Cross-Matching Surveys

Cross-matching surveys

- Aspects of mining data from large surveys are rapid
  - consider the *BETWEEN* queries for which we applied SQL in the previous lecture
  - or the *fGetNearbyObjEq(α,δ,θ)* query which returned objects in a "circle" (or radius θ around a specific coordinate)
- But, a particularly useful type of query on data is also one of the slowest to apply...cross-matching
- Cross-matching takes a (potentially long) list of (α,δ) positions in one sky survey and finds the closest set of objects at (α,δ) positions in a different survey
  - or, for that matter, in the same survey

Cross-matching surveys

- One goal of data mining would be to obtain a flux measurement for a source (a star etc.) at any position in any sky survey (at any time and/or wavelength)
- In the era of cloud computing, the grand vision is a set of web services, where any user can send code or a query (e.g., SQL) to a massive central database
  - that database would then return requested data, or run code on that data for a science application
- But, in truth, computer processing remains expensive, while disk space is cheap. So, it still makes a lot of sense to store local copies of data and run code locally
  - In today's tasks, we'll consider both a web-services-type query, and a local-style query

Python tasks

1. At */d/quasar2/FIRST/first_08jul16.fits* there is a file containing sources from the VLA FIRST (20cm radio) Survey that lie in the SDSS area (as of July 2008)
   - read it using *fitsio* (see syllabus link) and plot (α,δ)
2. In our SVN repository in my *week7* directory there is a file called *sdssDR9query.py* that can be used to query the SDSS database with SQL remotely over the web
   - try the example query provided in the code header
   - try a few random RA and dec (α,δ) positions
   - using the *SDSS Navigator Tool* linked from the syllabus, find the RA and dec of an object that exists in the SDSS and pass *sdssDR9query.py* that position

Python tasks

3. Write Python code that reads in the FIRST radio data, takes the coordinates of the first 100 objects and uses *sdssDR9query.py* to obtain the SDSS optical data
   - note how slow this is (web services are limited to 1 query per second so as to not overburden the server)
   - note that objects that are bright in the radio do not necessarily have matches in the SDSS optical data
4. Let's perform the query locally. */d/quasar2/dr8/301* contains popular measurements from the entire SDSS
   - Note we have the DR8 version of the data, which can differ slightly from the DR9 tools we have used so far
   - read in one or two of these files and examine them

Python tasks

5. These local SDSS files are called *sweep* files (more information about the sweep files is linked from the syllabus). One (slow) method to cross match FIRST and SDSS would be to read in *all* of the sweep files
   - note the two main sweep files: "*gal*" files which are objects that are extended in imaging and "*star*" files, which are objects that are point sources in imaging
6. An index file prevents us needing to read all the sweeps
   - In my week7 directory in SVN there is a file called *sdss_sweep_data_index.py* which can calculate the subset of sweeps files that intersect regions of the sky
   - use this code to calculate which sweep files would be read to find objects within 0.5^o of $(\alpha,\delta) = (180^o, 45^o)$

Python tasks

7. Note that you can pass *sdss_sweep_data_index* a large array of RAs and decs, instead of just one position
   - use *sdss_sweep_data_index* to find the files needed to cross-match the first 100 objects in the FIRST radio data...note that *sdss_sweep_data_index* defaults to listing the "star" files (what we want it to do)
   - use *astropy's search_around_sky* to return SDSS matches to the FIRST objects (you'll find many objects, as the sweeps contain *every* SDSS object, not just "primary" objects...more on that in the next HW)
   - for 100 objects, this method likely isn't quicker than using the web services approach, but, relatively, it will be *much* quicker for larger numbers of objects