Alex Kotz
10/16/23

CPBS 7711 – Module 2, Day 3 HW

a. Motivating Problem from Domain
    a. There are 12 known loci that are associated with Fanconi Anemia (FA). This disease is caused by genomic instability. Creating a functional network of all Fanconi anemia genes may assist in new findings related to this disease.
b. Computational Problem
    a. Given a parent network of gene to gene relationships, a subnetwork of strictly Fanconi Anemia gene-gene relationships, and two sets of independent subnetworks, compute the statistical significance (empirical p-value) of the two subnetworks edge counts.
c. Specific Approach
    a. Given a network of nodes (gene to gene mapping) and a list of known FA associated genes , a subnetwork containing FA related genes; create a set of 5000 subnetworks that show a pseudo-randomization of FA gene-gene connections and a set of 5000 subnetworks that show a pseudo-randomization of related non-FA gene-gene connections. Then run a permutation test on the mean of edge counts (number of gene-gene connections) between the two sets of subnetworks.
d. Specific Implementation
    a. First, create a set of 5000 subnetworks (stage 1 subnetworks), leverage the fa subnetwork to identify the gene-gene connections, store stage 1 subnetworks in an object. Second, create stage 2 subnetworks that are composed of non FA genes, where each gene in each subnetwork has the same number of edge connections as the stage 1 subnetwork gene it is being derived from. Third, create a permutation test function and calculate the Empirical p-value, given the mean edge counts from both sets of subnetworks.

Hypotheses:

Alternative 1: There is a significant statistical difference between the mean edge count of the stage 1 subnetworks and the stage 2 subnetworks.

Alternative 2: The expected edge density (mean edge count) of the stage 1 subnetworks will be greater than that of the stage 2 subnetworks.

Null: There is no significant difference between the mean edge count of the stage 1 subnetworks and the stage 2 subnetworks

Discussion:

The first half of this project was relatively straightforward and did not require much refactoring (creation of the bins object, faGenes object, nonfaGenes object, and the stage 1 fa subnetworks). The second half of the project took the form of three different iterations:

1) Standard creation of stage 2 subnetworks (reference Results section) by leveraging the stage 1 subnetworks and replacing each gene in each subnetwork with a non fa gene that had the same number of edge connections (calculated from STRING 1.txt).

2) Creation of stage 2 subnetworks by using the same strategy as 1, but integration of multiprocessing to improve the speed of the stage 2 subnetwork creation.

3) Creation of stage 2 subnetworks using the same strategy as 1 and 2 but use concurrent threading to improve the speed of the stage 2 subnetworks, in place of multiprocessing.

In iteration 1, the creation of the stage 2 subnetworks took roughly 2 hours and 45 minutes to run, resulting in stage2_random_subnetworks_v1.json. Upon the completion of this iteration, a new approach was needed to cut down the runtime of this project. Multiprocessing, threading, and use of the concurrent python module were all explored. Multiprocessing resulted in a similar run time and an application memory overload, due to the project maximizing the machines resources. A pool of (5 hard coded, and an automatically generated size) processes were created, attempting to chunk up the stage 1 subnetworks into smaller, more manageable pieces. This strategy did not significantly improve run time. This strategy was also limited due to global interpreter locks, massive overhead required for each process to have its own interpreter and memory stack, and the functions used to create the stage 2 subnetwork was not serializable (pickleable). Standard threading was implemented with some improvement (roughly 2 hour run time), but the results of this attempt were unusable due to several bugs that could not be resolved. Ultimately, the version of the project submitted took the form of concurrent threading, where the stage2_random_subnetworks.json were created using a thread pool created by the concurrent module. This strategy cut the runtime from 2 hours and 45 minutes to roughly 1 hour and 22 minutes. In retrospect it would have been more logical to make greater use of the NumPy package, as NumPy allows a release of the GLI for many of its operations.

Results:

The expected outcome of this project is an empirical p-value that represents the statistical significance of the edge count mean between two sets of 5000 subnetworks (FA genes and Non-FA genes). Stage 1 set of subnetworks (FA genes) was generated by creating 5000 randomized versions of a FA subnetwork. Each randomized FA subnetwork was generated by selecting one gene, at random, from each of the 12 specified loci (from Input.gmt.txt). The final structure returned from the Stage 1 set of subnetworks was a dictionary that contained the following structure:

"subnetwork identifier (auto-incremented primary key)": {"edgeCount": number of calculated edges in the subnet, "subnet": [randomized FA subnetwork (as specified above)]}

Stage 2 set of subnetworks (nonFA genes) was generated by creating 5000 randomized nonFA subnetworks. Each randomized nonFA subnetwork was created by referencing each of the 12 genes from a given stage 1 subnetwork, finding the bin (# of edges) in which the fa gene existed, randomly select a nonFA gene from the same bin, add nonFA gene to subnetwork. The stage 2 subnetwork followed a structure similar to:

"subnetwork identifier (auto-incremented primary key)": {"edgeCount": number of calculated edges in the subnet, "subnet": [randomized FA subnetwork (as specified above)], “faBinFlag”: True/False, “binNotFoundFlag”: True/False }

The Empirical P-Value was calculated using a permutation test:

The resulting P-Value from the permutation test is 0, indicating that the observed difference in means of stage 1 subnetworks and stage 2 subnetworks is extremely unlikely to have occurred by chance. This makes some sense, as it is possible that the stage 2 subnetworks, being composed of genes from the same bin as the stage 1 subnetworks genes, could result in a similar number of total edge counts. Although, I would expect to have a result that supports the hypothesis that the stage 1 subnetworks (fa genes) would have a higher edge density value (meaning the stage 1 subnetworks have more connected genes) than the stage 2 subnetworks. It is quite possible that the p_test() function was not correct, or the composition of the stage 2 subnetworks are resulting in unexpected data, both of which could be the root of the misunderstanding.

Lessons Learned:

- Create diagram of project components (classes and objects) before programmatic implementation.
    - I spent a lot of time adding and testing features, which I later found to be irrelevant.
- Create boiler plate of classes and functions before programmatic implementation to reduce time spent refactoring the code.
- Consult with faculty and TA's about unknown statistical knowledge, before designing the project to better understand what the precise outcome is supposed to mean.
- In retrospect, it probably would have improved runtime efficiency to use categorized bins (1-100 edge connections, 101-200 edge connections … etc), instead of creating a bin for every # of edge connections found in the parent network (STRING 1.txt).
- It would also increase the efficiency of accessing the bins object, if I were to remove the bins that only contain one type of gene (fa/non-fa).