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

Introduction to the June 2024 coding sprint

MPI-K Heidelberg

Current status: recent releases

- Bug fix release v1.0.2 is out since Dec 6th 2023

- Feature release v1.2 released on Feb 29th 2024:
  - initial plan was to release early Dec 2023
  - issues with pydantic and ray forced to postpone
  - main new features:
    - priors
    - metadata containers

Bug fixes branches status

- Future releases:
  - v1.0.3 : 16 PRs merged
    - several compatibility fixes.
    - few bug corrections, e.g. #5101 #5162
    - backports are becoming more and more difficult
      - several fixes have not been back ported to v1.0.x
    - likely no support for numpy 2.0
  - v1.2.1 : 15 merged PRs
    - a few important bug fixes, e.g. acceptance stacking
    - aim for a release date close to numpy v2.0 release?

Future milestones

- v 1.3:
  - already 61 PRs merged, 18 open.
  - 78 open issues, 18 closed.
    - 30 feature requests/18 bugs etc.
    - check the more urgent ones!
  - Target release in October 2024

- v 2.0 :
  - Aim is spring 2025. In line with first CTAO SAT release.
  - What are required features?
  - We need to plan development of missing ones
    - Aim for this week: build prototypes for major ones

Objectives of the week

- Preparation of v1.3
  - Define priorities for next release
    - check remaining bugs
    - what remaining new features are mandatory?

- Explore prototypes of missing features for v2.0

- Check issues with a coding sprint label on GitHub
  - Quentin has prepared a project with a number of selected issues.

Roadmap for v2.0: where do we stand?

- **gammapy.maps:**
  - RegionGeom should support sizes changing with axis.
  - Make maps fully re-usable for IRFs.
  - Allow ``Maps`` and ``MapCoord`` without spatial axes
  - Change to design without Geom, Introduce `WcsMapAxis`, `RegionMapAxis`, `HpxMapAxis` instead
  - Migrate from healpy to astropy-healpix or cds-healpix-python

- **Little progress so far. Postpone?**

Roadmap for v2.0: where do we stand?

- **Data model and formats**
  - define the internal data model and introduce a validation mechanism on input. - TODO
  - build a clear IO boundary between internal and external data representations that supports various versions of various formats. - TODO
  - define a metadata structure - done
    - support in data reduction workflow need to be studied and implemented

- **Build/evaluate prototypes of I/O structure for gammapy.data**
- **Prototype for CTAO observation model**

Roadmap for v2.0: where do we stand?

- **Documentation**
  - Introduce a deprecation system - done
  - Update pydata-sphinx-theme - done
  - More detailed and nicer - TODO?
  - Use type hints in Gammapy everywhere, no type hints for now - TODO?

- **Infrastructure**
  - Improve test coverage and quality
  - Improve tools helping releases - TODO
  - Creation of Docker images with an automatized tool. TODO
  - Update listing formatting CI to ruff/pre-commit.ci - in progress

Roadmap for v2.0: where do we stand?

- **Flexible statistics API**
  - Support for priors in likelihood - Partly done
    - finish correlated priors PR
  - Split of statistics definition from datasets - TODO
  - Support for statistical test associated with periodic signals, in the frequency domain
  - Add more tests on model hypothesis? - TODO
  - Likelihood weights?

- **Build prototype of fit statistic class split from dataset**

Roadmap for v2.0: where do we stand?

- **Modeling API**
    - Evaluate joint development with astromodels or astropy models - postpone
    - Rely more on the `SkyModel` then the submodel - TODO
        - e.g. move amplitude parameter to `SkyModel`
    - What about NPredModel, introduce consistently as concept? - TODO
    - Models to support systematic uncertainties - TODO
    - Handling the FitResult object. Serialisable? Rely on it in later API. - in progress

- **Build prototype of NPredModel framework.**
- **Replace MapEvaluator.**

Roadmap for v2.0: where do we stand?

- **Performance**
  - Support ray for distributed computing - Done
  - Make Dataset distributable with same API - Done
  - Probably rework Dataset API, split off model handling…
  - Evaluate Jax for GPU acceleration and autograd - TODO

- **Build JAX/Pytorch prototypes.**