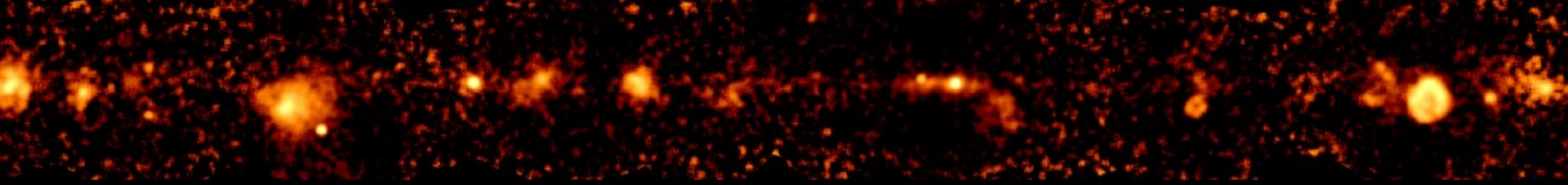
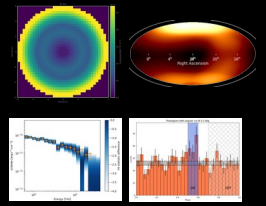


Introduction to the Gammapy library
and the data analysis workflow

Bruno Khélifi, Kirsty Feijen

Cherenkov Astronomy Data School

November 3rd-7th, 2025 – 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!)

Slide credit: Axel Donath

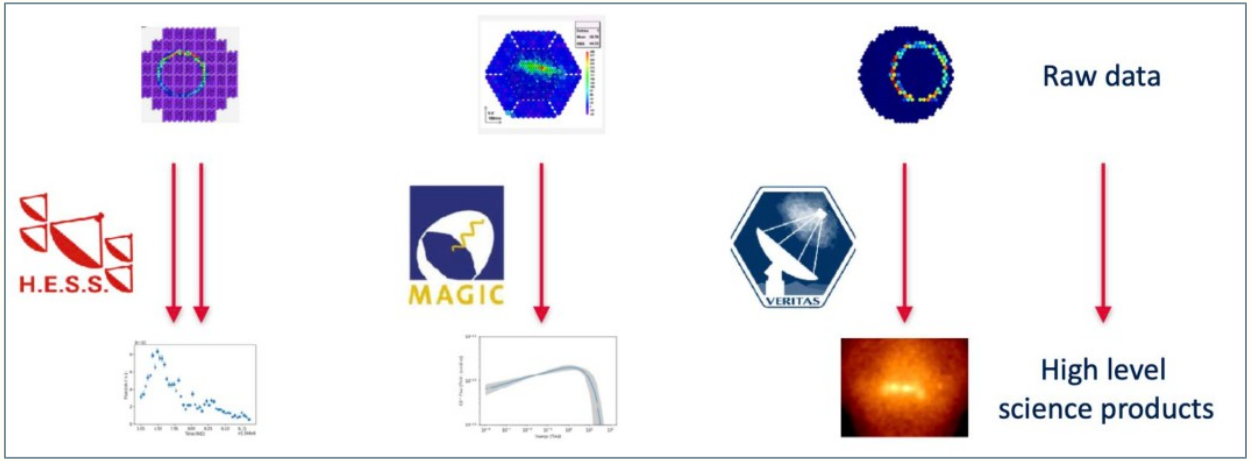
Gamma-ray instruments

- Ground based **Imaging Atmospheric Cherenkov Telescopes**, H.E.S.S., VERITAS, MAGIC, FACT, CTAO*
  - 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

Slide credit: Axel Donath

Gammapy 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 **Gammapy** we can

Slide credit: Atreyee Sinha

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

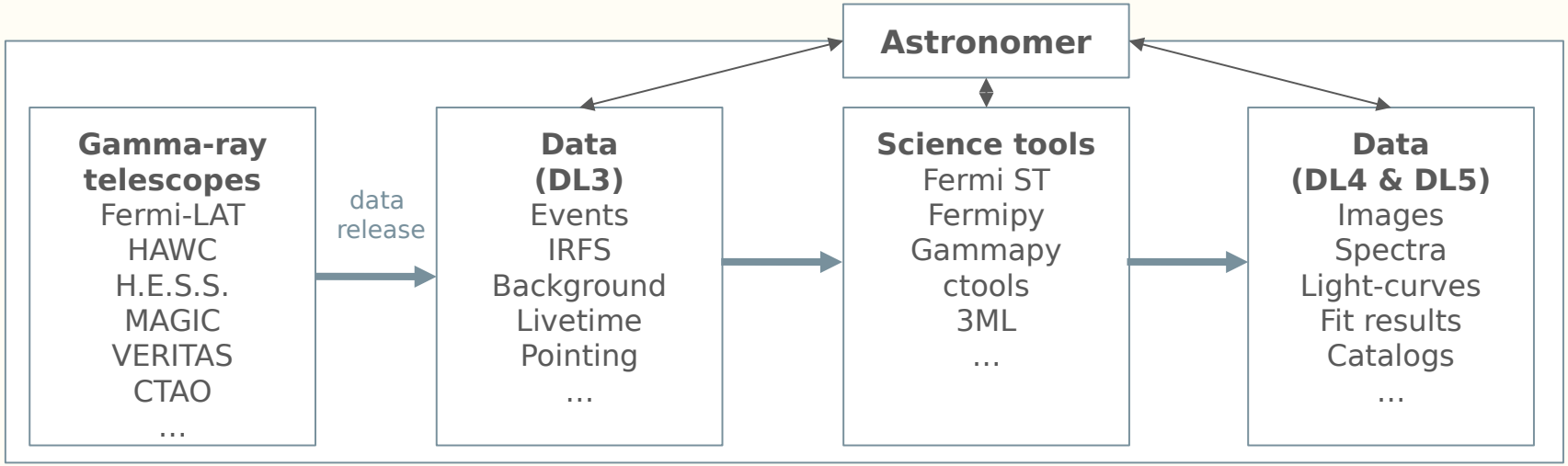
Slide credit: Atreyee Sinha

Format for the data?

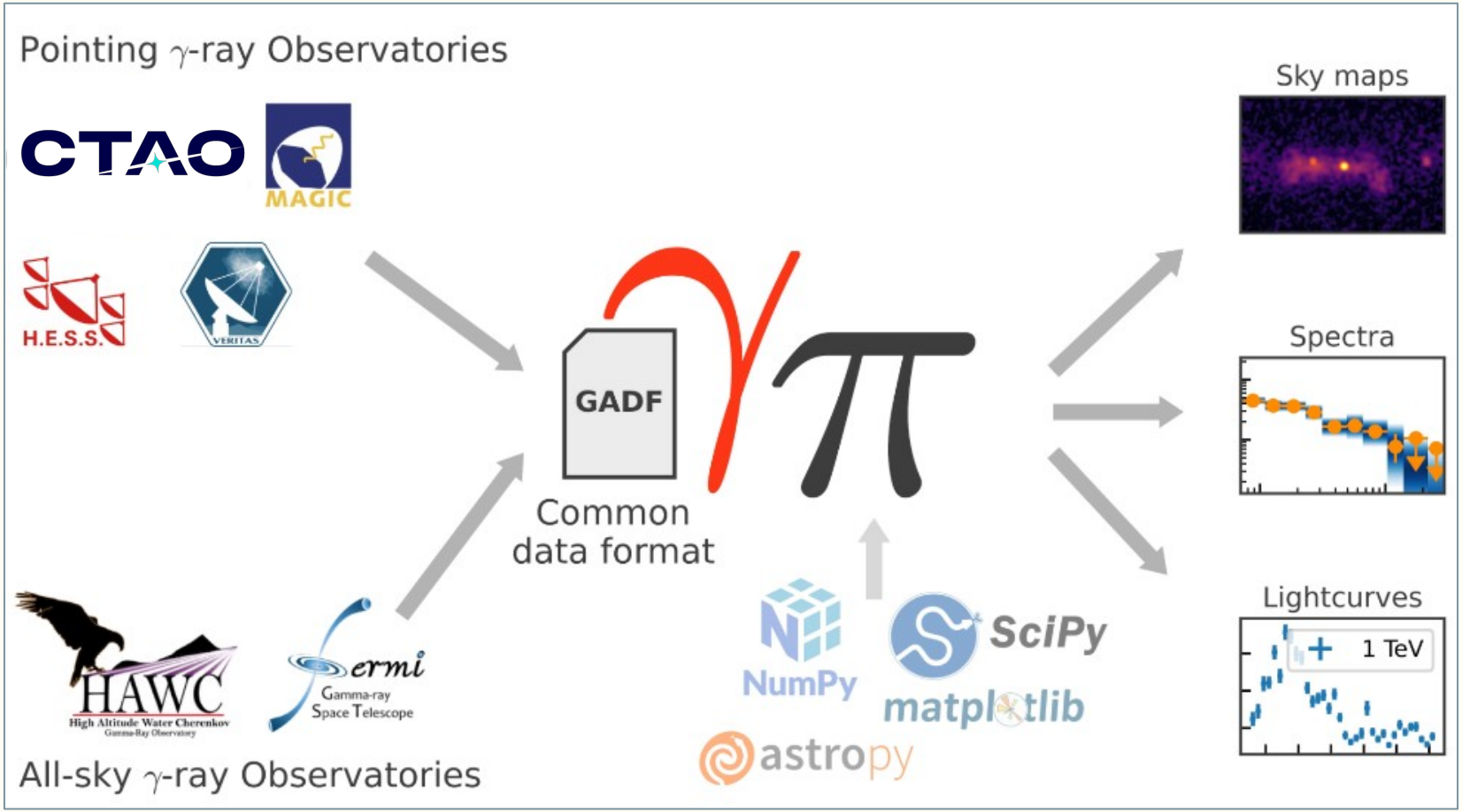
- In the ~10 years, there has been a big effort to define a common format for storing gamma-ray data
- We learn from existing standards for X-ray astronomy and Fermi
- Result: Gamma-Astro-Data-Format (GADF)
  - Based on FITS standards, follow FITS conventions for time and coordinates
  - Information 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!

GADF

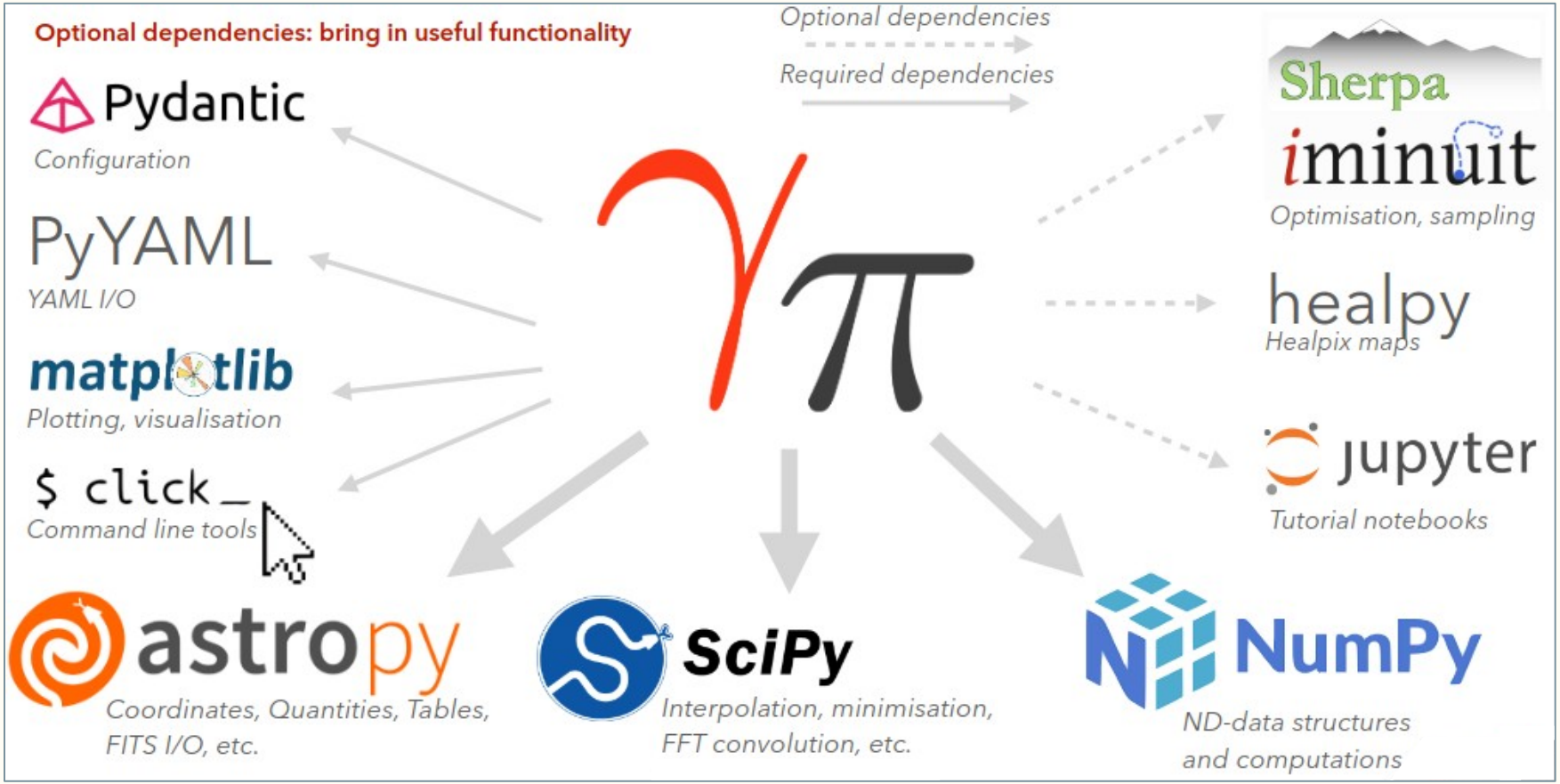
- Gamma Astro Data Format (GADF)
- Based on the Fermi-LAT format proposed in 2016
- Purpose of GADF is to encourage the collaboration between high-level gamma-ray data producers, science tool developers and data analysis
- Adopted by CTAO, H.E.S.S. DL3 DR1, MAGIC data release
- Goal: develop common data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



Gammapy overview

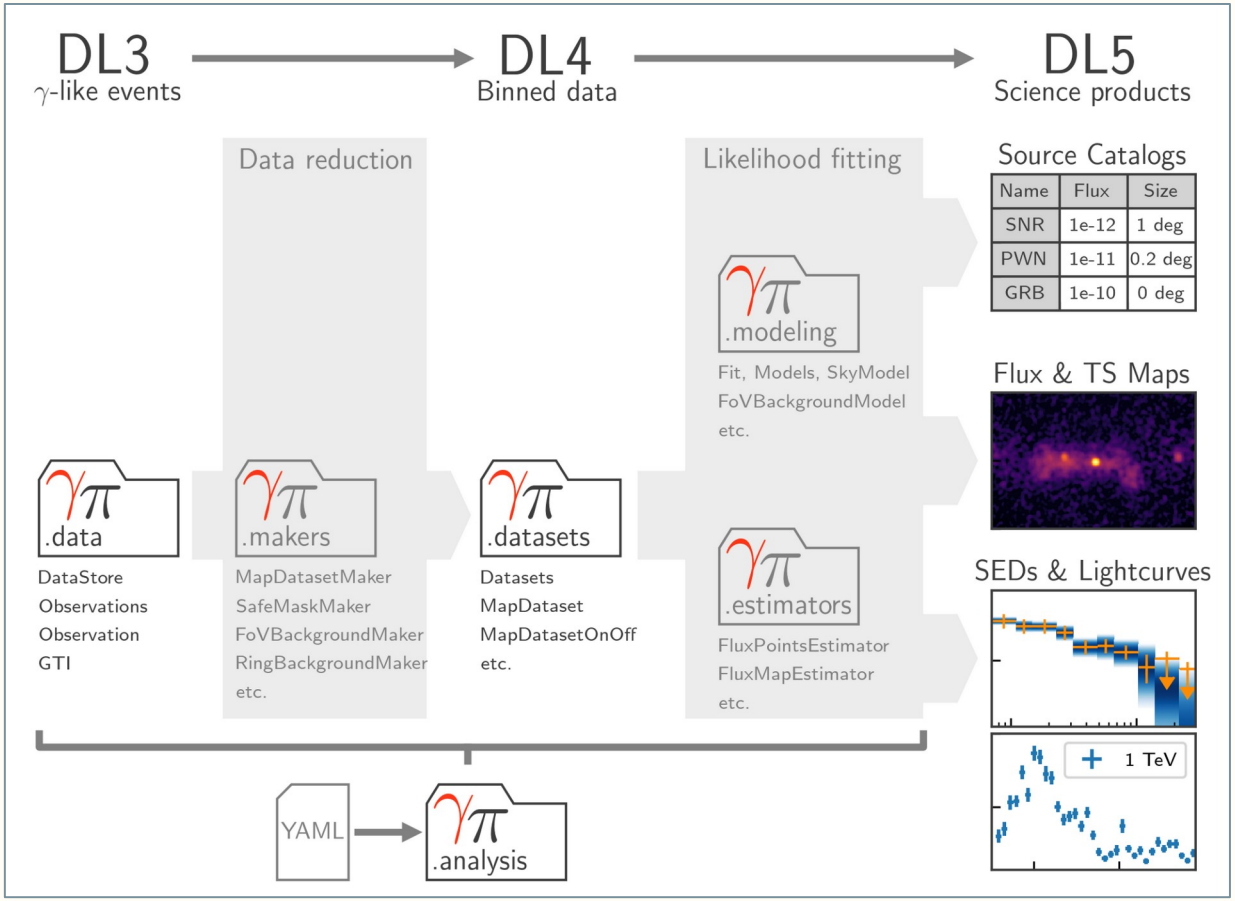


Dependencies



Slide credit: Axel Donath

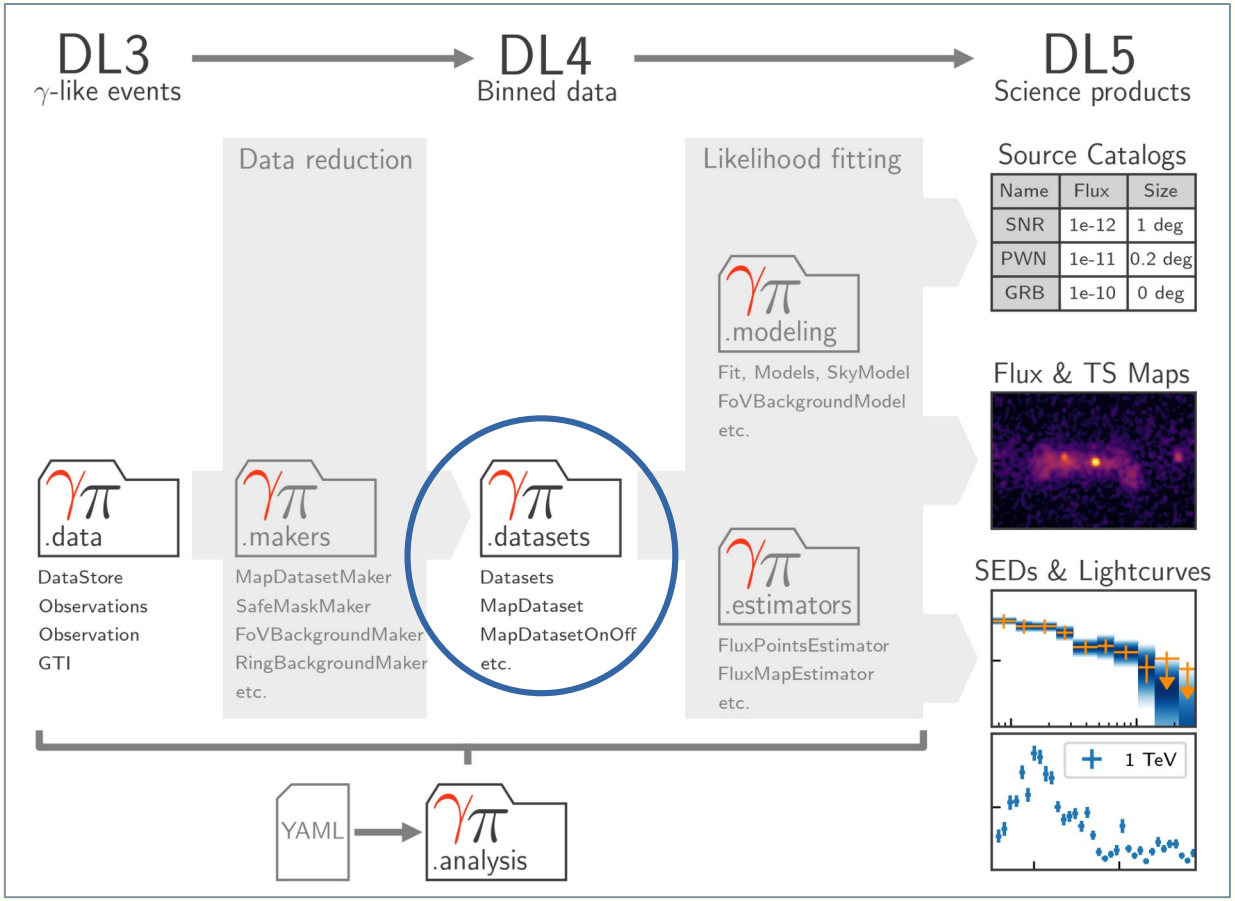
Internal workflow



2-step analysis procedure

- Data reduction :
  from DL3 to DL4
- Data modeling/fitting :
  from DL4 to DL5

Internal workflow



2-step analysis procedure

- Data reduction :
  from DL3 to DL4
- Data modeling/fitting :
  from DL4 to DL5

Getting Started

Getting helping, reporting issues

How to provide feedback/get help

- #help channel on gammapy.slack
  - There are private channels per experiment
- #gammapy channel on hesschat.slack (if you are a H.E.S.S. member)
- GitHub discussion, in particular the help category

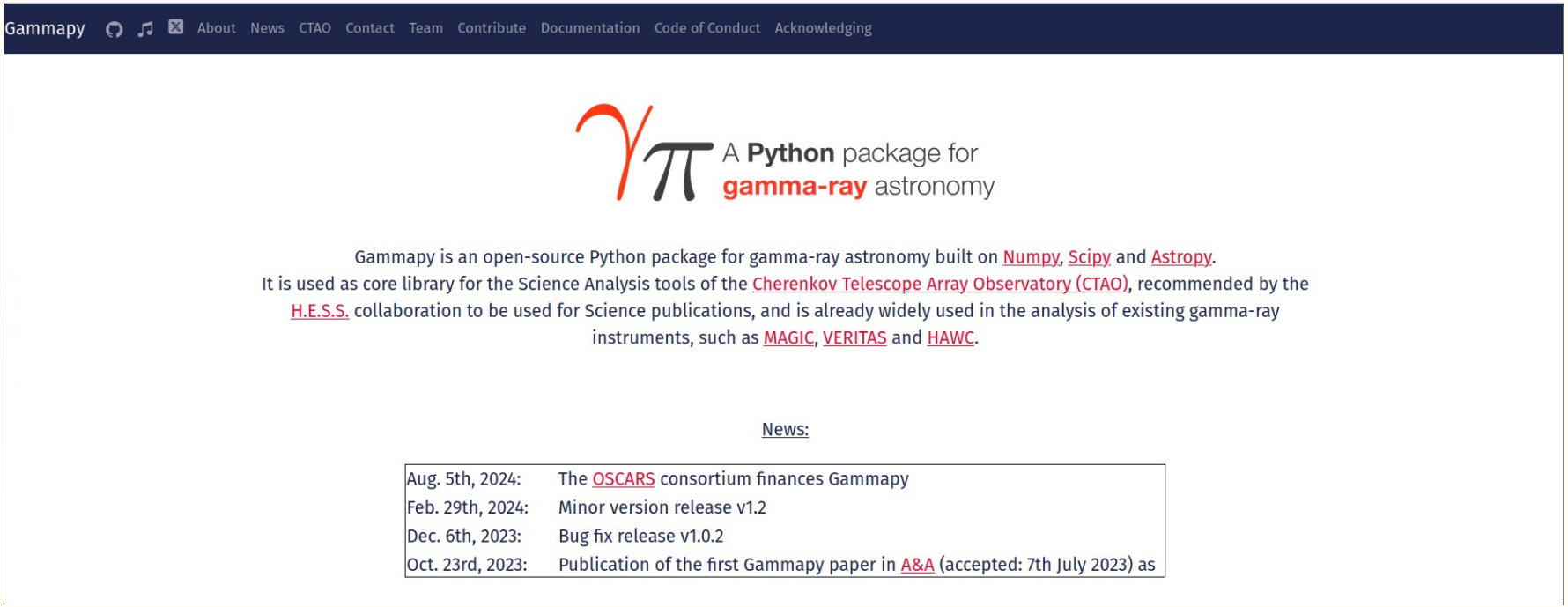
How to report issues and bugs or request a new feature

- GitHub issues page (requires a GitHub account)

Gammapy & CTAO

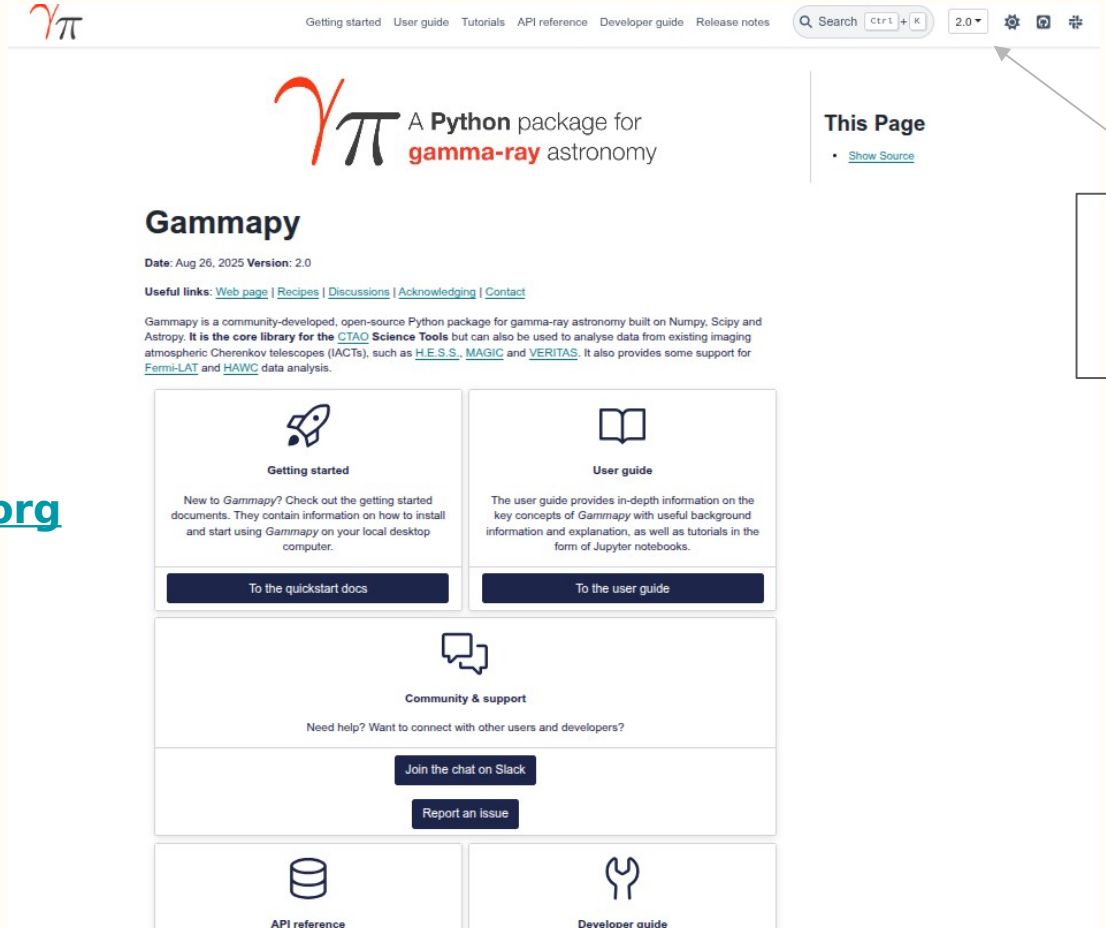
- **v2** was released August 2025
- Gammapy selected as the official CTAO Science Analysis Tool (SAT) - 2021
- Gammapy team involved in Science Data Challenge (SDC) effort
  - Event types are supported
    - Distributed as two different data stores
  - Stacked or joint analyses are possible
  - Missing features for SDC have been added:
    - Time dependent spectral models
  - Metadata containers to support CTAO data model
  - SDC will be important to prepare the first release

Getting started: webpage



See gammapy.org

Getting started: documentation

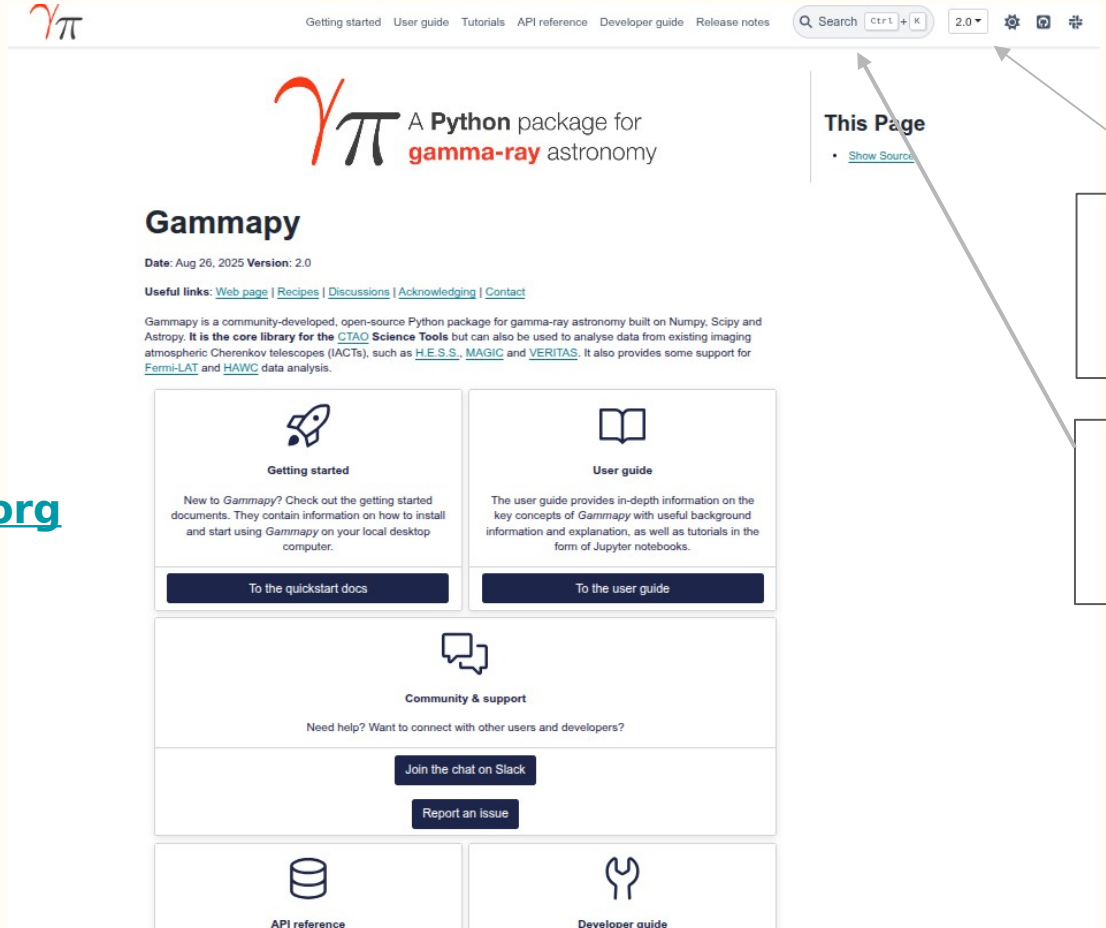


Switch between versions

See docs.gammapy.org

Getting started: documentation

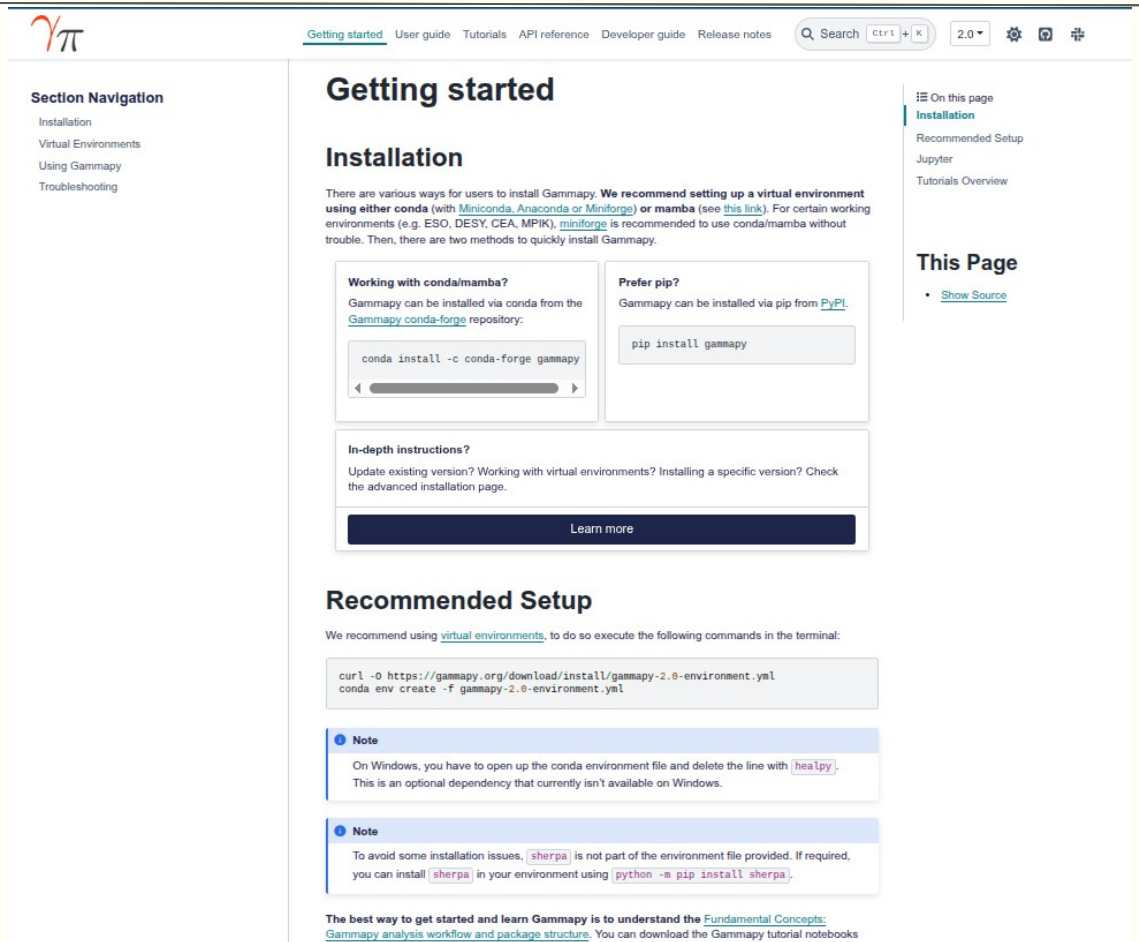
See docs.gammapy.org



Switch between versions

Search bar

Getting started



Getting started User guide Tutorials API reference Developer guide Release notes Search Ctrl + K 2.0

Section Navigation
- Installation
- Virtual Environments
- Using Gammapy
- Troubleshooting

Getting started

Installation

There are various ways for users to install Gammapy. **We recommend setting up a virtual environment using either conda** (with Miniconda, Anaconda or Miniforge) **or mamba** (see this link). For certain working environments (e.g. ESO, DESY, CEA, MPIK), miniforge is recommended to use conda/mamba without trouble. Then, there are two methods to quickly install Gammapy.

Working with conda/mamba?

Gammapy can be installed via conda from the Gammapy conda-forge repository:

```
conda install -c conda-forge gammapy
```

Prefer pip?

Gammapy can be installed via pip from PyPI.

```
pip install gammapy
```

In-depth instructions?

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

Learn more

Recommended Setup

We recommend using virtual environments, to do so execute the following commands in the terminal:

```
curl -O https://gammapy.org/download/install/gammapy-2.0-environment.yml
conda env create -f gammapy-2.0-environment.yml
```

Note

On Windows, you have to open up the conda environment file and delete the line with `healpy`. This is an optional dependency that currently isn't available on Windows.

Note

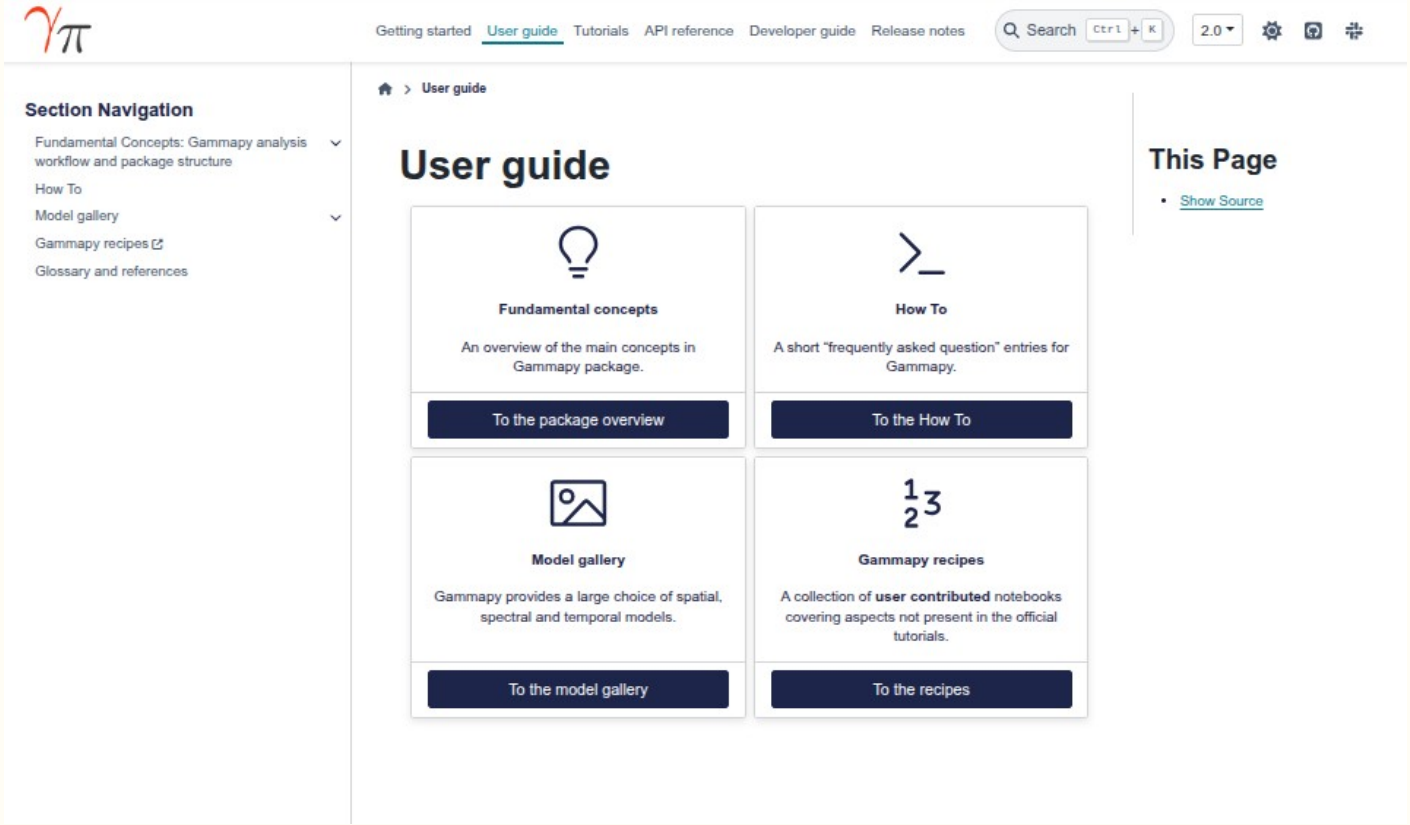
To avoid some installation issues, `sherpa` is not part of the environment file provided. If required, you can install `sherpa` in your environment using `python -m pip install sherpa`.

The best way to get started and learn Gammapy is to understand the Fundamental Concepts: Gammapy analysis workflow and package structure. You can download the Gammapy tutorial notebooks

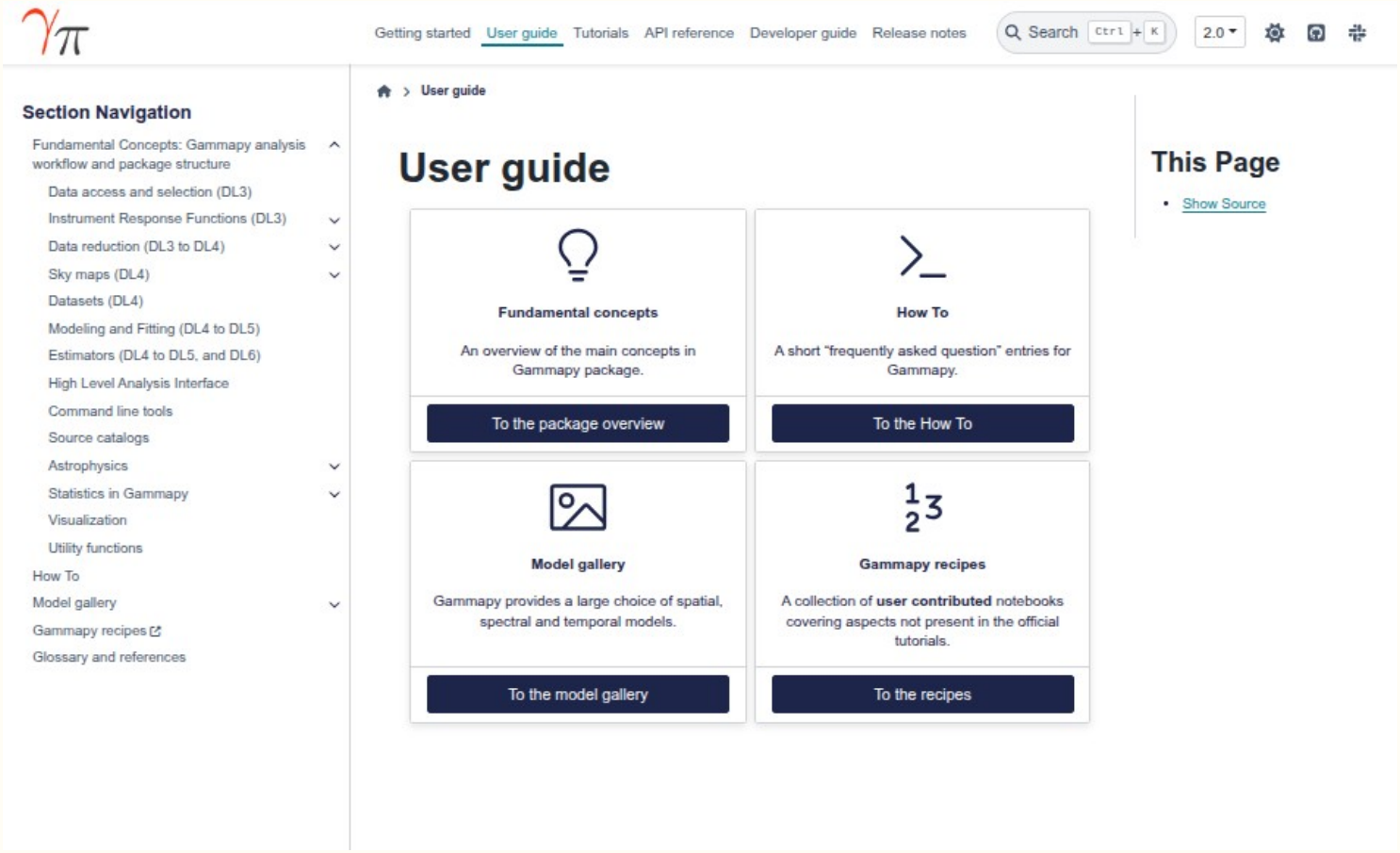
On this page
- Installation
- Recommended Setup
- Jupyter
- Tutorials Overview

This Page
- Show Source

User guide


Getting started
User guide
Tutorials
API reference
Developer guide
Release notes
Search Ctrl + K
2.0
Section Navigation
Fundamental Concepts: Gammapy analysis workflow and package structure
How To
Model gallery
Gammapy recipes
Glossary and references
User guide
User guide
Fundamental concepts
An overview of the main concepts in Gammapy package.
To the package overview
How To
A short "frequently asked question" entries for Gammapy.
To the How To
Model gallery
Gammapy provides a large choice of spatial, spectral and temporal models.
To the model gallery
Gammapy recipes
A collection of user contributed notebooks covering aspects not present in the official tutorials.
To the recipes
This Page
Show Source

Drop down menus



Fundamental concepts

Fundamental Concepts: Gammapy analysis workflow and package structure

Analysis workflow

Fig. 1 illustrates the standard analysis flow and the corresponding sub-package structure of Gammapy. Gammapy 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. Using data distributed following the Gamma Astro Data Formats, the different data levels and data reduction steps and how they map to the Gammapy API are explained in more detail in the following.

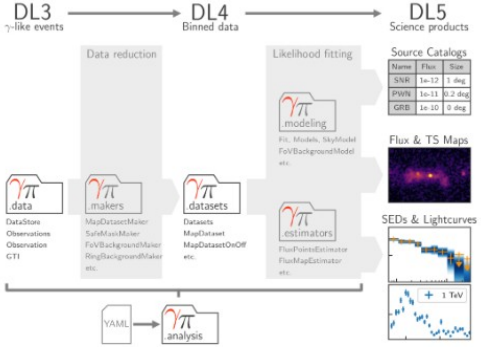


Fig. 1 Data flow and sub-package structure of Gammapy. The folder icons represent the corresponding sub-packages. The direction of the the data flow is illustrated with shaded arrows. The top section shows the data levels as defined by CTAO.

Analysis steps

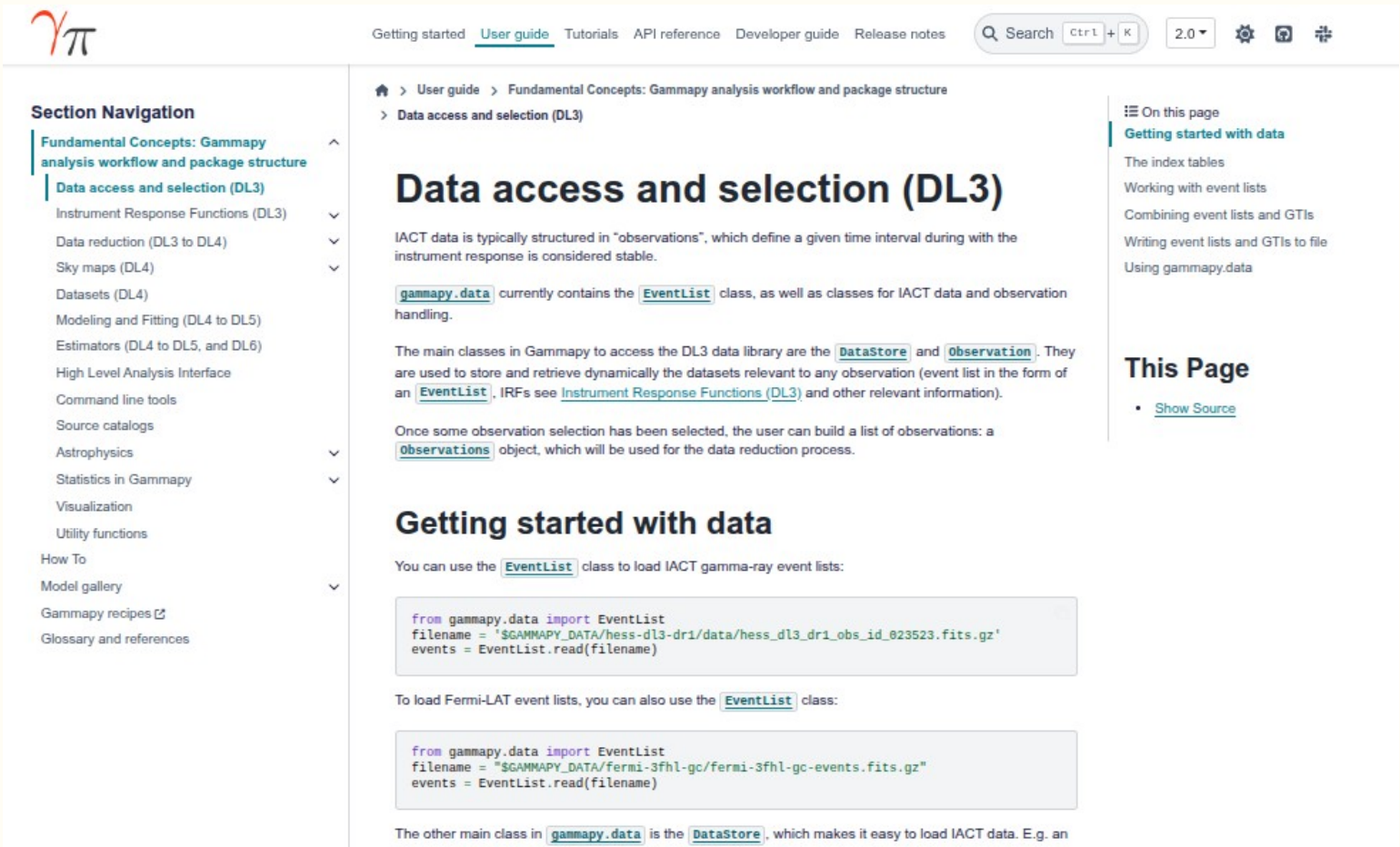
Data access and selection (DL3)

The analysis of gamma-ray data with Gammapy starts at the "data level 3". At this level the data is stored as lists of gamma-like events and the corresponding instrument response functions (IRFs).

Instrument Response Functions (DL3)

Gammapy supports various forms of instrument response functions (IRFs), which are represented as

DL3



Getting started | User guide | Tutorials | API reference | Developer guide | Release notes | Search Ctrl + K | 2.0

Section Navigation

- Fundamental Concepts: Gammapy 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
  - Statistics in Gammapy
  - Visualization
  - Utility functions
- How To
- Model gallery
- Gammapy recipes
- Glossary and references

User guide > Fundamental Concepts: Gammapy analysis workflow and package structure > Data access and selection (DL3)

Data access and selection (DL3)

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

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

The main classes in Gammapy 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.

Getting started with data

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

```
from gammapy.data import EventList
filename = '$GAMMAPY_DATA/hess-dl3-dr1/data/hess_dl3_dr1_obs_id_023523.fits.gz'
events = EventList.read(filename)
```

To load Fermi-LAT event lists, you can also use the `EventList` class:

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

The other main class in `gammapy.data` is the `DataStore`, which makes it easy to load IACT data. E.g. an

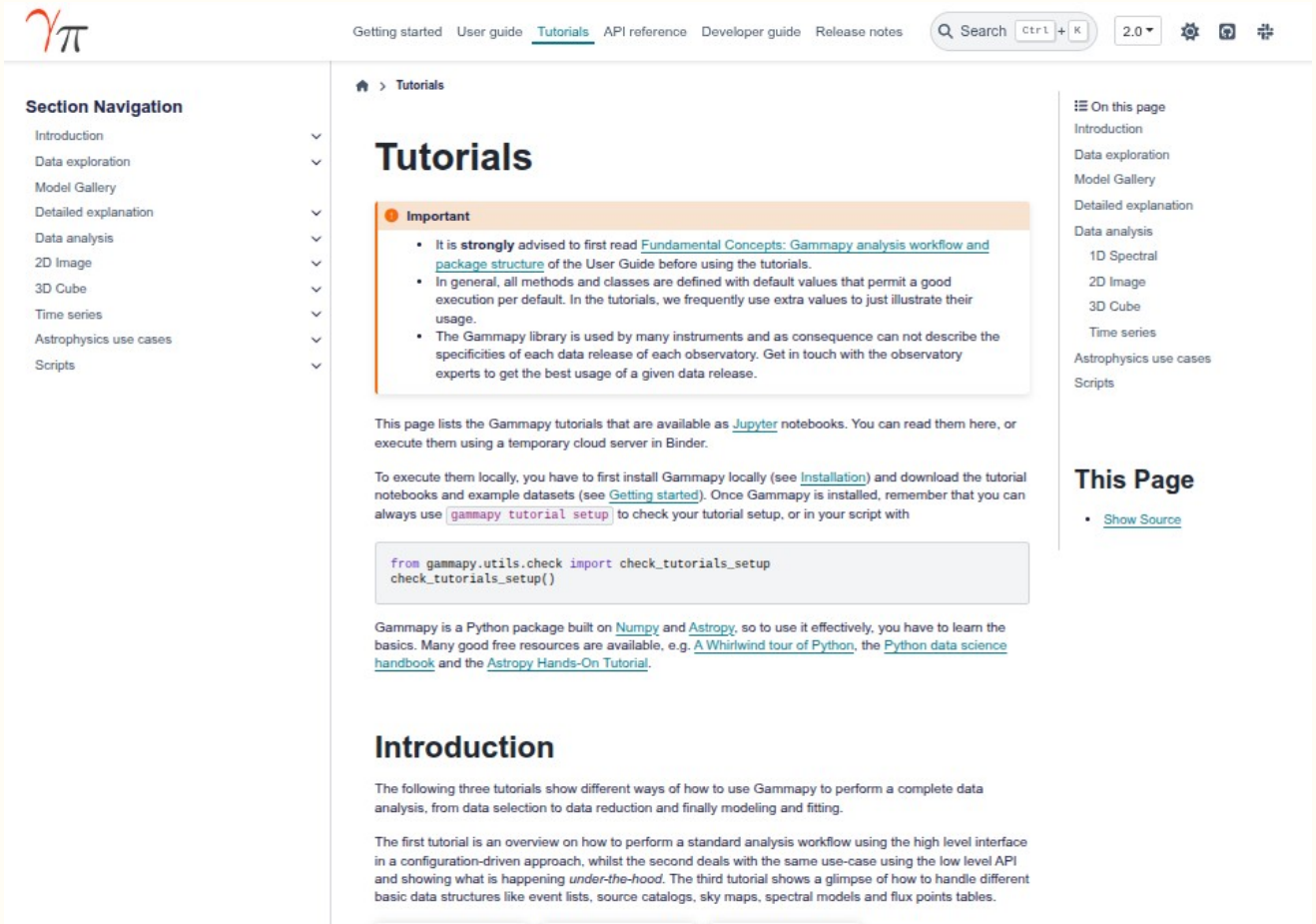
On this page

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

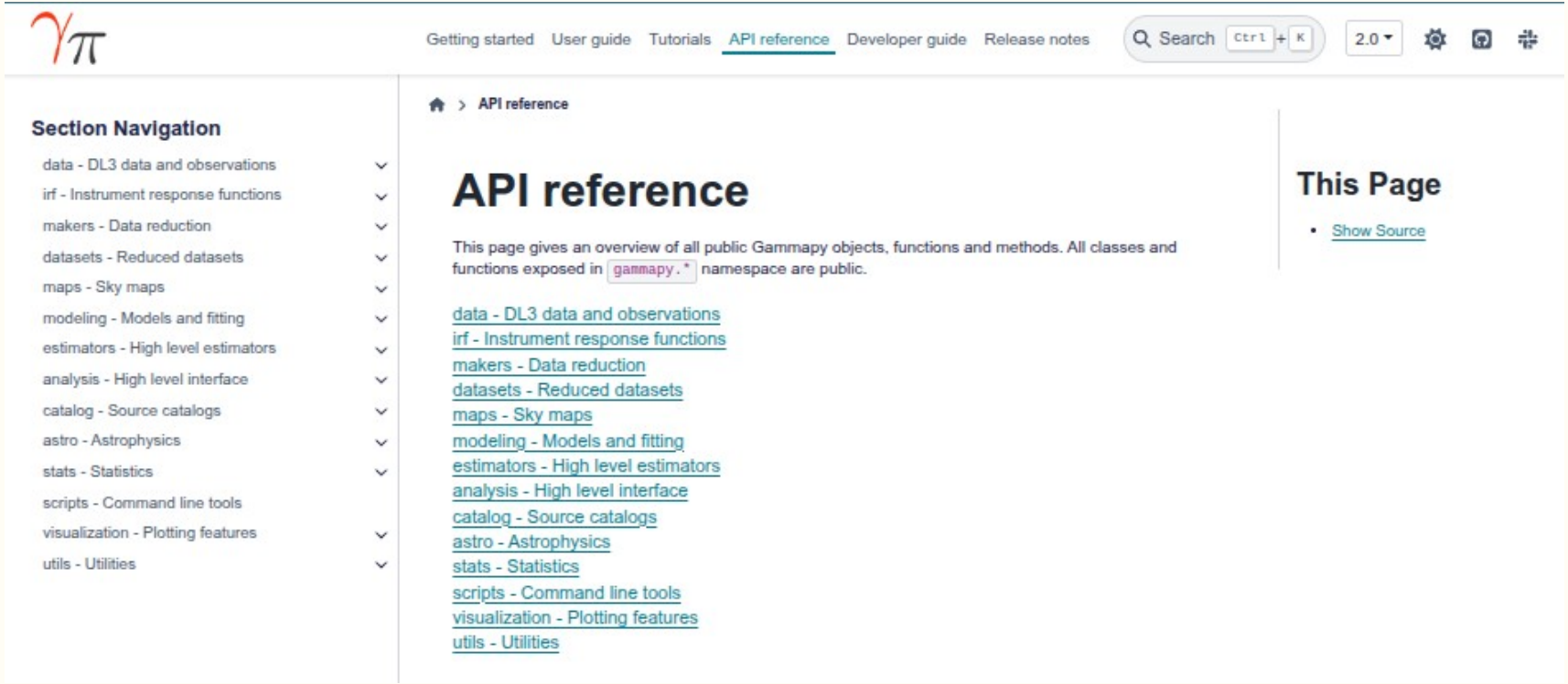
This Page

- Show Source

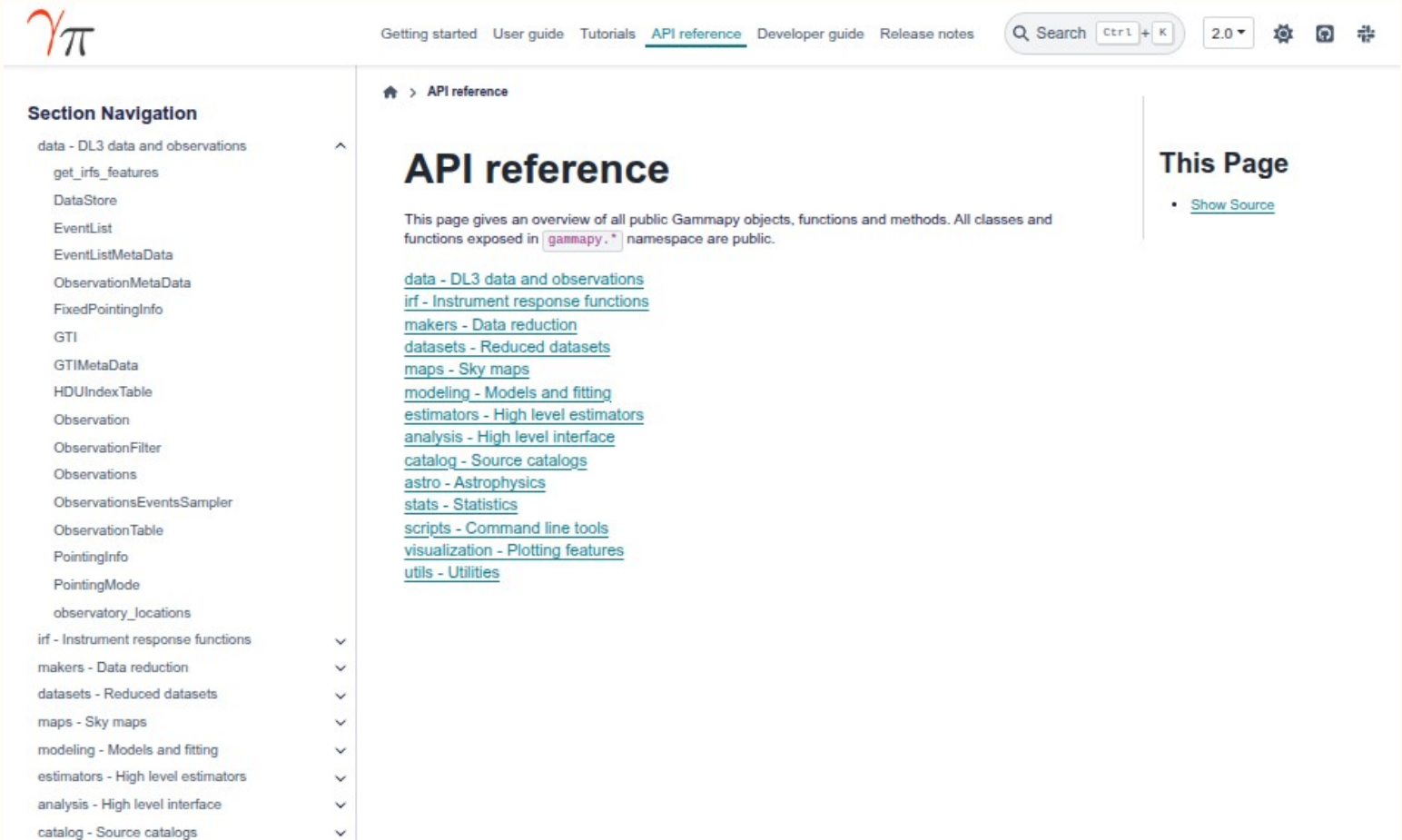
Tutorials


Getting started
User guide
Tutorials
API reference
Developer guide
Release notes
Search
Ctrl + K
2.0
Section Navigation
Introduction
Data exploration
Model Gallery
Detailed explanation
Data analysis
2D Image
3D Cube
Time series
Astrophysics use cases
Scripts
Tutorials
Tutorials
Important
It is strongly advised to first read Fundamental Concepts: Gammapy analysis workflow and package structure of the User Guide before using the tutorials.
In general, all methods and classes are defined with default values that permit a good execution per default. In the tutorials, we frequently use extra values to just illustrate their usage.
The Gammapy library is used by many instruments and as consequence can not describe the specificities of each data release of each observatory. Get in touch with the observatory experts to get the best usage of a given data release.
This page lists the Gammapy 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 Gammapy locally (see Installation) and download the tutorial notebooks and example datasets (see Getting started). Once Gammapy is installed, remember that you can always use gammapy tutorial setup to check your tutorial setup, or in your script with
from gammapy.utils.check import check_tutorials_setup
check_tutorials_setup()
Gammapy 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 Gammapy 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.
On this page
Introduction
Data exploration
Model Gallery
Detailed explanation
Data analysis
1D Spectral
2D Image
3D Cube
Time series
Astrophysics use cases
Scripts
This Page
Show Source

API reference



API reference



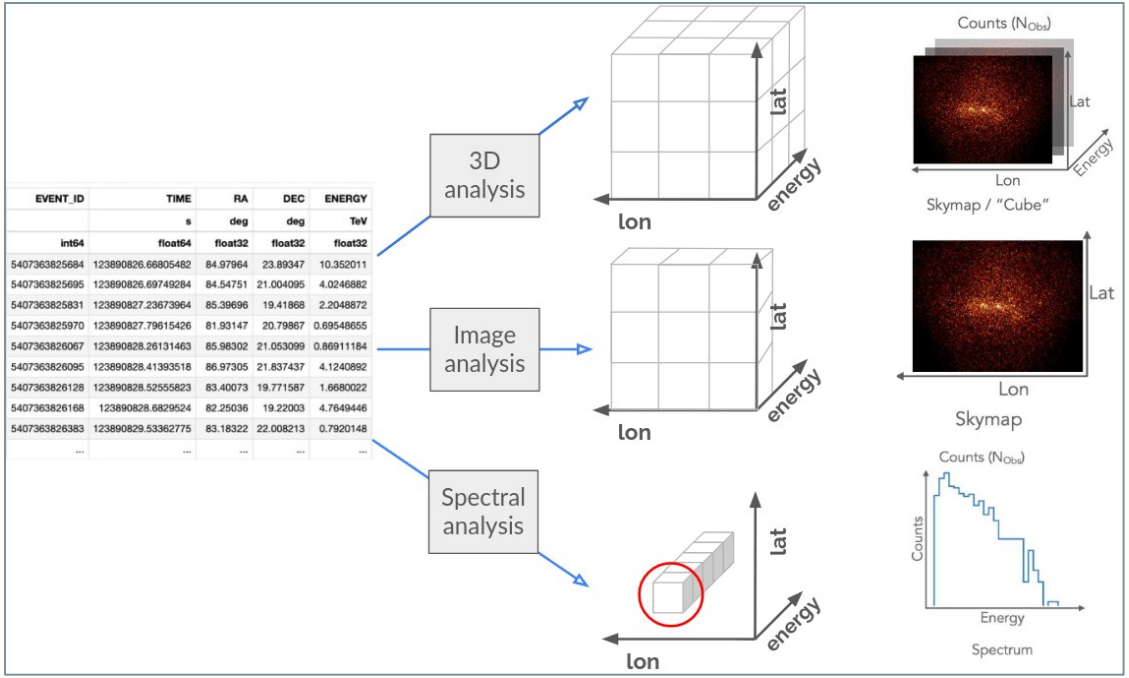
Workflow

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, with an event selection
  - 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

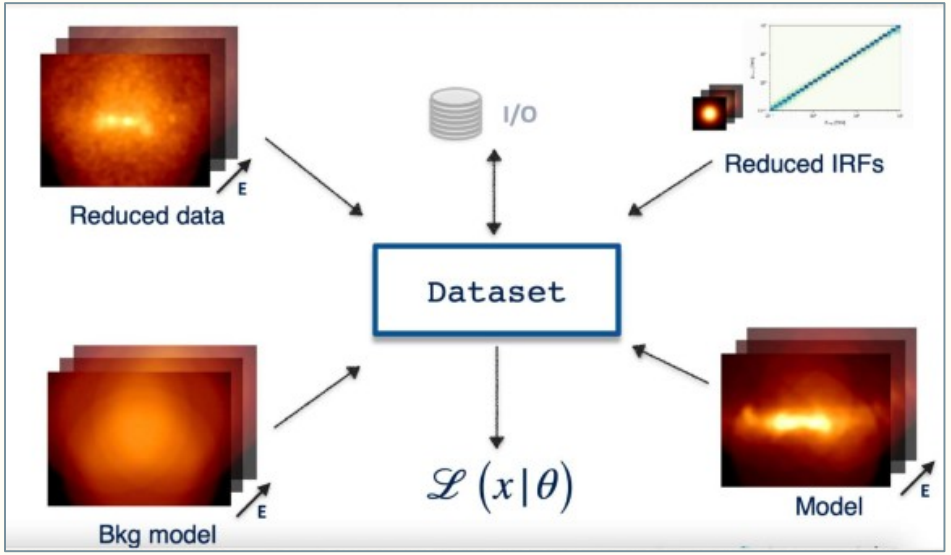
DL4→DL5

- 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"

$$\mathcal{C} = 2 \sum_i N^i_{Pred} - N^i_{Obs} \cdot \log N^i_{Pred}$$

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

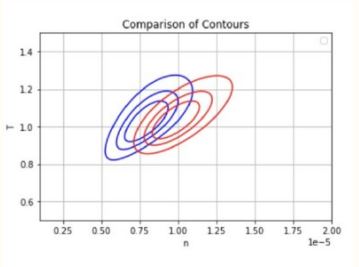


See Statistics in Gammapy
See Fit Statistics

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, BUT biases expected!
- DL6: catalogs
  - Support provide for common catalogs: Fermi 4FGL, H.E.S.S. galactic plane survey, HAWC catalog, etc
  - Create your own catalogs...



Slide credit: Atreyee Sinha

API and sub-packages

gammapy.data

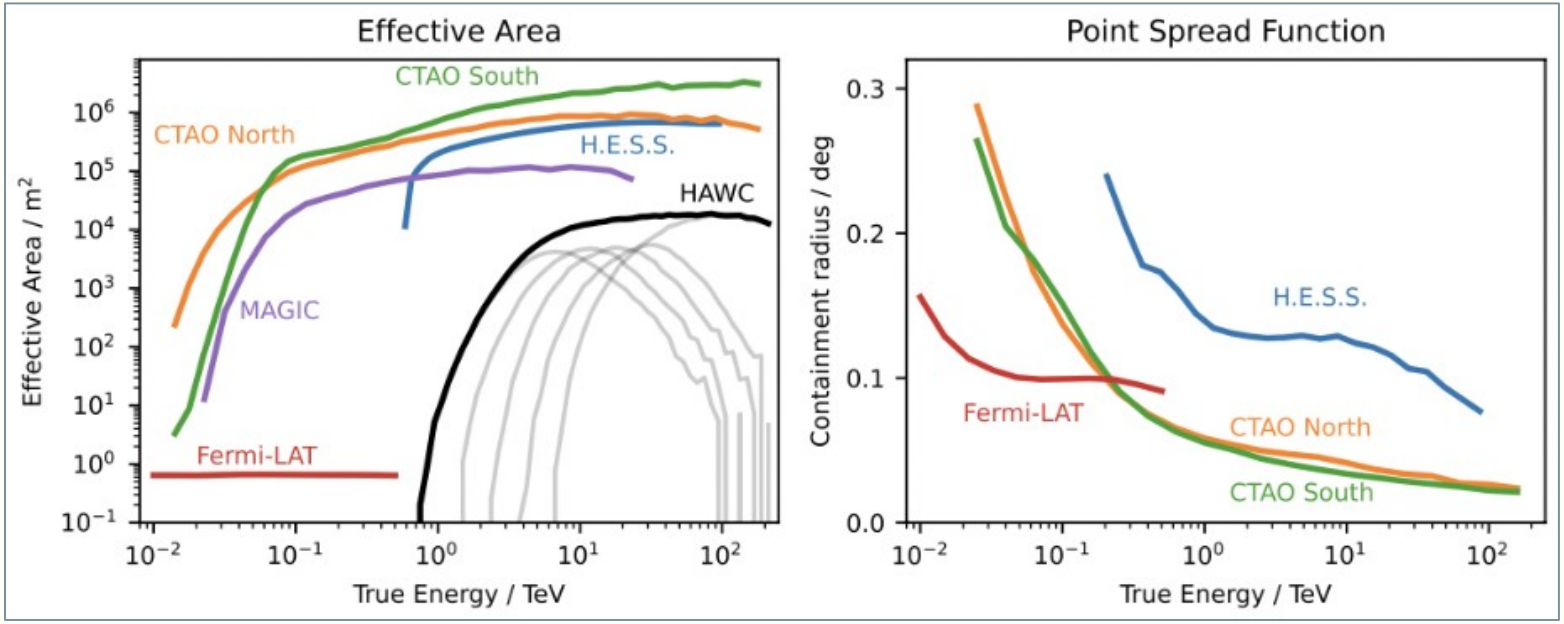
- Select, read and represent DL3 data in memory
- Compliant with GADF
- Observation and EventList
  - Associated IRFs
- hdu-index-table: link event list to associated IRFs
- obs-index-table: selecting observation

```
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
```

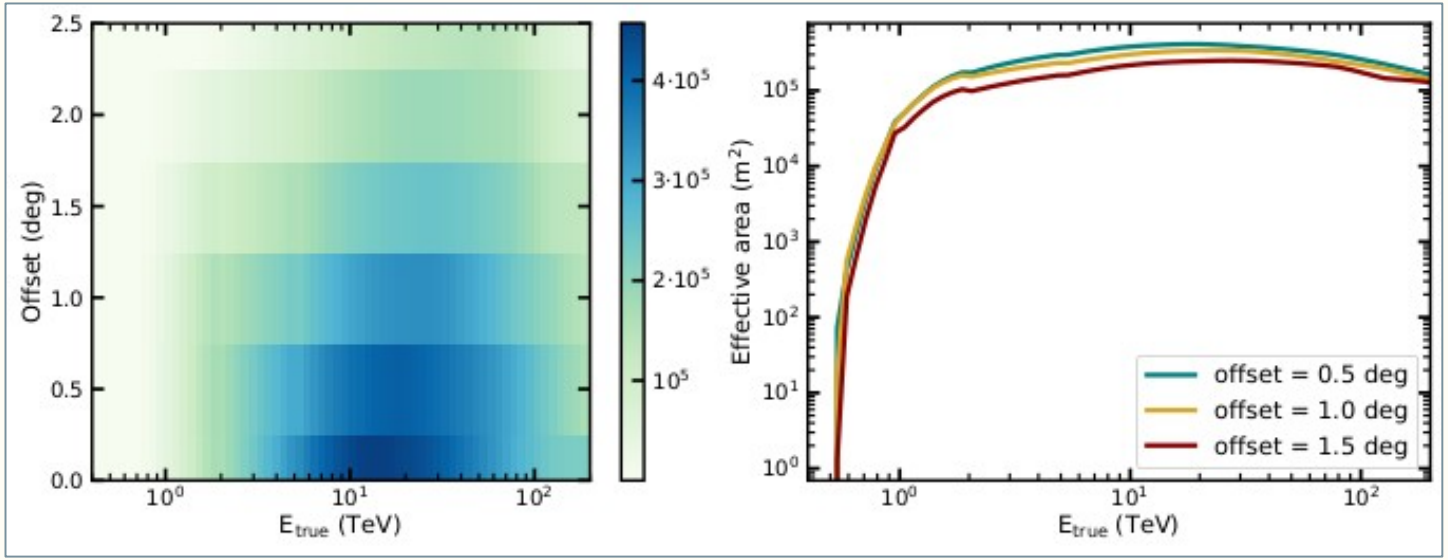
gammapy.irfs

- 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



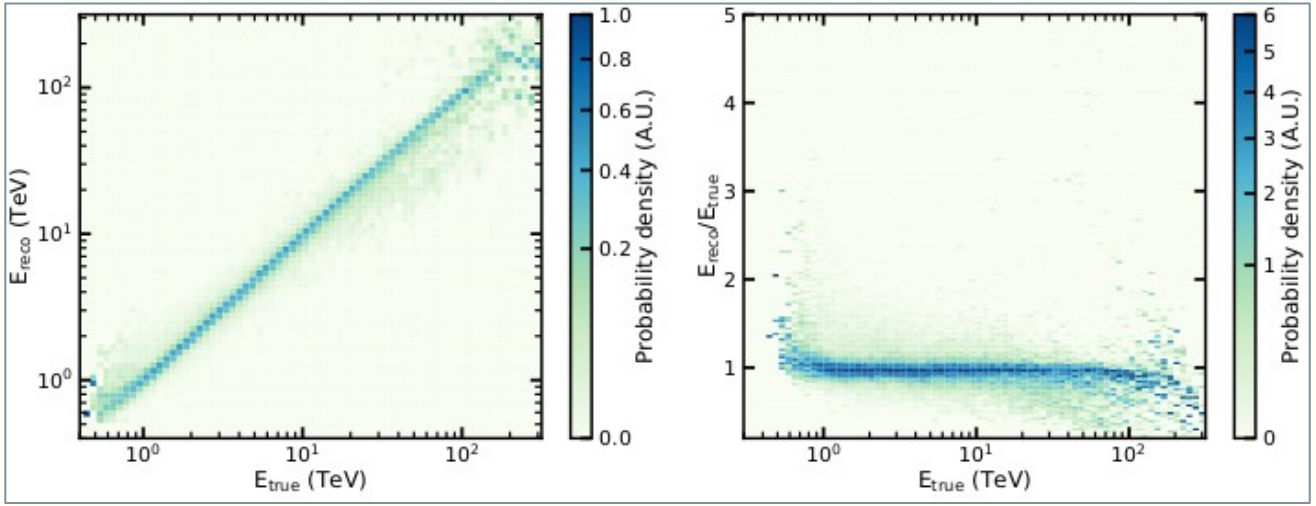
Effective area

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

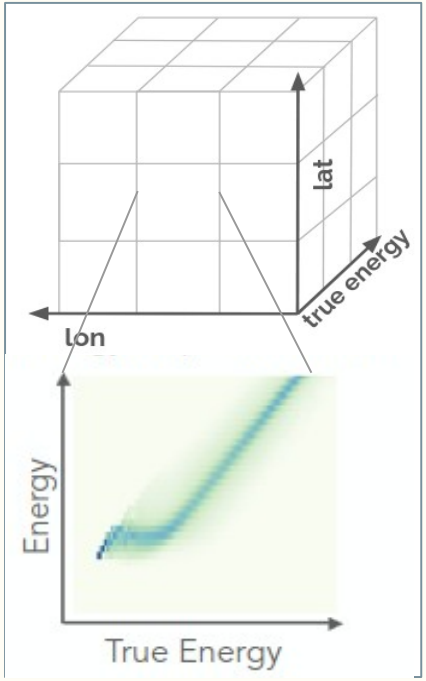


Energy dispersion

- 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)

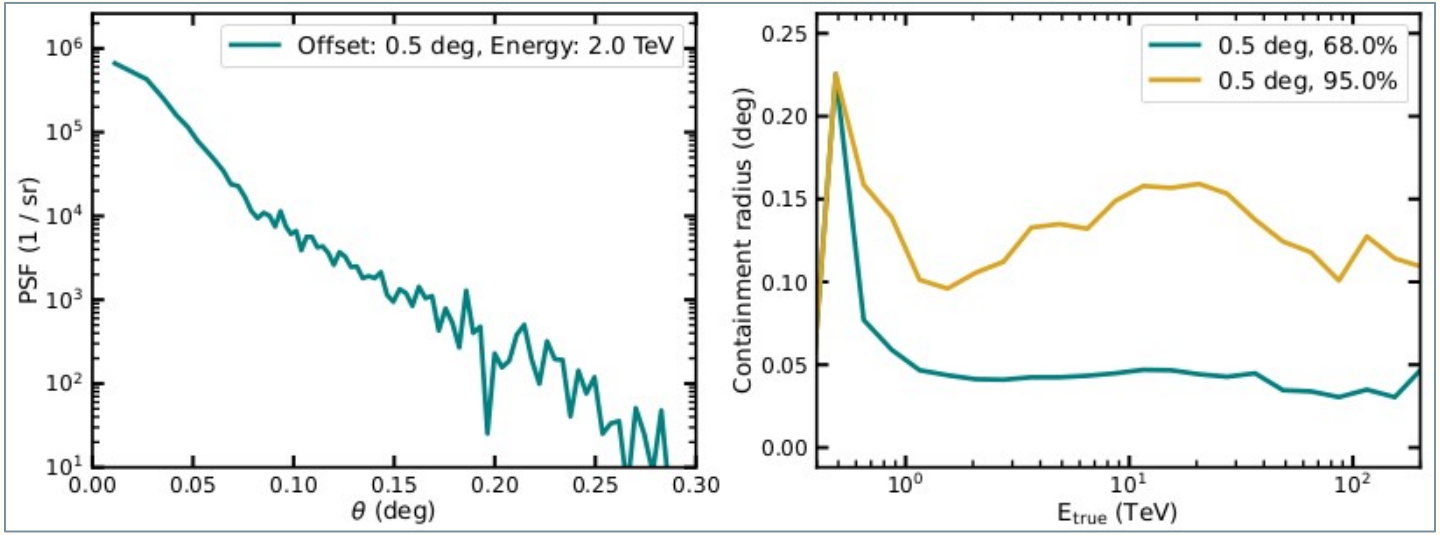


Energy Dispersion



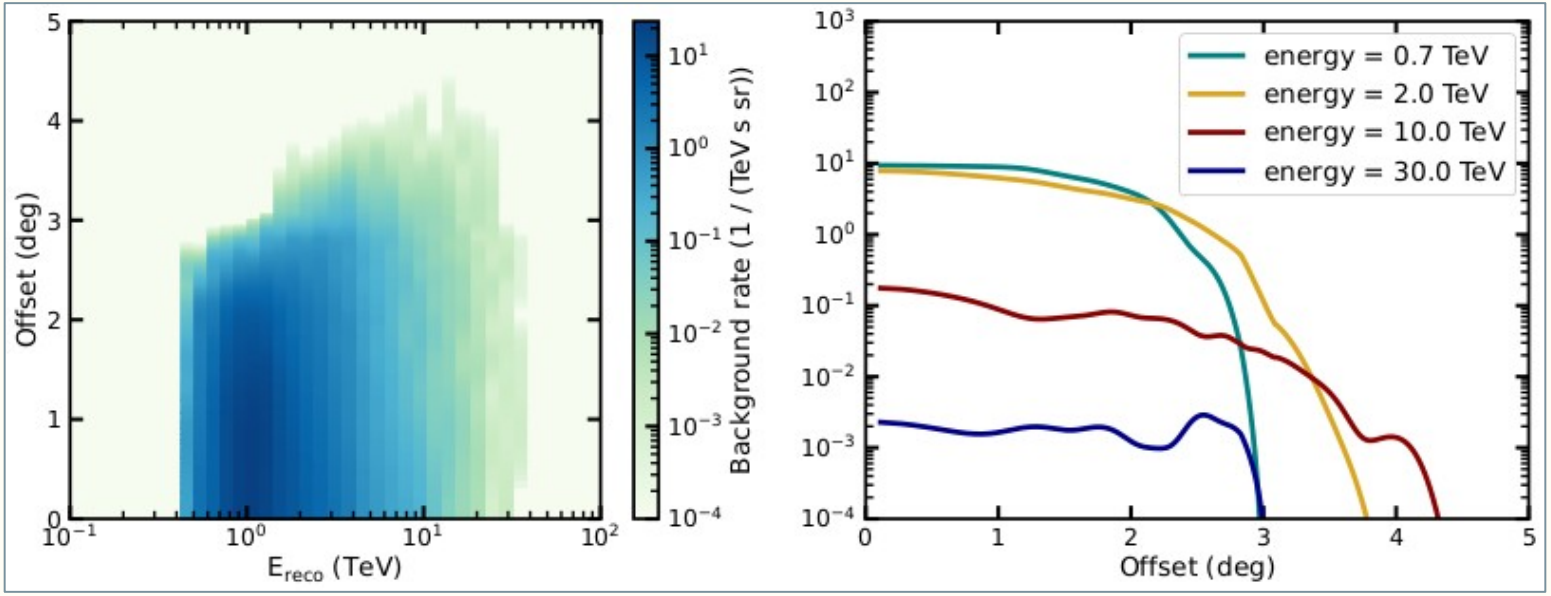
Point Spread Function

- 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

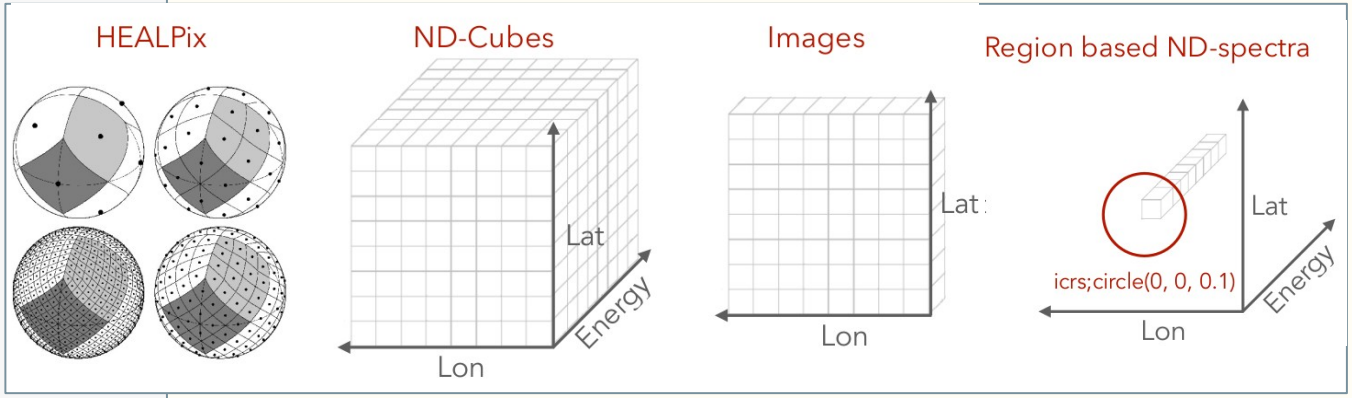
- Represents the expected remaining hadronic background rate after gamma-hadron separation due to misclassified events
- Depends on the reconstructed energy and the offset from the pointing position



gammapy.maps

```python
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



Slide credit: Axel Donath

gammapy.makers

```python
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",
    nbin=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)
```

```python
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 the represent an algorithm/data reduction step
- Data represented by a MapDataset is passed between the makers is modified and holds the state
- This allow 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

Slide credit: Axel Donath

gammapy.makers

MapDatasetMaker, SpectrumDatasetMaker

- 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,

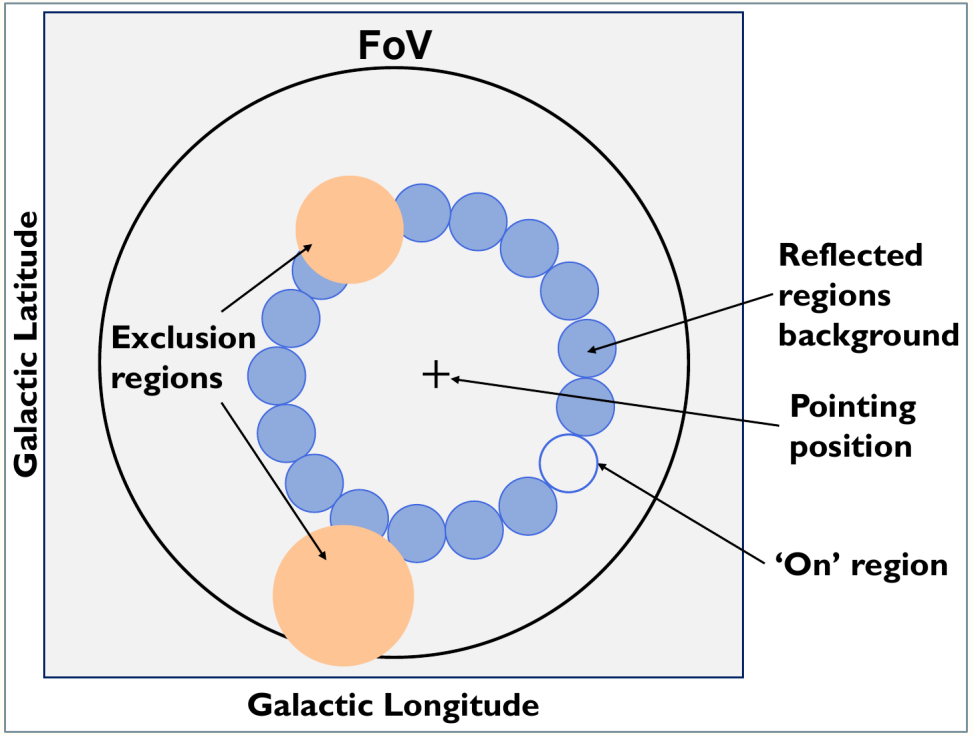
Cash statistics

1D analysis technique

Reflected regions method:

- Assumption: radial symmetry of the effective area/acceptance
- 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

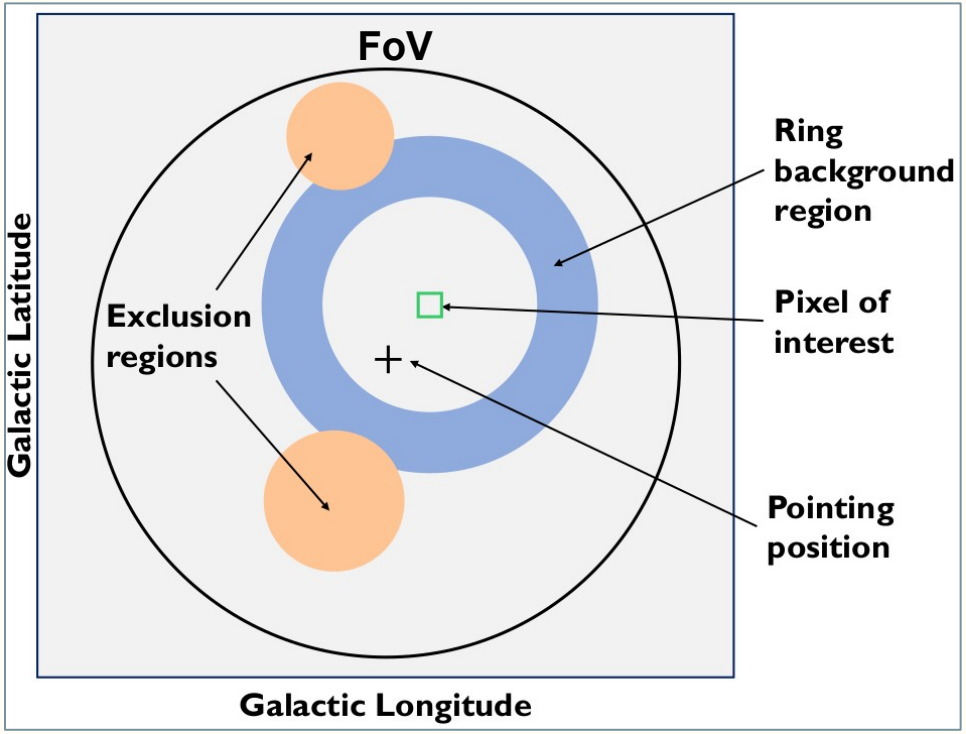
  lost when grouping the pixels inside the ‘on’ region



2D analysis technique

Ring background method:

- Assumption: radial symmetry of the effective area/acceptance
- 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 #1: this relies on a radial background model
- Note #2: estimated Off counts are correlated between bins



3D analysis technique

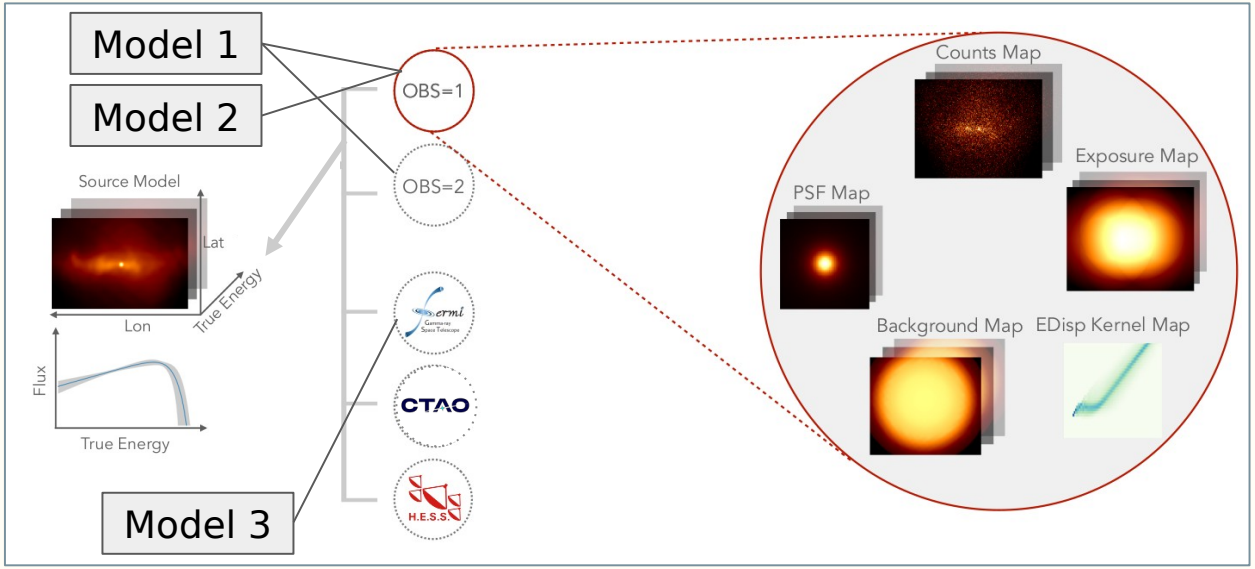
FoV background method:

- 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
- In Gammapy, adjustment of normalisation and tilt → varies the spectral shape for the model

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

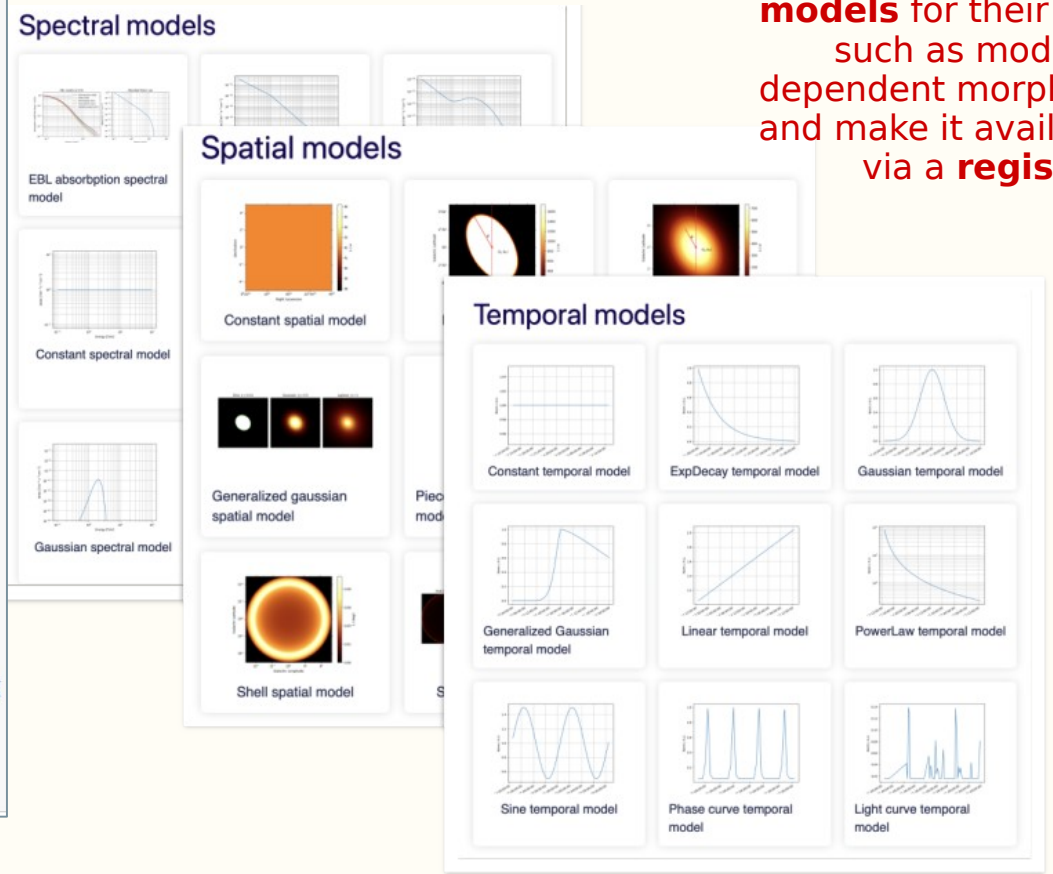
gammapy.datasets

- 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



gammapy.modeling

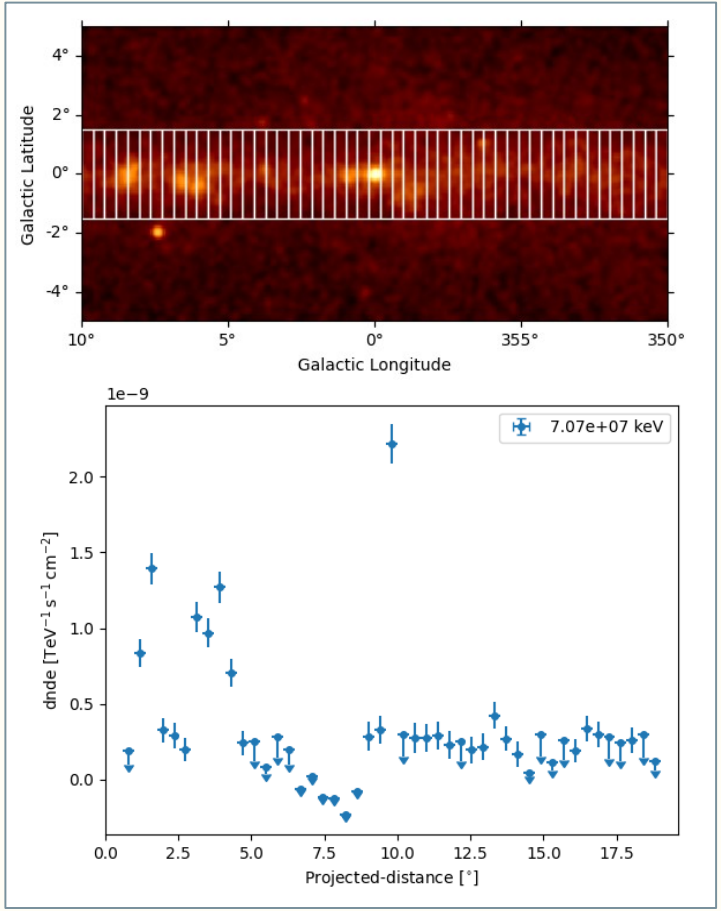
```python
from astropy import units as u
from gammapy.modeling.models import (
    ConstantTemporalModel,
    EBLAbsorptionNormSpectralModel,
    PointSpatialModel,
    PowerLawSpectralModel,
    SkyModel,
)

# 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)
```



Users can implement **custom models** for their specific use-case, such as modelling energy-dependent morphology of a source and make it available to Gammapy 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 best-fit 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

`SourceCatalog1LHAASO`	First LHAASO catalog.
`SourceCatalog2FHL`	Fermi-LAT 2FHL source catalog.
`SourceCatalog2HWC`	HAWC 2HWC catalog.
`SourceCatalog3FGL`	Fermi-LAT 3FGL source catalog.
`SourceCatalog3FHL`	Fermi-LAT 3FHL source catalog.
`SourceCatalog3HWC`	HAWC 3HWC catalog.
`SourceCatalog4FGL`	Fermi-LAT 4FGL source catalog.
`SourceCatalog2PC`	Fermi-LAT 2nd pulsar catalog.
`SourceCatalog3PC`	Fermi-LAT 3rd pulsar catalog.
`SourceCatalogGammaCat`	Gammacat open TeV source catalog.
`SourceCatalogHGPS`	HESS Galactic plane survey (HGPS) source catalog.
`SourceCatalogLargeScaleHGPS`	Gaussian band model.
`SourceCatalogObject`	Source catalog object.
`SourceCatalogObject1LHAASO`	One source from the 1LHAASO catalog.
`SourceCatalogObject2FHL`	One source from the Fermi-LAT 2FHL catalog.
`SourceCatalogObject2HWC`	One source from the HAWC 2HWC catalog.
`SourceCatalogObject3FGL`	One source from the Fermi-LAT 3FGL catalog.
`SourceCatalogObject3FHL`	One source from the Fermi-LAT 3FHL catalog.
`SourceCatalogObject3HWC`	One source from the HAWC 3HWC catalog.
`SourceCatalogObject4FGL`	One source from the Fermi-LAT 4FGL catalog.
`SourceCatalogObject2PC`	One source from the 2PC catalog.
`SourceCatalogObject3PC`	One source from the 3PC catalog.
`SourceCatalogObjectGammaCat`	One object from the gamma-cat source catalog.
`SourceCatalogObjectHGPS`	One object from the HGPS catalog.
`SourceCatalogObjectHGPSComponent`	One Gaussian component from the HGPS catalog.

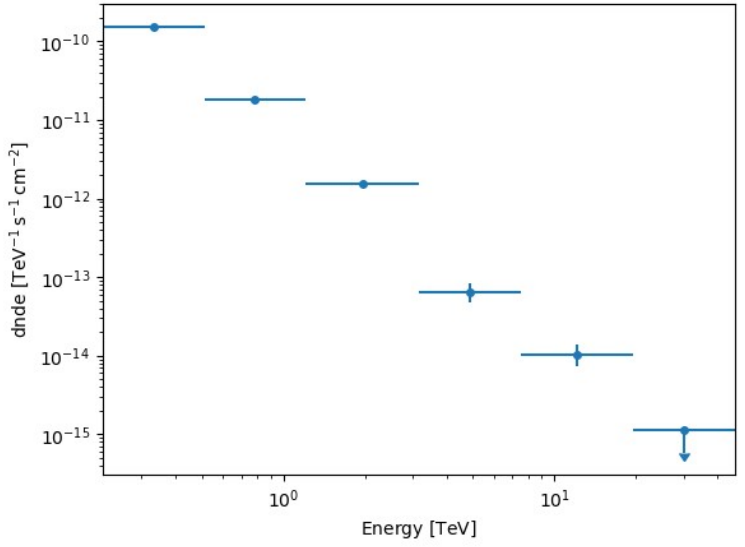
gammapy.catalog

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

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

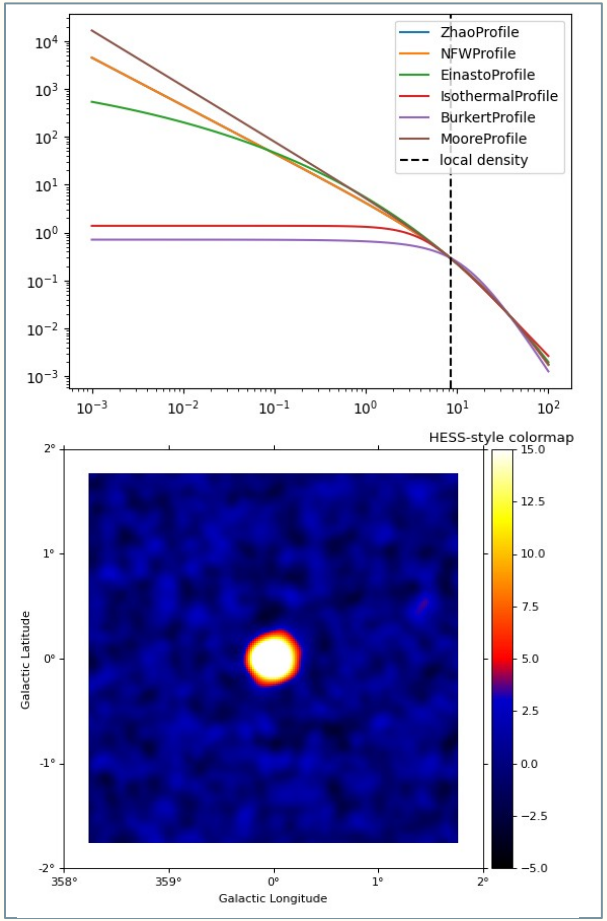
```
78
```

```
<Axes: xlabel='Energy [$\\mathrm{TeV}$]', ylabel='dnde [$\\mathrm{TeV^{-1}\\,s^{-1}\\,cm^{-2}}$]'>
```



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

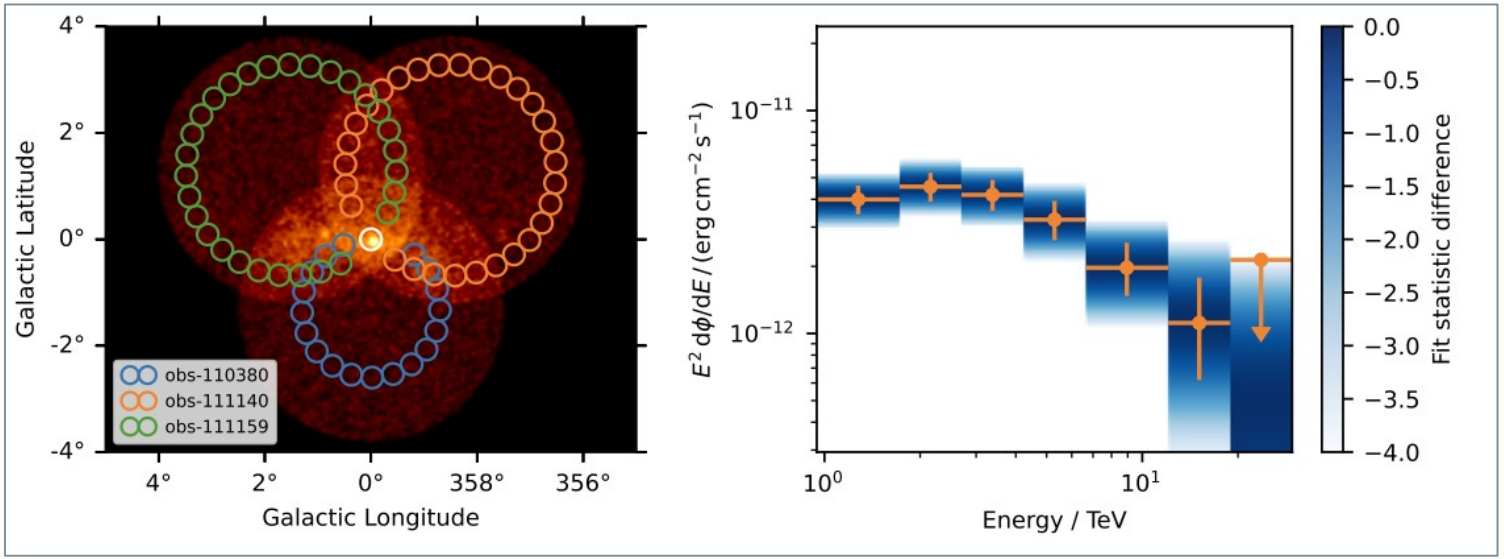


Analysis examples

Spectral Analysis

Classical Method

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



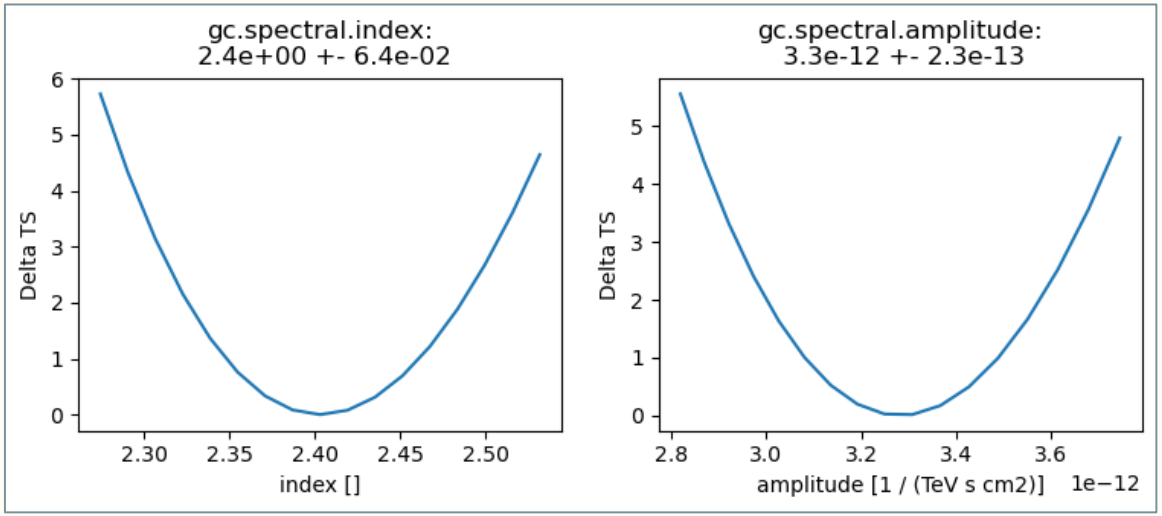
Spectral Analysis

Classical Method

- Maximum likelihood = minimum test statistic

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

- Is the minimum well defined? **YES!**



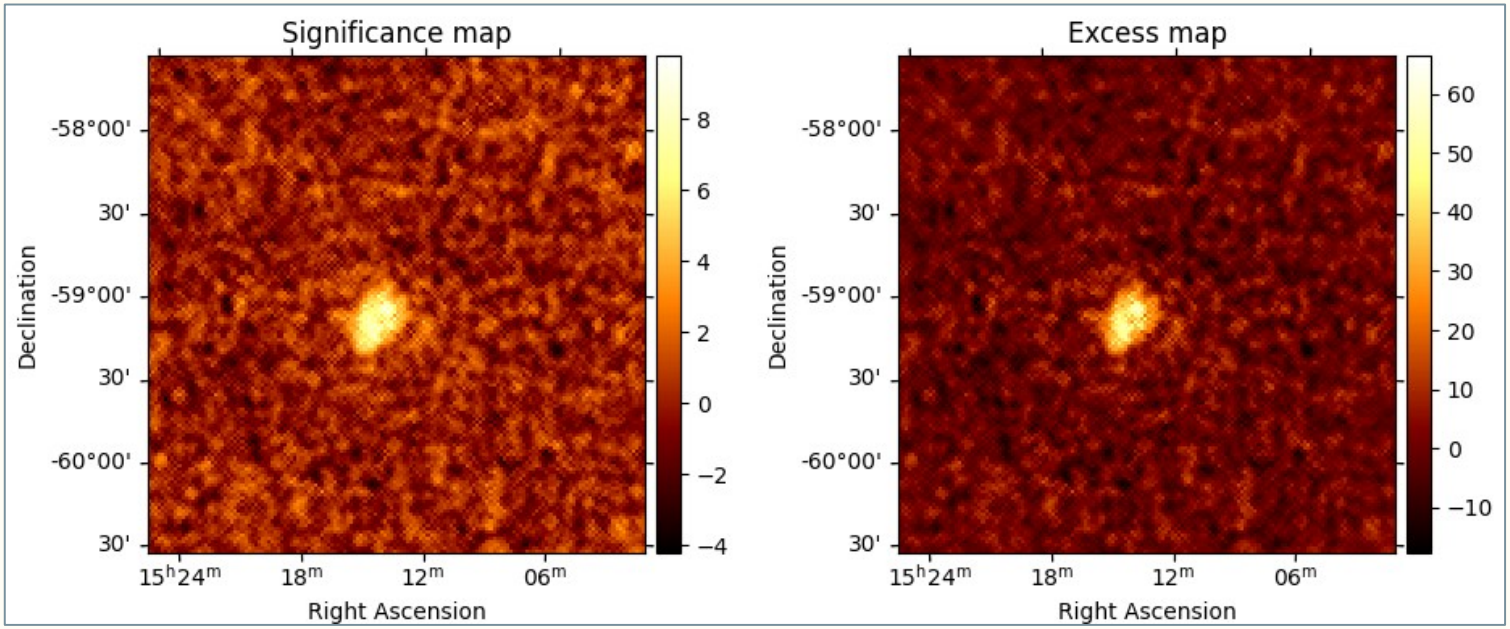
See this tutorial

- 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

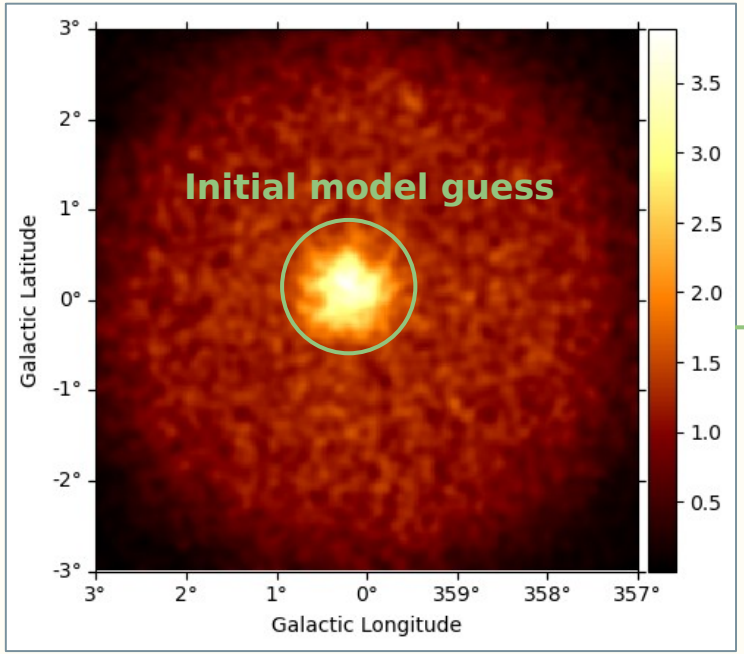
- Utilise the ring background method as explained previously
- Select OFF events from an annulus about the ON region
- Requires good knowledge of 2D acceptance!



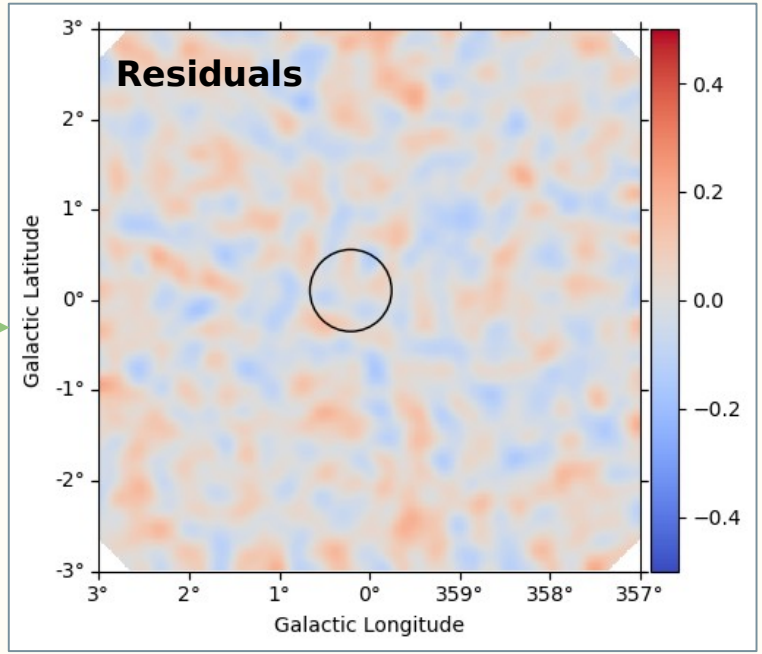
Spatial and Spectral Analysis

The new 3D Method

- 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



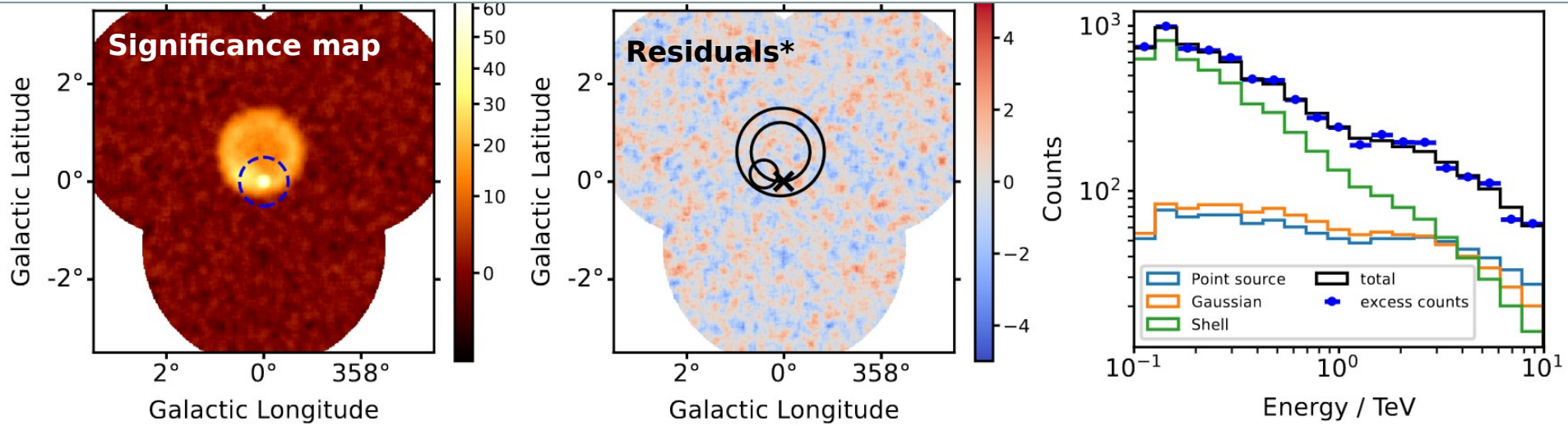
Fit both the spectral and spatial model



Power of the 3D method

It is possible to disentangle contributions of overlapping sources!

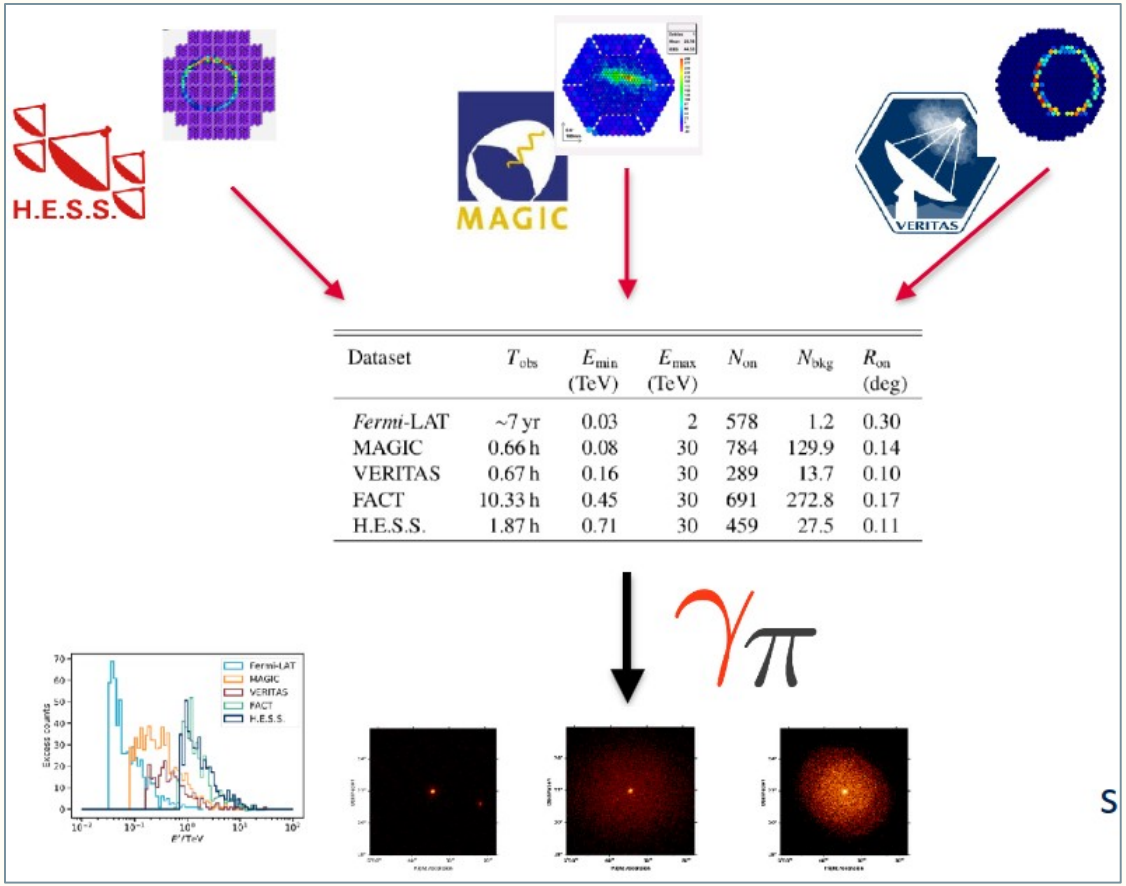
- For example:
  - Point source with power law spectrum
  - Gaussian source with log-parabola spectrum
  - Shell with power law spectrum



*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

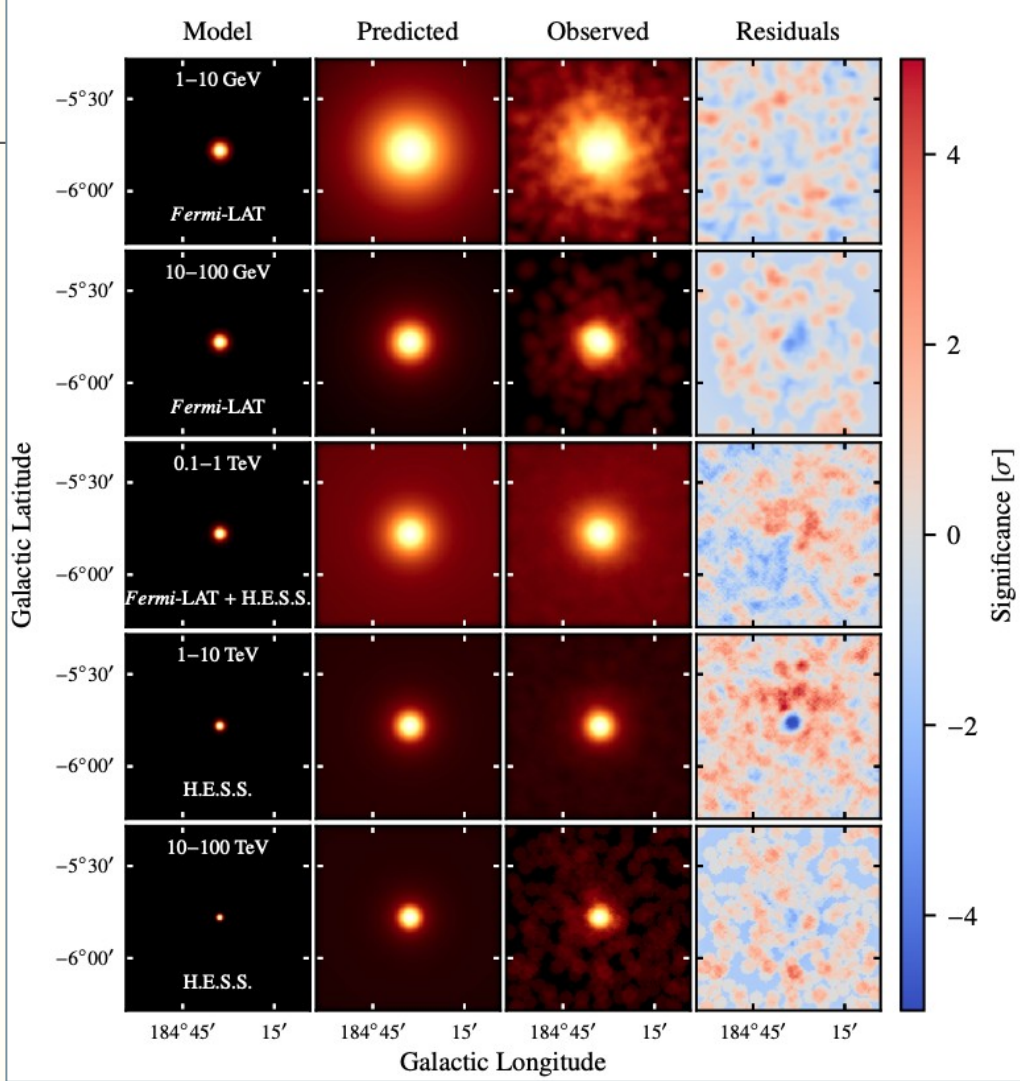
↓

High level/Advanced science products

Joint analysis

Constrain extensions

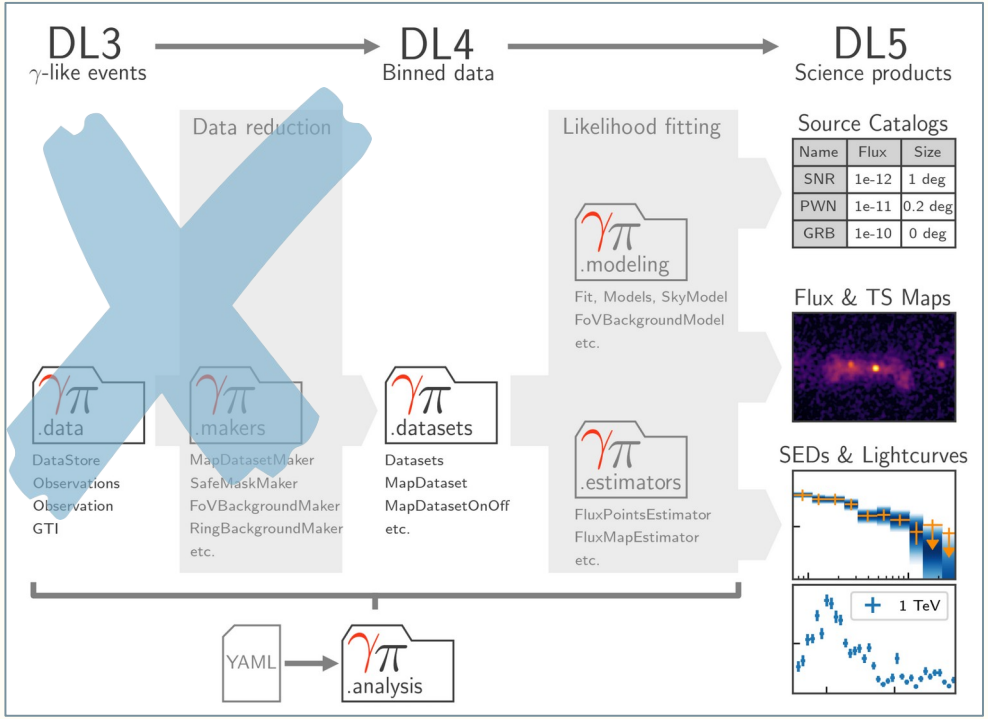
- Joint Fermi-LAT/H.E.S.S. analysis used to constrain the extension of the Crab Nebula
- Probe structures to understand the underlying mechanisms



Aharonian et al., 2024

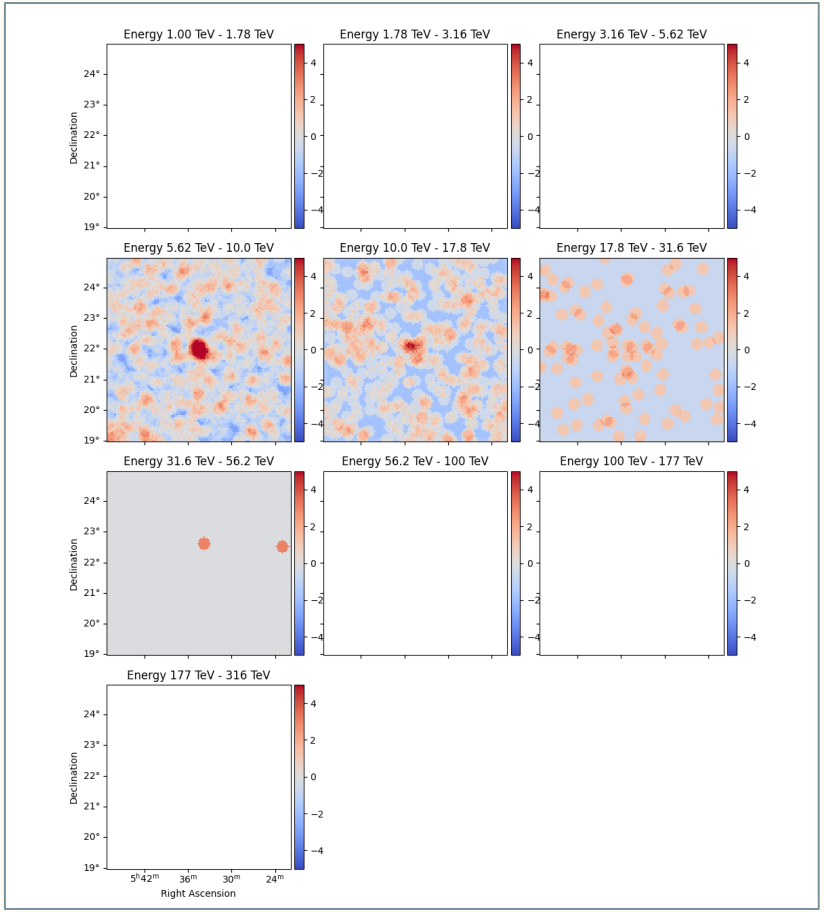
Fermi-LAT with Gammapy

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



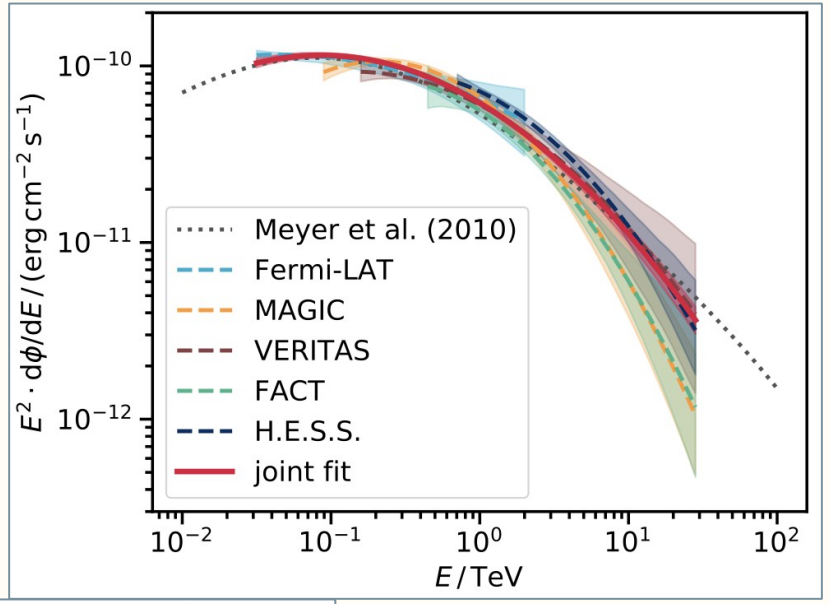
HAWC with Gammapy

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

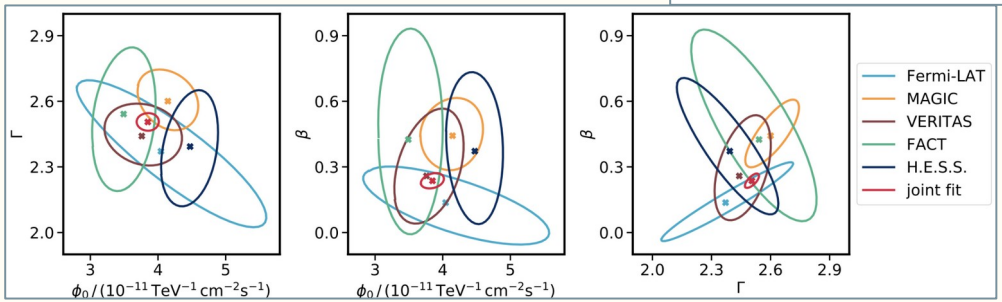


Multi-instrument analysis

- Spectral fit combining different types of data
  - Fermi-LAT DL4 data - full 3D analysis (7yrs 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

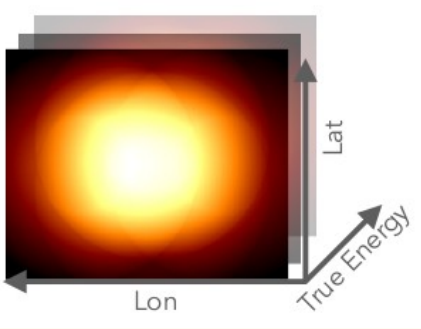
Thanks for your attention

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

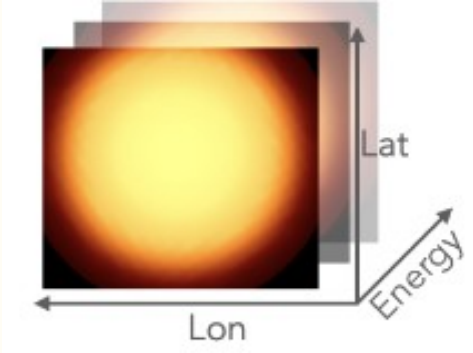
gammapy.irfs

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

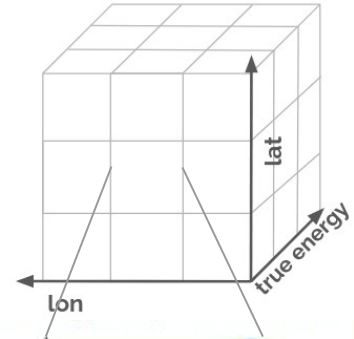
Exposure



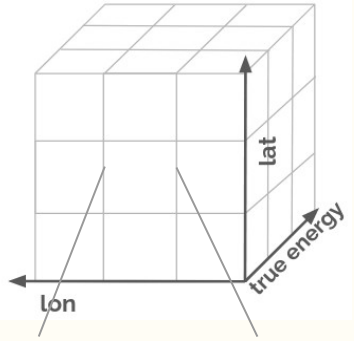
Background template



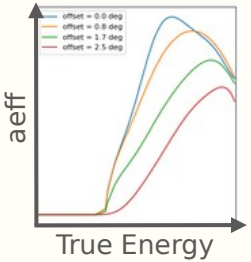
Energy Dispersion



PSF



Effective Area



Maximum likelihood

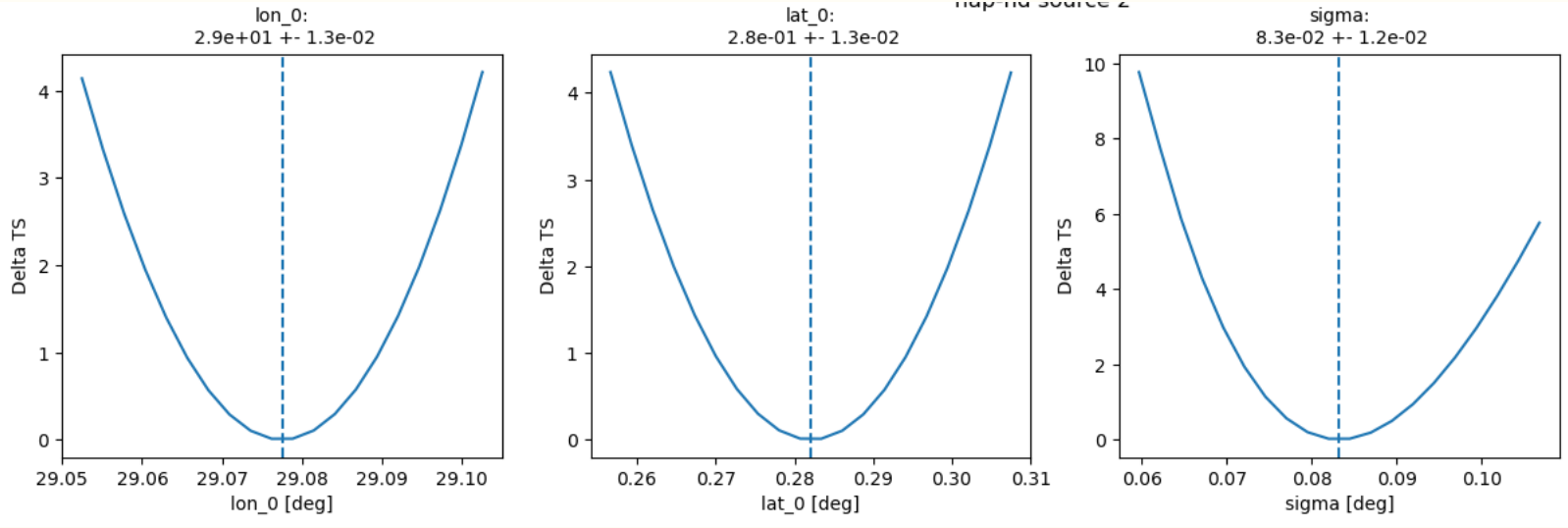
- We define the likelihood function for Poisson probability density functions

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

- Assume a model of family of models and compute the predicted counts for that set of models with their associated parameters
- Utilise the maximum likelihood technique to determine the best fit parameters and their associated uncertainties
  - The likelihood is different for whether you know the background or measure it on the background, i.e you utilise two different statistics but they are just two different ways of writing the likelihood

Maximum likelihood

- Another way of saying that is, we find the minimum of the log-likelihood function
- For example, here the source was fit with a Gaussian model with longitude, latitude and sigma free
- By plotting the scan over the parameters, we can see that it is well minimised



GADF

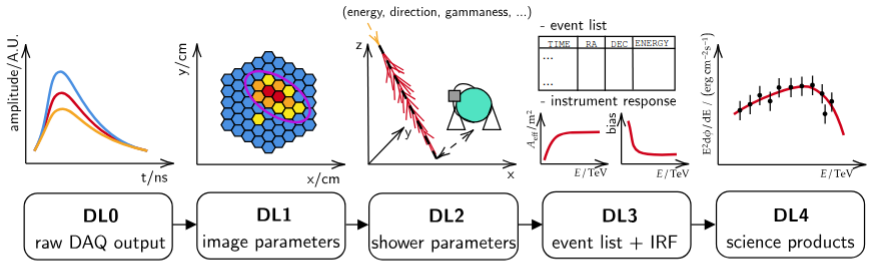


Figure 1. Schematisation of the progressive data reduction and data levels of an IACT. Raw data contain the signal sampled from the photomultipliers at the occurrence of a trigger event (Data Level 0). Calibrated data (Data Level 1) contain the pixelated image of the Cherenkov light of the shower. The latter can be parametrised with few geometrical quantities and used to determine the observables of the original shower, including its probability of being a gamma-ray shower (Data Level 2). The detected events can be gathered in a list of gamma ray candidates, together with the functions representing the response of the system (the so-called instrument response function, IRF), e.g., the collection area of the system as a function of the energy or the bias of its energy reconstruction (Data Level 3). This information can be used to perform a statistical analysis obtaining the so-called science products, in this case the spectrum of the source (Data Level 4).

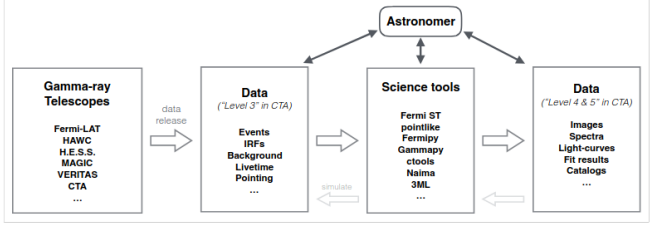


FIGURE 1. The purpose of the `gamma-astro-data-formats` effort is to encourage collaboration between high-level gamma-ray data producers, science tool developers, and data analysts. The goal is to develop common data formats to avoid duplication of efforts and confusion by astronomers working with multi-mission gamma-ray data or multiple analysis tools.

Introduction

The Flexible Image Transport System (FITS) format was created around 1980 [1] by optical astronomers. In the 1990s, the HEASARC FITS Working Group, also known as the OGIP (Office of Guest Investigator Programs) FITS Working Group, produced documents and recommendations concerning the storage of X-ray (and partly gamma-ray space telescope) data in FITS.[1] Several of these recommendations have subsequently been incorporated into the FITS standard, the latest version is FITS 3.0 from 2010 [2].

Today, very-high energy (VHE, energy > 50 GeV) gamma-ray astronomy is finding itself in a similar situation

- Gamma Astro Data Format (GADF)
- Based on the Fermi-LAT format proposed in 2016
- Adopted by CTAO
- H.E.S.S. DL3 DR1
- MAGIC data release

https://arxiv.org/pdf/1610.01884