Appendix: Numerical methods for the integral projection model

The dominant eigenvalue and associated eigenvectors must be found once the kernel is determined. Finding the eigenvalues and eigenvectors of an integral equation analytically is typically impossible. However, effective numerical methods are available to approximate the solutions of the eigenvalue problem

$$\int_a^b k(x,y)n(y)dy = \lambda n(x) \tag{A1}$$

We used integration methods (Baker 1977), which are the simplest method to implement. In integration methods, a numerical quadrature rule is applied to approximate the integral, resulting in an equation of the form

$$\sum_{i=0}^{m} w_j k(x, y_j)\tilde{n}(y_j) = \tilde{\lambda}\tilde{n}(x) \tag{A2}$$

where the y_j are the quadrature mesh points and the w_j are the quadrature weights. Equation (A2) is solved by setting $x = y_i$ to get a linear system of equations for the values $\tilde{n}(y_i)$,

$$\sum_{i=0}^{m} w_j k(y_i, y_j)\tilde{n}(y_j) = \tilde{\lambda}\tilde{n}(y_i),\ i = 1,2,\ldots m\ . \tag{A3}$$

This in turn can be written as the matrix eigenvector/eigenvalue equation

$$KD\tilde{n} = \tilde{\lambda}\tilde{n}\ , \tag{A4}$$

where $K_{ij} = k(y_i, y_j)$, $\tilde{n} = [\tilde{n}(y_0), \tilde{n}(y_1), \cdots \tilde{n}(y_m)]^T$ and $D = diag(w_0, w_1, \cdots w_m)$, which can be solved by standard methods (Caswell 1989). In all the calculations here, we used the Gauss-Legendre quadrature method (Baker 1977) for selecting the quadrature weights and mesh points.

It can be shown that the eigenvalues of (A3) converge to eigenvalues of the integral equation (A1) and that eigenvalues of (A1) are the limits of sequences of eigenvalues from (A3) (Hackbusch 1995; the key results are summarized in Easterling 1998 Appendix 5.4). It is important to note that although (A4) involves a matrix it is a numerical method for solving an integral equation, and not a matrix projection model in general.

Our numerical routine based on (A4) operated in the following manner. We started with a coarse approximation of the integral equation (e.g. 5 mesh points, resulting in a 5×5 matrix) and calculated the dominant eigenvalue. The dominant eigenvalue for the 6-point approximation was then computed. If the two eigenvalues were within a given tolerance of each other then the routine stopped, otherwise the 7-point approximation was computed and compared to the 6-point value. The routine continues in this way until consecutive dominant eigenvalues are within the tolerance.

Users who are already experienced with matrix population models may prefer a different choice of mesh points and weights, even though they are numerically less efficient (i.e., many more mesh points may be required to achieve a given degree of accuracy). These mesh points have the property that the numerical approximation to the integral model is equivalent to treating the matrix *KD* as if it were a population projection matrix. This is not recommended as a recipe for building a size-structured matrix model, but simply as a way of "tricking" your old matrix model routines into producing numerical solutions to the integral projection model.

These mesh points are defined as follows. Let *L* and *U* be the lower and upper limits of integration for the integral model, and define "stage boundary" points

$$\beta_i = L + (i/n)(U - L), \qquad i = 0,1,2,\cdots,n. \tag{A5}$$

The corresponding mesh points are

$$y_i = (\beta_{i-1} + \beta_i)/2, \qquad i = 1,2,\cdots,n \tag{A6}$$

and the resulting entries in the matrix $M = KD$ are

$$M_{ij} = (U - L)k(y_i, y_j)/n, \qquad i, j = 1,2,\cdots,n. \tag{A7}$$

The matrix size *n* needed for accurate results will be problem-dependent. As in our procedure described above, the user can start with a small value of *n* (20 or so, which is no challenge to current matrix software) and increase *n* until plots of the left and right eigenvectors as a function of size (y_i) cease to change by a meaningful amount. The matrix *M* can then be analyzed and nd interpreted exactly as if it were a population projection matrix with stage boundaries at the sizes β_i.

In the long run, general-purpose software for the integral model should be available that is based on efficient quadrature methods, but for now (A7) is a useful stopgap. **S-plus/R** code illustrating the use of (A7) on the Monkshood kernel, and which can be adapted straightforwardly to any other kernel, is available in the Supplementary Material for this article (**R** is a freeware implementation of the **S** language, which can be obtained from **Statlib**, http://lib.stat.cmu.edu).