Oct. 19

1. Administrative
2. Roles as relations (blockmodels)
3. Structural and regular equivalence
4. Discovering blocks algorithmically

Administrative

- Précis feedback this week
- Proposal due Nov 9

Roles as Relations

Roles as relations

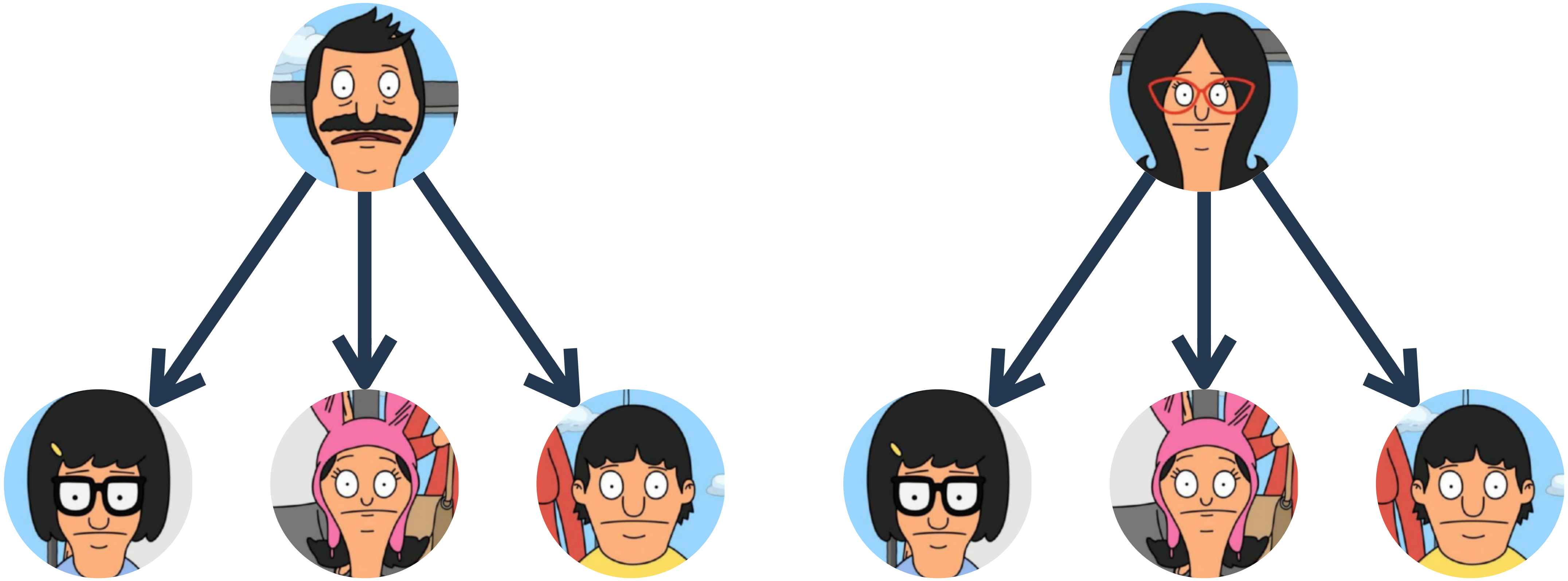
Sociology of roles

> **Roles are bundles of expectations in interaction**
> - the role of *parent* might be defined by expectations of caregiving toward certain children
> - the role of *child* might be defined (in part) by expectations of obedience and dependence toward parents
> - the role of *boxer* is defined by expectations of physical violence toward other boxers and deference toward a coach

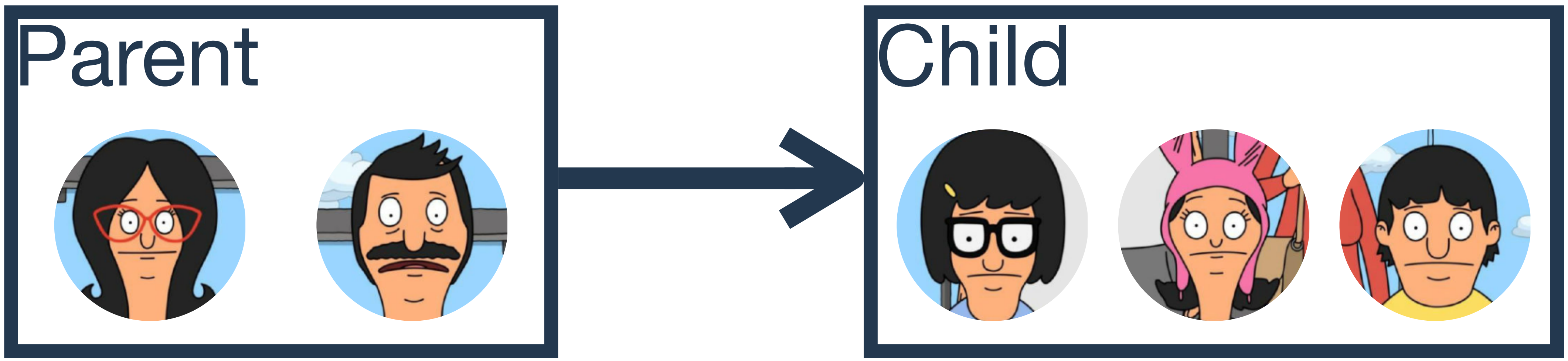
Roles are about *relations between categories*—perfect for network analysis!

Roles as relations

Network relations



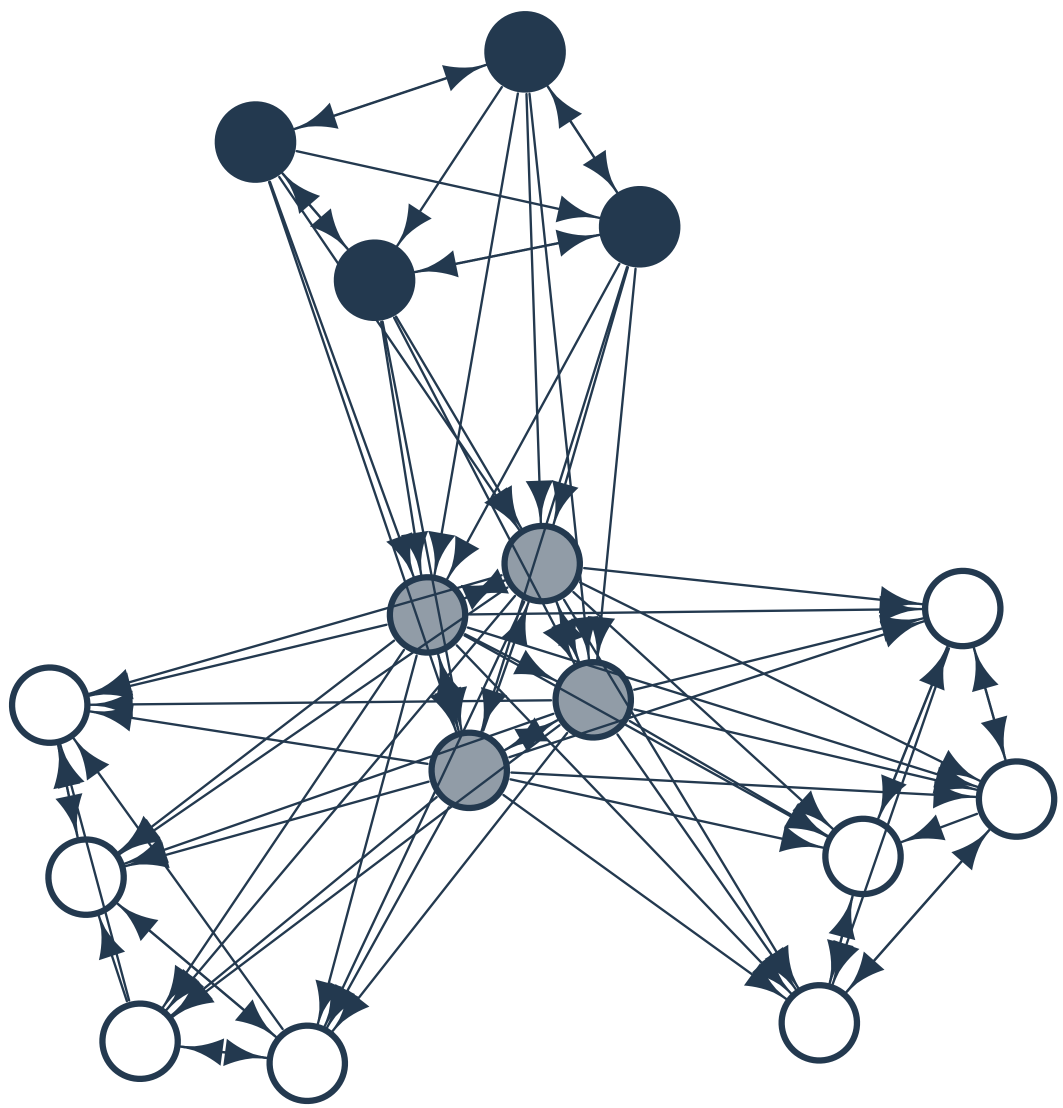
Role relations



Roles as relations

Blockmodelling **aims to formalize this intuition**

- Somewhat vague term, refers to methods, models, and theories that focus on the relational nature of roles
- Find categories of actors in “equivalent” positions in a network



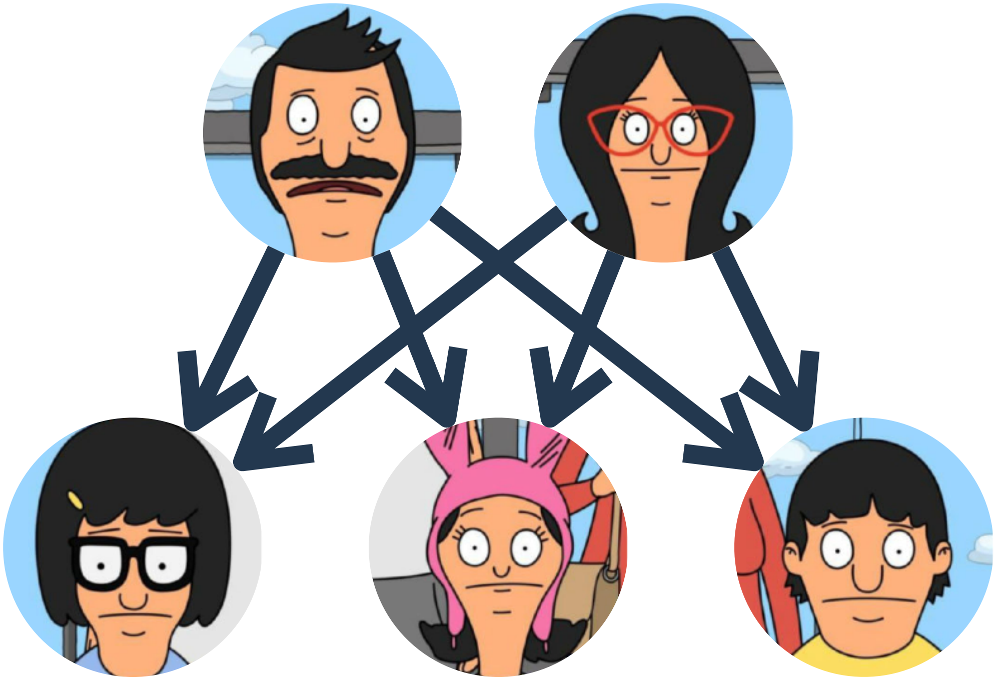
Structural & Regular Equivalence

Equivalence

Two major forms of "equivalence" of network position: *structural* and *regular*

Structural equivalence

- Two actors are *structurally equivalent* if they have the same ties to the same set of actors
- E.g. Bob and Linda Belcher are structurally equivalent in their role as "caregiver of Tina, Louise, and Gene"
- In a *sociogram*, this means swapping labels does not change the network
- In an *adjacency matrix*, this means having identical rows and columns



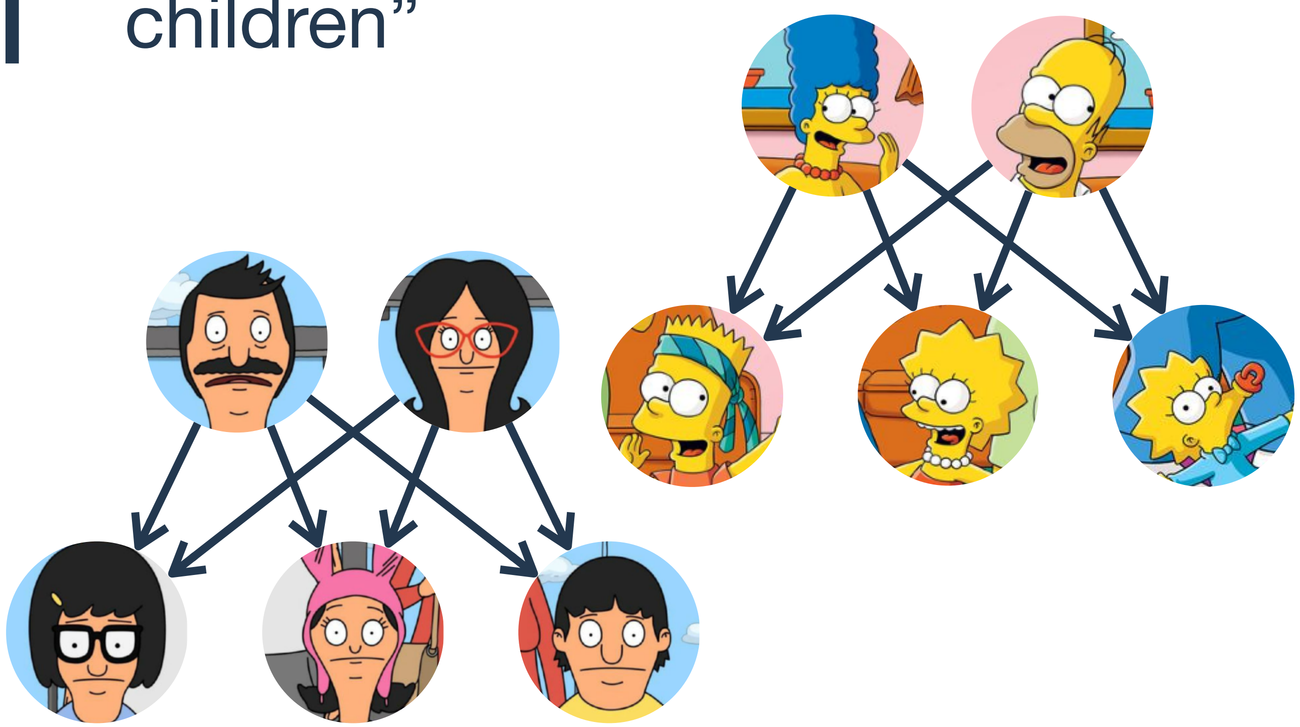
	1	2	3	4	5
Bob 1	0	0	1	1	1
Linda 2	0	0	1	1	1
Tina 3	0	0	0	0	0
Louise 4	0	0	0	0	0
Gene 5	0	0	0	0	0

Equivalence

Two major forms of "equivalence" of network position: *structural* and *regular*

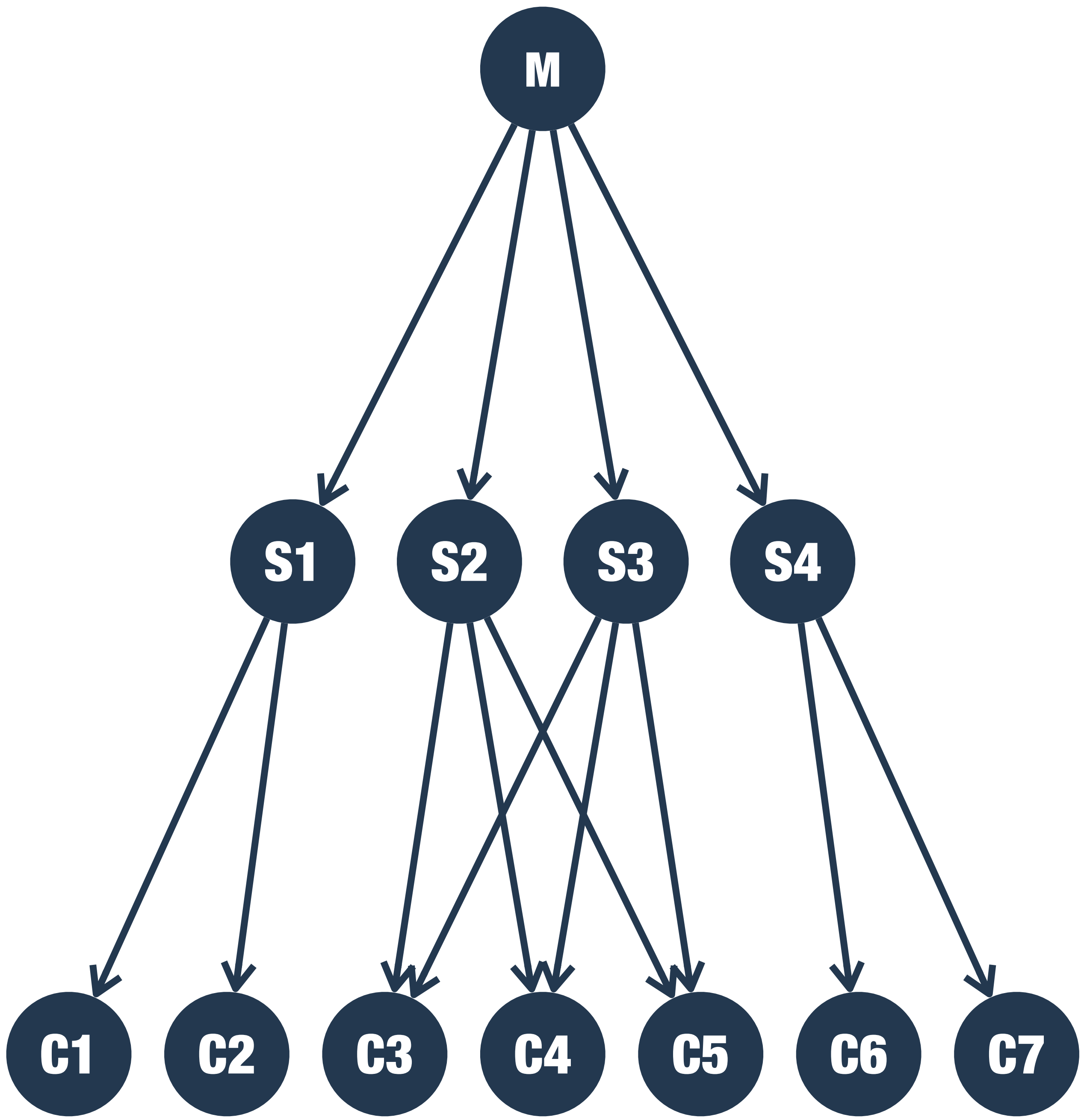
Regular equivalence

- Two actors are *regularly equivalent* if they have ties to the same *type* of actors
- E.g. Linda Belcher and Marge Simpson are regularly equivalent in their role as "caregiver of children"



		1	2	3	4	5	6	7	8	9	10
Bob	**1**	0	0	0	0	**1**	**1**	**1**	0	0	0
Linda	**2**	0	0	0	0	**1**	**1**	**1**	0	0	0
Marge	**3**	0	0	0	0	0	0	0	**1**	**1**	**1**
Homer	**4**	0	0	0	0	0	0	0	**1**	**1**	**1**
Tina	**5**	0	0	0	0	0	0	0	0	0	0
Louise	**6**	0	0	0	0	0	0	0	0	0	0
Gene	**7**	0	0	0	0	0	0	0	0	0	0
Bart	**8**	0	0	0	0	0	0	0	0	0	0
Lisa	**9**	0	0	0	0	0	0	0	0	0	0
Maggie	**10**	0	0	0	0	0	0	0	0	0	0

Equivalence



A restaurant with seven customers (C), four servers (S), and a floor manager (M)

Comparing *structural* and *regular* equivalence

Structural:

- C1 and C2 (both served by S1)
- S2 and S3 (both serving C3, C4, C5 and reporting to M)
- *Not* S1 and S2 (serving different customers

Regular:

- C1–C7 (all served by servers
- S1–S4 (all serving customers and reporting to a manager)
- *“Customer” and “server” are mutually dependent categories*

Example

Example

West Side Story

Jets vs Sharks

Friendship	R	T	I	A	B	C	P
Riff	•	1	1	1	•	•	•
Tony	1	•	1	1	•	•	•
Ice	1	1	•	1	•	•	•
Action	1	1	1	•	•	•	•
Bernardo	•	•	•	•	•	1	1
Chino	•	•	•	•	1	•	1
Pepe	•	•	•	•	1	1	•

Rivalry	R	T	I	A	B	C	P
Riff	•	•	•	•	1	1	1
Tony	•	•	•	•	1	1	1
Ice	•	•	•	•	1	1	1
Action	•	•	•	•	1	1	1
Bernardo	1	1	1	1	•	•	•
Chino	1	1	1	1	•	•	•
Pepe	1	1	1	1	•	•	•

Example

Friendship	R	T	I	A	B	C	P
Riff	0	1	1	1	0	0	0
Tony	1	0	1	1	0	0	0
Ice	1	1	0	1	0	0	0
Action	1	1	1	0	0	0	0
Bernardo	0	0	0	0	0	1	1
Chino	0	0	0	0	1	0	1
Pepe	0	0	0	0	1	1	0

→

	J	S
J	1	0
S	0	1

Example

Jets, Jet Girls, Sharks, and Shark Girls

		1	2	3	4	5	6	7	8	9	10	11
Riff	**1**		1	1	1	1						
Tony	**2**	1		1	1							
Ice	**3**	1	1		1		1					
Action	**4**	1	1	1								
Velma	**5**	1					1					
Graziella	**6**			1		1						
Bernardo	**7**								1	1		1
Chino	**8**							1		1		
Pepe	**9**							1	1		1	
Consuelo	**10**									1		1
Anita	**11**							1			1	

Example

Jets, Jet Girls, Sharks, and Shark Girls

<table>
<tr><th></th><th></th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th></tr>
<tr><td>Riff</td><td>1</td><td colspan="4" rowspan="4">Complete</td><td colspan="2" rowspan="4">Col. Reg.</td><td colspan="3" rowspan="4">Null</td><td colspan="2" rowspan="4">Null</td></tr>
<tr><td>Tony</td><td>2</td></tr>
<tr><td>Ice</td><td>3</td></tr>
<tr><td>Action</td><td>4</td></tr>
<tr><td>Velma</td><td>5</td><td colspan="4" rowspan="2">Row Regular</td><td colspan="2" rowspan="2">Comp.</td><td colspan="3" rowspan="2">Null</td><td colspan="2" rowspan="2">Null</td></tr>
<tr><td>Graziella</td><td>6</td></tr>
<tr><td>Bernardo</td><td>7</td><td colspan="4" rowspan="3">Null</td><td colspan="2" rowspan="3">Null</td><td colspan="3" rowspan="3">Complete</td><td colspan="2" rowspan="3">Col. Reg.</td></tr>
<tr><td>Chino</td><td>8</td></tr>
<tr><td>Pepe</td><td>9</td></tr>
<tr><td>Consuelo</td><td>10</td><td colspan="4" rowspan="2">Null</td><td colspan="2" rowspan="2">Null</td><td colspan="3" rowspan="2">Row Regular</td><td colspan="2" rowspan="2">Comp.</td></tr>
<tr><td>Anita</td><td>11</td></tr>
</table>

Example

Null-block–crossed lovers

		1	2	3	4	5	6	7	8	9	10	11	12
Riff	**1**		1	1	1	1							
Tony	**2**	1		1	1								1
Ice	**3**	1	1		1		1						
Action	**4**	1	1	1									
Velma	**5**	1					1						
Graziella	**6**			1		1							
Bernardo	**7**								1	1		1	
Chino	**8**							1		1			
Pepe	**9**							1	1		1		
Consuelo	**10**									1		1	1
Anita	**11**							1			1		1
Maria	**12**		1								1	1	

Discovering Blocks Algorithmically

Discovering block structure

Normally, network data dœs not come pre-sorted

- *Block structure not apparent until rows and columns are re-ordered*

		1	2	3	4	5	6	7	8	9	10	11	12
Ice	**1**		1				1	1					1
Graziella	**2**	1									1		
Bernardo	**3**				1					1		1	
Chino	**4**			1						1			
Maria	**5**							1	1			1	
Riff	**6**	1						1			1		1
Tony	**7**	1				1	1						1
Consuelo	**8**					1				1		1	
Pepe	**9**			1	1				1				
Velma	**10**		1				1						
Anita	**11**			1		1			1				
Action	**12**	1					1	1					

Discovering block structure

There are two main approaches to *fitting* (a.k.a. *estimating*) block structure

Traditional blockmodelling

- Define which blocks are "allowed"
- Re-arrange rows and columns in the adjacency matrix until it (approximately) fits the pattern
- Effective way to look for expected patterns, e.g. a core–periphery structure
- Can take advantage of multiple relations on a group, (friend, enemy, authority, etc.)

Stochastic blockmodelling

- Assume that there is some number of *latent* blocks in a network
- Edges within and between blocks follow simple probabilistic patterns (e.g. "actors in block A have a 10% chance to connect to any actor in block B")
- Algorithms try to simultaneously discover the number of blocks, the membership of the blocks, and the edge probability between blocks

Discovering block structure

Stochastic block model (SBM)

<table>
<tr><th></th><th></th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th></tr>
<tr><td>Riff</td><td>1</td><td colspan="4" rowspan="4">1.0</td><td colspan="2" rowspan="4">0.25</td><td colspan="3" rowspan="4">0.0</td><td colspan="3" rowspan="4">0.08</td></tr>
<tr><td>Tony</td><td>2</td></tr>
<tr><td>Ice</td><td>3</td></tr>
<tr><td>Action</td><td>4</td></tr>
<tr><td>Velma</td><td>5</td><td colspan="4" rowspan="2">0.25</td><td colspan="2" rowspan="2">1.0</td><td colspan="3" rowspan="2">0.0</td><td colspan="3" rowspan="2">0.0</td></tr>
<tr><td>Graziella</td><td>6</td></tr>
<tr><td>Bernardo</td><td>7</td><td colspan="4" rowspan="3">0.0</td><td colspan="2" rowspan="3">0.0</td><td colspan="3" rowspan="3">1.0</td><td colspan="3" rowspan="3">0.22</td></tr>
<tr><td>Chino</td><td>8</td></tr>
<tr><td>Pepe</td><td>9</td></tr>
<tr><td>Consuelo</td><td>10</td><td colspan="4" rowspan="3">0.08</td><td colspan="2" rowspan="3">0.0</td><td colspan="3" rowspan="3">0.22</td><td colspan="3" rowspan="3">1.0</td></tr>
<tr><td>Anita</td><td>11</td></tr>
<tr><td>Maria</td><td>12</td></tr>
</table>