README/Overview of Code for LEAPS Project Summer 2018 – Pixelisation/Undersampling in Metacalibration

Erik Rosenberg

August 22, 2018

1 Python Scripts and Modules

Note that more complete documentation of all modules / functions is provided in the python files themselves.

1.1 reconv_shear_metacal.py

Requirements

Uses functions from my `mcal` and `galFuncs` modules.

Description

This is the primary script that I use to run the metacal process as well as the control branch. Most of the work is done by functions from the `mcal` module, called from the `main` function. The two `makeGals` functions, for Grid and Cosmos, are also useful.

Functions Defined

- **removeBulge** – Return Disk part of a bulge+disk. If not bulge+disk return the galaxy profile
- **makeSersic** – Make a Sersic profile given n, hlr, flux, q, phi, gsparams. Used for grid.
- **makeGalsGrid** – Given a distribution in hlr and q and a constant phi and n, make a grid of Sersic galaxies and return as a 1d array.
- **makeGalsCosmos** – Make galaxies from COSMOS. Note that cuts are hard-coded into the function.
- **main** – The main function executed; create galaxies, perform metacal/control, measure shape, and save results in a pickle file.

Notes and Instructions

To modify parameters and such for each run, change the code after the end of the main function (currently line 230). Some parameters that might be useful to change or at least be aware of – I have bolded the ones I most often change:

- Grid parameters – Parameters used only for running on a grid, described in the makeGalsGrid section.
- **ngal** – The (total) number of galaxies in a COSMOS run
- sstep, sl, shearList – I usually leave these, but can be modified for different artifical shears.

- **methodI** – 1 for metacal or 2 for control. 0 usually not used.
- nrot – Usually leave at 2 for a 0/90deg rotated pair.
- **shear_est** – “REGAUSS” or “KSB”
- **cosmo_shear** – Cosmological shear
- noiseSNR, noise – Usually left at none, but could be useful if you use noise. If you do, see commented block below.
- **pixel_scale** – Often set from command line (see below) but also often set here.
- **numGalStep** – Set the number of galaxies to be done in this sub-run (see parallelization section)

In the next block I get values from the command line. Currently I have a very simplistic setup where I can only take one argument from the command line, and the rest are set here in the script (in the above-described parameters). When I use these, I do so using the scripts in /home/rosenberg/Documents/wl-bias-leaps-top/wl-bias-leaps/shell_scripts/.

The different values that can come from the command line are

- ii – This is for parallelizing a run. If you would like to split your run of 1000 galaxies into 4 subruns then make sure numGalStep is set to 1000 / 4 = 250 and that all other values (like pixel_scale) have been set in the script. Then run the script with ii=0, ii=1, ..., ii=3.
- rotAngleDeg – For setting the angle of all galaxies on the grid
- pixel_scale – Make sure that either ii / numGalStep are set to what you want, or instead that start, stop are set to None. Then you can run with different pixel scales (eg 0.05 0.06 ... 0.13)
- Noise – Specify SNR

Finally, the last couple lines are where the main function is called. Some things (notably **lamda**, **redrawScaleFactor**) are set explicitly here instead of having a variable (this could easily be changed). Note that I have one call to run with the grid and another to run with COSMOS. The main difference is that COSMOS has the first argument gridparams=None and ngal specified while grid has gridparams specified and ngal=None. Grid should also have rotGals=True (while it is almost always False for COSMOS). Note that there are also lots of other parameters to main, many of which have defaults that you may want to change.

After running, the results are returned in the **res** variable. They are also saved in a pkl file, currently in /home/rosenberg/Documents/wl-bias-leaps-top/shear_bias_outputs/.

1.2 mcal.py

Requirements

None

Description

This is a module that contains a range of functions called by `reconv_shear_metacal.py`, core to running the metacal process.

Functions

- **dilate** – Dilate a profile
- **dilatePSF** – Dilate analytical PSF and reconvolve with pixel response
- **measureShapeBasic** – Called by the other `measureShape` functions, this one convolves a galaxy and psf profile and measures the shape.
- **measureShapeControl** – Called in control branch – Dilate psf and shear galaxy, then call `measureShapeBasic`.
- **measureShapeReconv** – Called in metacal branch – Perform metacal process (convolve w/ PSF, draw, interpolate, deconvolve by psfii, shear) then call `measureShapeBasic` to measure shape.
- **galShiftErrs** – Perform subpixel shifts then perform any of the three `measureShape` algorithms on all pixel-shifted galaxy images. Called by `galShiftErrsBatch`.
- **galShiftErrsBatch** – Call `galShiftErrs` for an array of galaxies and artificial shears. Called directly from `reconv_shear_metacal.py`.

To summarize, the call sequence goes as follows
`User` →`reconv_shear_metacal.py` →`galShiftErrsBatch` →`galShiftErrs` →`measureShapeReconv` OR `measureShapeControl` →`measureShapeBasic`

1.3 galFuncs.py

Requirements

None

Description

Miscellaneous utility functions for galaxy manipulations and for bootstrapping.

Functions

- **rotGal** – Rotate a galaxy to a given angle relative to x-axis.
- **circularize** – Circularize profile, removing all shears (no bulge+disk!)
- **makeGalaxy**
- **nanav** – Take weighted average, ignoring nans
- **Bootstrapping functions** – These are mostly called from the bootstrap function contained in the `responsivity.ipynb` notebook.
  - **getRandomWeights**
  - **matchShapes**
  - **getWeightArr**
  - **bootstrapArr**

1.4 responsivity.py

Requirements

`galFuncs`

Description

Load data output by reconv_shear_metacal.py, calculate R and recover the cosmological shear, and plot.

Functions

- **e2g**: Convert e-type to g-type
- **bootstrap**
- **calculateR**: Calculate R matrix from ellipticities
- **matchNans**: Match nans for all galaxy rotations and all artificial shears
- **recover_cosmoshear**: Calculate recovered cosmological shear from R and ellipticity.
- **matchNansBetween**: Match nans between control/metacal branch, different wavelengths, or similar.
- **main1**: Filename to recovered shear pipeline for one set of files (eg just control, metacal)
- **main**: Filename to recovered shear pipeline for multiple sets of files (eg control and metacal, multiple wavelengths)
- **plot**

2 Shell Scripts

I have a few short shell scripts used in processing the metacal results.

The **reconv_shear_metacal_loop_*.sh** scripts are all exactly identical (and don't really need to be separate scripts I guess) except for the numbers in the foreach loop. For each of these numbers the main function of reconv_shear_metacal.py is run, with that number passed as the command line argument. Note that the interpretation of the number is set in reconv_shear_metacal.py script (near the bottom). The anacondaOff line can probably be removed when you run it – it is an alias I've defined that makes sure that Anaconda (which I've locally installed to let me run Jupyter Notebook on veersemeer) is removed from the path before running.

There are also a few other short scripts that can be useful for processing. Currently all results of reconv_shear_metacal are saved in /home/rosenberg/Documents/wl-bias-leaps-top/shear_bias_outputs/. Usually it is good to put them in directories instead. Also sub-runs made with parallelization need to be combined. This is done with **combine_pickles.py** (not actually a shell script, but I use it as such, so it's here). The way it's currently set up, this script should be called from directory *A* containing subdirectories *A1*, *A2*, etc. where each of *An* contains pickle files that should be combined (ie files with the same pixel scale, wavelength etc.). They are combined and the resulting pickle deposited in directory *A*.

Finally **sortByWlPs.sh** is for making a directory structure of lambda / pixel size. I don't use it so much but it removes some tedium in making sub-directories for running **combine_pickles.py**.

3 Notebooks

- **responsivity.ipynb** – Load data from `reconv_shear_metacal.py` runs, calculate R and mean ellipticities, bootstrapping, and plotting of the results. Essentially replaced by responsivity.py, but keeping it around for now.
- **responsivity-control_matchpopulations.ipynb** – Almost identical to the above but slightly more complicated, includes matching populations of responsivity and control. Essentially replaced by responsivity.py, but keeping it around for now.

- **responsivity-histograms&otherplots.ipynb** – Not really used – mostly stores older code that was used to make histograms, R, and other diagnostic plots
- **bootstrap_unittest.ipynb** – Unit tests on the bootstrap
- **err_vs_pixelsize.ipynb** – Old studies of error vs pixel size
- **ngmix-metacal.ipynb** – Old study of how to run the DES ngmix/metacal code