

Chapter 2

Viability Selection

Let's start you on your journey towards an understanding of the ideas of basic calculus by just looking at how far we can go with applying algebra alone. We are going to look carefully at a model of natural selection called **viability selection**. We are going to build some nice models using lots of letters to represent things of interest and all we need to understand this is your basic algebra skills from high school. Now in our experience, most of you probably dislike mathematical things because they seem so divorced from something real. So our tact here is to build something useful and along the way, we have to do various manipulations which will help you gently remember all your basic skills. Now these discussions come from the first chapter of a great book on evolutionary biology (McElreath and Boyd 2007) which we encourage you to pick up and study at some point. We are not going to cover these ideas in the great detail there; instead we will focus on a few portions. We also use a bit different language as our target is people who are just seeing calculus perhaps although we are also hoping to get you to read this as a return to old things. At any rate, remember, the purpose of modeling is insight into complicated stuff and mathematics is a great tool to help us achieve that. Now let's get to it.

2.1 A Basic Evolutionary Model

We are interested in understanding the long term effects of genes in a population. Obviously, it is very hard to even frame questions about this. One of the benefits of our use of mathematics is that it allows us to build a very simplified model which nevertheless helps us understand general principles. These are biological versions of the famous Einstein *gedanken* experiments: i.e. thought experiments which help develop intuition and clarity.

We will explicitly use *letters* to represent quantities in the biology we want to keep track of. We will then want to make some assumptions about how these quantities depend and interact with one another. Of course, in doing so, we always make error

which is called **model error** as there is no way we can specify everything in a real biological scenario. We must make assumptions.

We start by assuming we have a population of N individuals at a given time. It also seems reasonable to think of our time unit as **generations**. So we would say the population at generation t is given by $N(t)$. Hence, the number of individuals in our population is not necessarily fixed but can change from one generation to the next. To understand this change, we need to know how our population reproduces so as to create the next generation. We want a very simple model here, so we will also assume each individual in our population is **haploid** which means new individuals are produced without sex or any sort of genetic recombination. Wait a minute, you say! What could this mean? Usually, an adult has a certain number of chromosomes; call this number $2P$. The gametes are the cells with half of the genetic material and therefore have only P chromosomes. Then in sexual reproduction there is a complicated process by which the sperm and the egg interact to create a new cell called a *zygote* which has $2P$ chromosomes. The gametes are considered **haploid** as they each have half of the chromosomes of the adult. Sexual reproduction allows a mixing of the chromosomes from two adults to form a new individual having $2P$ chromosomes. The cell formed by the union of the sperm and egg is called a **zygote** and it is diploid as it has $2P$ chromosomes. So we are making a very big simplifying assumption. Essentially we are saying each adult has Q chromosomes and the reproduction process does not mix genetic information from another adult and hence the zygote formed by what is evidently some form of asexual reproduction also has Q chromosomes. Note, calling the cell formed by this reproductive process a zygote is a bit odd as usually that word is reserved for the cell formed by a sexual reproduction. So we have a really simplistic population dynamic here! Hence, we use the term *zygote* here loosely!

With this said, we also assume in each generation individuals go through their life cycle exactly the same: all individuals are born at the same time and all individuals reproduce at the same time. We will call this a **discrete** dynamic. Note a zygote does not have to live long enough to survive to an adult.

Since we want to develop a very simple model we assume there are only two genotypes, type **A** and type **B**. We also assume **A** is more likely to survive to an adult. We need to start defining quantities of interest: i.e. variables now.

- N_A is the number of individuals of phenotype **A** and N_B is the number of individuals of phenotype **B** in a given generation.
- N is the number of individuals in the population and this number changes each generation as

$$N = N_A + N_B.$$

This is our first equation and note it is pretty simple. It simply counts things. Now if we really wanted to be careful, we might let t represent the generation we are in: i.e we start at generation 0 so $t = 0$ initially. Then the first generation is $t = 1$ and so forth. Eventually, we will want to know what happens in the population as

t increases, but that is for later. So if we wanted to be really explicit, we would add the variable t to our equation above to get

$$N(t) = N_A(t) + N_B(t).$$

It is a bookkeeping device really and not terribly essential. However, without adding (t) in there variables, we have to remember that each one does depend on the generation we are in. Also, note generations here are integers not numbers like $1/2$ or 2.4 . Nothing but integers for now!

- It is also convenient to keep track of the fraction of individuals in the population that are genotype **A** or **B**. This fraction is also called the **frequency** of type **A** and **B** respectively. We use new variables for this.

$$P_A = \frac{N_A}{N_A + N_B}$$

$$P_B = \frac{N_B}{N_A + N_B}$$

We can also explicitly add the generation variable t to get

$$P_A(t) = \frac{N_A(t)}{N_A(t) + N_B(t)}$$

$$P_B(t) = \frac{N_B(t)}{N_A(t) + N_B(t)}$$

- Finally, as we said, individuals of each phenotype do not necessarily survive to adulthood. We will assume each phenotype survives to adulthood with a certain probability, V_A and V_B , respectively.

2.1.1 Examples

Let's do some calculations.

Example 2.1.1 If $N_A(0) = 25$ and $N_B(0) = 15$, find $P_A(0)$ and $P_B(0)$.

Solution *This is an easy computation.*

$$P_A(0) = \frac{N_A(0)}{N_A(0) + N_B(0)}$$

$$= \frac{25}{25 + 15} = \frac{25}{40}$$

$$\begin{aligned}
 P_B(t) &= \frac{N_B(0)}{N_A(0) + N_B(0)} \\
 &= \frac{15}{25 + 15} = \frac{15}{40}.
 \end{aligned}$$

2.1.2 Homework

Exercise 2.1.1 If $N_A(0) = 35$ and $N_B(0) = 60$, find $P_A(0)$ and $P_B(0)$.

Exercise 2.1.2 If $N_A(0) = 1500$ and $N_B(0) = 900$, find $P_A(0)$ and $P_B(0)$.

Exercise 2.1.3 If $N_A(0) = 82$ and $N_B(0) = 47$, find $P_A(0)$ and $P_B(0)$.

2.2 The Next Generation

We now have enough of a setup to look more carefully at how our population moves from one generation to the next. Let's do this very carefully. We will keep track of how the frequency P_A changes. Let $P_A(t)$ be the frequency for **A** at generation t . What is the frequency at the next generation $t + 1$? Here is how we figure it out.

- The number of zygotes from individuals of genotype **A** at generation t is assumed to be $z N_A(t)$ where z is the number of zygotes each individual of type **A** produces. Note that z plays the role of the **fertility** of individuals of type **A**. We will assume that individuals of type **B** also create z zygotes. Hence, their fertility is also z . We could also have assumed these fertilities are different and labeled them as z_A and z_B , respectively, but we aren't doing that here. So the number of zygotes of **B** type individuals is $z N_B(t)$.
- The frequency of **A** zygotes at generation t is then

$$P_{AZ}(t) = \frac{z N_A(t)}{z N_A(t) + z N_B(t)}$$

where we add an additional subscript to indicate we are looking at zygote frequencies. Note the z 's cancel to show us that the frequency of **A** zygotes in the population does not depend on the value of z at all. We have

$$P_{AZ}(t) = \frac{N_A(t)}{N_A(t) + N_B(t)} = P_A(t).$$

- But not all zygotes survive to adulthood. If we multiply numbers of zygotes by their probability of survival, V_A or V_B , the number of **A** zygotes that survive to adulthood is $V_A N_A(t)$ and the number of **B** zygotes that survive to adulthood is

$V_B N_B(t)$. We see the frequency of **A** zygotes that survive to adulthood to give the generation $t + 1$ must be

$$P_{AZS}(t + 1) = \frac{V_A N_A(t)}{V_A N_A(t) + V_B N_B(t)}.$$

where we have added yet another subscript *S* to indicate survival. Note we add the generation label $t + 1$ to P_{AZS} because this number is the frequency of adults that start generation $t + 1$. Also, note this fraction is exactly how we define our usual frequency of **A** at generation $t + 1$. Hence, we can say

$$P_A(t + 1) = \frac{V_A N_A(t)}{V_A N_A(t) + V_B N_B(t)}.$$

Now for the final step. From the way we define stuff, notice that

$$P_A(t) = \frac{N_A(t)}{N_A(t) + N_B(t)}$$

Now replace the denominator by $N(t)$ and multiply both sides by this denominator to get

$$N(t) P_A(t) = N_A(t).$$

Then consider the frequency for **B**. Note

$$\begin{aligned} 1 - P_A(t) &= 1 - \frac{N_A(t)}{N_A(t) + N_B(t)} \\ &= 1 - \frac{N_A(t)}{N(t)}. \end{aligned}$$

Getting a common denominator, we find

$$1 - P_A(t) = \frac{N(t) - N_A(t)}{N(t)}.$$

But we can simplify $N(t) - N_A(t)$ to simply $N_B(t)$. Using this in the last equation, we have found that

$$1 - P_A(t) = \frac{N_B(t)}{N(t)}$$

which leads to the identity we wanted: $N_B(t) = (1 - P_A(t)) N(t)$. This analysis works just fine at generation $t + 1$ too, but at that generation, we have

$$\begin{aligned}
 N_A(t+1) &= N(t) V_A P_A(t) \\
 N_B(t+1) &= N(t) V_B P_B(t) \\
 &= N(t) V_B (1 - P_A(t))
 \end{aligned}$$

So we can say

$$\begin{aligned}
 P_A(t+1) &= \frac{N_A(t+1)}{N_A(t+1) + N_B(t+1)} \\
 &= \frac{P_A(t) N(t) V_A}{P_A(t) N(t) V_A + (1 - P_A(t)) N(t) V_B}.
 \end{aligned}$$

Since $N(t)$ is common in both the numerator and denominator, we can cancel them to get

$$P_A(t+1) = \frac{V_A P_A(t)}{P_A(t) V_A + (1 - P_A(t)) V_B}.$$

This tells us how the frequency of the **A** genotype changes each generation.

2.2.1 Examples

Example 2.2.1 Let $V_A = 0.7$ and $V_B = 0.45$ and assume $P_A(0) = 0.03$. Find $P_A(1)$.

Solution We know

$$\begin{aligned}
 P_A(1) &= \frac{V_A P_A(0)}{P_A(0) V_A + (1 - P_A(0)) V_B} \\
 &= \frac{0.7 (0.03)}{0.7 (0.03) + 0.45 (0.97)} = \frac{0.0210}{0.0210 + 0.4365} = 0.0459.
 \end{aligned}$$

2.2.2 Homework

Exercise 2.2.1 Let $V_A = 0.8$ and $V_B = 0.25$ and assume $P_A(0) = 0.02$. Find $P_A(1)$.

Exercise 2.2.2 Let $V_A = 0.6$ and $V_B = 0.1$ and assume $P_A(0) = 0.01$. Find $P_A(1)$.

Exercise 2.2.3 Let $V_A = 0.85$ and $V_B = 0.3$ and assume $P_A(0) = 0.05$. Find $P_A(1)$.

2.3 A Difference Equation

We can also derive a formula for the change in frequency at each generation by doing a subtraction. We consider

$$P_A(t+1) - P_A(t) = \frac{V_A P_A(t)}{P_A(t) V_A + (1 - P_A(t)) V_B} - P_A(t).$$

Now the algebra gets a bit messy, but bear with us. Get a common denominator next.

$$P_A(t+1) - P_A(t) = \frac{V_A P_A(t)}{P_A(t) V_A + (1 - P_A(t)) V_B} - P_A(t) \frac{P_A(t) V_A + (1 - P_A(t)) V_B}{P_A(t) V_A + (1 - P_A(t)) V_B}.$$

Multiply everything out and put into one big fraction.

$$\begin{aligned} P_A(t+1) - P_A(t) &= \frac{V_A P_A(t) - P_A(t) \left(P_A(t) V_A + (1 - P_A(t)) V_B \right)}{P_A(t) V_A + (1 - P_A(t)) V_B} \\ &= \frac{V_A P_A(t) - (P_A(t))^2 V_A - \left(P_A(t) (1 - P_A(t)) V_B \right)}{P_A(t) V_A + (1 - P_A(t)) V_B}. \end{aligned}$$

Whew! Messy indeed. Now factor a bit to get

$$P_A(t+1) - P_A(t) = \frac{\left(P_A(t) - P_A^2(t) \right) V_A - \left(P_A(t) (1 - P_A(t)) V_B \right)}{P_A(t) V_A + (1 - P_A(t)) V_B}.$$

Almost done. As a final step note that $P_A(t) - P_A^2(t)$ is the same as $P_A(t)(1 - P_A(t))$. So in the numerator of this complicated fraction, we can factor that term out to give

$$P_A(t+1) - P_A(t) = \frac{\left(P_A(t) (1 - P_A(t)) \right) \left(V_A - V_B \right)}{P_A(t) V_A + (1 - P_A(t)) V_B}.$$

This is what is called a **recursion** equation. It is called that because it gives us a formula which tells us how to find the next generation results give the previous generation's frequency. For example, if the frequency of **A** in the population started at $P(0) = p_0$, the change in frequency of **A** in the next generation is given by

$$P_A(1) - P_A(0) = \frac{\left(P_A(0) (1 - P_A(0)) \right) \left(V_A - V_B \right)}{P_A(0) V_A + (1 - P_A(0)) V_B}.$$

Substituting in the value p_0 we find we can solve for $P_A(1)$ as follows:

$$P_A(1) = p_0 + \frac{p_0 (1 - p_0) (V_A - V_B)}{p_0 V_A + (1 - p_0) V_B}$$

and continuing in this vein, the frequency at generation 2 would be

$$P_A(2) = p_1 + \frac{p_1 (1 - p_1) (V_A - V_B)}{p_1 V_A + (1 - p_1) V_B}$$

where we denote $P_A(1)$ more simply by p_1 . We can then continue and calculate as many of these as we want. Note, this will give us a table of numbers: for each generation t , we calculate a frequency of **A** in the population $P_A(t)$. We can then plot these ordered pairs on a standard set of axis: the horizontal **generation** axis and the vertical **frequency of A** axis. Of course, these means a lot of computation! So as time goes on, we will use a tool called **MatLab** to do this more easily. We also notice that an interesting question is what happens to this frequency as the number of generations increase? Does the frequency plateau at some level or does it rise forever? A little thought should show you that the frequency can't go above 1 as frequency is a fraction between 0 and 1 at each generation. So the question is does the frequency plateau at 1 or something less? With this question, we are really asking about what is called the **limiting** behavior of our frequency model. Our studies in Calculus will give us a variety of tools and ideas to let us handle this question, but already you should see that it is an important thing to think about.

Another important thing to notice is that the idea of **generation** is a fluid concept as **generation** means very different things in different species. So when we use t to represent a generation, the time interval to get to generation $t + 1$ can be years, months, days, hours and even less. Our severely odd thought experiment creature of this model can be replaced by other creatures with sexual reproduction and all sorts of other more accurate assumptions. But the basic questions will still be the same. We use our model to find how the frequencies change in each generation and we use calculation to generate plots so we can see the behavior visually. And we ask the big question: what is the limiting behavior?

2.3.1 Examples

We should do some examples of how to use these ideas. First, some computations.

Example 2.3.1 Find the frequency of type **A** individuals at generation 1 given that initially the frequency of **A** individuals is $p_0 = 0.01$ and the probabilities of **A** and **B** are $V_A = 0.8$ and $V_B = 0.3$.

Solution We use the equation above. We have

$$\begin{aligned}
 P_A(1) &= p_0 + \frac{p_0 (1 - p_0) (V_A - V_B)}{p_0 V_A + (1 - p_0) V_B} \\
 &= 0.01 + \frac{0.01 (0.99) (0.8 - 0.3)}{0.01 (0.8) + 0.99 (0.3)} \\
 &= 0.01 + \frac{0.00495}{0.305} = 0.01 + 0.01623 = 0.0262.
 \end{aligned}$$

So the change in frequency of **A** is $P_A(1) - P_A(0) = 0.0262 - 0.01 = 0.0162$ with the new frequency given by $P_A(1) = 0.0262$. We can clearly calculate as many of these as we wish.

2.3.2 Homework

Exercise 2.3.1 Find the frequency of type **A** individuals at generation 1 given that initially the frequency of **A** individuals is $p_0 = 0.02$ and the probabilities of **A** and **b** are $V_A = 0.85$ and $V_B = 0.25$.

Exercise 2.3.2 Find the frequency of type **A** individuals at generation 1 given that initially the frequency of **A** individuals is $p_0 = 0.03$ and the probabilities of **A** and **b** are $V_A = 0.75$ and $V_B = 0.45$.

Exercise 2.3.3 Find the frequency of type **A** individuals at generation 1 given that initially the frequency of **A** individuals is $p_0 = 0.04$ and the probabilities of **A** and **b** are $V_A = 0.65$ and $V_B = 0.2$.

Exercise 2.3.4 Find the frequency of type **A** individuals at generation 1 and generation 2 given that initially the frequency of **A** individuals is $p_0 = 0.02$ and the probabilities of **A** and **b** are $V_A = 0.75$ and $V_B = 0.3$.

Exercise 2.3.5 Find the frequency of type **A** individuals at generation 1 and generation 2 given that initially the frequency of **A** individuals is $p_0 = 0.01$ and the probabilities of **A** and **b** are $V_A = 0.6$ and $V_B = 0.2$.

2.4 The Functional Form of the Frequency

Let's derive a functional form for $P_A(t)$. We will start with the number of **A** in the population initially, $N_A(0)$. From our discussions, we know that at the next generation, the number of **A** is

$$\begin{aligned}
 N_A(1) &= \left(\text{Fertility of } \mathbf{A} \right) \times \left(\text{The number of } \mathbf{A} \text{ at generation 0} \right) \\
 &\quad \times \left(\text{The probability an } \mathbf{A} \text{ type lives to adulthood} \right) \\
 &= z N_A(0) V_A.
 \end{aligned}$$

Then, it is easy to see that $N_A(2)$ is given by

$$\begin{aligned}
 N_A(2) &= \left(\text{Fertility of } \mathbf{A} \right) \times \left(\text{The number of } \mathbf{A} \text{ at generation 1} \right) \\
 &\quad \times \left(\text{The probability an } \mathbf{A} \text{ type lives to adulthood} \right) \\
 &= z N_A(1) V_A = z (z N_A(0) V_A) V_A = N_A(0) (z V_A)^2.
 \end{aligned}$$

We can do this over and over again. We find

$$\begin{aligned}
 N_A(3) &= N_A(0) (z V_A)^3 \\
 N_A(4) &= N_A(0) (z V_A)^4 \\
 &\vdots
 \end{aligned}$$

We can easily extrapolate from this to see that, in general,

$$N_A(t) = N_A(0) (z V_A)^t$$

and a similar analysis shows us that

$$N_B(t) = N_B(0) (z V_B)^t$$

We are now ready to figure out a functional form for $P_A(t)$. From the definition of the frequency for \mathbf{A} , we have

$$P_A(t) = \frac{N_A(t)}{N_A(t) + N_B(t)}.$$

Now divide top and bottom of this fraction by $N_A(t)$ to get

$$\begin{aligned}
 P_A(t) &= \frac{\frac{N_A(t)}{N_A(t)}}{\frac{N_A(t) + N_B(t)}{N_A(t)}} \\
 &= \frac{1}{1 + \frac{N_B(t)}{N_A(t)}}.
 \end{aligned}$$

Now plug in our formulae for $N_A(t)$ and $N_B(t)$. Note the fraction

$$\begin{aligned}\frac{N_B(t)}{N_A(t)} &= \frac{N_B(0) (z V_B)^t}{N_A(0) (z V_A)^t} \\ &= \frac{N_B(0)}{N_A(0)} \left(\frac{V_B}{V_A} \right)^t.\end{aligned}$$

Using this in our frequency formula, we have

$$P_A(t) = \frac{1}{1 + \frac{N_B(0)}{N_A(0)} \left(\frac{V_B}{V_A} \right)^t}.$$

2.4.1 Examples

Let's redo our previous example using this new formula.

Example 2.4.1 Find the frequency of type **A** individuals at generation 1 given that initially the frequency of **A** individuals is $p_0 = 0.01$ and the probabilities of **A** and **B** are $V_A = 0.8$ and $V_B = 0.3$.

Solution Since $p_0 = 0.01$, we can use that to find the initial values of $N_A(0)$ and $N_B(0)$. We have

$$P_A(0) = 0.01 = \frac{N_A(0)}{N_A(0) + N_B(0)}.$$

Now rewrite this as by dividing top and bottom of the fraction by $N_A(0)$ to get

$$P_A(0) = 0.01 = \frac{1}{1 + \frac{N_B(0)}{N_A(0)}}.$$

We can then solve for

$$\begin{aligned}\frac{1}{1 + \frac{N_B(0)}{N_A(0)}} &= 0.01 \\ 1 + \frac{N_B(0)}{N_A(0)} &= \frac{1}{0.01} = 100 \\ \frac{N_B(0)}{N_A(0)} &= 99.\end{aligned}$$

Using the probabilities, we thus have

$$\begin{aligned}
 P_A(1) &= \frac{1}{1 + \frac{N_B(0)}{N_A(0)} \frac{V_B}{V_A}} \\
 &= \frac{1}{1 + 99.0 \left(\frac{0.3}{0.8}\right)^1} \\
 &= \frac{1}{1 + 99.0 \frac{3}{8}} \\
 &= \frac{1}{38.125} = 0.0262
 \end{aligned}$$

So the change in frequency of **A** is $P_A(1) - P_A(0) = 0.0262 - 0.01 = 0.0162$ with the new frequency given by $P_A(1) = 0.0264$. This is just like we calculated before. However, this formula makes it easy for us to find out what happens after many generations! Note, $P_A(5)$ would be

$$P_A(5) = \frac{1}{1 + 99.0 \left(\frac{0.3}{0.8}\right)^5}$$

and $P_A(15)$ would be

$$P_A(15) = \frac{1}{1 + 99.0 \left(\frac{0.3}{0.8}\right)^{15}}$$

2.4.2 Homework

Now use our new frequency formula to find some frequencies of **A** in the population. Note how, although we are not doing it yet, we can now do all the calculations to generate a plot of $P_A(t)$ versus generation t !

Exercise 2.4.1 Find the frequency of type **A** individuals at generation 5 given that initially the frequency of **A** individuals is $p_0 = 0.02$ and the probabilities of **A** and **b** are $V_A = 0.85$ and $V_B = 0.25$.

Exercise 2.4.2 Find the frequency of type **A** individuals at generation 6 given that initially the frequency of **A** individuals is $p_0 = 0.03$ and the probabilities of **A** and **b** are $V_A = 0.75$ and $V_B = 0.45$.

Exercise 2.4.3 Find the frequency of type **A** individuals at generation 10 given that initially the frequency of **A** individuals is $p_0 = 0.04$ and the probabilities of **A** and **b** are $V_A = 0.65$ and $V_B = 0.2$.

Exercise 2.4.4 Find the frequency of type **A** individuals at generation 5, generation 10 and generation 20 given that initially the frequency of **A** individuals is $p_0 = 0.02$ and the probabilities of **A** and **b** are $V_A = 0.75$ and $V_B = 0.3$. What do you think is happening as the number of generations increases?

Exercise 2.4.5 Find the frequency of type **A** individuals at generation 10, and generation 20 and generation 30 given that initially the frequency of **A** individuals is $p_0 = 0.01$ and the probabilities of **A** and **b** are $V_A = 0.6$ and $V_B = 0.2$. What do you think is happening as the number of generations increases?

2.4.3 Biology and the Model

Let's step back a minute and ask what we are doing. We have been interested in exploring how a population of two phenotypes spread throughout a population. Since the size of the population can change at each generation, it is much more useful to look at the frequencies of each phenotype in the population. We have developed a model that gives us insight into how this happens. Recall our recursion equation for the change in frequency for type **A**:

$$P_A(t+1) - P_A(t) = \frac{(P_A(t)(1 - P_A(t)))(V_A - V_B)}{P_A(t)V_A + (1 - P_A(t))V_B}.$$

It is traditional in evolutionary biology to look at the denominator term $P_A(t)V_A + (1 - P_A(t))V_B$ and try to understand what it means biologically. A common way to interpreting it is to tag the term $P_A(t)V_A$ as the **fitness** of type **A** in the population and the other term, $(1 - P_A(t))V_B$ as the fitness of type **B**. Hence, the denominator term is like a weighted or average fitness of the entire population. We often use a **bar** above a variable to indicate we are looking at that quantities *average value*. The fitness is usually denoted by the variable w and hence, this denominator term is \bar{w} .

To get additional insight, let's let the symbol ΔP_A represent the change $P_A(t+1) - P_A(t)$; we read the symbol Δ as **change in**. And let's drop all the (t) labels so we can rewrite our recursion as

$$\Delta P_A = \frac{P_A(1 - P_A)(V_A - V_B)}{\bar{w}}.$$

Finally, we could do this sort of analysis for any phenotype not just type **A**. So let's replace the specific term P_A by just p . Also the term $V_A - V_B$ is the difference in *fitness* between **A** and **B**. Let's denote that by Δf where f represents **fitness**. Then we can rewrite our recursion again as

$$\Delta p = p(1 - p) \frac{\Delta f}{\bar{w}}.$$

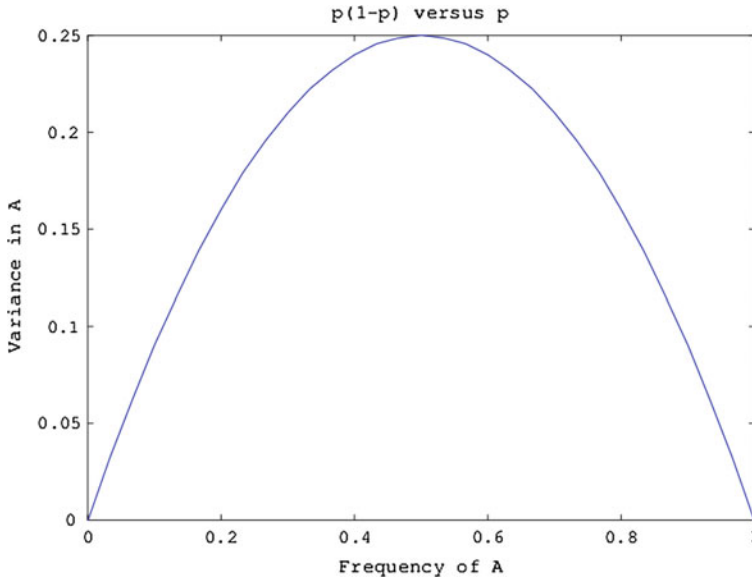


Fig. 2.1 The product $p(1 - p)$

If we assume the difference in fitness is constant, then the amount of change in the frequency of **A** is determined by the leading term $p(1 - p)$. To see what this means, consider the graph of p on the interval $[0, 1]$. We don't need to use any other values of p as it is a frequency so it ranges from 0 to 1. This graph is shown in Fig. 2.1.

The largest value the product $p(1 - p)$ can be seen by looking at the plot. This maximum value occurs at $p = 0.5$. Hence, the change in **fitness** is largest when the product $p(1 - p)$ is largest. If p is small or close to 1, the product $p(1 - p)$ is also small. We usually interpret our equation

$$\Delta p = p(1 - p) \frac{\Delta f}{\bar{w}}.$$

as the magnitude of **natural selection** acting on our population. Hence, we can see that **natural selection** is made largest (we say **maximized**) when the product $p(1 - p)$ is maximized. It turns out the product $p(1 - p)$ is known as the **variance** in genotypes in the population which is why we used the term *variance* in the title of Fig. 2.1! So our thought experiment has shed light on an interesting principle: **fitness is maximized when the variance of the genotypes in the population is maximal!!**

Remember, our goal with the mathematics and the computations we do is to gain insight into how to understand biological processes.

2.5 A Gentle Introduction to MatLab

In this book, we will use a computational tool called **MatLab** (see <http://www.mathworks.com/>) so that you can learn a little about a typical interactive program that many scientists use for their work. Once you have it installed, if you are using your laptop or computer, just click on the Matlab icon to get started. Next, create a folder or directory on your laptop for your work in this book. Something like **neanderthal**. All of our stuff for this book will then go into that folder. When you start up MatLab, you see the **command** screen shown in Fig. 2.2. Your screen image might be a bit different but it will be similar.

Next, we have to set your path. With Matlab running do this:

- On the left, there is a **File** option you can click on. Click on that and go to the **Save Path** option.
- When the program Matlab is searching for its instructions on how to do things, it first looks for them in all of the folders that were setup for Matlab as part of its installation. We don't want to put any of our stuff in those folders.
- We want our stuff in our personal folder **neanderthal**. So scroll down to find this folder, choose it and then click on the **Add Path** button to add this folder to the *search path*. Then choose **Save** and you are done.

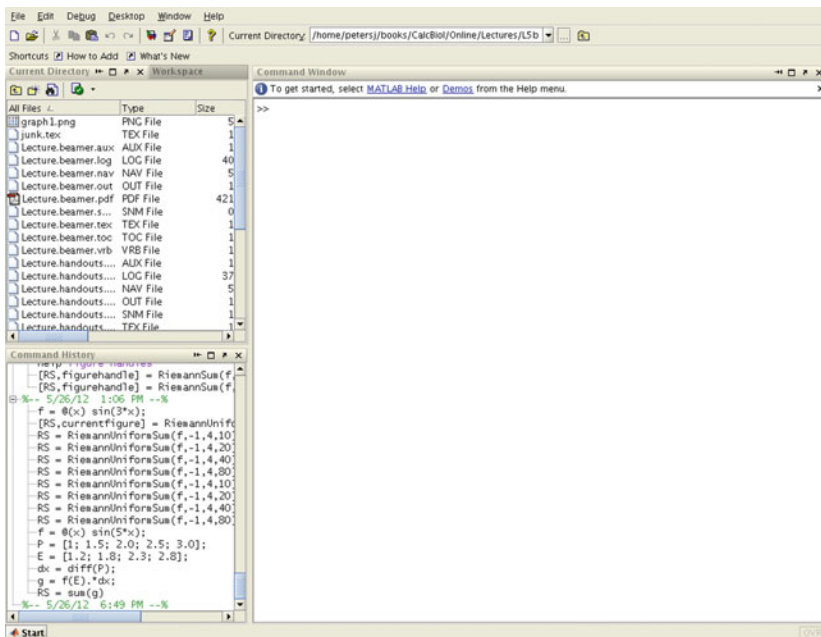


Fig. 2.2 The MatLab startup screen

- Now at the top of the running Matlab, choose the folder **neanderthal** so that Matlab is running in that folder. You can check this by typing **pwd** in the big middle window. It should spit back this folder name.

2.5.1 Matlab Vectors

First, set up our function. MatLab allows us to define a function inside the MatLab environment as follows

Listing 2.1: Defining a Matlab Function

```
f = @(x) (x.^2);
```

This defines the function $f(x) = x^2$. If we had wanted to define $g(x) = 2 * x^2 + 3$, we would have used

Listing 2.2: Defining a second Matlab Function

```
g = @(x) (2*x.^2+3);
```

In MatLab, variables that have columns are what in mathematics are called *vectors*. We will talk about *vectors*, *matrices* and so forth in Chap. 18 a bit later in this book. For now, consider this example. Set up the variable X as one that has columns. The ; between the numbers let MatLab know we want each number to start a new row. So X is a variable which has 3 rows and 1 column.

Listing 2.3: A column vector

```
X = [1; 2; 3]
X =
4      1
      2
      3
```

Note there is no semicolon at the end of the line below so Matlab displays what X is after we type the command. Adding the ; turns off the display.

Listing 2.4: A column vector with no display

```
X = [1; 2; 3];
```

If we had used , between the numbers instead of the ; we would have made a variable which consists of 1 row and 3 columns:

Listing 2.5: A row vector

```
Y = [1, 2, 3]
Y =
     1     2     3
```

Adding the ; turns off the display.

Listing 2.6: A row vector with no display

```
Y = [1, 2, 6, -8];
```

Now let Z be another column vector the same *size* as X .

Listing 2.7: New Vector Z

```
Z = [4;-2;6]
Z =
     4
    -2
     6
```

The MatLab notation $X.*Z$ means to multiply the components of X and Z as follows

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \cdot * \begin{bmatrix} 4 \\ -2 \\ 6 \end{bmatrix} = \begin{bmatrix} (1)(4) \\ (2)(-2) \\ (3)(6) \end{bmatrix} = \begin{bmatrix} 4 \\ -4 \\ 18 \end{bmatrix}$$

So in MatLab, we have

Listing 2.8: Component wise Multiplication of Vectors

```
X.*Z
ans =
     4
    -4
    18
```

If we wanted to square everything in X , we would write $X.^2$ to square each *component* creating a new *vector* with each entry squared.

Listing 2.9: A Vector Squared

```
X.^2
ans =
3      1
      4
      9
```

The way we set up the function

Listing 2.10: The function $f(x)$

```
f = @(x) (x.^2);
```

makes use of this. We don't know if the variable x is a *vector* or not. So we write $x.^2$ so that if x is a vector, we handle the squaring of each component properly. Otherwise, MatLab *ignores* the extra $.$ in front of the multiplication symbol $*$ when the variable x is just a number. So for our function, to find $f(2)$, we just type

Listing 2.11: $f(2)$

```
f(2)
ans =
4      4
```

as the argument sent into \mathbf{f} is just a number. But if we had sent in a vector, like \mathbf{X} , it would still be handled properly because of the way we wrote the code for \mathbf{f} . So for our function, to find f for all the values in X , we just type

Listing 2.12: $f(X)$

```
f(X)
ans =
      1
      4
5      9
```

2.5.1.1 Examples

Example 2.5.1 Let's set up some vectors. Write the Matlab code to set up the row vector $A = [1, 2, -5, 8]$ and the column vector

$$B = \begin{bmatrix} -4 \\ 2 \\ -5 \\ 10 \end{bmatrix}$$

Solution *We write*

Listing 2.13: Set up some vectors

```
A = [1,2,-5,8]
B = [-4;2;-5;10]
```

2.5.1.2 Homework

Exercise 2.5.1 Write the Matlab code to set up the row vector $A = [-2, 4, 3, -18]$ and the column vector

$$B = \begin{bmatrix} 6 \\ 10 \\ 7 \\ 1 \end{bmatrix}$$

Exercise 2.5.2 Write the Matlab code to set up the row vector $A = [4, 0, -50, 80]$ and the column vector

$$B = \begin{bmatrix} -40 \\ 22 \\ 35 \\ 8 \end{bmatrix}$$

Exercise 2.5.3 Write the Matlab code to set up the row vector $A = [1, 2, 0, -28]$ and the column vector

$$B = \begin{bmatrix} -6 \\ -32 \\ 43 \\ 1 \end{bmatrix}$$

2.5.2 Graphing a Function

To graph f we need to set up a variable which tells us how many data points to use in the plot. This variable is different from our partition variable. The **linspace** command below sets up a variable y to be a vector with 21 points in it. The first point is 1 and the last point is 3 and the interval $[1, 3]$ is divided into 20 equal size pieces. So this command **linspace(1,3,21)** creates y values spaced 0.1 apart:

$$\{y_1 = 1, y_2 = 1.1, y_3 = 1.2, \dots, y_{20} = 2.9, y_{21} = 3.0\}.$$

We use the pairs $(y_i, f(y_i))$ to make a plot by connecting the dots determined by the pairs using lines. To do the plot in Matlab is easy

Listing 2.14: Setting up a function plot

```
y = linspace(1,3,21);
plot(y,f(y));
```

We can add stuff to this bare bones plot.

Listing 2.15: Adding labels to the plot

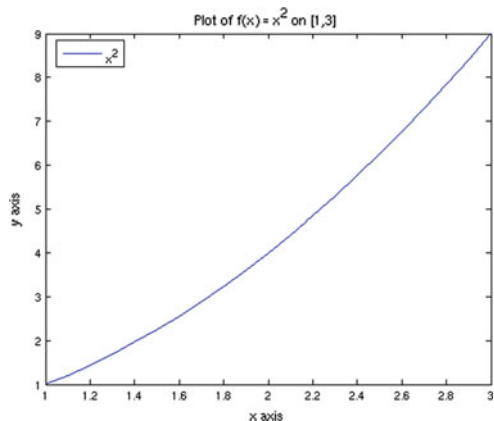
```
xlabel('x axis');
ylabel('y axis');
3 legend('x^2','location','best');
title('Plot of f(x) = x^2 on [1,3]');
```

where

- **xlabel** sets the name printed under the horizontal axis.
- **ylabel** sets the name printed next to the vertical axis.
- **legend** sets a blurb printed inside the graph explaining the plot. Great when you plot multiple things on the same graph.
- **title** sets the title of the graph.

The graph pops up in a separate window as you can see. Using the **file** menu, select **save as** and scroll through the choices to save the graph as a **.png** file—a Portable Network Graphics file. You'll need to give the file a name. We chose **graph1.png** and it is shown in Fig. 2.3.

Fig. 2.3 Graph of f



2.5.2.1 Examples

You should type in these commands in Matlab on your laptop or some other computer and make sure you know how to do all the steps. To turn in the homework, make a screen print of your Matlab session. Later we will take the session and put it into a word document but that is to come.

Example 2.5.2 Let's plot $f(t) = t^3 + 2t + 3$ on the interval $[-1, 1]$.

Solution *Type in these commands to see the plot—we don't show it here but you should see it!*

Listing 2.16: Graphing $f(t) = t^3 + 2t + 3$

```
1 f = @(t) (t.^3 + 2*t + 3);
  T = linspace(-1,1,41);
  plot(T,f(T));
  xlabel('t axis');
  ylabel('y axis');
6 legend('t^3+2t+3', 'location', 'best');
   title('Plot of f(t) = t^3+2t+3 on [-1,1]');
```

2.5.2.2 Homework

Graph the following functions on the given interval nicely with labels and so forth. You'll probably have to play with the **linspace** command to get a nice plot.

Exercise 2.5.4 Graph $f(x) = 2x + x^4$ on the interval $[-2, 3]$.

Exercise 2.5.5 Graph $f(t) = 2t - 5t^2$ on the interval $[-1, 2]$.

Exercise 2.5.6 Graph $h(y) = 12y - 6y^3$ on the interval $[-4, 4]$.

2.5.3 A Simple Virus Infection Model

Let's look now at a sample use of MatLab to plot some data. We will use some data arising from computational models of West Nile Virus infection. West Nile Virus is in the Flavivirus family which is a family of viruses transmitted by mosquitoes and ticks with an impact that is important for varied sociological and economic reasons. These diseases include dengue, yellow fever, tick-borne encephalitis and West Nile fever. They are widely distributed throughout the world with the exception of the polar regions, although a specific flavivirus may be geographically restricted to a continent or a particular part of it. With global warming, these single-stranded RNA viruses are entering the radars of more regions of the world than ever. West Nile virus, for example, has emerged in recent years in temperate regions of Europe and

North America, presenting a threat to public and animal health. The most serious manifestation of the West Nile virus infection is fatal encephalitis (inflammation of the brain) in humans and horses, as well as mortality in certain domestic and wild birds. The virus is maintained in nature through a transmission cycle involving mosquitoes and birds. Children will usually experience an apparent or a mild febrile illness. Adults may experience a dengue-like illness while the elderly may develop an encephalitis which is sometimes fatal. The diagnosis is usually made by serology although the virus can be isolated from the blood in tissue culture. No vaccine for the virus is available and there is no specific therapy. The West Nile Virus infections feature a substantial up-regulation of cell surface molecules of a variety of cell types which are in the G_0 resting state of cell division. Curiously, cells that are dividing (i.e. in the G_1 state) do not have this up-regulation. Although many cell surface molecules are expressed at a much higher rate in the quiescent cells, the model developed in Peterson et al. (2015) so far focuses primarily on the increase in the MHC-I complex. A simulation model was built as closely tied to the epidemiological and biological literature as possible (although there is a mathematical framework as well). The model was then used to run simulations to help us understand the data on West Nile Virus infections that have been collected by cellular biologists that involve multiple hosts. In a survival experiment, a population of mice of size N are all infected with the same amount of antigen. In the laboratory, the level of antigen used is measured in *pfu* or *plaque forming units*. This level is actually measured visually by looking at a slide infected with antigen and seeing how many “plaques” form when the virus is exposed to a phage. Hence, it is a somewhat imprecise measurement even in the cell biologist’s laboratory setting. After these N mice are infected, we wait to see how many die. We then repeat the experiment using N mice for a reasonable number of different pfu levels. Say we did this for 12 pfu levels on 12 different populations of size N . We could then graph the number of mice that survive versus the pfu level. This is called a survival experiment and as you can see, it is fairly expensive to mount, requiring $12N$ mice. If we used $N = 100$, this would be 1200 mice at perhaps \$20.00 per mouse. The experiment thus costs over \$24,000 for the mice alone. Needless to say, this is costly. Now a traditional survival experiment gives a survival plot which smoothly decays down to a survival of 0 at high pfu level. West Nile Virus has a peculiar survival curve which is shown in Fig. 2.4. Note that survival actually increases sometimes with increasing pfu. This is quite different from the data that we see in other viral infections. The computer simulations are used to generate this kind of survival data for 10 hosts infected at 18 different pfu levels. This data is shown in Fig. 2.6b and has been entered into a file called **survival.dat** (Fig. 2.5). We want to generate a plot of the number of mice that survive versus the logarithm of the pfu level. We can use MatLab to do this easily.

We want to load the information contained in the file **survival.dat** into a MatLab variable. You want to make sure the file is a simple text file—in a Windows machine, use *Notepad* and in a Mac, use *textedit*. Just make sure you save the file in a text format. Also, as we said above, make sure you set the path in MatLab so that MatLab can find your data file. Now, note that we have entered the data in a three column format.

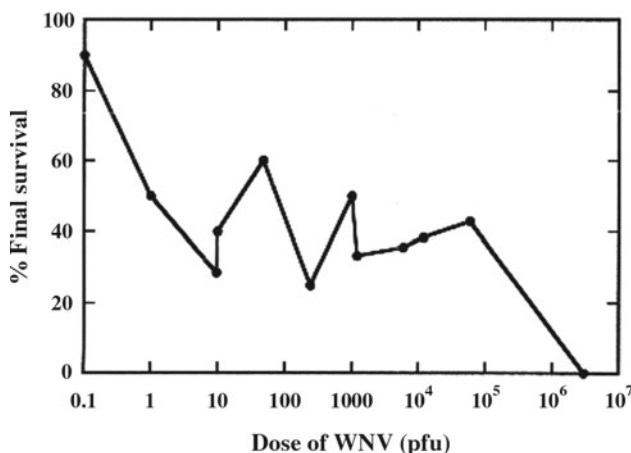


Fig. 2.4 Actual survival curve

Surviving Mice	Infection Level	Surviving Fraction	Surviving Mice	Infection Level	Surviving Fraction
10	100	0.999990	4	25000	0.439213
10	250	0.999982	2	50000	0.358584
10	500	0.808539	2	75000	0.367035
8	750	0.589315	3	90000	0.379991
10	1000	0.739119	4	120000	0.359158
8	2500	0.497502	0	360000	0.296874
5	5000	0.454120	1	600000	0.301097
3	7500	0.425226	2	900000	0.356034
3	10000	0.370141	0	1200000	0.297576

Fig. 2.5 Simulation WNV survival data

The numbers in each row are entered separated by spaces. You can't use comma's in numbers here. For example, if you entered 25000 as 25,000, the would be interpreted in MatLab as the second entry in the row is 25, the third entry in 000 and the fourth entry is 0.439213. Of course, there should be only 3 entries in each row, so this would cause MatLab to get very confused. To load the data in this file, we use the following command.

Listing 2.17: Loading Data From a File

```
> Data = load('survival.dat');
```

The information in the file is now placed into a MatLab variable called **Data** which consists of 18 rows with 3 columns each. We take all the numbers in column one and store them in a new variable called **Survival** with this line.

Listing 2.18: Putting Survival Data into a Matlab Variable

```

Survival = Data(:,1)

Survival =

5      10
      10
      10
      8
      10
10      8
      5
      3
      3
      4
15      2
      2
      3
      4
      0
20      1
      2
      0

```

You see the data is echoed to your screen. This is because we did not put a final ; after the line. If we had, there would have been no additional output. Note the syntax here. The command **Data(:,1)** tells MatLab to use all the data in column 1 to load into the variable **Survival**. We could also have used the command **Data(1:18,1)**, but this is a little more work as we need to know exactly how many rows of data there are. The first way, using a : is a lot easier! Next, we load the second column of data into a variable called **Pfu** where we have the ellipsis to indicate data we are not showing because it takes up so many lines! Of course, in the MatLab environment, you would see all the data printed out.

Listing 2.19: Loading data into MatLab variables

```

Pfu = Data(:,2)

Pfu =

5      100
      250
      500
      ...
      900000
10     1200000

```

Finally, we load the last or third column into a variable called **HealthyPercent**.

Listing 2.20: Loading Healthy Percent

```

HealthyPercent = Data(:,3)

HealthyPercent =

5      1.0000
      1.0000
      0.8085
      ...
      0.3560
10      0.2976

```

Now we will look at plots involving the logarithm of the data. You have seen the logarithm function in high school and we are going to explain what logarithms are very carefully in a bit, but for now, let's assume you know about them. So, if we wanted to graph **Survival** versus the logarithm of **Pfu**, we first compute the natural logarithm of each number in **Pfu** with the line

Listing 2.21: Logarithm of Pfu

```
logPfu = log(Pfu);
```

This takes the column of information in the variable *Pfu* and applies the natural logarithm to each entry. Now we haven't discussed natural logarithms yet, but you probably remember them from high school classes. Our discussion will be in Chap. 10. But you should be able to remember enough to get the gist of what we are doing here. Now we put the ; on the end of this line and so the new variable *logPfu* is not printed out. If we had left off ;, we would have seen

Listing 2.22: Listing logPFU Values

```

logPfu = log(Pfu)

logPfu =

4      4.6052
      5.5215
      6.2146
      6.6201
9      6.9078
      7.8240
      8.5172
      8.9227
      9.2103
14     10.1266
      10.8198
      11.2252
      11.4076
      11.6952
19     12.7939
      13.3047
      13.7102
      13.9978

```

The plot is then generated with the lines

Listing 2.23: Simple logPfu versus Surviving Hosts Plot

```

plot(logPfu, Survival);
xlabel('Logarithm Of PFU Level');
3 ylabel('Surviving Hosts');
axis([4.0 14.5 -0.5 10.5]);
title('Survival Experiment: 10 Hosts, final time = 960');

```

The plot command here will use default colors and will be generated with no axis labels, title and so forth. The **xlabel**, **ylabel** commands above set the axis labels to the string we want to use. The **title** command allows us to pick the title for our graph. Finally, the **axis** command allows us to override the default minimum x, maximum x, minimum y and maximum y values used in the plot. MatLab automatically chooses these for you, but the **axis** command lets you choose more pleasing settings if you want. The first line, **plot**, generates the figure right away and you will see it pop up. As each of the other lines is typed and you hit the carriage return key, the strings are added to the existing figure. When all is done, you can go to the figure and save it as a graphics file with an appropriate extension. For us, since we want to add these files to word or open office documents, we choose **.png** or **.jpg** files. You will have to choose where you save the file also before you save it.

A similar set of lines generates the plot of **HealthyPercent** versus **logPfu**.

Listing 2.24: Healthy Percent versus logPfu Plot

```

plot(logPfu, HealthyPercent);
xlabel('Logarithm Of PFU Level');
ylabel('Percentage of Healthy Cells Left');
axis([4.0 14.5 -0.5 10.5]);
5 title('Survival Experiment: 10 Hosts, final time = 960');

```

The plots we have generated are shown in Fig. 2.6a, b.

Note, that the simulated survival curve is qualitatively similar to the real data shown in Fig. 2.4. To end this section, note that the entire MatLab session to build both plots is quite compact. Here it is without commentary.

Listing 2.25: Entire Matlab Survival Data Session

```

Data = load('survival.dat');
Survival = Data(:,1);
Pfu = Data(:,2);
HealthyPercent = Data(:,3);
5 logPfu = log(Pfu);
plot(logPfu, Survival);
xlabel('Logarithm Of PFU Level');
ylabel('Surviving Hosts');
axis([4.0 14.5 -0.5 10.5]);
10 title('Survival Experiment: 10 Hosts, final time = 960');
plot(logPfu, HealthyPercent);
xlabel('Logarithm Of PFU Level');
ylabel('Percentage of Healthy Cells Left');
axis([4.0 14.5 -0.1 1.05]);
15 title('Survival Of Healthy Cells : 10 Hosts, final time = 960');

```

Not bad, eh?

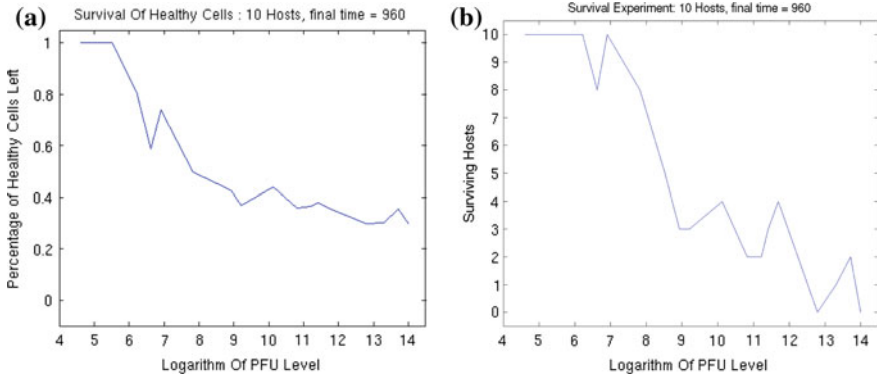


Fig. 2.6 Survival experiment results for 10 hosts and 18 pfu Levels. **a** Healthy cells left in survival experiment: we infect 10 hosts with 18 different pfu levels. The simulation is run for 960 time units. **b** Survival curve: we clearly see up and down variability in survival as pfu levels increase

2.6 Long Term Consequences

Now, let's go back to our original question about long term behavior of the frequency of **A**. Let's generate the plot in Matlab. Recall the frequency formula for **A** is given by

$$P_A(t) = \frac{1}{1 + \frac{N_B(0)}{N_A(0)} \left(\frac{V_B}{V_A} \right)^t}.$$

So we need to setup variables in Matlab for various parameters. We will use

- **VA** to represent V_A .
- **VB** to represent V_B .
- **p0** to represent p_0 .
- **r** to represent the ratio $N_B(0)/N_A(0)$. It is easy to see that when we solve for this ratio, we find $r = 1 - 1/p_0 = (1 - p_0)/p_0$. In Matlab, this is **r = (1-p0/p0)** also!
- **v** to represent the ratio V_B/V_A .
- **N** to represent the last generation we are interested in looking at.

We set up these initializations like this

Listing 2.26: Initialize our constants

```

VA = .8;
VB = .3;
p0 = 0.01;
r = (1-p0)/p0;
5 v = VB/VA;
N = 25;

```

We then define the **P** function. Since Matlab starts everything at the counter 1, we will use $t - 1$ in our formula to make sure the Matlab **P(1)** corresponds to the mathematical $P(0)$. The Matlab line is

Listing 2.27: Setting up the frequency function

```

P = @(t) 1./(1+r*v.^(t-1));

```

Note we use the Matlab **./** and **.^** to allow our function to deal with data that is a vector (we talked about this!). Then, we set up our **linspace** to give us generations 0 to 25 and generate the plot. We add labels and a title too.

Listing 2.28: Plotting the frequency of A

```

T = linspace(0,N,N+1);
plot(T,P(T),'o');
xlabel('Generation');
ylabel('Frequency of A');
5 title('The Frequency of Genotype A vs Generation');

```

This shows an open circle at each generation. We could let Matlab connect the open circles to show a plot that is smooth, but of course, this plots data points for generations like 2.75 which don't really make sense. Still, generalizing from data points each generation to data points at in between times is something we often do. We show the generated plot in Fig. 2.7.

2.7 The Domestication of Wheat

Have you ever thought about how the varieties of wheat we use today came about? We are going to tell you a plausible story which we found in the fantastic book by Steve Mithen on what we know about humanity in the post ice age time (Mithen 2004). Here is the tale paraphrased from Mithen's words—you should really read his book though!

In wild wheat the **ear** that contains the **spikelets** is very brittle and when ripe, the ear falls apart spontaneously. This scatters the seed on the ground and it is difficult to harvest it from there. However, about 1 or 2 out of a million wheat plants are

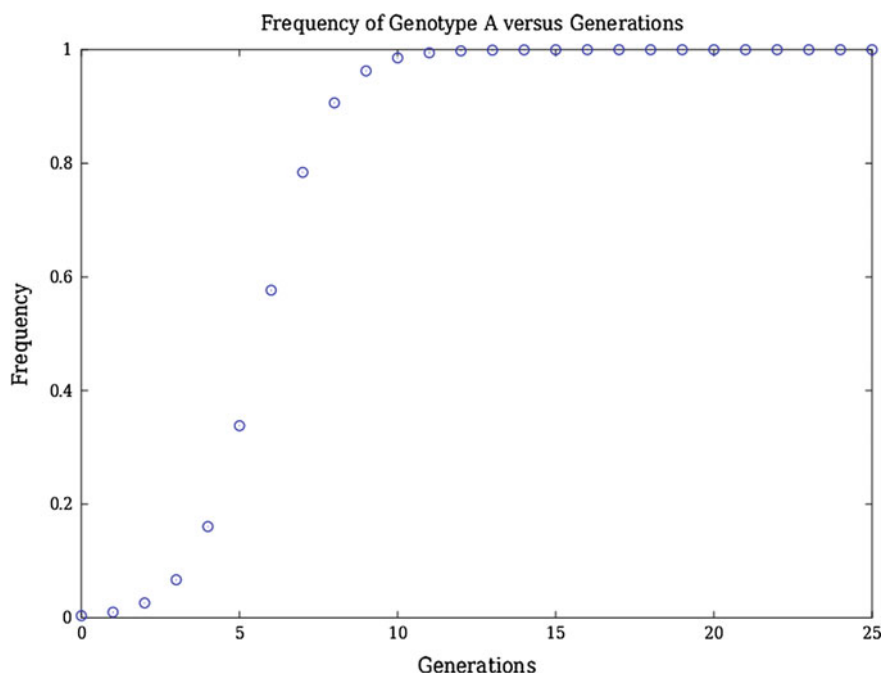


Fig. 2.7 The frequency of A

genetic mutants whose ears are not very brittle. When ripe, these ears do not shatter spontaneously and so they are easier to collect. As Mithen says on pp. 37–38

Imagine the situation in which a small party of Natufians [the people Mithen is talking about here] arrived to begin cutting a stand of wild cereals. If the wheat was already ripe, then much of the grain from the brittle plants would already have been scattered. But the rare non-brittle plants would still be intact. So when the stalks were cut, the grain from those plants would still be intact. So when the stalks were cut, the grain from those plants would have been relatively more abundant within the harvest than it had been in the woodlands or on the steppe. ... The next step is to imagine what would happen if the Natufian people began to reseed the wild stands of cereal by scattering grain saved from a previous harvest....

When this began to happen, the frequency of the non-brittle variant would have been enhanced. Over many generations, this frequency would have continued to climb until eventually the non-brittle variety was dominant. This was a fundamental change in how humans lived for many reasons. One is that the wheat now required human intervention to grow as the non-brittle ears. We are going to use our nice viability model to understand how this might happen. If you look at Hillman and Davies (1990), you can find out a lot more about how to make educated guesses about how long it might take for the domesticated (i.e. the non-brittle variety) wheat to become dominant. Hillman and Davies suggest somewhere between 20 and 50 cycles of sowing and harvesting for this to happen.

Our model will work like this. We will let the **A** variant correspond to the domesticated wheat variety and start its probability as $V_A = 1.0e-6$ which corresponds to 1 non-brittle wheat plant per one million brittle plants. This matches the frequencies we think are realistic. We won't assume the Natufians planted their own wheat plots every year. They might have harvested wild stands and every few years planted their own stands because they had settled down to one region for awhile. Whenever they planted their own stands, the probability of the domestic variety would go up a bit. So we will let **domfrequency** be our MatLab variable which tells us how often the Natufians planted their own wheat; perhaps every 5 years or so. So if we had 20 Nautifian plantings, we would be looking at a total of $5 \times 20 = 100$ harvestings, 90 of which are wild and only 10 are human planting events. We will let **vdom** be the probability that the wild wheat generates the domesticated wheat mutant: we typically set this as **vdom = 1.0e-6** and then set the probability of the domesticated variety **VB** to be a multiple of **vdom**. We will keep our model simple: every **domfrequency** harvests, we reset the domesticated probability to be **VB = vdom * (1+ epsilon)^j** where **j** is our counter which tells us how many Nautifian plantings there have been. So for example, if the Natufian plant every 5 years, for the first 5 years, the domesticated variety has probability **VB = vdom * (1+ epsilon)** where **epsilon** is the amount the domesticated probability goes up each planting. A typical value might be **epsilon = 0.8** in which case the domesticated probability is **vdom** for the first 5 years, is **vdom*1.8** for the next 5, **vdom*1.8^2** for the next five and so on. Even though **vdom** is very small—say $1.0e-6$, after 20 Nautifian plantings, the probability is **vdom * 1.8^(20) = 0.127**. We reset the probability for the brittle variety to be **1 - VB** at each planting so after 20 plantings, the probability of brittle has fallen to 0.873. And of course, we use a given domesticated probability for say 5 years, so it increases for 5 years following our usual P_A equation during those years.

When we do this simulation, eventually, **vdom*(1+epsilon)^j** will exceed the wild probability. For example, after a number of Nautifian harvests, we might find **vdom*(1+epsilon)^j = 0.51**. At that point, **VA = 0.51** and **VB = 0.49**. The ratio **VB/VA > 1** and in the P_A equation, the value of P_A rapidly switches from 0 to 1! This switch point is what we are looking for as it tells us when the domesticated variety becomes dominant. So we will implement this with a **while loop** like this:

Listing 2.29: Finding the switching point

```

% set our simulation counter to 1: this is our first harvesting
i = 0;
% as long as quitloop is 0, we will keep going.
% in the calculations inside the loop as soon
5 % VAnew exceeds VBnew, we know domesticated
% wheat is dominant. So we set quitloop = 1
% so we will stop our calculations there.
quitloop = 0;
while quitloop < 1
10 % find the probability ratio
    v = VBnew/VAnew;
    probratio(i) = v;
    % now calculate the new Pdom
    % until a Nautifian planting is done which
15 % will increase the domesticated probability
    % this loop is for domfrequency harvests
    for j = 1:domfrequency
        k = i+domfrequency + j;
        PWild(k) = P(k,v);
20        Pdom(k) = 1- PWild(k);
        if (Pdom(k) > PWild(k) )
            % print out some variables
            v
            VAnew
25            VBnew
            k
            i
            % setting quitloop = 1 here will stop our
            % calculations once we go to the top
            % and see the test while quitloop < 1
30            quitloop = 1;
            break;
        end
    end
    % As long as we are ok to continue
    % we reset the probabilities because of the
    % Nautifian planting
    VBnew = VBnew*(1+epsilon);
    VAnew = 1-VAnew;
40    % we increase the counter for the number of
    % Nautifian plantings
    i = i+1;
end

```

Now we want to wrap all this up into a function and do some plots to, so we take the basic loop we just looked at and at some stuff. The basic code as a function returns a number of things: **Pdom**, the vector of the domesticated variety frequencies, **PWild**, the vector of brittle variety frequencies, **VDom**, the vector of domesticated probabilities, **VWild**, the vector of brittle variety probabilities and two more. For convenience, we return the ratio of the domesticated and brittle variety probabilities, **probratio** and the vector of harvest times **T** so we can construct additional plots easily.

Listing 2.30: Our domesticated wheat model

```

function [probratio,Pdom,PWild,VWild,Vdom,T] = domesticatedwheat(Vdom,domfrequency
,epsilon)
%
% Vdom = probability of domesticated wheat
% domfrequency = how often harvest strategy leads to
5 % an increase in domestic wheat probability
% epsilon = the change in the probability of domesticated wheat which is
% modeled as (1 + epsilon) VA
%
% set the initial domesticated probability
10 VA = 1-Vdom;
% set the initial brittle probability
VB = Vdom;
% set the initial frequency of the domesticated variety
p0 = VA;
15 % find r = NB(0)/NA(0) = (1 - p0)/p0
r = (1-p0)/p0;
r
% define the population function for domesticated wheat
P = @(t,v) 1./(1+r*v.^(t-1));
20 %
% every domfrequency generations, the probability of
% domesticated wheat is increased by the multiplier 1 + epsilon
%
% setup vectors to hold wild and domesticated probabilities
25 % as well as the ratios VB/VA at each step
PWild = [];
Pdom = [];
probratio = [];
% initial VAnew and VBnew
30 VAnew = VA;
VBnew = VB;
% we start our model with counter i = 1
i = 0;
%
35 quitloop = 0;
while quitloop < 1
    v = VBnew/VAnew;
    probratio(i) = v;
    for j = 1:domfrequency
40        k = (i-1)*domfrequency + j;
        PWild(k) = P(k,v);
        Pdom(k) = 1- PWild(k);
        VWild(k) = VAnew;
        Vdom(k) = VBnew;
45        if (Pdom(k) > PWild(k) )
            k
            i
            quitloop = 1;
            break;
50        end
    end
    VBnew = VBnew*(1+epsilon);
    VAnew = 1-VBnew;
    i = i+1;
55 end
% clear any old plot and plot the results
clf;
N = k;
T = linspace(0,N,N);
60 plot(T,PWild,'o');
xlabel('Generations');
ylabel('Frequency of Wild Wheat');
title('The Frequency of Wild Wheat vs Generation');
end

```

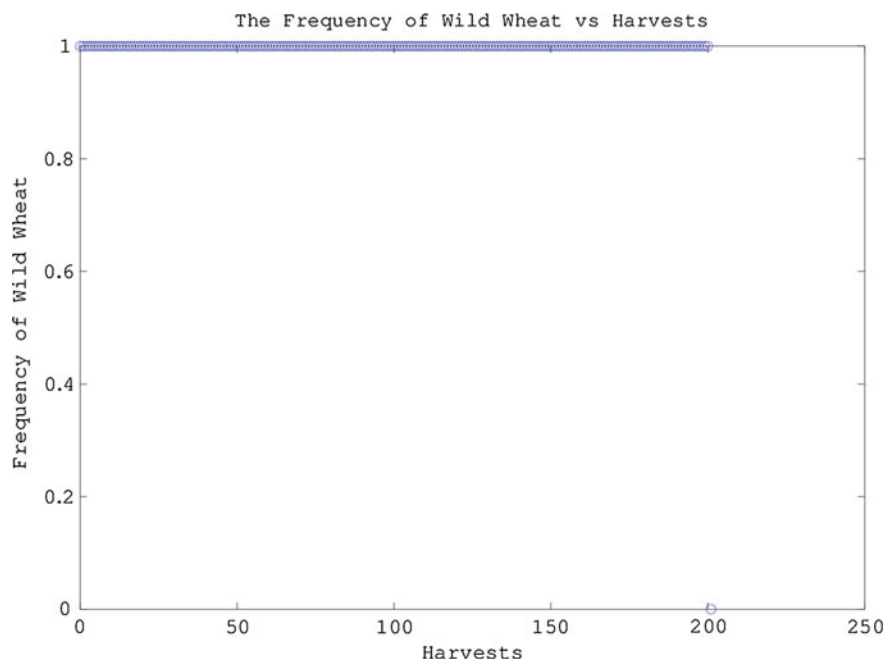



Fig. 2.8 Domesticated wheat dominance: Nautifian harvests every 5 years, domesticated wheat probability multiplier is 0.4 and the initial domesticated wheat probability is $1.0e-6$

Now let's try a simulation. Let's set **epsilon** = 0.4 and **domfrequency** = 5 and use an initial domestic wheat probability **vdom** = $1.0e-6$. Then in MatLab run this:

Listing 2.31: Sample Domesticated Wheat Simulation

```
[probratio ,Pdom,PWild ,VWild ,Vdom,T] = domesticatedwheat(1.0e-6,5,.4);
```

This returns the plot we see in Fig. 2.8. If you look at this graph, you'll see all the points are at the top—where the frequency of the brittle variety is 1. It isn't until you get to harvest 200 that you see the jump down to 0.

So we can clearly see that the switch occurs after $k = 200$ harvests; i.e. about 200 years which matches well what Himman and Davies say in their paper. However, It is even easier to see the switch if we plot the domesticated wheat probability versus harvests. This is easily done as shown below.

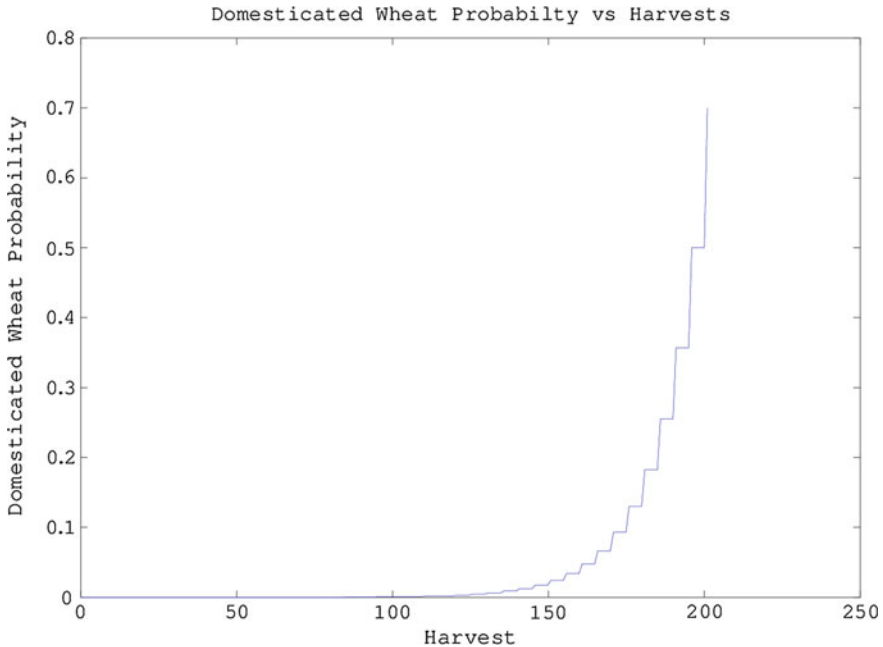


Fig. 2.9 Domesticated wheat probabilities: Natufians plant every 5 years, domesticated wheat probability multiplier is 0.4 and the initial domesticated wheat probability is $1.0e-6$

Listing 2.32: The Domesticated Wheat Probability versus Harvests

```
plot(T,Vdom);
xlabel('Harvest');
ylabel('Domesticated Wheat Probability');
4 title('Domesticated Wheat Probability vs Harvests');
```

We can see the resulting plot in Fig. 2.9. Again, it is easy to see the domesticated wheat variety becomes dominant at harvest 200. Note how the probability of our domesticated variety jumps up every 5 years as that is the interval between the Natufian plantings. We could do a similar graph for the probability for the brittle variety too. Cool beans!

2.7.1 Project

Your project is to use our viability models to develop a domesticated wheat dominance model just as we have done in this section. First, download the MatLab code, **domesticatedwheat.m** from the course website as usual. Make sure you start up

MatLab in your working directory and, if you do, after you download the file, you will see it in the left panel when MatLab is up and running. For our project, use the following basic template for our simulations: in this template, we are showing the same values we used in the example we worked out above.

Listing 2.33: Sample Domesticated Wheat Simulation

```

1 % this is the basic simulation
  Vdom = 1.0e-6;
  domfrequency = 5;
  epsilon = 0.4;
  [probratio ,Pdom,PWild ,VWild ,Vdom,T] = domesticatedwheat(Vdom,domfrequency , epsilon)
  ;
6 % this is the plot for the domesticated wheat variety probabilities
  plot(T,Vdom);
  xlabel('Harvest');
  ylabel('Domesticated Wheat Probability');
  title('Domesticated Wheat Probability vs Harvests');

```

Then, all you have to do is plug in the values you are supposed to use for the project. This is what we want from you. The project report is in Word using singlesspaced format. This is a **50 Point** project.

Introduction: We want three pages on a careful discussion of wild versus domesticated wheat using primary sources. Make sure you reference them using any standard scheme: i.e. reference them in the text with a number and list them at the end of the report in a section called **References**. This is **12 Points**.

Annotated discussion of the code domesticatedwheat.m: Here you explain very carefully what the code **domesticatedwheat.m** is doing. Think of explaining this code to a literate person who does not know MatLab—a friend or family member. We are not interested in you simply rehashing the explanations in the text. Try to be fresh and original. MatLab code should be bold font and explanations should be in italic font prefaced by the usual % sign. This is **10 points**.

Simulation One: Use **Vdom = 9.0e-7**, **domfrequency = 6** and **epsilon = 0.6**. Generate two plots: one is the frequency of wild wheat and the other is the domesticated wheat probabilities. Write a short introduction to the simulation problem and then insert the MatLab code to run the simulation. Just annotate this code lightly as the other section did the main work. Then insert the two plots and comment on what they show. This is **10 Points**.

Simulation Two: Use **Vdom = 1.5e-6**, **domfrequency = 4** and **epsilon = 0.3**. Generate two plots: one is the frequency of wild wheat and the other is the domesticated wheat probabilities. Organize the simulation results just like the first one. This is **10 Points**.

Conclusion: Here explain what the simulations are showing. Go back to how we derive the viability model and discuss why it is reasonable to use this model for this problem. For example, the viability model assumes the fertility of both phenotypes is the same. Is that reasonable? Think carefully about all the assumptions! This is **8 Points**.

References: This will show the references you used to prepare your report; make sure you reference this text as you use the code and so forth from it.

References

- G. Hillman, M. Davies, Measured domestication rates in wild wheats and barley under primitive cultivation, and their archaeological implications. *J. World Prehistory* **4**, 157–222 (1990)
- R. McElreath, R. Boyd, *Mathematical Models of Social Evolution: A Guide for the Perplexed* (University of Chicago Press, Chicago, 2007)
- S. Mithen, *After The Ice: A Global Human History, 20,000–50,000 BC* (Harvard University Press, Cambridge, 2004)
- J. Peterson, A.M. Kesson, N.J.C. King, A simulation for flavivirus infection decoy responses. *Adv. Microbiol.* **5**(2), 123–142 (2015)

Calculus for Cognitive Scientists

Derivatives, Integrals and Models

Peterson, J.K.

2016, XXXI, 507 p. 105 illus. in color., Hardcover

ISBN: 978-981-287-872-4