Practice4

Dasha Asienga

Due by midnight, Wednesday, Novemeber 2

Practicing Academic Integrity

If you worked with others or used resources outside of provided course material (anything besides our textbook(s), course materials in Moodle, R help menu) to complete this assignment, please acknowledge them below using a bulleted list.

I acknowledge the following individuals with whom I worked on this assignment:

Name(s) and corresponding problem(s)

-

I used the following sources to help complete this assignment:

Source(s) and corresponding problem(s)

-

Prompt

```
# Data is stored in a .rda file
# Open connection and grab the data
con <- url("https://awagaman.people.amherst.edu/stat240/studentnets.M182.rda")
print(load(con))
```

```
## [1] "social_df"            "task_df"              "friend_df"
## [4] "m182_full_data_frame"
```

```
close(con) # url() always opens the connection
```

This data set was originally in the NetData package which is no longer available for R (stopped being maintained). Professor Wagaman used the Github repo for the package and obtained the .rda with the data set to use it.

The data set is from this source: McFarland, Daniel A. (2001) "Student Resistance." American Journal of Sociology, 107(3), p 612-678. From the original package help file "This data consists of a sociometric friendship survey (2 = best friend, 1 = friend, 0 = not friend)." While there are three components to the data set, we will just look at the friendship network and the social network (not the task network), and looking at edges that had non-zero values. The social network edges are actually weighted differently from 0, 1, 2, as there are fractions, etc. but we're just going to say if there is a non-zero weight, we will keep an edge in the network, since there is an interaction between those students.

```
m182_full_nonzero_edges <- subset(m182_full_data_frame,
                          (friend_tie > 0 | social_tie > 0 | task_tie > 0))
head(m182_full_nonzero_edges)
```

```
##    ego alter friend_tie social_tie task_tie
## 5    1     5          0       1.20     0.30
## 8    1     8          0       0.15     0.00
## 9    1     9          0       2.85     0.30
## 10   1    10          0       6.45     0.30
## 11   1    11          0       0.30     0.00
## 12   1    12          0       1.95     0.15
```

```
m182_full <- graph.data.frame(m182_full_nonzero_edges)
summary(m182_full)
```

```
## IGRAPH 3366702 DN-- 16 144 --
## + attr: name (v/c), friend_tie (e/n), social_tie (e/n), task_tie (e/n)
```

```
friend <- delete.edges(m182_full,
          E(m182_full)[get.edge.attribute(m182_full,name = "friend_tie")==0])
friend <- as.undirected(friend, mode='collapse')
summary(friend)
```

```
## IGRAPH bd83689 UN-- 16 42 --
## + attr: name (v/c)
```

```
social <- delete.edges(m182_full,
          E(m182_full)[get.edge.attribute(m182_full,name = "social_tie")==0])
social <- as.undirected(social, mode='collapse')
summary(social)
```

```
## IGRAPH f236c56 UN-- 16 67 --
## + attr: name (v/c)
```

The code above basically takes the networks which originally had edges between all students and used edge weights to distinguish them, and now only keeps edges that had non-zero weights, to make them easier to work with. (You could investigate weighted graphs in future projects, if desired). In other words, originally, since there were edges between all students, you had to check the edge weight to see if there was really an interaction or stated friendship between the students.

To demonstrate our understanding of network analysis, we want to:

- 1) compare the two networks (broadly - you have many descriptive statistics you can examine),
- 2) determine which students are most important in both networks (and whether that agrees for both), and
- 3) determine what community structure is present in the networks, and whether or not it is the same in both networks.

You should envision that you are writing the report for the researcher of the original paper, who just isn't able to enact the analysis in R themselves. This means that you don't need to provide details about the data set creation (how data was collected), etc., but should describe the networks as they are implemented in R. For example, these have been implemented as undirected networks, where all edges kept had non-zero weights in the original dataset, and the researcher doesn't know how many vertices and edges your networks have now because there has been some pre-processing.

Introduction

Purpose of the Analysis

The purpose of this analysis is to conduct network analysis in R for a researcher working on student social data. The researcher is unable to enact the analysis in R. We will conduct network analysis on the friendship data and the social data in order to understand student interactions in the data collected. As part of the network analysis, I have 3 main tasks.

1) Compare the two networks broadly through descriptive statistics in order to have an understanding on the makeup and structure of the network.

2) Determine which students are most important in both networks and compare the results using different centrality measures.

3) Determine the community structure in both networks and compare any similarities or differences between the 2 networks.

Ultimately, the purpose of this analysis is to present a report to the researcher with friendship and social network analysis on the student data. We will use R packages to conduct the analysis.

The Data Set

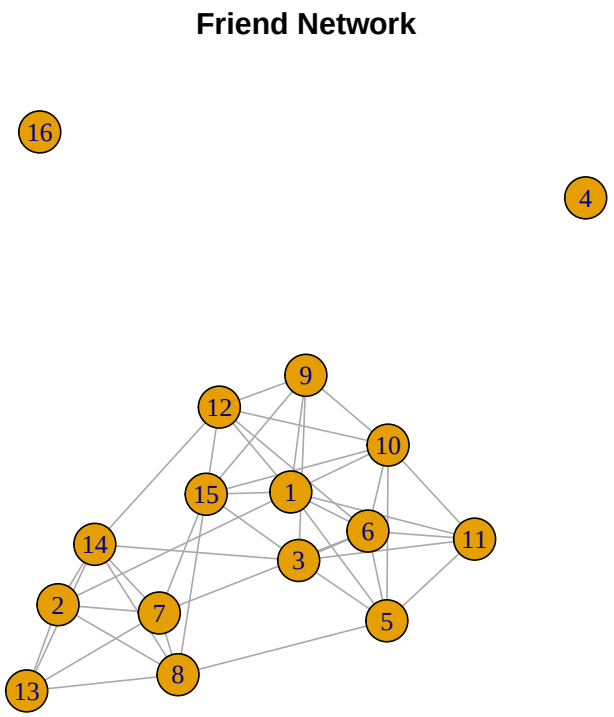
The data set is from McFarland, Daniel A. (2001) "Student Resistance." American Journal of Sociology, 107(3), p 612-678. While there are 3 components in the data set, we will focus on the friendship network and social network. The original data set had edges between all students and used edge weights to distinguish them. To make them easier to work with, the augmented data set now only keeps edges that had non-zero weights. We will also implement them as undirected networks. Because of the additional processing of the data, we will begin by describing the new data set through some preliminary analysis before beginning the network analysis.

Preliminary Analysis

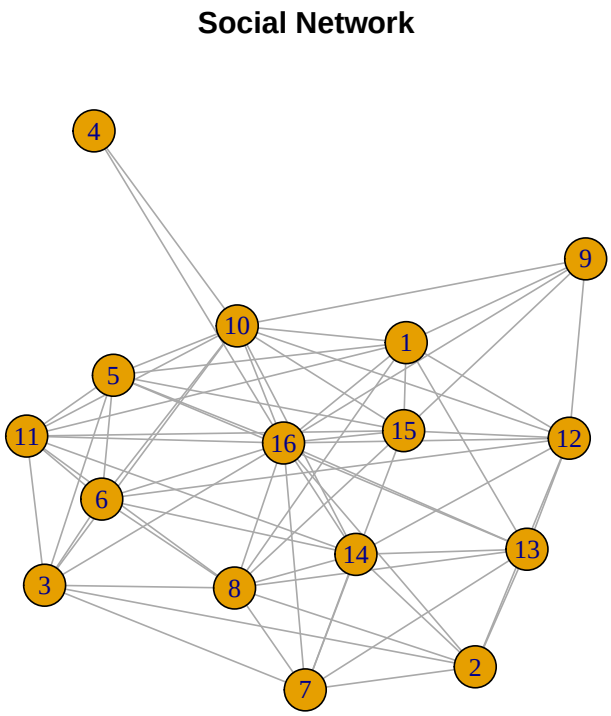
Visualizing the Networks

First, let's observe a visualization of the entire network. Both networks have the same number of vertices. However, the friend network appears to have less edges between the vertices than the social network. This network is also disconnected with 3 components. The largest component has 14 vertices and we will use that for some of the analysis that must be done on a connected component or graph. The social network, however, is connected and seems to have a higher transitivity (more connections between vertices). We will explore this by looking at the actual statistics. Finally, both networks are simple (no self-edges or multi-edges) and undirected.

```
set.seed(240)
plot(friend, main = "Friend Network")
```




```
plot(social, main = "Social Network")
```



Now that we have visualized the networks, let's examine some descriptive statistics on the networks.

Size and Order of the Networks

Both networks have an order of 16 (that is, there are 16 vertices). However, the friend network has a size of 42 (there are 42 edges) and the social network has a size of 67 (there are 67 edges). Indeed, the social network has more vertices as visualized above. Note that this is with the processed data.

```
vcount(friend)
```

```
## [1] 16
```

```
ecount(friend)
```

```
## [1] 42
```

```
vcount(social)
```

```
## [1] 16
```

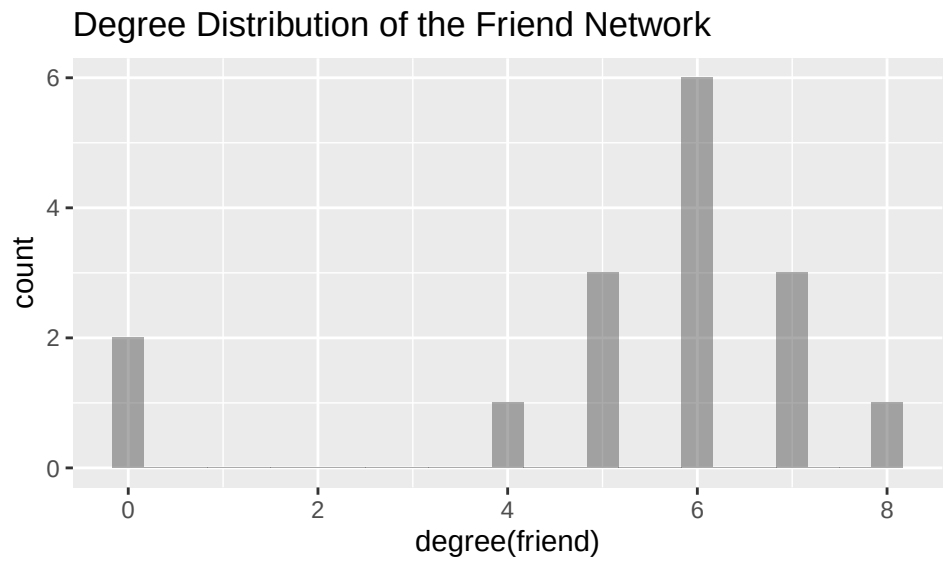
```
ecount(social)
```

```
## [1] 67
```

Degree Distributions of the Network

From the degree distribution, we notice that most vertices in the friend distribution have a degree of 6. All vertices, with the exception of the 2 that are disconnected from the rest of the graph (degree of 0), have a degree between 4 and 8. The degree distribution of the social network is scaled rightwards in comparison with the friend network. All vertices have a degree between 2 and 15, with most having a degree between 7 and 9. The vertex with the highest degree has a degree of 15, meaning that it has an edge to every vertex in the graph. This vertex is likely the most central with regard to degree centrality, but we will examine other centrality measures as well. There are no vertices with a degree of 0 in the social network.

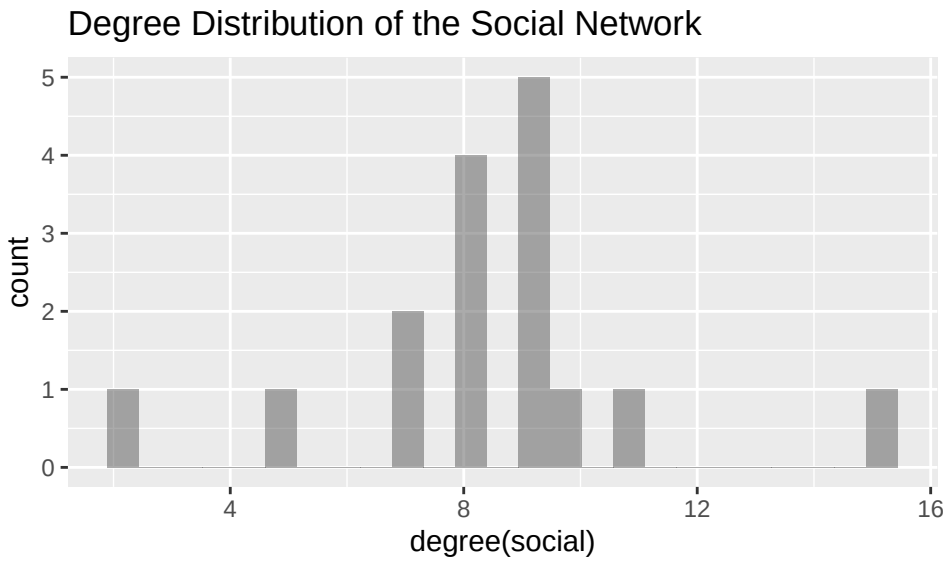
```
gf_histogram( ~ degree(friend), title = "Degree Distribution of the Friend Network")
```




```
tally(~ degree(friend))
```

```
## degree(friend)
## 0 4 5 6 7 8
## 2 1 3 6 3 1
```

```
gf_histogram( ~ degree(social), title = "Degree Distribution of the Social Network")
```




```
tally(~ degree(social))
```

```
## degree(social)
##  2  5  7  8  9 10 11 15
##  1  1  2  4  5  1  1  1
```

Now that we have explored the data and have a sense of both networks, let's move on to perform some more analysis on the networks.

Methods

Network Analysis

A network is a collection of interconnected things, that is, a collection of vertices and of edges that show relationships between these vertices. Network analysis, essentially, entails visualizing, characterizing, and modeling networks, through manipulating network data (relational data). There are many descriptive statistics and measures that we can use to describe and interpret networks. As seen in the exploratory analysis, we have already began to describe our 2 networks, and during the results section, we will use more statistics of comparison as well as look at some centrality measures and run community detection algorithms.

Statistics for Comparison

We will use 3 main statistics to compare our 2 networks: transitivity, average path length and diameter, and density.

Transitivity refers to how likely it is for 2 vertices connected to a third vertex to be connected to themselves. It measures how tightly clustered the graph is through clustering coefficients.

Average path length (APL) describes the average/ typical shortest path between any 2 vertices. Diameter, on the hand, describes the longest of such shortest paths between any 2 vertices in the network. This allows us to assess how well-connected a network graph is and how easy it is for any 2 vertices to communicate and pass information.

Finally, density is measured by dividing the number of realized edges by the number of potential edges. This also allows us to describe how connected a graph is.

Centrality Measures

We are also interested in identifying which vertices are most central in our 2 networks. We will use 3 different centrality measures and compare whether the results differ or agree.

Closeness centrality (CC) computes how close a vertex is to other vertices in the network using a distance measure as a measure of importance of the vertex.

Betweenness centrality (BC) measures how important a vertex is based on the extent to which it falls between other vertices (on many shortest paths).

Eigenvector centrality (EC) computes the relative importance of a vertex based on status and prestige, that is, the more central the neighbors of a vertex are, the more central that vertex is.

All measures are scaled for 0 to 1, with higher values denoting more centrality/ importance of a given vertex.

Community Detection Algorithms

Finally, we will run 2 community detection algorithms to detect clusters in our networks. The 2 main classes of algorithms I am familiar with are hierarchical methods and spectral partitioning methods, the latter being very similar to k-means clustering.

Hence, we will run both a hierarchical algorithm and a spectral partitioning algorithm on both our networks and use modularity, a statistic, to assess the strength and validity of our clustering solutions. This measure works by considering random graphs with the same degree distribution (in the exploratory analysis) as the network in consideration, but with edges placed at random. High values of modularity mean that the communities (clusters) in the network capture non-trivial group structure.

In particular, we will use the fastgreedy.community R function to run the hierarchical algorithm and the leadingeigenvector.community R function to run the spectral partitioning algorithm.

Results

Transitivity

As suspected from the preliminary analysis, the social network graph has higher transitivity than the friend network graph. Transitivity is a measure of clustering, hence, this is expected because the social graph is more connected, and therefore, more clustered.

```
transitivity(friend)
```

```
## [1] 0.511521
```

```
transitivity(social)
```

```
## [1] 0.621818
```

The local measures of transitivity agree with the global measures of transitivity. Local measures tell us how likely each vertex is to be in a triple or triangle. From the output below, we can see that the vertices in the friend network have lower transitivities than those in the social network. The smallest local coefficient in the social network is 0.5, while the smallest coefficient in the friend network is 0.27. This confirms to us that the social network is more connected and clustered than the friend network.

```
sort(transitivity(friend, "local"), method = "shell", decreasing = TRUE)
```

```
##  [1] 1.000000 0.800000 0.700000 0.619048 0.600000 0.533333 0.533333 0.476190
##  [9] 0.466667 0.466667 0.464286 0.400000 0.380952 0.266667
```

```
sort(transitivity(social, "local"), method = "shell", decreasing = TRUE)
```

```
##  [1] 1.000000 1.000000 0.761905 0.722222 0.714286 0.714286 0.678571 0.666667
##  [9] 0.642857 0.642857 0.638889 0.611111 0.600000 0.583333 0.545455 0.495238
```

Average Path Length and Diameter

The friend graph has an average path length (APL) that is slightly longer than that for the social network. This tells us how long a typical path is between any 2 vertices. The APL for the friend network is 1.57 and the APL for the social network is 1.44.

The diameter of the friend network is also slightly longer than that of the social network, with the former being 3 and the latter being 2. The diameter gives us the longest shortest path length, that is, the largest shortest distance between any 2 vertices in the network.

Because the friend network is disconnected, these values were computed on the largest connected component (LCC). Overall, there is not much difference between the 2, but as expected, the paths are slightly longer for the friend network than for the social network.

```
average.path.length(friend)
```

```
## [1] 1.57143
```

```
average.path.length(social)
```

```
## [1] 1.44167
```

```
diameter(friend)
```

```
## [1] 3
```

```
diameter(social)
```

```
## [1] 2
```

Density

Finally, let's examine the density of both networks. Density is measured by dividing the number of realized edges by the number of potential edges. A complete graph has a density of 1 because every possible edge is realized.

As expected, the friend network is sparser than the social network, with a density of 0.35 as compared to 0.56. Therefore, only 35% of all possible edges in the friend network are present while 56% of all possible edges in the social network are present.

```
graph.density(friend)
```

```
## [1] 0.35
```

```
graph.density(social)
```

```
## [1] 0.558333
```

This completes task 1 which asks us to compare the 2 networks using various statistics of comparisons.

Now, let's move on to task 2 and begin to examine the centrality measures. Based on degree centrality in the preliminary analysis, vertex 16 was the most important vertex in the social network and vertex 1 was the most important in the friend network. Note that these measures will be computed on the largest connected component of the friend network.

Closeness Centrality

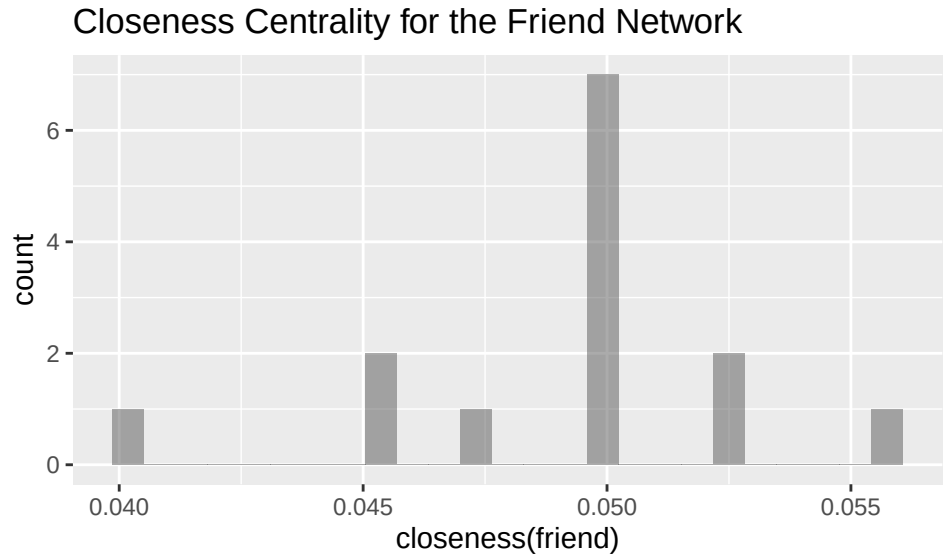
Closeness centrality (CC) computes how close a vertex is to other vertices in the network using a distance measure as a measure of importance of the vertex.

The friend network has CC values between 0.04 and 0.0556. It's not too skewed. However, the social network has CC values between 0.036 and 0.067. There is a single outlier with the highest centrality measure of 0.067, while most of the other vertices have a CC value between 0.036 and 0.05. Interestingly, the individual vertices in the social network seem to have lower centrality score than those in the largest connected component of the friend graph.

Indeed, vertex 16 is the most important vertex in the social network and vertex 1 is the most important vertex in the friend network according to this measure.

```
gf_histogram(~ closeness(friend), title = "Closeness Centrality for the Friend Network")
```

```
## Warning: Removed 2 rows containing non-finite values (stat_bin).
```



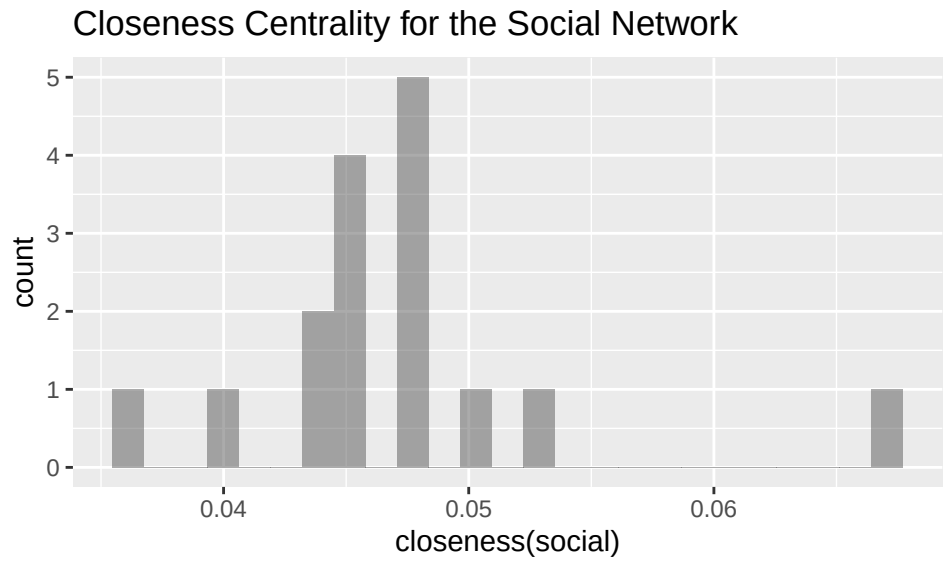

```
favstats(~ closeness(friend))
```

```
##   min        Q1 median   Q3       max      mean         sd  n missing
##  0.04 0.0482143   0.05 0.05 0.0555556 0.0492391 0.00375806 14       2
```

```
sort(closeness(friend), method = "shell", index.return = TRUE, decreasing = TRUE)
```

```
## $x
##         1         6        15         3         5         7         8        10
## 0.0555556 0.0526316 0.0526316 0.0500000 0.0500000 0.0500000 0.0500000 0.0500000
##        12        14         2         9        11        13
## 0.0500000 0.0500000 0.0476190 0.0454545 0.0454545 0.0400000
##
## $ix
##  [1]  1  5 14  3  4  6  7  9 11 13  2  8 10 12
```

```
gf_histogram(~ closeness(social), title = "Closeness Centrality for the Social Network")
```




```
favstats(~ closeness(social))
```

```
##        min        Q1    median       Q3       max      mean         sd  n
##  0.0357143 0.0449605 0.0465368 0.047619 0.0666667 0.0469927 0.00653266 16
##  missing
##        0
```

```
sort(closeness(social), method = "shell", index.return = TRUE, decreasing = TRUE)
```

```
## $x
##        16        10         8         1        11        12        14        15
## 0.0666667 0.0526316 0.0500000 0.0476190 0.0476190 0.0476190 0.0476190 0.0476190
##         3         5         6        13         2         7         9         4
## 0.0454545 0.0454545 0.0454545 0.0454545 0.0434783 0.0434783 0.0400000 0.0357143
##
## $ix
##  [1] 16 10  8  1 11 12 14 15  3  5  6 13  2  7  9  4
```

Now, let's examine another measure of centrality, betweenness centrality.

Beetweenness Centrality

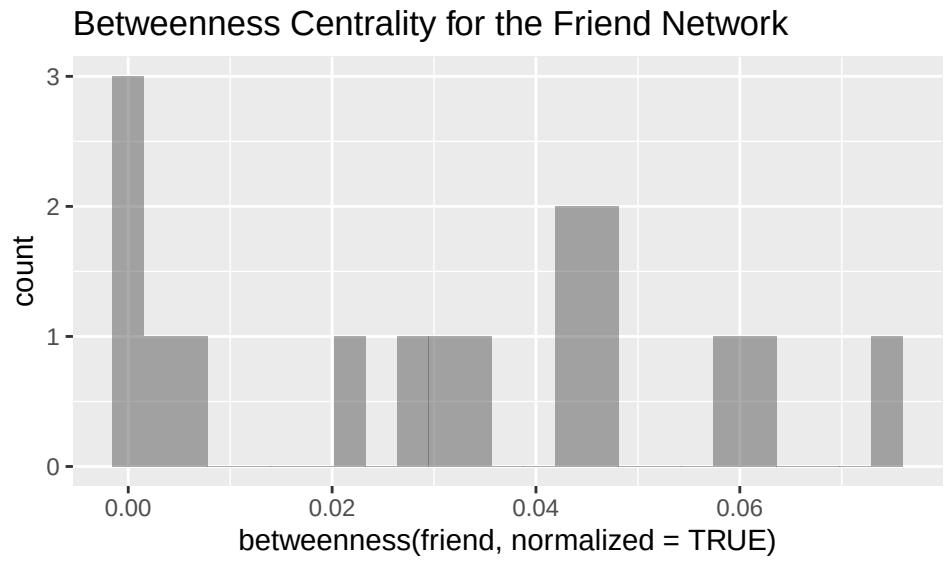
Betweenness centrality (BC) measures how important a vertex is based on the extent to which it falls between other vertices (on many shortest paths).

In the friend network, we observe that the vertices have a BC value between 0 and 0.074. In fact, it seems most have a value close to 0, which makes sense because there aren't many short paths between the vertices because the graph is less connected. The social network, on the other hand, has BC values between 0 and 0.17. It's right skewed because there are 2 vertices with higher values and most of the other vertices have values between 0 and 0.05.

The friend network has such a distribution because it likely has some vertices that are very close to each other and some that are very far from each other.

Vertex 16 is still the most important in the social network and vertex 1 is still the most important in the social network.

```
gf_histogram(~ betweenness(friend, normalized = TRUE), title = "Betweenness Centrality for the Friend N
```



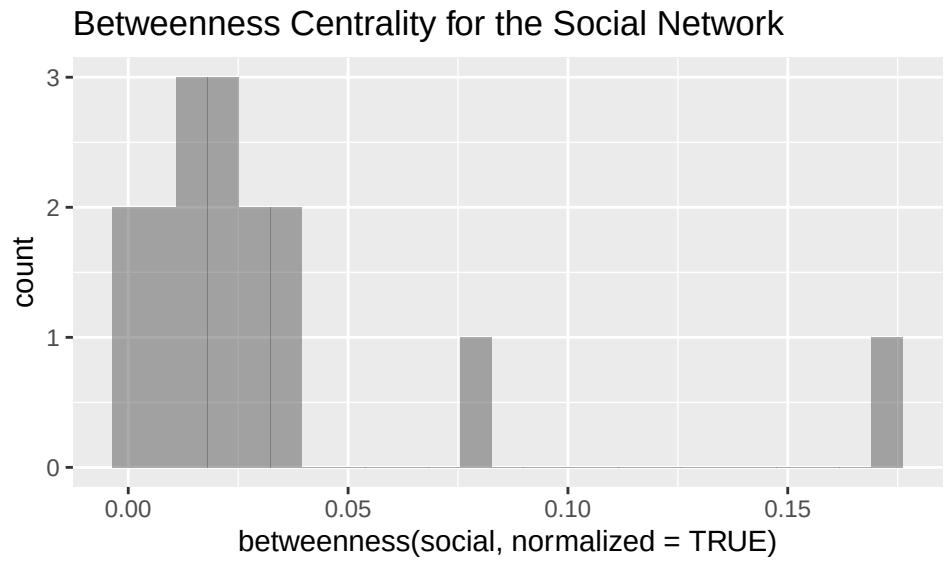

```
favstats(~ betweenness(friend, normalized = TRUE))
```

```
##  min         Q1    median        Q3       max      mean        sd  n missing
##    0 0.00559524 0.0327778 0.0450794 0.0744444 0.0309524 0.0239569 16       0
```

```
sort(betweenness(friend, normalized = TRUE), method = "shell", index.return = TRUE, decreasing = TRUE)
```

```
## $x
##          1         15         14          3          6          8          7
## 0.07444444 0.06238095 0.05777778 0.04507937 0.04507937 0.04396825 0.04190476
##          5         12          2         10          9         11          4
## 0.03317460 0.03238095 0.02650794 0.02253968 0.00619048 0.00380952 0.00000000
##         13         16
## 0.00000000 0.00000000
##
## $ix
##  [1]  1 15 14  3  6  8  7  5 12  2 10  9 11  4 13 16
```

```
gf_histogram(~ betweenness(social, normalized = TRUE), title = "Betweenness Centrality for the Social N
```




```
favstats(~ betweenness(social, normalized = TRUE))
```

```
##  min        Q1    median        Q3      max      mean        sd  n missing
##    0 0.0122789 0.0208163 0.0317347 0.172585 0.0315476 0.0417414 16       0
```

```
sort(betweenness(social, normalized = TRUE), method = "shell", index.return = TRUE, decreasing = TRUE)
```

```
## $x
##         16         10         12          8         15         14          1
## 0.17258503 0.07739229 0.03507937 0.03421769 0.03090703 0.02526077 0.02376417
##          3         13          5         11          6          7          2
## 0.02160998 0.02002268 0.01587302 0.01582766 0.01290249 0.01040816 0.00891156
##          4          9
## 0.00000000 0.00000000
##
## $ix
##  [1] 16 10 12  8 15 14  1  3 13  5 11  6  7  2  4  9
```

EigenVector Centrality

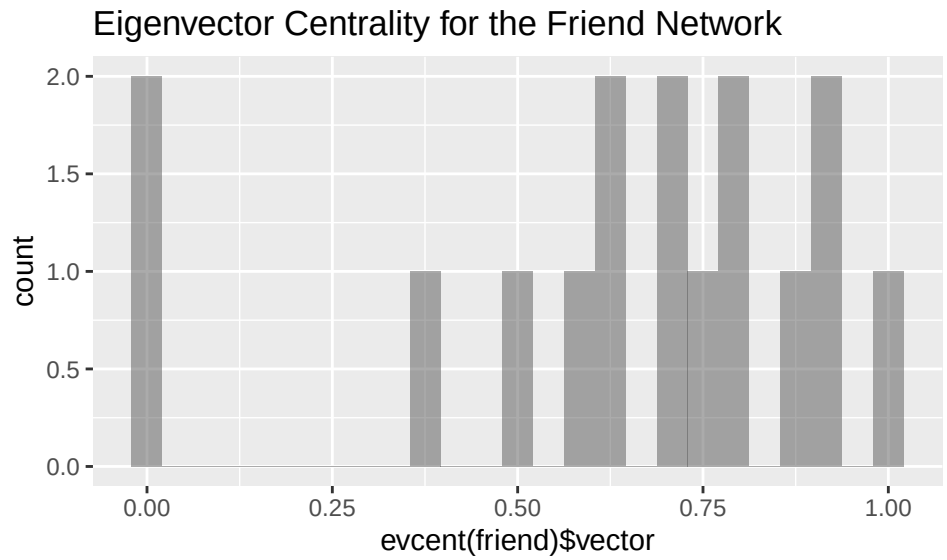
Lastly, let's examine the eigenvector centrality (EC) of the networks. Eigenvector centrality computes the relative importance of a vertex based on status and prestige, that is, the more central the neighbors of a vertex are, the more central that vertex is.

For the friend network, the vertices have EC values ranging for 0 to 1. Most lie between 0.57 and 0.82. For the social network, the vertices have EC values ranging from 0.2 to 1.

Because vertices are more closely connected and likely neighbors in the social network, we would expect the values to be higher for the network graph. We would expect a huge range for the friend graph because some vertices may be close to other important vertices and some may be very far.

Vertex 16 is still the most important vertex in the social network and vertex 1 is still the most important vertex in the friend graph.

```
gf_histogram(~ evcent(friend)$vector, title = "Eigenvector Centrality for the Friend Network")
```



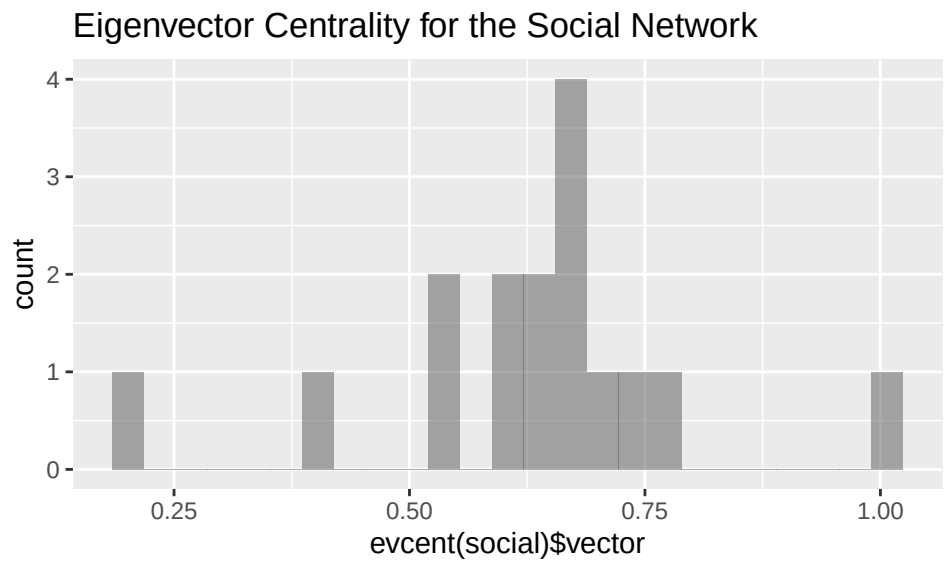

```
favstats(~ evcent(friend)$vector)
```

```
##          min       Q1   median       Q3 max     mean      sd  n missing
##  4.74894e-18 0.570901 0.699896 0.820425   1 0.632893 0.29482 16       0
```

```
sort(evcent(friend)$vector, method = "shell", index.return = TRUE, decreasing = TRUE)
```

```
## $x
##           1          10           6          15          12           5
## 1.00000e+00 9.28476e-01 8.98704e-01 8.69585e-01 8.04038e-01 7.84842e-01
##           3          11           9           7           8          14
## 7.32602e-01 7.00696e-01 6.99096e-01 6.21704e-01 6.05891e-01 5.89533e-01
##           2          13           4          16
## 5.15004e-01 3.76124e-01 2.36601e-17 2.36601e-17
##
## $ix
##  [1]  1 10  6 15 12  5  3 11  9  7  8 14  2 13  4 16
```

```
gf_histogram(~ evcent(social)$vector, title = "Eigenvector Centrality for the Social Network")
```




```
favstats(~ evcent(social)$vector)
```

```
##       min      Q1   median       Q3 max     mean       sd  n missing
##  0.194351 0.58488 0.649754 0.689051   1 0.628837 0.170249 16       0
```

```
sort(evcent(social)$vector, method = "shell", index.return = TRUE, decreasing = TRUE)
```

```
## $x
##       16       10        8       11        1       14       15       12
## 1.000000 0.761089 0.735564 0.707693 0.682837 0.681140 0.676084 0.661166
##        6        5        3       13        7        2        9        4
## 0.638343 0.626707 0.610915 0.601868 0.533914 0.532431 0.417284 0.194351
##
## $ix
##  [1] 16 10  8 11  1 14 15 12  6  5  3 13  7  2  9  4
```

All the three centrality measures and degree centrality (from the preliminary analysis) agree that vertex 16 is the most important vertex in the social network and vertex 1 is the most important in the friend network. Vertex 10 appears as the second most important vertex in both networks for most of the centrality measures.

Now that we have analyzed our networks' centrality, let's conclude the analysis through community detection.

Hiearchical Clustering

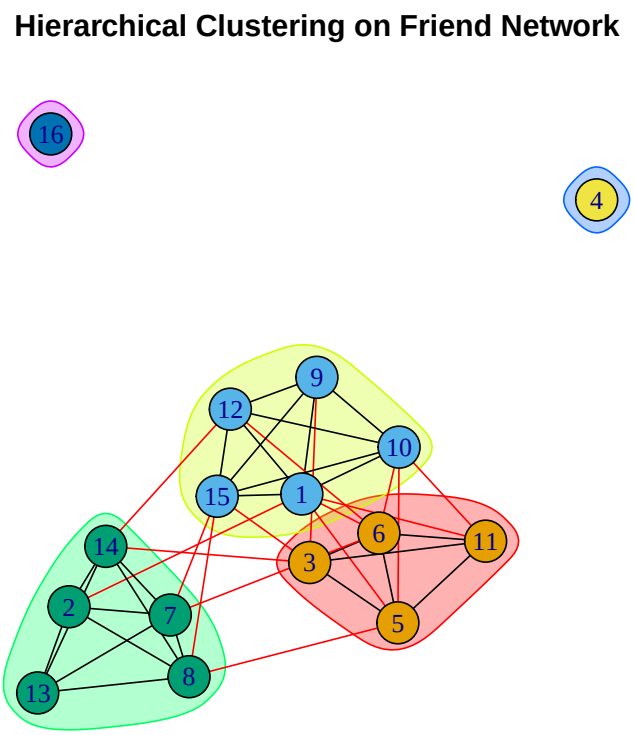
```
set.seed(240)
fgfriend <- fastgreedy.community(friend)
length(fgfriend)
```

```
## [1] 5
```

```
sizes(fgfriend)
```

```
## Community sizes
## 1 2 3 4 5
## 4 5 5 1 1
```

```
plot(fgfriend, friend, main = "Hierarchical Clustering on Friend Network")
```



Our hierarchical clustering solution found 5 clusters on the friend network. 2 of the clusters contain the 2 disconnected vertices. The remaining 3 clusters appear to have some visible difference and there is not much overlap. They are also of similar size with 4, 5, and 5 vertices in the clusters respectively.

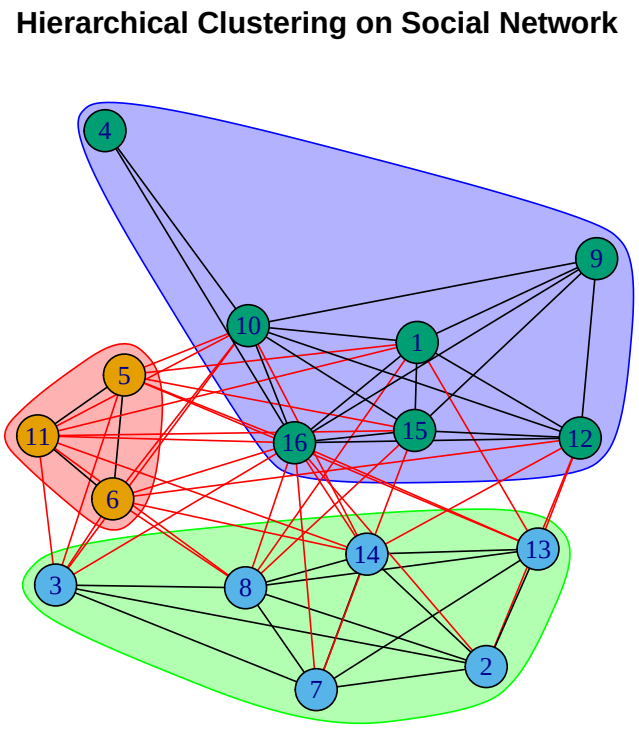
```
fgsocial <- fastgreedy.community(social)
length(fgsocial)
```

```
## [1] 3
```

```
sizes(fgsocial)
```

```
## Community sizes
## 1 2 3
## 3 6 7
```

```
plot(fgsocial, social, main = "Hierarchical Clustering on Social Network")
```



On the social network, our hierarchical clustering solution found 3 clusters. There also appears to be no overlap between these clusters, although they are all of different sizes. Cluster 1 has 3 vertices, cluster 2 has 6 vertices, and cluster 3 has 7 vertices.

Spectral Clustering

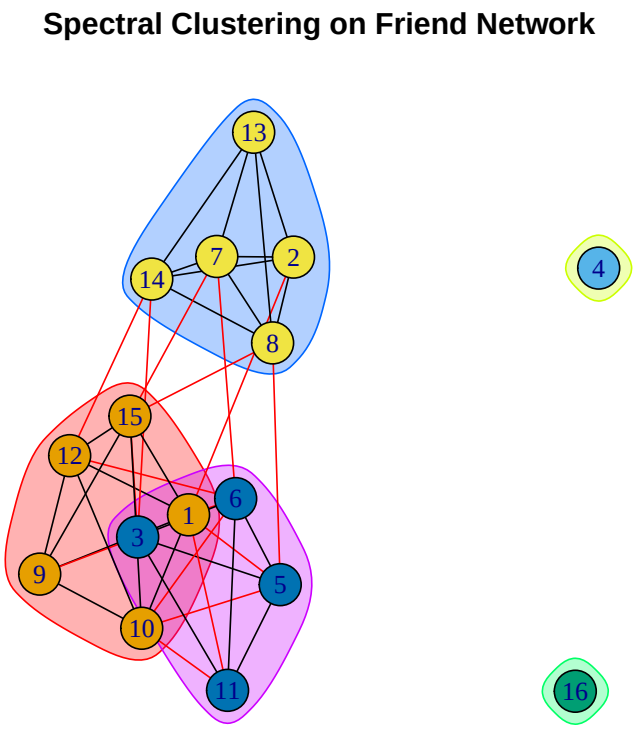
```
spfriend <- leading.eigenvector.community(friend)
length(spfriend)
```

```
## [1] 5
```

```
sizes(spfriend)
```

```
## Community sizes
## 1 2 3 4 5
## 5 1 1 5 4
```

```
plot(spfriend, friend, main = "Spectral Clustering on Friend Network")
```



The spectral clustering also found 5 clusters on the friend network. Similarly, 2 of the clusters only contain each of the 2 disconnected vertices. However, there appears to be some overlap between the other 3 clusters. They each have 4, 5, and 5 vertices respectively. This is the same as the hierarchical solution.

```
spsocial <- leading.eigenvector.community(social)
length(spsocial)
```

```
## [1] 3
```

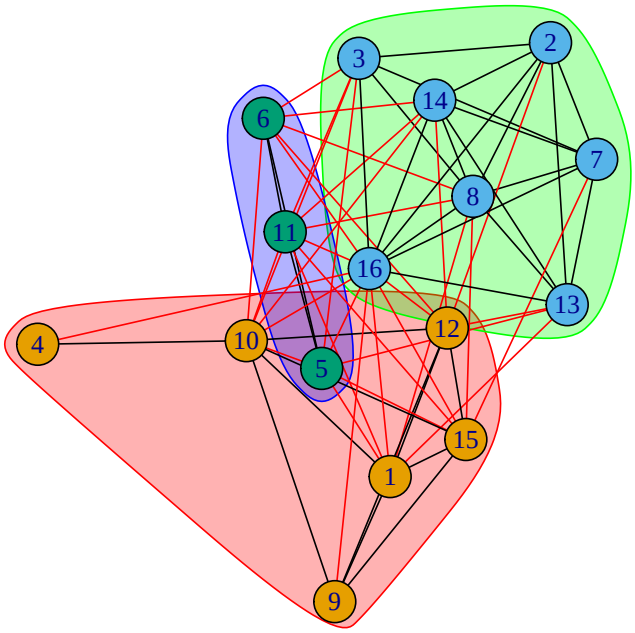
```
sizes(spsocial)
```

```
## Community sizes
## 1 2 3
## 6 7 3
```

```
membership(spsocial)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
##  1  2  2  1  3  3  2  2  1  1  3  1  2  2  1  2
```

```
plot(spsocial, social)
```



Lastly, the spectral clustering solution also found 3 clusters on the social network and there appears to be some overlap as well. One cluster has 3 vertices and the other 2 have 6 and 7 vertices respectively. This is the same as the hierarchical solution.

Modularity

Modularity is a statistic that can be used to assess clustering solutions. Interestingly, the friend network had higher modularity, even though it had lower transitivity. This may be because it was easier to detect differences between different clusters in the friend network than in the social network which was more connected and clustered as a whole.

Nevertheless, both the hierarchical and spectral clustering solutions found very similar results. There is no difference between the 2 solutions for the friend network, and the hierarchical solution is only slightly better than the spectral solution in the social network.

```
modularity(friend, fgfriend$membership)
```

```
## [1] 0.279762
```

```
modularity(friend, spfriend$membership)
```

```
## [1] 0.279762
```

```
modularity(social, fgsocial$membership)
```

```
## [1] 0.123524
```

```
modularity(social, spsocial$membership)
```

```
## [1] 0.116841
```

Overall, despite having a different number of clusters, the communities seem to be very similar across all solutions with some slight differences. For example, student 5, 11, and 6 as well as student 14, 7, 8 and 2 appear in the same cluster in all the 4 solutions. This suggests that these social connections are also friendship connections.

Conclusion

The purpose of this analysis was to conduct network analysis in R for a researcher working on student social data. We had 3 main tasks: 1) compare the two networks broadly through descriptive statistics in order to have an understanding on the makeup and structure of the network, 2) determine which students are most important in both networks and compare the results using different centrality measures, and 3) determine the community structure in both networks and compare any similarities or differences between the 2 networks.

Our network analysis concluded that the social network was more clustered (had higher transitivity), had more edge connections, had a shorter average path length, had a shorter diameter, and had a higher density than the friend network, which was a disconnected graph. This makes sense because social interactions are more easily facilitated and connected than friendships which require a lot more time, effort, and commitment. Social interactions don't always equate to friendships.

Vertex 16 was the most important vertex in the social network and vertex 1 was the most important vertex in the friend network as concluded by the different centrality measures we used. However, vertex 4 and 16 do not have any friend connections. This suggests that student 16 is very social and interacts with a lot of the students in the class, yet those interactions don't result in friendships and the student does not consider anybody a friend in the class. Student 4 is more socially isolated with no friends and only 2 social connections in the class.

Finally, our community detection algorithms found 5 communities (clusters) on the friend network and 3 communities (clusters) on the social network.