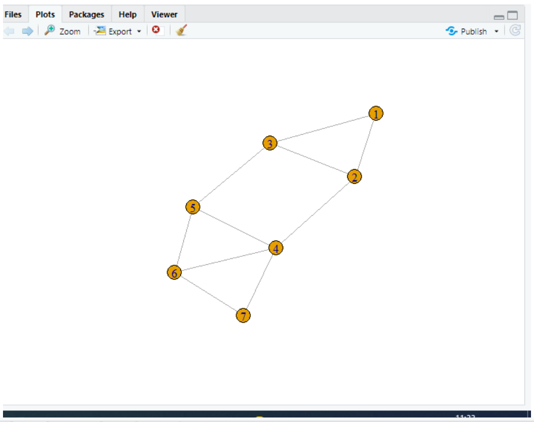
Name: Akhil R Gangan **Roll no:** 530

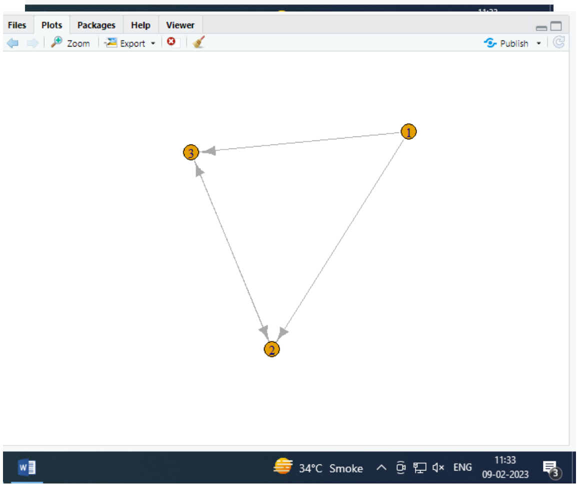
Practical No 1

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:





Name: Akhil R Gangan **Roll no:** 530

Console Jobs

R 3.5.1 · ~/

```
[1] 7
> degree(g)
1 2 3 4 5 6 7
2 3 3 4 3 3 2
> dg<-graph.formula(1-+2,1-+3,2++3)
> plot(dg)
> degree(dg,mode="in")
1 2 3
0 2 2
> degree(dg,mode="out")
1 2 3
2 1 1
> V(dg)$name[degree(dg)==min(degree(dg))]
[1] "1"
> V(dg)$name[degree(dg)==max(degree(dg))]
[1] "2" "3"
> neighbors(g,5)
+ 3/7 vertices, named, from 1c40ac2:
[1] 3 4 6
> neighbors(g,2)
+ 3/7 vertices, named, from 1c40ac2:
[1] 1 3 4
> get.adjlist(dg)
$`1`
+ 2/3 vertices, named, from 1c98f0b:
[1] 2 3

$`2`
+ 3/3 vertices, named, from 1c98f0b:
[1] 1 3 3

$`3`
+ 3/3 vertices, named, from 1c98f0b:
[1] 1 2 2

> get.adjacency(g)
7 x 7 sparse Matrix of class "dgCMatrix"
  1 2 3 4 5 6 7
1 . 1 1 . . . .
2 1 . 1 1 . . .
3 1 1 . . 1 . .
4 . 1 . . 1 1 1
5 . . 1 1 . 1 .
6 . . . 1 1 . 1
7 . . . 1 . 1 .
> |
```

Type here to search

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)

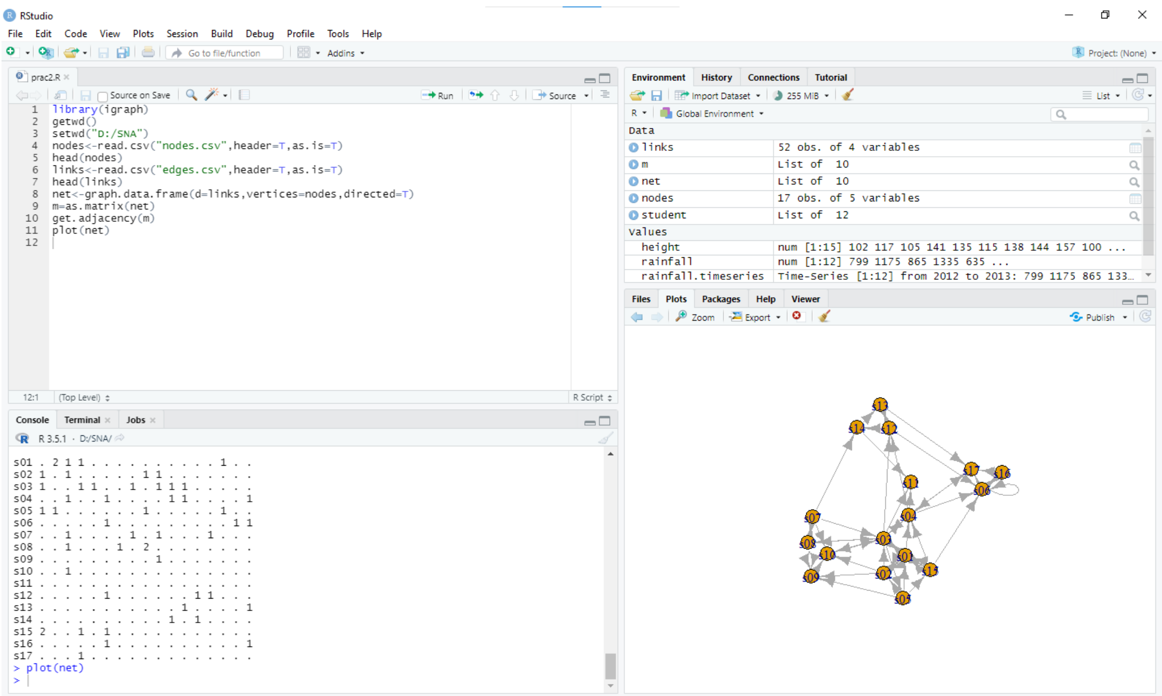
net<-graph.data.frame(d=links,vertices=nodes,directed=T)

m=as.matrix(net)

get.adjacency(m)

plot(net)

OUTPUT:

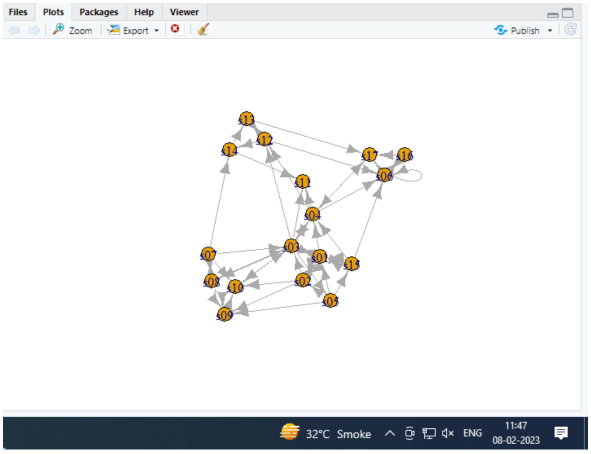


Name: Akhil R Gangan **Roll no:** 530

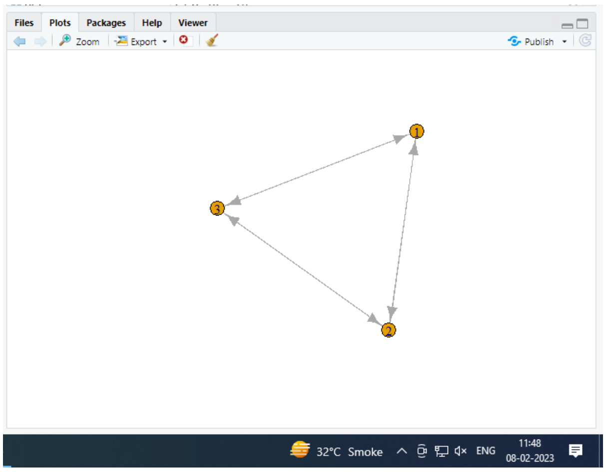
Practical no 3

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

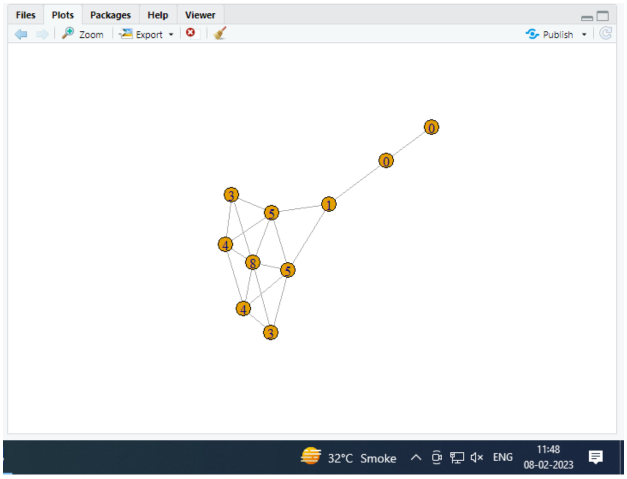
```
library(igraph)
getwd()
setwd("D:/SNA")
nodes<-read.csv("nodes.csv",header=T,as.is=T)
head(nodes)
links<-read.csv("edges.csv",header=T,as.is=T)
head(links)
net<-graph.data.frame(d=links,vertices=nodes,directed=T)
g=as.matrix(net)
get.adjacency(g)
plot(net)
```




```
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)
```



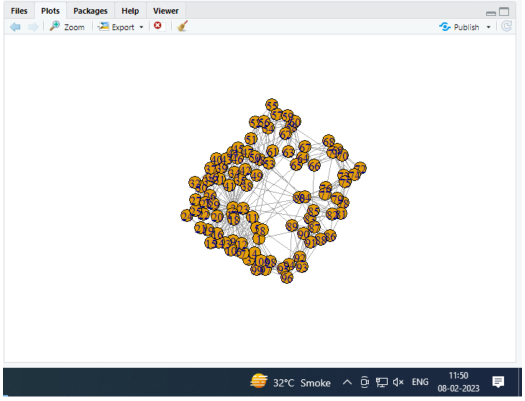
reciprocity(dg)
dyad.census(dg)
2*dyad.census(dg)$mut/ecount(dg)
kite<-graph.famous("Krackhardt_Kite")
atri<-adjacent.triangles(kite)
plot(kite,vertex.label=atri)



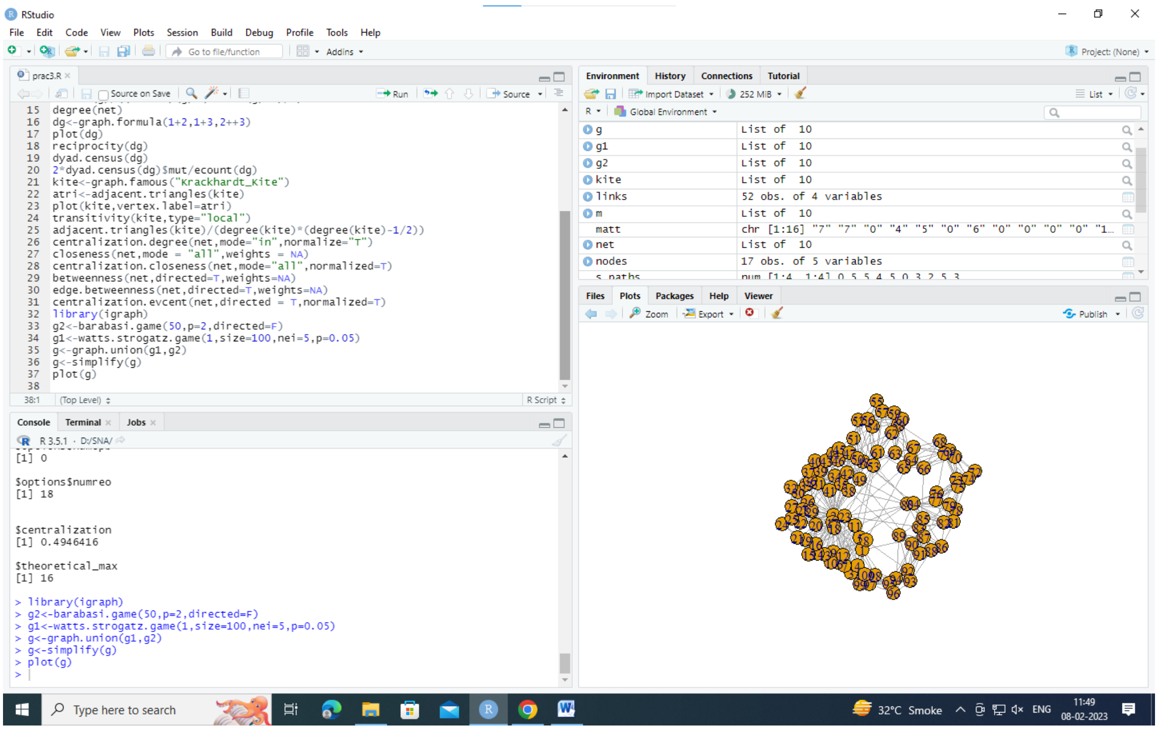
transitivity(kite,type="local")
adjacent.triangles(kite)/(degree(kite)*(degree(kite)-1/2))
centralization.degree(net,mode="in",normalize="T")
closeness(net,mode = "all",weights = NA)
centralization.closeness(net,mode="all",normalized=T)
betweenness(net,directed=T,weights=NA)
edge.betweenness(net,directed=T,weights=NA)
centralization.evcent(net,directed = T,normalized=T)
library(igraph)
#making own graph
#g2<-graph.formula(A++)

Name: **Akhil R Gangan** **Roll no:** 530

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:



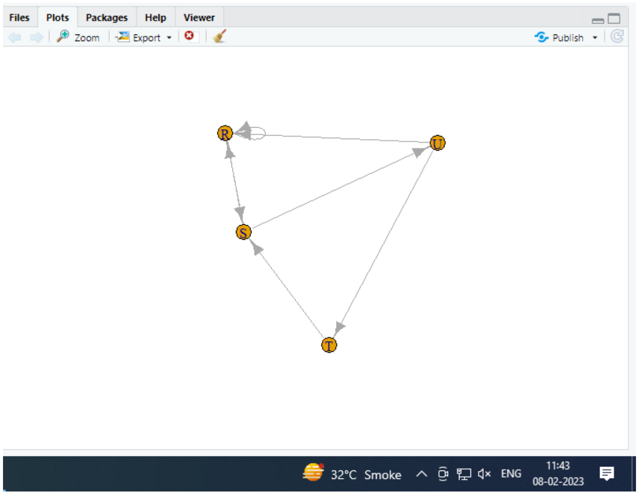
Name: Akhil R Gangan Roll no: 530

Practical no 4

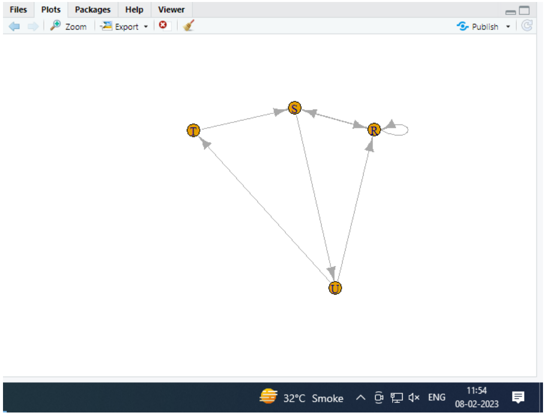
Aim: For a given network find the following: (i) Length of the shortest path from a given node to another; (ii) the density of the graph

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

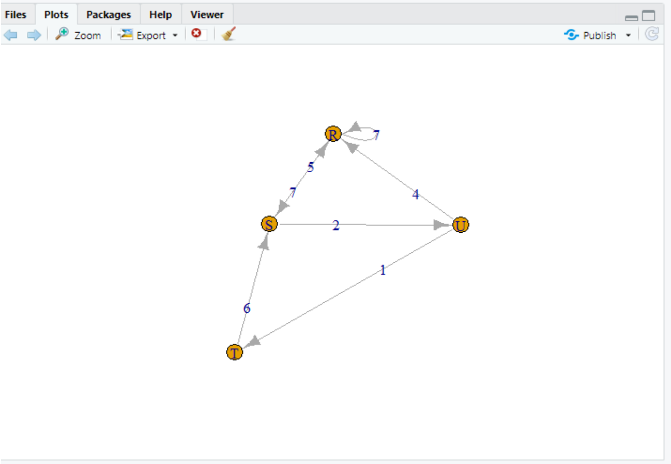
```
library(igraph)
matt <- as.matrix(read.table(text=
        "node  R  S  T  U
         R  7  5  0  0
         S  7  0  0  2
         T  0  6  0  0
         U  4  0  1  0", header=T))
nms <- matt[,1 ]
matt <- matt[, -1]
colnames(matt) <- rownames(matt) <- nms
matt[is.na(matt)] <- 0
g <- graph.adjacency(matt,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



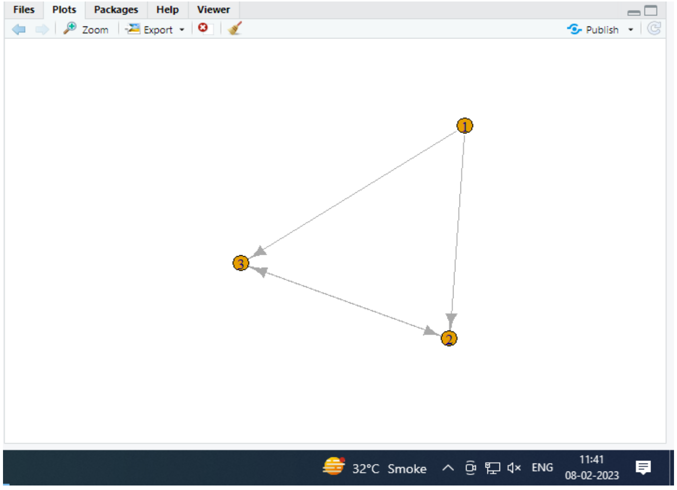

```
s.paths <- shortest.paths(g, algorithm = "dijkstra")
print(s.paths)
plot(g)
```



shortest.paths(g, v="R", to="S")
plot(g, edge.label=E(g)$weight)



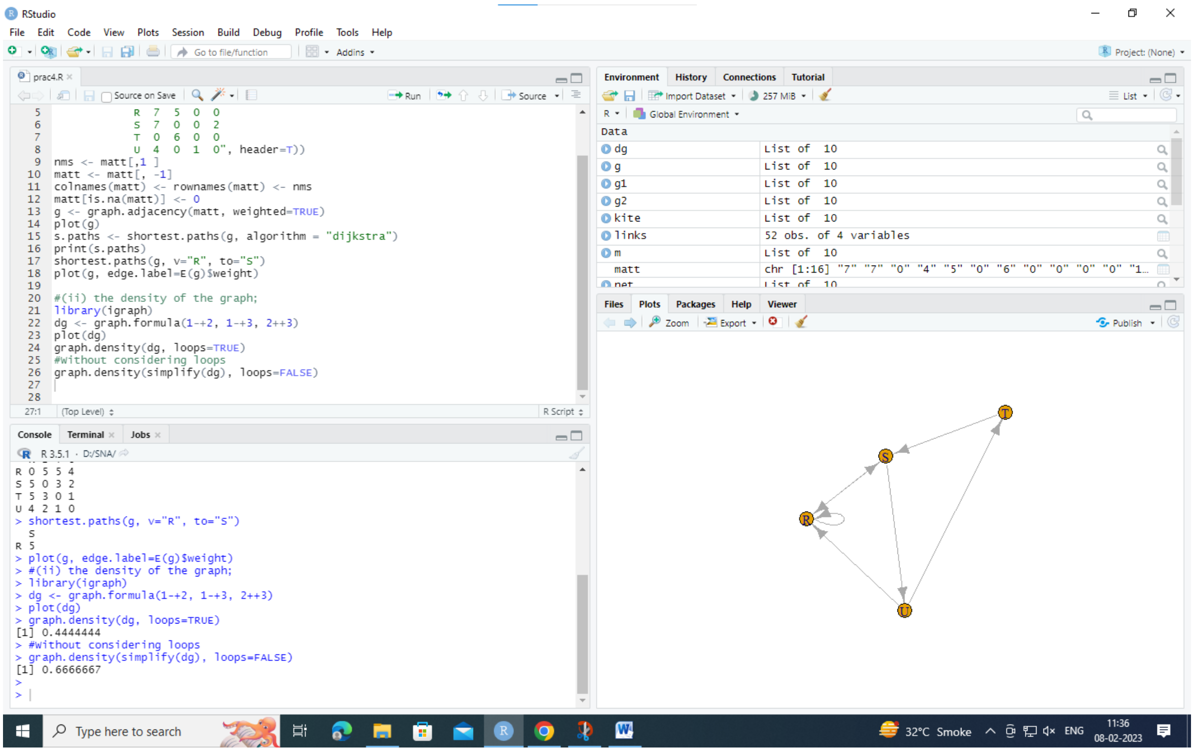
#(ii) the density of the graph;
library(igraph)
dg <- graph.formula(1-+2, 1-+3, 2++3)
plot(dg)



graph.density(dg, loops=TRUE)
#Without considering loops

Name: Akhil R Gangan **Roll no:** 530

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



Practical no 5

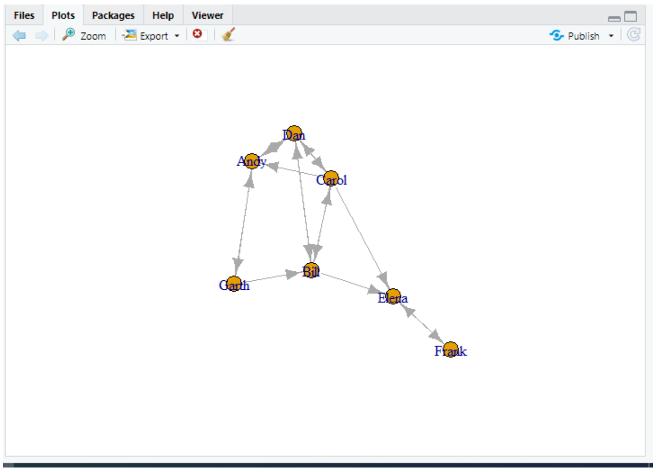
Aim: Write a program to distinguish between a network as a matrix, a network as an edge list, and a network as a sociogram (or “network graph”) using 3 distinct networks representatives of each.

#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)

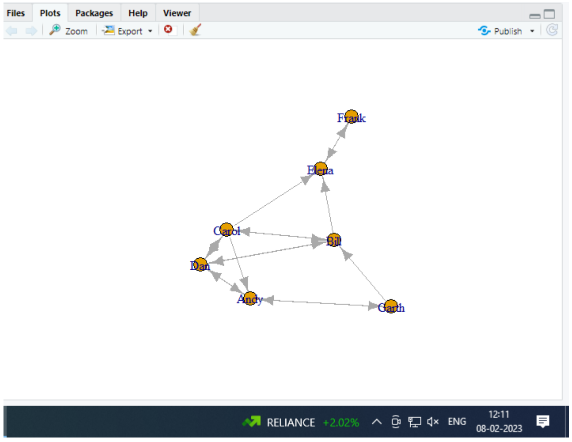
plot(ng)



#2) a network as a matrix,

get.adjacency(ng)

```
Console   Terminal   Jobs
R 3.5.1 · ~/
> #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)

```
> E(ng)
+ 16/16 edges from 1999753 (vertex names):
 [1] Andy ->Garth Andy ->Dan   Garth->Andy  Garth->Bill  Bill ->Elena
 [6] Bill ->Carol Bill ->Dan   Elena->Frank Frank->Elena Carol->Andy
[11] Carol->Bill  Carol->Elena Carol->Dan   Dan  ->Andy  Dan  ->Bill
[16] Dan  ->Carol
```

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

```
> get.adjedgelist(ng,mode="in")
$`Andy`
+ 3/16 edges from 1999753 (vertex names):
[1] Garth->Andy Carol->Andy Dan  ->Andy

$Garth
+ 1/16 edge from 1999753 (vertex names):
[1] Andy->Garth

$Bill
+ 3/16 edges from 1999753 (vertex names):
[1] Garth->Bill Carol->Bill Dan  ->Bill

$Elena
+ 3/16 edges from 1999753 (vertex names):
[1] Bill ->Elena Frank->Elena Carol->Elena

$Frank
+ 1/16 edge from 1999753 (vertex names):
[1] Elena->Frank

$Carol
+ 2/16 edges from 1999753 (vertex names):
[1] Bill->Carol Dan ->Carol

$Dan
+ 3/16 edges from 1999753 (vertex names):
[1] Andy ->Dan Bill ->Dan Carol->Dan
```

Practical No 6

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

i) structural equivalence

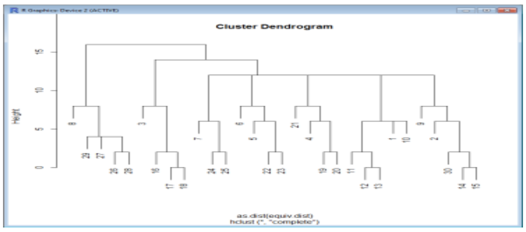
> library(sna)

> library(igraph)

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

> eq<-equiv.clust(links2)

> plot(eq)

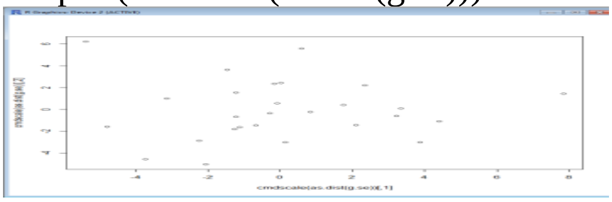


ii) automorphic equivalence,

>g.se<-sedist(links2)

<u>Plot a metric MDS of vertex positions in two dimensions</u>

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

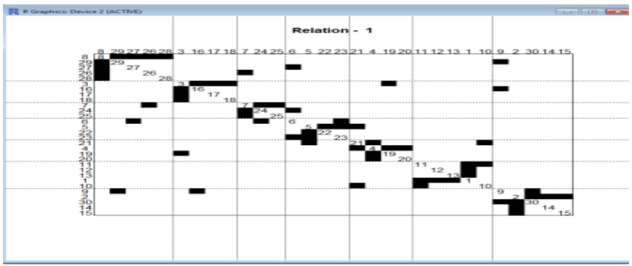


3) regular equivalence from a network.

<u>Blockmodeling</u>

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

> plot(b)



Practical No 7

Aim: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.

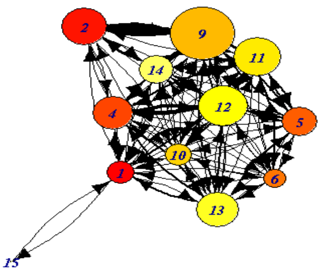
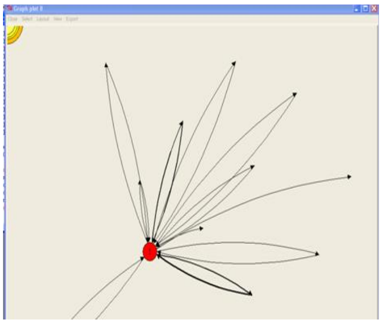
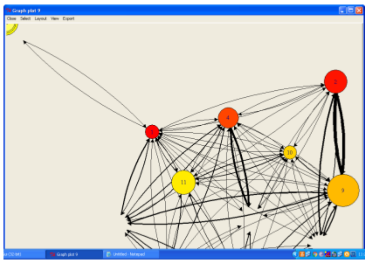
>library(Dominance)

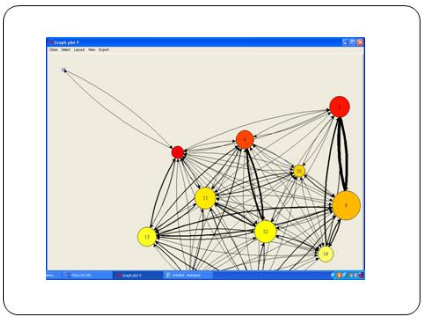
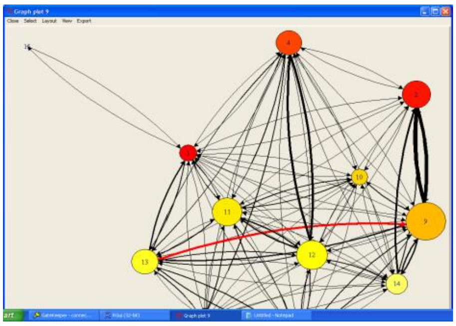
>data(data_Network_1)

 ## set 1 for action you want to show

>bytes= "0011111111000000000"

>Sociogram(data_Network_1,bytes)

```
> print(data_Network_1)
   Name Beschreibung item.number dominance.order  age  sex action.from.
1     1       Pferd1           1               1   NA    2            4
2     2       Pferd2           2               2   NA    1            9
3     3       Pferd3           3              NA   NA    1            4
4     4       Pferd4           4               5   NA    1           12
5     5       Pferd5           5              10   NA    1            5
6     6       Pferd6           6               3   NA    1            9
7     7       Pferd7           7               6   NA    1            5
8     8       Pferd8           8              NA   NA    1            9

   action.to kind.of.action                time test.2.kind.of.action
1          9             11                <NA>                     3
2          4             11 2009-06-07 03:30:00                     3
3         12             11                <NA>                     3
4          4             11                <NA>                     3
5          9             11                <NA>                     3
6          5             11                <NA>                     3
```

	test.3.kind.of.acttion	name.of.action	action.number	classification
1	3	leading	1	1
2	3	following	2	2
3	3	approach	3	1
4	3	bite	4	1
5	3	threat to bite	5	1
6	3	kick	6	1

	weighting
1	1
2	-1
3	1
4	1
5	1
6	1

Name: Akhil R Gangan **Roll no:** 530

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)

```
Console   Jobs
R 3.5.1 · ~/
> 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)

```
Console   Jobs
R 3.5.1 · ~/
 [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

> |
```