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(\alpha, \delta, \theta)* 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 often still makes sense to store local copies of data and run code locally
 - In today's tasks, we'll consider both a webservices-type query, and a local-style query

SDSS Sweeps

• In SDSS, the data "sweeps" are locally stored trimmed versions of photoObj containing the most commonly used parameters.

- e.g., RA, dec, *ugriz* magnitudes, and more

- Each SDSS field can be uniquely specified by three numbers: *run, camcol, field*. See the SDSS Imaging link.
- You can follow the SDSS DRD15 data link to see the sweeps. These are stored locally on tomservo, so we will use them to work with SDSS photometry.
- To find photometry for one source, you could (slowly) read in all the sweeps or...

SDSS Sweep Index

- There are index files that contain arrays of RA and dec to mark the center of each sweep across the sky. The margin on the index width is 0.36 deg.
- To get photometry near the position in cyan, nearby index values (black) suggest searching 11 different sweeps (color).



- 1.We'll be working with SDSS DR15, which is too much data to store locally.
 - Our data directory is /astro/astr8020/ on tomservo
 - du h ./dr15/ will print the data directory size.
- 2. You will need to clone a version of the git repo to your vpac account (ask about this if you're not sure):
 - *ssh -XYl username tomservo.phy.vanderbilt.edu* (gives vpac home with access to /astro/astr8020/).
 - create a location to work, e.g., ~/repos/ASTR8080/; cd repos/ASTR8080
 - git clone username@vpac01.phy.vanderbilt.edu:/ home/runnojc1/repos/bare/S25/ASTR8080/./

Some logistical notes

1. You'll need to install conda on vpac or use mine.

- To use my conda, in your .cshrc or .bashrc file put: (csh) alias sconda "/home/runnojc1/anaconda3/etc/ profile.d/conda.csh"
 (bash) alias sconda="/home/runnojc1/anaconda3/ etc/profile.d/conda.csh"
- You may also need to add the following to your path: /home/runnojc1/anaconda3/envs/astr8080/bin
- Then type: *sconda*; *conda activate astr8080*

2. You can alternatively install Anaconda yourself. Run:

• bash /astro/astr8020/Miniconda3-2024-Linuxx86_64.sh

Some new rules

- 1.No Jupyter notebooks. You can't* run them from the command line on a remote machine.
- 2. Don't save big files to the *Git* repo.
- 3.Don't write to /astro/astr8020/ or /home/runnojc1/ repos/bare/S25/ASTR8080/.
- 4. You may work in your *ASTR8080/username/week** directories on your vpac account or on your local machine. Treat each machine as a separate user with *git add, git commit, git fetch, git status, git pull, git push.*
- 5.I recommend writing code locally, running code that uses big data on tomservo, and then doing analysis on the results locally.

- 1.At /*astro/astr8020/FIRST/first_08jul16.fits* there is a file containing sources from the VLA FIRST (20cm radio) Survey that lie near the SDSS area (as of July 2008). Read it using *fits* and plot (α , δ) to *png*.
- 2. In our *git* repository in my *week7* directory there is a file called *sdssDR15query.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 *sdssDR15query.py* that position

- 3. Write Python code that reads in the FIRST radio data, takes the coordinates of the first 100 objects and uses *sdssDR15query.py* to obtain the SDSS optical data
 - Code that runs at the terminal can be run from Python using *os.system*, so you could call the example in my code as follows (after *import os*):
 - os.system("python sdssDR15query.py 145.285854 34.741254 >> file.txt")
 - the query results would then be written to *file.txt*
 - note how slow it is to match 100 objects in this manner (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". /*astro/astr8020/dr15/* contains popular measurements from the entire SDSS
 - download one from SDSS DR15 Data to examine
 - you can use *fits.open()* to read a FITS file, even if that file is gzipped
- 5. These local SDSS files are called *sweep* files (read about the sweep files in the syllabus links). 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

- Read the index data model linked in the syllabus, read in datasweep-index-star.fits, plot RA and dec. Compare this to the figures earlier in the notes.
- In my week7/ find *sdss_sweep_data_index.py* which calculates the subset of sweeps files that intersect regions of the sky.
- Import this and use it to calculate which sweep files would be read to find objects within 0.5° of (α,δ) =(180°,45°). Make sure you understand what this code is doing.
- Note that you can pass *sdss_sweep_data_index* an array of RAs and decs, instead of just one position

- 7.Use *sdss_sweep_data_index* to determine which files are needed to cross-match the first 100 objects in the FIRST radio data and match to those objects
 - note that *sdss_sweep_data_index* defaults to listing the "star" files (which is what we want, here)
 - 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

Some useful commands:

- ssh -XYl username tomservo.phy.vanderbilt.edu
 - X enables X11 forwarding
 - Y enables trusted X11 forwarding
 - 1 uses specific username syntax
 - Can also ssh into e.g., <u>vpac01.phy.vandebilt.edu</u>.
- du -h filename to print size of a file
 - h prints in human readable format
 - You can also use this on a directory.

Some useful commands:

- rsync -auvzn --progress ./download_dr15.csh <u>runnojc1@vpac01.phy.vanderbilt.edu</u>:/home/runnojc1/
 - a archive, makes it recursive
 - u update, don't overwrite the target file if it is newer
 - - v verbose, -z zip
 - –-progress, show progress
 - n dry run (remove to actually move files)
- screen
 - ctl-a ctl-d to detach. In ssh on Mac, detach to leave process running. screen -r to return.

Some useful commands:

- wget -r -N -np -e robots=off exclude_directories ='skip_this/' -nH --cut-dirs=2 --spider https:// dr15.sdss.org/sas/dr15/eboss/sweeps/dr13_final/
 - --spider, dry run remove to download
 - -e robots=off, ignore the robots.txt file that tells search engines to ignore these files
 - -np, never descend above the parent directory
 - -r, recursive
 - -N turns on time stamps