R Tutorial – Social Network Analysis

This tutorial will be investigating the network association between individuals within a group of big horn sheep population. The sample data that I will be using was partially collected by some of my lab-mates in 2012, but also partially changed so that I could look at the individual interactions instead of group data. Social network analysis studies the graphical representation of relationships between two or more discrete members/objects. It provides us with information of linkages between the members within a group which can be used in multiple ways. For example, the information can be used to determine patterns of disease transmission and prevalence in a population, which could inform us of key individuals and interactions that are responsible for majority of the spread of the infection. In this R project, I will be looking at the relationship and connectivity between the group members and making a network graph to better visualize the group and sub-group connectivity. For this, the first step is to install the package 'igraph' and run it. There are other network packages, for example, sna, asnipe and statnet to name a few, however, I chose to work with the igraph package as it is versatile and easy to calculate most network related analysis, faster than some of the other packages.

```
> install.packages("igraph")
> library (igraph)
```

After identifying the appropriate working directory and uploading the data file (, I check the structure of the datasheet as well as the first few values to understand the layout of the data.

```
> head(my_network)
  Observation       Date Group Location_abv  ID ID_interaction Weekly_census Sex Age
1           1 26/04/2012     1           hf 553            345            No   F   3
2           2 26/04/2012     1           hf 575            343            No   F   4
3           3 26/04/2012     1           hf 345            575            No   F   3
4           4 26/04/2012     1           hf 343            345            No   F   1
5           9 26/04/2012     2           hf 575            343            No   F   4
6          10 26/04/2012     2           hf 345            553            No   F   3
```

For the purposes of this project, the main information that I am interested in are the columns: ID, ID_interaction and Sex. However, to make an adjacency matrix showing the linkages between each members in the group, I am only going to be using the columns, ID and ID_interaction. I use the following code to create the adjacency matrix showing the interaction between each of the individuals:

```
> ad_mat <- matrix (length(unique(my_network$ID)),length(unique(my_network$ID_interaction)))
> ad_mat <- crossprod(table(my_network$ID,my_network$ID_interaction))
> diag (ad_mat) <- 0
> ad_mat
```

The first line of the code creates a matrix that has a length and width spanning the number of unique values that are present in the columns ID and ID_interaction within the my_network data frame. The second line of the code fills the matrix with the number of interactions between each of the individuals in the data-frame. For example, the value '0' is given if there is no interaction between the two individuals in question, the value '1' is given if there was only one interaction between the individuals and so on and

so forth such that, as you can see in the example table below, that individual 328 had one interaction with individual 282, three interactions with individual 314 and no interactions with individual 319. The crossproduct function is very useful in counting the total number of values between two categories. In the third line of the code, the matrix is filled with the value '0' diagonally to ensure that the individual is not counted as having interacted with itself. A sample view of the table is provided below:

	282	314	319	328	332	334	....
282	0	1	0	1	0	0	
314	1	0	0	3	1	2	
319	0	0	0	0	0	0	
328	1	3	0	0	1	2	
332	0	1	0	1	0	0	
334	0	2	0	2	0	0	
⋮							

The three digit numbers are the vertices (ID of the sheep interacting with each other), while the single digit number are the number of interactions between the individuals. While I am focusing on a subset of data regarding individual interactions for the purpose of this project, an alternative way of separating the information into relevant categories is by forming a period-based adjacency matrix and looking at the interaction between groups. That is to say, separating the interactions between individuals per day. This can be done by using a similar code as above, but also using the 'for' looping such that the adjacency matrix is developed for each day such that you get an array or a list of matrices that show you how many individuals from group 1 are also present in group 2, for example.

The matrix that you would get for the first day is shown below. The [,,1] shows that it's the first day, while the columns [,1] and the rows [1,] show the group number, while the values within the matrix (0 and 4) show the number of individuals that are present in group 1 and group 2.

```
> Interaction_array [,,1]
     [,1] [,2] [,3] [,4]
[1,]    0    4    0    0
[2,]    4    0    0    0
[3,]    0    0    0    0
[4,]    0    0    0    0
```

However, I will only be using the individual interaction data. If you want to convert the existing matrix into a binary matrix, you can use the following code to ensure that all numbers greater than 1, are converted into 1, while 0 remains constant. This can be useful if you are interested in simply knowing whether there is an interaction between the two individuals and not necessarily the number of times that the individuals interacted with each other. It can also be useful to compare the degree of interaction and the binary degree of interaction to, for example, test whether there are connectivity implications regarding the transmission of directly transmitted diseases or not. Following that, an adjacency graph is plotted.

```
> ad_mat [ad_mat > 0] <- 1
> trial_ag <- graph.adjacency(ad_mat, mode = "undirected", weighted = TRUE)
```

The graphical representation of this adjacency graph is shown in figure 1.

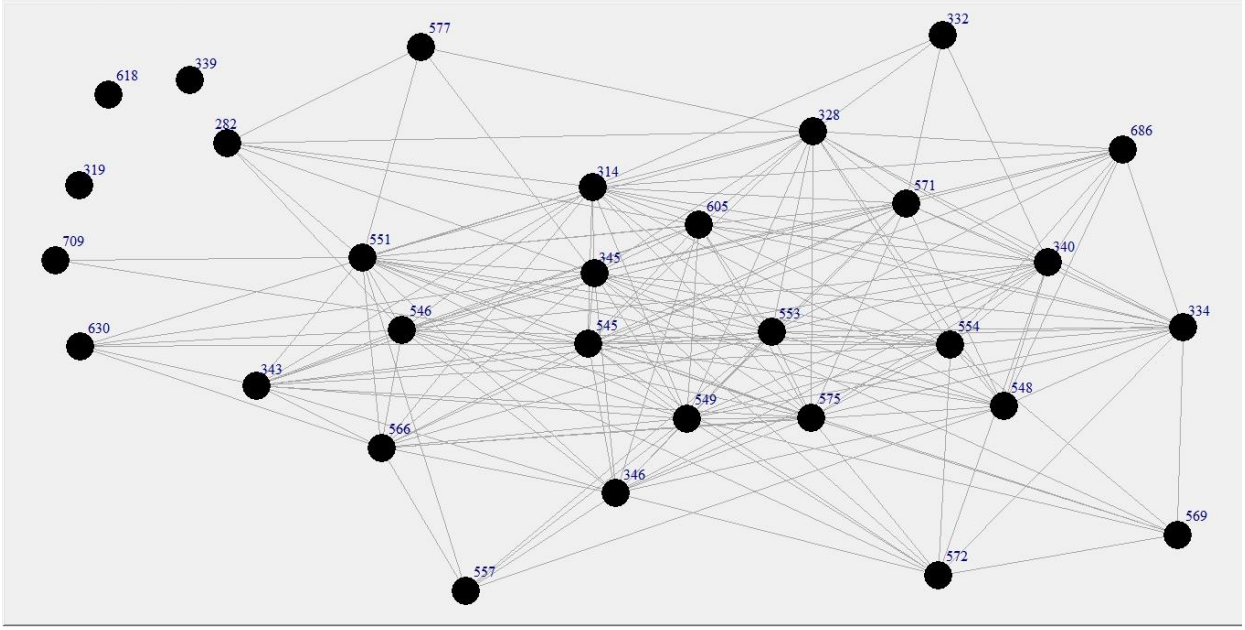

Figure 1. tkplot of the adjacency matrix showing the interaction between group members in a population of big horn sheep (`>tkplot(trial_ag, vertex.label.dist=1)`*)*

This graphical representation makes it clear that there are three individuals within the group that are not connected to any of the other members of the group (IDs 319, 618 and 339), while there are other members that could play a vital role in transmitting information or diseases along the rest of the group as they are connected to many individuals. We can also get the structure of this graph by running the name of the graph.

```
> trial_ag
IGRAPH UNW- 30 179 --
+ attr: name (v/c), weight (e/n)
+ edges (vertex names):
 [1] 282--314 282--328 282--345 282--546 282--551 282--577 282--605 314--328 314--332 314--334
[11] 314--340 314--343 314--345 314--545 314--546 314--548 314--549 314--551 314--553 314--566
[21] 314--571 314--575 314--605 314--630 314--686 328--332 328--334 328--340 328--345 328--545
+ ... omitted several edges
```

From the above, we can get some information regarding the number of vertices (30) as well as the number of interactions/edges (179). Based on the adjacency matrix and the graph, a number of measures can be calculated. The degree distribution depicts the number of connections or adjacent edges each vertex has. In this case, there are a range of values from 0.00 (the three individuals with no connections) to 0.13 (the individual with the highest number of connections). The function 'degree' also shows the same information, however, while the 'degree distribution' values are probability percentages of connections, the 'degree' function shows the actual number of connections or edges that each individual vertex has. For example, while the degree distribution code shows the following result,

```
> degree.distribution(trial_ag, loops = TRUE, normalize = FALSE)
[1]  0.10000000 0.00000000 0.03333333 0.00000000 0.06666667 0.00000000 0.06666667 0.06666667
[9]  0.00000000 0.00000000 0.06666667 0.00000000 0.00000000 0.06666667 0.03333333 0.13333333
 ⋮
```

the degree function shows the following results:

```
 > degree(trial_ag)
282 314 319 328 332 334 339 340 343 345 346 545 546 548 549 551 553 554 557 566 569 571 572 575
  7  19   0  17   4  15   0  16  13  21  15  22  16  13  19  19  18  17   6  14   7  15  10  18
577 605 618 630 686 709
  4  15   0   6  10   2
```

Similarly, the function 'edge connectivity' also shows that the group is not entirely connected because three individuals are completely isolated from the rest of the group, thus, giving a value of '0' since there is at least one individual (three in this case) that is disconnected from the rest of the group. 'Edge betweenness' is another function that can also be compared using the graph to find out the shortest paths that goes through each of the vertices. It demonstrates the connectivity of members within the group/population – potentially, the shortest pathway of communication that each individual has, which could also have implications on disease transmission.

```
> edge.betweenness(trial_ag, e = E(trial_ag), directed = FALSE, weights = NULL)
  [1]  7.7497835  4.1064214  0.8762626  0.5909091  6.9466089  3.0222222  7.2284271  0.0000000
  [9]  8.2283550  0.7857143  7.5643939  9.6326840  0.4444444  0.0000000  0.0000000  0.0000000
```

Similarly, the 'eigen centrality' function provides us with the eigenvector centralities of each vertex within a graph. For example, in this case, the eigenvector centrality has been calculated for each of the vertex in the graph and the values have been scaled such that the maximum score possible is 1 (ID 554). It is used to calculate the relative importance of each of the nodes in the graph.

```
> eigen_centrality(trial_ag, scale = TRUE, weights = NULL)
$vector
        282         314         319         328         332         334         339         340
0.167710787 0.690064684 0.005347296 0.596427645 0.088593975 0.712675179 0.005347296 0.551108165
        343         345         346         545         546         548         549         551
0.589751511 0.801679742 0.433393620 0.779922119 0.619173966 0.606726608 0.812058627 0.583037101
        553         554         557         566         569         571         572         575
0.698792246 1.000000000 0.131388495 0.380360583 0.165471114 0.470604048 0.264432179 0.819361481
        577         605         618         630         686         709
0.094068770 0.455071304 0.005347296 0.123094687 0.441027109 0.043453918

$value
[1] 31.36411
  ⋮
```

So far, while the graph and the connectivity manipulations show the connectedness of the group as a whole, it does not tell us whether there is a difference in the degree of connectivity between the males and females within the big horn sheep population. A quick calculation of the degree (done previously) was merged with the data regarding each individual's sex, using the cbind function as shown below:

```
> Degree_sex <- cbind(Sex, Degree)
```

Which gave me the following table:

```
     Sex Degree
 [1,]   F     18
 [2,]   M     76
 [3,]   F      0
 [4,]   F     60
   ⋮
```

Using the information above, the mean degree for each sex was calculated and a Welch two sample t-test was also conducted that gave the following output:

```
> welch_tt <- t.test(network_female[network_female>0], network_male[network_male>0])
> welch_tt
          Welch Two Sample t-test
data:  network_female[network_female > 0] and network_male[network_male > 0]
t = -2.396, df = 24.444, p-value = 0.02457
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -46.24850  -3.46579
sample estimates:
mean of x mean of y
 22.57143  47.42857
```

This calculation shows us that the mean 'degree' for females is 22.57 but for males is 47.43. The t-value that it returns is -2.396 and the p-value is 0.025, suggesting that there is a significant difference in the connectedness between males and females, with the males being a lot more connected to the rest of the population compared to the females. However, the sample size use for the purpose of this tutorial is small, so the final value could be very different.

Similarly, there are a lot of other social network relatedness measures that can be extracted from the table. However, one of the main interests in this project is, also, to know how the members in the population are connected to each other i.e., how is the community structured? This is calculated by finding out the highly connected subgroups within the graph and then the eigenvector in each subgraph.

```
community <- leading.eigenvector.community(trial_ag)
groups <- list()
for (i in 1:max(community$membership)){
  groups[[i]] <- which(community$membership==i)
}
cols <- c("seagreen", "blue", "gold", "brown2", "rose")
plot(trial_ag, vertex.color=cols[Sex], vertex.size=4, vertex.label=NA, mark.groups = groups)
```

In this for loop, the membership of each individual is broken into smaller groups depending on their interactions with each other, based on the community eigenvector function. Plotting this graph gives me the following result, where each sub-group within the population has been highlighted in a different color

to further illustrate the sub-group connections within the population. As we can see, some individuals are highly connected in all three groups, while there are also individuals that are not connected at all.

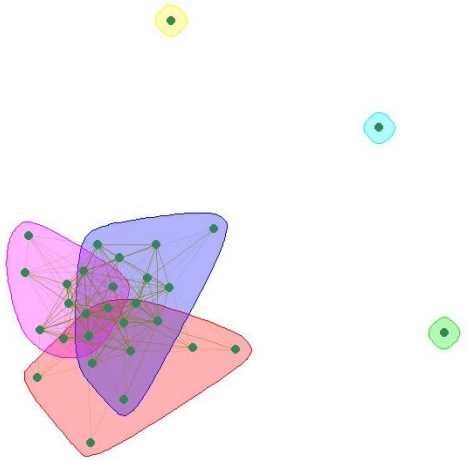

Figure 2. Sub-group connectivity within a population of big horn sheep

There is a lot more that can be done in R by using the package 'igraph', and this is only touching the surface of igraph capabilities, the functions that anyone chooses to use is entirely dependent on the focus of the project. My aim in this project tutorial was only to introduce the package and run through a few of the commands and functions to encourage someone who may be new to it to discover it further because there is a lot more to learn!