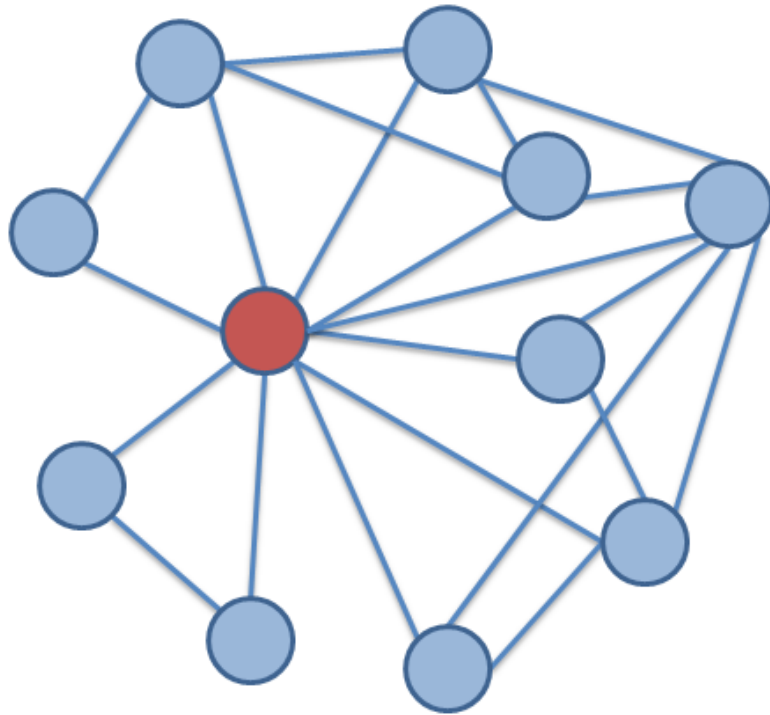


NetLogo Tutorial 2: Networks

Kevin J.S. Zollman
Carnegie Mellon University

Social Networks



- Nodes
 - Represent basic object, often agents
- Edges
 - Represent types of relationships

Edges

- Undirected
 - Imply a symmetric relationship
 - *Siblings, roommates, co-authors*
- Directed
 - Imply a non-symmetric relationship
 - *In love with, trusts, supervises*

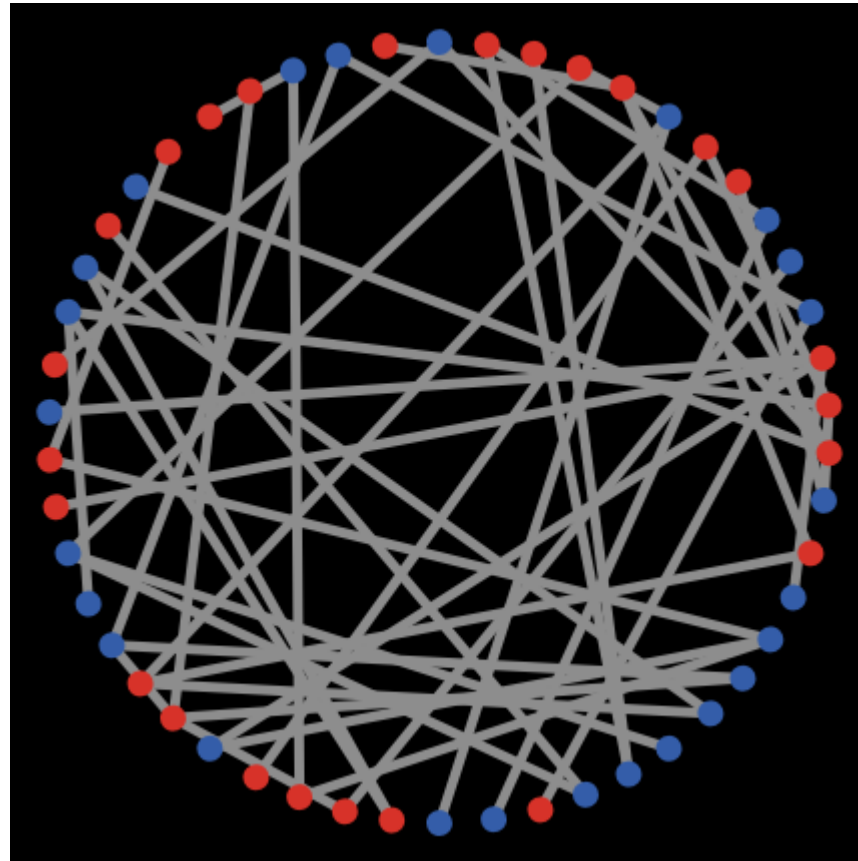
Edges

- Singular/multiple
 - *Loves / Shook hands*
- Labeled/unlabeled
 - *Knows / Facebook friends with*

Networks and game theory

- Nodes
 - Represent individuals who have strategies in a game
- Edges
 - Represent interactive pairs, who play the game
 - Also, the source of imitation

Stag Hunt



Setup function

Old

```
ask patches [
  sprout 1 [
    set payoff 0
    set strategy random 2
    set old-strategy strategy
    set shape "circle"
  ]
]
```

New

```
create-turtles NumberOfAgents [
  set payoff 0
  set strategy random 2 ;;
  set old-strategy strategy
  set shape "circle" ;;
]

layout-circle turtles 15
```

```
ask turtles [
  create-links-with n-of 1 other turtles
]

ask links [
  set thickness 0.4
]
```

Network formation

- This algorithm guarantees that everyone has at least one “friend”
- But, others might have more

Play/Imitate function

Old

```
;; These are the people with whom I am going to play  
let neighbor-patches [neighbors] of patch-here  
let players turtle-set [turtles-here] of neighbor-patches
```

New

```
;; These are the people with whom I am going to play  
let players link-neighbors
```

Effect of networks

- Creates “bottlenecks” that prevent the contagion process
- Multiplies the number of stable configurations

Two modifications

- Change the way the network is generated
- Change the method of learning and evaluating strategies

Two modifications

- Change the way the network is generated
- Change the method of learning and evaluating strategies

Other network algorithms

- More neighbors
- Random network
- Regular network
- Preferential attachment

Other network algorithms

- More neighbors
- Random network
- Regular network
- Preferential attachment

More neighbors

- Create a new slider *Average-Friends*
- Have each turtle connect to *Average-Friends* other turtles

Other network algorithms

- More neighbors
- Random network
- Regular network
- Preferential attachment

Random network

- Each edge is created with some probability
- The probability for each edge is independent of every other edge

Random network

Interface

```
Create slider called Edge-Probability
Have it range in value from 0 to 1, in
0.1 increments
```

Setup Function

```
Ask every turtle
  Create a link with every other
  turtle

Ask every edge
  Generate a random number, x, from
  [0,1)
  If  $x < (1 - \text{Edge-Probability})$ 
    Remove the link
  Otherwise
    Leave the link alone
```

Hints

```
all, ask,
create-links-
with, ifelse,
other, random-
float, turtles
```

Other network algorithms

- More neighbors
- Random network
- **Regular network**
- Preferential attachment

Regular network

- Every individual has exactly the same number of neighbors
- The ring is the simplest
- The grid used before was one as well

Modulus

```
n mod m
```

```
let x floor (n / m)  
n - x*m
```

Modulus

$0 \bmod 4, 1 \bmod 4, 2 \bmod 4, 3 \bmod 4,$
 $4 \bmod 4, 5 \bmod 4, 6 \bmod 4, 7 \bmod 4 \dots$

$0, 1, 2, 3,$
 $0, 1, 2, 3 \dots$

Regular network

Setup function

```
For each value of x in {0, 1, 2, ..., number of turtles}
  Ask turtle x to
    Let variable new-friend be equal to
      x mod number of turtles
    Form a link with turtle number new-friend
```

Hints

```
count, create-link-with, foreach, mod, n-values
```

Other network algorithms

- More neighbors
- Random network
- Regular network
- Preferential attachment

Preferential attachment

- *Gloss*: When people join a group, they tend to gravitate toward the already popular
- *Process*:
 - Two turtles are created
 - Each new turtle connects to one of the existing turtles at random
 - The probability that the new turtle connects to existing turtle t is proportional to the number of neighbors that t has

Preferential attachment

Header

```
Create new turtle variable called score
```

Hints

```
random, repeat,  
turtles-own
```

Setup function

```
Create two turtles  
Form a link between them  
Repeat n - 2 times  
  Assign each turtle a score: 1 + number of neighbors  
  Create new turtle  
    Let x be a random integer in {0, ... , sum of  
      turtle scores}  
  Ask other turtles  
    If x < number of neighbors  
      Form a link between the first turtle and  
        this one  
    Else  
      Let x equal (x - number of neighbors)
```

Two modifications

- Change the way the network is generated
- Change the method of learning and evaluating strategies

How payoff is calculated

- Currently, in the model each agent imitates her neighbor who has a higher average payoff
- What if they imitated the other who had the highest total payoff?
 - Can you guess what the effect would be?
- Find this in the code and change it

Imitation and interaction neighborhood

- Currently in the model the agents play with and imitate from the same group, their link-neighbors
- What if we changed that?
 - Imitate second-neighbors, but play with first neighbors
 - Have totally different neighborhoods

Imitate more widely

- Modify the imitate function to have agent's imitate a wider neighborhood
 - Their neighbors
 - And, their neighbors' neighbors
- Hints: `of`, `turtle-set`

Different imitation network

- More generally we could create two different networks, one for play and one for imitation
- This utilizes link “breeds”
 - One can define `link-breeds` in the header
 - Then the networks are created by `create-<breed>-with` in the setup function
- The two functions `play` and `imitate` would then utilize different networks using `<breed>-neighbors`

Best response

- “imitation” is not particularly sophisticated
- Myopic best response
 - On round t adopt what strategy would have been best on round $t - 1$

Different learning rule

New function "best-respond"

Ask turtles

Let stag-payoff be the payoff obtained from playing
stag against my neighbors' old-strategies

Let hare-payoff be the payoff obtained from playing
hare against my neighbors old-strategies

If stag-payoff > hare-payoff
Set strategy to stag

If hare-payoff > stag-payoff
Set strategy to hare

If stag-payoff = hare-payoff
Leave strategy the same

Step function

Replace imitate with best-respond