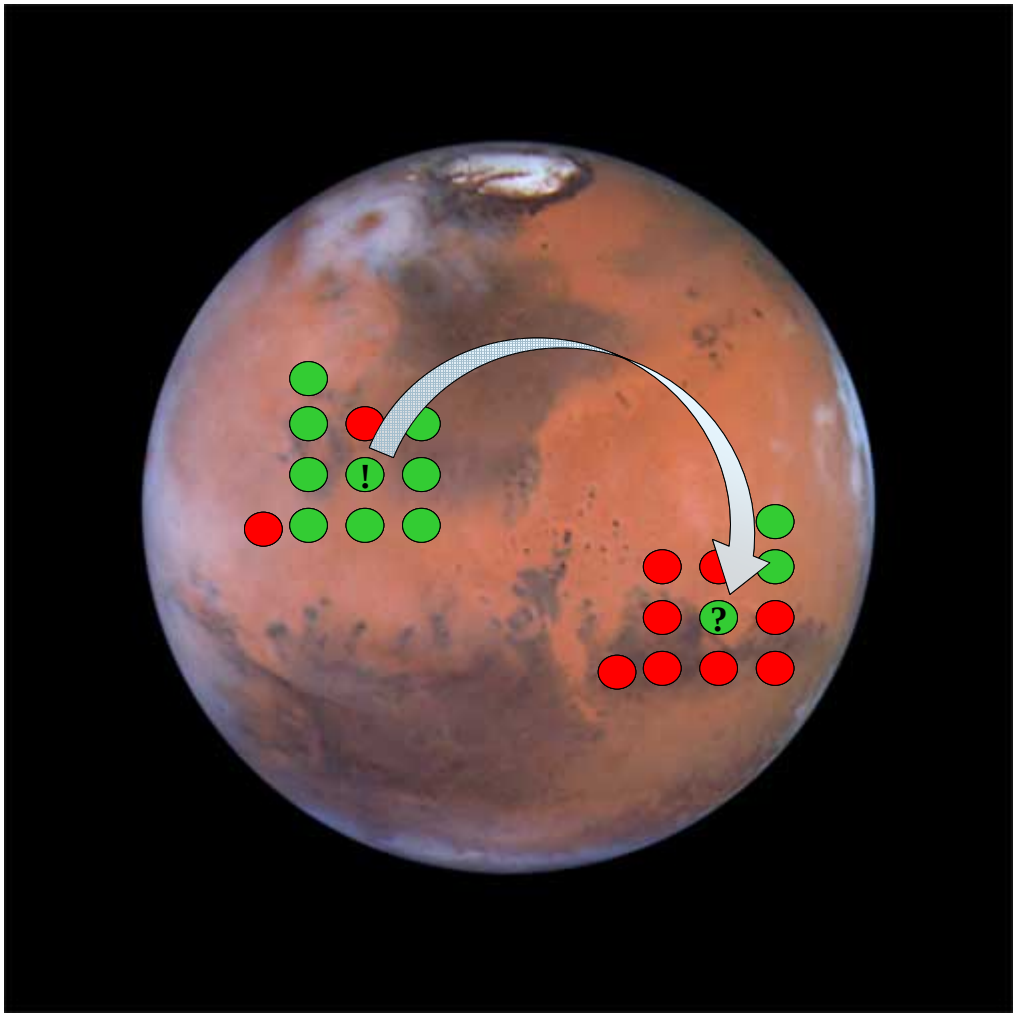
MARS User's Guide



Version 0.1
12/16/2004
By Lawrence Iannaccone and Michael Makowsky

Table of Contents

Introduction

MARS is a social science simulation of religious migration and conformity created within the Netlogo programmable modeling environment. Agents and the environment they exist within are programmed constructs whose attributes and the rules they operate by are a combination of programmed algorithms, endogenous variables, and exogenous parameters.

The focus of this guide is to help the user understand the model and what it simulates, and to create his or her own experiments by adjusting the observer controlled parameters and employing the full range of tools available.

All are encouraged to read the story behind the model, the basics of the observer interface, and the section on getting started. The tools and techniques described in the advanced section are just that - useful to researchers with specific goals, but not necessary for most experiments.

The Story

Agents exist within a two dimensional space (lattice) not unlike a checkerboard, Divided into a number of regions set by the observer (1 – 4). Agents move randomly, exogenously driven by the program to chosen patches, and upon arrival choose whether or not to maintain their religion (color) based on their original religion (when created), current religion (the one they had before moving), the religions of each of their new neighbors (community). The choice process amounts to evaluating a utility function for each possible choice of religion, and weighting original by the "origin" parameter, current religion by the "inertia" parameter, and the influence of neighbor religions by the "community" parameters.

Additionally, in more advanced models, agents have a choice to make regarding their congregation attendance, and quantity represented by shading. Attendance is our chosen variable for representation of agent religiosity, as it is the most accepted metric of religiosity and it has a clear cut community dynamic. Similar to religion selection, Attendance decisions are a utility evaluation based on original attendance, current attendance, and community attendance.

What is Available

MARS is available via the web as both 1) an immediately accessible Java Applet and 2) an downloadable program written for the Netlogo Platform.

Hardware and Software Requirements

MARS is a computationally light program and as such the hardware requirements are minimal. Any computer with 64 MB of RAM and 200 MHz processing speed should be able to run MARS. That said, the model will no doubt benefit from greater speed, especially if you wish to run batches of experiments.

Netlogo is entirely written in Java, and as such you must have some form of the Java Virtual Machine installed on your computer to run the program, and the Java plug-in to run the applet for MARS in your web browser.

If you're on Windows 98 or newer, you need to download the Java browser plugin from http://www.java.com/en/download/windows_manual.jsp.

If you are using any other operating system, please refer to the FAQ section of the Netlogo User Guide for how to proceed

Setting Up

Basics

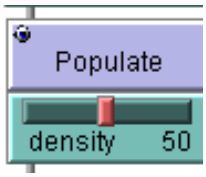
Before we can start talking about what kind of people we are going to put into the MARS world, we have to decide what kind of world it is going to be:

How many regions

Setting up regions is relatively simple. Drag the red bar in the slider to set the desired number of regions (0 – 4). Choosing zero regions will create a torus world where agents who move off the screen to the left will appear on the right side and those who move off the bottom will appear on the top.

After selecting a number of regions, the **Start/End** button must be pressed to establish the regional barriers.

How many agents in each region



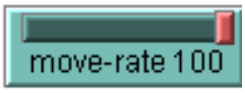
Agent populations are initialized based on desired densities. Drag the red bar in the slider to set the desired percentage population density of the region (0-100). Densities must be chosen for each region. As such, MARS starts in the upper right hand corner. The user chooses a density and then presses the **Populate** button, filling that region with agents. MARS will then move counter-clockwise to the next region, with the user repeating the procedure until all regions are filled.

Beyond this point the user has the option to run the model with whatever settings are in place

Manual Agent Placement

Agents May be added manually my pressing one of the agent type buttons (there is one for each agent type: red, green, blue, and yellow), and then placing the mouse pointer over the patch where the agent is to be placed and then left-clicking the mouse. Once the desired agents have been added simply re-click the button to turn the function off.

Movement Rate



Agents operating within the model once it is running will be exogenously forced to move on a percentage of their turns. The **move-rate** slider sets this percentage. How an agent moves, in terms of visualization, is dependent on the model **speed** setting (see below).

Model Speed

The speed the model runs at (how fast turns progress) can be adjusted with the **speed** slider. Settings 3 through 10 simply dictate how fast turns take in standard fashion. Settings 1 and 2 are special. Speed setting 2 will employ agents who flash white when it is their turn in the loop, flashing both before and after decisions and movement. Speed setting 1 will employ agents that flash white when it is their turn *and* they will "walk" to their new spot if they move.

Agents

Agents are the heart of the MARS model. Choosing the agent attributes for your model or experiment is perhaps the critical measure the user takes in setting up a simulation. These exogenously set parameters can be thought of as the agents' personalities; they are what give shape to the utility function that agents are maximizing.

Formal description:

Let ULjk(i) denote the utility that an agent will derive from choosing to become (or remain) type i, given that he was originally of type j, was previously of type k (in the last period, T-1), and is now located at patch L surrounded by NLi neighbors of type i. MARS operates with Moore neighborhoods, which entails a maximum number of neighbors, NL, of eight or less depending on how many border patches are adjacent to L. DL(i) is defined as the ratio [NL(i)/NL]. It can range from zero to one. The symbol "δij" is used in mathematical expression to switch terms "on" or "off" depending on whether i and j are equal or different. By definition, δ equals one if i = j and zero if $i \neq j$.

The utility of the function which agents attempt to maximize is as follows:

ULjk(i) = U(i, j, k, DL(i))

= (δij * [j_origin]) + (δik * [k_inertia]) + DL(i) * [i_community],

where [j_ origin], [k_ inertia], [i_ community] are parameter valuesi set by sliders which act as weightings in the agent's utility function.

Notes on parameter values:

[*j_ origin*] equals the utility associated with choosing to maintain (or return to) one's type-of-origin. If zero, then the agent displays no attachment to his religious upbringing.

[*k_ inertia*] equals the utility associated with *not* changing one's current type. If zero, then the agent displays no internalized "inertia" across periods.

[i_ *community*] equals the "social" utility that the agent receives from neighbors who share his type. If zero, then neighbors of type-i exert no "pull" on their co-religionists. This effect is weighted by the number of type-i neighbors *relative* to the number of potential (not actual) neighbors[ii].

Type and Rate

The most important agent attributes are **type** and **rate**. These two generic attributes were constructed as analogues to religion and rate of attendance (religiosity)[1]. Visually type and rate are represented by color and shading, respectively. With regards to an agent's type and rate, each type of agent (red, green, blue, or yellow) carries with it a set of

[1] There are, no doubt, a near infinite number of interpretations conceivable as to what behaviors and characteristics they could be interpreted as, but MARS was programmed to analyze dynamics of populations characterized by religion.

exogenously determined [user set] parameters. There is a *type-related* and a *rate-related* set.

Type-related

- **[j_ origin]** equals the utility associated with choosing to maintain (or return to) one's type-of-origin. If zero, then the agent displays no attachment to his religious upbringing.
- **[k_ inertia]** equals the utility associated with *not* changing one's current type. If zero, then the agent displays no internalized "inertia" across periods.
- **[i_ community]** equals the "social" utility that the agent receives from neighbors who share his type. If zero, then neighbors of type-i exert no "pull" on their co-religionists. This effect is weighted by the number of type-i neighbors *relative* to the number of potential (not actual) neighbors.

Rate-related

- **[j_ upbringing]** equals the utility associated with choosing to maintain (or return to) one's original attendance rate. If zero, then the agent displays no attachment to his religiosity upbringing.
- **[k_ habit]** equals the utility associated with *not* changing one's current attendance rate. If zero, then the agent displays no attachment to religious routine.
- **[i_ flock]** equals the "social" utility that the agent receives from regularly congregating with neighbors who share his type. If zero the agent feels no need to meet community religiosity standards.

NOTE: A different set of *_origin*, *_inertia*, and *_community* parameters are associated with each agent type. Hence, the "red" religion may be relatively effective in instilling loyalty (high *r_ origin*), but not very effective in holding on to converts (low *r_ inertia*), and only moderately effective in maintaining or attracting people via social effects (average *r_ community*).

By setting all *_ origin* and *_ inertia* parameters to zero and all *_ community* parameters equal to the same (positive) value, each agent will simply mimic the majority of his neighbors. This is equivalent to a standard agent-based model of social conformity. The result, over time, is complete uniformity within any given cluster of agents.

Running the Model

Starting

After selecting the number of **regions**, clicking **Start/End,** and using the **populate** button to fill each region with the desired number of agents, the **Run/Pause** button may be click to initiate the simulation.

While the simulation is running the user has a number of options available to her. The **speed** may be adjusted on the fly. Before the model is initiated or when the model is paused the user may also add additional agents as well as adjust certain functionality .

Options

The **options** button brings up a menu for the observer with tools useful before and during simulations. Seven Choices will appear:
Show, Search, Display, Replot, Skip Ahead, Quit, Halt

Show

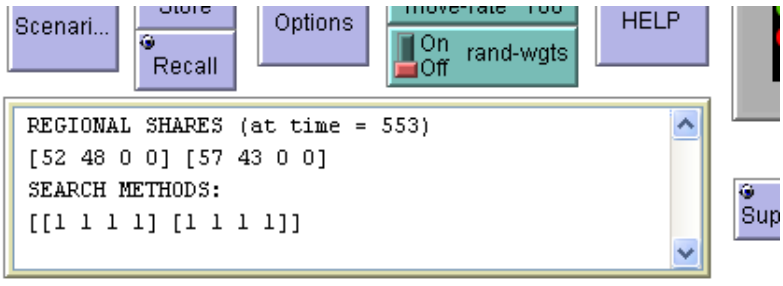
The **Show button** reveals two further sub-options.

Shares

This will cause the Command Center to display a matrix of the percentage share of each type in each region (see Fig X)

Search

The Command Center will display the search vector employed by agents (see Fig X).



Search

The **Search** option is one of the most interesting and useful in MARS. Agents when moving to a knew location must first *search* for their new location. How they search and how many spaces they enter into the analysis can be controlled by the user. After pressing the **Search** button the user will be given the option of 1, 2, 4, and 8. Each number represents the number of different locations the agent will pull information from as potential new locations. After choosing a number, the user will give the option of *n-type* or *any-type*. The agent can choose a space based on maximizing either the number of total neighbors, or the number of neighbors of *his specific type*. The second step is irrelevant if the user chose 1 earlier, as the agent has no means of comparison. With any number greater than one, however, the agent will maximize based on the criteria selected in the second step.

Display

Clicking on **Display** will bring up a menu allowing the user to choose to **Return** or **Restore.** The **Return** function will place every agent back on its original location while retaining all of their current attributes. The **Restore** function will restore every agent's attributes to their initial state, but leave them at their current location.

Replot

The **Replot** option clears the graphed plot and allows it to start over. This is useful with extended runs of the model, where the plot may become too condensed if it is allowed to display the run data in its entirety.

Skip Ahead

Running the model can be clumsy at times if the user wishes to proceed ahead a set number of moves. The **skip ahead** function in the **options** menu will progress the model forward the exact number of steps chosen. After clicking **skip ahead** the user will have the option of choosing *5, 10, 15, or 20* moves. This is especially useful when proceeding with the slower model **speed** settings.

Quit

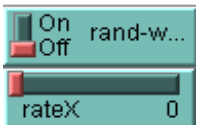
Pressing Quit takes you out of the **Options** menu

Halt

Pressing Halt stops the current simulation and program compilation

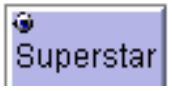
Advanced Tools

Random Weights



Several agent attributes are determined as observer set parameters. These parameters are discrete quantities unless the **Random Weights** option is turned on. **Random Weights** creates a "fuzzy" distribution of all observer set agent attributes, with the mean set by the observer, but with each individual agent receiving attributes falling within a normal distribution around that mean. This function allows for a increased stochastic nature in the model.

Superstar

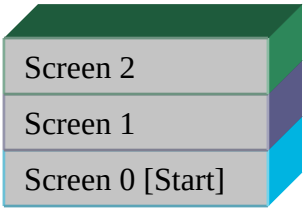


Pressing the **superstar** button and then clicking on any existing agent on the board will adjust that agent and give it a ten-fold 'pull' parameter compared to typical agents of its type. Note: The **rand-wgts** switch must be in the 'on' position for **superstar** to work.

Store/Recall

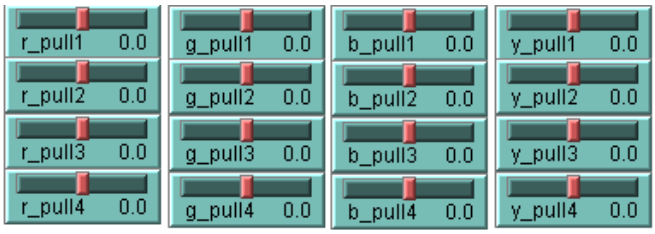
At any point in the progress of an experiment run, the observer has the option of pausing the model and **Storing the current** lattice and agent arrangement for later comparison. Storage works in a layering system, with each subsequent **Store** being laid onto the stack. Clicking **Recall** will cycle *down* through the layers of stored moments. Once the bottom layer is reached an additional click will return to the top, most recent screen shot.

Further Explanation. The observer initiates a model. If he clicks recall at any moment at this point he will return to screen 0. He can hit recall one more time to return to where he paused the screen (screen 1). However, should the observer begin to use the **Store** button, he will no longer be able to return to screen 0, instead he will only be able to cycle between the stored screen shots and the most recent screen (note: the most recent screen always represents the top layer.



Type-Specific Pull

In its default setting agents in MARS will calculate their utility function using a uniform set of parameters consistent to their chosen *type.* Their exists the option, however, to set the *pull* parameter to be customized such that different magnitudes are expressed relative to the interacting agents type. For example, red agents could be set to exert a pull of 1.0 to the first agent type (red), 1.5 to agent type 2 (green), 0.2 to agent type 3 (blue), and 0.0 to agent type 4 (yellow). As such one could imagine a scenario where a religion may be very attractive to one outside group, but grossly unappealing to another.



Setting Up Experiments – Behaviorspace

The best guide to using the Behaviorspace Functionality of Netlogo can be found in the html user guide at http://ccl.northwestern.edu/netlogo/docs/.

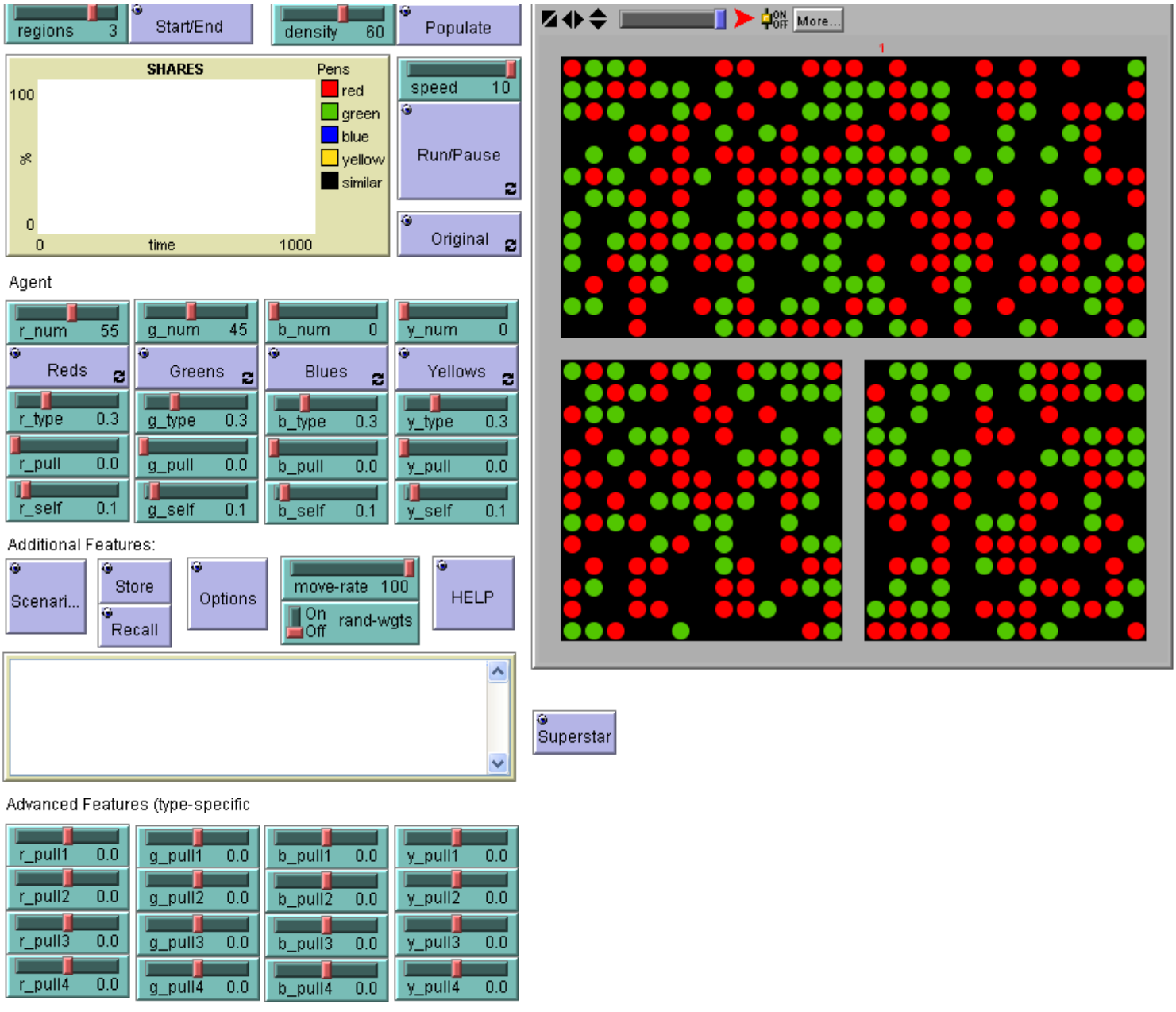
The value of Behaviorspace is to create test the significance of varying the parameters and rule sets that govern a model. Measures of distribution and central tendancy allow for the user to test not just what phenomena emerge, but the robustness of that emergence.

Time is measured in “turns” with one agent making a location, type, and rate decision each turn. How many turns an experiment runs should be, at least to some degree, a function of the total number of agents employed.

Data

Data sets created by Behaviorspace are comma-delimited text files. Only one “cell” of data may be tracked, but that cell can be a list of variable which is contained with the data file as a bracketed matrix. Refer to the Netlogo handbook for more information.

Full Interface Diagram



Trouble Shooting

Coming Soon

FAQ

Coming Soon

Appendix A

Quick Start

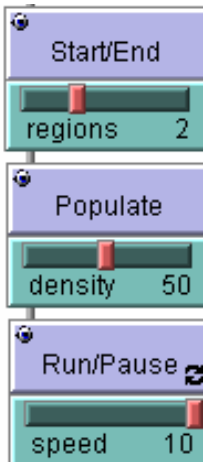
What kind of agents do you want?

This is for most experiments at the core of what you are trying to accomplish. There are two questions to be asked in the process:

1) How many different types (religions) of agents do you want?
2) In what relative ratios do you want them to populate your model?

These preferences are both met by adjusting the parameter slider adjacent to each color button. If you would like some portion of your agents to be red, simply set the **R_num** slider to a number greater than zero. Do this for each type you wish to be present. The ratio that will be realized in the initial conditions of your model will simply be that type's (color's) assigned number as a fraction of the sum pf all the **_num** slider settings.

Where do they go?



Designing your environment and populating it with agents is an obviously important, and relatively straightforward task.

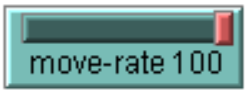
1) Choose your number of **regions** by moving the slider. You may have between one and four regions.
2) Click **Start/End**. Notice that if you chose multiple regions that lines of gray patches divide the lattice.
3) Choose a population **density.** You have between 1 and 100 percent density.
4) Click **Populate.** Each time you click populate a region will fill with agents to your prescribed density. If you have three regions you will have to click populate, then wait for it to fill three successive times

NOTE: If after all regions are filled you click populate an additional time the lattice spaces unoccupied will be filled by the density percentage chosen (ex if there are 10 unoccupied spaces and you click populate with 60% density, 6 of the remaining spaces will be filled).

Once these basic parameters have been set you may run the model by clicking **Run/Pause**. Clicking again will pause the model, and it can be restarted by clicking it yet again.

How much do they move?

Agents operating within the model once it is running will be exogenously forced to move on a percentage of their turns. The **move-rate** slider sets this percentage.



What are their attributes?

Finally, you wish to design your agents. For each type used within your model you have the option of assigning a set of relative values regarding the religion and attendance related attributes.

This is accomplished by adjusting a series of sliders that dictate the relative strength and weaknesses of your agents in six different categories.

- **[j_ origin]** equals the utility associated with choosing to maintain (or return to) one's type-of-origin. If zero, then the agent displays no attachment to his religious upbringing.
- **[k_ inertia]** equals the utility associated with *not* changing one's current type. If zero, then the agent displays no internalized "inertia" across periods.
- **[i_ community]** equals the "social" utility that the agent receives from neighbors who share his type. If zero, then neighbors of type-i exert no "pull" on their co-religionists. This effect is weighted by the number of type-i neighbors *relative* to the number of potential (not actual) neighbors.

- **[j_ upbringing]** equals the utility associated with choosing to maintain (or return to) one's original attendance rate. If zero, then the agent displays no attachment to his religiosity upbringing.
- **[k_ habit]** equals the utility associated with *not* changing one's current attendance rate. If zero, then the agent displays no attachment to religious routine.
- **[i_ flock]** equals the "social" utility that the agent receives from regularly congregating with neighbors who share his type. If zero the agent feels no need to meet community religiosity standards.

[i] For clarity it can be assumed that when we refer to a "parameter" or "parameter value" we are describing an exogenously controlled variable that is determined by the observer. Thus we can run a variety of "experiments" simply by adjusting the various parameters.

[ii] Note that a different set of *_origin*, *_inertia*, and *_community* parameters are associated with each agent type. Hence, the "red" religion may be relatively effective in instilling loyalty (high *r_ origin*), but not very effective in holding on to converts (low *r_ inertia*), and only moderately effective in maintaining or attracting people via social effects (average *r_ community*).

By setting all *_ origin* and *_ inertia* parameters to zero and all *_ community* parameters equal to the same (positive) value, each agent will simply mimic the majority of his neighbors. This is equivalent to a standard agent-based model of social conformity. The result, over time, is complete uniformity within any given cluster of agents.