maps
&
maps based analysis

gammapy coding sprint,
Oct 2018,
UCM, Madrid

Image and cube analysis with `maps`

- GitHub project: maps

  - https://github.com/gammapy/gammapy/projects/2

Image and cube analysis using `gammapy.maps`

- All dependencies on `SkyImage` and `SkyCube` have been removed:

  - 1D spectral analysis and 2D ring background classes have been updated

- Image and cube analysis is performed by `MapMaker` and `MapMakerObs`

  - Configurable to produce any map type in `counts, exposure, background`

  - Users need to provide a `WcsGeom`, a FoV angle cut and optionally a mask in the form of a `WcsNDMap`.

  - Background is integrated over energy bins but not normalized

Image and cube analysis using `gammapy.maps`

```python
energy_axis = MapAxis.from_edges(
    np.logspace(-1., 1., 10), unit="TeV", name="energy", interp="log"
)
geom = WcsGeom.create(
    skydir=(0, 0),
    binsz=0.02,
    width=(10, 8),
    coordsys="GAL",
    proj="CAR",
    axes=[energy_axis],
)
```

```python
%%time
maker = MapMaker(geom, offset_max=4. * u.deg)
maps = maker.run(obs_list)
```

```python
# mean PSF
src_pos = SkyCoord(0, 0, unit="deg", frame="galactic")
table_psf = obs_list.make_mean_psf(src_pos)

# PSF kernel used for the model convolution
psf_kernel = PSFKernel.from_table_psf(table_psf, geom, max_radius="0.3 deg")
```

```python
# define energy grid
energy = energy_axis.edges * energy_axis.unit

# mean edisp
edisp = obs_list.make_mean_edisp(
    position=src_pos, e_true=energy, e_reco=energy
)
```

`gammapy.maps`

- **Outstanding issues**: *astropy.units handling*

  - #1792: MapAxis/MapCoord do not deal properly with units

    - If you don't use the right unit, you will get wrong results but no failure!

  - Do we need some specialized `MapAxis`?

    - e.g. do we need some `EnergyMapAxis`?

      - Could replace `gammapy.utils.energy`

      - Could expose nature of axis i.e. true or reco energy

- Longer term issues:

  - #1805: Extend map concept to other type of data

Image and cube analysis using `gammapy.maps`

- **Outstanding issues**: *background integration*

  - #1830: Background integration is incorrect for too large energy bins (e.g. 2D analysis).

    - `irf.background.integrate_on_energy_range` is performing log trapezoidal integration

    - the oversampling factor needs to be adapted to the specific energy binning. Options:

      - expose parameter to the user.

      - impose minimal number of evaluation per decade

      - perform exact integration. `Background3D` performs only linear or nearest neighbor interpolation.

    - the latter requires more work but is likely more accurate and faster

Image and cube analysis using `gammapy.maps`

- **Outstanding issues**: *background normalization*

  - So far background normalization is only based on IRFs

  - This won't work in most real cases. Need to add background normalization schemes used in classical Cherenkov analysis:

    - `RingBackgroundEstimator` and `AdaptiveRingBackgroundEstimator` objects exist

    - Need to add a `FoVBackgroundEstimator`

  - How to pass this to `MapMaker`

    - pass normalization object directly

    - Create new distinct map maker class?

Image and cube analysis using `gammapy.maps`

- **Outstanding issues**: *handling true and reco energies*

  - #1583 : `e_true` and `e_reco` geometries are assumed to be identical in `MapMaker`

    - Not adapted to 2D analysis and likely error prone in general

    - Users should provide the `e_reco MapGeom`

    - true energy bins could be provided separately

      - e.g. `Emin, Emax, nbins`

      - Note there are typical default values per instrument/configuration

    - exposure map & PSF kernels in reco energy should be computed through `apply_edisp` function

Image and cube analysis using `gammapy.maps`

- **Longer term issues**: *IRF reprojection efficiency*

  - IRF reprojection is the most time-consuming task for maps creation

    - Computing reprojected IRFs on fine binning (e.g. 0.01°) is useless and inefficient when IRF are stored on ~0.5° theta grids.

    - Better interpolate and mosaic on large bins and interpolate final image on required binning

    - Define minimal spatial binning for IRF maps and interpolate by `Map.interp_by_coord` as final step.

-

Image and cube analysis using `gammapy.maps`

- Longer term issues: *IRF reprojection*

  - Finish `PSFMap` class

    - Include `PSFMap` summation function

  - Prepare proper energy dispersion treatment:

    - Create `EdispMap` class

    - How to deal with reco energy mask?

- Longer term issues: *Improving parameter exposure on MapMaker*

  - How many parameters should we expose in MapMaker? How much complex can it really get?

  - Config-file driven version?