**Practical No 1**

**Aim: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.**

library(igraph)

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

plot(g)

ecount(g)

vcount(g)

degree(g)

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

plot(dg)

degree(dg,mode="in")

degree(dg,mode="out")

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

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

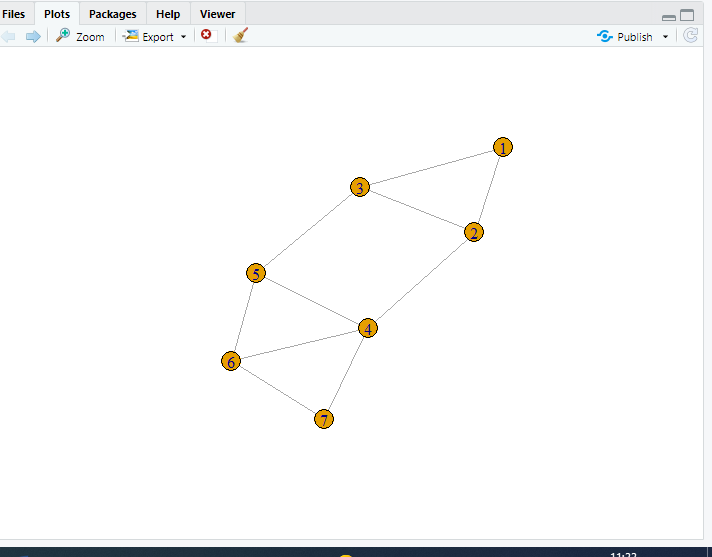
neighbors(g,5)

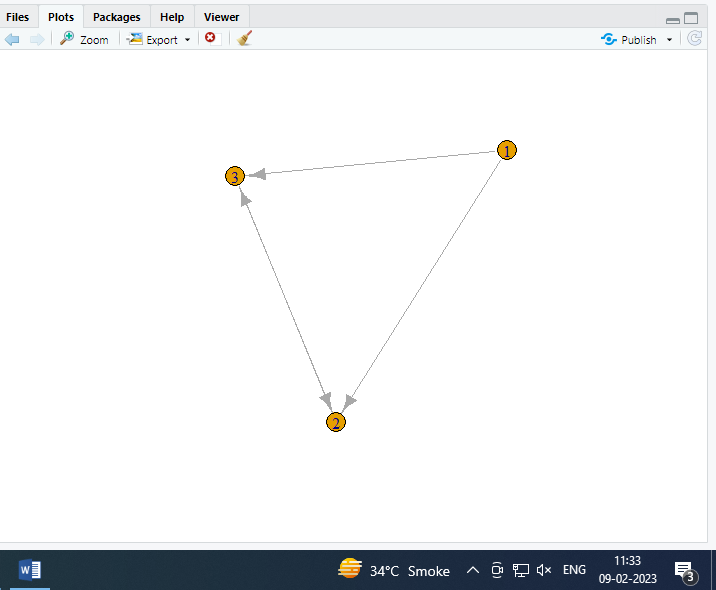
neighbors(g,2)

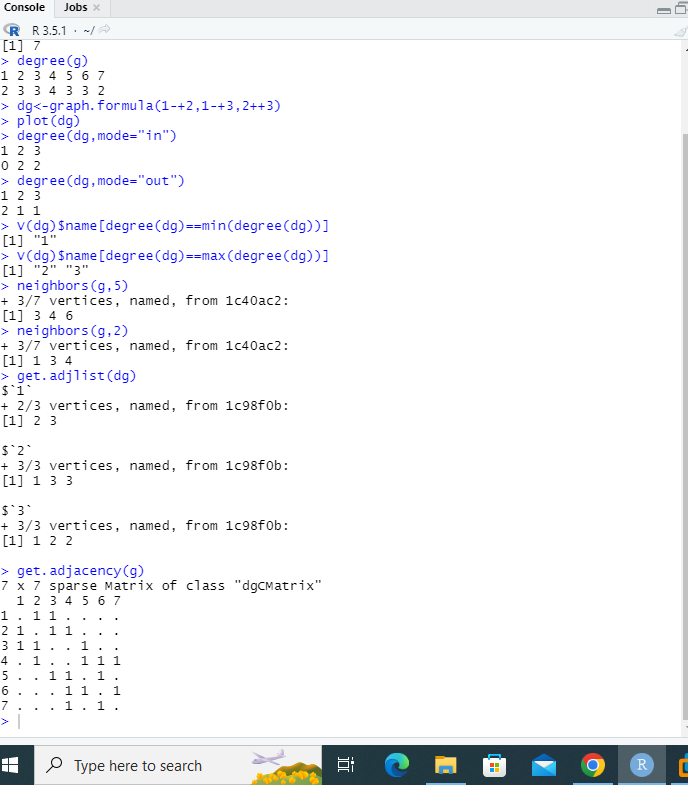
get.adjlist(dg)

get.adjacency(g)

**OUTPUT:**

****

****



**Practical no 2**

**Aim:Perform following tasks: (i) View data collection forms and/or import onemode/two-mode datasets;**

**(ii) Basic Networks matrices transformations**

**1.View data collection forms and/or import one-mode/ two-mode datasets;**

library(igraph)

getwd()

setwd("D:/SNA")

**#Reading data from a csv file**

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

head(nodes)

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

head(links)

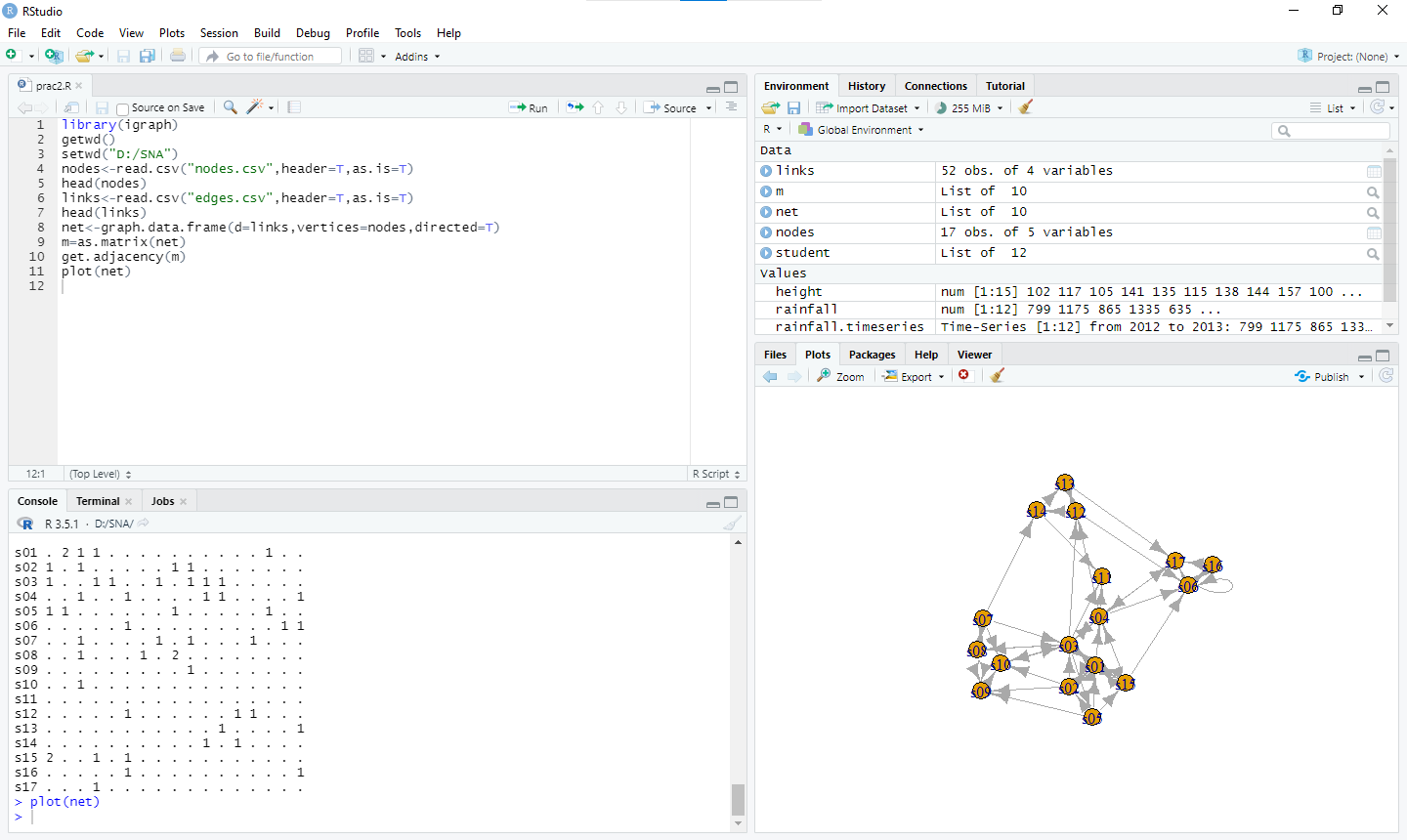
net<-graph.data.frame(d=links,vertices=nodes,directed=T)

m=as.matrix(net)

get.adjacency(m)

plot(net)

**OUTPUT:**



**Practical no 3:**

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

library(igraph)

getwd()

setwd("D:/SNA")

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

head(nodes)

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

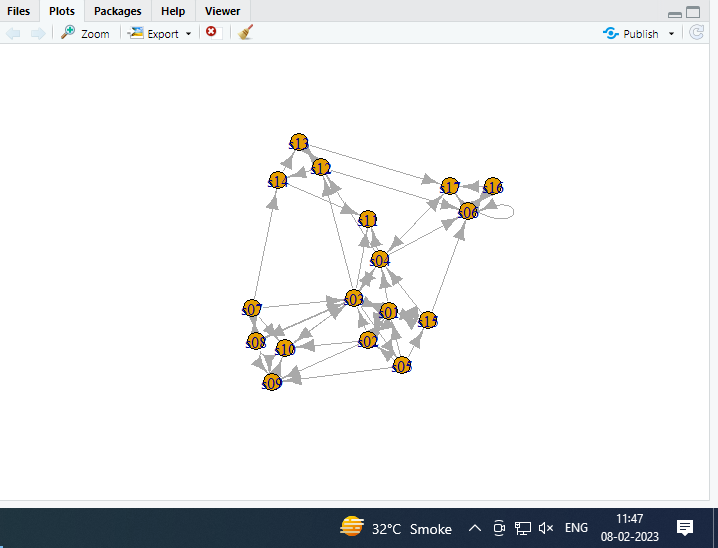
head(links)

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

g=as.matrix(net)

get.adjacency(g)

plot(net)



vcount(g)

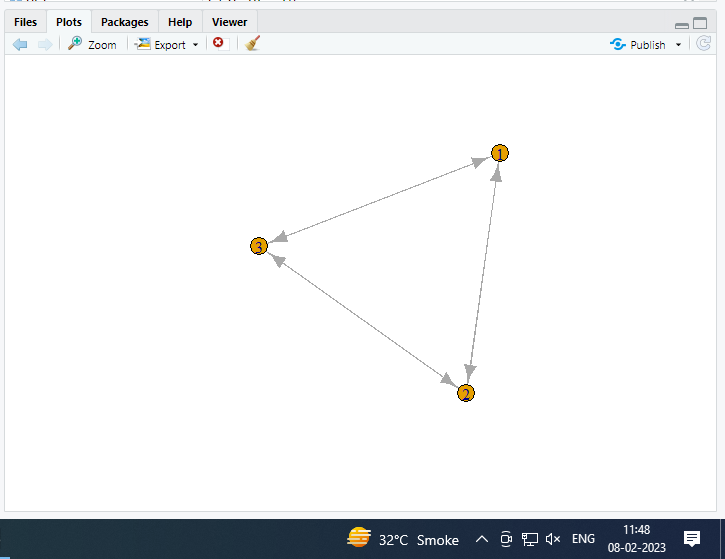
ecount(g)

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

degree(net)

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

plot(dg)



reciprocity(dg)

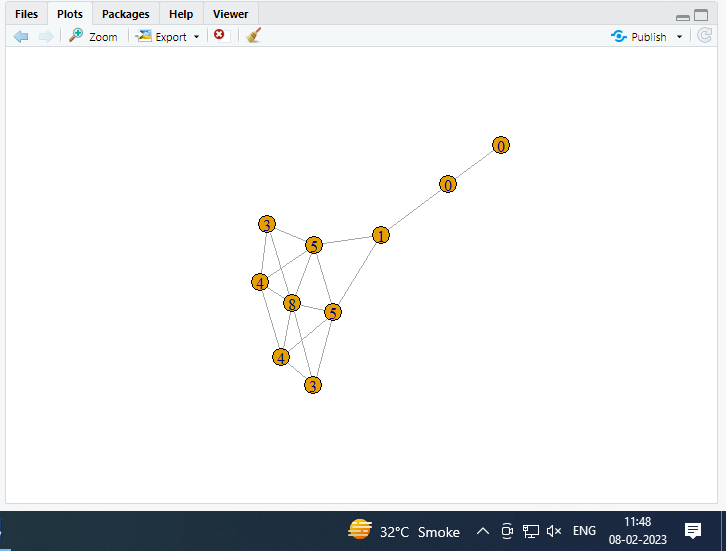
dyad.census(dg)

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

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

atri<-adjacent.triangles(kite)

plot(kite,vertex.label=atri)



transitivity(kite,type="local")

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

centralization.degree(net,mode="in",normalize="T")

closeness(net,mode = "all",weights = NA)

centralization.closeness(net,mode="all",normalized=T)

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

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

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

library(igraph)

#making own graph

#g2<-graph.formula(A++)

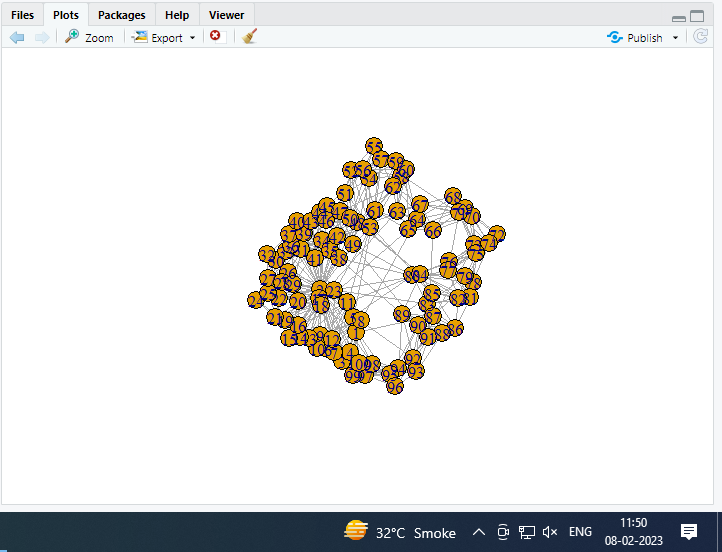
g2<-barabasi.game(50,p=2,directed=F)

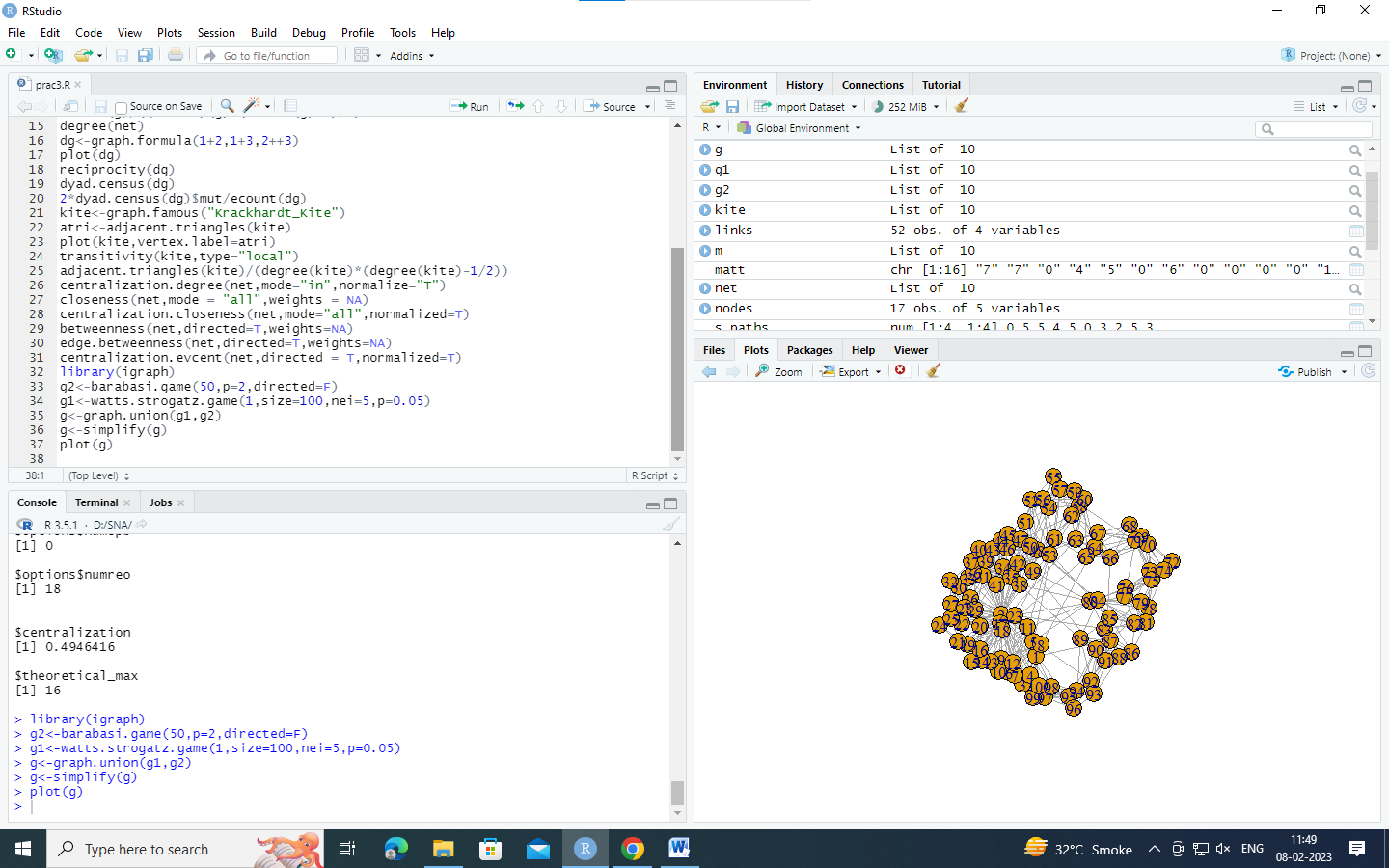
g1<-watts.strogatz.game(1,size=100,nei=5,p=0.05)

g<-graph.union(g1,g2)

g<-simplify(g)

plot(g)

**Output:**



**Practical no 4:**

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

**#(i) Length of the shortest path from a given**

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 ]

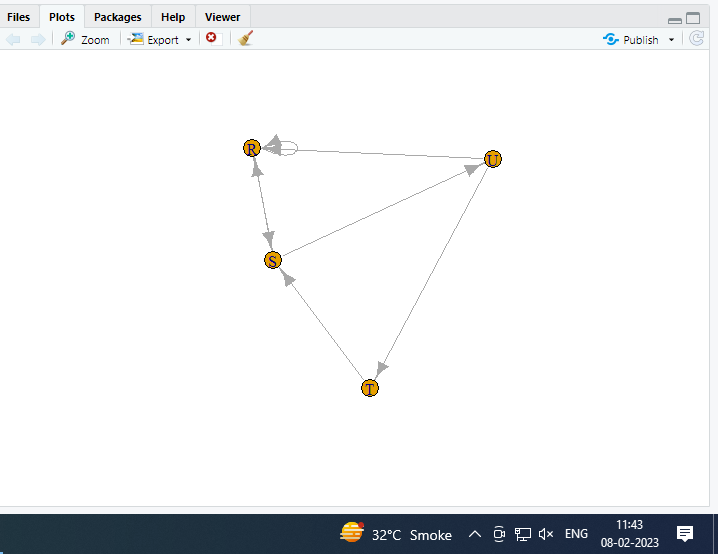
matt <- matt[, -1]

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

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

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

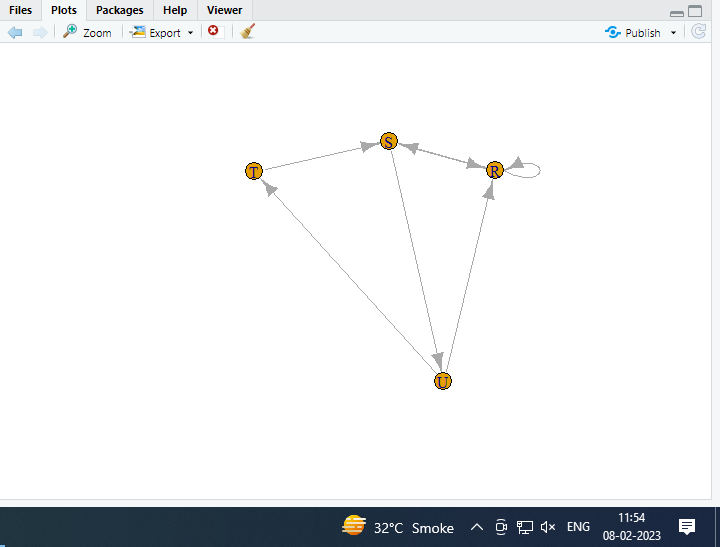
plot(g)



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

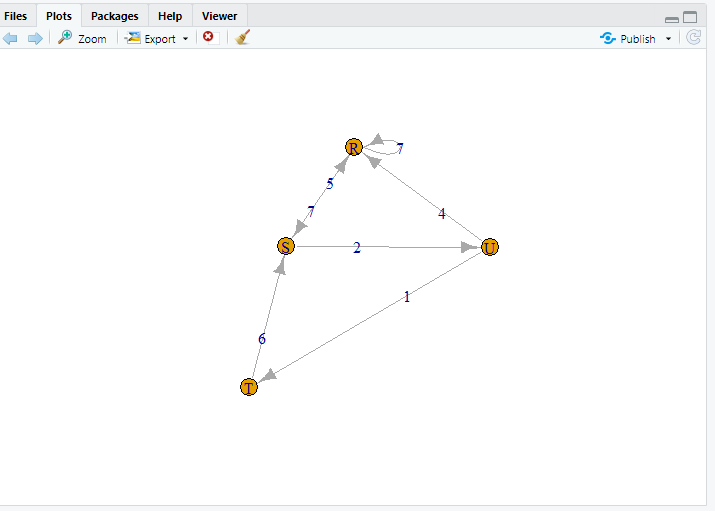
print(s.paths)

plot(g)



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

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

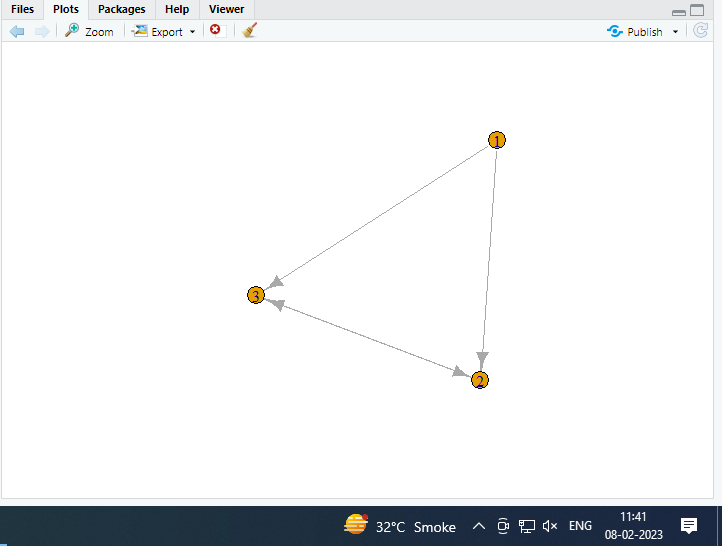


#(ii) the density of the graph;

library(igraph)

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

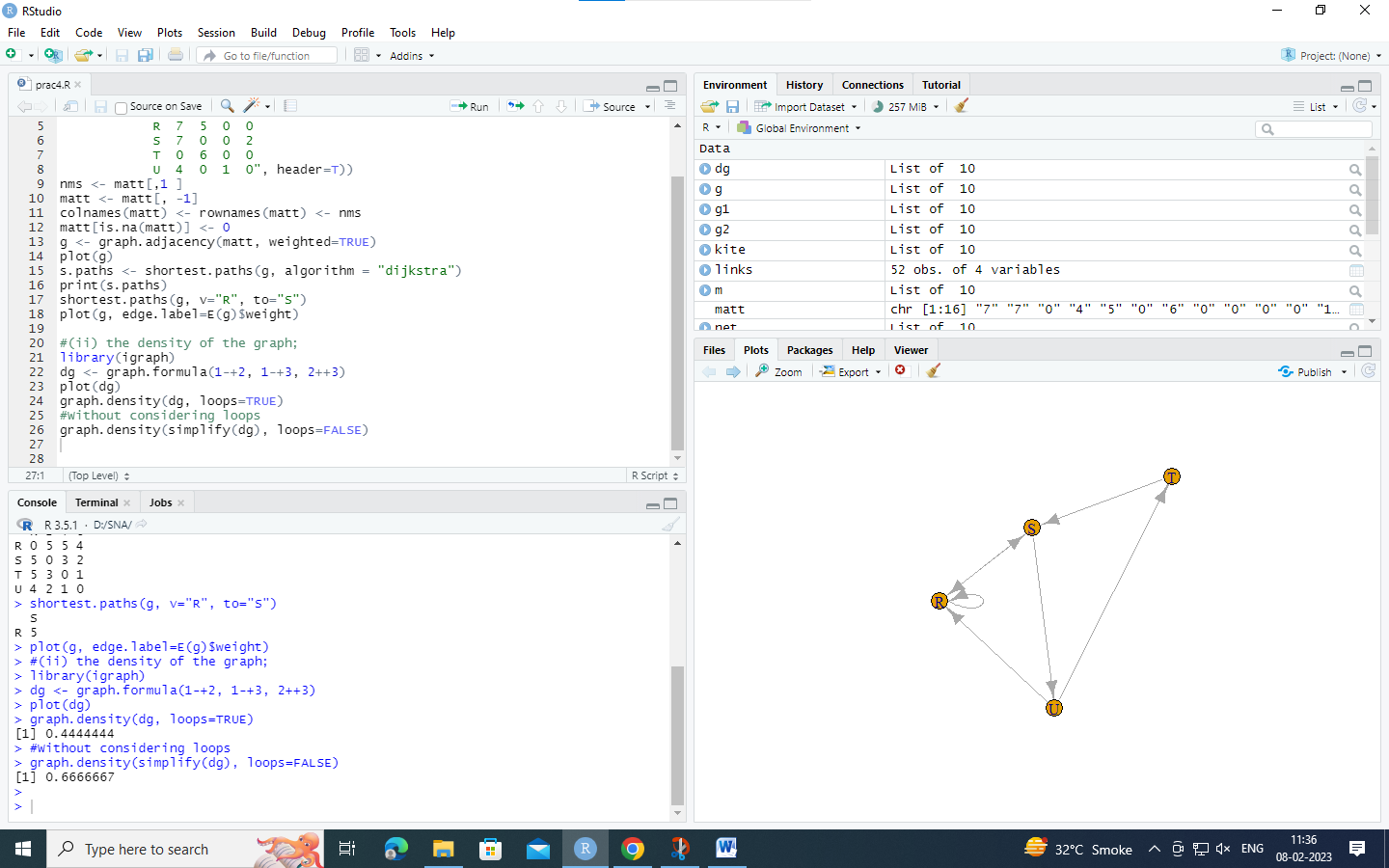
plot(dg)



graph.density(dg, loops=TRUE)

#Without considering loops

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



**Practical no 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**

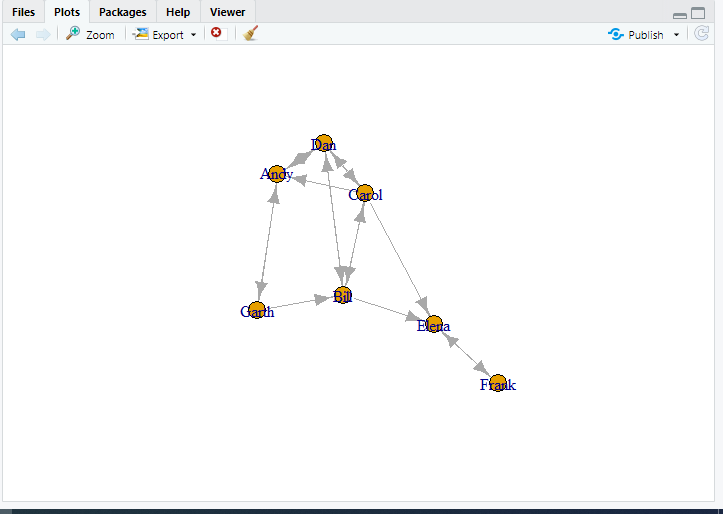
**networks representatives of each.**

**#1)a network as a sociogram (or “network graph”)**

library(igraph)

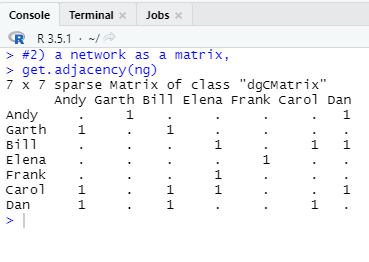
ng<-graph.formula(Andy++Garth,Garth-+Bill,Bill-+Elena,Elena++Frank,Carol-+Andy,Carol +Elena,Carol++Dan,Carol++Bill,Dan++Andy,Dan++Bill)

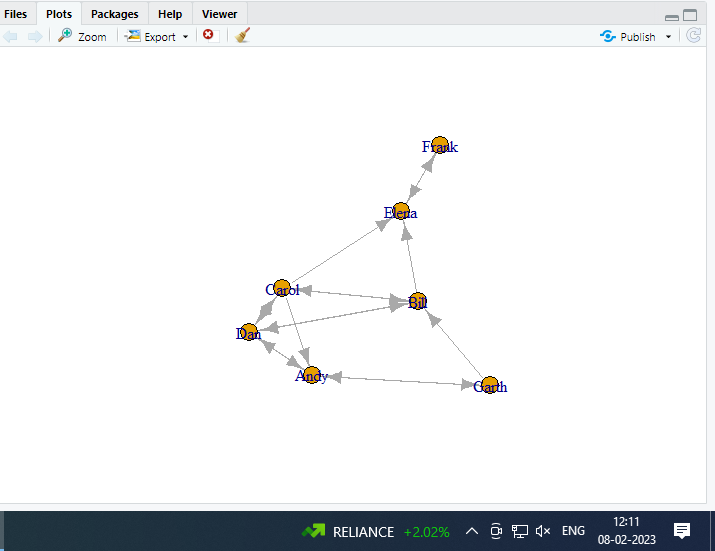
plot(ng)

****

**#2) a network as a matrix,**

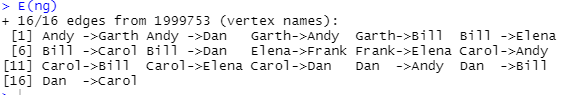
get.adjacency(ng)



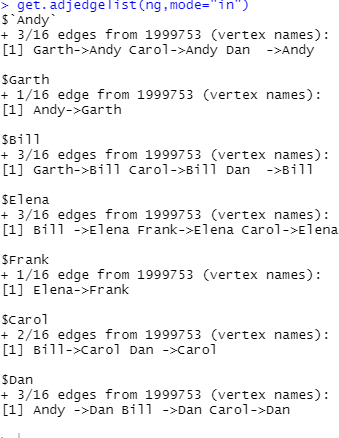


**#3) a network as an edge list.**

E(ng)



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



**Practical no:6**

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

**1) structural equivalence**

>library(sna)

>library(igraph)

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

>eq<-equiv.clust(links2)

>plot(eq)

Chart, box and whisker chart

Description automatically generated

**2) automorphic equivalence,**

>g.se<sedist(links2)

 Plot a metric MDS of vertex positions in two dimensions

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

Chart, application

Description automatically generated with medium confidence

**3) regular equivalence from a network.**

**Blockmodeling**

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

>plot(b)

A picture containing chart

Description automatically generated

**Practical No 7**

**Aim: Create sociograms for the persons-by-persons network and the committee-by committee network for a given relevant problem. Create one-mode network and two-node network for the same.**

>ibrary(Dominance)

>data(data\_Network\_1)

**## set 1 for action you want to show**

>bytes= "00111111111000000000"

>Sociogram(data\_Network\_1,bytes)

Graphical user interface

Description automatically generated with low confidence

A picture containing timeline

Description automatically generated

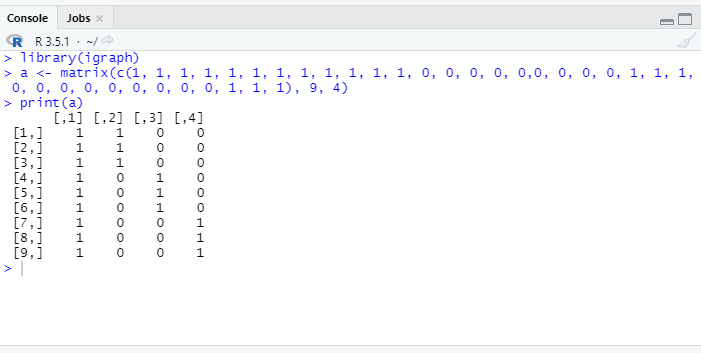
**Practical no:8**

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

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)

