Simulating Sexual Reproduction

In this lab we will create models of sexual reproduction, and examine how the frequency of alleles change with population size and relative fitness. We will use the model we create to examine some of the limitations of the theoretical models we discussed in class.

Our model will consist of hermaphroditic turtles with two alleles "A" and "B" for a gene which combine into one of three genotypes "AA", "AB" and "BB". Turtles will mate with a randomly chosen turtle in their neighborhood and produce one offspring that is a cross between them and their partner. If the population grows beyond the initial population turtles will be randomly "retired" from the game.

The Model

Here are the parameters for your model:

Make a `setup` procedure where you create a number of turtles (specified with a slider), which are located at random locations on the screen. Give each turtle a `genotype` variable which will be a two letter string (a string is a list of characters in quotes) specifying two alleles. Start with a proportion `p` (defined as a number between 0 and 1 with a slider) of homozygous "AA" turtles and let the rest be homozygous "BB". There will be no heterozygous turtles to start. I'll let you decide the colors for your turtles, but it would be a good idea to let the heterozygous turtles that eventually emerge have an intermediate color between the two homozygous genotypes.

You may want to assign color to the turtles in a separate procedure such as `update-phenotype`, since you will later be giving turtles of each phenotype different fitness values and this will be good way of assigning these values to new turtles that are born.

Define a `go` procedure in which turtles do the following:

- Move on a random walk for a number of steps that you define by a slider on the interface.
- Mate (ie. become pregnant) with a randomly chosen turtle on a neighboring patch (if there is one.)
- Give birth if they are pregnant (not all will be pregnant since not all will have neighbors).

It is important that all turtles complete each of the procedures before any move on to the next one, so put each of these in a separate ask turtle command. To finish the `go` procedure, if there are an excess of turtles beyond the initial number, ask that number to die. You may think it is better to kill off those who are older, or those who didn't mate or some other criteria, but if you have qualms about making these kinds of god-like decisions, just make it random and chalk it up to fate.

The key procedure to define above is the `mate` procedure. In this procedure the turtle must select a `partner`, and, if there is one, assign a turtles-own `zygote-genotype` variable by joining a randomly chosen allele from itself and adding it to a randomly chosen allele from the partner. To join two strings you concatenate them with `word` reporter. For example:

```
set my-string word "net" "logo"
```

will assign the string “netlogo” to the variable `my-string`. Note: you refer to different characters in a string in the same way as you do for a list. That is you use the `item` reporter. Do not actually create a new turtle with this genotype in this procedure, instead set a turtles-own `pregnant?` variable to true. You will be creating new turtles later in the `birth` procedure (if you don’t wait, you run the risk of an over-eager neighbor engaging your newly created offspring in carnal acts before they are legal).

In the `birth` procedure, which you call for pregnant turtles, create one turtle with genotype equal to the `zygote-genotype` and update the phenotype. (Be aware that the genotype variable could be one of four different strings: “AA”, “AB”, “BA” and “BB”. Make sure you assign the same color (and later fitness) to both “AB” and “BA” genotypes, since these are biologically identical. Make sure that the new offspring is born with a new heading and is not directly under the parent turtle. Both the parent and child should have the `pregnant?` variable set to false.).

Finally create monitors counting three different genotypes as a proportion of the total population Also create a plot showing each of the genotype frequencies.

For you assignment you will explore how the parameters of the model affect the genotype frequencies. You should compare the results of the model with predictions from the genetics models we discussed in class and be prepared to discuss the reasons for differences you observed. Answer assignment questions in the information section of your model.

Assignment

1. What genotype frequencies to you expect from theory if the initial allele frequency is equally divided between “A” and “B”? Run the simulation with your screen size set at 50 by 50, p set at 0.5, initial population 100 and the random walk steps to be about 10 (which should ensure random mating). Run the simulation several times with these parameter values. Do you obtain the results predicted by theory? (ie do the genotype frequencies reach a Hardy-Weinberg equilibrium?) Explain you observations.
2. In the previous run, you will have noticed that one of the alleles eventually fixes, and the other dies out. This is an example of genetic drift, which is quite pronounced for small populations. Repeat the simulation several times with population set at 500, 1000, 2000 and 5000. Run them long enough so that you can see if one allele or the other will fix? What do you observe? In which cases do the genotype frequencies best fit that predicted by the Hardy-Weinberg equilibrium? Does the fixing time fit within the expectations of genetic drift? (ie proportional to N, the population size.))
3. Repeat some of the simulations from questions 1 and 2 with different initial proportions of the alleles. How does this quantitatively affect the behavior of the model? Are the genotype frequencies the same as predicted by theory? How does it affect the time it takes for an allele to fix?
4. Now you will examine one way in which mating might not be random. Turn-off the motion of your turtles by setting the random walk steps to 0. Reset p=0.5. Systematically study the behavior of the turtles for population values ranging from 100 to 5000. Describe the behavior in terms of demographics, in terms of genetic drift, and in terms of gene frequencies (do they fit the Hardy Weinberg-Equilibrium? Is any equilibrium reached for large population numbers? What happens to the population of heterzygotes?).
5. Ok, now that we have explored the model with no selection, lets add some fitness parameters to the model. Define sliders for `AA-fitness`, `AB-fitness` and `BB-fitness`. These will be values between 0 and 1. In your `update-phenotype` procedure assign the appropriate fitness value to each genotype. Use the fitness as a criterion for mating. Each turtle will allowed to mate only if a randomly chosen floating point number between 0 and 1

is less than the fitness value.

6. Now you can test how fitness affects the frequencies of the different genotypes. Choose the population setting so that genotype frequency changes are not dominated by genetic drift. Make sure the random walk `steps` is high enough for random mating. For each of the following scenarios experiment with different initial values for p
   (a) Allele “A” is strictly dominant, and the homozygous “BB” is least fit.
   (b) Allele “A” is strictly dominant, and the homozygous “BB” is most fit.
   (c) The homozygous genotypes are the most fit.
   (d) The heterozygous genotype is the most fit.

   Which of the scenarios have stable equilibria in which both alleles persist? If heterozygotes have fitness of 1 and the homozygotes have a fitness of 0.8, what do you expect the equilibrium distribution to be? Does the model predict this?
7. Repeat the analysis above with stationary turtles. What difference do you observe?
8. The fitness values in question 6 are assigned by you and are therefore somewhat arbitrary. Lets change the way the fitness values are determined as follows:
   (a) Let “A” be an allele for cooperation and “B” be an allele for defection “AA” cooperate, “BB”s defect and for each time step “AB”’s choose to cooperate with some probability *x*, which you can set with a slider. (note *x=1* means “A” is dominant, and *x=0* means “B” is dominant.
   (b) Have each agent interact with all its neighbors as follows: All agents count how many neighbors they have and how many cooperating neighbors they have. All agents get gain *b* `points` for every cooperating neighbor they have, and cooperating agents lose 1 `point` for every neighbor they have. Make *b* a slider from 1 to 10.
   (c) The `fitness` of an agent should now be defined as follows: Each agent find the point value of the neighbor with the most points. Call this value `max-points`. The fitness of an agent is then defined as an agents `point` value divided by `max-points`.
   (d) Now test this model for different values of *b and x,* and with agents moving at different speeds. What are the conditions for which cooperative behavior is highest?
9. save your file as “lastname_firstname_lab8.nlogo. Add comments to your code, include answers to the homework questions and then upload your lab to the moodle site.