Chapter 3
A basic population model

3.1 Characterizing populations

One way to describe populations is to look at how many individuals they contain at various times. Or, instead of individuals, it may be more reasonable to consider total biomass—the total weight of all individuals in the population combined. For example, the number of trees may not be as important as their total weight, or the total area of their canopy. Density may also be relevant—how many individuals occupy a unit of area, or the percentage covered by the population in an area. All of these are gross properties of populations that can enter models.

Additional properties that can be taken into account include the age structure—the portions of the population of various ages, and the size structure—the portions of the population of various sizes. Such detail can be important because juveniles or older individuals may not reproduce. Genetic structure can also be important; it is often left out of ecological models, but evolutionary directions can affect the ecology as well.

Another important measure is the rate of change—how fast the population is changing. A population can be constant, increasing, or decreasing, or can fluctuate in complex ways.

Remember that a model is just a simpler view of something complex, and that all scientific models are just approximations of nature. When the full complexity cannot be understood, which is almost always, we can try to construct a simplified model in hopes of finding the essence.

3.2 Bacterial growth

The place to start a discussion of basic population models is in discrete-time at the macroscale—the most basic. Consider a hypothetical strain of bacteria reproducing every hour. Suppose a colony is started with a single bacterium when the clock reads zero, symbolized $t = 0$, and one hour later that bacterium divides into two, by time $t = 1$. Each of those divides in two one hour after that, by time $t = 2$. And so forth. Suppose you start this culture on a lab bench on Monday, let it go unchecked, and then come back late on Friday to see the results.

If the colony grows unchecked in this way, how many bacteria will there be at the end of the work week, after five full days of growth? You could figure that out on a calculator, or more easily you could use computer code. More easily, that is, once you know computer coding.

Because computer coding is becoming embedded in almost every aspect of life, appreciating the basics of coding is meaningful for each educated citizen of this century, and this book will expose you to the basics.

3.3 A first computer model

Below are two lines of computer code forming a program that models the bacterial colony as it doubles every hour. If you have not seen computer code before, don't be frightened by this; we will go through it carefully below.

```
N=1; t=0;
while(t<=5*24) { print(N); N=N*2; t=t+1; }
```

The above program is written in a generic programming language—it could be run as the programming language $\mathcal{R}$,⋆ as the language AWK,⋆ or, with minor adjustments as the language C,⋆ Java,⋆ or a number of others. This particular code is essentially identical in many languages.

The first "statement" is `N=1`. That instructs the computer to set the number of bacteria, `N`, equal to 1. A semicolon (`;`) ends the statement and separates it from subsequent statements. The next statement is `t=0`. That instructs the computer to set the time `t` to 0. One bacterium at time zero forms the "initial conditions," and once the computer has finished that line, the program is said to be "initialized."

The second line of the program is more involved, but can be understood in two parts. The first part on the left, `while(t<=5*24)`, instructs the computer to repeat a set of code for 5 simulated days of 24 hours each. The second part is the code to be repeated, within braces on the right, `{...}`.

Considering the first part, `while` is a "keyword" that instructs the computer to repeat something until the "condition" in parentheses is no longer true. In this case, inside the parentheses is `t<=5*24`, which itself consists of three parts,

`t`, `<=`, and `5*24`. The first part, `t`, represents the time, which has just been initialized to zero in the previous line of code. The second part, `<=`, is the symbol for "less than or equal to." Six such "comparison" symbols are possible, `==`, `<`, `<=`, `>`, `>=`, and `!=`, representing comparison for equal, less than, less than or equal, greater than, greater than or equal, and not equal, respectively. In the third part, the asterisk (`*`) is a symbol for multiplication, so `5*24` means "five times twenty-four," a way to represent the number 120, or the number of hours from Monday to Friday—the amount of time the hypothesized bacterial culture is to reproduce.

Computer coding is an exacting business, where tiny variations can make huge differences. The computer is the ultimate literal interpreter. An example of this just slipped by in the previous paragraph. In coding, a single equals sign, `=`, means "change something to be equal to," whereas two consecutive equals signs, `==`, means "compare to see if two things are the same."

If you are accustomed to coding, you will already be familiar with such subtleties; if this is all new to you, it is something to get used to. Various primers on the web can help, but don't be discouraged if it seems difficult at first; computer coding turns out to be one of the easiest things to jump into but one of the most difficult areas of all human endeavour to get exactly right. Time and patience will assist.

Getting back to the code, the phrase `while(t<=5*24)` means, in this case, to repeat something as long as the time, `t`, is less than or equal to 120 hours, 5 times 24. And that something to be repeated appears within braces to the right,

`{...}`. (By the way, many programming languages use three main symbols for grouping information—called braces, `{ }`, brackets, `[ ]`, and parentheses, `( )`. They are used for various kinds of groupings, but unfortunately their usage is not consistent across all languages.)

The first statement within braces is `print(N)`. (Refer back to the two-line program on page 20.) "Print" is a term left from the days when computers would communicate largely by printing on paper. Now the term just means "display." The statement thus means "display the number of individuals in the population, `N`, at this time." That was set to 1 in the previous line, so when the computer runs `print(N)` for the first time, it will display the number 1, typically on your screen.

The next statement, `N=N*2`, is read "`N` equals `N` times two." It is similar in form to the statement on the first line, `N=1`, which started things off with a single bacterium. The '`N=`' part is the same. It tells the computer that the number of bacteria, `N`, is about to change. (Of course, the computer has no clue what the program is about—that you are running a program about bacteria.) What `N` will change to is immediately to the right of the equal sign, `N*2`. The asterisk (`*`) means multiply. So this statement thus tells the computer to double the value of `N`. That is what the hypothesized bacterial population does every hour, so this statement models that doubling.

The third statement on the line, `t=t+1`, is read "`t` equals `t` plus one." It is similar in form to the statement on the first line, `t=0`, which started things off with a clock time of zero.

In other words, in this example of letting bacteria grow for a five-day work week, we are taking midnight Monday morning to be hour zero. Five days later, at midnight Friday night, that becomes hour 120 (5 days times 24 hours per day equals 120 hours). So similarly, `t=` tells the computer that time `t` is about change. Following the equals sign is what it should change to, `t+1`, or one more than what it is at the moment, which advances the time by one hour. This is a *discrete time* model, so it approximates the real system by modeling only specific moments.

Those three statements are run in order, from left to right, first displaying the number of bacteria, then modeling the doubling of the bacterial population, and then advancing to the next hour. By the way, it may have occurred to you that the last two statements could be written in either order, or even run at the same time—they are independent, so the ordering would not matter.

After all three statements are run, your display will contain the number 1, `N` will be `2`, and `t` will be `1`. The computer next examines the code inside the parentheses associated with the keyword `while` to see if the three statements inside the braces should be run again. That condition specifies that as as long as the time `t` is less than or equal to 120, the three statements must be repeated. At this point, `t` is equal to `1`, which certainly is less than 120. Therefore the three statements will be run again.

This is called a "loop," and now the computer will begin the second time around the loop, running the three statements again as they ran before, but now with altered values

of `N` and `t`. First it will display `N`, which is now equal to 2, so it will display 2. Then it will double `N` again, changing it from 2 bacteria to 4, and increase the time `t` by 1, from hour 1 to hour 2.

Thus the process repeats. Examining the condition inside the parentheses, the computer finds that 2 is less than or equal to 120, and so the three statements inside braces are run again. This goes on and on until `t` is greater than 120, at which time the loop is finished. At the end, `t` will be 121 and `N` will be whatever number has been reached by the process of doubling.

This code illustrates two fundamental aspects of computer coding: "condition testing" and "looping." In larger programs loops are "nested" within other loops and condition tests are nested correspondingly. But this two-line program, with the first line initializing and the second line running a loop, is sufficient for our first model. You will soon see that this is not a trivial model, but one that demonstrates an inviolable law of biology, which Darwin put directly to use in creating his theory of evolution.

3.4 Program results

Here is what the program produces, shortened to fit on a page of this book.

$$
\begin{array}{r}
1 \\
2 \\
4 \\
8 \\
16 \\
32 \\
\vdots \\
6.64614 \times 10^{35} \\
1.32923 \times 10^{36}
\end{array}
$$

If you run this program in $\mathcal{R}$ or another suitable language, you should see something essentially identical to the above. Between Monday and Friday, 120 bacterial doublings would produce over 10^{36} bacteria—that's 1 followed by 36 zeros.

That is the computational result. The scientific question is how many individuals this amounts to. Worked out exactly, it is this number: $2^{120} = 1,329,227,995,784,915,872,903,807,060,280,344,576$. To understand the size of this number, suppose the bacteria are roughly cubical 1 μm on a side—one millionth of a meter, or about four hundred-thousandths of an inch (a suitable order-of-magnitude for a bacterium). What volume will the colony occupy in cubic meters at the end of the work week, after five full days of growing unchecked? You might want to speculate: will it fill the culture plate, overflow onto the lab bench, fill the lab, or what?

Work it out and you will see that the answer is 2^{120} bacteria times 10^{-18} cubic meters per bacterium equals about 1.3×10^{18} cubic meters total. How large is that? Estimate the ocean to be a film averaging 3.7 kilometers deep and coating two-thirds of a sphere with a 6400 kilometer radius (this approximates the amount of the earth's surface that is covered by ocean). This is about 1.3×10^{18} cubic meters! At the end of five days, the colony unchecked would thus fill all oceans of the earth with a dense microbial mass, from the greatest depths up to the surface!

This result has deep-reaching implications. First, even though this bacterial model can be quite accurate for a day or so, it fails completely over the course of a week. All models are approximations to reality, at best applicable over a suitable range. Second, there are lessons in its failure. It illustrates one of the inviolable laws of biology—that no population growth can remain unlimited for long. And third, in a mind like Charles Darwin's, and coupled with other biological principles, it leads to the conclusion that organisms must evolve. That is the story of Darwin's elephants.

3.5 Darwin's elephants

With elephants recognized as the slowest breeders of all known animals, Darwin made a laborious calculation, similar to the bacterial calculation above but more detailed, assuming that elephants started breeding at age 30 and continued until age 90, producing 6 young in that time.

Figure 3.1. Elephants and Kilimanjaro.

Of course he had no computers, nor calculators, and apparently kept track of 90 or more age classes and made his calculations (which have never been found) on paper. He calculated by hand on paper and alas those notes have never been found. But he said it cost him "some pain" to reach the conclusion that at the end of the fifth century, fifteen million elephants would be walking the earth, descended from one original pair. From this, he concluded that unlimited growth is impossible.

> There is no exception to the rule that every organic being naturally increases at so high a rate that, if not destroyed, the earth would soon be covered by the progeny. —*Charles Darwin, 1859*

That he explained in Chapter Three of his *Origin of Species*. After explaining results of selection by people in the breeding of domestic animals, he introduced the concept of selection by natural causes in the wild, which he called "natural selection." The simplest model of unlimited population growth was thus useful in the extreme, leading to an inviolable law of biology and the theory of evolution as one of its consequences. Individuals with qualities that allow them to suffer lower mortality or to reproduce slightly faster, and who pass those qualities to their offspring, will be the ones whose qualities predominate.

Figure 3.2. *Charles Darwin was in his twenties when he realized that natural selection was a cause of evolution and started to formulate his theory.*

Chapter 4
Modeling a single population

4.1 Density-independent growth

Density, in the sense of population density, refers to how many individuals are present on average per unit area. One could say, "The density of elk in Yellowstone National Park during the summer is about 3 to 6 per square mile." Sometimes, however, you will see density used as the total number in a place. You may see, "The density of elk in Yellowstone National Park during the summer is about 10 to 20 thousand." The symbol N is often used for population density.

In the first case above, you would write $N = 4.5$, the midpoint between 3 and 6. In the second case you would write $N = 15000$. It should be clear from context what the area is.

With this in mind, all of the following statements are equivalent:

1. The population doubles each hour. (As in the bacterial example of the previous chapter.)
2. The population $N(t)$ doubles each hour.

   Here the number of individuals is represented by the letter N, and $N(t)$ means population at time t. In the bacterial example, there was one bacterium at the beginning of the experiment when the clock started running,

so you would write $N(0) = 1$. One hour later, the hypothetical population had doubled, so you would write $N(1) = 2$. Doubling successively then gives $N(2) = 4$, $N(3) = 8$, and so forth until after five days, or 120 hours, $N(120) = 10^{36}$, or slightly more—enough to fill all the oceans of the world.

3. The population N doubles each hour.

   Often the "(t)" is left off for simplicity, it being understood that the population N is a function of time.

4. N doubles each hour.

   Since N represents a population in this case, the word "population" will often be dropped for conciseness.

5. $N(t+1) = 2\,N(t)$.

   In English this would be read "N of t plus 1 equals two times N of t." That simply means that the population at some time, anytime, t, when multiplied by 2, is the population in the next time step, t plus one.

6. The change in the population each hour is equal to the size of the population that hour.

   This may sound pretty confusing. But it means that the amount that the population increases in the time step is equal in size to the whole population. Usually the increase is much less than that, perhaps a few percent, but here we are dealing with a rapidly increasing bacterial population.

7. The change in the population each hour is $N(t+1)$ minus $N(t)$, which is to say $N(t+1) - N(t) = 2\,N(t) - N(t) = N(t)$.

Here the population in the next hour, $N(t+1)$, minus the population now, $N(t)$, which is the change in population, is twice the current population, $2N(t)$, minus the current population, $N(t)$ (not less confusing, perhaps).

8. The change in the population each hour, call it "Delta N" or ΔN, is $\Delta N/\Delta t = 2\,N(t) - N(t) = N(t)$.

   Here the symbol delta (Δ) means *change in* or the *difference*. So ΔN means the change in N, and Δt means the change in t. So the change in N per unit of time is written $\Delta N/\Delta t$, where delta t is the time unit being used, such as hour or day. This statement is thus the same as the previous one, but with symbols to shorten it.

9. $\Delta N/\Delta t = N$.

   This means the population change in each time unit is equal to the population size itself. That is just because the population is doubling each time step.

10. $$\frac{1}{N}\frac{\Delta N}{\Delta t} = 1.$$

    This is just dividing both sides of the previous equation by N, and perhaps looks even more confusing. However, in what follows, it turns out to be the most useful of all.

To move forward, let's focus on the last equation, with its parts colored in the box below.

$\frac{1}{N}\frac{\Delta N}{\Delta t} = 1$	There is one new individual ...
$\frac{1}{N}\frac{\Delta N}{\Delta t} = 1$	every hour ...
$\frac{1}{N}\frac{\Delta N}{\Delta t} = 1$	for every member of the population.

In the first row, the "$\Delta N = 1$" refers to a change in the population of one individual, because delta (Δ) means change. In the second row, the "Δt" in the denominator modifies this to the change in each time step—in this, case each hour. In the third row, the $1/N$ modifies it drastically to mean the change in the population *per individual* in the population.

This could mean that one new individual is born while the parent lives on, or that two new individuals are born and the parent dies, or that the parent divides in two, or other equivalent events. In this model, these details are abstractions that do not matter for purposes of projecting the population. The model simply records the number of offspring produced by each member of the population and surviving to reproduce. Multiplied by 100, this becomes the percentage growth of the population. For humans, this is like the number of children

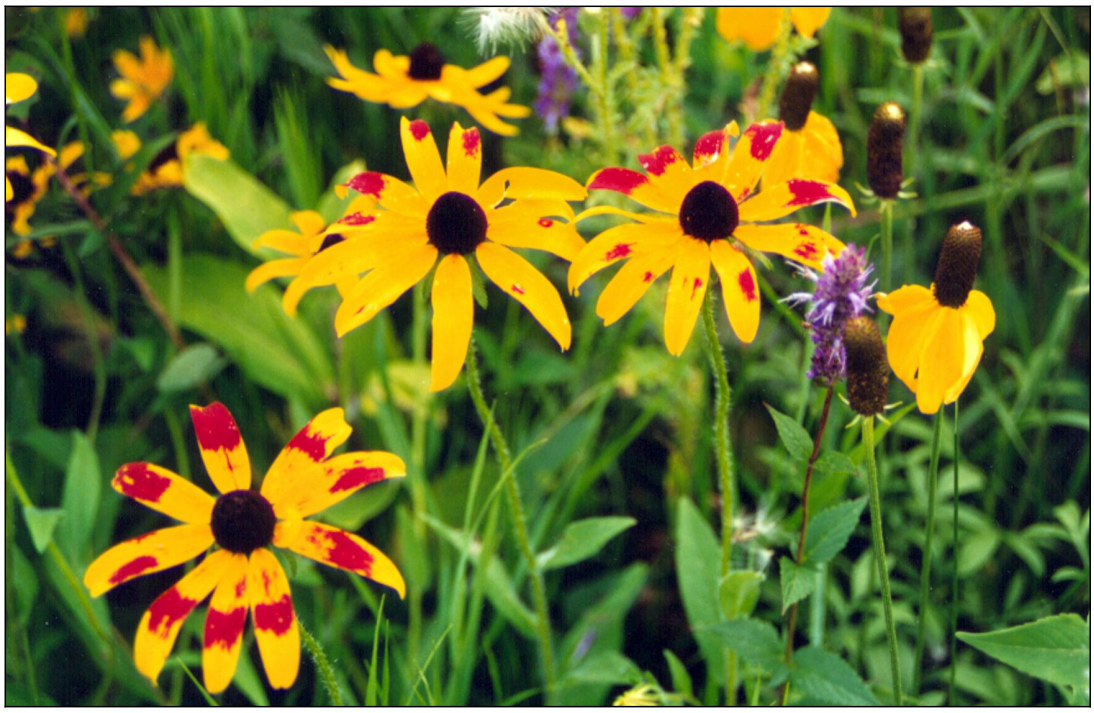

Figure 4.1. *Wild Rudbeckia hirta with an unusual stroke of red boldly bisecting its petals.*

per family who survive to adulthood. (Though it has to be divided by two if there are two parents per family.)

You have seen how rapidly that blows up, from the calculation on page 25.