CPBS 7711 MODULE 1 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 network representing various interactions between the nodes/genes. The computational task is to create and visualize a functional subnetwork of query nodes (genes associated with FA) including all nodes and edges connecting these query nodes.

SPECIFIC APPROACH:

Given that we have the disease (FA) associated genes and the protein-protein interactions, the approach is to identify and visualize subnetworks with all the nodes and edges connecting these query nodes.

SPECIFIC IMPLEMENTATION:

The protein-protein interactions network was retrieved from the STRING database [2] in the tab-delimited format. The FA disease genes were retrieved from the OMIM [3] database in the Gene Map Table (GMT) format. Three functional subnetworks were constructed using the approaches below.

a) Subnetwork 1: Generate a network connecting genes from different loci. That is, this network only has FA genes connected to FA genes from different loci and excludes connections from within the same loci. Here, the rows are included when both the genes from columns 1 and 2 in the STRING file are FA genes from different loci. Figure 1 summarizes this subnetwork.

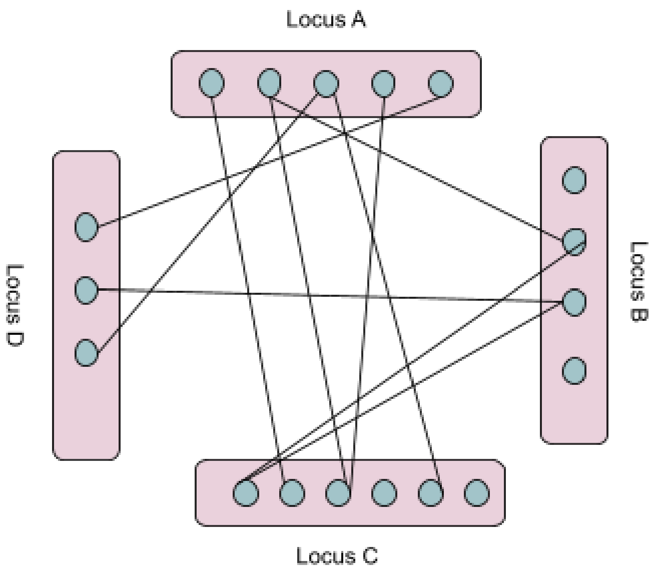


Figure 1

b) Subnetwork 2: Generate a network connecting genes regardless of the locus. This implementation uses all the disease genes and selects rows from the STRING file when both the genes in columns 1 and 2 are FA genes. Figure 2 summarizes this subnetwork.

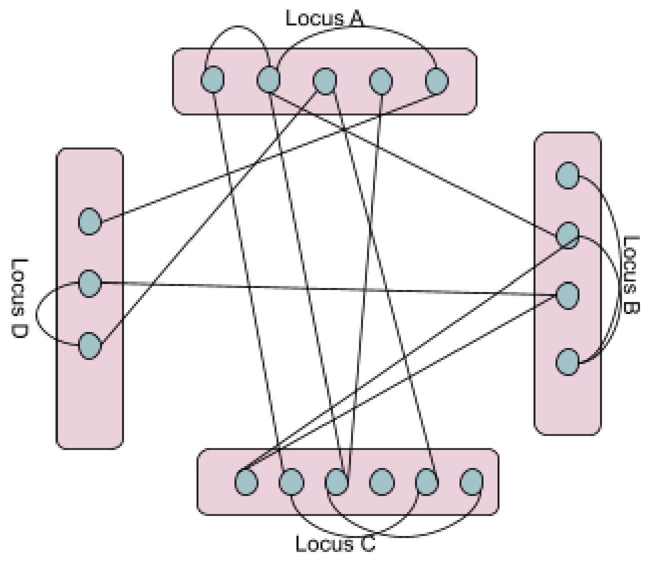


Figure 2

c) Subnetwork 3: Generate a network by including non-FA genes that form a network path with FA genes. This subnetwork is generated when either of the columns 1 or 2 in the STRING file are FA genes and a non-FA gene (if present) has to be connected to two or more FA genes in any loci.

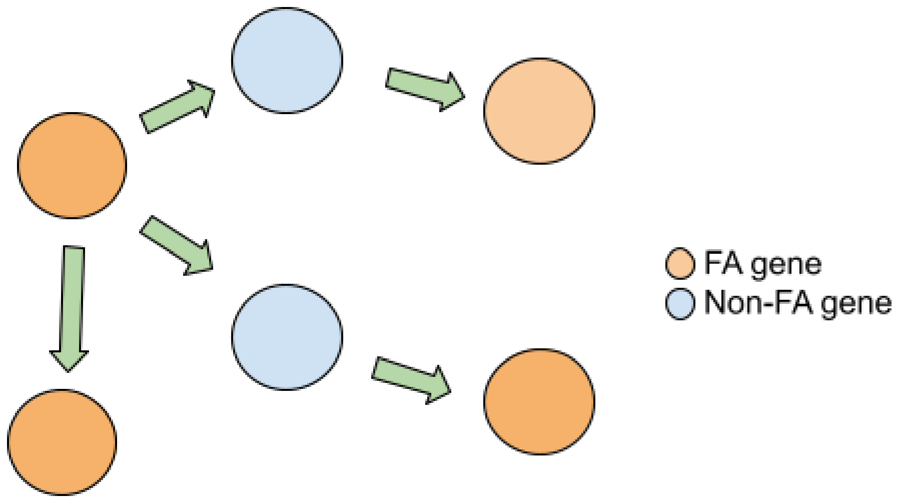


Figure 3

RESULTS:

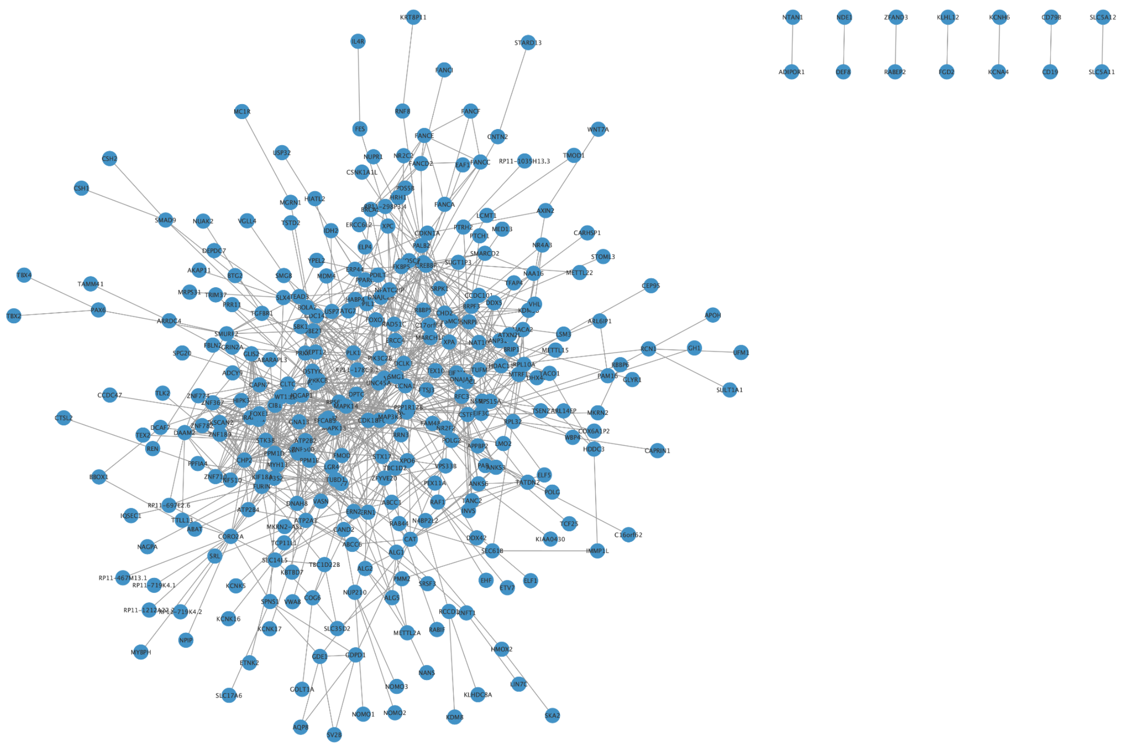


Figure 4

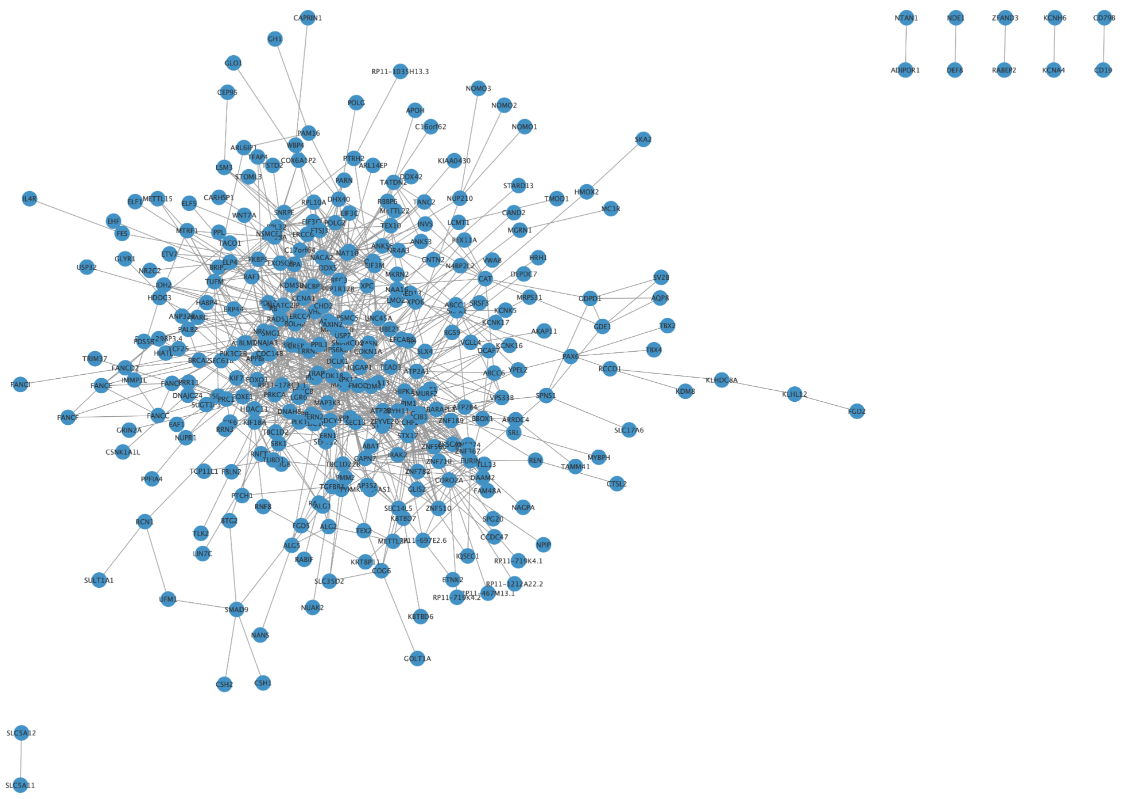


Figure 5

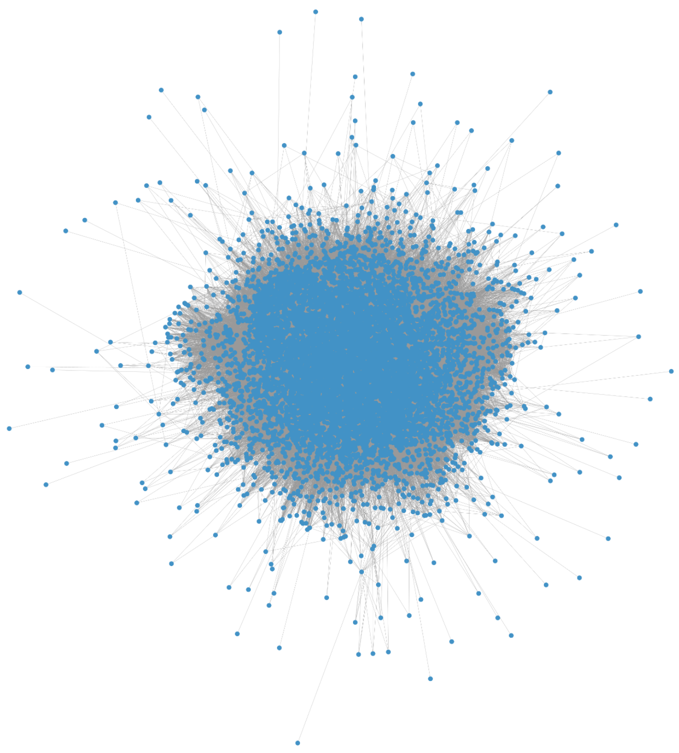
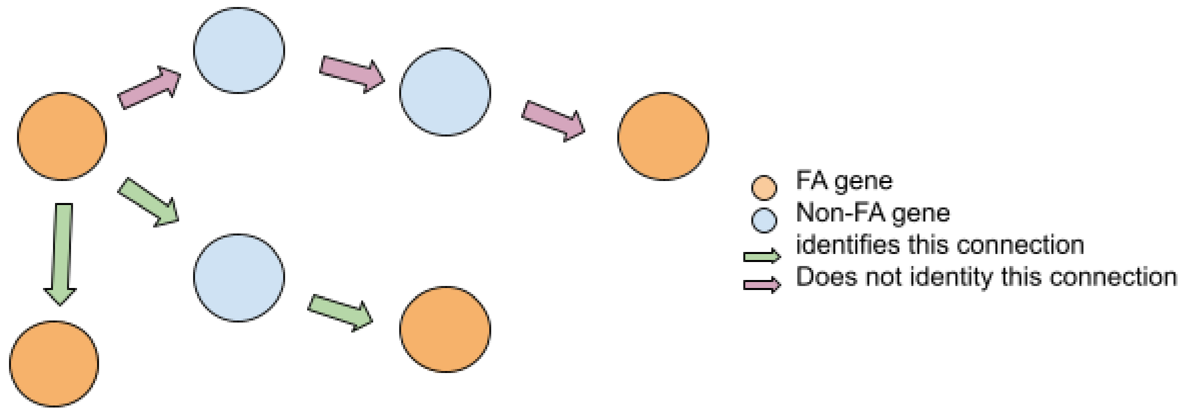


Figure 6

DISCUSSION:

Limitations: 1. This approach includes a non-FA gene that connects two or more FA genes. This does not take into account more than one non-FA gene connected to FA genes.



Challenges:

1. There are duplicate entries for the same connection and edge. For example:

RAB5C	ZFAND4	0.406000
ZFAND4	RAB5C	0.406000

2. There are entries with different values for the same nodes

RAB5C	ZFAND4	0.406000
ZFAND4	RAB5C	0.406000
RAB5C	ZFAND4	0.824608
ZFAND4	RAB5C	0.824608

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. Gustavsen JA, Pai S, Isserlin R et al. RCy3: Network biology using Cytoscape from within R [version 3; peer review: 3 approved]. F1000Research 2019, 8:1774 (https://doi.org/10.12688/f1000research.20887.3)

PSEUDOCODE:

```
// Function to create a dictionary for the Input.gmt.txt file
Function loci_dictionary(loci_path):
    INITIALIZE an empty dictionary loci_diction
    Open loci file at loci_path
    FOR each line in loci file:
        Initialize geneList as an empty list
        Extract gene names from the line
        Add gene names to loci_diction
    RETURN loci_diction

// Function to create a list for the Input.gmt.txt file
Function loci_list(loci_path):
    INITIALIZE an empty list loci_list
    Open loci file at loci_path
    FOR each line in loci file:
        Extract gene names from the line
        Add gene names to the loci_list
    RETURN loci_list

// Function to create a dictionary for the STRING database file
Function string_dictionary(string_path):
    Create an empty dictionary string_diction
    Open string file at string_path
    FOR each edge in string file:
        SORT columns of each row
IF gene in first column is in string_diction keys:
        IF gene in second column and edge value are not inner dictionary of string_diction:
            UPDATE the inner dictionary of string_diction by appending it
ELIF gene in first column is not in string_diction keys:
        Create a new key for this row
    RETURN string_diction

// Function to generate subnetwork 1
Function subnetwork_1(loci_geneList, string_geneList):
    INITIALIZE an empty dictionary subnetwork_diction
    FOR each locus_key, locus_value in loci_geneList.items():
        INITIALIZE a list for loci include_loci

        FOR each k in loci_geneList.keys():
            IF k is not equal to locus_key:
                Add k to include_loci
        INITIALIZE a list for comparison
        FOR each i in include_loci:
            Add all genes from loci_geneList[i] to compare_list

        FOR each node in locus_value:
            IF node is in string_geneList.keys():
                INITIALIZE a sub-dictionary for each node subnetwork_diction[node]

                FOR each comp in compare_list:
                    IF comp is in keys of string_geneList[node]:
                        Add string_geneList[node][comp] to subnetwork_diction[node]
    RETURN subnetwork_diction

// Function to generate subnetwork 2
Function subnetwork_2(loci_geneList,string_geneList):
    INITIALIZE an empty dictionary subnetwork_diction
    FOR each locus_key, locus_value in loci_geneList.items():
        INITIALIZE a list for loci include_loci
        FOR each k in loci_geneList.keys():
            Add k to include_loci

        INITIALIZE a list for comparison

        FOR each i in include_loci:
            Add all genes from loci_geneList[i] to compare_list

        FOR each node in locus_value:
            IF node is in string_geneList.keys():
                INITIALIZE a sub-dictionary for each node subnetwork_diction[node]

                FOR each comp in compare_list:
                    IF comp is in keys of string_geneList[node]:
                        Add string_geneList[node][comp] to subnetwork_diction[node]
    RETURN subnetwork_diction

// Function to create subnetwork 3
Function non_fa_list(fa_list,string_diction):
    INITIALIZE an empty dictionary string_subset
    INITIALIZE an empty list not_fa

    FOR each gene1, sub_string in string_diction.items():
        FOR each gene2 in sub_string.keys():
            IF gene1 is in fa_list and gene2 is in fa_list:
                IF gene1 is not in string_subset.keys():
```

```
                Create new key in string_subset with the inner dictionary as gene2, edge
            ELSE:
                Update the inner dictionary with gene2, edge
        ELIF gene1 is not in fa_list and gene2 is in fa_list:
            Add gene1 to the list of not_fa genes
        ELIF gene2 is in fa_list and gene2 is not in fa_list:
            Add gene2 to the list of not_fa genes

INITIALIZE an empty dictionary not_fa_nodes_count to count the number of connections each non-FA
gene has with an FA gene

FOR each node in not_fa:
    IF node is in not_fa_nodes_count.keys():
        INCREMENT the count for that node
    ELSE:
        INITIALIZE the count for the node to 1

INITIALIZE an empty list not_fa_nodelist

FOR each k,v in not_fa_nodes_count.items():
    IF v is greater than or equal to 2:
        Add k to not_fa_nodelist

APPEND not_fa_nodelist to fa_list to get the final list of nodes for subnetwork3

RETURN string_subset, final_nodelist

Function subnetwork_3(string_diction,final_nodelist):
    INITIALIZE an empty dictionary string_nonfa

    FOR each gene1, sub_string in string_diction.items():
        FOR each gene2 in sub_string.keys():
            IF gene1 is in final_nodelist and gene is in final_nodelist:
                IF gene1 is not in string_nonfa.keys()
                    Create new key in string_subset with gene2, edge
                ELSE:
                    Update the inner dictionary with gene2, edge

    RETURN string_nonfa

// Function to convert a nested dictionary to tab delimited file
Function diction_to_text(nested_dict):
    INITIALIZE an empty list result
```

```
FOR each key1, inner_diction in nested_dict.items():
    FOR each key2, value in inner_diction.items():
        APPEND (key1, key2, value) to the result

RETURN result
```