Class Project 1 – Graph Analytics

CSCI 6444 Introduction to big data and analytics

Professor: Stephen Kaisler

Group 2

Shaival Vora(G37099269)

Nithin Raghava Aitha(G43935136)

Part 1: Initial graph

The first step in the project is to download and import the dataset in R Studio. The dataset we are using for this project is “soc-Epinions1_adj.tsv”.

```
> # read the graph analytics project dataset
> graphAnalyticsDataset <- read.delim("C:/Users/anith/Downloads/soc-Epinions1_adj(1).tsv", sep = "\t", header = FALSE)
> #View the dataset
> View(graphAnalyticsDataset)
> # Make a matrix from the dataset
> optab = as.matrix(graphAnalyticsDataset)
```

This code snippet accomplishes several tasks related to handling a dataset named "soc-Epinions1_adj(1).tsv" within the R environment. Initially, it uses the **read.delim()** function to read the contents of the TSV file into a data frame called **graphAnalyticsDataset**, specifying that the columns are separated by tabs and that there is no header row. Lastly, it converts the data frame **graphAnalyticsDataset** into a matrix called **optab** using the **as.matrix()** function, thereby converting the dataset into a matrix format which may be suitable for certain types of analyses or computations, depending on the specific requirements of the analytical tasks.

The second step in the project is to install and import the igraph package.

```
install.packages("igraph")
library(igraph)
```

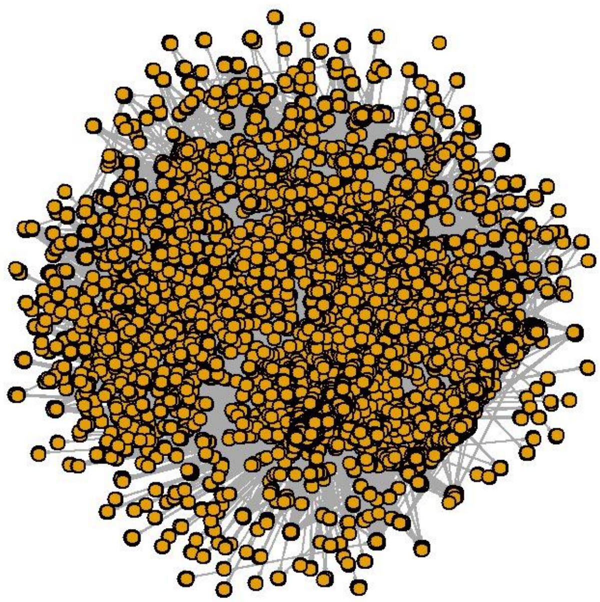
To utilize the igraph package we must first install the package, which could be done by the command “install.packages(“package_name”)”, then we must import the package using command library(package_name).

In the later part of the project, we require another package called “sna”, which could be initialized by the same process.

```
> # Make Vectors V1, V2
> v1 <- optab[1:811480, 1]
> v2 <- optab[1:811480, 2]
> v3 <- optab[1:811480, 3]
> # Make DataFrames
> relations = data.frame(from=v1, v2)
> g <- graph_from_data_frame(relations, directed = TRUE)
> plot(g)
```

Then, we take the columns in the matrix **optab** as vectors v1,v2,v3. Then we create a **data frame** called **relations** using the function **data.frame()** with the vectors **v1,v2.** Then we create a graph **g** from **relations** using the function **graph_from_data_frame()** function.

The function **plot()** is used to plot the graph **g**.



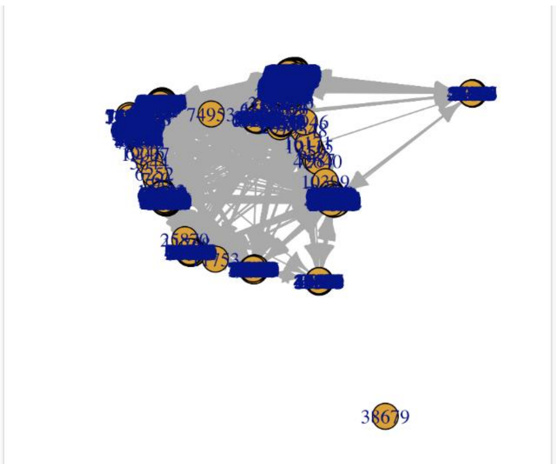
This is the plot of the graph **g** without labels.

Part 2: Graph simplification

Cleaning up the blob graph using brute force:

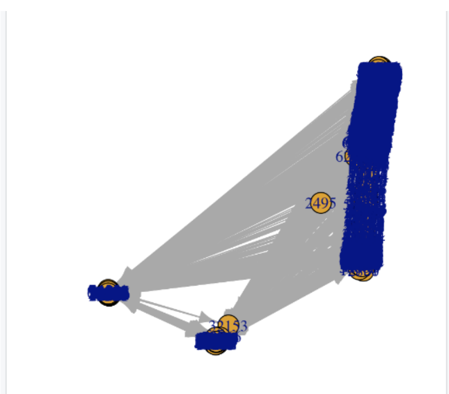
```
> newGV50 <- V(newGraph)[igraph::degree(newGraph)<50]
> newGraph3 <- igraph::delete.vertices(newGraph, newGV50)
> plot(newGraph3)
```

In the above code snippet, we store all the vertices with less than **degree 50** in **newGV50.** Then we delete those vertices from the graph.

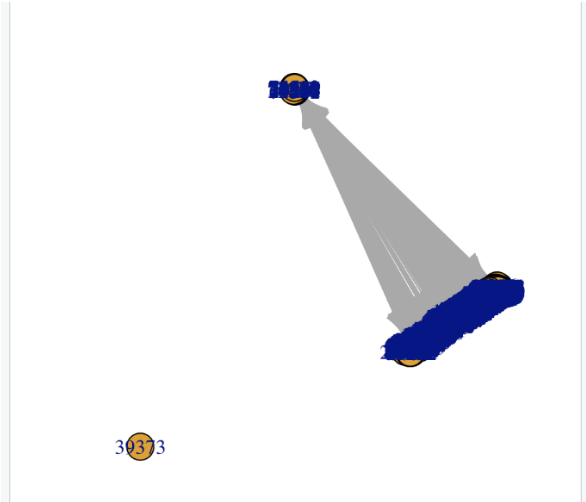


Similarly, we simplify for **degree 150** and **degree 200.**

```
newGV150 <- V(newGraph)[igraph::degree(newGraph)<50]
newGV150

newGraph3 <- igraph::delete.vertices(newGraph, newGV150)
plot(newGraph3)
```



```
newGV200 <- V(newGraph)[igraph::degree(newGraph)<200]
newGV200

newGraph3 <- igraph::delete.vertices(newGraph, newGV200)
plot(newGraph3)
```

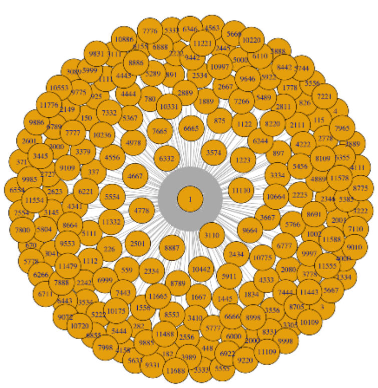


Graph Simplification using fewer data points:

A) Using first 200 nodes

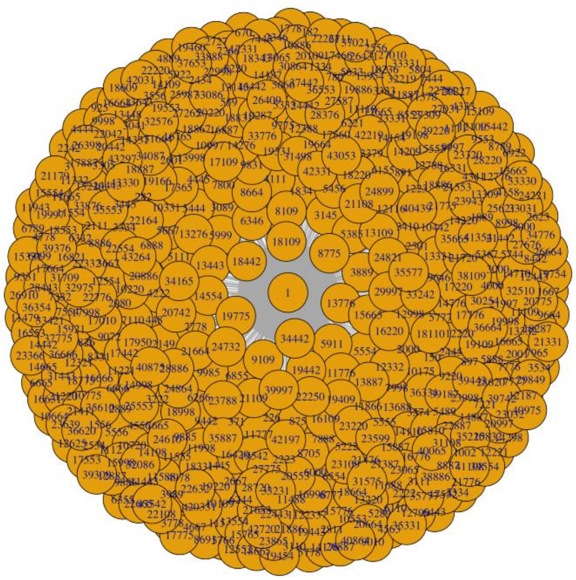
```
> # Plot the graph with first 200 data points
> df200 <- relations[1:200,]
> myGraph200 <- igraph:: graph_from_data_frame(df200)
> plot(myGraph200)
>
```

This code snippet selects the first 200 rows from a data frame called **relations** and assigns it to a new data frame named **df200**. It then creates a graph object **myGraph200** using the **igraph::graph_from_data_frame()** function, which interprets the rows of the data frame as edges of the graph. Subsequently, it plots the graph **myGraph200**. In summary, the code constructs a graph from the first 200 rows of a data frame, facilitating visualization and analysis of the relationship network represented by those rows. Similarly, we can do the same for 500,1000,1500 and 5000 nodes.



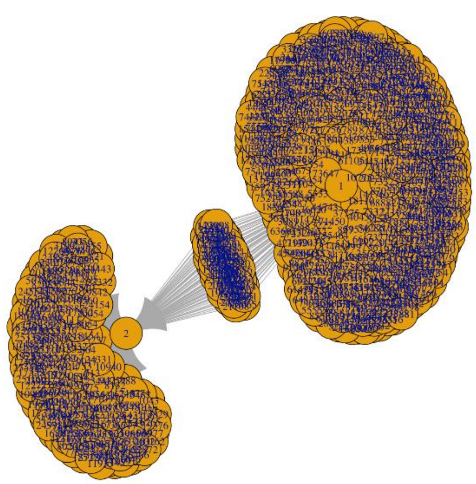
B) Using first 500 nodes

```
> # Plot the graph with first 500 data points
> df500 <- relations[1:500,]
> myGraph500 <- igraph:: graph_from_data_frame(df500)
> plot(myGraph500)
```



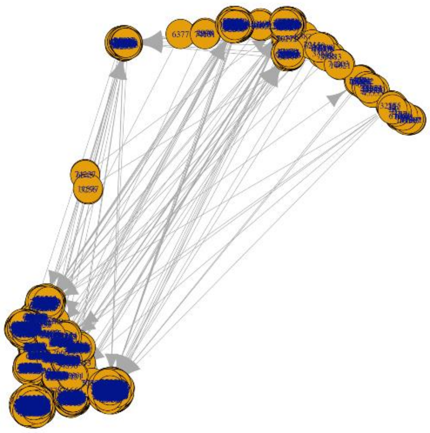
C) Using first 1000 nodes

```
> # Plot the graph with first 1000 data points
> df1000 <- relations[1:1000,]
> myGraph1000 <- igraph:: graph_from_data_frame(df1000)
> plot(myGraph1000)
```



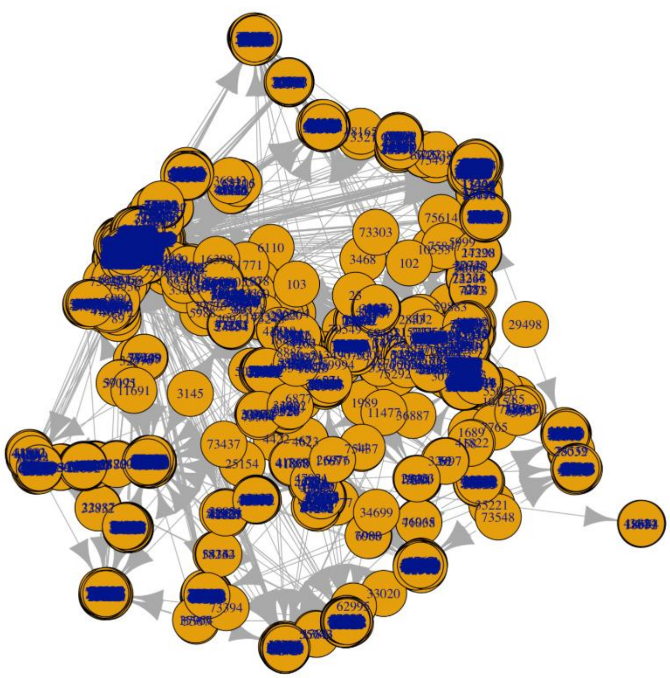
D) Using First 2500 nodes

```
> # Plot the graph with first 2500 data points
> df2500 <- relations[1: 2500,]
> myGraph2500 <- igraph:: graph_from_data_frame(df2500)
> plot(myGraph2500)
```



E) Using first 5000 nodes

```
> # Plot the graph with first 5000 data points
> df5000 <- relations[1: 5000,]
> myGraph5000 <- igraph:: graph_from_data_frame(df5000)
> plot(myGraph5000)
```



PART 3: Graph Analytics Document Functions

A) Str function:

str() function is used to display the internal structure of R objects.

> str(myGraph200)

```
> plot(myGraph200)
> # Str func
> str(myGraph200)
Class 'igraph'  hidden list of 10
 $ : num 201
 $ : logi TRUE
 $ : num [1:200] 0 1 2 3 4 5 6 7 8 9 ...
 $ : num [1:200] 200 200 200 200 200 200 200 200 200 200 ...
 $ : NULL
 $ : NULL
 $ : NULL
 $ : NULL
 $ :List of 4
  ..$ : num [1:3] 1 0 1
  ..$ : Named list()
  ..$ :List of 1
  .. ..$ name: chr [1:201] "3" "4" "115" "150" ...
  ..$ : Named list()
 $ :<environment: 0x12a70b210>
> 
```

B) rgraph:

This represents a graph as an adjacency matrix. An adjacency matrix is an alternate way of representing a graph

agraph <- rgraph(n=100, m=1, tprob=0.5, diag=FALSE)

rgraph() is a function used to generate random graphs in R

n: The number of nodes (vertices) in the graph

m: The number of edges (ties) each node should have.

tprob: The probability of forming an edge between any two nodes.

diag: A logical value indicating whether to allow self-loops

```
    neighborhood, triad.census

> # Note that it represents a graph as an adjacency matrix. An adjacency matrix is an alternate way of representing a graph
> agraph <- rgraph(n=100, m=1, tprob=0.5, diag=FALSE)
> agraph
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19]
 [1,]    0    0    1    0    0    0    1    0    1     1     0     0     0     1     1     0     1     1     1
 [2,]    0    0    0    1    1    1    1    1    1     1     0     1     0     1     1     0     1     1     1
 [3,]    0    0    0    1    1    0    1    0    0     1     0     0     1     0     0     0     1     1     1
 [4,]    0    0    0    0    0    0    0    1    1     1     1     0     1     0     1     1     1     1     0
 [5,]    1    1    0    0    0    0    1    1    0     0     1     0     1     0     1     1     0     0     1
 [6,]    1    1    0    0    1    0    1    0    1     1     0     0     0     1     1     1     0     0     0
 [7,]    0    0    0    0    0    1    0    0    1     0     1     0     1     0     1     0     1     1     0
 [8,]    0    0    1    0    0    0    1    0    0     1     0     1     0     0     1     0     0     1     0
 [9,]    1    0    0    1    0    0    1    0    0     1     0     0     1     1     1     1     1     1     0
[10,]    0    1    1    0    1    1    1    0    0     0     1     1     1     1     1     1     0     0     0
      [,20] [,21] [,22] [,23] [,24] [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37]
 [1,]     1     0     1     1     1     1     0     0     1     0     0     0     1     1     1     0     0     0
 [2,]     1     0     0     1     1     1     1     0     0     1     1     0     0     1     1     1     0     1
 [3,]     0     1     0     1     1     0     0     0     0     1     1     1     1     1     0     1     1     0
 [4,]     0     1     1     0     1     1     1     0     1     1     0     1     0     1     1     0     0     1
 [5,]     0     1     1     0     1     1     1     0     0     1     0     1     0     1     0     0     0     1
 [6,]     1     1     1     0     1     0     0     1     0     0     0     0     1     0     1     1     1     1
 [7,]     0     0     1     0     0     1     0     0     1     1     1     0     1     0     0     1     1     1
 [8,]     1     1     1     1     1     0     1     1     0     0     1     1     0     0     0     1     1     1
 [9,]     1     1     1     1     1     0     1     1     0     0     0     1     0     1     0     1     1     1
[10,]     0     0     1     1     0     0     0     1     0     0     1     0     1     1     0     1     0     1
      [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50] [,51] [,52] [,53] [,54] [,55]
 [1,]     0     1     1     1     1     0     0     0     0     0     0     1     1     1     0     1     1     1
 [2,]     1     1     1     1     0     1     1     0     0     1     0     1     0     0     0     1     1     1
 [3,]     1     1     0     0     1     1     0     0     1     1     1     0     0     1     1     1     0     1
 [4,]     0     0     1     0     1     1     1     0     1     0     0     1     0     1     1     1     1     1
 [5,]     0     0     0     0     0     0     0     1     1     0     1     0     1     0     1     0     1     0
 [6,]     0     1     1     1     1     1     1     0     1     0     0     1     1     1     0     0     1     0
 [7,]     1     0     1     0     1     1     0     0     1     1     1     1     0     0     1     0     1     1
 [8,]     1     0     0     0     1     0     0     1     0     0     0     1     1     1     1     0     1     1
 [9,]     1     0     1     0     0     1     1     0     1     1     0     0     1     0     0     0     1     1
[10,]     0     0     0     1     1     0     1     0     1     1     0     1     0     0     1     0     0     1
      [,56] [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73]
 [1,]     1     0     1     0     0     1     1     0     0     1     0     0     0     0     0     1     1     0
 [2,]     0     1     1     0     1     1     0     0     1     1     0     0     0     0     0     1     0     1
 [3,]     1     1     0     0     0     0     1     1     1     0     0     0     0     0     1     0     0     0
 [4,]     0     0     1     0     0     1     0     1     0     0     0     1     1     1     0     1     0     0
 [5,]     1     1     0     0     1     1     1     1     0     1     0     1     1     1     0     1     0     0
 [6,]     1     1     1     1     1     0     1     1     1     1     0     1     1     0     0     0     1     1
 [7,]     1     1     1     1     0     1     1     1     1     1     0     1     1     0     1     0     1     0
 [8,]     1     0     0     0     1     0     1     0     0     0     1     1     1     0     1     1     0     1
 [9,]     0     0     0     1     0     0     0     1     1     1     0     0     1     1     1     0     0     0
[10,]     1     0     1     1     1     0     1     0     0     0     0     0     0     0     1     0     1     0
      [,74] [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90] [,91]
 [1,]     1     1     1     1     0     1     0     0     1     1     1     1     0     1     1     1     0     1
 [2,]     0     1     0     0     0     1     1     0     1     0     0     1     0     0     1     0     1     0
 [3,]     0     0     1     1     1     0     0     1     0     1     1     0     0     0     0     1     1     0
 [4,]     0     0     0     1     1     1     1     0     0     1     1     1     0     1     0     0     1     0
 [5,]     1     1     0     1     1     0     1     1     0     1     1     0     0     1     0     1     0     1
 [6,]     1     1     0     0     0     0     0     0     1     1     0     0     0     1     0     1     1     1
 [7,]     0     1     1     0     0     1     1     1     1     0     1     0     0     0     1     0     0     0
 [8,]     0     1     0     1     0     0     1     0     1     1     1     0     1     1     0     1     0     0
 [9,]     0     1     0     0     0     1     0     1     1     1     1     0     0     1     0     0     0     1
[10,]     0     0     1     1     1     0     1     1     1     0     1     0     0     0     1     1     0     1
      [,92] [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100]
 [1,]     0     0     1     0     0     0     1     1      0
 [2,]     0     1     1     0     0     0     0     1      1
 [3,]     0     0     1     1     0     1     1     1      1
 [4,]     0     0     1     0     0     1     0     0      0
 [5,]     0     0     1     0     0     0     1     0      0
 [6,]     0     1     1     1     1     1     1     0      0
 [7,]     1     1     1     0     0     0     0     0      0
 [8,]     1     1     0     1     0     0     0     0      1
 [9,]     0     1     0     0     1     0     1     0      1
[10,]     1     0     0     0     1     0     1     1      0
 [ reached getOption("max.print") -- omitted 90 rows ]
> 
```

C) Return the rgraph as an edgeList

agraph1 <- rgraph(n=100, m=1, tprob=0.15,diag=FALSE, mode = "undirected", return.as.edgelist=TRUE)

mode = “undirected” means no arrows on the edges

EdgeList: It is essentially a list of pairs of nodes that are connected by edges.

```
> agraph1 <- rgraph(n=100, m=1, tprob=0.15,diag=FALSE, mode = "undirected", return.as.edgelist=TRUE)
> agraph1
      [,1] [,2] [,3]
 [1,]   95   97    1
 [2,]   97   95    1
 [3,]   94   95    1
 [4,]   95   94    1
 [5,]   92   98    1
 [6,]   98   92    1
 [7,]   92   93    1
 [8,]   93   92    1
 [9,]   91   96    1
[10,]   96   91    1
[11,]   91   95    1
[12,]   95   91    1
[13,]   91   94    1
[14,]   94   91    1
[15,]   90   96    1
[16,]   96   90    1
[17,]   90   93    1
[18,]   93   90    1
[19,]   89   96    1
[20,]   96   89    1
[21,]   89   90    1
[22,]   90   89    1
[23,]   88  100    1
[24,]  100   88    1
[25,]   88   93    1
[26,]   93   88    1
[27,]   88   91    1
[28,]   91   88    1
[29,]   87   96    1
[30,]   96   87    1
[31,]   87   94    1
[32,]   94   87    1
[33,]   87   88    1
[34,]   88   87    1
[35,]   83   96    1
[36,]   96   83    1
[37,]   83   92    1
[38,]   92   83    1
[39,]   83   86    1
[40,]   86   83    1
[41,]   82   95    1
[42,]   95   82    1
[43,]   82   87    1
[44,]   87   82    1
[45,]   82   85    1
[46,]   85   82    1
[47,]   82   84    1
```

D) Delete the third column

agraph[,-3] means keep all the rows, and all columns except the third.

agraph2 <- agraph1[, -3]

```
> # agraph[,-3] means keep all the rows, and all columns except the third.
> agraph2 <- agraph1[, -3]
> agraph2
      [,1] [,2]
 [1,]   99  100
 [2,]  100   99
 [3,]   97  100
 [4,]  100   97
 [5,]   95   98
 [6,]   98   95
 [7,]   94   97
 [8,]   97   94
 [9,]   93   98
[10,]   98   93
[11,]   93   95
[12,]   95   93
[13,]   93   94
[14,]   94   93
[15,]   92  100
[16,]  100   92
[17,]   91  100
[18,]  100   91
[19,]   91   94
[20,]   94   91
[21,]   90   97
[22,]   97   90
[23,]   90   93
[24,]   93   90
[25,]   89   98
[26,]   98   89
[27,]   89   97
[28,]   97   89
[29,]   89   96
[30,]   96   89
[31,]   87   94
[32,]   94   87
[33,]   86   99
[34,]   99   86
[35,]   86   89
[36,]   89   86
[37,]   85   96
[38,]   96   85
[39,]   85   86
[40,]   86   85
[41,]   84   94
[42,]   94   84
[43,]   84   85
[44,]   85   84
[45,]   83  100
[46,]  100   83
[47,]   83   98
[48,]   98   83
[49,]   83   96
[50,]   96   83
[51,]   82   97
[52,]   97   82
[53,]   82   92
```

E) Graph Object from edgeList

convert the edgelist to an igraph object. For this we will use graph_from_edgelist

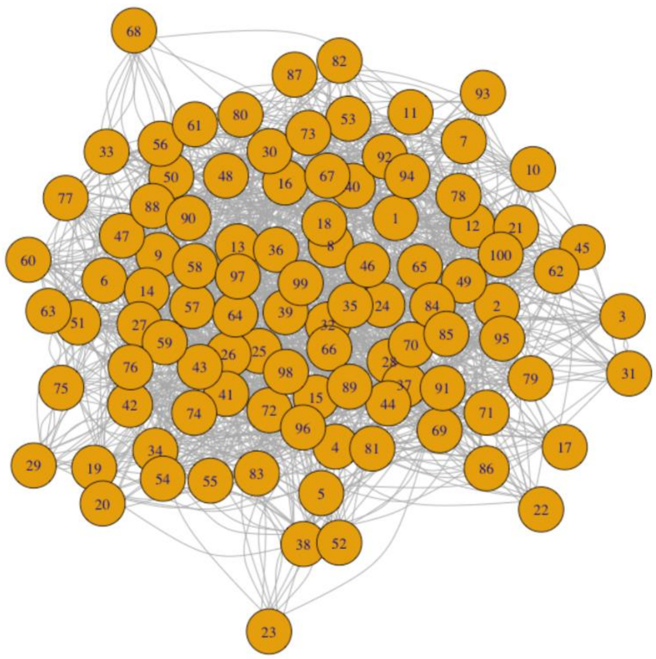
agraph3 <- graph_from_edgelist(agraph2, directed = FALSE)

agraph3 will be a graph object created from the edge list agraph2, with edges indicating the connections between nodes.

Agraph3

```
> agraph3 <- graph_from_edgelist(agraph2, directed = FALSE)
> agraph3
IGRAPH 9310e34 U--- 100 1488 --
+ edges from 9310e34:
  [1] 99--100 99--100 97--100 97--100 95-- 98 95-- 98 94-- 97 94-- 97 93-- 98 93-- 98 93-- 95 93-- 95 93-- 94 93-- 94 92--100 92--100 91--100 91--100
 [19] 91-- 94 91-- 94 90-- 97 90-- 97 90-- 93 90-- 93 89-- 98 89-- 98 89-- 97 89-- 97 89-- 96 89-- 96 87-- 94 87-- 94 86-- 99 86-- 99 86-- 89 86-- 89
 [37] 85-- 96 85-- 96 85-- 86 85-- 86 84-- 94 84-- 94 84-- 85 84-- 85 83--100 83--100 83-- 98 83-- 98 83-- 96 83-- 96 82-- 97 82-- 97 82-- 92 82-- 92
 [55] 82-- 90 82-- 90 81-- 96 81-- 96 81-- 86 81-- 86 80-- 98 80-- 98 80-- 93 80-- 93 79--100 79--100 79-- 86 79-- 86 78-- 99 78-- 99 78-- 94 78-- 94
 [73] 78-- 93 78-- 93 78-- 89 78-- 89 78-- 85 78-- 85 78-- 82 78-- 82 78-- 81 78-- 81 77-- 97 77-- 97 77-- 88 77-- 88 76-- 99 76-- 99 76-- 96 76-- 96
 [91] 76-- 92 76-- 92 76-- 83 76-- 83 76-- 77 76-- 77 75-- 96 75-- 96 74-- 99 74-- 99 74-- 91 74-- 91 74-- 80 74-- 80 74-- 76 74-- 76 73--100 73--100
[109] 73-- 97 73-- 97 73-- 96 73-- 96 73-- 92 73-- 92 73-- 88 73-- 88 72-- 99 72-- 99 72-- 92 72-- 92 72-- 90 72-- 90 72-- 83 72-- 83 72-- 76 72-- 76
[127] 72-- 74 72-- 74 71-- 96 71-- 96 71-- 89 71-- 89 71-- 85 71-- 85 71-- 84 71-- 84 71-- 72 71-- 72 70-- 97 70-- 97 70-- 95 70-- 95 69--100 69--100
[145] 69-- 94 69-- 94 69-- 81 69-- 81 69-- 79 69-- 79 69-- 74 69-- 74 69-- 70 69-- 70 68-- 88 68-- 88 67-- 95 67-- 95 67-- 91 67-- 91 67-- 90 67-- 90
+ ... omitted several edges
>
```

F) plot(agraph3)



G) Get all the vertices of the graph

For this we will use the V function which will take the graph object and return the set vertices of the graph

V(agraph3)

```
> #To get the vertices of a graph, we use the V function:
> V(agraph3)
+ 100/100 vertices, from 9310e34:
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37
 [38]  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74
 [75]  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
```

H) Get all the Edges of the graph

For this we will use the E function which will take the graph object and return the set of Edgesof the graph

E(agraph3)

```
> #To get the edges of a graph, we use the E function:
> E(agraph3)
+ 1488/1488 edges from 9310e34:
  [1] 99--100 99--100 97--100 97--100 95-- 98 95-- 98 94-- 97 94-- 97 93-- 98 93-- 98 93-- 95 93-- 95 93-- 94 93-- 94 92--100 92--100 91--100 91--100
 [19] 91-- 94 91-- 94 90-- 97 90-- 97 90-- 93 90-- 93 89-- 98 89-- 98 89-- 97 89-- 97 89-- 96 89-- 96 87-- 94 87-- 94 86-- 99 86-- 99 86-- 89 86-- 89
 [37] 85-- 96 85-- 96 85-- 86 85-- 86 84-- 94 84-- 94 84-- 85 84-- 85 83--100 83--100 83-- 98 83-- 98 83-- 96 83-- 96 82-- 97 82-- 97 82-- 92 82-- 92
 [55] 82-- 90 82-- 90 81-- 96 81-- 96 81-- 86 81-- 86 80-- 98 80-- 98 80-- 93 80-- 93 79--100 79--100 79-- 86 79-- 86 78-- 99 78-- 99 78-- 94 78-- 94
 [73] 78-- 93 78-- 93 78-- 89 78-- 89 78-- 85 78-- 85 78-- 82 78-- 82 78-- 81 78-- 81 77-- 97 77-- 97 77-- 88 77-- 88 76-- 99 76-- 99 76-- 96 76-- 96
 [91] 76-- 92 76-- 92 76-- 83 76-- 83 76-- 77 76-- 77 75-- 96 75-- 96 74-- 99 74-- 99 74-- 91 74-- 91 74-- 80 74-- 80 74-- 76 74-- 76 73--100 73--100
[109] 73-- 97 73-- 97 73-- 96 73-- 96 73-- 92 73-- 92 73-- 88 73-- 88 72-- 99 72-- 99 72-- 92 72-- 92 72-- 90 72-- 90 72-- 83 72-- 83 72-- 76 72-- 76
[127] 72-- 74 72-- 74 71-- 96 71-- 96 71-- 89 71-- 89 71-- 85 71-- 85 71-- 84 71-- 84 71-- 72 71-- 72 70-- 97 70-- 97 70-- 95 70-- 95 69--100 69--100
[145] 69-- 94 69-- 94 69-- 81 69-- 81 69-- 79 69-- 79 69-- 74 69-- 74 69-- 70 69-- 70 68-- 88 68-- 88 67-- 95 67-- 95 67-- 91 67-- 91 67-- 90 67-- 90
[163] 67-- 89 67-- 89 67-- 87 67-- 87 67-- 82 67-- 82 67-- 80 67-- 80 67-- 78 67-- 78 67-- 74 67-- 74 66-- 99 66-- 99 66-- 98 66-- 98 66-- 95 66-- 95
+ ... omitted several edges
> x |
```

I) adjacency matrix:

Get the adjacency matrix: For this we will use the as_adjacency_matrix method

agraph3.adj = igraph:: as_adjacency_matrix(agraph3)

Agraph3.adj

```
> # Get the adjacency matrix: For this we will use the as_adjacency_matrix method
> agraph3.adj = igraph:: as_adjacency_matrix(agraph3)
> agraph3.adj
100 x 100 sparse Matrix of class "dgCMatrix"

[1,] . . . . . 2 . . . . 2 2 . . . 2 . . . . 2 . . 2 2 . . . . . . . . . . . . . . 2 . . . . 2 . . 2 2 . . . . . 2 . . . . . 2 . . . . . . . . . ......
[2,] . . 2 2 . . . . . . . . . . . . . . . . 2 . . 2 2 . . . . . 2 . . . . . . . . . . . . 2 . 2 . . . . . . . . . . . . . . 2 . . 2 2 . . . 2 . ......
[3,] . 2 . . . . . . . . 2 . . . . . . . . . . . . . . . . . . . 2 . . . 2 . 2 . . . . . . . . . . . . . . . . . . . . . . . . 2 . . 2 2 . . . . ......
[4,] . 2 . . . 2 . . . . . . . . . . 2 . . 2 . . . . . . . . . . 2 . . . . . 2 . . . 2 2 . . 2 . . . . . . . . . . . . . . . . . 2 2 . 2 . . . . ......
[5,] . . . . . . . . 2 . . . . 2 . . . . 2 . . . 2 . . . . . . 2 2 . . . . . . . . . . . . . . . . . . . . 2 . . 2 . . . . . . . . 2 . 2 . . . . ......
[6,] 2 . . 2 . . . . 2 . . . . . . . . . 2 . . . . . . 2 . . . . . . . . . . . . . . 2 . . . . . . 2 . 2 2 . 2 2 . . 2 . . . . . . . . . . . . . ......
[7,] . . . . . . . . . . . . 2 . . 2 . 2 . . . . . 2 . . . . . 2 2 . . . . . 2 . . . . . . . . . . . . . . . . . . . 2 . . . . . . . . . . 2 . . ......

 .............................
 ........suppressing 30 columns and 86 rows in show(); maybe adjust 'options(max.print= *, width = *)'
 .............................

 [94,] 2 . . . . . . . . . . 2 2 . . . . . . . 2 . . . 2 . . . . . . . 2 . . . . . . . . . . . . . . . 2 2 . . . . . . . . 2 . . . . . . . . . 2 . ......
 [95,] 2 2 2 . 2 . 2 . . . . . . . 2 . . . . . 2 2 . 2 . . . . . . . . . . . 2 . . . . 2 . 2 . . . . . . . . . . . . . . . . . . . . . . . 2 2 . . 2 ......
 [96,] . . 2 . . . . . . . . 2 . . . . 2 . . . . . 2 2 . . . . . . . 2 . 2 . 2 . . . . . 2 . . . . . . . 2 2 . . 2 . . . . . . . . . . . . . . . . ......
 [97,] . . . . . 2 . . 2 . . . . . 2 . . . . . . . . . . . 2 . . . . . 2 2 . . 2 . 2 . . 2 . 2 . . . . . . . 2 2 . . . . . . 2 . . . 2 . . . . . 2 ......
 [98,] . 2 . 2 . . . 2 . . . . . . . . . . . . . . 2 . . . . . . . . 2 . 2 . . . . . 2 . . 2 . . . . . . . . . . 2 . . . . 2 2 . . . 2 . 2 . . . . ......
 [99,] . . . . . . . . . . . . . . . . . . . . 2 . . 2 . . . . . . . . 2 . 2 . . . 2 . . 2 . . 2 2 2 . . . 2 . . . . . . . 2 . . . . . . 2 . . . . ......
[100,] . . . . . . . . . . . . . . . . . 2 . . . . . . . . . . . 2 . . . . 2 . . . . 2 . . . . . . . . 2 . . . . . . . . . . . . . . . . . . . 2 . ......
> x
```

J) Density:

-> *Density* = # existing edges/ # possible edges.

The possible edges is the number of edges computed by assuming every node is connected to every other node in the graph.

gden takes an adjacency matrix as its argument.

agraph.density = gden(agraph)

agraph.density

```
>
> # Density = # existing edges/ # possible edges.
> #The # possible edges is the number of edges computed by assuming every node is connected to every other node in the graph.
> #gden takes an adjacency matrix as its argument.
> agraph.density = gden(agraph)
> agraph.density
[1] 0.4941414
>
```

Density is about 0.49 as we specified when we created the graph, but a little less since we allowed random selection of nodes.

-> A graph's density is the ratio of the number of edges and possible edges.
-> Loops specifies whether to allow loops in the graph. The default is F, but loops=T yields a smaller value because there are different paths with loops

igraph:: edge_density(agraph3)
igraph::edge_density(agraph3, loops = T)

```
> # Loops specifies whether to allow loops in the graph.
> # The default is F, but loops=T yields a smalle value because there are different paths with loops
> igraph:: edge_density(agraph3)
[1] 0.3006061
> igraph::edge_density(agraph3, loops = T)
[1] 0.2946535
> |
```

K) Egocentric network

An *egocentric network* of a vertex v is a subgraph consisting of v and its immediate neighbors. Vertices with lots of neighbors can serve in many roles, such as brokers of information passing through the network.

agraph3.ego[1] to access the ego network centered around the first node

agraph3.ego = ego.extract(agraph)

agraph3.ego[1]

```
>
> # An egocentric network of a vertex v is a subgraph consisting of v and its immediate neighbors.
> # Vertices with lots of neighbors can serve in many roles, such as brokers of information passing through the network.
> agraph3.ego = ego.extract(agraph)
> agraph3.ego[1]
$`1`
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
 [1,]    0    0    1    0    1    1    0    0    1     0     1     1     1     0     1     0     0     0     1     0     0     0     0     1     1     0
 [2,]    1    0    1    1    0    1    0    1    0     0     1     0     0     0     1     1     1     0     0     0     1     1     0     0     0     1
 [3,]    1    1    0    0    1    1    0    1    1     1     1     1     1     0     1     0     1     1     0     0     0     0     0     0     1     0
 [4,]    1    0    0    0    1    0    1    1    0     0     1     0     0     0     1     1     1     1     0     0     0     0     1     1     1     0
 [5,]    1    1    0    0    0    0    0    0    1     0     0     1     0     0     0     0     0     1     1     1     0     1     0     0     0     1
 [6,]    1    1    1    1    1    0    1    0    1     1     1     0     0     1     0     0     1     1     0     0     0     0     0     0     1     0
 [7,]    1    0    1    0    0    1    0    0    0     1     0     1     0     1     1     1     1     1     1     0     1     0     0     1     0     0
 [8,]    1    1    1    0    0    1    0    0    1     0     0     0     1     1     0     1     0     1     0     0     1     0     0     0     1     1
 [9,]    1    1    1    1    1    0    0    0    0     1     0     0     1     1     1     0     0     0     0     1     1     1     1     1     1     0
[10,]    1    0    1    0    1    0    0    1    0     0     1     0     1     0     0     1     0     1     1     1     1     0     0     1     1     1
[11,]    0    1    1    1    0    0    0    1    1     1     0     0     0     0     1     0     0     1     1     0     1     0     1     1     1     0
[12,]    1    1    0    1    0    1    0    1    1     0     1     0     1     1     0     0     0     0     1     1     1     0     0     0     1     0
      [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50]
 [1,]     0     1     1     0     1     0     1     0     1     0     0     0     1     1     0     1     0     0     1     0     0     0     0     1
 [2,]     1     0     1     0     1     0     1     1     1     0     0     0     0     1     0     1     0     1     1     1     1     0     0     1
 [3,]     1     1     0     1     0     1     0     1     1     1     1     0     0     0     0     1     1     1     0     0     0     1     1     0
 [4,]     1     1     0     1     0     0     1     1     0     1     1     1     0     0     0     1     0     0     0     1     1     0     0     0
 [5,]     1     0     1     0     1     1     0     1     1     0     1     0     0     1     1     0     1     1     0     1     0     1     0     1
 [6,]     0     1     0     1     1     1     1     0     0     1     0     1     0     0     0     1     1     0     1     0     0     1     1     0
 [7,]     0     0     1     1     0     1     1     0     0     0     1     0     1     0     1     1     0     0     1     1     0     1     1     1
 [8,]     0     1     1     0     0     0     1     0     0     0     1     1     0     1     1     0     0     0     1     1     0     1     0     0
 [9,]     1     1     0     1     0     1     1     1     0     0     0     0     0     0     1     0     0     1     0     0     0     1     0     1
[10,]     1     1     1     1     0     0     0     0     1     0     0     1     1     1     0     0     0     1     0     1     1     1     1     0
[11,]     0     0     1     0     1     0     0     0     0     1     0     0     1     0     1     1     1     0     1     0     1     0     0     0
[12,]     0     0     0     1     1     1     1     0     0     1     0     0     1     0     1     0     1     1     1     0     1     0     0     0
      [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74]
 [1,]     0     1     1     1     1     0     1     1     1     0     0     1     1     0     1     1     1     1     1     0     1     1     0     1
 [2,]     0     1     1     1     0     1     0     1     1     1     0     1     0     0     0     0     1     1     0     0     1     1     0     1
 [3,]     1     0     1     1     0     0     0     1     1     0     0     1     0     1     1     1     0     1     0     1     0     1     0     0
 [4,]     1     0     0     0     1     1     1     0     0     0     0     1     0     0     0     0     1     1     1     1     1     1     0     0
 [5,]     0     0     0     1     1     1     1     0     0     0     0     1     0     0     0     1     0     1     1     1     1     1     0     0
 [6,]     0     1     0     0     1     0     1     1     0     0     0     1     0     1     1     0     1     0     1     1     1     0     0     1
 [7,]     0     0     1     0     1     0     1     0     1     1     1     0     0     1     0     1     1     1     0     0     1     0     1     0
 [8,]     0     1     0     0     1     1     0     1     0     0     1     0     1     0     1     1     1     1     0     1     1     1     1     1
 [9,]     1     0     1     0     0     0     1     0     1     0     1     1     1     0     0     1     1     1     1     0     0     1     1     1
[10,]     1     1     1     1     0     1     0     0     0     1     0     1     0     0     1     1     0     0     1     0     1     0     1     1
[11,]     1     1     1     0     0     1     0     1     1     0     0     0     0     0     1     0     1     1     1     1     0     0     1     1
[12,]     0     1     0     1     1     0     0     0     1     1     1     0     0     1     0     1     0     1     0     1     0     1     0     0
      [,75] [,76] [,77] [,78] [,79]
 [1,]     1     0     0     1     1
 [2,]     1     0     0     0     1
 [3,]     0     0     0     0     0
 [4,]     1     0     0     1     1
 [5,]     0     1     1     1     1
 [6,]     0     1     0     0     1
 [7,]     1     0     0     0     1
 [8,]     1     1     1     1     1
 [9,]     0     0     1     0     1
[10,]     0     1     1     1     1
[11,]     1     0     1     1     1
[12,]     0     0     0     0     0
 [ reached getOption("max.print") -- omitted 67 rows ]

>
```

L) degree of each node:

We can find the degree of each node in the graph using th degree function from igraph package igraph:: degree(agraph3)

```
> # We can find the degree of each node in the graph:
> # the function 'degree' might be defined in several packages.
> # So, sometimes you need to preface the function name with the package name.
> # Here "igraph" is the package name, so the function call is "igraph::degree"
> igraph:: degree(agraph3)
 [1] 38 30 22 34 24 32 22 32 28 22 30 28 34 44 36 32 18 34 22 18 30 16 12 48 36 30 32 36 20 20 20 46 24 30 36 32 40 22 34 34 44 30 32 34 22 34 30 34 32 34
[51] 30 20 32 26 22 22 24 34 30 28 28 26 30 36 36 40 40 14 32 32 30 34 26 30 22 32 28 30 26 28 24 20 36 28 38 22 22 26 32 32 28 30 18 32 34 40 44 36 36 26
>
```

M) Some centrality metrics:

betweenness centrality: is a measure of centrality in a graph based on shortest paths. For every pair of vertices in a connected graph, there exists at least one shortest path between the vertices such that either the number of edges that the path passes through (for unweighted graphs) or the sum of the weights of the edges (for weighted graphs) is minimized. The betweenness centrality for each vertex is the number of these shortest paths that pass through the vertex.

agraph3.between = igraph::centr_betw(agraph3)
agraph3.between

```
> # betweenness centrality: is a measure of centrality in a graph based on shortest paths.
> # For every pair of vertices in a connected graph, there exists at least one shortest path between the vertices such that either the number of
> # edges that the path passes through (for unweighted graphs) or the sum of the weights of the edges (for weighted graphs) is minimized.
> #The betweenness centrality for each vertex is the number of these shortest paths that pass through the vertex.
> agraph3.between = igraph::centr_betw(agraph3)
> agraph3.between
$res
  [1]  65.395375  41.724105  19.554661  66.834725  43.315294  49.037725  32.666654  47.721945  33.922167  22.415924  42.448517  37.576519  61.673629
 [14] 105.718873  62.068953  52.705891  11.786084  65.599414  25.775573  16.500366  45.083674   7.878361   6.423030 109.081116  58.852795  42.646660
 [27]  49.733795  66.534240  15.677451  25.227745  21.498838  99.715613  27.345403  44.978141  68.564776  49.162741  81.112112  21.170277  47.447243
 [40]  51.002441  87.313063  34.542024  48.994060  62.802360  16.623075  76.677291  48.490532  50.644662  61.456526  54.323183  42.102443  21.655392
 [53]  53.709942  30.170643  24.731863  20.216799  32.236732  54.186144  44.084601  33.658648  39.169273  32.869282  38.173830  61.219116  73.689251
 [66]  76.059273  83.887997  10.072960  66.240241  59.734637  44.877491  54.941638  29.723038  45.199187  25.838060  46.979590  35.522723  40.169420
 [79]  35.383130  34.084930  24.464717  15.529474  69.023886  39.258489  73.908183  18.852077  20.415625  42.921606  56.580362  48.645052  35.903265
 [92]  42.679248  12.723340  50.375194  59.474183  93.769135 102.829182  69.763109  63.580756  31.269224

$centralization
[1] 0.01303514

$theoretical_max
[1] 480249

>
```

Closeness Centrality (CLC): is a measure defined for a given vertex. In a connected graph, **closeness centrality** (or **closeness**) of a node is a measure of centrality in a network, calculated as the reciprocal of the sum of the length of the shortest paths between the node and all other nodes in the graph. Thus, the more central a node is, the *closer* it is to all other nodes.

agraph3.closeness = igraph::centr_clo(agraph3)

agraph3.closeness

```
> agraph3.closeness = igraph::centr_clo(agraph3)
> agraph3.closeness
$res
  [1] 0.5351351 0.5238095 0.4876847 0.5294118 0.4852941 0.5265957 0.4900990 0.5238095 0.5103093 0.4805825 0.5156250 0.5183246 0.5294118 0.5625000 0.5322581
 [16] 0.5322581 0.4852941 0.5294118 0.4852941 0.4647887 0.5183246 0.4626168 0.4361233 0.5593220 0.5380435 0.5183246 0.5294118 0.5351351 0.4829268 0.4829268
 [31] 0.4782609 0.5593220 0.4974874 0.5238095 0.5439560 0.5265957 0.5439560 0.4925373 0.5351351 0.5351351 0.5593220 0.5265957 0.5238095 0.5322581 0.5051020
 [46] 0.5265957 0.5076923 0.5380435 0.5265957 0.5265957 0.5210526 0.4876847 0.5265957 0.5103093 0.5025381 0.4974874 0.5025381 0.5238095 0.5076923 0.5129534
 [61] 0.5210526 0.5051020 0.5156250 0.5469613 0.5294118 0.5530726 0.5409836 0.4439462 0.5210526 0.5294118 0.5210526 0.5156250 0.5210526 0.5322581 0.5000000
 [76] 0.5238095 0.5000000 0.5129534 0.5156250 0.5129534 0.4974874 0.4950000 0.5294118 0.5183246 0.5409836 0.5076923 0.4974874 0.5025381 0.5322581 0.5351351
 [91] 0.5210526 0.5103093 0.4829268 0.5156250 0.5294118 0.5500000 0.5593220 0.5322581 0.5351351 0.5183246

$centralization
[1] 0.09268964

$theoretical_max
[1] 49.24873

>
```

N) Shortest Path:

Shortest paths provides the length of the shortest path between any two nodes in a graph

From igraph package we will use the distances function which takes input a graph object and gives a result of shortest path agraph3.sp = igraph::distances(agraph3)

agraph3.sp

```
> agraph3.sp = igraph::distances(agraph3)
> agraph3.sp
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
 [1,]    0    2    2    2    2    1    2    2    2     2     1     1     2     2     2     1     2     2     2     3     1     2     3     1     1     2
 [2,]    2    0    1    1    2    2    2    2    2     3     2     2     2     2     2     2     2     2     2     2     1     2     2     1     1     2
 [3,]    2    1    0    2    2    3    2    3    3     3     1     2     2     2     2     2     2     2     2     3     2     2     2     2     2     2
 [4,]    2    1    2    0    2    1    2    2    2     2     3     3     2     2     2     2     1     2     2     1     2     2     2     2     2     2
 [5,]    2    2    2    2    0    2    2    2    1     3     3     2     2     1     2     2     3     2     1     2     2     2     1     2     2     2
 [6,]    1    2    3    1    2    0    2    2    1     2     2     2     2     2     2     2     2     2     1     2     2     3     3     2     2     1
 [7,]    2    2    2    2    2    2    0    2    2     2     2     2     1     2     2     1     3     1     2     3     2     2     3     1     2     2
 [8,]    2    2    3    2    2    2    2    0    2     2     2     2     2     2     2     2     3     1     2     2     2     2     2     1     2     2
 [9,]    2    2    3    2    1    1    2    2    0     2     2     2     1     2     2     2     2     1     2     3     2     2     2     3     2     2
[10,]    2    3    3    2    3    2    2    2    2     0     2     2     2     2     2     2     2     1     2     3     2     2     3     3     3     2
      [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50]
 [1,]     2     2     2     2     2     2     2     2     2     2     2     2     2     1     2     3     2     2     1     2     2     1     1     2
 [2,]     2     2     2     2     1     2     2     2     2     2     2     2     2     2     2     2     2     1     2     1     2     2     2     2
 [3,]     3     2     3     3     1     2     3     2     1     2     1     2     2     2     2     2     2     2     2     2     2     2     2     2
 [4,]     2     2     2     3     1     2     2     2     2     2     1     2     2     2     1     1     2     2     1     2     2     2     2     2
 [5,]     2     2     2     1     1     2     3     2     2     2     2     2     3     2     2     2     2     2     3     2     2     2     2     2
 [6,]     2     2     2     2     2     2     2     2     2     2     2     2     2     2     1     2     2     2     2     2     3     1     2     1
 [7,]     2     2     2     1     1     2     2     2     2     2     1     2     2     2     2     2     2     2     2     2     2     2     2     2
 [8,]     1     2     3     2     2     2     3     2     3     2     2     2     2     2     2     2     1     2     1     2     2     2     2     2
 [9,]     2     1     2     2     2     2     2     1     2     2     2     2     2     2     2     2     1     2     2     2     3     1     2     1
[10,]     2     2     3     2     2     2     2     3     2     2     2     2     3     2     2     2     3     2     2     2     2     2     2     2
      [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74]
 [1,]     2     2     2     2     1     2     2     2     2     2     1     2     2     2     2     2     2     3     2     2     2     3     2     2
 [2,]     2     2     2     2     3     3     2     3     2     2     1     2     2     1     1     2     2     2     1     2     2     3     2     2
 [3,]     2     2     2     2     2     3     2     2     3     3     2     1     3     2     1     1     2     3     2     2     2     2     2     2
 [4,]     2     2     2     2     2     2     2     2     2     2     2     2     1     1     2     1     2     2     2     2     1     2     2     2
 [5,]     3     1     2     3     1     2     3     2     2     2     2     2     3     1     2     1     2     2     2     2     3     2     2     3
 [6,]     1     2     1     1     2     2     1     2     2     2     2     2     2     2     2     2     2     2     2     2     2     2     2     2
 [7,]     3     3     3     3     2     2     1     2     3     2     3     2     2     2     2     2     2     1     2     2     2     2     3     2
 [8,]     2     3     1     2     2     1     2     1     1     2     2     2     2     1     2     2     2     3     2     1     2     1     2     2
 [9,]     2     2     1     2     2     2     2     2     2     2     1     2     3     2     2     2     2     2     2     1     2     3     2     2
[10,]     2     3     1     2     3     2     2     2     3     2     2     2     2     2     2     2     1     3     2     1     1     1     1     2
      [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90] [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98]
 [1,]     3     2     2     2     2     1     2     2     2     2     2     1     1     2     1     2     2     2     2     1     1     2     2     2
 [2,]     2     2     2     2     2     2     2     2     2     3     2     2     2     1     2     2     2     2     2     2     1     2     2     1
 [3,]     2     2     2     2     2     2     2     2     2     2     1     2     2     2     2     3     2     2     2     3     1     1     2     2
 [4,]     2     1     2     3     1     2     2     3     2     2     2     2     2     1     2     2     2     2     2     3     2     2     2     1
 [5,]     2     2     2     3     2     3     2     3     2     2     2     1     3     3     2     2     2     2     2     3     1     2     2     2
 [6,]     2     2     1     2     2     1     3     2     2     2     2     2     2     2     2     2     1     3     2     2     2     2     1     2
 [7,]     2     2     2     3     3     2     2     2     2     1     2     2     2     2     3     2     3     3     2     2     1     2     2     2
 [8,]     2     2     2     2     1     2     2     2     2     2     2     2     2     2     2     1     2     1     2     2     2     2     2     1
 [9,]     2     2     2     2     3     2     2     2     3     1     2     2     2     2     2     2     2     2     3     2     2     2     1     2
[10,]     3     2     3     2     1     2     2     2     2     1     2     2     1     2     2     2     2     1     3     2     2     2     2     3
      [,99] [,100]
 [1,]     2      2
 [2,]     2      2
 [3,]     2      2
 [4,]     2      2
 [5,]     2      2
 [6,]     2      2
 [7,]     2      2
 [8,]     2      2
 [9,]     3      2
[10,]     2      2
 [ reached getOption("max.print") -- omitted 90 rows ]
>
```

shortest_paths(): The shortest_paths() function from the igraph package calculates the shortest paths in a graph. Here we are finding the shortest path from node 5 to all other nodes in the graph.

igraph:: shortest_paths(agraph3, from = 5)

```
Error in igraph::shortest_paths(agraph3) :
  argument "from" is missing, with no default
> igraph:: shortest_paths(agraph3, from = 5)
$vpath
$vpath[[1]]
+ 3/100 vertices, from 9310e34:
[1]  5 55  1

$vpath[[2]]
+ 3/100 vertices, from 9310e34:
[1]  5 31  2

$vpath[[3]]
+ 3/100 vertices, from 9310e34:
[1]  5 31  3

$vpath[[4]]
+ 3/100 vertices, from 9310e34:
[1]  5 31  4

$vpath[[5]]
+ 1/100 vertex, from 9310e34:
[1] 5

$vpath[[6]]
+ 3/100 vertices, from 9310e34:
[1] 5 9 6

$vpath[[7]]
+ 3/100 vertices, from 9310e34:
[1]  5 30  7

$vpath[[8]]
+ 3/100 vertices, from 9310e34:
[1]  5 64  8

$vpath[[9]]
+ 2/100 vertices, from 9310e34:
[1] 5 9

$vpath[[10]]
+ 4/100 vertices, from 9310e34:
[1]  5  9 18 10

$vpath[[11]]
+ 4/100 vertices, from 9310e34:
[1]  5  9 18 11

$vpath[[12]]
+ 3/100 vertices, from 9310e34:
[1]  5 30 12

$vpath[[13]]
+ 3/100 vertices, from 9310e34:
[1]  5  9 13

$vpath[[14]]
```

Geodesic: A geodesic is the shortest path between any two nodes in the network.

A node has high betweenness if the geodesics between many pairs of other nodes pass through that node. A node with high betweenness, when it fails or is removed, has greater influence on the connectivity of the network.

agraph.geos = geodist(agraph)

agraph.geos

```
> agraph.geos = geodist(agraph)
> agraph.geos
$counts
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
 [1,]    1   13   24   30    1   19    1    1   21    15     1    20     1     1     1    17     1    22    17    20     1    24    21    21    19    15
 [2,]   26    1   24   26   19   25    1    1    1    15    19     1     1    27    24     1     1    29     1    21     1    24    22     1    21    17
 [3,]   26    1    1   23    1    1   21   19   21    18    21     1    21     1    25    21    22    27     1    22     1     1    22     1    19    20
 [4,]    1   20   27    1    1    1   19    1   23     1    23    23     1    17    24    20     1     1     1    21    22    19     1     1    17    18
 [5,]    1    1   32    1    1   26    1    1   30     1     1     1     1     1     1    22     1    25     1     1    30    25    23    25    20    20
 [6,]    1    1    1   24   28    1    1   24    1     1    21    23     1    25    24    25     1     1     1     1    31    24    27    24     1    23
 [7,]    1    1   26    1   24   22    1   19   24    15     1    25    22     1    26    20    18    21    21     1     1     1    19     1    20    17
 [8,]    1   24   29    1    1    1    1    1    1    23     1     1     1    28    25     1    26    29     1     1    32    29    26    26    21     1
 [9,]    1   17   32   35    1   26   25    1    1    19    23     1    33     1    27     1     1     1     1     1     1    33     1    22    24    25
[10,]    1   24    1    1    1   23   23    1   24     1     1    27    25    22     1     1    22     1    26     1    26    28     1    27    21     1
      [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50]
 [1,]     1    24    22     1    20    19    23     1     1    26     1    22     1    16     1    20    24    17    11     1     1    19    19    21
 [2,]    26    24    23     1    22    22    23     1     1     1     1    24     1     1    24     1     1    16     1    21    20     1    19     1
 [3,]     1     1    24    22     1     1     1     1    28     1    18     1     1    22    24     1     1     1    22    22     1     1     1     1
 [4,]    25    19    16    21    20     1     1    21     1    25     1    16     1     1     1    21     1    17    19    18     1    21    22    21
 [5,]    26    25    26     1     1    26     1     1    31     1    26     1    30     1     1     1     1     1    22    20    24     1    23     1
 [6,]     1     1    22     1    24    25     1     1    32     1    24    21     1     1    25     1     1     1     1    20    26    26    25     1
 [7,]    24    21     1    20     1     1     1    21     1    26     1     1    31     1     1    19     1     1    11    22     1    25     1    20
 [8,]    30    29    25     1    26    24    28     1    38     1     1     1     1    27    26     1     1    26     1    23    23    25    23     1
 [9,]     1     1     1    27     1    25    23    29     1     1    22     1     1    23    25    25     1     1    19     1    20     1     1    26
[10,]    26    23    23     1     1     1    27     1     1    27    20    27     1    29    28    26    27     1     1    20     1    25     1     1
      [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74]
 [1,]     1    20    18     1    25    18    19    16    19    22    22    14     1    21    25    19    15     1     1     1     1    20     1     1
 [2,]     1     1    19     1     1     1     1     1    19    27     1    17    17     1     1    26     1     1    22     1    21     1    21    26
 [3,]    23    20     1    24    28     1     1    14     1     1    22     1    13    19     1     1     1    26    23     1     1     1     1    26
 [4,]     1    20     1     1     1    23    19     1     1    19    22    16     1    23    18    22    17     1     1     1    25     1    21     1
 [5,]     1     1     1    30     1    25    26    18    26     1     1     1    22     1     1     1     1    32     1     1    27    30    19     1
 [6,]     1    20    21    32     1     1    28     1     1    25    18     1    18     1    23     1     1    26    26    21     1     1     1    28
 [7,]    23     1     1    30     1    20    24     1    21     1    22     9     1    16    23    18    15    22    19     1     1     1     1    23
 [8,]     1     1    22     1     1     1     1    20    27     1     1     1    19    24     1     1    25     1    30    26     1    31     1     1
 [9,]     1    26    24     1     1     1    28     1    27     1     1    16     1    23     1     1    23    22     1    29     1    28     1    33
[10,]    29    22    23     1     1    21     1     1    23     1    29    16    23    23    23    27    20     1    27    24     1     1    23     1
      [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90] [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98]
 [1,]     1    18    26    18     1     1    20     1     1     1    26    18     1     1    23     1     1    17     1     1    22    26    19    20
 [2,]    23     1     1    21     1    29    22    24    23    26     1     1     1    24    23    22    23     1    23     1     1    28     1    26
 [3,]    21    26    25    20    26    32    25    27     1     1     1     1    28     1    25    22    27    23    23     1    27    22     1     1
 [4,]     1    23     1    17     1    23    19    28    22     1     1    20     1    20    24     1     1    18     1     1    23    25    20    20
 [5,]     1     1    32    22     1    27     1     1     1    32     1    20     1    29     1    27     1    30    30    27     1    29    23    26
 [6,]    27    27    25    21     1    30    24    27    29     1     1     1     1     1     1     1     1    23    26     1    25    26    28    26
 [7,]    24    23    23    21     1    24    20    26     1    23    23    20     1     1     1     1     1    22    22    20     1     1     1    20
 [8,]    29     1    26    24     1    32     1     1    28     1    30     1    31     1     1     1    29    26     1    22     1     1    24     1
 [9,]     1     1     1     1    26    28     1    31     1     1    30     1     1    23    26     1    31     1    27     1     1    31    22    28
[10,]    28     1    21     1    32     1    23     1     1     1    33    22     1    23     1     1     1     1     1     1     1     1     1     1
      [,99] [,100]
 [1,]     1      1
 [2,]    25     23
 [3,]    22     21
 [4,]    18      1
 [5,]    27     26
 [6,]     1      1
 [7,]     1      1
 [8,]    26      1
 [9,]    25      1
[10,]     1      1
 [ reached getOption("max.print") -- omitted 90 rows ]
```

```
$gdist
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
 [1,]    0    2    2    2    1    2    1    1    2     2     1     2     1     1     1     2     1     2     2     2     1     2     2     2     2     2
 [2,]    2    0    2    2    2    2    1    1    1     2     2     1     1     2     2     1     1     2     1     2     1     2     2     1     2     2
 [3,]    2    1    0    2    1    1    2    2    2     2     2     1     2     1     2     2     2     2     1     2     1     1     2     1     2     2
 [4,]    1    2    2    0    1    1    2    1    2     1     2     2     1     2     2     2     1     1     1     2     2     2     1     1     2     2
 [5,]    1    1    2    1    0    2    1    1    2     1     1     1     1     1     1     2     1     2     1     1     2     2     2     2     2     2
 [6,]    1    1    1    2    2    0    1    2    1     1     2     2     1     2     2     2     1     1     1     1     2     2     2     2     1     2
 [7,]    1    1    2    1    2    2    0    2    2     2     1     2     2     1     2     2     2     2     2     1     1     1     2     1     2     2
 [8,]    1    2    2    1    1    1    1    0    1     2     1     1     1     2     2     1     2     2     1     1     2     2     2     2     2     1
 [9,]    1    2    2    2    1    2    2    1    0     2     2     1     2     1     2     1     1     1     1     1     1     2     1     2     2     2
[10,]    1    2    1    1    1    2    2    1    2     0     1     2     2     2     1     1     2     1     2     1     2     2     1     2     2     1
      [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50]
 [1,]     1     2     2     1     2     2     2     1     1     2     1     2     1     2     1     2     2     2     2     1     1     2     2     2
 [2,]     2     2     2     1     2     2     2     1     1     1     1     2     1     1     2     1     1     2     1     2     2     1     2     1
 [3,]     1     1     2     2     1     1     1     1     2     1     2     1     1     2     2     1     1     1     2     2     1     1     1     1
 [4,]     2     2     2     2     2     1     1     2     1     2     1     2     1     1     1     2     1     2     2     2     1     2     2     2
 [5,]     2     2     2     1     1     2     1     1     2     1     2     1     2     1     1     1     1     1     2     2     2     1     2     1
 [6,]     1     1     2     1     2     2     1     1     2     1     2     2     1     1     2     1     1     1     1     2     2     2     2     1
 [7,]     2     2     1     2     1     1     1     2     1     2     1     1     2     1     1     2     1     1     2     2     1     2     1     2
 [8,]     2     2     2     1     2     2     2     1     2     1     1     1     1     2     2     1     1     2     1     2     2     2     2     1
 [9,]     1     1     1     2     1     2     2     2     1     1     2     1     1     2     2     2     1     1     2     1     2     1     1     2
[10,]     2     2     2     1     1     1     2     1     1     2     2     2     1     2     2     2     2     1     1     2     1     2     1     1
      [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74]
 [1,]     1     2     2     1     2     2     2     2     2     2     2     2     1     2     2     2     2     1     1     1     1     2     1     1
 [2,]     1     1     2     1     1     1     1     1     2     2     1     2     2     1     1     2     1     1     2     1     2     1     2     2
 [3,]     2     2     1     2     2     1     1     2     1     1     2     1     2     2     1     1     1     2     2     1     1     1     1     2
 [4,]     1     2     1     1     1     2     2     1     1     2     2     2     1     2     2     2     2     1     1     1     2     1     2     1
 [5,]     1     1     1     2     1     2     2     2     2     1     1     1     2     1     1     1     1     2     1     1     2     2     2     1
 [6,]     1     2     2     2     1     1     2     1     1     2     2     1     2     1     2     1     1     2     2     2     1     1     1     2
 [7,]     2     1     1     2     1     2     2     1     2     1     2     2     1     2     2     2     2     2     2     1     1     1     1     2
 [8,]     1     1     2     1     1     1     1     2     2     1     1     1     2     2     1     1     2     1     2     2     1     2     1     1
 [9,]     1     2     2     1     1     1     2     1     2     1     1     2     1     2     1     1     2     2     1     2     1     2     1     2
[10,]     2     2     2     1     1     2     1     1     2     1     2     2     2     2     2     2     2     1     2     2     1     1     2     1
      [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90] [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98]
 [1,]     1     2     2     2     1     1     2     1     1     1     2     2     1     1     2     1     1     2     1     1     2     2     2     2
 [2,]     2     1     1     2     1     2     2     2     2     2     1     1     1     2     2     2     2     1     2     1     1     2     1     2
 [3,]     2     2     2     2     2     2     2     2     1     1     1     1     2     1     2     2     2     2     2     1     2     2     1     1
 [4,]     1     2     1     2     1     2     2     2     2     1     1     2     1     2     2     1     1     2     1     1     2     2     2     2
 [5,]     1     1     2     2     1     2     1     1     1     2     1     2     1     2     1     2     1     2     2     2     1     2     2     2
 [6,]     2     2     2     2     1     2     2     2     2     1     1     1     1     1     1     1     1     2     2     1     2     2     2     2
 [7,]     2     2     2     2     1     2     2     2     1     2     2     2     1     1     1     1     1     2     2     2     1     1     1     2
 [8,]     2     1     2     2     1     2     1     1     2     1     2     1     2     1     1     1     2     2     1     2     1     1     2     1
 [9,]     1     1     1     1     2     2     1     2     1     1     2     1     1     2     2     1     2     1     2     1     1     2     2     2
[10,]     2     1     2     1     2     1     2     1     1     1     2     2     1     2     1     1     1     1     1     1     1     1     1     1
      [,99] [,100]
 [1,]     1      1
 [2,]     2      2
 [3,]     2      2
 [4,]     2      1
 [5,]     2      2
 [6,]     1      1
 [7,]     1      1
 [8,]     2      1
 [9,]     2      1
[10,]     1      1
 [ reached getOption("max.print") -- omitted 90 rows ]
```

Number of paths between two nodes: We can multiply the adjacency matrix by itself. The cell numbers specify the number of paths.

agraph3.np = agraph3.adj%*%agraph3.adj

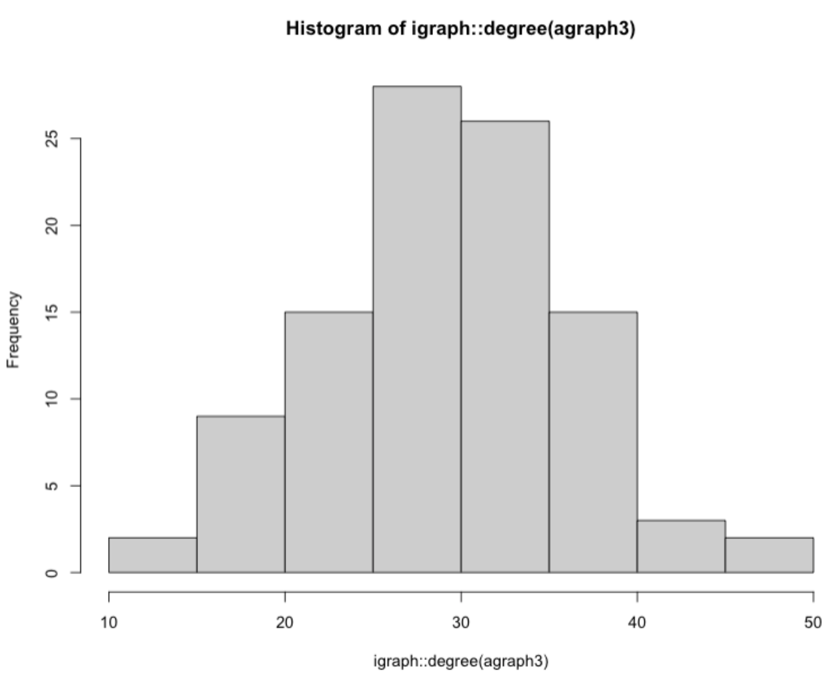
agraph3.np

```
$predecessors
NULL

$inbound_edges
NULL

> # Suppose we want to find the number of paths between two nodes.
> # We can multiply the adjacency matrix by itself. The cell numbers specify the number of paths.
> agraph3.np = agraph3.adj%*%agraph3.adj
> agraph3.np
100 x 100 sparse Matrix of class "dgCMatrix"

[1,] 76 20  8  8 12  8 12  8 12 4 12 12 20  8 12  8  4  8  8  .  8 12 . 20  8 12 12 12 4 12  4 20  8  4 28 28 12  4  4  8 20  .  8  8  8  8 8 16 12 20 8  4 32 12  .
[2,] 20 60 12 16 12  4 12 12  4 . 12  8 16  8  4  4 16  8  4 12  8 12 8 16 12  8  4  8 4  4 12  8  4  4 16 20 20  4  8  8  8  4 20  4 12  4 4  8  8  4 4  4  8  8  .
[3,]  8 12 44 16 12  . 12  .  . .  4 12  4 12 12  8  8  8  4  .  8  8 4 24 16 12  .  4 .  . 12 12  .  4  . 12  4  4  8  8 16 12  8 16 12 20 8 12  8  8 4  4  4  4  4
[4,]  8 16 16 68 12  4  8 16  4 8  .  .  8 12 12 16  4 12 16  4  8  4 4 16 12 16  8  4 8  .  4 16  4  4  8 12 12  8 12 12 28  8 16 20  4 24 8 16 16 16 4  4 12 16  8
[5,] 12 12 12 12 48  8 12  4  . .  .  4 16  4 16  4  .  8  .  8  8  4 . 12  4  4  4  4 8  .  .  4  . 12  8  8  4  4  .  4 20  8 20  4  .  4 8 16  4 16 .  .  4  .  4
[6,]  8  4  .  4  8 64  4  4 16 4 12  8  8 12  8 16  4  8  4  8 12  .  .  8 16 12 12 12 8  8  4 12 16 20 12  8 12 12 12 12 16 20 12  4 12  8 . 16 16  8 4 12 12  4 12
[7,] 12 12 12  8 12  4 44  8 12 8  8  8  4  8  8 12  .  4  4  .  8 16 .  8 12  4 12  8 4  .  . 20  4  4  4  8  8  4  4  4 12  8 12  4  4 12 8  8  4  4 .  .  .  .  4

[1,] 8 12  8 16  4  8 12  8 20 20 12 32 . 12 12 ......
[2,] .  8  . 12 16  8 12  8 16 12 16  8 8  4 16 ......
[3,] .  4  4  .  . 12 12  .  4  8 12 12 .  8  8 ......
[4,] 4  8  4  4 16  8  8  8  8 16 16  4 4 16  4 ......
[5,] 4  .  8  8  8 12  8  .  4  8 12  8 4  4 12 ......
[6,] 4  . 12  8 16 20  4 16 24  8 20 16 4  4 16 ......
[7,] 4  4  8  .  8  .  8  8 12  4 12 16 . 16  8 ......

 ..............................
 ........suppressing 30 columns and 86 rows in show(); maybe adjust 'options(max.print= *, width = *)'
 ..............................

 [94,] 20 12  .  .  . 16 8  4 16  8  8  4  8 12  8 16  4  8  8  . 16  4 8 12 12  . 12 24  4  8  4 16  8  8 12  4 16  4 20 16  4  8  8 16 12 12  . 12 12 16  .  8 20
 [95,]  8 16  8 16  4  8 4 16 12  8 16 20 12 12  4 16  4  4  8  . 12  8 8 20 16  8  8 12  . 20 20 16  4 12  8  4 32 16 12 20  8  8 12 12  4  8  .  8 12  8 16  8  4
 [96,] 12  8  4 16  4 12 4  4  8  8 16 16  . 24 16  8  4 16  4  4  4  8 4 20 16 12 16 16 24  8  8 20  8 12 24 12 16  8 12 12 16 12  4 20  4 20  8 20 16  4 12 12  8
 [97,] 12  8  4 16 12 12 4 20 16 12 16 12 20 24 16  4  8 20  4 12 12 16 . 20 20 20 12 16 12  8  4 20 12 12  8  4  8  8 20 16 20  4 20  4  8 12 16 24 28 28 24 12 12
 [98,] 16 12 12 12 16 12 4 12  8  .  .  4  4 16 12 16  4 20  .  8 12  8 8 28 20 12 12 12  8  4  8 12  8  8 12  8 12  8 16 12 24 12 20  4 12  4  8 12 16 16  8 12 12
 [99,] 16 12  8 16  8  4 4 16  .  4  4 12  4 20 20  4 16 12 12  8  8  8 4  4 12  8  8 12 12  8  4 20  8  .  8 12 28  4 16 28 20  8 12 12  . 12  4  8  8 24  .  .  8
[100,]  8  4  4  4  4  8 8 12  8 16  8 12  4  8 16  4  4  4  4  4 20  . 8 12  8  4 12  8  4  .  4 28 12  8  8  4  8  8  8  4  8 12  8  8 16 12 12 12 12  4  4  8 12

 [94,] 12 12  .  8  4 12  8  8  8  4 12  8 12 32  8  8 12 ......
 [95,]  8 20  8  8  8  8  4 12  8 12 20 16 24  4  4 16  4 ......
 [96,]  8  8  8 12 20 16  8  8  4 20 12 24 12 16  4  8 12 ......
 [97,] 12  4 12 16 16 16 12 16  8 12  8  8 16 20  4 16  8 ......
 [98,]  8  8  8 12  8 20  8 16  8  8 24 12 16 16  . 12 12 ......
 [99,]  8  8  4  8 16  8 20  8 12 20  8 20  4 20 12 16  8 ......
[100,]  8  8  4  4  8 12  8  8 16  . 16  4 16 12  .  8  8 ......
> |
```

O) Histogram:

A Histogram of thedegree of nodes in the given graph. The degree of a node in a graph represents the number of connections that node has. Analyzing the degree distribution helps in understanding the overall structure of the network.

hist(igraph:: degree(agraph3))



P) Smaller Graph:

Making the dataset smaller to get some more visibility. By generating random graphs and converting them into adjacency matrices, you can gain insights into the structural properties of graphs. While generating random graphs and converting them into adjacency matrices may not be directly required for analyzing a specific graph dataset, it can serve as a valuable tool for understanding, testing, and experimenting with graph-related concepts and techniques.

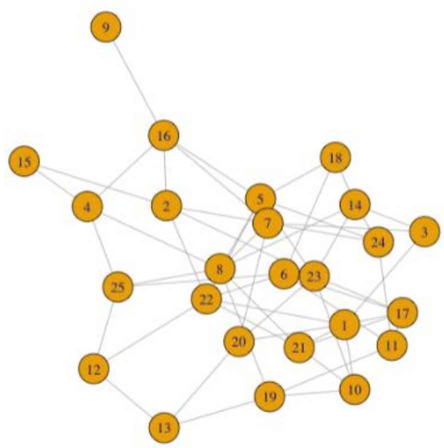
bgraph = sna:: rgraph(25,1,0.2,"graph",FALSE)

bgraph.adj <- igraph::graph_from_adjacency_matrix(bgraph, mode = "undirected")

bgraph.adj

```
> bgraph = sna:: rgraph(25,1,0.2,"graph",FALSE)
> bgraph.adj <- igraph::graph_from_adjacency_matrix(bgraph, mode = "undirected")
> bgraph.adj
IGRAPH c3e9a8a U--- 25 56 --
+ edges from c3e9a8a:
 [1]  1-- 3  1--10  1--17  1--20  1--21  1--22  2-- 7  2--15  2--16  2--22  2--23  3-- 7  3--14  4-- 8  4--15  4--16  4--25  5-- 7  5-- 8  5--16  5--18  5--22
[23]  5--23  5--24  6--11  6--17  6--18  6--22  6--25  7-- 8  7--16  7--20  7--24  8--14  8--19  8--21  8--25  9--16 10--19 10--21 10--23 11--17 11--19 11--24
[45] 12--13 12--22 12--25 13--19 13--20 14--18 14--23 14--24 17--21 17--23 20--23 21--22
> |
```

plot(bgraph.adj)



The difference between density with and without loop consideration as the link factor decreases:
Edge density is defined as the ratio of the number of edges present in the graph to the total number of possible edges. It quantifies how densely the nodes in the graph are connected to each other.

This line loops = T calculates the edge density of the graph, considering loops (self-edges) as well. By setting loops = TRUE, the function includes self-edges when calculating edge density.

Calculating edge density with and without loops lets you understand how self-edges affect the graph's overall connectivity.

igraph::edge_density(bgraph.adj)

igraph::edge_density(bgraph.adj,loops = T)

```
> igraph::edge_density(bgraph.adj)
[1] 0.1866667
> igraph::edge_density(bgraph.adj,loops = T)
[1] 0.1723077
>
```

Find the diameter of bgraph: The diameter of a graph is defined as the maximum shortest path length between any pair of vertices in the graph.

bgraph.d = igraph::diameter(bgraph.adj)

bgraph.d

```
> # Find the diameter of bgraph
> bgraph.d = igraph::diameter(bgraph.adj)
> bgraph.d
[1] 4
>
```

Find the max cliques for node 13: The max_cliques() function from the igraph package is used to find the maximal cliques in the graph. Maximal cliques are complete subgraphs (subsets of nodes) in which every pair of nodes is connected by an edge, and no additional node can be added to the subset while maintaining this property.

node <- c(13)

bgraph.13clique = igraph::max_cliques(bgraph.adj,min = NULL, max = NULL, subset = node)

Bgraph.13clique

```
> # Find the max-cliques for node 13(any random node):
> # The max_cliques() function from the igraph package is used to find the maximal cliques in the graph.
> # Maximal cliques are complete subgraphs (subsets of nodes) in which every pair of nodes is connected by an edge,
> # and no additional node can be added to the subset while maintaining this property.
> node <- c(13)
> bgraph.13clique = igraph::max_cliques(bgraph.adj,min = NULL, max = NULL, subset = node)
> bgraph.13clique
[[1]]
+ 2/25 vertices, from c3e9a8a:
[1] 14 24

[[2]]
+ 2/25 vertices, from c3e9a8a:
[1] 14 23

[[3]]
+ 2/25 vertices, from c3e9a8a:
[1] 14 18

>
```

Find the largest clique: The largest clique in a graph represents the largest subset of nodes where every node is connected to every other node within the subset. Finding the size of the largest clique provides insights into the maximum level of connectivity within the graph

bgraph.largestClique = igraph::clique_num(bgraph.adj)

Bgraph.largestClique

```
> # Find the largest clique
> # The largest clique in a graph represents the largest subset of nodes where every node is connected to every other node within the subset.
> # Finding the size of the largest clique provides insights into the maximum level of connectivity within the graph
> bgraph.largestClique = igraph::clique_num(bgraph.adj)
> bgraph.largestClique
[1] 3
>
```

Q) Simplify function

The function is.simple() helps us to determine if a graph is simple, which means it has no loops and only one edge between any two vertices. After analyzing our graph, we found that it was already simple without any additional simplification required. We confirmed this by running the graph through the simplify() function, and the is.simple() function continued to return true.

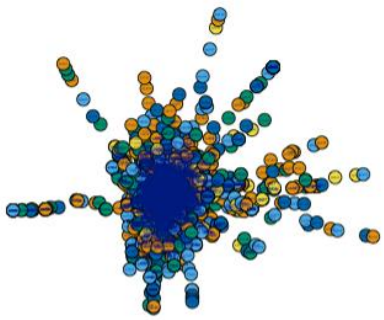
```
> # Simplify function
> sg <- simplify(g)
> is.simple(sg)
[1] TRUE
Warning message:
`is.simple()` was deprecated in igraph 2.0.0.
i Please use `is_simple()` instead.
This warning is displayed once every 8 hours.
Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
> is_simple(sg)
[1] TRUE
>
```

R) Detecting structures using walktrap.community()

The walktrap.community() function searches a graph for highly connected subgraphs or communities.

wc <- cluster_walktrap(agraph3)

plot(wc,g, vertex.size = 5, vertex.label.cex = 0.2, edge.arrow.size = 0.1, layout = layout.fruchterman.reingold)



S) Alpha Centrality:

Alpha centrality measures the extent to which a node's neighbors are influenced by the node itself.

acg <- alpha_centrality(agraph3)

sort(acg, decreasing = TRUE)

```
> # Alpha Centrality
> acg <- alpha_centrality(agraph3)
> sort(acg, decreasing = TRUE)
 [1]  0.978091572  0.829897048  0.800561326  0.799564369  0.729990577  0.717743933  0.702891318  0.676079523  0.635070451  0.621831793  0.587038427  0.552679839
[13]  0.534873604  0.491984491  0.458764562  0.424778502  0.408492152  0.371215617  0.359620248  0.356123553  0.351540932  0.345559121  0.331385669  0.322654503
[25]  0.297757628  0.275559945  0.271835169  0.234016691  0.225961221  0.213219540  0.209572918  0.199096304  0.191456593  0.185371545  0.173601194  0.172320887
[37]  0.159058749  0.148449525  0.140334360  0.116502858  0.109823354  0.099752510  0.076664572  0.073368427  0.038975024  0.034750814  0.009839388  0.007662714
[49] -0.010857413 -0.012969830 -0.069909584 -0.074625811 -0.076749007 -0.094327385 -0.108051755 -0.108687812 -0.109095596 -0.119410229 -0.119779594 -0.124957417
[61] -0.126725109 -0.130641259 -0.151105546 -0.152532578 -0.156066477 -0.165536541 -0.186695868 -0.194299619 -0.208237465 -0.210639036 -0.229394856 -0.233268926
[73] -0.260717739 -0.287562655 -0.297810821 -0.301098133 -0.359309782 -0.361192220 -0.389707331 -0.404332668 -0.415004309 -0.419383907 -0.421330573 -0.427042335
[85] -0.441454908 -0.446368674 -0.452449872 -0.486948875 -0.500508617 -0.511853836 -0.549510845 -0.558002193 -0.631622689 -0.684185803 -0.688099630 -0.702203726
[97] -0.769135920 -0.848870836 -0.857488696 -0.910050542
>
```

PART 4

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

a) Centeral Node

We can find the central node in two ways, either based on the sum of in and out degree or based on the node with the highest betweenness.

Identifying the most central node in a graph can provide insights into its structure and functioning.

most_central <- which.max(igraph:: degree(g, mode = "all"))

most_central

between <- igraph:: betweenness(g)

most_central <- which.max(between)

most_central

```
> # (a) central nodes
> # We can find the central node in two ways, either based on the sum of in and out degree or based on the node with the highest betweenness.
> most_central <- which.max(igraph:: degree(g, mode = "all"))
> most_central
8886
 156
```

b) Longest Path(s):

To find the longest path between two nodes in a graph, components() function is used to identify the connected components in the graph. A connected component is a subgraph in which any two vertices are connected to each other by paths. Once the largest connected component is identified, a subgraph is created containing only the vertices within that component. This subgraph will focus on the main connected structure of the graph. After creating the subgraph, each vertex in the subgraph is assigned a degree attribute. The degree of a vertex is the number of edges incident to it.

This process helps to focus the analysis on the main connected structure of the graph and extract relevant information about the longest path between two nodes and the connectivity of the vertices within the largest connected component

sg = induced_subgraph(g, which(igraph::components(g) $membership == 1))
V(sg)$degree = igraph::degree(sg)
result = dfs(sg, root = 1, dist = TRUE)$dist

sort(result, decreasing = TRUE)

```
> sg = induced_subgraph(g, which(igraph::components(g) $membership == 1))
> v(sg)$degree = degrees(sg)
Error in degrees(sg) : could not find function "degrees"
> v(sg)$degree = igraph::degree(sg)
Error in v(sg)$degree = igraph::degree(sg) : could not find function "v"
> V(sg)$degree = igraph::degree(sg)
> result = dfs(sg, root = 1, dist = TRUE)$dist
> sort(result, decreasing = TRUE)
61710 67561 36935 61709 61711 63401 63893 38748 69949 57193 61679 61680 61681 61682 41214 41212 41213 32077 75287 24951 55548 57192 59268 59269 59582 59583 60676
 9955  9955  9954  9954  9954  9954  9954  9953  9953  9953  9953  9953  9953  9953  9953  9953  9953  9952  9952  9952  9952  9952  9952  9952  9952  9952  9952
60677 68358 68359 61180 61687 61688 67355  4166  9044 32087 49257 24360 30164 55547 32139 39283 57788 57789 28171 59581 61099 28673 62459 67773 30931 13882 25170
 9952  9952  9952  9952  9952  9952  9952  9951  9951  9951  9951  9951  9951  9951  9951  9951  9951  9951  9951  9951  9951  9951  9951  9950  9950  9950  9950
46825 32078 32079 25244 15469 54098 18327 55830 55831  6962 20397 34901 61686 66399 66400 67158 67159  4168 75388 32085 25235 49256 14037 20036 55546 57804 57805
 9950  9950  9950  9950  9950  9950  9950  9950  9950  9950  9950  9950  9950  9950  9950  9950  9950  9949  9949  9949  9949  9949  9949  9949  9949  9949  9949
57806 57808 57809 57810 57811 57812 58535 59411 60955 32084 61393 66333 66334 66335 15875 75727 32074 20439 49237 30721 50441 54651 24757 29103 30819 55076 55077
 9949  9949  9949  9949  9949  9949  9949  9949  9949  9949  9949  9949  9949  9949  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948
55078 55079 55080 55081 56359 56360 56361 56363 56364 56365 56366 56367 56368 56369 56370 56371 56372 56374 56375 21621 22526 61100 61101 61102 61103 61464 64898
 9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948  9948
64900 43641 31985 25241 29933 46301 29996 16889 30827 48712 31983 14774 55490 31987 61134 65021 31113 66456 15382 43929 49221 45440 29995 32552 46925 46926 46927
 9948  9947  9947  9947  9947  9947  9947  9947  9947  9947  9947  9947  9947  9947  9947  9947  9947  9947  9946  9946  9946  9946  9946  9946  9946  9946  9946
46928 52471 53470 39160 30833 55471 55864 55866 64331 64332 66688 42982  4339 44536 15369 15949 46168 46174 46177 33572 11046  4508 55728   957 57688 57689 31984
 9946  9946  9946  9946  9946  9946  9946  9946  9946  9946  9946  9945  9945  9945  9945  9945  9945  9945  9945  9945  9945  9945  9945  9945  9945  9945  9945
60396 64349 64351 31474 67521 36942 42649 43080 44105 44537  3427 20103 27301 74604 52472 54195 55036 56793 57786 28296 28297 63699 64504 64881 64882 64885 67237
 9945  9945  9945  9945  9945  9944  9944  9944  9944  9944  9944  9944  9944  9944  9944  9944  9944  9944  9944  9944  9944  9944  9944  9944  9944  9944  9944
67580 41079 14801 36287 44100 44855 46175 38642 48822 39098 50417 50418 75220 16086 56187 56518 57247 26951 30937 30938 37186 60802 60803 60804 60805 60807 60808
 9944  9944  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943
60809 60810 60811 60812 60813 60814 61663 40213 64011 66466 66467 66468 66539 66540 66809 38810   991  8649 16491 28544 22388 21041 41758 17158 54499 54500 18925
 9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9943  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942
55938 56514 56515 56516 20383  2410 21040 21511 58449 58450 60671 60672 30939 30940 30941 65320 66598 66709 66710 35948 36048 36049 36050 36922 41014 41015 41959
 9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9942  9941
75166 73735 65328 43488 74674 20753 69738 28155 13607 11463 65673 48127 13819 13820 13822  2009 46777 51745  6894 17157 17294 32483 19335  6780 19504 56594 56596
 9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941
56597 56598 20469 56998 56999 57205 22252 57784 36100 36099 36101 41543 41726 46665 15885 28711 41960 41961 41962 41963 41965 43599  5731  7577 58317 65650 75563
 9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9941  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940
11281 75402 75782 16354 17341 46703 16888 10698 10344 47286 23822 48666 18235 18858  5450 14725 18739 27955 62344 34004 66614 66615 67097 40321 67098 67934 67935
 9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940  9940
38652 40857 40858 41073 41514 41515 25765 43166 43167 57184 75216  8520  7395   191 75515 34377 24427 46294 46295 75072 23586 13870 73615 49459 49460 49461 25587
 9940  9940  9940  9940  9940  9940  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939
30029  1344 56537 56538  4032 56567  6387 57000 57207 37334 60494 60496 60497 60498 60499 60500 61870 64514 64515 65134 66300 66301 66302 66980 34779 67811 67812
 9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939  9939
68123 36928 36929 36930 17186  6162 17698 32660 43165  7225 43817 15997 44123  4452 61562 17263 33131 67813 69883 47584  3520  6841  6799 13258 49434 27668 49716
 9939  9939  9939  9939  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938
59691 14897 51447 16841 32417 54816 54878  1034 55113 55862 27306 57848 57849 59690 60629  6498 61309 35737 41703 64479 65346 65380 65381 66617 67001 34881 67830
 9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938  9938
37036  8386 29275  7164 43094 43160 14800  1455 25500 70272 43903 44124 61884 75377  3821 48184 12430 48784 24482 49016 15936 14316 49521 49522 49523 49524 49937
 9938  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937
41232 49967 14358 31200 36343  6476 52451 52452 52453 20375 20536 54348 54349 54350 54351 54352 55863 55925 28101 59905 59907 60111 60112 60627 60630 60791 61308
 9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937
61310 61311 62702 41705 29556 64712 65534 65535 65693 65835 36920 26603   972  7823 13662 26608  6623 42975 36094 45058 71572 11030 14350 46500 26631 12335  7316
 9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9937  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936
69727 47637 47638 28499 30611 16504 48919 48920 75181 26124 32614 28481 34727 49876 49877 75737 72672 28077 50358 15255 50852 50854 15937 16052 23968 51839 51840
 9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936  9936
```

c) Largest Clique

The largest clique in a graph is a fully connected subgraph where every node is directly connected to every other node within that subgraph.

largest_cliques(agraph3)

```
> largest_cliques(agraph3)
[[1]]
+ 4/100 vertices, from 9b0e28b:
[1] 49 45 13 52

[[2]]
+ 4/100 vertices, from 9b0e28b:
[1] 49 45 83 40

[[3]]
+ 4/100 vertices, from 9b0e28b:
[1] 70 46 82 18

[[4]]
+ 4/100 vertices, from 9b0e28b:
[1] 15 94 89 33

[[5]]
+ 4/100 vertices, from 9b0e28b:
[1] 15 83 20  2

[[6]]
+ 4/100 vertices, from 9b0e28b:
[1] 18 86 73 72

[[7]]
+ 4/100 vertices, from 9b0e28b:
[1] 19 65 50 36

[[8]]
+ 4/100 vertices, from 9b0e28b:
[1] 20 68 53  1

[[9]]
+ 4/100 vertices, from 9b0e28b:
[1] 20 68 53 82

[[10]]
+ 4/100 vertices, from 9b0e28b:
[1] 20  2 82 53

[[11]]
+ 4/100 vertices, from 9b0e28b:
[1] 20  2 82 83

[[12]]
+ 4/100 vertices, from 9b0e28b:
[1] 22 61 92 40

[[13]]
+ 4/100 vertices, from 9b0e28b:
[1] 23  6 52 13

[[14]]
+ 4/100 vertices, from 9b0e28b:
[1] 23  6 52 43
```

```
[[15]]
+ 4/100 vertices, from 9b0e28b:
[1] 25 94 27 48

[[16]]
+ 4/100 vertices, from 9b0e28b:
[1] 25 94 43 52

[[17]]
+ 4/100 vertices, from 9b0e28b:
[1] 25 94 89 48

[[18]]
+ 4/100 vertices, from 9b0e28b:
[1] 25 94 89 52

[[19]]
+ 4/100 vertices, from 9b0e28b:
[1] 25 93 43 11

[[20]]
+ 4/100 vertices, from 9b0e28b:
[1] 29 33 95 75

[[21]]
+ 4/100 vertices, from 9b0e28b:
[1]  33 100  72  75

[[22]]
+ 4/100 vertices, from 9b0e28b:
[1]  33 100  72  86

[[23]]
+ 4/100 vertices, from 9b0e28b:
[1] 33 52  6 41

[[24]]
+ 4/100 vertices, from 9b0e28b:
[1] 33 52 94 89

[[25]]
+ 4/100 vertices, from 9b0e28b:
[1] 37 93 76 43

[[26]]
+ 4/100 vertices, from 9b0e28b:
[1]  37  72 100  75

[[27]]
+ 4/100 vertices, from 9b0e28b:
[1] 40 92 83 61

[[28]]
+ 4/100 vertices, from 9b0e28b:
[1] 51 66 91 79

[[29]]
+ 4/100 vertices, from 9b0e28b:
[1] 51 66 91 94
```

```
[[30]]
+ 4/100 vertices, from 9b0e28b:
[1] 52 13 56  6

[[31]]
+ 4/100 vertices, from 9b0e28b:
[1] 61 13 75 72

[[32]]
+ 4/100 vertices, from 9b0e28b:
[1] 66 94 91 67

[[33]]
+ 4/100 vertices, from 9b0e28b:
[1] 66 79 91 67

[[34]]
+ 4/100 vertices, from 9b0e28b:
[1] 69 96 99  7

[[35]]
+ 4/100 vertices, from 9b0e28b:
[1] 69 96 99 95

[[36]]
+ 4/100 vertices, from 9b0e28b:
[1] 79  5  4  3

>
```

d) ego(s)

The ego() function calculates the ego network for each node in the graph agraph3. An ego network is a subgraph that includes a focal node (the ego) and all its neighboring nodes (alters). By default, the ego network includes the focal node and all nodes that are directly connected to it.

ego.graph = igraph::ego(agraph3)

ego.graph

```
> # (d) ego(s)
> ego.graph = igraph::ego(agraph3)
> ego.graph
[[1]]
+ 17/100 vertices, from 9b0e28b:
 [1]  1  8 20 28 29 42 45 52 53 56 68 73 76 77 89 96 98

[[2]]
+ 13/100 vertices, from 9b0e28b:
 [1]  2 15 20 25 26 31 53 58 79 82 83 86 92

[[3]]
+ 14/100 vertices, from 9b0e28b:
 [1]  3  4  5  7 28 43 60 79 84 87 90 95 98 99

[[4]]
+ 16/100 vertices, from 9b0e28b:
 [1]   4   3   5   6  14  15  38  39  40  46  50  54  79  94  98 100

[[5]]
+ 17/100 vertices, from 9b0e28b:
 [1]  5  3  4 13 16 17 18 25 30 34 37 41 63 79 87 89 91

[[6]]
+ 18/100 vertices, from 9b0e28b:
 [1]   6   4  11  13  15  22  23  33  41  43  52  56  78  82  83  91  98 100

[[7]]
+ 17/100 vertices, from 9b0e28b:
 [1]  7  3 29 36 47 48 64 66 69 71 74 76 77 88 94 96 99

[[8]]
+ 16/100 vertices, from 9b0e28b:
 [1]   8   1  12  16  25  36  53  54  62  66  69  76  77  86  98 100

[[9]]
+ 15/100 vertices, from 9b0e28b:
 [1]   9  17  20  21  52  65  66  69  71  72  87  94  96  98 100

[[10]]
+ 14/100 vertices, from 9b0e28b:
 [1] 10 20 25 37 51 57 62 63 64 73 74 77 78 88

[[11]]
+ 14/100 vertices, from 9b0e28b:
 [1] 11  6 14 17 25 26 39 43 50 54 87 93 96 97

[[12]]
+ 17/100 vertices, from 9b0e28b:
 [1] 12  8 14 21 24 29 31 34 43 45 46 48 56 66 69 85 97

[[13]]
+ 20/100 vertices, from 9b0e28b:
 [1] 13  5  6 14 23 36 45 46 49 52 56 60 61 71 72 75 81 90 96 97

[[14]]
+ 12/100 vertices, from 9b0e28b:
 [1] 14  4 11 12 13 21 23 46 58 87 92 93

[[15]]
```

```
[[15]]
+ 13/100 vertices, from 9b0e28b:
 [1] 15  2  4  6 20 33 47 66 68 83 89 94 97

[[16]]
+ 20/100 vertices, from 9b0e28b:
 [1] 16  5  8 22 23 25 31 36 46 47 49 55 65 71 74 78 82 92 96 98

[[17]]
+ 15/100 vertices, from 9b0e28b:
 [1] 17  5  9 11 19 32 40 44 72 85 88 92 93 95 96

[[18]]
+ 18/100 vertices, from 9b0e28b:
 [1] 18  5 32 39 46 48 52 59 70 72 73 79 80 82 85 86 87 89

[[19]]
+ 15/100 vertices, from 9b0e28b:
 [1] 19 17 33 36 38 50 54 65 67 70 80 81 83 89 90

[[20]]
+ 21/100 vertices, from 9b0e28b:
 [1]  20   1   2   9  10  15  21  24  26  31  34  40  42  43  53  62  68  82  83  96 100

[[21]]
+ 15/100 vertices, from 9b0e28b:
 [1] 21  9 12 14 20 28 55 56 60 65 67 77 78 85 90

[[22]]
+ 18/100 vertices, from 9b0e28b:
 [1] 22  6 16 28 40 42 44 60 61 64 70 71 72 74 81 82 92 94

[[23]]
+ 16/100 vertices, from 9b0e28b:
 [1] 23  6 13 14 16 30 31 38 43 52 53 74 80 83 92 95

[[24]]
+ 9/100 vertices, from 9b0e28b:
[1] 24 12 20 28 30 40 75 77 97

[[25]]
+ 22/100 vertices, from 9b0e28b:
 [1] 25  2  5  8 10 11 16 27 29 43 44 48 52 66 77 84 87 89 90 93 94 99

[[26]]
+ 14/100 vertices, from 9b0e28b:
 [1] 26  2 11 20 40 44 45 47 61 62 66 90 91 96

[[27]]
+ 12/100 vertices, from 9b0e28b:
 [1] 27 25 37 48 53 54 57 58 85 93 94 98

[[28]]
+ 12/100 vertices, from 9b0e28b:
 [1] 28  1  3 21 22 24 34 37 62 83 91 99

[[29]]
+ 13/100 vertices, from 9b0e28b:
 [1] 29  1  7 12 25 33 42 43 45 50 75 86 95
```

e) Power Centrality

Power centrality is a measure of node centrality in a network that considers the number of neighbors a node has and their centrality.

Power centrality can be used to identify influential nodes in a network, detect central nodes in communication or information flow, or understand the spread of influence or information in a network.

powerCentrality <- power_centrality(agraph3)

PowerCentrality

```
>  # (e) power centrality.
> powerCentrality <- power_centrality(agraph3)
> powerCentrality
  [1] -0.57684484 -0.59069025 -1.56159640 -1.29812055 -0.73474656 -0.81664528 -1.18120597 -0.78120975 -1.12784301 -0.78865458 -0.46605185 -1.36412227 -1.08867109
 [14] -0.25894084 -0.62139465 -0.75813569 -0.29716364 -0.92929537 -0.98153172 -1.06057179 -1.19362338 -1.02728231 -1.30916420 -0.34693034 -0.59489433 -1.69614835
 [27] -0.74175520 -0.66459905 -1.01710675 -1.02694346 -1.38697000 -0.72513534 -0.37884966 -1.24702656 -0.92735744 -1.03724677 -1.75227442 -0.90837026 -1.19060761
 [40] -0.74733761 -0.75931024 -0.61338471 -1.54506681 -1.32238624 -0.82588424 -0.88551660 -1.27491318 -1.49684557 -0.58748239 -1.42151642 -1.09564675 -1.37656193
 [53] -0.54264746 -1.30213838 -0.02009872 -0.66801615 -0.18296442 -0.27256658 -0.77147689 -1.33247299 -1.62299978 -1.00393254 -1.05602064 -0.52770641 -1.54865735
 [66] -0.84706505 -0.49652771 -1.05732980 -1.28833042 -1.01652323 -0.64423497 -1.15657853 -1.30392424 -0.98780619 -0.60038202 -0.70271070 -1.32689412 -1.01748085
 [79] -1.03203243 -0.24770580 -0.81051753 -1.06925959 -1.13139707 -0.33478523 -0.85008892 -0.98585837 -0.18387902 -0.41037012 -1.03365411 -1.70405435 -0.42670551
 [92] -0.91036714 -1.24875350 -1.10843329 -1.42930636 -0.88164132 -0.72178994 -1.11063648 -0.15605192 -0.71010077
>
```

PART 5 Discussion

We have successfully completed a project focused on working with datasets and constructing graphs, particularly those of medium to large scale. Throughout this project, we honed our skills in utilizing R packages such as igraph and sna, allowing us to efficiently analyze and visualize graphs.

Our project involved various tasks aimed at improving graph clarity and interpretability. We employed techniques such as charting, visualization, and parameter adjustment to simplify and enhance graph representations. By doing so, we gained valuable insights into the underlying structure and interactions within the graphs.

Furthermore, we delved into essential graph metrics to better understand the problem space. Metrics like power centrality, longest paths, greatest cliques, egos, and alpha centrality provided us with crucial insights into the significance and connectivity of nodes within the graphs. Additionally, we explored the concept of random walks to identify communities within the graph, facilitating the discovery of patterns, structures, and connections within our data.

Having completed this project, we now possess a strong foundation in leveraging R for handling large datasets, constructing and visualizing graphs, and applying diverse functions and metrics to extract insights relevant to our problem domain. We are confident in our ability to efficiently analyze complex networks and simplify graph representations to facilitate more effective analysis and decision-making processes.