Note for Yen-Ting's COSMOS Galaxy Example

by Song Huang; 2016-05-20

Basic information:

- According to here: http://www.stsci.edu/~koekemoe/cosmos/current/
  - The ACS mosaic has resolution of 0.03"/pixel
  - "ensure that the gap between the chips was covered by at least 3 exposures in all cases"; And each exposure is 507 sec. We will just assume a 1500 sec exposure time for the mosaic.
- I renamed the image to `cosmos_1.fits`; and the PSF image to `cosmos_1_psf.fits`, just to make it more systematic.

"Correct" the exposure time of the image:

- Commands:

```
python correctExpTime.py cosmos_1.fits 1500.0
```

Correct the exposure time using: `correctExpTime.py`:

- The Python code: `correctExpTime.py` that can help you do this easily.
- Need to install Python and the `Astropy` library.
  - Under MacOSX, please take a look at the `AstroConda` environment: http://astroconda.readthedocs.io/en/latest/index.html
  - `homebrew` can be used to install Python; and `pip` can be used to install, update the `Astropy` library.
  - Will create `cosmos_1_cor.fits`
  - Should check the header to make sure the keyword is correct.
  - **This process can be automated easily**

Create mask image:

- Commands:

```
./run_sex.sh cosmos_1
python seg2Mask.py cosmos_1_seg.fits -s 2.0 -t 0.02
```

Simple `SExtractor` run using `run_sex.sh`

- When `SExtractor` is appropriately installed, you can use the `run_sex.sh` script to run `SExtractor` to create an object catalog, and a segmentation image.

- The default configuration file is `shuang.sex`. You need to play around a few key configuration parameters to make it works the best for your data. The most important ones are probably: `DETECT_MINAREA`, `DETECT_THRESH`, `ANALYSIS_THRESH`, `DEBLEND_NTHRESH`, `DEBLEND_MINCONT`, `CLEAN_PARAM`, `SEEING_FWHM`, `BACK_TYPE`, `BACK_SIZE`, and `BACK_FILTERSIZE`; Please refer to the `SExtractor` manual for the detailed meaning of these objects.
- **This process can be automated too; it is easy to call SExtractor from Python, and there is Python library, SEP, that can achieve most core functions of SExtractor.**

Convert the segmentation image into object mask using `seg2Mask.py`:

- The `seg2Mask.py` script can help you make a mask image using the segementation image (assuming the target is at the center).
- This depends on the `Astropy`, `numpy` and `scipy` packages (all can be installed using `pip`).
- This will create mask files: `cosmos_1_seg_mask.fits`
  - `-s 2.0`: Sigma of the Gaussian kernel in unit of pixel; Larger kernel will grow the mask more aggressively.
  - `-t 0.02`: Lower value cut to the convolved the mask image; The mask image has highest pixel value = 1.0, and after the convolution, any pixel with value lower than 0.02 will be excluded from the mask.

Run GALFIT

- A few tools are available for running GALFIT on images:
  - `galfitSimple.py`: The main program:
    - Can be integrated into other Python code; or used as a Bash script.
    - Depends on `Atropy` package.
    - [Optional] Use the `cubehelix` and `palletable` package for better colormap than the `Matplotlib` ones.
    - Depends on local code: `songUtils.py` for a few useful functions.
    - Depends on local code: `galfitParser.py` for parsing the GALFIT result.
  - `galfitLog2Input.py`: In case you want to run more tests yourself, this can help you rename the GALFIT log file (the one named like: `galfit.xxx`) into a GALFIT readin file.

Example: Fit `cosmos_1` with 1 Sersic model + 4th Fourier mode; without background

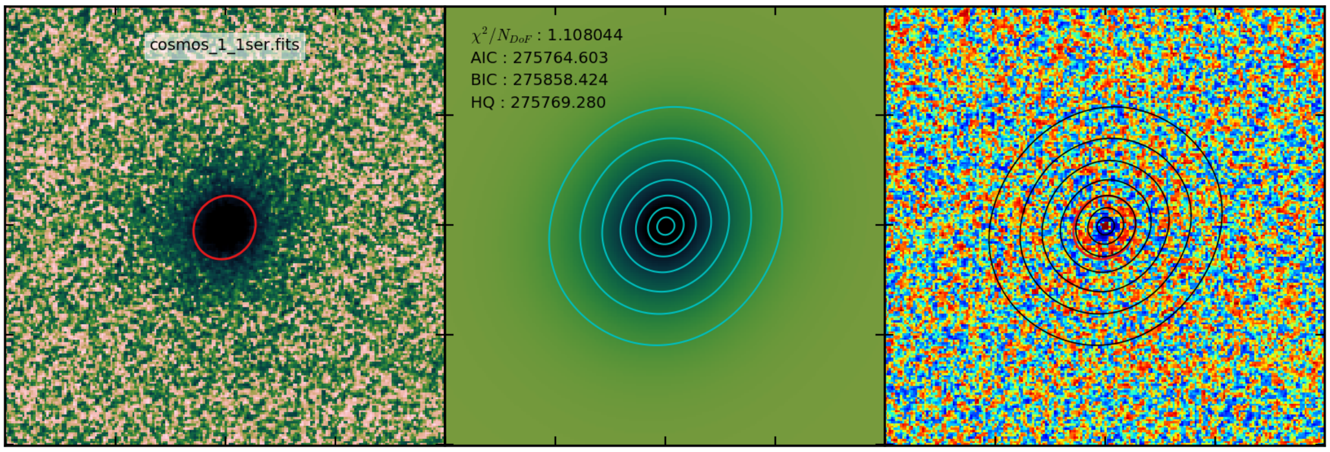
```
python galfitSimple.py cosmos_1 -t cor -m seg_msk -p 0.03 -z 27.0 --run1 --f4

python galfitSimple.py cosmos_1 -t cor -m seg_msk -p 0.03 -z 27.0 --run1 --2ser --r
un2 --f4 --bkg 0.0 --skyGrad
```

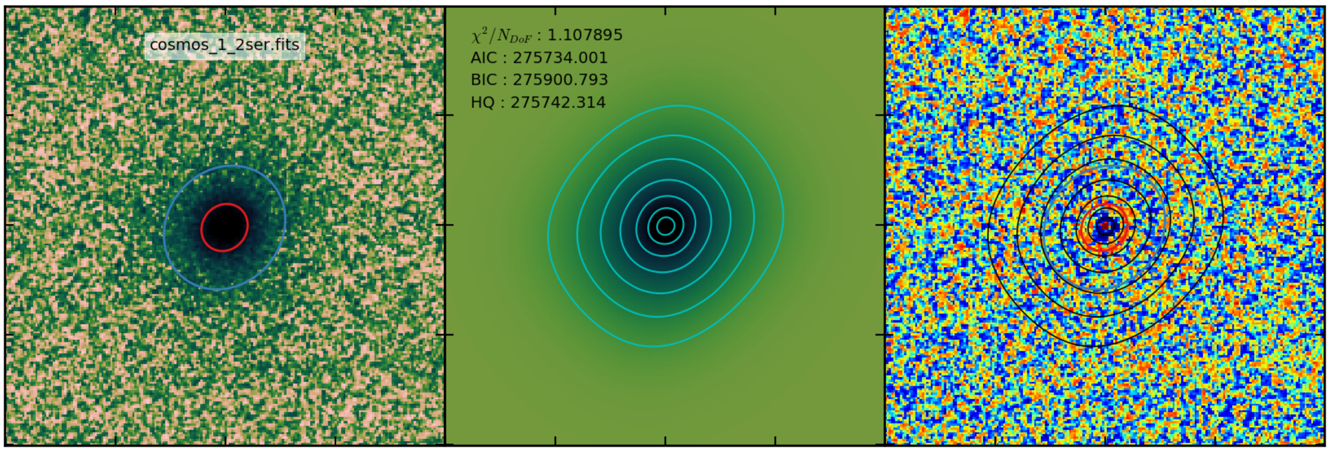
- Details:
  - The prefix of the image is always necessary: `cosmos_1`
    - `-t cor`: this is the "image type", basically the suffix of the image file, it will search for

    `cosmos_1_cor.fits` as input image.
        - `-m seg_msk`: this is the "mask type", basically the suffix of the mask file, it will search for `cosmos_1_seg_msk.fits`.
        - By default, the PSF image will be used, and it should be: `cosmos_1_psf.fits`; Just in case you don't want to use the PSF convolution during fitting, a `--noPsf` option is available.
        - By default, no external sigma image is used; But by adding the `--useSig` option, the code will search for `cosmos_1_sig.fits` as input sigma image.
    - `-p 0.03`: The pixel scale is 0.03 arcsec per pixel
    - `-z 27.0`: The zeropoint of the image (**not the accurate one for COSMOS!**)
    - `--run1`: Will actually run GALFIT to get the result, without this, the code will just generate the GALFIT input file.
        - In case you want to run 2- or 3-Sersic models, we have options of `--2ser`, `--run2`, `--3ser`, `--run3`. Their meanings should be clear.
    - `--f4` can be used to add 4th Fourier mode (to each component of the model); In case you want to fit the lopsidedness of the model, the 1st Fourier mode can be added too with the `--f1` option.
    - If the galaxy is at the center of the image, the default central position is the center of the image, the default magnitude is 21.0 mag, effective radius is 5.0 pixel, and the Sersic index is 1.5; The default axis ratio is 0.9, and position angle is 0.0 degree.
        - `--galX0 252.0 --galY0 252.0`: Can be used to set the center of the galaxy
        - `--galRe 2.5 --galSer 2.0`: Can be used to set the initial guess of effective radius (in unit of pixel) and the Sersic index.
        - `--galQ0 0.85 --galPA0 10.0`: Can be used to set the initial guess of the axis ratio and position angle.
    - **Right now, the whole image will be used for the fitting; And the convolution box is 0.9x the size of the image. More flexibility will be added to make it more efficient.**
    - `--bkg 0.00 --skyGrad`: These two options can be used to add a Sky component to the model, where the value is set by `--bkg`; And `--skyGrad` will allow the Sky component has gradient.

- Output files:
    - Output model FITS file: `cosmos_1_1ser.fits`:
        - Has 4 extentions; HDU=1 is the original image, HDU=2 is the model image, and HDU=3 is the residual image; All the parameters from the model is kept in the header of the HDU=2.
    - Updated input file: `galfit.xx` --> `cosmos_1_1ser.in`.
        - If the model fitting runs successfully, the GALFIT log file will be used to update the input file.
        - The original input file will be backuped as `cosmos_1_1ser.in_back`
    - The log of the fitting process: `cosmos_1_1ser.in.log`:
        - This is the shell output of the fitting process. It is useful to diagnose problem in the fitting.
    - The Pickle file of the results: `cosmos_1_1ser.pkl`:
        - The model parameters are parsed and saved in a Python Pickle file. (More details later)
    - Summary figure of the model: `cosmos_1_1ser.png`:
        - Left: Data (name of the model; circle that highlights the effective radius of each component)
        - Middle: Model (reduced chi^2 and other statistics of the model; contour of the model)

        - Right: Residual image (with high contrast; with both contour of the model and the effective radius of each component on top)
        - Examples of these figures are shown below. (Although the 2-Sersic model is reasonable, it does not show improvement compared with 1-Sersic model).
    - GALFIT input file based on the fitting result but with the PSF file removed: `cosmos_1_1ser_nopsf.in`
        - This is useful to generate a model image without PSF convolution.


cosmos_1_1ser.fits
χ^2/N_{DoF} : 1.108044
AIC : 275764.603
BIC : 275858.424
HQ : 275769.280


cosmos_1_2ser.fits
χ^2/N_{DoF} : 1.107895
AIC : 275734.001
BIC : 275900.793
HQ : 275742.314