**Name: Stebin Puthur**

**Roll No: 09**

**Social Network Analysis**

**Index**

| **Sr. No.** | **Title** | **Page No.** |
| --- | --- | --- |
| 1 | Write a program to compute the following for a given a network: number of edges, number of nodes, degree of node, node with lowest degree, the adjacency list, matrix of the graph. | 2 |
| 2 | Perform following tasks: View data collection forms and/or import onemode/two-mode datasets, Basic Networks matrices transformations | 11 |
| 3 | Compute the following node level measures: Density; Degree; Reciprocity; Transitivity; Centralization; Clustering. | 15 |
| 4 | For a given network find the following: Length of the shortest path from a given node to another node; the density of the graph; Draw egocentric network of node G with chosen configuration parameters. | 22 |
| 5 | 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. | 29 |
| 6 | Write a program to exhibit structural equivalence, automatic equivalence, and regular equivalence from a network. | 33 |
| 7 | Perform SVD analysis of a network. | 38 |

# Practical 1

**Aim:** Write a program to compute the following for a given a network:

* number of edges
* number of nodes
* degree of node
* node with lowest degree  the adjacency list
* matrix of the graph.

**Software(s) used:**

* R ver. 4.1.3
* RStudio ver. 2022.02.0+433

**External packages required:**

* igraph

**Description:**

● **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:

* pain-free implementation of graph algorithms,
* fast handling of large graphs, with millions of vertices and edges, allowing rapid prototyping via high level languages like R.

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

●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.

●plot():Use to plot any graph.

●ecount():Returns the count of number of edges in graph ●vcount():Returns the count of number of vertices in graph

●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.

●V():

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

●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 indegree or “total” for the sum of the two. For undirected graphs this argument is ignored. “all” is a synonym of “total”.

●max() and min():

* Maxima and Minima
* Returns the (regular or parallel) maxima and minima of the input values.

●get.adjacency(): Convert a graph to an adjacency matrix

●get.adjlist():

* Adjacency lists
* Create adjacency lists from a graph, either for adjacent edges or for neighboring vertices

**Source Code:** library(igraph)

u\_graph <- graph.formula(A - B, A - C, A - D, B - C, B - F, C D, C - E, C - F, D - E, E - F, F - G, G - H)

d\_graph <- graph.formula(A <+ B, A <+ D, A -+ C, B -+ C, B -+

E, B -+ F, C -+ D, C -+ F, D -+ E)

e\_count(u\_graph) e\_count(d\_graph)

v\_count(u\_graph) v\_count(d\_graph)

E(u\_graph)

E(d\_graph)

V(u\_graph)

V(d\_graph)

degree(u\_graph) degree(u\_graph, mode = "in") degree(u\_graph, mode = "out")

degree(d\_graph) degree(d\_graph, mode = "in") degree(d\_graph, mode = "out")

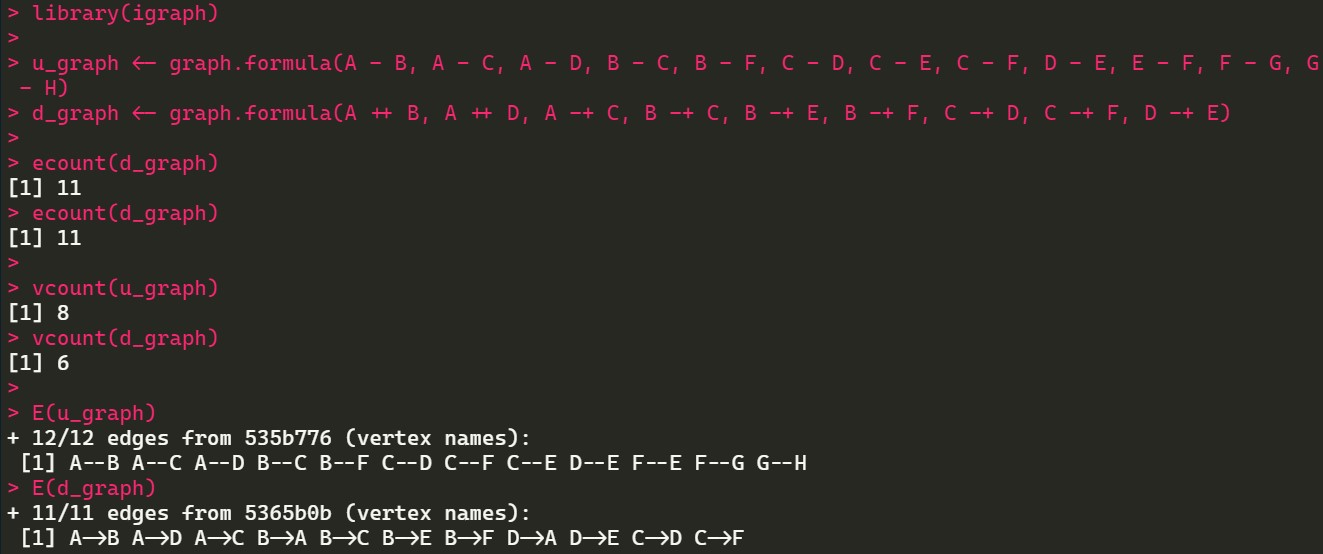
V(u\_graph)$name[degree(u\_graph) <= min(degree(u\_graph))] V(d\_graph)$name[degree(d\_graph, mode = "in") <= min(degree(d\_graph, mode = "in"))]

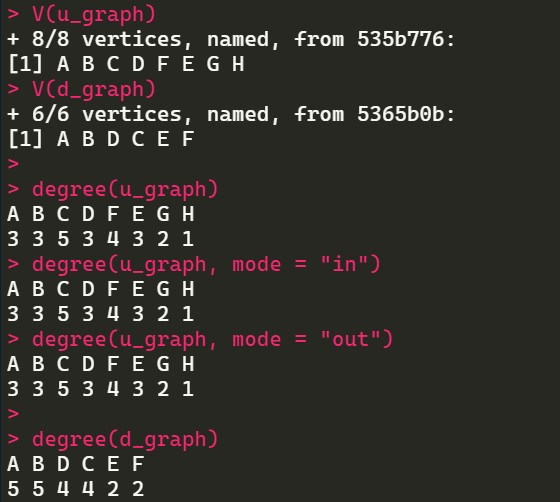
V(d\_graph)$name[degree(d\_graph, mode = "out") <= min(degree(d\_graph, mode = "out"))]

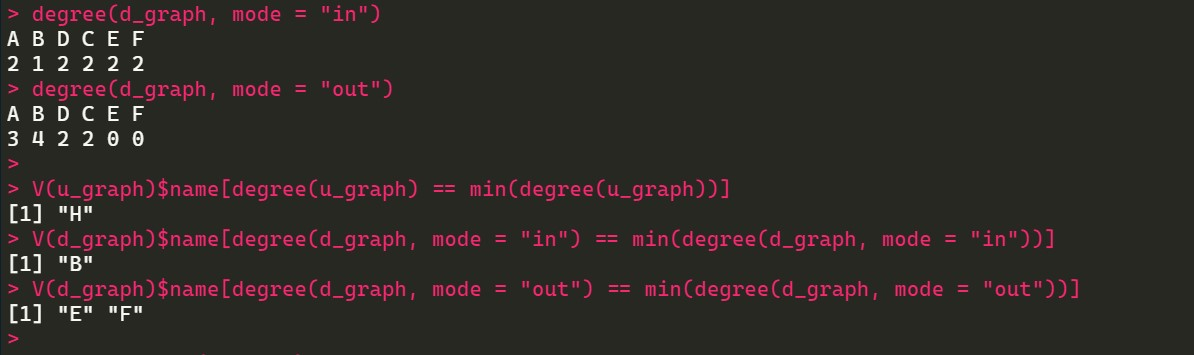
get.adjacency(u\_graph) get.adjacency(d\_graph)

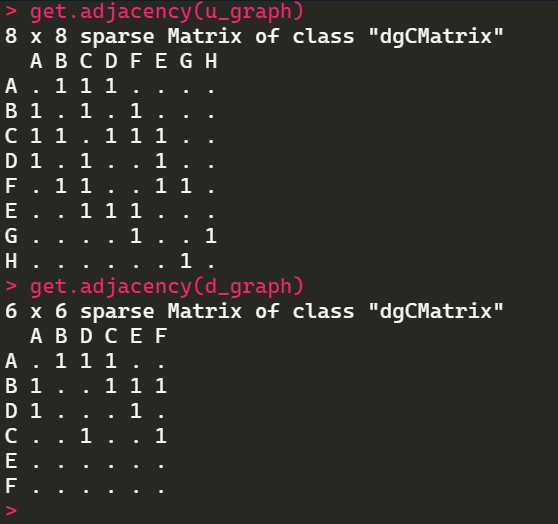
get.adjlist(u\_graph) get.adjlist(d\_graph)

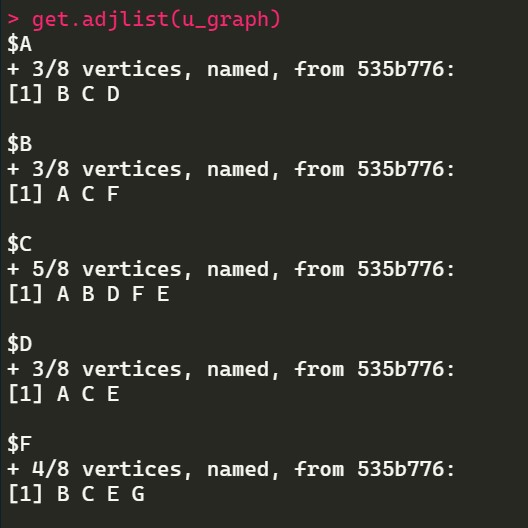
**Output:**



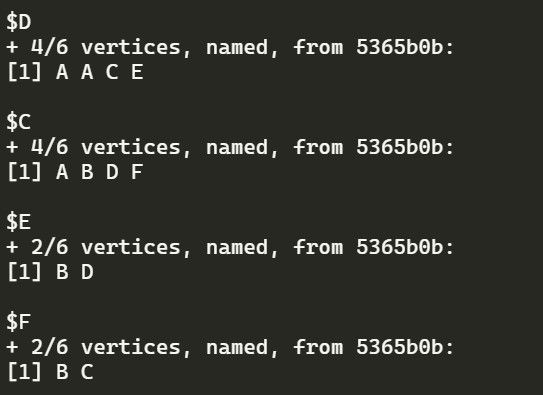












# Practical 2

**Aim:** Perform following tasks:

* View data collection forms and/or import onemode/two-mode datasets.
* Basic Networks matrices transformations.

**Software(s) used:**

* R ver. 4.1.3
* RStudio ver. 2022.02.0+433

**External packages required:**

* igraph

**Description:**

● **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:

* pain-free implementation of graph algorithms,
* fast handling of large graphs, with millions of vertices and edges, allowing rapid prototyping via high level languages like R.

●getwd(): Used to get the absolute filepath of the current R session.

●require(): library() and require() load and attach addon packages.

●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(): Returns the first part 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(): This function creates an igraph graph from one or two data frames containing the (symbolic) edge list and edge/vertex attributes.

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

●plot(): Draw a scatter plot with decorations such as axes and titles in the active graphics window.

**Source Code:**

require("igraph")

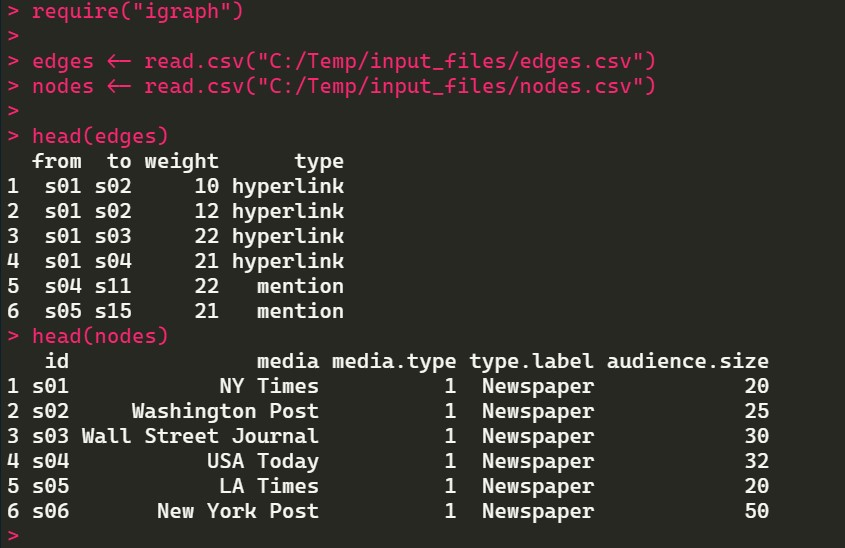
edges <- read.csv("C:/Temp/input\_files/edges.csv") nodes <- read.csv("C:/Temp/input\_files/nodes.csv")

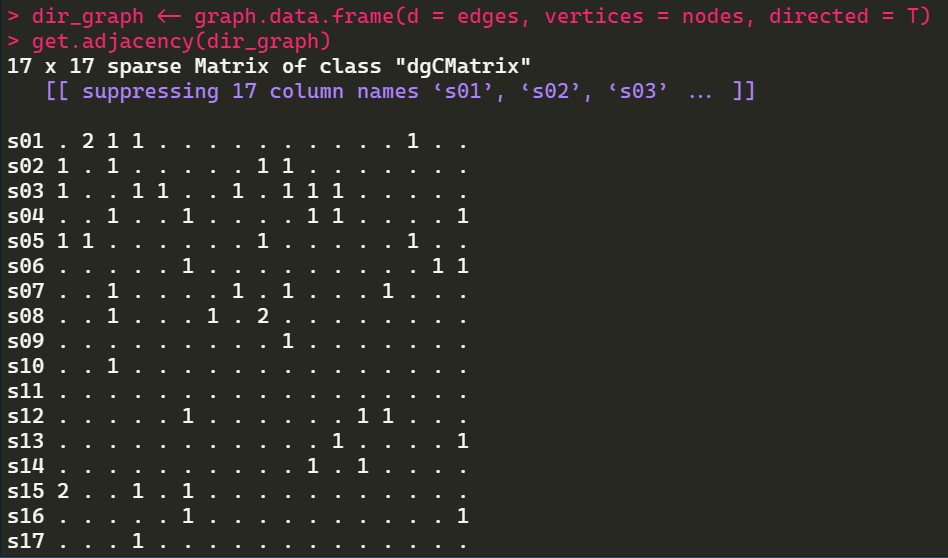
head(edges) head(nodes)

dir\_graph <- graph.data.frame(d = edges, vertices = nodes,

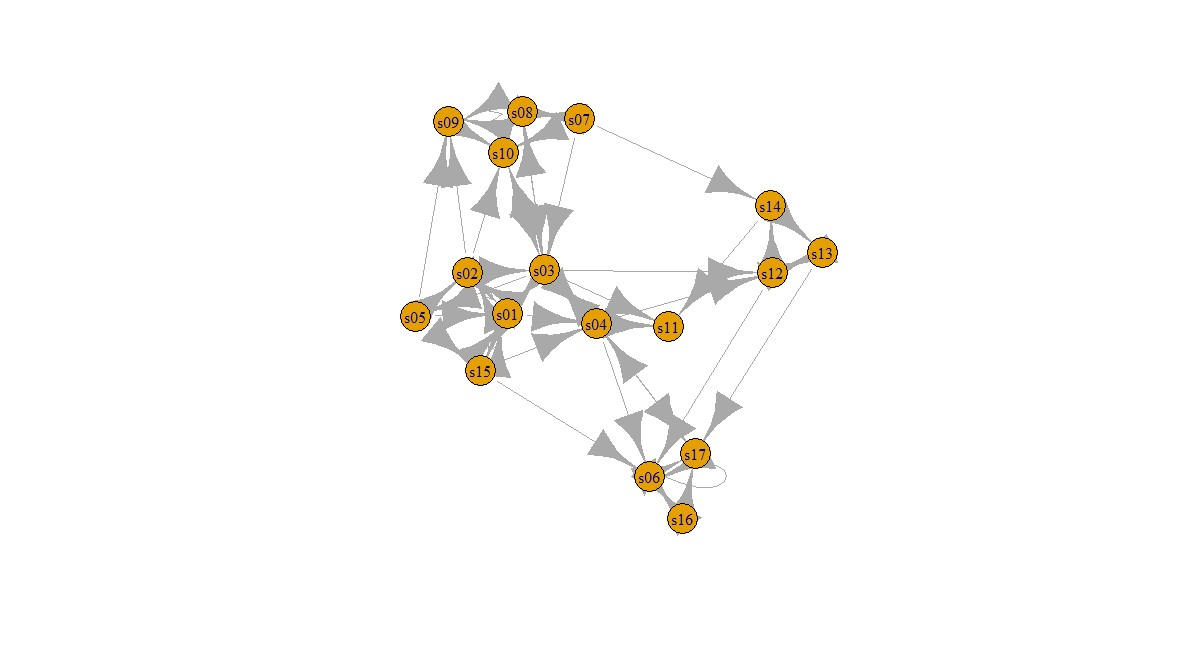
directed = T) get.adjacency(dir\_graph) plot(dir\_graph)

**Output:**









# Practical 3

**Aim:** Compute the following node level measures:

* Density
* Degree
* Reciprocity
* Transitivity
* Centralization  Clustering.

**Software(s) used:**

* R ver. 4.1.3
* RStudio ver. 2022.02.0+433

**External packages required:**

* igraph

**Description:**

● **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:

* pain-free implementation of graph algorithms,
* fast handling of large graphs, with millions of vertices and edges, allowing rapid prototyping via high level languages like R.

●library(): library() and require() load and attach addon packages.

●graph.famous(): Create an igraph graph from a list of edges, or a notable graph.

●ecount():Returns the count of number of edges in graph

●vcount():Returns the count of number of vertices in graph

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

●plot(): Draw a scatter plot with decorations such as axes and titles in the active graphics window.

●reciprocity(): Calculates the reciprocity of a directed graph.

●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.

●adjacent.triangles(): Count how many triangles a vertex is part of, in a graph, or just list the triangles of a graph.

●transitivity(): Transitivity measures the probability that the adjacent vertices of a vertex are connected. This is sometimes also called the clustering coefficient.

●degree(): The degree of a vertex is its most basic structural property, the number of its adjacent edges. ●barabasi.game(): The BA-model is a very simple stochastic algorithm for building a graph.

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

●graph.union(): The union of two or more graphs are created. The graphs may have identical or overlapping vertex sets.

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

**Source Code:** library("igraph")

kite <- graph.famous("Krackhardt\_Kite") vcount(kite) ecount(kite)

ecount(kite) / (vcount(kite) \* (vcount(kite) - 1) / 2)

dir\_graph <- graph.formula(A <+ B, A <+ D, A -+ C, B -+ C, B -+ E, B -+ F, C -+ D, C -+ F, D -+ E) plot(dir\_graph) reciprocity(dir\_graph) dyad.census(dir\_graph)

mutual <- dyad.census(dir\_graph)$mut

mutual / (ecount(dir\_graph))

atri <- adjacent.triangles(kite) plot(kite, vertex.label = atri) transitivity(kite, type = "local")

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

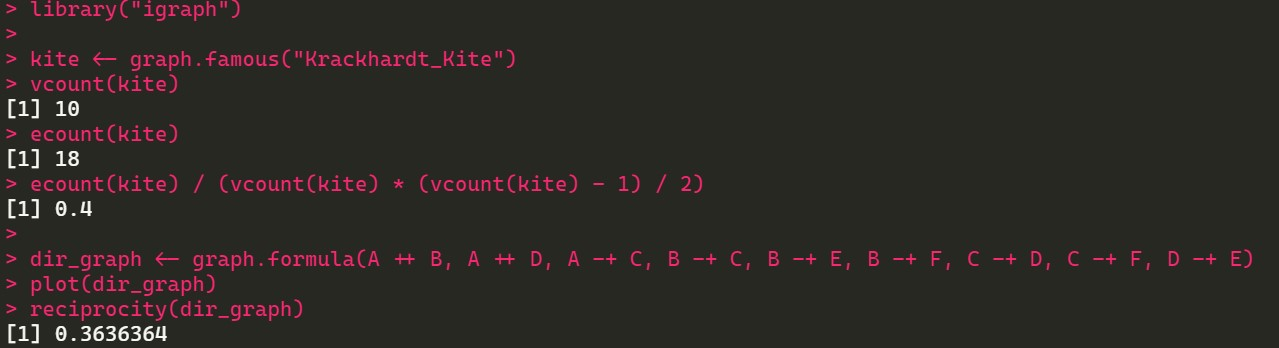
2)

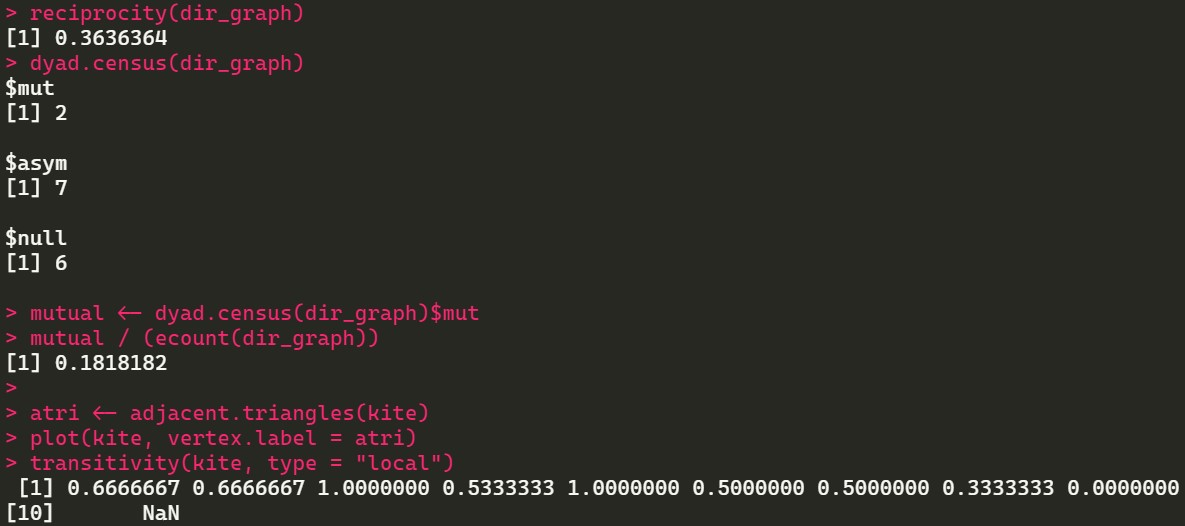
graph\_2 <- barabasi.game(50, p = 2, directed = F)

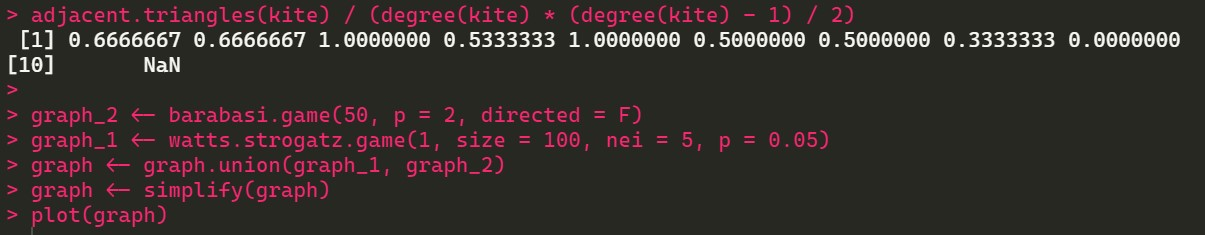
graph\_1 <- watts.strogatz.game(1, size = 100, nei = 5, p =

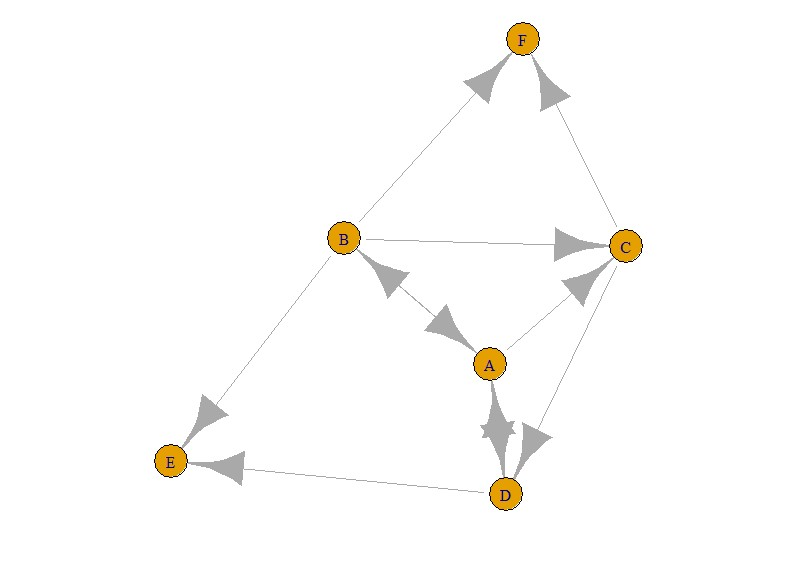
0.05)

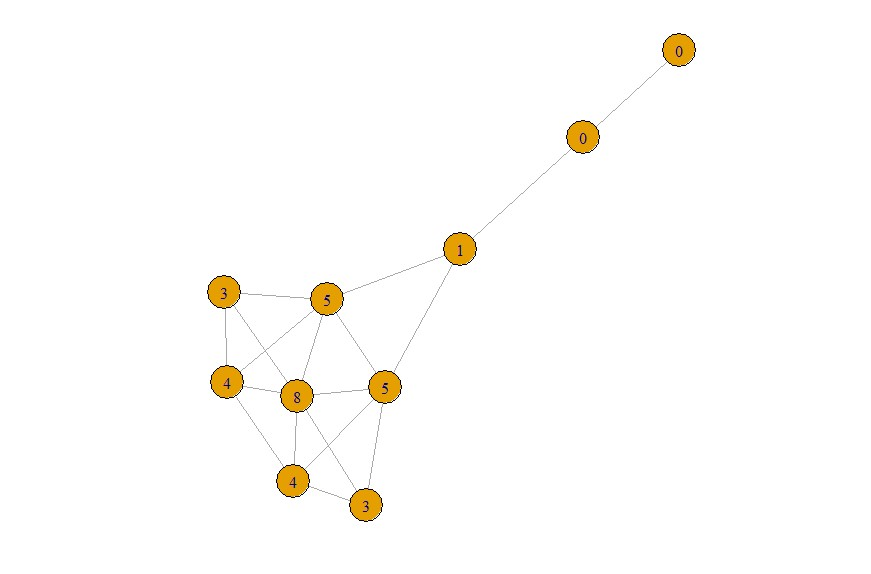
graph <- graph.union(graph\_1, graph\_2) graph <- simplify(graph) plot(graph) **Output:**

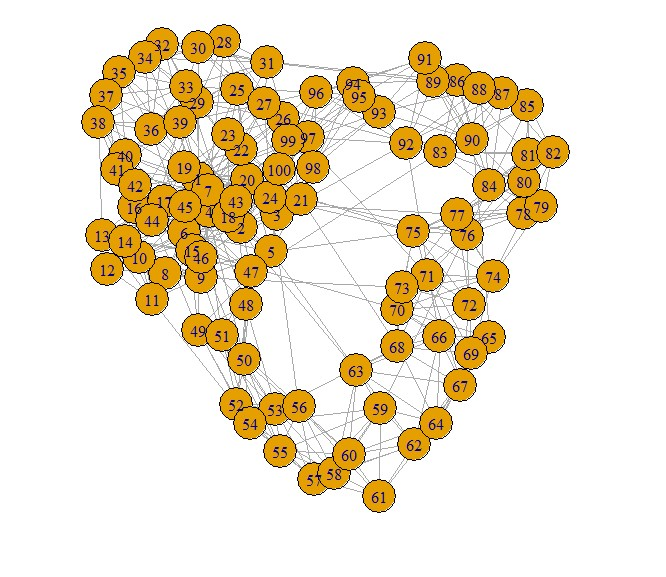












# Practical 4

**Aim:** For a given network find the following:

* Length of the shortest path from a given node to another node
* The density of the graph
* Draw egocentric network of node G with chosen configuration parameters.

**Software(s) used:**

* R ver. 4.1.3
* RStudio ver. 2022.02.0+433

**External packages required:**

* igraph

**Description:**

● **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:

* pain-free implementation of graph algorithms,
* fast handling of large graphs, with millions of vertices and edges, allowing rapid prototyping via high level languages like R.

●library(): library() and require() load and attach addon packages.

●as.matrix(): matrix() creates a matrix from the given set of values. 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.

●colnames() and rownames(): Retrieve or set the row or column names of a matrix-like object.

●is.na(): The generic function is.na() indicates which elements are missing.

●graph.adjacency(): graph\_from\_adjacency\_matrix() is a flexible function for creating igraph graphs from adjacency matrices.

●plot(): Draw a scatter plot with decorations such as axes and titles in the active graphics window.

●shortest.paths(): shortest\_paths() calculates one

shortest path (the path itself, and not just its length) from or to the given vertex.

●print(): print() prints its argument and returns it invisibly.

●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.

**Source Code:** library(igraph)

matt <- as.matrix(read.table(text=

"node R S T U

1. 7 5 0 0
2. 7 0 0 2
3. 0 6 0 0
4. 4 0 1 0", header=T))nms <- matt[, 1] matt <- matt[, -1]

colnames(matt) <- rownames(matt) <- nms matt[is.na(matt)] <- 0

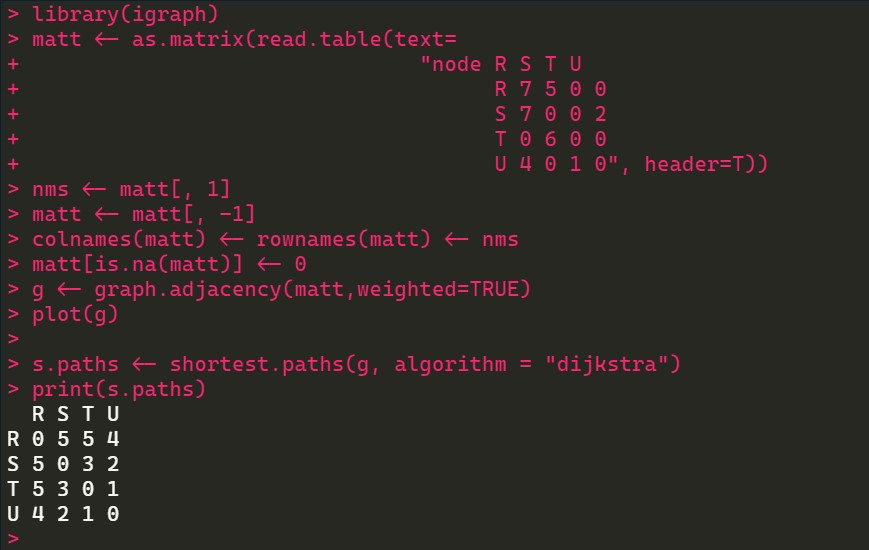
g <- graph.adjacency(matt,weighted=TRUE) plot(g)

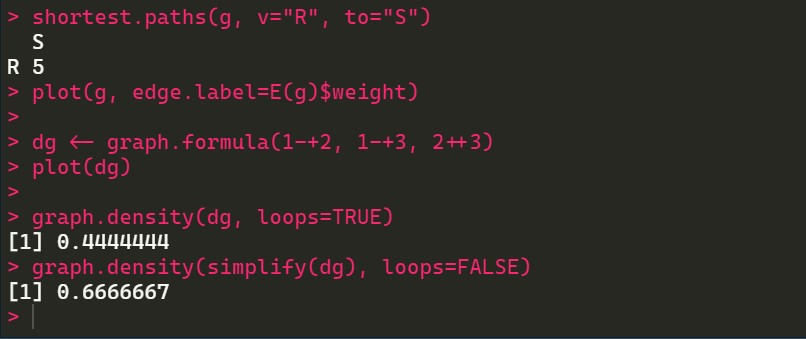
s.paths <- shortest.paths(g, algorithm = "dijkstra") print(s.paths)

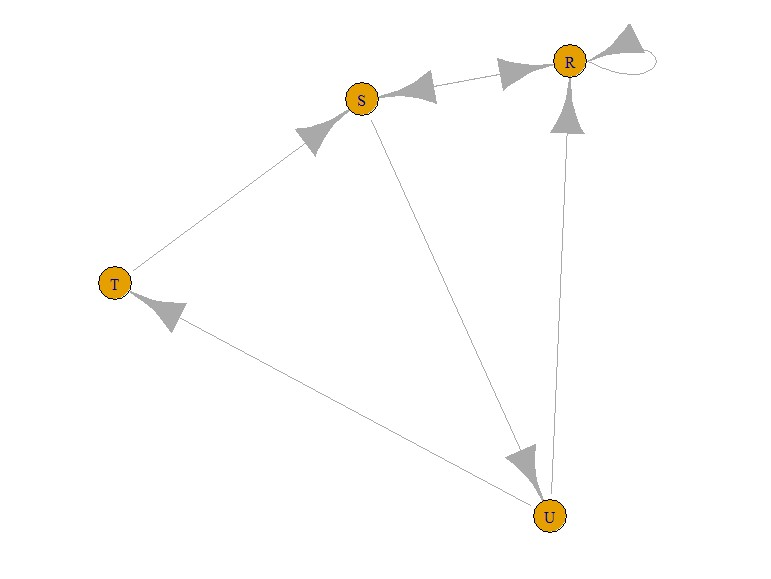
shortest.paths(g, v="R", to="S") plot(g, edge.label=E(g)$weight) dg <- graph.formula(1-+2, 1-+3, 2<+3) plot(dg)

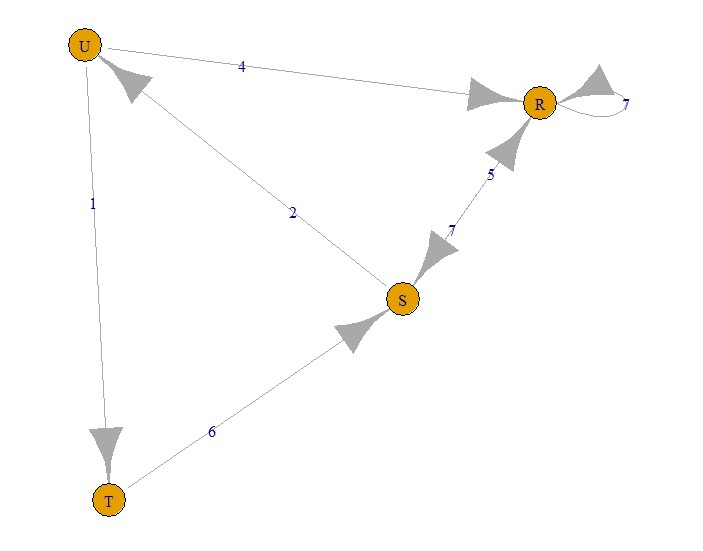
graph.density(dg, loops=TRUE) graph.density(simplify(dg), loops=FALSE)

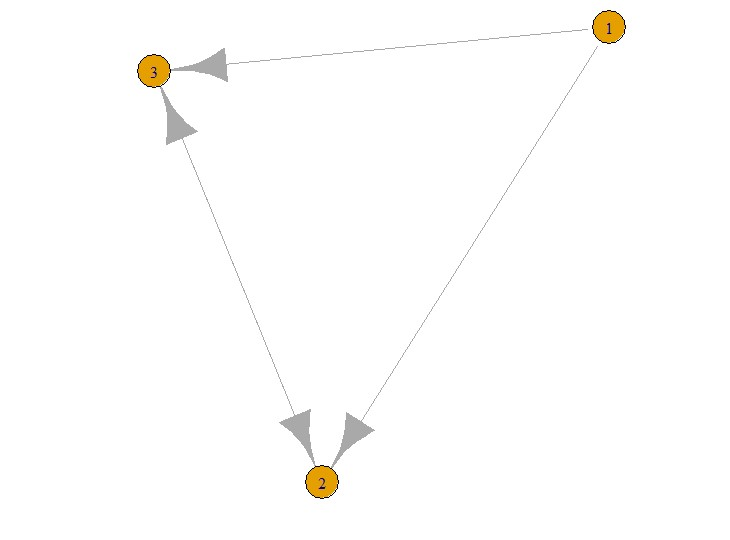
**Output:**











# 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(s) used:**

* R ver. 4.1.3
* RStudio ver. 2022.02.0+433

**External packages required:**

* igraph

**Description:**

● **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:

* pain-free implementation of graph algorithms,
* fast handling of large graphs, with millions of vertices and edges, allowing rapid prototyping via high level languages like R.

●library(): library() and require() load and attach addon packages.

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

●plot(): Draw a scatter plot with decorations such as axes and titles in the active graphics window.

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

●E(): 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.

**Source Code:** library(igraph)

sociogram <- graph.formula(Andy<+Garth,Garth-+Bill,Bill-

+Elena,Elena<+Frank,Carol-

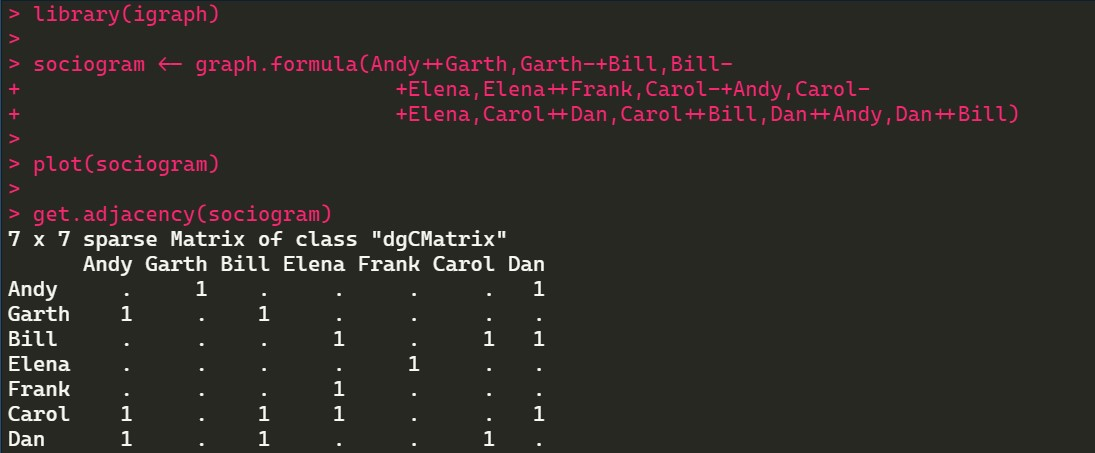
+Andy,Carol-+Elena,Carol+

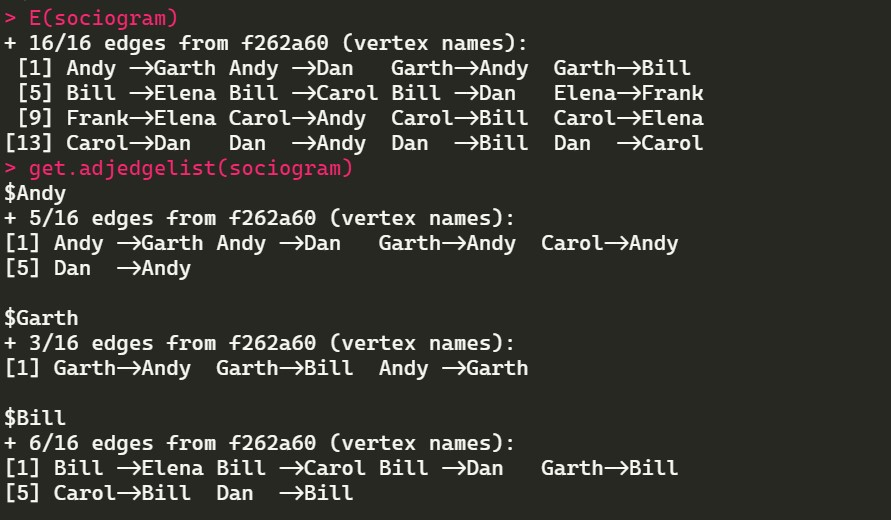
+Dan,Carol<+Bill,Dan<+Andy, Dan<+Bill)

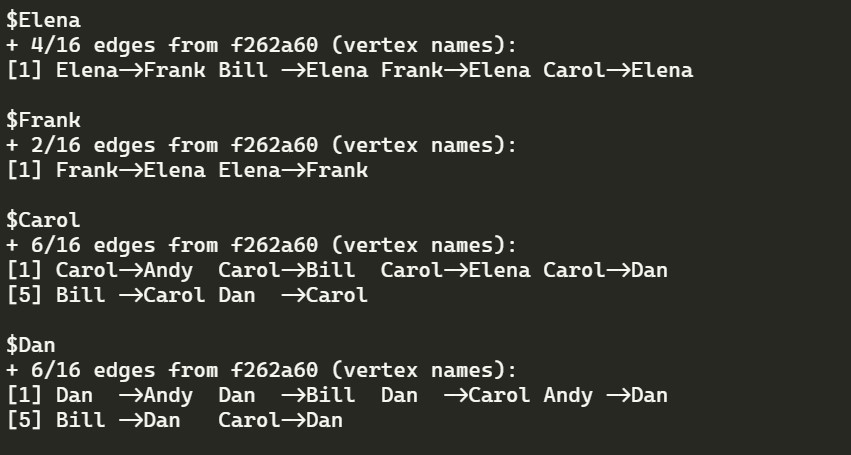
plot(sociogram)

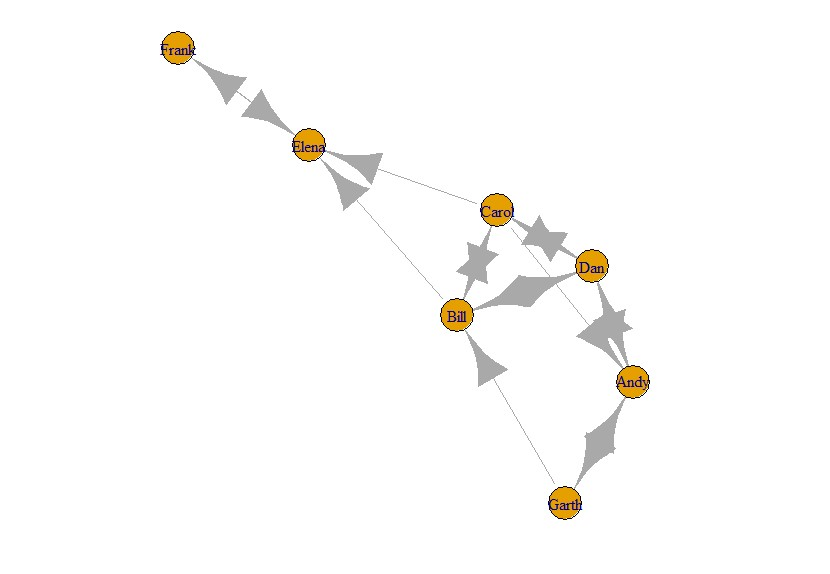
get.adjacency(sociogram) E(sociogram) get.adjedgelist(sociogram)

**Output:**









# Practical 6

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

**Software(s) used:**

* R ver. 4.1.3
* RStudio ver. 2022.02.0+433

**External packages required:**

* igraph
* sna

**Description:**

* + **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:

* pain-free implementation of graph algorithms,
* fast handling of large graphs, with millions of vertices and edges, allowing rapid prototyping via high level languages like R.
  + **The** sna **package**: sna is a package containing a range of tools for social network analysis. Supported functionality includes node and graph-level indices, structural distance and covariance methods, structural equivalence detection, p\* modeling, random graph generation, and 2D/3D network visualization (among other things).

●library(): library() and require() load and attach addon packages.

●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.

●equiv.clust(): equiv.clust() uses a definition of approximate equivalence (equiv.fun()) to form a hierarchical clustering of network positions.

●plot(): Draw a scatter plot with decorations such as axes and titles in the active graphics window.

●sedist(): sedist() uses the graphs indicated by g in the arguments to assess the extent to which each vertex is structurally equivalent.

●cmdscale(): Classical multidimensional scaling (MDS) of a data matrix.

●as.dist(): This function computes and returns the distance matrix computed by using the specified distance measure to compute the distances between the rows of a data matrix.

●blockmodel(): Given a set of equivalence classes and one or more graphs, blockmodel will form a blockmodel of the input graph(s) based on the classes in question

**Source Code:** library(sna) library(igraph)

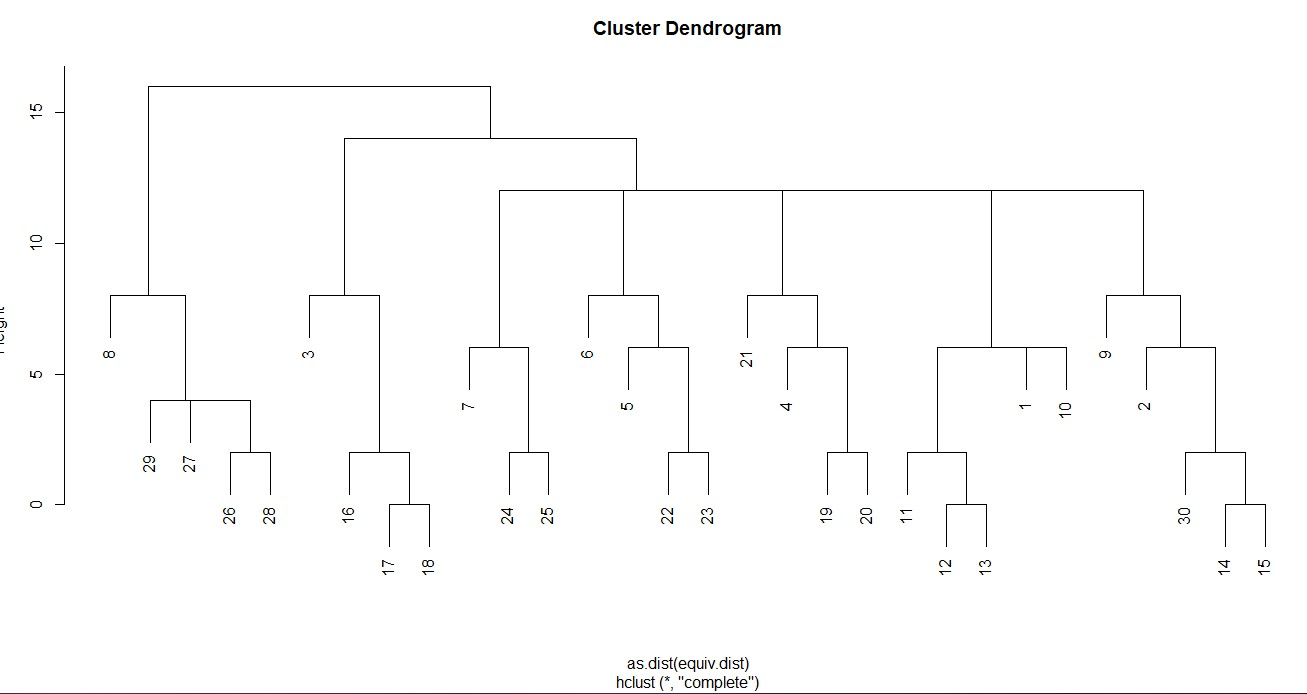
links2 <- read.csv("C:/Temp/input\_files/edges1.csv", header=T, row.names=1) eq<-equiv.clust(links2) plot(eq)

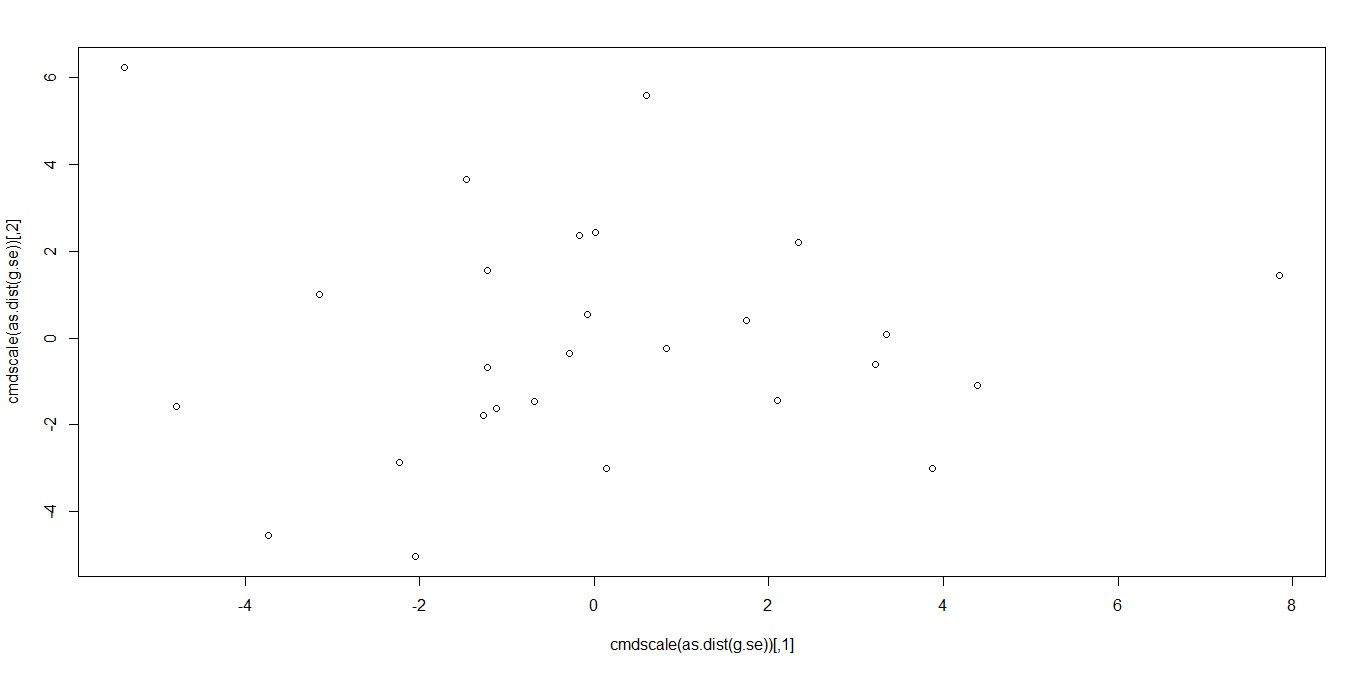
g.se<-sedist(links2)

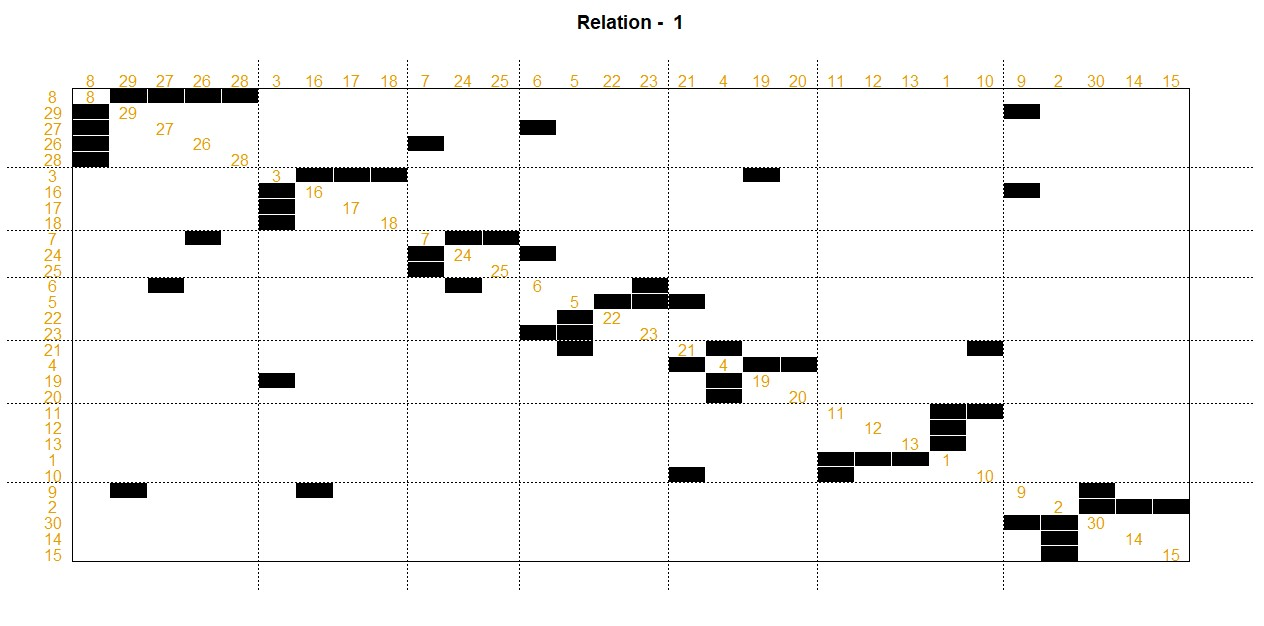
plot(cmdscale(as.dist(g.se))) b<-blockmodel(links2,eq,h=10) plot(b)

**Output:**









# Practical 7

**Aim:**  Perform SVD analysis of a network.

**Software(s) used:**

* R ver. 4.1.3
* RStudio ver. 2022.02.0+433

**External packages required:**

* igraph

**Description:**

● **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:

* pain-free implementation of graph algorithms,
* fast handling of large graphs, with millions of vertices and edges, allowing rapid prototyping via high level languages like R.

●matrix(): matrix() creates a matrix from the given set of values.

●c(): Combines values into a vector or list.

●print(): print() prints its argument and returns it invisibly. ●svd(): Compute the singular-value decomposition of a rectangular matrix.

**Source 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) svd(a)

**Output:**

