Part II - Cohesion

Solidarity, shared norms, identity, collective behavior, and social cohesion are considered to emerge from social relations. Therefore, the first concern of social network analysis is to investigate who are related and who are not. Why are some people or organizations related, whereas others are not? The general hypothesis here states that people who match on social characteristics will interact more often and people who interact regularly will foster a common attitude or identity.

In this part of the book, which covers Chapters 3 through 5, we discuss several measures of cohesion. You will learn to detect cohesive subgroups within several types of social networks.

3 Cohesive subgroups

3.1 Introduction

Social networks usually contain dense pockets of people who 'stick together.' We call them cohesive subgroups and we hypothesize that the people involved are joined by more than interaction. Social interaction is the basis for solidarity, shared norms, identity, and collective behavior, so people who interact intensively are likely to consider themselves a social group. Perceived similarity, for instance, membership of a social group, is expected to promote interaction. We expect similar people to interact a lot, at least more often than with dissimilar people. This phenomenon is called *homophily*: birds of a feather flock together.

In this chapter, we present a number of techniques to detect cohesive subgroups in social networks, all of which are based on the ways in which vertices are interconnected. These techniques are a means towards an end rather than an end in themselves. The ultimate goal is to test whether structurally delineated subgroups differ with respect to other social characteristics, for instance, norms, behavior, or identity. Does the *homophily* principle work; may we conclude that a cohesive subgroup represents an emergent or established social group?

3.2 Example

In 1948, American sociologists executed a large field study in the Turrialba region, which is a rural area in Costa Rica (Latin America). They were interested in the impact of formal and informal social systems on social change. Among other things, they investigated visiting relations between families living in 75 *hacienda's* (farms) in a neighborhood called San Juan Sur.

Visiting relations are classified according to three types: visits among kin, visits among families bound by god-parent or god-child ties ('church relations'), and other visits. We will focus on visits among kin first. The network of visiting relations (see Figure 1) is a simple directed graph and line values and loops are meaningless (`SanJuanSur_kin.net`). We will compare cohesive subgroups in this network with a classification of the families into family-friendship groupings which was made by the researchers (`SanJuanSur_grouping.clu`). In rural areas where there is little opportunity to move up and down the social ladder social groups are usually based on family relations. All relevant data are collected in the project file `SanJuanSur.paj`.

Which cohesive subgroups can we find in the San Juan Sur network and do they match the family-friendship groupings? Figure 1 offers a visual impression of the kin visits network and the family-friendship groupings, which are identified

by the colors of vertices. As we can see, the network contains large as well as small pieces and the largest piece is loosely knit but it has a dense center around family f69. Most families which belong to one grouping are connected by visiting relations, so they are rather close in the network. Exceptions occur, however, notably the 'blue' family f39 (bottom left), which is expected to be 'green'. In subsequent sections, we will set out this first impression in detail.

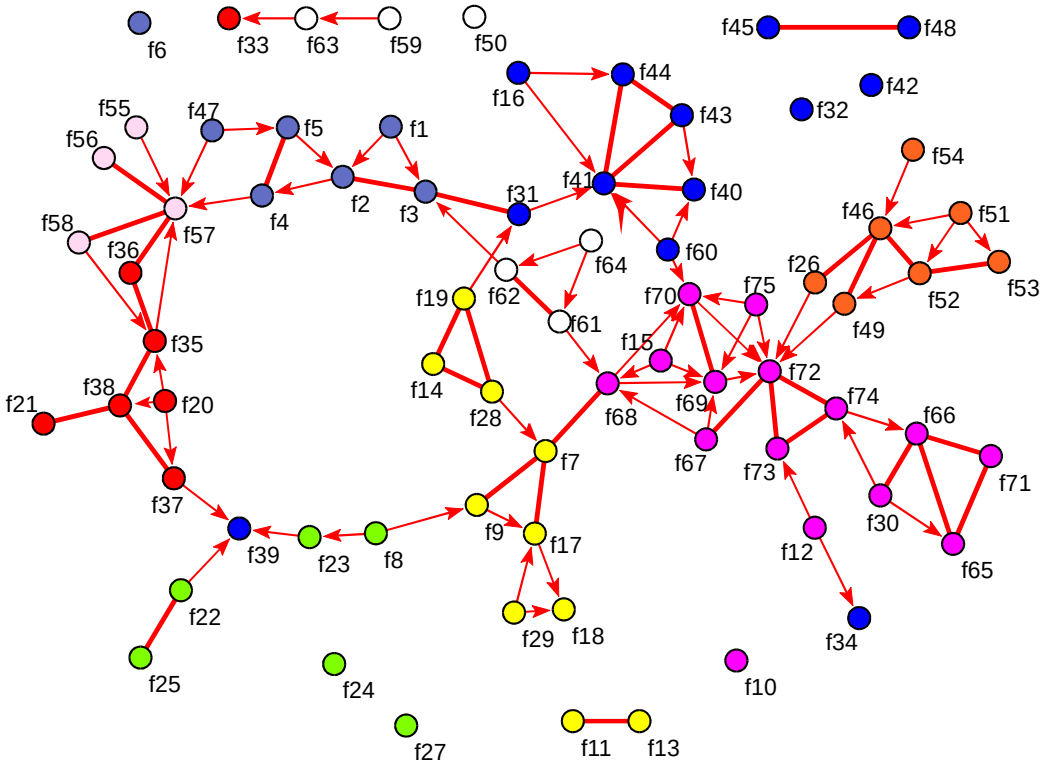


Figure 1 - Visiting relations among kin in San Juan Sur.

3.3 Density and degree

Intuitively, cohesion means that a social network contains many ties; more ties between people yield a tighter structure, which is, presumably, more cohesive. In network analysis, the density of a network captures this idea. It is the percentage of all possible lines which are present in a network. Maximum density is found in a complete simple network, that is, a simple network in which all pairs of vertices are linked by an edge or by two arcs, one in each direction. If loops are allowed, all vertices have loops in a complete network.

- **Density** is the number of lines in a simple network, expressed as a proportion of the maximum possible number of lines.
- A **complete network** is a network with maximum density.

In this definition of density, multiple lines and line values are disregarded. Intuitively, multiple lines between vertices and higher line values indicate more cohesive relations. Although density measures have been proposed which account for multiplicity and line values, we prefer not to present them. We count distinct

lines only, which means that we treat a multiple line as one line and multiple loops as one loop. We will discuss other measures which capture the contribution of multiple lines and line values to cohesion in Chapter 5.

In the kin visiting relations network, density is 0.024, which means that only 2.4 percent of all possible arcs are present. It is very common to find density scores as low as this one in social networks of this size. Density is inversely related to network size: the larger the social network, the lower the density because the number of possible lines increases rapidly with the number of vertices whereas the number of relations which each person can maintain is limited. In a visiting relations network, there is a practical limit to the number of families you can visit. Therefore, including more families in the network will reduce network density.

This is a problem if you want to interpret or compare network density. Density in the visiting network in Attiro, which is another neighborhood in the Turrialba region, is 0.041. This is slightly higher than in San Juan Sur but the difference may be due to a smaller number of families in Attiro (60 families). Therefore, we can not draw a conclusion from this comparison.

The **degree** of a vertex is the number of lines incident with it.

Network density is not very useful because it depends on the size of the network. It is better to look at the number of relations in which each vertex is involved. This is called the degree of a vertex. Vertices with high degree are more likely to be found in dense sections of the network. In Figure 1, family f69 is connected to six families by seven visiting relations (note that the edge between f69 and f70 means that they are linked by mutual visits), so its degree is seven. The lines incident with this family contribute substantively to the density of the network near this family.

A higher degree of vertices yields a denser network, since vertices entertain more relations. Therefore, we can use the **average degree** of all vertices to measure the structural cohesion of a network. This is a better measure of overall cohesion than density because it does not depend on network size, so average degree can be compared between networks of different sizes.

- Two **vertices** are adjacent if they are connected by a line.
- The **indegree** of a vertex is the number of arcs it receives.
- The **outdegree** is the number of arcs it sends.

In a simple undirected network, the degree of a vertex is equal to the number of vertices which are adjacent to this vertex: its neighbors. Each line that is incident with the vertex connects it to another vertex since multiple lines and loops, which contribute to the degree of a vertex without connecting it to new neighbors, do not occur. In a directed network, however, there is a complication because we must distinguish between the number of arcs received by a vertex (its indegree) and the number of arcs send (its outdegree). Note that the sum of the indegree and the outdegree of a vertex does not necessarily equal the number of its neighbors,

e.g. family f69 is involved in seven visits but it has six adjacent families because family f70 is counted twice (Figure 1).

In this section, we will restrict ourselves to degree in undirected networks. When we encounter a directed network, we will symmetrize it, which means that we turn unilateral and bi-directional arcs into edges. Discussion of indegree in directed networks will be postponed until Chapter 9, which presents the concept of prestige.

To **symmetrize** a directed network is to replace unilateral and bi-directional arcs by edges.

Application

Info>Network>General

Let us analyze the network of visiting relations among kin in San Juan Sur (`SanJuanSur_kin.net`), which does not contain multiple lines and loops. In Pajek, the density of a network can be obtained by means of the *Network* submenu of the *Info* menu. Choose the command *General* to display basic information on the selected network, such as the number of vertices and lines as well as its density. When executed, this command displays a dialog box asking the user to specify the number of lines to be displayed. When you are only interested in network density, request zero lines. Pajek computes two density indices in the Report screen. The first index allows for loops and the second does not. Since loops are meaningless in a visiting relations network - people do not visit themselves - the second index is valid. Density in the directed network is 0.024.

Net>Transform >Arcs -> Edges>All

Net>Transform>Remove >Multiple lines

In undirected simple networks, the degree of a vertex is equal to its number of neighbors. This is the easiest interpretation of degree, so we will concentrate on undirected simple networks in this section. The kin visiting network, however, is directed, so we must symmetrize it first. Use the *Arcs->Edges>All* command in the *Net>Transform* submenu to replace all arcs by edges. Pajek will ask whether you want to make a new network and we advise you to do so. Now, bilateral visiting relations are replaced by two edges because each unilateral arc is turned into an edge, so we must delete multiple lines from the new network to obtain a simple network. This can be done with the *Remove>Multiple lines* command in the *Net>Transform* submenu. It does not matter which command you choose with regard to values of multiple lines, because we will not pay attention to line values anyway. Now, the network is symmetrized and it is simple because multiple lines are removed and there were no loops in the original network.

Net>Partitions>Degree

Degree is a discrete attribute of a vertex (it is always an integer), so it is stored as a partition. We obtain the degree partition with a command from the *Net>Partitions>Degree* submenu. Select the command *Input* or *Output* and not the command *All* because the last command counts lines twice: it treats them as the equivalent of two arcs, namely one arc send and one received by the vertex. The command *All* should only be used for networks which contain arcs as well as edges. At first sight, it may seem strange that you should select *Input* or *Output*

when you are analyzing an undirected network since these commands refer to indegree and outdegree in a directed network. But an edge is simply regarded as a bi-directional arc, so you may count the received or send arcs to obtain the number of edges incident with a vertex.

Info>Partition

The command *Info>Parition* displays the partition as a frequency table (see Table 1). Class numbers represent degree scores, so we can see that the degree of vertices [wdn1]varies markedly from one to eight neighbors. Clearly, family f72 is connected to most families by kinship visits. Seven families, including family f6, are isolated in the network: they are not linked to other families by kinship visits. Maybe, they have no kin within San Juan Sur.

Table 1 - Frequency distribution of degree in the symmetrized network of visits.

Class	Freq	Freq%	CumFreq	CumFreq%	Representative
0	7	9.3333	7	9.3333	f6
1	12	16.0000	19	25.3333	f11
2	17	22.6667	36	48.0000	f1
3	23	30.6667	59	78.6667	f4
4	8	10.6667	67	89.3333	f2
5	2	2.6667	69	92.0000	f35
6	4	5.3333	73	97.3333	f41
7	1	1.3333	74	98.6667	f57
8	1	1.3333	75	100.0000	f72
Sum	75	100.0000			

Partition>Make Vector

Info>Vector

The average degree can be calculated from the degree distribution. In this example, the class numbers in the degree partition represent integers, namely the number of neighbors of a vertex, but this is not true for all partitions. As a consequence, no average class number is calculated and presented by the *Info>Partition* command. To obtain the average degree, we have to convert the degree partition to a vector and calculate summary statistics on the vector. In Chapter 2, you have learned how to do this: use *Partition>Make Vector* to create a vector from the degree partition and use the *Info>Vector* command to obtain the average, which is 2.61. Families in San Juan Sur meet approximately two and a half families on average.

3.4 Components

Vertices with a degree of one or higher are connected to at least one neighbor, so they are not isolated. However, this does not mean that they are necessarily connected into one ‘lump’. Sometimes, the network is cut up in pieces. Isolated sections of the network may be regarded as cohesive subgroups because the vertices within a section are connected whereas vertices in different sections are not. The network of visits among kin in San Juan Sur is clearly not connected (see Figure 1). In this section, we will identify the connected parts of a network, which are called components, but we must introduce some auxiliary graph theoretic concepts first.

Let us have a look at a simple example (Figure 2). Intuitively, it is clear that some vertices are connected to other vertices whereas others are not, for instance, vertex v2 is not adjacent to any other vertex, but the other four vertices have one or two neighbors. If we consider the arcs to be roads, we can walk from vertex v5 to v3 and, not considering the direction of the arcs, we can proceed from vertex v3 to v1. We say that there is a semiwalk from vertex v5 to vertex v1. From vertex v2, however, we can walk nowhere.

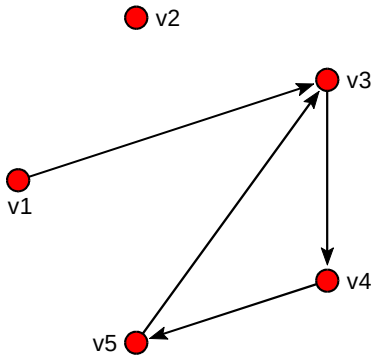


Figure 2 - A small simple directed network.

- A **semiwalk** from vertex *u* to vertex *v* is a sequence of lines such that the end vertex of one line is the starting vertex of the next line and the sequence starts at vertex *u* and ends at vertex *v*.
- A **walk** is a semiwalk with the additional condition that none of its lines is an arc of which the end vertex is the arc's tail.

Imagine that the arcs represent one-way streets, so we take into account the direction of the arcs. Now, we can drive from vertex v5 to vertex v3 but we cannot arrive at vertex v1. In graph theory, we say that there is a walk from vertex v5 to v3 but there is not a walk from vertex v5 to v1.

Walks and semiwalks are important concepts but we need another, related concept to define whether a network is connected. We should note that there are many - in fact, infinitely many - walks from vertex v5 to v3 in our example, for instance, v5→v3→v4→v5→v3 is also a walk and we may repeat the circular route v5→v3→v4→v5 as many times as we like. Clearly, we do not need these repetitions in order to establish whether vertices are connected, so we use the more restricted concepts of paths and semipaths, which demand that each vertex on the walk or semiwalk occurs only once, although the starting vertex may be the same as the end vertex. In the example, the walk v5→v3 is a path but the walk v5→v3→v4→v5→v3 is not because vertices v5 and v3 occur twice. A path is more efficient than a walk, one might say, because it does not pass through one junction more than once.

- A **semipath** is a semiwalk in which no vertex in between the first and last vertex of the semiwalk occurs more than once.
- A **path** is a walk in which no vertex in between the first and last vertex of the walk occurs more than once.

Now, we can easily define the requirements which a network must meet to be connected. A network is weakly connected - often we just say connected - if all vertices are connected by a semipath. In a (weakly) connected network, we can 'walk' from each vertex to all other vertices if we neglect the direction of the arcs - if there are any. The example of Figure 2 is not connected because vertex v2 is isolated: it is not included in any semipath to the other vertices.

In directed networks, there is a second type of connectedness: a network is strongly connected if each pair of vertices is connected by a path. In a strongly connected network, you can travel from each vertex to any other vertex obeying the direction of the arcs. Strong connectedness is more restricted than weak connectedness: each strongly connected network is also weakly connected but a weakly connected network is not necessarily strongly connected. Our example is not weakly connected, so it can cannot be strongly connected.

- A network is (weakly) **connected** if each pair of vertices is connected by a semipath.
- A network is **strongly connected** if each pair of vertices is connected by a path.

Although the network of our example is not connected as a whole, we can identify parts which are connected, for instance, vertices v1, v3, v4, and v5 are connected. In comparison with the isolated vertex v2, these vertices are relatively toghtly connected, so we may say that they are a cohesive group. If the relations represent communication channels, all vertices except vertex v2 may exchange information. Vertices v1, v3, v4, and v5 constitute a (weak) component because they are connected by semipaths and there is no other vertex in the network which is also connected to them by a semipath.

Formally, we say that a (weak) component is a maximal (weakly) connected subnetwork. Remember that a subnetwork consists of a subset of the vertices of the network and all lines between these vertices. The word **maximal** means that no other vertex can be added to the subnetwork without destroying its defining characteristic, in this case connectedness. If we would add the only remaining vertex - v2 - the subnetwork is no longer connected. In contrast, if we would omit any of the vertices v1, v3, v4, or v5, the subnetwork is not a component because it is not maximal: it does not comprise all connected vertices.

Likewise, we can define a strong component, which is a maximal strongly connected subnetwork. The example network contains three strong components. The largest strong component is composed of vertices v3, v4, and v5, which are connected by paths in both directions. In addition, there are two strong components consisting of one vertex each, namely vertex v1 and v2. Vertex v2 is isolated and there are only paths from vertex v1 but no paths to this vertex, so it is not strongly connected to any other vertex. It is asymmetrical linked to the larger strong component. In general, the ties among strong components are either asymmetrical or absent. In Chapter 10, we will elaborate on this feature.

- A (weak) **component** is a maximal (weakly) connected subnetwork.
- A **strong component** is a maximal strongly connected subnetwork.

In an undirected network, lines have no direction, so each semiwalk is also a walk and each semipath is also a path. As a consequence, there is only one type of connectedness, which is equivalent to weak connectedness in directed networks, and one type of component. In an undirected network, components are isolated from one another, there are no lines between vertices of different components. This is similar to weak components in directed networks.

In a directed network, should you look for strong or weak components? The choice depends on substantive and practical considerations. Substantive reasons pertain to the importance you attach to the direction of relations: does it matter to social processes whether actor A turns to actor B, actor B turns to actor A, or both? If the flow of communication is being investigated, it probably does not matter who initiates a contact. If family f22 visits both families f25 and f39 (Figure 3, bottom left), it may inform family f25 about family f39 and the other way around. Families f25 and f39 may share information although there is no path between them. In this case, direction of relations is quite unimportant and weak components are preferred.

If substantial arguments are indecisive, the number and size of components may be used to choose between strong and weak components. Recall that strong components are more strict than weak components, which means that strong components usually are smaller than weak components. It is a good strategy to detect weak components first. If a network is dominated by one large weak component, e.g., the kin visits network at San Juan Sur, we advise to use strong components to break down the weak component in a next step.

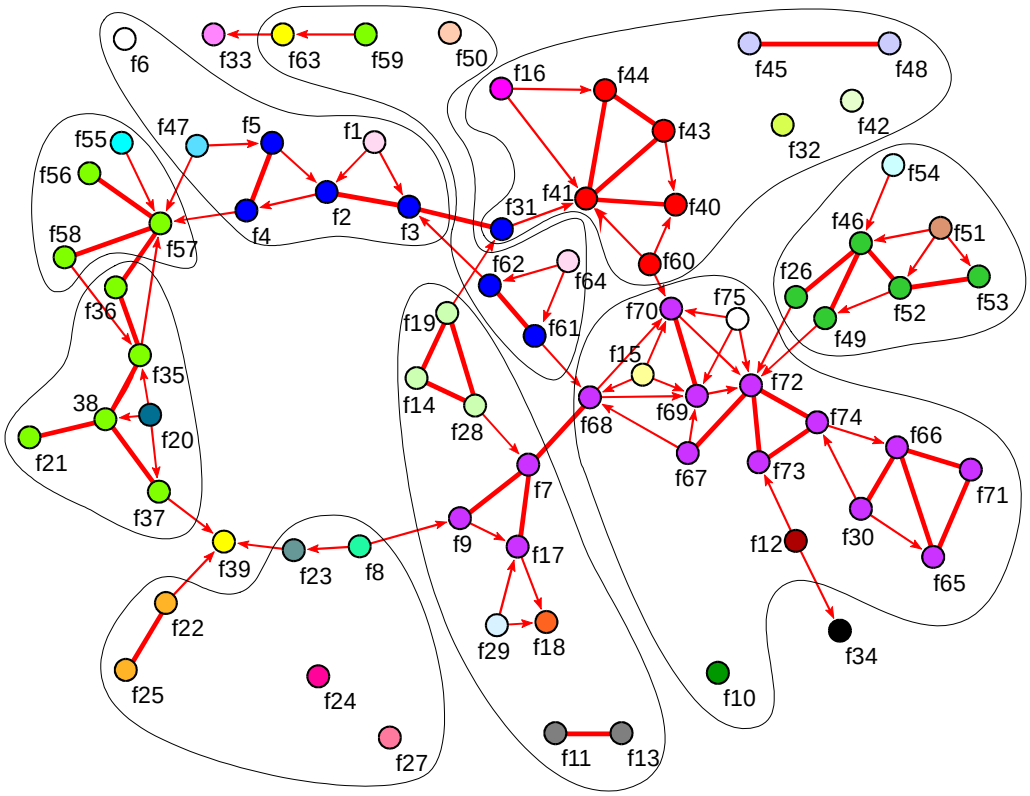


Figure 3 - Strong components (colors) in the kin visits network at San Juan Sur.

Figure 3 shows the strong components in the kin visiting network. Each component is identified by the color of its vertices. The original classification according to family-friendship groupings is represented by contours which we have added manually. We see that the largest weak component is split up in several small strong components, some of which approximate family-friendship groupings, for instance, the purple and green components.

Application

Net>Components>Strong

With Pajek, it is easy to find components in the kin visits network (`SanJuanSur_kin.net`). The *Net* menu has a submenu to find three types of components: strong, weak, and bi-components. We will discuss bi-components in Chapter 7. When you execute commands *Strong* or *Weak*, a dialog box appears asking for the minimum size of components. Sometimes, very small components are not interesting, for instance isolated vertices, which are counted as separate components if minimum component size is set to 1 vertex. Raise this number to exclude them.

Net>Components>Weak

In undirected networks, it makes no difference whether you select strong or weak components because the commands yield identical results. Furthermore, weak components in a directed network are equal to components in the symmetrized network. Therefore, it is not necessary to symmetrize a directed network when you want to know its components: just compute weak components in the directed network.

3.5 Cores

The distribution of degree reveals local concentrations of ties around individual vertices but it does not tell us whether vertices with high degree are clustered or scattered all over the network. In this section, we will use degree to identify clusters of vertices which are tightly connected because each vertex has a particular minimum degree within the cluster. We do not pay attention to the degree of one vertex but to the degree of all vertices within a cluster. These clusters are called *k*-cores and *k* indicates the minimum degree of each vertex within the core, for instance, a 2-core contains all vertices which are connected by degree two or more to other vertices within the core. A *k*-core identifies relatively dense subnetworks, so they help to find cohesive subgroups. As you will see, however, a *k*-core is not necessarily a cohesive subgroup itself!

A ***k*-core** is a maximal subnetwork in which each vertex has at least degree *k* within the subnetwork.

The definition of a *k*-core is more complicated than you might think at first sight. It is easiest to explain if we apply it to a simple undirected network and, as a rule, we will only apply it to this type of network. In a simple undirected network, the degree of a vertex is equal to the number of its neighbors as we have learned in Section 3.3, so a *k*-core contains the vertices which have at least *k* neighbors

within the core. A 2-core, then, consists of all vertices which are connected to at least two other vertices within the core. In the definition, the word *maximal* means that we are interested in the largest set of vertices which satisfy the required property, in this case a minimum number of *k* neighbors within the core.

The undirected kin visits network, which we obtained by symmetrizing the directed network, contains a 3-core (red vertices in Figure 4), a 2-core (green), a 1-core (yellow) and a 0-core (blue). In the 3-core, each family is connected to at least three other families. A subset of this core, e.g., families f69, f70, f72, and f75, is also internally connected with minimum degree three. However, this subset is not a 3-core, because we can add vertices which also belong to the 3-core (f15, f67, and f68); the subset is not maximal.

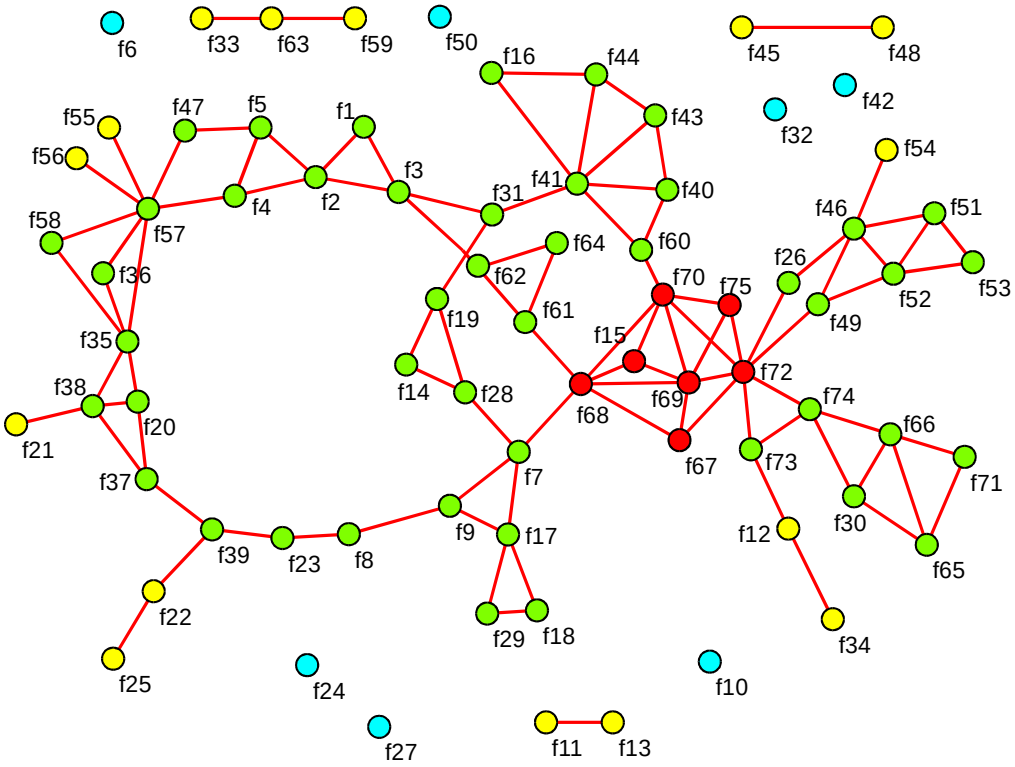


Figure 4 - *k*-cores in the kin visits network at San Juan Sur.

Do the *k*-cores in the kin visits network represent cohesive subgroups? For the 3-core, this seems to be true because it is clearly a dense pocket within the network. Now imagine that family f30 (at the right of Figure 4) would be connected to family f71 and not to family f74. Then, the network would have consisted of a 3-core as depicted in Figure 5. In this situation, we would conclude that there are two different cohesive subgroups within the 3-core because there are two groups with many in-group ties but there are no direct ties between the groups. Technically, however, all vertices belong to one 3-core because for all of them we can say that they have three neighbors within the core. The definition does not stipulate that a *k*-core is connected

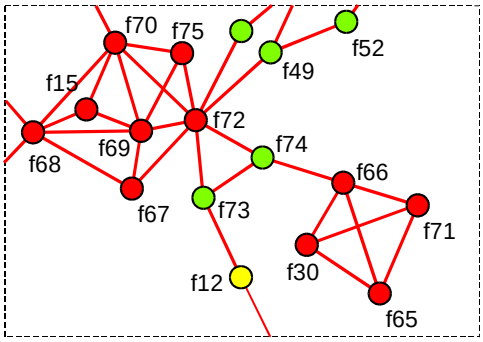


Figure 5 - A 3-core containing two cohesive subgroups (detail).

How can we distinguish between the two cohesive groups in the 3-core of Figure 5? In order to do so, we must understand that *k*-cores are **nested**: a vertex in a 3-core is also part of a 2-core, but not all members of a 2-core belong to a 3-core. Family f72, for instance, belongs to a 3-core because it has three neighbors (families f69, f70, and f75) which are interconnected. Of course, this implies that family f70 has two neighbors which are interconnected, e.g., families f69 and f75, which is a sufficient condition to belong to a 2-core. In Figure 5, imagine that red vertices are colored yellow, green, and red in succession: first they are part of a 1-core, then they are found to belong to a 2-core, and finally they are painted red because they constitute a 3-core. This illustrates the nesting of *k*-cores.

As a result of nesting, different cohesive subgroups within a *k*-core are usually connected by vertices which belong to lower cores. In Figure 5, family f74, which is part of the 2-core, connects the two segments of the 3-core. If we eliminate the vertices belonging to cores below the 3-core, we obtain a network consisting of two components, which identify the cohesive subgroups within the 3-core.

This is exactly how cohesive subgroups are found by means of the *k*-cores concept: remove the lowest *k*-cores from the network until the network breaks up into relatively dense components. Then, each component is considered to be a cohesive subgroup because they have at least *k* neighbors within the component. Note that the vertices which have been eliminating in the process are not part of a cohesive subgroup.

Application

Net>Partitions>Core >Input (or Output)

In Pajek, *k*-cores are detected with the *Core* command in the *Net>Partitions* submenu. For undirected networks, choose command *Input* or *Output* but not command *All* because command *All* counts edges twice. The command yields a partition which assigns each vertex to the highest *k*-core to which it belongs. Figure 4 displays the *k*-core partition. Note that isolated vertices are light blue; they belong to the 0-core. The *k*-core partition nicely shows that the kin visiting network contains one dense section (the 3-core with seven families) which is nested in a less dense group which is, in turn, surrounded by loosely connected region.

3.6 Cliques and complete subnetworks

In the kin visits network, most vertices belong to one large 2-core. If we want to split this large 2-core into subgroups, we need a stricter definition of a cohesive subgroup. In this section, we present the strictest structural form of a cohesive subgroup, which is called a clique: a set of vertices in which each vertex is directly connected to all other vertices. In other words, a clique is a subnetwork with maximum density.

A **clique** is a maximal complete subnetwork containing three vertices or more.

The size of a clique is the number of vertices in it. Maximal complete subnetworks of size one and two - cliques with one or two vertices - exist, but they are not very interesting because they are single vertices and edges or bi-directional arcs respectively. Therefore, cliques must contain a minimum of three vertices.

Unfortunately, cliques are very difficult to identify in large networks: the computational method is very time-consuming and even medium-sized networks may contain an enormous number of cliques. In this book, therefore, we will restrict ourselves to the analysis of small complete subnetworks, which may or may not be cliques. We will concentrate on complete triads, that is, complete subnetworks consisting of three vertices, but the argument is easily extended to complete subnetworks of size four or more.

Figure 6 shows the complete undirected and directed triad as well as an example of a network which contains several complete triads. Note that the complete triad with vertices v1, v5, and v6 is a clique because we cannot add another vertex from the network to this subnetwork such that it is still complete. This subnetwork is maximal with respect to completeness. In contrast, triad v2, v4, v5 is not a clique because we can add vertex v3 and the subnetwork is still complete. Vertices v2 to v5 constitute a clique of size four, which, by definition is made up of four complete triads.

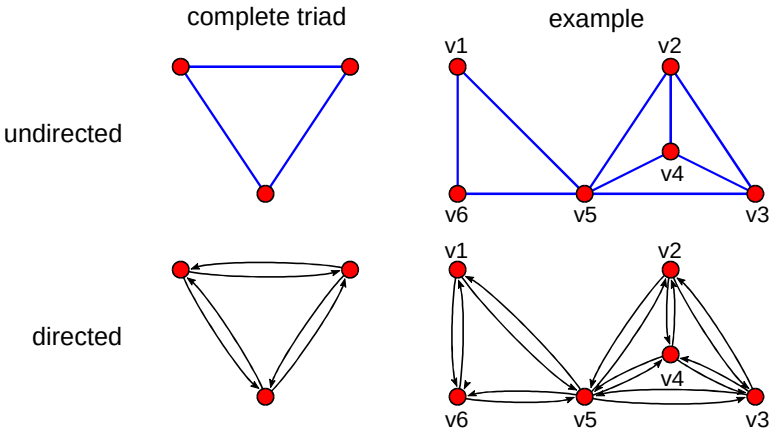


Figure 6 - The complete triad and an example.

Figure 6 shows a very important feature of cliques and complete subnetworks, namely that they can overlap. The complete triad v1, v5, v6 overlaps with the

complete triad v2, v4, v5 because they share vertex v5. As a consequence, it is impossible to assign all vertices unambiguously to one clique or complete subnetwork. We can not equate each clique or complete subnetwork with a cohesive subgroup and this is a serious complication if we want to classify vertices into cohesive subgroups.

In social network analysis, **structures of overlapping cliques**, which are thought to represent social circles, rather than individual cliques are regarded as cohesive subgroups. Cliques or complete triads are the densest sections or 'bones' of a network, so the structure of overlapping cliques can be considered its 'skeleton.' Sometimes, additional conditions are imposed on the overlap of cliques, e.g., a minimum number or percentage of vertices that two cliques must share, but we will not use them here.

Application

Because clique detection is useful for dense networks in particular, we will now analyze the network with all types of visiting relations in San Juan Sur (`SanJuanSur.net`). Visits among kin are represented by red arcs (the line value is two), visits among families bound by god-parent or god-child ties ('church relations') are blue (the line value is three), and other visits are black arcs (the line value is one). Note that the arcs are only colored if the option *Options>Colors>Arcs>As Defined on Input File* is selected in the Draw screen. We symmetrize the relations to obtain a rather dense network (average degree is 4.13).

Nets>First Network, Second Network

Nets>Fragment (1 in 2) >Find

This network is too dense to spot complete triads right away. Even the best energized drawing contains many crossing edges, which makes it difficult to see complete triads; probably, there are many. However, we can leave the detection of complete triads to Pajek, which is able to search for particular structures, which are called fragments, within a network. Open (`triad_undir.net`) or create a network (command *Net>Random Network*, see Chapter 1, Section 1.4) which represents the structure or fragment that you want to search: a complete undirected triad in our example. Select this network as the first network in the *Nets* menu and select the symmetrized visiting relations network as the second network. Now, you can find all complete triads in the network by executing the *Find* command of the *Fragment (1 in 2)* command.

Nets>Fragment (1 in 2) >Options

Executing this command, Pajek creates one or more new data objects, depending on the options selected in the *Options* submenu of the *Fragment* command. We recommend to check the option *Extract subnetwork* only. This will produce a network labeled 'Subnetwork induced by Ind fragments like 2 in 1 (62)'. It is called *induced* because Pajek selects vertices and lines within the fragments (complete triads) only. This network contains the overlapping cliques which we are looking for and we will discuss it at the end of this section. In addition, Pajek creates a hierarchy and a partition. The partition counts the number of fragments to which each vertex belongs and the hierarchy lists all fragments: complete triads in our example.

A hierarchy is a data object which we have not yet encountered. It is designed to classify vertices if a vertex may belong to several classes. In the visiting relations network, for instance, a family may belong to several complete triads. A hierarchy is a list of groups and each group may consist of groups or vertices. Ultimately, vertices are the units which are grouped. Figure 7 shows the hierarchy for the example of overlapping complete triads of Figure 6. There are five complete triads, each of them is represented by a green vertex in Figure 7. Each complete triad consists of three vertices. Note that most vertices appear more than once because the triads overlap. At the top of the hierarchy, one node (yellow) connects all groups; it is called the root of the hierarchy.

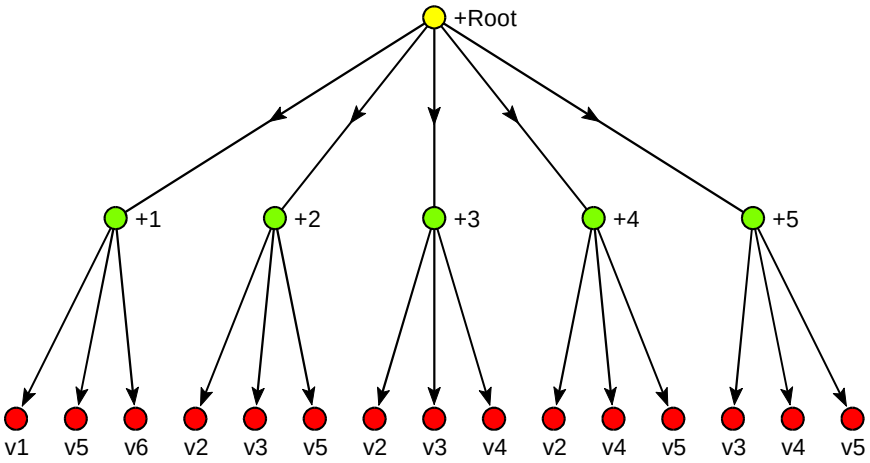


Figure 7 - A hierarchy of cliques.

Hierarchy drop list

File>Hierarchy>Edit

A hierarchy is stored as a data object in the Hierarchy drop list. You can browse a hierarchy in an Edit screen, which is opened with the *Edit* command in the *File>Hierarchy* submenu or by the Edit button to the left of the Hierarchy drop list. On opening, the Edit screen displays the root only. Click on the plus sign preceding the root to display the (first level of) groups in the hierarchy.

Figure 8 shows part of the 52 cliques of size three in the visiting relations network. Select a group with your left mouse button and click with the right mouse button to display its vertices in a separate window. If the original network is selected in the Network drop list, vertex labels are displayed next to their numbers in this window.

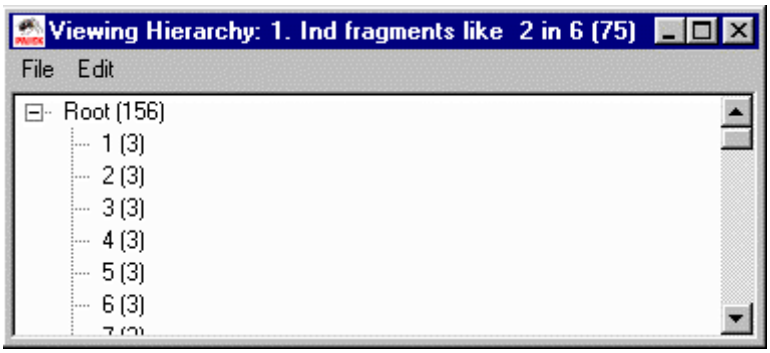


Figure 8 - Viewing a hierarchy in an Edit screen.

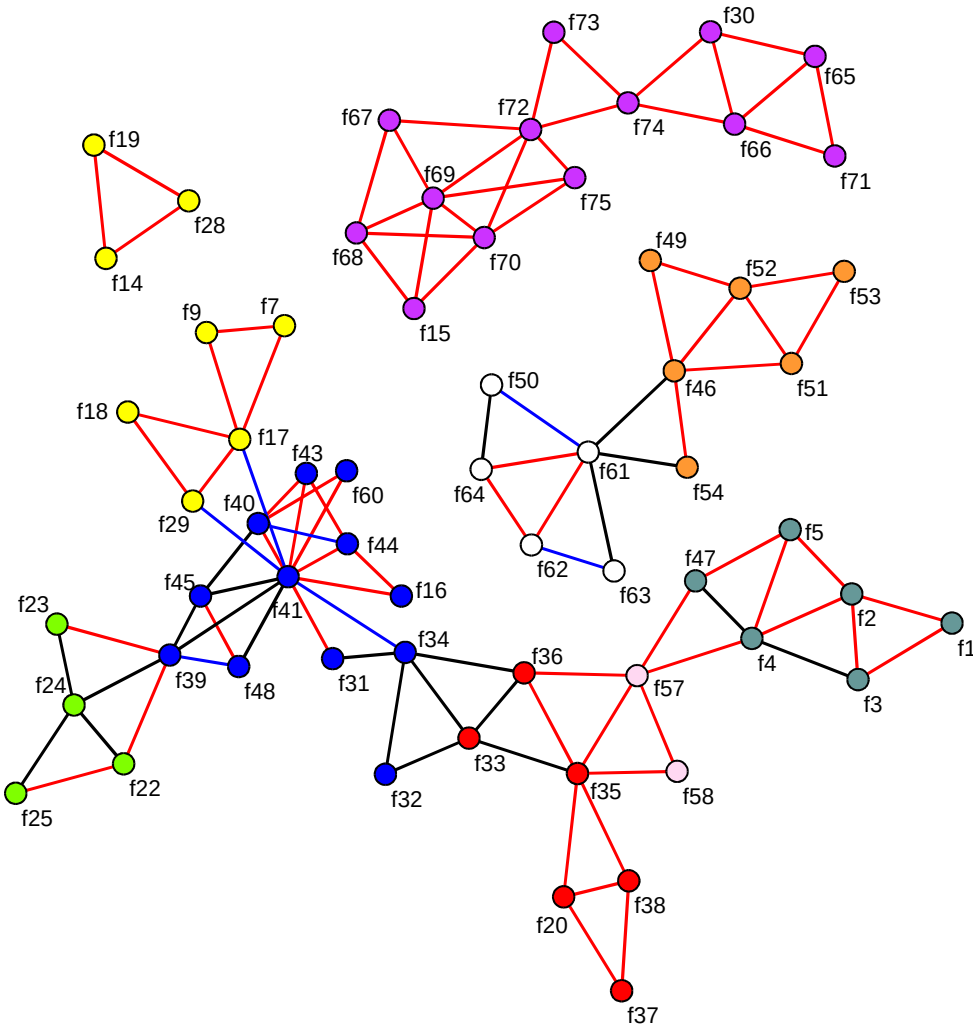


Figure 9 - Complete triads and family-friendship groupings (colors).

Partitions>Extract Second from First

The induced subnetwork is displayed in Figure 9. It contains 62 of the 75 families: thirteen families do not belong to a complete triad. Vertex colors indicate the family-friendship groupings. We could not draw this colored sociogram directly, because the number of families in the subgraph of complete triads (62) does not match the number of families in the family-friendship groupings partition (75). To obtain a new partition with family-friendship classes for the 62 families in the complete triads network, we extracted the families in complete triads (nonzero classes in the partition which was made by the *Fragments* command) from the partition with family-friendship groupings of all families. The command which we used (*Partitions>Extract Second from First*) is discussed in Chapter 2, Section 2.4.

Figure 9 shows three components of overlapping complete and one isolated triad, so we may say that we have found three social circles under the criterion of complete triads which share at least one member. Family groupings are nicely arranged in and over the components.

In a directed network, you may follow the same procedure but you have to use a complete directed triad as a fragment. Probably, you will find less directed cliques than undirected cliques. Note that the directed network of visiting

relations contains no more than five cliques of three vertices which are connected by mutual visits. Two cliques overlap, so we can hardly speak of social circles in the directed network.

3.7 Summary

In this chapter, social cohesion is linked to the structural concepts of density and connectedness. Density refers to the number of links between vertices. A network is strongly connected if it contains paths between all of its vertices and weakly connected when all of its vertices are connected by semipaths. Connected networks and networks with high average degree are thought to be more cohesive. This applies also to sections of a network (subnetworks). We expect local concentrations of ties in a social network which identify cohesive social groups.

There are several techniques to detect cohesive subgroups based on density and connectedness, three of which are presented in this chapter: components, *k*-cores, and cliques or complete subnetworks. All three techniques assume relatively dense patterns of connections within subgroups, but they differ in the minimal density required, which varies from at least one connection (weak components) to all possible connections (cliques). Two more techniques based on a similar principle (*m*-slices and bi-components) will be presented in later chapters.

Components identify cohesive subgroups in a straightforward manner: each vertex belongs to exactly one component. The link between cohesive subgroups and *k*-cores or cliques is more complicated. *k*-cores are nested, which means that higher *k*-cores are always contained in lower *k*-cores, so a vertex may belong to several *k*-cores simultaneously. In addition, *k*-cores are not necessarily connected: the vertices within one *k*-core can be spread over several components. In order to identify cohesive subgroups, the researcher has to eliminate vertices of low *k*-cores until the network breaks up into relatively dense components. Cliques or complete subnetworks, such as complete triads, may overlap, that is, share one or more vertices, so we regard a component of overlapping cliques as a cohesive subgroup rather than each clique on its own.

Because the techniques to detect cohesive subgroups are based on the same principle, substantive arguments to prefer one technique over another are usually not available. The choice of a technique depends primarily on the density of the network. In a dense network, the structure of overlapping cliques reveals the cohesive ‘skeleton’ best, whereas components and *k*-cores unravel loosely knit networks better. In exploratory research, we recommend to look for components first, then apply *k*-cores, and search for complete triads to subdivide large *k*-cores if necessary (see the decision tree in Figure 10).

Another choice pertains to the treatment of directed relations. In general, symmetrizing directed relations yields higher density, thus more or larger cohesive subgroups. For *k*-cores, we recommend to use simple undirected or symmetrized networks to make sure that *k* equals the number of neighbors to

whom each vertex is connected in a core. In a directed network, components may be weak or strong. Strong components and complete directed triads are based on reciprocal ties whereas weak subgroups consider unilateral ties as well.

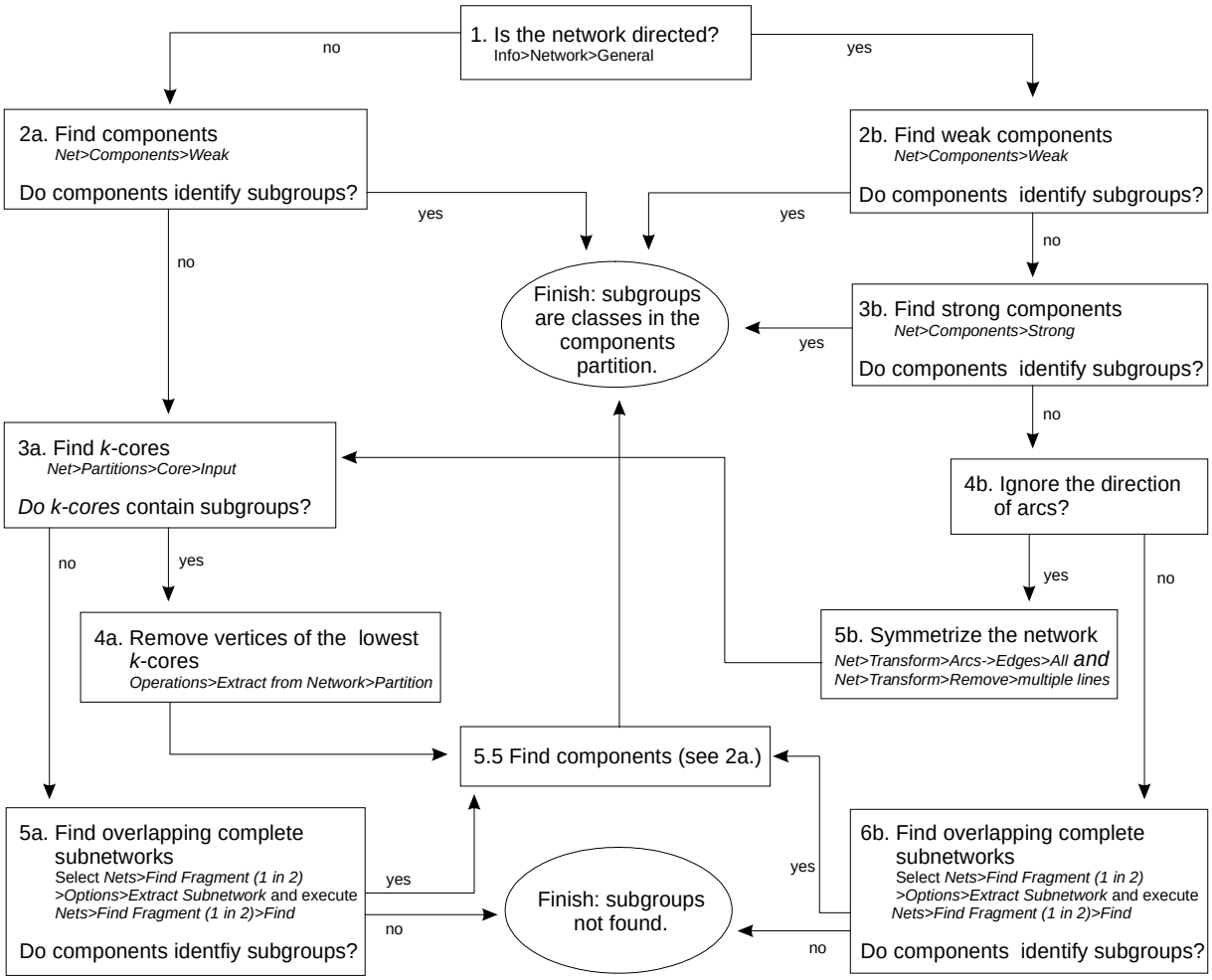
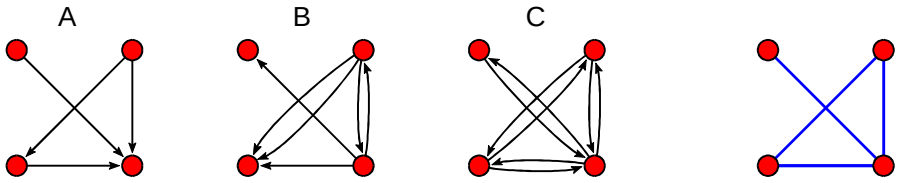


Figure 10 - Decision tree for the analysis of cohesive subgroups.

In this chapter, we use the word *subgroup* but a cohesive subgroup is not necessarily a social group. We need to check this by comparing the structural subgroups with respect to the characteristics, behavior, and opinions of their members. Sometimes, our prior knowledge about the entities in the network enables us to make sense of the cohesive subgroups we detect. Otherwise, we must systematically compare the partition which identifies cohesive subgroups with partitions representing social attributes of the vertices.

3.8 Exercises

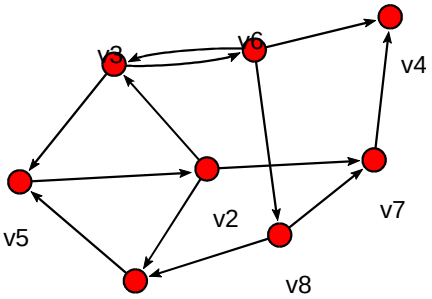
1 Inspect the networks which are depicted below. If we symmetrize the directed networks, which ones become identical to the undirected network?



a A, B, and C
b A and B only
c A and C only
d A only

2 Which of the following statements is correct?
a Density is the degree sum of all vertices in a simple undirected network.
b Density is the degree sum of all vertices divided by the number of vertices in a simple undirected network.
c Density is the number of arcs divided by the number of pairs of vertices in a simple directed graph.
d Density is the number of edges divided by the number of pairs of vertices in a simple undirected graph.

3 Which of the following statements about the network below are correct?



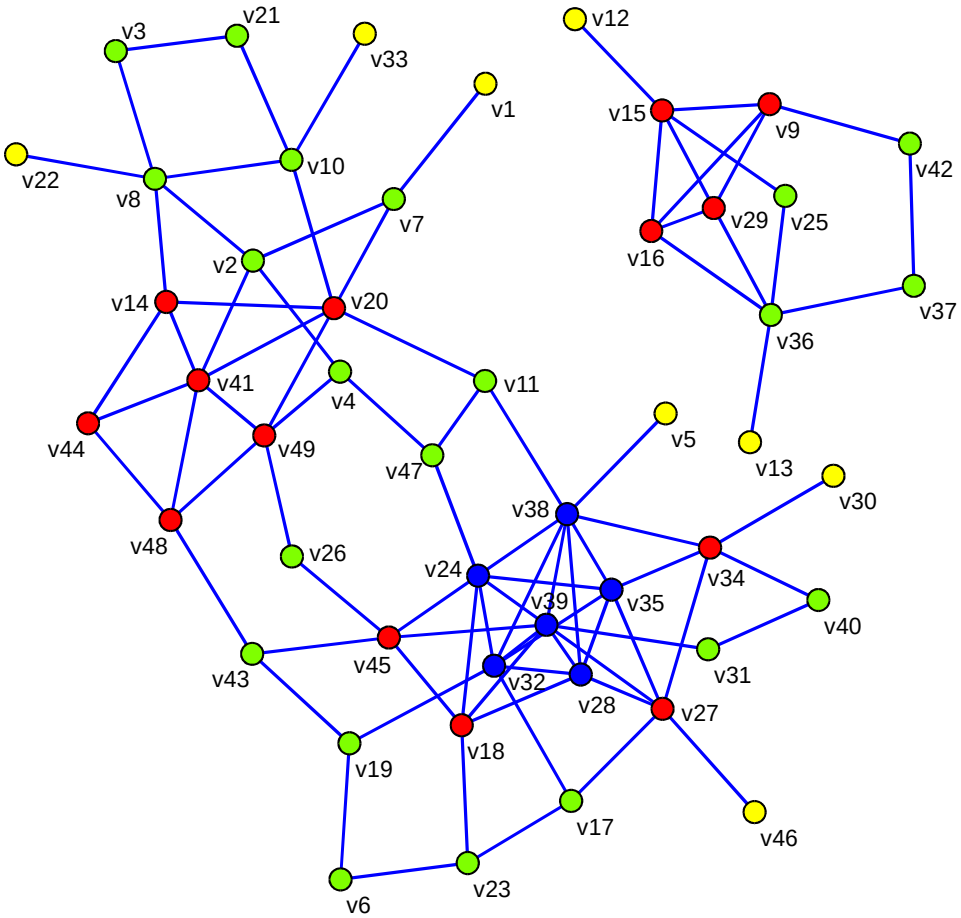
a v6 and v4 are not connected by a semipath
b v1 and v4 are not connected by a path
c there is no path from v7 to v3
d there is no path from v1 to v8

4 How many strong components does the network of Exercise 3 contain?
a no strong components
b 1 strong component
c 2 strong components
d 3 strong components

5 Which of the following statements is correct?
   a A subnetwork is maximal if it cannot be enlarged without loosing its structural characteristic.
   b A subnetwork is not maximal if it does not cover the whole network.
   c A subnetwork is maximal if it is connected.
   d A subnetwork is maximal if it is complete.

6 Which of the following statements is correct for simple undirected networks?
   a in a 1-core, each vertex has exactly one neighbor
   b a component is always a 1-core
   c each 1-core is a clique
   d each 3-core is a clique

7 Count the number of 3-cores in the network below. Vertex colors indicate the level of *k* (yellow: *k* = 1; green: *k* = 2; red: *k* = 3; blue: *k* = 4).
   a one 3-core
   b two 3-cores
   c three 3-cores
   d four 3-cores



3.9 Assignment

The researchers assigned the families of San Juan Sur to family-friendship groupings on the basis of their answers to the question: In case of a death in the family, whom would you notify first? Their choices are stored in the file `SanJuanSur_deathmessage.net`. In this file, the coordinates of families correspond with the locations of families in the original sociogram drawn by the researchers. Partition `SanJuanSur_deathmessage.clu` contains the family-friendship groupings for this network.

We would like to reconstruct the way the families were assigned to family-friendship groupings. Find out which type of cohesive subgroups match the family-friendship groupings best and use the indices of statistical association presented in Chapter 2, Section 2.6 to assess how well they match. Do you think that the researchers used additional information to assign families to family-friendship groupings?

3.10 Further reading

- The example is taken from Charles P. Loomis, Julio O. Morales, Roy A. Clifford & Olen E. Leonard, *Turrialba. Social Systems and the Introduction of Change* (Glencoe (Ill.): The Free Press, 1953). We will also use these data in Chapter 9 on prestige.
- Chapter 6 in John Scott, *Social network analysis : A handbook* (London: Sage Publications, 1991 or 2nd ed. in 2000) offers an overview with some additional types of connected subnetworks. Chapter 7 in Stanley Wasserman and Katherine Faust, *Social Network Analysis: Methods and Applications* (Cambridge: Cambridge University Press, 1994) is even more detailed.

3.11 Answers

1 Answer c is correct. Symmetrizing a network means that unilateral and bi-directional arcs are replaced by an edge. Multiple arcs, e.g., from the top-right vertex to the bottom-left vertex in network B, are replaced by multiple edges. Therefore, the undirected network is not a symmetrized version of network B.

2 Answer d is correct. Density is defined as the number of lines in a network, expressed as a proportion of the maximum possible number of lines. In a simple undirected network, a pair of vertices can be connected by one edge because multiple lines do not occur. In addition, loops do not occur. Therefore, the number of pairs of vertices equals the maximum possible number of lines and answer d is correct. Recall that an edge is defined as an unordered pair of vertices and an arc as an ordered pair. In a simple directed network, each pair of vertices may be connected by two arcs, so the maximum possible number of lines is twice the number of pairs. Besides, a

simple directed network may also contain loops. Thus, answer c is incorrect. Answers a and b refer to the sum of degrees and average degree, which are other kinds of indices.

3 Answer c is correct because the only path which originates in v7 leads to v4 where it stops because v4 sends no arcs. Answer a is incorrect because there are several semipaths between v6 and v4, e.g., v6→v3←v2→v7→v4, and each path (v6→v4) is also a semipath. Answer b is incorrect because there is a path from v1 to v4, e.g., v1→v5→v2→v7→v4. Answer d is incorrect because there is a path from v1 to v8: v1→v5→v2→v3→v6→v8

4 Answer d is correct. Vertex v4 is a strong component on its own because it has no path to any other vertex. Vertex v7 is the start of one path, which ends at v4. Since there is no path in the opposite dierction, v7 is not a member of a larger strong component. The third strong component consists of the remaining six vertices, which are connected by paths in both ways.

5 Statement a is correct, e.g., a component is a maximal connected subnetwork because no vertex can be added in such a way that the subnetwork is still connected. Connectedness in itself is not enough for a subnetwork to be maximal (statement c), nor is completeness (statement d). A maximal subnetwork does not need to include all vertices in the network (statement b is incorrect).

6 Answer b is correct for every simple undirected network. In a component, vertices must be linked to at least one other vertex for the component to be connected, so each component is at least a 1-core. A star network is a 1-core because all vertices except the central vertex has just one neighbor. Nevertheless, the central vertex has more than one neighbor, so answer a is incorrect. *k*-cores are not necessarily cliques, so answers c and d are not correct.

7 Answer a is correct. A *k*-core is not necessarily connected, so all unconnected parts of the 3-core still belong to one 3-core.