
Problem Set #2: Cohesive Subgroups & Core/Periphery Structures

Adapted from original by Peter V. Marsden, Harvard University

This problem set provides an overview of UCINET routines that are useful for identifying cohesive subgroups. I refer to those parts of the software that locate mathematically defined subsets of points in a graph, like cliques, n-cliques, k-plexes, k-cores, or components; most of these approaches can be found in the Network/Subgroups menu. I also mention some tools that can be helpful in clustering approaches to subgroup identification.

We will use the “Games” relation in the “Wiring” data set. It has an unusually clear structure to it, which makes it a good exemplar for using techniques like these for the first time. You should also use these techniques on the Padgett Marriage relations. In this case you may compare your results with those found in chapter 7 of Wasserman and Faust. Most of these routines assume that the dataset contains 1 relation (e.g., one worksheet); you will need to create single relation datasets for the analysis required here.

As before, be sure to answer the questions or do the work assigned in the parts in ***bold and italics***. Remember to do the extra analysis needed to get the 10th point.

Mathematically defined subsets of points

Most of the mathematical definitions of subgroups are formulated for binary data found in a graph or digraph. Many of them *require* that data be binary, and that they be symmetric. The “Games” relation is both, but you might have to use transformations in order to create binary and/or symmetric data yourself in other settings.

Cliques

The most basic mathematical model for a cohesive subgroup is the clique, or “maximally complete subgraph.” This is, of course, a demanding model; even one asymmetric relationship (as opposed to a reciprocal one) within a set of actors means that the set does not conform to it. A routine that enumerates cliques is found in the Network/Subgroups/Cliques menu.

When you run this program, it will begin by listing the sets of nodes that are found in fully-connected substructures. That listing is, in fact, the main result of the “clique-finding” algorithm.

The routine produces other results, including a tree diagram or dendrogram; see below for discussion of these. To see the clique listing and other text output, close the graphics window.

Run the cliques routine on the Padgett marriage data and the Games relation of the Wiring data. Include the output in your write-up.

N-Cliques and K-Plexes

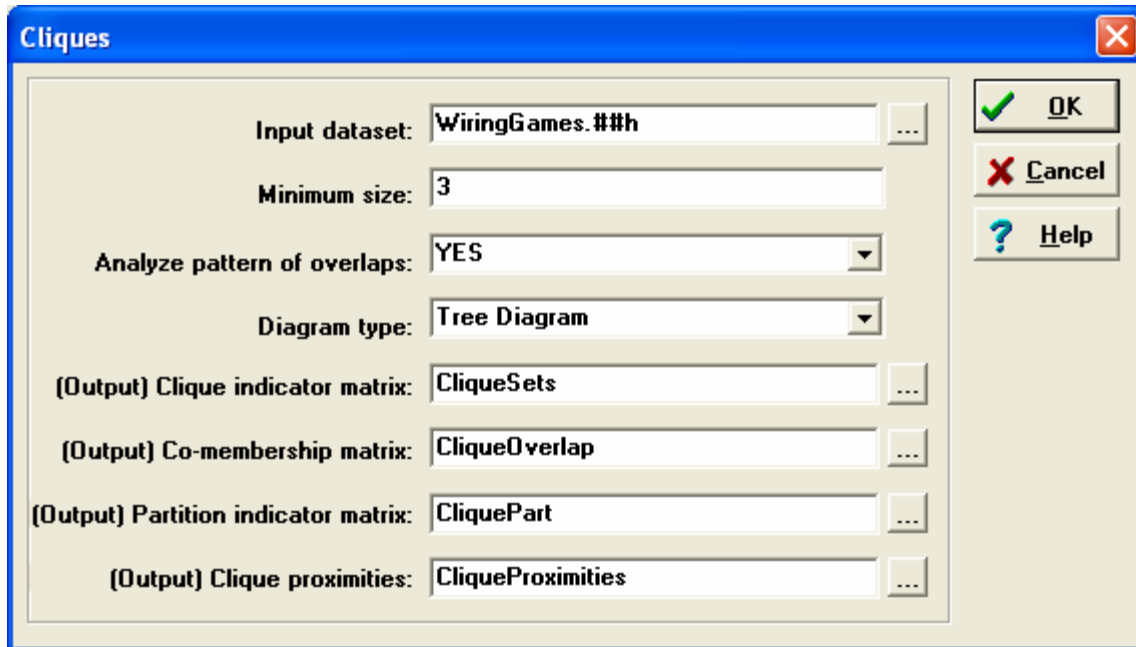
Other options in the Network/Subgroups menu allow you to identify other cohesive substructures. N-cliques are like cliques, except that they include individuals in a subgroup when they are linked by indirect paths of length less than or equal to N. Hence a clique can be seen as a 1-clique. K-plexes are substructures in which each actor is connected to all but K of the other actors. Since actors are generally not connected to themselves, a clique is a 1-plex. The greater K, the less demanding the density criterion for creating subgroups.

The N-clique and K-plex routines generate a listing of sets of nodes meeting the criterion for defining each type of substructure. You can control N and K in the menu that defines input for the routine.

Run the N-cliques and K-plex routines on the Padgett marriage data and the Games relation of the Wiring data. Try at least two levels of N or K. Include the output in your write-up.

Clustering the Overlapping Memberships of Cohesive Subgroups

Two or more actors may be found together in more than one of these cliques, N-cliques, or K-plexes. The program also constructs a “group co-membership matrix” that gives the number of cohesive substructures in which two actors appear together. Let’s look at parameters for the cliques routine:



Note that UCINET is going to output four matrices – the clique indicator matrix, the co-membership matrix, the partition indicator matrix, and the clique proximities matrix. The default file names for these matrices are rather generic. You probably want to create new names for each.

The co-membership matrix can be treated as an indicator of the extent to which two actors “belong together.” The standard applied is “how often do these actors appear together in cliques?” The more often they appear together, the more they belong together. The clique proximities matrix tells you what proportion of the members of a given clique are adjacent to the each node in the dataset. For cliques, the proportion will be 1.00 for a clique in which they are a member and something lower for all other cliques. Clique proximity is another measure of “belongs together.” For some actors, they may not be a “member” of a clique, but may have relationships with almost all of the members.

To motivate the remainder of this discussion, first re-run the cliques analysis for the Games relation. Let’s look at the textual output. Try to get an intuitive feel for this idea by looking at the first part of the output, which lists the cliques. ***Which actors seem to appear together in the five cliques identified? Put your answer in the write-up (don’t peek at the analytical answer you’ll get below).***

Instead of relying on inspection, which will fail with larger networks, we can find an analytic answer. By applying a hierarchical clustering algorithm to this matrix of overlaps, a partition of the actors into discrete subsets of “usually together” actors can be found. This is done in using “single-link hierarchical clustering”; the output from which is included with the cliques analysis. (“Single-link” clustering is only one option for hierarchical clustering, but it is the one that is automatically executed when these routines run. We will talk about some other clustering criteria and say what “single link” means later on.) The results of the clustering are reported in graphical

form (in a tree diagram or a dendrogram that is actually the first thing you see when you run this routine), and in the form of a cluster mapping at the end of the text output for the Cliques routine.

Scroll down until you find the “hierarchical clustering of equivalence matrix”. At its left we see a “Level” column, and then we see a column for each actor that has some dots and some Xs. The Xs indicate the actors that are grouped together at a given “level” of clustering. So, for example, in the output for the cliques routine applied to the Games data, we see that actors 3 (W1), 5 (W3), and 6 (W4) are clustered together at level 3. This means that these three actors are found together in three different cliques. Moving down to level 2, we see that two sets of actors are found together in at least two cliques: W1, W2, W3, W4 and S1; and W7, W8, and W9. Moving down one step further to groups of actors who are found together in at least one clique, we see that I1 and W5 get added to the first group, while W6 and S4 get added to the second one. At this level 1, we see that the group has been divided into two major factions, with two isolates (I3 and S2).

At level 0, all actors are clustered together. All actors are always grouped in one big cluster at the end of a hierarchical clustering routine; it is a matter of judgment to decide what cluster partition (someplace intermediate between keeping all actors separate and clustering them all together) is most informative in revealing the structure of a group.

The tree diagrams and dendrograms are read similarly. (You choose which graphic you want as an option in the menu.) The clustering process proceeds from left to right. At the left, we see each actor as a separate cluster; at the right we see that all actors have been joined into a single cluster. At the top, we see the criterion (here, the number of overlapping memberships in cliques) that has been used as a criterion for combining two or more actors into a cluster.

In the Wiring data, “W” refers to a wirer; “I” to an inspector “S” to a solderer. Given that this is all that you know about these people – that they belong to some cliques and that they co-belong in some cases, what might you infer about the different actors? Write a paragraph or two; think creatively – there is no right or wrong answer.

Imposing a partition on a data matrix and obtaining subgroup densities

The different partitions of actors generated by a hierarchical clustering routine can be superimposed on the original data via the Network/Network Properties/Density menu. This can provide a reasonably compact summary of the structure of a given set of ties.

The Cliques routine generates a “partition-by-actor” matrix that – by default – is called CliquePart (see the graphic above). The other routines generate similar partitioning matrices with different default names. In other UCINET routines, you can refer to successive partitions in this. The first partition refers to the most demanding clustering level, and the last partition to that in which all actors are placed together in a single cluster.

In the Network/Network Properties/Density menu, you can enter a “row partitioning/blocking” and a “column partitioning/blocking.” This will re-order the rows and columns of a data matrix (here, you should again use “Games”), and then report the subgroup densities within and between groups of actors created in the partitioning.

We will be using the Wiring data again. Start off by entering “CliquePart col 1”(if you changed the name of the clique partition output file, replace “CliquePart” with the name of the file that you entered) in both the row and column partitioning/blocking boxes. This is the first partitioning, that corresponds to clustering at 3 overlaps. It will collapse W1, W3, and W4 into a single group, but everyone else will be an individual actor.

Then successively alter the row and column partitioning/blockings to be “CliquePart col 2,” and repeat with “CliquePart col 3”. This will decrease the number of groups in the density matrix. You should see the structure of two cohesive subgroups appear, particularly for “CliquePart col 3”. Members of these subgroups will have dense relations with one another, but there are almost no game-playing ties across the group boundaries. Of course, you will also see the two isolates.

If you go one step further and use “CliquePart col 4” as the partition/blocking, you will see that everyone has been combined into one group, corresponding to the clustering at level 0. Include all your output in your write-up.

Components of a Graph

The “components” of a graph are subsets of actors who meet a particular criterion of connectedness. Strongly connected actors are joined by a path in each direction (this need not be the *same* path, however). Weakly connected actors are joined by a semi-path in each direction. “Unilaterally” connected actors are joined by a path in one direction, but not the other. Note that these definitions place no restrictions on the *length* of the paths or semi-paths involved.

Components of a graph can be identified using the Network/Regions/Components menu. You can choose to identify strong or weak components, and specify the minimum size of components to be reported.

Try this out on the Wasserman/Faust children’s friendship citations data set that was created in Problem Set 1. Specify that you want strong components, and set the minimum size to report as 1, so that every child will be assigned to a component (some will end up in “components” consisting of themselves only).

Once you’ve done this, you can superimpose the component assignments on the original data using the Network/Properties/Density menu. The component partition vector to use is named `Components_Partition` by default.

You can take this one step further and create the so-called “condensed digraph” in which all nodes in a component are collapsed into a single node; it provides a simple model of the

asymmetries in a network (a condensed digraph for strong components must be strictly asymmetric, since all reciprocal connections must be contained within strong components; the condensed digraph for a set of weak components must be disjoint).

To obtain the condensed digraph, use the Transform/Dichotomize menu, and enter the “Density” matrix produced by the Network/Network Properties/Density menu as the input matrix. Dichotomize at zero, the default value. This will produce a binary matrix (“DichDensity” is its default name) indicating the relationships among the components of the graph. Include all output in your write-up.

Finally, another way to find highly interconnected “regions” in a network is by apply the K-core criteria. A K-core is a group of actors in the network that are connected to at least K members of the subset. Using this routine it is possible to identify core-periphery differentiation in a network. I have used this extensively in research on policy networks (i.e., sets of organizations that have an abiding interest in some area of policy because they are all interdependent with respect to the decisions made by public officials). However, this concept can be used for all sorts of research.

Apply the K-core routine to the Games relation in the Wiring data. Does there appear to be a core-periphery structure? Why or why not? Compare the K-core findings to the earlier clique findings. How are the analyses similar or different? Conceptually, how are cliques different from core-periphery structures? How are they similar?

Many of the tools reviewed in this problem set – clustering, superimposing partitions, dichotomizing density matrices – will be useful for us in the course of positional analysis, which is one of the most important applications of network analytic techniques.

Independent analysis: Remember, to get full credit for the assignment, you need to undertake some additional analysis of your own choosing. In this case, you may wish to use the core/periphery routines built into UCINET and then compare the findings with the K-core output. However, it is up to you to decide what additional analysis you wish to do.