Session 1: Afternoon

Introduction to NetLogo
Outline

- Agents as “mini-programs”
- Turtles, patches, and globals
- Six types of rules
- Understanding the NetLogo interface
- Syntax: “nouns”, “verbs” and “paragraphs”
- Getting help
A NetLogo Example

Session 1b: July 5
Software

- Creates massively parallel system
  - Each “actor” a program (i.e. a set of rules)
  - No single governing equation or routine
  - Computer executes each program “simultaneously”
  - “Fitter” rules survive and propagate
  - New rules constantly explore
NetLogo Software

5th am Intro
5th pm NetLogo
7th am Theory
7th pm Modeling
12th am Emergence
12th pm Systems
14th am Design
14th pm Experiments
19th am Criticisms
19th pm Validation
21st am Smart Agents
21st pm Networks
26th Applications
28th Presentations

Session 1b: July 5
R1: If no traffic, speed up
R2: If traffic, slow down
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 Motorways vs. buy a radar detector*

Session 1b: July 5
NetLogo Basics

➤ Three Elements
➤ “Patches”
➤ “Turtles”
➤ The System or “global” variables
Elements of space

- Change
- Do not move
★ Social actors
  ★ Change
  ★ Mobile
All turtles and patches put together

Typically, we wish to observe the system

How many turtles are sick? Alive?
Turtles and patches have rules that can:

- Change themselves (reflexive)
- Change other turtles
- Change other patches
Reflexive behavior

```
ask turtles [ forward 1 ]
```
Rules for Turtles

- Reflexive state

```lisp
ask turtles
  [ if (sick?)
    [ set color blue ]
  ]
```

Session 1b: July 5
Rules for Turtles

» Change other turtles

If (sick?)

[ ask turtles here
[ set sick? true
  set color blue
]
]
Rules for Turtles

➤ Change patches

```plaintext
ask turtles [ ifelse (sick?)
    [ ask patch-here [ set grass grass - 5 ]
        set (sick?) false
    ]
    [ ask patch-here [ set grass grass - 1 ] ]
]
```
Rules for Turtles

» Change patches

```netlogo
ask turtles [ 
  if (sick?) and (time-sick > 50) [ 
    ask patch-here [ insert-tombstone ] 
    set dead dead + 1 
    die 
  ]
]
```
Rules

Questions so far?
Rules for Patches

- Reflexive: patches change themselves

```lisp
ask patches
[ set grass grass + 1 ]
```
Change other patches

ask patches in-radius 1
[ set grass 0.1 * my-grass ]
Rules for Patches

► Change turtles

ask turtles-here
[ set sick? true
set color blue ]
Rules in Summary

Session 1b: July 5
Rules in Summary

- $T_{\text{self}}$
- $P_{\text{self}}$
- $T$-to-$T$
- $P$-to-$P$
- $T$-to-$P$
- $P$-to-$T$
Session 1b: July 5

5th am Intro
5th pm NetLogo
7th am Theory
7th pm Modeling
12th am Emergence
12th pm Systems
14th am Design
14th pm Experiments
19th am Criticisms
19th pm Validation
21st am Smart Agents
21st pm Networks
26th Applications
28th Presentations
The Computer
The Computer

5th am Intro
5th pm NetLogo
7th am Theory
7th pm Modeling
12th am Emergence
12th pm Systems
14th am Design
14th pm Experiments
19th am Criticisms
19th pm Validation
21st am Smart Agents
21st pm Networks
26th Applications
28th Presentations

Session 1b: July 5
Interface Elements

NetLogo v. 4

Session 1b: July 5
Elements of NetLogo

- Interface tab
- Information tab
  - “Info”
- Procedures tab
  - “Code”
Interface Tab

Session 1b: July 5

5th am Intro
5th pm NetLogo
7th am Theory
7th pm Modeling
12th am Emergence
12th pm Systems
14th am Design
14th pm Experiments
19th am Criticisms
19th pm Validation
21st am Smart Agents
21st pm Networks
26th Applications
28th Presentations
Procedures Tab

```netlogo
; Sheep and wolves are both breeds of turtle.
breed [sheep n-sheep] ;; sheep is its own plural, so we use "a-sheep" as the singular.
breed [wolves n-wolves]
turtles-own [energy] ;; both wolves and sheep have energy
patches-own [countdown]

to setup
  clear-all
  ask patches [ set pcolor green ]
  ;; check GRASS2 switch.
  ;; if it is true, then grass grows and the sheep eat it
  ;; if it false, then the sheep don't need to eat.
  if grass? [ ask patches [ set countdown random grass2-growth-time ;; initialize grass grow clocks randomly set pcolor one-of [green brown] ] ]
  set-default-shape sheep "sheep"
  create-sheep initial-number-sheep ;; create the sheep, then initialize their variables
  [ set color white set size 1.5 ;; easier to see set label-color blue - 2 set energy random (2 * sheep-gain-from-food) setxy random-xor random-yor ]
  set-default-shape wolves "wolf"
  create-wolves initial-number-wolves ;; create the wolves, then initialize their variables
  [ set color black set size 1.5 ;; easier to see set energy random (2 * wolf-gain-from-food) setxy random-xor random-yor ]
  display-labels
```
WHAT IS IT?

This model explores the stability of predator-prey ecosystems. Such a system is called unstable if it tends to result in extinction for one or more species involved. In contrast, a system is stable if it tends to maintain itself over time, despite fluctuations in population sizes.

HOW IT WORKS

There are two main variations to this model.

In the first variation, wolves and sheep wander randomly around the landscape, while the wolves look for sheep to prey on. Each step costs the wolves energy, and they must eat sheep in order to replenish their energy - when they run out of energy they die. To allow the population to continue, each wolf or sheep has a fixed probability of reproducing at each time step. This variation produces interesting population dynamics, but is ultimately unstable.

The second variation includes grass (green) in addition to wolves and sheep. The behavior of the wolves is identical to the first variation; however this time the sheep must eat grass in order to maintain their energy - when they run out of energy they die. Once grass is eaten it will only regrow after a fixed amount of time. This variation is more complex than the first, but it is generally stable.

The construction of this model is described in two papers by Wilensky & Reisman referenced below.

HOW TO USE IT

1. Set the GRASS? switch to TRUE to include grass in the model, or to FALSE to only include wolves (red) and sheep (white).
2. Adjust the slider parameters (see below) or use the default settings.
Graphical User Interface Elements

- Buttons
- Sliders
- Switches
- Choosers
- Monitors and plots

Session 1b: July 5
The NetLogo “World”

- A matrix
  - 33 x 33 square patches
  - Cartesian coordinates

- World is customizable
  - Size, Scale
  - Shape of patch (square versus hexagon)
  - Wrapping (none, top-bottom, left-right, all)
  - Origin
NetLogo Interface

Questions?
NetLogo Exercises

- Learning outcomes
  - Creating agents
  - Using the command center
  - Thinking about NetLogo syntax
  - Thinking about patches
  - "Equifinality" in programming
  - Link code to interface elements (sliders, buttons)
Thinking about syntax

- Nouns (subjects, objects)
- Adjectives (properties, states, variables)
- Verbs (actions, states)
- Rules for relating the two
“Nouns”

- What are the “nouns” (subjects and objects) of our syntax?
- What “parts” of NetLogo are we talking to?
  - “Turtles”
  - “Patches”
  - “Agentset”
  - “Global” variables
Agents that act
Objects that receive actions
Three types in NetLogo
- Turtles
- Patches
- Global Variables
Agents

- Basic actors in the system
  - People in a society
  - Cars in a traffic jam
  - Nodes/links in a network
  - Firms in a supply chain
  - Consumers in a market

- NetLogo calls them “turtles”
Basic element of space
Agentset

- The entire set of turtles
  - All firms in a market
  - All people in a riot
- The entire set of patches
- Subsets of turtles or patches

let yellow-turtles turtles with [ color = yellow ]
Agent Instructions

➢ Recall six types

- \( T_{self} \)
- \( P_{self} \)
- \( T\text{-to-}T \)
- \( P\text{-to-}P \)
- \( T\text{-to-}P \)
- \( P\text{-to-}T \)
Agent Instructions

- Some rules require additional variables
- Agents may have
  - Age
  - Wealth
  - Gender
  - Ethnicity
Properties in NetLogo

- **Built-in properties**
  - Color
  - Shape
  - Size
  - Location \((x, y)\)
  - Text label

- **Add-on properties**

  ```
  turtles-own [ wealth age ]
  patches-own [ elevation ]
  ```
patches-own [ sugar ]
turtles-own [ vision endowment metabolism ]

to setup
  ca
  ask patch-at 12 12 [ set sugar 5000 ]
  ask patch-at -12 -12 [ set sugar 5000 ]
  repeat 100 [ diffuse sugar 0.5 ]
  ask patches [ set pcolor scale-color yellow sugar 0 50 ]
  create-agents
end

to create-agents
  crt 200
  ask turtles
    [ set shape "circle"
    set metabolism 1 + random 3
    set vision 1 + random 5
    setxy random world-width random world-height
    ifelse (random 1000 < 500)
      [ set color red ]
      [ set color blue ]
    ]
end
Global variables

- Properties of the model that do not “belong” to turtles, patches

- Examples
  - A “clock” variable
  - How many turtles have “died”?
To review

- Agents and agentsets
  - Properties ("adjectives")
- Patches
- Global variables
Questions?
“Verbs” — how we tell agents, patches or NetLogo to take action

Three key expressions:

- “ask . . .”
- “set . . .”
- “to do . . . end”
Have an agent or agentset do something

``` NetLogo
ask turtle 32 [ set shape "person" ]

ask turtles [ forward 1 right 20 ]

ask one-of patches [ set plabel "I am a patch"]

ask patches [ set pcolor gray ]
```

"ask . . ."

Session 1b: July 5
To set or change the value of a variable

```plaintext
ask turtle 0 [ set shape "person" ]

ask one-of patches [ set plabel "I am a patch"]

ask patches [ set pcolor gray ]

set clock clock + 1
```
Bounds a set of related commands or instructions

A kind of “paragraph”: We call it a “procedure”

to change-places
  ask turtles [ 
    ifelse (count turtles-on patch-ahead 1 > 0) [ 
      set heading heading + random 45 - random 45 [ 
    [ forward 1 ]
    ] ]
  end

“look ahead one space and count the number of turtles there”

“if someone is there, wiggle a bit to change directions”

“otherwise, move forward one spot”
NOTE: Syntax follows

- Boolean conventions “or” “and” “not”
- Logical expressions “if” “if else”

See handout
Power of NetLogo

➤ You can ask any or all agents to do something

➤ An agent or patch can ask any or all agents to do something

➤ “any to all”
Agent-based models susceptible to exponential growth in operations

Example:

```
ask patches
  [set my-neighbors count turtles in-radius 20]
```

\[(33 \times 33 \text{ patches}) \times (20 \times 20 \times 3.14159) = 1.4 \text{ million operations per step in time}\]
Runaway models

» Use the “halt” menu item
NetLogo Help

- Primitives dictionary
- Code examples
- Model library
- Tutorials
- User’s group
Questions?
On Thursday

- Complete Tutorials 1-3 in NetLogo library
- Complex Systems Theory