UCINET 6 for Windows

Software for Social Network Analysis

中文翻译版

By MR

www.linkding.cn

由于毕业论文需要，翻译了一下这个文档，水平有限，很多专业词汇只能是字面翻译了，不过至少应该可以对软件有个大概的了解了，另外，省略了第一章和第 5 章没必要的东西。查了一下对应的这个课程貌似是博士生课程，于是，我释然了，翻译的不好也就那样了。

对于一些令人费解的地方还是请各位参照一下英文原版。

MR

2012 年 1 月 6 日

0.1 Notational Conventions

Ucinet 是菜单驱动（menu-driver）的 windows 驱动程序，也即你可以通过选择菜单来选择需要做什么。菜单可能被隐藏（nested），因此点选一个菜单项可能会呼出有额外选项的子菜单。子菜单还可能有下一级子菜单。为了设定好选项，你可能必须要点选许多菜单项。为了表示选一个选项你需要的操作，我们使用了角括号。比如说，要运行 hierarchical clustering 程序，你必须先启动 ucinet，然后单击 tools，在下拉菜单中选 cluster，再从子菜单中点选 **Hierarchical**.，我们将会这么表示这个操作：

Tools>Cluster>Hierarchical

0.3 Programming Considerations

编写 Ucinet6 的宗旨是速度而不是舒适，在编写 ucinet 的过程中，我们必须在消耗许多内存的快速的算法和消耗较少资源的较慢的算法之间做出选择。在之前的版本中我们试图在这两者之前寻求平衡。在这个版本中，我们总是选择前者--速度为重，一个原因是因为处理大量数据时，数据是很重要的：一个能处理很多数据但是却需要好多天才能执行完毕的程序有什么好处呢？另一个原因是软硬件的持续进步拓展了程序可以使用的内存，所以为了编写节省内存资源的程序似乎是一种浪费。

菜单系统的一个需要是把程序功能和子功能有条理合理地组织起来，当然，这被验证为是不可能的。只用一点努力一个人便可以发现多种互相矛盾、难以区分好坏的功能组织方式。这些分类组织的方法没有完美的。每一种都有特定的缺点。我们采用的方法如下：在 network 菜单下我们安排了网络相关的技术：如果导入与网络无关的数据将会导致出错的技术。相对比的是，在 tools 菜单下我们安排了网络分析中经常被用到，并且也同时经常在与网络无关的分析中被用到的技术。Multidimensional scaling 和 cluster analysis 就是这方面的例子。当然，总有一些技术难以分类或者容易分类出错。

0.4 Content

Ucinet6.0 是 UCINET IV 的升级版，所以对 UCINET IV 熟悉的用户会很容易地习惯新环境。我们拓展了 ucinet 的潜力，并重组了一些功能。

也许我们曾经面对的最基本的设计考虑是程序应该具有何种性能。因为 Freeman 的第一个版本发布了，ucinet 引入了许多网络技术，技术的多不但表现在他们的功能上，也表现在他们的来源上。在 UCINET 6.0 中，我们继承了传统，看上去我们更像各种作品的编辑者和出版者而不是一种单独的流行作品的作者。

这种做法的一个问题是是不同的技术使用不同的数据格式。比如， graph-theoretic（图论）技术将数据描述为抽象的点和线的集合。代数（algebraic）学将数据描述为假设前提和关系，统计学以变量、向量、矩阵来描述。有时数学支系交叉，同种计算会出现在不同支系中。比如，图像的自重构，是图像本身的 1-1 映射，然而矩阵重构是单模矩阵中的行和列的重排序。同样地，图像的逆，是所有方向的逆，然而在矩阵中的逆，则是行和列的对换。不幸的是，在词典中也存在错误的同源词。比如，一种关系可能被表示为矩阵，但在离散代数学中，逆是矩阵的变换，而不是矩阵自乘所得的值。

在 UCINET 6.0 中，所有的数据都用矩阵描述。有些提示和输出结果反应了与一项技术关系最密切的语言，掌握以矩阵为中心的数据观对于用户是非常重要的。所有的 UCINET 数据最终都被存储和描述为矩阵集合。了解图像、网络、关系、超图以及所有其他网络分析的实体是如何用矩阵表示的对于高效地使用系统是非常必要的。

1.1 Hardware

UCINET 6.0 要求电脑运行 Windows 95 ， Windows 98, NT 或者其他兼容的操作系统，要求 2M（貌似译者拿到的版本不止 2M）的硬盘空间和 16M 内存。

2.2 Forms

假设你从菜单里选择一个程序，除了极少数意外，你将会看到参数表格。该格式是一个以上的需要填的空格，比如，你要是从 File 菜单中点选了 **networks>Subgroups>Cliques，你将会见到的就是以下的问题：**

Input dataset:	Camp92.##h
Minimum size:	3
Analyze pattern of overlaps?:	YES
Diagram Type	Tree Diagram
(Output) Clique indicator matrix:	CliqueSets
(Output) Co-membership matrix:	CliqueOverlap
(Output) Partition indicator matrix:	CliquePart

左栏是问题，右栏包括默认的答案。除了你要分析的数据名字一栏，通常情况下你可以保留默认值，其他栏可以通过直接输入或者双击然后在出现的对话框中选择来改变默认值。

当所有的表格都填好后，你可以单击确认键激活工作。

下次你在菜单中点选 Clique，你会发现默认输入数据是上次你运行 Clique 的时候用过的数据，除非 **Smartdefaultnames 被点选上了。在这种情况下，程序根据它所认为你所需要的填入默认数据。假如这是你要编辑的文件，点击 OK.如果不是，直接输入新文件的名字或者双击选择。**

无论什么时候你要输入文件名，然后右栏的有着三点的按键（浏览键）可以激活选择文件的标准窗口。

顺便提一句，无论什么时候你要输入参与者的名单，你都要通过编号（比如矩阵中的行或列）而不是通过姓名或者其他标签 ，用空格或者逗号分开编号，在这个过程中，UCINET 引导你点击标签，自动输入正确的操作员或者矩阵编号，又一个 L 标志的按键允许你选择列表，另外，你可以想如下所列指示编号组：

3 to 10

3-10

first 5

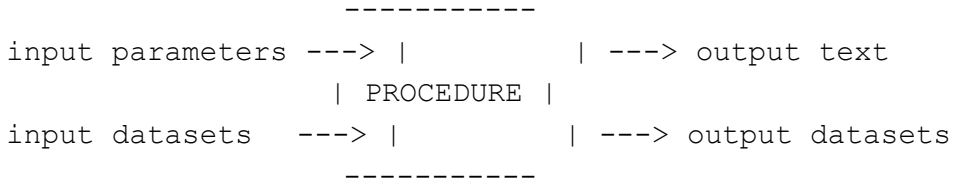
last 30

前两行都表示 id 号 3, 4, 5, 6, 7, 8, 9, 10。最后一行只有在总的操作员数目确定时才有效，这些协定可以如下混用：

List of actors to DROP: first 4, 18 12,5 19 to 25, last 2

2.3 Running An Analysis

像如下对话框所示，典型的 UCINET 6.0 程序有两种输入数据以及两种输出数据



一种输入包含了 UCINET 6.0 数据组，大部分程序只用一种输入格式，但是这种格式可能包括多个矩阵，一般代表相同参与者的不同的社交关系，其他程序（比如 **Join** 和 **QAP Regression 程序**）使用多种输入格式，每种都包括多个矩阵。

另外一种输入由一系列描述数据或者改变程序运行方式的参数组成。比如，就像你先

前看到的，**Clique 程序的参数是：输入数据组的名字，需要报告的 Clique 的最小尺寸，以及各种输出文件的名字。大部分参数有默认值，用户可以接受或者更改。**
从菜单选择一个程序以后，参数会立即以表格的形式展现。程序第一次在会话中被调用时，表格会包含出厂设置的默认参数。如果你修改了参数值，这些改变会在以后的运行中沿用，直到你下次更改这个参数值或者退出 ucinet。这条规则的唯一例外是一些默认设置基于其他默认设置，在这种情况下改变特定参数会使得程序修改其他默认设置，即使你之前设置过他们。
程序的输出文件名一般都有一系列的默认参数，输出文件是其他 UCINET 程序可以查看的 UCINET 6.0 数据，因此可以作为其他分析的输入数据。比如，CLIQUE 程序的一个输出是由参与者组成的矩阵，给出了参与者 i 和 j 作为成员的最大团（clique）的数目。矩阵适合作为 MIDS 程序的输入来获得重叠模式的空间模型.,默认情况下，该数据组的名字是 **CliqueOverlap**.
除了输出数据组，大部分 ucinet 程序同时产生一个文本报告，一般保存为 txt 文件，即日志文件（**Log File）（一般存储在**\Windows\System 目录下**），并自动显示在屏幕上。**
让我们运行一个 clique 分析的例子。，第一步是在工具栏点选 **Network>Subgroups>Cliques** ,然后会弹出一个表格如下：

```
      Input dataset:
      Minimum size:                          3
      Analyze pattern of overlaps?:      YES
      Diagram Type                           Tree Diagram
      (Output) Clique indicator matrix: CliquesSets
      (Output) Co-membership matrix:         CliquesOver
(Output) Partition indicator matrix:    CliquesPart
```

第一行需要包含 UCINET 6.0 数据的数据组的名称。如果数据组名为 TARO，保存在 C 盘的\uci3 文件夹下，你可以如此填入 **c:\uci3\TARO**.，你也可以通过单击右侧按键选择文件。
第二行询问要报告的团（clique）的最小尺寸默认值是 3，你可以直接对其进行修改。
第三行询问是否需要计算和分析由参与者组成的所有矩阵中的每一对参与者所共属的团的数量。这个矩阵提交给分级群聚（submitted to hierarchical clustering），默认值是 YES，为了在处理大量数据时节约运行时间你可以改选该选项。
第四行询问你想查看何种群集图，默认值是树形图，另一个是树枝型图，与树形图相类似。
从第五行到第七行询问保存关键输出的数据组的名称，这些数据组可以再以后的分析中作为输入数据使用，这三个空都有默认值。
当你完成输入修改参数以后，点击 OK 开始分析，由于 TARO 是标准的 ucinet 数据组所以你可以运行分析，结果将会是出现在屏幕上的树状图，如果你现在点击 OK 那么下面的日志文件会出现。

```
CLIQUES
------------------------------------------------------------
---------------

Minimum Set Size:  3
```

Input dataset: C:\Uci3\TARO

10 cliques found.

1: 2 3 17
2: 1 2 17
3: 17 18 22
4: 4 5 6
5: 4 6 7
6: 5 20 21
7: 8 9 10
8: 11 20 21
9: 12 13 14
10: 12 14 15

Group Co-Membership Matrix

								1	1	1	1	1	1	1	1	1	1	2	2	2			
	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	
	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
2	1	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	
3	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
4	0	0	0	2	1	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	0	0	0	1	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	
6	0	0	0	2	1	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	
12	0	0	0	0	0	0	0	0	0	0	0	2	1	2	1	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	
14	0	0	0	0	0	0	0	0	0	0	0	2	1	2	1	0	0	0	0	0	0	0	
15	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
17	1	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	3	1	0	0	0	1	
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
20	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2	2	0	
21	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2	2	0	
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	

```
SINGLE-LINK HIERARCHICAL CLUSTERING

          1 1 1 1 1 1 1        1 2 2      1 1 2
Level   8 9 0 3 2 4 5 6 9 5 6 4 7 1 1 0 1 3 2 7 8 2
-----   - - - - - - - - - - - - - - - - - - - - - -
    2   . . . . XXX . . . . XXX . . XXX . . XXX . .
    1   XXXXX XXXXXXX . . XXXXXXXXXXXXX XXXXXXXXXXX
    0   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX


Group indicator matrix saved as dataset CliquesSets
Clique co-membership matrix saved as dataset CliquesOver
Clique co-membership partition-by-actor indicator matrix saved as
dataset CliquesPart

Date and time: 15 Jan 99 13:38:43
Running time:  00:00:01
```

CLIQUE 程序的输出表格是大部分分析程序的典型。它以运行参数报告开头，这使得以后理解和再生输出结果变得容易。输出的下一段是关于最大团的数目的报告，并有每个最大团的列表，每行一个。每行的第一个数用以辨别最大团。其他数字表示了属于最大团的参与者，如果数据包括了参与者标签（名字），你就应该能看到名字。

在 cliques 列表后面是参与者参与者最大团合作矩阵，给出了每个参与者所共同拥有的最大团。这是下一个表格的基础，给出了合作矩阵的单链分级群聚。如果你喜欢不同的聚类算法，你可以提交给合作者矩阵，这会自动保存到你选择的聚类程序。

输出数据的最后部分声明了程序输出了什么东西。主要的输出文件是参与者群集二进制矩阵指示了参与者属于哪个团。这矩阵将来可以被用来分析，或者用来从打网络中释放出子图，使用 **Data>Extract** 程序。比如，假如你只是想从中提出某个团的关系网，只要告诉 **Extract** 程序原始数据组的名字是什么，参与者团矩阵的名字（默认是 CliquesSets），以及你需要的团（#4 代表第四行），这就够了。

2.4 The Log File

如上所述的输出文件一般被写入 \windows\system 文件夹下的文本文件中，默认名为 **Log File Number x**，**x** 是一个整数。当你运行分析时，程序输出到日志文件中并且展示文件中的数字到工具栏的容器中。文件可以使用工具栏上的按钮和选项像往常一样编辑，保存，打印。当你运行另外一个程序时会创建一个带有新的比原来数字大的日志文件，你可以不管日志文件或者把他关掉。也可使用 **File>Edit Previous Log File** 打开关闭的以前的日志文件。默认情况下 **ucinet** 可以保持 **50** 个历史文件，这可以通过 **Options>Number of Log Files** 修改。如果你想在切换对话时保存日志文件，可以在输出窗口中点选保存按钮或者选择 **File** 菜单下的 **Save**。必须要注意的是没保存的所有日志文件会在退出 **ucinet** 时被删除。

2.5 Datasets

在 ucinet6 数据组中有三件重要的事需要记住。第一，数据是矩阵的集合。不管你把你的数据理解为图像，关系、超图还是其他的，在 ucinet6 中，你的数据就是矩阵集合。这并不是意味着 ucinet 不能读取不是矩阵格式的数据，而是说在程序系统中，他们都是被看作为矩阵的。

网络分析人员一般把他们的数据理解为图，图就是一系列的节点和一系列连接这些点的线。图的信息可以用邻接矩阵表示，在邻接矩阵中给定元素 X（i,j）的值为 1 代表节点 i 和 j 是连接着的，0 代表这两者不是连接着的。

以下是用矩阵代表网络的一个示例：

	A	B	C	D	E
A	0	1	0	1	1
B	1	0	1	0	0
C	0	1	0	0	1
D	1	0	0	0	0
E	1	0	1	0	0

在这个网络中，参与者 A 和参与者 B，D 和 E 有联系，但和 C 以及他自己没有联系。参与者 B 和参与者 A,C 有联系，参与者 C 和参与者 B,E 有联系，参与者 D 只和 A 有联系，参与者 E 和参与者 A,C 有联系。

有向图是一系列点和连接这些点的圆弧（或者是箭头、有头尾的线）的集合。他们被用来表示节点间的不必是相互的关系，比如“爱上”或者“是……的老板”。有向图中的信息可以被记录为单模邻接矩阵（并不必须是对称的），在其中，如果 i 被连接到 j 则 X(i,j)=1，否则 X(i,j)=0.注意 X(i,j)可以等于 X(j,i)，但这不是必须得。

赋值图可以用一个长方形单模矩阵表示，在其中 X(i,j)给出了从 i 到 j 的连接的值，可以代表关系的强度，道路的长度，状态转换的可能性，联系的频繁性等。

超图是一系列节点的子集的集合。子集在概念上就像可能有两个末端的边界/连接。在超图可以用 2 模矩阵表示，在其中，假如 i 在子网 j 中，则 Y(i,j)=1，否则 Y(i,j)=0.。

Ucinet 中包含的矩阵可以有任何形状或者尺寸，而且并非都代表网络。比如，一下三个数字集合就都是矩阵：

Matrix #1:

1 3 2 5
1 5 7 2
1 2 7 2
2 4 5 2
9 6 5 1

Matrix #2:

1 3 8 9 2 3 5 1.7

Matrix #3:

3.1415
注意第二个矩阵有 8 列 1 行。第 3 个矩阵是 1 行 1 列。古怪的形状并不是问题。重要的是每一行包括了同样数目的列数，反之亦然。
Ucinet 数据表的一个重要特征是他们可能包括了一个以上的矩阵，虽然每个矩阵的行列必然反映同一个物体。这使得你可以吧与一系列人有关的所有网络数据放在同一个文件中。比如，你可能有一系列的家庭作为节点，并如此度量家庭间的关系：“嫁给了……中的一人”和“与……中的一人做生意”。这对于应用有一个或多个社会关系作为输入的网络技术非常有用，比如大部分的位置方法（positional methods），(比如 CONCOR,REGE).甚至把多种关系应用在没有多种关系的技术里面也会很有用，比如中心法（centrality measures）。在 ucinet 中，无论什么时候有可能，一个引入使用多关系无效的技术的程序会基于每种关系挨个运行。比如在包含数百个随即网络的数据组中运行中心法时，程序会为文件中的每个网络计算和保存方法。结果可用于统计分析。
使用多矩阵数据组的另一种方法被引入到 **Tools>Matrix Algebra 程序中。在此，程序把多矩阵数据组视为单个由行列级组成的 3 路矩阵，并允许用户同时在三个维度运行操作。**
第二件重要的事情是理解 ucinet 数据组并不是文本文件。因此你不能使用文字处理软件来修改。只有 ucinet（以及其他来自于 Analytic Technologies 的软件）可以读写他们。这会有所不便，但是这可以提升性能。当然，ucinet 也提供了把文本文件转换为 ucinet 数据组的方法（参见 Import 命令），反之亦然（参见 Export）。在这个方面 ucinet 和 SYSTAT，SAS,SPSS,GRADAP 以及其他致命的分析软件想象。
第三件重要的事是单个 ucinet 数据组实际上由两个文件组成。一个（后缀名为.##D）包括了实际数据，另一个（后缀名.##H）包含了数据的信息。当参考 ucinet 数据组的时候，你只能参考##H 文件（或者同时只是忽略后缀名）。你应该正确使用文件名：像“sampson”或“sampson.##h”。文件名中可以包括空格，可以数字符号打头。然而，有时候你需要使用括号扩住有空格的文件名。UCINET IV 数据组在 ucinet6.0 中兼容并且无需转换。

2.6 The Default Directory

在任何情况下，假如你输入要分析的文件名，ucinet 假设文件在当前目录，除非你输入完整目录，要改变默认目录，在 ucinet 窗口的右侧的按钮处更改，这将会调出一个属性文件对话框允许你选择你要读写的数据文件，确认一定要双击目录。

2.6 Program Organization

Ucinet 是从许多源头来的上百个程序和技术的一个广泛集合。我们把这无数的能力分为基本的六类。对应于主窗口的六个项目：文件（**File**），数据（**Data**），转换（**Transform**），工具（**Tools**），网络（**Network**）以及选项（**Options**）。以下我们将介绍每种分类。

2.7 File Submenu

文件（File）子菜单包括了关于文件和文件夹的常用命令以及打印和退出 ucinet 的命令。你可以改变默认文件夹，即在你不提供全路径的情况下 ucinet 寻找文件的文件夹。你也可以建立自己的文件夹保存文件。子菜单包含了重命名，复制，粘贴等命令，因为 ucinet 数据组文件实际由两个文件组成，意味着删除文件之类的操作只需要操作一次。同时也存在编辑和查看文本文件以及日志文件的命令。

2.8 Data Submenu

数据子菜单包括了处理 ucinet 数据组的命令。他被分为六个基本块。第一块只包含了直接输入和编辑 ucinet 文件的命令。第二块包含了建立或者引入 ucinet 数据的命令。

包括了 RANDOM 来建立基于不同分布的随机数据、Import 用来转换各种 ASCII 文件、Export 用来转换 ucinet 文件为其他文件，以及其他三个命令（**Attribute, Affilations 和 CSS**）用来转换特定文件为标准的网络数据，Display 用来以矩阵形式展示 ucinet 数据组。Describe 表述了 ucinet 数据的变化信息（比如行、列、标签、标题的序号），并允许用户输入和编辑标签。第四部分包括了 Export 用来部分数据组形成一个新的数据组，Join 用来把两个以上数据组连接起来，Unpack 用来把多关系的数据释放为单独矩阵。下一部分用来对 ucinet 数据进行非数值的转换和操作：Sort 用来将行列按给定值归类，Permute 用来把行列重排为特定的顺序。Transpose 用来转换行和列，最后一部分可以再许多方面重组数据。

2.9 Transform Submenu

Transform 子菜单分为四部分。第一部分联合行列的命令。Block 创建了可以被转换为块模型块密度。Collapse 给出了一个相似但是更大众化的功能，允许用户单独对待行列并且决定如何计算联合值。第二部分包括了在不转换维度的情况下操作所有单元的的命令，包括 symmetrizing（对称）, dichotomizing（二分）, recoding（重编码）, reversing（反转） and the **Diagonal（斜对）命令，Diagonal（斜对）允许用户改变矩阵主对角线的值并保存当前值到文件中。第三部分包括了 Rank 和 Normalize,分别用来排序和使用多种技术使整个矩阵或者行或列正规化。最后一部分的命令一般会产生新的节点，连接和关系。包括：Linegraph（线形图）构造一个由节点和原始连接组成的线形图；Incidence（关联）把一个邻接矩阵转换为一个长方形点线指示矩阵（a rectangular node-by-line indicator matrix）；Multigraph（多图），将一个赋值图装换为多个二值矩阵，每个矩阵代表图中的一个值。Bipartite（二分）将二分图的关联矩阵转换邻接矩阵。Multiplex（多路）通过一个多重关系图构建一个多路图。Semigroup（半群）从矩阵中输出一个半群。**

2.10 Tools Submenu

工具菜单包括了公制单位和非公制单位的多维度量（metric and non-metric multidimensional scaling**），群分析（** cluster analysis**），相似性分析（**correspondence analysis**），单值分解（**singular value decomposition**），因素分析（** factor analysis**）以及相似性和相异性分析方法（**measures of similarity and dissimilarity**）。Statistics（统计）部分包括了 P1 模型的标准描述方法以及一系列变换测试方法。这包括了 Matrix QAP（复原和关联），Autocorrelation（自动关联）以及 Regression, Anova 和 T-Tests，这一块也包含了**将 ucinet 数据转换为 dendrograms、scatterplots 和树形图的方法。

2.11 Network Submenu

在网络菜单下的 **Cohesion（内聚）下有度量节点间的距离值、节点间的最大流量、可达到性、节点间路径容量等的命令。 Regions（域）下有找到 Components（组件，强的以及弱的），Bi-components 和 K-Cores 的方法。Subgroups（子群）下有在网络中寻找各种子群的方法，包括** cliques, n-cliques, n-clans, k-plexes，ambda sets a 和 factions。**Ego Networks 可以用来计算网络中每个参与者的基于自我的度量。在 Centrality 中给出了计算节点中心的多种方法，包括了** degree, closeness, betweenness, flow betweenness, information centrality, eigenvector centrality, power 以及 Katz 和 Hubbell 的所有方法。**Core/Periphery（核心/周围）下即探测核心和周边并在结构中定位每个参与者的位置，Roles and Positions（角色和位置）下可以根据结构相似性辨别和聚拢参与者：**structural equivalence, automorphic equivalence, 和 regular equivalence. 计算每种 equivalence（相等）都有多种算法。**Properties 用来计算整个网络密度和转移传递性。**

2.12 Options Submenu

选项子菜单可以修改 ucinet6.0 的默认系统参数设置。

Importing Data

在使用 ucinet 做任何分析之前，必须先创建 ucinet 数据组。典型地，网络数据从问卷调查中或者书籍和采访中的数据表中来。在这两种情况下材料载体是纸张，你需要将对应数据输入到计算机文件中。最好最通用的方法是使用文件编辑器或者你喜欢的文字处理软件把数据保存为 ASCII 文件。，在本章中将会介绍输入数据的几种格式。一旦数据保存在了计算机磁盘中，你可以使用 **Data>Import/Export>Import 命令来将这些数据装换为 ucinet6.0 的数据组。使用过 SYSTAT 等统计软件的用户对这一部肯定不陌生，。在 SYSTAT 中，你可以使用 DATA 命令来读取 ASCII 文件、创建 SYSTAT 系统文件，读取 SPSS 和 SAS 程序文件，虽然他们不需要像 SYSTAT 和 UCINET6.0 一样需要永久的系统文件。**

Import 可以处理许多格式的 ASCII 数据。最普遍 RAW,DL,EXCEL 和 UCINET 3.0（ucinet6.0 和 UCINET IV 使用相同的数据格式所以不需要导入）。RAW 文件只包含数字，比如一个由问题答案数字编码组成的变量矩阵。DL 文件包括了和 RAW 一样的文件，另外还包括了数据的信息，比如行列的数目，变量的名字，研究的名字，一起其他。Ucinet3 文件与 DL 文件想象，但是在数据信息方面有更多的限制。EXCEL 则是标准的 EXCEL 数据表文件，这些文件格式将会在下面的部分详细讨论。

如果你通过抓取输入数据，我们强烈建议你是用 DL 格式（你可以在任何时候通过 Export 把数据输出为其他格式。）这种格式在接受数据方面是最可靠的。

无论你导入的文件格式是怎样的，输出总是一样的：即一个可以被应用于任何数值程序输入的 ucinet6.0 数据组。但是，必须注意，在保存数据方面你只有几种选择： Byte（字节）, Smallint（短整型）, and Real（实数）. 当从 ASCII 文件中导入数据时，可以选择三种之一。除非你又很大的数据量，否则默认的 Real（实数）应该是最好的选择。Real（实数）数据格式是最强大的，可以包括从-1E36 t 到+1E36 当中的值，他们也能包含缺失值，这些值在内部被存为 1E38.，Real 型的缺点是每个值需要 4 个字节存储，这会使得文件偏大，比如，一个 150*150 的矩阵需要 176kb 的磁盘空间（译者说，现在看来好小）。

Smallint（短整型）的每个值需要两个字节的存储空间，但只能代表从-32000 到+32000 之间的数，不允许缺失值。

Byte（比特）型是最节省的，每个值只需要 1 个字节，可以表示 0 到 255 之间的数，没有缺失值。

如果你需要缺失值，则必须使用 Real 型，当然，大部分网络分析技术不允许缺失值，只有少数 ucinet 子程序知道如何处理他们，不支持缺失值的程序自动把其转化为 0 或者其他合理值。

在选择数据型的时候，了解选型对存储空间的影响很重要，但是这对程序处理时所需要的内存空间并没有影响。不管数据时如何存储在磁盘中的，类似于 MIDS 之类的用来处理 Real 型数值的程序都是讲其看做 Real 型。类似的，想 Clique 之类只处理整数的程序乎会在读入内存中时自动将 Real 型的数据转换为整数。

关于缺失值需要注意学一点：所有的大于 1E37 的值都被认为是缺失值，包括在 ASCII 数据文件中的非数字符号，比如，一下的矩阵中存在 3 个缺失值：

0	9.7	1E38	.
16.	0	3.1	na

18.1 1e9 0 -1.2
a3 12 .013 0

与 SYSTAT 不同，ucinet6.0 把一个单独的句号 (lone period)看做 0，而不是缺失，如果你要导入包含了缺失值的 SYSTAT 数据，你应该用文本编辑器改变所有的单独句号，比如使用'NA'.

3.1 Raw Filetype

Raw 文件全都由数字构成，以矩阵形式输入。一下就是一个 raw 文件的例子：

```
0 1 1 0
1 0 1 1
1 1 0 0
0 1 0 0
```

程序通过读取第一行有几个数来确定有几列，通过计数来确定有多少行。
虽然这种文件很方便，但我们并不推荐使用，一个原因是电脑不会检查数据。如果第一行正好丢失了一个数字，程序就读不懂这个文件了，程序会认定的该文件中的矩阵列数比实际列数少 1,另一个原因是在数据文件中你不能把矩阵中的一行的数据扩散为多个记录。而且，你没法使用标签来辨别节点。

3.2 Excel Filetype

目前 ucinet 支持的 excel 版本是 4.0,5.0 和 7.0（office97）(这个东东是指导手册上说的,实际操作……)。如果你想使用其他版本的 excel，那你必须在保存数据（SAVE AS）时将他们保存为支持的格式。
注意 excel 最多只支持 255 列，所以不能被用来建立大型的网络数据表。

3.3 DL Filetype

典型的 DL 文件由一系列的数字以及一系列的描述数据的关键词，这些关键词被称为 meta-data。当然，DL 文件也可以只有 meta-data，和一个指向存储实际数据所在文件的指针。

3.4 Full Matrix Format

一个有四个参与者的 DL 文件如下：

```
dl n=4 format=fullmatrix
data:
0 1 1 0
1 0 1 1
1 1 0 0
0 1 0 0
```

关键词“dl”说明这是 DL 文件类型，他必须是文件的第一个词。“n=4”意即矩阵是 4 行 4 列，等号也可以换位空格或者逗号，形如"n = 4", "n 4", "n,4"."format=fullmatrix" 说明数据是以一个普通的矩阵格式输入的（这个值还可以使 linkedlist, lowerhalf matrix 等）。因为默认的就是 fullmatrix，所以这一短句可以省略。
“data”已经没有其他关于数据的信息了，以下的就是数据。关键词的顺序是很重要的,

如果是"dl data: n=4",整个过程就毁了。虽然我们加入了一些其他关键词，但是我们始终要保证 dl 放在第一位，然后是与矩阵维度有关的关键句，然后是其他关键词句，最后是"data"。
标点的注意事项：一般情况下，冒号表示后面有内容，比如数据集合或者标签集合。分号或者回车表示短句的结束。
每个数据值之间必须用一个以上的空格或者回车符号间隔开。所有的非数字值，除了单独的句号（lone periods），都被认为是缺失值。数据格式中行列不需要相等，只要所有的值按从左到右，从上到下的顺序排列就行，示例如下：

```
dl n=4 data:
0 1 1 0 1 0 1 1
1 1
0 0
0 1 0
0
```

3.5 Rectangular Matrices

长方形矩阵可以参照下面输入

```
dl nr = 6, nc = 4
data:
0 1 1 0
1 0 1 1
1 1 0 0
0 1 0 0
1 0 1 1
1 1 0 0
```

"nr = 6"说明矩阵有 6 行， "nc= 4"说明矩阵有 4 列。

3.6 Labels

DL 文件可能也包含参与者标签，比如：

```
dl n=4
labels:
Sanders,Skvoretz,S.Smith,T.Smith
data:
0 1 1 0
1 0 1 1
1 1 0 0
0 1 0 0
```

"labels:"表示一下四项是行列标签，标签名至多可有 18 个字符（当 **longlabels** 选项为 off 的情况下）或者 255 个字符（当 **longlabels 选项为 on 的时候** ），标签名可以由空格，逗号或者回车（或者两者一起使用）来分开。标签内不能存在空格，除非你用引号将其包住，比如"Tom Smith"。

"lable"这个词之后必须跟有冒号。
标签可以被分开为行标和列标，事实上，当矩阵不是方形的时候这是必须的，比如：

```
dl nr = 6, nc = 4
col labels:
hook,canyon,silence,rosencrantz
data:
0 1 1 0
1 0 1 1
1 1 0 0
0 1 0 0
1 0 1 1
1 1 0 0
```

另一种输入标签的方法是将其当做数据矩阵的一部分：

```
dl nr = 6, nc = 4
row labels embedded
col labels embedded
data:
      Dian Norm Coach  Sam
Mon     0    1     1    0
Tue     1    0     1    1
Wed     1    1     0    0
Thu     0    1     0    0
Fri     1    0     1    1
Sat     1    1     0    0
```

"row labels embedded" 和"column labels embedded" 说明行标和列标都嵌入在数据中，也可以简单用 "labels embedded" 来表示行列表企鹅都在数据中。

3.7 Multiple Matrices

有时在一个数据文件中存放几个相关的矩阵会比较方便。比如，我们可以度量给定的一系列参与者的多种社会关系，以下展示如何操作：

```
dl n = 4, nm = 2
labels:
GroupA,GroupB,GroupC,GroupD
matrix labels:
Marriage,Business
data:
0 1 0 1
1 0 0 0
0 0 1 0
1 0 0 1
```

```
0 1 1 1
1 0 0 0
1 0 0 1
1 0 1 0
```

"nm=2" 说明文件中包含了两个矩阵。 "matrix labels:"说明下两个词("marriage" 和 "business")是每个矩阵的标签，大致说明了每个矩阵度量的社会联系。

3.8 External Data File

有时将数据从 DL 的描述文件中分开出来会显得方便,这可以使得其他程序也可以读取数据文件（比如 SYSTAT, STRUCTURE, 和 NEGOPY）以及 ucinet，以下是示例：

```
dl n = 16
labels:
ACCIAIUOL,ALBIZZI,BARBADORI,BISCHERI,CASTELLAN,GINORI
GUADAGNI,LAMBERTES,MEDICI,PAZZI,PERUZZI,PUCCI,RIDOLFI,
SALVIATI,STROZZI,TORNABUON
datafile C:\DATA\PADGM.DAT
```

"datafile = c:\data\padgm.dat"说明了包含实际数据的文件，该文件只能包含数据。
使用 datafile 命令的缺点在于必须跟踪众多的文件。
当使用一个单独的数据文件时，ucinet 检查第一行，查看是否存在 NEGOPY 和 STRUCTURE 等所要求的 FORTRAN 格式声明。假如存在，ucinet 将从第二行开始读取，否则，就从第一行开始读取，这使得你可以在使用 STRUCTURE 和 NEGOPY 文件时中途改换为 ucinet 文件，只需确保文件中的每个值之间都有一个以上的空格。

3.9 Diagonal Absent

默认情况下，程序认为数据有一整个完整的矩阵组成，技术上我们称之为"fullmatrix format" ，可以通过如下所示方法明确之：

```
dl n=4 format=fullmatrix data:
0 1 1 0 1 0 1 1
1 1
0 0
0 1 0
0
```

在方形矩阵的情况下，省略一些值有时候显得更为方便，比如，可以省略掉对角线。

```
dl n = 4
diagonal = absent
labels:
Sanders,Skvoretz
S.Smith,T.Smith
data:
  1 1 0
1   1 1
```

```
1 1   0
0 1 0
```

程序会自动以缺失值编码填入空缺的位置中。

3.10 Lowerhalf and Upperhalf Matrices

一种做法是只输入对称矩阵的下半部分：

```
dl n = 4
format = lowerhalf
diagonal = absent
labels:
Sanders,Skvoretz
S.Smith,T.Smith
data:
1
1 1
0 1 0
```

Ucinet 会自动把上半部分补齐，并以缺失值填入对角线。同样，也可以输入对症矩阵的上半部分以关键词"upperhalf" 代替 "lowerhalf"即可。注意如果"diagonal absent" 语句呗省略了，程序展示出缺失值（expect a diagonal value to be present. ）

```
dl n = 4
format = lowerhalf
labels:
Sanders,Skvoretz
S.Smith,T.Smith
data:
2
1 2
1 1 2
0 1 0 2
```

3.11 Blockmatrix Format

另一种在建立模型矩阵时很有用的格式是“blockmatrix”（分块矩阵）。在这种格式中，你可以输入方向来创建数据而不是逐一输入。比如要输入如下矩阵：

```
2 1 1 1 1 0 0 0 0 0
1 2 1 1 1 0 0 0 0 0
1 1 2 1 1 0 0 0 0 0
1 1 1 2 1 0 0 0 0 0
1 1 1 1 2 0 0 0 0 0
0 0 0 0 0 2 1 1 1 1
0 0 0 0 0 1 2 1 1 1
0 0 0 0 0 1 1 2 1 1
0 0 0 0 0 1 1 1 2 1
```

```
0 0 0 0 0 1 1 1 1 2
```

使用 blockmatrix 格式你可以这样创建：

```
dl n = 10 format = blockmatrix
data:
rows 1 to 10
cols 1 to 10
value = 0

rows 1 to 5
cols 1 to 5
value = 1

rows 5 6 7 8 9 10
cols 5 to 10
value = 1

diagonal 0
value = 2
```

在 blockmatrix 格式中，你区分出一系列单元，并给他们幅值。在示例中，前三行语句给所有单元赋 0，次三行（忽略空格）给左上角的矩阵赋值为 1，再后三行作用类似，最后两行将主对角线上的所有值赋为 2.

考虑一下另一个矩阵：

```
100   0   0   0   0   0
 90 100   0   0   0   0
 80  90 100   0   0   0
 70  80  90 100   0   0
 60  70  80  90 100   0
 50  60  70  80  90 100
```

根据如下示例用 blockmatrix 格式输入这个矩阵：

```
dl n = 10 format = blockmatrix
data:
rows all
cols all
value 0
diag 0
val = 100
diag -1
val = 90
diag -2
val = 80
d -3
v = 70
```

```
d -4
v = 60
d -5
v = 50
```

关键词“all”表示所有行和列，"rows", "cols", "value", 和"diagonal"可以采用首字母缩写。

3.12 Linked List Formats

网络分析中的一系列重要格式就是被称为 linked list （链表）的格式，这种格式中参与者只保留数据之间实际发生的联系，忽略那些不发生的联系，这些格式的特别之处在于接受字符型数据。

There are two basic types of linked list formats: "nodelists" and "edgelists". Each of these types in turn has two variants, one for 1-mode data and one for 2-mode data. Only the edgelists allow valued data.

有两种基本的链表格式类型： "nodelists" （节点列表）and "edgelists"（边界列表）。这两种类型各有两种变式，一种用于单模数据，另一种用于 2 模数据，只有 "edgelists" 允许赋值数据。

3.12.1 Nodelists

节点列表由一系列与给定节点相连的节点组成。节点列表可以表示成两种格式：*nodelist1* 和 *nodelist2.nodelist1 用于方形单模矩阵，比如网络工作或者亲近性。* 第二种格式用于长方形的 2 模矩阵，比如说事例变量（case-by-variable）矩阵或者项目使用（item-by-use ）矩阵。两种格式都可以接受字符型数据，nodelist 格式形如：

```
dl n = 4, format = nodelist1
labels:
Sanders,Skvoretz,S.Smith,T.Smith
data:
1  2 3
2  1 3 4
3  1 2
4  2
```

"format=nodelist1"表示这种格式中每行的第一个数据代表拥有这行数据的参与者（我们称为“ego”（自我）），该行剩下的数字表示和 ego（自我）直接相连的参与者。比如。比如数据中的第三行表示参与者 3(S.Smith)和 Sanders 、 Skvoretz 有直接联系。ego 的顺序并不重要，ego 的联系者（alter）的顺序也不重要。没有联系者的 ego 被直接忽略（但是始终计算在“n=”短句中，也在标签列表中显示）。

如果在一行中你需要填入过多的联系者，你可以为这个参与者再开辟新的一行，但是在新的一行的开头必须指出这些联系者是属于哪个参与者的。比如，以下的例子和上面表示的相同意思：

```
dl n = 4, format = nodelist1
labels:
Sanders,Skvoretz,S.Smith,T.Smith
data:
 1  2 3
 2  1 3
 2  4
 3  1 2
 4  2
```

注意参与者 2 的联系是如何分布在两行之中的。
你也可以使用标签而不是数字来代表相同的数据，示例如下：

```
dl n = 4, format = nodelist1
labels embedded
data:
sanders   skvoretz s.smith
skvoretz  sanders s.smith t.smith
s.smith   sanders skvoretz
t.smith   skvoretz
```

把所有的标签放在文件的上半部分或者把标签嵌入在数据之中都是可以的（要确保存在“labels embedded”命令）都是可以的。这样做的好处是结果矩阵的行列会按照你所列的标签的顺序排列。比如，以下内容将会创建一个矩阵，在其中，行列将按字母顺序表排列：

Dl n = 4
format = nodelist1
labels:
Alfred, Betty, Chuck, David
Labels embedded
Data:
Betty chuck alfred
Alfred chuck david
David chuck betty
Chuck betty

如果没有把标签列在前面，结果矩阵会按照遇到标签的顺序排列行列：第一行/列会是 Betty，第二行/列是 Chuck，第三行/列回事 Alfred，如此种种。
nodelist2 格式用于表征两种不同实体间存在联系的 2 模矩阵数据，比如，个人和组织，个人和活动。示例如下：

```
dl nr=3, nc=4 format = nodelist2
row labels embedded
column labels embedded
data:
benny sex_roles, stats
```

```
ginny aging, intro
sally stats, intro, aging
```

这些数据记录了各自可以教四门课的三个人。第一行数据说明 Benny 教 Sex Roles 和 Statistics，第三行说明了 Sally 也教 Statistics，但还同时教授 Intro and Aging。每行的第一项表示行实体，该行剩下的项目表示列实体。

3.12.2 Rankedlist Formats

Rankedlist（排序表）格式是节点列表格式的一种简单变形，在其中，与一个给定节点有联系的节点被假定为按照一个有意义的顺序输入。程序会基于这种顺序安排联系的强度（tie strengths），比如：

```
dl n = 4, format = rankedlist1
labels embedded
data:
sanders  skvoretz s.smith
skvoretz  sanders s.smith t.smith
s.smith  sanders skvoretz
t.smith  skvoretz
```

我们看到第一行数据("sanders skvoretz s.smith")，这表明对于参与者 Sanders，参与者 Skvoretz 排在第一，而 S.Smith 排在第二。没有被列出的参与者即被默认设定为 0。除非在数据声明之前出现"recodena = no"短句。在这种情况下缺失值是被支持的。以上文件的结果就是如下矩阵：

	Sanders	Skvoretz	S.Smith	T.Smith
Sanders	0	1	2	0
Skvoretz	1	0	2	3
S.Smith	1	2	0	0
T.Smith	0	1	0	0

如同 nodelists，同样存在 rankelist2 用来表示 2 模数据：

```
dl nr=3, nc=4 format = nodelist2
row labels embedded
column labels embedded
data:
benny sex_roles, stats
ginny aging, intro
sally stats, intro, aging
```

在这种情况下，"benny sex_roles, stats" 这行表示在矩阵中 benny 那一行里，sex_roles 的值会是 1，stats 的值会是 2.

3.12.3 Edgelist Formats

Edgelist（边界列表）格式将重点放在每个连接（或者是矩阵中的单元）上而不是每个自我的参与者列表上。有两种 edgelist 格式：edgelist1 和 edgelist2.edgelist1 用于方形单模矩阵，比如网络或者亲近性。edgelist2 用于长方形 2 模矩阵，比如示例变量（case-by-variable）矩阵和项目使用（item-by-use）矩阵。两种格式都可以接受字符型数据。

edgelist1 格式由一系列的边界（即连接），有时候伴有他们的数值构成，示例如下：

```
dl n = 4 format = edgelist1
labels:
Sanders,Skvoretz,S.Smith,T.Smith
data:
1 2 1
1 3 2
2 1 1
2 3 1
2 4
3 1 1
3 2 na
4 2 10.3
```

在本例中，数据的第四行说明参与者 2 和参与者 3 之间又一个强度为 1 的联系。如果忽略联系强度，如同第五行，那么它将被假定为 1.0，如果忽略了一对，比如说（4,3）这一对，那么他们之间的联系江北假设为 0.0.和矩阵格式一样，强度值可以是整数或者实数，这由存储时的类型决定。如“na”等的非数字值会被编码为缺失。由输入数据产生的邻接矩阵示例如下：

	Sanders	Skvoretz	S.Smith	T.Smith
Sanders	0	1	2	0
Skvoretz	1	0	1	1
S.Smith	1		0	0
T.Smith	0	10.3	0	0

联系可以以任何顺序展示，但是每行只能有一种联系。
edgelist1 格式的一个特征是与字符数据交互的能力。它的主要作用体现在和“labels embedded”声明联合使用时：

```
dl n = 3 format = edgelist1
labels embedded
data:
sanders skvoretz 1
sanders s_smith 1.782
skvoretz sanders 1.09
```

这引发了一个问题：邻接矩阵中行列的标签将以他们在数据记录中的前两块区域中出现的顺序排列。因此，在本例中，3 行 3 列的矩阵中，第一行/列是“Sanders”，第二行/列是“Skvoretz”，第三行/列是“S_Smith”程序能够城阙处理任何长度的字符，但是在存储为行列标签时只取前 19（如果 longlable 值为 on 时则是 255）个字符。
注意由这些数据产生的矩阵的行列都会默认以字母顺序排列。比如，由如上的数据生成的矩阵第一行会对应 s_smith，第二行会对应 sander，第三行对应 skvoretz。通过加上“alpha = no”短句，可以防止自动按字母顺序排序，在这种情况下行和列会按照他们

在文件中的顺序排序。由如上数据生成的矩阵第一行会对应 sanders，第二行对应 skvoretz，第三行对应 s_smith。你也可以在将多种关系存储在一个文件中，只需用竖线分割，比如：

```
dl nm=2 n = 3 format = edgelist1
labels embedded
data:
sanders skvoretz 1
sanders s_smith 1.782
skvoretz sanders 1.09
!
sanders s_smith 1
skvoretz sanders 1
```

edgelist2 格式和 edgelist1 格式相同，只是被用于长方形 2 模数据，考虑如下示例：

```
            COHESION      SIMILARITY   CENTRALITY
CLIQUE        1             0             1
CLOSENESS         1             0             1
REGE          0             1             0
DENSITY       1             0             1
CONCOR        1             1             0
```

使用 edgelist2 格式的输入文件如下：

```
dl nr = 5 nc = 3, format = edgelist2,
labels embedded
data:
clique cohesion
clique centrality
closeness cohesion
closeness centrality
rege similarity
density cohesion
density centrality
concor cohesion
concor similarity
```

在示例中，所有出现的边界权重都为 1，因为数据文件中没有权重，以下的输入文件有权重边界：

```
dl nr=2, nc=3 format = edgelist2
labels embedded
data:
sanders skvoretz 1
sanders s_smith 1.782
skvoretz sanders 1.09
```

虽然该文件中的数据和用来解释 edgelist1 格式中的数据相同，但是结果却是相差很大的。

```
            Skvoretz S_Smith  Sanders
Sanders     1        1.782    0
Skvoretz    0        0        1.09
```

注意在生成矩阵中的只有两种不同实体（Sanders 和 Skvoretz）作为行，却又三种不同实体 (Skvoretz, S_Smith 和 Sanders)作为列。同样要注意的一个事实是，在本例中行和列中碰巧有相同的标签但这什么也代表不了，程序不会把他们当做一样的来对待。

3.12.4 The EdgeArray1 Format

Edgearray（边界阵列）格式和 edgelist 格式想象，但是该格式允许每个连接有多个值，比如：

```
dl nm = 4 n = 3 format = edgearray1
labels embedded
matrix labels:
avgcalls
yearstogether
pctincommon
kin
data:
sanders skvoretz  1    62   33.7 0
sanders s_smith   1.782 29 66.2 1
skvoretz sanders  1.09 59   99.0 0
```

3.15 UCINET Spreadsheet Editor

你可以直接输入或者从其他数据表复制到 ucinet 的数据表中。在 ucinet 中通过单击 edit 按钮或者点击 **Data>Edit** 或者 Ctrl+E 打开 ucinet 编辑器。将你的数据表的数据保存在剪贴板中，然后通过 Ctrl+V 快捷键以及点选 ucinet 中的 **Edit>Paste**。**然后你可以将你的数据保存为 ucinet 文件。如果你希望直接输入数据那么你可以在编辑器中输入。注意你只需输入一次标签，一些数据已经被填入相关联的行列中。如果你已经知道了你的数据有多大你可以先填写文件的最后一行和最后一列，你可以在文件头部输入标签，如果你的数据时对称的，在输入前店家 Asymmetric 模式按钮，这样你只需要输入一半数据，程序会自动填写另一半数据。在数据表中，只需要输入非零值，当所有非零值输入完成后，点击 Fill 按钮将会以 0 填充所有为空的单元。如果意外地超出了你要求的矩阵尺寸，那你必须删除多余的行列而不是用空格来充填这些行列。如果你的数据有一个以上的关系那么使用右侧工具栏上的+按钮来增加额外的矩阵。在右侧工具栏上也可以使用 rename sheet 重命名文件。**

编辑器允许使用二维或者三维图。要利用图来载入 ucinet 数据到编辑器中，虚拟将你想展示的数据分块。单击 **edit>copy 来将数据保存在剪贴板中，然后点击 edit>paste 来将数据部署到数据表图中。最后点击工具栏上在 Symmetric/Asymmetric Mode 按钮边**

上的 graph 按钮，程序向导将会引导你创建你所需要的图形。
如同 excel 一样，ucinet 数据表将列数限制在 255 列以内，所以本法不能被用在较大的数据组中。
~4~
Data Processing
raw 数据，在分析前经常需要经过转换或者修改。有时，修改的原因是纠正数据采集技术中的错误或者误差。比如，你让一个群体中的成员根据亲密度互相评价打分，这会出现一种情况即一些较低的分数值（比如说低于 10 分）实际上是随机的因为这些人根本不认识对方。在这种情况下，你可以把所有低于 10 分的评分都定位一个固定值。或者，数据中的一两个值在输入过程中出错了，如果数据量小的话，最好的处理方式就是使用数据表编辑器，这已经在之前提到过了。在这一部分我们将关注对数据的一些根本修改。
数据在用于特定的分析程序的时候可能会需要特定的格式，因此有必要对其转换。比如，你搜集表示了一系列人之前关系的赋值数据。然后你需要确定定位 Luce 和 Perry 的团，这时你就需要二分数据因为团被定义为节点之间是否存在连接的普通图。
数据转换的另一个原因是在原有数据的基础上建立一个新的更有效的分析。比如，你拥有一系列参与者在一系列事务上的数据，同时你有这些参与者之间联系强度的数据。一个自然需要检测的假设是这些关系强度的相似性，你可以使用 QAP 程序测试假设。不过首先你必须掌握参与者之间的相似性，这可以通过计算参与者矩阵的行关联得到。UCINET 6.0 可以实现多种数据转换。在讨论这些转换之前，必须注意这些程序并没有修改实际存在的数据，他们只是产生了一个新的反映相应转换关系的数据组（当然，新的数据组可以覆盖原有的数据组），然后新的数据组被提交分析。

4.1 Subgraphs and Submatrices
最通常的数据调整应该就是在网络中移除节点了，对应于邻接矩阵中就是移除对应的行或列。在 ucinet 中使用 **Data>Extract 程序可以很容易地由输入矩阵生成一个子矩阵。改程序的输入可以是：一，一个数据组，二，需要保存或者删除的行列的列表，你可以选择留下或者丢弃什么，如果你只是社区一两行/列，那么，指出他们肯定会比把剩下的列出要方便高效。当运行 Extract 时，需要提供以下信息：**

```
  |Input dataset:                                    |
|Are you going to KEEP or DELETE?  KEEP              |
  |Which rows to KEEP?              ALL              |
 |Which columns to KEEP?            ALL              |
 |Which matrices to KEEP?           ALL              |
  |Output dataset:                 EXTRACT            |
```

在 "Which rows to KEEP?"之后你需要输入参与者列表，比如 "1 TO 10, 12 TO 24"。行和列的操作必须统一，因为我们在网络中移除节点以后行和列总是保持一致的，在这种情况下你可以直接输入“same”。如果数据有标签并且你只记得标签，那么你可以在容器右边点击 L 键然后选择真确的参与者，使用 ctrl 键时你可以同时选择多个参与者。或者，你可以填入包含有群指示矩阵的 ucinet 数据组名，比如由几个分析程序产生的结果（例如 Clique）。一个群只是矩阵是一个二值的群-参与者矩阵，在其中如果 j 是 i 的群的成员则 X(i,j) = 1，否则 X(i,j) = 0。比如 Clique 程序将由其生成的群指示矩阵命名为 **CliquesSets，如果你想从中调出第六个团的节点组成的子图，那么你应该输入：**

Which rows to KEEP**? CliquesSets row 6**

Which rows to KEEP**? CliquesSets row 6**

在 column 的参数中你可以填入以上相同的内容或者直接填入“same”,这就是说，我想保存的参与者是由名为 CliqueSets 的数据组中的第 6 行给出的。重要提醒：如果数据名中包含了空格，必须使用引号包住。比如：

Which rows to KEEP**? “My Cliques Sets” row 6**

当然，你要提供的数据组不一定是由 Clique 创建的，你可以自己创建一个包含 1 和 0 的 ASCII 文件然后倒入数组局的信息，比如：

```
dl nr=1 nc=10 data:
1 0 0 1 1 1 0 0 1 0
```

如果你把数据命名为 mygroup，那么你可以如此填空：

Which rows to KEEP? **mygroup row 1**

然后你需要在 column 栏中重复上述内容。这将会产生一个子图，该子图中 1,4-6，以及 9 这几个参与者之间是有联系的。显然在其他场合你会想选择其他行列。

4.2 Merging Datasets

有时将关于同一个参与者的不同数据保存在不同的文件中是比较方便的，但是，把这些数据汇集到一个文件中也会很有用。比如，有人可能会希望将 Sampson 的关系单独分开分析，但是对于一些程序（比如 positional analysis 位置分析）你会希望分析是同时基于所有的方法的。**Data>Join 程序可以在行、列、矩阵三个维度上将数据整合到一起，比如，考虑以下两个矩阵：**

矩阵 1：

```
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
```

矩阵 2：

```
2 2 2 2
2 2 2 2
2 2 2 2
2 2 2 2
```

根据行整合即将第二个矩阵放在第一个矩阵底部形成 8 行 4 列矩阵：

```
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
2 2 2 2
2 2 2 2
2 2 2 2
2 2 2 2
```

根据列整合就是把第二个矩阵放在第一个的右侧组成 4 行 8 列矩阵：

```
1 1 1 1 2 2 2 2
1 1 1 1 2 2 2 2
1 1 1 1 2 2 2 2
1 1 1 1 2 2 2 2
```

根据级别整合就是将两个矩阵先后放在一个数据组中。构成一个 4 行 4 列的双矩阵数据组：

```
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1

2 2 2 2
2 2 2 2
2 2 2 2
2 2 2 2
```

4.3 Permutations and Sorts

与生成矩阵子集关系密切的一个话题是行和/或列的重排。一般来说，将一个矩阵重排使得同一个群的成员处在相邻位置，这使得整个关系网的结构更易于把握。Ucinet6.o 有两种基本的重排方法：permutations （交换）和 sorts（排序）。.

Permutations 即用户直接应用一种新的顺序。比如，你可能告诉 Permute 程序节点的顺序应该是“1, 6, 2 to 5, 10 9 8 7”这代表节点 1 排在第一个，节点 6 排在第二个……。示例如下：

```
|Input dataset:                              |
|New order of rows:                          |
|New order of cols:                          |
|New order of matrices:                      |
|Output dataset:            Permuted         |
```

Sorts 则是通过参考属性的非直接方式来重排序。比如，你可能希望程序将节点以向心性递减的顺序排列。需要达到这个目的，首先得确保磁盘中存有包括每个节点的向心性的数据、然后你告诉 Sort 程序那个数据组包含了标准向量（比如向心向量）以及是数据中那一行或列，示例如下：

```
 |Input dataset:                             |
|Dimensions to be arranged:     Both         |
 |Sort order:                  Ascending     |
|Criterion vector (sort key):                |
 |Output dataset:               Sorted       |
```

4.4 Transposing and Reshaping

行列对换也是很方便的。比如，在世界系统中，通常用 X（i,j）表示从 j 国转移到 i 过的物资。然而在网络中，X(i,j) 一般代表从 i 到 j 的关系。如果需要转换这个矩阵，只需要使用 **Data>Transpose** 程序。再举个例子，假设你有一个 **2** 行 **50** 列的数据组，你想得到第一行和第二行之间的散布式绘图（scatter plot ），但是在 **ucibet** 中只能绘制列的图，解决办法就是将行列互换。

如果你拥有 3 维数据（即由多个矩阵组成的单一的数据组），**Data>Transpose** 程序可以降每个矩阵输出到一个数据组中。那么，如果你想转换的不是行和列，二十列和级（即矩阵）时该怎么办呢？比如，在数据组中包含了 **12** 个 **17*17** 的矩阵。每个矩阵代表那一周搜集的数据。行代表自我，列代表联系人，级代表星期。数据中的值代表在每个每周中每个自我在联系人中的排名。假设我们将数据组织成所有的联系人在给定的自我中按周的排名，我们就需要以列代表星期，以行代表自我，以级代表联系人，也就是说，我们需要转换原始数据的列和级，这可以通过 **Tools>Matrix Algebra** 轻松完成：

-->tnewc1 = transp(newcomb1 columns levels)

这句话的意思将数据组 Newcomb1 的列和级互换并将结果存放在 Tnewc1 中。

与此相关的程序是 **Data>Reshape**。与 **Transpose** 不同的是，改程序改变矩阵的维度而不改变数据顺序。比如，假设你有一个 **10*10** 矩阵。有时将其转换为一个 **100** 个元素的向量会比较方便。这可以通过 **Reshape** 实现，你可以告诉程序你需要 **100** 行 **1** 列。文件中的数据元素会按照他们出现在矩阵中的顺序（从左到右，从上到下）写入新的文件。

4.5 Recodes

经常用到的一种转换是改变矩阵中的一系列值。比如，将 0 到 9 的值全部修改为 0，将 10 到 19 的值全部修改为 1，等等。实现该目的的最简单方法是 **Transform>Recode**。示例如下：

```
| Values     0      to     9      are recoded as    0      |
| Values    10      to    19      are recoded as    1      |
| Values    20      to   100      are recoded as    2      |
| Values    na      to    na      are recoded as    0      |
| Values            to           are recoded as           |
```

我们可以通过 tab 键来移动修改数值。注意在第四行我们将所有的缺失值（na）都转换为了 0.

Recode 命令经常被用于 0 和缺失值之间的互相转换。需要注意的是，改程序没法值转换主对角线上的值，如需转换主对角线上的值，应该使用 **Transform>Diagonal**。

经常遇到的重编码情况是将一个矩阵二分，即变成二值矩阵，只有 0 和 1.在度量参与者之间的练习强度的时候会用到这个。但是一般给定的这种分析都已经是二值矩阵数据了。有时我们需要它是因为你想要运行的 ucinet 程序需要二进制数据。如果这样，程序会自动将大于 0 的值转变为 1.但是这可能不适合你的数据（比如你的数据都是负数），所以提前自行二分一下数据可能会更好。

Transform>Dichotomize 程序将把你提供的每个中断值与矩阵的每个单元值进行比较

来对矩阵进行二分。根据你的选择，程序可以将 1 赋给任何大于、大于等于、等于、小于等于或小于给定中断值的单元数。

另一种重排的操作是对值得反转（ **Transform> Reverse**)。比如，对于一个由 1 和 0 构成的二值矩阵，这意味着将所有的 1 改变成 0，将所有的 0 改变为 1.，这可能用于改变“喜欢”和“不喜欢”的关系。对于赋值数据，该程序将会使得最大数和最小数对换，次大数和次小数对换，比如，对于由 0,1,2 构成的数据，程序会把所有的 0 改成 2，把所有的 2 改成 0，而保留 1.

4.6 Linear Transformations

矩阵数值的反转和其他重编码的输出数字会有微妙的不同，因为输出值和输入值之间有着简单而有意义的数值间的联系。实际上，反转只是一种现行运算 Y(i,j)=*X(i,j) + Min(X)-Max(X),为了得到更通用的* Y(i,j) = *bX(i,j) + m 的效果，需要使用* **Matrix Algebra 包，比如，将一个名为 DAVIS 的矩阵乘以 7 你可以在 Matrix Algebra 中如此输入：**

-->davis7 = linear(davis 7 0)

这意味着矩阵中的每个值被乘以 7，然后加上 0，，并肩结果保存在 **DAVIS7 数据组中，如果要将矩阵中的值除以 2 然后再减 1，可以如此输入：**

-->davis7 = linear(davis 0.5 -1)

4.7 Symmetrizing

对称化矩阵 X 是指强制所有 i 和 j 使得 X(i,j) 等于 X(j,i)。对称化数据的需求来自于多个方面。一种情况是数据采集的情况就是逻辑对称的，比如“和……一起吃过午饭”，但是由于计量错误，实际得到的数据不是对称的。比如 Oliver 回忆起曾经将武器卖给 Iran 和 Ronald，但是 Ronald 回忆不起这件事情。另一种情况是非对称的关系，采集数据时为“把钱借给……”，但是在分析关系时一般是“在经济上与……有联系”在这两种情况下，我们都需要将对应的 X(i,j) 和 X(j,i）化为一致。

Ucinet 提供了对称花的多种选项。包括用二者的平均值、最小值或者最大值代替这两者。一些不常使用的选项有：如果对应的 X(i,j) 和 X(j,i)相等，那么将他们都设为 1，否则都设为 0。在效果上，这等同于建立一个新的矩阵来表征原有矩阵是否对应相等，该矩阵所有值的平均值即给出了一致的对数的份额。

4.8 Geodesic Distances and Reachability

一个距离矩阵是包含了每一对节点之间的距离的矩阵。两个节点姐的距离是指所有联系中最短的数字。理论上，距离越大，影响和交流越小，距离矩阵可以用来指示影响力和内聚力。

运行 Network>Cohesion>Geodesics 程序来处理距离矩阵.在 ucinet6.0 中 ，距离的概念被应用于各种赋值图中。比如，你的数据可能记录了参与者间联系的强度。从 a 到 b 再到 c 再到 d 的强度被定义为他们之间最脆弱的联系。两点间的距离路径被定义为最强的联系。另外一种赋值数据由一对节点之间的路程和费用组成。在这种情况下，距离路径被定义为最实惠或者最短的路径，通过计算每对节点间联系值来获得的。

运行 Distance 会给出将距离转换为亲近性的选项。当距离矩阵被用来代表亲近度、影响时值越大表示关系或影响越大的时候这会比较有用 。最简单的选项是从 n（节点号）减去距离值，这和 Reverse 很像。另一个选项得到距离的倒数 (1/d(i,j))，并乘以可能

的最大距离（n-1）使得所有的数据值在 0 与 1 之间。理论上的一个选项（建立在一个有问题的假设上）将一个两个节点间的距离值定为一个常数。这样的话，如果常数是 0.5，那么直接连接的一对节点的记为 0.5.一对同时存在的两个连接被记为 0.25，一对同时存在的 3 个连接被记为 0.125（译者：怎么回事 0.125 呢？），诸如此类。另一个理论热点是由 Burt 于 1976 年提出的并在 STRUCTURE 中实现的：给定一个 i 和 j，由 1 减去从 i 到达参与者的步数小于从 i 到 j 的步数的参与者所占的份额。如果 i 和 j 关系疏远，那么，以从 i 到 j 的步数能到达的其他参与者就很多，份额大，因此得到的值会很小。

与距离（distant）关系密切的一个概念是抵达（reach）在邻接矩阵 X 中，该概念一般指数据可以达到的点。在其中，如果存在从 i 到 j 的路径，那么 Y（i,j）=1。在 ucinet6.0 中，我们扩展了数种赋值矩阵的概念。比如，对于一个记录关系强度、容量或者频率的矩阵，从 i 到 j 的抵达（reach）是两者之间的路径的最强者。如果强度值被重排为 1 和 0，两点之间的最强路径强度将始终是 1，除非不存在任何路径。这与一个相似概念 reachability（可达性）是不同的。对于消耗数据，reachability（可达性）被定义为最实惠的路径的消耗。对于可能性数据，reachability 被定义为两点之间最可能的路径，reachability 矩阵也可以用来度量向心力（内聚力）及影响力。

其他可以用来度量影响力或者内聚力的指数是最大流量（maximum flows）。运行 **Network>Cohesion>Maximum Flow 调用这些程序。**

4.9 Aggregation

另外一种转换包括了聚合图或者聚合矩阵，一个典型的例子就是将一个参与者的的多种关系聚合在一起。比如，一项研究可能会记录一个群众成员间的多项不同的关系。影响力，相互依赖或者联系强度可能会成为将他们捆绑在一起的关系。有些人称之为 "multiplexity"，但是在 ucinet 中，我们为其保留了一个更容易理解的概念，即将两个参与者捆绑在一起的联系的集合。

有两种方法可以将所有关系整合在一个数据组中的办法。第一种方法是使用 **data>CSS。CSS 程序有多重聚合的方法。在此我们的目标是，纤小的是正确的选择。第二种方法是使用 Matrix> Algebra**.比如，将名为 PADGETT 数据组中的两种关系聚合在一起：

```
-->agpadge = total(padgett rows cols)
```

该命令的意思是“将 PADGETT 的值总括起来，根据行和列将结果拆分，然后将输出保存在 **AGPADGE 数据组中**”。我们可以这么理解：将 PADGETT 理解为 3 维包含行、列、级的三维表。我们想要处理数据使得结果只有行和列。如果我们还想将列压缩处理，我们可以这么输入：

```
-->rowmarg = total(padgett rows)
```

类似地，为了把所有的数值集中在一个数据表中，我们可以使用 total 命令，不包含任何参数。

```
-->bign = total(padgett)
```

Transform>Multiplex 程序可以被用于将一个数据组中的一系列矩阵合并为一个单独的明确的赋值矩阵。在其中，每个关系值的合并结果都是一个独立的值，比如，有一个数据组包含以下两个矩阵：

```
0 1 0 1
0 0 1 1
1 0 0 0
1 1 0 0

0 1 1 0
1 0 1 1
1 0 0 1
0 1 0 0
```

该网络的成员总共展示了 4 个不同的关系的汇总（根据遇到的先后顺序）：两种联系都没有的，不存在第一种联系但是存在第二种联系的，存在第一种联系但是不存在第二种联系的，这些数据汇合在一起的多路矩阵如下：

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

这个单独的多路矩阵将上面两个矩阵的所有数据都包含在内。一个多路矩阵（或者任何赋值矩阵）可以由 **Transform> Multigraph 分解。该命令建立将为输入矩阵的每一个独立值建立一个新矩阵。比如，以上所示的多路矩阵将会被分解为：**

```
0 1 0 0
0 0 1 1
1 0 0 0
0 1 0 0

0 0 1 0
1 0 0 0
0 0 0 1
0 0 0 0

0 0 0 1
0 0 0 0
0 0 0 0
1 0 0 0
```

第一个矩阵指示了在 1 级中那些节点是相连的。第二个矩阵只是在 2 级中那些节点是相连的，等等。0 级也可以被认为是有效连接。输出矩阵可以被设置为累积的，这样第二个矩阵将可以指示在 2 级或者以下级那些节点是被连接的。

另一种分解是将节点分类或分群。这在分块密度矩阵的分解结果中比较常见。分块模型的主要输出是拥有同种特质的一部分参与者。这就使得一个给出不同块参与者之间联系的平均值的密度矩阵成为必要。比如，如果我们将如上的两个矩阵分块那么前两

个参与者将会是同一个分类，后两个参与者是另一个分类，我们可以得到以下的两个密度表：

```
.25  .75
.75   0

.50  .75
.50  .25
```

通过 **Transform>Block** 可以得到密度表。改程序将数据矩阵和分块向量作为输入。一个分块向量指出网络中的每个参与者所在的分类。比如，如果 **10** 个参与者被分为两类，分块向量可能会是：{1,2,1,2,2,1,1,2,1,1}。因此，每一个参与者都由一个数字指出属于哪一分类。分块向量可以人工输入或者由其他程序产生（比如由 CONCOR 或者 CLUSTER 生成）。

一个更可靠的分解节点的程序是 **Transform> Collapse**。该命令可以求数值和，求平均，求最大值或者最小值。但是，输入数据不是分块向量，而是需要被分解的节点的列表，比如，前面提到的分块向量可以这么表示：

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

4.10 Normalizing and Standardizing

另一种转换是 normalization（正态化），这是指改变数据矩阵中的数字的规模，不管是对于矩阵整体或者矩阵中的行或者列。比如，有人可能希望数据矩阵中同一行的数据有着相同的平均值或者标准差。这需要在从不同的受访者处采集每行数据时就做好。比如，如果受访者被要求评价从他们自己家到其他参与者家的距离，有些受访者会以英尺为单位，有些会以码为单位，或者米。在这种情况下，为了方便比较，有必要将每行的所有数据统一在一个共同的单位下。

Transform>Normalize 可以根据平均值，边界，标准差，平均值和标准差一起，欧几里得规范，以及最大值来使得数据正态化。每种正态化都可以应用在整个矩阵或者单独的行列中。比如，你可以使得每行拥有相同的平均值（**0.00**）以及相同的标准差（**1.00**），当然，你也可以使每一列正态化，**Normalize** 程序甚至允许你同时将行和列正态化。还有一种就是将整个矩阵正态化这样整个矩阵的平均值就是 **0**，当然某一行或者列的平均值不一定是 **0.**

Normalizing 操作也可以通过 **Matrix Algebra** 包来实现。比如，假设你想正态化一个称为 **trade** 的 **30*30** 的矩阵式的列加到 **100**，你可以如此实现：

```
-->COLSUM = TOTAL(TRADE COLUMNS)
-->NTRADE = DIV(TRADE FILL(COLSUM 30 30))
```

第一个命令建立了一个航向量（实际上是一个一行 30 列的矩阵）包括了列的和。第二个命令根据列的和将 trade 数据组的输入分解。由于 DIVIDE 命令要求矩阵有相同的尺寸，FILL 命令用于对行向量做 30 次复制以得到一个 30*30 的矩阵。

4.11 Mode Changes

与以上所讨论的很不同的一种转换是本节要介绍的由单模数据到 2 模数据的衍生。单模矩阵是指在该矩阵中行和列都代表同一个客体（一般是参与者）。2 模矩阵是指在其中行和列各自代表一个客体。在网络分析中，2 模数据经常是联合数据的格式：包括了与一系列的参与者有关的群或者事件的信息。如果 i 与群/事件有联系，那么 X(i,j) = 1，否则 X(i,j) = 0。从该矩阵中，我们可以通过计算每个参与者所共有的群/事件的数目来计算参与者-参与者矩阵。结果是共同发生或者共为参与者矩阵，此矩阵可以像普通（赋值）网络数据一样用于分析。我们也可以生成事件-事件矩阵，矩阵统计了了同时参与每对事件中的两个事件的的参与者的数目。

从 2 模矩阵中生成单模矩阵的最简单的方式是 **Data>Affiliations。程序将 2 模矩阵作为输入并且计算每对参与者所共同参与的矩阵数目或者计算每对活动所共有的参与者的数目。了解这个程序在计算“共同出现”上是和使用原始的非正态化的方法计算活动参与者相似性一样的。其他的类似方法也适用于给定的分析，特别是对于赋值数据。从一大堆的相似性方法中选择，运行 Tools>Similarities，比如，准确匹配**（exact matches **的方法在二值数据上应该会是一个热点选择，该程序会计算例如两个参与者共同参与或者都不参与的时间的编号，在该情况下，不参与同一个事件被用来评估两个参与者之间的联系。对于赋值数据，关联**（correlation）**方法应该是最好的选择。要使用相异**（dissimilarity）**方法比如欧几里得距离，运行 Tools> Dissimilarities。**

这种模式转换的一个特例是从一个向量转化为单模矩阵。比如，一次网络研究可能搜集每个参与者的基本信息，诸如年龄、性别、收入。接下来的问题是，这些数据和参与者之间的关系有什么联系？回答该问题的一种方法是将这些数据转换为网络并使用 QAO 程序联系他们。比如，考虑到性别因素，我们将其中男性假设为编号为 1 的向量 S，女性编号为 2。我们可以建立一个参与者-参与者矩阵，该矩阵记录每对参与者是否有着相同的性别。也就是说，我们假设如果 i 和 j 同性，X（i，j）=1，否则 X（i，j）=0.然后我们可以应用 QAP 来联系“与……同性”和“是……的密友”。对于共有变量比如收入，我们更可能考虑“收入水平的不同是否影响友情”而不是“有着相同收入的参与者之间是否倾向于成为朋友”在前者之中，我们使 X(i,j) 等于 i 和 j 之间的收入差的绝对值。。

从单独的属性数据中生成网络数据的最简单方式是 **Data>Attribute，该程序将一个单独的属性作为输入并使用你选择的三个维度（等同，不同，绝对不同）来生成一个方形矩阵。从上面的讨论来看，很明显的，将向量转换为矩阵或者将属性转换为网络是和计算 2 模矩阵中的相似性完全一样的。问题中的 2 模矩阵是属性向量，是一个简单的 n 维 1 列矩阵，在其中，行代表参与者，列代表占有物或者属性。**

反转——从单模矩阵生成 2 模矩阵——也是网络中的热点。比如，**Transform> Incidence 程序将一个节点-节点邻接矩阵转换为 2 模的节点-联系矩阵，在其中，** X(i,j) = 1 表示 i 和 j 相连。该矩阵可以被用于其他分析。