Bio720 - Simulating Data

Ian Dworkin

11/19/2018

Contents

Simulation as a key skill in your toolbox

One of the most important skills in your bag as new computational biologists is the ability to perform simulations. In particular, to simulate data or evaluate models numerically (or both).

Mathematical models abound

In biology mathematical models are the basis for much of the theoretical and conceptual background in disciplines like ecology, evolution, population genetics, molecular evolution & biochemistry.

When you can't solve a model. . .

Many of the models that are developed are not *analytically* tractable. That is, without making very strong biological assumptions there are no *closed form solutions*.

That is the models can not be solved for the general case.

Using computers to find numerical solutions

- For such models, "solutions" can still be found.
- For some models, stability analysis can be performed (among other techniques).
- However, most often, scientists resort to computers to identify *numerical solutions*

Deterministic VS. Stochastic

- Within *dynamical* models, that include the majority of models in biology there are two broad categories.
- **Deterministic** - where the outcome of the model entirely depends on the model itself and the starting conditions.
- **Stochastic** - where random events influence the outcomes of the model, and only the probability of events can be predicted, not exact outcomes.

A simple deterministic model one-locus model of natural selection.

- In population genetics we often start with a simple deterministic model of how selection on an allele influences its frequency in the population over time.
- In a simple haploid model we can envision two alternative alleles, A and a.
- a is initially fixed in the population, but the new mutation A arrives and it is beneficial. What will happen to this allele?

Haploid selection model

- Frequency of A at time t is $p(t)$
- Fitness of A is W_A, and for a is W_a

$p(t+1) = \frac{p(t)W_A}{p(t)W_A + W_a(1-p(t))}$

$p(t+1) = \frac{p(t)W_A}{\overline{W}}$

- Where $\overline{W}$ is mean fitness for the population
- How would you convert this into R code for one generation to the next?
- Start with what variables you need.

Converting this into R code - What variables

- We need variables for the fitness values for each allele, W_A, W_a and for allele frequency of A $p(t+1)$ at time t and t+1.
- With this we can write the equation for allele frequency change from one generation to the next.

Converting this into R code - What variables

```
p_t1 <- function(w_A, w_a, p_t0) {
        w_bar <- (p_t0*w_A) + ((1-p_t0)*w_a) # mean pop fitness
        p_t1 <- (w_A*p_t0)/w_bar
        return(p_t1)}

p_t1(w_A = 1.1, w_a = 1.0, p_t0 = 0.5)
```

```
## [1] 0.5238095
```

Is this deterministic or stochastic, what would be a quick way to check?

Is this deterministic or stochastic, what would be a quick way to check?

```
replicate(n = 100, p_t1(1.1, 1.0, 0.5))
```

```
##   [1] 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095
##   [8] 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095
##  [15] 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095
##  [22] 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095
##  [29] 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095
##  [36] 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095
##  [43] 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095
##  [50] 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095
##  [57] 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095
##  [64] 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095
##  [71] 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095
##  [78] 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095
##  [85] 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095
##  [92] 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095 0.5238095
##  [99] 0.5238095 0.5238095
```

Allele frequency dynamics.

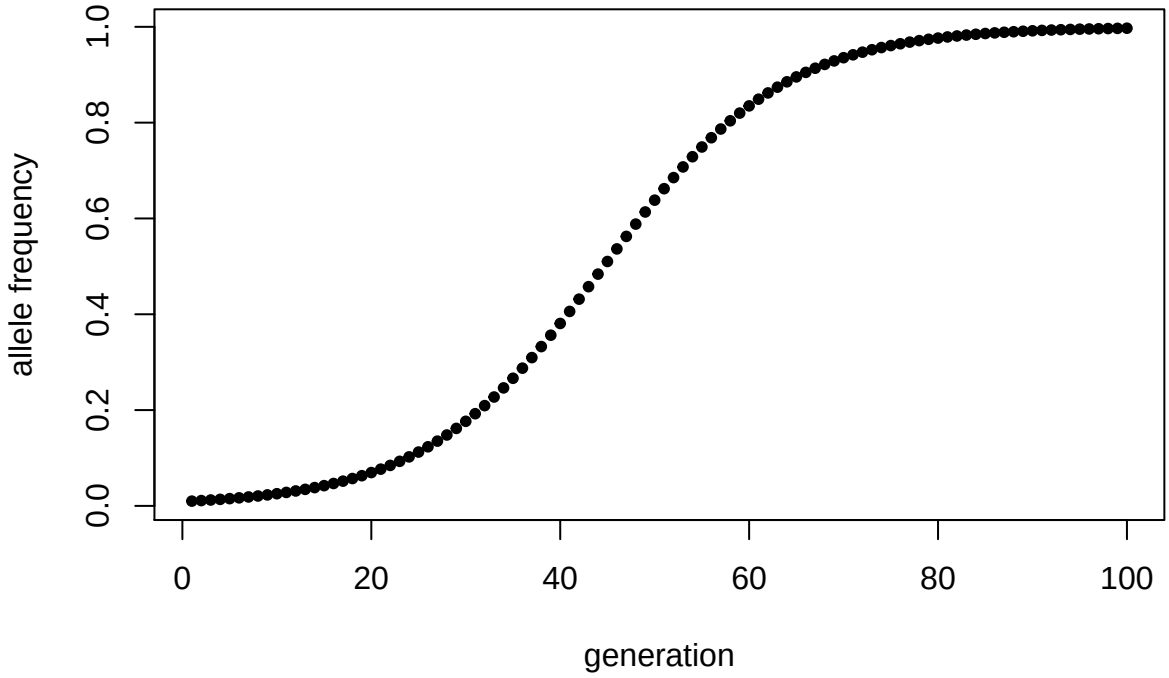
- Now let's extend this across many generations. We need to rewrite the function as we expect the allele frequencies to change each generation. Also, mean population fitness will change each generation (as allele frequency changes)
- write this simulation (a *for loop* would be sensible) and go for 200 generations, and look at the dynamics (starting allele frequency is p = 0.01). Wrap it all as a function called `haploid_selection`

```
haploid_selection <- function(p0 = 0.01, w1 = 1, w2 = 0.9, n = 100) {

    # Initialize vectors to store allele frequencies and mean pop fitness
    p <- rep(NA,n)  # a vector to store allele frequencies

    w_bar <- rep(NA, n)

    # starting conditions
    p[1] <- p0 # starting allele frequencies

    w_bar[1] <- (p[1]*w1) + ((1-p[1])*w2)

    # now we need to loop from generation to generation
    for ( i in 2:n) {
        w_bar[i - 1] <- (p[i - 1]*w1) + ((1-p[i - 1])*w2) # mean population fitness
        p[i] <- (w1*p[i - 1])/w_bar[i - 1]
    }
```

```
    return(p)
}
```

Test the model and plot it.

```
p <- haploid_selection()
generations <- 1:length(p)
plot(p ~ generations, pch = 20,
     ylab = "allele frequency",
     xlab = "generation")
```



Using simple numerical simulation to gain intuition for the system.

- Try altering fitness advantage of A a little bit. Or reducing initial allele frequency
- Is this deterministic or stochastic?

Making a more general function

```
haploid.selection <- function(p0 = 0.01, w1 = 1, w2 = 0.9, n = 100) {

# Initialize vectors to store p, delta p and mean pop fitness

p <- rep(NA,n)

delta_p <- rep(NA, n)

w_bar <- rep(NA, n)
```

```
# starting conditions
p[1] <- p0 # starting allele frequencies

delta_p[1] <- 0 #change in allele frequency

w_bar[1] <- (p[1]*w1) + ((1-p[1])*w2)

# now we need to loop from generation to generation
for ( i in 2:n) {
    w_bar[i - 1] <- (p[i - 1]*w1) + ((1-p[i - 1])*w2)
        p[i] <- (w1*p[i - 1])/w_bar[i - 1]
        delta_p[i] <- p[i] - p[i-1]
    }

    if (any(p > 0.9999)) {
    fixation <- min(which.max(p > 0.9999))
    cat("fixation for A1 occurs approximately at generation:", fixation )
    } else {
        maxAlleleFreq <- max(p)
        cat("fixation of A1 does not occur, max. allele frequency is:", print(maxAlleleFreq, digits = 2)
    }

# Let's make some plots
par(mfrow=c(2,2))

# 1. mean population fitness over time
plot(x = 1:n, y = w_bar,
     xlab = "generations",
     ylab = expression(bar(w)),
     pch=20, col="red", cex = 2, cex.lab = 1.5, cex.main = 1.5,
     main = paste("p0 = ", p0, "and s = ", (1 - (w2/w1))))

    # 2. change in allele frequency over time
plot(x = 1:n, y = p,
     xlab="generations",
     ylab="Allele frequency (p)",
     pch = 20, col = "red", cex.lab = 1.5)

    # 3. plot of p[t+1] vs p[t]
p.1 <- p[-n]
p.2 <- p[-1]

plot(p.2 ~ p.1,
     xlab = expression(p[t]),
     ylab = expression(p[t+1]),
     pch = 20, col = "red", cex = 2, cex.lab = 1.5)

# 4. plot of allele frequency change
plot(x = 2:n, y = delta_p[-1],
        xlab = "generation",
        ylab= expression(paste(Delta,"p")),
        pch = 20, col = "red", cex = 2, cex.lab = 1.5)
}
```

Let's see what this does.

```
haploid.selection(p0 = 0.0001, w1 = 1, w2 = 0.987, n = 1000)
```

```
## [1] 0.98
## fixation of A1 does not occur, max. allele frequency is: 0.9794053
```