Honors Thesis

Ben Roos

March 16, 2015

Table of Contents

1 Introduction

2 Model

2.1 Population

The simulation uses a Moran Model to describe a finite population of 1024 individuals. The simulation tracks a single allele for each individual. This allele is selected independently for each individual, and alleles are represented using numbers that begin at 0 and may be infinitely high. Each individual is geographically isolated from others in a grid that is 32 cells long and 32 cells wide. Each individual takes up only one cell, and each cell contains only one individual. Thus, the size of the population will never vary from 1024 at any step of simulation.

2.2 Initializing the Population

When the simulation initializes, a population is randomly generated. The allele number for each individual in the population is selected sequentially, beginning with the upper left corner of the grid, proceeding through each row from left to right, and ending with lower right corner of the grid. The first individual is immediately assigned the allele number 0. With probability $\frac{i}{i+\theta}$, where i is the number of individuals in the population generated so far, and theta is equal to $2N\mu$, where N is the number of individuals in the population and μ is the mutation rate, subsequent individuals in the population will be an assigned an allele number equal to that of a randomly selected member of the population. With probability $1-\frac{i}{i+\theta}$, subsequent individuals are assigned to new allele numbers, momentarily becoming the only member of the population with the given allele. Thus, individuals that are assigned alleles earlier in the initialization process are more likely to be assigned new alleles, while individuals that are assigned later in the process are more likely to be assigned an allele from the population.

2.3 Simulation

At each step of the simulation, a randomly selected individual dies, leaving its corresponding cell momentarily empty. A parent allele is then randomly selected from the immediate neighbors of the empty cell. With probability 1 - μ, the new individual will have the same allele as its parent, and with probability Mu a mutation will occur, and a unique allele number will be assigned to the new individual. 2000 steps occur during each generation, and the population is redrawn in the visualization once for each generation.

3 Implementation

3.1 Design

3.1.1 Layout

The application contains three components: a control panel, with which users can control the simulation and its parameters; a grid, on which the simulation is displayed; and graphs, on which statistics regarding the simulation are shown.

3.1.2 Input Paremeters

The control panel includes a text field in which users can enter the mutation rate that will be used during the simulation. The default mutation rate is 0.00001, and is displayed in the text field when the page loads.

3.1.3 Running the Simulation

To start and pause the simulation, users click on a single button, which will display either "Start" or "Pause" depending on whether the simulation is running. Users can reset the simulation by clicking "Reset." This control generates a new initial population, reinitializes the grid, and clears the graphs.

3.1.4 Export

If users would like to export a JSON representation of the population at a given point in the simulation, they can click "Export" while the simulation is paused. The format of the JSON object is { cells: [{color: hex_val, allele: [-2, ∞), mutationNumber: [0, ∞}, {...}, ...] }. Each cell object includes the CSS hex value for the color associated with the allele, the allele number, and the mutation number, and the JSON object that is created is a list of these cell objects.

3.1.5 Barriers

Users also have the ability to create a barrier in a cell by clicking on the desired cell. When a barrier is created, the allele number is set to -1 and color associated with the cell is #000000 (black). Barriers cannot act as parent cells, so they are never replicated, and barriers cannot die during the simulation and subsequently be replaced. Thus, using barriers, users can construct physical constraints around which the movement of alleles of subpopulations is restricted. Users can create multiple barriers by clicking on multiple cells or by clicking and dragging. If users want to remove a barrier, they can click on the chosen cell a second time. This action will set its allele number to -2 and its color to #FFFFFF (white). In this state, the cell can be considered empty; neighboring cells can replicate their alleles into an empty cell, but it cannot be chosen as a parent. Barriers can be cleared either by clicking or by clacking and dragging.

3.1.6 Barrier Templates

If users are running a local copy of the application, they can also create templates of barriers so that barrier patterns can be applied with a single click. In order to create a template, users must first create the barrier pattern that they would like to save by clicking, or clicking and dragging, over the cells that should be barriers. Users can then export the grid to a JSON object by clicking "Export." They can then locate BarrierTemplate.js in their local copy of the application. They should replace the JSON object that is assigned to the barrierTemplate variable with the JSON object that they just downloaded. Now, in their

local copy, the "Template" button on the control panel will refer to the barrier pattern they defined.

3.1.7 Forced Mutations

Users can force a mutation to occur in a similar manner to creating barriers. By holding the SHIFT button while clicking, or clicking and dragging, they can force grid cells to contain individuals with mutant alleles. When a user performs this action, the allele number is changed to the mutant allele number and a new color is assigned to the cell.

3.1.8 Graphs

Two graphs are displayed to the right of the grid. The upper graph displays the number of unique alleles in the population over time and the lower graph displayed the allele frequencies of each unique allele in the population over time. In the lower graph, the colors associated with each allele match the colors of the simulation.

3.2 Architecture

The simulation was written entirely in JavaScript, using jQuery, Bootstrap, and some CSS. In order to logically structure the code, the application was built using the revealing module design pattern. Using this design pattern, each module refers to a distinct portion of the application, and enables each module to contain public and private methods. Similar to a singleton pattern, the revealing module pattern creates a single instance of each JavaScript object. To organize this application's source code, one module was created for each major component: the control panel, the grid, and the two graphs.

3.2.1 Control Panel

The control panel refers to the series of buttons and text field above the grid.

3.2.2 Grid

3.2.3 Graphs

4 Lesson Plan

As mentioned previously, this application may be particularly useful for educational uses. Because the application can be run in any modern web browser, on any platform, students should easily be able to access the application and use it to improve their understanding of several concepts from genetics and evolution classes, including genetic drift, gene flow, and speciation. After explaining the model used by the simulation, students should answer the following questions.

4.1 Genetic Drift

The population represented in this simulation consists of only 1024 individuals, and the population does not expand as the simulation is run. Do you expect that genetic drift will have a significant effect on this population? Explain your answer. Run the simulation for approximately 100 generations. Has genetic drift occurred? How does this result compare with your answer to the previous question?

Genetic drift is caused by errors in sampling. We would only expect these errors to have a significant effect on allele frequencies in the population if only a small number of individuals are producing the next generation. Because the population in this simulation is small, we would expect genetic drift to have a significant effect.

If you allow the simulation to run indefinitely, what effect would you expect on allele frequencies in the long term? Allow the simulation to run for 300 generations. What effect does this have on allele frequencies? How does this result compare with your answer to the previous question?

Over many generations, random genetic drift will eventually lead to allele fixation. We

would expect that if the simulation is run indefinitely, the population would eventually reach a point of homogeneity, in which one allele represents the entire population.

Using the population from the previous step, create a subpopulation in one corner of the grid (recall, you can create barriers by clicking, or by clicking and dragging, on any cell). Can you think of a physical event that would create a subpopulation like this?

Would you expect the allele frequencies in this subpopulation to differ from the entire population? Using a rough visual estimate, do the allele frequencies of the subpopulation match the allele frequencies of the entire population?

What caused the difference in allele frequencies between the sample population and the population as a whole? Compare this phenomenon to the effect of genetic drift.

The subpopulation is made up of very few individuals, and is drawn from an already small population. As a result, the differences in allele frequency are simply due to sampling error. This has a very similar effect to genetic drift, and if this subpopulation continues to be isolated from the larger population, the difference in allele frequencies is called the founder effect. The population that results from this subpopulation may have drastically different allele frequencies to the original population

If the simulation is run for several more generations, what effect would you expect to occur on the genetic diversity of the subpopulation? Run the simulation for several more generations and compare the results with your prediction.

This population has suffered a loss of genetic diversity from the gene pool of the whole population. We would expect allele fixation to occur more rapidly in this subpopulation

Reset the grid to create a new founding population. Then apply the default template (by pressing the "Template" button). This should create four spatially isolated subpopulations.

What do you expect to happen to the allele frequencies of these subpopulations when you run the simulation? Would do you expect to happen to the allele frequencies of the entire population? Run the simulation for 200 generations and compare the results to your predications.

We would expect the four subpopulations to reach allele fixation quickly, but it is likely that each subpopulation will be represented by a different allele. Thus, the population as a whole would have 4 equally prevalent alleles.

4.2 Gene Flow

Using the population created in the previous question, run the simulation until allele fixation occurs for every subpopulation. How many generations did this take?

Remove all barriers between the upper and lower subpopulations, such that there is still a barrier dividing the population horizontally, but not vertically. What do you expect to happen to the allele frequencies of the subpopulations?

Create a path for migration between the 2 subpopulations by removing 2 of the barriers between them (the path should be 1 cell wide). What do you expect to happen to the allele frequency of the subpopulations? Run the simulation for 100 generations and compare the results with your predications.

Some genetic diversity will be introduced to each subpopulation, but because migration is so restricted, it is likely that each subpopulation will maintain allele fixation for some time

Remove all barriers from the grid. What do you expect to happen to each subpopulation? What do you expect to happen to the population as a whole?

5 Discussion