

People are responding to an environment that consists of other people responding to *their* environment, which consists of people responding to an environment of people's responses.

—Thomas C. Schelling

## I Residential Segregation

Racial discrimination in housing has a long and deplorable history in the United States. Before the mid-1960s, Jim Crow laws in the South and restrictive covenants in the North often prevented minorities from buying houses or renting apartments in neighborhoods intended as “whites-only” enclaves. African American families attempting to move into such areas often met with protests and violence (see Figure 19.1).

Sometimes residential segregation was enforced by more subtle means. Lorraine Hansberry's modern classic *A Raisin in the Sun* tells the story of an African American family struggling with poverty, racism, and inner conflict on the south side of Chicago in the decade after World War II. Walter and Ruth Younger and their son Travis, along with Walter's widowed mother Lena and sister Beneatha, live in poverty in a dilapidated two-bedroom apartment, filled, in Ruth's words, with “cracking walls,” “marching roaches” and an “cramped little closet which ain't now or never was no kitchen.”

Lena uses some of her late husband's life insurance to make a down payment on a house in the all-white neighborhood Clybourne Park, largely because it is less expensive than new homes in a development for black families. Soon the family receives a visit from Karl Lindner, “a quiet-looking middle-aged white man in a business suit holding his hat and a briefcase.” He represents the Clybourne Park Improvement Association and chairs the New Neighbors Orientation Committee. Although the Youngers had a legal right in Chicago to move to the Clybourne Park neighborhood, Lindner's remarks (excerpted below) show how subtler forms of racism were used to dissuade black families:

*We go around and see the new people who move into the neighborhood and sort of give them the lowdown on the way we do things out in Clybourne Park . . . and we also have the category of what the association calls—uh—special community problems . . . .*

*I am sure you people must be aware of some of the incidents which have happened in various parts of the city when colored people have moved into certain areas . . . . Well because we have what I think is going to be a unique type of organization in American community life*

**FIGURE 19 1** February 1942 protest sign at the Sojourner Truth Homes, a federal housing project in Detroit. A riot was caused by white neighbors' attempt to prevent black tenants from moving in. Prospective tenants were pelted with rocks; 40 people were injured and there were more than 200 arrests. Photo by Arthur Siegel for the Office of War Information.



*not only do we deplore that kind of thing but we are trying to do something about it . . . We feel—we feel that most of the trouble in this world, when you come right down to it most of the trouble exists because people just don't sit down and talk to each other.*

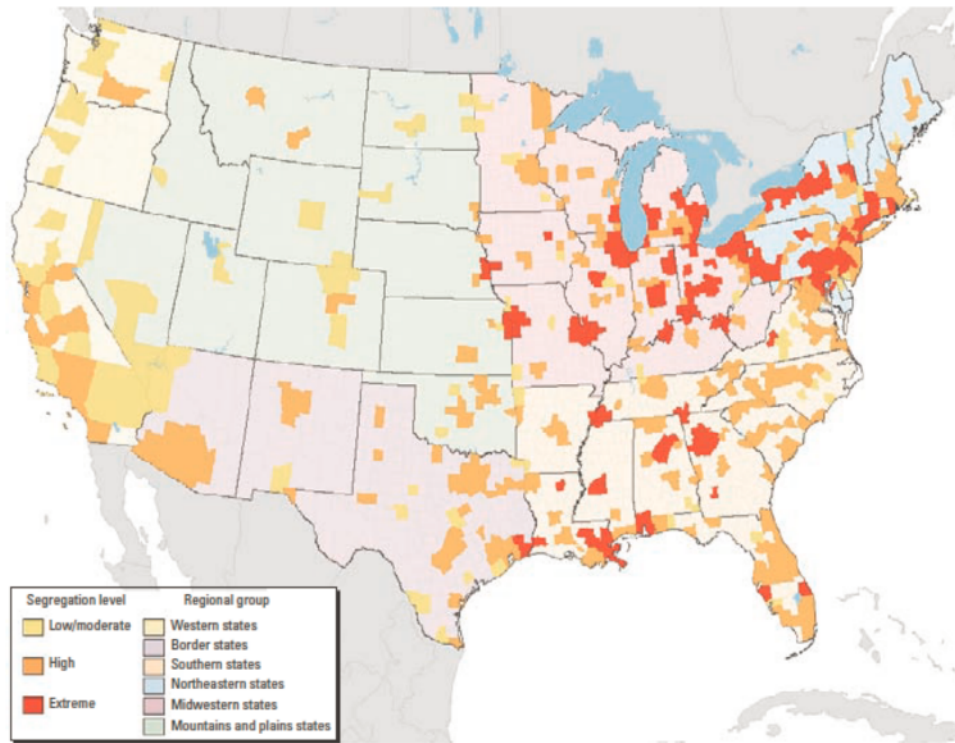
*That we don't try hard enough in this world to understand the other fellow's problem. The other guy's point of view . . . Yes that's the way we feel out in Clybourne Park. And that's why I was elected to come here this afternoon and talk to you people. Friendly like, you know, the way people should talk to each other and see if we couldn't find some way to work this thing out. As I say, the whole business is a matter of caring about the other fellow.*

*Anybody can see that you are a nice family of folks, hard working and honest I'm sure . . . Today everybody knows what it means to be on the outside of something. And of course, there is always somebody who is out to take advantage of people who don't always understand. . . . Well you see our community is made up of people who've worked hard as the dickens for years to build up that little community. They're not rich and fancy people; just hard-working, honest people who don't really have much but those little homes and a dream of the kind of community they want to raise their children in. Now, I don't say we are perfect and there is a lot wrong in some of the things they want.*

*But you've got to admit that a man, right or wrong, has the right to want to have the neighborhood he lives in a certain kind of way. And at the moment the over-whelming majority of our people out there feel that people get along better, take more of a common interest in the life of the community, when they share a common background. I want you to believe me when I tell you that race prejudice simply doesn't enter into it. It is a matter of the people of Clybourne Park believing, rightly or wrongly, as I say, that for the happiness of all concerned that our Negro families are happier when they live in their own communities. . . . You see in the face of all the things I have said, we are prepared to make your family a very generous offer . . . Our association is prepared, through the collective effort of our people, to buy the house from you at a financial gain to your family.*

Title VIII (“The Fair Housing Act”) of the Civil Rights Act of 1968 prohibited discrimination concerning the sale, rental, and financing of housing based on race, religion, or national origin. As a consequence of the law, racial separation in housing has declined but still remains a persistent problem for many groups, particularly blacks and Hispanics.

The **index of dissimilarity** is a widely used measure of the unevenness with which two groups are distributed across a geographic area. The index ranges from 0 (complete integration) to 100 (complete segregation). The dissimilarity index measures for whites and

**FIGURE 19 2**

National map of multigroup racial/ethnic segregation in the United States, from Morello-Frosch and Jesdale (2006); used with permission of Professor Morello and *Environmental Health Perspectives*.

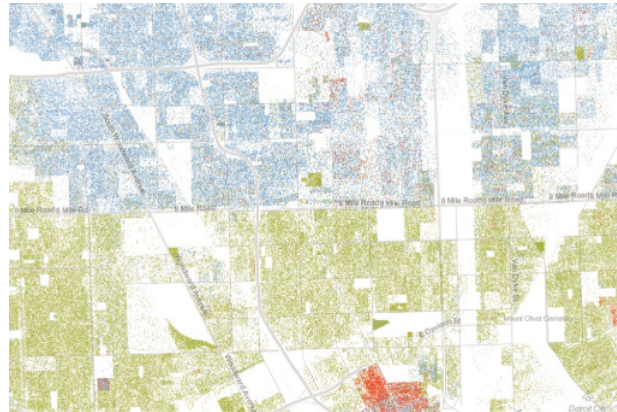
blacks in 1980, 1990, and 2000 were 72.7, 67.8, and 64.0, respectively. Blacks are hypersegregated in most of the largest metropolitan areas across the country, including Baltimore, Chicago, Cleveland, Detroit, Houston, Los Angeles, Milwaukee, New Orleans, New York, Philadelphia, and Washington, DC. Figure 19.2 displays regions of low, moderate, and high segregation in the United States.

Does the present situation of widespread segregation truly reflect how our citizens wish to live? In what kind of neighborhoods do people want to reside? The University of Chicago's National Opinion Research Center has conducted the General Social Survey since 1972. Using in-person interviews of adults in randomly selected households, the survey collects data on demographic characteristics and attitudes of U.S. residents. For the past quarter-century, the survey has found that between 55 and 60 percent of adults, black or white, prefer a neighborhood that is half-black and half-white.

Though people have reported a moderate to noticeable preference for integrated neighborhoods, the reality is quite different. In our major cities, only 3 or 4 percent of black residents live in neighborhoods in which the white population makes up 40 to 60 percent of the total. Surveys generally reveal more tolerance for mixed-race neighborhoods than we actually find in our urban areas. If large proportions of people want integrated neighborhoods, why don't we have more of them? Could it be that people are more intolerant than they report on surveys?

In this chapter, we present a very brief account of agent-based models, an approach that helps scientists understand how sometimes surprising global results emerge from the actions of individual actors. We will examine how future Nobel laureate Thomas C.

**FIGURE 19 3** A compressed view of a map of the Detroit metropolitan area displaying a dot for each individual. White residents are depicted with blue dots, African Americans with green, Asians with red, Latinos with orange, and all others with brown. In Detroit, among the most segregated cities in America, 8 Mile Road, the northern boundary of the city, serves as a sharp dividing line. Image copyright 2013, Weldon Cooper Center for Public Service, Rector and Visitors of the University of Virginia (Dustin A. Cable, creator); used with permission of the University of Virginia.

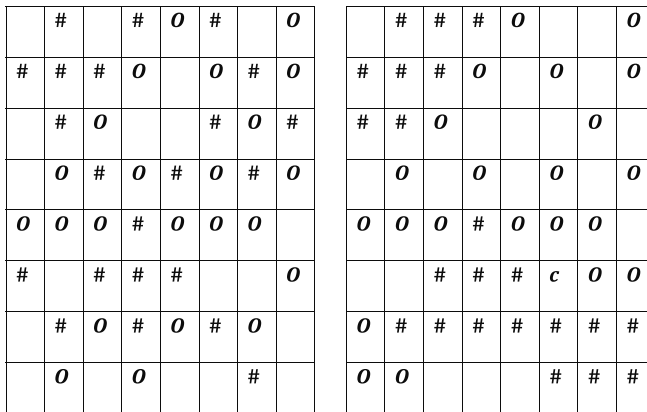


Schelling used an agent-based model to illustrate how large-scale (macro) social patterns can substantially differ from individual (micro) motives. Schelling's relatively simple model shows how segregated neighborhoods can result even when individuals are content to live in areas where only a minority of their race reside. Figure 19.3 shows the segregated pattern of housing in Detroit, Michigan, based on 2010 census data.

## II Schelling's Model

Schelling conceived an experiment that begins by placing, at random, equal numbers of two different types of households (black and white, Protestant and Catholic, Hispanic and Asian, French-speaking and English-speaking, etc.) in a large residential area. In Round 1, each household examines the makeup of its immediate neighborhood. If it is happy with the mix of families, then it stays put; otherwise, it moves to a new location. Round 1 ends when all the initial moves have been made. Round 2 is a repetition of Round 1. Each household looks at its new immediate neighborhood, which may have a different configuration than it had at the start of Round 1. Again, individual decisions are made to move or remain in place. We conclude Round 2 after all these decisions have been made and carried out. We continue the experiment with Round 3, Round 4, and as many rounds as needed until every household is content with its neighborhood. We then examine how segregated/integrated the results are.

It is not possible, of course, to carry out such an experiment in the real world. We can, however, *simulate* such an experiment. Imagine a large checkerboard acting as our residential area with each square being a possible location for a family. Take equal numbers of two different colored (say red and green) stones and scatter them initially at random among the squares. The stones represent our households. Some squares will contain one stone, others will have no stones. Then we set a *happiness threshold*  $q$ , the percentage of neighbors who must be of the same color as a stone for that stone to be happy with the neighborhood composition. If the percentage of nearest neighbors exceeds  $q$ , then we leave that stone in place, at least during Round 1. If the percentage falls below  $q$ , then we move that stone to a nearby empty square. Round 1 ends when we have carried out this process for every stone on the checkerboard. Round 2 operates in the same way as Round 1, except the initial configuration is the distribution of stones at the end of Round 1.



**FIGURE 19.4** A checkerboard representation of Schelling's residential segregation model simulation. On the left is the initial configuration of the board with two types of families denoted by # and O. On the right is the final configuration. This example assumes that a family is happy if at least half its neighbors are of the same type.

Schelling carried out many such experiments, beginning with an even simpler artificial world of one dimension. He recounted his initial experiences in a 2006 paper “Some Fun, Thirty-Five Years Ago”

*One afternoon, settling into an airplane seat, I had nothing to read. To amuse myself I experimented with pencil and paper. I made a line of x's and o's that I somehow randomized, and postulated that every x wanted at least half its neighbors to be x's and similarly with o's. Those that weren't satisfied would move to where they were satisfied. This was tedious, because I had no eraser, but I persuaded myself that the results could prove interesting.*

*At home I took advantage of my son's coin collection. He had quantities of pennies, both copper and the gray zinc ones we had all used during the war. I spread them out in a line, either in random order or any haphazard way, gave the coppers and the zincs their own preferences about neighbors, and moved the discontents—starting at the left and moving steadily to the right—to where they might inject themselves between two others in the line and be content. The results astonished me. But as I reflected, and as I experimented, the results became plausible and ultimately obvious.*

Although Schelling found the results fascinating, he realized that a two-dimensional framework would be more realistic:

*So I made a 16 × 16 checkerboard, located zincs and coppers at random with about a fifth of the spaces blank, got my twelve-year-old to sit across from me at the coffee table, and moved discontented zincs and coppers to where their demands for like or unlike neighbors were met. We quickly found out it didn't matter much in what order we selected the discontents to move—from middle outward, from out inward, from left to right or diagonally. We kept getting the same kind of results. The dynamics were sufficiently intriguing to keep my twelve-year-old engaged.*

Figure 19.4 shows a typical simulation result on an 8 × 8 board.

To have some confidence that the order in which the discontents are selected fundamentally changes neither the qualitative nature of the outcome nor exactly how the coins are initially distributed on the checkerboard, it is necessary to repeat the experiment many times, carefully recording the results and doing some statistical analysis on the data.

### III Agent Based Modeling

As we noted in chapter 15, simulation is an important and useful technique for studying complex phenomena. We can use simulation to complement a more rigorous deductive approach or to substitute for formal deduction when existing mathematical tools are too weak. We can't prove theorems with simulation, but repeated simulations can help us gain insight into complicated dynamic situations. An *agent-based* simulation is a model of an evolving system using autonomous interacting individuals.

In an agent-based model, we specify the characteristics of the individual actors. These properties can differ from individual to individual. They may change as time proceeds and the environment changes as the result of decisions made by other individuals. Agents may die and new ones may be born. We may endow the agents with objectives and describe what adaptive decisions they may make.



Steven F. Railsback (left) is an adjunct professor in the mathematical modeling graduate program at Humboldt State University and a consulting environmental engineer and ecologist in Arcata, California. Volker Grimm (right) is a senior scientist in the Department of Ecological Modeling, Helmholtz Center for Environmental Research, Leipzig, and a professor at the University of Potsdam, Germany.

Steven Railsback and Volker Grimm (2012) provide a useful description of the need for, and usefulness of, agent-based models (ABMs):

*Historically, the complexity of scientific models was often limited by mathematical tractability: when differential calculus was the only approach we had for modeling, we had to keep models simple enough to “solve” mathematically and so, unfortunately, we were often limited to modeling quite simple problems.*

*With computer simulation, the limitation of mathematical tractability is removed so we can start addressing problems that require models that are less simplified and include more characteristics of the real systems. ABMs are less simplified in one specific and important way: they represent a system's individual components and their behaviors. Instead of describing a system only with variables representing the state of the whole system, we model its individual agents.*

*ABMs are thus models where individuals or agents are described as unique and autonomous entities that usually interact with each other and their environment locally. Agents may be organisms, humans, businesses, institutions, and any other entity that pursues a certain goal. Being unique implies that agents usually are different from each other in such*

characteristics as size, location, resource reserves, and history. Interacting locally means that agents usually do not interact with all other agents but only with their neighbors—in geographic space or in some other kind of “space” such as a network. Being autonomous implies that agents act independently of each other and pursue their own objectives. Organisms strive to survive and reproduce; traders in the stock market try to make money; businesses have goals such as meeting profit targets and staying in business; regulatory authorities want to enforce laws and provide for the public well-being. Agents therefore use adaptive behavior: they adjust their behavior to the current states of themselves, of other agents, and of their environment.

Using ABMs lets us address problems that concern emergence: system dynamics that arise from how the system’s individual components interact with and respond to each other and their environment. Hence, with ABMs we can study questions of how a system’s behavior arises from, and is linked to, the characteristics and behaviors of its individual components. . . .

ABMs are useful for problems of emergence because they are across-level models. Traditionally, some scientists have studied only systems, modeling them using approaches such as differential equations that represent how the whole system changes. Other scientists have studied only what we call agents: how plants and animals, people, organizations, etc. change and adapt to external conditions. ABMs are different because they are concerned with two (and sometimes more) levels and their interactions: we use them to both look at what happens to the system because of what its individuals do and what happens to the individuals because of what the system does.

Schelling’s model was one of the first embodying the basic concept of *agent-based* (also called *individual-based*) models: autonomous agents interacting in a shared environment whose individual decisions lead to a complex emergent result. We endow the agents with lower-level microproperties and observe how macropatterns emerge.

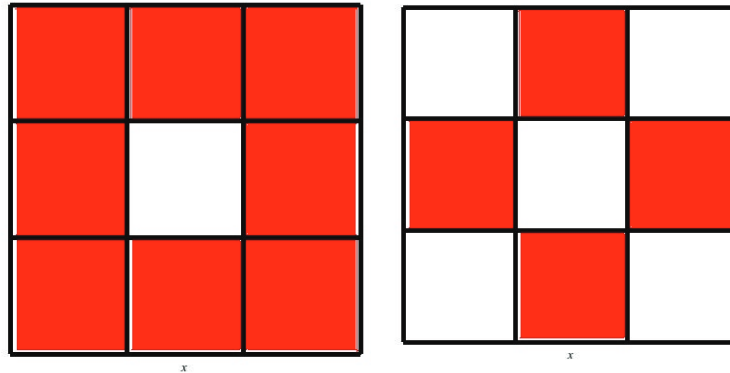
Though it is certainly possible to repeat what Schelling and his son did with coins and a checkerboard (and I encourage you to do so with a friend at least a few times), this is a rather tedious way to proceed. Fortunately, there are many ways to program a computer to replicate Schelling’s experiment. In this chapter, we will focus on the use of NetLogo as a programming language designed as a powerful, easy-to-use way to code agent-based models.

## IV NetLogo

NetLogo’s roots are in Logo, an educational programming language developed in the late 1960s for use by schoolchildren. Logo’s best-known feature is the *turtle*, a cursor that shows output on a computer screen from commands for movement. NetLogo is the creation of Uri Wilensky, who considers it a dialect of Logo. Wilensky writes that *Net* is meant “to evoke the decentralized, interconnected nature of the phenomena you can model with NetLogo, including network phenomena. It also refers to HubNet, the multiuser participatory simulation environment included in NetLogo.” He first created NetLogo in 1999 at the Center for Connected Learning and Computer-Based Modeling, then at Tufts University but now located at Northwestern University. NetLogo 1.0 was released in 2002. He released NetLogo 5.2 in April 2005.


NetLogo is free, open-source software you can download for Mac OS X, Microsoft Windows, or Unix operating systems from <https://ccl.northwestern.edu/netlogo/>. The download comes with an extensive user’s guide, tutorials, and a “Models Library” containing a great many examples from art, biology, chemistry, computer science, earth

**FIGURE 19 5** The shaded boxes show the Moore neighborhood (on the left) and the Von Neumann neighborhood (on the right) of a central cell.




science, mathematics, and the social sciences. You should download and install NetLogo before proceeding.

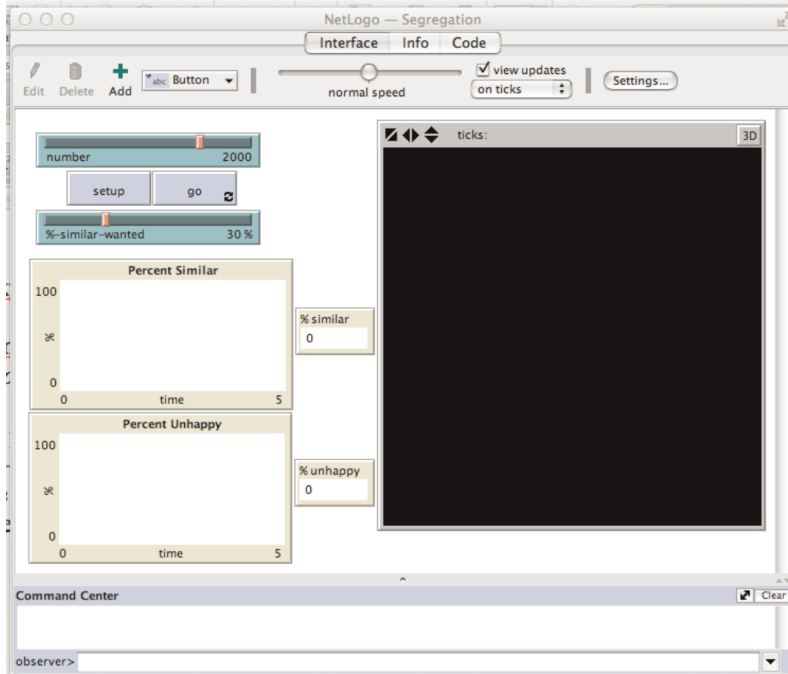
To implement Schelling's residential segregation model, we need to define what is meant by the *neighborhood* of a household. We will use what is known as a *Moore neighborhood*: the eight cells surrounding a central cell (the location of our family) on a two-dimensional square lattice. The Moore neighborhood is named after Edward F. Moore (1925–2003), one of the pioneers of cellular automata theory and the study of artificial life. The other type of neighborhood used in some studies is the *Von Neumann Neighborhood*. It consists of four cells, the two immediately above and below the central cell and the pair immediately to the left and right. Figure 19.5 illustrates the difference between the two types of neighborhoods.

We will demonstrate some runs of Schelling's model with NetLogo before we examine the details of the program itself. After downloading and installing NetLogo, open the application by double-clicking on the NetLogo icon . Navigate to the Segregation model by following the path *File Models Library Social Science Segregation* and then open the model. Figure 19.6 shows the initial display.

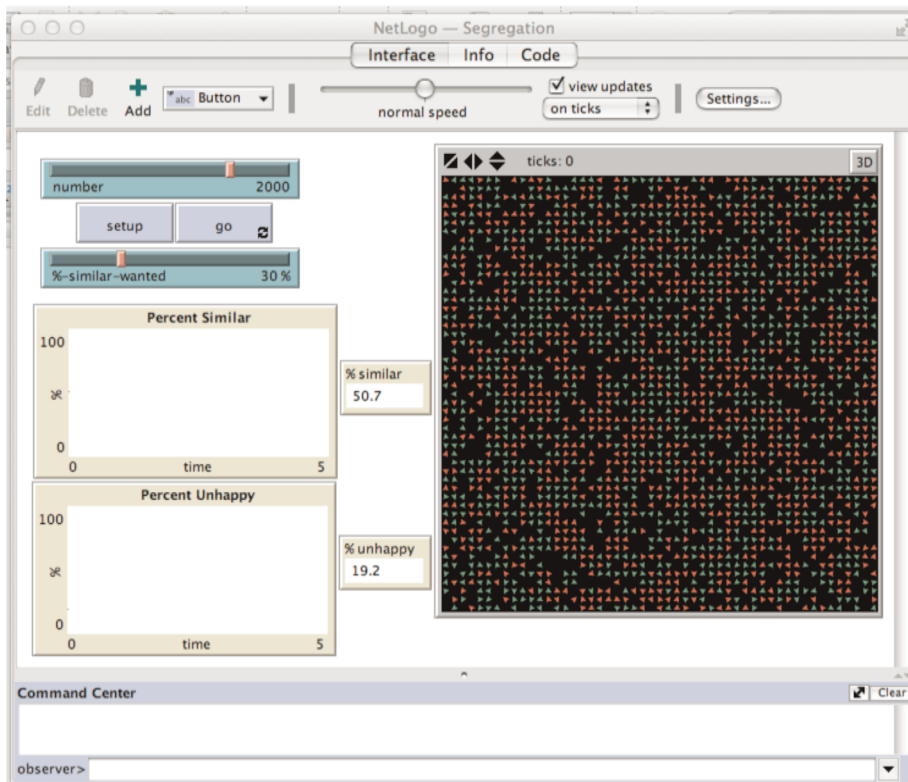
You will see three tabs at the top of NetLogo's main window, labeled *Interface*, *Info* and *Code*. Only one tab at a time can be active, but you can switch among them by clicking on the tabs at the top of the window. The *Interface* tab is highlighted. The black rectangle on the right is the *View*, a visual representation of the NetLogo world of turtles and patches. It is initially all black, because there are no agents (turtles), and none of the patches is occupied.

To populate the world with households, click on the *setup* button . The resulting screen will look similar to Figure 19.7. The exact distribution of red and green dots and the numbers in the *%similar* and *%unhappy* windows may be different. In this particular setup,  $51 \times 51 = 2,601$  patches (possible locations for houses) have been created. A total of 2,000 turtles have been distributed randomly among these patches, half red and half green. You can move the *number* slider to create fewer or more families. The other slider *%-similar-wanted* is set at 30 percent; this indicates that each family of one color wants at least 30% of its neighborhood to be made up of similarly colored families. If the actual percentage of the Moore neighborhood of a central cell is less than 30 percent, then the

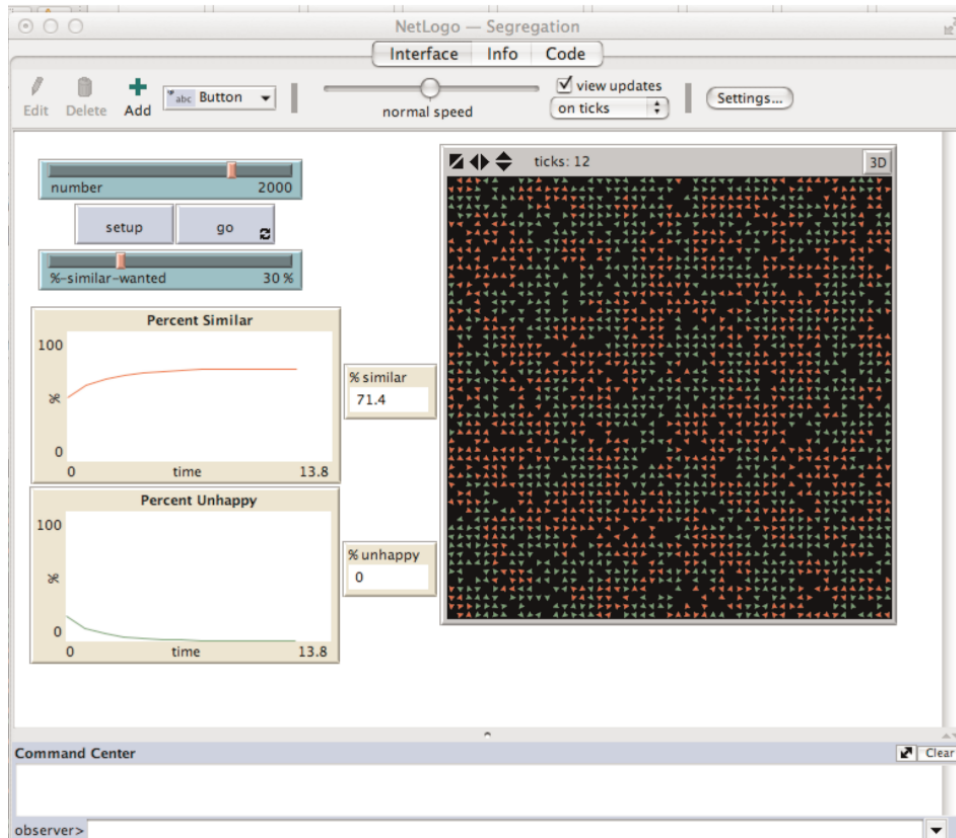




**FIGURE 19 6** The initial NetLogo window for Schelling's Segregation Model.



**FIGURE 19 7** The initial distribution of red and green families appears in the View after you click the *setup* button.

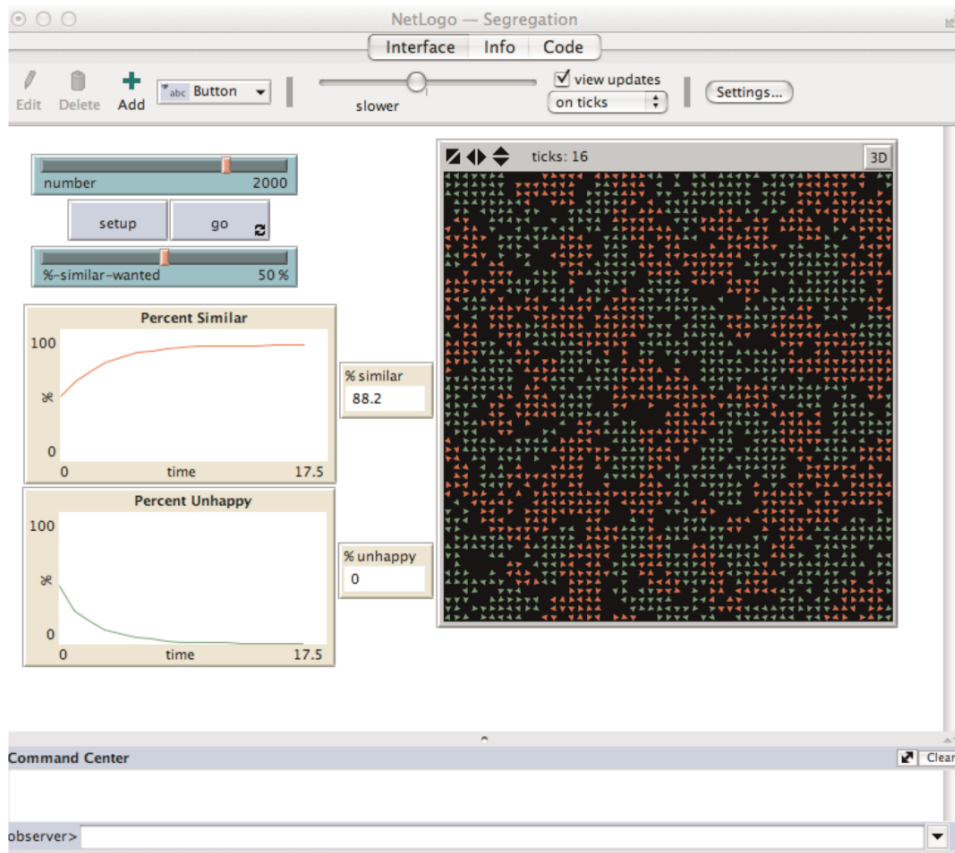


**FIGURE 19.8** The final distribution of red and green families when no one is unhappy. The graphs show the increase and leveling out of the percentage of neighbors of the same color and the almost exponential decline of unhappy families.


family in the central cell will be unhappy and will move to a new location in the current round. Note that initially 19.2 percent of the families are unhappy. If the percentage of one's immediate neighbors exceed *%-similar-wanted*, then we call that family *happy*, and it takes no action in this round to move.

We are now ready to run the model. Click the *Go* button. Note the resulting view, where there now are clumps of red and green neighborhoods (see Figure 19.8). After 12 rounds, the graphs of *Percent Similar* and *Percent Unhappy* are now 71.4 percent and 0 percent, respectively. Every family is now happy, so movement has stopped. Although each family would be content if its neighborhood was 30 percent of its own color and 70 percent of the other color, we see that 71.4 percent of all the nearest neighbors of a family are the same color as that family.

Let's examine what happens if we increase the *%-similar-wanted* to 50 percent. Now each family desires that families of the same color occupy at least four of the eight surrounding houses. Move the *%-similar-wanted* to 50 percent, click *setup* and then click *go*. Figure 19.9 shows a typical result. The program stopped after sixteen rounds, with no families unhappy and a very high degree (88.2 percent) of similarity. The view window shows a highly segregated residential tract.

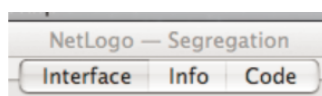


**FIGURE 19 9** The final distribution of red and green families with a run of the Segregation problem when the *%-similar-wanted* is increased to 50%.

The default topology of the NetLogo world is a *torus*. It wraps in both directions, meaning that the top and bottom edges of the world are connected and the left and right edges are connected. Thus if a turtle moves beyond the right edge of the world, it appears again on the left, and the same for the top and bottom. The wrapping features can be turned on or off by using options that become available by clicking on the *Settings . . .* button  near the top of the NetLogo window. For example, if both vertical and horizontal wrapping is turned off, the topology becomes a *box*: the world is bounded so turtles that try to move off the edge of the world cannot. The patches around edge of the world have fewer than eight neighbors: the corners have three, and the rest have five.

## V The NetLogo Code for Schelling's Model

To understand why the program behaves the way it does, we have to look under the hood and study the sequence of commands making up the program. When you open a model in NetLogo, at the very top of the screen you will see three buttons as shown in Figure 19.10.



**FIGURE 19 10** *Interface*, *Info*, and *Code* provide three different windows for a NetLogo program.

```

globals [
  percent-similar ;; on the average, what percent of a turtle's neighbors
                  ;; are the same color as that turtle?
  percent-unhappy ;; what percent of the turtles are unhappy?
]

turtles-own [
  happy? ;; for each turtle, indicates whether at least %-similar-wanted percent of
          ;; that turtles' neighbors are the same color as the turtle
  similar-nearby ;; how many neighboring patches have a turtle with my color?
  other-nearby ;; how many have a turtle of another color?
  total-nearby ;; sum of previous two variables
]

to setup
  clear-all
  if number > count patches
    [ user-message (word "This pond only has room for " count patches " turtles.")
      stop ]

  ;; create turtles on random patches.
  ask n-of number patches
    [ sprout 1
      [ set color red ] ]
  ;; turn half the turtles green
  ask n-of (number / 2) turtles
    [ set color green ]
  update-variables
  reset-ticks
end

to go
  if all? turtles [happy?] [ stop ]
  move-unhappy-turtles
  update-variables
  tick
end

```

**FIGURE 19 11** The initial part of the NetLogo code for the segregation model as it would appear on your computer screen.

By default, the *Interface* is the window initially shown. To see the complete program, click on the *Code* button. Figure 19.11 displays the initial lines of the code as they would appear on a computer monitor.

In this section, we show the full program with the lines numbered so that it is easy to make reference to specific portions. There are no line numbers in the NetLogo program itself.

1. `globals [`
2. `percent-similar ;; on the average, what percent of a turtle s neigh-`  
`neighbors ;; are the same color as that turtle?`
3. `percent-unhappy ;; what percent of the turtles are unhappy?`
4. `]`
5. `turtles-own [`
6. `happy? ;; for each turtle, indicates whether at least %-similar-wanted`  
`percent of`
7. `;; that turtles neighbors are the same color as the turtle`
8. `similar-nearby ;; how many neighboring patches have a turtle with my`  
`color?`
9. `other-nearby ;; how many have a turtle of another color?`

```

10. total-nearby ;; sum of previous two variables
11. ]
12. to setup
13. clear-all
14. if number > count patches
    [ user-message (word "This pond only has room for " count patches "
    turtles.")
    stop ]
    ;; create turtles on random patches.
15. ask n-of number patches
16. [ sprout 1
17. [ set color red ] ]
18. ;; turn half the turtles green
19. ask n-of (number / 2) turtles
20. [ set color green ]
21. update-variables
22. reset-ticks
23. end
24. to go
25. if all? turtles [happy?] [ stop ]
26. move-unhappy-turtles
27. update-variables
28. tick
29. end
30. to move-unhappy-turtles
31. ask turtles with [ not happy? ]
32. [ nd-new-spot ]
33. end
34. to nd-new-spot
35. rt random-float 360
36. fd random-float 10
37. if any? other turtles-here
38. [ nd-new-spot ] ;; keep going until we find an unoccupied patch
39. move-to patch-here ;; move to center of patch
40. end
41. to update-variables
42. update-turtles
43. update-globals
44. end
45. to update-turtles
46. ask turtles [
47. ;; in next two lines, we use "neighbors" to test the eight patches
48. ;; surrounding the current patch
49. set similar-nearby count (turtles-on neighbors)
    with [color = [color] of myself]
50. set other-nearby count (turtles-on neighbors)
    with [color = [color] of myself]
51. set total-nearby similar-nearby + other-nearby
52. set happy? similar-nearby >= (%-similar-wanted * total-nearby / 100)
53. ]
54. end

```

```

55. to update-globals
56. let similar-neighbors sum [similar-nearby] of turtles
57. let total-neighbors sum [total-nearby] of turtles
58. set percent-similar (similar-neighbors / total-neighbors) * 100
59. set percent-unhappy (count turtles with [not happy?]) /
    (count turtles) * 100
60. end
61. ; Copyright 1997 Uri Wilensky.
62. ; See Info tab for full copyright and license.

```

Note first that a dozen or so lines, beginning with Line 2, contain a double semicolon (;;) followed by some text. The double semicolon marks the start of a *comment*, intended for the user to understand better the NetLogo commands. The program itself ignores anything on a line appearing after a double semicolon. Including comments to document the code is a vital part of good computer programming practice. Not only are they helpful to a reader new to the program, but they also remind the original writer what was intended when the code was first written some months back.

Modeling complex phenomena often requires long and complicated programs that may be difficult to read. Formal programming languages are typically dense; it is hard to look at a chunk of code and figure out what it is designed to do. Comments in ordinary language are very helpful.

Second, notice that some of the words in the program appear in green: **globals**, **turtles-own**, **to**, and **end**, for example. These are NetLogo *keywords* that have a predefined meaning. The **globals** section lists the variables that all parts of the program (agents, patches, etc.) can see and change. In Lines 5–10, the program specifies the characteristics of our families (called *turtles* in NetLogo) by including a list after **turtles-own** within square brackets ([ ]) of which features we will be using. In this case, there are four programmer-defined characteristics (*happy?* (defined in line 51), *similar-nearby* (line 48), *other-nearby* (line 49), and *total-nearby* (line 50)). Note that these are all defined within a procedure called *update-turtles*.

Let's examine how the programmer defined some of these characteristics. Commands and reporters built into NetLogo are called **primitives**. The **NetLogo Dictionary** (accessible through the *Help* menu) has a complete list of built-in commands and reporters. Lines 48–51 use the primitive **set**, which begins an *assignment* statement. An assignment statement has the form **set variable value**, which directs the program to use what appears in the *value* position for a new value of what appears in the *variable* position. Thus Line 50

```
set total-nearby similar-nearby other-nearby
```

assigns to the programmer-defined variable *total-nearby* the sum of the current values of two other programmer-defined variables *similar-nearby* and *other-nearby*.

To carry out the command in Line 50, the program must know how to determine the values of *similar-nearby* and *other-nearby*. That is why they are defined earlier in the sequence of commands. Let's examine Line 48 to see how *similar-nearby* is determined:

```
set similar-nearby count (turtles-on neighbors)
    with [color = [color] of myself]
```

This command uses several NetLogo primitives, all shown in purple. For each patch containing a turtle, the program will count the number of turtles of the same color in the Moore neighborhood of that patch and assign that value to *similar-nearby*.

Line 49 does something similar for *other-nearby*:

```
set other-nearby count (turtles-on neighbors)
  with [color = [color] of myself]
```

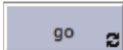
but uses the operator `≠`, which means “not equal to.”

Let's see whether we can figure out Line 52:

```
set happy? similar-nearby >= (%similar-wanted * total-nearby / 100)
```

Here *happy?* is what is called a *Boolean variable*. It has two possible values, **True** or **alse**, depending on whether the inequality `>=` (`>`) is satisfied. The NetLogo program can determine whether the inequality is valid, because at this point, it knows the values of *similar-nearby*, *%similar-wanted*, and *total-nearby*. Line 52 is the implementation of the idea that a family is happy if its Moore neighborhood contains at least many families of the same color as the center family desires.

Third, let's examine the overall structure of the program. There are basically two parts: *setup* and *go*. Each of these procedures calls upon other procedures to do its work:

- I. Pressing the *startup* button on the interface triggers the commands in the *setup* procedure, Lines 11–22. After checking to make sure there is enough space to create locations for the specified *number* of families, the procedure directs the program (through the primitive **ask**) to select *number* randomly chosen **patches** and create (**sprout**) that *number* of red turtles on these patches. Then it changes half of those turtles to the color green. Finally it calls the procedure *update-variables* (Lines 40–43), which in turn executes the procedures *update-turtles* (Lines 44–53) and *update-globals* (Lines 54–59). We have discussed how *update-turtles* determines which families are happy and which are not. The procedure *update-globals* should be self-explanatory.
- II. The small rounded arrows on the bottom right of the *go* button  in the *Interface* view indicate that pressing this button once will result in repeating applications of the commands in the *go* procedure until some conditions within that procedure cause the program to halt.
- III. The *go* procedure itself, found on Lines 23 through 28, is fairly short. It begins with stating the condition under which the program will halt—if every turtle (family) is happy because its desired proportion of similar neighbors is met or exceeded: if **all? turtles** [happy?] [ **stop** ]. Otherwise, the program executes, in order, the procedures *move-unhappy-turtles*, *update-variables*, and *tick*. The last step, **tick**, is a direction to advance the clock one step (move to the next Round in the language of our previous section) and update all requested plots.
- IV. The dynamic behavior displayed in the operation of the program comes from the *move-unhappy-turtles* procedure (Lines 29–32) and the procedure *nd-new-spot* that it, in turn, evokes. The *move-unhappy-turtles* simply directs any turtle that is currently not happy (recall that one of the characteristics of our agents, what *turtles own*, included the Boolean *happy*) to move using the *nd-new-spot* procedure which details how the movement should take place.
- V. Lines 33–39 (the *nd-new-spot* procedure) describe how an unhappy family is to go about finding a new place to live. Basically, the family picks a new spot and settles in (at least temporarily) if it is unoccupied; otherwise, it picks another spot and tries again. There are many ways of implementing just how a family selects a

new spot. Uri Wilensky, who wrote the NetLogo program we are examining, chose to have the family move along a random angle (between 0 and 360 degrees) a number of steps randomly chosen between 0 and 10. The primitive `rt` (you can use **right**) followed by a number says to turn right that number of degrees, whereas `fd` (or **forward**) followed by a number says to move forward that number of steps. This *nd-new-spot* procedure is a *recursive* one. If the spot to which the turtle has moved is already occupied, the procedure calls itself to try again.

We have by no means even begun to present all the features of the NetLogo software package. For example, there are four types of agents: *turtles*, *patches*, *links*, and the *observer*. We have only talked about the first two. There should be here, however, enough details of the segregation model for you to understand how it operates and to make adjustments to carry out the modifications suggested in the exercises. We strongly encourage you to work through the tutorials that accompany NetLogo, the programs in the *Models Library*, and the package's *User Manual* and *Dictionary*. Railsback and Grimm's 2011 text is an excellent source for learning how to design agent-based models and implement them in NetLogo. Wilensky and Rand's book is another excellent resource.

## VI Historical and Biographical Notes

The seeds for agent-based modeling were sown by in the late 1940s by two mathematicians we've already met, John von Neumann and Stanislaw Ulam. The wide availability of powerful individual computers a half-century later spurred the growth of such models.

As we noted earlier, Schelling's segregation model was one of the first agent-based models that drew significant attention. In an attempt to understand better the Prisoner's Dilemma Game of chapter 16, Robert Axelrod, a political scientist at the University of Michigan, hosted a "tournament" of strategies in the early 1980s to play the game in an agent-based manner (the "tit-for-that" approach emerged as the winner). Among the first agent-based simulations in biology were *flocking* models developed by computer scientist Craig Reynolds. Reynolds later worked on the films *Tron* and *Batman Returns*.

The development of software modeling packages such as NetLogo in the 1990s promoted a rapid expansion of agent-based modeling in the social and biological sciences which has continued through the present day. This chapter's references section provides sources for examining this development.

### A. Thomas Schelling

Ron Elisha is a practicing Australian medical doctor and playwright. In a 2010 drama, Elisha moves the audience back and forth between the making of the black comedy satirical film *Dr. Strangelove or How I Learned to Stop Worrying and Love the Bomb* and White House deliberations about how to deal with the Berlin blockade and the Cuban missile crisis. Six of the seven characters in the play are President John Kennedy, Secretary of Defense Robert McNamara, film director Stanley Kubrick, actors Peter Sellers and George Scott, and a dancer representing a number of different women linked to these men. We also hear excerpts of Frank Sinatra recordings in the background.

The seventh character, the only one interacting with both groups, is Thomas Schelling, and the play is entitled *The Schelling Point*. A *Schelling* (or *focal*) *point* is a term in game theory referring to a solution that players who cannot communicate with each other



will gravitate toward using because it has some special or natural attraction. According to the play's publisher,

*This drama explores the fanciful and ultimately romantic game theory notion that human behaviour is largely rational. Using the Schelling point—the point at which two parties who are unable to communicate can reach a common ground—as its focal point, the play undercuts the political landscape of the 60s with the personal crises of its protagonists, played out to the emotional rhythms of Sinatra's unrepentant romanticism.*

In the play, we see the Schelling character both lecturing Kennedy and McNamara on game theory and advising Kubrick on plausible ways a nuclear war could get started in an age of intercontinental ballistic missiles. Is this pure fantasy? Did the real Thomas Schelling do either of these things?

Thomas Crombie Schelling was born in Oakland, California on April 14, 1921. His father was a career naval officer, and the family lived in Panama for several years. Schelling began his college study at San Diego State University, switched to the University of California Berkeley, and took time off to study and work in Chile, where he was the only employee on duty at the American embassy in Santiago when news came of the Japanese attack on Pearl Harbor in December 1941. He completed his undergraduate degree at Berkeley and his Ph.D. at Harvard in economics.

Schelling has had extensive experience in government and in academia. He has served in Europe implementing the Marshall Plan, in the White House, and in the Executive Office of the President and has chaired a number of committees and commissions advising the government. After five years teaching at Yale, Schelling joined the Harvard faculty in 1958, as part of which he helped found the Kennedy School of Public Policy in 1969 and as part of which he taught for 20 years as the Lucius N. Littauer professor of political economy. He also spent several years doing research at the RAND Corporation in California and the International Institute for Applied Systems Analysis in Austria. After formal "retirement" from Harvard, Schelling served as distinguished professor of economics at the University of Maryland.

In October 1960, Schelling helped establish the Harvard-MIT Joint Seminar on Arms Control. His essay on "Reciprocal Measures for Arms Stabilization" was the first item discussed. Participants in the seminar numbered several who later served in prominent positions in the Kennedy and Johnson administrations. These included McGeorge Bundy, the president's national security adviser, and his assistant Walt Rostow; John McNaughton, who was an adviser to McNamara; and Jerome Weisner, Kennedy's science advisor. Another seminar participant was Henry Kissinger, who became national security advisor and secretary of state for presidents Richard Nixon and Gerald Ford.

One commission that Schelling chaired during the Kennedy administration was asked to make recommendations for preventing an accidental nuclear war between the United States and the Soviet Union. Schelling's group advised a "hot line" between the White House and the Kremlin so that the two national leaders could directly and instantaneously communicate with each other. Such a direct communications link via teletype was installed in June 1963. Although Elisha exercised considerable "poetic license" in staging Schelling lecturing Kennedy and McNamara, it is true that through this commission and others and his influential writings, Schelling has deeply influenced American public policy on nuclear weapons and other issues.

What about Schelling and *Dr. Strangelove*? Here Elisha's rendering is, in fact, less of an exaggeration. "It is perhaps fitting that the man responsible for the iconic hotline

between Washington and Moscow and other policies that helped stave off nuclear catastrophe at the height of the Cold War,” writes Matt Cadwallader “would also be the one to help Stanley Kubrick start World War 3.”



A scene from the Sydney Independent Theatre production of *The Schelling Point*

In the early 1960s, Schelling was asked to survey a series of fictional accounts of nuclear war for a magazine article. Peter George’s novel *Red Alert* made a strong impression on Schelling. He recalled the story’s plot as “the first plausible, detailed examination of how a war might actually get started.” The article drew the attention of Kubrick, who bought the movie rights to the novel and brought along George for a long meeting with Schelling. The three spent an afternoon wrestling with a considerable plot hole: When *Red Alert* was written in 1958, intercontinental ballistic missiles were not much a consideration in a potential U.S.–Soviet showdown. But by 1962, ICBMs had made much of the book’s plot points impossible. The speed at which a missile strike could occur would offer no time for the plot to unfold. “We had a hard time getting a war started,” recounts Schelling—but they eventually succeeded.

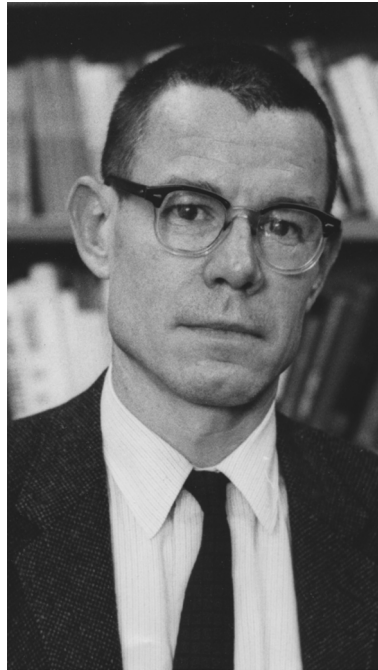
Schelling’s involvement with the film ended at that point. “However,” Cadwallader concludes, “his continued contributions to arms control and deterrence were instrumental in keeping nuclear Armageddon in the realm of fiction.”

The Royal Swedish Academy of Sciences selected Thomas C. Schelling and Robert J. Aumann as joint winners of the 2005 Nobel Prize in the Economic Sciences “for having enhanced our understanding of conflict and cooperation through game-theory analysis.” In his presentation speech at the Stockholm Concert Hall, Jörgen Weibull noted that

*[w]ar and other conflicts have tormented mankind since time immemorial, more often than not with great suffering and extensive material losses. Yet many societies live in prosperity founded on cooperation and peaceful competition. Why do some groups, organizations and countries*

*succeed in promoting cooperation while others suffer from conflict? Robert Aumann and Thomas Schelling approached this age-old question against the backdrop of the cold war, when an understanding of cooperation and conflict was imperative under the threat of nuclear war. Although they approached the topic from different angles—Aumann from mathematics and Schelling from economics—they shared a vision: that game theory had the potential to reshape the analysis of human interaction. . . .*

*Thomas Schelling showed how one party in a conflict can often strengthen its position by overtly worsening or limiting its own options—by “burning its ships”. He also demonstrated that the capability to retaliate can be more useful than the capability to resist an attack, and that uncertain and gradual retaliation can be more credible and less costly than immediate and certain retaliation. These insights have proven to be of great relevance for conflict resolution and the avoidance of war. They have also prompted new developments in economics, in particular concerning our understanding of market competition. For example, a firm can sometimes increase its profit through strategic investment in a large and expensive plant. Even if its average production costs would rise, losses due to inefficient production may be outweighed by the gains generated by competitors’ less aggressive behavior.*



Thomas C. Schelling about the time he developed the segregation model. Used with permission of Professor Schelling

The Nobel selection committee also emphasized Schelling’s residential segregation model in a statement issued when the prize was announced:

*A recurring theme in Thomas Schelling’s research is: what happens when individual plans and patterns of behavior are confronted in the social arena? The title of one of his most widely read books, *Micromotives and Macrobehavior*, reveals the overall theme. The book addresses different everyday phenomena such as professional ice-hockey players’ use of helmets, audiences’ choice of seats in an auditorium, and racial and sexual discrimination. Segregation is usually*

*associated with oppression. Historically, this has been an important part of the explanation, but segregation is also a stable phenomenon in developed societies, where considerable effort is devoted to counteracting it.*

*Schelling formulated a simple model where he assumed that all individuals are tolerant in the sense that they willingly live in the proximity of people with a different culture, religion or skin color, but that they want to have at least a few neighbors that share their own characteristics. If not, then they move to a neighborhood where they can find more people like themselves. Schelling showed that even rather weak preferences regarding the share of like persons in a neighborhood can result in strongly segregated living patterns. In other words, no extreme preferences on the part of individuals are required in order for a social problem to arise.*

Most recently, Schelling has published on military strategy and arms control, energy and environmental policy, climate change, nuclear proliferation, and terrorism. Other interests include organized crime, foreign aid and international trade, conflict and bargaining theory, racial segregation and integration, the military draft, health policy, tobacco and drugs policy, and ethical issues in public policy and in business.

In addition to his best-known works *The Strategy of Conflict* and *Micromotives and Macrobehavior*, Schelling's books include *National Income Behavior*, *International Economics*, *Strategy and Arms Control*, *Arms and Influence*, *Thinking through the Energy Problem*, and *Choice and Consequence*.

## **B. Uri Wilensky**

Uri Wilensky designed the NetLogo environment with the explicit objective to provide a low-threshold, no-ceiling tool for students and researchers alike to engage in agent-based modeling. “The vision,” he writes, “is that building simulations will become a common practice of Science and Social Sciences scholars investigating complex phenomena—the scholars themselves, and not hired programmers, build, run, and interpret the simulations. To these ends, the NetLogo ‘language’ has been developed so as to be accessible—easy to write, read, and modify.”

Uriel Joseph Wilensky is the son of two scholars of Jewish history and philosophy. His mother, Sarah Heller Wilensky, is the author of several books and essays focusing on philosophical sources of Kaballah and the work and thought of Don Isaac Abravanel and Isaac Arama. His father, Mordecai L. Wilensky, was a Zionist youth leader in Poland, escaped the Nazi holocaust, and later became a professor of Jewish history at Hebrew College in Brookline, Massachusetts. The elder Wilensky immigrated to Palestine in 1934; his parents and three brothers perished in Nazi concentration camps. One of the first graduates of Hebrew University in Jerusalem, Mordecai Wilensky's writings came to the attention of President Jimmy Carter. Carter appointed him as an adviser and invited him to the signing of the 1979 Israel–Egypt peace accord.

Uri Wilensky received undergraduate and master's degrees in mathematics and philosophy from Brandeis University in Waltham, Massachusetts. He later moved to another institution in the same state, the Massachusetts Institute of Technology, where he earned his Ph.D. in media arts and sciences in 1993. Wilensky's thesis, *Connected Mathematics: Building Concrete Relationships with Mathematical Knowledge* was completed under his adviser, the legendary Seymour Papert.

Papert, born in South Africa in 1928, holds two Ph.D.s in mathematics—one from the University of Witwatersrand and the other from Cambridge University. After research work at Cambridge, the Henri Poincaré Institute in Paris, the University of Geneva with Jean Piaget, and London's National Physical Laboratory, he joined the faculty of MIT. At MIT, Papert was a professor of both mathematics and education, director of the Artificial Intelligence Laboratory, and one of the founders of MIT's Media Lab. Papert developed a constructivist learning theory, based in part on Piaget's work, which he hoped would help children learn more effectively and become better problem solvers. He created Logo, a simple-to-use computer language to manipulate the movements of a small mobile robot called the Logo Turtle that children (and adults) could use to solve problems in an environment of play.

Wilensky worked for a number of years with Papert and other colleagues at MIT, serving on the faculty of Tufts University from 1994 to 2008. He founded the Center for Connected Learning and Computer-Based Modeling, which he subsequently relocated to Northwestern University. Wilensky directs the center and serves as a professor of learning sciences and computer science in the School of Education and Social Policy. He is involved in designing, deploying, and researching learning technologies—especially for mathematics and science education. His recent work has concentrated on the design of computer-based modeling and simulation languages, including networked collaborative simulations. He is very interested in the changing content of curriculum in the context of ubiquitous computation.

As we noted earlier, Wilensky created the NetLogo language and HubNet, a collaborative simulation software environment. He has published new accompanying tools, materials, curricula and written hundreds of programs using NetLogo as a modeling tool. He has generously made NetLogo free to any user. He reports that there have been more than 160,000 downloads of the software. “Our user community continues to expand—tens of thousands of students, teachers and researchers use NetLogo in their classrooms and work,” Wilensky writes. “The user group ranges from children downloading NetLogo to teach their parents how to model, to middle school classroom use, to researchers in academic institutions.”



Uri Wilensky. Used with permission of Professor Wilensky

Wilensky's work has received support from the National Science Foundation, the National Institutes of Health, and the United States Department of Education among other agencies. He is the recipient of a prestigious career award from the NSF.

Wilensky is a strong advocate for introducing computational modeling and simulation into a wide range of contexts. He argues that “both in and out of school,”

*[c]omputational modeling has the potential to give students means of expressing and testing explanations of phenomena both in the natural and social worlds. There are two approaches traditionally used to give explanations of phenomena: mathematical formulae and textual descriptions. Both of these have important deficiencies in an educational context. Mathematical formulae are not easily constructible by students. . . . As such they can be transmitted to students, but very few students will be able to understand the assumptions hidden in the formula nor challenge or critique the formula and offer a different formula as explanation. Moreover, mathematical formulae are brittle in that they apply under very specific conditions and if the conditions are not met, the formula cannot be easily adjusted to account for the new phenomenon. In contrast, textual descriptions are constructible by students and can be easily modified, but they suffer from a lack of testability. If two textual accounts of a phenomenon contradict each other, there are not clear ways to put them to the test to determine which is a better account.*

*Computational modeling can help students overcome both these problems. Clearly, contradictory computational models, as executable, can be put to the test. In so doing, students can uncover the assumptions behind the models and assess their adequacy. Computational models are more transparent and their assumptions are easier to detect and understand. . . . There is now also a body of research that shows that typical middle school and higher students can construct, modify and critique computational models. . . . The educational research shows that students learning with an ABM-based approach learn their subject material more deeply . . . they increase their computational thinking skills and [have] greater insight into the mechanisms of action that lead to patterns in nature and society. This has been shown in a wide range of disciplines such as electricity, population biology, evolution, kinetic molecular theory, materials science, sociology, and psychology. Furthermore, non-high-achieving students and members of underrepresented groups have shown increased motivation and understanding using ABM approaches. . . . This educational advantage combined with the increased use by scientists positions ABM to achieve significant penetration in schools, exposing all students to computational thinking, even those with no interest in computer science.*

## EXERCISES

Although it is possible to carry out some of these exercises with pencil and paper or coins and a checkerboard, we will assume that you have installed NetLogo on your computer or have access to a computer on which NetLogo is installed. These exercises suggest various experiments to carry out with original and suitably modified NetLogo programs and ask you to prepare reports on your observations. Different people will get slightly different results if for no other reason than the initial randomization.

1. Access the *NetLogo User Manual* through the *Help* menu and read the “What is NetLogo?” page listed under **Introduction**.
2. Work through *Tutorial 1: Models*, *Tutorial 2: Commands*, and *Tutorial 3: Procedures* listed under **Learning NetLogo**. Examine also Luis Izquierdo’s useful short guide to NetLogo (<http://luis.izqui.org/resources/NetLogo-5-0-QuickGuide.pdf>).
3. Run some of the Biology models in the Models Library under the **File** menu. Some suggestions which may remind you of models in earlier chapters are
  - (a) Simple Birth Rates (Chapter 3)
  - (b) Wolf Sheep Predation (Chapter 4)
  - (c) Tumor (Chapter 5)
  - (d) Virus (Chapter 14)

4. Run some of the Social Science models in the Models Library under the **File** menu. Some suggestions which may remind you of models in earlier chapters are
  - (a) Rumor (Chapter 14)
  - (b) Voting (Chapter 6)
  - (c) Altruism and Cooperation (Chapter 18)
5. Reopen the Segregation model. Run the model 10 times, record Percent Similar for each run, then compute the mean and standard deviation (see Chapter 10) of these percents with *%-similar-wanted* set to
  - (a) 30%
  - (b) 50%
  - (c) 67%
6. Comment on the amount of variation in the observed values of Percent Similar for a fixed *%-similar-wanted*.
7. Explain why Percent Similar is always about 50% or higher no matter the value of *%-similar-wanted*.
8. Run the segregation model with very small values of *%-similar-wanted*, and discuss the results. How many ticks usually occur before there are no unhappy families when *%-similar-wanted* is 10% or 15%, for example? About how large must *%-similar-wanted* be before Percent Similar exceeds 60%?
9. Run the segregation model with *%-similar-wanted* first set to 70%, then to 80%. What do you observe as the outcomes? By adjusting the slider for *%-similar-wanted* between 70% and 80%, show that there is a “tipping point” below which everyone is happy but Percent Similar is more than 99% and above which Percent Happy never goes to 0.
10. The default setting for the segregation model has 2,000 families initially spread at random among 2,601 possible locations; thus approximately 77% of the patches will be occupied. Investigate the behavior of the model if you increase or decrease the value of *number* using the slider.
11. The segregation model assumes equal numbers of red and green families. In many actual communities, however, the proportions may be quite different, with distinct majority and minority groups. In the *setup* procedure, the NetLogo code controls the number of green families (and hence the number of red) with lines **21** and **22**:
 

```
ask n-of (number / 2) turtles
[ set color green ]
```
12. If you change (**number / 2**) to (**2 \* number / 5**), for example, this creates a population having 40% green families and 60% red ones. Experiment with different sized minorities to report on the segregation outcomes.
13. Our segregation model has been set up to deal with **two** different types of families, but most cities have significant numbers of multiple racial groups. It is easy to modify the NetLogo code to create a third (or a fourth, or a fifth . . . ) group. For example, if we change
 

```
ask n-of (number / 2) turtles
[ set color green ]
```

 to
 

```
ask n-of (3 * number / 10) turtles
[set color green ]
ask n-of (2 * number / 10) turtles
[ set color blue ]
```

 then we create red, green, and blue families making up 50%, 30% and 20% of the overall population, respectively. Implement the following:
  14. The Hispanic population of Miami, Florida constitutes about 59.1% of the residents. Approximately 19.6% are African American and 21.3% non-Hispanic. What does Schelling's model predict about residential segregation in Miami?
  15. According to Business Insider ([www.businessinsider.com/the-most-diverse-cities-in-the-us-2013-7](http://www.businessinsider.com/the-most-diverse-cities-in-the-us-2013-7)) in 2013, “NerdWallet, a financial website, calculated which United States cities had the most equal distribution of residents across four ethnic groups: Hispanic/Latino, White, Black and Asian/Pacific Islander.” NerdWallet reported that Vallejo, California, was the most equally distributed city in the country, with 23.7% Hispanic or Latino, 24.1% non-Hispanic white, 21% non-Hispanic black, 24.5% Asian or Pacific Islander, and 6.7% either other or of multiple ethnicities. What does Schelling's model predict about residential segregation patterns for Vallejo?
  16. NerdWallet found another city in California, Huntington Park, to be the least diverse place in the United States, with 97.9% Hispanic/Latino, 1.1% White, 0.1% Black, and 0.9% Asian/Pacific Islander. What does Schelling's model predict about residential segregation patterns for Huntington Park?
  17. NetLogo interprets the term *neighbors* to refer to the Moore neighborhood of a cell. To use the Von

Neumann neighborhood, you need to use *neighbors4*. For the segregation model, only two changes in the NetLogo code are needed. In the *update-turtles* procedure, alter lines 51 and 53 to read

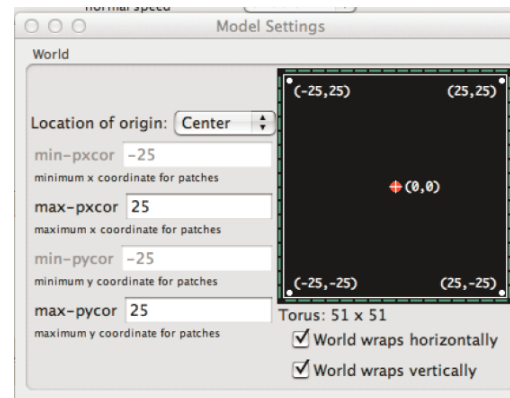
```
set similar-nearby count(turtles-on neighbors4)
set other-nearby count(turtles-on neighbors4)
```

18. Conduct experiments with this new version of the model and compare the results with those obtained using the Moore neighborhood.
19. Investigate the effect of changing the *nd-new-spot* procedure to implement alternative ways a family might seek out an empty patch. For example, try **lt** (or **left**) instead of **rt**. A more interesting search technique is to move systematically in a spiral (clockwise or counterclockwise) until it reaches an empty spot.
20. Run the segregation model with the box topology and see whether there are significant differences, in your view, of the outcomes with the torus topology. To

## SUGGESTED PROJECTS

1. Create a NetLogo version of Schelling's residential segregation model with at least three different kinds of families, each having a different *%-similar-wanted* value that can be adjusted with a slider by a potential user of the program. Run the program with different choices for the *%-similar-wanted* values and different proportions of the families. Discuss the results, paying special attention to how sensitive or insensitive the final patterns of segregation in housing are to changes in these values.
2. For a variation on Schelling's approach, examine Junfu Zhang's 2004 paper. Zhang uses a spatial game approach to show that persistent residential segregation emerges even if every person prefers to live in a half-black, half-white neighborhood.
3. Measuring residential segregation is a complicated task. Historically, the most commonly used tool was the *index of dissimilarity*, but researchers now use a variety of measures. See the 1988 paper by Douglas Massey and Nancy Denton for a review. Determine

change the topology, click on the settings button. The top half of the resulting window looks as follows:



Then uncheck the *World Wraps horizontally* and *World wraps vertically* boxes. Discuss the behavior of the model if you uncheck only one of these boxes.

which index (or indices) you think would be most effective and useful to employ.

4. Besides Schelling's model, there are alternative explanations for the persistence of residential segregation. Among these are the arguments that lower-income African Americans cannot afford to live in white neighborhoods; which are generally more expensive; that racial discrimination is still being practiced by real estate agents, mortgage lenders, and insurance providers; and that whites have less positive attitudes toward integration than other groups. How valid is Schelling's model as an explanation? Work carefully through William Clark's 1991 paper on testing Schelling's model.
5. In Exercise 8, you discovered that the Schelling model has a sudden change in behavior as the *%-similar-wanted* passes through some value between 70% and 80%. For a rigorous definition and study of the *tipping* phenomenon and its effect on residential segregation outcomes, a study of Zhang (2011) is a good place to start.

You can find a listing of references and suggestions for additional reading on the book's website, [www.wiley.com/college/olinick](http://www.wiley.com/college/olinick)