

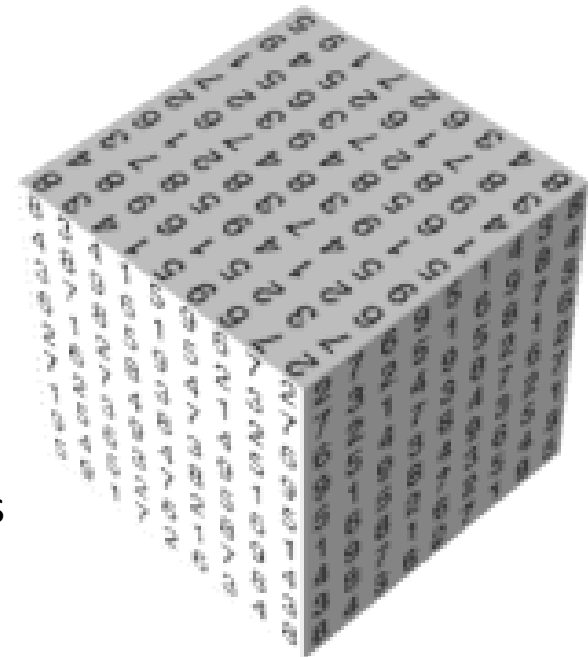


# Multi-Dimensional Matrices

Steve Borgatti

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

One  
multidimensional  
matrix, or layers of  
lower dimensional matrices

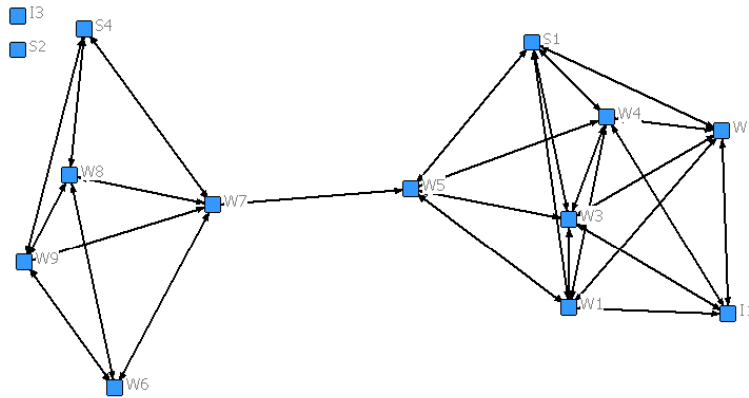


# 3-dimensional matrices

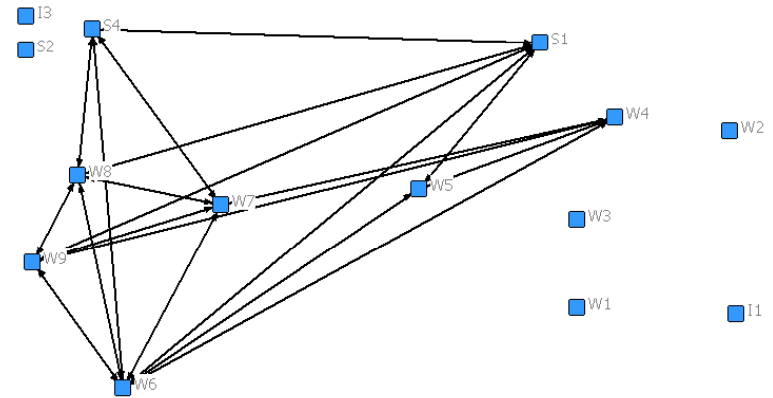
- 3-way matrix  $X$  with cells  $x_{ijk}$ . By convention, first subscript is rows, second subscript is columns, third subscript is levels
- 3-way, 3-mode: rows, cols & levels all different entities
  - Person by prod over time
- 3-way, 2-mode
  - Person by person by time
  - Person by person by relation

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

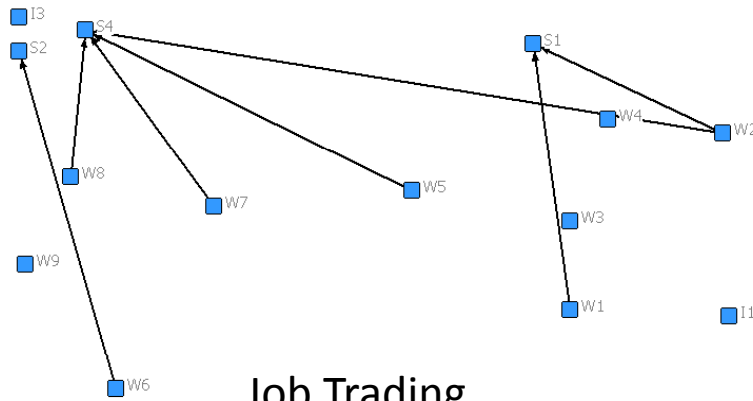
# Bank Wiring Room data



Games



Conflict



Job Trading

- Multiple social relations measured for the same group of actors
- 3-way, 2-mode matrices
  - Multiple 1-mode matrices

# File formats

- VNA
- DL

```

*Node data
"ID"
"I1"
"I3"
"W1"
"W2"
"W3"
"W4"
"W5"
"W6"
"W7"
"W8"
"W9"
"S1"
"S2"
"S4"
*Tie data
FROM TO
"RDGAM", "RDCON", "RDPOS", "RDNEG", "RDHLP", "RDJOB"
"I1", "W1", 1, 0, 0, 0, 0, 0
"I1", "W2", 1, 0, 0, 1, 0, 0
"I1", "W3", 1, 0, 1, 0, 0, 0
"I1", "W4", 1, 0, 0, 0, 0, 0
"W1", "I1", 1, 0, 0, 0, 0, 0
"W1", "W2", 1, 0, 0, 0, 0, 0
"W1", "W3", 1, 0, 1, 0, 1, 0
"W1", "W4", 1, 0, 1, 0, 0, 0
"W1", "W5", 1, 0, 0, 0, 0, 0
"W1", "S1", 1, 0, 1, 0, 1, 2

```

## VNA file with multiple relations

- For each ordered pair of nodes (rows) the columns indicate whether the pair have a given relation

# DL nodelist1 format

- Each relation separated by !  
mark

```
DL N=14 NM=6 FORMAT = NODELIST1
LABELS EMBEDDED
MATRIX LABELS EMBEDDED
DATA:
"RDGAM"
"I1" "W1" "W2" "W3" "W4"
"I3"
"W1" "I1" "W2" "W3" "W4" "W5" "S1"
"W2" "I1" "W1" "W3" "W4" "S1"
"W3" "I1" "W1" "W2" "W4" "W5" "S1"
"W4" "I1" "W1" "W2" "W3" "W5" "S1"
"W5" "W1" "W3" "W4" "W7" "S1"
"W6" "W7" "W8" "W9"
"W7" "W5" "W6" "W8" "W9" "S4"
"W8" "W6" "W7" "W9" "S4"
"W9" "W6" "W7" "W8" "S4"
"S1" "W1" "W2" "W3" "W4" "W5"
"S2"
"S4" "W7" "W8" "W9"
!
"RDCON"
"I1"
"I3"
"W1"
"W2"
"W3"
"W4" "W5" "W6" "W7" "W9"
"W5" "W4" "W6" "S1"
```

# DL edgelist1 format

- Just like nodelist1  
format, the relations are  
separated by “!” marks

```
DL N=14 NM=6 FORMAT = EDGELIST1
LABELS EMBEDDED
MATRIX LABELS EMBEDDED
DATA:
RDGAM
I1 W1 1
I1 W2 1
I1 W3 1
I1 W4 1
W1 I1 1
W1 W2 1
W1 W3 1
...
!
RDCON
W4 W5 1
W4 W6 1
W4 W7 1
...
!
RDNEG
...
```

# DL fullmatrix format

- No “!” marks needed since the number of numbers to be read is fixed and calculable in advance

DL N=14 NM=6 FORMAT = FULLMATRIX

LABELS EMBEDDED

MATRIX LABELS EMBEDDED

DATA:

"RDGAM"

"I1" "I3" "W1" "W2" "W3" "W4" "W5" "W6" "W7" "W8" "W9" "S1" "S2" "S4"

"I1" 0 0 1 1 1 1 0 0 0 0 0 0 0 0

"I3" 0 0 0 0 0 0 0 0 0 0 0 0 0 0

"W1" 1 0 0 1 1 1 1 0 0 0 0 1 0 0

"W2" 1 0 1 0 1 1 0 0 0 0 0 1 0 0

"W3" 1 0 1 1 0 1 1 0 0 0 0 1 0 0

"W4" 1 0 1 1 1 0 1 0 0 0 0 1 0 0

"W5" 0 0 1 0 1 1 0 0 1 0 0 1 0 0

"W6" 0 0 0 0 0 0 0 0 1 1 1 0 0 0

"W7" 0 0 0 0 0 0 1 1 0 1 1 0 0 1

"W8" 0 0 0 0 0 0 0 1 1 0 1 0 0 1

"W9" 0 0 0 0 0 0 0 1 1 1 0 0 0 1

"S1" 0 0 1 1 1 1 1 0 0 0 0 0 0 0

"S2" 0 0 0 0 0 0 0 0 0 0 0 0 0 0

"S4" 0 0 0 0 0 0 0 0 1 1 1 0 0 0

"RDCON"

"I1" "I3" "W1" "W2" "W3" "W4" "W5" "W6" "W7" "W8" "W9" "S1" "S2" "S4"

"I1" 0 0 0 0 0 0 0 0 0 0 0 0 0 0

"I3" 0 0 0 0 0 0 0 0 0 0 0 0 0 0

"W1" 0 0 0 0 0 0 0 0 0 0 0 0 0 0

"W2" 0 0 0 0 0 0 0 0 0 0 0 0 0 0

"W3" 0 0 0 0 0 0 0 0 0 0 0 0 0 0

"W4" 0 0 0 0 0 0 1 1 1 0 1 0 0 0

"W5" 0 0 0 0 0 1 0 1 0 0 0 1 0 0

"W6" 0 0 0 0 0 1 1 0 1 1 1 1 0 1

"W7" 0 0 0 0 0 1 0 1 0 1 1 0 0 1

"W8" 0 0 0 0 0 0 0 1 1 0 1 1 0 1

"W9" 0 0 0 0 0 1 0 1 1 1 0 1 0 0

"S1" 0 0 0 0 0 0 1 1 0 1 1 0 0 1

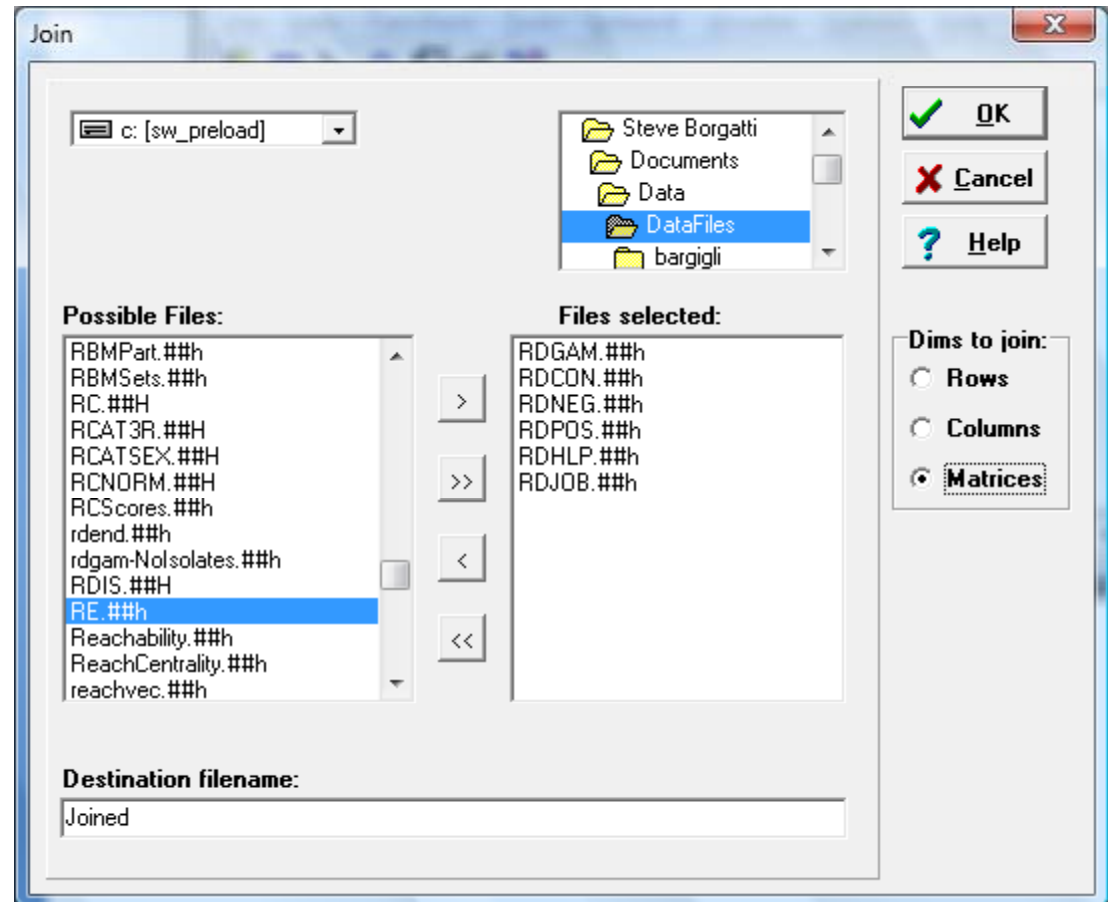
"S2" 0 0 0 0 0 0 0 0 0 0 0 0 0 0

"S4" 0 0 0 0 0 0 0 1 1 1 0 1 0 0

...

# Packing and Unpacking

- UCINET data | unpack command can create separate files for each relation in a 3-dimensional dataset
- UCINET data | join can combine separate 2D files (of the same size) into one 3D dataset
  - Make sure to pick “matrices” as the dimension

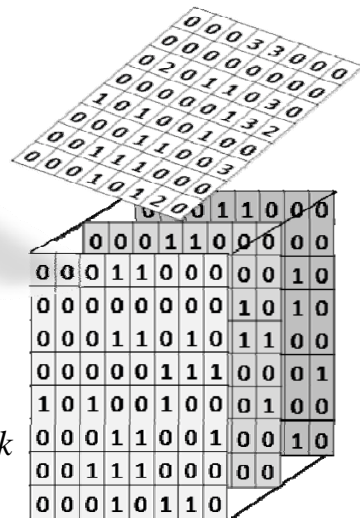
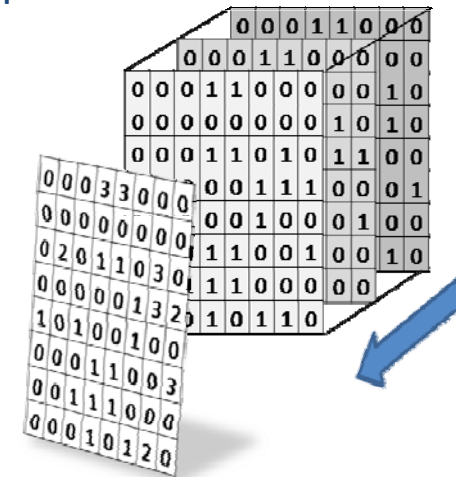


# Aggregations

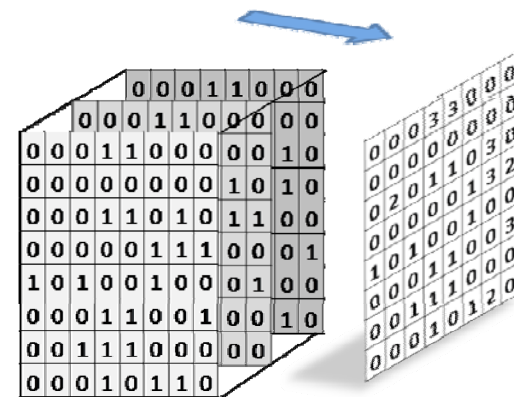
Summing across one dimension

$$y_{ij} = \sum_k x_{ijk}$$

- Summing across levels, creating 2D matrix from 3D matrix
  - E.g., summing over relations to get total number ties binding  $i$  to  $j$
  - E.g., summing over time to get total number of collaborations between  $i$  and  $j$  over all time



$$y_{jk} = \sum_i x_{ijk}$$



$$y_{ik} = \sum_j x_{ijk}$$

- Summing across rows, creating 2D horizontal "slice"
  - E.g., summing over all egos to get total number ties that alter  $j$  is receiving at time  $k$  (indegree over time)

- Summing across columns, creating 2D vertical "slice"
  - E.g., summing over all alters to get total number ties that  $i$  is sending at time  $k$  (outdegree over time)







# Application

(using Wiring dataset)

- Measuring extent of multiplexity
  - For a given pair of actors, how many different kinds of ties bind them together?
  - Sum across all the relations to obtain a single aggregate matrix  $M$  in which  $m_{ij}$  = number of different relations that connect  $i$  to  $j$
- In matrix algebra:
  - $\rightarrow M = \text{total}(\text{wiring rows cols})$
- From menus:
  - Transform | matrix ops | within datasets | aggregations

# Bundles of relations

Marriage ties

```

0000000010000000
0000011010000000
0000100010000000
0000001000100010
0010000000100010
0100000000000000
0101000100000001
0000001000000000
1110000000001101
0000000000000100
0001100000000010
0000000000000000
0000000010000011
0000000011000000
0001100000101000
0000001010001000
    
```

Business ties

```

0000000000000000
0000000000000000
0000110010100000
0000001100100000
0010000100100000
0010000010000000
0001000100000000
0001101000100000
0010010001000101
0000000010000000
0011100100000000
0000000000000000
0000000000000000
0000000000000000
0000000010000000
0000000000000000
0000000010000000
    
```

Multiplex

```

0000000020000000
0000022020000000
0000310030100000
0000003100300020
0030000100300020
0210000010000000
0203000300000002
0001103000100000
2230010001002303
0000000010000200
0013300100000020
0000000000000000
0000000020000022
0000000032000000
0002200000202000
0000002030002000
    
```

Padgett Dataset

## Legend

- 0 = no tie on either relation
- 1 = business tie only
- 2 = marriage tie only
- 3 = both business and marriage tie

# Correlating slices of 3-dimensional matrices

- UCINET commands
  - For slices contained in a single 3D object
    - Tools | similarity > select matrices as dimension to correlation
  - For slices stored as separate matrices:
    - Tools | testing hyps | dyadic | gap correlation

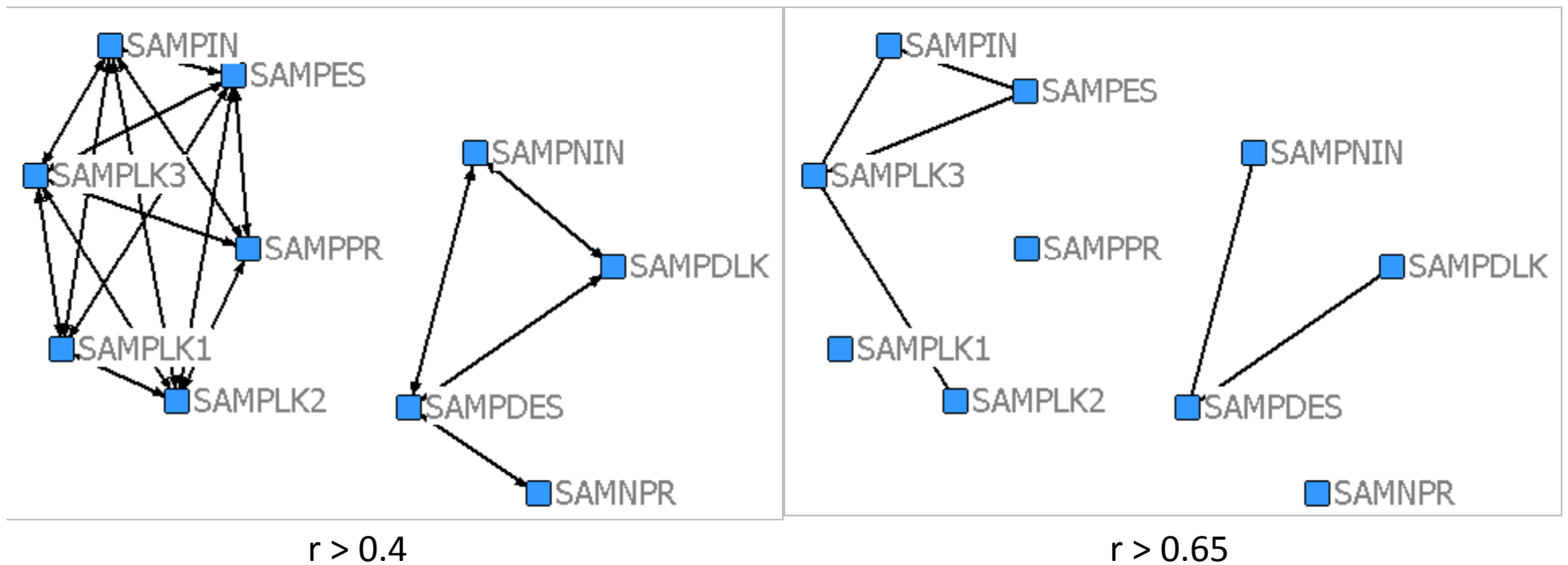
- Wiring example

	RDGAM	RDCON	RDPOS	RDNEG	RDHLP	RDJOB
RDGAM	1.000	0.243	0.544	-0.050	0.374	0.136
RDCON	0.243	1.000	0.099	0.002	0.159	0.087
RDPOS	0.544	0.099	1.000	-0.210	0.259	0.129
RDNEG	-0.050	0.002	-0.210	1.000	-0.160	-0.076
RDHLP	0.374	0.159	0.259	-0.160	1.000	0.085
RDJOB	0.136	0.087	0.129	-0.076	0.085	1.000

# Visualizing correlations among relations

- Open the correlation matrix in netdraw as if it were a relation
  - Adjust strength of tie criterion to show only strong ties

Sampson Monastery Dataset



# Transposing multiple dimensions

Description	Math	UCINET Matrix Algebra syntax
Interchange rows with columns for all levels - invert direction of ties	$y_{ijk} = x_{jik}$	CRL = transp(wiring row col)
Interchange levels with columns for all levels	$y_{ijk} = x_{ikj}$	RLC = transp(wiring lev col)
Interchange levels with rows for all levels	$y_{ijk} = x_{kji}$	LCR = transp(wiring lev row)
Interchange rows with cols then levs with (new) cols	$y_{ijk} = x_{jki}$	CRL = transp(wiring row col) CLR= transp(CRL col lev)
Interchange rows with cols then levs with (new) rows	$y_{ijk} = x_{kij}$	CRL = transp(wiring row col) LRC= transp(CRL row lev)

# Transposing

13	14	15	16
17	18	19	20
21	22	23	24
1	2	3	4
5	6	7	8
9	10	11	12

X

3 rows, 4 cols, 2 levs

transpose



13	17	21
14	18	22
15	19	23
16	20	24
1	5	9
2	6	10
3	7	11
4	8	12

Transp(X row col)

4r, 3c, 2l

# Transposing

13	14	15	16
17	18	19	20
21	22	23	24

1	2	3	4
5	6	7	8
9	10	11	12

Original matrix X  
3 rows, 4 cols, 2 levs

	M1	M2
R1	1	13
R2	5	17
R3	9	21

	M1	M2
R1	2	14
R2	6	18
R3	10	22

	M1	M2
R1	3	15
R2	7	19
R3	11	23

	M1	M2
R1	4	16
R2	8	20
R3	12	24

Transp(X col lev)  
3r, 2c, 4l

	C1	C2	C3	C4
M1	1	2	3	4
M2	13	14	15	16

	C1	C2	C3	C4
M1	5	6	7	8
M2	17	18	19	20

	C1	C2	C3	C4
M1	9	10	11	12
M2	21	22	23	24

Transp(X row lev)  
2r, 4c, 3l

# Transposing - cont

	M1	M2
C1	1	13
C2	2	14
C3	3	15
C4	4	16

	M1	M2
C1	5	17
C2	6	18
C3	7	19
C4	8	20

	M1	M2
C1	9	21
C2	10	22
C3	11	23
C4	12	24

	R1	R2	R3
M1	1	5	9
M2	13	17	21

	R1	R2	R3
M1	2	6	10
M2	14	18	22

	R1	R2	R3
M1	3	7	11
M2	15	19	23

	R1	R2	R3
M1	4	8	12
M2	16	20	24

$\text{Transp}(\text{transp}(X \text{ row col}) \text{ col lev})$

$\text{Transp}(\text{transp}(X \text{ row col}) \text{ row lev})$

# Multiplication table for all possible combinations of transpositions

	i	rc	rl	cl	rc*rl	rc*cl
i	i	rc	rl	cl	rc*rl	rc*cl
rc	rc	i	rc*rl	rc*cl	rl	cl
rl	rl	rc*cl	i	rc*rl	cl	rc
cl	cl	rc*rl	rc*cl	i	rc	rl
rc*rl	rc*rl	cl	rc	rl	i	i
rc*cl	rc*cl	rl	cl	rc	i	i



# Relational Composition

- Compositions of relations, such as a friend of a friend ( $F \circ F$ ) or enemy of a friend ( $F \circ E$ ) can be computed by boolean matrix multiplication

$$FE = F \times E$$

	Mary	Bill	John	Larry
Mary	0	1	0	1
Bill	1	0	1	0
John	0	0	0	1
Larry	0	0	0	0

F

	Mary	Bill	John	Larry
Mary	0	0	1	1
Bill	1	0	1	0
John	0	0	0	1
Larry	0	1	0	0

E

	Mary	Bill	John	Larry
Mary	1	1	1	0
Bill	0	0	0	1
John	0	1	0	0
Larry	0	0	0	0

FE

# Relational composition example (wiring dataset)

- Matrices
  - P = positive affect (rdpos)
  - C = conflict with (rdcon)
  - Construct matrix PC
- Interpretation
  - $iPC_j$  (i.e, cell  $(pc)_{ij} > 0$  means that person  $i$  has positive feeling toward someone who has a conflict with  $j$
- Uses
  - Predict that  $i$  will have negative affect toward  $j$
  - Outdegree on PC relation predicts lower morale

# Another relational composition example

- Matrices
  - $F$  = friends with
  - $W$  = works with
  - $FW$  = works with a friend of
    - $fw_{ij} > 0$  means that  $j$  works with a friend of  $i$ 's ( $i$  has a friend who works with  $j$ )
    - $Fw_{ij} > 0$  means  $j$  is an indirect source of information for  $i$
- Interpretation
  - High degree centrality on  $FW$  relation should mean the person has lots of access to organizational information

# An Improvement ...

- Matrices
  - $F$  = friends with
  - $W$  = works with
  - $FW$  = works with a friend of
    - $fw_{ij} > 0$  means that  $j$  works with a friend of  $i$ 's ( $i$  has a friend who works with  $j$ )
    - $Fw_{ij} > 0$  means  $j$  is an indirect source of information for  $i$
  - $X = F+FW$  = is a friend of or works with a friend of
    - $x_{ij} > 0$  means  $j$  is either a friend of  $i$  or works with a friend of  $i$
- Application
  - High degree centrality on  $X$  relation should mean the person has lots of access to organizational information

# Running analyses with multiple relations

- Two types
  - Automatically repeating analysis on each relation in turn
    - Works on many but not all ucinet procedures
  - Taking all relations into account simultaneously
    - Only for role analysis / blockmodeling techniques

# Automatically Repeating Analyses

- Example: eigenvector centrality on Wiring
  - Output is 3-dimensional matrix in which each matrix slice gives centralities for one relation

Matrix #1: RDGAM

		Eigenv	nEigen
		-----	-----
1	I1	0.307	43.368
2	I3	0.000	0.000
3	W1	0.417	58.960
4	W2	0.365	51.669
5	W3	0.417	58.960
6	W4	0.417	58.960
7	W5	0.323	45.718
8	W6	0.029	4.070
9	W7	0.085	12.011
10	W8	0.033	4.719
11	W9	0.033	4.719
12	S1	0.368	52.043
13	S2	0.000	0.000
14	S4	0.029	4.070

Matrix #2: RDCON

		Eigenv	nEigen
		-----	-----
1	I1	0.000	0.000
2	I3	0.000	0.000
3	W1	0.000	0.000
4	W2	0.000	0.000
5	W3	0.000	0.000
6	W4	0.290	40.946
7	W5	0.224	31.719
8	W6	0.466	65.942
9	W7	0.369	52.170
10	W8	0.380	53.742
11	W9	0.375	53.090
12	S1	0.356	50.308
13	S2	0.000	-0.000
14	S4	0.317	44.828

< SNIP >

# Displaying results for each relation side by side

- Having just run eigenvector centrality (which by default creates dataset “eigenvectorcentrality” as output) In matrix algebra:
  - > dsp transp(eigenvectorcentrality col lev)

		RDGAM	RDCON	RDPOS	RDNEG	RDHLP	RDJOB
		-----	-----	-----	-----	-----	-----
1	I1	0.307	0.000	0.147	0.173	0.000	0.000
2	I3	0.000	0.000	-0.000	0.474	0.000	0.000
3	W1	0.417	0.000	0.438	0.000	-0.291	-0.009
4	W2	0.365	0.000	-0.000	0.261	-0.214	-0.185
5	W3	0.417	0.000	0.472	0.000	-0.328	0.000
6	W4	0.417	0.290	0.438	0.114	-0.374	0.000
7	W5	0.323	0.224	0.000	0.485	-0.078	-0.224
8	W6	0.029	0.466	-0.000	0.310	-0.417	0.000
9	W7	0.085	0.369	0.254	0.360	-0.298	-0.064
10	W8	0.033	0.380	0.159	0.287	-0.323	-0.641
11	W9	0.033	0.375	0.159	0.287	-0.322	0.000
12	S1	0.368	0.356	0.498	0.114	-0.192	-0.102
13	S2	0.000	-0.000	0.000	0.114	-0.100	0.000
14	S4	0.029	0.317	0.099	0.112	-0.315	-0.700