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, London, England*
[c]*APC, Paris, 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 us with detailed images and spectra of the γ-ray universe for the first time. 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 (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 current *Gammapy* features and example analyses on real as well as simulated γ-ray datasets. 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.

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike Licence. http://pos.sissa.it/

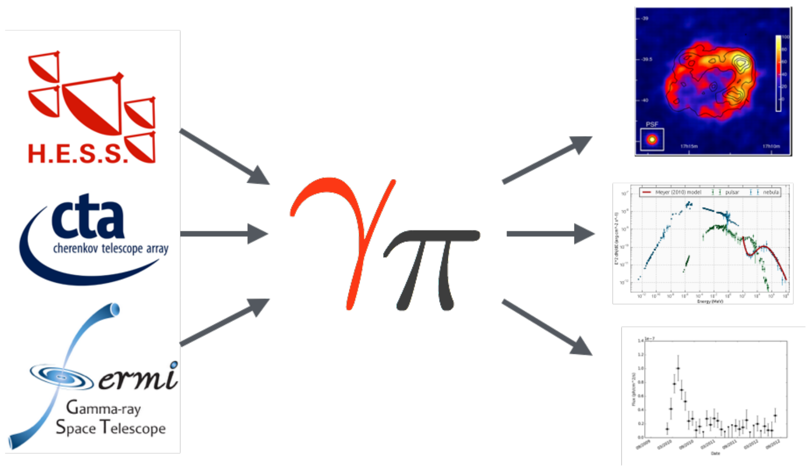

Figure 1: Gammapy is a Python package for high-level γ-ray data analysis. Using event lists, exposures and point spread functions as input you can use it to generate science results such as e.g. images, spectra, light curves or source catalogs. So far it has been used to simulate and analyse H.E.S.S.,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

1. What is *Gammapy* ?

Gammapy is an open-source Python package for γ-ray astronomy. It was started by Christoph Deil and Axel Donath in 2013 as a place to share morphology fitting Python scripts that were developed for the work on the H.E.S.S. Galactic plane survey. Recently *Gammapy* version 0.3 was released, which is available via the Python package index https://pypi.python.org/pypi/gammapy. *Gammapy* holds the status of an in-development Astropy affiliated package. Research making use of *Gammapy* is presented in [1], [2] and [3].

The general idea behind *Gammapy* is illustrated in fig. 1. *Gammapy* is a instrument-independent high-level analysis toolbox to generate science results such as images, spectra, light curves and source catalogs from general input data.

2. The *Gammapy* stack

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

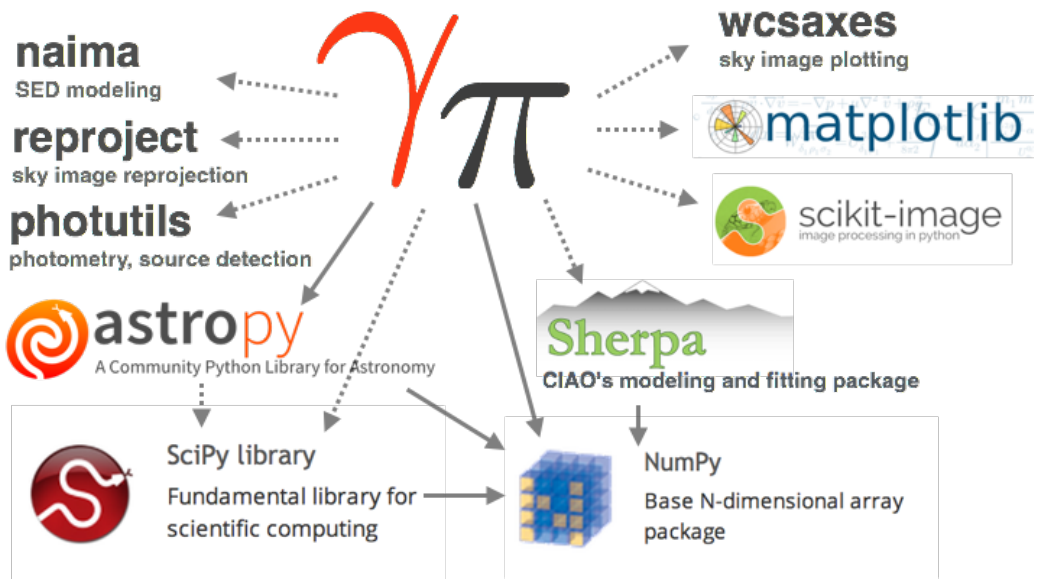
naima
SED modeling
reproject
sky image reprojection
photutils
photometry, source detection
astropy
A Community Python Library for Astronomy
wcsaxes
sky image plotting
matplotlib
scikit-image
image processing in python
Sherpa
CIAO's modeling and fitting package
SciPy library
Fundamental library for scientific computing
NumPy
Base N-dimensional array package

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

To allow morphology and spectral fitting of TeV sources with *Gammapy* we use the x-ray modelling and fitting package Sherpa [5]. Sherpa recently also became an open source project. Modelling of non-thermal radiation processes and SEDs is provided with the Naima package.

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, reproject for re-projection of sky images.

One may argue that this is a large number of dependencies, but as most of them are optional and only needed for small specific tasks, we don't see any major disadvantages of our approach. Optional packages can be easily installed using package managing tools like *pip* or *conda*. By reusing other packages, useful and partly complex functionality can be easily integrated in *Gammapy* and help to quickly obtain scientific results with minimal coding effort.

3. Development workflow and user support

Gammapy uses all the standard tools of modern community-driven open-source software development. The code repository is hosted on *GitHub* https://github.com/gammapy/gammapy, which allows a convenient and public interaction of developers, contributors and users via the *GitHub* interface. This includes e.g. so called *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 https://gammapy.readthedocs.org/en/latest, including tutorials, code and analysis examples. 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. We also set up a mailing list for user support and discussion: https://groups.google.com/forum/#!forum/gammapy

4. The *Gammapy* toolbox

4.1 Sub-packages

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

- `gammapy.astro` – Galactic population and emission models of TeV sources
- `gammapy.background` – Background estimation and modeling
- `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` – Statistics 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 in one level further down in the hierarchy, 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, but functions are also used, where it makes sense and leads to a better user interface.

4.2 Command line tools

Gammapy includes various ready to use command line tools, that provide a familiar interface to data processing for many astronomers. A complete overview of all available tools can be found on https://gammapy.readthedocs.org/en/v0.3/scripts/index.html. When *Gammapy* is installed one can just type `gammapy-` in 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 the description of available parameters is shown when calling the corresponding tool with the standard `-h` or `--help` option.

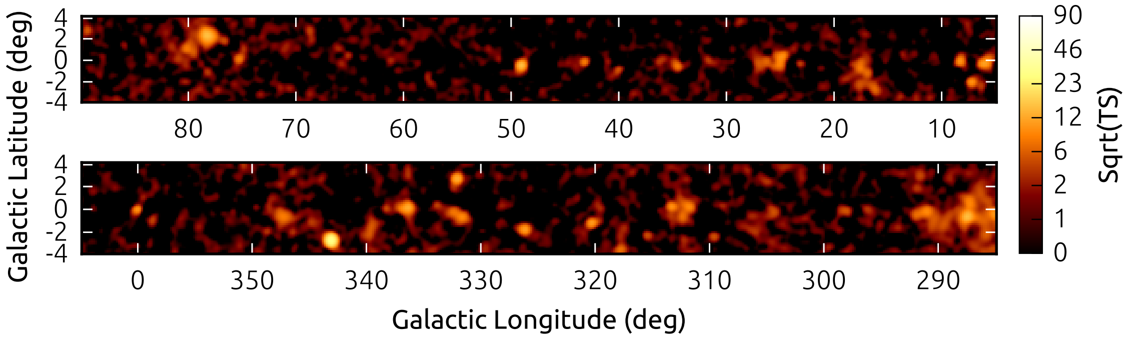

Figure 3: Fermi survey TS map.

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 even 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:

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

The fit results and additional info 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 statistics (TS) maps for γ-ray survey data. The implementation is based on the method described in [6]. As input data a counts, background and exposure map have to be provided. The following code example demonstrates the computation of a TS map for prepared Fermi survey data, which is provided in gammapy-extra:

```python
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)
```

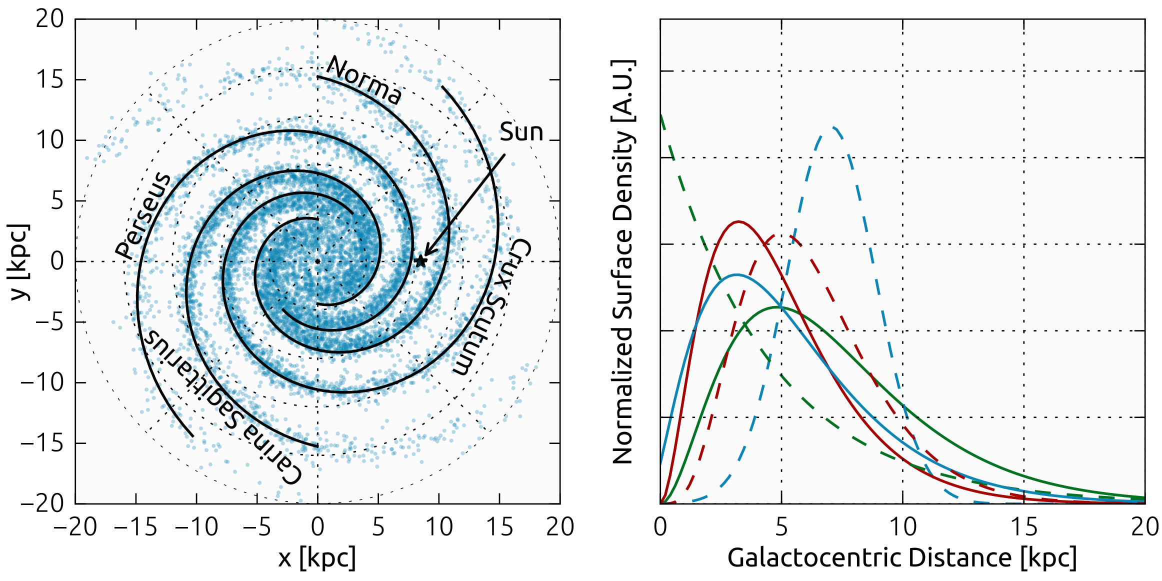

Figure 4: Galactic source population simulated using *Gammapy* , assuming a radial distribution of sources after [7] and the spiral-arm model of [8].

```
result = compute_ts_map(hdu_list['On'].data,
                        hdu_list['Background'].data,
                        hdu_list['ExpGammaMap'].data, kernel)
```

5.3 Galactic population models

The `gammapy.astro` sub-package provides tools to simulate Galactic TeV source populations. This is mainly useful in the context of surveys and population studies. The user can choose from different radial and velocity distributions (illustrated in figure 4) and also include a spiral-arm model. The following example illustrates how to simulate a catalog of Galactic TeV sources:

```
import astropy.units as u
from gammapy.astro.population import make_base_catalog_galactic

max_age = 1E6 * u.yr
SN_rate = 3. / (100. * u.yr)
n_sources = max_age * SN_rate
table = make_base_catalog_galactic(n_sources=n_sources,
                                   rad_dis='L06',
                                   vel_dis='F06B',
                                   max_age=max_age,
                                   spiralarms=True)
```

The total number of sources is determined assuming a maximum age and a supernova rate. The table returned is an instance of `astropy.table.Table` which can be used for further processing. The example population with spiral-arms is shown in figure 4.

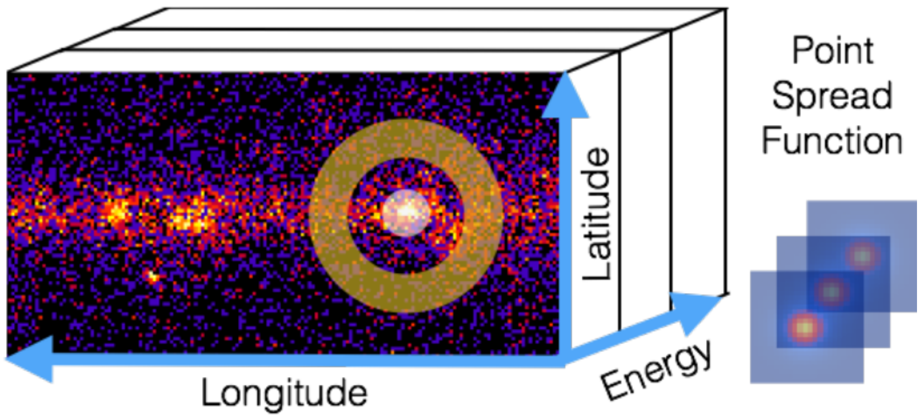

Figure 5: Gammapy data model illustration. Binned analysis of lon-lat-energy cube data is supported via joint likelihood analysis of one image per energy bin. On-off-region based spectral analysis is supported as well.

6. Summary and future plans

Gammapy 0.3 is still alpha quality software. So far it only contains limited functionality, but the setup of documentation, testing and deployment is already very advanced. Key analysis features like morphology and spectral fitting are available, but currently limited to 2D i.e. image based data. As a major next step we plan to support joint likelihood fitting of datasets, as illustrated in fig. 5. Events are binned into longitude, latitude and energy cubes (or other z-axes such as event class) and compared with spectral and morphological models, taking background, exposure and point spread function (PSF) into account. Support for un-binned analyses is not planned. A 1.0 release is planned for late fall this year.

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]. The progress on CTA data management, data formats and software is documented in various other contributions at ICRC 2015 ([10, 11]).

As long-term goal we would like *Gammapy* to grow into a community developed package, where standard γ-ray analyses are available on the one hand and integration and prototyping of new methods is easily possible on the other hand. Your contributions are welcome! If you don't know how to turn your scripts into production-quality, reusable code, get in touch with us and we'll help you get there.

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*, *ArXiv e-prints* (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.

[10] J. Knödlseder al. for the CTA consortium, *Observer Access to the Cherenkov Telescope Array*, in *these proceedings*, 2015.

[11] J. Ward et al. for the CTA consortium, *The Instrument Response Function Format for the Cherenkov Telescope Array*, in *these proceedings*, 2015.