Module 4 Day 3 Report

Motivating Problem From Domain

Fanconi anemia (FA) is a rare disease that affects how DNA is translated and repaired. This leads to abnormalities in bone marrow and skeletal structure as well as an increased risk for cancer. Due to causing low red and white blood cells, patients are at an increased risk for anemia, infections, and excessive bleeding.[1] There are 22 known genes that will lead to FA if mutated. Implementation of a genetic algorithm to these genes 5,000 enriched subnetworks will be created to identify the optimized connections between genes. By evaluating the contribution of connections of each gene in 5000 enriched subnetworks, genes can be assigned a score in which a higher score correlates to a higher contribution. Additionally, a t-test can be done to determine the significance of the enriched subnetworks compared to random gene networks.

Computational Problem Formation

Given 5,000 random subnetworks of FA genes, implement a genetic algorithm to enrich connections and assign each gene a score based on contribution to connections within the subnetwork. Perform a t-test to determine significance of enrichment.

Specific Approach

Given 5,000 subnetworks of FA genes, implement a genetic algorithm by mutating and mating the most dense subnetworks to enrich. Given a STRING network of gene connections, score each FA gene in each loci based on its contribution to connections in each subnetwork, while all other loci remain the same. Implement a t-test of the densities compared to 1,000 permutations to determine significance.

Specific Implementation of Approach

For each of the 5,000 FA gene subnetworks, randomly mutate each gene with a 5% chance. Giving preference to the most dense subnetworks after the mutation step, mate two subnetworks where each loci has an equal chance of being from either parent. Continue mutating and mating until average density increases by <0.5% between generations. For each loci in the enriched 5,000 FA gene subnetworks, replace the gene from the specific loci with every other gene from that loci while keeping all other loci the same and get the densities of connections. Get the density again of the subnetwork but with the target loci removed from the subnetwork, creating an empty loci case. The difference between the density of the subnetwork with the replaced gene and the empty loci case gives the gene score. Gene scores can be visualized in a network graph where color indicates loci and size is proportional to the genes score. Using 1,000 permutations of the enriched subnetworks in a t-test, determine p-value significance of enriched subnetworks.

Pseudo Code

```
For subnetwork in FA_subnetworks:
    pre_mutation_densities = {dict of densities for each subnetwork}
    For gene in subnetwork:
        5% chance of swapping gene for different gene in same loci
    Mutated_subnetworks = {dict of mutated subnetworks}

For mutated_network in mutated_subnetworks:
    Mutated_densities = {dict of mutated densities}

For x in range(5000):
    Randomly choose two mutated subnetworks giving a higher weight to more dense
    For genes in same loci in parents:
        Randomly pick 1 gene
        Add gene to [mated_subnetwork]
    Mated_subnetworks = {dict of 5,000 mated subnetworks}

For mated_network in mated_subnetworks:
    Mated_densities = {densities of each mated_subnetwork}

Avg_den_before_mutating = sum(pre_mutation_densities.values()) / 5000
Avg_mated_densities = sum(mated_densities.values()) / 5000

Get percent increase between before_mutating_density and mated_densities

If percent_increase > 0.5:
    Repeat whole code using mated_subnetworks as starting subnetworks

If percent_increase < 0.5
    Save optimized networks
    Report top 10 dense subnetworks with p_value as cytoscape input
    Report optimized networks in GMT format
    Score genes using optimized network (see M3D3 report for pseudo code)
    Visualize gene scores
    Report optimized networks with gene scores in GMT format
    Perform t-test (see M3D2 report for pseudo code)

    Break
```

Discussion

The genetic algorithm was implemented and gene scores were re-calculated based on the optimized network. The genetic algorithm took 5 generations to optimize, with a final density increase of 0.36%. The edge weight of the connections were used to calculate density. Weight was used as opposed to the number of connections as the strengths of the connections differ. Using the weight accounts if there are many low strength connections vs. a few high strength connections. This density reflects more accurately the contribution of each gene. The gene scores were visualized and shown below in figure 1. The network showing scores before and after implementation of the genetic algorithm (GA) are the images on the left and right respectively. Genes that were not in the STRING input of all connections, the gene was scored with "NA" and was not included in the visualization.

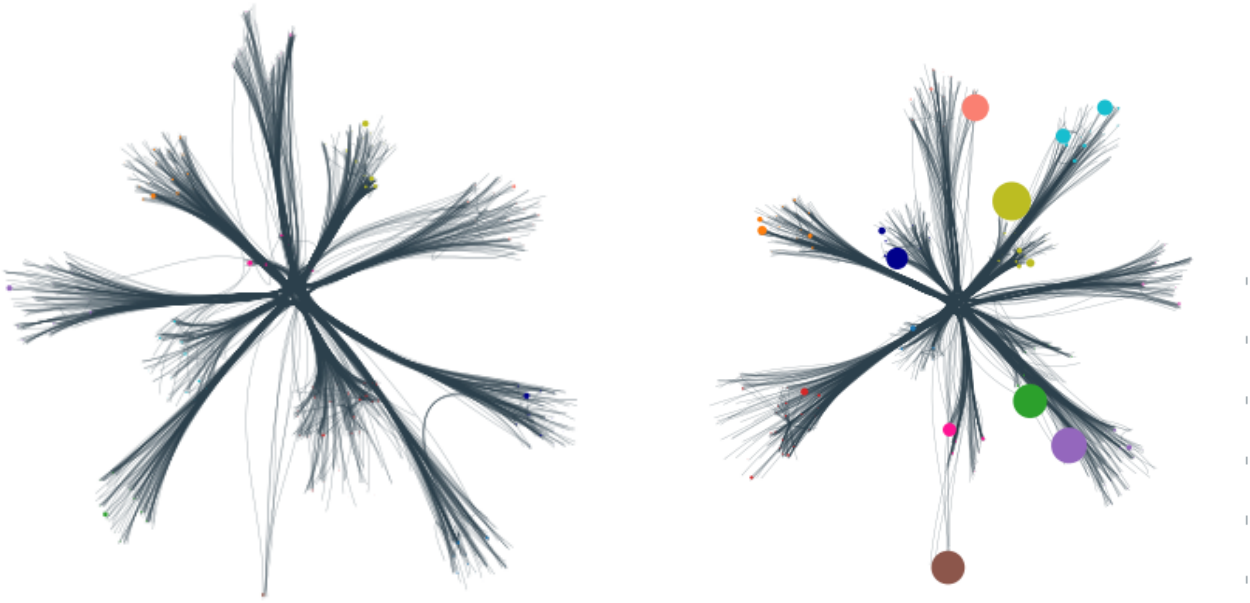

Figure 1.[2] *Network of genes where node size is proportional to gene score and color separates loci. Gene scores before GA implementation on left, after GA implementation on right.*

For visible genes, the implementation of the GA clearly increased the genes scores. This supports that implementation of the genetic algorithm successfully optimized the networks. The GA created the most dense subneworks, since the subnetworks were optimized, each gene is more likely to contribute to connections within the networks, causing their gene scores to increase.

In addition to the gene scores, a t-test was performed to determine the p-value for the optimized population and the p-value for the top 10 most dense subnetworks. For all cases the p-value was <0.01. This p-value supports that the GA did successfully optimize the subnetworks. There was no other case in which the density for the null networks was larger than the optimized networks.

The genes used in the null set were non-FA genes, which are less likely to have connections and since they were not optimized this makes them even less likely to have connections within the subnetworks, making these p-values reasonable. The most and the tenth most dense subnetwork visualized using cytoscape can be seen below in figure 2.

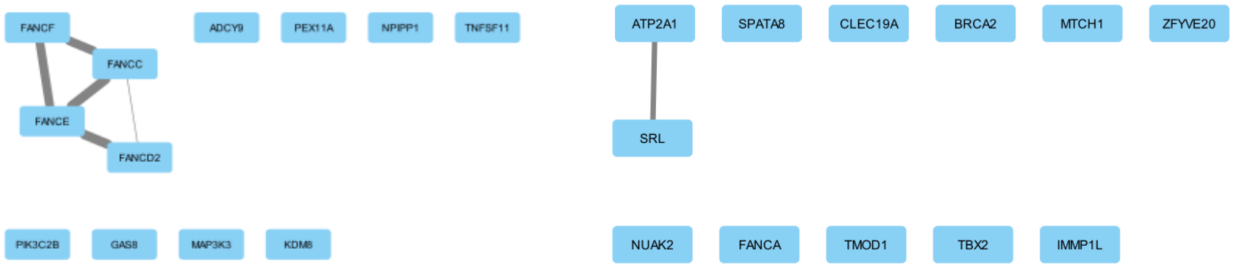

Figure 2.[3] *Left: Most dense subnetwork, $p < 0.01$. Right: Tenth most dense subnetwork, $p < 0.01$. Edge thickness is proportional to connection weight.*

These subnetworks further show the importance of using edge weight as opposed to number of connections for density. While the tenth most dense subnetwork only has one connection, it is a stronger connection and therefore is more dense than another subnetwork that may have 2 connections that are much weaker.

<u>Limitations</u>

The main limitation in this algorithm was the t-test code. In the module 4 github, there is a code for performing a t-test in which 1,000 null cases of 5,000 null subnetworks were created for the permutations. In two days, this code was able to run 100 null cases and the code was unable to finish. Instead, the same t-test code from module 3 was used, in which the densities of 1 null case of 5,000 subnetworks were shuffled 1,000 times for permutation. This could result in a less accurate p-value as making 1000 null populations would add more variety in the permutations. Let it be noted, for calculating the p-value on the 10 individual subnetworks 5000 unique null cases were used for the permutations. As each of these p-values were also < 0.01, this validates that the overall population was also < 0.01.

Sources

1. National Center for Advancing Translational Sciences |. rarediseasesinfonihgov. https://rarediseases.info.nih.gov/diseases/6425/fanconi-anemia.

2. Holtz, Yan. “Python Graph Gallery.” *The Python Graph Gallery*, python-graph-gallery.com/. Accessed 17 Nov. 2023.

3. Shannon P. 2003. Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks. Genome Research. 13(11):2498–2504. doi:https://doi.org/10.1101/gr.1239303. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC403769/.