The Stata Journal (2011)
11, Number 1, pp. 52–63

Visualization of social networks in Stata using multidimensional scaling

Rense Corten
Department of Sociology
Interuniversity Center for Social Science Theory and Methodology
Utrecht University
The Netherlands
r.corten@uu.nl

Abstract. I describe and illustrate the use of multidimensional scaling methods for visualizing social networks in Stata. The procedure is implemented in the `netplot` command. I discuss limitations of the approach and sketch possibilities for improvement.

Keywords: gr0048, netplot, mds, social network analysis, visualization, multidimensional scaling

1 Introduction

Social network analysis (SNA) is the study of patterns of interaction between social entities (Wasserman and Faust 1994; Scott 2000). In the past few decades, SNA has emerged as a major research paradigm in the social sciences (including economics) and has also attracted attention in other fields (Newman, Barabási, and Watts 2006). While dedicated software for SNA exists (for example, UCINET [Borgatti, Everett, and Freeman 1999] or Pajek [Batagelj and Mrvar 2009]), Stata currently lacks readily available facilities for SNA. In this article, I illustrate how methods for SNA can be developed in Stata, using network visualization as an example.

Visualization is one of the oldest methods in SNA and is still one of its most important and widely applied tools for uncovering patterns of relations (Freeman 2000). I describe a procedure for network visualization using Stata's built-in procedures for multidimensional scaling (MDS) and describe an implementation as a Stata command. While I believe that network visualization in itself can be highly useful, the example also illustrates how SNA problems can be handled in Stata more generally.

2 Methods

2.1 Some terminology

Network visualization is concerned with showing binary relations between entities. Adopting the terminology of graph theory, I refer to these entities as vertices. Relations between vertices may be considered directed if they can be understood as flowing from one vertex to another or may be considered nondirected if no such direction can be identified. I refer to directed relations as arcs and to nondirected relations as edges.

A typical representation of a network of relations is an adjacency matrix, as shown in figure 1 for a network of 10 vertices. In this matrix, every cell represents a relation from a vertex (row) to another vertex (column); for nondirected networks, this matrix is symmetrical. Vertices that have no edges or arcs are called isolates. The number of edges connected to a vertex is called the degree of the vertex. Lastly, the distance between two vertices is defined as the shortest path between them. If there is no path between two isolates, I define the distance between them as infinite.

	1	2	3	4	5	6	7	8	9	10
1	0	1	1	0	1	0	0	0	0	0
2	1	0	0	1	0	1	0	1	0	0
3	1	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	1	0	0	0
5	1	0	0	0	0	0	0	0	0	0
6	0	1	0	0	0	0	1	0	0	0
7	0	0	0	1	0	1	0	0	0	0
8	0	1	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	0	0	0

Figure 1. An adjacency matrix, $N = 10$

2.2 Data structure

One particular obstacle in analyzing network data in conventional statistics packages such as Stata is the specific structure of relational data. Whereas in conventional datasets one line in the data typically represents an individual entity, observations in relational datasets represent relations between entities.

I assume that data are available as a list of edges or arcs. That is, for a network of k relations, I have a $k \times 2$ data matrix in which every row represents an edge (if the network is nondirected) or an arc (if the network is directed) between two vertices in the cells. The use of edgelists and arclists is often a more economical way to store network data than is an adjacency matrix, especially for networks that are relatively sparse.

I extend the traditional edgelist and arclist formats by allowing the use of missing values. I use missing values to include isolates in the list (figure 2). In figure 1, vertex 10 is isolated; in figure 2, its vertex number appears in one column accompanied by a missing value in the other column. The order of appearance might be reversed, thus a network consisting of k edges and N vertices, of which h isolates, can be represented by a $(k + h) \times 2$ matrix.

	col 1	col 2
1	2	1
2	3	1
3	4	2
4	5	1
5	6	2
6	7	6
7	7	4
8	8	2
9	9	8
10	10	.

Figure 2. Edgelist based on the adjacency matrix in figure 1

2.3 Procedure

The main task in network visualization is to determine the positions of the vertices in a (typically two-dimensional) graphical layout. Obviously, the optimal placement of vertices depends on the purpose of the analysis; however, it is often desirable to centrally locate in the graphic those vertices that have a central position in the SNA and to represent a larger distance in the network by a larger distance in the two-dimensional graph. Various algorithms have been proposed toward this ideal. Among them, those by Kamada and Kawai (1989) and Fruchterman and Reingold (1991) are probably most widely used. Instead, I use MDS to compute coordinates for the vertices. This strategy has the advantage of being available in Stata by default. The use of MDS for network visualization has a long history in SNA and was first used in this way by Laumann and Guttman (1966).

Assuming that I have a relational dataset formatted as an edgelist, I propose visualizing the network by the following procedure:

1. Reshape the data into an adjacency matrix.
2. Compute the matrix of shortest paths (the distance matrix).
3. Arrange the vertices on a circle in a random order, and then compute their coordinates.
4. Using the coordinates circle layout obtained in the previous step as a source of starting positions, use the modern method to compute coordinates for the vertices by `mds`.
5. Draw the graphic by combining the `twoway` plot types `pcspike` or `pcarrow` with `scatter`.

In my implementation, steps 1–3 are performed in Mata. The calculation of the distance matrix (step 2) involves calculating higher powers of the adjacency matrix and can be rather time consuming for larger networks. More efficient procedures for obtaining distances in a network are feasible, but they are not implemented in my example.

I chose Stata's iterative modern `mds` method for step 4 because it allows for the specification of starting positions and appears to provide better results in tests. In particular, the modern method performs better than the classic method with regard to the placement of vertices that have identical distances to all other vertices (for example, vertices on the periphery of a "star"). Experimentation furthermore suggests that starting with a circular layout provides the best results.[1]

3 Implementation: The netplot command

3.1 Syntax

`netplot` *var1 var2* [*if*] [*in*] [, `type(mds|circle)` `label` `arrows`
  `iterations(#)`]

The `netplot` command produces a graphical representation of a network stored as an extended edgelist or arclist in *var1* and *var2*.

3.2 Options

`type(mds|circle)` specifies the type of layout. Valid values are `mds` or `circle`.

  `mds` calculates positions of vertices using MDS. This is the default if `type()` is not specified.

  `circle` arranges vertices on a circle.

`label` specifies that vertices be labeled using their identifiers in *var1* and *var2*.

`arrows` specifies that arrows rather than lines be drawn between vertices. Arrows run from the vertex in *var1* to the vertex in *var2*. This option is useful for arclists that represent directed relations.

`iterations(#)` specifies the maximum number of iterations in the MDS procedure. The default is `iterations(1000)`.

4 Examples

To illustrate the process outlined above, I use the well-known Padgett's Florentine Families dataset, which contains information on relations among 16 families in fifteenth-

1. Internally, my program issues the command `mdsmat` *distance_matrix*`, noplot method(modern) initialize(from(`*circle_matrix*`)) iterate(#)`.

century Florence, Italy (Padgett and Ansell 1993). The part of the data I use represents marital relations between the families. These relations are by nature nondirected. The data are described below:

```
. describe
Contains data from Padgett_marital02_undir.dta
  obs:            21                          Padgett marital data with
                                                undirected ties
 vars:             2                          22 Jan 2010 17:37
 size:           588 (99.9% of memory free)   (_dta has notes)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
from            str12  %12s                   family 1 name
to              str12  %12s                   family 2 name
-------------------------------------------------------------------------------
Sorted by:  from  to
. list, sepby(from)

     +---------------------------+
     |        from            to |
     |---------------------------|
  1. |                     Pucci |
     |---------------------------|
  2. |     Albizzi      Guadagni |
  3. |     Albizzi        Medici |
     |---------------------------|
  4. |   Barbadori        Medici |
     |---------------------------|
  5. |    Bischeri      Guadagni |
  6. |    Bischeri       Peruzzi |
  7. |    Bischeri       Strozzi |
     |---------------------------|
  8. |  Castellani     Barbadori |
  9. |  Castellani       Strozzi |
     |---------------------------|
 10. |      Ginori       Albizzi |
     |---------------------------|
 11. |    Guadagni   Lamberteschi |
     |---------------------------|
 12. |      Medici    Acciaiuoli |
 13. |      Medici      Salviati |
 14. |      Medici    Tornabuoni |
     |---------------------------|
 15. |       Pazzi      Salviati |
     |---------------------------|
 16. |     Peruzzi    Castellani |
 17. |     Peruzzi       Strozzi |
     |---------------------------|
 18. |     Ridolfi        Medici |
 19. |     Ridolfi    Tornabuoni |
     |---------------------------|
 20. |     Strozzi       Ridolfi |
     |---------------------------|
 21. |  Tornabuoni      Guadagni |
     +---------------------------+
```

The data are in this case formatted as strings that simply use the family names as identifiers for the vertices of the network.

The first example (figure 3) shows the most basic usage of `netplot`. It uses the `netplot from to` command to produce a network plot of the data resulting from MDS.

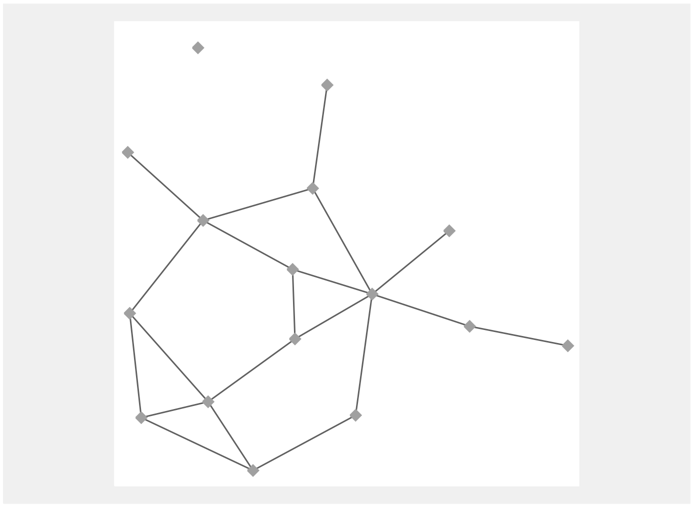

Figure 3. Marital relations among Florentine families, with vertex placement by MDS

In many analyses, it is useful to be able to identify specific vertices in the network. Identification is facilitated by adding labels to the plot using the `label` option (figure 4).[2] I can now observe that this network has a cohesive core formed by the Medici, Ridolfi, and Tornabuoni families, and that the isolated vertex is the Pucci family.

2. The placement of labels outside the plot region is part of the default behavior of `twoway scatter`, which is used by `netplot`. This can be easily adjusted afterward.

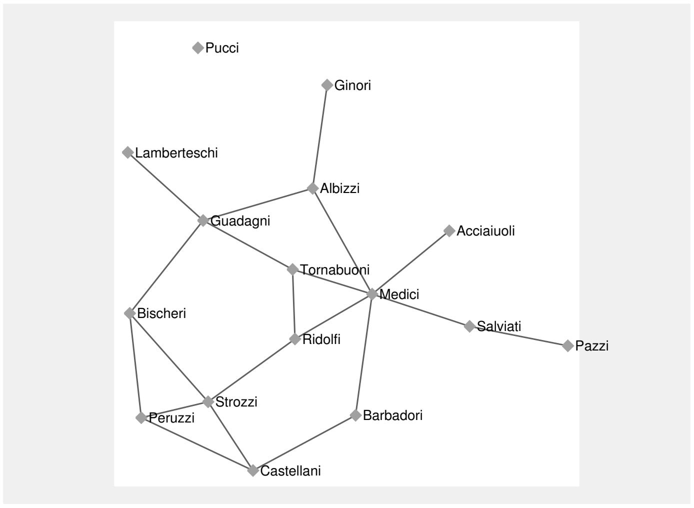

Figure 4. Marital relations among Florentine families, with vertex placement by MDS and labels added

Sometimes it is not necessary to have the relatively complicated plot as produced by MDS. Then a simple view on the data can be produced by the `circle` option (figure 5).

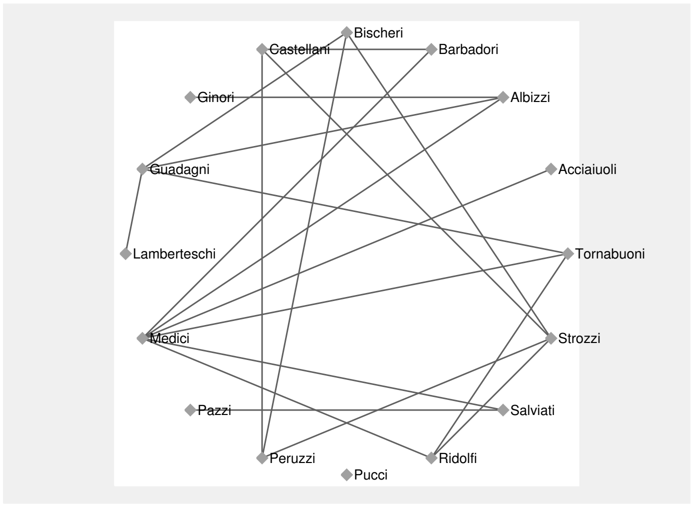

Figure 5. Marital relations among Florentine families, with circular vertex placement and labels

For my final example with these data, I assume that the data are directed. That is, I assume that each line in the data represents a directed relation from one vertex to another vertex. Imagine, for instance, that the data now represent whether a family has ever sold goods to another family. Such situations can be visualized using the `arrows` option, which draws arrows instead of lines between vertices (figure 6). The graph in this example was slightly adjusted afterward by using the Graph Editor to reduce the sizes of the markers and to make the arrowheads better visible.

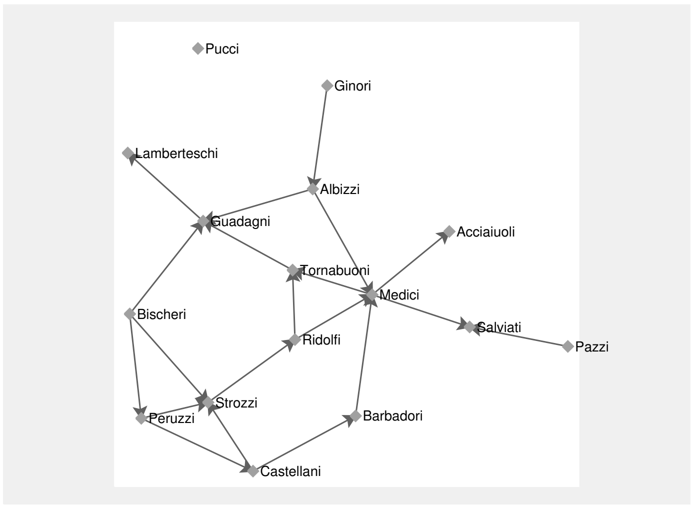

Figure 6. Marital relations among Florentine families, shown as directed relations with vertex placement by MDS and labels

As a final example, I draw a plot of a somewhat larger network of 100 vertices. The data for this example were simulated using the "preferential attachment" algorithm proposed by Barabási and Albert (1999) to construct the network shown in figure 7.[3]

This example highlights two limitations of `netplot`. First, as figure 7 shows, vertex placement can be suboptimal: several vertices in the figure are placed too close together, while others are placed too far from neighboring vertices, which leads to crossings of edges. The reason is that in this particular treelike network structure, there are many vertices that have the exact same distance to all other vertices, which makes placement by MDS difficult. Second (not visible in the figure), the procedure becomes considerably more time consuming with this number of vertices. I discuss this issue in more detail in the next section.

3. The actual simulation was conducted in Mata. The function in which the Albert–Barabási algorithm is implemented is part of a larger library of functions for network analysis under development by the author.

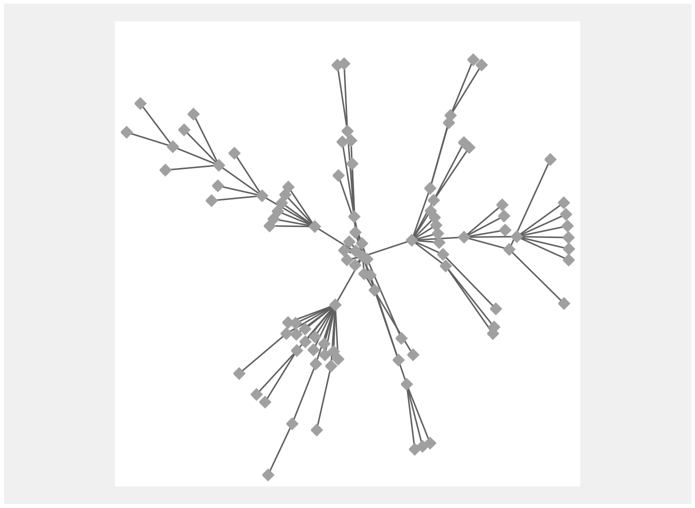

Figure 7. Marital relations among Florentine families, shown as directed relations with vertex placement by MDS

5 Performance

To get a rough idea of the performance of `netplot` in terms of computation time, I conduct two simulated tests. First, I draw plots of networks of increasing network size, keeping network density constant at 0.5. This leads to an exponentially increasing number of edges in the network. To draw the plots, I use `netplot` without any options. The input networks are randomly generated Erdös–Rényi graphs (Erdös and Rényi 1959).[4]

For the second test, I again draw plots with increasing network size but keep average degree constant rather than density. This implies a linear increase in the number of edges in the network. I use an average degree of 3.

In both tests, I look at networks with sizes ranging from 5 to 100 in increments of 5. In addition, I simulate networks of 500 nodes and networks of 1,000 nodes. I keep track of the average time needed to draw a graph over 10 iterations per network size.

The results are shown in figure 8 for the networks of up to 100 nodes. The figure indicates that average time increases quadratically with network size, although time increases more strongly with constant density than with constant degree. For networks of 500 nodes, computation times average 1,545 seconds for networks with a density of 0.5 and 1,182 seconds for networks with an average degree of 3. For networks of 1,000 nodes, the average times are 6,936 seconds and 7,769 seconds, respectively.

4. The tests were run in Stata/SE 11 on a PC with a 2.66-GHz dual-core processor and 1 GB of memory and running the Microsoft Windows XP 32-bit operating system.

Obviously, computation time becomes a major obstacle when using `netplot` on larger networks. In addition, convergence and computational problems of the MDS procedure become more frequent in larger networks. Closer analysis (not reported) of the running time of the different components of the command reveals that the computation of coordinates using MDS after computation of distances is the most time-consuming step in the procedure.

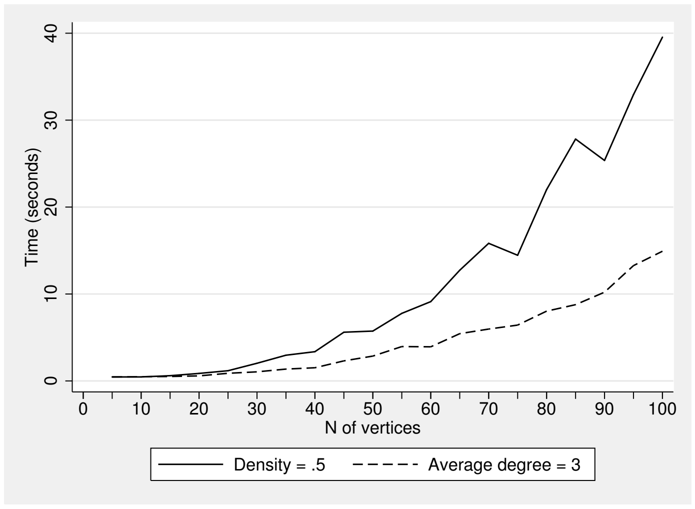

Figure 8. Average computation time by network size

6 Discussion

In this article, I have demonstrated how to use built-in techniques for MDA and graphics to visualize network data in Stata. This method often produces useful results, although not always for all networks. A major drawback is the long computation time needed to compute vertex coordinates on larger networks. As a workaround for this problem, the number of iterations may be limited by using the `iterations()` option.

Visual results could likely be improved by using vertex placement algorithms different from MDS. Good candidates are the often-used "spring embedding" algorithms by Kamada and Kawai (1989) and Fruchterman and Reingold (1991). Given the command architecture of `netplot`, these methods could be added relatively easily, and implementing them would be an obvious target for future development.

A second reason to focus on placement algorithms different from MDS in future development is that the MDS procedure appears to be the major cause of the long computation time needed for large networks. At this moment, however, it is not clear how, for example, the Kamada–Kawai and Fruchterman–Reingold algorithms compare with MDS in terms of computation time.

Another approach to improving efficiency is to use more-efficient methods for computing distances in the network. The simple approach currently implemented, which is based on repeated matrix squaring, computes some quite unneeded information in the process. More-efficient algorithms for computing shortest paths exist (see Cormen et al. [2001]) and might be implemented in the future.

The introduction of Mata with Stata 9 has made matrix programming more effective and more accessible for the average user. This opens up further possibilities for the development of SNA methods in Stata. The fact that Mata can be used interactively makes it easier to use the alternative data structures representing networks common in SNA. The quickly growing interest in social networks in and outside the social sciences certainly justifies the further development of SNA methods for Stata.

7 References

Barabási, A.-L., and R. Albert. 1999. Emergence of scaling in random networks. *Science* 286: 509–512.

Batagelj, V., and A. Mrvar. 2009. *Pajek*. Program for Large Network Analysis. Ljubljana, Slovenia. http://vlado.fmf.uni-lj.si/pub/networks/pajek/.

Borgatti, S. P., M. G. Everett, and L. C. Freeman. 1999. *UCINET*. Program for Social Network Analysis. Lexington, KY. http://www.analytictech.com/ucinet/.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein. 2001. *Introduction to Algorithms*. 2nd ed. Cambridge, MA: MIT Press.

Erdös, P., and A. Rényi. 1959. On random graphs, I. *Publicationes Mathematicae (Debrecen)* 6: 290–297.

Freeman, L. C. 2000. Visualizing social networks. *Journal of Social Structure* 1. http://www.cmu.edu/joss/content/articles/volume1/Freeman.html.

Fruchterman, T. M. J., and E. M. Reingold. 1991. Graph drawing by force-directed placement. *Software—Practice and Experience* 21: 1129–1164.

Kamada, T., and S. Kawai. 1989. An algorithm for drawing general undirected graphs. *Information Processing Letters* 31: 7–15.

Laumann, E. O., and L. Guttman. 1966. The relative associational contiguity of occupations in an urban setting. *American Sociological Review* 31: 169–178.

Newman, M., A.-L. Barabási, and D. Watts, ed. 2006. *The Structure and Dynamics of Networks*. Princeton, NJ: Princeton University Press.

Padgett, J. F., and C. K. Ansell. 1993. Robust action and the rise of the Medici, 1400–1434. *American Journal of Sociology* 98: 1259–1319.

Scott, J. 2000. *Social Network Analysis: A Handbook*. 2nd ed. London: Sage.

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

About the author

Rense Corten is a postdoctoral researcher at the Department of Sociology and the Interuniversity Center for Social Science Theory and Methodology, Utrecht University.