**Motivating Problem**

Identifying functionally related genes of a disease of interest is crucial when understanding the pathways and which genes are associated with the disease phenotype. Within a network, if a gene has a higher degree of edges than others, this could possibly suggest that the gene plays a role in these pathways. By assigning each gene with a connectivity score, where the higher the number, the more connections to other genes it has, we could quantify their contribution to the network. Therefore, generating a network of the functional relatedness of genes allows for an easier way to identify candidate genes of a disease of interest by identifying the molecular pathways that underlie the pathophysiology of a disease.

**Computational Problem**

Given a full network file of genes (where the first two columns represent genes and the third column are their corresponding edge weights) and a file of disease-associated genes and their loci, assign scores to each gene based on its degree of edges within the network to characterize its functional relatedness in relation to the density of the network itself. Visualize these scores in order to characterize these complex relationships between genes.

**Specific Approach**

After reading in the full network file and the gene-locus file, randomly generate a 5000 subnetworks where each locus is represented by one gene that lies within it. To determine functional relatedness of each gene, the density of the networks with the gene included will be scored and an empty locus case where the gene is removed, and therefore, the locus is not represented will be scored based on the density without it. When subtracting the empty locus case from the density of the network with the locus included, the gene’s contribution is determined. Repeat this process for each network and then average the scores. Export out the dataframes of the Fanconi anemia source and target genes and the data frame of the gene, gene score and associated locus. Once this has been completed, visualize these scores by coloring the nodes based on their association with a locus and size the nodes based on the gene scores. Use Cytoscape as a way to visualize the network.

**Specific Implementation of Approach**

After the full network (STRING) file and the Fanconi anemia gene-locus file has been read in, a random subpopulation of FA genes is generated where one gene represents each of the 12 loci. One network would therefore have a total of 12 genes. The function random\_subnetworks creates a dictionary random\_nets and uses the FA\_genes dictionary where the keys are the loci and the values are genes associated with each loci to randomly choose a gene using dictionary comprehension and the random.choice function. The function will then return a list of 5000 dictionaries where each list is a random subnetwork and each locus is associated with a randomly selected gene. This output is now the prix\_fixe list of 5000 random subnetworks used for the next step in the algorithm. A limitation of this approach is the amount of memory usage required since the function generates the 5000 subnetworks and stores them in memory, so if the number of loci and genes are long, it could consume a significant amount of memory. This approach also attempts to randomize choices uniformly but some genes are more likely to be chosen more often than others (e.g., their frequency in the genome or the frequency in the full network file) and this function may not be an accurate representation of this. This function also does not track which genes have been selected in previous iterations, so the same gene can be selected multiple times, while others may not have been selected at all.

The next step in the algorithm is to score the genes based on how much the gene contributes to the connectivity of the network. The function gene\_scoring takes three arguments (prix\_fixe, FA\_genes and num\_networks) as inputs where it initializes an empty dictionary gene\_scores to store the scores for each gene. It does this by converting each network in prix\_fixe to a set for faster membership tests and then iterates over each subnetwork within prix\_fixe. After, it iterates over each locus in FA\_genes and selects a random gene designated as g\_star from the network and creates a new network where g\_star is removed. This will then serve the empty locus case where the contribution of a gene to the connectivity of a network is calculated without its presence in a network. Within this function, a helper function compute\_density calculates the density of a subnetwork where the density is defined as the number of edges within the subnetwork and is stored in full\_densities. The density of the empty locus case (new\_network) is calculated and then subtracted from the original network to get the contribution of g\_star to the connectivity of the network. The function will then record the contribution of g\_star and normalize by the number of loci in gene\_scores. Some limitations of this function are its performance since the function has to iterate over each network and each locus which can be very slow if the number of networks or loci is large. It also computes the density of each network twice, so the code would need to be updated so the performance can be optimized. The function also selects a random gene from each network to be removed, and some genes may be selected multiple times and other genes may not be selected at all. This could lead to some bias in the network and not be an accurate representation of the connectivity of each gene and their gene scores. One last limitation that was noticed was that content from FA\_df was used within the function but not passed as an argument, rendering the function less reusable since it depends on the specific structure of FA\_df.

The final output is a dictionary of genes as keys and their corresponding scores as the values. The dictionary was converted into a data frame and the associated loci to each gene was added to the dataframe. The data frames of gene\_scores\_df and just the source and target genes of Fanconi anemia (FA\_df) were exported to a .txt file to visualize the network in Cytoscape. FA\_df.txt now serves as the full network input where the gene\_scores\_df.txt file serves as the node table in the program.

**Pseudocode**

1. Generate a random prix\_fixe population from a defined number of networks:
   1. DEFINE a function random\_subnetworks:
      1. Input a dictionary FA\_genes:
      2. Initialize an empty dictionary random\_nets
         1. For each locus in the keys of FA\_genes:
         2. Select a random gene from the list of genes at that locus in FA\_genes
         3. Add the locus and the selected gene to random\_nets
      3. Initialize an empty list to store the random subnetworks
      4. Repeat 5000 times:
         1. Initialize an empty list for the current subnetwork
         2. For each locus in the keys of FA\_genes:
            1. Select a random gene from the list of genes at that locus in FA\_genes
            2. Add the selected gene to the current subnetwork
         3. Add the current subnetwork to the list of random subnetworks
      5. Return the list of random subnetworks
2. DEFINE a function gene\_scoring:
   1. pass three required arguments list prix\_fixe, a dictionary FA\_genes, and an integer num\_networks:
   2. Define a helper function compute\_density that takes a subnetwork and a full\_network:
      1. Filter the full\_network to only include rows where both genes are in the subnetwork
      2. Return the number of rows in the filtered network
   3. Initialize an empty dictionary gene\_scores
   4. Get a list of loci from the keys of FA\_genes
   5. Calculate the full density of each network in prix\_fixe using compute\_density and store in full\_densities
   6. Convert each network in prix\_fixe to a set for faster membership tests
   7. For each index i from 0 to num\_networks:
      1. If the size of the network at index i in prix\_fixe is less than the number of loci:
      2. Print a warning message
      3. Continue to the next iteration
   8. For each index j and locus in the list of loci:
      1. Select a random gene g\_star from the network at index i in prix\_fixe
      2. Create a new network from the network at index i in prix\_fixe and remove g\_star from it
      3. Calculate the density of the new network using compute\_density and store in empty\_density
      4. Subtract empty\_density from the full density of the network at index i to get the contribution of g\_star
      5. Record the contribution of g\_star to the connectivity of the loci in gene\_scores
   9. Return gene\_scores
3. Visualize the gene scores by inputting gene\_scores\_df.txt and FA\_df.txt into Cytoscape

References

Data formats. Data formats - GeneSetEnrichmentAnalysisWiki. (2020, December 15). <https://software.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats>

Taşan, M., Musso, G., Hao, T. et al. Selecting causal genes from genome-wide association studies via functionally coherent subnetworks. Nat Methods 12, 154–159 (2015). <https://doi.org/10.1038/nmeth.3215>