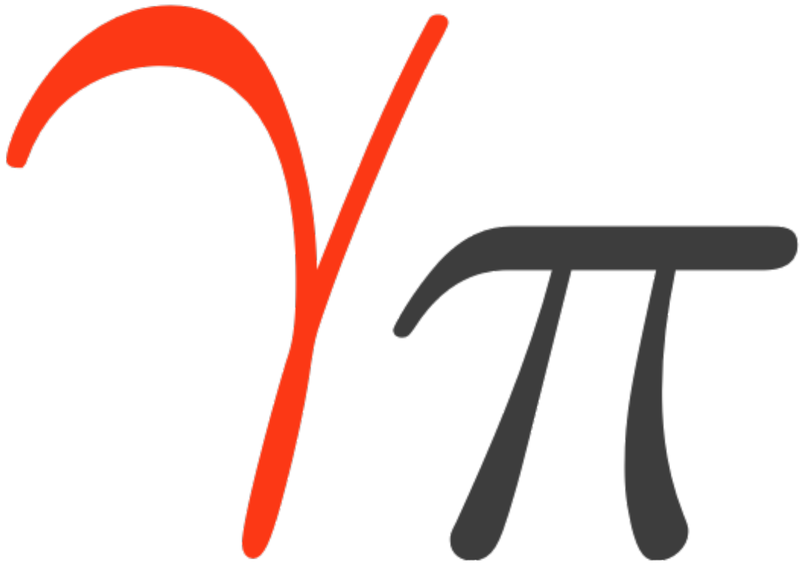


GAMMAPY

A Python Package for (not only) Gamma-Ray Astronomy

Axel Donath for the Gammapy developers

Gammapy User Call, Oct. 26th

INTRODUCTION

WHAT IS GAMMAPY?

H.E.S.S.

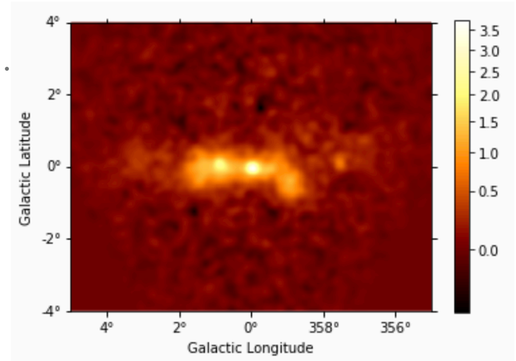
cta
cherenkov telescope array

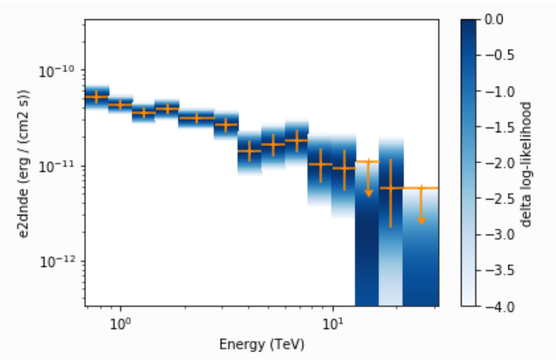
Fermi
Gamma-ray
Space Telescope

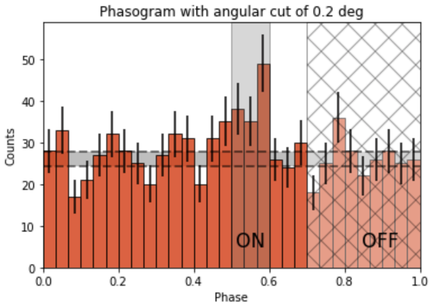
HAWC
High Altitude Water Cherenkov
Gamma-Ray Observatory

KM3NeT

etc.







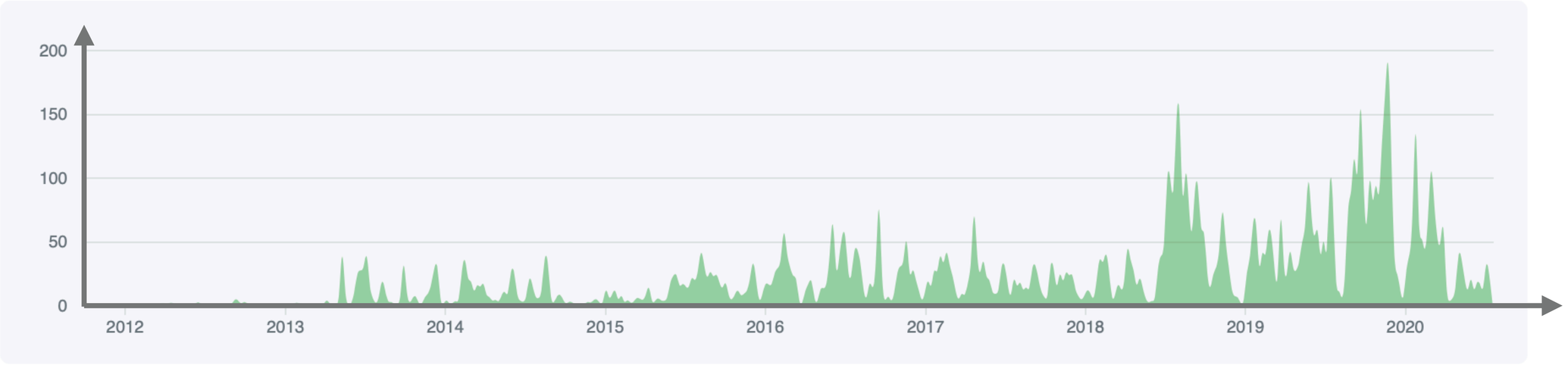
astropy

NumPy

SciPy

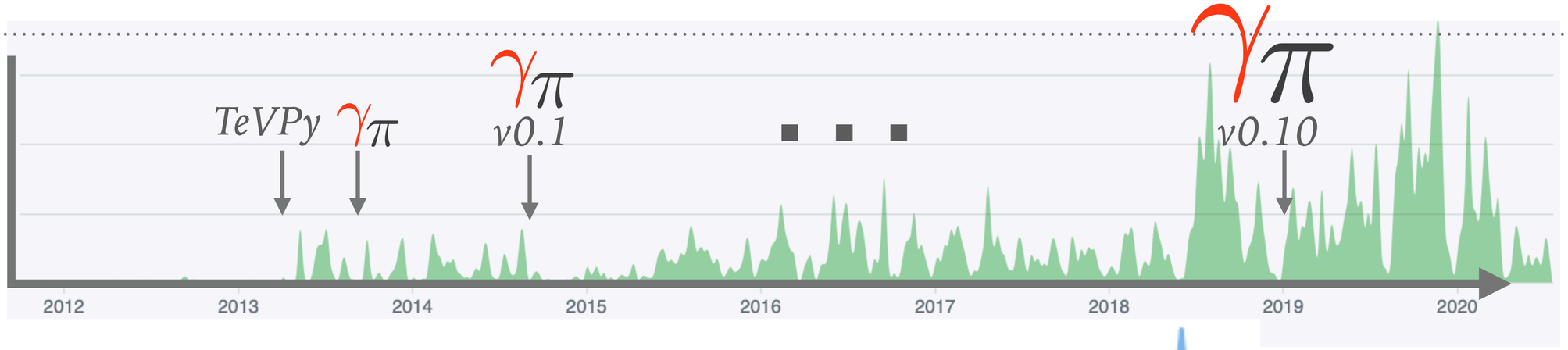
WHAT IS GAMMAPY?

- An openly developed **Python package for Gamma-Ray astronomy**
- Started in ~2012 with a set of Python scripts developed for the HESS Galactic Plane Survey analysis by Christoph Deil and myself
- Approximately 8 years of experience in developing and maintaining an open source Python package and have probably re-written the package already 3 times…:-)
- **~120 forks, ~60 contributors**
- **~13.000 commits**

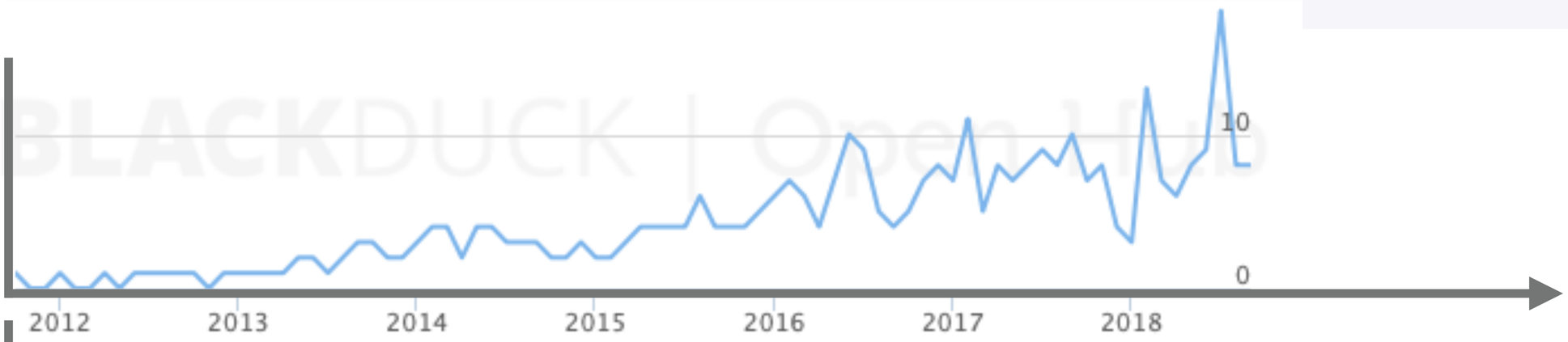


WHAT IS GAMMAPY?

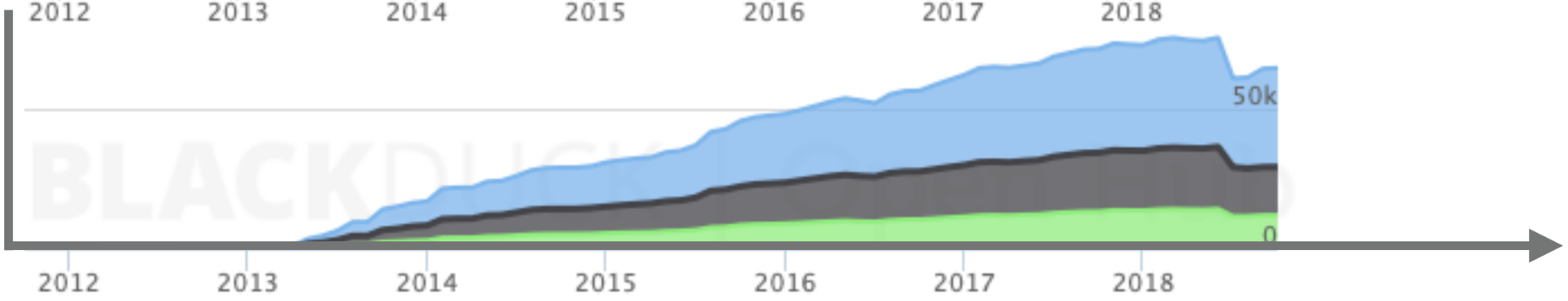
Activitity



Contributors



Lines of code

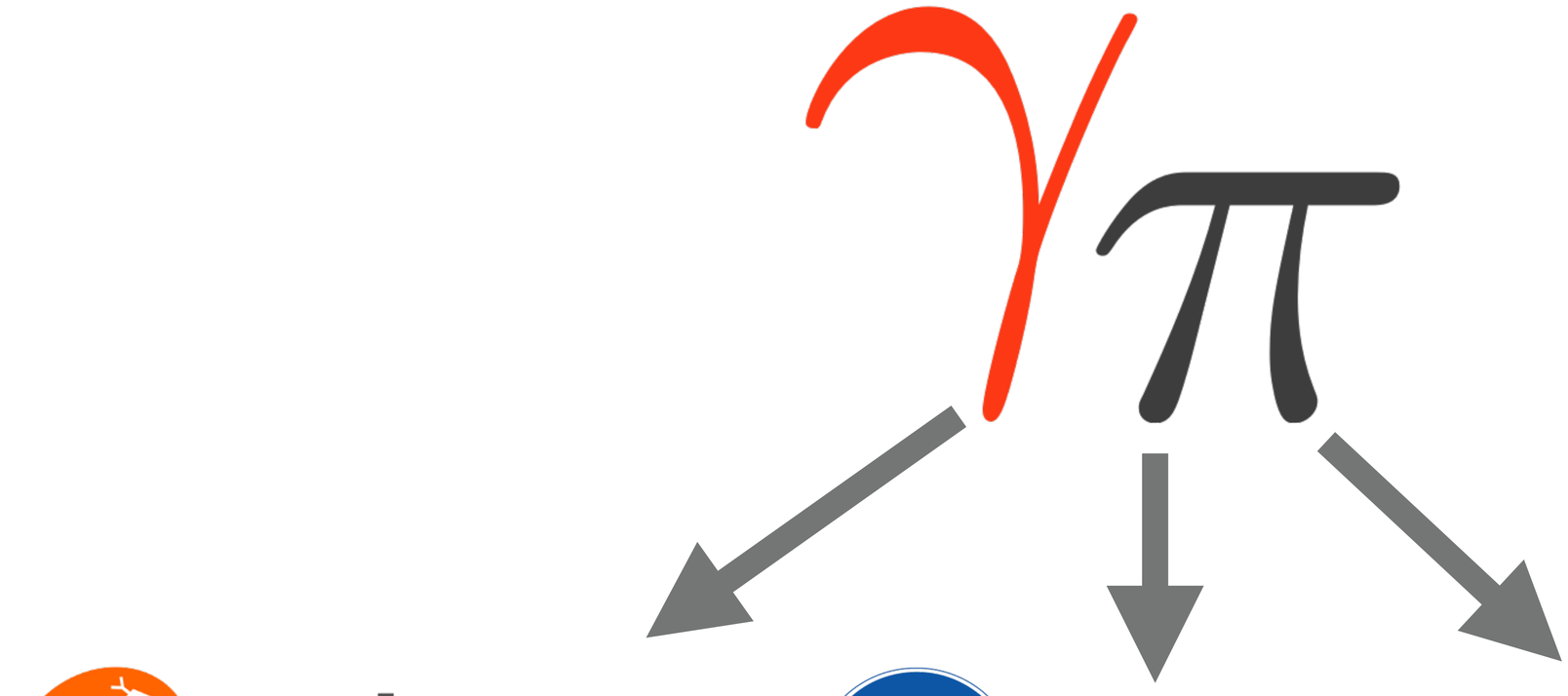


https://www.openhub.net/p/gammapy

DEVELOPMENT & SETUP

DEPENDENCIES

Required dependencies →



astropy

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

SciPy

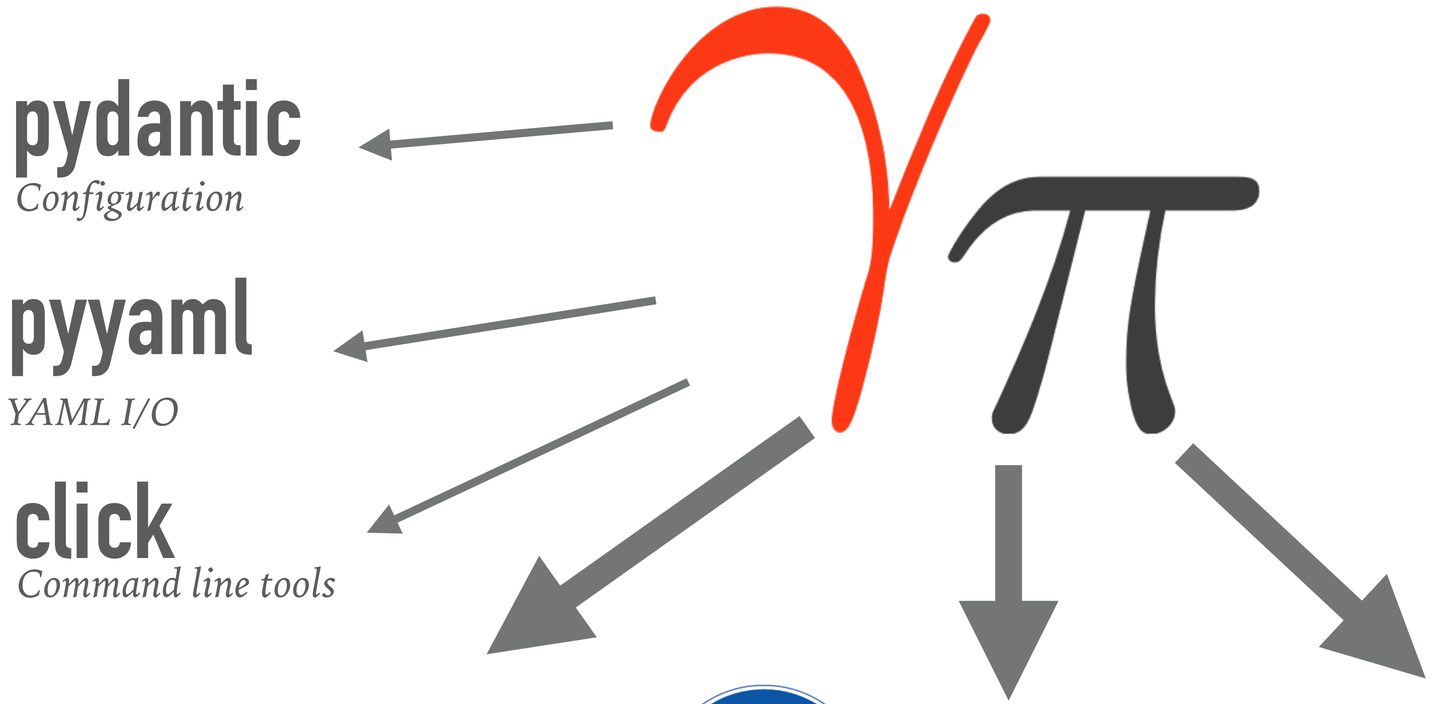
Interpolation, minimisation, FFT convolution, etc.

NumPy

ND-data structures and computations

DEPENDENCIES

Required dependencies →



astropy

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

SciPy

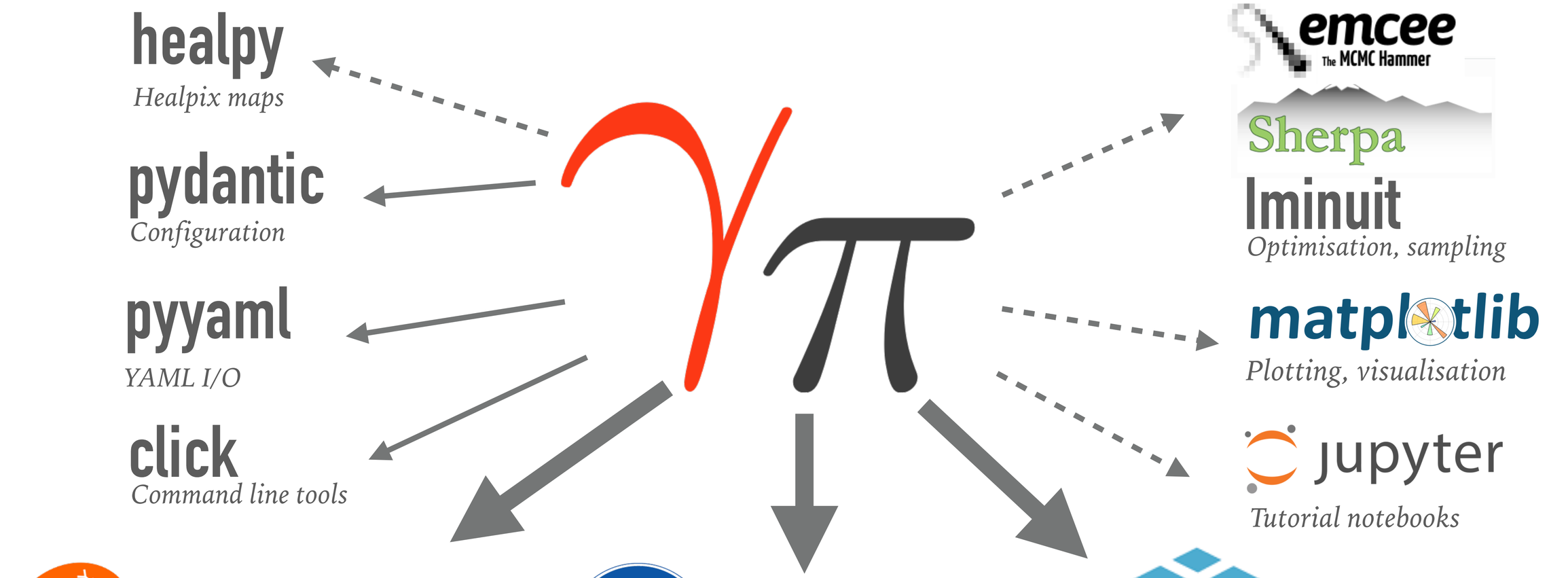
Interpolation, minimisation, FFT convolution, etc.

NumPy

ND-data structures and computations

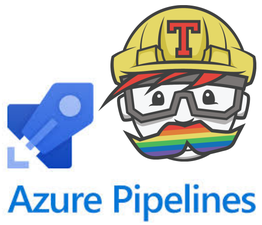
DEPENDENCIES

Optional dependencies ⇢ *Required dependencies* →



DEVELOPMENT AND CI SETUP

Hosted and openly developed on Github:
https://github.com/gammapy/gammapy



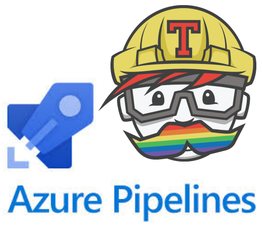
Travis-CI and Azure Pipelines used
for continuous integration

Coveralls used for monitoring
of code test coverage

Docs are build and deployed
manually: https://docs.gammapy.org/

DEVELOPMENT AND CI SETUP

Hosted and openly developed on Github:
https://github.com/gammapy/gammapy



Travis-CI and Azure Pipelines used
for continuous integration

Coveralls used for monitoring
of code test coverage

Docs are build and deployed
manually: https://docs.gammapy.org/

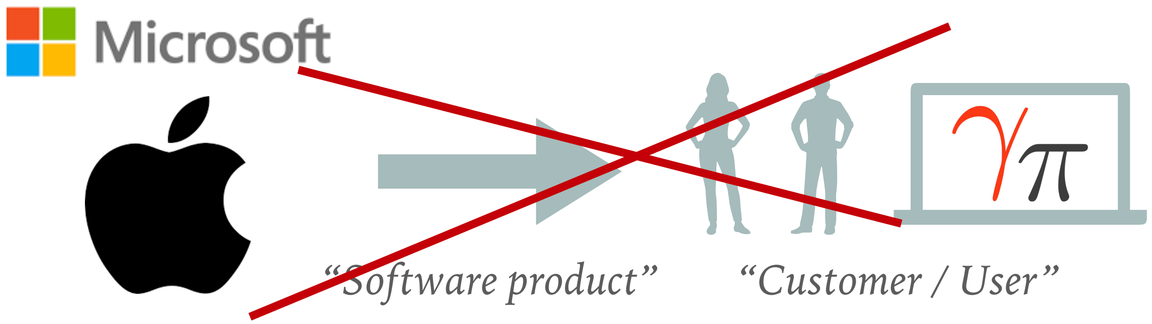
Black code formatting used
for a consistent code format.

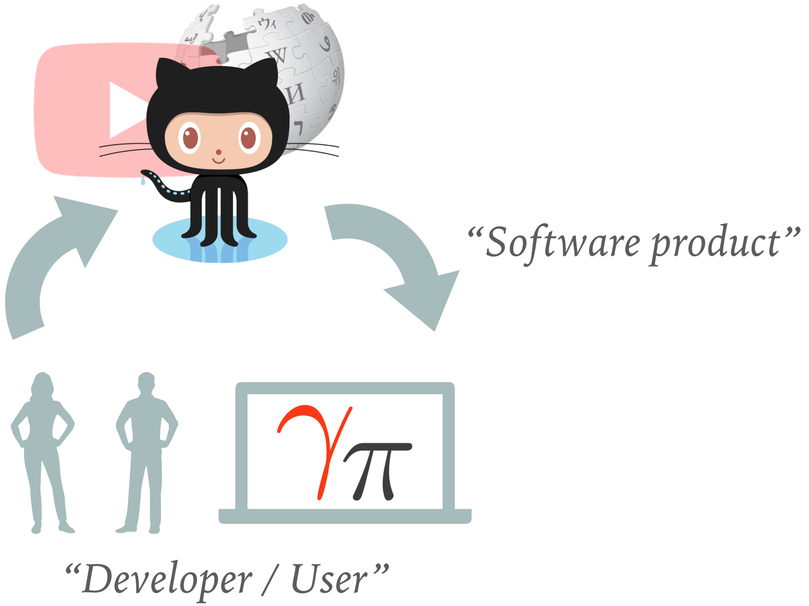
Sphinx used to build the
documentation

Pytest used for testing

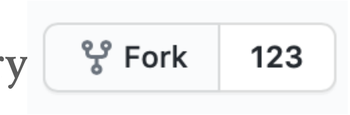
WHAT IS “COMMUNITY DRIVEN” / OPEN DEVELOPMENT?

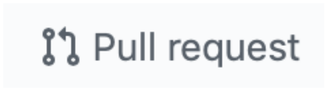
- Not a classical linear “developer -> user” or “company -> customer” relationship
- **No strict, structural boundary between developers and users**
  - Users know best what they need and deliver it for the benefit of the community
- Maybe **best understood in a Web 2.0 context**? There is no strict boundary between content creator and consumer…(Youtube / Wikipedia etc.)



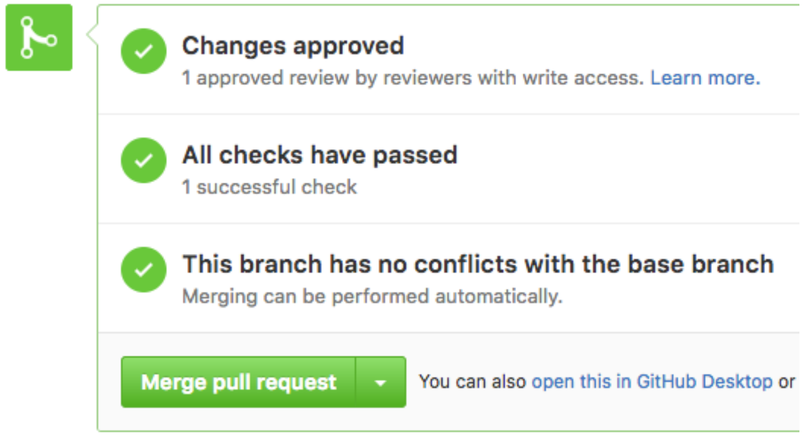


“GITHUB” WORKFLOW

- Standard **multi-branch git workflow**:
  - Contributors **fork a repository** 

  - Features are developed in **new a branch** “on the side”
  - A **pull requests** is opened 

- Every pull requests (PR) is **reviewed at least once** by more experienced developers (lead developers). Sometimes “all fine”, sometime “Here is a number of substantial comments”
- Once review comments are implemented and the CI builds pass a PR gets merged



INSTALLATION & SETUP

https://docs.gammapy.org/0.17/install/index.html

Install Gammapy:

1. `$ curl -O https://gammapy.org/download/install/gammapy-0.17-environment.yml`
2. `$ conda env create -f gammapy-0.17-environment.yml`
3. `$ conda activate gammapy-0.17`

INSTALLATION & SETUP

https://docs.gammapy.org/0.17/install/index.html

Install Gammapy:

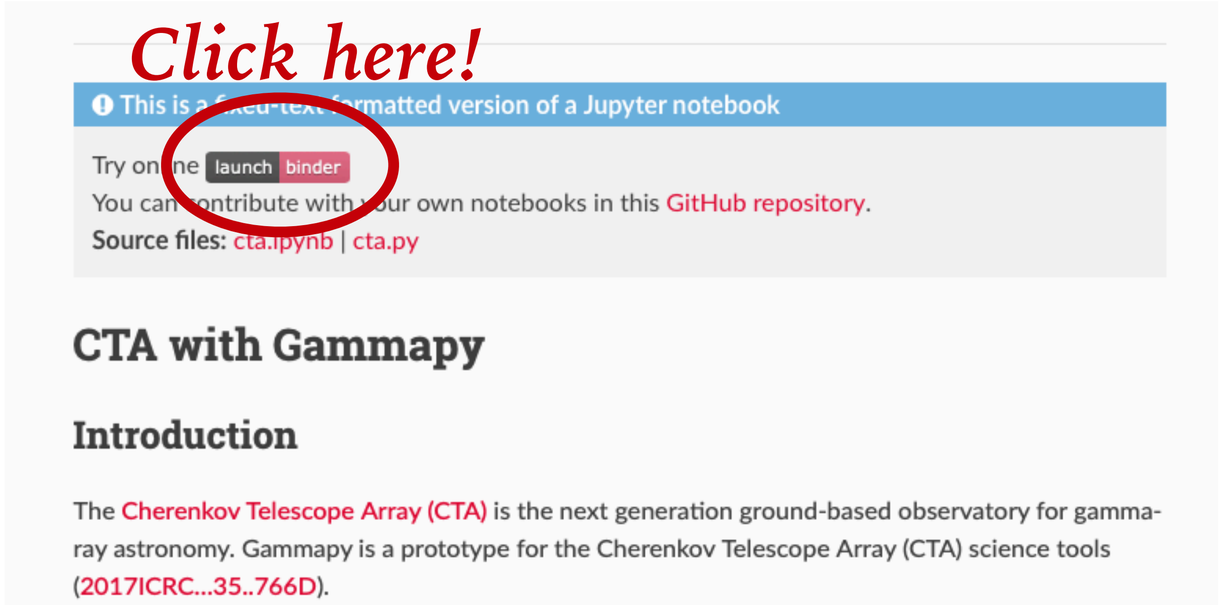
1. `$ curl -O https://gammapy.org/download/install/gammapy-0.17-environment.yml`
2. `$ conda env create -f gammapy-0.17-environment.yml`
3. `$ conda activate gammapy-0.17`

Download tutorials:

1. `$ gammapy download tutorials`
2. `$ cd gammapy-tutorials`
3. `$ export GAMMAPY_DATA=$PWD/datasets`

TUTORIALS

- Best starting point are the tutorials notebooks: https://docs.gammapy.org/0.17/tutorials/index.html
  - CTA data access: https://docs.gammapy.org/0.17/notebooks/cta.html
  - CTA data analysis: https://docs.gammapy.org/0.17/notebooks/cta_data_analysis.html
  - And many, many others…
- Tutorials can be executed online as well (using Binder):



jupyter

binder

RELATED PROJECTS

gamma-cat

"An open data collection and source catalog for VHE gamma-ray astronomy"

https://github.com/gammapy/gamma-cat

PyGamma19

"Python and open data for Gamma-Ray Astronomy Workshop", March 18th - 22nd

https://indico.cern.ch/event/783425/

gammasky.net

"A portal to the gamma-ray sky"

https://github.com/gammapy/gamma-sky

Gamma-astro-data-formats

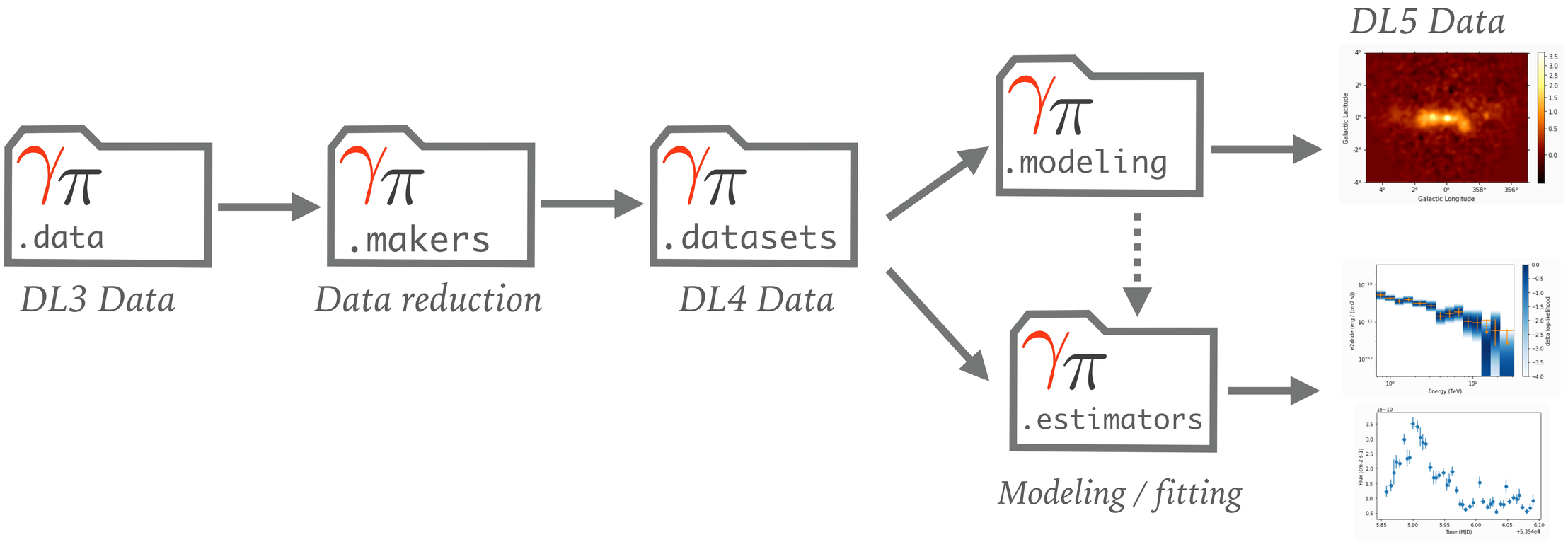
"A place to propose and share data format descriptions for gamma-ray astronomy."

https://github.com/open-gamma-ray-astro

FEATURES

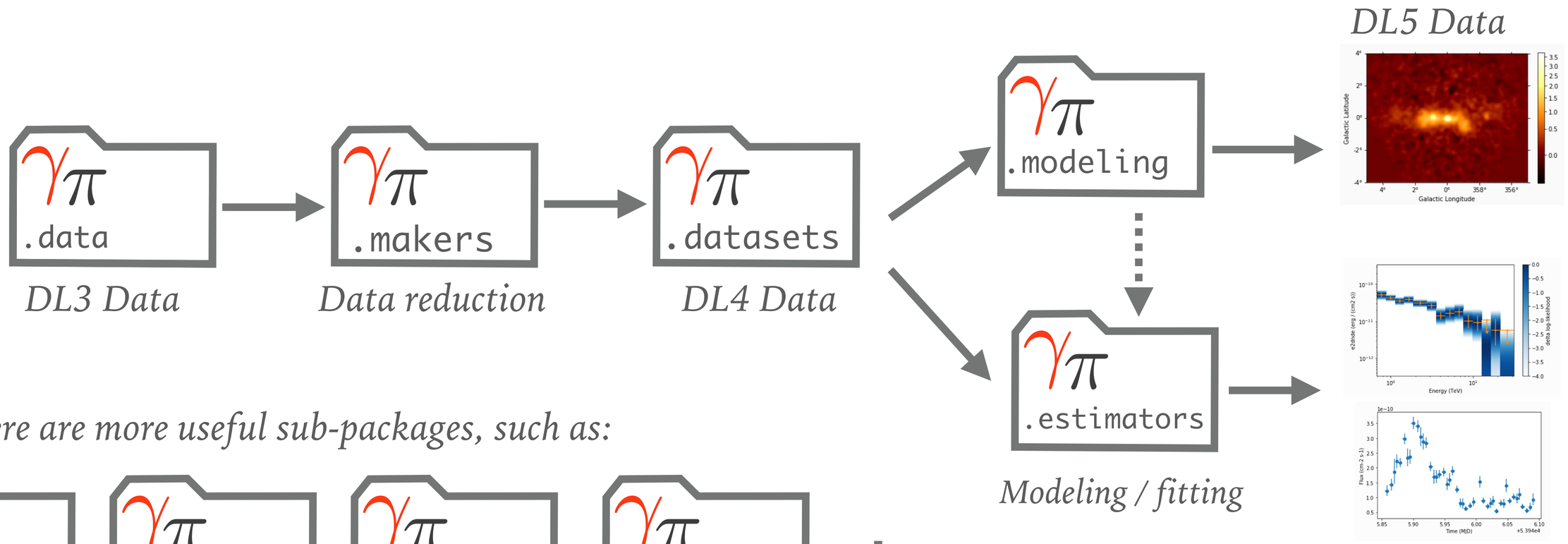
PACKAGE STRUCTURE

- Gammapy is sructured into sub-package, mostly based **by API and data level.** The core functionality for a typical analysis workflow is distributed as following:



PACKAGE STRUCTURE

- Gammapy is sructured into sub-package, mostly based **by API and data level.** The core functionality for a typical analysis workflow is distributed as following:

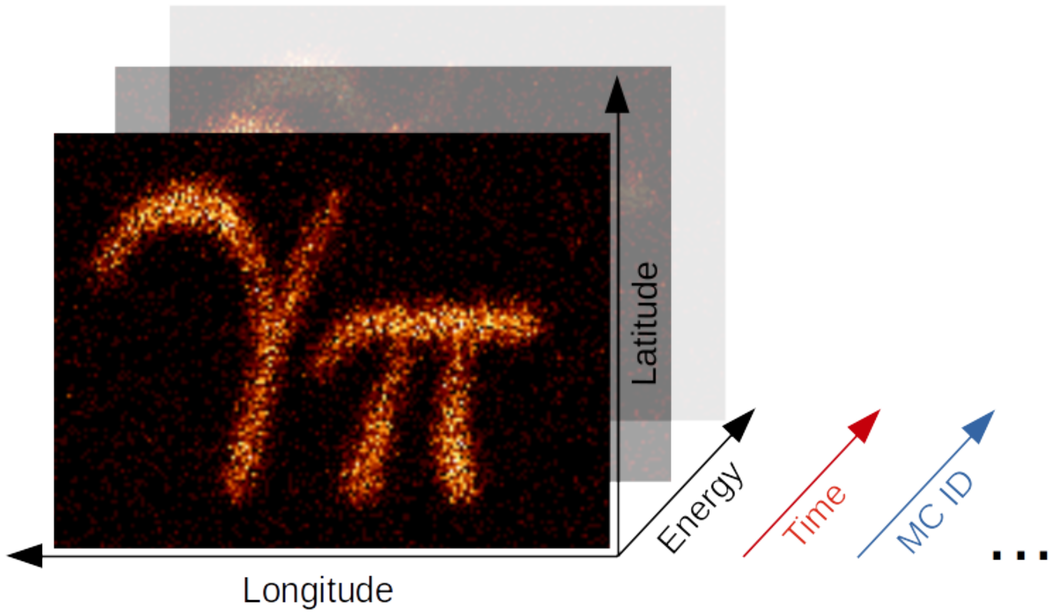


But there are more useful sub-packages, such as:

.astro .irf .maps .analysis **etc.**

GAMMAPY.MAPS

https://docs.gammapy.org/0.17/maps/index.html



- WCS and HEALPix image based data structures with arbitrary number of non-spatial extra axes such as energy or time
- Uniform API for WCS and HEALPix based maps
- Interpolation, re-projection, smoothing, convolution, FITS serialisation, (interactive) plotting etc. all generalised to N-dimensions

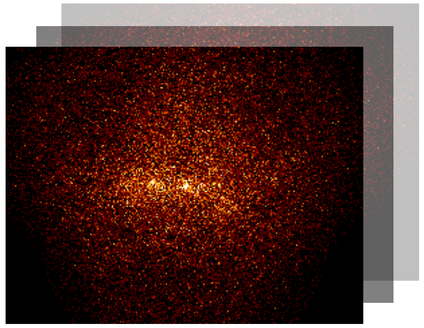
Uniform API for WCS and HEALPix…

```
from gammapy.maps import Map
from astropy.coordinates import SkyCoord

position = SkyCoord(0.0, 5.0, frame='galactic', unit='deg')

# Create a WCS Map
m_wcs = Map.create(binsz=0.1, map_type='wcs', skydir=position, width=10.0)

# Create a HPX Map
m_hpx = Map.create(binsz=0.1, map_type='hpx', skydir=position, width=10.0)
```

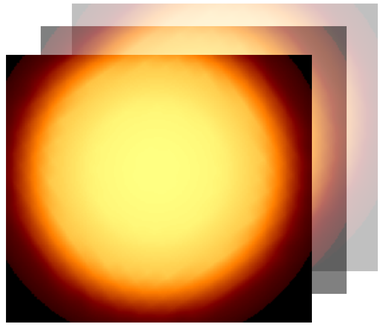
Handling of arbitrary non-spatial extra axes…

```
from gammapy.maps import Map, MapAxis
from astropy.coordinates import SkyCoord

position = SkyCoord(0.0, 5.0, frame='galactic', unit='deg')
energy_axis = MapAxis.from_bounds(100., 1E5, 12, interp='log', unit='GeV', name='energy')
time_axis = MapAxis.from_bounds(0., 12, 12, interp='lin', unit='h', name='time')

# Create a WCS Map
m_wcs = Map.create(binsz=0.02, map_type='wcs', skydir=position, width=10.0,
                   axes=[energy_axis, time_axis])
```

GAMMAPY.DATASETS



Reduced data



BKG model

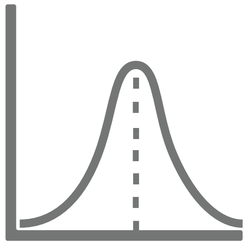
$\gamma\pi_{.\text{maps}}$

$\gamma\pi_{.\text{irf}}$ $\gamma\pi_{.\text{modeling}}$

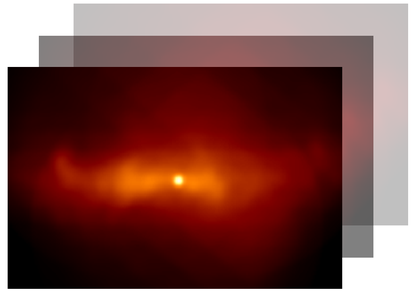
$\gamma\pi_{.\text{stats}}$

MapDataset

$\mathscr{C}(x, \theta)$



Reduced IRFs

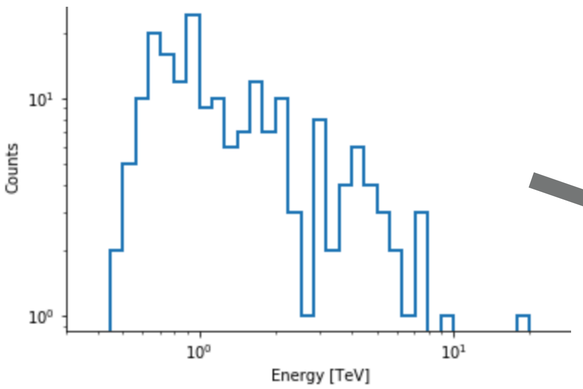


Source model

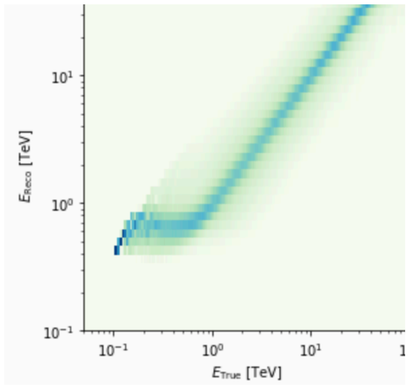
I/O

GAMMAPY.DATASETS

OGIP Format



Reduced data



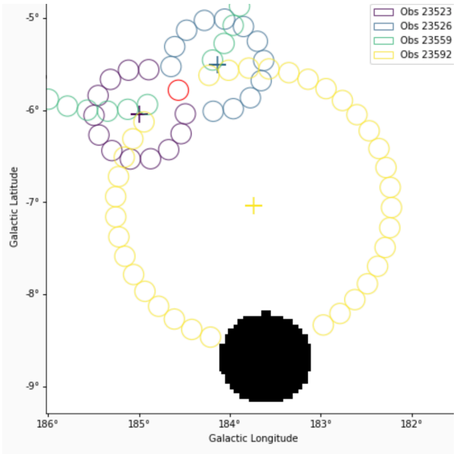
Reduced IRFs

$\gamma\pi$.datasets

$\gamma\pi$.modeling

$\gamma\pi$.irf

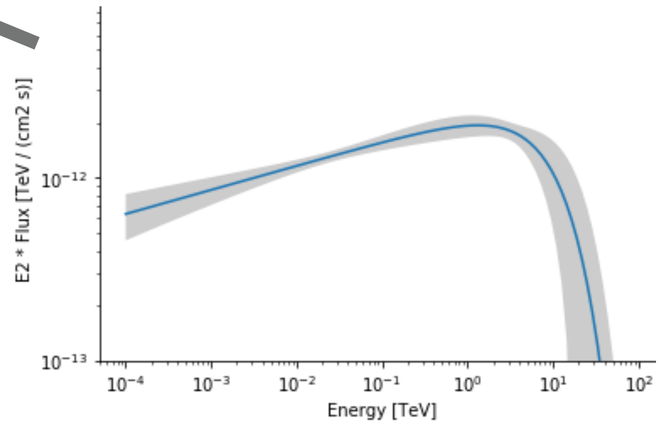
$\gamma\pi$.stats



BKG data / model

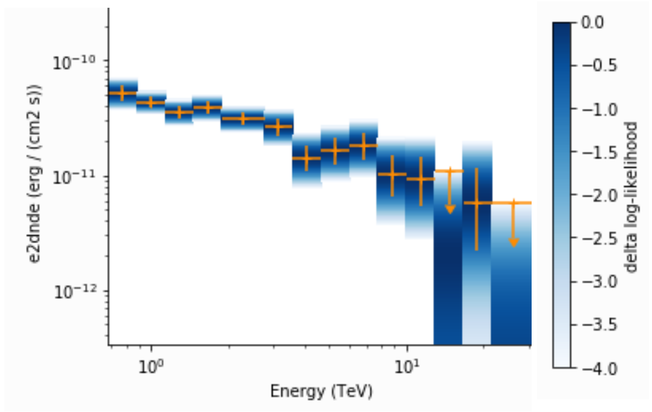
SpectrumDatasetOnOff

$\mathscr{W}(x, \theta)$



Spectral model

GAMMAPY.DATASETS



Reduced data

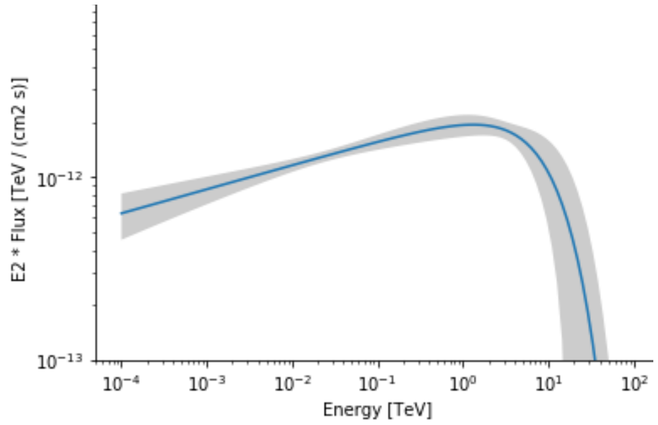
$\gamma\pi$.estimators.flux_points

$\gamma\pi$.modeling.models

$\gamma\pi$.stats

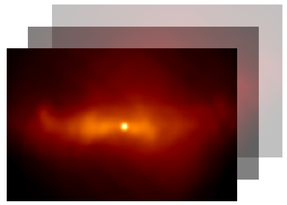
FluxPointsDataset

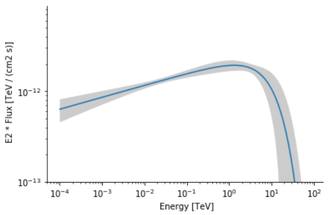
$\chi^2(x, \theta)$



Spectral model

JOINT LIKELIHOOD FITTING





Model

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

gammapy.utils.fitting

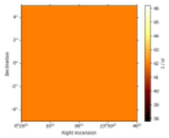
- .optimize
- Sherpa .optmethods
- Iminuit

```
fit = Fit([dataset])
fit = Fit([dataset_1, dataset_2, …])

fit.optimize(backend="sherpa")
fit.optimize(backend="scipy")

fit.covariance(backend="minuit")
fit.confidence(parameter="amplitude")

fit.likelihood_profile(parameter="amplitude")
```

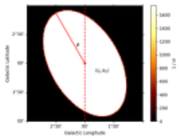
GAMMAPY.MODELING.MODELS

- Model gallery: https://docs.gammapy.org/0.17/modeling/gallery/index.html#model-gallery

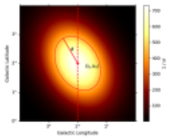
Spatial models



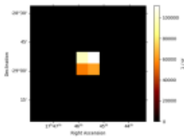
Constant Spatial Model



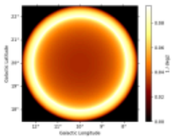
Disk Spatial Model



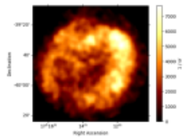
Gaussian Spatial Model



Point Spatial Model

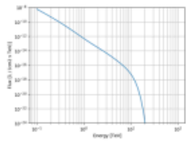


Shell Spatial Model

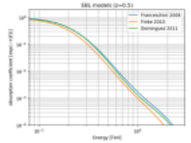


Template Spatial Model

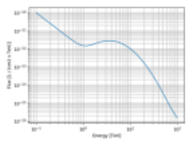
Spectral models…



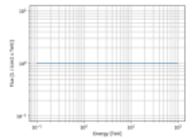
Absorbed Spectral Model



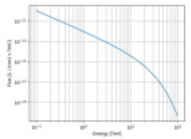
Absorption Spectral Model



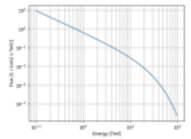
Compound Spectral Model



Constant Spectral Model

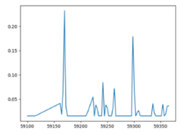


Exponential Cutoff Powerlaw Spectral Model

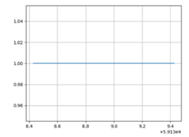


Exponential Cutoff Powerlaw Spectral Model used for 3FGL

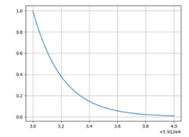
Temporal models…



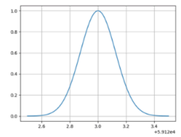
LightCurve Temporal Model



Constant Temporal Model



ExpDecay Temporal Model



Gaussian Temporal Model

- Models tutorial: https://docs.gammapy.org/0.17/notebooks/models.html, includes example how to implement a custom model and register it for I/O
- Configure IRF application to model components using: `SkyModel(…, apply_irf={“psf”: False})`

FITTING OF TEMPORAL MODELS

https://docs.gammapy.org/0.17/notebooks/light_curve_simulation.html

- In Gammapy v0.17 we added support to evaluate and fit temporal models to a list of datasets

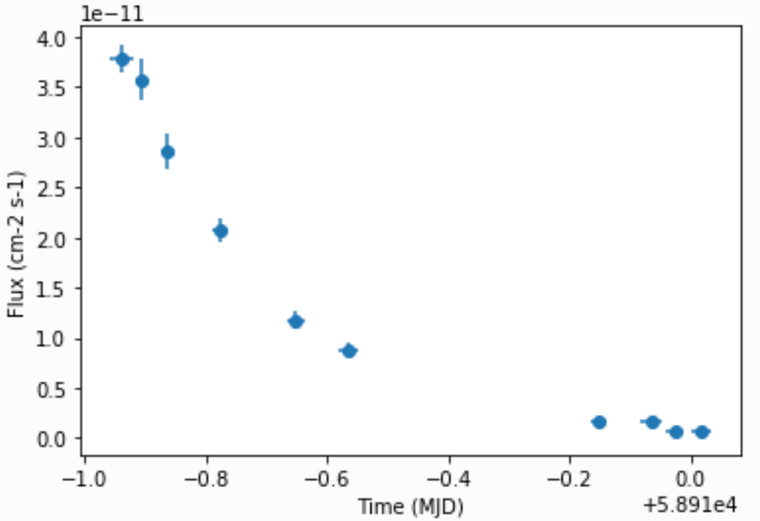
```
# Define the model:

spectral_model1 = PowerLawSpectralModel(
    index=2.0, amplitude="1e-12 cm-2 s-1 TeV-1", reference="1 TeV"
)
temporal_model1 = ExpDecayTemporalModel(t0="10 h", t_ref=gti_t0.mjd * u.d)
model = SkyModel(
    spectral_model=spectral_model1,
    temporal_model=temporal_model1,
    name="model-test",
)
```

Define an additional temporal model component…

```
for i in range(n_obs):
    obs = Observation.create(
        pointing=pointing,
        livetime=lvtm[i],
        tstart=tstart[i],
        irfs=irfs,
        reference_time=gti_t0,
    )
    empty_i = empty.copy(name=f"dataset_{i}")
    maker = SpectrumDatasetMaker(selection=["aeff", "background", "edisp"])
    dataset = maker.run(empty_i, obs)
    dataset.models = model_simu
    dataset.fake()
    datasets.append(dataset)
```

Simulate datasets in a loop…

Compute lightcurve…



Available temporal models

```
Registry
--------

ConstantTemporalModel
LightCurveTemplateTemporalModel
ExpDecayTemporalModel
GaussianTemporalModel
```

CONVENIENCE COVARIANCE HANDLING

- After running `Fit.run()` or `Fit.covariance()` the **covariance is now available on the fitted model**
- Sub-covariance are **automatically tracked** to sub-model components and can be accessed like:

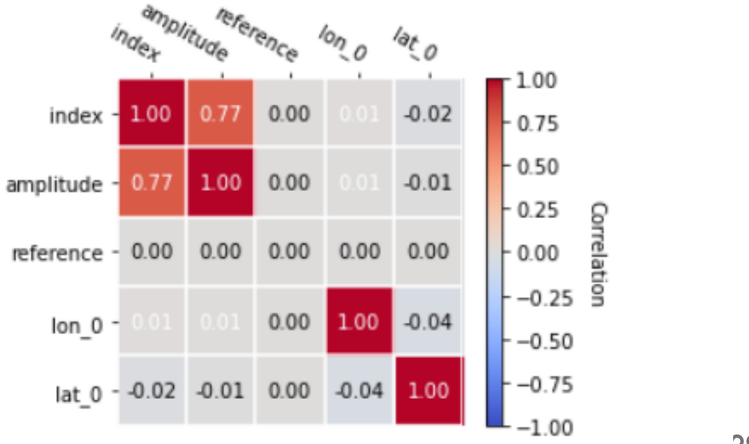
```
print(models.covariance)
print(models["my-model"].covariance)
print(models["my-model"].spectral_model.covariance)
```

- Write / read covariance to file:

```
models.write_covariance("covariance.dat")
models.read_covariance("covariance.dat")
```

Plot correlation between parameters

```
1  sky_model.covariance.plot_correlation();
```

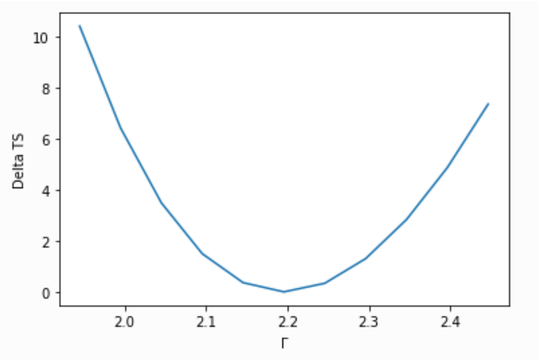


MODELLING

https://docs.gammapy.org/0.17/notebooks/modeling.html

- Any fitting / modelling of data requires **careful checking of fit convergence and results!**
- Demonstrates how to use different optimisers & fitting backends, such as iminuit, scipy and sherpa
- Shows how to compute likelihood profiles and contours for fit diagnostics…

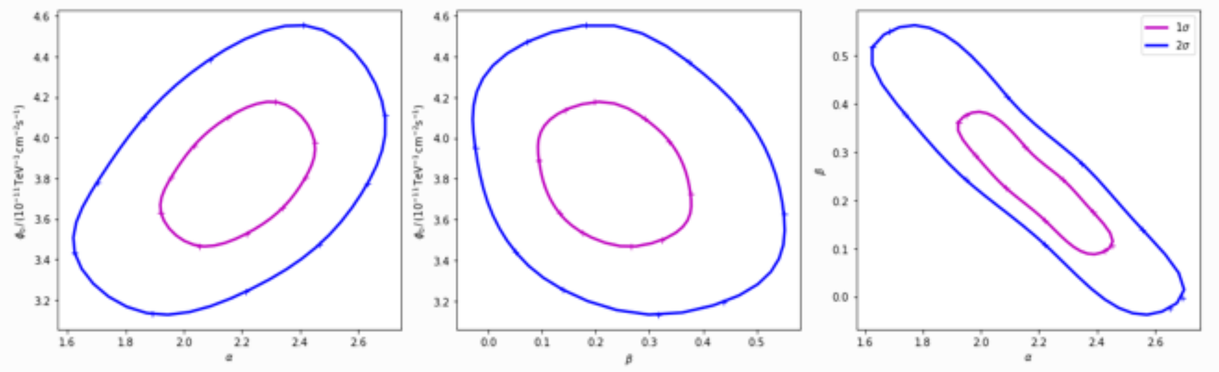
Create likelihood profiles




```
profile = fit.stat_profile(parameter="alpha")
```

```
total_stat = result_minuit.total_stat
plt.plot(profile["values"], profile["stat"] - total_stat)
plt.xlabel(r"$\Gamma$")
plt.ylabel("Delta TS");
```

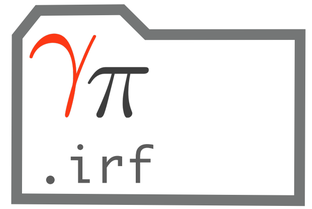
Create likelihood contours




```
for par_1, par_2 in combinations(["alpha", "beta", "amplitude"], r=2):
    contour = fit.minos_contour(
        result.parameters[par_1],
        result.parameters[par_2],
        numpoints=npoints,
        sigma=sigma,
    )
```

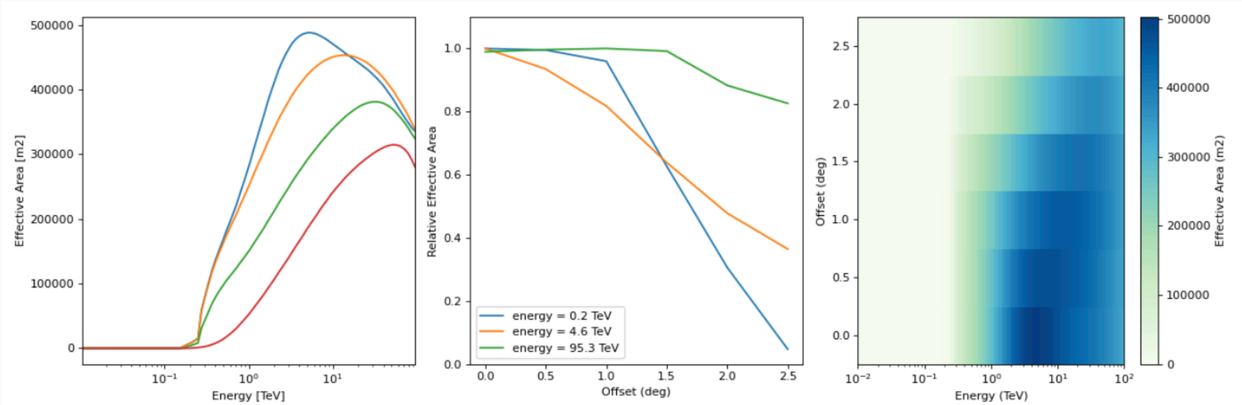
GAMMAPY.IRF

https://docs.gammapy.org/0.17/irf/index.html



Read and plot effective area…

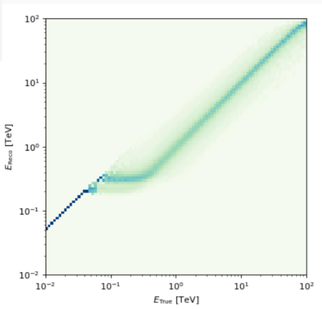
```
"""Plot an effective area from the HESS DL3 data release 1."""
import matplotlib.pyplot as plt
from gammapy.irf import EffectiveAreaTable2D

filename = "$GAMMAPY_DATA/hess-dl3-dr1/data/hess_dl3_dr1_obs_id_020136.fits.gz"
aeff = EffectiveAreaTable2D.read(filename, hdu="AEFF")
aeff.peek()
plt.show()
```



- Handling of **instrument response functions** such as: effective area, point spread function (PSF), energy resolution (EDisp) and instrumental background….
- Handling of position dependent IRFs via `PSFMap` and `EDispKernelMap`
- Mostly visualisation, interpolation and serialisation from and to FITS

Read and plot energy resolution….

```
"""Plot an energy dispersion from the HESS DL3 data release 1."""
import matplotlib.pyplot as plt
from gammapy.irf import EnergyDispersion2D

filename = "$GAMMAPY_DATA/hess-dl3-dr1/data/hess_dl3_dr1_obs_id_020136.fits.gz"
edisp = EnergyDispersion2D.read(filename, hdu="EDISP")
edisp.peek()
plt.show()
```



GAMMAPY.CATALOG

https://docs.gammapy.org/0.17/tutorials/catalog.html

```python
from gammapy.catalog import SourceCatalog3FHL

fermi_3fhl = SourceCatalog3FHL()
crab_3fhl = fermi_3fhl["Crab Nebula"]

crab_3fhl.spectral_model.energy_flux("1 TeV", "10 TeV").to("erg cm-2 s-1")
```

$$1.112798 \times 10^{-10} \ \frac{\mathrm{erg}}{\mathrm{s\ cm^2}}$$

Source_Name	RAJ2000	DEJ2000	GLON	GLAT	Conf_95_SemiMajor	Conf_95_SemiMinor	Conf_95_PosAng	ROI_num	Signif_Avg	Pivot_Energy	Flux_Density
	deg	deg	deg	deg	deg	deg	deg			GeV	1 / (cm2 GeV s)
bytes18	float32	float32	float32	float32	float32	float32	float32	int16	float32	float32	float32
3FHL J0001.2-0748	0.3107	-7.8075	89.0094	-67.3118	0.0424	0.0424	nan	64	5.362	23.73	5.3174e-13
3FHL J0001.9-4155	0.4849	-41.9303	334.1216	-72.0697	0.1018	0.1018	nan	429	5.638	28.42	5.4253e-13
3FHL J0002.1-6728	0.5283	-67.4825	310.0868	-48.9549	0.0357	0.0357	nan	386	8.470	20.82	1.2062e-12

- Convenience access to existing gamma-ray catalogs, from Fermi-LAT, HESS, HAWC, etc.
- Uniform access API and translation of the info to Gammapy objects, such as models, flux points, light-curves

H.E.S.S. HGPS

Fermi Gamma-ray Space Telescope 2FHL 3FHL 3FGL 4FGL

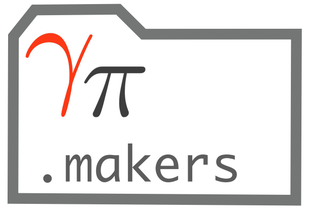
2HWC HAWC High Altitude Water Cherenkov Gamma-Ray Observatory

gamma-cat

"An open data collection and source catalog for VHE gamma-ray astronomy"

GAMMAPY.MAKERS

https://docs.gammapy.org/0.17/makers/index.html



- A uniform "Maker" API for data reduction:

```
from gammapy.datasets import  MapDataset
from gammapy.makers import MapDatasetMaker, SafeMaskMaker
from gammapy.data import DataStore
from gammapy.maps import MapAxis, WcsGeom

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

energy_axis = MapAxis.from_bounds(1, 10, nbin=11, name="energy", unit="TeV", interp="log")
geom = WcsGeom.create(skydir=(83.63, 22.01), axes=[energy_axis], width=5, binsz=0.02)

maker = MapDatasetMaker()
safe_mask_maker = SafeMaskMaker(methods=["aeff-default", "offset-max"], offset_max="3 deg")

stacked = MapDataset.create(geom)

for obs in observations:
    cutout = stacked.cutout(obs.pointing_radec, width="6 deg")
    dataset = maker.run(cutout, obs)
    dataset = safe_mask_maker.run(dataset, obs)
    stacked.stack(dataset)

print(stacked)
```

Makers are created and configured first…

…and then executed using .run(dataset, obs) in a chain in the data reduction loop.

Available makers:

`ReflectedRegionsBackgroundMaker`	([...])
`AdaptiveRingBackgroundMaker`	(r_in, r_out_max, ...)
`FoVBackgroundMaker`	([method, exclusion_mask])
`PhaseBackgroundMaker`	(on_phase, off_phase)
`RingBackgroundMaker`	(r_in, width[, ...])
`SpectrumDatasetMaker`	([...])
`MapDatasetMaker`	([background_oversampling, ...])
`SafeMaskMaker`	([methods, aeff_percent, ...])

https://docs.gammapy.org/dev/makers/index.html#classes

- Example for reflected regions: https://docs.gammapy.org/dev/makers/reflected.html#using-the-reflected-background-estimator
- Also check tutorials…

EVENT SAMPLING

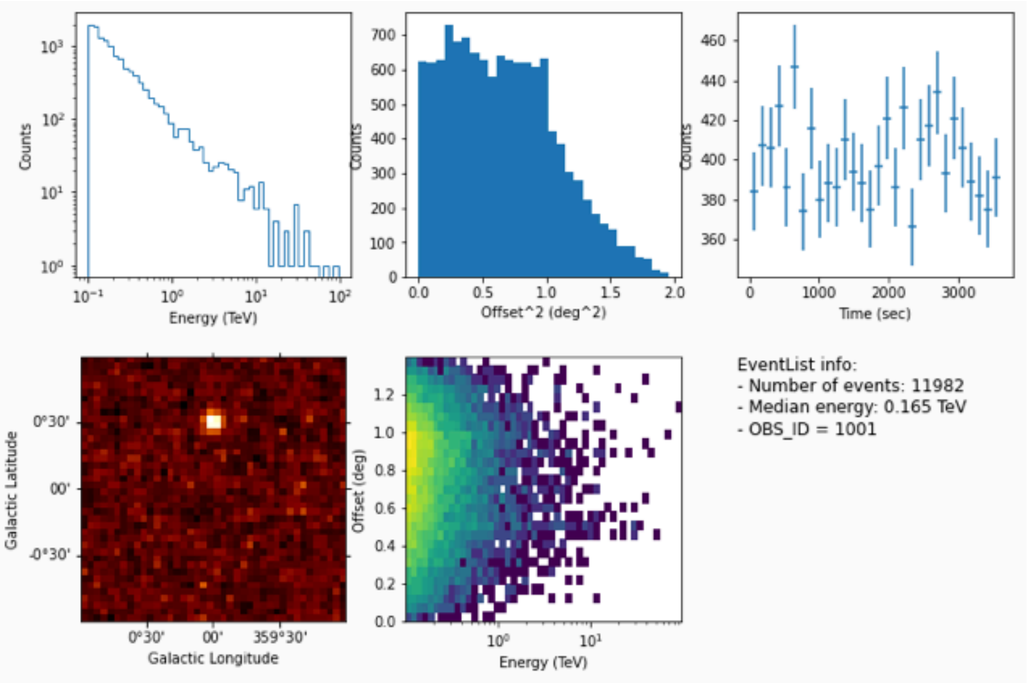
https://docs.gammapy.org/0.17/tutorials/event_sampling.html

- In Gammapy v0.17 we introduced event sampling via the `MapDatasetEventSampler` object, which simulates a list of events for a given dataset comprising a sky model and reduced IRFs

Simulate events for multiple observations

```
for idx, tstart in enumerate(tstarts):

    observation = Observation.create(
        obs_id=idx,
        pointing=POINTING,
        tstart=tstart,
        livetime=livetimes[idx],
        irfs=irfs,
    )

    dataset = maker.run(empty, observation)
    dataset.models.extend(models_pntpwl)

    sampler = MapDatasetEventSampler(random_state=idx)
    events = sampler.run(dataset, observation)
    events.table.write(
        f"./event_sampling/events_{idx:04d}.fits", overwrite=True
    )
```

Statistics of simulated events



- Event sampling is validated here: https://github.com/gammapy/gammapy-benchmarks/tree/master/validation/event-sampling

MISCELLANEOUS

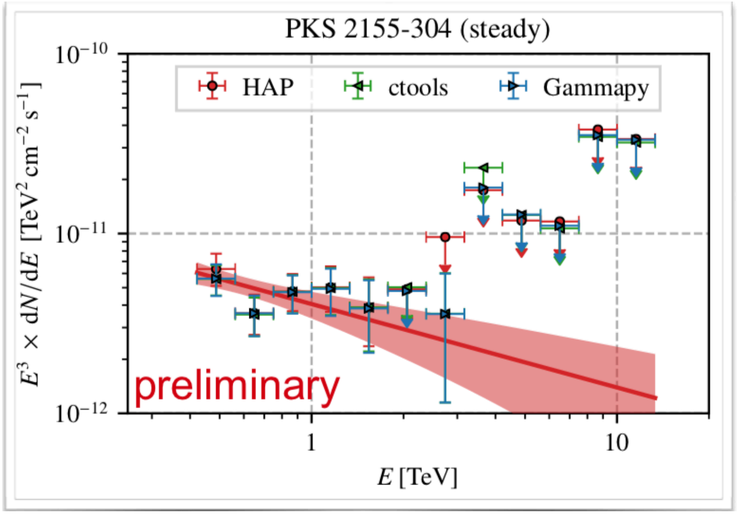
- With `MapDataset.to_spectrum_dataset(region=)` there is the possibility to extract a spectrum dataset from a 3D datasets for further analysis, see https://docs.gammapy.org/0.17/notebooks/extended_source_spectral_analysis.html
- To unify the API between spectral and “cube” analysis we introduced `RegionGeom / RegionNDMap` classes, which bundle the region information with the spectral information and are compatible in API with the `Map` classes. So far not fully documented, but used in `SpectrumDataset` and `SpectrumDatasetOnOff`
- In Gammapy v0.16 we started to adapt the `Estimator` API to work with `Dataset` objects as well
- An example for a multi-instrument analysis tutorial combining HESS / Fermi-LAT and HAWC can be found here: https://docs.gammapy.org/0.17/notebooks/analysis_mwl.html
- Introduced unified statistics API with `CashCountsStatistics` and `WStatCountsStatistics` (represent standard Li & Ma solutions using cash and wstat fit statistics)

SCIENCE VERIFICATION

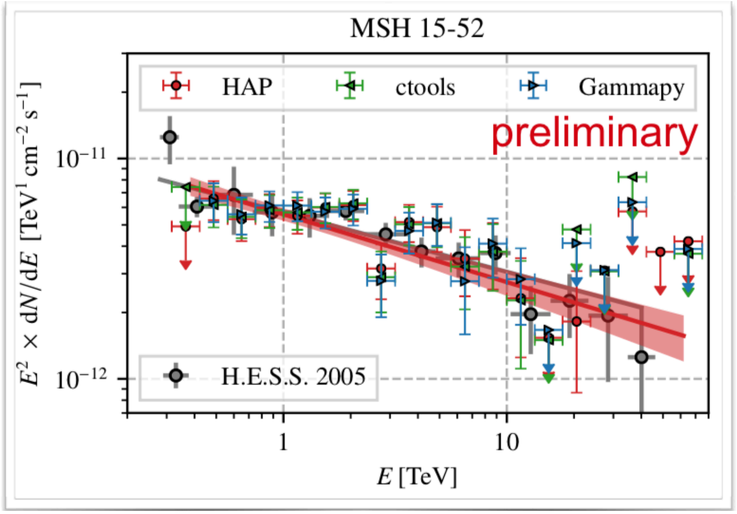
H.E.S.S. DL3 VALIDATION PAPER

For joint crab analysis see Cosimo's presentation…

- H.E.S.S public data released last year (https://www.mpi-hd.mpg.de/hfm/HESS/pages/dl3-dr1/)
- Validation of FITS data and open source tools Gammapy and ctools (lead by Lars Mohrmann from ECAP)
- Perfect reference analysis for a CI / benchmarking setup for the science tools
- Setup a science verification CI system: https://github.com/gammapy/gammapy-benchmarks/tree/master/validation



H.E.S.S. DL3



Source	Type	Size	Livetime	Period
Crab nebula	PWN	Point-like	1.9 h	2004
PKS 2155-304	AGN	Point-like	9.8 h	2006/2008
MSH 15-52	PWN	Small	9.1 h	2004
RX J1713.7-3946	SNR	Large	7.0 h	2004
Off data	-	-	20.7 h	2004/2005

SUMMARY

- Gammapy is a Python package for Gamma-Ray astronomy, based on Numpy, Astropy and Scipy
- It implements standard high level analysis use-cases for event based data, such as model fitting, flux points estimation, light-curve estimation, etc.
- It implements a joint-joint-likelihood API, that allows to combine data of multiple instruments
- **Gammapy v0.18 will be released on Oct 30th,** which will include a 98% final API. It will be the **"release candidate" for v1.0**. After v0.18 only bug fixes, docs improvements and minor clean up…
- Questions?

CONTACT AND SUPPORT

- **Slack:** gammapy.slack.com (quick questions, immediate help)
- **Github issues**: https://github.com/gammapy/gammapy/issues (feature requests & bug reports)
- Gammapy **mailing list** (not used much…):
  - gammapy@googlegroups.com
  - Archive: https://groups.google.com/forum/#!forum/gammapy
- Gammapy **dev meetings** (technical meetings in case you'd like to contribute):
  - Every Friday 10am on Vibe
- Regular **Gammapy user calls,** will be announced on the CTA mailing list