gamma-cat Documentation

Release

gamma-cat contributors

Nov 15, 2017

About

gamma-cat is an open data collection and source catalog for gamma-ray astronomy.

- Data and Docs (this site): https://gamma-cat.readthedocs.io
- Repository: https://github.com/gammapy/gamma-cat
- Explore *gamma-cat* and other gamma-ray data at gamma-sky.net
- Access and analyse *gamma-cat* and other gamma-ray data from Python: gammapy.catalog
- For now, see also here: https://gammapy.github.io/gamma-cat (will be moved to Sphinx docs soon)

This site is organised in the following sections:

- *About* - Quick introduction to *gamma-cat*
- data-docs - Get *gamma-cat* data
- *User Documentation* - Detailed info about *gamma-cat*
- *Contributor Documentation* - How to give feedback and contribute

CHAPTER 1

About

1.1 Brief

This project was started in August 2015 and first presented as a poster (see gamma-cat poster at ADASS 2016) at the ADASS 2016 conference.

There is a second gamma-cat poster from 2017 with more recent information on gamma-cat (and gamma-sky.net).

This repo contains a *gammacat* Python package to work with the data. It is BSD licensed (like Astropy, Gammapy). This license only applies to the code, it has nothing to do with the data.

This is work in progress. Feedback and contributions welcome!

Contact: use whatever you prefer:

- Email: gammapy@gmail.com
- Github issue tracker: https://github.com/gammapy/gamma-cat/issues (requires you to make a Github account, which is free and takes a minute)
- TODO: should we make a twitter or facebook account?

For now, we just collect TeV gamma-ray source information. Later we might try to ingest and interconnect other catalogs (e.g. Fermi-LAT GeV catalogs).

1.2 Why? There's already TeVCat!

Yes, there is TeVCat .

But TeVCat isn't really open.

You can view the info on their webpage, and copy & paste individual numbers, but you can't download a catalog and use it for your research.

To quote http://tevcat.uchicago.edu/terms.html (accessed August 26, 2016):

> Users may not perform systematic downloads of TeVCat data for any purpose,

> whether commercial or not, without the written permission of the
> TeVCat team. This includes scripted parsing of the website data,
> webpage 'scraping', and the use of robots. If you need access to bulk
> TeVCat data products, please contact the TeVCat team at tevcat@gmail.com

A second major problem with TeVCat is that there's no version history. Updates and corrections happen, but only the maintainers know when that happens and (presumably) what the older values were.

The goal here is to have a fully open TeV catalog that you can download and use as you like (well, we still require attribution, see next section).

The data here is maintained as text (YAML and ECSV) files in a git repo on Github following the model of astrocats, so it's fully transparent and we have version history.

The concrete motivation for Christoph Deil to start this catalog in August 2016 was to have a TeV catalog for gamma-sky.net, as well as for checks of all sources in the H.E.S.S. Galactic plane survey catalog against previous publications, and to have a TeV source catalog available for the upcoming CTA science challenge.

Open and reproducible research for gamma-ray astronomy!

1.3 Terms of use

All data collected here was originally generated by others. If you intend to use this data in a publication, we ask that you please cite the linked sources and/or contact the sources of the data directly.

Collecting, cleaning and maintaining this catalog has taken a lot of time. We require that you give attribution if you're using it.

For now, while we don't have a website / paper yet that can be cited, please use this attribution:

> Data taken from https://github.com/gammapy/gamma-cat,
> an open data collection and source catalog for gamma-ray astronomy.

Otherwise, you are free to use this data as you like.

Of course, we appreciate feedback can contributions (additions and corrections)!

1.4 Contributors

The following people have directly contributed to *gamma-cat* (alphabetical order by first name)

- Arjun Voruganti (@vorugantia)
- Axel Donath (@adonath)
- Christoph Deil (@cdeil)
- Gernot Maier (@GernotMaier)
- Matthias Wegen (@wegenmat)
- Peter Deiml (@pdeiml)

Many others have contributed indirectly, e.g. given data or feedback via private communication.

Thank you!

1.5 Acknowledgements

The following tools and services were used to produce this catalog:

- SIMBAD
- Astrophysics data system (ADS)
- Astropy
- Gammapy
- TGeVCat was used to look up info and we had good discussions and collaboration with the *TGeVCat* team.
- TeVCat was used to look up some info. (We did comply with their terms and conditions and didn't scrape their website to collect data for this catalog!)

1.6 Related resources

Here are some related resources:

- Some of the data formats we use are defined or being defined at the gamma-astro-data-formats project.
- gamma-sky.net is a portal to the gamma-ray sky
- Fermi high-level data products (e.g. catalogs): https://fermi.gsfc.nasa.gov/ssc/data/access/lat
- **H.E.S.S. source catalog: https://www.mpi-hd.mpg.de/hfm/HESS/pages/home/sources.**
  - TSV (tab-separated value format) version: https://www.mpi-hd.mpg.de/hfm/HESS/pages/home/sources/HESS_catalog.tsv
  - Copy on HEASARC with option to download in FITS or VOTable format: https://heasarc.gsfc.nasa.gov/W3Browse/all/hesscat.html
- HESS data collection for gamma-cat: https://github.com/gammapy/gamma-cat/blob/master/todo/todo_hess_aux.md
- Collection of MAGIC data: http://vobs.magic.pic.es/fits
- VERITAS data can be found by following the links here: http://veritas.sao.arizona.edu/veritas-science/veritas-results-mainmenu-72
- Another TeV source catalog is at TeVCat and with a new web interface at TeVCat2.
- Another TeV source catalog is the TeGeV Catalogue @ ASDC
- A light curve archive @ DESY
- A collection and search interface for HESS spectra and lightcurves of blazars: http://hess.obspm.fr
- A collection of Galactic TeV source info in FITS format by Andrea Giuliani and others from ASTRI is here: http://www.iasf-milano.inaf.it/~giuliani/astrisim/gsed

CHAPTER 2

Data

2.1 Overview

With gamma-cat, we provide two data products:

1. A source catalog
2. The full data collection

The following sections describe how to download them.

For now the latest version of the files is here: https://github.com/gammapy/gamma-cat/tree/master/docs/data

TODO: set up an easier way to download the files, directly from this page.

2.2 Source catalog

The source catalog is available as a single table in a single file. It contains only part of the data available in gamma-cat.

- `gammacat.fits.gz` – Main version of the source catalog (in FITS format)
- `gammacat.ecsv` – Partial source catalog (in ECSV format)
- `gammacat.yaml` – Partial source catalog (in YAML format)

Why multiple formats?

- FITS is the main format for the catalog we release. It supports vector columns, which we use for spectral points.
- The ECSV and YAML variant are more for us working on gamma-cat, to have a text-based, version control friendly format where it's easy to see which changes occurred from one version to the next.

2.3 Data collection

The full data collection is available as a collection of files in ECSV and JSON format.

- `gammacat-datasets.json` – Main index file, with links to all other data files
- `gammacat-sources.json` – TODO: remove?
- TODO: add "bundled" version of the files, with all information put in a single large JSON (or alternatively, equivalently HDF5) file.

CHAPTER 3

Use

This page contains some info how to use *gamma-cat*.

3.1 Browse

3.2 Download

TODO: Explain available data products

3.3 Analyse

TODO: Info on how to analyse catalog data from Python, i.e. work with spectra etc.

CHAPTER 4

Stats

4.1 Basics

- Version: {{ version }}

4.2 Source locations

TODO:

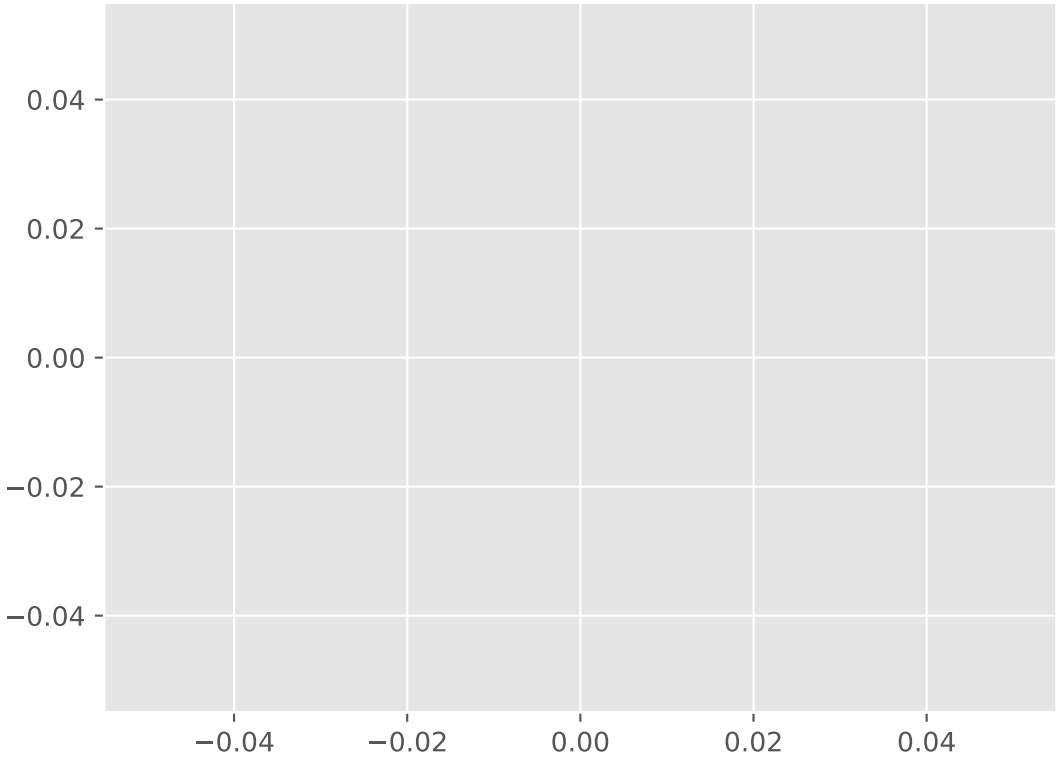
- AIT plot of sky locations
- Add Bokeh or Aladin Lite interactive Figure?

4.3 Kifune plot

The so-called Kifune plot shows the cumulative number of detected TeV sources versus time.

4.4 Source classes

Plot source class pie chart


0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04

CHAPTER 5

Changes

This is the changelog for gamma-cat.

It lists releases and major changes, not every small data update.

See also: *Contributors* and *Acknowledgements*.

5.1 Version 0.1 (unreleased)

- tbd

CHAPTER 6

Introduction

So you want to contribute to gamma-cat?

Really? Are you sure?

Great! Read on . . .

There are many ways to contribute to **gamma-cat**. We need more people that help with data entry, review data in gamma-cat for completeness and accuracy, improve the documentation, the data format schemas and the Python scripts.

We will try to describe and explain how everything works here in the contributor documentation. But we realise that in the end we will probably fall short, and if you try to contribute you will have questions about YAML or ECSV or schemas or get stuck with a git on Python question. If that happens, please don't give up, but contact us and we will help!

Everything happens on **Github** here: https://www.github.com/gammapy/gamma-cat

This is a git repository that contains everything related to gamma-cat: the data entry in the `input` folder, the script `make.py` and `gammacat` folder with the Python scripts to generate the output files, which at the moment are in the `docs/data` folder. The documentation at https://gamma-cat.readthedocs.io is generated from the RST files in that repo. But not just all the files and version control is there, Github is also the place to file "issues" for question, discussion, feature or data entry requests, bug reports (see https://github.com/gammapy/gamma-cat/issues). And it is also the place where all changes and additions happen, via "pull requests" (see https://github.com/gammapy/gamma-cat/pulls).

So if you want to contribute to gamma-cat, you have to make an account on Github. It's free and should just take a minute. You can then find a lot of information about git and Github here: https://help.github.com/

There you will find documentation how to open "issues" and "pull requests" and resources how to learn git, as well as how to do basic things like edit or add a file directly via the Github web interface. This means that you can do some data entry or documentation improvements for gamma-cat in a simple way. If you're new to Github and git, and the explanations below aren't clear to you, then what you can do is to open a new issue in the gamma-cat issue tracker where you describe what you want to do (i.e. add or change something), and then we'll try to help you do it, i.e. make your first pull request.

The following pages give you more information, focusing mostly on how to do data entry for gamma-cat, since this will be the most common way for people to contribute.

CHAPTER 7

Input

We've already mentioned it in the introduction: all data entry for gamma-cat happens by editing or adding text files in the `input` folder. We use file formats that are both human- and machine-readable:

- YAML files for hierarchical data
- ECSV files for tabular data

This section describes the format and content of the data entry files for gamma-cat.

All data entry is done in the folder named `input`. It contains three sub-folders of interest:

- `sources` contains yaml-files with basic information about the gamma-ray sources.
- `data` contains the data from publications stored in `YAML` and `ECSV` files. The folder contains subfolders named by years and there subsubfolders named by reference_ids. E.g. the data from the publication with reference `2015ApJ...802...65A` is stored in the folder `input/data/2015/2015ApJ...802...65A`. All these files are named corresponding to the source_id of the gamma-ray source defined in its definition file.
- `schemas` contains files which define the structure of the data entry files.

Now, these input files will be discussed in more detail, firstly the source definition files in *sources*:

The information (and a short description) which can be stored in such a file are defined by some keywords in basic_source_info.schema.yaml. It starts with properties, like the `common_name`, the `source_id` used in gamma-cat or the *tevcat_name* and goes on with information about experiments which investigated this source. Two important information are the `reference_ids`, which are all ADS reference to publication which deal with this source, and the `source_id` from which the names in the data folder are built. At the end of basic_source_info.schema.yaml after the keyword `required`, there are all of the upper information written down which have to be defined in a source definition file.

A good example to get familiar with this is e.g. tev_000049.yaml and compare it with basic_source_info.schema.yaml

The folders in /input/data/<year> contain ecsv files with measured fluxes in it, e.g. tev-000034-sed.ecsv. Additional information like `source_id` or `telescope` are stored as meta data in the file.

Moreover, there is always a info.yaml file, e.g. info.yaml, and the information which can be stored in is defined in dataset_info.schema.yaml.

Thirdly, there are YAML files named by the `source_id` in which the model parameters given in the publication are stored. Analogously, the information which can/ must be stored in such a yaml file are defined in dataset_source_info.schemas.yaml.

To get familiar with the data files, compare e.g. tev_000159.yaml with dataset_source_info.schemas.yaml.

When you add input data you have to do two things. Firstly, you must update the data status in the corresponding `info.yaml` file. Secondly, you have to tell gamma-cat about the new data. This is done in gammacat_database.yaml where you must add the `reference_id` of the publication of the added data. Gamma-cat will only contain data whose `reference_id` is listed in gammacat_database.yaml.

CHAPTER 8

Workflow

This page describes the git workflow you are recommended to use if you want to contribute to gamma-cat. In short words, we use the fork-branch-pullrequest way.

If you want to run the `gamma-cat` Python scripts locally, go through the Installation chapter.

But contribution can be done without the installation because we have continous integration tests set up on travis-ci that check that everything is working OK. If that is what you want to do, jump over the Installation chapter.

For information about the `gammacat` Python packages and the structure of the input files, please go to Code

8.1 Fork/ clone the repo

The first step is forking gamma-cat into your github account. Easily done by clicking on `fork` in the right top corner of gamma-cat

Create a folder `gamma-cat` (or as you like) on your local machine and in there execute:

```
git clone https://github.com/<your username>/gamma-cat.git
```

8.2 Installation (optional)

If you want to run the `gamma-cat` Python scripts locally, you need to install Python 3.6 and some Python packages. We recommend you to download Anaconda and then run the following commands in the `gamma-cat` folder:

```
conda config --set always_yes yes --set changeps1 no
conda update -q conda
conda info -a
# Now install our dependencies
conda env create -f environment.yml
# Activation of the installed environment
source activate gamma-cat
```

8.3 Make changes

This section describes the git workflow for changes and merging your changes into the gamma-cat master branch afterwards. For details about the code itself, please go to Code

If you want to make changes in the `gamma-cat` code, go to your local repository and make sure that your master-branch is up-to-date. Then create a branch from the current status of master by executing:

```
git checkout -b <Name of branch>
```

(Note: If there is a corresponding issue to the changes you do, the name of the branch should be something like *issue_xxx*. Otherwise, we highly recommend you to use resonable and easy names.)

When your work is done and you commited the changes locally(!) go back to the master branch and download all changes which might be done during your work (git pull upstream master). Then, go back to your working branch and execute:

```
git rebase master
```

Then upload the branch to your github account (git push) and open a Pull Request in the browser. A short description of the PR is always useful.

CHAPTER 9

Code

This page contains information about the whole code stored in the gamma-cat repository.

9.1 make.py

There is a command line interface to run the gamma-cat scripts, the `make.py` file in the top-level folder.

To see the available sub-commands and options:

```
$ ./make.py --help
```

To run the full pipeline, i.e. generate all output files and run all checks:

```
$ ./make.py all
```

After adding/ changing data in the input folder, one should always execute:

```
$ ./make.py checks
```

which checks the format/ structure of the input files.

9.2 gammacat package

The `make.py` command line interface just imports and executes functions and classes from the `gammacat` Python package (i.e. the `.py` files in the folder with name `gammacat`).

We list the modules in `gammacat` and comment on the code organisation.

The following are more basic modules:

- `utils.py` has some helper utility functions (e.g. for JSON / YAML / ECSV I/O)

- `modeling.py` has a `Parameter` and `ParameterList` class to help process input spatial and spectral source models from the YAML files.
- `info.py` has some helpers for versions, filenames, ...
- `sed.py` has a class to process and validate the spectral energy distributions (SEDs) in the input folder. The SEDs in the output folder can be read directly with `gammapy.spectrum.FluxPoints`.
- `lightcurve.py` has a class to process and validate the lightcurves in the input folder. The lightcurves in the output folder can be read directly with `gammapy.time.LightCurve`.

In additions there are classes in `gammapy.catalog.gammacat` that are used in the `gammacat` scripts to process the data: `GammaCatResource`, `GammaCatResourceIndex`, `GammaCatDatasetCollection`.

Then there is a hierarchy of higher-level modules (that import from the basic modules and modules representing lower-level steps in the processing pipeline):

- `input.py` has classes to read / clean up / process the data in the `input` folder.
- `collection.py` has classes to create the files in the `output` folder (only the dataset files and index files, not the catalog files).
- `cat.py` is the code to create the catalog files
- `checks.py` is the code to run checks. At the moment the methods there just dispatch to methods called `validate` or `check` in lower-level modules (such as `gammacat.input`), and the actual checks are thus scattered throughout the `gammacat` modules. There's also checks on data content in `gammacat/tests` (which is probably a bad idea, but pytest is convenient to have asserts)

9.3 Tests

There is a folder `gammacat/tests` with some unit tests for the code in the `gammacat` package, that can be executed via `python -m pytest gammacat/tests`

We don't have the relation quite figure out, what goes where:

- `gammacat/tests`
- `gammapy/catalog/tests/test_gammacat.py`
- The various check / validate methods throughout `gammacat` and executed via `./make.py check`.

9.4 Tools

The following tool is helpful to lint YAML files:

- yamllint

TODO: it's too picky, showing errors for things that are OK. Figure out how to make it less picky and document that here.

9.5 Website build

The *gamma-cat* website is a static website generated by Python and Sphinx.

We have a Sphinx test page where we can try out things locally and check if they also work on ReadTheDocs. It's an orphan page, i.e. doesn't show up for normal users.

We use several Sphinx extensions, and also have our own in *gammacat/sphinx/exts*.

More info soon . . . for now this is just a link collection:

- **sphinxcontrib.rawfiles**
  - Useful example. Not very useful directly, because can't control destination of the copy!?
  - Code: https://bitbucket.org/birkenfeld/sphinx-contrib/src/master/rawfiles/sphinxcontrib/rawfiles.py
  - PyPI: https://pypi.python.org/pypi/sphinxcontrib-rawfiles/
  - Although for now, this is working for what I need: http://www.sphinx-doc.org/en/1.5.1/config.html#confval-html_extra_path
- Looks very useful! http://sphinxcontribdatatemplates.readthedocs.io/en/latest/
- A good Sphinx table extension would be useful: https://github.com/sphinx-doc/sphinx/issues/786
- Not sure if this is useful! https://pythonhosted.org/sphinxcontrib-restbuilder/
- Maybe: https://pypi.python.org/pypi/sphinxcontrib-jsoncall
- Looks interesting, probably not useful here: https://jsdoc-toolkit-rst-template.readthedocs.io/en/latest/

CHAPTER 10

Details

This page contains some notes with details about *gamma-cat*.

10.1 Data

10.1.1 Reference identifiers

For paper identifiers, we use the ADS identifiers.

These are unique, well-known and stable. This corresponds to the *bibcode* in https://github.com/andycasey/ads/, i.e. can be used to obtain further info from ADS

Note that some bibcodes contain characters that don't work well (or at all) as directory or filenames, e.g. *2011A&A...531L..18H* with the *&* character. What ADS does in that case for URLs is to quote them, i.e. the URL is http://adsabs.harvard.edu/abs/2011A%26A...531L..18H . In *gamma-cat* we do the same, we URL quote the bibcode for filenames, folder names and URLs, and otherwise use the normal bibcode.

```
>>> papers = list(ads.SearchQuery(bibcode='2011A&A...531L..18H'))
>>> print(papers[0].bibcode)
'2011A&A...531L..18H'
```

```
>>> import urllib.parse
>>> urllib.parse.quote('2011A&A...531L..18H')
'2011A%26A...531L..18H'
>>> urllib.parse.unquote('2011A%26A...531L..18H')
'2011A&A...531L..18H'
```

10.1.2 Source identifiers

- We use integer source identifiers. In many cases they are the same as for the *TeGeV* catalog, but generally that is not the case.

- We do not introduce new sources "names" (for now). TBD: how should people reference sources from our catalog? Maybe we should do a position-based identifier like *TeVCat* or *TeGeV* cat?
- TBD: How do we handle sources that split out into multiple sources with deeper observations?

10.1.3 Source classes

TODO: document properly.

For now, see the list of source classes we're using at the end of this schema file:

https://github.com/gammapy/gamma-cat/blob/master/input/schemas/basic_source_info.schema.yaml

10.1.4 Positions

Sometimes source positions aren't measured or given in the paper. This is commonly the case for AGN. In those cases, we use the position from SIMBAD and look it up like this:

```
>>> from astropy.coordinates import SkyCoord
>>> SkyCoord.from_name('Crab nebula').to_string(precision=7)
'83.6330830 22.0145000'
```

and store it like this:

```
position:
  simbad_id: Crab nebula
  ra: 83.6330830
  dec: 22.0145000
```

The presence of the *simbad_id* key means that it's a position from SIMBAD.