NetLogo Exercise

With a little help from our friend ‘copy and paste’, we are all going to quickly build a simple agent-based model.

The purpose of the exercise is for you to get a real feel for what an agent-based model is.

We are using a famous example model, the Segregation model, based on the work of Thomas Schelling. This is a simple toy model, but speaks to some of the concepts of complex systems we have covered, and is a model that has been extended in many ways to explore real world policy questions.

Following the instructions during the class, you will be copy and pasting sections of the code below into NetLogo, and running the model. We will also need to create some buttons, sliders, and plots in the model.

*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 turtle's 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*

 *; create turtles on random patches.*

 *ask patches [*

 *set pcolor white*

 *if random 100 < density [ ; set the occupancy density*

 *sprout 1 [*

 *; 105 is the color number for "blue"*

 *; 27 is the color number for "orange"*

 *set color one-of [105 27]*

 *set size 1*

 *]*

 *]*

 *]*

 *update-turtles*

 *update-globals*

 *reset-ticks*

*end*

*to update-turtles*

 *ask turtles [*

 *; in next two lines, we use "neighbors" to test the eight patches*

 *; surrounding the current patch*

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

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

 *set total-nearby similar-nearby + other-nearby*

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

 *; add visualization here*

 *if visualization = "old" [ set shape "default" set size 1.3 ]*

 *if visualization = "square-x" [*

 *ifelse happy? [ set shape "square" ] [ set shape "X" ]*

 *]*

 *]*

*end*

*to update-globals*

 *let similar-neighbors sum [ similar-nearby ] of turtles*

 *let total-neighbors sum [ total-nearby ] of turtles*

 *set percent-similar (similar-neighbors / total-neighbors) \* 100*

 *set percent-unhappy (count turtles with [ not happy? ]) / (count turtles) \* 100*

*end*

*; run the model for one tick*

*to go*

 *if all? turtles [ happy? ] [ stop ]*

 *move-unhappy-turtles*

 *update-turtles*

 *update-globals*

 *tick*

*end*

*; unhappy turtles try a new spot*

*to move-unhappy-turtles*

 *ask turtles with [ not happy? ]*

 *[ find-new-spot ]*

*end*

*; move until we find an unoccupied spot*

*to find-new-spot*

 *rt random-float 360*

 *fd random-float 10*

 *if any? other turtles-here [ find-new-spot ] ; keep going until we find an unoccupied patch*

 *move-to patch-here ; move to center of patch*

*end*