pyblocxs
Bayesian Low-Counts X-ray Spectral Analysis in Sherpa

Aneta Siemiginowska
Harvard-Smithsonian Center for Astrophysics

Vinay Kashyap (CfA), David Van Dyk (UCI), Alanna Connors (Eureka), T.Park(Korea)+CHASC
Brian Refsdal (CfA) + CXC Data System
Outline

• Motivation and Statistical Introduction
• MCMC algorithm and Python Implementation
• Application - include calibration uncertainties
• Summary

“Analysis of Energy Spectra with Low Photon Counts via Bayesian Posterior Simulations” -
Low Counts X-ray Data

- Standard X-ray analysis in XSPEC or Sherpa
- Parameterized Forward Fitting of the data
- Assuming statistics - typically $\chi^2$
- Modified/weight $\chi^2$ to account for low counts
- Bias when the distributions are not normal.
- Poisson data - need to use the Poisson likelihood (e.g. Cash)
- MCMC methods probe the entire parameter space and do not get stuck in local minima (i.e. it can get out).
Statistical Model For Low Counts Data

Bayesian Framework

\[ p(\theta|d,I) = \frac{p(d|\theta,I)p(\theta|I)}{p(d|I)} \]

- \( \theta \) - model parameters
- \( d \) - observed data
- \( I \) - initial information

Posterior distribution

Poisson Likelihood

\[ p(d|\lambda_s,\lambda_b,I) = \frac{\exp(-\lambda_s-\lambda_b) (\lambda_s+\lambda_b)^d}{d!} \]

- \( d \) - data
- \( \lambda_s \) - source
- \( \lambda_b \) - background
Statistical Model For Low Counts Data

Model Predicted X-ray Spectra

Predicted Intensity = Instrument Response \left( \text{Source Model Intensity} \times \text{Effective Area} \right) + \text{Background}

\lambda_s(\theta_s) + \lambda_b(\theta_b)

Prior

- allows us to include a priori knowledge, e.g. range of parameters
- non-informative - e.g. flat within the range
- normal, log-normal, γ - gamma etc.
Simulations from Posterior

• Example:
  • An absorbed power law model => \( M_j(a, \Gamma, N_H) = a^*E_j^{-\Gamma} \cdot f_j(N_H) \)
  • Poisson Likelihood:

\[
\prod_{j=1}^{J} \frac{e^{-M_j} M_j^{d_j}}{d_j!}
\]

Log-likelihood
\[
\sum_j -M_j + d_j \log(M_j)
\]
(similar to Cash)

Gaussian distributions are typical prior distributions for \((a, \Gamma, N_H)\) and

**Log Posterior Distribution** is then:

\[
\sum_j [-M_j + d_j \log(M_j)] + \left[ \log G(\log(a), \mu_a, \sigma_a) + \log G(\Gamma, \mu_\Gamma, \sigma_\Gamma) \\
+ \log G(N_H, \mu_N, \sigma_N) \right]
\]
Simulations from Posterior

\[ \sum_j [-M_j + d_j \log(M_j)] + \left[ \log G(\log(a), \mu_a, \sigma_a) + \log G(\Gamma, \mu_\Gamma, \sigma_\Gamma) \right. \]

\[ \left. + \log G(N_H, \mu_N, \sigma_N) \right] \]

Simulation from the posterior distribution requires careful and efficient algorithms:

- Draw parameters from a "proposal distribution", calculate likelihood and posterior probability of the "proposed" parameter value given the observed data,
- use a Metropolis-Hastings criterion to accept or reject the "proposed" values.
pyblocxs
Python Implementation in Sherpa

• **Sherpa** is a general fitting and modeling application written in Python. It provides a library of models, statistics and optimization methods.
  
  [http://cxc.harvard.edu/contrib/sherpa/](http://cxc.harvard.edu/contrib/sherpa/)  - Python package
  
  [http://cxc.harvard.edu/sherpa/index.html](http://cxc.harvard.edu/sherpa/index.html)  - in CIAO

• It can accommodate Python code that extends the initial functionality.

• We use **Sherpa** to fit the data at the initial step and estimate the scale for setting priors and use the **Sherpa** statistics (Cash) to calculate the likelihood.
pyblocxs
Python Implementation in Sherpa

• [http://hea-www.harvard.edu/AstroStat/pyBLoCXS/index.html](http://hea-www.harvard.edu/AstroStat/pyBLoCXS/index.html) - documentation and downloads

• **pyblocxs** - samples from a multivariate t-distribution with a multivariate scale determined by Sherpa `covar()` function, at the best fit values.

• It has two samplers:
  • **Metropolis-Hastings:**
    » centered on the best fit values
  • **Metropolis-Hastings mixed with Metropolis jumping rule:**
    » centered on the current draw of parameters
    » the scale can be specified as a scalar multiple of `covar()`

• **pyblocxs:**
  ✓ Explores parameter space and summarized the full posterior or profile posterior distributions.
  ✓ Computed parameter uncertainties can include calibration errors.
  ✓ Simulates replicate data from the posterior predictive distributions.
  ✓ Tests for added spectral components by computing the Likelihood Ratio Statistic on replicate data and the ppp-value.
Running it!

Usage

The primary way to run pyBLoCXS within Sherpa is to call the function `pyblocxs.get_draws()`. First read in the spectrum:

```python
load_pha("pha.fits")
```
and define the model:

```python
set_model(xsphsbe.abs1*powlawid.pl)
```
and carry out a regular fit to define the covariance matrix:

```python
set_stat("cash")
fit()
covar()
```
then invoke pyBLoCXS with MetropolisMH as follows:

```python
import pyblocxs
pyblocxs.set_sampler("MetropolisMH")
stats, accept, params = pyblocxs.get_draws(niter=104)
```
to change to MH:

```python
pyblocxs.set_sampler("MH")
stats, accept, params = pyblocxs.get_draws(niter=104)
```
Trace of a parameter during MCMC run

3D Parameter space probed with MCMC

Cumulative distribution of a parameter
Application: Calibration Uncertainties

Chandra ACIS-S Effective Area

- Non-linear errors cannot simply add to stats errors.
- Include a draw from an ensemble of effective area curves in the simulations.

Drake et al. 2006 Proc. SPIE, 6270,49
Application: Calibration Uncertainties

Effects of ARF uncertainty on parameters

Simulations of $10^5$ counts
Sim1: $\Gamma=2$ $N_H=1e23$
Sim2: $\Gamma=1$ $N_H=1e21$

Summary

- **pyblocxs** can be used for the Poisson X-ray data.
- Provides the MCMC simulations to explore parameter space of models applied to observed data.
- Caveats:
  - Needs Sherpa
  - Tested on simple models only!
  - Parameter space can be complex for composite models with different modes.

- Available as a Sherpa Python extension at
  [http://hea-www.harvard.edu/AstroStat/pyBLoCXS/index.html](http://hea-www.harvard.edu/AstroStat/pyBLoCXS/index.html)

**Focus Demo** at 3.30pm by Brian Refsdal on
Advanced Python scripting using Sherpa

Check **CIAO booth**, talk to developers and get personal demos of the software!