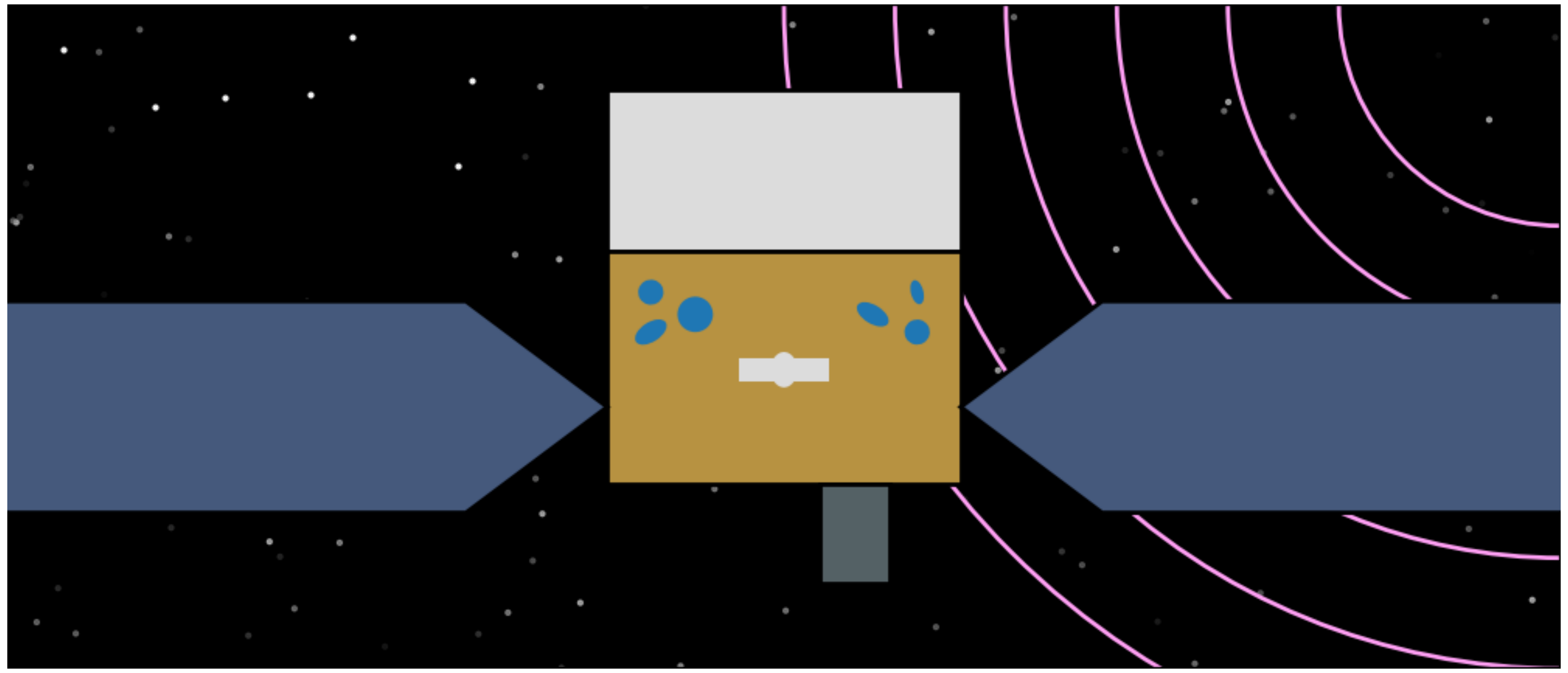
The Fermi GBM Data Tools



https://fermi.gsfc.nasa.gov/ssc/data/analysis/gbm/

Adam Goldstein (USRA)
Bill Cleveland (USRA)
Dan Kocevski (NASA/MSFC)

What is it?

- Python API for GBM Data
  - Interface to GBM Data
    - CSPEC/CTIME, TTE, RSP(2)
    - POSHIST, TRIGDAT
    - Localization HEALPix
  - Trigger **and** continuous data
  - The aim is to have a sufficiently **high-level** part of the API so that it is easily accessible to many, but also **lower-level** part of the API for expert users

What Can I Do With It?

- Lots of stuff!
  - Reduce and Analyze data
    - Binning algorithms for pre-binned and unbinned data
    - Background fitting/estimation for pre-binned and unbinned data
  - Observing conditions — Source visibility, GTIs, detector angles, etc
  - Export of PHAII time-series data to PHA/BAK data
  - Spectral analysis
  - Simulations
  - Wide range of visualizations
  - Interface to HEASARC FTP archive and Browse Catalogs

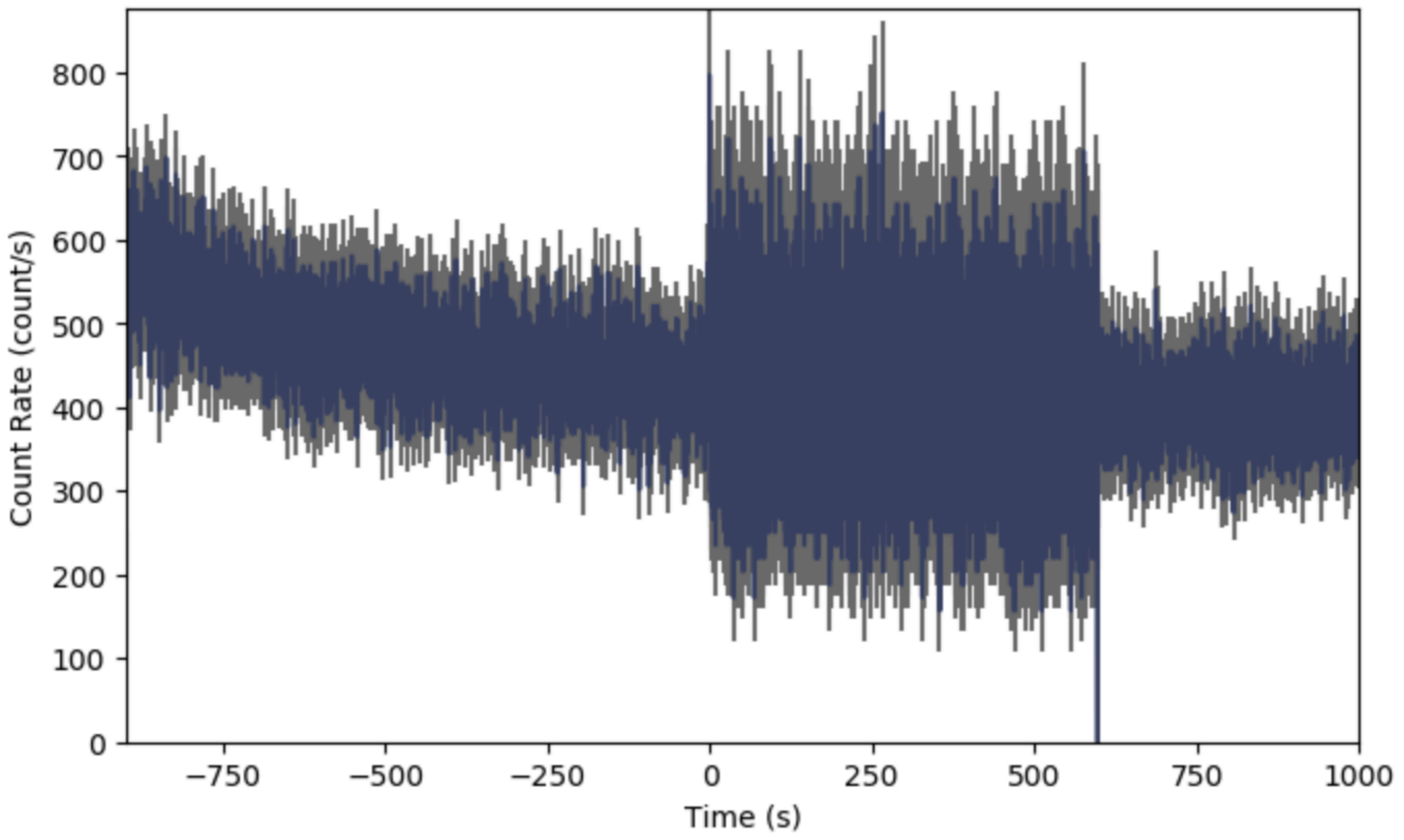
High-Level API - Lightcurves

Read a file and convert to lightcurve

```python
# import the CTIME and CSPEC data classes
from gbm.data import Ctime, Cspec

# read a ctime file
ctime = Ctime.open(test_data_dir+'/glg_ctime_nb_bn120415958_v00.pha')
# integrate over 50-300 keV
lightcurve = ctime.to_lightcurve(energy_range=(50.0, 300.0))
```

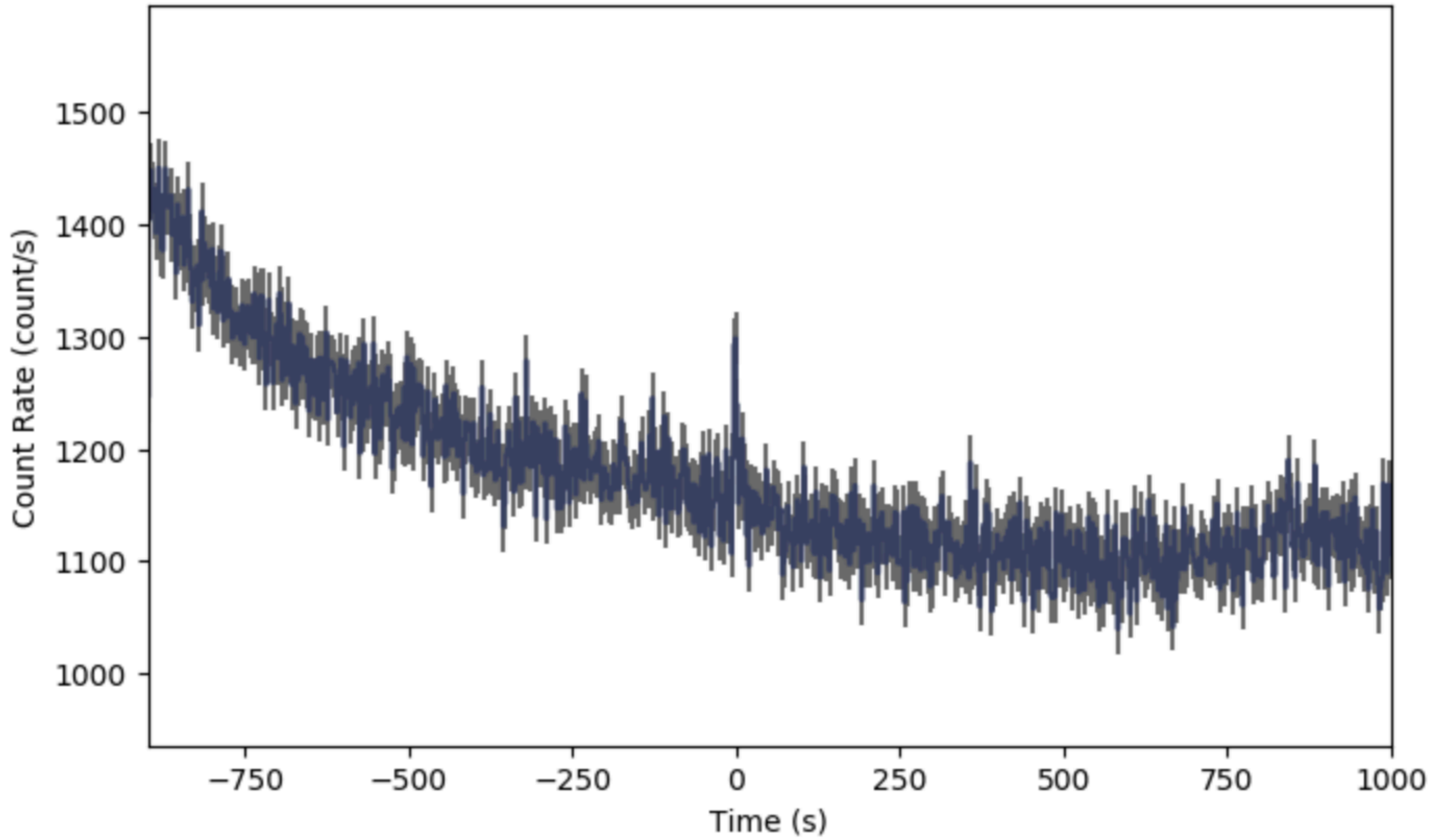
Plot it!

```python
from gbm.plot.lightcurve import Lightcurve
lcplot = Lightcurve(data=lightcurve)
plt.show()
```



Rebin it!

```python
# the data binning module
from gbm.binning.binned import rebin_by_time

# rebin the data to 2048 ms resolution
rebinned_ctime = ctime.rebin_time(rebin_by_time, 2.048)

# and replot
lcplot = Lightcurve(data=rebinned_ctime.to_lightcurve())
```



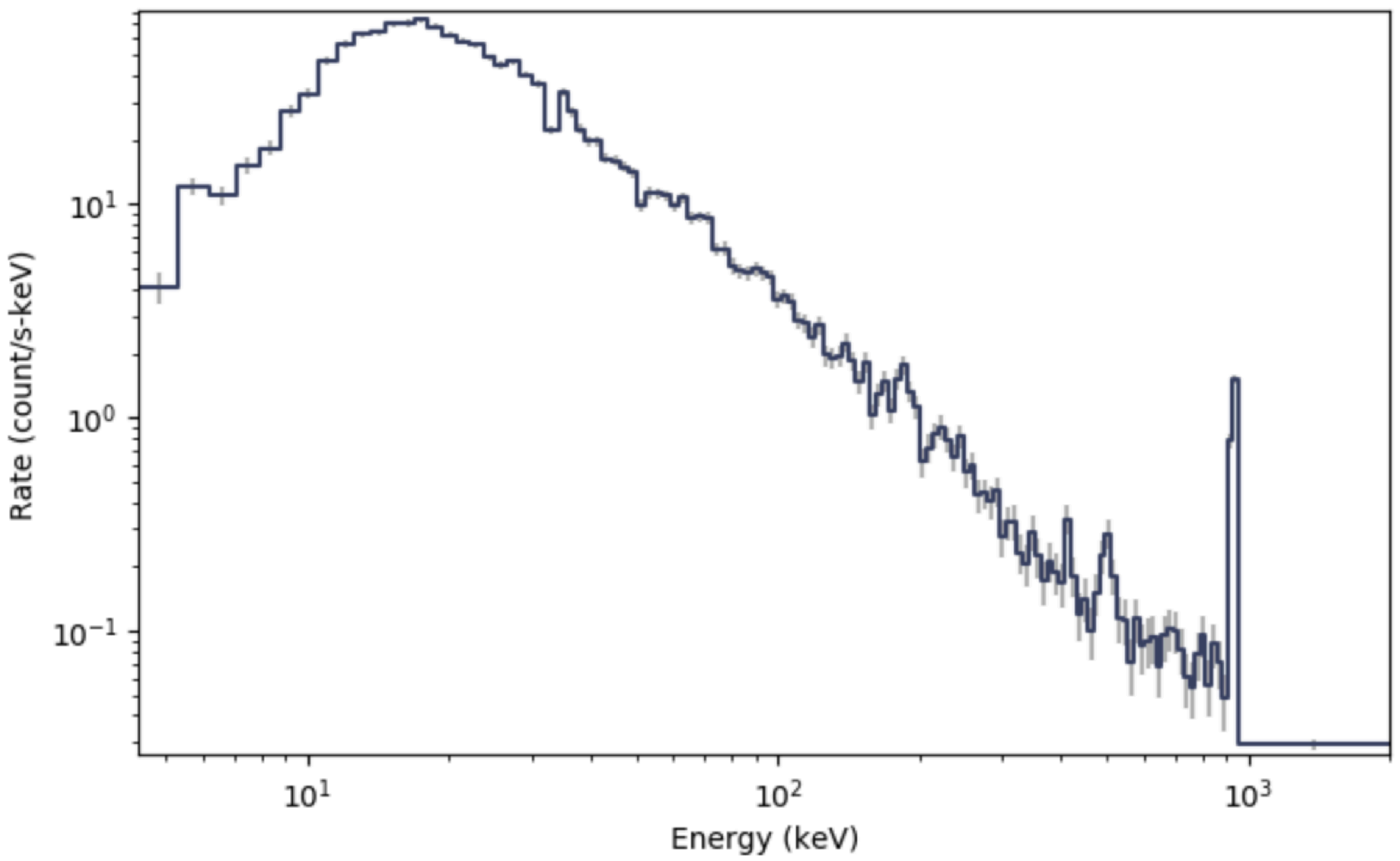
High-Level API - Spectra

Read a TTE file

```
# import the TTE data class
from gbm.data import TTE

# read a tte file
tte = TTE.open(test_data_dir+'/glg_tte_n9_bn090131090_v00.fit')
```

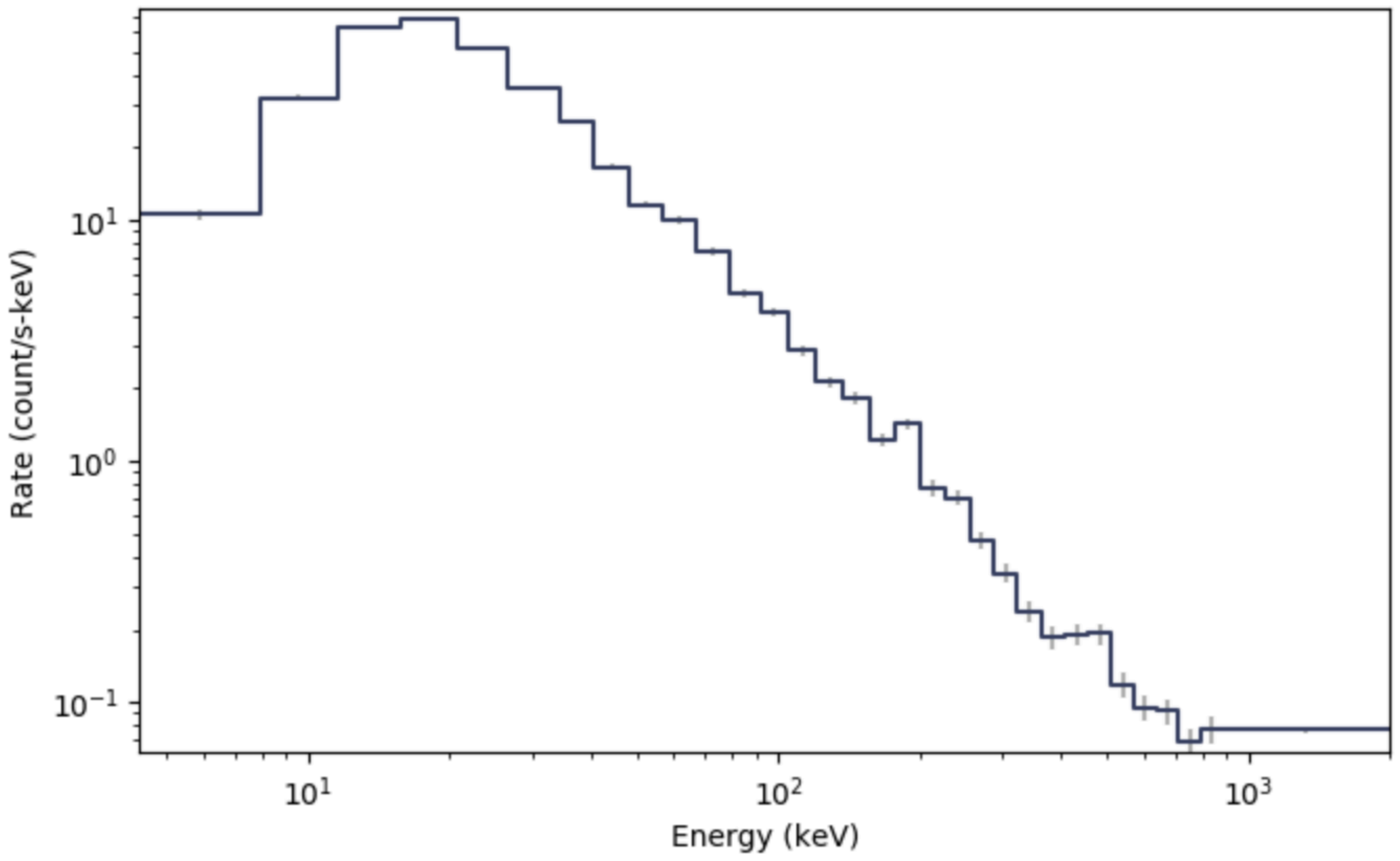
Convert to spectrum and plot

```
# integrate over time from 0-10 s
spectrum = tte.to_spectrum(time_range=(0.0, 10.0))

specplot = Spectrum(data=spectrum)
plt.show()
```



Rebin

```
from gbm.binning.binned import combine_by_factor

# rebin the count spectrum by a factor of 4
rebinned_energy = tte.rebin_energy(combine_by_factor, 4)

rebinned_spectrum = rebinned_energy.to_spectrum(time_range=(0.0, 10.0))
specplot = Spectrum(data=rebinned_spectrum)
```



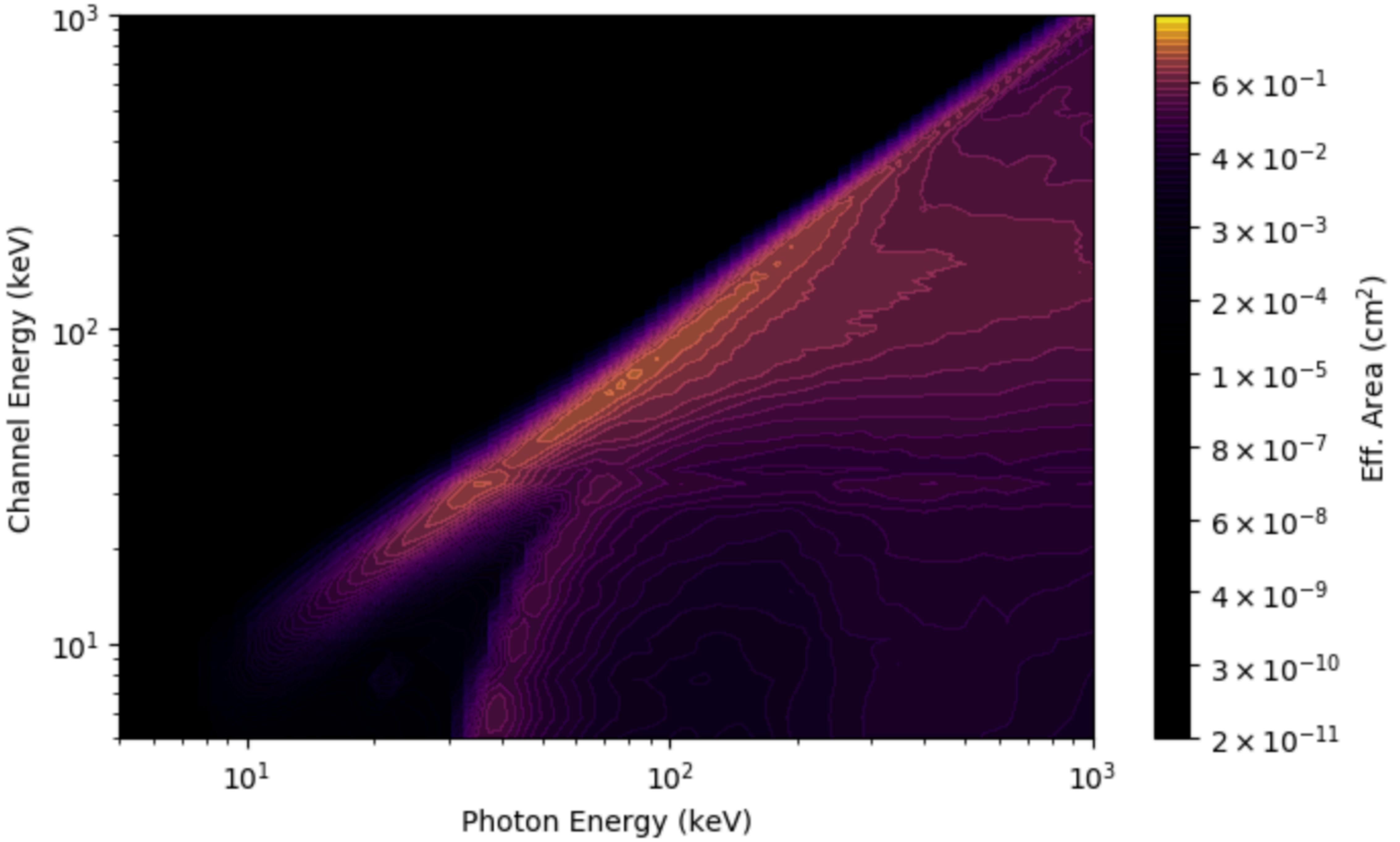
Responses

Read a Response file

```
from gbm.data import RSP
rsp = RSP.open(test_data_dir+'/glg_cspec_n4_bn120415958_v00.rsp2')
```

Plot the DRM

```
from gbm.plot import ResponseMatrix

rsp_plot = ResponseMatrix()
rsp_plot.set_response(rsp, color='plasma') # a pretty color gradient
rsp_plot.xlim = (5.0, 1000.0)
rsp_plot.ylim = (5.0, 1000.0)
```



Fold a photon model through the response

```
# a power-law function.
# params is a list of parameters: (amplitude, index)
def powerlaw(params, energies):
    return params[0]*(energies/100.0)**params[1]

# fold a power law with amplitude 0.1 and index -2.0 through the DRM at trigger time
rsp.fold_spectrum(powerlaw, (0.1, -2.0), atime=0.0)
```

```
array([ 2.04555274,  2.41331594,  2.0801156 ,  1.56281085,  1.57124845,
        1.95612002,  2.18619054,  2.68707728,  3.09026986,  3.87129313,
        4.53683755,  5.03903868,  5.73969901,  6.52557411,  8.28145565,
        8.7270274 ,  9.29967452,  9.82871379, 10.27252461, 10.64788631,
       12.61452885, 13.15436336, 14.44362474, 11.80630877,  9.50993977,
```

Observing Conditions

Read a position history file

```
from gbm.data.poshist import PosHist

# open a poshist file
poshist = PosHist.open(test_data_dir+'/glg_poshist_all_170101_v00.fit')
```

Is a position visible at some time?

```
t0 = 504975500.0
# the position of our source
ra = 324.3
dec = -20.8
poshist.location_visible(ra, dec, t0)
```

array([True])

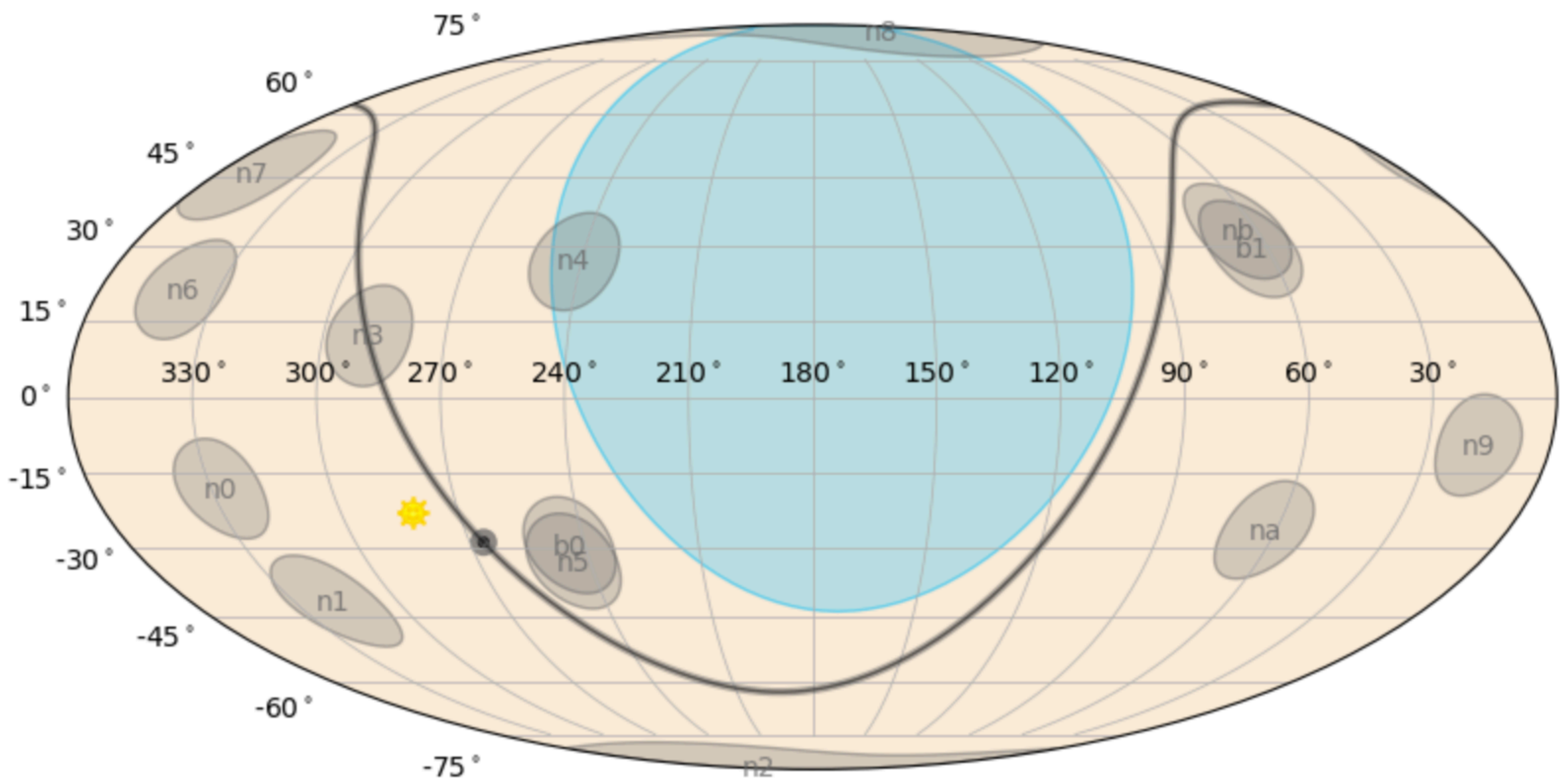
Angle of the position to detector n0:

```
poshist.detector_angle(ra, dec, 'n0', t0)
```

4.2721980564266975

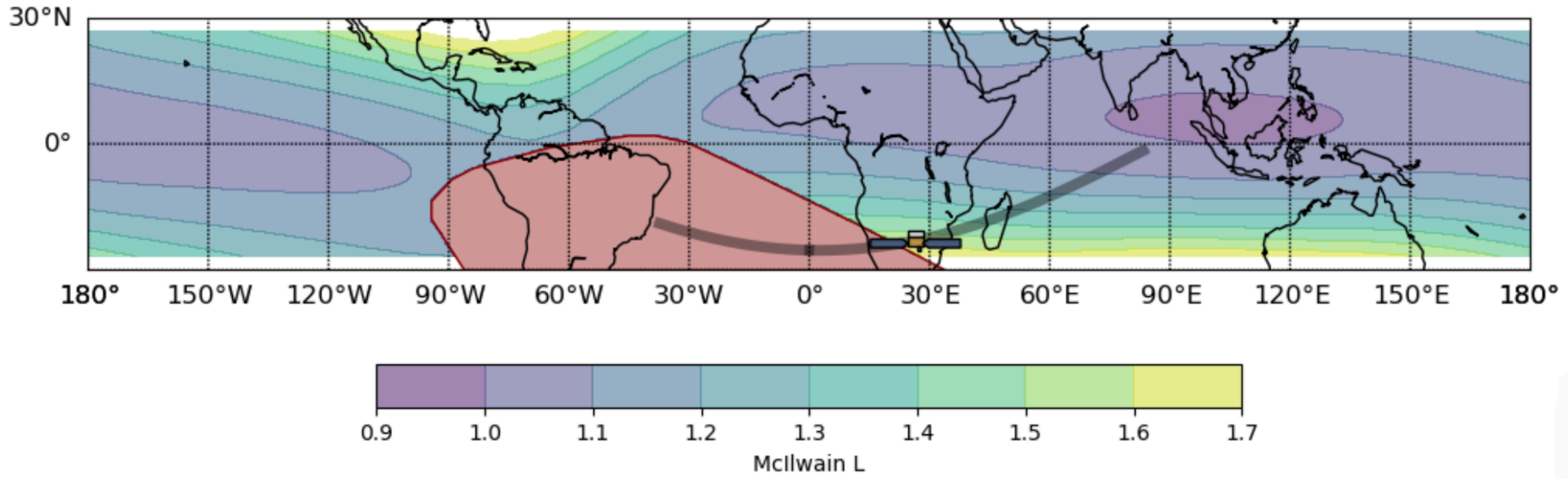
Plot the detector pointing

```
from gbm.plot.skyplot import SkyPlot, FermiSkyPlot

# initialize plot
skyplot = SkyPlot()
# plot the orientation of the detectors and Earth blockage at our time of interest
skyplot.add_poshist(poshist, trigtime=t0)
plt.show()
```



Plot the orbital position

```
from gbm.plot.earthplot import EarthPlot

# initialize plot
earthplot = EarthPlot()

# let's how the orbital path for +/-1000 s around our t0
earthplot.add_poshist(poshist, trigtime=t0, time_range=(t0-1000.0, t0+1000.0))
```



Localizations

Read a HEALPix localization file

```
from gbm.data.localization import GbmHealPix

# open a GBM localization
loc = GbmHealPix.open(test_data_dir+'/glg_healpix_all_bn190915240_v00.fit')
```

The confidence level at a point

```
loc.confidence(40.0, 4.0)
```

0.865783539232832

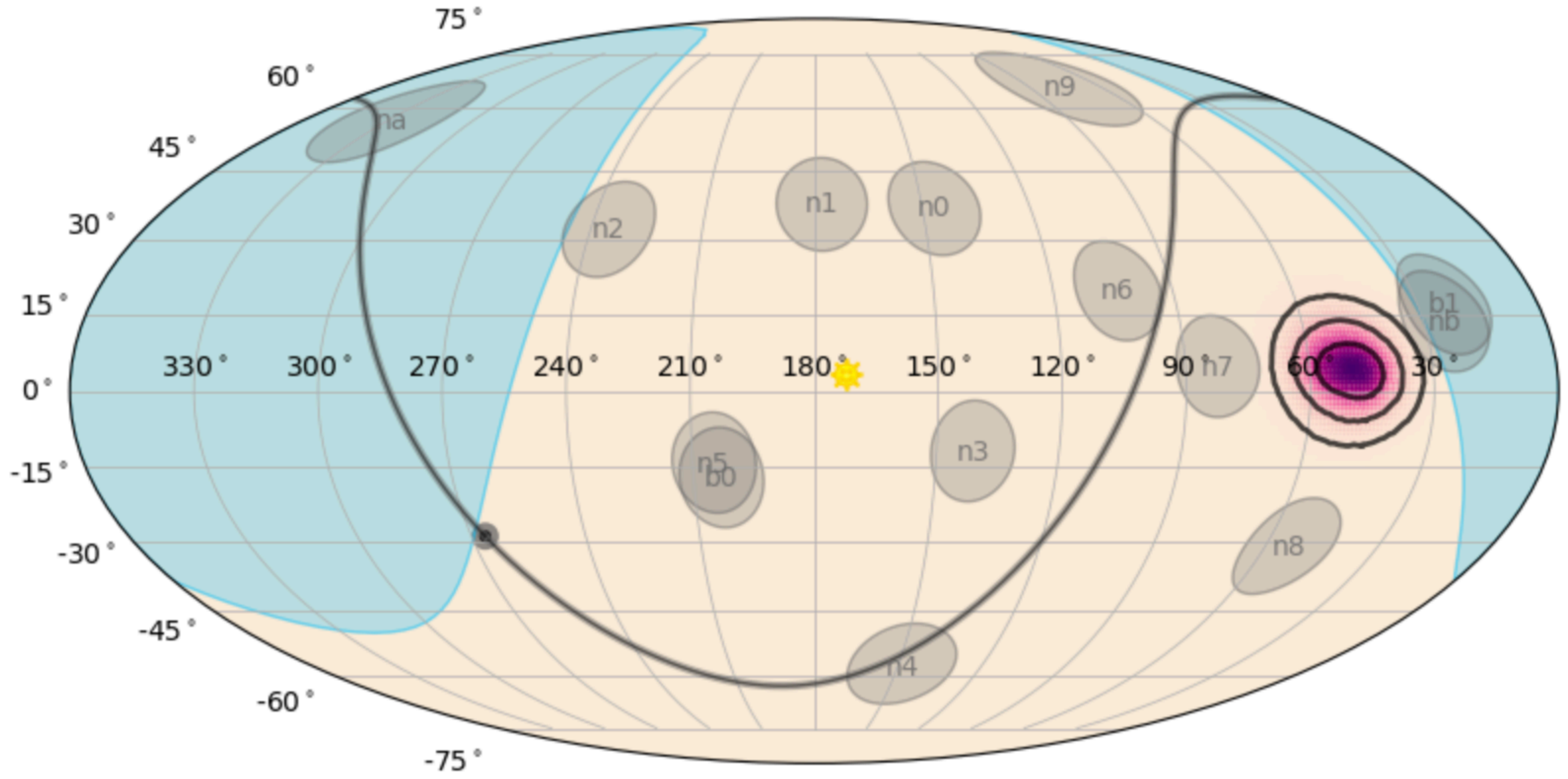
Area of the 90% conf. region

```
loc.area(0.9) # 90% confidence in units of sq. degrees
```

281.1633711457409

Plot the localization

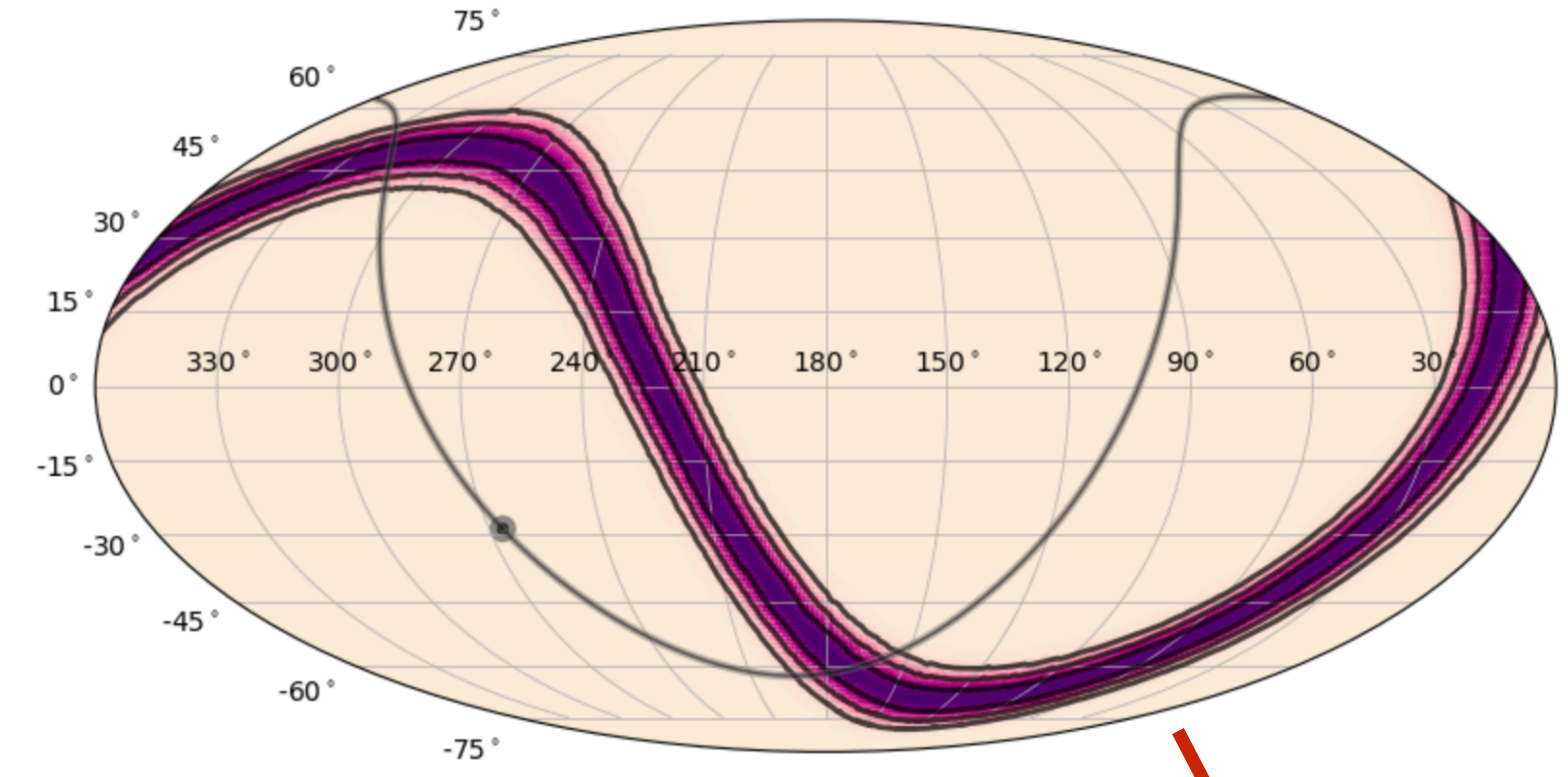
```
from gbm.plot.skyplot import SkyPlot

# initialize
skyplot = SkyPlot()

# add our HEALPix object
skyplot.add_healpix(loc)
```



Custom HEALPix Maps

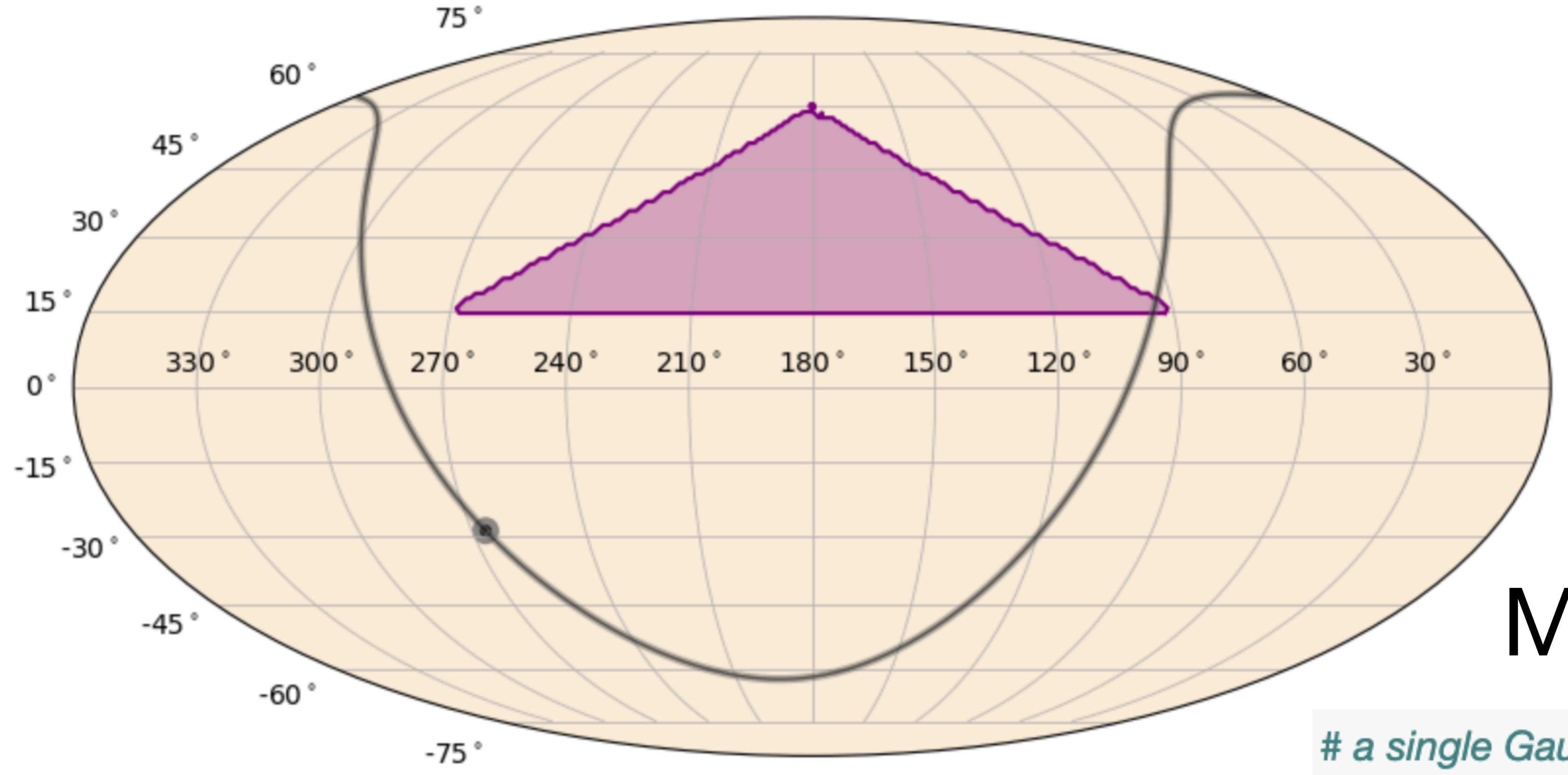
Annulus on the sky (e.g. IPN)

```
annulus_map = HealPix.from_annulus(300.0, -30, 80.0, 3.0, nside=128)

skyplot = SkyPlot()
skyplot.add_healpix(annulus_map, earth=False, sun=False, detectors=[])
```



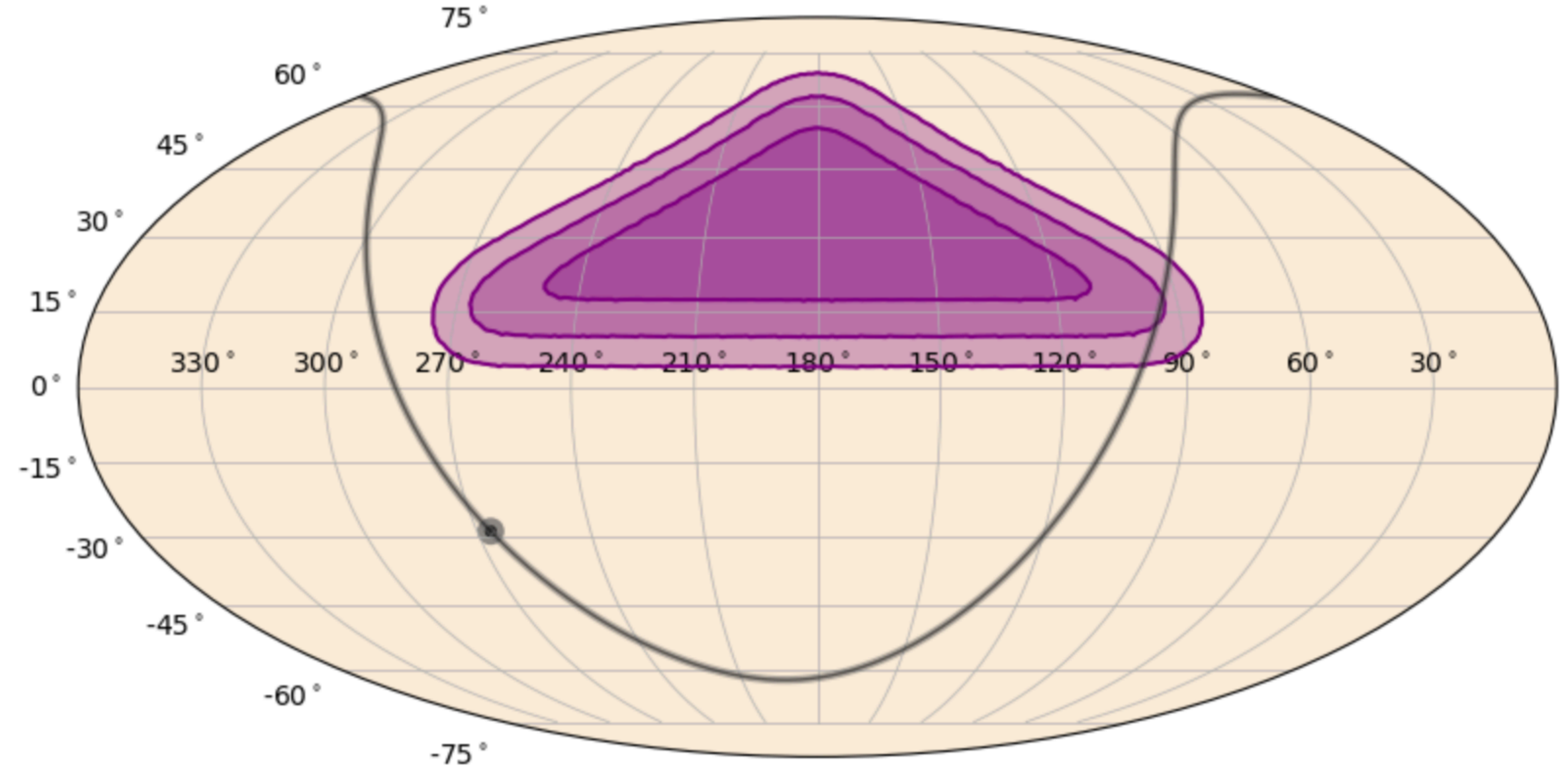
List of vertices

```
ra_pts = [270.0, 180.0, 90.0]
dec_pts = [15.0, 60.0, 15.0]

verts_map = HealPix.from_vertices(ra_pts, dec_pts, nside=128)
skyplot = SkyPlot()
skyplot.add_healpix(verts_map, earth=False, sun=False, detectors=[], gradient=False, clevels=[0.999])
```



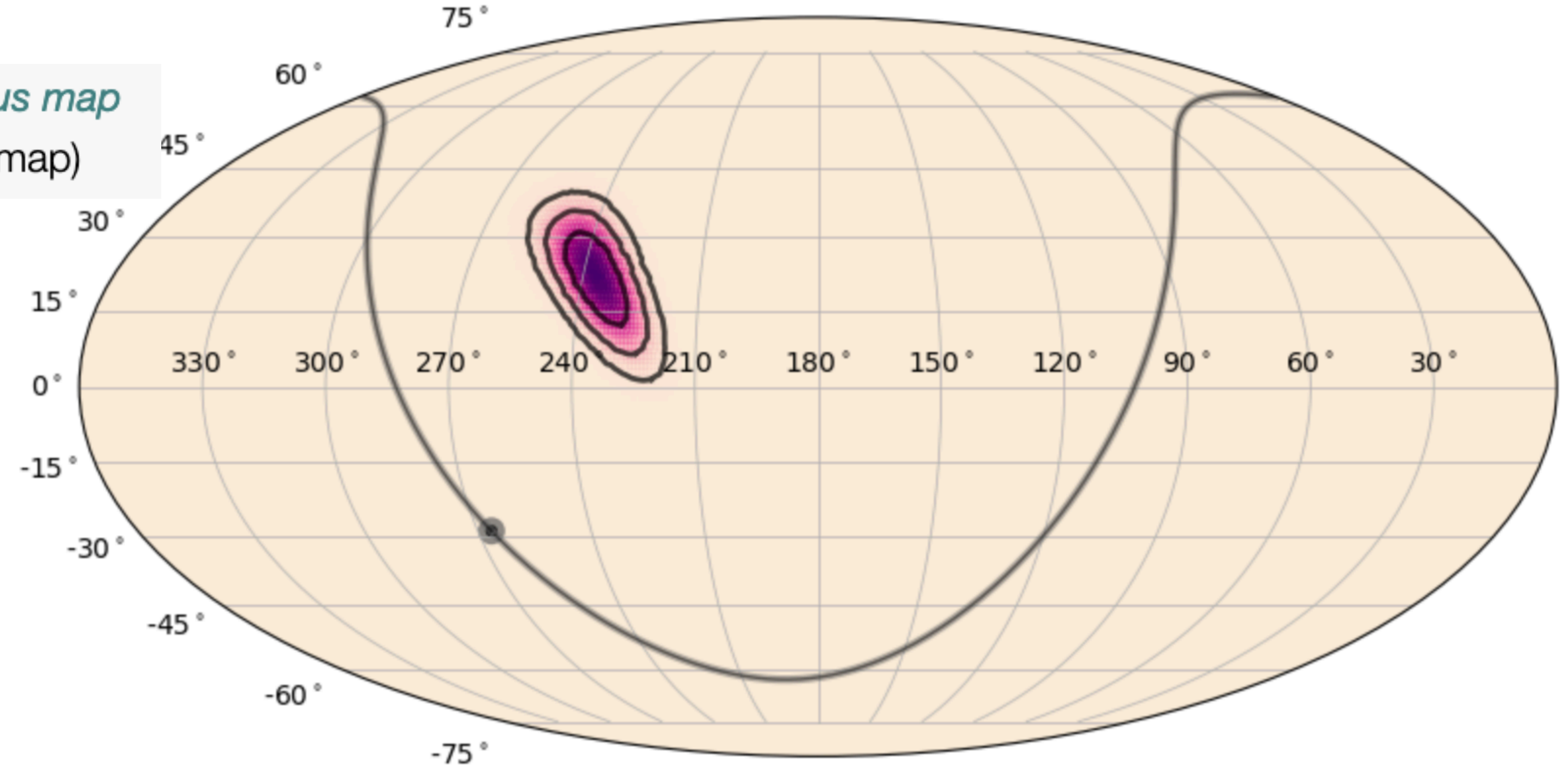
Map convolutions

```
# a single Gaussian of sigma_deg radius
def gauss_model(sigma_deg):
    sigma = deg2rad(sigma_deg)
    return ([sigma],[1.0])

# convolved with a 5-deg radius Gaussian
verts_convolved = verts_map.convolve(gauss_model, 5.0)
```

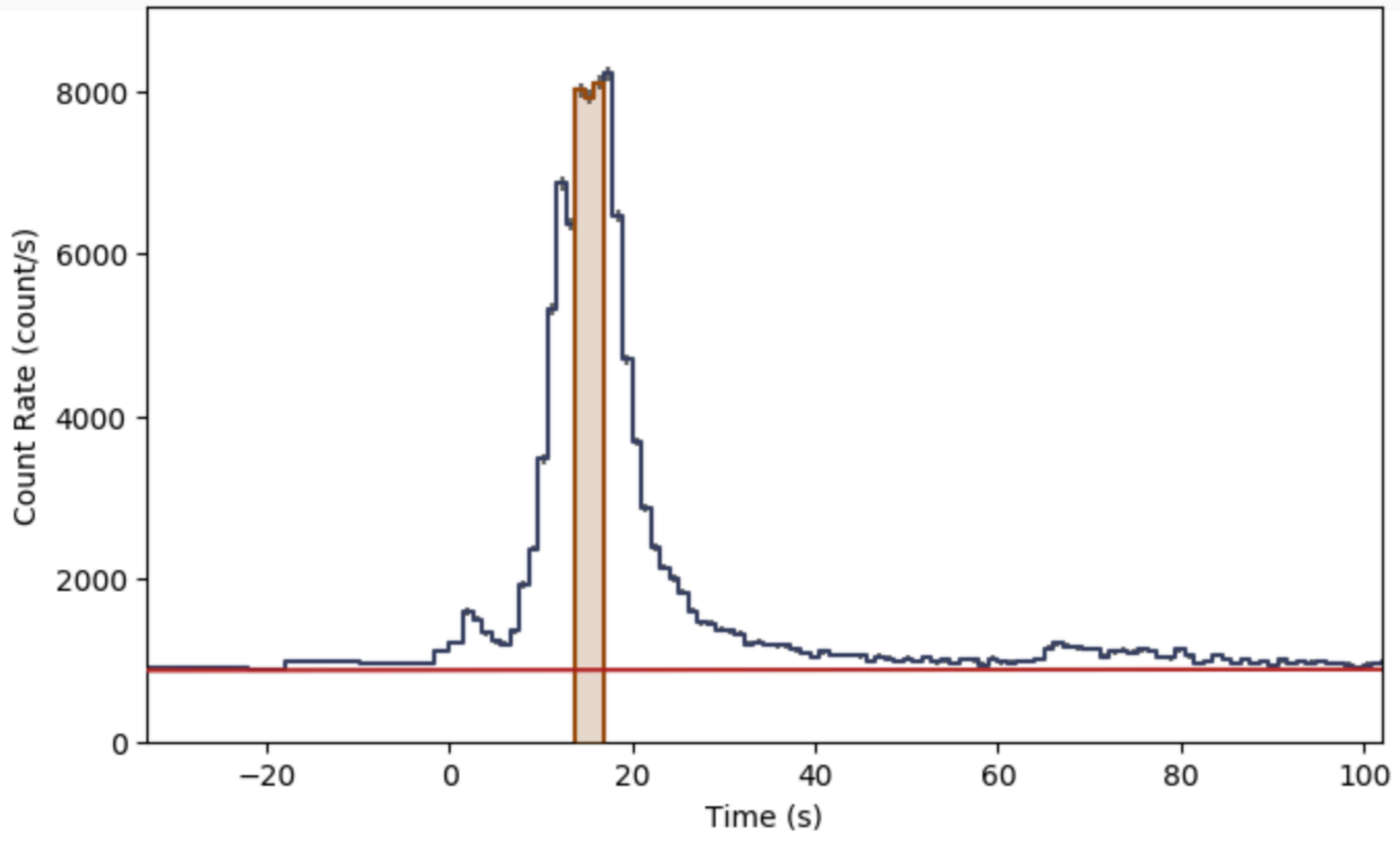


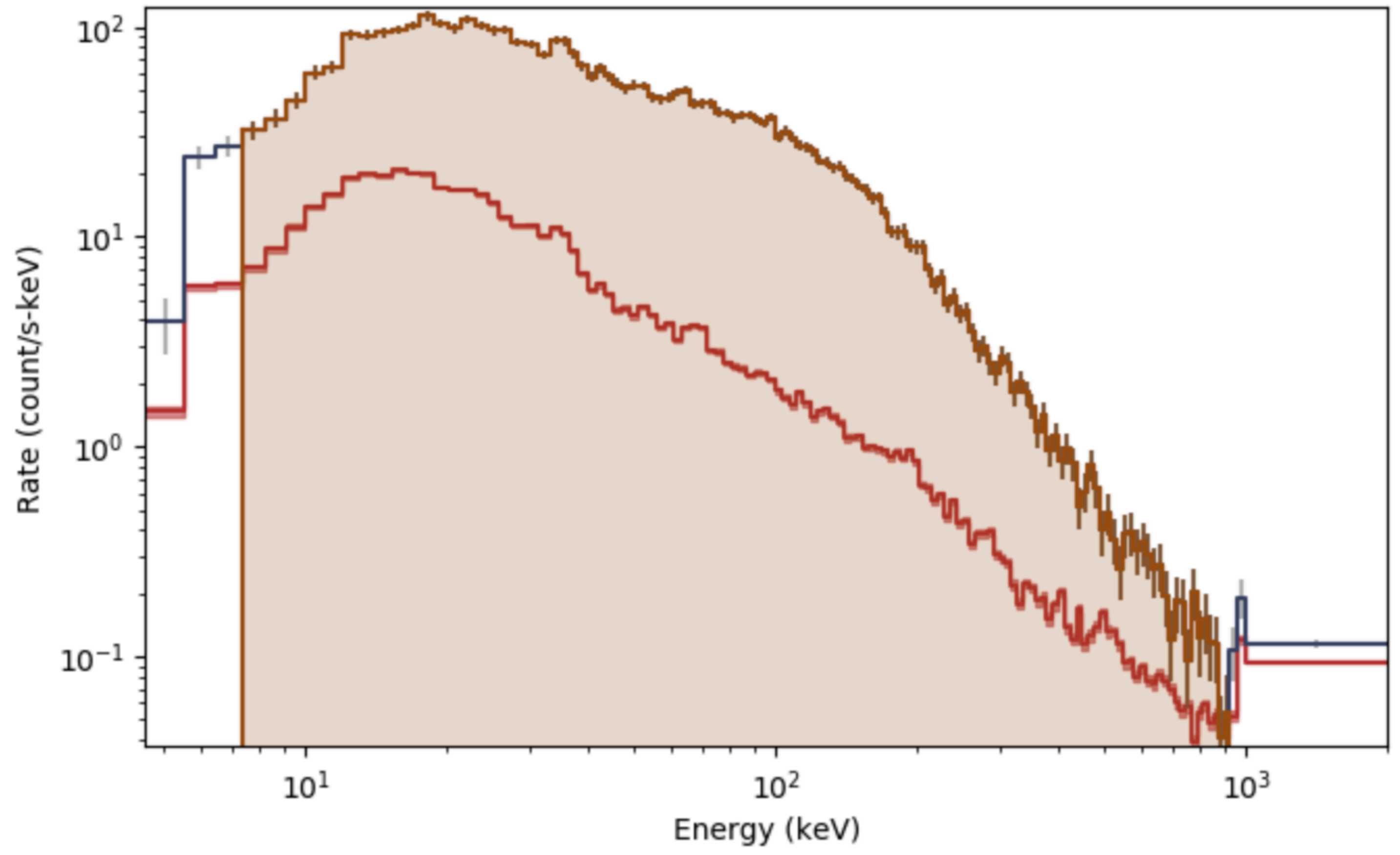
Map multiplication

```
# multiply the Vertices-convolved map with the annulus map
multiplied = HealPix.multiply(verts_convolved, annulus_map)
```

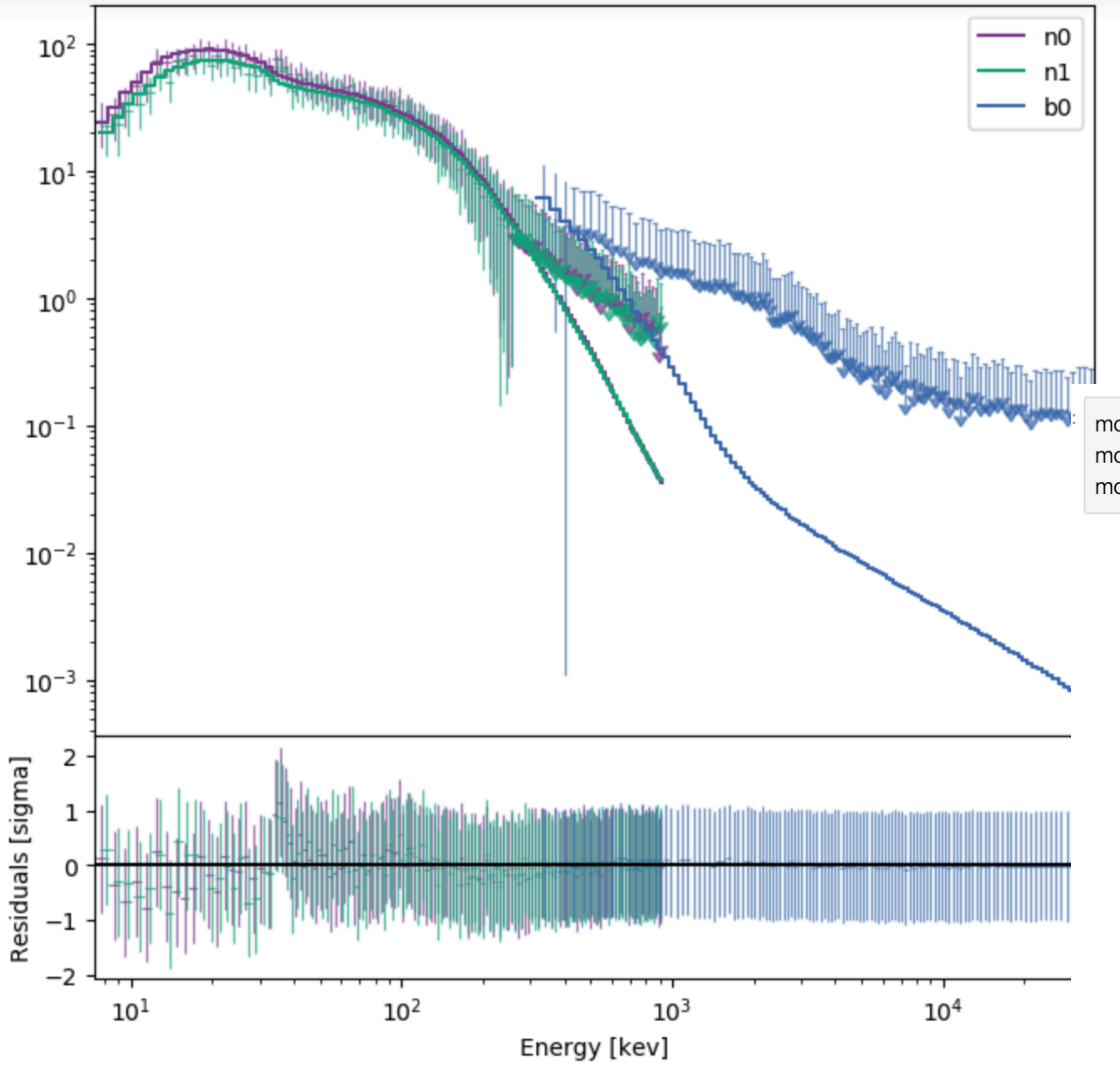


Spectral Fitting



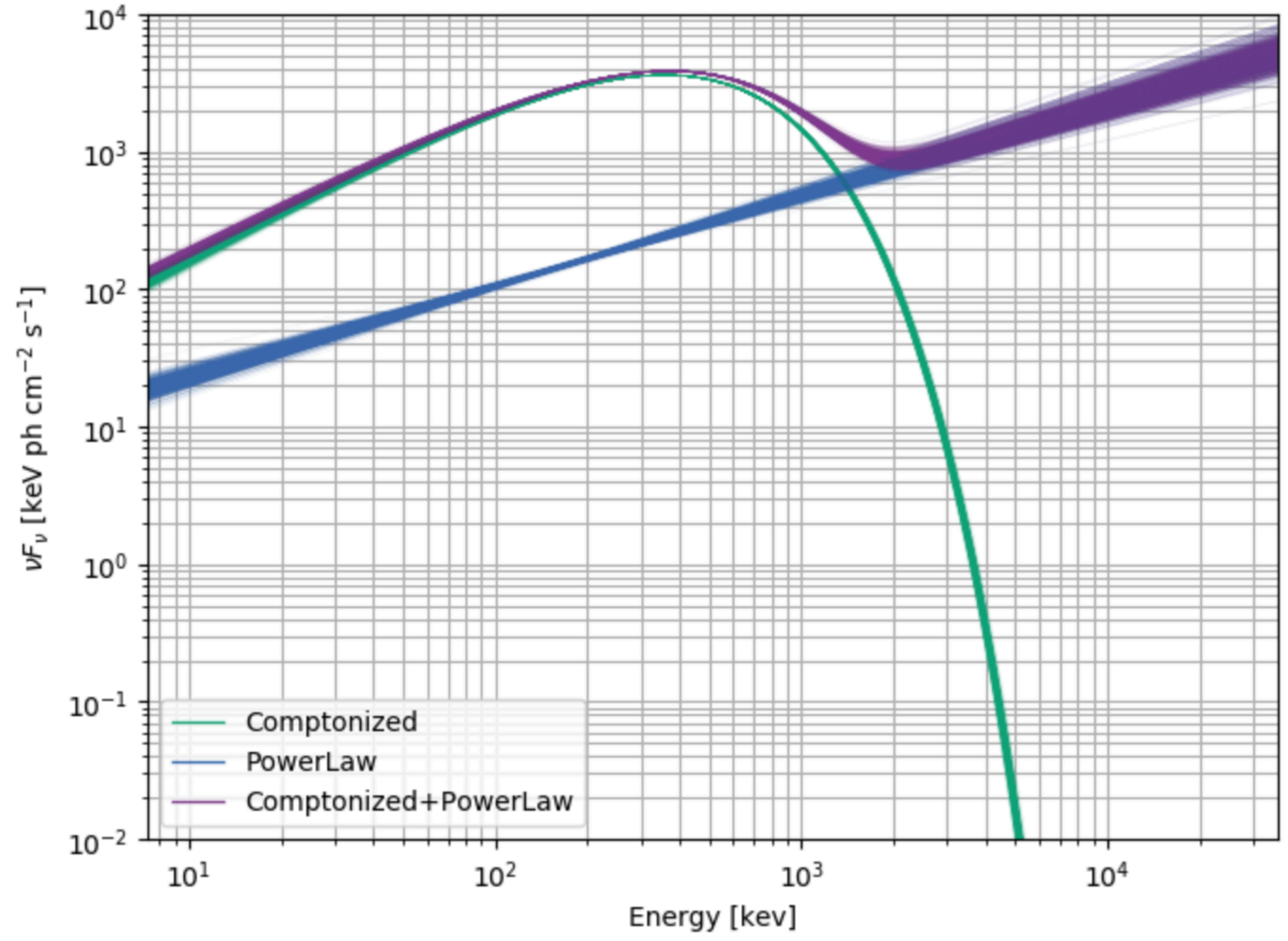



```python
modelplot = ModelFit(fitter=specfitter)
```



Can fit multiple components, plot the fit, and the spectrum for each component

```python
modelplot = ModelFit(fitter=specfitter, view='nufnu')
modelplot.ylim = (0.01, 10000.0)
modelplot.ax.grid(which='both')
```



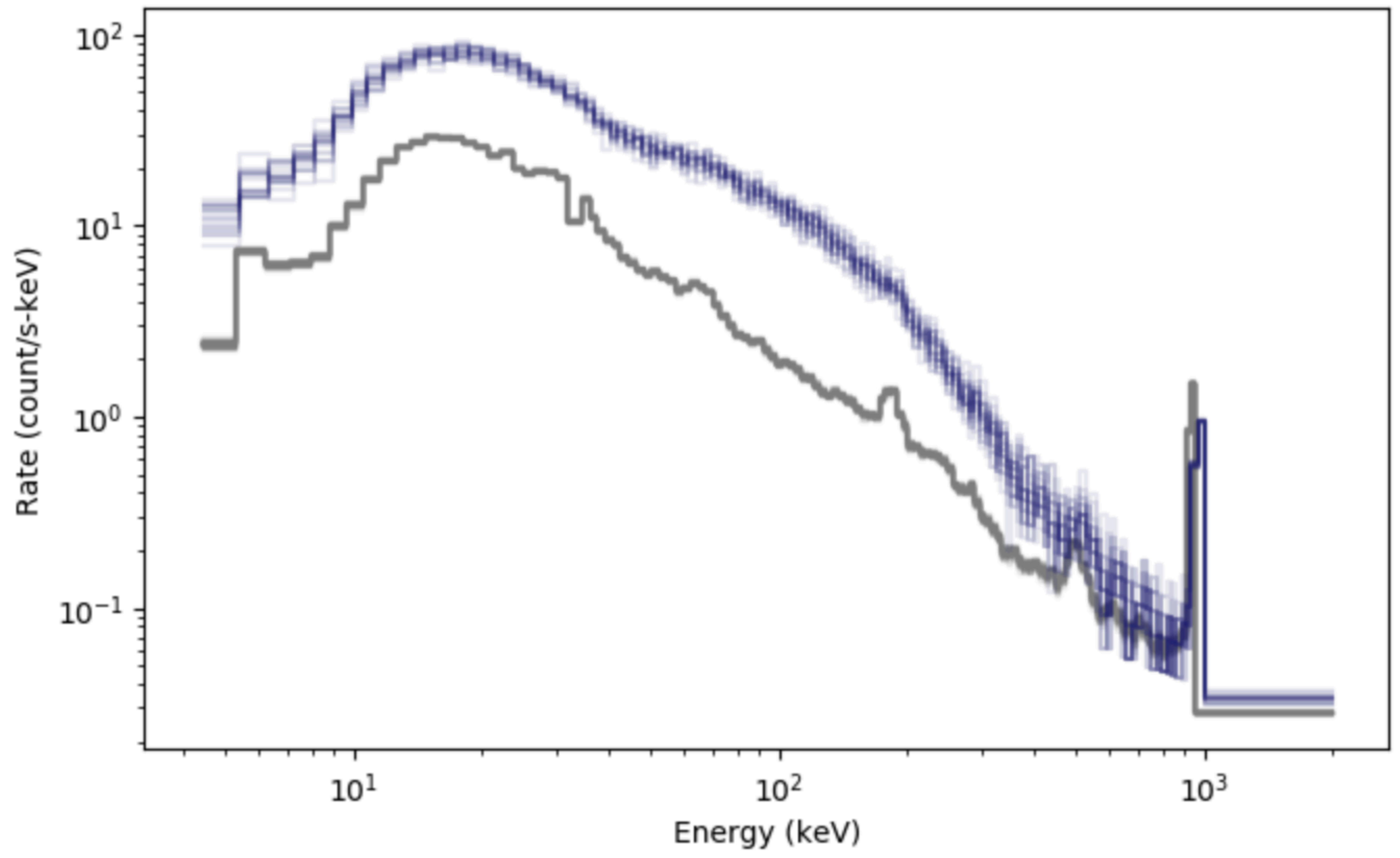
MLE with PG-Stat

```python
# we initialize with our PHAs, backgrounds, and responses:
specfitter = SpectralFitterPgstat(phas, bkgds.to_list(), rsps.to_list(), method='TNC')
# a power law, cut-off power law, and a Band function
from gbm.spectra.functions import PowerLaw, Comptonized, Band
# we've defined a new model that is the sum of a Comptonized function and a power law
comp_pl = Comptonized() + PowerLaw()
specfitter.fit(comp_pl, options={'maxiter': 1000, 'ftol': 1e-6})
```

Simulations

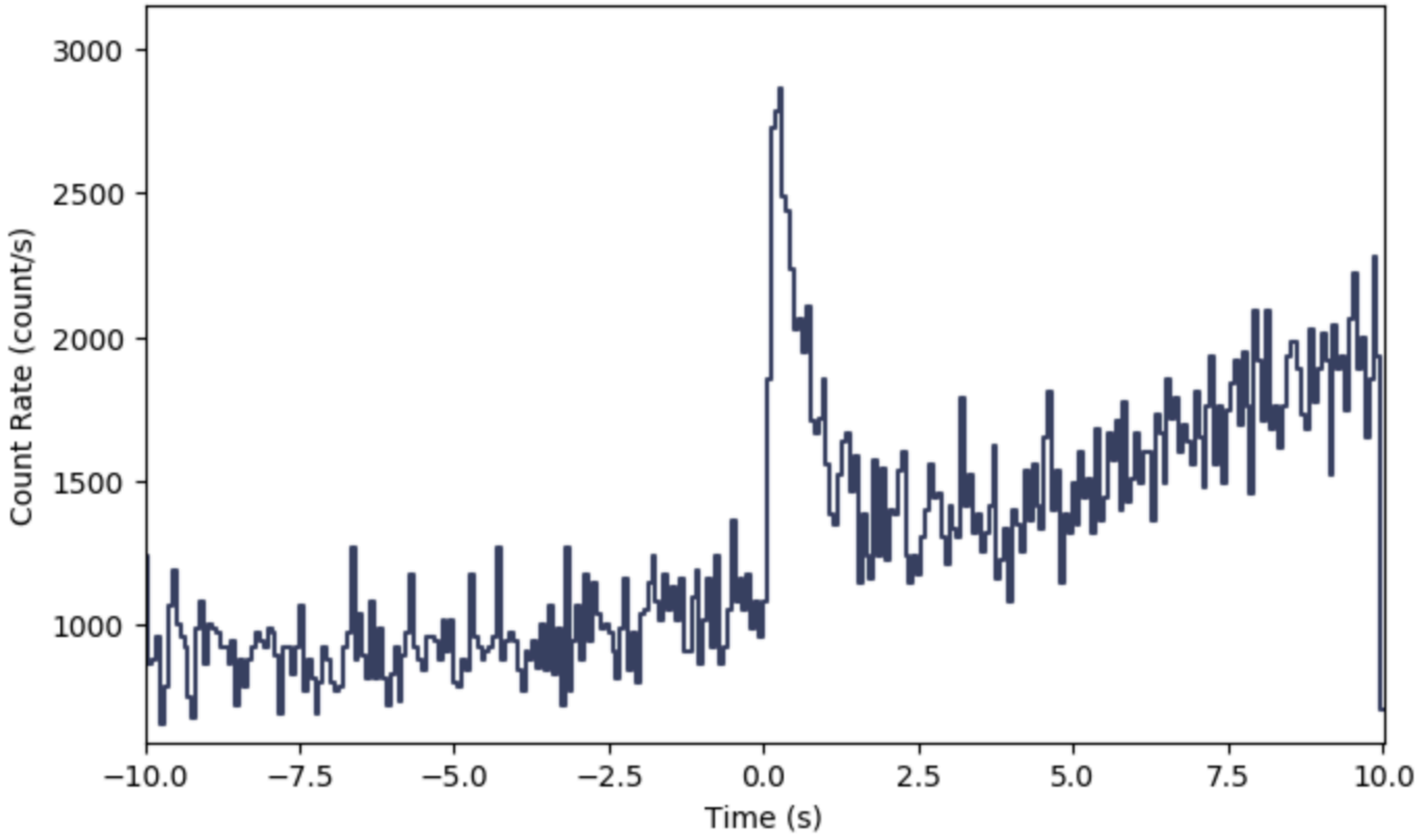
Simulate a spectrum (20 sims shown)

```
from gbm.simulate import PhaSimulator
pha_sims = PhaSimulator(rsp, Band(), band_params, exposure, spec_bkgd, 'Gaussian')
```



Simulate TTE/spectra

```
# a Norris pulse shape and a quadratic background
from gbm.simulate.profiles import norris, quadratic

norris_params = (0.05, 0.0, 0.1, 0.5)
quadratic_params = (1.0, 0.05, 0.003)

# source simulation
tte_sim = TteSourceSimulator(rsp, Band(), band_params, norris, norris_params)
tte_src = tte_sim.to_tte(-5.0, 10.0)

# background simulation
tte_sim = TteBackgroundSimulator(spec_bkgd, 'Gaussian', quadratic, quadratic_params)
tte_bkgd = tte_sim.to_tte(-10.0, 10.0)

# merge the background and source
tte_total = TTE.merge([tte_bkgd, tte_src])

# bin to 64 ms resolution so we can make a lightcurve plot
phaii = tte_total.to_phaii(bin_by_time, 0.064)
lcplot = Lightcurve(data=phaii.to_lightcurve(energy_range=(8.0, 900.0)))
```



Modules

gbm

- background
  background fitting, binned/unbinned algorithms
- binning
  binned/unbinned algorithms
- data
  PHAII, RSP, HEALPix, POSHIST, etc
- plot
  data plotting classes/library
- simulate
  simulate source and background spectra and event data
- spectra
  photon models/functions and spectral fitting classes

Utilities — coordinate/time conversion, data finder, detector/file definitions

Getting Started

Launching the Notebooks

If you have installed Jupyter as suggested above, you can run the notebooks provided with the Data Tools. After successful installation, the notebooks can be launched by calling:

```
$ gbm-demos
```

Welcome to the **GBM Data Tools**

Below is a list of tutorials that cover some of the most important aspects of the GBM Data Tools. Divided into two sections, the "Primary" tutorials cover high-level aspects and "Deeper Dives" explore the more detailed and fine-grained controls.

Primary Tutorials

GBM Science Data: Time History Spectra

TTE Data

Detector Response Files

Position History Data

Quicklook Trigger Data

GBM Localizations and Sky Maps

Finding GBM Data

Analysis Workflow: Reduction and Export

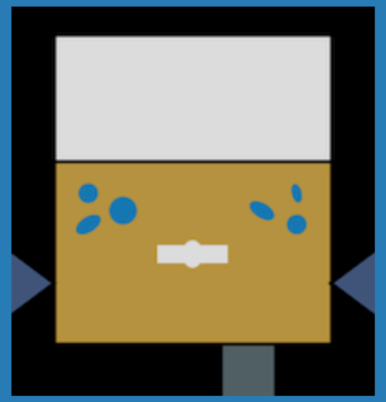
Analysis Workflow: Spectral Fitting

Documentation

On successful installation, you can launch the local HTML documentation by calling:

```
$ gbm-docs
```

GBM Data Tools



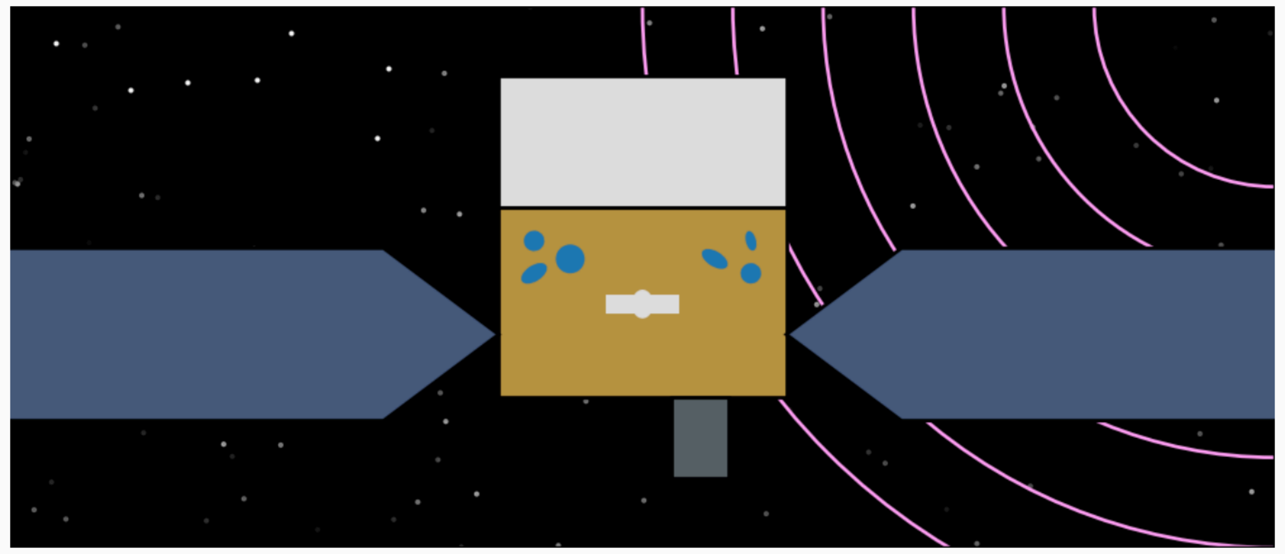
1.1.0

Search docs

Installation
Jupyter Notebooks
GBM Data Tools API
Change Logs

» Welcome to the Fermi GBM Data Tools documentation! View page source

Welcome to the Fermi GBM Data Tools documentation!



Hello, I'm Fermi. Pleased to meet you!

The Fermi GBM Data Tools is an Application Programming Interface (API) for GBM data. The fundamental purpose of the Data Tools is to allow general users to incorporate GBM analysis into their scripts and workflows without having to sweat very many details. To this end, the Data Tools have a fairly high-level API layer allowing a user to read, reduce, and visualize GBM data with only a few lines of code. For expert users, and users who want fine control over various aspects of their analysis, the Data Tools exposes a lower-level API layer, which can also be used to generalize the GBM Data Tools to data from other like instruments.

Architecture

Summary and other tidbits

- Toolkit to support a variety of analyses using GBM data
- Extensive API documentation, several notebook tutorials
- Coming attractions: Interface to an improved response generator and the GBM localization algorithm
- The tools are designed to be extended to other similar data BurstCube, and concept studies for LEAP and MoonBEAM
- NASA grant to expand to legacy missions such as BATSE, HETE-2, Suzaku, etc.
- Interested in feedback, bug reports, and suggestions on generalization