©ESO 2022

Gammapy: A Python package for gamma-ray astronomy

Axel Donath[7], Christoph Deil[14], Régis Terrier[3], Johannes King[23], Jose Enrique Ruiz[18], Quentin Remy[19], Léa Jouvin[30], Atreyee Sinha[27], Matthew Wood[12], Fabio Pintore[15], Manuel Paz Arribas[30], Laura Olivera[19], Luca Giunti[3], Bruno Khelifi[4], Ellis Owen[22], Brigitta Sipőcz[6], Olga Vorokh[30], Julien Lefaucheur[30], Fabio Acero[8], Thomas Robitaille[2], David Fidalgo[30], Jonathan D. Harris[30], Cosimo Nigro[16], Lars Mohrmann[19], Dirk Lennarz[1], Jalel Eddine Hajlaoui[3], Alexis de Almeida Coutinho[28], Matthias Wegenmat[30], Dimitri Papadopoulos[30], Maximilian Nöthe[26], Nachiketa Chakraborty[30], Michael Droettboom[21], Helen Poon[30], Arjun Voruganti[30], Jason Watson[9], Thomas Armstrong[30], Vikas Joshi[13], Erik Tollerud[25], Erik M. Bray[30], Domenico Tiziani[30], Gabriel Emery[30], Hubert Siejkowski[30], Kai Brügge[24], Luigi Tibaldo[17], Arpit Gogia[30], Ignacio Minaya[30], Marion Spir-Jacob[30], Yves Gallant[30], Andrew W. Chen[10], Roberta Zanin[5], Jean-Philippe Lenain[30], Larry Bradley[25], Kaori Nakashima[13], Anne Lemière[3], Mathieu de Bony[30], Matthew Craig[30], Lab Saha[7], Zé Vinicius[30], Kyle Barbary[30], Thomas Vuillaume[29], Adam Ginsburg[30], Daniel Morcuende[30], José Luis Contreras[30], Laura Vega Garcia[30], Oscar Blanch Bigas[30], Víctor Zabalza[30], Wolfgang Kerzendorf[20], Rolf Buehler[11], Sebastian Panny[30], Silvia Manconi[30], Stefan Klepser[11], Peter Deiml[30], Johannes Buchner[30], Hugo van Kemenade[30], Eric O. Lebigot[30], Benjamin Alan Weaver[30], Debanjan Bose[30], Rubén López-Coto[30], and Sam Carter[30]

(Affiliations can be found after the references)

March 12, 2022

ABSTRACT

Historically the data as well as analysis software in gamma-ray astronomy is proprietary to the experiments. With the future Cherenkov Telescope Array (CTA), which will be operated as an open gamma-ray observatory with public data, there is a corresponding need for open high-level analysis software. In this article we present the first major version v1.0 of Gammapy, a community-developed open-source Python package for gamma-ray astronomy. We present its general design and provide anoverview of the analysis methods and features it implements. Starting from event lists and a description of the specific instrument response functions (IRF) stored in open FITS based data formats, Gammapy implements . Thereby it handles the dependency of the IRFs with time, energy as well as position on the sky. It offers a variety of background estimation methods for spectral, spatial and spectro-morphological analysis. Counts, background and IRFs data are bundled in datasets and can be serialised, rebinned and stacked. Gammapy supports to model binned data using Poissonmaximum likelihood fitting. It comes with built-in spectral, spatial and temporal models as well as support for custom user models, to model e.g., energy dependent morphology of gamma-ray sources. Multiple datasets can be combined in a joint-likelihood approach to either handle time dependent IRFs, different classes of events or combination of data from multiple instruments. Gammapy also implements methods to estimate flux points, including likelihood profiles per energy bin, light curves as well as flux and signficance maps in energy bins. We further describe the generaldevelopment approach and how Gammapy integrates into ecosystem of other scientific and astronomical Python packages. We also present analysis examples with simulated CTA data and provide results of scientific validation analyses using data of existing instruments such as H.E.S.S. and Fermi-LAT .

Key words. Gamma rays: general - Astronomical instrumentation, methods and techniques - Methods: data analysis

1. Introduction

TODO: Axel and Regis write this...

Gamma-ray astronomy is a rather young field of research. By detecting and reconstructing arrival direction, time and energy of primary cosmic gamma-rays The gamma-ray sky is either observed by ground based instruments, driven by experiments with proprietary software often based on ROOT, because of the particle physics background. Such as HESS, Veritas or Magic.

The Cherenkov Telescope Array (CTA) will be operated as an open observatory for the first time. Thus there is a need for open analysis software as well.

Moreover, the operation of CTA as an observatory introduces the necessity of sharing its data publicly. The data-reduction workflow of different IACTs of the current generation is remarkably similar, resulting in a high data level that can be finally used to derive scientific results (, spectra, sky maps, light curves). The information in this high data level is independent on the data reduction, and eventually of the detection technique. This implies, for example, that data from IACT and WCD can be represented within the same model. The efforts to prototype the future CTA data model and to convert current IACT data in a format usable with the newly-available science tools converged in the so-called Data Format for Gamma-ray Astronomy initiative (Deil et al. 2017; Nigro et al. 2021a), abbreviated in gamma-astro-data-format (GADF). The latter proposes prototypical specifications to produce files based on the flexible image transport system (FITS) format (Pence et al. 2010) encapsulating this high-level information. This is realized by storing a list of gamma-like events with their measured quantites (energy, direction, arrival time) and a parametrisation of the response of the system (see Sec. 3.2 and Sec. 3.3 for more information).

In recent years Python [1] has established as one of the standard programming languages for astronomy [2] as well as data sciences in general [3]. The success is mostly attributed to the simple and easy to learn syntax, the ability to act as a "glue" language between different programming languages, the rich eco-system of packages and the open and supportive community.

Astronomical data analysis software written in Python existed since 2000. e.g., sherpa (Refsdal et al. 2011, 2009), or for gamma-ray even PyFACT (Raue & Deil 2012).

The short-term success of Pythion lead to a prolifaration of packages, until Astropy (Astropy Collaboration et al. 2013) was created in 2012. Astropy is and Gammapy is a Python package for gamma-ray astronomy.

TODO: Figure 1: Data -> Gammapy -> Spectra etc with some details

Basic idea: build on Numpy and Astropy, use Python stack

TODO: Figure 2: Gammapy software stack

Here's a list of references I'd like to cite ... to be incorporated into the main text somewhere:

– Gammapy webpage[4]
– Naima[5] (Zabalza 2015)
– Gammapy use in science publications: (Owen et al. 2015), SNR shell, HGPS

* Gammapy – A Python package for gamma-ray astronomy * Gammapy – A prototype for the CTA science tools * Astropy: A community Python package for astronomy * THE ASTROPY PROJECT: BUILDING AN INCLUSIVE, OPEN-SCIENCE PROJECT AND STATUS OF THE V2.0 CORE PACKAGE * GammaLib and ctools * Fermipy proceedings * SunPy: Python for Solar Physics. An implementation for local correlation tracking *

2. Analysis Workflow Overview

TODO: Regis and Fabio F.

3. Gammapy package

The Gammapy package is structured into multiple subpackages which mostly follow the stages in the data reduction workflow.

3.1. Overview

Outline: * List typical analysis use cases * Can use from Python and Jupyter -> show Figure with Jupyter notebook here. * Gammapy code structure * How Numpy and Astropy is used

[1] http://fits.gsfc.nasa.gov/
[2] Citation missing
[3] Citation missing
[4] http://gammapy.org
[5] https://github.com/zblz/naima

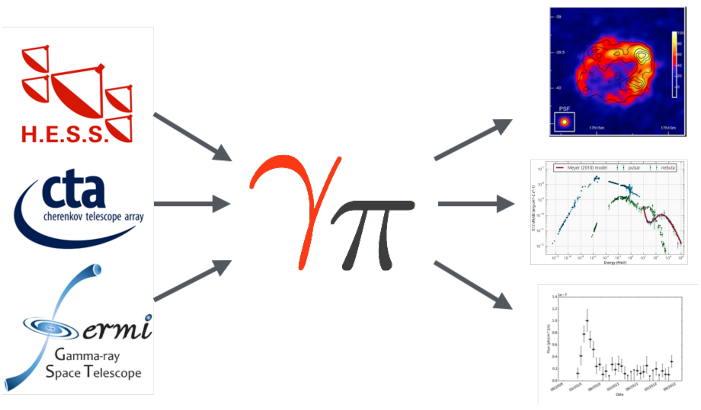

Fig. 1. Gammapy is a Python package for high-level gamma-ray data analysis. Using event lists, exposures and point spread functions as input you can use it to generate science results such as images, spectra, light curves or source catalogs. So far it has been used to simulate and analyse H.E.S.S., CTA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

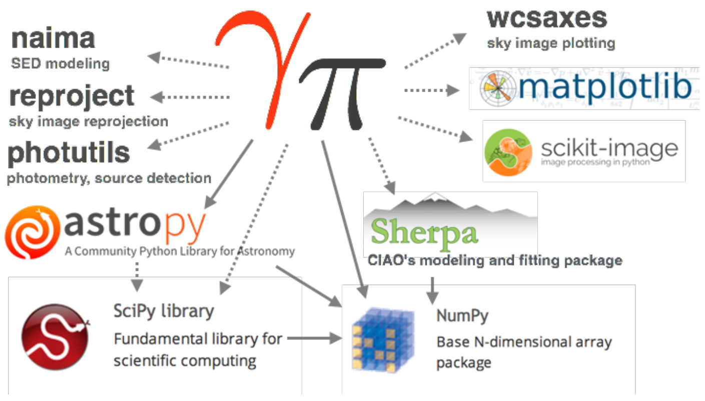
naima
SED modeling
reproject
sky image reprojection
photutils
photometry, source detection
astropy
A Community Python Library for Astronomy
γπ
wcsaxes
sky image plotting
matplotlib
scikit-image
image processing in python
Sherpa
CIAO's modeling and fitting package
SciPy library
Fundamental library for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

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

Figures: * Add a Figure showing dataflow in a typical application DL3 at the top, spectrum, map, lightcurve, fit results at the bottom. Mention major classes in between (DataStore, EventList, Map, MapMaker, MapFit, ...) * Probably not: Figure showing sub-packages and how they relate (gammapy.data and gammapy.irf at the base, then gammapy.maps, etc. * The code example Figure how to make a counts map, to explain how the package works.

TODO: How to sort the sub-packages? After data flow or alphabetically? What about maps?

3.2. gammapy.data

TODO: Cosimo Nigro

As illustrated in Fig. 3, the gammapy.data sub-package implements the lowest data level, providing the input interface to gamma-ray data. The Gammapy data model follows the GADF specifications (Sec. 1), therefore gamma-ray data compliant with those can be directly read with the package. Version X.X of the GADF specifica-

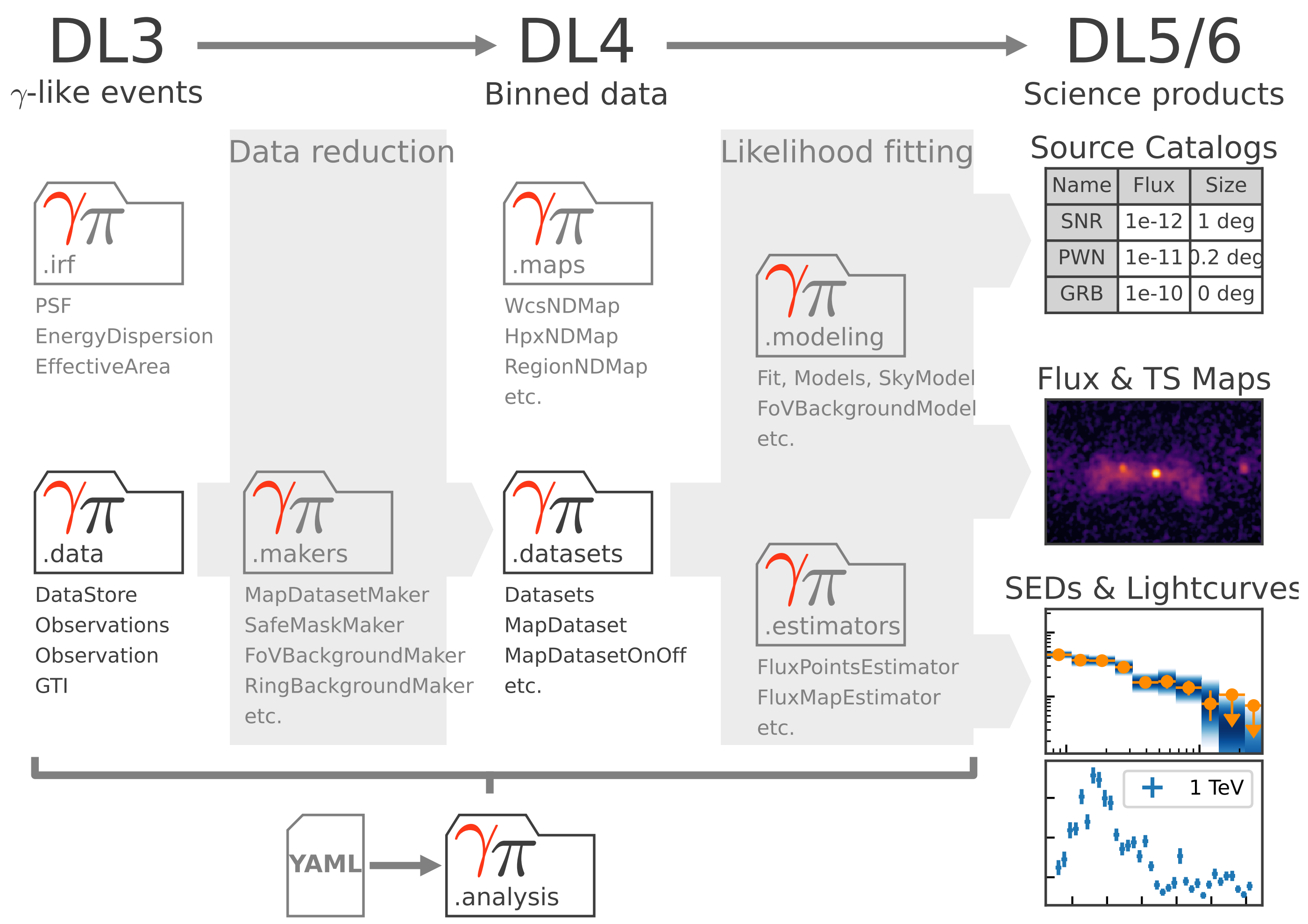

Fig. 3. Gammapy sub-package structure and data analysis workflow.

tion is supported in this release version. As both the GADF data model and Gammapy were initially conceived for IACT data analysis, input data are typically grouped in Observations, corresponding to stable periods of data acquisitions (typically 20 – 30 min). A DataStore object, gathering a collection of observations, can be created porviding ancillary files containing information about the telescope observation mode and the content of the data unit of each file. The DataStore allows for selecting a list of observations based on specific filters. As an illustrative example, Fig. 4 shows how to create a datastore and how to obtain the observations corresponding to given identification numbers (IDs).

The so-called DL3 files represented by the Observation class consist of two types of elements: a list of gamma-ray events with relevant physical quantities for the successive analysis (estimated energy, direction and arrival times) that is handled by the EventList class and an instrument response function (IRF), providing the response of the system, typically factorised in independent components (see the description in Sec. 3.3). The separate handling of event lists and IRFs additionally allows for data from other gamma-ray instruments to be read. For example, to read Fermi-LAT data, the user can read separately their event list (already compliant with the GADF specifications) and then find the appropriate IRF class representing the response functions provided by Fermi-LAT , see Sec. ??.

```
from gammapy.data import DataStore

data_store = DataStore.from_dir("$GAMMAPY_DATA")
obs_ids = [1, 2, 3]
observations = data_store.get_observations(obs_ids)
```

Fig. 4. Using gammapy.data to access DL3 level data with a DataStore

3.3. gammapy.irf

TODO: Fabio Pintore IRF classes

3.4. gammapy.maps

TODO: Laura Olivera-Nieto The gammapy.maps sub-package provides classes for representing data structures associated with a set of coordinates or a region on a sphere. In addition it allows to handle an arbitrary number of non-spatial data dimensions, such as time or energy. It is organized around three types of structures: geometries, sky-maps and map axes, which inherit from the base classes Geom, Map and MapAxis respectively.

The geometry defines the pixelization scheme and map boundaries. It also provides methods to transform between sky and pixel coordinates. Maps consist of a geometry instance together with a data array containing the corresponding map values. Map axes contain a sequence of ordered values which define bins on a given dimension, spatial or not. Map axes can have physical units attached to them, as well as non-linear step sizes.

The sub-package provides a uniform API for the FITS World Coordinate System (WCS) (Calabretta & Greisen 2002), the HEALPix pixelization (Górski et al. 2005) and region-based data structures.

- WCS: The FITS WCS pixelization supports a different number of projections to represent celestial spherical coordinates in a regular rectangular grid. Gammapy provides full support to data structures using this pixelization scheme.
- HEALPix: This pixelization provides a subdivision of a sphere in which each pixel covers the same surface area as every other pixel. As a consequence, however, pixel shapes are no longer rectangular, or regular. Gammapy provides limited support to HEALPix-based maps, relying in some cases on projections to a local WCS grid.
- Region geometries: In this case, instead of a fine spatial grid dividing a rectangular sky region, the spatial dimension is reduced to a single bin with an arbitrary shape, describing a region in the sky with that same shape. Typically they are is used together with a non-spatial dimension, for example an energy axis, to represent how a quantity varies in that dimension inside the corresponding region.

Additionally, the MapAxis class provides a uniform API for axes representing bins on any physical quantity, such as energy or angular offset. The special case of time is covered by the dedicated TimeMapAxis, which allows time bins to be non-contiguous, as it is often the case with observation time-stamps. The generic class LabelMapAxis allows the creation of axes for non-numeric entries.

3.5. gammapy.makers

TODO: Regis Terrier

The data reduction contains all tasks required to process and prepare data at the DL3 level for modeling and fitting. The gammapy.makers sub-package contains the various classes and functions required to do so. First, events are binned and IRFs are interpolated and projected onto the chosen analysis geometry. This produces counts, exposure, background, psf and energy dispersion maps. The MapDatasetMaker and SpectrumDatasetMaker are responsible for this task, see Fig 6.

Because the background models suffer from strong uncertainties it is required to correct them from the data themselves. Several techniques are commonly used in gamma-ray astronomy such as field-of-view background normalization or background measurement in reflected regions regions, see Berge et al. (2007). Specific Makers such as the FoVBackgroundMaker or the ReflectedRegionsBackgroundMaker are in charge of this step.

```
from gammapy.maps import Map, MapAxis
from astropy.coordinates import SkyCoord
from astropy import units as u

skydir = SkyCoord("0d", "5d", frame="galactic")
energy_axis = MapAxis.from_bounds(lo_bnd=0.1, hi_bnd=100,
               nbin=5, unit='TeV', interp='log', name='energy')

# Create a WCS Map
m_wcs = Map.create(
    binsz=0.1,
    map_type="wcs",
    skydir=skydir,
    width=[10.0, 8.0] * u.deg,
    axes = [energy_axis])


# Create a HEALPix Map
m_hpx = Map.create(
    binsz=0.1,
    map_type="hpx",
    skydir=skydir,
    width=10.0 * u.deg,
    axes = [energy_axis])

# Create a region map
region = "galactic;circle(0, 5, 1)"
m_region = Map.create(
    region = region,
    map_type='region',
    axes = [energy_axis])
```

Fig. 5. Using gammapy.maps to create a WCS, a HEALPix and a region map. Note the uniform API for map creation.

```
from gammapy.makers import MapDatasetMaker, FoVBackground

maker = MapDatasetMaker()
bkg_maker = FoVBackgroundMaker()
mask_maker = SafeMaskMaker()

dataset = maker.run(dataset, observation)
dataset = mask_maker.run(dataset,observation)
dataset = bkg_maker.run(dataset,observation)
```

Fig. 6. Using gammapy.makers to reduce DL3 level data into a Dataset.

Finally, to limit other sources of systematic uncertainties, a data validity domain is determined by the SafeMaskMaker. It can be used to limit the extent of the field of view used or to limit the energy range to e.g., a domain where the energy reconstruction bias is below a given value.

3.6. gammapy.datasets

TODO: Atreyee Sinha

The end product of the data reduction process described in Section 3.5 are a set of binned counts, background and IRF maps, at the DL4 level. The gammapy.datasets subpackage contains classes to bundle together binned data

```
from gammapy.datasets import MapDataset, FluxPointsDataset, Datasets

dataset1 = MapDataset.read(
    "$GAMMAPY_DATA/cta-1dc-gc/cta-1dc-gc.fits.gz", name="map_dataset"
)
dataset2 = FluxPointsDataset.read(
    "$GAMMAPY_DATA/tests/spectrum/flux_points/diff_flux_points.fits",
    name="fluxpoints_dataset",
)
datasets = Datasets([dataset1, dataset2])
print(datasets)
```

Fig. 7. A Datasets container with FluxPointsDataset and MapDataset

along with associated models and the likelihood, which provides an interface to the Fit class (Sec 3.7) for modeling and fitting purposes. Depending upon the type of analysis and the associated statistics, different types of Datasets are supported. MapDataset is used for 3D (spectral and morphological) fitting, and a 1D spectral fitting is done using SpectrumDatastet. While the default statistics for both of these is Cash, their corresponding OnOff versions are adapted for the case where the background is measured from real off counts, and support wstat statistics. The predicted counts are computed by convolution of the models with the associate IRFs. Fitting of precomputed flux points is enabled through FluxPointsDatasets, using chi2 statistics. Multiple datasets of same or different types can be bundled together in Datasets (e.g., Figure 7), where the likelihood from each constituent member is added, thus facilitating joint fitting across different observations, and even different instruments across different wavelengths. Datasets also provide functionalities for manipulating reduced data, eg: stacking, sub-grouping, plotting, etc. Users can also create their customized datasets for implementing modified likelihood methods.

3.7. gammapy.modeling

TODO: Quentin Remy

gammapy.modeling contains all the functionality related to modeling and fitting data. This includes spectral, spatial and temporal model classes, as well as the fit and parameter API.

3.8. Models

The models are grouped into the following categories:

- SpectralModel: models to describe spectral shapes of sources
- SpatialModel: models to describe spatial shapes (morphologies) of sources
- TemporalModel: models to describe temporal flux evolution of sources, such as light and phase curves

The models follow a naming scheme which contains the category as a suffix to the class name.

The Spectral Models include a special class of Normed models, which have a dimension-less normalisation. These spectral models feature a norm parameter instead of amplitude and are named using the NormSpectralModel suffix.

```
from gammapy.modeling.models import (
        SkyModel,
        PowerLawSpectralModel,
        PointSpatialModel,
)

pwl = PowerLawSpectralModel()
point = PointSpatialModel()

model = SkyModel(
        spectral_model=pwl,
        spatial_model=point,
        name="my-model",
)
```

Fig. 8. Using gammapy.modeling.models

They must be used along with another spectral model, as a multiplicative correction factor according to their spectral shape. They can be typically used for adjusting template based models, or adding a EBL correction to some analytic model. The analytic Spatial models are all normalized such as they integrate to unity over the sky but the template Spatial models may not, so in that special case they have to be combined with a NormSpectralModel.

The SkyModel is a factorised model that combine the spectral, spatial and temporal model components (by default the spatial and temporal components are optional). SkyModel objects represents additive emission components, usually sources or diffuse emission, although a single source can also be modeled by multiple components. To handle list of multiple SkyModel components, Gammapy has a Models class.

The model gallery provides a visual overview of the available models in Gammapy. Most of the analytic models commonly used in gamma-ray astronomy are built-in. We also offer a wrapper to radiative models implemented in the Naima package ?. The modeling framework can be easily extended with user-defined models. For example agnpy models that describe leptonic radiative processes in jetted Active Galactic Nuclei (AGN) can wrapped into gammapy (see section3.5 of Nigro et al. 2021b) .

3.9. Fit

The Fit class provides methods to fit i.e., optimise parameters and estimate parameter errors and correlations. It interfaces with a Datasets object, which in turn is connected to a Models object containing the model parameters in its Parameters object. Models can be unique for a given dataset, or contribute to multiple datasets and thus provide links, allowing e.g., to do a joint fit to multiple IACT datasets, or to a joint IACT and Fermi-LAT dataset. Many examples are given in the tutorials.

The Fit class provides a uniform interface to multiple fitting backends: "minuit" (Dembinski & et al. 2020), "scipy", (Virtanen et al. 2020), and "sherpa" (?Refsdal et al. 2011). Note that, for now, covariance matrix and errors are computed only for the fitting with MINUIT. However depending on the problem other optimizers can better perform, so sometimes it can be useful to run a pre-fit with alternative optimization methods. In future we plan to extend

```python
from gammapy.datasets import MapDataset
from gammapy.estimators import TSMapEstimator
from astropy import units as u

dataset = MapDataset.read(
    "$GAMMAPY_DATA/cta-1dc-gc/cta-1dc-gc.fits.gz"
)

estimator = TSMapEstimator(
    energy_edges=[0.1, 1, 10] * u.TeV
)

maps = estimator.run(dataset)
```

Fig. 9. Using the TSMapEstimator from gammapy.estimators to compute a sqrt(TS) map.

the supported Fit backend, including for example MCMC solutions. [6]

3.10. gammapy.stats

TODO: Regis Terrier Statistics methods

3.11. gammapy.estimators

TODO: Axel Donath The gammapy.estimators sub-module features methods to compute flux points, light curves, flux maps, flux profiles from data.

The initial fine binning of MapDataset is grouped into larger bins.

Internal representation with a reference spectral model and an array of normalisation values given in energy, time and spatial bins.

No unfolding.

- Regrouping of dataset bins in time, energy etc.
- Reference spectral model scaled in energy bin
- "Forward folding" / "Backward folding": but there is no difference between the two, backwards folding is forward folding with one bin
- Likelihood profiles
- Uniform N-dimensional data structure
- Uniform API for plotting etc.
- FluxMaps and FluxPoints
- Serialisation to multiple formats, Astropy's Table and BinnedTimeSeries
- Additional quantities for debugging, such as predicted counts, fit convergence, sum of fit statistics
-

3.12. gammapy.analysis

TODO: Jose Enrique writes this... High level analysis API

3.13. gammapy.visualisation

TODO: Axel Donath Plotters etc.

[6] a prototype is available in gammapy-recipes, https://gammapy.github.io/gammapy-recipes/_build/html/notebooks/mcmc-sampling-emcee/mcmc_sampling.html

```python
from gammapy.catalog import SourceCatalog4FGL

catalog = SourceCatalog4FGL()
print("Number of sources :", len(catalog.table))

source = catalog["PKS 2155-304"]
model = source.sky_model()
flux_points = source.flux_points
lightcurve = source.lightcurve()
```

Fig. 10. Using gammapy.catalogs: Accessing underlying model, flux points and lightcurve from the Fermi-LAT 4FGL catalog for the blazar PKS 2155-304

3.14. gammapy.astro

TODO: Axel Donath Dark matter models, source population modelling

3.15. gammapy.catalog

TODO: Atreyee Gamma-ray catalog access

Comprehensive source catalogs are increasingly being provided by many high energy astrophysics experiments. gammapy.catalog provides a convenient access to some common gamma-ray catalogs. A global catalog table is provided, along with source models flux points and light curves (if available) for individual objects, which are internally created from the supplied FITS tables. This module works independently from the rest of the package, and the required catalogs are supplied in GAMMAPY_DATA. Presently, catalogs from Fermi-LAT (standard (Acero et al. 2015; Abdollahi et al. 2020) and high energy (Ackermann et al. 2016; Ajello et al. 2017)), H.E.S.S. galactic plane survey (H. E. S. S. Collaboration et al. 2018), HAWC (Abeysekara et al. 2017; Albert et al. 2020) and gamma-cat (Gamma-cat ????); an open source gamma ray catalog combining information from multiple catalogs) are supported.

3.16. gammapy.utils

Utility functions...

4. Applications

Gammapy can be used for a variety of science cases by different IACT experiments. (Refer to publications) In this section, we show a non-exhaustive list of some typical analysis that can be performed. In general from the Gammapy side there is not limitation on which kind of analysis can be done with which instrument, however in practice it is limited by the availability of public data.

4.1. 1D Analysis

TODO: Axel One of the most common analysis cases in gamma-ray astronomy is measuring the spectrum of a source in a given region defined on the sky, in conventional astronomy also called aperture photometry. The spectrum is typically measured in two steps: first a parametric spectral model is fitted to the data and secondly flux points are computed in a pre-defined set of energy bins. The result of such an analysis performed on three simulated CTA observations is shown in Fig. 11. In this case the spectrum was measured in a circular aperture centered on the Galactic Center, in γ-ray astronomy often called "on region". For such analysis the users first chooses a region of interest and energy binning, both defined by a ‘RegionGeom‘. In a second step the events and instrument response are binned into maps of this geometry, by the ‘SpectrumDatasetMaker‘. All the data and reduced instrument response are bundled into a ‘SpectrumDataset‘. To estimate the expected background in the on region a "reflected regions" background method was used Berge et al. (2007), represented in Gammapy by the ‘ReflectedRegionsBackgroundMaker‘ class. The resulting reflected regions are illustrated for all three observations on top of the map of counts. After reduction of the data it was modelled using a forward-folding method and a power-law spectral shape, using the ‘PowerLawSpectralModel‘ and ‘Fit‘ class. Based on this best fit model the final flux points and corresponding log-likleihood profiles are computed using the ‘FluxPointsEstimator‘.

4.2. 3D Analysis

TODO: Laura

Simulation of overlapping sources, 3D fitting, npred map, TS maps

TODO: What to do with Figure 12 ? Ref: (Stewart 2009)

4.3. Temporal analysis

TODO: Atreyee

Lightcurve in two energy bands (600 GeV - 1.5 TeV, and 1.5 TeV to 20 TeV) for PKS2155-304 flare seen by H.E.S.S. as in our notebooks.

4.4. Multi instrument analysis

TODO: Cosimo Nigro

4.5. Broadband SED Modeling

Figure 14. Add explanation text.

4.6. Population studies

TODO: Quentin Remy

5. Gammapy project

Open development, roadmap, communities, science tool aspect Infrastructure etc.

community driven vs. institutional driven

Validation and benchmarks? Validation as online appendix...

5.1. Development, testing

TODO: Jose Enrique writes this... -Github, pytest, CI, PIGs?

Language	files	blank	comment	code
Python API	236	11138	18160	29710
Python Tests	123	5779	1014	20385
DocStrings	236	0	0	18160
reStructuredText	105	4097	2981	11132
Notebooks	33	0	20628	3664
YAML	13	25	29	700
SVG	4	4	4	298
make	2	44	18	229
CSS	1	55	8	228
Batch	1	21	1	148
Cython	1	16	35	68
Markdown	3	17	0	53
TOML	1	0	0	9
Total	400	15417	41864	46239

Table 1. Coding languages statistics in Gammapy project

5.2. Documentation

- Notebooks

5.3. Software distribution and user support

- Pip, conda, versions, gammapy download

5.4. Community

TODO: Figure: Screenshot of Jupyter notebook or docs with notebook, could show the interactive maps view

m = Map.read(“diffuse.fits”)
m.plot_interactive()

5.5. PIGs

6. Summary and Outlook

TODO: Axel and Regis write this...

Summary what we have in v0.9 and presented in this paper.

Roadmap to v1.0, about half a page.

Short conclusion: Gammapy has potential to be the Python package for gamma-ray astronomy.

Prospects for HAWC / SWGO? Or speak in general about water Cherenkov observatories...

Acknowledgements. Mention Christoph here? We would like to thank the Numpy, Scipy, IPython and Matplotlib communities for providing their packages which are invaluable to the development of Astropy. We thank the GitHub team for providing us with an excellent free development platform. We also are grateful to Read the Docs (https://readthedocs.org/), and Travis (https://www.travis-ci.org/) for providing free documentation hosting and testing respectively. Finally, we would like to thank all the Gammapy users that have provided feedback and submitted bug reports. TODO: copy over stuff from http://docs.gammapy.org/en/latest/about.html#thanks. TODO: add the ANR for Luca (and Atreyee in LUPM?)

References

Abdollahi, S., Acero, F., Ackermann, M., et al. 2020, The Astrophysical Journal Supplement Series, 247, 33
Abeysekara, A. U., Albert, A., Alfaro, R., et al. 2017, ApJ, 843, 40
Acero, F., Ackermann, M., Ajello, M., et al. 2015, ApJS, 218, 23
Ackermann, M., Ajello, M., Atwood, W. B., et al. 2016, ApJS, 222, 5
Ajello, M., Atwood, W. B., Baldini, L., et al. 2017, ApJS, 232, 18

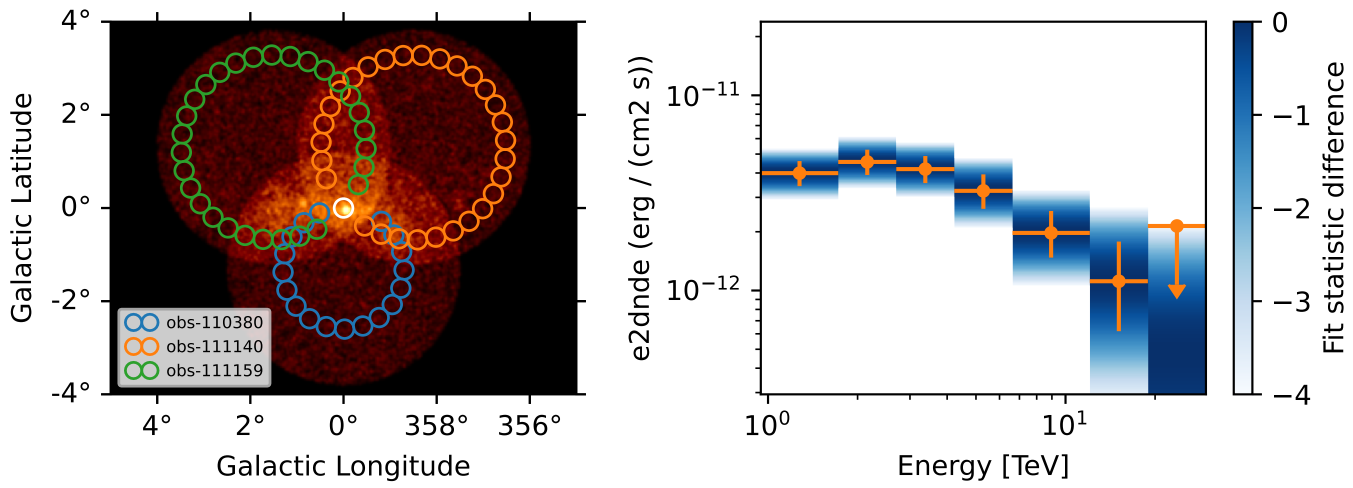

Fig. 11. Example spectral analysis of the Galactic Center for three simulated CTA observations. The left image shows the maps of counts with the measurement region and background regions overlaid in different colors. The right image shows the resulting spectral points and their corresponding log-likelihood profiles.

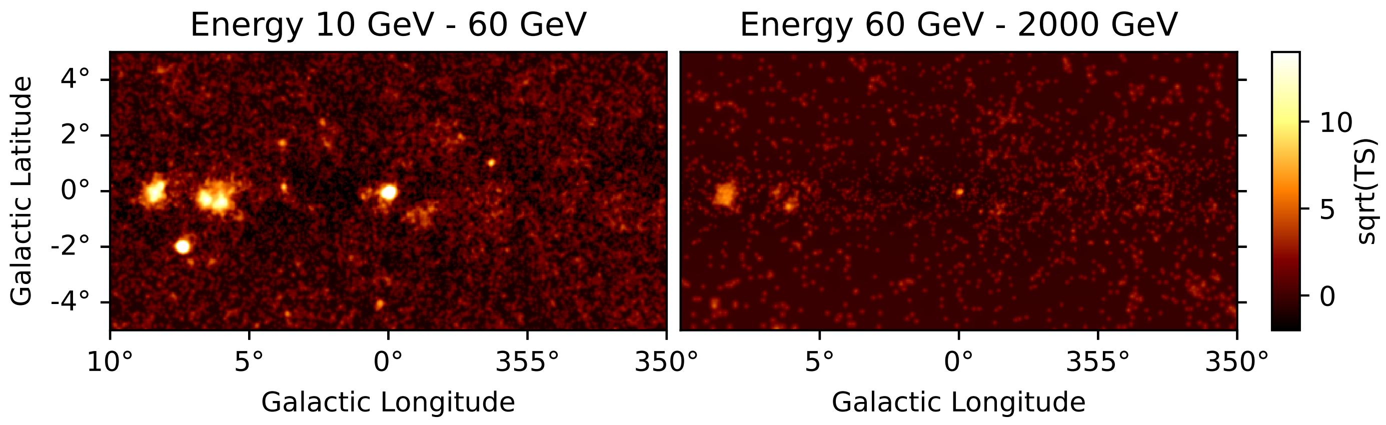

Fig. 12. Fermi-LAT TS map in two energy bands

Albert, A., Alfaro, R., Alvarez, C., et al. 2020, ApJ, 905, 76
Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33
Berge, D., Funk, S., & Hinton, J. 2007, A&A, 466, 1219
Calabretta, M. R. & Greisen, E. W. 2002, A&A, 395, 1077
Deil, C., Boisson, C., Kosack, K., et al. 2017, in American Institute of Physics Conference Series, Vol. 1792, 6th International Symposium on High Energy Gamma-Ray Astronomy, 070006
Dembinski, H. & et al., P. O. 2020
Gamma-cat. ????, Gammapy/gamma-cat: An open data collection and source catalog for Gamma-Ray Astronomy
Górski, K. M., Hivon, E., Banday, A. J., et al. 2005, ApJ, 622, 759
H. E. S. S. Collaboration, Abdalla, H., Abramowski, A., et al. 2018, A&A, 612, A1
Nigro, C., Hassan, T., & Olivera-Nieto, L. 2021a, Universe, 7, 374
Nigro, C., Sitarek, J., Gliwny, P., et al. 2021b, arXiv e-prints, arXiv:2112.14573
Owen et al., E. 2015, PoS(SciNeGHE2014), 34 [arXiv:1506.02319]
Pence, W. D., Chiappetti, L., Page, C. G., Shaw, R. A., & Stobie, E. 2010, AAP, 524, A42+
Raue, M. & Deil, C. 2012, in American Institute of Physics Conference Series, Vol. 1505, American Institute of Physics Conference Series, ed. F. A. Aharonian, W. Hofmann, & F. M. Rieger, 789–792
Refsdal, B., Doe, S., Nguyen, D., & Siemiginowska, A. 2011, in 10th SciPy Conference, 4 – 10
Refsdal, B. L., Doe, S. M., Nguyen, D. T., et al. 2009, in 8th SciPy Conference, 51 – 57
Stewart, I. M. 2009, A&A, 495, 989
Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, Nature Methods, 17, 261
Zabalza, V. 2015, ArXiv e-prints [arXiv:1509.03319]

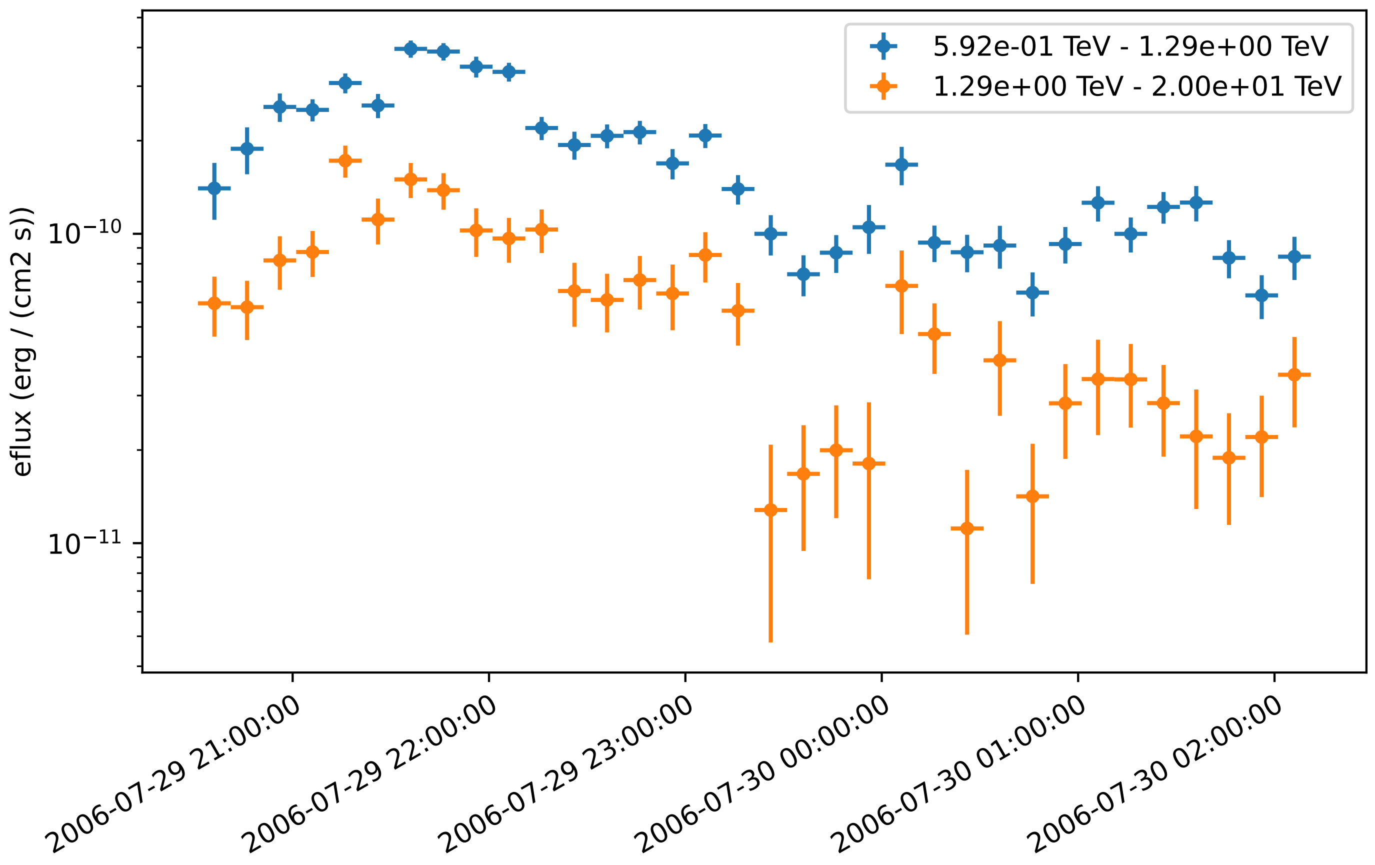

Fig. 13. H.E.S.S. PKS 2155-304 flare in two energy bands

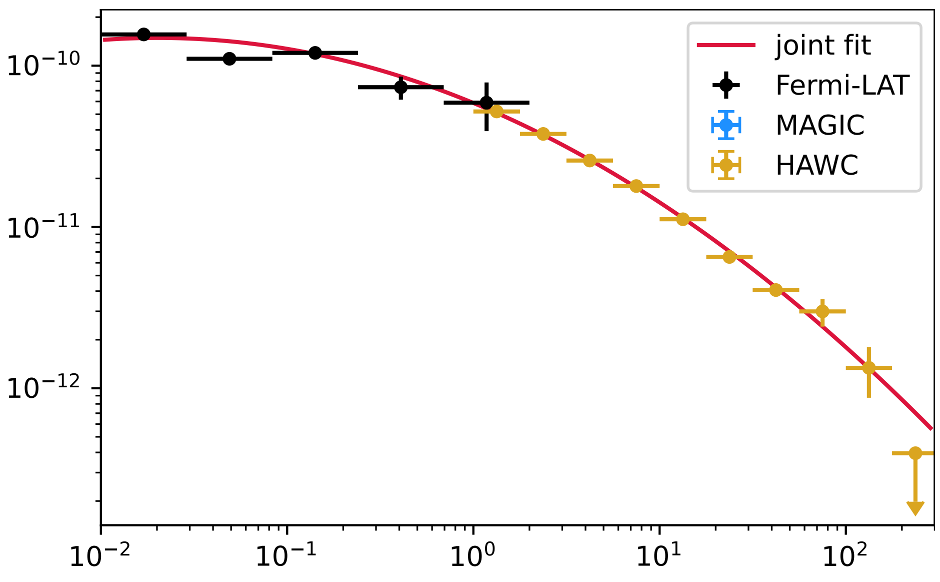

Fig. 14. A multi-instrument analysis of the Crab Nebula

[1] Amazon, Atlanta GA, US
[2] Aperio Software, Insight House, Riverside Business Park, Stoney Common Road, Stansted, Essex, CM24 8PL, UK
[3] Astroparticle and Cosmology Laboratory CNRS, Université de Paris, 10 Rue Alice Domon et Léonie Duquet, 75013 Paris, France
[4] CNRS
[5] CTA Observatory gGmbH, Via Piero Gobetti 93/3 40129 Bologna, Italy
[6] California Institute of Technology, Pasadena CA, US
[7] Center for Astrophysics | Harvard & Smithsonian, CfA, 60 Garden St., 02138 Cambridge MA, US
[8] DAp/AIM, CEA-Saclay/CNRS, France
[9] DESY
[10] Department of Physics, University of the Free State, PO Box339, Bloemfontein 9300, South Africa
[11] Deutsches Elektronen-Synchrotron, DESY, Platanenallee 6 15738 Zeuthen, Germany
[12] Facebook, 1 Hacker Way, Menlo Park, CA 94025, US

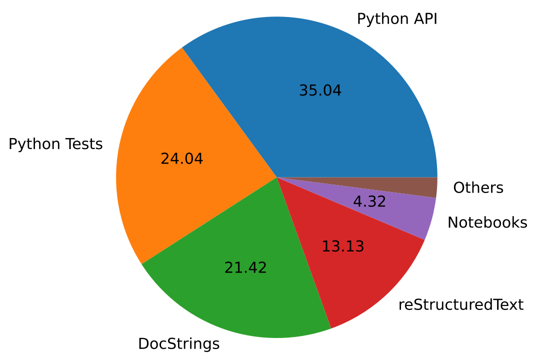

Fig. 15. Percentage of lines of code in Gammapy project