Help File for the Maoz Network Program[1]

The Maoz Network Program (MaozNet) is a Social Networks Analysis (SNA) package that implements many existing SNA programs, as well as new applications developed by Zeev Maoz. The programming was done with the help of Andrey Goder, Nicky Chan, and Peter Marteen. This is a beta version that may well have numerous bugs. Users are encouraged to send questions, comments, and suggestions to zmaoz@ucdavis.edu.

In order to run the Network Program you will need the .NET 2005 compiler from Microsoft. You can download it from http://www.microsoft.com/downloads/details.aspx?FamilyID=9655156b-356b-4a2c-857c-e62f50ae9a55&DisplayLang=en.

This help file is divided into modules. Each module should be inserted as a separate help feature in the relevant menu items in the Network Program. The modules are organized by menu item.

FILE MENU

1. LOAD MENU

This program enables the user to load files in various formats. All files must be comma-delimited (.csv) files, however. The following formats can be loaded.

a. **Dyadic File format**: dyadic files consist of four columns as follows:

- Network index: four-digit integer index of the network. Each network has a different index.
- Row index: this index can be a numeric (integer) index, or an alphanumeric (string) index.
- Column index: same as row index. Column and row indexes must match exactly. **Note: Entries for diagonals must be present, whether the main diagonal is zero throughout or whether self-ties are allowed. Otherwise the program will crash.**
- Cell entry. Cell entry can be any real, integer, or binary number.
- **The user can specify if the first row is a labels row**

Example of a Dyadic File Format

Network Index	Row Index	Column Index	Cell Entry
2220	A	A	1
2220	A	B	1
2220	A	C	0
2220	A	D	1
2220	B	A	1

[1] I would like to thank my research assistant Andrey Goder for helping programs the software package.

2220	B	B	1
2220	B	C	0
2220	B	D	1
2220	C	A	0
2220	C	B	0
2220	C	C	1
2220	C	D	0
2220	D	A	1
2220	D	B	1
2220	D	C	0
2220	D	D	0
2221	A	A	1
2221	A	B	1
2221	A	C	0
2221	A	D	1
2221	B	A	1
2221	B	B	1
2221	B	C	0
2221	B	D	1
2221	C	A	0
2221	C	B	0
2221	C	C	1
2221	C	D	0
2221	D	A	1
2220	D	B	1
2220	D	C	0
2220	D	D	0

b. **Matrix format**

A matrix file is a .csv file with the following structure.

- Top left cell, first matrix index
- Following matrix index, a $(n+1) \times (n+1)$ matrix. First row in the matrix provides column labels; first column of each matrix provides row labels. Entries in each matrix can be binary, integers, or real numbers denoting relations between row *i* and column *j*.
- Next matrix follows the same format starting with a network index on the top leftmost cell.

Note that the size of the matrix can change from one matrix to another.

Example of a matrix format:

1	Column Index (not shown)					
Row Index	**1**	**2**	**3**	**4**	**5**	
1	0	0	1	1	1	
2	1	0	0	0	1	
3	0	1	0	0	0	
4	0	0	0	1	0	
5	1	0	1	1	1	
2	Column Index (not shown)					
	1	**2**	**3**	**4**	**5**	**6**
1	0	1	0	0	1	0
2	1	0	1	1	0	1
3	1	1	1	1	1	0
4	1	1	1	1	1	0
5	0	1	0	0	0	0
6	0	0	0	1	1	1
3	Column Index					
	1	**2**	**3**	**4**	**5**	
1	0	0	0	1	0	
2	0	0	1	0	0	
3	0	0	0	0	0	
4	1	1	1	0	1	
5	1	1	1	1	0	
4	Column Index					
	1	**2**	**3**	**4**	**5**	
1	0	1	1	0	1	
2	1	0	1	1	1	
3	0	0	0	1	1	
4	0	0	0	0	0	
5	0	0	1	0	0	

Note: Example colors special cells of the matrices. In practice, cells are not colored or boldfaced. In this example green cells are network index; blue cells are row/column labels.

i. **Multi Variable Dyadic File**
Same format as single variable dyadic file except that it contains multiple variables in the columns following row and column labels.

Example

Network Index	Row Index	Column Index	Var1	Var2	Var3
2220	A	A	1	0.1	-3
2220	A	B	1	0.2	1
2220	A	C	0	0.1	-1

2220	A	D	1	0.3	0
2220	B	A	1	0.2	1
2220	B	B	1	0.1	-2
2220	B	C	0	0.3	-1
2220	B	D	1	0.4	2
2220	C	A	0	0.01	-3
2220	C	B	0	0.25	2
2220	C	C	1	0.4	-3
2220	C	D	0	0	-1
2220	D	A	1	0.1	0
2220	D	B	1	0.4	0
2220	D	C	0	0.7	1
2220	D	D	1	0.3	0
2221	A	A	1	0.5	-2
2221	A	B	0	0.2	1
2221	A	C	4	0.5	4
2221	A	D	1	0.3	0
2221	B	A	0	0.2	-1
2221	B	B	1	0.1	-2
2221	B	C	0	0.3	-1
2221	B	D	1	0.4	2
2221	C	A	0	0.01	-3
2221	C	B	0	0.25	2
2221	C	C	1	0.4	-3
2221	C	D	0	0	-1
2221	D	A	0	0.5	2
2221	D	B	1	0.4	0
2221	D	C	0	0.7	1
2221	D	D	1	0.3	0

ii. **Multiple Matrix Format**

Same as single matrix format, with multiple matrices. This option requires a set of files each containing a group of sociomatrices with network identifiers (ni=1,…,k). The program requires the user to specify individual files where each file must have an identical range of matrix indexes, and, for each matrix index, all matrices must be of the same dimension. For example, consider file *A* with matrices identified by 1, 2, 3, 5, 10, and with M_1 of size 3 × 3, M_{A2} of size 5 × 5, M_{A3} of size 7 × 7, M_{A5} of size 4 × 4, M_{A5} of size 6 × 6, and M_{A10} of size 3 × 3. The multiple matrix load program requires that files B, C, D…, have also the same matrix indexes (1, 2, 3, 5, 10) and that each File M_{Bi}, M_{Ci}, M_{Di} have the same size as M_{Ai} (where i is the corresponding matrix index).

iii. **Affiliation Matrix format**

An affiliation matrix is an $n \times k$ matrix that connects nodes with events. Examples include lists of department members and their affiliation in professional societies, state membership in international organizations, religious or linguistic breakdowns of a state's population, and so forth. The format of an affiliation matrix is:

- Network index
- Row index
- Event1, event2, …, event k

An example of an affiliation network format is given by:

Network Index	Row Index										
		R1	R2	R3	R4	R5	R6	R7	R8	R9	
1	1	0	0	0	0	1	0	0	1	1	
1	2	1	1	0	0	1	0	0	0	0	
1	3	1	0	0	0	0	0	1	0	0	
1	4	0	0	1	1	0	1	0	1	1	
1	5	1	1	0	0	1	0	0	0	0	
1	6	1	1	1	0	1	1	0	0	1	
1	7	1	1	1	1	0	1	1	0	0	
1	8	0	1	0	1	0	1	0	0	0	
1	9	1	0	1	0	0	0	0	0	0	
1	10	0	0	0	1	0	0	1	1	0	
2	1	0	0	0	0	1	1	0	1		
2	2	1	1	1	1	1	1	1	1		
2	3	1	0	0	1	1	1	1	0		
2	4	1	1	0	0	1	1	0	1		
2	5	1	1	1	1	0	0	0	0		
2	6	0	0	1	1	0	0	1	0		
2	7	1	1	1	1	0	0	0	0		
2	8	1	0	0	0	1	0	0	0		
2	9	0	1	1	0	0	0	0	1		
2	10	0	0	0	0	0	1	1	0		
3	1	1	1	0	0	0	1	0	0	1	1
3	2	0	0	0	0	1	1	1	0	0	0
3	3	1	0	0	1	1	1	1	1	0	1
3	4	0	0	0	0	0	0	1	1	0	0
3	5	1	1	1	1	1	1	0	0	0	0
3	6	0	1	1	0	0	0	0	0	1	0
3	7	1	1	1	1	0	0	0	1	0	0
3	8	0	0	1	1	1	0	1	1	0	0
3	9	1	1	1	1	1	1	0	0	0	0
3	10	0	0	0	0	0	0	0	0	1	1

Note: Each network may have a different number of events. The first row in the dataset contains labels only for the first network. The network programs will assign labels to all networks regardless of the number of events.

iv. Monadic Diagonal Files

These are what SNA calls attribute files. The file forms an $n \times 1$ vector with the following format:

- Network index
- Row label
- Variable

An example is given by

Network Index	Row Index	Var1
2220	A	1
2220	B	1
2220	C	0
2220	D	1
2221	A	1
2221	B	1
2221	C	0
2221	D	1

The program converts the file into a diagonal sociomatrix with $s_{ii}=r_i$ and $s_{ij}=0 \ \forall j \neq i$.

v. Random Matrices

Random matrices are matrices that the program produces randomly. A number of options exist for random matrix generation:

1. **Binary Matrices.** This option produces either symmetric (non-directional) or asymmetric (directional) matrix from a range of $s_{ij}=0 \ \forall i,j \in N$ to $s_{ij}=1 \ \forall i,j \in N$ (where N is the size of the matrix's dimension). The user can specify a fixed N or a variable N within a given range, and the beginning network index (defaults are $N=3$ and network index=1820). The program also allows the user to specify a fixed probability of a nonzero link for all random networks, or a variable probability range, so that the probability of a link within one network would differ from the probability of a link in another network.
2. **Valued Matrices.** This produces random valued networks. Here there are several options. First, as in the binary case, the program allows for symmetric or asymmetric matrices. Second, ranges of N's can be specified for different matrices. Third, the probabilities of nonzero links can be fixed or variable—within a range specified by the user. However, for valued networks there are two more options. First, the user can specify if the link values are integers or real numbers. Second, the range of values

can be specified. Finally, the user can decide whether diagonals are zero or valued entries.

Random matrix modes are suitable for simulations.

2. **SAVE AS MENU**

a. **Dyadic File.** File is saved as a dyadic file. The user is asked to specify the start and end values of the network index. The output is a .csv file with the following format:

- Network index
- Row label
- Column Label
- Value

An example output is:

Network Index	Row Index	Column Index	Cell Entry
2220	A	A	1
2220	A	B	1
2220	A	C	0
2220	A	D	1
2220	B	A	1
2220	B	B	1
2220	B	C	0
2220	B	D	1
2220	C	A	0
2220	C	B	0
2220	C	C	1
2220	C	D	0
2220	D	A	1
2220	D	B	1
2220	D	C	0
2220	D	D	1

b. **Matrix File:**

The program asks the user to specify the range of matrix indexes to be written. The output file is a .csv file with the following format

- Matrix index
- Top row is column labels
- Rightmost column is row labels
- All other 2…n, rows and 2…n columns are matrix entries.

An example is:

1111			Column Index			
Row Index	**1**	**2**	**3**	**4**	**5**	
1	0	0	1	1	1	
2	1	0	0	0	1	
3	0	1	0	0	0	
4	0	0	0	1	0	
5	1	0	1	1	1	
1112			Column Index			
	1	**2**	**3**	**4**	**5**	**6**
1	0	1	0	0	1	0
2	1	0	1	1	0	1
3	1	1	1	1	1	0
4	1	1	1	1	1	0
5	0	1	0	0	0	0
6	0	0	0	1	1	1
1113			Column Index			
	1	**2**	**3**	**4**	**5**	
1	0	0	0	1	0	
2	0	0	1	0	0	
3	0	0	0	0	0	
4	1	1	1	0	1	
5	1	1	1	1	0	
1114						
	1	**2**	**3**	**4**	**5**	
1	0	1	1	0	1	
2	1	0	1	1	1	
3	0	0	0	1	1	
4	0	0	0	0	0	
5	0	0	1	0	0	

Note that the size of the matrix can change from one matrix to another.

c. **Network Characteristics Output File**. This is an output file of the counter program that provides general network attributes for each network. The format of the .csv output file is:

- **Network index**
- **N:** Network size
- **No. Clqs:** Number of cliques
- **Clq. Size:** Average Clique Size—Average number of members in cliques
- **Clq. Mem:** Average number of clique memberships per unit in networks
- **No. Components.** Number of components (clusters of reachable nodes)
- **G/N.** The proportion of the number of nodes in largest component to the network size.
- **Simple NPOL**: Simple Node POLarization index (defined in the matrix program)

- **CMOI**: Clique membership overlap index (defined in the matrix program)
- **Simple NPI**: Simple Network Polarization Index (defined in the matrix program)
- **Simple COC**: Simple Clique overlap Index (defined in the matrix program)
- **Complex COC:** Modified clique overlap index (defined in the matrix program).
- **NPOL Cohesion: NPOL** modified by average clique cohesion (defined in the matrix program)
- **NPI Cohesion CMOI: NPI** based on NPOL Cohesion and CMOI
- **NPI Cohesion COC: NPI** based on NPOL Cohesion and COI
- **NPOL Size: NPOL** based on clique size (specify external file in options menu).
- **NPOL Coh. Size: NPOL** based on both cohesion and clique size (external file)
- **NPICoh. Size CMO: NPI** based on NPOL Coh. Size and CMOI
- **NPI Coh. Size COC: NPI** based on NPOL Coh. Size and COI
- **Density**: Network density (defined in the matrix program)
- **Transitivity**: Transitivity index (defined in the matrix program)
- **Interdependence**: Systemic interdependence measure (defined in the matrix program)
- **ERPOL**. Esteban and Ray Polarization index (defined in matrix program).
- **Clusterability Coefficient**. (defined in matrix program).
- **Average DI**. Average no. of incoming (column) ties.
- **Average DO.** Average no. of outgoing (row) ties.
- **DIG.** Incoming Group Degree centralization (defined in matrix program).
- **DOG.** Outgoing Group Degree centralization (defined in matrix program).
- **CIG.** Incoming Group Closeness centralization (defined in matrix program).
- **COG.** Outgoing Group Closeness centralization (defined in matrix program).
- **BIG.** Incoming Group Betweenness centralization (defined in matrix program).
- **BOG.** Outgoing Group Betweenness centralization (defined in matrix program).
- **EIG.** Incoming Group Eigenvector centralization (defined in matrix program).
- **EOG.** Outgoing Group Eigenvector centralization (defined in matrix program).

General Note: This program will be expanded shortly to include a set of generalized NPI-related statistics

d. **Multiple Matrix Files**. When a network procedure produces multiple matrix outputs, you can save it in multiple matrix files. The program tells you how many matrix files need to be written and prompts you to write the names of the files (as .csv files) successively. Once you finished inserting the names of the output files, it asks you to specify the network index range for the files.

e. **Multi Variable File**. When a network procedure produces multiple matrix outputs, you can save it in multiple variable dyadic file. The file has the same structure as the output single-variable file, except that the table has multiple variables. Example of a multivariable file is given below.

Network Index	Row	Column	Variable	Variable

	Index	Index	1	2
2220	A	A	.3	1
2220	A	B	.5	1
2220	A	C	.7	0
2220	A	D	.1	1
2220	B	A	0	1
2220	B	B	0	1
2220	B	C	0	0
2220	B	D	.5	1
2220	C	A	.4	0
2220	C	B	0	0
2220	C	C	.5	1
2220	C	D	.4	0
2220	D	A	0	1
2220	D	B	0	1
2220	D	C	0	0
2220	D	D	0	1

3. **Close File.** This closes the file in the memory but does not eliminate other aspects of the program's data in memory.

4. **RESET Command**. This command empties the memory of the program, and requires it to reload input files. This command is useful after a program crash due to an error. Use it with caution.

5. **DATA MANAGEMENT Menu**

This menu enables several operations on the matrix/ces in memory.

a. **Dichotomization**. For a valued network, some operations require dichotomization. If you wish to dichotomize the network, click this menu, and then select the cutoff value for dichotomization in the dialog box. The program also allows you to load an attribute file (see above) with cutoff values for each of the networks in the input file.
b. **Recode**. This operation allows you to recode the values of the input networks. The dialog box requires the user to specify the minimum and maximum value of each category and the value assigned to it.
c. **Affiliation to Sociomatrix Conversion**. If the input file is an affiliation network (2-mode network) file, this allows you to convert it into a Sociomatrix (1-mode network). Given an affiliation matrix **A** of order *n (nodes)* × *k (groups),* the program converts it into a relational matrix. If the user chooses this operation, the dialog box that opens offers three conversion options. For each option, the user must choose the resulting unit of analysis. There are two possibilities here: (1) unit-based conversion, this is the nodal level. The conversion operations result in $n \times n$ sociomatrices with nodes as row and column labels. (2) an event-based conversion, this is the event (group) level. The conversion operations result in $k \times k$ matrices with the with groups as row and column labels.
    (i) **Sociomatrix.** This is the common SNA method. Given an affiliation matrix **A** of order *n (nodes)* × *k (groups),* if the user chooses the unit-based

conversion, this matrix is converted into a sociomatrix **S,** such that **S = A × A'.** If the user chooses the event-based conversion, this matrix is converted into a sociomatrix such that **S = A' × A.**

(ii) **Correlation.** If the user chooses a unit-based conversion, the resulting matrix is a set of Pearson product-moment correlations between sets of rows of the **A** matrix. Thus the entries of **S**, $s_{ij} = r(a_i,a_j)$ where *i* and *j* index two rows of the matrix **A.** If the user chooses event-based conversion then $s_{ij} = r(a_q,a_r)$ where *q* and *r* are two columns of the matrix **A.**

(iii) **Euclidean**. If the user chooses a unit-based conversion, the resulting matrix is a set of standardized Euclidean distance proximity scores between the respective rows of the affiliation matrix **A**. Thus, the entry s_{ij} in the resulting matrix **S** is defined as $s_{ij} = 1 - \frac{\sum_k (a_i - a_j)^2}{\mathrm{k[max}\,(A)^2]}$ (where max(**A)** is the maximum value in matrix **A.** If the user chooses the event-based conversion, then the entry s_{ij} is defined as $s_{qr} = 1 - \frac{\sum_k (a_q - a_r)^2}{\mathrm{n[max}\,(A)^2]}$, where *q* and *r* are two columns in **A** and *n* is the number of nodes in the network.

d. **Matrix Multiplication**. This is a utility program allowing simple matrix multiplication of a series of matrices. It uses the current dataset that is composed of one or more matrices (of a general $m \times n$ order, with *m* and *n* varying from one input matrix to another), and asks the user to specify another matrix (that can be inputted either as a matrices dataset or as a dyadic dataset). It then performs the multiplication operation. The user can choose a file that is either in matrix form or dyadic form for the multiplications.

e. **Elementwise Multiplication**. This procedure performs elementwise multiplication of the current matrix *S* by a matrix *M* or by a vector *V* that are specified by the user. The result is an elementwise multiplication of s_{ij} by m_{ij} or by v_i. For the procedure to work, the matrix *M* must be of the same dimensions as matrix *S*, or the vector must have the same number of rows as does *S*. The user can specify three forms of input for the multiplied datasets: (a) matrix file, (b) dyadic file, or (c) monadic file (vector). **Example**. Given a dataset consisting of the following matrices $S_{1(3\times2)}$, $S_{2(4\times5)}$, $S_{3(7\times3)}$ (expressions in the subscripted parentheses denote the dimensions of each matrix), the dataset used for elementwise matrix multiplication must be $O_{1(3\times2)}$, $O_{2(4\times5)}$, $O_{3(7\times3)}$. If the user chooses a vector option, then the vectors must be $V_{1(3\times1)}$, $V_{2(4\times1)}$, $V_{2(7\times1)}$. The output screen presents the first product matrix $T_{1(3\times a)} = S_1 \times O_1$. Other matrices in the resulting dataset can be displayed via the **Scroll Menu.**

SCROLL MENU

The program allows you to see the data in matrix/spreadsheet format. It presents each matrix on the screen separately. The scroll menu allows you to move from one matrix to another. It contains three options for moving:

a. **Next** (also goes with Ctrl+n key). Moves you to the next matrix in the network index (NI) sequence.

b. **Previous** (also goes with Ctrl+p key). Moves you to the previous matrix in the NI sequence.

c. **Jump** (also goes with the Ctrl+j key). Allows you to jump to a specified NI.
d. **First** (goes with the Ctrl+o key). Jumps to the first matrix in the file.
e. **Last** (goes with the Ctrl+i key). Jumps to the last matrix in the file.

6. STANDARDIZE MENU

The input data can be standardized into one of the following types:

a. **None.** Default. Data are unstandardized.

b. **Row Standardization**. Each entry s_{ij} in the matrix is standardized such that $s'_{ij} = s_{ij} / s_{.i} = s_{ij} / \sum_{j=1}^{n} s_{ij}$.
Each row-standardized cell entry is the unstandardized entry's proportion of its respective row.

c. **Column Standardization**. Each entry s_{ij} in the matrix is standardized such that $s'_{ij} = s_{ij} / s_{i.} = s_{ij} / \sum_{i=1}^{n} s_{ji}$.
Each row-standardized cell entry is the unstandardized entry's proportion of its respective column.

d. **Diagonal Standardization. There are a number of possible diagonal standardizations. These are:**

i. *Row.* $s'_{ij} = s_{ij} / s_{ii}$. Each entry is divided by the respective row diagonal.

ii. *Column.* $s'_{ij} = s_{ij} / s_{jj}$. Each entry is divided by the respective column diagonal.

iii. *Minimum.* $s'_{ij} = s_{ij} / \min(s_{ii}, s_{jj})$. Each entry is divided by the smallest of its respective row or column diagonals.

iv. *Maximum.* $s'_{ij} = s_{ij} / \max(s_{ii}, s_{jj})$. Each entry is divided by the largest of its respective row or column diagonals.

7. OPTIONS MENU

This menu defines the central options that are used to derive various SNA statistics. You may change the options at any time during the session. The change of options will affect subsequent procedures.

The Options menu consists of the following options (from top-left to bottom right):

a. **Save Method.** You can select whether to overwrite an existing file or to append the data produced by the current run to an existing file.

b. **Affiliation/Clique Display.** You can choose the number of cliques to display. Options are (1) Up to 500; (2) All cliques, or (3) Do not display cliques. This option is useful if you have files that produce a very large number of cliques.
c. **Cohesion Matrix Source.** When partitioning networks into subsets (cliques, blocks, communities, clusters, components), you may wisht to characterize each group in terms of cohesion due to the similarity, affinity, or some proximity attribute of the nodes in that group. The group cohesion index is given by:

$$GC = \frac{\sum_{i=1}^{n_i} a_{ij|i,j \in g}}{n_i(n_i - 1)}$$

Where *GC* is the group cohesion of a given group, $a_{ij|i,j \in g}$ is the proximity or affinity attribute of nodes *i* and *j* that are elements of the group, and n_i is the number of nodes in that group.

This option allows you to specify the kind of variable that defines group cohesion. It includes the following possibilities:

i. External matrix file. If you choose this option, you are required to specify a matrix file that has the same network index(es) as the current file, and in which each matrix has the same dimension and the same row and column labels as the matrix in memory.
ii. External dyadic file. Same as above, except that the file is a dyadic file (as shown in the load file menu).
iii. Structural equivalence matrix. See in the matrix menu. This option defines group cohesion endogenously based on a standardized Euclidean distance structural equivalence measure. It first calculates a Standardized Euclidean Distance Structural Equivalence (SEDSE) matrix, and then selects the entries in the SEDSE matrix corresponding to group members to calculate group cohesion.

d. **Attributes File.** Certain network operations require an external attribute file. If you wish to measure network or group sizes in terms of a given attribute (for example, the relative capabilities of states in an international network, or the proportion of seats in parliament in a party system network, or the wealth of people in a friendship network), you need to specify an external file that has the structure of a monadic file (see **LOAD** menu instructions). Once such a file is selected, you can mark the type of GPOL statistic you wish to use.
e. **Viable Coalition Cutoff.** This is a cutoff value for the coalition program (see coalitions menu below). Here you can specify a cutoff point that will cover all networks in the run, or you can use an external file. The structure of the external file is the following:

Network Index	Cutoff Value
2021	0.5
2022	0.2
2023	1.5
2024	0.75

f. **ERPOL TYPE.** The ERPOL index (discussed below) has a parameter *a* whose values range between 0.25 and 1. Here the user can define the value of this parameter either exogenously, by specifying a specific value (this value will apply to all matrices in the file), or to derive it endogenously, where $\alpha_r = GC_r$.

g. **Transitivity Type.** The transitivity index (see matrix program) is an index based on triadic relations. A transitive triad is $t_{ijk} = 1$ *iff* $i \leftrightarrow j \leftrightarrow k \leftrightarrow i$. In other words, a non-directional tie between *i* and *j* and a non-directional tie between *i* and *k* implies also a tie between *j* and *k*. In order to measure transitivity in directional and valued network, a rule must be defined that binarizes and symmetrizes valued on directional links. This depends on the user's interest. Therefore there are a number of options.
   i. **Simple Link Transitivity.** A triad is considered transitive ($t_{ijk} = 1$) if $s_{ij}>0$ and $s_{ik}>0$ and $s_{jk}>0$.
   ii. **Weak Link Transitivity.** $t_{ijk} = 1$ if $s_{ij}>0$, $s_{ik}>0$ and $s_{jk}>min(s_{i(jk)})$ where $min(s_{i(jk)})$ is the minimum value of the link between *i* and *j* or *i* and *k*.
   iii. **Strong Link Transitivity.** $t_{ijk} = 1$ if $s_{ij}>0$, $s_{ik}>0$ and $s_{jk}>max(s_{i(jk)})$ where $max(s_{i(jk)})$ is the maximum value of the link between *i* and *j* or *i* and *k*.

h. **Network Characteristics.** This set of columns specifies which network characteristics we wish to select in the network characteristics file or program. Note that some of the characteristics are shaded and can be selected only if an appropriate option (e.g., cohesion, attributes file) is selected prior to the selection of the network characteristics. There are also two general options (deselect all and select all). Default is all allowable characteristics are selected.

   The network characteristics selected are explained in the file—save menu.

i. **Reachability.** The reachability matrix is defined as $R^m = \sum_{i=1}^{m} S^i$ where *S* is the sociomatrix, and $m = 1 \ldots n-1$ is the order of the matrix. Thus, S^1 is the first order matrix (the original sociomatrix) that contains only direct ties (friends, neighbors, allies, trading partners), S^2 matrix is the second-order matrix that contains second-order (friend of friend, neighbor of neighbor, ally of ally, trading partner of trading partner) ties, and so forth. The R^m matrix contains all ties that are reachable from level 1 to level *m*. The options here allow the user to do a number of things.
   - Define the value of the maximum power matrix *m*. The user can select a number (that can vary between 1 to *n-1*). This will apply to all matrices in the input dataset. Alternatively, the user can insert a vector file (with structure specified in the input monadic matrix file) that contains a different *m* for each matrix in the input file.
   - Define if one wishes to generate the sum of the matrices from 1 to *m* or just display the m^{th} order matrix.
   - Decide if the diagonal of S^i is set to zero so as to eliminate self-ties in higher-order matrices.

j. **Matrix Density.** When calculating the density of submatrices based on groups, it may be useful to define a maximum density value for a given group. For example, suppose a network is split into four blocks, and we wish to use the blockmatrices for

blockmodeling. The density of each block (given a valued sociomatrix) would vary depending on the maximum strength of ties within each block. In that case, the overall maximum value in the sociomatrix may not be appropriate. We therefore wish to specify the maximum density for each block. This is done by using an external file with the following structure:

Network Index	Block No.	Max Value of ties
2021	1	0.5
2021	2	1
2021	3	0.8
2021	4	0.9
2022	1	0.2
2022	2	1.5
2022	3	0.75

Alternatively, the default procedure uses the maximum value of relationship in the original sociomatrix to calculate densities for all subgroups.

k. **Clique Options.** There are several options for clique extraction. Because cliques are based on binary and symmetric matrices, a number of decisions have to be made by the user in the case of directional and/or valued and/or signed networks.

l. **Cutoffs for clique extraction**. The default value for defining a relationship in the clique extraction program is 0.0. This means that any relationship that has a value of $s_{ij}>0$ is treated as 1. You can change this value manually, by inserting a number different from zero in the appropriate window. In this case, the program will use the same cutoff value for all the matrices in the dataset. Alternatively, if you wish to use different cutoffs for different networks in the dataset, you can define an external (.csv) file that has the following structure:

Network Index	**Cutoff**
2220	0.2
2221	0.5
2222	0.1
2223	0.8

You insert the file name at the prompt for Use Cutoff file.

m. **Minimum clique members**. The default value is 1. This means that that each isolate in a clique of its own, dyads form two-member cliques, and so forth. You can select the minimum number of members per clique or use an external file with the following structure.

Network Index	**Min. No. of Clique Members**
2220	1
2221	2
2222	3
2223	2

n. **K-Cliques.** A K-clique is a clique composed of nodes that are reachable at level *k* = *1, n–1*. A 2-clique is a clique of nodes reachable directly or at second order (friends and friends of friends), a 3-clique is composed of nodes that are reachable directly or indirectly at third-order (friends, friends of friends, and friends of friends of friends). Here too, you can specify a specific value that applies to all networks in the file, or use an external file that specifies a different *k*-order for each network in the file. In addition, you can decide whether you wish to eliminate self-ties by calculating the *k*-order matrices for the clique extraction or leave them in.

o. **Clique Extraction.** As noted, cliques are extracted only from binary and symmetric networks. When dealing with non-symmetric and/or valued networks, users need to apply a decision rule to convert the network into a binary and symmetric matrix. This suggests several options.

i. *Max*. In binary directional matrices, values in a symmetrized matrix are defined as

$$s'_{ij} = \begin{cases} 1 & iff\ s_{ij} \geq c\ \&\ s_{ji} \geq c \\ 0 & otherwise \end{cases}$$

ii. *Option 2: Upper*. Here symmetrized values are defined as

$$s'_{ij} = \begin{cases} 1 & iff\ s_{ij} \geq v\ \&\ s_{ji} \geq v \\ 0 & otherwise \end{cases}$$

iii.

specify a value in the window next to the option; this value will apply to all matrices in the dataset. Or you can use an external (.csv) file that has the following structure:

Network	Max density value
2220	1
2221	0.7
2222	55
2223	0.3

This will define varying maximum density values for each matrix according to its network index.

p. **Cohesion Matrix Source**. This option allows you to specify whether to use clique cohesion indices in the calculations of **NPOL** (see matrix menu), and if so—to define the source from which cohesion data will be obtained. There are several options here:

i. None (do not use cohesion scores)

ii. External matrix file. The file should have the same format as the input matrix format and its (Network Identifiers) NIs should match exactly those of the NIs in the current input file. If you choose this option, you must click the **Select File** command next to the option and specify the matrix file containing the cohesion scores.

iii. External dyadic file. The file should have the same format as the input dyadic file and its NIs should match exactly those of the NIs in the current input file. If you choose this option, you must click the **Select File** command next to the option and specify the dyadic file containing the cohesion scores.

iv. Structural equivalence. This option uses the current input file as the source for generating standardized Euclidean distance structural equivalence scores. You do not need to specify an external file here.

v. Use **NPOL Size**. In order to calculate **NPOL** based on an attribute file that specifies the relative size of units in each clique—rather than the number of units in each clique—you need to select an external (.csv) file that has a monadic format from which relative sizes of units are derived. The monadic file should have the following format

Network Index	Row Index	Unit Rel. Size
2220	A	0.5
2220	B	0.3
2220	C	0.1
2220	D	0.1
2221	A	0.1
2221	B	0.1
2221	C	0.4
2221	D	0.4

vi. Use **NPOL Coh. Size.** This option specifies the use of both cohesion scores (that may be obtained from an external file or defined endogenously—see above in the Cohesion matrix source) and size data derived via a user selected relative size file, as for **NPOL Size**.

q. **Sum/Mean Attributes**. When you select files that denote the attributes of the cliques, the program defines the clique attributes on the basis of user specified statistics. The external vector or matrix files contain characteristics of units or of dyads. The program aggregates these characteristics into a clique-level characteristic. The nature of this aggregation may be specified by the user. Several options are available.

i. None. The default option.

ii. Mean. This generates the mean of the Clique Characteristics Vector (CCV) or the mean of all dyadic values in a given clique if you specify a Dyadic Clique Characteristics Matrix (DCM).

iii. Sum. This generates the sum of the Clique Characteristics Vector (CCV) or the sum of all dyadic values in a given clique if you specify a Dyadic Clique Characteristics Matrix (DCM).

r. Clique Extraction Options. Given a nonsymmetrical matrix ($s_{ij} \neq s_{ji}$), the clique extraction algorithm requires symmetricizing the matrix. This set of options defines the rules for symmetrization.

i. Option 1 **max.** The Semmetricized matrix S' is defined such that for each cell of S' (s'_{ij}) we have:

$$s'_{ij} = s'_{ij} \begin{cases} 1 & \text{if } s_{ij} > c \text{ or } s_{ji} > c \\ 0 & \text{otherwise} \end{cases}$$

where *c* is the cutoff value specified above. This implies that whenever one of the cells in *S*, s_{ij} or s_{ji} assumes a value exceeding the cutoff, the symmetricized matrix assigns a value of 1 to both corresponding cells.

ii. Option 2 **upper.** The symmetricized matrix *S'* is defined such that for each cell of *S'* (s'_{ij}) we have:

$$s'_{ij} = s'_{ji} = \begin{cases} 1 & \text{if } s_{ij|j \geq i} > c \\ 0 & \text{otherwise} \end{cases}$$

where *c* is the cutoff value specified above and $i, j \in N$ are index numbers of rows and columns in the *S* matrix respectively. This implies that, regardless of the value of s_{ji}, the value of both s'_{ij} and of s'_{ji} assume the value of *1* or 0 based on whether s_{ij} exceeds the cutoff, but this applies only to cells whose column index number is equal to or higher than the row index number.

iii. Option 3 **lower.** The symmetricized matrix *S'* is defined such that for each cell of *S'* (s'_{ij}) we have:

$$s'_{ij} = s'_{ji} = \begin{cases} 1 & \text{if } s_{ij|j \leq i} > c \\ 0 & \text{otherwise} \end{cases}$$

where *c* is the cutoff value specified above and $i, j \in N$ are index numbers of rows and columns in the *S* matrix respectively. This implies that, regardless of the value of s_{ji}, the value of both s'_{ij} and of s'_{ji} assume the value of *1* or 0 based on whether s_{ij} exceeds the cutoff, but this applies only to cells whose column index number is equal to or lower than the row index number.

iv. Option 4 **min.** The Semmetricized matrix S' is defined such that for each cell of S' (s'_{ij}) we have:

$$s'_{ij} = s'_{ij} \begin{cases} 1 & \text{if } s_{ij} > c \ \& \ s_{ji} > c \\ 0 & \text{otherwise} \end{cases}$$

where *c* is the cutoff value specified above. This implies that only when both the cells in *S*, s_{ij} and s_{ji} assumes a value exceeding the cutoff, the symmetricized matrix assigns a value of 1 to both corresponding cells.

6. **Network Operations Menu (currently labeled Matrix menu)**

The Networks Operations Menu performs most of the SNA operations. It is useful before using this menu to go to the options menu and specify whatever options you need in order to perform various operations. **Note:** Operations in this menu are performed on the current sociomatrix or affiliation matrix that is on display. To move to another matrix use the **Scroll** menu. When scrolling, the current operation will be automatically performed on the matrix to which you have chosen to move.

a. **Default.** Data are presented in the input format. If you choose to standardize the data, they will be standardized according to the standardize option selected, and subsequent matrix operations will be performed on the standardized data.

b. **Dependency Matrix.** This matrix produces dependency data using the procedure described in Maoz (2009 and 2010: 85-89). For a sociomatrix S of order n, the dependency procedure requires the user to specify the number of iterations for the reachability matrix. This number can range from 1 (which uses the first-order sociomatrix $S=S^1$), 2 (which uses the sum of the first-order (S^1) and second-order (S^2) matrices, and up to n-1 ($D=S^1+S^2+\ldots S^{n-1}$). The resulting matrix is a $(n+1) \times (n+1)$ matrix with that has the following characteristics:

(i) $d_{ij} \neq d_{ji}$ denotes the dependence of the column entry j on the row entry i. It is defined as $d_{ij} = \dfrac{r_{ij}}{\sum_{i=1}^{m} n^{i-1}k^i} = \dfrac{(nk-1)r_{ij}}{k(nk)^m}$

Where k is the maximum value of a relationship in the original sociomatrix, r_{ij} is the reachability score (at order m) between nodes i and j, and n is the dimension of the sociomatrix S.

(ii) d_{ii} denotes the self-dependence of unit i,

(iii) Entry $d_{i(n+1)}$ is the outdependence of node i, that is, the dependence of all other nodes in the network on node i (including i's self-dependence). It is defined as

$$OUTD_i = \frac{[1-k(n-1)]\sum_{j=1}^{n} r_{ij}}{k(n-1)-[k(n-1)]^{(m+1)}}$$

With the notation the same as (i) above.

(iv) Entry $d_{(n+1)j}$ is the ondependence score of unit i, defined as:

$$OND_j = \frac{[1-k(n-1)]\sum_{i=1}^{n} r_{ji}}{k(n-1)-[k(n-1)]^{(m+1)}}$$

(v) Entry $d_{(n+1)(n+1)}$ is the Systemic Interdependence Score SYSIN and is defined as

$$SYSIN = \frac{[1-k(n-1)]\sum_{i=1}^{n}\sum_{j=1}^{n} r_{ij}}{nk(n-1)-[k(n-1)]^{(m+1)}}$$

c. **Reachability Matrix (R)**. The reachability matrix is an $n \times n$ matrix whose entries r_{ij} specify by how many paths unit j could be reached from unit i. Diagonal entries, r_{ii} specify reachability *cycles*, paths leading from a unit to itself through other units. This procedure allows the user to specify the number of iterations used for the reachability matrix. For example, given a sociomatrix S, specifying 3 in the option implies that $R = S^1 + S^2 + S^3 = \sum_{i=1}^{3} S^i$. Likewise, specifying 5 in the option implies that $R = \sum_{i=1}^{5} S^i$.

The submenu of the reachability matrix contains two options: a binarized reachability matrix, is a matrix R^b whose entries are defined as $r_{ij}^b = 1 iff\ r_{ij} > 0$ and zero otherwise. The valued matrix is the matrix defined above.

d. **Components Matrix.** A component is a subset of reachable nodes. Two nodes can be reachable directly ($s_{ij}>0$) or indirectly ($r_{ik} > 0$ *if* $s_{ik} = 0$, $s_{ij} > 0$, $s_{jk} > 0$). This command produces a component affiliation matrix of order $n \times k$ (where k is the number of components). The entries in the components affiliation matrix are 1 if node i is a member of component j and zero otherwise. Note: Components are discrete—a given node can be in one and only one components, two distinct components cannot have a member in common.

e. **Network Power Matrix.** The power of a node is the probability of the node being pivotal in winning subgroups induced by the network (Maoz 2011b). The general structure of network power is explained in the source. The network power menu has a submenu with several options:

(i) The user first selects the kind of subgroup he/she wishes to use to measure network power. One can select cliques, blocks, clusters, or communities. If the user selects blocks, the block menu requires to select types of blocks—those based on structural equivalence or on role equivalence.

(ii) Nodal attributes. Users define whether they wish to use the default. This assings each node an attribute of $a_i = 1/n$. Alternatively, the user can insert an attribute file by selecting this option. This enables the user to define nodal attributes exogenously (so that attributes vary across nodes and across networks).

(iii) If users select the block-based network power, they also need to decide whether the CONCOR program that produces block

stops iterating at a certain maximum correlation level (the default is 0.9 but user can specify any number from -1 to +1), and how many steps of block production are required. **Note**: typically, the larger the number of steps, the more blocks are produced. (The default is 2 but any positive integer can be inserted).

(iv) Finally, users can select whether they wish to generate the spoiling power statistic. Spoiling Power (Maoz 2011c) is an index of the ability of a given node to reduce the network power of other nodes by breaking ties with them. **Note**: selecting this option increases considerably the computing time of the network power index. It is not advised to select it with large networks.

f. **Single Matrix Structural Equivalence**. This program produces structural equivalence scores for single matrices. Given a sociomatrix ***X***, there are three options for producing structural equivalence scores.

i. **Correlations.** The output of this procedure is a structural equivalence matrix of Pearson product-moment correlation coefficients, SE_c. Each entry in this matrix is defined as:

$$se_{ij} = se_{ji} = \frac{\sum_{k=1}^{n}(x_{ik} - \bar{x}_{\bullet i})(x_{jk} - \bar{x}_{\bullet j}) + \sum_{k=1}^{n}(x_{ki} - \bar{x}_{i\bullet})(x_{kj} - \bar{x}_{j\bullet})}{\sqrt{\sum_{k=1}^{n}(x_{ik} - \bar{x}_{\bullet i})^2 + \sum_{k=1}^{n}(x_{ki} - \bar{x}_{i\bullet})^2}\sqrt{\sum_{k=1}^{n}(x_{jk} - \bar{x}_{\bullet j})^2 + \sum_{k=1}^{n}(x_{kj} - \bar{x}_{j\bullet})^2}}$$

where $\bar{x}_{\square i}$ is the mean of the *i*th row of the matrix and $\bar{x}_{i\square}$ is the mean of the *i*th column of the matrix ($\bar{x}_{j\square}$ *and* $\bar{x}_{\square j}$ are defined in the same way for the *j*th row and column, respectively). The se_{ij} scores here are defined in the range [-1,+1] where higher *se* scores indicate *higher* structural equivalence.

ii. **Raw Euclidean Distance**. The output of this procedure is a structural equivalence matrix of raw Euclidean distances. Each entry in the output matrix SE_{reu} is the raw Euclidean distance score between the row and column, and is defined by:

$$se_{ij} = se_{ji} = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2 + \sum_{k=1}^{n}(x_{ki} - x_{kj})^2}$$

Note that higher se_{ij} scores in this option are defined in the range of [0,∞] with higher *se* scores indicating *lower* level of structural equivalence.

iii. **Standardized Euclidean Distance.** The output of this procedure is a structural equivalence matrix of standardized (in the range of [0,1]) Euclidean distance scores. The entries here are defined as:

$$se_{ij} = se_{ji} = 1 - \frac{\sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2 + \sum_{k=1}^{n}(x_{ki} - x_{kj})^2}}{2n \max(x_{ik} - x_{jk})}$$

Where $\max(x_{ik} - x_{jk})$ is the maximum distance between any two entries in the matrix. Here structural equivalence increases with the size of se_{ij}.

Note: output SE matrices can also be scrolled via the **scroll menu** or they can be saved via the **save as** menu.

g. **Multi-matrix Structural Equivalence**

This program is equivalent to the simple structural equivalence program, for multiple matrices. For example, consider a set of $\Re = [r_1, r_2, \ldots, r_r]$ sociomatrices of size $n \times n$, all having the same units in the same order. Suppose r_1 is a friendship matrix (with entry r_{1ij} indicating whether person i is a friend of person j; r_2 is a neighborhood matrix (entry r_{2ij} indicates whether person i lives in the same block as person j), and r_3 is a soccer team membership matrix (r_{3ij} indicates whether person i is in the same soccer team as person j). The program measures structural equivalence for any pair of units across all three matrices. The options here are the same as for the single-matrix structural equivalence program:

i. **Multiple Correlation Structural Equivalence.** This is given by:

$$SE_{ij}^{R} = \frac{\sum_{r=1}^{2R}\sum_{k=1}^{n}(x_{ikr} - \bar{x}_{ir\bullet})(x_{jkr} - \bar{x}_{jr\bullet})}{\sqrt{\sum_{r=1}^{2R}\sum_{k=1}^{n}(x_{ikr} - \bar{x}_{ir\bullet})^2}\sqrt{\sum_{r=1}^{2R}\sum_{k=1}^{n}(x_{jkr} - \bar{x}_{jr\bullet})^2}}$$

where $R=[1,\ldots,r]$ is a set of relationships. Since each matrix r can be transposed, the multiple correlation includes the transpose of the relations. The *SE* scores here are defined in the range [-1,+1] where higher *SE* scores indicate *higher* structural equivalence.

ii. **Euclidean Distance Structural Equivalence.** This is given by:

$$SE_{ij}^{R} = SE_{ji}^{R} = \sqrt{\sum_{r=}^{r}\sum_{k=1}^{n}[(x_{ikr} - x_{jkr})^2 + (x_{kir} - x_{kjr})^2]}$$

Note that higher se_{ij} scores in this option are defined in the range of [0,∞] with higher *se* scores indicating *lower* level of structural equivalence.

iii. **Standardized Euclidean Distance Structural Equivalence.** This is given by

$$SE_{ij}^{R} = SE_{ji}^{R} = 1 - \frac{\sqrt{\sum_{r=1}^{R}\sum_{k=1}^{n}(x_{ikr} - x_{jkr})^2 + (x_{kir} - x_{kjr})^2}}{2Rn\sum_{r=1}^{R}\max(x_{ikr} - x_{jkr})}$$

Here structural equivalence varies in the range of [0,1] and increases in SE_{ij}.

h. **Centrality Indices Matrix**. This program calculates various centrality scores of units of a given sociomatrix. Before calculating these indices, the program requires the user to define three options: (1) the value of $s_{ij(max)}$, that is the maximum value of any of the entries in the sociomatrix *S,* (2) in betweenness centrality (defined below) it requires the user to specify whether cycles are to be included, and (3) in the course of generating centrality indices, for some purposes diagonal values need to be zeroed (self-links are ignored); the option asks whether to do this in cases where sociomatrices have nonzero diagonal elements.[2] The centrality indices program contains the following measures of centrality.

i. **Degree Centrality:** Degree centrality is the centrality of a given node in terms of the number of nodes to which it is directly connected. There are two measures of centrality **OutDegree Centrality**, measured for each node $i \in N$ as: $DO_i = \sum_{j=1}^{n} s_{ij} \Big/ n[\max(s_{ij})]$. This measures the centrality of outgoing links between a node i and other nodes in the network. **InDegree Centrality (**that is also used as a measure of *Degree Prestige*)**,** measured for each node $j \in$ N as: $DO_j = \sum_{i=1}^{n} s_{ji} \Big/ n[\max(s_{ij})]$.

ii. **Closeness Centrality.** This measures centrality in terms of the closeness of nodes (the number of vertices separating them defines their closeness). Here too we have **OutCloseness**, defined as $CO_i = (\sum_{j=1}^{n} r_{ij} - r_{ii}) / [(n-1)\max(s_{ij})]$ (where r_{ij} is an element of the reachability matrix *R, j* indexes columns, and $s_{ij(max)}$ is specified by the user), and **InCloseness**, defined as $CI_j = (\sum_{i=1}^{n} r_{ji} - r_{jj}) / [(n-1)\max(s_{ij})]$ (where r_{ij} is an element of the reachability matrix *R, i* indexes rows and $s_{ij(max)}$ is specified by the user).

---

2 A cycle is a link of a node to itself through at least one other node.

iii. **Betweenness Centrality**. A node *i* is said to have a betwenness value of 1 if $s_{ij} \neq 0$ and $s_{ik} \neq 0$. In this case, *i bridges between j and k*. Betweenness centrality measures the ratio of cases where a given node serves as a bridge between two other nodes and the possible number of bridges for this node. Here too we have **Out Betwenness Centrality,** measuring only the links where the outgoing ties of a given node serve as a bridge between other nodes, and **InBetwenness Centrality** that measure incoming nodes.

iv. **Eigenvector Centrality**. This index weights the degree centrality scores of a node by the degree centrality of the nodes to which it is connected. Here too we have **OutEigenvector Centrality**, based on outgoing ties, and **InEigenvector Centrality**, based on the incoming ties.

The output for this program is a $n \times 8$ table that lists the **Out** and **In** centrality scores for each of the *n* nodes.

i. **Unit Dependency Matrix**. This procedure produces unit dependency data (see **Dependency Matrix**) in an $n \times 6$ output matrix. The output matrix is organized as follows.

Network Index	Unit	*Rwoutd*	*Rwind*	*Stdoutd*	*Stdind*

j.

k. **Clique Affiliation (CA) Matrix.** This procedure extracts cliques according to the **clique extraction options** specified in the **Options** menu. The output in the matrix window is the $n \times k$ clique affiliation (*CA*) matrix of the current matrix in the data file. Each entry ca_{ij} is 1 if row-unit *i* is a member of column-clique *j*, and zero otherwise. You can use the **scroll menu** to scroll through clique affiliation matrices, and use the **save as** menu to save the clique affiliation matrices.

l. **Clique Member Overlap (CMO) Matrix.** This is a $n \times n$ matrix (obtained as $CMO = CA \times CA'$) that has the following characteristics: (i) $cmo_{ij}=cmo_{ji}$ denotes the number of cliques that units *i* and *j* share in common, and (ii) cmo_{ii} is the number of cliques that have unit *i* as a member. You can use the **scroll menu** to scroll through clique member overlap matrices, and use the **save as** menu to save the clique member overlap matrices.

m. **Clique-by-Clique Overlap (CCO) Matrix.** For a network of size *m* with a clique affiliation (*CA*) matrix of dimensions $n \times k$, the clique-by-clique overlap (obtained as: $CCO=CA' \times CA$) is a matrix of dimensions $k \times k$ that has the following characteristics: (i) $cco_{ij}=cco_{ji}$ is the number of units that cliques *i* and *j* share in common, and (ii) cco_{ii} is the number of members in clique *i*. You can use

the **scroll menu** to scroll through clique-by-clique overlap matrices, and use the **save as** menu to save the clique-by-clique overlap matrices.

n. **Dependency Matrix (D).** This matrix produces dependency data using the procedure described in Maoz (2006a and 2006b). For a sociomatrix S of order n, the dependency procedure requires the user to specify the number of iterations for the reachability matrix. This number can range from 1 (which uses the first-order sociomatrix $S=S^1$), 2 (which uses the sum of the first-order (S^1) and second-order (S^2) matrices, and up to n-1 ($D=S^1+S^2+\ldots S^{n-1}$). The resulting matrix is a $(n+3) \times (n+3)$ matrix with that has the following characteristics: (i) $d_{ij} \neq d_{ji}$ denotes the dependence of the column entry j on the row entry i, (ii) d_{ii} denotes the self-dependence of unit i, (iii) Entry $d_{i(n+1)}$ is the total outdependence of unit i, that is, the total dependence of all units in the network on unit i (including i's self-dependence), (iv) Entry $d_{i(n+2)}$ is the standardized outdependence of unit i, defined as: $stoutd_i = (\sum_{j=1}^{n} d_{ij} - d_{ii}) \Big/ \sum_{j=1}^{n} d_{ij}$, (v) Entry $d_{i(n+3)}$ is the *raw outdependence* defined as $rwoutd_i = \sum_{j=1}^{n} d_{ij} - d_{ii}$, and defines the total dependence of all units in the network on unit i, excluding i's self-dependence. (vi) Entry $d_{(n+1)j}$ is the *total ondependence* of unit j, which means the total dependence of unit j on other units in the system, (vii) Entry $d_{(n+2)j}$ is the *standardized ondependence* calculated in the same way as the *standardized outdependence* for the columns of the matrix. (viii) Entry $d_{(n+3)j}$ is the *raw ondependence* defined in the same way as *rwoutd* for the appropriate column. (ix) Entry $d_{(n+2)(n+2)}$ is the system average of $sdtoutd_i$ and $stdond_j$ and (x) Entry $d_{(n+3)(n+3)}$ is the system average of $rwoutd_i$ and $rwond_j$. Dependency matrices can also be scrolled via the **scroll menu** or they can be saved via the **save as** menu.

o. **Reachability Matrix (R)**. The reachability matrix is an $n \times n$ matrix whose entries r_{ij} specify by how many paths unit j could be reached from unit i. Diagonal entries, r_{ii} specify reachability *cycles*, paths leading from a unit to itself through other units. This procedure allows the user to specify the number of iterations used for the reachability matrix. For example, given a sociomatrix S, specifying 3 in the option implies that $R = S^1 + S^2 + S^3 = \sum_{i=1}^{3} S^i$. Likewise, specifying 5 in the option implies that $R = \sum_{i=1}^{5} S^i$.

p. **Single Matrix Structural Equivalence**. This program produces structural equivalence scores for single matrices. Given a sociomatrix X, there are three options for producing structural equivalence scores.

i. **Correlations.** The output of this procedure is a structural equivalence matrix of Pearson product-moment correlation coefficients, SE_c. Each entry in this matrix is defined as:

$$se_{ij} = se_{ji} = \frac{\sum_{k=1}^{n}(x_{ik} - \bar{x}_{\bullet i})(x_{jk} - \bar{x}_{\bullet j}) + \sum_{k=1}^{n}(x_{ki} - \bar{x}_{i\bullet})(x_{kj} - \bar{x}_{j\bullet})}{\sqrt{\sum_{k=1}^{n}(x_{ik} - \bar{x}_{\bullet i})^2 + \sum_{k=1}^{n}(x_{ki} - \bar{x}_{i\bullet})^2}\sqrt{\sum_{k=1}^{n}(x_{jk} - \bar{x}_{\bullet j})^2 + \sum_{k=1}^{n}(x_{kj} - \bar{x}_{j\bullet})^2}}$$

where $\bar{x}_{\square i}$ is the mean of the *i*th row of the matrix and $\bar{x}_{i\square}$ is the mean of the *i*th column of the matrix ($\bar{x}_{j\square}$ *and* $\bar{x}_{\square j}$ are defined in the same way for the *j*th row and column, respectively). The se_{ij} scores here are defined in the range [-1,+1] where higher *se* scores indicate *higher* structural equivalence.

ii. **Raw Euclidean Distance**. The output of this procedure is a structural equivalence matrix of raw Euclidean distances. Each entry in the output matrix SE_{reu} is the raw Euclidean distance score between the row and column, and is defined by:

$$se_{ij} = se_{ji} = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2 + \sum_{k=1}^{n}(x_{ki} - x_{kj})^2}$$

Note that higher se_{ij} scores in this option are defined in the range of [0,∞] with higher *se* scores indicating *lower* level of structural equivalence.

iii. **Standardized Euclidean Distance.** The output of this procedure is a structural equivalence matrix of standardized (in the range of [0,1]) Euclidean distance scores. The entries here are defined as:

$$se_{ij} = se_{ji} = 1 - \frac{\sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2 + \sum_{k=1}^{n}(x_{ki} - x_{kj})^2}}{2n\max(x_{ik} - x_{jk})}$$

Where $\max(x_{ik} - x_{jk})$ is the maximum distance between any two entries in the matrix. Here structural equivalence increases with the size of se_{ij}.

Note: output SE matrices can also be scrolled via the **scroll menu** or they can be saved via the **save as** menu.

q. **Multi-matrix Structural Equivalence**

This program is equivalent to the simple structural equivalence program, for multiple matrices. For example, consider a set of $\Re = [r_1, r_2, \ldots, r_r]$ sociomatrices of size $n \times n$, all having the same units in the same order. Suppose r_1 is a friendship matrix (with entry r_{1ij} indicating whether person *i* is a friend of person *j*; r_2 is a neighborhood matrix (entry r_{2ij} indicates whether person *i* lives in the same block as person *j*), and r_3 is a soccer team

membership matrix (r_{3ij} indicates whether person *i* is in the same soccer team as person *j*). The program measures structural equivalence for any pair of units across all three matrices. The options here are the same as for the single-matrix structural equivalence program:

i. **Multiple Correlation Structural Equivalence.** This is given by:

$$SE_{ij}^{R} = \frac{\sum_{r=1}^{2R}\sum_{k=1}^{n}(x_{ikr} - \bar{x}_{ir\bullet})(x_{jkr} - \bar{x}_{jr\bullet})}{\sqrt{\sum_{r=1}^{2R}\sum_{k=1}^{n}(x_{ikr} - \bar{x}_{ir\bullet})^2}\sqrt{\sum_{r=1}^{2R}\sum_{k=1}^{n}(x_{jkr} - \bar{x}_{jr\bullet})^2}}$$

where $R=[1,\ldots,r]$ is a set of relationships. Since each matrix *r* can be transposed, the multiple correlation includes the transpose of the relations. The *SE* scores here are defined in the range [-1,+1] where higher *SE* scores indicate *higher* structural equivalence.

ii. **Euclidean Distance Structural Equivalence.** This is given by:

$$SE_{ij}^{R} = SE_{ji}^{R} = \sqrt{\sum_{r=}^{r}\sum_{k=1}^{n}[(x_{ikr} - x_{jkr})^2 + (x_{kir} - x_{kjr})^2]}$$

Note that higher se_{ij} scores in this option are defined in the range of [0,∞] with higher *se* scores indicating *lower* level of structural equivalence.

iii. **Standardized Euclidean Distance Structural Equivalence.** This is given by

$$SE_{ij}^{R} = SE_{ji}^{R} = 1 - \frac{\sqrt{\sum_{r=1}^{R}\sum_{k=1}^{n}(x_{ikr} - x_{jkr})^2 + (x_{kir} - x_{kjr})^2}}{Rn^2\sum_{r=1}^{R}\max(x_{ikr} - x_{jkr})}$$

Here structural equivalence varies in the range of [0,1] and increases in SE_{ij}.

r. **Centrality Indices Matrix**. This program calculates various centrality scores of units of a given sociomatrix. Before calculating these indices, the program requires the user to define three options: (1) the value of $s_{ij}(max)$, that is the maximum value of any of the entries in the sociomatrix *S,* (2) in betweenness centrality (defined below) it requires the user to specify whether cycles are to be included, and (3) in the course of generating centrality indices, for some purposes diagonal values need to be zeroed (self-links are ignored); the option asks whether to do this in cases where

sociomatrices have nonzero diagonal elements.[3] The centrality indices program contains the following measures of centrality.

i. **Degree Centrality:** Degree centrality is the centrality of a given node in terms of the number of nodes to which it is directly connected. There are two measures of centrality **OutDegree Centrality**, measured for each node $i \in N$ as: $DO_i = \sum_{j=1}^{n} s_{ij} \Big/ n[\max(s_{ij})]$. This measures the centrality of outgoing links between a node i and other nodes in the network. **InDegree Centrality (**that is also used as a measure of *Degree Prestige*)**,** measured for each node $j \in$ N as: $DO_j = \sum_{i=1}^{n} s_{ji} \Big/ n[\max(s_{ij})]$.

ii. **Closeness Centrality.** This measures centrality in terms of the closeness of nodes (the number of vertices separating them defines their closeness). Here too we have **OutCloseness**, defined as $CO_i = (\sum_{j=1}^{n} r_{ij} - r_{ii}) / [(n-1)\max(s_{ij})]$ (where r_{ij} is an element of the reachability matrix *R, j* indexes columns, and $s_{ij(max)}$ is specified by the user), and **InCloseness**, defined as $CI_j = (\sum_{i=1}^{n} r_{ji} - r_{jj}) / [(n-1)\max(s_{ij})]$ (where r_{ij} is an element of the reachability matrix *R, i* indexes rows and $s_{ij(max)}$ is specified by the user).

iii. **Betweenness Centrality**. A node *i* is said to have a betwenness value of 1 if $s_{ij} \neq 0$ and $s_{ik} \neq 0$. In this case, *i bridges between j and k*. Betweenness centrality measures the ratio of cases where a given node serves as a bridge between two other nodes and the possible number of bridges for this node. Here too we have **Out Betwenness Centrality,** measuring only the links where the outgoing ties of a given node serve as a bridge between other nodes, and **InBetwenness Centrality** that measure incoming nodes.

iv. **Eigenvector Centrality**. This index weights the degree centrality scores of a node by the degree centrality of the nodes to which it is connected. Here too we have **OutEigenvector Centrality**, based on outgoing ties, and **InEigenvector Centrality**, based on the incoming ties.

The output for this program is a $n \times 8$ table that lists the **Out** and **In** centrality scores for each of the *n* nodes.

s. **Network Characteristics Data**. This procedures produces the **network characteristics data** for the current sociomatrix that are specified in the **options** menu.

Example. Suppose the input dataset consists of the following matrices, $S_{1(3\times2)}$, $S_{2(4\times5)}$, $S_{3(7\times3)}$ (expressions in the subscripted parentheses denote the dimensions of each matrix).

---

[3] A cycle is a link of a node to itself through at least one other node.

In order to multiply this dataset by another set of matrices *O*, the dataset to be inputted in the option must be: $O_{1(2\times a)}$, $O_{2(5\times b)}$, $O_{3(3\times c)}$ (where *a, b,* and *c* are any positive integers). Otherwise the procedure will crash. The output screen presents the first product matrix $T_{1(3\times a)} = S_1 \times O_1$. Other matrices in the resulting dataset can be displayed via the **Scroll Menu**.

t. **Binary Complement**. This procedure produces a binary complement of a sociomatrix *S* of order $n \times m$ a binary matrix that is a complement of the binary version of the *S* matrix. Let *SB* is the binary version of the matrix *S* as defined by the cutoff procedure in the **Options** menu. The binary complement matrix *BC* is a matrix such that *SB–BC=0*.

u. **Triadic Matrix.** This program generates a triadic census matrix of order $[n(n-1)(n-2)/6] \times 12$. The rows of the matrix are first designated by the network index (NI), and then the triad identifier (for example, in a matrix with $n = 10$ nodes, the rows are 123, 124, …, 1210, 134, 135,…, 1310,….8910). The entries of the matrix are seen below:

NI	Triad	A→ B	A→ C	B→ A	B→ C	C→ A	C→ B	AB C	AC B	BA C	BC A	CA B	BC A
1820	123	1	1	1	0	1	0	7	7	16	16	16	16
1820	124	1	0	1	0	0	0	6	6	6	6	11	11
1820	134	1	0	1	1	0	1	16	16	7	7	16	16
1820	234	0	0	0	1	0	1	11	11	6	6	6	6

The columns are structured in the following manner. The columns with two elements connected by an arrow (e.g., a→b, a→c) show directional relationships between the elements of the triads as listed in the row. For example, the in the first row a→b = 1 means that there is a link going from node #1 to #2. Likewise, the entry in the first row of the c→b column means that there is no link going from node #3 to node #2.

The entries of the columns are numbers going from 1 to 36. A given number reflects a type of triad in the Burt (1990) triadic classification. For example, the number 7 in the first row of the column labeled abc means that this is triad type 7 (which is 1↔2, 1↔3, 2∅3). likewise, triad number 16 in the first row of the column labeled bac implies that the structure of the triad 213 is: 2↔1, 2∅3, 1↔3.

v. **Role Equivalence Matrix**. Following Burt (1990: 86) role equivalence measures the equivalence, defined as a comparison between any two units in the network in terms of the convergence of their pattern of roles in the network as "a triad census, a pattern of relative frequencies with which kinds of triads describe the individual's orientation to others." Two individuals are said to be "role equivalent" to the extent that the distribution of their triadic census positions with others is identical. The current procedure uses standardized Euclidean

distances between the role positions of any two units in the networks. The role equivalence between two nodes, *i* and *j* is given by:

$$RE_{ij} = 1 - \frac{2\sqrt{\sum_{q=1}^{36}(t_{iq} - t_{jq})^2}}{(n-1)(n-2)}$$

Where t_{iq} and t_{jq} are the frequencies of triangle type *q* associated with nodes *i* and *j*, repectively, and *n* is the dimension of the Sociomatrix.

w. Distance matrix. The distance matrix is the matrix of the shortest distances between nodes, a distance of 1 in cell *ij* of the distance matrix means that nodes *i* and *j* are connected directly. A distance of 2 means that nodes *i* and *j* are connected at the second order (through their joint connection to a third node), and so forth. The distance matrix module contains a number of options.

- **i. Cost Matrix.** Assume that connecting to a node carries a cost. The cost is specified by the user (the default is 1). Denote the cost of connection as *c*. When node *i* connects directly to node *j*, the cost is *c*. If they are connected indirectly via a third node the cost is *2c*, and so forth.
- **ii. Distance Matrix.** As specified above.
- **iii. Strength Matrix.** Denote the strength of a tie going from *i* to *j* by $0 \leq v_{ij} \leq 1$. The tie going from one node may be only a direct tie, only an indirect tie, or both a direct and indirect tie. Consider a relationship of the sort $i \rightarrow j \rightarrow k \leftrightarrow a$. Here, *i* and *k* have both a direct tie of value *v* and an indirect tie of value v^2. The strength of the tie between *i* and *k* is therefore $v + v^2 = v(1+v)$. The strength matrix represents the strength of ties between two nodes that includes both direct and indirect ties.

7. Cliques Menu

Cliques are closed subsets of a network. A clique is composed of a subset of the nodes, all of which are tied to each other at level $s_{ij} \geq q$ where *q* is a user specified cutoff (the default is $q > 0$), at distance *k* ($1 \leq k \leq n-1$). The default for *k* is 1. Both *q* and *k* are defined in the cliques section of the **Options** menu. For example, in a binary network if the user does not change the default values of *q* and *k*, cliques are all groups of directly connected nodes. All members of a given clique have direct ties with all other members of the clique at a level $s_{ij}>0$. Cliques are not discrete. Two cliques can share 0, 1. …, *n-2* members. However, *no clique can be equal to or a proper subset of another clique.* Any two cliques, *q* and *r,* must differ with respect to at least two members, one in *q* but not in *r* and one in *r* but not in *q*. If $k = 2$, then cliques consist of all nodes that are tied either directly or indirectly via only one other node. The cliques menu contains a number of submenus that perform various clique-related operations and functions.

a. **Clique Affiliation Matrix**. The Clique Affiliation Matrix (CA) is a $n \times k$ binary matrix with entry ca_{ij} scoring 1 if node *i* is a member of clique *j* and zero otherwise. The clique affiliation matrix allows the user to specify some statistics for the CA matrix. These include:

i. Clique Size. If the user defines an attribute vector in the **Options** menu, the program returns the sum of the attributes of the nodes in each clique. This is done at row $n + 1$ of the CA matrix.

ii. Clique Cohesion. If the user defines a cohesion option (either structural equivalence or via an external file) in the **Options** menu, the program will return the clique cohesion. The Cohesion of clique j is defined as:

$$C_j = \frac{2\sum_{i=1}^{n_j-1}\sum_{k=i+1}^{n_j} c_{ik}}{n_j(n_j-1)}$$

Where c_{ik} is the externally or internally (structural equivalence) cohesion score between nodes i and k ($i, k \in j$), and n_j is the number of nodes in clique j.

iii. **Esteban/Ray Index**. This is the index of Duclos Esteban and Ray (2004) for clique j. It is defined as $ER_j = \left(\sum_{i \in j} a_i\right)^{1+\alpha}$ where a_i is the attribute value of node $i \in j$ and α is a user-defined exponent (default is $a = 1$).

b. **Clique membership Overlap Matrix**. This is a $n \times n$ **CMO** matrix, with elements cmo_{ij} denoting the number of cliques that nodes i and j share in common. The main diagonal element of the matrix, cmo_{ii} is the number of cliques of which node i is a member. This matrix can be diagonally standardized such that its entries reflect the proportion of joint memberships between nodes ij out of the cliques of which i is a member. The **CMO** matrix is symmetrical ($cmo_{ij} = cmo_{ji}\ \forall i, j \in \boldsymbol{CMO}$). However, the standardized **CMO** matrix is asymmetrical.

c. Clique-by-Clique Overlap Matrix (CO). The clique-by-clique overlap matrix (or Clique Overlap matrix in short) is a $k \times k$ matrix with entry co_{qr} reflecting the number of nodes that are common to cliques q and r. The diagonal entries of this matrix, co_{qq} are the number of nodes in clique q. This matrix too can be diagonally standardized with entries reflecting the proportion of the members of clique q shared by cliques q and r. Here too, the raw **CO** matrix is symmetrical; the standardized **CO** matrix is not.

d. Clique Characteristics Matrix. The clique characteristics matrix allows using the clique as the unit of obervation. It characterizes cliques via the attributes of nodes or the attributes of nodal ties. To activate this program, the user needs to specify external files with attributes, as well as the kind of clique-related statistics desired. The input files can be attribute vectors (a variable with network index, node label, and a single attribute variable), attribute matrices (network index, node lablels, and a set of attribute variables), or it can be a dyadic characteristics matrix (this is a matrix that specifies attributes of dyadic relationships (e.g., whether a dyad is jointly democratic, the volume of trade between members, etc.). The user can specify also which statistic he/she wishes to measure for the clique. These statistics include:

- Clique mean
- Clique standard deviation
- Clique minimum
- Clique maximum

e. **Inter-clique distance matrix.** This procedure specifies the relative distance between any two cliques in a given network in terms of the pattern of membership between a given clique c_i and all other cliques $c_{j \neq i}$. This is the standardized Euclidean distance matrix performed on the **CO** matrix that is derived from the sociomatrix S.

8. Blockmodels

The blockmodels menu (to be renamed as subgroups menu) allows extraction of different kind of subgroups from a network or set of networks. Specifically, beyond the cliques (discussed in the cliques menu), the program currently supports extraction of three types of subgroups: Blocks, clusters, and communities. Each set of subgroups comes with a number of submenus that perform various operations on the subgroup.

a. **Blocks.** Blockmodels partition the network into a set of groups composed of "roughly" equivalent nodes. The current method of block extraction is CONCOR. Given a sociomatrix *S* the Convergence of Correlations (CONCOR) procedure generates blocks (mutually exclusive subsets) of all units in the network based on iterative convergence of structural or role equivalence scores.

   i. **CONCOR menu.** The choice of this program opens a submenu with two options: *Correlation*—this uses a correlation-based structural equivalence matrix to generate blocks. *Standardized Euclidean Distance.* It uses the standardized Euclidean distance method to generate structural equivalence scores.
   ii. **CONCOR options.** Once the user has selected the method of measuring structural equivalence, the program opens a dialog box with the following steps: (1) The user specifies a correlation cutoff that serves as the minimum correlation between units *i* and *j* (se_{ij}) that assigns these units to the same block, (2) the user must specify the depth of the partitioning process in terms of the maximum number of steps that the partitioning program follows. **NOTE:** the larger the number of steps the user specifies, the more blocks might be produced. The default depth is two steps.

   The program generates a block affiliation matrix of size $n \times k$ (n nodes by k blocks). **Note:** blocks are discrete, each node can be in one and only one block; no two block can have a member in common. All isolates may be placed in the same block.

b. **Hierarchical Clustering.** The HC algorithm is similar to the blocks algorithm in principle, but it produces a different set of groups—clusters, via a hierarchical tree-process. Here too, the user must specify a number of parameters.
   i. **Number of clusters desired.** This tells the program when to stop partitioning nodes into clusters.
   ii. **Clustering method.** The HC program starts out with an input matrix from which it starts the clustering process. There are three choices here: (1) correlation—the program starts with a structural equivalence matrix based on correlations between nodal profiles; (2) Euclidean distances—here the program allows the user to create clusters made up of *most different* nodes; (3)

standardized Euclidean distances—the structural equivalence matrix is based on standardized ED scores.

Here too, the output is a clustering affiliation matrix of dimension $n \times k$. Clusters are discrete as well.

c. **Block partitioning matrix.** This procedure allows us to examine the relationships between nodes and blocks. It contains two options.

i. **Sociomatrix Entries.** This produces a "partitioned" sociomatrix. The order of the nodes in this sociomatrix corresponds to their block affiliation. For example, suppose a matrix of five nodes is partitioned into three blocks such that nodes 1 and 3 are in the first block, nodes 2 and 4 are in the second block, and node 5 is in the third block all by itself. The partitioned sociomatrix will be ordered such that nodes 1 and 3 will be first, followed by nodes 2 and 4, and ending with node 5. The ties in the original sociomatrix will be preserved, but now we can see how a given node is tied to nodes in its own block or in other blocks.

ii. **Block Identity Matrix.** This procedure assigns nodes to block-relationship. Using the previous example, the matrix is ordered 1,3,2,4,5. The identity matrix would look as follows

	1	**3**	**2**	**4**	**5**
1	11	11	12	12	13
3	11	11	12	12	13
2	21	21	22	22	23
4	21	21	22	22	23
5	31	31	32	32	33

Nodes that belong to the same block will have an entry of *kk* (where *k* is the block number); nodes that belong to a different block will have an entry of *kr* where *k* is the number of the row node and *r* is the number of the column node.

d. **Blockmatrices.** This set of procedures uses the image matrix (a block matrix of size $k \times k$) to characterize relationships within and between blocks. It contains several elements.

i. **Block densities.** This is the density of the nodes within an image. Entries in the main diagonal d_{kk} indicate within-block densities; off-diagonal entries d_{kr} indicate the density between nodes belonging to block *k* and nodes belonging to block *r*. Due to the manner in which blocks are constructed, it is possible to have zero within block densities and nonzero between block densities. Likewise, it is possible to have lower within-block densities than between-block densities.

ii. **Relative Densities.** This is a similar matrix to the block density matrix. However, entries in this matrix are a ratio of the density of an image to the network density. Thus entry $rd_{kr} = d_{kr} / \nabla$ where ∇ is the network density.

e. **Block characteristics.** This procedure is identical to the clique characteristics program, except that the unit of analysis is a block.

f. **Cluster partition matrix.** This procedure produces a cluster partition matrix with a structure identical to the block partition matrix.

g. **Cluster densities.** The menus and submenues here are the same as for the block densities program.

h. **Cluster characteristics.** Same as for the block and clique characteristics.

i. **Communities matrix.** A community is a subset of nodes that are sufficiently related to each other. The difference between communities and the other grouping methods is twofold. First, communities are different from blocks and clusters in that we do not need to restrict the number of groups that the program generates. In addition, blocks and clusters form discrete subsets of nodes; communities—like cliques—may be non-discrete. The difference between communities and cliques is also twofold. First, cliques form between k-connected nodes; all nodes in a clique must be connected at order 1,…,k. Communities may form between nodes that have no—direct or indirect—connection, but whose neighborhoods are very similar. Second, the communities program produces a smaller number of communities than a clique extraction algorithm for the same network. The communities program uses the Leicht and Newman (2008) algorithm.[4] The communities program contains a number of elements.
   i. **Community affiliation matrix.** This is a binary $n \times k$ affiliation matrix with entry ca_{ij} indicating membership of node i in community j.
   ii. **Community density matrix.** A matrix indicating the density of members in community k with members of community r.
   iii. **Community characteristics matrix.** Same as the clique, block, or cluster characteristics matrices.

9. Coalition analysis

****

x. **Clique Characteristics Matrix**. This program allows weighing cliques by various characteristics as specified in the **Options→Sum/Mean Attributes.** The output is a $k \times (m+2)$ table where the k rows represent the cliques in the network, the leftmost column is the NI, the next column is the number of members in clique i, and the following column represent for each DVC or CCM file specified in the options for this program the attributes of the clique for the variable specified in this file.

y. **Affiliation to Matrix Conversion.** This procedure converts affiliation matrices to sociomatrices. There are three possible ways to do this:

   i. **Sociomatrix conversion**. This is the traditional SNA approach to conversion. Given an affiliation matrix A of size $n \times k$, the conversion is done such that the resulting sociomatrix S (of size $n \times n$) is: $S = A \times A'$.

[4] I would like to thank Elizabeth Leicht of Oxford University for sharing this algorithm.

Each entry in S, s_{ij} is the number of events common to nodes i and j. Diagonal entries s_{ii} denote the number of events in which i is a member.

ii. **Correlation conversion**. This method converts an affiliation matrix A of size $n \times k$ into a sociomatrix S, whose entries s_{ij} vary in the [-1,+1] range, and reflect Pearson product-moment correlations between rows i and j in matrix A (diagonal entries s_{ii} are correlations of nodes with themselves, thus designated values of 1, by definition).

iii. **Standardized Euclidean distance**. This method converts an affiliation matrix A of size $n \times k$ into a sociomatrix S, whose entries s_{ij} vary in the [0,1] range and reflect standardized Euclidean distances between rows i and j in matrix A such that,

$$s_{ij} = s_{ji} = 1 - \frac{\sqrt{\sum_{k=1}^{n} (a_{ik} - a_{jk})^2}}{n[\max(a_{ik} - a_{jk})]}$$

(diagonal entries s_{ii} are distances of nodes with themselves, thus designated values of 1, by definition).

z. **Event Overlap Matrix**. This procedure converts an affiliation matrix A of order $n \times k$ into an event overlap matrix E of order $k \times k$, defined as $E = A' \times A$. Entries in E, e_{ij} reflect the number of members that cliques i and j share in common. Diagonal entries e_{ii} reflect the number of members in clique i.

aa. **Unit Dependency Matrix**. This procedure produces unit dependency data (see **Dependency Matrix**) in an $n \times 6$ output matrix. The output matrix is organized as follows.

Network Index	Unit	*Rwoutd*	*Rwind*	*Stdoutd*	*Stdind*

bb. **Network Characteristics Data**. This procedures produces the **network characteristics data** for the current sociomatrix that are specified in the **options** menu.

cc. **Matrix Multiplication**. This is a utility program allowing simple matrix multiplication of a series of matrices. It uses the current dataset that is composed of one or more matrices (of a general $m \times n$ order, with m and n varying from

one input matrix to another), and asks the user to specify another matrix (that can be inputted either as a matrices dataset or as a dyadic dataset). It then performs the multiplication operation.

Example. Suppose the input dataset consists of the following matrices, $S_{1(3\times2)}$, $S_{2(4\times5)}$, $S_{3(7\times3)}$ (expressions in the subscripted parentheses denote the dimensions of each matrix). In order to multiply this dataset by another set of matrices *O*, the dataset to be inputted in the option must be: $O_{1(2\times a)}$, $O_{2(5\times b)}$, $O_{3(3\times c)}$ (where *a, b,* and *c* are any positive integers). Otherwise the procedure will crash. The output screen presents the first product matrix $T_{1(3\times a)} = S_1 \times O_1$. Other matrices in the resulting dataset can be displayed via the **Scroll Menu**.

dd. **Elementwise Multiplication**. This procedure performs elementwise multiplication of the current matrix *S* by a matrix *M* or by a vector *V* that are specified by the user. The result is an elementwise multiplication of s_{ij} by m_{ij} or by v_i. For the procedure to work, the matrix *M* must be of the same dimensions as matrix *S*, or the vector must have the same number of rows as does *S*. The user can specify three forms of input for the multiplied datasets: (a) matrix file, (b) dyadic file, or (c) monadic file (vector).

Example. Given a dataset consisting of the following matrices $S_{1(3\times2)}$, $S_{2(4\times5)}$, $S_{3(7\times3)}$ (expressions in the subscripted parentheses denote the dimensions of each matrix), the dataset used for elementwise matrix multiplication must be $O_{1(3\times2)}$, $O_{2(4\times5)}$, $O_{3(7\times3)}$. If the user chooses a vector option, then the vectors must be $V_{1(3\times1)}$, $V_{2(4\times1)}$, $V_{2(7\times1)}$. The output screen presents the first product matrix $T_{1(3\times a)} = S_1 \times O_1$. Other matrices in the resulting dataset can be displayed via the **Scroll Menu**.

ee. **Binary Complement**. This procedure produces a binary complement of a sociomatrix *S* of order $n \times m$ a binary matrix that is a complement of the binary version of the *S* matrix. Let *SB* is the binary version of the matrix *S* as defined by the cutoff procedure in the **Options** menu. The binary complement matrix *BC* is a matrix such that *SB–BC=0*.

ff. **Inter-Clique Distance Matrix**. This procedure examines the relative distance between any two cliques in a given network in terms of the pattern of membership between a given clique c_i and all other cliques $c_{j\neq i}$. This is in fact a standardized Euclidean distance matrix performed on the **CCO** matrix that is derived from the sociomatrix S.

gg. **CONCOR Matrix**. Given a sociomatrix *S* the Convergence of Correlations (CONCOR) procedure generates blocks (mutually exclusive subsets) of all units in the network based on iterative convergence of structural equivalence scores based on correlations. It proceeds in the following steps: (1) The user specifies a correlation cutoff that serves as the minimum correlation between units *i* and *j* (se_{ij}) that assigns these units to the same block, (2) the program generates the *SE* correlation matrix and assigns all units that have correlations

hh. **Triadic Data Matrix**. From any sociomatrix *S*, this procedure lists all possible nondirectional triads in the matrix with all dyadic and triadic relational patterns

that characterize each of the triads. This characterization is based on a table derived from Burt (1990: 87). Each triad can be characterized by one of 36 types of ties (that include positive and/or negative signs). For example, consider the following matrix.

	1	2	3	4	5
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	1	1	1	1
5	0	0	0	1	1

The output of the triadic data matrix procedure takes all triads in this matrix, and defines: first, the relationship between all of the dyads making up a given triad, and then defines the "Burt" type of the triad by assigning it the type index in Burt's list. The output of the procedure for the matrix shown above is given below.

MI	Triad	A→B	A→C	B→A	B→C	C→A	C→B	ABC	ACB	BAC	BCA	CAB	CBA
1820	123	0	1	0	0	0	0	2	2	21	21	4	4
1820	124	0	0	0	0	0	1	21	21	4	4	2	2
1820	125	0	0	0	0	0	0	1	1	1	1	1	1
1820	134	1	0	0	0	0	1	31	22	5	5	22	31
1820	135	1	0	0	0	0	0	2	2	4	4	21	21
1820	145	0	0	0	1	0	1	11	11	6	6	6	6
1820	234	0	0	0	0	1	1	32	24	32	24	3	3
1820	235	0	0	0	0	0	0	1	1	1	1	1	1
1820	245	0	0	1	1	0	1	14	14	9	9	26	26
1820	345	0	0	1	1	0	1	14	14	9	9	26	26

Note that each triad has six permutations. The triadic columns list all six permutation in the order of elements in the triad for the appropriate row. Thus, for triad 245 the corresponding columns mean: ABC=245, ACB=254, BAC=425, BCA=452, CAB=524, CBA=542.

ii. **Role Equivalence Matrix**. Following Burt (1990: 86) role equivalence measures the equivalence, defined as a comparison between any two units in the network in terms of the convergence of their pattern of roles in the network as "a triad census, a pattern of relative frequencies with which kinds of triads describe the individual's orientation to others." Two individuals are said to be "role equivalent" to the extent that the distribution of their triadic census positions with others is identical. The current procedure uses standardized Euclidean distances between the role positions of any two units in the networks.

Bibliography

Ronald S. Burt 1990. Detecting Role Equivalence. *Social Networks*, 12(1): 83-97.

Zeev Maoz 2006a. The Effects of Strategic and Economic Interdependence on International Conflict: A Cross-Levels-of-Analysis Study. Mimeographed. University of California, Davis.

———— 2006b. Systemic Polarization, Interdependence, and International Conflict, 1816-2002. *Journal of Peace Research*, 43(4): 391-411.