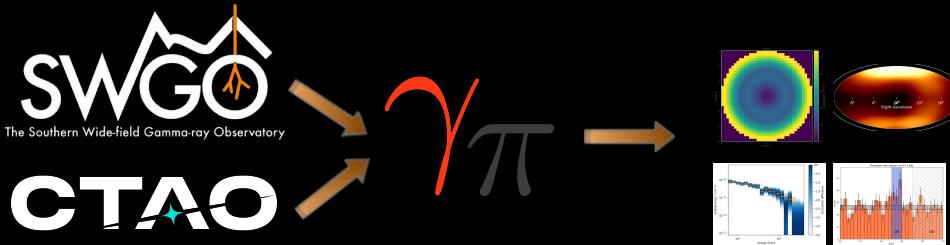


Introduction to the Gammapy library and the data analysis workflow

Kirsty Feijen

Cherenkov Astronomy Data School

14th–18th October 2024 – Observatoire de Paris



Some software history

- Gamma-ray astronomy originated from particle physics and the search for the origin of cosmic rays
- Has evolved into its own branch of astronomy, with sky images, catalogs spectral analyses etc.
- Typically root (<https://root.cern>) based analysis frameworks
- Closed experiments with proprietary data and Python has been proven to be very successful
- Gammapy is built after the idea of [Astropy](#), (thanks Astropy folks!)

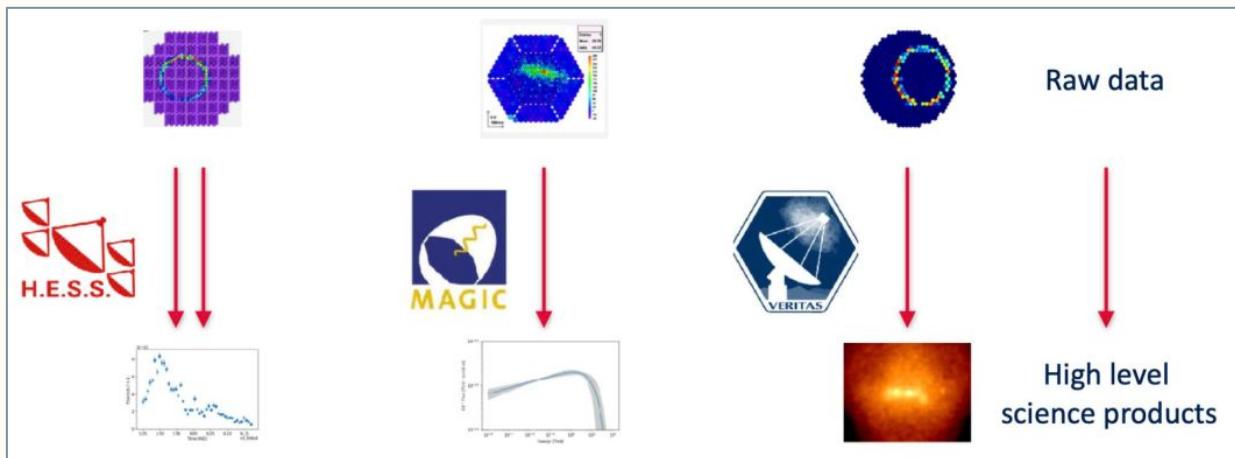
Gamma-ray instruments



- Ground based **Imaging Atmospheric Cherenkov Telescopes**, H.E.S.S., VERITAS, MAGIC, FACT, CTA*
 - Pointing instruments with good angular and energy resolution. Short duty cycle, can only operate by night. Large effective area, suited for VHE range, above a few tens of GeV
- **Water Cherenkov Observatories** HAWC, LHAASO and SWGO*
 - All sky coverage, long duty cycle, poorer angular and energy resolution. Large effective area, suited for VHE and UHE range above 1 TeV
- **Satellite based instruments**, currently only Fermi-LAT.
 - Good angular and energy resolution, but limited effective area. Thus limited to GeV energy range long duty cycle. Below 500 GeV

Gammappy concept

- Each telescope has its own data format and software
 - Multiple analysis chains within each collaboration
 - Cross checking analysis is a never-ending issue
- Combination of data from different experiments needs ‘hacking’ into proprietary analysis



All the previously mentioned observatories have complementary properties, previously one could not make advantage of this BUT with **Gammappy** we can

Gammapy concept



A high level gamma-ray astronomy
package based on common data formats



A flexible, open source, community driven
python library

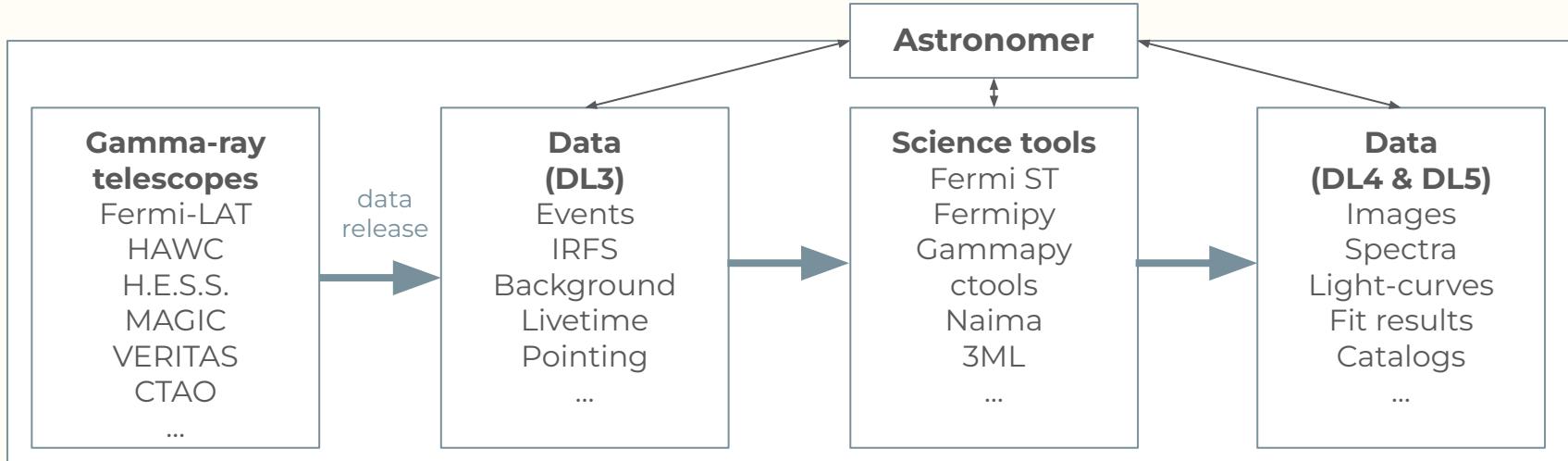


Science tools for the CTAO

Format for the data?

- In the ~10 years there has been a big effort to define a common format for storing gamma-ray data
- We learn from existing standards for x-ray astronomy and Fermi
- Result: Gamma-Astro-Data-Format (GADF)
 - Based on FITS standards, follow FITS conventions for time and coordinates
 - Information stored in binary tables in specific Header Data Unit (HDU)
- If the data looks the same, we can easily share it in a tool
- This is where gammapy comes in!

- Gamma Astro Data Format (GADF)
- Based on the Fermi-LAT format proposed in 2016
- Purpose of GADF is to encourage the collaboration between high-level gamma-ray data producers, science tool developers and data analysis
- Adopted by CTAO, H.E.S.S. DL3 DR1, MAGIC data release
- Goal: develop common data format



Gammapy overview

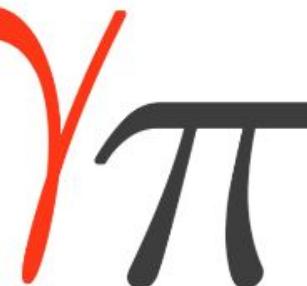


Pointing γ -ray Observatories



All-sky γ -ray Observatories

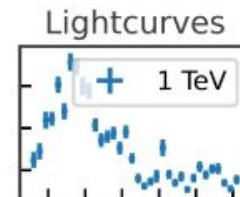
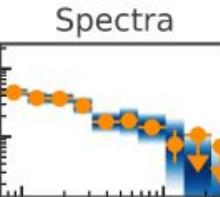
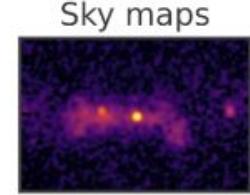
Common
data format



NumPy



SciPy
matplotlib



Dependencies



Optional dependencies: bring in useful functionality

Pydantic

Configuration

PyYAML

YAML I/O

matplotlib

Plotting, visualisation

click_

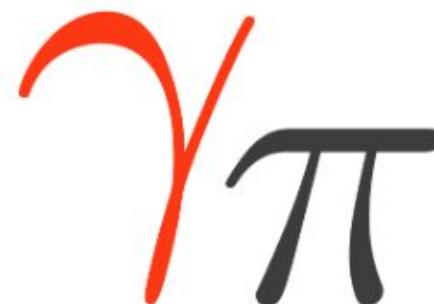
Command line tools

astropy

Coordinates, Quantities, Tables,
FITS I/O, etc.

Optional dependencies

Required dependencies



Sherpa

iminuit

Optimisation, sampling

healpy

Healpix maps

jupyter

Tutorial notebooks

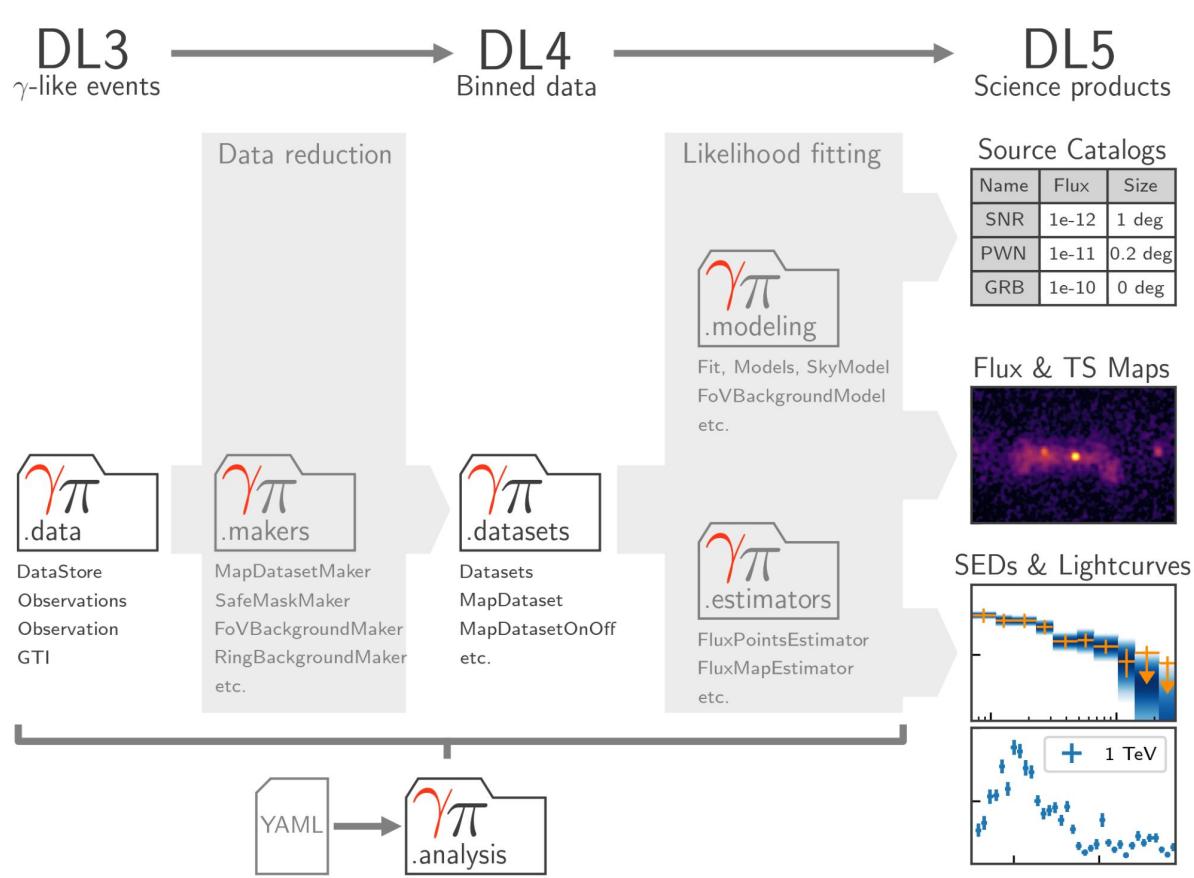
NumPy

ND-data structures
and computations

SciPy

Interpolation, minimisation,
FFT convolution, etc.

Internal workflow



2-step analysis procedure

- Data reduction (DL3 to DL4)
- Data modeling/fitting (DL4 to DL5)

Getting Started

Getting helping, reporting issues

How to provide feedback/get help

- #help channel on [gammapy.slack](#)
- #gammipy channel on [hesschat.slack](#) (if you are a H.E.S.S. member)
- [GitHub discussion](#), in particular the help category

How to report issues and bugs or request a new feature

- [GitHub issues](#) page (requires a GitHub account)

- v1.2 was released February 2024

Gammapy team involved in Science Data Challenge (SDC) effort

- Event types are supported
 - Distributed as two different data stores
 - Stacked or joint analyses are possible
- Missing features for SDC have been added:
 - Time dependent spectral models in v1.1
 - Metadata containers to support CTAO data model
- SDC will be important to prepare the first release

Getting started: webpage



Gammappy About News CTAO Contact Team Contribute Documentation Code of Conduct Acknowledging



Gammappy is an open-source Python package for gamma-ray astronomy built on [Numpy](#), [Scipy](#) and [Astropy](#). It is used as core library for the Science Analysis tools of the [Cherenkov Telescope Array Observatory \(CTAO\)](#), recommended by the [H.E.S.S.](#) collaboration to be used for Science publications, and is already widely used in the analysis of existing gamma-ray instruments, such as [MAGIC](#), [VERITAS](#) and [HAWC](#).

News:

- | | |
|------------------|---|
| Aug. 5th, 2024: | The OSCARs consortium finances Gammappy |
| Feb. 29th, 2024: | Minor version release v1.2 |
| Dec. 6th, 2023: | Bug fix release v1.0.2 |
| Oct. 23rd, 2023: | Publication of the first Gammappy paper in A&A (accepted: 7th July 2023) as |

See gammappy.org



Getting started: documentation

[docs.gammapy.org](#)

See [docs.gammapy.org](#)

The screenshot shows the Gammapi documentation homepage. At the top, there's a navigation bar with links for "Getting started", "User guide", "Tutorials", "API reference", "Developer guide", and "Release notes". A dropdown menu for "1.2" is open, showing icons for GitHub, GitLab, and a pull request. Below the navigation is a search bar with placeholder text "Search the docs ...". The main content area features the Gammapi logo and the text "A Python package for gamma-ray astronomy". It includes a section titled "Gammappy" with a date ("Feb 29, 2024") and version ("1.2"). Below this is a "Useful links" section with links to the "Web page", "Recipes", "Discussions", "Acknowledging", and "Contact". A descriptive paragraph explains that Gammappy is a community-developed, open-source Python package for gamma-ray astronomy built on Numpy, Scipy and Astropy. It is the core library for the CTA Science Tools but can also be used to analyse data from existing imaging atmospheric Cherenkov telescopes (IACTs), such as H.E.S.S., MAGIC and VERITAS. It also provides some support for Fermi-LAT and HAWC data analysis.

Switch between versions

Getting started

New to Gammappy? Check out the getting started documents. They contain information on how to install and start using Gammappy on your local desktop computer.

[To the quickstart docs](#)

User guide

The user guide provide in-depth information on the key concepts of Gammappy with useful background information and explanation, as well as tutorials in the form of Jupyter notebooks.

[To the user guide](#)

Getting started: documentation



Search bar

Switch between versions

Gammify A Python package for gamma-ray astronomy

Date: Feb 29, 2024 **Version:** 1.2

Useful links: [Web page](#) | [Recipes](#) | [Discussions](#) | [Acknowledging](#) | [Contact](#)

Gammify is a community-developed, open-source Python package for gamma-ray astronomy built on Numpy, Scipy and Astropy. It is the core library for the CTA Science Tools but can also be used to analyse data from existing imaging atmospheric Cherenkov telescopes (IACTs), such as H.E.S.S., MAGIC and VERITAS. It also provides some support for Fermi-LAT and HAWC data analysis.



Getting started

New to Gammify? Check out the getting started documents. They contain information on how to install and start using Gammify on your local desktop computer.

[To the quickstart docs](#)



User guide

The user guide provide in-depth information on the key concepts of Gammify with useful background information and explanation, as well as tutorials in the form of Jupyter notebooks.

[To the user guide](#)



Getting started



Getting
started

User
guide

Tutorials

API
reference

Developer
guide

Release
notes

1.2



Search the docs ...

Installation

Virtual Environments

Using Gammappy

Troubleshooting

Getting started

Installation

Working with conda?

Gammappy can be installed with [Anaconda](#) or Miniconda:

```
$ conda install -c conda-forge gammappy
```

Prefer pip?

Gammappy can be installed via pip from [PyPI](#).

```
$ pip install gammappy
```

In-depth instructions?

Update existing version? Working with virtual environments? Installing a specific version? Check the advanced installation page.

[Learn more](#)

On this
page

Installation

Quickstart Setup

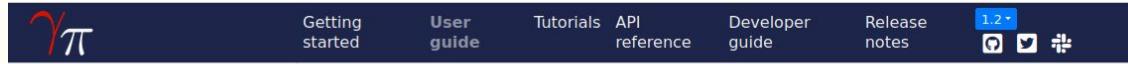
Tutorials Overview

Quickstart Setup

The best way to get started and learn Gammappy are the [Tutorials](#). For convenience we provide a pre-defined conda environment file, so you can get additional useful packages together with Gammappy in a virtual isolated environment. First install [Miniconda](#) and then just execute the following commands in the terminal:

```
$ curl -O https://gammappy.org/download/install/gammappy-1.2-environment.yml
$ conda env create -f gammappy-1.2-environment.yml
```

User guide



The navigation bar includes the $\gamma\pi$ logo, a search bar with placeholder "Search the docs ...", and links for "Getting started", "User guide", "Tutorials", "API reference", "Developer guide", "Release notes", and version "1.2". It also features social media icons for GitHub, Twitter, and a feed.

User guide



Analysis workflow and package structure

An overview of the main concepts in Gammapy package.

[To the package overview](#)



How To

A short “frequently asked question” entries for Gammappy.

[To the How To](#)



Model gallery

Gammappy provides a large choice of spatial, spectral and temporal models.

[To the model gallery](#)



Gammappy recipes

A collection of **user contributed** notebooks covering aspects not present in the official tutorials.

[To the recipes](#)

Drop down menus



- [Getting started](#)
- [User guide](#)
- [Tutorials](#)
- [API reference](#)
- [Developer guide](#)
- [Release notes](#)

1.2+

Search the docs ...

Gammappy analysis workflow and package structure

- [Data access and selection \(DL3\)](#)
- [Instrument Response Functions \(DL3\)](#)
- [Data reduction \(DL3 to DL4\)](#)
- [Sky maps \(DL4\)](#)
- [Datasets \(DL4\)](#)
- [Modeling and Fitting \(DL4 to DL5\)](#)
- [Estimators \(DL4 to DL5, and DL6\)](#)
- [High Level Analysis Interface](#)
- [Command line tools](#)
- [Source catalogs](#)
- [Astrophysics](#)
- [Statistical utility functions](#)
- [Visualization](#)
- [Utility functions](#)

How To

- [Model gallery](#)
- [Gammappy recipes](#)
- [Glossary and references](#)

User guide



Analysis workflow and package structure

An overview of the main concepts in Gammappy package.

[To the package overview](#)



How To

A short “frequently asked question” entries for Gammappy.

[To the How To](#)



Model gallery

Gammappy provides a large choice of spatial, spectral and temporal models.

[To the model gallery](#)

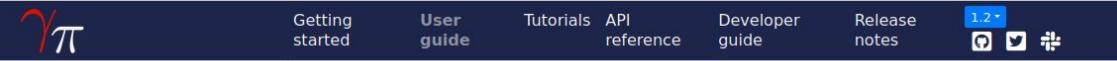


Gammappy recipes

A collection of **user contributed** notebooks covering aspects not present in the official tutorials.

[To the recipes](#)

Analysis workflow



Search the docs ...

Gammappy analysis workflow and package structure

Data access and selection (DL3)

Instrument Response Functions (DL3)

Data reduction (DL3 to DL4)

Sky maps (DL4)

Datasets (DL4)

Modeling and Fitting (DL4 to DL5)

Estimators (DL4 to DL5, and DL6)

High Level Analysis Interface

Command line tools

Source catalogs

Astrophysics

Statistical utility functions

Visualization

Utility functions

How To

Model gallery

Gammappy recipes

Glossary and references

Gammappy analysis workflow and package structure

Analysis workflow

Fig. 1 illustrates the standard analysis flow and the corresponding sub-package structure of Gammappy. Gammappy can be typically used with the configuration based high level analysis API or as a standard Python library by importing the functionality from sub-packages. The different data levels and data reduction steps and how they map to the Gammappy API are explained in more detail in the following.

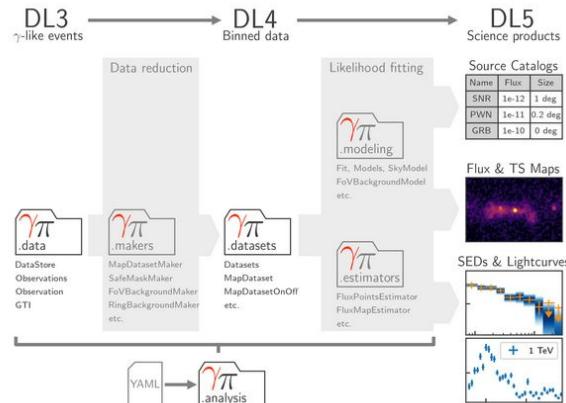


Fig. 1 Data flow and sub-package structure of Gammappy. The folder icons represent the corresponding sub-packages. The direction of the data flow is illustrated with shaded arrows. The top section shows the data levels as defined by CTA.

 On this page

Analysis workflow

Analysis steps

Configurable

analysis

Additional utilities

Getting started User guide Tutorials API reference Developer guide Release notes 1.2+ GitHub Twitter LinkedIn

Search the docs ...

Gammappy analysis workflow and package structure

- Data access and selection (DL3)**
- Instrument Response Functions (DL3)
- Data reduction (DL3 to DL4)
- Sky maps (DL4)
- Datasets (DL4)
- Modeling and Fitting (DL4 to DL5)
- Estimators (DL4 to DL5, and DL6)
- High Level Analysis Interface
- Command line tools
- Source catalogs
- Astrophysics
- Statistical utility functions
- Visualization
- Utility functions

How To

- Model gallery
- Gammappy recipes ↗
- Glossary and references

Data access and selection (DL3)

IACT data is typically structured in “observations”, which define a given time interval during which the instrument response is considered stable.

`gammappy.data` currently contains the `EventList` class, as well as classes for IACT data and observation handling.

The main classes in Gammappy to access the DL3 data library are the `DataStore` and `Observation`. They are used to store and retrieve dynamically the datasets relevant to any observation (event list in the form of an `EventList`, IRFs see [Instrument Response Functions \(DL3\)](#) and other relevant information).

Once some observation selection has been selected, the user can build a list of observations: a `Observations` object, which will be used for the data reduction process.

On this page

- Getting started with data
- The index tables
- Working with event lists
- Combining event lists and GTIs
- Writing event lists and GTIs to file
- Using `gammappy.data`

Getting started with data

You can use the `EventList` class to load IACT gamma-ray event lists:

```
from gammappy.data import EventList
filename = "${GAMMAPY_DATA}/hess-dl3-dr1/data/hess_dl3_drl_obs_id_023523.fits.gz"
events = EventList.read(filename)
```

To load Fermi-LAT event lists, use the `EventListLAT` class:

```
from gammappy.data import EventList
filename = "${GAMMAPY_DATA}/fermi-3fhl-gc/fermi-3fhl-gc-events.fits.gz"
events = EventList.read(filename)
```

The other main class in `gammappy.data` is the `DataStore`, which makes it easy to load IACT data. E.g. an alternative way to load the events for observation ID 23523 is this:

```
from gammappy.data import DataStore
data_store = DataStore.from_dir("${GAMMAPY_DATA}/hess-dl3-dr1")
events = data_store.obs(23523).events
```

Tutorials


[Getting started](#)
[User guide](#)
[Tutorials](#)
[API reference](#)
[Developer guide](#)
[Release notes](#)
[1.2](#)


Search the docs ...

- [High level interface](#)
- [Low level API](#)
- [Data structures](#)
- [CTA with Gammapp](#)
- [Fermi-LAT with Gammapp](#)
- [HAWC with Gammapp](#)
- [H.E.S.S. with Gammapp](#)
- [Point source sensitivity](#)
- [Spectral analysis of extended sources](#)
- [Flux point fitting](#)
- [Spectral analysis](#)
- [Spectral analysis with the HLI](#)
- [Spectral analysis with energy-dependent directional cuts](#)
- [1D spectrum simulation](#)
- [Source detection and significance maps](#)
- [2D map fitting](#)
- [Ring background map](#)
- [3D detailed analysis](#)
- [Multi instrument joint 3D and 1D analysis](#)
- [Basic image exploration and fitting](#)
- [Morphological energy dependence estimation](#)
- [Event sampling](#)
- [Sample a source with energy-dependent temporal evolution](#)
- [Flux Profile Estimation](#)
- [3D map simulation](#)

Tutorials

Notice : it is advised to first read [Gammapp analysis workflow and package structure](#) of the User Guide before using the tutorials.

This page lists the Gammapp tutorials that are available as [Jupyter notebooks](#). You can read them here, or execute them using a temporary cloud server in [Binder](#).

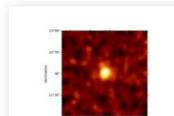
To execute them locally, you have to first install Gammapp locally (see [Installation](#)) and download the tutorial notebooks and example datasets (see [Getting started](#)). Once Gammapp is installed, remember that you can always use `gammapp info` to check your setup.

Gammapp is a Python package built on [Numpy](#) and [Astropy](#), so to use it effectively, you have to learn the basics. Many good free resources are available, e.g. [A Whirlwind tour of Python](#), the [Python data science handbook](#) and the [Astropy Hands-On Tutorial](#).

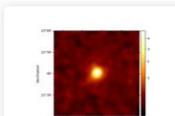
Introduction

The following three tutorials show different ways of how to use Gammapp to perform a complete data analysis, from data selection to data reduction and finally modeling and fitting.

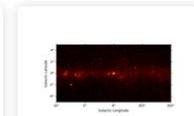
The first tutorial is an overview on how to perform a standard analysis workflow using the high level interface in a configuration-driven approach, whilst the second deals with the same use-case using the low level API and showing what is happening *under-the-hood*. The third tutorial shows a glimpse of how to handle different basic data structures like event lists, source catalogs, sky maps, spectral models and flux points tables.



High level interface



Low level API

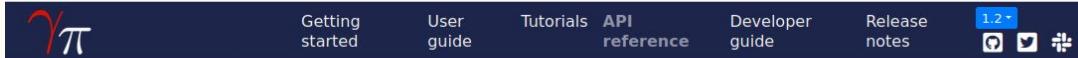


Data structures

On this page

- [Introduction](#)
- [Data exploration](#)
- [Data analysis](#)
- [Package / API](#)
- [Scripts](#)

API reference



API reference

This page gives an overview of all public Gammappy objects, functions and methods.
All classes and functions exposed in `gammappy.*` namespace are public.

- [data - DL3 data and observations](#)
 - [gammappy.data Package](#)
- [irf - Instrument response functions](#)
 - [gammappy.irf Package](#)
- [makers - Data reduction](#)
 - [gammappy.makers Package](#)
 - [gammappy.makers.utils Module](#)
- [datasets - Reduced datasets](#)
 - [gammappy.datasets Package](#)
- [maps - Sky maps](#)
 - [gammappy.maps Package](#)
- [modeling - Models and fitting](#)
 - [gammappy.modeling Package](#)
 - [gammappy.modeling.models Package](#)
 - [gammappy.modeling.models.utils Module](#)
- [estimators - High level estimators](#)
 - [gammappy.estimators Package](#)
 - [gammappy.estimators.utils Module](#)
- [analysis - High level interface](#)
 - [gammappy.analysis Package](#)
- [catalog - Source catalogs](#)
 - [gammappy.catalog Package](#)
- [astro - Astrophysics](#)
 - [gammappy.astro.darkmatter Package](#)
 - [gammappy.astro.population Package](#)
 - [gammappy.astro.source Package](#)
- [stats - Statistics](#)
 - [gammappy.stats Package](#)
- [scripts - Command line tools](#)
 - [gammappy](#)
- [visualization - Plotting features](#)
 - [gammappy.visualization Package](#)
- [utils - Utilities](#)
 - [gammappy.utils.cluster Module](#)



API reference



Q Search the docs ...

[data - DL3 data and observations](#)

[get_irfs_features](#)

[DataStore](#)

[EventList](#)

[EventListMetaData](#)

[FixedPointingInfo](#)

[GTI](#)

[HDUIndexTable](#)

[Observation](#)

[ObservationFilter](#)

[Observations](#)

[ObservationsEventsSampler](#)

[ObservationTable](#)

[PointingInfo](#)

[PointingMode](#)

[observatory_locations](#)

[irf - Instrument response functions](#)

[makers - Data reduction](#)

[datasets - Reduced datasets](#)

[maps - Sky maps](#)

[modeling - Models and fitting](#)

[estimators - High level estimators](#)

[analysis - High level interface](#)

[catalog - Source catalogs](#)

[astro - Astrophysics](#)

[stats - Statistics](#)

[scripts - Command line tools](#)

[visualization - Plotting features](#)

[utils - Utilities](#)

Getting started

User guide

Tutorials

API reference

Developer guide

Release notes

1.2



API reference

This page gives an overview of all public Gammapy objects, functions and methods.

All classes and functions exposed in `gammapy.*` namespace are public.

- [data - DL3 data and observations](#)
 - [gammapy.data Package](#)
- [irf - Instrument response functions](#)
 - [gammapy.irf Package](#)
- [makers - Data reduction](#)
 - [gammapy.makers Package](#)
 - [gammapy.makers.utils Module](#)
- [datasets - Reduced datasets](#)
 - [gammapy.datasets Package](#)
- [maps - Sky maps](#)
 - [gammapy.maps Package](#)
- [modeling - Models and fitting](#)
 - [gammapy.modeling Package](#)
 - [gammapy.modeling.models Package](#)
 - [gammapy.modeling.models.utils Module](#)
- [estimators - High level estimators](#)
 - [gammapy.estimators Package](#)
 - [gammapy.estimators.utils Module](#)
- [analysis - High level interface](#)
 - [gammapy.analysis Package](#)
- [catalog - Source catalogs](#)
 - [gammapy.catalog Package](#)
- [astro - Astrophysics](#)
 - [gammapy.astro.darkmatter Package](#)
 - [gammapy.astro.population Package](#)
 - [gammapy.astro.source Package](#)
- [stats - Statistics](#)
 - [gammapy.stats Package](#)
- [scripts - Command line tools](#)
 - [gammapy](#)
- [visualization - Plotting features](#)
 - [gammapy.visualization Package](#)
- [utils - Utilities](#)
 - [gammapy.utils.cluster Module](#)

Workflow

Data reduction

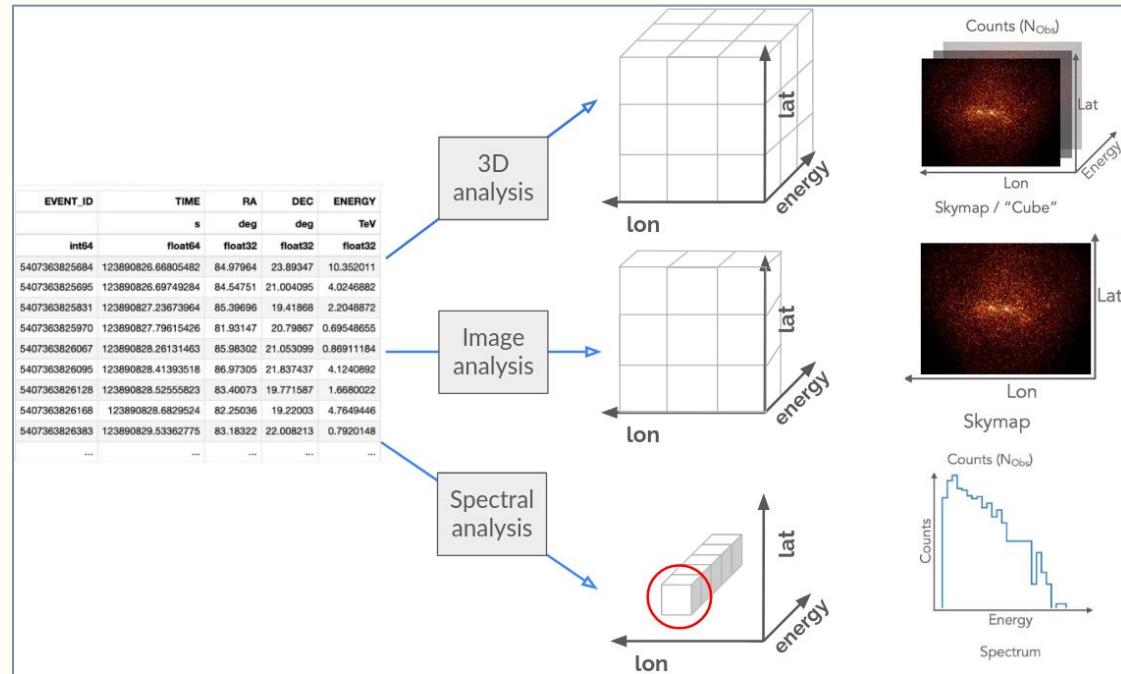
DL3→DL4

- Select and retrieve relevant observations from the data store
- Define the reduced dataset geometry
 - Is the analysis 1D (spectral only) or 3D?
 - Define target binning and projection
- Initialise the data reduction methods ([makers](#))
 - Data and IRF projection
 - Background estimation
 - Safe Mask determination
- Loop over selected observations
 - Apply makers to produce [reduced datasets](#)
 - Optionally combine them ([stacking](#))

Data reduction

DL3 → DL4

- Bin events (and IRFs) into n-dim sky maps
 - Apply event selections (time, offset, etc)
 - Spatial and energy binning
- Generalised case: 3D maps
 - Image analysis:
Cube with one energy bin
 - Spectral analysis:
Cube with one spatial bin



Data fitting

DL4→DL5

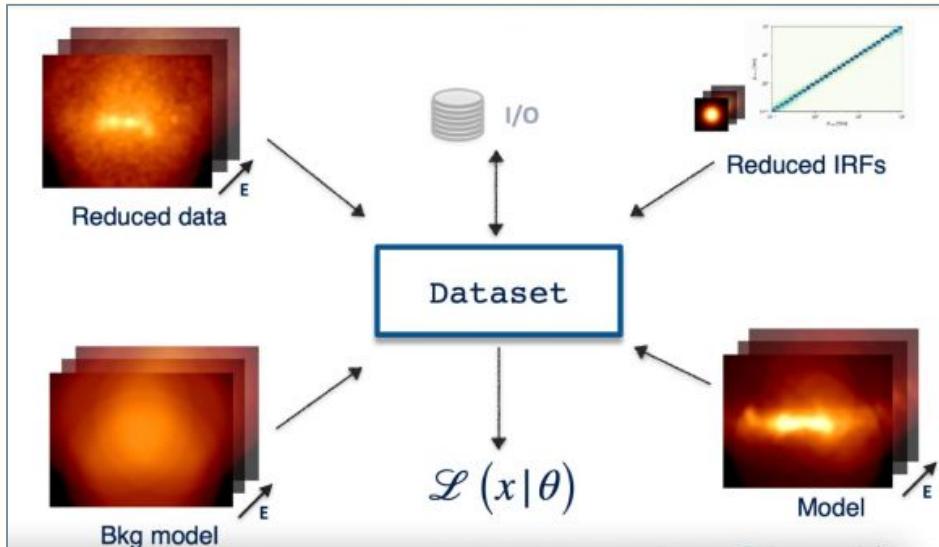


- Fitting on pre-computed datasets
 - eg: From HAWC, Fermi-LAT, OGIP files, etc
- Forward folding with maximum likelihood estimation

"Cash statistics": summed over all "bins"

$$\mathcal{C} = 2 \sum_i N_{Pred}^i - N_{Obs}^i \cdot \log N_{Pred}^i$$

$$N_{Pred} = N_{Bkg} + \sum_{Src} N_{Pred,Src}$$



End products

DL5→DL6



- DL5: Flux points, light curves and flux/TS maps
- Possible to fit DL5 data
 - Eg: published flux points, lightcurves
 - Chi-squared statistics used
- DL6: catalogs
 - Support provide for common catalogs: Fermi 4FGL, H.E.S.S. galactic plane survey, HAWC catalog, etc
 - Create your own catalogs...

API and sub-packages

gammapy.data

- Select, read and represent DL3 data in memory
- Compliant with GADF
- [Observation](#) and [EventList](#)
 - Associated IRFs
- hdu-index-table: link event list to associated IRFs
- obs-index-table: selecting observation

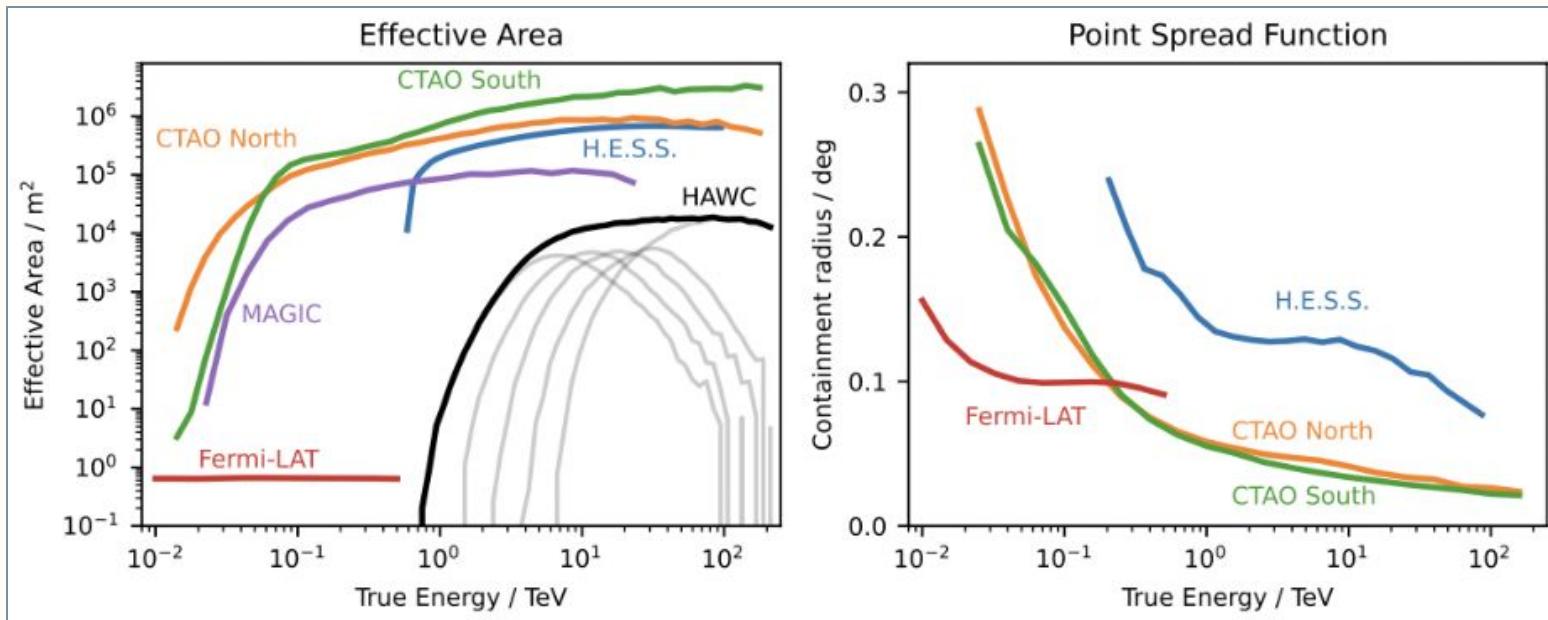
```
from gammapy.data import DataStore
data_store = DataStore.from_dir('$GAMMAPY_DATA/hess-dl3-dr1')
obs_ids = [23523, 23526, 23559, 23592]
observations = data_store.get_observations(obs_id=obs_ids)
for obs in observations:
    print(f'Observation id: {obs.obs_id}')
    print(f'N events: {len(obs.events.table)}')
    print(f'Max. area: {obs.aeff.quantity.max()}')

Observation id: 23523
N events: 7613
Max. area: 699771.0625 m2
Observation id: 23526
N events: 7581
Max. area: 623679.5 m2
Observation id: 23559
N events: 7601
Max. area: 613097.6875 m2
Observation id: 23592
N events: 7334
Max. area: 693575.75 m2
```

gammapy.irfs

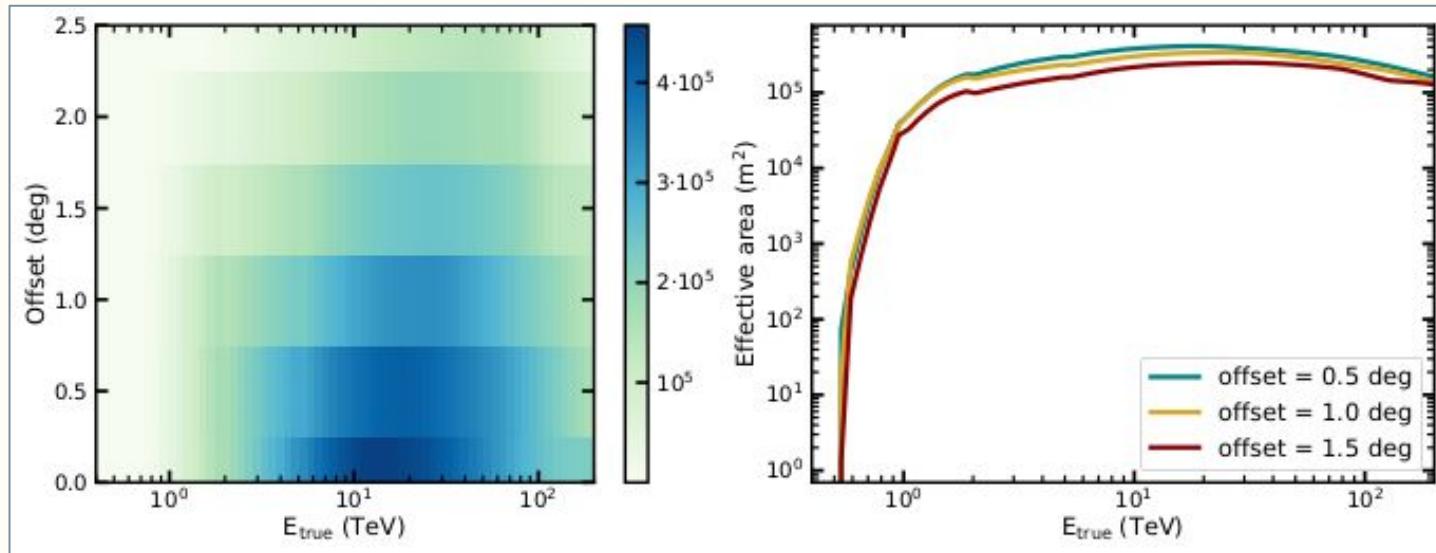


- There are a number of different Instrument Response Functions (IRFs)
 - Effective area, Point Spread Function (PSF), Energy dispersion, Background
- DL3 IRFs are reprojected onto the target geometry



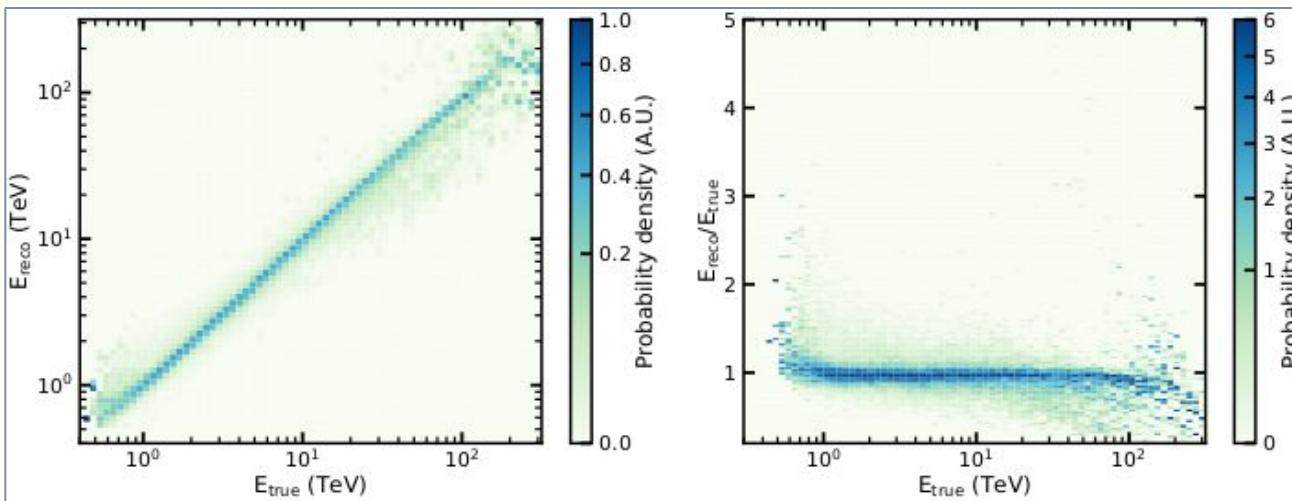
Effective area

- Effective area: gives the number of expected (true) gamma-like events per area
- Varies with offset from pointing position and true energy
- Estimated from "Monte Carlo" simulations
- Required to measure flux/brightness of gamma-ray sources
- Typically combined with observation time to derive the effective exposure

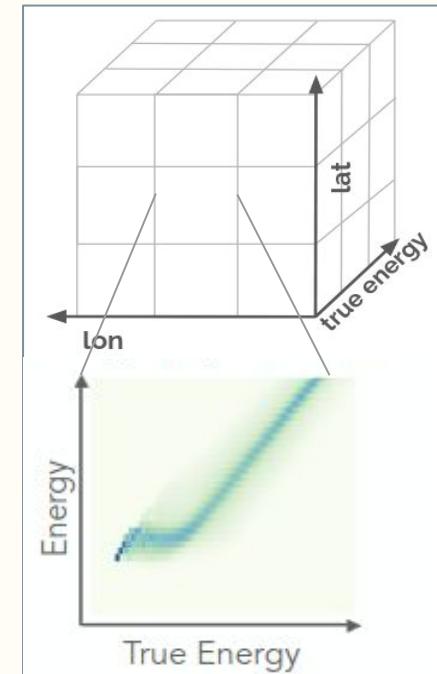


Energy dispersion

- For each true gamma-ray energy, what is the probability that the event gets assigned a certain reconstructed energy?
 - Accuracy and precision to reconstruct the energy of an event
- Varies with offset from pointing position and true energy
- Required to measure precise spectra of source, especially at low energies (for IACTs)

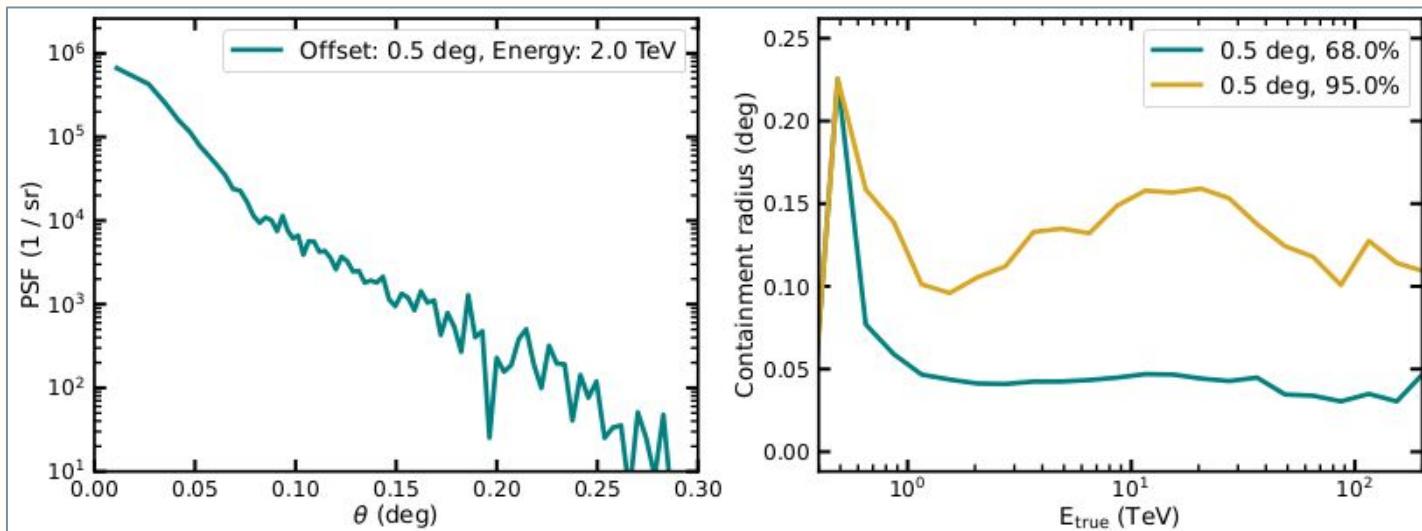


Energy Dispersion



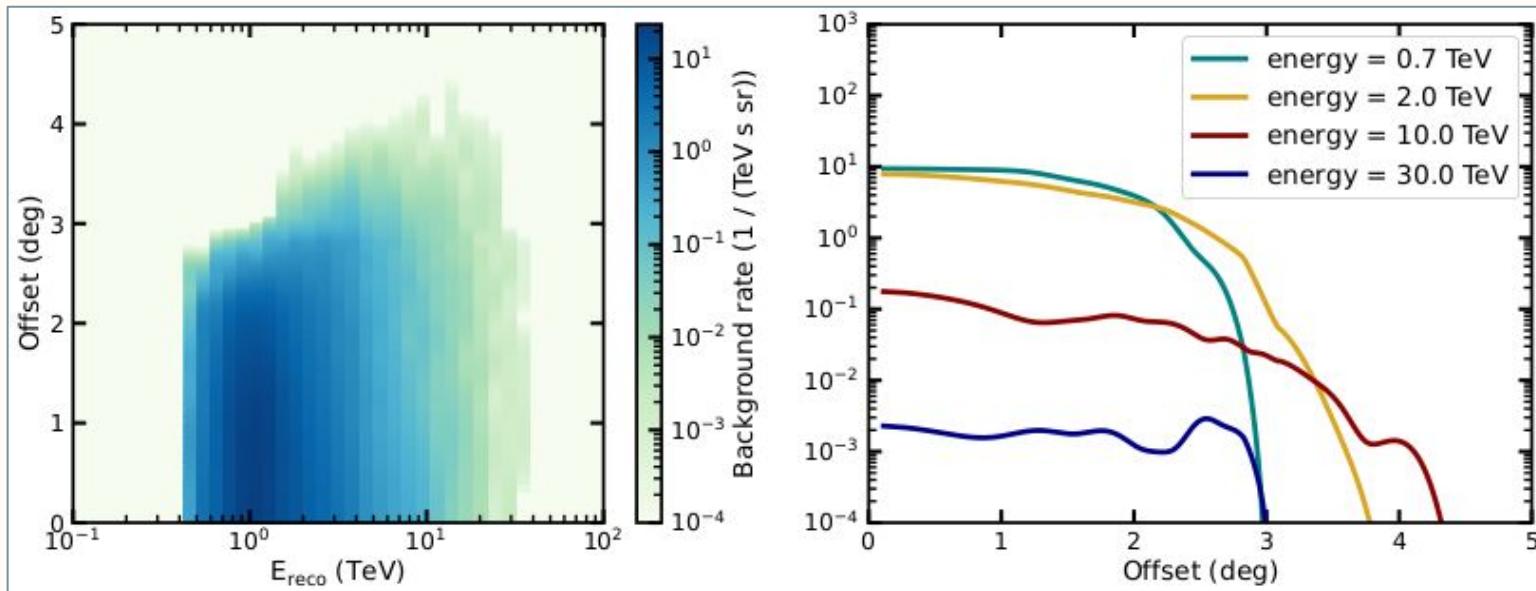
Point Spread Function

- PSF: angular resolution of the instrument, precision to reconstruct the arrival direction of an event
- Estimated from "Monte Carlo" simulations and by binning events into offset and true energy
- Typically stored as a "radial profile"
- Required to measure extension of galactic sources and precise flux of point sources



Background

- Represents the expected remaining hadronic background after gamma-hadron separation due to misclassified events
- Depends on the reconstructed energy and the offset from the pointing position



gammapy.maps

```

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_energy_bounds(
    energy_min="1 TeV", energy_max="10 TeV", nbin=10
)

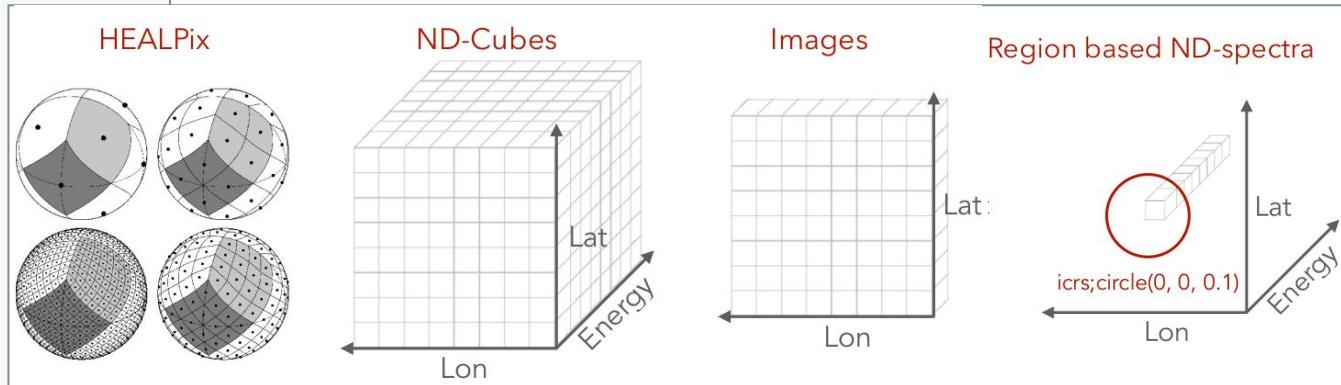
# Create a WCS Map
m_wcs = Map.create(
    binsz=0.1,
    map_type="wcs",
    skydir=skydir,
    width=[10.0, 8.0] * u.deg,
    axes=[energy_axis])

# Create a HEALPix Map
m_hpx = Map.create(
    binsz=0.1,
    map_type="hpx",
    skydir=skydir,
    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]
)

```

- **N-dimensional coordinate aware data structures** for storing gamma-ray data, with arbitrary number of non-spatial dimensions, such as energy, time or a label for the data class
- **Uniform API for WCS, HEALPix and region based pixelization schemes**



gammapy.makers

```

import astropy.units as u

from gammapy.data import DataStore
from gammapy.datasets import MapDataset
from gammapy.makers import (
    FoVBackgroundMaker,
    MapDatasetMaker,
    SafeMaskMaker
)
from gammapy.maps import MapAxis, WcsGeom

data_store = DataStore.from_dir(
    base_dir="$GAMMAPY_DATA/hess-dl3-dr1"
)

obs = data_store.obs(23523)

energy_axis = MapAxis.from_energy_bounds(
    energy_min="1 TeV",
    energy_max="10 TeV",
    nbins=6,
)

geom = WcsGeom.create(
    skydir=(83.633, 22.014),
    width=(4, 3) * u.deg,
    axes=[energy_axis],
    binsz=0.02 * u.deg,
)

empty = MapDataset.create(geom=geom)

```

```

maker = MapDatasetMaker()

mask_maker = SafeMaskMaker(
    methods=["offset-max", "aeff-default"],
    offset_max="2.0 deg",
)

bkg_maker = FoVBackgroundMaker(
    method="scale",
)

dataset = maker.run(empty, observation=obs)
dataset = bkg_maker.run(dataset, observation=obs)
dataset = mask_maker.run(dataset, observation=obs)
dataset.peek()

```

- [Maker](#) are configurable, stateless objects that represent an algorithm/data reduction step
- Data represented by a [MapDataset](#) is passed between the makers and holds the state
- This allows users to define data reduction chains, without access to data and also implement custom steps
- The whole process can be also defined from YAML based configuration files and executed from a “high level” [Analysis](#) class

- Reprojection of counts and IRFs

Background estimation

- A run wise estimation of the background necessary
- Different methods supported

- Reflected Background
- Ring Background

- Field of View Background

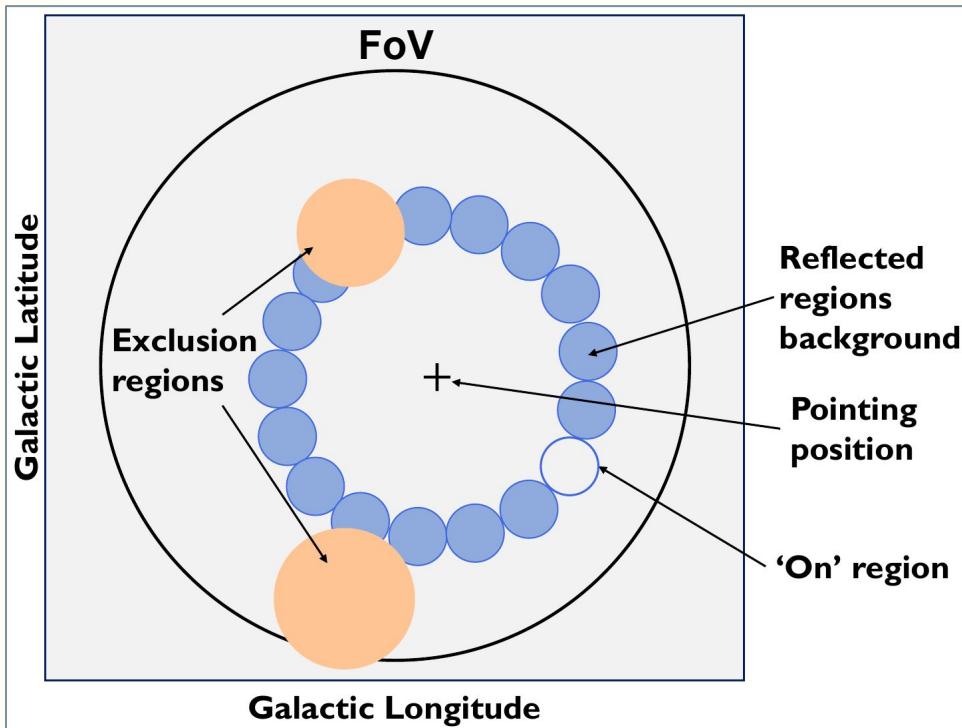
Traditional methods:
Measured off counts,
WSTAT statistics

Novel implementation:
Background modelled
simultaneous with source,
Cash statistics

1D analysis technique

Reflected regions method:

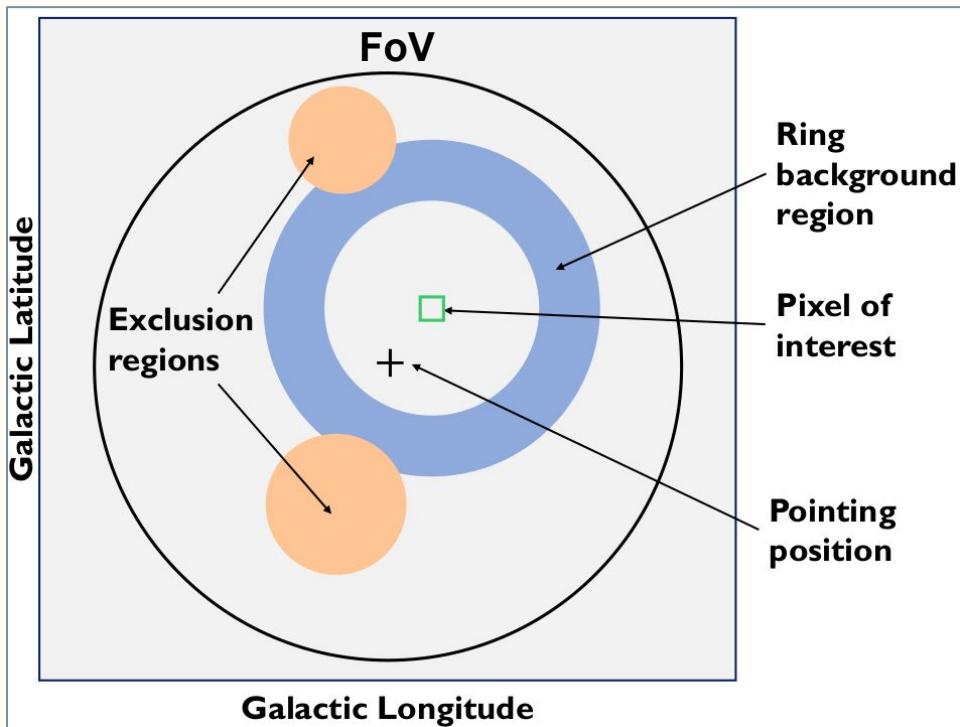
- The 'off' region has the same shape, size and offset from the pointing position as the 'on' region
- This technique is used for spectral analysis, as the spatial dimension is lost when grouping the pixels inside the 'on' region



2D analysis technique

Ring background method:

- Take a ring region for each pixel in the FoV where counts inside the ring estimate the background
- As this is performed for every point within the FoV it can be utilised to create a map
- Note: this relies on a background model



3D analysis technique

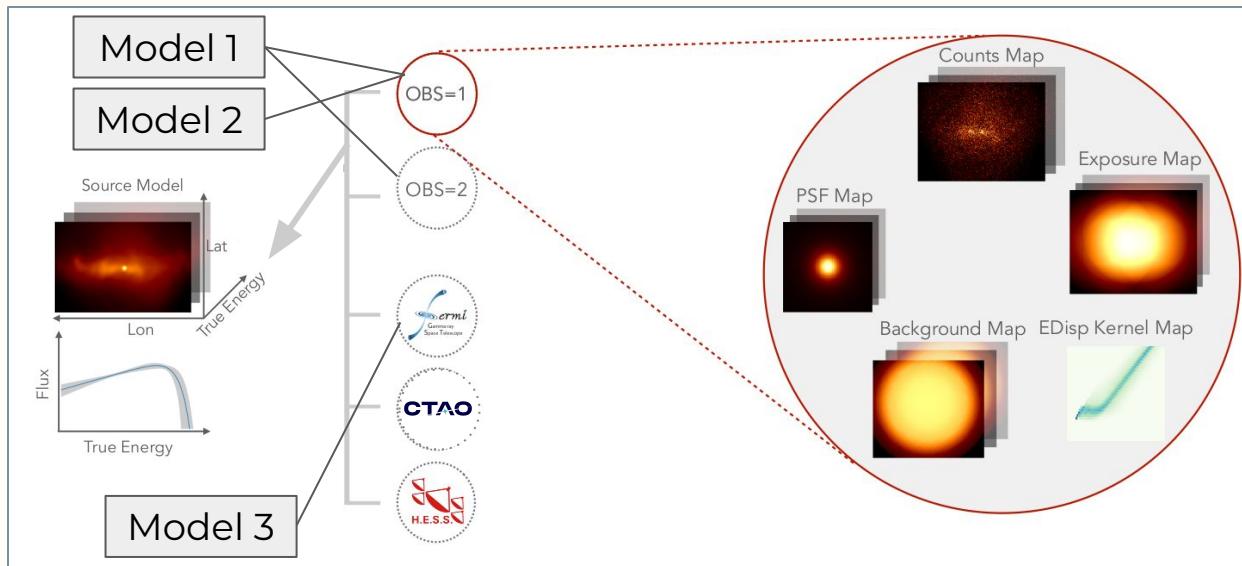
FoV background method:

- Advanced method to estimate the background is through a 3D model
- Construct a model of acceptance, and then predict the background over the field of view (FoV) as a function of energy
- Runwise FoV background is adjusted to the counts measured outside the exclusion regions, implement a specific spectra model
 - Corrects for effects that are not taken into account during the FoV background model construction
- Below shows normalisation and tilt varies in the spectral shape for the model

$$\mathcal{B}(E) \longrightarrow \mathcal{B}(E) \times \text{norm} \times \left(\frac{E}{1 \text{ TeV}} \right)^{-\text{tilt}}$$

gammapy.datasets

- Bundles binned data models & likelihood function
 - Sharing of models across datasets
- Interface to the [Fit](#) class
- Different types of datasets based upon analysis type & statistic
- Joint fitting between same/different types of datasets



gammify.modeling

```

from astropy import units as u
from gammify.modeling.models import (
    ConstantTemporalModel,
    EBLAbsorptionNormSpectralModel,
    PointSpatialModel,
    PowerLawSpectralModel,
    SkyModel,
)

# define a spectral model
pwl = PowerLawSpectralModel(
    amplitude="1e-12 TeV-1 cm-2 s-1", index=2.3
)

# define a spatial model
point = PointSpatialModel(
    lon_0="45.6 deg",
    lat_0="3.2 deg",
    frame="galactic"
)

# define a temporal model
constant = ConstantTemporalModel()

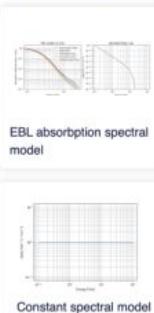
# combine all components
model = SkyModel(
    spectral_model=pwl,
    spatial_model=point,
    temporal_model=constant,
    name="my-model",
)
print(model)

ebl = EBLAbsorptionNormSpectralModel.read_builtin(
    reference="dominguez", redshift=0.5
)

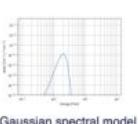
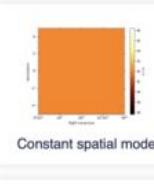
absorbed = pwl * ebl
absorbed.plot(energy_bounds=(0.1, 100) * u.TeV)

```

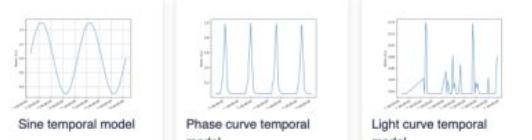
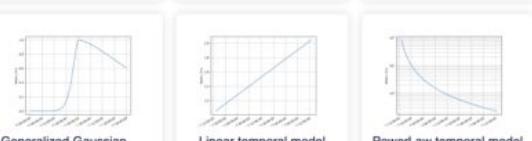
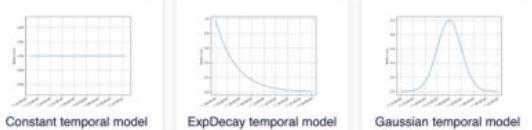
Spectral models



Spatial models



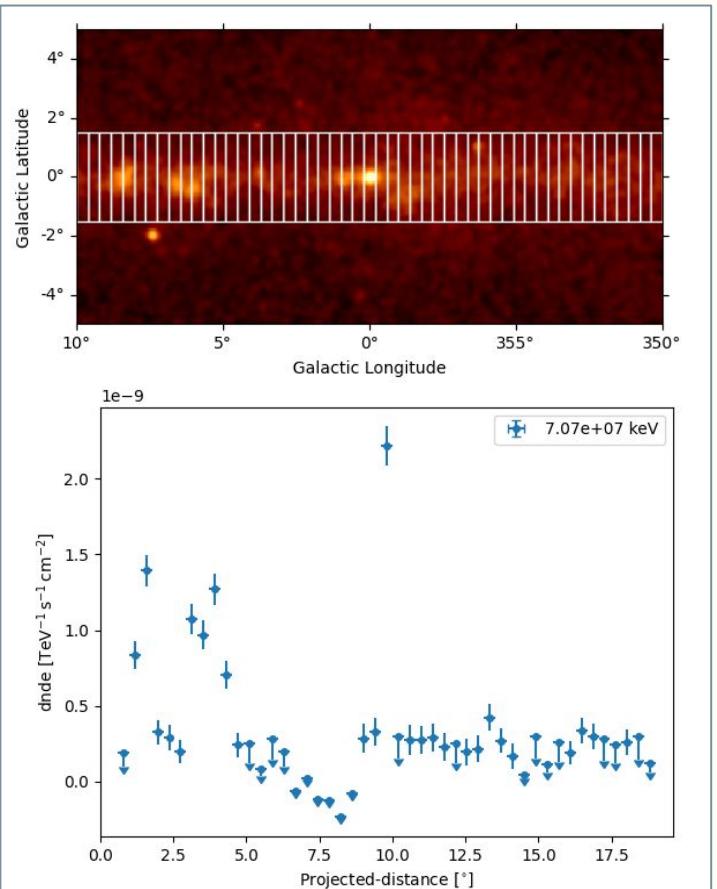
Temporal models



Users can implement **custom models** for their specific use-case, such as modeling energy dependent morphology of a source and make it available to Gammify via a **registry system**

gammapy.estimators

- Initial fine bins of a dataset are grouped to create new bins
- Multiplicative correction factor (norm) fitted to the spectral model in each bin
 - Stores the likelihood profile in each bin
 - Diagnostic and re-computation later
- Compute [FluxMaps](#), [FluxPoints](#), Lightcurves, FluxProfiles, etc
- Multiple serialisation formats supported
 - GADF-SED, Astropy table, Binned Time Series...



gammapy.catalog

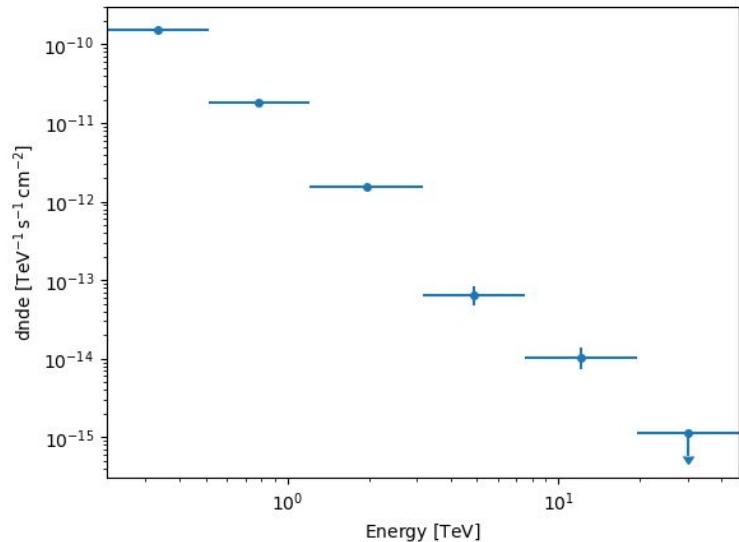
- Access to common catalogs
- Maps catalog information to gammapy objects
 - Easy to plot, access models, spectra, etc

- `hgps / SourceCatalogHGPS` - H.E.S.S. Galactic plane survey (HGPS)
- `gamma-cat / SourceCatalogGammaCat` - An open catalog of gamma-ray sources
- `3fgl / SourceCatalog3FGL` - LAT 4-year point source catalog
- `4fgl / SourceCatalog4FGL` - LAT 8-year point source catalog
- `2fhl / SourceCatalog2FHL` - LAT second high-energy source catalog
- `3fhl / SourceCatalog3FHL` - LAT third high-energy source catalog
- `2hwc / SourceCatalog2HWC` - 2HWC catalog from the HAWC observatory
- `3hwc / SourceCatalog3HWC` - 3HWC catalog from the HAWC observatory

```
from gammapy.catalog import SourceCatalogHGPS
catalog = SourceCatalogHGPS()
print(len(catalog.table))
src = catalog['HESS J1804-216']
src.flux_points.plot()
```

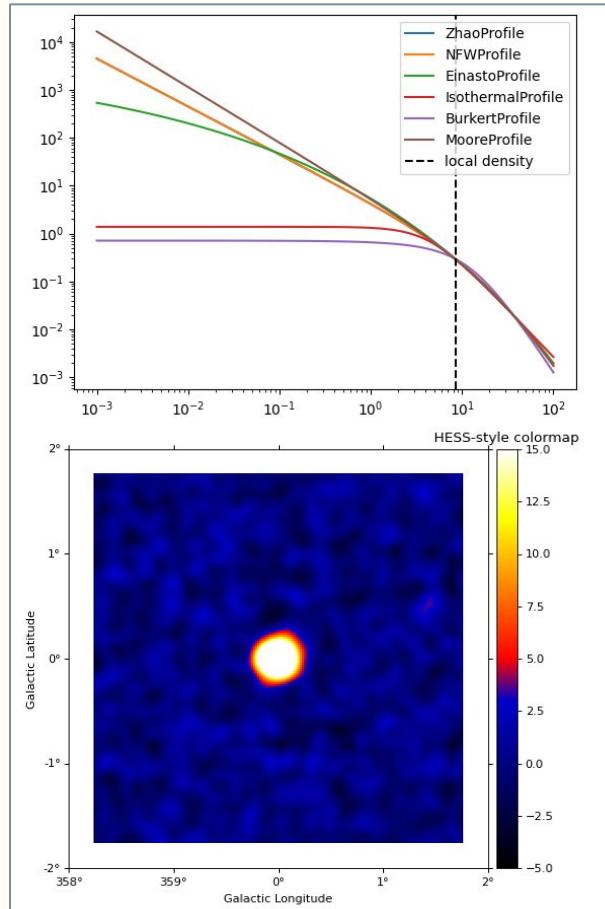
78

<Axes: xlabel='Energy [TeV]', ylabel='dnde [TeV^-1 s^-1 cm^-2]'



Additional packages

- [gammapy.stats](#)
 - Statistical utility functions
- [gammapy.astro](#)
 - Utility functions for studying physical scenarios in high-energy astrophysics
- [gammapy.visualization](#)
 - Helper functions for plotting and visualising analysis results and Gammapy data structures
- [gammapy.analysis](#)
 - High level interface
 - python scripts, notebooks...
 - Automatize work flows driven by parameters declared in a configuration file in YAML format

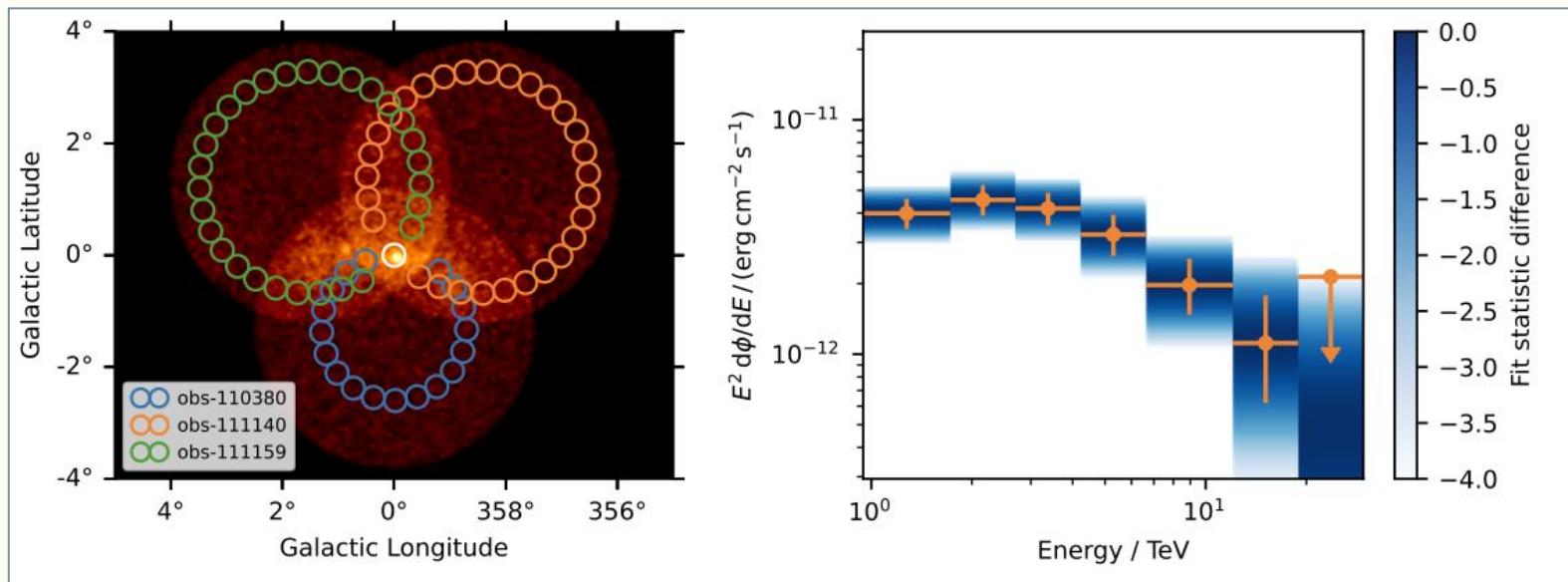


Analysis examples

Spectral Analysis

Classical Method

- Utilise the reflected regions method as mentioned previously
- Along with the forward folding method → maximum likelihood
- Utilise simulated CTA data – likelihood per energy bin is shown by the blue colored band



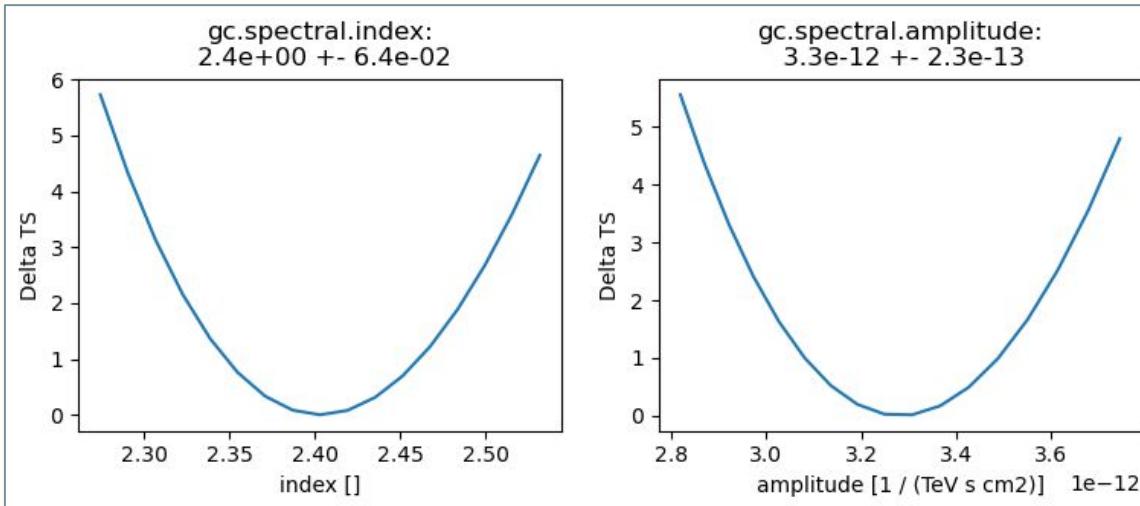
Spectral Analysis

Classical Method

- Maximum likelihood = minimum test statistic
- Is the minimum well defined?

YES!

$$TS = -2 \log \left(\frac{\mathcal{L}_0}{\mathcal{L}_1} \right)$$

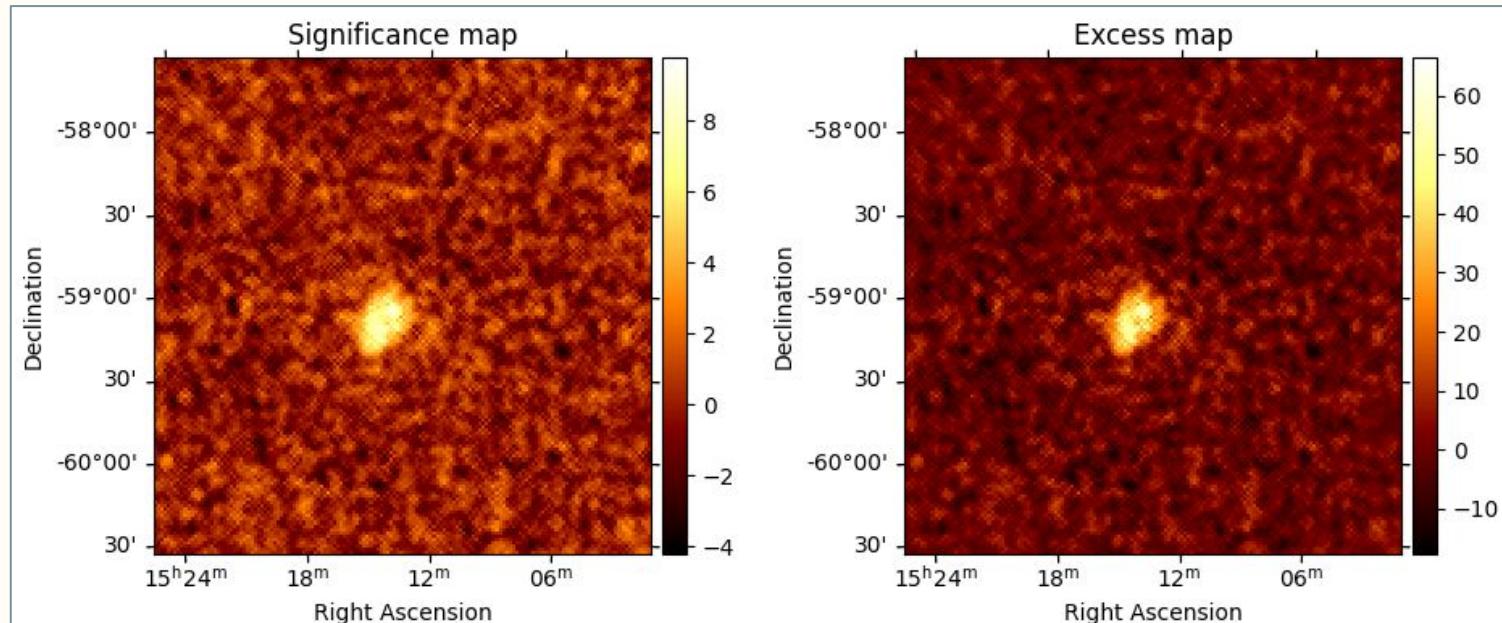


[See this tutorial](#)

- The same methodology is utilised for an extended source with one key difference:
 - The values of the IRFs are averaged over the entire region
 - [See this tutorial](#)

Spatial Analysis Classical 2D Method

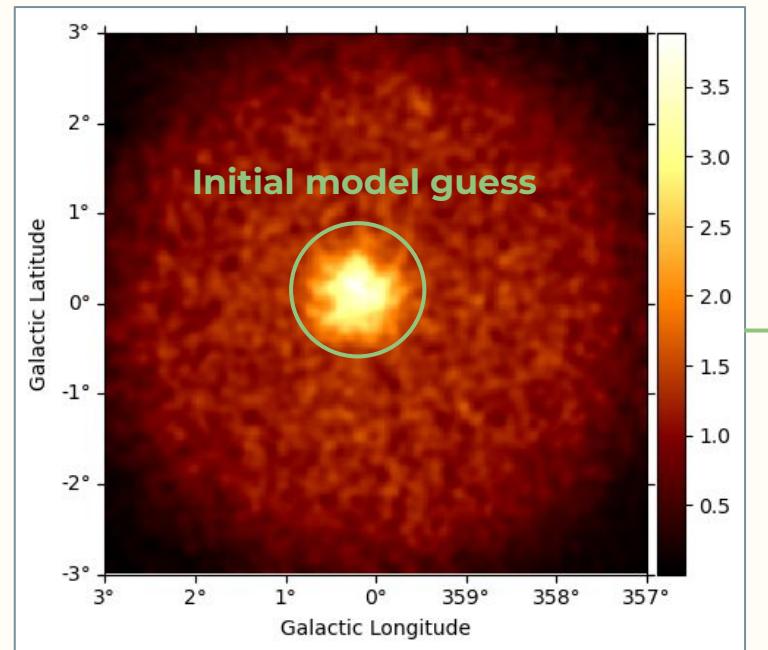
- Utilise the ring background method as explained previously
- Select OFF events from an annulus about the ON region
- Requires good knowledge of 2D acceptance!



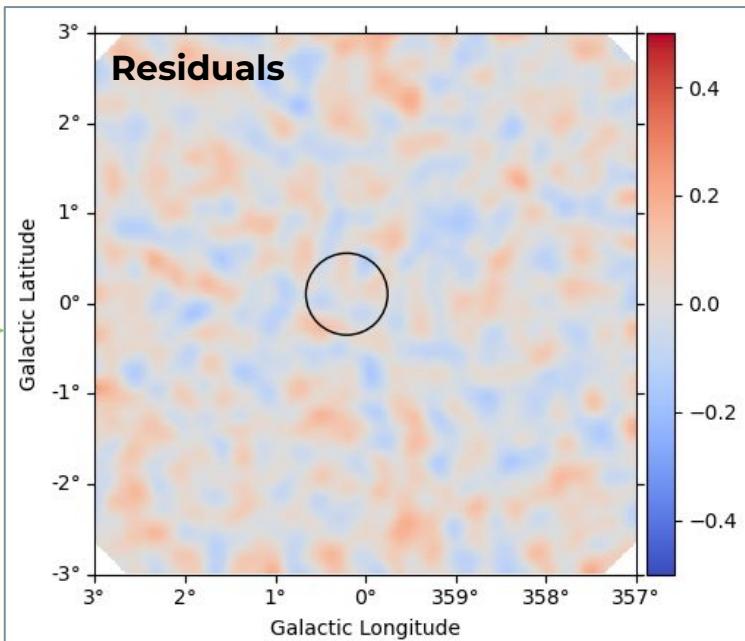
Spatial and Spectral Analysis

★ The new 3D Method ★ $\gamma\pi$

- Utilise the FoV background method which allows for a 3D analysis
- In one analysis chain we can fit both the spectra and the morphology of a source



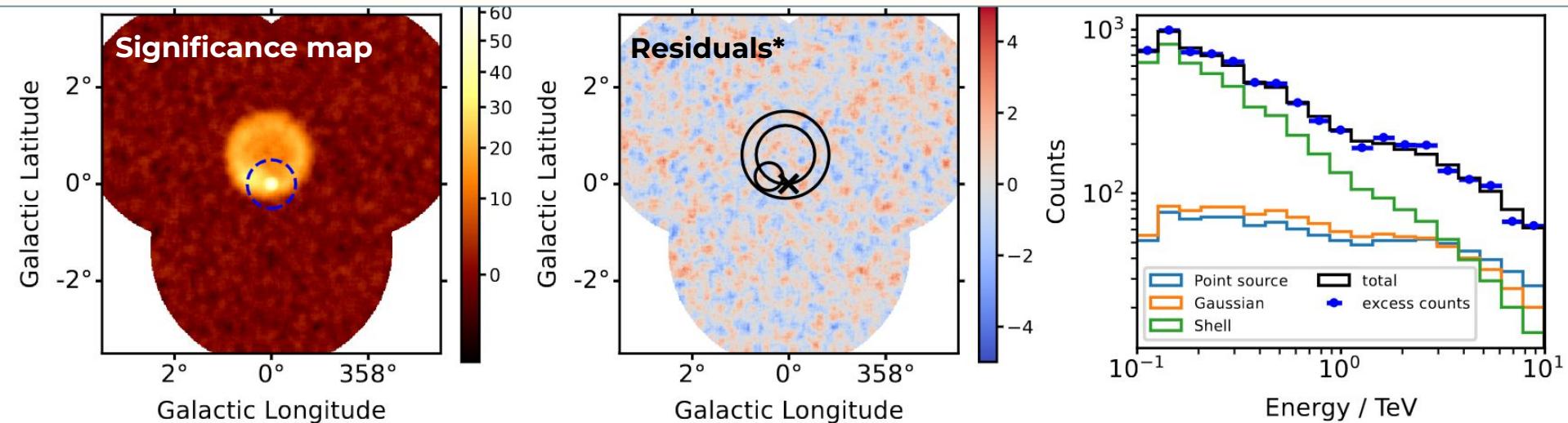
Fit both the
spectral
and spatial
model



Power of the 3D method

It is possible to disentangle contributions of overlapping sources!

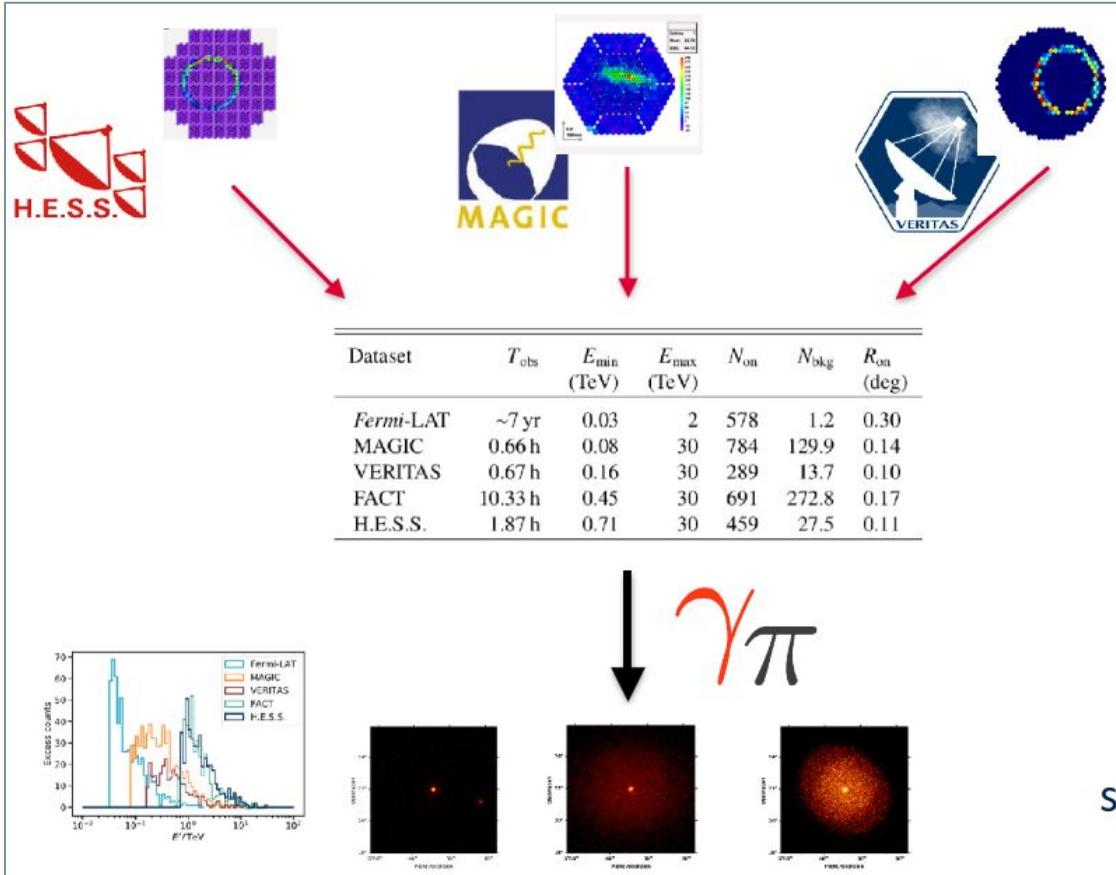
- For example:
 - Point source with power law spectrum
 - Gaussian source with log-parabola spectrum
 - Shell with power law spectrum



*Significance map after
subtracting the best-fit model for
each of the sources

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.

Joint analysis



Raw data

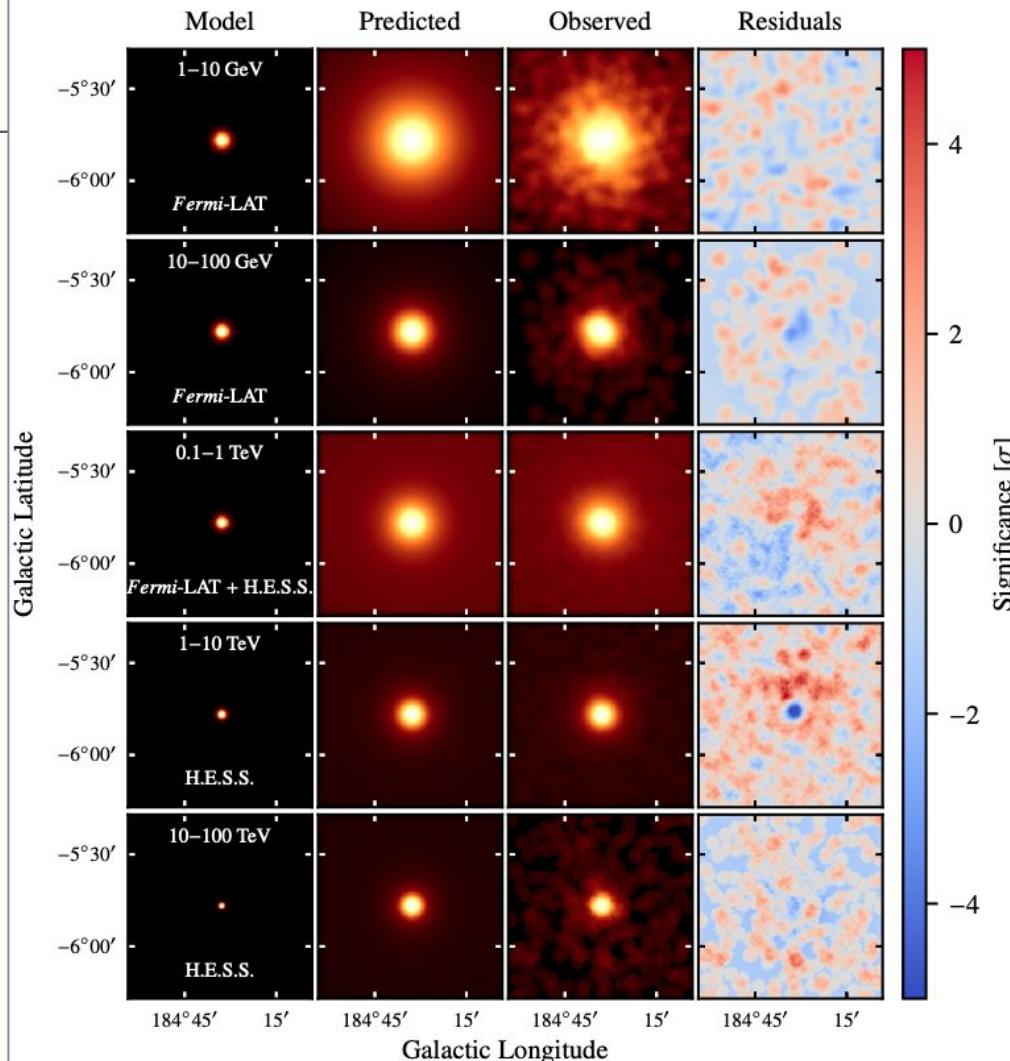
DL3 GADF

High level science
products

Joint analysis

Constrain extensions

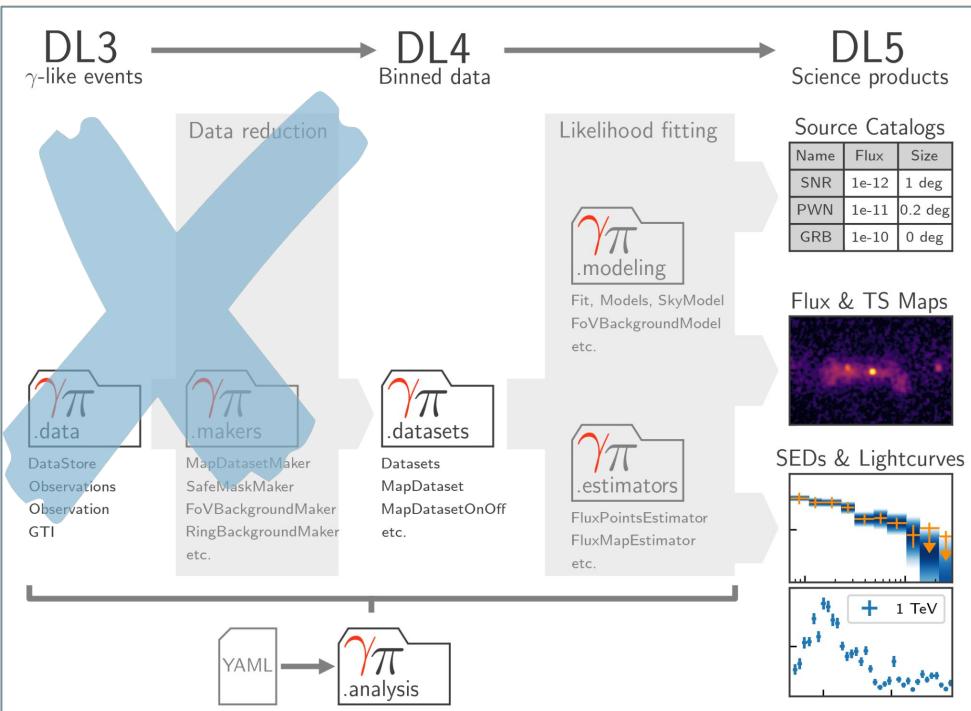
- Joint Fermi-LAT H.E.S.S. analysis used to constrain the extension of the Crab Nebula
- Probe structures to understand the underlying mechanisms



Fermi-LAT with gammapy

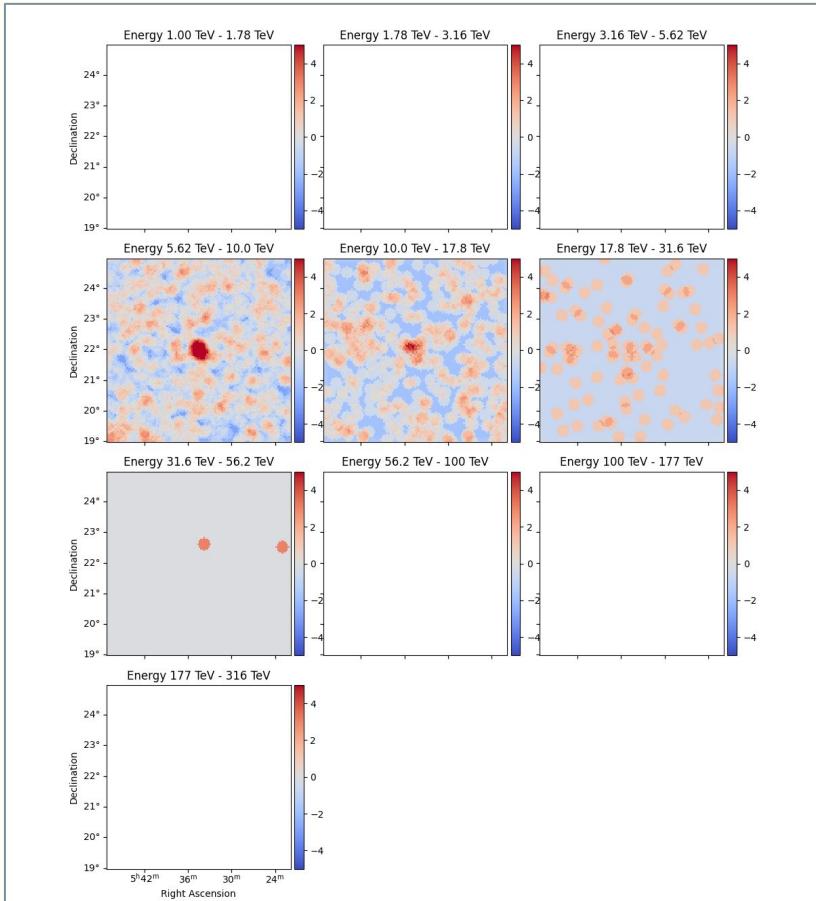


- Analysis starts from DL4 data levels:
 - After binning and reproduction
- Once you have a DL4 product “Dataset”, modelling and fitting proceeds as before
- Bonus: Simulating datasets
- Note: Fermi-LAT analysis is always 3D



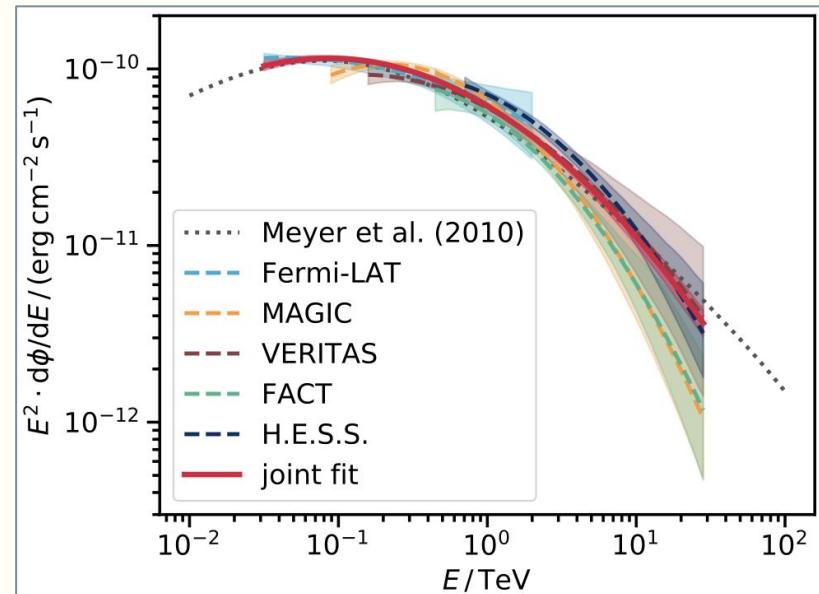
HAWC with gammipy

- Events: DL3 level
- IRFs: DL4 level
- Joint analysis for different fHit bins
- Background and exposure calculated per transit for a source
- Correct by the number of transits per source computed from the GTIs

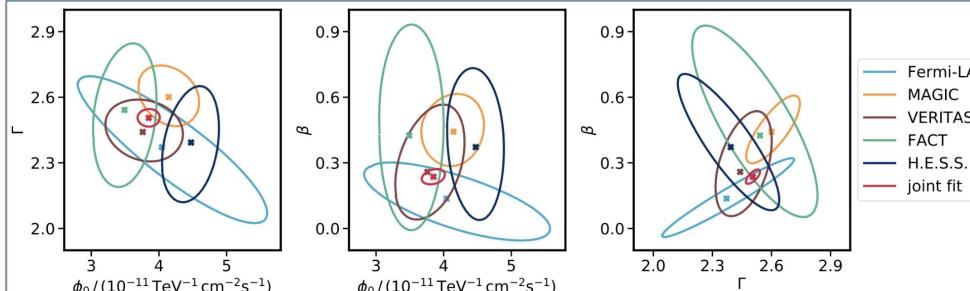


Multi-instrument analysis

- Spectral fit combining different types of data
 - Fermi-LAT DL4 data - full 3D analysis
7 yrs of data
 - MAGIC DL3 data - point like 1D spectral analysis
40 mins of data
 - VERITAS DL3 data - point like 1D spectral analysis
40 mins of data
 - FACT - point like 1D spectral analysis
10.3 hrs of data
 - H.E.S.S. - full containment 3D analysis
2 hrs of data



Better constrained parameters



Nigro et al., 2019

Thanks for your attention



```
from gammypy import song  
song(karaoke=True)
```