Part V - Roles

Cohesion, brokerage, and ranking are connected to social roles: being a member of a group, being a mediator, or being a superior. Each of these roles is associated with a particular pattern of ties. A blockmodel describes the social roles and associated patterns of ties in the network at large. Blockmodels offer a different perspective on the concepts discussed in previous chapters.

12. Blockmodels

12.1 Introduction

In previous parts of this book, we have presented a wide range of techniques for analyzing social networks. We have discovered that one structural concept can often be measured in several ways, e.g., centrality. We have not encountered the reverse, that is, a single technique which is able to detect different kinds of structures, e.g., cohesion and centrality. In this final chapter, we present such a technique, which is called blockmodeling.

Blockmodeling is a flexible method for analyzing social networks. Several network concepts are sensitive to exceptions, e.g., a single arc may turn a ranking into a rank-less cluster (Chapter 10). Empirical data are seldom perfect, so we need a tool for checking the structural features of a social network which allows for exceptions or error. Blockmodeling and hierarchical clustering, which are closely related, are such tools.

Although blockmodeling is a technique capable of detecting cohesion, core-periphery structures, and ranking, it does not replace the techniques presented in previous chapters. At present, blockmodeling is feasible and effective only for small dense networks. In addition, blockmodeling is grounded on other structural concepts: equivalence and positions, which are related to the theoretical concepts of social role and role sets. Centrality or prestige, as properties of individual vertices, are not measured by blockmodels.

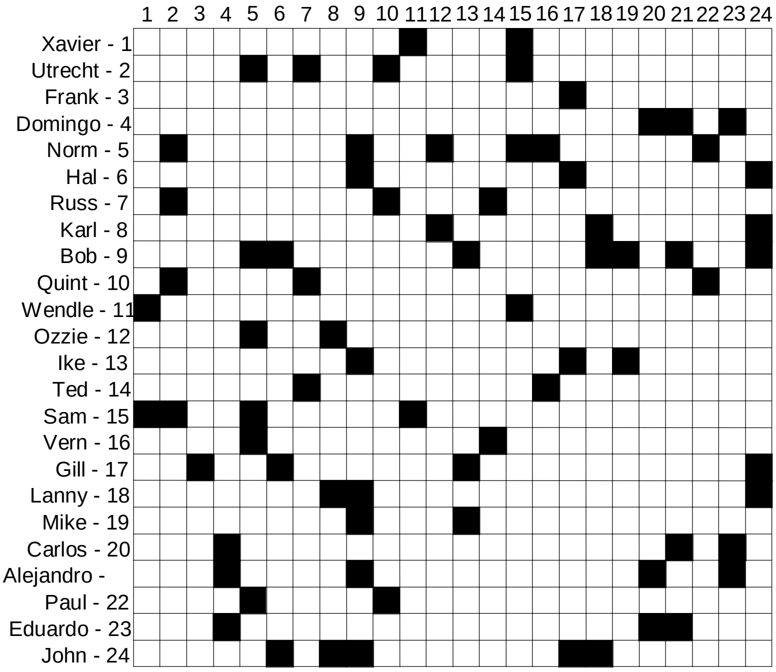
Blockmodeling uses matrices as computational tools and for the visualization of results. Therefore, we will introduce the matrix as a means to represent social networks before we will proceed to the concept of equivalence and the technique of blockmodeling.

12.2 Matrices and permutation

In social network analysis, matrices have been used in addition to sociograms for a long time. A matrix is an efficient tool for representing a small social network and for computing results on its structure. In addition, matrices offer visual clues on the structure of small and dense networks, which is what we will use them for in the present section.

A matrix is a two-way table containing **rows** and **columns**. The intersection of a row and a column is called a **cell** of the matrix. Figure 1 displays the matrix of the communication network of striking employees in a wood-processing facility (see Chapter 7). In this matrix, each row and column represent one vertex of the network, e.g., the first (highest) row and the first (left) column feature

Xavier. The cells in this row or column show Xavier's relations. In Figure 1, a black cell indicates that Xavier communicates with another employee (or with himself) and a white (empty) cell means that there is no communication. Note that a matrix usually contains numbers, e.g., '1' for the presence of a relation and '0' if there is no relation. In Figure 1, we replaced the numbers by black or white squares to highlight the pattern of communication relations.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Xavier - 1											■				■									
Utrecht - 2					■		■			■					■									
Frank - 3																	■							
Domingo - 4																				■	■		■	
Norm - 5		■							■			■			■	■						■		
Hal - 6									■								■							■
Russ - 7		■								■				■										
Karl - 8												■						■						■
Bob - 9					■	■							■					■	■		■			■
Quint - 10		■					■															■		
Wendle - 11	■														■									
Ozzie - 12					■			■																
Ike - 13									■								■		■					
Ted - 14							■									■								
Sam - 15	■	■			■						■													
Vern - 16					■									■										
Gill - 17			■			■							■											■
Lanny - 18								■	■															■
Mike - 19									■				■											
Carlos - 20				■																	■		■	
Alejandro -				■					■											■			■	
Paul - 22					■					■														
Eduardo - 23				■																■	■			
John - 24						■		■	■								■	■						

Figure 1 - Comunication lines among striking employees.

This type of matrix is called an **adjacency matrix** because we can tell from it which vertices are neighbors (adjacent) in the network, for instance, the black cells in the first row mean that Xavier (vertex 1) communicates with Wendle (vertex 11) and Sam (vertex 15). To be more precise, these black cells indicate that there is a relation *from* Xavier *to* Wendle and Sam. The row-entry contains the sender of the relation and the column-entry its recipient, so the first row contains relations *from* Xavier and the first column shows the relations *to* Xavier, e.g., from Wendle and Sam. It is not a coincidence that Xavier has the same neighbors in his row and column: the network is undirected, so Xavier's communication with Wendle implies that Wendle communicates with Xavier, etc. An edge is equivalent to a bi-directional arc, so an edge is represented by two arcs in an adjacency matrix. In general, the adjacency matrix of an undirected network is symmetric around the diagonal running from the top left to the bottom right of the matrix, which is usually referred to as 'the diagonal' of the matrix.

The adjacency matrix of Figure 1 does not contain black cells on the diagonal because these cells represent the relation of a vertex to itself and the employees were not considered to communicate with themselves. Cells on the diagonal of an adjacency matrix often receive special treatment because they feature loops.

Because the same vertices define the rows and columns of an adjacency matrix, the adjacency matrix is square by definition. In contrast, a two-mode network such as the network of multiple directors in Scotland (Chapter 5), is represented by a matrix which may be rectangular but not square. We can place the firms in the rows and the directors in the columns and still include all relations in the cells of this matrix because firms can only be directly related to directors. Such a matrix is called an **incidence matrix**. In an incidence matrix, diagonal cells do not represent loops.

The pattern of black cells in a matrix offers visual clues on the structure of the network because we see which lines are present (black) or absent (white). Just like a sociogram, however, a matrix discloses network structure only if its vertices are carefully placed. Figure 1, for instance, shows a seemingly random pattern of black cells. It does not reveal the structure of the network because the employees are listed in an arbitrary order. If we order them by their language (English or Spanish) and age (below or over 30 years), the black cells display a much more regular pattern (Figure 2). Now, it is easy to see that lines occur predominantly within the ethnic and age groups: no more than two lines (Karl-Ozzie and Bob-Norm) exist between the groups.

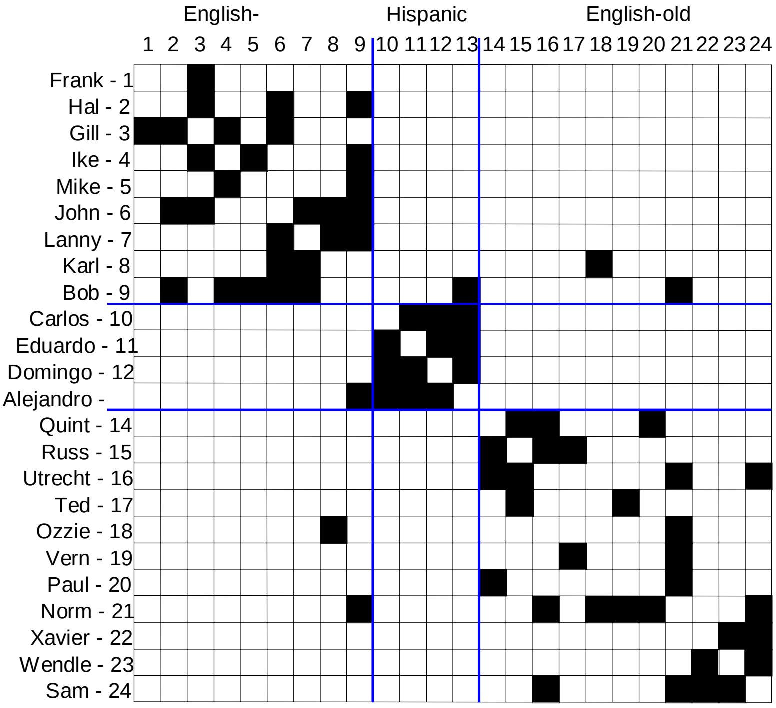


Figure 2 - The matrix of the strike network sorted by ethnic and age group.

A reordering or sorting of vertices is called a permutation of the network. Essentially, a permutation is a list with an entry for each vertex in the network, specifying its new vertex number. In other words, a permutation is a renumbering of the vertices in the network.

A **permutation** of a network is a renumbering of its vertices.

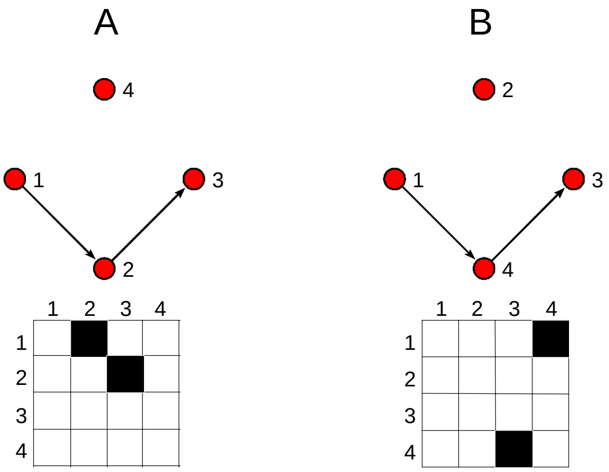


Figure 3 - A network and a permutation.

If we assign new numbers to the vertices, the structure of the network does not change. Compare, for example, networks A and B in Figure 3. We exchanged the numbers of vertices two and four, but this does not affect the structure of the network: networks A and B are **isomorphic**, that is, they have the same structure. In the matrix, we exchanged vertex numbers in the rows *and* the columns and we reordered the matrix obtaining a different matrix for the same structure.

The matrices look different but they describe the same structure. This means that we can represent the same network by a number of different matrices, just as we can draw many different sociograms for one network. A permutation rearranges a matrix just like an energy command redraws a sociogram. Therefore, we can use permutations to find matrices which reveal the structure of a network. Subsequent sections will show how to do this.

The strike network permuted by ethnic and age groups (Figure 2) shows the pattern which characterizes cohesive subgroups: the black (non-empty) entries 'cluster' around the diagonal of the matrix where they form clumps. The clumps identify subgroups of actors who maintain relations predominantly within their groups. In our example, the clumps nicely reflect the ethnic and age groups.

Application

Network droplist

In Pajek, you can display the matrix of a network by double-clicking its name in the Network drop list. In the dialog box which appears, enter a '1' if you want to display a binary matrix, that is, a matrix which only tells whether a line is present (# sign) or absent (. sign). Figure 4 shows part of the original strike network (included in the Pajek project file `strike.paj`) displayed as a binary matrix. Note that the listing consists of raw unformatted text, so it should be displayed in a fixed wide type font such as Courier. If you want to display the line values in the adjacency matrix, type a '2' in the dialog box to obtain a valued matrix. In a valued matrix, an absent line is represented by a '0' in the cell.

```
                                   1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
                 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
----------------------------------------------------------------
 Xavier      1.  . . . . . . . . . . # . . . # . . . . . . . . .
 Utrecht     2.  . . . . # . # . . # . . . . # . . . . . . . . .
 Frank       3.  . . . . . . . . . . . . . . . . # . . . . . . .
 Domingo     4.  . . . . . . . . . . . . . . . . . . . # # . # .
 Norm        5.  . # . . . . . . # . . # . . # # . . . . . # . .
```

Figure 4 - Partial listing of the strike network as a binary matrix.

In Pajek, networks of 100 vertices or more can not be displayed as matrices, because they yield enormous matrices. Therefore, the options 'binary' and 'valued' are not available for larger networks, which are automatically reported as lists of arcs and edges. In these lists, each line represents a vertex, which is identified by its number and label followed by the numbers of all vertices which receive a line from it. This type of listing is also the third option ('Lists') for displaying small networks.

File>Network >Export Matrix to EPS >Original

The raw text matrices are not suited for high quality printing. For this end, the matrix can be saved with the command *File>Network>Export Matrix to EPS>Original* in PostScript format. Line values are automatically translated to the darkness of cells, as in Figure 1 and Figure 2. Larger networks can be exported in this way, but large matrices are usually not very helpful visualizations for detecting the structure of a network.

Partition >Make Permutation

As we have argued, a matrix is usually more informative if it is reordered. In the example of the strike network, we must reorder the vertices according to their membership of the ethnic and age groups, which is available to us as a partition (`strike_groups.clu` included in the project file `strike.paj`). It is easy to derive a permutation from a partition such that vertices in the same class receive consecutive numbers: select the partition in the Partition drop list and execute the *Make Permutation* command, which is located in the *Partition* menu. Pajek creates a new permutation assigning the lowest vertex numbers to the vertices in the first class of the partition, et cetera.

Permutation drop list

The permutation is displayed in the Permutation drop list of Pajek's main screen. You may inspect and edit a permutation in the usual ways. When you edit a permutation, you will see one line for each vertex containing two numbers. The first number is the new vertex number and the second number is the original vertex number. If a compatible network is active in the Network drop list, the vertex label is also displayed.

File>Network >Export Matrix to EPS >Using Permutation

When the network, the partition, and the permutation are selected in their respective drop lists, you can export the adjacency matrix to an Encapsulated PostScript file with the command *File>Network>Export Matrix to EPS>Using Permutation*. A dialog box prompts for a name of the file in which the matrix must be saved and another dialog box asks whether blue lines should be drawn between classes according to the active partition. In a viewer capable of reading PostScript, the result should look like Figure 5.

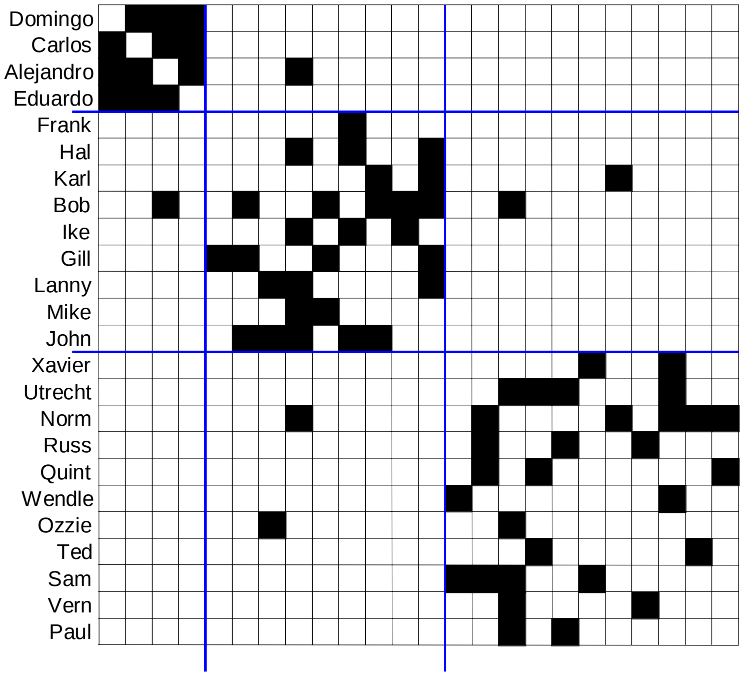


Figure 5 - The strike network permuted according to ethnic and age groups.

Operation>Reorder >Network

Operation>Reorder >Partition

The permutation of ethnic and age groups can also be used to reorder the network itself. If the network and permutation are selected in their drop lists, the *Operation>Reorder>Network* command creates a new permuted network. Display the reordered network as a binary matrix by double-clicking its name in the drop list and you will see that the four Hispanic employees have received vertex numbers from one to four (Figure 6). Note that the original partition according to ethnicity and age is not compatible with the permuted network but it can also be reordered: make sure that the original partition and permutation are active in their drop lists and execute the command *Operation>Reorder>Partition*.

```
                                    1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
                  1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
-----------------------------------------------------------------
  Domingo     1.  . # # # . . . . . . . . . . . . . . . . . . . .
  Carlos      2.  # . # # . . . . . . . . . . . . . . . . . . . .
  Alejandro   3.  # # . # . . . # . . . . . . . . . . . . . . . .
  Eduardo     4.  # # # . . . . . . . . . . . . . . . . . . . . .
  Frank       5.  . . . . . . . . . # . . . . . . . . . . . . . .
```

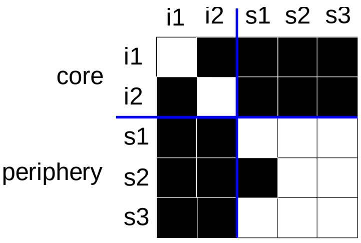
Figure 6 – Part of the permuted strike network displayed as a binary network.

12.3 Roles and positions: equivalence

In social theory, positions and roles are important and related theoretical concepts. A position, e.g., the position of being an instructor at a university, is usually connected to a social role or a role set, e.g., tutoring students, conferring with colleagues, etc. It is hypothesized that this role or role set involves a particular pattern of ties and relations, e.g., towards students, colleagues, and superiors. Sociologists, social psychologists, and other social scientists investigate the nature of social roles and role sets by observing interactions and by interviewing people about their motives and their perceptions of the roles they play.

In social network analysis, we concentrate on the patterns of relations. We want to identify actors which have similar patterns of ties to find out whether they are associated with a particular role or role set, or we want to check whether people with similar role sets are involved in characteristic patterns of relations. In social network analysis, a position is equated to a particular pattern of ties. Actors with similar patterns of relations are said to be relationally **equivalent**, to constitute an **equivalence class**, or to occupy equivalent **positions** in the network.

Figure 7 offers a simple example illustrating these ideas. Two instructors (i1 and i2) within one department supervise three students (s1 to s3). They contact the students and they are contacted by the students. The instructors interact, so they are a cohesive subgroup and their interaction may cause them to behave in a similar way. The three students, however, do not necessarily interact. Nevertheless, they are in the same position with respect to the supervisors, hence they may act similarly towards them. They are relationally equivalent although they are not a cohesive subgroup. It is important to note that the external relations to members of other positions are just as important as internal relations within a position in the concept of equivalence.



		i1	i2	s1	s2	s3
core	i1		■	■	■	■
	i2	■		■	■	■
periphery	s1	■	■			
	s2	■	■	■		
	s3	■	■			

Figure 7 - Hypothetical relations among two instructors (i) and three students (s).

Figure 7 is an example of a small core-periphery structure in which the two instructors constitute the core (one position) and the students the periphery (the other position). Ties occur predominantly within the core and between the core and the periphery, so we see a horizontal and a vertical ‘strip’ of ties in the permuted matrix.

So far, we loosely described the concept of equivalence. Now, let us define one type of equivalence formally, namely, structural equivalence: two vertices are structural equivalent if they have identical ties with themselves, each other, and all other vertices. This definition implies that structural equivalent vertices can be exchanged without consequences to the structure of the network.

Two vertices are **structural equivalent** if they have identical ties with themselves, each other, and all other vertices.

In our example, in which arcs are either present or absent, let us compare the two vertices in the core (instructors i1 and i2). Clearly, the two instructors have identical ties to themselves and to each other: none of them communicates with himself or herself (no loops), and the relation among them is symmetric. In addition, their relations with vertices in the other position – the students – are also identical. If instructor i1 is connected to a student, e.g., student s2, then the other instructor is also connected to this student. As a consequence, the rows of the two instructors are identical, except for the cells on the diagonal because they are not supposed to contact themselves. The same is true for their columns, which represent the relations received by the instructors. We may exchange the two instructors without changing the structure of the network.

In general, we can say that vertices which are structural equivalent have identical rows and columns (except for the cells on the diagonal) in the adjacency matrix. With this in mind, it is easy to see that the three students in the periphery (s1, s2, and s3) are not completely structural equivalent because vertex s2 is related to vertex s1 but the reverse is not true, so they do not have identical relations to each other. Student s3 is not related to s1, so s/he is not structural equivalent to s1 or s2.

Structural equivalence is based on the similarity or dissimilarity between vertices with respect to the profile of their rows and columns in the adjacency matrix. The dissimilarity of two vertices can be calculated and expressed by an index which ranges from zero (completely similar) to one (completely different). In Figure 7, the row and column of instructor i1 is perfectly similar to the row and column of instructor i2, so their dissimilarity score is zero (see Table 1). Students s1, s2, and s3 are not completely similar in this respect, so their dissimilarity score is larger than zero (ranging from 0.0625 to 0.125) but they are more similar to each other than to the instructors in the core (dissimilarities of 0.1875 or 0.25).

Table 1 - Dissimilarity scores in the example network.

	i1	i2	s1	s2	s3
i1	0.0000	0.0000	0.1875	0.1875	0.2500
i2	0.0000	0.0000	0.1875	0.1875	0.2500
s1	0.1875	0.1875	0.0000	0.1250	0.0625
s2	0.1875	0.1875	0.1250	0.0000	0.0625
s3	0.2500	0.2500	0.0625	0.0625	0.0000

Knowing the dissimilarities between all pairs of vertices, how can we cluster vertices which are (nearly) structural equivalent into positions? This can be achieved with a well-known statistical technique, which is called **hierarchical clustering**. First, this technique groups the vertices which are most similar. In our example, instructors s1 and s2, who are completely similar with respect to their ties, are merged into a cluster. Then, hierarchical clustering groups the next pair of vertices or clusters which are most similar and it continues until all vertices have been joined.

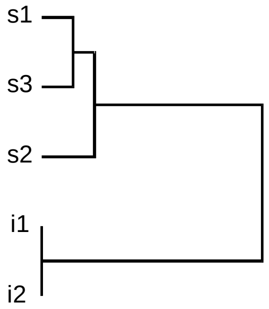


Figure 8 - A dendrogram of similarities.

Figure 8, which is called a **dendrogram**, visualizes the clustering process. You must read it from left to right. First, instructors i1 and i2 are joined because they are perfectly similar: their dissimilarity is zero. Then, students s1 and s3 are joined (at dissimilarity level 0.06, see Table 1). In the third step, student s2 is added to the cluster of s1 and s3. Finally, this cluster is merged with the cluster of core vertices (i1 and i2) in the last step of the clustering process.

In the dendrogram, the length of a horizontal 'branch' represents the dissimilarity between two vertices or clusters at the moment when they are joined, so you can see that the last step merges two very different clusters. If you want to partition the vertices into two clusters, you should separate instructors i1 and i2 from the students. In general, a hierarchy of clusters is split at the place or places where the branches make large jumps. In this way, you can detect clusters of vertices which are structural equivalent or nearly structural equivalent.

Application

Operations
>Extract from Network
>Partition

Let us apply the concept of structural equivalence to the world trade network, which we introduced in Chapter 2. The Pajek project file `world_trade.paj` contains the network and a partition identifying world system positions in 1980. Figure 9 shows the matrix containing the countries with known world system position in 1980 (we extracted classes 1 to 4 of the partition from the network with the *Operations>Extract from Network>Partition* command). Line values indicate the gross value of imports; they are represented by the color of the cells in the PostScript matrix: higher values are represented by darker cells. The distribution of gross imports is highly skewed because a couple of countries trade very high volumes of goods. We changed all imports over 1 billion US$ to 1 billion to obtain slightly darker cells for trade relations with lower gross value. Note that these adjustments are made only for a better display of the matrix. We will use the original trade network in the remainder of this section.

The network is directed, so the matrix is not symmetric although the values of imports are often in the same range as the values of exports. The matrix reveals some characteristics of a core-periphery structure which we have noted before: many and strong ties within the core and between the core and the semiperiphery but few and weak ties within the semiperiphery and the periphery. As a result, the ties concentrate in the horizontal and vertical strip which is associated with the core countries.

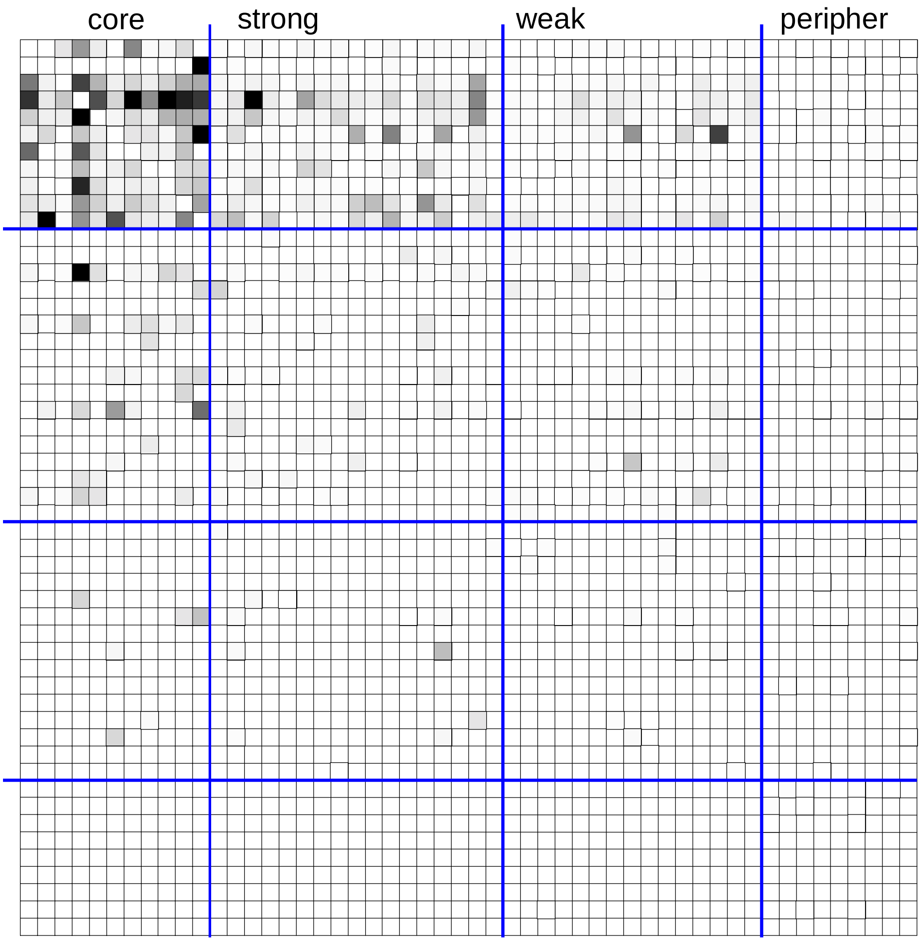


Figure 9 - Imports of manufactured products and world system position in 1980.

Cluster >Create Complete Cluster

Now, let us calculate the dissimilarity of the rows and columns of the countries in the original trade network. First, we must make a preliminary step. The dissimilarity method is computationally complex, so it should be used for small networks or for a small part of a large network. Therefore, the method requires that we indicate which vertices it should use. We must identify them in a special data object, which is called a cluster. In our example, we want to include all countries, so we create a cluster containing all vertices with the *Cluster>Create Complete Cluster* command. The total number of vertices in the network is shown by default in the dialog box issued by this command, so you may simply press the *OK* button. The cluster created by this command is listed in the Cluster drop list and it can be edited in the usual way.

Partition>Make Cluster

If you want to restrict your analysis to a part of the network, however, identify the vertices for which you want to compute dissimilarities in a partition and translate the desired class or classes from this partition into a cluster with the *Partition>Make Cluster* command. Dialog boxes will prompt you for the class number or range of class numbers of the partition which must be selected. For example, you may restrict the calculation of dissimilarities to the core countries of 1980 by translating class 1 of the world system positions partition to a cluster.

In this case, the *Dissimilarities* command, which we will discuss in the next paragraph, calculates dissimilarities for the core countries only but it takes into account the relations of the core countries to non-core countries.

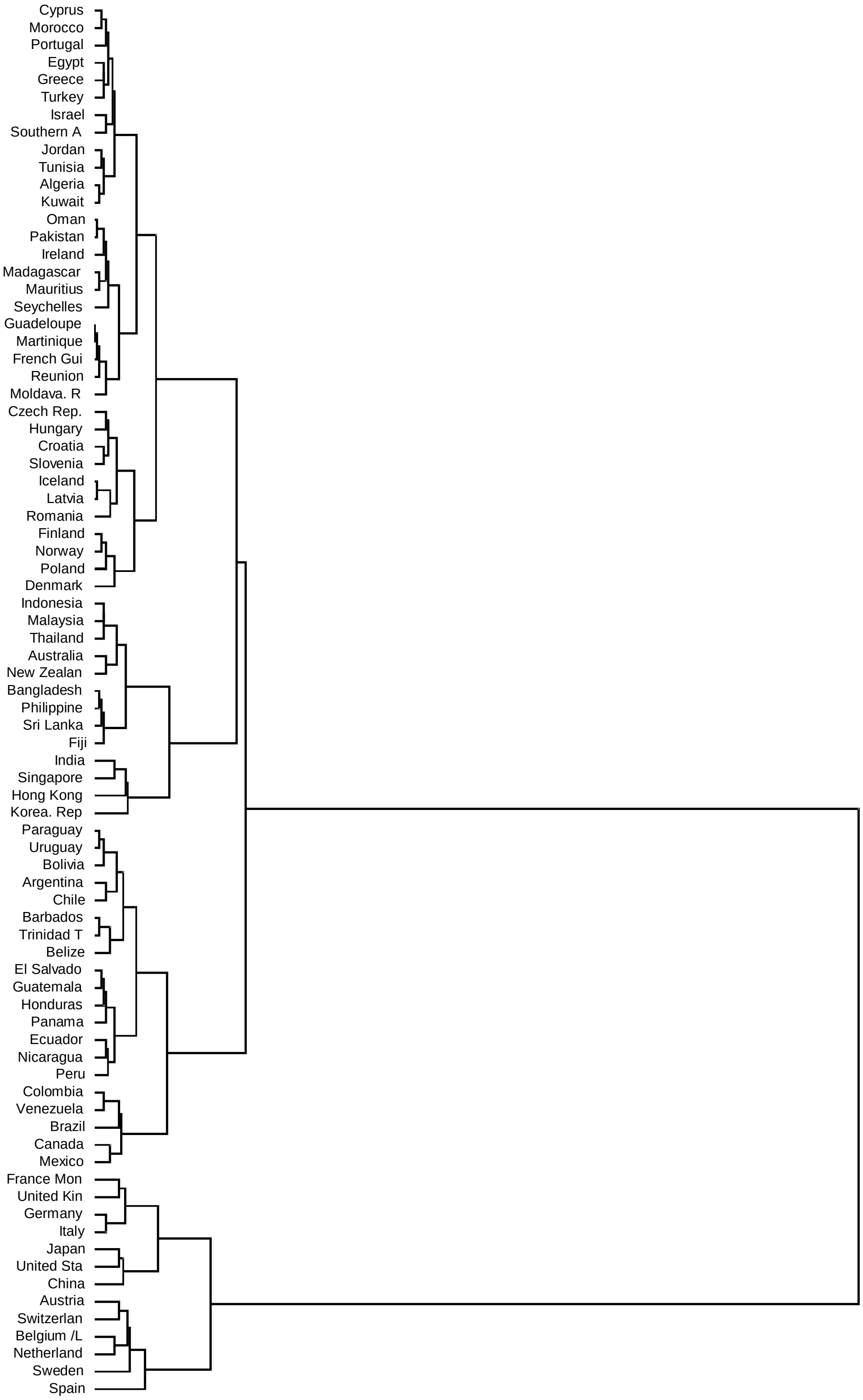


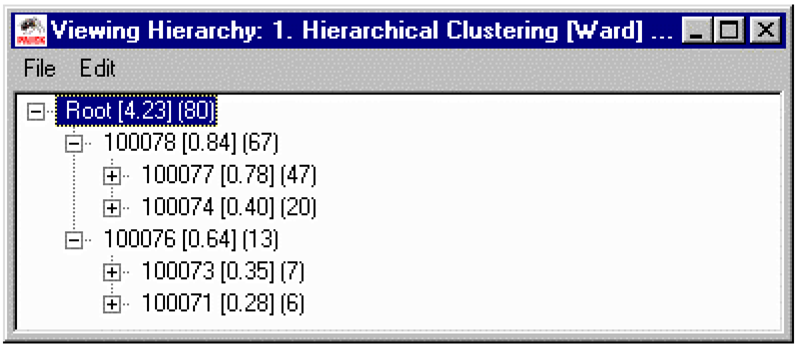
Figure 10 - Hierarchical clustering of the world trade network.

Operations>Dissimilarity >d1>All

Since we need a network and a cluster to compute dissimilarities in Pajek, the *Dissimilarity* commands are located in the *Operations* menu. There are several dissimilarity indices but we will only present and use the index *d1*. Consult a statistics handbook to learn more about the other indices. Calculate it for all arcs – input and output – to obtain suitable dissimilarity values. Executing this command, Pajek calculates the dissimilarities and reports them in the Report screen if the option *Operations>Dissimilarity> Options>Report Matrix* has previously been selected (note: do not select the other options in this submenu). The command stores the dissimilarities as line values in a new network, which you may list or print in the usual ways (see Section 12.2). Note that this network is directed and very dense because each pair of vertices which are not completely similar (hence: have a dissimilarity larger than zero) are connected by a pair of arcs. As a rule, do not attempt to draw and energize this network.

While executing a dissimilarities command, Pajek automatically tries to apply hierarchical clustering to the newly created network of dissimilarities. It prompts the user to specify the name of a file in which the dendrogram of the clustering is stored. The dendrogram is saved and not shown because it is in Encapsulated PostScript format. You can view it (Figure 10) with a PostScript interpreter, e.g., Ghostscript (see Appendix 2) or print it on a PostScript printer. In addition, the results of the hierarchical clustering are saved as a permutation and a hierarchy.

The dendrogram of the world trade network, which is depicted in Figure 10, shows two very dissimilar clusters of countries in the world trade network: ten Western European countries, the US, Japan, and China are clearly separated from the remaining 67 countries, most of which are poorer countries.



```
⊟ Root [4.23] (80)
    ⊟ 100078 [0.84] (67)
        ⊞ 100077 [0.78] (47)
        ⊞ 100074 [0.40] (20)
    ⊟ 100076 [0.64] (13)
        ⊞ 100073 [0.35] (7)
        ⊞ 100071 [0.28] (6)
```

Figure 11 - Hierarchical clustering of countries in the Hierarchy Edit screen.

Hierarchy Edit screen

This can also be inferred from the hierarchy created as a result of the *Dissimilarity* command, which is labeled `Hierarchical Clustering [Ward] of N2 (80)`. Open the hierarchy in an Edit screen (click on the button with the writing hand at the left of the Hierarchy drop list) and expand the root as well as the next layer of clusters by clicking on them to obtain the listing depicted in Figure 11. The root unites the two principal clusters. Except for the root, the figures in square brackets tell you the dissimilarity of the clusters or vertices which are joined; a larger value means that they are more dissimilar. The cluster of thirteen countries is internally more similar (0.64) than the larger cluster

(0.84), which corresponds to the fact that the first split within the larger group is more to the right than the split within the smaller group in the dendrogram.

[Hierarchy Edit screen] Edit>Show Subtree

How do we know which countries belong to a particular cluster? We can find the names of the countries in the Hierarchy Edit screen in the following way, provided that a compatible network is active in the Network drop list. First, make sure that the option *Show Subtree* is selected in the *Edit* menu of the hierarchy's Edit screen. Otherwise, Pajek only displays the names of the vertices which were added to the cluster in the present step of hierarchical clustering. Second, select a cluster in the Edit screen by left-clicking it and right-clicking it subsequently. In a new window, the numbers and labels of the vertices in this cluster and all of its subclusters are listed. If you apply this to the cluster labeled '100071', for example, you will see that it contains Austria, Switzerland, Belgium /Luxembourg, The Netherlands, Sweden, and Spain.

Hierarchical clustering gradually merges vertices into clusters and small clusters into larger clusters. Which clusters represent structural equivalence classes and which do not? Under a strict approach to structural equivalence, vertices with zero dissimilarity are structural equivalent. In real social networks, however, such vertices are seldom found, so we consider clusters of vertices which are 'not very dissimilar' to represent structural equivalence classes.

Which vertices are 'not very dissimilar'? There is no general answer to this question. It is up to you to decide on the number of equivalence classes that you want, that is, how many times you want to 'cut up' the dendrogram, but you should always cut it from 'right to left': separate the most dissimilar clusters first. In the world trade example, you should separate the thirteen rich countries from the other countries first. Then, you could make a subdivision within the latter cluster because these countries are more dissimilar (0.84) than the thirteen rich countries (0.64), and so on, until you reach the desired number of equivalence classes or further subdivisions seem to be arbitrary or meaningless.

Let us divide the trade network into four structural equivalence classes, because we have a partition into four world system positions (core, strong semiperiphery, weak semiperiphery, and periphery). We split the cluster of 67 countries (dissimilarity is 0.84) and its largest subcluster (dissimilarity is 0.78). Now, we can create a partition from the hierarchy which identifies these four clusters. This is done in two steps.

[Hierarchy Edit screen] Edit>Change Type

First, we must close the clusters in the hierarchy which we do not want to split up any further. Select a cluster by left-clicking it in the Hierarchy Edit screen and select *Change Type* from the *Edit* menu of the Hierarchy Edit screen or press Ctrl-t. Now, the message `(close)` appears behind the selected cluster. Repeat this for the other clusters which must be closed but do *not* apply it to any cluster which must be subdivided.

Hierarchy >Make Partition

Second, execute the *Make Partition* command from the *Hierarchy* menu in the Main screen. This command creates a partition in which each closed cluster is represented by a class. When you draw this partition in the original world trade network, you will notice that the equivalence classes represent a mixture of trade position and geography: the core countries, which are Western European countries, the USA, Japan, and China, are delineated from three regional positions: the Americas, Asia with Oceania, and Europe (including former colonies) with the Middle East.

File>Network
>Export Matrix to EPS
>Using Permutation

So far, we have discussed the dendrogram and the hierarchy created by the *Dissimilarities* command but not the permutation. The permutation is labeled `Hierarchical Clustering Permutation [Ward]` and it identifies the order of the vertices as represented in the dendrogram. When you want to print the matrix reordered by the results of hierarchical clustering, you can use this permutation. It is compatible with the partition which you created from the hierarchy, so you can obtain a matrix with blue lines which indicate the splits that you have made in the hierarchy of clusters (see Section 12.2).

12.4 Blockmodeling

In previous sections, we have drawn adjacency matrices with (blue) lines demarcating classes of vertices, e.g., ethnic/age groups among the striking employees (Section 12.2), advisors versus students in a small example network, and world system positions of countries in the trade network (Section 12.3). By now, we should note that these lines divide the adjacency matrix into rectangles and these rectangles are called **blocks**.

A **block** contains the cells of an adjacency matrix which belong to the cross-section of one or two classes.

We can describe the structure of the network (within and between positions) by analyzing the blocks of the adjacency matrix. The blocks along the diagonal express the ties within a position. In an ideal core-periphery structure, e.g., Figure 12, vertices are linked within the core (vertices i1 and i2), whereas the peripheral vertices (s1 through s3) are not directly linked. The blocks off the diagonal represent the relations between classes, e.g., the relations between the core and periphery. The students derive their identity from their dependence on the instructors but not from their internal ties.

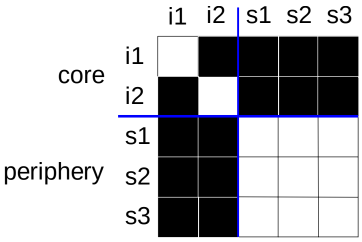


Figure 12 - An ideal core-periphery structure.

12.4.1 Blockmodel

Adjacency matrices of networks containing structural equivalence classes have a very remarkable feature, namely, their blocks are either complete or empty (null-blocks), if we disregard cells on the diagonal. This results from the criterion of structural equivalence that equivalent vertices have identical rows and columns.

To understand this, imagine that there is one relation among the students of Figure 12, e.g. from s2 to s1. Structural equivalent vertices must have identical relations to each other, so s1 must also be connected to s2. If all students are structural equivalent, s3 must have identical relations as s1 and s2, so it must be linked with s1 and s2. Now, the block is complete, except for the diagonal. This is also true for relations between positions.

Now that we know that the adjacency matrix of a network with structural equivalence classes contains only complete and null blocks, we may simplify the adjacency matrix by shrinking each class of vertices to one new vertex (entry in the matrix) and mark the block type of each cell in the new matrix, which is either complete (com) or empty (- or null) in the case of structural equivalence. This shrunk matrix is called an image matrix and it contains all information which was present in the original adjacency matrix. Figure 13 shows the image matrix of a simple core-periphery structure and a graphical representation of the relations within and between equivalence classes (positions) in which an arc indicates a complete block and the absence of an arc signifies a null block.

	core	periphery
core	com	com
periphery	com	- (null)

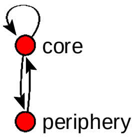


Figure 13 - Image matrix and shrunk network.

A **blockmodel** assigns the vertices of a network to classes and it specifies the permitted type(s) of relation within and between classes.

The image matrix is the last ingredient we need to define a blockmodel. A **blockmodel** for a network consists of a partition and an image matrix. The partition assigns vertices to equivalence classes and it divides the adjacency matrix of the network into blocks. The image matrix specifies the types of relations within and between the classes because it says which kinds of blocks are allowed and where they may occur. The blockmodel of the core-periphery structure of Figure 12, for instance, consists of a partition which assigns instructors i1 and i2 to one class and the three students (s1, s2, and s3) to another class and the image matrix specifying the relations between the blocks shown in Figure 13.

A blockmodel describes the overall structure of a network and the position of each vertex within this structure. In the example of the instructors and students, the image matrix shows the type of equivalence which applies to the network. This network contains structural equivalence classes because there are only complete and empty blocks. In addition, the image matrix reveals the core-periphery structure of the network because the complete blocks are arranged within one horizontal strip and one vertical strip. Class 1 represents the core, which is internally linked, and class 2 identifies the periphery. Finally, the partition tells us which actors are part of the core (the two instructors, who constitute the first class) and which actors belong to the periphery (the three students in class 2). A blockmodel is an efficient devise for characterizing the overall structure of a network and the positions of individual vertices.

12.4.2 Blockmodeling

Until now, we have assumed that we knew the blockmodel of a network, that is, the partition of vertices into classes and the image matrix specifying the permitted types of blocks. In a research project, naturally, we work the other way around: we have a network and we want to find the blockmodel which captures the structure of the network. The technique to obtain this blockmodel is called **blockmodeling**.

In general, blockmodeling consists of three steps. In the first step, we specify the number of classes in the network, e.g., two classes or positions if we hypothesize a simple core-periphery structure. In the second step, we choose the types of blocks which are permitted to occur and, optionally, the locations in the image matrix where they may occur. In the case of structural equivalence, for instance, we permit only complete and empty blocks to occur and we expect one complete block (the core) and one empty block (the periphery) along the diagonal. Finally, the computer partitions the vertices into the specified number of classes according to the conditions specified by the model and, if necessary, it chooses the final image matrix for the model. In this third step, the blockmodel is completed.

The first two steps define the image matrix: we fix the number of classes and the types of blocks (relations) but we do not yet know which vertices belong to a particular class and sometimes we do not know exactly which block type will be found in which part of the image matrix. That will be settled in the third step. It goes without saying that we must have some knowledge or expectations about the network in order to choose an appropriate number of classes and to specify types of relations among classes which make sense. We should have reasons for expecting a core-periphery structure and structural equivalence in the example of contacts between instructors and students.

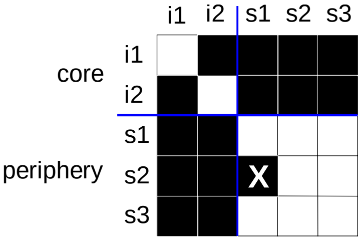


Figure 14 - Error in the imperfect core-periphery matrix.

Empirical networks, however, seldom match the ideal represented by the image matrix. Errors occur but they can be checked easily. Suppose you know which vertices belong to each class, then you can check whether each block of the adjacency matrix is of the right type according to the image matrix. In fact, you compare an ideal matrix (Figure 12) to the real matrix (Figure 7). In the case of structural equivalence, count the missing lines within the blocks that should be complete (none in this example) and count the number of lines which occur in the blocks that should be empty (one error: the arc from student s1 to student s2, see Figure 14) to obtain an error score which indicates how well the ideal matrix fits the real network.

In this approach, the third step of blockmodeling boils down to finding the partition of vertices into equivalence classes which yields the lowest error score, that is, which fits the ideal matrix best. First, the computer assigns vertices at random to the specified number of classes. Then, it calculates the error score of this solution by comparing the actual matrix to an ideal matrix represented by the image matrix. Next, it tries to decrease the error score by moving a randomly selected vertex from one to another cluster or by interchanging two vertices in different clusters. It continuous this process until it can not improve the error score anymore.

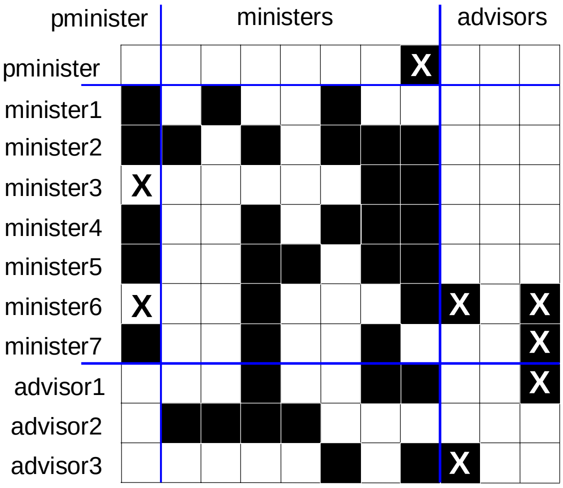
This optimization approach to blockmodeling has the advantages and disadvantages of all optimization techniques, e.g., the *Balance* command, namely, if applied repeatedly, it is likely to find the optimal solution but most of the times you cannot be sure that no better solution exists. In addition, you must be aware that another number of classes or other permitted types of blocks may yield blockmodels which fit better. Usually, it is worthwhile to apply several slightly different blockmodels to the data set, e.g., with another number of classes or other constraints on the relations within or between blocks. This underlines the importance of careful considerations on the part of the researcher concerning the image matrix which is hypothesized. Moreover, trees are troublesome in exploratory blockmodeling because they contain many vertices which may be exchanged between classes without much impact on the error score, so apply blockmodeling only to rather dense (sections of) networks.

In this optimization technique, errors can be weighted and line values can be used. We do not go into details here, but it should be noted that lower error scores indicate better fit and an error score of zero always represents a perfect fit.

12.4.3 Regular equivalence

Structural equivalence requires that equivalent actors have the same neighbors. In several applications of social network analysis, this criterion is too strict because it does not cluster actors who fulfill the same role in different locations, e.g., teachers at different universities, who have different students, so they have ties with similar people but not with the same people.

For these situations, another type of equivalence has been defined: **regular equivalence**. Vertices which are regular equivalent do not have to be connected to the same vertices, but they have to be connected to vertices in the same classes. This sounds like a circular argument but it is not. In the student government discussion network (Chapter 10), for instance, all advisors are expected to choose ministers for discussing student politics because they are supposed to advise the ministers. However, they do not have to advise the same ministers and they do not have to advise all ministers, e.g., advisor1 chooses ministers one to four but advisor3 selects ministers five and seven (Figure 15). In reverse, each minister is supposed to use the services of at least one advisor but s/he is not obliged to take advice from all advisors. This is also true for relations within a class: if one minister selects another minister, each minister must select a peer and must be selected by a peer. One peer, however, suffices: they do not have to be related to all peers, so their block is not necessarily complete.



	pminister										
pminister								☒			
minister1	■		■			■					
minister2	■	■		■		■	■	■			
minister3	X						■	■			
minister4	■			■		■	■	■			
minister5	■			■	■		■	■			
minister6	X			■				■	☒		☒
minister7	■			■			■				☒
advisor1				■			■	■			☒
advisor2		■	■	■	■						
advisor3						■		■	☒		

Figure 15 - Matrix of the student government network.

We can detect regular equivalence by means of blockmodeling because there is a special block type associated with regular equivalence, which is called a **regular block**. A regular block contains at least one arc in each row (everyone selects at least one actor) and in each column (everyone is selected at least once). Regular equivalence allows regular blocks and null blocks. Note that a complete block is always a regular block, so structural equivalence is a special kind of regular equivalence or, in other words, regular equivalence is more general than structural equivalence.

A **regular block** contains at least one arc in each row and in each column.

In the student government network with three classes (one class for each formal position – see Figure 15), two blocks are regular: the choices of advisors among ministers and the choices among ministers. The block containing choices from ministers to the prime minister would have been complete (thus, regular) if ministers three and six had also chosen the prime minister. The two missing choices are represented by black crosses in Figure 15; they contribute two units to the error score of the regular equivalence model for this network.

In Figure 15, two blocks are empty: the choices from advisors to the prime minister and vice versa. The social distance between these two classes seems to be too large to be spanned by direct consultation. The remaining three blocks are neither null nor regular, so they contain at least one violation against the regular equivalence model. The number of errors is minimal if we assume these blocks to be empty, so all six choices in these blocks are errors (white crosses) and we assume that the ideal matrix contains null blocks here. In our image matrix, we merely specified that all blocks should be empty or regular. While evaluating the error score, we discover that it is least erroneous to expect empty blocks here. Thus, we fix the type of these blocks to null blocks.

Figure 16 shows the image matrix and the number of errors in each block (the error matrix), which summarize the results. Class one contains the prime minister, class two contains the ministers, and the advisors are grouped in class three.

image matrix

	1	2	3
1	- (null)	- (null)	- (null)
2	com	reg	- (null)
3	- (null)	reg	- (null)

error matrix

	1	2	3
1	0	1	0
2	2	0	3
3	0	0	2

Figure 16 - Image matrix and error matrix for the student government network.

The student government discussion network is an example of a ranked structure which entails a particular location of block types. In a ranked structure, actors are supposed to choose ‘up’. If the ranks are ordered such that the highest rank is in the first rows (and columns) and the lowest rank occupies the last rows (and columns), we should not encounter choices in the blocks above the diagonal of the matrix because they would point from a higher rank (rows) towards a lower rank (columns). Indeed, we find empty (null) blocks only above the diagonal in the image matrix of the student government network, which is a general property of a ranked structure.

Instead of using a particular type of equivalence to define the block types which are allowed, we may use any combination of permitted block types to characterize a network by specifying the type(s) allowed for each individual block, for instance, a complete block for the ministers to prime minister block, a regular block for the ministers themselves, and an empty block for the ministers to advisors block. This is known as **generalized blockmodeling**. Note that there are more block types than the three which we present here. Some patterns of block types are known to contain classes of networks, e.g., core-periphery models and models of ranks. These classes have a particular substantive meaning, so it is easy to interpret them. Further applications to empirical social networks will probably reveal more classes of blockmodels in the near future.

In exploratory social network analysis, we are mainly interested in detecting the blockmodel which fits a particular network. The blockmodel tells us the general structure of the network and the equivalence classes which we find can be used as a variable in further statistical analysis. But we should issue a warning here. We will always find a best fitting blockmodel, even on a random network which is not supposed to contain a regular pattern. Therefore, we should restrict ourselves to blockmodels which are supported by theory or previous results. We should start out with a motivated hypothesis about the number and types of blocks in the network. As in other cases of exploratory network analysis, we should try to validate the result, e.g., by linking its classes to external data, such as actor attributes. If equivalence classes of actors have different properties, tasks, or attitudes, this corroborates the interpretation that the blockmodel identifies social roles or role sets.

Application

As we noted previously in this section, blockmodeling consists of three steps. In the first two steps, the image matrix is specified: the number of classes and the types of blocks or relations within and between classes which are allowed. Then, the computer completes the blockmodel by searching the partition of vertices into classes which matches the hypothesized image matrix best. If several image matrices are possible, it chooses the one which fits best. The error score shows how well the selected image matrix fits the network.

The blockmodeling commands of Pajek reflect these three steps. Before we discuss these commands, however, we must warn you that the method, like all optimization techniques, is time-consuming, so it should not be applied to networks with more than some hundreds of vertices, in which case the computer may need a full day to execute the command. For this reason the command is marked by a star in the menu.

In Pajek, there are two blockmodeling methods: one searches for the best fitting partition from scratch (*Random Start*), whereas the other only tries to improve an existing partition (*Optimize Partition*). Let us start with the latter method and apply it to the student government discussion network, using the formal positions within the government as the starting partition. Both files are available in the Pajek project file `student_government.paj`.

Operations>Blockmodeling >Optimize Partition

When you select the *Optimize Partition* command from the *Operations>Blockmodeling* submenu, the active partition specifies the number of equivalence classes you are looking for, which is the first step of blockmodeling. On selection of the command, a dialog box opens (Figure 17) showing two buttons. With the first button, you can select the type of equivalence or the image matrix of your blockmodel. We restrict our discussion to the standard types of equivalence (structural and regular) but you can also assemble your own blockmodel interactively (select option 3 - Define) or use an external command file (option 4 - Load MDL File). Select Regular Equivalence from the dialog box appearing when you press the top button. Then press the *Run* button to execute the command.

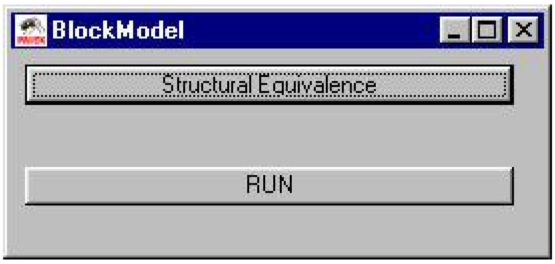


Figure 17 - *Optimize Partition* dialog box.

Pajek lists the adjacency matrix of the selected network in the Report screen, as well as the image matrix, the error matrix, and the error score of the initial partition. In our example, eight errors are reported which confirms our manual count (see Figure 15). Next, Pajek tries to improve the partition but it seems that the initial partition cannot be improved in this example. Finally, it creates the best fitting partition it has found and reports the image matrix, the final error matrix (see Figure 18), and the associated error score.

```
Image Matrix:

         1   2   3
     1   -  reg  -
     2   -  reg com
     3   -   -   -


Error Matrix:

           1    2    3
     1     2    0    0
     2     3    0    2
     3     0    1    0

Final error =       8.000
```

Figure 18 - Output of the *Optimize Partition* procedure.

Partitions>First Partition

Partitions>Second Partition

Partitions>Info>Cramer's V

In the student government network, the best fitting partition is equal to the initial partition according to formal position. You may check this by selecting the initial partition and the new partition as first and second partition in the *Partitions* menu and execute the *Partitions>Info>Cramer's V* command. Table 2 shows the cross-tabulation of the original (rows) and optimized partition (columns): they are identical because each vertex of the original first class is placed in the first class of the optimized partition, etc.

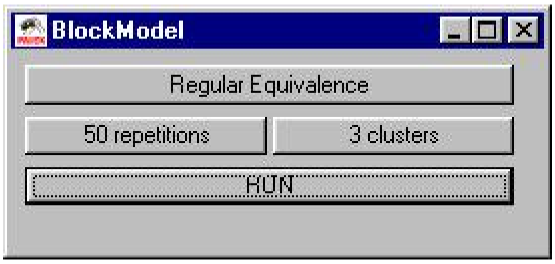
Table 2 - Cross-tabulation of initial and optimal partition.

	1	2	3	Total
1	3	0	0	3
2	0	7	0	7
3	0	0	1	1
Total	3	7	1	11

Because the *Optimize Partition* command is not able to improve the original partition, the formal roles seem to match the structural positions quite well. However, we do not know whether eight is a small or large error score for a network like this one and maybe another number of classes or other permitted block types would yield a better solution.

Operations>Blockmodeling >Random Start

The second method searches for the best fitting partition without taking into account an initial partition provided by the user of the program. Therefore, no initial partition is needed for the *Random Start* command. The dialog box displayed by this command offers the possibility to specify the number of classes (step 1), the kind of equivalence or blockmodel (step 2), and the number of repetitions (see Figure 19). Each repetition uses a new, random partition as a starting point in order to avoid settling on a local optimum.



Regular Equivalence

50 repetitions

3 clusters

Figure 19 - *Random Start* dialog box.

Applied to student government network, the *Random Start* command yields seven solutions with seven errors (under regular equivalence with three classes and hundreds of repetitions). This is a minimal improvement in comparison to the solution with the formal roles as equivalence classes and it has the disadvantage that a choice must be made among seven alternative solutions. Note that another number of classes and another type of equivalence may yield even better solutions, e.g., we find four errors in regular equivalence solutions with two classes but the interpretation is difficult: in one solution advisor2 is separated from the rest of the network, which seems to be a trivial solution, and in the other solution, advisors one and two are joined by minister4.Therefore, we prefer the original classification according to formal role within the student government.

We advise the following strategy for exploratory blockmodeling: (1) use the results of other analyses and theoretical considerations to assemble an image matrix; (2) try stricter blockmodels and block types first (structural equivalence is more strict than regular equivalence), and (3) try a smaller number of classes first. Select the blockmodel with the lowest error score, but if a model with a slightly higher error score yields a single solution which is easy to interpret, you should prefer the latter.

12.5 Summary

With this chapter, we conclude our course on social network analysis. The families of networks presented in previous parts of this book are reviewed once more: cohesive subgroups, core-periphery structures, and systems of ranks. We present a technique capable of detecting each of these structures, namely, blockmodeling.

In the case of blockmodeling, we need a new representation for networks: the matrix. The adjacency matrix of a network contains its structure: each vertex is represented by a row and a column and arcs are located in the cells of the matrix: the first row and column belong to the first vertex, the second row and column to the second vertex, etc. When sorted in the right way, the adjacency matrix offers visual clues on the structure of the network. Such a sorting is called a permutation of the network, which is actually a renumbering of the vertices.

Blockmodeling is not an easy technique to understand. Basically, this technique compares a social network to an ideal social network with particular structural features: a model. The researcher must suggest the model and the computer checks how well this model fits the actual data.

The model, which is called a blockmodel, contains two parts: a partition and an image matrix. The partition assigns the vertices of the network to classes, which are also called equivalence classes or positions. In the adjacency matrix of the network, the classes demarcate blocks: rectangles of cells. Blocks along the diagonal of the adjacency matrix contain relations within classes, while off-diagonal blocks represent relations between classes.

In the image matrix, which is the second part of a blockmodel, each cell represents a block of the adjacency matrix. It is a shrunk and simplified model of the adjacency matrix. If the vertices within a class are structurally similar – equivalent, we say – the blocks in the adjacency matrix have particular features: they are either empty, complete, or regular, which means that there is at least one relation from and to each vertex in a block. More types of blocks exist but we do not present them here.

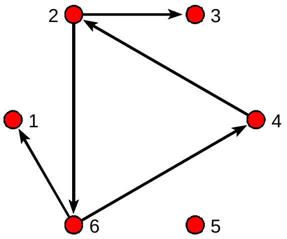
The image matrix shows which block types are allowed and, possibly, where to expect them. In addition, the distribution of non-empty blocks in the image matrix reveals the overall structure of the network. If the network contains cohesive groups, the non-empty blocks are found along the diagonal of the image matrix. If the network is dominated by a core-periphery structure, we find all non-empty blocks in one horizontal and one vertical strip in the image matrix. Finally, if there is a system of ranked clusters and the vertices are sorted according to their ranks, we find the non-empty blocks in the lower or upper half of the image matrix.

In exploratory blockmodeling, we search for the partition and image matrix which fit a social network best. Empirical social networks seldom match a blockmodel perfectly: arcs which should be present are absent or some absent arcs should be present. The number of errors expresses how well a blockmodel fits the network. This error score is used to evaluate different blockmodels for the same network.

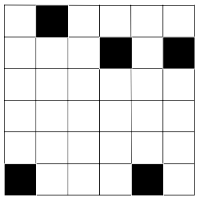
Blockmodeling is a powerful technique for analyzing rather dense networks but it needs the right input from the researcher to produce interesting results. The number of blockmodels which may be fitted to a social network is large, so it is not sensible to embark on blockmodeling without clear conceptions and expectations on the overall structure of the network. A researcher needs an informed hypothesis about network structure for a fruitful application of blockmodeling. In this sense, blockmodeling is used for hypothesis testing rather than exploration. Here, we reach the limits of the domain we want to cover in this book, so let us stop now.

12.6 Exercises

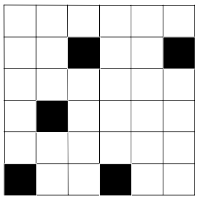
1 Below, one adjacency matrix and three sociograms are presented. Which adjacency matrix belongs to the sociogram?



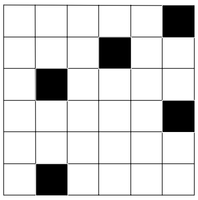
A



B



C



- a Matrix A.
- b Matrix B.
- c Matrix C.
- d Each matrix.

2 Which of the following statements is correct?

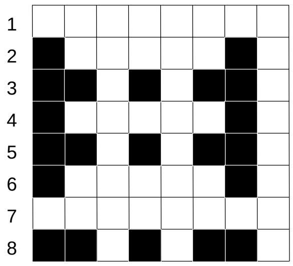
- a An adjacency matrix may contain more rows than columns.
- b An adjacency matrix is always symmetric with respect to the diagonal.
- c An incidence matrix may contain more columns than rows.
- d An incidence matrix is always symmetric with respect to the diagonal.

3 Of the three adjacency matrices in Exercise 1, which are isomorphic? It may help to draw the sociograms of the matrices.

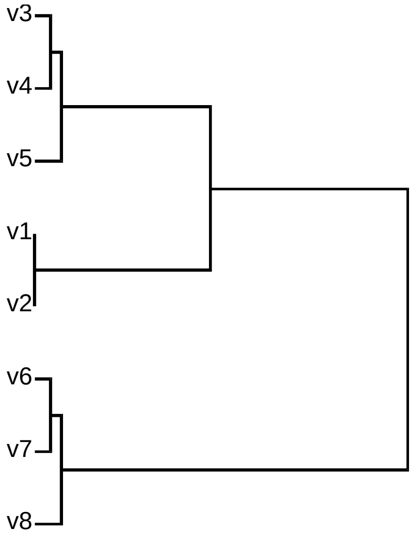
- a Matrices A and B.
- b Matrices A and C.
- c Matrices B and C.
- d All three matrices are isomorphic.

4 Write down the permutation which reorders one matrix of Exercise 1 into another matrix of that exercise.

5 According to the adjacency matrix below, which vertices are structural equivalent?



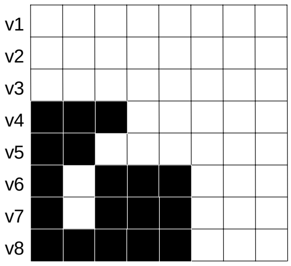
6 The dendrogram below displays the results of hierarchical clustering. Which equivalence classes would you make?



7 Which statement is correct?
  - a Regular equivalence allows for regular and empty blocks, but not complete blocks.
  - b Structural equivalence allows for complete and empty blocks, but no kind of regular blocks.
  - c Regular equivalence is a special case of structural equivalence.
  - d Structural equivalence is a special case of regular equivalence.

8 Assign the vertices of the adjacency matrix depicted at the right to a minimum number of regular equivalence classes.

9 What kind of structure does the adjacency matrix of exercise 8 represent?
  - a No particular structure.
  - b Cohesive subgroups.
  - c A core-periphery structure.
  - d A system of ranks.



v1								
v2								
v3								
v4	■	■	■					
v5	■	■						
v6	■		■	■	■			
v7	■		■	■	■			
v8	■	■	■	■	■			

12.7 Assignment

In Mexico, political power has been in the hands of a relatively small set of people who are connected by business relations, family ties, friendship, and membership of political institutions throughout most of the 20th century. A striking case in point is the succession of presidents, especially the nomination of the candidates for the presidential election. Since 1929, each new president was a secretary in the previous cabinet, which means that he worked closely together with the previous president. Moreover, the candidates always entertained close ties with former presidents and their closest collaborators. In this way, a political elite has maintained control over the country.

The network `mexican_power.net` contains the core of this political elite: the presidents and their closest collaborators. In this network, edges represent significant political, kinship, friendship, or business relations.

Notwithstanding the fact that one political party (the Partido Revolucionario Institucional) won all elections in the period under consideration, two (or more) groups within this party have been competing for power. The main opposition seems to be situated between civilians and members of the military (`mexican_military.clu`: the military in class 1 and civilians in class 2). After the revolution, the political elite was dominated by the military but gradually the civilians have assumed power. The partition `mexican_year.clu` specifies the first year (minus 1900) in which the actor occupied a significant governmental position. All data are available in the project file `mexican_power.paj`.

Draw the network into layers according to the year of 'accession to power' and use it to see when the civilians assume power. Use hierarchical clustering and blockmodeling to assess whether the political network consists of two (or more) cohesive subgroups and check whether these subgroups match the distinction between military and civilians or whether they cover a particular period.

12.8 Further reading

- For an introduction to matrices, see Chapter 3 in J. Scott (1991), *Social Network Analysis : A Handbook* (London [etc.]: Sage Publications) and Section 4.9 in S. Wasserman & K. Faust (1994), *Social Network Analysis: Methods and Applications* (Cambridge: Cambridge University Press).
- Blockmodeling is explained in Chapters 9, 10, and 12 in S. Wasserman & K. Faust (ibidem). For generalized blockmodeling, consult Doreian, Batagelj, & A. Ferligoj, 'Positional analyses of sociometric data' (in press).
- The example analyzed in the assignment is taken from J. Gil-Mendieta and S. Schmidt, 'The political network in Mexico' (in: *Social Networks* 18 (1996), 4: 355-381).

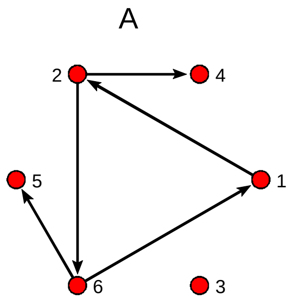
12.9 Answers

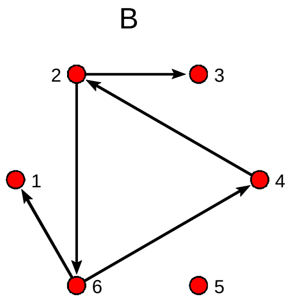
1 By convention, the first (top) row and the first (left) column represents the vertex with number one. Vertex two is identified by the second row and column, et cetera. The sociogram contains an arc from vertex 2 to vertex 3, so the cell at the intersection of the second row and the third column must contain an arc. In adjacency matrix B, this is the case so this is the only matrix which may represent the sociogram. Check the remaining arcs and black cells: they match. Answer b is correct.

2 Answer c is correct. In an incidence matrix, the rows represent actors and the columns contain the events to which the actors can be affiliated. The number of actors (rows) is not necessarily equal to the number of events (columns), so the number of columns may be larger than the number of rows (and vice

versa). In an adjacency matrix, the rows as well as the columns represent all vertices in the network, so their numbers must be equal (answer a is not correct). The adjacency matrix of an undirected network is always symmetric with respect to the diagonal, but this is not necessarily so in the case of a directed network (answer b is incorrect). Finally, it is a coincidence and not a necessity if for each actor *u* who is affiliated to event *v*, there would be an actor *v* who is affiliated to event *u*. Incidence matrices are seldom symmetric (answer d is incorrect).

3 In matrices A and B, three vertices do not send arcs (their rows are empty), one vertex sends one arc, and two vertices send two arcs. These matrices may describe isomorphic networks. The vertices of matrix C, however, have different outdegree: one vertex has zero outdegree and the remaining five have one outdegree. The network of matrix C cannot be isomorphic to the networks of matrices A and B. Are the networks of matrices A and B isomorphic? If we draw them, we can see that their structures are identical.





Answer a is correct.

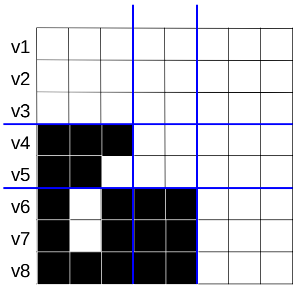
4 The permutation which transforms matrix A into matrix B can be read from the sociograms drawn in the answer to Exercise 3 (clockwise): vertex 2 remains vertex 2, 4 becomes 3, 1 becomes 4, 3 becomes 5, 6 remains 6, and 5 becomes 1.

5 Vertices with identical rows and columns are structural equivalent. In the adjacency matrix, vertices 1and 7 are structural equivalent, because their rows are empty and their columns contain six arcs send by the remaining six vertices. Vertices 2, 4, and 6 are also structural equivalent, and this is also true for vertices 3, 5, and 8.

6 In the first step, vertices v6, v7, and v8 must be separated from the rest. Then, vertices v1 and v2 can be split of from vertices v3, v4, and v5. At this stage, the dissimilarities between vertices within a cluster are low, so it is not sensible to subdivide the three equivalence classes further.

7 Answer d is correct. A complete block is also a regular block because each row and each column contains at least one entry (arc) within the block. Structural equivalence is a special type of regular equivalence (answer d). The reverse is not true (answer c is incorrect). Therefore, complete blocks are allowed under regular equivalence (answer a is incorrect) and a special type of regular block, namely, the complete block, is allowed under structural equivalence (answer b is incorrect).

8 A regular equivalence block is regular (at least one arc in each row and column) or empty. The rows of vertices v1, v2, and v3 are empty, so their horizontal blocks are null blocks. No other vertex has an empty row, so we cannot add a vertex to this cluster because we would get blocks in the rows of these vertices which are not empty and not regular because not every row contains an arc. For similar reasons, we may cluster vertices v6, v7, and v8: they have empty columns. If we cluster the remaining vertices v4 and v5, we obtain a solution with three regular equivalence classes: (1) v1, v2, and v3, (2) v4 and v5, and (3) v6, v7, and v8. In the adjacency matrix with blue lines separating classes, we can easily check that each block is either empty or regular.



v1								
v2								
v3								
v4	■	■	■					
v5	■	■						
v6	■		■	■	■			
v7	■		■	■	■			
v8	■	■	■	■	■			

9 Answer d is correct because all entries are situated at one side of the diagonal, which means that vertices consistently choose up (or down). It is clearly not a structure of cohesive subgroups or a core-periphery structure because the blocks on the diagonal are empty.