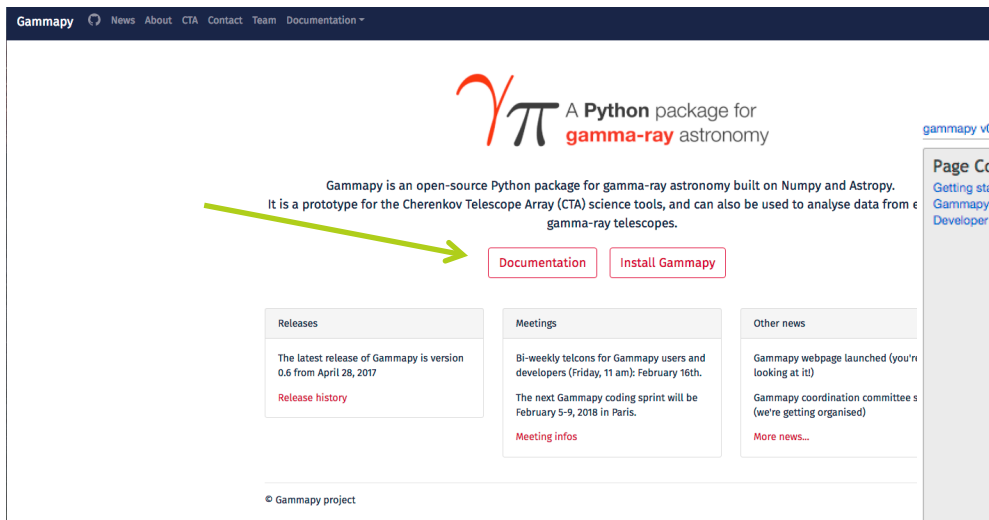


Documentation


Roberta

Where to find it?

1



Gammapy [News](#) [About](#) [CTA](#) [Contact](#) [Team](#) [Documentation](#)

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

Gammapy is an open-source Python package for gamma-ray astronomy built on Numpy and Astropy. It is a prototype for the Cherenkov Telescope Array (CTA) science tools, and can also be used to analyse data from gamma-ray telescopes.

[Documentation](#) [Install Gammapy](#)

Releases	Meetings	Other news
The latest release of Gammapy is version 0.6 from April 28, 2017. Release history	Bi-weekly telcons for Gammapy users and developers (Friday, 11 am): February 16th. The next Gammapy coding sprint will be February 5-9, 2018 in Paris. Meeting infos	Gammapy webpage launched (you're looking at it!) Gammapy coordination committee s (we're getting organised) More news...

© Gammapy project

gammapy v0.7.dev5423

Page Contents

[Getting started](#)
[Gammapy package](#)
[Developer documentation](#)



Gammapy is a community-developed, open-source Python package for gamma-ray astronomy. It is a prototype for the CTA science tools. This page (<http://docs.gammapy.org>) contains the Gammapy documentation. The Gammapy webpage (<http://gammapy.org>) contains information about Gammapy, including news and contact information if you have any questions, want to report an issue or request a feature, or need help with anything Gammapy-related.

Getting started

Gammapy works with Python 2 and 3, on Linux, Mac OS X and (partly) Windows. See [Installation](#) for information how to get started and the [Gammapy tutorial notebooks](#) to start to learn how to use Gammapy.

- [Installation](#)
- [Getting Started](#)
- [Gammapy tutorial notebooks](#)
- [Data Formats](#)
- [References](#)
- [Changelog](#)

Gammapy package

As mentioned in the [Getting Started](#), the Gammapy package is structured as a series of sub-packages. We recommend that you start to learn Gammapy via the [Gammapy tutorial notebooks](#), and then consult the following pages for further information about each sub-package. Those pages also contain very detailed reference documentation for every function and class in Gammapy.

- Astrophysical source and population models ([gammapy.astro](#))
- Background estimation and modeling ([gammapy.background](#))
- Source catalogs and objects ([gammapy.catalog](#))
- Cube Style Analysis ([gammapy.cube](#))
- Data and observation handling ([gammapy.data](#))
- Access datasets ([gammapy.datasets](#))
- Source detection tools ([gammapy.detect](#))
- Image processing and analysis tools ([gammapy.image](#))
- Instrument response function (IRF) functionality ([gammapy.irf](#))
- Spectrum estimation and modeling ([gammapy.spectrum](#))
- Statistics tools ([gammapy.stats](#))
- Time handling and analysis ([gammapy.time](#))
- Utility functions and classes ([gammapy.utils](#))
- Data Structures for Images and Cubes ([gammapy.maps](#))
- Command line tools ([gammapy.scripts](#))

Developer documentation

The Gammapy webpage contains information about the [Gammapy project and team](#) as well as information about Gammapy contact and communication channels. Most development takes place on the [Gammapy GitHub page](#).

- [Developer documentation](#)

■ www.gammapy.org

■ <http://docs.gammapy.org/dev/>

What is available?

2

■ Sub-package documentation:

- Introduction
- Getting started
- Examples
- API documentation

Data Structures for Images and Cubes ([gammapy.maps](#))

Warning

The code in [gammapy.maps](#) is currently in an experimental/development state. Method and class names may change in the future.

Introduction

[gammapy.maps](#) contains classes for representing pixelized data structures with at least two spatial dimensions representing coordinates on a sphere (e.g. an image in celestial coordinates). These classes support an arbitrary number of non-spatial dimensions and can represent images (2D), cubes (3D), or hypercubes (4+D). Two pixelization schemes are supported:

- WCS : Projection onto a 2D cartesian grid following the conventions of the World Coordinate System (WCS). Pixels are square in projected coordinates and as such are not equal area in spherical coordinates.
- HEALPix : Hierarchical Equal Area Iso Latitude pixelation of the sphere. Pixels are equal area but have irregular shapes.

[gammapy.maps](#) is organized around two data structures: *geometry* classes inheriting from [MapGeom](#) and *map* classes inheriting from [Map](#). A geometry defines the map boundaries, pixelization scheme, and provides methods for converting to/from map and pixel coordinates. A map owns a [MapGeom](#) instance as well as a data array containing map values. Where possible it is recommended to use the abstract [Map](#) interface for accessing or updating the contents of a map as this allows algorithms to be used interchangeably with different map representations. The following reviews methods of the abstract map interface. Documentation specific to WCS- and HEALPix-based maps is provided in [HEALPix-based Maps](#) and [WCS-based Maps](#).

Getting Started ¶

All map objects have an abstract interface provided through the methods of the [Map](#). These methods can be used for accessing and manipulating the contents of a map without reference to the underlying data representation (e.g. whether a map uses WCS or HEALPix pixelization). For applications which do depend on the specific representation one can also work directly with the classes derived from [Map](#). In the following we review some of the basic methods for working with map objects.

Constructing with Factory Methods

The [Map](#) class provides a `create` factory method to facilitate creating an empty map object from scratch. The `map_type` argument can be used to control the pixelization scheme (WCS or HPX) and whether the map internally uses a sparse representation of the data.

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

What is available?

2

■ Sub-package documentation:

- Introduction
- Getting started
- Examples
- API documentation

Data Structures for Images and Cubes (**gammapy.maps**)

Warning

The code in **gammapy.maps** is currently in an experimental/development state. Method and class names may change in the future.

Introduction

gammapy.maps contains classes for representing pixelized data structures with at least two spatial dimensions representing coordinates on a sphere (e.g. an image in celestial coordinates). These classes support an arbitrary number of non-spatial dimensions and can represent images (2D), cubes (3D), or hypercubes (4+D). Two pixelization schemes are supported:

Reference/API

gammapy.maps Package

Maps (2D and 3D).

This is work in progress, we're prototyping.

- Names and API might change.
- Not mentioned to users in the HTML docs at this point
- Contributions and feedback welcome!

Classes

HpxGeom (nside[, nest, coordsys, region, ...])	Geometry class for HEALPIX maps.
HpxMap (geom, data)	Base class for HEALPIX map classes.
HpxNDMap (geom[, data, dtype])	Representation of a N+2D map using HEALPix with two spatial dimensions and N non-spatial dimensions.
HpxSparseMap (geom[, data, dtype])	Representation of a N+2D map using HEALPIX with two spatial dimensions and N non-spatial dimensions.
Map (geom, data)	Abstract map class.
MapAxis (nodes[, interp, name, node_type, unit])	Class representing an axis of a map.
MapCoords (data[, coordsys])	Represents a sequence of n-dimensional map coordinates.
MapGeom	Base class for WCS and HEALPix geometries.
SparseArray (shape[, idx, data, dtype, ...])	Sparse N-dimensional array object.
WcsGeom (wcs, npix[, cdelt, axes, conv])	Geometry class for WCS maps.
WcsMap (geom, data)	Base class for WCS map classes.
WcsNDMap (geom[, data, dtype])	Representation of a N+2D map using WCS with two spatial dimensions and N non-spatial dimensions.

What is available?

2

■ Sub-package documentation:

- Introduction
- Getting started
- Examples
- API documentation

■ Developer documentation

Developer documentation

The developer documentation is a collection of notes for Gammapy developers and maintainers.

If something is not covered here, have a look at the very extensive Astropy developer documentation [here](#), we do most things the same way.

But you don't have to read all this stuff if you want to contribute something to Gammapy.

We're happy to help out with any Gammapy-related questions or issues!

Contact points

If you want to talk about Gammapy, what are the options?

- You can always post on the [Gammapy mailing list](#)! E.g. if you have an issue with Gammapy installation, analysis, a feature request or found a possible issue.
- You can also file an issue or pull request on the [Gammapy Github page](#). Making an account on Github takes only a minute and is free. Github is where Gammapy development takes place, i.e. anything that might / will lead to a change to the Gammapy code or documentation will eventually result in a pull request on Github.
- Often it's unclear if some issue (e.g. an error during installation) is the result of you executing the wrong command, or an actual issue in Gammapy that must be addressed by a documentation or code change. In those cases we suggest you post on the mailing list first.
- We have a [Slack](#) set up for Gammapy developer chat. You can use it via the Slack app or from your web browser at [Gammapy on Slack](#). If you need help with git, Github, Python or have questions about Github development (e.g. how to run tests, build the docs, ...), i.e. chat that isn't directly related to an issue or pull request on Github, you can use Slack.
- If you want to have a non-public conversation about Gammapy, or added to the Gammapy slack, please email [Christoph Deil](#).

Pages

- [How to contribute to Gammapy?](#)
- [Gammapy project setup](#)
- [Developer HOWTO](#)
- [How to make a Gammapy release](#)
- [PIGs](#)

What is available?

2

■ Sub-package documentation:

- Introduction
- Getting started
- Examples
- API documentation

■ Developer documentation

■ Notebook tutorials

- few datasets in gammapy-extra:
 - **HESS public release**
(see Christoph's talk) https://github.com/gammasky/HESS-DL3-DR1/tree/master/release_store
 - **CTA DC1** (see my previous talk)
<https://github.com/gammapy/gammapy-extra/tree/master/datasets/cta-1dc>
 - **3FHL Fermi LAT**
<https://github.com/gammapy/gammapy-fermi-lat-data/tree/master/3fhl>
- All to be run with binder?

Notebooks

For a quick introduction to Gammapy, go here:

- First steps with Gammapy | [first_steps.ipynb](#)

Interested to do a first analysis of simulated CTA data?

- CTA first data challenge (1DC) with Gammapy | [cta_1dc_introduction.ipynb](#)
- CTA data analysis with Gammapy | [cta_data_analysis.ipynb](#)

To learn how to work with gamma-ray data with Gammapy:

- IACT DL3 data with Gammapy (H.E.S.S. data example) | [data_iact.ipynb](#)
- Fermi-LAT data with Gammapy (Fermi-LAT data example) | [data_fermi_lat.ipynb](#)

2-dimensional sky image analysis:

- Image analysis with Gammapy (run pipeline) (H.E.S.S. data example) | [image_pipe.ipynb](#)
- Image analysis with Gammapy (individual steps) (H.E.S.S. data example) | [image_analysis.ipynb](#)
- Source detection with Gammapy (Fermi-LAT data example) | [detect_ts.ipynb](#)
- CTA 2D source fitting with Sherpa | [image_fitting_with_sherpa.ipynb](#)

1-dimensional spectral analysis:

- Spectral models in Gammapy | [spectrum_models.ipynb](#)
- Spectral analysis with Gammapy (run pipeline) (H.E.S.S. data example) | [spectrum_pipe.ipynb](#)
- Spectral analysis with Gammapy (individual steps) (H.E.S.S. data example) | [spectrum_analysis.ipynb](#)
- Spectrum simulation and fitting (CTA data example with AGN / EBL) | [cta_simulation.ipynb](#)
- Fitting gammapy spectra with sherpa | [spectrum_fitting_with_sherpa.ipynb](#)
- Flux point fitting with Gammapy | [sed_fitting_gammacat_fermi.ipynb](#)

3-dimensional cube analysis:

- Cube analysis with Gammapy (part 1) (compute cubes and mean PSF / EDISP) | [cube_analysis_part1.ipynb](#)
- Cube analysis with Gammapy (part 2) (likelihood fit) | [cube_analysis_part2.ipynb](#)

Time-related analysis:

- Light curve estimation with Gammapy | [light_curve.ipynb](#)

Extra topics

These notebooks contain examples on some more specialised functionality in Gammapy.

Most users will not need them. It doesn't make much sense that you read through all of them, but maybe browse the list and see if there's something that could be interesting for your work (or contribute to Gammapy if something is missing).

- Template background model production with Gammapy | [background_model.ipynb](#)
- Continuous wavelet transform on gamma-ray images | [cwt.ipynb](#)
- Interpolation using the NDDataArray class | [nddata_demo.ipynb](#)
- Rapid introduction on using numpy, scipy, matplotlib | [using_numpy.ipynb](#)

Reorganization

3

Global reorganization needed

- Many duplications
- Often not reader friendly
- 1.5 h this week to design it and appointing responsible people
- mid-term project
- Sub tasks
 - General design for the documentation
 - Installation button and instruction
 - How to organize the “getting started”?
 - Polishing/updating the sub-package docs
 - Polishing/updating the notebooks
 - Polishing/updating the developer docs
 - Notebooks in binder?