**Protocol – Data Cloud Geometry R Code**

The R code for data cloud geometry is designed to find multi-level structure in undirected network data (e.g. edgelist or matrix)[[Chen and Fushing 2012](#_ENREF_1)]. This multi-level structure is equivalent to finding community structure, similar to the Girvan-Newman method of finding network modularity [[Newman and Girvan 2004](#_ENREF_2)]. This method begins with a similarity matrix, where the entries of the matrix are a measure of how similar a pair of individuals is with respect to a particular feature. Similarity can be measured based on grooming relationships (where the strength of grooming per dyad is the measure of similarity) or other affiliative relationships. Similarity can also be measured using physical traits or genetic code.

The data cloud geometry method involves performing a random walk across all nodes in the network. The probability of making a step from one node to another is based upon their strength of similarity (i.e. from the value in the similarity matrix). Such random walks proceed locally, going to and from a local subset of nodes. Once a subset of nodes has been visited sufficiently (e.g. when one node from that community has been visited at least 5 times, M = 5) that node is removed from the network such that it may no longer be visited. Once all nodes from a community have been removed in this manner, the random walk is forced to jump to another community. This is how communities are defined – the subset of nodes that is each visited 5 times before jumping to a new set of nodes is one community.

Because the random walk proceeds based upon similarity, the range of variance in similarity will influence how close two nodes appear to be. Therefore, the similarity matrix must be evaluated at several different ‘temperatures’. For example**, the similarity matrix Sim** can be transformed to Sim^0.5, Sim^2, Sim^5, and Sim^10; the scale of variance within each matrix will allow the random walk to assess similarity at different levels.

1. Create an edgelist from raw data observations. An edgelist has two columns: Initiator and Recipient. You do not need to summarize the total frequency per dyad – simply assemble the raw interactions. DCG is designed for undirected networks, i.e. where the direction of the interaction doesn’t matter. Although grooming behavior is directed, the relationship represented by grooming might be considered undirected (i.e. friendship, kinship). Other relevant behavioral data include huddling and proximity.
2. Save your edgelist as a .csv file – this is a file format that is easily readable by R.
3. Open R and then open the DCG R code file (GetEns.R, from Chen Chen). Run the first several lines of code:
   1. Set your working directory (setwd)
   2. Indicate which file you want from that directory (file = “FileName.csv”), and read it in (data = read.csv(file, head=TRUE))
   3. Sort the unique subject IDs in your data set and call this variable “subjects”
   4. Create a rawsim similarity matrix from your edgelist using the rawsim = matrix function
4. Transform your rawsim matrix into a matrix whose values range from 0 to 1 by dividing this by the maximum value of the matrix. Call this Sim.
5. Copy the **MakeSeries** function into R – this is the random walk component of the DCG function. This function is quite long, about 1 full page of code and notes.
6. Next, copy **the EstClust function** into R – this is the primary DCG function. Note again that you are not yet running any analysis, just defining these functions in R so you can use them later.
7. Open the **GetEigen.R code file** – this is the code for the function GetNClust. Copy this function into R.
8. Now it’s time to run the EstClust function, and we need to do this at multiple different temperatures. Chen Chen suggests at least 20-30 different temperatures, because the number of distinct clusters in the network may be different at each, and these values can be plotted to note which cluster numbers hold stable for multiple temperatures (meaning this number is likely a logical division of the network) and which cluster numbers are ephemeral.
   1. Temperature 1 will be something like Sim^0.01 or Sim^2 (depending upon what values are in the Sim matrix).
   2. Run the following bits of code:
      1. Ens1 <- EstClust(Sim^0.01, MaxIt = 1000, m = 5)
      2. Tree1 <- hclust(as.dist(1-Ens1))
      3. plot(Tree1)
      4. GetNCluster(Ens1)
   3. Repeat similar runs across your selected range of 20-30 different temperatures
9. Once you have run all temperatures, inspect each eigenvalue plot using the code GetNClust(Ens1). These plots show the approximate number of clusters. Red dots connected by a red line represent a cluster. The plot below shows two red lines that connect three red dots on the left, which indicates the presence of two clusters.



1. After inspecting the eigen plots, choose an ensemble matrix (Ens1, Ens2….Ens20) whose number of clusters appears to be stable across multiple temperatures. For example, if matrices Ens1 – Ens4 all show two clusters, Ens5 shows three clusters, and Ens6 – Ens20 all show four clusters, then the divisions of two or four clusters are likely reasonable divisions of your network.
2. Look at the Tree plot of the Ens matrix you have chosen to see how your nodes are divided into clusters.
3. The Ens matix contains valuable information as well. These values range from 0 to 1 and represent the proportion of the 1000 iterations in which each pair of nodes appeared in the same cluster.

**References**

Chen C, Fushing H. 2012. Multiscale community geometry in a network and its application. Physical Review E 86:041120.

Newman MEJ, Girvan M. 2004. Finding and evaluating community structure in networks. Physical Review E., Statistical, Nonlinear, and Soft Matter Physics 69:026113.