EEB 200A: Genetic drift computer exercises

Dr. Kirk E. Lohmueller

November 17, 2015

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
```

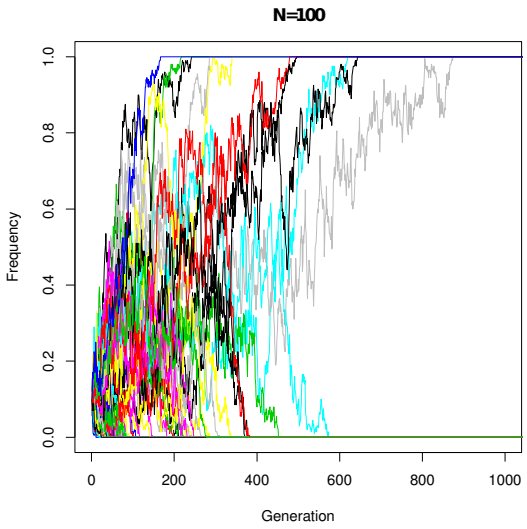
Another hint: Consider initializing the first row of this matrix with the initial allele frequency (*p*). Then, write a loop that will iterate through each generation and will perform the binomial sampling.

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?

Coalescent simulations

So far, you have modeled genetic drift forward in time. Now, let's simulate genetic drift going backwards in time, using the coalescent.

The coalescent sounds daunting, but it's actually quite simple to perform coalescent simulations for simple scenarios. You will perform some coalescent simulations in R for a sample size of 2 chromosomes from the population. You will simulate genealogies. Typically, genealogies are the trees that have 1) a topology, and 2) a series of times of the coalescent events. Note, for a sample size of *n*=2 chromosomes, the genealogies are trivial—they are just a pairwise coalescent time. There is only one topology for a sample size of 2 chromosomes. Thus, you only need to simulate the times. Simulating

larger sample sizes often (because there are some neat tricks that allow you to compute some things just based on the times) requires simulating the topology too.

Recall that the coalescent models the genealogical process (genetic drift) going backwards in time. The time intervals between coalescent events are exponentially distributed, when using continuous time. Conveniently, R can generate exponentially distributed random variables.

Type

```
> ?rexp
>
```

to learn more about them.

The `rexp` function takes 2 arguments:

```
rexp(n, rate = 1)
```

`n` is the number of exponential random variables that you would like to draw.

`rate` is the rate parameter for the exponential distribution. Note,1/`rate` is the expected value of the exponential distribution.

To convince yourself of this, simulate some exponential random variables. Let's draw 10 values from an exponential distribution with rate 2:

```
> rexp(10,2)
 [1] 0.141477287 1.121009207 0.167331187 0.664809961 0.598584073
0.351103565 0.622968316 1.074076987
 [9] 0.970647004 0.005346774
```

We can use the simulation to find the expected value of the exponential distribution with rate 2:

```
> mean(rexp(10000,2))
[1] 0.5112463
>
```

Theoretically, the expected value for an exponential distribution is 1/rate. Thus, in this example, the theoretical expectation is 0.5. The average value from the simulation is pretty close to this theoretical expected value.

Thus, in order to simulate coalescent times, we need the rate of coalescence for a sample size of 2 chromosomes in the population.

Note, as we discussed in lecture, there are different ways to scale the coalescent times. Typically, when using the continuous time coalescent (i.e. the coalescent times are drawn from an exponential distribution), we typically measure time in units of 2*N* generations (where 2*N* is the number of chromosomes in the population). Thus, 1 coalescent time unit would actually correspond to 2*N* generations. However, in order to help you understand the intuition behind the coalescent, here we will directly measure time in units of generations.

5) What is the rate of coalescence for a sample size of 2 chromosomes in a population of size 2*N* chromosomes? This is akin to the probability that 2 chromosomes find a common ancestor in the previous generation (if you prefer to think in discrete time). We derived this in lecture.

6) Perform 10,000 simulations of coalescent times for a sample size of 2 chromosomes from this population of size 2*N*=20,000. **a) What is the average time to the most recent common ancestor (TMRCA) in your simulations? b) What is the theoretical expectation? c) How do the two values compare? d) Make a density plot of your simulated coalescent times. e) What is the standard deviation of the coalescent times?**

7) Repeat the simulation described in question 6, but this time set 2*N*=2,000. Answer parts a-e from question 6 for this new set of simulations. f) How does the average TMRCA from the simulations in question 6 (where 2*N*=20,000) compare to the average TMRCA from the simulations in question 7 (2*N*=2,000)? g) What can you conclude about how the population size affects the expected coalescent time?

So far you have just simulated genealogies. That's nice, but we want to use the coalescent to relate the model (which right now only contains a parameter for the population size) to actual data. The data is some type of genetic variation in a sample of chromosomes.

Thus, we need to simulate mutations. Recall that neutral genetic variation can be modeled by "sprinkling" mutations onto the simulated genealogies.

Let μ be the per-generation per-chromosome mutation rate. Note, μ here is summed up over all bases under consideration. For example, assume a mutation rate of 1×10^{-8} per base pair per generation. But, if we sequenced 10,000 bases, then μ would be 1×10^{-4}.

Under the infinite sites mutational model, mutations arise via a Poisson process. Said another way, conditional on the mutation rate and the genealogy, we can draw the number of SNPs in a dataset from a Poisson distribution.

A few comments on the Poisson distribution: The Poisson distribution is a neat distribution to model the number of occurrences of an event in a given time interval (say a generation). Many biological processes converge to a Poisson distribution when 1) the

probability of an event happening at a particular time point is low, and 2) there are many opportunities for the event to occur. These conditions are often met when thinking about mutations!

As you might have guessed, R can also simulate Poisson random variables. This is done using the “rpois” function.

Type

```
> ?rpois
>
```

to learn more.

The function takes 2 arguments:

```
rpois(n, lambda)
```

`n` is the number of Poisson random variables that you wish to draw.

`Lambda` is the rate parameter for the Poisson distribution. The rate is equal to the expected value (and variance) of the Poisson distribution.

Let’s simulate some Poisson random variables. We’ll draw 10 values from a Poisson distribution with rate 2.

```
> rpois(10,2)
 [1] 2 5 2 0 1 0 2 5 1 2
```

Notice that the Poisson distribution returns discrete values (i.e. integers). Thus, this gives the number of occurrences of an event in a given time interval with a given rate of occurrence (here 2).

We can use the simulations to confirm that the expected value of the Poisson distribution is equal to the rate:

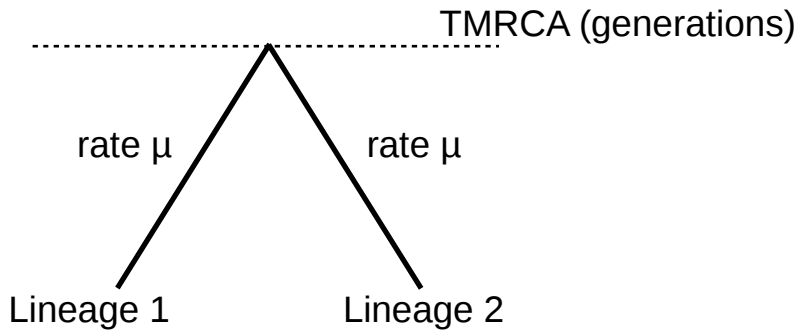
```
> mean(rpois(10000,2))
[1] 1.9964
>
```

What is the rate parameter for the Poisson distribution when trying to simulate mutations on genealogies? [This is a rhetorical question. I will answer it below…] Recall that for a given chromosome, mutations occur at rate μ per generation. Note, that for any particular time point, there are actually 2 chromosomes that we care about in our sample (see diagram on the next page). In other words, a mutation on either of these

chromosomes would generate a SNP in our dataset. Thus, per unit time, the mutation rate is 2μ.

But, be careful, 2μ is the rate of mutation for a particular generation. We also need to consider the length of the genealogies (i.e. the number of generations over which mutations can be accumulated). Luckily, we've already simulated this in questions 6 and 7!

The mutation rate for a given genealogy is then the length of that genealogy in units of generations, multiplied by 2μ. Note, this only applies for a sample size of 2. For larger sample sizes, it's a bit more complicated…



Total mutation rate/generation= $\mu_{Lineage\ 1}+\mu_{Lineage\ 2}=2\mu$

8) Let's add mutations to the genealogies you've simulated and then look at what these models predict in terms of the genetic variation you might see in a sample. Set μ =1 x 10^{-4}**. a) Add mutations to the genealogies (really the coalescent times) that you simulated in question 6 (N=20,000) using the algorithm I just outlined. b) What is the average number of SNPs per genealogy?**

c) Recall that θ=4*N*μ. Note, the factor-of-2-confusion—*N* refers to the number of diploids. So, θ is twice the number of chromosomes multiplied by μ. **What is θ predicted to be in this example? d) How does the value of θ compare to the average number of SNPs seen in the simulations? e) Pretty neat, huh?** (the correct answer is "yes, pretty neat!") **f) Make a density plot of the number of SNPs per simulation replicate.**

10) Repeat all the parts in question 8, but this time, set *N*=2,000.

11) Make a scatter plot of the number of SNPs seen in each simulation replicate vs. the TMRCA for that simulation replicate. Hint: plot the number of SNPs on the y-axis and the TMRCA on the x-axis. How are these variables related to each other? Does the number of SNPs tell you anything about the TMRCA?

12) Imagine that you sequenced a 10kb region of DNA from single individual (a diploid species; i.e. you have a sample of 2 chromosomes). You observed 10 SNPs in that 10kb interval. Note that independent evidence suggests a mutation rate of 1×10^{-8} per base pair/generation. (Hint: this is very similar to what you've simulated). **a) Use coalescent simulations to evaluate whether a model with *N*=10,000 is compatible with your observed data? Put another way, what proportion of simulation replicates from this model have ≥10 SNPs? b) Is the model with *N*=1,000 compatible with your observed data? What is the proportion of simulation replicates that have ≥10 SNPs?**

Congratulations! You've now explored a very simple population genetic model using coalescent simulations. I hope that you can begin to appreciate the power of the coalescent approach for being able to relate population genetic models to data in a straightforward way.