Fake Objects Pipeline

- This is based on Claire's new version

Make the catalog for a given tract:

```
makeSourceList.py /lustre/Subaru/SSP/ --rerun
=clackner/SSP361_complete/testClean --id tract=0
filter='HSC-I' patch='1,1' inputCat='/home/clackner
/cosmos_25p2_profiles/ser_25.2_listarcsec_clean
.fits'
```

- The patch number is a dummy that just makes sure the code runs once per tract, by default it gives 500 ra/dec per patch, which can be configured. It's probably a bit on the high end.
- this matches the random ra/decs with random galaxies drawn from inputCat, which is the morphology catalog to **I=25.2** from COSMOS-HST used.
- There's also one that only goes to 23.5. This catalog should be culled of weird big/bright sources and poor quality fits before being used.
- I've done some really crude cuts, which are in the `*_clean.fits` catalog:

```
b_a > 0.1
0.5 < n < 5.5
reff < 5 arcsec
```

* `makeSourceList.py` makes an output catalog of fake sources called `src_TRACT_radec.fits` which is used by `runAddFakes.py`
* It takes ~3-4 seconds per tract
* I've added columns `g1` and `g2` for shear by hand, but you'd want to either add that to makeSourceList or write new code to do that.

Add the fake sources to a single visit:

```
runAddFakes.py /lustre/Subaru/SSP/ --rerun clackner
/SSP361_complete/testClean:clackner/testAdd --queue
small --job testAdd --nodes 3 --procs 12 --id visit
=1228 --clobber-config -C config_src0
```

- It takes ~30 seconds per CCD, I've been having trouble running it fully parallelized (That's either because I had a now-fixed bug or because it's hitting the harddrive too much all at once).
- It took 6 minutes to run 5 visits (104 chips) on 36 cores. This could maybe be a bit faster if we were willing to go to smaller galaxy postage stamp sizes (see below). It's probably possible to parallelize more.

- I've been skipping sources that don't work well in Galsim (high n, elongated, but that needs to be checked on, to make sure we aren't introducing biases, and Song probably needs to send warnings not throw errors and ease up on the restrictions)

- **Extra notes**:

    a. Galsim assumes `Reff` is the geometric mean of the semi-major and semi-minor axes. That's not the definition I used to make the input catalog from COSMOS (I used `Reff=semi-major axis`). We should do this all consistently. I would advocate changing makeFakeGalaxy to assume the given reff refers to the semi-major axis. Talk to Rachel Mandelbaum to make sure we understand this correctly
    b. The size of the fakes mask set by galsim such that 99.5% of the object flux is inside the mask. This makes the masked regions really big for high Sersic-n galaxies. We should probably set a maximum image size of the same order as the `maxMargin` in the positionGalSimFakes configuration (by default 600 pixels, but I imagine 100 (16") is probably enough. Should cull the source list of really big/bright objects from COSMOS before starting this.

Measure fake sources

```
stack.py /lustre/Subaru/SSP/ --rerun=clackner
/testAdd --id tract=0 filter=HSC-I --selectId visit
=1228^1230^1232^1236^1238 --queue small --nodes 4
--procs 9 --job myJob --clobber-config -C config_me
asSrc0 --config doOverwriteOutput=True doOverwriteC
oadd=True makeCoaddTempExp.doOverwrite=True
```

- This command should be done using `multiband.py` , but additional configuration parameters need to be set to **turn off the forced photometry and catalog merging**. Talk to Paul Price about how to do that, as I don't think it's implemented now.
- If you are running pipelines later than 3.7.1, you'll want to prevent the pipeline from measuring large sources, you can tweak the configuation parameter:
    - `root.processCoadd.measurement.algorithms['cmodel'].region.maxArea`
    - Making it smaller (10000 should work) so the pipeline doesn't attempt to measure things with more than 10000 pixels in the footprint, eliminating really bright objects and speeding the processing.
- This runs in 20 minutes on 48 cores. Per patch (121 patches) the break down is (for the 87 non-empty patches):
    a. 2.8 minutes for warping
    b. 2.7 minutes for coadding
    c. 2.4 minutes for processing/measuring
- Part of the slow-down is the pipeline won't proceed until warping is done, which takes 5.7 minutes for the slowest patches. But we are at the point where culling the measurements won't buy much until we also speed up the other steps. It might be true that fewer exposures (~3 in the wide instead of ~5 here) will be faster. I used 5 to get to the expected wide depth (20 min exposure) so the source density was reasonable.
- If I run everything, (with the fake sources in there, but without culling the detection list, it takes 40 minutes, with all of the extra time going to processing/measurements

Extract fake sources from the catalog

```
runMatchFakes.py -t 2.0 -w -c src_0_radec_shear
.fits -o fakesrc_0_5vis -f HSC-I /lustre/Subaru/SSP
/rerun/clackner/testAdd 0
```

- This codes uses the astropy pacakge. If you are on master.ipmu, you can set up against my installation, but otherwise you need to install it (using the python version in the pipeline!)
- This command aggregates all the measured fakes sources from all the patches in the visit into a single file. It does this by an RA/Dec match with the input fake catalog. It's basically optional to run (presumably you just want to extract the shears, which will be faster).
- For a single tract, it takes 8 minutes, and extract 90k sources. This code could use some serious rewrites to be sped up (it's serial, not parallel, it's writing out new output tables, it's saving all the columns, not just the interesting ones...), but it really depends on what kind of output you want.

Example: config_src0

```
from fakes import positionGalSimFakes

root.fakes.retarget(positionGalSimFakes.PositionGalSimFakesTask)
root.fakes.galList = 'src_0_radec_shear.fits'
root.fakes.galType = 'sersic'
root.fakes.maxMargin = 150
root.fakes.addShear = True
#if you want to ignore most ccds (except the central 6), good for testing)
#root.ignoreCcdList=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 43, 44, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103]
```

Example: config_measSrc0

```
from fakes import detectOnlyFakes
root.processCoadd.detectCoaddSources.detection.retarget(detectOnlyFakes.OnlyFakesDetectionTask)
```

Example: config_measSrc0_noCmodel

- Turn off the CModel measurements

```python
from fakes import detectOnlyFakes

root.processCoadd.detectCoaddSources.detection.retarget(detectOnlyFakes.OnlyFakesDetectionTask)

root.processCoadd.calibrate.measurement.algorithms.names=['flux.psf', 'flags.pixel', 'flux.gaussian', 'flux.aperture', 'flux.naive', 'centroid.naive', 'flux.sinc', 'shape.sdss', 'shape.hsm.moments', 'multishapelet.psf', 'correctfluxes', 'classification.extendedness', 'skycoord', 'shape.hsm.psfMoments']

root.processCoadd.measurement.algorithms.names=['flux.psf', 'flags.pixel', 'shape.hsm.moments', 'flux.aperture', 'flux.naive', 'focalplane', 'flux.gaussian', 'centroid.naive', 'flux.sinc', 'shape.sdss', 'jacobian', 'shape.hsm.regauss', 'flux.kron', 'correctfluxes', 'classification.extendedness', 'skycoord', 'shape.hsm.psfMoments']

root.processCoadd.calibrate.measurement.slots.modelFlux='flux.gaussian'
root.processCoadd.measurement.slots.modelFlux='flux.gaussian'
```