Gammapy – A prototype for the CTA science tools

C. Deil[1], J. Lefaucheur[2], R. Terrier[3], B. Khélifi, M. Wood, R. Zanin, L. Mohrmann, N. Chakraborty, J. Watson, R. López Coto, S. Klepser, M. Cerruti, J.-P. Lenain, F. Acero, A. Djannati-Ataï, S. Pita, Z. Bosnjak, J.E. Ruiz, C. Trichard, T. Vuillaume for the CTA Consortium, A. Donath[1], J. King[1], L. Jouvin[1], E. Owen[1], M. Paz Arribas[1], B. Sipocz[1], D. Lennarz[1], Al Voruganti[1], M. Spir-Jacob[1]

[1]MPIK, [2]LUTH, [3]APC, [4]SLAC, [5]FAU, [6]DESY, [7]LPNHE, [8]CEA, [9]Rijeka University, [10]IAA, [11]CPPM, [12]LAPP, [13]UCL-MSSL, [14]Humboldt University, [15]Cambridge, [16]Georgia Tech. See www.cta-observatory.org

cherenkov telescope array

Gammapy – A Python package for gamma-ray astronomy

ABSTRACT

Gammapy is a Python package for gamma-ray astronomy, built on Numpy and Astropy, and a prototype for the Cherenkov Telescope Array (CTA) science tools.
Here we give an introduction to Gammapy and show an application example how to create a sky image using simulated CTA event data and instrument response functions.
For further information, visit docs.gammapy.org and follow the links to “tutorials” and “CTA”.

Supported by:

Max-Planck-Institut für Kernphysik Heidelberg · l'Observatoire de Paris · APC, CNRS, France

CTA science tools

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

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 characterisation, 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 is in development. 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.

Getting started

To install Gammapy, use one of

```
pip install gammapy
conda install -c astropy gammapy
```

then visit docs.gammapy.org and follow the links to “tutorials” and “CTA”.

If you have any questions, issues or feature requests, contact the Gammapy mailing list or issue tracker.

ACKNOWLEDGEMENTS

We gratefully acknowledge support from the agencies and organizations under *Funding Agencies* at www.cta-observatory.org

Fig. 1: The Gammapy approach – use Python, Numpy, Scipy, Astropy, Sherpa

```python
"""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')
```

Fig. 2: Code example – efficiently work with events and pixels from Python, by always storing data in Numpy arrays

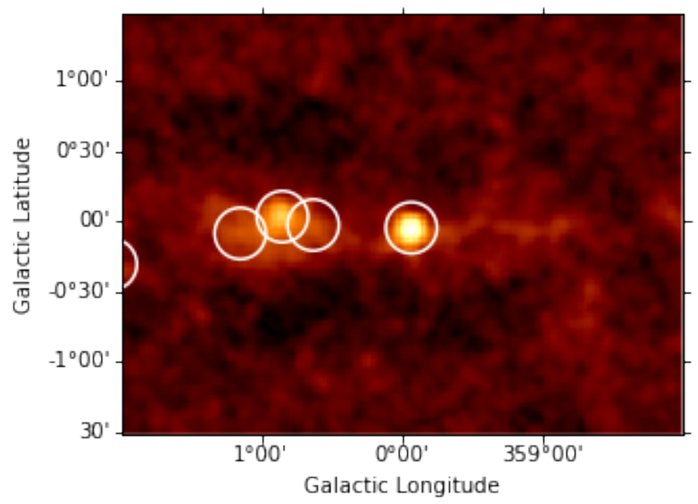

Fig.3: Application example – significance image for the Galactic centre region using 1.5 hours of simulated CTA data. White circles are peaks above 5 sigma.

Behind the scenes …

Figure 2 shows a Gammapy code example that generates a counts image from an event list. 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 eventlist as 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 (using e.g. Cython).

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.