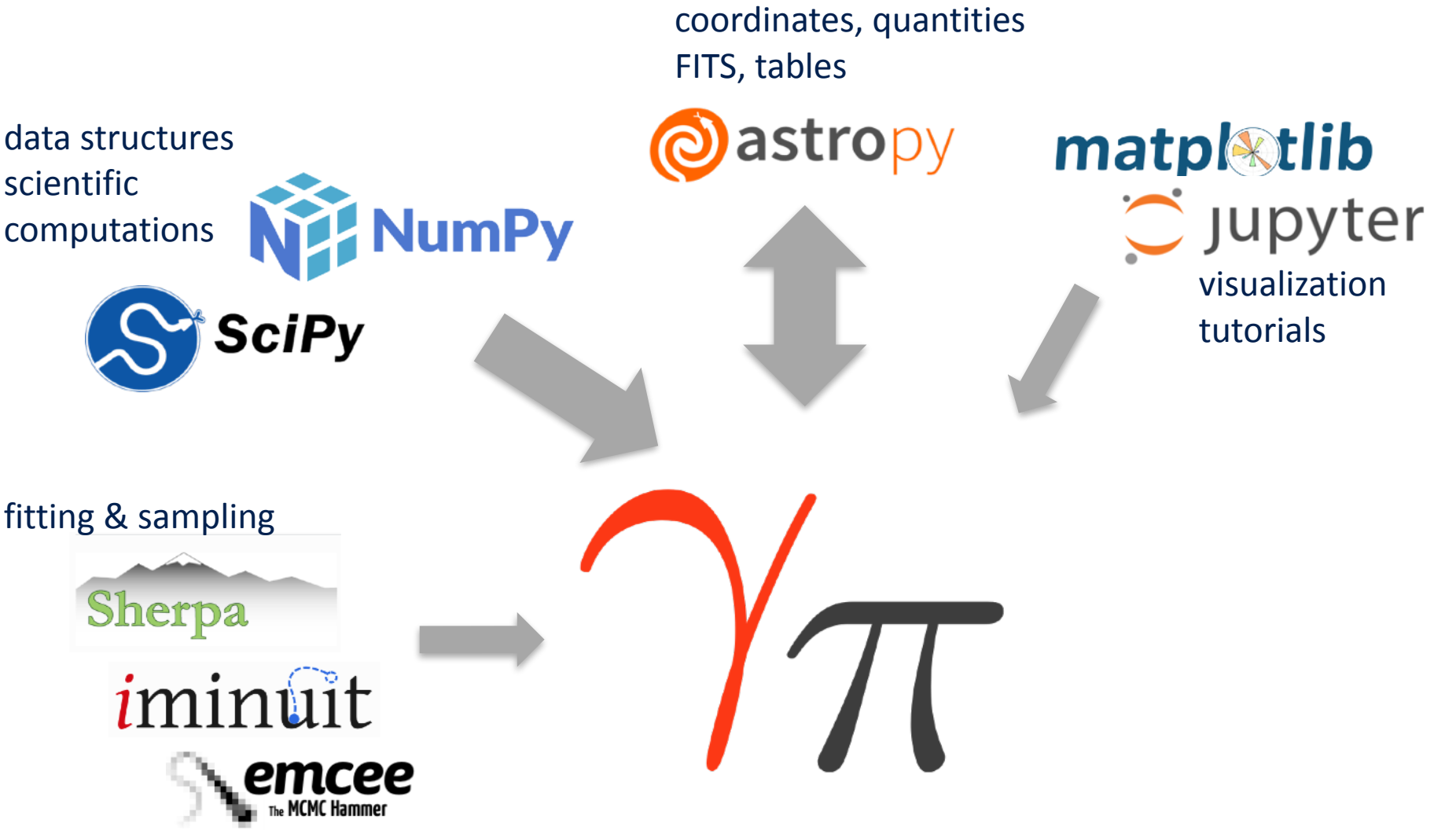
Gammapy overview

Gammapy hands-on session
CTA meeting, Bologna
May 18th 2022

Gammapy overview



v0.20 released on May 12th

Getting the software

- **Recommended gammapy installation**

```
curl -O https://gammapy.org/download/install/gammapy-0.20-environment.yml
conda env create -f gammapy-0.20-environment.yml
conda activate gammapy-0.20
```

- **Download tutorials & associated data**

```
gammapy download notebooks --release 0.20
gammapy download datasets
export GAMMAPY_DATA=$PWD/gammapy-datasets
```

Note: mamba might prove a better/faster package manager

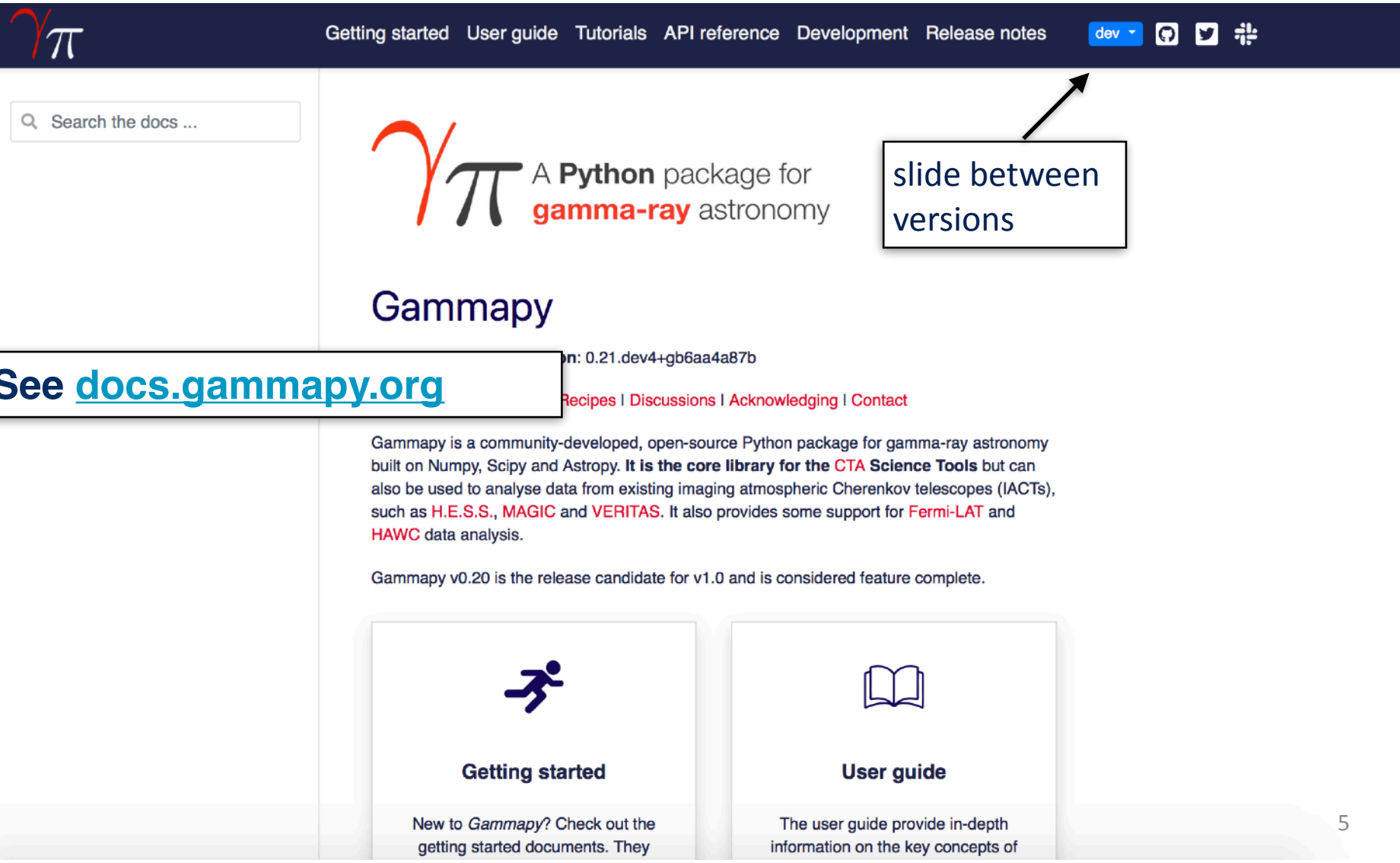
See: https://docs.gammapy.org/0.20/getting-started/index.html#quickstart-setup

What's new in v0.20?

- **Since v1.0 is LTS, rely on current astropy LTS:**
  - python >= 3.8, astropy >= 5.0, regions >= 0.5
- **New required dependencies:**
  - iminuit >= 2.8, matplotlib >= 3.4
- **Complete new documentation theme:**
  - Follows PyData theme (as scipy/numpy/pandas)
- **Main new feature:**
  - energy dependent ON region spectral analysis

See: https://docs.gammapy.org/0.20/changelog/v0.20.html

Getting started: documentation



Getting started: documentation

See docs.gammapy.org

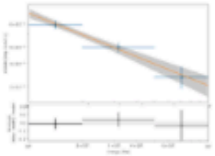
Tutorials to learn simple data analysis recipes

- spectral analysis
- lightcurve extraction
- 3D fitting
- simulation

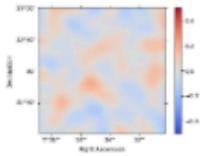
Introduction

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.

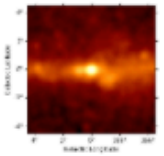
The first tutorial is an overview on how to perform a standard analysis workflow using the high level interface in a configuration-driven approach, whilst the second deals with the same use-case using the low level API and showing what is happening *under-the-hood*. The third tutorial shows a glimpse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



High level interface



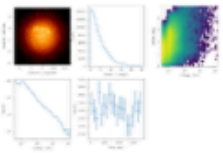
Low level API

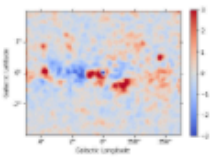


Data structures

Data exploration

These three tutorials show how to perform data exploration with Gammapy, providing an introduction to the CTA, H.E.S.S. and Fermi-LAT data and instrument response functions (IRFs). You will be able to explore and filter event lists according to different criteria, as well as to get a quick look of the multidimensional IRFs files.

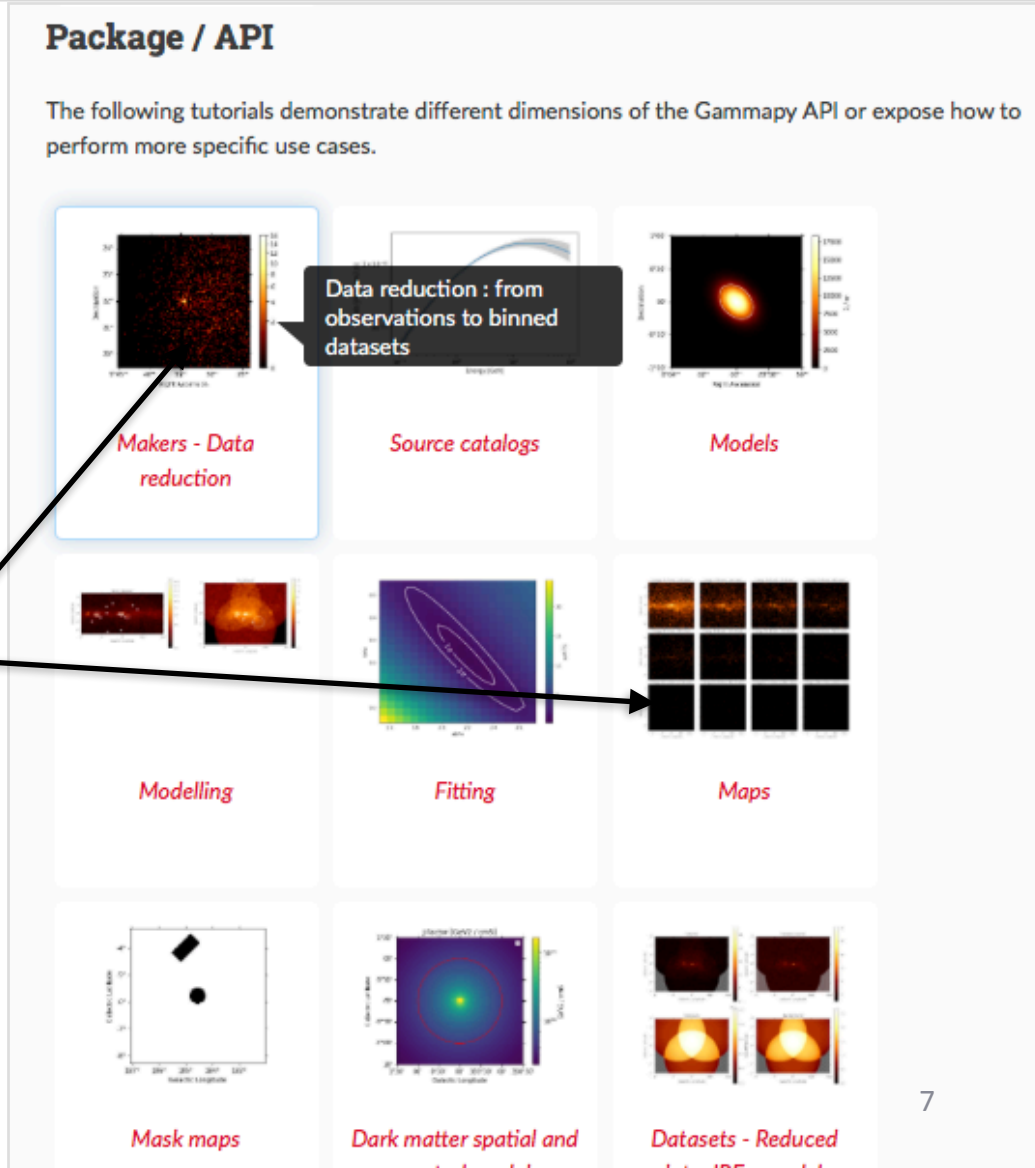




Getting started: documentation

Learn how to use the general API

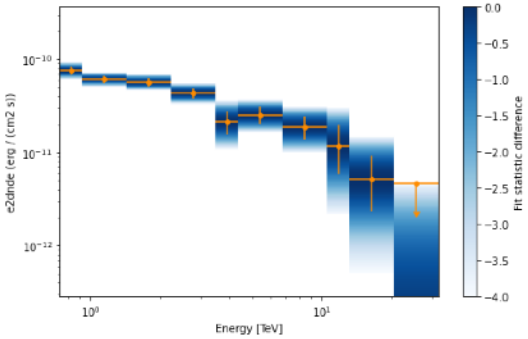
- go beyond tutorials use cases
- exploit Gammapy flexibility



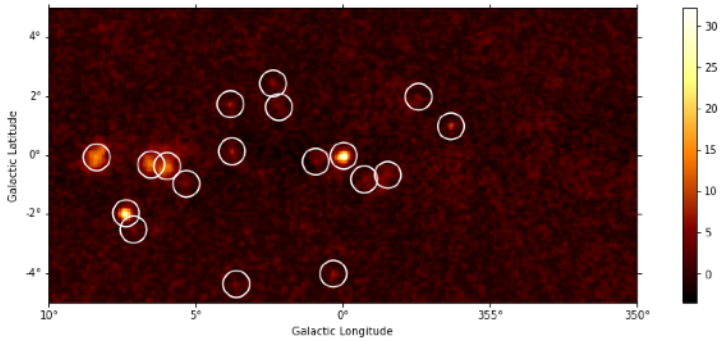
Getting help

- Where/How to interact with dev team and experienced users, provide feedback, get help:
  - gammapy.slack
    - In particular: #help channel
  - GitHub discussions
    - help category
  - GitHub issues to report bugs or feature requests

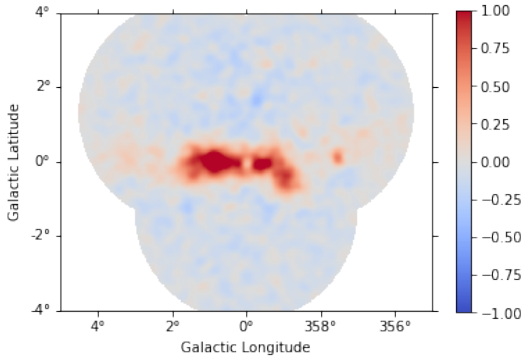
Typical analysis use cases



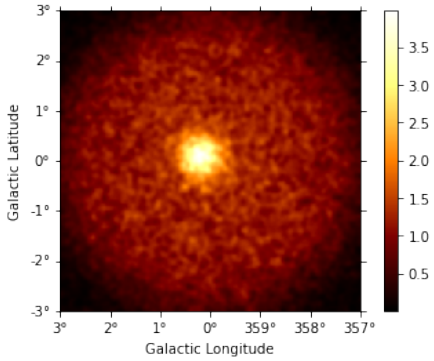
1D spectral analysis



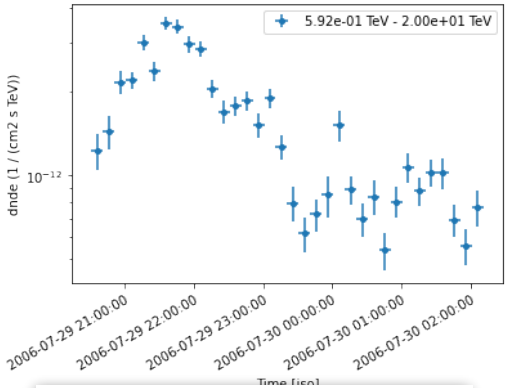
Source detection



3D analysis



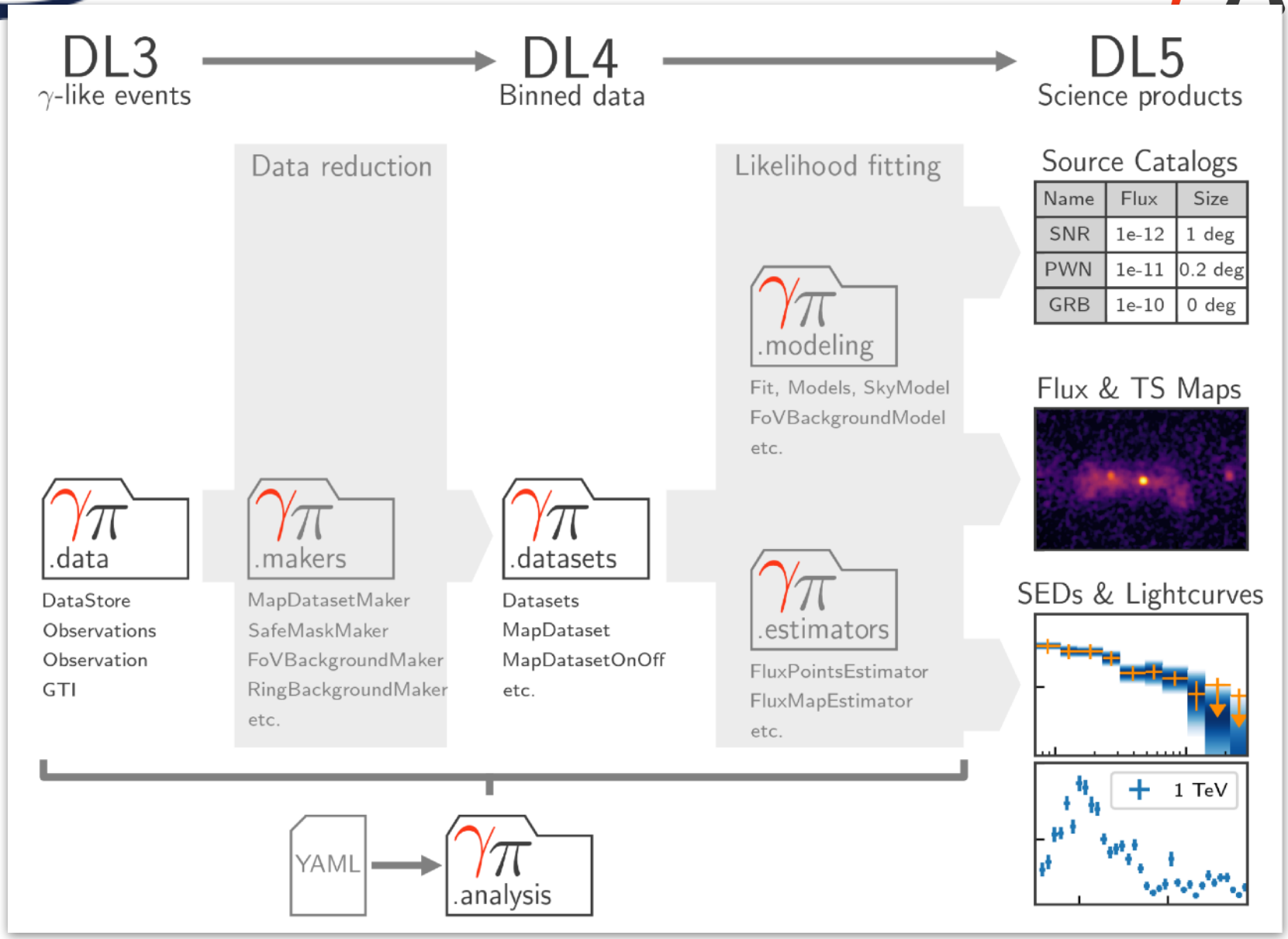
observation simulation



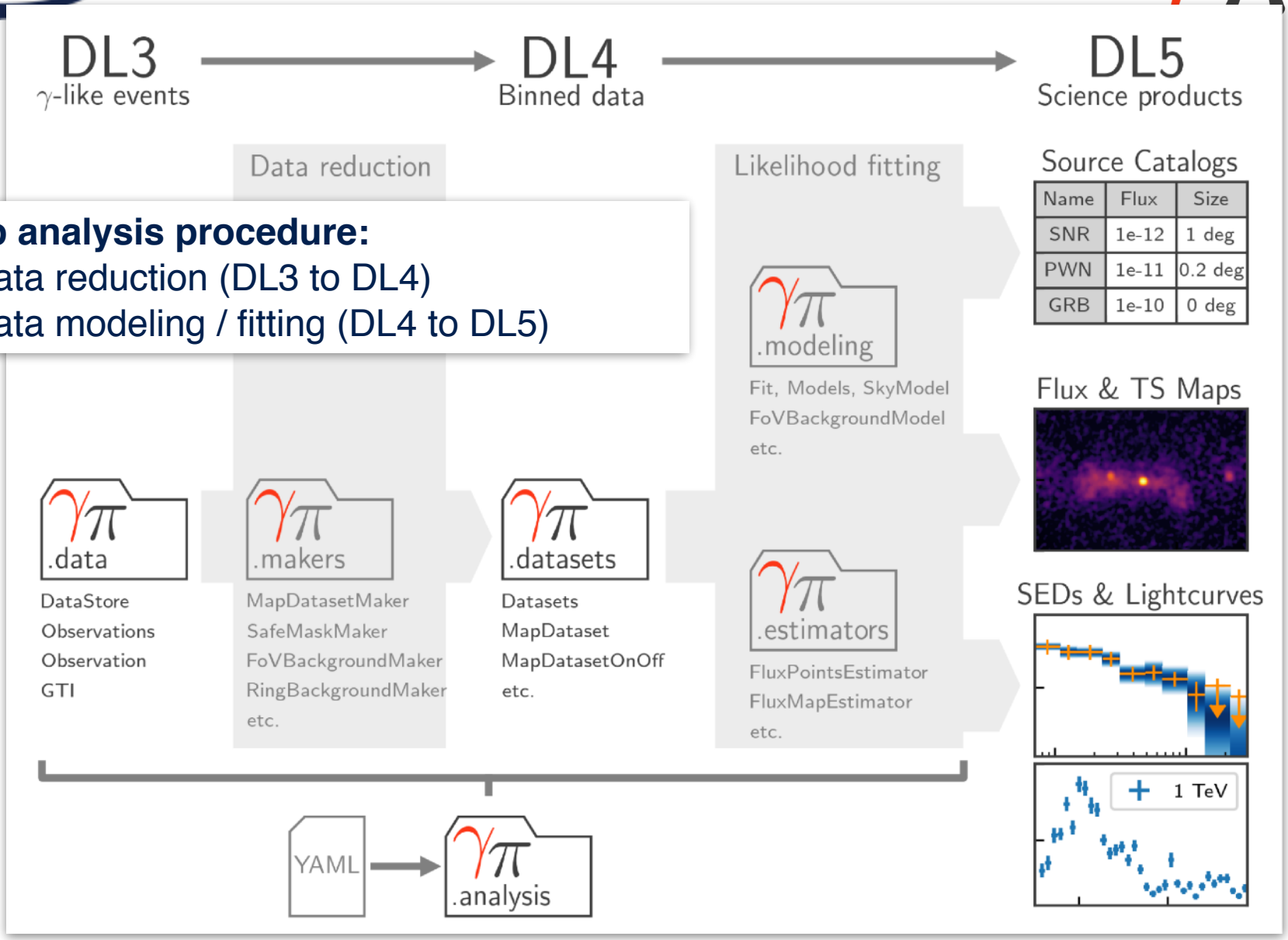
light-curve extraction

All analysis types follow the same workflow and the same API

Data workflow and package structure



Data workflow and package structure



2-step analysis procedure:

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

DL3 to DL4: data reduction

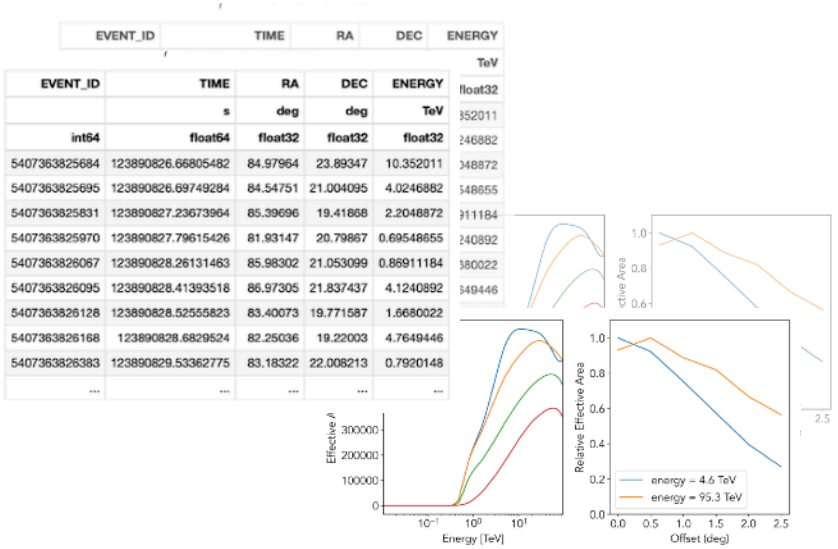
DL3

γ-like events

1. Select and retrieve relevant observations

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
5407363825970	123890827.79615426	81.93147	20.79867	0.69548655
5407363826067	123890828.26131463	85.98302	21.053099	0.86911184
5407363826095	123890828.41393518	86.97305	21.837437	4.1240892
5407363826128	123890828.52555823	83.40073	19.771587	1.6680022
5407363826168	123890828.6829524	82.25036	19.22003	4.7649446
5407363826383	123890829.53362775	83.18322	22.008213	0.7920148
...	...	...	...	...

DataStore



Observation / Observations

DL3 to DL4: data reduction

DL3

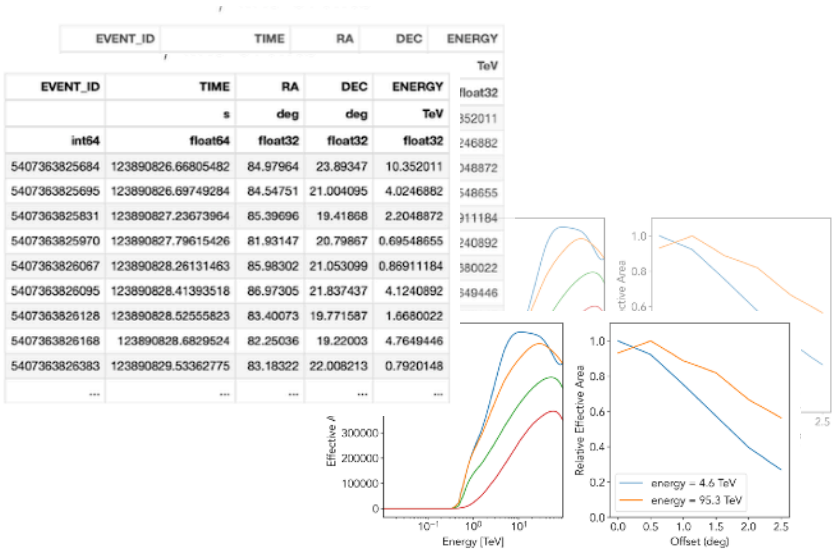
γ-like events

1. Select and retrieve relevant observations

```
datastore = DataStore.from_dir("$GAMMAPY_DATA/hess-dl3-dr1/")
obs_ids = [23523, 23526, 23559, 23592]
observations = datastore.get_observations(obs_ids)
```

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
5407363825970	123890827.79615426	81.93147	20.79867	0.69548655
5407363826067	123890828.26131463	85.98302	21.053099	0.86911184
5407363826095	123890828.41393518	86.97305	21.837437	4.1240892
5407363826128	123890828.52555823	83.40073	19.771587	1.6680022
5407363826168	123890828.6829524	82.25036	19.22003	4.7649446
5407363826383	123890829.53362775	83.18322	22.008213	0.7920148
...	...	...	...	...

DataStore



Observation / Observations

DL3 to DL4: data reduction

DL3

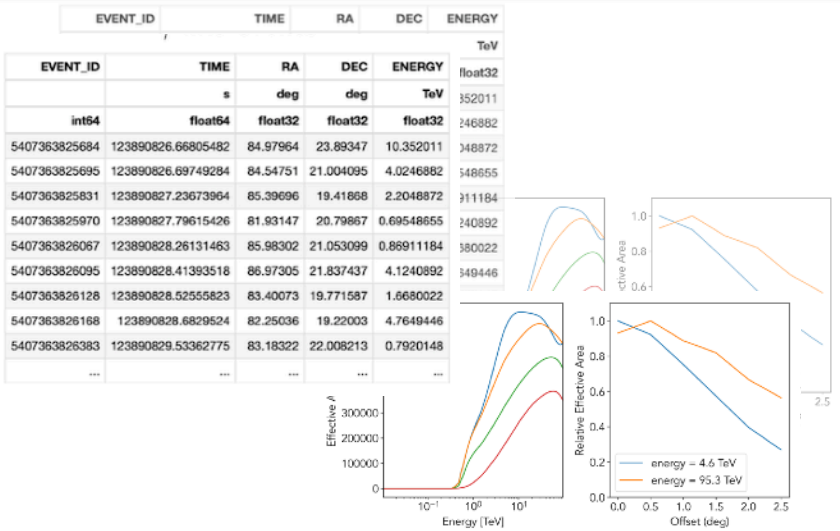
γ-like events

1. Select and retrieve relevant observations

```
# Create an in-memory observation
location = observatory_locations["cta_south"]
obs = Observation.create(
    pointing=pointing, livetime=livetime, irfs=irfs, location=location
)
```

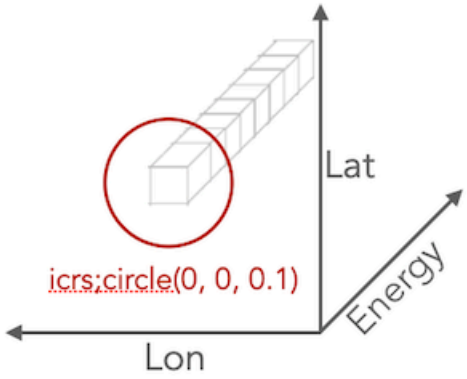
EVENT_ID	TIME	RA	DEC	ENERGY
	s	deg	deg	TeV
int64	float64	float32	float32	float32
5407363825684	123890826.66805482	84.97964	23.89347	10.352011
5407363825695	123890826.69749284	84.54751	21.004095	4.0246882
5407363825831	123890827.23673964	85.39696	19.41868	2.2048872
5407363825970	123890827.79615426	81.93147	20.79867	0.69548655
5407363826067	123890828.26131463	85.98302	21.053099	0.86911184
5407363826095	123890828.41393518	86.97305	21.837437	4.1240892
5407363826128	123890828.52555823	83.40073	19.771587	1.6680022
5407363826168	123890828.6829524	82.25036	19.22003	4.7649446
5407363826383	123890829.53362775	83.18322	22.008213	0.7920148
...	...	...	...	...

DataStore

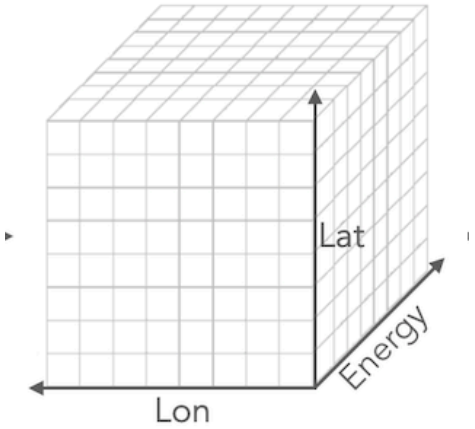


Observation / Observations

DL3 to DL4: data reduction



region & energy



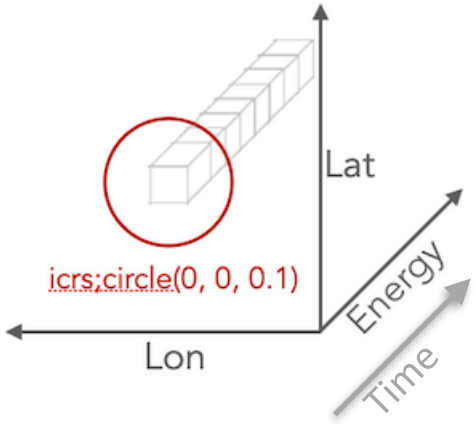
WCS & energy

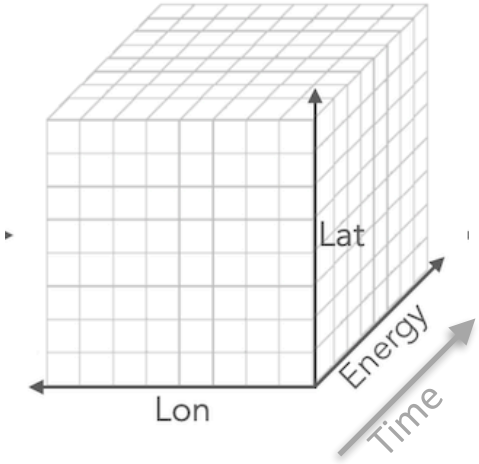
1. Select and retrieve relevant observations

2. Define the reduced dataset geometry
   - Is the analysis 1D (spectral only) or 3D?
   - Define target binning and projection

Geometry : multidimensional maps

- Gammapy maps represent data on the sky with non-spatial dimensions (in particular energy)
  - World Coord. System (WCS) for 3D analyses (lon, lat, E)
  - Region geometry for 1D analysis





MapAxis

RegionGeom / RegionNDMap WcsGeom / WcsNDMap

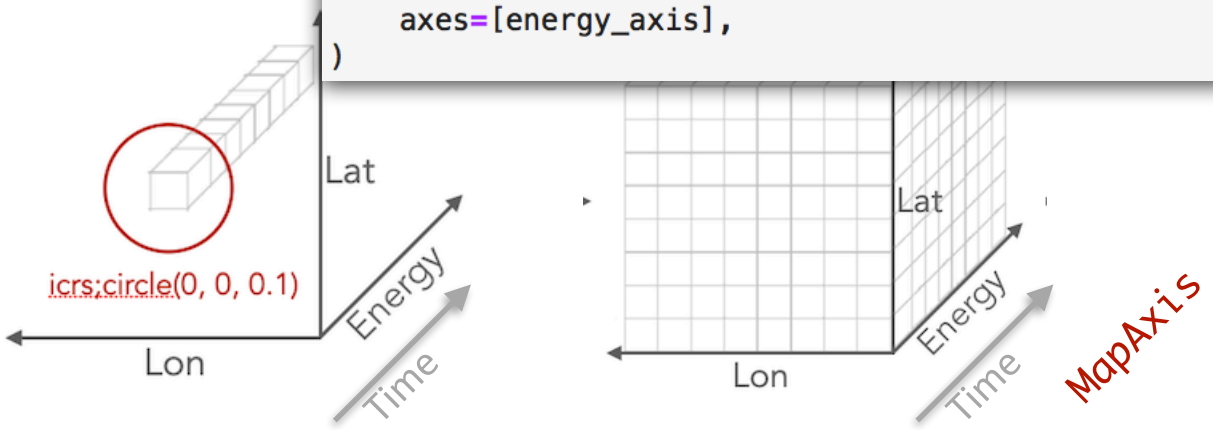
See : working with maps

Geometry : multidimensional maps

- Gammapy maps represent data on the sky with non-spatial dimensions (in particular energy)
  - World Coord. System (WCS) for 3D analyses (lon, lat, E)
  - Region g

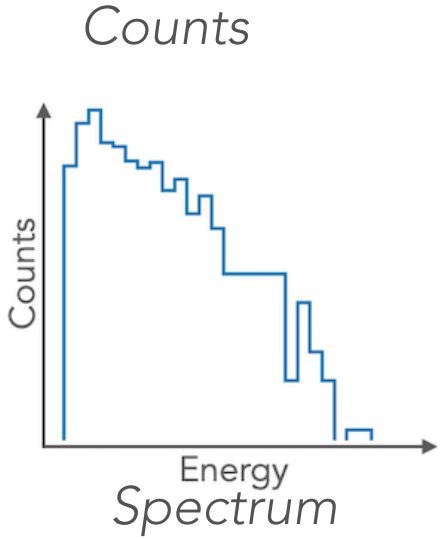
```
energy_axis = MapAxis.from_energy_bounds(
    "0.02 TeV", "100 TeV", nbin=10, per_decade=True
)

geom = WcsGeom.create(
    skydir=pointing,
    width=(12, 12),
    binsz=0.02,
    frame="galactic",
    axes=[energy_axis],
)
```

orking with maps



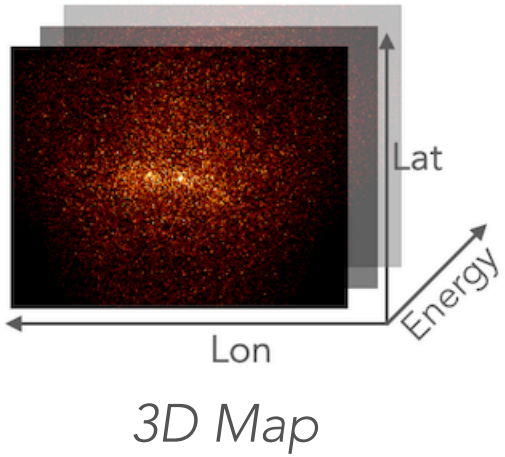
RegionGeom / RegionNDMap WcsGeom / WcsNDMap

DL3 to DL4: data reduction

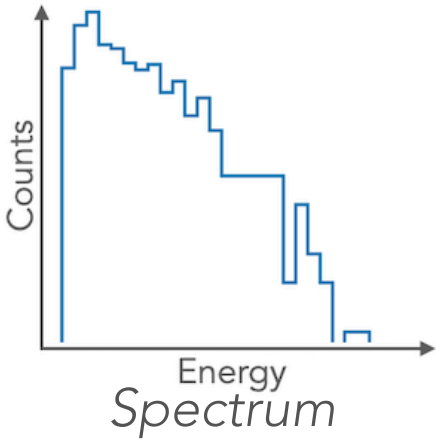


1. Select and retrieve relevant observations
2. Define the reduced dataset geometry
3. Initialize the data reduction methods (makers)
   - Data and IRF projection
   - Background estimation
   - Safe Mask determination

`MapDatasetMaker / SpectrumDatasetMaker`

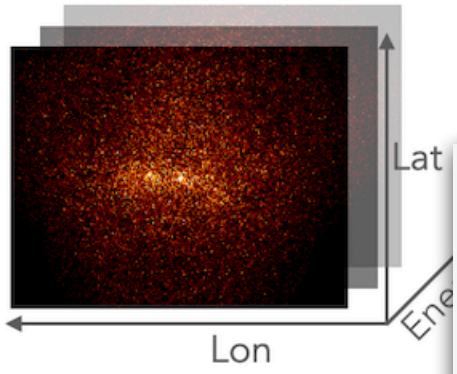


DL3 to DL4: data reduction



1. Select and retrieve relevant observations
2. Define the reduced dataset geometry
3. Initialize the data reduction methods (makers)
   - Data and IRF projection
   - Background estimation
   - Safe Mask determination

MapDatasetMaker / SpectrumDatasetMaker



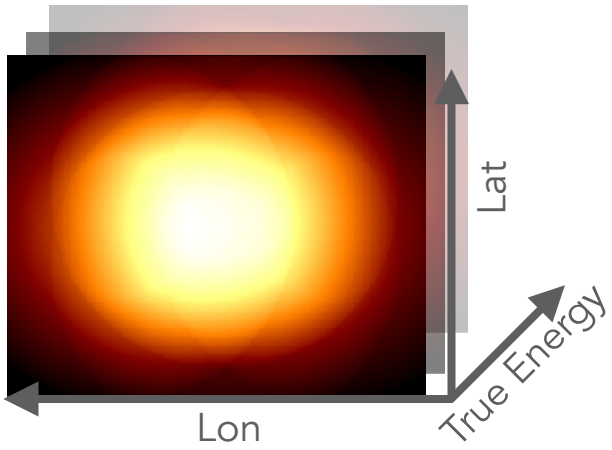
3D Map

```
empty = MapDataset.create(
    geom,
    energy_axis_true=energy_axis_true,
    migra_axis=migra_axis,
    name="my-dataset",
)
maker = MapDatasetMaker(selection=["exposure", "background", "psf", "edisp"])
dataset = maker.run(empty, observation)
```

Reduced IRFs

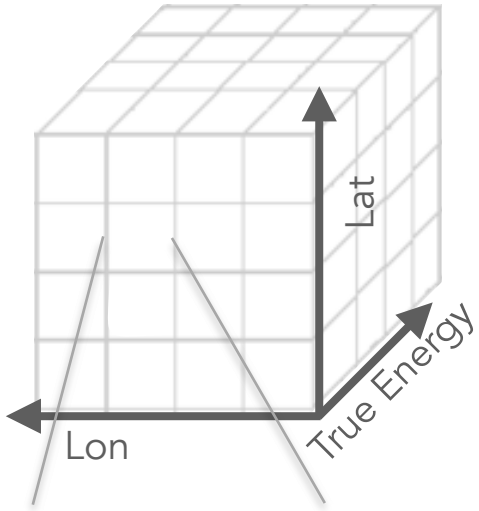
- DL3 IRFs are reprojected by the DatasetMaker on the target geometry

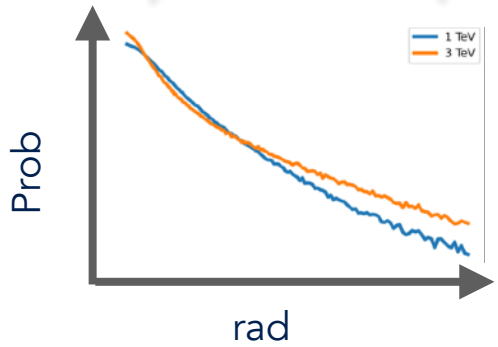
Exposure



PSF

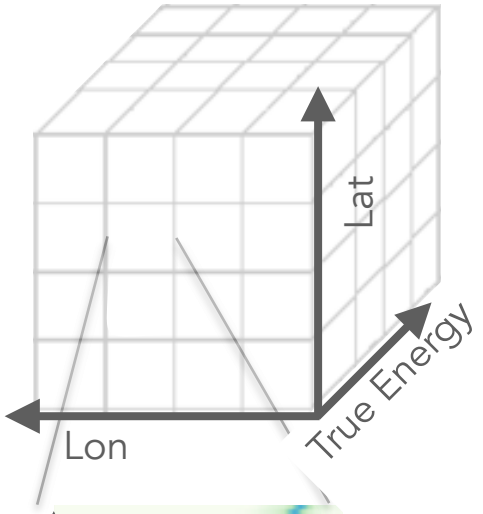
4th Dimension: rad

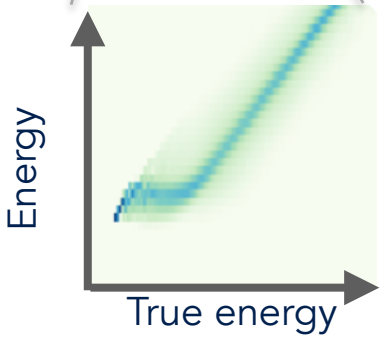




EDisp

4th Dimension: Energy

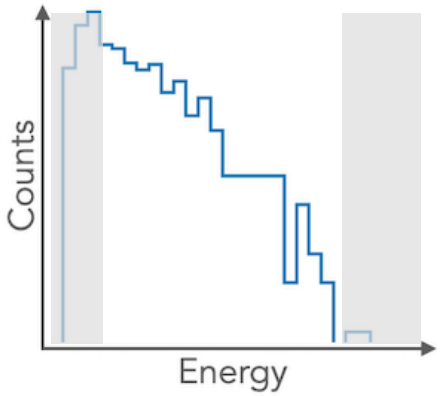




`PSFMap / EDispKernelMap`

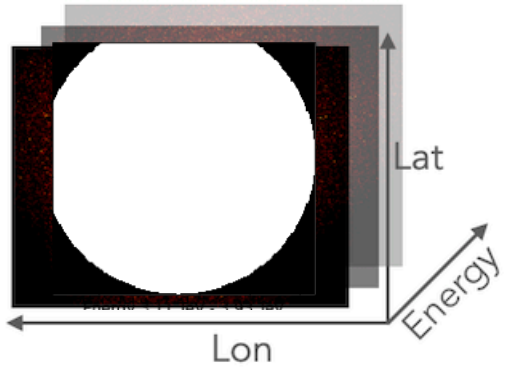
DL3 to DL4: data reduction

Safe Mask



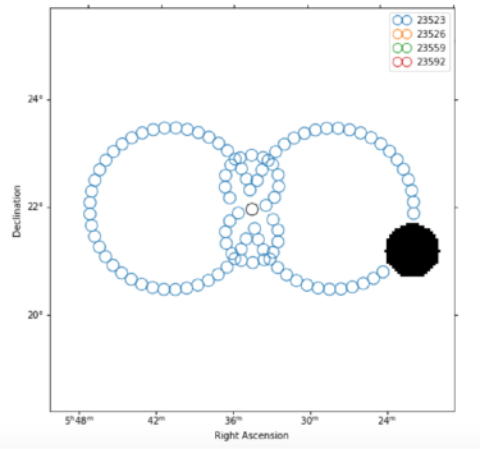
1. Select and retrieve relevant observations
2. Define the reduced dataset geometry
3. Initialize the data reduction methods (makers)
   - Data and IRF projection
   - Safe Mask determination
   - Background estimation

`SafeMaskMaker`

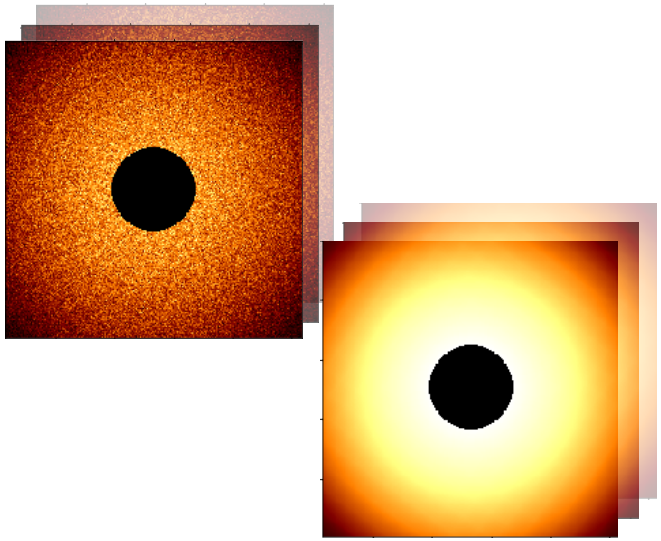


DL3 to DL4: data reduction

Background estimation

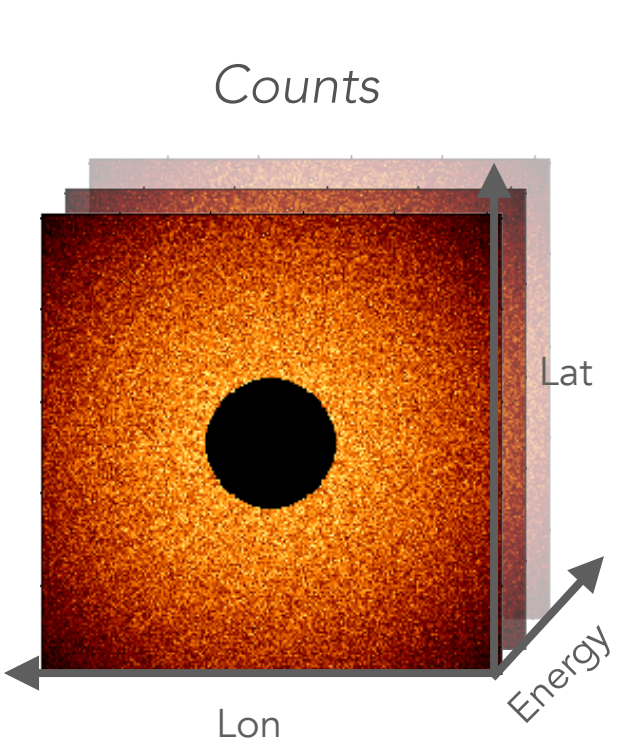


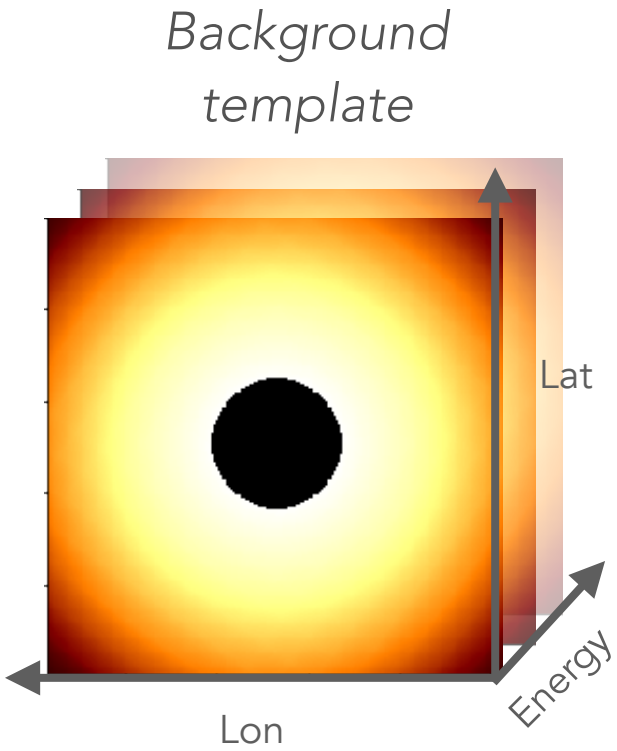
1. Select and retrieve relevant observations
2. Define the reduced dataset geometry
3. Initialize the data reduction methods (makers)
   - Data and IRF projection
   - Safe Mask determination
   - Background estimation

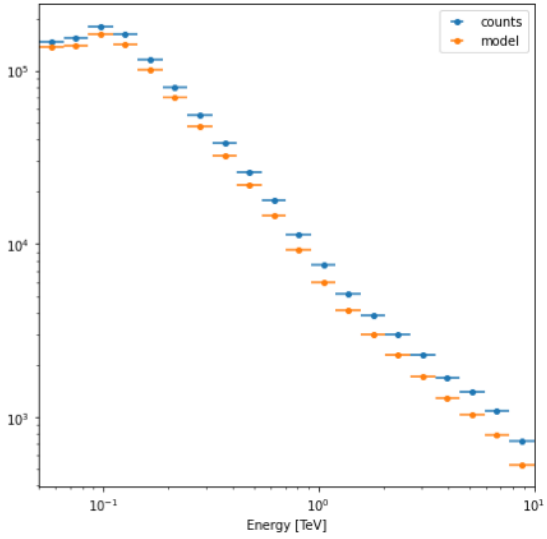


Correcting background from data

- $BKG(p, E)$ is usually corrected on the observed data themselves.
  - Field of View (FoV) background estimation
    - $BKG(p, E)$ is normalized in regions devoid of signal



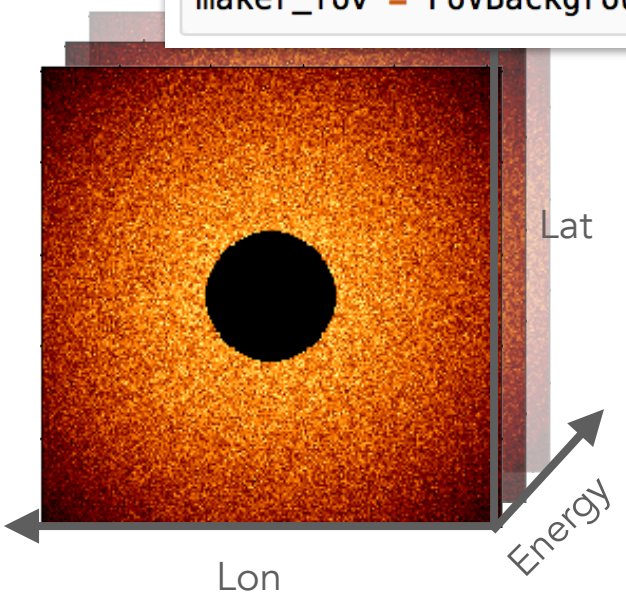


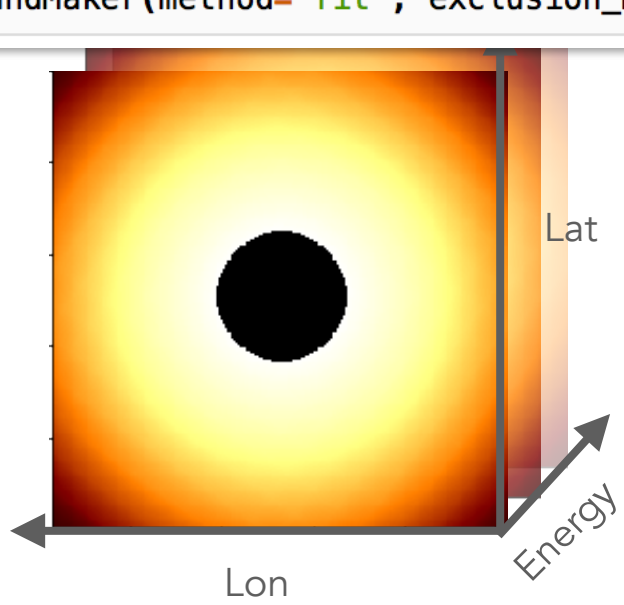


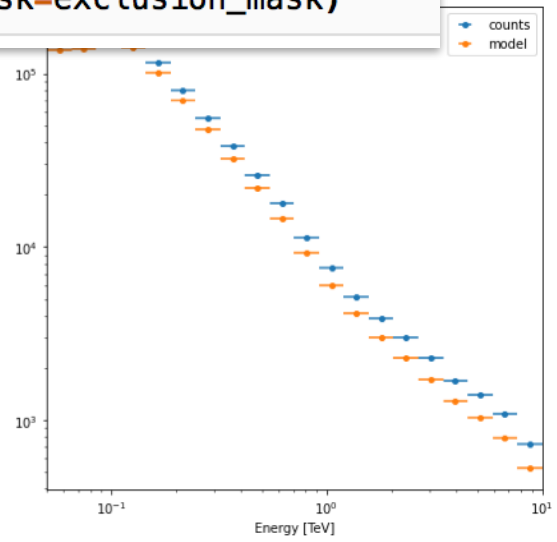
Correcting background from data

- $BKG(p, E)$ is usually corrected on the observed data themselves.
  - Field of View (FoV) background estimation
    - $BKG(p, E)$ is normalized in regions devoid of signal

```
circle = CircleSkyRegion(
    center=SkyCoord("83.63 deg", "22.14 deg"), radius=0.2 * u.deg
)
exclusion_mask = ~geom.region_mask(regions=[circle])
maker_fov = FoVBackgroundMaker(method="fit", exclusion_mask=exclusion_mask)
```



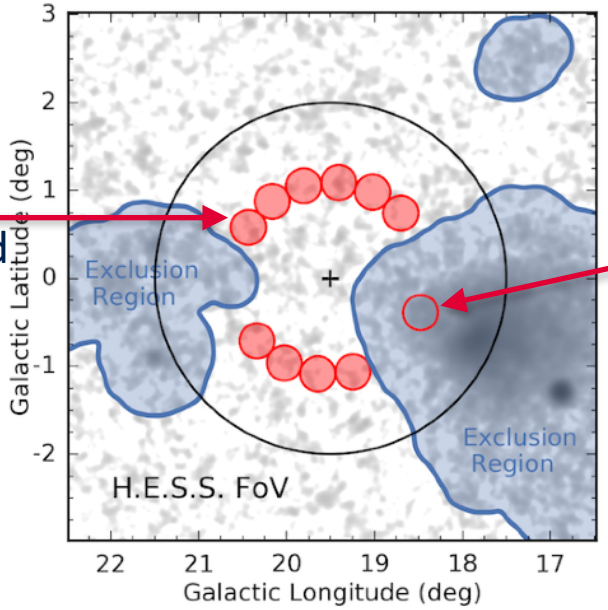




Correcting background from data

- To further reduce systematics, the background is sometimes measured directly in the data e.g. in regions of the FoV where the background is assumed to be identical
  - Common approach used for 1D spectral analysis
  - e.g. reflected regions or wobble regions background

OFF regions
containing only background



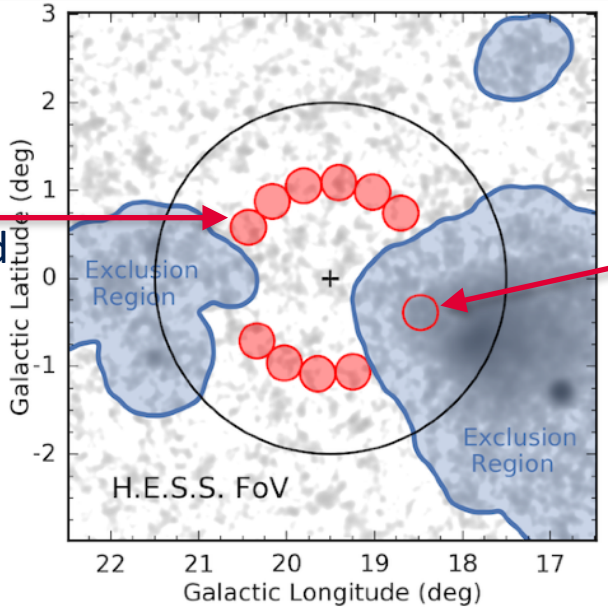
ON region
containing signal and background

Correcting background from data

- To further reduce systematics, the background is sometimes measured directly in the data e.g. in regions of the FoV where the background is assumed to be identical
  - Common approach used for 1D spectral analysis

```
bkg_maker = ReflectedRegionsBackgroundMaker(exclusion_mask=exclusion_mask)
```

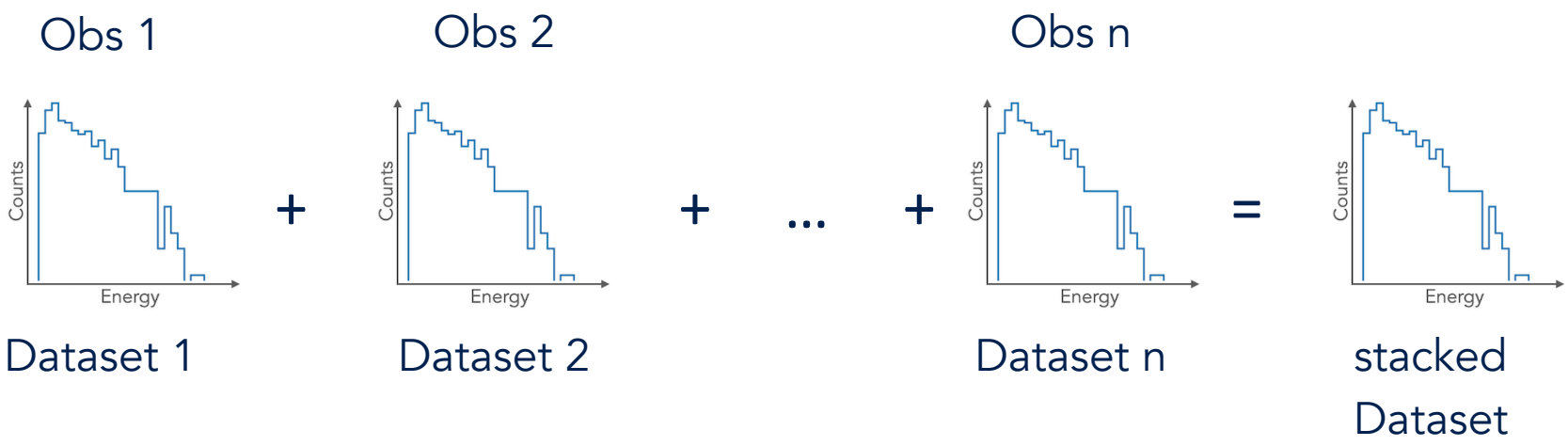
OFF regions
containing only background



ON region
containing signal and background

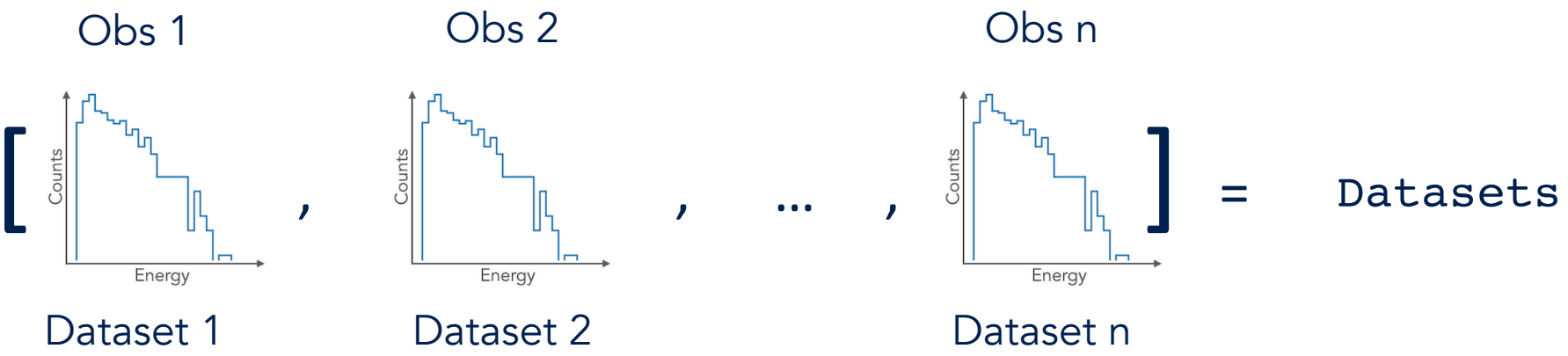
DL3 to DL4: data reduction

1. Select and retrieve relevant observations
2. Define the reduced dataset geometry
3. Initialize the data reduction methods (makers)
4. Loop over selected observations
   - Apply makers to produce reduced datasets
   - Combine them for stacked or joint analysis

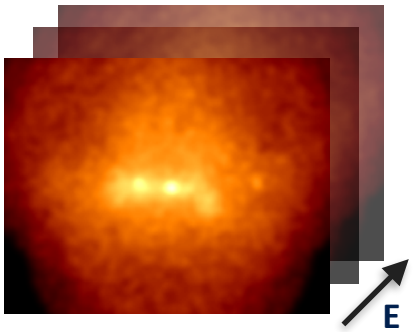


DL3 to DL4: data reduction

1. Select and retrieve relevant observations
2. Define the reduced dataset geometry
3. Initialize the data reduction methods (makers)
4. Loop over selected observations
   - Apply makers to produce reduced datasets
   - Combine them for stacked or joint analysis

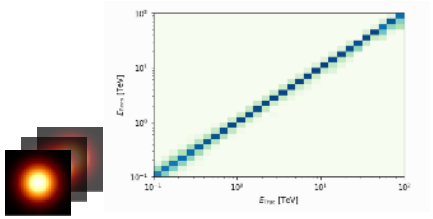


DL4 structures: Datasets



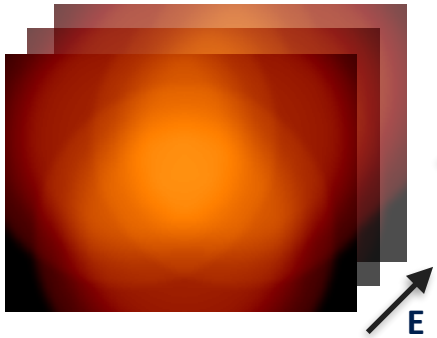
Reduced data

I/O



Reduced IRFs

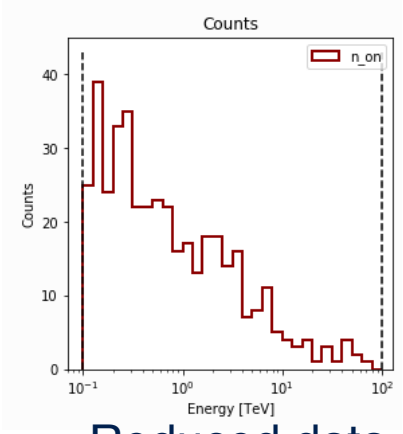
MapDataset



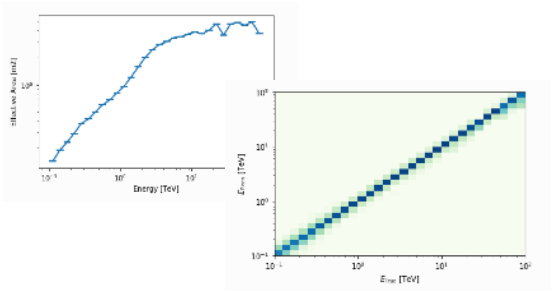
Bkg model

see: Dataset API tutorial

DL4 structures: Datasets



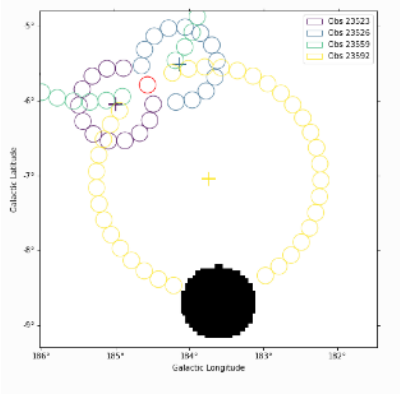
Reduced data



Reduced IRFs

I/O

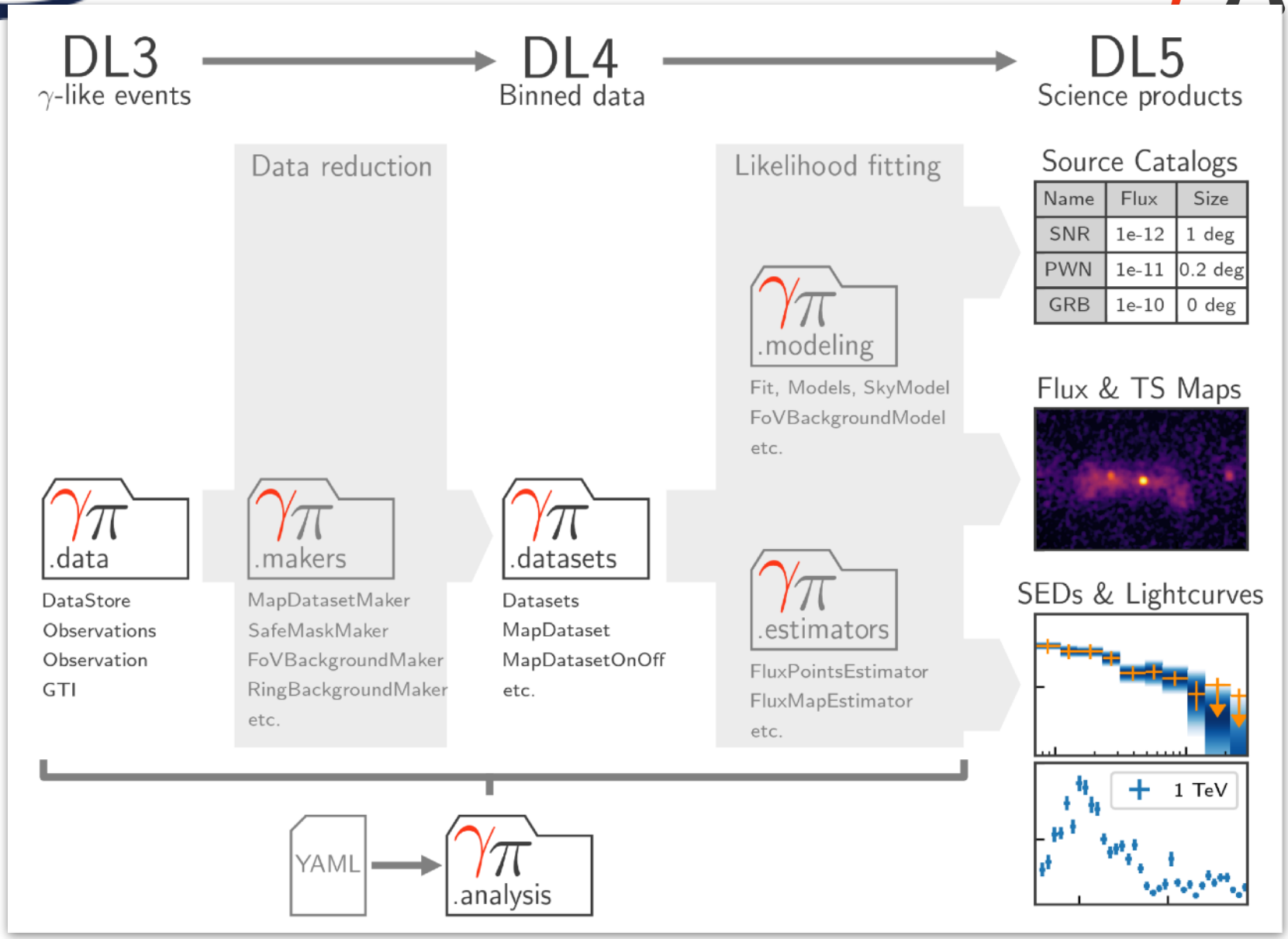
`SpectrumDataset`



Bkg data or model

see: Dataset API tutorial

Data workflow and package structure



DL4 to DL5: Modeling and fitting

- For modeling and fitting, Gammapy relies on ***forward-folding***:
  - Measured counts N is compared to predicted counts N_{pred}

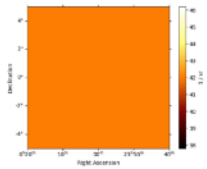
$$N_{\mathrm{pred}}(p, E) = \sum_S E_{disp} \left[PSF \star \left(expo \times \Phi_S(p_t, E_t) \right) \right] + N_{\mathrm{bkg}}(p, E)$$

- Model parameter estimation is performed through maximum likelihood technique.
  - Cash statistics is used for counts data with a known background

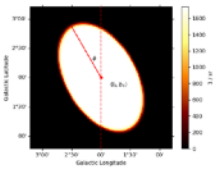
$$TS = -2 \log L = 2 \sum \left(N \log N_{\mathrm{pred}} - N_{\mathrm{pred}} \right)$$

  - Wstat statistics is used for counts data with a measured background

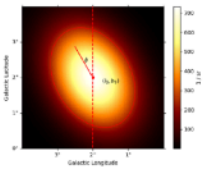
Datasets modeling and fitting



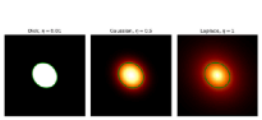
Constant spatial model



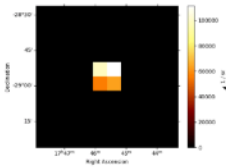
Disk spatial model



Gaussian spatial model

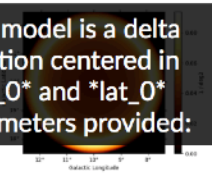


Generalized gaussian spatial model

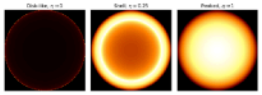


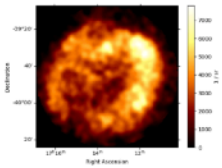
This model is a delta function centered in *lon_0* and *lat_0* parameters provided:

Point spatial model

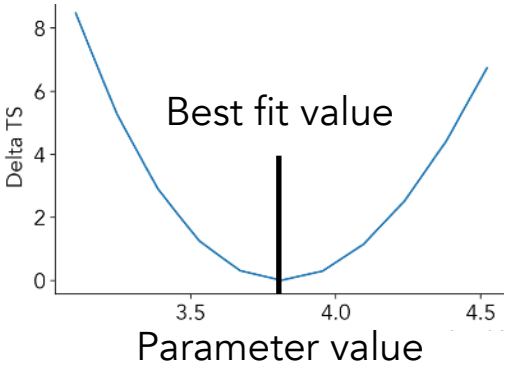


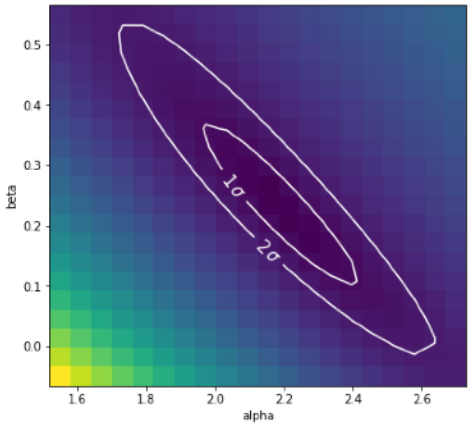
Shell spatial model



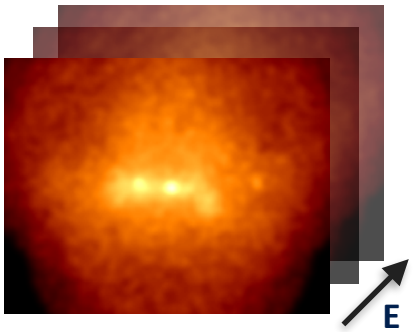


A library of models and a Fitting interface



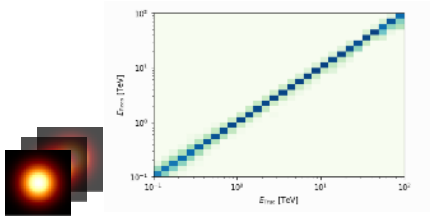


Datasets modeling and fitting



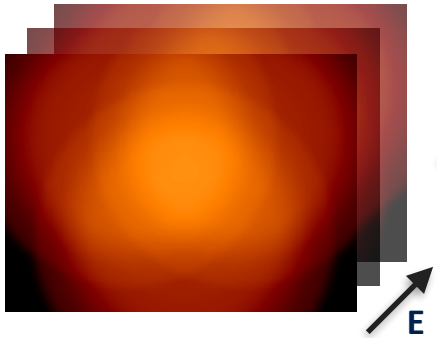
Reduced data

I/O



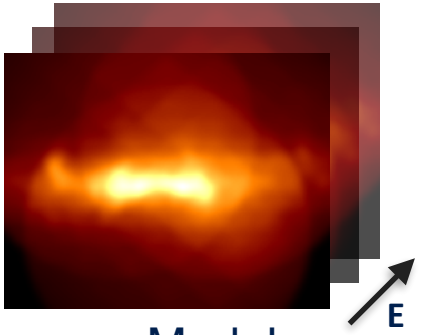
Reduced IRFs

Dataset



Bkg model

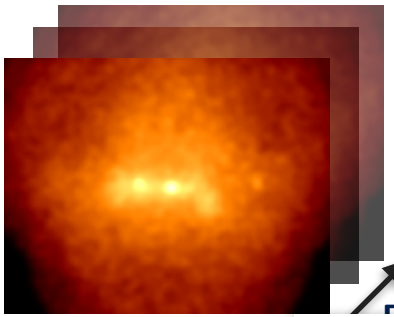
N_{pred}



Model

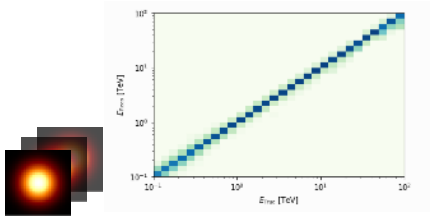
see:Dataset simulation tutorial

Datasets modeling and fitting



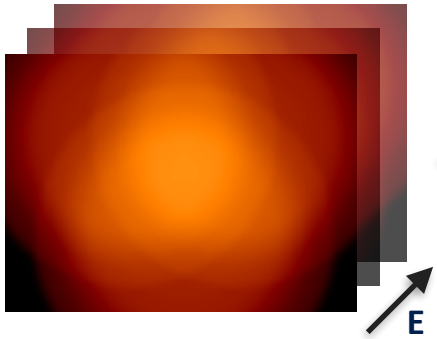
Reduced d

I/O



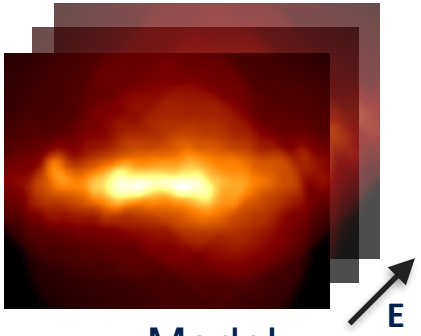
Reduced IRFs

```
# Add the model on the dataset and Poission fluctuate
dataset.models = models
dataset.fake()
# Do a print on the dataset - there is now a counts maps
print(dataset)
```



Bkg model

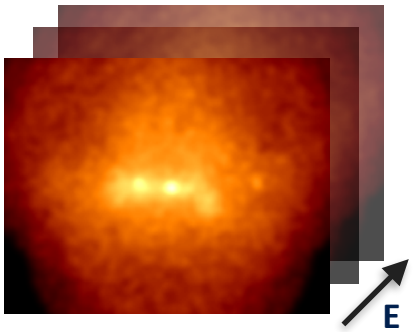
N_{pred}



Model

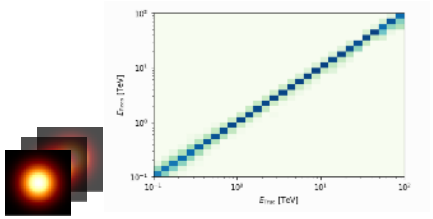
see:Dataset simulation tutorial

Datasets modeling and fitting



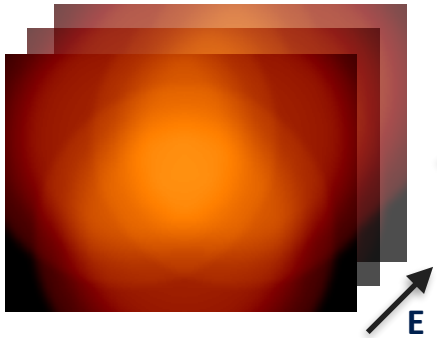
Reduced data

I/O



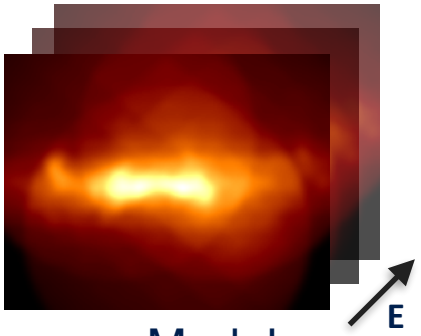
Reduced IRFs

Dataset



Bkg model

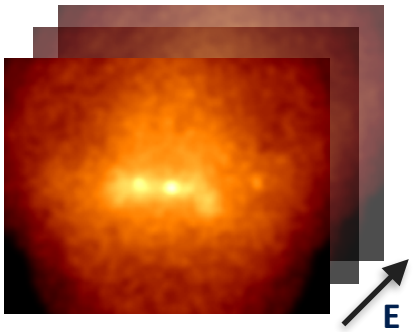
$$\mathscr{L}\left(x \mid \theta\right)$$



Model

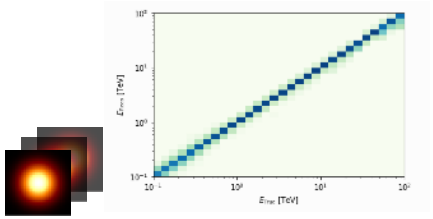
see: Dataset fitting tutorial

Datasets modeling and fitting



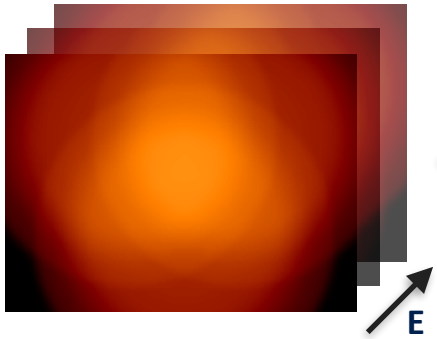
Reduced data

I/O



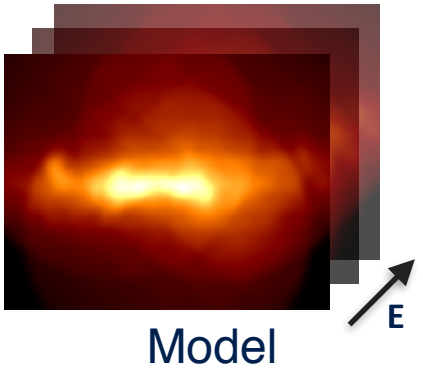
Reduced IRFs

```
fit = Fit(optimize_opts={"print_level": 1})
result = fit.run(datasets=[dataset])
```



Bkg model

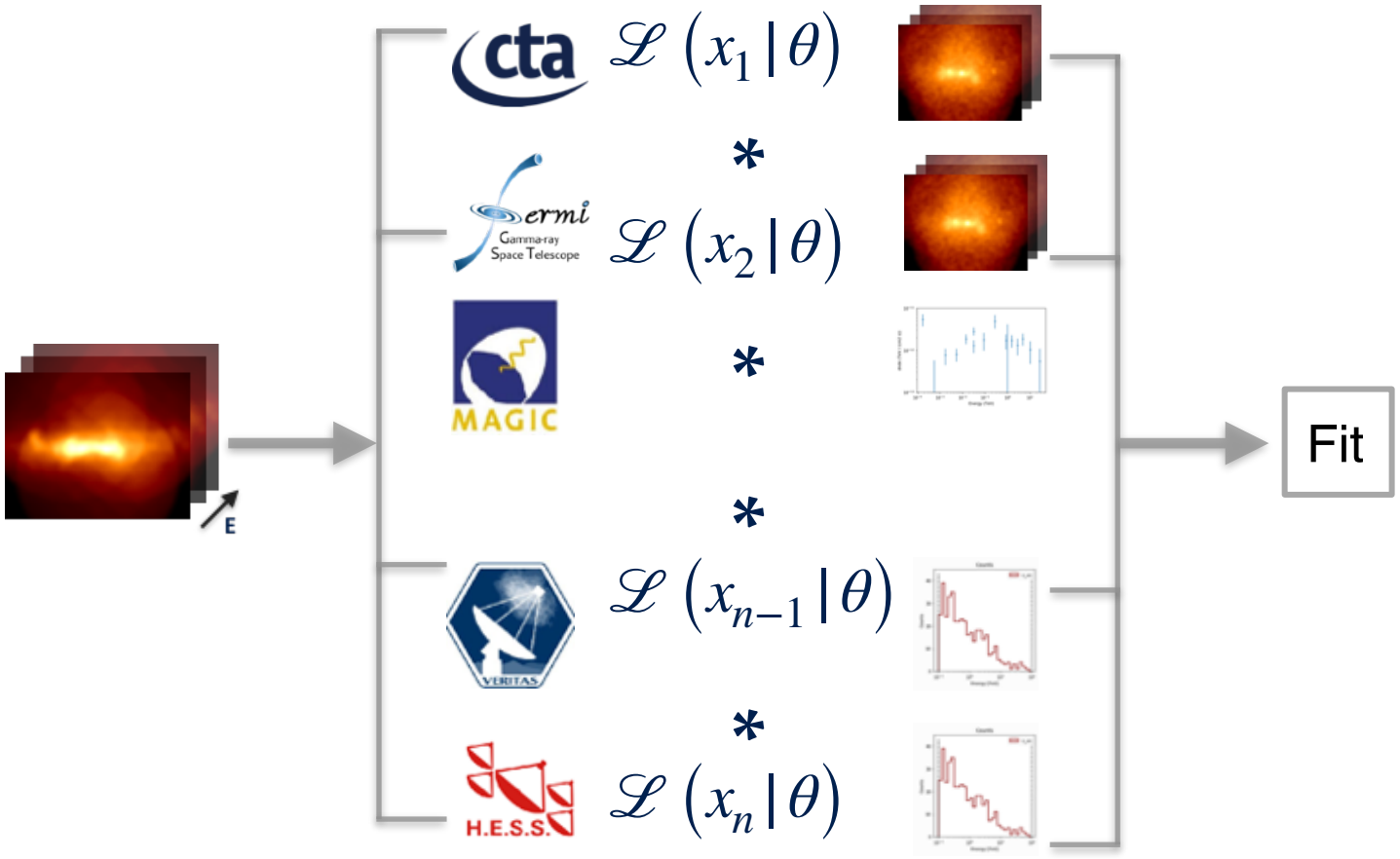
$$\mathscr{L}\left(x \mid \theta\right)$$



Model

see: Dataset fitting tutorial

DL4 to DL5 : Joint fitting

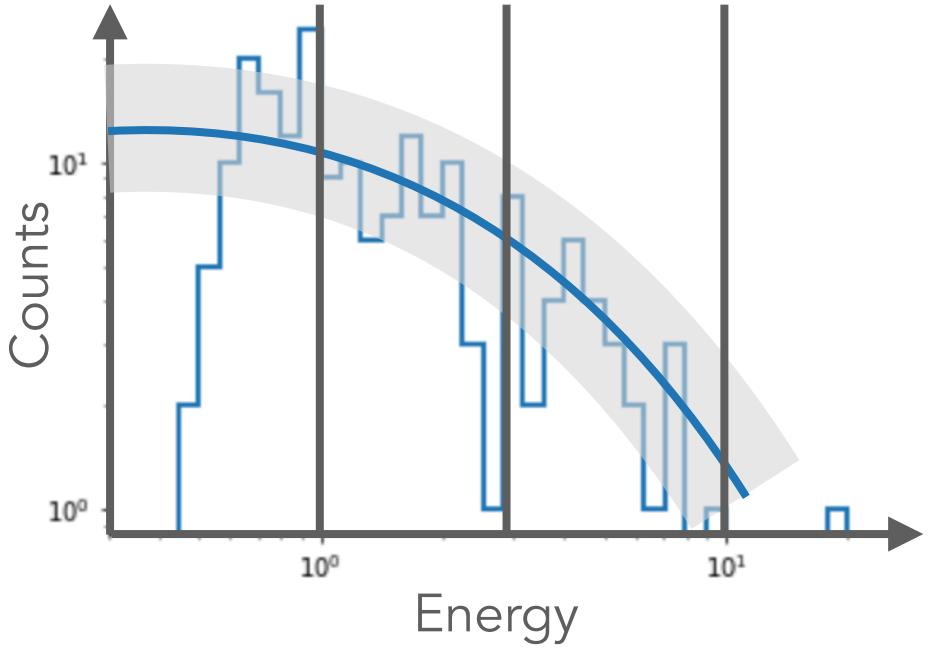


Gammapy Dataset structure allows heterogeneous data modeling and fitting:

- See joint fit tutorial

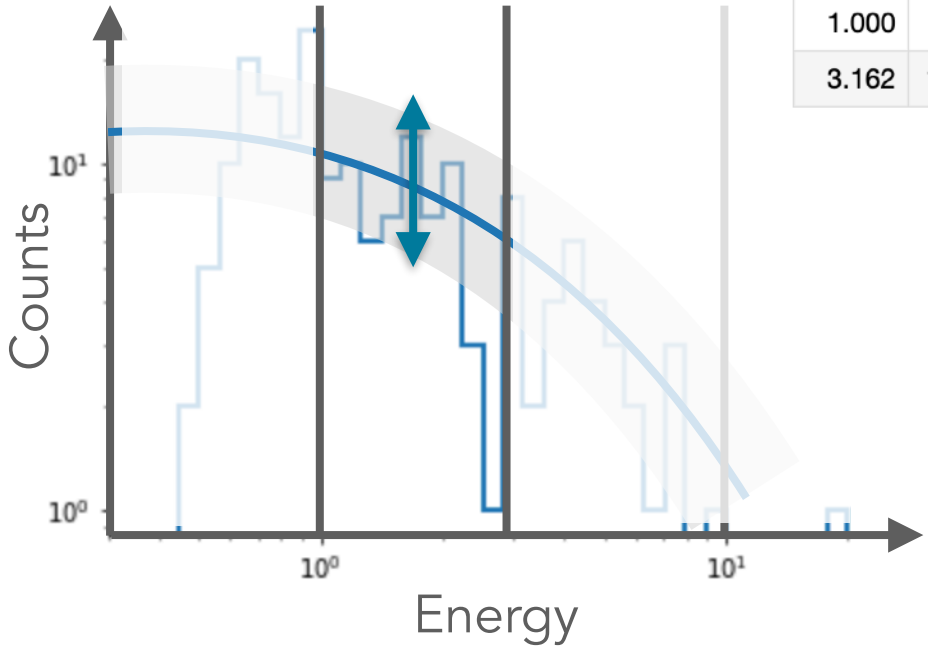
DL4 to DL5: estimating fluxes

- Gammapy provides a set of estimator objects which create DL5 data products based on a model assigned to one or more datasets.



DL4 to DL5: estimating fluxes

- Gammapy provides a set of estimator objects which create DL5 data products based on a model assigned to one or more datasets.

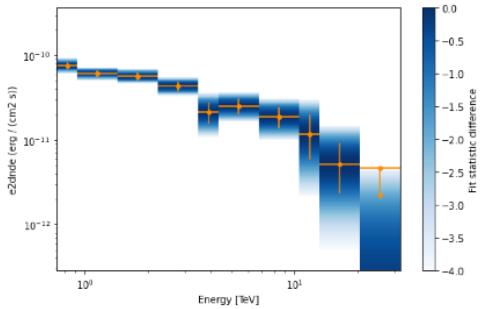


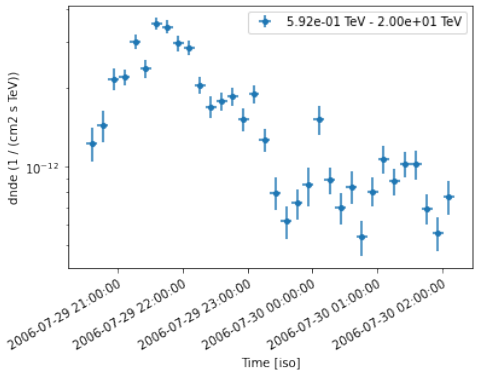
e_min	e_max	ref_flux	ref_eflux	norm	norm_err	norm_ul
TeV	TeV	1 / (cm2 s)	TeV / (cm2 s)			
float64	float64	float64	float64	float64	float64	float64
0.300	1.000	1.699e-10	8.184e-11	nan	nan	nan
1.000	3.162	2.433e-11	3.845e-11	1.032	0.058	1.152
3.162	10.000	3.847e-12	1.923e-11	0.879	0.103	1.099

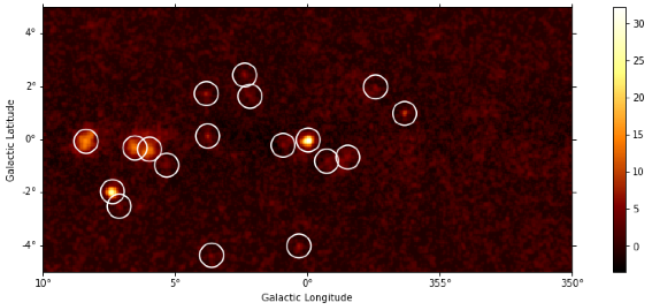
`FluxPointsEstimator`
`LightCurveEstimator`
`TSMapEstimator`
`ExcessMapEstimator`

DL4 to DL5: estimating fluxes

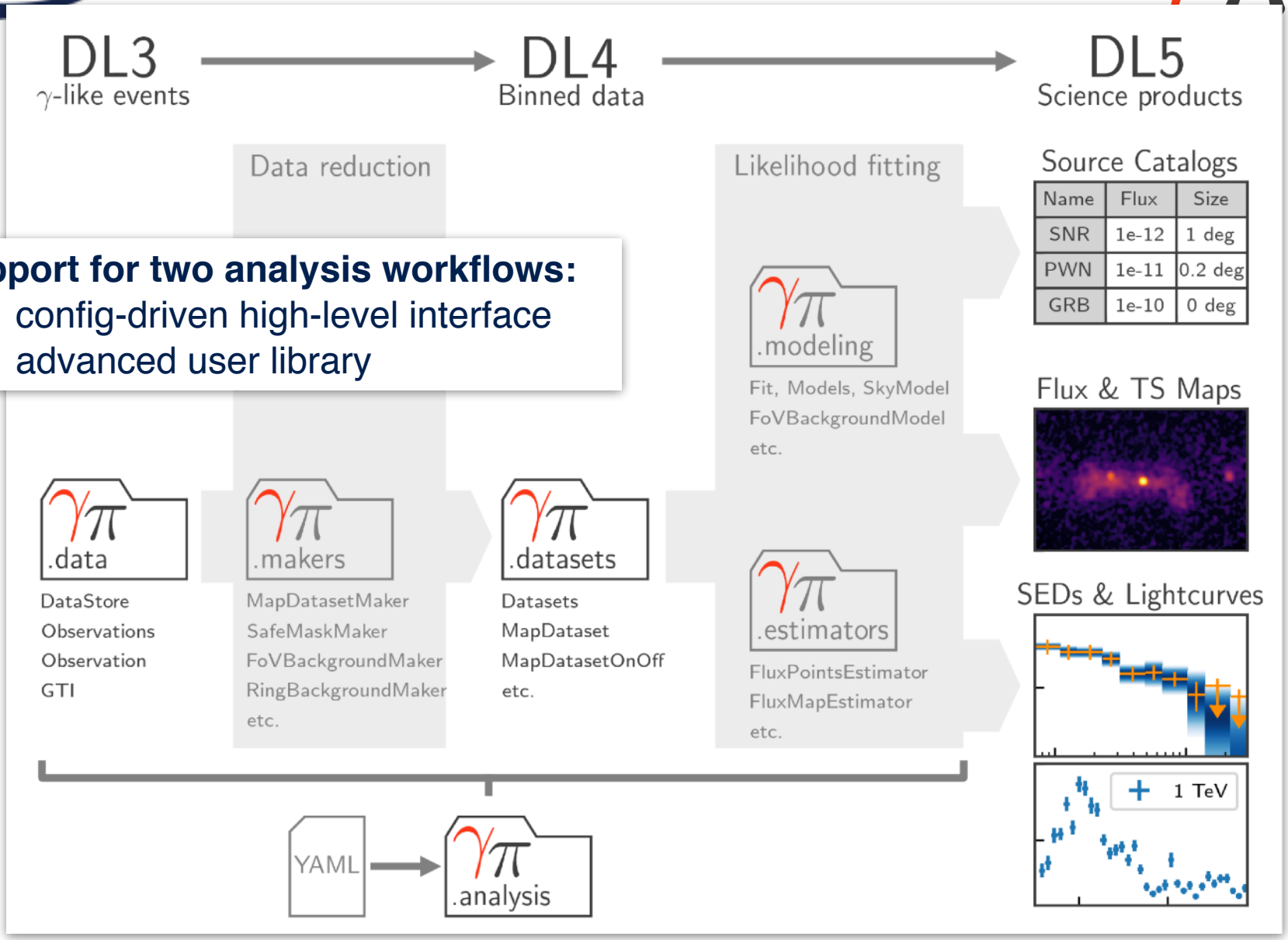
- Gammapy provides a set of estimator objects which create DL5 data products based on a model assigned to one or more datasets.
  - Once a proper model is determined
  - In predefined energy intervals, estimators compute:
    - fluxes errors and associated significance
    - fit statistic scan etc.
  - They can produce flux points, light curves, flux maps







Data workflow and package structure



support for two analysis workflows:

- config-driven high-level interface
- advanced user library

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

Tutorials

We have prepared two specific analysis tutorials:

Spectral analysis of PKS 2155-304:
A full 1D (spectral) analysis from A to Z for a point-like extra-Galactic source.

3D analysis of MSH 15-52
A full 3D analysis from A to Z for an extended Galactic source.

You can retrieve them with:

```
git clone https://github.com/luca-giunti/CTAO-CTAC_Meeting_Bologna_2022.git
```

You can try to execute the tutorials along or simply follow.