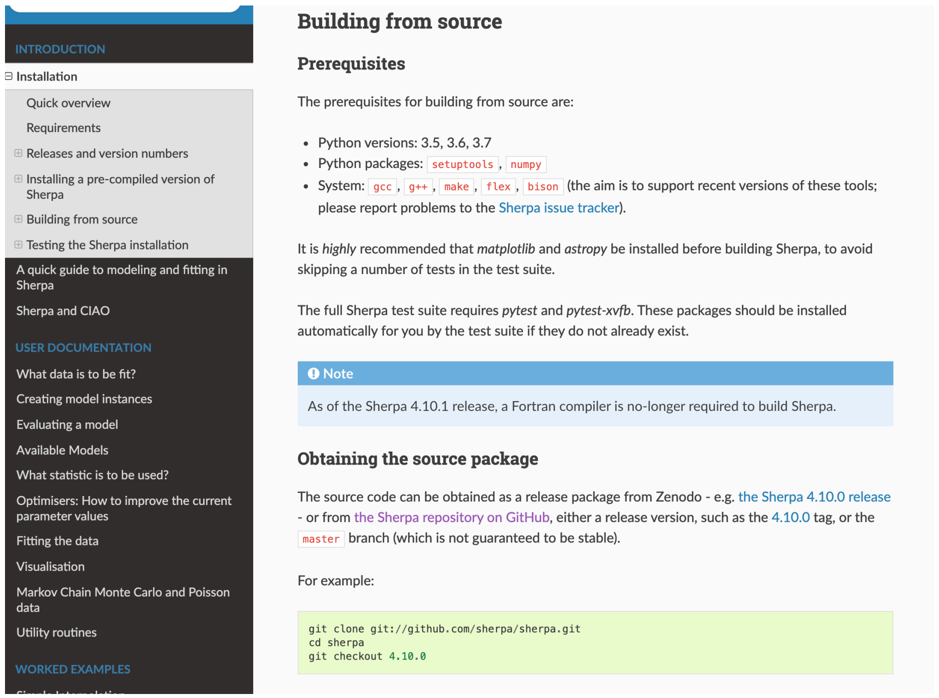
Requirements

Sherpa has the following requirements:

- Python 3.5, 3.6, or 3.7 (there has been limited testing with Python 3.8)
- NumPy (the exact lower limit has not been determined, but it is likely to be 1.7.0 or later)
- Linux or OS-X (patches to add Windows support are welcome)

Sherpa can take advantage of the following Python packages if installed:

- Astropy: for reading and writing files in FITS format. The minimum required version of astropy is version 1.3, although only versions 2 and higher are used in testing (version 3.2 is known to cause problems, but version 3.2.1 is okay).
- matplotlib: for visualisation of one-dimensional data or models, one- or two- dimensional error analysis, and the results of Monte-Carlo Markov Chain runs. There are no known incompatabilities with matplotlib, but there has only been limited testing. Please report any problems you find.



INTRODUCTION
⊟ Installation
Quick overview
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
Sherpa and CIAO
USER DOCUMENTATION
What data is to be fit?
Creating model instances
Evaluating a model
Available Models
What statistic is to be used?
Optimisers: How to improve the current parameter values
Fitting the data
Visualisation
Markov Chain Monte Carlo and Poisson data
Utility routines
WORKED EXAMPLES

Building from source

Prerequisites

The prerequisites for building from source are:

- Python versions: 3.5, 3.6, 3.7
- Python packages: `setuptools`, `numpy`
- System: `gcc`, `g++`, `make`, `flex`, `bison` (the aim is to support recent versions of these tools; please report problems to the Sherpa issue tracker).

It is *highly* recommended that *matplotlib* and *astropy* be installed before building Sherpa, to avoid skipping a number of tests in the test suite.

The full Sherpa test suite requires *pytest* and *pytest-xvfb*. These packages should be installed automatically for you by the test suite if they do not already exist.

Note

As of the Sherpa 4.10.1 release, a Fortran compiler is no-longer required to build Sherpa.

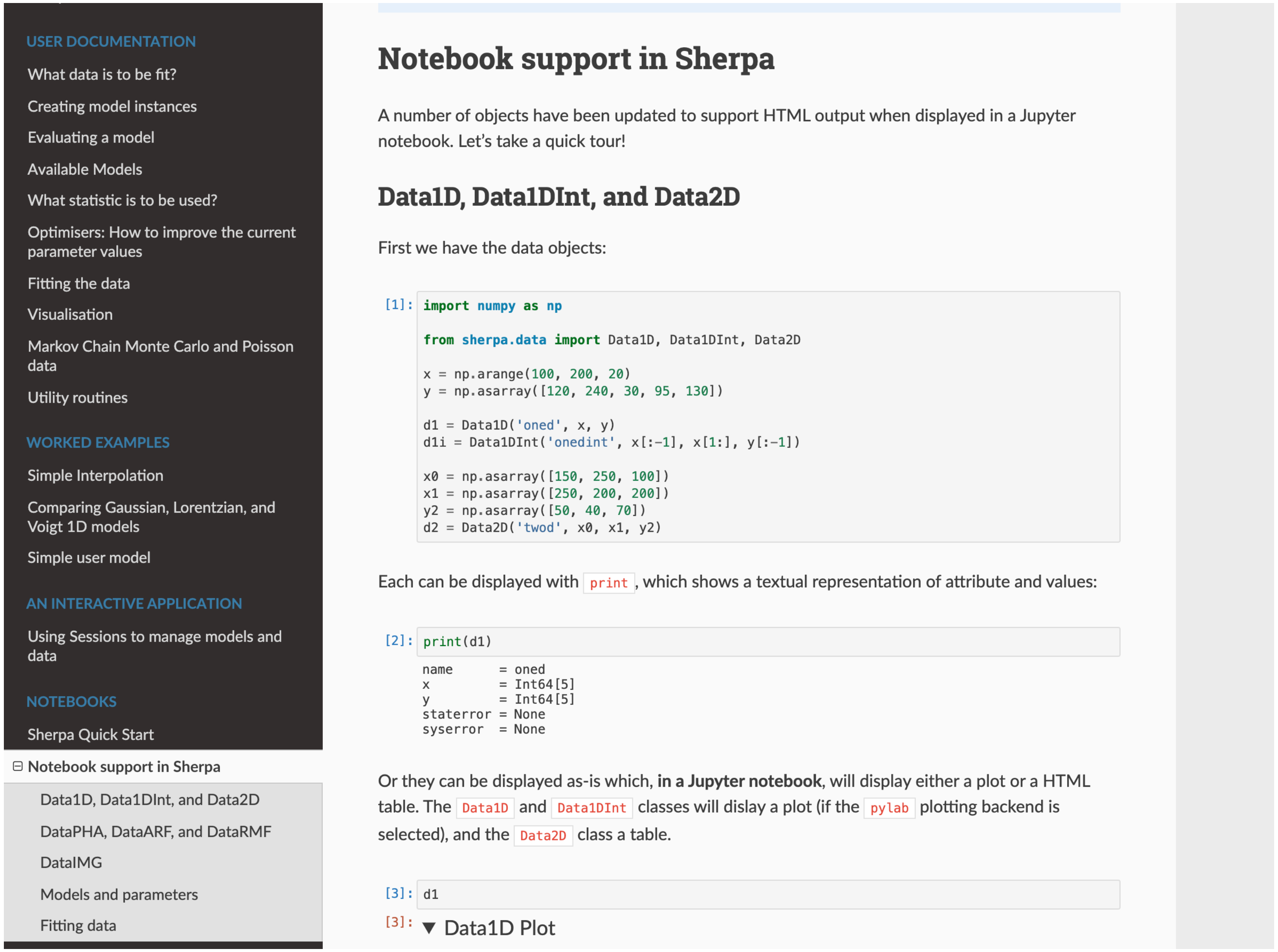
Obtaining the source package

The source code can be obtained as a release package from Zenodo - e.g. the Sherpa 4.10.0 release - or from the Sherpa repository on GitHub, either a release version, such as the 4.10.0 tag, or the `master` branch (which is not guaranteed to be stable).

For example:

```
git clone git://github.com/sherpa/sherpa.git
cd sherpa
git checkout 4.10.0
```

Sherpa in Python



USER DOCUMENTATION

- What data is to be fit?
- Creating model instances
- Evaluating a model
- Available Models
- What statistic is to be used?
- Optimisers: How to improve the current parameter values
- Fitting the data
- Visualisation
- Markov Chain Monte Carlo and Poisson data
- Utility routines

WORKED EXAMPLES

- Simple Interpolation
- Comparing Gaussian, Lorentzian, and Voigt 1D models
- Simple user model

AN INTERACTIVE APPLICATION

- Using Sessions to manage models and data

NOTEBOOKS

- Sherpa Quick Start
- ⊟ Notebook support in Sherpa
  - Data1D, Data1DInt, and Data2D
  - DataPHA, DataARF, and DataRMF
  - DataIMG
  - Models and parameters
  - Fitting data

Notebook support in Sherpa

A number of objects have been updated to support HTML output when displayed in a Jupyter notebook. Let's take a quick tour!

Data1D, Data1DInt, and Data2D

First we have the data objects:

```
[1]: import numpy as np

     from sherpa.data import Data1D, Data1DInt, Data2D

     x = np.arange(100, 200, 20)
     y = np.asarray([120, 240, 30, 95, 130])

     d1 = Data1D('oned', x, y)
     d1i = Data1DInt('onedint', x[:-1], x[1:], y[:-1])

     x0 = np.asarray([150, 250, 100])
     x1 = np.asarray([250, 200, 200])
     y2 = np.asarray([50, 40, 70])
     d2 = Data2D('twod', x0, x1, y2)
```

Each can be displayed with `print`, which shows a textual representation of attribute and values:

```
[2]: print(d1)

     name      = oned
     x         = Int64[5]
     y         = Int64[5]
     staterror = None
     syserror  = None
```

Or they can be displayed as-is which, **in a Jupyter notebook**, will display either a plot or a HTML table. The `Data1D` and `Data1DInt` classes will dislay a plot (if the `pylab` plotting backend is selected), and the `Data2D` class a table.

```
[3]: d1
[3]: ▼ Data1D Plot
```

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
- Open development with continuous integration via Travis

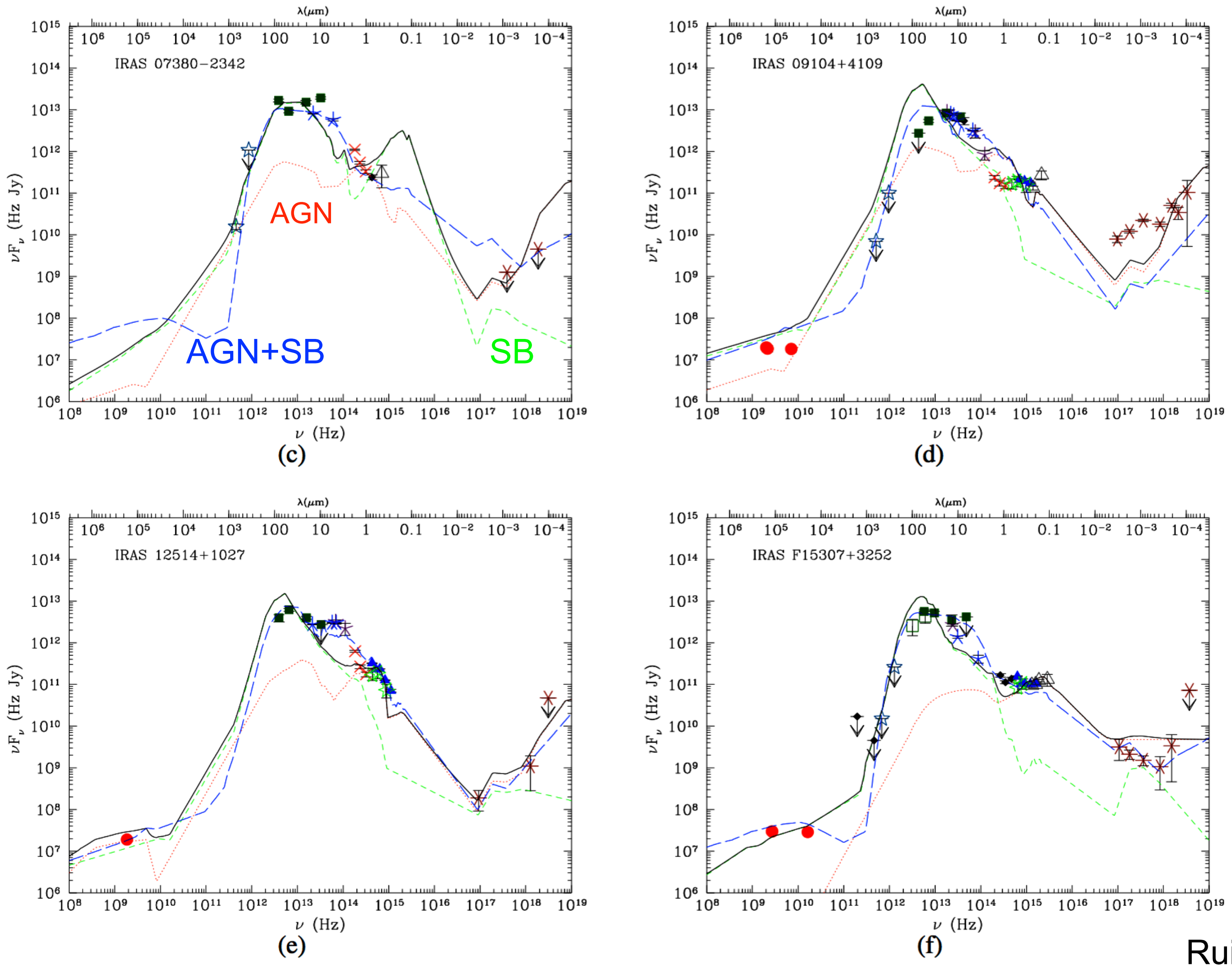
Using Sherpa in Astronomy Software

- 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 Affiliated packages:
  - GammaPy https://gammapy.readthedocs.io/en/latest/
  - Naima https://naima.readthedocs.io/en/latest
- 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.

Science Results

- Read scientific papers:
    - concentrate on understanding analysis and statistics applied to the data.
- Present scientific papers:
    - consider reproducibility of the results.
    - Focus on a description of the data analysis and statistical methodology understandable to the other scientists.

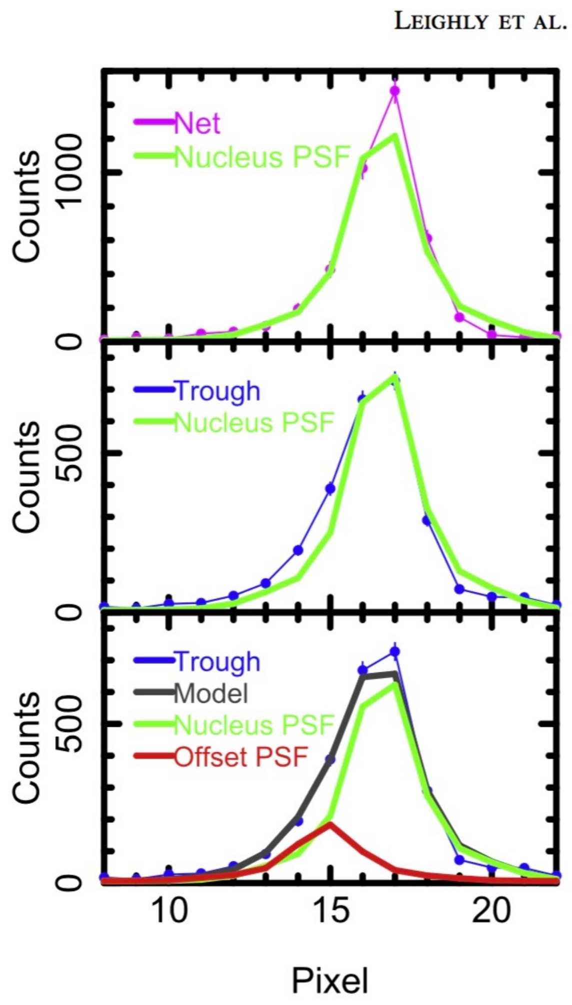
Spectral (SED) Fitting with Composite Templates

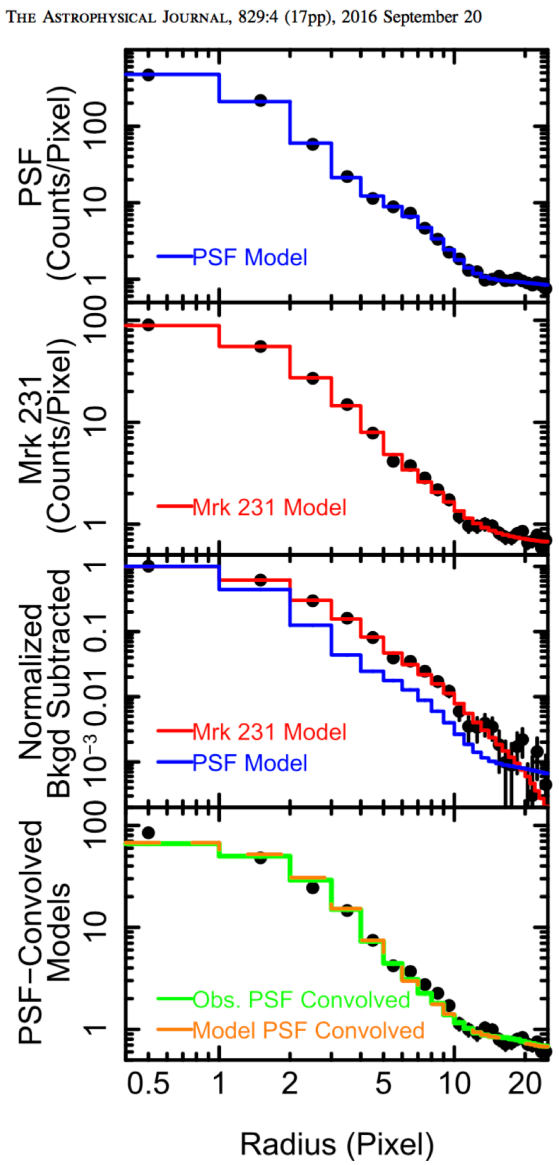


Ruiz et al. (2010)

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

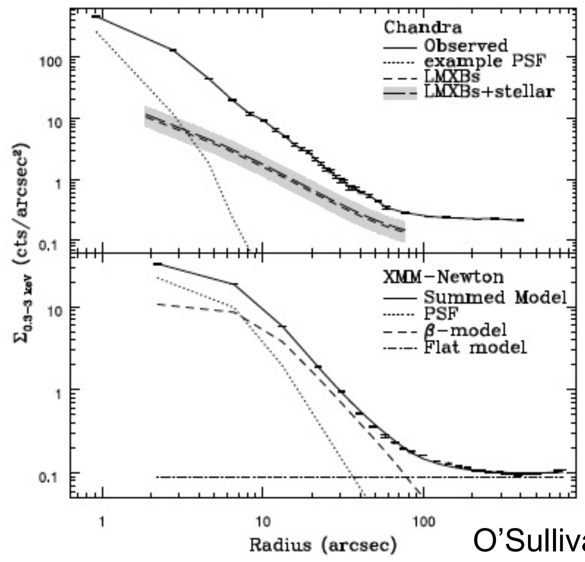




Leighly et al. (2016)

Surface Brightness Profiles
(with & without PSF)

Chandra and XMM



O'Sullivan et al. (2011)

HST Images

Radio loudness and surface brightness profile 2167

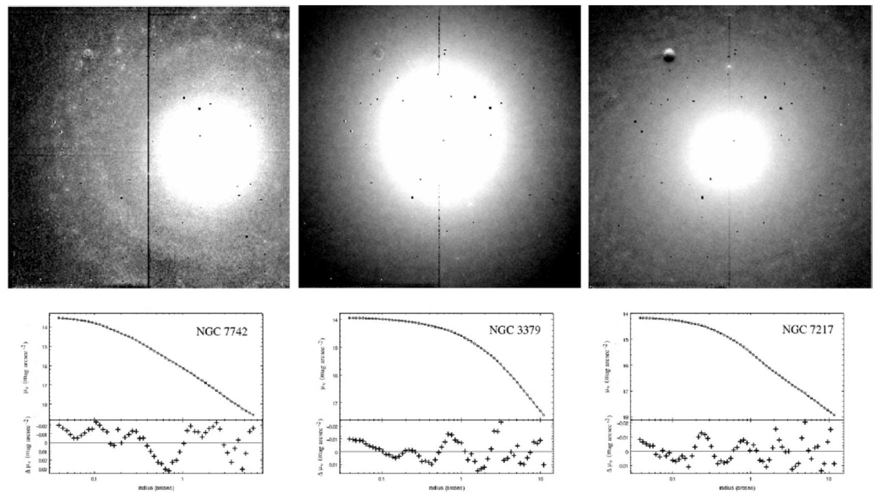
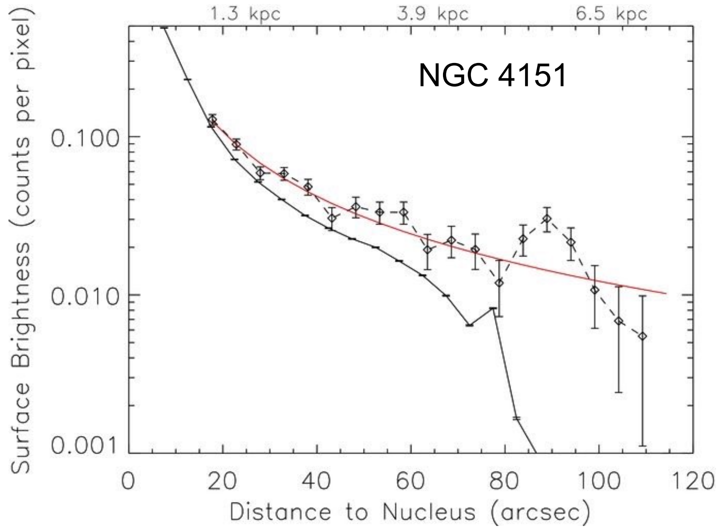


Figure 1. Galaxy images (top row) and radial brightness profiles (bottom row) for a confident Sérsic fit (NGC 7742; left), Core fit (NGC 3379; centre) and Double-Sérsic fit (NGC 7217; right).

Richings, Utley & Kording (2011)

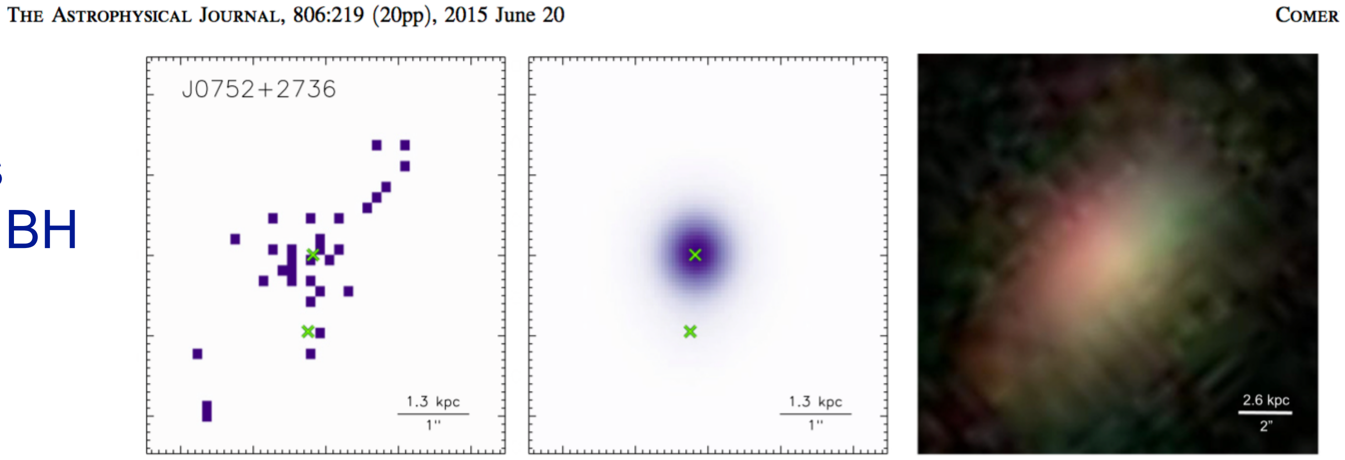
Chandra



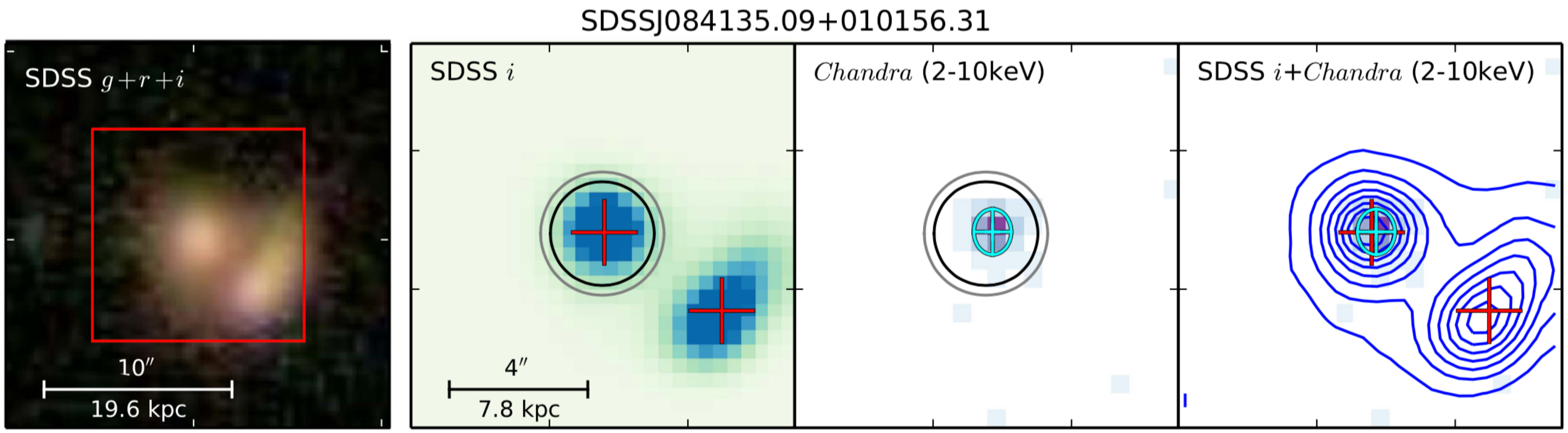
Wang et al. (2010)

Image Analysis

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

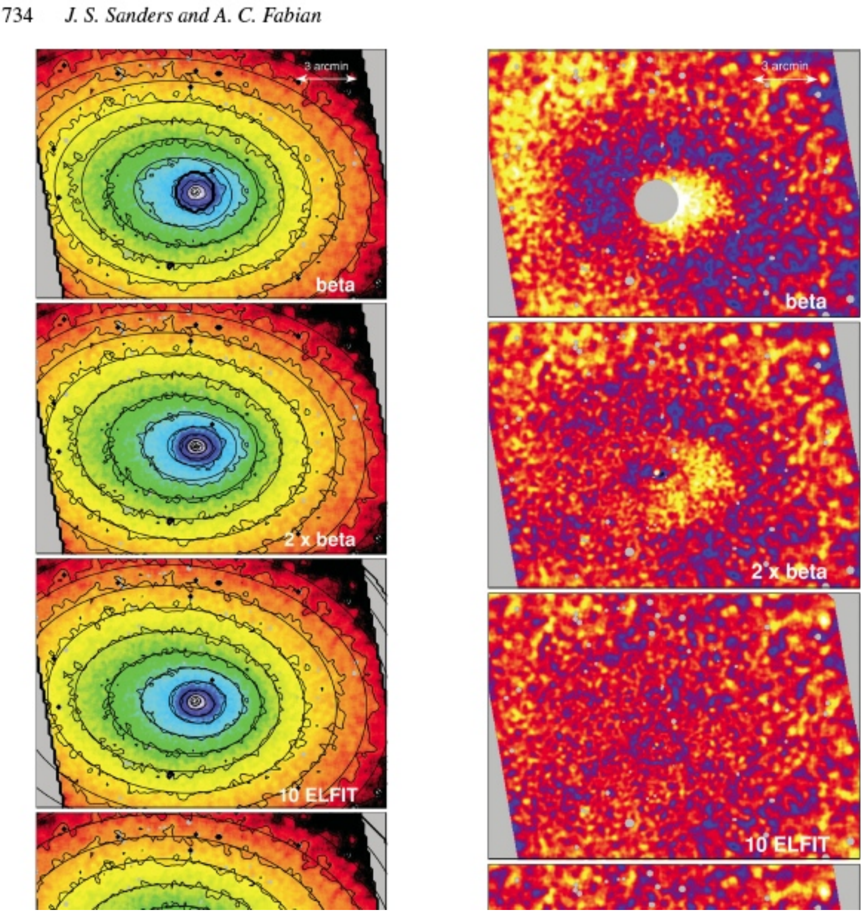


Comerford et al. (2015)

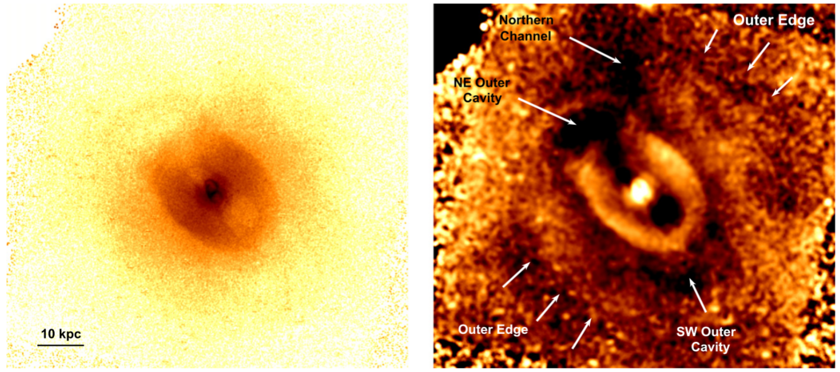


Barrows et al. (2016)

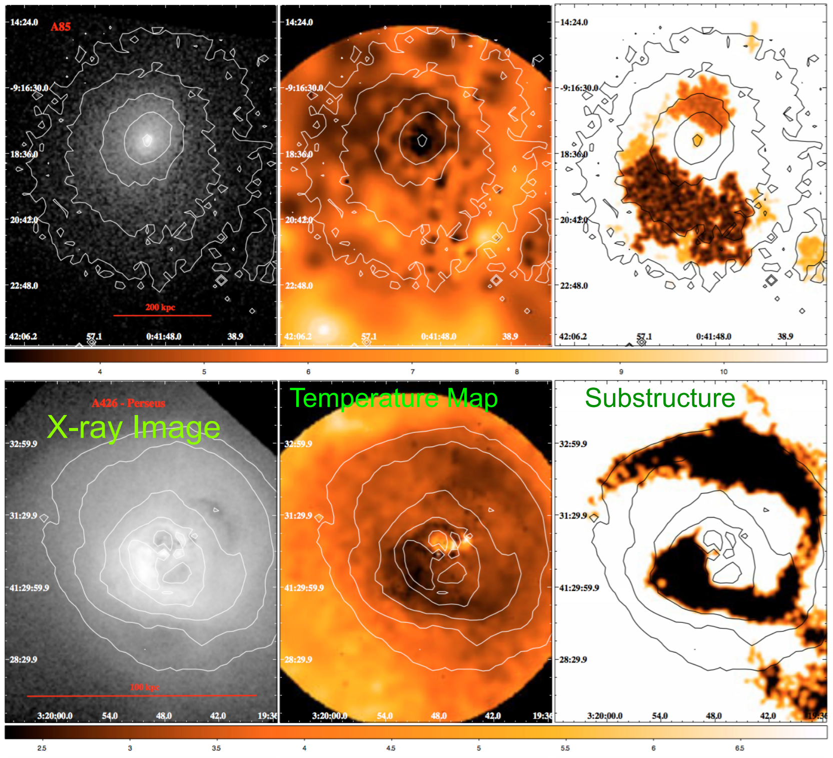
Identifying Substructures in X-ray Clusters


734 J. S. Sanders and A. C. Fabian

Sanders & Fabian (2012)



Randall et al. (2015)



Lagana, Santos & Lima Neto (2010)

Composite Models in BXA Bayesian X-ray Analysis

Buchner et al.: Absorption and reflection model comparison of AGN in the CDFS

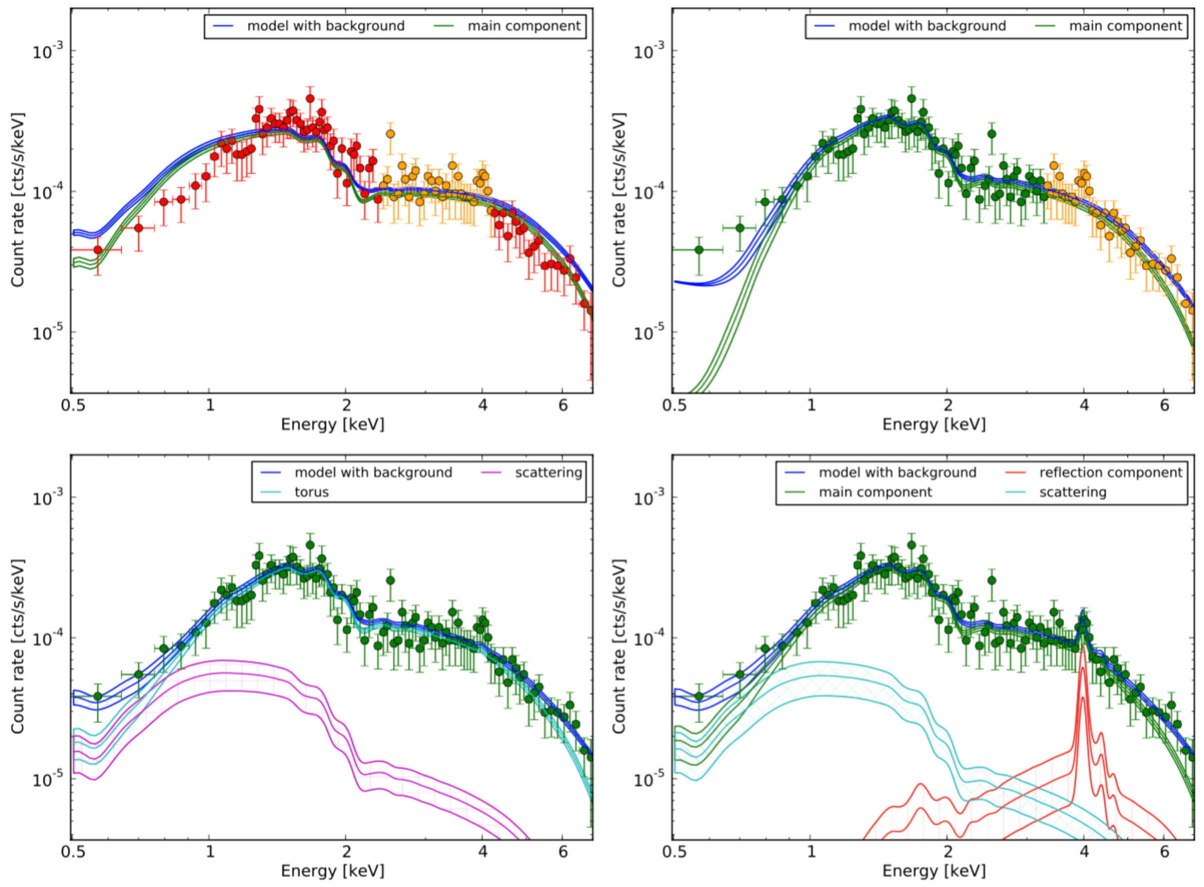


Figure 5: Observed (convolved) spectrum of object 179, binned for plotting to 10 counts per bin. Shown are analyses using various models and their individual components: **powerlaw** (upper left), **wabs** (upper right), **torus+scattering** (lower left) and **wabs+pexmon+scattering** (lower right).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

Chandra Deep Field South X-ray Spectrum of an object fit with different composite models

Buchner et al. 2014

Spatial Fitting of the TeV emission in H.E.S.S. observations

A&A 541, A5 (2012)

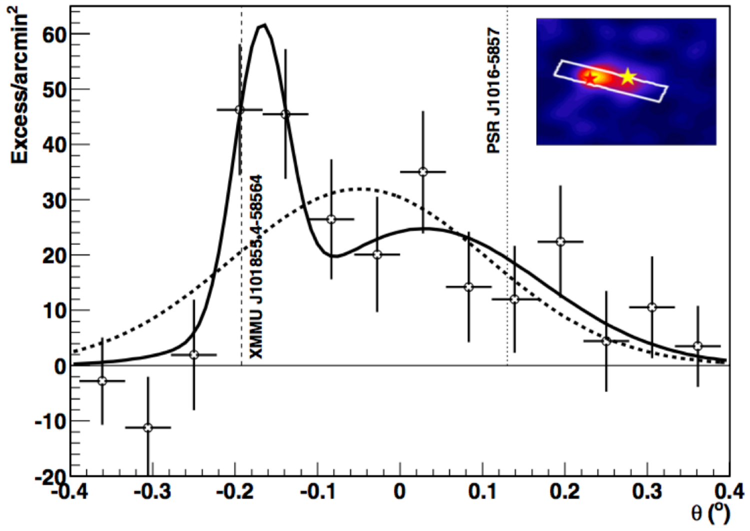


Fig. 3. Profile of the VHE emission along the line between the peak of the point-like emission and the peak of the diffuse emission, as illustrated in the inset. Fits using a single and a double Gaussian function are shown in dashed and solid lines respectively. The positions of XMMU J101855.4–58564 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.

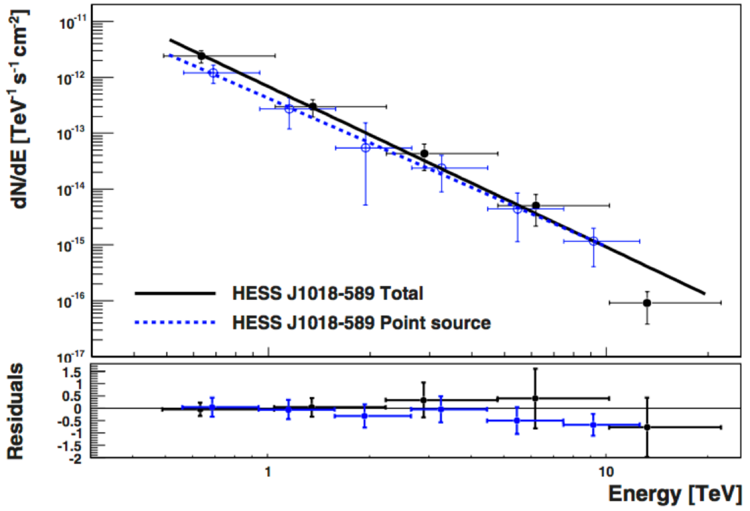


Fig. 4. VHE photon spectrum of HESS J1018–589 for a point-like source at position A (in blue dots and dashed blue line) and derived from a region of size 0.30° comprising the point-like and diffuse emission (in black dots and solid black line). The residuals to the fit are shown in the bottom panel.

Abramowski et al. (2012)