Using Islands_3.1 to see drift and selection in action Anth/Biol 5221, 21 March 2022

Islands_3.1.py implements a simple Monte Carlo simulation of the genetic evolution of a single diploid locus in 100 small, isolated populations, each consisting of N=25 individuals. It displays all 100 populations graphically, showing each individual's genotype as a colored square (yellow for the A1A1 homozygotes, green for the A1A2 heterozygotes and blue for the A2A2 homozygotes). The islands are updated "in real time" as the simulation advances, generation by generation. New mutations do not occur during the simulations, so once a population fixes for A1 or A2, it remains entirely yellow or blue.

Mating occurs at random, so the *expected* allele frequencies next generation are equal to the *actual* allele frequencies this generation, as *adjusted* for any fitness differences among the genotypes: W(A1A1) = 1, W(A1A2) = 1 + h*s, W(A2A2) = 1 + s. Drift occurs because the genotypes of the 25 offspring produced each generation are generated by ***sampling*** A1 and A2 gametes ***at random*** from the island's gamete pool: If random( ) < p2p (the expected frequency of A2 next generation), then we draw an A2; otherwise A1. You can see the code that implements this sampling in lines 198-210 of islands_3.1.py.

To run the program, you will need to put the ***graphics.py*** module in the folder where islands_3.1 is located. Graphics.py was written by John Zelle, a professor of computer science at Wartburg College in Iowa. You should read his documentation (***graphics.pdf***) if you want to use it in programs of your own.

In islands_3.1, each population begins with 24 A1A1 homozygotes and one A1A2 heterozygote, so the initial frequency of the "new" A2 allele is p2 = 0.02 (one in 50). Future versions will allow for adjustable starting frequencies (and other biological complications such as migration), but the case of a single newly-mutated allele is by far the most interesting because that's the most critical moment in the history of any beneficial or harmful new allele. You can and should adjust the selection parameters ***s*** (the fitness advantage of the A2A2 homozygotes) and ***h*** (the dominance parameter, determining the relative fitness of the A1A2 heterozygotes). These parameters are set on lines 9 and 10 (the first non-comment statements in the program). Note that negative values of ***s*** will make A2 a deleterious allele (as long as ***h*** is between 0 and 1), while positive values of ***s*** make A2 advantageous, relative to A1.

With islands_3.1.py open in IDLE and the focus on the program window, just press the F5 function key (or use the RUN dropdown) to launch a run. This should cause the graphical window to appear on your screen. You can drag it any place you like, but you can't resize it. The text display on the right will show the selection parameters and genotypic fitnesses, and there should be one green (heterozygous) individual in the middle of each island. Click the mouse pointer anywhere inside the graphical window to start the simulation. The text display will update itself and the island populations will evolve until each is fixed for one of the alleles (or 1000 generations have elapsed, which is very unlikely). To make the simulations run more slowly or quickly, you can adjust the per-generation sleep-time on line 29.

When a simulation ends, it writes to a summary file called "islands_3_results.txt" (example below). Here I did one run with no selection (neutrality, s = 0.00), then one with selection against A2 (s = -0.05) and one with selection in favor of A2 (s = 0.05). But I left the dominance parameter at its default value of h = 0.5 (additivity) in all cases. In the neutral run, A2 fixed on 2 of the 100 islands, then on one island in the run where it was deleterious, and on 3 islands when it was advantageous. To close the graphical window at the end of a run, click inside it again. Then if you want to start another run with the same parameters, put the focus on IDLE's program window and press F5.

```
p0 = 0.02  s =  0.00  h =  0.5  fixA1 =  98  fixA2 =   2  g = 192  gA2fix: 93 131
p0 = 0.02  s = -0.05  h =  0.5  fixA1 =  99  fixA2 =   1  g = 138  gA2fix: 138
p0 = 0.02  s =  0.05  h =  0.5  fixA1 =  97  fixA2 =   3  g = 116  gA2fix: 37 76 116
```

To understand the *distributions* of expected outcomes, and to see how *dominance* affects them, let's do a big class study in which each of us runs 5-10 of the neutral case, and of each of the six cases where s is either 0.05 or -0.05, and h is 0, 0.5 or 1.0. By combining our results files, we should have more than 200 runs of each kind. They will tell an interesting story about how drift can often overpower selection. And just for fun, please do a few runs with much stronger positive (and negative) selection coefficients.