BDA Experiment No.: 06

Aim: Social Network Analysis using R (e.g.: Community Detection Algorithm)

Theory:

Social Network Analysis (SNA) is the process of exploring or examining the social structure by using graph theory. It is used for measuring and analyzing the structural properties of the network. It helps to measure relationships and flows between groups, organizations, and other connected entities.

Before we start let us see some network analysis terminology

- A network is represented as a graph, which shows links (if any) between each vertex (or node) and its neighbors.
- A line indicating a link between vertices is called an edge.
- A group of vertices that are mutually reachable by following edges on the graph is called a component.
- The edges followed from one vertex to another are called a path.

The following software is required in order to perform network analysis

- R software
- Packages:
- igraph
- sna (social network analysis)

Functions used in the Social Network Analysis

- **library()function** library() function load and attach add-on packages.
- **Syntax:**
  library(package, help, logical.return = FALSE....)
- **make_full_graph()function**
  This function is used to create a full graph.
- **Syntax:** make_full_graph(n, loops = FALSE, directed = FALSE)
- **make_ring()function**
  A ring is a one-dimensional lattice and it can create lattices of arbitrary dimensions, periodic or non-periodic ones.
- **Syntax:** make_ring(n, directed = FALSE, circular = TRUE, mutual = FALSE)
- **make_star()function**
  This Function creates a star graph, where every single vertex is connected to the center vertex and nobody else.
- **Syntax:**
  make_star(n, center = 1, mode = c("in", "out", "mutual", "undirected"))
- **sample_gnp()function**
  This is a simple model where every possible edge is created with the same constant probability.
- **Syntax:** sample_gnp(n, p, loops = FALSE, directed = FALSE)
- **plot()function**
  This function is used to draw the given graph in the active graphics window.

- **Syntax:** plot(defined_graph_name)

Creating Sample Graphs

Full Graph Syntax:
make_full_graph ()

Parameters:
- Number of vertices.
- directed = TRUE/FALSE Whether to create a directed graph or not.
- loops = TRUE/FALSE Whether to add self-loops to the graph or not.

Analyzing graphs
Connectedness of graph
One of the basic measures of the vertices in a graph is how many connections they have with other vertices. This measure can either be the number of connections to the total possible connections.
Now let us find the degree of each node/vertex in a random graph.

Syntax:
degree(graph)
The degree function is used to find out the number of vertices does each vertex is connected to.
Betweenness of graph
In social networks, betweenness is defined as bridges between and among groups of network members. One way to calculate the betweenness is to calculate the betweenness of each vertex. In general, the higher the betweenness score associated with a vertex, the more control over the network.

Syntax: betweenness(graph) betweenness() `function is defined by the number of shortest paths going through a vertex or an edge.`

Network Density
The Network's density is defined as the number of connections to the total number of possible connections. A complete graph has density = 1 while other networks can have a decimal value.

Syntax: edge_density(graph)

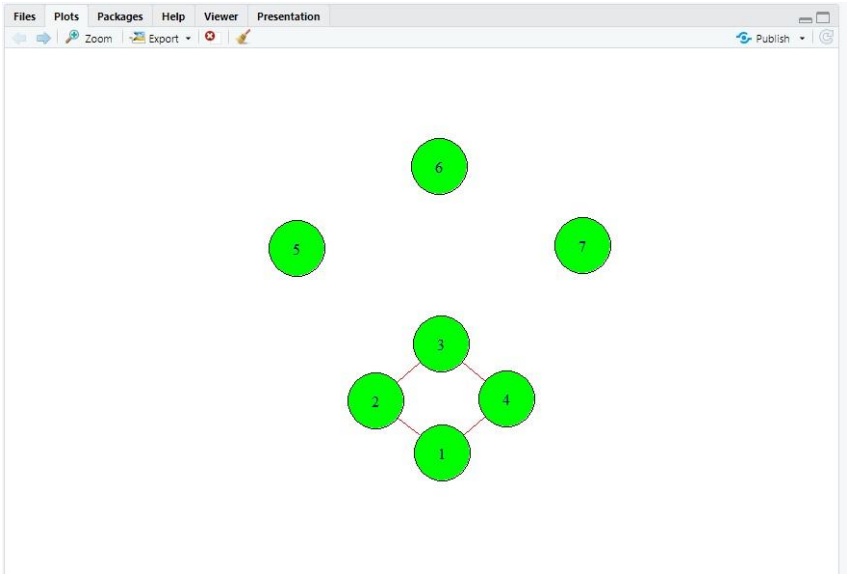
It is the ratio of the number of edges to the total number of possible edges.
Finding components of a graph
A group of connected network vertices is called a component. So it's possible that a can have multiple components that aren't interconnected. **Syntax:** components(graph)
This will calculate the strongly or weakly connected components of a graph. **Program :**

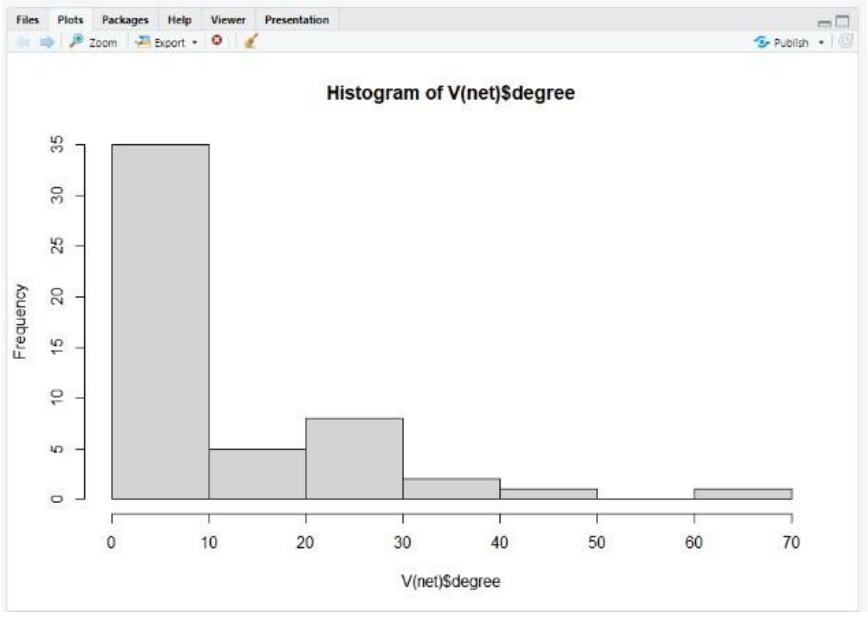
```
# Social Network Analysis library(igraph)
g <- graph(c(1,2,2,3,3,4,4,1),
        directed = F,
n=7) plot(g,
vertex.color = "green",
vertex.size = 40,
    edge.color = 'red')
```

```
g1 <- graph(c("asmit", "Raghu", "Priyanka", "pooja", "pooja", "Asmit",
              "Asmit", "pooja", "Karan", "pooja"),
directed=T)
# Network measures degree(g1,
mode='all') degree(g1,
mode='in')
degree(g1, mode='out')

diameter(g1, directed=F, weights = NA)
edge_density(g1, loops = F)
ecount(g1)/(vcount(g1)*(vcount(g1)-1))
reciprocity(g1) closeness(g1, mode='all',
weights = NA) betweenness(g1,
directed=T, weights=NA)
edge_betweenness(g1, directed=T, weights=NA)

# Read data file
data <- read.csv('https://raw.githubusercontent.com/bkrai/R-files-from-
YouTube/main/networkdata.csv', header=T)
y <- data.frame(data$first, data$second)

# Create network net <-
graph.data.frame(y, directed=T)
V(net)$label <- V(net)$name
V(net)$degree <- degree(net)

# Histogram of node degree
hist(V(net)$degree)

# Network diagram
plot(net)

# Highlighting degrees & layouts
plot(net,      vertex.color =
rainbow(52),      vertex.size =
V(net)$degree*0.4,
edge.arrow.size = 0.1,
    layout=layout.fruchterman.reingold)

# Hub and authorities hs <-
hub_score(net)$vector as <-
authority.score(net)$vector
par(mfrow=c(1,2)) set.seed(222)
plot(net,      vertex.size=hs*30,
main = 'Hubs',      vertex.color =
rainbow(52),
edge.arrow.size=0.1,
```

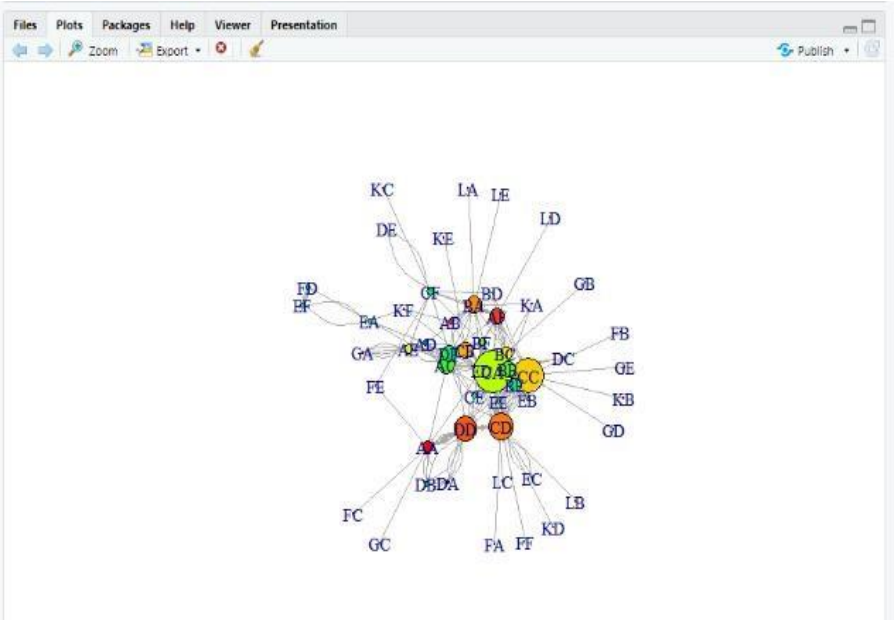
```
    layout = layout.kamada.kawai)

plot(net, vertex.size=as*30, main
= 'Authorities', vertex.color =
rainbow(52), edge.arrow.size=0.1,
layout = layout.kamada.kawai)
par(mfrow=c(1,1)) # Community
detection net <- graph.data.frame(y,
directed = F) cnet <-
cluster_edge_betweenness(net)
plot(cnet)
```

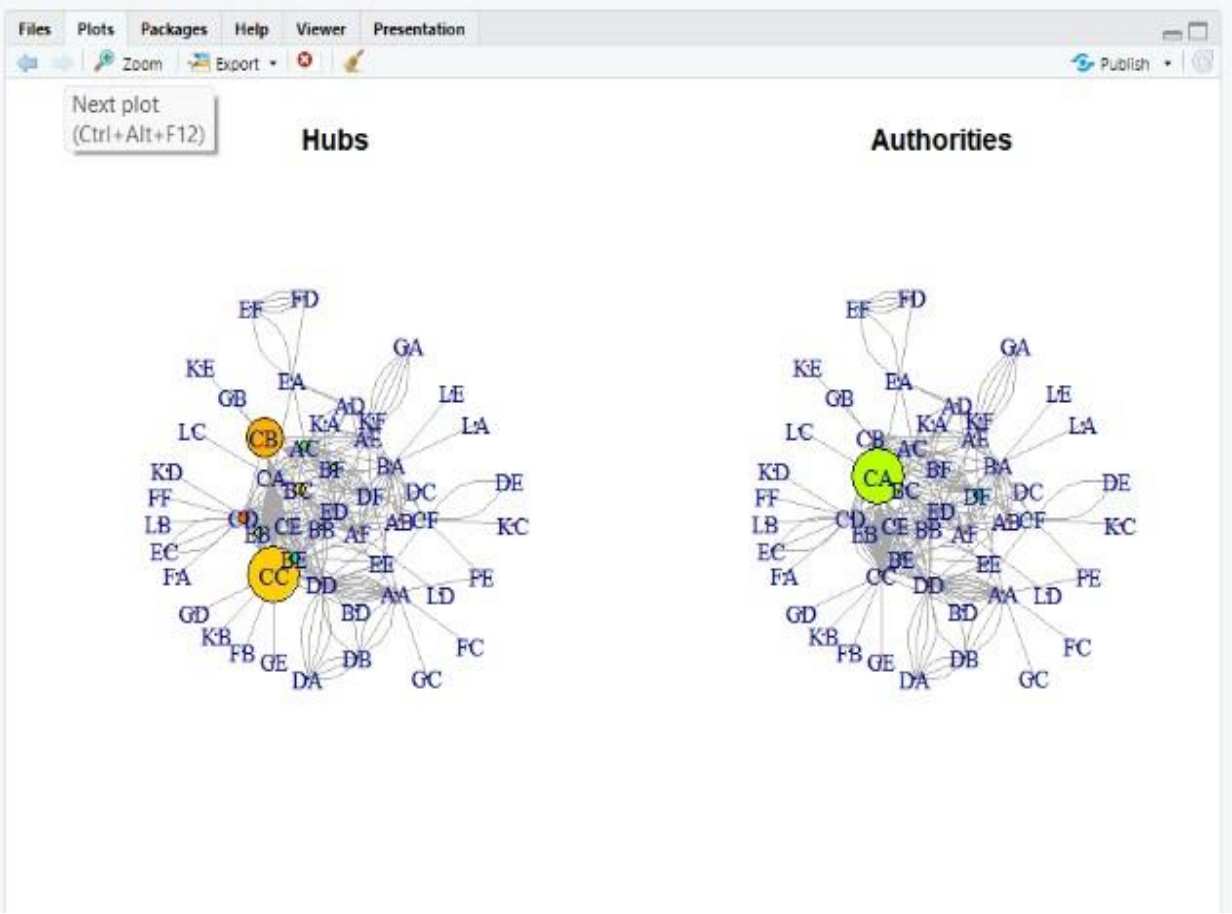
Output:






Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish



Conclusion : Hence , we have successfully implemented experiment on Social Network Analysis using R .