CPBS 7711 MODULE 3 DAY 3 ASSIGNMENT: Aishwarya Mandava

MOTIVATION:

Fanconi Anemia (FA) is a rare genetic disorder inherited in an autosomal recessive pattern and is characterized by physical abnormalities, bone marrow failure, and increased risk of malignancy. Around 90% of the individuals with FA have impaired bone marrow function leading to decrease in the production of Red blood cells, White blood cells, and Platelets [1]. Previous studies have found 12 genes associated with FA, including BRCA and FANC genes that play a role in DNA damage response and repair mechanisms. The construction of functional networks for FA-associated genes could offer valuable insights into novel molecular mechanisms and pathways.

COMPUTATIONAL PROBLEM:

We have a protein-protein interactions (PPI) network representing various interactions between the nodes/genes and a set of loci associated with FA disease. The computational objective is to generate a population of random subnetworks, consisting of 5000 subnetworks specifically comprising disease-associated loci, followed by computing the average gene scores for the FA disease associated genes across these 5000 subnetworks. Genes with high connectivity to the other loci in the subnetwork will be assigned high scores. Once we have calculated the average gene scores, the next step is to visualize the generated subnetworks along with their corresponding gene scores. Visualizing and analyzing these subnetworks would facilitate exploring dynamic interactions within the protein-protein network in the context of disease-associated loci.

SPECIFIC APPROACH:

Given that we have the disease (FA) associated genes and the protein-protein interactions (PPI), the approach is to first generate a population of 5000 random subnetworks, such that each subnetwork has a random representative (chosen) gene from each loci. We then score each candidate gene in each subnetwork by replacing the representative gene with every other gene in that loci and counting the number of edges within the subnetwork. We then compute the “empty case” score by completely ignoring that specific locus. The difference between the candidate gene score and the empty case indicates the contribution of the candidate gene to the rest of the subnetwork. The final average gene scores are computed by taking an average of these candidate gene scores across the 5000 subnetworks.

SPECIFIC IMPLEMENTATION:

The protein-protein interactions (PPI) network was retrieved from the STRING database [2] in the tab-delimited format. This file has 1,972,248 interactions across various genes including both FA genes and non-FA genes. The FA disease genes were retrieved from the OMIM [3] database in the Gene Map Table (GMT) format. This file has 12 loci, and 584 FA disease genes. However, not all 584 interact with other genes, only 508 disease associated genes interact with other genes from the PPI networks. Additionally 329 disease associated genes interact with other disease associated genes. Table 1 shows disease associated genes across loci before and after filtering.

Locus	0	1	2	3	4	5	6	7	8	9	10	11
All FA genes	46	50	79	50	12	50	50	50	50	48	50	50
FA genes from STRING file (FA gene connected to FA gene)	28	27	54	31	4	27	29	27	28	24	25	26

Table1: Number of disease associated genes

A) Data Cleaning:

Data cleaning and wrangling can significantly cut down the run time. Since we are only interested in the disease associated gene connections, the first step is to filter out the STRING file to keep only FA gene to FA gene networks. As mentioned earlier, there are 329 distinct genes that are in the FA gene to FA gene network. Moving forward, we are using these subsets of genes and networks for generating the 5000 random subnetworks and computing the gene scores.

B) Generating 5000 subnetworks with FA genes:

This step involves selecting a random gene from each loci to form a subnetwork. This is repeated 5000 times resulting in the generation of distinct subnetworks. There are 4.67e+16 potential ways of creating a subnetwork by randomly selecting one gene per locus (note - these 12 genes may or may not have interactions with the other genes in that subnetwork) and each subnetwork could have at most 12C2 or (12*11)/2 or 66 edges. Figure1 shows 4 loci and illustrates the 5000 random subnetworks generated from the 12 loci. The script uses min_edges parameter to select the minimum number of edges for the subnetwork to be considered for the population. By default it is set to 1, i.e., at least one edge in that subnetwork.

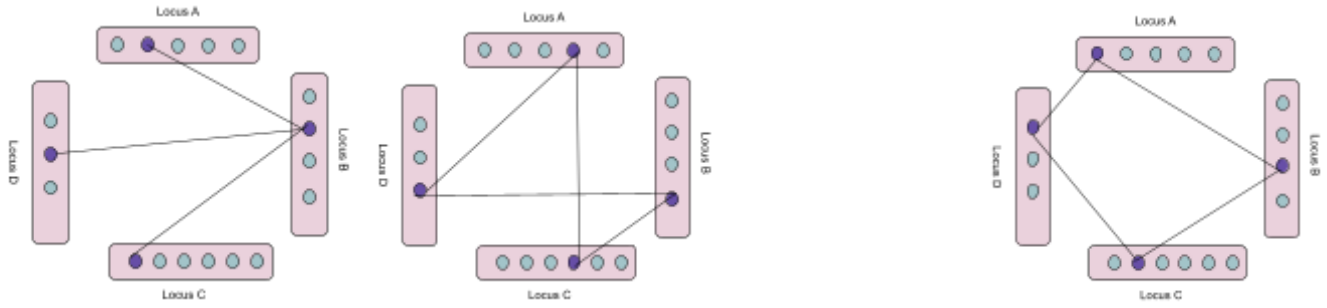
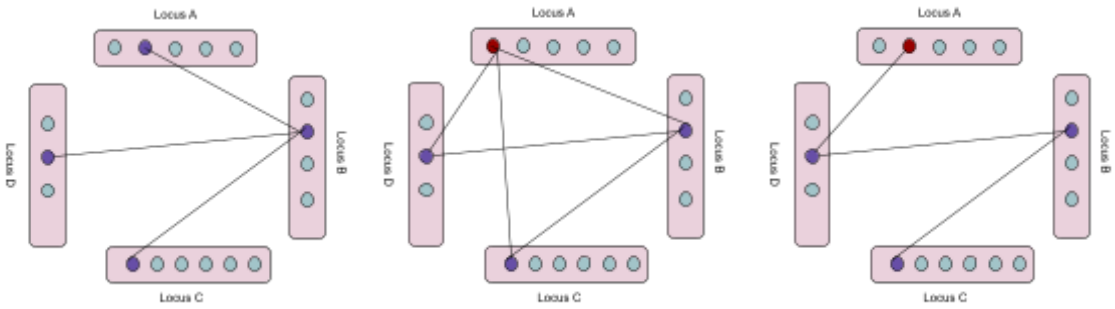


Figure1: 5000 random FA subnetworks

B) Candidate gene scores and the empty locus case:

In each subnetwork generated by randomly choosing a representative gene from each loci, we first compute the gene score which is equal to the number of edges in that subnetwork. We now replace each representative gene with every other gene in that specific loci and re-compute the gene score (edge counts). For instance, for loci L, comprising of Gi genes (g1, g2, g3, …., gn), if g* is the representative gene, we replace it with g1 through gn and re-compute their associated edge counts. For the empty case, we compute

the edge counts by completely eliminating the loci L. The final gene score for this subnetwork is the difference between the candidate gene scores and edge density of the empty locus case. Genes with higher gene scores imply high connectivity to the other loci in that subnetwork, while genes with lower scores imply lower connectivity to the other loci in that subnetwork.



.....

Figure2: Candidate gene scores

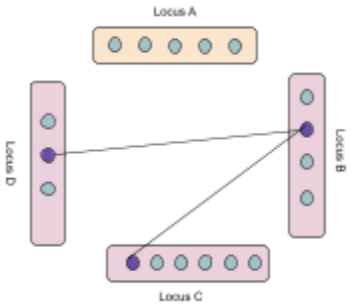


Figure3: Empty locus case

C) Average gene scores:

The final gene scores in (B) are repeated across all the 5000 subnetworks in the final population and an average gene score is computed for each gene. If a gene at a locus is not found in the network file, we give it an “NA” score, indicating its absence in the network.

RESULTS:

A) 5000 subnetworks:

There are 1024 edges that connect FA gene to FA gene. The 5000 random subnetwork generation uses a constraint of 1,i.e., at least one edge for it to be considered for the population of subnetworks. In the 5000 subnetworks, the minimum number of edges are 1 and the maximum edges are observed to be 7 for subnetwork_3018. There are 3 subnetworks with size 6 and 11 subnetworks with size 5.

B) Gene Scores:

Figure 4 shows distribution of the final average gene scores computed for 508 disease associated genes (note: some genes have NAs and are removed). The data distribution has a high frequency of genes that have gene scores between 0 and 1. There are less than ten genes that have gene scores between 2.5

and 3.5 and less than ten genes with scores between 3.5 and 4.5. There are a few genes with scores between 5.5 and 7. The highest gene score is observed as 6.6592 for *DCLK1* in Locus 8, followed by 6.112 for *PLK1* in locus 0, 5.9622 for *CDK18* in locus 5, and 5.9476 for *RAD51C* in locus 2.

Figure 5 and Table 2 further summarize the gene scores across the loci. Locus 9 has the least Max gene score of 0.103 for gene *SMG1* and the smallest mean of 0.008. Followed by Locus 9, locus 4 has a Max gene score of 2.826 for *FACA* and an average gene score of 0.318.

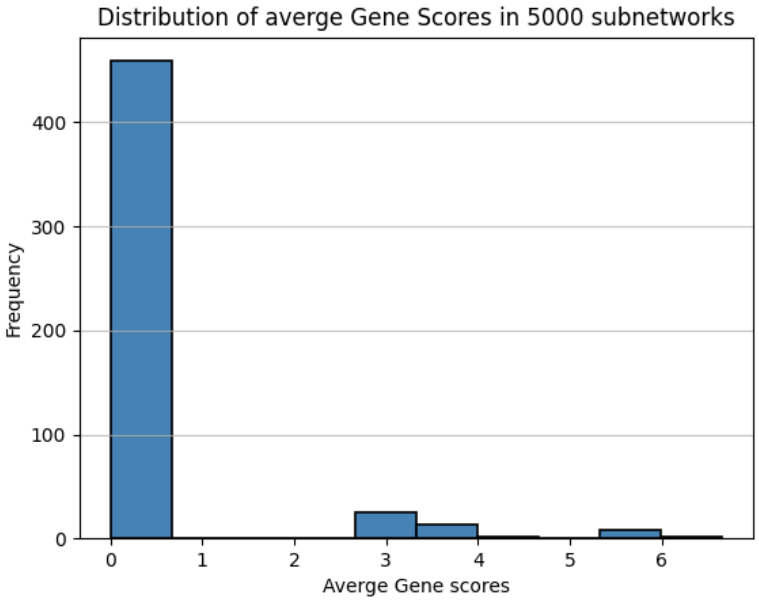


Figure 4: Distribution of gene scores

Locus ID	Mean Gene score	Min Gene score	Max Gene score	Max score gene
0	0.656	0.0	6.112	PLK1
1	0.466	0.0	5.696	KIF18A
2	0.418	0.0	5.947	RAD51C
3	0.397	0.0	5.645	ANKS6
4	0.318	0.0008	2.826	FANCA
5	0.371	0.0018	5.962	CDK18
6	0.389	0.001	3.934	IRAK2
7	0.680	0.0018	5.880	MAPK14

8	0.353	0.001	6.659	DCLK1
9	0.008	0.0002	0.103	SMG1
10	0.540	0.0	3.782	CIB1
11	0.091	0.0	3.74	SLX4

Table 2: Summary statistics for gene scores per loci

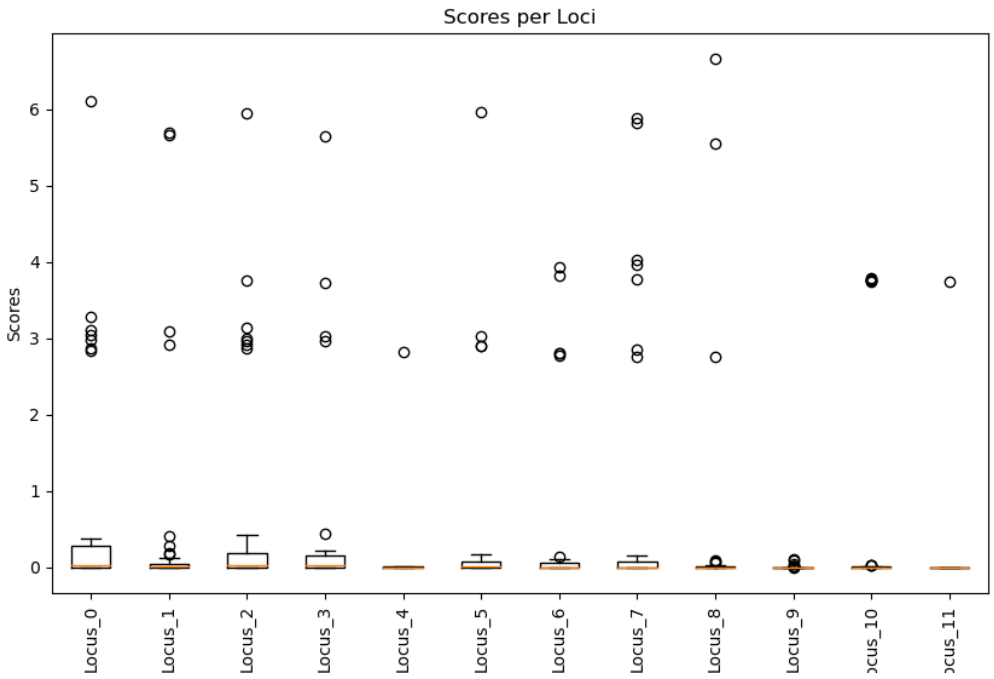


Figure 5: Distribution of gene scores across loci

C) Visualizing the subnetworks:
Figures 6 and 7 show the first 200 and 500 sub networks respectively. The subnetworks are color coded based on the locus ID and the size here indicates the log10 normalized gene scores.

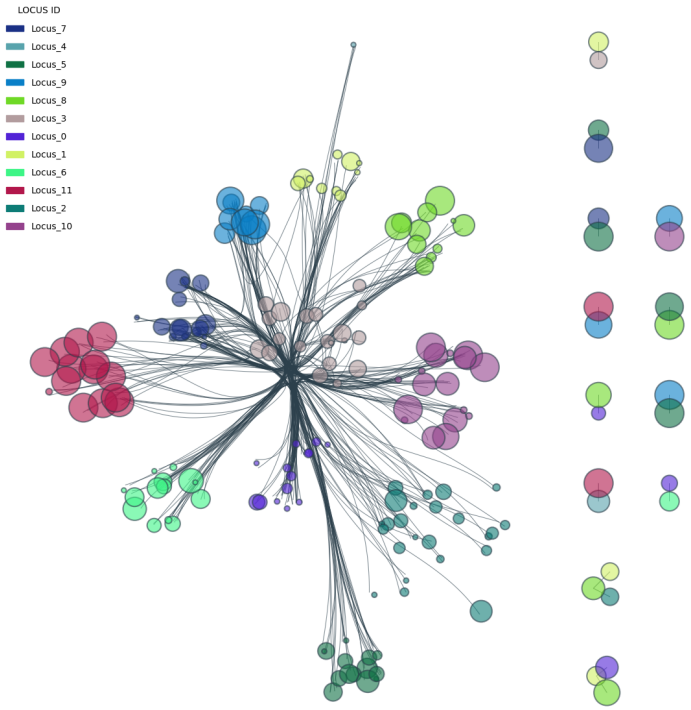


Figure 6: First 200 random subnetworks

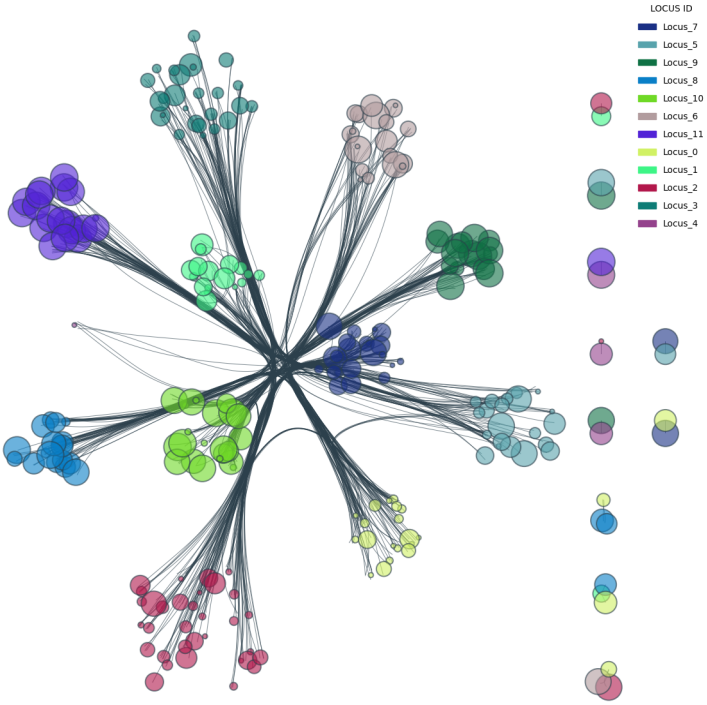


Figure 7: First 500 random subnetworks

DISCUSSION:

Genes with high connectivity to other loci in the subnetwork have high scores, and similarly, genes with low connectivity to other loci in the subnetwork have low scores. Table2 shows genes in each loci with highest gene scores. [4,5] note that PLK1 is implicated in FA-dependent centrosome replication under genotoxic stress. Biallelic inactivation of RAD51C (FANCO) and BRAC2 (FANCD1) also causes Fanconi Anemia [6]. CDK pathways are studied in Fanconi Anemia [7]. It is interesting that these genes have high average gene scores. This analysis thus provides an overview of potential functional associations among the genes associated with Fanconi Anemia.

LIMITATIONS:

Computational: The current runtime is 54 minutes.

REFERENCES:

1. https://www.ncbi.nlm.nih.gov/medgen/325420
2. Szklarczyk D, Gable AL, Lyon D, Junge A, Wyder S, Huerta-Cepas J, Simonovic M, Doncheva NT, Morris JH, Bork P, Jensen LJ, Mering CV. STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. Nucleic Acids Res. 2019 Jan 8;47(D1):D607-D613. doi: 10.1093/nar/gky1131. PMID: 30476243; PMCID: PMC6323986.
3. McKusick, V.A.: Mendelian Inheritance in Man. A Catalog of Human Genes and Genetic Disorders. Baltimore: Johns Hopkins University Press, 1998 (12th edition)
4. Aditya S Sheth, Ka-Kui Chan, Grzegorz Nalepa, D. Wade Clapp, Elizabeth Sierra Potchanant; Targeting PLK1 As a Novel Strategy for Acute Myeloid Leukemias with Fanconi Anemia Pathway Mutations. *Blood* 2022; 140 (Supplement 1): 6233–6234. doi: https://doi.org/10.1182/blood-2022-169361
5. Nalepa G, Clapp DW. Fanconi anemia and the cell cycle: new perspectives on aneuploidy. F1000Prime Rep. 2014 Apr 1;6:23. doi: 10.12703/P6-23. PMID: 24765528; PMCID: PMC3974572.
6. Rego MA, Harney JA, Mauro M, Shen M, Howlett NG. Regulation of the activation of the Fanconi anemia pathway by the p21 cyclin-dependent kinase inhibitor. Oncogene. 2012 Jan 19;31(3):366-75. doi: 10.1038/onc.2011.237. Epub 2011 Jun 20. PMID: 21685936; PMCID: PMC3974337.

PSEUDOCODE:

// Function to create dictionary of input file (loci_diction) and FA network (fa_diction), sets of all fa genes (fa_set) i.e., 584 genes; FA genes that are in the STRING file (string_set), i.e., 508 genes; and FA genes that are only connected to other FA genes (fa_string_set) i.e., 329 genes

```
FUNCTION data_to_diction(input_path,string_path):
    OPEN input_path as input:
        CREATE loci_diction as loci ID as key and genes as values
        CREATE fa_set with all the genes
        INITIALIZE fa_diction empty dictionary
        INITIALIZE fa_string_set empty set
```

```
        INITIALIZE string_set empty list
    OPEN string_path as string_file:
        FOR edge in string_file:
            SORT columns of each row

            IF gene in first column and second column are in fa_set:
                UPDATE genes to fa_string_set
                IF genes in first column is in fa_diction.keys():
                    IF genes in two columns not in fa_diction inner keys:
                        UPDATE inner dictionary to the key
                ELIF first column not in fa_diction.keys():
                    ADD inner dictionary to the key
            ELIF gene in first column or second column are in fa_set:
                IF gene in first column is in fa_set:
                    APPEND gene in first column to string_set list
                IF gene in second column is in fa_set:
                    APPEND gene in second column to string_set list

    CONVERT string_set list to a set

RETURN loci_diction, fa_set,fa_diction,fa_string_set,string_set

// Function to calculate gene scores
FUNCTION gene_score (iteration_fa_genes_diction_2):
    INITIALIZE empty gene_score dictionary
    FOR key,value in iteration_fa_genes_diction_2.items():

        // The first part of the function calculates edge case scores
        Chosen_gene = iteration _fa_genes_diction_2[key]
        INITIALIZE an empty list temp_list_2
        FOR outer_g, inner_d in fa_diction.items():
            FOR inner_g,edge_value in inner_d.items():
                IF outer_g and inner_g are NOT EQUAL TO chosen_gene:
                    IF outer_g and inner_g are IN iteration_fa_genes_diction_2.values():
                        APPEND outer_g,inner_g,edge_value to temp_list_2

        // The second part of the function calculates candidate gene scores
        FOR loci_fa_gene in loci_diction[key]:
            ASSIGN loci_fa_gene to iteration_fa_genes_diction_2[key]
            INITIALIZE an empty list temp_list
            INITIALIZE an empty dictionary gene_score_diction
```

```
                FOR outer_g,inner_d in fa_diction.items():
                        FOR inner_g,edge_value in inner_d.items():
                                IF outer_g and inner_g in iteration_fa_genes_diction_2.values():
                                        APPEND outer_g,inner_g,edge_value to temp_list
                IF key NOT IN gene_score.keys():
                        ASSIGN (len(temp_list)-len(temp_list_2) to gene_score[key]
                ELSE:
                        UPDATE (len(temp_list)-len(temp_list_2) in gene_score[key]

        RETURN gene_score


// Function to generate 5000 random subnetworks and calculate candidate gene scores
FUNCTION subnetwork_genescore(loci_diction, num_subnetwork=5000,min_edges=0):
        INITIALIZE an empty dictionary subnetwork_diction
        INITIALIZE an empty dictionary fa_subnetwork
        INITIALIZE an empty dictionary iteration_fa_genes_diction
        INITIALIZE an empty dictionary iteration_fa_genes_diction_2
        INITIALIZE an empty dictionary gene_score_func
        INITIALIZE an empty list fa_subnetwork_list
        INITIALIZE a dictionary with 0s for each loci in gene_scores_loci
        ASSIGN a random seed of 123
        INITIALIZE i to 0

        WHILE len(fa_subnetwork.keys()) < num_subnet:  //5000 subnetworks
                INITIALIZE an empty set iteration_fa_genes

                FOR key,value in loci_diction.items():
                        SELECT random_index with random.randint between 0 and len(value)-1
                        ADD value to iteration_fa_genes list

                IF NOT ANY (iteration_fa_genes EQUAL TO values FOR values IN
iteration_fa_genes_diction.values()):
                        ASSIGN iteration_fa_genes to iteration_fa_genes_diction

                INITIALIZE an empty list temp_list

                FOR outer_g, inner_d in fa_diction.items():
                        FOR inner_g, edge_value in inner_d.items():
                                IF outer_g and inner_g in iteration_fa_genes_diction for that iteration:
                                        APPEND outer_g, inner_g, edge_value to temp_list
                                        APPEND outer_g, inner_g, edge_value to fa_subnetwork_list
```

```
                        //default min_edges is 0 and can be increased for specific length for subnetworks
                        IF len(temp_list) > min_edges:
                            ASSIGN len(temp_list) to subnetwork_diction
                            ASSIGN temp_list to fa_subnetwork
                            ASSIGN the gene and locus pair to iteration_fa_genes_diction_2

                    CALL gene_score function and ASSIGN to gene_score_func
                    FOR loci_num, inner_diction in gene_score_func.items():
                        FOR loci_gene, score in inner_diction.items():
                            ASSIGN scores to gene_scores_loci[loci_num][loci_gene]

    RETURN fa_subnetwork, fa_subnetwork_list, gene_scores_loci, iteration_fa_genes_diction,
    iteration_fa_genes_diction_2


//Function to calculate the gene scores average and assign NA to genes that do not exist in
FUNCTION gene_scores_avg(loci_diction,gene_scores_loci,compare_set,num_subnet=5000):
    INITIALIZE a dictionary with 0 values for loci ID and genes

    FOR locus, inner_d in gene_scores_loci.items():
        FOR inner_g,score_sum in inner_d.items():
            IF inner_g NOT IN compare_set:
                ASSIGN 'NA' to gene_scores_final[locus][inner_g]
            ELSE:
                ASSIGN average gene score to gene_scores_final[locus][inner_g]

    RETURN gene_scores_final


// Function to create Visualization using NetworkX module

FUNCTION fa_subnet_viz(iteration_fa_genes_diction, gene_scores_final, fa_subnetwork, loci_diction,
fa_subnetwork_list):


    INITIALIZE an empty set nodes_subs
    FOR fa in fa_subnetwork.values():
        FOR fa_subs in fa:
            ADD fa_subs[0] to nodes_subs
            ADD fa_subs[1] to nodes_subs
```

```
INITIALIZE an empty dictionary gene_scores_nodes
FOR l,g in loci_diction.items():
        FOR gns in g:
                IF gns in nodes_subs:
                        ASSIGN l to gene_scores_nodes

INITIALIZE an empty dictionary node_size
FOR inner_scores in gene_scores_final.values()
        FOR g,s in inner_scores.items()
                IF g in nodes_subs:
                        Node_size[g] = math.log10(float(s)+0.01) //transform node sizes to log10 scale

INITIALIZE an empty nx Graph G

FOR nd in nodes_subs:
        ADD NODE list nd to G

FOR edge_ in fa_subnetwork_list:
        STRIP edges from edge_ and ASSIGN to j
        CREATE edge format in k
        ADD EDGE list k to G

// Give random colors to the nodes based on loci ID
ASSIGN locus ID to unique_colors
ASSIGN random colors to locus ID in random_colors dictionary
ASSIGN colors to genes by locusID in node_to_colors

CREATE a figure and subplots
CREATE a graph instance
ADD legend
RETURN figure_path
```