**Communities within Disease Network**

Xiaojing

**Overview**

This project investigates the communities within a public health network. The first part of the project will at the relationships between single features related outcomes (specifically, are those outcomes also significantly connected too). In the second part, the aim will be to identify strongly connected sub-communities within the health network. Finally, the goal is to explore the shortest path to a specific disease.

**Data**

The in-house dataset contains about 3000 individuals and 270 features, including their self-reported diseases, self-reported lifestyles, clinical tests, and genetic information.

**Pre-processing**

A linear regression will be performed on the dataset with adjusted age, sex, ethnicity, cohort and testing season. The output result files will include false discovery rate (FDR), p-value and weight (standardized from absolutely regression coefficients) between any of the two features.

**Questions**

**Easier.** For a given feature, which of its related features are also significantly connected? For examples, if the give feature, *smoking*, is significantly related with *stroke*, and *depression*, if *stroke* and *diabetes* are not connected, we would like *smoking* is likely to be the common etiology of both *stroke* and *depression*.

**Middle.** Figure out the smallest set of features who are connected with other features in the network, to form a biology cluster. For examples, if one feature in the smallest set was changed, every feature else is very likely to change as well.

**Harder.** Figure out the shorted pathway to a disease outcome. For example, to figure out the mechanism how dose *smoking* causes *depression,* the result may show as *smoking – high body weight – depression*.

**Algorithm, Limitations, Risk**

**Pre-processing.** It needs a lot of data pre-processing, including data cleaning, model testing, statistical analysis, and so on. For instance, I need to think about the data structure of how to read-in the result file, and storage the information properly.

**Easier Question.** There should be some criteria, i.e., only relationship with p-value < 0.05 should be considered as connected, relationship within subcategory (e.g., *smoking* and *drinking* are both under lifestyle subcategory) are less interesting than across-subcategories (e.g., *smoking* and depression, meaning lifestyle and disease subcategories).

**Middle Question.** I need to think of how to setup the directions. For example, genetic can also lifestyle changes, but nothing can change genetic (born with it). But lifestyles and clinical test can be bi-direction.

**Harder Question.** This is tough. The direction and how to present the weight will the two major concerns. I need to do more research to determine which shortest path algorithm that I want to use.

Others. I would like to read more about how to apply community analysis, graph based probability analysis in biology data science, and also try to implement and test their performance.

**Code overview**

**Class name:** Record

**Purpose:** How to store single linear regression results, i.e., the statistical analysis results from *smoking* to *depression*

**Class name:** RecordParser

**Purpose:** Read-in one line from the result file, separated the string, and store it as one record

**Class name:** ResultLoader

**Purpose:** Load the whole result file, and store all the result into a list of records

**Interface name:** Graph

**Purpose:** Claim the function that a graph includes

**Class name:** Edge

**Purpose:** Store the edge information, including the weight and the destination

**Class name:** CapGraph

**Purpose:** The graph class setup the constructers, including the function of finding egonet, finding the communities, and calculate the shortest path

**Class name:** Graphload

**Purpose:** Load the list of records into a graph, including different function, e.g., load record with p-value < 0.05 into the graph, load record from certain subcategory into the graph

**Class name:** main

**Purpose:** test and run the whole program

**Class name:** MapNode

**Purpose:** Convert an input string to a mapnode, with pre, distance from start, and all edges

**Class name:** Path

**Purpose:** Perform path analysis, i.e., using Dijkstra to calculate the shortest path

**Testing plan**

I have tested using a small testing file, and the code currently works well. Next plan is to load the whole result file and further testing it.

**Reflection so far**

I have changed some class compared to my initial plan, and add some new class. Currently, the pro-processing has been finished, the loading part works well, and the easier question has also been solved. I will focus on the middle and harder question later.