Simulation & Agent Based Models
For Social Networks & Health
Outline

1. Introduction:
   1. What, why & types

2. Simulation Elements
   1. Environment, Agents & rules
   2. Runtime meta-rules

3. Types of Network Simulations
   1. ...of networks, on networks or both?

4. Case Studies
   1. Simulations of networks
   2. Simulations on networks
   3. Simulation of AND on networks

5. Pragmatic Advice/Implementation pointers

6. Conclusion
Introduction
What is a simulation?

Many times SN&H scholars will simulate – make up – data. This is the one time that making up data is a good thing. 😊

Simulations run a gamut from simple randomizations through predictions from models to full-blown agent based models.

Fundamentally the goal is to make up data in a way that answers some question that cannot be answered well otherwise.
Introduction

Why simulation?

• Lack of data
  • Don’t observe the full network or transmission events

• Unethical Treatment effects
  • Who should we prioritize for vaccines? What effect would targeted vs. random have?

• Extending theory
  • Complex/unexpected emergent properties
  • Push unobserved extremes
  • Disentangle otherwise conflated processes

• Methods input
  • Statistical validation
    • Are estimators unbiased?
  • Sample construction/design
    • How do we optimize a field deployment?
Introduction

Types of simulations

• Purpose: Prediction or illustration
  *Why do you want to simulate?* Is it to guide policy in a way that can’t be done with statistical models (for whatever reason) or is it to understand a theory that is not amenable to observational tests?
  Example: Diffusion model of a city to allocate healthcare resources or diffusion model of sexual network to understand role of concurrency?

• Grounding: data vs. theory
  *How much does the question depend on observed facts or real situations?*
  Example: for peer influence model, do you want to control the network structure or use extant data?

• Realism: Single dimension vs. complex detail
  *How much of the social situation do you include?* What are the lowest level of detail you need to include?
Introduction

Types of simulations

- Schelling Segregation model
- SIS/SIR demonstration models
- Traffic simulation

Realism/Prediction

- CoreSim (SSA prediction models)
- SIENA simulations
- Covid-19 prediction models

Theory Based

Data Based

Illustration/Toy

Comparing diffusion mechanism on real data
Introduction
Most common SN&H Network simulation

Going to focus here on agent based network diffusion simulations with an eye toward core theory problems grounded in data.

→ users can then extend beyond this as suited to their problem.
Simulation Elements

Setup: Elements

Agent Based Models have three main elements:

**Environment:** The setting within which agents act. This usually consists of features that your agents draw on, including other agents.
- examples:
  - N x N grid of potential places to move in a Schelling model
  - Space of resources in the “sugarscape” model
  - neighborhood block groups in a disease sim

**Agents:** The actors in your model. Agents are active, they draw on resources from the environment and react to other agents. Agents can have multiple characteristics, fixed or variable.

**Rules:** The rules encode your theory of how actors behave and respond to situations. The set of rules you write should encapsulate the social process you are trying to mimic. Subtle differences in rules can matter.
- a) *agent rules:* rules governing how agents respond to situation.
- b) *global processing rules.* Rules governing how the simulation proceeds.
Simulation Elements

Setup: Elements

Example Schelling segregation model

Environment: 50 x 50 grid of spaces where agents can live. Each space is occupied (blue, orange) or open (white). Space is a torus (top cells are adjacent to bottom cells, right to left). 2466 agents

Agents: “people” with preferences for the color of their neighbors.

Rules:

agent rule: if fewer than $\alpha\%$ of your neighbors are same color as you, move to open spot. $\alpha=30$.

Global processing rules:
• agents selected at random
• updates made immediately
• stop on user input or when no more moves can be made
Simulation Elements

Setup: Elements

Example Schelling segregation model

Environment: 50 x 50 grid of spaces where agents can live. Each space is occupied (blue, orange) or open (white). Space is a torus (top cells are adjacent to bottom cells, right to left). 2092 agents

Agents: people with preferences for the color of their neighbors.

Rules:

agent rule: if fewer than α% of your neighbors are same color as you, move to open spot. α=30.

Global processing rules:
• agents selected at random
• updates made immediately
• stop on user input or when no more moves can be made
Simulation Elements

Setup: Elements
Example Schelling segregation model

Environment: 50 x 50 grid of spaces where agents can live. Each space is occupied (blue, orange) or open (white). Space is a torus (top cells are adjacent to bottom cells, right to left). 2466 agents

Agents: people with preferences for the color of their neighbors.

Rules:

agent rule: if fewer than $\alpha\%$ of your neighbors are same color as you, move to open spot. $\alpha=50$.

Global processing rules:
• agents selected at random
• updates made immediately
• stop on user input or when no more moves can be made
In a simulation, everything that happens is due to what you program. So you want to think very carefully about how elements interact. We often (rightly) focus on the agent rules; but the environmental conditions & updating rules are often just as important.

Update rules.
parallel vs. serial? When do agents “see” changes in the setting?

• Serial: One agent acts, that affects the environment, then the next agent acts.

• parallel: each agent responds to the state of the system at t, then updates are made all at once for next iteration.

Reality is usually parallel – people act in their own time; but some situations depend on prior actors constraints. For example, parallel action in Schelling could imply people moving to the same cell. So the fixed nature of choice constrains us to serial.
Simulation Elements

Global elements

In a simulation, everything that happens is due to what you program. So you want to think very carefully about how elements interact. We often (rightly) focus on the agent rules; but the environmental conditions & updating rules are often just as important.

Update rules.

- Fixed or random order?

Most simulations have an iterative character – “time” is instantiated in “ticks.”

Fixed: Do you array actors 1:N and run in the same way each time?
Random 1: Do you randomly order nodes each round?
Random 2: Do you randomly select each interactant fresh (so some will act twice before another ever gets chosen)?

Think through how you schedule behavior. In most cases you’ll want to use a randomized order.
Simulation Elements

Global elements

In a simulation, everything that happens is due to what you program. So you want to think very carefully about how elements interact. We often (rightly) focus on the agent rules; but the environmental conditions & updating rules are often just as important.

Update rules.
- Fixed population?

Does your population turn over? If so, what governs entry & exit?

Just a cautionary note: population turnover models are difficult. In settings where agents age & die, one needs to be very careful to initialize the population age distribution such that you don’t get massive “cohorts” of agents entering & exiting at the same time.

➔ That requirement then has knock on effects – if you initialize agents as “old” then what other attributes do you give them? Likely many of the ones you want to figure out from the simulation.
Simulation Elements

Global elements Picking parameters.

Most simulations have a set of control parameters – variables that guide the action between “a little” and “a lot.” **Picking the values and number of parameters is often non-trivial.** The simulation state space is defined by the range of all your parameters.

It is easy to produce ABMs with more parameters than you can make sense of. In data-grounded simulations based on models, the range & such are constrained by the data. But in whole-cloth ABMs you generally are free to pick your parameters as you want.

**Number of parameters.** Avoid adding parameters that you don’t really care about. The goal for most non-data ABMs is to evaluate a key theoretical mechanism. As one tries to get realistic and add more parameters the scope tends to get out of hand.

**Value of parameters.** Often behavior is uninteresting for wide ranges of parameter settings – all above x produce a fully connected graph, all below x an empty one, etc. So trick is to picking parameters that fall in an informative range.
Network Simulations

Specific points

Key distinction is simulation **ON** networks or simulations **OF** networks (or both)

**On Networks:**

The network is given and interest is in how action propagates over the network.

For example, if you wanted to compare differences in spread for simple vs. complex diffusion rules, you would apply the two rules to the exact same network to hold constant differences in communication structure.

Most of the effort in these sorts of simulations are in building the **agent rules** that govern how agents react to the state of their neighbors.

Disease diffusion simulations are the archetype here.
Network Simulations

Specific points

Key distinction is simulation **ON** networks or simulations **OF** networks (or both)

Of Networks:

The goal is to grow the network based on rules about the setting & agent similarity.

Broadly two sorts:

- **Exogenous**: Build the network from given actor similarity features. Example would be simulating a network based on race homophily. In itself only “weakly” agentic – really more a probability model on the state space of agent similarity; but often the first step for many other models.

- **Endogenous**: grow the network based on patterns of ties currently in place. Example would be social balance simulations.
Network Simulations

Specific points

Key distinction is simulation ON networks or simulations OF networks (or both)

Simulation on and of Networks:

Oftentimes we are interested in how the network evolves as a result of how things move through the network. As agents change states they become more/less attractive for others and that creates incentives for how ties are allocated.

SIENA models do this with respect to the observed data – effectively asking what rules (instantiated as probability weights) most accurately interpolate between waves of data.

Belief polarization models are another common example, where people influence their peers on issues and then form ties based on belief similarity.
Network Simulations

Some (perhaps) nonobvious cautionary points:

- **Degree constraints.** Does the setting you are modeling come with activity constraints? Most social settings do; so a good indicator that a bottom-up model is nonsensical is if it generates networks with too large a degree. Ditto distributions.
  - One “solution” is to fix degree and focus the simulation on allocation given degree. Sometimes this is sensible; sometimes not.

- **Agent “memory.”** Do agents have memories beyond the last iteration? Key to “breaking up” relations – if ego drops a tie with alter, are they allowed to reform it again later? Up to what limit? Oftentimes rules that are sensible for two time steps are nonsensical for T timesteps, so need to think through that sort of dynamic.

- **Runaway configurations.** Feedback process models (of & on) can easily walk into clique-like configurations (or highly dissasortative hub & spoke). That’s good if its telling you the (unrealistic) result of a well-specified agent rule; bad if you think your rule is accurate.
Case Studies
Simulation of networks

Substantive question: how does concurrency affect reachability in a population?

Imagine interpolating a hypothetical network from a world where nobody has more than one partner within a short period of time and one where many do.

Environment: Random graph. Only constraints are number of other actors.
Agents: Nodes who have relations & a fixed number of partners.
Agent rule: Form relations at random until degrees are allocated.
Global processing rules: one shot, randomized formation
Case Studies
Simulation of networks

Emergent Connectivity in low-degree networks

Partner Distribution

Component Size/Shape

Largest component: 215
Largest bicomponent: 0
Number in components < 10: 4,973

Largest component: 991
Largest bicomponent: 91
Number in components < 10: 4,154

Largest component: 4,195
Largest bicomponent: 538
Number in components < 10: 3,166

Largest component: 6,433
Largest bicomponent: 1,439
Number in components < 10: 2,281
Case Studies
Simulation of networks
In both distributions, a giant component & reconnected core emerges as density increases, but at very different speeds and ultimate extent.
Case Studies
Simulation of networks

What distinguishes these two distributions?

Shape
What distinguishes these two distributions?
Shape:

The scale-free network’s signature is the long-tail

So what effect does changes in the shape have on connectivity?
Case Studies
Simulation of networks

Dispersion x Skewness

<table>
<thead>
<tr>
<th>Volume</th>
</tr>
</thead>
</table>
Case Studies
Simulation of networks

Search Procedure:
1) Identify all valid degree distributions with the given mean degree and a maximum of 6 w. brute force search.
2) Map them to this space
3) Simulate networks each degree distribution
4) Measure size of components & Bicomponents
Case Studies
Simulation of networks
Case Studies
Simulation of networks

Figure 4. Degree Distribution and Simulated Giant Components

Based on work supported by R21-HD072810 (NICHD, Moody PI), R01-HD065852-01 (NICHD, Moody PI), R01-DA012831-05 (NIDA Morris, Martina PI).
Case Studies
Simulation of networks

C:45%, B: 8.5%
Case Studies
Simulation of networks

Based on work supported by R21-HD072810 (NICHD, Moody PI), R01-HD068523-03 (NICHD, Moody PI), R01-DA012831-05 (NIDA Morris, Martina PI).
Case Studies
Simulation of networks

Based on work supported by R21-HD072810 (NICHD, Moody PI), R01-HD08523-01 (NICHD, Moody PI), R01-DA012831-05 (NIDA Morris, Martina PI).
Case Studies
Simulation of networks

C: 99%, B: 86%
Case Studies
Simulation of networks

Figure 4. Giant Components as a function of average degree and degree distribution shape.

Figure 5. Size of the Largest bicomponent by mean degree and distribution variability.

Largest Component
(at least 1 path)

Largest Bicomponent
(at least 2 paths)

Based on work supported by R21-HD072810 (NICHD, Moody PI), R01-HD068523-01 (NICHD, Moody PI), R01-DA012831-05 (NIDA Morris, Martina PI).
Concurrency has been hypothesized to affect spread of HIV by how it changes the wider temporal structure. Can we identify how?

Who can “A” reach?

Contact network: Everyone, it is a connected component
Concurrency has been hypothesized to affect spread of HIV by how it changes the wider temporal structure. Can we identify how?

Who can “A” reach?

Exposure network: here, node “A” could reach up to 8 others

Node “D” can reach 5 others. While A can reach D, A can only reach 1 of D’s exposure set (c).
Concurrency has been hypothesized to affect spread of HIV by how it changes the wider temporal structure. Can we identify how?

Who can “A” reach?

Transmission network: upper limit is 8, but achieved reach depends on each link successfully passing: transmission is path dependent.
Case Studies
Simulation of networks

The mapping between the contact network and the exposure network is based on relational timing. In a *dynamic* network, edge timing determines if something can flow down a path because *things can only be passed forward in time*.

*Definitions:*

Two edges are *adjacent* if they share a node.

A *path* is a sequence of adjacent edges \((E_1, E_2, ...E_d)\).

A *time-ordered path* is a sequence of adjacent edges where, for each pair of edges in the sequence, the start time \(S()\) of the first edge is less than or equal to the end time \(E()\) of the second: \(S(E_1) \leq E(E_2)\)

Adjacent edges are *concurrent* if they share a node and have start and end dates that overlap. This occurs if:

\[
S(E_1) \leq E(E_2) \& S(E_2) \leq E(E_1)
\]
Case Studies
Simulation of networks

The mapping between the contact network and the exposure network is based on relational timing. In a dynamic network, edge timing determines if something can flow down a path because things can only be passed forward in time.

A picture is simpler:

Concurrent Relations generate multiple paths in the exposure network
Case Studies
Simulation of networks

Unfortunately, hard to find sufficient real-world data to see how variants in timing affect transmission potential. So we simulate!

**Environment:** Sample of Real-world sparse contact networks (collaboration nets), treated as fixed in structure.

**Agents:** Nodes

**Rules:** Randomize when to start/end relations

**Global process rules:** One shot randomization
Case Studies
Simulation of networks
Case Studies
Simulation of networks

“High Cohesive”
Case Studies
Simulation of networks

Relation between Concurrency & Exposure in 4 contact networks

“Low Cohesive”
Social balance is the idea that friends of friends should be friends and has many implications for how networks should evolve.

But, most are static, assuming the network equilibrates.

Must it?

Environment: Network of nodes,
Agents: Adolescents who make and drop friends
Agent Rule: Built around a set of transitions into and out of triadic configurations.
Global Process Rule: Serial, randomized. Perhaps not optimal.
Case Studies
Simulation of networks
*Dynamic Social Balance*

A periodic table of social elements:

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(1)</td>
</tr>
<tr>
<td>003</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>012</td>
<td></td>
</tr>
<tr>
<td>(2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>102</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>111D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>111U</td>
<td></td>
</tr>
<tr>
<td>(4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>201</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>210</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Intransitive**
- **Transitive**
- **Mixed**
Case Studies
Simulation of networks
*Dynamic Social Balance*

(some transitions will both increase transitivity & decrease intransitivity – the effects are independent – they are colored here for net balance)
Case Studies

Simulation of networks

Dynamic Social Balance

Observed triad transition patterns, from Sorensen and Hallinan (1976)
Case Studies

Simulation of networks

*Dynamic Social Balance*

At the micro level, we can ask how different rules about individual behavior affect which paths in the overall network of triad states that will be preferred.

This gives us the first step in understanding which features can simultaneously create coordinated (rule-based) action while maintaining a fluid state-space.

Example: Favor transitions that avoid intransitivity
Case Studies

Simulation of networks

Dynamic Social Balance

Random Walk

Favor Transitivity only (moderate)

Avoid Intransitivity only (strong)
Based on these results, I simulate network dynamics controlling the extent to which actors seek transitivity and avoid intransitivity.

- This simulation builds on the empirical models in specifying separate effects for transitivity and intransitivity based on ego’s returns to a change in relations.

- Adds a parameter to limit the marginal returns to forming new relations, that effectively dampens (but does not hard-code) out-degree.

- Reciprocity & dyad attribute parameters are held constant across all simulations.

- Time is encoded as each node’s opportunity to change relations as iterations pass.

- Summary statistic is the overall graph transitivity score
Case Studies
Simulation of networks
*Dynamic Social Balance*

Final Graph Transitivity

\[ R^2 = 0.82 \]
Case Studies
Simulation of networks

*Dynamic Social Balance*

Structural Stability
Correlation of network structure at $t_{\text{final}}$ with $t_{-5\%}$

$R^2 = 0.52$
Total Graph Transitivity
At moderate transitivity/intransitivity

A single simulation run, showing the wide swings in graph transitivity. Similar trends evident in reciprocity, though the number of arcs and general shape (variance/skew) of the popularity distribution does not fluctuate much.
Case Studies

Simulation of networks

*Dynamic Social Balance*

Total Graph Density
At moderate transitivity/intransitivity

(So the graph has converged on the number of arcs, while the pattern remains fluid)
Case Studies
Simulation of networks
*Dynamic Social Balance*
Case Studies
Simulation ON networks

This is arguably the most active sort of simulation study done in SN&H. The general outline is usually some variant on an SI or SIR sort of infection process:

For a given network structure do:
   Seed the network with initial bit
   For as long as there are discordant pairs do:
      for each discordant pair do:
         transfer based on some rule
      for all infected nodes do
         recover/remove based on rule

Environment: Population of actors linked by network ties.
Agents: represent people who can be infected.
Agent Rule: Biology governs recovery and infectivity
General Process Rules: serial spread, randomized order, stopping rule implicit in state

Many of the interesting questions here turn on how the network structure affects the spread
Case Studies
Simulation ON networks

• Question: What features of a network promote SIR-like diffusion in real-world networks?

• For each network trial:
  • Fix dyadic transmission probability
  • Randomly select a node as seed
  • Trace the diffusion path across the network
    • Measure speed & extent of spread
    • Model extent of spread by structural characteristics

• First run: Add Health:
  • simple diffusion process
  • dyadic probability set to 0.08
  • 500 trials in each network.

• Then expand to :
  • Different assumptions of dyadic transmission probability
  • Different data set (Facebook)
  • Complex diffusion models

This is joint work with Ashton Verdery, Penn State
Node size is proportional to \( \ln(\text{degree}+1) \).
This is joint work with Ashton Verdery, Penn State

Case Studies
Simulation ON networks
Define as a general measure of the “diffusion susceptibility” of a graph as the ratio of the area under the observed curve to the area under the random curve. As this gets smaller than 1.0, you get effectively slower median transmission.

This is joint work with Ashton Verdery, Penn State
Case Studies
Simulation ON networks

Table 2. OLS Regression of Relative Diffusion Ratio on Network Structure

<table>
<thead>
<tr>
<th>Variable</th>
<th>Model 1</th>
<th>Model 2</th>
<th>Model 3</th>
<th>Model 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>1.62***</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Connectivity</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distance</td>
<td>-0.207***</td>
<td>-0.179***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Independent Paths</td>
<td>-0.077***</td>
<td>-0.056***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distance x Paths</td>
<td>0.023***</td>
<td>0.015***</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Clustering</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clustering Coefficient</td>
<td></td>
<td>-0.692***</td>
<td>-0.653***</td>
<td></td>
</tr>
<tr>
<td>Grade Homophily</td>
<td></td>
<td>-0.026**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peer Group Strength</td>
<td></td>
<td>-0.868***</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Degree Distribution</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Degree Skew</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assortative Mixing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Control Variables</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network Size/100</td>
<td></td>
<td>-0.005***</td>
<td>-0.005***</td>
<td></td>
</tr>
<tr>
<td>Proportion Isolated</td>
<td></td>
<td>-1.106***</td>
<td>-0.300*</td>
<td></td>
</tr>
<tr>
<td>Non-Complete</td>
<td></td>
<td>-0.052*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adj- $R^2$</td>
<td></td>
<td></td>
<td></td>
<td>0.76</td>
</tr>
<tr>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td>124</td>
</tr>
</tbody>
</table>

This is joint work with Ashton Verdery, Penn State
### Case Studies

Simulation ON networks

#### Table 2. OLS Regression of Relative Diffusion Ratio on Network Structure

<table>
<thead>
<tr>
<th>Variable</th>
<th>Model 1</th>
<th>Model 2</th>
<th>Model 3</th>
<th>Model 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>1.62 ***</td>
<td>1.90 ***</td>
<td>1.02 ***</td>
<td>1.81 ***</td>
</tr>
<tr>
<td><strong>Connectivity</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distance</td>
<td>-0.207 ***</td>
<td>-0.179 ***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Independent Paths</td>
<td>-0.077 ***</td>
<td>-0.056 ***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distance ( \times ) Paths</td>
<td>0.023 ***</td>
<td>0.015 ***</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Clustering</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clustering Coefficient</td>
<td>-0.692 ***</td>
<td></td>
<td>-0.653 ***</td>
<td></td>
</tr>
<tr>
<td>Grade Homophily</td>
<td>-0.026 **</td>
<td></td>
<td>-0.007</td>
<td></td>
</tr>
<tr>
<td>Peer Group Strength</td>
<td>-0.868 ***</td>
<td></td>
<td>-0.141</td>
<td></td>
</tr>
<tr>
<td><strong>Degree Distribution</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Degree Skew</td>
<td></td>
<td>-0.023</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assortative Mixing</td>
<td></td>
<td>-0.189*</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Control Variables</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network Size/100</td>
<td>-0.005 ***</td>
<td>-0.005 ***</td>
<td>-0.005***</td>
<td></td>
</tr>
<tr>
<td>Proportion Isolated</td>
<td>-1.106 ***</td>
<td>-0.984 ***</td>
<td>-0.984***</td>
<td></td>
</tr>
<tr>
<td>Non-Complete</td>
<td>-0.052*</td>
<td>-0.078**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adj- R(^2)</td>
<td>0.76</td>
<td>0.60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>124</td>
<td>124</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is joint work with Ashton Verdery, Penn State
## Case Studies

Simulation ON networks

<table>
<thead>
<tr>
<th>Table 2. OLS Regression of Relative Diffusion Ratio on Network Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variable</strong></td>
</tr>
<tr>
<td>--------------------------------------</td>
</tr>
<tr>
<td><strong>Intercept</strong></td>
</tr>
<tr>
<td><strong>Connectivity</strong></td>
</tr>
<tr>
<td>Distance</td>
</tr>
<tr>
<td>Independent Paths</td>
</tr>
<tr>
<td>Distance x Paths</td>
</tr>
<tr>
<td><strong>Clustering</strong></td>
</tr>
<tr>
<td>Clustering Coefficient</td>
</tr>
<tr>
<td>Grade Homophily</td>
</tr>
<tr>
<td>Peer Group Strength</td>
</tr>
<tr>
<td><strong>Degree Distribution</strong></td>
</tr>
<tr>
<td>Degree Skew</td>
</tr>
<tr>
<td>Assortative Mixing</td>
</tr>
<tr>
<td><strong>Control Variables</strong></td>
</tr>
<tr>
<td>Network Size/100</td>
</tr>
<tr>
<td>Proportion Isolated</td>
</tr>
<tr>
<td>Non-Complete</td>
</tr>
<tr>
<td>Adj- $R^2$</td>
</tr>
<tr>
<td>N</td>
</tr>
</tbody>
</table>

This is joint work with Ashton Verdery, Penn State
Case Studies
Simulation ON networks

Figure 4. Relative Diffusion Ratio
By Distance and Number of Independent Paths

This is joint work with Ashton Verdery, Penn State
Case Studies
Simulation ON networks: Dual diffusion

Note that simulations on networks can very profitably simulate multiple things simultaneously.

Here both flu & preventive behavior diffuse through the network.
Note that simulations on networks can very profitably simulate multiple things simultaneously.

Here both flu & preventive behavior diffuse through the network...and the simulation is embedded in real geography.
Use local network information from two sources (general population & FSW) to generate a synthetic population, simulate dynamic sexual contact networks and then layer a biologically realistic transmission probabilities to assess the potential extent of generalized HIV spread in China.
Case Studies
Simulation of AND on networks (no feedback)

The main local information was age & SES Mixing (tables are marginal, simulation used joint distribution).
Case Studies
Simulation of AND on networks (no feedback)

The main local information was age & SES Mixing (tables are marginal, simulation used joint distribution).
Case Studies
Simulation of AND on networks (no feedback)

The main local information was age & and the degree distribution, for general population (by category specific) and FSW.
Case Studies
Simulation of AND on networks (no feedback)

The main local information was age & and the degree distribution, for general population (by category specific) and CSW.
Case Studies
Simulation of AND on networks (no feedback)

The main local information was age & and the degree distribution, for general population (by category specific) and FSW.

Fig. 4 Observed degree distribution of female sex workers, Shanghai. Source: SWHS
Case Studies
Simulation of AND on networks (no feedback)

Result 1: Component size (ignores transmission elements). Size results very robust to assumptions about FSW degree distribution.

Structural simulation was done using a variant of the “zipper” method – starting with category-specific edges and drawing nodes to fill out the population.

Fig. 5 Size distribution of connected components and of largest component as proportions of all nodes.
Case Studies
Simulation of AND on networks (no feedback)

Result 1: Component size (ignores transmission elements). Size results very robust to assumptions about FSW degree distribution.

Most of the connectivity emerges due to (a) FSW sharing male partners, (b) young general population men w. more partners.

Fig. 6 Single run of largest component (left) and one eight-step walk from a randomly chosen node (right)
Result 2: Transmission simulation within the network.

Assume high degree (mainly FSW) starting nodes, with “spontaneous” outside infections (to capture boundary effects). Simulation runs for 30 years, with realistic relationship dynamics (concurrency, turnover, etc.).

Results indicate very small proportion... (but China is big!)

Fig. 8 Proportion of the population infected, by dyadic transmission probability and coital frequency with noncommercial sex partners, based on dynamic network disease transmission simulation (N = 1,200 in each cell)
Pragmatic points

*Implementation*

One can implement a network simulation in any software you’re comfortable with, but there are some basic implementation hints.

For models on the theory/illustration end, NetLogo is remarkably efficient and is a very easy-to-use language with a large sample library of models.
Pragmatic points

Implementation

Simulation modeling often requires substantial computational power even when its programmed very efficiently, but poor programming organization will make a difficult task unbearable. So be sure to keep the following in mind as you design your study:

- data structures.
  → which data elements do you have to update in place? What do you add to a list? Which do you store for later summary? Etc.

- order of operations & data structure
  → Much of the logic of simulations is iterative, but often its faster to program it otherwise. You want to avoid unnecessary sorting, searching in data frames, etc. So, instead of this:

```r
say you have 50 actor that have a probability of learning of 0.8. You could flip a coin with probability 0.8 for all of them:

`learners <- rep(NA, 50)`

`for (i in 1:50) {
  explores <- rbinom(1,1,0.8)
  learners[i] <- explores
}`
```

Code example thanks to Nicolas Restrepo Ochoa
Pragmatic points

*Implementation*

Simulation modeling often requires substantial computational power even when its programmed very efficiently, but poor programming organization will make a difficult task unbearable. So be sure to keep the following in mind as you design your study:

- data structures.
  → which data elements do you have to update in place? What do you add to a list? Which do you store for later summary? Etc.

- order of operations & data structure
  → Much of the logic of simulations is iterative, but often its faster to program it otherwise. You want to avoid unnecessary sorting, searching in data frames, etc. Do this:

```r
explorers <- sample(c(TRUE, FALSE), 50, prob = c(0.8, 1 - 0.8), replace = TRUE)
```

Code example thanks to Nicolas Restrepo Ochoa
Pragmatic points

*Implementation*

Simulation modeling often requires substantial computational power even when its programmed very efficiently, but poor programming organization will make a difficult task unbearable. So be sure to keep the following in mind as you design your study:

- **data structures.**
  - which data elements do you have to update in place? What do you add to a list? Which do you store for later summary? Etc.

- **order of operations & data structure**
  - While the simulation space might be growing or shrinking, avoid changing dimensions of your data space. So instead of this:

```r
new_offspring <- tibble(age = rep(1, 5),
                        trait = sample(c("A", "B"), 5,
                                       replace = T))
```

```r
df <- rbind(previous_df, new_offspring)
```

Code example thanks to Nicolas Restrepo Ochoa
Pragmatic points

Implementation

Simulation modeling often requires substantial computational power even when its programmed very efficiently, but poor programming organization will make a difficult task unbearable. So be sure to keep the following in mind as you design your study:

- data structures.
  → which data elements do you have to update in place? What do you add to a list? Which do you store for later summary? Etc.

- order of operations & data structure
  → While the simulation space might be growing or shrinking, avoid changing dimensions of your data space. Do this:

```r
# Baseline fertility
b <- 0.01
# Bonus fertility
b <- 0.05
# Number of turns
t_max <- 500
# Initial population size
N <- 500
# Repository (enough rows; plus one column for age)
population <- matrix(NA,
  nrow= N+N*t_max*b+bonus_fitness,
  ncol = t_max+1)
```

Code example thanks to Nicolas Restrepo Ochoa
Pragmatic points

Implementation

Simulation modeling often requires substantial computational power even when its programmed very efficiently, but poor programming organization will make a difficult task unbearable. So be sure to keep the following in mind as you design your study:

- Parameter space
  - Even simple simulations can explode combinatorically. For diffusion, you might have variability in recovery, transmissibility, contact frequency and duration. With only 5 levels per, that’d give you $5^4=625$ sample points in the state space. If each run takes 5 seconds, that’s just under an hour to get 1 draw from the state space. You probably want 1000...

- For R users, Purr & Map() are good tools for exploring a parameter space:

```r
library(tidyverse)

vector_si <- rep(seq(0, 1, by = 0.1), 11)
vector_g <- rep(seq(0, 1, by = 0.1), each = 11)
d2 <- map2(.x = vector_si, # Parameter of interest 1
            .y = vector_g, # Parameter of interest 2
            diff_inf_sim, # Your model
turns = 100, init_prob = 0.25, # Other arguments
            n_agents = 100, rounds = 25)
```

Code example thanks to Nicolas Restrepo Ochoa
Pragmatic points

Implementation

Simulation modeling often requires substantial computational power even when its programmed very efficiently, but poor programming organization will make a difficult task unbearable. So be sure to keep the following in mind as you design your study:

- Build the model slowly
  Build your simulations like onions (Ogres?) – in layers from the inside out. You want to make sure that 1 tic does exactly what you think it’s doing, then that a single parameter run from t=1 to T works as you expect.

  You should be able to specify corner conditions or modular bits that give you known results that you can check.

  Once you know that the inner layers are working right, then go on to integrating over your parameter space.
Pragmatic points

Implementation

Simulation modeling often requires substantial computational power even when its programmed very efficiently, but poor programming organization will make a difficult task unbearable. So be sure to keep the following in mind as you design your study:

- Build in real-time progress checks.
  Have your model spit some sort of output to a device you can monitor in real time. There are few things more frustrating than having the machine hang or the routine stop and not know it. By writing steps to a logfile you can see ... you can tell the difference between stalled out and still running.

- Avoid “Do While” sorts of loops, where the “while” condition is an outcome of your simulation that could possibly not happen. i.e. you never want to get your simulation stuck in an infinite loop.
  - Very easy to make this mistake – you could specify an integer stopping point (do until i=100), but find that it jumped over 100 because two actors were infected at once. Then it can never get back to 100...etc.

- I generally add in exit/non-convergence checks if I have to do large iterative processes. Just a safety valve.
Pragmatic points

*Implementation*

Simulation modeling often requires substantial computational power even when its programmed very efficiently, but poor programming organization will make a difficult task unbearable. So be sure to keep the following in mind as you design your study:

- Audit/Replication
  Your code is your data in simulations, so it’s useful to make sure that the project produced replicable, auditable code.
  - Particularly when testing/debugging, use a known random number seed, for example.
  - Comment the hell out of your code.
  - If working in a team, build modularly and test each other’s modules.
Conclusions

Many of the questions we ask of the world cannot be answered with real-life data.

Simulations provide a tool to think through these sorts of issues that can be very flexible.

The space of models is wide...and each type has its own challenges.