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)

June 9, 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), abbrevi-

ated 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

The data processing workflow in Gamma-ray astronomy is usually split in two sequences. The first one deals with the data processing from camera measurement, calibration, event reconstruction and selection to yield a list of reconstructed gamma-ray event candidates. This sequence, sometimes referred to as low-level analysis, is usually very specific to a given observation technique and even to a given instrument.

The other sequence, referred to as high-level analysis, deals with the extraction of physical quantities related to gamma-ray sources and the production of high-level products such as spectra, lightcurves and catalogs. The methods and tools applied here are more generic and are broadly
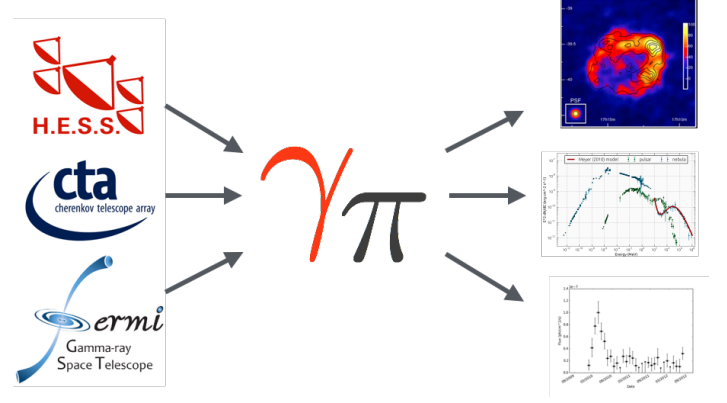
Fig. 1. Gammapy is a Python package for high-level gamma-ray data analysis. Using event lists, exposures and point spread functions as input you can use it to generate science results such as images, spectra, light curves or source catalogs. So far it has been used to simulate and analyse H.E.S.S., CTA and Fermi-LAT data, hopefully it will also be applied to e.g., VERITAS, MAGIC or HAWC data in the future.
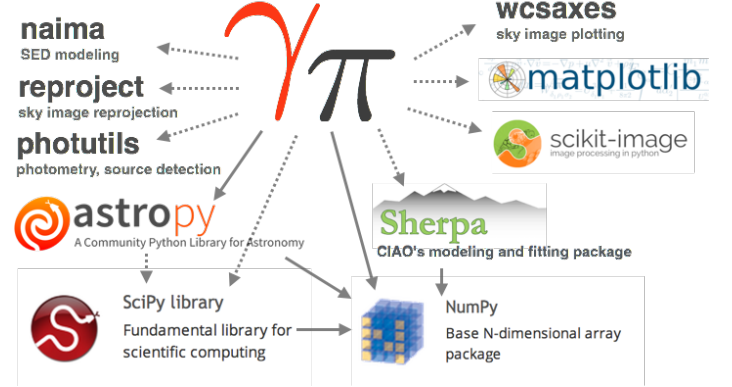

Fig. 2. The Gammapy stack. Required dependencies Numpy and Astropy are illustrated with solid arrows, optional dependencies (the rest) with dashed arrows.

shared across the field. They also frequently imply joint analysis of multi-instrument data. To extract physically relevant information, the measured data are usually compared to a model of the expected gamma-ray emitters in the instrument field-of-view using statistical techniques such as maximum likelihood.

We can write the expected number of detected events at measured position p and energy E:

$$N(p,E)\mathrm{d}p\mathrm{d}E = t_{\mathrm{obs}} \int_{E_{\mathrm{true}}} \int_{p_{\mathrm{true}}} R(p,E|p_{\mathrm{true}},E_{\mathrm{true}}) \qquad (1)$$

$$\times \Phi(p_{\mathrm{true}},E_{\mathrm{true}})\mathrm{d}E_{\mathrm{true}}\mathrm{d}p_{\mathrm{true}} \qquad (2)$$

where

– $R(p,E|p_{\mathrm{true}},E_{\mathrm{true}})$ is the instrument response
– $\Phi(p_{\mathrm{true}},E_{\mathrm{true}})$ is the sky flux model
– t_{obs} is the observation time

A common assumption is that the instrument response can be simplified as the product of three independent functions:

$$R(p, E|p_\text{true}, E_\text{true}) = A_\text{eff}(p_\text{true}, E_\text{true}) \times \tag{3}$$
$$PSF(p|p_\text{true}, E_\text{true}) \times \tag{4}$$
$$E_\text{disp}(E|p_\text{true}, E_\text{true}) \tag{5}$$

where:

- $A_\text{eff}(p_\text{true}, E_\text{true})$ is the effective collection area of the detector. It is the product of the detector collection area times its detection efficiency at true energy E_true and position p_true.
- $PSF(p|p_\text{true}, E_\text{true})$ is the point spread function. It gives the probability of measuring a direction p when the true direction is p_true and the true energy is E_true. Gamma-ray instruments consider the probability density of the angular separation between true and reconstructed directions $\delta p = p_\text{true} - p$, i.e. $PSF(\delta p|p_\text{true}, E_\text{true})$.
- $E_\text{disp}(E|p_\text{true}, E_\text{true})$ is the energy dispersion. It gives the probability to reconstruct the photon at energy E when the true energy is E_true and the true position $:p_\text{true}$. Gamma-ray instruments consider the probability density of the migration $\mu = \frac{E}{E_\text{true}}$, i.e. $E_\text{disp}(\mu|p_\text{true}, E_\text{true})$.

Gamma-ray data at the Data Level 3 therefore consists in lists of gamma-like events and their corresponding instrument response functions (IRFs). The latter include the aforementioned effective area, point spread function (PSF), energy dispersion and residual hadronic background. The handling of DL3 data is performed by classes and methods in the gammapy.data (see 3.2) and the gammapy.irf (see 3.3) subpackages.

The first step in the analysis is the selection and extraction of observations based of their meta data including information such as pointing direction, observation time and observation conditions.

The next step of the analysis is the data reduction where all observation events and instrument responses are projected onto a user-defined geometry. A typical geometry consists in a spectral representation with a measured energy axis, and in a spatial representation, either a coordinates system with a projection (for 3-dimensional or cube analysis) or a region on the sky (for regular spectral analysis). The gammapy.maps subpackage provides general multidimensional geometry objects (Geom) and the associated data structures (Maps), see 3.4.

All observation events and instrument responses are projected onto the user defined geometry. Because residual hadronic background models can be subject to significant uncertainties, background correction must be applied, such as the ring or the field-of-view background techniques or background measurements must be performed within, e.g. reflected regions (Berge et al. 2007). Parts of the data with high associated IRF systematics must also be excluded by defining a "safe" data range. These data reduction steps are performed by classes and functions implemented in the gammapy.makers subpackage (see 3.5).

The counts data and the reduced IRFs in the form of maps are bundled into dataset objects that represent the data level 4 (DL4). They can be written to disk, in a format specific to Gammapy to allow users to read them back at any time later for modeling and fitting.

This latter step datasets classes bundle reduced data in form of maps, reduced IRFs, models and fit statistics. Different sub-classes support different analysis methods and fit statistics (e.g. Poisson statistics with known background or with OFF background measurements). The datasets are used to perform joint-likelihood fitting allowing to combine different measurements, e.g. from different observations but also from different instruments or event classes. They can also be used for binned simulation as well as event sampling to simulate DL3 events data.

The next step is then typically to model and fit the datasets, either individually, or in a joint likelihood analysis. For this purpose Gammapy provides a uniform interface to multiple fitting backends. It also provides a variety of :ref:'built in models <model-gallery>'. This includes spectral, spatial and temporal model classes to describe the gamma-ray emission in the sky. Where spectral models can be simple analytical models or more complex ones from radiation mechanisms of accelerated particle populations (e.g. inverse Compton or π^o decay). Independently or subsequently to the global modelling, the data can be re-grouped to compute flux points, light curves and flux as well as significance maps in energy bands.

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

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 specification 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 aquisitions (typically $20 - 30\,\text{min}$). 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
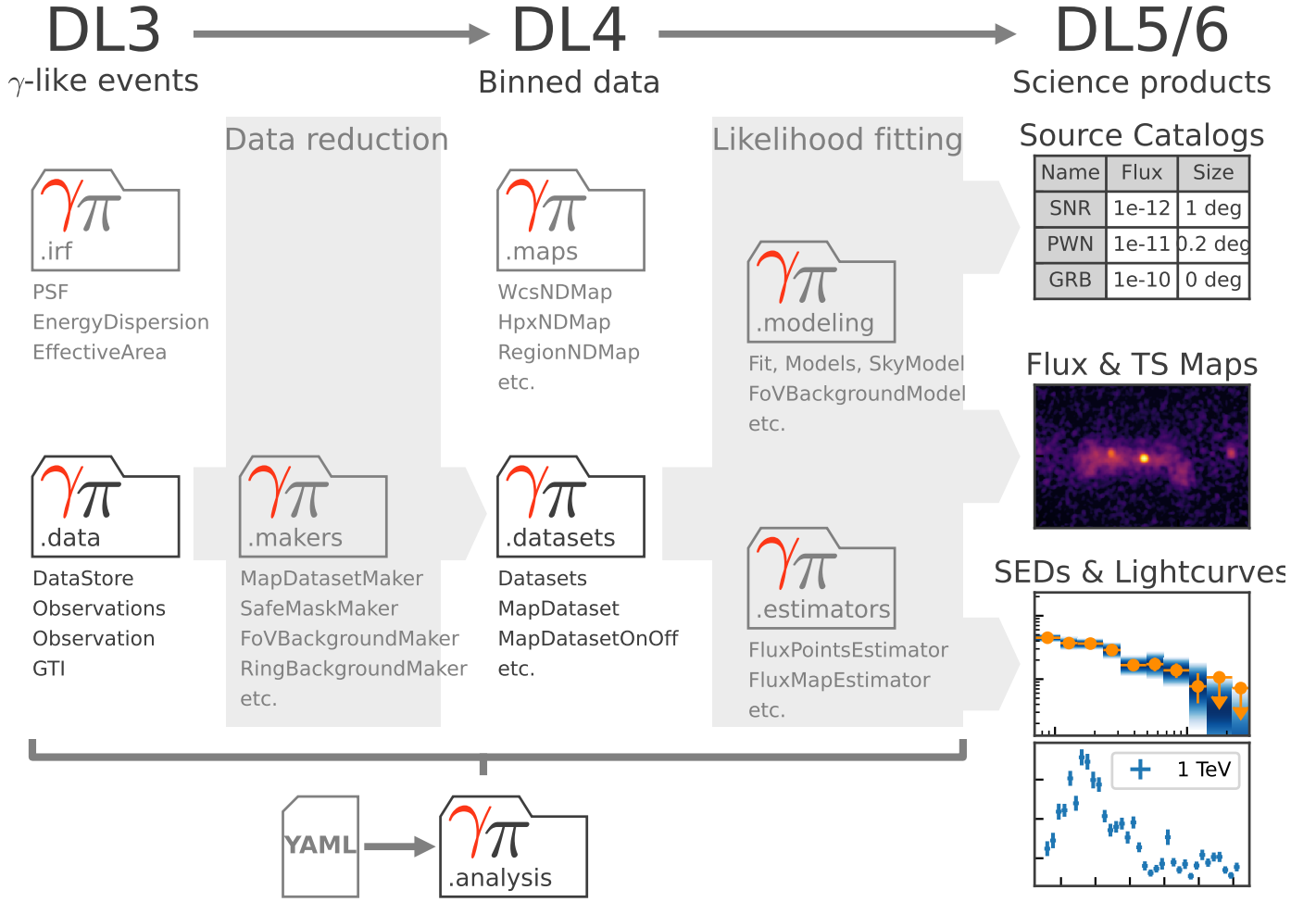
Fig. 3. Gammapy sub-package structure and data analysis workflow.

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. ??.

3.3. gammapy.irf

TODO: Fabio Pintore The gammapy.irf sub-package contains all classes and functionalities able to handle Instrument Response Functions (IRF) in a wide variety of formats. Usually, IRFs store instrument properties in the form of multi-dimension tables, with quantities expressed in

```
1  from gammapy.data import DataStore
2
3  data_store = DataStore.from_dir("$GAMMAPY_DATA")
4  obs_ids = [1, 2, 3]
5  observations = data_store.get_observations(obs_ids)
```

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

terms of energy (true or reconstructed), off-axis angles, detector coordinates and so on. The main information stored in the common γ-ray IRFs are the effective area (Aeff), energy dispersion (Edisp), point spread function (PSF) and background (BKG). The gammapy.irf sub-package can open and access specific IRF extensions, interpolate and evaluate the quantities of interest on both energy and spatial axes, convert their format or units in different kinds, plot or write them into output files. In the following, we list the main sub-classes of the sub-package:

– Effective area: Gammapy provides the class EffectiveAreaTable2D to manage Aeff, which is usually defined in terms of true energy and offset angle. The class functionalities offer the possibility to read from files or to create it from scratch. EffectiveAreaTable2D can also

Fig. 5. In Fig. 5, we show some example of Aeff, PSF, Edisp and BKG read and plotted from a typical IRF .

- convert, interpolate, write, and evaluate the effective area for a given energy and offset angles, or even plot the multi-dimensional Aeff table.
- Point spread function: Gammapy allows user to treat different kinds of PSFs, in particular, multi-dimensional gaussians (EnergyDependentMultiGaussPSF), King functions (PSFKing), or a parametric one. The EnergyDependentMultiGaussPSF class is able to handle up to three gaussians, defined in terms of amplitudes and sigma given for each true energy and offset angle bin. Similarly, PSFKing takes into account the gamma and sigma parameters as defined here. The ParametricPSF allows to create a PSF with a representation different from gaussian(s) or King profile(s). Finally, the user can take advantage from the PSFMap class, which creates a multi-dimensional map of the PSF in WCS coordinates. At each position, a PSF kernel map (PSFKernel) provides a PSF as a function of the true energy. The creation of PSF kernel maps, where the PSF is defined for each sky-position, is also given to the user. The latter two can speed up analyses.
- Energy dispersion: Edisp, in IACT , is generally given in terms of the so called migration parameter (μ), which is defined as the ratio between the reconstructed energy and the true energy. This ratio should be as close as one and its dispersion can assume the shape of a gaussian (or even more complex distributions). Migration parameter is given at each offset angle and reconstructed energy. The main sub-classes are EnergyDispersion2D, designed to interpret Edisp, EDispKernelMap, which builds an Edisp kernel map, i.e., a 4-dimensional WCS map where at each sky-position is associated an Edisp kernel. The latter is a representation of the Edisp as a function of the true energy only thanks to the sub-class EDispKernel.
- Background: the BKG can be represented in Gammapy as either 1) a 2D map (Background2D) of count rate normalised per steradians and energy at different energies and offset-angles or 2) as rate per steradians and energy, as a function of reconstructed energy and detector coordinates (Background3D). In the former, the background is expected to follow a radially symmetric shape, while in the latter, it can be more complex.

3.4. gammapy.maps

TODO: Laura Olivera-Nieto The gammapy.maps sub-package provides classes that represent 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 object 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 structure (see Figure 6).

- 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 7.

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.

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

```
1  from gammapy.maps import Map, MapAxis
2  from astropy.coordinates import SkyCoord
3  from astropy import units as u
4
5  skydir = SkyCoord("0d", "5d", frame="galactic")
6  energy_axis = MapAxis.from_bounds(lo_bnd=0.1, hi_bnd=10,
7          nbin=5, unit='TeV', interp='log', name='energy')
8
9  # Create a WCS Map
10 m_wcs = Map.create(
11     binsz=0.1,
12     map_type="wcs",
13     skydir=skydir,
14     width=[10.0, 8.0] * u.deg,
15     axes = [energy_axis])
16
17
18 # Create a HEALPix Map
19 m_hpx = Map.create(
20     binsz=0.1,
21     map_type="hpx",
22     skydir=skydir,
23     width=10.0 * u.deg,
24     axes = [energy_axis])
25
26 # Create a region map
27 region = "galactic;circle(0, 5, 1)"
28 m_region = Map.create(
29     region = region,
30     map_type='region',
31     axes = [energy_axis])
```

Fig. 6. Using gammapy.maps to create a WCS, a HEALPix and a region map. Note the uniform API for map creation.

```
1  from gammapy.makers import MapDatasetMaker, FoVBackgroundMaker, SafeMaskMaker
2
3  maker = MapDatasetMaker()
4  bkg_maker = FoVBackgroundMaker()
5  mask_maker = SafeMaskMaker()
6
7  dataset = maker.run(dataset, observation)
8  dataset = mask_maker.run(dataset,observation)
9  dataset = bkg_maker.run(dataset,observation)
```

Fig. 7. Using gammapy.makers to reduce DL3 level data into a Dataset.

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 sub-package contains classes to bundle together binned data 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 pre-

```
1  from gammapy.datasets import MapDataset, FluxPointsDataset, D
2
3  dataset1 = MapDataset.read(
4      "$GAMMAPY_DATA/cta-1dc-gc/cta-1dc-gc.fits.gz", name="m
5  )
6  dataset2 = FluxPointsDataset.read(
7      "$GAMMAPY_DATA/tests/spectrum/flux_points/diff_flux_p
8      name="fluxpoints_dataset",
9  )
10 datasets = Datasets([dataset1, dataset2])
11 print(datasets)
```

Fig. 8. A Datasets container with FluxPointsDataset and Map-Dataset

dicted 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 8), 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.

3.7.1. 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. 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).

```
1   from gammapy.modeling.models import (
2       SkyModel,
3       PowerLawSpectralModel,
4       PointSpatialModel,
5   )
6
7   pwl = PowerLawSpectralModel()
8   point = PointSpatialModel()
9
10  model = SkyModel(
11      spectral_model=pwl,
12      spatial_model=point,
13      name="my-model",
14  )
```

Fig. 9. Using gammapy.modeling.models

```
1   from gammapy.stats import WStatCountsStatistic
2
3   n_on = [13, 5, 3]
4   n_off = [11, 9, 20]
5   alpha = [0.8, 0.5, 0.1]
6   stat = WStatCountsStatistic(n_on, n_off, alpha)
7   # Excess
8   stat.n_sig
9   # Significance
10  stat.sqrt_ts
11  # Asymmetrical rrors
12  stat.compute_errn(1.)
13  stat.compute_errp(1.)
```

Fig. 10.

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) .

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

3.8. gammapy.stats

TODO: Regis Terrier

The gammapy.stats subpackage contains the fit statistics and associated statistical estimators that are commonly used in gamma-ray astronomy. In general, gamma-ray observations count Poisson-distributed events at various sky positions, and contain both signal and background events. Estimation of the number of signal events is done through likelihood maximization. In Gammapy, the fit statistics are Poisson log-likelihood functions normalized like chi-squares, i.e., they follow the expression $2 \times logL$, where L is the likelihood function used. The statistic function used when the expected number of background events is known is the Cash statistic (Cash 1979). It is used by datasets using background templates such as the MapDataset. When the number of background events is unknown and an off measurement where only background events are expected is used, the statistic function is WStat. It is a profile log-likelihood statistic where background counts are marginalized parameters. It is used by datasets containing off counts measurements such as the SpectrumDatasetOnOff, used for classical spectral analysis.

To perform simple statistical estimations on counts measurements, CountsStatistic classes encapsulate the aforementioned statistic functions to measure excess counts and estimate the associated statistical significance, errors and upper limits. They perform maximum likelihood ratio tests to estimate significance (the square root of the statistic difference) and compute likelihood profiles to measure errors and upper limits. The code example 10 shows how to compute the Li & Ma significance (Li & Ma 1983) of a set of measurements.

3.9. gammapy.estimators

TODO: Axel Donath By fitting parametric models to the data, the total integrated flux and overall temporal, spectral and morphological shape of the γ-ray emission can be constrained. In many cases it useful to make a more detailed follow-up analysis by measuring the flux in smaller spectral, temporal or spatial bins. This possibly reveals more detailed emission features, which are relevant for studying correlation with counterpart emissions.

The gammapy.estimators sub-module features methods to compute flux points, light curves, flux maps, flux profiles from data. The basic method for all these measurements is equivalent. The initial fine bins of MapDataset are grouped into larger bins. The norm of the best fit "reference" spectral model is fitted ("scaled") in the restricted data range only.

---

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

The base class of all algorithms is the 'Estimator' class.

Signifiance is estimated based on hypothesis testing against the background only model. This allows to assign a significance to the given flux bin. In addition to the best fit flux norm all estimators compute quantities corresponding to this flux. This includes the predicted number of total, signal and background counts per flux bin. The total fit statistics of the best fit model, the fit statistics of the null hypothesis and the difference between both, the so-called TS value. From the TS value the significance of the signifcance of the measured signal and associated flux can be derived.

Optionally it can also compute more advanced quantities such as assymmetric flux errors, flux upper limits and one dimensional profiles of the fit statistic, which show how the likelihood functions varies with the flux norm parameter around the fit minimum. This information is useful for inspecting the quality of the fit, where assumptoticcally a parabolic shape of the profile is expected at the best fit values (TODO: Reference to modeling / fitting section?).

The result of the flux point estimation are either stored in a FluxMaps or FluxPoints object. Both objects are based on an internal representation of the flux which is independent of the SED type. The flux is represented by a the reference spectral model and an array of normalisation values given in energy, time and spatial bins, which factorise the deviation of the flux in the given bin form the reference spectral model. This allows user to conveniently transform between different SED types. See Table 3.9. The actual flux values are obtained by mutiplication of the norm with the reference flux.

Both result objects support the possibility to serialise the data into multiple formats. This includes the GADF SED (reference?) format, FITS based ND skymaps and formats compatible with Astropy's Table and BinnedTimeSeries data structures. This allows users to directly further analyse the results, e.g. standard algorithms for time analysis such as computing Lomb-Scargle periodograms or Bayesian blocks for time series.

So far Gammapy does not support "unfolding" of γ-ray spectra. Methods for this will be implemented in future version of Gammapy.

- Regrouping of dataset bins in time, energy etc.
- "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
-

Code example 11 shows how to use the TSMapEstimator with a given input 'MapDataset'. In addition to the model it allows to specify the energy bisn of the resulting flux and TS maps.

```
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. 11. Using the TSMapEstimator from gammapy.estimators to compute a sqrt(TS) map.

3.10. gammapy.analysis

TODO: Quentin, Axel...

The gammapy.analysis sub-module provides a high-level interface for the most common use cases identified in gamma-ray analyses. The classes and methods included can be used in Python scripts, notebooks or as commands within IPython sessions. The high-level user interface can also be used to automatise workflows driven by parameters declared in a configuration file in YAML format. This way a full analysis can be executed via a single command line taking the configuration file as input.

The Analysis class has the responsibility of orchestrating of the workflow defined in the configuration (AnalysisConfig) and triggering the execution of the AnalysisStep classes that defines the common use cases identified. These steps include the following: observations selection from the DataStore, data reduction, excess map computation, model fitting, flux-points estimation, and light-curves production. The structure of the Analysis class allows to define custom AnalysisStep, so ones could design and configure arbitrarily complex wokflows.

3.11. gammapy.visualisation

TODO: Axel Donath Plotters etc.

3.12. gammapy.astro

TODO: Atreyee

The gammapy.astro sub-package contains utility functions for studying physical scenarios in high energy astrophysics. The dark matter module computes the so called J-factors and the gamma-ray spectra expected from annihilation of dark matter in different channels according to the recipe described in Cirelli et al. (2011). In the source sub-module, dedicated classes exist for modeling galactic sources according to some come physical models, eg: SNR evolution models (Taylor 1950; Truelove & McKee 1999), evolution of PWN during the free expansion phase (Gaensler & Slane 2006) or computation of physical parameters in a pulsar assumed to be a simple dipole. There are also dedicated tools for simulating synthetic populations based on physical models derived from observational or theoretical considerations for different classes of Galac-

Type	Description	Unit Equivalency
dnde	Differential flux at a given energy	TeV^{-1} cm^2 s^{-1}
e2dnde	Differential flux at a given energy	TeV cm^2 s^{-1}
flux	Integrated flux in agiven energy range	cm^2 s^{-1}
eflux	Integrated energy flux in agiven energy range	erg cm^2 s^{-1}

Table 1. Definition of sed types.

```
1  from gammapy.catalog import SourceCatalog4FGL
2
3  catalog = SourceCatalog4FGL()
4  print("Number of sources :", len(catalog.table))
5
6  source = catalog["PKS 2155-304"]
7  model = source.sky_model()
8  flux_points = source.flux_points
9  lightcurve = source.lightcurve()
```

Fig. 12. Using gammapy.catalogs: Accessing underlying model, flux points and lightcurve from the Fermi-LAT 4FGL catalog for the blazar PKS 2155-304

tic VHE gamma-ray emitters: PWNe, SNRs Case & Bhattacharya (1998), pulsars Faucher-Giguère & Kaspi (2006); Lorimer et al. (2006); Yusifov & Küçük (2004) and gamma-ray binaries. While the present list of use cases is rather preliminary, this can be enriched with time with by users and/or developers according to future needs.

3.13. 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. 2018b), 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.

3.14. 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. 13. 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 The 1D analysis approach is a powerful tool to measure the spectrum of an isolated source. However, more complicated situations require a more careful treatment. In a field of view containing several overlapping sources, the 1D approach can't disentangle the contribution of each source to the total flux in the selected region. Sources with extended or complex morphology can result in the measured flux being underestimated, and heavily dependent on the choice of extraction region. Additionally, the 1D approach neglects the energy-dependence of the PSF.

For such situations a more complex approach is needed, the so-called 3D analysis. The three relevant dimensions are the two spatial angular coordinates and an energy axis. In this framework, a combined spatial and spectral model (that is, a SkyModel, see Section 3.7) is fitted to the sky-maps that were previously derived from the data and bundled into a MapDataset (see Sections 3.5 and 3.6).
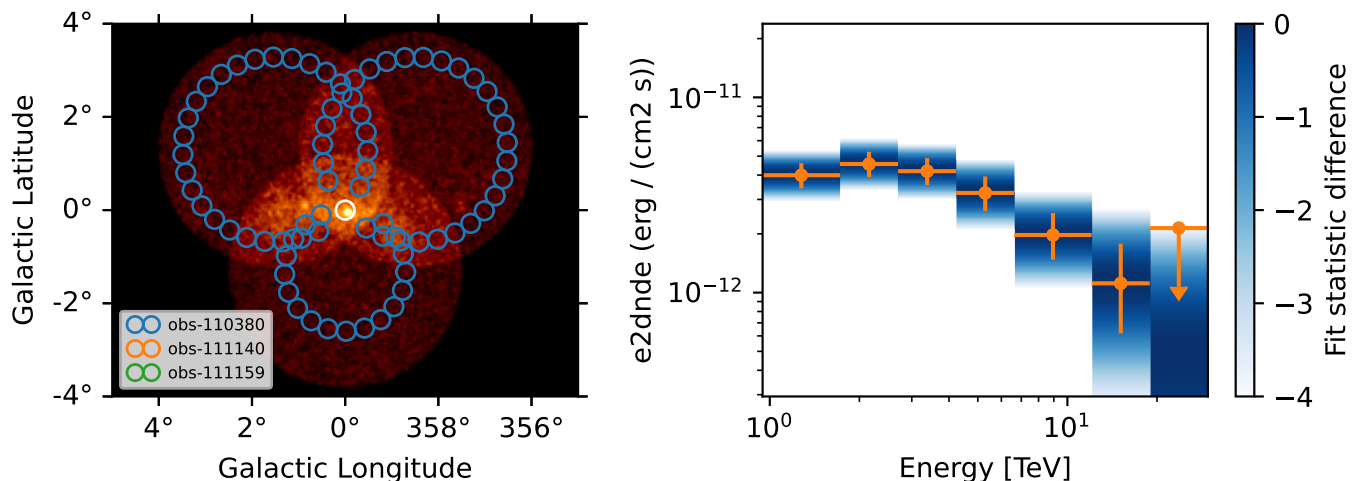
Fig. 13. Example spectral analysis of the Galactic Center for three simulated CTA observations. The left image shows the maps of counts with the measurement region and background regions overlaid in different colors. The right image shows the resulting spectral points and their corresponding log-likelihood profiles.

A thorough description of the 3D analysis approach and multiple examples that use Gammapy can be found in Mohrmann et al. (2019). Here we present a short example to highlight some of its advantages.

Starting from the IRFs corresponding to the same three simulated CTA observations used in Section 4.1, we can create a MapDataset via the MapDatasetMaker. However, we will not use the simulated event lists provided by CTA but instead, use the method MapDataset.fake() to simulate measured counts from the combination of several SkyModel instances. In this way, a DL4 dataset can directly be simulated. In particular we simulate:

1. A point source located at (l=0°, b=0°) with a power-law spectral shape.
2. An extended source with Gaussian morphology located at (l=0.4°, b=0.15°) with σ=0.2°and a log-parabola spectral shape.
3. A large shell-like structure centered around (l=0.06°, b=0.6°) with a radius and width of 0.6°and 0.3°respectively and a power-law spectral shape.

The position and sizes of the sources have been selected so that their contributions overlap. This can be clearly seen in the significance map shown in the left panel of Figure 14. This map was produced with the ExcessMapEstimator (see Section 3.9) with a correlation radius of 0.1°.

We can now fit the same model shapes to the simulated data and retrieve the best-fit parameters. To check the model agreement, we compute the residual significance map after removing the contribution from each model. This is done again via the ExcessMapEstimator. As can be seen in the middle panel of Figure 14, there are no regions above or below 5σ, meaning that the models describe the data sufficiently well.

As the example above shows, the 3D analysis allows to characterize the morphology of the emission and fit it together with the spectral properties of the source. Among the advantages that this provides is the ability to disentangle the contribution from overlapping sources to the same spatial region. To highlight this, we define a circular RegionGeom of radius 0.5° centered around the position of

the point source, which is drawn in the left panel of Figure 14. We can now compare the measured excess counts integrated in that region to the expected relative contribution from each of the three source models. The result can be seen in the left panel of Figure 14.

Note that all the models fitted also have a spectral component, from which flux points can be derived in a similar way as described in 4.1.

4.3. Temporal analysis

TODO: Atreyee

A common use case in most astrophysical scenarios is to study the temporal variability of the source. The most basic way to do this is to construct a lightcurve, ie, the flux of the source in each given time bin. In gammapy, this is done by using the LightCurveEstimator which fits the normalisation of the source in each energy band per observation, keeping other parameters constant. For custom time binning, an observation needs to be split into finer ones using the Observation.select_time method. Figure 15 shows the lightcurve of the blazar PKS 2155-304 in different energy bands as observed by the H.E.S.S. telescope during an exceptional flare on the night of July 29 - 30, 2006 Aharonian et al. (2009). The data is available publicly as a part of the HESS-DL3-DR1 H.E.S.S. Collaboration (2018), and shipped with GAMMAPY_DATA. Each observation is first split into 10 min smaller observations, and spectra extracted for each of these within a 0.11 deg radius around the source. A PowerLawSpectralModel is fit to all the datasets, leading to a reconstructed index of 3.54 ± 0.02. With this assumed spectral model the LightCurveEstimator run directly for two energy bands, 0.5 - 1.5 TeV, and 1.5 - 20 TeV, respectively.

The obtained flux points can be analytically modelled using the available, or user-implemented temporal models. Alternatively, instead of extracting a lightcurve, it is also possible to directly fit temporal models to the reduced datasets. By associating an appropriate SkyModel, consisting of both temporal and spectral components, or using custom temporal models with spectroscopic variability, to
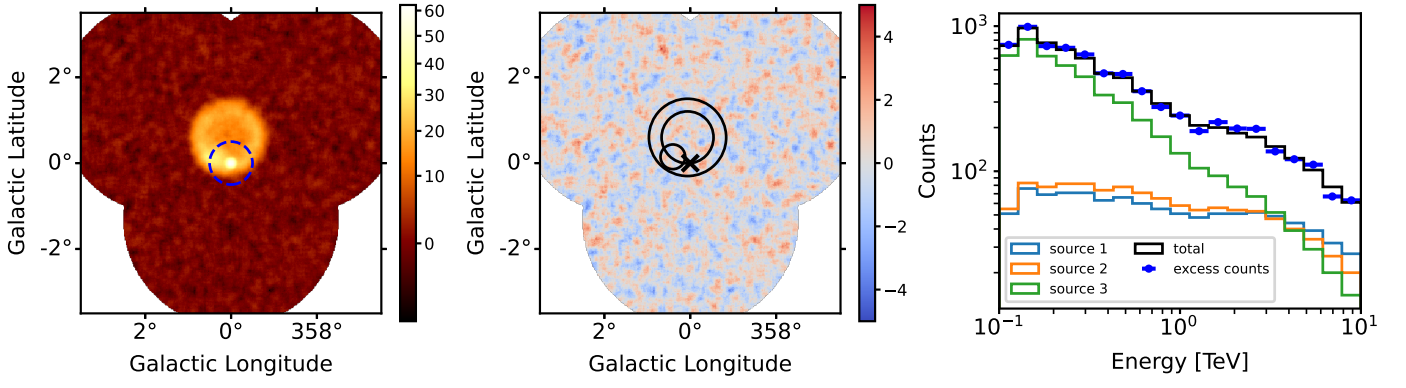
Fig. 14. Example 3D analysis for simulated sources using the CTA IRFs . The left image shows a significance map where the three simulated sources can be seen. The middle figure shows another significance map, but this time after subtracting the best-fit model for each of the sources, which are displayed in black. The right figure shows the contribution of each source model to the circular region of radius 0.5°drawn in the left image, together with the excess counts inside that region.
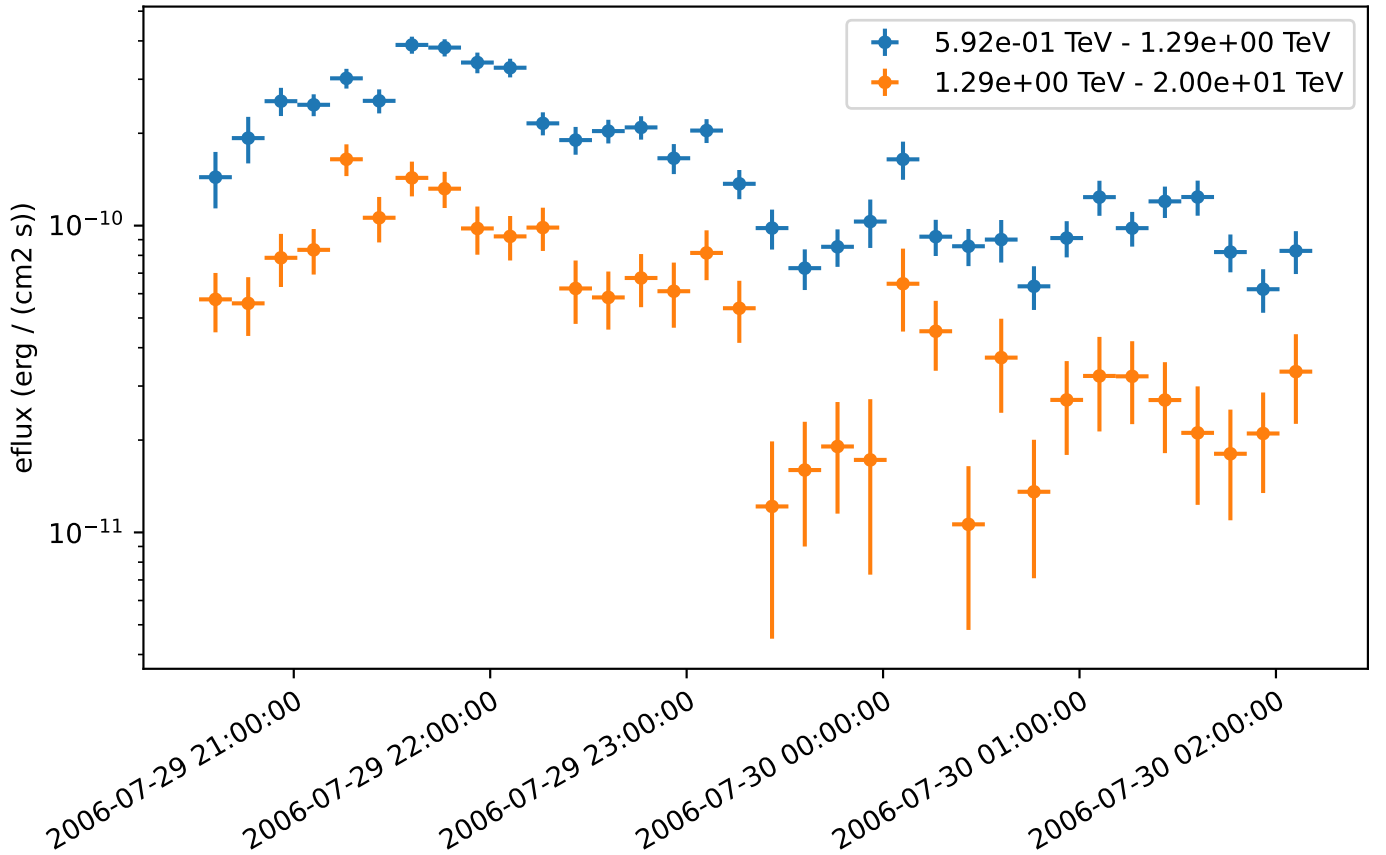

Fig. 15. 10 min binned lightcurve for PKS 2155-304 in two energy bands, (500 GeV - 1.5 TeV, and 1.5 TeV to 20 TeV) as observed by the H.E.S.S. telescopes in 2006.

each dataset, a joint fitting across the datasets will directly return the best fit temporal and spectral parameters.

4.4. Multi instrument analysis

In this multi-instrument analysis example we showcase the capabilities of Gammapy to perform a simultaneous likelihood fit incorporating data from different instruments and at different levels of reduction. We estimate the spectrum of the Crab Nebula combining data from Fermi-LAT ,

MAGIC and HAWC . Maps of Fermi-LAT data are prepared selecting a region of $X°$ around the position of the Crab Nebula applying the same selection criteria of the 3FHL catalog (7 years of data with energy from $10\,\mathrm{GeV}$ to $2\,\mathrm{TeV}$, Ajello et al. 2017). The MAGIC data are twoobservations of $20\,\mathrm{min}$ each, chosen from the dataset used to estimate the performance of the upgraded stereo system (MAGIC Collaboration 2016) and already included in Nigro et al. (2019). The observations were taken at small zenith angles ($< 30°$) in wobble mode (Fomin et al.
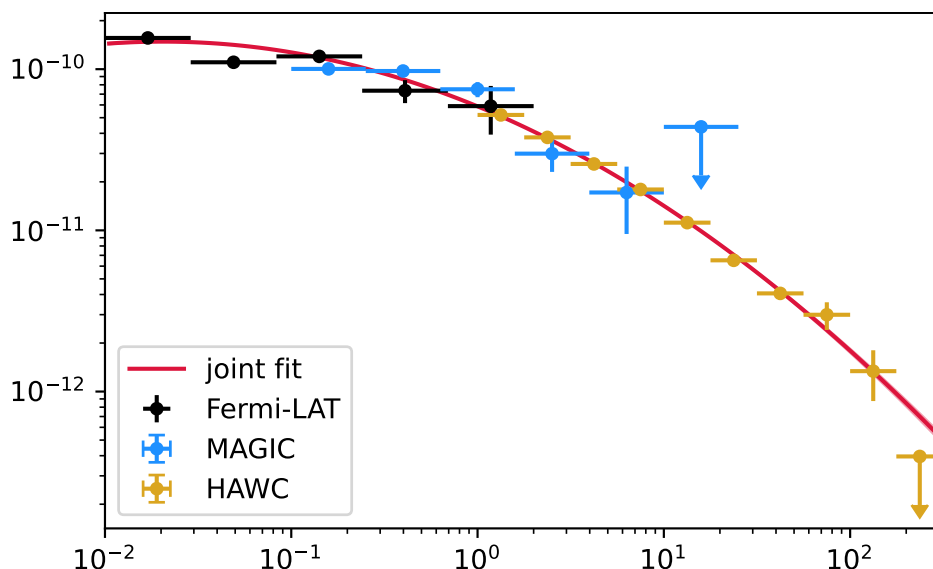
Fig. 16. A multi-instrument analysis of the Crab Nebula

1994), with the source sitting at an offset of 0.4° from the FoV center. Their energy range spans $80\,\text{GeV} - -20\,\text{TeV}$. They are reduced to ON/OFF dataset before being fitted. HAWC flux points data are estimated in HAWC Collaboration (2019) with $2.5\,\text{years}$ of data and span an energy range $300\,\text{GeV} - -300\,\text{TeV}$, and directly read with Gammapy.

Gammapy automatically generates a likelihood including three different types of terms. Two poissonian likelihoods: one for the Fermi-LAT map and one for the ON/OFF counts, and a χ^2 accounting for the flux points. For Fermi-LAT , a a three-dimensional forward folding of the sky model with the IRF is performed, in order to compute the predicted counts in each sky-coordinate and energy bin. For MAGIC , a one-dimensional forward folding of the spectral model with the IRF is performed to predict the counts in each estimated energy bin. A log parabola is fitted to the almost fived decades in energy $10\,\text{GeV} - -300\,\text{TeV}$.

The result of the joint fit is displayed in Fig. 16. We remark that the objective of this exercise is illustrative, we display the flexibility of Gammapy in simultaneously fitting multi-isntrument data even at different levels of reduction, we do not aim to provide a new measurement of the Crab Nebula spectrum.

4.5. Broadband SED Modeling

TODO: Cosimo Nigro Figure 16. Add explanation text.

4.6. Surveys, catalogs, and population studies

Sky surveys have a large potential for new source detections, and new phenomena discovery. They also offer less selection bias to perform source population studies over a large set of coherently detected and modelled objects. Early versions of gammapy were developed in parallel of the preparation of the HESS Galactic plane survey catalog (HGPS H. E. S. S. Collaboration et al. 2018c) and the associated PWN and SNR populations studies (H. E. S. S. Collaboration et al. 2018a,d).

The increase in sensitivity and resolution provided by new generation of instruments scale up the number of sources detectable and the complexity of the models needed

to represent them accurately. As an example If we compare the results of the HGPS to the expectations from the CTA Galactic Plane survey simulations we jump from 78 sources detected by HESS to about 500 detectable by CTA (Remy et al. 2021).

Studies performed on simulations not only offer a first glimpse on what could be the sky seen by CTA (according to our current knowledge on source populations), but also give us the opportunity to test the software on complex use cases[7]. So we can improve performances, optimize our analyses strategies, and identify the needs in term of parallelisation to process the large datasets provided by the surveys.

In short the production of catalogs from gamma-ray surveys can be divided in four main steps : data-reduction; object detection; model fitting and model selection; associations and classification. The IACTs data-reduction step is done in the same way than described in the previous sections but scale-up to few thousand of observations. The object detection step consists a minima in finding local maxima in the significance, or TS maps, given by the ExcessMapEstimator, or TSMapEstimator, respectively. Further refinements can include for example filtering and detection on these maps with techniques from scikit-image package (van der Walt et al. 2014), and outlier detection from scikit-learn package (Pedregosa et al. 2011). Tests on simulations shown that it reduces the spurious detections at this stage and then speed up the next step as less objects will have to be fitted simultaneously. During the modelling step each object is alternatively fitted with different models in order to determine their optimal parameters, and the best-candidate model. The subpackage gammapy.modeling.models offers a large variety of choice, and the possibility to add custom models. Several spatial models (point-source, disk, gaussian...), and spectral models (power-law, log-parabola...) may be tested for each object, so the complexity of the problem increases rapidly in regions crowded with multiple extended sources. Finally an object is discarded if its best-fit model is not signifi-

---

[7] Note that the CTA-GPS simulations were performed with the ctools package (Knödlseder et al. 2016) and analysed with both ctools and gammapy packages in order to cross-validate them.

cantly preferred over the null hypothesis (no source) comparing the difference in log-Likelihood between these two hypotheses. For the association and classification step, that is tightly connected to the population studies, we can a minima compare the fitted models to the set of gammapy-ray catalogs available in gammapy.catalog. However further multi-wavelength cross-matches are usually required to characterize the sources.

5. Gammapy project

In this section, we provide an overview of the organization of the Gammapy project. We briefly describe the main roles and responsibilities within the team, as well as the technical infrastructure designed to facilitate the development and maintenance of Gammapy as a high quality software. We use common tools and services for software development of Python open-source projects, code review, testing, package distribution and user support, with a customized solution for a versioned thoroughly tested documentation in the form of user-friendly playable tutorials. This section concludes with an outlook on the roadmap for future directions.

5.1. Organizational structure

Gammapy is a well-organised international open-source project with a broad developer base and the presence of commitments from strong groups and institutes in the very high-energies astrophysics domain[8]. The main development roadmaps are discussed and validated by a Coordination Committee, composed of representatives of the main contributing laboratories. This committee is chaired by a Project Manager and his deputy while two Lead Developers manage the development strategy and organise technical activities. Because of this institutionally driven characteristic, the permanent staff and commitment of supporting institutes ensure the continuity of the executive teams. A core team of developers from the contributing laboratories is in charge of the regular development, which benefits from punctual contributions of the community at large.

5.2. Technical infrastructure

Gammapy follows an open-source and open-contribution development model based on the cloud repository service GitHub. A GitHub organization gammapy[9] hosts different repositories related with the project. The software codebase may be found in the gammapy repository (see Table 2 and Figure 17 for code lines statistics). We make extensive use of the pull request system to discuss and review code contributions.

Several automated tasks are set as GitHub actions[10], blocking the processes and alerting developers when fails occur. This is the case of the content integration workflow, which monitors the execution of the test coverage suite[11] using datasets from the gammapy-data repository. Tests scan not only the codebase but also the code snippets

---

[8] https://gammapy.org/team.html
[9] https://github.com/gammapy
[10] https://github.com/features/actions
[11] https://pytest.org

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

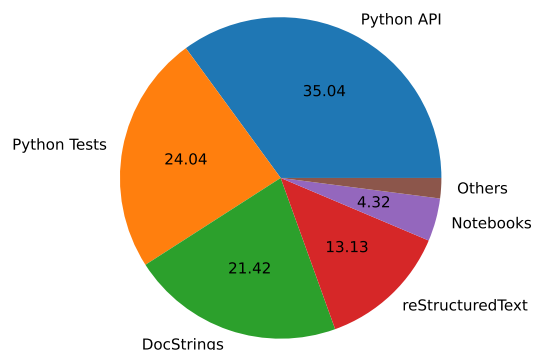Table 2. Coding languages statistics in Gammapy project

Fig. 17. Percentage of lines of code in Gammapy project

present in docstrings of the scripts and in the RST documentation files, as well as in the tutorials provided in the form of Jupyter notebooks.

Other automated tasks, executing in the gammapy-benchmarks repository, are responsible for numerical validation tests and benchmarks monitoring. Also tasks related with the release process are partially automated, and every contribution to the codebase repository triggers the documentation building and publishing workflow within the gammapy-docs repository (see Sec. 5.3 and Sec. ??).

This small ecosystem of interconnected up-to-date repositories, automated tasks and alerts, is just a part of a bigger set of GitHub repositories, where most of them are related with the project but not necessary for the development of the software (i.e., project webpage, complementary high-energy astrophysics object catalogs, coding sprints and weekly developer calls minutes, contributions to conferences, other digital assets, etc.) Finally, third-party services for code quality metrics are also set and may be found as status shields in the codebase repository.

5.3. Software distribution

Gammapy is distributed for Linux, Windows and Mac environments, and installed in the usual way for Python pack-

ages. Each stable release is uploaded to the Python package index[12] and as a binary package to the conda-forge and astropy Anaconda repository[13] channels. At this time, Gammapy is also available as a Debian Linux package[14]. We recommend installing the software using the conda installation process with an environment definition file that we provide, so to work within a virtual isolated environment with additional useful packages and ensure reproducibility.

Gammapy is indexed in Astronomy Source Code Library[15] and Zenodo[16] digital libraries for software. The Zenodo record is synchronised with the codebase GitHub repository so that every release triggers the update of the versioned record. In addition, the next release of Gammapy will be added to the Open-source scientific Software and Service Repository[17] and indexed in the European Open Science Cloud catalog[18].

5.4. Documentation and user-support

Gammapy provides its user community with a tested and versioned up-to-date on-line documentation[19] (Boisson et al. 2019) built with Sphinx[20] scanning the codebase Python scripts, as well as a set of RST files and Jupyter notebooks. The documentation includes a handwritten user guide, a set of executable tutorials, and a reference to the API automatically extracted from the code and docstrings. The Gammapy code snippets present in the documentation are tested in different environments using our continuous integration (CI) workflow based on GitHub actions.

The Jupyter notebooks tutorials are stored stripped of their output cells, which greatly helps in identifying differences in the contribution review. During the CI these are executed, the output cells parsed and, in case an error is found in one of the output cells then validation fails. Code formatting of the input cells is also done, parsing their content and using a particular syntax to provide links to the API documentation pages. The resulting web published tutorials also provide links to playground spaces in myBinder (Project Jupyter et al. 2018), where they may be executed on-line in versioned virtual environments hosted in the myBinder infrastructure. Users may also play with the tutorials locally in their laptops. They can download a specific version of the tutorials together with the associated datasets needed and the specific conda computing environment, using the gammapy download command.

We have also set up a solution for users to share recipes as Jupyter notebooks that do not fit in the Gammapy core documentation but which may be relevant as specific use cases. Contributions happen via pull requests to the gammapy-recipes GitHub repository and merged after a short review. All notebooks in the repository are tested and published in the Gammapy recipes webpage[21] automatically using GitHub actions.

---

[12] https://pypi.org
[13] https://anaconda.org/anaconda/repo
[14] https://packages.debian.org/sid/python3-gammapy
[15] https://ascl.net/1711.014
[16] https://zenodo.org/record/5721467o
[17] https://projectescape.eu/ossr
[18] https://eosc-portal.eu
[19] https://docs.gammapy.org
[20] https://www.sphinx-doc.org
[21] https://gammapy.github.io/gammapy-recipes

A growing community of users is gathering around the Slack messaging[22] and GitHub discussions[23] support forums, providing valuable feedback on the Gammapy functionalities, interface and documentation. Other communication channels have been set like mailing lists, a Twitter account[24], regular public coding sprint meetings, hands-on session within collaborations, weekly development meetings, etc.

5.5. Roadmap

TODO: Axel, Regis

– pigs
– roadmap

6. Reproducibility

One of the most important goals of the Gammapy project is to support open and reproducible science results. Thus we decided to write this manuscript openly and publish the Latex source code along with the associated Python scripts to create the figures in https://github.com/gammapy/gammapy-v1.0-paper. This GitHub repository also documents the history of the creation and evolution of the manuscript with time. To simplify the reproducibility of this manuscript including figures and text, we relied on the tool "showyourwork" (Luger 2021). This tool coordinates the building process and software as well as data dependencies such, that the complete manuscript can be reproduced with a single "make" command, after downloading the source repository. See TODO: reference? for detailed instructions. Almost all figures in this manuscript provide a link to a Python script, that was used to produce it. This means all example analyses presented in Sec.4 link to actually working Python source code.

7. Summary and Outlook

TODO: Axel and Regis write this...

In this manuscript we presented the first long term support version of Gammapy. Gammapy is a Python package for γ-ray astronomy, which relies on the scientific Python ecosystem, including Numpy and Scipy and Astropy as main dependencies. It also holds the status of an Astropy affiliated package. It supports high-level analysis of astronomical γ-ray data from intermediate level data formats, such as the FITS based GADF . Starting from lists of γ-ray events and corresponding description of the instrument response users can reduce and project the data to WCS, HEALPix and region based data structures. The reduced data is bundled into datasets, which serve as a basis for Poisson maximum likelihood modelling of the data. For this purpose Gammapy provides a wide selection of built-in, spectral, spatial and temporal models as well as unified fitting interface with connection to multiple optimization backends.

With the v1.0 milestone the Gammapy project enters a new development phase. Future work will not only include maintenance of the v1.0 release, but also parallel development of new features, improved API and data model

---

[22] https://gammapy.slack.com
[23] https://github.com/gammapy/gammapy/discussions
[24] https://twitter.com/gammapyST

support. While v1.0 provides all the features required for standard and advanced astronomical γ-ray data analysis, we already identified specific improvements to be considered in the roadmap for a future v2.0 release. This includes the support for scalable analyses via distributed computing. This will allow users to scale an analysis from a few observations to multiple hundreds of observations as expected by deep surveys of the CTA observatory. In addition the high level interface of Gammapy is planned to be developed into a fully configurable API design. This will allow users to define arbitrary complex analysis scenarios as YAML files and even extend their workflows by user defined analysis steps via a registry system. Another important topic will be to improve the support of handling meta data for data structures and provenance information to track the history of the data reduction process from the DL3 to the highest DL5/DL6 data levels.

Around the core Python package a large diverse community of users and contributors has developed. With regular developer meetings, coding sprints and in-person user tutorials at relevant conferences and collaboration meetings, the community has constantly grown. So far Gammapy has seen 80 contributors from 10 different countries. With typically 10 regular contributors at any given time of the project, the code base has constantly grown its range of features and improved its code quality. With Gammapy being selected as the base library for the future science tools for CTA, we expect the community to grow even further, providing a stable perspective for further usage, development and maintenance of the project.

Besides the future use by the CTA community Gammapy has already been used for analysis of data from the H.E.S.S. and MAGIC instruments With the availability of more γ-ray data from existing Cherenkov telescopes, γ-ray

While Gammapy was mainly developed for the science community around IACT instruments, the data model and software design is general enough to be applied to other γ-ray instruments as well. The use of Gammapy for the analysis of data from the High Altitude Water Cherenkov Observatory (HAWC) has been successfully demonstrated by TODO: Olivera et al.. This makes Gammapy a viable choice for the base library for the science tools of the future Southern Widefield Gamma Ray Observatory (SWGO) as well. Gammapy has the potential to further unify the community of γ-ray astronomers, by sharing common tools and a common vision of open and reproducible science for the future.

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

Aharonian, F., Akhperjanian, A. G., Anton, G., et al. 2009, A&A, 502, 749

Ajello, M., Atwood, W. B., Baldini, L., et al. 2017, ApJS, 232, 18

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

Boisson, C., Ruiz, J. E., Deil, C., Donath, A., & Khelifi, B. 2019, in Astronomical Society of the Pacific Conference Series, Vol. 523, Astronomical Data Analysis Software and Systems XXVII, ed. P. J. Teuben, M. W. Pound, B. A. Thomas, & E. M. Warner, 357

Calabretta, M. R. & Greisen, E. W. 2002, A&A, 395, 1077

Case, G. L. & Bhattacharya, D. 1998, ApJ, 504, 761

Cash, W. 1979, ApJ, 228, 939

Cirelli, M., Corcella, G., Hektor, A., et al. 2011, J. Cosmology Astropart. Phys., 2011, 051

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

Faucher-Giguère, C.-A. & Kaspi, V. M. 2006, ApJ, 643, 332

Fomin, V. P., Stepanian, A. A., Lamb, R. C., et al. 1994, Astroparticle Physics, 2, 137

Gaensler, B. M. & Slane, P. O. 2006, ARA&A, 44, 17

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. 2018a, A&A, 612, A2

H. E. S. S. Collaboration, Abdalla, H., Abramowski, A., et al. 2018b, A&A, 612, A1

H. E. S. S. Collaboration, Abdalla, H., Abramowski, A., et al. 2018c, A&A, 612, A1

H. E. S. S. Collaboration, Abdalla, H., Abramowski, A., et al. 2018d, A&A, 612, A3

HAWC Collaboration. 2019, ApJ, 881, 134

H.E.S.S. Collaboration. 2018, H.E.S.S. First Public Test Data Release

Knödlseder, J., Mayer, M., Deil, C., et al. 2016, A&A, 593, A1

Li, T. P. & Ma, Y. Q. 1983, ApJ, 272, 317

Lorimer, D. R., Faulkner, A. J., Lyne, A. G., et al. 2006, MNRAS, 372, 777

Luger, R. 2021, showyourwork, https://github.com/rodluger/showyourwork

MAGIC Collaboration. 2016, Astroparticle Physics, 72, 76

Mohrmann, L., Specovius, A., Tiziani, D., et al. 2019, A&A, 632, A72

Nigro, C., Deil, C., Zanin, R., et al. 2019, A&A, 625, A10

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]

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, Journal of Machine Learning Research, 12, 2825

Pence, W. D., Chiappetti, L., Page, C. G., Shaw, R. A., & Stobie, E. 2010, AAP, 524, A42+

Project Jupyter, Matthias Bussonnier, Jessica Forde, et al. 2018, in Proceedings of the 17th Python in Science Conference, ed. Fatih Akici, David Lippa, Dillon Niederhut, & M. Pacer, 113 – 120

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

Remy, Q., Tibaldo, L., Acero, F., et al. 2021, arXiv e-prints, arXiv:2109.03729

Taylor, G. 1950, Proceedings of the Royal Society of London Series A, 201, 159

Truelove, J. K. & McKee, C. F. 1999, ApJS, 120, 299

van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., et al. 2014, PeerJ, 2, e453

Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, Nature Methods, 17, 261

Yusifov, I. & Küçük, I. 2004, A&A, 422, 545
Zabalza, V. 2015, ArXiv e-prints [arXiv:1509.03319]

[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
[13] Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen Centre for Astroparticle Physics, Erwin-Rommel-Str. 1 91058 Erlangen, Germany
[14] HeidelbergCement AG, Berliner Straße 6, 69120 Heidelberg, Germany
[15] INAF/IASF, via Ugo La Malfa 153, 90146 Palermo, Italy
[16] Institut de Fisica d'altes Energies, IFAE, Edifici Cn, Campus UAB, 08193 Bellaterra (Barcelona), Spain
[17] Institute de Recherche en Astrophysique et Planetologie, IRAP,
[18] Instituto de Astrofísica de Andalucía - CSIC, Gta. de la Astronomía, 18008 Granada, Spain
[19] Max Planck Institut für Kernphysik, MPIK, Saupfercheckweg 1, 69117 Heidelberg, Germany
[20] Michigan State University, Department of Physics and Astronomy, 428 S Shaw Ln East Lansing, MI 48824, US
[21] Mozilla, Mountain View, CA
[22] National Tsing Hua University Institute of Astronomy, Hsinchu, Taiwan
[23] Pamyra
[24] Point 8 GmbH, Rheinlanddamm 201, 44139 Dortmund, Germany
[25] Space Telescope Science Institute, STScI, 3700 San Martin Drive, 21218 Baltimore MD, US
[26] TU Dortmund University, Otto-Hahn-Str. 4a 44227 Dortmund, Germany
[27] Universidad Complutense de Madrid EMFTEL, UCM, Plaza de la Ciencias, 28040 Madrid, Spain
[28] Universidade de São Paulo, Brazil
[29] Université Savoie Mont Blanc, CNRS, Laboratoire d'Annecyde Physique des Particules - IN2P3, 74000 Annecy, France
[30] unknown