SOCIAL NETWORK ANALYSIS

ADV.R PROJECT

3501 Dhruvi Swadia

3506 Aishani Anavkar

3509 Sandesh Kotwal

INDEX

INTRODUCTION

Social Network Analysis (SNA) is the process of exploring or examining the social structure by using graph theory. It is used for measuring and analysing the structural properties of the network. It helps to measure relationships and flows between groups, organizations, and other connected entities. SNA is the process of investigating social structures through the use of networks and graph theory. It characterizes networked structures in terms of nodes (individual actors, people, or things within the network) and the ties, edges, or links (relationships or interactions) that connect them.

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 neighbours.

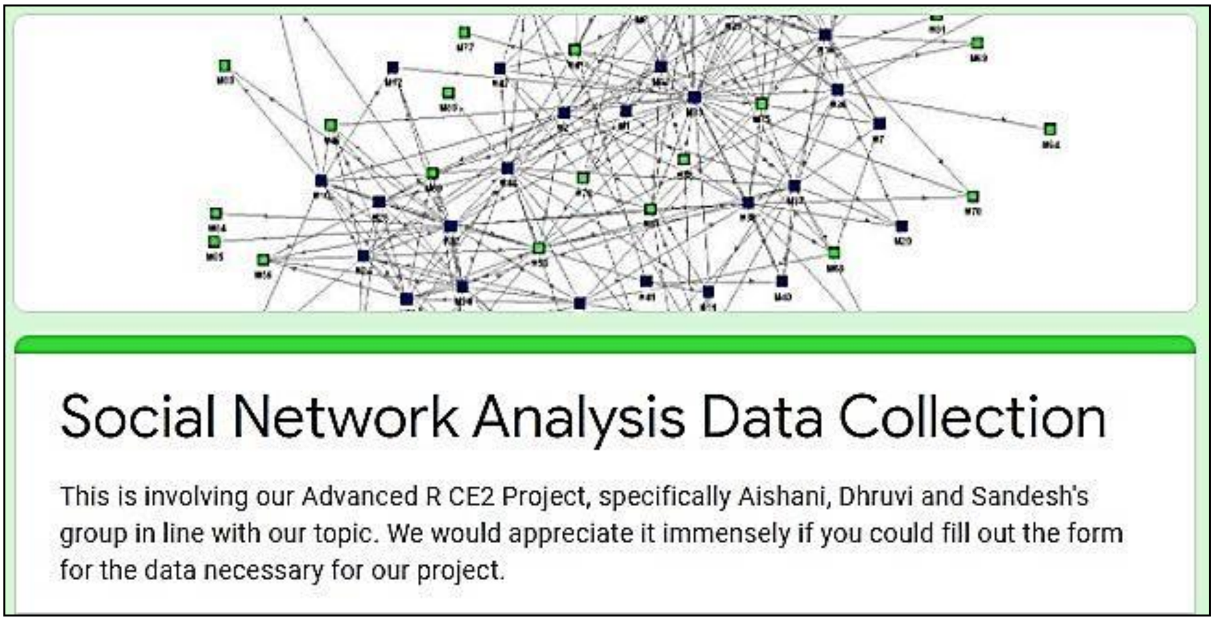
A line indicating a link between vertices is called an edge.

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.

DATA COLLECTION

Data – we hear so much about it, but do we really understand the importance of data collection? At its most basic, data is simply a collection of different facts, including numbers, measurements, and observations, that have been translated into a form that computers can process. This might sound easy, but data is effectively changing the world we live in and the way that we work. For example, here:



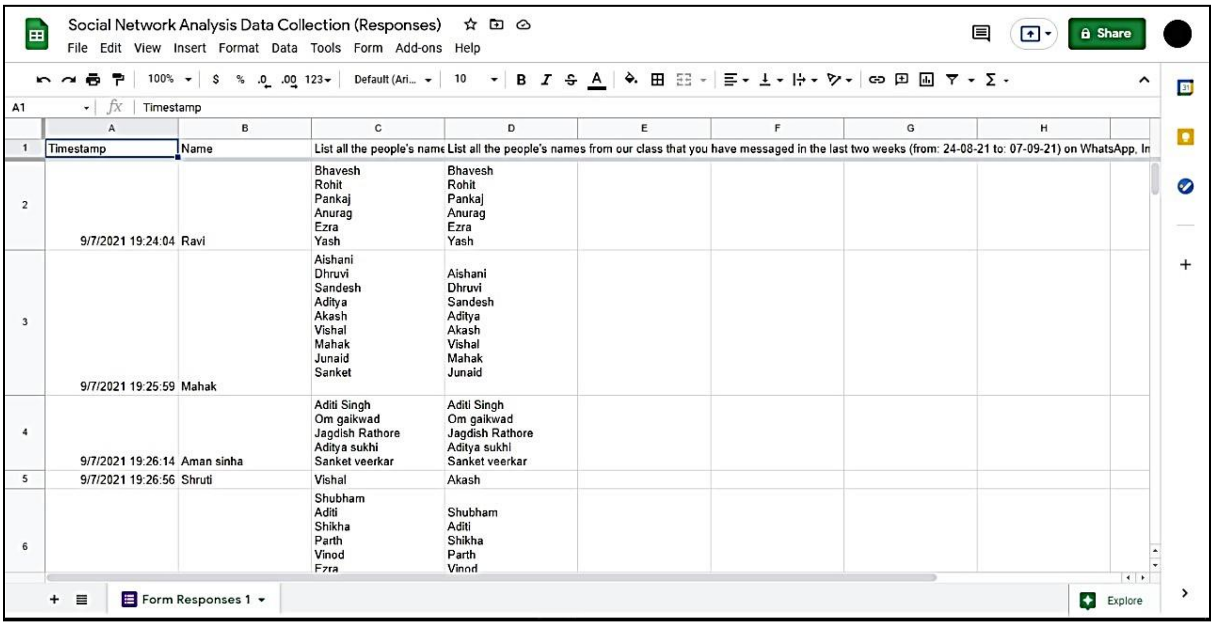
We performed a survey within our class in order to analyse the social interactions among our classmates primarily through means of WhatsApp and Instagram.

The questions included the name of the person who responded in order to track the initial and/or terminal node for a given interaction.

List all the people's names from our class that you have messaged in the last two weeks (from: 24-08-21 to: 07-09-21) on WhatsApp, Instagram or other platforms (Gmail, Linkedin etc.)

List all the people's names from our class that you have messaged in the last two weeks (from: 24-08-21 to: 07-09-21) on WhatsApp, Instagram or other platforms (Gmail, Linkedin etc.)

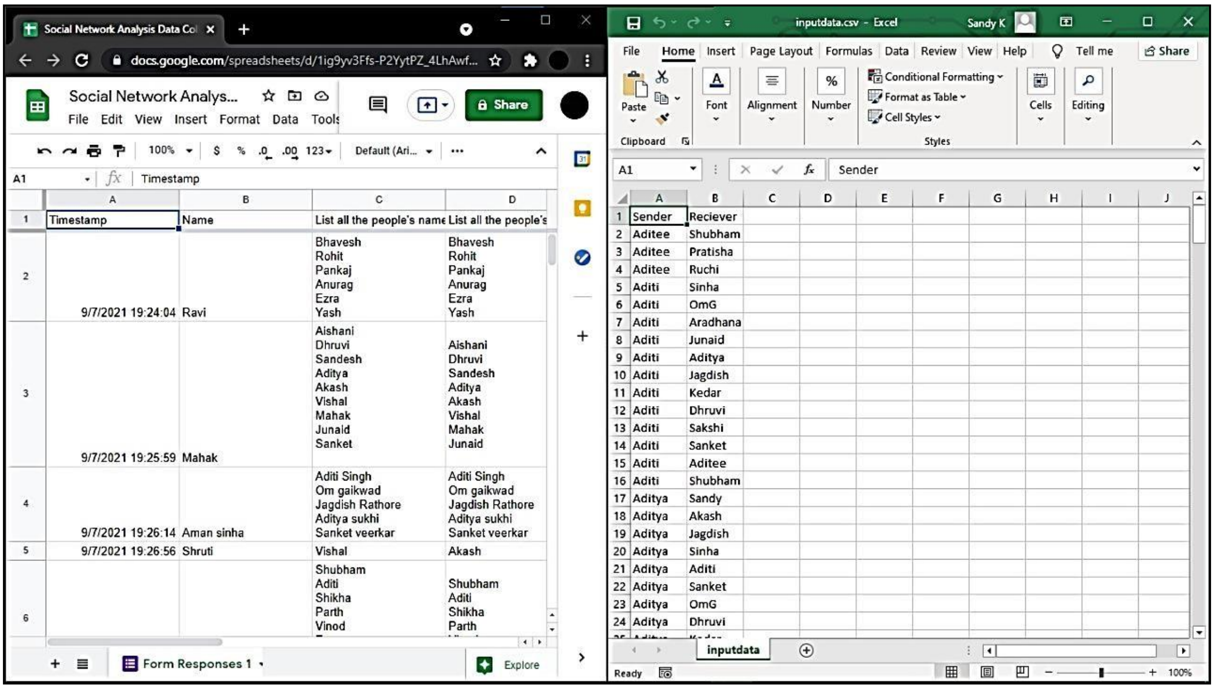
Responders were asked to filled out both received and sent messages (interactions) in order to minimize human error and/or bias in case of e.g., a responder forgetting about a message they received (would otherwise make a "hole" in the datapoints) would be compensated by the opposite node since they likely report on the same interaction that the first responder forgot.



Social Network Analysis Data Collection (Responses)

	A	B	C	D
1	Timestamp	Name	List all the people's name	List all the people's names from our class that you have messaged in the last two weeks (from: 24-08-21 to: 07-09-21) on WhatsApp, In
2	9/7/2021 19:24:04	Ravi	Bhavesh Rohit Pankaj Anurag Ezra Yash	Bhavesh Rohit Pankaj Anurag Ezra Yash
3	9/7/2021 19:25:59	Mahak	Aishani Dhruvi Sandesh Aditya Akash Vishal Mahak Junaid Sanket	Aishani Dhruvi Sandesh Aditya Akash Vishal Mahak Junaid
4	9/7/2021 19:26:14	Aman sinha	Aditi Singh Om gaikwad Jagdish Rathore Aditya sukhl Sanket veerkar	Aditi Singh Om gaikwad Jagdish Rathore Aditya sukhl Sanket veerkar
5	9/7/2021 19:26:56	Shruti	Vishal	Akash
6			Shubham Aditi Shikha Parth Vinod Ezra	Shubham Aditi Shikha Parth Vinod

Above displayed image shows raw Data from all the 40 responses.



All spam/troll responses were removed manually alongside removing "invalid" response (people that failed to submit a response but were part of the response pool received) followed by transposing data into a usable .csv file.

For example name like Mahak was often observed to be misspelled as Mahek, Mehak, Mehek, Mahek therefore the above changes were made to manually correct/standardize spellings of names preventing "false" nodes of alternate spellings.

1	Sender	Reciever	Aditya	Junaid	Akash	Sandy	Jagdish	Junaid	Karan	Pratyush	Mirthula	Sandy	Parth	Akash	Sakshi	Ruchi	Sanket	Kedar	Shubham	Sandy	Visha
2	Aditee	Shubham	Aditya	Vishal	Akash	Mahak	Jagdish	Aditi	Kedar	Sandy	Mirthula	Junaid	Parth	Shubham	Sakshi	Aditee	Sanket	Vishal	Shubham	Parth	Visha
3	Aditee	Pratisha	Aditya	Mahak	Akash	Karan	Jagdish	Aditya	Kedar	Aishani	OmB	Aishani	Prathame	Kedar	Samreen	Aishani	Sanket	Sanchita	Shubham	Sakshi	Visha
4	Aditee	Ruchi	Aditya	Aishani	Aradhana	Vishal	Jagdish	Sinha	Kedar	Dhruvi	OmB	Dhruvi	Prathame	Shiv	Samreen	Dhruvi	Sanket	Jagdish	Shubham	Aditi	Visha
5	Aditi	Sinha	Aditya	Karan	Aradhana	Aditi	Jagdish	Aradhana	Kedar	Prathame	OmB	Mahak	Prathame	Sanchita	Samreen	OmB	Sanket	Junaid	Shubham	Sanket	Visha
6	Aditi	OmG	Aditya	Parth	Aradhana	Sanket	Jagdish	OmG	Kedar	OmB	OmB	Kedar	Prathame	Sanket	Sanchita	Pratisha	Sanket	Mahak	Shubham	Aditee	Visha
7	Aditi	Aradhana	Aditya	Mirthula	Aradhana	Jagdish	Jagdish	Sanket	Kedar	Aditya	OmB	Samreen	Prathame	OmB	Sanchita	Sanket	Sanket	Sinha	Shubham	Pratyu	(Ctrl)
8	Aditi	Junaid	Aditya	Shubham	Dhruvi	Sandy	Jagdish	Kedar	Kedar	Sanket	OmB	Junaid	Pratisha	Sanchita	Sanchita	Akash	Sanket	Dhruvi	Sinha	Aditi	
9	Aditi	Aditya	Aditya	Pratyush	Dhruvi	Aishani	Junaid	OmG	Kedar	Jagdish	OmB	OmG	Pratisha	Aditee	Sanchita	Sanket	Sanket	Sandy	Sinha	OmG	
10	Aditi	Jagdish	Aishani	Dhruvi	Dhruvi	Kedar	Junaid	Mirthula	Kedar	OmG	OmB	Prathame	Pratyush	Akash	Sandy	Aditya	Sanket	Aditya	Sinha	Jagdish	
11	Aditi	Kedar	Aishani	Kedar	Dhruvi	Shubham	Junaid	Mahak	Kedar	Aditi	OmB	Sandy	Pratyush	Shruti	Sandy	Aishani	Sanket	Akash	Sinha	Aditya	
12	Aditi	Dhruvi	Aishani	Shubham	Dhruvi	Aditya	Junaid	Dhruvi	Kedar	Junaid	OmB	Aditya	Pratyush	Shubham	Sandy	Akash	Sanket	Pratyush	Sinha	Sanket	
13	Aditi	Sakshi	Aishani	Aditya	Dhruvi	Mirthula	Junaid	Aditi	Mahak	Aishani	OmB	Akash	Pratyush	Ruchi	Sandy	Dhruvi	Shiv	Pratyush	Sinha	Sandy	
14	Aditi	Sanket	Aishani	Samreen	Dhruvi	Ruchi	Junaid	Jagdish	Mahak	Dhruvi	OmB	Vishal	Pratyush	Shiv	Sandy	Ezra	Shiv	Prathame	Vibha	Karan	
15	Aditi	Aditee	Aishani	Mirthula	Dhruvi	Junaid	Junaid	Aditya	Mahak	Sandy	OmG	Aditya	Ravi	Ezra	Sandy	Kedar	Shiv	Karan	Vibha	Shruti	
16	Aditi	Shubham	Aishani	OmB	Dhruvi	Sanket	Junaid	Kedar	Mahak	Aditya	OmG	Sandy	Ruchi	Aditee	Sandy	Mirthula	Shiv	Akash	Vinod	Akash	
17	Aditya	Sandy	Aishani	Mahak	Dhruvi	Mahak	Junaid	OmB	Mahak	Akash	OmG	Vishal	Ruchi	Shubham	Sandy	OmG	Shruti	Akash	Vinod	Sakshi	
18	Aditya	Akash	Aishani	Sandy	Dhruvi	OmB	Jitendra	Pratyush	Mahak	Vishal	OmG	Sanket	Ruchi	Pratyush	Sandy	Sanket	Shruti	Vibha	Vinod	Ezra	
19	Aditya	Jagdish	Akash	Dhruvi	Dhruvi	Aditi	Karan	Aditya	Mahak	Junaid	OmG	Kedar	Ruchi	Dhruvi	Sandy	Shubham	Shruti	Karan	Vinod	Pratyush	
20	Aditya	Sinha	Akash	Shiv	Dhruvi	Aditee	Karan	Akash	Mahak	Mirthula	OmG	Junaid	Ruchi	Sakshi	Sandy	Sinha	Shruti	Pratyush	Vinod	Aditya	
21	Aditya	Aditi	Akash	Sanket	Dhruvi	Akash	Karan	Shruti	Mahak	OmB	OmG	Aditi	Sakshi	Shubham	Sandy	Vishal	Shubham	Akash	Vinod	Jitendra	
22	Aditya	Sanket	Akash	Shubham	Ezra	Sakshi	Karan	Parth	Mirthula	Mahak	OmG	Sinha	Sakshi	Aditi	Sandy	Mahak	Shubham	Aditya	Vishal	Aditya	
23	Aditya	OmG	Akash	Pratyush	Ezra	Ravi	Karan	Vibha	Mirthula	Dhruvi	OmG	Jagdish	Sakshi	Parth	Sanket	Aradhana	Shubham	Ruchi	Vishal	Sandy	
24	Aditya	Dhruvi	Akash	Shruti	Ezra	Sandy	Karan	Shiv	Mirthula	Aditya	OmG	OmB	Sakshi	Vinod	Sanket	Aditi	Shubham	Dhruvi	Vishal	Mahak	

After filtering everything out a .csv is created that roughly distinguishes the collected data.

inputdata.txt - Notepad

File Edit Format View Help

```
Sender,Reciever
Aditee,Shubham
Aditee,Pratisha
Aditee,Ruchi
Aditi,Sinha
Aditi,OmG
Aditi,Aradhana
Aditi,Junaid
Aditi,Aditya
Aditi,Jagdish
Aditi,Kedar
Aditi,Dhruvi
Aditi,Sakshi
Aditi,Sanket
Aditi,Aditee
Aditi,Shubham
Aditya,Sandy
Aditya,Akash
Aditya,Jagdish
Aditya,Sinha
Aditya,Aditi
Aditya,Sanket
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

A list of the sender and receiver is now sorted and generated for further use!!

CODE EXPLANATION

The code that one writes must be understood not only by the machine but also by humans. Quality should always be everyone's priority. Throughout the development process, the goal should be the delivery of good quality and working code. Hence the code that we have here not only shows the precise and easy to understand visualizations but is also well-documented and thoroughly analysed.

```
package 'igraph' successfully unpacked and MD5 sums checked
package 'readr' successfully unpacked and MD5 sums checked
package 'tidyr' successfully unpacked and MD5 sums checked
package 'RColorBrewer' successfully unpacked and MD5 sums checked
```

First all the required packages are installed.

```
R ▾ | Global Environment ▾
Data
● B          1225 obs. of 3 variables
● B1         254 obs. of 3 variables
● cnet       List of 10
● data       255 obs. of 2 variables
● df         List of 10
● DF         254 obs. of 12 variables
● dfha       List of 10
Values
  as         Named num [1:35] 0.6231 0.7981 0.0201 0.1776 0.177 ...
  df_bw      Named num [1:35] 28.8 38.5 1.7 4.5 19.4 ...
  df_deg     Named num [1:35] 23 28 4 10 12 23 36 8 16 20 ...
  df_eig     Named num [1:35] 0.6558 0.8224 0.0317 0.1993 0.179 ...
  hs         Named num [1:35] 0.6645 0.8181 0.0348 0.2144 0.1863 ...
  pal        chr [1:3] "#8DD3C7" "#FFFFB3" "#BEBADA"
```

Next as you can see the data is stored in the variables of the R global environment.

```
IGRAPH eaebb3b UNW- 35 254 --
+ attr: name (v/c), Freq (e/n), weight (e/n)
+ edges from eaebb3b (vertex names):
 [1] Aditi    --Aditee   Dhruvi  --Aditee   Pratisha--Aditee   Ruchi   --Aditee   Sakshi  --Aditee   Shubham --Aditee   Aditi   --Aditya
 [8] Aditi    --Aradhana Aditi   --Dhruvi   Aditi   --Jagdish  Aditi   --Junaid   Aditi   --Kedar    Aditi   --OmG      Aditi   --Sakshi
[15] Aditi    --Sanket   Aditi   --Shubham  Aditi   --Sinha    Aditi   --Aditya   Aditya  --Aishani  Aditya  --Akash    Dhruvi  --Aditya
[22] Aditya   --Jagdish  Aditya  --Junaid   Aditya  --Karan    Aditya  --Kedar    Aditya  --Mahak    Aditya  --Mirthula Aditya  --OmB
[29] Aditya   --OmG      Aditya  --Sandy    Aditya  --Sanket   Shubham --Aditya   Aditya  --Sinha    Aditya  --Vinod    Aditya  --Vishal
[36] Aditya   --Aishani  Dhruvi  --Aishani  Kedar   --Aishani  Aishani --Mahak    Aishani --Mirthula Aishani --OmB      Aishani --Samreen
[43] Aishani  --Sandy    Shubham --Aishani  Aditya  --Akash    Dhruvi  --Akash    Akash   --Karan    Akash   --Mahak    Akash   --OmB
[50] Akash    --Parth    Akash   --Pratyush Akash   --Sanchita Akash   --Sandy    Sanket  --Akash    Akash   --Shiv     Akash   --Shruti
+ ... omitted several edges
>
> #1. igraph summary
> gsize(df)
[1] 254
> gorder(df)
[1] 35
>
> #2. Nodelist
> V(df)
+ 35/35 vertices, named, from eaebb3b:
 [1] Aditi     Dhruvi    Pratisha  Ruchi     Sakshi    Shubham   Aditya    Aradhana  Jagdish   Junaid    Kedar     OmG
[13] Sanket    Sinha     Aishani   Akash     Karan     Mahak     Mirthula  OmB       Sandy     Vinod     Vishal    Samreen
[25] Parth     Pratyush  Sanchita  Shiv      Shruti    Ravi      Vibha     Prathamesh Aditee   Jitendra  Ezra
>
```

A raw network of data and nodelist is created using the igraph library.

```
> #3. Edgelist
> E(df)
+ 254/254 edges from eaebb3b (vertex names):
 [1] Aditi    --Aditee   Dhruvi   --Aditee   Pratisha--Aditee   Ruchi    --Aditee   Sakshi   --Aditee   Shubham --Aditee   Aditi    --Aditya
 [8] Aditi    --Aradhana Aditi    --Dhruvi   Aditi    --Jagdish  Aditi    --Junaid   Aditi    --Kedar    Aditi   --OmG      Aditi    --Sakshi
[15] Aditi    --Sanket   Aditi    --Shubham  Aditi    --Sinha    Aditi    --Aditya   Aditya   --Aishani  Aditya  --Akash    Dhruvi   --Aditya
[22] Aditya   --Jagdish  Aditya   --Junaid   Aditya   --Karan    Aditya   --Kedar    Aditya   --Mahak    Aditya  --Mirthula Aditya   --OmB
[29] Aditya   --OmG      Aditya   --Sandy    Aditya   --Sanket   Shubham  --Aditya   Aditya   --Sinha    Aditya  --Vinod    Aditya   --Vishal
[36] Aditya   --Aishani  Dhruvi   --Aishani  Kedar    --Aishani  Aishani  --Mahak    Aishani  --Mirthula Aishani --OmB      Aishani  --Samreen
[43] Aishani  --Sandy    Shubham  --Aishani  Aditya   --Akash    Dhruvi   --Akash    Akash    --Karan    Akash   --Mahak    Akash    --OmB
[50] Akash    --Parth    Akash    --Pratyush Akash    --Sanchita Akash    --Sandy    Sanket   --Akash    Akash   --Shiv     Akash    --Shruti
[57] Shubham  --Akash    Akash    --Vinod    Akash    --Vishal   Aditi    --Aradhana Aradhana--Jagdish  Aradhana--Sanket   Aradhana--Vishal
[64] Aditi    --Dhruvi   Dhruvi   --Aditya   Dhruvi   --Aishani  Dhruvi   --Akash    Dhruvi   --Junaid   Dhruvi  --Kedar    Dhruvi   --Mahak
+ ... omitted several edges
>
> #5. Adjacency matrix
> df[c(1:10),c(1:10)]
10 x 10 sparse Matrix of class "dgCMatrix"
   [[ suppressing 10 column names 'Aditi', 'Dhruvi', 'Pratisha' ... ]]

Aditi    . 2 . . 2 2 2 2 2 2
Dhruvi   2 . . 2 . 2 2 . . 2
Pratisha . . . . . . . . . .
Ruchi    . 2 . . 2 2 . . . .
Sakshi   2 . . 2 . 2 . . . .
Shubham  2 2 . 2 2 . 2 . . .
Aditya   2 2 . . . 2 . . 2 2
Aradhana 2 . . . . . . . 2 .
Jagdish  2 . . . . . 2 2 . 2
Junaid   2 2 . . . . 2 . 2 .
```

Then a slice of the interaction matrix and an edge list is generated to further simply the data.

```
>
> df_bw<-betweenness(df, directed = FALSE)
> V(df)$betweenness<-df_bw
> V(df)$betweenness
 [1] 28.8491844 38.4581667  1.7028266  4.4953142 19.3918976 29.6842279 77.8123078  0.7260081  2.5635763  5.5433534 11.9406751  4.2265001
[13] 46.7605535  0.6505495  7.0631995 62.8708423 33.5572733  3.8831162  0.5222222 15.8834348 49.8505812 17.4326406 18.4011896  1.5412659
[25]  4.0589265 43.4014098 14.1171958  5.7025253 10.2826495  0.0000000  0.0000000  6.7433232 19.3816719  0.0000000 34.5013913
> which.max(df_bw)
Aditya
     7
> df[c(1:10),c(1:10)]
10 x 10 sparse Matrix of class "dgCMatrix"
   [[ suppressing 10 column names 'Aditi', 'Dhruvi', 'Pratisha' ... ]]

Aditi    . 2 . . 2 2 2 2 2 2
Dhruvi   2 . . 2 . 2 2 . . 2
Pratisha . . . . . . . . . .
Ruchi    . 2 . . 2 2 . . . .
Sakshi   2 . . 2 . 2 . . . .
Shubham  2 2 . 2 2 . 2 . . .
Aditya   2 2 . . . 2 . . 2 2
Aradhana 2 . . . . . . . 2 .
Jagdish  2 . . . . . 2 2 . 2
Junaid   2 2 . . . . 2 . 2 .
```

In the above image it is shown the betweenness of the raw values is shown and it is observed that Aditya has highest betweenness value i.e 7.

```
> #=================== Measuring Centrality =====================#
> #============================================================#
> # In this section, you will measure the centrality of the igraph   #
> # object, "df". You will be able to see how the theoretical   #
> # concept of each centrality such as degree, eigenvector, and      #
> # betweenness centrality is measured by the igraph.               #
> #============================================================#
>
>
> #1. Degree centrality
> # Network measures
> degree(df, mode='all')
    Aditi    Dhruvi  Pratisha     Ruchi    Sakshi    Shubham    Aditya  Aradhana   Jagdish    Junaid     Kedar       OmG    Sanket
       23        28         4        10        12         23        36         8        16        20        22        20        29
    Sinha   Aishani     Akash     Karan     Mahak   Mirthula       OmB     Sandy     Vinod    Vishal   Samreen     Parth  Pratyush
       12        18        25        14        19         12        20        27         8        17         6         7        16
 Sanchita      Shiv    Shruti      Ravi     Vibha Prathamesh    Aditee  Jitendra      Ezra
        6         8         9         2         4          8         9         2         8
>
> df_deg<-degree(df,mode=c("All"))
> V(df)$degree<-df_deg
> V(df)$degree
 [1] 23 28  4 10 12 23 36  8 16 20 22 20 29 12 18 25 14 19 12 20 27  8 17  6  7 16  6  8  9  2  4  8  9  2  8
> which.max(df_deg)
Aditya
     7
```

Next the degrees data is shown where again the person with maximum degrees of nodes is Aditya.

```
> #2. Eigenvector centrality
> df_eig <- evcent(df)$vector
> V(df)$Eigen<-df_eig
> V(df)$Eigen
 [1] 0.65578558 0.82238724 0.03172143 0.19929308 0.17904258 0.57477013 1.00000000 0.24558783 0.50781154 0.65314655 0.70988875 0.65494073
[13] 0.82142237 0.43567336 0.55995482 0.60905180 0.22653326 0.62642791 0.43782729 0.57295036 0.80113474 0.12267920 0.51841305 0.16258828
[25] 0.16631464 0.29498626 0.16302204 0.12923592 0.13986391 0.01071669 0.03594372 0.18681239 0.16033461 0.02048658 0.10924202
> which.max(df_eig)
Aditya
     7
>
> #3. Betweenness centrality
> df_bw<-betweenness(df, directed = FALSE)
> V(df)$betweenness<-df_bw
> V(df)$betweenness
 [1] 28.8491844 38.4581667  1.7028266  4.4953142 19.3918976 29.6842279 77.8123078  0.7260081  2.5635763  5.5433534 11.9406751  4.2265001
[13] 46.7605535  0.6505495  7.0631995 62.8708423 33.5572733  3.8831162  0.5222222 15.8834348 49.8505812 17.4326406 18.4011896  1.5412659
[25]  4.0589265 43.4014098 14.1171958  5.7025253 10.2826495  0.0000000  0.0000000  6.7433232 19.3816719  0.0000000 34.5013913
> which.max(df_bw)
Aditya
     7
```

Next on further checking the eigenvector centrality and the betweenness centrality the same maximum is observed.

```
> DF<-as_long_data_frame(df)
> DF
   from to Freq weight from_name from_betweenness from_degree from_Eigen  to_name to_betweenness to_degree  to_Eigen
1     1 33    1      1     Aditi       28.8491844          23 0.65578558   Aditee     19.3816719         9 0.1603346
2     2 33    1      1    Dhruvi       38.4581667          28 0.82238724   Aditee     19.3816719         9 0.1603346
3     3 33    1      1  Pratisha        1.7028266           4 0.03172143   Aditee     19.3816719         9 0.1603346
4     4 33    1      1     Ruchi        4.4953142          10 0.19929308   Aditee     19.3816719         9 0.1603346
5     5 33    1      1    Sakshi       19.3918976          12 0.17904258   Aditee     19.3816719         9 0.1603346
6     6 33    1      1   Shubham       29.6842279          23 0.57477013   Aditee     19.3816719         9 0.1603346
7     1  7    1      1     Aditi       28.8491844          23 0.65578558   Aditya     77.8123078        36 1.0000000
8     1  8    1      1     Aditi       28.8491844          23 0.65578558 Aradhana      0.7260081         8 0.2455878
9     1  2    1      1     Aditi       28.8491844          23 0.65578558   Dhruvi     38.4581667        28 0.8223872
10    1  9    1      1     Aditi       28.8491844          23 0.65578558  Jagdish      2.5635763        16 0.5078115
11    1 10    1      1     Aditi       28.8491844          23 0.65578558   Junaid      5.5433534        20 0.6531465
12    1 11    1      1     Aditi       28.8491844          23 0.65578558    Kedar     11.9406751        22 0.7098887
13    1 12    1      1     Aditi       28.8491844          23 0.65578558      OmG      4.2265001        20 0.6549407
14    1  5    1      1     Aditi       28.8491844          23 0.65578558   Sakshi     19.3918976        12 0.1790426
15    1 13    1      1     Aditi       28.8491844          23 0.65578558   Sanket     46.7605535        29 0.8214224
16    1  6    1      1     Aditi       28.8491844          23 0.65578558  Shubham     29.6842279        23 0.5747701
17    1 14    1      1     Aditi       28.8491844          23 0.65578558    Sinha      0.6505495        12 0.4356734
18    1  7    1      1     Aditi       28.8491844          23 0.65578558   Aditya     77.8123078        36 1.0000000
19    7 15    1      1    Aditya       77.8123078          36 1.00000000  Aishani      7.0631995        18 0.5599548
20    7 16    1      1    Aditya       77.8123078          36 1.00000000    Akash     62.8708423        25 0.6090518
21    2  7    1      1    Dhruvi       38.4581667          28 0.82238724   Aditya     77.8123078        36 1.0000000
22    7  9    1      1    Aditya       77.8123078          36 1.00000000  Jagdish      2.5635763        16 0.5078115
23    7 10    1      1    Aditya       77.8123078          36 1.00000000   Junaid      5.5433534        20 0.6531465
24    7 17    1      1    Aditya       77.8123078          36 1.00000000    Karan     33.5572733        14 0.2265333
25    7 11    1      1    Aditya       77.8123078          36 1.00000000    Kedar     11.9406751        22 0.7098887
26    7 18    1      1    Aditya       77.8123078          36 1.00000000    Mahak      3.8831162        19 0.6264279
27    7 19    1      1    Aditya       77.8123078          36 1.00000000 Mirthula      0.5222222        12 0.4378273
28    7 20    1      1    Aditya       77.8123078          36 1.00000000      OmB     15.8834348        20 0.5729504
29    7 12    1      1    Aditya       77.8123078          36 1.00000000      OmG      4.2265001        20 0.6549407
30    7 21    1      1    Aditya       77.8123078          36 1.00000000    Sandy     49.8505812        27 0.8011347
```

Then a summary of the data frame is printed to give a clearer view.

```
> hist(V(df)$degree,
+      col = 'green',
+      main = 'Histogram of Node Degree',
+      ylab = 'Frequency',
+      xlab = 'Degree of Vertices')
>
```

Next the visualizations are generated starting with the plotting of a histogram.

```
> #==================================================================#
> #================== Measuring Network Structure ===================#
> #==================================================================#
> # In this section, you will measure the indicators of the network  #
> # structure such as network density
> #==================================================================#
>
> #1. Network Density
> edge_density(df) # Global density
[1] 0.4268908
>
>
> #1. Plotting a network with the degree centrality
>
> set.seed(1001)
> library(RColorBrewer) # This is the color library
> pal<-brewer.pal(length(unique(V(df)$Sender)), "Set3") # Vertex color assigned per each class number
Warning message:
In brewer.pal(length(unique(V(df)$Sender)), "Set3") :
  minimal value for n is 3, returning requested palette with 3 different levels

> plot(df,edge.color = 'black',vertex.label.cex = 0.9,
+      vertex.color='yellow',
+      vertex.size = df_deg, edge.width=sqrt(E(df)$weight/800),
+      layout = layout.fruchterman.reingold, main="Network graph with degree centrality")
>
>
> #1. Plotting a network with the eigenvector centrality
>
> set.seed(1001)
> plot(df,vertex.label.color= "black", edge.color = 'green',vertex.label.cex = 1,
+      vertex.color='orange',
+      vertex.size = df_eig*22, edge.width=sqrt(E(df)$weight/800),
+      layout = layout.circle, main="Network graph with eigen vector centrality",)
```

The measure of the indicators of the network structure density is then graphed. Here the edge density. is observed to be 0. 4268908.

```
> #2. Plotting a network with the betweenness centrality
>
> par(bg="white")
> set.seed(1001)
> plot(df,edge.color = 'lightblue',vertex.label.color= "black", vertex.label.cex =1, vertex.color='orange',
+      vertex.color=pal[as.numeric(as.factor(vertex_attr(df, "Class")))],
+      vertex.size = df_bw/4, edge.width=sqrt(E(df)$weight/800),
+      layout = layout.sphere, main = "Network with betweenness centrality")
>
>
> #3.1. between degree and betweenness centrality
> plot(V(df)$degree, V(df)$betweenness)
>
> #3.2. between degree and eigenvector centrality
> plot(V(df)$degree, V(df)$Eigen)
>
> #1. Louvain clustering
> cnet <- cluster_edge_betweenness(df) # You can check which vertices belongs to which clusters.
Warning messages:
1: In cluster_edge_betweenness(df) :
  At community.c:460 :Membership vector will be selected based on the lowest modularity score.
2: In cluster_edge_betweenness(df) :
  At community.c:467 :Modularity calculation with weighted edge betweenness community detection might not make sense -- modularity treats edge weights
 as similarities while edge betwenness treats them as distances
>
> #2. Plotting the Betweenness Centrality network with the community detection
>
> set.seed(1001) # To duplicate the computer process and create exactly the same network repetitively you should set the seed.
> plot(cnet, df, edge.color = 'lightblue',vertex.label.cex =0.9,
+      vertex.color='yellow', vertex.label.color='black',
+      vertex.size = df_bw/3, edge.width=sqrt(E(df)$weight/800),
+      layout = layout.fruchterman.reingold, main = "Betweenness Centrality network with the community detection")
>
> set.seed(1001) # To duplicate the computer process and create exactly the same network repetitively you should set the seed.
```

```
> set.seed(1001) # To duplicate the computer process and create exactly the same network repetitively you should set the seed.
> plot(cnet, df, edge.color = 'blue',vertex.label.cex =0.9,
+      vertex.color='yellow', vertex.label.color='black',
+      vertex.size = df_bw/3, edge.width=sqrt(E(df)$weight/800),
+      layout = layout.circle, main = "Betweenness Centrality network with the community detection (part 2)")
>
> #interactive visualization
>
> -----#visualization
+ tkplot.fit.to.screen( width = NULL, height = NULL)
Error in .tkplot.get(tkp.id) :
  argument "tkp.id" is missing, with no default
> tkplot(df, vertex.color = rgb(0.8,0.2,0.2,0.9), directed = T)
[1] 1
>
>
> set.seed(1001)
> tkplot(df, edge.color ='blue', vertex.label.cex = 0.9, vertex.size = hs*50,
+        vertex.color = 'yellow', vertex.label.color = 'black', edge.color = 'lightblue', layout = layout.fruchterman.reingold)
Error in i.parse.plot.params(graph, list(...)) : object 'hs' not found
> #-------or-------
>
>
> dfha<-graph_from_data_frame(B1, directed = T)
>
> hs <- hub_score(dfha)$vector
> as <- authority.score(dfha)$vector
> par(mfrow=c(1,2))
> set.seed(1001)
> plot(dfha,
+      vertex.size=hs*30,
+      main = 'Hubs',
+      vertex.color = rainbow(52),
+      edge.arrow.size=0.1,
+      layout = layout.kamada.kawai)
```

The visualizations of the network with the betweenness centrality, between degree and eigenvector centrality, between degree and betweenness centrality, betweenness centrality network with the community detection and other visualization using tkplot are charted.

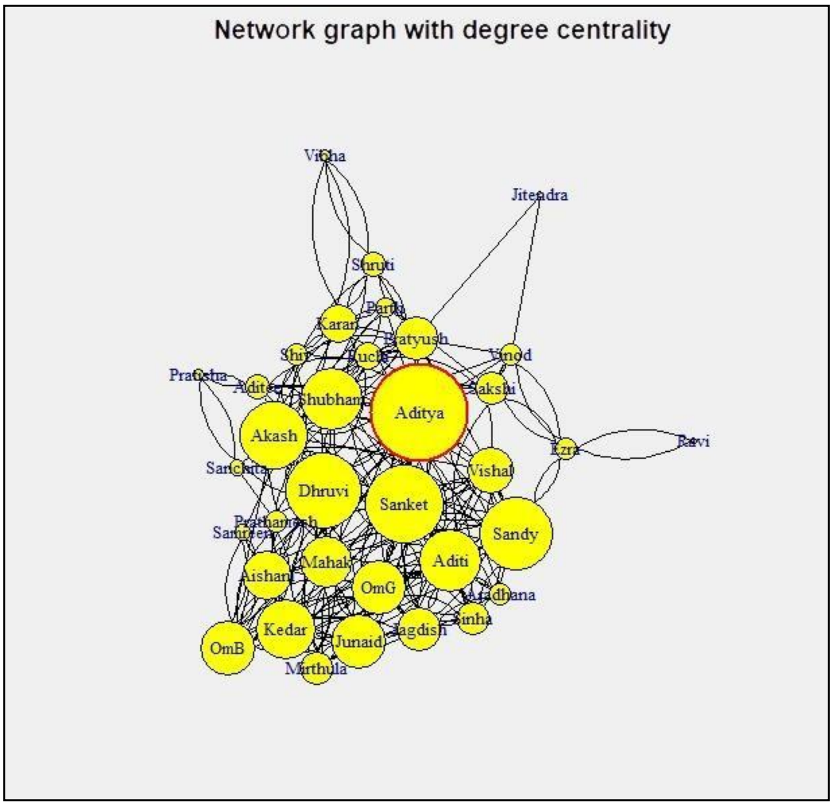
DATA VISUALIZATION

With so much information being collected through data, we must have a way to paint a picture of that data so we can interpret it.

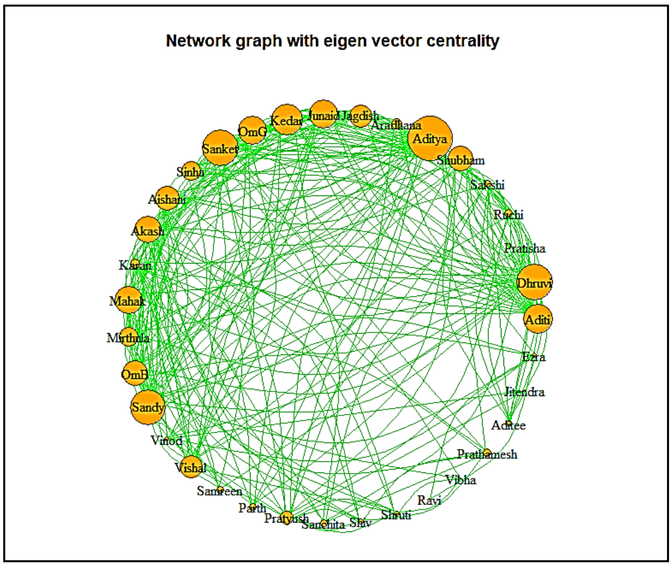
Data visualization gives us a clear idea of what the information means by giving it visual context through maps or graphs.

This makes the data more natural for the human mind to comprehend and therefore makes it easier to identify trends, patterns, and outliers within large data sets.

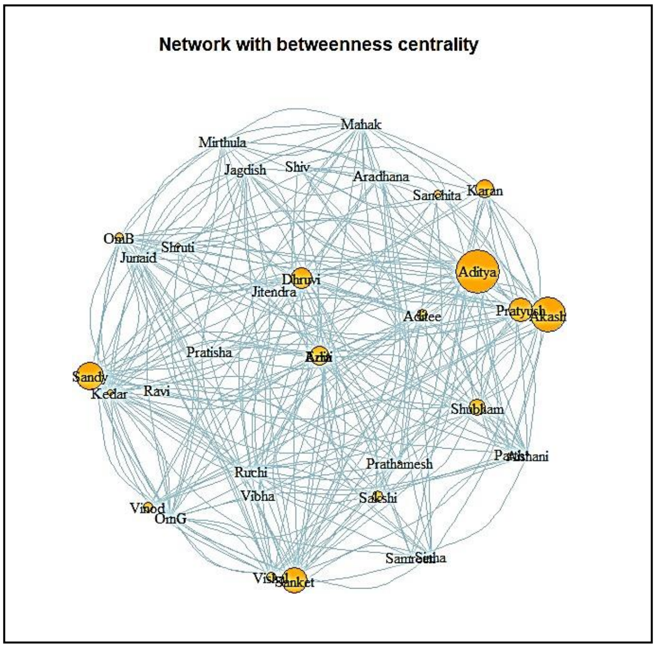
Here we have different types of graphs, plots and visualizations that are used for the project:



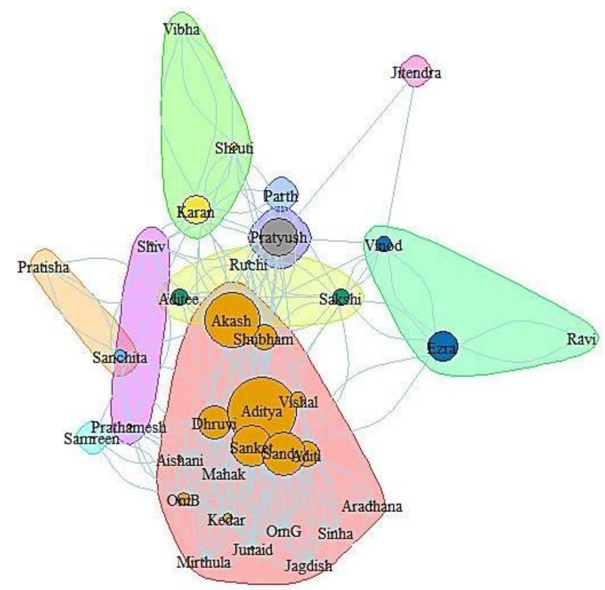
Network graph with degree centrality: In a connected graph, the normalized closeness centrality (or closeness) of a node is the average length of the shortest path between the node and all other nodes in the graph. Thus, the more central a node is, the closer it is to all other nodes. is the number of nodes in the graph. This is a graph which shows the network on the basis of degree centrality. Its vertices depict the betweenness. I.e.: Bigger the vertex, higher the betweenness. The layout used here is the Fruchterman Re.ingold layout.



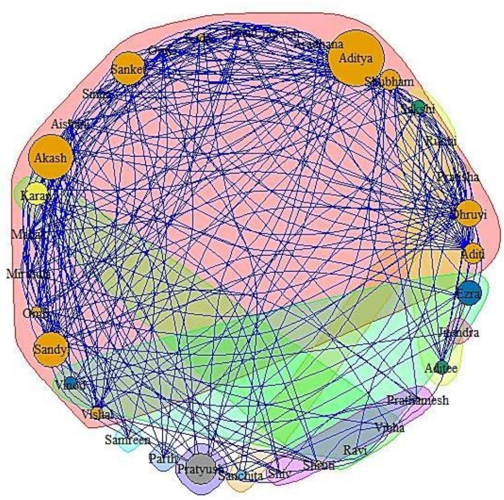
Network graph with eigen vector centrality: Eigenvector centrality is a measure of the influence a node has on a network. If a node is pointed to by many nodes (which also have high eigenvector centrality) then that node will have high eigenvector centrality. Eigenvector centrality differs from in-degree centrality: a node receiving many links does not necessarily have a high eigenvector centrality (it might be that all linkers have low or null eigenvector centrality). Moreover, a node with high eigenvector centrality is not necessarily highly linked (the node might have few but important linkers).



A network with betweenness centrality: Betweenness centrality finds wide application in network theory; it represents the degree to which nodes stand between each other. a node with higher betweenness centrality would have more control over the network, because more information will pass through that node.

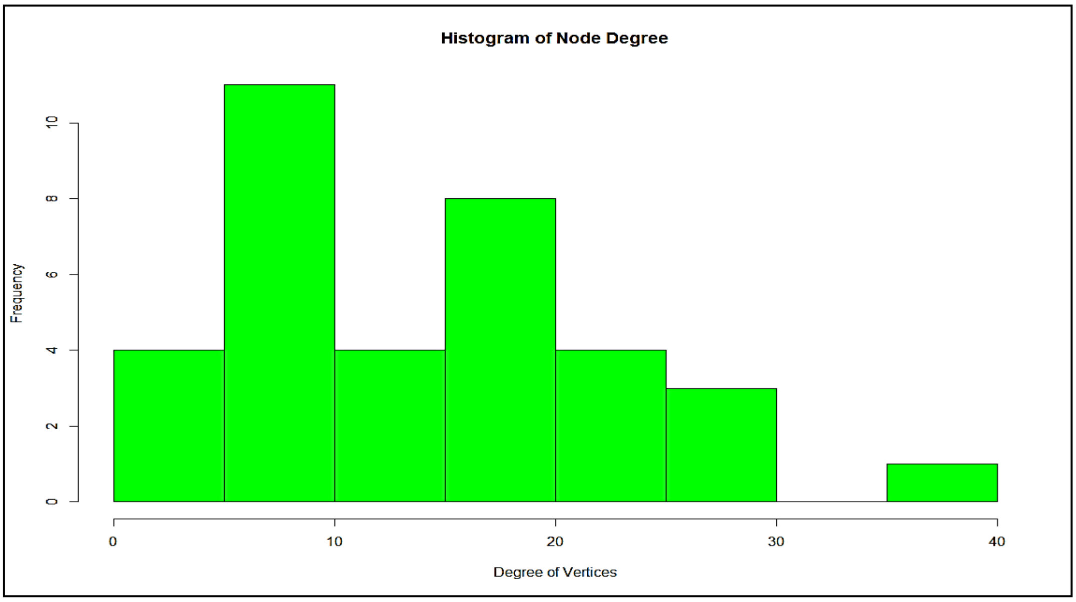


Betweenness Centrality network with the community detection



Betweenness Centrality network with the community detection (part 2)

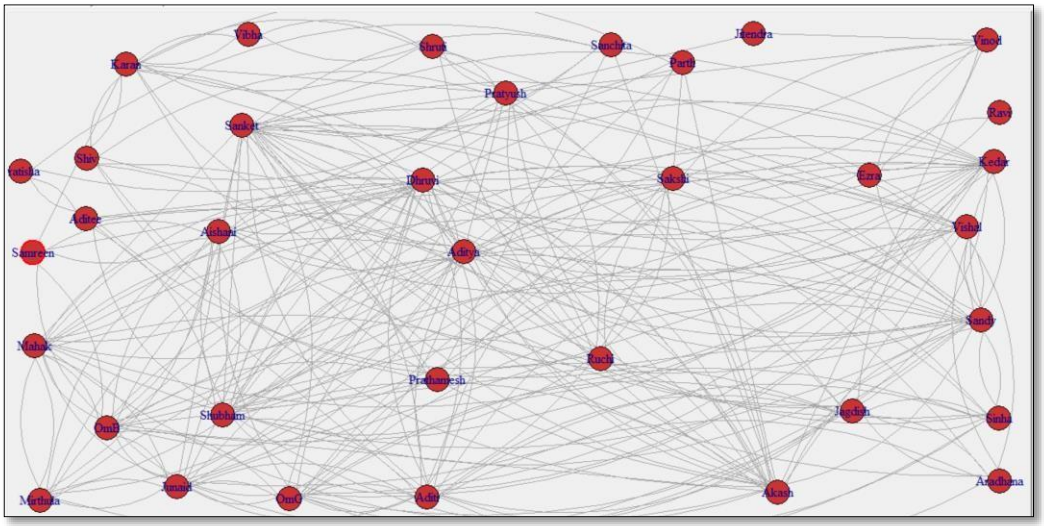
Betweenness clarity network with community detection: Detecting communities in a network is one of the most important tasks in network analysis. In a large-scale network, such as an online social network, we could have millions of nodes and edges. Detecting communities in such networks becomes a herculean task. Therefore, we need community detection algorithms that can partition the network into multiple communities.

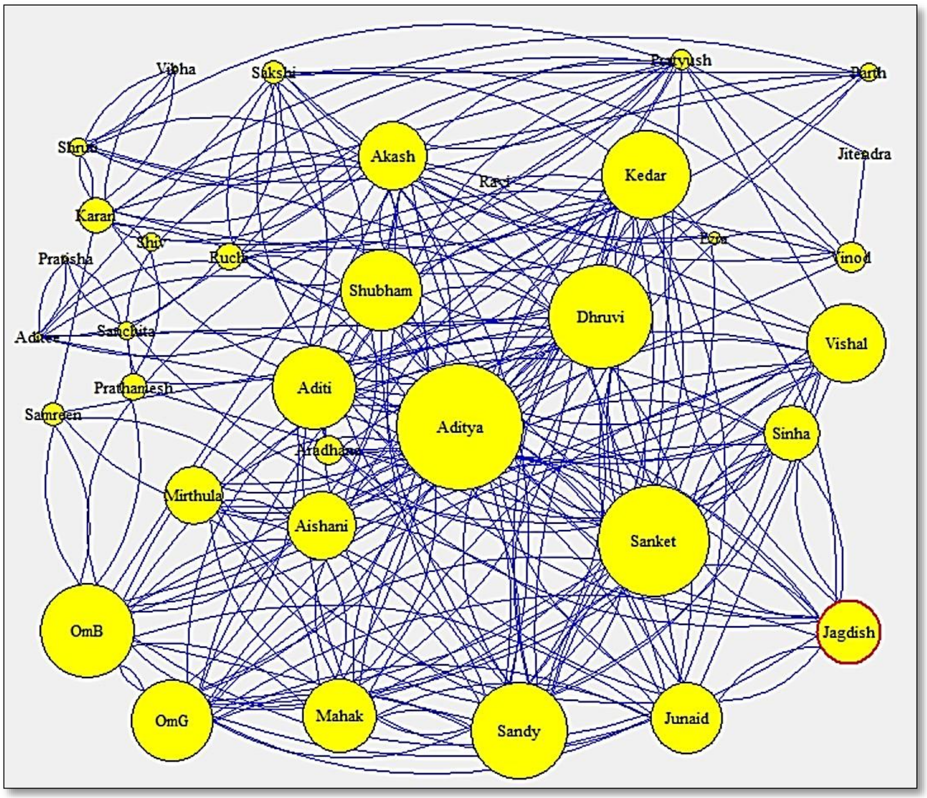


A Histogram is a chart plotting a variable, against the number of occurrences in the variable category. It is a quick way to get information about a sample distribution without detailed statistical graphing or analysis. This plot will show you if your data values are centered (normally distributed), skewed to one side or the other, or have more than one 'mode' - localized distribution concentrations. Here, a histogram of node degree with the frequency on the Y-axis and the degree of vertices on X-axis is plotted.

INTERACTIVE PLOTTING:

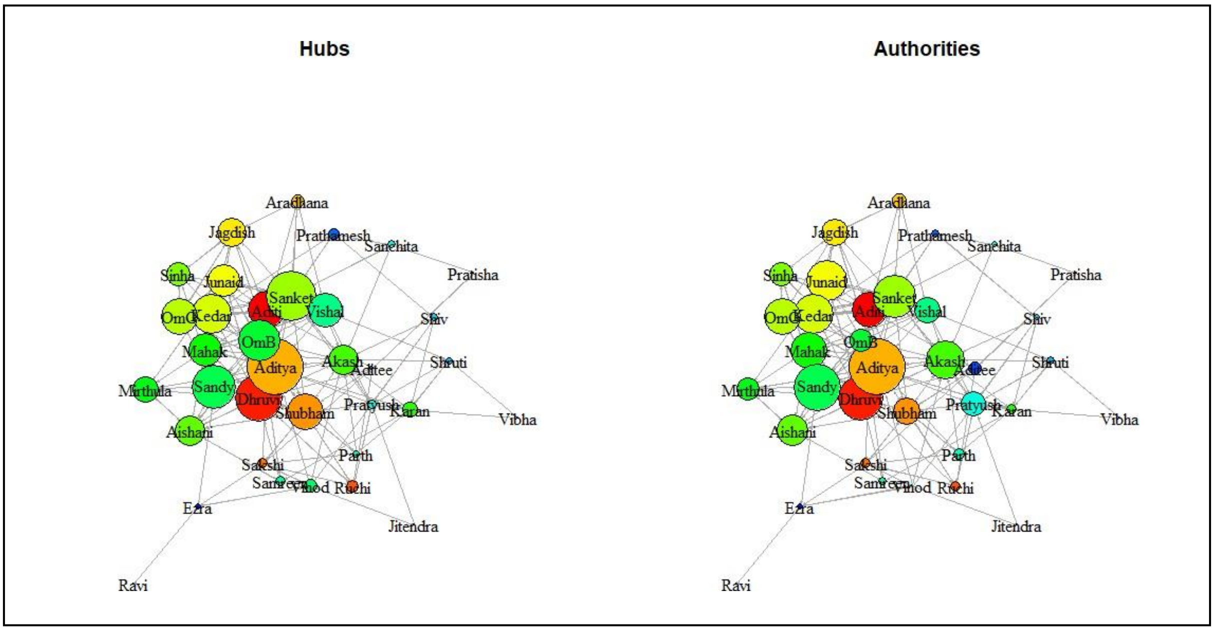
TkPlot





The tkplot command creates a new Tk window with the graphical representation of graph. tkplot and its companion functions serve as an interactive graph drawing facility. Not all parameters of the plot can be changed interactively right now though, eg. the colours of vertices, edges, and also others have to be pre-defined. tkplot is an interactive graph drawing facility. It is not very well developed at this stage, but it should be still useful.

HUBS AND AUTHORITIES:



Authority and hub values are defined in terms of one another in a mutual recursion. An authority value is computed as the sum of the scaled hub values that point to that page. A hub value is the sum of the scaled authority values of the pages it points to.

CONCLUSION

Social Network Analysis in R, Social Network Analysis (SNA) is the process of exploring the social structure by using graph theory. It is mainly used for measuring and analysing the structural properties of the network. It helps to measure social network relationships (Facebook, Twitter likes comments following etc..), Email connectivity, flows between groups, organizations, and other connected entities.

In this project we observe that the interaction between students of SY Data Science is quite frequent and Aditya being the CR of the class has the highest number of interactions with the students.

BIBLIOGRAPHY

Form Link for Data collection:

https://forms.gle/fzr1sSa4H8QMDTfX6

Information regarding SNA:

https://www.geeksforgeeks.org/social-network-analysis-using-r-programming/

https://medium.com/@615162020004/social-network-analysis-with-r-centrality-measure-86d7fa273574

https://medium.com/analytics-vidhya/social-network-analysis-in-r-part-1-ego-network-ab6b0d23ebc8