PROCEEDINGS OF SCIENCE

Gammapy – A Python package for γ-ray astronomy

Axel Donath[*a], **Christoph Deil**[a], **Manuel Paz Arribas**[e], **Johannes King**[a], **Ellis Owen**[b], **Régis Terrier**[c], **Ignasi Reichardt**[c,d], **Jon Harris, Rolf Bühler**[f], **Stefan Klepser**[f]

[a]*MPIK, Heidelberg, Germany*
[b]*UCL-MSSL, Dorking, United Kingdom*
[c]*APC, University of Paris 7, France*
[d]*INFN, Padova, Italy*
[e]*Humboldt University, Berlin, Germany*
[f]*DESY, Zeuthen, Germany*
E-mail: Axel.Donath@mpi-hd.mpg.de, Christoph.Deil@mpi-hd.mpg.de

In the past decade imaging atmospheric Cherenkov telescope arrays such as H.E.S.S., MAGIC, VERITAS, as well as the *Fermi*-LAT space telescope have provided high quality images and spectra of the γ-ray universe. Currently the γ-ray community is preparing to build the next-generation Cherenkov Telecope Array (CTA), which will be operated as an open observatory.
Gammapy v0.3 (available at https://github.com/gammapy/gammapy under the open-source BSD license) is a new in-development Astropy affiliated package for high-level analysis and simulation of astronomical γ-ray data. It is built on the scientific Python stack (Numpy, Scipy, matplotlib and scikit-image) and makes use of other open-source astronomy packages such as Astropy, Sherpa and Naima to provide a flexible set of tools for γ-ray astronomers.
We present an overview of the *Gammapy* scope, development workflow, status, structure, features, application examples and goals. We would like *Gammapy* to become a community-developed project and a place of collaboration between scientists interested in γ-ray astronomy with Python. Contributions welcome!

The 34th International Cosmic Ray Conference,
30 July- 6 August, 2015
The Hague, The Netherlands

*Speaker.

 http://pos.sissa.it/

PoS(ICRC2015)789

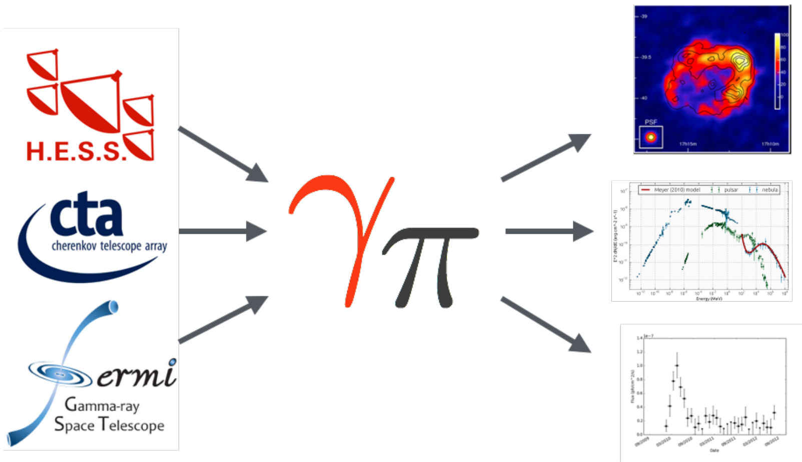

Figure 1: Gammapy is a Python package for high-level γ-ray data analysis. Using event lists, exposures and point spread functions as input you can use it to generate science results such as images, spectra, light curves or source catalogs. So far it has been used to simulate and analyse H.E.S.S., CTA and *Fermi*-LAT data, hopefully it will also be applied to e.g. VERITAS, MAGIC or HAWC data in the future.

1. Introduction

1.1 What is *Gammapy* ?

Gammapy is an open-source Python package for γ-ray astronomy. Originally *Gammapy* started as a place to share morphology fitting Python scripts for the work on the H.E.S.S. Galactic plane survey [1] two years ago. Since that time *Gammapy* has grown steadily: functionality as well as development infrastructure has been improved and it has been accepted as an in-development Astropy-affiliated package. Now, in this proceeding, we would like to introduce *Gammapy* to the community and present our vision of *Gammapy* as a future community-developed, general purpose analysis toolbox for γ-ray astronomers.

The general concept of *Gammapy* is illustrated in Figure 1. Based on pre-processed input data (e.g. event lists) provided by instruments such as H.E.S.S., *Fermi* or CTA, *Gammapy* offers the high-level analysis tools to generate science results such as images, spectra, light curves and source catalogs. By using common data structures and restriction to binned analysis techniques, all input data can be treated the same way, independent of the instrument. Research already making use of *Gammapy* is presented in [1, 2, 3].

1.2 How to get *Gammapy*

Recently *Gammapy* version 0.3 was released. It is available via the Python package index [1] or using package manager tools like pip and conda. *Gammapy* works on Linux and Mac (Windows most likely as well, but this has not been tested yet) and is compatible with Python 2.7 and Python

[1] https://pypi.python.org/pypi/gammapy/

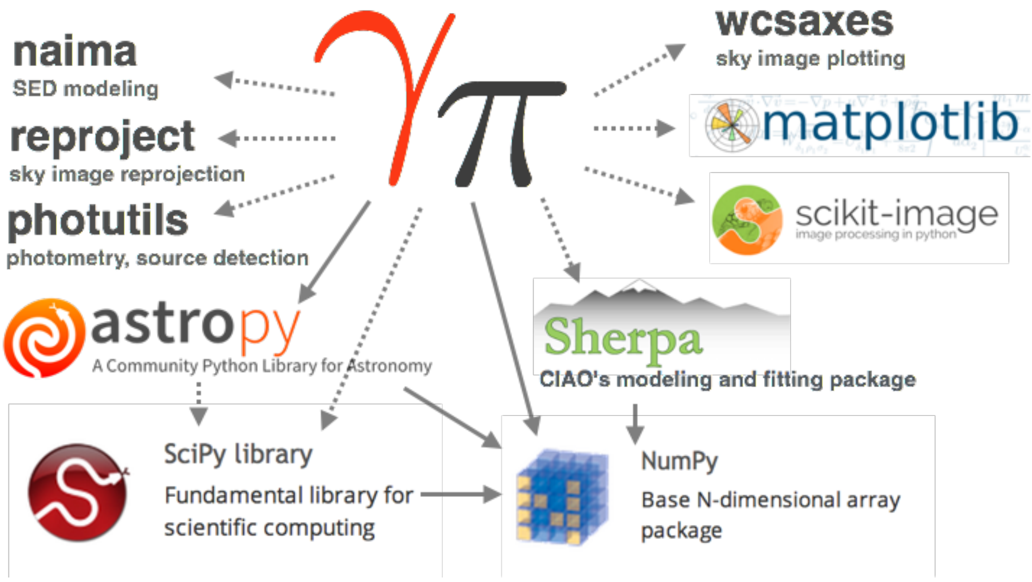

Figure 2: The Gammapy stack. Required dependencies Numpy and Astropy are illustrated with solid arrows, optional dependencies (the rest) with dashed arrows.

3.3 or later. Further details on requirements and installation are available online [2]. The latest documentation is available on *Read the Docs* [3], including tutorials, code and analysis examples. We have also set up a mailing list for user support and discussion [4].

2. The *Gammapy* stack

Gammapy is primarily built on the scientific Python stack. We employ Numpy and Astropy [4] as main dependencies and integrate other packages as optional dependencies where necessary. The current dependency structure is illustrated in Figure 2. Numpy provides the low level data structures and the framework for numerical/array computations. Astropy is used for higher level data structures like tables (Table) and n-dimensional data objects (NDData), for I/O, coordinates and WCS transformations, and handling of physical quantities with units. Scipy is used for advanced numerical and data processing algorithms. For specific image processing routines we additionally use scikit-image.

To allow morphology and spectral fitting of TeV sources with *Gammapy* we use the established X-ray modelling and fitting package Sherpa [5]. Sherpa allows interactive and scripted fitting of data sets with various spectral, light curve and morphology models taking instrument response functions into account. Additionally, it is possible to determine confidence levels on best-fit model parameters, compute likelihood profiles and goodness of fit measures. Sherpa recently also became an open-source project which makes it possible for users and external developers to contribute missing functionality or fix issues themselves in future.

[2] https://gammapy.readthedocs.org/en/latest/install.html
[3] https://gammapy.readthedocs.org/en/latest
[4] https://groups.google.com/forum/#!forum/gammapy

Modelling and fitting of non-thermal radiation processes to spectral energy distributions (SEDs) is provided with the Naima package. It uses Markov-Chain Monte Carlo emcee sampling to find best-fit model parameters of physical radiation models and thus determine the radiation mechanism that leads to the observed emission. The radiative models available in Naima can be called as source models in the *Gammapy* environment and be used, for example, to simulate spectra for source population studies.

Data visualizing and plotting of sky images is done with matplotlib and the astropy-affiliated package wcsaxes. Further astropy-affilated packages we allow as optional dependencies are photutils and reproject. Photutils is used for source detection and photometry and reproject for re-projection of sky images.

While this is a large number of dependencies, most of them are optional and only needed for small specific tasks so we do not see this as a strong disadvantage. Optional packages can be easily installed using package managing tools like *pip* or *conda*. By re-using other packages' functionality which would be complex to re-implement, new tools and techniques can be easily integrated into *Gammapy* to help obtain scientific results more quickly with a minimal coding effort.

3. Development workflow

Gammapy uses all the standard tools of modern community-driven open-source software development. The code repository is hosted on *GitHub* [5], which allows for convenient and public interaction of developers, contributors and users via the *GitHub* interface. This includes, for instance, *feature requests* for missing functionality, *issues* for bugs or questions and *pull requests* for code contributions. The code base is continuously built and tested in different virtual environments using the *continuous integration* service Travis CI. This helps to maintain high coding standards and compatibility with different versions of Numpy and Astropy.

The documentation of *Gammapy* is generated using Sphinx. An online version of the latest documentation is built automatically and is available on *Read the Docs* [6]. In addition to the main code repository we maintain gammapy-extra. This repository contains prepared catalog and map data and IPython notebooks with more extensive analysis examples and tutorials.

4. The *Gammapy* toolbox

4.1 Sub-packages

The *Gammapy* code base is structured into several sub-packages where each of the packages bundle corresponding functionality in a namespace. This follows the concept of other Python packages such as Astropy and Scipy. The following list gives a rough overview of the different sub-packages with a short description:

- `gammapy.astro` – Galactic population and emission models of TeV sources
- `gammapy.background` – Background estimation and modeling

[5] https://github.com/gammapy/gammapy
[6] https://gammapy.readthedocs.org/en/latest

- **gammapy.catalog** – γ-ray source catalog access and processing
- **gammapy.datasets** – Easy access to bundled and remote datasets
- **gammapy.detect** – Source detection tools and algorithms
- **gammapy.hspec** – Interface to spectral fitting with Sherpa
- **gammapy.image** – Image processing and analysis tools
- **gammapy.irf** – Instrument response function (IRF) access and handling
- **gammapy.morphology** – Morphology models and tools
- **gammapy.obs** – Observation handling
- **gammapy.spectrum** – Spectrum models and tools
- **gammapy.stats** – Statistical functions
- **gammapy.time** – Handling of time series and γ-ray lightcurves
- **gammapy.utils** – Utility functions and classes (in sub-modules)

There are some cases (e.g. **gammapy.spectrum.models** and several sub-modules of **gammapy.utils**) where the end-user functionality is exposed one level further down in the hierarchy. This is because putting everything into the top-level **gammapy.spectrum** or **gammapy.utils** namespace would lead to an unstructured collection of functions and classes. A large part of *Gammapy* 's functionality uses an object-oriented API. Functions are only used where it leads to an improved or more intuitve user interface.

4.2 Command-line tools

Gammapy includes various ready to use command-line tools which provide a familiar interface to data processing for many astronomers. A complete overview of all available tools can be found online [7]. When *Gammapy* is installed the user can simply type `gammapy-` on the command-line and use tab completion to see the list of tools, as all *Gammapy* tools start with the same `gammapy-` prefix. Specific information on single command-line tools and a description of available parameters is shown when calling the corresponding tool with the standard `-h` or `--help` option.

5. Usage examples

The *Gammapy* package should be considered as a toolbox out of which powerful analysis scripts can be composed easily by astronomers, even if they have very little programming experience. In the following section we present a selection of code examples demonstrating how to set up complex analysis steps with just a few lines of code.

5.1 Morphology fitting command-line tool

Gammapy includes a simple to use command-line script `gammapy-sherpa-like` to perform a Poisson maximum likelihood morphology fitting using FITS files as input. The input counts, exposure and background maps can be specified by the corresponding parameters `--counts`, `--exposure` and `--background`. The source model is defined in a JSON file and should be passed using the `--sources` option. The model parameters for a multi-Gaussian point spread function (PSF) can be specified in JSON format using the `--psf` option.

[7] https://gammapy.readthedocs.org/en/v0.3/scripts/index.html

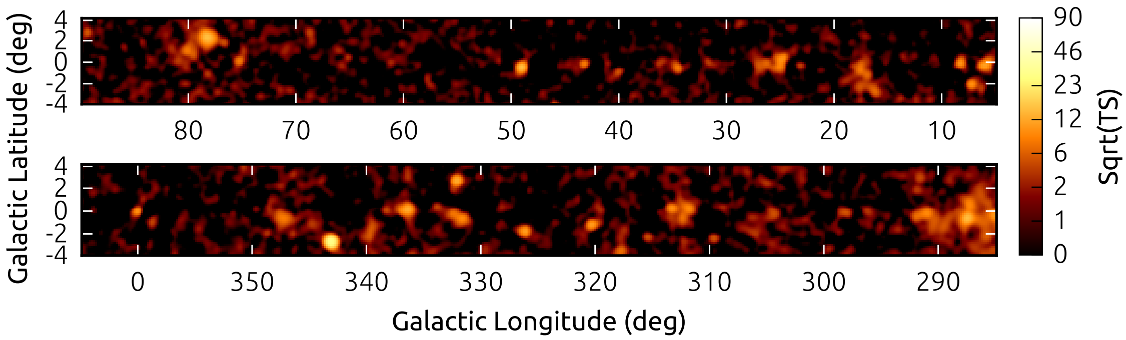

Figure 3: *Fermi* survey TS map.

```
$ gammapy-sherpa-like --counts counts.fits --exposure exposure.fits
 --background background.fits --psf psf.json --sources sources.json
  result.json
```

Fit results and additional information are stored in `result.json`.

5.2 Test statistics maps

The **gammapy.detect** module includes a high performance `compute_ts_map` function to compute test statistic (TS) maps for γ-ray survey data. The implementation is based on the method described in [6]. As input data, the user provides counts, background and exposure maps. The following code example demonstrates the computation of a TS map for prepared *Fermi* survey data, which is provided in gammapy-extra. The resulting TS map is shown in Figure 3.

```
from astropy.io import fits
from astropy.convolution import Gaussian2DKernel
from gammapy.detect import compute_ts_map
hdu_list = fits.open('all.fits.gz')
kernel = Gaussian2DKernel(2.5)
result = compute_ts_map(hdu_list['On'].data,
                        hdu_list['Background'].data,
                        hdu_list['ExpGammaMap'].data, kernel)
result.write('ts_map.fits')
```

5.3 Galactic population models

Gammapy also offers functionality for modeling and simulation. The **gammapy.astro** sub-package provides tools to simulate Galactic TeV source populations and source characteristics, which is useful in the context of surveys and population studies. Figure 4 illustrates a source population simulated with *Gammapy* and the different radial distribution models the user can choose from.

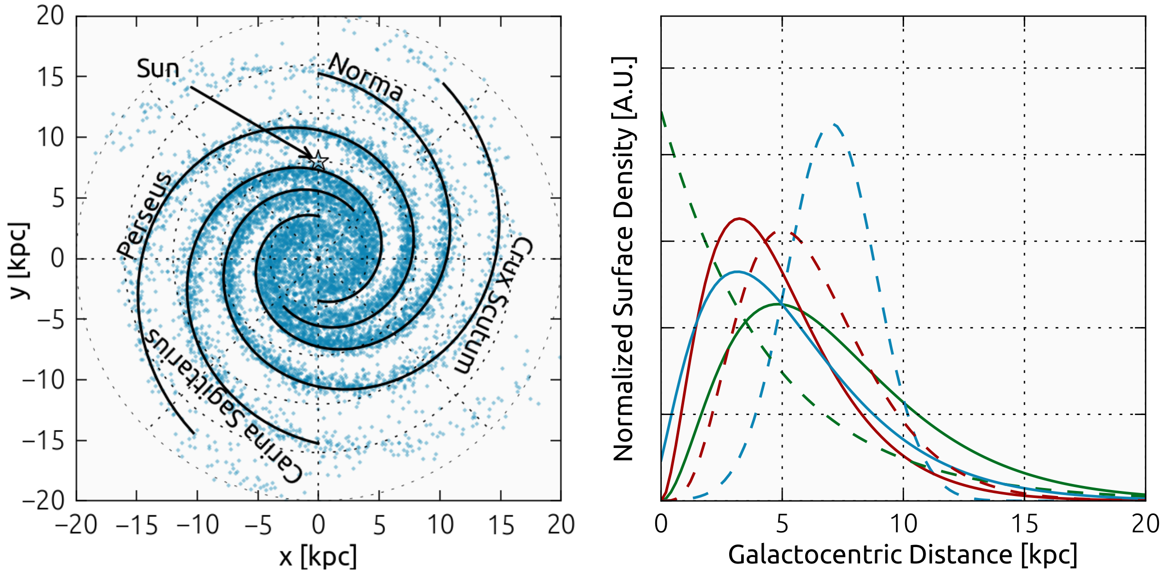

Figure 4: Galactic source population simulated using *Gammapy* , assuming a radial distribution of sources after [7] and the spiral-arm model of [8].

6. Planned functionality

Gammapy already includes key analysis features like morphology and spectrum fitting but both are currently limited to either 2D image-based data or 1D spectral data. As a major next step we plan to support joint likelihood fitting of datasets, as illustrated in Figure 5. Events are binned into longitude, latitude and energy cubes and fitted simultaniously with spectral and spatial models taking energy-dependent background, exposure and point spread function (PSF) into account.

Support for un-binned analyses is not planned.

7. Summary

Gammapy 0.3 is still alpha quality software. It is a package where standard γ-ray analyses are available on the one hand while, on the other hand, integration and prototyping of new methods is easily possible. So far it only contains limited functionality but the setup of documentation, testing and deployment is already very advanced. It's scope will continously grow and we hope that many users and developers show interest in open and reproducible γ-ray astronomy with Python. As long-term goal we would like *Gammapy* to turn into a fully community-developed package. So all contributions to *Gammapy* are welcome! If you don't know how to turn your scripts into production-quality, reusable code, please get in touch with us (e.g. using the mailing list) and we will help you get there!

For further information on how *Gammapy* and other tools are being used for H.E.S.S. data analysis, we encourage you to look at [9].

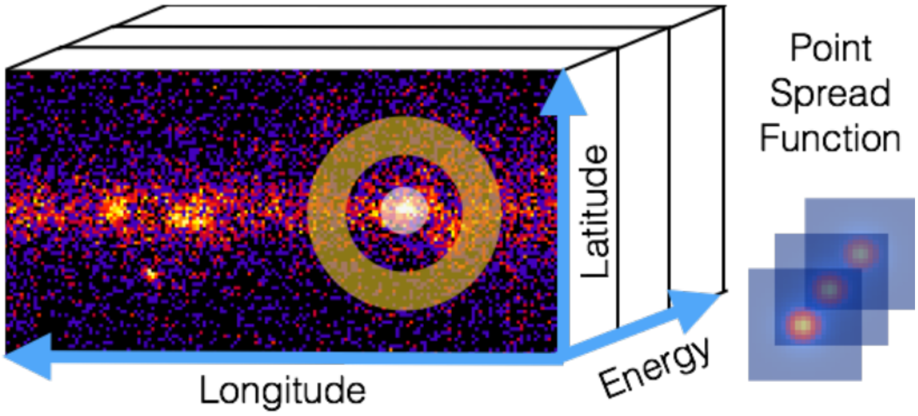

Figure 5: Gammapy data model illustration. Binned analysis of lon-lat-energy cube data is supported via joint likelihood analysis of one image per energy bin. On-off-region based spectral analysis is supported as well.

References

[1] C. Deil et al. for the H.E.S.S. collaboration, *The H.E.S.S. Galactic plane survey*, in *these proceedings*, 2015.

[2] E. Owen, C. Deil, A. Donath, and R. Terrier, *The gamma-ray Milky Way above 10 GeV: Distinguishing Sources from Diffuse Emission*, *PoS(SciNeGHE2014)* **34** (June, 2015) [arXiv:1506.0231].

[3] G. Pühlhofer et al., for the H.E.S.S. collaboration, *Search for new supernova remnant shells in the Galactic plane with H.E.S.S.*, in *these proceedings*, 2015.

[4] Astropy Collaboration, T. P. Robitaille, E. J. Tollerud, and P. Greenfield et al., *Astropy: A community Python package for astronomy*, *AAP* **558** (Oct., 2013) A33.

[5] B. Refsdal et al., *Sherpa: 1d/2d modeling and fitting in python*, in *Proceedings of the 8th Python in Science Conference*, (Pasadena, CA USA), pp. 51 – 57, 2009.

[6] I. M. Stewart, *Maximum-likelihood detection of sources among Poissonian noise*, *AAP* **495** (Mar., 2009) 989–1003, [arXiv:0901.3276].

[7] I. Yusifov and I. Küçük, *Revisiting the radial distribution of pulsars in the Galaxy*, *AAP* **422** (Aug., 2004) 545–553, [astro-ph/0405559].

[8] J. P. Vallée, *New Velocimetry and Revised Cartography of the Spiral Arms in the Milky Way - A Consistent Symbiosis*, *aj* **135** (Apr., 2008) 1301–1310.

[9] C. Deil et al. for the H.E.S.S. collaboration, *H.E.S.S. data analysis with open source science tools*, in *these proceedings*, 2015.