Building cube analysis on maps:

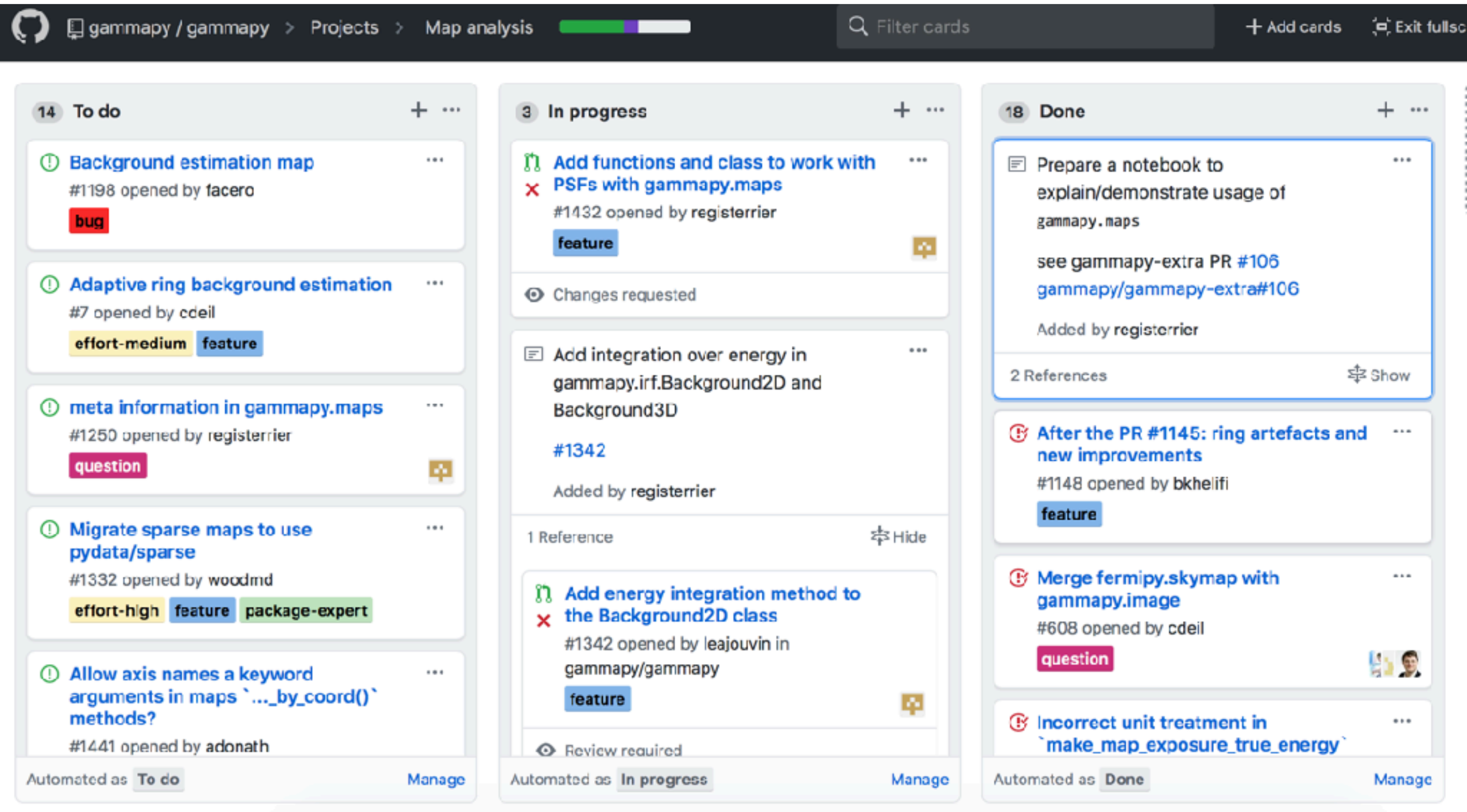
status & objectives

`gammapy` coding sprint,
July 2018,
MPI-K Heidelberg

Cube analysis with `maps`

- Objectives presented and discussed in PIG 2:
  - see PR #1277
    - presented at last coding sprint (slides)
- GitHub project: maps
  - https://github.com/gammapy/gammapy/projects/2

Cube analysis with `maps`



Todos at the last coding sprint

- Add metadata (`OrderedDict`) and units to maps (esp. serialization)
- Improve Cutout/Slices approach
- Modify background IRF (`background3D`).
  Define `.integrate` method. Same for `EffectiveArea2D?`
- Implement/test bkg normalization scheme
- Work on exposure maps
- Develop PSFKernel, EDISPKernel classes for convolution

What's new since the last coding sprint

- Added basic functions to fill `Map` from `EventList` and build Maps of reprojected IRFs (exposure, bkg) for single observations
  - FoV selection: max offset only
  - Relies only on `WcsNDMap` for the moment
  - Analysis on list of observations performed with `MapMaker`
    - No `energy reco` - `energy_true`

```python
class MapMaker(object):
    """Make all basic maps for a single observation.

    Parameters
    ----------
    ref_geom : `~gammapy.maps.WcsGeom`
        Reference image geometry
    offset_max : `~astropy.coordinates.Angle`
        Maximum offset angle
    """

    def __init__(self, ref_geom, offset_max):
        self.offset_max = offset_max
        self.ref_geom = ref_geom

        # We instantiate the end products of the Make
        self.count_map = WcsNDMap(self.ref_geom)

        data = np.zeros_like(self.count_map.data)
        self.exposure_map = WcsNDMap(self.ref_geom, d

        data = np.zeros_like(self.count_map.data)
        self.background_map = WcsNDMap(self.ref_geom,

        # We will need this general exclusion mask fo
        self.exclusion_map = WcsNDMap(self.ref_geom)
        self.exclusion_map.data += 1
```

```python
def process_obs(self, obs):                                                    [d
    """Process one observation.

    Parameters
    ----------
    obs : `~gammapy.data.DataStoreObservation`
        Observation
    """
    # First make cutout of the global image
    try:
        exclusion_mask_cutout, cutout_slices = make_cutout(
            self.exclusion_map, obs.pointing_radec,
            [2 * self.offset_max, 2 * self.offset_max],
        )
    except PartialOverlapError:
        # TODO: can we silently do the right thing here? Discuss
        print("Observation {} not fully contained in target image. Skipping it.".forma
        return

    cutout_geom = exclusion_mask_cutout.geom

    count_obs_map = make_map_counts(
        obs.events, cutout_geom, obs.pointing_radec, self.offset_max,
    )

    expo_obs_map = make_map_exposure_true_energy(
        obs.pointing_radec, obs.observation_live_time_duration,
        obs.aeff, cutout_geom, self.offset_max,
    )

    acceptance_obs_map = make_map_hadron_acceptance(
        obs.pointing_radec, obs.observation_live_time_duration,
        obs.bkg, cutout_geom, self.offset_max,
    )

    background_obs_map = make_map_fov_background(
        acceptance_obs_map, count_obs_map, exclusion_mask_cutout,
    )

    self._add_cutouts(cutout_slices, count_obs_map, expo_obs_map, background_obs_map)
```

Todos at the last coding sprint

- Add metadata (`OrderedDict` serialization)
- Improve Cutout/Slices approac
- Modify background IRF (`back` Define `.integrate` method.
- Implement/test bkg normalization scheme
- Work on exposure maps
- Develop PSFKernel, EDISPKernel classes for convolution

meta added as keyword in Map header

unit added as a string
- added Map.quantity property
- Map.unit property returns astropy Unit
- PR #1374

added type to MapAxis:
- based on Unit
- `energy`, `time`, `any`

MapGeom.get_coord() returns MapCoord
- See PR #1395

Todos at the last coding sprint

- Add metadata (`OrderedDict`) and units to maps (esp. serialization)
- Improve Cutout/Slices approach
- Modify background IRF (`backgrou`
  Define `.integrate` method. Same
- Implement/test bkg normalization scheme
- Work on exposure maps
- Develop PSFKernel, EDISPKernel classes for convolution

Added get_image_by methods:
- PR #1438

Added Map.slice_by_idx()
- PR #1443

Added WcsNDMap.cutout
- PR #1446

Todos at the last coding sprint

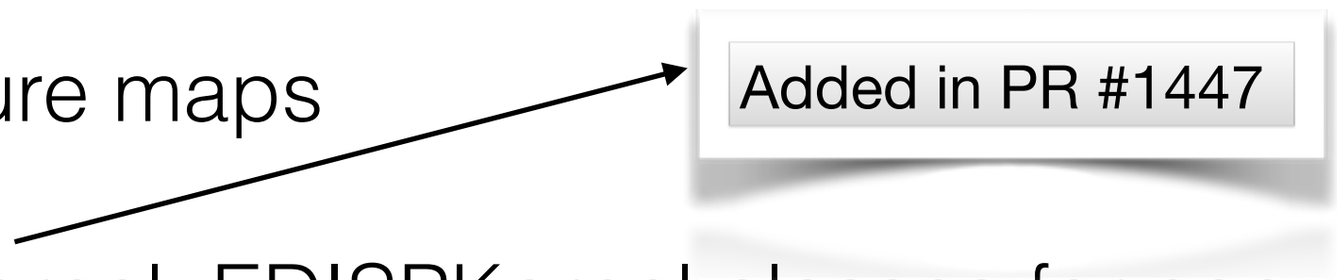
- Add metadata (`OrderedDict`) and units to maps (esp. serialization)
- Improve Cutout/Slices approach
- Modify background IRF (`backgrou`
  Define `.integrate` method. Same for `EffectiveArea2D?`
- Implement/test bkg normalization scheme
- Work on exposure maps
- Develop PSFKernel, EDISPKernel classes for convolution

Work started but stalled.
see e.g. PR # 1342

First implementation in
`make_map_fov_background`
Not adapted as is.

Todos at the last coding sprint

- Add metadata (`OrderedDict`) and units to maps (esp. serialization)
- Improve Cutout/Slices approach
- Modify background IRF (`background3D`).
  Define `.integrate` method. Same for `EffectiveArea2D?`
- Implement/test bkg normalization scheme
- Work on exposure maps
- Develop PSFKernel, EDISPKernel classes for convolution



What to work on this week and later?

- Finish background integration methods on IRF classes
- Finish **`PSFMap`** class
  - Implement **`PSFMap.contaminent_radius_map()`**
  - Include **`PSFMap`** summation function
- Prepare proper energy dispersion treatment:
  - Create **`EdispKernel`** class for convolution
  - Adapt **`MapMaker`** to handle **`geom_ereco`** and **`geom_etrue`**
  - Create **`EdispMap`** class

What to work on this week and later?

- Work on modular (config driven?) class to perform cube style analysis on list of observations to replace `MapMaker`

- Prepare cube-style analysis of CTA 1DC data

- Remove dependencies on SkyImage, SkyImageList and SkyCube:

- In 1D spectral analysis: see issue #1391 and PR#1421

- In 2D image analysis:

  - `RingBackgroundEstimator` et al.

  - `IACTBasicImageEstimator`

What to work on this week and later?

- Implement function to compute reprojected IRFs on a `MapCoord`:
  - allows to reduce computing to valid coords in a `Map`
  - allows to work transparently on any type of `MapGeom` (e.g. multiresolution maps)
  - use `MapCoord.skycoord.transform_to(frame)` to change to FoV coordinates
  - Need to modify behavior of `NDDataArray.evaluate`
    - i.e. no `itertools.products`
  - implement `fill_map_exposure` etc