THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Class Project I

Intro to Big Data & Analytics _CSCI_6444_80

Team - 9

Bhavya Sree Gudiseva - G41949795
Shejal Shankar - G32395894

Prof. Stephen Kaisler

26th February 2024

Installation of the igraph package:

To install the igraph package, you can use the following command in R:

> install.packages("igraph")

> library(igraph)

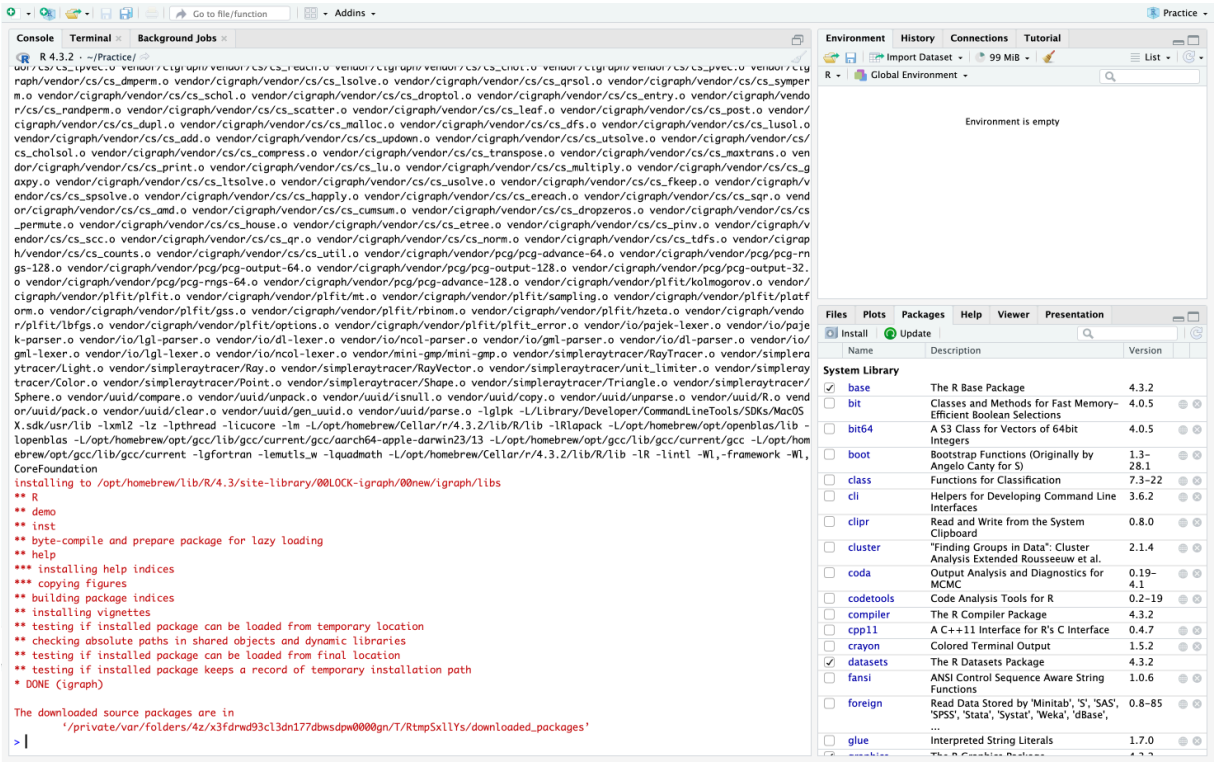

Figure 1

Set Working Directory:

First let's set the working directory in the RStudio -> Session -> Set Working Directory.

> setwd("/Users/bhavya/Academics/ThirdSem/BDA/Project1")

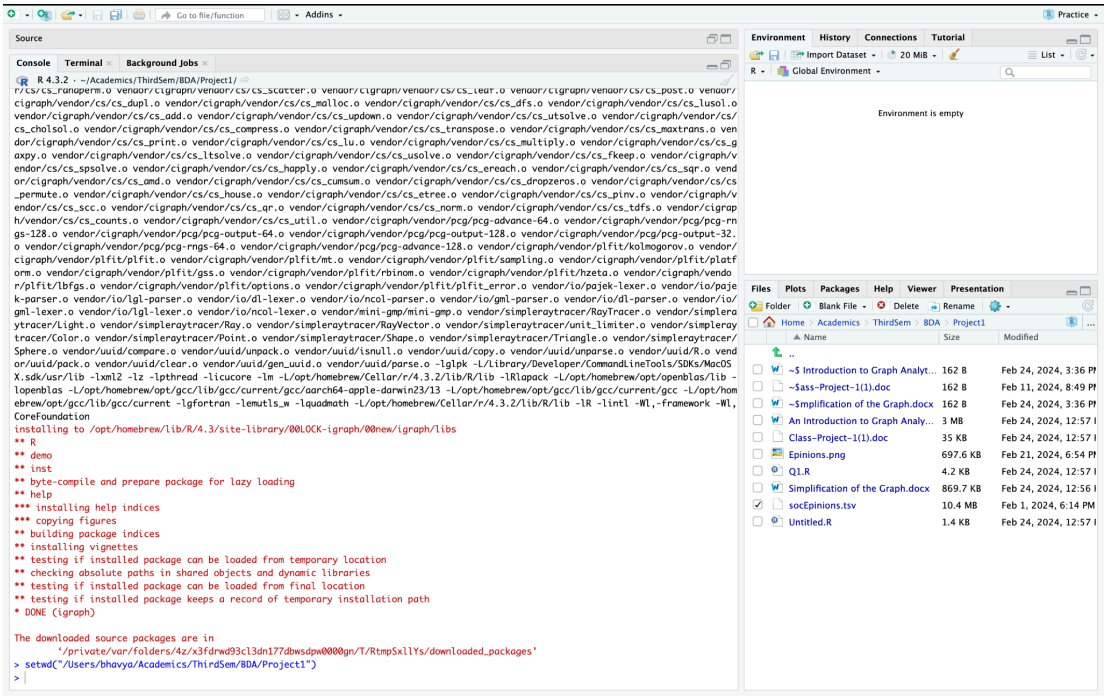

Figure 2

Import Dataset:

Now, let's import the specified dataset.

```
> socEpinions <- read_tsv("socEpinions.tsv")
```

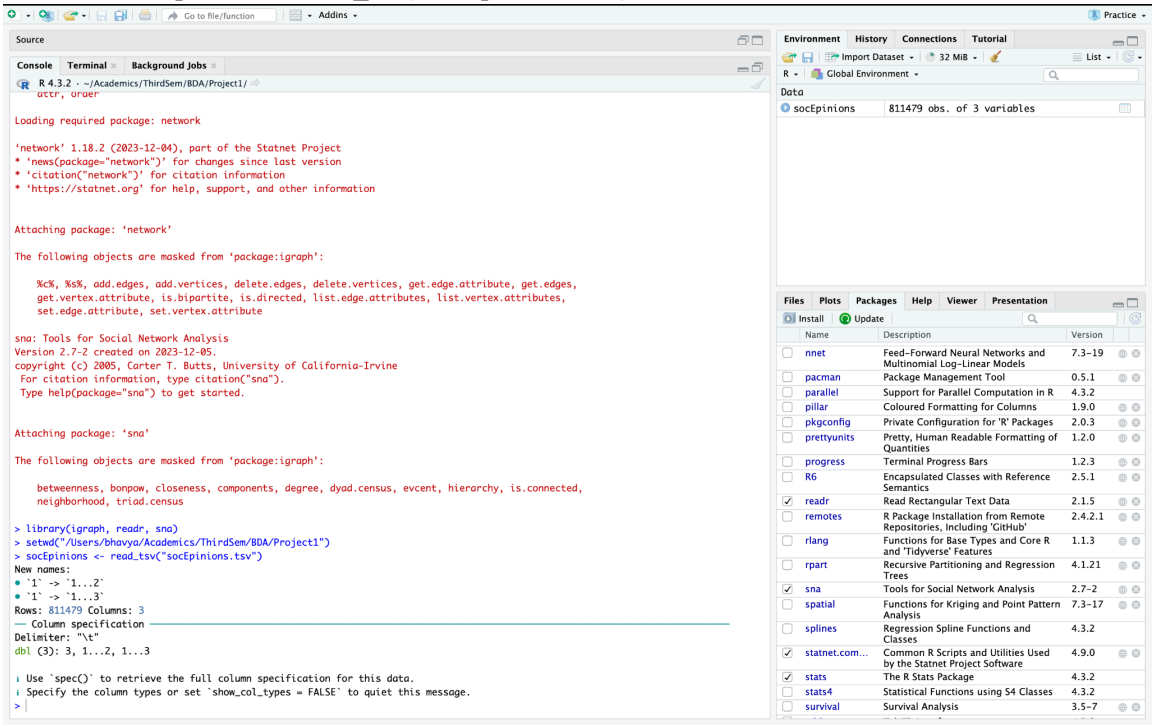

Figure 3

Create the Graph:

Now converting the dataset to matrix

> opTab <- as.matrix(socEpinions[,-3])
> head(opTab)

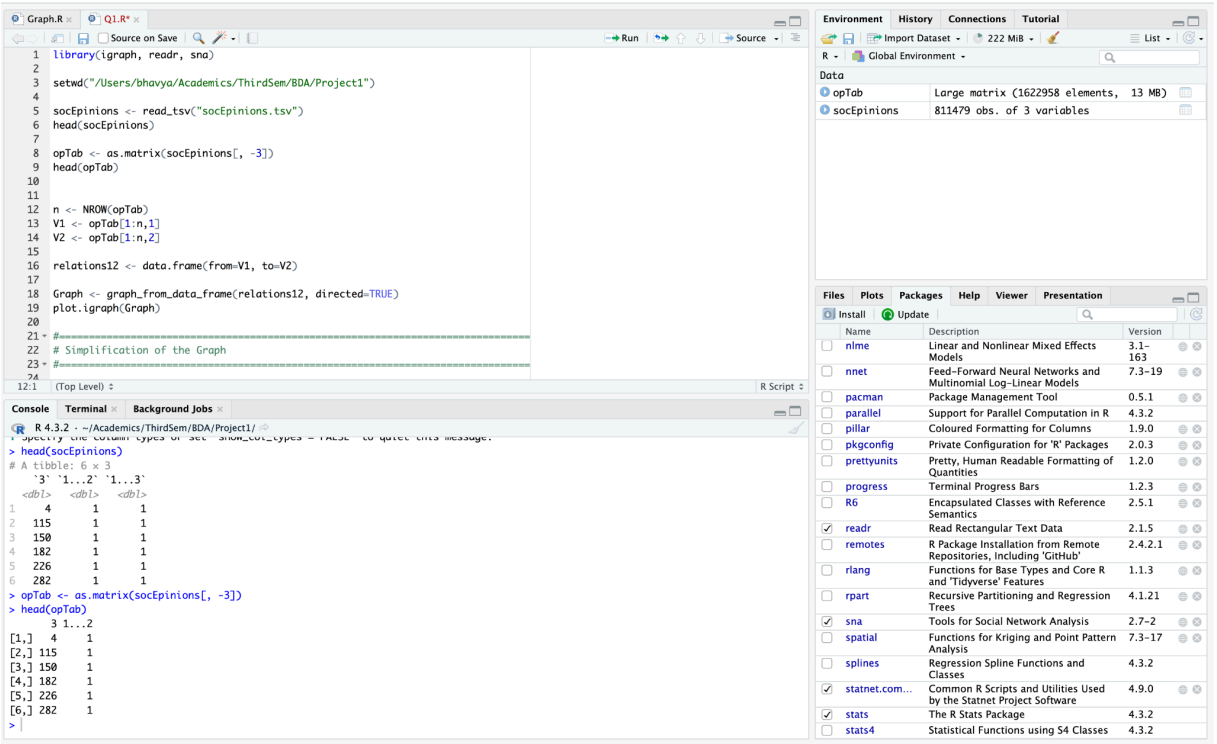

Figure 4

Now extracting the vectors

> n <- NROW(opTab)
> V1 <- opTab[1:n,1]
> V2 <- opTab[1:n,2]

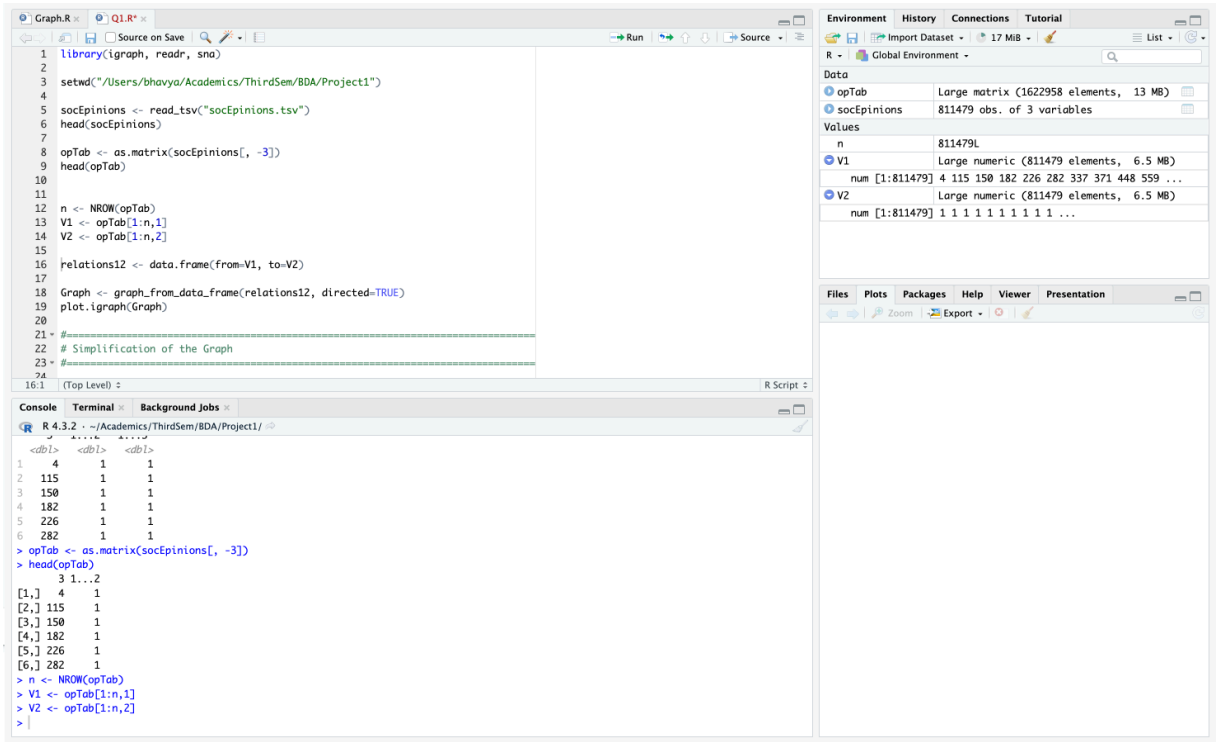

Figure 5

Create the data frame using the vectors

> relations12 <- data.frame(from=V1, to=V2)

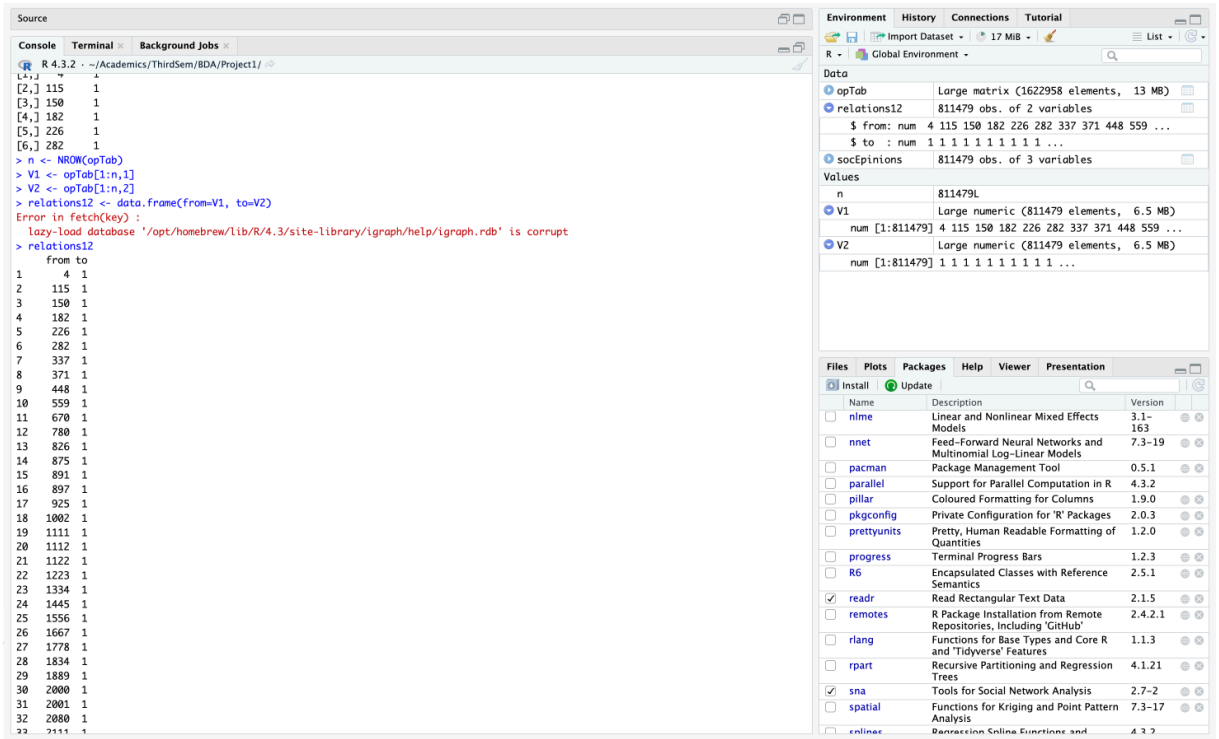

Figure 6

Now generate the graph as shown below.

> Graph <- graph_from_data_frame(relations12, directed=TRUE)

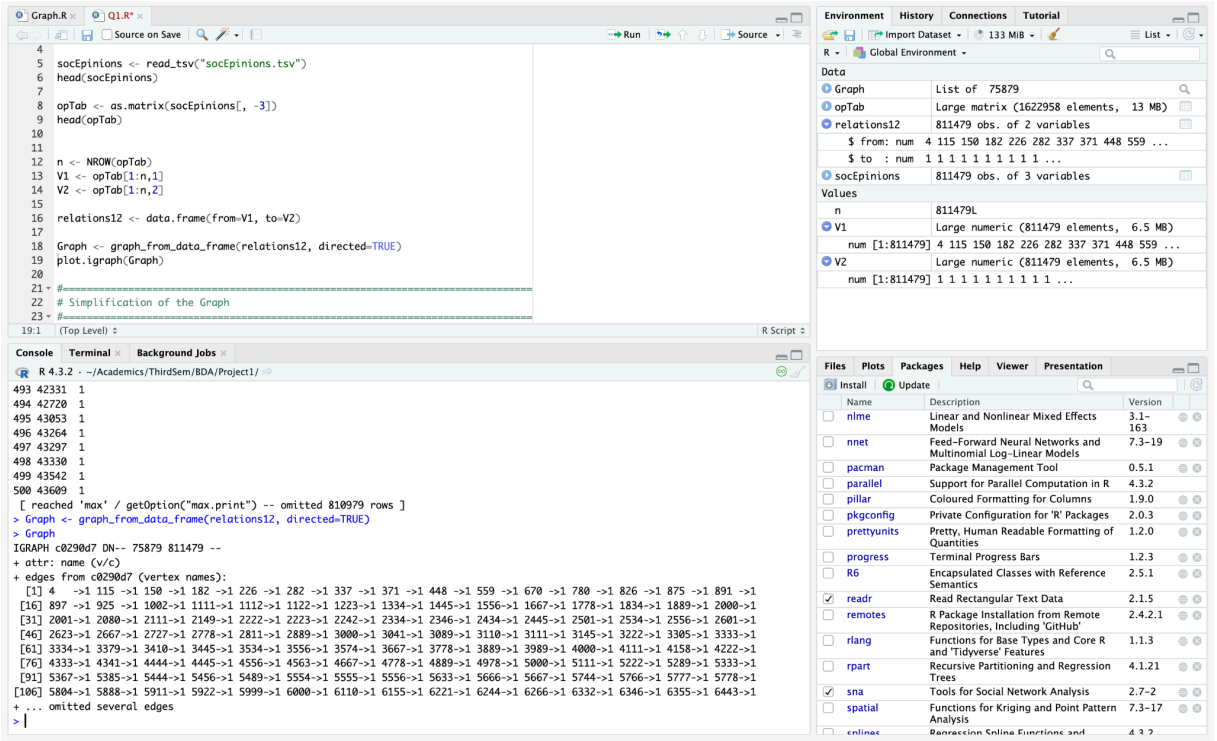

Figure 7

Now generate the plot for the graph. Command: plot.igraph(Graph)

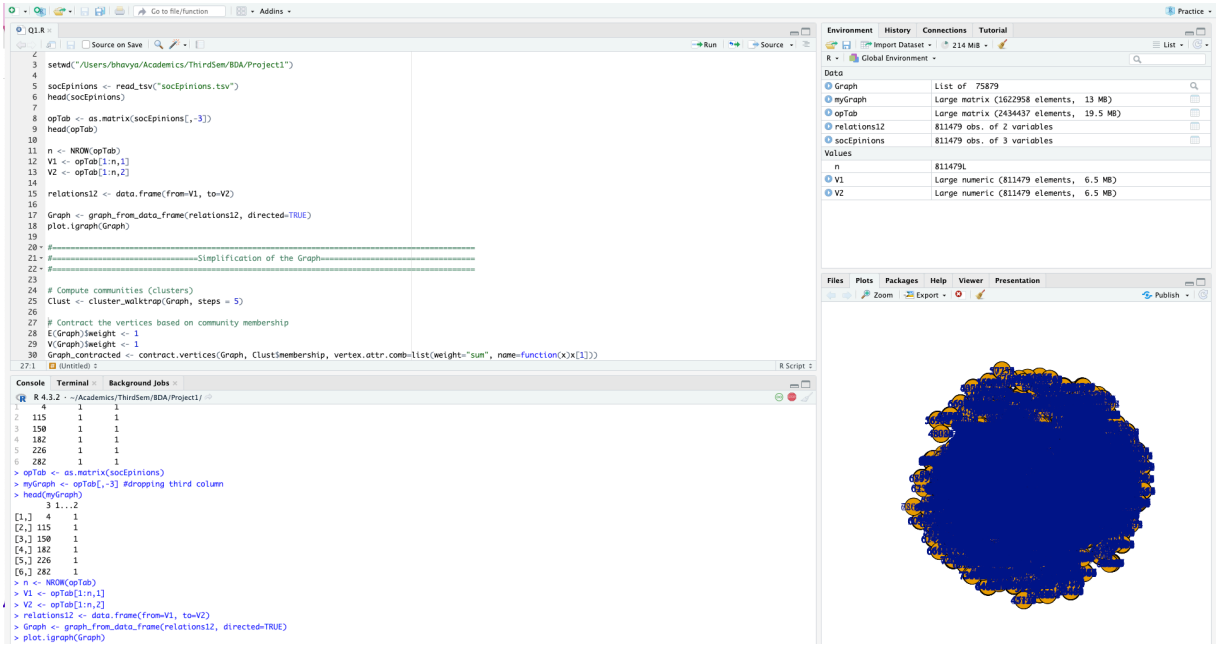

Figure 8

Graph Simplification:

This will create a plot of the graph, which may be difficult to read due to the large number of nodes. So we need to simplify the graph. Now compute the clusters, contract the vertices and simplify the edges.

```
# Compute communities (clusters)
> Clust <- cluster_walktrap(Graph, steps = 5)
# Contract the vertices based on community membership
> E(Graph)$weight <- 1
> V(Graph)$weight <- 1
> Graph_contracted <- contract.vertices(Graph, Clust$membership,
vertex.attr.comb=list(weight="sum", name=function(x)x[1]))

# Simplify the graph by removing multiple edges and loops
> Graph_simplified <- simplify(Graph_contracted, edge.attr.comb=list(weight="sum",
  function(x)length(x)))

# Remove vertices with a degree less than a threshold (e.g., 20)
> Graph_simplified <- delete.vertices(Graph_simplified, which(degree(Graph_simplified) <
  20))

# Plot the simplified graph
> plot.igraph(Graph_simplified)
```

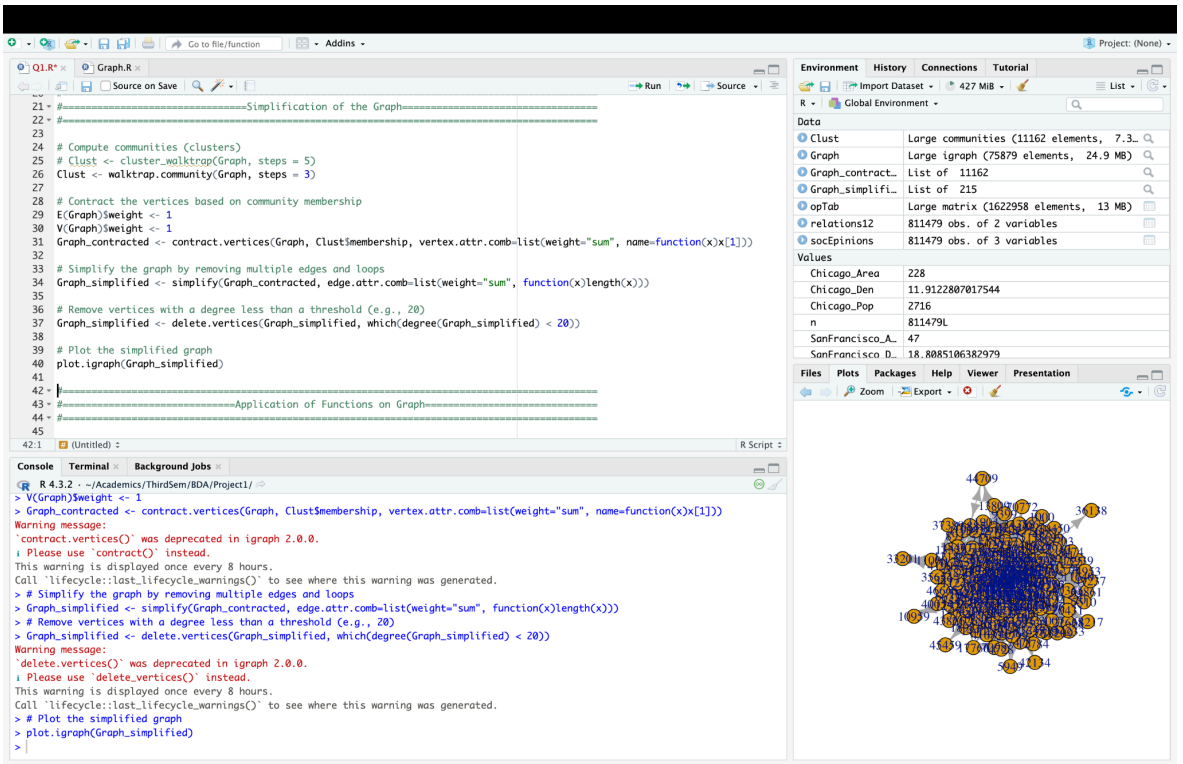

Figure 10

Functions:

- To create arbitrary graphs for experimentation, we use rgraph.
  > aGraph <- rgraph(n=200, m=1, tprob = 0.5, diag = FALSE)
  > aGraph <- rgraph(n=200, m=1, tprob = 0.5, diag = FALSE, mode='undirected',return.as.edgelist=TRUE)
  > myGraph <- graph_from_edgelist(aGraph[,-3], directed=FALSE)
  plot.igraph(myGraph)

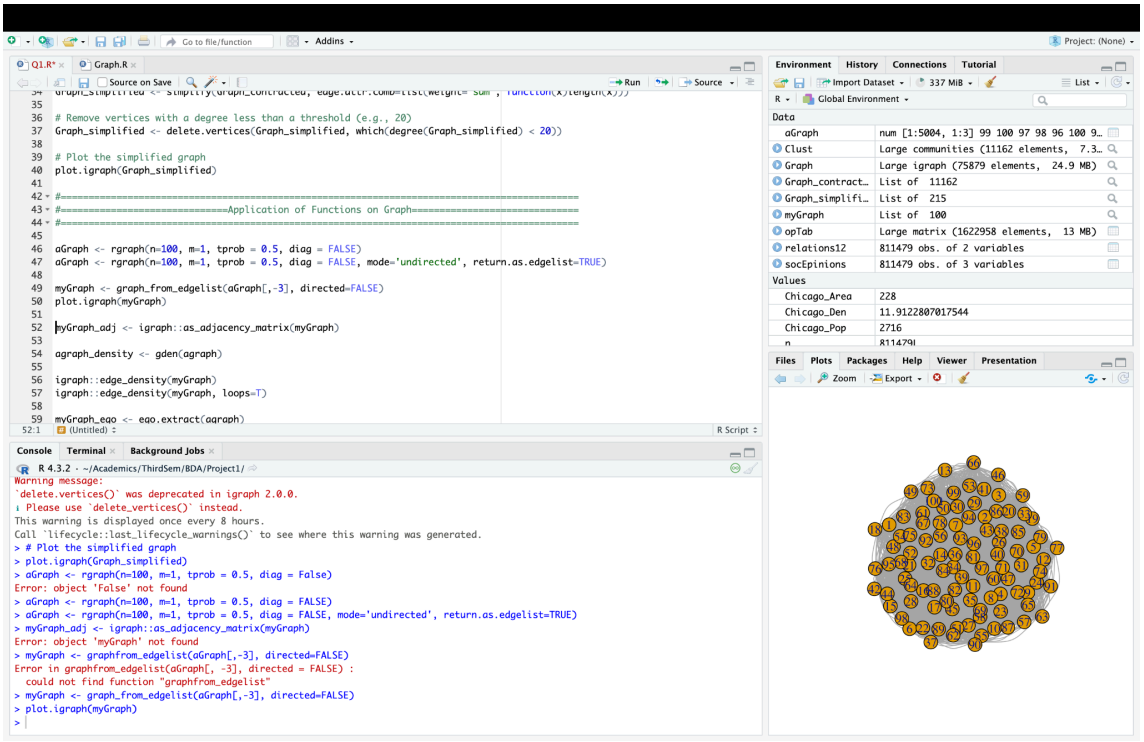

Figure 11

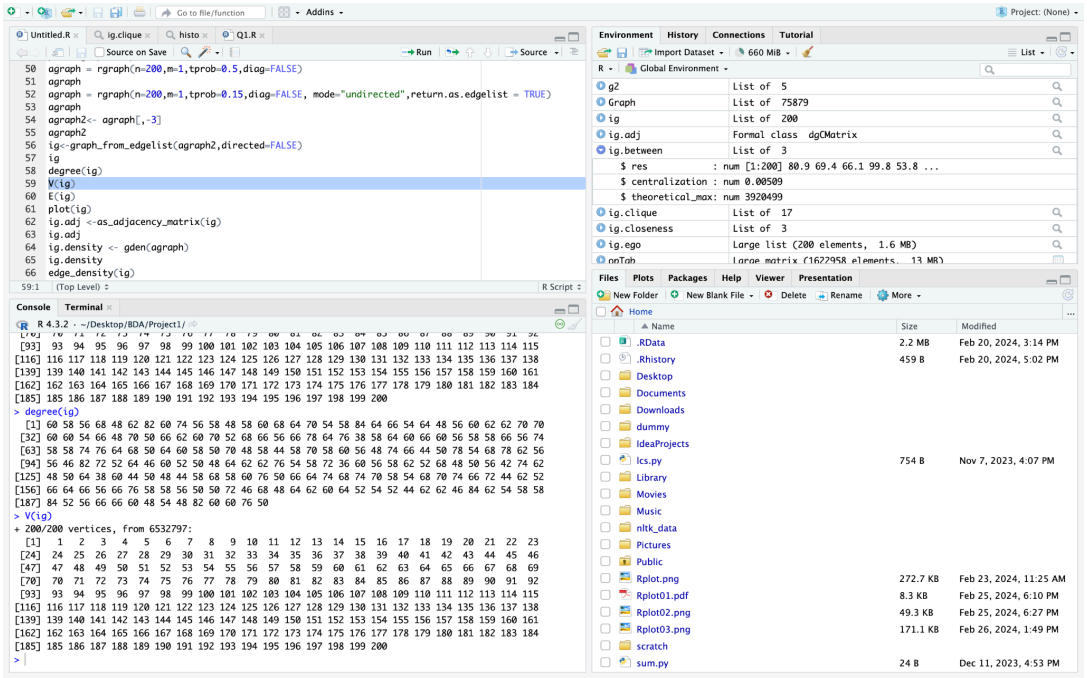

Figure 12

To get the edges of a graph, we use the E function.

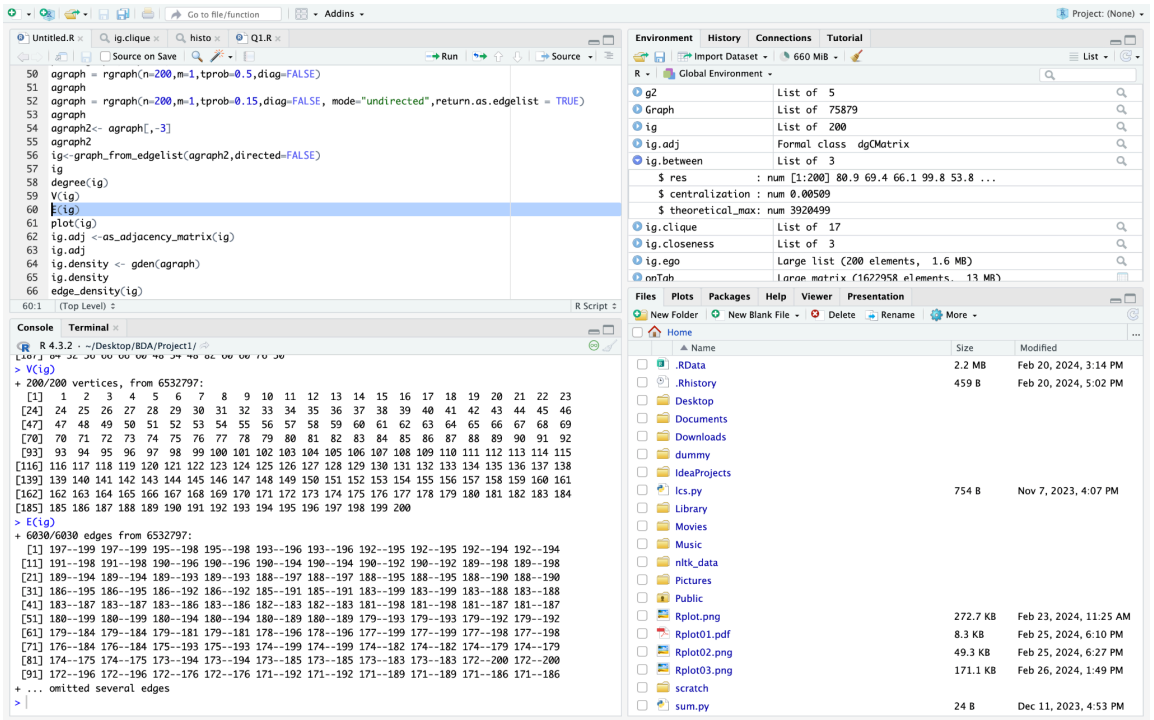

Figure 13

Now to get the adjacency matrix of the graph, use the following command.

```
> myGraph_adj <- igraph::as_adjacency_matrix(myGraph)
```

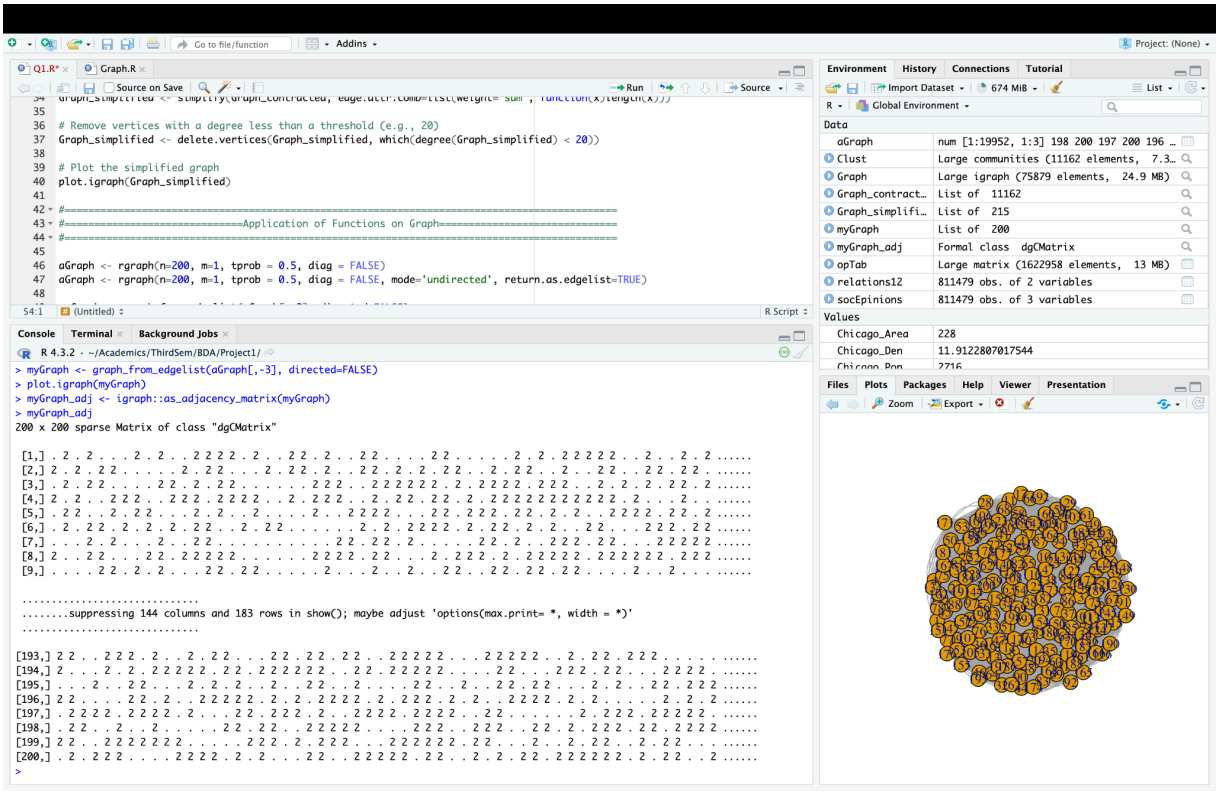

Figure 14

gden is one of the analytic functions which returns the density of the graph which is the ratio of the number of edges and the number of possible edges.

> agraph_density <- gden(agraph)

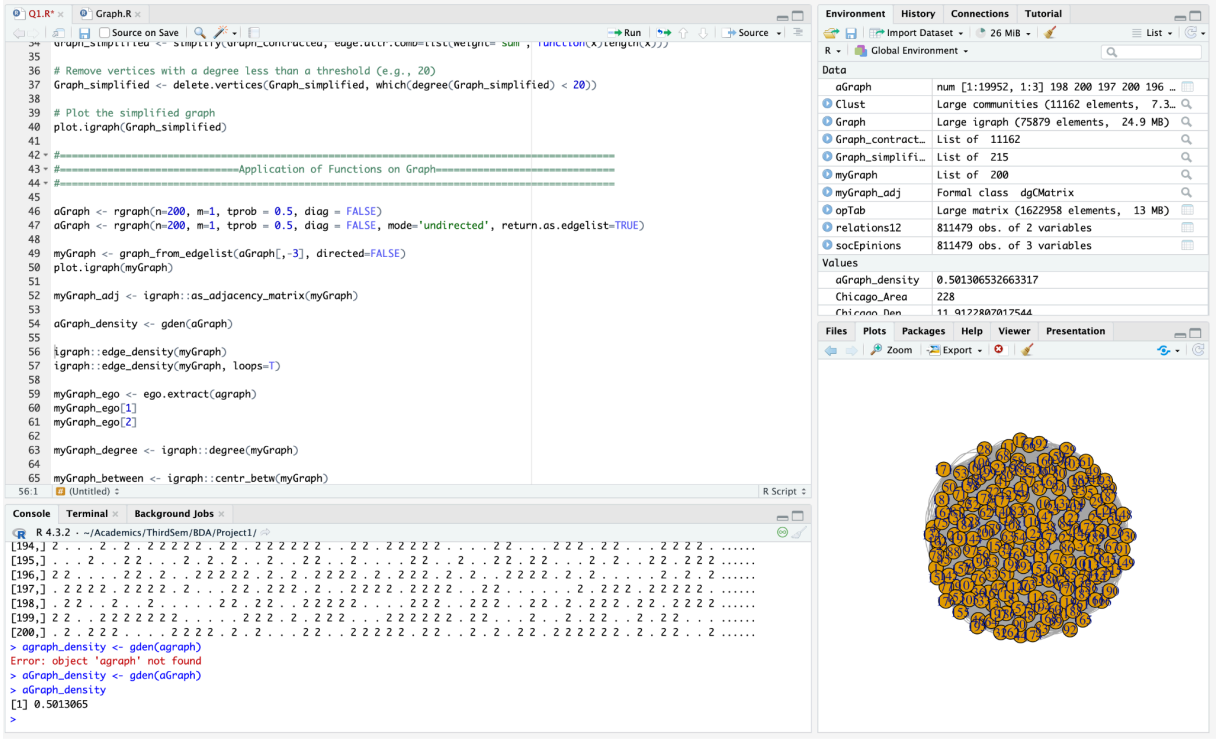

Figure 15

The edge_density() function returns a floating-point value representing the edge density of the graph.

> igraph::edge_density(myGraph)
> igraph::edge_density(myGraph, loops=T)

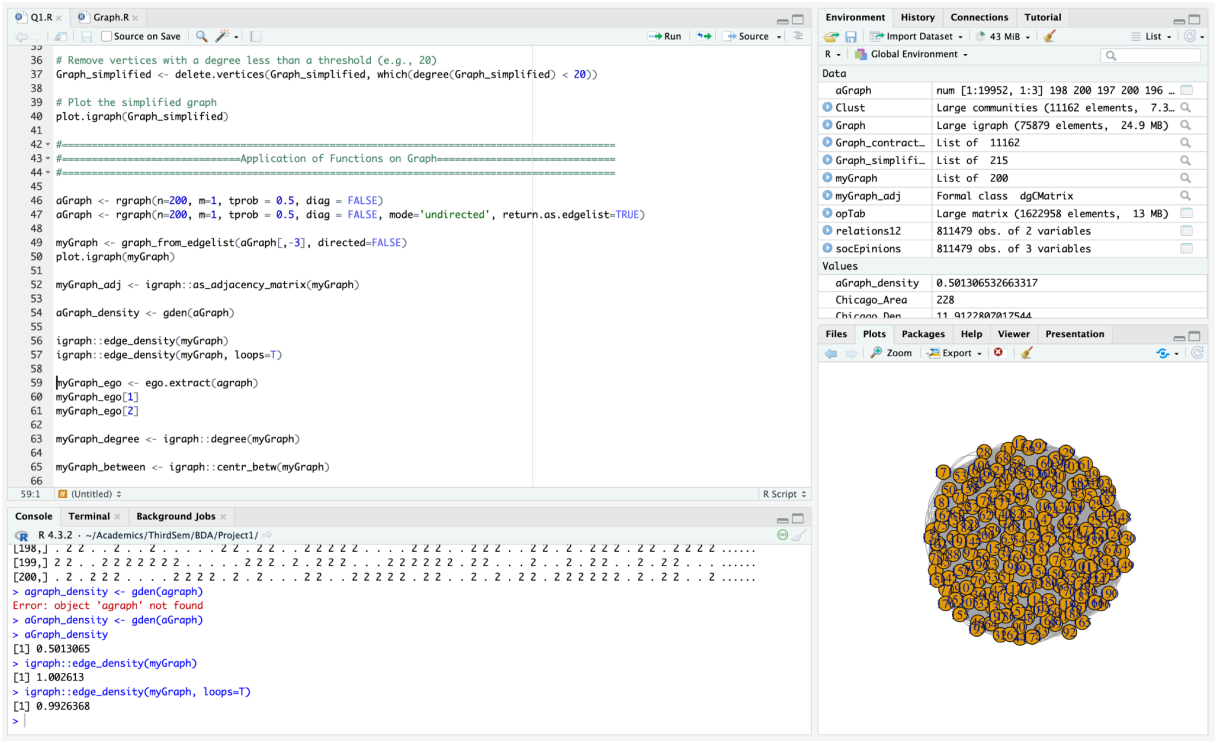

Figure 16

Ego function returns a subgraph object representing the ego network of the specified vertex.

> myGraph_ego <- ego.extract(agraph)
> myGraph_ego[1]
> myGraph_ego[2]

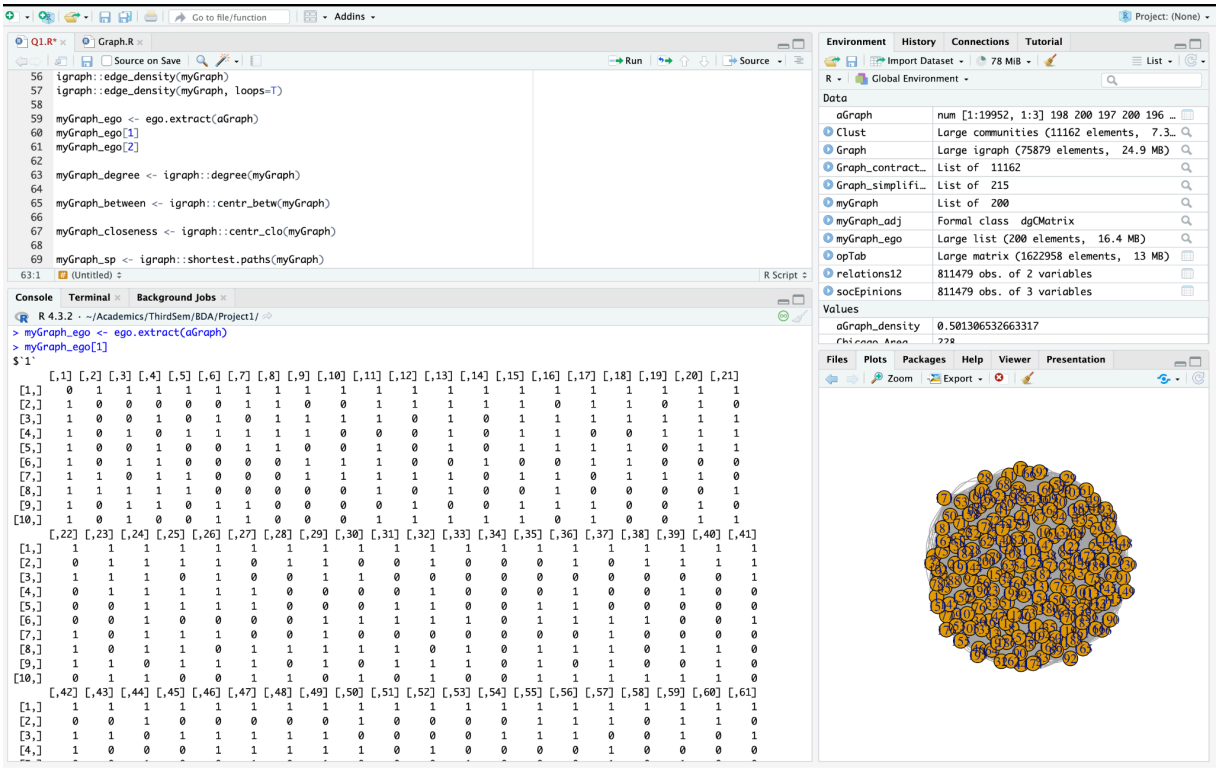

Figure 17

The degree function returns the degree of each node in the graph.

```
> myGraph_degree <- igraph::degree(myGraph)
```

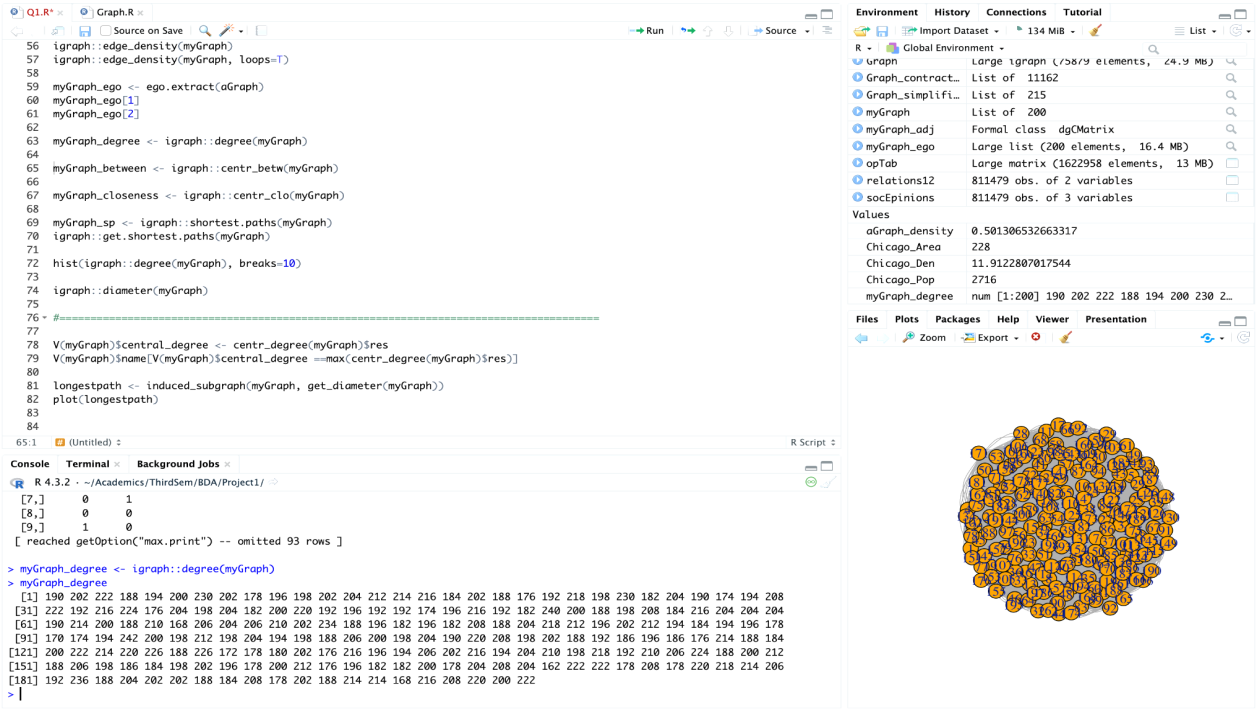

Figure 18

The centr_betw function returns a numeric vector representing the betweenness centrality of each vertex in the graph.

```
> myGraph_between <- igraph::centr_betw(myGraph)
```

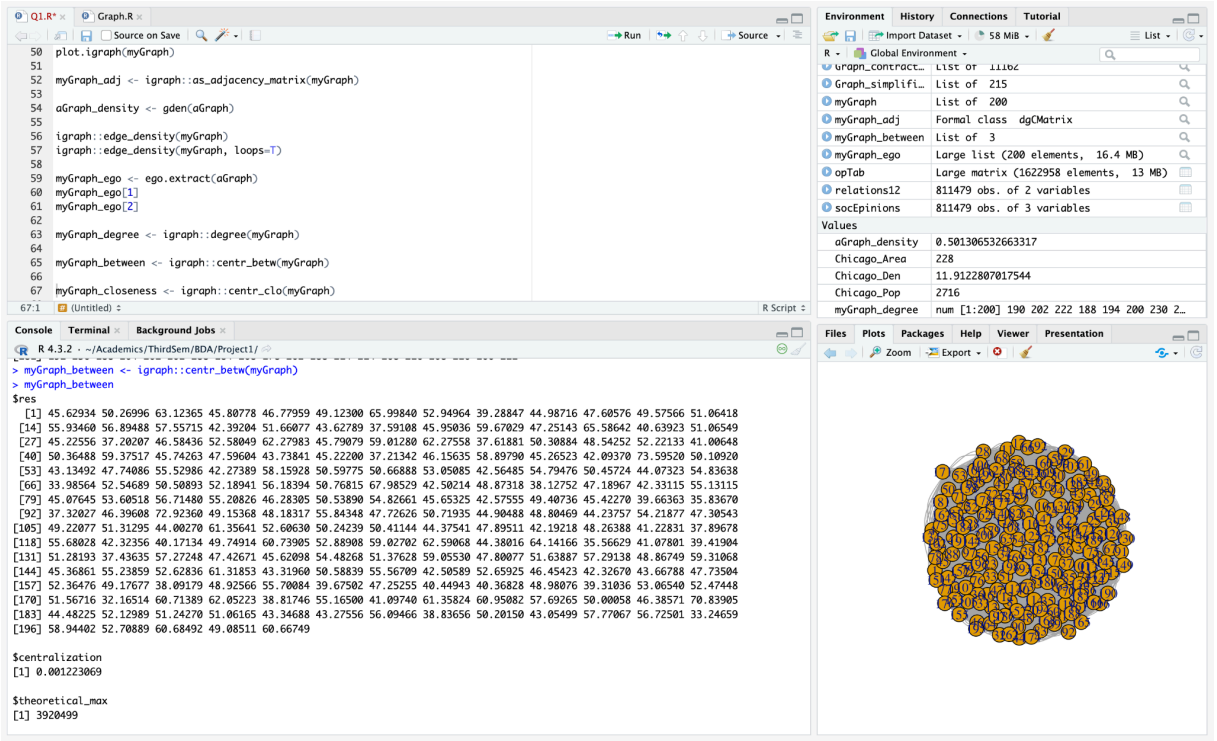

Figure 19

The centr_clo returns a numeric vector representing the closeness centrality of each vertex in the graph. Closeness centrality measures how close a node is to all other nodes in the network, based on the shortest paths. Thus, the more central a node is, the *closer* it is to all other nodes.

```
> myGraph_closeness <- igraph::centr_clo(myGraph)
```

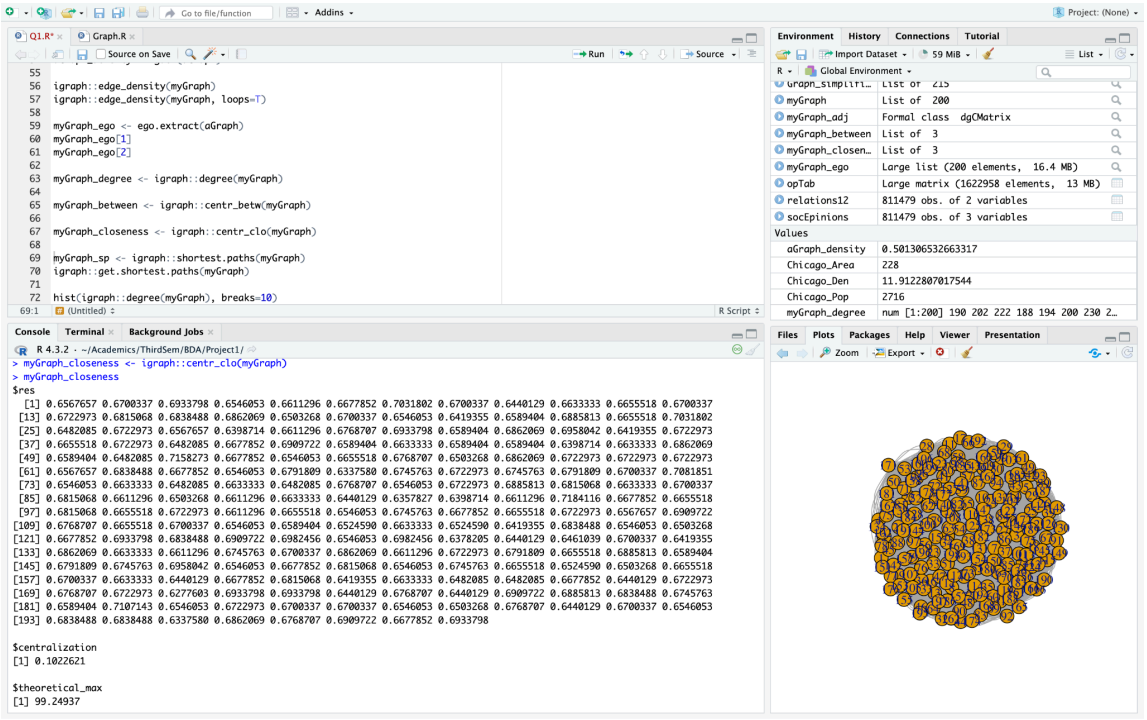

Figure 20

The shortest.paths function is to find the shortest path between the nodes.

> myGraph_sp <- igraph::shortest.paths(myGraph)

> igraph::get.shortest.paths(myGraph)

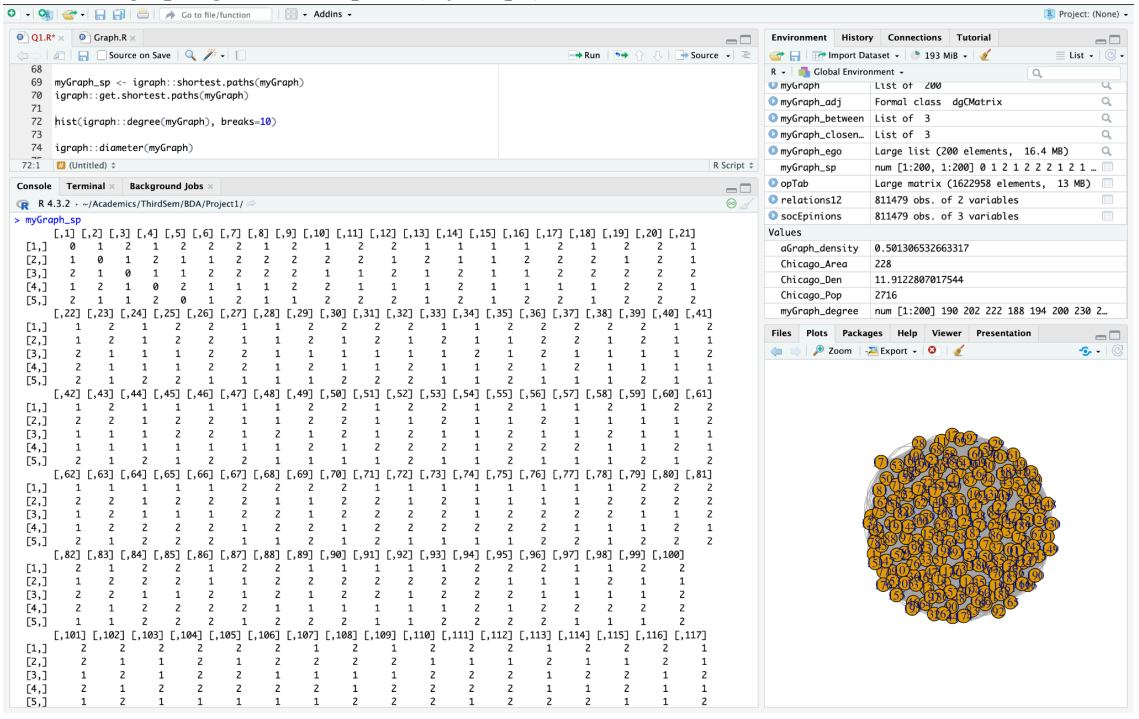

Figure 21

The below command is used to plot the histogram of the graph's degree.

```
> hist(igraph::degree(myGraph), breaks=10)
```

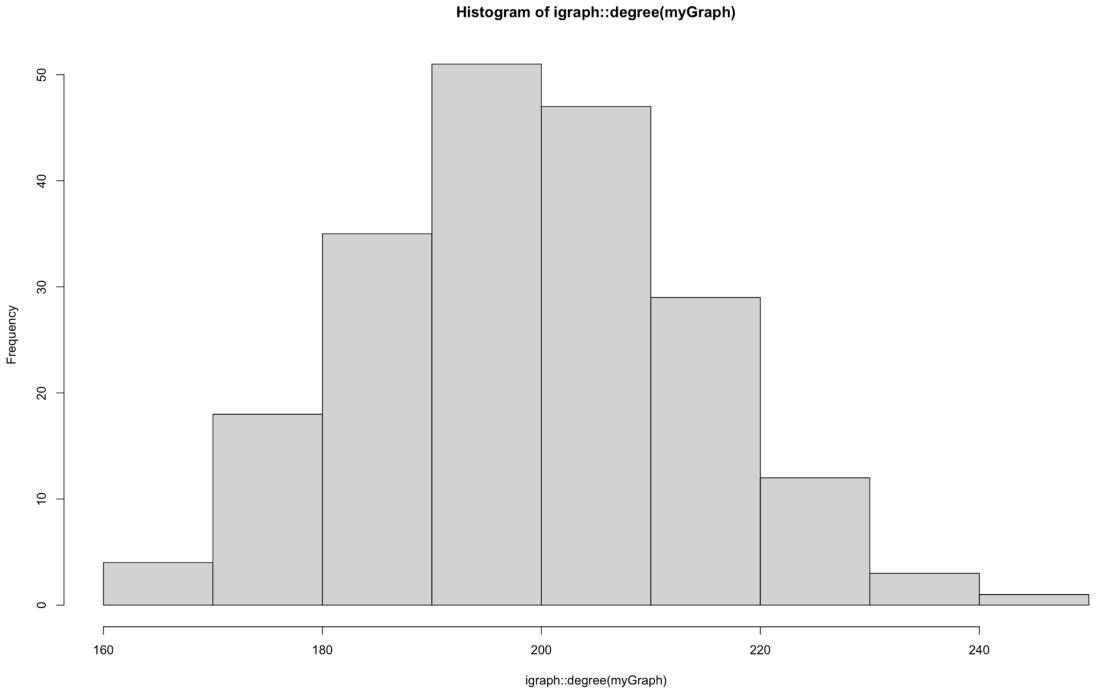

Figure 22

The function diameter returns the numerical diameter of the graph as shown below

```
> igraph::diameter(myGraph)
```

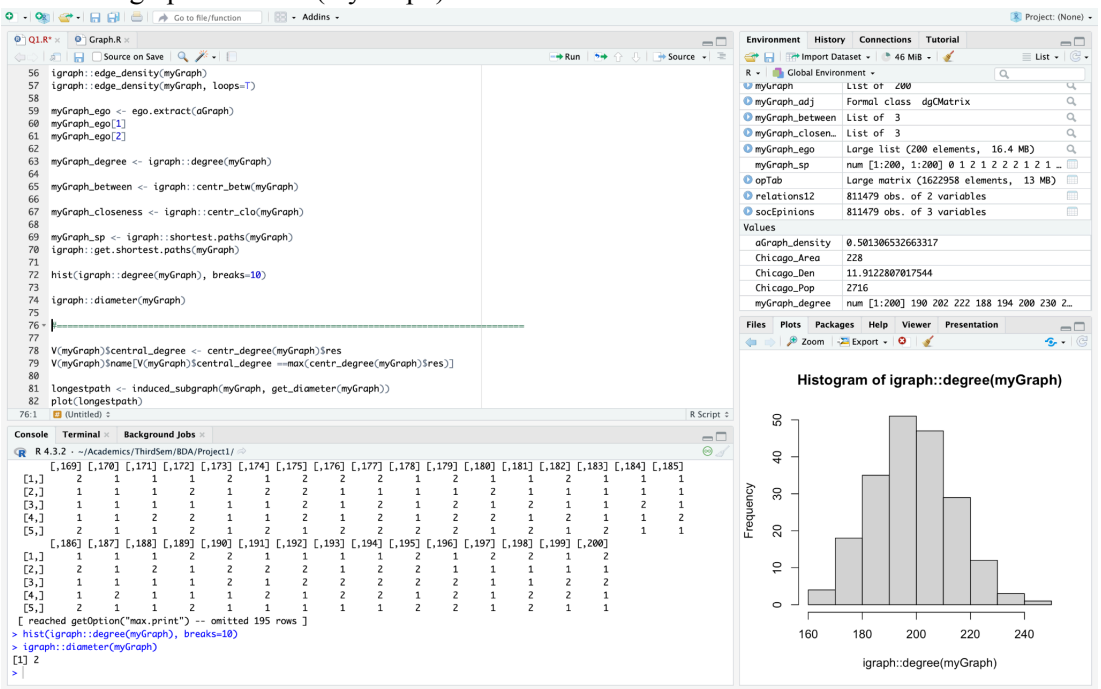

Figure 23

What do node and edge density indicate about the problem space?

Node density and edge density are crucial metrics in graph analytics, offering insights into the structure and complexity of the graph representing a problem space. While the concept of node density is less commonly emphasized in graph theory, it can be interpreted as the ratio of existing nodes to the maximum possible number of nodes within a certain context, indicating the graph's complexity. A higher node density might suggest a problem space with many entities, highlighting its intricate nature.

Edge density, more commonly analyzed, measures the ratio of existing edges to the maximum possible number of edges in the graph. This metric illuminates how interconnected the nodes are. A high edge density signifies a network where nodes are densely connected, suggesting a problem space characterized by significant interdependency or interaction among the entities represented. This could imply that the entities within the network have numerous relationships or dependencies, indicating a tightly knit system.

Both metrics combined offer a nuanced view of the graph's representation of the problem space:

High Node Density: May imply a complex and potentially challenging problem space, with a vast number of entities that could complicate navigation and analysis. Understanding this can be vital for determining the appropriate analytical strategies or algorithms to employ, as it highlights the richness and potential complexity of the network.

High Edge Density: Suggests a robust level of interconnectivity among the entities, potentially indicating a system where interactions or dependencies are paramount. This can be critical for analyses focused on network resilience, communication pathways, or the spread of information or disease, as it points to a network where pathways between entities are abundant.

In summary, understanding node and edge density within a graph provides valuable context for the problem space, guiding the selection of analytical approaches and interpretation of the network's structure and dynamics. High densities might indicate a complex, interconnected system, requiring sophisticated tools and methods for effective analysis and visualization.

Determine the (a) central nodes(s) in the graph, (b) longest path(s), (c) largest clique(s), (d) ego(s), and (e) power centrality.

Central Nodes: The node with maximum central score in the graph is the central node.

```
> most_central <-which.max(betweeness)

> most_central <- which.max(degree(myGraph, mode=”all))
```

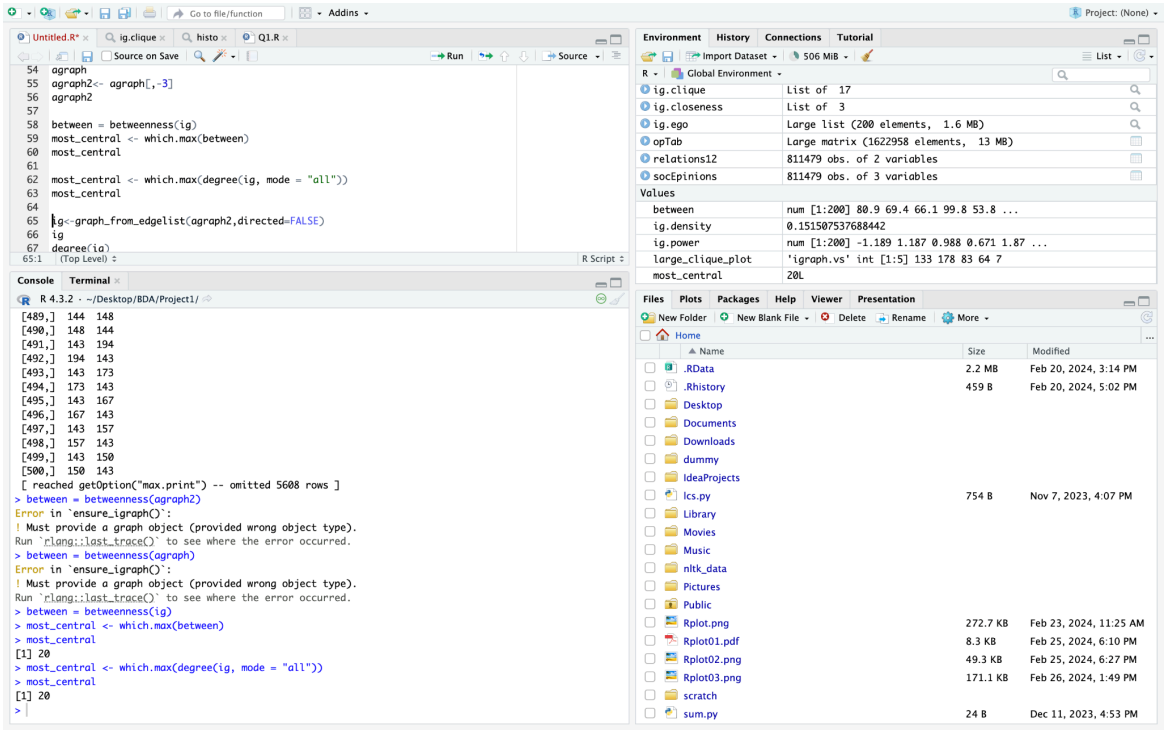

Figure 24

Longest path(s): The longest path of the graph determines the longest connected component of the graph. Use the following commands.

> longestpath <- induced_subgraph(myGraph, get_diameter(myGraph))
> plot(longestpath)

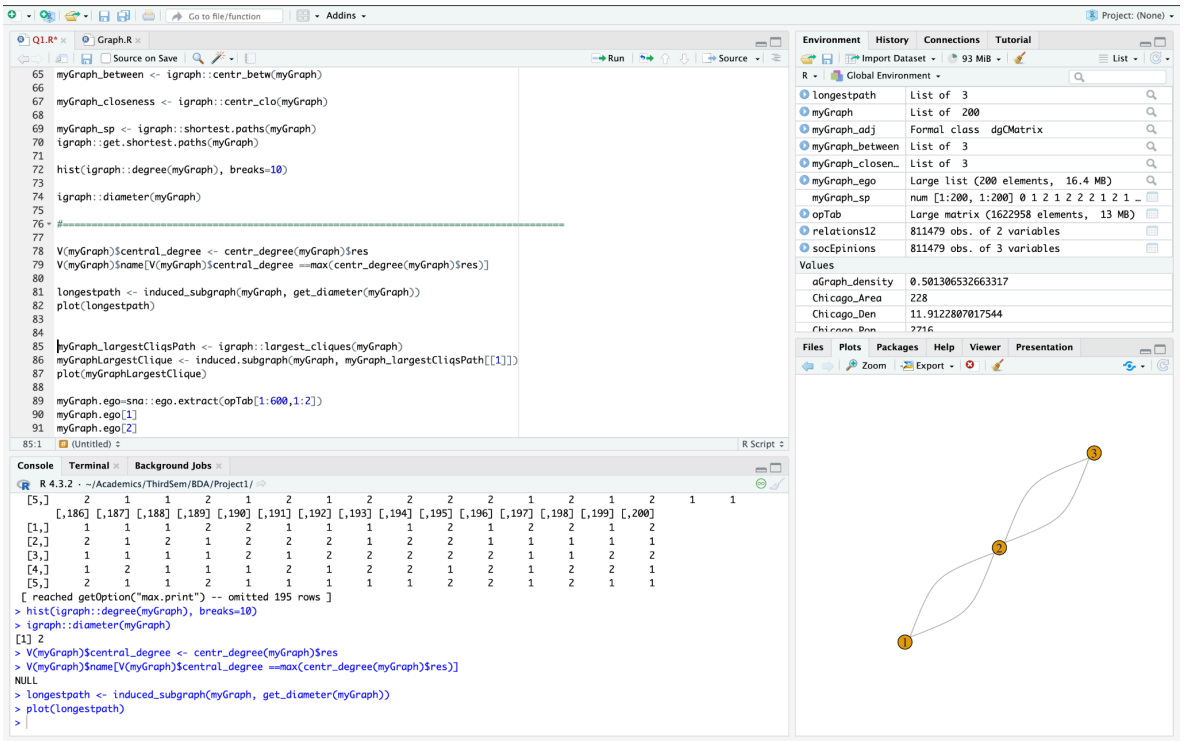

Figure 25

Largest clique(s):

This function returns all the largest cliques in an undirected graph. The following commands are used to get and plot one of the largest cliques.

> Graph.largestCliqsPath <- igraph::largest_cliques(myGraph)

> graphLargestClique <- induced.subgraph(myGraph, Graph.largestCliqsPath[[1]])

> plot(graphLargestClique)

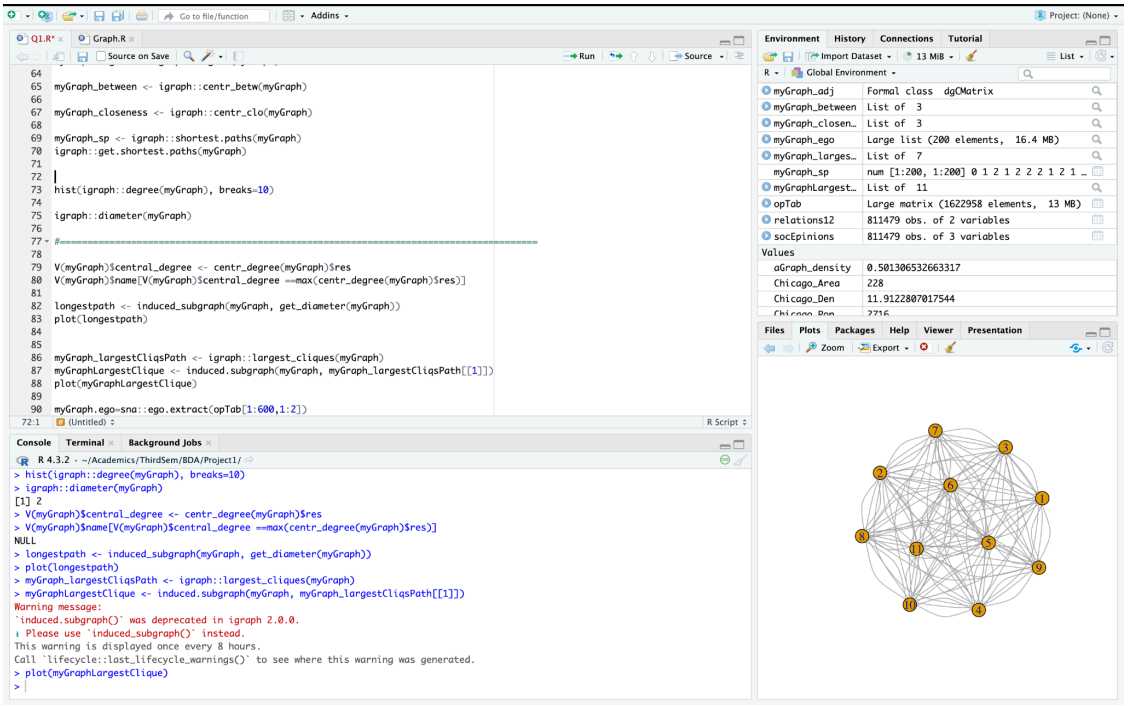

Figure 26

Ego(s):

Ego function returns a subgraph object representing the ego network of the specified vertex. An egocentric network is a type of network graph that focuses on a single node, referred to as the "ego," and the direct connections or relationships that this node has with other nodes, known as "alters. It maps out the personal social network surrounding an individual, known as the ego, by illustrating the connections that individual has with others across the wider network.

> myGraph.ego=sna::ego.extract(opTab[1:600,1:2])
> myGraph.ego[1]

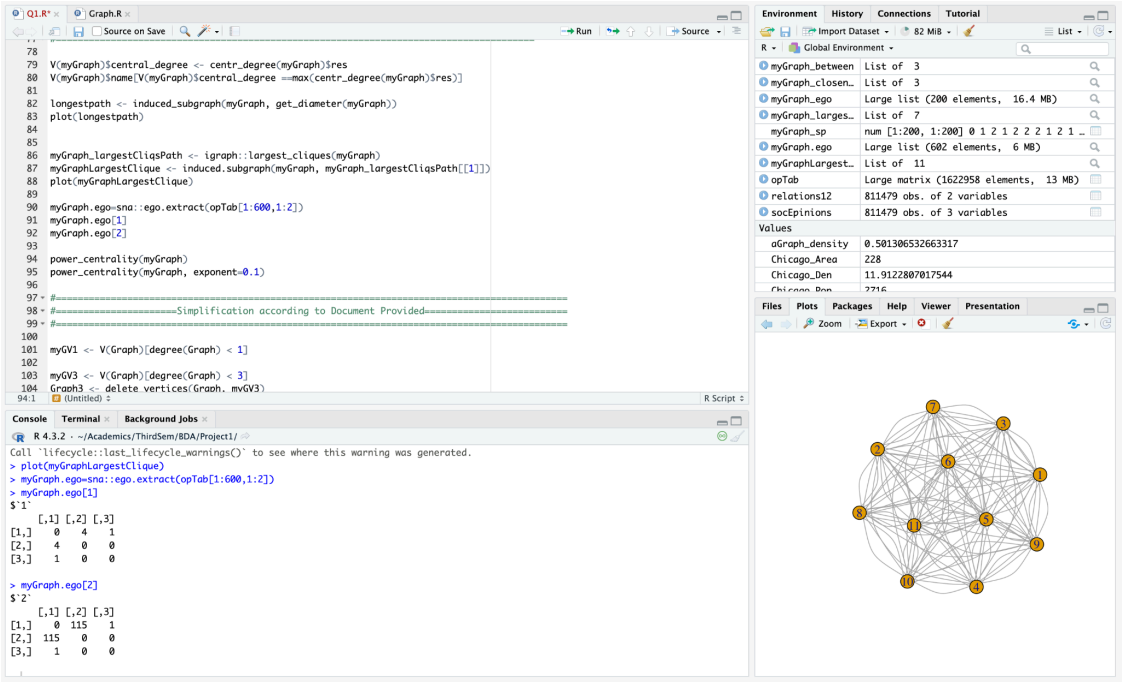

Figure 27

Power centrality:

Bonacich proposed that both centrality and power were a function of the connections of the actors in one's neighborhood. The more connections the actors in the neighborhood have, the more central the node is. The fewer the connections the actors have in the neighborhood, the more powerful the node is. It aims to measure the power or influence a node has within the network's structure.

> power_centrality(myGraph)

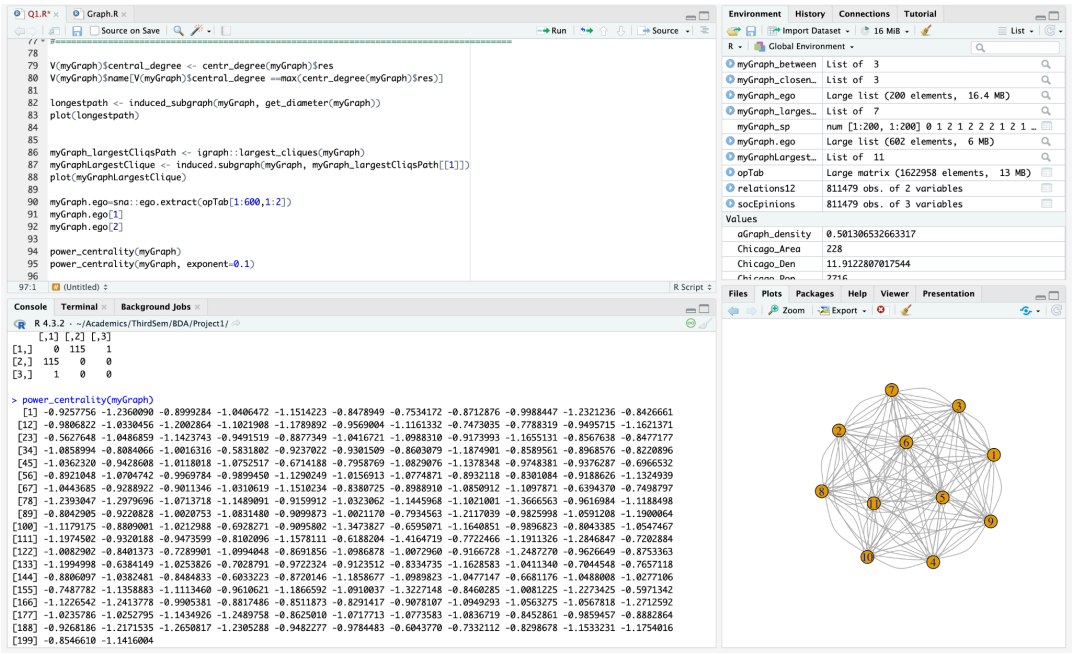

Figure 28

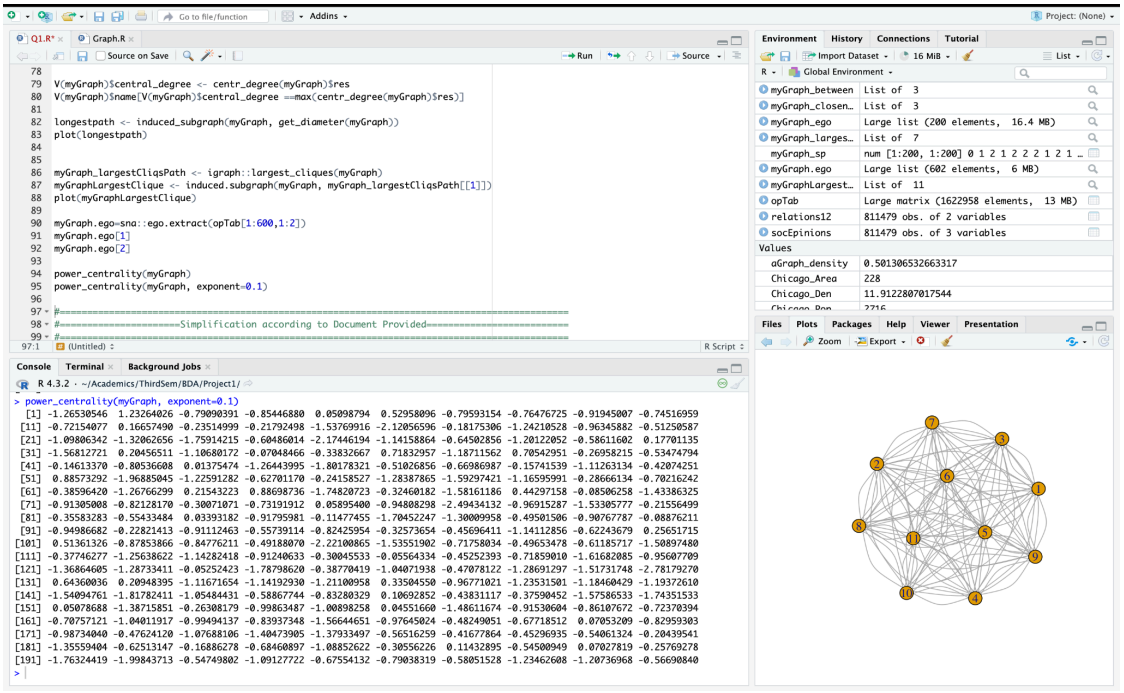

Figure 29

List of R functions Used:

setwd()
install.packages()
library()
read_tsv()
as.matrix()
nrow
data.frame(from , to)
graph_from_data_frame(dataFrame, directed = TRUE)
plot.igraph()
walktrap.community()
degree()
contract.vertices()
simplify()
induced.subgraph()
V()
E()
get.adjacency()
gden()
edge_density()
sna::ego.extract()
centr_betw()
centr_clo()
shortest.paths()
hist()

diameter()
centr_degree()
get_diameter()
largest_cliques()
plot()
ego.extract()
power_centrality()

Discuss what you learned from this project.

Reflecting on the project, it becomes evident that a significant amount of knowledge and experience was gained in various areas:

1. Introduction to R and the `igraph` package: The project served as an excellent introduction to the R programming language, with a particular focus on utilizing the `igraph` package for graph analytics. This experience expanded my skills in programming within R, especially in data manipulation and visualization techniques.

2. Data Handling and Graph Theory Application: Working with a dataset comprising approximately 10 million nodes provided a substantial learning curve in handling and processing large datasets in R. The conversion of this data into a matrix and subsequently into a graph format underscored the practical applications of graph theory, enhancing my understanding of nodes, edges, and their interrelations within a data structure.

3. Graph Analytics Functions: The application of functions from the Introduction to Graph Analytics allowed for a deep dive into analyzing the graph to extract meaningful insights. This part of the project enriched my knowledge on how to apply and interpret various graph analytics functions, including node and edge density calculations, centrality measures, and the identification of significant components such as central nodes, longest paths, largest cliques, egos, and power centrality.

4. Problem Solving and Analytical Skills: Simplifying the graph to a manageable and interpretable form was a crucial phase of the project, significantly enhancing my problem-solving skills. Learning to use the simplify function and other techniques to reduce the complexity of the graph while retaining its essential characteristics was a critical learning outcome.

5. Interpretation and Presentation of Results: The task of presenting findings, including creating and cropping screenshots for the report, improved my skills in data presentation and interpretation. Articulating the significance of these findings in relation to the problem domain provided a deeper understanding of the practical applications of graph analytics.

6. Collaboration and Project Management: As a group project, it also offered insights into collaboration and project management. This included dividing tasks, integrating work from different team members, and adhering to project timelines, all of which are essential skills in any data analytics endeavor.

7. Technical Challenges and Learning: Addressing and resolving issues encountered throughout the project contributed significantly to the learning experience. This involved troubleshooting problems related to data importation, graph creation, function application, or interpretation of results, each of which provided valuable lessons in resilience and adaptability.

In conclusion, this project provided a comprehensive learning experience that ranged from acquiring technical skills in R programming and graph analytics to developing analytical, problem-solving, and presentation abilities. Moreover, it underscored the relevance and applicability of graph theory in

analyzing complex datasets, offering insights that extend beyond the academic realm into practical, real-world applications.