CSS692\_HW2

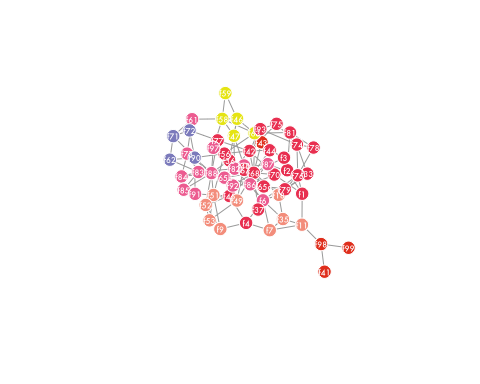
Greg Tozzi

2/13/2017

### Importing and Exploring the Graph

The igraph package in R had difficulty parsing the provided Pajek version of the Attiro graph. I wrote the helper function import\_attiro in R to assist in importing the file and reducing it to an undirected graph. A plot of the resulting graph follows. In this plot, and the plots that follow, node groupings map to vertex colors.

All the helper functions I wrote to assist in this assignment are available in the helper\_functions.R file on [my GitHub page](https://github.com/gregtozzi/CSS692_HW2) .



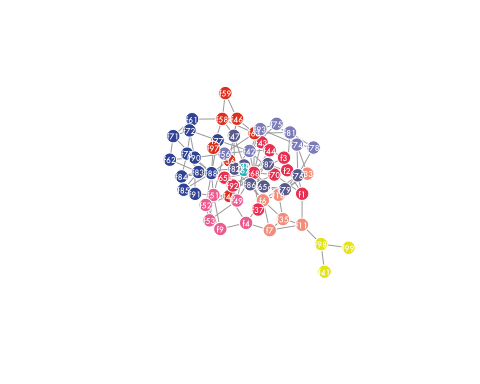
The modularity associated with the grouping assigned by the investigators can be computed by first treating the graph as undirected by applying the as.undirected function with mode = collapse, then applying the modularity function to the graph using the given groupings. The resulting modularity is 0.26 which is rather low.

#### Checking igraph's modularity calculations

The relatively low value of modularity for the graph using the investigators' partitioning made me question how igraph is computing modularity. As a check, I implemented a modularity computation based on the matrix formulation available on Wikipedia. As expected, it runs substantially slower than the igraph implementation, which is written in C. The values returned by my function and those returned by igraph's modularity are equal for very small graphs, but diverge slightly as graph size increases. Using the clustering provided by the investigators, my function yields a modularity of 0.39.

### Applying the Girvan–Newman Algorithm

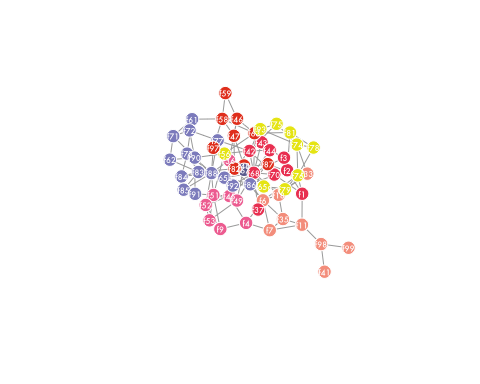
I applied the Girvan-Newman algorithm with a call to the igraph package's cluster\_edge\_betweenness function. Following Newman and Girvan (2003), I treated the graph as undirected by setting the parameter directed = FALSE. The resulting division of the graph into communities resulted in a modularity of 0.49. A plot of the graph showing the new community divisions follows.



Compared with the investigators' graph, the communities detected by the Girvan-Newman algoritm appear to be more consistent with the topology of the graph.

### Applying the Fast Greedy Algorithm

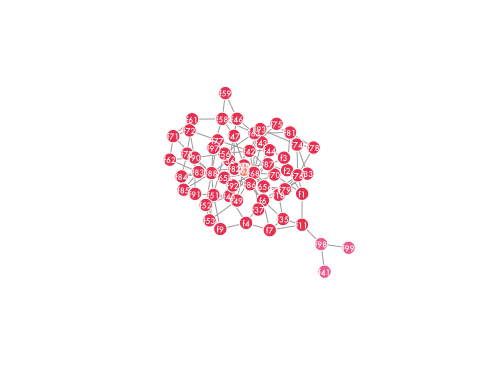
I applied the fast greedy algorithm described in Cluaset et al. (2004) with a call to the igraph package's cluster\_fast\_greedy function. The implantation in igraph requires that the graph first be converted to an undirected graph. I accomplished this through a call to as.undirected. The resulting division of the graph into communities produced a modularity of 0.49. A plot of the graph showing the new community divisions follows.



The fast greedy algorithm produced a bit more intuitive graph than did the Girvan–Newman algorithm.

### Applying the Propagating Labels Algorithm

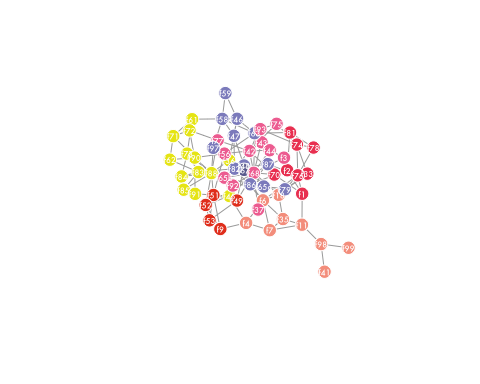
I applied the propagating labels algorithm described in Raghavan et al. (2007) with a call to the igraph package's cluster\_label\_prop function. The principal benefit of this algorithm is its speed, although that benefit is not evident when the algorithm is applied to this fairly modest graph. As with the fast greedy algorithm, using the propagating labels algorithm in igraph requires that the graph first be converted to an undirected graph. The resulting division of the graph into communities produced a modularity of 0.03. A plot of the graph showing the new community divisions follows.



The algorithm opens with random label assignment, so the resulting memberships and modularity vary from run to run. In general, the resulting modularities are comparable to those returned by applying the Girvan-Newman algorithm.

### Applying the Louvain Method in Gephi

I imported the Attiro network as an undirected graph in Gephi and applied the Louvain method to detect communities. Like the igraph package in R, Gephi attempted to treat the edge types as weights. In applying the algorithm, I neglected these weights. I exported the resulting groupings as a CSV file and imported them into R for analysis. The resulting division into groups is shown in the plot below.



The Louvain method produced a modularity of 0.5.

### Conclusions

The Louvain method in Gephi appears to provide the best modularity of the methods considered. Given that the groupings generated by the investigators produced such poor modularity scores, it is likely that they used additional information, including the types of relationships, to construct them. Absent their specific rationale, it is difficult to compare their results to those returned using the various methods considered here.