UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Novos Arranjos de Pesquisa e Inovação
FENÔMENOS EXTREMOS DO
UNIVERSO

γπ A Python package for gamma-ray astronomy

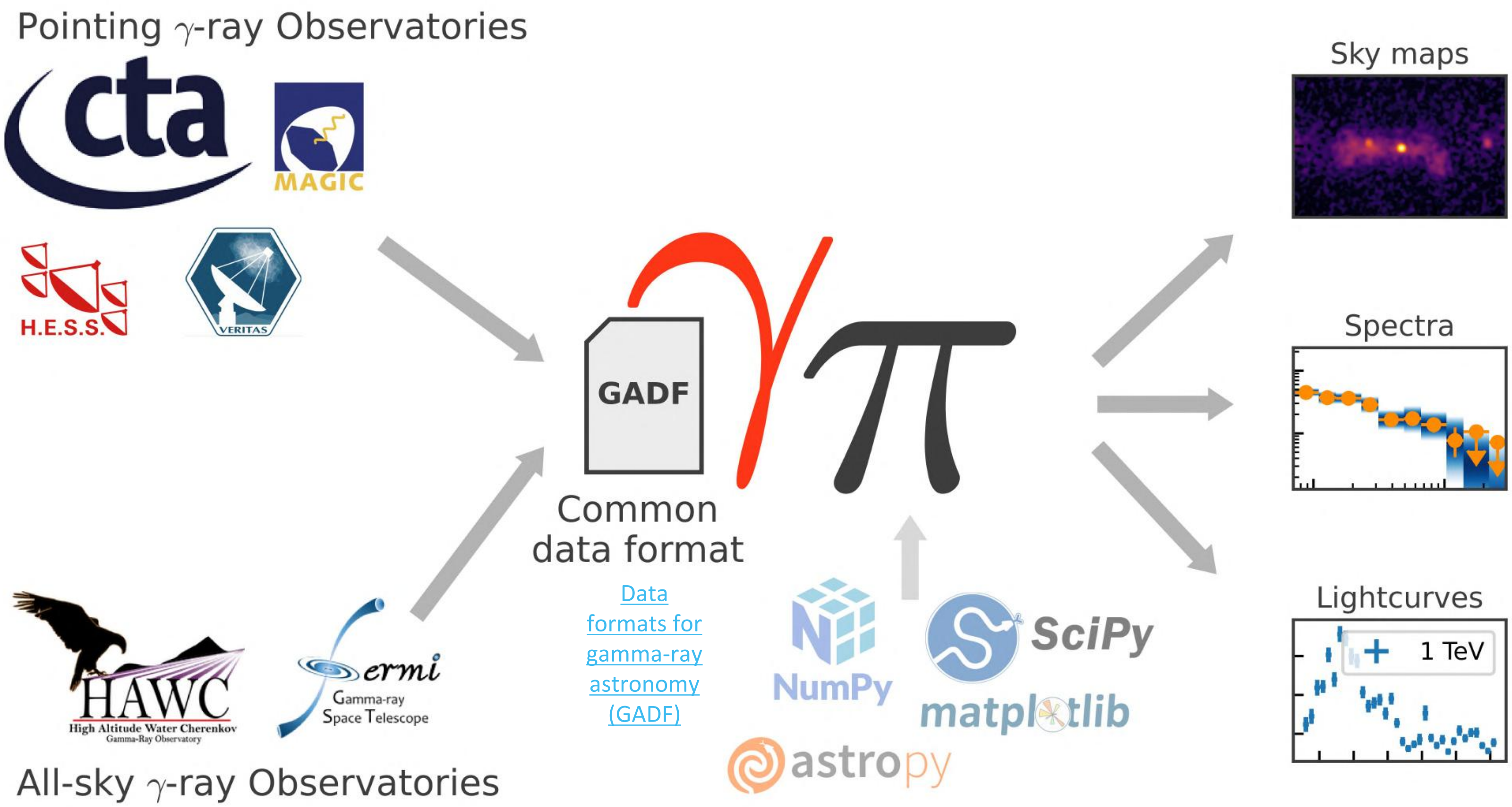
Introduction to Gammapy

M. Felipe Sousa
Rubens Costa Jr.

Palotina, PR, March 5th, 2023

What is Gammapy?

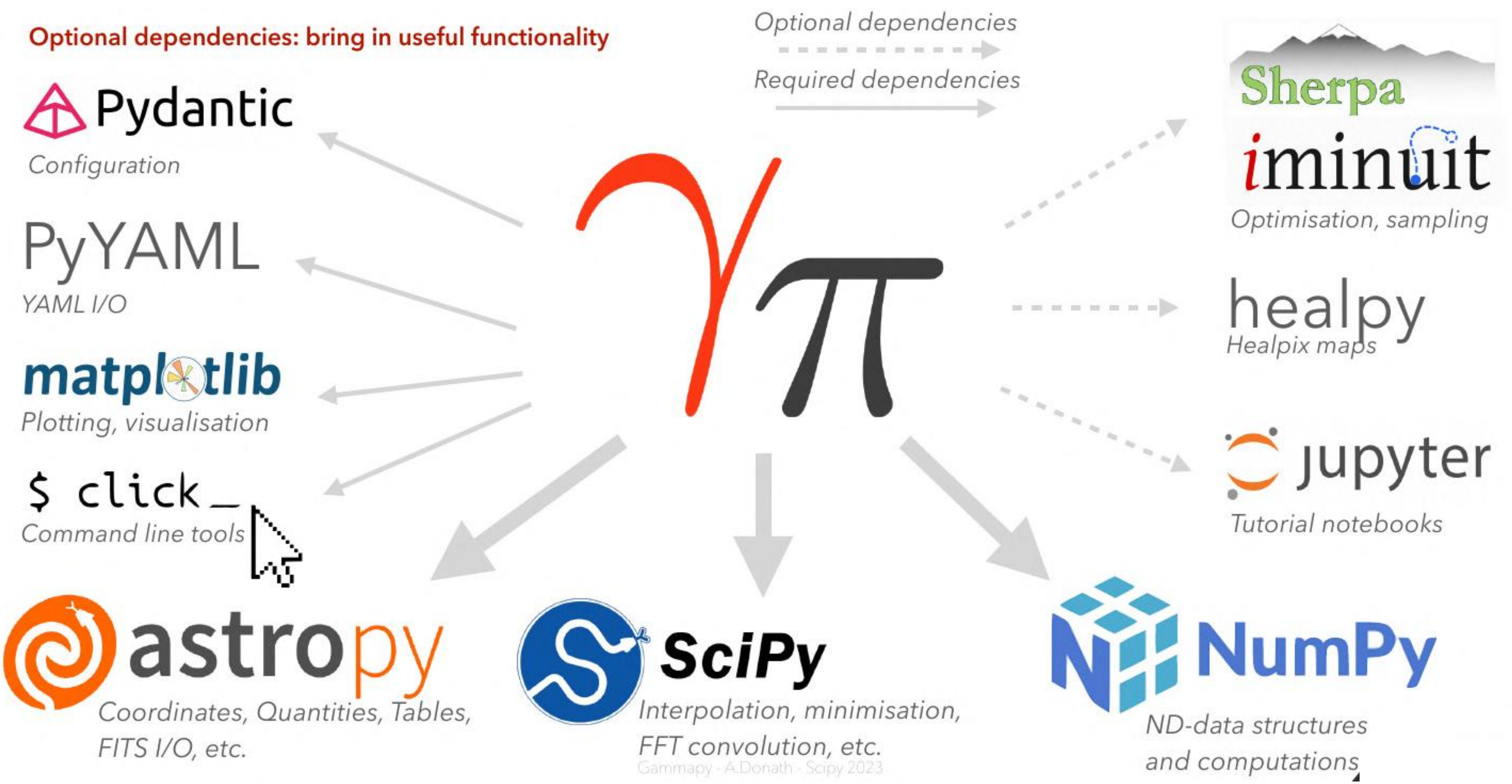
- Gammapy is an open-source Python package for gamma-ray astronomy;
- It is used as core library for the Science Analysis tools of the Cherenkov Telescope Array (CTA);
- It is already widely used in the analysis of existing gamma-ray instruments, such as H.E.S.S. MAGIC, VERITAS and HAWC.



Data formats for gamma-ray astronomy (GADF)

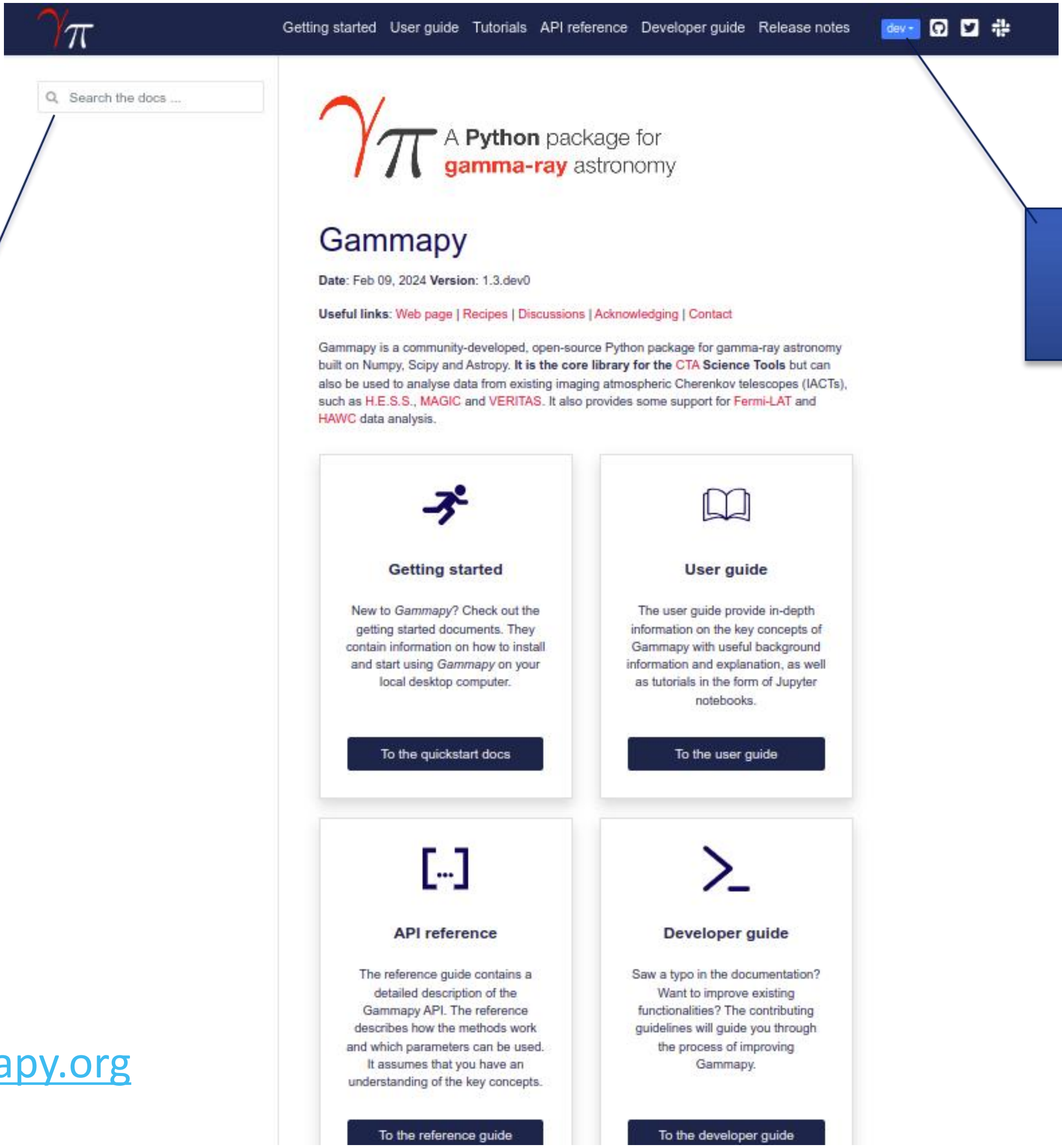
Donath, A., et al.(2023)

Gammapy dependencies



Donath, A., et al.(2023)

Getting started: documentation



A **Python** package for **gamma-ray** astronomy

Gammapy

Date: Feb 09, 2024 **Version**: 1.3.dev0

Useful links: Web page | Recipes | Discussions | Acknowledging | Contact

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

To the quickstart docs

User guide

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

To the user guide

API reference

The reference guide contains a detailed description of the Gammapy API. The reference describes how the methods work and which parameters can be used. It assumes that you have an understanding of the key concepts.

To the reference guide

Developer guide

Saw a typo in the documentation? Want to improve existing functionalities? The contributing guidelines will guide you through the process of improving Gammapy.

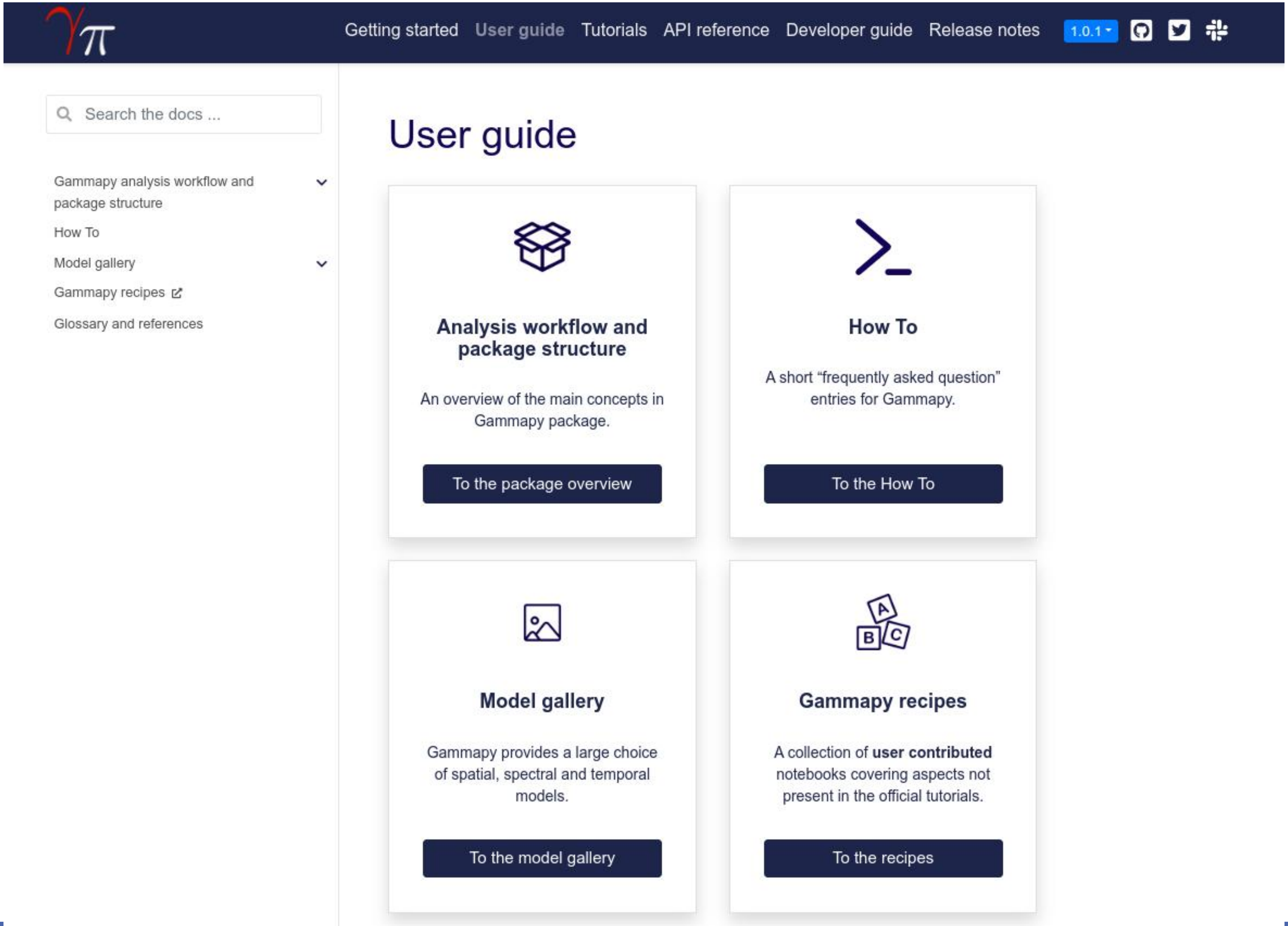
To the developer guide

Search bar

Slide between versions

See docs.gammapy.org

Getting started: documentation



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 Gammapy.

To the How To

Model gallery

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

To the model gallery

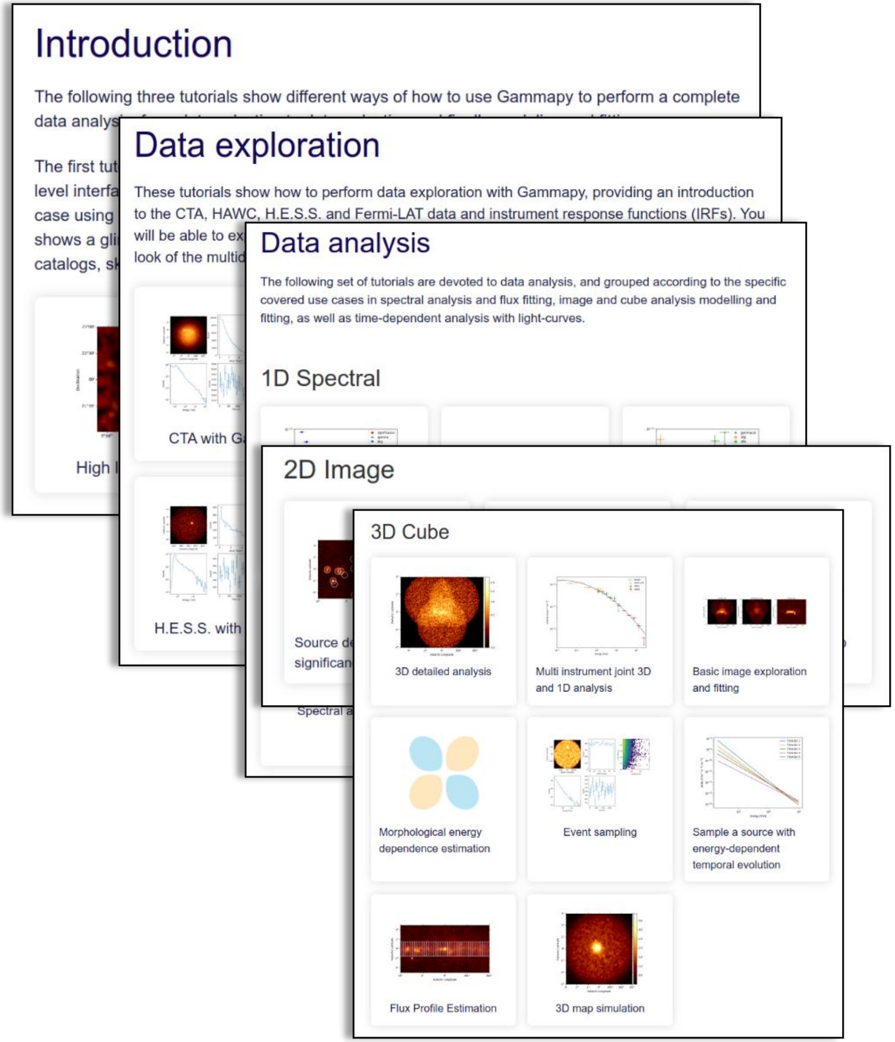
Gammapy recipes

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

To the recipes

Getting started: documentation

- **Learning with examples: the Tutorials**
- **More in depth: the API description**



Introduction

The following three tutorials show different ways of how to use Gammapy to perform a complete data analys

The first tut
level interfa
case using
shows a gli
catalogs, sk

High l

Data exploration

These tutorials show how to perform data exploration with Gammapy, providing an introduction to the CTA, HAWC, H.E.S.S. and Fermi-LAT data and instrument response functions (IRFs). You will be able to ex
look of the multid

CTA with G

H.E.S.S. with

Data analysis

The following set of tutorials are devoted to data analysis, and grouped according to the specific covered use cases in spectral analysis and flux fitting, image and cube analysis modelling and fitting, as well as time-dependent analysis with light-curves.

1D Spectral

Spectral a

2D Image

Source de
significan

3D Cube

3D detailed analysis

Multi instrument joint 3D and 1D analysis

Basic image exploration and fitting

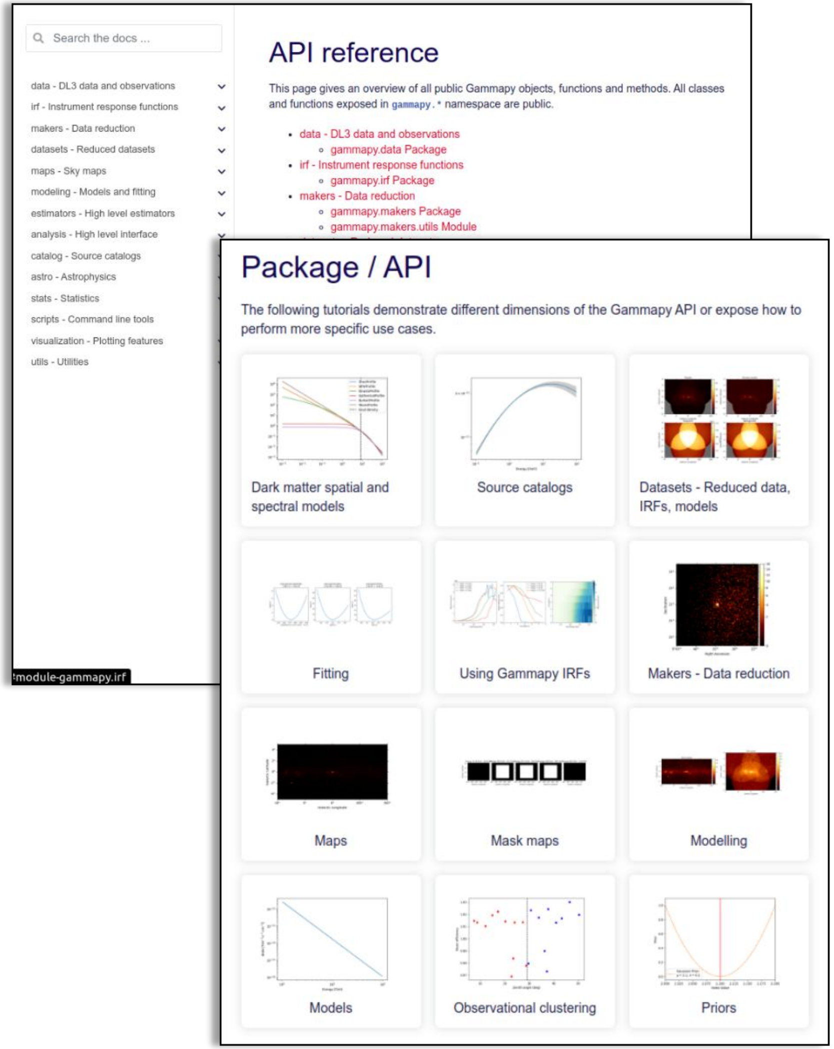
Morphological energy dependence estimation

Event sampling

Sample a source with energy-dependent temporal evolution

Flux Profile Estimation

3D map simulation



Search the docs ...
data - DL3 data and observations
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
:module-gammapy.irf

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

Package / API

The following tutorials demonstrate different dimensions of the Gammapy API or expose how to perform more specific use cases.

Dark matter spatial and spectral models

Source catalogs

Datasets - Reduced data, IRFs, models

Fitting

Using Gammapy IRFs

Makers - Data reduction

Maps

Mask maps

Modelling

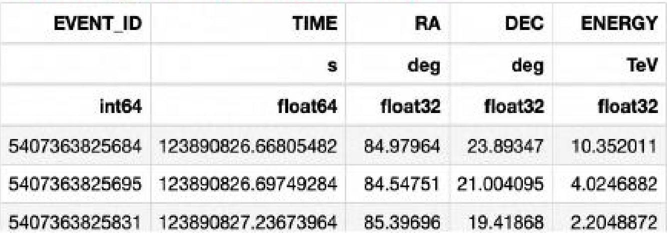
Models

Observational clustering

Priors

Gammapy: Gamma-ray Data Analysis

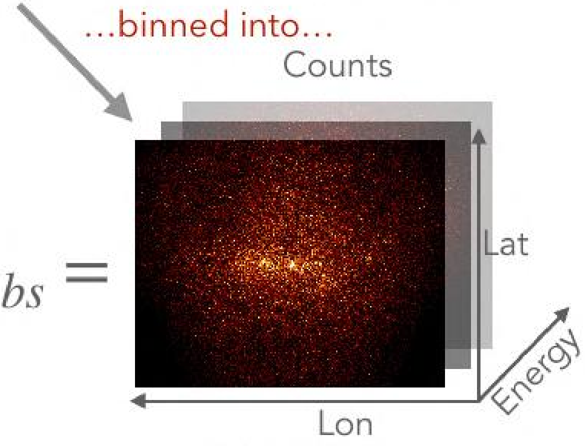
List of gamma-like events…



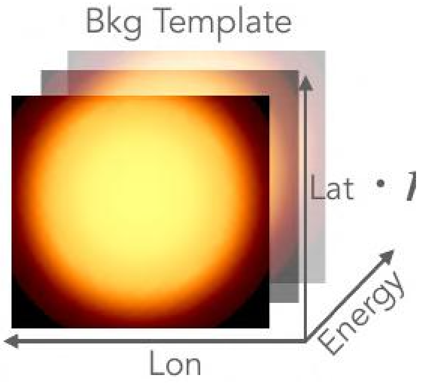
EVENT_ID	TIME	RA	DEC	ENERGY
	s	deg	deg	TeV
int64	float64	float32	float32	float32
5407363825684	123890826.66805482	84.97964	23.89347	10.352011
5407363825695	123890826.69749284	84.54751	21.004095	4.0246882
5407363825831	123890827.23673964	85.39696	19.41868	2.2048872

…binned into…

$$N_{Obs} =$$



$$N_{Bkg} =$$



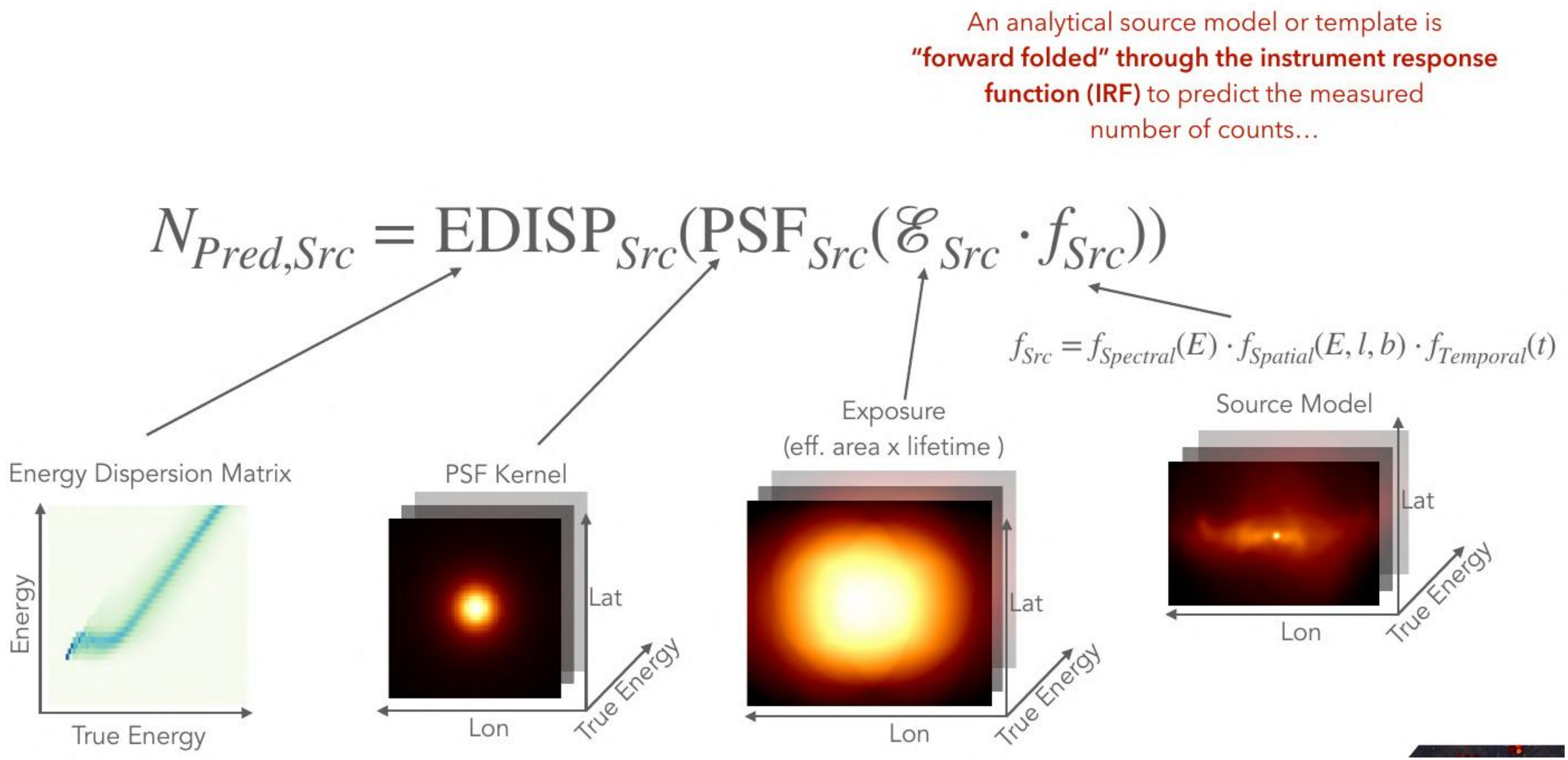
"Cash statistics": summed over all "bins"

$$\mathscr{C} = 2\sum_{i} N^{i}_{Pred} - N^{i}_{Obs} \cdot \log N^{i}_{Pred}$$

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

Slide from A. Donath (Presentation made during the Scipy 2023 conference)

Gammapy: Gamma-ray Data Analysis



An analytical source model or template is **"forward folded" through the instrument response function (IRF)** to predict the measured number of counts…

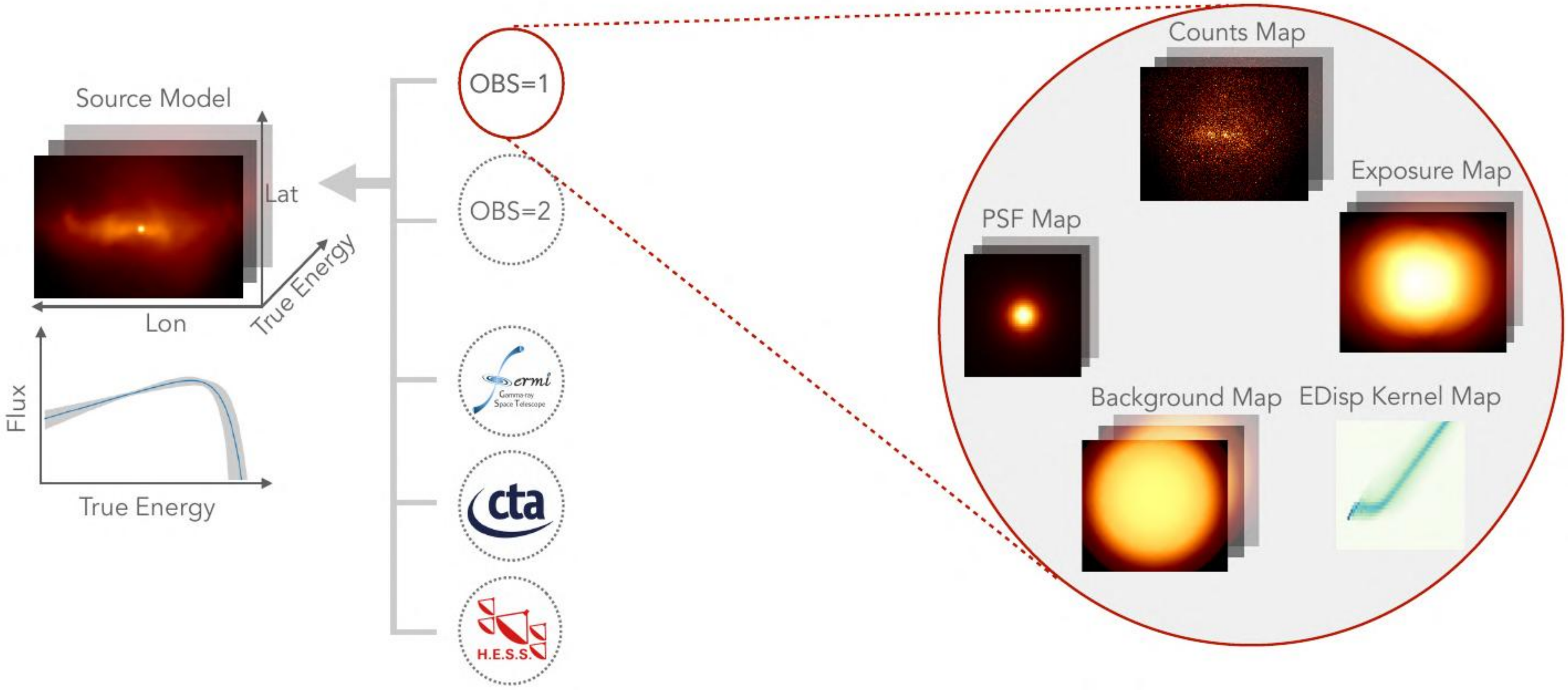
$$N_{Pred,Src} = \mathrm{EDISP}_{Src}(\mathrm{PSF}_{Src}(\mathscr{E}_{Src} \cdot f_{Src}))$$

$$f_{Src} = f_{Spectral}(E) \cdot f_{Spatial}(E, l, b) \cdot f_{Temporal}(t)$$

Slide from A. Donath (Presentation made during the Scipy 2023 conference)

Gammapy: Gamma-ray Data Analysis

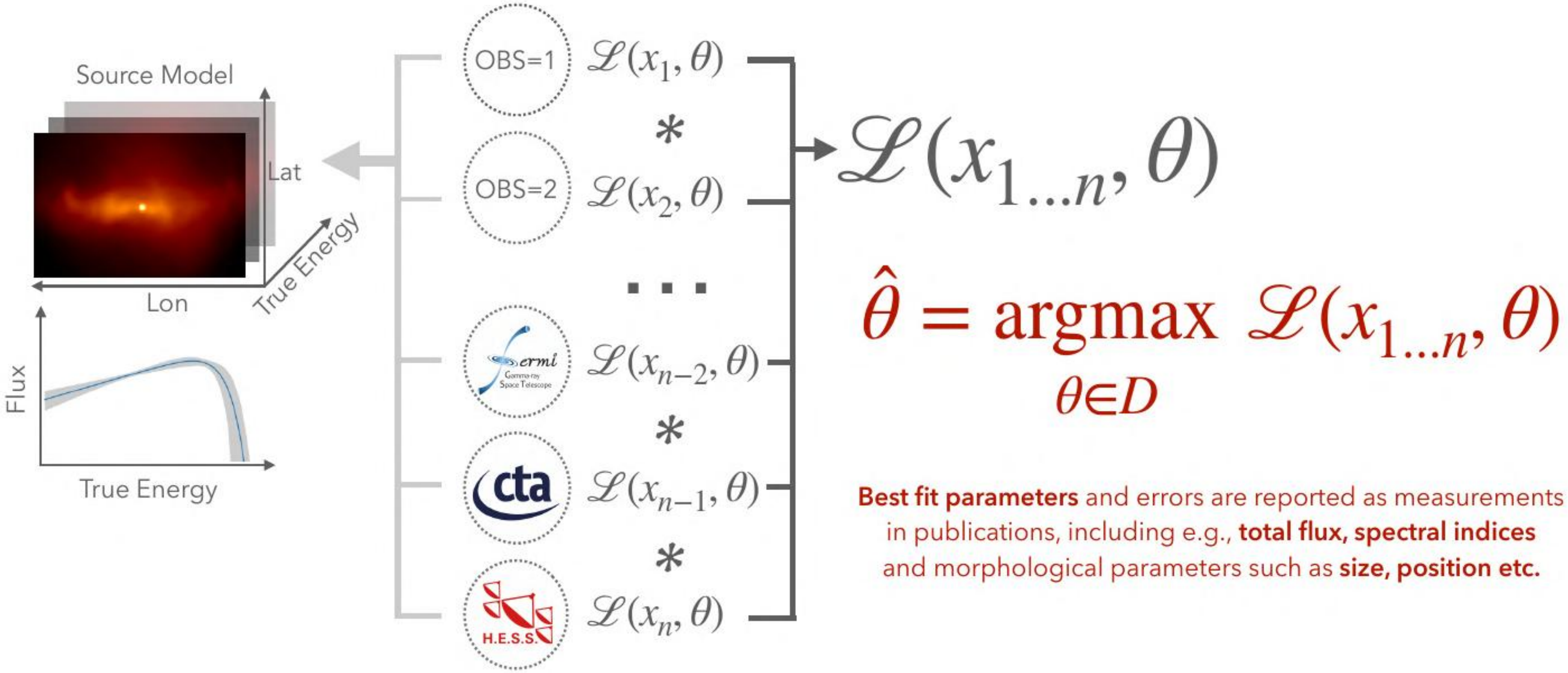
Joint Likelihood



Slide from A. Donath (Presentation made during the Scipy 2023 conference)

Gammapy: Gamma-ray Data Analysis

Joint Likelihood



$\mathscr{L}(x_1, \theta)$ * $\mathscr{L}(x_2, \theta)$ . . . $\mathscr{L}(x_{n-2}, \theta)$ * $\mathscr{L}(x_{n-1}, \theta)$ * $\mathscr{L}(x_n, \theta)$ → $\mathscr{L}(x_{1...n}, \theta)$

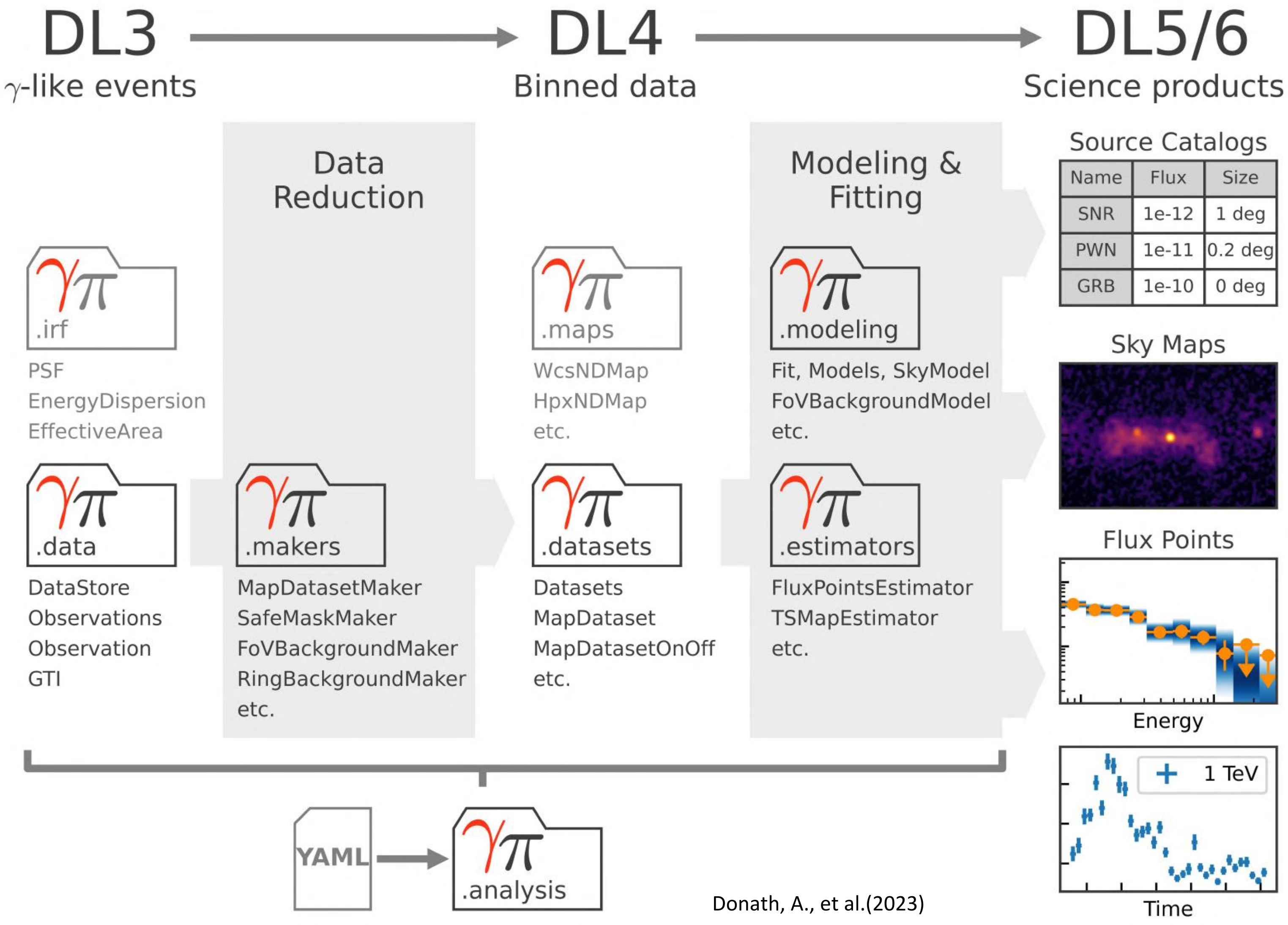
$$\hat{\theta} = \underset{\theta \in D}{\operatorname{argmax}} \; \mathscr{L}(x_{1...n}, \theta)$$

Best fit parameters and errors are reported as measurements in publications, including e.g., **total flux, spectral indices** and morphological parameters such as **size, position etc.**

Slide from A. Donath (Presentation made during the Scipy 2023 conference)

Data workflow and package structure

https://docs.gammapy.org/1.1/user-guide/package.html



DL3 → DL4 → DL5/6

γ-like events

Binned data

Science products

Data Reduction

Modeling & Fitting

.irf

PSF
EnergyDispersion
EffectiveArea

.data

DataStore
Observations
Observation
GTI

.makers

MapDatasetMaker
SafeMaskMaker
FoVBackgroundMaker
RingBackgroundMaker
etc.

.maps

WcsNDMap
HpxNDMap
etc.

.datasets

Datasets
MapDataset
MapDatasetOnOff
etc.

.modeling

Fit, Models, SkyModel
FoVBackgroundModel
etc.

.estimators

FluxPointsEstimator
TSMapEstimator
etc.

YAML → .analysis

Source Catalogs

Name	Flux	Size
SNR	1e-12	1 deg
PWN	1e-11	0.2 deg
GRB	1e-10	0 deg

Sky Maps

Flux Points

Donath, A., et al.(2023)

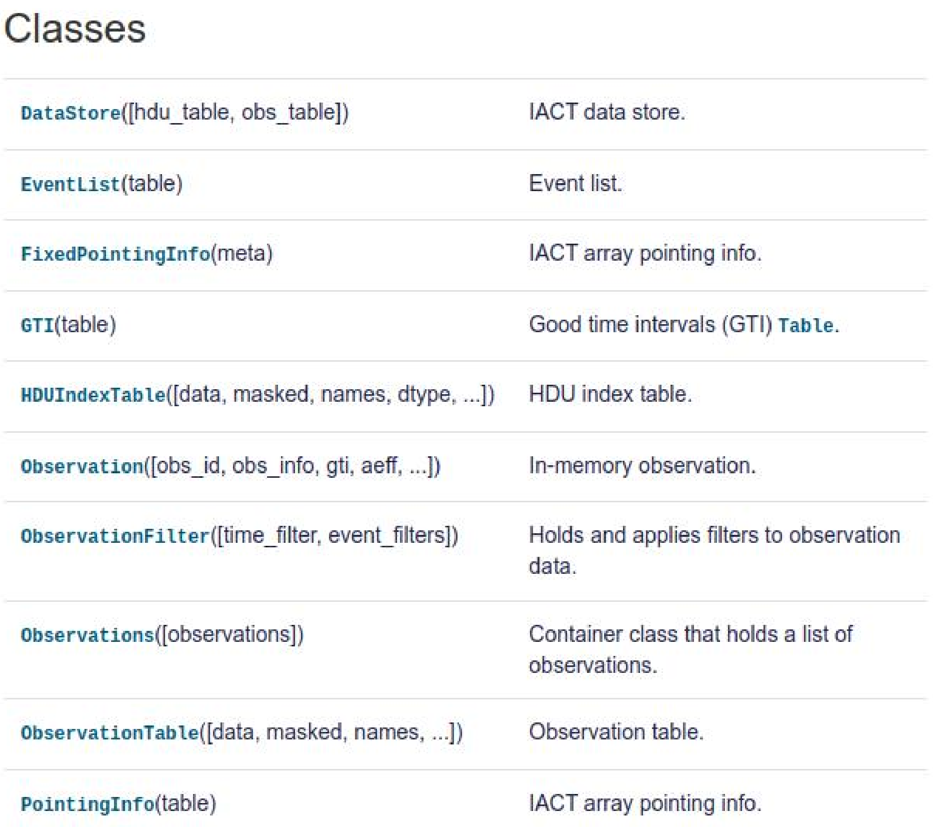
Gammapy package

- The **Gammapy package** is structured into multiple **sub-packages**.
- **Sub-packages** contain structures representing data at different reduction levels and/or algorithms to transition between these different levels.

gammapy.data

- The *gammapy.data* sub-package implements the functionality to select, read, and represent DL3 γ-ray data in memory.

https://docs.gammapy.org/1.1/api-reference/data.html#module-gammapy.data



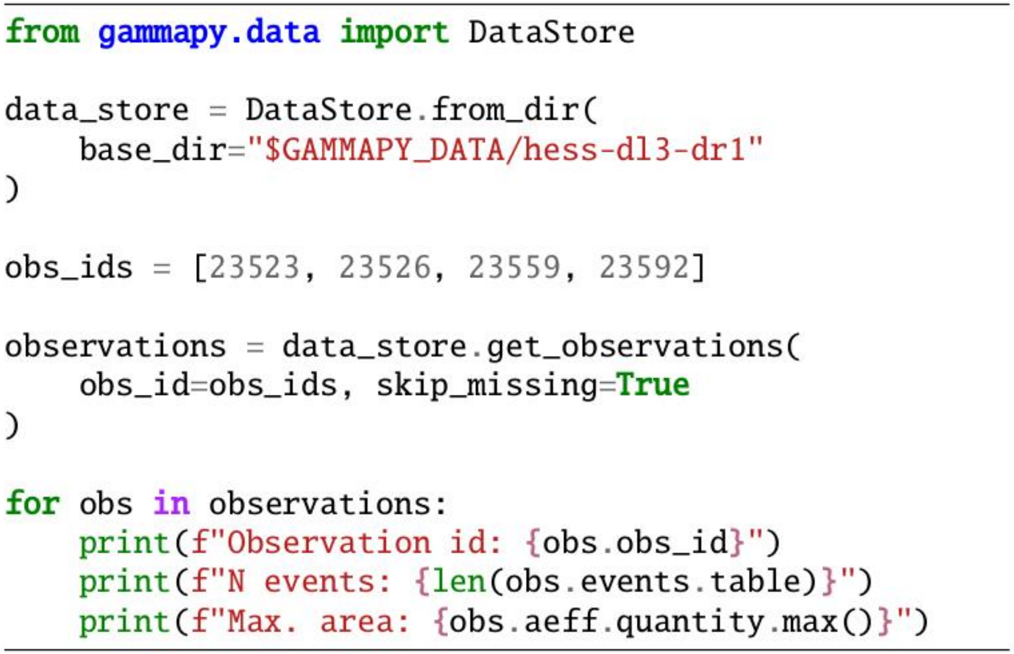
Classes

`DataStore`([hdu_table, obs_table])	IACT data store.
`EventList`(table)	Event list.
`FixedPointingInfo`(meta)	IACT array pointing info.
`GTI`(table)	Good time intervals (GTI) `Table`.
`HDUIndexTable`([data, masked, names, dtype, ...])	HDU index table.
`Observation`([obs_id, obs_info, gti, aeff, ...])	In-memory observation.
`ObservationFilter`([time_filter, event_filters])	Holds and applies filters to observation data.
`Observations`([observations])	Container class that holds a list of observations.
`ObservationTable`([data, masked, names, ...])	Observation table.
`PointingInfo`(table)	IACT array pointing info.

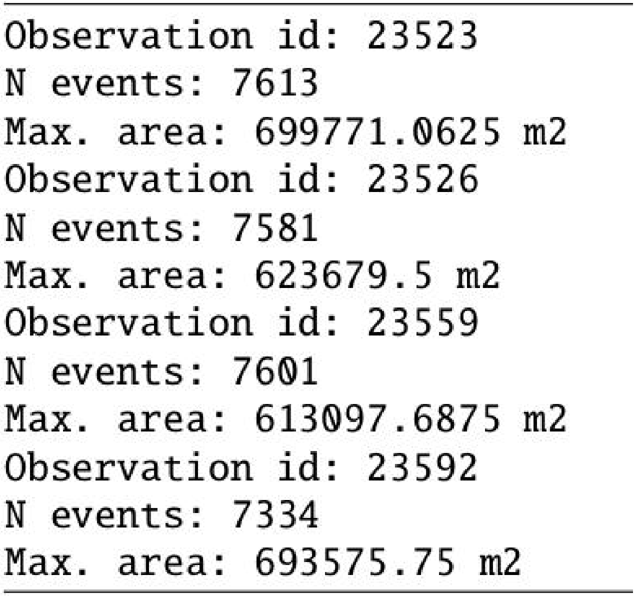
Gammapy package

gammapy.data

- The *gammapy.data* sub-package implements the functionality to select, read, and represent DL3 γ-ray data in memory.



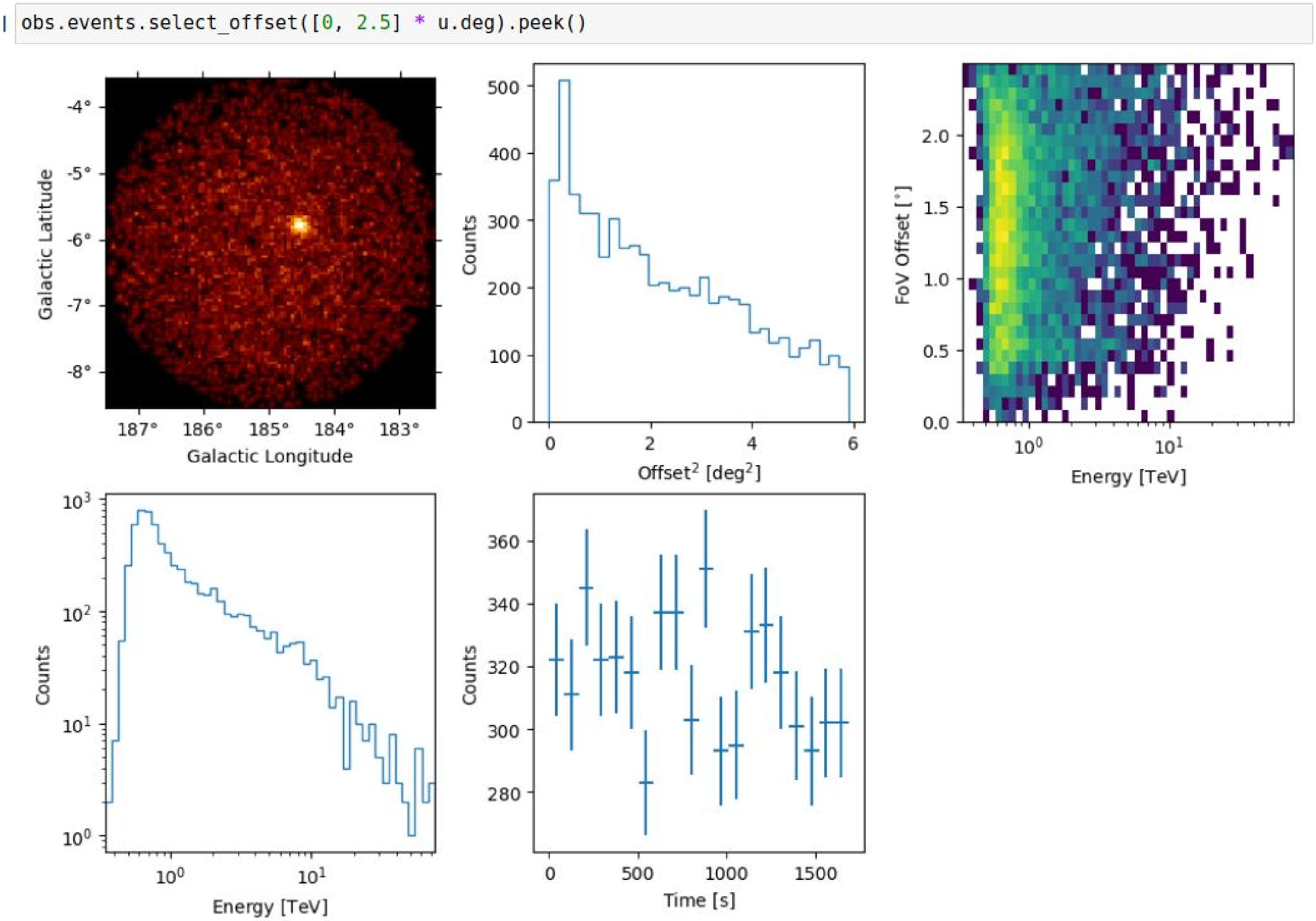
```
from gammapy.data import DataStore

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

obs_ids = [23523, 23526, 23559, 23592]

observations = data_store.get_observations(
    obs_id=obs_ids, skip_missing=True
)

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 package

gammapy.data



```
obs.events.select_offset([0, 2.5] * u.deg).peek()
```

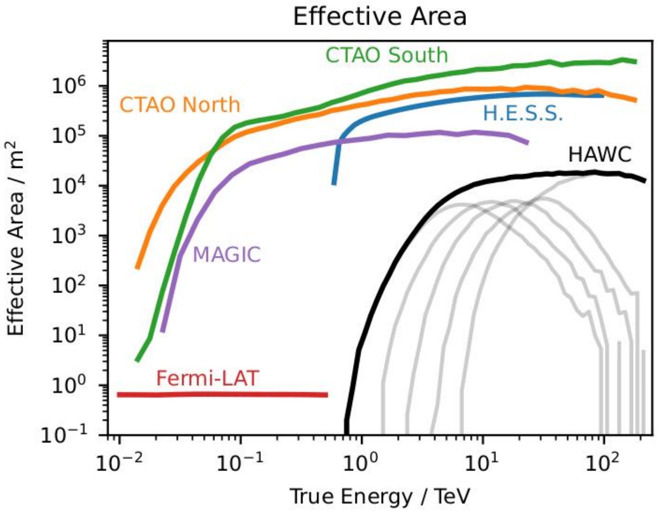
Gammapy package

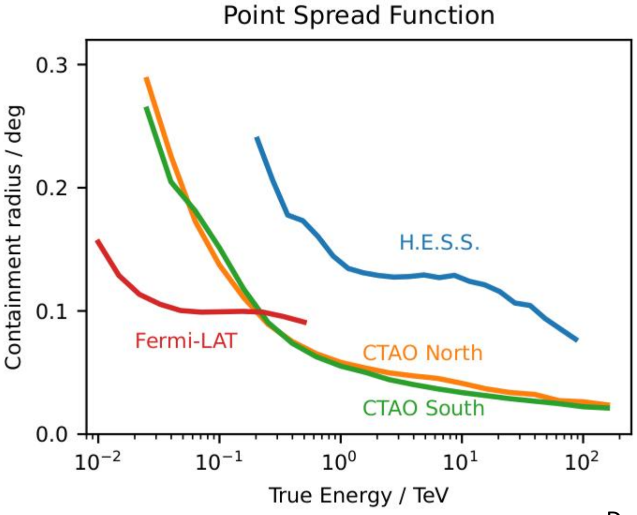
gammapy.irf

- The *gammapy.irf* sub-package contains all classes and functionalities to handle IRFs (Instrument Response Functions) in a variety of functional forms.

- The main quantities stored in the common γ-ray IRFs:
  - Effective area
  - Point spread function (PSF)
  - Energy dispersion
  - Background rate

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

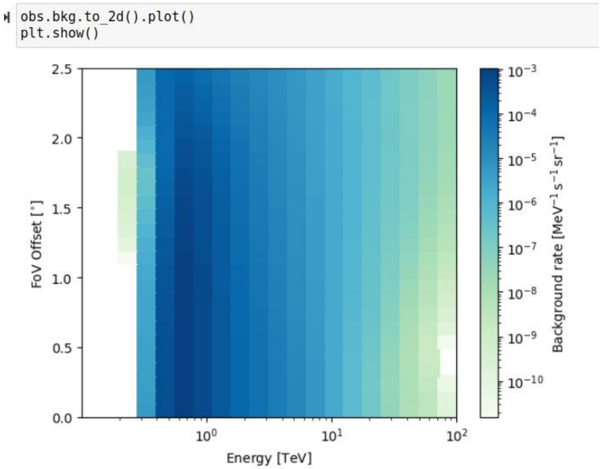




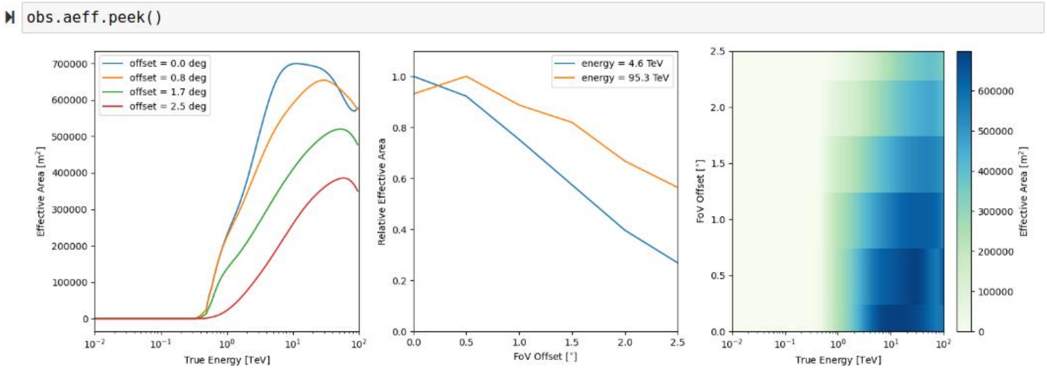
Donath, A., et al.(2023)

Gammapy package

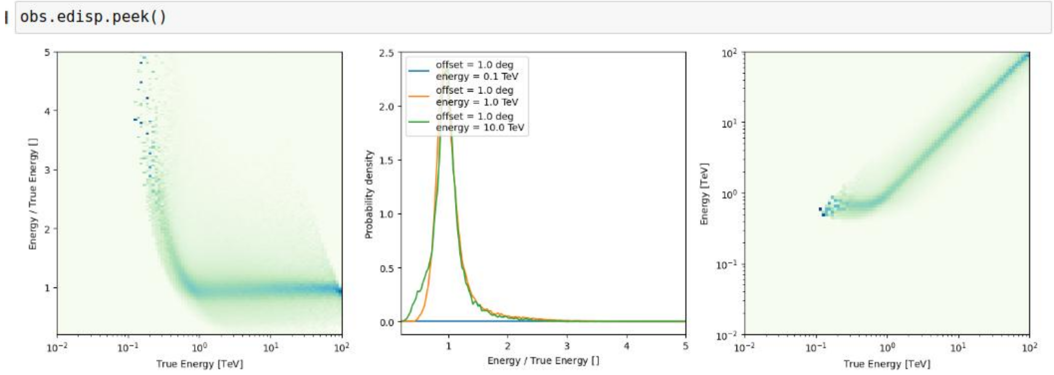
gammapy.irf



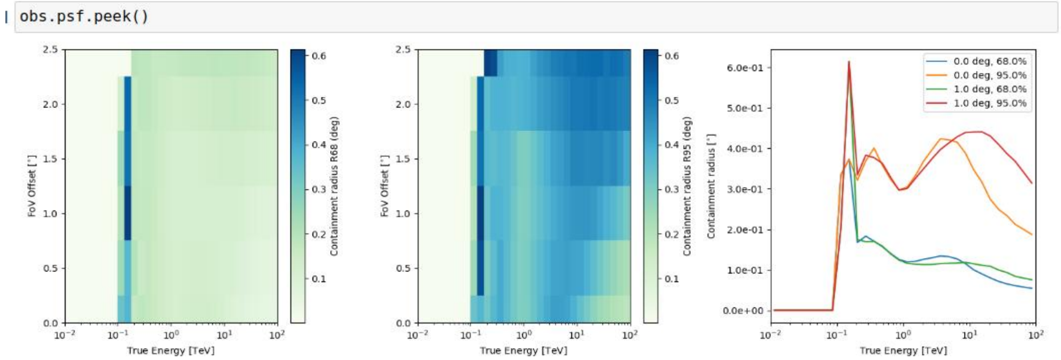
```
obs.bkg.to_2d().plot()
plt.show()
```



```
obs.aeff.peek()
```



```
obs.edisp.peek()
```



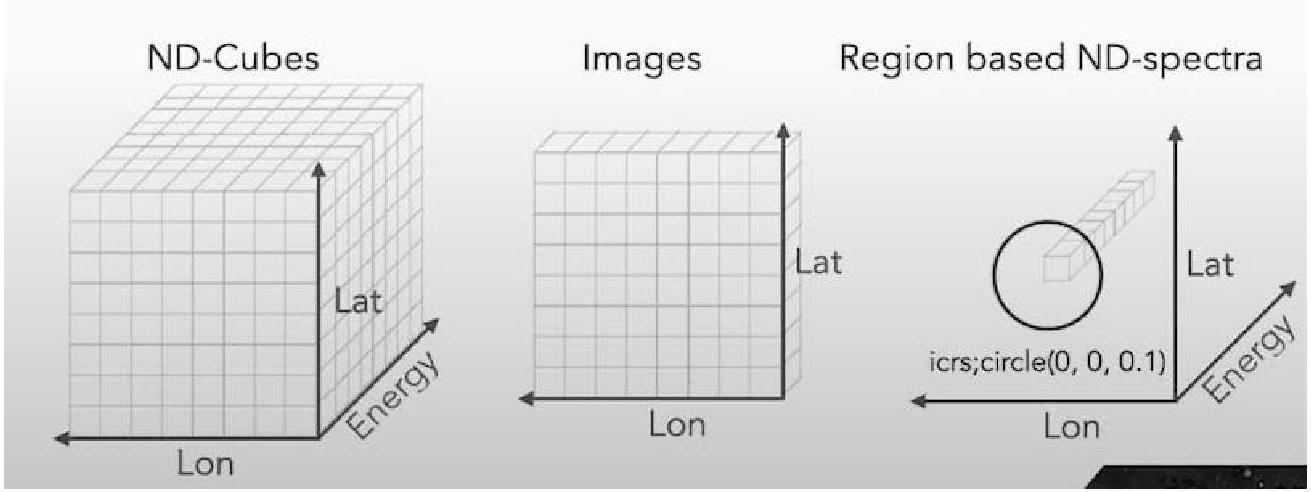
```
obs.psf.peek()
```

Gammapy package

gammapy.maps

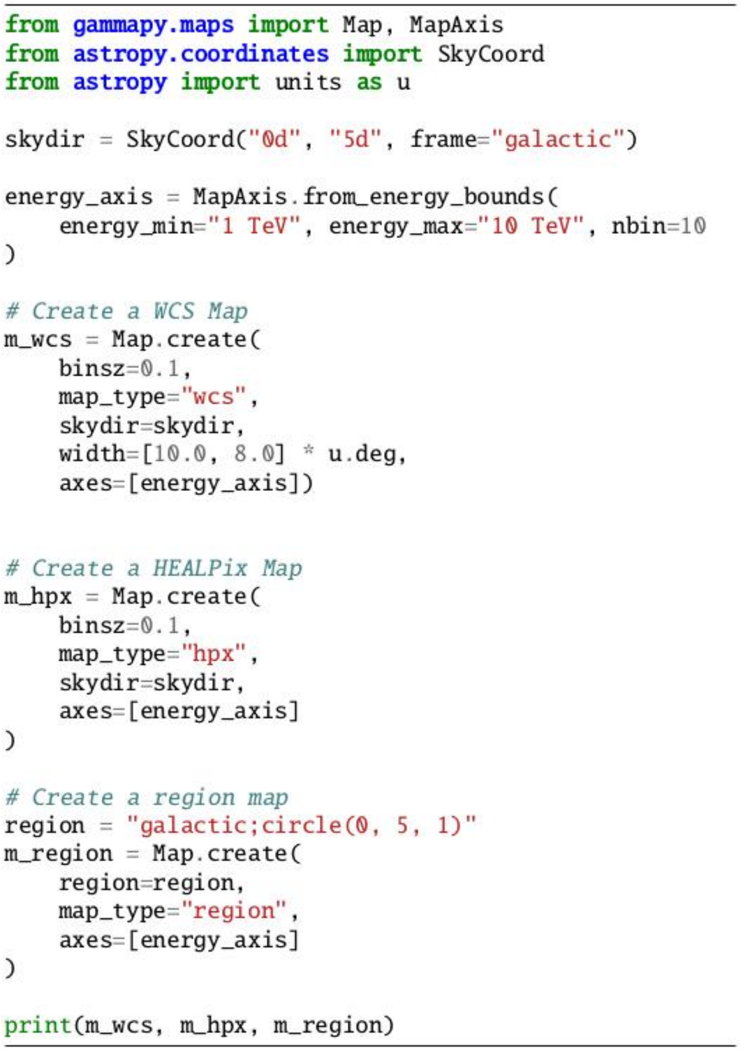
- The *gammapy.maps* sub-package:
  - provides classes that represent data structures associated with a set of coordinates or a region on a sphere;
  - allows one to handle an arbitrary number of nonspatial data dimensions, such as time or energy.

- It is organized around three types of structures:
  - geometries;
  - sky maps;
  - map axes.

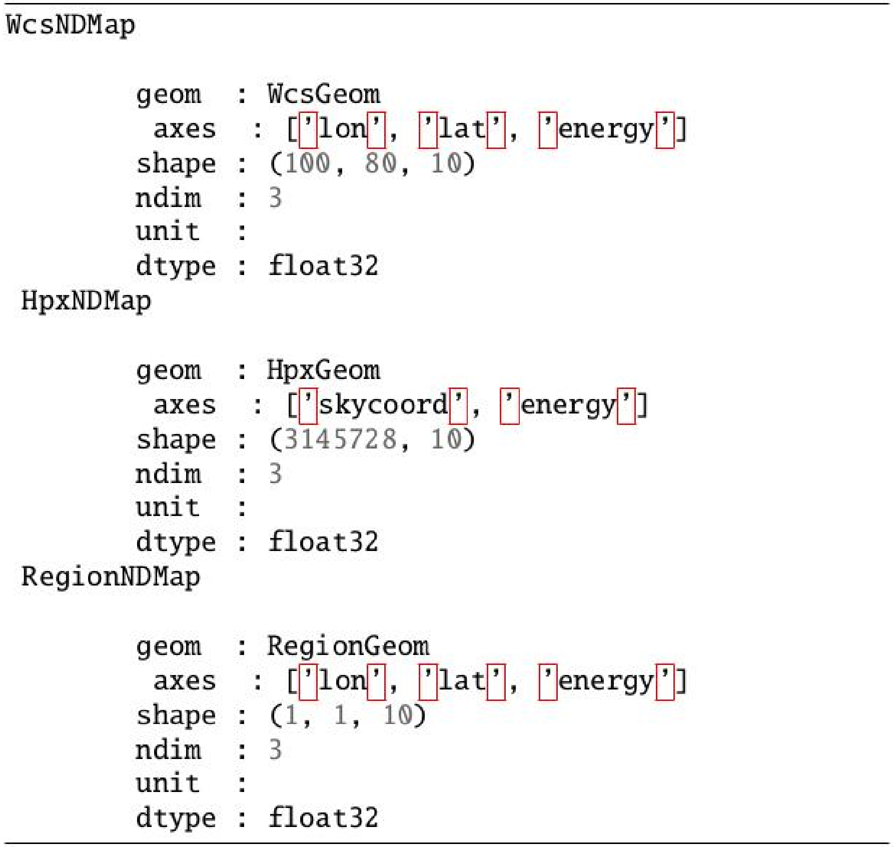


Gammapy package

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]
)

print(m_wcs, m_hpx, m_region)
```

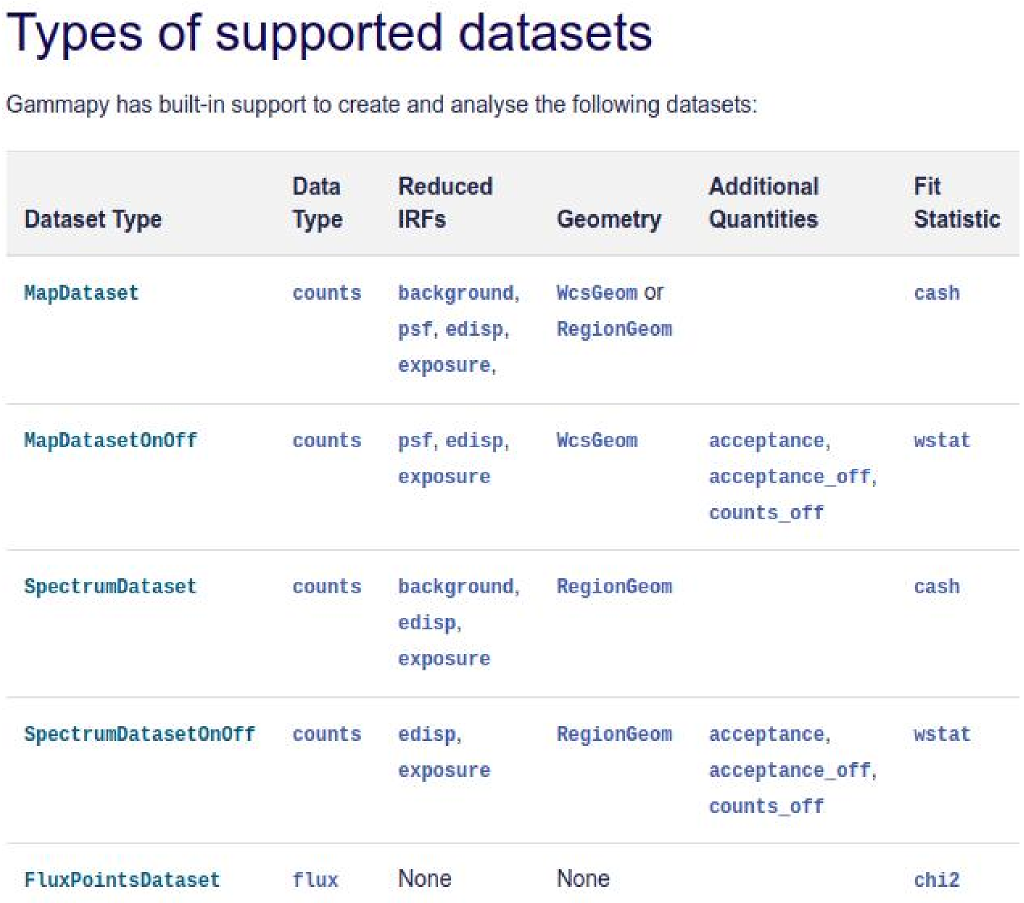


```
WcsNDMap

        geom  : WcsGeom
         axes  : ['lon', 'lat', 'energy']
        shape : (100, 80, 10)
        ndim  : 3
        unit  :
        dtype : float32
 HpxNDMap

        geom  : HpxGeom
         axes  : ['skycoord', 'energy']
        shape : (3145728, 10)
        ndim  : 3
        unit  :
        dtype : float32
 RegionNDMap

        geom  : RegionGeom
         axes  : ['lon', 'lat', 'energy']
        shape : (1, 1, 10)
        ndim  : 3
        unit  :
        dtype : float32
```

Gammapy package

gammapy.datasets

- The *gammapy.datasets* sub-package contains classes to handle reduced gamma-ray data for modeling and fitting.
- The Dataset class, for example, bundles reduced data, IRFs and model to perform likelihood fitting and joint-likelihood fitting.
- To model and fit data in Gammapy, you have to create a Datasets container object with one or multiple Dataset objects.



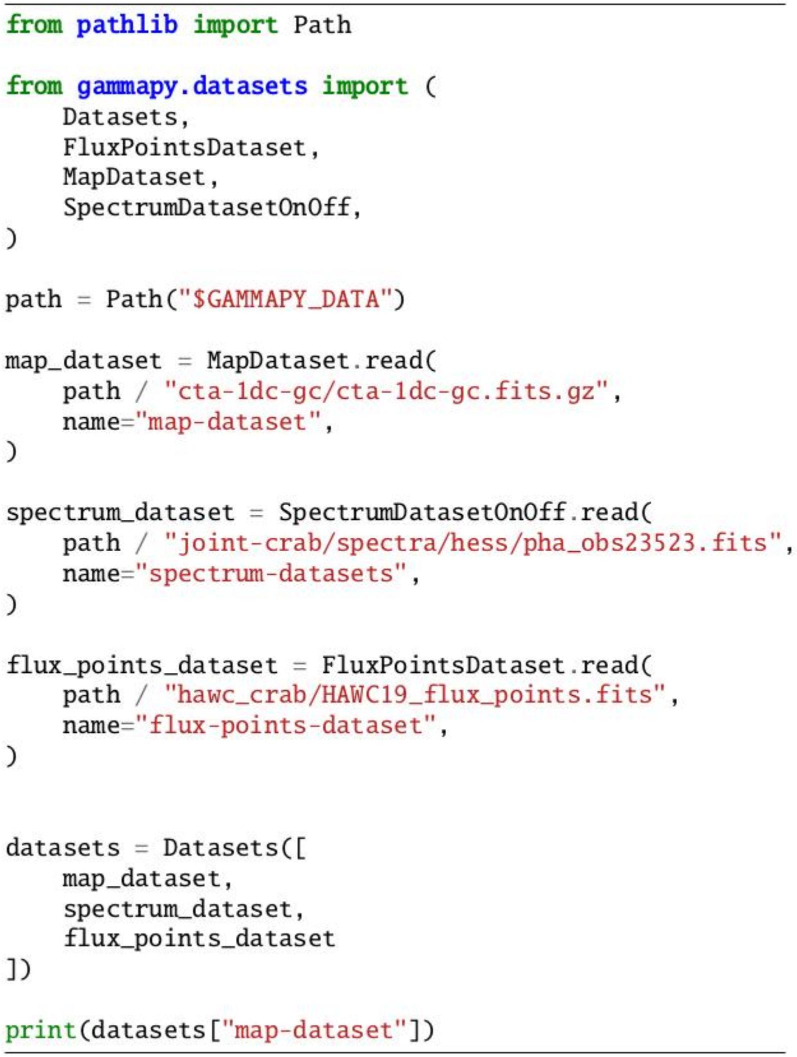
Types of supported datasets

Gammapy has built-in support to create and analyse the following datasets:

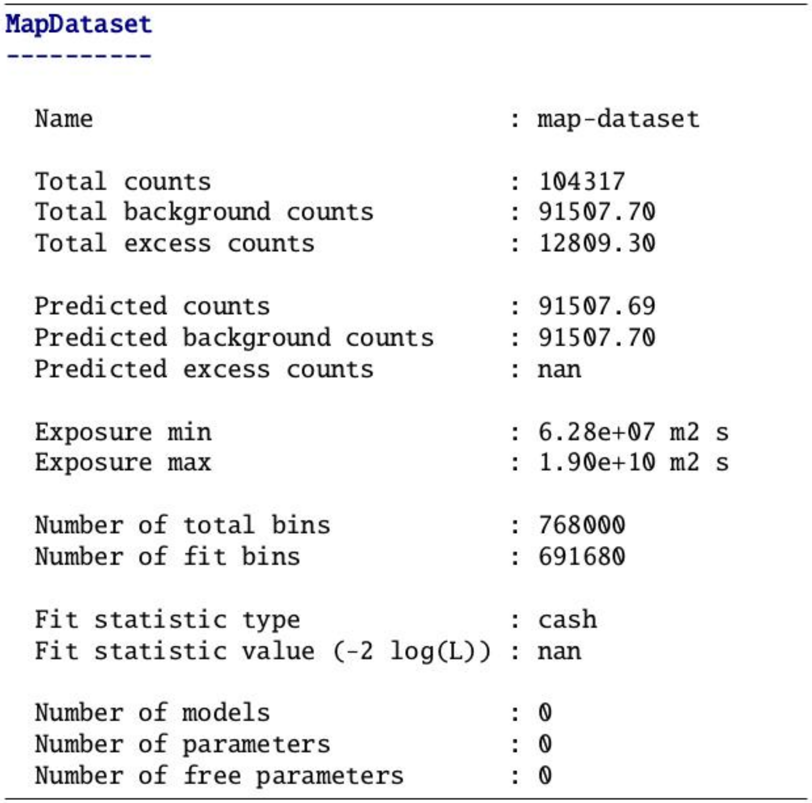
Dataset Type	Data Type	Reduced IRFs	Geometry	Additional Quantities	Fit Statistic
`MapDataset`	`counts`	`background`, `psf`, `edisp`, `exposure`,	`WcsGeom` or `RegionGeom`		`cash`
`MapDatasetOnOff`	`counts`	`psf`, `edisp`, `exposure`	`WcsGeom`	`acceptance`, `acceptance_off`, `counts_off`	`wstat`
`SpectrumDataset`	`counts`	`background`, `edisp`, `exposure`	`RegionGeom`		`cash`
`SpectrumDatasetOnOff`	`counts`	`edisp`, `exposure`	`RegionGeom`	`acceptance`, `acceptance_off`, `counts_off`	`wstat`
`FluxPointsDataset`	`flux`	None	None		`chi2`

Gammapy package

gammapy.datasets



```python
from pathlib import Path

from gammapy.datasets import (
    Datasets,
    FluxPointsDataset,
    MapDataset,
    SpectrumDatasetOnOff,
)

path = Path("$GAMMAPY_DATA")

map_dataset = MapDataset.read(
    path / "cta-1dc-gc/cta-1dc-gc.fits.gz",
    name="map-dataset",
)

spectrum_dataset = SpectrumDatasetOnOff.read(
    path / "joint-crab/spectra/hess/pha_obs23523.fits",
    name="spectrum-datasets",
)

flux_points_dataset = FluxPointsDataset.read(
    path / "hawc_crab/HAWC19_flux_points.fits",
    name="flux-points-dataset",
)


datasets = Datasets([
    map_dataset,
    spectrum_dataset,
    flux_points_dataset
])

print(datasets["map-dataset"])
```



```
MapDataset
----------

  Name                            : map-dataset

  Total counts                    : 104317
  Total background counts         : 91507.70
  Total excess counts             : 12809.30

  Predicted counts                : 91507.69
  Predicted background counts     : 91507.70
  Predicted excess counts         : nan

  Exposure min                    : 6.28e+07 m2 s
  Exposure max                    : 1.90e+10 m2 s

  Number of total bins            : 768000
  Number of fit bins              : 691680

  Fit statistic type              : cash
  Fit statistic value (-2 log(L)) : nan

  Number of models                : 0
  Number of parameters            : 0
  Number of free parameters       : 0
```

Gammapy package

gammapy.maker

- The *gammapy.datasets* sub-package contains the various classes and functions required to **process and prepare** γ-ray data from the DL3 to the DL4.

- The DL3 data is prepared for modeling and fitting, by binning events into a counts map and interpolating the exposure, background, psf and energy dispersion on the chosen analysis geometry.

gammapy.stats

- The *gammapy.stats* sub-package contains the **fit statistics** and the associated statistical estimators commonly adopted in γ-ray astronomy.

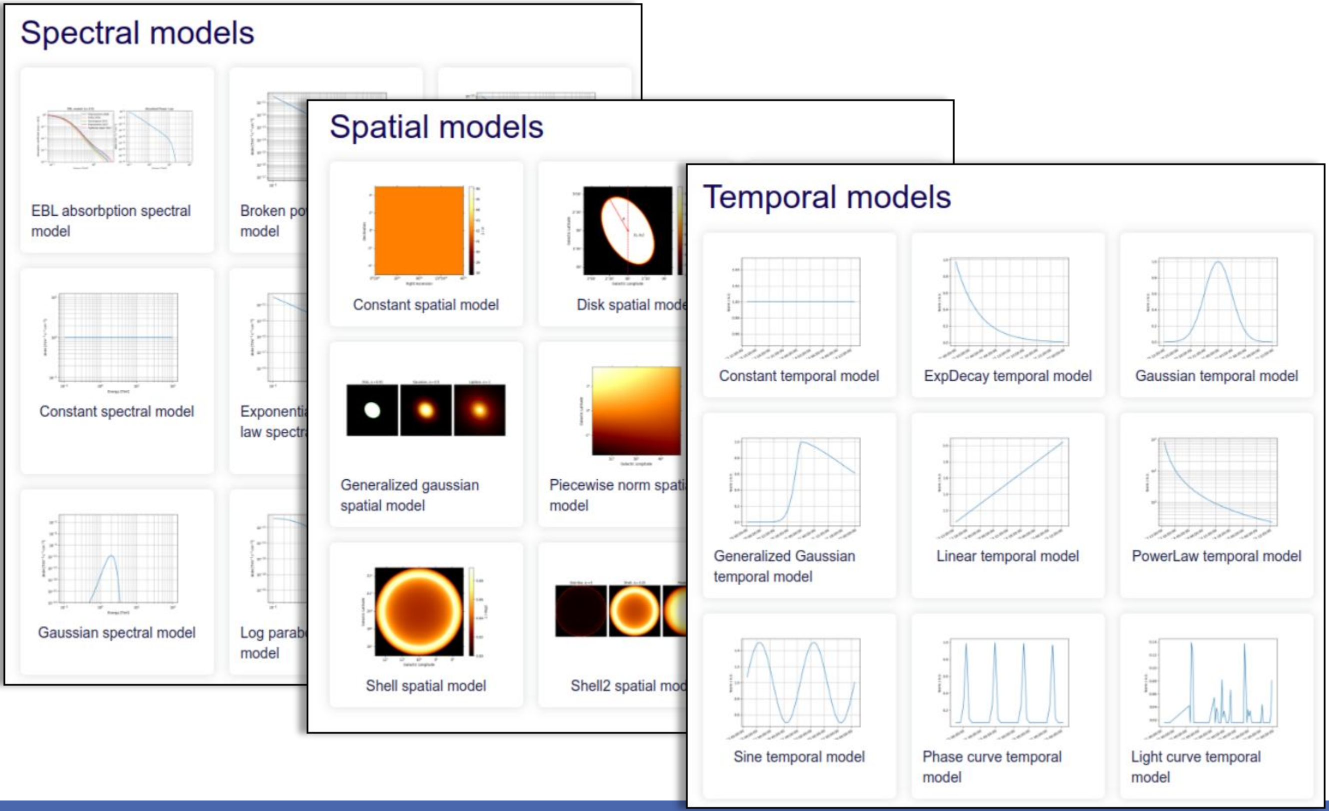
- It contains classes that perform maximum likelihood ratio tests to estimate significance and compute likelihood profiles to measure errors and upper limits.

Gammapy package

gammapy.modeling

- The *gammapy.modeling* sub-package contains all the functionality related to modeling and fitting data. This includes **spectral, spatial and temporal model** classes, as well as the **fit**.

- **Models**



Spectral models

EBL absorbption spectral model

Broken po model

Constant spectral model

Exponenti law spectr

Gaussian spectral model

Log parab model

Spatial models

Constant spatial model

Disk spatial mode

Generalized gaussian spatial model

Piecewise norm spati model

Shell spatial model

Shell2 spatial mod

Temporal models

Constant temporal model

ExpDecay temporal model

Gaussian temporal model

Generalized Gaussian temporal model

Linear temporal model

PowerLaw temporal model

Sine temporal model

Phase curve temporal model

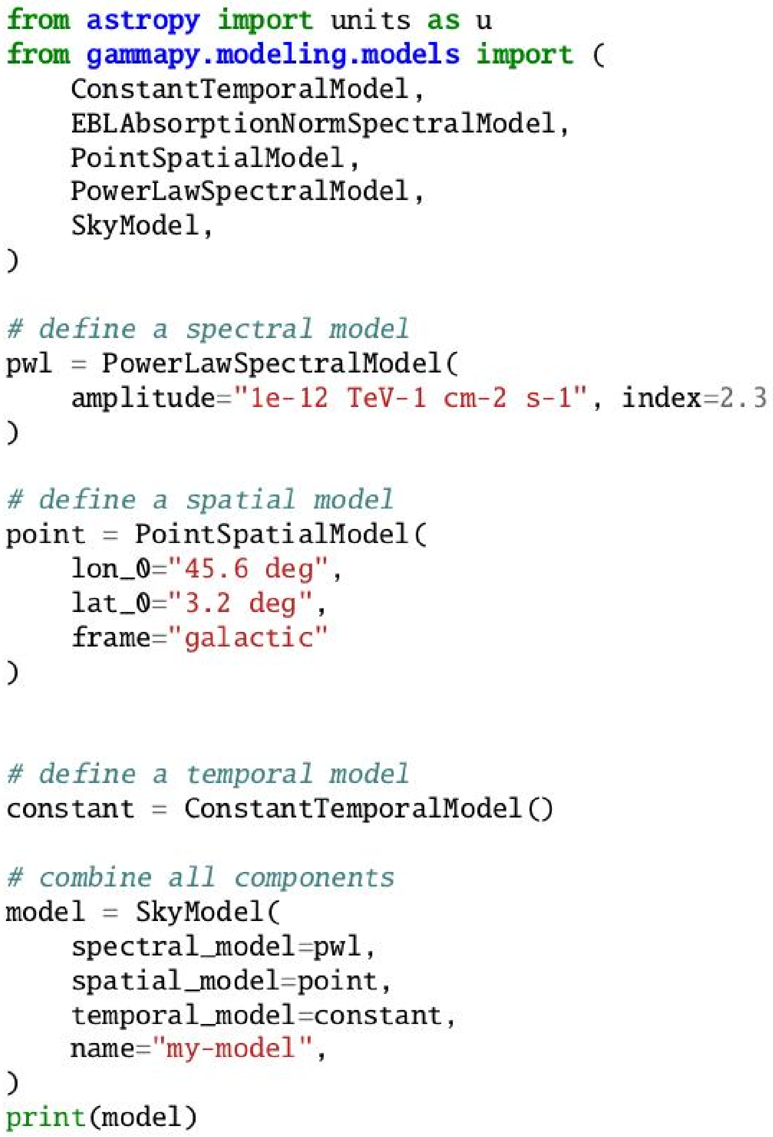
Light curve temporal model

Model Gallery

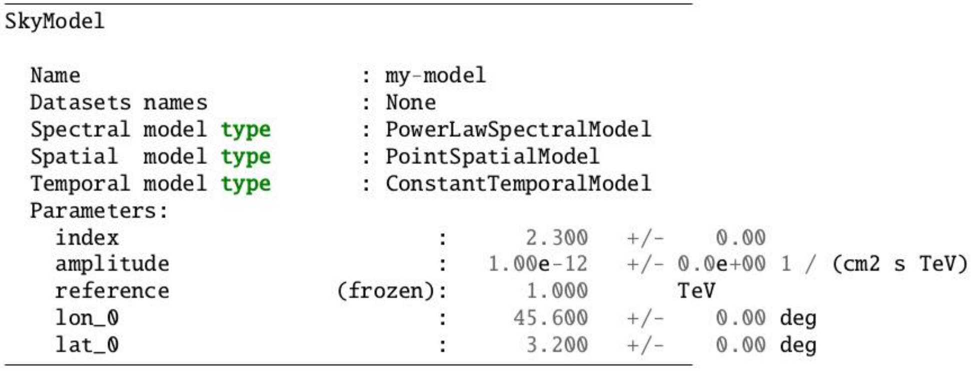
Gammapy package

gammapy.modeling

- **Models**



```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)
```



```
SkyModel

  Name                      : my-model
  Datasets names            : None
  Spectral model type       : PowerLawSpectralModel
  Spatial  model type       : PointSpatialModel
  Temporal model type       : ConstantTemporalModel
  Parameters:
    index                         :      2.300   +/-    0.00
    amplitude                     :   1.00e-12   +/- 0.0e+00 1 / (cm2 s TeV)
    reference             (frozen):      1.000       TeV
    lon_0                         :     45.600   +/-    0.00 deg
    lat_0                         :      3.200   +/-    0.00 deg
```

Gammapy package

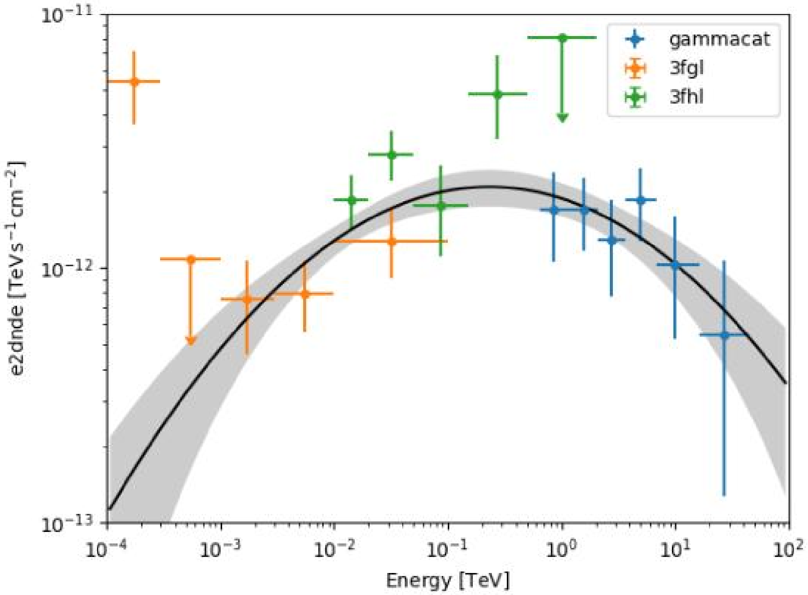
gammapy.modeling

- **Fit**

- It provides **methods to optimize**, model parameters and estimate their errors and correlations.

- Models can be **unique** for a given dataset, or contribute to **multiple** datasets, allowing one to perform a **joint fit** to multiple IACT datasets, or to jointly fit IACT and Fermi-LAT datasets.

- The Fit class provides a uniform interface to multiple fitting backends:
  - iminuit (Dembinski et al. 2020)
  - scipy.optimize (Virtanen et al. 2020)
  - Sherpa (Refsdal et al. 2011; Freeman et al. 2001)

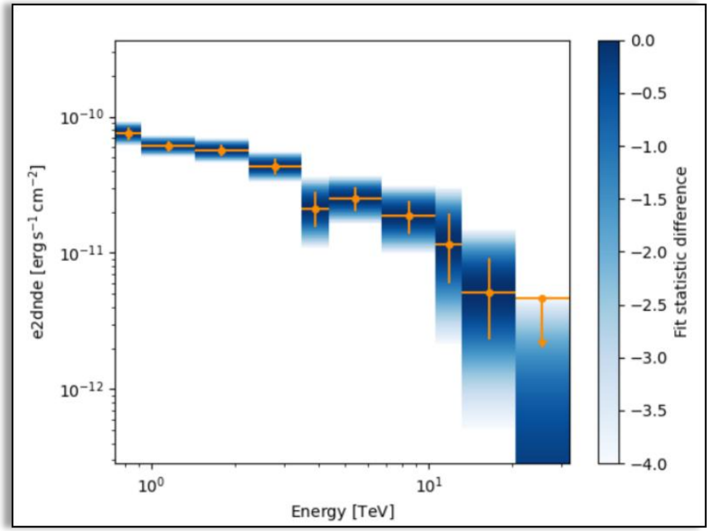


Gammapy package

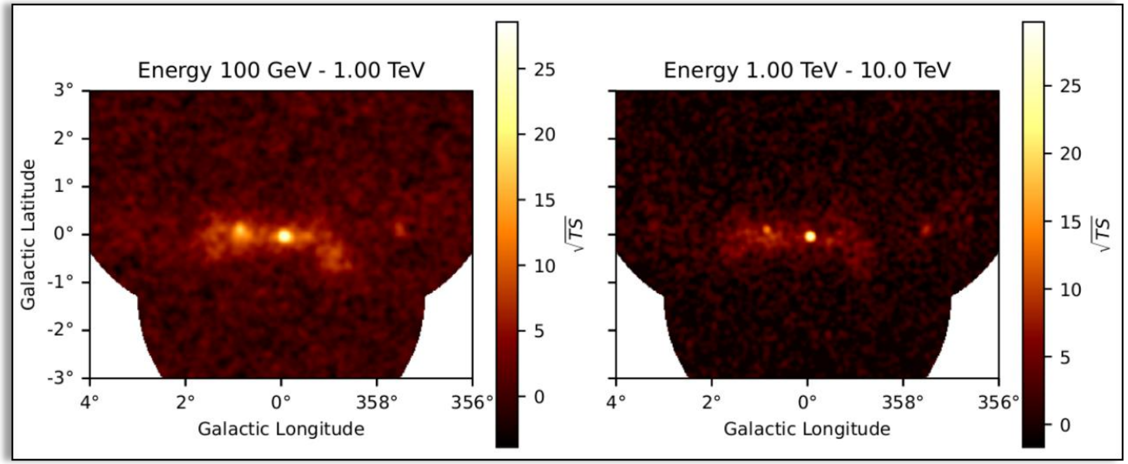
gammapy.estimators

- The *gammapy.estimators* sub-package features methods to compute flux points, light curves, flux maps and flux profiles from data.

- In general the flux can be estimated using two methods:
  - Based on model fitting;
  - Based on excess.



gammapy.estimators.FluxPointsEstimator

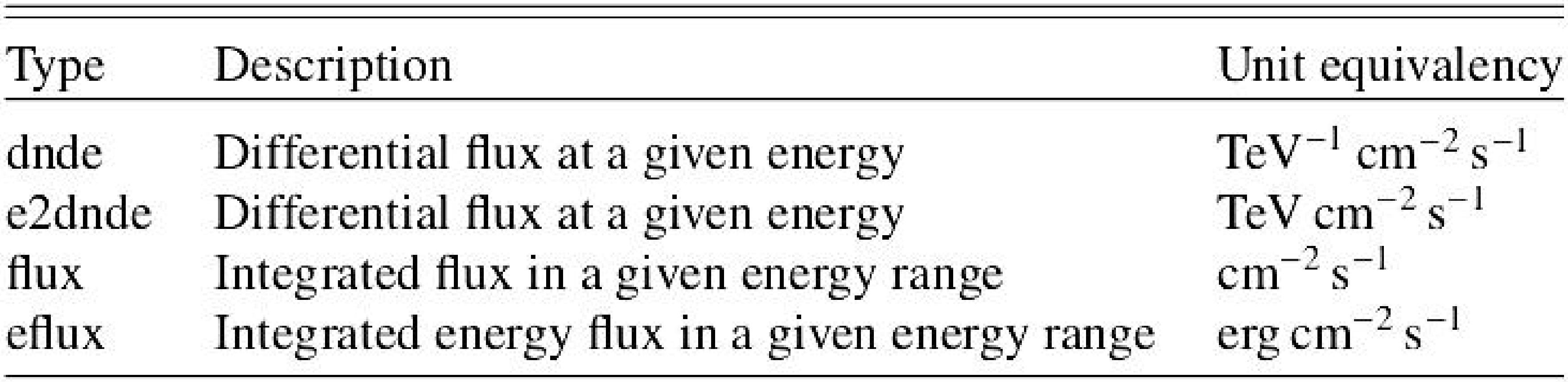


gammapy.estimators.TSMapEstimator

Gammapy package

gammapy.estimators

- Definition of the different SED types supported in Gammapy.



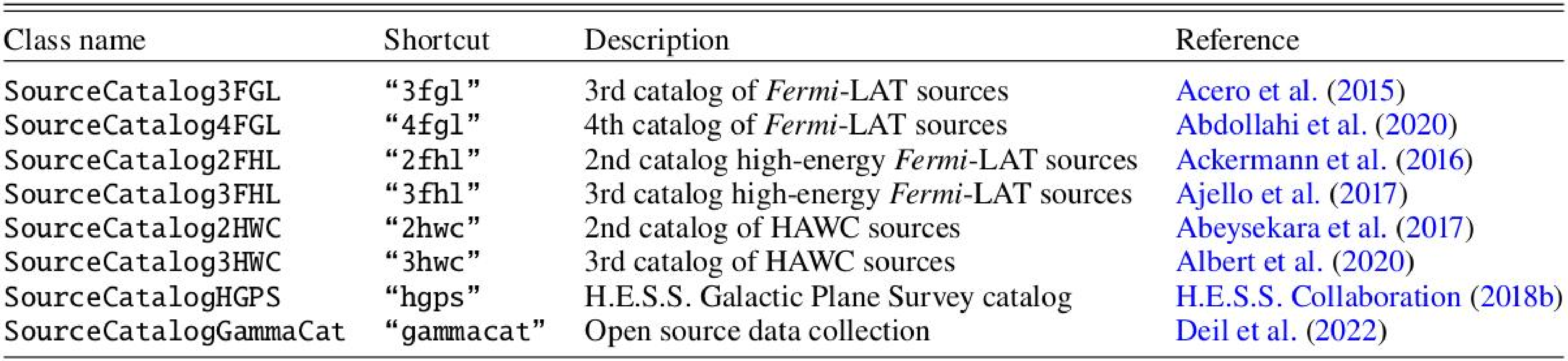
Type	Description	Unit equivalency
dnde	Differential flux at a given energy	$\mathrm{TeV^{-1}\,cm^{-2}\,s^{-1}}$
e2dnde	Differential flux at a given energy	$\mathrm{TeV\,cm^{-2}\,s^{-1}}$
flux	Integrated flux in a given energy range	$\mathrm{cm^{-2}\,s^{-1}}$
eflux	Integrated energy flux in a given energy range	$\mathrm{erg\,cm^{-2}\,s^{-1}}$

Gammapy package

gammapy.catalog

- The *gammapy.catalog* sub-package provides a convenient access to the most important γ-ray catalogs.

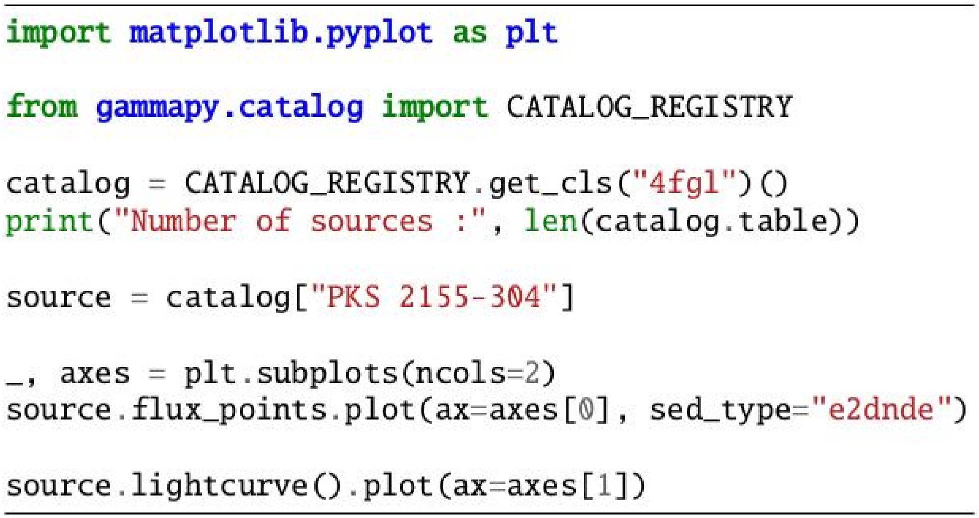
- Supported catalogs in *gammapy.catalog*:



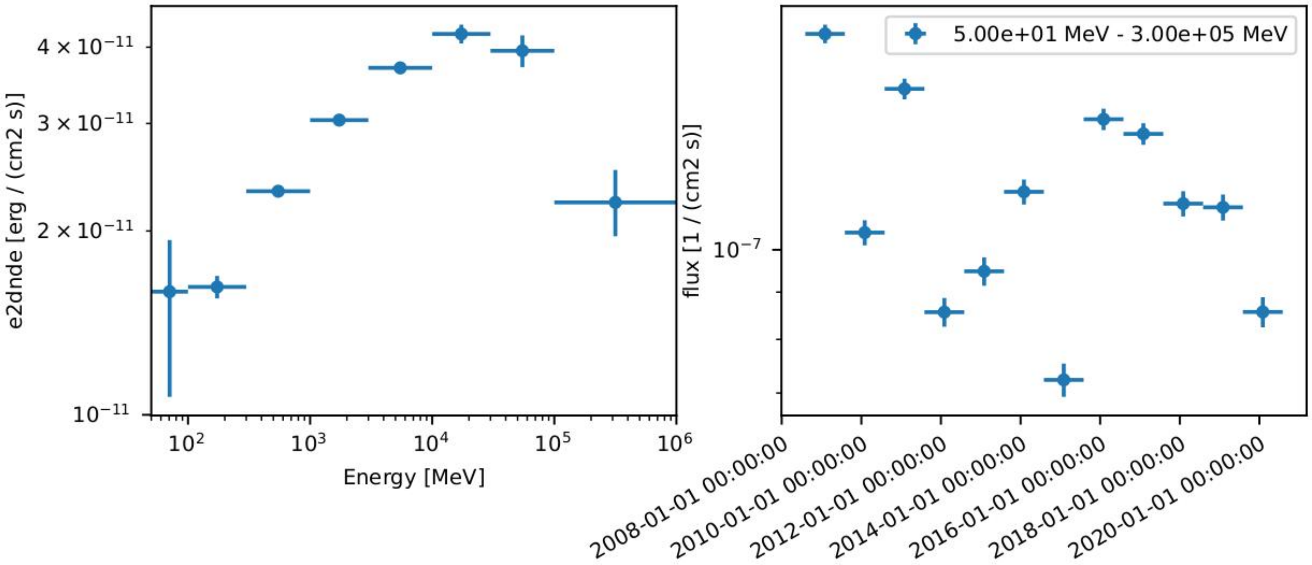
Class name	Shortcut	Description	Reference
`SourceCatalog3FGL`	“`3fgl`”	3rd catalog of *Fermi*-LAT sources	Acero et al. (2015)
`SourceCatalog4FGL`	“`4fgl`”	4th catalog of *Fermi*-LAT sources	Abdollahi et al. (2020)
`SourceCatalog2FHL`	“`2fhl`”	2nd catalog high-energy *Fermi*-LAT sources	Ackermann et al. (2016)
`SourceCatalog3FHL`	“`3fhl`”	3rd catalog high-energy *Fermi*-LAT sources	Ajello et al. (2017)
`SourceCatalog2HWC`	“`2hwc`”	2nd catalog of HAWC sources	Abeysekara et al. (2017)
`SourceCatalog3HWC`	“`3hwc`”	3rd catalog of HAWC sources	Albert et al. (2020)
`SourceCatalogHGPS`	“`hgps`”	H.E.S.S. Galactic Plane Survey catalog	H.E.S.S. Collaboration (2018b)
`SourceCatalogGammaCat`	“`gammacat`”	Open source data collection	Deil et al. (2022)

Gammapy package

gammapy.catalog

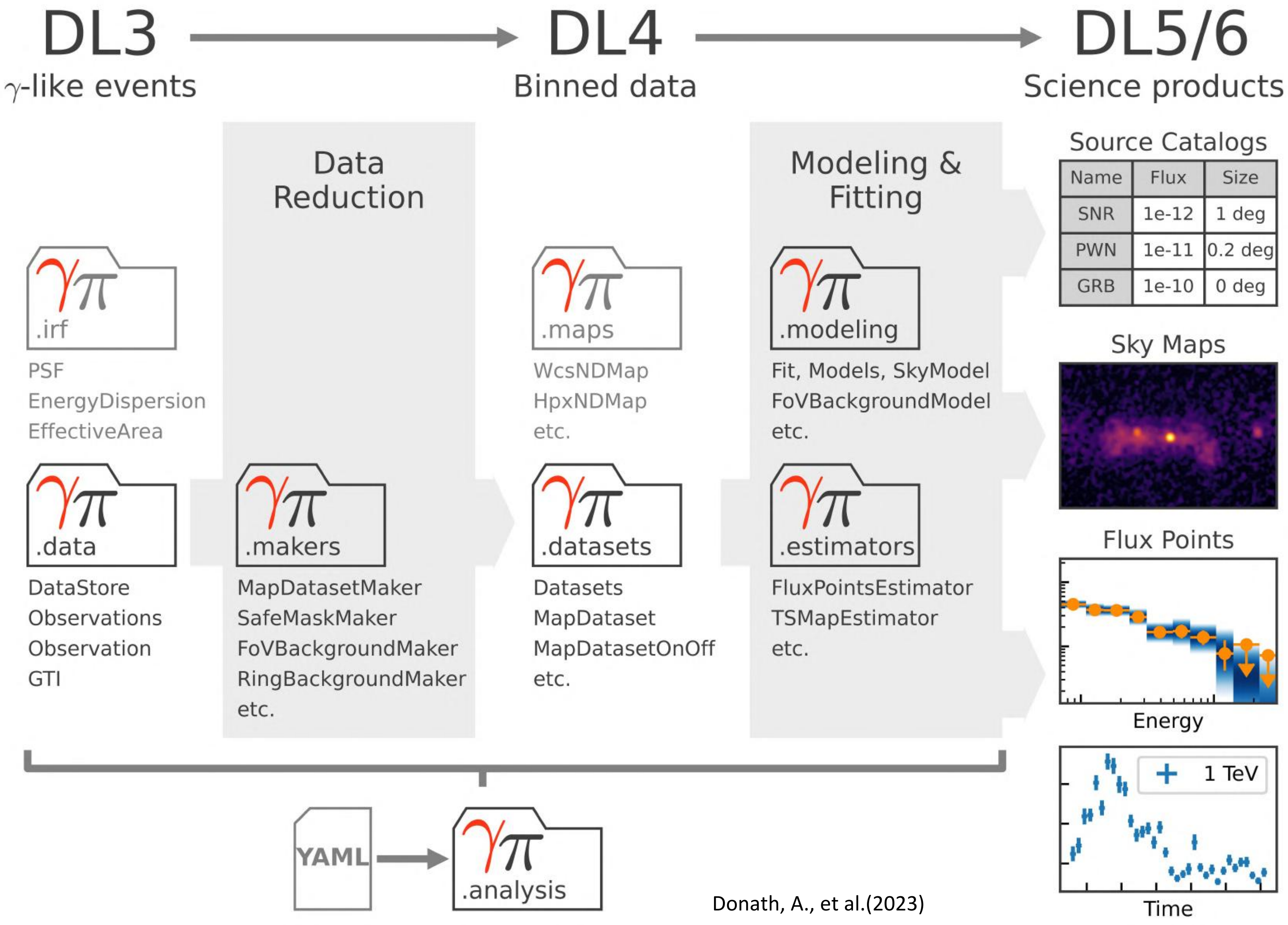


```
import matplotlib.pyplot as plt

from gammapy.catalog import CATALOG_REGISTRY

catalog = CATALOG_REGISTRY.get_cls("4fgl")()
print("Number of sources :", len(catalog.table))

source = catalog["PKS 2155-304"]

_, axes = plt.subplots(ncols=2)
source.flux_points.plot(ax=axes[0], sed_type="e2dnde")

source.lightcurve().plot(ax=axes[1])
```



Gammapy analysis workflow

https://docs.gammapy.org/1.1/user-guide/package.html



DL3 → DL4 → DL5/6

DL3: γ-like events

DL4: Binned data

DL5/6: Science products

Data Reduction

Modeling & Fitting

.irf
PSF
EnergyDispersion
EffectiveArea

.data
DataStore
Observations
Observation
GTI

.makers
MapDatasetMaker
SafeMaskMaker
FoVBackgroundMaker
RingBackgroundMaker
etc.

.maps
WcsNDMap
HpxNDMap
etc.

.datasets
Datasets
MapDataset
MapDatasetOnOff
etc.

.modeling
Fit, Models, SkyModel
FoVBackgroundModel
etc.

.estimators
FluxPointsEstimator
TSMapEstimator
etc.

YAML → .analysis

Source Catalogs

Name	Flux	Size
SNR	1e-12	1 deg
PWN	1e-11	0.2 deg
GRB	1e-10	0 deg

Sky Maps

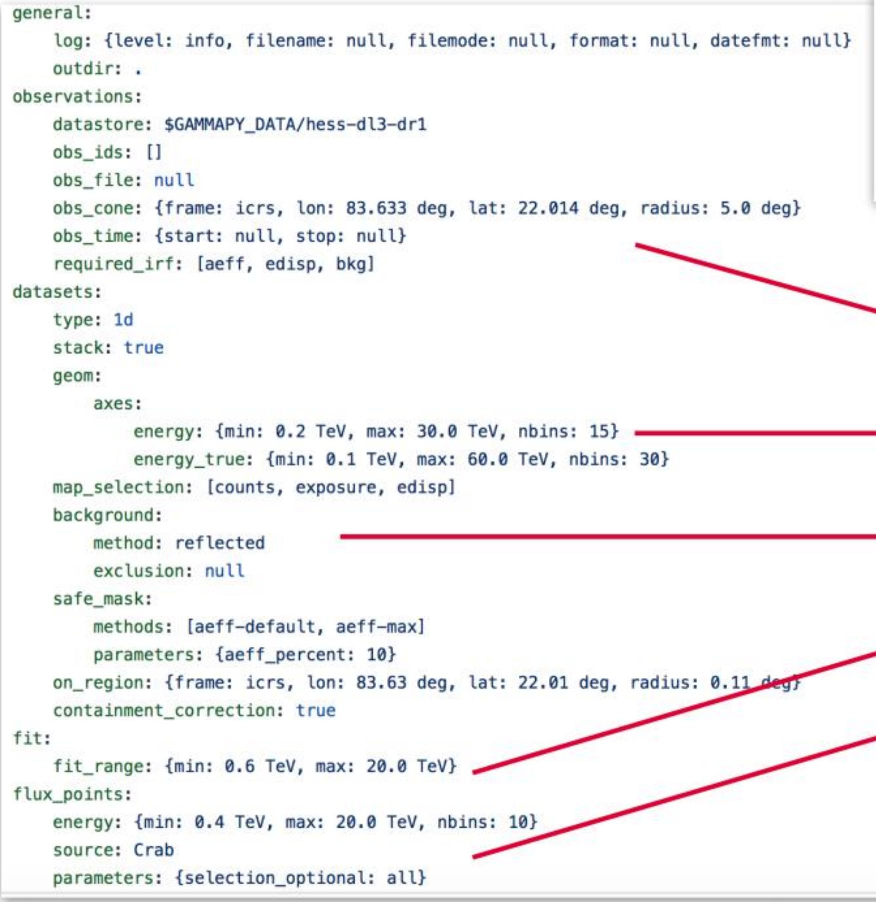
Flux Points

Donath, A., et al.(2023)

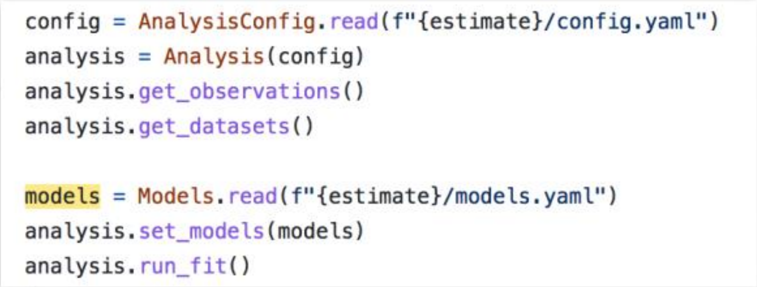
Gammapy analysis workflow

Config-file driven analysis

The YAML configuration file



```yaml
general:
    log: {level: info, filename: null, filemode: null, format: null, datefmt: null}
    outdir: .
observations:
    datastore: $GAMMAPY_DATA/hess-dl3-dr1
    obs_ids: []
    obs_file: null
    obs_cone: {frame: icrs, lon: 83.633 deg, lat: 22.014 deg, radius: 5.0 deg}
    obs_time: {start: null, stop: null}
    required_irf: [aeff, edisp, bkg]
datasets:
    type: 1d
    stack: true
    geom:
        axes:
            energy: {min: 0.2 TeV, max: 30.0 TeV, nbins: 15}
            energy_true: {min: 0.1 TeV, max: 60.0 TeV, nbins: 30}
    map_selection: [counts, exposure, edisp]
    background:
        method: reflected
        exclusion: null
    safe_mask:
        methods: [aeff-default, aeff-max]
        parameters: {aeff_percent: 10}
    on_region: {frame: icrs, lon: 83.63 deg, lat: 22.01 deg, radius: 0.11 deg}
    containment_correction: true
fit:
    fit_range: {min: 0.6 TeV, max: 20.0 TeV}
flux_points:
    energy: {min: 0.4 TeV, max: 20.0 TeV, nbins: 10}
    source: Crab
    parameters: {selection_optional: all}
```



```python
config = AnalysisConfig.read(f"{estimate}/config.yaml")
analysis = Analysis(config)
analysis.get_observations()
analysis.get_datasets()

models = Models.read(f"{estimate}/models.yaml")
analysis.set_models(models)
analysis.run_fit()
```

Select observations

Define target Dataset geometry

Define data reduction methods

Define Fit configuration

Define high level estimators config.

See High Level Interface tutorial

Slide from B. Khélifi (Presentation made during a course in Brazil)

Outlook and Links

- **Gammapy Current version: 1.2** -- > Date: Feb 29, 2024

- **Useful Links:**
  - Gammapy Web page
  - Gammapy GitHub Discussions
  - Acknowledging or Citing gammapy
  - Contact points and communication channels for Gammapy
  - Gammapy paper in A&A

- **ESCAPE Data Science Summer School 2021:**
  - School main page
  - GitHub repository for all course material
  - YouTube recording of the presentations

Novos Arranjos de Pesquisa e Inovação

FENŌMENOS EXTREMOS DO

UNIVERSO

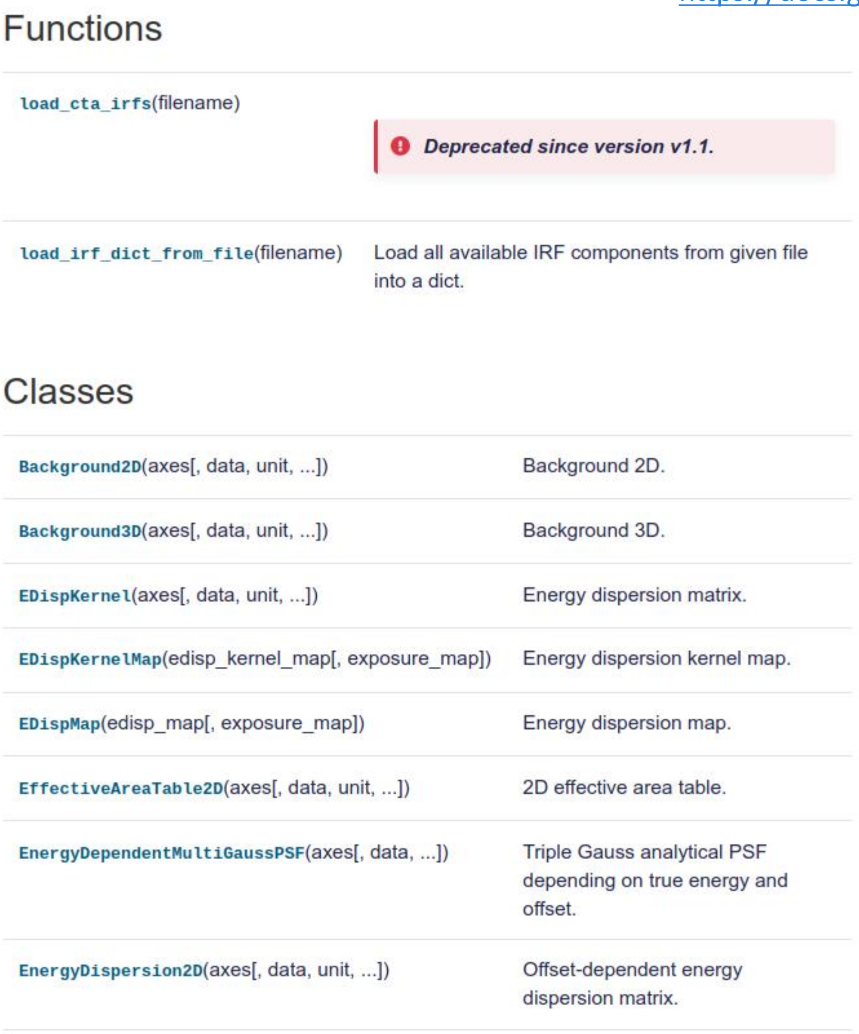
Now, let's move on to gammapy in practice!

Backup Slides

Gammapy package

gammapy.irf

https://docs.gammapy.org/1.1/api-reference/irf.html#module-gammapy.irf

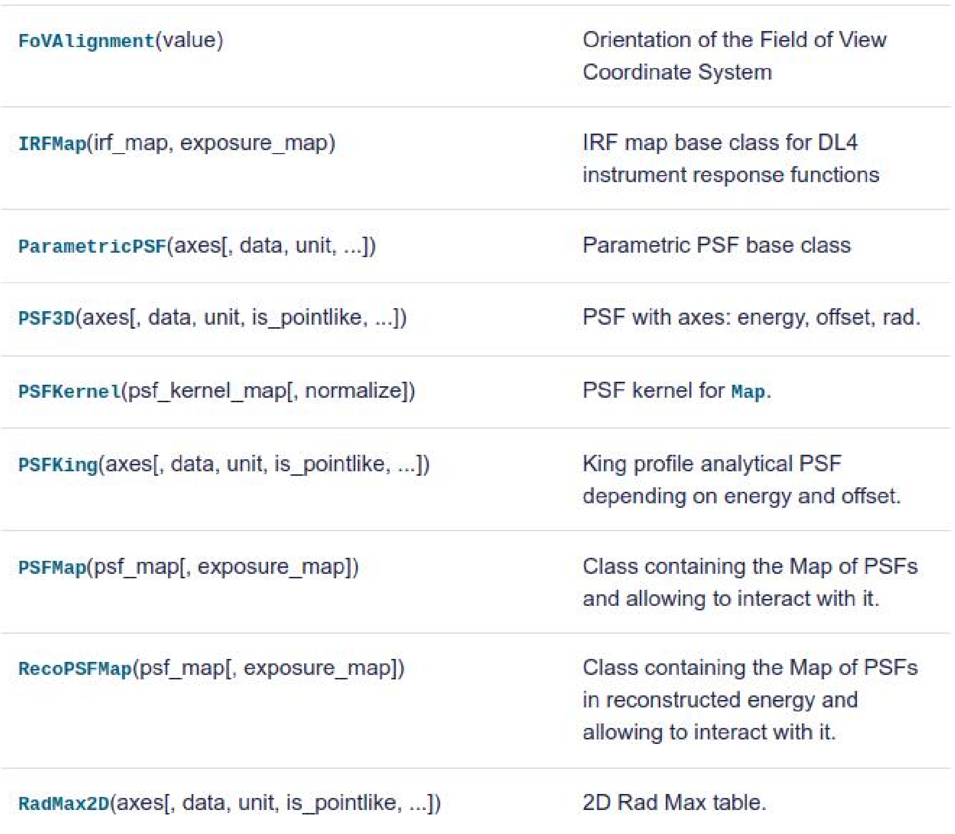


Functions

`load_cta_irfs`(filename)	❗ ***Deprecated since version v1.1.***
`load_irf_dict_from_file`(filename)	Load all available IRF components from given file into a dict.

Classes

`Background2D`(axes[, data, unit, ...])	Background 2D.
`Background3D`(axes[, data, unit, ...])	Background 3D.
`EDispKernel`(axes[, data, unit, ...])	Energy dispersion matrix.
`EDispKernelMap`(edisp_kernel_map[, exposure_map])	Energy dispersion kernel map.
`EDispMap`(edisp_map[, exposure_map])	Energy dispersion map.
`EffectiveAreaTable2D`(axes[, data, unit, ...])	2D effective area table.
`EnergyDependentMultiGaussPSF`(axes[, data, ...])	Triple Gauss analytical PSF depending on true energy and offset.
`EnergyDispersion2D`(axes[, data, unit, ...])	Offset-dependent energy dispersion matrix.

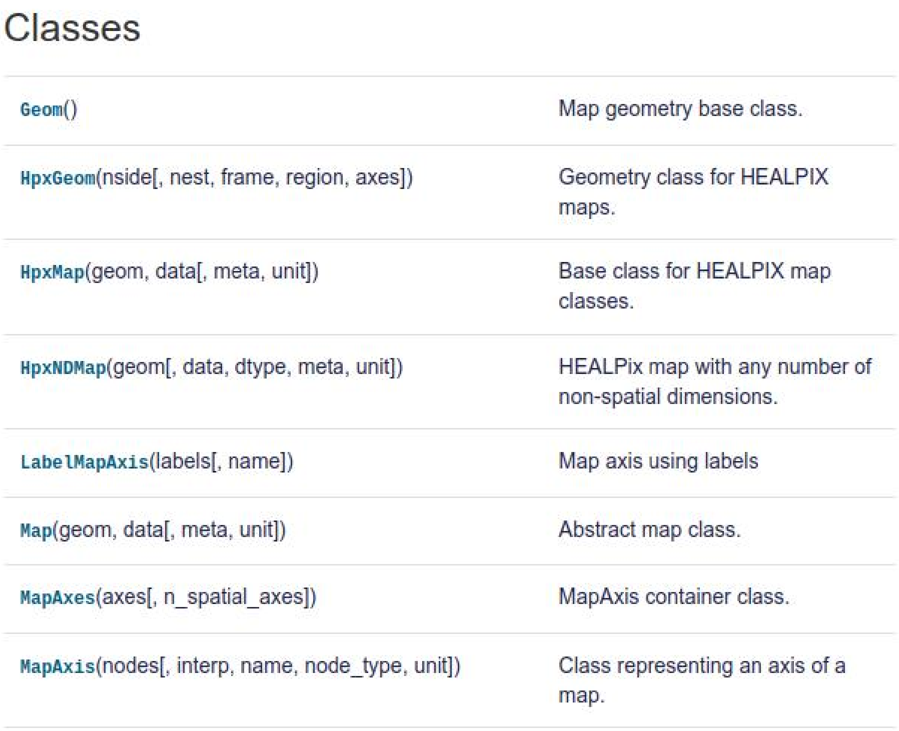


`FoVAlignment`(value)	Orientation of the Field of View Coordinate System
`IRFMap`(irf_map, exposure_map)	IRF map base class for DL4 instrument response functions
`ParametricPSF`(axes[, data, unit, ...])	Parametric PSF base class
`PSF3D`(axes[, data, unit, is_pointlike, ...])	PSF with axes: energy, offset, rad.
`PSFKernel`(psf_kernel_map[, normalize])	PSF kernel for `Map`.
`PSFKing`(axes[, data, unit, is_pointlike, ...])	King profile analytical PSF depending on energy and offset.
`PSFMap`(psf_map[, exposure_map])	Class containing the Map of PSFs and allowing to interact with it.
`RecoPSFMap`(psf_map[, exposure_map])	Class containing the Map of PSFs in reconstructed energy and allowing to interact with it.
`RadMax2D`(axes[, data, unit, is_pointlike, ...])	2D Rad Max table.

Gammapy package

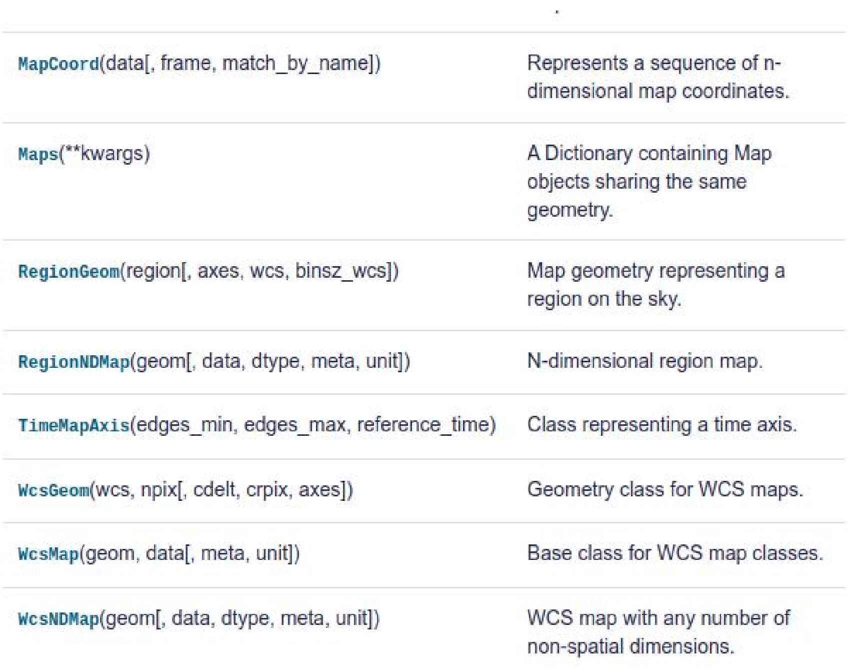
gammapy.maps

https://docs.gammapy.org/1.1/api-reference/maps.html#module-gammapy.maps



Classes

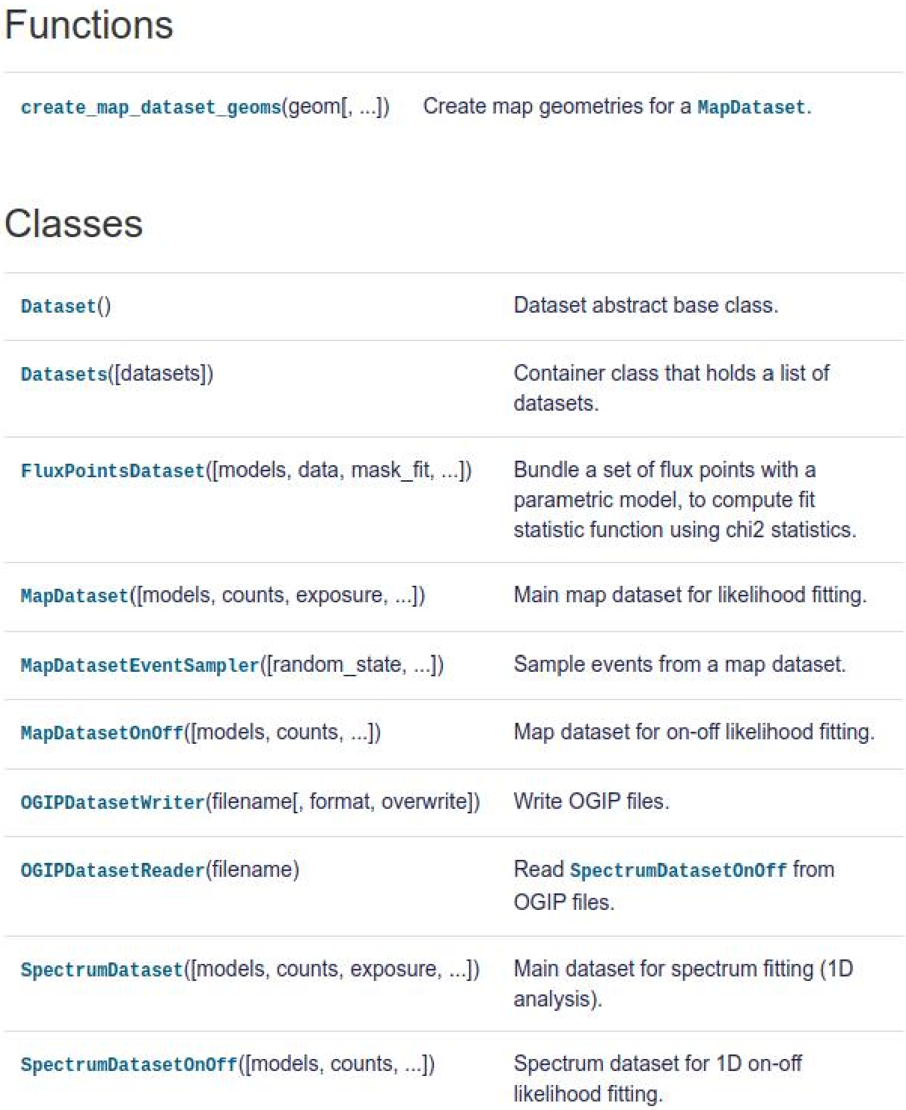
`Geom`()	Map geometry base class.
`HpxGeom`(nside[, nest, frame, region, axes])	Geometry class for HEALPIX maps.
`HpxMap`(geom, data[, meta, unit])	Base class for HEALPIX map classes.
`HpxNDMap`(geom[, data, dtype, meta, unit])	HEALPix map with any number of non-spatial dimensions.
`LabelMapAxis`(labels[, name])	Map axis using labels
`Map`(geom, data[, meta, unit])	Abstract map class.
`MapAxes`(axes[, n_spatial_axes])	MapAxis container class.
`MapAxis`(nodes[, interp, name, node_type, unit])	Class representing an axis of a map.



`MapCoord`(data[, frame, match_by_name])	Represents a sequence of n-dimensional map coordinates.
`Maps`(**kwargs)	A Dictionary containing Map objects sharing the same geometry.
`RegionGeom`(region[, axes, wcs, binsz_wcs])	Map geometry representing a region on the sky.
`RegionNDMap`(geom[, data, dtype, meta, unit])	N-dimensional region map.
`TimeMapAxis`(edges_min, edges_max, reference_time)	Class representing a time axis.
`WcsGeom`(wcs, npix[, cdelt, crpix, axes])	Geometry class for WCS maps.
`WcsMap`(geom, data[, meta, unit])	Base class for WCS map classes.
`WcsNDMap`(geom[, data, dtype, meta, unit])	WCS map with any number of non-spatial dimensions.

Gammapy package

gammapy.datasets



Functions

`create_map_dataset_geoms`(geom[, ...])	Create map geometries for a `MapDataset`.

Classes

`Dataset`()	Dataset abstract base class.
`Datasets`([datasets])	Container class that holds a list of datasets.
`FluxPointsDataset`([models, data, mask_fit, ...])	Bundle a set of flux points with a parametric model, to compute fit statistic function using chi2 statistics.
`MapDataset`([models, counts, exposure, ...])	Main map dataset for likelihood fitting.
`MapDatasetEventSampler`([random_state, ...])	Sample events from a map dataset.
`MapDatasetOnOff`([models, counts, ...])	Map dataset for on-off likelihood fitting.
`OGIPDatasetWriter`(filename[, format, overwrite])	Write OGIP files.
`OGIPDatasetReader`(filename)	Read `SpectrumDatasetOnOff` from OGIP files.
`SpectrumDataset`([models, counts, exposure, ...])	Main dataset for spectrum fitting (1D analysis).
`SpectrumDatasetOnOff`([models, counts, ...])	Spectrum dataset for 1D on-off likelihood fitting.

https://docs.gammapy.org/1.1/api-reference/datasets.html#module-gammapy.datasets