ZTF & Fakes

The goal of this document is to scope out both the type of person needed and how much work they will be doing, as well as the effort required by IPAC and the ZTF team, for someone to work on injecting fakes into the ZTF pipeline and to recover them in order to best measure the efficiency of the discovery and follow-up pipelines. As reference everyone should understand what was done in the Frohmaier et al. paper: http://adsabs.harvard.edu/abs/2017ApJS..230....4F

So the skill set of someone doing the fake injection is, roughly speaking, knowledge of shell scripts, Python, C, and Perl. These are the primary languages spoken by the IPAC folks. It does not need to be complete, but it should be good enough to understand and diagnose issues that come up with running the pipeline. They will also have to be familiar with astronomy, and in particular image processing and analysis. They can certainly be a good postdoc or graduate student - a la Chris Frohmaier.

The next thing is that someone from IPAC will have to teach this FTE how to run the pipeline from end-to-end. By the time they are good to go, they will understand exactly how the subtractions are done, candidates are loaded into the database and machine learning is run to get their RB scores. They do not need to know how the image processing is carried out. Basically everything for them starts after a calibrated image (astrometry and photometry) is produced.

Next up is that they need to add fakes to an image. The code is available from Chris (Uli has a copy) and some studies will have to be done to ascertain how many fakes per image can be placed before one starts to distort the image too far from its original state. You also need to know how you are going to place the fakes down (i.e. following the light on potential host galaxies as well as dropping them on blank patches of sky) and how many iterations on a particular subtraction one needs to run in order to get the statistics up to develop your efficiency curves. If you want to do stars, this could be done with Chris' code as well, but streaks would be a separate thing. I would use all the images you can (under as large a set of sky conditions you can get during the engineering phase) with pointings both at low and high galactic latitude.

Now you have to keep track of the fakes you injected. Here we created flat files (sqlite) for each image we added fakes to. Then we basically replicated the candidate database, pushed everything into it, ran the RB code, and then sucked in the sqlite file of injected fakes into a fakes table in the main db. It was then a simple cross-match q3c query to determine which fakes were found and which were not per subtraction.

Next up is creation of the efficiency hypersurface. This is a non-trivial process and access to some good statisticians is key to making this manageable. We pulled in everything we could from the candidate and subtraction tables in our database (mag, seeing, surface brightness under the candidate, sky background, moon separation and illumination, etc.). We started off with a dozen and got it down to six which were important. Along the way we realized we should have done things differently in the PTF pipeline in terms of what we stored. Surface brightness under the candidate was not stored in the best way… This hypersurface is then the % efficiency of finding a transient given a nominal RB on a galaxy of a certain

brightness given certain sky conditions, etc. The way we used this hypersurface was via interpolation. However, if you are feeling lucky, one could use ML on this as well….

Of course I would use this to train the RB, so this is when you want to involve these folks….

Finally from here you can use this to create a set of supernova lightcurves via MC's across a set of images. The cool thing about separating this part of the problem out is that for a given field/chip combo you can inject thousands of supernova which would be impossible to do with real images (or at least it would be ***very*** painful).

It took Chris 2 full weeks with me at his side the entire time to learn my pipeline. He had the CS and astro background before to step into this easily. It took him 2 months to write the code (which we have now in Uli's hands), two weeks for he and I to run several hundred thousand subtractions and another 2 months to diagnose the proper hypersurface to use to measure efficiencies.

I'd estimate that this could be done in 4 months. Should only be started ***after*** images, worthy of making subtractions, start to come in on a regular basis.

Notes / thoughts from Frank:

1. First, the earliest that development (and integration) could start is probably March 2018, when the baseline system has processed (or is processing) a significant amount of science-survey data and hopefully been shaken/tuned to a steady-state.
2. I want to make this fit seamlessly into the Data System, with minimal disruption to the existing architecture, DBs, and archive infrastructure. It is way too risky to re-engineer the system at this point and certainly not when we have commenced science operations. The points below suggest a design that modularizes and streamlines this functionality, at least at IPAC, independent of the production pipeline. The basic idea is that additional file-based products can be generated and consumed by the collaboration to enable efficiency/rate analyses as a function of any variable (see below).
3. The existing realtime pipeline will have a switch to turn on/off the injection of fake events. There will be two additional (required) inputs when "fake mode" is invoked: **(i)** an ascii (master) table of lightcurves: names, positions, filters, field, ccdid,...; I already have the specs for how such a table would look like; **(ii)** an output file that will contain a table of the injected fake event IDs + simulated properties, and beside each one, all the source features and RB scores that were computed/extracted from the subtraction image if indeed there was an extraction on/near that fake event. This output file would also show any new extracted events and their features, RB scores etc. Fake events with no extractions would simply have "nulls" in their feature columns. As you see, this design implicitly does the cross-matching. There is also an ancillary file that lists all the image-based features and survey metrics known at its observation epoch.
4. The above pipeline (in "fake mode") would be executed using the IPAC compute cluster during the day, meshed with other processing tasks. Presumably, it would execute on a preselected list of

fields, ccdids, and exposures spanning some time window, i.e., primarily on what is provided in input file 3i described above. This execution will generate the output files (3ii above) and make them available to the collaboration for analysis. This does not touch (write to) any database on the IPAC side.

5. The file products described in 3ii contain all the information for further filtering, thresholding, lightcurve reconstruction, and efficiency/rate analyses. This would be the same information present in the alert packets, but without associated 30 day histories. Generating avro-formatted alert packets from the files in 3ii (with self-contained 30 day histories) would be more complicated because it will require interfacing with a new DB, primarily a "fake DB" as to not contaminate the ops DBs. This is where the real cost is, if we want to push it this far that is. As you see, I'm treading on eggshells.
6. The only new code required at IPAC is the fake-event image-injection code according to the input spec briefly outlined in (3i). **This is the bare minimum to get things going.** The other coding part is generating the output proposed in 3ii. This is what the current pipeline is wired to generate anyway. The latter would be simple and requires little effort by an IPACer (probably me). For the IPAC I/O interfacing and coding alone, I envisage this would take ~1.5 to 2 work months involving a collaborator (Chris F. or other) and an IPACer.