Session 2: Afternoon

Model Construction

Outline

- Why use ABM?
- Explanation versus Prediction
- Process of Model Construction
  - Induction vs. Deduction
  - Homomorphism
  - Pseudo Code
- Boolean Operators

Uses of ABM

- Gilbert & Troitzsch's seven uses
  - Gain a better understanding
  - Prediction
  - Substitute for human expertise
  - Training
  - Entertainment
  - Discovery
  - Formalization

Why use ABM? (1)

- Theory: fidelity to actual system
  - Nonlinearity
    - Cannot be understood analytically
    - Unpredictable
  - Space
    - Physical (GIS)
    - Network
  - (Bounded?) rationality of actors (Simon)
  - Dynamics rather than equilibria

Why use ABM? (2)

- Methodology
  - Formalization: expose hidden assumptions
    - Pepinsky, p. 374: "Simulation forces the researcher to examine deeply the assumptions that she makes about the environment, the agents, the rules and parameters."
  - Relationships between levels of analysis
    - Emergence
    - The agent-structure debate (Wendt)
  - Quasi-experimentation
  - Counter factuals (Fearon)
  - "micro foundations"

Why use ABM? (3)

- Agent learning
  - Inverts rational choice assumption
  - Change through learning
  - Appeal of ABM to social constructivism

	Rational Choice	ABM
Assumptions	Interests	Behavior
Inferences	Behavior	Interests

Epistemology

- Two issues
  - Explanation vs. Prediction
  - Point vs. Trend Prediction

Explanation vs. Prediction

- Gilbert and Troitzsch
  - ABM best for "discovery" and explanation
  - Not good for prediction
- Prediction is not the only epistemological criterion
  - Scientific realism vs. positivism
  - Theories that predict well but are based on the wrong explanation are problematic

Point vs. Trend Prediction

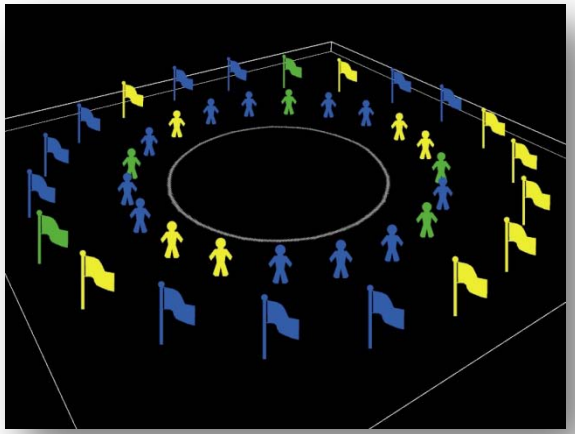
- Types of prediction
  - ABM good for "trend" prediction
    - Explore the statistical distribution of outcomes in the model
    - Provide a good sense of probabilities of events
  - ABM not good for "point" prediction
    - Due to sensitivity of macro parameters to micro values

"Model"

- All agent-based models:
  - Represent the environment of the system
  - Stipulate relevant actors
  - Specify rules (algorithms) and parameters
  - Explore behavior over (simulated) time
- NOTE: does not require assumption of linearity
  - Or indeed of any functional form
  - Contra regression analysis
- Clausewitz: "The map is not the territory"

Notes about vocabulary

- "Target" vs. "Model"
- Simulation
  - "creating an artificial representation [model] of a real world system [target] in order to manipulate and explore the properties of that system." Pepinsky, p. 369



Notes about vocabulary

- "Inputs"
  - Similar to "independent" variables"
- "Outputs"
  - Similar to "dependent" variables
- "Parameters"
  - Not "variables" because we are not sampling
- Time vs. simulated time
  - Time in the model may or may not correspond to actual time
- Data
  - "Empirical": from the target
  - "Meta": from the model

ABM for Business & Marketing

- Core concerns of social theory
  - Change through actor learning
  - Reciprocal influence of agents and structures
  - Counterfactual reasoning

Business examples

- We'll read for July 26th

Epistemology

- Questions?

I. The Building Process

Steps of Research

1. Puzzle
2. Definition of target
3. Observations of target
4. Assumptions and design the model
   a) "Environment"
   b) "Agents"
   c) Parameters and rules
   d) Time
5. Simulate
6. Record data
7. Verification
8. Validation
9. Sensitivity analysis

Your Team Project

- Suggestions for picking a topic:
  - Identify a process of interest
  - Identify a (possible) complex adaptive social system
  - Identify an historical example

Building a model

- An example: The Protests in Iran

Steps

- Identify agents and environment
- Write pseudo code
- Translate into NetLogo

1. Identify Agents & Environment

- Two approaches
  - "Inductive" approach
  - "Deductive" approach

Inductive Approach

- GOAL: represent an actual process or system as an ABM
  - A "high fidelity" model
    - Lots of data points
    - Very detailed
  - For predicting the behavior of the target system
    - e.g. what will happen in the Iranian protests?

Inductive Approach

- Methods: ethnographic research
  - Identify relevant actors
  - Interview them to identify
    - Relevant agent attributes
    - Behavioral routines
  - Build ABM procedures based on empirical observation

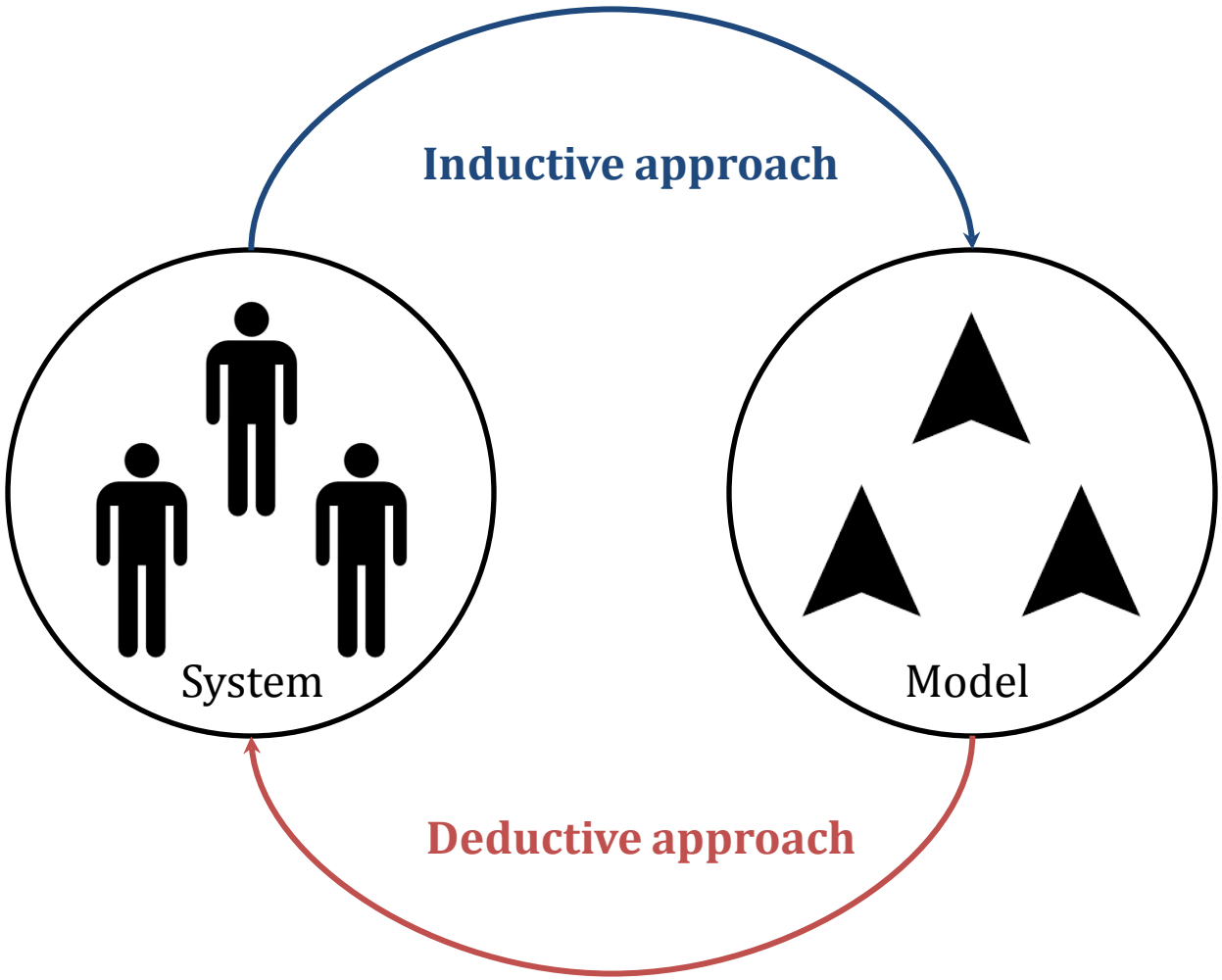
Deductive Approach

- GOAL: build an "artificial society"
  - Highly abstract
  - Generalize across empirical domains
    - e.g. theories of social mobilization
  - For explanation rather than prediction

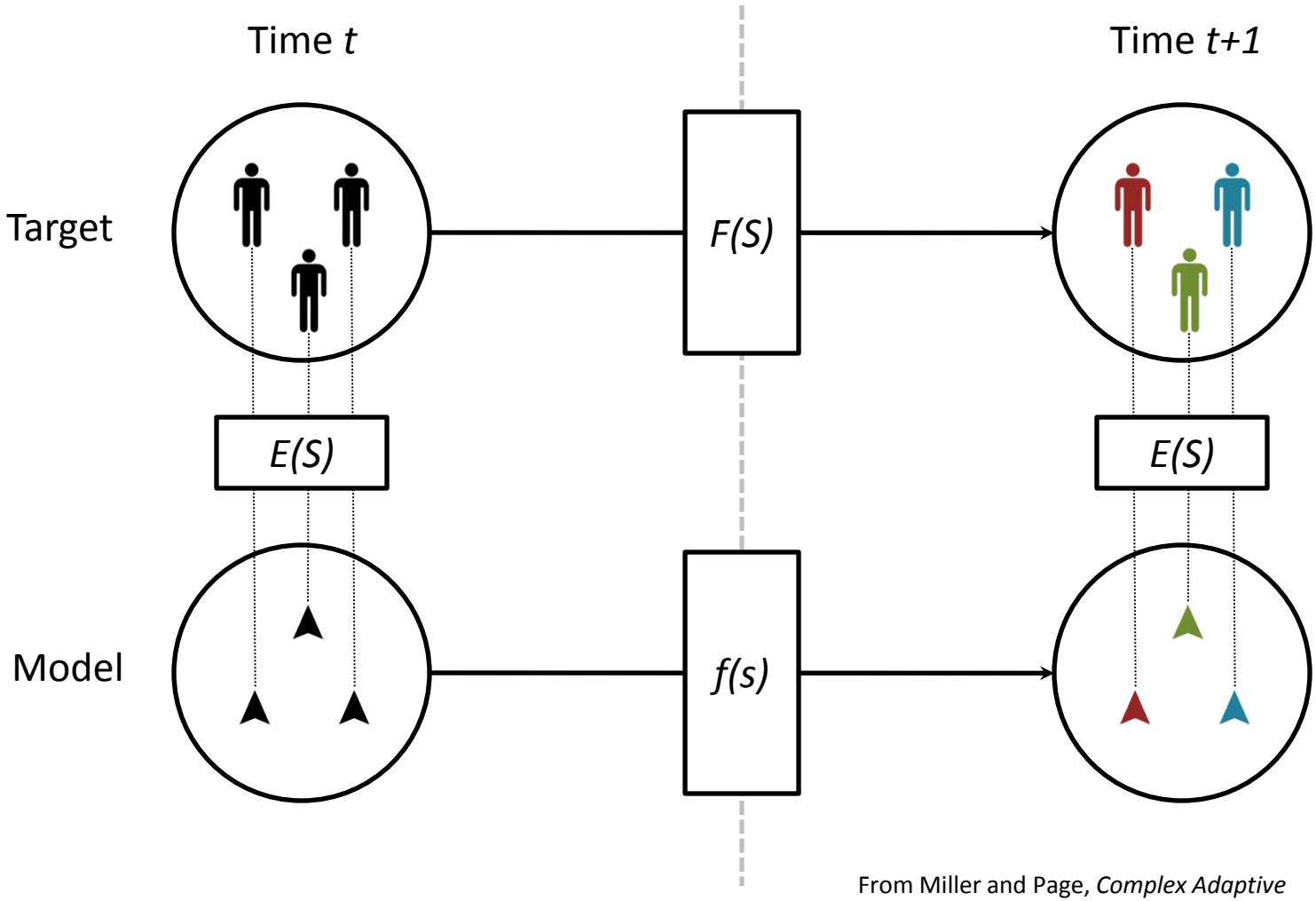
Deductive Approach

- Methods: formalization
  - Make assumptions about actors, attributes
  - Build ABM behavioral procedures based on general social theory
    - e.g. rational choice, satisficing, prospect theory
  - Compare formal model to empirical system
- Note: most library models in NetLogo fall in this category

Approaches compared

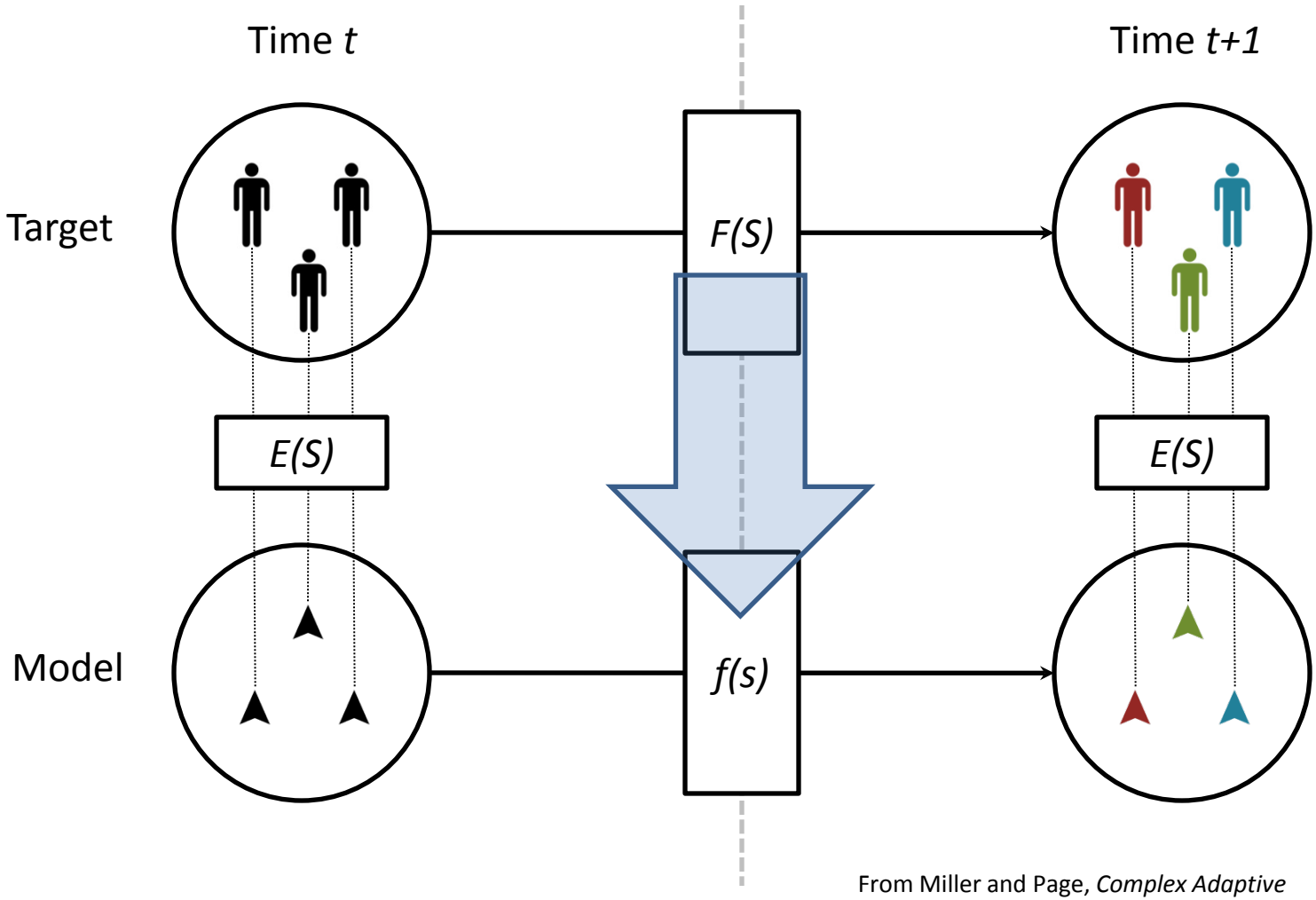


Homomorphism



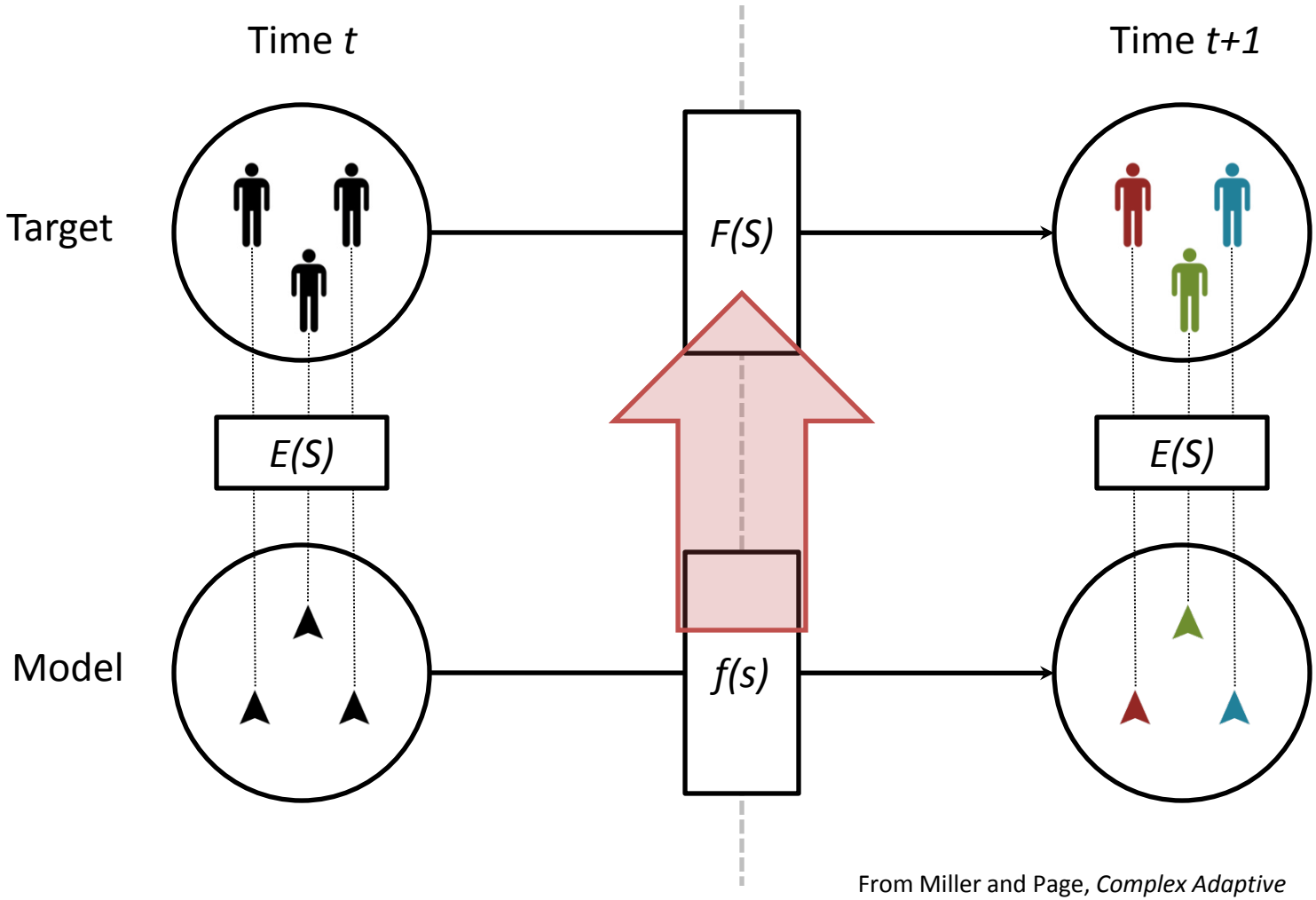
From Miller and Page, *Complex Adaptive Systems* (2007) p. 38.

Homomorphism and Induction



From Miller and Page, *Complex Adaptive Systems* (2007) p. 38.

Homomorphism and Deduction



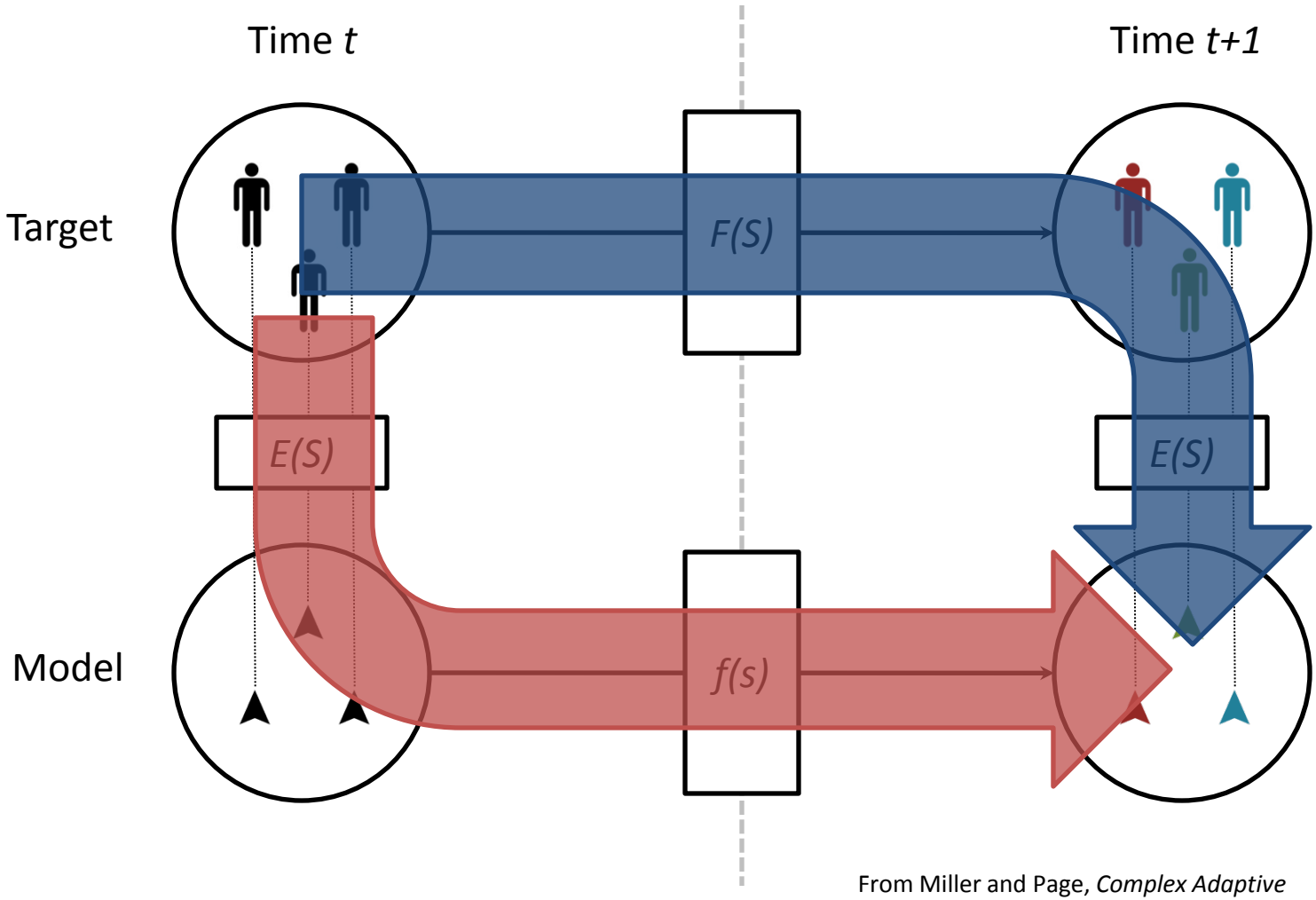
From Miller and Page, *Complex Adaptive Systems* (2007) p. 38.

Homomorphism

- Induction
  - Observe change S' in real system, then map onto the model
  - F(E(S'))
- Deduction
  - Model change s', then compare to the real world
  - E(f(s'))

$$F(E(S')) = E(f(s'))$$

Homomorphism



From Miller and Page, *Complex Adaptive Systems* (2007) p. 38.

1. Identify Agents & Environments

- Questions?

2. Pseudo Code

- Gilbert, p. 54: "an informal mixture of natural language and programming conventions that makes the structure and flow of a program clear without requiring the reader to be familiar with any particular programming."
  - NOTE: no conventions (of which I'm aware)

Pseudo code

- GOAL:
  - Begin to think about the process of coding
  - Identify relevant agents, patches
  - Identify relevant variables
  - Identify globals

Pseudo Code Example

```
Initialization
    Create three types of agents
        protestors endowed with
            social network
            preference
        police
        militia
    Create patches as Azadi Square

Execution
    Loop forever:
        each protestor:
            perceptions
                counts number of nearby protestors (in-radius ?)
                counts number of nearby police + nearby militia
                checks preferences of nearby protestors, with some error
            performance
                communicates: checks preferences of social network
            policy
                evaluations: if ratio of protestors to (police + militia) is greater than P_threshold
                    Join the protest
                    acts: moves-with-crowd
                otherwise: wander
        each police and militia:
            counts number of nearby police + nearby militia
            counts number of nearby protestors
            if number of police + militia is greater than M threshold
                move as a group toward protestors
    end loop
    . . . .
```

Reminder!

- Pseudo code for your team's model due on Tuesday, July 12th

3. Translate into NetLogo code

- Declaration of variables
- "Setup" procedure
- "Go" procedure

NetLogo code example

Azadi Square - NetLogo {D:\My Documents\Data\Courses\2009-10\IS 795-895 Fall 2009\Mode...

File Edit Tools Zoom Tabs Help

Interface | Information | Procedures

Find... Check | Procedures | Indent automatically

```
globals [ police-size protester-size ]

turtles-own
    [
      protester?
      militia?
      police?
      p-threshold
      m-threshold
    ]

patches-own [ walkable? ]


to setup
  ca
  crt 1000
  ask turtles
   [
     let target one-of patches with [ walkable? ]
     setxy [ pxcor ] of target [ pycor ] of target
     set shape "person"
     set protester? false
     set militia? false
     set police? false

     ifelse (random 1000 < 200)
      [
        ifelse (random 1000 < 50)
         [ set police? true set militia? false ]
         [ set police? false set militia? true ]
      ]
      [
        set protester? true
      ]
   ]
   setup-patches
end

to go
  count-neighbors
  check-neighbor-preferences
  check-network-preferences
  move-with-crowd
```

Notice . . .

- "Nested" procedures
  - A "go" procedure can call another procedure
  - Any procedure can call any procedure
- Clunky turtle variables
  - true/false turtles-own variables to keep track of
    - Protesters
    - Police
    - militia

"breeds"

- NetLogo "breed" primitive
  - Allows you to declare different types of agents
  - Then write code using breed name
  - Compare

```
ask turtles with (police?) [ fd 1 ]

ask police [ fd 1 ]
```

Use of Breeds

Azadi Square - NetLogo {D:\My Documents\Data\Courses\2009-10\IS 795-895 Fal...

File Edit Tools Zoom Tabs Help

Interface | Information | Procedures

Find... Check | Procedures | Indent automatically

```
breed [ police pol ]
breed [ protesters protester ]
breed [ militia militant ]

globals [ police-size protester-size ]

police-own [ m-threshold ]
protesters-own [ p-threshold ]
patches-own [ walkable? ]


to setup
  ca
  create-protesters 800
  create-police 50
  create-militia 150

  ask turtles
   [
     let target one-of patches with [ walkable? ]
     setxy [ pxcor ] of target [ pycor ] of target
     set shape "person"
   ]
  ask police [ set color blue ]
  ask militia [ set color blue - 3]
  ask protesters [ set color green ]
  setup-patches
end

to go
  count-neighbors
  check-neighbor-preferences
  check-network-preferences
  move-with-crowd
end
```

Model Construction

- Questions?

II. Topics in Modeling

Boolean operators

- CAS
  - Are massively parallel
  - Have strictly local decision-making
- Modeling this micro-level agency requires modeling contingency . . .
  - . . . in time
  - . . . in space
  - . . . in states/variables

Same Rules, Different Behaviors

- Heterogeneity of capabilities
- Heterogeneity of locations
- Heterogeneity of histories
- Heterogeneity of strategies

- Yugo vs. Lamborghini
- Red vs. Green Light
- Pulling out of the driveway vs. arriving at the store
- Stay off the Motorway vs. buy a radar detector

Contingency

- NetLogo primitives
  - if . . .
  - ifelse . . .
  - while . . .
  - foreach . . .

"if"

- Specifies actions only if some condition is true
- NetLogo syntax:

```
if (TRUE CONDITION)
        [ commands to execute ]


ask turtles
        [
        if (energy < 5)
                [ eat-grass ]
        ]
```

"ifelse"

- Specifies actions for both the true condition and the false condition
- NetLogo syntax:

```
ifelse (TRUE CONDITION)
        [ commands if condition is true]
        [ commands if condition is false ]


ask turtles
        [
        ifelse (energy > 100)
                [ make-new-turtle ]
                [ eat-grass]
        ]
```

"while"

- Execute a loop of instructions as long as some condition is true
- NetLogo syntax

```
while [ TRUE CONDITION ]
  [ commands to execute ]


while [ count turtles with [ shape = "bug" ] < 0.5 * count turtles ]
  [
  ask one-of turtles with [ shape = "bug" ]
      [
      ask one-of turtles with [ shape != "bug" ]
          [ set shape "bug" ]
      ]
  ]
```

"foreach"

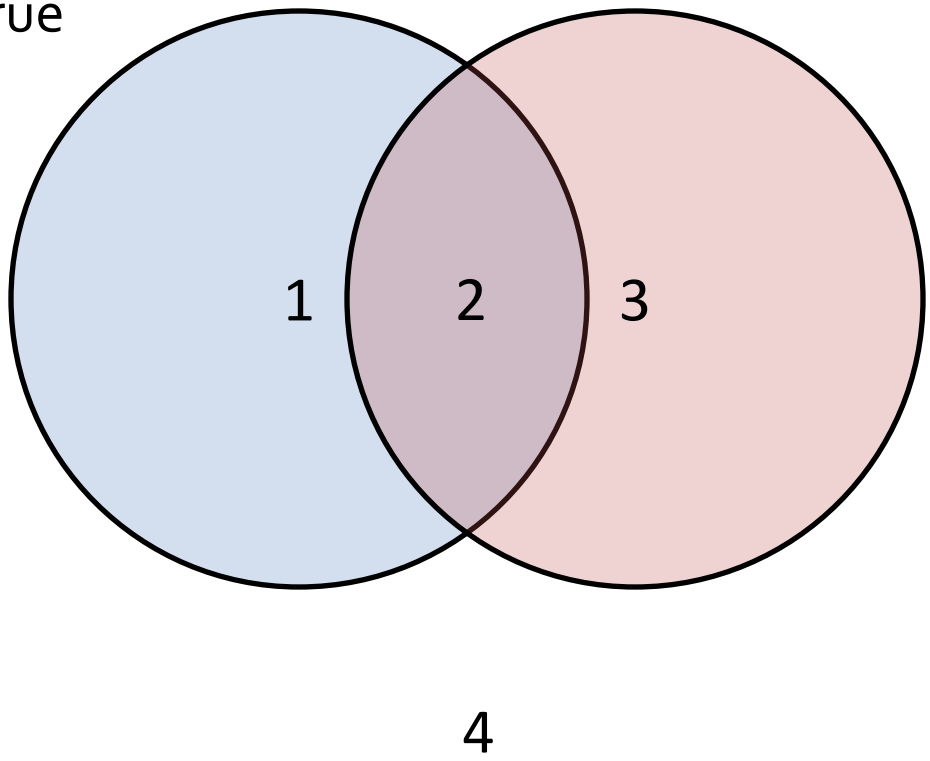
- Execute a loop of commands for each item in a list
- NetLogo syntax

```
foreach [ x y z ]
        [ commands to execute ]

foreach [ 1 3 8 ]
        [
        ask turtle ? [ set color yellow ]
        ]
```

Boolean conjunctions

- "and"
  - Both conditions are true
  - Area (2)
- "or"
  - Either is true
  - Area (1 + 2 + 3)
- "xor"
  - One or other, but not both
  - Areas (1 + 3)
- "not"
  - Condition is not true
  - Area (4)



Boolean Operators

- Questions?

Measuring results

- What outcomes or "emergent properties" do I wish to measure?
- What constitutes "success" or "failure"?

Measuring results

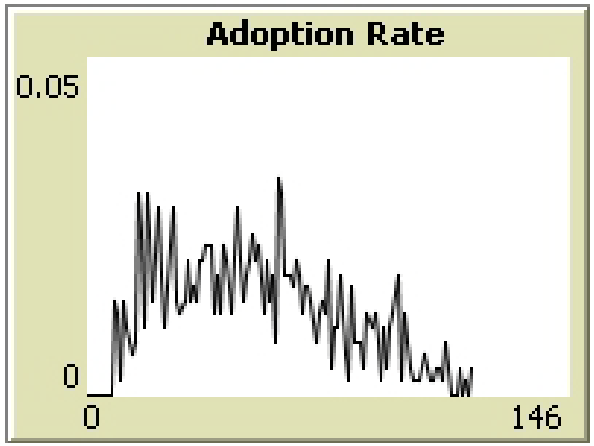
- Monitors
- Plots and histograms
- Output

Creating Monitors

- Global variables
- Turtle variables
- Patch variables

Creating Plots and Histograms

- Summarize change in variables . . .
  - Over time
  - Across the population



Output

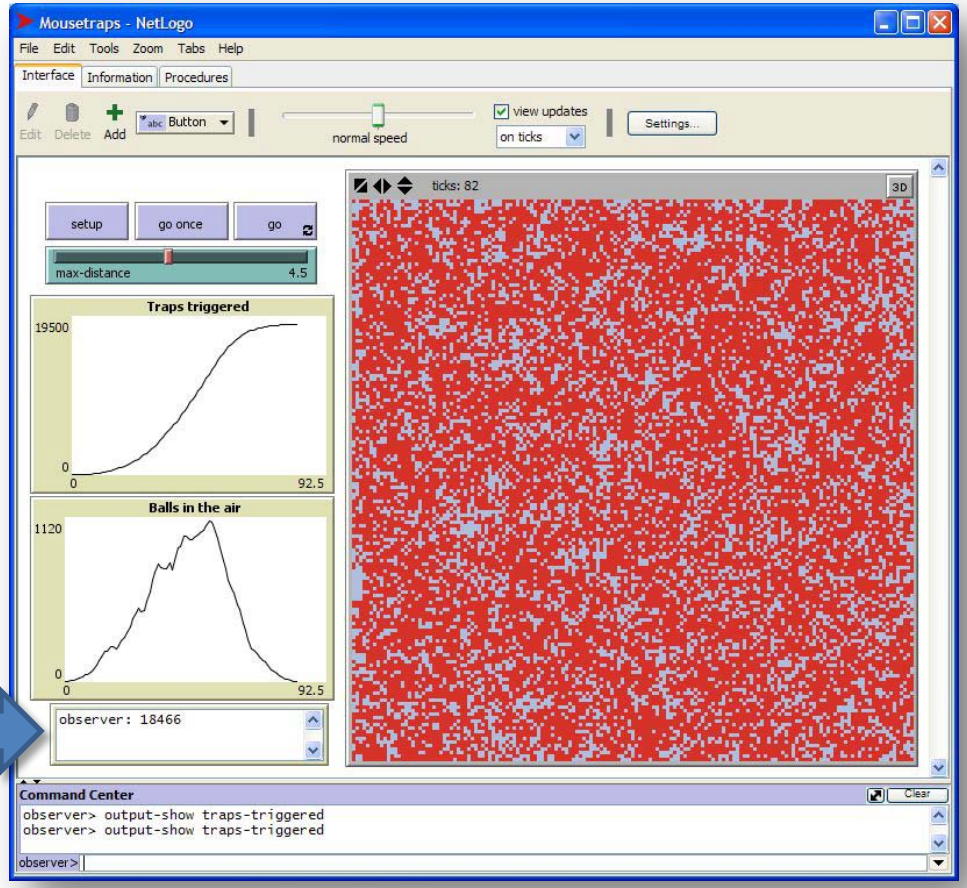
- Write data to
  - Command center (temporary)
  - Output Interface (temporary)
  - External file (saved)

Output to Command Center

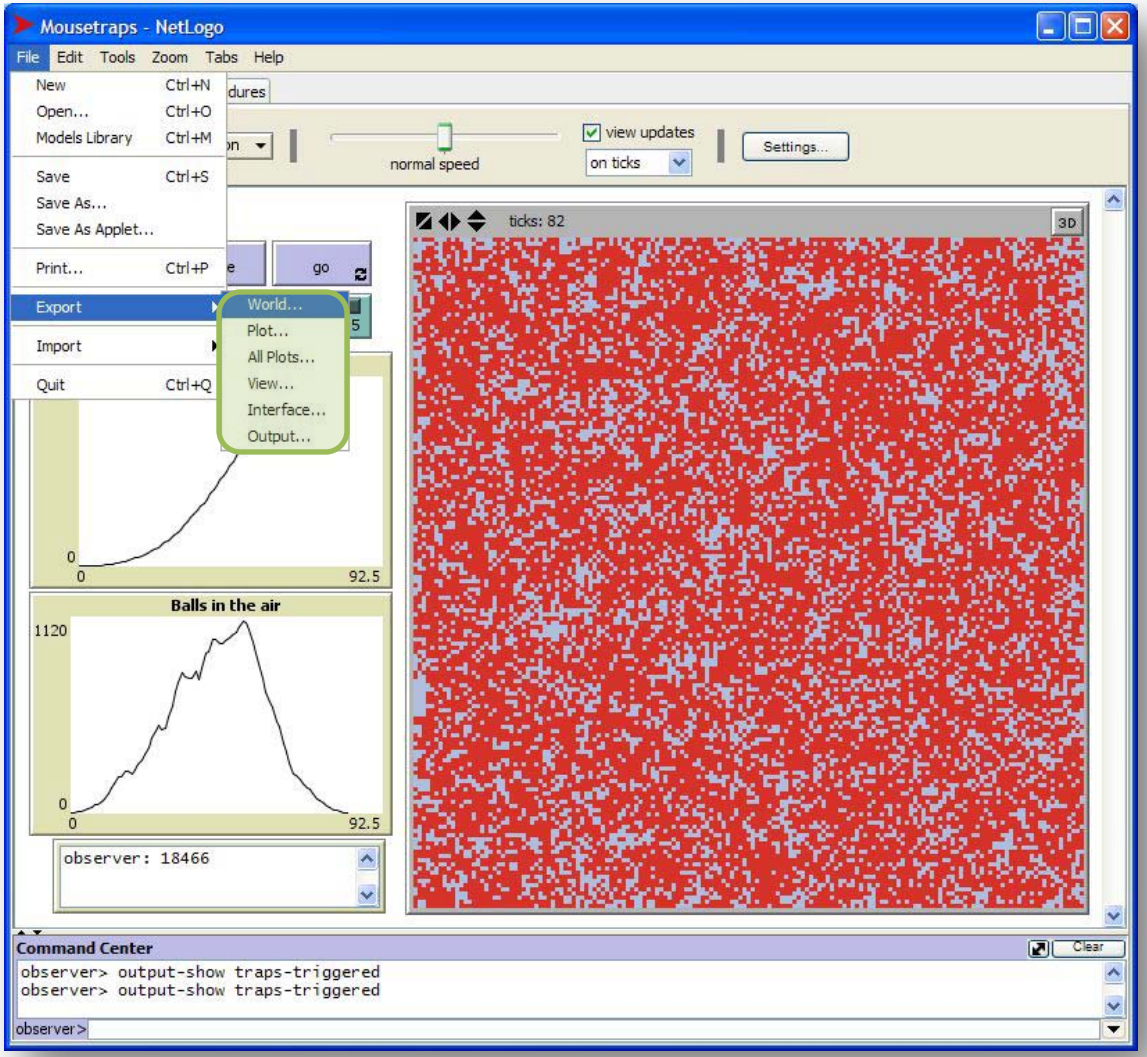
- NetLogo primitives
  - "show"
  - "print"
  - "write"
  - "type"

Output to Interface

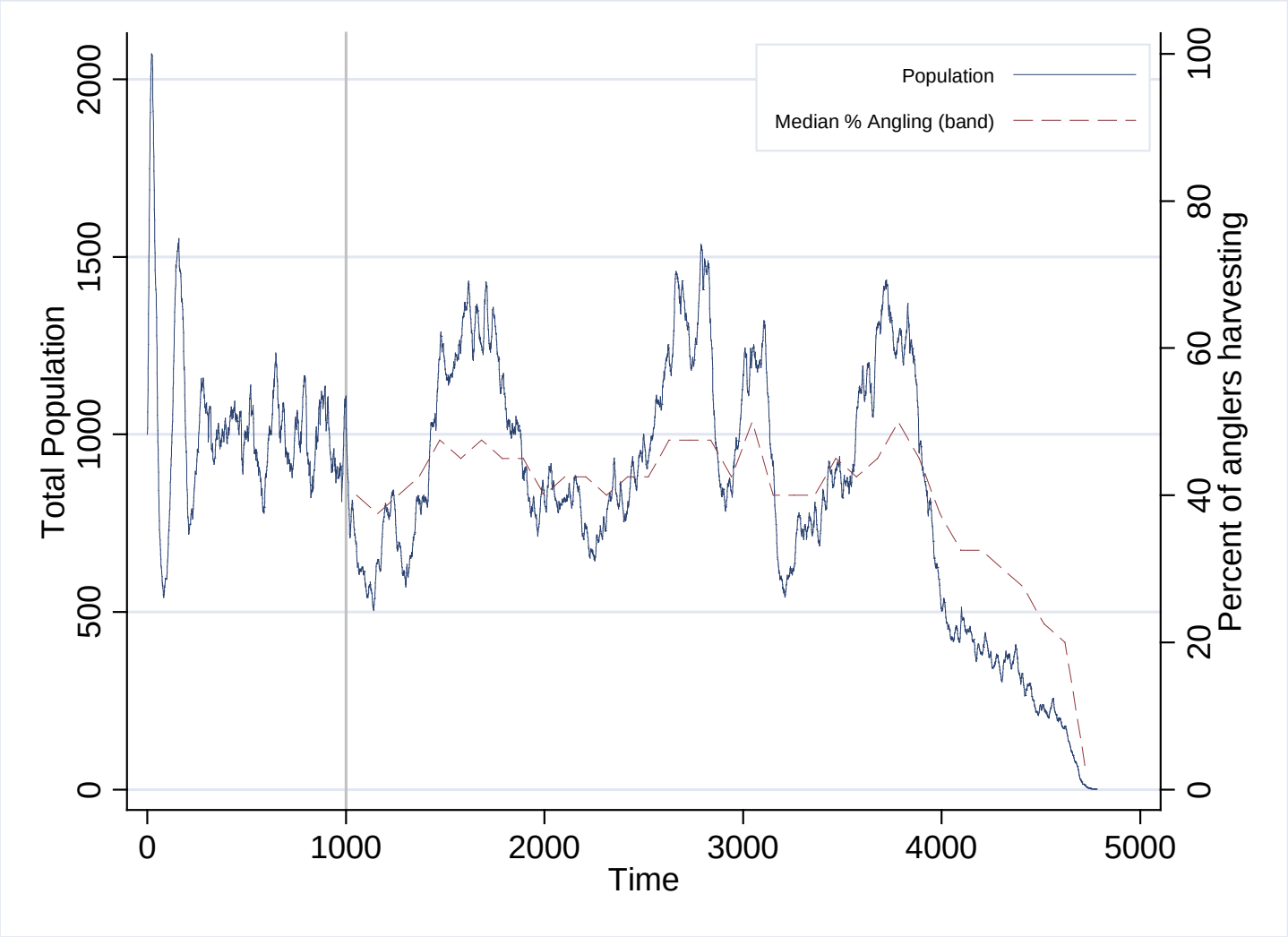
- NetLogo primitives
  - "output-show"
  - "output-print"
  - "output-write"
  - "output-type"



Output to External File



Sample Data



Output

- Questions?

Tuesday

- Emergence, nonlinearity, tipping and equilibria
- Social versus physical, biological, and ecological systems