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

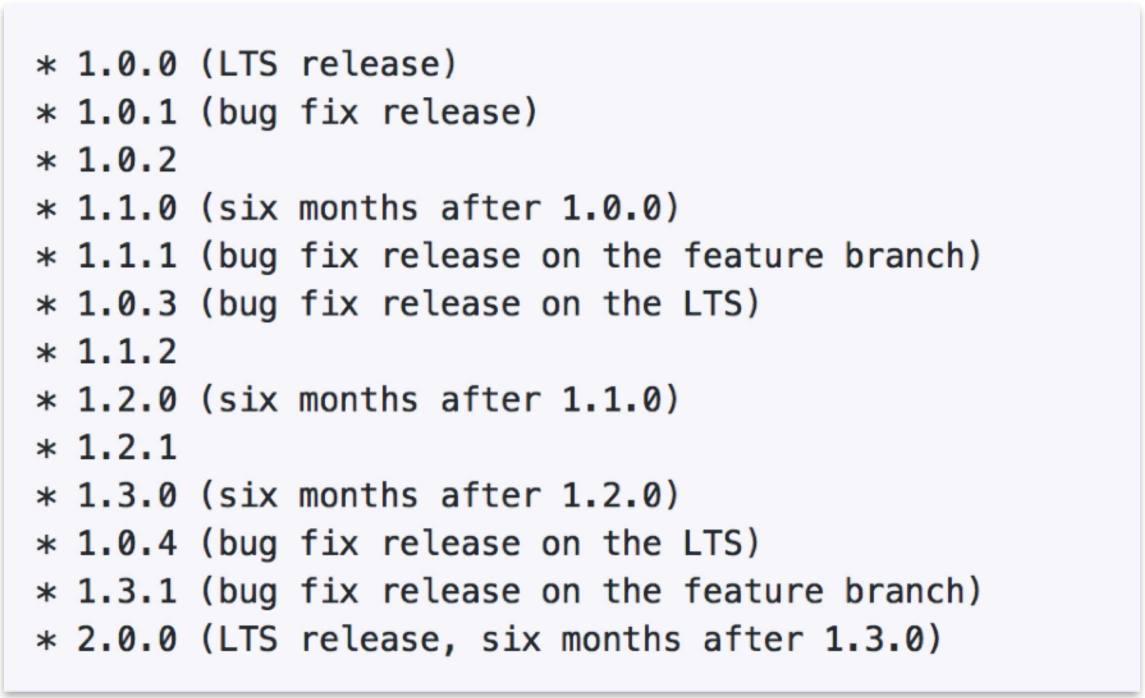
Contributing to gammapy: what's new?

R. Terrier

Dec 2022 Coding sprint , APC Paris

New release scheme

- Long Term Stable (LTS) release expected every ~2 yrs
- Feature (or minor) releases (e.g. v1.x) every <~ 6 months
- Bug fix releases for the last feature release and last LTS



```
* 1.0.0 (LTS release)
* 1.0.1 (bug fix release)
* 1.0.2
* 1.1.0 (six months after 1.0.0)
* 1.1.1 (bug fix release on the feature branch)
* 1.0.3 (bug fix release on the LTS)
* 1.1.2
* 1.2.0 (six months after 1.1.0)
* 1.2.1
* 1.3.0 (six months after 1.2.0)
* 1.0.4 (bug fix release on the LTS)
* 1.3.1 (bug fix release on the feature branch)
* 2.0.0 (LTS release, six months after 1.3.0)
```

See Release and version numbering PIG

Authorship

- Authorship policy defined by PIG 24 for
  - code releases (e.g. on Zenodo and Software Heritage)
  - gammapy paper(s) and conferences

- Author list for v1.0 software and paper based on invitation and "opt-in"

- After v1.0: author list:
  - includes all people that contributed since the last LTS release
  - builds up with minor releases, "reset" after next LTS version

- Contributions to be listed as co-author :
  - commit to code or documentation, maintenance & devops tasks on main or associated repos, etc

Contributing main changes

- See developer documentation
- Code style and Continuous Integration
- DCO and commits sign-offs
- Bug fixes on the LTS: backporting changes to v1.0.x
- How to handle API changes and deprecation?

Code style checks on CI

- To ensure code quality and limit clean up work for releases, PEP8 code style is enforced on CI (Continuous Integration)

- To enforce this, we advise to use the pre-commit hook.
- Install it with: `pre-commit install`

```
$ git commit -s -m 'Some file'
isort...........................................................Passed
black...........................................................Passed
flake8............................................Passed
[test a69469324] Some file
 1 file changed, 3 insertions(+)
```

Commits sign-off and DCO

- To clarify user contributions, PIG 24 proposes to ask contributors to explicitly accept the Developer Certification of Origin (copied from Linux foundation)

- To explicitly agree, one has to explicitly sign-off commits

`Signed-off-by: Random Developer <random@example.org>`

- ensure `user.name` and `user.email` are correctly configured in your git.local

```
$ git config –global user.email "YOUR_EMAIL"
```

- Add the `–s` option to your `git commit`

```
$ git commit -s -m ‘a commit message’
```

Commits sign-off and DCO

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I have the right to submit it under the open source license indicated in the file; or

(b) The contribution is based upon previous work that, to the best of my knowledge, is covered under an appropriate open source license and I have the right under that license to submit that work with modifications, whether created in whole or in part by me, under the same open source license (unless I am permitted to submit under a different license), as indicated in the file; or

(c) The contribution was provided directly to me by some other person who certified (a), (b) or (c) and I have not modified it.

(d) I understand and agree that this project and the contribution are public and that a record of the contribution (including all personal information I submit with it, including my sign-off) is maintained indefinitely and may be redistributed consistent with this project or the open source license(s) involved.

Commits sign-off and DCO

- We will introduce a CI check for signed-off commits.

- A pre-commit hook can be configured as well to ensure all commits are properly signed-off
  - Installing it explicitly would be considered an acceptation of DCO for future commits

See : https://github.com/gammapy/gammapy/pull/4205

- Note that you can amend a commit with no sign-off

```
$git commit --amend --no-edit --signoff
```

See [this page]

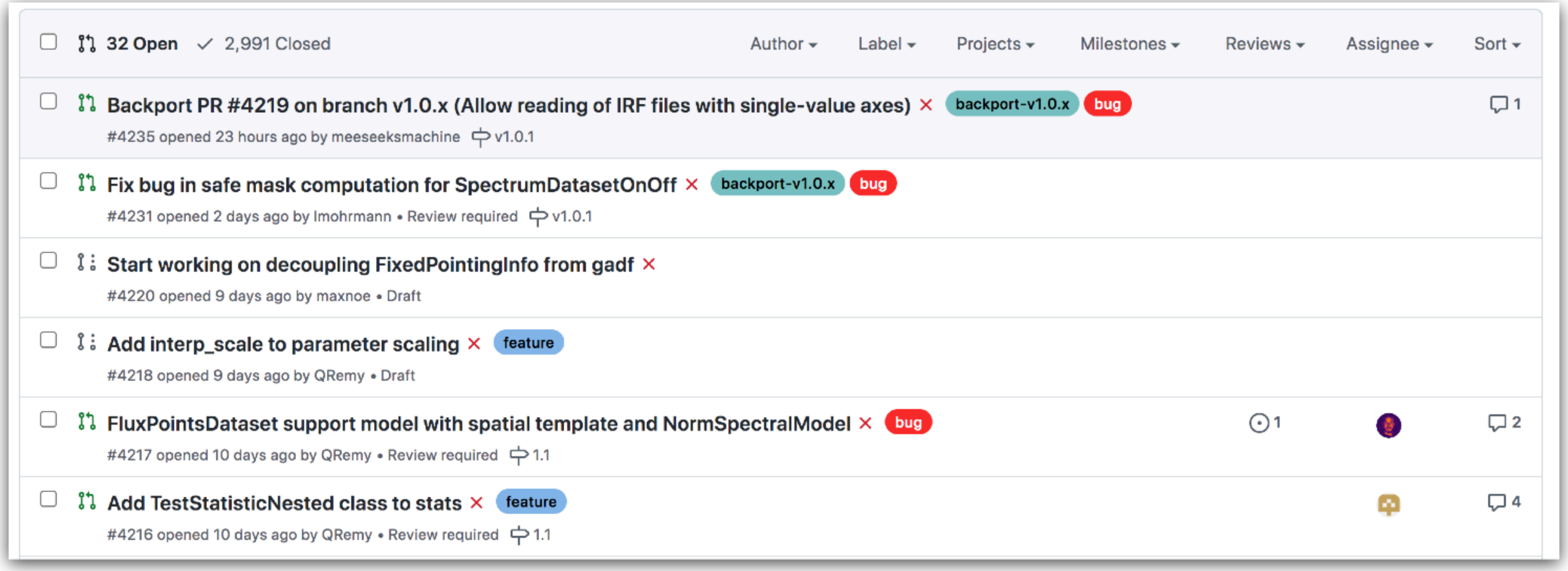
Bug fixes on LTS

When correcting a bug on the LTS release:

- Usually needs to be corrected on *main* branch too
  - Create branch from *main,* and make commits
  - Open PR and
    - apply **backport-v1.0.x** label
    - select **v1.0.1** (or later) milestone
  - On merge:
    - commits will be *automatically* cherry-picked
    - a PR will be opened on *v1.0.x* branch

- If bug applies to LTS only, branch from *v1.0.x*
  - same milestone but not backporting

Bug fixes on LTS



32 Open ✓ 2,991 Closed

Author ▾ Label ▾ Projects ▾ Milestones ▾ Reviews ▾ Assignee ▾ Sort ▾

Backport PR #4219 on branch v1.0.x (Allow reading of IRF files with single-value axes) ✕ backport-v1.0.x bug 1
#4235 opened 23 hours ago by meeseeksmachine v1.0.1

Fix bug in safe mask computation for SpectrumDatasetOnOff ✕ backport-v1.0.x bug
#4231 opened 2 days ago by lmohrmann • Review required v1.0.1

Start working on decoupling FixedPointingInfo from gadf ✕
#4220 opened 9 days ago by maxnoe • Draft

Add interp_scale to parameter scaling ✕ feature
#4218 opened 9 days ago by QRemy • Draft

FluxPointsDataset support model with spatial template and NormSpectralModel ✕ bug 1 2
#4217 opened 10 days ago by QRemy • Review required 1.1

Add TestStatisticNested class to stats ✕ feature 4
#4216 opened 10 days ago by QRemy • Review required 1.1

API breaking changes

- API breaking changes can be introduced in feature and LTS releases.
- These changes must be clearly identified in the release notes.
- API changes and deprecation must be announced in the previous release:
  - in the form of a deprecation warning to warn users of future changes affecting their code.
  - in the code docstring

- **Approach : rely on astropy decorators**

API breaking changes

- Deprecating a function or a class

```
from gammapy.utils.deprecation import deprecated

@deprecated("1.1", alternative="new_function")
def deprecated_function(a, b):
    return a + b
```

```
>>> print(deprecated_function.__doc__)

.. deprecated:: 1.1
    The deprecated_function function is deprecated
and may be removed in a future version.
        Use new_function instead.
```

API breaking changes

- Deprecating a function or a class

```
from gammapy.utils.deprecation import deprecated

@deprecated("1.1", alternative="new_function")
def deprecated_function(a, b):
    return a + b
```

```
>>> deprecated_function(1,2)
GammapyDeprecationWarning: The deprecated_function
function is deprecated and may be removed in a future
version.
        Use new_function instead.
  deprecated_function(1,2)
3
```

API breaking changes

- Renaming an argument in a function/method

```
from gammapy.utils.deprecation import
deprecated_renamed_argument

@deprecated_renamed_argument("a", "x", "1.1")
def deprecated_argument_function(x):
    return x + 2
```

```
>>> print(deprecated_argument_function(a=1, b=2))
GammapyDeprecationWarning: "a" was deprecated in version
1.1 and will be removed in a future version. Use argument
"x" instead.
  print(deprecated_argument_function(a=1))
3
```

API breaking changes

- Renaming a kwarg in a function/method

```
from gammapy.utils.deprecation import
deprecated_renamed_argument

@deprecated_renamed_argument("old", « new", "1.1",
arg_in_kwargs=True)
def deprecated_argument_function_kwarg(new=1):
    return new
```

```
>>>print(deprecated_argument_function_kwarg(old=3))
GammapyDeprecationWarning: "old" was deprecated in version
1.1 and will be removed in a future version. Use argument
"new" instead.
  print(deprecated_argument_function_kwarg(old=3))
3
```