Stat Network Analysis - Final Project

19203753

16/08/2020

Aim : The goal of the exhibition was to investigate the mechanisms of contagion.

Let us clear encironment variables before we start:

```r
rm(list=ls())
```

Load the data set: The data set contains Adjacency matrix X, interaction matrix Y, object "layout" contains coordinates of the nodes and the vector time_frame indicates in which of three time periods each individual interacted most frequently.

```r
load("C:/Users/Nisarga Gangadhar/Downloads/dataset.RData")
```

Load the required libraries for all the further computations:

```r
library(igraph) # for network analysis
library(blockmodels) #for fitting stochastic block model
library(intergraph)
library(mclust) # for clustering
library(Bergm) #random graph model
library(ergm) # for exponential random graph model
library(latentnet)#latent position model
```

convert the given X and Y matrix into graph objects.

```r
Xgraph <- graph_from_adjacency_matrix(X,mode = "undirected")
Ygraph <- graph_from_adjacency_matrix(Y,mode = "undirected",weighted=TRUE)
#since Y is weighted
```

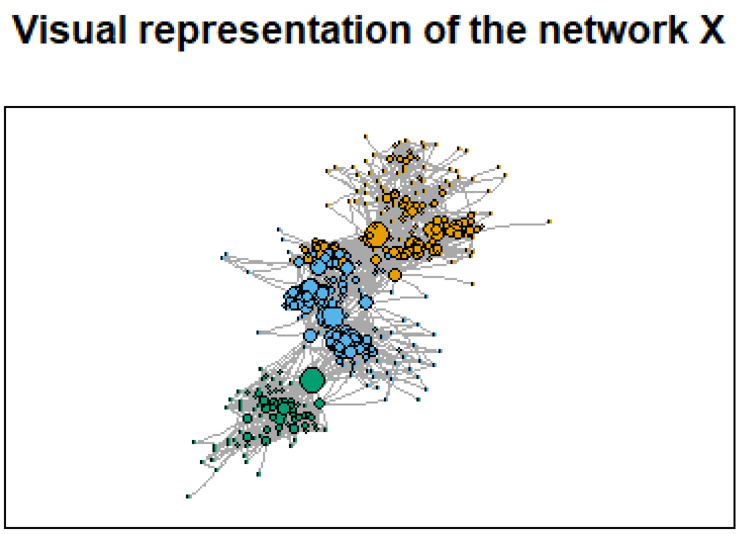
Question 1:

```r
binary_degree = degree(Xgraph) # binary degree
wt_degree = degree(Ygraph) #weighted degree

#find the top 5 nodes number with highest binary degree
ind=order(binary_degree,decreasing = TRUE)[1:5]

#printing their weights
cat("The weighted degrees of the first 5 nodes with highest binary degrees-
",wt_degree[ind])
```

```
## The weighted degrees of the first 5 nodes with highest binary degrees- 50
47 43 43 34
```

Question 2:

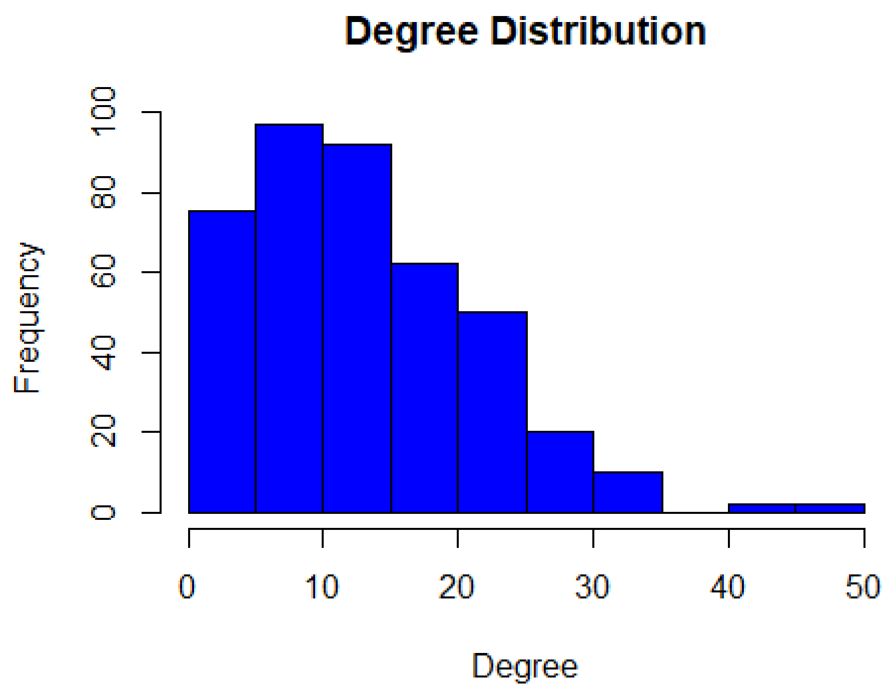
```
# weighted_degree is reduced by 0.3 for a better and clear graph although
proportion remains same.
plot(Xgraph,layout=layout,vertex.size=0.3*wt_degree,edge.curved=0.1,vertex.co
lor=time_frame,vertex.label=NA,main="Visual representation of the network
X",edge.width=1)
box(which = "plot",lty = "solid")
```



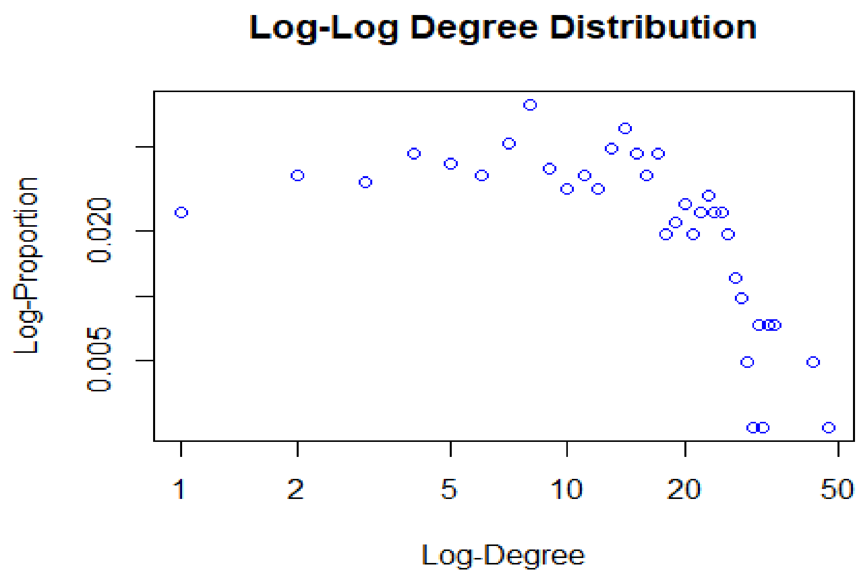
Question 3:

All the heavy and right tailed distributions possess powerlaw behavior, the distribution below is positively skewed that is it has long right tail, hence it satisfies powerlaw behavior. Also, in the log-log scale we see a straight line with a negative slope, therefore there is evidence of powerlaw behavior.

```
#degree distribution in natural scale:
d.dist <- degree(Xgraph)
hist(d.dist,col="blue",xlab="Degree", ylab="Frequency",main="Degree
Distribution")
```

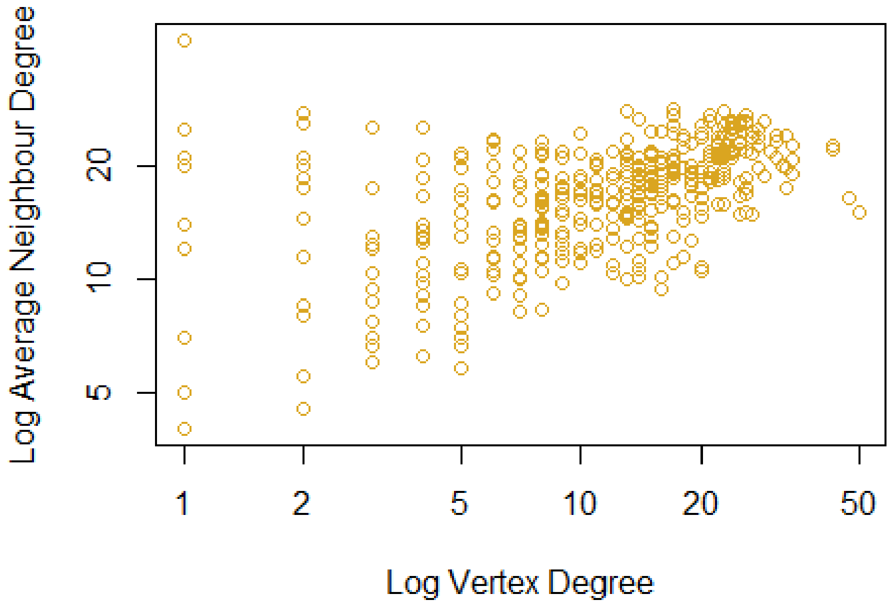


```
#degree distribution in log-log scale:
dd.dist <- degree.distribution(Xgraph)
d <- 1:max(d.dist)-1
ind <- (dd.dist != 0)
plot(d[ind], dd.dist[ind], log="xy", col="blue",xlab=c("Log-Degree"),
ylab=c("Log-Proportion"),
     main="Log-Log Degree Distribution")
```



Question 4:

```
#average total degree
cat("The average total degree is",mean(binary_degree),"\n")

## The average total degree is 13.4878

#clustering coefficient
cat("The clustering coefficient is",transitivity(Xgraph),"\n")

## The clustering coefficient is 0.4356933

#average path length
cat("The average path length is",average.path.length(Xgraph,unconnected =
T),"\n")

## The average path length is 3.630855

#average degree of neighbors of node
a.nn.deg <- graph.knn(Xgraph,V(Xgraph))$knn

#plot
plot(d.dist, a.nn.deg, log="xy",
col="goldenrod", xlab=c("Log Vertex Degree"),
ylab=c("Log Average Neighbour Degree"))
```



```
print("The summary of average degree of the neighbours of a node as a
function of the node’s degree:")
```

```
## [1] "The summary of average degree of the neighbours of a node as a
function of the node’s degree:"
```

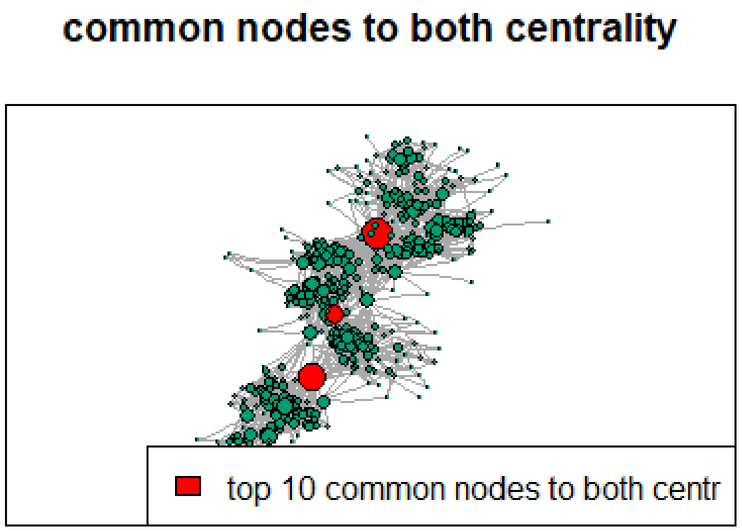
```
summary(a.nn.deg)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    4.00   13.18   17.29   17.10   20.98   43.00
```

Comments: From the summary of the last result we can say that the minimum value is 4 nodes and the maximum value is 43 nodes. The mean is found out to be approximately 17 nodes which means half of the nodes have an average degree of the neighboring node as 17. We can also find from the plot of Log average neighbor degree v/s Log Vertex Degree as shown above.

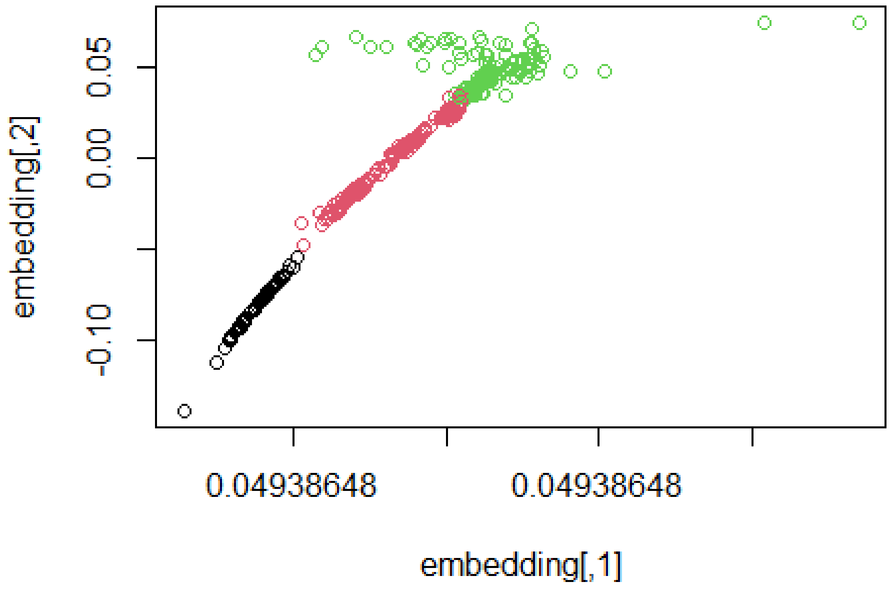
Question 5:

```
#page_rank centrality
page_cent <- page_rank(Xgraph)$vector
#ordering page rank centrality in decreasing manner
ind1 <- order(page_cent,decreasing = T)

#betweenness centrailty
bet_cent <- centr_betw(Xgraph)$res
#ordering betweenness centrality in decreasing manner
ind2 <- order(bet_cent,decreasing=T)

#taking top 10 accoring to both centrailty
ind3=intersect(ind1[1:10],ind2[1:10])

#creating a col vector of colours different for top 10 centrality
col=rep(3,410)
col[ind3]="red"

#plotting
plot(Xgraph,layout=layout,vertex.size=2000*page_cent,vertex.color=col,edge.cu
rved=0.1,vertex.label=NA,main="common nodes to both centrality")
legend("bottomright",legend = "top 10 common nodes to both centr",fill =
"red")
box(which = "plot",lty = "solid")
```

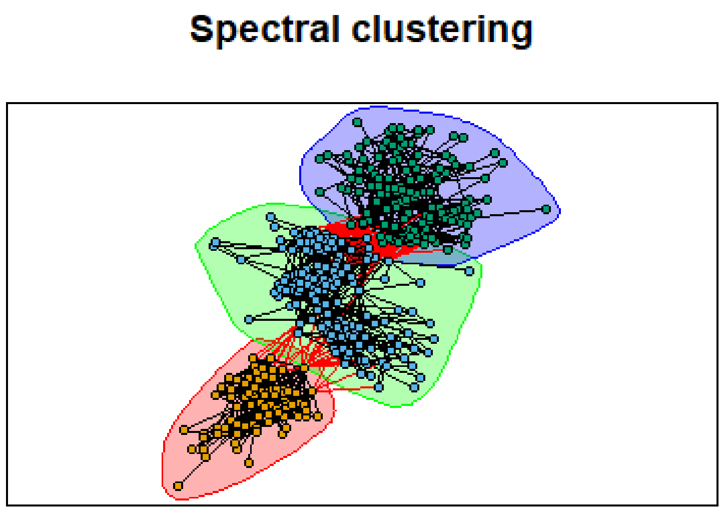


Question 6:

```r
#setting seed to get same clustering everytime
set.seed(1)

L <- graph.laplacian(Xgraph) # calculate the Laplacian matrix

eigen_decomposition <- eigen(L) # finds all eigenvalues and eigenvectors

n_nodes <- length(V(Xgraph)) # extract the number of nodes from the network

n_dimensions <- 3

selected_dimensions <- n_nodes:(n_nodes - n_dimensions + 1) # consider the
sequence of indices of the eigenvalues that will be considered.

embedding <- eigen_decomposition$vectors[, selected_dimensions]#Embedding for
k-means

memberships <- kmeans(embedding, n_dimensions,nstart = 100)$cluster # the
number of groups is chosen at this stage, so it may be determined using the
clustering algorithm

plot(embedding, col = memberships)
```



```
res <- make_clusters(Xgraph, memberships)
plot(x = res, y = Xgraph, layout =
layout,vertex.size=5,vertex.label=NA,main="Spectral clustering")
box(which = "plot",lty = "solid")
```



```
# Summary.

predicted=table(time_frame,memberships)
print(predicted)

##            memberships
## time_frame   1   2   3
##          1   0  21 143
##          2   0 158   1
##          3  86   1   0

# Misclassified nodes.

cat("The number of nodes that are misclassified is",sum(table(time_frame,
memberships)) - sum(diag(table(time_frame, memberships))))

## The number of nodes that are misclassified is 252
```

None of the members in the clusters 1 and 3 are correctly classified because the diagonal elements of cluster 1 and 3 are 0 . In cluster 2, only 1 member is misclassified.The number of nodes that are misclassified are 252.

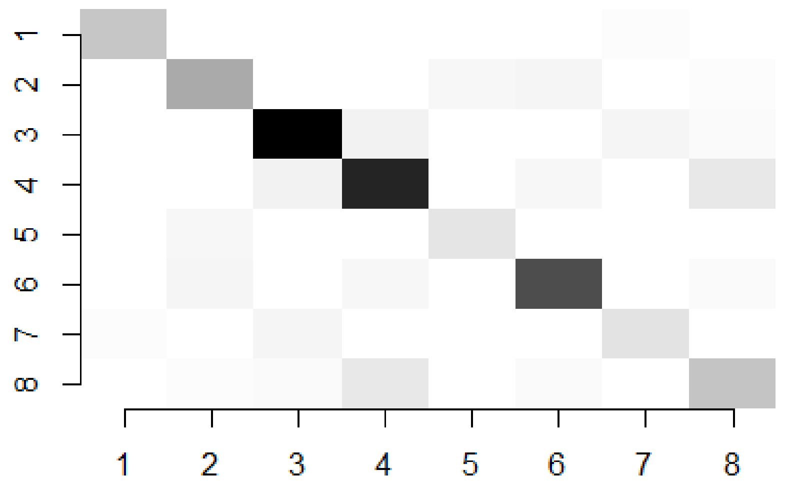
Question 7:

The maximum likelihood estimates for the probability that any two nodes allocated to the first time frame are joined with an edge is 0.07040251.

```
# Extracting the subgraph with time frame=1.

sub <- induced_subgraph(Xgraph, time_frame==1)

# Finding the nodes which are joint and unjoint with an edge.

joint <- gsize(sub)
unjoint <- (gorder(sub) * (gorder(sub)-1)/2) - joint

# Calculating the probability.

prob <- joint/(joint + unjoint)
print(prob)

## [1] 0.07040251
```

Question 8:

```
# Fitting a stochastic block model on X. We use bernoulli distribution as our
adjacency matrix X is comprised of 0's and 1's only.

sbm <- BM_bernoulli(membership_type = "SBM_sym",
                    adj = X,
                    verbosity = 0,
                    plotting="",
                    explore_min=2,
                    explore_max = 8)

# Extracting the estimates.

sbm$estimate()

# Choosing a best model based on Integrated Completed Likelihood value.

K_star=which.max(sbm$ICL)
cat("The Best Model according to the Integrated Completed Likelihood
criterion is with",K_star,"\n")

## The Best Model according to the Integrated Completed Likelihood criterion
is with 8

# Posterior probabilities for group memberships.

soft_clustering <- sbm$memberships[[K_star]]$Z

# Maximum a-posteriori clustering.

hard_clustering <- apply(soft_clustering, 1, which.max)

# Connection Pribabilities Plot.

sbm$plot_parameters(K_star)
```



```
# Comparison.

table(hard_clustering,time_frame)
```

```
##                time_frame
## hard_clustering  1  2  3
##               1  0  0 75
##               2 56  4  0
##               3  0 31  0
##               4  0 30  0
##               5 91  1  0
##               6 17 17  0
##               7  0 37 12
##               8  0 39  0
```

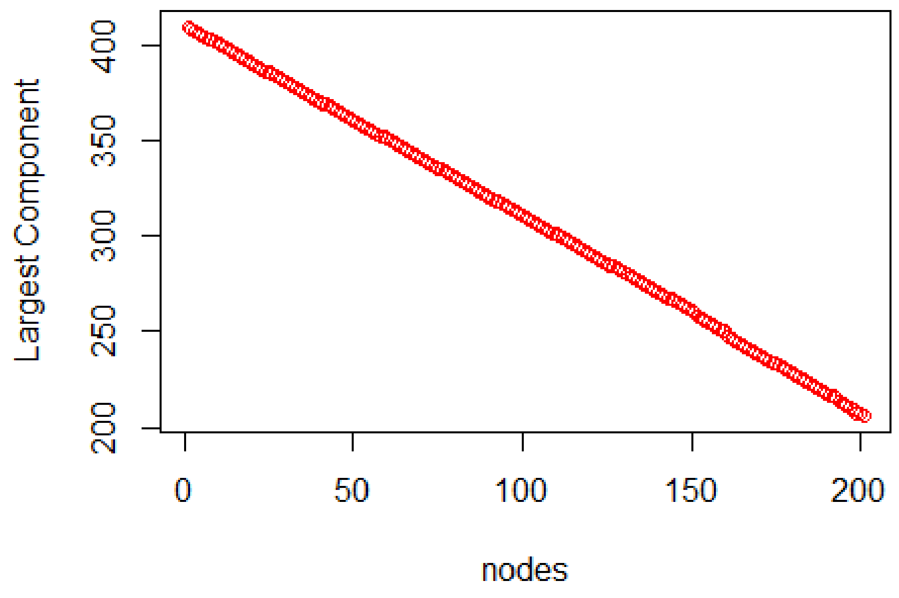
Comparing the clusters with time-frame: After fitting a Stochastic Block Model , 8 is optimal, We observe that 56 nodes,91 nodes,17 nodes of time frame 1 are in cluster 2, cluster 5 & 6 respectively.Except cluster 2, we observe a distribution of values of time frame 2 in the remaining 7 clusters.75 nodes of time frame 3 are in cluster 1 and the remaining 12 nodes are in cluster 12.

Assortative and Disassortative mixing? and connectivity behavior: The Connection Probabilities Plot's basic interpretation is, darker the colour, stronger the community structure i.e an indication of assortative mixing.
We observe darkest colour in cluster 3 and 4 hence they exhibit the strongest community structure followed by cluster 6. These clusters exhibit stong assortative mixing while clusters 1, 2 and 8 also exhibit assortative mixing but not as strong as these clusters.
We observe that clusters 5 and 7 seem to be neutral, i.e. neither assortative mixing nor disassortative mixing.

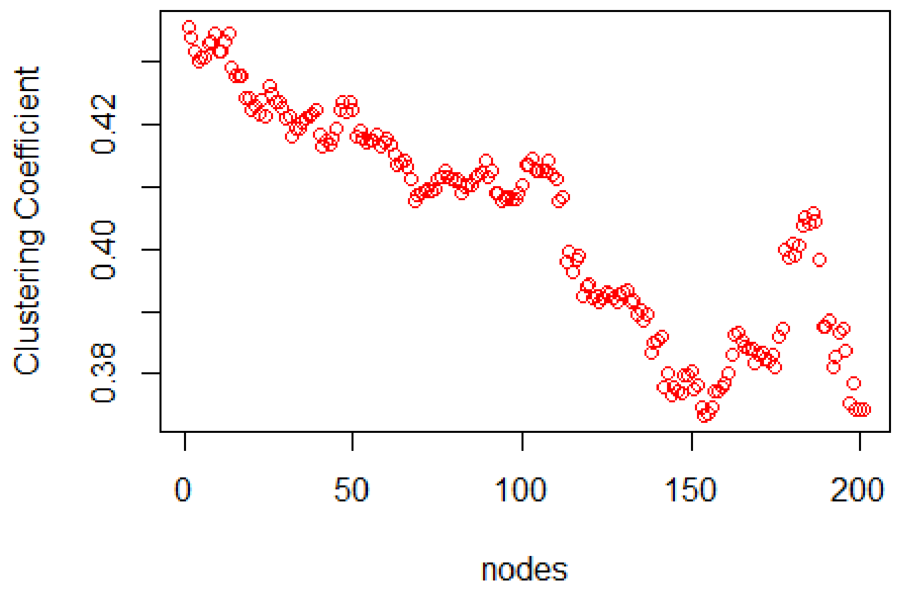
Question 9:

```
# Setting a same seed to obtain the same results over repeated runs.

set.seed(1)

# Number of Nodes to remove during simulation.

n_nodes_to_remove <- 200

# Matrices for storing the values.

largest_comp= matrix(NA,X,n_nodes_to_remove+1)
clustering_coeff=matrix(NA,X,n_nodes_to_remove+1)
mod=matrix(NA,X,n_nodes_to_remove+1)

# Storing the first value for each statistics in the above NA matrices.

largest_comp[1]=max(igraph::components(Xgraph)$csize)
```

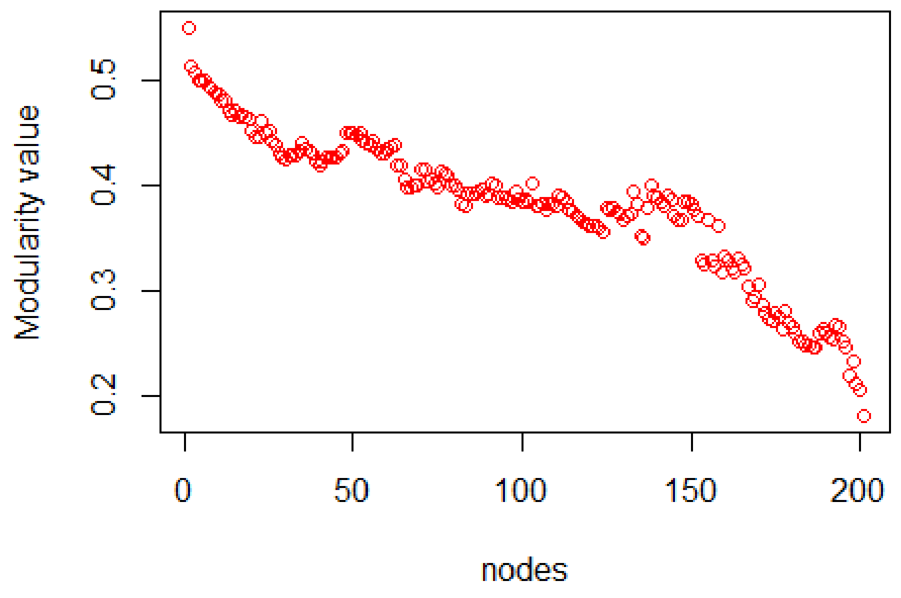
```r
clustering_coeff[1]=transitivity(Xgraph)
mod[1]=modularity(Xgraph,time_frame)

# Removal of each nodes based on the given condition.

for (i in 1:n_nodes_to_remove)
  {

    # Removing using runif.

    remove_me <- runif(1,1,length(V(Xgraph)))

    # Deleting the vertex from the graph and storing the new statistics.

    Xgraph <- igraph::delete.vertices(Xgraph, remove_me)
    largest_comp[i+1]=max(igraph::components(Xgraph)$csize)
    clustering_coeff[i+1]=transitivity(Xgraph)
    mod[i+1]=modularity(Xgraph,time_frame)

}

plot(largest_comp,ylab = "Largest Component",col="red",xlab="nodes")
```



```r
plot(clustering_coeff,ylab = "Clustering Coefficient",col="red",xlab =
"nodes")
```



```r
plot(mod,ylab = "Modularity value",col="red",xlab = "nodes")
```

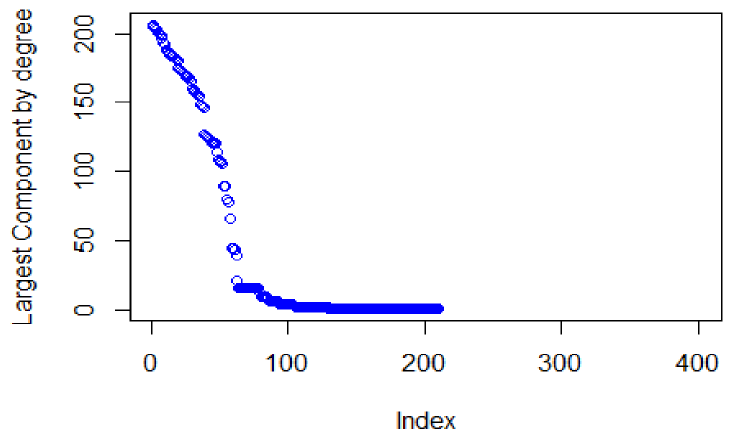


The plot of largest component shows a strong negative linear relationship which says as we keep on removing nodes one by one, size of the largest component linearly decreases. An overall decreasing pattern is observed in the clustering coefficient value with a few upward values in the trend as we keep on removing a node sequentially.
A steadier decreasing pattern than that of clustering coefficient is observed in the modularity value.

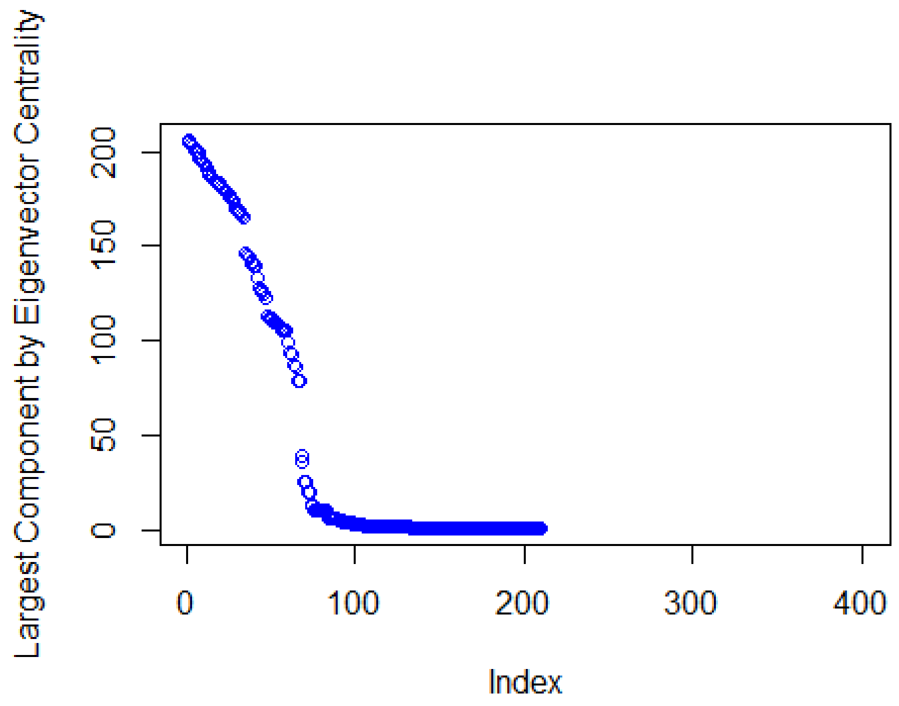
Question 10:

```r
# Setting a same seed to obtain the same results over repeated runs.

set.seed(1)

# Removing by degree.

gra<-Xgraph

# Number of Nodes to remove during simulation.

n_nodes_to_remove <- 400

# Matrix for storing the largest component.

largest_comp_degree= matrix(NA,X,n_nodes_to_remove+1)

# Storing the first value for the largest component in the above NA matrix.

largest_comp_degree[1]=max(igraph::components(gra)$csize)

# Removal of each nodes based on the given condition.

for (i in 1:n_nodes_to_remove)
  {

    # Removing the node with the highest degree.

    remove_me <- which.max(igraph::degree(gra))

    # Deleting the vertex from the graph and storing the new statistics.

    gra <- igraph::delete.vertices(gra, remove_me)
    largest_comp_degree[i+1]=max(igraph::components(gra)$csize)
}

# Removing by eigenvector centrality.

gra<-Xgraph
```

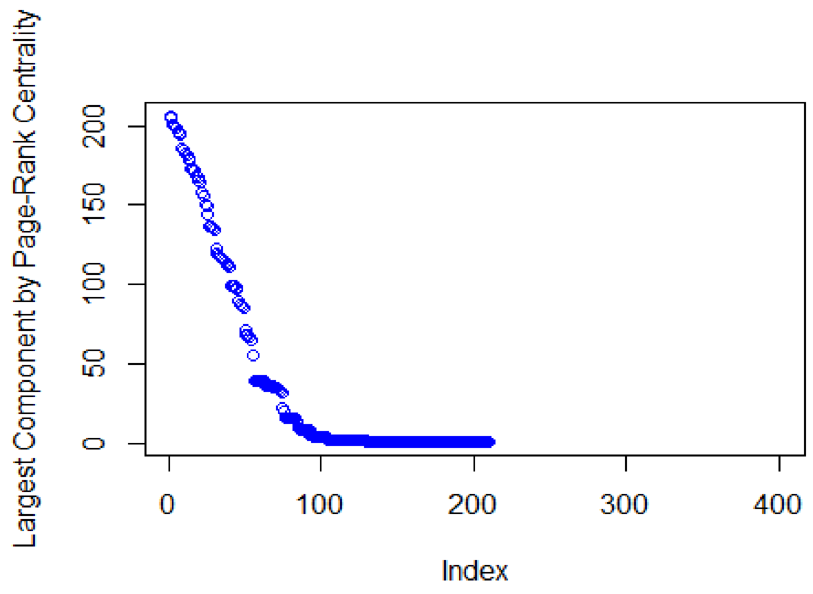
```r
largest_comp_ec= matrix(NA,X,n_nodes_to_remove+1)
largest_comp_ec[1]=max(igraph::components(gra)$csize)

for (i in 1:n_nodes_to_remove)
  {

    # Removing the node with the highest eigen centrality value.

    remove_me <- which.max(eigen_centrality(gra)$vector)

    gra <- igraph::delete.vertices(gra, remove_me)
    largest_comp_ec[i+1]=max(igraph::components(gra)$csize)
}

# Removing by Page-Rank centrality.

gra<-Xgraph
largest_comp_pr= matrix(NA,X,n_nodes_to_remove+1)
largest_comp_pr[1]=max(igraph::components(gra)$csize)
for (i in 1:n_nodes_to_remove)
  {
    # Removing the node with the highest Page-Rank centrality value.

    remove_me <- which.max(page_rank(gra)$vector)

    gra <- igraph::delete.vertices(gra, remove_me)
    largest_comp_pr[i+1]=max(igraph::components(gra)$csize)
}

# Plotting.

plot(largest_comp_degree,ylab = "Largest Component by degree",col="blue")
```



```r
plot(largest_comp_ec,ylab = "Largest Component by Eigenvector Centrality",col="blue")
```



```r
plot(largest_comp_pr,ylab = "Largest Component by Page-Rank Centrality",col="blue")
```

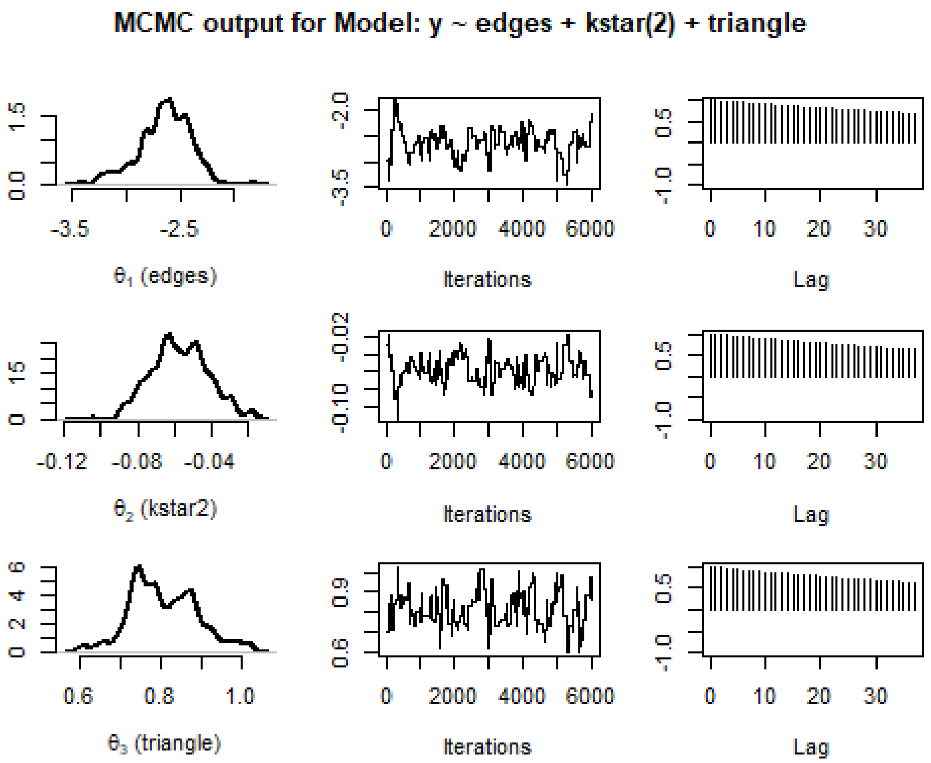


In the plot By Degree We observe two dips (big drop) in the size of the largest components in the range of nodes 120-180 and observe a constant value of the largest component soon after crossing 200 nodes.
In the plot by Eigenvector Centrality we observe a similar trend with 2 dips in the range 160-200 with a constant value of the largest component from 220 nodes.
In the plot by Page-Rank Centrality we observe that the first dip is observed early in the range 110-130 and then again a steady pattern is observed followed by the second dip in the range 170-190 and observe a constant value of the largest component soon after crossing 220 nodes.

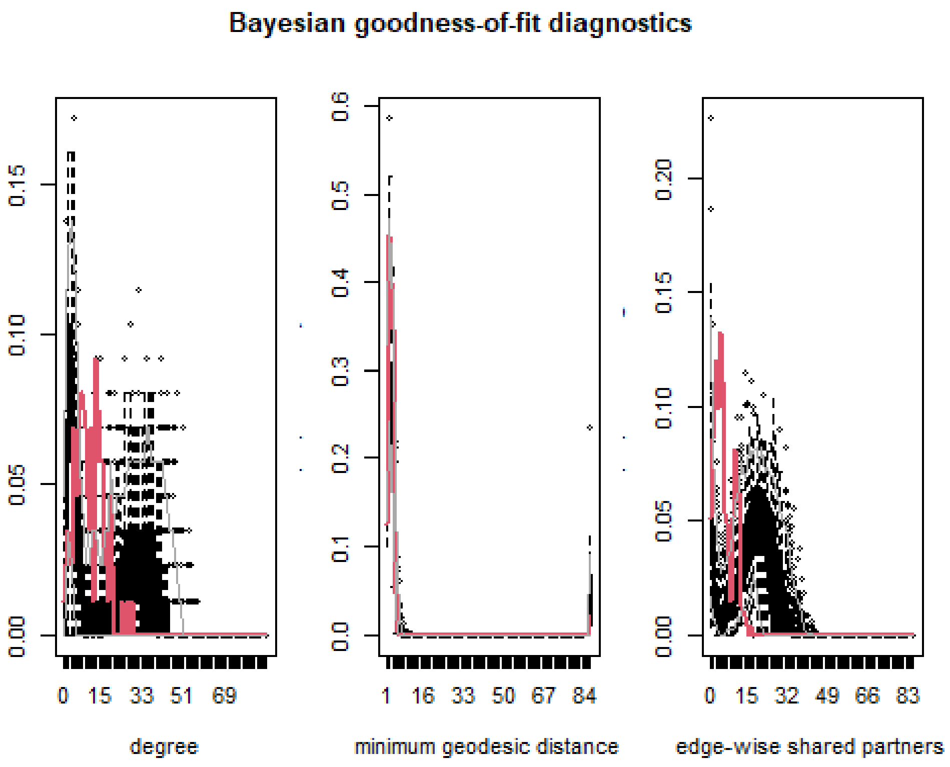
Question 11:

```
# Considering a subnetwork of the nodes that mostly interact in the third
time frame.

time_frame_3=which(time_frame==3)

# Fitting an Exponential Random Graph Model.

X_ERGM=as.network(X[time_frame_3,time_frame_3],directed=FALSE,matrix.type =
"adjacency")
model= X_ERGM ~ edges + kstar(2) + triangle
x_bergm=bergm(model)

##  > MCMC start

# Analysing the summary statistics.

summary(x_bergm)

##
##  Posterior Density Estimate for Model: y ~ edges + kstar(2) + triangle
##
##                          Mean         SD    Naive SE Time-series SE
## theta1 (edges)    -2.63696417 0.25435601 0.003283722    0.047166537
## theta2 (kstar2)   -0.05683622 0.01515824 0.000195692    0.002488998
## theta3 (triangle)  0.80998852 0.08328867 0.001075252    0.013487866
##
##                          2.5%         25%         50%         75%
97.5%
## theta1 (edges)    -3.23422363 -2.79444143 -2.61584863 -2.46072088 -
2.22006509
## theta2 (kstar2)   -0.08580896 -0.06718301 -0.05710219 -0.04711966 -
0.02781264
## theta3 (triangle)  0.65550379  0.74813198  0.79806394  0.86799049
0.99286618
##
##  Acceptance rate: 0.04
##
##
```

```
plot(x_bergm)
```



```
goodness_fit= bgof(x_bergm)
```



From the summary statistics, we observe that the overall acceptance rate is 0.04.
The sequence of the plot column wise is as, the estimated posterior density for the parameters (left), the trace plots of the Markov chains(middle), and the autocorrelation plot for the values explored by the Markov chain (right).Ideally, the plots in the middle and on the right columns should characterise a "white noise" process, i.e. no patterns in the trace plots and very small autocorrelation values after the first lag.
We observe that the density for the edges is in the range (-3.4,-2), the density for the kstart2 variable is in the range (-0.15,-0.02) and the density for the traingle is (0.5,1.1).
The MCMC plot does not follow any pattern.The autocorrelation plot does not have a very small value after the first lag indicating that we need more iterations to get a proper interpretation of this model.
Hence, based on the above plotting, these number of iterations are not enough to get reliable interpretation. From the goodness of fit plot, we can give a similar conclusion that our model is not a good fit for the given dataset.

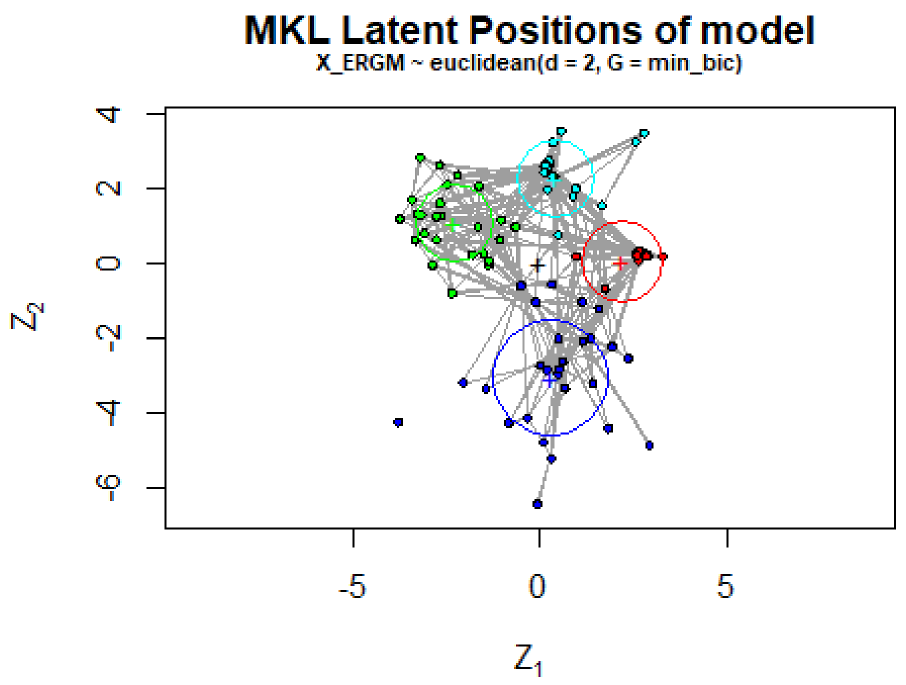
Question 12:

We run a for loop considering up to 4 clusters, and model with minimum BIC is chosen as optimal. Here model with 4 clusters has minimum BIC value.

```r
# Setting a same seed to obtain the same results over repeated runs.

set.seed(1)

# Storing the BIC values.

bic_values=rep(NA,4)

# Finding the optimum cluster based on minimum BIC value.

for( i in 1:4){
  model=ergmm(X_ERGM ~ euclidean(d=2, G=i))
  bic_values[i]= bic.ergmm(model)$overall
}



# Storing the minimum BIC value.

min_bic=which.min(bic_values)

# Fitting the model again on the minimum BIC value for analysis.

model=ergmm(X_ERGM ~ euclidean(d=2, G=min_bic))

# Analysis.
bic.ergmm(model)
```

```
## $Y
## [1] 1680.663
##
## $Z
## [1] 721.7092
##
## $sender
## [1] 0
##
## $receiver
## [1] 0
##
## $sociality
## [1] 0
##
## $overall
## [1] 2402.372
```

```
plot(model)
```



Above is the plot of a model with 4(minimum BIC) clusters ,The three circles indicate the components means and component variances for the clusters. They can be interpreted as the location and dispersion of each of the communities in the social space. Note that the given classes correspond match really well the clusters obtained with this procedure. Also, it is interesting to note that we would obtain a stochastic block model structure if the nodes were all located in one of the cluster centres.