Author: Karen Ng karenyng@ucdavis.edu
Code - refactored version of Will Dawson 's nfwMCMC.py
Written with markdown

Abstract of this project

This summarizes the difficulties and thoughts behind the reorganization of the MCMC code for fitting the mass of dark matter halo.

- most of the code involves preparing the data in a suitable coordinate frame (between sky and image coordinates) / format but in the current code, the preparation and the MCMC steps are tangled, making it impossible to debug the code modify or extend the MCMC to make infererence for more than two parameters. There are many other methods to infer two parameters by fitting, MCMC is an inefficient way if we only try to infer two parameters.

- the current loop structure can be confusing and contained a subtle bug I had to fix
- reorganizing the large number of variables (~40) of various nature
(physical, cosmological, tuning, statistical etc.) in a
logical, computationally efficient and concise manner (no unnecessary copies)

Notations

I use the following notation to denote
[x] what I have done in detailed,
[0] what I skimmed through and,
[ ] what I planned / thought about but did not do - empty

Goals

[x] identify where the calculation can contribute a factor of 2 / 3 difference
[0] fix bug in the new version of the code.
[0] make sure we can reproduce collaborator's published result on the same set of data
[x] make sure we can reproduce collaborator's result on his mock dataset
[x] make sure we can get correct mass estimates of mock data (currently we can only get results of only a self-produced noiseless data set
correctly) and every version of this code has got correct result for this mock data
[0] modularize the code so we can prepare the data once,
then use multiple stat / data analysis techniques on the prepared data
to verify the results

Methods

Tools learned throughout the debugging / code refactoring process

- python debugger - ipdb

- unit test module - pytest

- how python store variables by binding the variable names to the memory address of the values

- **how to output them / read them back in elsewhere automatically with correct variable types and same names efficiently.** In R, this can easily done by saving objects / variables in .rda files, but there is no such built-in facilities in python but it is doable with some hacking. For each python function the namespace is stored as a dictionary mapping the variable name to the values. The hack is to output the entire dictionary representing the namespace to a python pickle file which would also preserve variable type. Printing / writing numpy arrays / panda dataframe naively will destroy the type / data structure and would require extra conversion while reading them back in. Reading the variables back using the hack in is as easy as updating the local namespace with the dictionary. But updating the local namespace is done implicitly - no information of what variables is output by default.
- how to write python classes

Approach - key diagnostics

breadth-first search, examine key areas for possible errors:

[x] Wrong data catalog / Error in the catalog - can be checked by plotting the catalog (See Figure 1)

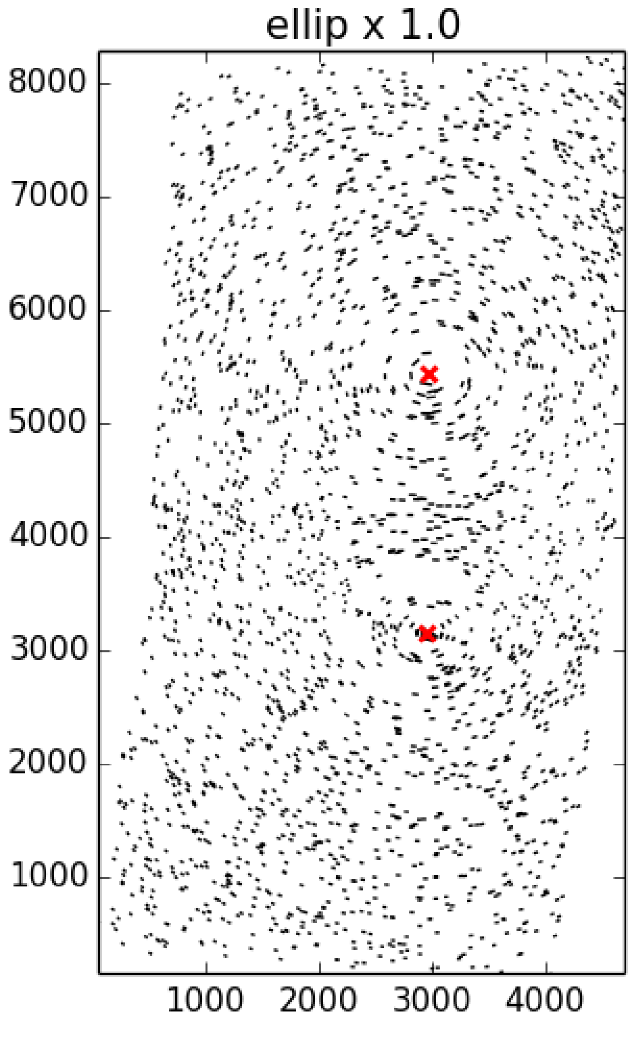

Figure 1: Visualize mock data - red points are the assumed halo centers. This looks ok.

[x] transformation of data - can be checked by plotting the transformed quantities of the mock data (See Figure 2)

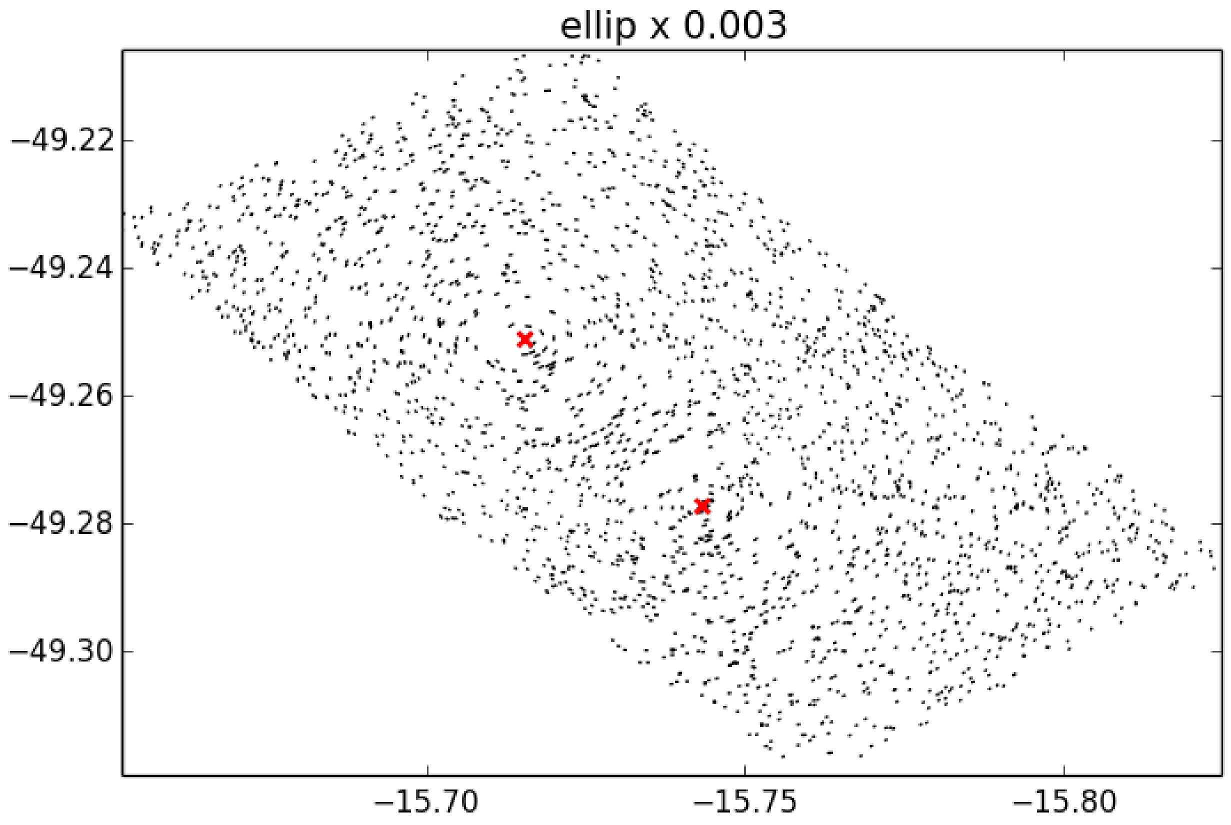

Figure 2: Visualize mock data after transformation to sky coordinates - red points are the assumed halo centers. We can still see the circular distortion around the red centers . So this should be ok.

[x] masking of data - can be checked again by plotting (See Figure 3.)

[x] physics / cosmology - visually examined equations
[x] **Other data preparation steps** - can be checked by plotting if log likelihood of fit between mock data and physical model look ok over the relevant parameter space - (See Figure 4.) And it totally did not look ok so we can be certain there is something wrong before the MCMC is performed.

[ ] data analysis - can be checked by swapping out the MCMC code with something else and see if we get the same answers

Once we narraw down which area(s) is(are) problematic, then we perform more in-depth tests. And we found that it is the data preparation that is problematic so it narrows down the search area to only half of the code. There might still be bugs in the MCMC part but we are certain there is something wrong upstream of the MCMC and we have to fix the bug upstream first.

RESULTS

Where / what the two bugs are

1. two of the variables in one of the input array with the following form is swapped and these denote which column in the data catalog that the code should grab for the calculation:

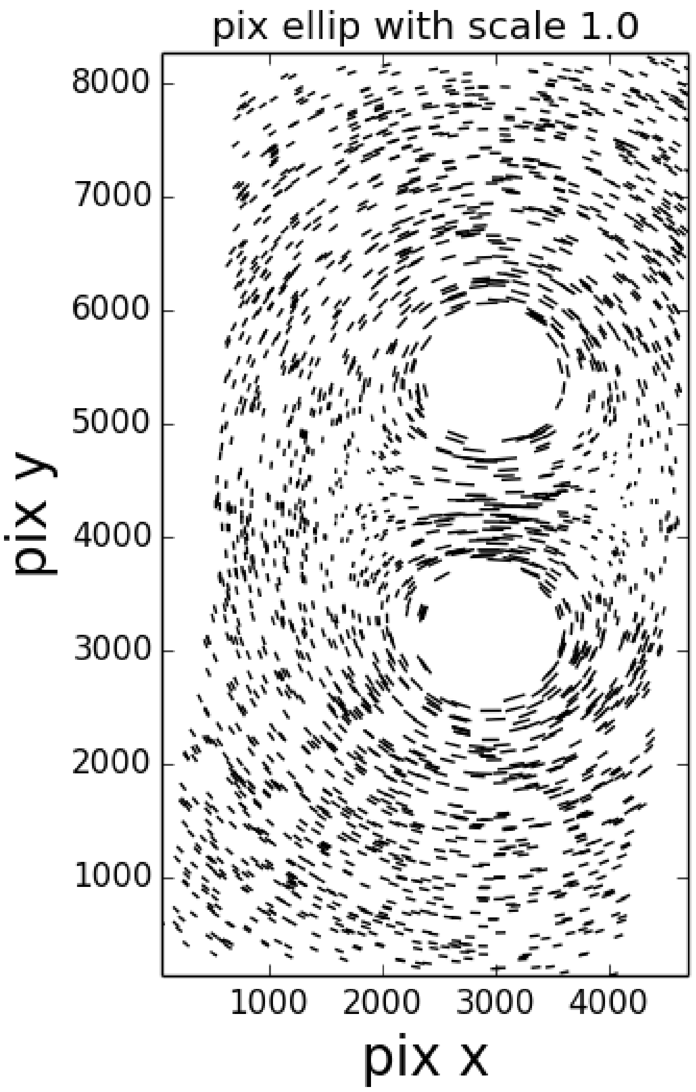

Figure 3: Check if mask is applied correctly for our model with unknown mass. Same mask is used for masking in sky coordinates. This looks ok.

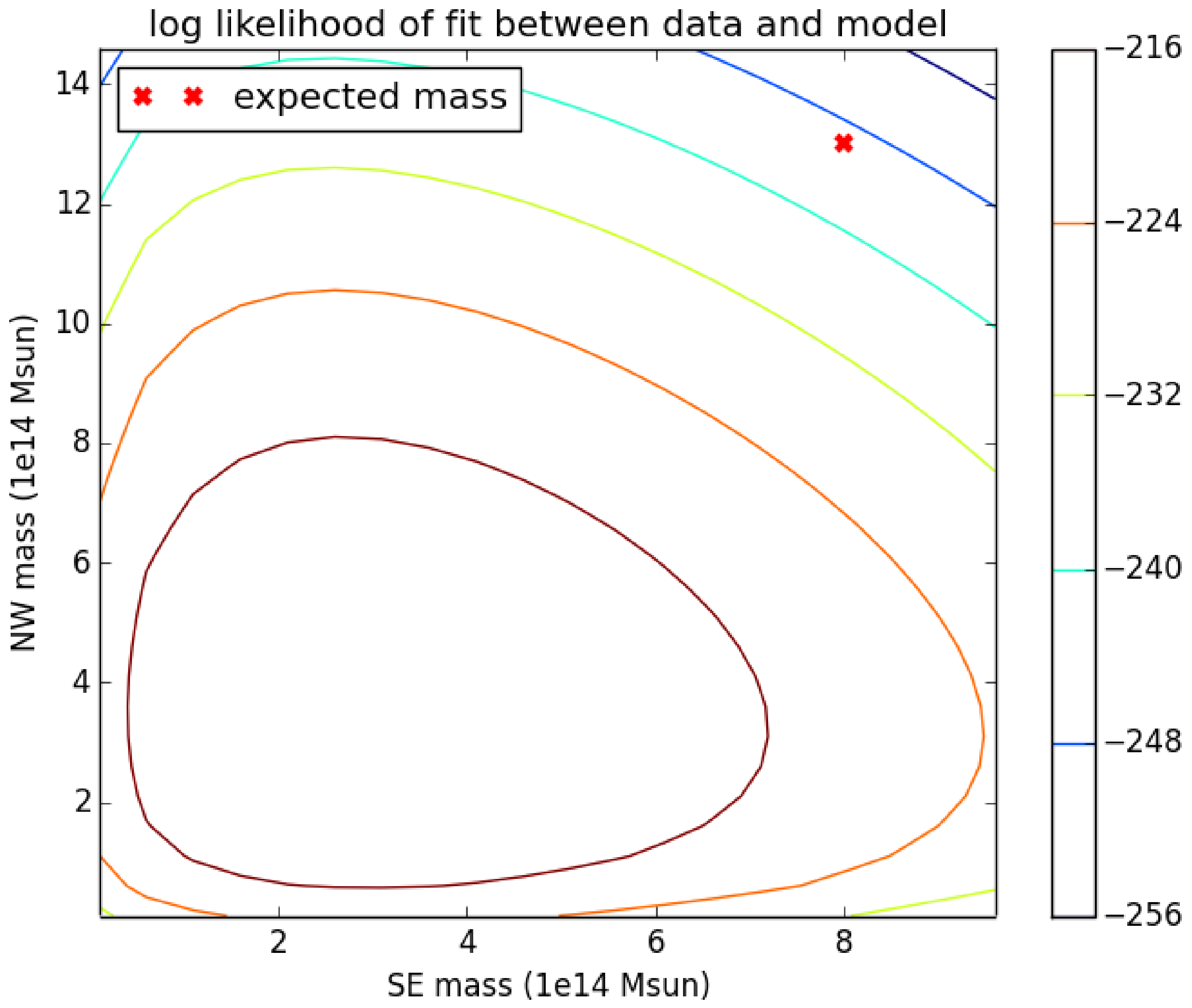

Figure 4: Check if the calculated log likelihood of fit between model and data makes any sense. And it didn't! The expected mass is way off and the log likelihood tells us the mass should be way lower than expected.

```
# wrong form
ellip_meas = ['e1_pix', 'de', 'e2_pix', 'de']

# correct form
ellip_meas = ['e1_pix', 'e2_pix', 'de', 'de']
```

This bug caused part of the ellipticities (data) to be swapped with the noise column of the data catalog. In the mock catalog 'e1__pix' has similar magnitude with 'de' so I could not spot it even if I printed the values out. I had to debug it with the real data in order to find this.

2. physical units of the outputs were wrong both in the documentation and in the printed output causing us to quote the estimated physical parameter wrongly by (a factor of ~1.4 smaller) and make result seem artificially low.

```
# wrong units
1e14 Msun

# correct units
1e14 h^{-1} Msun
```

where h = 0.7 in our case and depends on the cosmology used. this is why I am always paranoid about people not putting units in their calculation or documenting their code properly!)

Mass estimates from collaborator

Collaborator generated the mock catalog based on his previous mass estimates of the real data:

$$M_{NW} \approx 11.8 \pm^{2.94}_{2.31} \times 10^{14} h^{-1} M_{sun}$$

and

$$M_{SE} \approx 5.74 \pm^{1.33}_{1.05} \times 10^{14} h^{-1} M_{sun}$$

Result from global test 1 - collaborator's mock catalog

Output from the code which has confusing units - this is to be fixed

```
Halo NW:
M_200 = 13.5e+14 +/- 1.86e+14 x h^{-1} solar mass Gelmans & Rubin Confidence Limits
M_200 Mean and Bias Corrected 1sigma Credible Limits:
Mean = 13.5e+14 UCL = 15.5e+14 LCL = 11.7e+14

Halo SE:
M_200 = 7.05e+14 +/- 1.43e+14 x h^{-1} solar mass Gelmans & Rubin Confidence Limits
M_200 Mean and Bias Corrected 1sigma Credible Limits:
Mean = 7.05e+14 UCL = 8.63e+14 LCL = 5.67e+14
```

Our estimates are within the 1 sigma credible limits of our collaborator and vice versa.

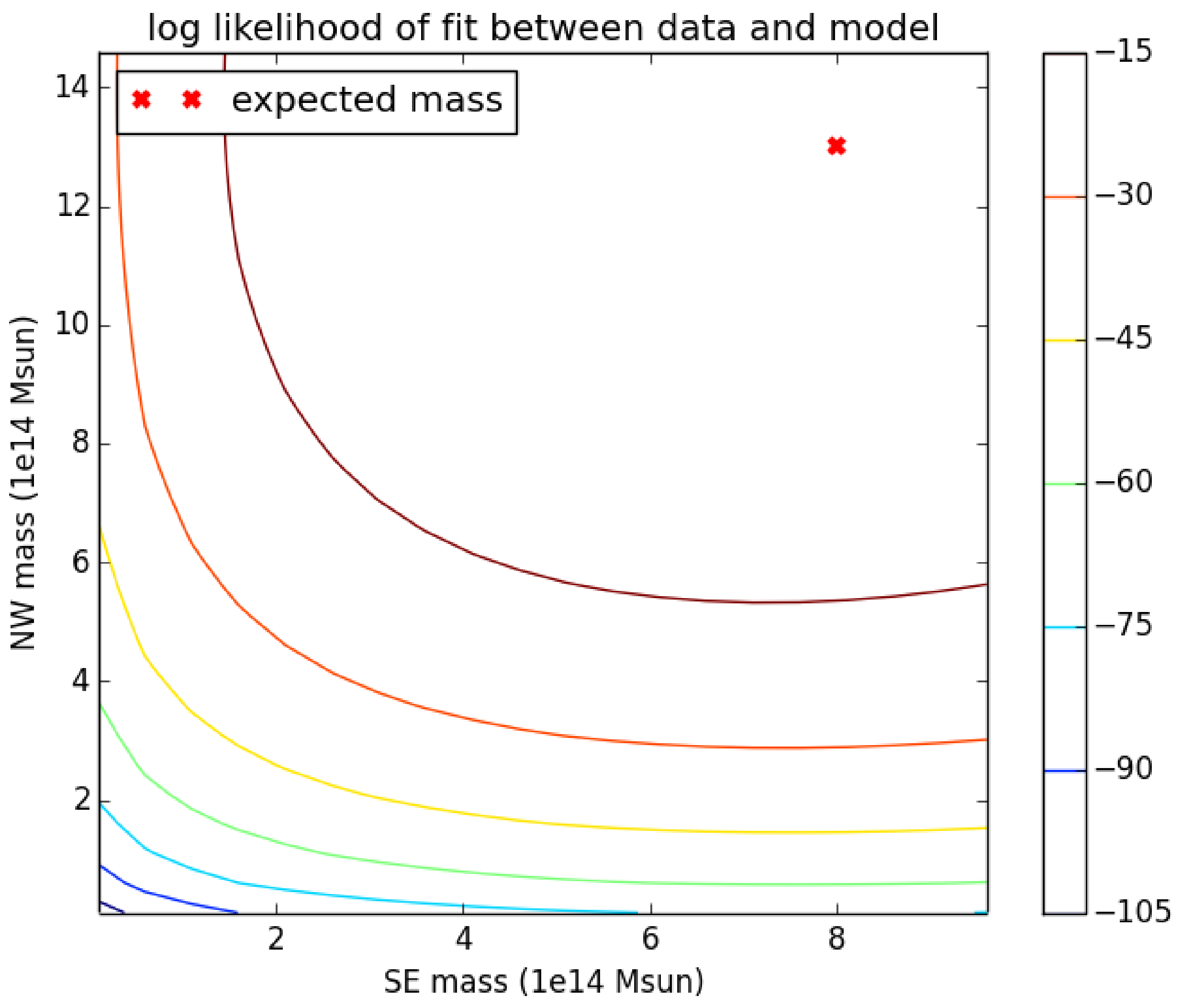

Figure 5: Check if the calculated log likelihood of fit between model and data makes any sense. And it now does after bug fix.

Result for real data

```
Halo NW:
M_200 = 11.0e+14 +/- 1.62e+14 x h^{-1} solar mass Gelmans & Rubin Confidence Limits
M_200 Mean and Bias Corrected 1sigma Confidence Limits:
Mean = 11.0e+14 UCL = 12.7e+14 LCL = 9.32e+14

Halo SE:
M_200 = 5.07e+14 +/- 1.19e+14 x h^{-1} solar mass Gelmans & Rubin Confidence Limits
M_200 Mean and Bias Corrected 1sigma Confidence Limits:
Mean = 5.07e+14 UCL = 6.35e+14 LCL = 3.91e+14
```

Our estimates for the real data are closer to those of our collaborator than the mock data.

Discussion

Superficially the summary statistics look more consistent than before after the bug fixes. Since our uncertainties for the mass estimates is not small and the other part of the data analysis does not depend sensitively on this mass estimate, it seems like diminishing return to refactor and test every part of the code at this stage.

what might still be worth improving at this stage

- restructure the code so the preparation of the catalog is separate from the stat inference step so multiple stat methods can be used more easily
- correct all documentation about the units in outputs
- put better documentation about how to write out and read back in the intermediate outputs
- add documentation for making the diagnostics plots
- add documentation for unit tests so others can reuse them

Other possible extensions / tests

- found literature for how to test the data integrity using alternative method by taking azimuthal averages of the data about halo center to do 1D fitting halo by halo (Umetsu 2013)
- found *glafic* online which is a simulation package of halo models which can be used to check our halo models
- refactor the code with OOP and add floating centers of the halos as part of the inference

Files:

- code to debug - nfwMCMC.py
- refactored code - work in progress
  - MCMC_nfw.py - replaced numpy arrays with pandas dataframe for representing the data catalog
  - nfwClass.py - draft of how to use data structure to encapsulate different types of variables
  - prepare_cat.py - code to make mock catalogs / massage catalogs

Unit tests

- under test directory

- to run tests, install pytest module and then at a terminal, run:

  ```
  py.test test_*.py
  ```

Global tests

```
python ./tests/data/call_*.py n
```

Appendix

What I have also tried

- use groupmate 's latest git commit 98de1d6
- used last working version in commit 9ec8ba8 that had 20% discrepancy (actually only 10% but we had wrong units), run it with new inputs, the mass estimate still a factor of 2 / 3 too low
- reverting code back to vanilla version git commit d40fdc2 from master branch on private git repository - the mass estimate is even lower

Hypotheses of what could have contributed to a factor of ~3 difference

The following is actually a log of what I have done.

Wrong physics / cosmology?

some of the physics equations we use can have wrong form during transcription from paper to code
[?] different ellipticity definitions - not seeing that explicitly anywhere
[?] the way we are adding shear - still not likely
[0] lensing definitions - checked the forms in the function visually
[x] unit error – conversion between 10^{14} solar mass and solar mass
[x] cosmological distance calculations
[x] wrong physical constants
[x] coordinate transformations / approximations
[x] compare definitions of shear signal that we used from Umetsu 2000 and another paper Brainerd 1999

Wrong statistics / data analysis?

anything related to implementation of MCMC / stat inference
[?] check log likelihood of fit between data and model near expected mass values
[ ] check how the mass values are stored after the MCMC is run
[ ] result is highly sensitive to inputs such as centroid locations of halos – extremely unlikely

Bugs during data preparation?

[ ] the way that we are applying correction to the shear is different than our collaborator - he applied all the corrections to the model,
we applied all the corrections to the data . . . not very likely and effect should be at a level of several percent
[0] visualize how the masking is done - the central region seems ok
[0] model ellipticities are not computed correctly - magnitude might be off, orientation looks ok
[x] examined percentage differences of various approximations
taken during coordinate transformation - only discrepant at sub-percent level

Wrong inputs?

[x] double check all the corrections we apply to the data are identical to those of collaborator's
[x] wrote functions to raise errors for problematic inputs
[x] wrote a debugging mode of the input code so it forces lazy user (me) to examine inputs one by one before starting the code - fixed 2 problematic inputs
[x] wrote function to plot the data in correct coordinate frames before and after coordinate transformation
[x] wrote function to have the exact inputs written to a file
to diagnose problems after running the code

but it could be cumulative effects of a bunch of smaller errors ....