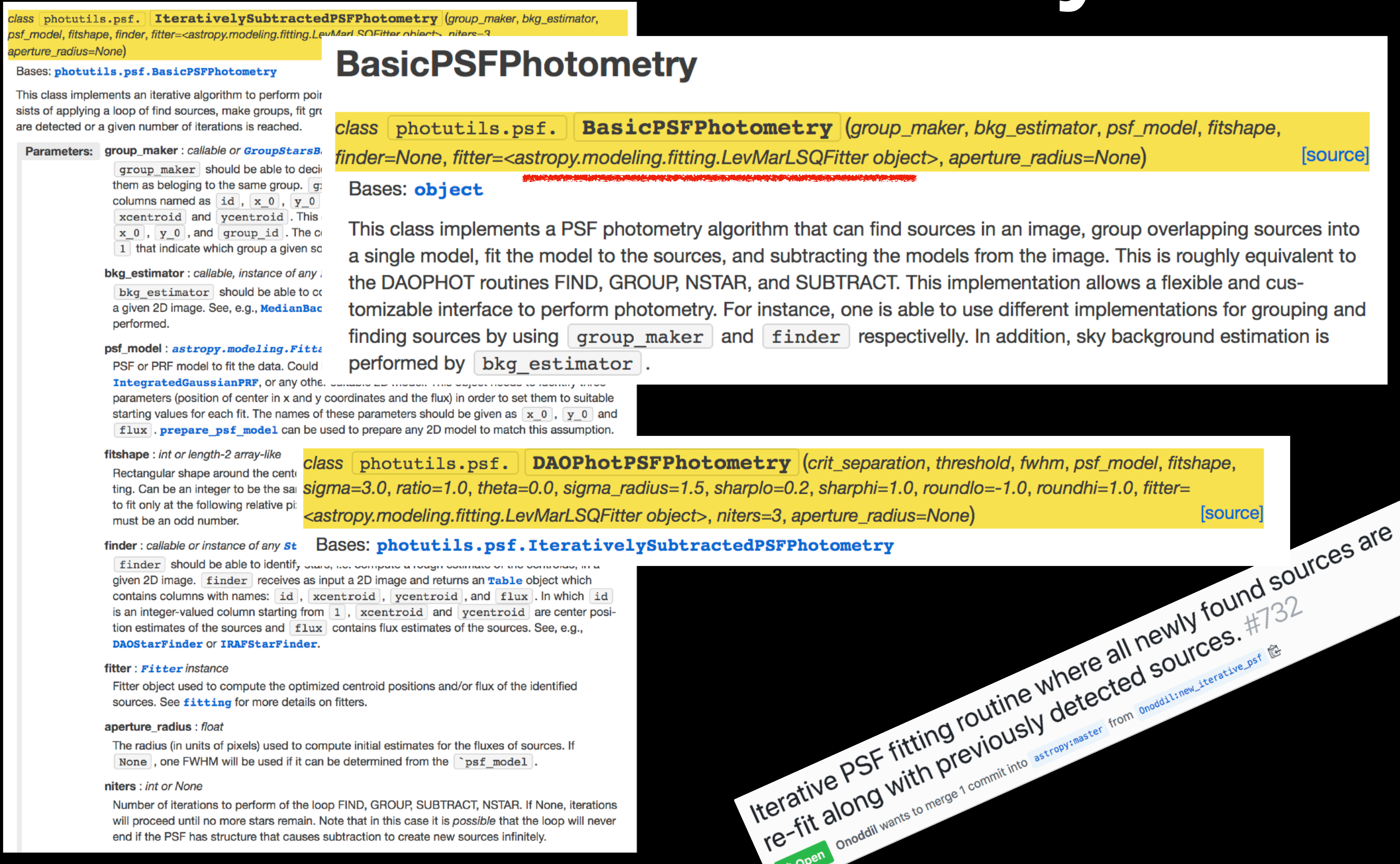
What has *photutils.psf* ever done for us?

Tom J. Wilson, STScI, Python in Astronomy, 7/31/19

PSF Photometry



class `photutils.psf.` `IterativelySubtractedPSFPhotometry`(*group_maker, bkg_estimator, psf_model, fitshape, finder, fitter=<astropy.modeling.fitting.LevMarLSQFitter object>, niters=3, aperture_radius=None*)

Bases: `photutils.psf.BasicPSFPhotometry`

This class implements an iterative algorithm to perform poi
sists of applying a loop of find sources, make groups, fit gr
are detected or a given number of iterations is reached.

Parameters: **group_maker** : *callable or* `GroupStarsB`

`group_maker` should be able to deci
them as beloging to the same group. `g`
columns named as `id`, `x_0`, `y_0`
`xcentroid` and `ycentroid`. This
`x_0`, `y_0`, and `group_id`. The c
`1` that indicate which group a given so

bkg_estimator : *callable, instance of any*

`bkg_estimator` should be able to c
a given 2D image. See, e.g., `MedianBac`
performed.

psf_model : `astropy.modeling.Fitta`

PSF or PRF model to fit the data. Could
`IntegratedGaussianPRF`, or any other suitable 2D model. This object needs to identify three parameters (position of center in x and y coordinates and the flux) in order to set them to suitable starting values for each fit. The names of these parameters should be given as `x_0`, `y_0` and `flux`. `prepare_psf_model` can be used to prepare any 2D model to match this assumption.

fitshape : *int or length-2 array-like*

Rectangular shape around the cent
ting. Can be an integer to be the sa
to fit only at the following relative pi
must be an odd number.

finder : *callable or instance of any* `St`

`finder` should be able to identify stars, i.e. compute a rough estimate of the centroids, in a given 2D image. `finder` receives as input a 2D image and returns an `Table` object which contains columns with names: `id`, `xcentroid`, `ycentroid`, and `flux`. In which `id` is an integer-valued column starting from `1`, `xcentroid` and `ycentroid` are center position estimates of the sources and `flux` contains flux estimates of the sources. See, e.g., `DAOStarFinder` or `IRAFStarFinder`.

fitter : `Fitter` *instance*

Fitter object used to compute the optimized centroid positions and/or flux of the identified sources. See `fitting` for more details on fitters.

aperture_radius : *float*

The radius (in units of pixels) used to compute initial estimates for the fluxes of sources. If `None`, one FWHM will be used if it can be determined from the ``psf_model`.

niters : *int or None*

Number of iterations to perform of the loop FIND, GROUP, SUBTRACT, NSTAR. If None, iterations will proceed until no more stars remain. Note that in this case it is *possible* that the loop will never end if the PSF has structure that causes subtraction to create new sources infinitely.

BasicPSFPhotometry

class `photutils.psf.` `BasicPSFPhotometry`(*group_maker, bkg_estimator, psf_model, fitshape, finder=None, fitter=<astropy.modeling.fitting.LevMarLSQFitter object>, aperture_radius=None*) [source]

Bases: `object`

This class implements a PSF photometry algorithm that can find sources in an image, group overlapping sources into a single model, fit the model to the sources, and subtracting the models from the image. This is roughly equivalent to the DAOPHOT routines FIND, GROUP, NSTAR, and SUBTRACT. This implementation allows a flexible and customizable interface to perform photometry. For instance, one is able to use different implementations for grouping and finding sources by using `group_maker` and `finder` respectivelly. In addition, sky background estimation is performed by `bkg_estimator`.

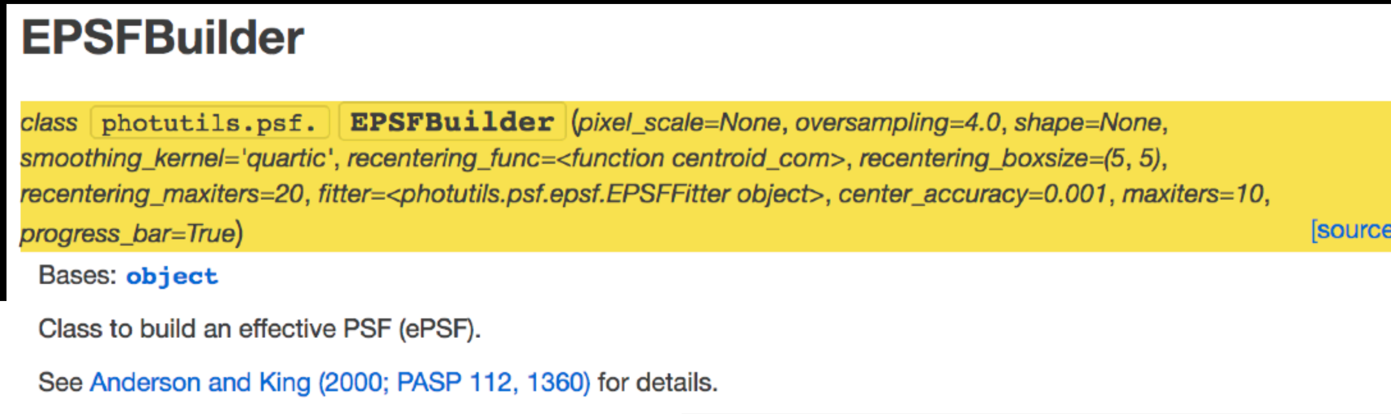
class `photutils.psf.` `DAOPhotPSFPhotometry`(*crit_separation, threshold, fwhm, psf_model, fitshape, sigma=3.0, ratio=1.0, theta=0.0, sigma_radius=1.5, sharplo=0.2, sharphi=1.0, roundlo=-1.0, roundhi=1.0, fitter=<astropy.modeling.fitting.LevMarLSQFitter object>, niters=3, aperture_radius=None*) [source]

Bases: `photutils.psf.IterativelySubtractedPSFPhotometry`

Iterative PSF fitting routine where all newly found sources are re-fit along with previously detected sources. #732

Open Onoddil wants to merge 1 commit into `astropy:master` from `Onoddil:new_iterative_psf`

“Effective” PSF Framework



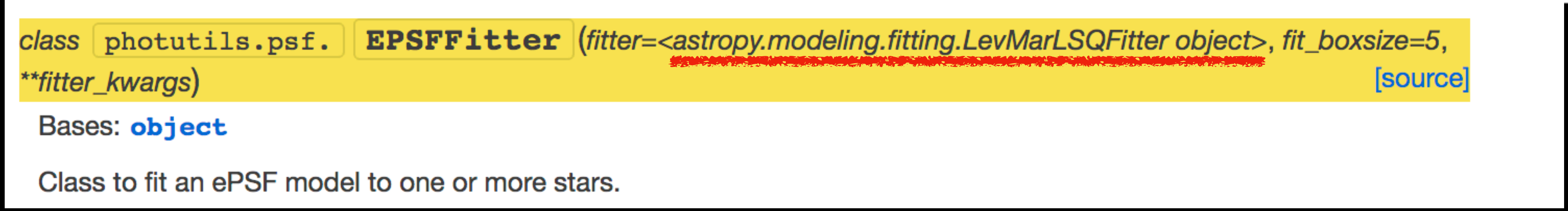
EPSFBuilder

class `photutils.psf.` **`EPSFBuilder`**(*pixel_scale=None*, *oversampling=4.0*, *shape=None*, *smoothing_kernel='quartic'*, *recentering_func=<function centroid_com>*, *recentering_boxsize=(5, 5)*, *recentering_maxiters=20*, *fitter=<photutils.psf.epsf.EPSFFitter object>*, *center_accuracy=0.001*, *maxiters=10*, *progress_bar=True*) [source

Bases: `object`

Class to build an effective PSF (ePSF).

See Anderson and King (2000; PASP 112, 1360) for details.

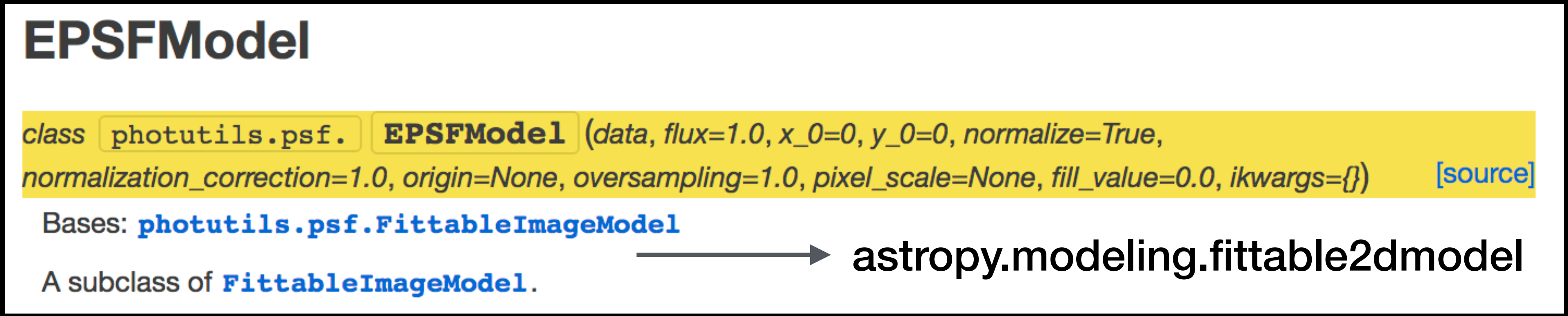


EPSFFitter

class `photutils.psf.` **`EPSFFitter`**(*fitter=<astropy.modeling.fitting.LevMarLSQFitter object>*, *fit_boxsize=5*, ***fitter_kwargs*) [source]

Bases: `object`

Class to fit an ePSF model to one or more stars.



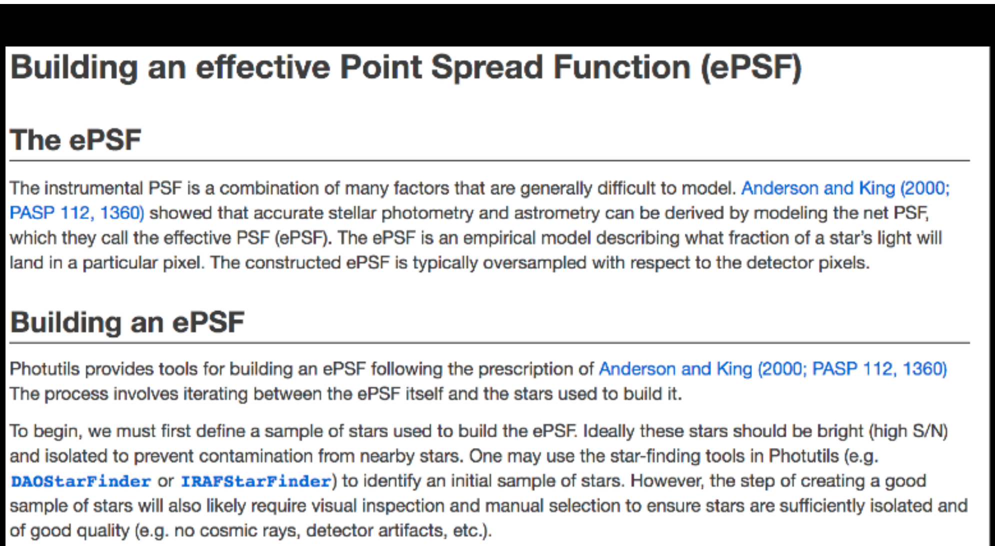
EPSFModel

class `photutils.psf.` **`EPSFModel`**(*data*, *flux=1.0*, *x_0=0*, *y_0=0*, *normalize=True*, *normalization_correction=1.0*, *origin=None*, *oversampling=1.0*, *pixel_scale=None*, *fill_value=0.0*, *ikwargs={}*) [source]

Bases: `photutils.psf.FittableImageModel`

A subclass of `FittableImageModel`.

→ astropy.modeling.fittable2dmodel



Building an effective Point Spread Function (ePSF)

The ePSF

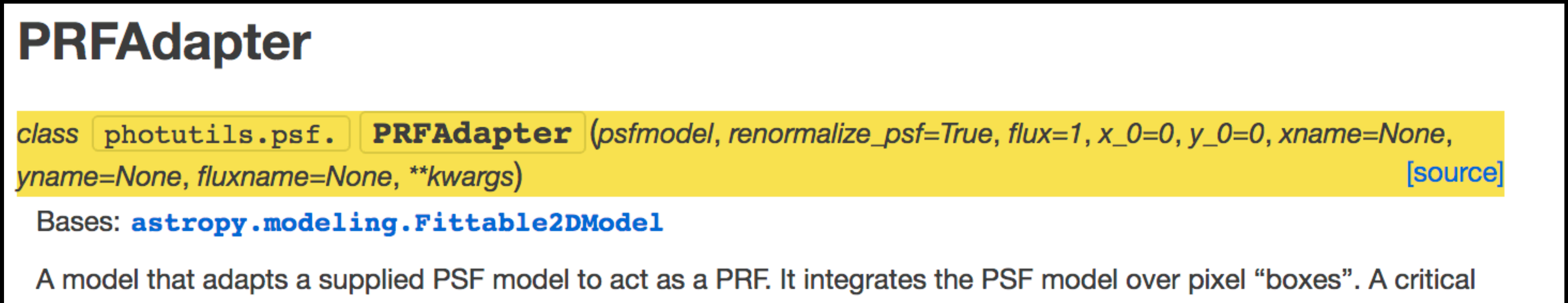
The instrumental PSF is a combination of many factors that are generally difficult to model. Anderson and King (2000; PASP 112, 1360) showed that accurate stellar photometry and astrometry can be derived by modeling the net PSF, which they call the effective PSF (ePSF). The ePSF is an empirical model describing what fraction of a star's light will land in a particular pixel. The constructed ePSF is typically oversampled with respect to the detector pixels.

Building an ePSF

Photutils provides tools for building an ePSF following the prescription of Anderson and King (2000; PASP 112, 1360). The process involves iterating between the ePSF itself and the stars used to build it.

To begin, we must first define a sample of stars used to build the ePSF. Ideally these stars should be bright (high S/N) and isolated to prevent contamination from nearby stars. One may use the star-finding tools in Photutils (e.g. `DAOStarFinder` or `IRAFStarFinder`) to identify an initial sample of stars. However, the step of creating a good sample of stars will also likely require visual inspection and manual selection to ensure stars are sufficiently isolated and of good quality (e.g. no cosmic rays, detector artifacts, etc.).

Other PSFs Framework

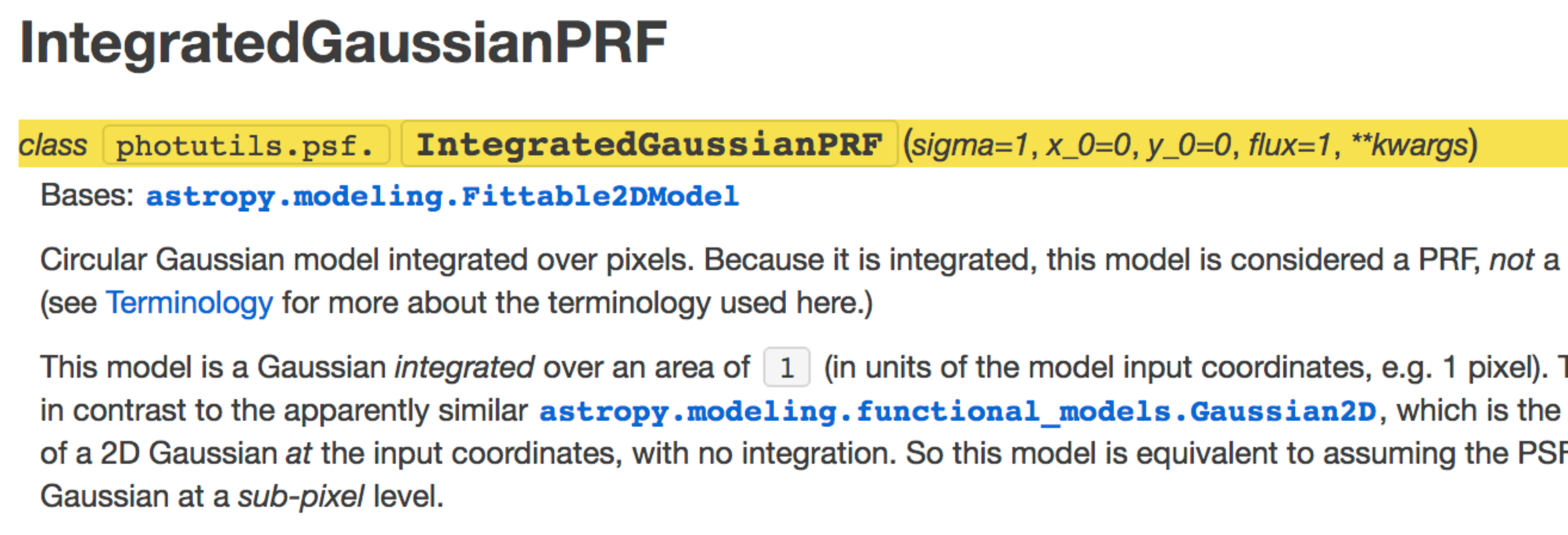


PRFAdapter

class `photutils.psf.` **`PRFAdapter`**(*psfmodel*, *renormalize_psf=True*, *flux=1*, *x_0=0*, *y_0=0*, *xname=None*, *yname=None*, *fluxname=None*, ***kwargs*) [source]

Bases: `astropy.modeling.Fittable2DModel`

A model that adapts a supplied PSF model to act as a PRF. It integrates the PSF model over pixel "boxes". A critical built-in assumption is that the PSF model scale and locat



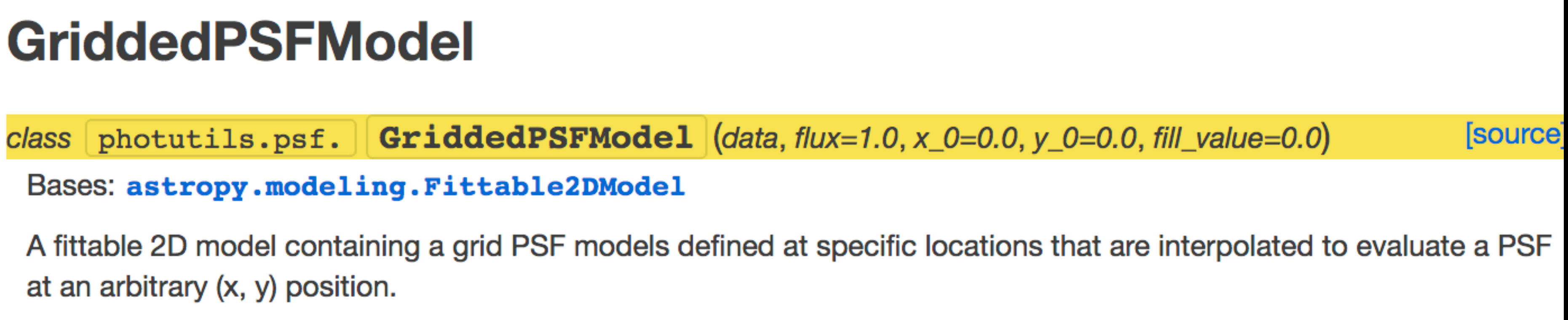
IntegratedGaussianPRF

class `photutils.psf.` **`IntegratedGaussianPRF`**(*sigma=1*, *x_0=0*, *y_0=0*, *flux=1*, ***kwargs*)

Bases: `astropy.modeling.Fittable2DModel`

Circular Gaussian model integrated over pixels. Because it is integrated, this model is considered a PRF, *not* a (see Terminology for more about the terminology used here.)

This model is a Gaussian *integrated* over an area of `1` (in units of the model input coordinates, e.g. 1 pixel). T in contrast to the apparently similar `astropy.modeling.functional_models.Gaussian2D`, which is the of a 2D Gaussian *at* the input coordinates, with no integration. So this model is equivalent to assuming the PSF Gaussian at a *sub-pixel* level.



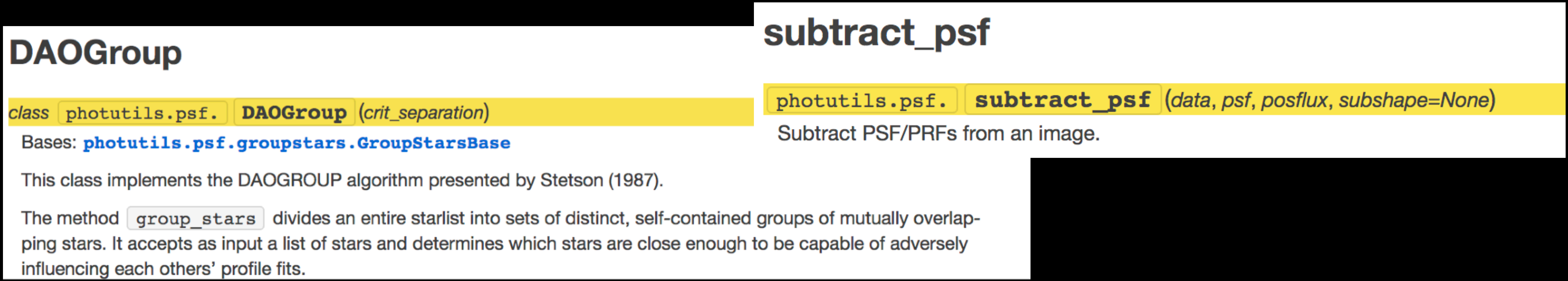
GriddedPSFModel

class `photutils.psf.` **`GriddedPSFModel`**(*data*, *flux=1.0*, *x_0=0.0*, *y_0=0.0*, *fill_value=0.0*) [source]

Bases: `astropy.modeling.Fittable2DModel`

A fittable 2D model containing a grid PSF models defined at specific locations that are interpolated to evaluate a PSF at an arbitrary (x, y) position.

Other Things



DAOGroup

class `photutils.psf.` **`DAOGroup`**(*crit_separation*)

Bases: `photutils.psf.groupstars.GroupStarsBase`

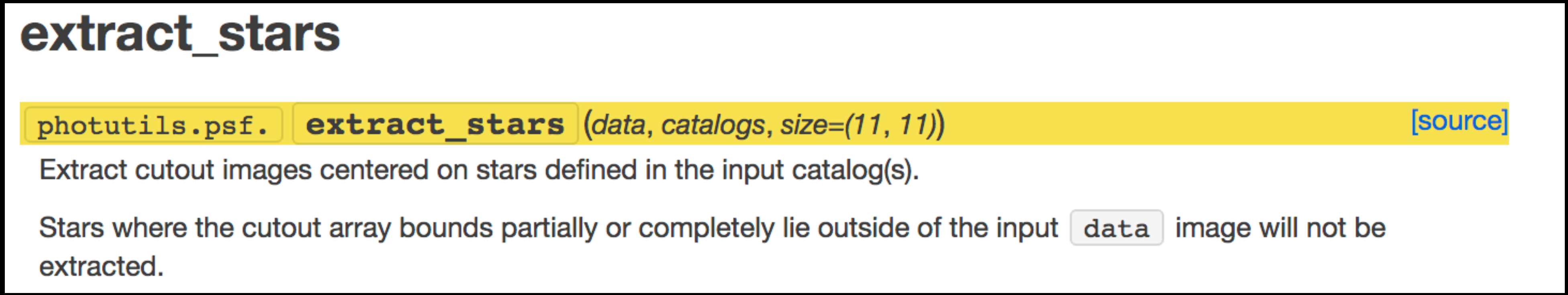
This class implements the DAOGROUP algorithm presented by Stetson (1987).

The method `group_stars` divides an entire starlist into sets of distinct, self-contained groups of mutually overlapping stars. It accepts as input a list of stars and determines which stars are close enough to be capable of adversely influencing each others' profile fits.

subtract_psf

`photutils.psf.` **`subtract_psf`**(*data*, *psf*, *posflux*, *subshape=None*)

Subtract PSF/PRFs from an image.

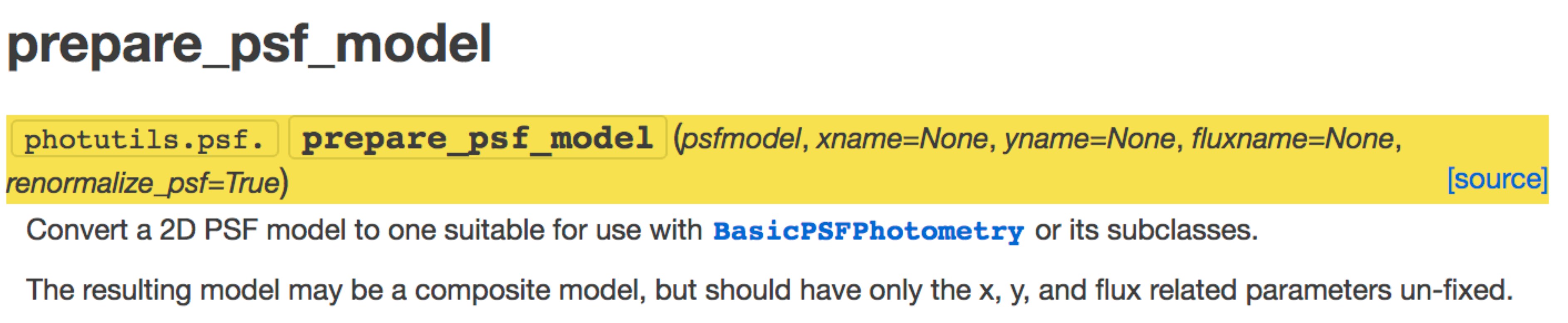


extract_stars

`photutils.psf.` **`extract_stars`**(*data*, *catalogs*, *size=(11, 11)*) [source]

Extract cutout images centered on stars defined in the input catalog(s).

Stars where the cutout array bounds partially or completely lie outside of the input `data` image will not be extracted.



prepare_psf_model

`photutils.psf.` **`prepare_psf_model`**(*psfmodel*, *xname=None*, *yname=None*, *fluxname=None*, *renormalize_psf=True*) [source]

Convert a 2D PSF model to one suitable for use with `BasicPSFPhotometry` or its subclasses.

The resulting model may be a composite model, but should have only the x, y, and flux related parameters un-fixed.

What will *photutils.psf* ever do for us?

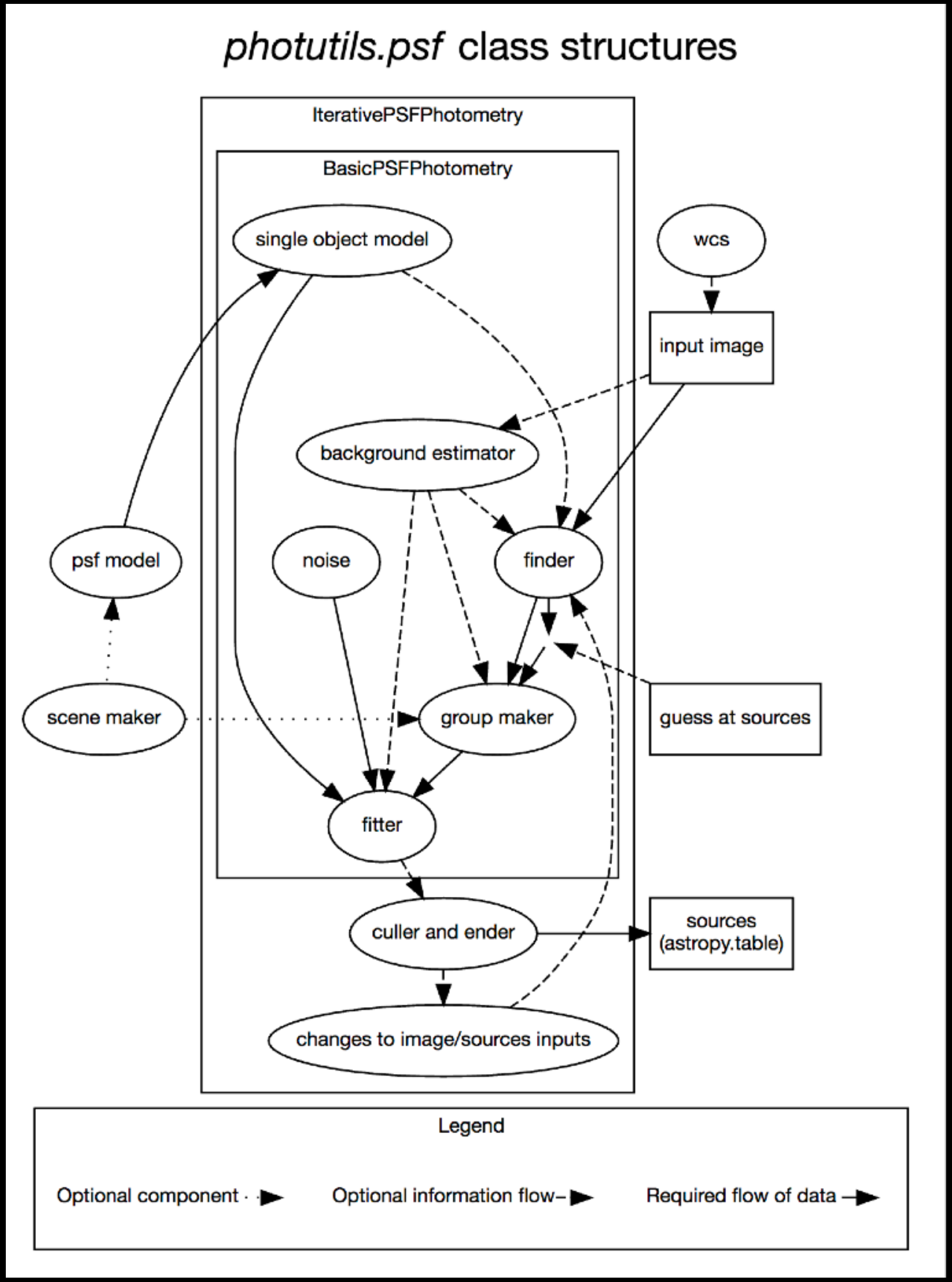
Additional PSF fitting routines

Non-point source fitting

2D ePSF framework

API stability, convenience functions

??? You tell me!



Further inclusion of astropy convenience

What can I ever do for *photutils.psf*?

Find bugs, submit issues

Suggest changes/improvements

Review code

Tell us how it can be made easier to use or swap to using