PROCEEDINGS OF SCIENCE

Gammapy – A prototype for the CTA science tools

Christoph Deil*[a], Julien Lefaucheur[b], Régis Terrier[c], Bruno Khélifi[c], Matthew Wood[d], Roberta Zanin[a], Lars Mohrmann[e], Nachiketa Chakraborty[a], Jason Watson[a], Rubén López Coto[a], Stefan Klepser[f], Matteo Cerruti[g], Jean-Philippe Lenain[g], Fabio Acero[h], Arache Djannati-Ataï[c], Santiago Pita[c], Zeljka Bosnjak[i], Jose Enrique Ruiz[j], Cyril Trichard[k], Thomas Vuillaume[l], for the CTA Consortium, Axel Donath[a], Johannes King[a], Léa Jouvin[c], Ellis Owen[m], Manuel Paz Arribas[n], Brigitta Sipocz[o], Dirk Lennarz[p], Arjun Voruganti[a], Marion Spir-Jacob[c]

[a] *MPIK, Heidelberg, Germany*
[b] *LUTH, Obs. de Paris/Meudon, France*
[c] *APC/CNRS, Paris, France*
[d] *SLAC National Accelerator Laboratory, US*
[e] *FAU, Erlangen, Germany*
[f] *DESY, Zeuthen, Germany*
[g] *LPNHE, Paris, France*
[h] *CEA/IRFU, Saclay, France*
[i] *University of Rijeka, Croatia*
[j] *Instituto Astrofísica de Andalucía, Granada, Spain*
[k] *CPPM, Marseille, France*
[l] *LAPP, Annecy-le-Vieux, France*
[m] *UCL-MSSL, Dorking, United Kingdom*
[n] *Humboldt University, Berlin, Germany*
[o] *Cambridge, UK*
[p] *Georgia Tech, Atlanta, US*

E-mail: Christoph.Deil@mpi-hd.mpg.de, julien.lefaucheur@obspm.fr, Roberta.Zanin@mpi-hd.mpg.de, khelifi@apc.in2p3.fr,

Gammapy is a Python package for high-level gamma-ray data analysis, written in Python and built on Numpy, Scipy and Astropy. Starting with event lists and instrument response information, it is possible to analyse gamma-ray data and to create for example sky images, spectra and lightcurves, and to determine the position, morphology and spectra of gamma-ray sources.
So far Gammapy has mostly been used to analyse data from H.E.S.S. and Fermi-LAT, and now it is being used for the simulation and analysis of observations from the Cherenkov Telescope Array (CTA). We have proposed Gammapy as a prototype for the CTA science tools. This contribution will give an overview of the Gammapy package and show analysis application examples with simulated CTA data.

35th International Cosmic Ray Conference — ICRC2017
10–20 July, 2017
Bexco, Busan, Korea

*Speaker.

1. Introduction

The Cherenkov Telescope Array (CTA) will observe the sky in very-high-energy gamma-ray light soon. All astronomers will have access to CTA high-level data, as well as CTA science tools (ST) software. The ST can be used for example to generate sky images and to measure source properties such as morphology, spectra and light curves, using event lists as well as instrument response function (IRF) and auxiliary information as input.

Gammapy is a prototype for the CTA ST, built on the scientific Python stack and Astropy, optionally using Sherpa or other packages for modeling and fitting (see Figure 1). Initially the focus was to implement the "classical TeV analysis", using 2-dimensional sky images for detection and morphology characterization, followed by spectral analysis for a given source region. A 3-dimensional analysis with a simultaneous spatial and spectral models of the gamma-ray emission, as well as background (called "cube analysis" in the following) is in development. At the moment likelihood fitting in Gammapy is done on binned data (images, spectra, cubes, lightcurves), unbinned likelihood is not implemented. Simulating observations is done by Poisson-fluctuating predicted counts according to given sky models, IRFs and observation parameters, event samplers are not implemented. So far we have not encountered important use cases that require an unbinned likelihood, but if needed for efficient analysis of some use cases, a likelihood function using unbinned events or sparse or multi-resolution maps could be added in the future.

A first study comparing spectra obtained with the classical 1D analysis and the 3D cube analysis using point source observations with H.E.S.S., to compare the methods and to validate Gammapy, is presented in [1]. Further developments and verification using data from existing Cherenkov telescope arrays such as H.E.S.S. and MAGIC, as well as simulated CTA data is ongoing. Gammapy is starting to be used for scientific studies for existing ground-based gamma-ray telescopes [2, 3], the Fermi-LAT space telescope [4], as well as for CTA [5, 6, 7].

2. Context

Before moving on to a description of Gammapy in the next section, we would like to give some context for it's development and mention some related projects. An early prototype package that was similar to Gammapy was "PyFACT: Python and FITS Analysis for Cherenkov Telescopes" [8]. It was developed in 2011/2012 and hasn't been updated since, we mainly mention it to show that the idea to build the CTA science tools as a Python package using Numpy. In 2011, the Astronomical Python community came together and created the Astropy project and package [9], which is a key factor making Python the most popular language for astronomical research codes (at least according to this informal survey [10]). Gammapy is an Astropy affiliated package, which means that where possible it uses the Astropy core package instead of duplicating it's functionality, as well as having a certain quality standard such as having automated tests and documentation for the available functionality. In recent years, several other packages have adopted the same approach, to build on Python, Numpy and Astropy. To name just a few, there is ctapipe (https://github.com/cta-observatory/ctapipe), the prototype for the low-level CTA data processing pipeline (up to DL3); Naima for modeling the non-thermal spectral energy distribution of astrophysical sources [11]; and PINT, a new software for high-precision pulsar timing (https://github.com/nanograv/PINT),

and Fermipy (https://github.com/fermipy/fermipy), a Python package that facilitates and analysis of data from the Large Area Telescope (LAT) with the Fermi Science Tools and adds some extra functionality.

We note that many other astronomy projects have chosen Python/Astropy as the basis both for their data calibration and reduction pipeline as well as the science tools used by astronomers. Some prominent examples are the Hubble space telescope (HST) (TODO: add reference), the upcoming James Webb Space Telescope (JWST) (TODO: add reference) and the Chandra X-ray observatory [12, 13]. Even projects like LSST that started their analysis software developments before Astropy existed and are based on C++/SWIG are now actively looking for ways to collaborate and make their software stack interoperate with Numpy and Astropy to avoid code duplication, but also to leverage the fact that a large fraction of the astronomical community already knows and is using Astropy [14].

Another open-source package that has been proposed as a prototype for the CTA science tools is Gammalib/ctools [15]. Gammalib is a C++ library with SWIG Python wrappers that doesn't have any dependencies besides CFITSIO, instead implementing the functionality needed from scratch. ctools is a set of command line tools, following the FTOOLs model of using FITS files as input and output and being able to chain them into analysis chains using Python. Concerning analysis methods, Gammalib/ctools is supporting binned and unbinned likelihood analysis and implemented "cube analysis" first, adding support for "classical analysis" now (the other way around compared to Gammapy).

The choice for the official CTA science tools, supported and distributed by the CTA observatory, has not been made yet; at this time both Gammapy and Gammalib/ctools are open-source codes being used for CTA as well as exiting gamma-ray telescopes. One issue noticed in the past years was that in the current situation of having multiple telescopes converting their data to FITS format and multiple science tools, it became necessary to write down the details of the data formats being used. At this time, the data formats used in existing science tool codes (Gammapy, Gammalib/ctools and partly also others, like Fermipy, 3ML [16] or Naima [11]) are to a large degree the same, although one has to mention that the development of the DL3 data model and formats is work in progress, and especially the IRF formats and DL3 data linking of events to IRFs to support multiple event types will have to be extended [17], a process Gammapy is participating in and contributing to.

3. Gammapy package

Gammapy is a Python package built on Numpy [18], Scipy [19] and Astropy [9]. Optional dependencies are Scipy for integration and interpolation, and Sherpa [13, 20, 21] for modeling and fitting and Matplotlib [22] for plotting. The Gammapy dependency stack is shown in Figure 1.

The functionality is organized into sub-packages, such as for example *gammapy.data*, *gammapy.irf*, *gammapy.spectrum*, ... The Gammapy features are described in detail in the Gammapy documentation at http://docs.gammapy.org and many examples given in the tutorial-style Jupyter notebooks, as well as in [23]. Here, we wanted to focus on Gammapy as a prototype software and will continue in the next section with a code example explaining how Gammapy works.

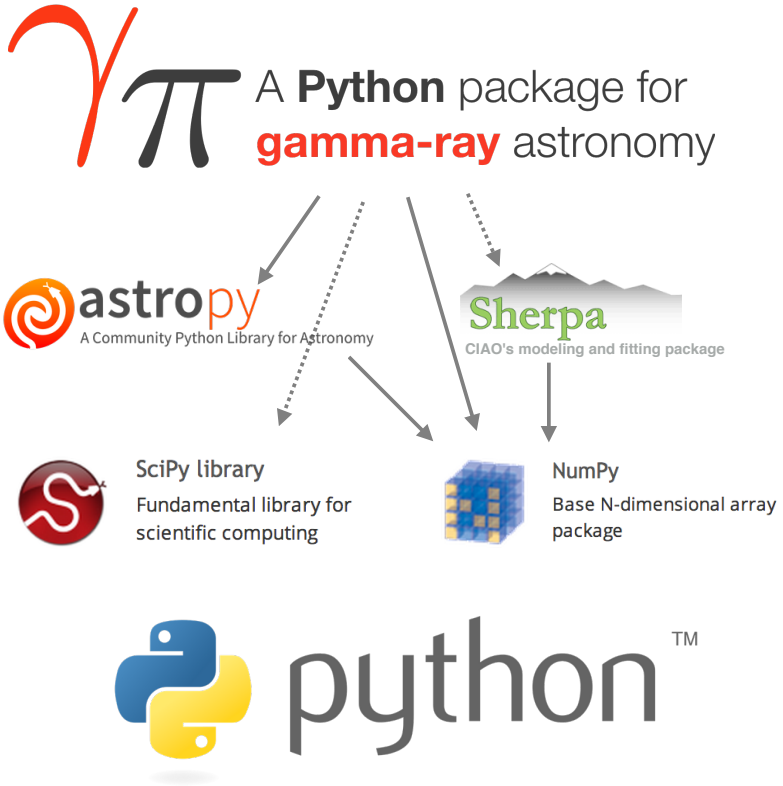

Figure 1: The Gammapy stack. Required dependencies Numpy and Astropy are illustrated with solid arrows, optional dependencies (the rest) with dashed arrows.

4. How Gammapy works: a code example

Gammapy is written high-level Python code, the data is stored in Numpy arrays or objects such as *astropy.coordinates.SkyCoord* or *astropy.table.Table* that hold Numpy array data members. Almost all functionality needed has been written in C and Python wrappers already. Specifically, *astropy.wcs* is calling into WCSLib ([24]), it astropy.io.fits uses CFITSIO ([25]) and *astropy.coordinates* as well as *astropy.time* are built on ERFA (https://github.com/liberfa/erfa), the open-source variant of this IAU Standards of Fundamental Astronomy (SOFA) C library (http://www.iausofa.org/).

An example script that generates a counts image from an event list using Gammapy is shown in Figure 2. The remarkable point we would like to make here is that it is possible to efficiently work with events and pixels and to implement algorithms from Python, by storing all data in Numpy arrays and processing via calls into existing C extensions in Numpy and Astropy. E.g. here *EventList* stores the RA and DEC columns from the event list as Numpy arrays, and *SkyImage* the pixel data as well, and *image.fill(events)*, and all processing happens in existing C extensions (Numpy histograms and Astropy calls into the CFITSIO and WCSLib C libraries). Should the need arise, that some algorithm can't be efficiently implemented in pure Python, a small C extension can be added to Gammapy, using e.g. Cython [26].

We note that a huge eco-system of scientific Python packages that operate on Numpy arrays is available for advanced users, to implement analysis methods for special use cases (e.g. sky or

```
"""Make a counts image with Gammapy."""
from gammapy.data import EventList
from gammapy.image import SkyImage
events = EventList.read('events.fits')
image = SkyImage.empty(
  nxpix=400, nypix=400, binsz=0.02,
  xref=83.6, yref=22.0,
  coordsys='CEL', proj='TAN',
)
image.fill(events)
image.write('counts.fits')
```

Figure 2: An example script using Gammapy to make a counts image from an event list. This is used in Section 4 to explain how Gammapy achieves efficient processing of event and pixel data from Python: all data is stored in Numpy arrays and passed to existing C extensions in Numpy and Astropy.

background models, IRF handling or likelihood or Bayesian analysis methods). To name just a few that we have seen used in conjunction with Gammapy so far (by users, not within the Gammapy package itself): healpy, Scipy, pandas, scikit-image, scikit-learn, iminuit, emcee.

5. Gammapy development

TODO: write me, keep it very short!

For testing we use pytest (https://pytest.org). For documentation Sphinx (http://www.sphinx-doc.org), for tutorial-style documentation Jupyter notebooks (https://jupyter.org/).

Gammapy is distributed and installed in the usual way for Python packages. Each stable release is uploaded to the Python package index (https://pypi.python.org), and downloaded and installed by users via *pip install gammapy* (https://pip.pypa.io). Binary packages for conda are available via the conda Astropy channel (https://anaconda.org/astropy/gammapy) for Linux, Mac and Windows, which conda users can install via *conda install gammapy -c astropy*. Binary packages for the Macports package manager are also available, which users can install via *port install gammapy*. Concerning Linux, at this time, Gammapy is available as a Gentoo package (https://packages.gentoo.org/packages/dev-python/gammapy) and a Debian package is in preparation.

Mailing list: https://groups.google.com/forum/#!forum/gammapy and for Gammapy developer team communication gammapy.slack.com. Roughly bi-yearly Gammapy meetings, so far focused on developers and advanced users (Heidelberg, June 2016 and Paris in Feb 2017). TODO: mention code review, continuous integration, coverage and other code quality things.

Note that the current setup described in this section is different from what it will be if Gammapy is chosen to be the or part of the CTA science tools, since CTA will set up their own systems for software development, maintenance, testing, documentation, issue tracking, support and distribution.

6. Application example

In this poster we focused on the software and technical aspects of Gammapy. For examples of

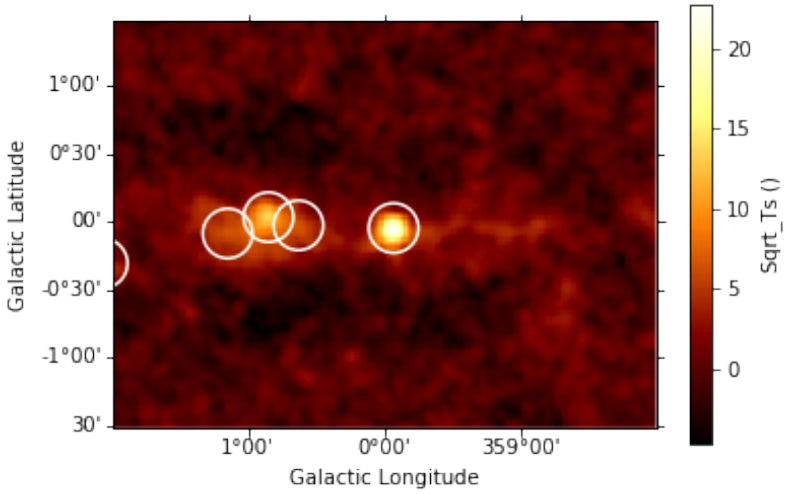

Figure 3: Application example: significance image for the Galactic centre region using 1.5 hours of simulated CTA data. White circles are peaks above 5 sigma.

CTA science studies using Gammapy, we refer you to other posters presented at this conference: Galactic survey [6], PeVatrons [7] and extra-galactic sources [5].

Several other examples using real data from H.E.S.S. and Fermi-LAT, as well as simulated data for CTA can be found via http://docs.gammapy.org by following the link to "tutorial notebooks". Figure 3 shows one result of the "CTA data analysis with Gammapy" notebook: a significance sky image of the Galactic center region using 1.5 hours of simulated CTA data. The background was estimated using the ring background estimation technique, and peaks above 5 sigma are shows with white circles.

7. Conclusions

In the past two years, we have developed Gammapy as an open-source analysis package for existing gamma-ray telescope and as a prototype for the CTA science tools.

We find that the Gammapy approach, to build on the powerful and well-tested Python packages Numpy and Astropy, brings large benefits: A small codebase that is focused on gamma-ray astronomy in a single high-level language is easy to understand and maintain. It is also easy to modify and extend as new use cases arise, which is important for CTA, since it can be expected that the modeling of the instrument, background and astrophysical emission, as well as the analysis method in general (e.g. likelihood or Bayesian statistical methods) will evolve and improve over the next decade. Last but not least, the Gammapy approach is inherently collaborative, sharing development effort as well as know-how with the larger astronomical community, that to a large degree already has adopted Numpy and Astropy as the basis for astronomical analysis codes in the past 5 years.

8. Acknowledgements

This work was conducted in the context of the CTA Consortium. We gratefully acknowledge financial support from the agencies and organizations listed here: http://www.cta-observatory.org/consortium_acknowl

We would like to thank the Scientific Python and specifically the Astropy community for providing their packages which are invaluable to the development of Gammapy, as well as tools and help with package setup and continuous integration, as well as building of conda packages.

We thank the GitHub (http://www.github.com) team for providing us with an excellent free development platform, ReadTheDocs (https://readthedocs.org) for free documentation hosting, Travis (https://www.travis-ci.org) and Appveyor (https://appveyor.com) for free continuous integration testing, and Slack (https://slack.com/) for a free team communication channel.

References

[1] L. Jouvin et al., *Toward a 3D analysis in Cerenkov gamma-ray astronomy*, in *these proceedings*, 2017.

[2] S. Carrigan et al. for the H.E.S.S. collaboration, *The H.E.S.S. Galactic Plane Survey - maps, source catalog and source population*, *ArXiv e-prints* (July, 2013) [arXiv:1307.4690].

[3] G. Puehlhofer 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 *34th ICRC (2015)*, p. 886, July, 2015. arXiv:1509.0387.

[4] 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].

[5] J. Lefaucher for the CTA consortium, *Gammapy: high level data analysis for extragalactic science cases with the Cherenkov Telescope Array*, in *these proceedings*, 2017.

[6] R. Zanin for the CTA consortium, *Observing the Galactic Plane with Cherenkov Telescope Array*, in *these proceedings*, 2017.

[7] C. Trichard for the CTA consortium, *Searching for PeVatrons in the CTA Galactic Plane Survey*, in *these proceedings*, 2017.

[8] M. Raue and C. Deil, *PyFACT: Python and FITS analysis for Cherenkov telescopes*, vol. 1505 of *American Institute of Physics Conference Series*, pp. 789–792, Dec., 2012.

[9] 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.

[10] I. Momcheva and E. Tollerud, *Software Use in Astronomy: an Informal Survey*, *ArXiv e-prints* (July, 2015) [arXiv:1507.0398].

[11] V. Zabalza, *naima: a Python package for inference of relativistic particle energy distributions from observed nonthermal spectra*, *ArXiv e-prints* (Sept., 2015) [arXiv:1509.0331].

[12] T. Aldcroft, *Keeping the chandra satellite cool with python*, in *Proceedings of the 9th Python in Science Conference*, pp. 30 – 34, 2010.

[13] P. Freeman et al., *Sherpa: a mission-independent data analysis application*, in *Astronomical Data Analysis*, vol. 4477 of *SPIE Conference Series*, pp. 76–87, Nov., 2001. astro-ph/0108426.

[14] T. Jenness et al., *Investigating interoperability of the lsst data management software stack with astropy*, vol. 9913, pp. 99130G–99130G–13, 2016.

[15] J. Knödlseder et al., *GammaLib and ctools. A software framework for the analysis of astronomical gamma-ray data*, *AAP* **593** (Aug., 2016) A1, [arXiv:1606.0039].

[16] G. Vianello et al., *The Multi-Mission Maximum Likelihood framework (3ML)*, *ArXiv e-prints* (July, 2015) [arXiv:1507.0834].

[17] C. Deil and C. Boisson et al., *Open high-level data formats and software for gamma-ray astronomy*, *ArXiv e-prints* (Oct., 2016) [arXiv:1610.0188].

[18] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, *The numpy array: a structure for efficient numerical computation*, *Computing in Science & Engineering* **13** (2011), no. 2 22–30.

[19] *SciPy: Open source scientific tools for python*, 2001.

[20] 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.

[21] B. Refsdal, S. Doe, D. Nguyen, and A. Siemiginowska, *Fitting and estimating parameter confidence limits with sherpa*, in *10th SciPy Conference*, pp. 4 – 10, 2011.

[22] J. D. Hunter, *Matplotlib: A 2d graphics environment*, *Computing In Science & Engineering* **9** (2007), no. 3 90–95.

[23] A. Donath et al., *Gammapy - A Python package for {\gamma}-ray astronomy*, *ArXiv e-prints* (Sept., 2015) [arXiv:1509.0740].

[24] M. R. Calabretta, "Wcslib and Pgsbox." Astrophysics Source Code Library, Aug., 2011.

[25] W. D. Pence, "CFITSIO: A FITS File Subroutine Library." Astrophysics Source Code Library, Oct., 2010.

[26] S. Behnel at al., *Cython: The best of both worlds*, *Computing in Science Engineering* **13** (march-april, 2011) 31 –39.