Sherpa
Modeling and Fitting

Aneta Siemiginowska

Chandra X-ray Center

http://cxc.harvard.edu/sherpa

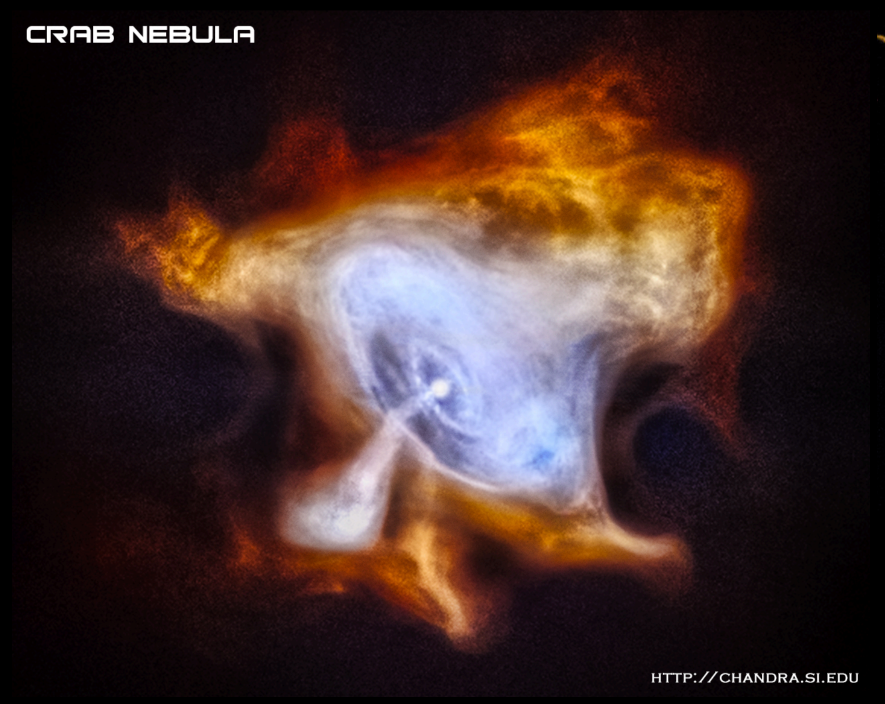
Chandra/CIAO Workshop at UMass Lowell   May 2025

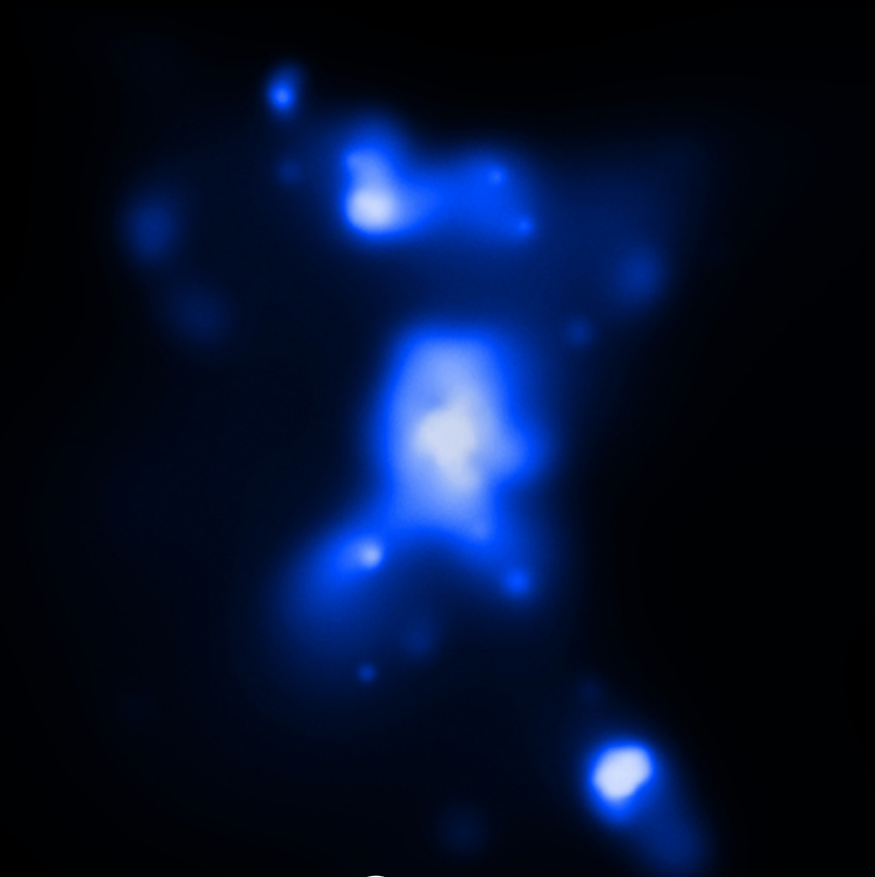
X-ray Universe

Solar System

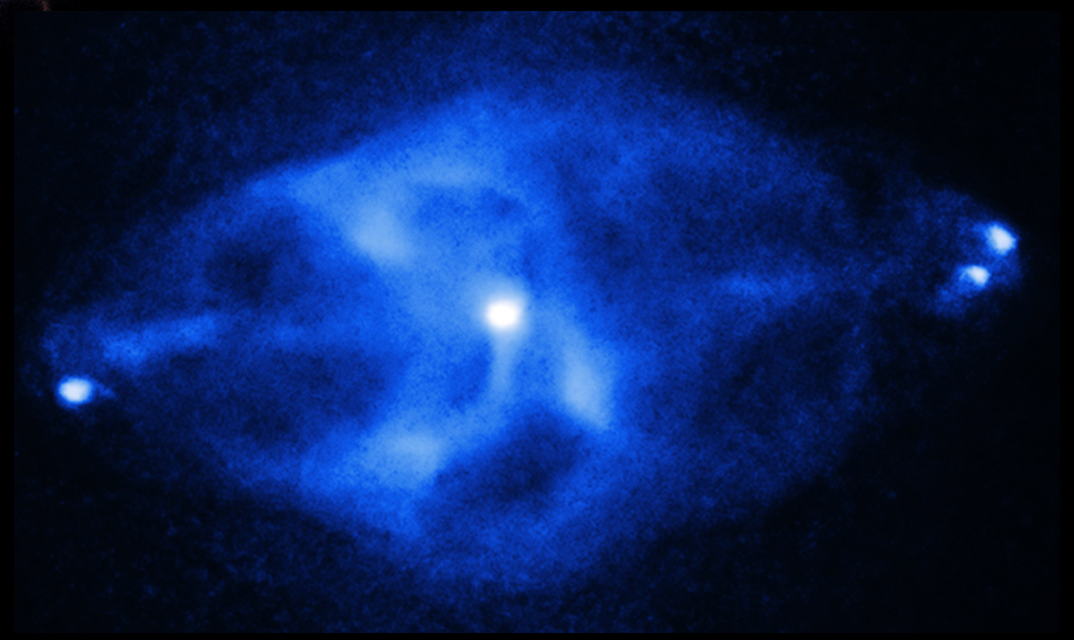
Hot gas > 10^5 K
Energetic particles

Supernova Remnants

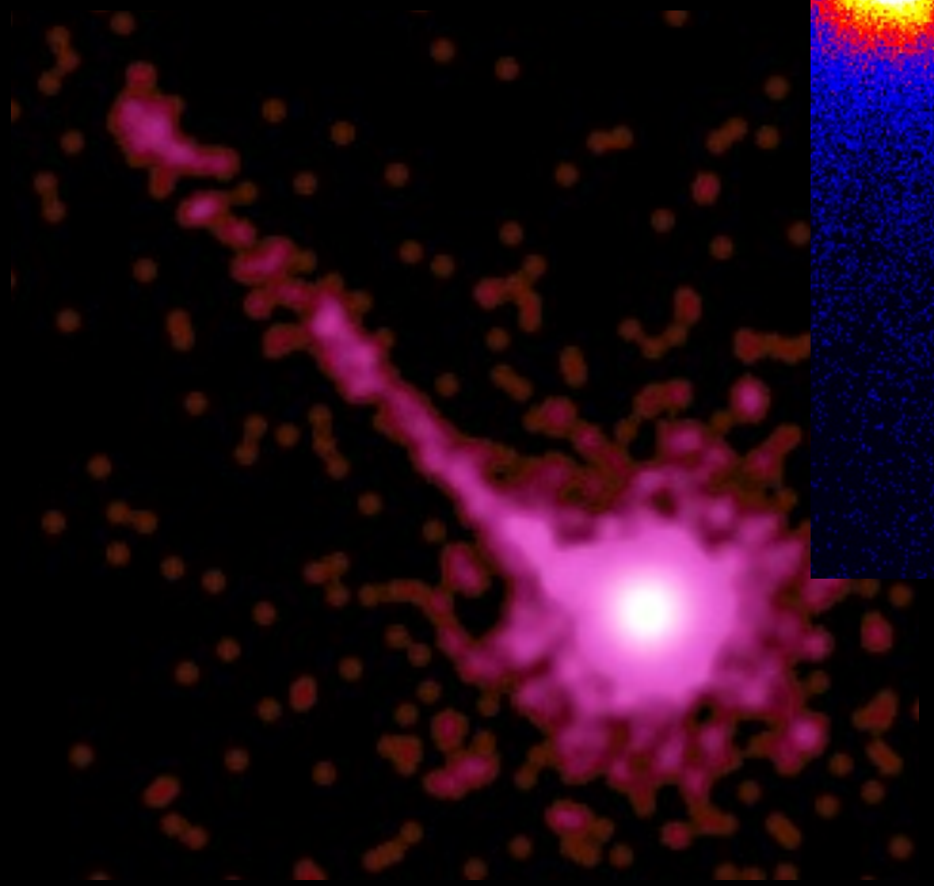


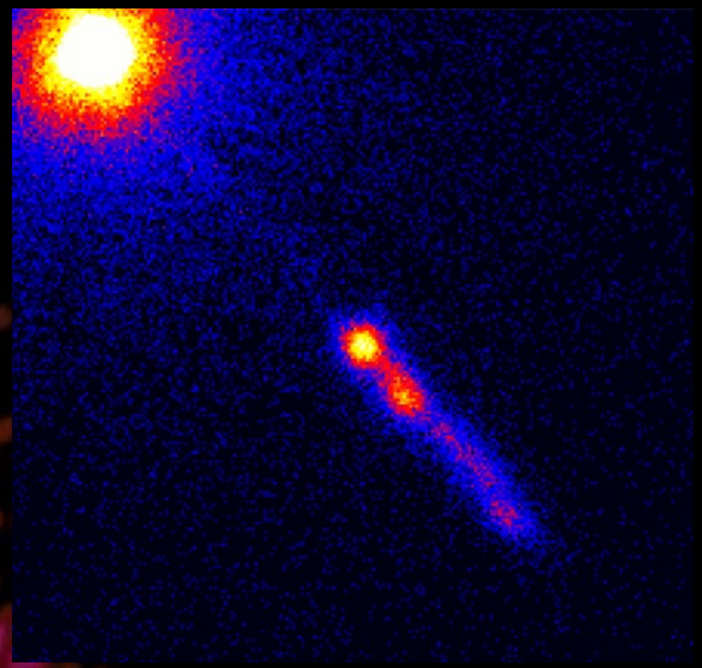


Radio Galaxies

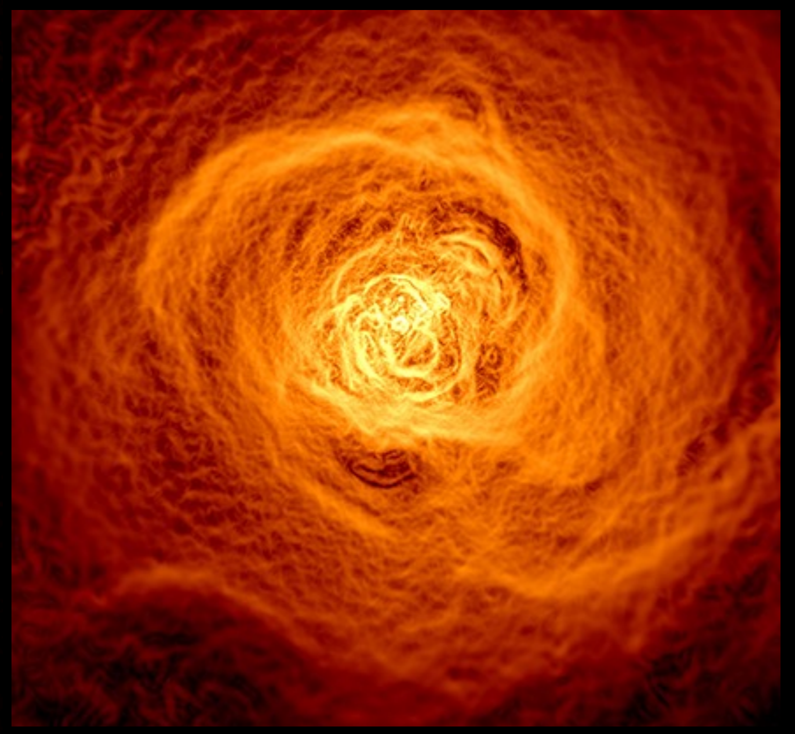


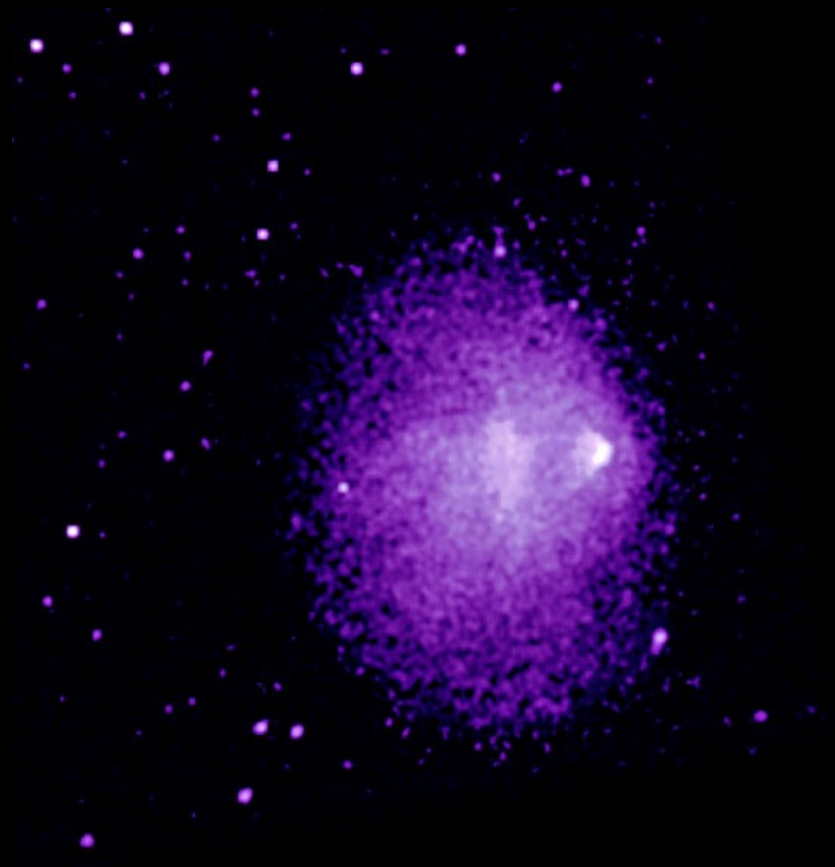
Quasar Jets





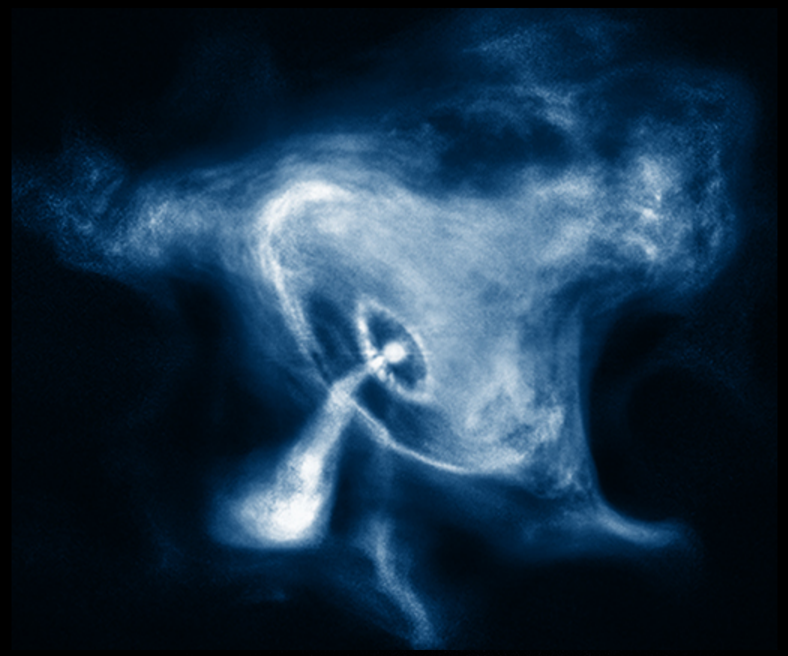
Clusters of Galaxies





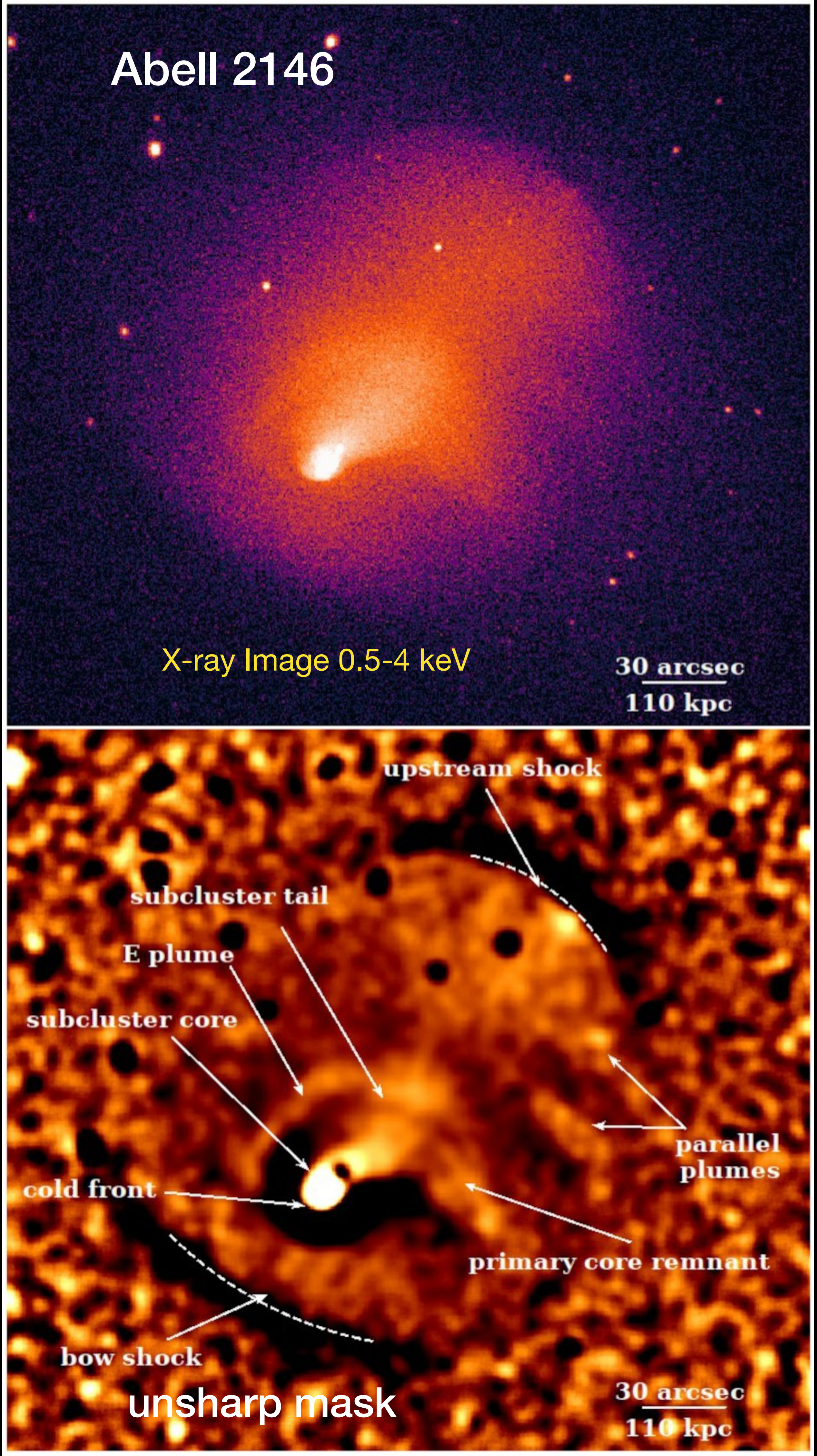
X-ray Images obtained with the Chandra X-ray Observatory
(False Color)

Scientific Measurements

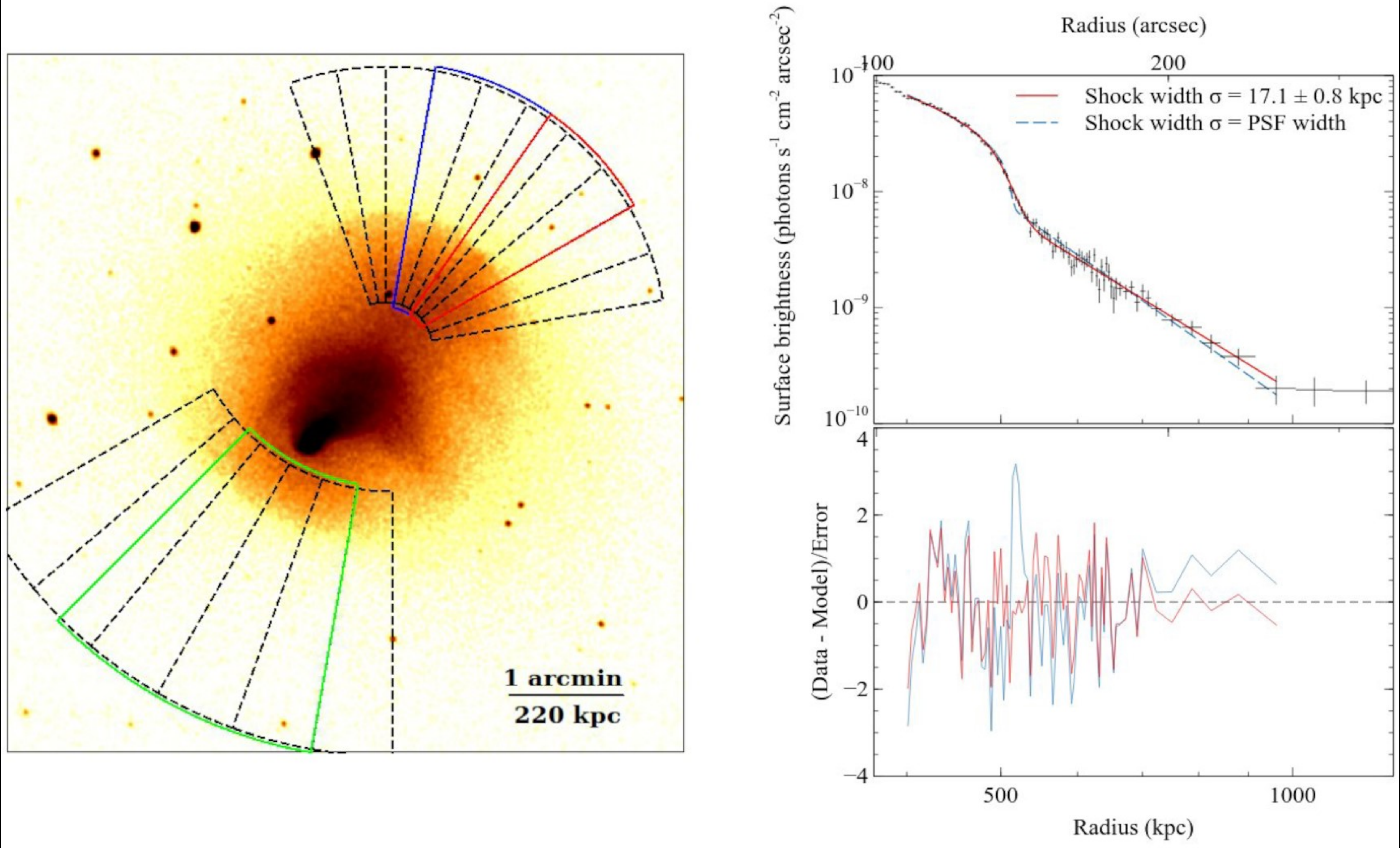


Measurements	Examples	Methods
Morphology	point source, diffuse structures, filaments	detection algorithm, smoothing, unsharp mask, deconvolution, model fitting
Scale and Size	emission features, boundary, clusters, unresolved structures, mass	surface brightness profiles, extent, deconvolution, variability timescales, correlation between different bands
Source Properties	flux, luminosity, temperature, abundance, density, obscuration, age	model fit, aperture photometry
Population	members, intensity, identification, flux distributions	detection algorithms, hardness ratios, catalog matches, spectral modeling

Scientific Measurements



Russell et al 2022



Morphology
Scale and Size
Source Properties
Populations

NOTE on some Data Issues and Source of Uncertainties:

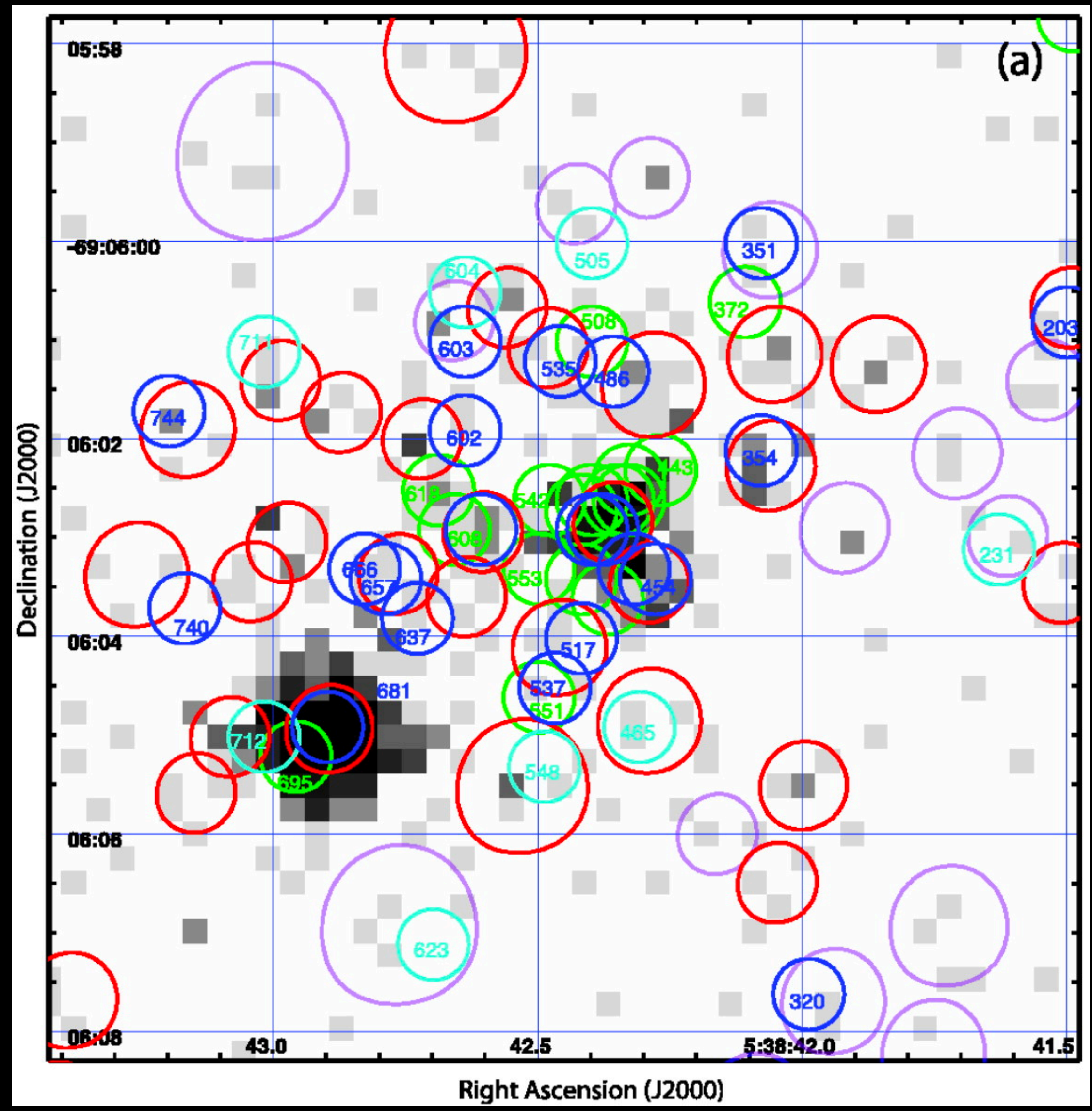
- combined multiple observations
- background level
- region selection
- model fit to the surface brightness profile
- PSF (blurring) impact on the measurements

Scientific Measurements

Star Cluster

Chandra X-ray image
Red: 0.5-2 keV
Blue: 2-7 keV

Townsley et al 2006



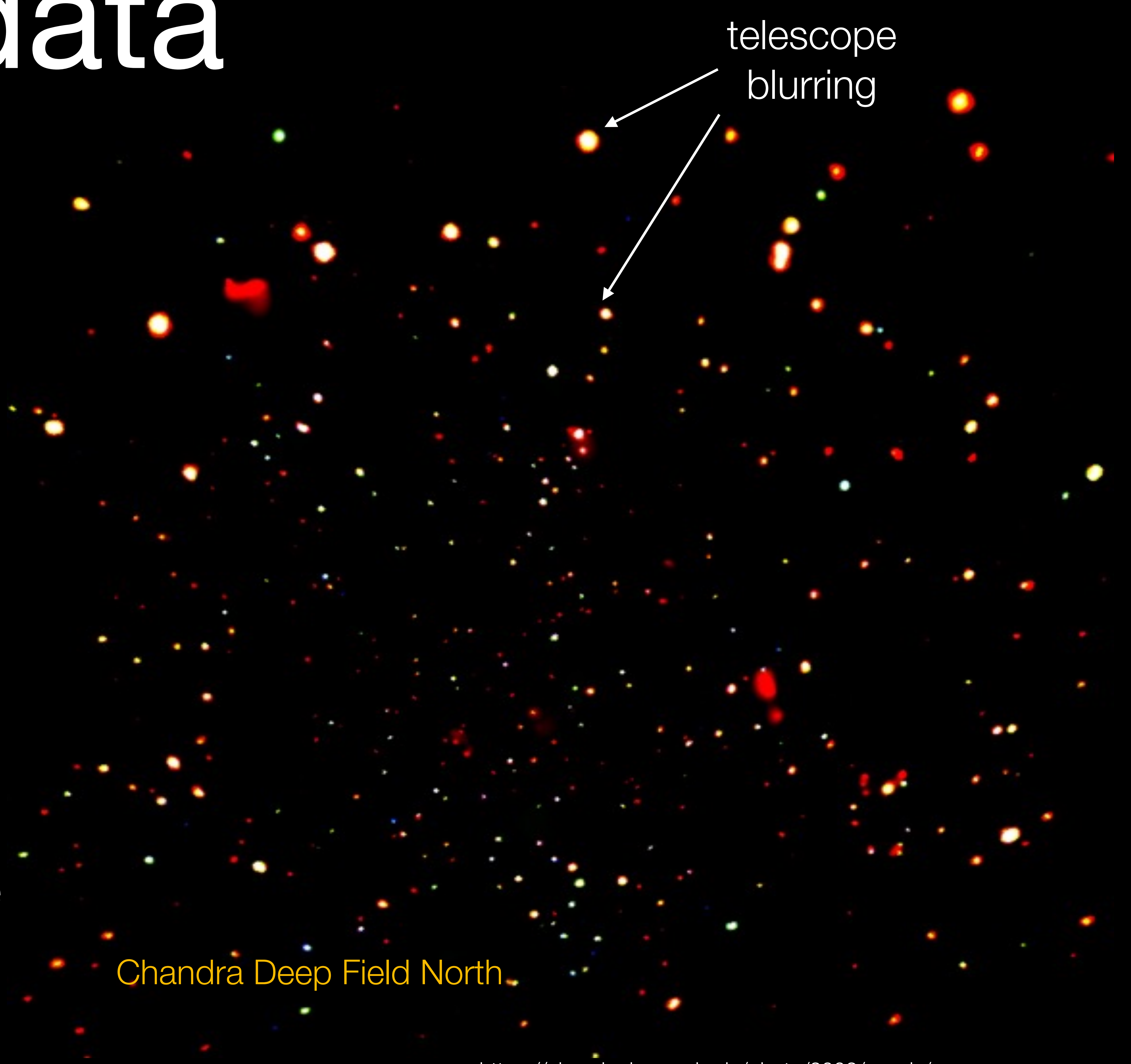
Morphology
Scale and Size
Source Properties
Populations

NOTE on some Data Issues and Source of Uncertainties:

- combined multiple observations
- background level
- region selection
  - PSF (blurring) impact on source detection
  - overlapping PSFs for source counts measurements

X-ray data

- Counting arriving photons (Poisson counts) - different from optical data
- For each photon location on the sky (x,y), arrival time (t) and energy (E) are recorded (x,y,t,E) - events
- X-ray observations take a long time - a short observation with Chandra X-ray Observatory lasts ~10 ksec (~3 hours) while typical observations take a day or more. The Chandra Deep Field observations took about **23 days**.



https://chandra.harvard.edu/photo/2003/goods/

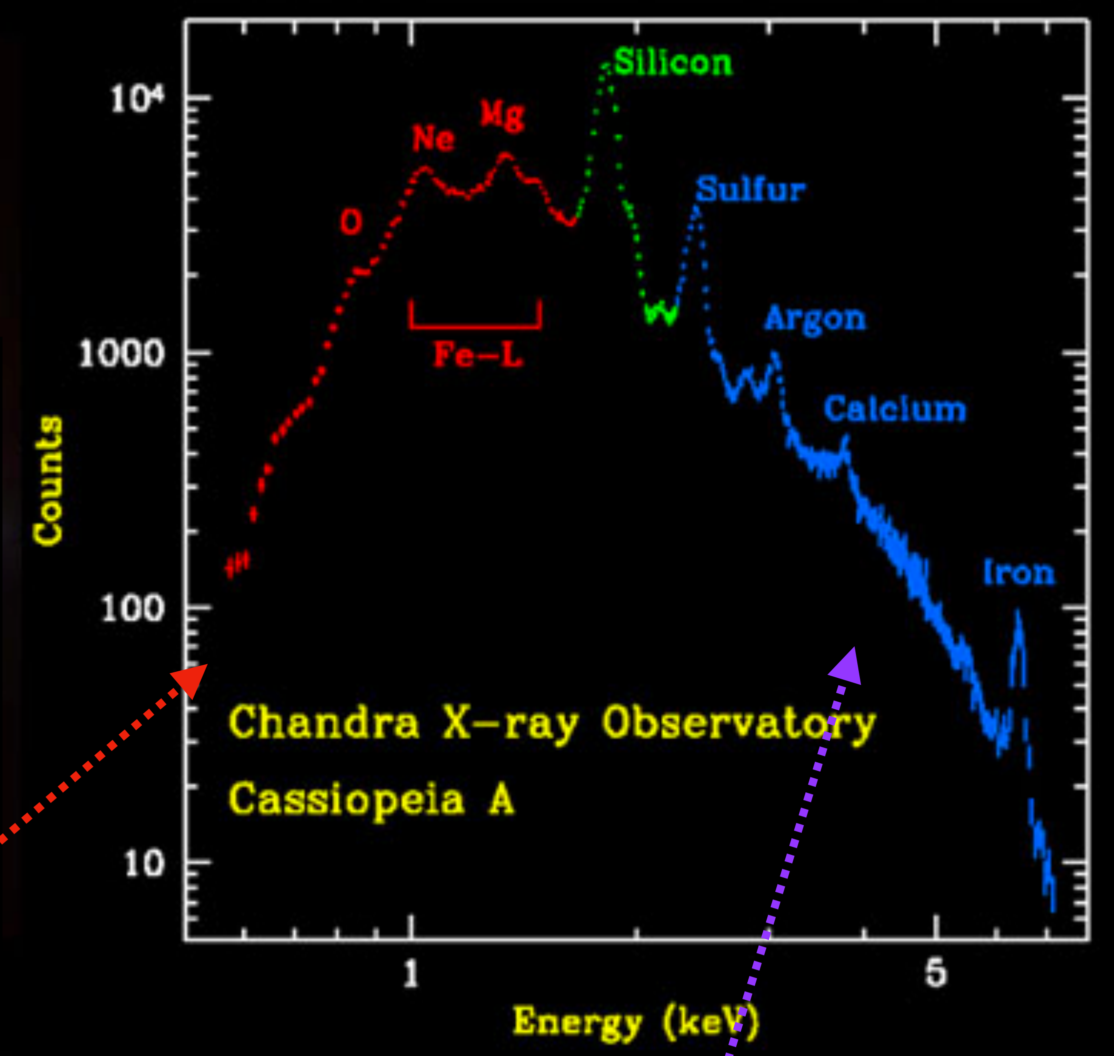
The faintest sources - one X-ray photon every 4 days!

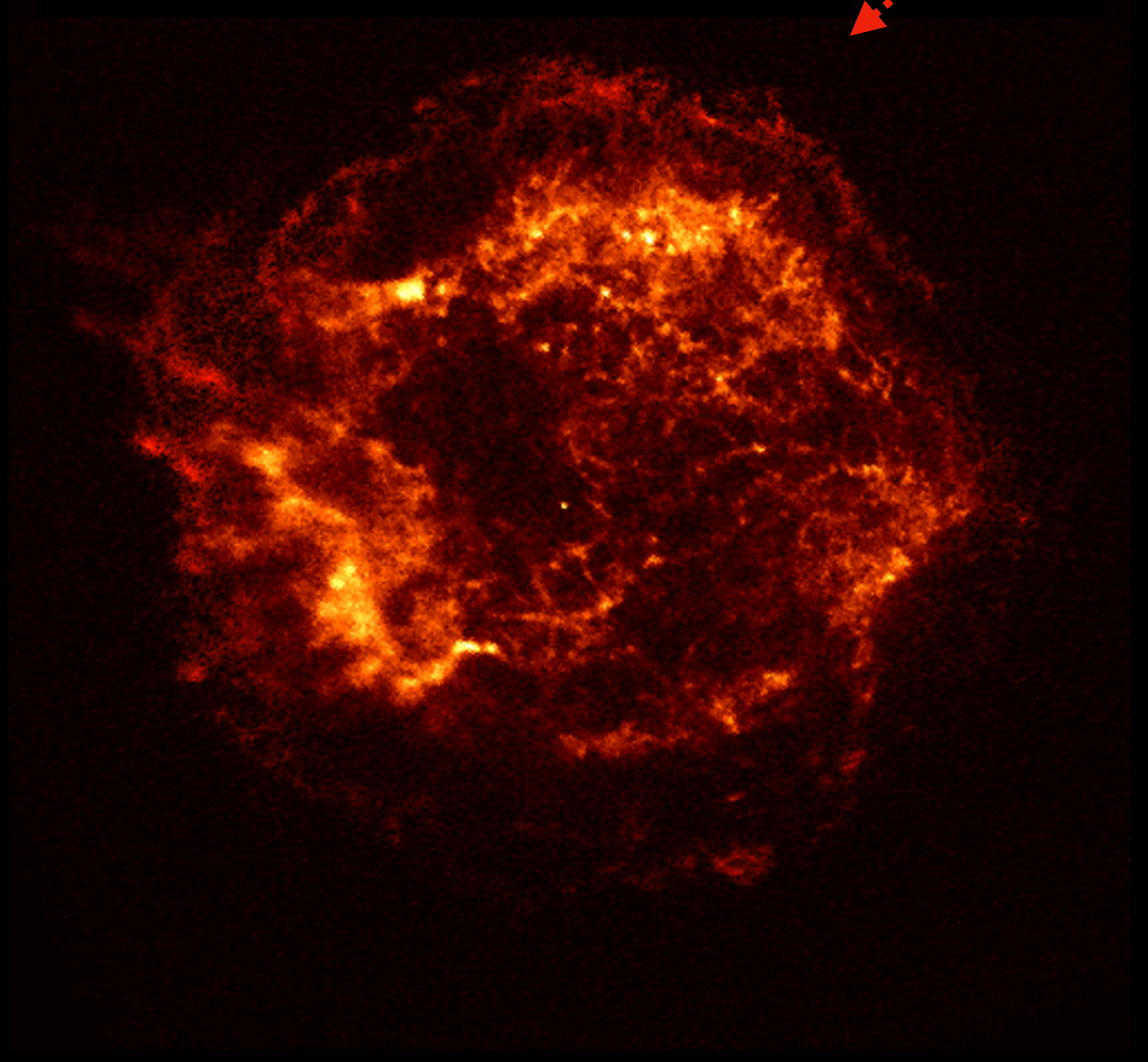
X-ray Analysis Data Domains

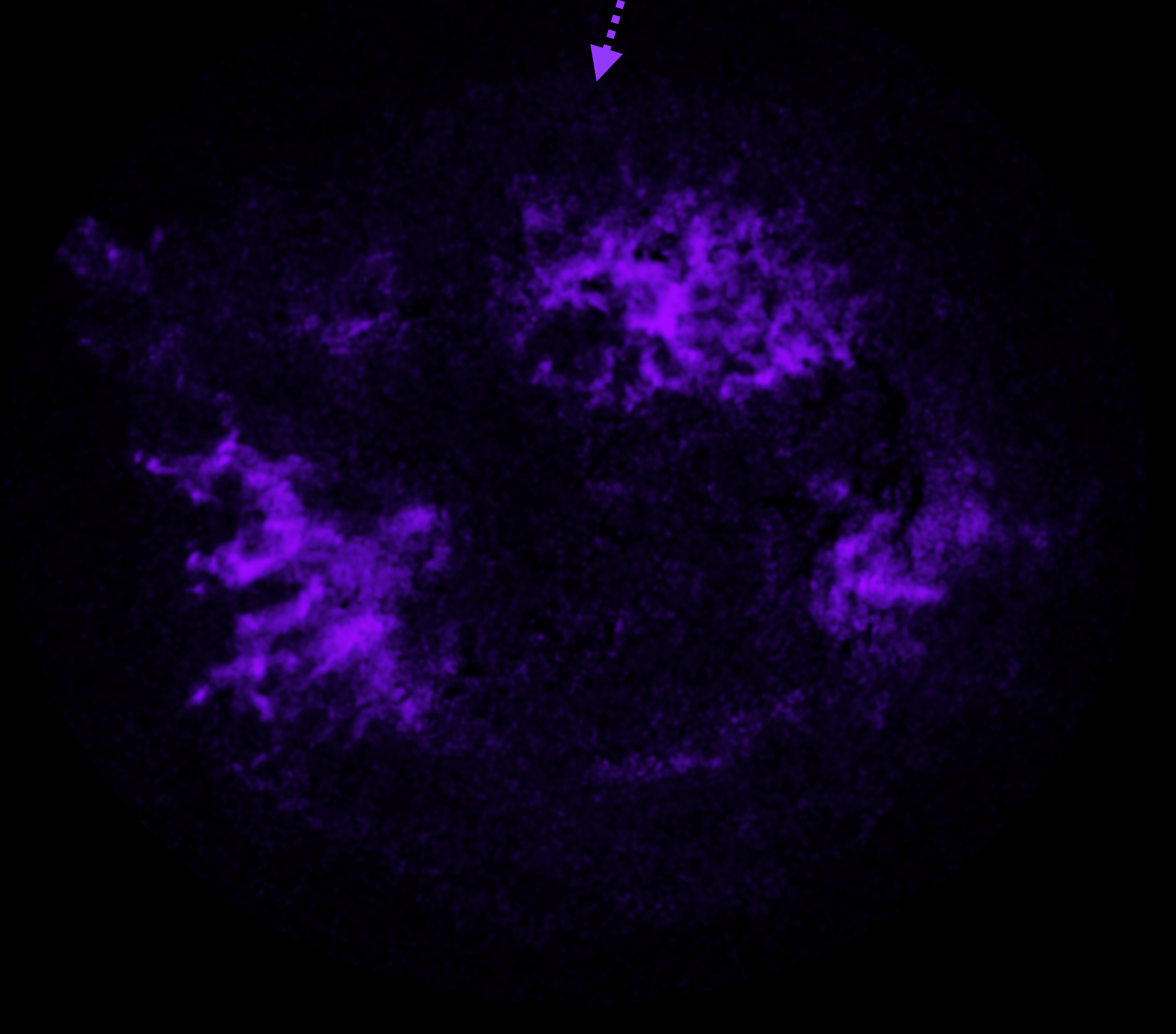
Event $e_i = (x_i, y_i, t_i, E_i)$

- **X-ray image** is made by binning events into images, e.g. accumulating photons in a selected energy band and fixed exposure time: $e_i(x, y) = \int e(x, y, t, E)\, dE\, dt$
  - *no spectral or temporal information*
  - *analysis require a point spread function*
- **Energy Spectrum** for selected regions are generated by binning the events in energy: $e_i(E) = \int e(x, y, t, E)\, d(x, y)\, dt$
  - *no spatial or temporal information*
  - *require additional calibration files*
- **Lightcurve - time series** for selected region and energy band binning the events in time: $e_i(t) = \int e(x, y, t, E) d(x, y) d(E)$
  - *no spatial or energy information*

Cassiopeia A Supernova Remnant







Single Domain Analysis

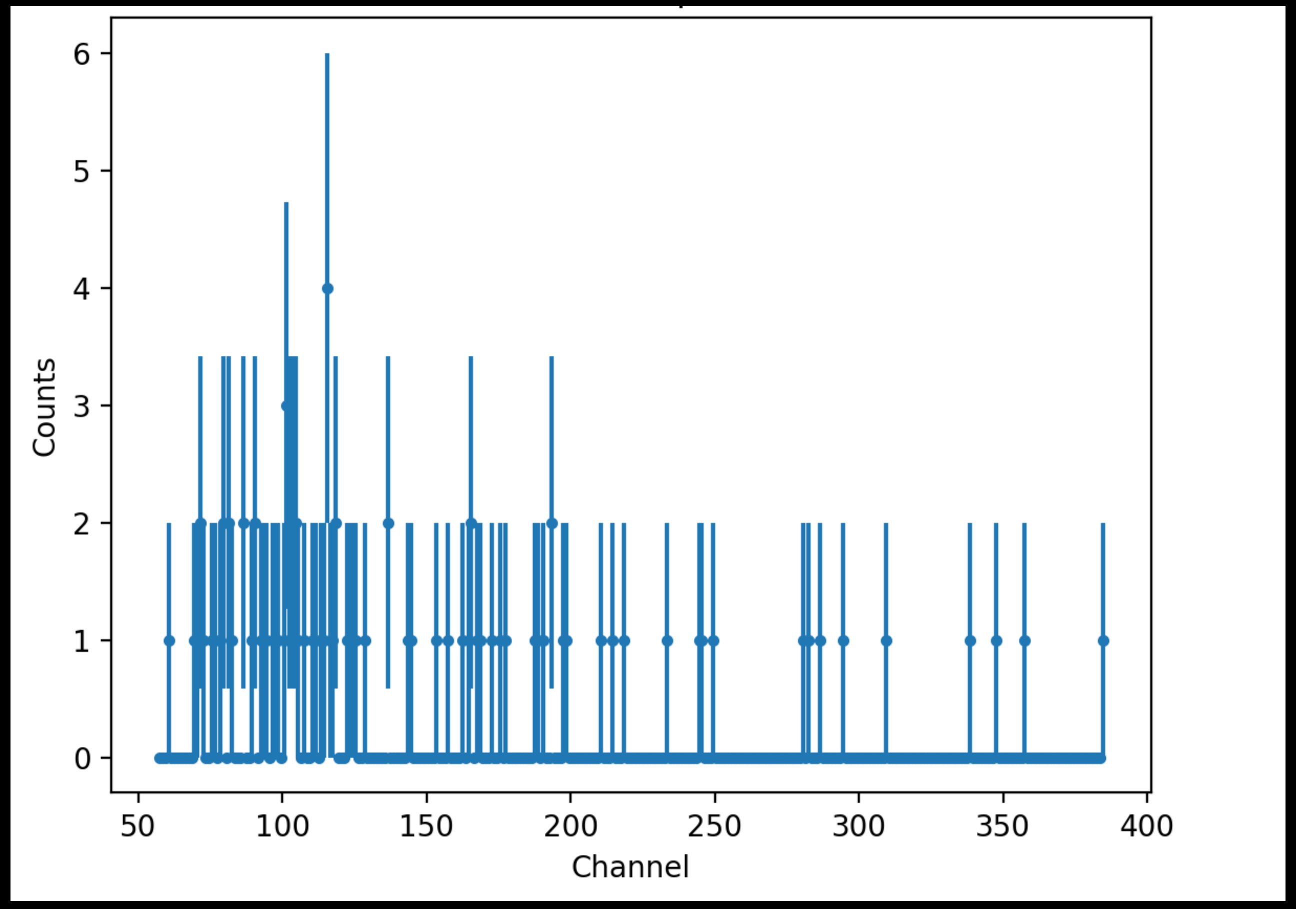
Analysis Domain	Description	Standard Methods
Spectra $e(E) = \int e(x, y, t, E)dx\,dy\,dt$	only energy, loss of time and morphology	forward fitting, multi-spectra, Poisson likelihood, model and instrument uncertainties
Image $e(x, y) = \int e(x, y, t, E)dE\,dt$	only location and morphology, loss of energy and time	source detections, morphology, contours, image reconstruction, deconvolution
Time variability $e(t) = \int e(x, y, t, E)dx\,dy\,dE$	only time, loss of energy and source morphology	differences image/spectra, power spectra, periodogram, Bayesian Blocks, flares

X-ray Energy Spectra

- Model fitting:
    - Includes instrument response directly -> calibration impact on the results

$$\text{Counts}\,(i) = \int R(i,E)\,A(E)\;M(E)\,dE$$

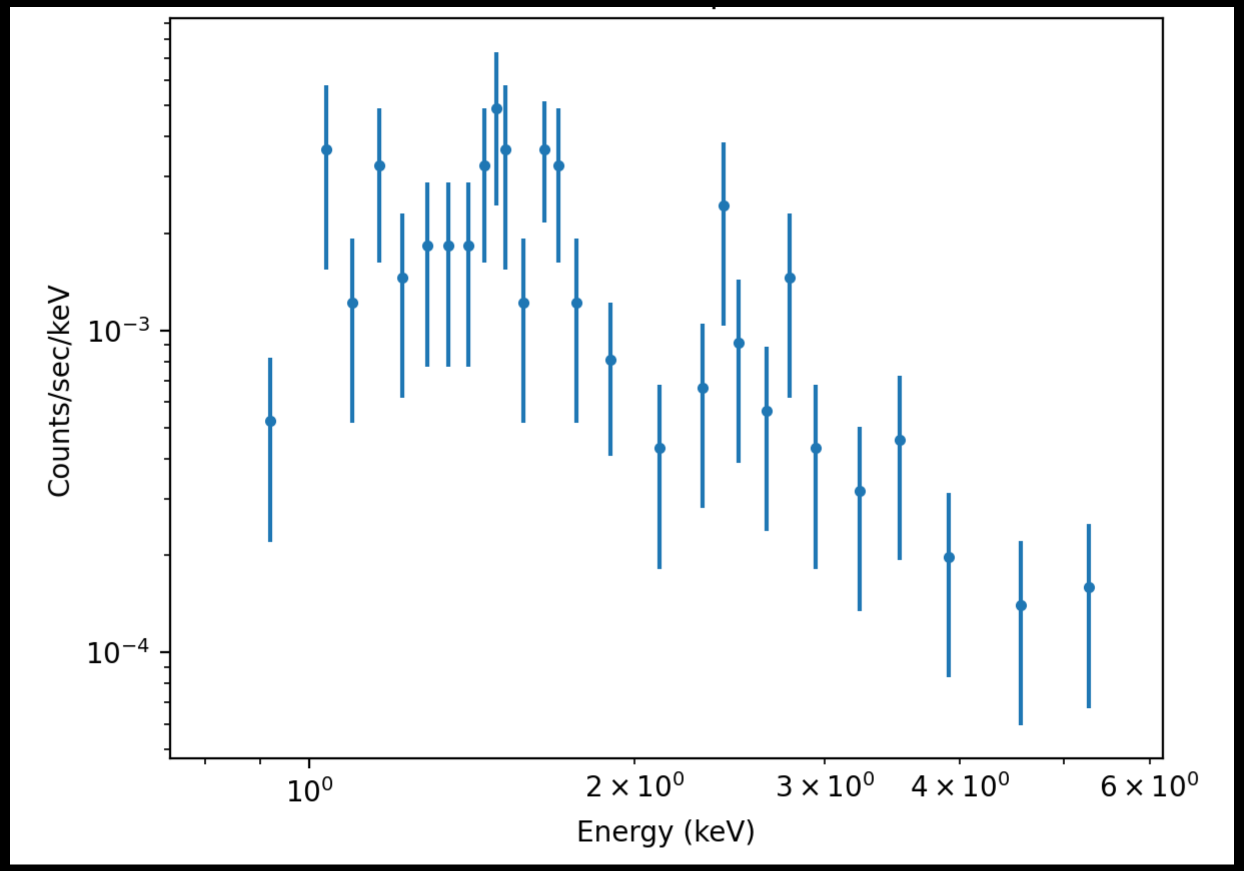
    - Non-linear astrophysical models, computer generated models
    - Appropriate fit statistics, no binning/grouping data, no background subtraction
    - Formulations for the Poisson likelihood



channel

keV

convert matrix (RMF)



log(Energy) [keV]

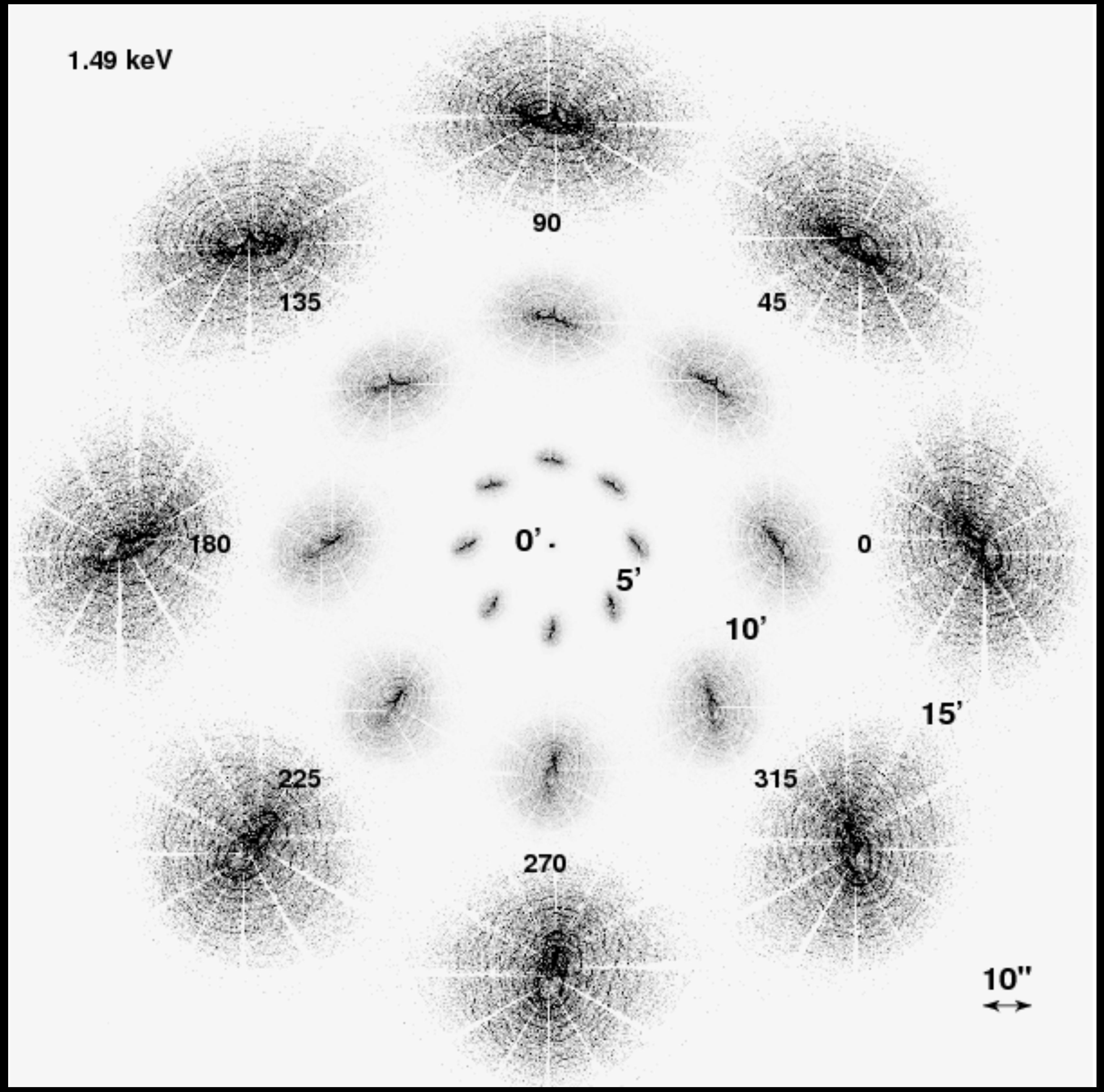
X-ray Images

- Chandra X-ray Observatory takes the highest angular resolution X-ray images of the Universe
- Poisson counts - sparse images with many empty pixels
- PSF variable across the images cannot be described in an analytical form, the PSF image is a simulation from the computer model of the Chandra mirrors with calibration measurements

$$e(x, y) = \int e(x, y, t, E)dE\,dt$$

- Some issues:
    - detecting and identifying low surface brightness structures
    - resolving source in crowded fields - overlapping sources, diffuse emission
    - finding source boundaries
    - PSF uncertainties

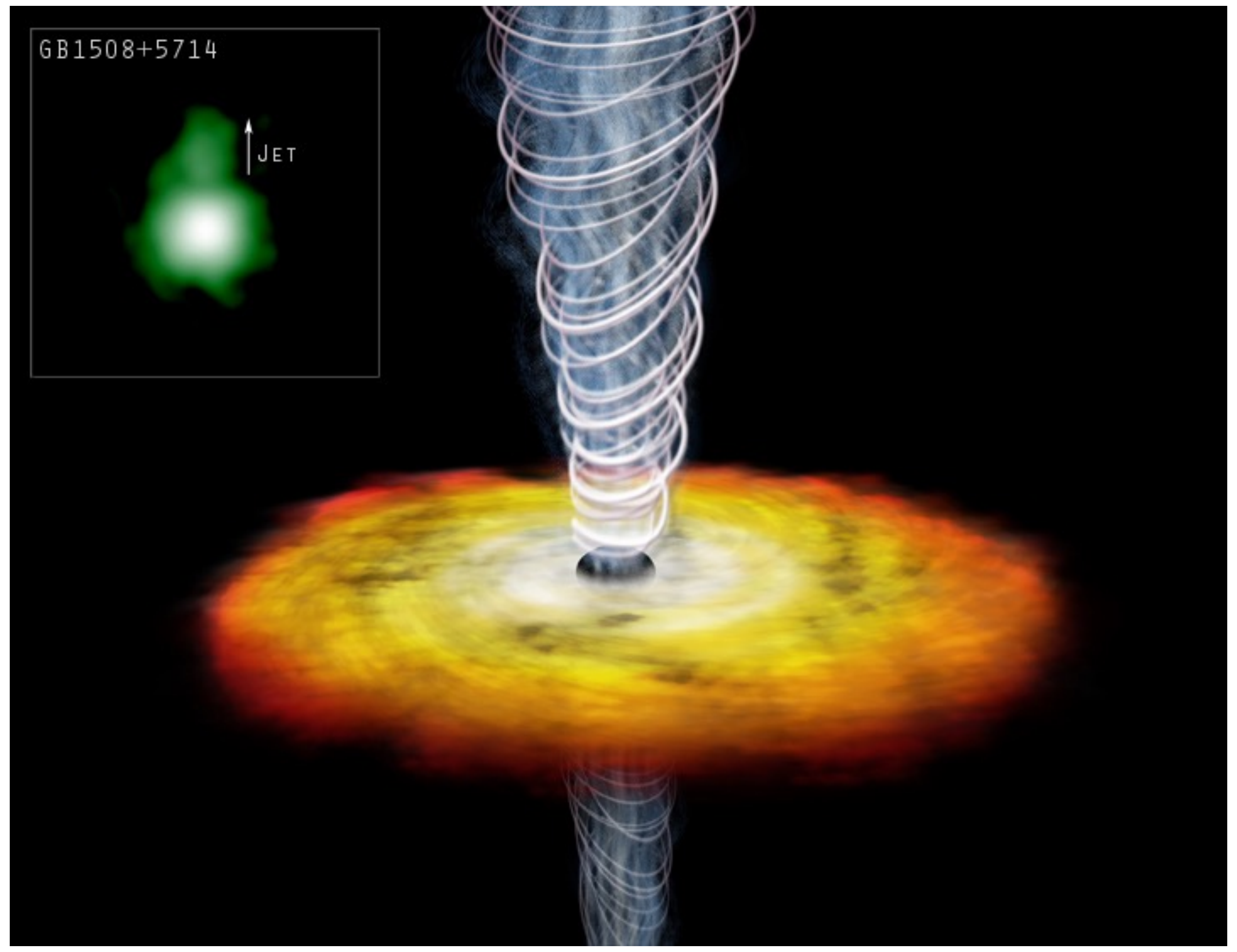
Chandra PSF



http://cxc.harvard.edu/proposer/POG/html/chap4.html#tth_fIg4.14

Observations and Data Collection

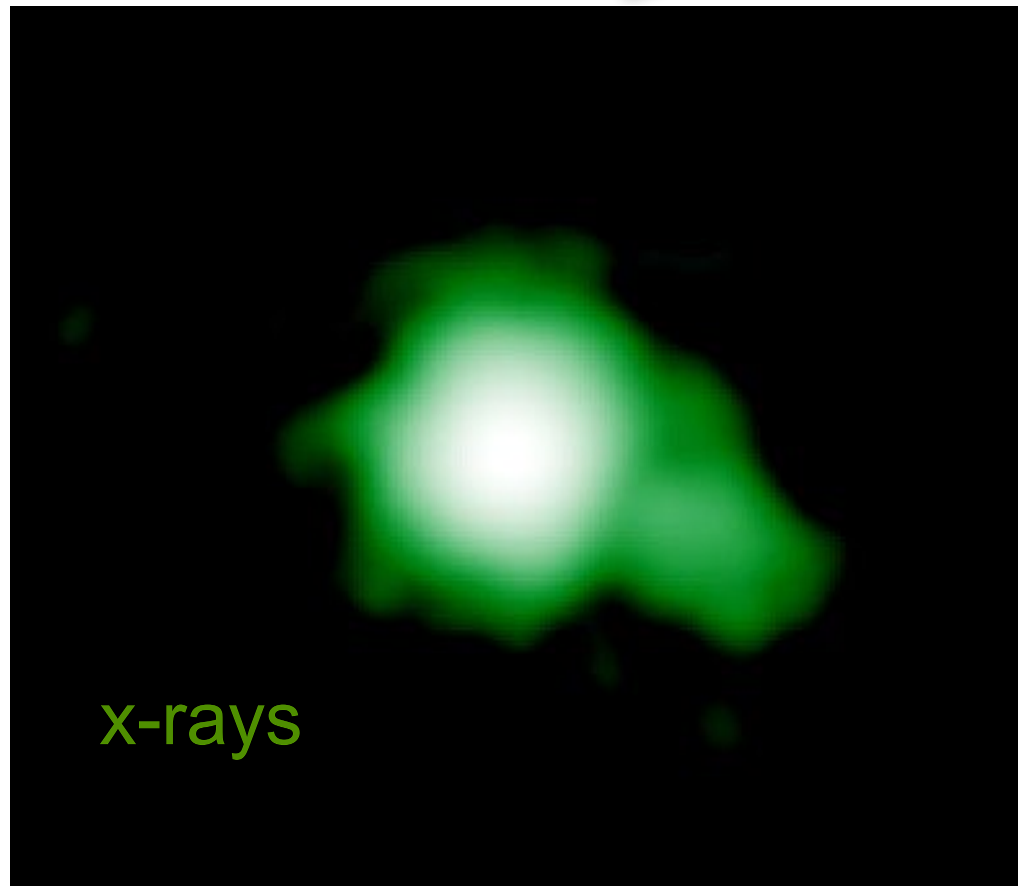
Astrophysical process



Detector collects photons, adds noise

Random number
of photons reach
the detector

?



draw conclusion
about the
astrophysical source

Scientific Experiment

Observations

Data Preparation

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

Modeling and Fitting Software for X-rays

- Sherpa is a modeling and fitting package in CIAO and Python; can fit models to spectra and images.
- X-ray spectral modeling tools:
  - XSPEC
  - ISIS
  - Spex
  - Bayesian X-ray Analysis (BAX) - package for Sherpa and XSPEC

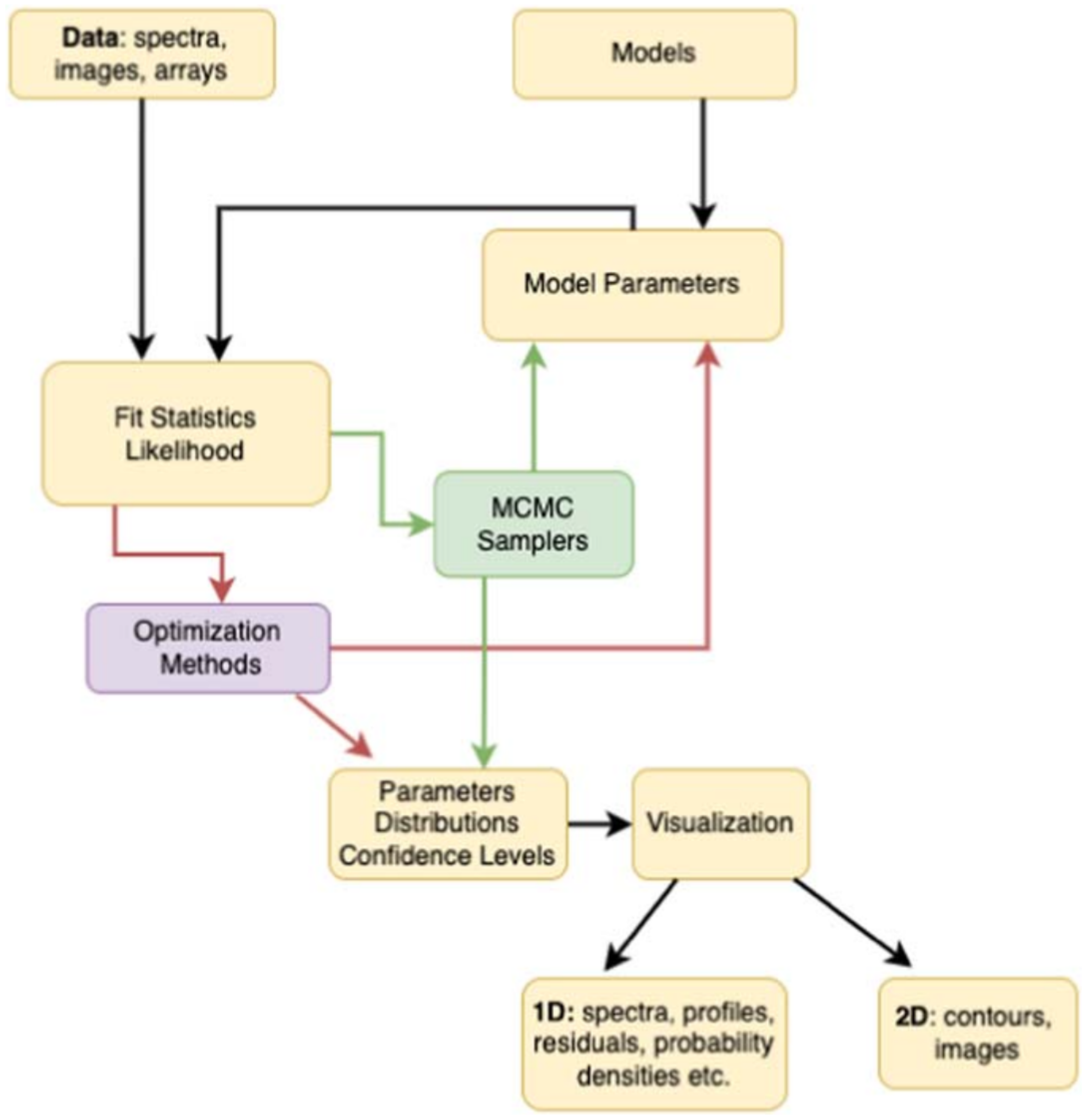
Sherpa

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:
ds9, matplotlib, bokeh

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_rmfs
  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)

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: $f(x_i, p_k)$

    absorption - N_H

    photon index of a power law function - Γ

    blackbody temperature kT

- Library of models

```
sherpa> list_models()
['absorptionedge',
 'absorptiongaussian',
'atten',
 'bbody',
 'bbodyfreq',
 'beta1d',
 'beta2d',
 ‘blackbody',
.........
```

- User Models can be added
- Model language to build compound model expressions

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

Sherpa

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

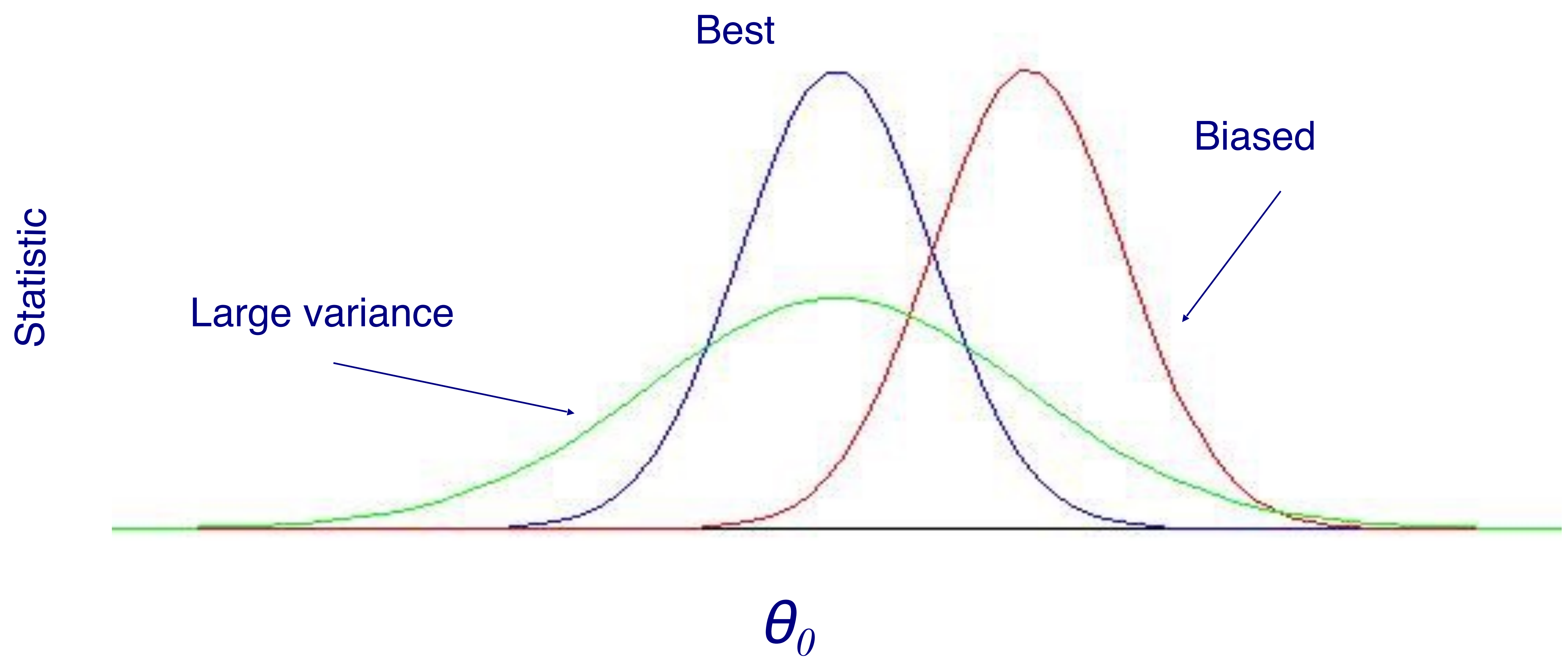
- Source and Background models:

```
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 as defined by different weights
and Poisson likelihood - cash/cstat/wstat

Fitting: Finding Best Model

```
sherpa-28> fit()
```

```
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()
```

```
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: Optimization Methods in Sherpa

- 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

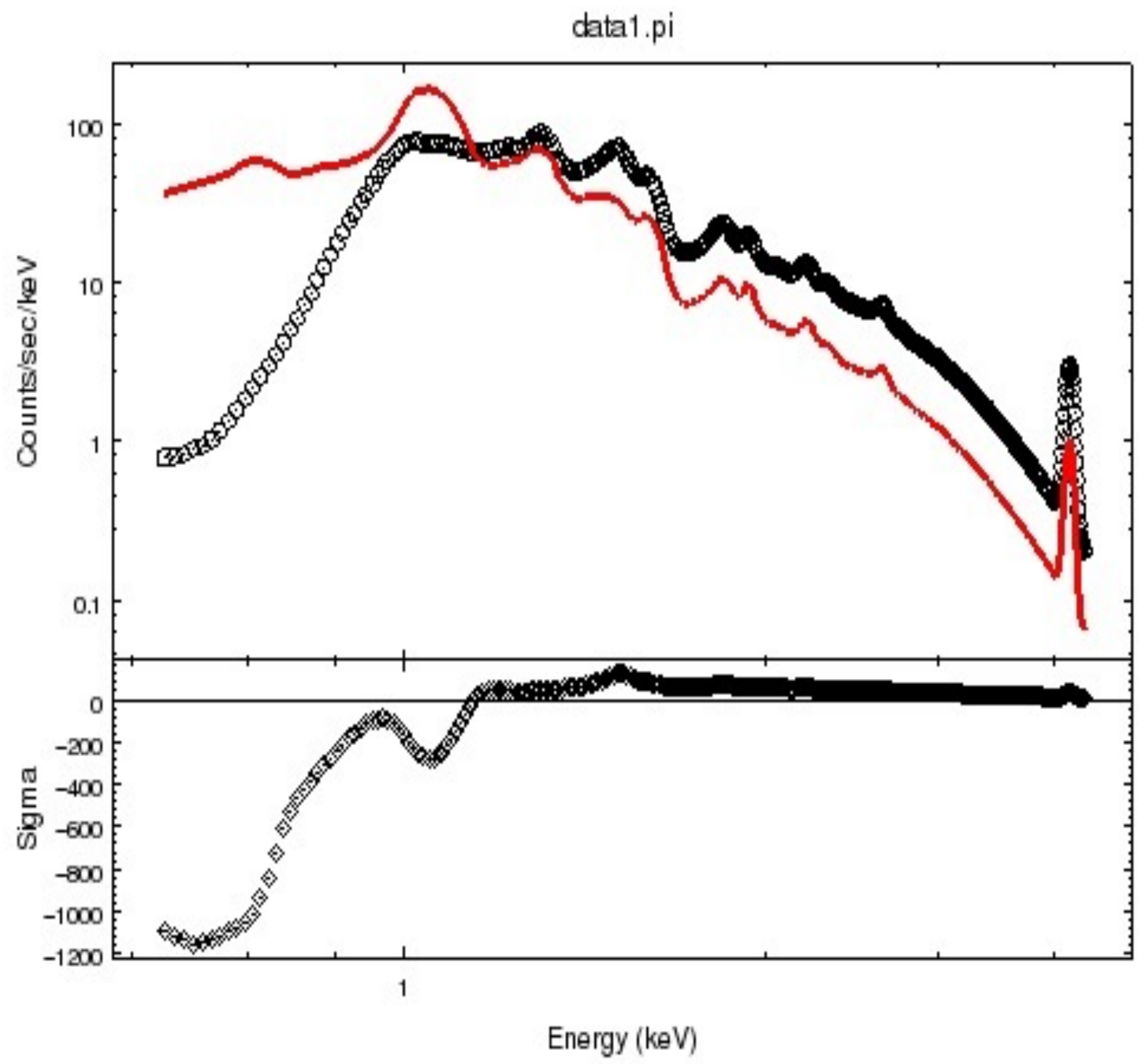
Optimization Methods: Comparison

Example: Spectral Fit with 3 methods

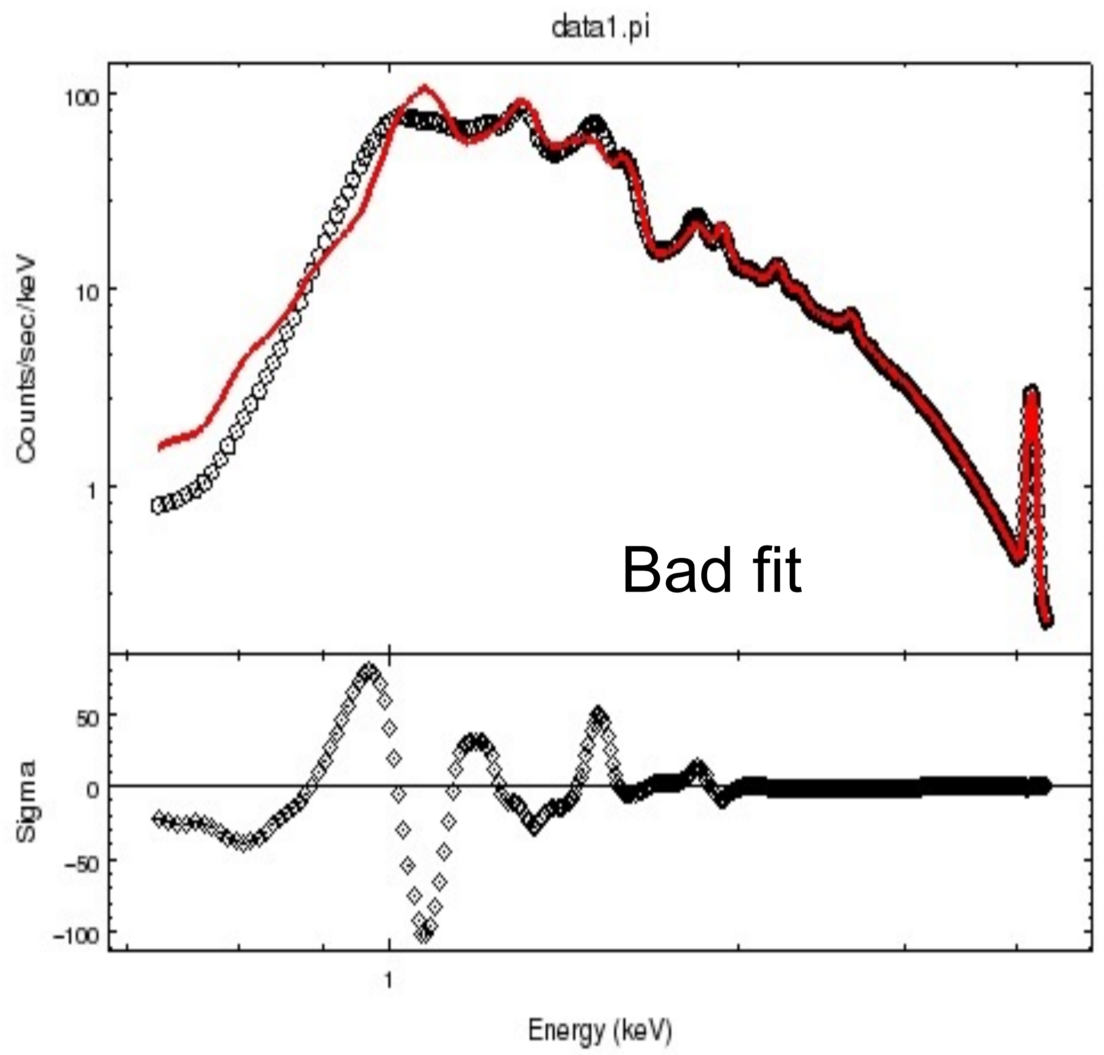
Data: high S/N simulated ACIS-S spectrum of the two temperature plasma
Model: photoelectric absorption plus two MEKAL components (correlated!)

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

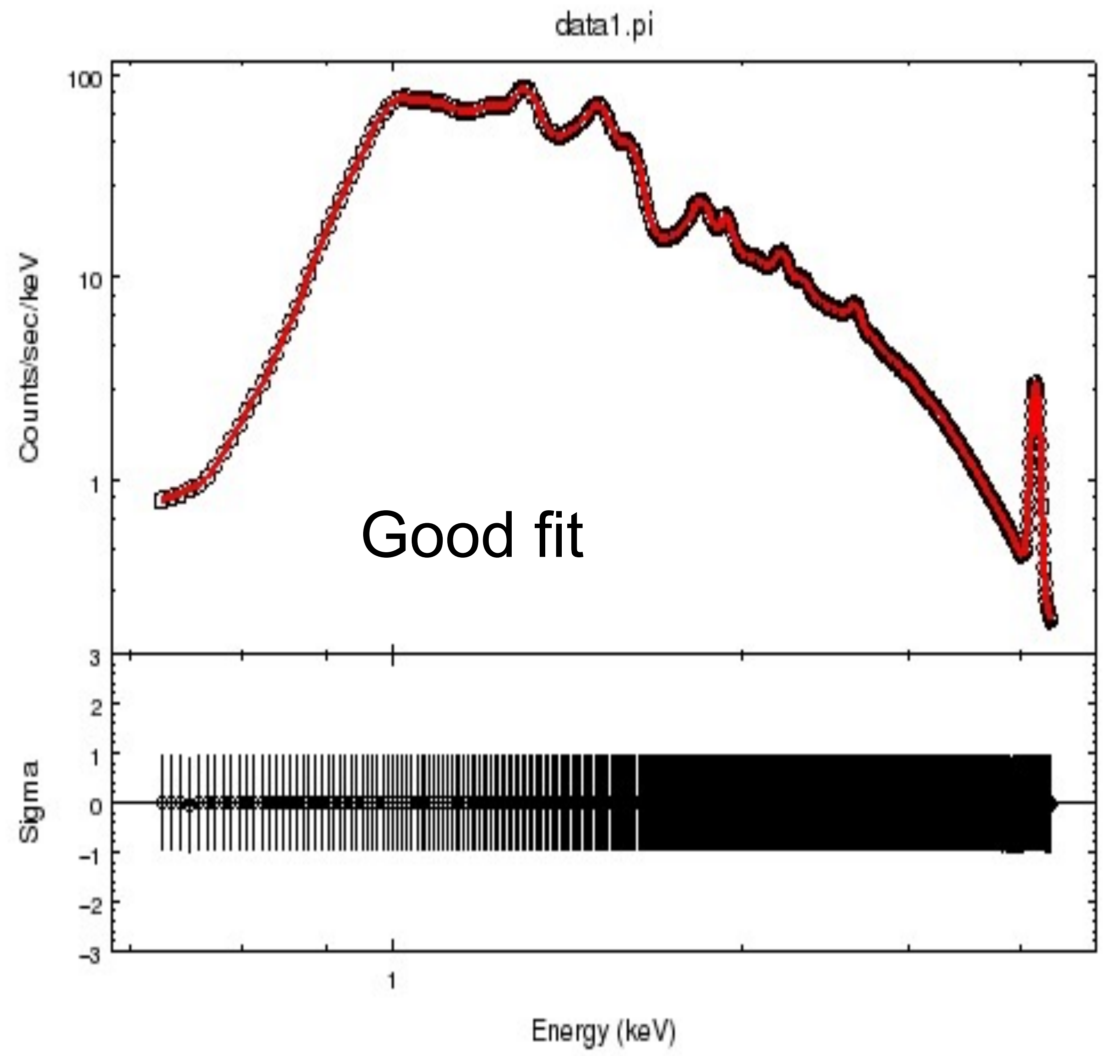
Method	Number of Iterations	Final Statistics
Levmar	31	1.55e5
Neldermead	1494	0.0542
Moncar	13045	0.0542



Data and Model with initial parameters

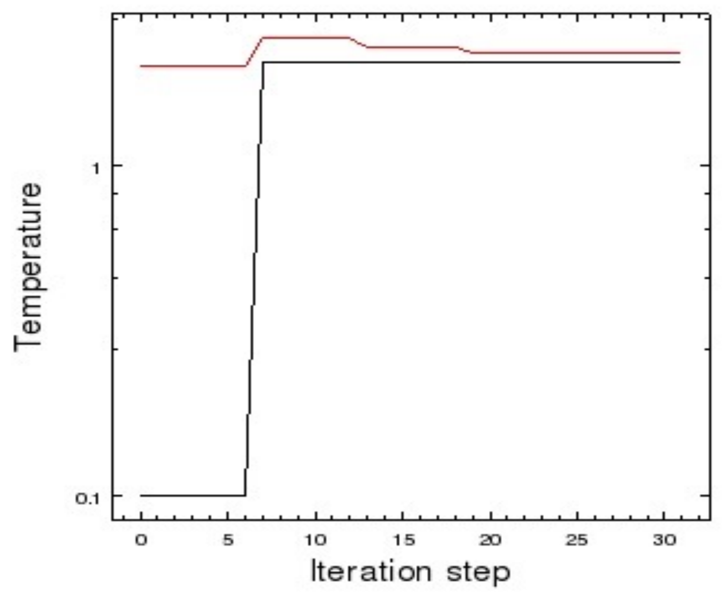
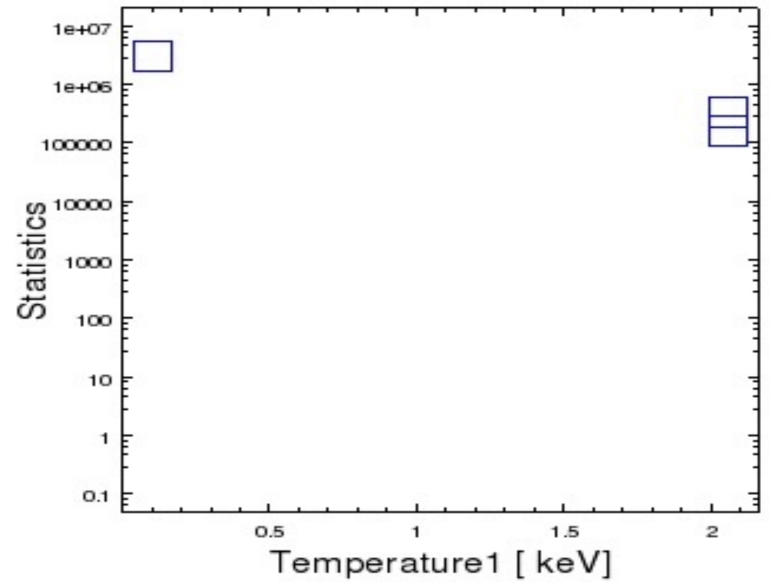
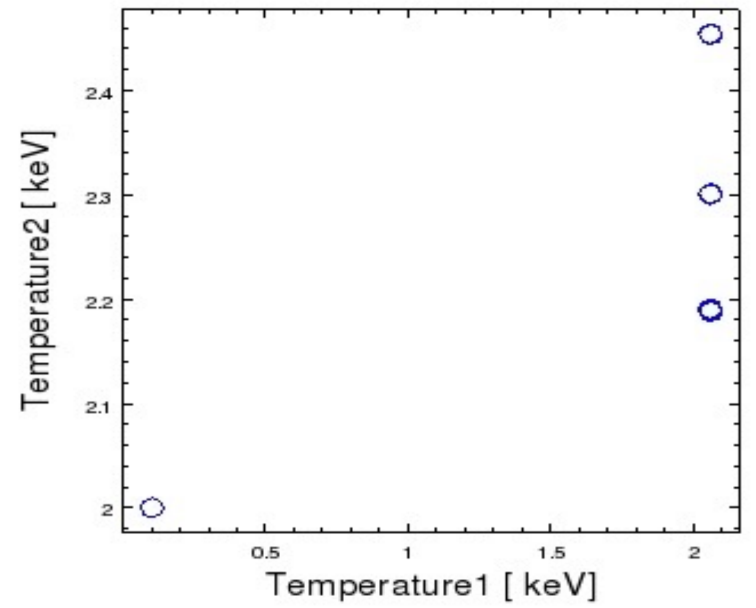
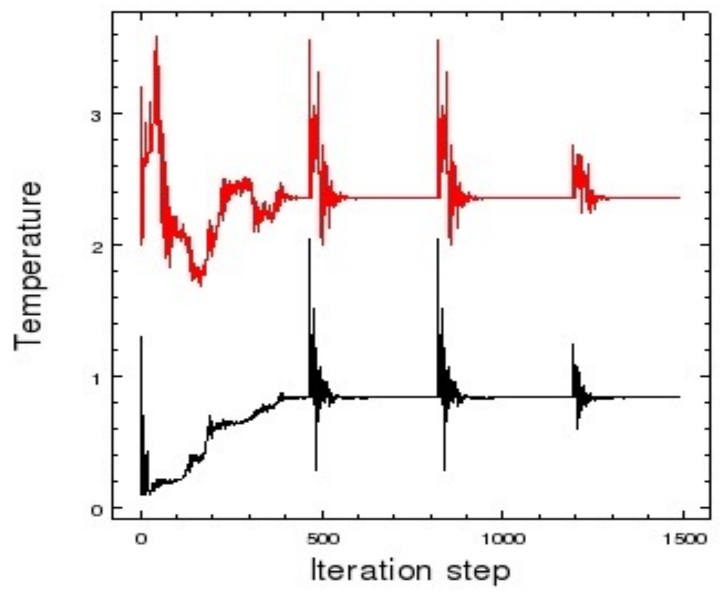
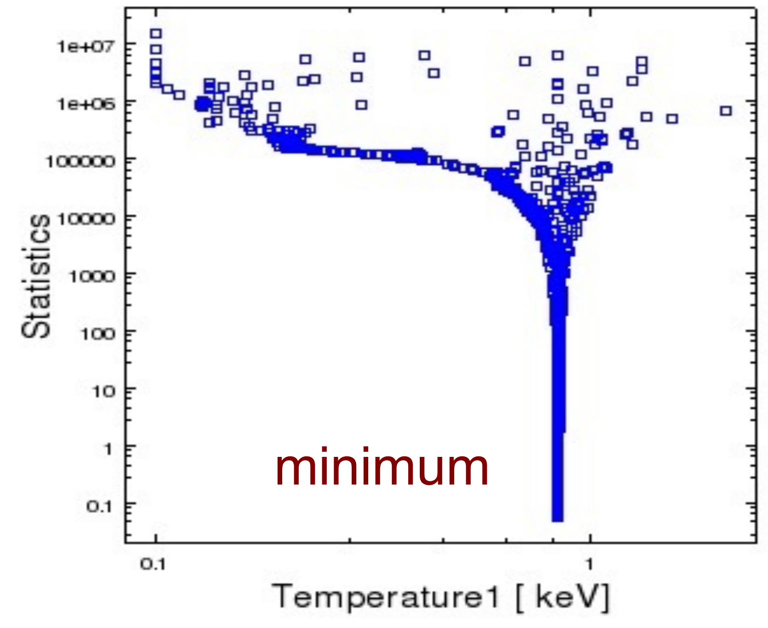
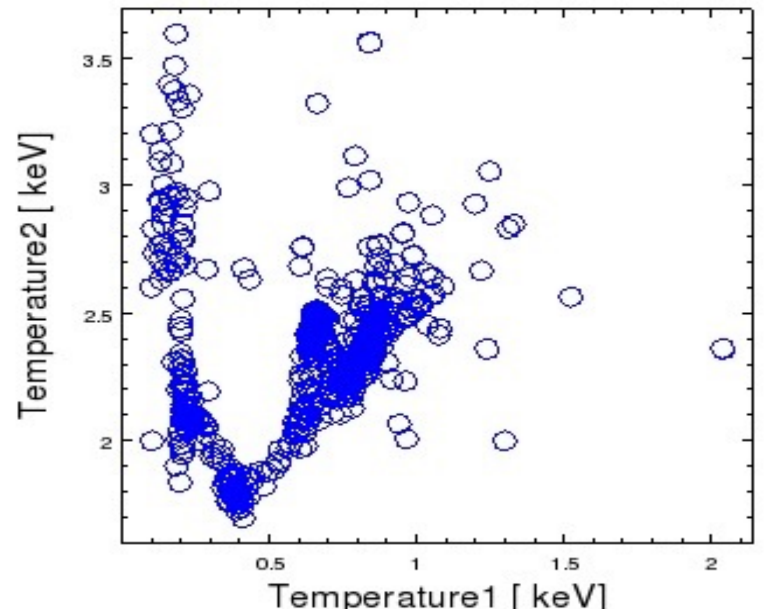
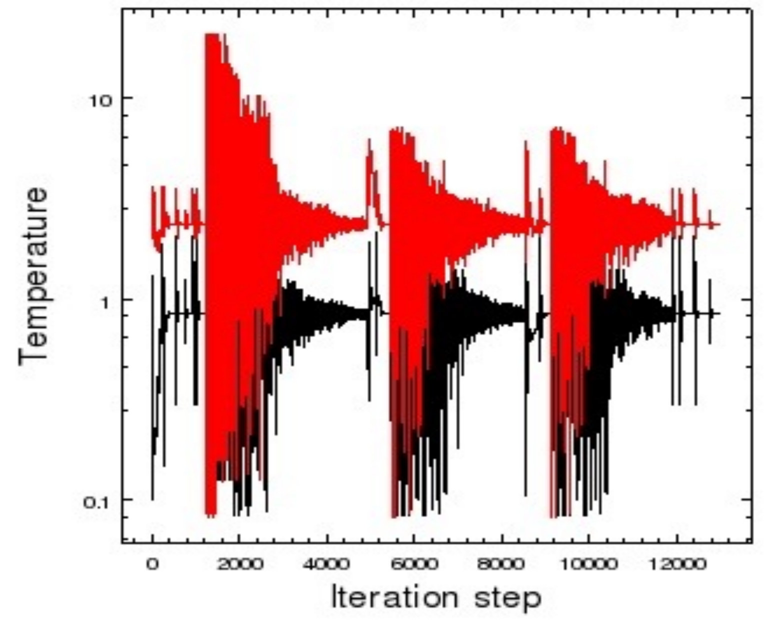
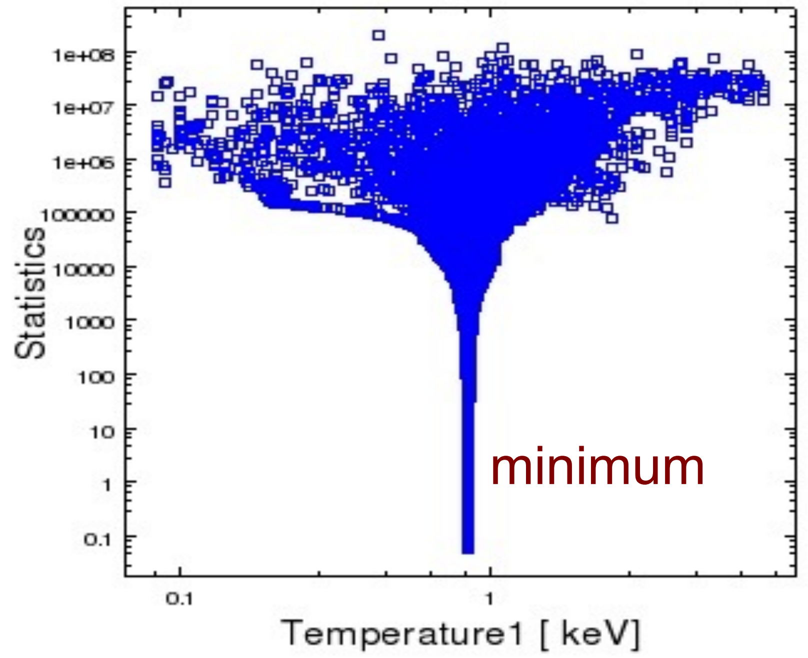
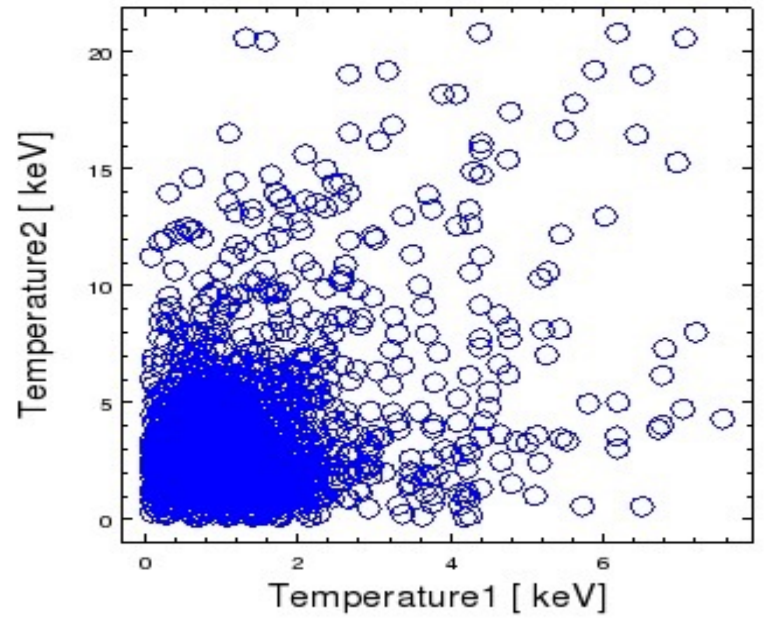


Levmar fit



Nelder-Mead and Moncar fit

Optimization Methods: Comparison

	Temperature	Statistics vs. Temperature	2D slice of Parameter Space probed by each method
levmar			
simplex			
moncar			

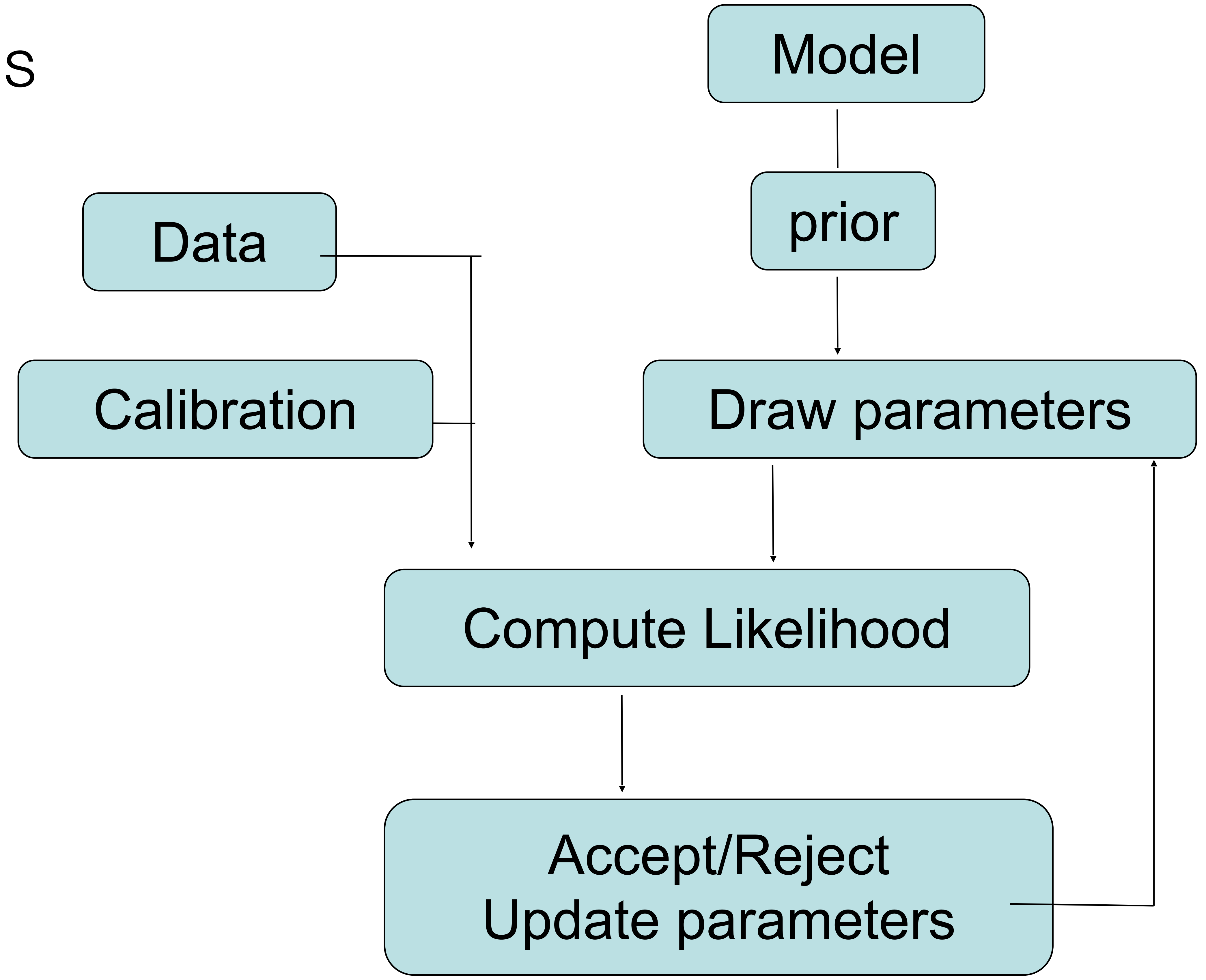
Sherpa, MCMC and Bayesian Analysis

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.



Sherpa, MCMC and Bayesian Analysis

MCMC samplers:

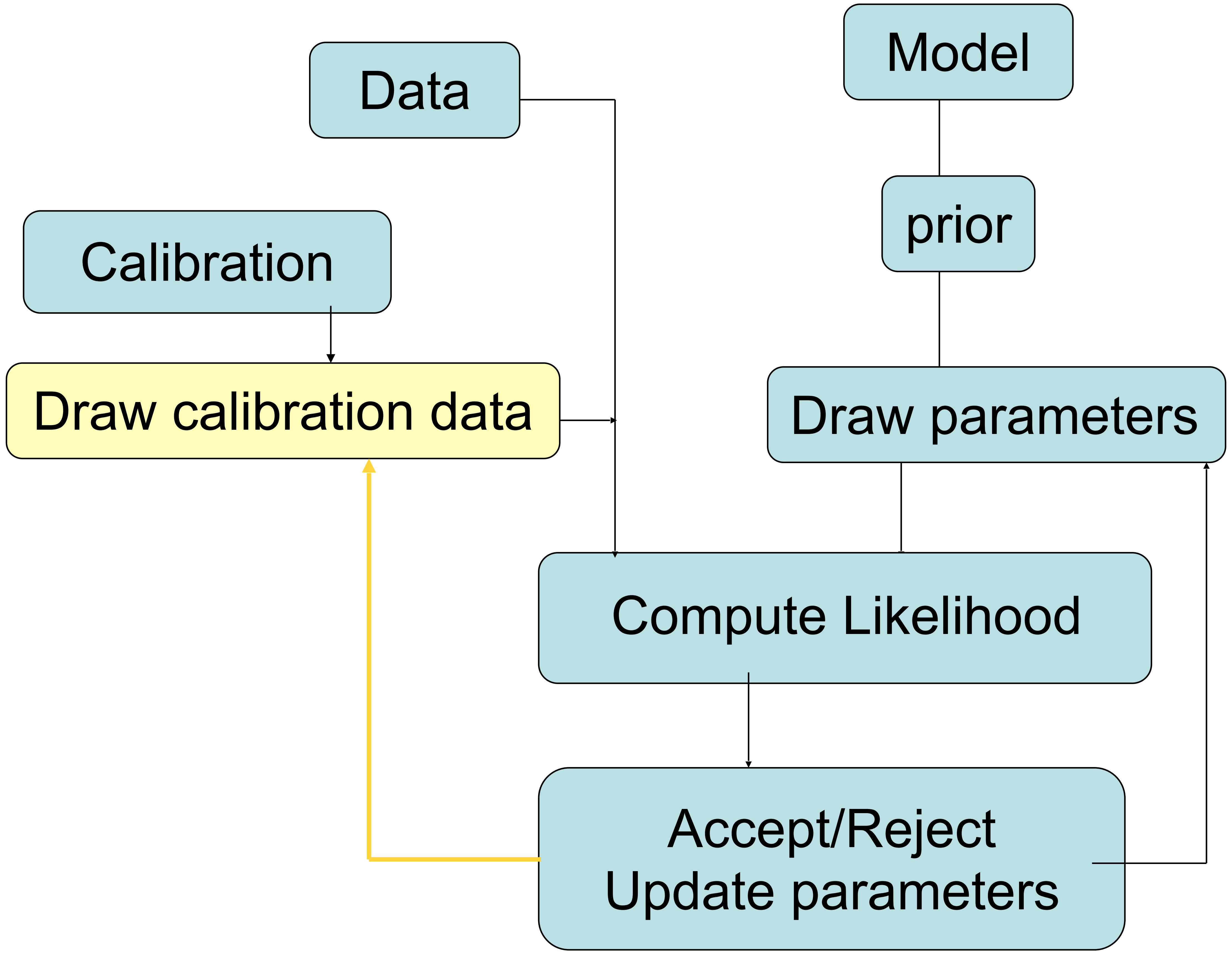
Metropolis and Metropolis-Hastings algorithms

Support for Bayesian analysis with priors.

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

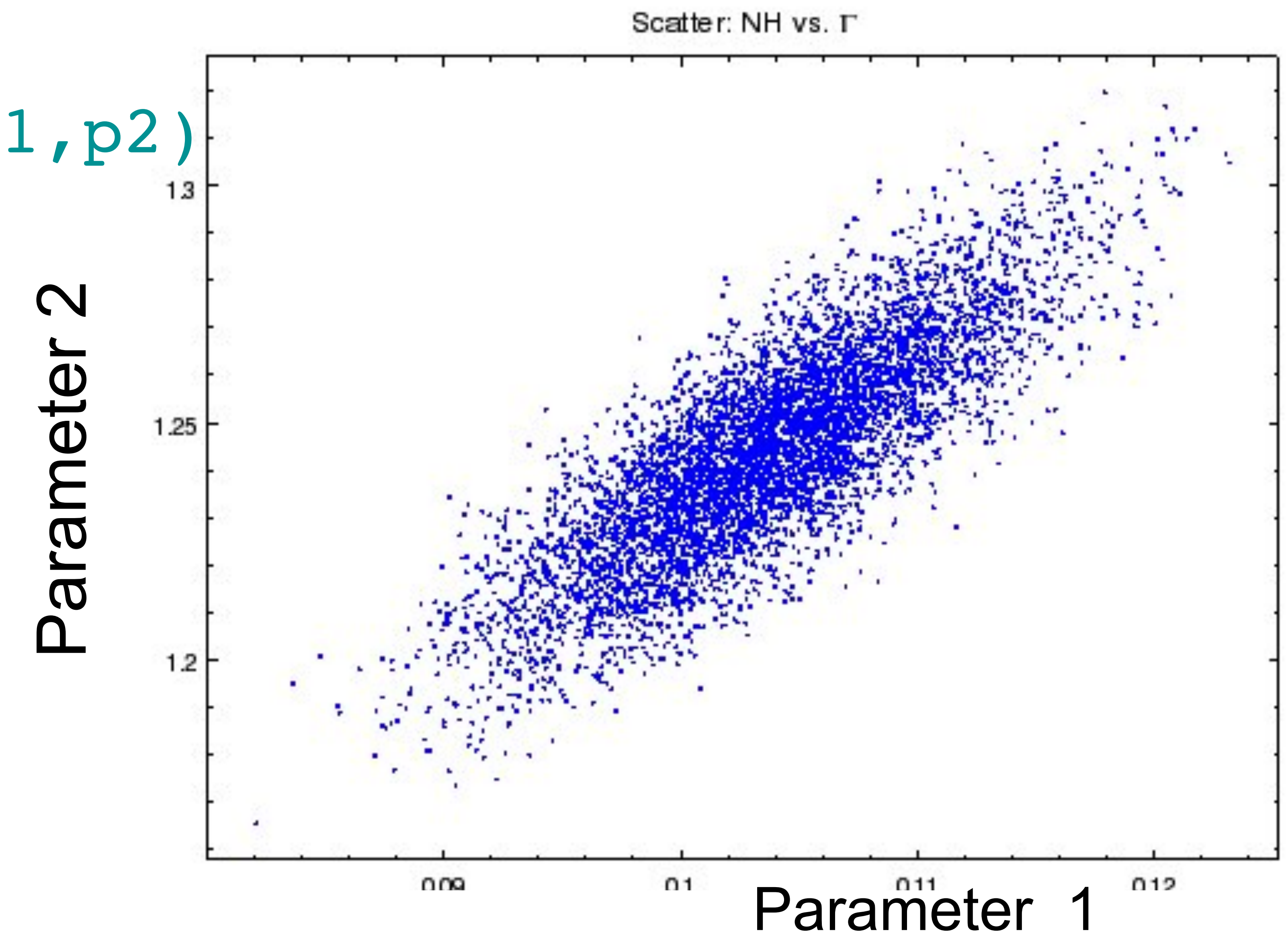
- Computed parameter uncertainties can include systematic or calibration errors.

- Simulates replicate data from the posterior predictive distributions.

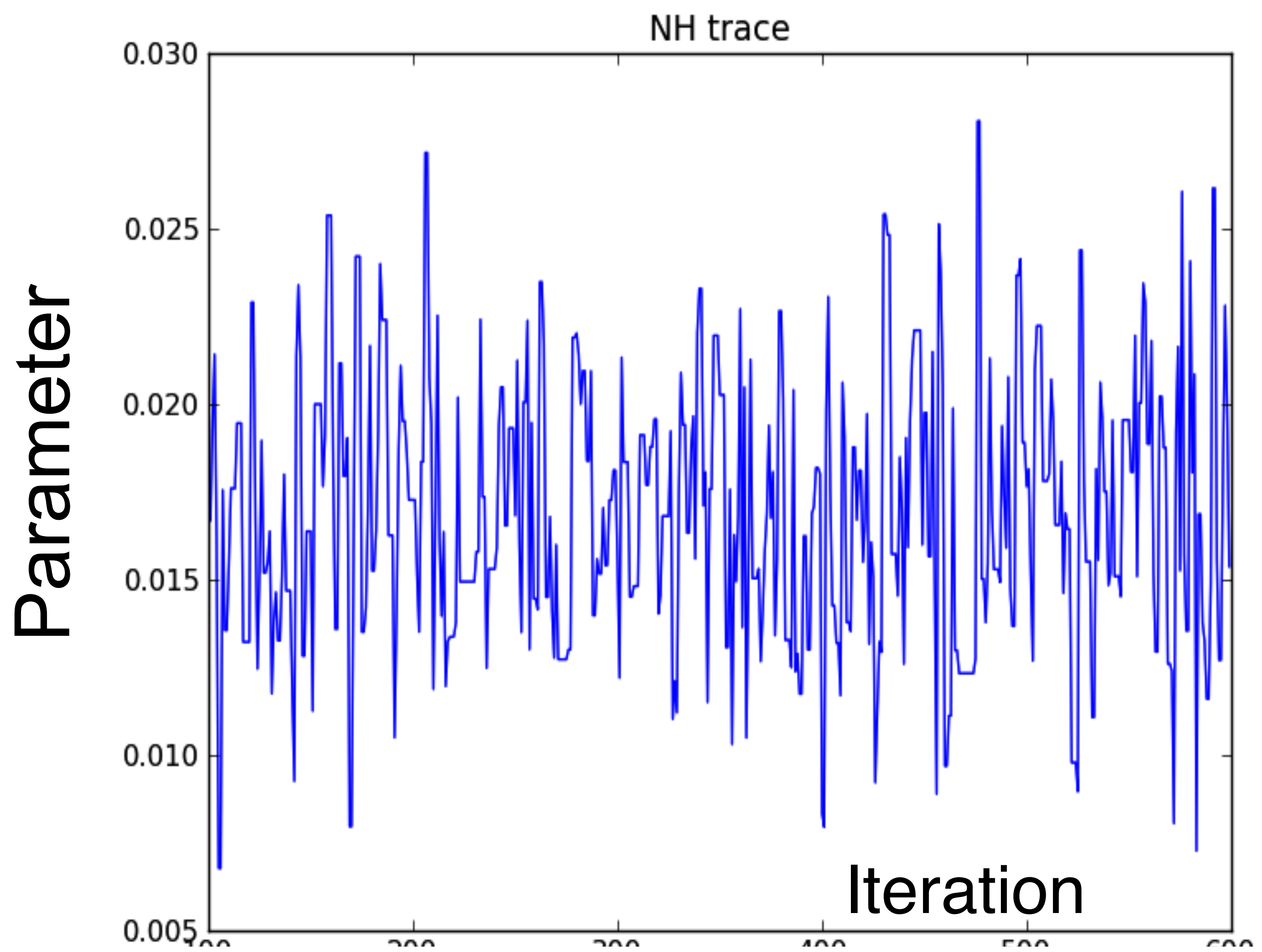


Visualization of the MCMC Results

plot_scatter(p1,p2)

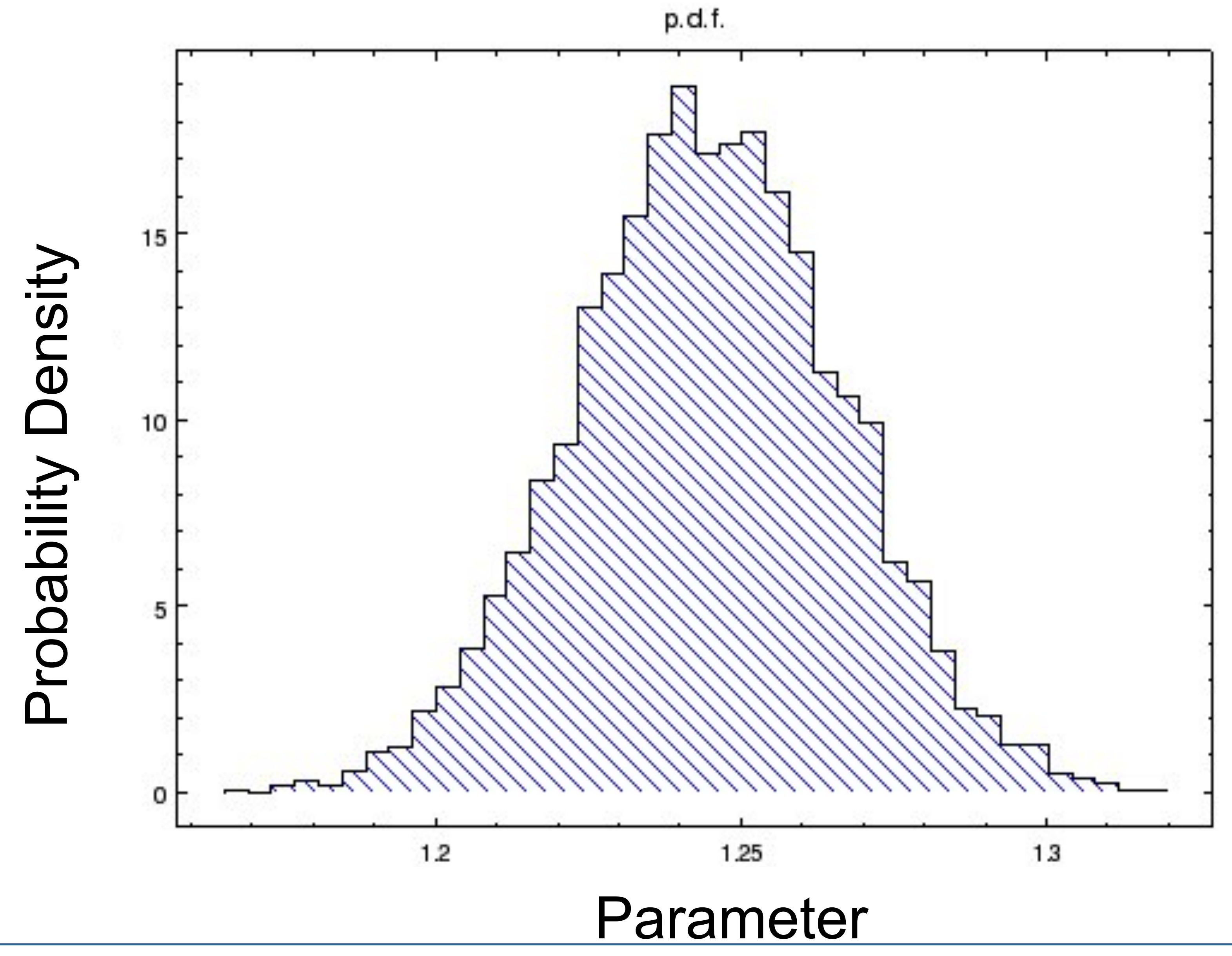


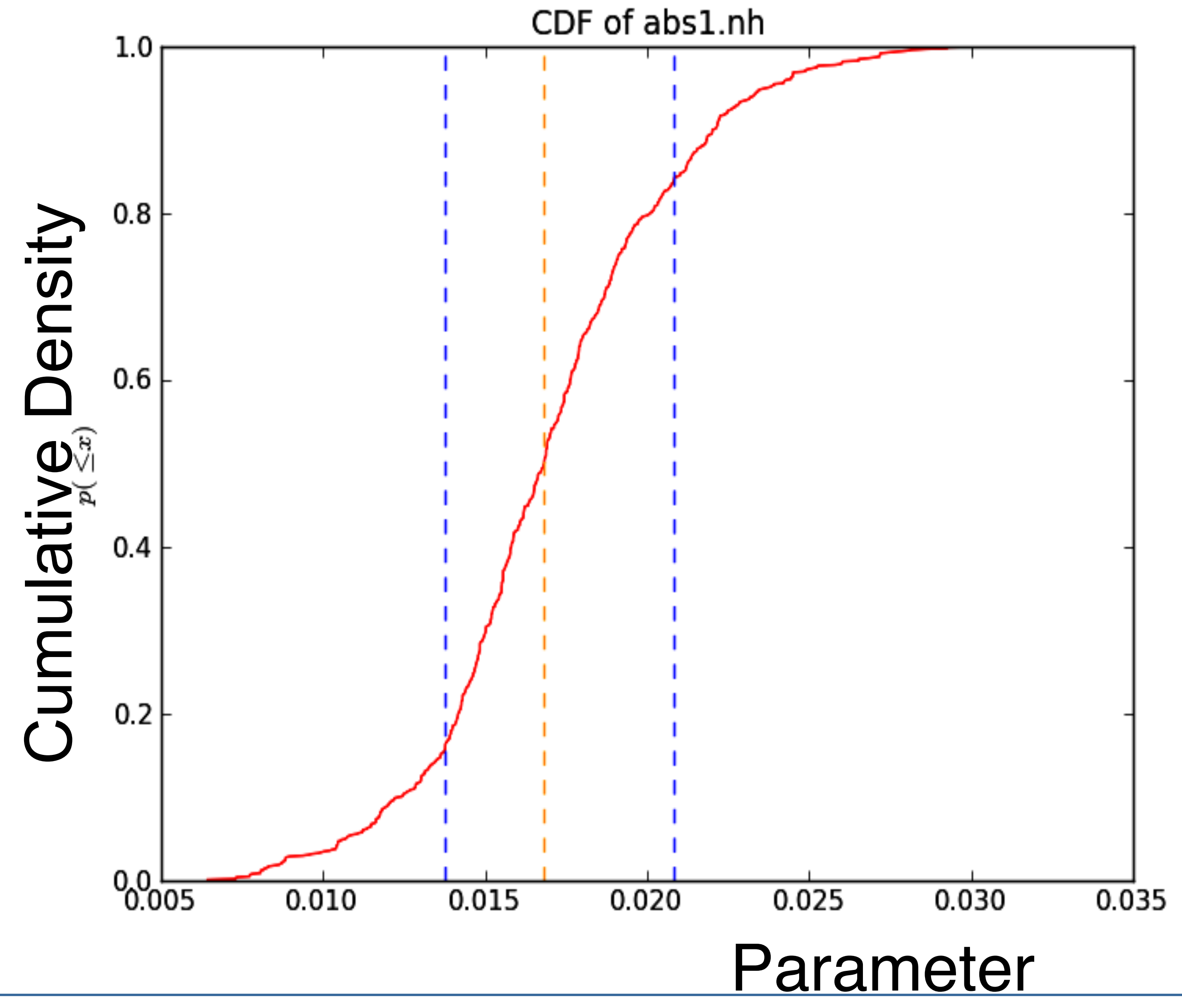
Trace of a parameter during MCMC run



plot_trace(p1)

plot_pdf(p1)





plot_cdf(p1)

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?

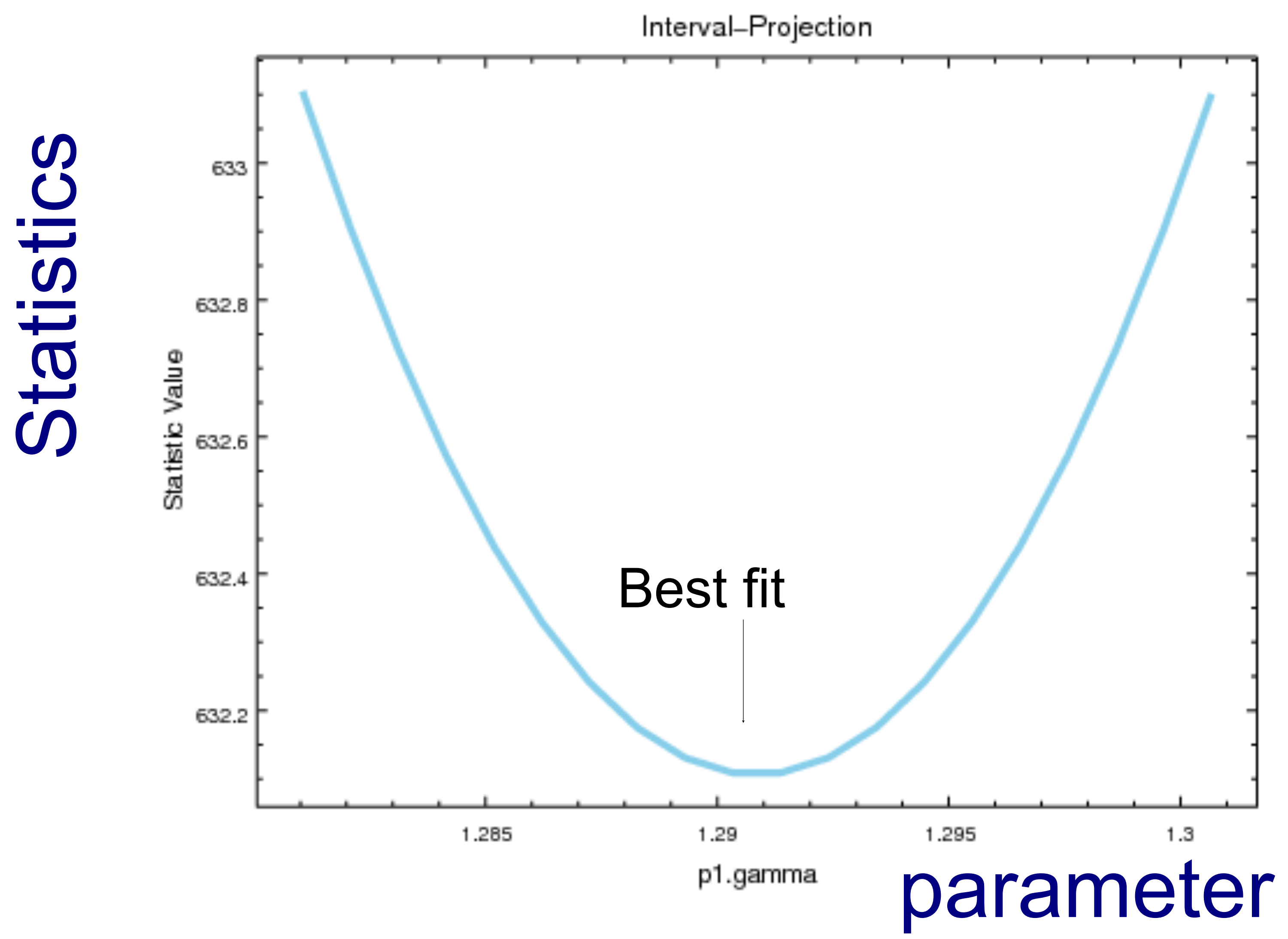
Confidence Intervals

sherpa-40> **covariance()**

```
Dataset              = 1
Confidence Method     = covariance
Fitting Method        = neldermead
Statistic             = chi2datavar
covariance 1-sigma (68.2689%) bounds:
   Param            Best-Fit  Lower Bound  Upper Bound
   -----            --------  -----------  -----------
   abs1.nH           1.1015  -0.00153623   0.00153623
   mek1.kT         0.841024  -0.00115618   0.00115618
   mek1.norm       0.699764  -0.00395776   0.00395776
   mek2.kT          2.35844  -0.00371253   0.00371253
   mek2.norm        1.03725  -0.00172503   0.00172503
```

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
mek2.norm lower bound:  -0.00164417
mek2.norm upper bound:  0.00164816
abs1.nH lower bound:    -0.00147622
mek1.norm upper bound:  0.00376011
abs1.nH upper bound:    0.00147268
Dataset              = 1
Confidence Method     = confidence
Fitting Method        = neldermead
Statistic             = chi2datavar
confidence 1-sigma (68.2689%) bounds:
   Param            Best-Fit  Lower Bound  Upper Bound
   -----            --------  -----------  -----------
   abs1.nH           1.1015  -0.00147622   0.00147268
   mek1.kT         0.841024  -0.00113811    0.0011439
   mek1.norm       0.699764  -0.00377224   0.00376011
   mek2.kT          2.35844  -0.00365452   0.00364805
   mek2.norm        1.03725  -0.00164417   0.00164816
```



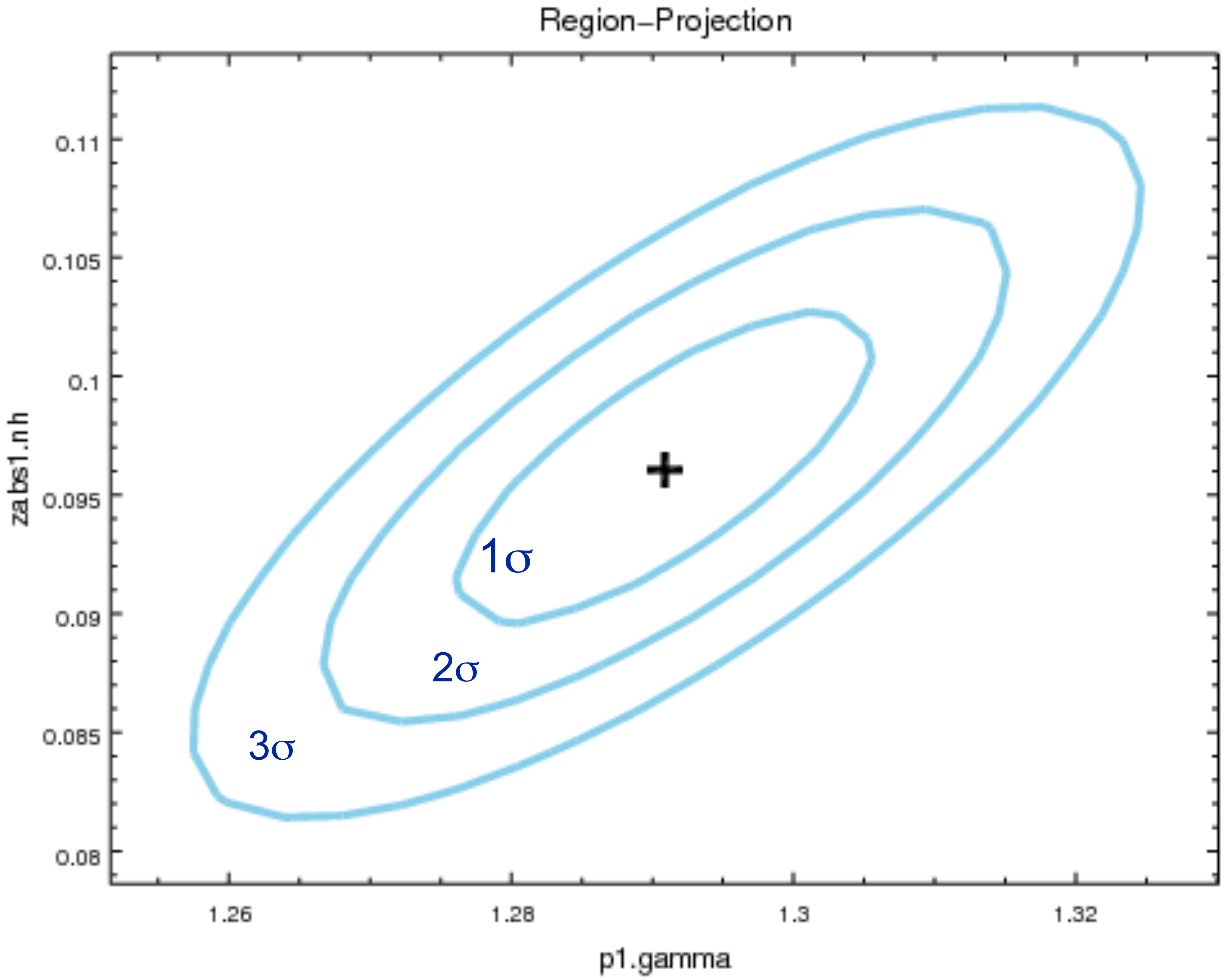
sherpa-42> **print get_conf_results()**

```
----------> print(get_conf_results())
datasets    = (1,)
methodname = confidence
fitname     = neldermead
statname    = chi2datavar
sigma       = 1
percent     = 68.2689492137
parnames    = ('abs1.nH', 'mek1.kT', 'mek1.norm', 'mek2.kT', 'mek2.norm')
parvals     = (1.1015003421601872, 0.84102381214069499, 0.69976355976410642,
2.3584395600380756, 1.0372453037692799)
parmins     = (-0.0014762187156509565, -0.001138111192153346, -0.0037722356859711814,
-0.0036545192286010497, -0.0016441656050858455)
parmaxes    = (0.001472679745547989, 0.0011439029752089436, 0.0037601110158367312,
0.003648045819133916, 0.001648162229710648)
nfits       = 103
```

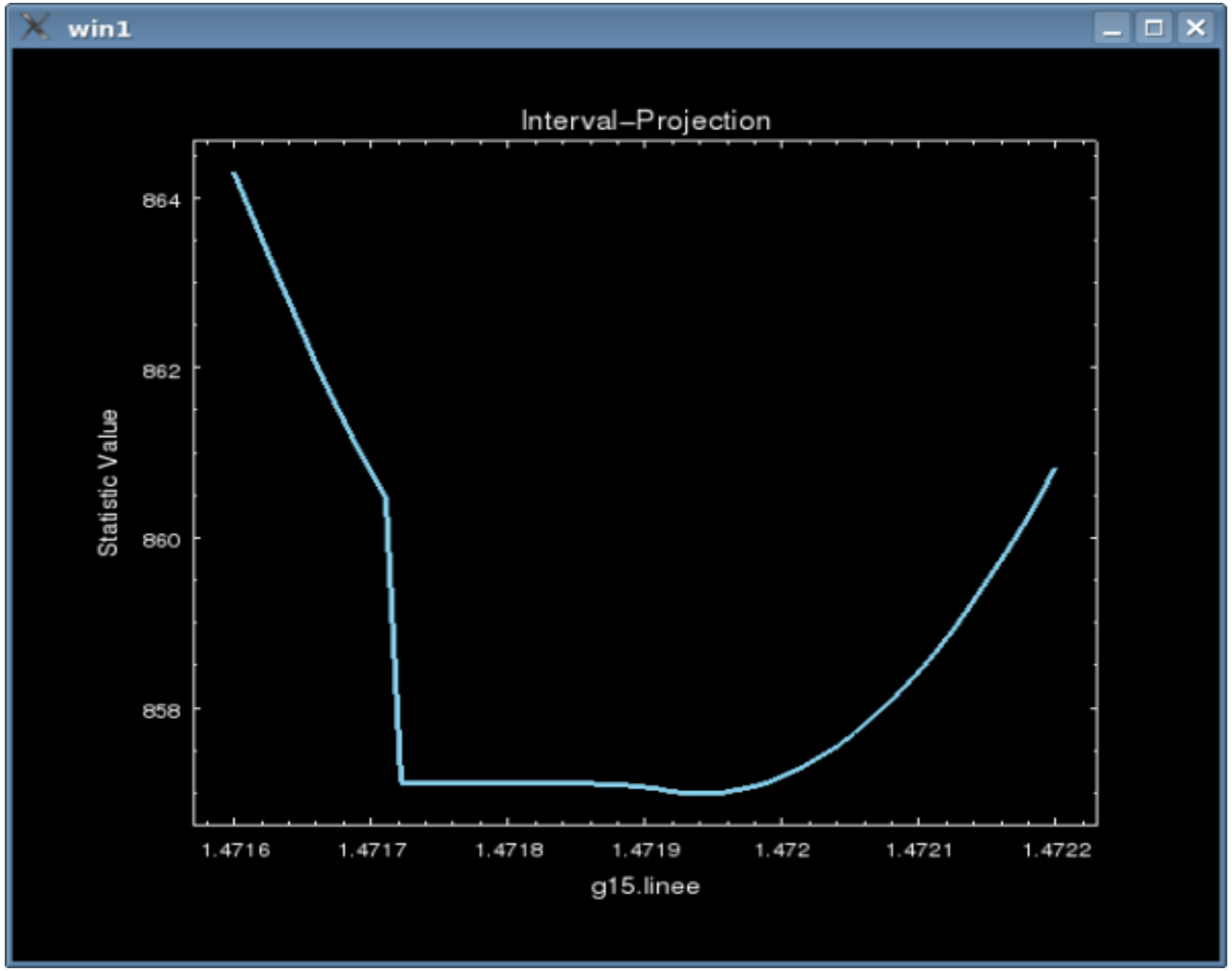
Confidence Regions

```
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]

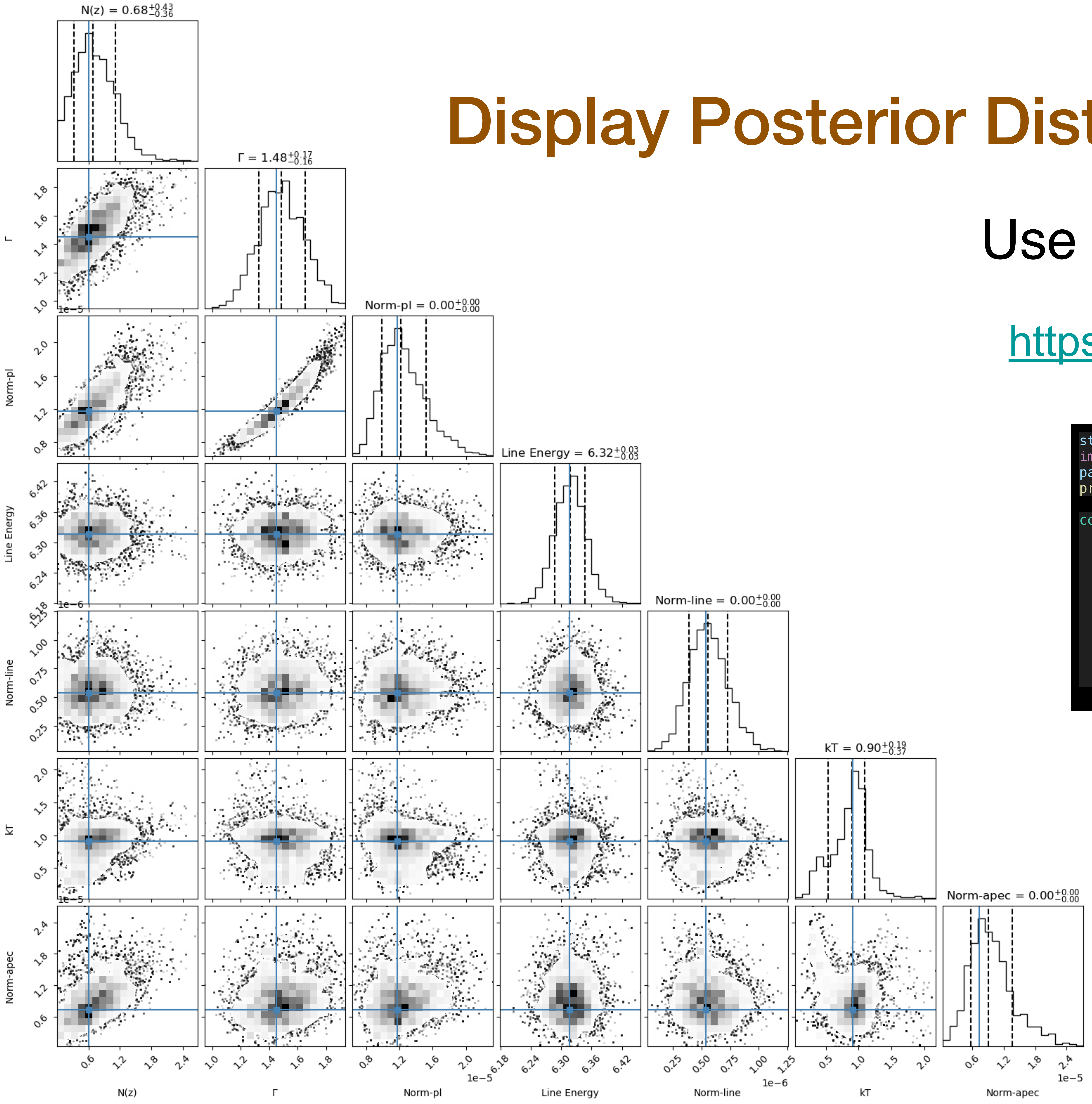


Not well-behaved Surface



Non-Gaussian Shape

Display Posterior Distributions from MCMC



Use corner python package:

https://corner.readthedocs.io/en/latest/

```
stats, accept, params = get_draws(niter=5e4)
import corner
parvals = [p.val for p in get_model().pars if not p.frozen]
print(parvals)

corner.corner(params[0:].T,
              quantiles=[0.16, 0.5, 0.84],
              labels=[r"N(z)",
                      r"$\Gamma$",
                     r"Norm-pl",
                     r"Line Energy",
                     r"Norm-line",
                     r"kT",
                     r"Norm-apec"],
              show_titles=True,
              plot_contours=False, truths=parvals)
```

Distributions of Flux and Parameters

Functions: sample_energy_flux, sample_flux

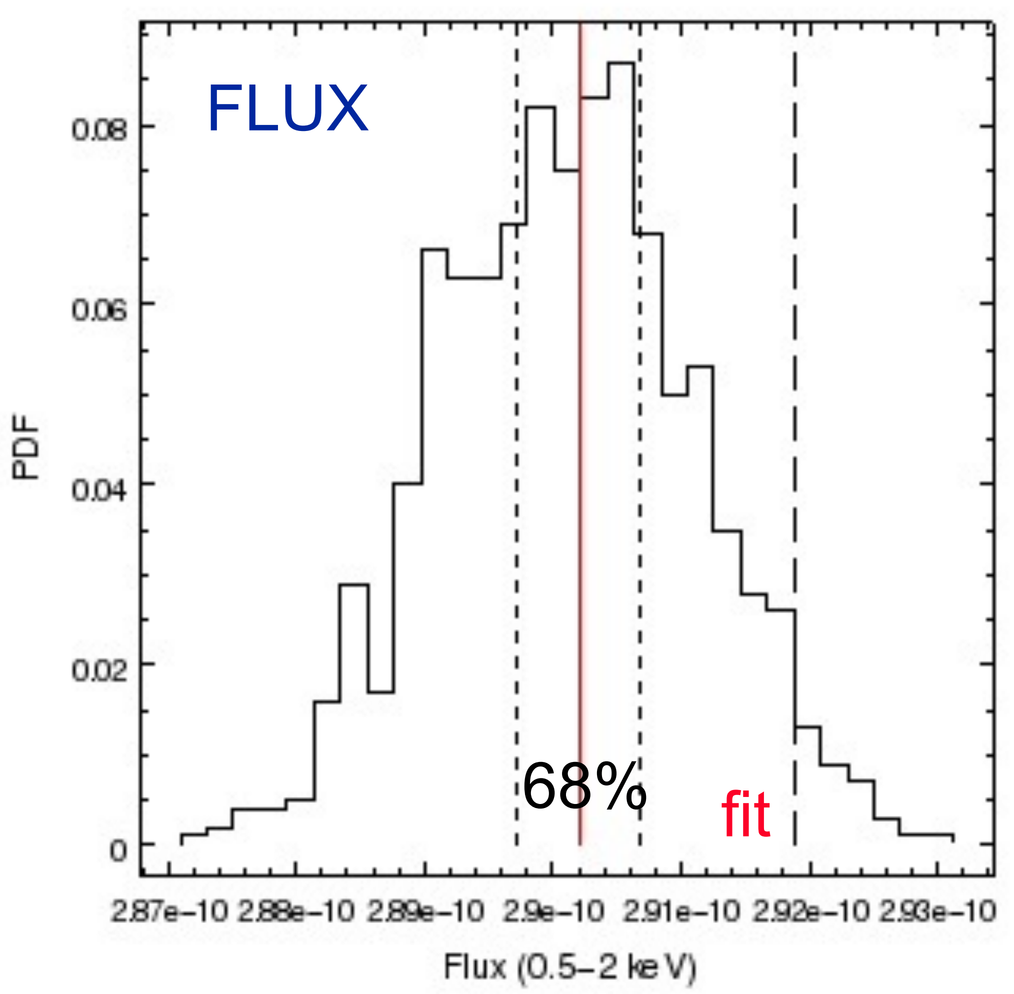
Monte Carlo Simulations of parameters assuming Gaussian distributions for all the parameters
Characterized by the covariance matrix, includes correlations between parameters.

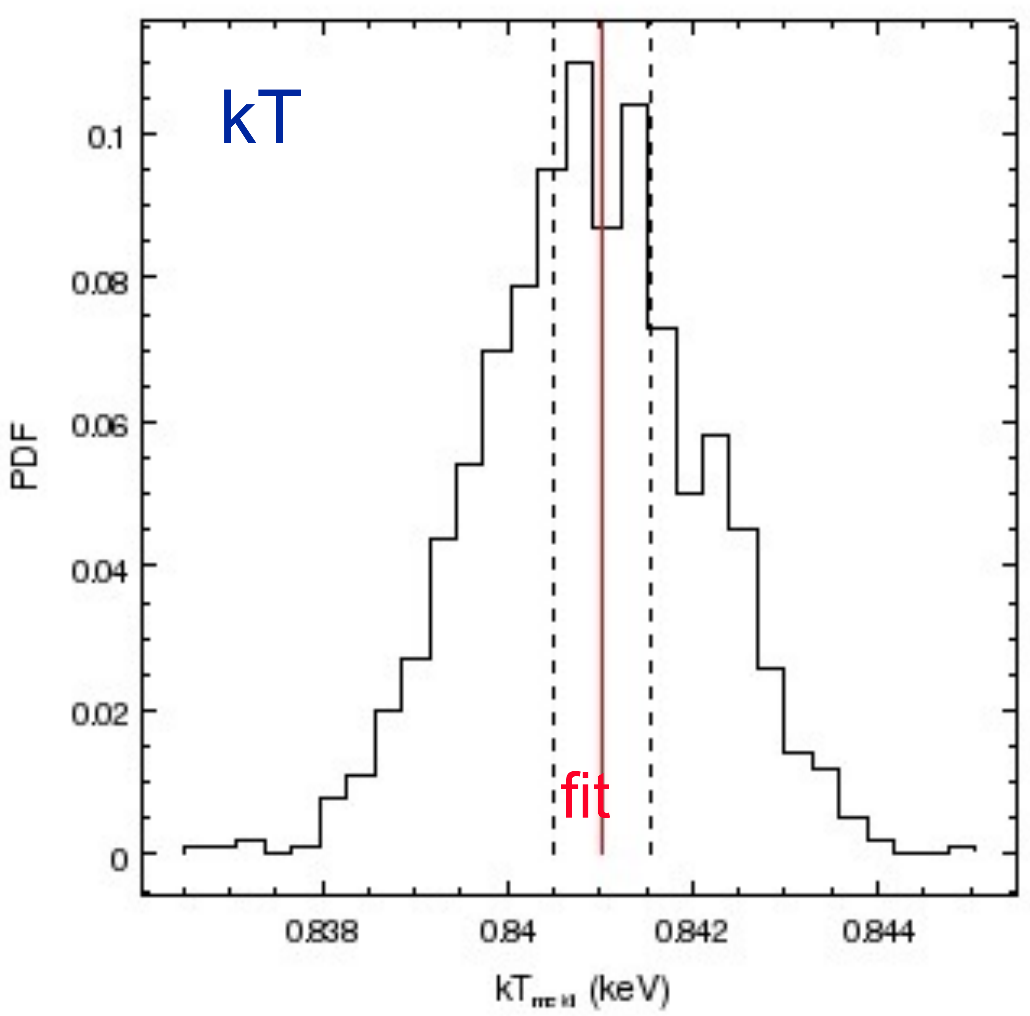
```
sherpa-19> flux100=sample_energy_flux(0.5,2.,num=100)
sherpa-20> print flux100
---------> print(flux100)
[[  2.88873592e-10   1.10331438e+00   8.40356670e-01   6.97503733e-01
    2.35411369e+00   1.03580042e+00]
 [  2.90279483e-10   1.10243140e+00   8.41174148e-01   7.01009661e-01
 sherpa-26> plot_energy_flux(0.5,2,num=1000)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

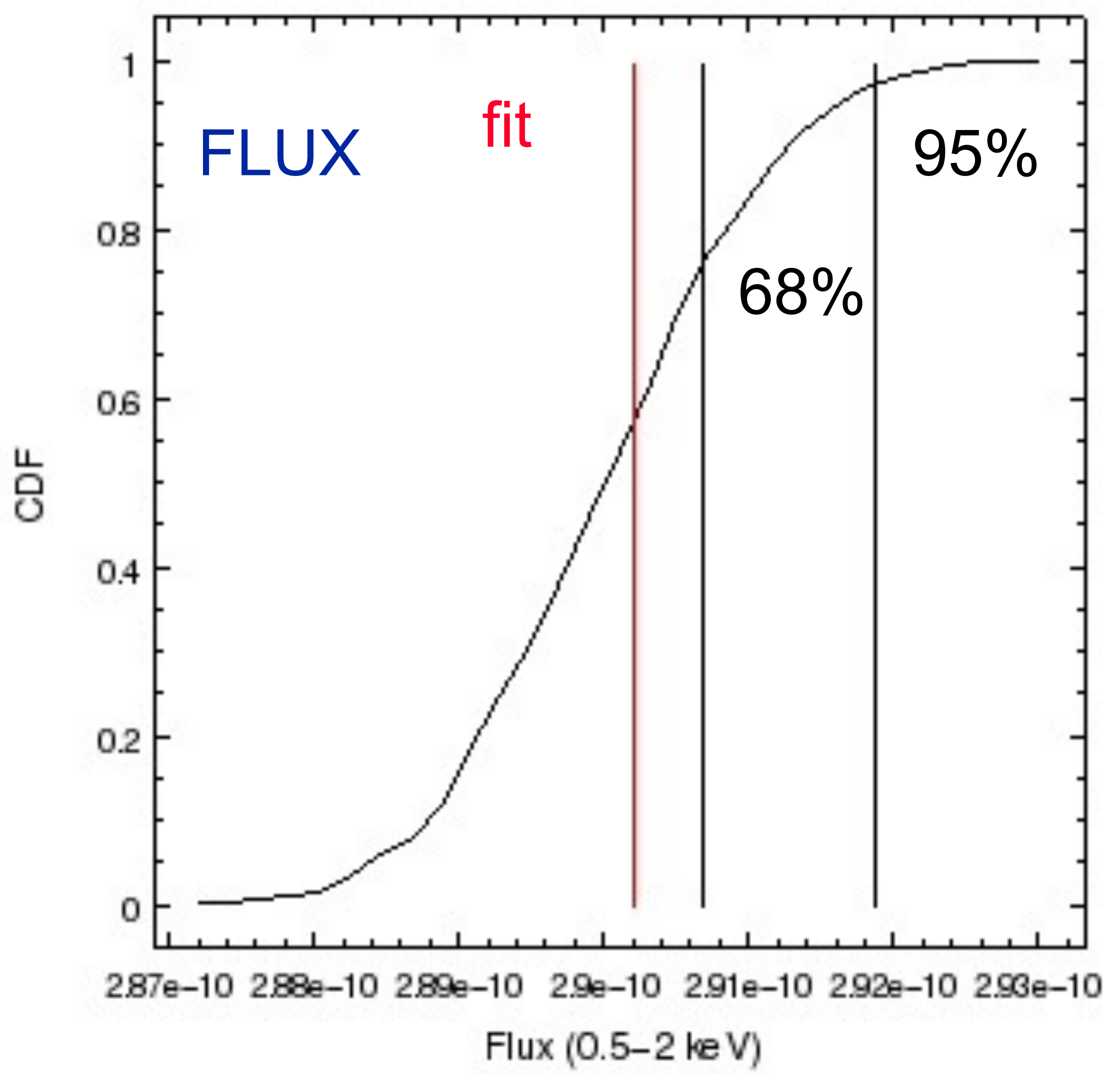
* Characterize distributions: plot PDF and CDF and obtain Quatiles of 68% and 95%

```
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)
```

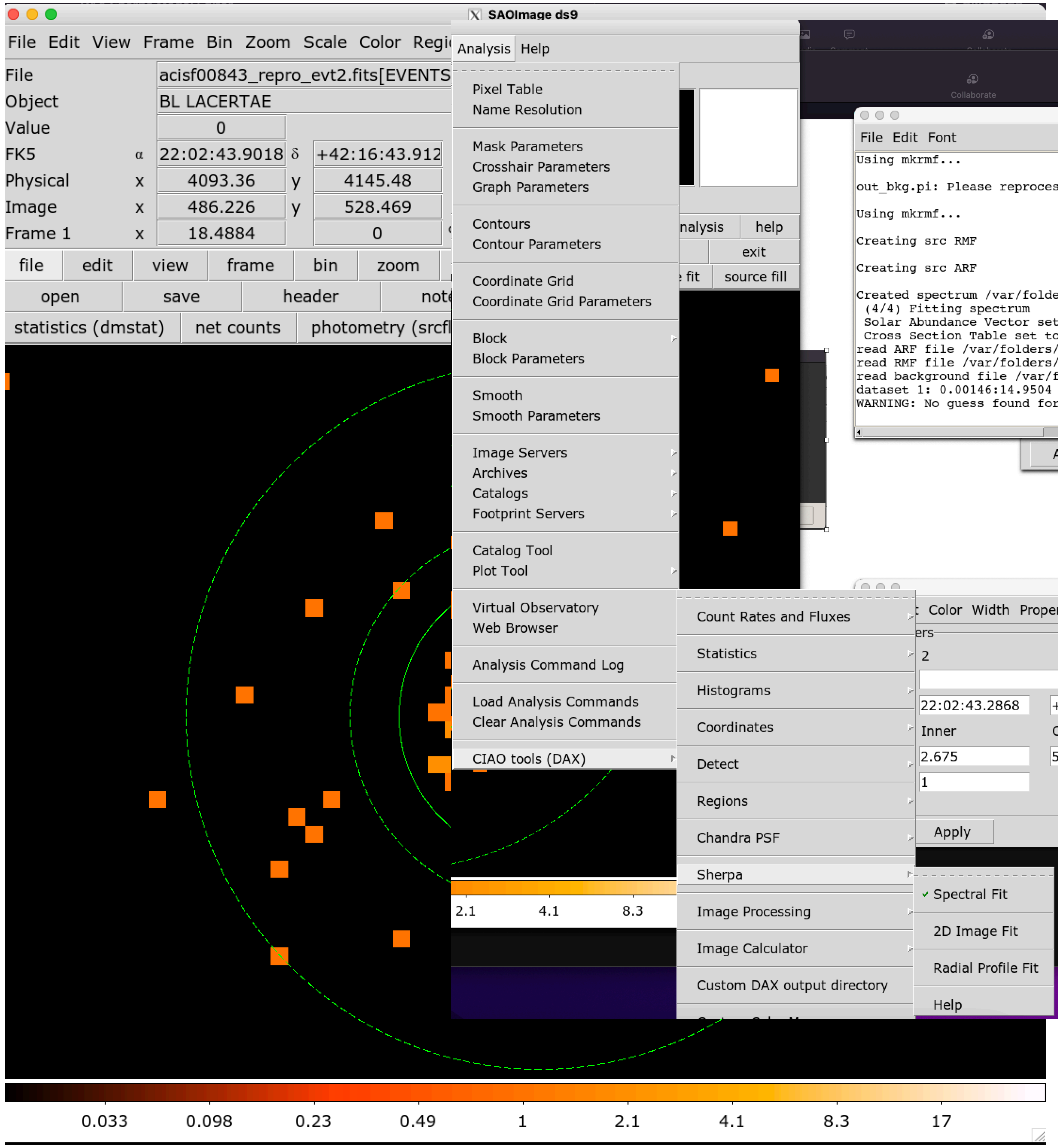
Probability Distribution

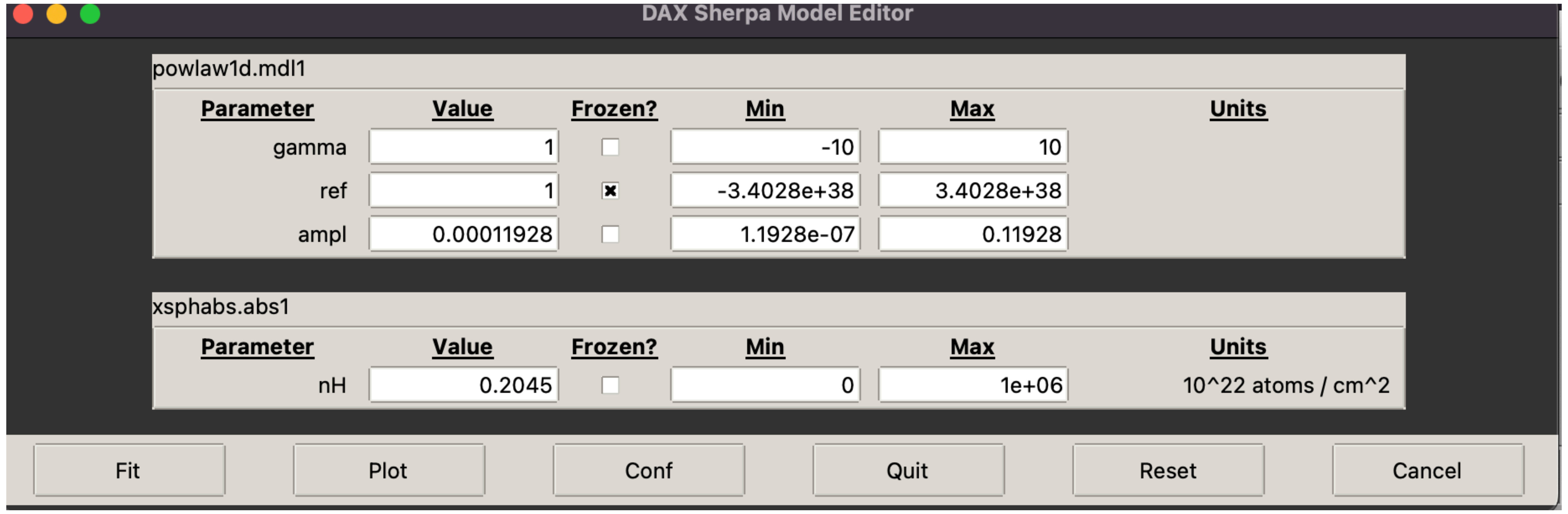






Sherpa in DS9 /DAX



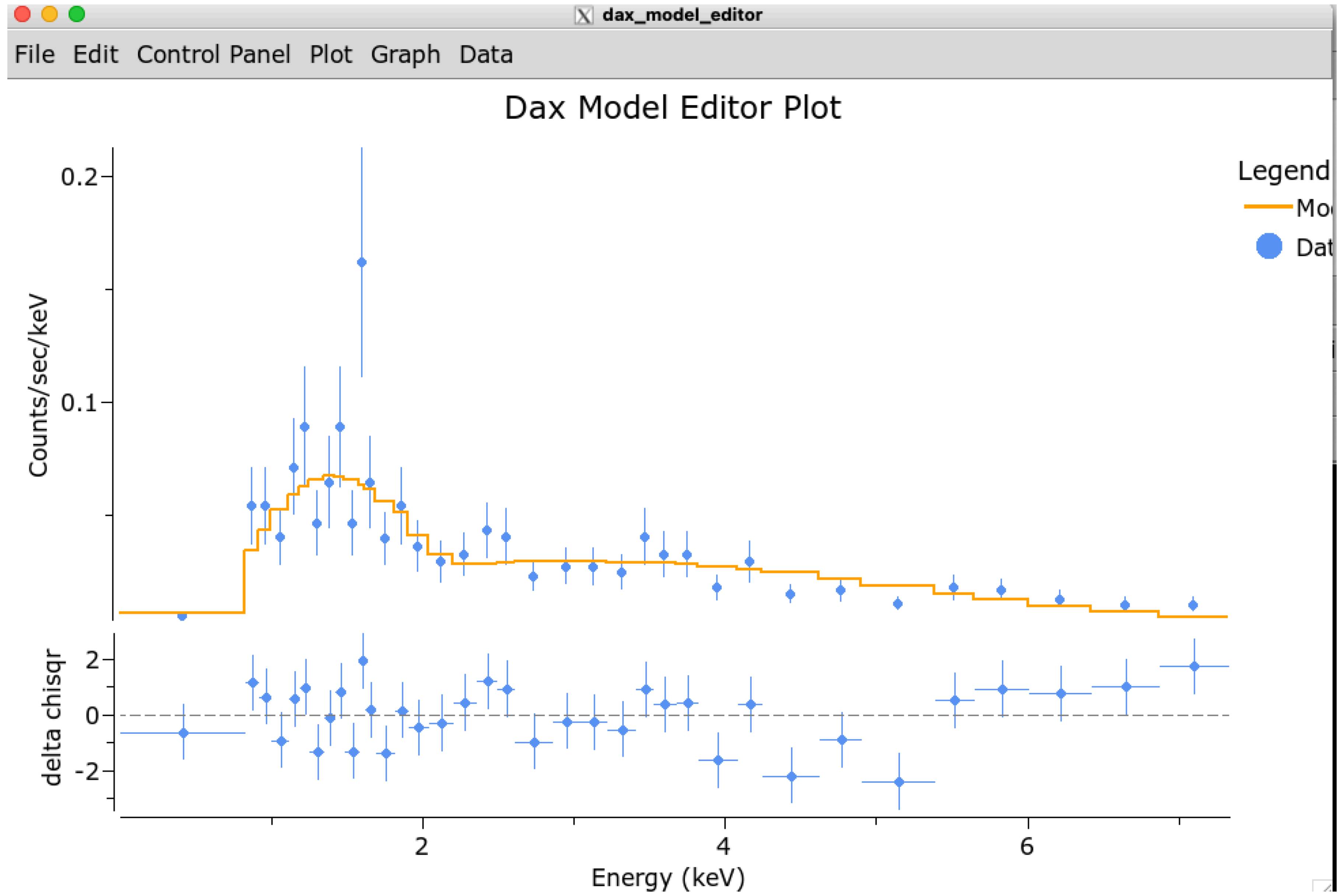


Sherpa in DS9 /DAX

Spectral Fit

File Edit Font

```
read ARF file /var/folders/1c/m66cd7qn0q73ym_z331_g5j80000gp/T//ds9_dax.aneta/sp
read RMF file /var/folders/1c/m66cd7qn0q73ym_z331_g5j80000gp/T//ds9_dax.aneta/sp
read background file /var/folders/1c/m66cd7qn0q73ym_z331_g5j80000gp/T//ds9_dax.a
dataset 1: 0.00146:14.9504 Energy (keV) (unchanged)
WARNING: No guess found for xsphabs.abs1
WARNING: data set 1 has associated backgrounds, but they have not been subtracte
Dataset               = 1
Method                = neldermead
Statistic             = cstat
Initial fit statistic = 230.373
Final fit statistic   = 40.5462 at function evaluation 271
Data points           = 36
Degrees of freedom    = 34
Probability [Q-value] = 0.203925
Reduced statistic     = 1.19253
Change in statistic   = 189.826
   mdl1.gamma     0.643789
   mdl1.ampl      0.000188472
WARNING: The displayed errorbars have been supplied with the data or calculated
```



Spectral Fit

File Edit Font

```
Change in statistic   = 189.826
   mdl1.gamma     0.643789
   mdl1.ampl      0.000188472
WARNING: The displayed errorbars have been supplied with the data or calculated
WARNING: data set 1 has associated backgrounds, but they have not been subtracte
mdl1.gamma lower bound: -0.0883185
mdl1.gamma upper bound: 0.0883185
mdl1.ampl lower bound: -1.76262e-05
mdl1.ampl upper bound: 1.87634e-05
Dataset               = 1
Confidence Method     = confidence
Iterative Fit Method  = None
Fitting Method        = neldermead
Statistic             = cstat
confidence 1-sigma (68.2689%) bounds:
   Param            Best-Fit  Lower Bound  Upper Bound
   -----            --------  -----------  -----------
   mdl1.gamma       0.643789   -0.0883185    0.0883185
   mdl1.ampl     0.000188472 -1.76262e-05  1.87634e-05
```

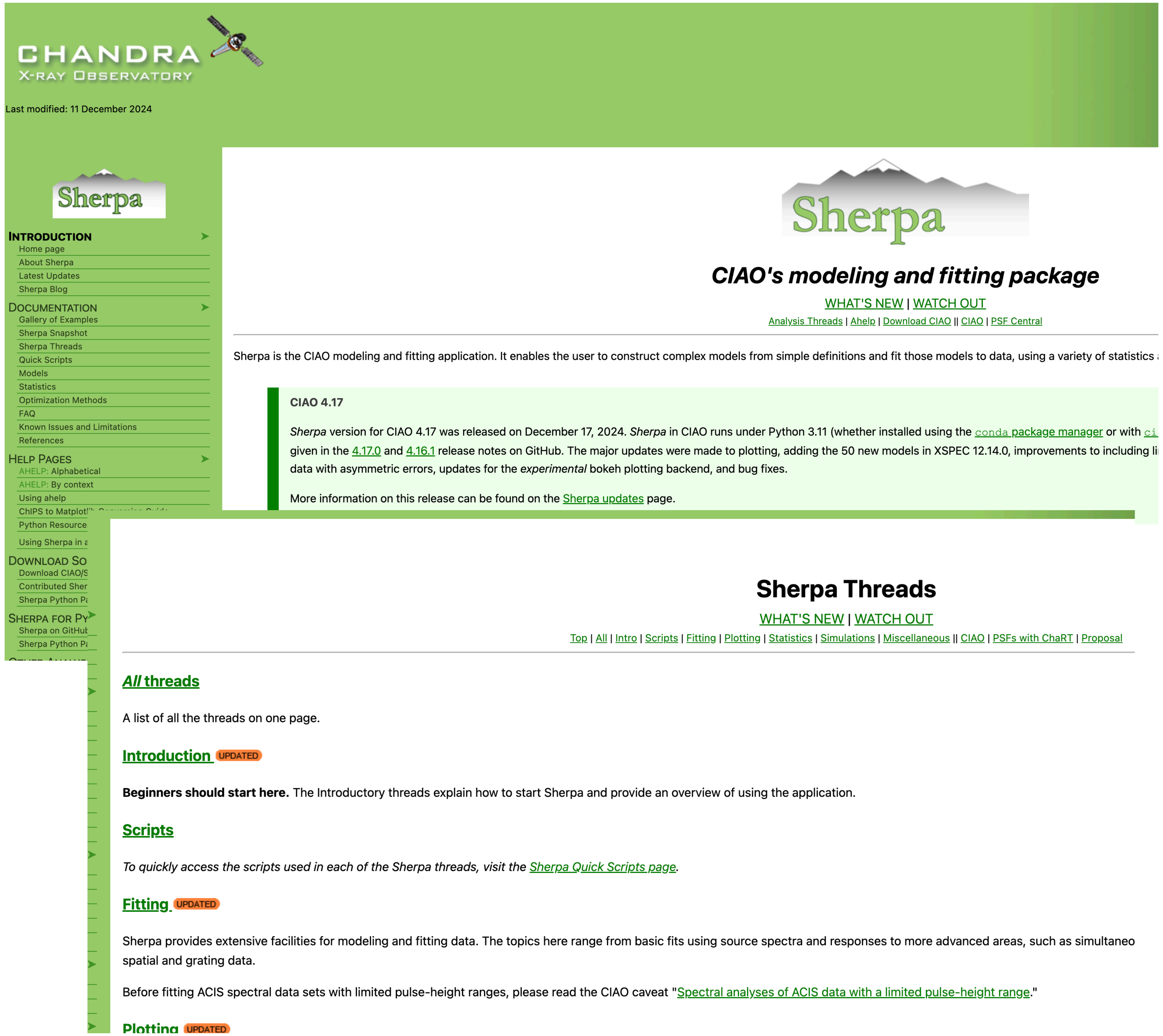
Spectral Fit

File Edit Font

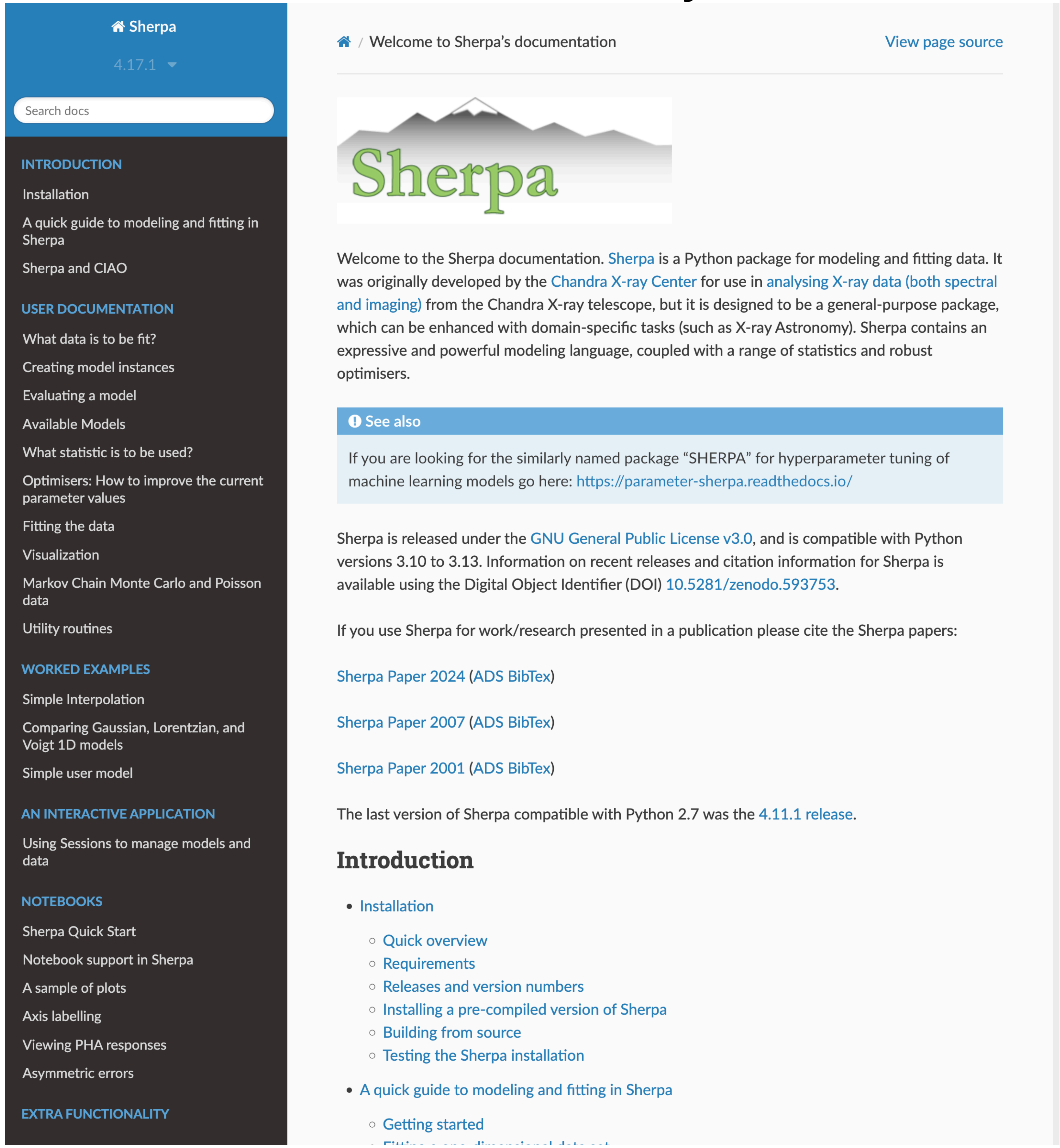
```
   mdl1.ampl     0.000188472 -1.76262e-05  1.87634e-05

(powlaw1d.mdl1 * xsphabs.abs1)
   Param        Type          Value          Min          Max      Units
   -----        ----          -----          ---          ---      -----
   mdl1.gamma   thawed     0.643789          -10           10
   mdl1.ref     frozen            1 -3.40282e+38  3.40282e+38
   mdl1.ampl    thawed  0.000188472  1.19275e-07     0.119275
   abs1.nH      frozen       0.2045            0        1e+06 10^22 atoms / cm^2

Photon Flux = 0.0005338969789672283 photon/cm^2/s

Energy Flux = 2.8428914772629414e-12 ergs/cm^2/s

WARNING: The displayed errorbars have been supplied with the data or calculated

To restore session, start sherpa and type

restore('/var/folders/1c/m66cd7qn0q73ym_z331_g5j80000gp/T//ds9_dax.aneta/specfit
```

Sherpa Documentation

CIAO



Python



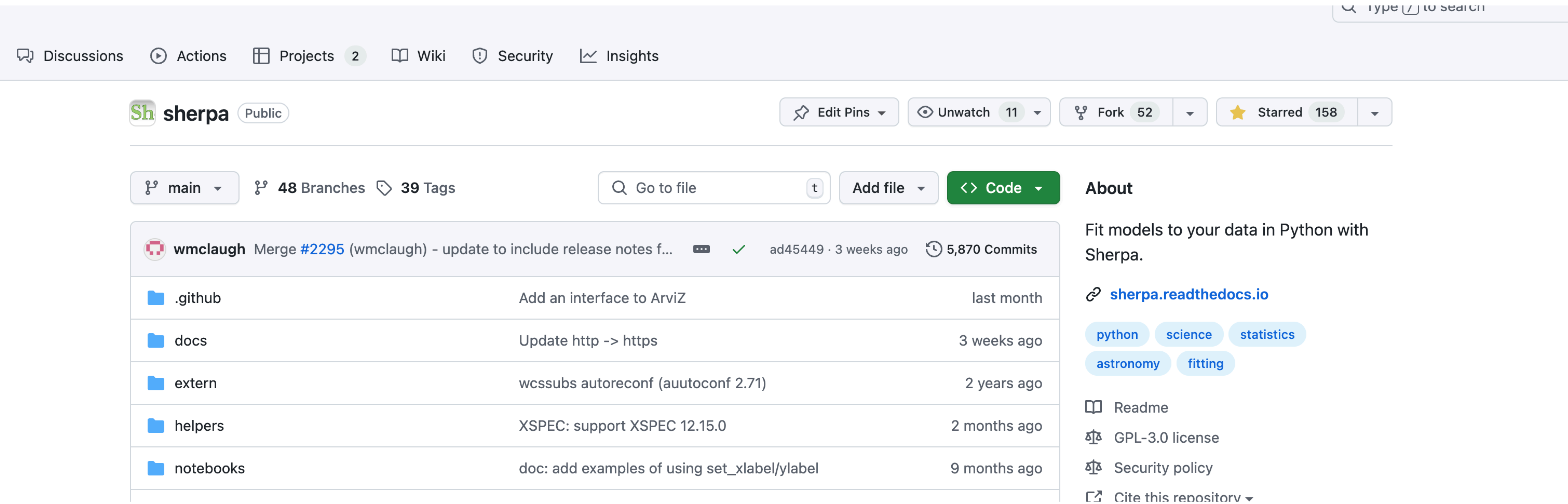
Sherpa Development on GitHub

Published May 13, 2025 | Version 4.17.1

sherpa/sherpa: Sherpa 4.17.1

Doug Burke ; Omar Laurino[1] ; wmclaugh ; Hans Moritz Günther[2] ; Marie-Terrell ; dtnguyen2 ; Aneta Siemiginowska ; Harlan Cheer ; Jamie Budynkiewicz[1] ; Tom Aldcroft[3] ; luzpaz ; Christoph Deil[4] ; Brigitta Sipőcz[5] ; nplee ; Johannes Buchner ; Axel Donath[3] ; Iva Laginja[6] ; Katrin Leinweber[7] ; Todd

https://github.com/sherpa/sherpa



Sherpa in the Scientific Python Ecosystem

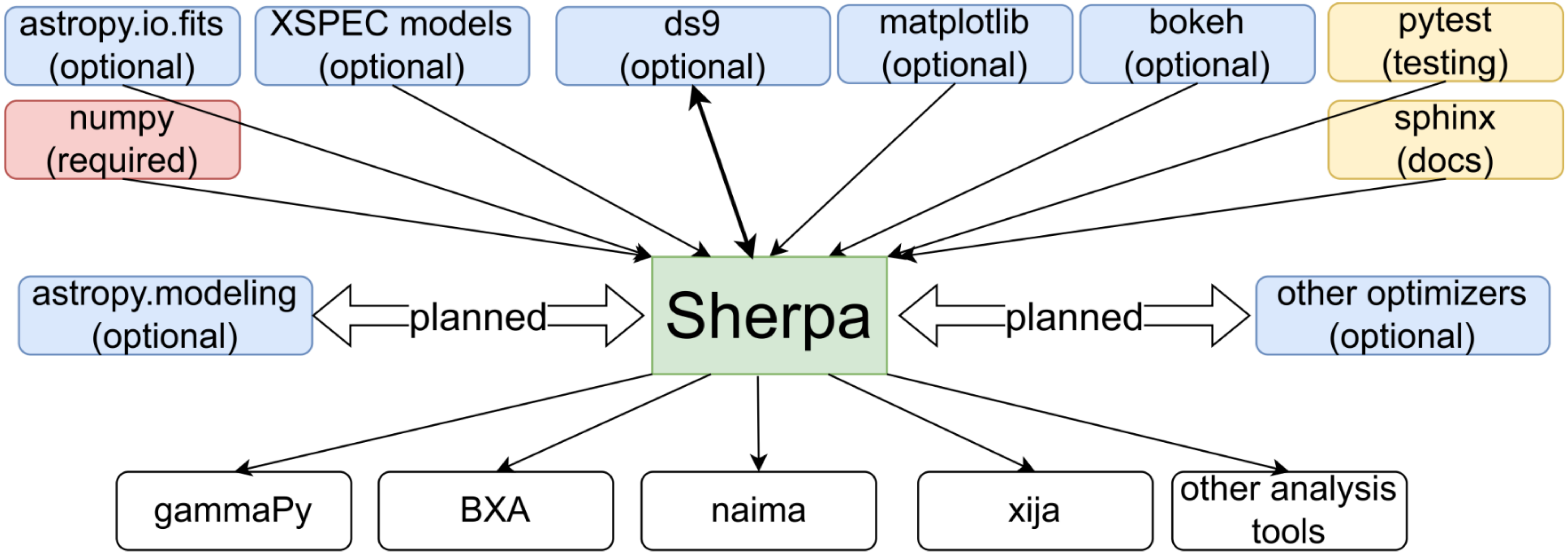


Figure 11. Sherpa's place in the scientific Python ecosystem. Sherpa requires a numeric library (red box), and interfaces with several optional libraries for I/O, external model libraries, and visualization (blue boxes). It enables the fitting and modeling in several domain-specific packages (white boxes). The two large white arrows mark future implementation of interoperability with `astropy.modeling` and external optimizers. Infrastructure packages needed for developers, but not users, are marked with yellow boxes.

Paper describing Sherpa Python and Open Source Development published in October 2024 in ApJ Supplements

The Astrophysical Journal Supplement Series, 274:43 (9pp), 2024 October https://doi.org/10.3847/1538-4365/ad7bab

Sherpa: An Open-source Python Fitting Package

Aneta Siemiginowska[1], Douglas Burke[1], Hans Moritz Günther[2], Nicholas P. Lee[1], Warren McLaughlin[1], David A. Principe[2], Harlan Cheer[1], Antonella Fruscione[1], Omar Laurino[1], Jonathan McDowell[1], and Marie Terrell[1]

[1] Center for Astrophysics | Harvard & Smithsonian, 60 Garden St., Cambridge, MA 02138, USA; asiemiginowska@cfa.harvard.edu

[2] MIT Kavli Institute for Astrophysics and Space Research, Massachusetts Institute of Technology, Cambridge, MA, USA

Received 2024 August 23; revised 2024 September 11; accepted 2024 September 15; published 2024 October 11

Abstract

We present an overview of Sherpa, an open-source Python project, and discuss its development history, broad design concepts, and capabilities. Sherpa contains powerful tools for combining parametric models into complex expressions that can be fit to data using a variety of statistics and optimization methods. It is easily extensible to include user-defined models, statistics, and optimization methods. It provides a high-level user interface for interactive data analysis, such as within a Jupyter notebook, and it can also be used as a library component, providing fitting and modeling capabilities to an application. We include a few examples of Sherpa applications to multiwavelength astronomical data.

Unified Astronomy Thesaurus concepts: Astronomy software (1855)

1. Introduction

Data-processing software developed by astronomical observatories typically generates standard data products which constitute the basis for scientific analysis. Example data products include spectra and images, data cubes, or for the

L. Barquín-González et al. 2024; H. Fan et al. 2024), the Hubble Space Telescope (HST; K. S. Green et al. 2023), the Spitzer Infrared Spectrograph infrared telescope (A. Ruiz et al. 2013), Chandra, XMM-Newton, and NuStar in X-rays, the Fermi Space Telescope for high-energy γ-rays (F. Aharonian et al. 2024), and H.E.S.S. for TeV data (H. E. S. S.

Sherpa - Summary

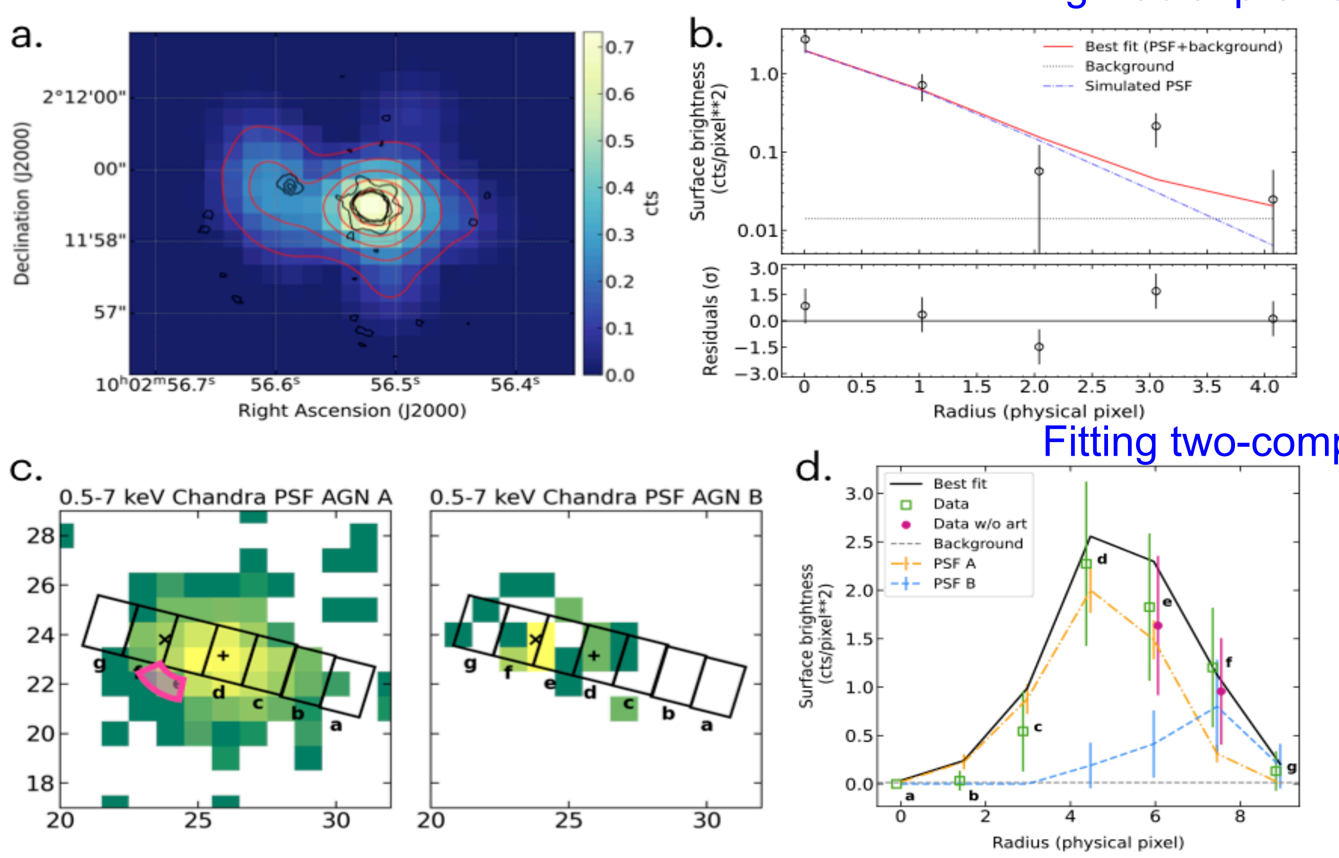
- Modeling and fitting application for Python.
- User Interface and high level functions written in Python.
- Modeling 1D/2D (N-D) data: arrays, spectra, images.
- Powerful language for building complex expressions.
- Provides a variety of statistics and optimization methods (including Bayesian analysis) .
- 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 Research

Recent Examples:

Morphology of Dual AGN in Chandra observations:
GA-NIFS: High number of dual active galactic nuclei at z ~ 3 by Perna et al (2025) doi:[10.1051/0004-6361/202453430](https://doi.org/10.1051/0004-6361/202453430)

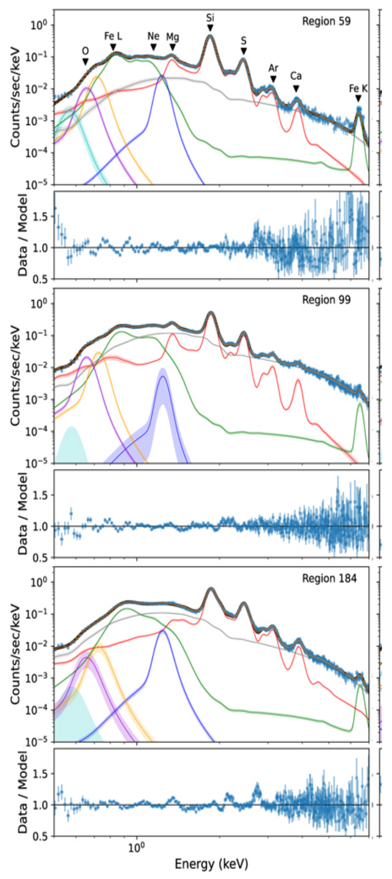
Fitting Radial profile



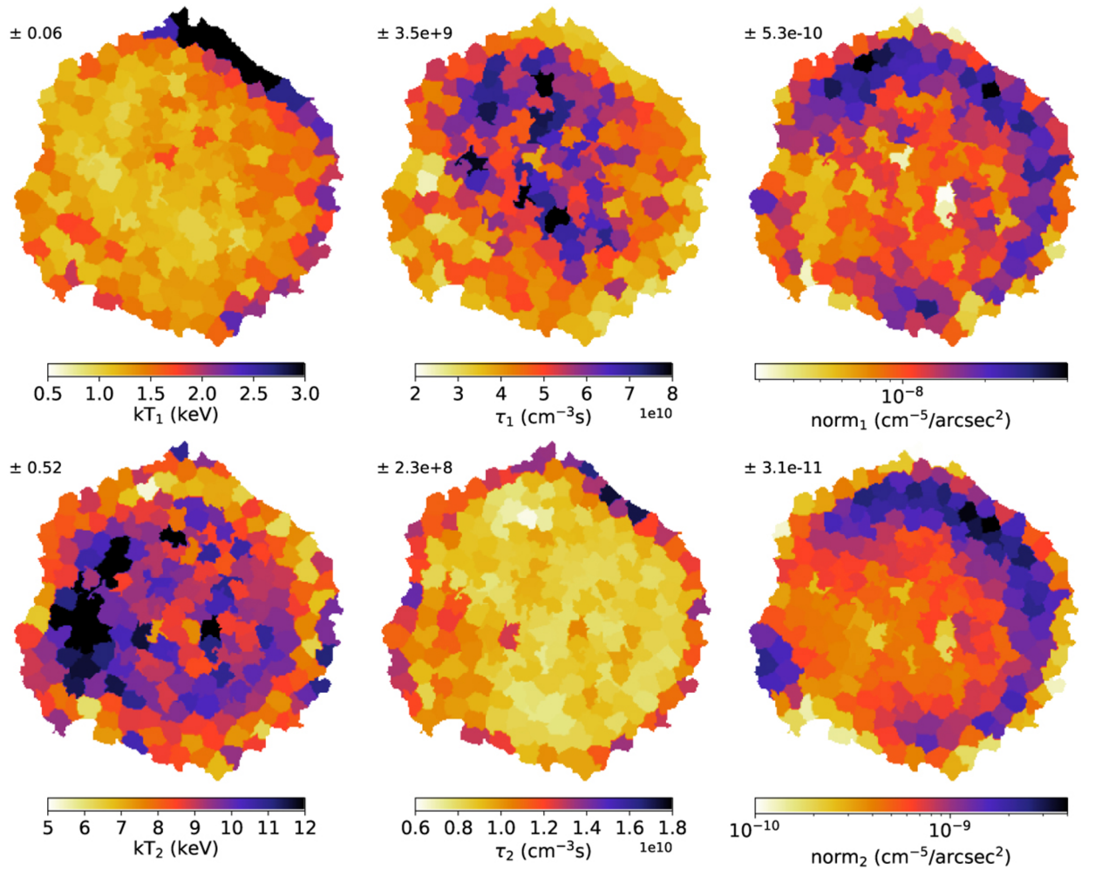
Fitting two-component profiles

COS1638: (a) Chandra X-ray image overlaid with the red X-ray contours and NIRSpec [OIII] black contours. (b) X-ray Radial Profiles; c) simulated PSF images; (d) Fitting profiles

Model fitting of the Segmented SNR: *Bayesian insights into the Tycho supernova remnant: A detailed mapping of ejecta properties* by Godinoud et al (2025). doi: [10.1051/0004-6361/202450518](https://doi.org/10.1051/0004-6361/202450518)



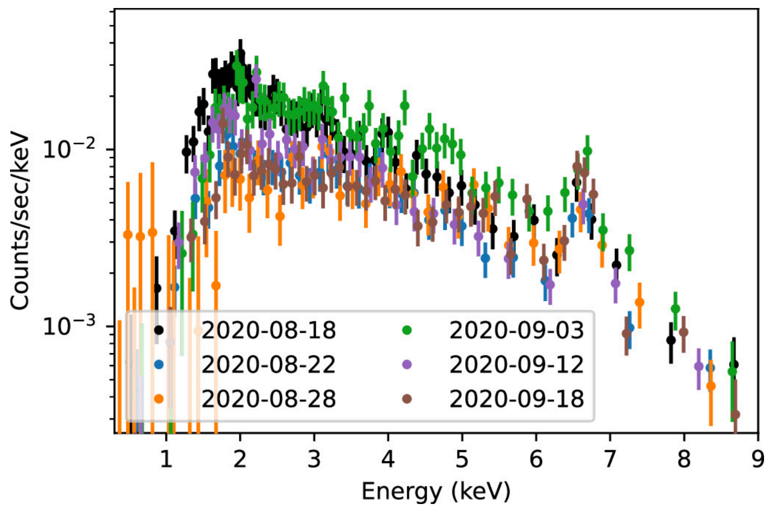
Each segment is fit with complex models using BXA and Sherpa



Maps of the model parameters for the segmented SNR in Chandra

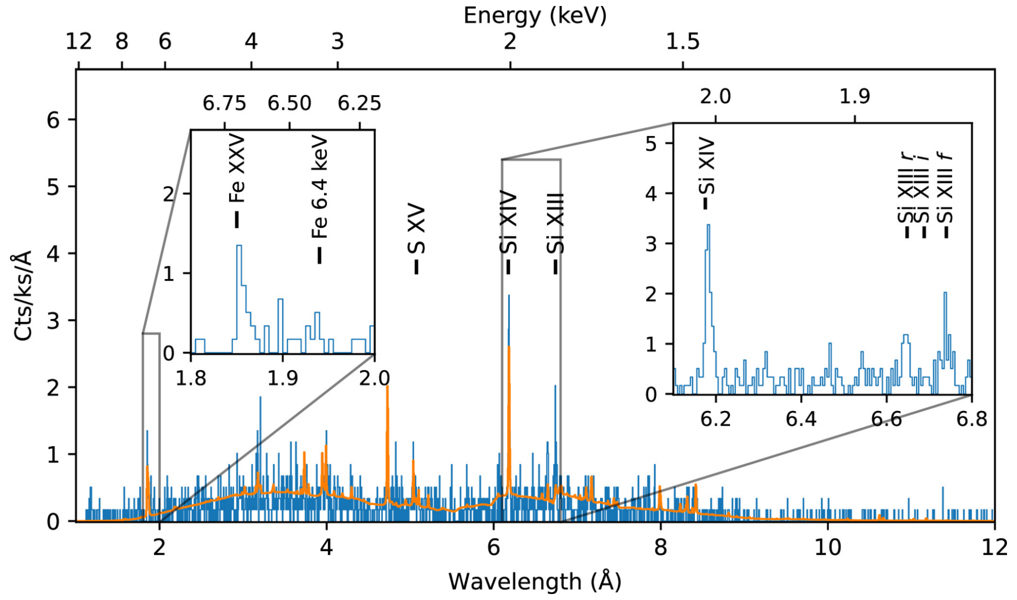
Fitting X-ray Spectra: ***Long-term X-Ray Variability on the Benchmark YSO HL Tau*** by Silverberg et al (2025). doi: [10.3847/1538-3881/adb284](https://doi.org/10.3847/1538-3881/adb284)

Fitting XMM Spectra in different epoch - looking for spectral variability



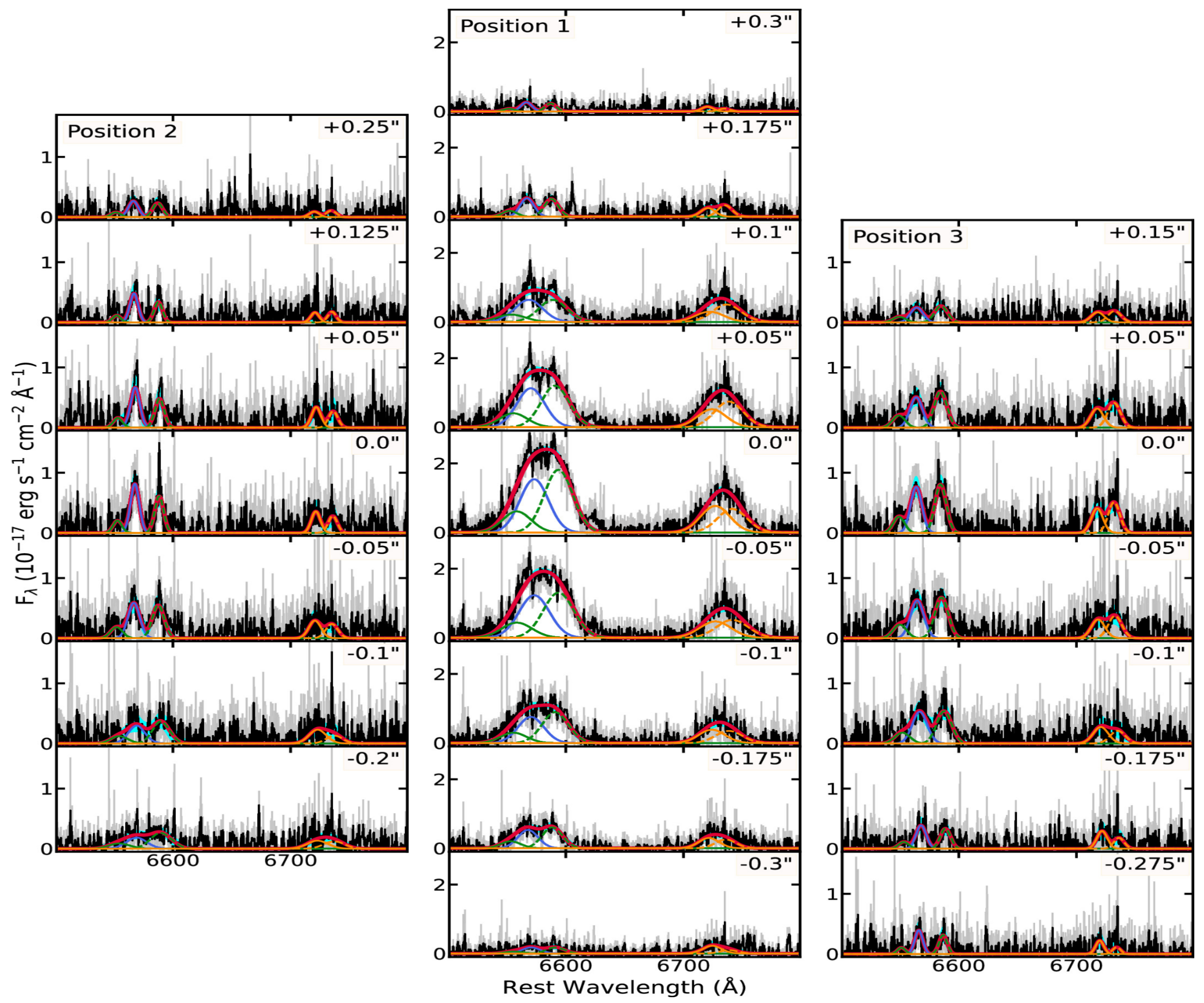
XMM EPIC Spectra in six 40 ksec time steps during 2020 campaign

Fitting High Resolution Spectra from Chandra HETG Grating



HL Tau Chandra HETG spectra (blue) overlay with the best-fit model (orange). The inserts zoom into the Fe and Si line regions.

Fitting Emission lines in the HST/STIS Spectra: *Hubble Space Telescope Observations within the Sphere of Influence of the Powerful Supermassive Black Hole in PKS 0745-191*
by Hlavacek-Larrondo et al (2025) doi: [10.3847/1538-4357/ada7ed](https://doi.org/10.3847/1538-4357/ada7ed)



Sherpa Impact on Astronomy Research

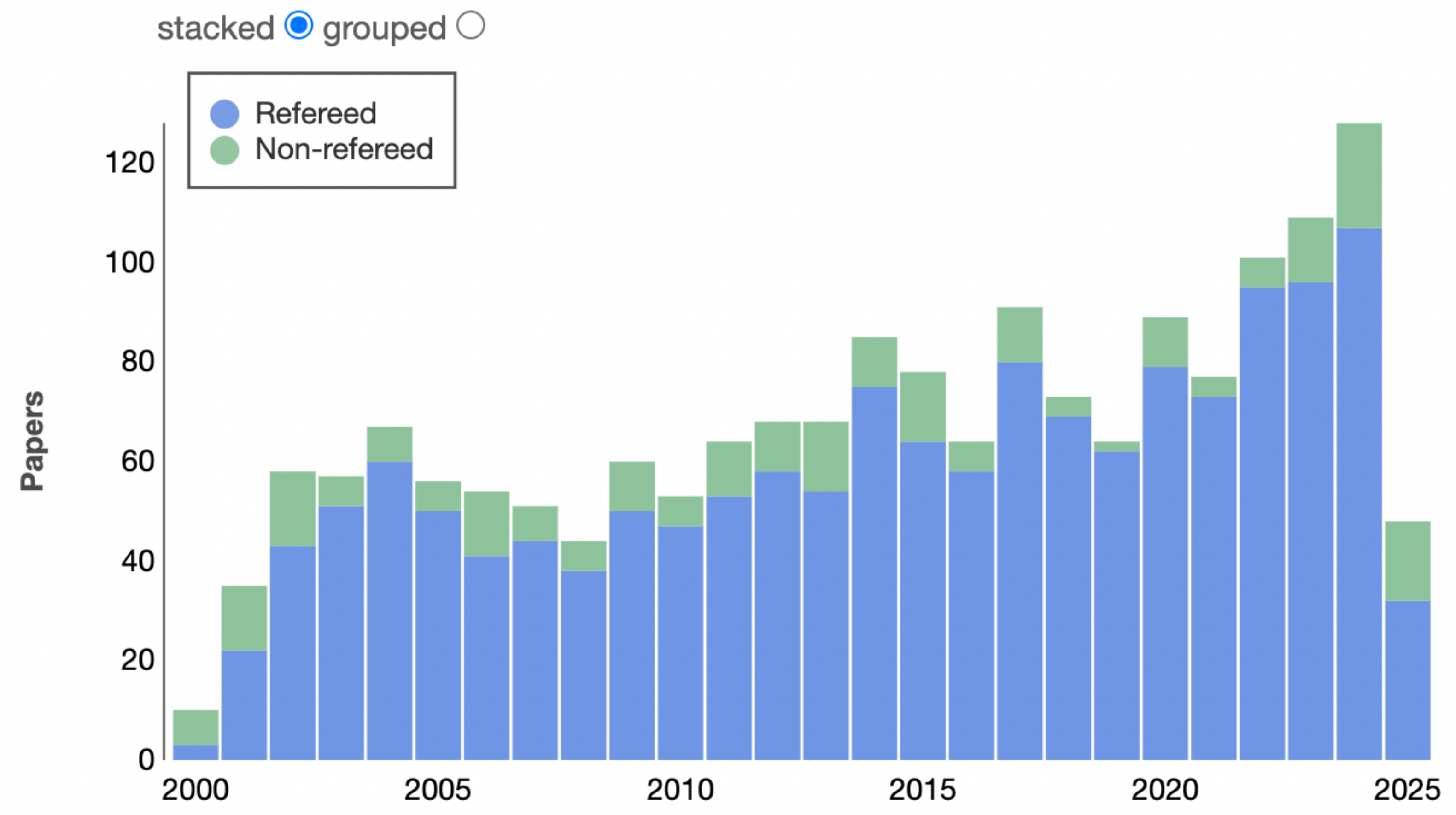
1752 publications in ApJ, AJ, MNRAS, A&A and others use Sherpa (since 2001 and including astro-ph abstracts)
https://ui.adsabs.harvard.edu/public-libraries/X6orMXwpRtSPy8x1uiiRMg

486 citations to Freeman et al 2001 SPIE paper
101 citations to Doe et al 2007 ADASS paper on Python implementation
81 citations to zenodo releases: DOI: [10.5281/zenodo.593753](https://doi.org/10.5281/zenodo.593753)
7 PhD theses listed in ADS that used Sherpa

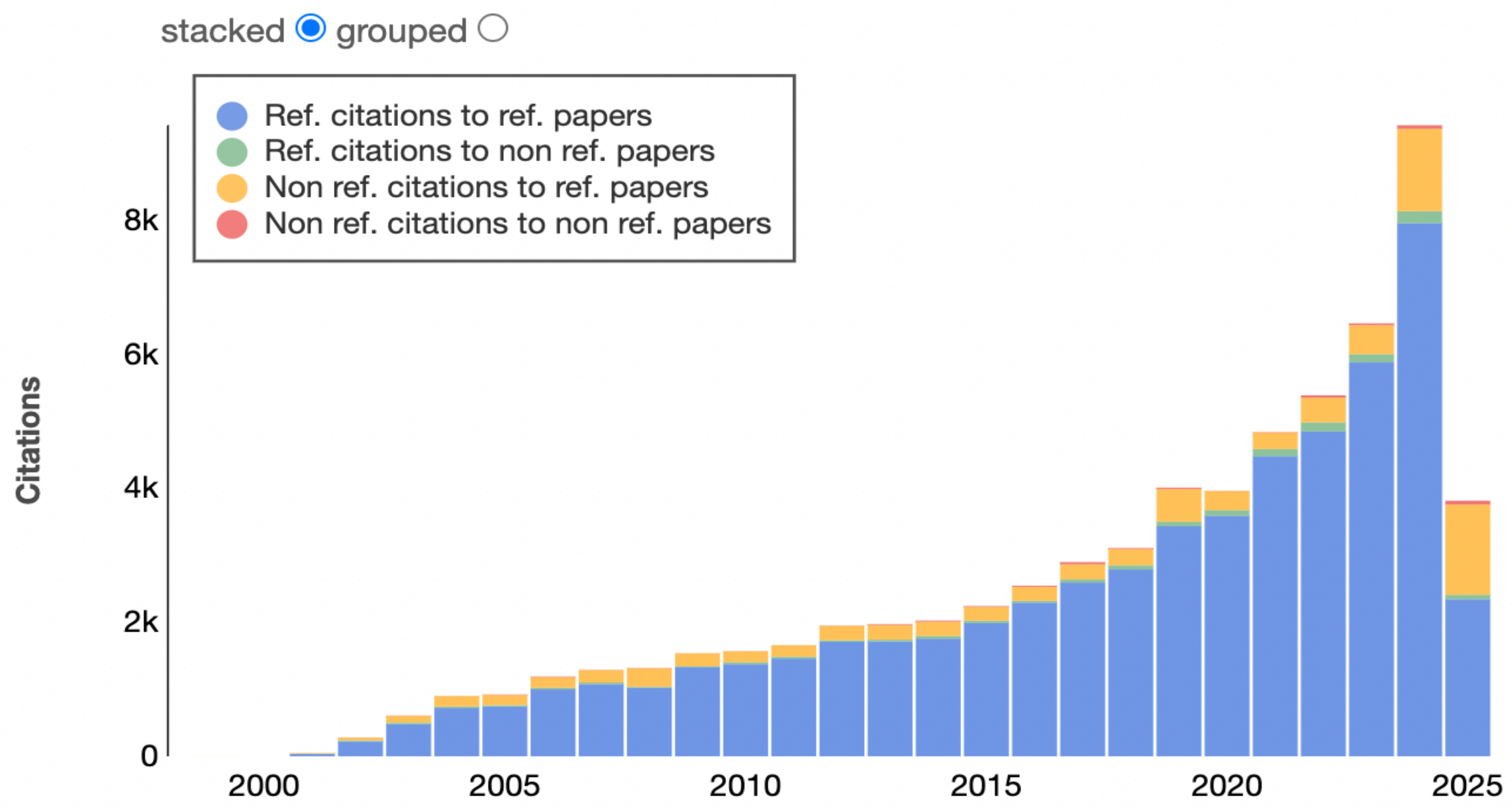
176 research papers published in 2024-2025

IMPACT 2001-2025 Statistics from ADS

Sherpa used in research



Cited Papers which used Sherpa



Confidence Limits on Parameters

Essential issue =>. after the best-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