**CS3907: Big Data and Analytics - Project #1 Write Up**

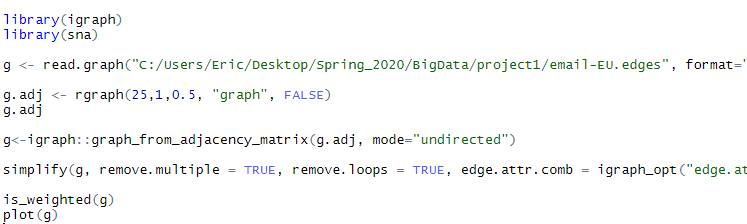
As a team, we began the project by first creating a repository on Github and uploading the data from blackboard. We placed all these files (the data, our own R file, that interacts with the data, and write-up file) into our shared repository. Our next step was to read the graph through the R function “read.graph” and begin our analysis. We then looked at 10 different R functions in order to better grasp what the data was showing us. From looking at the graphs connectivity, its diameter, and neighboring nodes, we were able to figure out different characteristics of the data. Finally, through our research, we were then able to determine who the central person(s) is, the longest path, the largest clique, ego, and power centrality.

Below is the set of 10 functions we utilized in order to figure out how each of the 25 inputs is connected and how they interacted with one another.

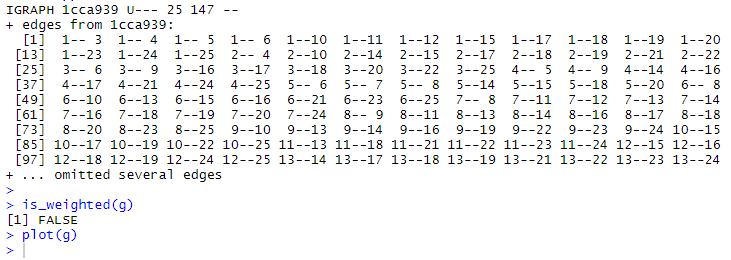
**Function 1:** is\_weighted

This function allows users to determine if an inputted graph, either directed or undirected, has an edge attribute, which identifies if it is weighted or not. We first simplified the graph through the simplify function. This function removes multiples and loops. The outputted results are pictured below, showing that our graph was not weighted.

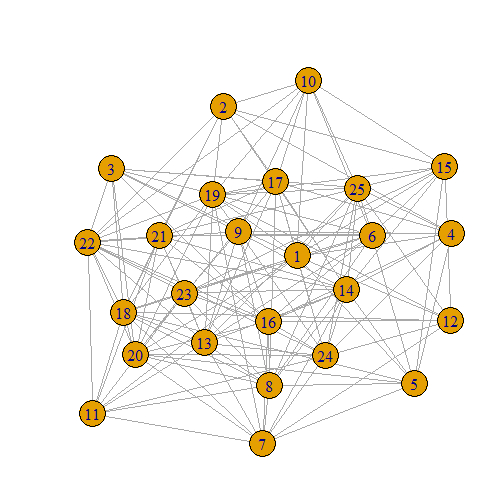
Code:



Result:



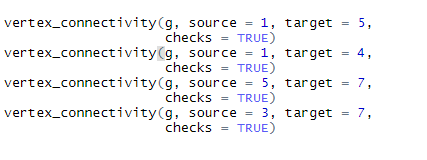
Graph:



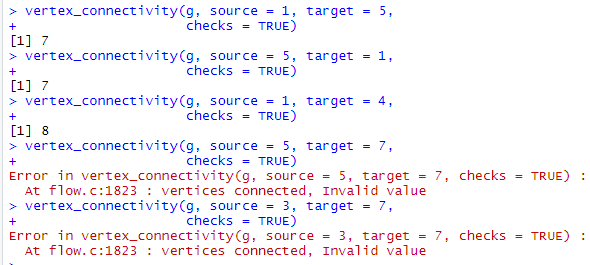
**Function 2:** vertex\_connectivity

This function analyses the connection between a source and target node in a directed graph, to identify how many nodes need to be removed in order for there to be no direct paths between the source and target node. We tested this function with several nodes. The output showed that the connectivity of nodes one and five is seven. This means that seven nodes would need to be removed in order for there to be no direct paths from one to five.

Code:



Result:



**Function 3:** diameter

This function is almost self-explanatory. This determines the length of the largest geodesic, the shortest path between two nodes. The output was 2.

Code:



Result:



**Function 4:** Isomorphic

This function is also self-explanatory. It determines whether two groups correspond to each other in a manner that is one-to-one (whether they are isomorphic). The output was True.

**Code:**



**Result:**



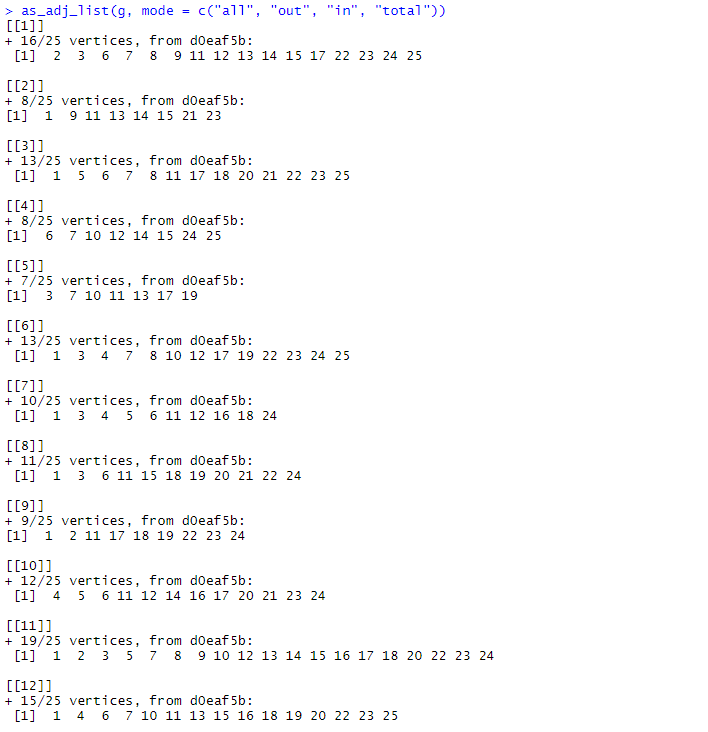
**Function 5:** as\_adj\_list

This function creates an adjacency list from the inputted graph. The result returns all of the neighbors of each node in the graph. This accounts for outgoing edges and incoming edges since this is a directed graph.

Code:



Result:



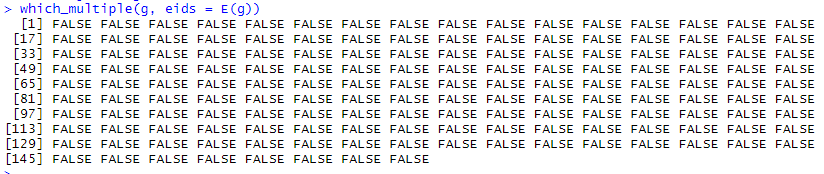
**Function 6:** which\_multiple

This function returns if there are any duplicate connections within the graph. The result was false for every connection, meaning that there were no connections, connected more than once.

Code:



Result:



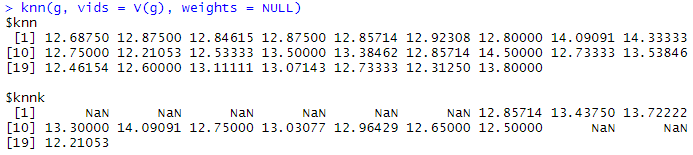
**Function 7:** knn

The function, knn, returns the average nearest neighbor degree.

Code:



Result:



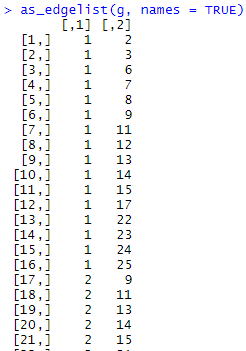
**Function 8:** as\_edgelist

This function outputs the neighbors of each node in a list format. This is similar to function five.

Code:



Result:



**Function 9:**

Code:

Result:

**Function 10:**

Code:

Result:

For getting the answers to who the central person(s) is, the longest path, the largest clique, ego, and power centrality, we looked to five different R functions that will be described below, with an explanation, and answer. These functions were included in the packages that were installed.

**Longest Path:**

The function used to extract the longest path from the graph by using the path that has the largest diameter. The result outputted is 9, highlighting that the longest path goes through 9 nodes.

Code:



Result:



**Largest Clique:**

A clique is a completely connected subgraph. In this, every node is connected to every other node. These cliques have a strong connection, as every node is connected.

Code:

Result:

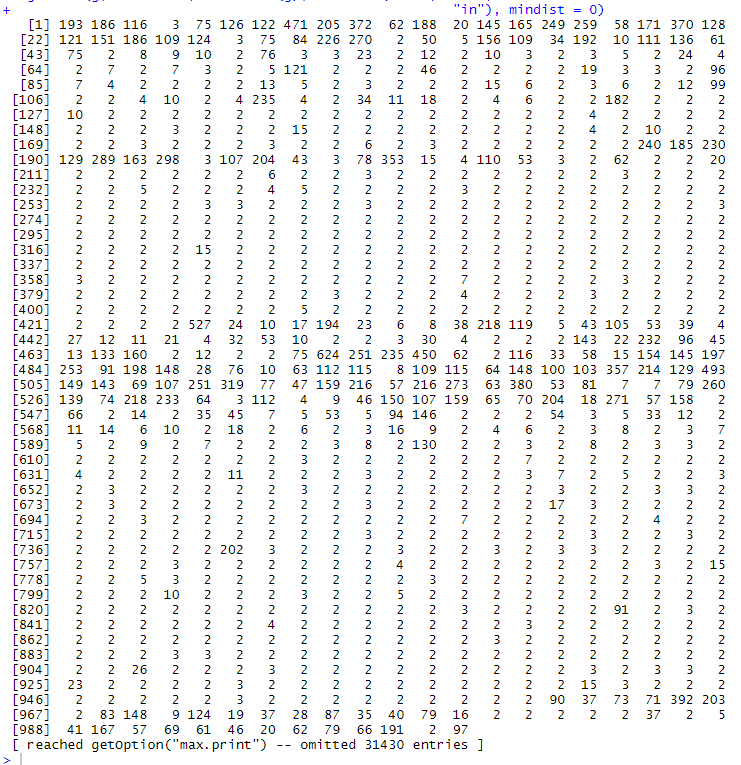
**Ego:**

Ego focuses upon seeing which nodes are directly and indirectly connected to one another. For example, if a node has a direct link to one node or if a node goes through another node to get to the target node. Below is the result of all the connections of nodes to one another.

Code:



Result:



**Power Centrality:**

Code:

Result: