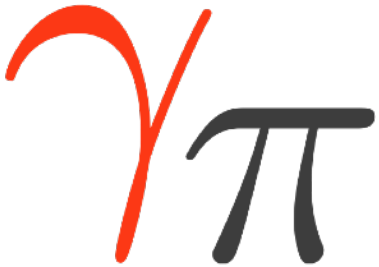

A **Python** package for
gamma-ray astronomy

PIG 22

A general *Estimator* result object

gammapy co-working week
November 23, 2020

Flux formats: gadf SED types

- Definition of SED types in gamma-astro-data-formats:
  - https://gamma-astro-data-formats.readthedocs.io/en/latest/spectra/flux_points/index.html#sed-types

- `dnde` differential flux which is defined at a given `e_ref`
- `e2dnde` differential energy flux which is defined at a given `e_ref`
- `flux` integral flux defined between `e_min` and `e_max`
- `eflux` integral energy flux defined between `e_min` and `e_max`
- `likelihood` type introduced by `fermipy`:
  - `norm` parameter and reference fluxes, e.g. `ref_dnde`, `ref_flux`, `ref_npred`
  - see : https://gamma-astro-data-formats.readthedocs.io/en/latest/spectra/binned_likelihoods/index.html#likelihood-sed

Current situation in gammapy

- Most `Estimators` return flux estimates:
  - `FluxPointsEstimator` returns a `FluxPoints` object, i.e. an astropy `Table` with a `SED-type` following the likelihood convention
  - `LightCurveEstimator` returns a `Table` containing arrays of `flux` SED type quantities ordered in time
  - `TSMapEstimator` and `ExcessMapEstimator` return dictionaries of `Map` objects following the `flux` SED type
  - `ExcessProfileEstimator` returns a Table of `flux` SED type quantities order along some spatial direction

Some issues

- Difficult to serialize easily flux map output
- Inhomogeneity of SED types produced
- Difficult to convert flux quantities in different SED type
  - only `FluxPoints` has a `.to_sed_type(type)` method
  - but all conversions are not implemented
- Only one source in FluxPoints, but potentially we could extract several at once.
- Export information from initial datasets (e.g. GTI, meta)
- Potentially lots of code duplication if we are to have dedicated, complex objects for each categories

A proposal: `FluxEstimate`

- Create a `FluxEstimate` base class for most of `Estimators` results.
- Internally rely on `Likelihood` SED type
- Contains a table or dictionary and a reference spectral model.
- Would allow all flux type conversions:
  - `FluxEstimate.dnde`
  - `FluxEstimate.flux_ul`
- Would require adapting `FluxPoints`

A proposal: `FluxMap` API

```
model = SkyModel(PointSpatialModel(), PowerLawSpectralModel(index=2.5))

estimator = TSMapEstimator(model, energy_edges=[0.2, 1.0, 10.0]*u.TeV)

flux_maps = estimator.run(dataset)

# plot differential flux map in each energy band
flux_maps.dnde.plot_grid()

# plot energy flux map in each energy band
flux_maps.eflux.plot_grid()

# one can access other quantities
flux_maps.sqrt_ts.plot_grid()
flux_maps.excess.plot_grid()

# Extract flux points at selected position
position = SkyCoord(225.31, 35.65, unit="deg", frame="icrs")
fp = flux_maps.get_flux_points(position)
fp.plot()

# Save to disk as a fits file
flux_maps.save("my_flux_maps.fits", overwrite=True)

# Read from disk
new_flux = FluxMap.read("my_flux_maps.fits")
```

A proposal: `FluxPointsCollection`

- Most products are just list (ordered?) of FluxPoints:
  - per time interval
  - per region
  - per source
- A general class to handle them all
- Relying internally on astropy Table
- Specialized classes inheriting from it?
  - enable I/O with externally produced data
  - dedicated plotting methods
  - extract information (fluxes) within specific region/time/source