Demographic analysis in R

ESM 211

January 18, 2019

We will analyze the sea turtle population model that was briefly introduced in lecture on Wednesday. It appears in Table 2 of Crowder et al. (1994), which is on the GauchoSpace page.

Please write out the answers to the questions below, and turn them in (email is ok, unless you do the handwritten option) at the end of lab. Depending on your preference, you can create an Rmd file, fill in the word "worksheet" on the GS page, or print out the worksheet and fill it in by hand.

Please answer the questions in each section before moving on to the next section!

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.

```
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)
```

- 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:

```
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,
            nrow = 5, ncol = 5, byrow = TRUE)
```

  (but it runs off the page!). Likewise, the extra spacing to line things up in columns is purely aesthetic (but 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 remaing 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`).

- You can give the rows and columns names, if you like, by using the `dimnames` argument:

```
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))
```

This makes the matrix take up more room to print, but it is easier to remember what the stages are!

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".
- In mathematical notation, a matrix is conventionally named with a capital letter, and displayed bold face: $\mathbf{A}$ (vectors are also bold; to distinguish them, use lower case, which is why we use $\mathbf{n}$ instead of $\mathbf{N}$). 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 $\mathbf{A}$ 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

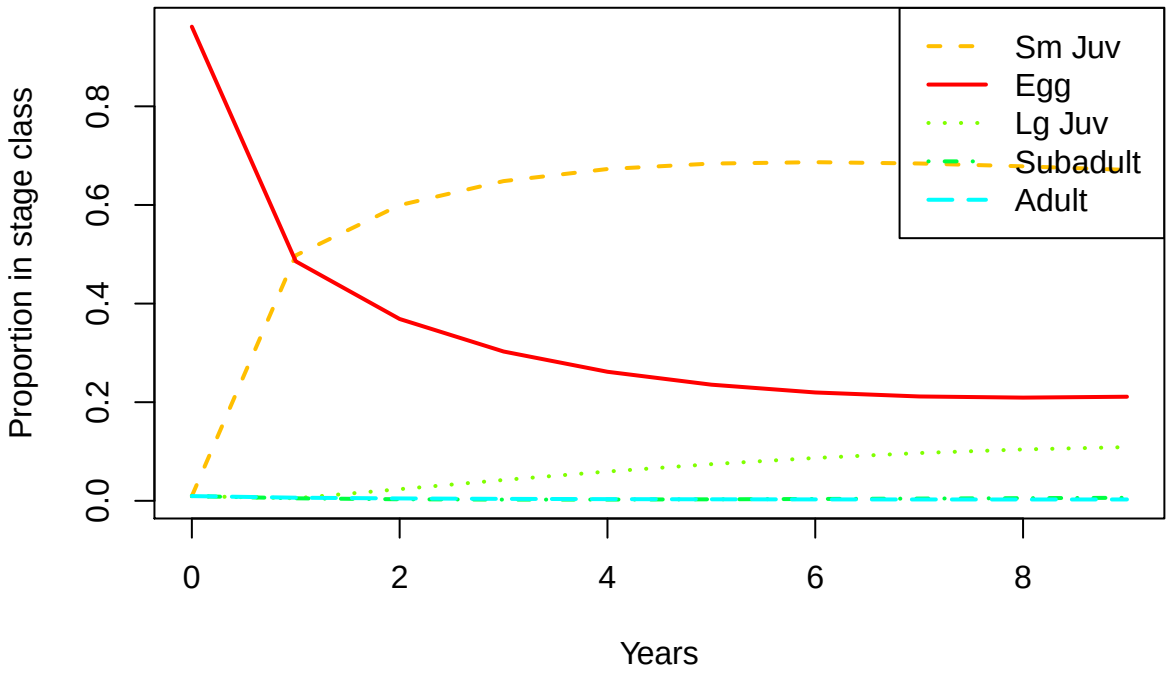
1. Print the matrix you have just created, and ensure that it matches the one in Table 2 of Crowder et al. (1994) (linked on the GauchoSpace page)
2. Print out the subsets of `A` described in the list above. Do you get the values you expect? Do you understand how matrix subsetting works? If not, what don't you understand?
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.

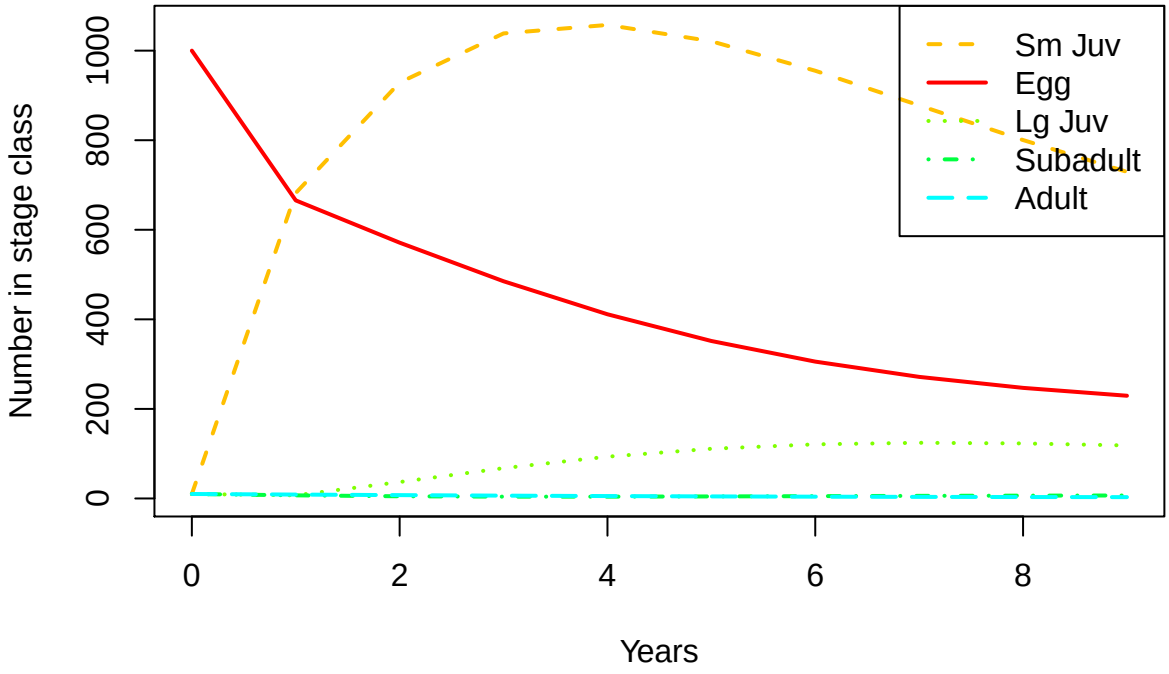
However, if you don't want to wrap your head around `for` loops, the **popbio** package provides some functions that simplify the process. For example, to initialize the population with 1000 eggs and 10 individuals in each stage, and simulate for 10 years, you would do:

```
library(popbio) # You may need to install this first with install.packages("popbio")
n_0 <- c(1000, 10, 10, 10, 10) # Initial abundance
pop <- pop.projection(A, n_0, iterations = 10) # Project the matrix
stage.vector.plot(pop$stage.vector) # Plot each stage through time
```



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

```
stage.vector.plot(pop$stage.vector, proportions = FALSE)
```



2.1 Exercises

4. 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).
5. Plot `pop$pop.sizes` and `pop$pop.changes` through time. What do these tell you?
6. 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`)

7. 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?

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(A)
```

```
## [1] 0.9515489
```

```
stable.stage(A)
```

```
##         Egg      Sm Juv      Lg Juv    Subadult       Adult
## 0.238508404 0.647732505 0.103356123 0.007285382 0.003117586
```

3.1 Exercises

8. 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?
9. 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?
    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?
    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?
    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?