

# INDEX

1. SOCIAL NETWORK: INTRODUCTION

2. ABOUT

- i. THE PROJECT
- ii. THE CODE

3. CODING FOR THE PROGRAM

4. OUTPUT OF THE PROGRAM

- i. SIMPLE SOCIAL NETWORK GRAPH
- ii. NETWORK GRAPH
- iii. HUBS GRAPH
- iv. AUTHORITY GRAPH
- v. CLUSTER GRAPH
- vi. HISTOGRAM

---

# SOCIAL NETWORK ANALYSIS

---

**Network:** A network is simply a graph structure having number of vertices (nodes) that are connected by links or edges.

**Social Network:** A social network is a social structure represented as a graph, where the vertices are individuals or organizations, and the links are interdependencies between the vertices, representing social connection as friendship, common interests, marital/family ties or collaborative activities.



## ABOUT THE PROJECT

Social Network Analysis is a set of methods used to visualize networks, describe specific characteristics of overall network structure, and build mathematical and statistical models of network structures and dynamics. Social networks are formally defined as a set of nodes that are tied by one or more types of relations.

It is possible to analyze and study social networks in RStudio.

---

In this project we will be using a small network that indicates interactions in the famous *Sanskrit epic Ramayana*.

Here, each node is a character and each edge indicates the interactions of one character with another character in the epic.

# ABOUT THE CODE

---

- IMPORTING NETWORK DATA INTO R

The first step is to read the data in this network with `read.csv` command.

- HOW DO WE CONVERT THIS DATASET INTO A NETWORK OBJECT IN R?

There are multiple packages to work with networks, but the most popular is *igraph* library because it's very flexible and easy to do.

- NETWORK VISUALIZATION

How can we visualize this network? The `plot()` function works out of the box, but the default options are often not ideal.

- MODIFICATIONS IN PLOTTING ATTRIBUTES

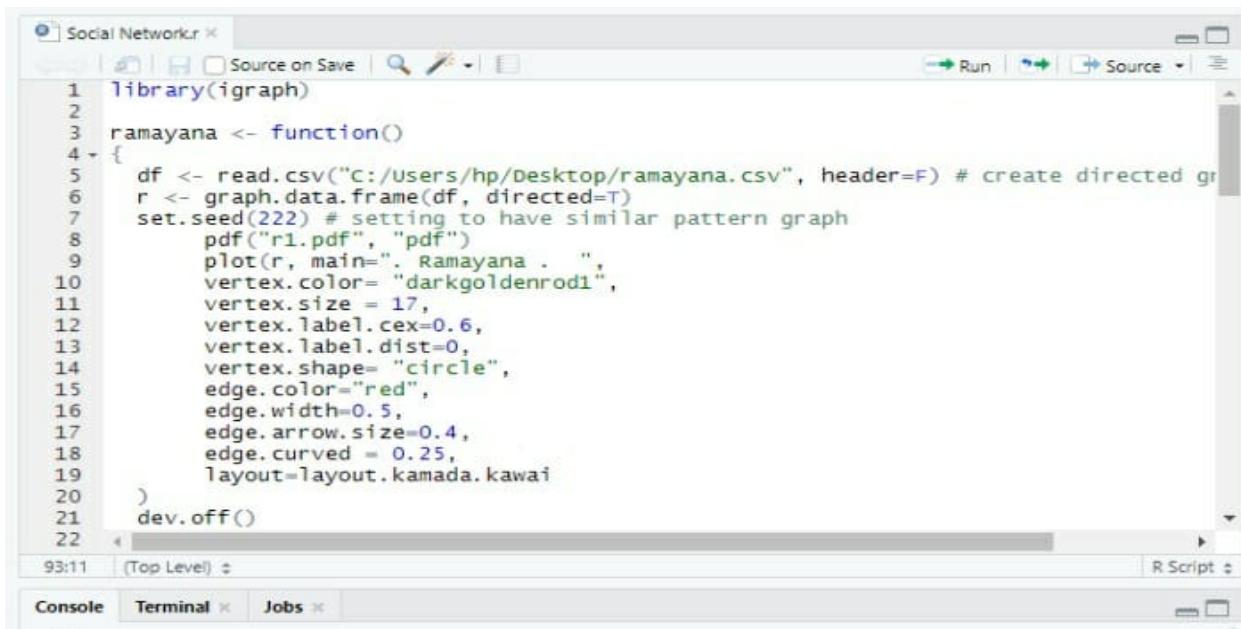
Common adjustments such as changing size/color/border of node or edges or labels can be made to show correspondence with each other in RStudio.

We can also specify the layout for the plot; that is, the (x,y) coordinates where each node will be placed. *igraph* has a few different layouts built-in, that uses different algorithms to find an optimal distribution of nodes.

## CODING SECTION

---

// CODING FOR SIMPLE SOCIAL NETWORK FOR RAMAYANA CHARACTERS



The screenshot shows the RStudio interface with the following code in the script pane:

```
1 library(igraph)
2
3 ramayana <- function()
4 {
5   df <- read.csv("C:/users/hp/desktop/ramayana.csv", header=F) # create directed graph
6   r <- graph.data.frame(df, directed=T)
7   set.seed(222) # setting to have similar pattern graph
8   pdf("r1.pdf", "pdf")
9   plot(r, main=". Ramayana . .",
10        vertex.color= "darkgoldenrod1",
11        vertex.size = 17,
12        vertex.label.cex=0.6,
13        vertex.label.dist=0,
14        vertex.shape= "circle",
15        edge.color="red",
16        edge.width=0.5,
17        edge.arrow.size=0.4,
18        edge.curved = 0.25,
19        layout=layout.kamada.kawai
20   )
21 dev.off()
22 }
```

The code defines a function `ramayana` that reads a CSV file, creates a directed graph, and plots it as a PDF. The plot uses red edges and vertices colored `darkgoldenrod1`.

// CODING FOR PLOTTING GRAPH 2

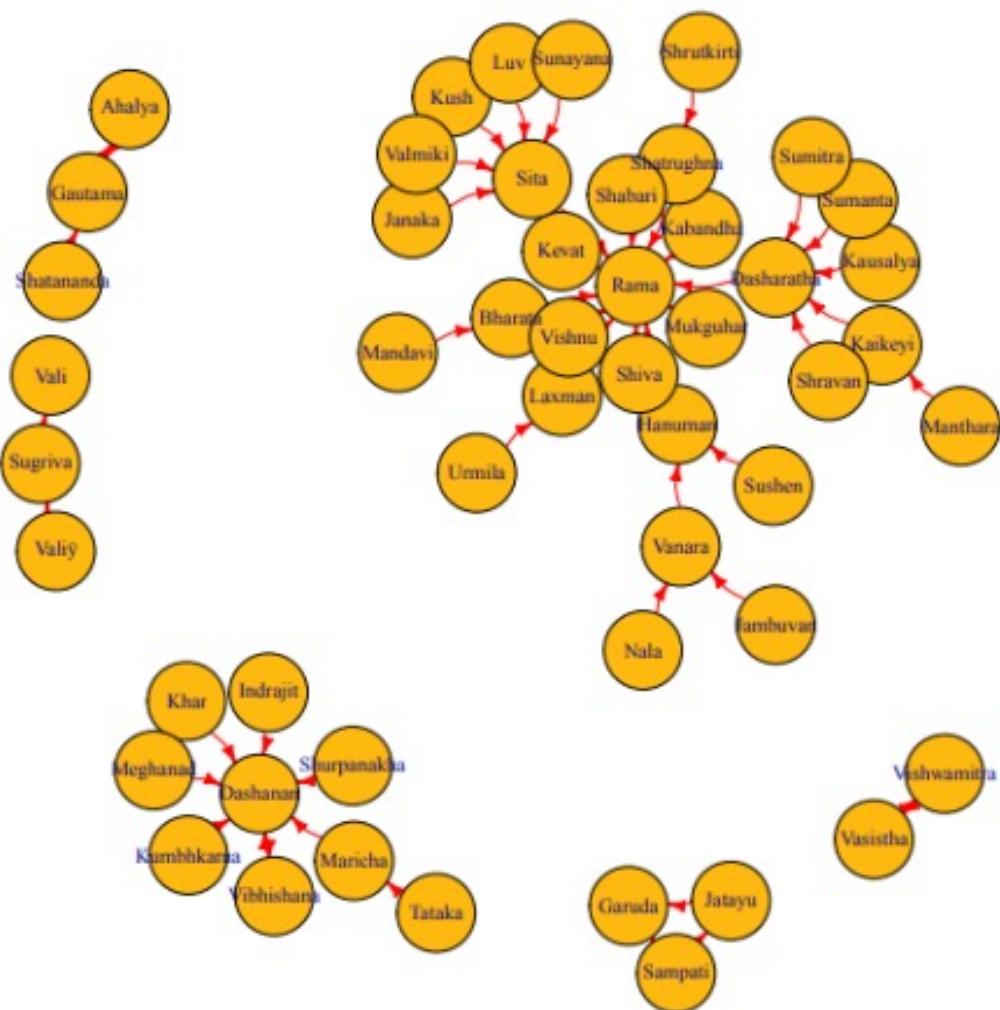
Social Networker x

Source on Save | Run | Source

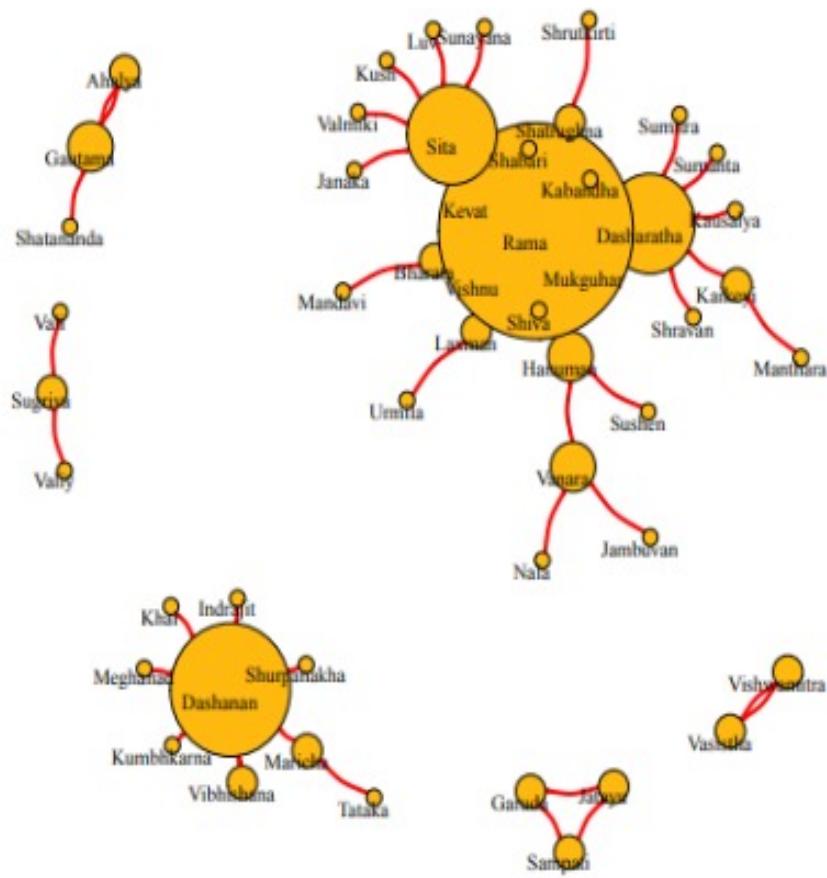
```
21 dev.off()
22 pdf("r2.pdf", "pdf")
23 plot(r, main="Ramayana",
24       vertex.color="darkgoldenrod1",
25       vertex.frame.color="black",
26       vertex.size = degree(r)*4, # size as per degree
27       vertex.label.cex=0.6,
28       vertex.label.dist=-0.55,
29       vertex.label.color="black",
30       edge.color="red",
31       edge.width=2,
32       edge.arrow.size=0.1,
33       edge.curved = 0.25,
34       layout=layout.kamada.kawai
35     )
36 dev.off()
37
```

## OUTPUT FOR SIMPLE NETWORK GRAPH

### . Ramayana .



## Ramayana



// CODING FOR PLOTTING HUB GRAPH

The screenshot shows the RStudio interface with a code editor window titled "Social Networker". The code is written in R and plots a hub graph. It starts by calling `dev.off()`, then creates a layout using `layout.circle(r)`. It prints the layout and extracts hub scores from `hub_score(r)$vector` and authority scores from `authority_score(r)$vector`. It then opens a PDF file named "r3.pdf" and creates a 2x1 grid plot. The main title is "Hubs". The plot uses orange vertices, gray labels, and red edges. The vertex size is proportional to its hub score. The labels are positioned below the vertices. The edges are thin red lines. The layout is generated using the Kamada-Kawai algorithm. Finally, it calls `dev.off()`.

```
36 dev.off()
37 l <- layout.circle(r)
38 print(l)
39 hs <- hub_score(r)$vector
40 as <- authority_score(r)$vector|
41
42 pdf("r3.pdf", "pdf")
43 par(mfrow=c(1,2))
44 plot(r, main="Hubs",
45       vertex.color="orange",
46       vertex.label.color="gray2",
47       vertex.size = hs*30,
48       # size as per degree
49       vertex.label.cex=0.3,
50       vertex.label.dist=-0.7,
51       edge.color="red",
52       edge.arrow.size=0.1,
53       layout=layout.kamada.kawai
54 )
55 dev.off()
56
```

//CODING FOR PLOTTING AUTHORITY GRAPH

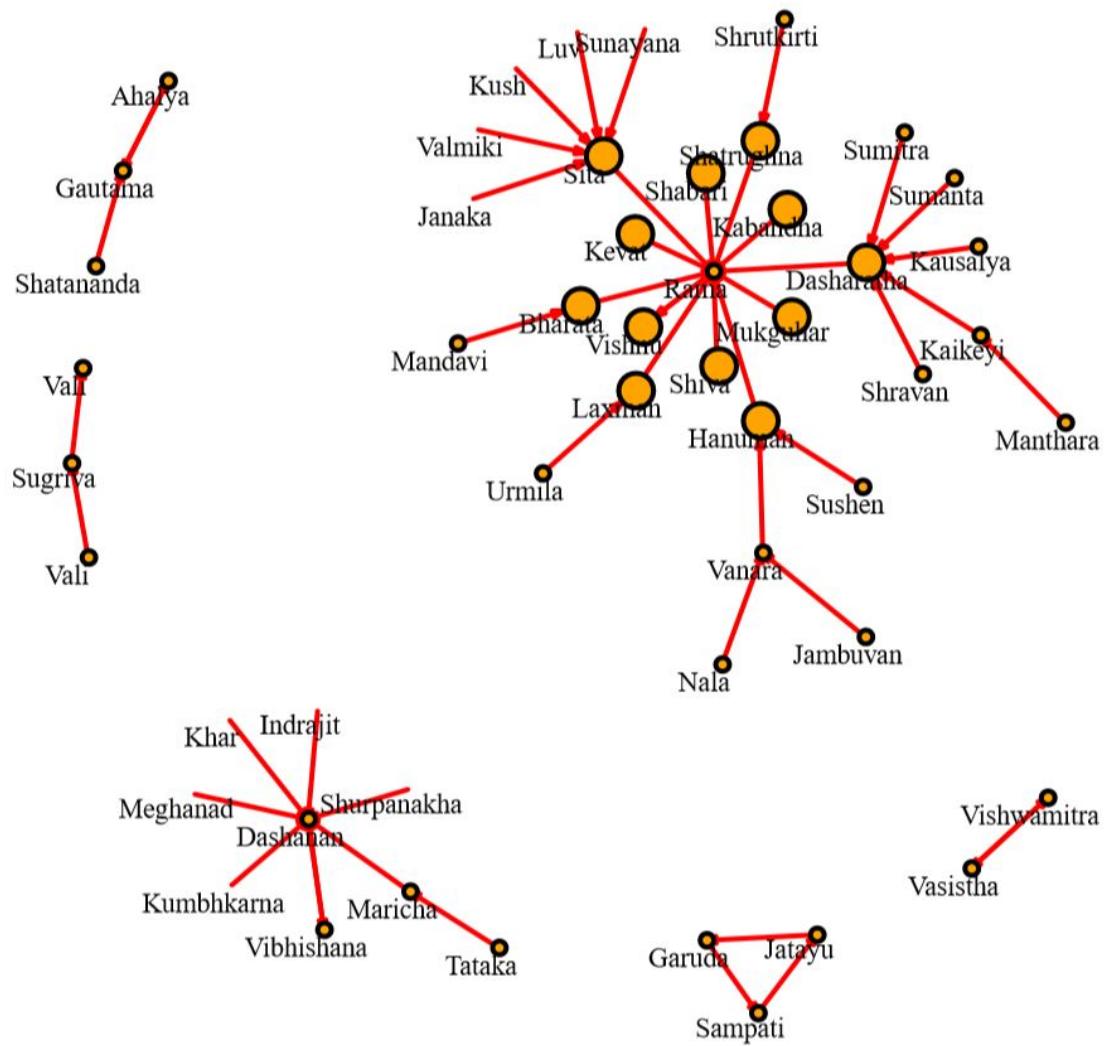
The screenshot shows the RStudio interface with a code editor window titled "Social Networker". The code is written in R and plots an authority graph. It starts by calling `dev.off()`, then opens a PDF file named "r4.pdf". The main title is "Authority". The plot uses rainbow-colored vertices, gray labels, and red edges. The vertex size is proportional to its authority score. The labels are positioned below the vertices. The edges are thin red lines. The layout is generated using the Kamada-Kawai algorithm. Finally, it calls `dev.off()`.

```
55 dev.off()
56 pdf("r4.pdf", "pdf")
57 plot(r, main="Authority",
58       vertex.color=rainbow(15),
59       vertex.size = as*30, # size as per degree
60       vertex.label.cex=0.6,
61       vertex.label.dist=0.5,
62       vertex.label.color="gray2",
63       edge.color="red",
64       edge.arrow.size=0.1,
65       layout=layout.kamada.kawai
66 )
67 dev.off()
68
```

---

## OUTPUT FOR HUB GRAPH

---

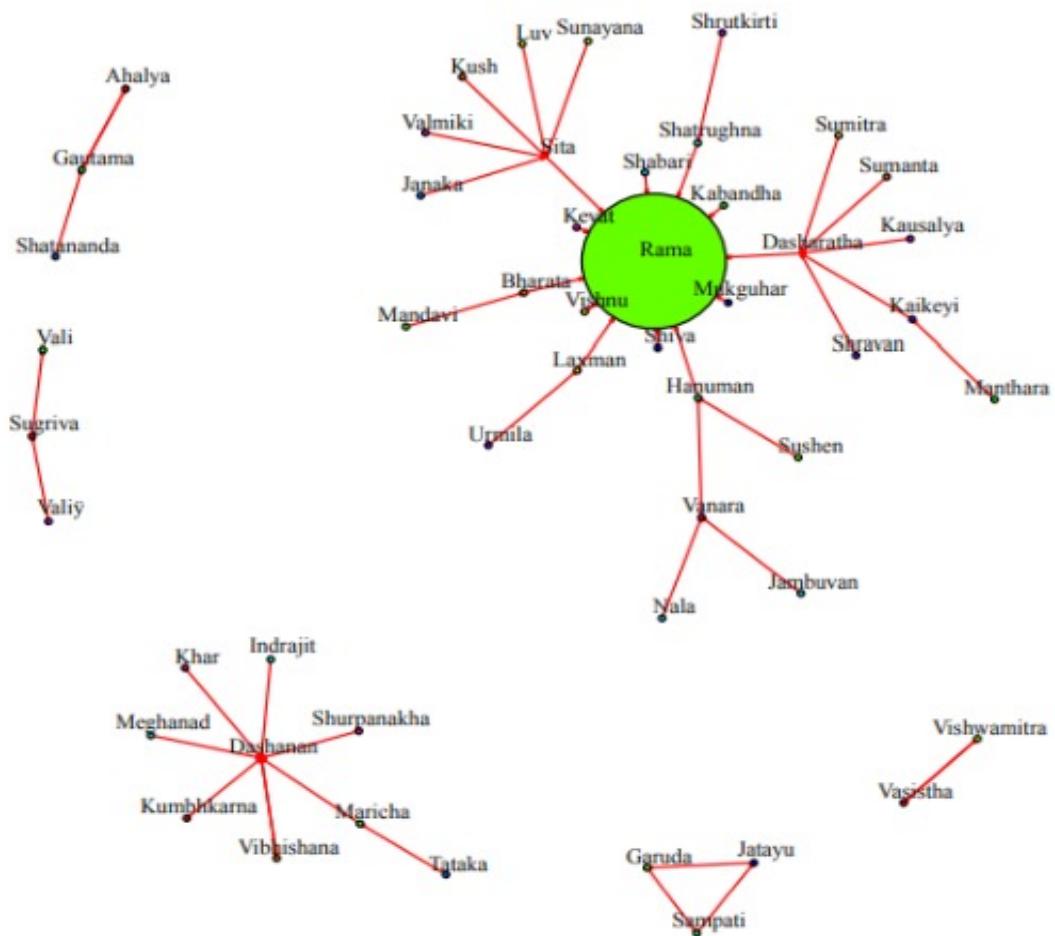


---

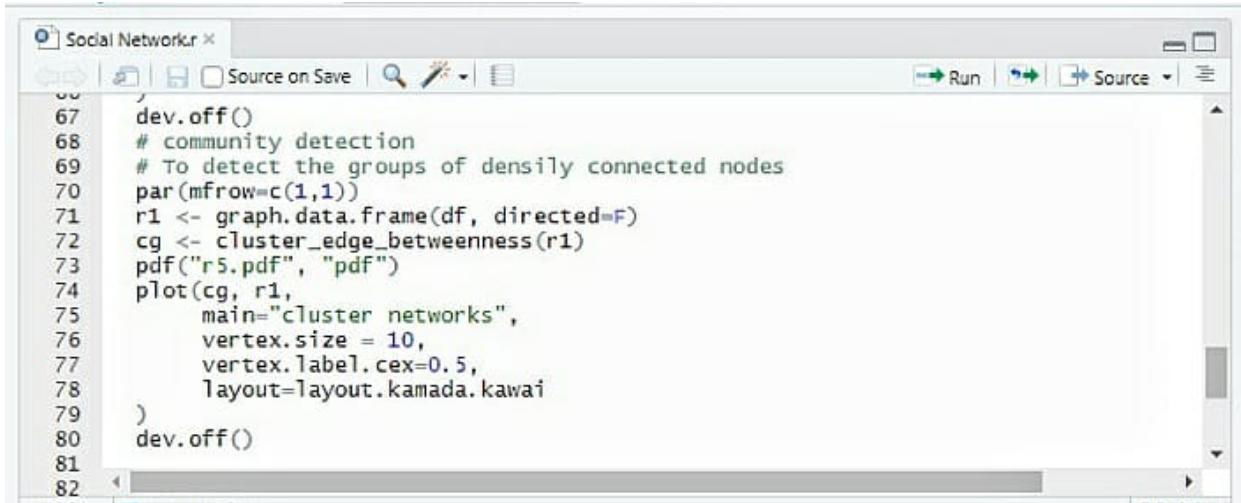
## OUTPUT FOR AUTHORITY GRAPH

---

### Authority



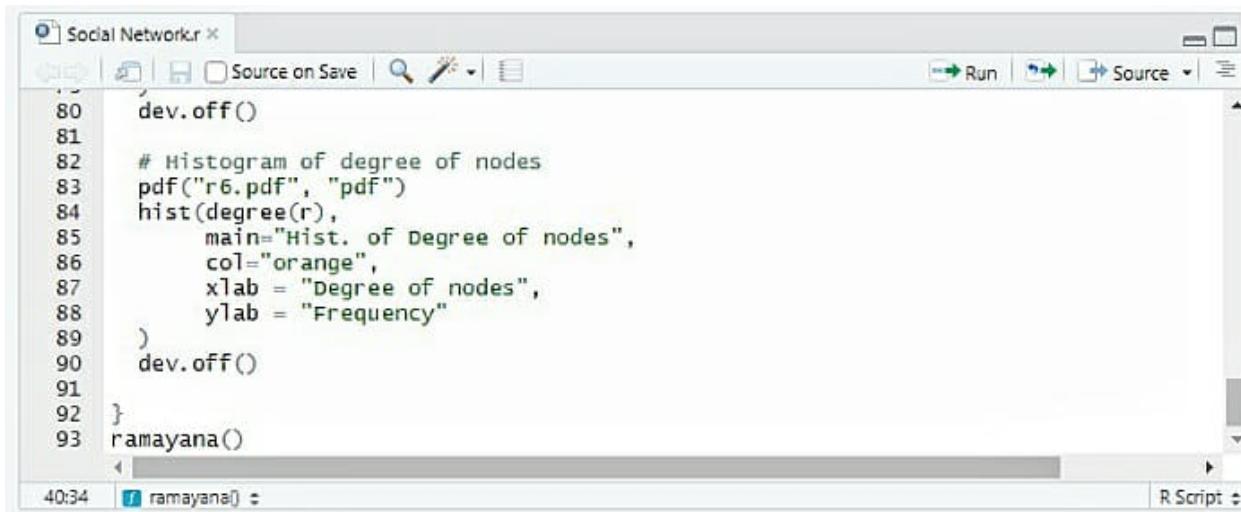
// CODING FOR PLOTTING A CLUSTER GRAPH :



The screenshot shows the RStudio interface with a script editor window titled "Social Network.r". The code is as follows:

```
67 dev.off()
68 # community detection
69 # To detect the groups of densely connected nodes
70 par(mfrow=c(1,1))
71 r1 <- graph.data.frame(df, directed=F)
72 cg <- cluster_edge_betweenness(r1)
73 pdf("r5.pdf", "pdf")
74 plot(cg, r1,
75      main="cluster networks",
76      vertex.size = 10,
77      vertex.label.cex=0.5,
78      layout=layout.kamada.kawai
79 )
80 dev.off()
81
82
```

//CODING FOR PLOTTING A HISTOGRAM



The screenshot shows the RStudio interface with a script editor window titled "Social Network.r". The code is as follows:

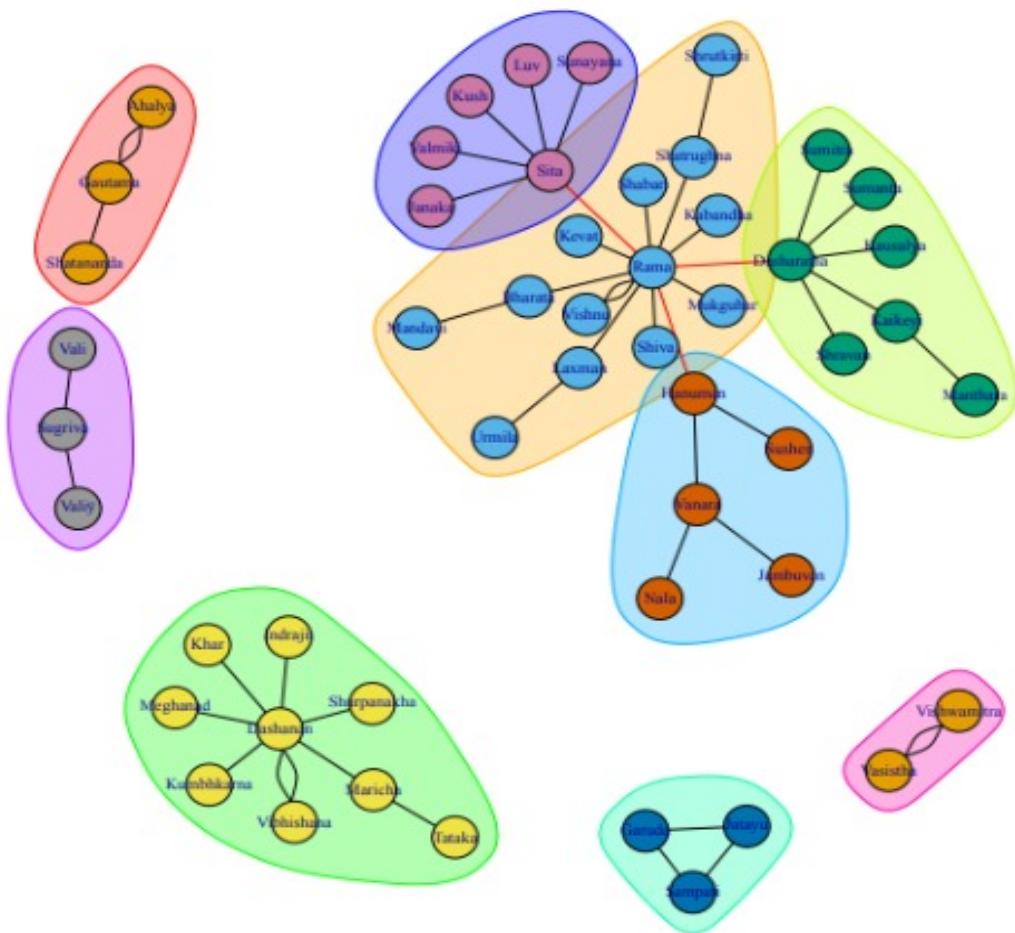
```
80 dev.off()
81
82 # Histogram of degree of nodes
83 pdf("r6.pdf", "pdf")
84 hist(degree(r),
85       main="Hist. of Degree of nodes",
86       col="orange",
87       xlab = "Degree of nodes",
88       ylab = "Frequency"
89 )
90 dev.off()
91
92 }
93 ramayana()
```

---

## OUTPUT FOR CLUSTER GRAPH

---

### cluster networks



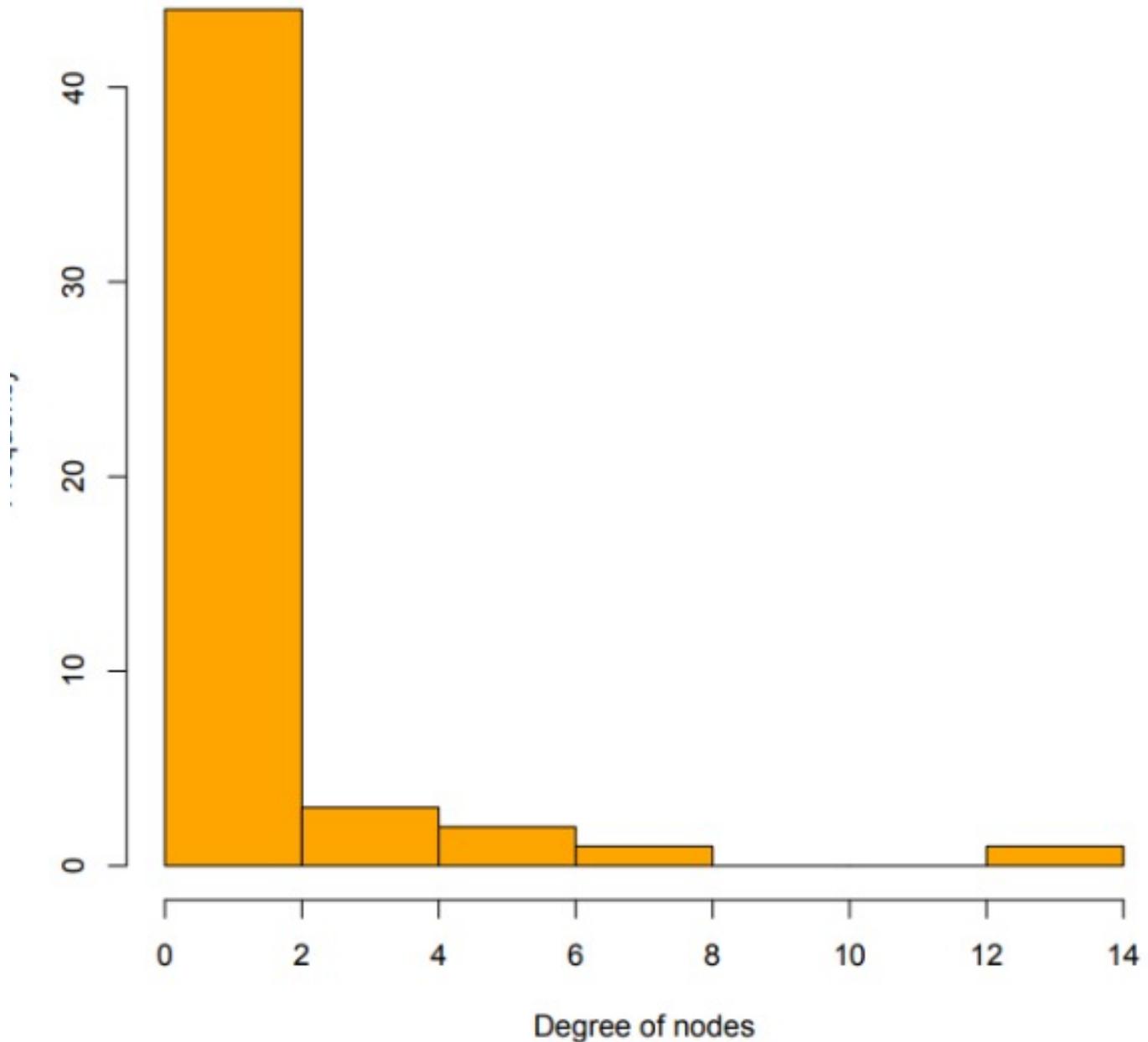
---

---

HISTOGRAM OUTPUT

---

**Hist. of Degree of nodes**





*THANKYOU*