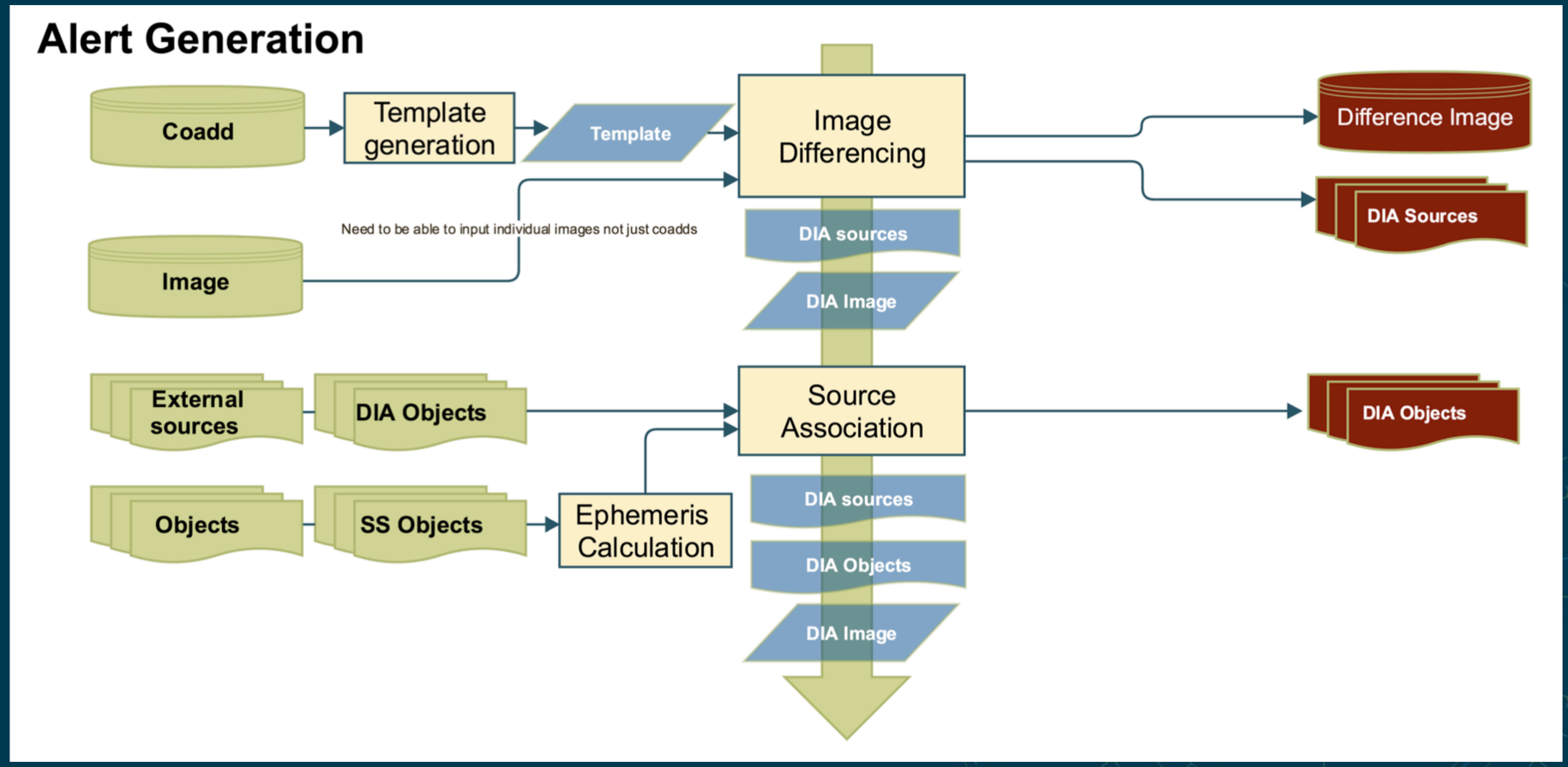
DIA Processing, Testing, and Development

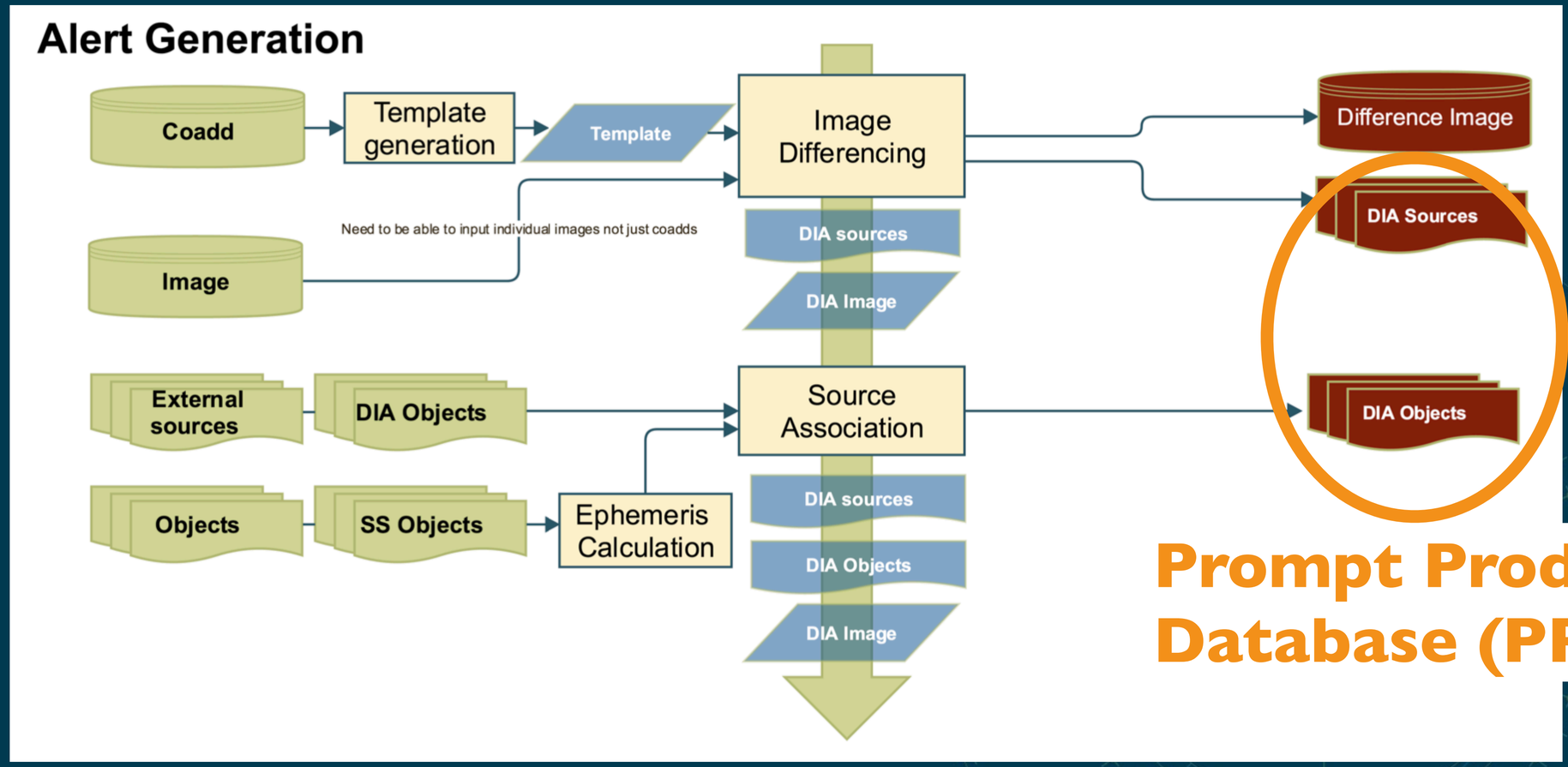
Finding real bumps in the night

@merrdiff
@mrawls

The AP team regularly processes real data from raw images to DIA sources + DIA objects

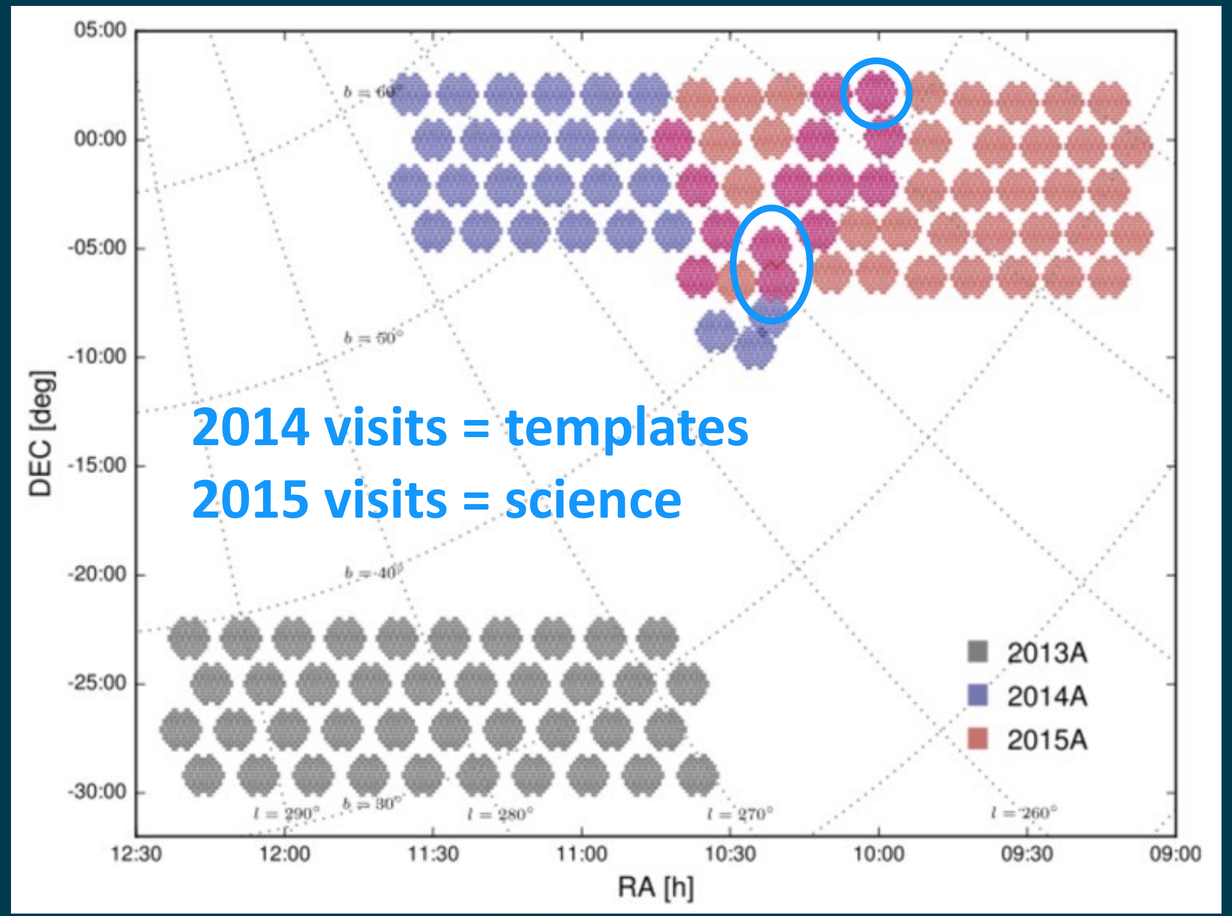


The AP team regularly processes real data from raw images to DIA sources + DIA objects

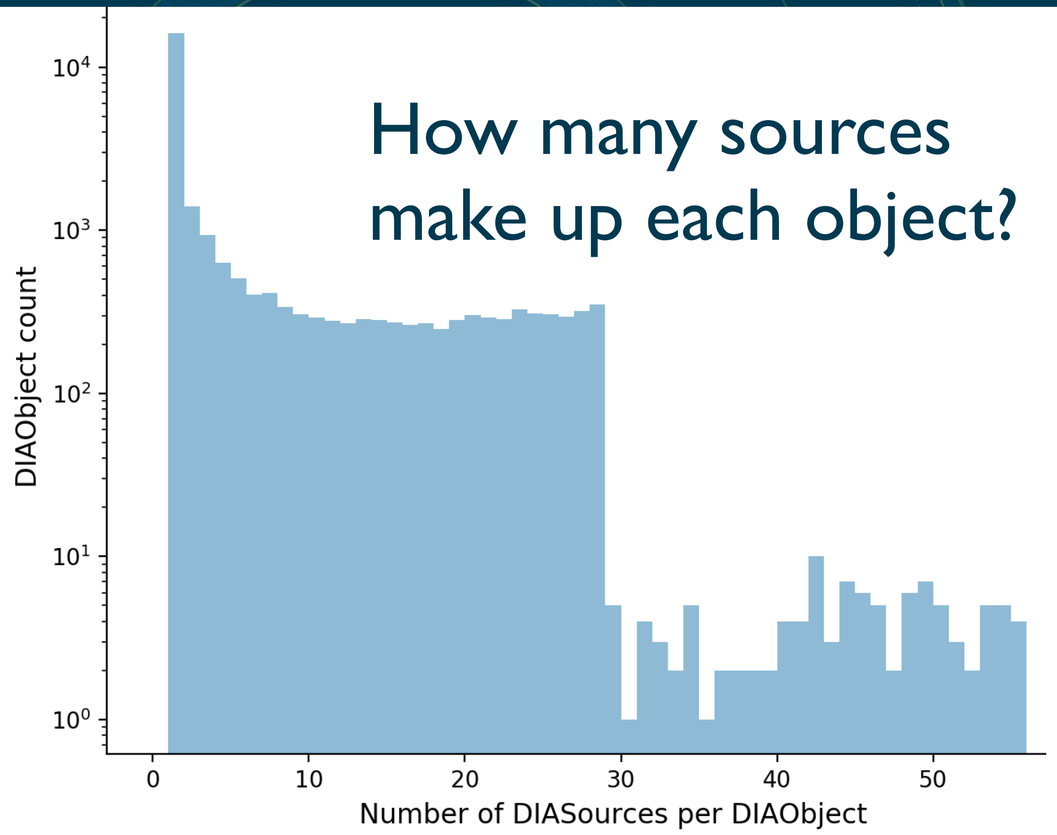


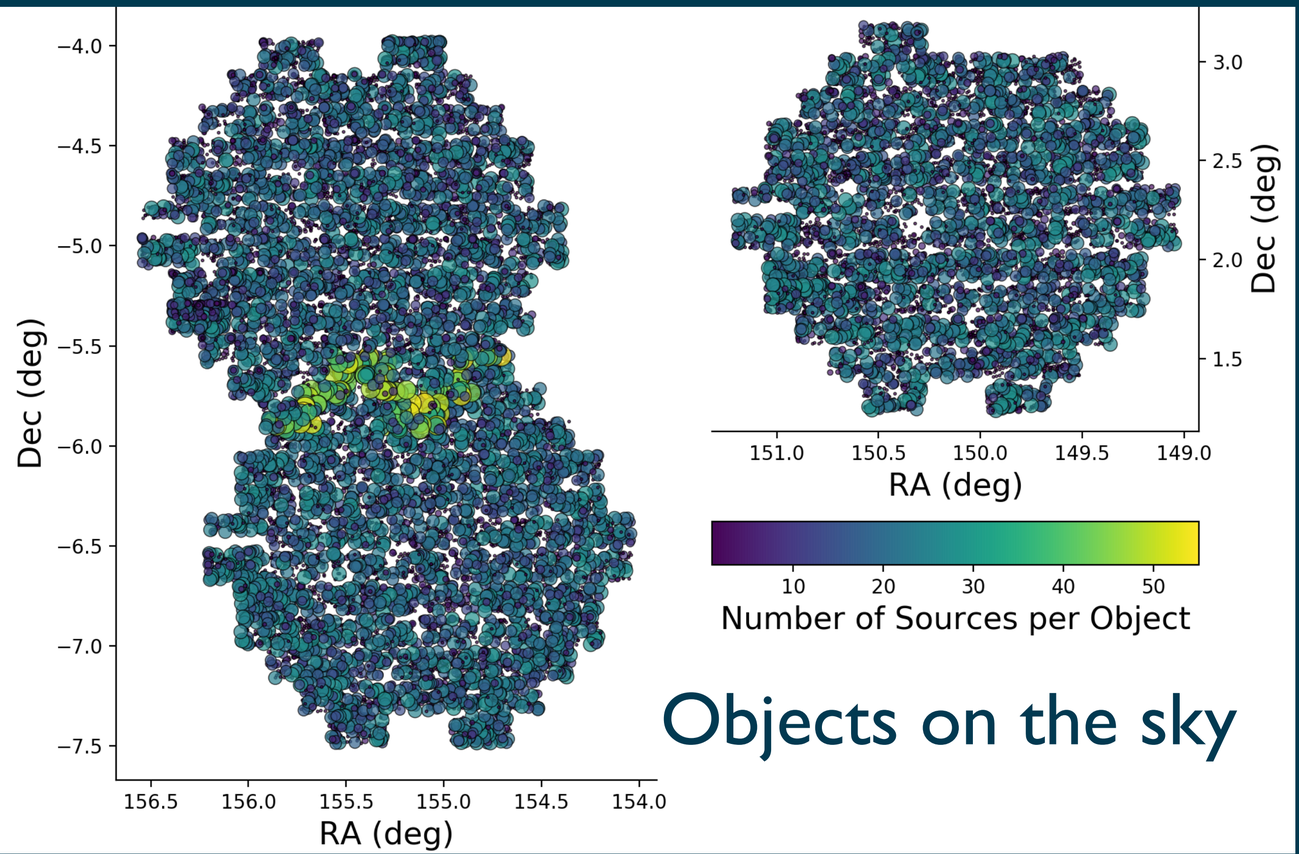
AP test dataset is raw g images from DECam HiTS

- "Static" coadd templates
- Three fields, each with 28 visits and 60 CCDs
- AP-style datasets are setup-able Git LFS repos
  - `ap_verify_hits2015`
  - `ap_verify_ci_hits2015`
  - New: `ap_verify_hsc_pdr1`



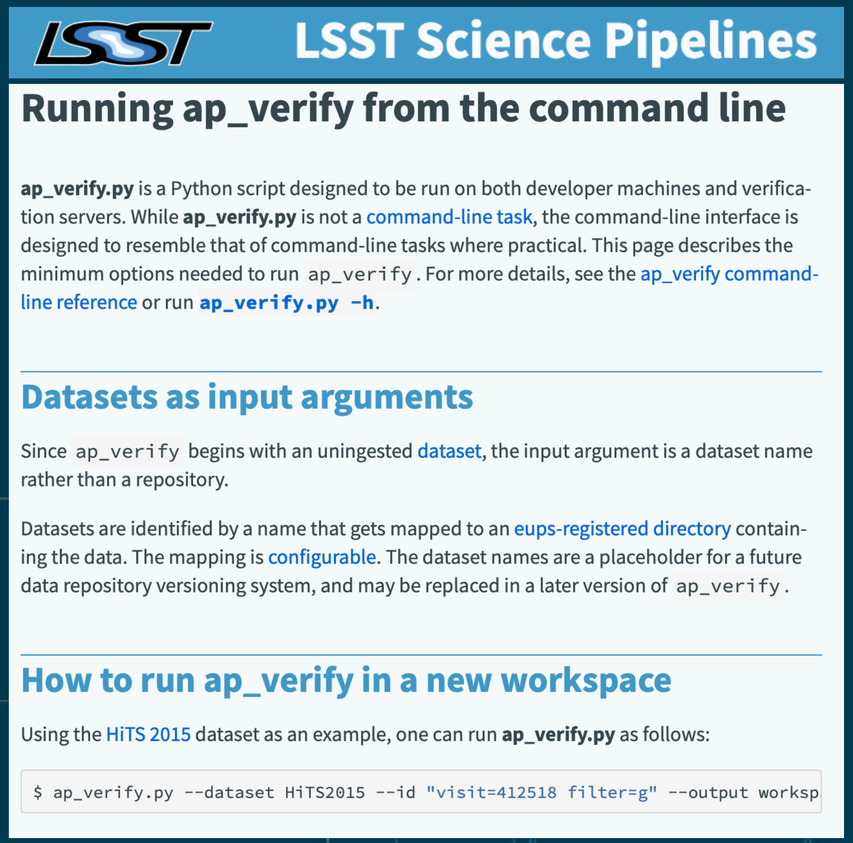
DIA sources associated into DIA objects in the DECam HiTS 2015 dataset





The ap_pipe and ap_verify packages

- ap_pipe does image processing (ISR, background, and calibration), image differencing, and association
- ap_verify runs ap_pipe, collects metrics, and helps developers track performance



LSST Science Pipelines

Running ap_verify from the command line

ap_verify.py is a Python script designed to be run on both developer machines and verification servers. While **ap_verify.py** is not a command-line task, the command-line interface is designed to resemble that of command-line tasks where practical. This page describes the minimum options needed to run `ap_verify`. For more details, see the ap_verify command-line reference or run **`ap_verify.py -h`**.

Datasets as input arguments

Since `ap_verify` begins with an uningested dataset, the input argument is a dataset name rather than a repository.

Datasets are identified by a name that gets mapped to an eups-registered directory containing the data. The mapping is configurable. The dataset names are a placeholder for a future data repository versioning system, and may be replaced in a later version of `ap_verify`.

How to run ap_verify in a new workspace

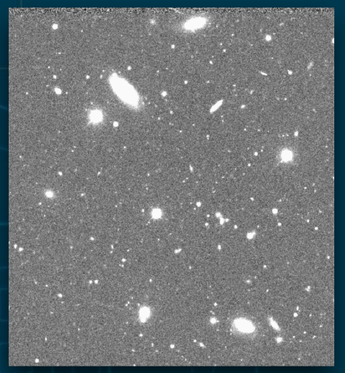
Using the HiTS 2015 dataset as an example, one can run **ap_verify.py** as follows:

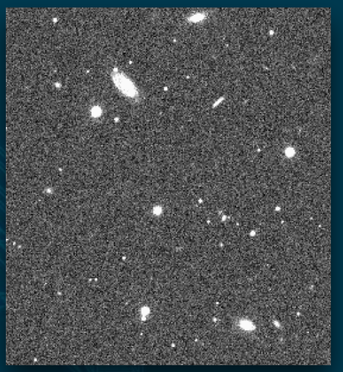
```
$ ap_verify.py --dataset HiTS2015 --id "visit=412518 filter=g" --output worksp
```

Try it: https://pipelines.lsst.io/modules/lsst.ap.verify/

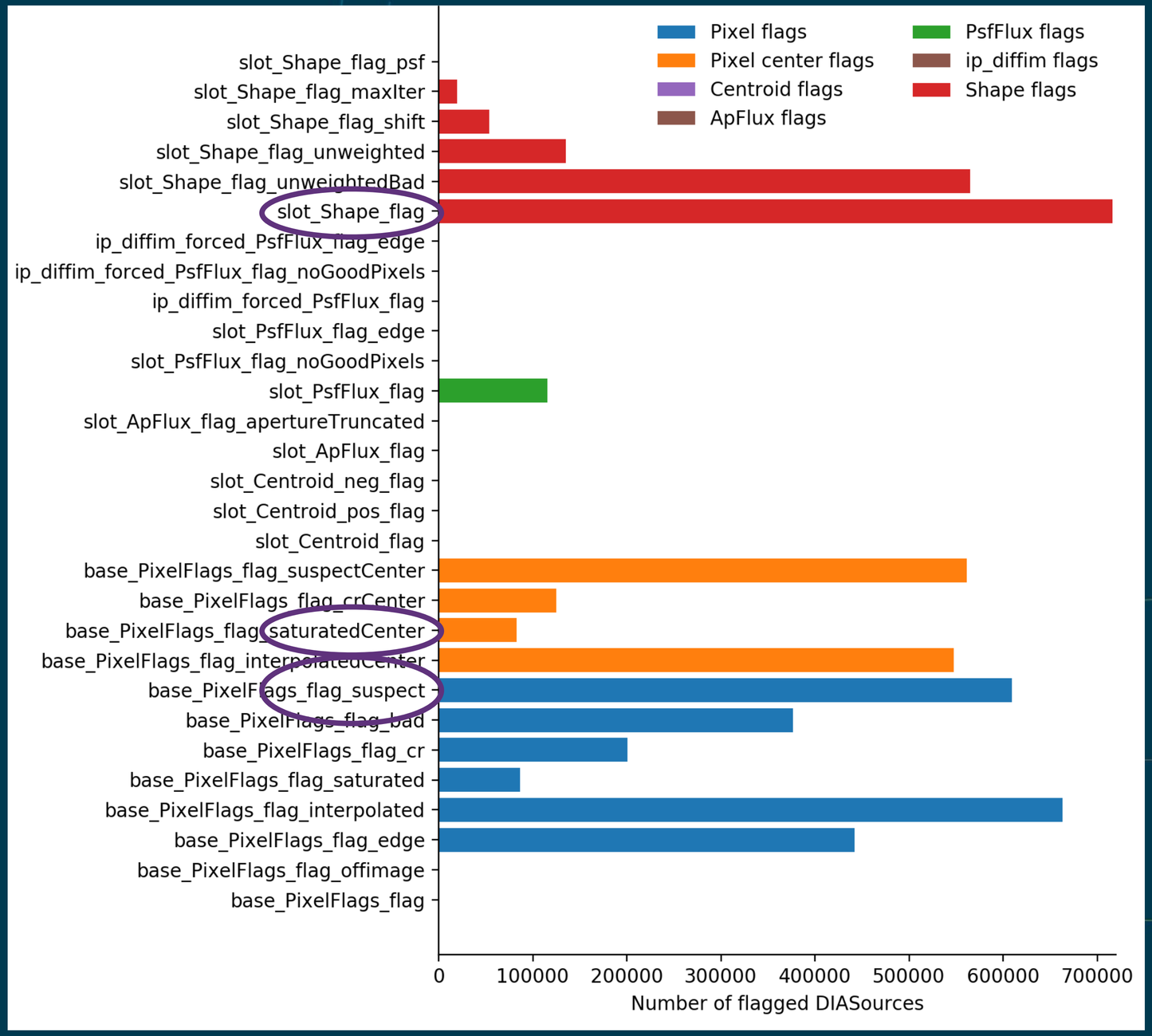
Quantifying false positives in real difference images

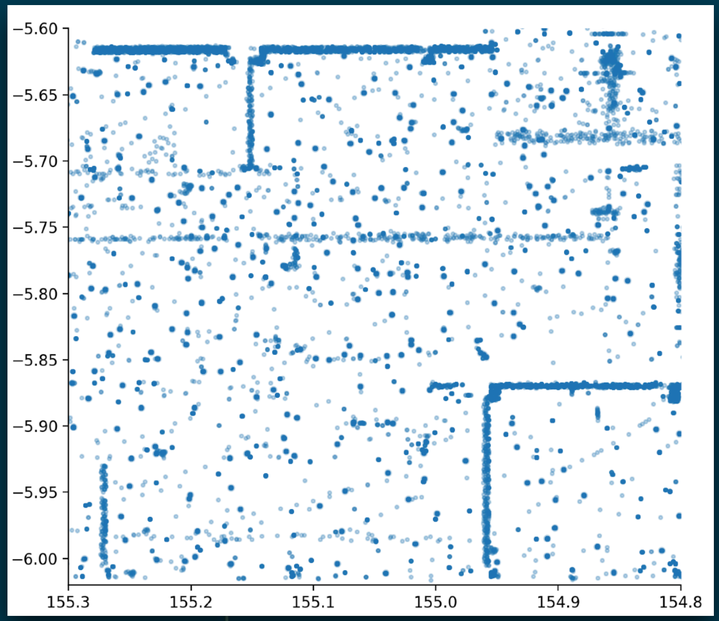
- There is no truth catalog 😱
- Different templates yield different results
  - Direct CompareWarp coadds with good seeing (similar to what AP will use in operations)
  - A single processed visit image
- Make some first cuts with flag information 🚩🚩



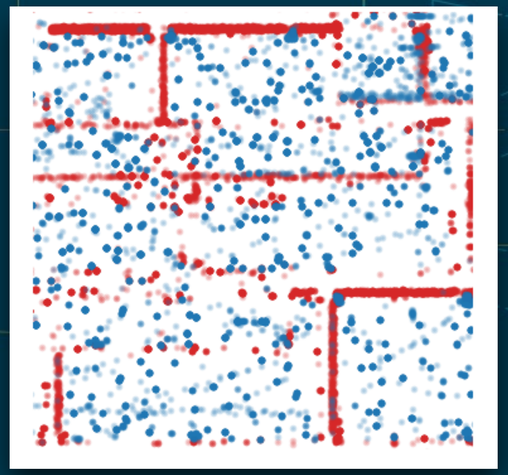


Coadd template

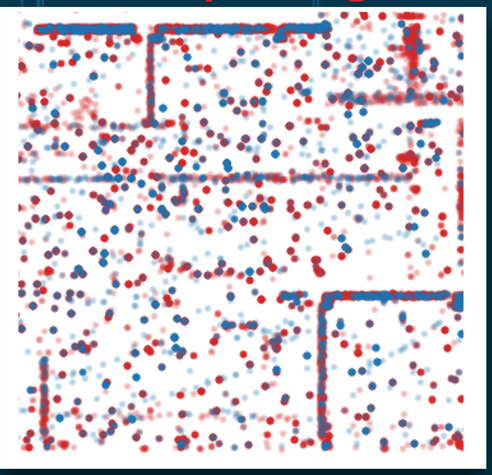




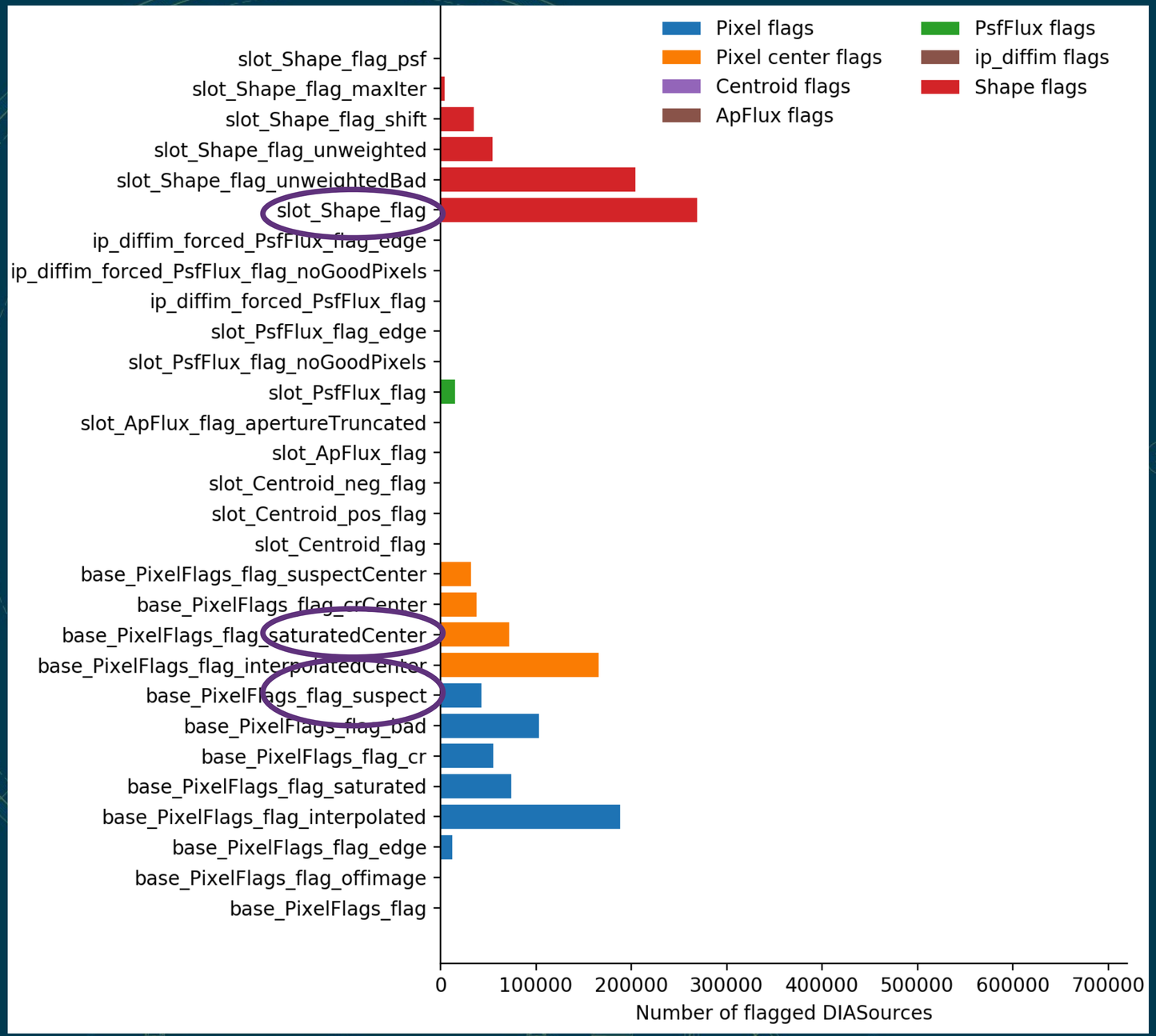
Suspect flag

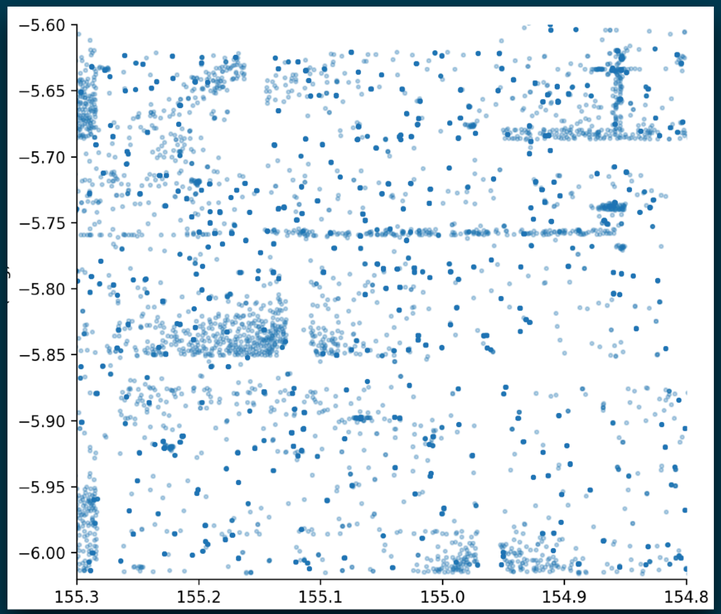


Shape flag

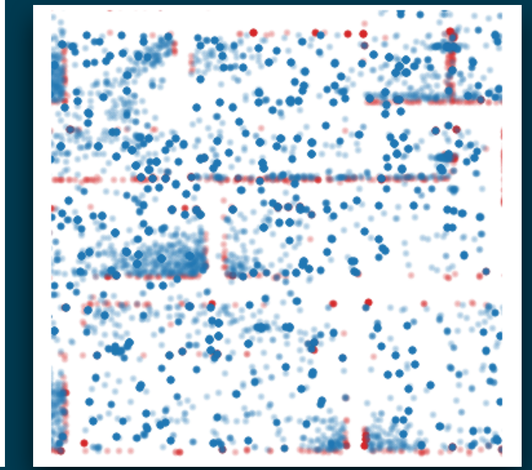


Single visit template

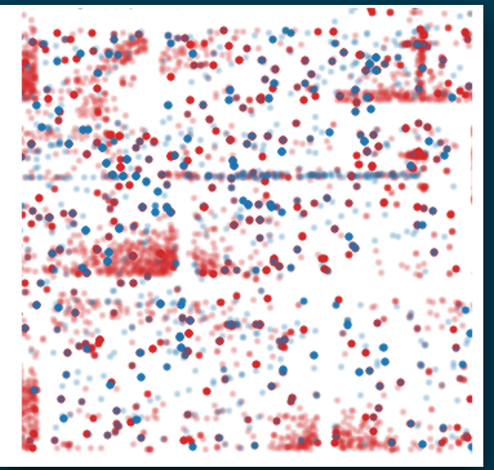




Suspect flag

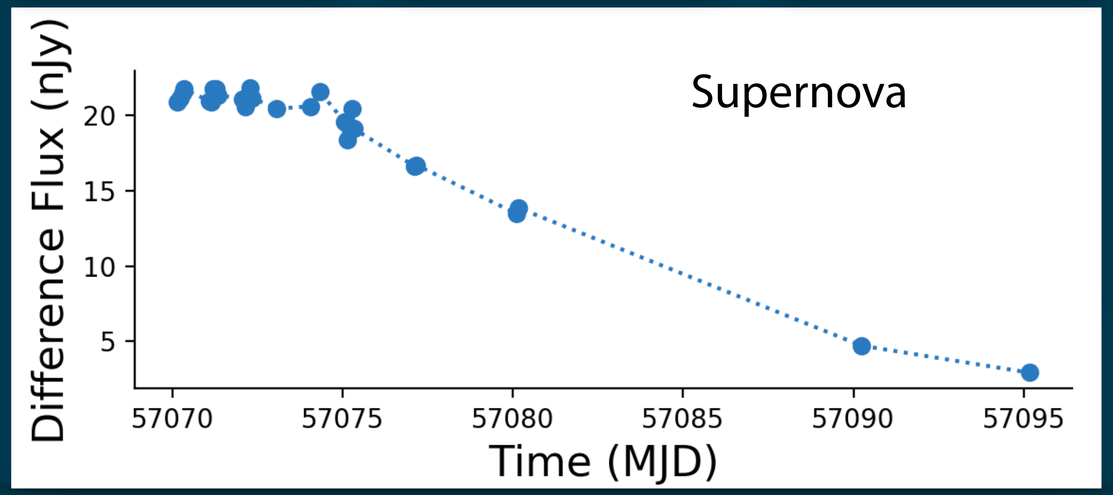


Shape flag

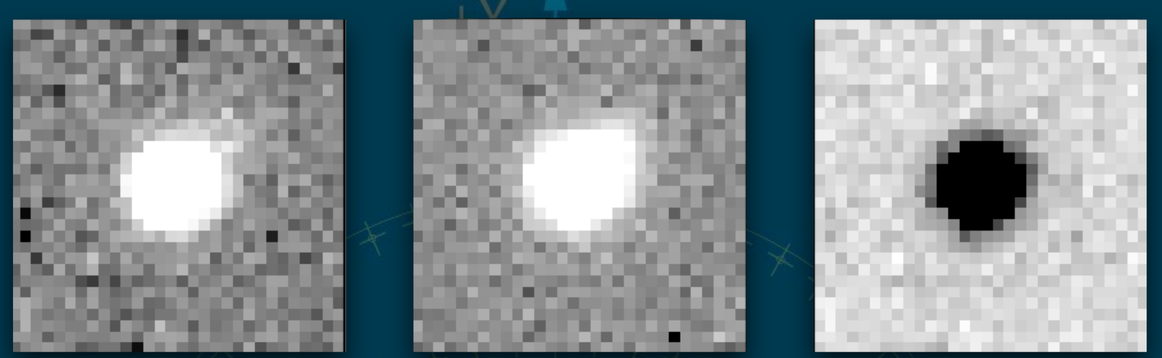


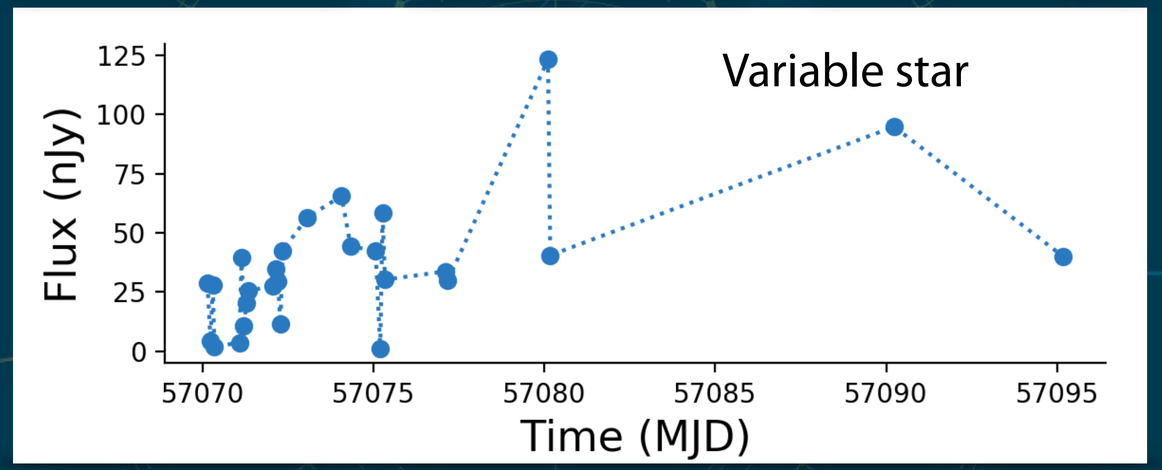
Constructing light curves from data in the PPDB

Processed Template Difference

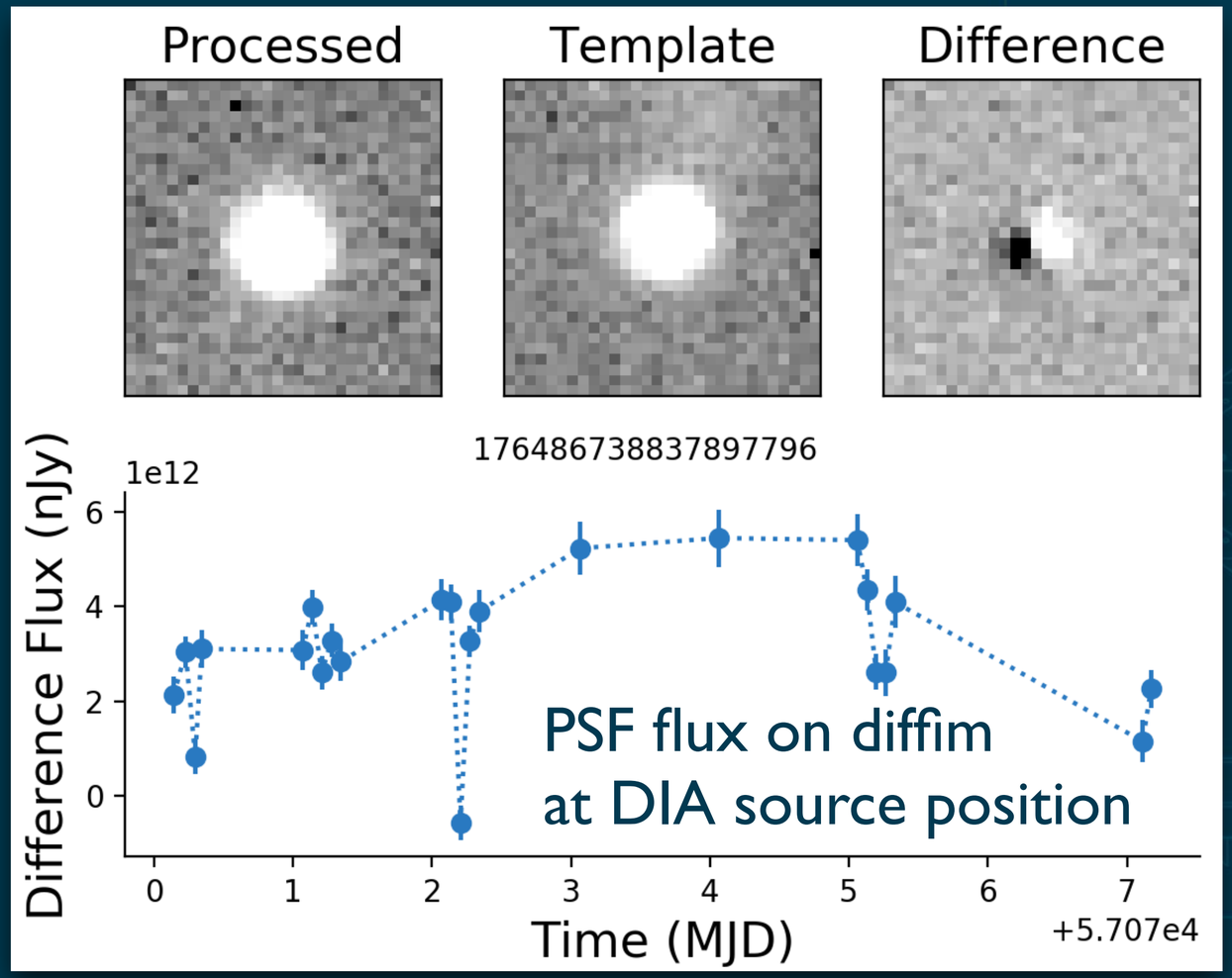


Processed Template Difference

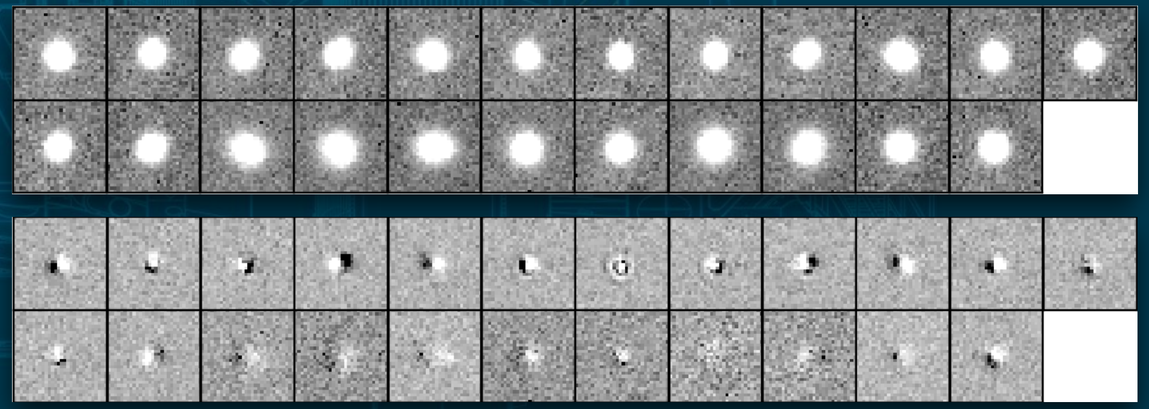




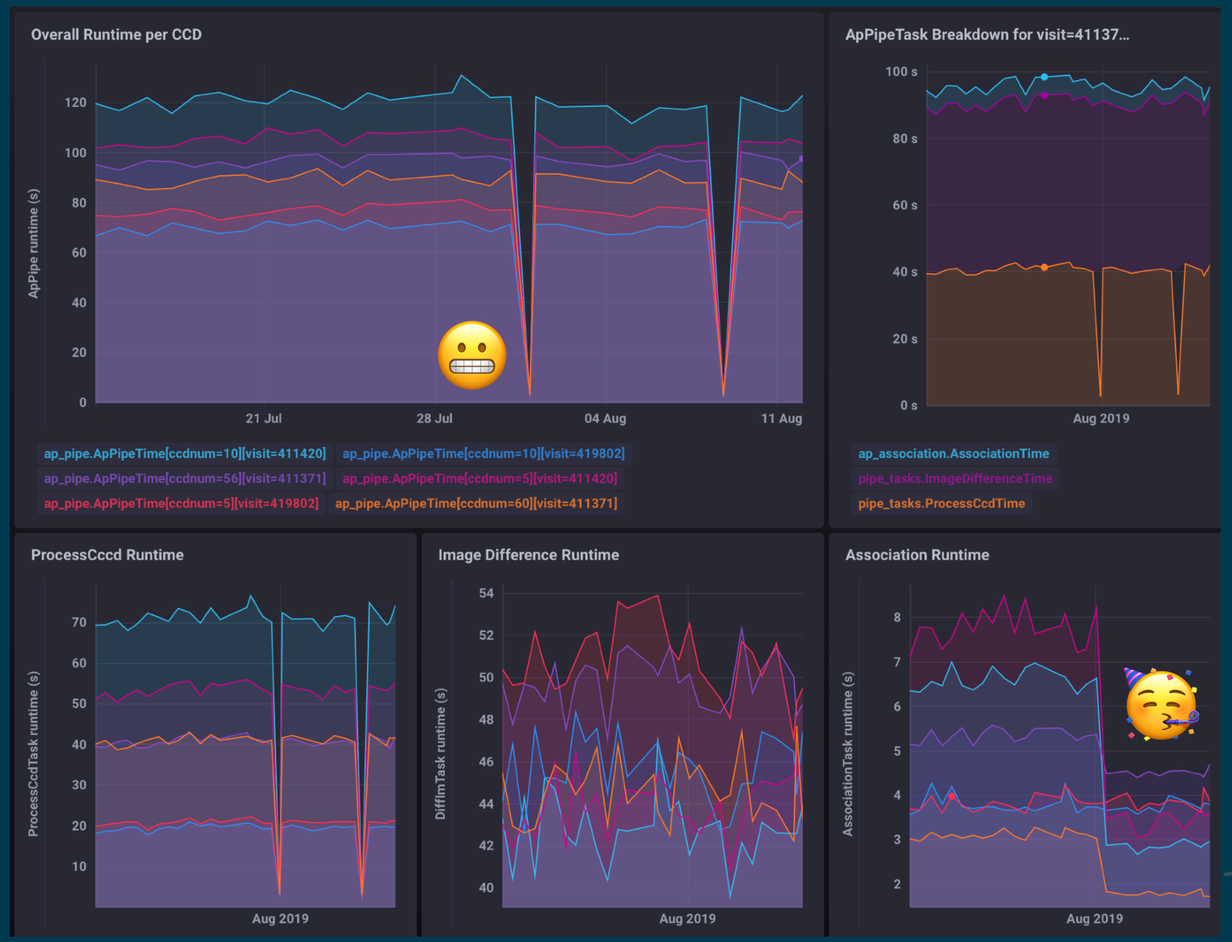
Constructing light curves from data in the PPDB



- Flux quality strongly depends on PSF and astrometry
- psFlux, apFlux, totFlux
- I can help if you want to do this!

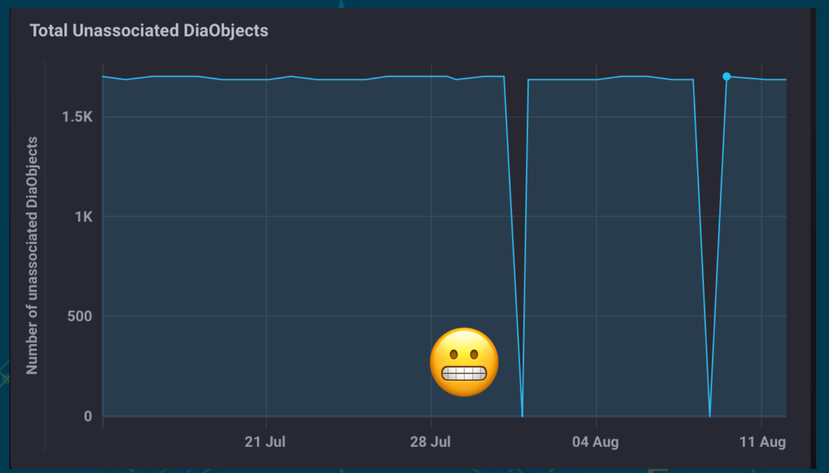


Visualizing ap_verify metrics with chronograf



Runtime broken down by Task

https://chronograf-demo.lsst.codes



Tracking unassociated DIA objects

Your favorite DIA metrics here

We can find real variable sources with DIA, and much more is coming soon™

- Track more **metrics** and automate clear diagnostic **plots**
- Regularly process non-DECam data in **CI**
- Compare findings with **known** variable sources
- Inject and recover **fake** variable sources
- Improve template coadds with **DCR** corrections
- Compare **ZOGY** differencing with A-L
- Include the ability to handle **moving** objects
- Create prototype **alert** packets