EEB 200B: Genetic drift computer exercises

Dr.Michael Alfaro (based upon exercises developed by Kirk Lohmueller)
February 14, 2022

Exercise due date: February 28th

Simulating genetic drift forward in time

Recall that genetic drift is essentially binomial sampling from one generation to the next. Put another way, given the allele frequency in the present generation, the allele frequency in the next generation follows a binomial distribution.

R is equipped to simulate binomial random variables quite easily. To simulate from the binomial distribution, use the `rbinom` function.

Read about R's implementation of the binomial distribution using the following command:

```
> ?rbinom
starting httpd help server ... done
>
```

The `rbinom` function takes 3 arguments:

```
rbinom(n, size, prob)
```

`n` is the number of random variables that you wish to draw. For example, if you were simulating 10 independent SNPs at one time, you could set *n*=10. If you're only simulating 1 SNP, set *n*=1. Note, this can get confusing. Previously, when we spoke of the binomial distribution, *n* was the "size" (i.e. the maximum number of successes). In R, it is the number of random values drawn.

`Size` is the maximum value that any particular random variable can take on. When simulating drift, this value will be the number of chromosomes in the population (twice the number of individuals for a diploid population). In other arenas, this argument is sometimes called "*n*".

`Prob` is the probability of success. What is a success? Well, that's in the eye of the beholder. For coin flips, a success could be the coin coming up "heads". When simulating drift, a "success" could be the allele being transmitted. The probability of success will be the allele frequency (not count) in the previous generation. This is a probability so it must range from 0 to 1.

The `rbinom` command will return a vector of length *n* with random draws from the binomial distribution.

Example:

```
> rbinom(3,10,0.1)
[1] 3 0 1
```

This means that we've drawn 3 random variables from a binomial distribution. In the first draw, we saw 3 out of 10 successes. In the second draw, we saw 0 out of 10 successes. In the third draw, we saw 1 out of 10 successes.

1. What is the expected number of successes in a sample of size 10 from the binomial distribution with probability of success *p*=0.1? First, figure this out analytically based on the formulas from class. Second, write a simulation in R to confirm this.

Now, let's use the `rbinom` function in R to simulate genetic drift.

Let *p* be the allele frequency in the present generation, *N* is the population size. *N* can be the number of diploid individuals, or the number of chromosomes in the population. Either will work, as long as you are consistent. For the purposes of this lab exercise, let's define *N* to be the number of diploid individuals in the population. Thus, there would be 2*N* chromosomes in the population.

Then,

```
> p<-0.1
> N<-10
> count<-rbinom(1,2*N,p)
> count
[1] 3
> count/(2*N)
[1] 0.15
```

Thus, the allele changed in frequency from 0.1 to 0.15 in a single generation!

This is pretty boring. Let's do a real simulation with 1) more than 1 SNP at a time, and 2) more than 1 generation at a time!

2. Write a function in R that will simulate *T* generations of genetic drift for *L* independent SNPs. Keep track of the allele frequencies of each of the *L* SNPs in each of the *T* generations. All SNPs should start in the initial generation at frequency *p*.

Hint: Start by initializing a matrix to keep track of the frequencies each generation:

```
> T<-5 #number generations
> L<-3 #number independent SNPs
> freqs<-matrix(nrow=T,ncol=L) #now initialize a matrix of the
allele freqs each generation. Let each row be a generation, and
each column be a SNP.
>
> freqs
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
[3,]   NA   NA   NA
[4,]   NA   NA   NA
[5,]   NA   NA   NA
```

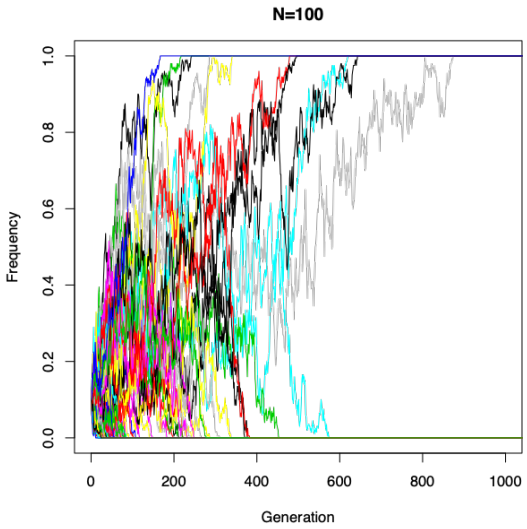
Another hint: Consider initializing the first row of this matrix with the initial allele frequency (*p*). Then, write a loop that will iterate through each generation and will perform the binomial sampling.

3. Use the function that you just wrote to simulate drift with the following parameters:

N=100 (*N* is the number of diploids, so there should be 2*N*=200 chromosomes)
L=1000
T=10000
p=0.1

Now, let's compute some things: **a) How many of the 1000 SNP are at frequency 0 at the end of the simulation (in generation 10000)? b) How many are at frequency 1? c) Does this value agree with the theoretical prediction for the probability of fixation of a neutral allele?**

d) Make a plot of the allele frequency trajectories for 100 of the SNPs. It should look something like this:



Repeat the simulation, but this time set *p*=0.6. e) How many of the 1000 SNP are at frequency 0 at the end of the simulation (in generation 10000)? f) How many are at frequency 1? g) Does this value agree with the theoretical prediction for the probability of fixation of a neutral allele?

4. Let's look at the effect of the population size on patterns of genetic drift. Repeat the simulation, but this time, set *N*=10, 500 and 1000. Keep the other parameters the same (*p*=0.1; *L*=1000; *T*=10000). Again, *N* is the number of diploids.

a) Make plots similar to the one shown above, but for all 4 populations sizes (*N*={10, 100, 500, 1000}). Plot all 4 plots on the same page so you can compare them. b) Based on examination of the plots, how does the population size affect allele frequency change? c) For each population size, in what proportion of simulation replicates did the derived allele become fixed by the end of the simulation? d) How is this probability affected by the population size? e) How does this probability of fixation estimated from the simulations match with the theoretical prediction?

5. Based on your simulations from the previous questions, in which population size (e.g. *N*={10, 100, 500, 1000}) would you expect an allele to go to fixation the fastest, given that it goes to fixation? Why?