Fit SDSS Galaxy Using GALFIT

by Song Huang 2016/05/20

- Make sure you read through the GALFIT website, and run the example:
  - https://users.obs.carnegiescience.edu/peng/work/galfit/galfit.html
  - Especially the "Top 10 rules of Thumb"
- For SDSS image:
  - Be aware that there is background over-subtraction issue for nearby, bright galaxies.
  - Be aware that the i-band has strong "red-halo" effect in its PSF. So, do not trust flux at the outskirt of a bright object, or the color at low S/N
  - Generally speaking, the g-, r-, i-bands are useful for image fitting; r-band is probably the best; u- and z-band are often two shallow, or have other issues.
  - For SDSS image, no solid **sigma image** is available, so it is better correct the SDSS images for the exposure time; Basically, multiply the SDSS image with an appropriate exposure time; Normally, **60** secs works fine.
- Photometric unit:
  - SDSS images are often in unit of "nanomaggies", 1 nanomaggy == 3.631×10^{-6} Jy; Hence can be easily converted into AB magnitude assuming:
    - mag_ab = 22.5 - 2.5 * log10(flux_naomaggies)
  - SDSS catalogs use "asinh magnitude" system
    - See details here: https://www.sdss3.org/dr8/algorithms/magnitudes.php
- **DO NOT** forget to correct the Galactic extinction!

Download SDSS images:

1. SDSS Science Archive Server:
   - For single Frame: http://dr12.sdss3.org./fields
   - In case of large galaxy, use the mosaic: http://dr12.sdss3.org./mosaics
   - Can also use the Explore tool in skyserver to search for images and other information: http://skyserver.sdss.org/dr12/en/tools/explore
     - FITS image can be downloaded from the "FITS" link under "Imaging Summary"
   - **Image normalized to 1 sec exposure time; In unit of "nanomaggies"**
   - **Should be no SOFTBIAS anymore (used to be)**
2. Montage: Image Mosaic Service at IPAC:
   - Web service here: http://hachi.ipac.caltech.edu:8080/montage
   - Create an account; Use the RA, DEC to create mosaic; Pixel resolution should be 0.4 arcsec.

- **Pro**: Use all frames available in SDSS; better background subtraction;
- **Con**: Can not do bulk search
- **Image normalized to 1 sec exposure time; In unit of "nanomaggies"**

3. Data from NASA-Slaon Atlas:
   - Data available here: http://nsatlas.org/
   - The catalog is available here: http://sdss.physics.nyu.edu/mblanton/v0/nsa_v0_1_2.fits
   - **Image normalized to 1 sec exposure time; In unit of "nanomaggies"**
   - **Pro**: Well organized; PSF image available
   - **Con**: Only a small sample of nearby galaxies are available; Sometimes the cutout is not large enough

Download NSA data in batch mode: `hs_down_nsa_data.pro`**:**

```
* Under the `sdss_galfit` folder, the IDL code `hs_down_nsa_data.pro` is useful fo
r downloading NSA data (images and PSF in g, r, i band, and a preview JPEG image).

* Take catalog in the format of NSA catalog as input:
``` IDL
hs_down_nsa_data, "input_nsa.fits", loc_atlas='atlas', loc_data='nsa_data'
```
* Depends on `wget` for downloading data.
```

PSF Images:

1. For galaxy with available NSA data, just use the `bpsf` FITS file provided by NSA.
2. For large mosaic created by Montage or other mosaic tools, it is better to do the PSF modelling by yourself using `SExtractor` + `PSFEx`
   - More details later...
   - On MacOSX, `SExtractor` is available through `homebrew`; `PSFEx` is slightly more difficult to install
3. For most of the fitting, PSF model constructed for single frame SDSS image is good enough. One can download the `psField-xxxx.fit` file, and reconstruct the PSF images.
   - You need to know the `RUN`, `RERUN`, `CAMCOL`, `FIELD` parameters to download the `psField-xxxx.fit` file; And use the `readAtlas` tool, and the `ROW`, `COL` parameter to reconstruct the PSF image.
   - All the above information is available in SDSS catalogs; and can be found on the Explorer tool webpage; or queried through the SkyServer.
   - The `psField-xxxx.fit` file is stored here in the format of : "http://data.sdss3.org/sas/dr12/boss/photo/redux/301/$run/objcs/$camcol/psField-00$run-$camcol-$field.fit"
   - About how to reconstruct the PSF image, read this: https://www.sdss3.org/dr8/imaging/images.php

Download `psField-xxx.fit` files in batch mode: `down_psfield.sh`

```
- Under `sdss_galfit` folder, the `down_psfield.sh` tool can help you download the
`psField-xxxx.fit` files in batch modes, and put the commands to reconstruct the PS
F images into a file called `temp`
- Usage:

    ``` bash
    ./down_psfield.sh input.csv
    ```

    * input.csv should be something that looks like this:

    ``` bash
    # ID,run,rerun,camcol,field,rowc,colc
    halphablob,2566,301,6,44,1304.261,1683.937
    ```
```

Example of GALFIT procedures with examples:

1. Example 1: Nearby S0 galaxy: **ID=37434** in g-band (Bright, isolated galaxy)
    - Image (From NSA): `37434_g.fits`
    - PSF (From NSA): `37434_g_psf.fits`
2. Example 2: Faint MaNGA galaxy with interesting Halpha extension in r-band (Small galaxy contaminated by big star)
    - Image (From Montage): `haBlob_r.fits`
    - PSF (From NSA): `haBlob_r_psf.fits`

"Correct" the exposure time

- The images for both examples are normalized to flux unit (1 sec exposure time). To make sure that GALFIT can internally calculate meaningful sigma image, we need to correct the exposure time, assuming the typical SDSS exposure time: ~60 sec.
- One simply needs to read in the image as an array; multiply it by 60.0; Change the `EXPTIME` keyword in the header to 60.0 (in case there is no such keyword, create a new one); save the updated array to a new fits file with the correct header.
- Under IDL, something like this should work: `IDL exptime = 60.0 ; sec image = mrdfits('xxx.fits', 0, header) image *= exptime sxaddpar, header, 'EXPTIME', exptime mwrfits, image, 'xxx_cor.fits', header=header`

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

- Under `sdss_galfit`, there is a 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.

- Usage: `bash python correctExpTime.py 37434_g.fits 60.0 python correctExpTime.py haBlob_r.fits 60.0`
  - Will create `37434_g_cor.fits` and `haBlob_r.fits`
  - Should check the header to make sure the keyword is correct.

Select the region to fit, and the convolution box size

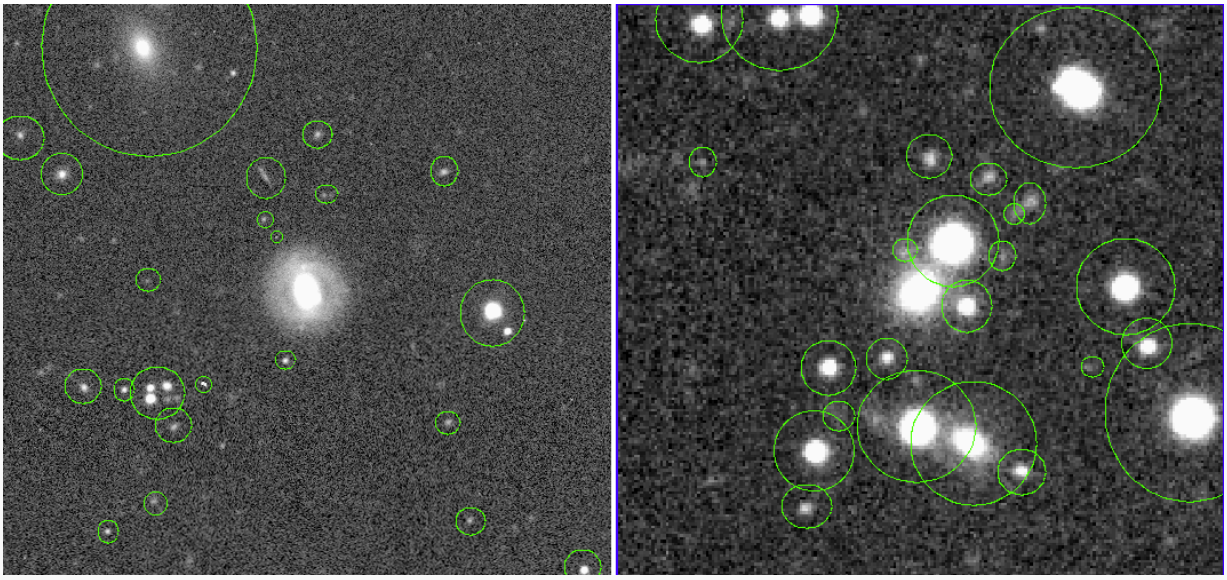
- Now we need to define a box region around the galaxy for the fitting.
- We want the fitting region to be large enough to cover the whole galaxy and the enough sky area; But we do not want it to be too big for the galaxy, so `GALFIT` will not waste time fitting the sky area.
- Generally speaking, box size > 10 x Re should be good enough (unless you have super deep image).
- The following figure show the appropriate box size for the two galaxies (`37434` at the left; `haBlob` at right side; Notice that the bright star around `haBlob`).
  - `37434`: 150 490 150 490 (xmin xmax ymin ymax)
  - `haBlob`: 150 300 150 300
- For SDSS image, the PSF is well-behaved and not very large, normally we do not need to convolve the entire image if we just want to fit one galaxy. For both of the cases, **100 pixels** are large enough for appropriate convolution.

```
![sdss_galfit_1](media/14636644312465/sdss_galfit_1.png)
```

Object mask:

- We need to create a mask image for GALFIT, so that pixels belong to other objects will be ignored by `GALFIT`.
  - For `37434`, the galaxy is very isolated, hence we can probably fit the galaxy well without any mask; but, in principle, a mask is still better.
  - For `haBlob`, on the other hand, good object mask is key to good fit as there are many bright contaminations nearby.
- It is never an easy task to create appropriate masks, you want to mask out all pixels belong to contaminations, but that is often impossible. In case of very nearby galaxy, or bright contaminations, one should iterative fitting process (fit the brightest contamination, remove the model from the image; iterate this process until the area around the real target is clean) or fit the object and contamination together.
- There are two simple approaches to create object masks:

1. Manual masks using `ds9` regions:
   - Manually create a bunch of `ds9` regions in whatever shape you find useful; Save the region file using `IMAGE` coordinate; Convert the mask into a fits image using Python or IDL script.
   - If you have a small sample, or you want to mask out everything carefully, this is a better approach. Example regions for `37434` and `haBlob` are shown here:
   - Save them into `37434_g.reg` and `haBlob_r.reg`; Make sure to select the `Coordinate System` as `image`; and `Format` as `ds9`.



`DS9` **region to mask using IDL:** `region_to_mask.fits`

- Under `sdss_galfit`, the `region_to_mask.fits` can help you convert the region file into a FITS file
- Usage:

```
region_to_mask, 'haBlob_r.fits', 'haBlob_r.reg', output='haBlob_r_mask.fits', savejpeg='haBlob_r_mask.jpg'
```

- This will create the mask FITS image, `haBlob_r_mask.fits`; and a simple JPEG image to show you the coverage of the mask.

`DS9` **region to mask using Python:** `ds9Reg2Mask.py`

- Under `sdss_galfit`, the `ds9Reg2Mask.py` can help you convert the region file into a FITS file.
- It depends on the `Astropy`, `pyregion`, and the `cubehelix` color scheme. `pyregion` can be installed through `pip install pyregion`; The latter can be installed using the `pip` tool: `pip install git+git://github.com/jradavenport/cubehelix.git`
- Usage:

```
python ds9Reg2Mask.py 37434_g.fits 37434_g.reg -m 37434_g_mask.fits -s
```

- This will create mask FITS image, `37434_g_mask.fits`; and a much better visualization of the mask as PNG file: `37434_g_mask.png`

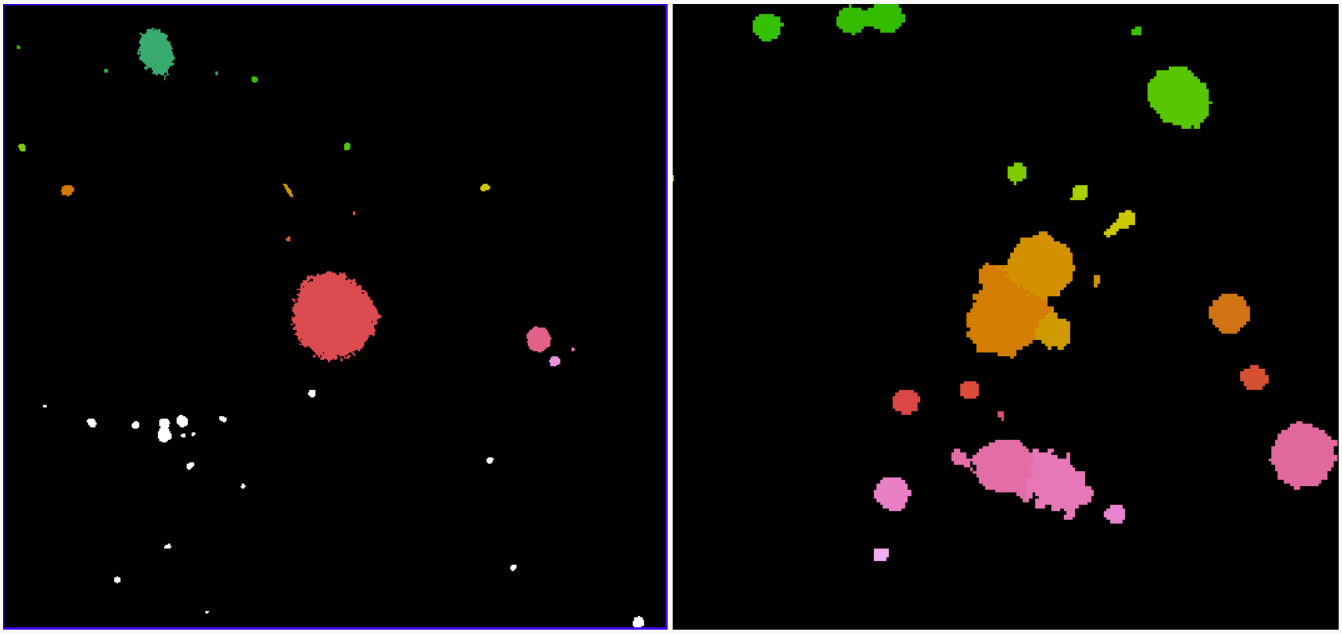
1. Create object masks in batch mode using `SExtractor`:
    - In many cases, one needs to create object masks for large sample of images. For simple situations, one can use the `SExtractor` tool to detect all objects on the image; If we can assume the object at the center is the target to be fit, we can remove it from the segmentation image; Convolve the segmentation image using a Gaussian kernel to "grow" the object mask a little bit to create an appropriate object mask.
    - One has to be careful with the detection threshold and the deblending related parameters for `SExtractor` to create appropriate object mask.

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.
- Usage:

```
./run_sex.sh 37434_g_cor
./run_sex.sh haBlob_r_cor
```

```
- This will create segmentation files: `37434_g_cor_seg.fits` and `haBlob_r_cor_seg.fits` (as shown below); and object catalogs: `37434_g_cor_cat.fits` and `haBlob_r_cor_cat.fits`.
```



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

- Under `sdss_galfit`, 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`).
- Usage:

```
python seg2Mask.py 37434_g_cor_seg.fits -s 2.0 -t 0.02
python seg2Mask.py haBlob_r_cor_seg.fits -s 2.0 -t 0.02
```

```
- This will create mask files: `37434_g_cor_seg_mask.fits` and `haBlob_r_cor_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.
```

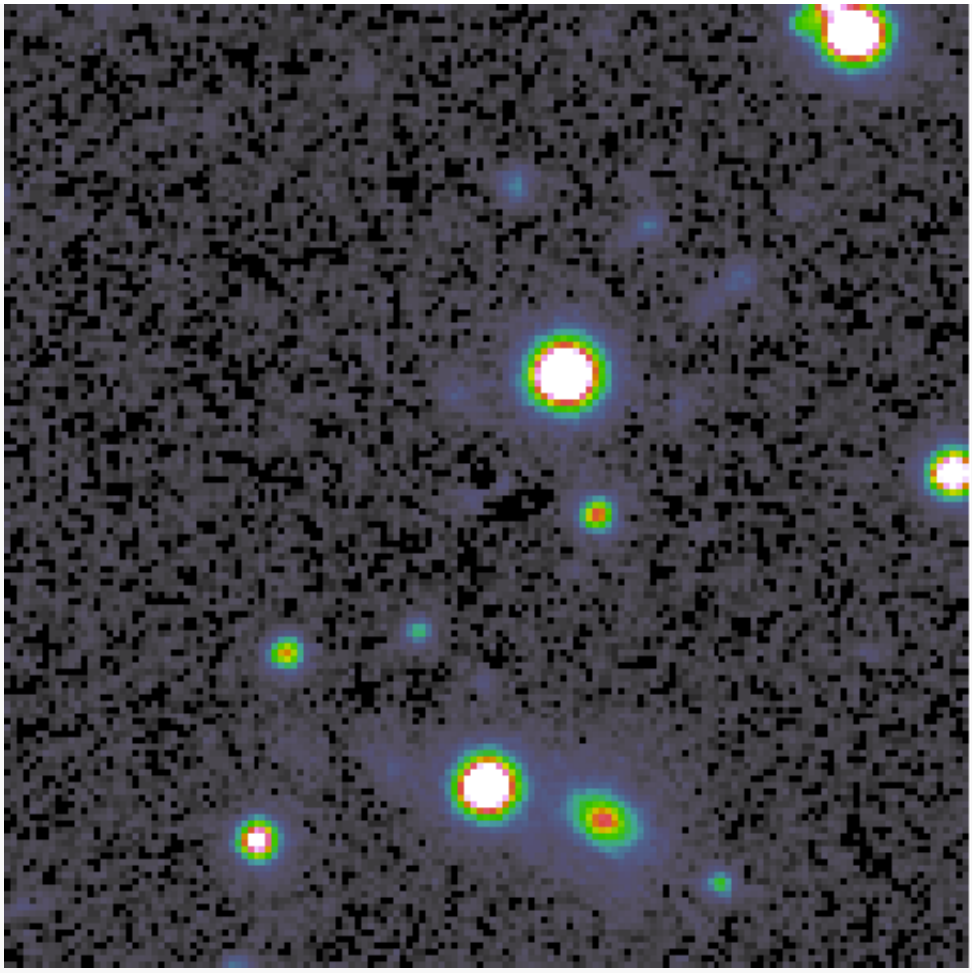
Choice of models and initial guesses of parameters:

- Model choice: **Basic principle -- Start with simple model (e.g. 1 Sersic), gradually build-up complexity of the model; Always check the residual image**
    - For `37434`, Sersic bulge + n=1 Disk model can be a reasonable start; But as the galaxy has a strong bar, and potentially some oval (or lens) structure, the residual is not going to pretty.
    - For `haBlob`, it seems that there is no sub-structure in the galaxy, and the galaxy itself is small enough, single Sersic component should be good enough as start.
    - Additional `sky` component can be included even when the sky is subtracted well enough; Can use it to fit any potential gradient in the background.
    - In case of multiple components in the model, one can "soft" constrain the central coordinates of the two components to be the same using constraint files.
        - The `2comp.cons` and `3comp.cons` files under `sdss_galfit` are example constraint files for 2 and 3 components.
- Initial guesses: **Always starts from fainter magnitude, smaller size, and lower Sersic index**
- The `GALFIT` run should not strongly depends on the initial guess; But when the choice of model can **never** provide a good fit to the data, the result can be very different with different initial guesses (basically...no clear minimum on the chi^2 surface)

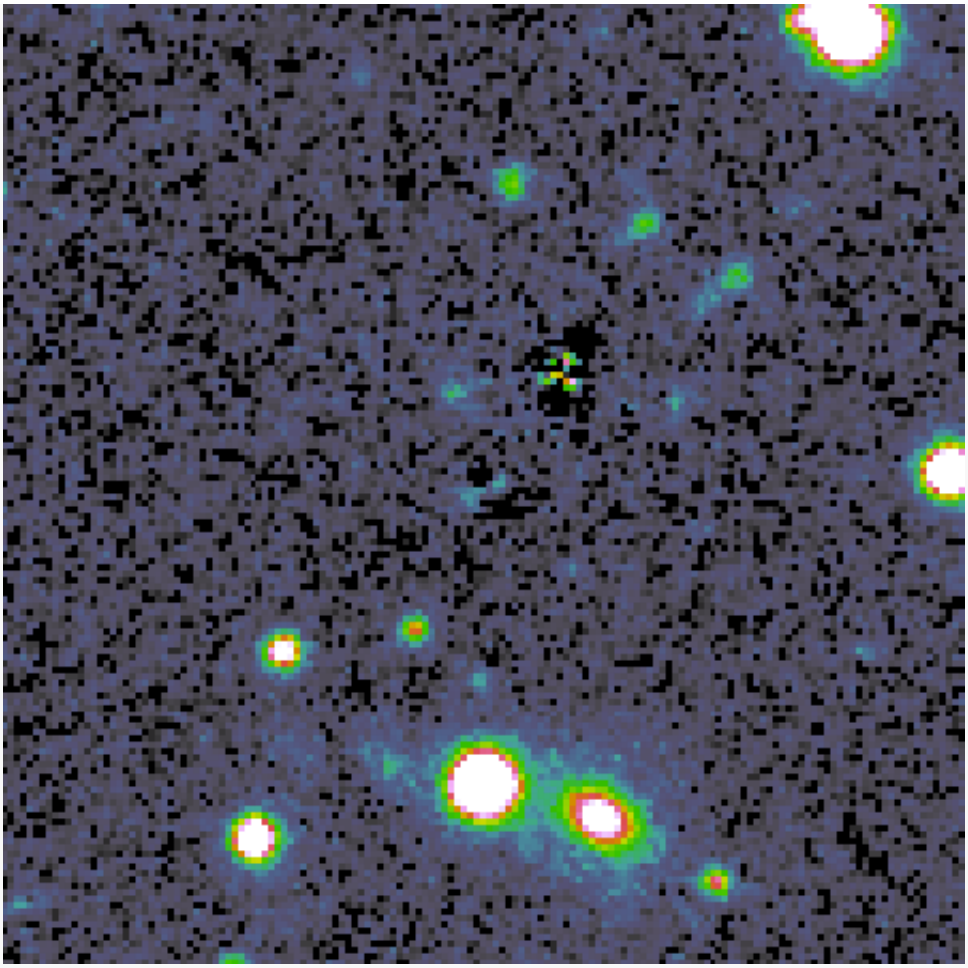
GALFIT workflow:

---

1. `haBlob_r`
    - 1 Sersic model + sky: `haBlob_r_1ser.in`
        - Command: `galfit haBlob_r_1ser.in`
        - Output models: `haBlob_r_1ser.fits`
        - Results: Re=6.51 pix; nSer=3.69; b/a=0.85
            - At z=0.038, 1"=0.608 Kpc; Re=2.58" and 1.57 Kpc
        - The residual image looks like:



- 1 Sersic model + sky; Also fit the nearby contaminations together: `haBlob_r_1ser_b.in`
  - Using a new region file that exclude the nearby bright star from the mask: `haBlob_r_b.reg`
  - Create new mask files using: `python ds9Reg2Mask.py haBlob_r_cor.fits haBlob_r_b.reg -m haBlob_r_mask_b.fits -s`
  - Command: `galfit haBlob_r_1ser_b.in`
  - Output models: `haBlob_r_1ser_b.fits`
  - Results: Re=6.70 pix; nSer=3.69; b/a=0.83 At z=0.038, Re=2.65" and 1.61 Kpc
  - The residual image looks like:



- The single Sersic model is pretty stable, and the result is very reasonable.

---

1. `37434_g`
    - 2 Sersic model + sky: `37434_g_2ser.in`
        - 2 component constraint file, `2comp.cons` is used
        - The Sersic index of the second component is fixed at 1.0 (equals to Exponential disk)
        - Command: `galfit 37434_g_2ser.in`
        - Output models: `37434_g_2ser.fits`
        - Results:

```
# comp    :    cen_X      cen_Y      mag*   re_pixel  sersic   b_a      PA
sersic    : (  320.41,    320.67)   17.41      1.22    3.33    0.25     8.29
sersic    : ( {320.41}, {320.67})   17.19     23.57   [1.00]   0.89   -26.37
```

```
- **NOTICE**: The magnitude here is derived assuming zeropoint=24.0; Conver
t the magnitude back into flux in unit of nanomaggies using this zeropoin
t, then convert it into SDSS AB magnitude (using either log or asinh magni
tude).
- The residual image is not crazy, but also not great;  Clearly the inner
region of galaxy is dusty, and has some complicated sub-structures.
- The "bulge" has a pretty high Sersic index (3.33), but very small axis ra
tio (0.25), it is very likely that this is due to confusion/degeneracy be
tween a bulge and a bar (or a lens) structure.

![sdss_galfit_6](media/14636644312465/sdss_galfit_6.png)
```

- 3 Sersic model + sky: `37434_g_3ser.in`
  - 3 component constraint file, `3comp.cons` is used
  - The Sersic index of the third component is fixed at 1.0 (equals to Exponential disk)
  - The 4th Fourier modes are added to the first two components to increase their flexibility in fitting local features (Please read GALFIT 3.0 paper by Peng et al. 2010 for more details)
  - Command: `galfit 37434_g_3ser.in`
  - Output models: `37434_g_3ser.fits`
  - Results:

```
# comp    :    cen_X     cen_Y      mag*   re_pixel  sersic   b_a      PA
sersic    : (  320.40,   320.66)   18.09      0.85   *0.05*   1.00  -107.96
    fourier : (4:  0.17,  -33.43)
sersic    : ( {320.40}, {320.66})  18.18      2.91    2.72   *0.08*    7.73
    fourier : (4:  0.23,   20.43)
sersic    : ( {320.40}, {320.66})  17.22     23.14   [1.00]   0.90   -41.34
```

```
- As you can see, the disk component is pretty stable; The inner two compon
ents start to have problematic parameters; and the residual does not show
visible improvement.
- Clearly, the complex substructures and dust features at the inner region
make it intrinsically difficult to fit the inner region well.
```

- 3 Sersic model (all free Sersic index) + sky: `37434_g_3ser_b.in`
  - 3 component constraint file, `3comp.cons` is used
  - The Sersic index of the third component is set free (Many disk of S0 galaxies are not exponential)
  - Command: `galfit -imax 200 37434_g_3ser_b.in`
    - For complicated model, `GALFIT` often takes more than 100 iteration to get to the best result. `-imax 200` increases the maximum number of iterations to be 200.
  - Output models: `37434_g_3ser_b.fits`
  - Results:

```
# comp    :    cen_X     cen_Y      mag*   re_pixel  sersic   b_a      PA
sersic    : (  320.40,   320.67)   17.62      1.53   *0.03*   0.46     7.04
sersic    : ( {320.40}, {320.67})  19.15      8.64   *0.04*   0.13     8.69
sersic    : ( {320.40}, {320.67})  17.32     21.91    0.64    0.91   -31.96
```

- As you can see, the parameters **for** the central two components are still n
ot reliable (problematic); However, the residual **at** the outskirt does look
slightly better, and the Sersic index of the disk becomes smaller than 1.0,
 which often means that the disk is "truncated" (**normal** behaviour).
- Although there are intrinsically difficulties in fitting the inner region
 very well, the total flux, size, and shape of the disk are quite stable, w
hich can be used **as** a baseline; You can still use the flux in the **first** t
wo components **as** the total flux of the "bulge", and calculate the bulge-**to**-
disk ratio.