# sewpy Documentation

***Release 1.0dev***

**The MegaLUT developers**

**Sep 12, 2017**

# Contents

The tiny `sewpy` module let's you run SExtractor as if it would all be native python...

```
import sewpy
sew = sewpy.SEW(params=["X_IMAGE", "Y_IMAGE", "FLUX_RADIUS(3)", "FLAGS"],
        config={"DETECT_MINAREA":10, "PHOT_FLUXFRAC":"0.3, 0.5, 0.8"})
out = sew("myimage.fits")
print out["table"] # this is an astropy table.
```

... but also allows for a more sophisticated use, for instance if you want to use existing SExtractor input files.

Why yet another SExtractor wrapper ? Because we needed one that:

- is based on astropy (in particular `astropy.table`),
- works with both python 2 and 3,
- uses standard `logging` (and also logs SExtractor's stdout & stderr to file),
- uses `tempfile` to hide all input and output files, *except if you want to see them*
- has some convenience functionality to use SExtractor's `ASSOC` process (give me an input catalog, and I append columns with SExtractor measurements to it).

The **demos** in the `examples` directory can be run without installing sewpy, and provide a quick overview.

Your are viewing the documentation of `sewpy`. The code, including the sources of the present documentation, is hosted on github.com/megalut/sewpy.

CHAPTER 1

# Contents

## Installing sewpy

The requirements are astropy (version 0.4.2 or later), and obviously SExtractor (`sewpy` does not require a specific version).

To get `sewpy`, clone the project on github, or click here to get a tarball of the latest version. You now have (at least) two options to install the module.

### Method 1, if you just want to use the module

Extract the tarball if required, `cd` into it, and

```
python setup.py install --user
```

For a system-wide install, remove the `--user`. More information about how to install python modules with distutil can be found in the official Distutils documentation.

### Method 2, if you intend to develop the module

A simple solution is to add your local clone of `sewpy` to your python path, for instance by modifying your `.bash_profile` or equivalent:

```
export PYTHONPATH=${PYTHONPATH}:/path/to/megalut-sewpy-1.0
```

# User manual

## General philosophy (to be rewritten)

- If you do not specify a workdir, I'll write all my required internal files somewhere in /tmp, and you don't have to bother about this (this is done using the tempfile module).
- `params` (a list) refers to the features that you want SExtractor to measure (e.g., settings you find in "default.param").
- `config` (a dict) refers to the settings (e.g., stuff you find in "default.sex").
- A SEW instance can well be reused for different images that you want to analyse with the same params but a different config. Indeed you usually don't want to change params from image to image, but you might have to change the config (e.g., gain, seeing, ...).
- In the params list, you have to specify all the parameters that you want to be measured.
- On the other hand, in the config dict, you only have to give those settings that deviate from the *default*! We take as *default* the output of "sextractor -d" (if not told otherwise).
- When repeatedly calling run(), we avoid writing the SExtractor input files to disk over and over again. Instead, param is written only once, and config settings are passed as command line arguments to the SExtractor executable, superseding the default config. So to change config from image to image, simply edit se.config between calls of run().
- There is special helper functionality for using ASSOC (see note below)

## Logging

Sewpy uses the logging module. To see a detailed log of what is going on, insert this into your script:

```
import logging
logging.basicConfig(format='%(levelname)s: %(name)s(%(funcName)s): %(message)s',␣
↪level=logging.DEBUG)
```

## The ASSOC helper

The ASSOC helper assists you in measuring galaxies from an existing input catalog, instead of just making a new catalog of all sources. In summary, you pass an existing input catalog, and you'll get this same catalog as output, but with the new columns corresponding to the SExtractor params appended.

To use the ASSOC helper:

- Add `VECTOR_ASSOC(3)` to your params (The following is valid only for Astropy v1.1.2 and prior: at the beginning, not at the end, of the params list).
- Add for instance `{"ASSOC_RADIUS":10.0, "ASSOC_TYPE":"NEAREST"}` to your config. These values are the defaults used if you don't specify anything.
- Give the relevant arguments (`assoc_cat, assoc_xname, assoc_yname`) when calling.

The output will contain an astropy table, with the same rows as `assoc_cat`, but to which the new SExtractor columns will be appended. Those SExtractor columns might be **masked** columns (leading to a masked table), as some of your sources might not have been found by SExtractor. Note that the attribute mytable.masked tells you if an astropy table "mytable" is masked. To make it even more foolproof, I systematically add a boolean column named `prefix + "assoc_flag"`. True means that the source was found.

## API documentation of the `sewpy` module

sewpy: Source Extractor Wrapper for Python

Recent improvements (latest on top):

- new loglevel option to adjust sewpy's overall "verbosity" on instantiation.
- better verbosity about masked output of ASSOC procedure
- ASSOC helper implemented
- run() now returns a dict containing several objects, such as the output astropy table, catfilepath, workdir, and logfilepath.
- now also works with vector parameters such as MAG_APER(4)
- possibility to "nice" SExtractor
- a log file is written for every run() if not told otherwise
- filenames change according to FITS image file name where required
- but you can also pass an "imgname" argument to run, and this will be used instead.
- params and config files are written only once, as discussed
- appropriate warnings and behaviour when a workdir already exists, or when you rerun on the same file
- possibility to use existing param / config / conv / nnw files
- run() returns either the catalog, or the filepath to the catalog

To do:

- move "config" to run ?
- check that all masked columns of ASSOC do indeed share the same mask.
- implement _check_config()
- better detection of SExtractor failures
- implement raising Exceptions when SExtractor fails
- implement CHECK IMAGE "helper" ?
- give access to several conv and nnw settings (if needed)

**class** `sewpy.sewpy.`**`SEW`**(*workdir=None*, *sexpath='sex'*, *params=None*, *config=None*, *configfilepath=None*, *nice=None*, *loglevel=None*)

Holds together all the settings to run SExtractor executable on one or several images.

**`__init__`**(*workdir=None*, *sexpath='sex'*, *params=None*, *config=None*, *configfilepath=None*, *nice=None*, *loglevel=None*)

All arguments have default values and are optional.

**Parameters**

- **`workdir`** – where I'll write my files. Specify this (e.g., "test") if you care about the output files. If None, I create a unique temporary directory myself, usually in /tmp.
- **`sexpath`** – path to the sextractor executable (e.g., "sex" or "sextractor", if in your PATH)
- **`params`** (*`list of strings`*) – the parameters you want SExtractor to measure (i.e., what you would write in the "default.param" file)

- **config** (*dict*) – config settings that will supersede the default config (e.g., what you would change in the "default.sex" file)
- **configfilepath** – specify this if you want me to use an existing SExtractor config file as "default" (instead of the sextractor -d one)
- **nice** (*int*) – niceness with which I should run SExtractor. Use e.g. `19` for set lowest priority.
- **loglevel** (*string or int or logging.level...*) – verbosity, e.g. the python-level logging threshold for the sewpy module logger. For example, set this to "WARNING" and sewpy will no longer log simple INFOs. Choices are "DEBUG", "INFO", "WARNING", "ERROR", "CRITICAL". To disable logging, set `loglevel="CRITICAL"`

To use an existing SExtractor param-, conv-, or nnw-file, simply specify these in the config dict, using the appropriate SExtractor keys (PARAMETERS_NAME, FILTER_NAME, ...)

> **Warning:** When using *vector*-type params resulting in multiple columns (such as "FLUX_RADIUS(3)" in the example above), do not put these in the last position of the params list, otherwise astropy fails reading the catalog! This is probably due to the fact that the SExtractor header doesn't give a hint that multiple columns are expected when a vector-type param comes last. A workaround would be way too complicated.

**get_version**()
To find the SExtractor version, we call it without arguments and parse the stdout.

**Returns** a string (e.g. '2.4.4')

**__str__**()
A string summary representing the instance

**_check_params**()
Compares the params to a list of known params, and spits out a useful warning if something seems fishy.

**_check_config**()
Not yet implemented

**_set_instance_config**()
Sets config parameters that remain fixed for this instance. Called by __init__(). If needed, you could still mess with this config after __init__() has run.

**_get_params_filepath**()
Stays the same for a given instance.

**_get_config_filepath**()
Idem, stays the same for a given instance. Might return the non-default configfilepath, if set.

**_get_conv_filepath**()
Stays the same for a given instance.

**_get_psf_filepath**()
Stays the same for a given instance.

**_get_cat_filepath**(*imgname*)
This changes from image to image

**_get_assoc_filepath**(*imgname*)
Changes from image to image

**_get_log_filepath**(*imgname*)
Changes from image to image

**_write_params**(*force=False*)
Writes the parameters to the file, if needed.

**Parameters** **force** – if True, I overwrite any existing file.

**_write_default_config**(*force=False*)
Writes the *default* config file, if needed. I don't write this file if a specific config file is set.

**Parameters** **force** – if True, I overwrite any existing file.

**_write_default_conv**()
Writes the default convolution matrix, if needed.

**_write_default_psf**()
Writes the default psf file, if needed.

**_clean_workdir**()
Removes the config/param files related to this instance, to allow for a fresh restart. Files related to specific images are not removed.

**_write_assoc**(*cat*, *xname*, *yname*, *imgname*)
Writes a plain text file which can be used as sextractor input for the ASSOC identification. And "index" for each source is generated, it gets used to identify galaxies.

**__module__ = 'sewpy.sewpy'**

**_add_prefix**(*table*, *prefix*)
Modifies the column names of a table by prepending the prefix *in place*. Skips the VECTOR_ASSOC stuff !

**__call__**(*imgfilepath*, *imgname=None*, *assoc_cat=None*, *assoc_xname='x'*, *assoc_yname='y'*, *returncat=True*, *prefix=''*, *writelog=True*)
Runs SExtractor on a given image.

**Parameters**

- **imgfilepath** – Path to the input FITS image I should run on
- **assoc_cat** – optional input catalog (astropy table), if you want to use the ASSOC helper
- **assoc_xname** – x coordinate name I should use in the ASSOC helper
- **assoc_yname** – idem
- **returncat** – by default I read the SExtractor output catalog and return it as an astropy table. If set to False, I do not attempt to read it.
- **prefix** (*string*) – will be prepended to the column names of the astropy table that I return
- **writelog** – if True I save the sextractor command line input and output into a dedicated log file in the workdir.

**Returns**

a dict containing the keys:

- **catfilepath**: the path to the sextractor output catalog file
- **table**: the astropy table of the output catalog (if returncat was not set to False)
- **workdir**: the path to the workdir (all my internal files are there)
- **logfilepath**: the path to the SExtractor log file (in the workdir)

Everything related to this particular image stays within this method, the SExtractor instance (in particular config) is not modified !

**fullparamtxt = "\n#NUMBER Running object number \n#EXT_NUMBER FITS extension number \n#FLUX_ISO Iso**

**fullparamlist = ['NUMBER', 'EXT_NUMBER', 'FLUX_ISO', 'FLUXERR_ISO', 'MAG_ISO', 'MAGERR_ISO', 'F**

## How to generate this documentation

This documentation is generated using http://sphinx-doc.org/ .

To build these pages, go into the `sewpy/sphinx` directory, and run `make html` (no need for apidoc). The result is in `build/html/index.html`.

CHAPTER 2

# Indices and tables

- genindex
- modindex
- search

Last build of this documentation : Sep 12, 2017.

# Python Module Index

## S

# Index

## Symbols



## F



## G



## S