Logistic Growth of a Population App

Jacob S. Townson

April 5, 2016

Abstract

The classroom is becoming more of a technological advanced place with each passing year. It is becoming increasingly more common to use apps and other electronic forms of learning to be present in a class. In order to try to see this in action, our group has made an app for an Evolution and Ecology course at our college. This app was made using different probability models and theoretical differential equations in a Shiny App made in R Studio. The app is used to show/simulate a population's growth when there is a specific carrying capacity, birth rate, and death rate. The app's purpose is to help teach the biology class this lesson of a population with a carrying capacity through the use of dynamic graphs and an interesting yet easy to use interface. The app has been successful in the implementation behind it, as it is fully functional and ready for the classroom. To test it's use in a class, we have given a survey to two separate classes to test their knowledge of the subject. One class used the app and the other did not. *Insert results here*

Contents

1 Introduction

1.1 Question

Over the summer of 2015, I worked in a research group on Georgetown College's campus with Dr. Homer White and fellow student Andrew Giles. Originally, we were working on learning about Data Science, and began taking classes in a Data Science Specialization Course online. After Andrew and I learned some of the material, Dr. White recommended we work on Shiny apps, and make a successful Shiny app by the end of the summer. We were approached by a biology professor, Dr. Timothy Griffith, with a project to make a shiny app for his Evolution and Ecology class at Georgetown. We chose to do this project and made a functional and easy to use app for the class. So the question arose, how well will this app work in the classroom setting with biology students? In other words, does using an app rather than learning the full and heavy math behind the model help students more, or does it mask too much of the details?

As mentioned, our app works with the math behind an evolution and ecology course. The app is used to show/simulate a population's growth when there is a specific carrying capacity, birth rate, and death rate. Below I will explain the design and methods we used to make the app. To see our code, and more details on how we did our work, feel free to visit our GitHub repository for this project at `https://github.com/obewanjacobi/shinyBio/tree/master`. Or to see the app itself, visit `https://obewanjacobi.shinyapps.io/logistic`.

1.2 Significance

Using technology in the classroom has been an up and coming subject in the past few years. Schools all around the US and even out of the country have begun bringing in new tech to try and help enhance the learning experience to make it easier for students. So, the reason that my question matters is because if apps like these are easy to make, if they are helpful, then we can start integrating them into classes more regularly. Students wouldn't be required to try and understand the gory details of a subject that don't apply to them, unless they really want to. The teacher can easily regulate what is shown at one certain time versus another. When asked, future teachers prefer having apps because it helps to get all of the students involved in the learning process, as well as organize thoughts to help manage the class and how they learn.

Studies show especially for the subject of evolution and ecology (the course that this app was made for), because understanding the theory behind many things requires the conceptualization of some fairly difficult mathematics, biology students have difficulty learning it. This is referenced in the article *Creative Education* (2011) when they say "...population dynamics is a complex branch of population ecology that has an essentially quantitative nature. The effective assimilation of this topic should consider basic aspects of population theory, which involves the conceptual understanding of mathematical models." People are looking for new ways to teach this subject, and many others as well. Using

an app like this one could make a huge difference to students everywhere. Other studies, like Scott McDaniel's article in *EScholarship*, show that the entrance of apps and applets into the classroom have actually increased results on tests. So making these apps more integrated into the way we teach could actually help students learn more easily.

1.3 Population Biology Overview

Population biology is a branch of biology involving evolution, ecology, and behavioral biology. It is also heavily reliant on some knowledge of mathematical modelling. Unfortunately being on the more mathematical sides of things, it is difficult for myself to define population biology in such a way that I convey it correctly. Luckily, Don Alstad (an actual biologist) puts it quite well in his book *Basic Populus Models of Ecology* by describing: "Population biology is a quantitative science dealing with changes in the size and composition of populations, and population biologists often use mathematical models to infer population dynamics." Because students aren't necessarily population biologists yet, this is where my app development group comes into play to help with the education of these students so that they can see the importance of mathematical modelling in population biology.

2 Methods

2.1 Tools

Before going into the design and models used in the app, one must first understand what R and RStudio are. R is a coding language and environment, mostly used for statistical purposes. I won't go into the entire history of how the language was made, but it is based on of the S language. When I say that it's a language and environment, I mean that it is a language in that R is a type of coding language, and it's an environment because it is a system of tools rather than a group of seemingly unrelated tools as is the case with some other languages. It has a large range of graphical, and many different kinds of statistical abilities, thus why it is extremely useful for this project.

RStudio is the IDE, or integrated development environment for R. What that means is that RStudio can be used to make R code and run it, while also helping along the way by attempting to help correct errors when they are made. This is where our group did all of our coding with R to make the app.

The other tools that were essential to our project were git and GitHub. Git is a version control system, which means that it is a method used for keeping track of the versions made of software. We used git to handle when we would make changes to our app overtime. Having version control is important so that if something went wrong, we could trace it back to a certain version that we had made.

GitHub is used for similar things, however it has a nicer user interface, and

is on the internet for everyone to see. So when we would make an updated version of the app and commit it using git, we would then "push" our changes to GitHub so that the rest of the group could get the newest version of the app. This helped so that the entire group was on the same page of where the app was in development. If you would like to see the GitHub repository with all of our work, please visit `https://github.com/obewanjacobi/shinyBio/tree/master`. Feel free to make an account as well, that way you can comment on any of our work by using the "Issues" button on GitHub. This helps us so that we can make any necessary changes as soon as possible.

2.2 Mathematical Modeling

Before one can understand the methods and reasoning behind each step of making this app, one must first understand where the math comes from that the app is based on. Our model relies on the use of differential equations and their solutions for the theoretical portion of the app, and probability and statistical models for the simulations made and run in the app. One last thing to understand before I delve into how we made the models used in the app is the notation and variables that we used for the models themselves. Let us establish the following notation:

- n_0 = initial population
- $n(t)$ = population at time t, sometimes abbreviated as n
- b = unconstrained birth rate (or the maximum birth rate)
- d = unconstrained death rate (or the minimum death rate)
- m = carrying capacity (max population that an environment can safely sustain)
- b_t = per capita birthrate at time t. Defined as $b_t = \frac{E(B_t)}{n}$
- d_t = per capita deathrate at time t. Defined as $d_t = d + max(0, b(\frac{n}{m} - 1))$

2.2.1 Differential Equations

This section will explain how we found the differential equation that was used in the theoretical calculation of the population and its growth. We will begin with what we know, and expand from there using our previous definitions given.

We know that the rate of population growth is going to be the the number of rabbits times the birth rate at that specific time minus the death rate at that specific time. To put this in terms of math, we would write it as $\frac{dn}{dt} = nb_t - d_t$ which simplifies to be

$$\frac{dn}{dt} = n\left(\frac{b-d}{m}\right)(m-n) \tag{1}$$

by our model. To solve this theoretical equation, we must use knowledge of differential equations. From here we will split into 3 cases, when $n_0 = m$, $n_0 > m$, and $n_0 < m$.

To start, take note of the almost trivial answer when $n_0 = m$. Here, I propose that based off of our differential equation we made, we can find that $\frac{dn}{dt} = 0$. While proving this using differential equations and algebra is difficult and hard to follow mathematically, let's look at it from a logical point of view. If a population starts off at it's carrying capacity, it would only make sense that it would not grow anymore or drop any lower than that value. If we solve the differential equation using the initial value of $n_0 = m$, we find that the equation simplifies in such a way that in fact for any time value, $n = m$. This tells us that the theoretical equation is a constant, which makes sense considering that if a population starts off at it's carrying capacity, theoretically it wouldn't grow (even though this isn't necessarily true, which we make note of with our simulation).

Next, we must look at our other two cases, $n_0 > m$ and $n_0 < m$. For each we want to move all of the n values to one side of the equation to make it easy to integrate. This leads us to get

$$\int \frac{dn}{n(m-n)} = \int \frac{b-d}{m} dt = \frac{b-d}{m} t + C \tag{2}$$

where C is some constant that we will find later. So now we want to integrate the left side of the equation. In order to do this, we will need to use partial fractions to simplify it to make it easy to integrate. After doing this, we get that

$$\int \left(\frac{1}{mn} + \frac{1}{m^2 - mn} \right) dn = \frac{b-d}{m} t + C \tag{3}$$

When we integrate this simpler integral and apply some natural log rules, we get that

$$\frac{1}{m} \left(\ln \left| \frac{n}{m-n} \right| \right) = \frac{b-d}{m} t + C \tag{4}$$

Note the absolute value in the above equation. This is why we need two cases for this one integral. For the case of $m > n_0$, the value inside the absolute value is unchanged. From here, we want to find what the C value is. In order to do this, let $t = 0$ so $n = n_0$. After solving equation (4) for C, we get that $C = \frac{1}{m}(ln(\frac{n_0}{m-n_0}))$. This gives us that

$$\frac{1}{m} \ln \left(\frac{n}{m-n} \right) = \frac{b-d}{m} t + \frac{1}{m} \ln \left(\frac{n_0}{m-n_0} \right) \tag{5}$$

Now we can simply solve the equation for n. This mostly just involves some elementary algebra, which leads us to get that

$$n = \frac{m}{1 + \left(\frac{m-n_0}{n_0}\right) \mathrm{e}^{(d-b)t}} \tag{6}$$

if $m > n_0$. This is our equation for the theoretical model. However as mentioned, we must realize that you cannot take the natural log of a negative number hence our third case of $n_0 > m$. To account for this we must go back to equation (4). Here, we notice that because $n_0 > m$ we get a negative value inside the natural log giving us the new equation:

$$\frac{1}{m}\left(\ln\left(\frac{n}{n-m}\right)\right) = \frac{b-d}{m}t + C \tag{7}$$

Here we have taken the absolute value of the number inside the natural log. Next we can go through the steps as we did in the second case. After doing these algebra steps, we find that

$$n = \frac{-m}{\frac{n_0-m}{n_0}\mathrm{e}^{(d-b)t} - 1} \tag{8}$$

if $n_0 > m$. This is our model for the theoretical graph.

In order to be sure of our deterministic model, we wanted to consult a biology text to see if there was anything similar done before. After some research, our group found that this model follows the theoretical equation for other biologists as well. Their equation was

$$N(t) = \frac{K}{1 + (\frac{K-N(0)}{N(0)})\mathrm{e}^{-rt}} \tag{9}$$

found by Roughgarden(1979), Emlen(1984), and Neuhauser(2000). This relates to our equation because $N(t) = n$, $K = m$, $N(0) = n_0$, and $r = b - d$.

2.2.2 Probability Model

In order to run simulations to convey to the students that the deterministic model isn't always true, we need a probability model for the simulation to follow. This is where the variables defined before as being reliant on time come into play. They are reliant on time is because at each given time interval, the app will run a simulation based on what the previous time values were. We will get more into the programming of this later, but for now, here is the math behind our model.

To begin, let's define some variables that will be needed to create our model:

- L_t = number of litters at time t
- S_t^i = size of the i^{th} litter born at time t. These are independent of each other and of L_t
- B_t = number of births at time t

These are random variables that are dependent on time. Random variables are variables that are dependent on chance factors. To put this in perspective of the app, the number of baby rabbits born to a litter won't always be the same. Rather, the number of babies is dependent on multiple chance factors such as health of the rabbits, rabbit genetics, etc. To represent this chance variation, we will use a Poisson distribution to describe the variables here and give us our simulated data. A Poisson distribution is convenient for this situation because it is used to express the probability of a given number of events happening in a certain interval of time. So for us, this is used to represent the number of rabbits born to a litter at time t, and the number of litters born at time t.

Next, one must understand the concept of an expected value. An expected value is an average value of a variable that you would expect to happen if you gathered an infinite amount of data for the subject, and is found in different ways depending on the distribution used to describe that variable. In our case, we are using a Poisson distribution to describe the population of rabbits. Luckily, RStudio can calculate this easily for us. For now, we will simply represent the expected value of a random variable as $E(X)$ where X is the random variable.

Now we can proceed with the explanation of the model, so let's make two more variables just to help set up the overall calculations: let

$$s_t = E(S_t^i)$$

and

$$l_t = \frac{E(L_t)}{n} \tag{10}$$

We know that the number of births at time t will be the sum of the sizes of all of the litters, so

$$B_t = \sum_{i=1}^{L_t} S_t^i \tag{11}$$

We call the per capita birth rate $b_t = \frac{E(B_t)}{n}$, as we defined before. We would like to show then that the expected number of births is the expected number of litters times the expected size of the litters, or $E(B_t) = E(S_t^i) \times E(L_t)$. To do this, we must use properties of expected values. First take the expected value of B_t, giving us

$$E(B_t) = E(\sum_{i=1}^{L_t} S_t^i) \tag{12}$$

From this using conditional expected value properties we see that

$$E(B_t) = E(E(\sum_{i=1}^{L_t} S_t^i | L_t)) \tag{13}$$

In words, equation (13) means we want the expected value of the expected value of the sum given L_t. We can assume that S_t is independent of L_t here because

the litter sizes do not depend on the number of litters birthed in a population. To explain a real life situation, to say that these things are dependent is equivalent to saying that if a rabbit is near or witnesses another rabbit having babies, it will control or naturally have a different amount of births because of this, which isn't true. Thus we can make our assumption. Now using properties of expected values and independent conditional circumstances, we then simplify such that $E(B_t) = E(\sum_{i=1}^{L_t} s_t)$. The sum can now be changed such that $E(L_t \times s_t)$, and because s_t is a constant here, we know that

$$E(B_t) = s_t \times E(L_t) \tag{14}$$

When (14) is simplified based on our definitions we can write it as

$$b_t = s_t \times l_t \tag{15}$$

Which completes our proof.

The s_t and the l_t are what we simulate using the rpois() function in R, which as previously stated, we will address more later. Our group needed an expected average litter size, so we simply googled the average rabbit litter sizes for this value. The number born to each litter isn't entirely important to our model because the model still accounts for carrying capacity and theoretical growth rather than the litters affecting the outcome. We found that the average litter sizes for rabbits was 8, so we let the

$$s_t = \frac{8\sqrt{b_t}}{\sqrt{b}} \tag{16}$$

and

$$l_t = \frac{\sqrt{b_t}}{8} \times \sqrt{b} \tag{17}$$

in order to be able to calculate the the number of births at time t for the simulated numbers. Then to calculate the number of deaths for each time interval for the simulation, we have our d_t function that we defined previously that depends on how close the population is to the carrying capacity,

$$d_t = d + max(0, b(\frac{n}{m} - 1)) \tag{18}$$

By simply subtracting the values of d_t from b_t we obtain our simulated data.

2.3 App Design - The User Interface

Before going deep into the app design on the user interface (or ui for short), I invite the reader to experience the app for themselves so that the design process makes more sense. The app can be found at `https://obewanjacobi.shinyapps.io/logistic`.

2.3.1 Sidebar

This section of the app is where the students input their values for the graphs and simulations. It is always on the left side of the app, and is dynamic such that when the user changes any input on a slider, the graph will change depending on what they have chosen. The only case where this isn't true is if the user has hit the simulate button, in which case the sidebar simplifies accordingly. The descriptions of the input are described below.

- **Time**: This lets you choose the time value displayed on the x axis of the graph under the *Population Size* tab.
- **Initial Population**: This lets you choose the initial population, or the y intercept under the *Population Size* tab.
- **Birth Rate**: This lets you choose the maximum birth rate for the population. The effective birth rate will change according to how close the population gets to the carrying capacity.
- **Death Rate**: This lets you choose the minimum death rate for the population. The effective death rate will change according to how close the population gets to the carrying capacity.
- **Carrying Capacity**: This lets you choose the carrying capacity for a population. It could be thought of as the amount of resources a population has to survive/thrive. Note, carrying capacity option is removed if death rate is above birth rate because a carrying capacity doesn't affect this given situation.
- **Setting the Seed**: This check box gives the user the option to set the seed of the simulated values. By setting the seed, the user can get the same outcomes for every time the *Simulate* button is pushed while the seed is set to that specific value. When the box is unchecked, the option goes away and the simulation is random again.
- **Simulate**: This button runs the app with a new simulation with the given input parameters.

2.3.2 Population Graphs

This section is the main portion of the application. This is the only tab available before the **Simulate** button is hit. This tab displays the graphs used to convey information on the population growth. Before the **Simulate** button is hit, the graph displays only the theoretical results. These theoretical results are what would happen "on average". This means that if we can imagine an infinite amount of populations were observed, the average would look like our theoretical results. If the *Size* button is chosen, then the population size graph will be shown. This graph shows the growth of a population over an arbitrary amount of time as it approaches its carrying capacity. Here the population is conveyed

by the red line and the carrying capacity is shown in blue. The next option under this tab is to plot the growth rate. Here the growth rate is plotted against either time or the population itself. This switching between what is plotted on the x axis can help to teach students how the growth rate changes, and makes it more easily understood through a look at it from both perspectives. Similar to this structure, the next option under this tab is to plot the per capita growth rate. Again, the user can choose what to put on the x axis, either time or population to give the user an easier way of understanding depending on their perspective.

The only change that is made under this tab by hitting the **Simulate** button is that a simulated population is displayed on the *Size* graph. This is conveyed by a black line. This simulated line is made using our model and with the help of the rpois() function in R. So this creates a possible outcome for a single population to show what it could look like over a certain given time. This is not the same thing as the theoretical graph, as the theoretical graph shows the "average" situation, whereas the simulated shows a possible situation for one group made by our probability model.

2.3.3 Field

This portion of the application displays a model for the field, which is a hypothetical situation for the simulated data shown in the *Population Graphs* tab. Here there are two plots next to each other, with an animation bar above them. The two plots are used to show the total population at the time that is shown in the animation slider. When the play button is hit below the slider, the slider will move from time value to the next time value for every second or so. Whatever the simulated population is at that time, that is the total number of dots on the plots. The plot on the left shows the adults in the field foraging for food for their children.

The plot on the right shows the newborn rabbits in what is cleverly called the Warren. Here, the litters are grouped closely together using a plotting system we made involving keeping litters close together using the rnorm() function. In the field function, you may also notice the color of the graph changing over time. If there is a carrying capacity used in the simulation, then the color of the graph will change. The idea is that if the rabbits are far away from the carrying capacity, then the field will be more green. On the other hand, if the population is above, close to, or on the carrying capacity, they will naturally be running out of resources, making the field a more brown and barren color. This makes the app more interactive, and maybe fun for the students to learn the concepts, and actually get the feeling that they are seeing the changes in the environment.

Below both graphs, a table will be displayed if any rabbits were born and added to the Warren. The table will show the number of litters at that time, and also show the average size of the litters born.

This section of the app is important because we were tasked with giving visual representations that could be fun for the students, but also educational. One thing we were assigned is to attempt to make the students learn how to read

graphs better. Because of this, we not only made the different graph settings in the population section, but we also made this section to make the students read a graph in which they can watch the population grow and decrease. This makes the experience more interactive and much more interesting for students.

2.3.4 Graveyard

This portion of the application is similar to the field in that it is a model used to present hypothetical information. It uses the simulated population to show the deaths at each time. It is uses a similar format to the field in that it uses an animation slider to show the graveyard over time.

This tab also gives the rabbits a cause of death, again, to give the user a feeling of the whole situation being more real. The deaths are divided into three categories, death by lawnmower, death by foxes, and death by malnutrition. The death by lawnmowers is more likely to happen when the population is small, and the death by foxes and malnutrition's likelihood go up as the population grows. The app also has an info link to tell the user this explanation. There is also a table to show the population and the carrying capacity at the bottom, just to remind the user to give more of a reference on the amount of deaths at that certain time.

Similarly to how we made the *Field* section of the graph more interactive, we did the same with this section for the same purposes. We want these sections to make the user learn how to read graphs better, and to do so, we have tried to make a fun environment for the student to learn.