

Name: Ashwinikumar Singh

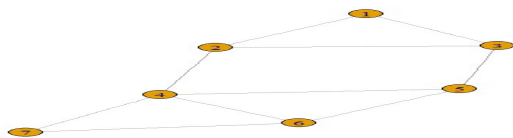
Roll\_no:539

### libraryPractical 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)
```



1)number of edges

```
> ecount(g)
```

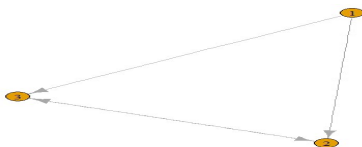
```
[1] 10
```

2)no of nodes

```
> vcount(g)
```

```
[1] 7
```

3)Degree Of nodes



```
> 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
```

#### 4) Node with lowest degree

```
> V(dg)$name[degree(dg)==min(degree(dg))]
[1] "1"
```

#### Node with highest degree

```
> V(dg)$name[degree(dg)==max(degree(dg))]
[1] "2" "3"
```

#### 5) To find neighbours / adjacency list:

```
> neighbors(g,5)
```

```
[1] 3 4 6
```

```
> neighbors(g,2)
```

```
[1] 1 3 4
```

```
> get.adjlist(dg)
```

```
$`1`
```

```
[1] 2 3
```

```
$`2`
```

```
[1] 1 3 3
```

```
$`3`
```

```
[1] 1 2 2
```

#### 6) Adjacency Matrix

```
> 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 .

## Practical No 2

### Aim:

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

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

```
getwd()
```

```
[1] "C:/Users/admin/Documents"
```

```
> setwd("d:/SNA_pract")
```

### Reading data from a csv file

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

A	B	C	D	E	F	G	H	I
1	id	media	media.type	type.label	audience.size			
2	s01	NY Times	1	Newspaper	20			
3	s02	Washington Post	1	Newspaper	25			
4	s03	Wall Street Journal	1	Newspaper	30			
5	s04	USA Today	1	Newspaper	32			
6	s05	LA Times	1	Newspaper	20			
7	s06	New York Post	1	Newspaper	50			
8	s07	CNN	2	TV	56			
9	s08	MSNBC	2	TV	34			
10	s09	FOX News	2	TV	60			
11	s10	ABC	2	TV	23			
12	s11	BBC	2	TV	34			
13	s12	Yahoo News	3	Online	33			
14	s13	Google News	3	Online	23			
15	s14	Reuters	3	Online	12			
16	s15	NYTimes	3	Online	24			
17	s16	Washington Post	3	Online	28			
18	s17	AOL.com	3	Online	33			
19								
20								
21								

```
> head(nodes)
```

### Output:-

	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("edges.csv", header=T, as.is=T)
```

A	B	C	D	E	F	G	H	I
id	media	type	label	audience	size			
s01	NY Times	1	Newspaper	20				
s02	Washington	1	Newspaper	25				
s03	Wall Street	1	Newspaper	30				
s04	USA Today	1	Newspaper	32				
s05	LA Times	1	Newspaper	20				
s06	New York	1	Newspaper	50				
s07	CNN	2	TV	56				
s08	MSNBC	2	TV	34				
s09	FOX News	2	TV	60				
s10	ABC	2	TV	23				
s11	BBC	2	TV	34				
s12	Yahoo News	3	Online	33				
s13	Google News	3	Online	23				
s14	Reuters.com	3	Online	12				
s15	NYTimes.com	3	Online	24				
s16	Washington	3	Online	28				
s17	AOL.com	3	Online	33				

```
> head(links)
```

Output:-

```

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

```

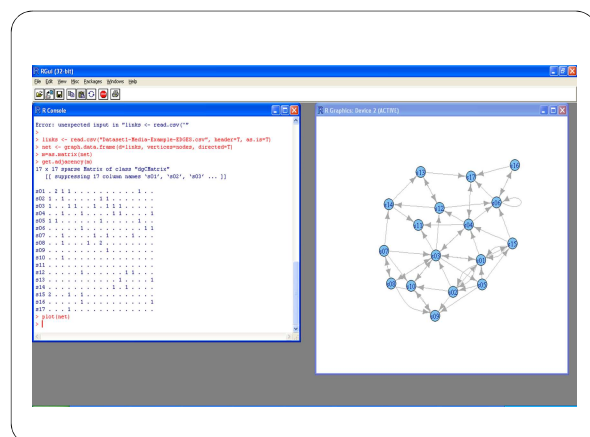
## (ii) Basic Networks matrices transformations

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

```
> m=as.matrix(net)
```

```
> get.adjacency(m)
```

```
> plot(net)
```



## Practical N0 3

**Aim:**

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

### 1) Density

```
>vcount(g)
```

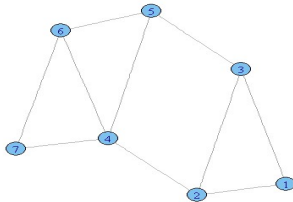
```
[1] 7
```

```
> ecount(g)
```

```
[1] 10
```

```
> ecount(g)/(vcount(g)*(vcount(g)-1)/2)
```

```
[1] 0.4719
```



### 2) Degree

```
> degree(net)
```

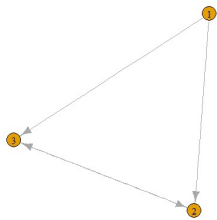
```
s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 s11 s12
```

```
10  7  13  9  5  8  5  6  5  5  3  6
```

```
s13 s14 s15 s16 s17
```

```
4  4  6  3  5
```

### 3) Reciprocity:



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

```
>plot(dg)
```

```
> reciprocity(dg)
```

```
[1] 0.5
```

### ● **Formula**

```
> dyad.census(dg)
```

```
$mut
```

```
[1] 1
```

```
$asym
```

```
[1] 2
```

```
$null
```

```
[1] 0
```

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

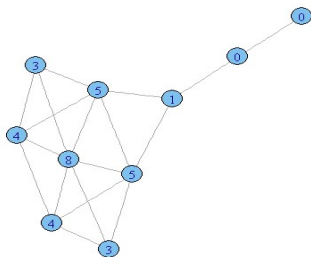
```
[1] 0.5
```

### **4)Transitivity**

```
> 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
```

```
[7] 0.5000000 0.3333333 0.0000000      NaN
```

### **Formula**

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

```
[1] 0.6666667 0.6666667 1.0000000 0.5333333 1.0000000 0.5000000
```

```
[7] 0.5000000 0.3333333 0.0000000      NaN
```

## 5)Centralization

### ● Degree of centrality

```
> centralization.degree(net, mode="in", normalized=T)
```

### ● Closeness Centralization

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

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

### ● Betweenness Centrality

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

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

```
> centralization.betweenness(net, directed=T, normalized=T)
```

### ● Eigenvector centrality

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

## 6) Clustering

```
>library(igraph)
```

```
# let's generate two networks and merge them into one graph.
```

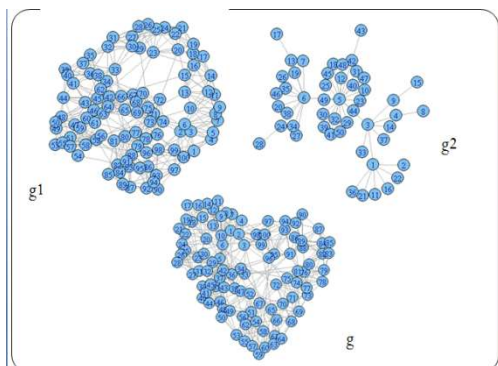
```
>g2 <- barabasi.game(50, p=2, directed=F)
```

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

```
>g <- graph.union(g1,g2)
```

```
#Let's remove multi-edges and loops
```

```
>g <- simplify(g)
```



## Practical No 4

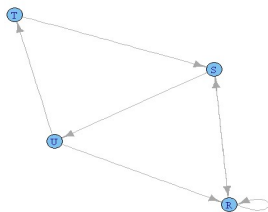
### Aim:

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

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

node to another node;

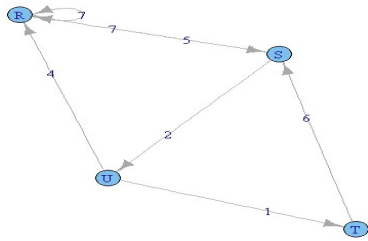
```
> library(igraph)
> matt <- as.matrix(read.table(text=
"node R S T U
  R 7 5 0 0
  S 7 0 0 2
  T 0 6 0 0
  U 4 0 1 0", header=T))
> nms <- matt[,1 ]
> matt <- matt[, -1]
> colnames(matt) <- rownames(matt) <- nms
> matt[is.na(matt)] <- 0
> g <- graph.adjacency(matt, weighted=TRUE)
> plot(g)
```



```
> 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 No 5

### Aim:

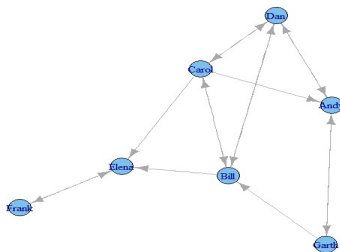
Write a program to distinguish between a network as a matrix, a network as an edge list, and a network as a sociogram (or “network graph”) using 3 distinct networks representatives of each.

#### 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)
```

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 .

#### iii) a network as an edge list.

```
> E(ng)
```

Edge sequence:

```
[1] Andy -> Garth
```

[2] Andy -> Dan  
[3] Garth -> Andy  
[4] Garth -> Bill  
[5] Bill -> Elena  
[6] Bill -> Carol  
[7] Bill -> Dan  
[8] Elena -> Frank  
[9] Frank -> Elena  
[10] Carol -> Andy  
[11] Carol -> Bill  
[12] Carol -> Elena  
[13] Carol -> Dan  
[14] Dan -> Andy  
[15] Dan -> Bill  
[16] Dan -> Carol

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

\$Andy

[1] 3 10 14

\$Garth

[1] 1

\$Bill

[1] 4 11 15

\$Elena

[1] 5 9 12

\$Frank

[1] 8

\$Carol

[1] 6 16

\$Dan

[1] 2 7 13

## Practical No 6

### Aim:

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

#### i) structural equivalence

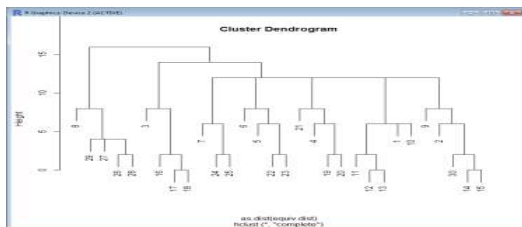
```
> library(sna)
```

```
> library(igraph)
```

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

```
> eq<-equiv.clust(links2)
```

```
> plot(eq)
```

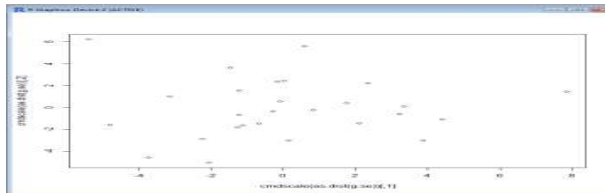


#### ii) automorphic equivalence,

```
>g.se<-sedist(links2)
```

Plot a metric MDS of vertex positions in two dimensions

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

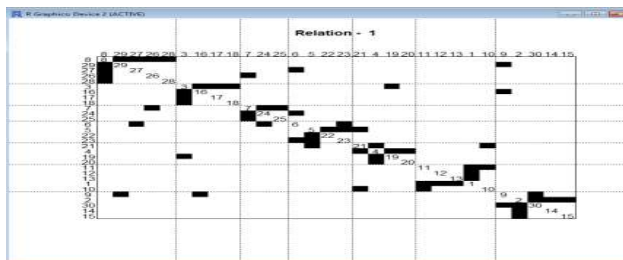


### 3) regular equivalence from a network.

#### Blockmodeling

```
> 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.

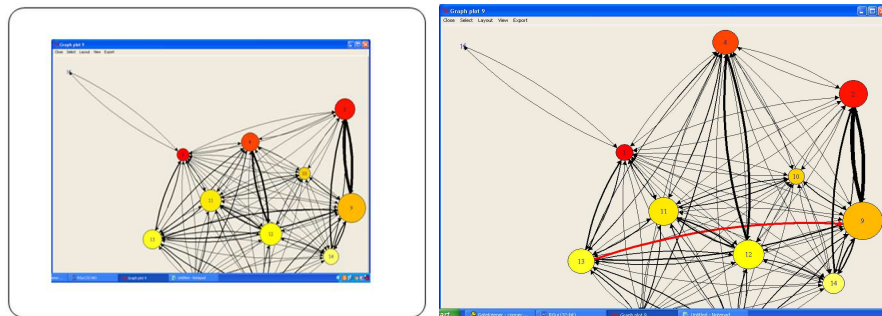
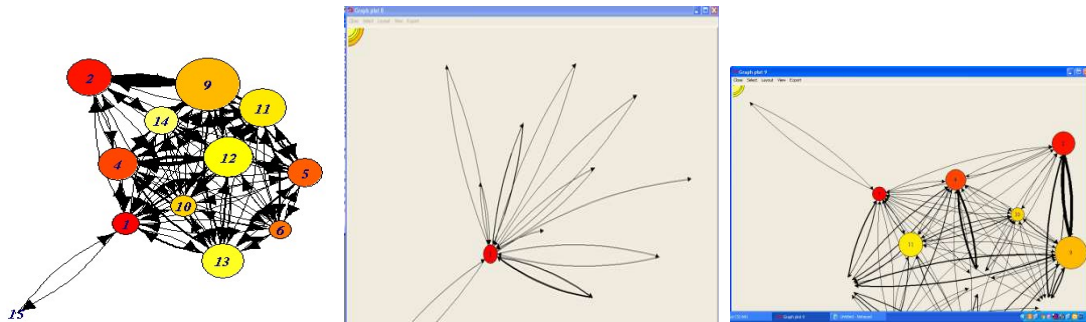
```
>library(Dominance)

>data(data_Network_1)

## set 1 for action you want to show

>bytes= "001111111111000000000"

>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.action	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

## Practical N0 8

### Aim:

### Perform SVD analysis of a network.

```
>library(igraph)

>a <- matrix(c(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, 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 9.687693e-17

$u
      [,1] [,2] [,3] [,4]
[1,] -0.3333333 0.4687136 0.05029703 3.375152e-01
[2,] -0.3333333 0.4687136 0.05029703 -8.126230e-01
[3,] -0.3333333 0.4687136 0.05029703 4.751078e-01
[4,] -0.3333333 -0.2779153 0.38076936 1.160461e-16
[5,] -0.3333333 -0.2779153 0.38076936 1.160461e-16
[6,] -0.3333333 -0.2779153 0.38076936 1.160461e-16
[7,] -0.3333333 -0.1907983 -0.43106639 -7.755807e-17
```



[8,] -0.3333333 -0.1907983 -0.43106639 -7.755807e-17

[9,] -0.3333333 -0.1907983 -0.43106639 -7.755807e-17

\$v

[,1] [,2] [,3] [,4]

[1,] -0.8660254 -2.464364e-17 0.00000000 0.5

[2,] -0.2886751 8.118358e-01 0.08711702 -0.5

[3,] -0.2886751 -4.813634e-01 0.65951188 -0.5

[4,] -0.2886751 -3.304723e-01 -0.74662890 -0.5