ESM 211 Homework #3

Laura Ingulsrud

2/18/2020

Overview

We will analyze a loggerhead sea turtle population model. It appears in Table 2 of Crowder et al. (1994), which is on the GauchoSpace page. We will use it (as did the authors) to address the questions:

- Is a reduction of fisheries bycatch needed to arrest the decline in the sea turtle population?
- If so, by how much?

We will work through the exercises in class. For homework 3, please answer the questions at the end of each section, and turn them in via email by the homework deadline.

1 Enter the model

Demographic data come from many places, and take many forms. A few types of data are easy to analyze (one such is when you have observations of the stage, aliveness, and fecundity of each individual every year; section 2.3 of Stevens talks about how to analyze such data). However, in getting data from the literature for your project, we expect that in most cases you will be getting synthesized information on vital rates, perhaps already formed into a projection matrix.

We will look at turning vital rates into matrix models in a later class; but for now, let's assume that you have the matrix. How do you get it into R?

The easiest way is with the `matrix` function.

```
# Crate matrix for 5 class stages of Loggerhead turtle population
class_names <- c("Egg", "Sm Juv", "Lg Juv", "Subadult", "Adult")
A <- matrix(c(0,     0,     0,     4.665, 61.896,
              0.675, 0.703, 0,     0,      0,
              0,     0.047, 0.657, 0,      0,
              0,     0,     0.019, 0.682,  0,
              0,     0,     0,     0.061,  0.809),
            nrow = 5, ncol = 5, byrow = TRUE, dimnames = list(class_names, class_names))
```

- The first argument to `matrix` is the values to put into the matrix, given as a vector. Although the way I spaced the values makes it look like a matrix, it is actually a single set of numbers inside the `c()` function; I could have gotten the same result by putting all 25 values on a single line. However, the extra spacing to line things up in columns is useful: it makes it easier to ensure that you've got everything in the right place. For example, I initially accidentally typed in 0.19 for $a_{4,3}$; it would have been really hard to check that against the published matrix without having things lined up.
- The remainng arguments set the number of rows and columns, and whether we want to fill in the data row by row (as here) or column by column (which would use `byrow = FALSE`).
- The last argument gives the rows and columns names. This makes the matrix take up more room to print, but it is easier to remember what the stages are! You can also add row and column names to an existing matrix using the `rownames` and `colnames` functions (e.g., `rownames(A) <- class_names`).

1.1 Matrix conventions

- Matrices have two dimensions. They are indexed first by **R**ow, then by **C**olumn. This is a universal convention, but as it is arbitrary, it can be hard to remember. The mnemonic I use to remember this is "RC Cola", which was heavily advertised on TV in my childhood with the slogan "Me and my RC," and a catchy tune.
- In mathematical notation, a matrix is conventionally named with a capital letter, and displayed bold face: **A** (vectors are also bold; to distinguish them, use lower case, which is why we use **n** instead of **N**). Each element of a matrix is denoted by the lower case equivalent, with subscipts denoting its position: thus the element in the 4th row and 3rd column of **A** would be written $a_{4,3}$.
- To extract a single value from a matrix in R, use the square brackets, just as you would to subset a vector, but specify both the row and column. Thus `A[4, 3]` returns the element in the 4th row and 3rd column.
- If you leave one of the indices blank (but don't forget the comma!) you will get an entire row or column: `A[ , 3]` gives the 3rd column, and `A[4, ]` gives the 4th row.
- You can also specify ranges, multiple values, or exclusions for either index. Thus, `A[3:4, 3]` gives elements (3,3) and (4,3), `A[ , c(3,5)]` gives columns 3 and 5, and `A[-1, ]` gives everything except the first row.

1.2 Exercises

1.1. Print the matrix you have just created, and ensure that it matches the one in Table 2 of Crowder et al. (1994) (linked on the GauchoSpace page)

```
print(A)
```

1.2. Print out the subsets of `A` described in the list above. Do you get the values you expect? Do you understand how matrix subsetting works? If not, what don't you understand?

```
# Print subset for row 4, column 3
print(A[4, 3])

# Print subset for column 3
print(A[ , 3])

# Print subset for row 4
print(A[4, ])

# Print subset for (3,3) and (4,3)
print(A[3:4, 3])

# Print subset for columns 3 and 5
print(A[ , c(3,5)])

# Print subset for everything but the first row
print(A[-1, ])
```

Yes, I got the values I expected after understanding that matrices are indexed first by row, and then by column.

1.3. From the matrix you have just created, draw the life cycle graph, putting in the values for each transition. This can be hand-drawn.

2 Projecting the population matrix

In lecture we saw how to project the matrix model using matrix multiplication (`%*%`) in R. The full set of codes for analyzing the semipalmated sandpiper model are in the script file `SPsandpiper.R` on GauchoSpace. Take a look at this file. If you like, adapt it to iterate the sea turtle model.

However, if you don't want to wrap your head around `for` loops, the **popbio** package provides some functions that simplify the process. For example, to initialize the population with 1000 eggs and 10 individuals in each stage, and simulate for 10 years, you would do:

```
# You may need to install this first with install.packages("popbio")
library(popbio)

# Initial abundance
n_0 <- c(1000, 10, 10, 10, 10)

# Project the matrix - does matrix multiplication (A %*% n_0) over and over again
pop <- pop.projection(A, n_0, iterations = 10)

# Plot each stage through time
stage.vector.plot(pop$stage.vector)
```

By default, this plots the *proportions* in each stage through time. To see the actual abundances, use

```
# Plot each stage through time using actual abundances rather than proportions
stage.vector.plot(pop$stage.vector, proportions = FALSE)
```

2.1 Exercises

2.1. The output of `pop.projection` has a number of other elements besides `stage.vector`. Look at all of the elements of `pop` and make sure that you understand them (referring to the help page if needed).

```
print(pop)
```

The "pop.projection" function in the "popbio" package calculates the population growth rate and stable stage distribution by repeated projections of the equation n(t+1) = An(t). Eventually, structured populations will converge to a stable stage distribution where each new stage vector is changing by the same proportion (lambda). The ouput of "pop" gives 5 elements:

1. "lambda": Estimate of lambda using change between the last two population counts.
2. "stable.stage": Estimate of stable stage distribution using proportions in the last stage vector
3. "stage.vector": A matrix with the number of projected individuals in each stage class
4. "pop.sizes": Total number of projected individuals
5. "pop.changes": Proportional change in population size

2.2. Plot `pop$pop.sizes` and `pop$pop.changes` through time. What do these tell you?

```
# Plot total number of projected individuals
plot(pop$pop.sizes)

# Plot the proportional change in population size
plot(pop$pop.changes)
```

'pop$pop.sizes' tells us the total number of projected loggerhead sea turtle individuals across all the life stages. This plot tells us that the number of sea turtle individuals first increases over time until it hits around 1600 individuals, but then decreases over time down to around 1100 individuals.

'pop$pop.changes' tells us the proportional change in loggerhead sea turtle population size. This plot tells us the proportional change starts out around 1.3, then decreases over time to level off around 0.2.

2.3. Once the population has reached the stable stage distribution (SSD), all stages will grow or decline exponentially with the same growth rate. Looking at the stage vector plot, has this been acheived by the end of your simulated time series? (Tip: this might be easier to determine if you make the plot with abundance on a log scale. You can do this by including `log = "y"` in the call to `stage.vector.plot`)

```
# Plot each stage through time using actual abundances rather than proportions on a log scale
stage.vector.plot(pop$stage.vector, proportions = FALSE, log = "y")
```

No, this has not been achieved, as the subadult and adult stages obviously do not have the same growth rate.

2.4. If the population has not reached the SSD, run the simulation for longer. How many years are required before the population appears to be at the SSD?

```
# Increase # of iterations from 10 to 20 so that the population reaches the SSD
pop_SSD <- pop.projection(A, n_0, iterations = 20)

# Plot each stage through time using actual abundances rather than proportions on a log scale
stage.vector.plot(pop_SSD$stage.vector, proportions = FALSE, log = "y")

# Plot each stage through time using proportions
stage.vector.plot(pop_SSD$stage.vector)
```

Where the lines are parallel, the population has reached the SSD. If you plot the proportions, it's where all the lines are flat. It takes around 20 years before the population appears to reach the SSD.

3 Analyzing the population matrix

Eigenvalues and eigenvectors give a lot of information about the "asymptotic" (once the population reaches the SSD) dynamics and structure. You can calculate them yourself (see the code in `SPsandpiper.R`), but the **popbio** package has two functions to extract the key information:

```
# Lambda is the true asymptotic growth rate
print(lambda(A))

# Corresponding eigenvector expressed as proportion
print(stable.stage(A))
```

3.1 Exercises

3.1. Compare the values of `lambda` and `SSD` with the equivalent outputs of `pop.projection` from the initial run (with only 10 years of simulation). Why are they different?

The values of lambda

lambda(A) is the true asymptotic growth rate of the matrix at that time. stable.stage(A) is the corresponding eigenvector expressed as proportion. Because pop.projection is projected values over 10 iterations or years of simulation.

3.2. You want to improve the status of the population so that it is no longer declining. You think that your best options are to manage the nesting beaches to increase egg/hatchling survival (e.g., controlling poaching, motorized vehicles, dogs, bright lights that disorient hatchlings) or to reduce the bycatch of adult turtles in shrimp trawling nets (e.g., by requiring a modified design with a "turtle excluder device" or by reducing fishing effort). Use the model to evaluate the effects of these two strategies:

a. Which element of the projection matrix represents egg/hatchling survival? Which represents adult survival?

From eggs to small juveniles: 0.675. Could increase this value to represent higher survival of eggs to small juveniles due to decreased beach threats. From adults to adults (adult survival): 0.8091. Could increase this value to represent high survival rate of adults due to decreased bycatch threats.

b. Increase egg/hatchling survival in the model, and re-calculate λ_1. By how much does it increase? Experiment with different values of this survival term until you get an asymptotic growth rate of 1 or more. How large does egg survival need to be to achieve this?

```
# Change egg/hatching survival rate by increasing A[2,1]
A[2,1] <- 1.5

# Check if asymptotic growth rate is now >1
lambda(A)
```

Egg survival would need to be 1.5 to achieve an asymptotic growth rate of larger than 1, which is impossible.

c. Put the egg survival back to its original value, increase adult survival in the model, and re-calculate λ_1. By how much does it increase? Experiment with different values of this survival term until you get an asymptotic growth rate of 1 or more. How large does adult survival need to be to achieve this?

```
# Change egg/hatching survival rate back to its original value
A[2,1] <- 0.675

# Increase adult survival rate
A[5,5] <- 0.93

# Check if asymptotic growth rate is now >1
lambda(A)
```

Adult survival rate needs to be larger than 0.9 in order to achieve an asymptotic growth rate of 1 or more.

d. Based on this analysis, which life stage seems the more promising one to target managment at? What else would you need to know to reach a final conclusion?

As achieving an asymptotic growth rate of greater than 1 is impossible by increasing survival rate of the egg/hatchling stage, this indicates that threats must be mitigated at a different life stage in order to reverse the decline of the loggerhead population. As we achieved an asymptotic growth rate greater than 1 with a feasible survival rate of the adult stage, this life stage seems to be the more promising one to target management at.

4 Sensitivity and elasticity analysis in R

Make sure you have the sea turtle matrix loaded:

```
class_names <- c("Egg", "Sm Juv", "Lg Juv", "Subadult", "Adult")
A <- matrix(c(0,     0,     0,     4.665, 61.896,
              0.675, 0.703, 0,     0,      0,
              0,     0.047, 0.657, 0,      0,
              0,     0,     0.019, 0.682,  0,
              0,     0,     0,     0.061,  0.809),
            nrow = 5, ncol = 5, byrow = TRUE, dimnames = list(class_names, class_names))
```

Like the asymptotic growth rate and the stable stage structure, the sensitivities and elasticities can be calculated from the eigenvalues and eigenvectors of the matrix. The **primer** library (which accompanies the book *A Primer of Ecology in R*) bundles the calculations together to produce the biologically relevant output:

```
library(primer) # You may need to install this first with install.packages("primer")
DemoInfo(A)
```

4.1 Exercises

4.1. Referring to the help page and section 2.2 of the Stevens chapter, make sure you understand what each of the outpurs of `DemoInfo` represents. The "RV" (reproductive value) is the only bit we haven't covered in lecture.

4.2. Looking at the sensitivity and elasticity matrices, what can you conlude about which matrix elements would likely have the biggest impact on λ if they were changed?

4.3. Compare the elasticity matrix with Fig. 1 in Crowder et al. (1994). Do you understand where the values in the figure come from?

4.4. Look at the sensitivity matrix produced by `DemoInfo`. What does the sensitivity for element $a_{5,1}$ represent? Does it make sense to have a non-zero value here? Why or why not?

5 Sensitivity and elasticity of λ to vital rates

The above analyses evaluate the sensitiviities and elasticities of λ to matrix elements. To get the effect for a vital rate v you need to sum over all the matrix elements that v affects, weighted by the strength of the effect:

$$
\begin{aligned}
S_v = \frac{\partial \lambda}{\partial v} &= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial \lambda}{\partial a_{i,j}} \frac{\partial a_{i,j}}{\partial v} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} S_{i,j} \frac{\partial a_{i,j}}{\partial v} \\
E_v = \frac{v}{\lambda}\frac{\partial \lambda}{\partial v} &= \sum_{i=1}^{n}\sum_{j=1}^{n} \left[\frac{a_{i,j}}{\lambda}\frac{\partial \lambda}{\partial a_{i,j}}\right]\left[\frac{v}{a_{i,j}}\frac{\partial a_{i,j}}{\partial v}\right] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} E_{i,j} \frac{v}{a_{i,j}} \frac{\partial a_{i,j}}{\partial v} \\
&= \frac{v}{\lambda}\sum_{i=1}^{n}\sum_{j=1}^{n} S_{i,j} \frac{\partial a_{i,j}}{\partial v}
\end{aligned}
$$

If you have a single, relatively simple matrix then you can do this by hand; but a more robust and scalable solution is available from the **popbio** package where you can write out the matrix symbolically.

For the turtle model, we write the matrix as

```
A.vr <- expression(0,  0,         0,         p4*g4*f,   p5*f,
                   p1, p2*(1-g2), 0,         0,         0,
                   0,  p2*g2,     p3*(1-g3), 0,         0,
                   0,  0,         p3*g3,     p4*(1-g4), 0,
                   0,  0,         0,         p4*g4,     p5)
```

We then set the values of the vital rates. The survivals and fecundity come from Table 1 of Crowder et al. (1994). The growth terms emerged from a complex calculation (described in the paper), but we can reconstruct them from the matrix by noticing that $g_i = a_{i+1,i}/p_i$ (ensure that you understand this). Thus we can do:

```
p <- c(0.6747, 0.75, 0.6758, 0.7425, 0.8091) # Survivals
vr.vals <- list(p1 = p[1], p2 = p[2], p3 = p[3], p4 = p[4], p5 = p[5],
                g2 = A[3,2]/p[2], g3 = A[4,3]/p[3], g4 = A[5,4]/p[4],
                f = 76.5)
```

Finally, apply the `vitalsens()` function:

```
library(popbio)
vitalsens(A.vr, vr.vals)
```

5.1 Exercises

5.1. Compare the elasticities to survival with those plotted in Fig. 2 of Crowder et al. (1994). Which ones seem to be a good match?

5.2. What distinguishes the elasticities that match from those that don't?

> If you are feeling mathematically ambitious, read Appendix 1 of the paper and try to understand what is missing from your analysis

5.3. How would you re-parameterize the matrix so that you can look at the effects of proportional changes in *mortality* (rather than survival), as plotted in Fig. 4 of the paper?