Manual for BLOCKS version 1.8

Tom A.B. Snijders

Krzysztof Nowicki *

June 23, 2007

Abstract

BLOCKS is a computer program that carries out the statistical estimation of models for stochastic blockmodeling according to Nowicki and Snijders (2001). This manual gives information about BLOCKS version 1.8 (June, 2007).

*We are grateful to Peter Boer for his cooperation in the development of the StOCNET and BLOCKS programs.
Authors' addresses: Tom Snijders, ICS, Department of Sociology, Grote Rozenstraat 31, 9712 TG Groningen, The Netherlands, t.a.b.snijders@ppsw.rug.nl; Krzysztof Nowicki, Department of Statistics, University of Lund, S-220 07 Lund, Sweden, krzysztof.nowicki@stat.lu.se .

Contents

1 BLOCKS

The program BLOCKS is designed for stochastic blockmodeling of relational data according to the methods described in Nowicki & Snijders (2001). This article extends the earlier work presented in Snijders and Nowicki (1997).

This manual gives information about the version, BLOCKS 1.8 (June 2007). The program was programmed in Turbo Pascal / Delphi by Tom Snijders, and Peter Boer made the transition to Delphi under Windows.

The program and this manual can be downloaded from the web site,
http://stat.gamma.rug.nl/snijders/socnet.htm/. The best way to run it is as part of the StOCNET program collection (Boer, Huisman, Snijders, & Zeggelink, 2003), which can be downloaded from http://stat.gamma.rug.nl/stocnet/. For the operation of StOCNET, the reader is referred to the corresponding manual. If desired, BLOCKS can be operated also independently of StOCNET.

This manual consists of two parts: the user's manual and the programmer's manual. There are two parallel pdf versions: blocks_man_s.pdf for screen viewing and blocks_man_p.pdf for printing. They were made with the LaTeX pdfscreen.sty package of C.V. Radhakrishan which made it possible, e.g., to insert various hyperlinks within the manual. Both versions can be viewed and printed with the Adobe Acrobat reader. This is the print version.

1.1 Differences with earlier versions

Version 1.8 (June 2007) has the following changes.
A somewhat longer lasting overdispersion is used when overdispersed burn-in is requested.
The `.in` file now requires a line with the random number seed, enabling the user to exactly reproduce results obtained.
There are some small improvements in input and output.

Version 1.7 (September 2006) outputs Pajek (Batagelj & Mrvar, 2002) files of the estimated coloring.

Version 1.6 (March 2004) differs from earlier versions in that the number of nodes now does not have a fixed limit, in the way to obtain good starting values for the latent classes, and in the choice of vertices differentiating well between the latent classes. In practice, the number of nodes is limited by the computer's memory and by numerical precision. The output will contain a message if there are risks of unsatisfactory precision.

Versions 1.51 – 1.53 (July 2002) differed only in minor aspects.

Part I
User's manual

The user's manual gives the information for using BLOCKS. It is advisable also to consult the user's manual of StOCNET because normally the user will operate BLOCKS from within StOCNET.

2 Stochastic blockmodeling

Blockmodels are relational structure models which allow one to represent pairwise relationships among social *actors* (individuals, organizations, countries, etc.). These relationships can be social interactions such as friendship, acquaintance, collaboration, information flow, etc., or combinations of such interactions. Various general introductions to blockmodeling can be found in the literature on social networks, e.g., Wasserman and Faust (1994). The actors will also be called the *vertices*, adhering to the usual graph-theoretic terminology.

In an extreme case, the actors can be grouped into classes in such a way that the classes determine the relational pattern: actors in the same class are tied to others in exactly the same way. In such a case, actors in the same class are said to be *structurally equivalent* (Lorrain and White, 1971). This is of course rare, and the literature contains various proposals to express some kind of approximate structural equivalence. (In addition, there is a different equivalence concept, *regular equivalence* not treated in this manual, and for which we refer the reader to the literature on social networks.)

A problem in modeling network data arises when it is unclear *a priori* which are the equivalent groups. Sometimes a researcher has ideas about attributes (gender, age, etc.) that might define approximately equivalent groups. But often the researcher does not wish to use prior ideas to form the groups, and would like to infer from the observed relations what groupings there are which best can be used to define the relational pattern. This is called *posterior blockmodeling*.

Stochastic blockmodeling (Holland, Laskey, and Leinhardt, 1983) is an approach to blockmodeling which, like other statistical models, assumes that the observed data are generated by some probabilistic, or (which is the same) stochastic, mechanism and which defines the equivalence within this stochastic model. Actors are *stochastically equivalent* if they have the *same probability distribution* of their relations to other actors. E.g., there could be three types of actors, A, B and C; actors of types A and B like to relate to actors of the other group (probability .5) but not the own group (probability .1), actors of type C like to relate to other Cs (probability .5), and the probability of ties between Cs on one hand and As or Bs on the other hand is rather low (probability .2). Type A then is one group of equivalent actors, so is B, and so is C. Because of the randomness of the resulting observed pattern, it may be hard to find out *a posteriori* who are the As, the Bs and the Cs.

The method implemented in BLOCKS tries to do this, using the following approach.

1. Since this is about *a posteriori* blockmodeling, only the observed relations are used as data.
2. The model is about *probabilities* of relations between the various groups, so it is assumed that there is random variability between the actually observed relations even if they are relations of stochastically equivalent actors. The main conclusions of the method are about the probabilities that actors belong to the same group, and about the probability distributions of the relations.
3. The unobserved attribute defining the groups is referred to as the "latent class" or also, arbitrarily, as the 'color'. Thus, in the example above, A might be blue, B green, and C purple. In the output, the colors are represented just by numbers (like 1, 2, 3). The number

of latent classes (or colors) is provisionally fixed by the user at one value or a sequence of values. E.g., the user can request to investigate for 2, 3, or 4 groups how a pattern with this number of groups would represent the observed network. Guidance is given about how to determine the best number of groups.

4. In most cases, nothing is known *a priori* about the classes. We then say that the colors are *unidentified*; instead of A being blue, B green, and C purple, it could be just as well to call A purple, B blue, and C green. This arbitrariness of the colors leads to some technical problems treated in Nowicki and Snijders (2001). A consequence is that we do not get results of the type "actor 1 is in group 4 with probability .8", because group 4 is not identified. Rather, results are of the type "actors 1 and 3 are in the same group with probability .8".

5. Sometimes we can say that we know *a priori* that some actors don't belong in the same group. More specifically, we could say that, e.g., actor number 4 is in group 1, actor 7 is in group 2, and actor 21 is in group 3. To be a bit more cautious we would say in the statistical model that the probability of actor 4 being in group 1 is .95, and similarly for the other two actors. Such an assumption would indeed identify the three groups, and we then say that we have an *identified model*. The program allows this but in practice, it is rare to specify an identified model.

6. To find the groups, the program uses random simulations, which is also called a *Markov chain Monte Carlo* (MCMC) approach. The particular Monte Carlo approach used is *Gibbs sampling*. One computation is called a Gibbs sequence. It is explained below what a Gibbs sequence is. The simulations are a bit time-consuming, so you have to be patient. The precision will be greater when you use more simulation iterations within each Gibbs sequence. The randomness implies that results will be different every time you run the program. If the stochastic blockmodel fits well, the differences will be very small. To check this, it is advised to let the program make 2 or 3 similar calculations (the technical expression is that you generate 2 or 3 independent Gibbs sequences) and see whether the results are close enough to each other.

7. In addition to analyse the standard dichotomous (on/off) type of relation, it is also possible to analyse relations with a larger numer of values. Ordinarily it is not advisable to analyse relations with more than about 5 different values, however. If you have relations with more values, it is better to recode them to a smaller set of values.

8. Since reciprocation is a basic phenomenon in social networks, the method uses the *dyad*, i.e., the two-way set of relations between two actors, as the basic relational unit.
   For this purpose the data are recoded, using code numbers 1, 2, etc. These new codes are called the *new alphabet*. E.g., consider two actors i and j, the relation from i to j being denoted y_{ij} and the opposite relation denoted y_{ji} .
   The code for the relation from i to j also depends on the relation from j to i. What is recoded is really the dyad, i.e., the pair (y_{ij}, y_{ji}). If the relation is originally dichotomous, represented by 0 and 1, four code numbers 1, 2, 3, and 4 are required. Symmetric relations $y_{ij} = y_{ji}$ will receive code 1 for $y_{ij} = y_{ji} = 0$ and code 2 for $y_{ij} = y_{ji} = 1$. Asymmetric relations will receive codes 3 and 4: if $y_{ij} = 0$ and $y_{ji} = 1$ then y_{ij} will be recoded to 3 and y_{ji} to 4. If the relation is dichotomous and two-sided (symmetric), then the codes 3 and 4 are not used.
   If the original relation is not dichotomous but has more than 2 values, more codes will be required. The recoding is mentioned in the output. This leads to a lot of code numbers, but this representation by dyads gives a more faithful representation of the relational structure. It may be noted that in recoded form, it is sufficient to know either the upper-diagonal or the lower-diagonal half of the adjacency matrix, since one half determines the other half.

9. Missing data are allowed; the user just needs to indicate what is the code used for the missings in the original data file. If there are missing data, then each whole dyad is considered either missing or non-missing.
10. As a final result of the method, the program tries to get beyond the conclusions about the probabilities that pairs of actors are in the same group, and actually tries to find groups of actors who (probably) are in the same group. But it may be necessary to disregard some of the actors, of whom the group identification is too ambiguous.

2.1 The Bayesian approach

The stochastic blockmodeling method in BLOCKS is a Bayesian statistical method. In the Bayesian approach, all uncertainty about the conclusions of the statistical inference is expressed in probability statements about the model parameters. The main component of the stochastic blockmodel (one might say: the main unknown parameter) is called the *partition* or *coloration* of the set of actors, or the *configuration of the class structure*; this is the division of the set of all actors into subsets of stochastically equivalent actors. These subsets are also called *latent classes*. This coloration is *a priori* unknown, and the purpose of the blockmodeling exercise is to get to know it. But it will not become known with certainty; in the Bayesian approach, the uncertainty is expressed by saying that we get to know the *probability distribution* of the coloration, more precisely, the posterior probability distribution of the coloration given the network data.

2.2 The Gibbs sequence

This posterior distribution is estimated on the basis of a so-called *Gibbs sequence*. This is a sequence of colorations, constructed by a chance mechanism devised so that, whatever the starting point, the coloration eventually will approximate a sample from the posterior distribution of the coloration given the network data. The number of colors (latent classes) is constant within one Gibbs sequence (although in some steps there might be zero vertices in some of the classes). Every step in the sequence is also called an iteration. When you run BLOCKS, the screen will show the generated colorations by patterns of colored squares (the coloration is not shown for every step in the sequence but only one in so many steps, because showing it for each step would make you dizzy and the program slow). Since the sequentially generated colorations are (approximately) a sample from the distribution, the probabilities can be inferred from averages of a long sequence.

The Gibbs sequence will need a start-up (some mechanics say a 'burn-in') period before it has indeed converged to a sample from the posterior distribution. The problem is that there are no definite guidelines about how long this start-up period needs to be. Therefore it is best to take it pretty long. The user must specify the number of iterations for the start-up part of the sequence; after this number of iterations it is assumed that convergence has taken place; the user also specifies the number of iterations after this point, the so-called post-convergence iterations. The parameters of the posterior distribution are estimated as averages over the post-convergence iterations, so-called *posterior means*. In most cases, 10,000 to 100,000 iterations should suffice both for the pre-convergence and the post-convergence number of iterations. If you are unsure whether you had enough iterations, take the preferred number of latent classes and make a real long Gibbs sequence to check that it gives the same answers as those obtained with a shorter sequence.

In one execution of BLOCKS, several Gibbs sequences can be constructed, because the number of colors can assume several values from a minimum to a maximum value, and for each color it is possible to make more than one Gibbs sequence. The several Gibbs sequences made for the same color are replications of one another; if they yield approximately the same results, this gives confidence in the outcomes.

2.3 The number of classes

After running BLOCKS for several values of the number of classes, or colors, say, 2 to 5, you have to decide what is the best number of classes. This is a matter of fit and interpretability. For the fit, two statistics are offered.

1. The extra *information* contained in observing the relations, if you would already know the colors of the vertices. This is measured by the parameter I_y in Nowicki and Snijders (2001). It is 0 if, the relation between any pair of vertices is fully determined by the two classes to which they belong. The larger the information, the less the colors tell about the relations between the vertices.
2. The *clarity* of the block structure. This is measured by the parameter H_x in Nowicki and Snijders (2001). It is 0 if, for each pair of vertices, it is certain whether they have the same color. It is 1 if every pair of vertices has a probability of .5 to have the same color – which means there really is no block structure at all.

Both of these parameters are estimated in BLOCKS by their posterior means (like anything in the Bayesian approach).

To choose the best number of classes, you compare the values of these statistics across the numbers of classes, and the number of classes for which these parameters are smallest has the best fit. It is quite common that both parameters lead to different conclusions, e.g., H_x might be smallest for 3 and the information I_y for 5 classes. The interpretability of the results then is the main consideration. Tentatively, we suggest that H_x gives more important indications than the information I_y.

It may be helpful to give the mathematical formula for H_x. This is based on the matrix of probabilities that two vertices have the same color, denoted π_{ij} . H_x is defined by

$$H_x = \frac{4}{n(n-1)} \sum_{i \neq j} \pi_{ij}(1 - \pi_{ij}) \ .$$

The block structure is presented in the output by the matrix of the π_{ij}, called in the output the "Matrix of pairwise color equality". Like other matrices of posterior probabilities, the probabilities are indicated by the first digit after the decimal point, e.g., a 4 denotes a value at least 0.4 and less than 0.5. The value 9 denotes a value at least 0.9. If there is an extremely clear-cut block structure, then for all pairs of vertices the digit presented is either 0 – for vertex pairs of different colors, $\pi_{ij} < 0.1$ – , or 9 – vertex pairs of the same color, $\pi_{ij} \geq 0.9$. The formula shows that in this case, H_x would be very close to 0. At the other extreme, if there is no block structure at all then all π_{ij} would be close to 0.5, so all digits in the matrix would be 4 or 5, and H_x would be almost 1.

2.4 Estimated probabilities

The stochastic blockmodel has two important parts: the division of the set of actors into latent classes (the coloration), and the probability distribution of the relations within and between these classes – which is analogous to what is called the *image matrix* in deterministic blockmodeling procedures (see, e.g., Wasserman & Faust, 1994). The nonidentifiability mentioned in point 4 of Section 2 poses problems here: since we cannot meaningfully talk about "class 1" and "class 2", we also cannot talk about "the relations between actors in classes 1 and 2". There are several ways out of this.

1. The method which stays closest with the stochastic blockmodel is to give conclusions about the fitted probability distributions of relations between the vertices, rather than the classes. This method is treated in Nowicki and Snijders (2001). The colorations are averaged out in

this approach. The output of BLOCKS gives the fitted probabilities for all relations in the new 'alphabet'.

2. Another method is to reduce the set of vertices to a smaller set for which it is pretty clear how it should be partitioned into classes – i.e., a reduction to a subset of vertices for which the coloration is not highly chance-dependent any more. This is done by throwing out the vertices for which it is not clear what is the set of other vertices with whom they tend to be in the same class.
   In the output, this is referred to as "finding strictly separated classes". The current version of BLOCKS gives the fitted probabilities as well as the observed frequencies of relations within and between these classes, because this comes closest to the well-known practice of constructing an image matrix.

3. A third method is, if there are c classes, to choose c or $c-1$ vertices that have a very small probability to be in the same class. If for $c = 3$ these are, e.g., vertices 1, 4, and 11, then one could define color 1, 2, 3 as the colors of vertices 1, 4, and 11, respectively, under the condition that these vertices must have different colors.
   The disadvantage of this method is that it may give too much influence to the vertices with an ambivalent position.

3 Operation of the program

The program performs Gibbs sampling (single or multiple sequences). As explained in the preceding section, each sequence has the following structure. First initial, 'pre-convergence' iterations are performed. Then it is assumed that convergence has taken place and the post-convergence iterations are used for calculating posterior means and standard deviations. You see on the screen the iteration number and the coloration which changes continuously during the simulations.

The program can be interrupted in two ways. A complete stop is requested when you press the Stop button. During the post-convergence iterations, you may restart the post-convergence iterations in the current Gibbs sequence by pressing the Restart button. This can be done when you doubt that the Gibbs sampler indeed converged by the moment that the indicated pre-convergence number of iterations had transpired – but usually this will not be necessary.

The program writes the output to two files, as described below. What you seen on the screen during the program's operation is mainly for fun and to be convinced that something is happening. You also see how far the calculations have progressed. For the results the screen doesn't tell you much, you have to look in the output file.

4 Project

The *project* consists of data and the specified options, possibly together with an output file. The project is defined by a so-called *basic information file*. The project is identified by a *project name*, which is the root name of the basic information file and the output files. The extension names are `.in` for the basic information (input) file, `.out` for the main output file, and `.pqr` for the secondary output file. E.g., if the project name is `block`, then the basic information file name is `block.in`, and the output files produced have names `block.out` and `block.pqr`. The basic information file is written and shown by StOCNET but it can also be read using any text editor which can handle ASCII (text) data. The output files are shown by StOCNET but can also be read by any text editor.

5 Input data

The *name* of the data file and the path must not contain blanks. Only characters with ASCII numbers from 43 to 126 are allowed in the file name; this implies that all letters `a - z`, `A - Z`, digits `0 - 9`, and various characters including `.` and `~` and _ are permitted, but not some other characters of which `¨ % & ( ) ! # $ *` are examples. This applies to the entire path, so e.g. "Program Files" (with blank!) should not be used.

The input data file itself consists of a square adjacency matrix (see the example data files). Values in each line must be separated by blanks. The values in the adjacency matrix are integers from $-cc$ to $+cc$, where cc is a value defined in the unit *B_CONS* (see the programmer's manual), currently $cc = 9$. This allows to analyse relations which have more detail than the usual dichotomous ("on/off") relations. The values are treated as unordered categories. The diagonal values of the square matrix must be present in the data file, but their values are disregarded by the program.

A missing value indicator must also be specified. Relations equal to this missing value indicator are treated as missing data. If the relation from some actor i to another actor j is missing, then the reverse relation (from j to i) also is disregarded (this is because the dyads are the unit of analysis). If you have no missing data, use any value for the missing data indicator which is not present in the data set. This missing value indicator may be outside the permitted data range ($-cc$ to $+cc$) – this is interpreted, of course, as a complete absence of missing data.

For internal use in the program, the data first are recoded to a new "alphabet". This is described in point 8 of Section 2 above. This alphabet is represented by numbers 1 to r where $r = r_0 + 2\,r_1$, r_0 being the number of symmetric relations, and $2\,r_1$ being the number of asymmetric relations. As a result of the recoding, the upper diagonal part of the recoded matrix defines the lower diagonal part of the recoded matrix (and vice versa). The recoded values are used internally in the program and presented in the output.

6 Output files

Two output files are produced: *pname*`.out` and *pname*`.pqr`, where *pname* is the project name. Both are ASCII files which can be read by any text editor. The first is the main output file and presented in StOCNET. You only need to take a look at the second output file if you want to have extra information about some secondary results mentioned in Nowicki and Snijders (2001), or if you desire greater precision in the form of an extra decimal or standard errors.

The output is meant to be self-explanatory, given that you understand the statistical method. The output is divided into sections using the symbol @ followed by a number. The number indicates the sectioning level. E.g., major sections (prologue, output for a given number of colors) start with @1, subsections (output of each of the multiple Gibbs sequences) start with @2, subsubsections with @3, etc. (up to @6). Output produced by a new run of BLOCKS starts with @0. You can use this symbol for dividing the output into pages or skipping to the next (sub)section.

In addition, an output file *pname*`_r.paj` is created, which contains Pajek (Batagelj & Mrvar, 2002) files of the network, with the post-hoc obtained colorings as partitions of the node set. This file is overwritten at each new run of BLOCKS!

7 Options

The options can be specified in StOCNET (or by editing the basic information file). The main options are as follows.

1. The data set to be analysed.
2. The number of latent classes (or colors): indicate the minimum and maximum number of latent classes for which you wish to get an analysis. (You may choose these to be equal.)
3. Number of iterations.
   In Section 2.2, it is described that the Gibbs sequence consists of two parts. You have to specify the number of iterations for the before-convergence and also for the after-convergence part. Values between 10,000 and 100,000 usually are reasonable.
4. Number of Gibbs sequences.
   As described in Section 2, it is advised to run more than one Gibbs sequence, as replications of one another, to check if the results are stable. (If they are not, increasing the number of iterations may help.) It is advised to have 3 Gibbs sequences, but in preliminary explorations you could use just 1.
5. Identification.
   It was mentioned in Section 2, and it is elaborated below in Section 7.2, that the latent classes are not identified: in most cases, you cannot say "actor 3 is in class 1" although you can say "actors 3 and 4 are in the same class". This default situation is indicated by the option "no identification". However, in some situations there is *a priori* information that can be used for identification; see Section 7.2.
6. If within one project you are running BLOCKS repeatedly, a question is whether the output produced earlier still is valuable. You have the choice between letting the newly produced output overwrite the old output, or appending the new output to the old output.

7.1 Advanced options

In Section 5.2, Nowicki and Snijders (2001) mentions three options for improving convergence.

1. Start with a good ("optimal") configuration of the latent classes. If this is not selected, then the initial colorations are determined randomly.
2. Use overdispersed class configurations in the pre-convergence iterations.
3. Use overdispersed probabilities in the pre-convergence iterations.

The two overdispersion options lead to better "mixing" in the pre-convergence runs, which is intended to bring the Gibbs sequence more quickly to a good class configuration.

The user can choose to employ these options or to leave them out. The default is that these options are "on" (they will usually not have much influence, but they may help and they won't harm).

Another advanced option is the value of the *concentration parameter*. This defines the parameter T mentioned in Section 5 of Nowicki and Snijders (2001): the value of T is the concentration parameter multiplied by the number of classes. The higher the concentration parameter, the more preference there is for equally sized classes. If the concentration parameter is too low, there is a risk that some of the classes will be so small that they do not contain any actors at all. The advice is to choose a concentration parameter of 100, as a good medium value. This is the default value.

Finally, the random number seed can be determined by the user, which enables the user to reproduce results exactly.

7.2 Identified and non-identified models

The user makes the choice between either working with a model that is non-identified (because the color labels are arbitrarily ordered) or working with a model that is identified by prior probabilities for the colors of some of the vertices.

There are two ways of specifying a model identification. The easiest way is to give, for each color, a vertex that has this color with a high probability. If there are c colors, there can be given $c-1$ or c vertices that have a high prior probability of having a specific color. Specifically, this high prior probability is taken as 0.95. The main interpretation is that the prior probability that two or more of this set of color-identifying vertices have *the same* color, is very low. (The prior probabilities are not chosen as 1.0, in order to leave open the possibility that the data override the prior ideas.)

The more complicated way to identify the model is by giving prior color probabilities for a set of vertices in the form of a $k \times c$ matrix of prior probabilities, where k is the number of identifying vertices, which must be equal to c or $c-1$, and c is the number of colors. This matrix has to be given in the form of a text file containing a rectangular data matrix, where the numbers are separated by blanks, all numbers are nonnegative, and all row sums are 1.0.

If you choose an identified model and also the option of starting with an "optimal" set of initial colorings, then it is possible that the "optimal" colorings do not correspond to the prior probabilities for the colors. Therefore, if you wish to use prior probabilities, choose random instead of "optimal" initial colorings.

8 Getting started

Reading the user's manual helps to get started, especially Section 2 which explains the basic concepts of the operation and interpretation of this method.

It is best to start working with BLOCKS with a dichotomous adjacency matrix, corresponding to the usual representation of a social network as a (directed or ubdirected) graph. Take a network with no more than 50 actors to start with. As indicated in Section 5, the data must be presented in the form of a square data matrix in ASCII (text) format, the entries in the columns being separated by spaces. Choose 3 latent classes (i.e., the minimum and the maximum number of latent classes both are 3), only one Gibbs sequence, and let there be 10,000 iterations before as well as after convergence. Select the default "no identification" option.

With this specification, let the program run. It won't take very long, and the screen shows a bar which indicates how far the program is from finishing. After finishing, StOCNET takes you automatically to the output file. Given your knowledge of Section 2, the output should be understandable. The next section walks you through an example output file, to assist you with the interpretation.

After having understood operating and interpreting the program for one choice of the number of latent classes, select a range from 2 to 4 latent classes, and run the program again. Now you will have to use Section 2.3 to decide on the most appropriate number of latent classes.

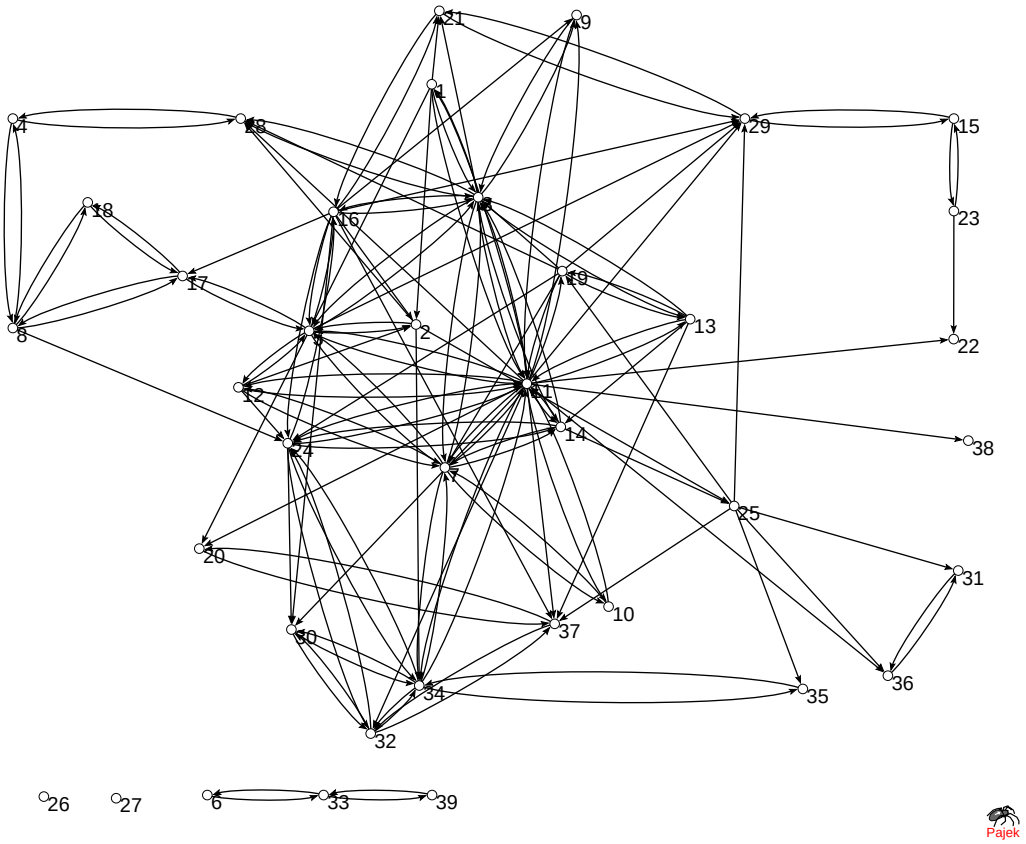
Finally, do the same thing for 3 Gibbs sequences. If, for a given number of colors, the results of the three sequences are similar, it can be concluded that the program converged to a good solution. If not, increase the number of iterations and try again. To compare the similarity of the 3 sequences, look mainly at the information and the clarity of the block structure expressed in H_x (see Section 2.3), and at the matrix of pairwise color equality.

9 Example: Kapferer's Tailor Shop

Block modelling is illustrated below by walking you through the output file using a data set collected by Kapferer. He observed interactions in a tailor shop in Zambia over a period of ten months, focusing on the changing patterns of alliance among $n = 39$ workers during extended negotiations for higher wages. Data were recorded on two ocassions between which an abortive strike occurred and they are available in Ucinet 5 (Borgatti, Everett, and Freeman, 1998). A second successful strike occurred just after the second data set was collected.

In the example below, Kapferer's data relating to friendship interactions (a directed relation), for the second occasion, are analyzed using BLOCKS. Kapferer's friendship and assistance data together, for the first occasion, are analyzed in Nowicki and Snijders (2001). The rank orders of the vertices in the *Ucinet* data set are used as vertex labels. The network is represented in the graph below, which was drawn using `Pajek` (Batagelj & Mrvar, 2002).

In the original data set, the employees had been placed in the following occupational categories in the order of their prestige: head tailor (worker number 19), cutter $\{16\}$, line 1 tailor $\{1-3, 5-7, 9, 11-14, 21, 24\}$, button machiner $\{25-26\}$, line 3 tailor $\{8, 15, 20, 22-23, 27-28\}$, ironer $\{29, 33, 39\}$ and cotton boy $\{30-32, 34-38\}$. An additional category was line 2 tailor, which included employees $\{4, 10, 17-18\}$.



9.1 Annotated output file

We present parts of the output file obtained for 3 colors. Note that since the MCMC analysis depends on stochastic simulation, the results will differ slightly between runs. After the header and a report about the chosen options, the report of the data input is as follows.

```
@1
Input of data file
******************
Data read from file Kapff2.dat and reproduced on file kap2.PQR.
39 vertices

Set of symmetric relations is
 1 (  0,  0)   ( 646 dyads)
 2 (  1,  1)   (  52 dyads)
Set of asymmetric relations is
 3 (  0,  1)   (  43 dyads)
and its mirrored counterpart
 4 (  1,  0)   (  43 dyads)
Total number of relations is 4.
Number of dyads with observed values is 741.
Number of dyads with missing  values is 0.

Information calculated with natural logarithms.
Entropy of empirical dyad distribution =    0.47125
```

This gives the report of the recoding: the new alphabet is 1, 2, 3, 4, where
1 refers to a null dyad
2 refers to a symmetric tie
3 refers to the (0,1) asymmetric dyad,
4 refers to the (1,0) asymmetric dyad.
Since (0,1) and (1,0) dyads are each others' mirror image, there are as many (0,1) as (1,0) dyads. The total number of dyads is $\binom{39}{2} = 741 = 646 + 52 + 43$.

The entropy of the empirical dyad distribution, 0.47125, is what would be obtained for the information I_y in the observations, if there were only 1 color, cf. formula (24) in Nowicki and Snijders (2001). This value serves as a reference for the information values obtained later for more than one color.

Some lines later follows the report of the colorations obtained in some of the iterations in the Gibbs sequence. The colors for the 39 vertices are shown, together with the average information in the observations for the colorations in the last so many (here: 400) iterations.

It can be seen from the information values that the warming up takes place during approximately the first 6,000 iterations: the information decreases from values higher than 0.5 to values mostly between 0.40 and 0.41. Approximate stability of the information values and the colorations seems to set in about iteration 6,000. Between iterations 2800 and 3200 we see a change in the color labels 1, 2, 3, illustrating the arbitrariness (non-identification) of the labels.

```
@3
Iterations: values of some random variables
===========================================

Before convergence is assumed
iter., Info. y, last x:
   400  0.5564   111113111111111111111111111111113111113
```

```
  800  0.4956   12222322221112212322222223222223222222
 1200  0.4768   22313131223222121121111221211111211111
 1600  0.4452   31323231113333231231122332211121231112
 2000  0.4444   22131323331121313313333123333333333333
 2400  0.4419   12232321332111321113333221333333333333
 2800  0.4213   11131313312111313313333113333333133333
 3200  0.4287   22212121223222121121122221111111111111
 3600  0.4232   33313131332333131131111331111111311111
 4000  0.4197   33212131332333123123311331133311311311
 4400  0.4200   33313131132333131131111331111111311111
 4800  0.4198   33313131332333131131111331111111111111
 5200  0.4164   33313131112333131131111331111111111111
 5600  0.4167   33313131332333131131111331111111111111
 6000  0.4148   33313131332333131131111331111111311111
 6400  0.4126   33313131332333131131111331111111311111
 6800  0.4136   33313131112333131131111331111111311111
 7200  0.4093   33212121312333131121111231111111111111
 7600  0.4094   33212121112333131131111231111111111111
 8000  0.4105   33212121332333131321111131111111111111
 8400  0.4099   33212121312333131131111231111111111111
 8800  0.4082   33212121132333131131111231111111111111
 9200  0.4072   33313131112333121131111331111111111111
 9600  0.4052   33212121312323131131111231111111311111
10000  0.4016   32212121112333131131111231111111111111
10400  0.4056   33313131112323121131111221111111311111
10800  0.4070   33212121132333131131111231111111111111
11200  0.4095   33313131312333131131111331111111311111
11600  0.4080   33212111312323131131111231111111111111
12000  0.4042   33212121112333121131111231111111211131
12400  0.4092   33313131312333131131111331111131311111
12800  0.4067   33313131312333131131111331111111111111
13200  0.4087   33313131332333131131111331111111111111
13600  0.4095   33313131312333131131111331111111311111
14000  0.4093   33313131312333131131131331111111111111
14400  0.4133   33313131112333131131113331111111331111
14800  0.4100   33313131132333131131111331111111111111
15200  0.4103   33313131112133131131111331111111311111
15600  0.4092   33313131132333131131111331111111111111
16000  0.4124   33313131312333131131111331111111311111
16400  0.4105   33212131112333121131111321111111311111
16800  0.4090   33313131332333131131111331111111311111
17200  0.4090   33313131112333131131111331111111311111
17600  0.4068   33313131312333131131111331111111111111
18000  0.4042   33212121332333131131111231111111331111
18400  0.4105   33212121112333131131111231111111111111
18800  0.4073   33313131112333131131111331111111111111
19200  0.4034   33313131312333131131111331111111311111
19600  0.4061   33212121112333131131111231111111211111
20000  0.4100   33313131332333131131111331111131331111

20000 runs before convergence was assumed.

After convergence is assumed
iteration i, Information y, last x:
```

```
   400  0.4090   333131311323331311311113311111111111111
   800  0.4084   333131313323331311311113311111111311111
  1200  0.4048   333131313123331311311113311111111111111
  1600  0.4078   333131313323331311311113311111111311111
  2000  0.4105   333131313323331311311113311111111111111
  2400  0.4070   333131311323331311311113311111111111111
  2800  0.4063   333131311323331211311113311111111111111
  3200  0.4083   332121213323331311311112311111111311111
  3600  0.4060   332121211123331211311113311111111111111
  4000  0.4101   332121211323331211311112311111111111111
  4400  0.4111   222121211132221211211112211111111111111
  4800  0.4065   222121212232221211211112211111111211211
  5200  0.4050   223131311132221211211113211111111311111
  5600  0.4044   222121211232221211211112211111111211111
  6000  0.4090   222121212132221211211112211111111211111
  6400  0.4085   222121211132221211211112211111111111111
  6800  0.4084   222121212232221211211112211111111211111
  7200  0.4089   222121212232221211211112211111111221111
  7600  0.4095   222121211232221211211112211111111111111
  8000  0.4069   222121211132221211211112211111111211111
  8400  0.4075   223131312132221211211112211111111111111
  8800  0.4135   222121211232221211211112211111111211111
  9200  0.4109   222121212232221211211112211111111211111
  9600  0.4123   222121211132221211211112211111111211111
 10000  0.4080   223131311132221211211112211111111111111
 10400  0.4079   222121211132221211211112211111111211111
 10800  0.4137   222121211132221311211112211211111111111
 11200  0.4096   222121212232221311211212211111111111111
 11600  0.4088   223131312232221221211112211211111111111
 12000  0.4123   333131311133331311322112311122121211211
 12400  0.4082   333131311123331311311113311111111111111
 12800  0.4081   333131311123331311311113311111111311111
 13200  0.4064   333131313323331311311113311111111111111
 13600  0.4056   333131313123331311311113311111111111111
 14000  0.4101   333131311123331311311113311111111311111
 14400  0.4090   333131311123331311311113311111111311111
 14800  0.4091   333131313123331311311113311111111111111
 15200  0.4067   333131313323331311311113311111111311111
 15600  0.4116   332121313323331211331113211313131211311
 16000  0.4098   333131311123331311311113311111111311111
 16400  0.4111   333131313323331311313113311111111311111
 16800  0.4067   332121213123331311311112311111111111111
 17200  0.4037   332121213323331311311112311111111111111
 17600  0.4062   332121313123331311311112311113111311111
 18000  0.4134   333131313323331311311113311111111311111
 18400  0.4099   333131313123331211313113311111111111111
 18800  0.4089   333131311123331311311113311111111111111
 19200  0.4053   332121311123331311311112311111111311111
 19600  0.4085   333131311323331311311113311111111311111
 20000  0.4051   332121313123331311311113311111111111111
After assuming convergence, 20000 runs for estimating posterior distribution.
```

After this point come the results. (Incidentally, we see several switches between color labels 2 and 3.)

```
@3
Results
=======

Information y :   0.4084
A new ranking of the vertices was determined to bring out the block structure.
New ranking is
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
                                       . . . 28 29 30 31 32 33 34 35 36 37 38 39
 1  2  3  5  7 12 13 14 16 19 24 25  4  6  8  9 10 15 17 18 20 21 22 23 26 27 28
                                       . . . 29 30 31 32 33 34 35 36 37 38 39 11
```

The new ranking is based on the matrix of pairwise color equality (which follows below), and is determined by a heuristic that tries to obtain a clearly visible block structure. To evaluate this new order visually, the program presents the adjacency matix (in the new codes) using this vertex order. The first two lines for the next matrix (and all presented square matrices below) give the original vertex numbers, with the first line presenting the factors of 10 and the second line the units.

```
Recoded observed adjacency matrix in new ranking is as follows:
         1111122    11112222222233333333331
    12357234694546890578012367890123456789 1
   1   42411111111111111111141111111111111112
   2  3 12131131141111111111111114111114111111
   3  21 22122231111121111141112111111111112
   5  322 22112111111111214111111411111111112
   7  1122 21213111111211111111111111112111112
  12  14122 11114111111111111111111111111111112
  13  112111 41211111111111111111111111114112
  14  1121213 11241111111111111111111111111112
  16  14221111 12111141141121111141111114111
  19  114141211 43111111111111114411111111112
  24  1111131223 11131111111111111141212111112
  25  13111113141 11111111111111414111444111
   4  111111111111 12111111111112111111111111
   6  1111111111111 11111111111111121111111
   8  11111111114121 11122111111111111111111
   9  112111113111111 11111111111111111111112
  10  1111211111111111 11111111111111111112
  15  11111111111111111 11111211121111111111
  17  111211113111112111 21111111111111111111
  18  1111111111111121112 11111111111111111111
  20  11131111111111111111 11111111111112113
  21  3131111121111111111111 11111211111111111
  22  111111111111111111111 3111111111111113
  23  1111111111111111211114 111111111111111
  26  11111111111111111111111 11111111111111
  27  111111111111111111111111 1111111111111
  28  13211111131121111111111111 111111111113
  29  11131111331311111211211111 11111111113
  30  1111111131311111111111111111 1212111113
  31  11111111113111111111111111111 111121111
  32  111111111121111111111111111121 12112113
  33  1111111111111211111111111111111 1111121
```

```
34  131121111121111111111111111112121 211112
35  111111111131111111111111111111112 11111
36  111111111131111111111111111121111 1113
37  111111313113111111121111111121111 113
38  111111111111111111111111111111111111 13
39  111111111111111111111111111111211111 1
11  212222221221111221114141114441414214441
```

The block structure shows that vertices 1 – 3, 5, 7, 12 – 14, 16, 19, 24, 25, tend to belong to the same class, (now labeled group 1), and vertices 4, 6, 8–10, 15, 17, 18, 20– 23, 26 – 33, 35 – 39, and (less evidently) 34 also belong to one class (group 2). Vertex 11 is a class of its own: group 3.

Comparing this to the occupational categories mentioned earlier, it can be concluded that the a posteriori classes obtained by our analysis have clearly different occupational profiles. Group 1 consists of high prestige workers while group 2 mainly those with lower prestige.

We use the word 'group' rather than 'class' to indicate the provisional groups of vertices meant to approximate the latent classes. In statistical terminology, we could say that the groups are estimates for the classes.

The adjacency matrix above shows that group 1 is especially cohesive: many codes 2, 3, and 4, corresponding to asymmetric and symmetric ties. Group 2 has few relations in general: many codes 1, both within group 2 and with group 1. Group 3, the single vertex 11, is involved in many ties; this vertex has relatively many symmetric ties (code 2) with group 1 and sends many unreciprocated ties (code 4) to group 2. This visual inspection is elaborated in the following matrices. In all the matrices of probabilities, like the following one, the probabilities are represented by their first decimal. Thus, values p in the range $0.0 \leq p < 0.1$ are represented by 0, the range $0.1 \leq p < 0.2$ by 1, etc., and the range $0.9 \leq p \leq 1.0$ by 9.

The following symmetric matrix represents the estimated values for π_{ij}, the probabilities that two vertices i and j have the same color. In a well fitting blockmodel, if the new ranking is appropriate, this matrix should show a block structure with diagonal blocks of high values (preferably 8 and 9) and low non-diagonal values (preferably 0 and 1). The following matrix indeed conforms to this pattern, with the 3 groups indicated above, except that actors 9, 10, and 34 do not fit very well in their groups. They are outliers that could belong to group 1 or group 2 but do not conform well to either of those groups.

```
@4
Matrix of pairwise color equality
=================================
           1111122    11112222222233333333331
      12357234694546890578012367890123456789 1
   1   9677999898900043000000000000004000000
   2  9 67799989890004300001000000000004000000
   3  66 99677778700032000000000000004000004
   5  779 97777797000320000000000000004000003
   7  7799 787779700032000000000000004000003
  12  99677 99898900043000010000000004000000
  13  997789 98989000430000000000000004000000
  14  9977799 898900043000010000100004000000
  16  88777888 879000430000000000000004000002
  19  997779998 89000430000000000000004000000
  24  8889988878 80003300001000000000004000002
  25  99777999998 00043000000000000004000001
   4  000000000000 99679999999999999995999990
   6  0000000000009 96799999999999999995999990
   8  00000000000099 669999999999999995999990
   9  443334444434666 6666666666666665666660
  10  3322233333337766 7777677677777775777770
  15  00000000000099967 999999999999995999990
  17  000000000000999679 99999999999995999990
  18  0000000000009996799 999999999995999990
  20  00000000000099967999 99999999995999990
  21  010001010010999669999 99999999995999990
  22  0000000000009996799999 9999999995999990
  23  00000000000099966999999 999999995999990
  26  000000000000999679999999 99999995999990
  27  0000000000009996799999999 9999995999990
  28  000000010000999679999999999 999995999990
  29  0000000000009996799999999999 99995999990
  30  00000000000099967999999999999 9996999990
  31  000000000000999679999999999999 995999990
  32  0000000000009996799999999999999 96999990
  33  00000000000099967999999999999999 5999990
  34  444444444444555555555555556565 555550
  35  0000000000009996799999999999999995 99990
  36  00000000000099967999999999999999959 9990
  37  000000000000999679999999999999999599 990
  38  0000000000009996799999999999999995999 90
  39  00000000000099967999999999999999959999 0
  11  00433000202100000000000000000000000000

Average of pairwise = 0.4851 .
H_x = Average of 4*pairwise*(1-pairwise) = 0.2553 .
(Given the first average, H_x must be between 0 and 0.9991).
```

The clarity of the block structure is given by $H_x = 0.2553$. This always is a value between 0 and 1. Given that the average of the π_{ij} is 0.485, the value of 1.0 could not really be attained, but the maximum value for H_x – in case of a worthless block structure – is 0.9991. In this case the maximum is not appreciably less than 1.0.

```
Best separated vertices for identifying the 3 classes:
```

```
 39
 11  (errors  0.0000)
  2  (errors  0.0000).
```

If one would want to indicate the vertices that most clearly belong to three different classes (cf. Point 3 in Section 2.4), then the best choice would be 39, 11, and 2; the probability that they are not all in different classes is 0.0000, rounded to 4 decimal places.

By now we have an impression of the three classes (one of them a quite small one) that determine, to a great deal, the probabilities of the relations between the actors. But what are the corresponding probabilities? In Section 2.4 we already saw that these probabilities can only be expressed in a slightly indirect way.

The first method was treated in point 1 in Section 2.4 and the results are given in the next part of the output. This indicates, within the current 3-class model, the fitted probabilities of the relation types between the vertices. The first matrix gives (again, represented by the first decimal) the probabilities of the relation type with code 1, which corresponds to a null (0, 0) dyad.

```
@4
Probabilities of relation types between vertices
================================================

Probabilities of relation type 1 = (  0,  0) between vertices
           1111122    11112222222233333333331
      12357234694546890578012367890123456789 1
   1   766677676699988999999999999998999993
   2  7 666777676699988999999999999998999993
   3  66 556665656999789999999999999997999993
   5  665 566666669998899999999999999997999993
   7  6655 666666699988999999999999997999993
  12  77666 776766999889999999999999998999993
  13  776667 766669998899999999999999998999993
  14  7766677 676699988999999999999999998999993
  16  66566666 666999889999999999999998999993
  19  776667676 669998899999999999999998999993
  24  6656666666 699988999999999999999998999993
  25  66666666666 99988999999999999999998999993
   4  999999999999 999999999999999999999999996
   6  9999999999999 999999999999999999999999996
   8  99999999999999 999999999999999999999999996
   9  887888888888999 99999999999999999999995
  10  8888888888889999 9999999999999999999995
  15  99999999999999999 999999999999999999999996
  17  999999999999999999 999999999999999999999996
  18  999999999999999999 999999999999999999999996
  20  9999999999999999999 999999999999999999999996
  21  99999999999999999999 99999999999999999999996
  22  999999999999999999999 9999999999999999999996
  23  9999999999999999999999 999999999999999999999996
  26  99999999999999999999999 99999999999999999999996
  27  999999999999999999999999 9999999999999999999996
  28  9999999999999999999999999 999999999999999999999996
  29  99999999999999999999999999 99999999999999999999996
  30  999999999999999999999999999 9999999999999999999996
  31  9999999999999999999999999999 999999999999999999999996
  32  99999999999999999999999999999 99999999999999999999996
```

```
 33  99999999999999999999999999999 9999996
 34  88777888888899999999999999999 999995
 35  99999999999999999999999999999 99996
 36  99999999999999999999999999999 9996
 37  99999999999999999999999999999 996
 38  99999999999999999999999999999 96
 39  99999999999999999999999999999 6
 11  33333333333366655666666666666666566666
```

Within the first group 1-25 of vertices, the probabilities of a null dyad are between .5 and .8. For the second group 4-39 they are higher, mostly over .9, both within the group and with the first group. Actor 11 has probabilities about .3 of a null dyad with the first group, and about .6 of a null dyad with the second group.

The second matrix gives the probabilities of the relation type with code 2, corresponding to a mutual (1, 1) dyad.

```
Probabilities of relation type 2 = (  1,  1) between vertices
           1111122    11112222222233333333331
      12357234694546890578012367890123456789 1
   1   2332222222200011000000000000000001000005
   2  2 3322222222000110000000000000000001000005
   3  33 33333333300011000000001000001000006
   5  333 3333333300011000000000000000001000006
   7  2233 2223232000110000000000000000001000006
  12  22332 2222220001100000000000000001000005
  13  223322 2222200011000000000000000001000005
  14  2233222 222200011000000000000000001000005
  16  22333222 232000110000000000000000001000005
  19  223322222 220001100000000000000000001000005
  24  2233322232 200011000000000000000000001000006
  25  22332222222 000110000000000000000001000005
   4  000000000000 110011111111111111110111111
   6  0000000000001 100111111111111111110111111
   8  00000000000011 001111111111111111110111111
   9  111111111111000 100000000000000001000003
  10  1111111111110001 000000000000000001000002
  15  00000000000011100 111111111111110111111
  17  000000000000111001 11111111111110111111
  18  0000000000001110011 111111111110111111
  20  00000000000011100111 11111111110111111
  21  000000000000111001111 11111111110111111
  22  0000000000001110011111 1111111110111111
  23  00000000000011100111111 111111110111111
  26  000000000000111001111111 11111110111111
  27  0000000000001110011111111 1111110111111
  28  0010000000001110011111111 111110111111
  29  00000000000011100111111111 11110111111
  30  000000000000111001111111111 1110111111
  31  0000000000001110011111111111 110111111
  32  00000000000011100111111111111 10111111
  33  000000000000111001111111111111 0111111
  34  111111111111000110000000000000 000003
  35  0000000000001110011111111111110 11111
  36  00000000000011100111111111111101 1111
```

```
 37  00000000000011100111111111111111011 111
 38  00000000000011100111111111111110111 11
 39  00000000000011100111111111111101111 1
 11  55666555556511132111111111111311111
```

The matrix above shows that a mutual dyad (1,1) (code 2) is rare: the probabilities are between .2 and .4 within the first group 1-25 of vertices, less than .2 for all others except for actor 11 who has probabilities between .5 and .7 for mutual ties with actors in the first group.

The next two matrices in the output show similar results for the asymmetric relation types (0,1) and its counterpart (1,0). The first is reproduced below. It shows that the probability of an asymmetric dyad is between .1 and .2 within the first group (first digit 1); the probability of a (0,1) dyad where the 0 is from the second group to the first, and the 1 from the first to the second group, is between .1 and .2; an asymmetric dyad in the opposite direction between vertices in groups 1 and 2 has a probability less than .1; and within group 2 the probability also is less than .1. This contrasts with the probability of a mutual tie within group 1, which is between .1 and .2 (see the preceding matrix).

```
Probabilities of relation type 3 = (  0,  1) between vertices
           1111122    11112222222233333333331
      12357234694546890578012367890123456789 1
   1   1111111111000000000000000000000000001
   2  1 1111111111000000000000000000000000001
   3  11 1111111111000000000000000000000000001
   5  111 1111111111000000000000000000000000001
   7  1111 1111111111000000000000000000000000001
  12  11111 1111111111000000000000000000000000001
  13  111111 1111111111000000000000000000000000001
  14  1111111 1111111111000000000000000000000000001
  16  11111111 1111111111000000000000000000000000001
  19  111111111 1111111111000000000000000000000000001
  24  1111111111 1111111111000000000000000000000000001
  25  11111111111 0000000000000000000000000001
   4  111111111111 0000000000000000000000000003
   6  1111111111110 0000000000000000000000000003
   8  11111111111100 0000000000000000000000000003
   9  111111111111000 0000000000000000000000000002
  10  1111111111110000 0000000000000000000000002
  15  11111111111100000 0000000000000000000000003
  17  111111111111000000 0000000000000000000000003
  18  1111111111110000000 0000000000000000000000003
  20  11111111111100000000 0000000000000000000003
  21  111111111111000000000 0000000000000000000003
  22  1111111111110000000000 0000000000000000000003
  23  11111111111100000000000 0000000000000000000002
  26  111111111111000000000000 0000000000000000000003
  27  1111111111110000000000000 0000000000000000000003
  28  11111111111100000000000000 0000000000000000000003
  29  111111111111000000000000000 0000000000000000000003
  30  1111111111110000000000000000 0000000000000000000003
  31  11111111111100000000000000000 0000000000000000000003
  32  111111111111000000000000000000 0000000000000000000003
  33  1111111111110000000000000000000 0000000000000000000003
  34  11111111111100000000000000000000 0000000000000000000002
```

```
35  11111111111100000000000000000000 00003
36  11111111111100000000000000000000 0003
37  11111111111100000000000000000000 003
38  11111111111100000000000000000000 03
39  11111111111100000000000000000000 3
11  11111111111100000000000000000000
```

Finally, the output elaborates point 2 in Section 2.4. This means that the vertices with an ambiguous position are thrown out, the resulting clear class structure is presented (thus the latent classes are observed - or, rather, estimated), and the image matrix for this block structure is presented. Since the units are dyads rather than single relations, the image matrix really consists of several matrices.

```
@4
Finding strictly separated classes.
==================================
Vertices now can be thrown out because they are not compatible
with a partition of vertices into colors.
The following vertices are thrown out: 34 9 .
Overall maximum pairwise value for vertices in different classes is 0.416
Overall minimum pairwise value for vertices in the same class is    0.568
```

This analysis starts with the group structure discussed earlier that is apparent from the ordering and the matrix of pairwise color equality, but with the two worst fitting vertices disregarded, these being 9 and 34. The groups then are
group 1: 1-3, 5, 7, 12-14, 16, 19, 24, 25
group 2: 4, 6, 8, 10, 15, 17, 18, 20-23, 26-33, 35-39
group 3: the single vertex 11.

The maximum probability π_{ij} that two vertices in different groups yet have the same color (are in the same latent class) is 0.416, and the minimum probability that two vertices in the same group have the same color is 0.568. This shows that the groups are not very convincing estimates for the latent classes. The output immediately following elaborates the class structure implied by these 3 groups. We skip this here, because a similar but more interesting part of the output follows below.

Furhter ambiguous vertices now are thrown out. The result is as follows.

```
@5
Reduced vertex set
------------------
The following vertices were thrown out because of poor fit in the block structure:
9 10 11 34 .

Overall maximum pairwise value for vertices in different classes is 0.065
Overall minimum pairwise value for vertices in the same class is    0.683
Classes now are labeled 1 to 2.
```

At this point, the program finds that the remaining groups are well enough separated, because the minimum probability π_{ij} that two vertices in the same group have the same color exceeds by more than 0.6 the maximum probability that two vertices in different groups yet have the same color. (The difference here is $0.683 - 0.065 = 0.618$.) To some extent, the baby has been thrown out with

the bathwater, because vertex 11 with its special role was thrown out. This reflects that, although in some sense this vertex defines a class on its own, it also had a non-negligible probability of belonging to group 1.

There follows a table indicating how well the vertices belong in the current group structure and how much this would improve if any given vertex were thrown out. We present only the table in the new order of the vertices.

```
Vertices, as far as not thrown out, with their class number,
their maximum pairwise value for vertices in a different class,
and their minimum pairwise value for vertices in the same class;
and, if this vertex were to be deleted:
the overall maximum pairwise value for vertices in different classes,
and the overall minimum pairwise value for vertices in the same class:

.....................................
In new ranking:
   1  1  0.056  0.686  0.065  0.683
   2  1  0.056  0.683  0.065  0.685
   3  1  0.051  0.683  0.065  0.729
   5  1  0.046  0.729  0.065  0.683
   7  1  0.058  0.768  0.065  0.683
  12  1  0.056  0.685  0.065  0.683
  13  1  0.054  0.691  0.065  0.683
  14  1  0.054  0.693  0.065  0.683
  16  1  0.039  0.711  0.065  0.683
  19  1  0.051  0.692  0.065  0.683
  24  1  0.065  0.763  0.058  0.683
  25  1  0.044  0.692  0.065  0.683
   4  2  0.018  0.945  0.065  0.683
   6  2  0.017  0.946  0.065  0.683
   8  2  0.028  0.933  0.065  0.683
   9 out
  10 out
  15  2  0.019  0.945  0.065  0.683
  17  2  0.024  0.940  0.065  0.683
  18  2  0.018  0.944  0.065  0.683
  20  2  0.017  0.947  0.065  0.683
  21  2  0.065  0.909  0.049  0.683
  22  2  0.021  0.939  0.065  0.683
  23  2  0.049  0.909  0.065  0.683
  26  2  0.018  0.942  0.065  0.683
  27  2  0.017  0.942  0.065  0.683
  28  2  0.043  0.922  0.065  0.683
  29  2  0.027  0.949  0.065  0.683
  30  2  0.034  0.934  0.065  0.683
  31  2  0.018  0.946  0.065  0.683
  32  2  0.034  0.927  0.065  0.683
  33  2  0.016  0.946  0.065  0.683
  34 out
  35  2  0.034  0.928  0.065  0.683
  36  2  0.017  0.946  0.065  0.683
  37  2  0.021  0.943  0.065  0.683
  38  2  0.016  0.946  0.065  0.683
  39  2  0.017  0.946  0.065  0.683
  11 out
```

The low values in the column "maximum pairwise in different classes" show that in this class structure, the groups 1 and 2 really are different. The column "minimum pairwise in the same class" shows that group 2 is very homogenous (all values larger than 0.9) but group 1 is less so (values between .68 and .77).

Next the adjacency matrix in recoded and reordered form is shown again, now with blanks for the deleted vertices and with rows of dots for the separation between the groups.

```
Recoded adjacency matrix with block structure :
           1111122.    11112222222233333333331
      123572346945.468905780123678901234567891
   1   42411111111.111  111141111111111 11111
   2  3 1213113114.111  111111111411111 11111
   3  21 221222311.111  111141111211111 11111
   5  322 22112111.111  121411111141111 11111
   7  1122 2121311.111  111111111111111 11111
  12  14122 111141.111  111111111111111 11111
  13  112111 41211.111  111111111111111 11411
  14  1121213 1124.111  111111111111111 11111
  16  14221111 121.111  141121111144111 11411
  19  114141211 43.111  111111111441111 11111
  24  1111131223 1.113  111111111114121 11111
  25  13111113141 .111  111111111141411 44411
  ........................................
   4  111111111111. 12  111111111211111 11111
   6  111111111111.1 1  111111111111112 11111
   8  111111111141.21   122111111111111 11111
   9
  10
  15  111111111111.111    11111211121111 11111
  17  111211113111.112  1 2111111111111 11111
  18  111111111111.112  12 111111111111 11111
  20  111311111111.111  111 11111111111 11211
  21  313111112111.111  1111 1111121111 11111
  22  111111111111.111  11111 311111111 11111
  23  111111111111.111  211114 11111111 11111
  26  111111111111.111  1111111 1111111 11111
  27  111111111111.111  11111111 111111 11111
  28  132111111311.211  111111111 11111 11111
  29  111311113313.111  2111211111 1111 11111
  30  111111113131.111  11111111111 121 11111
  31  111111111113.111  111111111111 11 12111
  32  111111111121.111  1111111111121 1 11211
  33  111111111111.121  11111111111111  11112
  34
  35  111111111113.111  111111111111111  1111
  36  111111111113.111  111111111111211 1 111
  37  111111313113.111  111211111111121 11 11
  38  111111111111.111  111111111111111 111 1
  39  111111111111.111  111111111111112 1111
  11
```

This adjacency matrix shows visually quite a convincing block structure.

Now follow, for the relations in this reduced vertex set, the fitted probabilities and the observed frequencies of the relations. First the fitted probabilities:

```
@6
Posterior probabilities and frequencies for post-hoc coloring
.........................................................

For these classes, two threeway tables :

Posterior probabilities
-----------------------
Average posterior probabilities of relation 1 = (  0,  0) in these blocks.
       1      2
  1   0.61   0.90
  2   0.90   0.94

Average posterior probabilities of relation 2 = (  1,  1) in these blocks.
       1      2
  1   0.23   0.03
  2   0.03   0.06

Average posterior probabilities of relation 3 = (  0,  1) in these blocks.
       1      2
  1   0.08   0.01
  2   0.07   0.00

Average posterior probabilities of relation 4 = (  1,  0) in these blocks.
       1      2
  1   0.08   0.07
  2   0.01   0.00
```

And next the observed frequencies:

```
Observed frequencies
--------------------
Observed relative frequencies of relation 1 = (  0,  0) in these blocks.
       1      2
  1   0.59   0.92
  2   0.92   0.94

Observed relative frequencies of relation 2 = (  1,  1) in these blocks.
       1      2
  1   0.23   0.01
  2   0.01   0.06

Observed relative frequencies of relation 3 = (  0,  1) in these blocks.
       1      2
  1   0.09   0.00
  2   0.07   0.00

Observed relative frequencies of relation 4 = (  1,  0) in these blocks.
       1      2
  1   0.09   0.07
  2   0.00   0.00
```

The fitted probabilities and the observed frequencies are almost the same, but not quite, due to the Bayesian estimation method. The differences between them are not important. These matrices

provide, in more compact form, similar information as the matrices of probabilities of relations between the vertices.

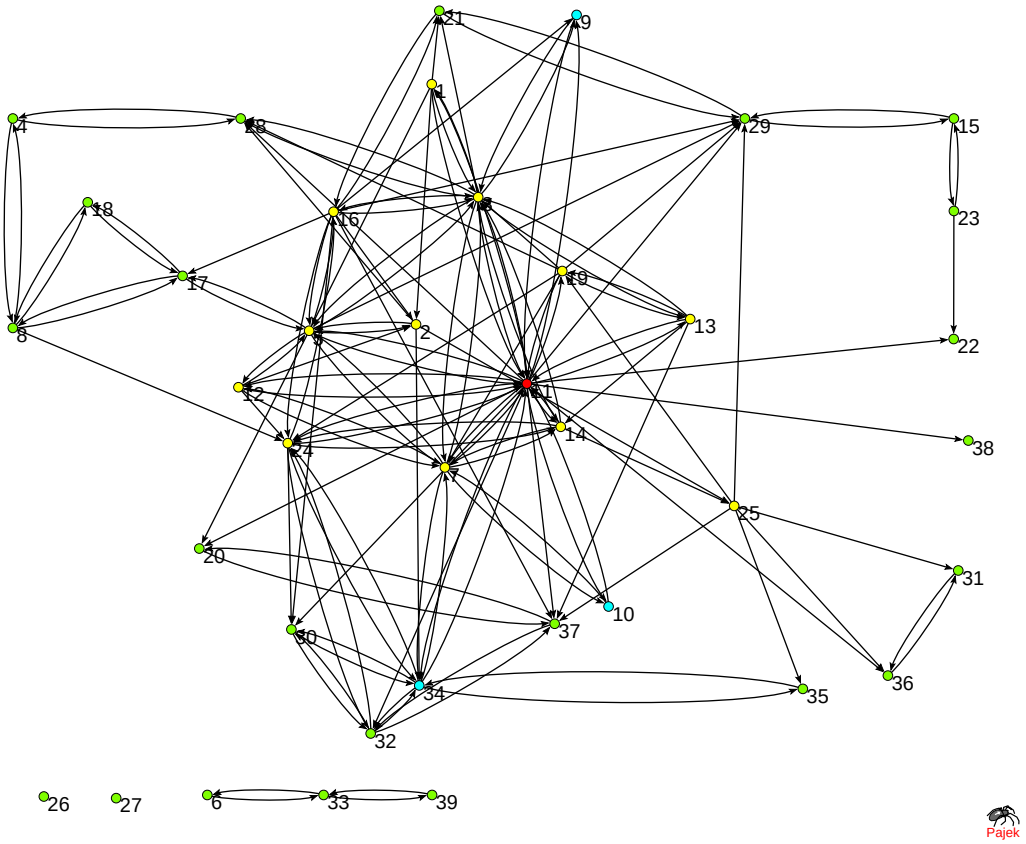
The very end provides a possibility for diagnosing the position of the vertices which were thrown out because they did not conform well to the block structure. In this case, vertex 11 is the most interesting.

```
Relations of poorly fitting vertices with these classes :
.................................................
Observed relative frequencies of relations between vertex 11 and these classes.

               1      2
(  0,  0)    0.25   0.61
(  1,  1)    0.75   0.00
(  0,  1)    0.00   0.00
(  1,  0)    0.00   0.39
```

This shows again, now in compact form, the propensity of actor 11 to have mutual relations with group 1 and to send non-reciprocated friendship relations to group 2.

The graph with vertices colored according to their class (1 = yellow, 2 = green, 3 = red; excluded = light blue) represents a kind of center-periphery structure, in which actor 11 (red) is very central, the yellow class further is the center and the green class is the periphery. The blue vertices are atypical in this classification. The isolated vertices are classified as green; this indicates that the probabilities for a friendship relation for green vertices are so low that it is not too improbable that a green vertex are isolated.



At this moment we have come to the end of the output for a single Gibbs sequence for 3 colors. A next possibility is to run the program for more or less colors. Running it for 2-4 colors gives the following results for the information I_y and the clarity of the block structure H_x . The figures given in Table 1 show that the clearest class structure, as expressed by H_x , is obtained for $c = 2$. This table shows a slight decrease in the information I_y when c goes from 2 to 4.

Table 1: Parameters for the class structure for Kapferer's data set

c	I_y	H_x
2	0.43	0.12
3	0.41	0.25
4	0.40	0.41

This table leads to a preference for 2 rather than 3 classes. In Nowicki and Snijders (2001) we analyzed the friendship and assistance relation in this network simultaneously. For this multiplex or multivariate relation the method found 3 classes, with again actor 11, a line 1 tailor named Lyashi, as a class on his own.

Part II
Programmer's manual

The programmer's manual will not be important for most users. It is intended for those who wish to run BLOCKS outside of the StOCNET environment and to those who want to have a look inside the source code.

The program consists of a basic computation part programmed by the author in Turbo Pascal and Delphi; and the StOCNET windows shell, programmed by Peter Boer in Delphi, with Mark Huisman (earlier Evelien Zeggelink) as the project leader (see Boer, Huisman, Snijders, and Zeggelink, 2003). The computational part can be used both directly and from the windows shell. The StOCNET windows shell is much easier for data specification and model definition.

10 Parts and units

The calculations of the program are carried out by the executable file `BLOCKS.EXE` which reads the basic information file and executes the statistical procedures accordingly. No user interaction is required, although there are possibilities of early termination.

If you wish to run `BLOCKS` outside of StOCNET, the project name *must* be given in the command line, e.g.

```
BLOCKS kapfer
```

if `kapfer` is the name of the project, and there exists a `kapfer.in` file. This `kapfer` is called a *command line parameter*. There are the following three ways to specify a command with a command line parameter in Windows. The command line can be given at the DOS prompt (in a Windows environment); it can be given in the Windows "Run" command (for Windows 98 and higher); and it can be indicated in the "target" in the "properties" of a shortcut.

The source code is divided into the following units.

1. `BLOCKMO`, which contains some basic variables, a procedure for reading the basic information file, and some initialisation routines.
2. `MULBLO`, which contains the computational routines.
3. `BLOCKI`, which contains routines for user interaction (and can be adapted to the desired operating system, e.g., DOS or Windows).
4. `B_CONS`, which is a short unit containing constants which can be chosen to compile the program in the desired size.
5. `RANGEN`, which contains procedures for random number generation.
6. `WBLIB`, which contains some utilities.

11 Basic information file

The basic information file (named *pname*`.in`, where *pname* is the project name) is an ASCII file which contains the commands for the computation part. It consists of the following lines. If a line contains more than one number, these numbers must be separated by one or more blanks.

After the required information, the lines are allowed to contain at least one blank and after that more text, which can be used for helpful information. However, the line with identifying vertices or the name of the file containing identifying probabilities, if it is present (in which case it is line 13), should not contain additional text.

Items 9–12 can be considered advanced options.

1. A title line which is reproduced in the output but has not other effects.
2. The data file name.
   (Must be composed of characters with ASCII numbers ranging from 43 to 126.)
3. The minimum and maximum number of colors, separated by a blank.
   (Denote these number by *cmin* and *cmax*.)
   Suggested defaults: 2 and 4.
4. The number of runs before convergence.
   Suggested default: 10,000.
5. The number of runs after convergence.
   Suggested default: 10,000.
6. The number of Gibbs sequences per estimated model.
   Suggested default: 1 for data exploration, 3 for serious analysis.
   The number 0 is allowed, and leads to an input check and a description of the alphabet, without any Gibbs sequences.
7. The number 0 for a non-identified model, 1 for an identified model with a set of identifying vertices, 2 for an identified model with an input matrix of prior probabilities. The value 2 should only be used if *cmin* = *cmax*.
   (Denote this number by *ident*.)
   Suggested default: 0.
8. The missing number indicator (one integer number).
   Suggested default: 9.
9. The concentration parameter.
   Suggested default: 100.
10. 1 if a search for good initial colorings is to be performed, 0 if the initial colorings are to be determined randomly.
    Suggested default: 1.
11. 1 if the pre-convergence iterations should use overdispersed colorings, 0 otherwise.
    Suggested default: 1.
12. 1 if the pre-convergence iterations should use overdispersed probabilities, 0 otherwise.
    Suggested default: 1.
13. If *ident* = 0 (see item 7 above), this line is absent.
    If *ident* = 1, the numbers of the identifying vertices, followed by the number 0.
    (Normally this is at least *cmin* - 1 numbers and at most *cmax*, before the terminating 0.)
    If *ident* = 2, the name of the file with the prior probabilities.
    (Requirements for this file are given above in section 7.2).

14. 0 if new output is to overwrite an output file of the same name (if this exists),
    1 if new output is to be appended to an output file of the same name (if this exists).
15. The seed for the random number generator.
    This determines the result, since the random number generator is only pseudo-random, which means that it deterministic but functions just like a real random number generator. The value 0 implies that the program will determine a seed based on the clock time. Using the same random number seed implies that exactly the same results will be obtained.

An example of a basic information file, as used for the analysis of Kapferer's data set presented above, is as follows.

```
Kapferer Taylor Shop time 2,  friendship
Kapff2.dat     (data file)
 2 4       (minimum and maximum number of colors)
 20000     (number of runs before convergence is assumed)
 20000     (number of runs after convergence is assumed)
 3         (number of Gibbs sequences per color)
 0         (identification mode)
 9         (missing number indicator)
  100.00   (concentration parameter)
  123      (random number seed)
 1         (yes/no search for good starting colorings)
 1         (yes/no overdispersed colorings in pre-convergence iterations)
 1         (yes/no overdispersed probabilities in pre-convergence iterations)
 0         (0 = rewrite, 1 = append output file )
```

The explanations between parentheses are not read by BLOCKS, and can be included or omitted as desired.

12 Constants

The constants in unit B_CONS are the following. They can be adapted to the possibilities of the computer configuration and the desires of the user.

1. *cmax*, the maximum number of colors; a reasonable range is 5 to 10.
2. *cc*, the maximum absolute value of permitted data values; suggested $cc = 9$.
3. *rmax*, the maximum alphabet size; a reasonable range is 6 to 12.
4. *r01max*, the maximum of r_{01} (see the paper); a value higher than *rmax* is meaningless; a reasonable value is *rmax* or slightly less.
5. *mulmax*, the maximum number of Gibbs sequences per model; this is only important for the start of the program, where a search is done for optimal starting colorings; a reasonable value is 5.

13 References

Batagelj, V., and Mrvar, A. 1997–2002. *Pajek. Program for Large Network Analysis.* Ljubljana: University of Ljubljana.

Boer, P., Huisman, M., Snijders, T.A.B., and E.P.H. Zeggelink. 2003. *StOCNET: An open software system for the advanced statistical analysis of social networks.* Version 1.4. Groningen: Pro*GAMMA*/ICS. Available from http://stat.gamma.rug.nl/stocnet/.

Borgatti, S., Everett, M.G., and Freeman, L.C. (1998), *UCINET V, Reference Manual.* Columbia, SC: Analytic Technologies.

Holland, P., Laskey, K.B., and Leinhardt, S. (1983). "Stochastic blockmodels: Some first steps". *Social Networks, 5*, 109 – 137.

Kapferer, B. (1972). *Strategy and transaction in an African factory.* Manchester: Manchester University Press.

Lorrain, F., and White, H.C. (1971). "Structural equivalence of individuals in social networks". *Journal of Mathematical Sociology, 1*, 49 – 80.

Nowicki, K., and Snijders, T.A.B. (2001). "Estimation and prediction for stochastic blockstructures". *Journal of the American Statistical Association, 96*, 1077-1087.

Snijders, T., and Nowicki, K. (1997). "Estimation and prediction for stochastic blockmodels for graphs with latent block structure". *Journal of Classification, 14*, 75 – 100.

Wasserman, S., and Faust, K. (1994). *Social Network Analysis: Methods and Applications.* New York and Cambridge: Cambridge University Press.