Introduction to Agent-Based Modeling

Continuing Holfield and McCoy Model

A Practical Guide to Building Models in Netlogo

Creating the Hatfield and McCoy Model

What Are We Modeling?

Can Inter-Group Marriage End Inter-Group Conflict?
Two Feuding Factions (the Hatfield and McCoy Families)
Rules for Marriage, Fighting, Birth, and ...?
Measure(s) of Conflict Level and Tolerance
Visualization(s) of the System Dynamics

Agent Behavior: Actions Depend on Types

Create Conditionals for All 21 Types of Agents (nested ifelse)
Use Systematic Properties to Limit Necessary Rules
Still Need Nesting, Consider Optimal Nesting Order
Write a Separate Method for Each Behavior
Take Baby Steps: Minimize Change between Runnable Versions

A Practical Guide to Building Models in Netlogo

Creating the Hatfield and McCoy Model

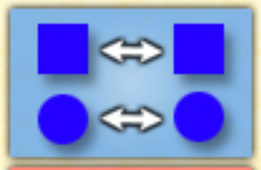
Improving Code: Condensing and Refining

```
to setup2
  ca
  create-custom-turtles population
    setxy random-xcor random-ycor
    ifelse random 100 > 50
      [ set breed hatfields
        set color blue
        set alignment 0.0 ]
      [ set breed mccoys
        set color red
        set alignment 9.0  ]
    set sex random 2
    set age random 10
    ifelse sex = 0
      [set shape "circle"]
      [set shape "square"]
    set label sex
  ]
end
```

- Creates Turtles All at Once
- Set Ratio by Making This a Parameter
- Breed Properties Set Here
- All-Turtle Properties Set Here
- Conditional Properties Set with ifelse
- Always Check Your Work

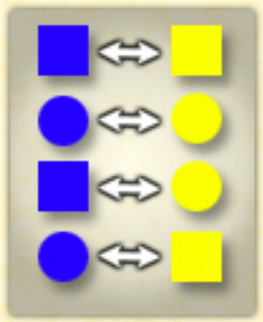
A Practical Guide to Building Models in Netlogo

Agent Behavior: What Are the Effects of Interaction?



Both Alignments -1 (min 0)

Both Alignments +1 (max 9)



What to Do?
Average Alignments, Birth, Fight

Average Alignments

Birth (and Alignment Adjustment)

Fight! (Probability of Death)

Intergroup Marriage:
Birth and Average Alignments

A Practical Guide to Building Models in Netlogo

Creating the Hatfield and McCoy Model

Improving Code: Condensing and Refining

```
to go2
  ask turtles [
    rt random 90 - 45
    fd 1
    without-interruption [
      let agent1 self
    if any? turtles in-radius radius [
      let agent2 one-of turtles in-radius radius
      ;;rule for making babies
      if sex-of agent1 = 0 and sex-of agent2 = 1 [
        let baby-alignment find-average-alignment agent1 agent2
        make-baby baby-alignment
        ]
      ;;rule for fighting behavior
      if (sex-of agent1 = sex-of agent2) [
        ifelse (breed-of agent1 = breed-of agent2)
          [ reinforce-alignment agent1 agent2 ]
          [ fight agent1 agent2 ]
      ]
    ]
    ]
  ]
  ;;to prevent run-away models that freeze your computer
  if count turtles > 500 [stop]
end
```

- Simulates Simultaneous Updating
- Temporarily Store the Activated Agent
- Eliminate "nobody" and "agent/agentset" errors
- Can't Use != for Babies. Why?
- Method/Function takes inputs
- Nested Conditions to Minimize Code
- Model/Measure Interaction

A Practical Guide to Building Models in Netlogo

Creating the Hatfield and McCoy Model

Improving Code: Condensing and Refining

```
;;rule for making babies
if sex-of agent1 = 0 and sex-of agent2 = 1 [
  let baby-alignment find-average-alignment agent1 agent2
  make-baby baby-alignment
  ]
;;rule for fighting behavior
if (sex-of agent1 = sex-of agent2) [

to-report find-average-alignment [agent1 agent2]
  let average-alignment ((alignment-of agent1 + alignment-of agent2) / 2 )
  report average-alignment
end

to make-baby [new-alignment]
  hatch 1 [
    set alignment new-alignment
    set sex random 2
    ifelse sex = 0
      [set shape "circle"]
      [set shape "square"]
    set label sex
  ]
end
```

- Call Outside Functions to Simplify Code and Improve Reuse
- Can Perform Action or Calculation
- Can Take Multiple Inputs of Different Kinds