## Evolution of Teaching Simulation

## MODEL GUIDE

Written by Carrie Easter

(Last Updated: 17/04/2018)

A useful guide on how to use the social learning / teaching NetLogo simulation available from the University of Leeds Research Data Repository (<https://doi.org/10.5518/360>).

This guide is for anyone wishing to use the simulation for their own research or interests.

The purpose of this simulation was to investigate how altering certain parameters affects the likelihood that teaching (actively increasing the probability of learning in other, naïve individuals) will evolve. Please see the Model Summary and Processes documentation available at <https://doi.org/10.5518/360> for a detailed description of how the model functions.

In order to run this simulation, users must download the free software from <https://ccl.northwestern.edu/netlogo/download.shtml>. It is recommended that this model is run in NetLogo 6.0.2 or 6.0.3, due to slight changes in syntax from older versions.

Please note: The model was originally created in NetLogo 5.2.0 and then made compatible with 6.0.2 (and 6.0.3) using the program’s automatic conversion system. It is possible to convert the model back to NetLogo 5.2.0 syntax by opening the file in this version of NetLogo and agreeing to the conversion prompt. If the model is run on future versions, however, alterations to the code may be required to make it compatible. Please see the transition guide (<http://ccl.northwestern.edu/netlogo/6.0/docs/transition.html>) for information about changes between versions.

Please give full credit if using this model or a modified version of it in your own research.

### **NetLogo Jargon**

‘World’ = the simulated space

‘Turtles’ = individuals / agents; represented in this model as moving triangles

‘Patches’ = the squares that make up the world

‘Interface’ = the screen where the world and changeable variables are located (when not in this view, navigate here by clicking the ‘interface’ tab on the top-left of the window)

For further clarification on generic NetLogo features and general information on the software itself, see the NetLogo Programming Guide

(<https://ccl.northwestern.edu/netlogo/docs/programming.html>).

### **Changeable Variables on the Interface**

Below is a brief explanation of the changeable variables (e.g. sliders, switches and input choosers) on the Interface.

|  |  |
| --- | --- |
| *restart* | Once all the interface variables are set up with the values you wish to test, press this button to set up variables ready for data collection and set up the first iteration of the model. Note: This only needs to be pressed once for a given set of parameter values. The model is programmed to automatically set up the next iteration until it reaches *number-of-iterations.* |
| *go* | Press this button to run the simulation. If this button is pressed again, the simulation will ‘pause’ and will continue if pressed again. This button should remain pushed down until all iterations (*number-of-iterations*) have been run and the data exported. The *restart* button does not need to be pressed again until *go* button lifts itself again. |
| *number-of-iterations* | Box to input how many iterations of the simulation, using the current variable values, should be run before the simulation stops. |
| *turtle-number* | Number of turtles (agents) in the simulation. |
| *number-of-subgroups* | When >1, agents are separated into subgroups (or ‘families’), with 0 relatedness between agents in separate subgroups. |
| *number-of-options* | Number of possible actions agents can attempt to perform on a food patch. If more than *number-of-food-patch-types*, some of the actions that agents will attempt to use won’t be correct for any of the food patches in the world – which can be used to represent cases where there is a large amount of behavioural flexibility in the actions an individual may perform, but few of them are likely to produce a successful outcome. |
| *number-of-food-patch-types* | Number of ‘correct actions’ to be distributed between the food patches in the world. Each individual food patch is allocated one of these correct actions. This value should not be greater than *number-of-options*, unless one wants to model a scenario where some correct actions are impossible to learn. |
| *asocial-learning-probability* | The probability of attempting to learn asocially (independently) when given the opportunity. If an agent attempt to learn asocially, it will choose a random action from *number-of-options* to perform on the food patch. Agents will attempt to learn asocially when alone on a food patch it does not know the correct action for, or after unsuccessfully attempting to learn socially and/or via teaching. |
| *social-learning-probability* | The probability of learning from another, informed agent on the same food patch. The agent will cycle through any available informed individuals on this food patch and learn the action they are using at this probability. Learning from a non-teacher depends entirely on this probability, while learning from a teacher depends both on this probability and other factors, such as the strength of teaching. |
| *selective-teaching* | The strength of teaching for agents with a selective-teaching (Ts) phenotype. The probability of learning from a selective-teacher depends on the *social-learning-probability,* this *selective-teaching* strength and the level of relatedness between itself and the teacher. |
| *non-selective-teaching* | The strength of teaching for agents with a non-selective-teaching (Tns) phenotype. The probability of learning from a selective-teacher depends on the *social-learning-probability* and this *selective-teaching* strength. |
| *seeded?* | Should the population be seeded with an agent who automatically knows all possible correct feeding actions? One can choose the population to be either ‘unseeded’, ‘seeded with one informed individual’ or ‘seeded with one informed individual per subgroup’. |
| *initial-selective-freq \** | Frequency of selective-teaching alleles in the background population used to generate alleles in the simulated population. |
| *initial-non-selective-freq \** | Frequency of non-selective-teaching alleles in the background population used to generate alleles in the simulated population. |
| *initial-non-teaching-freq \** | Frequency of non-teaching alleles in the background population used to generate alleles in the simulated population. |
| *dominant* | The most dominant allele in the population. Used to generate which phenotype an agent should have based on its genotype. |
| *second-dominant* | The second most dominant allele in the population. |
| *recessive* | The recessive allele in the population. |
| *number-of-generations* | How many generations are present in the population at any given time. |
| *generation-time* | The number of timesteps before the oldest generation dies and the youngest generation breeds to produce the next generation. |
| *stop-at-generation* | How many generations should be reached before this iteration of the simulation stops and the next is set up. |
| *kin-follow-preference* | Determines how much the relatedness between two agents affects the likelihood of one following the other. Larger values promote stronger kin association within the population. A value of 0 means agents are no more likely to follow kin than non-kin. A value of -1 means agents are more likely to follow non-kin. |
| *only-follow-parent-offspring?* | When set to ‘on’, agents are constrained to only follow their parents or offspring (i.e. individuals most related to themselves). The value of the *kin-follow-preference* therefore does not matter in these cases. When set to ‘off’, agents are allowed to follow any other agent, with the effect of relatedness being entirely determined by the *kin-follow-preference.* |
| *following-radius* | If an agent decides to follow another agent, any agents within this radius can potentially be chosen as its leader. |
| *following-probability* | The probability during any given timestep that an agent will try and follow another individual, providing there is at least one other agent within its *following-radius.* |
| *deciding-time \*\** | The number of timesteps an agent will remain on a food patch before it attempts to learn. |
| *feeding-time \*\** | The number of timesteps an agent will remain on a food patch while it feeds. |
| *food-patch-distribution* | Food patches can be distributed in two ways: ‘scattered’ or ‘clustered’. Scattered food patches are distributed randomly over the world. Clustered food patches are located at the top-left of the world and are all clustered together to form a large circle. |
| *food-patch-radius* | Only applicable if the *food-patch-distribution* is clustered. Determines the radius of the food patch cluster. |
| *food-patch-bias* | Only applicable if the *food-patch-distribution* is clustered. Once an agent has visited the food patch cluster once, it will learn its location. This value determines how strongly an agent which knows the location of the cluster will move towards the cluster when not currently located on it. |
| *food-patch-cover* | Only applicable if the *food-patch-distribution* is scattered. The percentage cover that the food patches take up in the world. |

\*Because second-generation agents gain only one of their alleles from the background population, gaining the second at random from their mother, the actual allele frequencies of the simulated population will vary on set up from these inputted parameters. The simulation is set to record the initial allele frequencies of the simulated population in its output.

\*\*Increasing the deciding and feeding times will keep an agent on a food patch for longer, thus giving other agents time to reach the same food patch to promote learning.

### **Exportation of Data**

This model is set up to record the following data, for all iterations run, in a variable called *dataset*:

* The maximum generation at the end of each iteration
* The initial selective-teaching allele frequency
* The selective-teaching allele frequency at the end of each iteration
* The initial non-selective-teaching allele frequency
* The non-selective-teaching allele frequency at the end of each iteration
* The initial non-teaching allele frequency
* The non-teaching allele frequency at the end of each iteration
* The initial number of individuals with a selective-teaching phenotype
* The number of individuals with a selective-teaching phenotype at the end of each iteration
* The initial number of individuals with a non-selective-teaching phenotype
* The number of individuals with a non-selective-teaching phenotype at the end of each iteration

If you wish to export this data from the model, you will need to alter the code so that a .csv file can be exported to a particular directory of your computer. To do this, go into the ‘Code’ tab at the top of the NetLogo window, find the ‘Reset and Stop Procedures’ (this can be found easily by using CTRL + F and finding ‘file-name’). Find the line:

*; csv:to-file (word "C:/PathName/" file-name) dataset*

Change *C:/PathName* to your file directory, making sure to leave the rest of the code, including the speech marks and the forward slash at the end of the directory, intact and uncomment the line by removing the semicolon from the start.

Notice that the code is set up so that *file-name* changes depending on the inputted *only-follow-parent-offspring?,* *selective-teaching, kin-following-preference, social-learning-probability* and *asocial-learning-probability* values. This means a unique file name will be given to all .csv files for sets of simulations with different values for these variables. This ensures files are easily identifiable and are not overwritten. If other parameters are tested, however, file names will need to be altered to account for this.

Alternatively, if you wish to name each file individually with different names, you can leave this code alone and export the file via the command centre. To do this, wait until all your iterations have run (i.e. until the *go* button lifts itself up again), and type the following into the command centre at the bottom of the NetLogo interface:

*csv:to-file "C:/PathName/FileName.csv" dataset*

Making sure to replace *PathName* and *FileName* with the correct directory and file name. Ensure *.csv* remains at the end of your file name or you will receive an error.

### **Further Alterations of the Model**

For users with a knowledge of NetLogo coding, further aspects of the model, beyond the changeable variables in the interface, can be modified by changing the code directly. The code for the model can be found under the ‘Code’ tab in the NetLogo window. For further information on NetLogo coding, see the NetLogo Dictionary (V6.0.3 <https://ccl.northwestern.edu/netlogo/docs/dictionary.html>) and Programming Guide (V6.0.3 <https://ccl.northwestern.edu/netlogo/docs/programming.html>). If modelling in NetLogo 5.2.0, refer to the Transition Guide (<https://ccl.northwestern.edu/netlogo/docs/transition.html>) to see the differences between the versions.

Feel free to contact the model creator, Carrie Easter, at [bs13ce@leeds.ac.uk](mailto:bs13ce@leeds.ac.uk) to discuss any aspects about the model, including how it can be altered to include other aspects.