Multiple Languages change

Kehang Li

Model Phenomenon

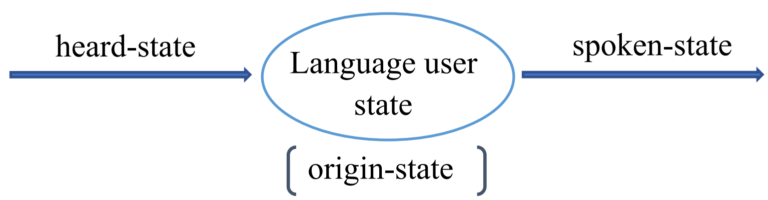
Multiple languages change model explores the properties of language users and the structure of their social networks can affect the course of languages change. This model is the revised version of "Language change" model from NetLogo library. Over time and iterations, language users interact with connected neighbors, language spread and change via communication.

Model Components

1. Agent properties

In this model, there are three linguistic variants "state 0", "state 1", "state 2", that generated by grammar 0, grammar 1, grammar 2 respectively. Each agent (language user) has 4 linguistic properties: "spoken-state", "heard-state", "origin-state", "state".

The "spoken-state" refers to the linguistic variant that agent speaks by passing an utterance using corresponding grammar to the neighbors connected in the network. The "heard-state" refers to the linguistic variant of utterance that was heard by receiving agent and passed by connected neighbors. The "origin-state" is the initial grammar of agent at setup before network connection and language change iterations. The "state" indicates the status of agent's inherited linguistics variants after change iterations.

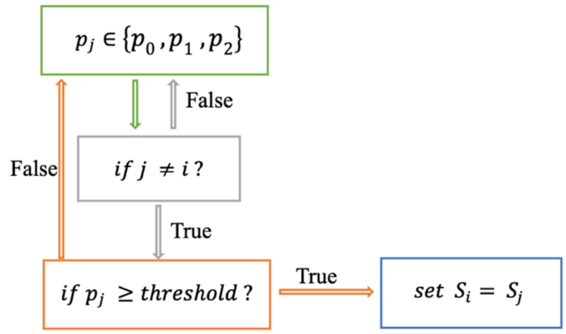


2. Updating algorithm

The updating algorithm of network in this model is defined as competition process of three grammars, including how the linguistic properties of agents change and how the language users hear, learn and speak the language. There are three modes of linguistic variants competition in the network in this model: "Individual", "Threshold", "Reward".

Individual: Agents randomly choose one of their connected neighbors and adopt that neighbor's grammar, then agent "state" change to the chosen neighbor's "spoken-state".

Threshold: Agents adopt the grammar other than its current state when the proportion of agents connected neighbors already using this grammar exceed the threshold value. Assume p_0, p_1, p_2 represent the connections proportion of grammar 0, grammar 1, grammar 2 respectively, S_i represents agent current "state", $i \in \{0, 1, 2\}$.



Reward: Agents update their probabilities of using the grammars at each iteration. Assume $prob_0$, $prob_1, prob_2$ represent the probability of using grammar 0, grammar 1, grammar 2 respectively for an agent, if an agent hears an utterance from grammar 2, the $prob_2$ is increased and agent will more likely to use grammar 2 in next iteration. In this mode, value of agent "state" is continuous instead of integer 0, 1, 2, using z to represent the agent continuous "state". Each agent has preference bias of its origin-state corresponding grammar in spoken-state, based on the output of logistic function. In each iteration, if agent hear grammars other than its origin-state, its "state" value would be biased away from the "origin-state" by reward algorithm.

Table 1. Reward algorithm and probabilistic models for state iteration

Origin-state	State (z = state, γ = 0.01)		Spoken-state (β = 100α + 10)		
	heard-state = origin-state	heard-state ≠ origin-state	$prob_0$	$prob_1$	$prob_2$
0	z (1- γ)	γ + z (1- γ)	$\frac{1}{1+e^{-(\beta(2-z)-1.5\beta)}}$	$\left(1-\frac{1}{1+e^{-(\beta(2-z)-1.5\beta)}}\right)*0.5$	
1	z + γ (1- z)	z (1- γ)	$\frac{1}{1+e^{-(\beta(1-\lvert z-1\rvert)-0.5\beta)}}$	$\left(1-\frac{1}{1+e^{-(\beta(1-\lvert z-1\rvert)-0.5\beta)}}\right)\lvert 1-round[z]\rvert$	
2	z + γ (1- z)	z (1- γ)	$\frac{1}{1+e^{-(\beta z-1.5\beta)}}$	$\left(1-\frac{1}{1+e^{-(\beta z-1.5\beta)}}\right)*0.5$	

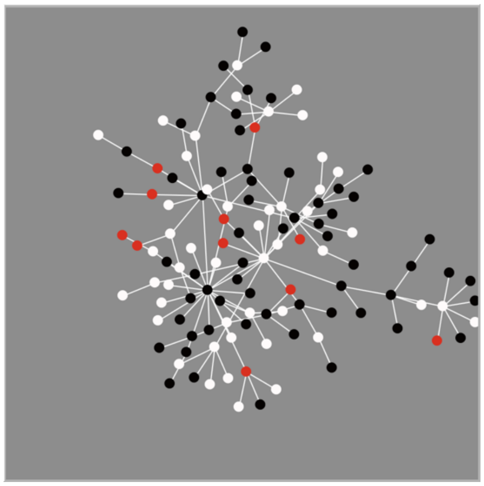
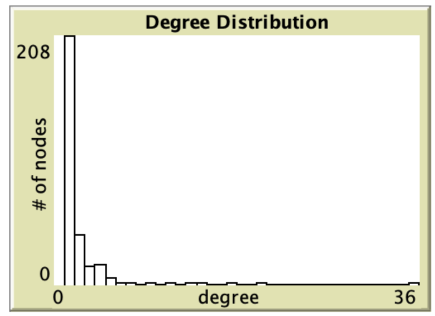
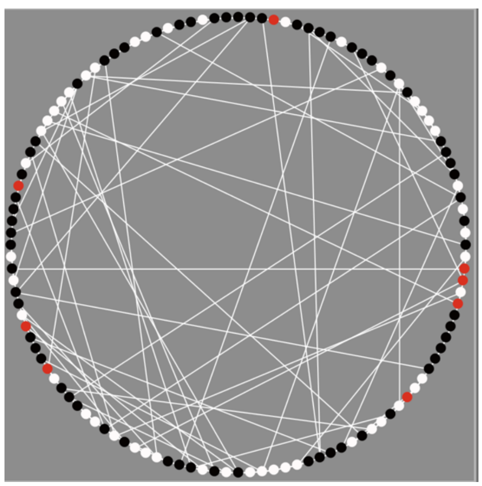
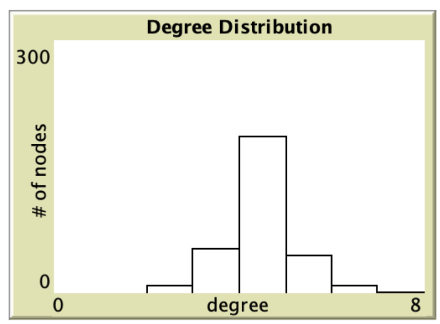
3. Network structure

There are two different process options to set up model network structure: Preferential attachment and Small world.

Preferential attachment: In this process, agents are added into the network one by one at each step. A new agent randomly chooses an existing agent to connect with the preferential bias that they prefer to select agents who are already well connected. More specifically, agent's probability of being connected by other agents is directly proportional to the numbers of its existing connections. The connections number of an agent is called "degree" in the model.

Small world: In this process, agents are generated in a circle spatial location, each agent begins with a simple network that it is connected with two neighbors on either side. Then re-create the initial network and visit all the connections, rewire each link independently with current rewiring probability. Rewiring connection means that the link is disconnected from one of its nodes and randomly connected to another node anywhere in the network.

Table 2. NetLogo visualization of network and agent degree distribution (N = 300) in a random simulation by Multiple language change model

Network	Visualization	Degree distribution
Preferential attachment		
Small world		

4. Parameters

SETUP to generate a new network based on N, percent-grammar-1 and percent-grammar-2, and network-type.

GO allow all language users to “speak” and “listen” according to the algorithm in the UPDATE-ALGORITHM drop down menu, and keep simulation running continuously.

N slider determines the number of agents to be included in the network population.

PERCENT-GRAMMAR-1 determines the proportion of language users in the network who will be initialized to use grammar 1.

PERCENT-GRAMMAR-2 determines the proportion of language users in the network who will be initialized to use grammar 2, the upper limit of the slier is (100 - percent-grammar-1). The remaining agents will be initialized to use grammar 0.
THRESHOLD-VAL slider set the proportion of threshold and applies only for threshold updating algorithm.
ALPHA slider applies only for reward updating algorithm. It represents a bias in favor of its origin-state corresponding grammar, probabilities of selecting origin-state grammar are pushed to extremes by alpha through logistic function.
REWIRE-ALL button only applies for small world network type, to re-create the initial network and visit all the connections, rewire each link independently with current rewiring probability.
REWIRING-PROBABILITY slider determines the probability that a link will get rewired.

Code Modification and Implementation

Table 3. Comparison between origin code and modified code

<table>
<tr><th></th><th>Origin model code</th><th>Modified model code</th></tr>
<tr><td>setup</td><td><pre>to setup
  clear-all
  set-default-shape nodes "circle"
  ask patches [set pcolor gray]
  repeat num-nodes [make-node]
  distribute-grammars
  create-network
  repeat 100 [layout]
  reset-ticks
end</pre></td><td><pre>to setup
  clear-all
  set-default-shape turtles "circle"
  ask patches [set pcolor gray]
  if network-type = "Preferential Attachment"
  [repeat N [make-node]
    create-network
    repeat 300 [layout]]

  if network-type = "Small world"
  [repeat N [make-node]
    layout-circle (sort turtles) max-pxcor - 1
    wire-them]
  distribute-grammars
  set sum-of-states sum [state] of turtles
  plot-degree
  reset-ticks
end</pre></td></tr>
<tr><td>distribute-grammars</td><td><pre>to distribute-grammars
  ask nodes [set state 0]
  ;; ask a select proportion of people to switch to 1
  ask n-of ((percent-grammar-1 / 100) * num-nodes) nodes
    [set state 1.0]
  ask nodes [
    set orig-state state
    set spoken-state state
    update-color
  ]
end</pre></td><td><pre>to distribute-grammars
  ask turtles [set state 0.0]
  ask n-of ((percent-grammar-1 / 100) * N) turtles
    [set state 1.0]
  ask n-of ((percent-grammar-2 / 100) * N) turtles
    [set state 2.0]
  ask turtles [
    set orig-state state
    set spoken-state state
    update-color
  ]
end</pre></td></tr>
<tr><td>update-color</td><td><pre>to update-color
  set color scale-color red state 0 1
end</pre></td><td><pre>to update-color
  set color scale-color red state 0 2
end</pre></td></tr>
<tr><td>Plot-degree</td><td>N/A</td><td><pre>to plot-degree
  let max-degree max [count link-neighbors] of turtles
  set-current-plot "Degree Distribution"
  plot-pen-reset
  set-plot-x-range 0 (max-degree + 1)
  histogram [count link-neighbors] of turtles
end</pre></td></tr>
</table>

speak	to speak ;; node procedure if logistic? [let gain (alpha + 0.1) * 20 let filter-val 1 / (1 + exp (- (gain * state - 1) * 5)) ifelse random-float 1.0 <= filter-val [set spoken-state 1] [set spoken-state 0]] if not logistic? [let biased-val 1.5 * state if biased-val >= 1 [set biased-val 1] ifelse random-float 1.0 <= biased-val [set spoken-state 1] [set spoken-state 0]] end	to speak let Sj random 3 while [Sj = orig-state] [set Sj random 3] ifelse orig-state = 1 [let gain 100 * alpha + 10 let func gain * (1 - abs(state - orig-state)) - 0.5 * gain let filter-val 1 / (1 + exp (- func)) ifelse random-float 1.0 <= filter-val [set spoken-state orig-state] [set spoken-state round state]] [let gain 100 * alpha + 10 let func gain * (2 - abs(state - orig-state)) - 1.5 * gain let filter-val 1 / (1 + exp (- func)) ifelse random-float 1.0 <= filter-val [set spoken-state orig-state] [set spoken-state Sj]] end
listen-threshold	;; Speaking & Listening to listen-threshold ;; node procedure let grammar-one-sum sum [state] of link-neighbors ifelse grammar-one-sum >= (count link-neighbors * threshold-val) [set state 1] [;; if there are not enough neighbors with grammar 1, ;; and 1 is not a sink state, then change to 0 if not sink-state-1? [set state 0]] end	;; Speaking & Listening to listen-threshold ;; node procedure let Sj random 3 while [Sj = state] [set Sj random 3] let S1 random 3 while [(S1 = Sj) or (S1 = state)] [set S1 random 3] let di count link-neighbors with [state = [state] of myself] let dj count link-neighbors with [state = Sj] let d0 count link-neighbors - di - dj let d1 max (list d0 dj) if d1 >= (count link-neighbors * threshold-val) [ifelse d1 = dj [set state Sj] [set state S1]] end
listen [heard-state]	;; Listening uses a linear reward/punish algorithm to listen [heard-state] ;; node procedure let gamma 0.01 ;; for now, gamma is the same for all nodes ;; choose a grammar state to be in ifelse random-float 1.0 <= state [;; if grammar 1 was heard ifelse heard-state = 1 [set state state + (gamma * (1 - state))] [set state (1 - gamma) * state]][;; if grammar 0 was heard ifelse heard-state = 0 [set state state * (1 - gamma)] [set state gamma + state * (1 - gamma)]] end	to listen [heard-state] let gamma 0.01 ifelse orig-state = 0 [ifelse heard-state = orig-state [set state state * (1 - gamma)] [set state gamma + state * (1 - gamma)]][ifelse heard-state = orig-state [set state state + gamma * (orig-state - state)] [set state (1 - gamma) * state]] end
New code added for small world network and new index		

rewire-all	to rewire-all
	set number-rewired 0
	ask links [
	;; whether to rewire it or not?
	if (random-float 1) < rewiring-probability
	[let node1 end1
	if [count link-neighbors] of end1 < (count turtles - 1)
	[let node2 one-of turtles with [(self != node1) and (not link-neighbor? node1)]
	ask node1 [create-link-with node2 [set rewired? true]]
	set number-rewired number-rewired + 1 ;; counter for number of rewirings
	set rewired? true
	]
	]
	if (rewired?)
	[die]
	]
	update-plots
	end	

wire-them	<pre>to wire-them
	  ;; iterate over the turtles
	  let a 0
	  while [a < count turtles]
	  [
	    ;; make edges with the next two neighbors
	    ;; this makes a lattice with average degree of 4
	    make-edge turtle a
	              turtle ((a + 1) mod count turtles)
	    make-edge turtle a
	              turtle ((a + 2) mod count turtles)
	    set a a + 1
	  ]
	end</pre>

make-edge	<pre>;; connects the two turtles
	to make-edge [node1 node2]
	  ask node1 [create-link-with node2 [
	    set rewired? false
	  ]]
	end</pre>

go	<pre>to go
	  ask turtles [communicate-via update-algorithm]
	  ask turtles [update-color]
	  tick
	  plot-degree
	  set sum-of-states sum [state] of turtles
	  set ratio-2 count turtles with [state = 2.0]/ ((percent-grammar-2 / 100) * N)
	  set ratio-1 count turtles with [state = 1.0]/ ((percent-grammar-1 / 100) * N)
	  set ratio-0 count turtles with [state = 0.0]/ ((1 - ((percent-grammar-2 + percent-grammar-1)/ 100)) * N)
	end</pre>

Important Index

1. Magnetization

Magnetization indicates the mean state of language users in the network, this index can represent the overall weight of grammars, update the grammars distribution at every tick. If magnetization value is closer to 0, it indicates that "state 0" has higher proportion in the network than "state 2"; if magnetization value is closer to 2, it indicates that "state 2" has higher proportion in the network than "state 0". When magnetization value tends to be stable over the iterations, representing the overall network state reached equilibrium.

2. Ratio

Ratio for each state is the ratio of current number of agents with specified state to the initialized number of agents with specified state. Ratio index can indicate the changing trends of each state proportion in overall network, has complementary effectiveness to the magnetization index. Ratio index can directly show observer which state is inclining and which state is declining in the network, therefore to obtain the qualitative relationship between initial proportion of specified state and corresponding changing trends over the iterations.

Behavior Space Analysis

Preferential attachment

1. Relationship between threshold value and ratio for each state.

In this behavior space experiment, set N = 150, percent-grammar-1 = 40, percent-grammar-2 = 40, choose "threshold" updating algorithm, varying threshold value from 0 to 1 with interval 0.1, time limit for each model run is ticks = 200, the repetition for each specified threshold value is 50 model runs. The output is final ratio-0, ratio-1, ratio-2.

Theoretically, when threshold value become higher, the more difficult for agent to change its state. In preferential attachment network, the degree distribution is highly right-skewed, which indicate that minority of agents are well connected and majority of them only have one to three connections. Therefore, agents with few links are more sensitive to change their state regardless of threshold value set.

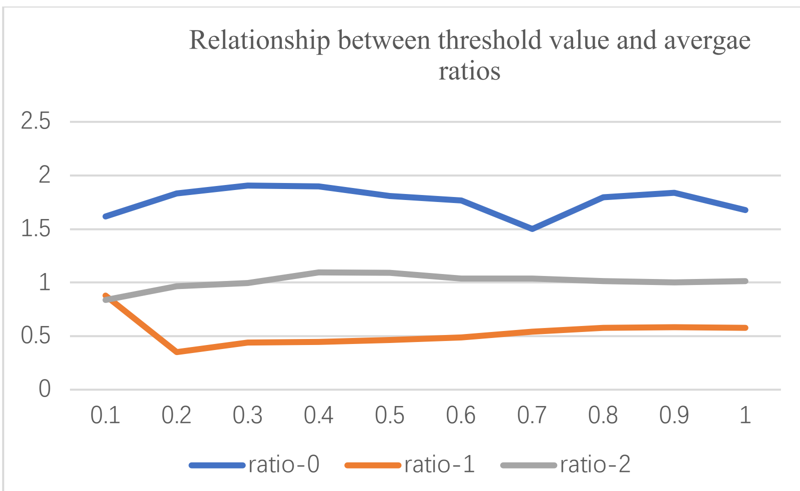


Fig1. Plot between average ratios for each state and threshold value

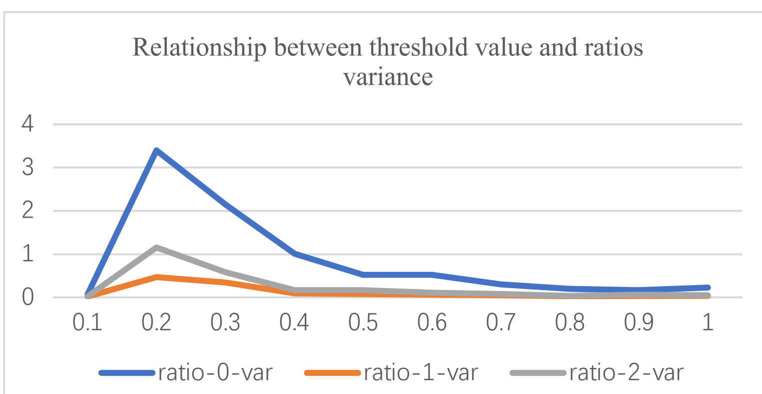


Fig2. Plot between variance of ratios for each state and threshold value

As the plots show above, Under the 40% grammar 1, 40% grammar 2, 20% grammar 0 initialization, equilibrium proportion of state 0 incline over the threshold value range from 0 to 1, with higher variance which reached peak at 0.2 threshold; equilibrium proportion of state 2 stably fluctuates around 40% (ratio-2 approximate to 1), with relatively low variance; equilibrium proportion of state 1 decline over the threshold value range from 0 to 1, ratio-1 approximate to 0.5, indicate 20% equilibrium proportion.

2. Analysis of magnetization when varying threshold value and percent-grammar-2.

Conduct behavior space experiment, set N = 150, percent-grammar-1 = 20, choose "threshold" updating algorithm, varying threshold value from 0 to 1 with interval 0.1, and varying percent-grammar-2 from 0 to 80 with interval 10, time limit for each model run is ticks = 300, the repetition for each specified threshold value is 50 model runs. The output is final (equilibrium) magnetization.

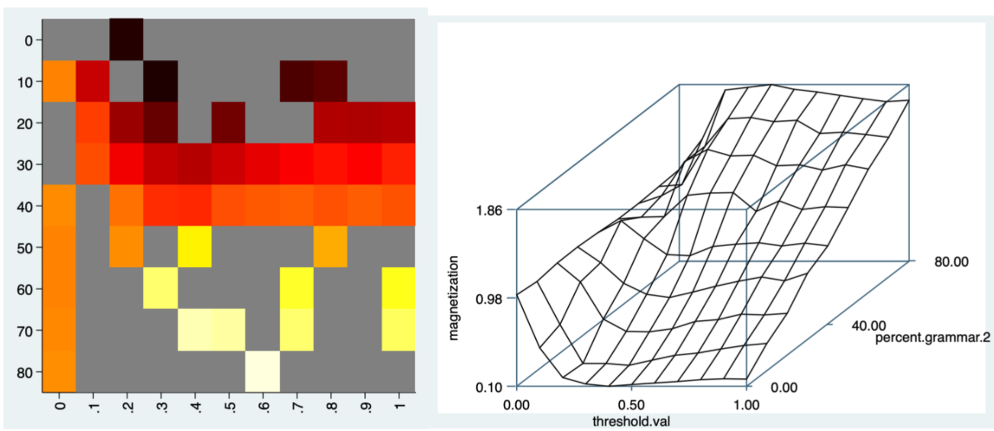


Fig3. Heatmap and 3D surface of magnetization

As the heatmap showed above, it's obvious that when the initial proportion of grammar 2 become higher, the magnetization tends to be larger and approximate to 2, there is no clear relationship between threshold value and magnetization, but basically, when percent-grammar-2 under 40, threshold value increase, the equilibrium magnetization tends to be lower after controlling for percent-grammar-2.

Small world

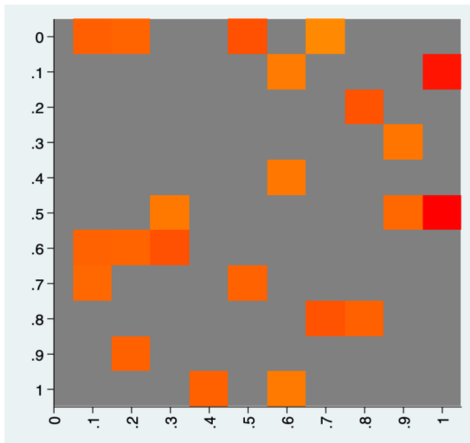
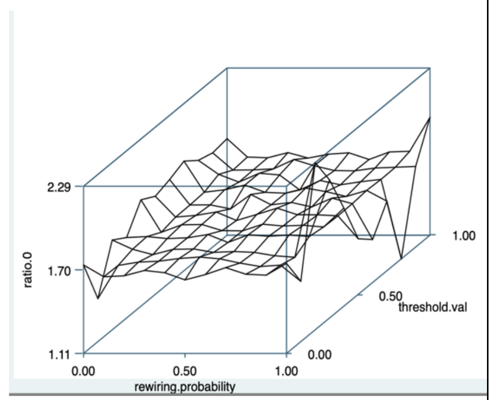
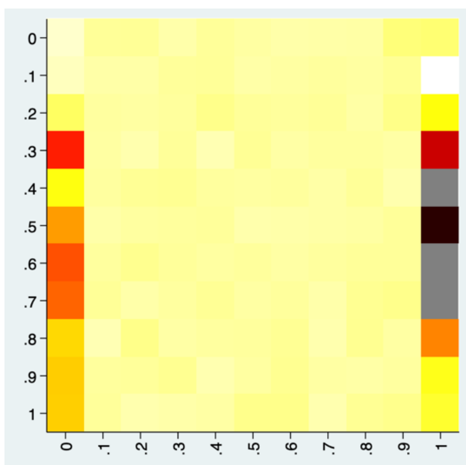
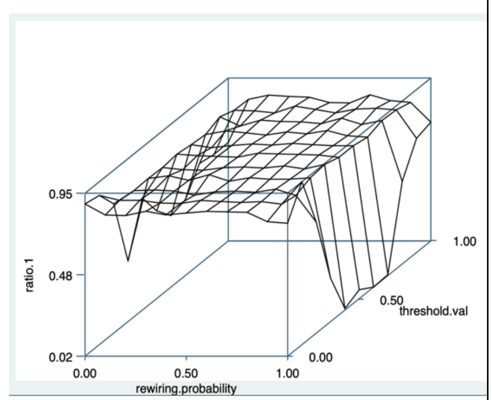
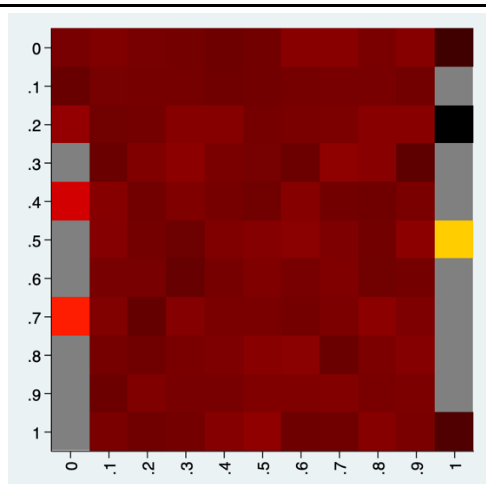
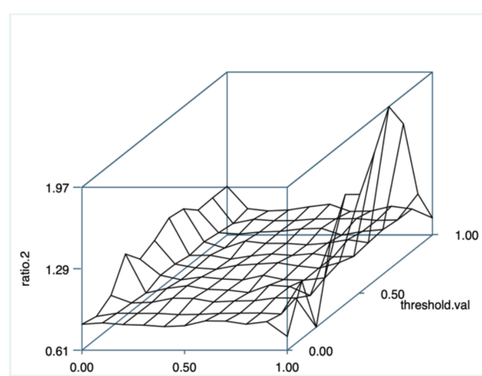
1. Analysis of ratios for each state when varying rewiring-probability and threshold-val.

Conduct behavior space experiment, set N = 150, percent-grammar-1 = 40, percent-grammar-2 = 40, choose "threshold" updating algorithm, varying threshold value from 0 to 1 with interval 0.1, and varying rewiring-probability from 0 to 1 with interval 0.1, time limit for each model run is ticks = 300, the repetition for each specified threshold value is 50 model runs. The output is final (equilibrium) ratio-0, ratio-1, ratio-2.

Generally, average degree in the network has positive association with rewiring-probability, when more agents are well connected, the transmission and communication of grammars is more active.

However, according to table 4, there is no significant relationship between ratio of state and varying parameters rewiring-probability and threshold value, the ratios of states are similar except extreme fluctuation at the edge.

Table 4. Heatmap and 3D Surface of ratio for each state

Ratio	Heatmap	3D Surface
Ratio for state 0		
Ratio for state 1		
Ratio for state 2		

2. Analysis of magnetization when varying threshold value and percent-grammar-2.

Conduct behavior space experiment, set N = 150, percent-grammar-1 = 20, choose "threshold" updating algorithm, set threshold value at 0.3, varying rewiring-probability from 0 to 1 with interval 0.1, and varying percent-grammar-2 from 0 to 80 with interval10. Time limit for each model run is ticks = 300, the repetition for each specified threshold value is 50 model runs. The output is final (equilibrium) ratio-2.

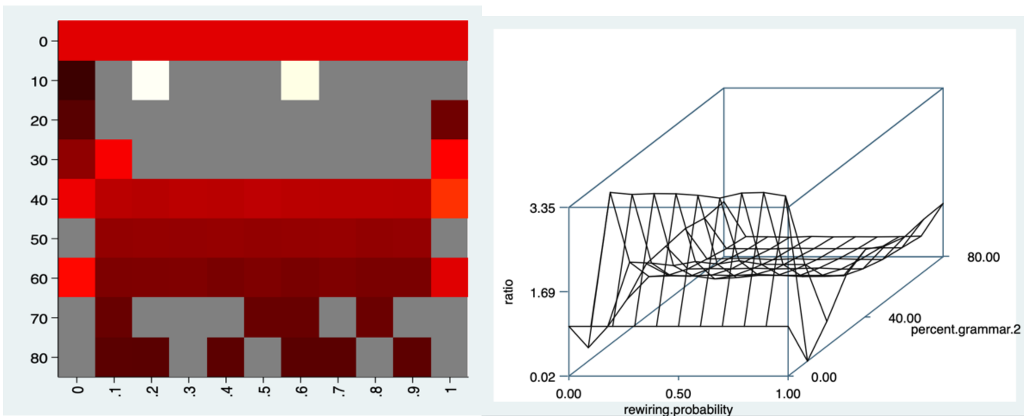


Fig4. Heatmap and 3D surface of ratio-2

As fig.4 show, the equilibrium proportion of state 2 incline when percent-grammar-2 increase from 0 to 20, controlling for rewiring probability, and reached peak at around 30. However, ratio 2 start to drop when percent-grammar-2 keep increase, regardless varying rewiring probability. It also indicates that percent-grammar-2 has dominant effect of ratio 2, rewiring-probability seem to be less effective.