Kevin R. Clemmons
(oduprogrammer16)
CS432 — Spring 2016

Assignment 5

March 4, 2016

All files mentioned in this file are uploaded into the *github* repository.

The LaTeXcode invovled in the generation of this document was aided by the example code provided in the links that Dr. Nelson sent out on January 17, 2016[1].

This document was compiled on `www.sharelatex.com`

Problem 1

To determine whether or not the result of the split of the karate club could have been predicted, an R-Script called *karateClubSimulation*, was created which implements the Girvan-Newman algorithm which was discussed in class[3]. The original code code for this implementation was found via the following url `https://github.com/maturban/cs595-f13/blob/master/assignment6/rcode.r`, modified to accept a datafile instead of retrieveing the data on the internet and the variables were renamed for clarification.

The steps performed in the Girvan-Newman algorithm are as follows[3]:

1. Calculate the edge betweeness for all the edges
2. Determine the edge with the highest edge-betweeness.
3. Delete the edge with the highest edge-betweenness
4. Repeat until the number of groups have been created from the initial graph.

The code in Listing 1 shows the implementation of a single iteration of the Girvan-Newman algorithm.

Listing 1: Single Iteration of Girvan-Newman Algorithm

```
# Calculate betweeness of all edges
edgeBetweeness <- edge.betweenness(karate_network)

# determine the highest edge betweeness
highestEdgeBetweeness <- max(edgeBetweeness)

# Find the index with the highest edge betweeness
edgeToDelete <- match(c(highestEdgeBetweeness),edgeBetweeness)

# Delete the edge with the highest edgebetweeness

# Get the start and end nodes
start <- get.edgelist(karate_network)[edgeToDelete,1] # The starting vertex
end <- get.edgelist(karate_network)[edgeToDelete,2]# The ending

# Delete the edge with the highest edgebetweeness
karate_network <-
    delete.edges(karate_network,E(karate_network,P=c(start,end)))
```

To draw the graph, igraph has a function which can determine groups or clusters of nodes as seen below in Listing 2[2, 4].

Listing 2: Code for determining the number of clusters

```
1 # Get the number of groups of nodes
2 numberOfClusters <- clusters(karate_network)['no']
```

The code in Listing 1 and Listing 2 exist inside a loop. The loop will terminate when the number in line 2 of Listing 2 is equal to two, indicating that two clusters of nodes have been created. Once an iteration is complete, a graph will be drawn depicting the current state of the graph using the code in Listing 3.[4].

Listing 3: Code for Drawing the Current State of the Graph

```
# Get list communities
communityList <- leading.eigenvector.community(karate_network,steps=1)

# Vertex Coloring
V(karate_network)$color <-
    ifelse(communityList$membership==1,"orange","yellow")

```

```
# Deterline scale of graph
scale_graph <- function(vertex,alpha,beta){
      vertex <- vertex-min(vertex); vertex <-vertex/max(vertex); vertex <-
         vertex * (beta-alpha); vertex+alpha
}


E(karate_network)$color <- "grey"
# Plot the graph
tkplot(karate_network,layout=layout.kamada.kawai,vertex.label.font=2)
```

Once the algorithm is complete, a list all the edges in the order that they were deleted will be produced along with their computed edge-betweenness will be produced in a manner that can be copied and pasted into a table in LaTeX.

The data that was used in the program, is written in a graph modeling format, models weighted graph of social interactions among friendship within a University in the United States during the 1970s [6, 5].

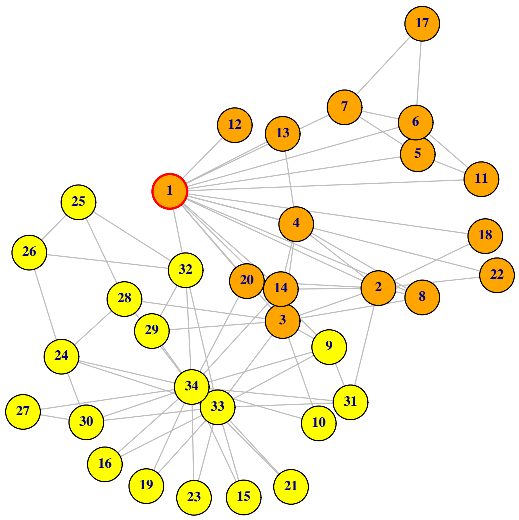
The edges and the order in which they were deleted can be found in Table 1.

Iteration Number	Starting Vertex	Ending Vertex	Edge-Betweeness
1	1	32	71.3
2	1	3	66.8
3	1	9	77.3
4	14	34	72.0
5	20	34	123.2
6	3	33	100.2
7	2	31	143.6
8	2	3	109.2
9	3	4	107.6
10	3	8	142.7
11	3	14	285

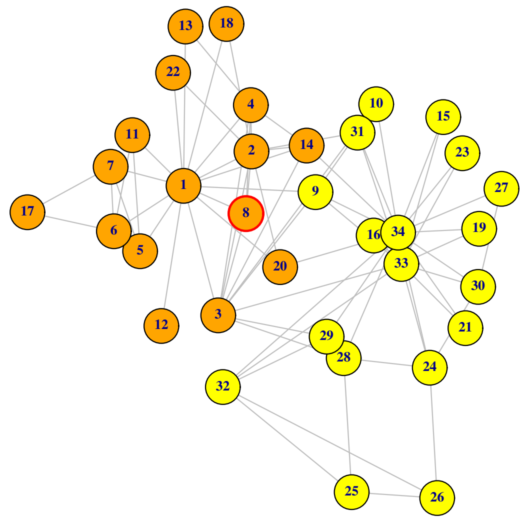
Table 1: Order of edges deleted during each iteration of the Girvan-Newman Algorithm

The following figures below visually depict each iteration of the Girvan-Newman algorithm.

This space intentionally left blank

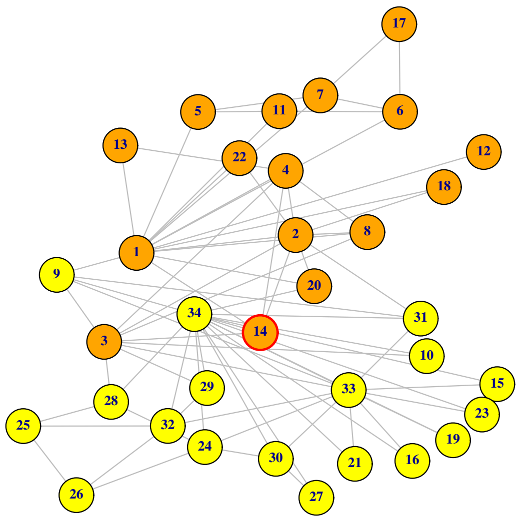


(a) Initial Graph

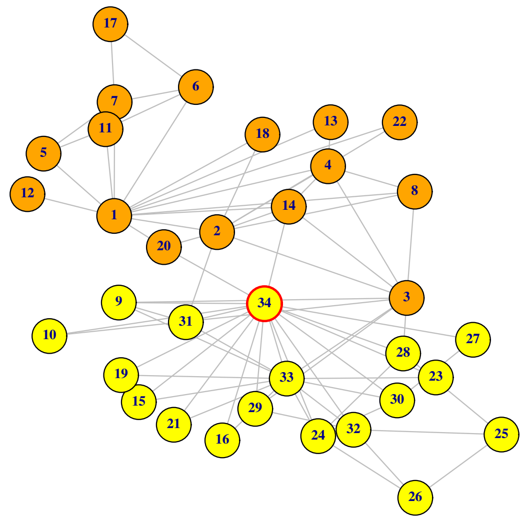


(b) After removing 1 − > 32

Figure 1: Inital Graph and Iteration 1

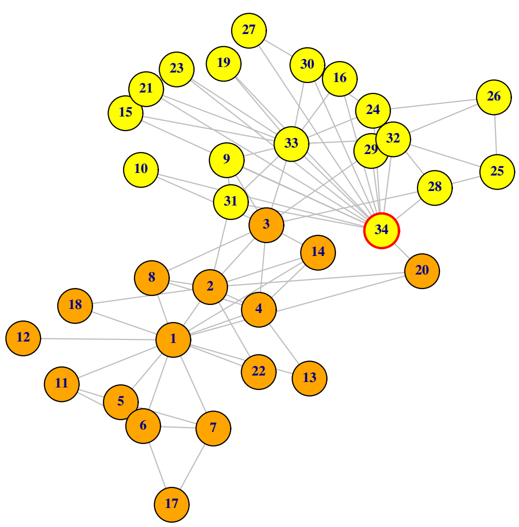


(a) After Removing 1 − > 3

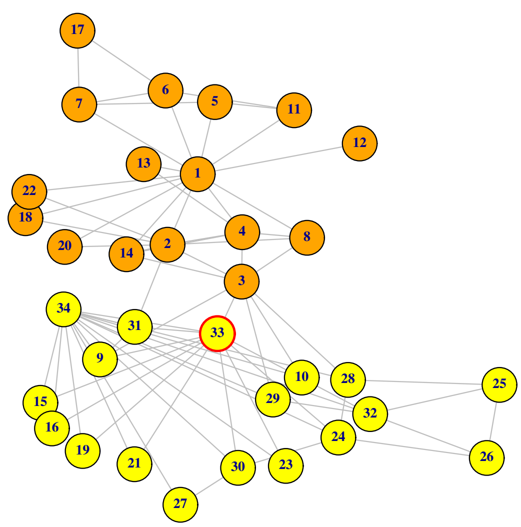


(b) After removing 1 − > 9

Figure 2: Iterations 2 and 3

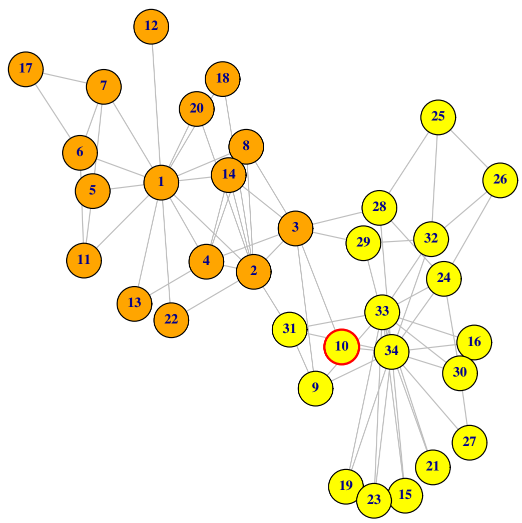


(a) After Removing 14 − > 34

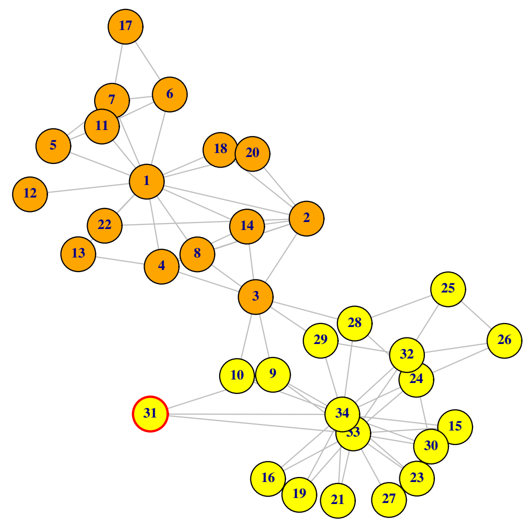


(b) After removing 20 − > 34

Figure 3: Iterations 4 and 5

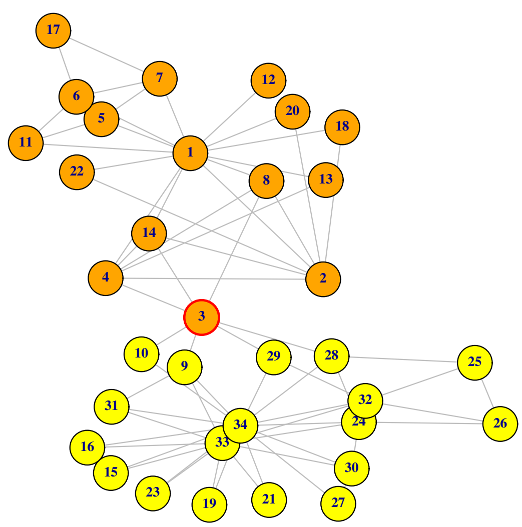


(a) After Removing 3 − > 33

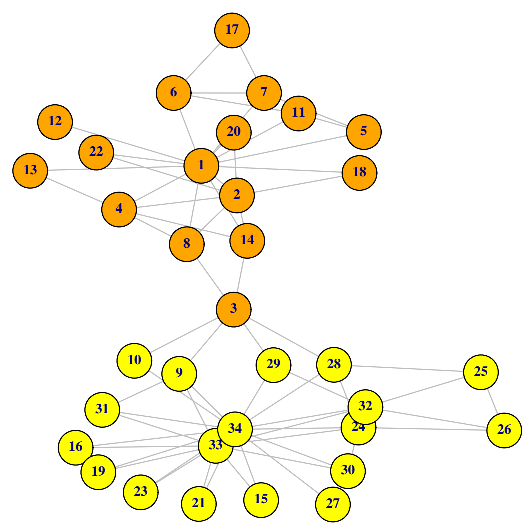


(b) After removing 2 − > 31

Figure 4: Iterations 6 and 7

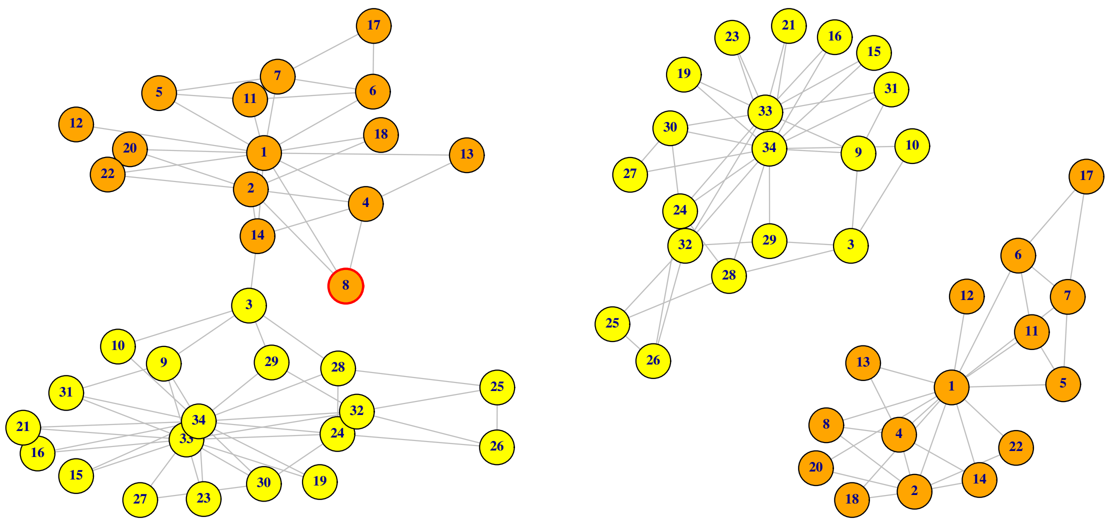


(a) After Removing 2 − > 3



(b) After removing 3 − > 4

Figure 5: Iterations 8 and 9



(a) After Removing 3 − > 8

(b) After removing 3 − > 14

Figure 6: Iterations 10 and 11

Based on the graph obtained from running the program it is clearly evident that a weighted graph of social interactions can be used to predict the result of a split. It should be noted that in Figure 5-b and 6-a, that node 3 has switched groups. This indicates that the node may have changed it's mind to which group it belongs to or that perhaps maybe there exists limitation to how well the model can predict a split.

References

[1] Mohammed Aturban. *Assigment 9*. 2013. URL: https://github.com/maturban/cs595-f13/blob/master/assignment9/latex/Mohamed_Aturban_Assignment_9.tex.

[2] Ricky Ho. *Basic graph analytics using igraph*. Apr. 2012. URL: http://horicky.blogspot.com/2012/04/basic-graph-analytics-using-igraph.html.

[3] Michael Nelson. *Social Networks*. URL: https://github.com/phonedude/cs532-s16/blob/master/slides/week-06-social-networks.ppt?raw=true.

[4] The igraph core team. *Connected components of a graph*. URL: http://igraph.org/r/doc/components.html.

[5] W. W. Zachary. *Zachary's Karate Club*. URL: https://networkdata.ics.uci.edu/data.php?id=105.

[6] Wayne W. Zachary. "An Information Flow Model for Conflict and Fission in Small Groups". In: *Journal of Anthropological Research* 33.4 (1977), pp. 452–473.