R_Assignment_Simulations

Ian Dworkin

11/27/2018

Due Monday December 3rd, 5:59PM

1 - Please do the DataCamp course (all chapters) on "Introduction to Bioconductor".

2- In class we developed the code to perform the deterministic simulation for a simple population genetics model with one locus with 2 alleles in a haploid. Use what you learned in class, and the following information to write a general simulator for diploids. Your code should be flexible enough (i.e. arguments in a function call for instance) to try different sets of parameters including initial allele frequencies, fitness of each genotype, number of generations. Your script should do the following at a minimum:

- Report whether the beneficial allele fixes by the end of your simulation.
- Plot the trajectory of the beneficial allele across generations.

Some things you will need:

mean population fitness in a given generation (t) is

$\overline{W} = p^2 W_{AA} + 2pqW_{Aa} + q^2 W_{aa}$

and

$p_{(t+1)} = p_{(t)}^2 \frac{W_{AA}}{\overline{W}} + p_{(t)} q_{(t)} \frac{WAa}{\overline{W}}$

Where

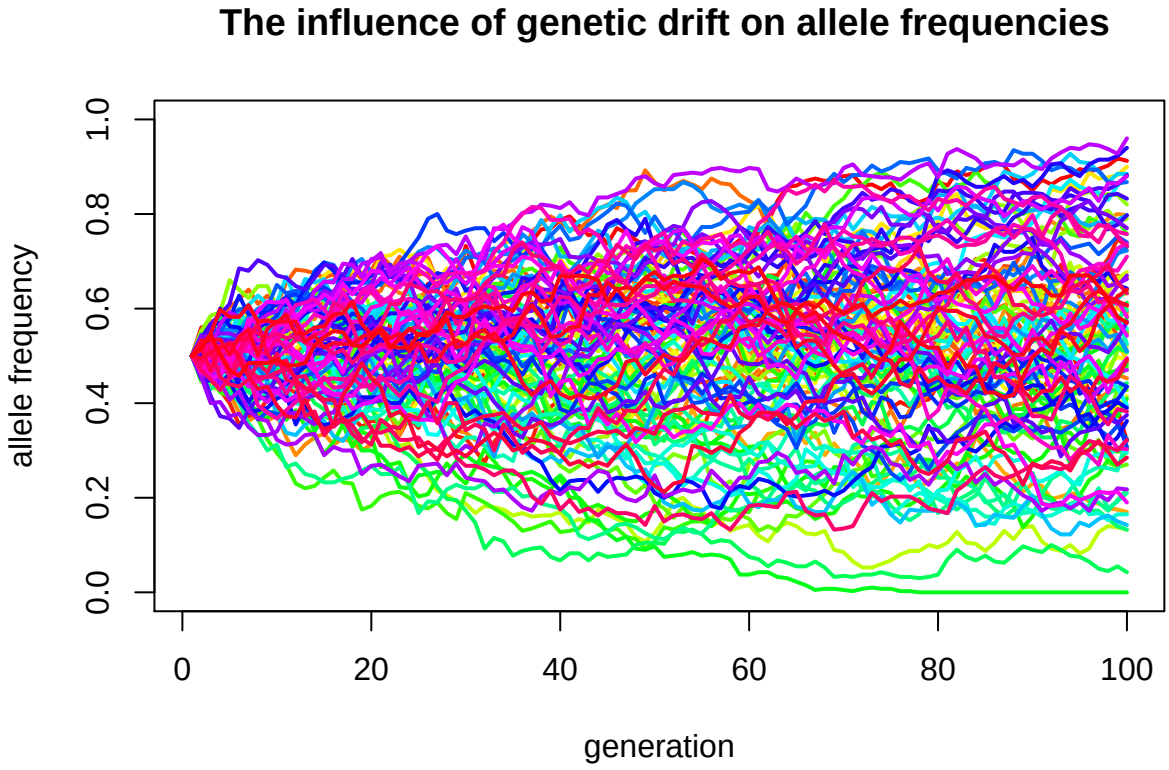
- $p_{(t+1)}$ is the frequency of the A allele in the population in generation $t+1$.
- $p_{(t)}$ and $q_{(t)}$ are the allele frequencies of A and a in generation t.
- $p_{(t)}^2$ is the genotypic frequency for the AA diploid genotype in generation t.
- W_{AA}, W_{Aa} & W_{aa} are the fitnesses for each genotype AA, Aa and aa respectively.
- You can also use (if you want, but do not need to) the fact that $p + q = 1$ for allele frequencies, and $p^2 + 2pq + q^2 = 1$ for genotypic frequencies.
- While mean fitness of the population changes with allele frequencies (according to the equation above), individual genotypic fitnesses do not (i.e. no frequency dependent selection)

3 - Genetic drift simulator. In class we saw how with a simple stochastic simulation, and the use of the `sample()` function we could simulate how allele frequencies changed in small populations due to genetic drift alone (i.e. random sampling of gametes). However we only did this over the course of two generations. Please write a general purpose simulator that allows you to examine the effects of genetic drift allowing for variable arguments/parameters minimally including:

- Number of alleles in population (i.e. for diploids 2 times the number of individuals).
- Starting allele frequency
- Number of generations.
- plot allele frequency changes over time

4 - Repeating your stochastic simulation. Now that you have a working simulator of genetic drift, you want to use it to assess how likely it is for the allele to be lost ($p = 0$) after a certain number of generations of drift (we will use 100 generations, but your function should be flexible). Using your function (you can modify it if you need to), perform a simulation 1000 times each with starting allele frequencies of $p = f(A)$ of 0.5, 0.25 and 0.1, with 200 diploid individuals in the population each generation. Have your function record the proportion (out of 1000) of simulated runs that the A allele is lost from the population ($p = 0$).

5 - Write some code that allows you to plot the allele trajectories for drift for 100 of the simulations starting at $p = 0.5$. hint: I showed you an example of how to draw multiple lines with a single function last week... It could look something like this.



6 - A simple power analysis. One common use for stochastic simulations is to evaluate the statistical power of a particular method or model given some known parameters such as effect size (like the slope of the regression line), sample size, and residual standard error (the uncertainty in your estimator). While I kind of detest teaching anything with *p-values* (it is not a great way of evaluating statistical models) we will use them for this exercise.

We saw in class how to simulate data for the relationship between y and x and plot the regression fits (best fit lines) for the simulated data. You are going to use that same approach to determine how often you observe a "significant" p-value ($p < 0.05$) under some different scenarios.

In class we did something approximately like this (with a few changes)

```
x <- seq(from =1, to = 10, length.out = 20) # length.out is how many observations we will have
a <- 0.5 # intercept
b <- 0.1 # slope
y_deterministic <- a + b*x

y_simulated <- rnorm(length(x), mean = y_deterministic, sd = 2)

mod_sim <- lm(y_simulated ~ x)
p_val_slope <- summary(mod_sim)$coef[2,4] # extracts the p-value
p_val_slope
```

```
## [1] 0.09681437
```

- First re-write this as a function where the intercept, slope, sample size (number of observations) and residual standard error (the stochastic variation) are all arguments in the function, so that they can be changed easily to different values. Check that it works. How can you confirm (given this is stochastic)

that you can get the same results whether it is in the code I wrote above or the function you wrote?

- Now with your new function, run it 1000 times and generate a histogram of the p-values from these simulations? Check what proportion of times the p-value was less than 0.05.
- Redo this, but change the slope to 0. Examine this histogram and determine the proportion of times that the p-value is less than 0.05. Explain this result.
- Finally, using either a for loop or an *Rish* method (i.e. one of the apply family of functions) try a *grid* of sample sizes from 10 to 100 by 5 (i.e. 10, 15, 20. . . ., 95, 100) with the slope being fixed at 0.1, intercept = 0.5 and residual standard error as 1.5. At each different sample size, run the simulation 100 times and report the frequency at which the p value is less than 0.05. What pattern do you observe. Plotting the proportion of p-values less than 0.05 on the Y axis VS sample size (X axis) may be very helpful