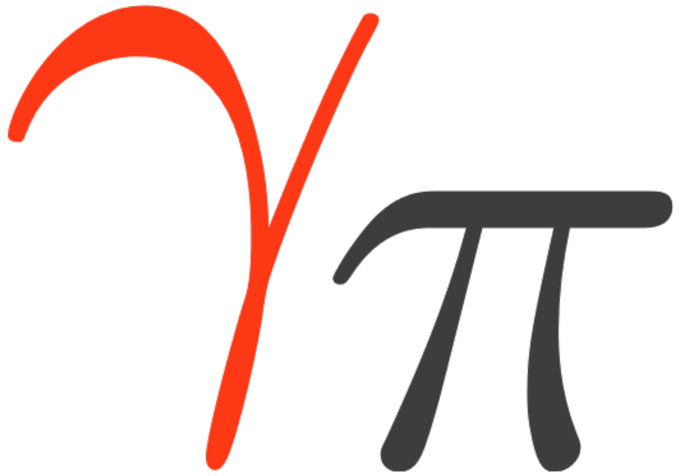
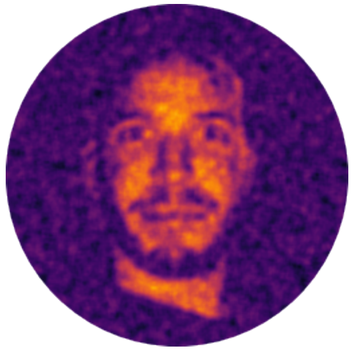
Gammapy: a Python Package for Gamma-Ray Astronomy

Axel Donath - for the Gammapy Team

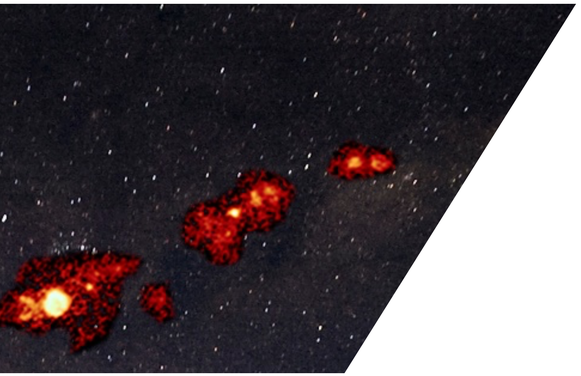


Scipy 2023 - Austin, TX - July 12th

About Me

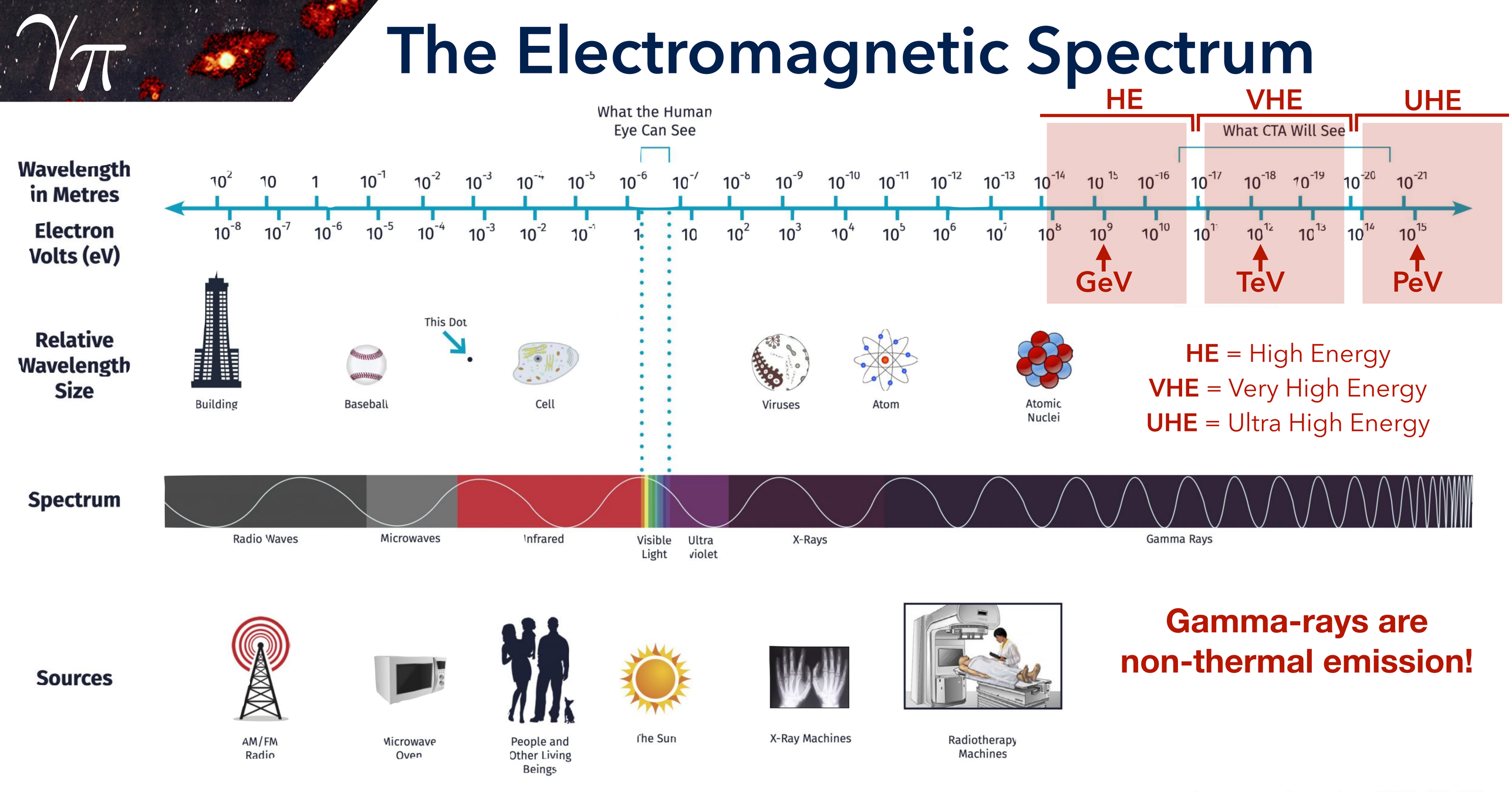


- Post-Doc @ Center for Astrophysics | Harvard Smithsonian
- Gamma-ray & X-ray astronomer, open source scientific software developer, self-taught “statistician”
- One of the lead developers of Gammapy a Python package for analysis of astronomical Gamma-ray data and “Science Tool” library for the Cherenkov Telescope Array
- I am also a member of the CHASC Astro-Statistics Collaboration, where I work on statistical methods for Gamma-ray & X-ray analysis. Most of the work can be found on: https://github.com/astrostat/
- You can find some more info about me on my webpage: https://axeldonath.com
- Presenting on behalf of the Gammapy team…



A short Introduction to Gamma-ray Astronomy

The Electromagnetic Spectrum



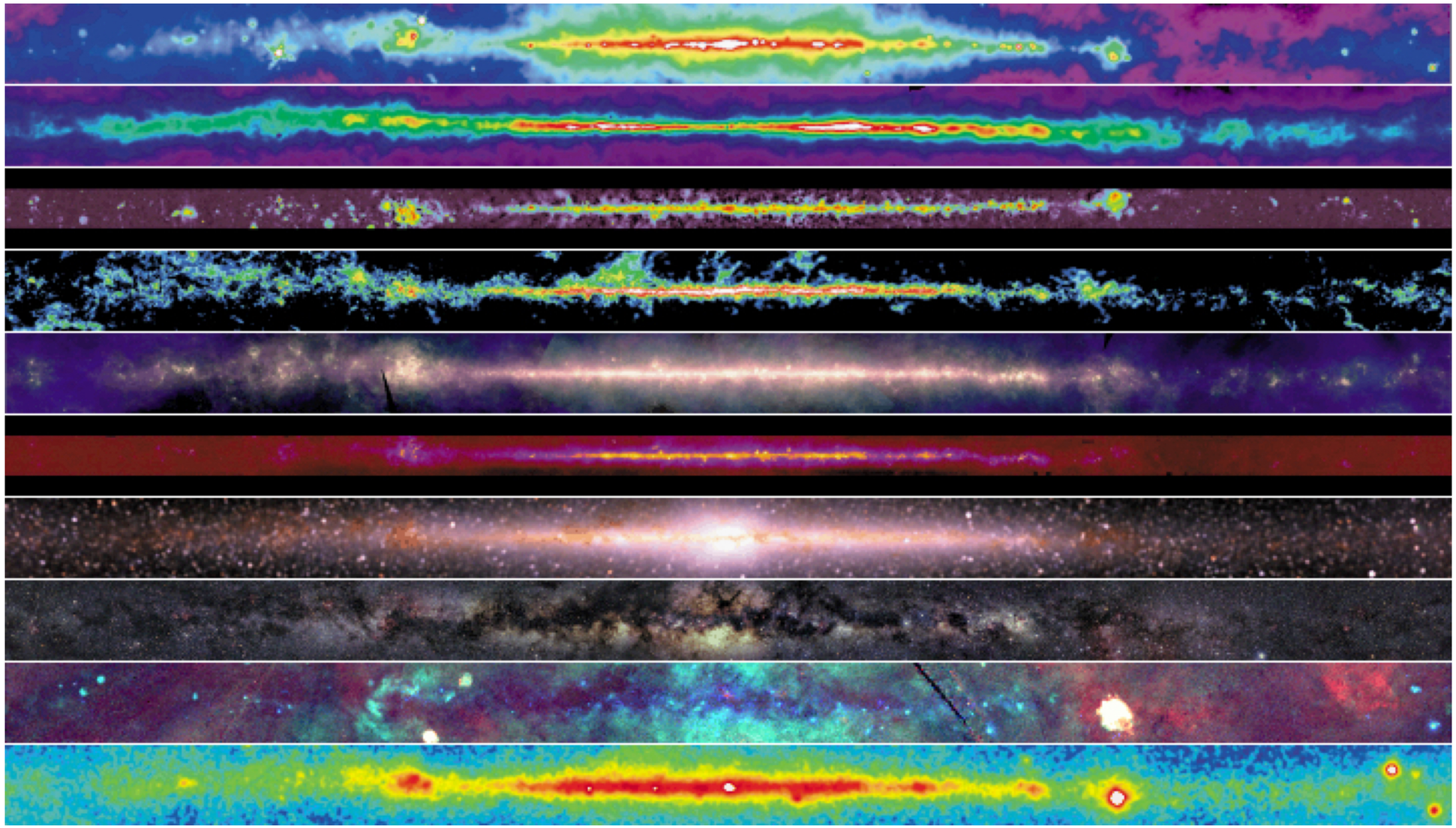
HE = High Energy
VHE = Very High Energy
UHE = Ultra High Energy

Gamma-rays are non-thermal emission!

Image credits: Vecteezv.com. Dragonartz.net. NAOI. NCi. CERN NASA

The Milky Way in MWL

Era of Multiwavelength (MWL) / Multimessenger (MM) Astronomy !

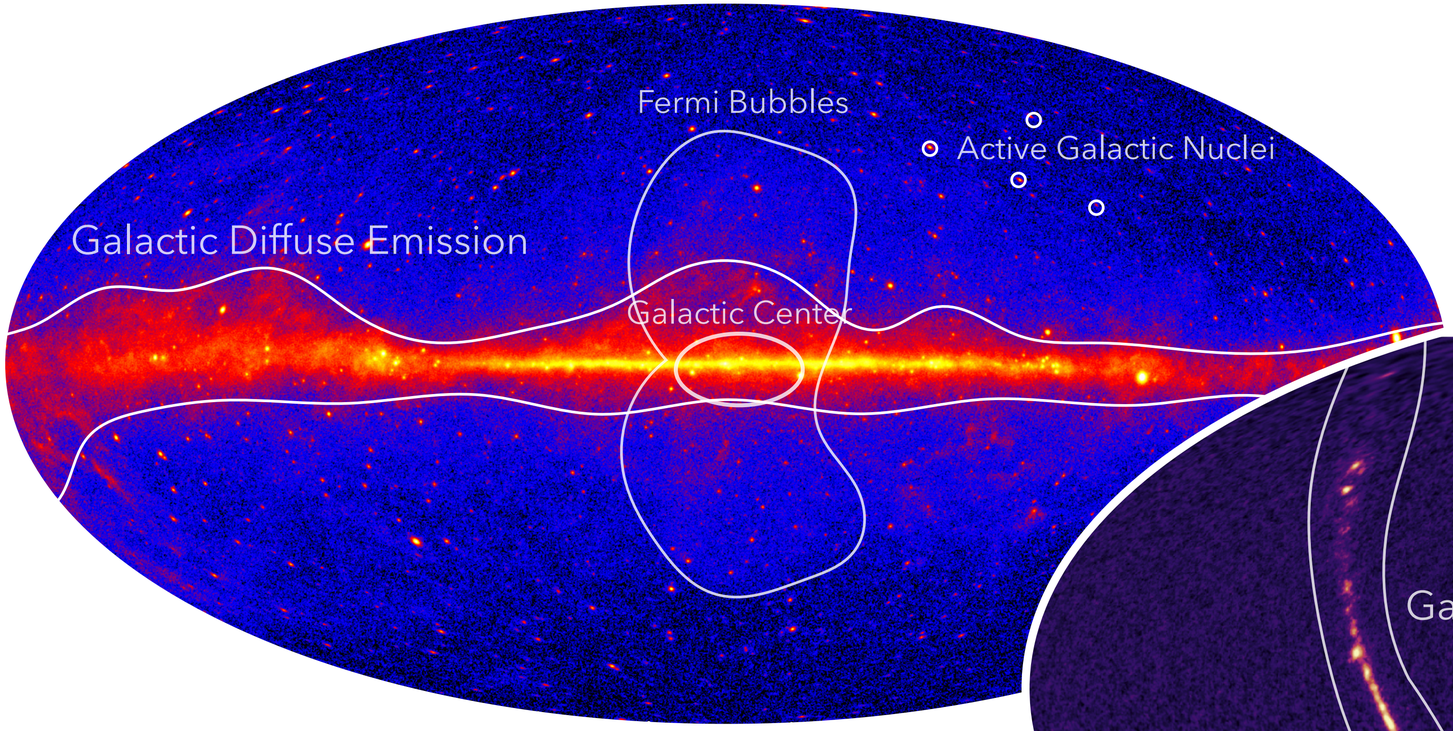


https://asd.gsfc.nasa.gov/archive/mwmw/mmw_images.html

The GeV - TeV Gamma-Ray Sky

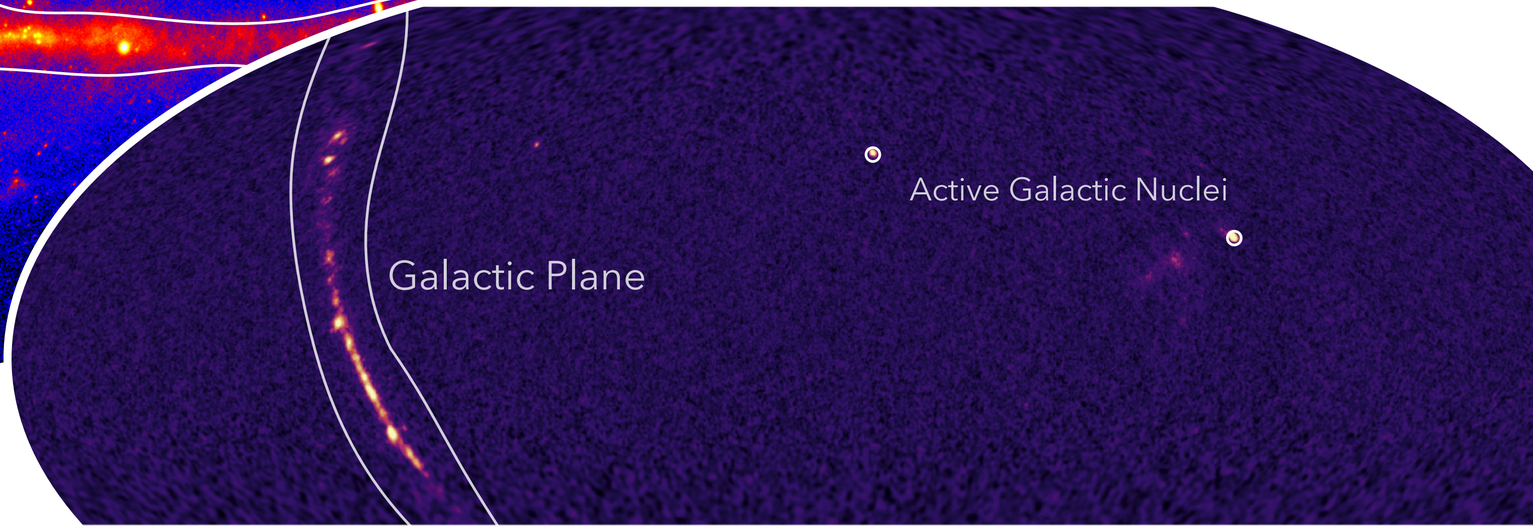
Lot's of interesting science questions:

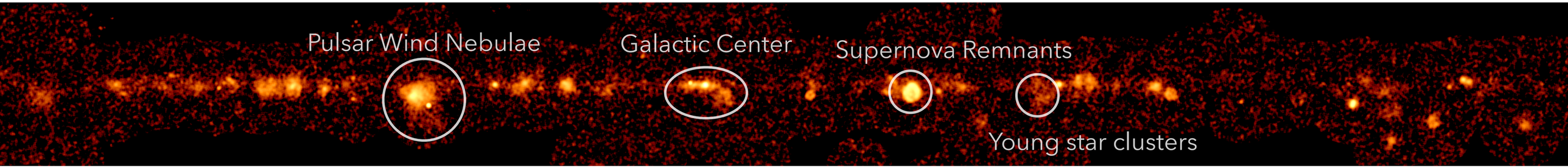
- Where are the acceleration sites of cosmic rays?
- Can particles reach PeV energies?
- Search for signatures of Dark Matter



Fermi-LAT all-sky counts map > 1 GeV

HAWC all-sky significance map > ~1 TeV





H.E.S.S. Galactic Plane Survey significance map >1 TeV

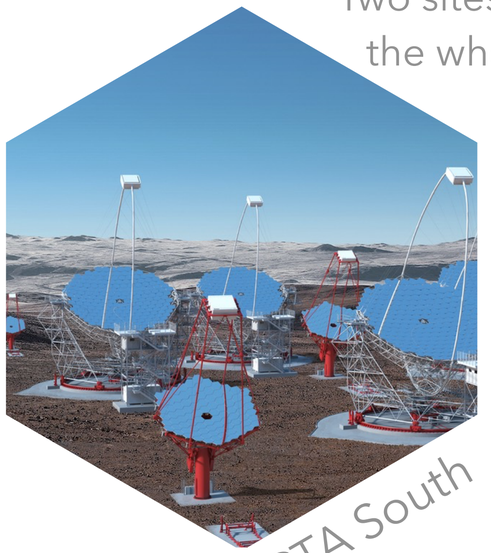
Gamma-Ray Instruments

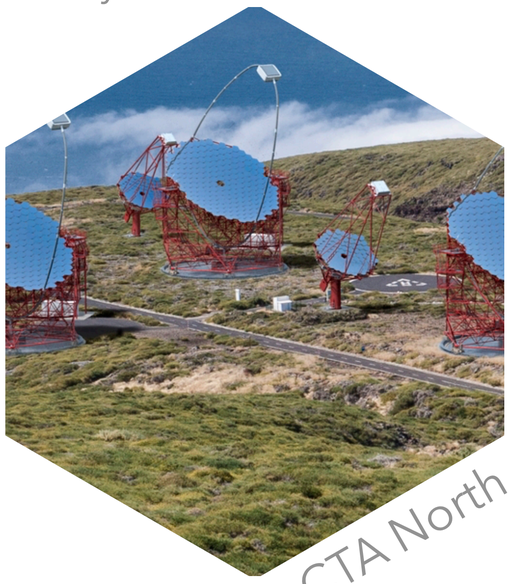
* Planned or being build

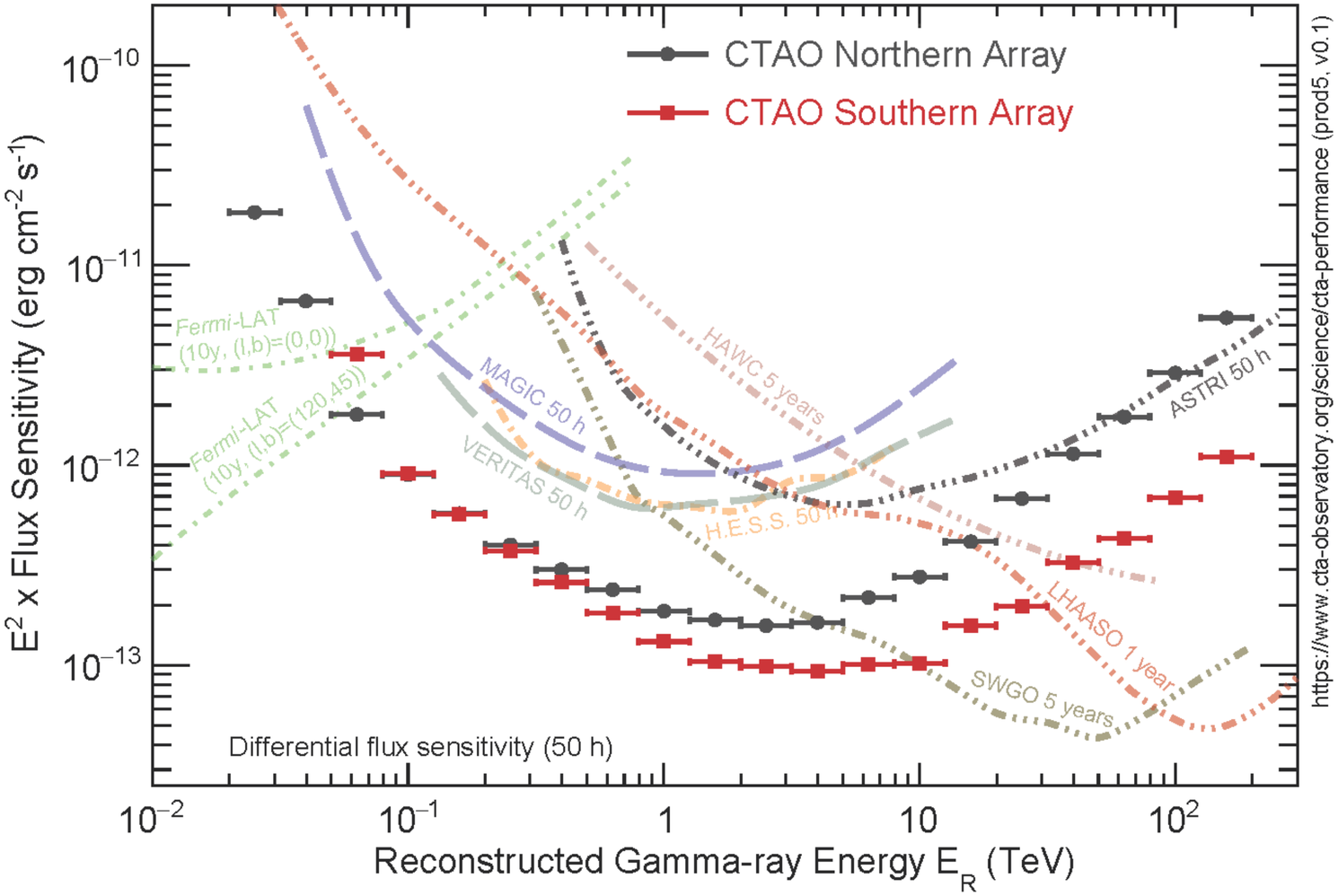
- Ground based **Imaging Atmospheric Cherenkov Telescopes**, H.E.S.S., VERITAS, MAGIC, FACT, CTA*
  - Pointing instruments with **good angular and energy resolution**. Short duty cycle, can only operate by night. Large effective area, suited for VHE range, above a few tens of GeV
- **Water Cherenkov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

Cherenkov Telescope Array Observatory

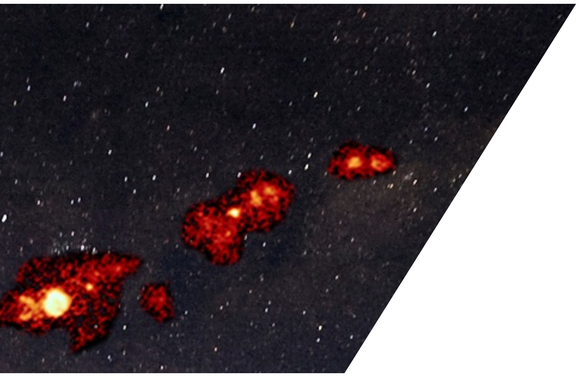
Two sites to access the whole sky…







- **CTAO will improve observation sensitivity ~10 times** compared to existing instruments
- **Improved angular resolution by a factor ~3**
- Will operate as an **open gamma-ray observatory for the first time**, with making data public after some disclosure period
- Gammapy has been selected in 2021 as **library for the CTA science tools** in an internal selection process



A short Introduction to Gamma-Ray Data Analysis

Binned Poisson Log-Likelihood

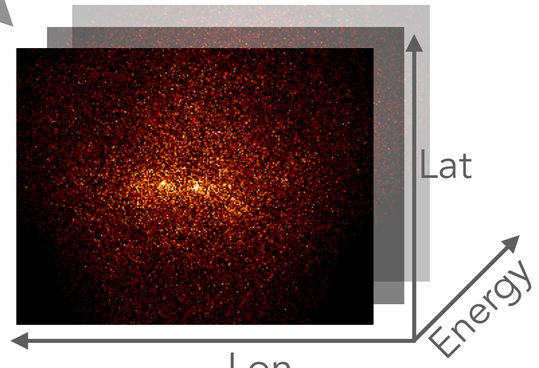
List of gamma-like events…

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…

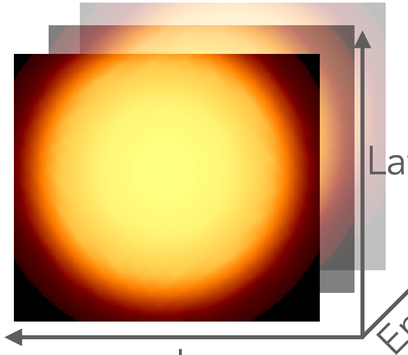
Counts

$$N_{Obs} =$$



Bkg Template

$$N_{Bkg} =$$



$$\cdot\, n_0 \cdot \left(\frac{E}{E_0}\right)^{-\Gamma}$$

"Cash statistics": summed over all "bins"

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

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

- Predicted counts are **computed per model component** ("source / object") and summed
- A **"global" background model** template with "correction parameters" is added

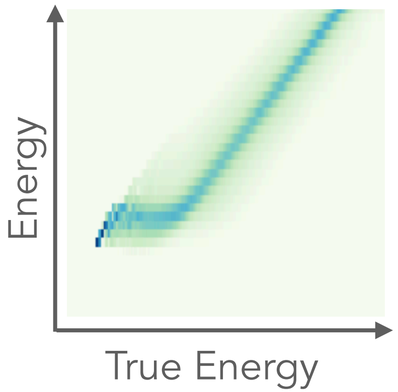
Data Model / Instrument Response

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

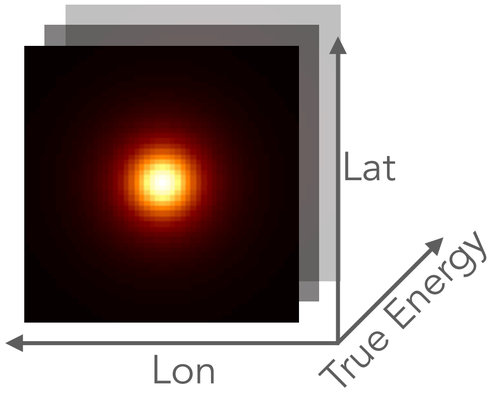
$$N_{Pred,Src} = \mathrm{EDISP}_{Src}(\mathrm{PSF}_{Src}(\mathscr{E}_{Src} \cdot f_{Src}))$$

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

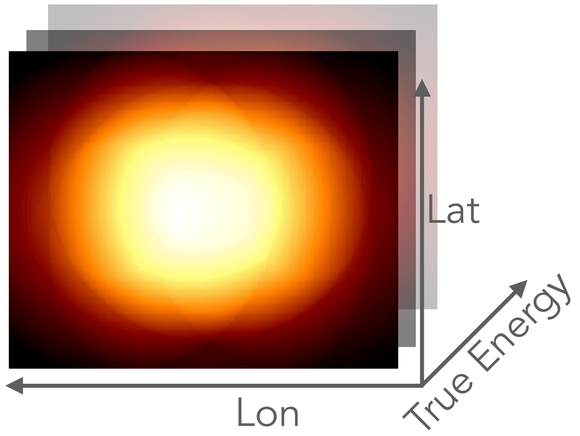
Energy Dispersion Matrix



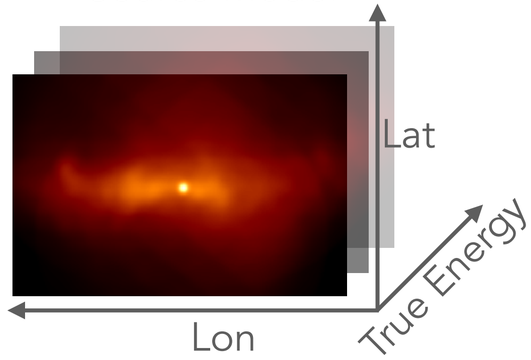
PSF Kernel



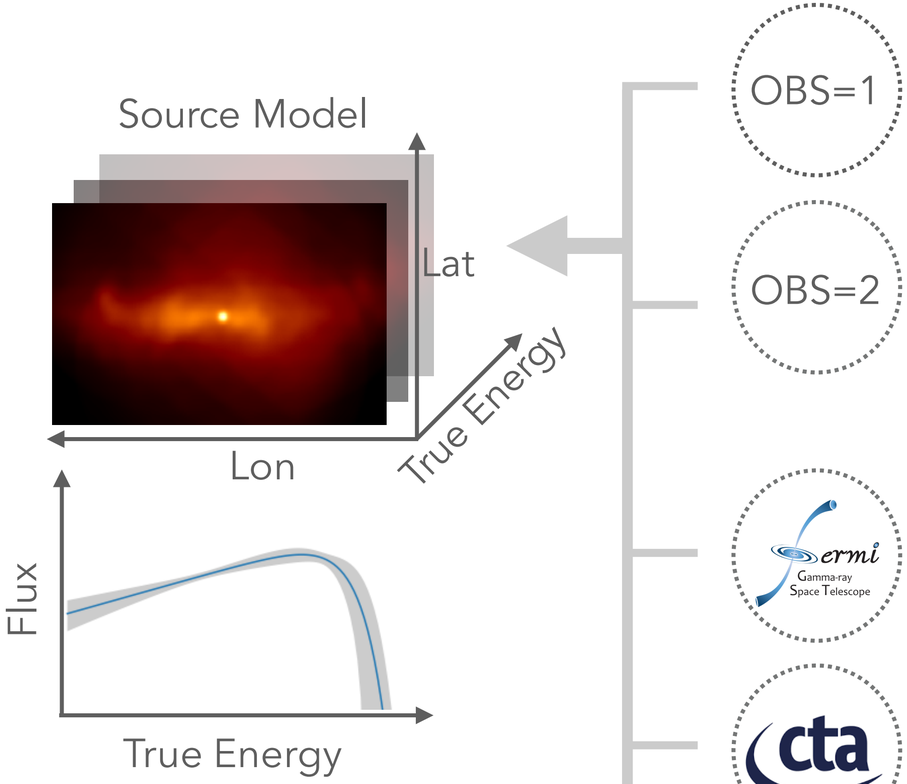
Exposure
(eff. area x lifetime)



Source Model



Joint Likelihood



OBS=1

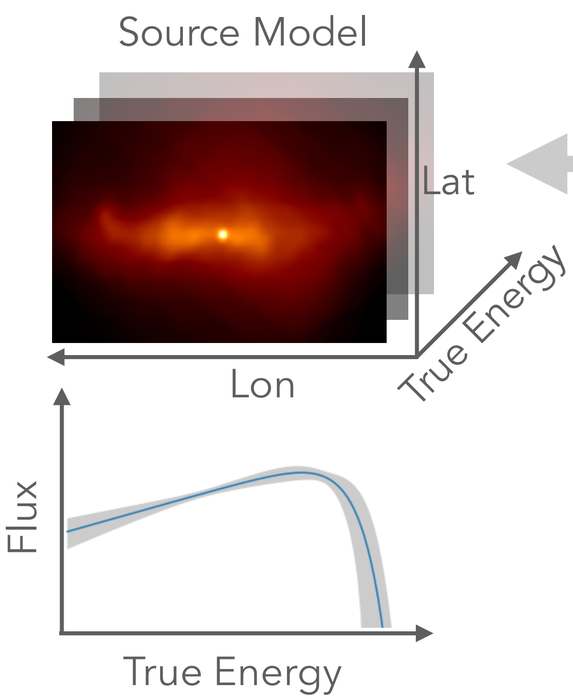
OBS=2

Fermi Gamma-ray Space Telescope

cta

H.E.S.S.

Joint Likelihood



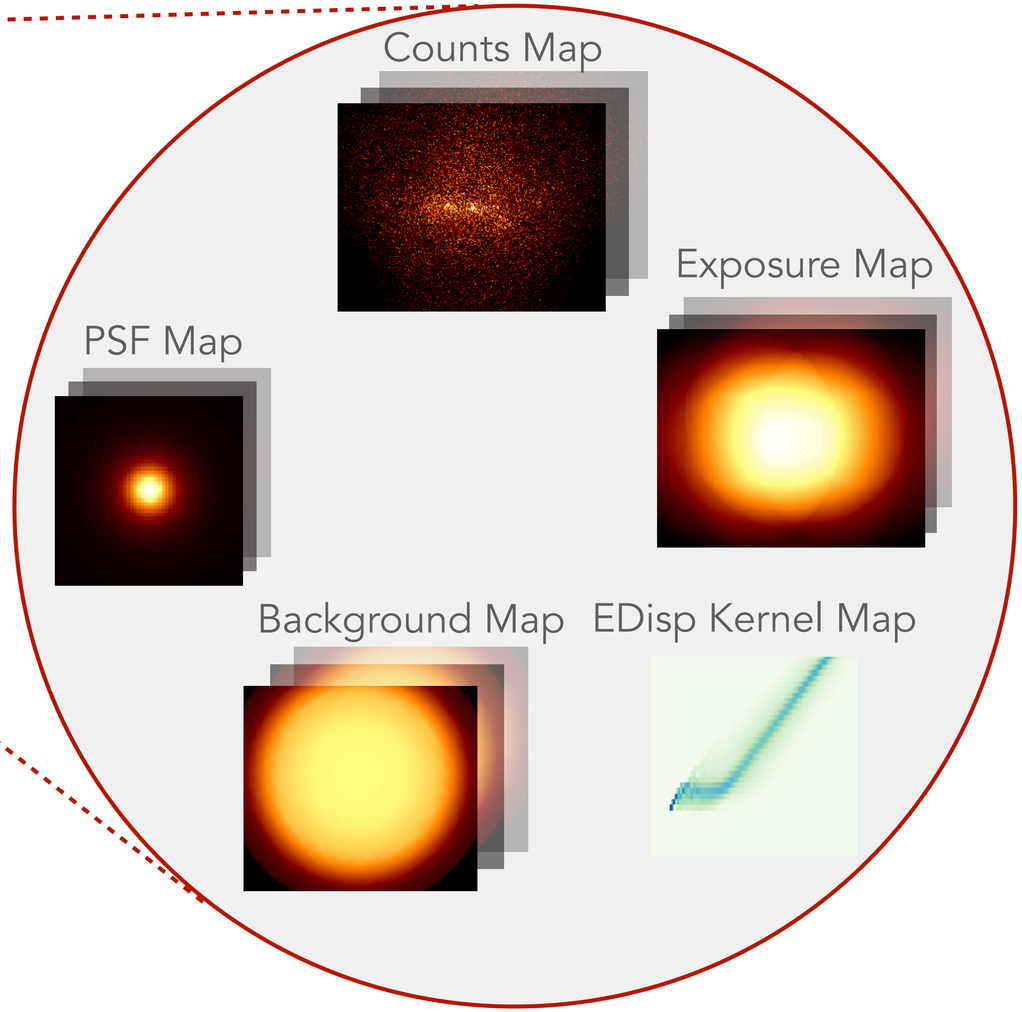
OBS=1

OBS=2

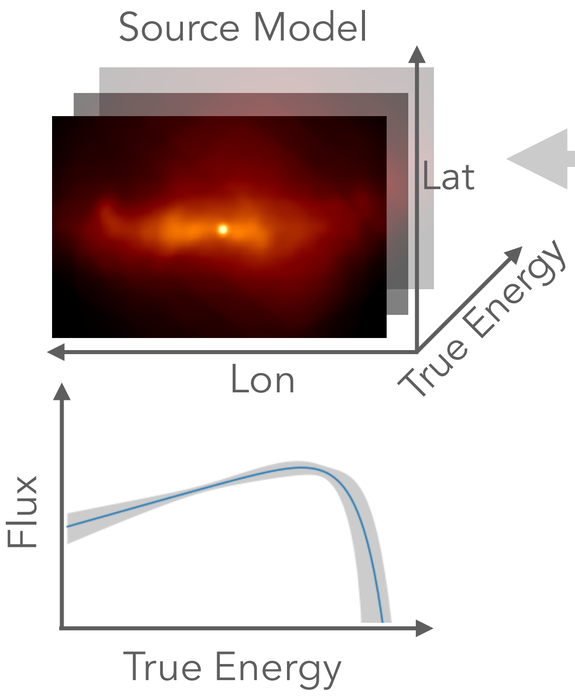
Fermi Gamma-ray Space Telescope

cta

H.E.S.S.



Joint Likelihood



OBS=1 $\mathscr{L}(x_1, \theta)$

$*$

OBS=2 $\mathscr{L}(x_2, \theta)$

$\cdots$

$\mathscr{L}(x_{n-2}, \theta)$

$*$

$\mathscr{L}(x_{n-1}, \theta)$

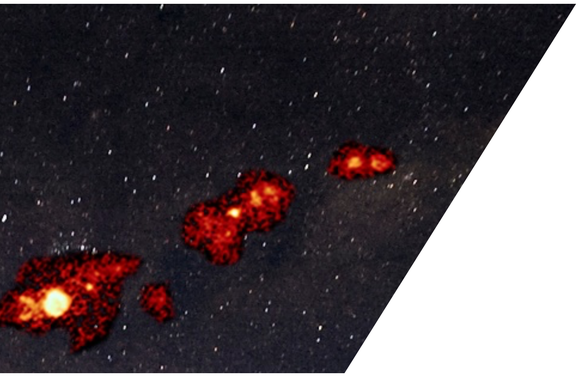
$*$

$\mathscr{L}(x_n, \theta)$

$$\mathscr{L}(x_{1\ldots n}, \theta)$$

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

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.**



Gammapy Package

Main Idea of Gammapy

Pointing γ-ray Observatories

cta

MAGIC

H.E.S.S.

VERITAS

HAWC
High Altitude Water Cherenkov
Gamma-Ray Observatory

Fermi
Gamma-ray
Space Telescope

All-sky γ-ray Observatories

GADF

Common
data format

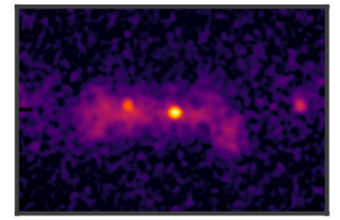
NumPy

SciPy

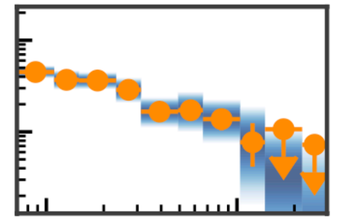
matplotlib

astropy

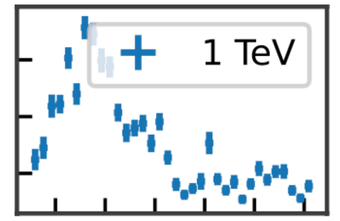
Sky Maps



Spectra



Lightcurves



Dependencies

Optional dependencies: bring in useful functionality

Optional dependencies

Required dependencies

Pydantic

Configuration

PyYAML

YAML I/O

matplotlib

Plotting, visualisation

$ click_

Command line tools

Sherpa

iminuit

Optimisation, sampling

healpy

Healpix maps

jupyter

Tutorial notebooks

astropy

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

SciPy

Interpolation, minimisation, FFT convolution, etc.

NumPy

ND-data structures and computations

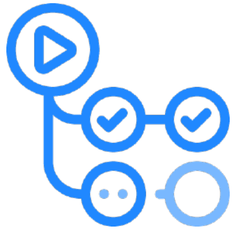
Development and CI setup

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

GitHub actions used to run CI on each PR and a release pipeline

Codecov

codecov.io used for monitoring of code test coverage



Docs

Docs are build and deployed using GitHub actions:
https://docs.gammapy.org/

Sphinx used to build the documentation, Sphinx Gallery for the tutorial gallery

Pytest used for testing

pytest

Defined as pre-commit hooks

Black code formatting used for a consistent code format.

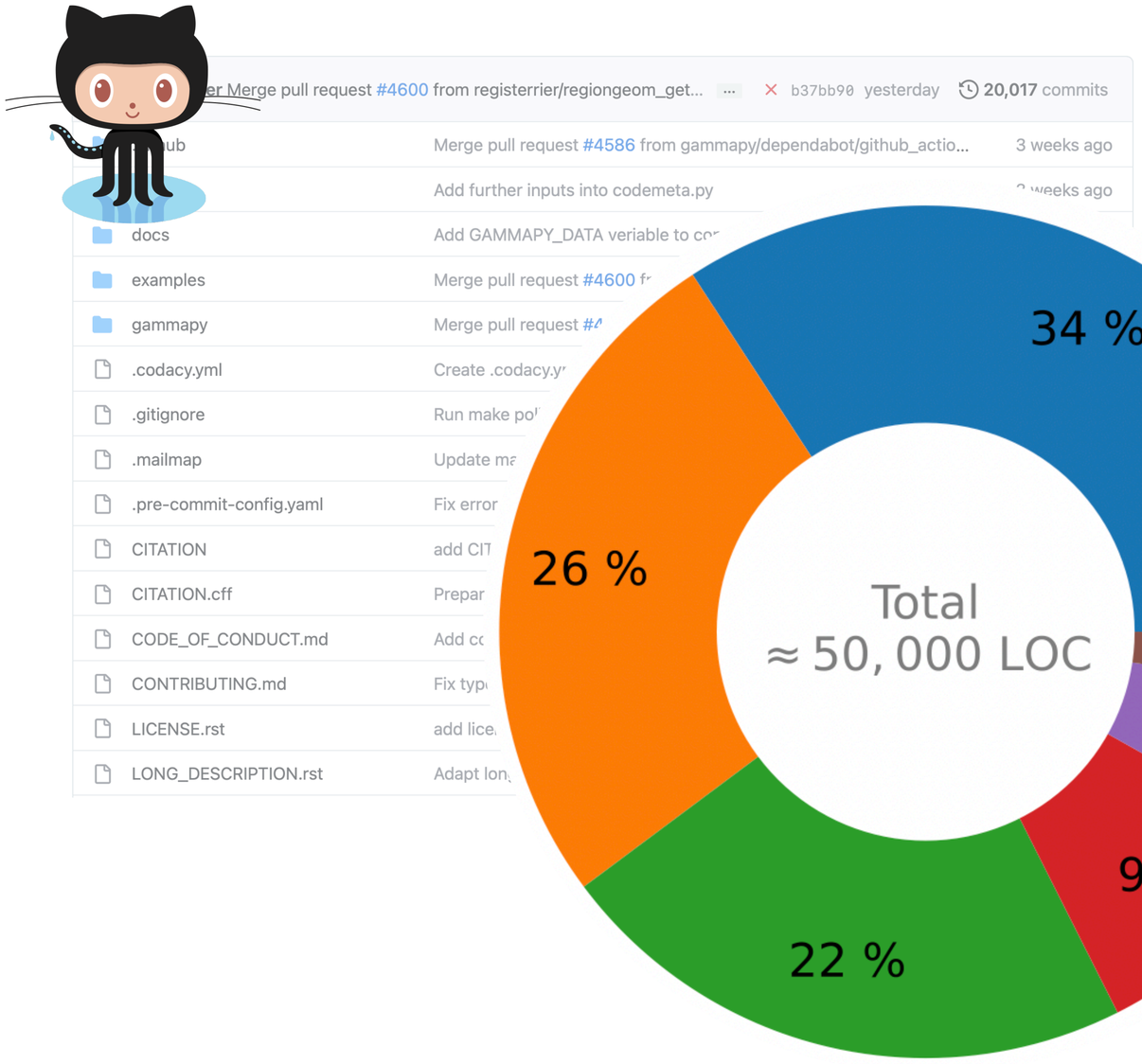
Isort to automatically **sort and format imports**

Flake 8:

Flake 8 to **check PEP8** standard

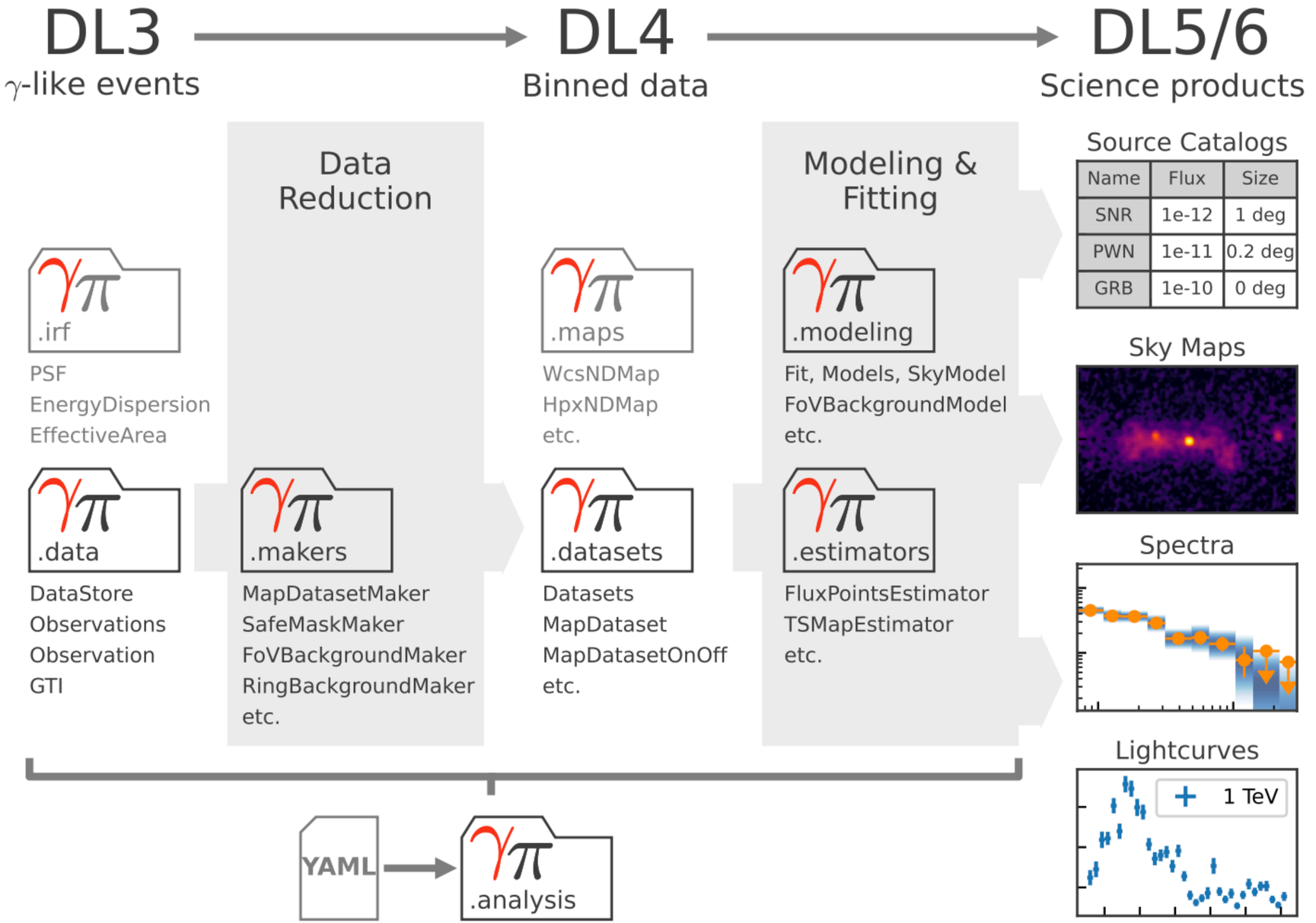
Some Code Statistics

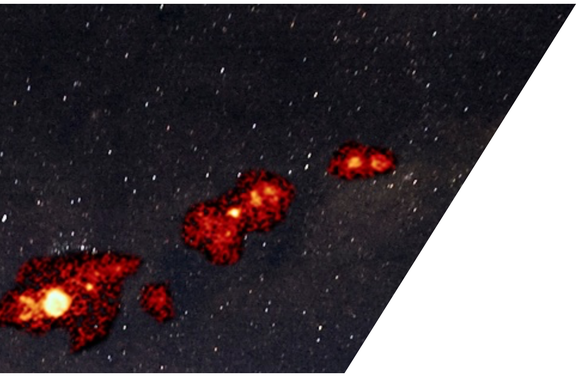
https://github.com/gammapy/gammapy-v1.0-paper/blob/main/src/figures/codestats.py



- We have **achieved a healthy distribution of approximately 1/3 code, 1/3 docs and 1/4 tests**, however docs and test could still improve…
- There is a **small amount of Cython code** that re-implements certain masked Numpy expressions to **speed up critical parts of the likelihood fit**…

Gammapy Subpackage Structure





Code Examples

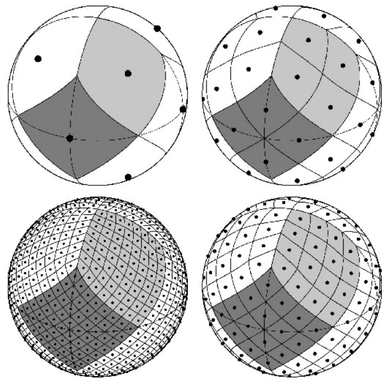
Gammapy Maps

https://docs.gammapy.org/1.0.1/tutorials/api/maps.html

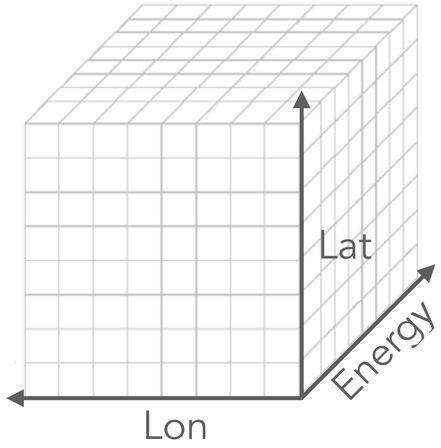
```python
from gammapy.maps import Map, MapAxis
from astropy.coordinates import SkyCoord
from astropy import units as u

skydir = SkyCoord("0d", "5d", frame="galactic")

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

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


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

# Create a region map
region = "galactic;circle(0, 5, 1)"
m_region = Map.create(
    region=region,
    map_type="region",
    axes=[energy_axis]
)

print(m_wcs, m_hpx, m_region)
```

- **N-dimensional coordinate aware data structures** for storing gamma-ray data, with arbitrary number of non-spatial dimensions, such as energy, time or a label for the data class
- **Uniform API for WCS, HEALPix and region** based pixelization schemes
- Support for `MapAxis`, `TimeMapAxis`, `LabelMapAxis`
- There are `ndcube` and `astropy.nddata` and `xarray`, however focus is on the uniform API for WCS, HEALPix and region based pixelization, as well as subtile but important differences, such as keeping track of both the bin edges and centers for non-spatial axes…

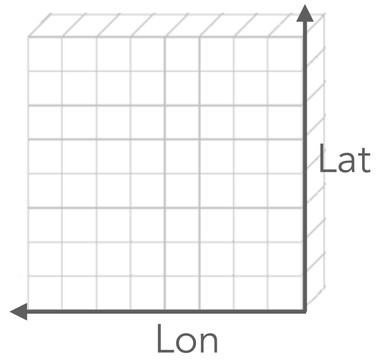
HEALPix



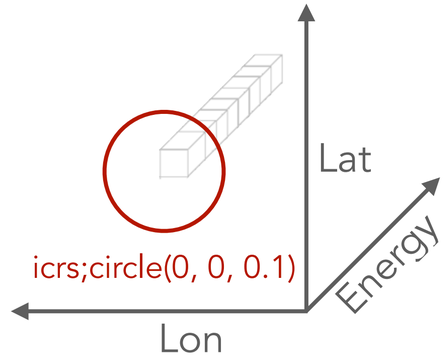
ND-Cubes



Images



Region based ND-spectra



Gammapy Makers

https://docs.gammapy.org/1.0.1/tutorials/api/makers.html

```
import astropy.units as u

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

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

obs = data_store.obs(23523)

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

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

empty = MapDataset.create(geom=geom)
```

```
maker = MapDatasetMaker()

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

bkg_maker = FoVBackgroundMaker(
    method="scale",
)

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

Partly inspired by the API from scikit-learn, however the model state is stored independently…

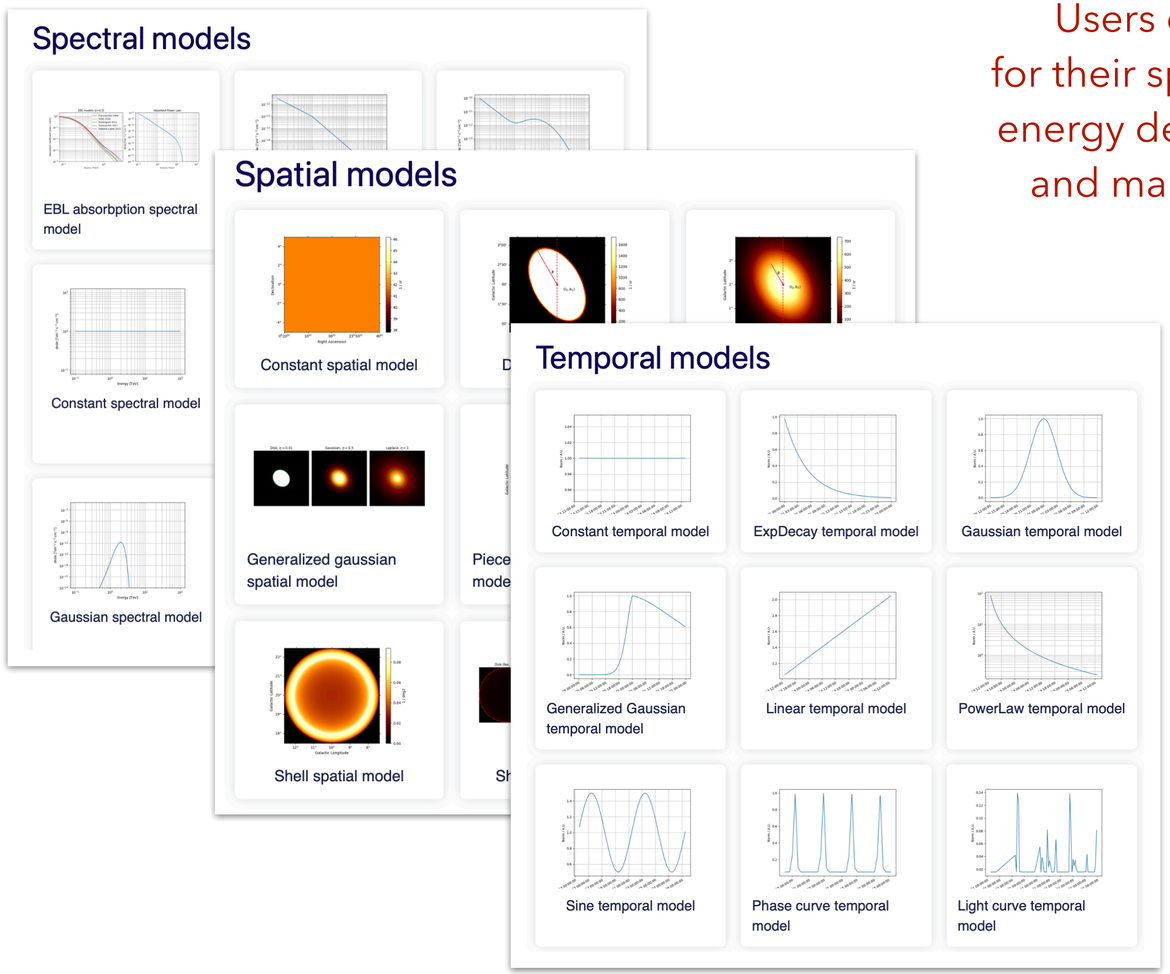
- `Maker` are configurable, stateless objects the represent an algorithm / data reduction step
- Data represented by a `MapDataset` is passed between the makers is modified and holds the state
- This allow users to define data reduction chains, without access to data and also implement custom steps
-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

Gammapy Models

https://docs.gammapy.org/1.0/user-guide/model-gallery

https://docs.gammapy.org/1.0.1/tutorials/api/models.html

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

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

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


# define a temporal model
constant = ConstantTemporalModel()

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

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

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

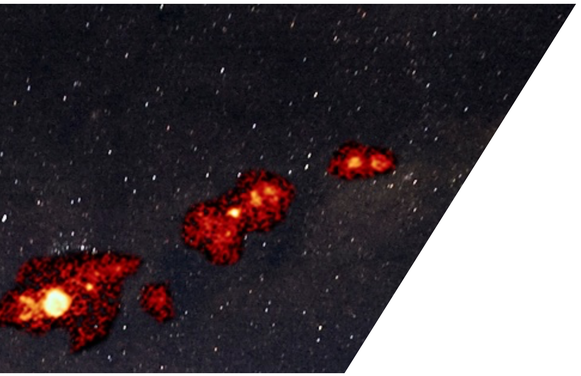


Users can implement **custom models** for their specific use-case, such as modeling energy dependent morphology of a source and make it available to Gammapy via a **registry system**

All **models can be serialized to YAML:**

```yaml
components:
-   name: pwl-gauss-model
    type: SkyModel
    spectral:
        type: PowerLawSpectralModel
        parameters:
        -   name: index
            value: 2.0
        -   name: amplitude
            value: 1.0e-12
            unit: cm-2 s-1 TeV-1
        -   name: reference
            value: 1.0
            unit: TeV
    spatial:
        type: GaussianSpatialModel
        frame: icrs
        parameters:
        -   name: lon_0
            value: 0.0
            unit: deg
        -   name: lat_0
            value: 0.0
            unit: deg
```

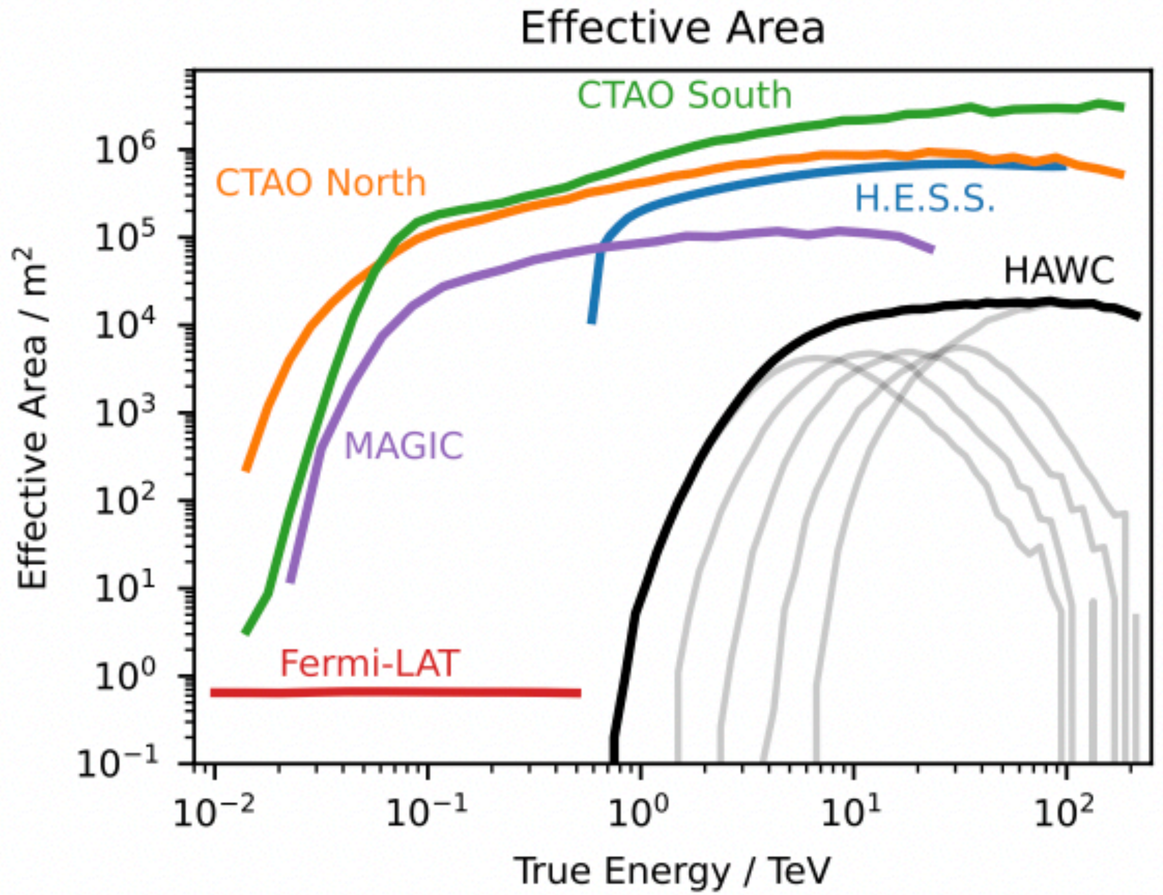
$$f_{Src} = f_{Spectral}(E) \cdot f_{Spatial}(E, l, b) \cdot f_{Temporal}(t)$$

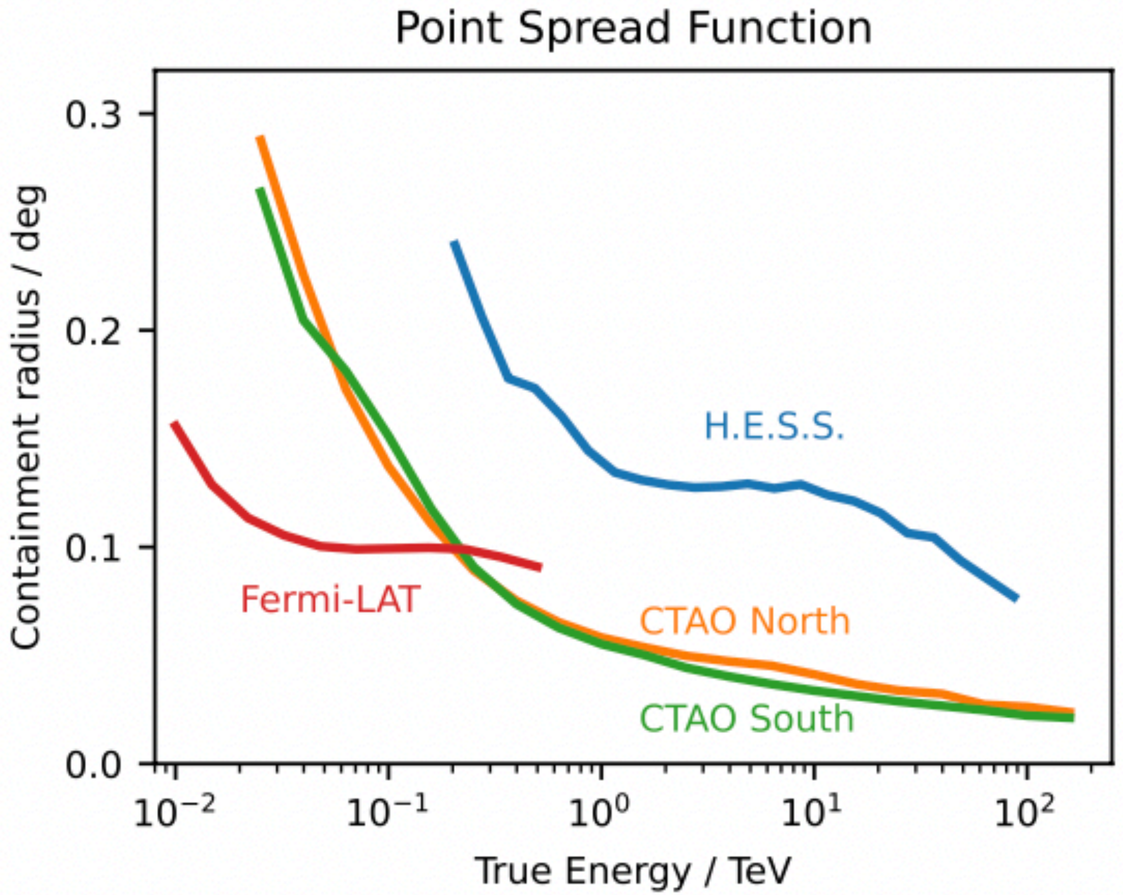


Analysis Examples

Visualizing IRFs

https://github.com/gammapy/gammapy-v1.0-paper/blob/main/src/figures/irfs.py

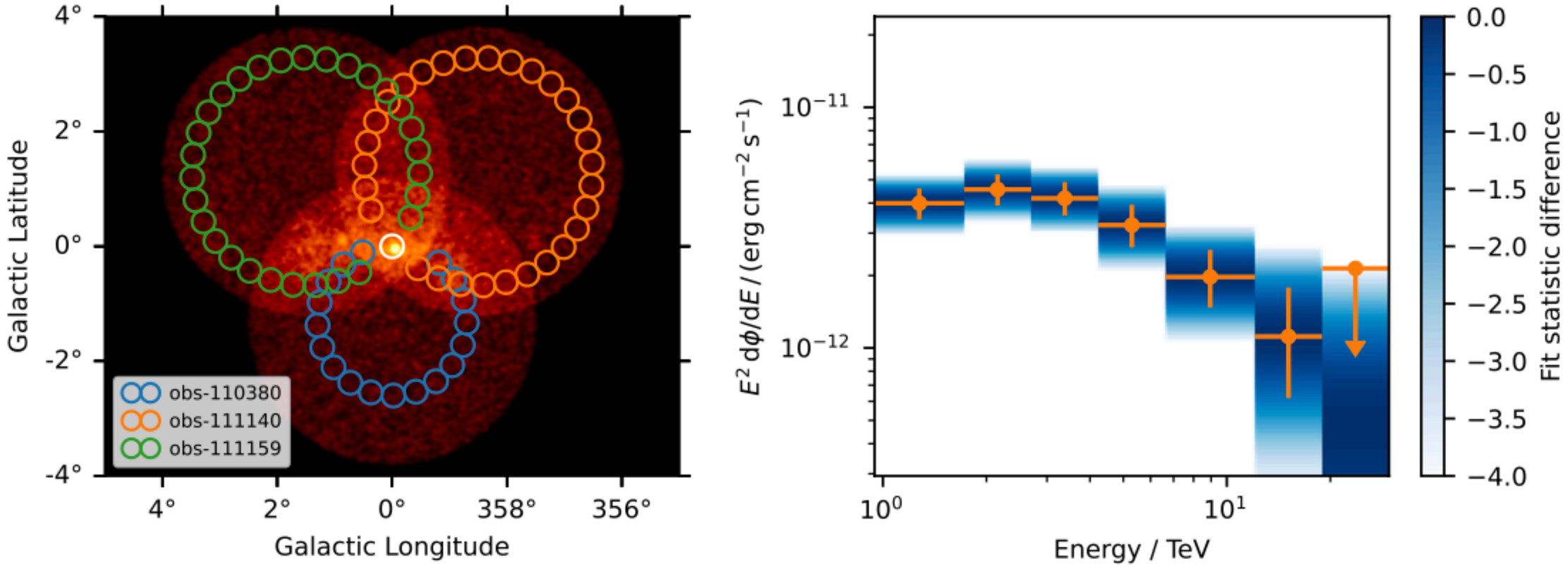




This seems trivial, but the sole fact that users can **read and visualize IRFs from all these different instruments and telescopes with a single software** package is novel and unique and required years of work from the community!

Example I: a CTA Spectral Analysis

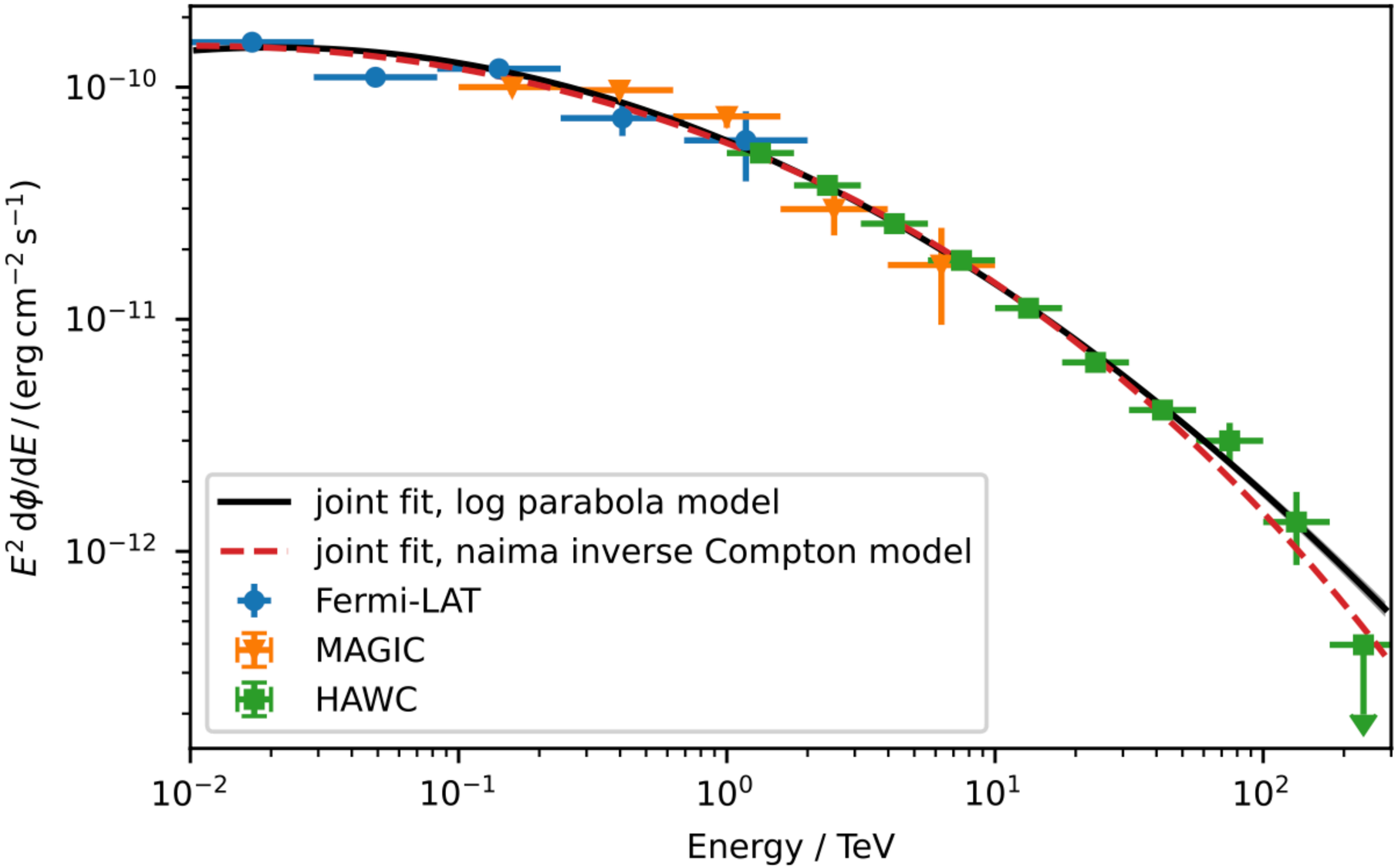
https://github.com/gammapy/gammapy-v1.0-paper/blob/main/src/figures/cta_galactic_center.py



- This is a **"classical" gamma-ray analysis**: measuring a spectrum of a source and estimating the background from these ring-like arrangements of multiple regions.
- This uses simulated data from CTA and also exports the **likelihood per energy bin (**shown as the blue colored band). This is a lot more information then e.g. reporting only errors or upper limits for flux points

Example II: Multi-Instrument Analysis

https://github.com/gammapy/gammapy-v1.0-paper/blob/main/src/figures/multi_instrument_analysis.py



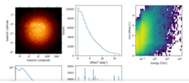
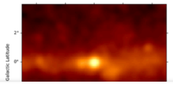
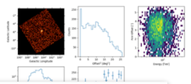
- A spectral fit c**ombining data from Femi-LAT, MAGIC and HAWC**
- The combination of data leads to smaller errors, see e.g. Nigro et al 2019, which is important for analysis of gamma-ray data which is usually limited by statistics.
- Via Python wrapper classes one can easily i**nterface to 3rd party packages**, such as Naima, agnpy or jetset to fit astrophysical emission models for inverse Compton emission to gamma-ray data

Tutorial Gallery

https://docs.gammapy.org/1.0/tutorials/

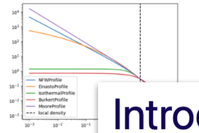
Data exploration

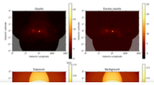
These three tutorials show how to perform data exploration with Gammapy, providing an introduction to the CTA, H.E.S.S. and Fermi-LAT data and instrument response functions (IRFs). You will be able to explore and filter event lists according to different criteria, as well as to get a quick look of the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

Package / API

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





Dark matter s
spectral mode

Fitt

Mask maps

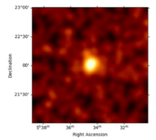
Modelling

Models

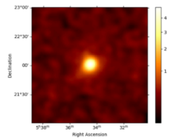
Introduction

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.

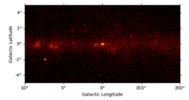
The first tutorial is an overview on how to perform a standard analysis workflow using the high level interface in a configuration-driven approach, whilst the second deals with the same use-case using the low level API and showing what is happening *under-the-hood*. The third tutorial shows a glimpse 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

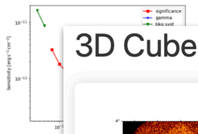


Data structures

Data analysis

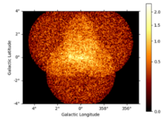
The following set of tutorials are devoted to data analysis, and grouped according to the specific covered use cases in spectral analysis and flux fitting, image and cube analysis modelling and fitting, as well as time-dependent analysis with light-curves.

1D Spectral

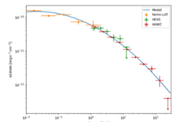
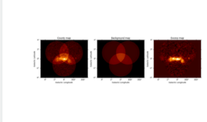
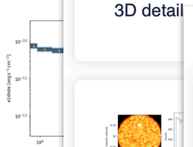


Point s

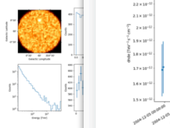
3D Cube



3D detail

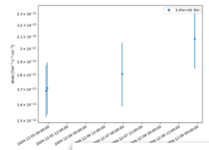




Spe

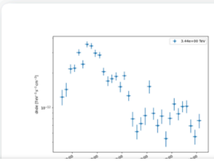
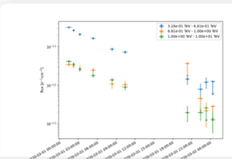
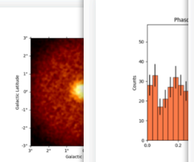


Event

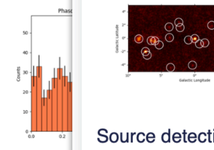
Time



Li

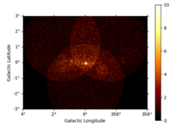
3D map



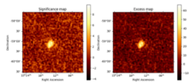
Pul

2D Image

Source detection and significance maps



2D map fitting



Ring background map

launch binder

`Download Python source code: model_management.py`

`Download Jupyter notebook: model_management.ipynb`

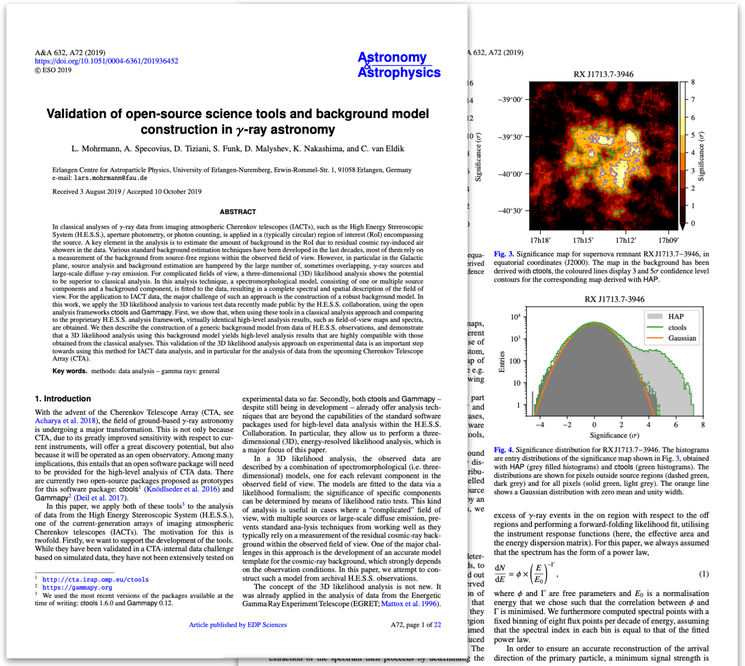
Provide users with a selection of the most important **analysis scenarios** as well as **API specific tutorials**. In total **37 tutorials.**

Science Validation

Validation of standardized data formats and tools for ground-level particle-based gamma-ray observatories

Albert et al. 2022

https://doi.org/10.1051/0004-6361/202243527



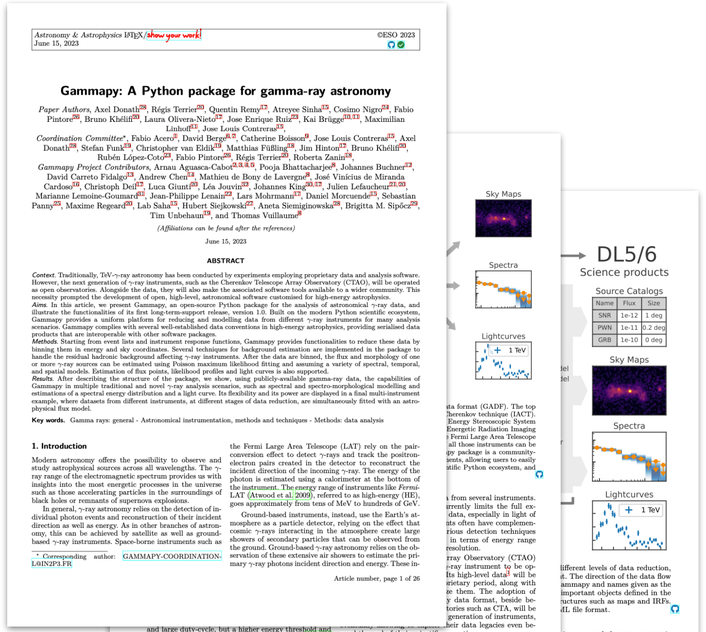
Validation of open-source science tools and background model construction in γ-ray astronomy

Mohrmann et al 2019.

https://doi.org/10.1051/0004-6361/201936452

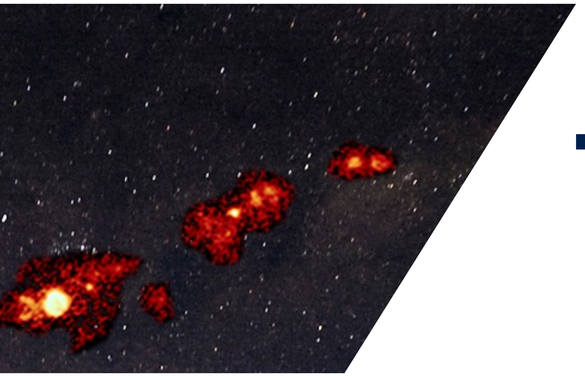
- Re-analysis of published results from the **HAWC Observatory and H.E.S.S. experiment using Gammapy** for multiple source and analysis scenarios, such as spectral fitting, morphology fitting, timing analysis
- Both found **excellent agreement** between the re-analysis and the previously published results.
- Important work to **convince the sub-communities of the quality of the results of open source tools**
- We found the **acceptance of Gammapy increased noticeably within the communities** after those papers have been published!
- We hope that further work follows, e.g. for LHASSO, VERITAS etc.

Gammapy v1.0 Paper



Donath et al. 2023 (TBR)

- In the past 1.5 years the Gammapy developers have written a paper about the software. A lot of the material of this talk is based on it…includes many more application examples
- **Written collaboratively and openly** on https://github.com/gammapy/gammapy-v1.0-paper
- It uses the tool show your work! https://show-your.work/ to make the paper reproducible and automatize the build process.
- Accepted for publication by **Astronomy & Astrophysics,** in preparation
- Will serve as the main reference for the LTS version, together with the exact version used in publications cited via Zenodo
- Provides **academic credit to the contributors**, which is important for their career development (!)



The Gammapy Project

"**Gammapy** is not only a software package, but a **community** of gamma-ray astronomers interested in **collaboration** on methods and **open and reproducible science**"

Project Organization

Project Managers

"Non-technical executive leads"

Lead Developers

"Technical executive leads"

Sub-package maintainers

"Lead the maintenance of sub-packages"

Contributors

Coordination Committee (CC)

"Promotes, coordinates and steers Gammapy developments"

Consists of representatives of the contributing institutions:

MAX-PLANCK-INSTITUT FÜR KERNPHYSIK HEIDELBERG

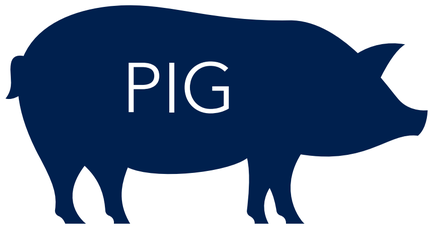
DESY.

APC, CNRS, FRANCE

irfu cea saclay

UNIVERSIDAD COMPLUTENSE MADRID

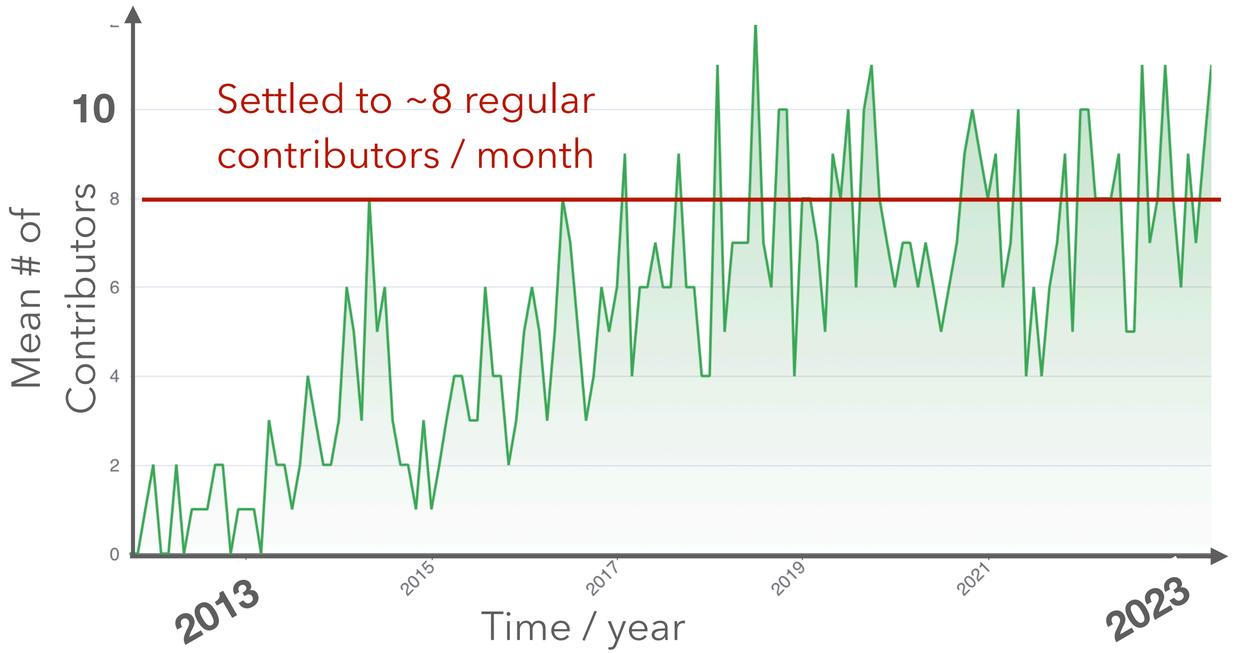
l'Observatoire de Paris

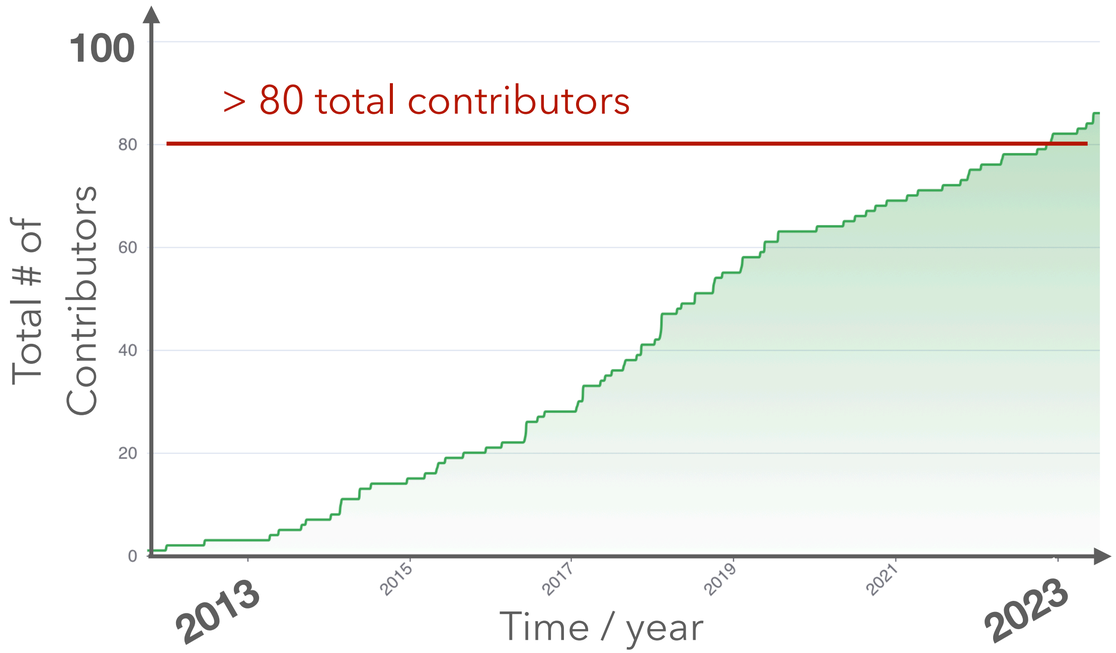


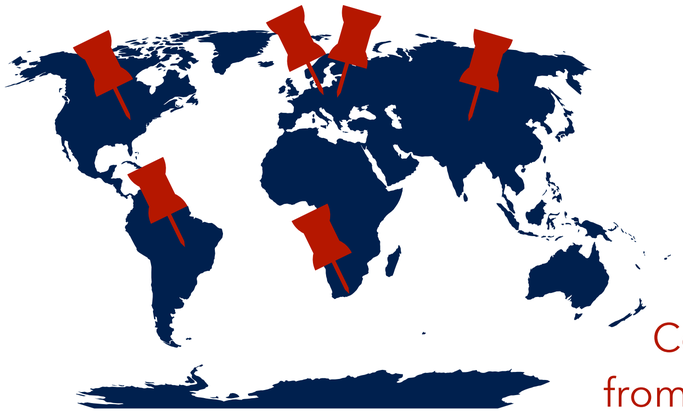
We have P**roposals for Improving Gammapy** (PIG), which follow the idea of PEPs, NEPs etc.

https://gammapy.org/team.html

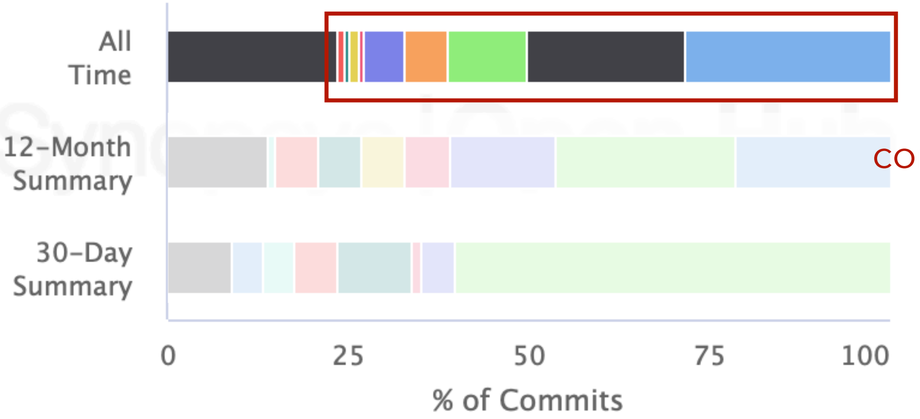
Some Contributor Statistics







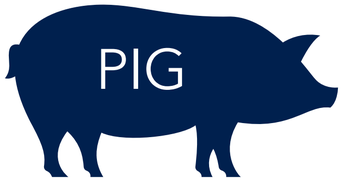
Contributions from >10 countries

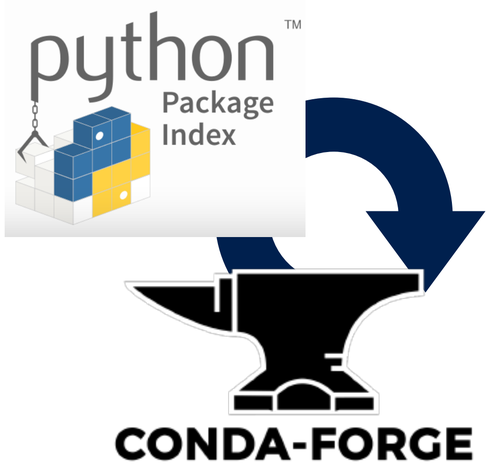


Commits follow a "Pareto" distribution: 20% of the top contributors did 80% of the commits

Release Cycle and Procedures

- After first LTS v1.0 release in Nov 2022, we introduced a release cycle following:
  - PIG 23 https://docs.gammapy.org/1.1/development/pigs/pig-023.html
  - Minor releases every ~6 months, **LTS releases every ~2 years***, bug fix releases as required.
  - All Releases are **archived on Zenodo**: https://zenodo.org/record/8033275
- We have a developed a minimal Zenodo authorship policy:
  - PIG 24 https://docs.gammapy.org/1.1/development/pigs/pig-024.html
  - The Zenodo author list includes **all people that contributed at least one commit** since the last LTS release in alphabetical order. After an LTS release the author list is "reset".
  - This was a compromise to avoid an ever-growing author list and give new contributors a chance to gain visibility. Non-code contributors can always **request to be part of the author list**, approved by the CC.
  - I'd be happy to hear how other projects have addressed the same question for Zenodo?



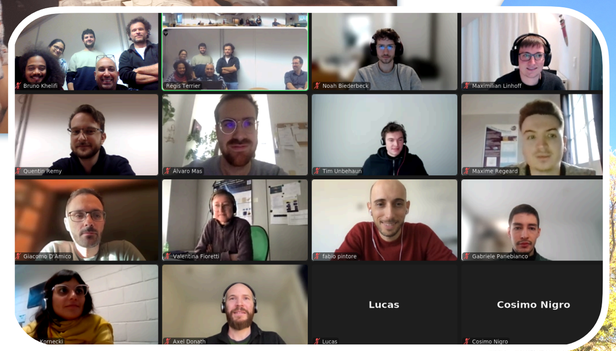


* I know some projects in the scientific Python community have turned away from the idea of supporting LTS releases, I'd be happy to learn more about the reasons…

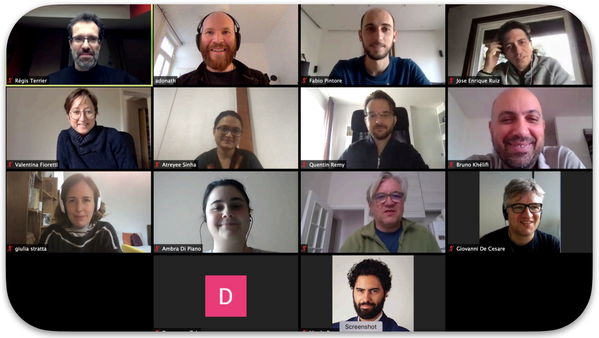
Coding Sprints / Dev Meetings

https://github.com/gammapy/gammapy-meetings/blob/master/coding-sprints/README.md

Madrid 2023



Hybrid coding sprint, Paris 2022



Remote coding sprint 2020

Pre-covid in Granada 2019

- **We organize 2-3 coding sprints a year**, which proved to be effective for:
  - Helping new contributors to get started
  - Make project organization, intense discussions among developers
  - Getting work done before releases…
  - Sometimes users would join and get f2f support for important projects…
- **Gammapy dev meetings** (technical meetings for contributors):
  - Coordinate every day work, discussing PRs and doing code review
  - Every Friday 2pm (CEST) / 8am (EST) on Zoom (direct link in the Slack #dev channel)

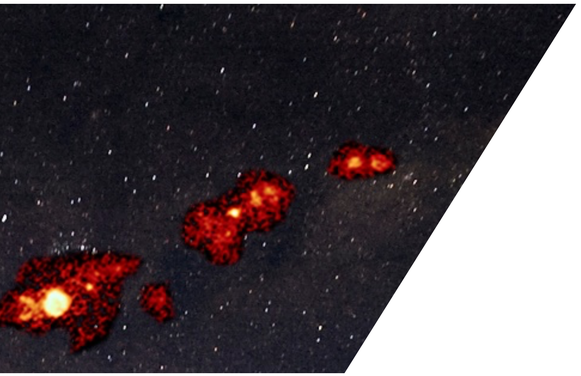
Community / User Support

- **Slack**: gammapy.slack.com (quick questions, immediate help, preferred by most users)

- **GitHub issues**: https://github.com/gammapy/gammapy/issues (feature requests & bug reports)

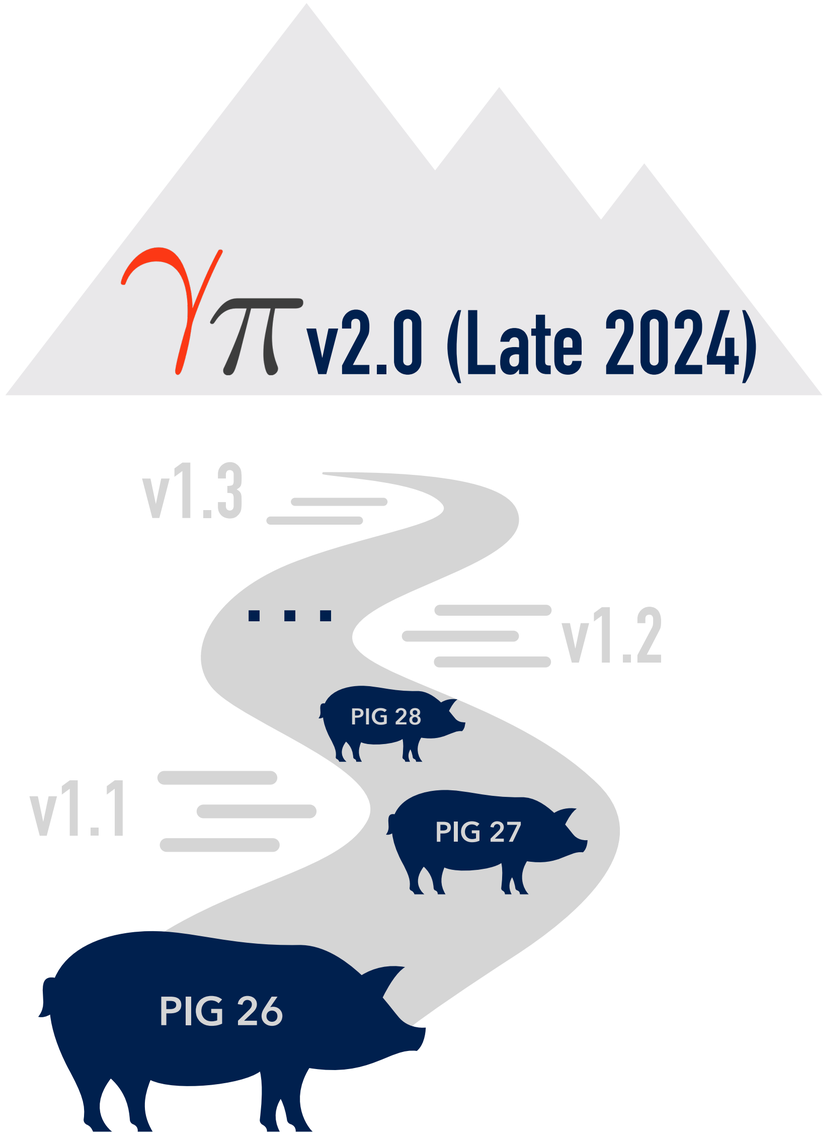
- **GitHub discussions**: https://github.com/gammapy/gammapy/discussions

- Gammapy **mailing list** (not used much…):
  - gammapy@googlegroups.com
  - Archive: https://groups.google.com/forum/#!forum/gammapy



Outlook & Summary

Future Developments

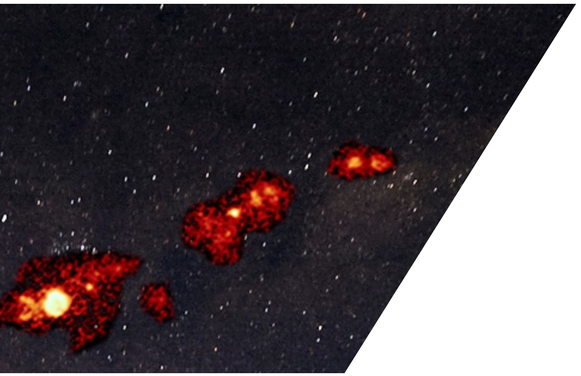


- Development of **LTS versions are coordinated via roadmap documents**, which are PIGs. They describe a common vision of the development of the software for a time range of ~2 years
- Following the roadmap there is a sequence of other PIGs proposing the actual development work in more detail
- We are **currently working on the roadmap for v2.0**
- Some topics to be addressed:
  - Fully configurable API (based on Pydantic)
  - Improved meta data handling (based on Pydantic)
  - Support for hierarchical model/ parameter priors
  - Solution for distributed computing and / or GPU support, `ray` / pytorch / jax
  - Split out `gammapy.maps` and possibly coordinate missing features with `ndcube` and `astropy.nddata` or `xarray`
- Gammapy is also being evaluated for the analysis of neutrino data from KM3Net. This allows interesting science for case, where the gamma-ray emission is coupled to neutrino emission

If you have experience with one of these topics, please talk to me!

Summary

- Gammapy is an **openly developed Python package for Gamma-ray astronomy in a mature state**, marked by the v1.0 release.
- Gammapy is ready to be used for gamma-ray data analysis. It has an estimated user base of **200 - 300 users and is expected to grow further**
- There is some **exciting science expected with CTA and Gammapy** in future
- However the scale of gamma-ray analyses will grow in terms of handling data, meta data and computational resources. There is some work ahead of us…



Thanks!

https://gammapy.org/contact.html