Social Network Analysis

Objective: Social Network Analysis (SNA) package using R for Social Network analysis as part of Lab Migration Project.

Methods:

(i) Import and load the dataset and view information about it using str() function.
(ii) Create the graph network using graph.data.frame() function.
(iii) Plot the histogram graph for node degrees.
(iv) Plot the network graph diagram.
(v) Compute the hub and authority scores and depict the vertices with maximum hub and authority in the network diagram.
(vi) Find out the communities present in the network.
(vii) Conclusion

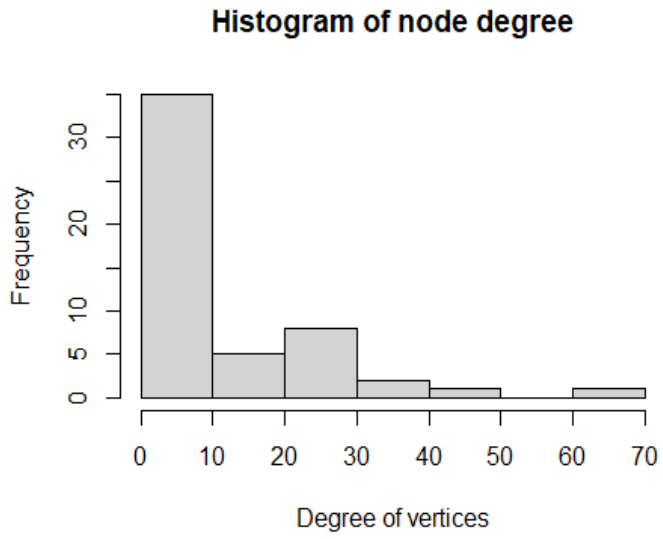
```
#To clear the environment
rm(list=ls())

#Import the required libraries
library(sna)

## Warning: package 'sna' was built under R version 4.1.3

## Loading required package: statnet.common

## Warning: package 'statnet.common' was built under R version 4.1.3

##
## Attaching package: 'statnet.common'

## The following objects are masked from 'package:base':
##
##     attr, order

## Loading required package: network

## Warning: package 'network' was built under R version 4.1.3

##
## 'network' 1.17.2 (2022-05-20), part of the Statnet Project
## * 'news(package="network")' for changes since last version
```

```
## * 'citation("network")' for citation information
## * 'https://statnet.org' for help, support, and other information

## sna: Tools for Social Network Analysis
## Version 2.7 created on 2022-05-09.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##  For citation information, type citation("sna").
##  Type help(package="sna") to get started.

library(igraph)

## Warning: package 'igraph' was built under R version 4.1.3

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:sna':
##
##     betweenness, bonpow, closeness, components, degree, dyad.census,
##     evcent, hierarchy, is.connected, neighborhood, triad.census

## The following objects are masked from 'package:network':
##
##     %c%, %s%, add.edges, add.vertices, delete.edges, delete.vertices,
##     get.edge.attribute, get.edges, get.vertex.attribute, is.bipartite,
##     is.directed, list.edge.attributes, list.vertex.attributes,
##     set.edge.attribute, set.vertex.attribute

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

#Import and load the dataset
d <- read.csv('socialnetworkdata.csv')
data <- data.frame(d$first,d$second)
str(data)

## 'data.frame':    290 obs. of  2 variables:
##  $ d.first : chr  "AA" "AB" "AF" "DD" ...
##  $ d.second: chr  "DD" "DD" "BA" "DA" ...

#To create the network
network <- graph.data.frame(data)
V(network)  #vertices

## + 52/52 vertices, named, from e5167af:
##  [1] AA AB AF DD CD BA CB CC BC ED AE CA EB BF BB AC DC BD DB CF DF BE EA
CE EE
```

```
## [26] EF FF FD GB GC GD AD KA KF LC DA EC FA FB DE FC FE GA GE KB KC KD KE
LB LA
## [51] LD LE

E(network)  #edges

## + 290/290 edges from e5167af (vertex names):
##   [1] AA->DD AB->DD AF->BA DD->DA CD->EC DD->CE CD->FA CD->CC BA->AF CB-
>CA
##  [11] CC->CA CD->CA BC->CA DD->DA ED->AD AE->AC AB->BA CD->EC CA->CC EB-
>CC
##  [21] BF->CE BB->CD AC->AE CC->FB DC->BB BD->CF DB->DA DD->DA DB->DD BC-
>AF
##  [31] CF->DE DF->BF CB->CA BE->CA EA->CA CB->CA CB->CA CC->CA CD->CA BC-
>CA
##  [41] BF->CA CE->CA AC->AD BD->BE AE->DF CB->DF AC->DF AA->DD AA->DD AA-
>DD
##  [51] CD->DD AA->DD EE->DD CD->DD DB->AA AA->FC BE->CC EF->FD CF->FE BB-
>DD
##  [61] CD->DD BA->AB CD->EC BE->EE CE->CC CD->CC ED->CC BB->CC BE->CE DD-
>CE
##  [71] AC->CD ED->CD FF->CD AC->CD DD->CD DD->CD AE->GA AE->GA AE->GA AE-
>GA
##  [81] BA->ED BE->ED EB->ED CD->ED FD->EF FD->EF CD->BB BF->BB BC->BB BB-
>CF
##  [91] AE->AC DD->DA BE->CA BE->CA CB->CA CB->CA CC->CA BE->CC BE->CC DB-
>DD
## + ... omitted several edges

degree(network)  #degree of each node in graph

## AA AB AF DD CD BA CB CC BC ED AE CA EB BF BB AC DC BD DB CF DF BE EA CE EE
EF
## 18  9 23 36 40 26 24 50 21 27 15 62  7 12 23 27  2  4  8 12 23 20  8 10  6
8
## FF FD GB GC GD AD KA KF LC DA EC FA FB DE FC FE GA GE KB KC KD KE LB LA LD
LE
##  1  8  1  1  1  9  3  3  1  7  3  1  1  2  1  2  5  1  1  1  1  1  1  1  1
1

betweenness(network)  #betweenness of each node

##         AA         AB         AF         DD         CD         BA
CB
##  89.196239   4.564734 109.762225 162.990640 375.992047 184.496219
36.237779
##         CC         BC         ED         AE         CA         EB
BF
## 225.269107  94.821412 343.016093 105.368424 292.157605   0.000000
28.446981
##         BB         AC         DC         BD         DB         CF
```

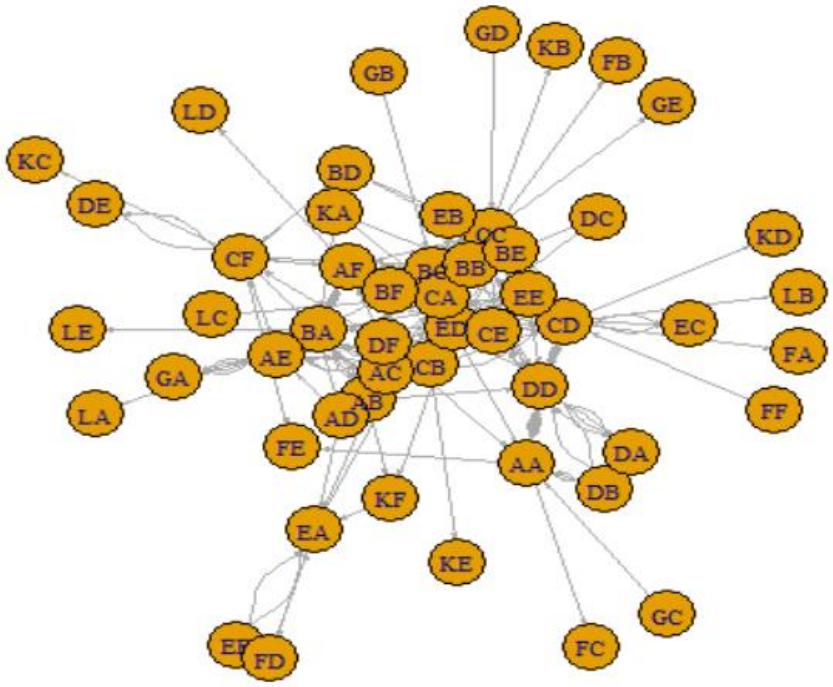
```
DF
## 244.963181 172.427422   0.000000   5.966115   0.000000 181.814414
42.768995
##          BE          EA          CE          EE          EF          FF
FD
##  55.964872 183.604022  11.943347   4.002137   0.000000   0.000000
33.000000
##          GB          GC          GD          AD          KA          KF
LC
##   0.000000   0.000000   0.000000  48.996857   0.000000  17.229134
0.000000
##          DA          EC          FA          FB          DE          FC
FE
##   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
0.000000
##          GA          GE          KB          KC          KD          KE
LB
##   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
0.000000
##          LA          LD          LE
##   0.000000   0.000000   0.000000
```

```
V(network)$label <- V(network)$name  #using names as labels
V(network)$degree <- degree(network)
#To plot the histogram of node degree
hist(V(network)$degree,main='Histogram of node degree',xlab='Degree of
vertices',ylab='Frequency')
```



Inference: From the graph, we can infer that nearly 35 nodes have degrees less than 10. There are few nodes with a high degree of 60-70 connections.

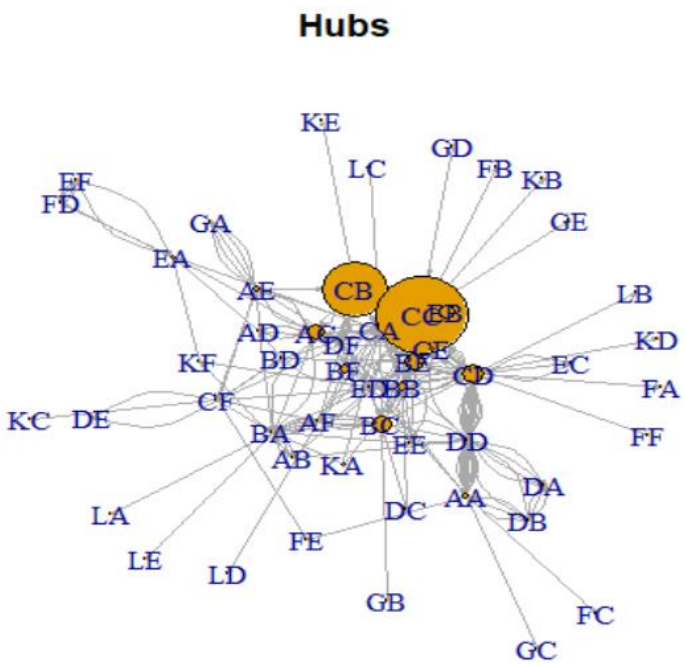
```
#Plotting the network diagram
plot(network,vertex.label.cex=0.7,vertext.size=2,edge.arrow.size=0.1)
```




```
#To calculate the Hub score and Authority score
hs <- hub_score(network)$vector
hs
```

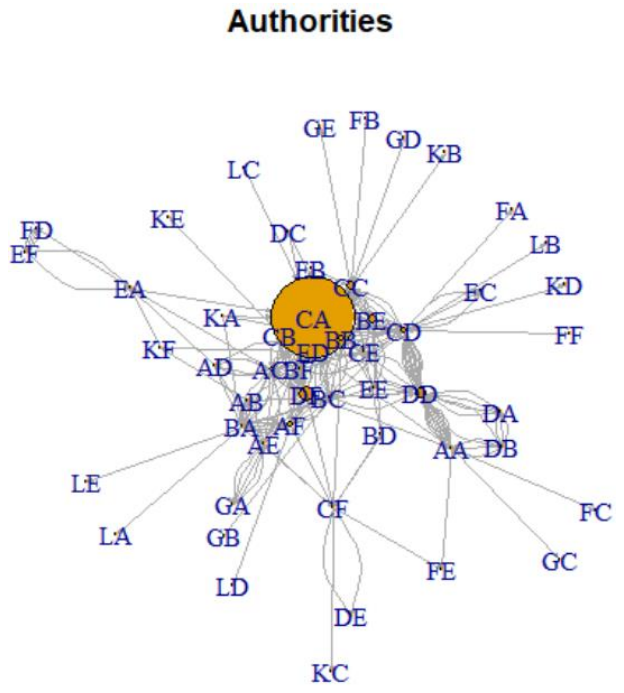
```
##           AA           AB           AF           DD           CD
BA
## 8.042807e-02 1.300950e-02 4.731440e-03 1.255496e-02 2.350165e-01
3.239013e-02
##           CB           CC           BC           ED           AE
CA
## 7.145888e-01 1.000000e+00 2.321828e-01 4.858357e-02 7.949708e-02
4.494956e-02
##           EB           BF           BB           AC           DC
BD
## 1.870058e-01 1.272314e-01 1.150808e-01 2.093518e-01 7.455515e-03
1.305960e-02
##           DB           CF           DF           BE           EA
CE
## 1.738176e-02 7.102816e-03 2.505787e-02 2.195271e-01 5.693409e-02
6.319085e-02
##           EE           EF           FF           FD           GB
GC
## 1.273824e-02 1.119098e-03 4.485458e-03 4.563724e-04 3.594258e-04
2.507046e-04
##           GD           AD           KA           KF           LC
DA
## 6.374463e-03 9.643170e-03 6.276463e-02 4.418478e-04 5.681639e-02
8.403409e-17
##           EC           FA           FB           DE           FC
```

```
FE
## 3.267993e-17 1.167140e-17 1.167140e-17 2.334280e-17 1.167140e-17
2.334280e-17
##           GA           GE           KB           KC           KD
KE
## 4.668561e-17 1.167140e-17 1.167140e-17 1.167140e-17 1.167140e-17
1.167140e-17
##           LB           LA           LD           LE
## 1.167140e-17 1.167140e-17 1.167140e-17 1.167140e-17

as <- authority.score(network)$vector
as

##           AA           AB           AF           DD           CD
BA
## 4.412541e-03 3.968841e-03 8.375004e-02 1.409698e-01 7.894656e-02
2.146204e-02
##           CB           CC           BC           ED           AE
CA
## 3.660444e-03 1.121941e-01 6.326094e-03 4.799095e-02 1.524499e-02
1.000000e+00
##           EB           BF           BB           AC           DC
BD
## 5.863552e-17 6.072030e-03 8.323026e-02 6.262068e-02 1.832360e-17
2.486679e-04
##           DB           CF           DF           BE           EA
CE
## 7.329440e-17 8.146610e-03 1.925650e-01 1.095155e-01 7.776767e-03
2.437300e-02
##           EE           EF           FF           FD           GB
GC
## 2.419075e-02 6.390996e-05 9.161800e-18 2.071608e-03 9.161800e-18
9.161800e-18
##           GD           AD           KA           KF           LC
DA
## 9.161800e-18 1.806051e-02 2.565304e-17 2.156359e-03 9.161800e-18
3.583779e-03
##           EC           FA           FB           DE           FC
FE
## 2.468362e-02 8.227873e-03 3.500976e-02 4.973358e-04 2.815768e-03
3.064436e-03
##           GA           GE           KB           KC           KD
KE
## 1.391587e-02 3.500976e-02 3.500976e-02 2.486679e-04 8.227873e-03
2.501759e-02
##           LB           LA           LD           LE
## 8.227873e-03 1.133971e-03 1.656466e-04 1.133971e-03

plot(network,vertex.size=hs*30,main='Hubs',edge.arrow.size=0.1,layout=layout.
fruchterman.reingold)
```



Inference: CC has the maximum outgoing links, followed by CB.

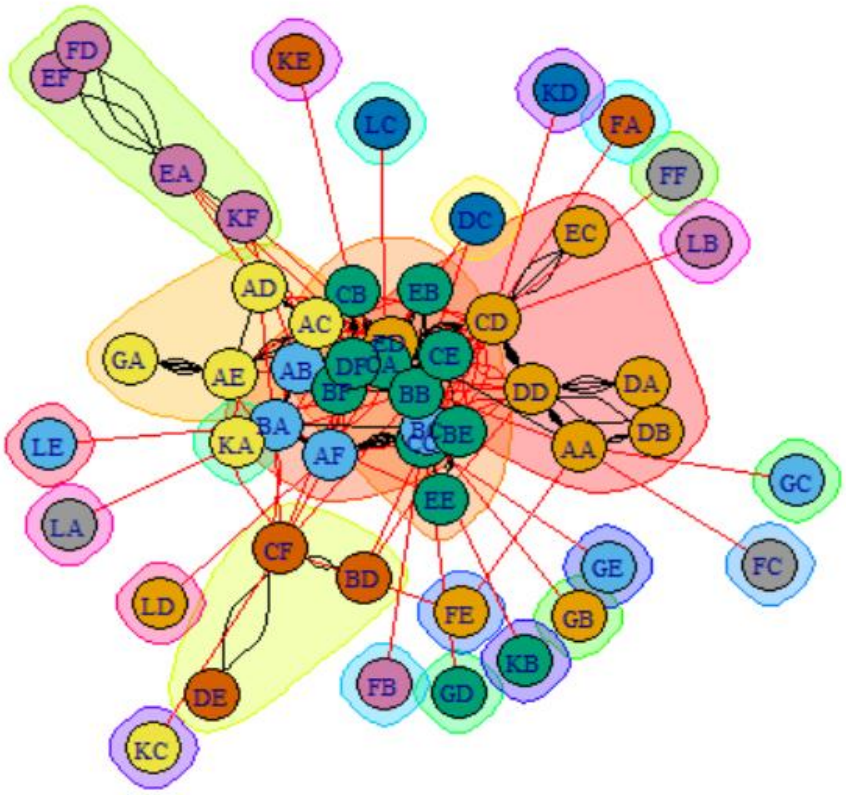
```
plot(network,vertex.size=as*30,main='Authorities',edge.arrow.size=0.1,layout=
layout.fruchterman.reingold)
```



Inference: CA has the maximum incoming links from hubs.

```
#To detect groups of densely connected nodes known as communities
network <- graph.data.frame(data,directed=F)
```

```
cnetwork <- cluster_edge_betweenness(network)
plot(cnetwork,network,vertex.label.cex=0.7)
```



Inference: From the plot, we can infer different groups in this network. We can observe dense connections within the group, while there are sparse connections between the groups.

Conclusion:

The sna and igraph packages have been explored for social network analysis using R. igraph is a package for network analysis and visualization, and sna is a package with a variety of tools for social network analysis. The dataset used included network information from different nodes. The hub and authority scores of each node were calculated and displayed as a network diagram. The network's communities were found, which revealed that nodes inside groups were well connected while those between groups were only loosely connected.