THEORETICAL MODELING, ICPSR SUMMER II 2024 HOMEWORK 1

Ziyuan Gao

July 20 2024

1 Simple Economy

1.1

Algorithm 1 Simple Economy

```
setup:
create 500 turtles:
for each turtle in turtles do
    turtle.wealth ← 100
    turtle.shape ← "circle"
    turtle.color ← "green"
    turtle.size ← 2
    // Position turtles based on wealth
    turtle.xcor ← turtle.wealth
    turtle.ycor ← random value within the y-coordinate range
end for

go:
for each turtle in turtles do
    if turtle.wealth >0 then
        turtle.transact()
    end if
    if turtle.wealth ≤ max-pxcor then
        turtle.xcor ←—turtle.wealth
    end if
end for

transact: //give a dollar to another turtle
turtle.wealth ←turtle.wealth + 1
oneoftheotherturtle.wealth ←one of the other turtle.wealth - 1

top-10-pct-wealth:
report the total wealth of the top 10

bottom-50-pct-wealth:
report the total wealth of the bottom 50
```

1.2

From Boltzmann-Gibbs law: Any set of interacting agents that exchange a conserved quantity that cannot become negative will always result in such an exponential distribution. In the Simple Economy model, wealth is the conserved quantity being exchanged among agents. Each agent can give away wealth but cannot accumulate it beyond what is initially distributed. The wealth cannot go below zero for any agent, leading to fewer agents holding large amounts of wealth.

1.3

To increase wealth inequality:

1. Enhance wealth accumulation for wealthier turtles
2. Add transaction bias to wealthier turtles, for example, top 10 rich turtles are free of transaction to others. Make rich richer.

To decrease wealth inequality:

1. Redistribute wealth among turtles to reduce the disparity. i.e: Tax wealthier turtles.
2. Set caps on wealth accumulation to balance wealth more evenly across all turtles

I tested on Redistribute wealth among turtles by taxing top-10 rich turtles and distribute the tax to the bottom-50 turtles in Netlogo.

```
to go
  ;; transact and then update your location
  ask turtles with [ wealth > 0 ] [ transact ]

    ; Calculate total wealth of the top 10% of turtles
  let top-10pct-turtles max-n-of (count turtles * 0.10) turtles [ wealth ]
  let top-10pct-wealth sum [ wealth ] of top-10pct-turtles

  ; Calculate the total tax amount
  let tax-amount top-10pct-wealth * tax-rate

  ; Determine the bottom 50% of turtles
  let pool-turtles min-n-of (count turtles * 0.50) turtles [ wealth ]

  ; Calculate amount to redistribute to each turtle in the bottom 50%
  let num-pool-turtles count pool-turtles
  let redistribution-amount tax-amount / num-pool-turtles

  ask top-10pct-turtles[set wealth wealth - (wealth * tax-rate)]

  ; Redistribute wealth to the bottom 50% of turtles
  ask pool-turtles [set wealth wealth + redistribution-amount]

  ;; prevent wealthy turtles from moving too far to the right -- that is, outside the view
  ask turtles [ if wealth <= max-pxcor [ set xcor wealth ] ]

  tick
end
```

Figure 1: Code for taxing top-10 rich turtles

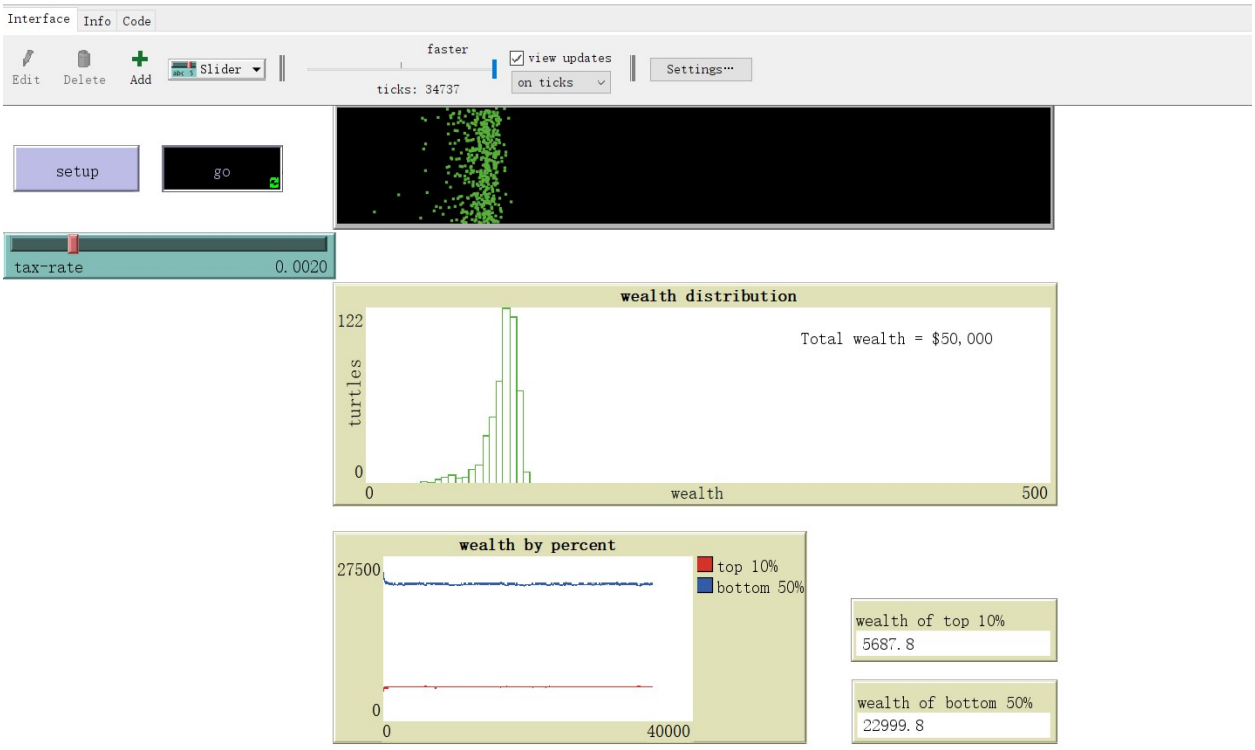

Figure 2: Netlogo GUI after 30000 simulation

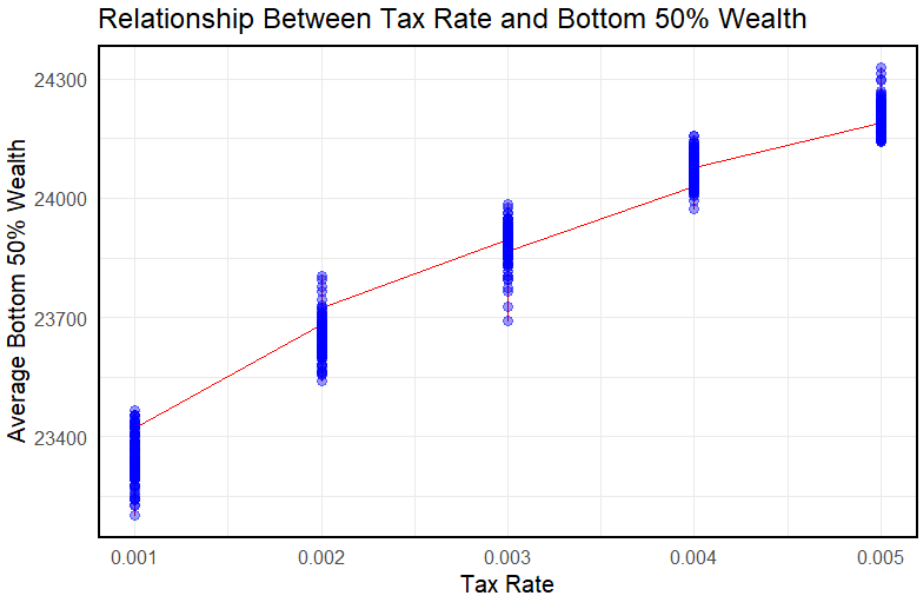

Figure 3: Relationship Between Tax Rate and Bottom 50 Wealth

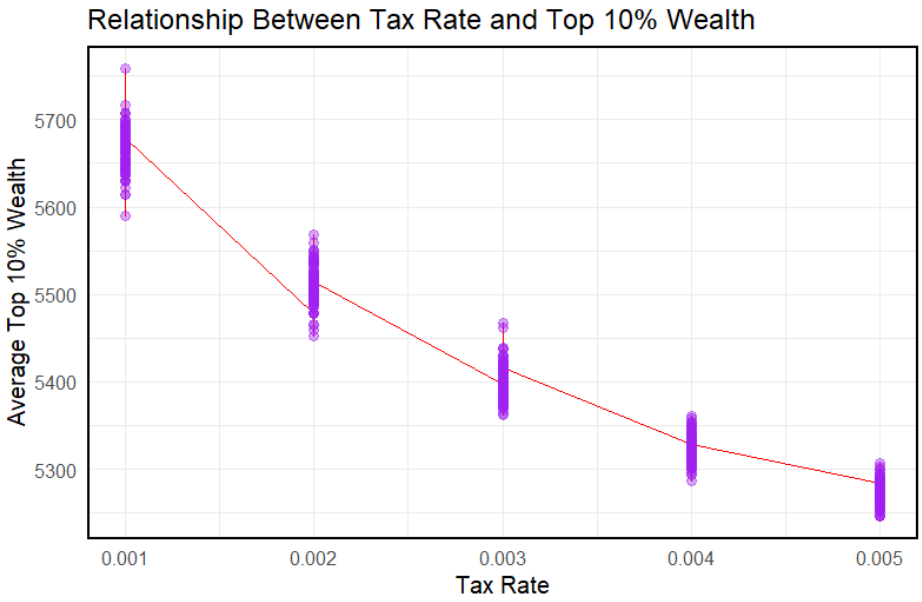

Figure 4: Relationship Between Tax Rate and Top 10 Wealth

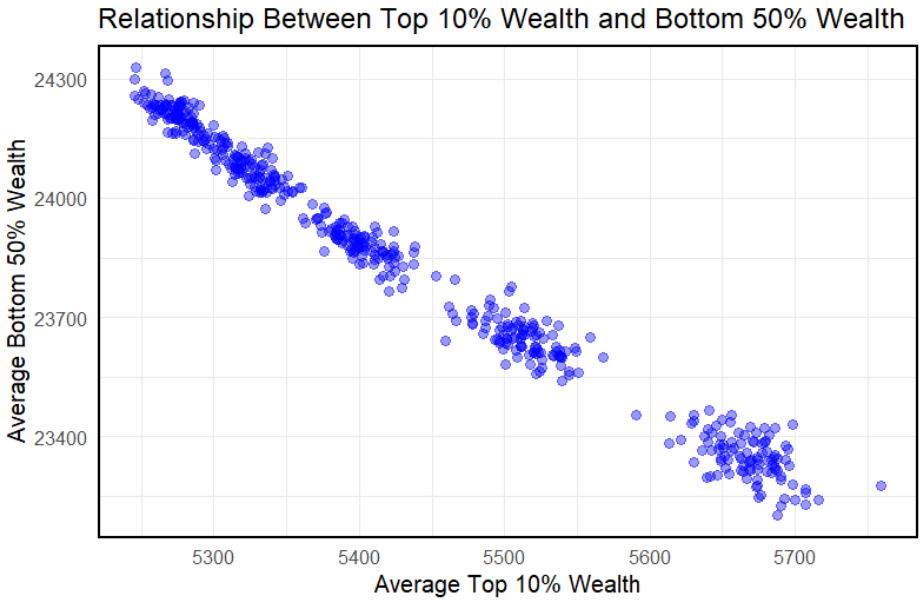

Figure 5: Relationship Between Top 10 Wealth Bottom 50 Wealth

2 Comparing segregation metrics

2.1

```
;;set turtles to unhapy if proportion of similar neighbors is below threshold, otherwise set to happy
to update-turtles
  ask turtles [
    ;; in next two lines, we use "neighbors" to test the eight patches
    ;; surrounding the current patch
    set uniform 1 ;; Initialize the uniform attribute to 1 for each turtle
    let similar-nearby count (turtles-on neighbors)  with [ color = [ color ] of myself ]
    let total-nearby count (turtles-on neighbors)
    ifelse (total-nearby = 0)
    [set prop-similar-neighbors 1]  ;;always happy if alone.
    [set prop-similar-neighbors (similar-nearby / total-nearby)
      if(prop-similar-neighbors < 1)
      [set uniform 0] ;; Set uniform to 0 if not all neighbors are similar
    ]
    set happy? (prop-similar-neighbors >= similarity-threshold)
  ]

end

to update-globals
  let similar-neighbors sum [ prop-similar-neighbors ] of turtles
  let total-uniform sum [uniform] of turtles ;; Sum of all turtles' uniform attributes
  set average-similarity (similar-neighbors / count turtles)
  set unhappiness (count turtles with [ not happy? ]) / (count turtles)
  set prop-uniform  (total-uniform / count turtles) ;; Proportion of turtles with uniform = 1
end
```

Figure 6: Comparing segregation code

2.2

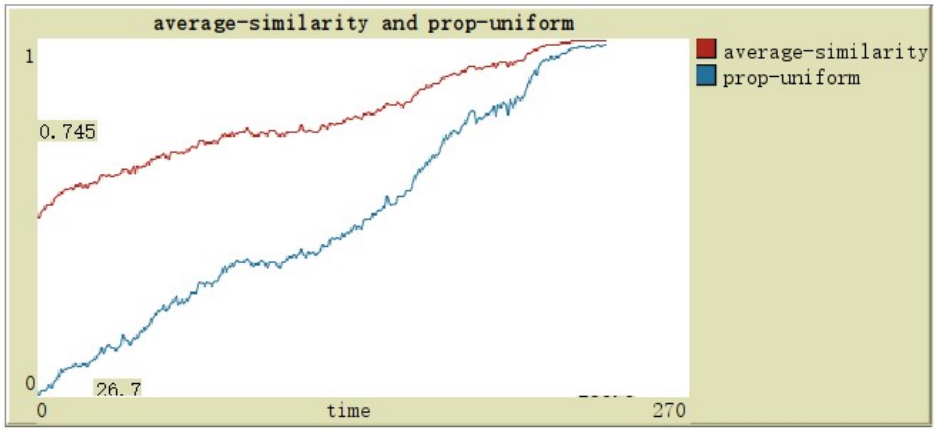

Figure 7: average-similarity and prop-uniform

2.3

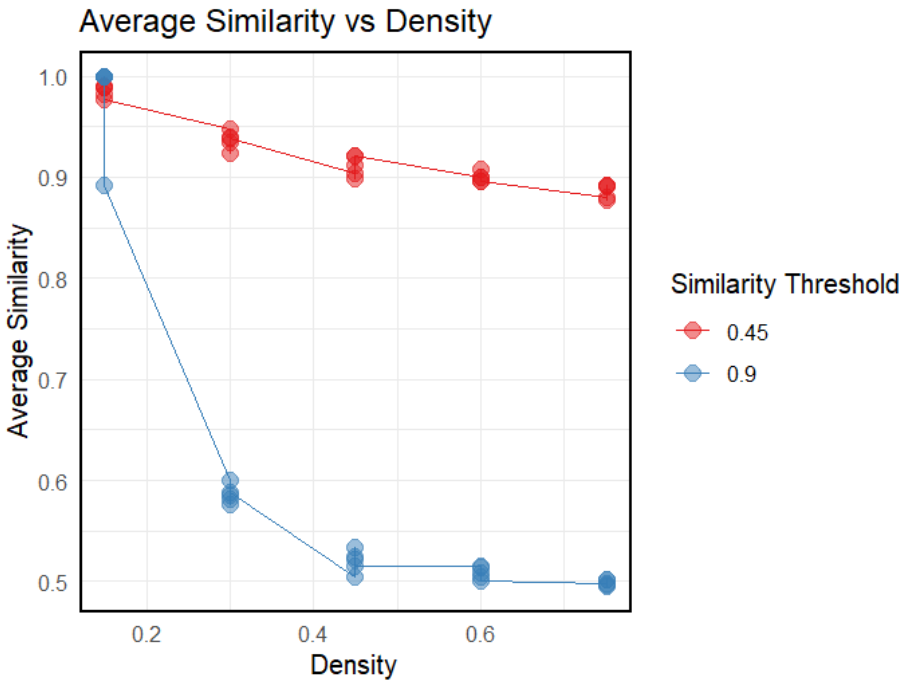

Figure 8: Average Similarity vs Density

Both measures of segregation show, as the similarity threshold increases, segregation generally increases. Higher density (0.90) tends to lead to higher segregation compared to lower density (0.45).

3 Vaccine efficacy

3.1

$$\Delta I = \tau I \left(1 - \frac{I}{N}\right) ((1 - V) + V(1 - e)) - \gamma I$$

when the outbreak is new

$$\frac{I}{N} \approx 0$$

$$\Delta I = \tau I \left((1 - V) + V(1 - e)\right) - \gamma I$$

Infection WON'T spread if

$$\tau I \left((1 - V) + V(1 - e)\right) - \gamma I \leq 0$$

$$\tau I \left(1 - Ve\right) - \gamma I \leq 0$$

$$\tau \left(1 - Ve\right) \leq \gamma$$

$$\frac{\tau}{\gamma} \left(1 - Ve\right) \leq 1$$

$$R_0 (1 - Ve) \leq 1$$

$$V \leq \frac{1}{e}\left(1 - \frac{1}{R_0}\right)$$

when the outbreak is new, the threshold vaccination rate for herd immunity is:

$$V^* = \frac{1}{e}\left(1 - \frac{1}{R_0}\right)$$

3.2

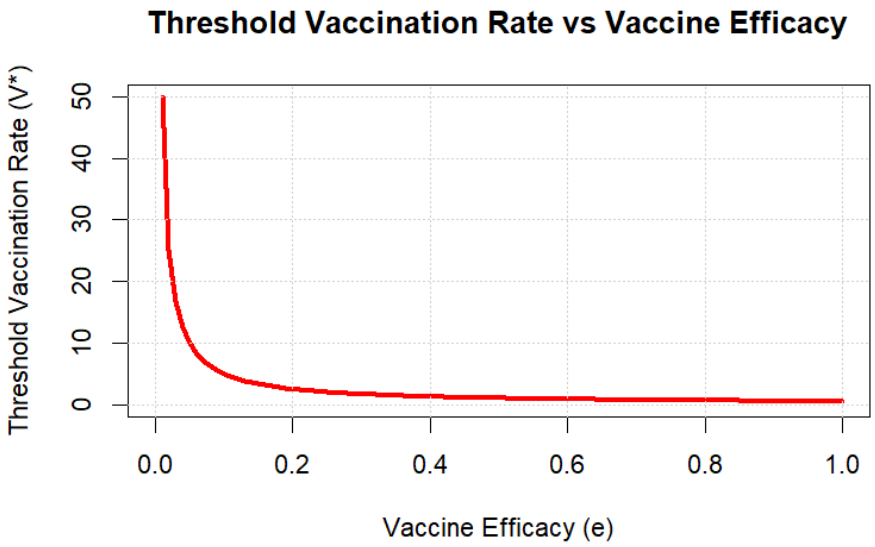

Figure 9: Threshold Vaccination Rate vs Vaccine Efficacy

The resulting curve shows that as vaccine efficacy decreases, the required vaccination rate for herd immunity increases. This means that with a less effective vaccine, a larger portion of the population needs to be vaccinated to achieve herd immunity. Conversely, with a highly effective vaccine, fewer people need to be vaccinated to stop the spread of the disease.

4 Vaccine connectivity

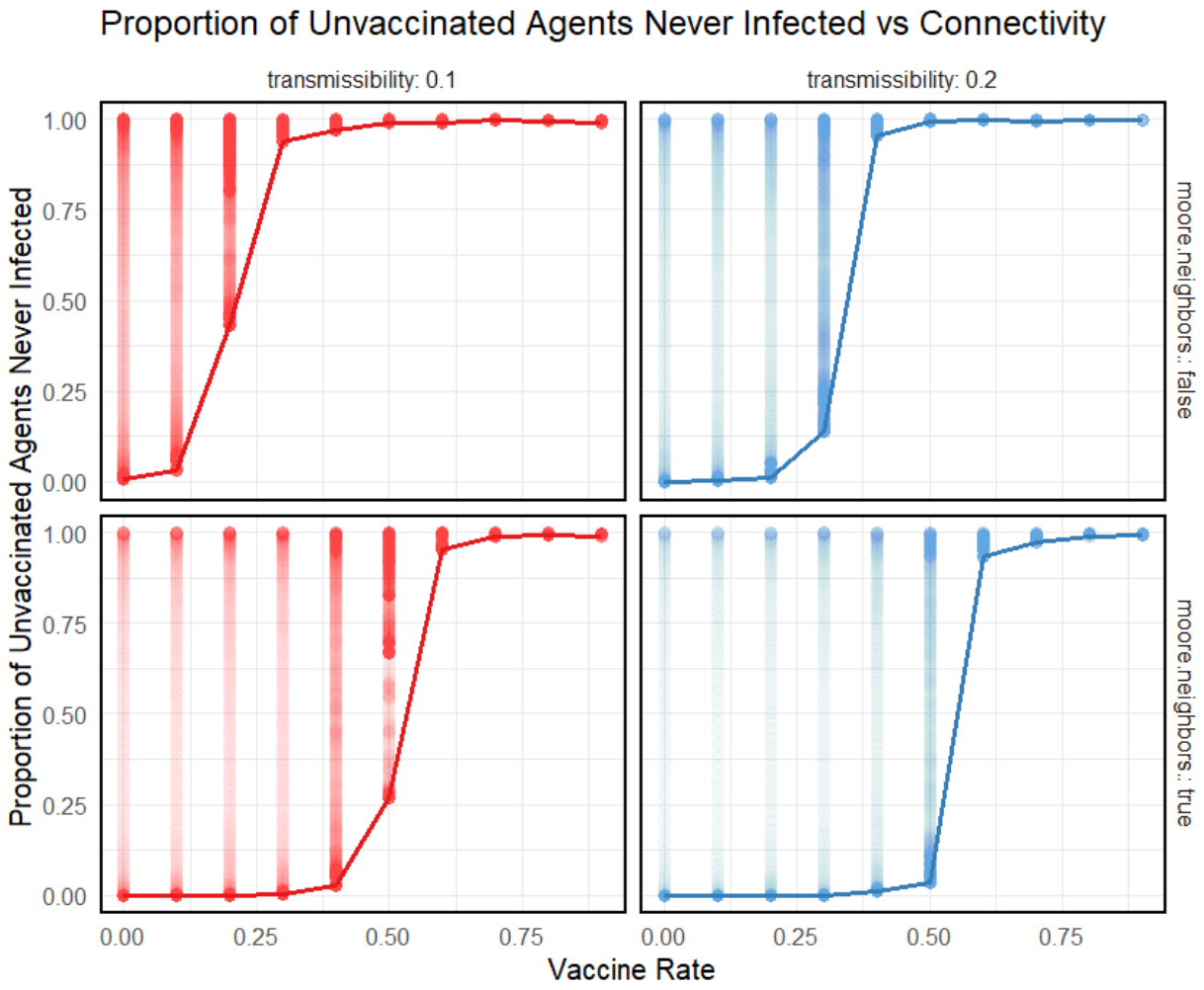

Figure 10: Proportion of Unvaccinated Agents Never Infected vs Connectivity

It can be concluded that the higher connectivity leads to lower proportion of unvaccinated agents that never been infected. Higher connectivity requires a higher vaccine rate to reach herd immunity.

5 Space takes longer

5.1

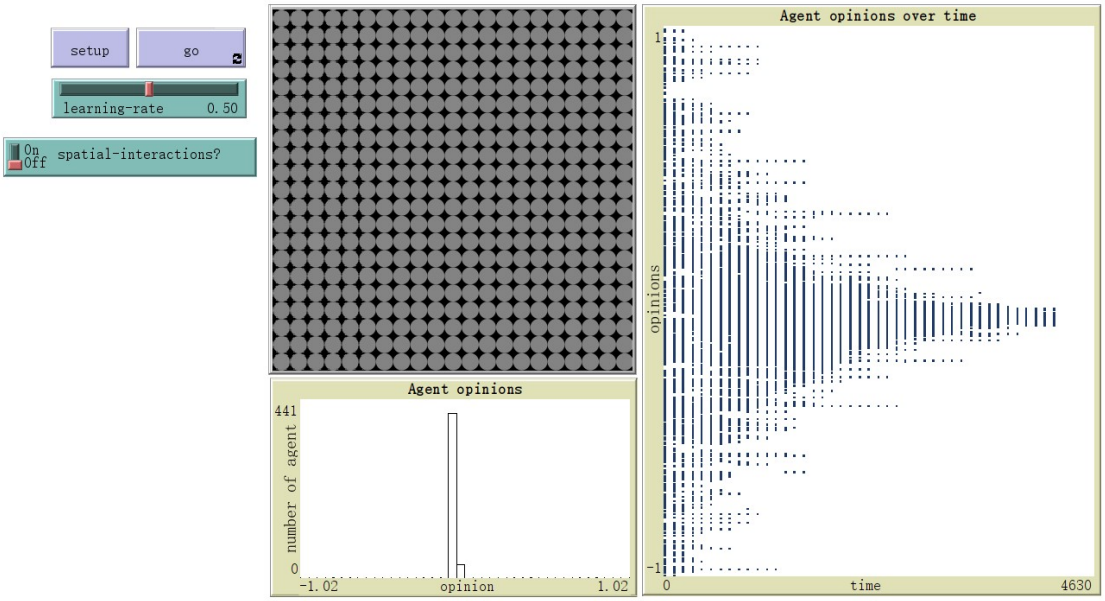

Figure 11: spatial interaction GUI

```
to go
  ;; a random agent chooses another agent for a potential interaction
  ask one-of turtles [
    let x1 opinion ;;my opinion
    let other-turtle one-of other turtles ;;choose interaction partner at random

    if spatial-interactions? ;;restrict interactions to spatial neighbors
    [set other-turtle one-of other turtles-on neighbors4]

    let x2 [opinion] of other-turtle ;;other guy's opinion
    ;;do they have an interaction and become more similar?
      let x1-new (x1 + learning-rate * (x2 - x1))
      let x2-new (x2 + learning-rate * (x1 - x2))
      set opinion x1-new
      ask other-turtle [ set opinion x2-new]
  ]
  update-colors

  ;;define consensus within certain threshold
  let max-opinion max [opinion] of turtles
  let min-opinion min [opinion] of turtles
  let opinion-difference max-opinion - min-opinion
  ;;stop when consensus reached
  if opinion-difference < 0.05[
    user-message "Consensus reached!"
    stop
  ]
  tick
end
```

Figure 12: spatial and consensus code

5.2

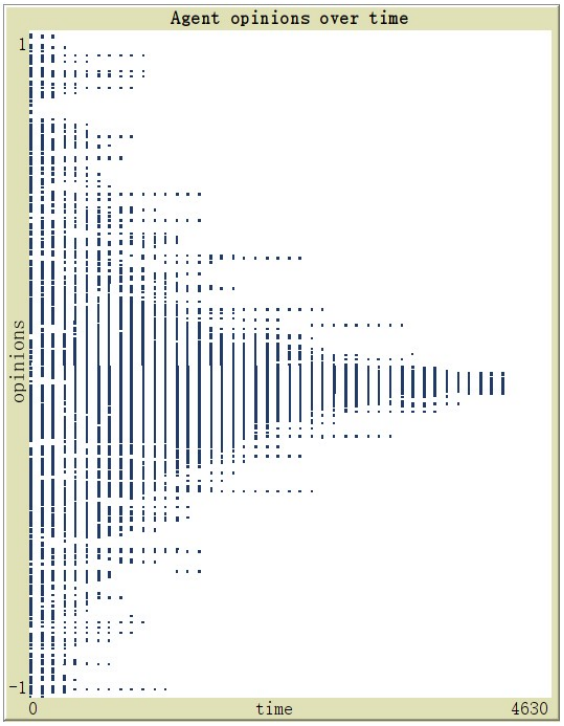

Figure 13: Simulation of unconstraining interaction reaching consensus

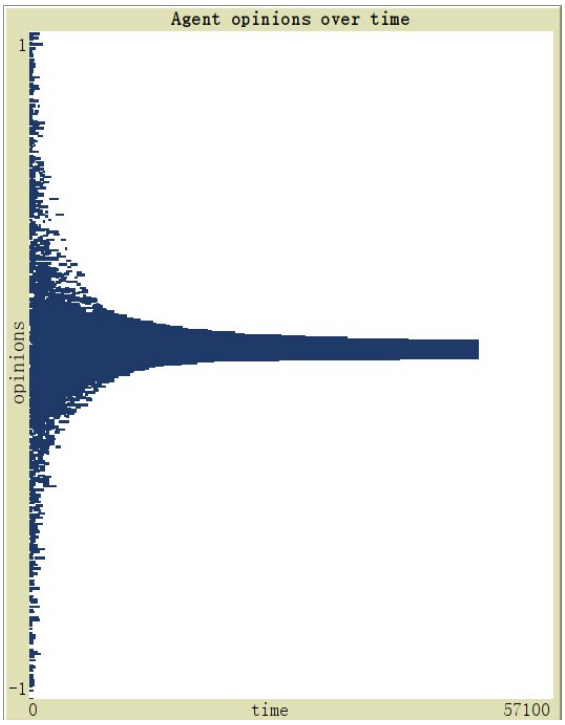

Figure 14: Simulation of constraining interaction reaching consensus

Unconstraining interaction takes 4630 to reach consensus, while constraining interaction takes 57100 to reach consensus. Constraining interactions in local space means the system takes longer to reach consensus than when the population is well mixed.

5.3

Figure 15: Time to Consensus with different Learning Rate

When the learning rate is set to 1, each agent fully adopts the opinion of their interaction partner in just one step. This can cause two main issues: Oscillation: Agents may rapidly switch opinions back and forth, leading to constant changes without settling into a stable consensus. No Consensus: The model may never reach a final consensus because the opinions keep fluctuating too much to converge. In conclusion, a learning rate of 1 can make the simulation unstable, preventing the agents from ever agreeing on a single opinion.

6 DIY

Hypothesis: Exploring the built environment under the economy model of reducing monopoly

This study aims to understand how economic competition can be reflected in urban design,infrastructure, and spatial utilization.

- **Economic Models to Consider:**
  - Monopoly vs. Perfect Competition: Compare scenarios where a few large firms dominate versus many small, competitive firms. Use models like Cournot or Bertrand competition to analyze how firms' pricing and production decisions affect the built environment.
  - Urban Economic Models: Incorporate concepts from urban economics, such as the Alonso-Muth-Mills model, to study land use and economic rent.
- **Decomposition of the System:**
  - **Components:**