BIOU9PC: Population and Community Ecology

Lab Practical week 1, part B: Exponential population growth

***15 September 2016***

**Objective**

The purpose of this practical is for you to interact with simple exponential models of population dynamics using the R language and environment. In so doing, you will

1. Understand the linkages between ecological processes and their representations in models
2. Experience how changing parameter values and starting conditions affect model predictions
3. Understand the consequences of density-independent population growth

**Introduction**

Mathematical models provide an abstraction of nature. They allow us to make quantitative predictions about the behaviour of natural systems without time-consuming experiments. These predictions can then be tested against reality to evaluate the realism of the model, and to determine the conditions under which the model fails to predict observations. These discrepancies are always interesting, because they indicate that unexpected phenomena are occurring.

Exponential growth can be modelled using discrete or continuous time. We begin with discrete time.

**Exponential growth in discrete time**

Nt+1 = R·Nt

If R = 1, then the number of births and deaths is exactly balanced, and the population size in the next time step (Nt+1) is exactly the same as that in the current time-step (N).

If R > 1, then there are more births than deaths, and the population grows.

If R < 1, then there are fewer births than deaths, and the population shrinks.

This equation projects the population size one timestep (often one year) into the future., which we can do like this:

N <- numeric(10) # A vector to hold the population size predictions.

N[1] <- 100 # the initial population size (N0)

r <- 1.1 # the intrinsic growth rate

N[2] <- r\*N[1]

N[2]

The last line prints our prediction for the population size in time-step 2 to the screen. Looking forward to time-step 3, we could write

N[3] <- r\*N[2]

But making projections like this will clearly become boring fast. To project further, it is useful to use a programming structure known as a ‘loop’. Loops tell R to do a certain task as many times as you command. They look like this:

for(t in 1:9){

N[t+1] <- r\*N[t]

}

The first time through this loop, t takes the value 1, as that is the first element in the vector. Inside the loop, r, which we asserted has a value of 1.1, is multiplied by the tth element of the vector N. Because t = 1 (on the first time through the loop), N[t] (i.e., N[1]) is 100. This value (1.1\*100 = 110) is assigned into the t+1th (i.e., the 2nd) element of N to represent the population size in time-step 2. The trick is that on the next time through the loop, t takes the next value in the vector (i.e. 2, 3, 4, 5, 6…). Finally, once t takes the value 9, R sees that there are no more values in the vector, and the loop ends.

With simple structures such as this, you can make predictions about the population size of a population as far into the future as you would like.

**Exponential growth in continuous time**

The discrete-time model is only strictly appropriate for populations with non-overlapping generations, such as annual plants in which all adults are killed by frost. A more general model is the continuous-time version:

Nt=N0ert

Just like in the continuous-time model,

If r = 0, then the number of births and deaths is exactly balanced, and the population size in the next time step (Nt+1) is exactly the same as that in the current timestep (N).

If r > 0, then there are more births than deaths, and the population grows.

If r < 0, then there are fewer births than deaths, and the population shrinks.

We implement this model in R using a function. The use of functions is a technique that we will use widely in this module. It forms the foundation of programming in R and other computer languages, so it is worth becoming familiar with it. Functions encapsulate code that will be used repeatedly, isolating them from the user, so that he or she does not need to interact with their details.

This is a function definition:

pred.size <- function(N0, r, t){

Nt <- N0\*exp(r\*t)

return(Nt)

}

The first line of code provides the name of the function (‘pred.size’), and tells R what arguments (also referred to as ‘parameters’) need to be provided as input to the function. Here, there are three, just like in the discrete-time model. Inside the curly brackets is the body of the function. The second line makes the prediction of the future population size. The final line returns the output back to the user. Functions only need to be defined once. Afterwards, they may be called (i.e., run) as many times as necessary by the user.

This is how the function ‘pred.size’ is called:

pred.size(N0 = initial.pop.size, r = intrinsic.growth.rate, t = time)

Note that we do not need to name the arguments, AS LONG AS they are provided in the order expected by R. Thus, this provides the exact same output:

pred.size(initial.pop.size, intrinsic.growth.rate, time)

Predictions for multiple points in time can be generated by providing a vector of times to pred.size().

**Approach**

Opening the script “practical 1.r” in Rstudio. Your task is to manipulate the discrete and continuous time exponential models and make observations on the resulting dynamics. In lab, I encourage you to work with your neighbours. The practical report (see below) is to be completed individually.

**Assignment**

Answer the following questions using your own words. Some, but not all, require figures to illustrate your answers. Answers should be concise and accurate. Do not just write everything you know. Though you may work with your classmates on this lab, each student is to write up their report independently. Turn in this assignment using Turnitin by **Monday September 19 at noon.** Indicate your identity only with your student number.

**Use the discrete-time model for questions 1-6.**

1. Take a population with initial population size 2 and intrinsic population growth rate 1.05.
   1. What is the predicted population size after 100 time steps?
   2. What is the effect on the final population size of doubling the initial population size to 4 individuals?
   3. What is the effect on the final population size of doubling intrinsic population growth rate to 1.10?
2. Make a graph with labelled axes (hint: use plot()) that shows the dynamics of the population as initially described in Question 1.
   1. Add points to your graph to show the dynamics of the population with the initial size of 4 individuals (hint: use points()).
   2. Add points to your graph to show the dynamics of the population with the intrinsic population growth rate of 1.10 (hint: use col = ‘red’ as an argument to points()).
   3. Make a new version of your graph with the y-axis log-transformed, and explain the resulting graph to a nearby student (hint: use log=‘y’ as an argument to plot()).
3. African bush elephants (*Loxodonta africana*), have an average individual mass of 2700 kg. There are currently approximately 500,000 African elephants alive in the wild. If their population increased exponentially, with an intrinsic annual population growth rate of 1.1, in how many years will the total mass of the elephant population exceed that of Earth (5.972×10^24 kg)?
4. A single *Escherichia coli* has a mass of 1×10−15 kg. If their population increased exponentially, with an intrinsic ***daily*** population growth rate of 58.7 and an initial population size of 10, in how many days will the total mass of the *E. coli* population exceed that of earth?
5. Using the parameters given in Question 3, what is the doubling time (in years) for African elephants?
6. Using the parameters given in Question 4, what is the doubling time (in days) for *E coli*?
7. Repeat questions 1-6 using the continuous-time model.
8. Note that both models can generate fractional predictions of population size. Why is this unrealistic? How could the models be modified to avoid this error?
9. Write a function to encapsulate the essence of the discrete-time exponential model, following the function definition used for the continuous-time model.