AegeanTools Documentation

Release 2.3.0

PaulHancock

Jan 12, 2024

CONTENTS:

CHAPTER

ONE

AEGEANTOOLS MODULES

1.1 angle_tools

1.2 BANE

1.3 catalogs

1.4 cluster

1.5 fits_tools

1.6 fitting

1.7 flags

Flag constants for use by Aegean.

1.8 MIMAS

1.9 models

1.10 msq2

1.11 regions

1.12 source_finder

1.13 wcs_helpers

CHAPTER

TWO

AEGEANTOOLS SCRIPTS

The following scripts are provided as part of the AegeanTools package:

- *aegean* - Aegean source finding
- *BANE* - Background and Noise Estimation
- *MIMAS* - Multi-resolution Image Mask for Aegean Software
- *AeRes* - Aegean Residuals
- *AeReg* - Aegean Regrouping
- *SR6* - Shrink Ray

CHAPTER

THREE

AEREG

The regrouping and rescaling operations that were introduced as part of the priorized fitting have been moved into the cluster module. The script `AeReg` will allow a user to access these operations from the command line such that they can see how the regrouping and rescaling operations will work before having to do the priorized fitting.

```
usage: regroup [-h] --input INPUT --table TABLES [--eps EPS] [--noregroup] [--ratio␣
↪RATIO] [--psfheader PSFHEADER]
               [--debug]

optional arguments:
  -h, --help            show this help message and exit

Required:
  --input INPUT         The input catalogue.
  --table TABLES        Table outputs, format inferred from extension.

Clustering options:
  --eps EPS             The grouping parameter epsilon (~arcmin)
  --noregroup           Do not perform regrouping (default False)

Scaling options:
  --ratio RATIO         The ratio of synthesized beam sizes (image psf / input catalog␣
↪psf).
  --psfheader PSFHEADER
                        A file from which the *target* psf is read.

Other options:
  --debug               Debug mode.
```

CHAPTER

FOUR

AERES

If you want to get residual maps, or model maps, from Aegean then this tool is what you are looking for.

AeRes will take an image, and Aegean catalog, and write a new image with all the sources removed. You can also ask for an image that has just the sources in it.

You can use AeRes as shown below:

```
usage: AeRes [-h] [-c CATALOG] [-f FITSFILE] [-r RFILE] [-m MFILE] [--add] [--mask] [--
↪sigma SIGMA] [--frac FRAC]
             [--racol RA_COL] [--deccol DEC_COL] [--peakcol PEAK_COL] [--acol A_COL] [--
↪bcol B_COL] [--pacol PA_COL]
             [--debug]

optional arguments:
  -h, --help            show this help message and exit

I/O arguments:
  -c CATALOG, --catalog CATALOG
                        Catalog in a format that Aegean understands. RA/DEC should be in␣
↪degrees, a/b/pa should be in
                        arcsec/arcsec/degrees.
  -f FITSFILE, --fitsimage FITSFILE
                        Input fits file.
  -r RFILE, --residual RFILE
                        Output residual fits file.
  -m MFILE, --model MFILE
                        Output model file [optional].

Config options:
  --add                 Add components instead of subtracting them.
  --mask                Instead of subtracting sources, just mask them
  --sigma SIGMA         If masking, pixels above this SNR are masked(requires input␣
↪catalogue to list rms)
  --frac FRAC           If masking, pixels above frac*peak_flux are masked for each␣
↪source

Catalogue options:
  --racol RA_COL        RA column name
  --deccol DEC_COL      Dec column name
  --peakcol PEAK_COL    Peak flux column name
  --acol A_COL          Major axis column name
```

(continues on next page)

(continued from previous page)

```
  --bcol B_COL          Minor axis column name
  --pacol PA_COL        Position angle column name

Extra options:
  --debug               Debug mode.
```

The acceptable formats for the catalogue file are anything that Aegean can write. Use `aegean.py --tformats` to see the formats that Aegean can support on your machine. Usually the best idea is to just edit a table that Aegean has created.

CHAPTER

FIVE

BANE

5.1 Motivation

Aegean has an inbuilt background and noise calculation algorithm (the zones algorithm) which is very basic and is useful for images that have a slowly changing background and noise. For images with more complicated background and noise statistics it is advised that you use an external program to pre-compute these maps and then feed them into Aegean with the –background and –noise flags. Since I have not come across a program that can calculate these images in a speedy manner I have built one myself.

5.2 Aim

The quick-and-dirty method for calculating the background and noise of an image is to pass a sliding boxcar filter over the image and, for each pixel, calculate the mean and standard deviation of all pixels within a box centred on that pixel. The problem with this approach is two-fold: one - although it is easy to code it is very time consuming, and two - the standard deviation is biased in the presence of sources.

The aim of BANE is to provide an accurate measure of the background and noise properties of an image, and to do so in a relatively short amount of time.

5.3 Methodology

There are two main techniques that BANE uses to reduce the compute time for a background and noise calculation, whilst retaining a high level of accuracy.

- Since radio images have a high level of correlation between adjacent pixels, BANE does not calculate the mean and standard deviation for every pixel. It will calculate these quantities on a sparse grid of pixels and then interpolate to give the final background and noise images. For a grid spacing of 5x5 pixels this reduces the total computations by a factor of 25, with only a small amount of time required for interpolation.
- To avoid contamination from source pixels BANE performs sigma clipping. Pixels that are greater than 3sigma from the mean are masked, and this processes is repeated 3 times. The non-masked pixels are then used to calculate the median and std which are equated to be the background and rms.

BANE offers the user a set of parameters that can be used to tune the speed/accuracy to a users desire. The parameters are the grid spacing (in each of the x,y directions), and the size of the box (again in x,y directions) over which the background and noise is calculated. A grid spacing of 1x1 is equivalent to a traditional box-car smooth using the median and std.

Since we define the noise to be the variance about the median, it is necessary for BANE to make two passes over the data: the first pass calculates the background level, and the second pass calculates the deviation from this background

level. This requirement doubles the run time of BANE, however for images where the background level is known to be slowly changing (on scales of the box size), a single pass is all that is required.

5.4 Processing steps

The implementation of the process isn't that important but the idea is as follows:

1. select every Nth pixel in the image to form a grid (where N is the grid size, and can be different in the x and y directions).
2. around each grid point draw a box that is MxM pixels wide (where M is the box size, and can be different in the x,y directions).
3. do sigma clipping (3 rounds at 3sigma) to remove the contribution of source pixels
4. calculate the median of all pixels within the box and use that as the background
5. run a linear interpolation between the grid points to make a background image
6. calculate a background subtracted image (data-background)
7. repeat steps 1-4 on the background subtracted image, but instead of calculating the median, use the std.

5.5 Usage

The usage of BANE is described in the help text as follows:

```
usage: BANE [-h] [--out OUT_BASE] [--grid STEP_SIZE STEP_SIZE] [--box BOX_SIZE BOX_SIZE]␣
↪[--cores CORES]
            [--stripes STRIPES] [--slice CUBE_INDEX] [--nomask] [--noclobber] [--debug]␣
↪[--compress] [--cite]
            [image]

positional arguments:
  image

optional arguments:
  -h, --help            show this help message and exit

Configuration Options:
  --out OUT_BASE        Basename for output images default: FileName_{bkg,rms}.fits
  --grid STEP_SIZE STEP_SIZE
                        The [x,y] size of the grid to use. Default = ~4* beam size␣
↪square.
  --box BOX_SIZE BOX_SIZE
                        The [x,y] size of the box over which the rms/bkg is calculated.␣
↪Default = 5*grid.
  --cores CORES         Number of cores to use. Default = all available.
  --stripes STRIPES     Number of slices.
  --slice CUBE_INDEX    If the input data is a cube, then this slice will determine the␣
↪array index of the image which
                        will be processed by BANE
  --nomask              Don't mask the output array [default = mask]
  --noclobber           Don't run if output files already exist. Default is to␣
```

(continues on next page)

(continued from previous page)

```
↪run+overwrite.
  --debug               debug mode, default=False
  --compress            Produce a compressed output file.
  --cite                Show citation information.
```

5.6 Description of options

- `--compress`: This option causes the output files to be very small. This compression is done by writing a fits image without any interpolation. Files that are produced in this way have extra keys in their fits header, which are recognized by Aegean. When compressed files are loaded by aegean they are interpolated (expanded) to their normal sizes.
- `--nomask`: By default BANE will mask the output image to have the same masked pixels as the input image. This means that nan/blank pixels in the input image will be nan in the output image. This doesn't happen if `--compress` is selected.
- `--stripes`: BANE will break the image into this many sections and process each in turn. By default this is equal to the number of cores, so that all stripes will be processed at the same time. By setting stripes>cores it is possible to reduce the instantaneous memory usage of BANE at the cost of run time.

CHAPTER

SIX

MIMAS

6.1 Motivation

Prior to 1.8.1, the Aegean source-finding program operated on the entire input image. To return a list of sources that were contained within a sub region other programs were required (For example stilts). Normally this is not a big concern as the filtering process is rather fast. Since radio telescopes have circular primary beam patterns, and fits images are forced to be rectangular, the images produced by imaging pipelines would contain the area of interest along with some amount of extra sky. If the pixels outside the area of interest are not flagged or masked by the imaging pipeline then extra tools are required. Not being able to find any nifty tools to do this job for me, I decided to create the Milti-resolution Image Mask for Aegean Software - MIMAS. There are three main features that I was looking for, each of which are solved by MIMAS.

6.2 Aims

MIMAS was created with the following three goals in mind:

- to be able to create and manipulate arbitrary shaped regions that could be used to describe areas of sky. The method of manipulation is intended to parallel that of set operations so that you can easily take the intersection, union, or difference of regions, in order to create regions as simple as circles and polygons, to some horrendous thing that describes the sky coverage of a survey.
- to be able to store these regions in a file format that can be easily stored and transmitted.
- to be able to use these regions to mask image files, or to restrict the operation of Aegean to a sub section of a given image.

6.3 Methodology

MIMAS is a wrapper script that uses the regions module that is now part of AegeanTools. The regions module contains a suite of unit tests and a single class called Region. The Region class is built on top of the HealPy module, which is in turn a wrapper around the HEALPix software.

6.4 Usage

MIMAS has five modes of operation:

- create a new region from a combination of: stored regions, circles, or polygons.
- create a new region from a DS9 .reg file
- convert a region.mim file into a .reg format that can be used as an overlay for DS9.
- use a .fits image to create a .mim region file as if the image were a mask
- use a region file and a .fits image to create a new fits image where pixels that are OUTSIDE the given region have been masked.

The operation of MIMAS is explained by the following help text:

```
usage: MIMAS [-h] [-o OUTFILE] [-depth N] [+r [filename [filename ...]]] [-r [filename␣
↪[filename ...]]]
             [+c ra dec radius] [-c ra dec radius] [+p [ra [dec ...]]] [-p [ra [dec ...
↪]]] [-g]
             [--mim2reg region.mim region.reg] [--reg2mim region.reg region.mim] [--
↪mim2fits region.mim region_MOC.fits]
             [--mask2mim mask.fits region.mim] [--intersect region.mim] [--area region.
↪mim]
             [--maskcat region.mim INCAT OUTCAT] [--maskimage region.mim file.fits␣
↪masked.fits]
             [--fitsmask mask.fits file.fits masked_file.fits] [--negate] [--colnames RA_
↪name DEC_name]
             [--threshold THRESHOLD] [--debug] [--version] [--cite]

optional arguments:
  -h, --help            show this help message and exit

Creating/modifying regions:
  Must specify -o, plus or more [+-][cr]

  -o OUTFILE            output filename
  -depth N              maximum nside=2**N to be used to represent this region.␣
↪[Default=8]
  +r [filename [filename ...]]
                        add a region specified by the given file (.mim format)
  -r [filename [filename ...]]
                        exclude a region specified by the given file (.mim format)
  +c ra dec radius      add a circle to this region (decimal degrees)
  -c ra dec radius      exclude the given circles from a region
  +p [ra [dec ...]]     add a polygon to this region ( decimal degrees)
  -p [ra [dec ...]]     remove a polygon from this region (decimal degrees)
  -g                    Interpret input coordinates are galactic instead of equatorial.

Using already created regions:
  --mim2reg region.mim region.reg
                        convert region.mim into region.reg
  --reg2mim region.reg region.mim
                        Convert a .reg file into a .mim file
  --mim2fits region.mim region_MOC.fits
```

(continues on next page)

(continued from previous page)

```
                        Convert a .mim file into a MOC.fits file
  --mask2mim mask.fits region.mim
                        Convert a masked image into a region file
  --intersect region.mim, +i region.mim
                        Write out the intersection of the given regions.
  --area region.mim     Report the area of a given region

Masking files with regions:
  --maskcat region.mim INCAT OUTCAT
                        use region.mim as a mask on INCAT, writing OUTCAT
  --maskimage region.mim file.fits masked.fits
                        use region.mim to mask the image file.fits and write masekd.fits
  --fitsmask mask.fits file.fits masked_file.fits
                        Use a fits file as a mask for another fits file. Values of blank/
↪nan/zero are considered to be
                        mask=True.
  --negate              By default all masks will exclude data that are within the given␣
↪region. Use --negate to exclude
                        data that is outside of the region instead.
  --colnames RA_name DEC_name
                        The name of the columns which contain the RA/DEC data.␣
↪Default=(ra,dec).

Extra options:
  --threshold THRESHOLD
                        Threshold value for input mask file.
  --debug               debug mode [default=False]
  --version             show program's version number and exit
  --cite                Show citation information.

Regions are added/subtracted in the following order, +r -r +c -c +p -p. This means that␣
↪you might have to take multiple passes to construct overly complicated regions.
```

6.4.1 Data model and operation

At the most basic level, The Regions class takes a description of a sky area, either a circle or a polygon, and converts it into a list of HELAPix pixels. These pixels are stored as a python set, making it easy to implement set operations on these regions. HEALpix is a parameterization of the sky that maps diamond shaped regions of equal area, onto a pixel number. There are many interesting properties of the nested HEALPix parameterization that make it easy to implement the Region class. Firstly, HEALPix can represent areas of sky that are as coarse as 1/12th of the entire sky, to regions that are 1/2^30 times smaller. A depth or resolution parameter of 2^12 represents a pixel size of less than one arcminute. By making use of different resolutions of pixels, it is possible to represent any region in an efficient manner. The sky area that is represented by a Region is a combination of pixels of different resolutions, with the smallest resolution being supplied by the user.

6.5 File format

The MIMAS program is able to take a description of a region and save it to a file for use by many programs. Since he underlying data model is a dictionary of sets, the fastest and easiest file format to use is that given by the cPickle module (a binary file). These files are small, fast to read and write, and accurately reproduce the region object that was stored. The MIMAS program writes files with an extension of .mim.

6.6 Interaction with Aegean

Region files with .mim extension that are created by MIMAS can be used to restrict Aegean to the given region of an image. Use the `--region region.mim` option when running Aegean to enable this.

CHAPTER

SEVEN

SR6

BANE is able to output compressed background and rms images using the `--compress` option. If you have a compressed file and want to expand it to have the same number of pixels as your original image then you need to use SR6.

If you have an image that you, for some reason, want to compress using a super-lossy algorithm known as decimation, then SR6 is what you want.

Usage is:

```
usage: SR6 [-h] [-o OutputFile] [-f factor] [-x] [-m MaskFile] [--debug] [--version] [--
↪cite] [infile]

optional arguments:
  -h, --help     show this help message and exit

Shrinking and expanding files:
  infile         input filename
  -o OutputFile  output filename
  -f factor      reduction factor. Default is 4x psf.
  -x             Operation is expand instead of compress.
  -m MaskFile    File to use for masking pixels.

Other options:
  --debug        Debug output
  --version      show program's version number and exit
```

In order to be able to expand a file, the file needs to have some special keywords in the fits header. These are inserted automatically by BANE, but you could probably fidget them for yourself if you had the need.

You should be able to shrink any file that you choose.

CHAPTER

EIGHT

AEGEAN

8.1 Simple usage

Suggested basic usage (with mostly default parameters):

`aegean RadioImage.fits --table=Catalog.fits`

Usage and short description can be obtained via `aegean`, which is replicated below.

```
This is Aegean 2.3.0-(2022-08-17)
usage: aegean [-h] [--find] [--hdu HDU_INDEX] [--beam BEAM BEAM BEAM] [--slice SLICE] [--
↪progress] [--forcerms RMS]
              [--forcebkg BKG] [--cores CORES] [--noise NOISEIMG] [--background␣
↪BACKGROUNDIMG] [--psf IMGPSF]
              [--autoload] [--out OUTFILE] [--table TABLES] [--tformats] [--blankout] [--
↪colprefix COLUMN_PREFIX]
              [--maxsummits MAX_SUMMITS] [--seedclip INNERCLIP] [--floodclip OUTERCLIP]␣
↪[--island] [--nopositive]
              [--negative] [--region REGION] [--nocov] [--priorized PRIORIZED] [--ratio␣
↪RATIO] [--noregroup]
              [--input INPUT] [--catpsf CATPSF] [--regroup-eps REGROUP_EPS] [--save] [--
↪outbase OUTBASE] [--debug]
              [--versions] [--cite]
              [image]

positional arguments:
  image

optional arguments:
  -h, --help            show this help message and exit

Configuration Options:
  --find                Source finding mode. [default: true, unless --save or --measure␣
↪are selected]
  --hdu HDU_INDEX       HDU index (0-based) for cubes with multiple images in extensions.
↪ [default: 0]
  --beam BEAM BEAM BEAM
                        The beam parameters to be used is "--beam major minor pa" all in␣
↪degrees. [default: read from
                        fits header].
  --slice SLICE         If the input data is a cube, then this slice will determine the␣
```

(continues on next page)

(continued from previous page)

```
↪array index of the image which
                        will be processed by aegean
  --progress            Provide a progress bar as islands are being fit. [default: False]
  --cores CORES         Number of CPU cores to use when calculating background and rms␣
↪images [default: all cores]

Input Options:
  --forcerms RMS        Assume a single image noise of rms. [default: None]
  --forcebkg BKG        Assume a single image background of bkg. [default: None]
  --noise NOISEIMG      A .fits file that represents the image noise (rms), created from␣
↪Aegean with --save or BANE.
                        [default: none]
  --background BACKGROUNDIMG
                        A .fits file that represents the background level, created from␣
↪Aegean with --save or BANE.
                        [default: none]
  --psf IMGPSF          A .fits file that represents the local PSF.
  --autoload            Automatically look for background, noise, region, and psf files␣
↪using the input filename as a
                        hint. [default: don't do this]

Output Options:
  --out OUTFILE         Destination of Aegean catalog output. [default: No output]
  --table TABLES        Additional table outputs, format inferred from extension.␣
↪[default: none]
  --tformats            Show a list of table formats supported in this install, and␣
↪their extensions
  --blankout            Create a blanked output image. [Only works if cores=1].
  --colprefix COLUMN_PREFIX
                        Prepend each column name with "prefix_". [Default = prepend␣
↪nothing]

Source finding/fitting configuration options:
  --maxsummits MAX_SUMMITS
                        If more than *maxsummits* summits are detected in an island, no␣
↪fitting is done, only
                        estimation. [default: no limit]
  --seedclip INNERCLIP  The clipping value (in sigmas) for seeding islands. [default: 5]
  --floodclip OUTERCLIP
                        The clipping value (in sigmas) for growing islands. [default: 4]
  --island              Also calculate the island flux in addition to the individual␣
↪components. [default: false]
  --nopositive          Don't report sources with positive fluxes. [default: false]
  --negative            Report sources with negative fluxes. [default: false]
  --region REGION       Use this regions file to restrict source finding in this image.␣
↪Use MIMAS region (.mim) files.
  --nocov               Don't use the covariance of the data in the fitting proccess.␣
↪[Default = False]

Priorized Fitting config options:
  in addition to the above source fitting options
```

(continues on next page)

(continued from previous page)

```
  --priorized PRIORIZED
                        Enable priorized fitting level n=[1,2,3]. 1=fit flux, 2=fit flux/
↪position, 3=fit
                        flux/position/shape. See the GitHub wiki for more details.
  --ratio RATIO         The ratio of synthesized beam sizes (image psf / input catalog␣
↪psf). For use with priorized.
  --noregroup           Do not regroup islands before priorized fitting
  --input INPUT         If --priorized is used, this gives the filename for a catalog of␣
↪locations at which fluxes will
                        be measured.
  --catpsf CATPSF       A psf map corresponding to the input catalog. This will allow␣
↪for the correct resizing of
                        sources when the catalog and image psfs differ
  --regroup-eps REGROUP_EPS
                        The size in arcminutes that is used to regroup nearby components␣
↪into a single set of components
                        that will be solved for simultaneously

Extra options:
  --save                Enable the saving of the background and noise images. Sets --
↪find to false. [default: false]
  --outbase OUTBASE     If --save is True, then this specifies the base name of the␣
↪background and noise images.
                        [default: inferred from input image]
  --debug               Enable debug mode. [default: false]
  --versions            Show the file versions of relevant modules. [default: false]
  --cite                Show citation information.
```

8.1.1 Example usage:

The following commands can be run from the Aegean directory right out of the box, since they use the test images that are included with Aegean.

- Blind source finding on a test image and report results to stdout
  - `aegean tests/test_files/1904-66_SIN.fits`
- As above but put the results into a text file
  - `aegean tests/test_files1904-66_SIN.fits --table out.csv`
  - The above creates a file `out_comp.csv` for the components that were fit
- Do source finding using a catalog input as the initial parameters for the sources
  - `aegean --priorized 1 --input out_comp.csv tests/test_files/1904-66_SIN.fits`
- Source-find an image and save results to multiple tables
  - `aegean --table catalog.csv,catalog.vot,catalog.fits tests/test_files1904-66_SIN.fits`
- Source-find an image and report the components and islands that were found
  - `aegean --table catalog.vot --island tests/test_files1904-66_SIN.fits`

  - The above creates two files: `catalog_comp.vot` for the components, and `catalog_isle.vot` for the islands. The island column of the components maps to the island column of the islands.

- Source-find a sub-region of an image
  - `aegean --region=region.mim tests/test_files1904-66_SIN.fits`
  - The `region.mim` is a region file in the format created by *MIMAS*

8.2 Output formats

Aegean supports a number of output formats. There is the Aegean default, which is a set of columns separated by spaces, with header lines starting with #. The format is described within the output file itself.

The Aegean default output (which goes to STDOUT) does not contain all of the columns listed below. Tables created with the `--table` option contain all the following columns, and as much meta-data as I can manage to pack in.

8.2.1 Table description

Columns included in output tables have the following columns:

- island - numerical indication of the island from which the source was fitted
- source - source number within that island
- background - background flux density in Jy/beam
- local_rms - local rms in Jy/beam
- ra_str - RA J2000 sexigessimal format
- dec_str - dec J2000 sexigessimal format
- ra - RA in degrees
- err_ra - source-finding fitting error on RA in degrees
- dec - dec in degrees
- err_dec - source-finding fitting error on dec in degrees
- peak_flux - peak flux density in Jy/beam
- err_peak_flux - source-finding fitting error on peak flux density in Jy/beam
- int_flux - integrated flux density in Jy. This is calculated from a/b/peak_flux and the synthesized beam size. It is not fit directly.
- err_int_flux - source-finding fitting error on integrated flux density in Jy
- a - fitted semi-major axis in arcsec
- err_a - error on fitted semi-major axis in arcsec
- b - fitted semi-minor axis in arcsec
- err_b- error on fitted semi-minor axis in arcsec
- pa - fitted position angle in degrees
- err_pa - error on fitted position angle in degrees
- flags - fitting flags (should be all 0 for a good fit)

- residual_mean - mean of the residual flux remaining in the island after fitted Gaussian is subtracted
- residual_std - standard deviation of the residual flux remaining in the island after fitted Gaussian is subtracted
- uuid - a universally unique identifier for this component.
- psf_a - the semi-major axis of the point spread function at this location (arcsec)
- psf_b - the semi-minor axis of the point spread function at this location (arcsec)
- psf_pa - the position angle of the point spread function at this location (arcsec)

An island source will have the following columns:

- island - numerical indication of the island
- components - the number of components within this island
- background - background flux density in Jy/beam
- local_rms - local rms in Jy/beam
- ra_str - RA J2000 sexigessimal format
- dec_str - dec J2000 sexigessimal format
- ra - RA in degrees, of the brightest pixel in the island
- dec - dec in degrees, of the brightest pixel in the island
- peak_flux - peak flux density in Jy/beam, of the brightest pixel in the island
- int_flux - integrated flux density in Jy. Computed by summing pixels in the island, and dividing by the synthesized beam size.
- err_int_flux - Error in the above. Currently Null/None since I don't know how to calculate it.
- eta - a correction factor for int_flux that is meant to account for the flux that was not included because it was below the clipping limit. For a point source the true flux should be int_flux/eta. For extended sources this isn't always the case so use with caution.
- x_width - the extent of the island in the first pixel dimension, in pixels
- y_width - the extent of the island in the second pixel dimension, in pixels
- max_angular_size - the largest distance between to points on the boundary of the island, in degrees.
- pa - the position angle of the max_angular_size line
- pixels - the number of pixels within the island
- beam_area - the area of the synthesized beam (psf) in deg^2
- area - the area of the island in deg^2
- flags - fitting flags (should be all 0 for a good fit)
- uuid - a universally unique identifier for this island.

Note: Column names with 'ra/dec' will be replaced with a 'lat/lon' version if the input image has galactic coordinates in the WCS.

8.2.2 Table Types

The most useful output is to use tables. Table output is supported by sqlite and astropy and there are three main types: database, votable, and ascii table. Additionally you can output ds9 region files by specifying a .reg file extension.

8.2.3 Database:

This format requires that the sqlite module is available. This is nearly always true by default, but if you get a crash then check that you can `import sqlite3` from a python terminal before submitting a bug report.

Use `--table out.db` to create a database file containing one table for each source type that was discovered. The table names are 'components', 'islands', and 'simples'. Islands are created when –island is enabled. Components are elliptical gaussian fits and are the default type of source to create. Simples are sources that have been created by using the –measure option.

The columns of the database are self explanatory though they have no units. All fluxes are in Jy, major and minor axes are in arcseconds, and the position angle is in degrees. Errors that would normally be reported as -1 in other formats are stored as nulls in the database tables.

8.2.4 VOTable:

VOTables are difficult to work with as a human, but super awesome to work with when you have TopCat or some other VO enabled software.

VOTable output is supported by AstroPy (0.3+ I think). If you don't have the right version of AstroPy you can still run Aegean but will not be able to write VOTables. You will be told this when Aegean runs.

Use `--table out.vot` or `--table out.xml` to create a VOTable. Each type of sources that you find will be saved to a different file. Components are saved to out_comp.vot, islands are saved to out_isle.vot, and simple sources will be saved to out_simp.vot (or xml as appropriate). See above for a description of the source types.

8.2.5 ASCII tables:

ASCII tables are supported by AstroPy (0.4+ I think). As with VOTables, if you don't have the right version of AstroPy then Aegean will still run but it will tell you that you can't write ASCII tables.

There are currently four types of ascii tables that can be used:

- csv -> comma separated values
- tab -> tab separated values
- tex -> LaTeX formatted table
- html -> an html formatted table

Use `--table out.html,out.tex` etc.. for the type of table you are interested in. All tables have column headers that are the same as the variable names. These should be easily discernible. The units are Jy for fluxes, arcseconds for major/minor axes, and degrees for position angles.

As with other table formats the file names will be modified to out_comp.html, out_simp.csv, etc… to denote the different types of sources that are contained within.

8.2.6 FITS binary tables

use extension `fits` or `FITS` (but not `fit` or `FIT`) to write output tables. Functionality supported by AstroPy. These are binary tables and only the header is human readable.

8.2.7 DS9 region files

Use extension `reg` for the output table to get `DS9` region files. Both components and islands are supported in this format with `_comp.reg` and `_isle.reg` being the corresponding filenames.

Component sources in the `_comp.reg` files will be shown as ellipses at the location of each component, with the fitted size/orientation. Each ellipse will be annotated with the island and component number such that Island 10, component 0 will appear as `(10,0)`.

Island sources will appear as an outline of the pixels that comprise the island. Each island also has an annotation of the island number, and a diagonal line that represents the largest angular scale.

8.2.8 Flags

There are six different flags that can be set by Aegean during the source finding and fitting process. In the `STDOUT` version of the Aegean catalog the flags column is written in binary format with a header that read ZWNCPES. These six flags correspond to:

Abbreviation	Name	Numerical value	description
S	FITERRSMAL	1	This flag is set when islands are not able to be fit due to there being fewer pixels than free parameters.
E	FITERR	2	This flag is set when an error occurs during the fitting process. eg the fit doesn't converge.
P	FIXED2P	4	If a component is forced to have the shape of the local point spread function then this flag is set. This flag is often set at the same time as the FITERRSMALL, or FIXEDCRICULAR
C	FIXEDCRICULAR	8	If a source is forced to have a circular shape then this flag will be fit.
N	NOTFIT	16	If a component is not fit then this flag is set. This can because and island has reached the `--maxsummits` limit, or `--measure` mode has been invoked.
W	WCSERR	32	If the conversion from pixel to sky coordinates doesn't work then this flag will be set. This can happen for strange projections, but more likely when an image contains pixles that don't have valid sky coordinates.
Z	PRIORIZED	64	This flag is set when the source was fit using priorized fitting.

Note that the flags column will be the summation of the numerical value of the above flags. So flags=7 means that flags P, E, and S have been set. This all makes more sense when you print the flags in binary format.

8.3 Priorized fitting

This functionality is designed to take an input catalog of sources (previously created by Aegean), and use the source positions and morphologies to measure the flux of these sources within an image.

When `--priorized x` is invoked the following will happen:

- input catalog is read from the file specified by `--input`. This file needs to contain all the properties of a source, including island numbers and uuids. The easiest way to make these files is to just take the output from Aegean and modify it as needed.
- The sources within the catalog are regrouped. The regrouping will recreate islands of sources based on their positions and morphologies. Sources will be grouped together if they overlap at the FHWM. Note that this is different from the default island grouping that Aegean does, which is based on pixels within an island. If `--noregroup` is set then the island grouping will be based on the (isle,source) id's in the input catalog.
- Fitting will be done on a per island basis, with multiple sources being fit at the same time. The user is able to control which parameters are allowed to vary at this stage by supplying a number x to `--priorized x`.
- Fitting will be done on all pixels that are greater than the `--floodclip` limit. If an island has no pixels above this limit then no output source will be generated. Note the special case of `--floodclip -1` which will simply use all pixels within some rectangular region around each input source.
- Output will be written to files as specified by `--table`.

The parameters that are free/fixed in the fitting process depends on the 'level' of priorized fitting that is requested. Level:

1. Only the flux is allowed to vary. Use this option where you would have otherwise used `--measure`.
2. Flux and positions are allowed to vary, shape is fixed.
3. Everything is allowed to vary.

In the case that the psf of the input catalogue and the supplied image are different there are three options for describing this difference:

1. Use the `--ratio` option, which specifies the ratio of major axes (image psf / catalogue psf). This method works well for small images where the psf doesn't really change over the image, or when the difference is small.
2. Supply a psf map for the input catalogue using the `--catpsf` option. This will give you ultimate fine control over what the psf of your input catalogue is.
3. Include the psf parameters in the input catalogue as columns `psf_a, psf_b, psf_pa`

Note: If you know how to perform the deconvolve-convolve step for two synthesized beams that are not simply scaled versions of each other, then please let me know so that I can implement this.

8.3.1 Notes on input tables:

Any [[format|Output-Formats]] that Aegean can write, is an acceptable input format. The easiest way to create an input table is to modify and existing catalogue. The following columns are used for priorized fitting:

- Required:
  - `ra`, `dec`, `peak_flux`, `a`, `b`, `pa`
- Optional:
  - `psf_a`, `psf_b`, `psf_pa` used for re-scaling the source shapes.
  - `uuid` copied from input to output catalogues

- err_ra, err_dec copied from input to output catalogues when positions are not being fit
- err_a, err_b, err_pa copied from input to output catalogues when shapes are not being fit

Parameters a, b, err_a, err_b, psf_a, and psf_b all have units of arcsec. Parameters ra, dec, pa,err_ra, err_dec, and err_pa all have units of degrees.

PYTHON MODULE INDEX

INDEX

A

M