Forward Modeling Code

Model gravitationally-lensed objects in the source plane!

Code written by Michael Florian, and Keren Sharon,

This document created by Alex Navarre

Table of Contents

How to Use the Code

Note from Alex - I used the files named "sersic_run_mcmc5.py", "clump_forward_modeling.py", and "realSourceReconstructions_4.py". There may be newer versions available. Ask Michael Florian.

Required Files

- alpha maps (aka deflection maps). One map (fits file) for displacement in each x and y.
    - These tell us how to move the light of our objects between the image and source planes.
    - E.g. dplx_4.1422.fits, dply_4.1422.fits
- The science data you are interested in, cropped and aligned to the alpha maps. See my function "align" in Section 7.
    - E.g. 1152_F110W_aligned.fits
- A PSF model. If there are stars in your field, you can model your PSF from your (uncropped) science data. See "PythonPSF.py" in Section 6.
    - E.g. 1152_f110w_Empirical_largePSF.fits
- The input text file.
    - See below for an exhaustive review of the parameters.
    - E.g. 1152_clump_input.par

The Input File

This file contains general run information and clump model parameters.

- Deflect - Two values. File paths to your alpha/deflection maps
- Ps - The plate scale of your science image (arcsec/pixel)
- Rms - The root-mean-squared of the sky background distribution surrounding the object of interest.
- Kernel - File path to the PSF
- Data - File path to the aligned science image
- Aperture - File path to the aperture file.
- Offset - Two values. Honestly ???. I've only ever had them be 0 0.
- Resolution - How many times finer you want your source plane grid to be simulated. Try 10x.
    - If a 10x finer pixel scale corresponds to a physical pixel size too large to accurately model your object, try increasing resolution to

~magnification value. Note, this significantly increases computation time.

- Img_crop - Four values. Read out from processed source plane header.
- Src_crop - Four values. Read out from processed source plane header.
- Nthreads - Number of cores to assign for multi-threading. I've found that 1 or 2 cores works best, and adding more actually slows down completion time.
- Burnin - The number of "burn in" iterations in the MCMC chain. 50 has always been fine for me.
    - Here is a reference for how the MCMC (Monte Carlo Markov Chain) sampling works.
- Sample - The number of iterations MCMC will run for.
    - 5000 is a good lower baseline to sample the parameter distributions. If your project doesn't take forever to run, I like to do 20,000.

Clump Parameters

A sequence of clump parameters and parameter limits. Note that there should be no spaces between the clump code block and the limit code block. Neither should there be a space between model code blocks if you are simulating multiple models.

In the limit section, the 1 (or 0) after the parameter name determines if the parameter should be free during this run or not.

Example clump 1 profile sersic x 285.29 y -153.35 flux -1.75 size 0.135 e 0.836 pa 174.3 n 4.0 end limit 1 x 1 284.30 286.30 y 1 -154.35 -152.35 flux 1 -2 -1	X, Y - positions in the source plane. Calculate by: $x_{source} = \frac{x_{ds9}}{resolution} + srccrop[0]$ $y_{source} = \frac{y_{ds9}}{resolution} + srccrop[2]$ Where x_ds9 and y_ds9 are the centroid of the distribution in the source plane fits file. Src_crop[0] and src_crop[2] are also called XFOOT1 and YFOOT1 in the source plane header.

size 1 0.05 0.5 e 1 0.0 1.0 pa 1 0 180 n 1 0.5 7.5 end clump 2 profile gaussian x 285.30 y -153.35 flux -2.75 size 0.5 e 0.95 pa 174.3 end limit 2 x 1 282.30 286.30 y 1 -156.35 -152.35 flux 1 -6 -1.5 size 1 0.05 10.0 e 1 0.2 1.0 pa 1 0 180 end	Flux - A measure of how bright the distribution is. See Sersic2D and Gaussian2D to see how they are formulated. Size - A measure of how extended the distrbution is. See Sersic2D and Gaussian2D to see how they are formulated. E - The ellipticity of the model. A value of 0 is perfectly circular and a value of 1 is infinitely elliptical. PA - The position angle. How is your model rotated with respect to the data axis? N - Sersic index. Only for Sersic models. Is degenerate with size. See Sersic2D to see how it is formulated.

Steps

1. In all of your code files, make sure your paths are correctly set up so that they point to the correct files.
2. With your aligned science image, create a square (or rectangular) region in ds9 that encompasses the area you want to delens to the source plane.
    a. If you have the critical curve for your field, make sure your region does not contain it. The code currently does not handle that well.
    b. This area can contain other objects, we will handle that in the next step.
    c. You can have multiple square regions at once for multiple images if you wish.
    d. Save the region file with the square region(s) that you made, and double check that nothing else is contained in the region file from misclicks or unwanted regions.
3. For each square reigon that you made, isolate it. If there are other sources of light within the region, convert the square region into a polygon and exclude

them. Using my code block “maskfromregion” (see Section 7), create a fits file where the data is 1s inside the region and 0s outside. This is our aperture.
    a. Save the fits file and write the path to it under aperture in the input file.
4. Run realSourceReconstruction to create source-plane models on the finer-scale grid.
    a. Note you have to manually change the ps (plate scale), ps_s (1/resolution, NOT ps/resolution), and z (redshift) parameters for your object and data.
    b. This will create 2-3 files for each square region you had. They are numbered 0, 1, 2… etc based on the order of the square regions in the region file.
    c. The source plane image is the one with _drz on the end.
    d. If you have a PSF, you can supply it and turn mapPSF = True. This will create images of the image_plane PSF in the source plane at each of your locations.
5. For the clump you are going to simulate, open that source plane image and look at the header.
    a. XFOOT1, XFOOT2, YFOOT1, YFOOT2 correspond to src_crop[0] - src_crop[3] in the input file
    b. IXMIN, IXMAX, IYMIN, IYMAX correspond to img_scrop[0] - img_crop[3] in the input file.
6. Choose your model(s) you want to simulate and populate the lower part of the input file.
    a. Find valid values of x and y using the equations above.
    b. For the limits on x and y, remember that 1 pixel in ds9 corresponds to 1/resolution pixels in the source plane.
        i. So 159 to 161 would correspond to 20 source plane pixels at 10x resolution.
7. Save the input file and run sersic_run_mcmc.
    a. realSourceReconstructions creates a file with the extension “.p” (not footprint.p, the other one). You need to manually change the file string in run_sersic_mcmc to recognize the correct file string.
    b. It is set up so that you are told every 50 iterations how long those 50 iterations took to complete. Based on the size of your region you're simulating and your resolution, this will change. For a small region and 10x, this should take about 1-5 minutes per 50 iterations.

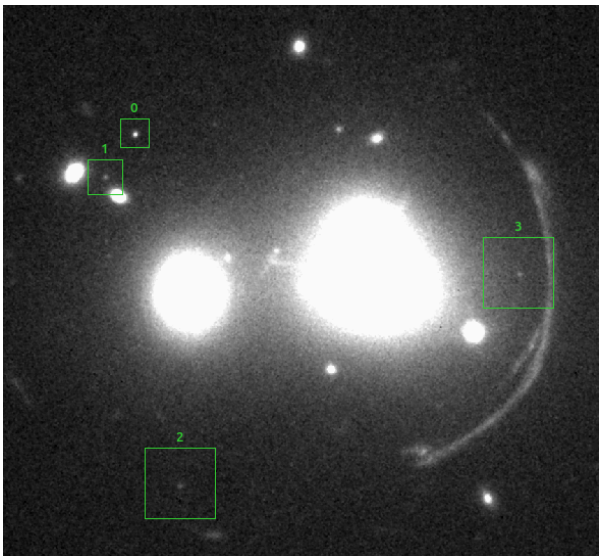
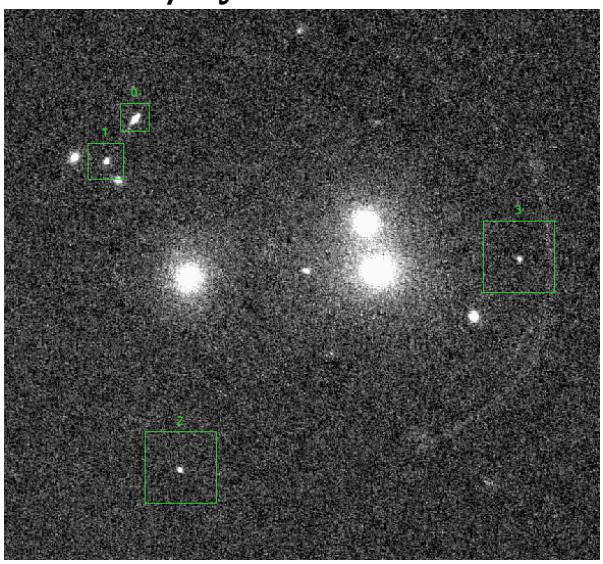
   c. When it's done you will most likely get an error: "*The chain is shorter than 50 times the integrated autocorrelation time for ___ parameter(s). Use this estimate with caution and run a longer chain!"* I find this often to be over-cautious. You should be able to either find a good model or see why you aren't getting a good model with 5000+ iterations.
8. When the run is done, use my functions "bestfitplot", "cornerplot" and "distplots" to analyze the run. If you are not satisfied with these outputs, apply any necessary changes to the input file and either 1.) start over by moving/deleting the output files and running again, or 2.) running the model for more iterations.
   a. If you choose to run the model for a longer time, make sure your version of the code allows the files to be appended. Else, you will just overwrite the old data with the new data

Notes,Tips, and Tricks

These are in no particular order

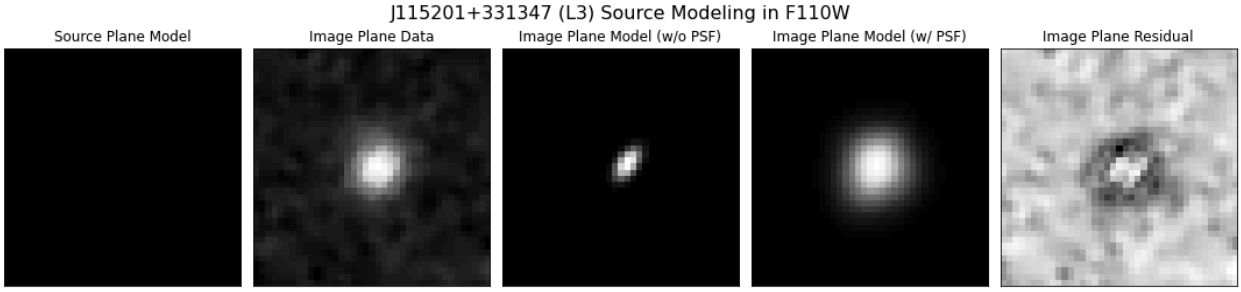
- My testing shows that the code does will if the object in the image plane is resolved. If it isn't resolved, whatever model you choose will just try to be as small and pointlike as possible. In that case, any limits on the size of the object are dependent on the PSF.
- Generally, going above 10x resolution is not worth it. At least at redshift ~4-5. If your object is unresolved, then a smaller pixel size most likely won't help. You can do a sanity check and calculate the physical size of a source plane pixel at any resolution. Do this by calculating the angle the pixel would subtend, convert it to radians, and multiply by the angular diameter distance for your object's redshift. This can be done with astropy.
- For the objects I have worked with, using multiple models for 1 clump doesn't seem to work much better than 1 model. Especially if they are allowed to be completely degenerate with each other.
    - I was working with 2 models for the Lya (one bright and compact, one faint and extended) of 1152 (see Section 3), and the jury is out if this is a good idea or not.
- Do not let your models get arbitrarily large. If this happens, they will try to be super large and faint. These "pancake" models can be interpreted by the code as good.
    - This is partly because the function that calculates goodness of fit treats each part of the fitting region equally.
    - The code only compares the residual rms to the rms you provide from the blank sky. I don't think these pancake models are trying to be a sky subtraction either, because the code doesn't care about the average value, only the width of the distribution. (The code doesn't care if your residual distribution is centered on zero).

Results for SGAS J1152+3313

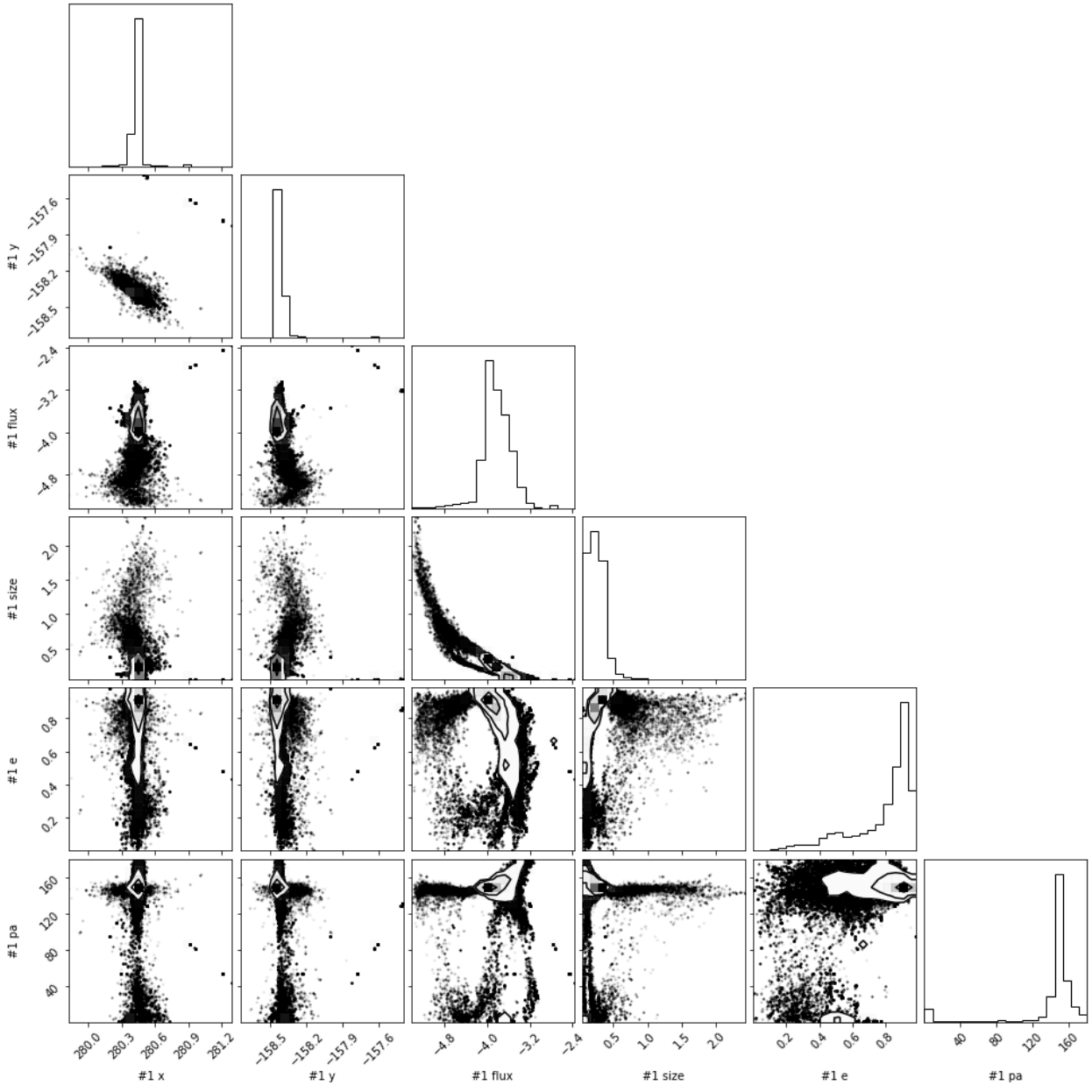
F110W	**FR601N / Lya**
	

F110W - Area 0

Current Best Fit - Gaussian, Unresolved

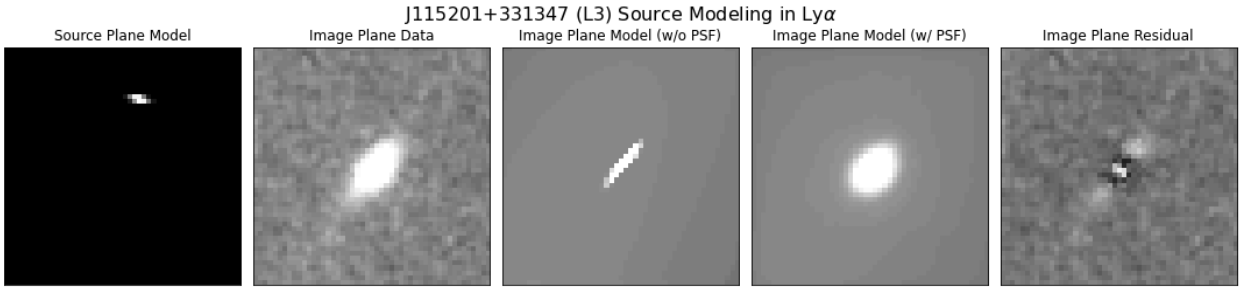


X = 280.43330375 Y = -158.42393905 Flux = -3.99877623 Size = 0.33053308 E = 0.91867791 PA = 148.02936018 --- Here size = 1 stddev in the x direction before rotation	Note that I used a logarithmic prior on the flux. I let 10^(x) be the flux value supplied to the modeler, but the flux parameter was the exponent with a tophat prior. Any size constraints here are dependent on the PSF.

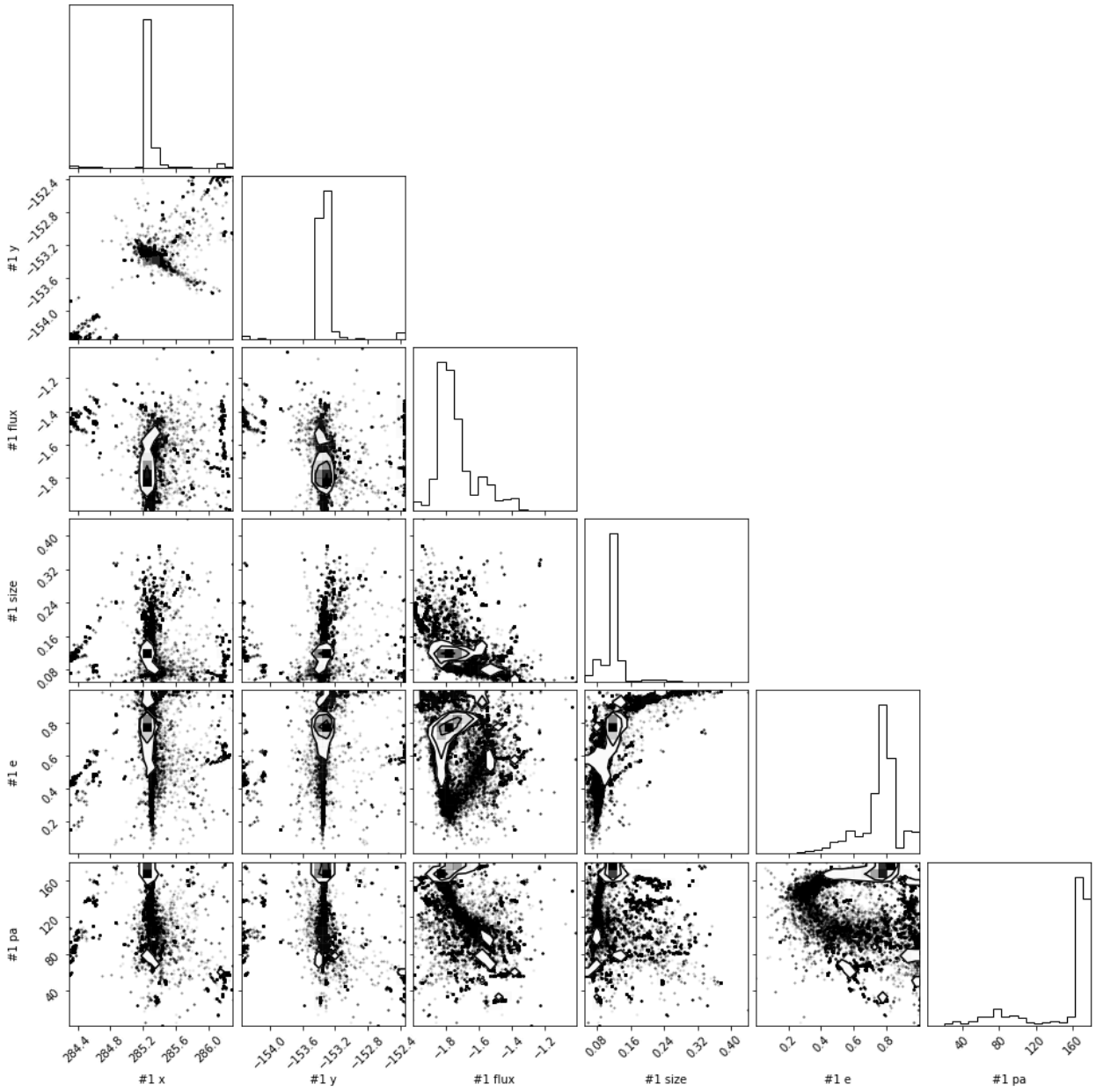

#1 x
#1 y
#1 flux
#1 size
#1 e
#1 pa
280.0
280.3
280.6
280.9
281.2
−157.6
−157.9
−158.2
−158.5
−2.4
−3.2
−4.0
−4.8
0.5
1.0
1.5
2.0
0.2
0.4
0.6
0.8
40
80
120
160

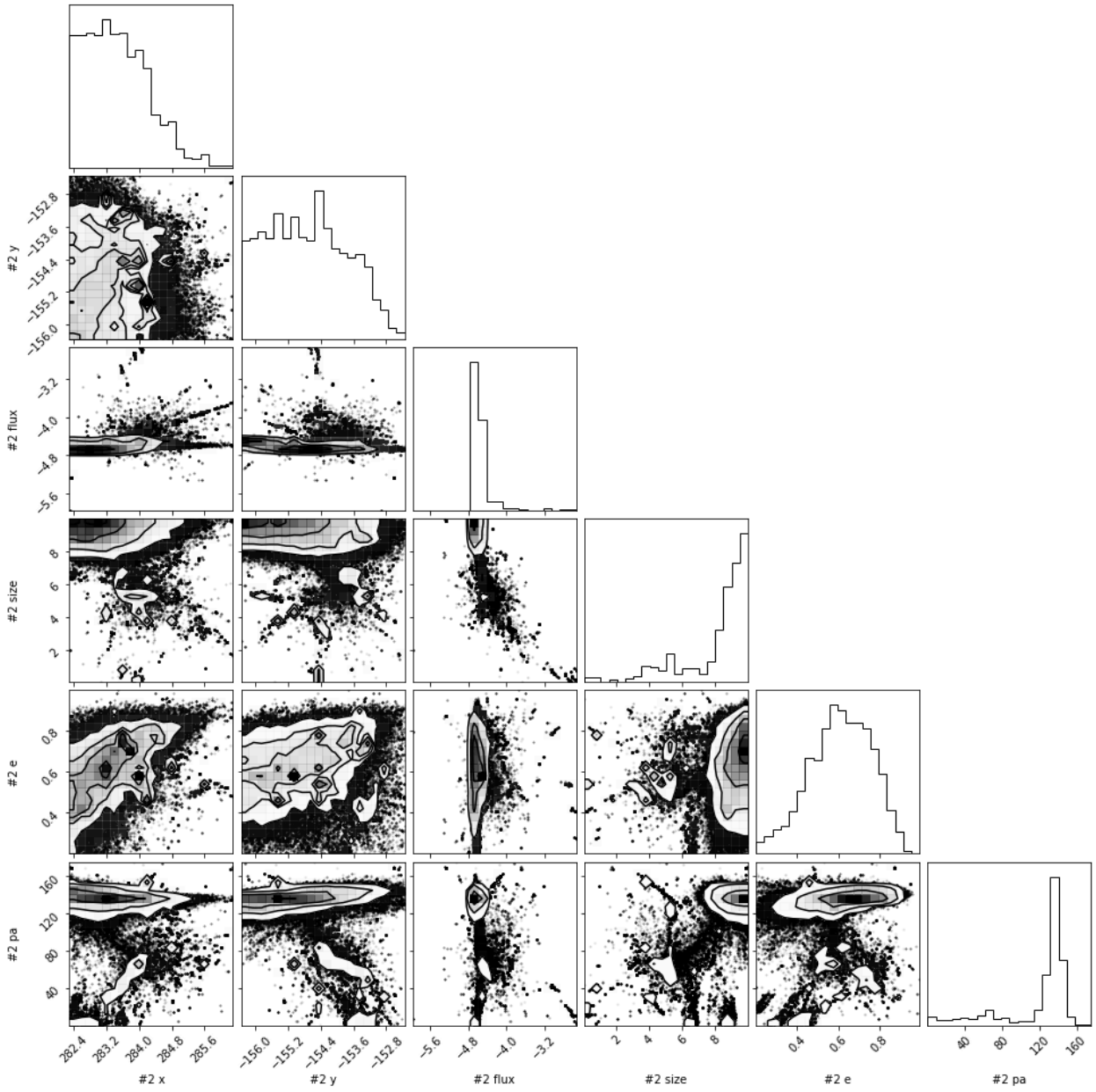
FR601N - Area 0

Current Best Fit - 2 Gaussians, Unresolved?

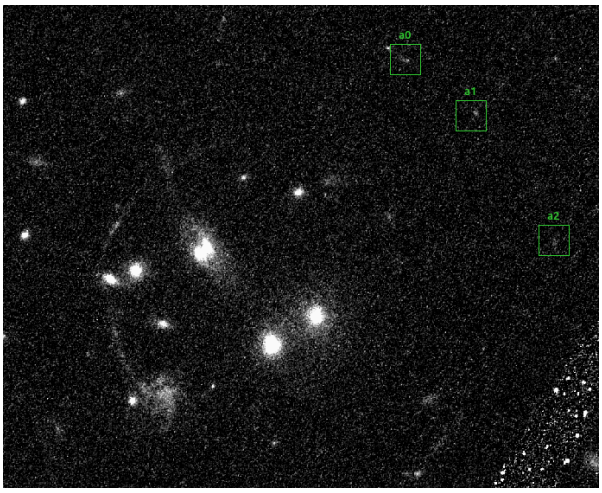
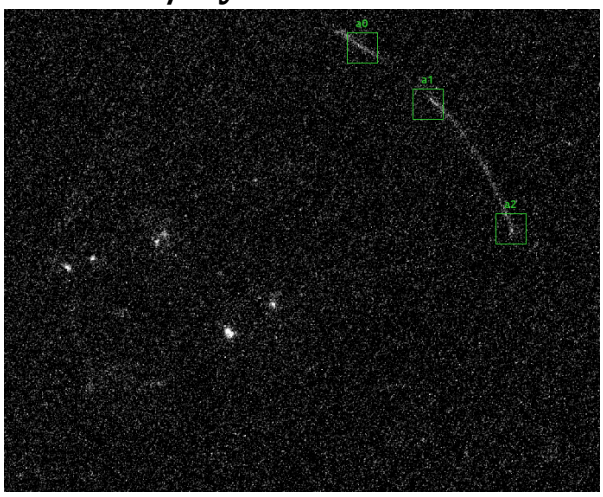


X1 = 2.85294264e+02 Y1 = -1.53350418e+02 Flux1 = -1.66343855e+00 Size1 = 1.18407819e-01 E1 = 8.25301386e-01 PA1 = 1.73597085e+02 --- X2 = 2.84821499e+02 Y2 = -1.55935123e+02 Flux2 = -4.58482863e+00 Size2 = 9.97711012e+00 E2 = 7.98751919e-01 PA2 = 1.32850588e+02 --- Here size = 1 stddev in the x direction before rotation	Note that I used a logarithmic prior on the flux. I let 10^(x) be the flux value supplied to the modeler, but the flux parameter was the exponent with a tophat prior. This was a result of letting size2 get very large. Between 1-5, size2 seemed to be doing some actual work modeling the faint Lya. But it always wanted to go higher. And now the 2nd gaussian has turned into a pancake.


#1 x
#1 y
#1 flux
#1 size
#1 e
#1 pa

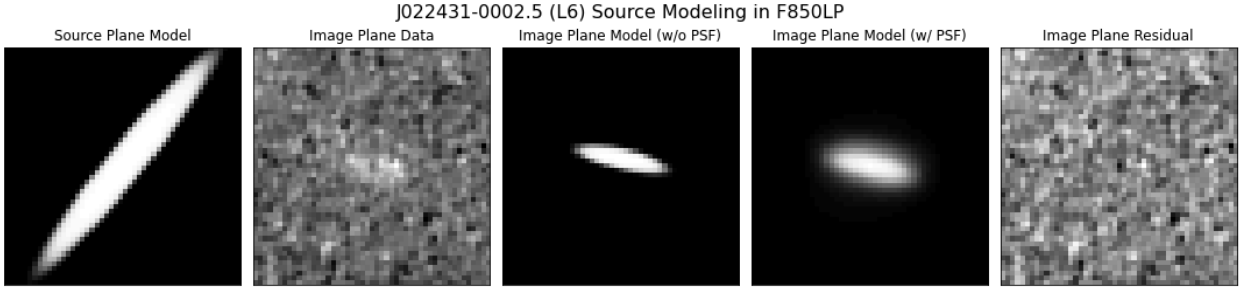

#2 x
#2 y
#2 flux
#2 size
#2 e
#2 pa
282.4
283.2
284.0
284.8
285.6
−156.0
−155.2
−154.4
−153.6
−152.8
−5.6
−4.8
−4.0
−3.2
2
4
6
8
0.4
0.6
0.8
40
80
120
160

Results for RCS1 J0224-0002

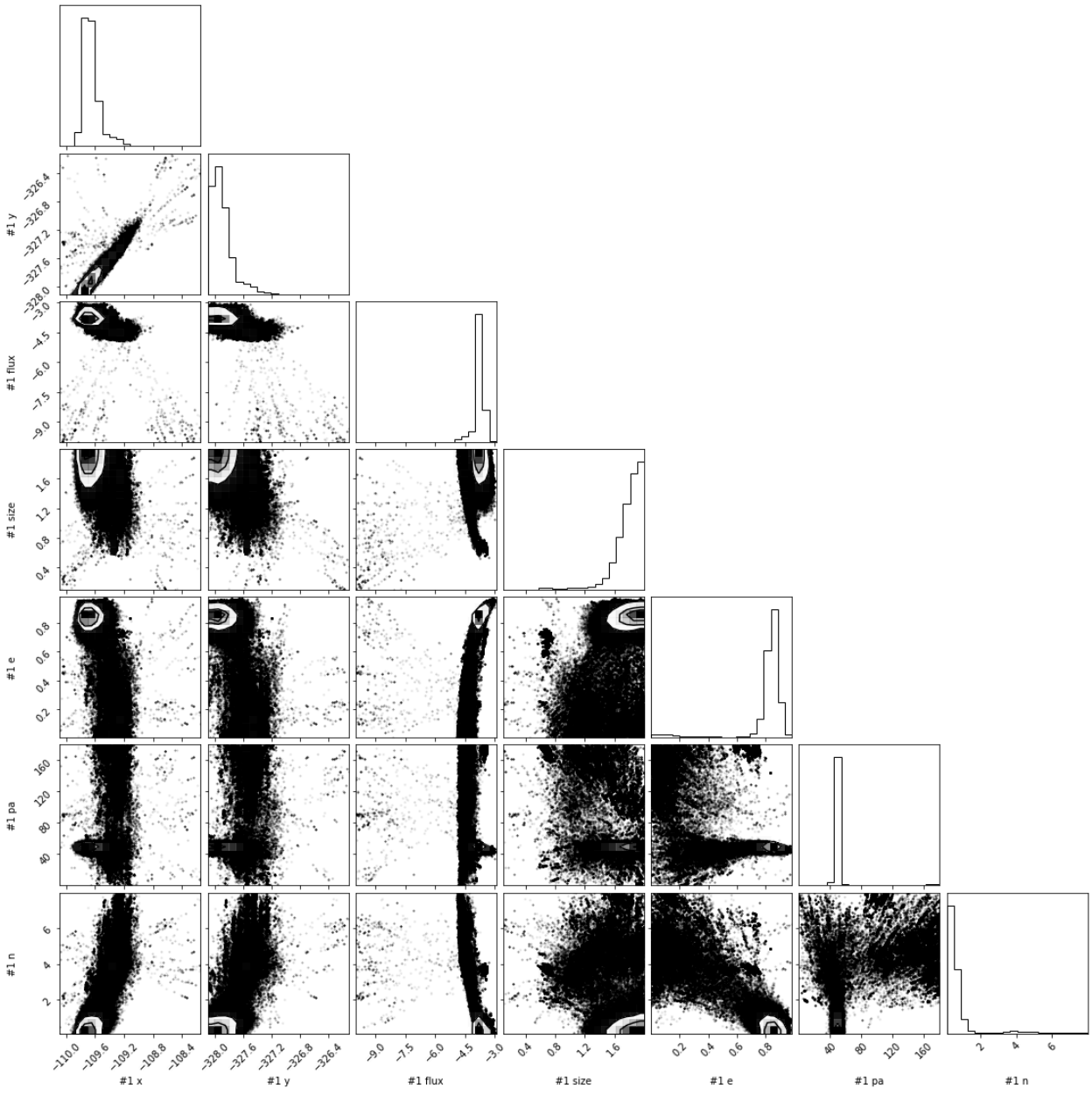
F850LP	FR716N / Lya
	

F850LP - Area 0

Current Best Fit - Sersic, Resolved

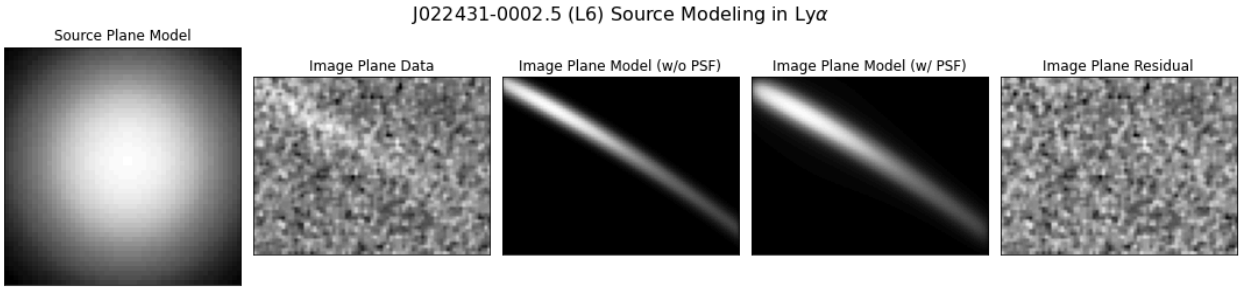


X = -3.28059495e+02 Y = -158.42393905 Flux = -3.65826003e+00 Size = 1.93045068e+00 E = 8.66164066e-01 PA = 5.09434931e+01 N = 1.12646956e-01 --- Here size = the effective half-light radius	Note that I used a logarithmic prior on the flux. I let 10^(x) be the flux value supplied to the modeler, but the flux parameter was the exponent with a tophat prior.

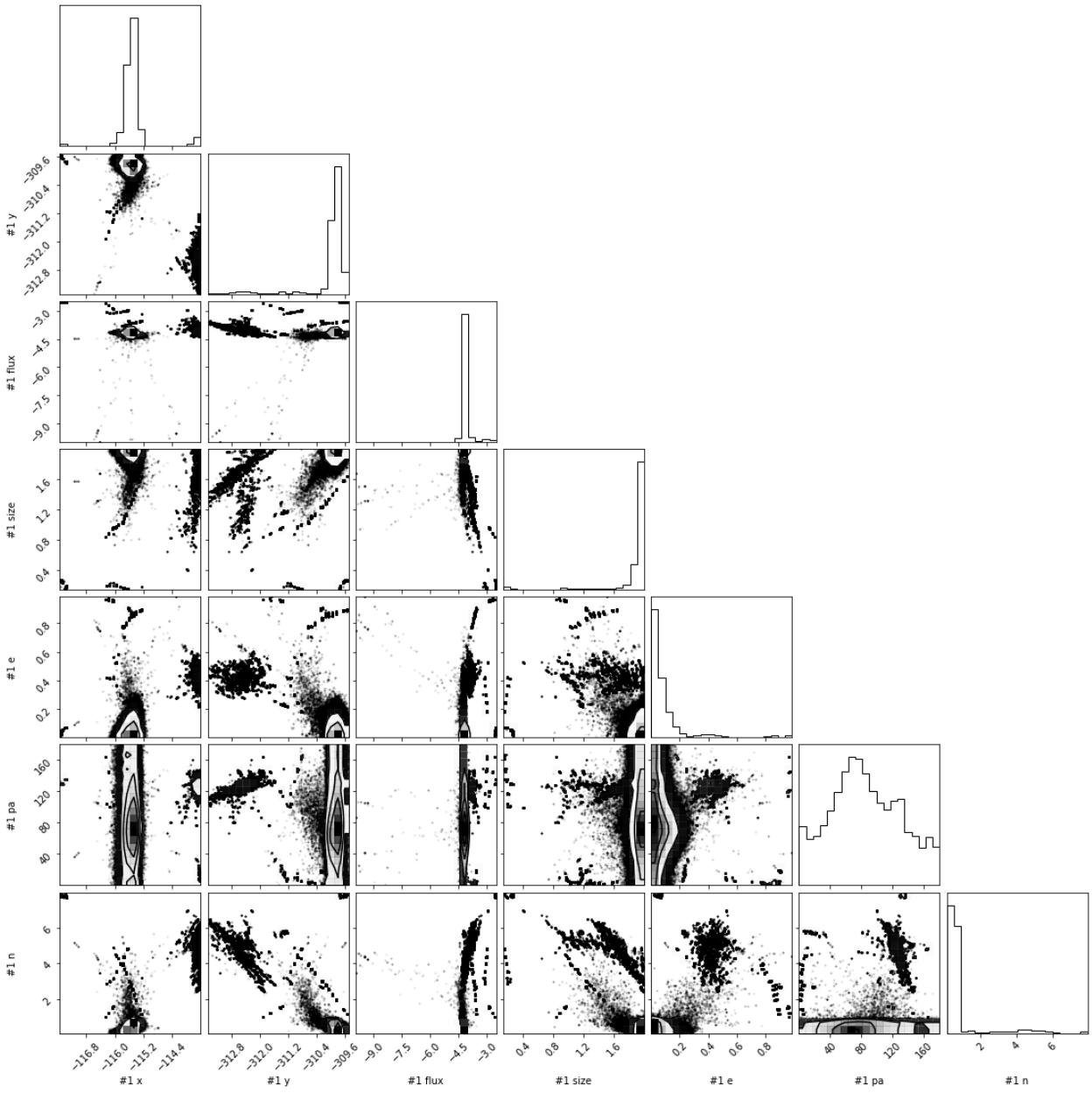

#1 x
#1 y
#1 flux
#1 size
#1 e
#1 pa
#1 n

FR716N - Area 0

Current Best Fit - Sersic, Resolved



X = 1.15543761e+02 Y = -3.09865889e+02 Flux = -4.31454726e+00 Size = 1.99906512e+00 E = 1.28968810e-02 PA = 9.03950385e+01 N = 5.50307234e-01 --- Here size = the effective half-light radius	Note that I used a logarithmic prior on the flux. I let 10^(x) be the flux value supplied to the modeler, but the flux parameter was the exponent with a tophat prior. I limited size at 2.0. The cornerplot shows it wanting to be larger, but I fear that it will turn into a pancake model.


#1 x
#1 y
#1 flux
#1 size
#1 e
#1 pa
#1 n

Suggestions for Code Improvement

- Identify old, commented-out code that isn't used and remove from files.
- Create a folder architecture so that file paths are more sensible. Here is a basic example of what I used.

```
inputmain='/home/alex/research/floriancode''
cluster = '0224'
area = 'a0_lya'
folder = '/'+cluster+'_'+area+'/'
```

- Remove functions that aren't used from clump_forward_modeling
- Across the board, either comment in what variables mean or give them more context-friendly names.
- Take a look at the create_outputs_drz function.
    - Sometimes the output files are not made with the best of the MCMC chain. My own functions sometimes find better fits.
    - Add functionality for creating outputs that include models with fixed parameters. Currently, it only makes outputs with free parameters.
- Create a normalized file naming system.
    - For example the .p file that realSourceReconstructions makes. Have realSourceReconstructions save the file name with the object's identifier and the clump number. Or something like that. And then have sersic_run_mcmc automatically search for the relevant filename. Right now you have to manually change the strings.
- Make sure that when accessing files, they allow for appending. Make sure that if you add 1000 iterations to your input file that the rest of the files append the extra data, and doesn't overwrite it.
- Add in a user input in the beginning of files to make sure files are correct. Something like:

```
…print out file names…
print('Are These Files Correct? (y/n)')
ans = input()
if ans !='y':
        sys.exit()
```

    - I have made the mistake of supplying the wrong file too many times.
- If there are variables that you have to enter manually between executions of the code, then make sure they are at the top of the code.

PythonPSF

PythonPSF is a set of functions that model a PSF from stars in-field of your science data. Everything was originally written by Michael Gladders in IDL, and Alex Navarre translated it to Python.

All you need to supply to this is

1. The science data
2. A ds9 region file of circles around non-saturated stars in the science data.
3. A path to a galfit executable.
    a. GALFIT can be accessed here.
    b. Alex Navarre has written two files to help with GALFIT
        i. Galfit Setup
        ii. Galfit Tips and Tricks

Access to PythonPSF is on github here:
https://github.com/astronavarre/PythonEmpiricalPSF

Since this is proprietary software to the SGAS collaboration, you likely won't be able to access the page. Email Alex Navarre with your github ID at aenavarre@gmail.com to be added to the whitelist. Or you can download a copy of the entire repository with git:

```
git clone
https://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@github.com/astronavarre/PythonEmpiricalPSF.git
```

Alex's Navarre's Functions for Analysis

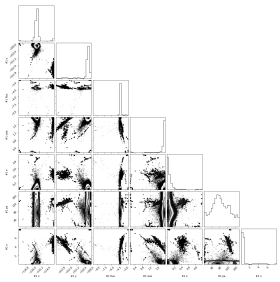
Code Access - Link Here

General Note - Since there is currently no standard path naming system in this code, you will need to change path/file names to those you want to use.

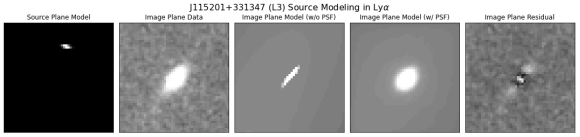
General Note - I have commented on all the functions what I think you might need to change. If I missed something, or you are confused, feel free to message me, Alex Navarre, on slack.

Function List

- **find_files** and **find_fits** - Convenience functions to find files in the specified directories.
- **cornerplot** - Creates triangle plots (confusograms) like those seen in the previous sections



- **align** - Crops and aligns your science data to your displacement maps. Needed for sersic_run_mcmc to work.
- **bestfitplot** - Creates best fit plots like those seen in previous sections.



- **maskfromregion** - Takes a region file and the aligned science data, and outputs the aperture file. Needed for sersic_run_mcmc to work.
- **distplots** - Creates individual plots of all the free parameters in the run. Use this if they are too small in the triangle plot or you want to see the distribution with a finer bin size. Also calculates percentiles of the distributions (16th, 50th, 84th - Gaussian mean and +/- 1 stddev, and 90th and 95th).