PROCEEDINGS OF SCIENCE

Gammapy – A prototype for the CTA science tools

Christoph Deil[a], **Roberta Zanin**[a], **Julien Lefaucheur**[*b], **Catherine Boisson**[b], **Bruno Khélifi**[c], **Régis Terrier**[c], **Matthew Wood**[d], **Lars Mohrmann**[e], **Nachiketa Chakraborty**[a], **Jason Watson**[q], **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], **José 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] *IAA-CSIC, 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*
[q] *University of Oxford, UK*

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

Gammapy is a Python package for high-level gamma-ray data analysis built on Numpy, Scipy and Astropy. It enables us to analyze gamma-ray data and to create sky images, spectra and lightcurves, from event lists and instrument response information, and to determine the position, morphology and spectra of gamma-ray sources.

So far Gammapy has mostly been used to analyze data from H.E.S.S. and Fermi-LAT, and is now 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 gives an overview of the Gammapy package and project and shows an analysis application example 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 (VHE, E > 20 GeV) gamma-ray light soon. CTA will consist of large telescope arrays at two sites, one in the southern (Chile) and one in the northern (La Palma) hemisphere. It will perform surveys of large parts of the sky, targeted observations on Galactic and extra-galactic sources, and more specialized analyses like a measurement of charged cosmic rays, constraints on the intergalactic medium opacity for gamma-rays and a search for dark matter. Compared to current Cherenkov telescope arrays such as H.E.S.S., VERITAS or MAGIC, CTA will have a much improved detection area, angular and energy resolution, improved signal/background classification and sensitivity. CTA is expected to operate for thirty years, and 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 [1], optionally using Sherpa [2, 3, 4] or other packages for modeling and fitting (see Figure 1). A 2-dimensional analysis of the sky images for source detection and morphology fitting, followed by spectral analysis, was first implemented. A 3-dimensional analysis with a simultaneous spatial and spectral model of the gamma-ray emission, as well as background (called "cube analysis" in the following) is under development. 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. is presented in [5]. 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 now used for scientific studies with existing ground-based gamma-ray telescopes [6, 7], the Fermi-LAT space telescope [8], as well as for CTA [9, 10, 11].

In this writeup we focus on the software and technical aspects of Gammapy.We start with a brief overview of the context in Section 2, followed by a description of the Gammapy package in Section 3, the Gammapy project in Section 4 and finally our conclusions concerning Gammapy as as CTA science tool prototype in Section 5.

2. Context

Prior to Gammapy, in 2011/2012, a collection of Python scripts for the analysis of IACT data was released as a first test of open-source VHE data analysis software: "PyFACT: Python and FITS Analysis for Cherenkov Telescopes" [12]. This project is not updated anymore, but it was the first implementation of our idea to build a CTA science tools package based on Python and using Numpy and Scipy, during the first development stages of the CTA project. In 2011, the astronomical Python community came together and created the Astropy project and package [1], which is a key factor making Python the most popular language for astronomical research codes (at least according to this informal analysis [13] and survey [14]). Gammapy is an Astropy affiliated package, which means that where possible it uses the Astropy core package instead of duplicating its 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

Figure 1: The Gammapy software stack. Required dependencies (Python, Numpy and Astropy) are illustrated with solid arrows, optional dependencies (Scipy and Sherpa) with dashed arrows.

the same approach, to build on Python, Numpy and Astropy. To name just a few, there is ctapipe (github.com/cta-observatory/ctapipe), the prototype for the low-level CTA data processing pipeline (up to the creation of lists of events and IRFs); Naima for modeling the non-thermal spectral energy distribution of astrophysical sources [15]; PINT, a new software for high-precision pulsar timing (github.com/nanograv/PINT), and Fermipy (github.com/fermipy/fermipy), a Python package that facilitates 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 and Astropy as the basis both for their data calibration and reduction pipelines and their science tools. Some prominent examples are the Hubble space telescope (HST) [16], the upcoming James Webb Space Telescope (JWST) [17] and the Chandra X-ray observatory [2, 18]. Even projects like LSST that started their analysis software developments before Astropy existed and are based on C++/SWIG are now actively working towards making their software interoperable with Numpy and Astropy, to avoid duplication of code and development efforts, but also to reduce the learning curve for their science tool software (since many astronomers already are using Python, Numpy and Astropy) [19].

For current ground-based IACTs, data and software are mostly private to the collaborations operating the telescopes. CTA will be, for the first time in VHE gamma-ray astronomy, operated as an open observatory. This implies fundamentally different requirements for the data formats and software tools. Along this path, the current experiments, H.E.S.S., MAGIC and VERITAS, have started converting their data to FITS format, yet relying on different (some private, some open) analysis tools, and many slightly different ways to store the same information in FITS files appeared. The CTA high-level data model and data format specifications are currently being written and will give a framework to the current experiments to store data. The methods to link events to IRFs still have to be extended to support multiple event types and the IRF and background model

formats will have to be extended to be more precise [20]. The Gammapy team is participating in and contributing to the effort to prototype and find a good high-level data model and formats for CTA.

3. Gammapy package

Gammapy offers the high-level analysis tools to generate science results (images, spectra, light curves and source catalogs) based on input data consisting of reconstructed events (with an arrival direction and an energy) that are classified according to their types (e.g. gamma-like, cosmic-ray-like). In the data processing chain of CTA, the reconstruction and classification of events are realized by another Python package, ctapipe[1]. Data are stored in FITS format. The high-level analysis consists of:

- selection of a data cube (energy and positions) around a sky position from all event lists,
- computation of the corresponding exposure,
- estimation the background directly from the data (e.g. with a ring background model [21]) or from a model (e.g. templates built from real data beforehand),
- creation of sky images (signal, background, significance, etc) and morphology fitting,
- spectrum measurement with a 1D analysis or with a 3D analysis by adjusting both spectral and spatial shape of gamma-ray sources,
- computation of light-curves and phasograms, search for transient signals,
- derivation of a catalog of excess peaks (or a source catalog).

For such an analysis, Gammapy is using the IRFs produced by the ctapipe package and processes them precisely to get an accurate estimation of high-level astrophysical quantities.

The Gammapy code base is structured into several sub-packages dedicated to specific classes where each of the packages bundle corresponding functionality in a namespace (e.g. data and observation handling in *gammapy.data*, IRF functionality in *gammapy.irf*, spectrum estimation and modeling in *gammapy.spectrum* ...). The Gammapy features are described in detail in the Gammapy documentation (docs.gammapy.org) and many examples given in the tutorial-style Jupyter notebooks, as well as in [22]. 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 shown with white circles. For examples of CTA science studies using Gammapy, we refer you to other posters presented at this conference: Galactic survey [10], PeVatrons [11] and extra-galactic sources [9]. 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 docs.gammapy.org by following the link to "tutorial notebooks".

The Gammapy Python package is primarily built on Numpy [23], and Astropy [1] as core dependencies. Data is stored in Numpy arrays or objects such as *astropy.coordinates.SkyCoord* or *astropy.table.Table* that hold Numpy array data members. Numpy provides many functions for array-oriented computing and numerics, and Astropy provides astronomy-specific functionality. The Astropy functionality most commonly used in Gammapy is *astropy.io.fits* for FITS data I/O, *astropy.table.Table* as a container for tabular data (e.g. event lists, but also many other things like

[1] github.com/cta-observatory/ctapipe

```
"""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(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 3 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.

spectral points or source catalogs), *astropy.wcs.WCS* for world coordinate systems mapping pixel to sky coordinates, as well as *astropy.coordinates.SkyCoord* and *astropy.time.Time* objects to represent sky coordinates and times. Astropy.coordinates as well as astropy.time are built on ERFA (github.com/liberfa/erfa), the open-source variant of this IAU Standards of Fundamental Astronomy (SOFA) C library (www.iausofa.org). In Gammapy, we use *astropy.units.Quantity* objects extensively, where a quantity is a Numpy array with a unit attached, supporting arithmetic in computations and making it easier to read and write code that does computations involving physical quantities.

As an example, a script that generates a counts image from an event list using Gammapy is shown in Figure 2. The point we want 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 implemented or wrapped in Numpy and Astropy. In this example, the *read* and *write* methods call into *astropy.io.fits* which calls into CFITSIO ([24]), and the *image.fill(events)* method calls into *astropy.wcs.WCS* and WCSLib ([25]) as well as *numpy.histogramdd*.

Gammapy aims to be a base package on which other more specialized packages such as Fermipy (github.com/fermipy/fermipy) for Fermi-LAT data analysis or Naima [15] for the modeling of non-thermal spectral energy distributions of astrophysical sources can build. For this reason we avoid introducing new required dependencies besides Numpy and Astropy. That said, Gammapy does import the following optional dependencies to provide extra functionality (sorted in the order of how common their use is within Gammapy). Scipy [26] is used for integration and interpolation, Matplotlib [27] for plotting and Sherpa [2, 3, 4] for modeling and fitting. In addition, the following packages are used at the moment for functionality that we expect to become available in the Astropy core package within the next year: *regions* (astropy-regions.readthedocs.io) to handle sky and pixel regions, *reproject* (reproject.readthedocs.io) for reprojecting sky images and cubes and *healpy* (healpy.readthedocs.io) for HEALPix data handling.

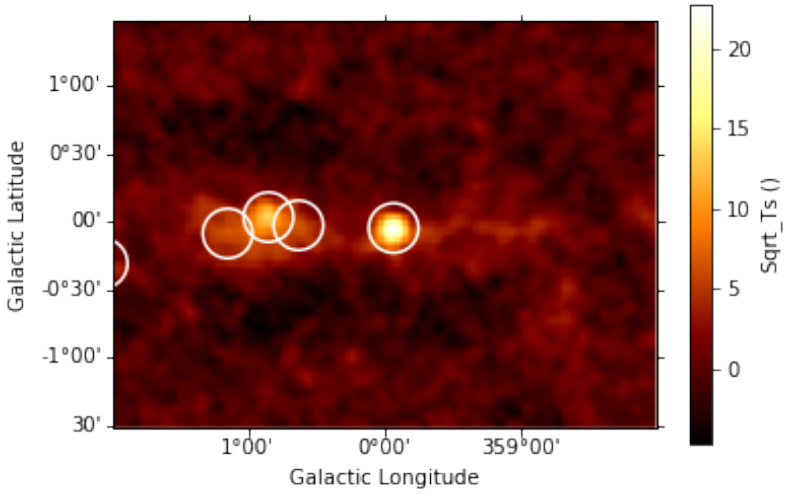

Figure 3: Application example: significance image for the Galactic centre region using 1.5 hours of simulated CTA data. White circles are peaks above 5 sigma.

4. Gammapy project

In this section we describe the current setup of the Gammapy project. We are using the common tools and services for Python open-source projects for software development, code review, testing, documentation, package distribution and user support.

Gammapy is distributed and installed in the usual way for Python packages. Each stable release is uploaded to the Python package index (pypi.python.org), and downloaded and installed by users via *pip install gammapy* (pip.pypa.io). Binary packages for conda are available via the conda Astropy channel (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*. At this time, Gammapy is also available as a Gentoo Linux package (packages.gentoo.org/packages/dev-python/gammapy) and a Debian Linux package is in preparation.

Gammapy development happens on Github (github.com/gammapy/gammapy). We make extensive use of the pull request system to discuss and review code contributions. For testing we use pytest (pytest.org), for continuous integration Travis-CI (Linux and Mac) as well as Appveyor (Windows). For documentation Sphinx (www.sphinx-doc.org), for tutorial-style documentation Jupyter notebooks (jupyter.org) are used. For Gammapy developer team communication we use Slack (gammapy.slack.com). A public mailing list for user support and discussion is available (groups.google.com/forum/#!forum/gammapy). Two face-to-face meetings for Gammapy were organized so far, the first on in June 2016 in Heidelberg as a coding sprint for developers only, the second on in February 2017 in Paris as a workshop for both Gammapy users and developers.

Gammapy is under very active development, especially in the area of modeling, and in the implementation of a simple-to-use, high-level end-user interface (either config file driven or command line tools). We will write a paper on Gammapy later this year and are working towards a version 1.0 release.

5. 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. Gammapy is a Python package, consisting of functions and classes that can be used as a flexible and extensible toolbox to implement and execute high-level gamma-ray data analyses.

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 (contributions from $\sim$30 gamma-ray astronomers so far), 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.

6. Acknowledgements

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

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

References

[1] 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.

[2] P. Freeman, S. Doe and A. Siemiginowska, *Sherpa: a mission-independent data analysis application*, in *Astronomical Data Analysis*, vol. 4477 of *SPIE*, pp. 76–87, Nov., 2001, astro-ph/0108426.

[3] 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.

[4] 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.

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

[6] 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) , [1307.4690].

[7] 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.*, vol. 34 of *ICRC*, p. 886, July, 2015, 1509.03872.

[8] 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) , [1506.02319].

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

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

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

[12] 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, DOI.

[13] P. Greenfield, *What Python Can Do for Astronomy*, vol. 442 of *Astronomical Society of the Pacific Conference Series*, pp. 425–+, July, 2011.

[14] I. Momcheva and E. Tollerud, *Software Use in Astronomy: an Informal Survey*, *ArXiv e-prints* (July, 2015) , [1507.03989].

[15] V. Zabalza, *naima: a Python package for inference of relativistic particle energy distributions from observed nonthermal spectra*, *ArXiv e-prints* (Sept., 2015) , [1509.03319].

[16] P. Greenfield and R. L. White, *Where Will PyRAF Lead Us? The Future of Data Analysis Software at STScI*, p. 437, Jan., 2006.

[17] H. Bushouse et al., *The James Webb Space Telescope Data Calibration Pipeline* , in *Proceedings of the 14th Python in Science Conference*, pp. 44 – 48, 2015.

[18] T. Aldcroft, *Keeping the Chandra Satellite Cool with Python*, in *Proceedings of the 9th Python in Science Conference*, pp. 30 – 34, 2010.

[19] T. Jenness et al., *Investigating interoperability of the LSST data management software stack with Astropy*, vol. 9913, pp. 99130G–99130G–13, 2016, DOI.

[20] C. Deil and C. Boisson et al., *Open high-level data formats and software for gamma-ray astronomy*, *ArXiv e-prints* (Oct., 2016) , [1610.01884].

[21] D. Berge, S. Funk and J. Hinton, *Background modelling in very-high-energy γ-ray astronomy*, *AAP* **466** (May, 2007) 1219–1229, [astro-ph/0610959].

[22] A. Donath et al., *Gammapy - A Python package for gamma-ray astronomy*, *ArXiv e-prints* (Sept., 2015) , [1509.07408].

[23] 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) 22–30.

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

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

[26] *SciPy: Open source scientific tools for Python*, 2001.

[27] J. D. Hunter, *Matplotlib: A 2D graphics environment*, *Computing In Science & Engineering* **9** (2007) 90–95.