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 or assortativity: Birds of a feather flock together. We will learn how to measure this phenomenon in Chapter 6.

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 to 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 an 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 *haciendas* (farms) in a neighborhood called Attiro. The network of visiting ties (`Attiro.net`, drawn in Figure 31)

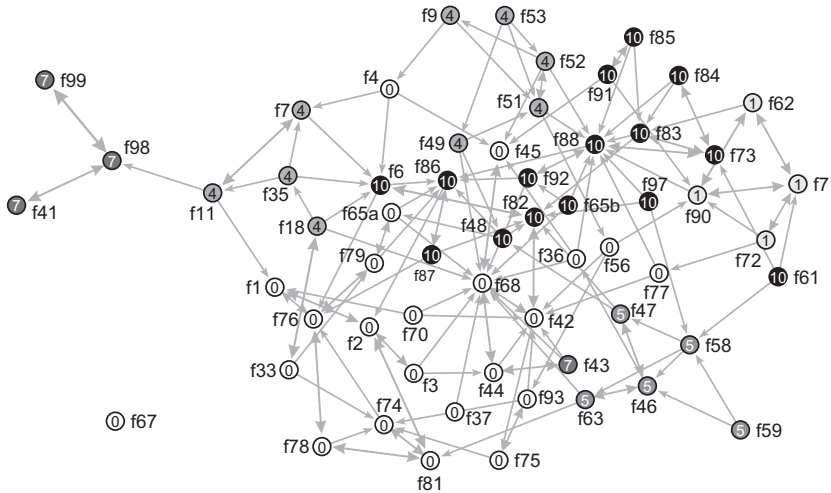


Figure 31. Visiting ties in Attiro.

is a simple directed graph: Each arc represents "frequent visits" from one family to another. The exact number of visits was not recorded. Line values classify the visiting relation as ordinary (value 1), visits among kin (value 2), and visits among ritual kin (i.e., between godparent and godchild); but we do not use them in this chapter. Loops do not occur because they are meaningless.

We compare cohesive subgroups in this network with an ethnographic classification of the families into six family–friendship groupings that were made by the researchers on substantive criteria (`Attiro_grouping.clu`; we adjusted the class numbers to get optimal grays with the *Options*> *Colors*> *Partition Colors*> *for Vertices*> *Default Greyscale 2* option). 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 `Attiro.paj`. Open this file now, and draw the network with its partition to obtain a sociogram like Figure 31 (*Draw*> *Network + First Partition* command). You may want to use real colors instead of grays for easy recognition of the family–friendship groupings.

Which cohesive subgroups can we find in the Attiro network, and do they match the family–friendship groupings? Figure 31 offers a visual impression of the kin visits network and the family–friendship groupings, which are identified by the colors and numbers within the vertices. As is shown, the network is tightly knit with family–friendship groupings 0 and 10 dominating the center. Most families that belong to one grouping are connected by visiting ties, so they are rather close in the

network. Exceptions occur, however; notably family f43 (bottom right), which is separated from the other vertices in the seventh family–friendship grouping (left). In subsequent sections, we set out this first impression in detail.

3.3 Density and Degree

Intuitively, cohesion means that a social network contains many ties. More ties between people yield a tighter structure that is, presumably, more cohesive. In network analysis, the density of a network captures this idea. It is the percentage of all possible lines that 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 ties. Although density measures have been proposed that 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 discuss other measures that capture the contribution of multiple lines and line values to cohesion in Chapter 5.

In the kin visiting relation network, density is 0.045, which means that only 4.5 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 ties that each person can maintain is limited. In a visiting relation 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 San Juan Sur, which is another neighborhood in the Turrialba region, is 0.036. This is slightly lower than in Attiro, but the difference may be due to a larger number of families in San

Juan Sur (seventy-five families). Therefore, we cannot 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 ties 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 31, family f88 (a member of family–friendship grouping number 10) is connected to thirteen families by fifteen visiting ties (note that the double-sided arcs between f88 and f73, f92 indicate that these families are linked by mutual visits), so its degree is 15. 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, because vertices entertain more ties. 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. For example, the average degree of the Attiro network is 5.37, which is slightly higher than the average degree of the San Juan Sur network (5.28).

> 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 that are adjacent to this vertex: its *neighbors*. Each line that is incident with the vertex connects it to another vertex because 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 sent (its outdegree). Note that the sum of the indegree and the outdegree of a vertex does not necessarily equal the number of its neighbors; for instance, family f88 is involved in fifteen visiting ties, but it has thirteen adjacent families because families f73 and f92 are counted twice (Figure 31).

In this section, we restrict ourselves to degree in undirected networks. When we encounter a directed network, we symmetrize it, which means that we turn unilateral and bidirectional arcs into edges. A discussion of indegree in directed networks occurs in Chapter 9, which presents the concept of prestige.

> To *symmetrize* a directed network is to replace unilateral and bidirectional arcs with edges.

Application

[Main] Network> Info> General

Let us analyze the network of visiting ties in Attiro (`Attiro.net`), which contains neither multiple lines nor loops. In Pajek, the density of a network can be obtained by means of the *Info* submenu of the *Network* menu in the Main screen. 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. You can also press the "*I*" (*I* stands for "Info") button at the left of the Network drop-down menu. 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 and average degree, request zero lines. Pajek computes two density indices in the Report screen. The first index allows for loops, and the second does not. Because loops are meaningless in a visiting relation network – people do not visit themselves – the second index is valid. Density in the directed network is 0.045. Finally, the average degree is reported, which is 5.37 for Attiro.

Network> Create New Network> Transform> Arcs→Edges> All

File> Network> Save

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 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 *Network> Create New Network> Transform* submenu to replace all arcs with edges. Pajek will ask whether you want to make a new network, and we advise you to do so because you may want to use the directed network later. Next, Pajek asks whether you want to remove multiple lines. To obtain a simple undirected network, that is, a network without multiple lines and loops, you should choose option 1 (sum the line values of lines that will be joined into a new line), 2 (count the number of lines that are joined), 3 (preserve the minimum value of joined lines), 4 (take their maximum line value), or 5 (the value of the new line will be 1) in this dialog box. It does not matter which of these five options you choose because in this chapter we pay no attention to line values. Now, the network is symmetrized, and it is simple because multiple lines are removed and there were no loops in the original network.

Table 5. *Frequency distribution of degree in the symmetrized network of visits*

Clusters	Freq	Freq%	CumFreq	CumFreq%	Representative
0	1	1.6667	1	1.6667	f67
1	3	5.0000	4	6.6667	f37
2	1	1.6667	5	8.3333	f59
3	19	31.6667	24	40.0000	f3
4	20	33.3333	44	73.3333	f1
5	4	6.6667	48	80.0000	f45
6	6	10.0000	54	90.0000	f51
7	2	2.3333	56	93.3333	f6
8	1	1.6667	57	95.0000	f86
9	1	1.6667	58	96.6667	f42
13	1	1.6667	59	98.3333	f88
14	1	1.6667	60	100.0000	f68
SUM	60	100.0000			

You may want to save it (*File> Network> Save*) for future use under a new name (e.g., `Attiro_symmetrized.net`).

Network> Create Partition> 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 *Network> Create Partition> Degree* submenu: *Input*, *Output*, or *All*. *Input* counts all incoming lines (indegree), *Output* counts all outgoing lines (outdegree), and *All* counts both. Note that an edge, which has no direction, is considered to be incoming as well as outgoing, so each edge is counted once by all three degree commands. In an undirected network, therefore, it makes no difference whether you select *Input*, *Output*, or *All*.

Partition> Info

The command *Partition> Info* displays the partition as a frequency table (see Table 5). Class numbers represent degree scores, so we can see that the degree of vertices varies markedly from zero to fourteen neighbors in the symmetrized network. Clearly, family f68 is connected to most families by visiting ties. One family, family f67, is isolated in the network: It is linked to no other families by regular visits.

Network> Info> General

The average degree of all vertices 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 *Partition> Info* command. To obtain the average degree of the symmetrized network, we can use the *Network> Info> General* command again, which will report an average degree of 4.27. Families in Attiro have regularly visiting relations with more than four families on average. Note that this average degree differs from the average degree reported for the original directed network (5.37) because the latter sums indegree (visits received) and outdegree (visits paid), which may count families twice.

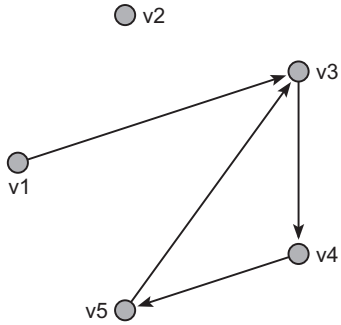


Figure 32. A simple unconnected directed network.

Exercise I
Open the visiting relation network in San Juan Sur (`SanJuanSur.net`), symmetrize it, and determine the average degree. Is this network more cohesive than the Attiro network in this respect?

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 in Attiro is not entirely connected (see Figure 31). In this section, we 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 32). Intuitively, it is clear that some vertices are connected to other vertices, whereas others are not; for instance, vertex v2 is adjacent to no other vertex, but the other four vertices have one or more 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.

> 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 are 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. In a walk, you have to follow the direction of the arcs.

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 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 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, provided that there are any. The example of Figure 32 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 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 that are connected; for instance, vertices v1, v3, v4, and v5 are connected. In comparison with the isolated vertex v2, these vertices are relatively tightly connected, so we may say that they are a cohesive group. If the relation represents 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 that 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 vertices v1 and v2. Vertex v2 is isolated and there are paths only from vertex v1 but no paths to v1, so vertex v1 is not strongly connected to any other vertex. It is asymmetrically linked to the larger strong component. In general, the ties among strong components are either asymmetrical or absent. In Chapter 10, we 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 a relation: Does it matter to social processes whether actor A turns to

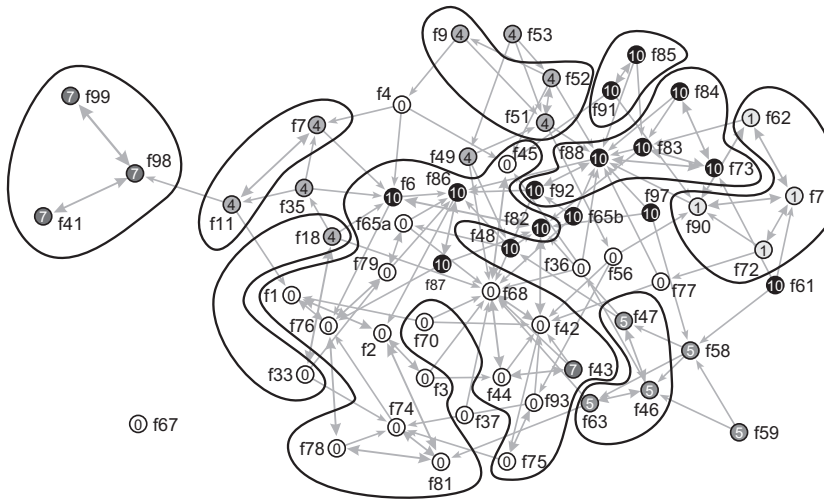


Figure 33. Strong components (contours) and family–friendship groupings (vertex colors and numbers) in the network of Attiro.

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 f98 visits both families f11 and f99 (Figure 33, left), it may inform family f11 about family f99 and the other way around. Families f11 and f99 may share information although there is no path between them. In this case, direction of the relation is quite unimportant and weak components are preferred.

If substantive arguments are indecisive, the number and size of components may be used to choose between strong and weak components. Recall that strong components are stricter 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 network in Attiro), we advise using strong components to break down the weak component in a next step.

Figure 33 shows the strong components in the visiting relation network. Each strong component of more than one vertex is manually delineated by a contour. Each vertex outside the contours is a strong component on its own (e.g., families f67 and f59). The original classification according to family–friendship groupings is represented by vertex colors and by the numbers inside the vertices. We see that the large weak component is split up into several small strong components, some of which approximate family–friendship groupings, for instance, family–friendship groupings 1 (at the right) and 7 (at the left).

Components can be split up further into denser parts by considering the number of distinct, that is noncrossing, paths or semipaths that connect the vertices. Within a weak component, one semipath between each pair of vertices suffices, but there must be at least two different semipaths in a *bi-component*. The concept of a bi-component is discussed in Chapter 7. This can be generalized to k-connected components: maximal subnetworks in which each pair of vertices is connected by at least k distinct semipaths or paths. A weak component, for instance, is a 1-connected component and a bi-component is a 2-connected component.

Application

With Pajek, it is easy to find components in the visiting relation network (`Attiro.net`). The *Network* menu has a submenu to find three types of components: strong, weak, and strong-periodic. Strong-periodic components will not be discussed here. 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. The command creates a partition in which each class represents a component. Draw the network with the strong components partition (*Draw> Network + First Partition*) to see the clusters enclosed in contours in Figure 33. Draw it with the original family–friendship groupings partition to obtain the clusters represented by vertex color in Figure 33. Figure 33 combines these two layouts.

Network> Create Partition> Components> Strong

Draw> Network + First Partition

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.

Network> Create Partition> Components> Weak

3.5 Cores

The distribution of degree reveals local concentrations of ties around individual vertices, but it does not tell us whether vertices with a high degree are clustered or scattered all over the network. In this section, we use degree to identify clusters of vertices that are tightly connected because each vertex has a particular minimum degree within the cluster. We pay no 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 that are connected by degree 2 or more to other vertices within

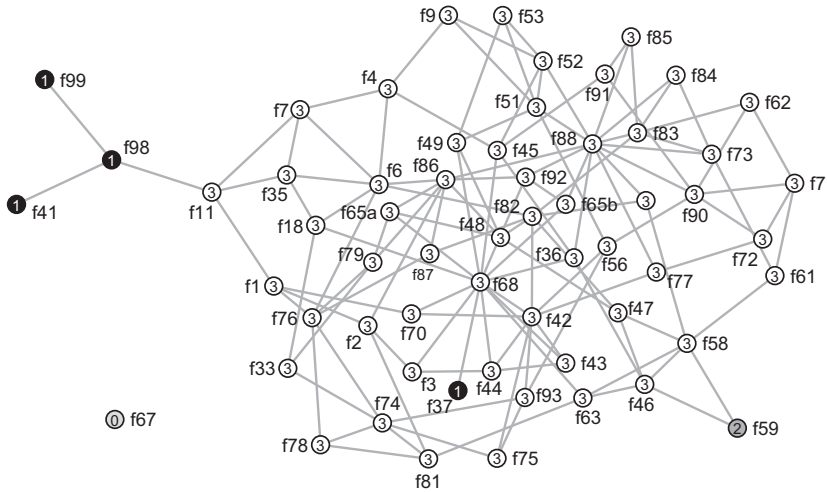


Figure 34. *k*-cores in the visiting network at Attiro.

the core. A *k*-core identifies relatively dense subnetworks, so they help to find cohesive subgroups. As is shown, 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. It is easiest to explain if we apply it to a simple undirected network and, as a rule, we apply it only to this type of network. In a simple undirected network, the degree of a vertex is equal to the number of its neighbors, as discussed in Section 3.3, so a *k*-core contains the vertices that have at least *k* neighbors within the core. A 2-core, then, consists of all vertices that 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 that satisfy the required property, in this case a minimum number of *k* neighbors within the core.

The undirected visiting relation network, which we obtained by symmetrizing the directed network, contains a large 3-core (white vertices in Figure 34). In the 3-core, each family is connected to at least three other families. In addition, there is a 2-core (dark gray), a 1-core (black), and a 0-core (light gray). 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. The 2-core and the 0-core, however, consist of one vertex (families f59 and f67), and the 1-core is situated at

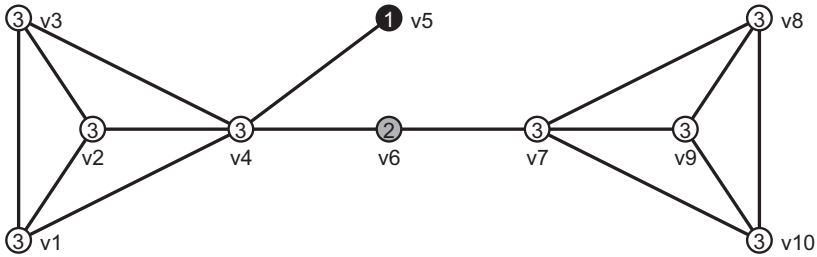


Figure 35. *k*-cores.

two different places in the network (at the left and at the bottom). It is silly to regard them as cohesive subgroups.

The meaning of the lower *k*-cores can be illustrated by the simple example in Figure 35. This little network is connected, so all ten vertices are linked to at least one other vertex. As a result, all vertices belong to the 1-core, which is drawn in black at the bottom of Figure 36. One vertex, v5, has only one neighbor, so it is not part of the 2-core (gray, in the middle of Figure 36). Vertex v6 has a degree of 2, so it does not belong to the 3-core (white, in the top of Figure 36). Other vertices belong to the highest *k*-core, so the resulting sociogram looks like Figure 35: The different levels are stacked one on top of the other. We say 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.

The example illustrates another feature of *k*-cores, namely, that a *k*-core does not have to be connected. As a result of nesting, different cohesive subgroups within a *k*-core are usually connected by vertices that belong to lower cores. In Figure 36, vertex v6, 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 that identify the cohesive subgroups within the 3-core.

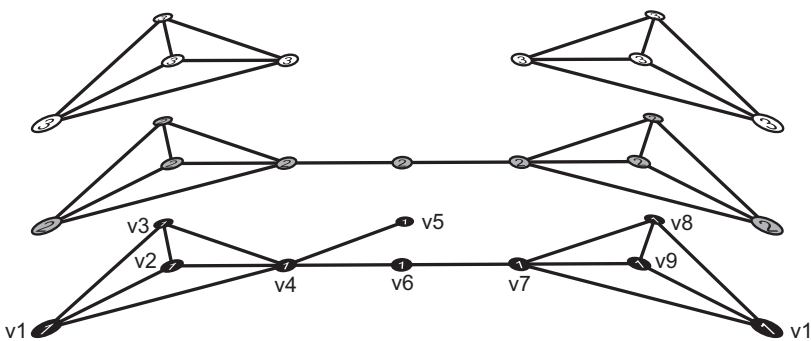


Figure 36. Stacking or nesting of *k*-cores.

This is exactly how *k*-cores help to detect cohesive subgroups: 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 it has at least *k* neighbors within the component. In (very) large networks, this is an effective way of finding cohesive subgroups. In the Attiro visiting relation network, however, this strategy does not work because there are no unconnected *k*-cores. Elimination of the lower *k*-cores does not split the network into separate components.

Application

Network> Create Partition> k-Core> Input, Output, All

In Pajek, *k*-cores are detected with the k-*Core* command in the *Network> Create Partition* submenu. The *Input*, *Output*, and *All* commands operate exactly in the same way as in the *Network> Create Partition> Degree* submenu, distinguishing among input cores, output cores, and cores that ignore the direction of lines. We advise using the *All* command and applying it only to simple undirected networks. The command yields a partition that assigns each vertex to the highest *k*-core in which it appears. Vertex colors and the numbers inside the vertices display the *k*-core partition in Figure 34. In this example, the *k*-cores do not match the ethnographic clustering into family–friendship groupings.

Operations> Network + Partition> Extract> SubNetwork Induced by Union of Selected Clusters

With the *k*-core partition, you can easily delete low *k*-cores from the network to distill the densest sections in the network. Select the *k*-core partition in the *Partition* drop list and execute the *Operations> Network + Partition> Extract> SubNetwork Induced by Union of Selected Clusters* command (see Section 2.4.1). Select the lowest and highest *k*-cores that you want to extract from the network, in this case the third *k*-core. Subsequently, use the *Network> Create Partition> Components> Strong* command to check whether the selected *k*-core levels are split into two or more components.

Network> Create Partition> Components> Strong

Exercise II

Determine the *k*-cores in the network `ExerciseII.net` and extract the 4-core from this network.

3.6 Cliques and Complete Subnetworks

In the visiting relation network, most vertices belong to one large 3-core. If we want to split this large 3-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.

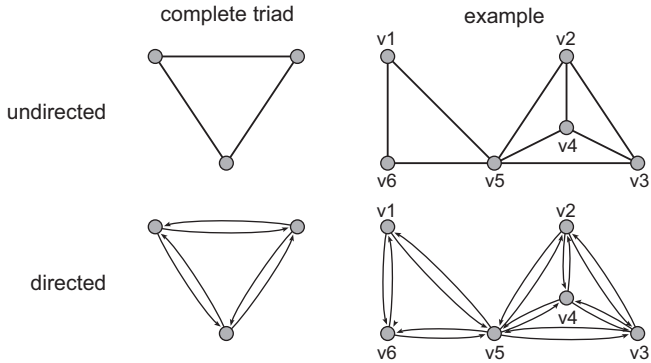


Figure 37. The complete triad and an example.

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 1 and 2 exist, but they are not very interesting because they are single vertices and edges or bidirectional arcs, respectively. Therefore, cliques must contain a minimum of three vertices.

Unfortunately, it is very difficult to identify cliques 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 restrict ourselves to the analysis of small complete subnetworks, which may or may not be cliques. We concentrate on complete *triads*, that is, complete subnetworks consisting of three vertices; but the argument is easily extended to complete subnetworks of size 4 or more.

Figure 37 shows the complete undirected and directed triad as well as an example of a network that 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 4, which, by the way, is made up of four complete triads.

Figure 37 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 cannot 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 are 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 do not use them here.

Application

Network> Create New Network> Transform> Arcs→Edges> All

Because clique detection is particularly useful for dense networks, we now analyze the symmetrized (undirected) network of visiting ties in Attiro, which has higher density (0.072) than the directed network (0.045). Symmetrize the network with the *Network> Create New Network> Transform> Arcs→Edges> All* command and avoid multiple lines by selecting option 1, 2, 3, 4, or 5 in the "Remove multiple lines?" dialog box. This network is too dense to spot complete triads and structures of overlapping triads visually. Even the best energized drawing contains many crossing edges, which makes it difficult to see complete triads; there are probably many.

Networks

The first step, then, is to detect all complete triads within the network. In other words, we have to find all occurrences of one particular network or fragment – in our case, a complete triad – in another network, namely, the original network. The command is situated in the *Networks* menu, which contains all operations on two networks, and it requires that the fragment and the original network are identified as the *First Network* and *Second Network*, respectively. The project file `Attiro.paj` contains the network `triad_undir.net`, which is a single complete undirected triad. Select this network in the first *Network* drop-down menu menu, and select the symmetrized visiting ties network in the second *Network* drop-down menu.

Networks> Fragment (First in Second)

Next, we can find all complete triads in the network by executing the *Find* command of the *Fragment (First in Second)* submenu. Executing this command, Pajek reports the number of fragments it has found, and it creates one or more new data objects, depending on the options selected in the *Options* window of the *Fragment* command. We recommend having the options *Extract subnetwork* and *Same vertices determine one fragment at most* checked only. The latter option ensures that unique instances of the fragment are counted, for example, that three vertices are counted only once as a complete triad. For large networks, this check may take a lot of time so the option can be deselected as well as its suboption *Create*

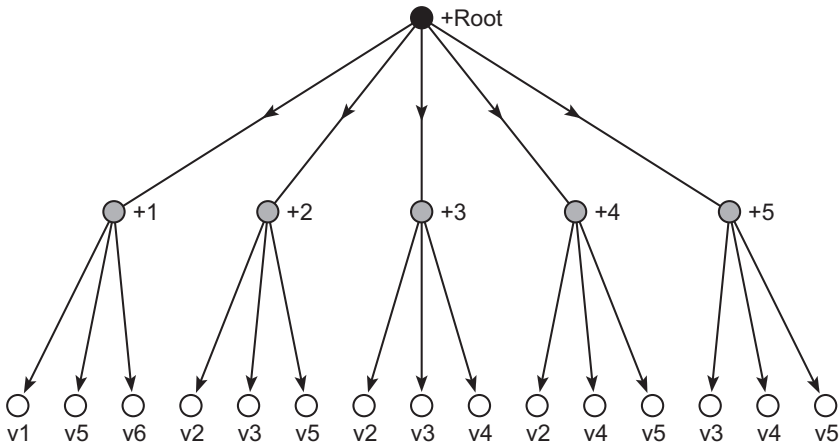


Figure 38. A hierarchy of cliques.

Hierarchy with Fragments. The search is quicker but there are multiple counts of the same fragment, depending on the structure of the fragment. For example, each undirected complete triad is counted six times, so one should divide the counts by this number. Note that the option *Same vertices determine one fragment at most* should usually not be set in searches of directed fragments because there can be different fragments for the same vertices, for example, several transitive triples in a complete directed triad.

This produces a new network labeled "Subnetwork induced by Sub fragments." It is called *induced* because Pajek selects vertices and lines within the fragments (complete triads) only. This network contains the overlapping cliques that we are looking for, and we 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 that we have not yet encountered. It is designed to classify vertices if a vertex may belong to several classes. In the visiting relation 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 that are grouped. Figure 38 shows the hierarchy for the example of overlapping complete triads of Figure 37. There are five complete triads; each of them is represented by a gray vertex in Figure 38. Each complete triad consists of three vertices (white in Figure 38). Note that most vertices appear more than once because the triads overlap. At the top of the hierarchy, one node (black) connects all groups; it is called the root of the hierarchy.

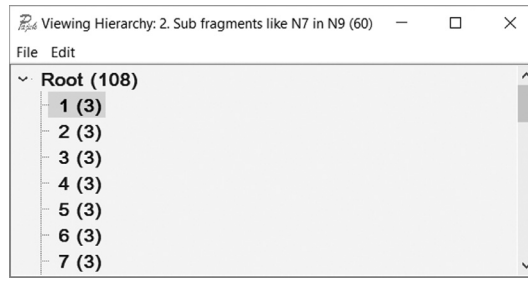


Figure 39. Viewing a hierarchy in an Edit screen.

Hierarchy drop-down menu

File> Hierarchy> View/Edit

A hierarchy is stored as a data object in the *Hierarchy* drop-down menu. You can browse a hierarchy in an Edit screen, which is opened with the *View/Edit* command in the *File> Hierarchy* submenu or by the View/Edit button to the left of the *Hierarchy* drop-down menu. 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 39 shows part of the thirty-six complete triads in the visiting relation network of Attiro. 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-down menu, vertex labels are displayed next to their numbers in this window. In this way, you can see which vertices belong to a complete triad.

Partition> Info

Now let us turn to the induced network and the partition created by the *Networks> Fragment (First in Second)> Find* command. The partition, labeled "Sub fragments," shows the number of triads that include a particular vertex. Using the *Partition> Info* command in the Main screen, you can see that two vertices belong to no fewer than seven complete triads, whereas thirteen vertices are not included in any of the complete triads. The latter vertices are not part of the structure of overlapping cliques, so they are eliminated from the induced network (labeled "Subnetwork induced by Sub fragments"), which contains the remaining forty-seven vertices of the Attiro network.

Partitions > Extract SubPartition (Second from First)

With this partition, we can make the original partition according to family–friendship groupings (in `Attiro_grouping.clu`) match the new induced network. Select the original partition as the first partition, and select the fragments partition as the second partition. Then execute the *Partitions> Extract SubPartition (Second from First)* command and specify 1 as the lowest class number and 7 (or higher) as the highest class number to be extracted. Pajek creates a new partition holding the family–friendship groupings of the forty-seven vertices in the induced network of overlapping complete triads. Draw this network and partition, and energize it with Kamada–Kawai to obtain a sociogram such as Figure 40 (use

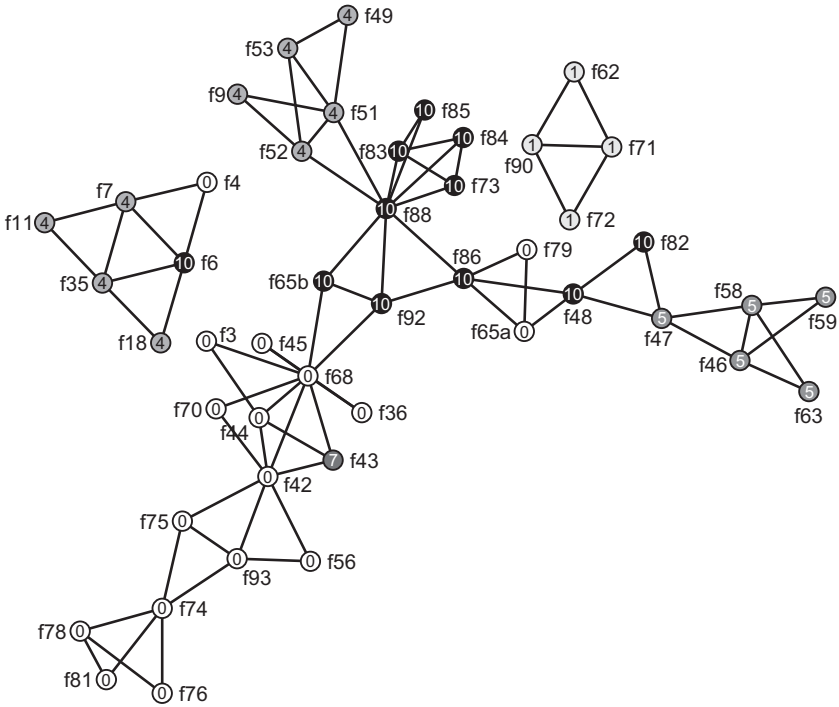


Figure 40. Complete triads and family–friendship groupings (colors and numbers inside vertices).

Default GreyScale 2 in the *[Draw] Options> Colors> Partition Colors> for Vertices* dialog screen to get the grays).

The induced subnetwork is displayed in Figure 40. It has three components of overlapping complete triads, so we say that we have found three social circles under the criterion of complete triads that share at least one member. Family–friendship grouping 1 is a separate social circle, but the other family–friendship groupings are interconnected although they are clearly clustered within the largest component. Family–friendship grouping 10 occupies a pivotal position in this structure, connecting groupings 0, 5, and part of family–friendship grouping 4.

In a directed network, you may follow the same procedure, but you have to use a complete directed triad as a fragment (e.g., `triad_dir.net`). In general, you will find fewer cliques in the directed network than undirected cliques in the symmetrized network. In the directed Attiro network, for instance, there is just one complete directed triad, containing families f62, f71, and f90, so we cannot speak of overlapping cliques in the directed network.

3.7 Summary

In this chapter, social cohesion was 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 it is 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 also applies to sections of a network (subnetworks). We expect local concentrations of ties in a social network to 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 (islands and bi-components) are presented in later chapters. There are many more formal concepts for cohesive subgroups, but all of them are based on the notions of density and connectedness.

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. 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 a component of overlapping cliques is regarded 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 looking for components first and then applying *k*-cores and searching for complete triads to subdivide large *k*-cores if necessary (see the decision tree in Figure 41).

Another choice pertains to the treatment of directed relations. In general, symmetrizing directed relations yields higher density, thus more

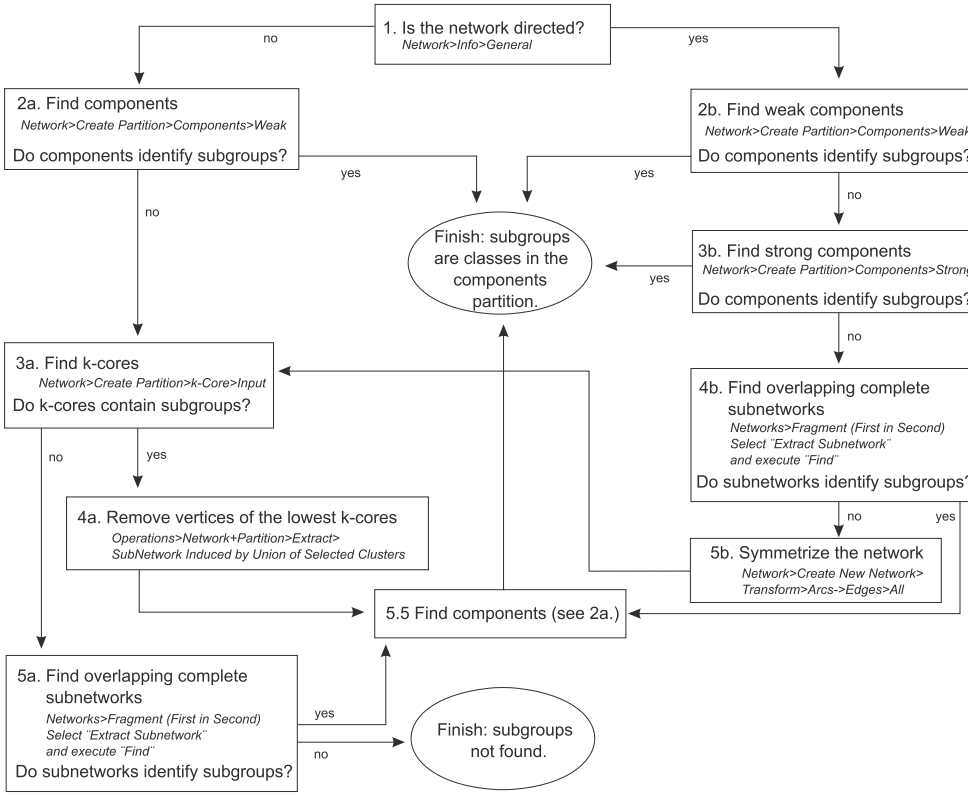


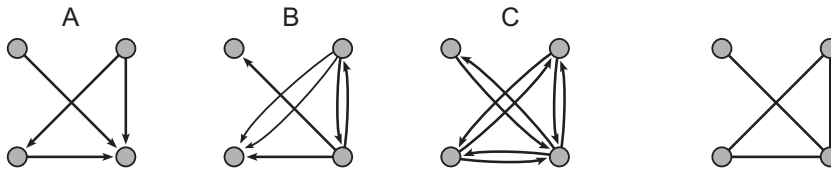
Figure 41. Decision tree for the analysis of cohesive subgroups.

or larger cohesive subgroups. For *k*-cores, we recommend using simple undirected or symmetrized networks to make sure that *k* equals the number of neighbors to which 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.

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 social 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 that identifies cohesive subgroups with partitions representing social attributes of the vertices.

3.8 Questions

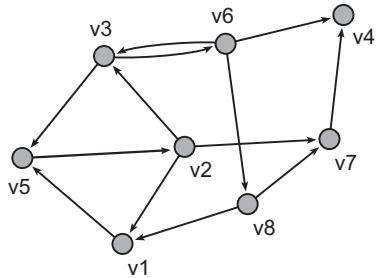
1. Inspect the networks that are depicted in the following figure. If we symmetrize the directed networks, which ones become identical to the undirected network?



   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?

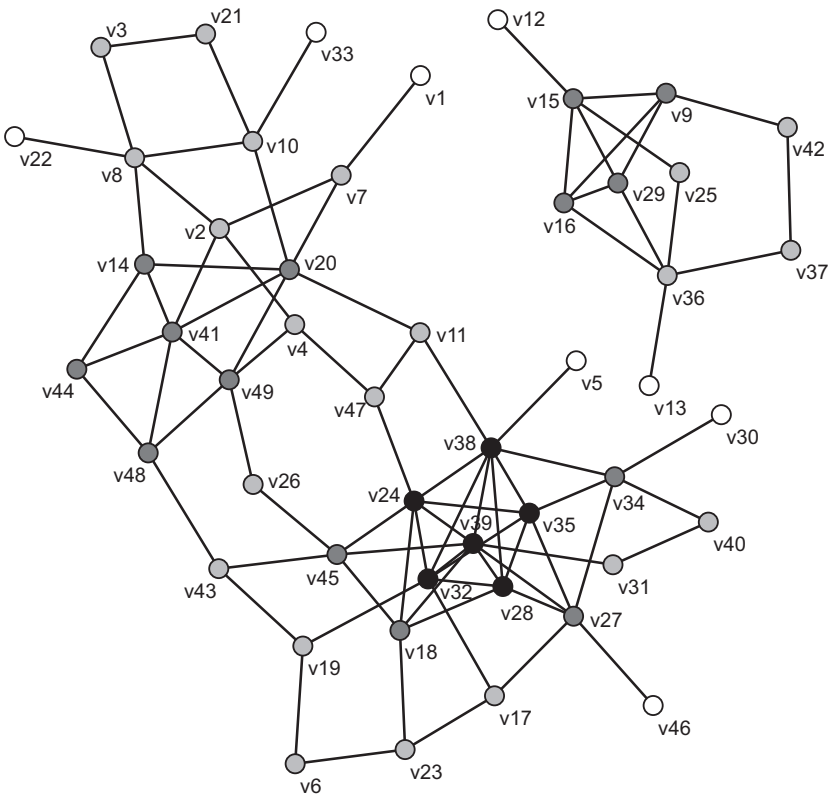


   a. v6 and v4 are not connected by a semipath.
   b. v1 and v4 are not connected by a path.
   c. There is no path from v7 to v3.
   d. There is no path from v1 to v8.

4. How many strong components does the network of Question 3 contain?
   a. No strong components
   b. One strong component

c. Two strong components
d. Three strong components

5. Which of the following statements is correct?
   a. A subnetwork is maximal if it cannot be enlarged without losing 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 containing two or more vertices 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 in the following figure. Vertex colors indicate the level of k (white: $k = 1$; light gray: $k = 2$; dark gray: $k = 3$; black: $k = 4$).



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 another village in the Turrialba region, 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. The 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, and Olen E. Leonard, *Turrialba: Social Systems and the Introduction of Change* (Glencoe, IL: The Free Press, 1953). We will also use these data in Chapter 9 on prestige.
- Chapter 7 in John Scott, *Social Network Analysis: A Handbook* (London: SAGE [4th edn. 2017], 1991) 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.
- S. B. Seidman's "Network structure and minimum degree" (*Social Networks* [1983], 269–87) introduced *k*-cores. R. D. Luce and A. Perry formally defined cliques in "A method of matrix analysis of group structure" (*Psychometrika* 14 [1949], 95–116).

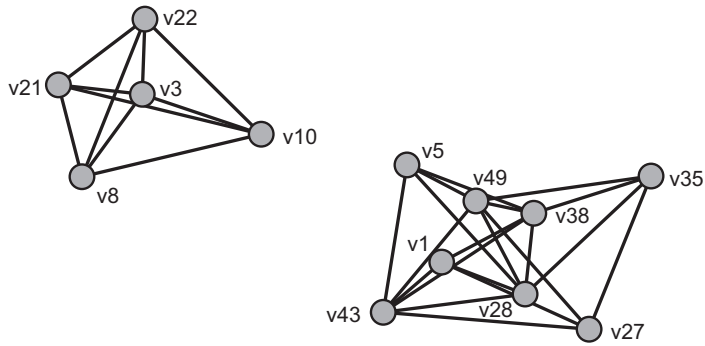
3.11 Answers

Answers to the Exercises

I. In the symmetrized network of San Juan Sur without multiple lines, vertex degree ranges from 1 to 12 and the average degree is 4.13

(change the degree partition into a vector and inspect the vector with the *Vector> Info* command).

II. Open the network `ExerciseII.net` and determine the *k*-core partition with one of the *Network> Create Partition> k-Core* commands. Inspecting the core partition with the *Partition > Info* command, you will see that 13 vertices belong to the 4-core, 33 (13 + 20) to the 3-core, 47 (33 + 14) vertices to the 2-core, and all 49 vertices (47 + 2) to the 1-core. You can extract the 4-core from the network with the *Operations> Network + Partition> Extract> SubNetwork Induced by Union of Selected Clusters* command (just extract class 4). The extracted 4-core is not connected, as shown in the following figure; it consists of two parts.



Answers to the Questions in Section 3.8

1. Answer c is correct. Symmetrizing a network means that unilateral and bidirectional 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. Moreover, 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 that 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 as follows: 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. Because there is no path in the opposite direction, 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; for example, 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 of size 2 or more, 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 have 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.