**NAME: ASIF ERFAN KHAN**

**ROLL NUMBER: 546**

**COURSE: MSc CS**

**SUBJECT: SOCIAL NETWORK ANALYSIS**

**PRACTICAL: 1-8**

|  |  |  |  |
| --- | --- | --- | --- |
| INDEX | | | |
|  | DATE | TITLE | SIGN |
| 1 |  | **Write a program to compute the following for a given**  **a network: (i) number of edges, (ii) number of nodes; (iii) degree of node; (iv) node with lowest degree; (v) the adjacency list; (vi) matrix of the graph.** |  |
| 2 |  | **Perform following tasks: (i) View data collection forms**  **and/or import onemode/two-mode datasets; (ii) Basic**  **Networks matrices transformations** |  |
| 3 |  | **Compute the following node level measures: (i) Density; (ii) Degree; (iii) Reciprocity; (iv) Transitivity; (v) Centralization; (vi) Clustering.** |  |
| 4 |  | **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** |  |
| 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.** |  |
| 6 |  | **Write a program to exhibit structural equivalence,**  **automorphic equivalence, and regular equivalence from a network.** |  |
| 7 |  | **Create sociograms for the persons-by-persons** |  |
|  | **network and the committee-bycommittee network**  **for a given relevant problem. Create one-mode**  **network and two-node network for the same** |  |
| 8 |  | **Perform SVD analysis of a network.** |  |

**Practical 1: Write a program to compute the following for a given a network: (i) number of edges, (ii) number of nodes; (iii) degree of node; (iv) node with lowest degree; (v) the adjacency list; (vi) matrix of the graph.**

Code:

# Install and load the igraph package

install.packages("igraph")

library(igraph)

# Create a graph object 'g' using graph.formula function with edges 1-2, 1-3, 2-3, 2-4, 3-5, 4-5, 4-6, 4-7, 5-6, 6-7

g <- graph.formula(1-2, 1-3, 2-3, 2-4, 3-5, 4-5, 4-6, 4-7, 5-6, 6-7)

# Plot the graph object 'g'

plot(g)

# Count the number of edges in 'g'

ecount(g)

# Count the number of vertices in 'g'

vcount(g)

# Calculate the degree of each vertex in 'g'

degree(g)

# Create another graph object 'dg' using graph.formula function with edges 1->2, 1->3, 2<-3

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

# Plot the graph object 'dg'

plot(dg)

# Calculate the in-degree of each vertex in 'dg'

degree(dg, mode="in")

# Calculate the out-degree of each vertex in 'dg'

degree(dg, mode="out")

# Print the name of the vertex with the minimum degree in 'dg'

V(dg)$name[degree(dg)==min(degree(dg))]

# Print the name of the vertex with the maximum degree in 'dg'

V(dg)$name[degree(dg)==max(degree(dg))]

# Find the neighbors of vertex 5 in 'g'

neighbors(g,5)

# Find the neighbors of vertex 2 in 'g'

neighbors(g,2)

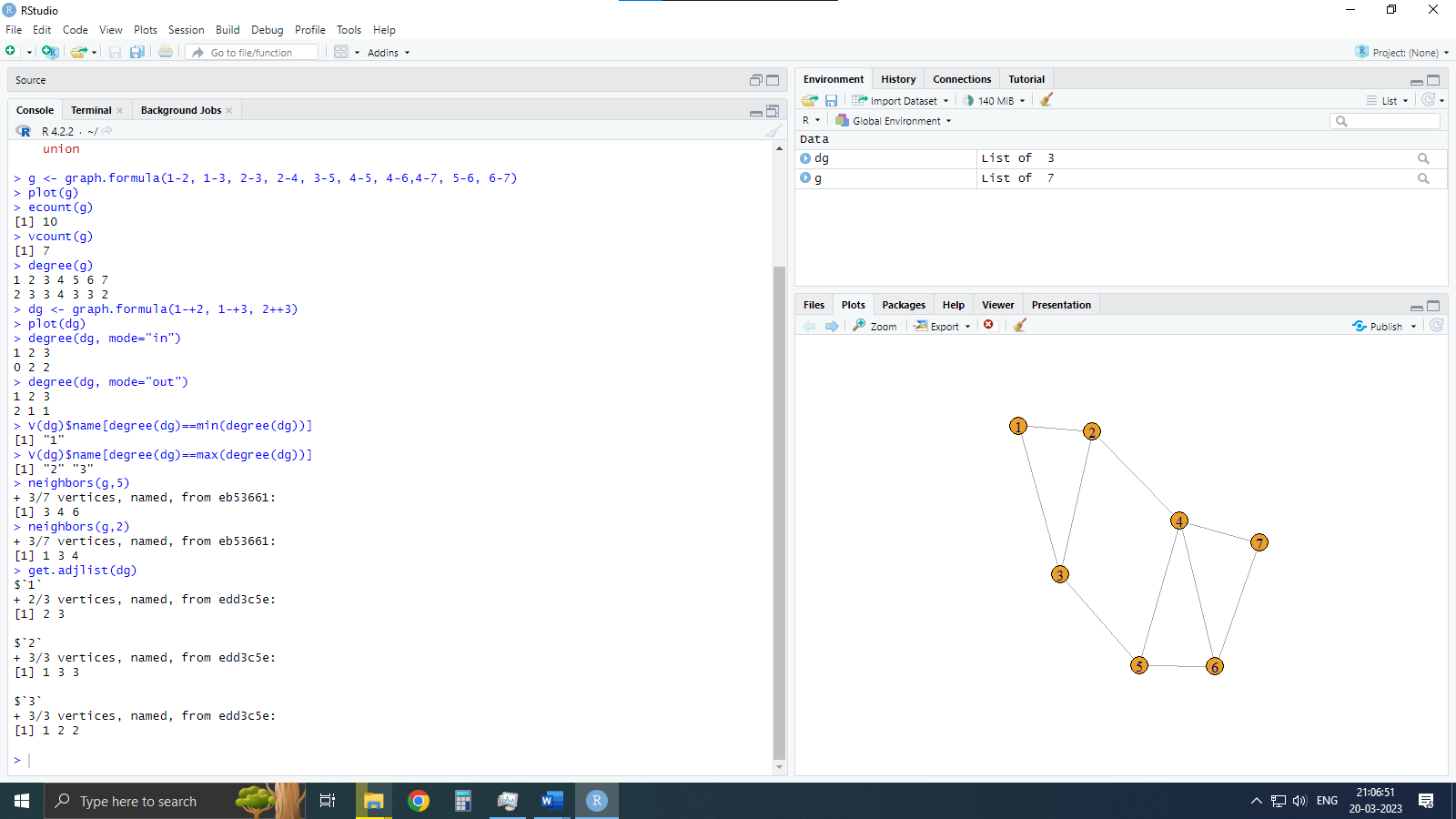
# Get the adjacency list of 'dg'

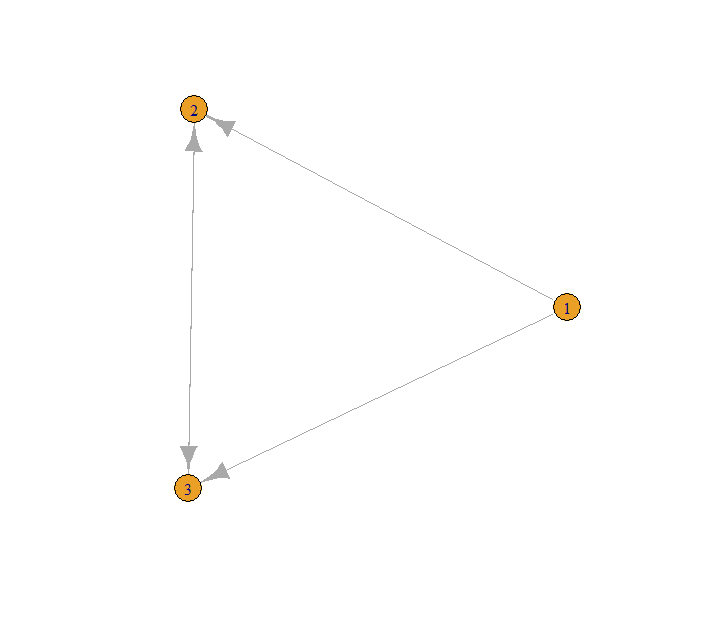
get.adjlist(dg)

# Get the adjacency matrix of 'g'

get.adjacency(g)

**OUTPUT**





**Practical 2: Perform following tasks: (i) View data collection forms and/or import onemode/two-mode datasets; (ii) Basic Networks matrices transformations**

Code:

# Get the current working directory

getwd()

# Set the working directory to "d:/SNA\_pract"

setwd("d:/SNA\_pract")

# Read the nodes.csv file into a data frame 'nodes'

nodes <- read.csv("nodes.csv", header = T, , as.is = T)

# Print the first few rows of 'nodes'

head(nodes)

# Read the edges.csv file into a data frame 'links'

links <- read.csv("edges.csv", header = T, as.is = T)

# Print the first few rows of 'links'

head(links)

# Create a graph object 'net' from the data frames 'nodes' and 'links'

net <- graph.data.frame(d = links,

                        vertices = nodes,

                        directed = T)

# Convert the graph object 'net' to an adjacency matrix 'm'

m = as.matrix(net)

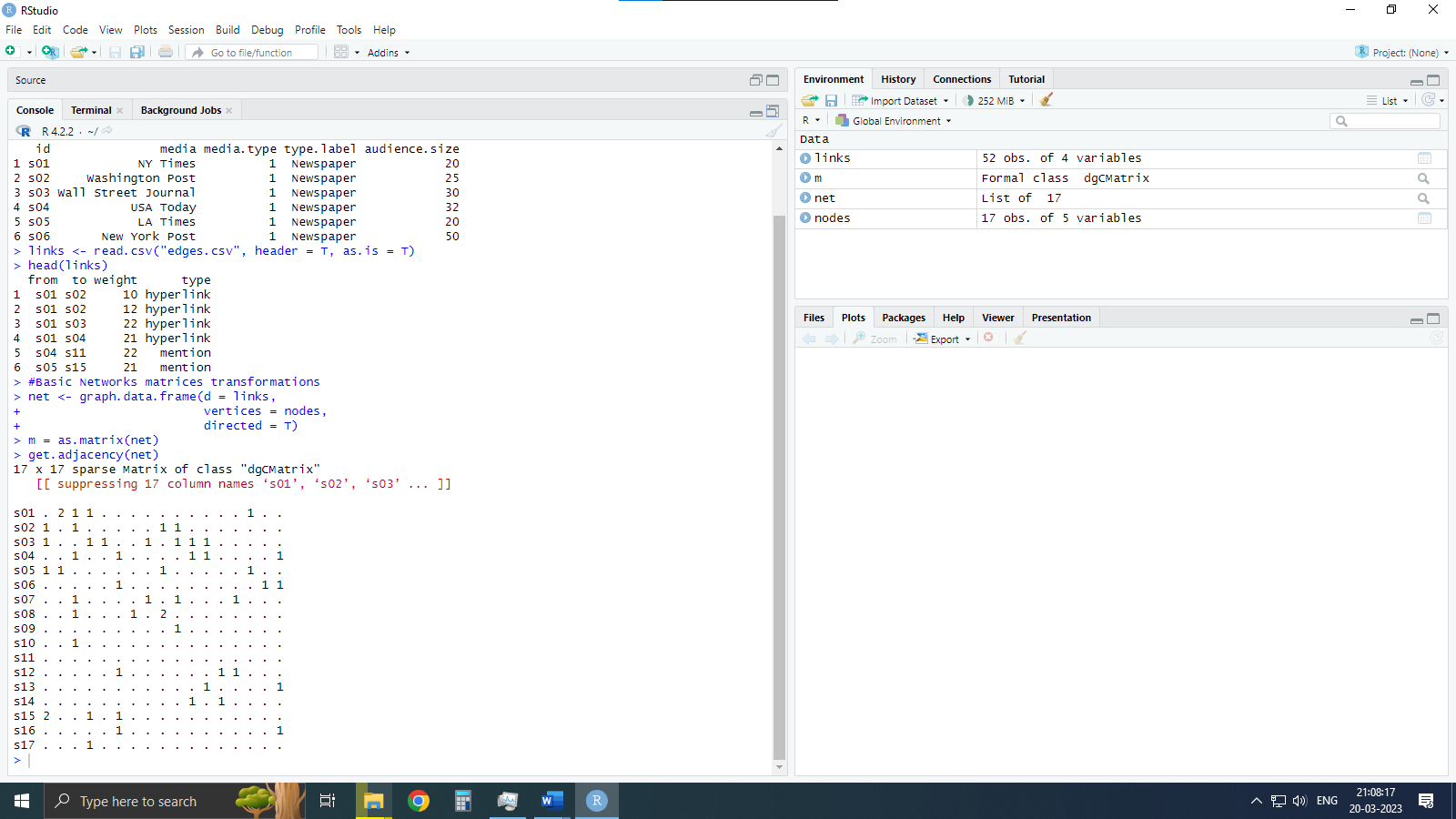
# Print the adjacency matrix of 'net'

get.adjacency(net)

# Plot the graph object 'net'

plot(net)

**OUTPUT**



**Practical 3: Compute the following node level measures: (i) Density; (ii) Degree; (iii) Reciprocity; (iv) Transitivity; (v) Centralization; (vi) Clustering.**

Code:

# Load the igraph library

library(igraph)

# Create a graph object 'g'

g <- graph.formula(1-2, 1-3, 2-3, 2-4, 3-5, 4-5, 4-6,4-7, 5-6, 6-7)

# Density

# Number of vertices

vcount(g)

# Number of edges

ecount(g)

# Density of the graph

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

# Degree

degree(g)

# Reciprocity:

# Create a directed graph 'dg'

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

# Plot the directed graph 'dg'

plot(dg)

# Reciprocity of the directed graph 'dg'

reciprocity(dg)

# Formula for reciprocity

(2 \* dyad.census(dg)$mut / ecount(dg))

# Transitivity

# Create a famous graph 'kite'

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

# Find the adjacent triangles in the 'kite' graph

atri <- adjacent.triangles(kite)

# Plot the 'kite' graph with vertex labels as adjacent triangles

plot(kite, vertex.label = atri)

# Local transitivity of the directed graph 'dg'

transitivity(dg, type = "local")

# Proportion of adjacent triangles to all possible triangles in the 'kite' graph

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

# Centralization

# Degree of centrality

centralization.degree(g, mode = "in", normalized = T)

# Closeness Centralization

closeness(g)

centralization.closeness(g, mode = "all", normalized = TRUE)

# Betweeness Centrality

betweenness(g, directed = T, weights = NA)

edge.betweenness(g, directed = T, weights = NA)

centralization.betweenness(g, directed = T, normalized = T)

# Eigenvector centrality

centralization.evcent(g, directed = T, normalized = T)

# Clustering

# Create two graphs 'g1' and 'g2'

g2 <- barabasi.game(50, p = 2, directed = F)

g1 <- watts.strogatz.game(1, size = 100, nei = 5, p = 0.05)

# Combine the two graphs 'g1' and 'g2'

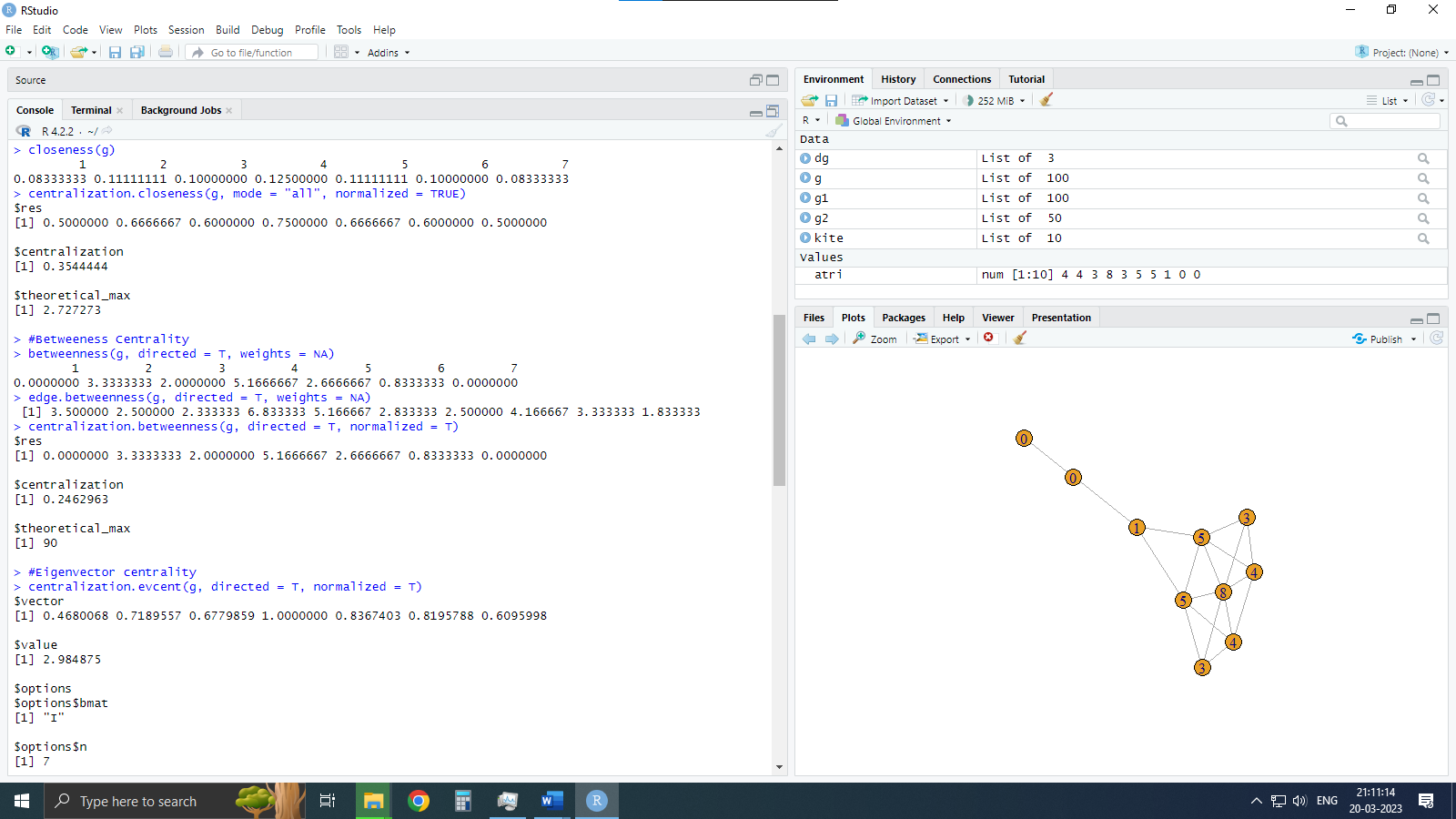
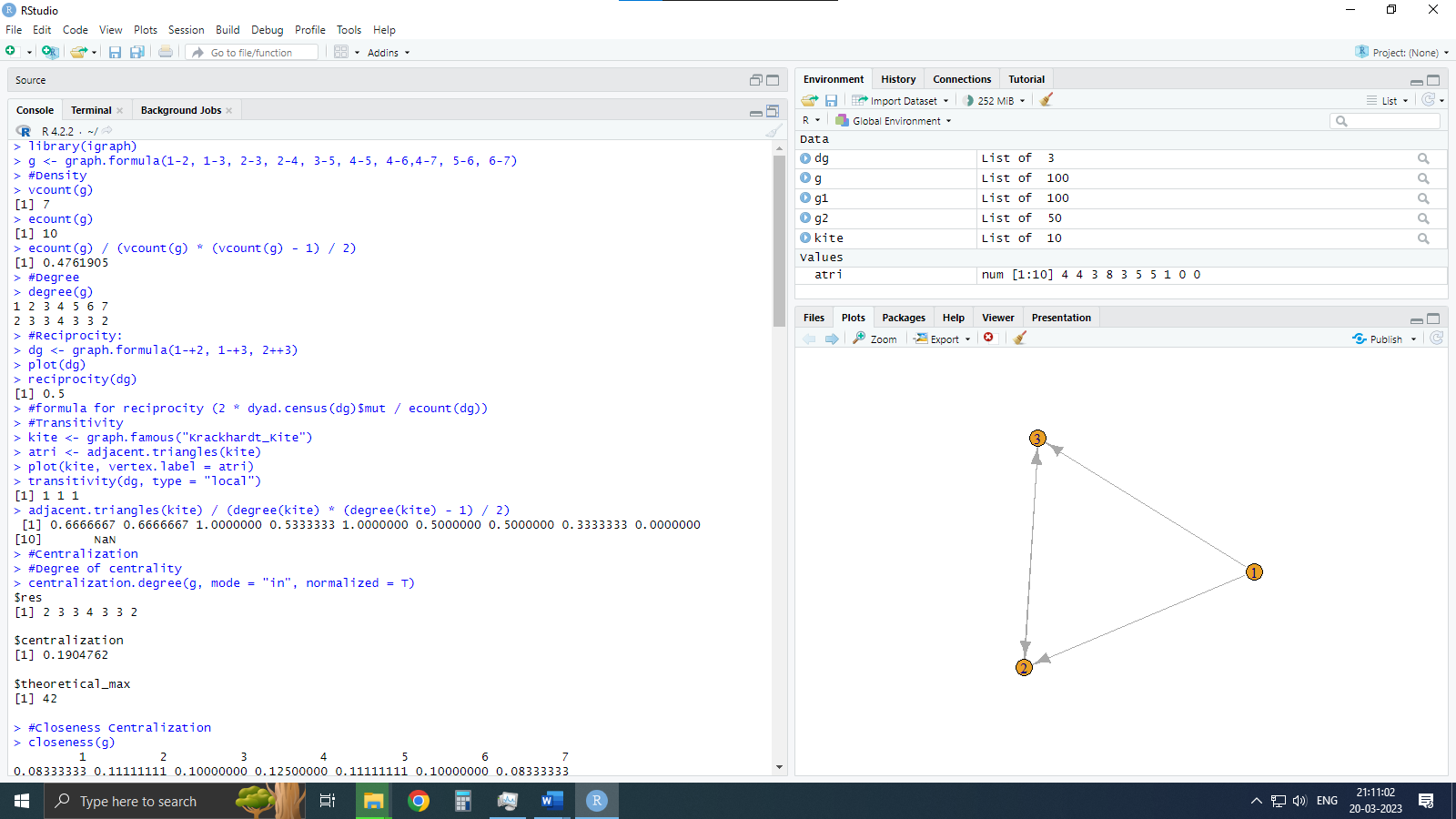
g <- graph.union(g1, g2)

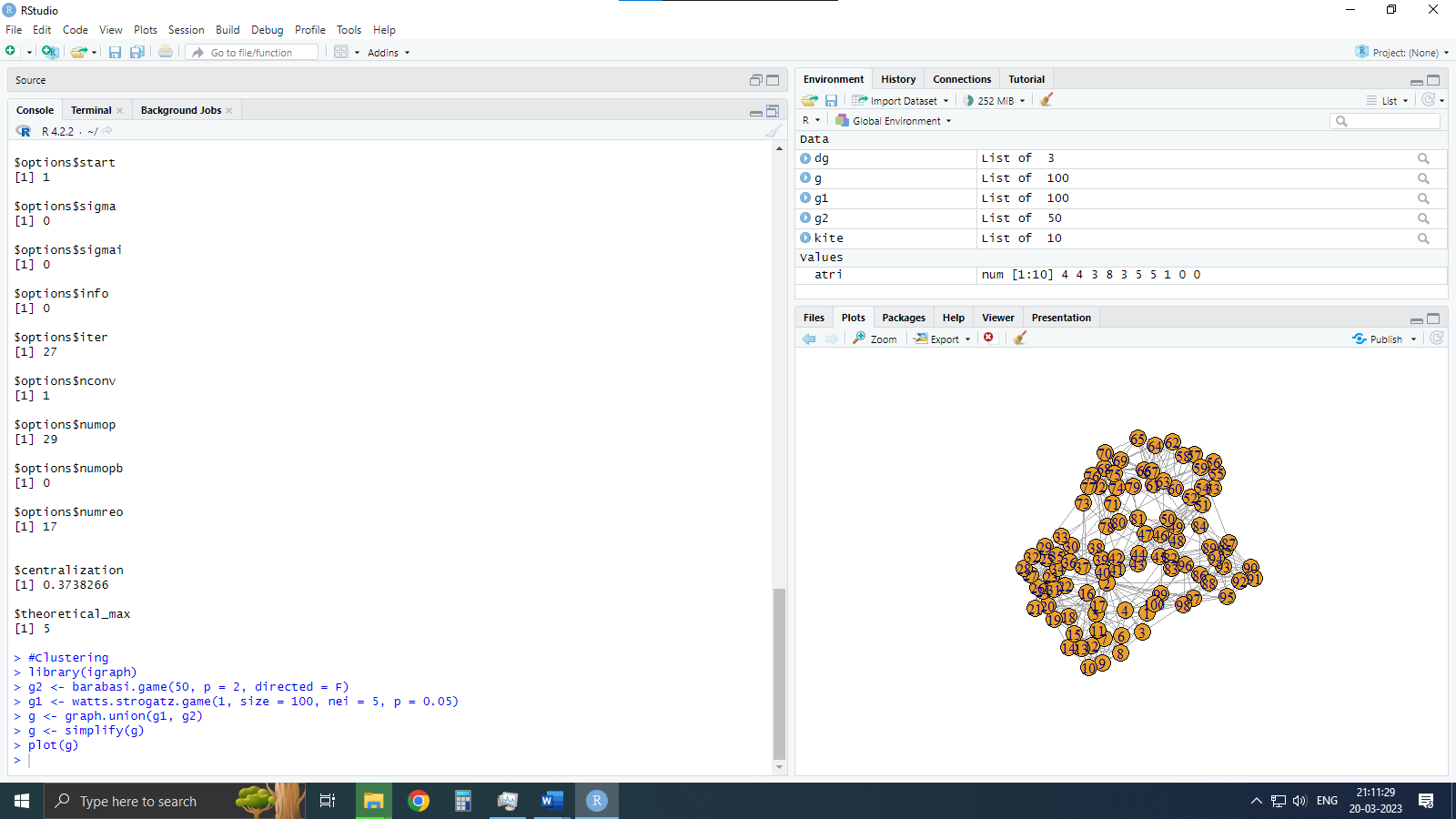
# Simplify the combined graph 'g'

g <- simplify(g)

# Plot the simplified graph 'g'

plot(g)

**OUTPUT**



**Practical 4: 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**

Code:

library(igraph)

# creating a matrix from a table

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))

# storing the row names in nms and removing the first column

nms <- matt[,1]

matt <- matt[, -1]

# setting the column and row names to be the same

colnames(matt) <- rownames(matt) <- nms

# replacing NA values with 0

matt[is.na(matt)] <- 0

# creating a weighted graph from the matrix

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

# plotting the graph

plot(g)

# calculating the shortest paths between all pairs of nodes

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

print(s.paths)

# calculating the shortest path between R and S

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

# plotting the graph with edge weights as labels

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

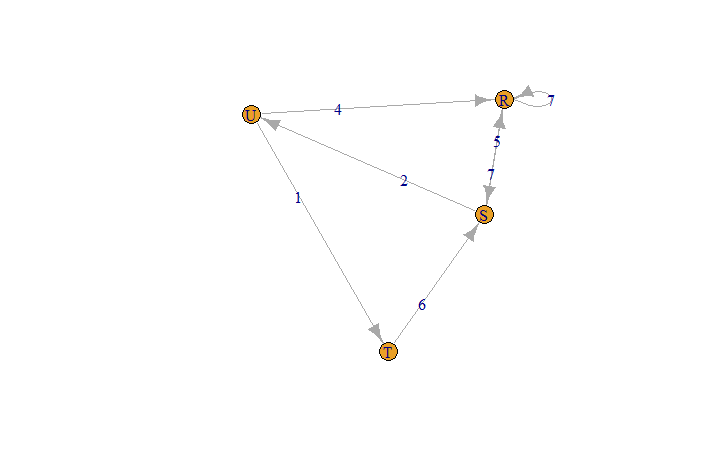
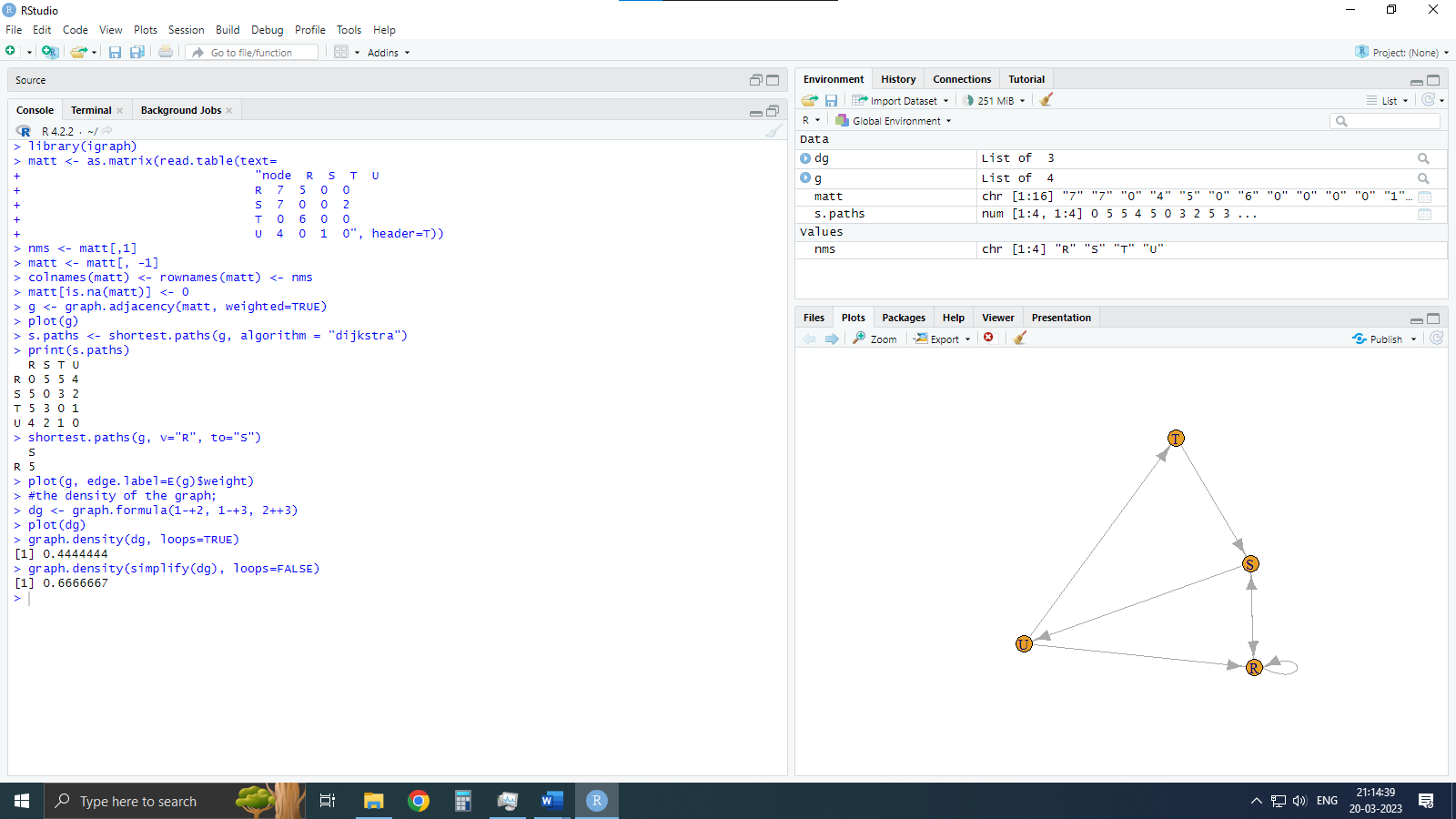
# calculating the density of the graph

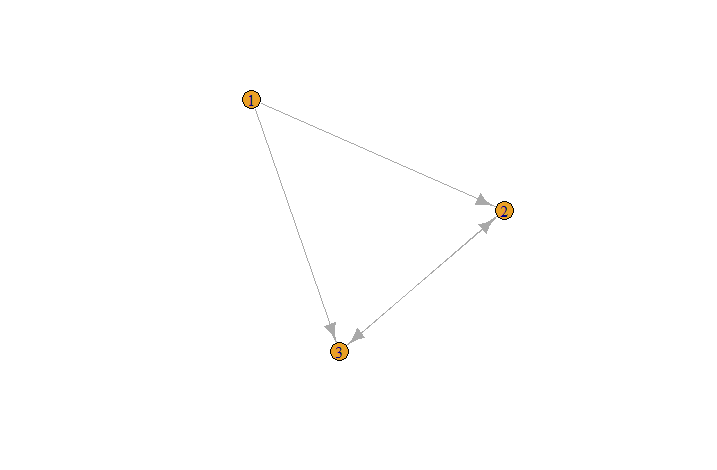
dg <- graph.formula(1-+2, 1-+3, 2++3)

plot(dg)

graph.density(dg, loops=TRUE)

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

**OUTPUT**



**Practical 5: 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.**

Code:

# Load igraph package

library(igraph)

# Define network using graph formula notation

ng <- graph.formula(Andy ++ Garth,

                    Garth -+ Bill,

                    Bill -+ Elena,

                    Elena ++ Frank,

                    Carol -+ Andy,

                    Carol -+ Elena,

                    Carol ++ Dan,

                    Carol ++ Bill,

                    Dan ++ Andy,

                    Dan ++ Bill)

# Plot the network

plot(ng)

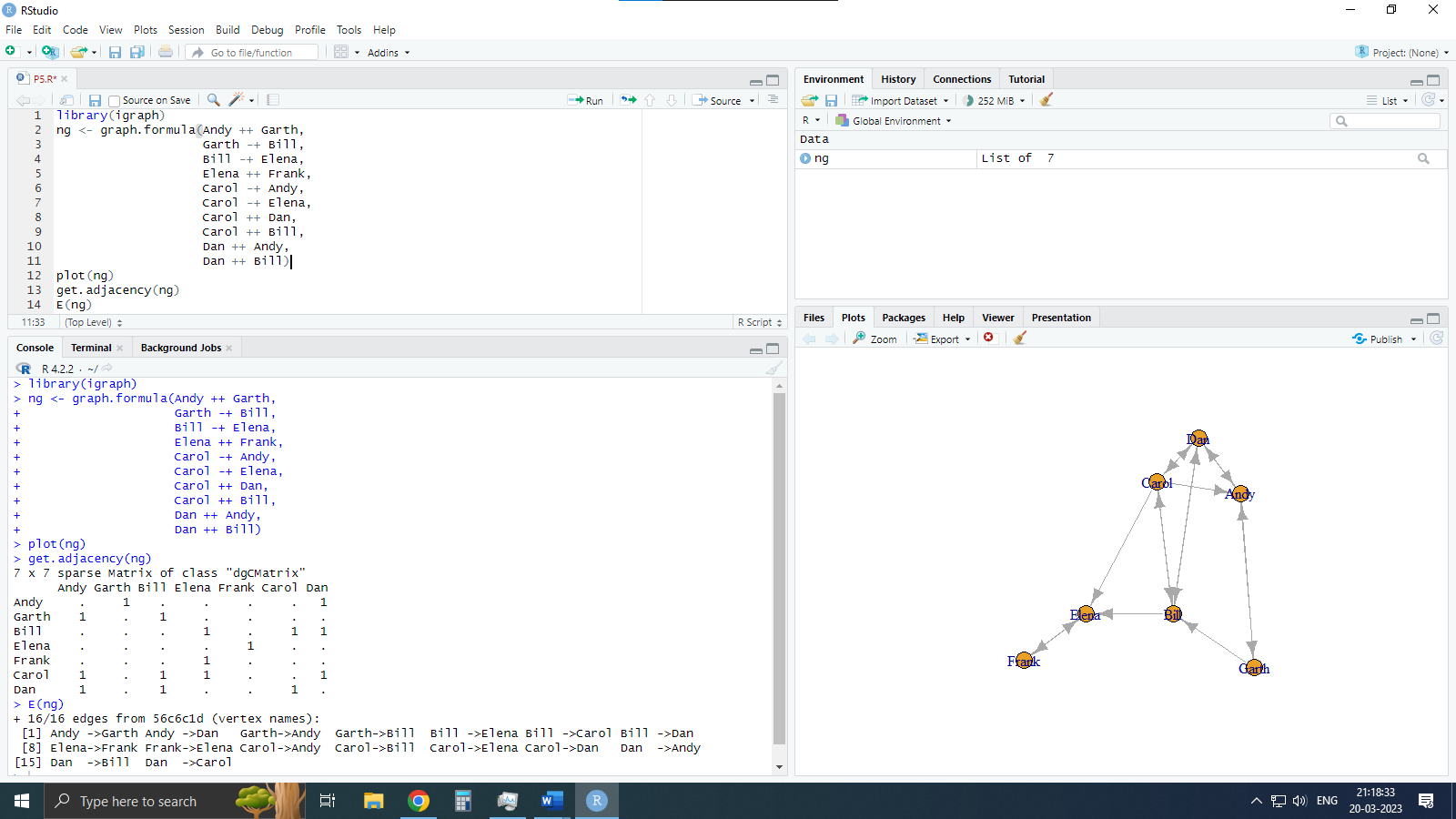
# Display adjacency matrix of the network

get.adjacency(ng)

# Display edges of the network

E(ng)

**OUTPUT**



**Practical 6: Write a program to exhibit structural equivalence, automorphic equivalence, and regular equivalence from a network.**

Codes:

# Install and load necessary packages

install.packages("sna")

library(sna)

library(igraph)

# Read data from file

links2 <- read.csv("edges1.csv", header = TRUE, row.names = 1)

# Equivalence clustering

eq <- equiv.clust(links2)

plot(eq)

# Automorphic equivalence

g.se <- sedist(links2)

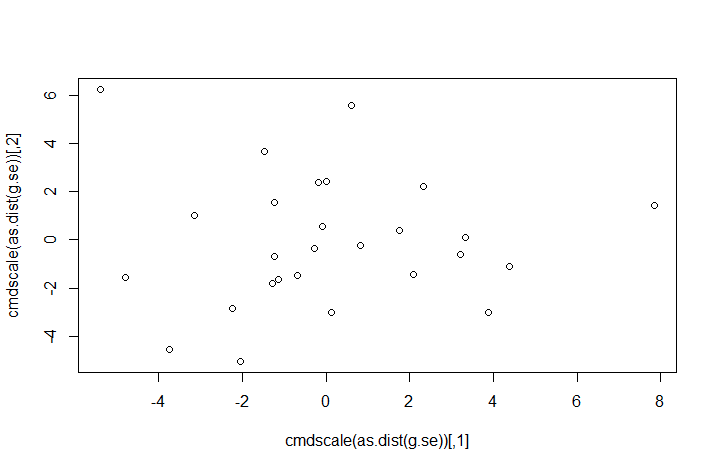
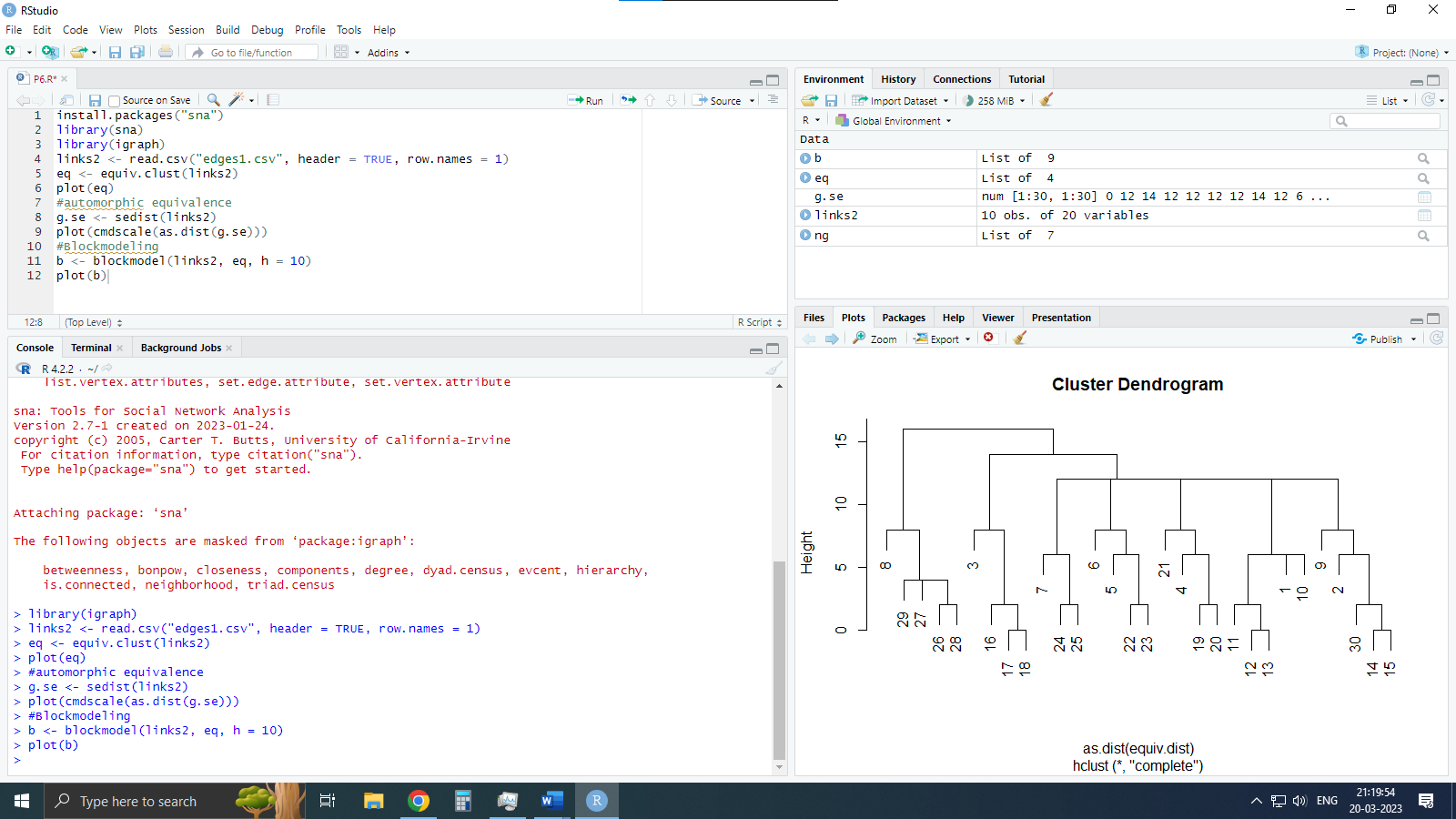
plot(cmdscale(as.dist(g.se)))

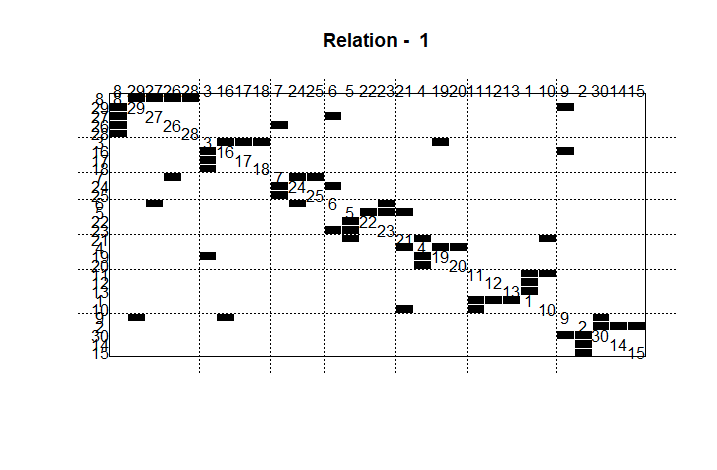
# Blockmodeling

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

plot(b)

**OUTPUT**





**Practical 7: Perform SVD analysis on network**

Code:

# Load the igraph library

library(igraph)

# Create a 9x4 matrix with specific values

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 the matrix to the console

print(a)

# Perform singular value decomposition on the matrix

svd(a)

**OUTPUT**