Data Mining – Home Assignment 3

Shweta Suran

- **Exercise 1**

  First, load the given data in R and saves in two different variables 'Edges & Nodes' and visualize the original graph as shown in Figure 1.

Graph_Of_Input

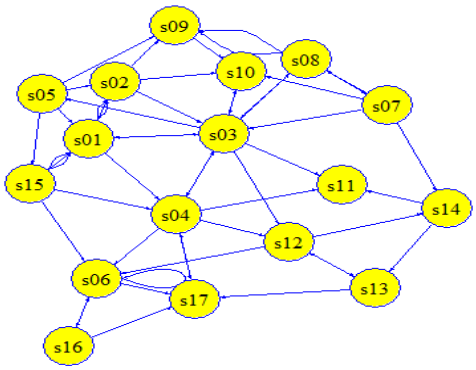
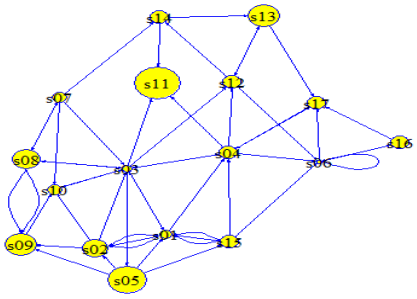


Figure 1: Original Graph

Local clustering coefficient

To calculate the Local Clustering Coefficient, first program computes an *Adjacency_matrix* and use the given formula for directed graph. Then it computes *Degree* & *Pair_Of_Friends_Of_Friends.* Finally, it computes *Local_Clustering_Coefficient* as shown in below output.

Local_Clustering_Coefficient



Output

Pairs_of_Friend_Of_Friend*:* 6 6 9 7 5 9 2 1 2 3 1 3 1 1 3 1 2
Degree*:* 10 7 13 9 5 7 5 6 5 5 3 6 4 4 6 3 5
Local_Clustering_Coefficient *:* 0.07 0.14 0.06 0.1 0.25 0.21 0.1 0.03 0.1 0.15 0.17 0.1 0.08 0.08 0.1 0.17 0.1

DegreeCentrality_DegreePrestige_NodeGregariousness

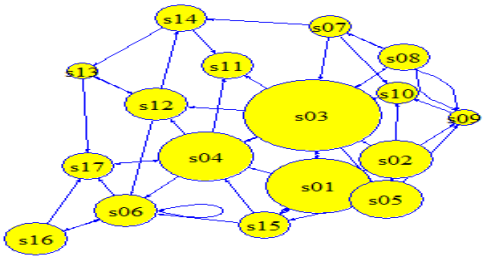
First, program computes the Nodes degree and after that it computes *In_NodeDegree* & Out_NodeDegree. Finally, it uses the following formula:

DegreeCentrality = NodeDegree/(NodeCount-1)

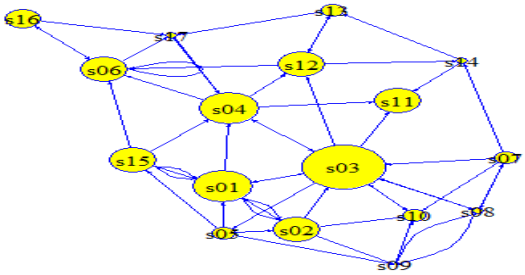
DegreePrestige = In_NodeDegree/(NodeCount-1)

NodeGregariousness = Out_NodeDegree/(NodeCount-1)

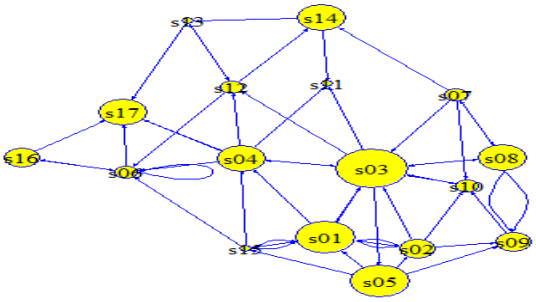
Degree_Centrality



Node_Gregariousness



Degree_Prestige



Output

Degree_Centrality: 0.625 0.4375 0.8125 0.5625 0.3125 0.4375 0.3125 0.375 0.3125 0.3125 0.1875 0.375 0.25 0.25 0.375 0.1875 0.3125
Degree_Prestige: 0.3125 0.1875 0.375 0.25 0.0625 0.3125 0.0625 0.125 0.25 0.25 0.1875 0.1875 0.125 0.125 0.125 0.0625 0.25
Node_Gregariousness: 0.3125 0.25 0.4375 0.3125 0.25 0.125 0.25 0.25 0.0625 0.0625 0 0.1875 0.125 0.125 0.25 0.125 0.0625

Closeness centrality & proximity prestige

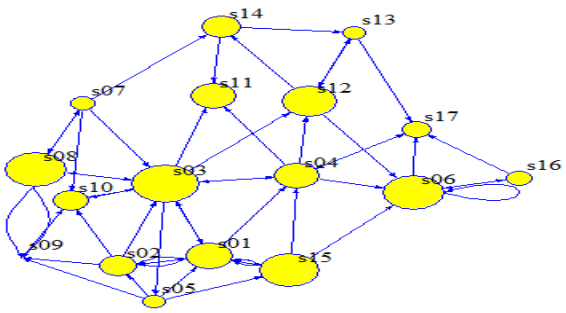
First, program computes the sum of shortest distance and the influence (based on identifying how many nodes are reachable from other nodes). Then it uses

the two formula to find out the Closeness centrality & proximity prestige.

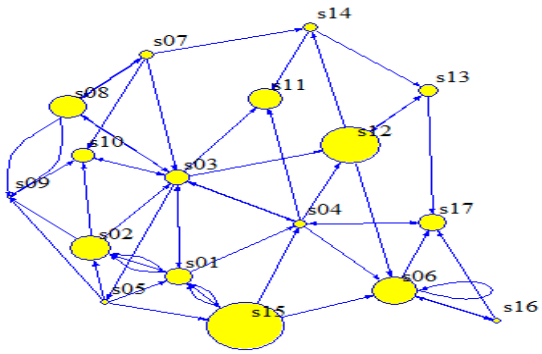
ClosenessCentrailty = (NodeCount - 1)/SumOfShrtDist

*ProximityPrestige = (influence^2) / ((NodeCount-1) * SumOfShrtDist)*

Closeness_Centrality



Proximity_Prestige



Output

ClosenessCentrailty: 0.03818616 0.03082852 0.05536332 0.03678161 0.03755869 0.01995012 0.04923077 0.05047319 0.02461538 0.05015674 0 0.02222222 0.02941176 0.02025316 0.04519774 0.0199005 0.03219316

ProximityPrestige: 0.2386635 0.3382707 0.2078287 0.1208333 0.2840376 0.08486596 0.6469231 0.3817035 0.2308654 0.3134796 0 0.08342014 0.2026654 0.1329905 0.4959393 0.1717195 0.1369467

Betweenness Centrality

To compute the Betweenness Centrality first program computes the fraction of pairs as 'FactOfPart' and the number of shortest paths that exist between pair of nodes. And finally, program uses the below formula:

*BetweenCent = FactOfParts / ((NodeCount - 1) * (NodeCount - 2))*

Output

BETWEENNESS_CENTRALITY: 0.09166667 0.0125 0.25 0.1291667 0.0125 0.0625 0.004166667 0.02916667 0 0.02916667 0 0.08333333 0.0125 0.004166667 0.008333333 0 0.05

Common neighbor based measure

First, program computes number of common pairs nodes have. Then number of IN & OUT consider as *Common neighbor based measure : OUT* & *Common neighbor based measure : IN*

Output

Common neighbor based measure : OUT
[s01 , s02] --> [s03] : 1
[s01 , s03] --> [s04] : 1
[s01 , s04] --> [s03] : 1
[s01 , s05] --> [s02 s15] : 2 ……
Common neighbor based measure : IN
[s01 , s02] ----**[s05] : 1
[s01 , s03] ----**[s02] : 1
[s01 , s04] ----**[s15 s03] : 2
[s01 , s05] ----**[s03] : 1 ……

Jaccard Measure

First, program computes the number of neighbors for all nodes. Then compute the *Jaccard Measure_In* & *Jaccard Measure_Out* by dividing the *Common neighbor based measure : OUT* & *Common neighbor based measure : IN* for each node pair by the number of all neighbors (ODD & IN)

Output

Jaccard Measure_In
[s01 , s02] -- 1 / 5 : 0.2
[s01 , s03] -- 1 / 9 : 0.1111111
[s01 , s04] -- 2 / 6 : 0.3333333
[s01 , s05] -- 1 / 4 : 0.25…….
Jaccard Measure_Out
[s01 , s02] -- 1 / 7 : 0.1428571
[s01 , s03] -- 1 / 10 : 0.1
[s01 , s04] -- 1 / 8 : 0.125
[s01 , s05] -- 2 / 6 : 0.3333333……

Morgan Index

For l = 1; the Morgan index is computed as the number of neighbors that reached from that node.

Output

Morgan_index:First_Order
[s01] -- 4 [s02] -- 4 [s03] -- 7 [s04] -- 5
[s05] -- 4 [s06] -- 3 [s07] -- 4 [s08] -- 3
[s09] -- 1 [s10] -- 1 [s11] -- 0 [s12] -- 3
[s13] -- 2 [s14] -- 2 [s15] -- 3 [s16] -- 2
[s17] -- 1

Wiener index

Program computes this index as "Sum of the pairwise shortest path distances between all pairs of nodes in the graph".

Output

Wiener_Index: 8210

Hosoya index

To compute this index, program use the *tidyverse* library. And uses the following definition “equal to the number of valid pairwise node-node matchings in the graph”.

Output

Hosya_Index: 916

Estrada index

To compute this index formula is given in lecture ppt used.

Output

Estrada_Index :30.01872

Cirquit rank

Below formula is used to compute the Cirquit Rank

CirquitRank = EdgeCount - (NodeCount - 1)

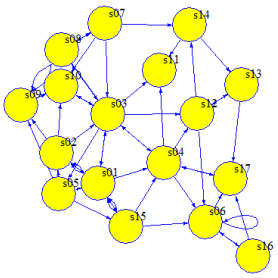
Output

CirquitRank_(For:Undirected_Graph) 36

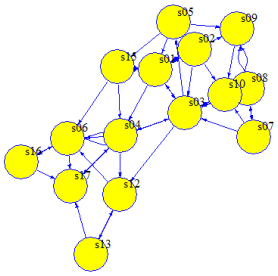
- **Exercise2**

First, load the given data in R and saves in two different variables ‘Edges & Nodes’. Then produce 2 subgraphs: SubGraph_*First* & *SubGraph_Second* using igraph. After this program computes the Maximum Cliques using the function *max_cliques.* In next step, program does matching of all *max_cliques* in both subgraphs and find out the common *max_cliques* in both graph. Finally produce the *MaximumCommSubgrp* as shown in below figure.

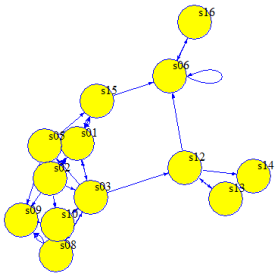
Original



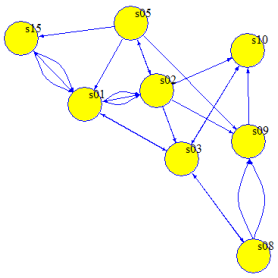
SubGraph-First



SubGraph-Second



MaximumCommSubgrp



- **References**
- Local clustering coefficient- https://www.coursera.org/lecture/python-social-network-analysis/clustering-coefficient-ZhNvi
- Degree-and-closeness-centrality - https://fr.coursera.org/lecture/python-social-network-analysis/degree-and-closeness-centrality-noB1S
- Tutorials Point-https://www.tutorialspoint.com/
- Statistical Network Analysis with igraph https://sites.fas.harvard.edu/~airoldi/pub/books/BookDraft-CsardiNepuszAiroldi2016.pdf
- Tutorial Gateway-https://www.tutorialgateway.org
- how-to-find-common-subgraph-using-igraph https://stackoverflow.com/questions/27672022/how-to-find-common-subgraph-using-igraph