These are the main sub-projects we need to tackle to really get to the bottom of gene conversion, doing "maximal" dataset comparisons, taking intersections of gene sets only when necessary, fully integrating pathway and gene information, etc. These modifications will lay the actual groundwork for when we start thinking about multi-omics and multi-species stuff. See the "in the longer term" section on the third page, too.

*-**Nuch***

Modification of CCGroup and CCDataSet for cross-format flexibility:

- *Note: See /data3/darpa/calcom/geo_data_processing/ for examples of h5 files which keep all of the raw information of the GEO datasets – e.g., "ccd_gse20346.h5", "ccd_gse61754.h5", etc. The ones that have the suffix "_geneid.h5" have undergone the conversion and don't keep all the original data from the GEO dataset.*
- These CCDataSets for **single datasets** (GSE73072, GSE61754, etc) need to keep track of some extra information that they don't currently, that will allow us to join datasets together "just-in-time" in the CCGroup:
  - The **type** of *omics of the data, e.g., "transcriptomics", "metabolomics", "proteomics" (nearly everything we're working with from GEO is transcriptomics as far as I'm aware – this is just for future-proofing right now. **(let's call it "*omics_type*" for now)
  - The **format** of the variables, e.g., "geneid" (GeneID), "illuminaid" (IlluminaID), "affyid" (AffyID). I don't know if this has any relevance outside of transcriptomics data, but it could always be left as an empty string or something. **(let's call it "*omics_format*" for now)
  - Ensure that the **species** is defined, both in name (e.g. "homo sapiens") and in the Taxon ID (e.g. "9606")
  - Note: to make this easy to incorporate, these pieces of information might just fit in the "study_attrs" dictionary that already is cleanly handled by the load/save functions in CCDataSet. All that would need to be done is to make a new CCDataAttr for each of these and add it to that dictionary. This was originally intended as a container to put attributes common to all data only for efficiency reasons – but it's appropriate for our current purposes too.
- The CCGroup needs to be modified so that decisions about conversions, gene intersections, etc, are made on the fly:
  - In the short term: given a list of ccds in the current structure to merge, check each of their ccd.study_attrs['omics_format'].value.
    - If these **don't** all agree, reference the database described below, and make decisions about how to convert formats. In the easiest case, for each gene, we'd basically search our gene database for that number, and look at the corresponding GeneInfo.geneid. Default to converting everything to GeneID format (if it isn't already).
    - Once the conversions are done (if necessary), then all that's needed is to take the intersections of genes – which we can currently handle in the code.
  - Feature sets collected on the CCGroup level need to keep track of some extra metadata associated with them – which CCDataSet they came from, and what the corresponding *ccd.study_attrs['omics_format'].value* is.
  - I might be forgetting about some other things.

Construction of a database for relations between gene formats, pathways, etc, which can be constructed using queries to *bioDBnet*:

- See the file calcom/io/gene_info.py for the kind of thing I'm looking for – you can run this file directly and check the source - the *if __name__=="__main__"* section has two example calls to this class.
- Basically, I've worked out how to make the REST calls (is this the right word?) to get at the information we need. I take the raw XML output and use regular expressions to pull out (most) of the information we want, with a couple bugs I already know about (see the "TODO" in the __init__() of the class). I assume there's a better way to get the fields of this XML page, if either of you know how to already, or want to look it up.
- The big question I have right now: where can we get at a "master list" of GeneIDs? We could probably look at https://biodbnet-abcc.ncifcrf.gov/db/dbOrg.php as a starting point but I don't know if this is exhaustive.
- The pieces of information we're interested right now, for a given gene (some have been implemented already):
  - All other "names" it has in other formats. For example, in that script, the object *g1* was initialized with IlluminaID "ILMN_2416019", and we know through the query that this is also known as GeneID "377841". This is currently buggy – for an input gene, cases where there are multiple output genes isn't handled well.
  - All pathways associated with that gene. This functionality is mostly set up – but might want to check the __init__() of the GeneInfo class to see if anything here is on the TODO list.
  - A longer description of what the gene does. This is pulled from the raw HTML of https://www.ncbi.nlm.nih.gov/gene/ for that gene, from the "Summary" section. See for example https://www.ncbi.nlm.nih.gov/gene/1234 where the summary begins "*This gene encodes a member of...*" This is pretty inefficient – this isn't a very small webpage – but I couldn't find any faster way of getting at the summary data only.
  - Anything else on the TODO list in the __init__(). **These are ones that haven't been implemented.**
- We want to have this database be searchable, in the sense that there is something like genedb.search('1234') and it checks for an **exact match** of all the GeneInfo objects contained within. Optional arguments (**kwargs) to this genedb.search() would just be *field* (defualting to "geneid") for now. Returns a list of GeneInfo classes of hits. I don't know what we should do if multiple are returned.
- **We want to save this to disk** so that we can work off-line. We should probably just save it as another HDF file, but it doesn't make sense to use the CCDataSet format for this stuff. It's up to the designer to put this into an HDF file in a way that makes sense.
- In a sentence, we're basically trying to recreate a graph subset of the network on https://biodbnet-abcc.ncifcrf.gov/dbInfo/netGraph.php with the pieces of information we care about.
- A side note - the bioDBnet server is really fragile! Even trying to make 2-3 queries at the same time, or repeated queries quickly, I've gotten a large amount of errors thrown (the server throws an error instead of serving the actual XML page). So there needs to be some degree of fault-tolerance in the code (there is a little bit already). Some kind of check needs to be done to differentiate between an error page, and just an empty page (a given gene might not have any other information associated with it through that website).

In the longer term:

- In the long term, we'd also like the following:
  - CCGroup is its own HDF file, with all our current CCDataSets for this task stored within (as "subfolders" in the HDF file – funny enough these themselves are called "datasets" in HDF parlance).
  - All necessary metadata for converting formats, taking intersections, etc, is contained in the CCGroup. This includes the gene database described above.
  - The CCGroup is a monolithic file on the dataset which is meant to be relatively static – quasi-read-only, where pretty much the only thing we'd like people to write to it are:
    - new datasets,
    - new feature sets.

    I don't necessarily mean that this "read-only" is something that we need to hard-code into the data structure; it's more of just a state of mind.
  - I **believe** that h5py package is designed such that 'loading' any piece of information is put off until explicitly called for, which means that we should be able to look at each ccd.study_attrs without needing to load the entire dataset. This will let us browse the data quickly, and put off the heavy i/o for when an actual data matrix (a numpy array) is needed.
  - The current "create_dataset" function can still be used to create offshoot CCDataSets from the CCGroup, and these offshoots have the same CCDataSet functionality as we currently have, but making sure they also have extra metadata describing where they came from (again, could probably be done in the ccd.study_attrs). Since the offshoot is a CCDataSet, these can be saved to disk and modified at will (but for reproducibility maybe we don't really want people to do this too much, either!)