Applications of data analysis, modelling and image processing with open-source Python packages for Astrophysics, Imaging and Remote Sensing

MAX-PLANCK-INSTITUT FÜR KERNPHYSIK HEIDELBERG

Ellis R. Owen[1], Christoph Deil[2], Axel Donath[2]

[1]Department of Space & Climate Physics, University College London, Mullard Space Science Laboratory, Dorking, United Kingdom
[2]Max-Planck-Institut für Kernphysik, Saupfercheckweg 1, Heidelberg, Germany

e-mail: ellis.owen.12@ucl.ac.uk; christoph.deil@mpi-hd.mpg.de; axel.donath@mpi-hd.mpg.de

UCL

Data analysis in γ-ray astronomy has seen a shift in recent years towards collaborative open-source software packages used across many institutes. Over time, these have come to replace proprietary and non-open codes. This model to develop research codes is beginning to take root in other areas of science as well (particularly other areas of astronomy), and has much potential in the coming years. Examples in Python (which is popular, as it is great for prototyping as well as production codes), such as Astropy, Numpy, Scipy and scikit-image have broad applications across physics, astronomy and beyond (into imaging, surveying and remote sensing). As an example, we present a kernel-background / signal separation method implemented for gamma-ray astronomy, galactic plan survey test-statistic maps and galactic dynamical modelling simulations, all available in the open-source Astropy-affiliated Python package, Gammapy.

Open-Source in Astrophysics

1) Data Analysis with Open Source

Across many disciplines in science, recent years have seen a move towards **large collaborations** and experimental projects collecting **vast amounts of data**. This is particularly evident in astronomy where observational surveys have grown to involve scientists across many institutes.

A number of models for working with large collaborative data-intensive projects have been tried, from completely closed groups (e.g. Planck 2015) which publish only the end science products, to those which and publish data in large releases for the wider community to work with (e.g. Fermi).

- **Software may be developed 'in-house' by the collaboration**, and never released to the community, such that only the final and checked scientific results are published with the aim of ensuring a standard quality of science, and avoiding misunderstandings in analysis methods and results.
- **Raw data or minimally-processed data may be regularly released** to the wider community who choose to develop their own software which may then be published and shared as 'open source'.

In recent years, γ-ray astronomy has increasingly adopted the approach of collaborative open-source software codes and packages being shared across many institutes with **analysis pipelines** (a complete, start-to-finish code which processes raw or published data to calculate the desired scientific results) and **packages** (large collections of routines and functions grouped together as a re-usable and versatile toolbox) being provided to the community. This has offered number of advantages:

- Source code can be **tested and checked more rigorously** by more people
- Science results are **reproducible**
- **Transparency** in data analysis increases the community's **confidence** in the research
- **Avoids duplication** of work by different groups needing to do the same analysis
- Publicly-available open-source software packages can be used to **develop new tools and functionality** on a rapid, as-required basis without the need to re-implement existing analysis code to build on existing results from other research groups

Here, we introduce some useful packages increasingly being used in γ-ray astronomy, and show examples of science results achieved through the open-source software package, Gammapy, to demonstrate the scope of this approach in other areas of astrophysics, science and technology.

2) Python Packages

Python is a popular language among the open-source community. This is because it is straightforward to use and can be used at a range of levels, from a basic first-time user to a seasoned software developer. Python is great for prototyping as well as production codes, and one of its main strengths is the availability of existing, well-tested and established software packages offering open-source functionality from which new packages can be developed. Some useful examples for astrophysics, imaging and remote sensing are:

NumPy Base N-dimensional array package	SciPy library Fundamental library for scientific computing
Matplotlib Comprehensive 2D Plotting	IPython Enhanced Interactive Console
Sympy Symbolic mathematics	pandas Data structures & analysis

SciPy

*The **SciPy** stack is a collection of open source software for scientific computing including NumPy for numerical computation (largely implemented in C to take advantage of the speed of the compiled code with a more user-friendly Python wrapper) offering powerful linear algebra capabilities and N-dimensional array-based computation, the SciPy library for a range of numerical methods for signal processing, optimization and statistics, Matplotlib for 2D and 3D plotting, pandas for easy to use data structures and analysis, SymPy for algebra and symbolic mathematics, and IPython for interactive prototyping and scripting (SciPy 2016)*

scikit-image
image processing in python

***scikit-image** is a package useful particularly for image processing. This includes algorithms such as edge detection, feature detection, contour finding, de-noising, corner detection, various transformations, phase unwrapping, filters, cross-correlation, de-convolution, image comparison methods, template matching, adaptive thresholding, segmentation, and much more. The most intensive routines are implemented in Cython to improve performance. Methods such as edge and feature detection are particularly useful for source detection and removal methods in astronomy surveys when building source catalogues or estimating unresolved or background contributions to flux images, while these methods along with contour finding, de-noising and filters have many applications in imaging and more conventional non-scientific automated image processing (scikit-image 2016)*

astropy
A Community Python Library for Astronomy

***Astropy** is a community effort to develop a core package for astronomy in Python. It contains functionality intended to be useful for all astronomers and astrophysicists, including data manipulation with tables and arrays, automated units and quantities in calculations, spectral and spatial models and fitting methods, coordinate systems and conversions, analytic functions and input and output data writing routines. Additionally, a number of community driven specialist 'affiliated' packages exist (Astropy 2016, Robitaille et al. 2013).*

γπ A **Python** package for **gamma-ray** astronomy

*One such affiliated astropy packaged, **Gammapy**, is the extension of Astropy to the gamma-ray astronomy community. This includes numerous tools useful for gamma-ray astronomers and data analysis for systems such as Fermi-LAT, Imaging Atmospheric Cherenkov Telescopes such as HESS, Veritas and MAGIC and the upcoming CTA, and provides tools like power law spectral models, significance calculations, gamma-ray source population models and source/background separation algorithms for survey data. Some examples of the use of this package in gamma-ray astronomy are shown below (Gammapy 2016, Donath et al. 2015)*

Examples In Practice: γ-Ray Astronomy Imaging & Data Analysis

1) Source Detection

Standard methods to calculate significance maps and test statistics (TS) required in producing survey catalogues and images can be implemented in open-source packages. Gammapy, for example, includes code to calculate TS maps from counts (photon detection) maps, a background model and an exposure map (essentially measuring how much time each region of the map has been observed for). Typically, the test statistic used in the Cash statistic (see below) due to the Poisson nature of the noise expected in γ-ray datasets. Below is shown an example of a TS map computed using a Fermi-LAT gamma-ray dataset over the galactic plane (+/- 180 degrees in galactic longitude, +/- 4 degrees in galactic latitude).

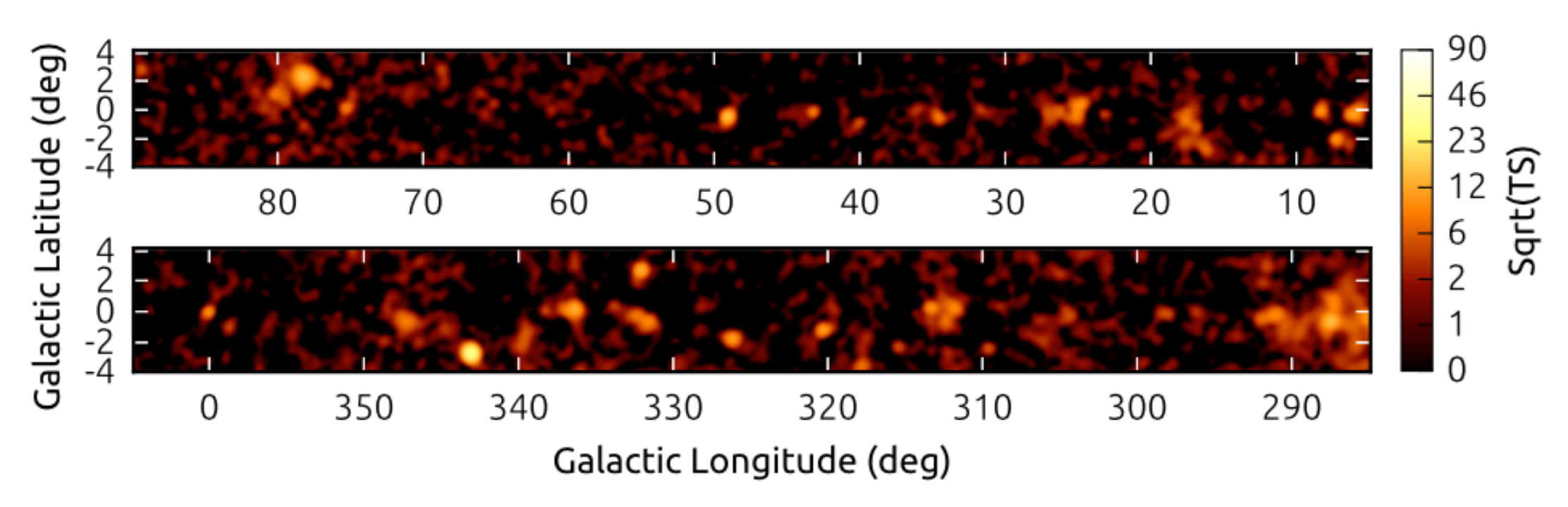


Figure 1: *Map of the Cash statistic calculated from Fermi-LAT data in a single spectral band. This demonstrates the use of test-statistic methods for source detection and catalog creation in high-energy gamma-ray astronomy amongst Poisson noise.*

Assuming a certain source morphology, which can be defined by Astropy 2D kernel models, the amplitude of the morphology model is fitted at every pixel of the input data using a Poisson maximum likelihood procedure. As input, a counts (data), background model and survey exposure map are provided. Based on the best fit flux amplitude, the change in test statistic (TS), compared to the null hypothesis (where amplitude is zero) is computed over N pixels as

$$U_{\mathrm{Cash}} = 2\sum_{i=1}^{N} c_i \ln\left(\frac{B_i + \hat{\alpha} S_i}{B_i}\right) - 2\hat{\alpha}\sum S_i$$

where U_{Cash} is the pixel Cash test statistic value, $\hat{\alpha}$ is the fitted amplitude of the morphology model at the pixel, B_i is the background and S_i is the point spread function (PSF). The resulting probability is of finding the fitted amplitude $\hat{\alpha}$ when there is only background signal.

To optimize the performance of the code, the fitting procedure is simplified by finding roots of the derivative of the fit statistics with respect to the flux amplitude (Stewert 2009). To further improve the performance, Python's multiprocessing facility is used (Donath et al. 2015).

2) Galaxy Modelling

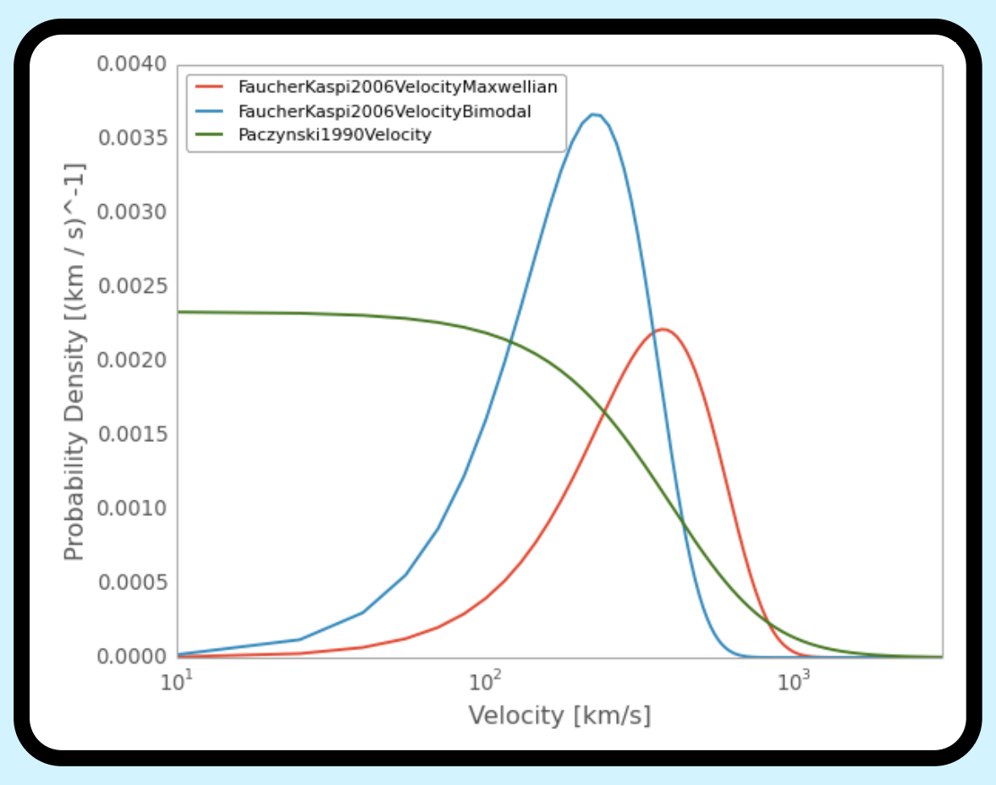
It can be useful in the context of surveys and population studies to simulate a population of sources according to observed or theoretical parameters. Gammapy offers the option to do this for γ-ray sources, using Monte Carlo methods without the need for the user to re-implement basic functionality. For example, a basic catalog of sources of various astrophysical types and their distributions can be generated including a galactic spiral arm model (Vallée, 2008).

The surface density of sources can be distributed according to a number of radial fitting functions, for which a given distribution is based on that for a given type of source

$$f(r) = A\begin{cases}\left(\frac{r}{r_\odot}\right)^{\alpha}\exp\left[-\beta\left(\frac{r-r_\odot}{r_\odot}\right)\right] & \textit{Surface density of supernova remnants in the galaxy (Case \& Battacharya, 1998)}\\ r_{\mathrm{exp}}^{-2}\exp\left(-\frac{r}{r_{\mathrm{exp}}}\right) & \textit{Birth surface density of neutron stars (Paczynski, 1990)}\\ \left(\frac{r}{r_\odot}\right)^{B}\exp\left[-C\left(\frac{r-r_\odot}{r_\odot}\right)\right] & \textit{Surface density of pulsars in the galaxy (Lorimer, 2006)}\\ \left(\frac{r+r_1}{r_\odot+r_1}\right)^{a}\exp\left[-b\left(\frac{r-r_\odot}{r_\odot+r_1}\right)\right] & \textit{Alternative surface density of pulsars in the galaxy (Yusifov \& Kucuk, 2004)}\\ \left(\frac{r}{r_\odot}\right)^{a}\exp\left[-b\left(\frac{r}{r_\odot}\right)\right] & \textit{Surface density of OB stars in the galaxy (Yusifov \& Kucuk, 2004)}\end{cases}$$

where A is a scaling parameter for the distributions, $r_\odot$ is the position of the Sun in the galaxy, and the other parameters allow for are fitting the distribution. Similarly, the velocity distribution of sources can be selected from a range according to the simulation requirements based on, for example, maxwellian or bimodal distributions. The results of these are plotted in the figures below.

Figure 2 (right): *Plot to show the different velocity distributions available in Gammapy galaxy simulations.*



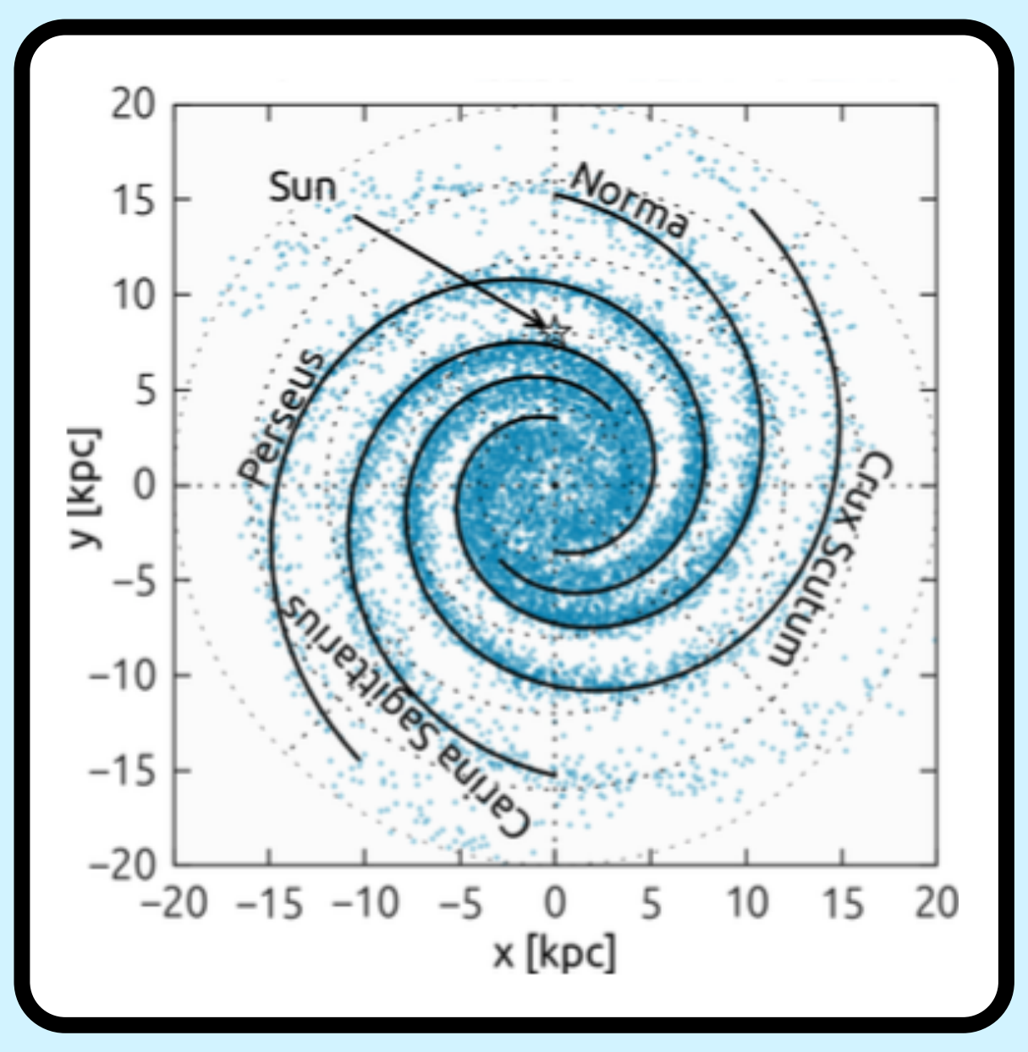


Figure 3 (left): *The spatial distribution of generated sources in a galaxy simulated using Gammapy.*

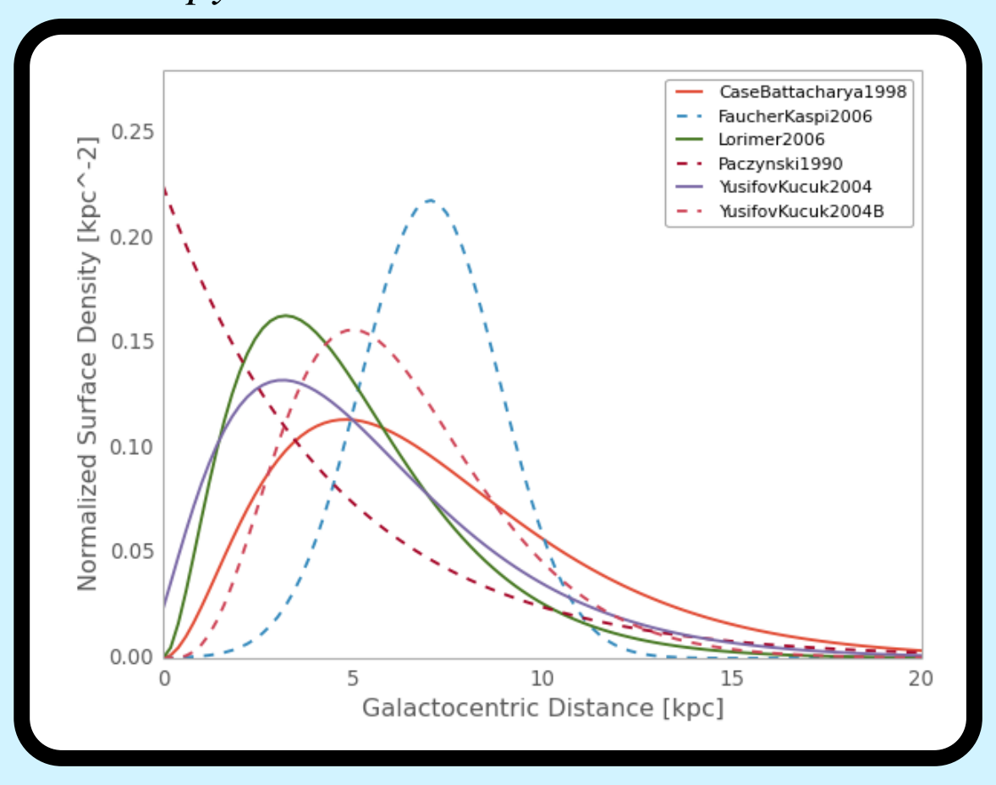


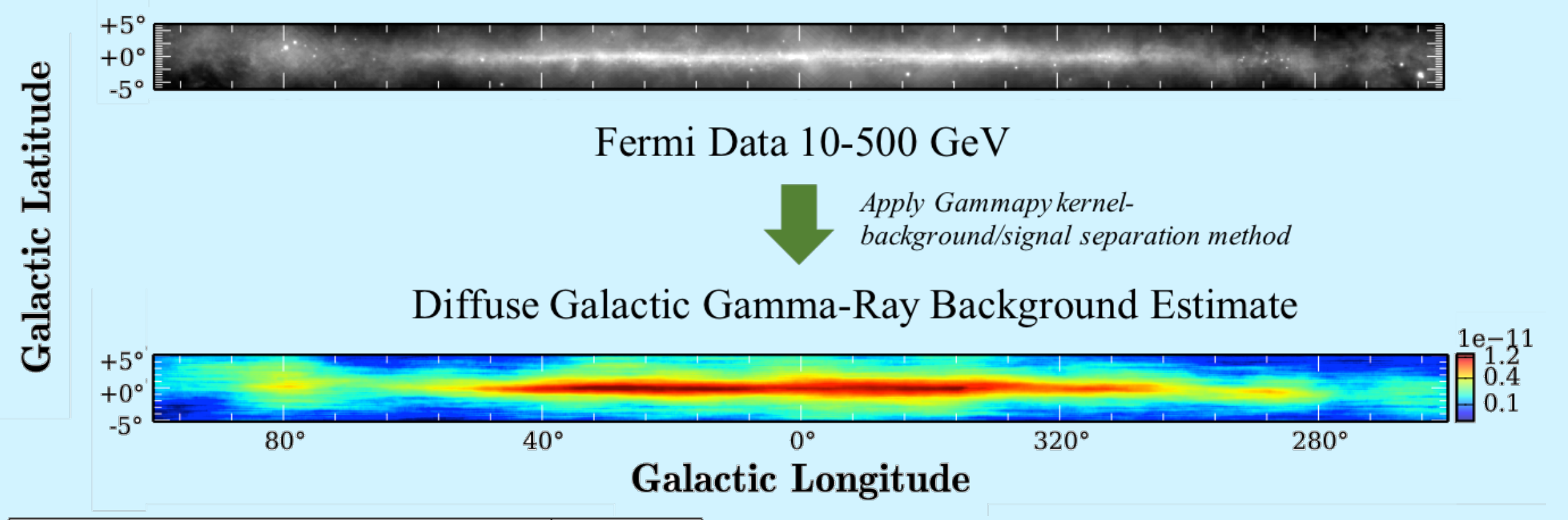
Figure 4 (right): *The various surface density radial distributions for simulated galaxies, according to the fitting functions described in the text above.*

3) Background Estimation/Signal Separation

An important task in many survey source catalogues is the estimation of the background signal to better allow sources to be identified. Estimating background images in high-energy gamma-ray observations can be particularly challenging due to the low number of photons, leading to Poisson noise. We show here an algorithm, included as open-source in Gammapy, appropriate for making background estimates in these cases, presented in Owen et al. (2015) as a generalization of the method described in Berge et al. (2008). The method takes a background kernel and a counts map (photon events observed) of the survey region. The counts map is correlated with an estimate of the point-spread function of the observing instrument, then the method then operates as follows:

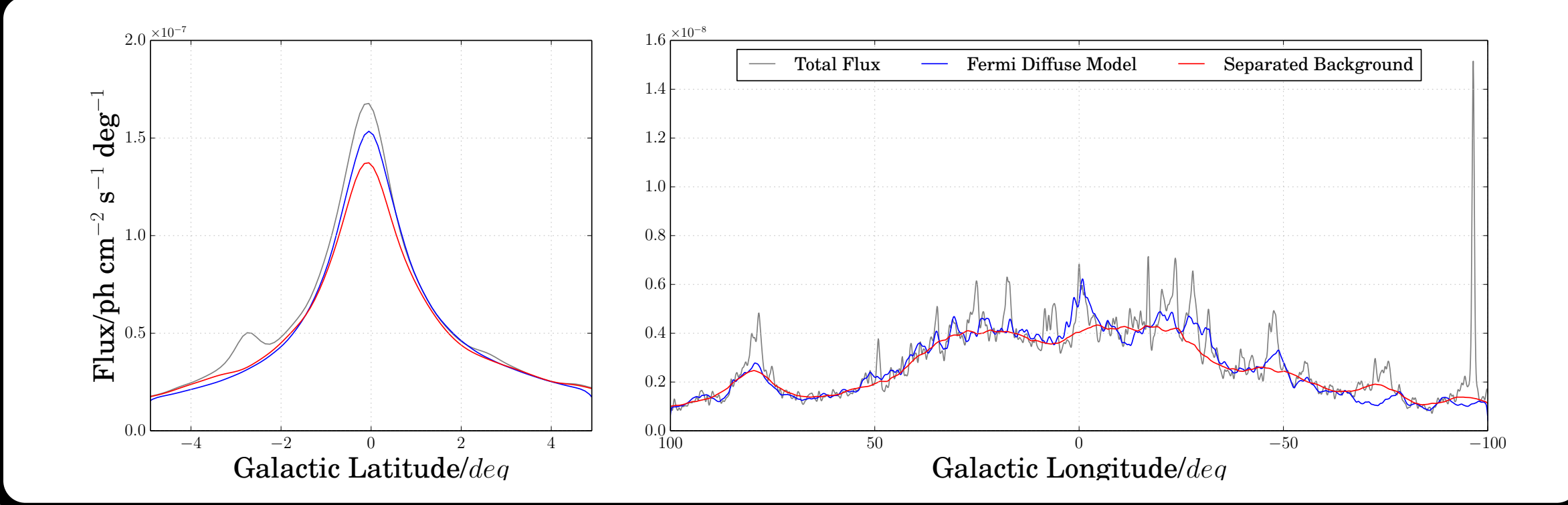
- The correlated counts map is convolved with the background kernel to give an initial background estimation
- A significance map is calculated from the counts map and initial background estimation, using the method described in Li & Ma (1983)
- Pixels of high significance above some threshold (in this case, 4 σ) are excluded from the initial background estimation and replaced with a value determined from the flux around the excluded source within a region the size of the background kernel. The result is a new background estimation
- A new significance map is created from the new background estimation

The last two steps then repeat iteratively with the mask around regions above the significance threshold dilated in each step until no change occurs in the background estimation.



Parameter	Value
Significance threshold	4 σ
Mask dilation radius	0.3°
Height of background kernel	0.5°
Width of background kernel	10°
Pixel size	0.1°

Figure 5 (above): *Fermi counts data (top), and the resulting diffuse galactic background estimate*
Table 1 (left): *Key parameters for the background estimation algorithm described above*
Figure 6 (below): *The resulting galactic latitude and longitude flux profile showing the background estimate (red line) in comparison to the total flux in this region, and the conventional background model (blue line)*



REFERENCES

*Astropy (2016): astropy.org – Berge et al. (2008): A&A 466(3):1219 – Case & Battacharya (1998): ApJ 504, 761 – Donath et al. (2015): PoS(ICRC2015)789 – Faucher & Kaspi (2006): ApJ 643, 332 – Fermi-LAT FSSC: fermi.gsfc.nasa.gov
Gammapy (2016): gammapy.readthedocs.org – Li & Ma (1983): ApJ 272, 317 – Lorimer (2006): MNRAS 372(2):777 – Owen et al. (2015): PoS(Scineghe2014)034 – Paczynski (1990): ApJ 348, 485 – Planck (2015): A&A 2015 Planck Results – Robitaille et al. (2013): AAP 558, A33
scikit-Image (2016): scikit-image.org – SciPy (2016): SciPy.org – Stewert (2009): AAP 495, 989-1003 – Vallée (2008): AJ 135, 1301-1310 – Yusifov & Kucuk 2004: A&A 422, 545*