Name: Manali Mange **Academic Year:** 2021 - 2022

Class:MSC CS -I

Roll No. : 39

Social Network Analysis

PRACTICAL 1

Aim: Write a program to compute the following for a given network: (i) Number of edges, (ii)Number of Nodes, (iii)Degree of Nodes; (iv)Node with Lowest degree, (v)the adjacency list, (vi) matrix of the graph

Software- RStudio

Packages and Methods:

1. **The igraph package-** igraph is a library and R package for network analysis.
   The main goals of the igraph library is to provide a set of data types and functions for
   1) pain-free implementation of graph algorithms,
   2) fast handling of large graphs, with millions of vertices and edges,
   3) allowing rapid prototyping via high level languages like R.

2. **library()**
   library() loads and attach add-on packages.

3. **graph.formula():-**Creating (small) graphs via a simple interface
   This function is useful if you want to create a small (named) graph quickly, it works for both directed and undirected graphs.

4. **plot():-** Use to plot any graph
   Use to plot the directed or undirected graph.

5. **ecount():-**The size of the graph (number of edges)
   Returns the count of number of edges in graph

6. **vcount():-**Order (number of vertices) of a graph
   Returns the count of number of vertices in graph

7. **E()-**Edges of a graph
   An edge sequence is a vector containing numeric edge ids, with a special class attribute that allows custom operations: selecting subsets of edges based on attributes, or graph structure, creating the intersection, union of edges, etc.

8. **V()-**Vertices of a graph

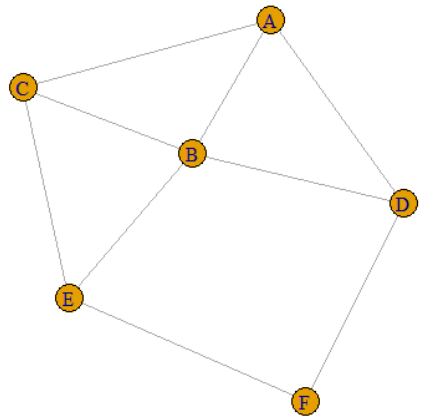
Create a vertex sequence (vs) containing all vertices of a graph.

9. **degree()-**Degree and degree distribution of the vertices
   The degree of a vertex is its most basic structural property, the number of its adjacent edges.
   Mode-Character string, “out” for out-degree, “in” for in-degree or “total” for the sum of the two. For undirected graphs this argument is ignored. “all” is a synonym of “total”.

10. **max() and min()-**Maxima and Minima
    Returns the (regular or parallel) maxima and minima of the input values.
11. **get.adjacency()-** Convert a graph to an adjacency matrix
12. **get.adjlist()-**Adjacency lists
    Create adjacency lists from a graph, either for adjacent edges or for neighboring vertices

R Code:

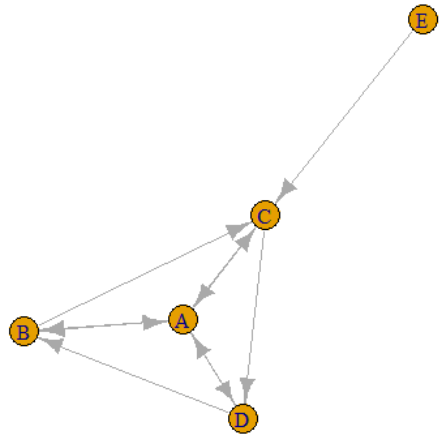
```
>library(igraph)
>ug<-graph.formula(A-B,A-C,A-D,B-C,B-D,B-E,C-E,D-F,E-F)        //undirected graph
>plot(ug)                                                      //plotting undirected graph
```




```
>ecount(ug)                                                    //(i)Number of edges
[1] 9
>vcount(ug)                                                    //(ii)Number of Nodes

[1] 6
>E(ug)
+ 9/9 edges from 69eee83 (vertex names):
[1] A--B A--C A--D B--C B--D B--E C--E D--F E—F
>V(ug)
+ 6/6 vertices, named, from 69eee83:
[1] A B C D E F
```

>dg<-graph.formula(A-+B,A+-B,A++C,B-+C,C-+D,D++A,D-+B,E-+C)

>plot(dg)



>degree(ug) //(iii)degree of nodes(undirected)

A B C D E F

3 4 3 3 3 2

>degree(dg,mode="in") //(iii)degree of nodes(directed) Mode-in

A B C D E

3 2 3 2 0

>degree(dg,mode="out") //(iii)degree of nodes(directed)Mode-Out

A B C D E

3 2 2 2 1

>degree(dg) //(iii)degree of nodes(directed)

A B C D E

6 4 5 4 1

>V(ug)$name[degree(ug)==max(degree(ug))] //(iv) Node with Highest degree

[1] "B"

>V(dg)$name[degree(dg,mode="in")==min(degree(dg,mode = "in"))] //(iv) Node with Lowest degree

[1] "E"

>get.adjacency(ug) //(vi)Adjacency Matrix

6 x 6 sparse Matrix of class "dgCMatrix"

A B C D E F

A . 1 1 1 . .

B 1 . 1 1 1 .

C 1 1 . . 1 .

D 1 1 . . . 1

E . 1 1 . . 1

F . . . 1 1 .

>get.adjlist(ug) //(v)Adjacency List

$A

+ 3/6 vertices, named, from 69eee83:

[1] B C D

$B

+ 4/6 vertices, named, from 69eee83:

[1] A C D E

$C

+ 3/6 vertices, named, from 69eee83:

[1] A B E

$D

+ 3/6 vertices, named, from 69eee83:

[1] A B F

$E

+ 3/6 vertices, named, from 69eee83:

[1] B C F

$F

+ 2/6 vertices, named, from 69eee83:

[1] D E

PRACTICAL-2

Aim: Perform following tasks: (i) View data collection forms and/or import onemode/two-mode datasets; (ii) Basic Networks matrices transformations

Software: RStudio

Packages and Method

Getwd()-Get or Set Working Directory. Returns an absolute filepath representing the current working directory of the R process; setwd(dir) is used to set the working directory to dir.

read.csv- Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

Head()-Return the First or Last Parts of an Object. Returns the first or last parts of a vector, matrix, table, data frame or function. Since head() and tail() are generic functions, they may also have been extended to other classes.

graph.data.frame()-Creating igraph graphs from data frames or vice-versa. This function creates an igraph graph from one or two data frames containing the (symbolic) edge list and edge/vertex attributes

as.matrix- attempts to turn its argument into a matrix.

get.adjacency()-Convert a graph to an adjacency matrix. Sometimes it is useful to work with a standard representation of a graph, like an adjacency matrix

Code:

(i) View data collection forms and/or import onemode/two-mode datasets;

```
getwd()

[1] "C:/Users/Admin/Documents"

nodes <- read.csv("E:/Chinmay Classes Online/Masters_Computer_Science/SEM3(02-08-
21)/Social_Network_Analysis/Practicals/Codes/PRACTICAL_2/nodes.csv")

head(nodes)

  id              media media.type type.label audience.size

1 s01          NY Times          1  Newspaper           20

2 s02    Washington Post          1  Newspaper           25

3 s03 Wall Street Journal          1  Newspaper           30

4 s04          USA Today          1  Newspaper           32

5 s05          LA Times          1  Newspaper           20

6 s06      New York Post          1  Newspaper           50

links <- read.csv("E:/Chinmay Classes Online/Masters_Computer_Science/SEM3(02-08-
21)/Social_Network_Analysis/Practicals/Codes/PRACTICAL_2/edges.csv")

head(links)

  from  to weight      type

1  s01 s02     10 hyperlink
```

2 s01 s02 12 hyperlink

3 s01 s03 22 hyperlink

4 s01 s04 21 hyperlink

5 s04 s11 22 mention

6 s05 s15 21 mention

(ii) Basic Networks matrices transformations

net <- graph.data.frame(d=links, vertices=nodes, directed=T)

m=as.matrix(net)

get.adjacency(m)

s01 . 2 1 1 . . . . . 1 . .

s02 1 . 1 . . . . . 1 1 . . . . . .

s03 1 . . 1 1 . . 1 . 1 1 1 . . . . .

s04 . . 1 . . 1 . . . . 1 1 1

s05 1 1 . . . . . 1 . . . . . 1 . .

s06 . . . . . 1 . . . . . 1 1

s07 . . 1 . . . . 1 . 1 . . . 1 . . .

s08 . . 1 . . . 1 . 2 . . . . . .

s09 . . . . . 1 . . . . . .

s10 . . 1 . . . . . .

s11 . . . . . .

s12 . . . . . 1 . . . . . 1 1 . . .

s13 . . . . . 1 1

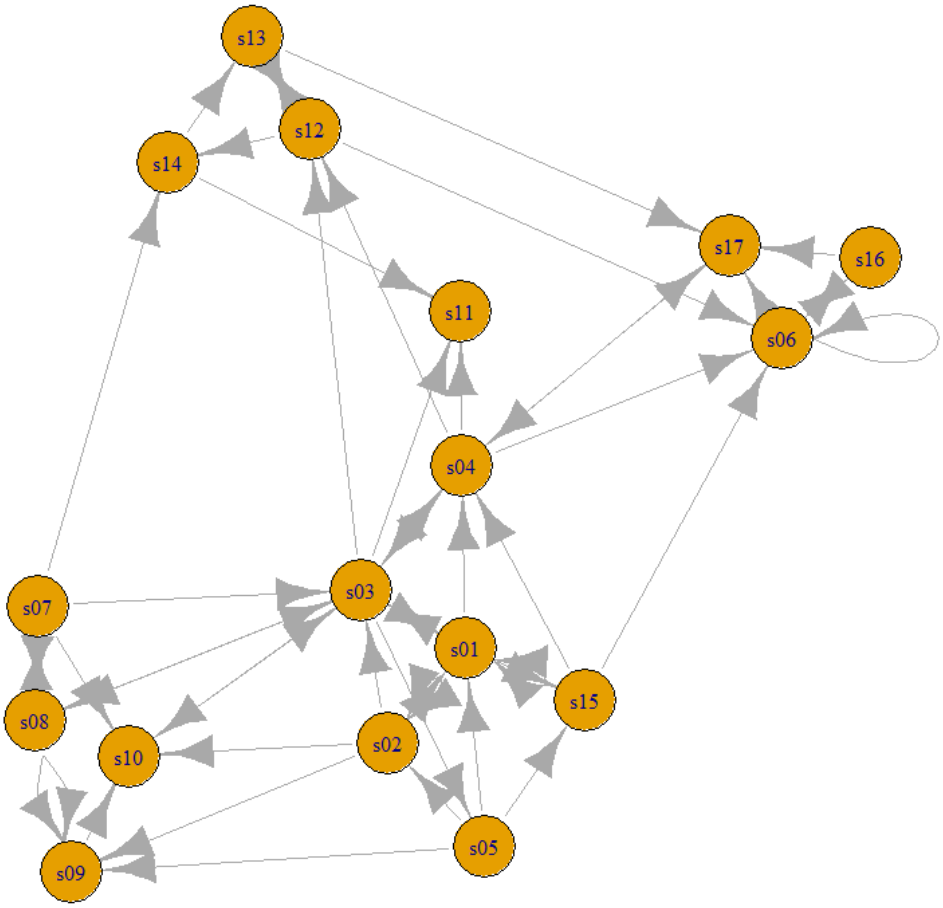
s14 . . . . . 1 . 1 . . . .

s15 2 . . 1 . 1 . . . . . .

s16 . . . . . 1 1

s17 . . . 1 . . . . . .

plot(net)



PRACTICAL- 3

Aim: Compute the following node level measures: (i) Density; (ii) Degree; (iii) Reciprocity; (iv) Transitivity; (v) Clustering.

Software – Rstudio

Theory

Packages and method

1. Density : It refers to the “connection” between participants. It defined as the number of connections a participant has, divided by the total possible connections a participant could have.
2. vcount() : The size of the graph(number of vertices)
   Returns the count of number of vertices in graph
3. ecount() : The size of the graph(number of edges)
   Returns the count of number of edges in graph
4. degree : The degree of a vertex is its most basic structural property, the number of its adjacent edges. Degree takes one or more graphs and returns the degree centralities of positions within the graph.
5. Reciprocity: Calculates the reciprocity of a directed graph.
   The measure of reciprocity defines the proportion of mutual connections, in a directed graph. It is most commonly defined as the probability that the opposite counterpart of a directed edge is also included in the graph.
6. graph.formula(): Creating (small) graphs via a simple interface
   This function is useful if you want to create a small (named) graph quickly, it works for both directed and undirected graphs.
7. plot(): Use to plot any graph
   Use to plot the directed or undirected graph.
8. Dyad.census() : Classify dyads in a directed graphs. The relationship between each pair of vertices is measured. It can be in three states: mutual, asymmetric or non-existent.
   A named numeric vector with 3 elements:
   mut : The number of pairs with mutual connections.
   asym : The number of pairs with non-mutual connections.
   null : The number of pairs with no connection between them.
9. Transitivity : Transitivity measures the probability that the adjacent vertices of a vertex are connected. This is sometimes also called the clustering coefficient.

10. Adjacent.triangles : Count how many triangles a vertex is part of in a graph or just list the triangles of a graph.

11. barabasi.game : generate a scale free graph according to the barabasi-albert model. The BA-model is a very simple stochastic algorithm for building a graph.

N : number of vertices

P : numeric constant, the contant out-degree of the vertices.

Directed : whether to create directed graph

12. watts.strogatz.game : Generate a graph according to the Watts-Strogatz network model.

dim : Interger constant, the dimension of the starting lattice.

size : Integer constant, the size of the lattice along each dimension.

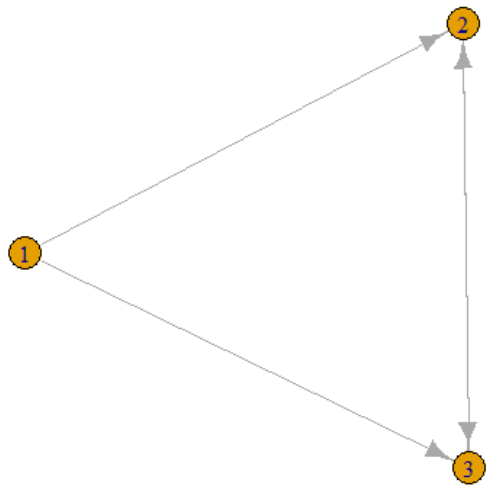
nei : Interger constant, the neighbourhood within which the vertices of the lattice will be connected.

p : Real constant between zero and one, the rewiring probability

13. simplify() : Simple graphs are graphs which do not contain loop and multiple edges.

Code:

```
g <- graph.famous("Krackhardt_Kite")
vcount(g)
[1] 10
ecount(g)
[1] 18
ecount(g)/(vcount(g)*(vcount(g)-1)/2)
[1] 0.4
degree(net)
s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 s11 s12 s13 s14 s15 s16 s17
 10   7  13   9   5   8   5   6   5   5   3   6   4   4   6   3   5
dg <- graph.formula(1-+2, 1-+3, 2++3)
plot(dg)
```




```
reciprocity(dg)
[1] 0.5
```

dyad.census(dg)

$mut

[1] 1

$asym

[1] 2

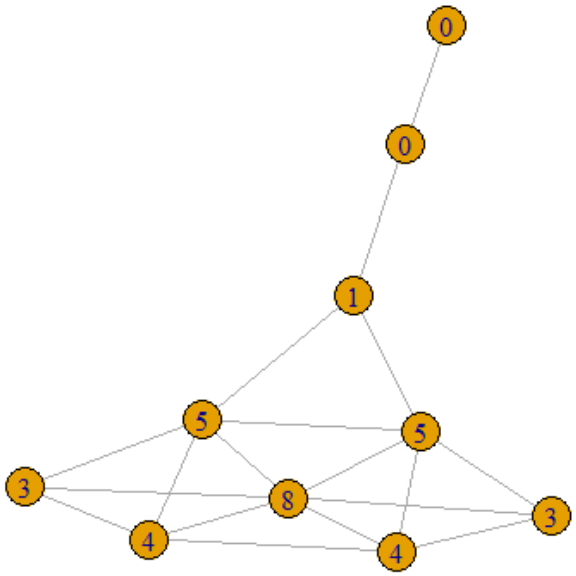
$null

[1] 0

kite <- graph.famous("Krackhardt_Kite")

atri <- adjacent.triangles(kite)

plot(kite, vertex.label=atri)



transitivity(kite, type="local")

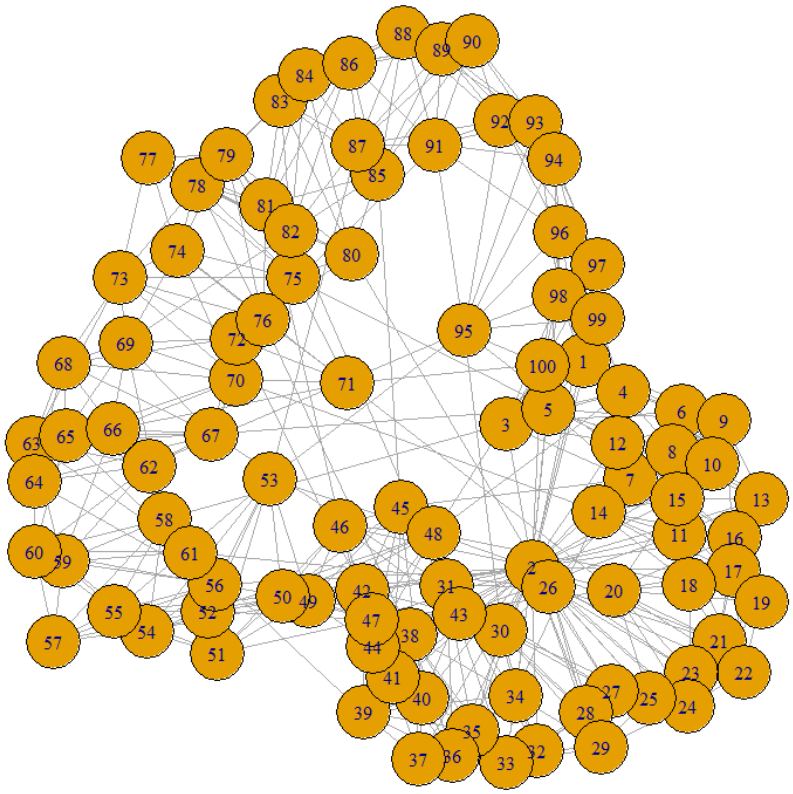
[1] 0.6666667 0.6666667 1.0000000 0.5333333 1.0000000 0.5000000 0.5000000 0.3333333 0.0000000 NaN

adjacent.triangles(kite) / (degree(kite) * (degree(kite)-1)/2)

[1] 0.6666667 0.6666667 1.0000000 0.5333333 1.0000000 0.5000000 0.5000000 0.3333333 0.0000000 NaN

library(igraph)

```
g2 <- barabasi.game(50, p=2, directed=F)
g1 <- watts.strogatz.game(1, size=100, nei=5, p=0.05)
g <- graph.union(g1,g2)
g <- simplify(g)
plot(g)
```



PRACTICAL -4

Aim: For a given network find the following: (i) Length of the shortest path from a given node to another node; (ii) the density of the graph

Software: RStudio

Theory

Dijkstra's Algorithm

Dijkstra's Algorithm finds the shortest path between a given node (which is called the "source node") and all other nodes in a graph. This algorithm uses the weights of the edges to find the path that minimizes the total distance (weight) between the source node and all other nodes.

The density of the graph

The density of a graph is a measure of how many ties between actors exist compared to how many ties between actors are possible, given the graph size (number of nodes) and the graph order (number of links).

Packages and Methods:

Library()-load and attach add-on packages.

Igraph- igraph is a library and R package for network analysis. The main goals of the igraph library is to provide a set of data types and functions for 1) pain-free implementation of graph algorithms, 2) fast handling of large graphs, with millions of vertices and edges, 3) allowing rapid prototyping via high level languages like R.

as.matrix- attempts to turn its argument into a matrix.

read.table- Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

is.na- Not Available' / Missing Values.

graph.adjacency- a flexible function for creating igraph graphs from adjacency matrices.

shortest.paths- calculates the length of all the shortest paths from or to the vertices in the network. shortest_paths calculates one shortest path (the path itself, and not just its length) from or to the given vertex.

graph.formula- This function is useful if you want to create a small (named) graph quickly, it works for both directed and undirected graphs.

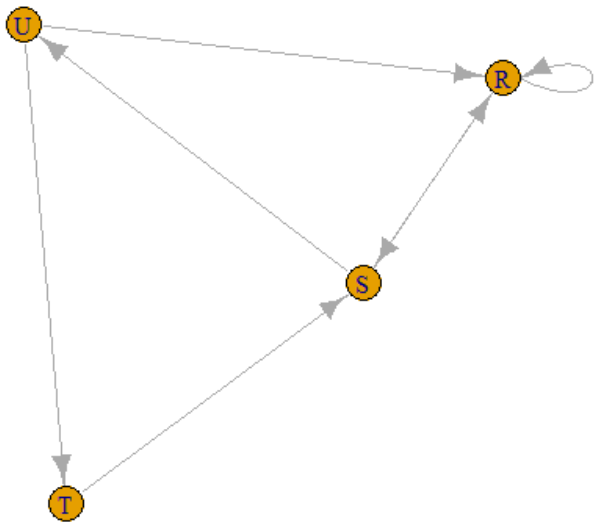
graph.density- The density of a graph is the ratio of the number of edges and the number of possible edges.

Simplify()-Simple graphs are graphs which do not contain loop and multiple edges.

Code:

(i) Length of the shortest path from a given node to another node

```
library(igraph)
 matt <- as.matrix(read.table(text=
                                "node R S T U
                               R 7 5 0 0
                               S 7 0 0 2
                               T 0 6 0 0
                               U 4 0 1 0", header=T))
 nms <- matt[,1 ]
 matt <- matt[, -1]
 colnames(matt) <- rownames(matt) <- nms
 matt[is.na(matt)] <- 0
 g <- graph.adjacency(matt,weighted=TRUE)
 plot(g)
```




```
 s.paths <- shortest.paths(g, algorithm = "dijkstra")
 print(s.paths)
R S T U
R 0 5 5 4
S 5 0 3 2
T 5 3 0 1
```

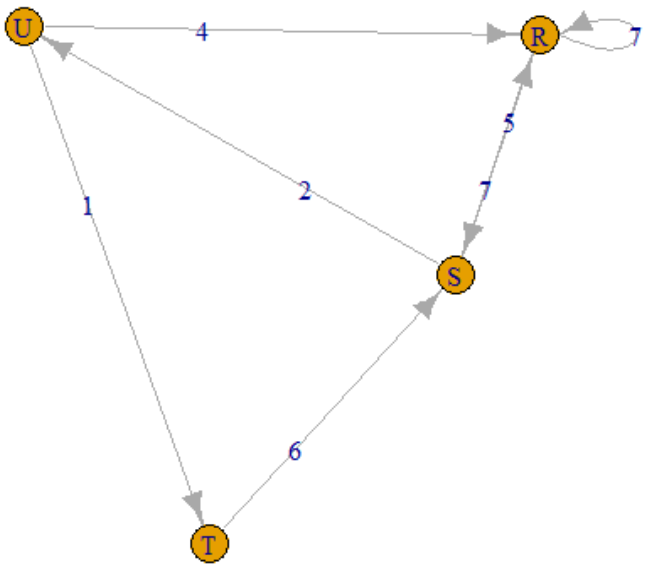
U 4 2 1 0

shortest.paths(g, v="R", to="S")

S

R 5

plot(g, edge.label=E(g)$weight)

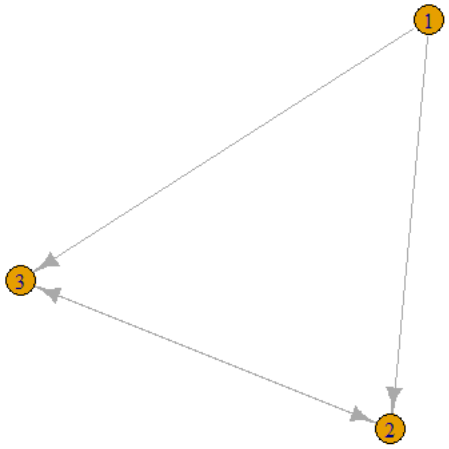


(ii) the density of the graph

library(igraph)

dg <- graph.formula(1-+2, 1-+3, 2++3)

plot(dg)



graph.density(dg, loops=TRUE)

[1] 0.4444444

Without considering loops

graph.density(simplify(dg), loops=FALSE)

[1] 0.6666667

PRACTICAL -5

Aim: Write a program to distinguish between a network as a matrix, a network as an edge list, and a network as a sociogram (or "network graph") using 3 distinct networks representatives of each.

Software: RStudio

Theory

Sociogram

A sociogram is a graphic representation of social links that a person has. It is a graph drawing that plots the structure of interpersonal relations in a group situation.

Network as an Matrix

Network matrices show how objects in a system are related to one another. Compared to other network diagrams like force-directed graphs, network matrices are more structured and can be easier to read.

Network as an Edge

An edge list is a two-column list of the two nodes that are connected in a network. In the case of a directed network, the convention is that the edge goes from the vertex in the first column to the vertex in the second column. In an undirected network, the order of the vertices don't matter.

Packages and Methods

Library()-load and attach add-on packages.

Igraph- igraph is a library and R package for network analysis. The main goals of the igraph library is to provide a set of data types and functions for 1) pain-free implementation of graph algorithms, 2) fast handling of large graphs, with millions of vertices and edges, 3) allowing rapid prototyping via high level languages like R.

graph.formula- This function is useful if you want to create a small (named) graph quickly, it works for both directed and undirected graphs.

get.adjacency- Convert a graph to an adjacency matrix

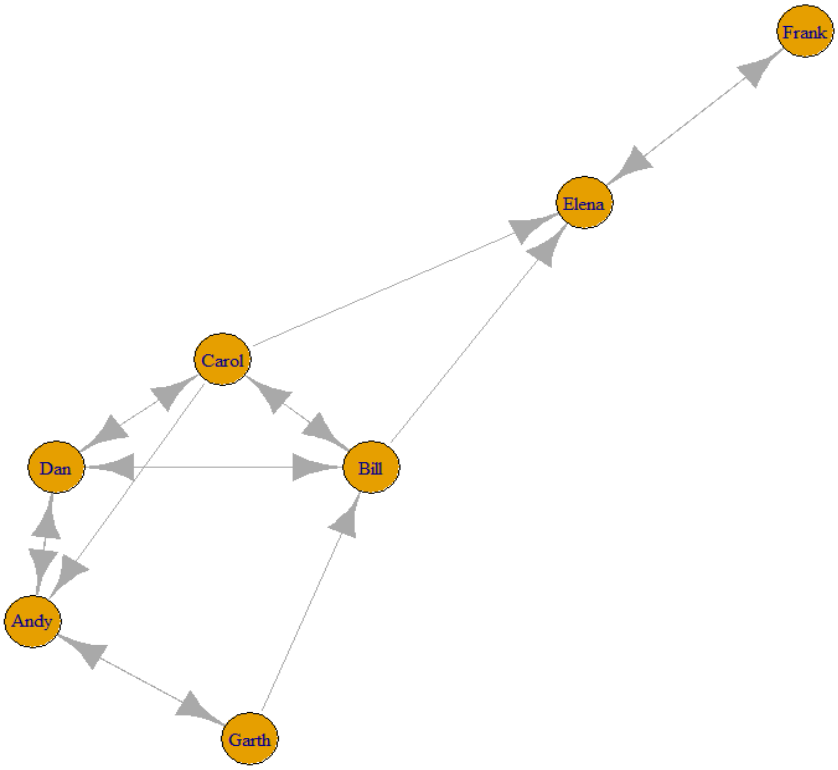
E()-Edges of a graph. An edge sequence is a vector containing numeric edge ids, with a special class attribute that allows custom operations: selecting subsets of edges based on attributes, or graph structure, creating the intersection, union of edges, etc.

get.adjedgelist- Create adjacency lists from a graph, either for adjacent edges or for neighboring vertices.

Code:

1. **a network as a sociogram (or "network graph")**

```
library(igraph)

ng<-graph.formula(Andy++Garth,Garth-+Bill,Bill-
                  +Elena,Elena++Frank,Carol-+Andy,Carol-
                  +Elena,Carol++Dan,Carol++Bill,Dan++Andy,Dan++Bill)

plot(ng)
```



2. a network as a matrix

get.adjacency(ng)

7 x 7 sparse Matrix of class "dgCMatrix"

	Andy	Garth	Bill	Elena	Frank	Carol	Dan
Andy	.	1	.	.	.	.	1
Garth	1	.	1	.	.	.	.
Bill	.	.	.	1	.	1	1
Elena	.	.	.	.	1	.	.
Frank	.	.	.	1	.	.	.
Carol	1	.	1	1	.	.	1
Dan	1	.	1	.	.	1	.

3. a network as an edge list.

E(ng)

16/16 edges from bb1d6c8 (vertex names):

[1] Andy ->Garth Andy ->Dan Garth->Andy Garth->Bill Bill ->Elena Bill ->Carol Bill ->Dan

[8] Elena->Frank Frank->Elena Carol->Andy Carol->Bill Carol->Elena Carol->Dan Dan ->Andy

[15] Dan ->Bill Dan ->Carol

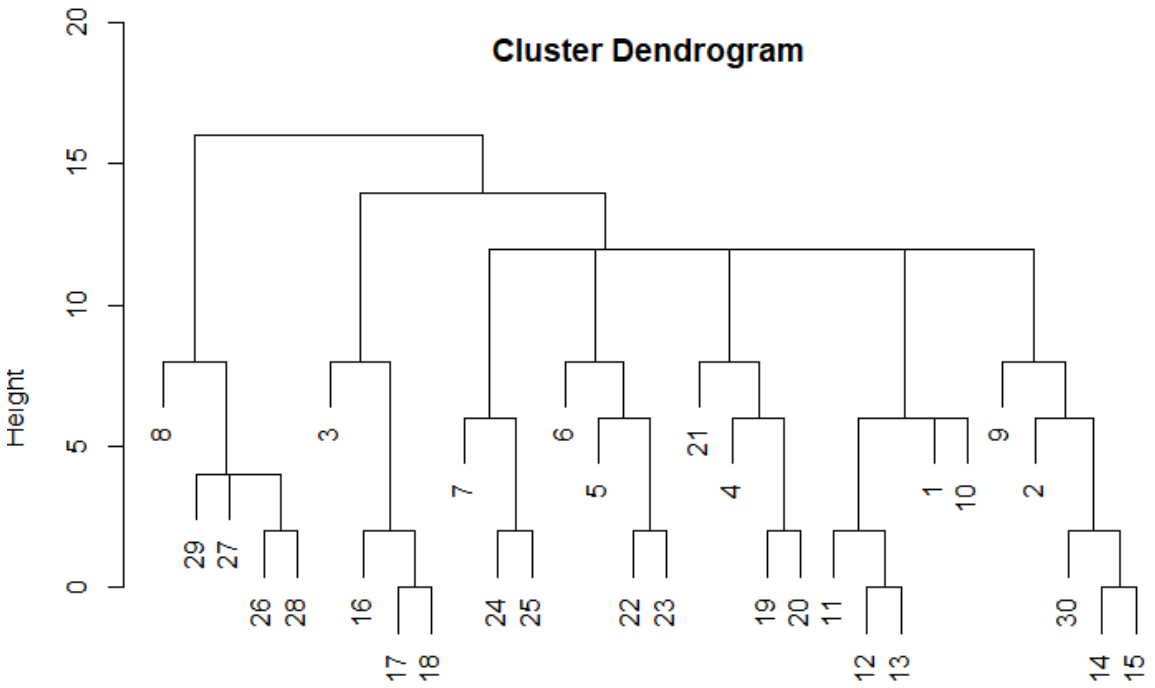
```
get.adjedgelist(ng,mode="in")

$Andy
+ 3/16 edges from bb1d6c8 (vertex names):
[1] Garth->Andy Carol->Andy Dan  ->Andy

$Garth
+ 1/16 edge from bb1d6c8 (vertex names):
[1] Andy->Garth

$Bill
+ 3/16 edges from bb1d6c8 (vertex names):
[1] Garth->Bill Carol->Bill Dan  ->Bill

$Elena
+ 3/16 edges from bb1d6c8 (vertex names):
[1] Bill ->Elena Frank->Elena Carol->Elena

$Frank
+ 1/16 edge from bb1d6c8 (vertex names):
[1] Elena->Frank

$Carol
+ 2/16 edges from bb1d6c8 (vertex names):
[1] Bill->Carol Dan ->Carol

$Dan
+ 3/16 edges from bb1d6c8 (vertex names):
[1] Andy ->Dan Bill ->Dan Carol->Dan
```

PRACTICAL-6

Aim: Write a program to exhibit structural equivalence, automorphic equivalence, and regular equivalence from a network.

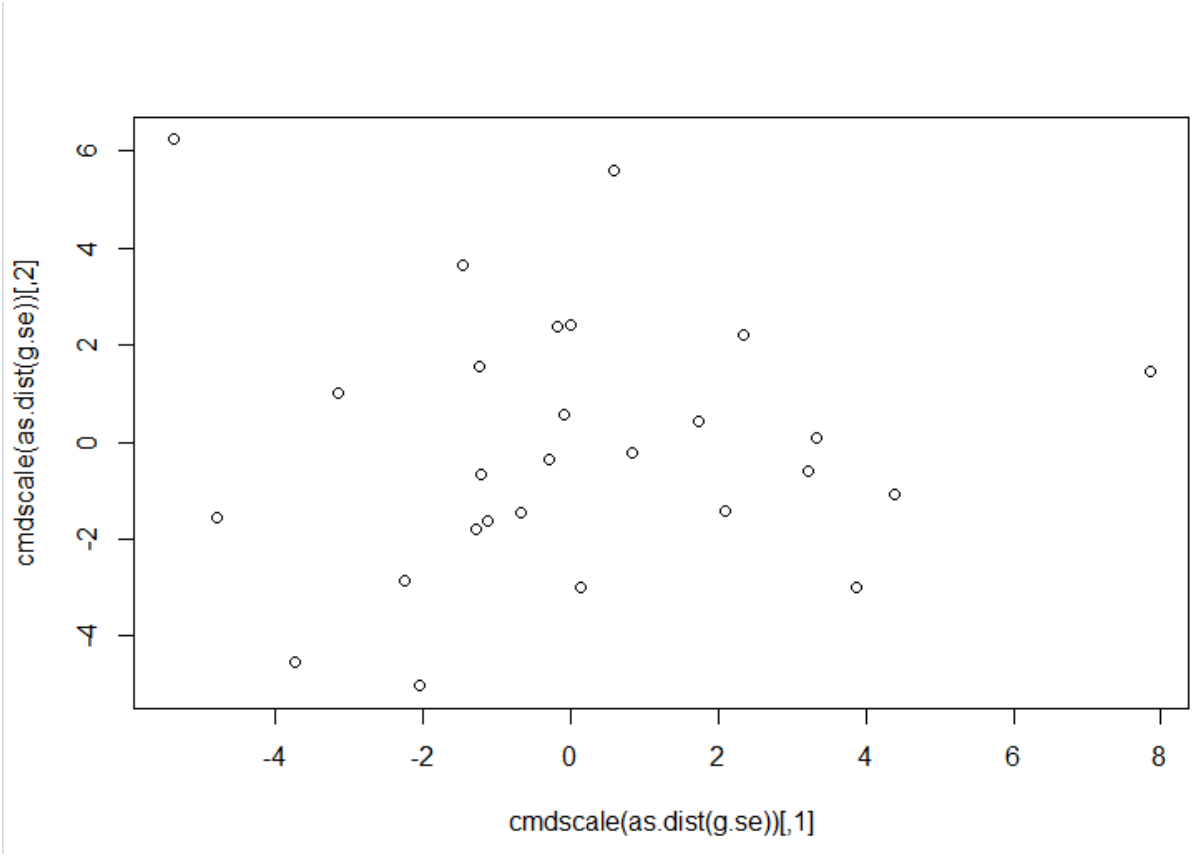
i) structural equivalence

```
library(sna)

library(igraph)

links2 <- read.csv("E:/Chinmay Classes Online/Masters_Computer_Science/SEM3(02-08-21)/Social_Network_Analysis/Practicals/Codes/PRACTICAL_6/edges1.csv", header=T, row.names=1)

eq<-equiv.clust(links2)

plot(eq)
```

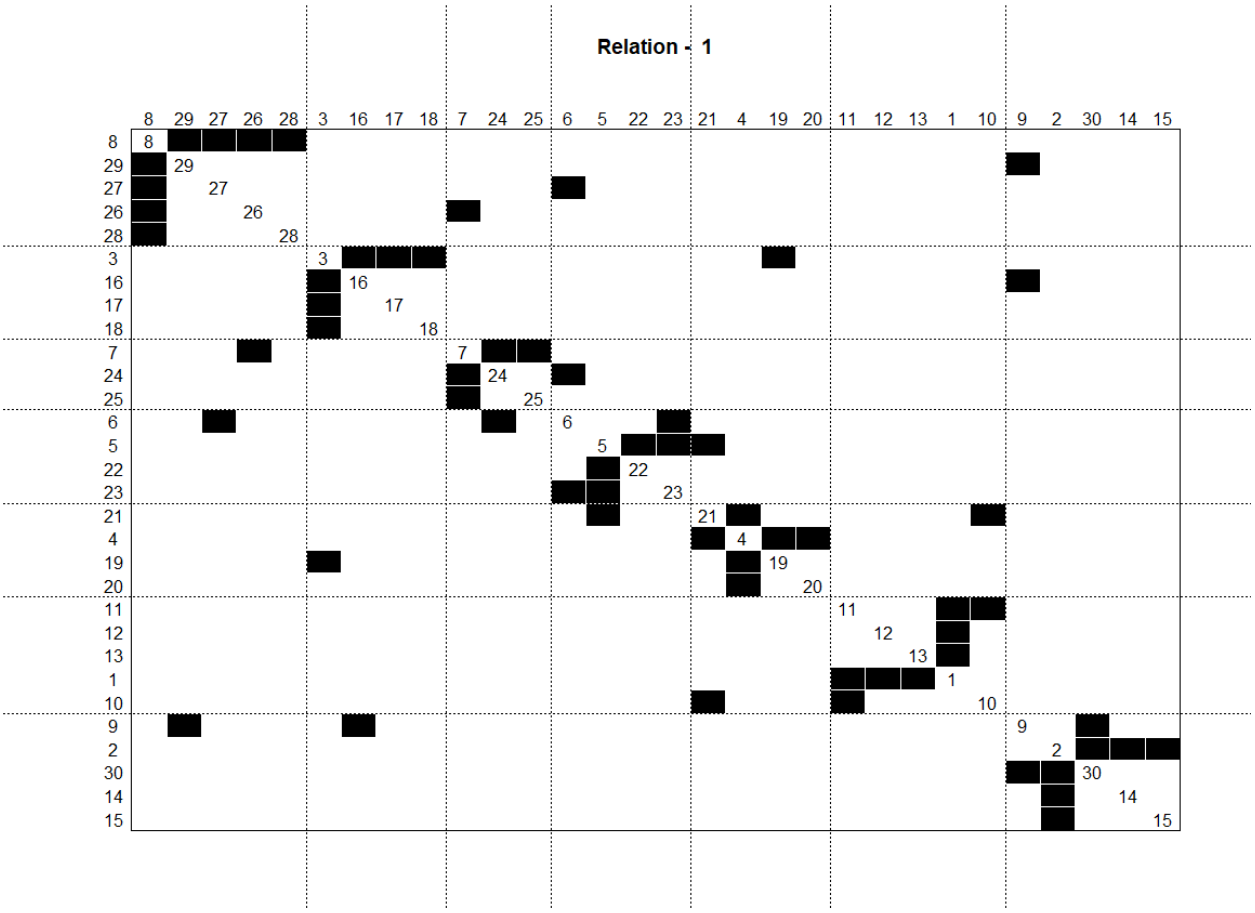


g.se<-sedist(links2)

plot(cmdscale(as.dist(g.se)))




```
b<-blockmodel(links2,eq,h=10)
plot(b)
```



PRACTICAL- 7

Aim: Perform SVD analysis of a network.

Software: RStudio

Theory

SVD Analysis

Singular value decomposition (SVD) is one method of identifying the factors underlying two-mode (valued) data. The method of extracting factors (singular values) differs somewhat from conventional factor and components analysis, so it is a good idea to examine both SVD and 2-mode factoring results.

The "singular values" are analogous to "eigenvalues" in the more common factor and components scaling techniques. The result here shows that the joint "space" of the variance among donors and initiatives is not well captured by an simple characterization. If we could easily make sense of the patterns with ideas like "left/right" and "financial/moral" as underlying dimensions, there would be only a few singular values that explained substantial portions of the joint variance. This result tells us that the ways that actors and events "go together" is not clean, simple, and easy.

Packages and Methods

Library()-load and attach add-on packages.

Igraph- igraph is a library and R package for network analysis. The main goals of the igraph library is to provide a set of data types and functions for 1) pain-free implementation of graph algorithms, 2) fast handling of large graphs, with millions of vertices and edges, 3) allowing rapid prototyping via high level languages like R.

Matrix- creates a matrix from the given set of values.

Svd()-Singular Value Decomposition of a Matrix. Compute the singular-value decomposition of a rectangular matrix.

Code:

```
library(igraph)

a<- matrix(c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0,

                0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1), 9, 4)

print(a)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    1    0    0
[2,]    1    1    0    0
[3,]    1    1    0    0
[4,]    1    0    1    0
[5,]    1    0    1    0
[6,]    1    0    1    0
[7,]    1    0    0    1
[8,]    1    0    0    1
[9,]    1    0    0    1
```

svd(a)

```
$d
[1] 3.464102e+00 1.732051e+00 1.732051e+00 1.922963e-16

$u
             [,1]       [,2]          [,3]          [,4]
 [1,] -0.3333333  0.4714045 -1.741269e-16  7.760882e-01
 [2,] -0.3333333  0.4714045 -3.692621e-16 -1.683504e-01
 [3,] -0.3333333  0.4714045 -5.301858e-17 -6.077378e-01
 [4,] -0.3333333 -0.2357023 -4.082483e-01  6.774193e-17
 [5,] -0.3333333 -0.2357023 -4.082483e-01  6.774193e-17
 [6,] -0.3333333 -0.2357023 -4.082483e-01  6.774193e-17
 [7,] -0.3333333 -0.2357023  4.082483e-01  5.194768e-17
 [8,] -0.3333333 -0.2357023  4.082483e-01  5.194768e-17
 [9,] -0.3333333 -0.2357023  4.082483e-01  5.194768e-17

$v
           [,1]       [,2]          [,3] [,4]
[1,] -0.8660254  0.0000000 -4.378026e-17  0.5
[2,] -0.2886751  0.8164966 -2.509507e-16 -0.5
[3,] -0.2886751 -0.4082483 -7.071068e-01 -0.5
[4,] -0.2886751 -0.4082483  7.071068e-01 -0.5
```