Journal of Statistical Software

September 2007, Volume 22, Issue 11. *http://www.jstatsoft.org/*

Estimating and Analyzing Demographic Models Using the popbio Package in R

Chris Stubben
Los Alamos National Lab

Brook Milligan
New Mexico State University

Abstract

A complete assessment of population growth and viability from field census data often requires complex data manipulations, statistical routines, mathematical tools, programming environments, and graphical capabilities. We therefore designed an R package called **popbio** to facilitate both the construction and analysis of projection matrix models. The package consists primarily of the R translation of MATLAB code found in Caswell (2001) and Morris and Doak (2002) for the analysis of projection matrix models. The package also includes methods to estimate vital rates and construct projection matrix models from census data typically collected in plant demography studies. In these studies, vital rates can often be estimated directly from annual censuses of tagged individuals using transition frequency tables. Because the construction of projection matrix models requires careful management of census data, we describe the steps to construct a projection matrix in detail.

Keywords: demography, matrix population model, projection matrix, vital rates, stochastic growth rate.

1. Introduction

Demographic studies focus on estimating the growth, survival, and reproductive success of individuals within a population. Often these fundamental parameters are strongly influenced by the age, size, or life-history stage of the individuals involved. For example, mature loggerhead sea turtles (*Caretta caretta*) have higher survival rates than new hatchlings and these stages contribute to population growth and viability in different ways (Crouse *et al.* 1987). As a result, age- or stage-based projection matrix models are the principal tool for assessing population growth and viability in structured populations (Leslie 1945; Lefkovitch 1965).

Estimation of demographic rates and the subsequent analysis of projection matrix models

remains a challenge in ecology (Caswell 2001; Morris and Doak 2002). For example, selection of the appropriate size or stage classes often requires fitting statistical models, such as logistic regressions, to the census observations in order to identify those state variables that have a strong influence on vital rates. Construction of the projection matrix from the set of individual life-history observations often involves complex data manipulations to correctly summarize vital rates. Determining the precision with which demographic parameters have been estimated generally requires resampling methods which can become complex for large datasets (Kalisz and McPeek 1992). Calculation of population growth rates, sensitivities, and other demographic measures of long-term viability requires eigenvalues and eigenvectors of the projection matrix. Finally, projection of stochastic growth rates based on a random sequence of matrices or other stochastic processes often requires a programming environment.

To address these challenges, we have developed an R package called **popbio** to construct and analyze matrix population models. R is a free software program that is primarily used for statistical computing and graphics (R Development Core Team 2007). However, R is also a complete programming language that supports mathematical tools for matrix analyses, so all the statistical, mathematical, and graphical capabilities needed for demography are available within the R environment.

The **popbio** package consists primarily of the R translation of code from Caswell (2001) and Morris and Doak (2002) written in MATLAB (The MathWorks, Inc. 2007) for the analysis of projection matrix models. In addition, the package includes methods to construct projection matrix models from census data typically collected in plant demography studies. Benefits of this package include standardizing a variety of basic methods to ease the burden of initiating a demographic analysis, even for ecologists not yet familiar with the R language. Use of the package introduces a convenient way to document methods used to summarize census data and to build projection matrices, because simple R scripts can be written that make use of the standardized functions within the package. Those same scripts will enable other researchers to reproduce the analysis simply by re-executing the scripts on the same dataset.

1.1. Outline of the paper

The first section of the paper focuses on the analysis of projection matrix models. We discuss the calculation of population growth rates in R and then present two specific examples of R functions translated from MATLAB code in Caswell (2001) and Morris and Doak (2002). In the first example, we introduce three functions to calculate age-specific rates from a stage-classified matrix described in Chapter 5 in Caswell (2001). In the second example, we present three functions to calculate stochastic growth rates described in Chapter 7 in Morris and Doak (2002). A complete list of converted functions for the analysis of projection matrix models within each book can be accessed within R using the **popbio** help pages.

```
R> help("Caswell")
R> help("Morris")
```

The second section of the paper focuses on the estimation of vital rates and construction of a single projection matrix. We follow the approach described in Caswell (2001, section 6.1.1) and estimate state transition rates using transition frequency tables. Therefore, much of our discussion and examples apply to demographic studies of plants or other sessile organisms where individuals are tagged and their survival, growth, and reproductive success are

consistently measured over several projection intervals. Morris and Doak (2002) present additional methods to estimate vital rates based on logistic regression and the **RCapture** package includes methods to estimate vital rates in capture-recapture experiments (Baillargeon and Rivest 2007). Because the construction of transition frequency tables and projection matrices requires careful management of census data, we describe these steps in some detail for new R users. We also describe how to perform logistic regression to identify potential stage classes and generate a series of resampled projection matrices.

1.2. R for new users

For ecologists that are new to the R programming language, the extensive online documentation (R Development Core Team 2007) is invaluable. A number of introductory books (Dalgaard 2002; Crawley 2005) complement the primary R documentation. In particular, Crawley (2005) includes many examples relevant to ecologists and the short discussion of basic matrix operations in the appendix centers around an age-classified demographic model. Because R is based on an open-source model, a large user-community contributes functions and data through add-on packages, which are available from the Comprehensive R Archive Network. For example, the **RCapture** package includes functions for the estimation of vital rates in a capture-recapture experiment (Baillargeon and Rivest 2007). In addition, the **demogR** package includes functions for analyzing life tables and matrix population models in human and other age-structured populations (Jones 2007).

Finally, the **popbio** package itself contains help pages for each function and data set, examples of useful analyses, and demonstrations of the range of capabilities, all oriented toward easing the process of learning how to analyze demographic data. Two demonstrations are available within the package; one (`stage.classify`) illustrates the use of logistic regression to identify potential stage classes and the other (`fillmore`) illustrates the construction and analysis of population projection matrices using census data from a plant population. These demos may be run as follows.

```
R> demo("stage.classify")
R> demo("fillmore")
```

Finally, every function and data set includes a detailed help page describing the input parameters, defaults, and working examples. These may be accessed as illustrated below for the `stoch.growth.rate` function.

```
R> help("stoch.growth.rate")
R> example("stoch.growth.rate")
```

2. Analyzing a projection matrix

2.1. Calculating growth rates by projection

At the center of any demographic study of structured populations are the state vector $\mathbf{n}_t$ describing the distribution of individuals across age, size, or stage categories and the population projection matrix $\mathbf{A} = \mathbf{T} + \mathbf{F}$ describing the transformation of the number of individuals

from one census to the next. The matrix **T** (transitions) represents the set of transitions due to growth and survival and the matrix **F** (fertilities) represents the transitions due to reproduction (Caswell 2001). Projections of the age or stage distribution through time are given by the following recursion equation.

$$\mathbf{n}_{t+1} = \mathbf{A}\mathbf{n}_t \tag{1}$$

The population growth rate and stable stage distribution can be calculated by deterministic projections or by matrix computations using the dominant eigenvalue and right eigenvector (Caswell 2001). In the first case, Equation 1 is used repeatedly to create a time series of stage vectors. Eventually, the stage vector $\mathbf{n}_t$ converges to a stable stage distribution $\hat{\mathbf{n}}$, where each component changes by the same proportion (λ) with each progressive iteration.

The function `pop.projection` projects the growth of a population over time by specifying a projection matrix, initial stage vector, and number of iterations. In this example, the projection matrix is the mean matrix for a rare plant (Freville *et al.* 2004) and projections are repeated over 15 intervals. The population growth rate λ and stable stage distribution are estimated from the final iteration. As illustrated in Figure 1, a plot using `stage.vector.plot` of the resulting stage vectors can be helpful for identifying convergence to the stable stage distribution and to explore short-term dynamics.

```
R> stages <- c("seedling", "vegetative", "flowering")
R> A <- matrix(c(0, 0, 5.905, 0.368, 0.639, 0.025, 0.001, 0.152,
+     0.051), nrow = 3, byrow = TRUE, dimnames = list(stages,
+     stages))
R> n <- c(5, 5, 5)
R> p <- pop.projection(A, n, 15)
R> p

$lambda
[1] 0.997

$stable.stage
  seedling vegetative  flowering
    0.4525     0.4711     0.0763

$stage.vectors
           0     1      2     3     4    5     6     7     8     9    10
seedling   5 29.53  6.023  5.11 13.03 10.8  8.85 10.11 10.32  9.78  9.81
vegetative 5  5.16 14.188 11.30  9.16 10.7 10.86 10.24 10.31 10.43 10.30
flowering  5  1.02  0.866  2.21  1.84  1.5  1.71  1.75  1.66  1.66  1.68
              11    12    13    14
seedling    9.92  9.81  9.75  9.74
vegetative 10.24 10.23 10.19 10.14
flowering   1.66  1.65  1.65  1.64

$pop.sizes
 [1] 15.0 35.7 21.1 18.6 24.0 23.0 21.4 22.1 22.3 21.9 21.8 21.8 21.7 21.6
```

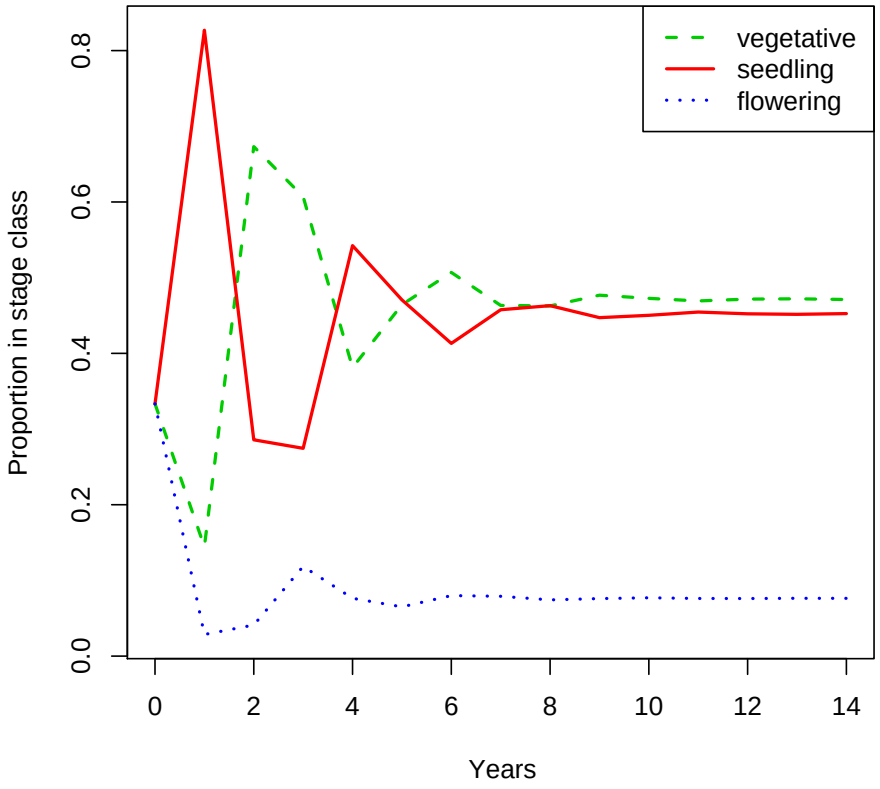

Figure 1: Projected time series for a rare plant with three stage classes.

```
[15] 21.5

$pop.changes
 [1] 2.380 0.590 0.884 1.290 0.959 0.930 1.032 1.008 0.981 0.997 1.001
[12] 0.994 0.995 0.997

R> stage.vector.plot(p$stage.vectors, col = 2:4)
```

2.2. Calculating growth rates using eigenvalues

The function `eigen.analysis` uses the built-in function `eigen` to calculate common demographic parameters from eigenvalues and eigenvectors of the projection matrix. The population growth rate λ is the dominant eigenvalue and the stable stage distribution and reproductive values are the corresponding right and left eigenvectors, respectively. The sensitivity and elasticity matrices identify matrix elements with the greatest absolute or proportional effects on population growth rate (de Kroon *et al.* 2000). The damping ratio measures the

rate of convergence to a stable stage distribution and is determined by the ratio of the dominant eigenvalue to the second largest eigenvalue. Additional details on reproductive value, damping ratio, sensitivity, and elasticity can be found in Caswell (2001). By default, only the sensitivities for non-zero elements in the projection matrix are displayed by `eigen.analysis`.

```
R> eigA <- eigen.analysis(A)
R> eigA

$lambda
[1] 0.996

$stable.stage
  seedling vegetative  flowering
    0.4523     0.4714     0.0763

$sensitivities
           seedling vegetative flowering
seedling      0.000      0.000    0.0346
vegetative    0.552      0.576    0.0932
flowering     1.298      1.353    0.2189

$elasticities
           seedling vegetative flowering
seedling    0.00000      0.000   0.20536
vegetative  0.20406      0.369   0.00234
flowering   0.00130      0.206   0.01121

$repro.value
  seedling vegetative  flowering
      1.00       2.69       6.32

$damping.ratio
[1] 6.51
```

2.3. Age-specific rates from a stage-classified matrix

Demographic studies often classify individuals by size or stage class, especially when life history stages are easy to measure but ages are difficult to determine. In other cases, size variables are better than age at predicting survival, growth and reproduction. This is commonly the case for most plants and for animals with indeterminate growth. Because stage-structured demography is based on observations revealing both survival and growth of individuals from one census to the next, age-specific rates may be calculated from the resulting stage-structured model (Cochran and Ellner 1992; Tuljapurkar and Horvitz 2006). Important quantities include age-specific survival, the net reproductive rate, and generation time.

Caswell (2001, Chapter 5) develops a number of methods for determining age-specific traits from stage-classified models using examples from killer whales (*Orcinus orca*) and teasel

(*Dipsacus sylvestris*). For illustrative purposes, the growth and survival matrix (**T**) and the fertility matrix (**F**) for these two examples are available within the **popbio** package in the data sets `whale` and `teasel`.

The most fundamental aspect of age-specific survival is given by the mean time spent by individuals in each stage class. Related quantities include the variance and the coefficient of variation of the mean time spent in each stage class. Also useful are the mean and variance of the time to death. The R function `fundamental.matrix` returns a list containing all five of these quantities given a transition matrix **T** as input. In the illustration below, only the mean time in each stage class (**N**) is displayed. On average, a yearling killer whale spends one year as a yearling, 11 years as a juvenile, and about 17.4 years as a reproductive adult.

```
R> data("whale")
R> fundamental.matrix(whale$T)$N
```

```
           yearling juvenile mature postreprod
yearling        1.0      0.0    0.0          0
juvenile       11.0     11.2    0.0          0
mature         17.4     17.8   21.5          0
postreprod     40.0     41.0   49.5         51
```

The net reproductive rate R_0 is a measure of the replacement rate of a population and equals the mean number of offspring produced by a single newborn individual during its lifetime. In plant demography, many plants may produce more than one type of offspring class in a prebreeding census, since seeds will be released just after the census and may remain in the seed bank or germinate and develop into seedlings or small vegetative classes during the census interval. In contrast, mature plants only produce one type of offspring in a postbreeding census, the recently formed and immediately censused seeds. To cover the prebreeding census situation, Caswell (2001) recommends using the dominant eigenvalue of the matrix $\mathbf{R} = \mathbf{FN}$ to calculate net reproductive rate.

```
R> data("teasel")
R> net.reproductive.rate(teasel$T, teasel$F)
```

```
[1] 14.4
```

The generation time for a structured population is defined in several different ways. Within the **popbio** package it is defined as the time required for a population to increase by a factor of R_0 (Caswell 2001, equation 5.73).

```
R> generation.time(teasel$T, teasel$F)
```

```
[1] 3.15
```

2.4. Modeling population viability and stochastic growth rates

Morris and Doak (2002) model population growth and extinction risks of structured populations in variable environments in Chapter 7 and the following stochastic growth functions

were converted from the MATLAB code in Boxes 7.3 to 7.5. Details on default parameters are found in the corresponding help pages and all three examples in this section use four projection matrices for mountain golden heather (*Hudsonia montana*) in the `hudsonia` dataset. All three functions accept any list of two or more projection matrices as the main input parameter and the `sample` function is then used internally to select matrices at random. By default, all three functions print helpful comments that track the progress during long iterations. For clarity and presentation below, we have suppressed the output of comments with the `verbose=FALSE` option.

First, the function `stoch.projection` can be used to simulate stochastic population growth. The function requires a list of projection matrices and an initial population vector as input; reasonable defaults are given for the number of time steps, iterations, and selection probabilities. The output is a matrix listing the final population vector in the last time step for each iteration. In this example, the golden heather population is projected through 50 time steps for both a uniform (`x.eq`) and a nonuniform (`x.uneq`) selection of matrices. In the latter case, the probability distribution for sampling is given as the vector argument `prob`, which is used by `sample` to select the matrices.

```
R> data("hudsonia")
R> n <- c(4264, 3, 30, 16, 25, 5)
R> names(n) <- c("seed", "seedlings", "tiny", "small", "medium",
+     "large")
R> x.eq <- stoch.projection(hudsonia, n, nreps = 1000, verbose = FALSE)
R> x.uneq <- stoch.projection(hudsonia, n, nreps = 1000, prob = c(0.2,
+     0.2, 0.2, 0.4), verbose = FALSE)
```

The vast majority of final population sizes of *Hudsonia montana* are much less than the initial size ($n = 4343$), a dramatic illustration of a population unlikely to persist under these two stochastic models (Figure 2). In the figure, the blue color representing matrix selection from a nonuniform distribution where the "best" year with a growth rate of 1.02 is selected 40% of the time is partially transparent, allowing the overlap between the two histograms to be clearly visible.

Second, the methods used by the `stoch.projection` function can also be modified to yield an estimate of the log stochastic growth rate by averaging successive growth rates over a very long simulation (50,000 iterations by default). In addition to calculating stochastic growth rates by matrix selection and projection, they may also be calculated using Tuljapukar's second-order approximation (Tuljapurkar 1990). The function `stoch.growth.rate` calculates the logarithm of the stochastic growth rate using both methods.

```
R> stoch.growth.rate(hudsonia, verbose = FALSE)

$approx
[1] -0.0371

$sim
[1] -0.0366
```

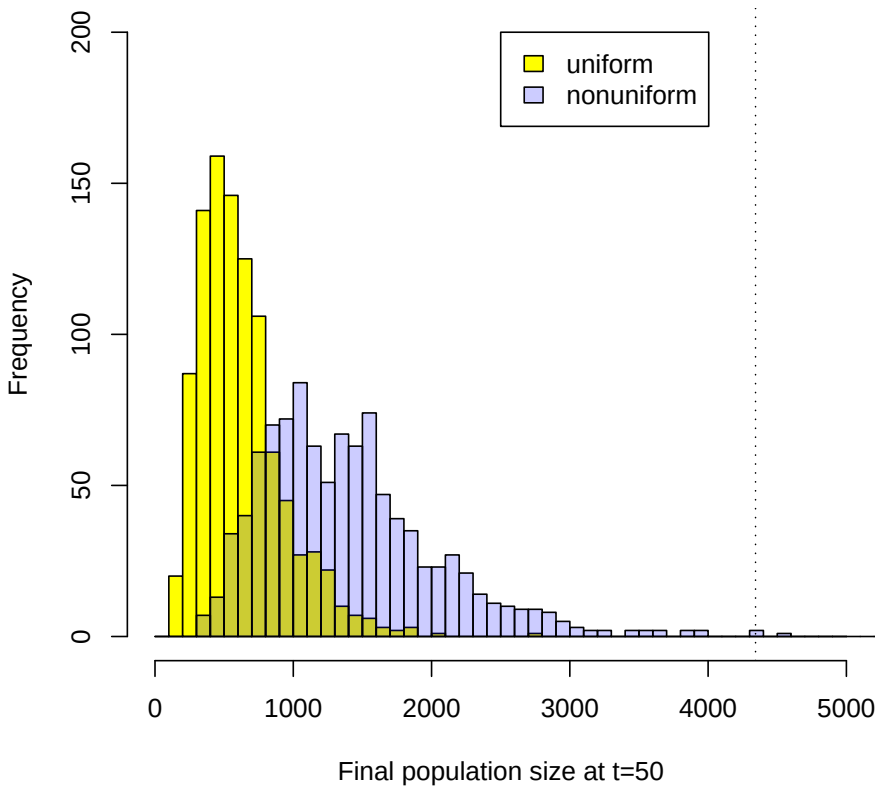

Figure 2: Projection of stochastic growth rates for *Hudsonia montana* with histograms for both uniform and nonuniform sampling probabilities. The dotted line represents the starting population size.

```
$sim.CI
[1] -0.0372 -0.0360
```

Finally, the function `stoch.quasi.ext` estimates the probability of reaching a quasi-extinction threshold. Figure 3 illustrates the results obtained using an extinction threshold of 10 above-ground individuals in ten separate runs, each with 500 iterations of population growth over 50 years. Since the stage distribution of most plant populations like *H. montana* are predominantly seeds, this stage is excluded from the extinction threshold using the `sumweight` option.

```
R> sqe <- stoch.quasi.ext(hudsonia, n, Nx = 10, nreps = 500,
+     sumweight = c(0, 1, 1, 1, 1, 1), verbose = FALSE)
R> matplot(sqe, xlab = "Years", ylab = "Quasi-extinction probability",
+     type = "l", lty = 1, col = rainbow(10))
```

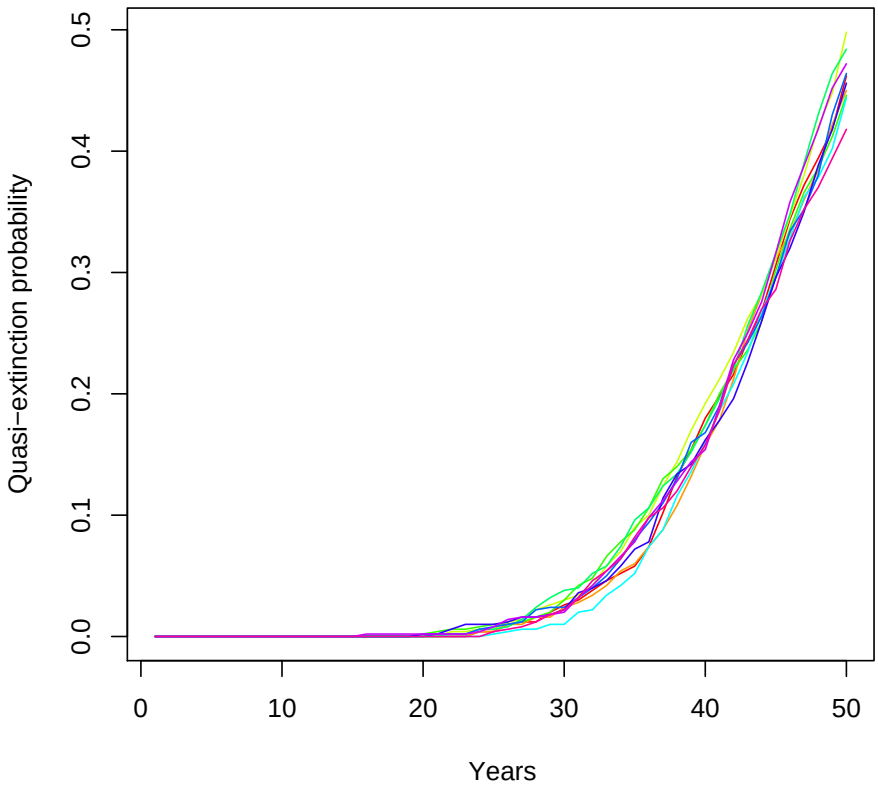

Figure 3: Time for the *Hudsonia montana* population to reach a quasi-extinction threshold of 10 above-ground individuals. The lines are separate estimates of the cumulative distribution of extinction probabilities based on 500 iterations of population growth over 50 years.

3. Constructing a projection matrix

From an experimental viewpoint, a demographic study follows marked or tagged individuals and, via multiple observations throughout their lives, estimates their growth, survival, and reproductive success. These individuals may be classified by age, size, or life-history stage at each of a repeated set of censuses. Thus, the initial census data consist of a life-history trajectory for each identified individual, with each point in the trajectory composed of the date or time of the census, the identifier of the individual, and information on the individual's current age, size, or stage, and its reproductive status.

Census records are easily organized into an R data frame in longitudinal format. To illustrate this, we provide two census data sets, one (`test.census`) from a hypothetical plant population sampled over a three year period and one (`aq.census`) from an ongoing study of *Aquilegia* (Ranunculaceae) populations in the southwestern United States. The following fragments from these two data frames illustrate the general structure typical of demographic datasets.

```
R> data("test.census")
R> head2(test.census)

   plant year    stage fruits
1      1 2001 seedling      0
2      2 2001 seedling      0
3      3 2001 seedling      0
.      .    .        .      .
41    18 2003 seedling      0

R> data("aq.census")
R> head2(aq.census)

     plot year plant  status rose leaf infl fruits
1     903 1996     1 dormant    0    0   NA      0
2     903 1996     2  flower   NA   NA   NA      1
3     903 1996     3 dormant    0    0   NA      0
.       .    .     .       .    .    .    .      .
2853  930 2003   258    dead   NA   NA   NA      0
```

Construction of a projection matrix from a census data frame typically requires the following steps, each of which will be described in the following sections. Because the state transition rates are estimated using transition frequency tables (Caswell 2001, Section 6.1.1), these methods apply to demographic studies of plants or other sessile organisms where all tagged individuals are observed and measured during each census.

1. Creating a stage-fate data frame by linking each observation of an individual during one census with its fate as revealed by the subsequent census.
2. Possibly adding one or more fertility columns to the stage-fate data frame to record the reproductive success of each individual. This may be necessary if individual reproduction cannot be calculated completely from the census data.
3. Converting the stage-fate data frame into a transition frequency table to construct a transition matrix and summarizing individual fertility rates to construct a fertility matrix.
4. Possibly adding directly to the transition matrix additional life-cycle transitions that do not correspond to those observable in the census data. For example, survival rates for cryptic stages such as seeds in the seed bank may be estimated from experimental data rather than observed in the census.

3.1. Creating stage-fate data frames

To illustrate the linking of census observations on each individual to create stage-fate data, consider the `test.census` example. It includes step-by-step instructions to build a stage-fate data frame using two R methods: `reshape` and `merge`. In the latter case, the `merge` function

executes a self-join on the census data frame using common identifiers (in this case `plant`, the identifier of individuals) and then `subset` matches rows where `year` in the first copy of the census table equals `year-1` in the second copy of the census table.

```
R> trans <- subset(merge(test.census, test.census, by = "plant",
+     sort = FALSE), year.x == year.y - 1)
R> head2(trans)
```

```
   plant year.x    stage.x fruits.x year.y    stage.y fruits.y
2      1   2001   seedling        0   2002       dead        0
7      2   2001   seedling        0   2002 vegetative        0
12     2   2002 vegetative        0   2003 vegetative        0
.      .      .          .        .      .          .        .
97    16   2002   seedling        0   2003       dead        0
```

The simple self-join above yields column names that may not express their content as clearly as possible. Rename the rows and columns to improve their clarity by assigning new names.

```
R> rownames(trans) <- 1:nrow(trans)
R> colnames(trans)[2:7] <- c("year", "stage", "fruits", "year2",
+     "fate", "fruits2")
R> head2(trans)
```

```
   plant year      stage fruits year2       fate fruits2
1      1 2001   seedling      0  2002       dead       0
2      2 2001   seedling      0  2002 vegetative       0
3      2 2002 vegetative      0  2003 vegetative       0
.      .    .          .      .     .          .       .
23    16 2002   seedling      0  2003       dead       0
```

3.2. Adding fertility rates

The stage-specific fertility rates in the final projection matrix are estimated by averaging individual fertility rates by stage, so census records should contain counts or estimates of offspring production. For certain types of life cycles, a partial estimate of reproductive output is observable in a census but not the entire transition corresponding to reproduction. For example, fruit or seed production is observable, but during a complete census interval seeds are dispersed and not reobserved until they emerge as germinating seedlings. In such cases, the census dataset lacks information on the direct link between parents and offspring that is required in the fertility matrix. One solution is to calculate relative fertilities based on the proportion of total reproductive output attributable to an individual times the total number of seedlings at the end of the projection interval. This may be calculated and inserted into a new stage-fate data frame for a single census interval (`trans01`) as follows.

```
R> trans01 <- subset(trans, year == 2001, c(plant, stage, fruits,
+     fate))
```

```
R> seedlings <- nrow(subset(test.census, year == 2002 & stage ==
+     "seedling"))
R> seedlings

[1] 5

R> trans01$seedling <- trans01$fruits/sum(trans01$fruits) *
+     seedlings
R> trans01

   plant        stage fruits         fate seedling
1      1     seedling      0         dead    0.000
2      2     seedling      0   vegetative    0.000
4      3     seedling      0         dead    0.000
5      4     seedling      0         dead    0.000
6      5     seedling      0   vegetative    0.000
8      6     seedling      0   vegetative    0.000
10     7   vegetative      0 reproductive    0.000
13     8   vegetative      0   vegetative    0.000
14     9 reproductive      2         dead    0.667
15    10 reproductive      4 reproductive    1.333
17    11 reproductive      9 reproductive    3.000
```

Several subtleties emerge when estimating individual fertility rates in species with anonymous reproduction (Caswell 2001). First, the timing of the birth pulse (e.g., when seeds are released from fruits in plant reproduction) relative to the timing of the census influences the values of fertility elements within the projection matrix $\mathbf{A}$. Second, the longevity of anonymous reproductive propagules influences the handling of fertilities; for example, persistent seed banks may require the introduction of additional, unobservable stages or other complexities. Finally, differences in survival and growth (e.g., germination for plants) for propagules of different ages (e.g., an age-structured seed bank) introduce further challenges. Because these generally must be handled in special ways to account for unique biological features of different life cycles, there is no general modeling method available. However, the ability to add specific fertilities to the stage-fate data frame and the mathematical sophistication of R allow appropriate calculations to be performed without reliance on external software.

Specific comments are possible, however, for the influence of simple pre- and postbreeding censuses of plants without seed banks on the estimates of individual fertility in the stage-fate data frame. For a prebreeding census, i.e., when each census immediately precedes the pulse of reproduction, fertility rates should include the survival of offspring during the census interval. For a postbreeding census, however, the seeds were just released immediately prior to each census, so the fertility estimates are just the total number of seeds produced during the immediately preceding reproductive pulse.

3.3. Constructing transition and fertility matrices

After having followed the steps outlined above, the original census data have been transformed into a data frame (`trans01`) describing the connection between the stage of an individual

during one census and its fate and reproductive output as revealed by the subsequent one. The next step creates the projection matrix **A**, which is composed of the transitions representing growth and survival (**T**) and those representing fertility (**F**) (Caswell 2001). All of these are created by the `projection.matrix` function and the function includes a number of reasonable defaults for the stage, fate, and fertility column names. By default, the matrix is sorted by the `factor` levels in the stage column. To change the order of columns, add the option to sort by a given stage vector or first order the stage column using `ordered` as illustrated below. The function also includes an option to output a list with separate transition and fertility matrices if needed.

```
R> stages <- c("seedling", "vegetative", "reproductive")
R> projection.matrix(trans01, sort = stages)
```

```
              seedling vegetative reproductive
  seedling         0.0        0.0        1.667
  vegetative       0.5        0.5        0.000
  reproductive     0.0        0.5        0.667
```

```
R> trans01$stage <- ordered(trans01$stage, levels = stages)
R> projection.matrix(trans01, TF = TRUE)
```

```
$T
```

```
              seedling vegetative reproductive
  seedling         0.0        0.0        0.000
  vegetative       0.5        0.5        0.000
  reproductive     0.0        0.5        0.667
```

```
$F
```

```
              seedling vegetative reproductive
  seedling           0          0         1.67
  vegetative         0          0         0.00
  reproductive       0          0         0.00
```

Since the step of constructing a projection matrix is so central to demographic analyses, it warrants more detailed description. The following outlines the steps that are carried out internally by `projection.matrix`. First, the `table` function is used to cross-tabulate stage by fate and create a state transition frequency table with state at time t in columns and fate at time $t+1$ in rows. This arrangement corresponds to the traditional orientation of demographic projection matrices, which postmultiply the projection matrix by the state vector $\mathbf{n}_t$ to yield the state for the following time, $\mathbf{n}_{t+1}$ (see Equation 1). In this example, the fates are sorted with the `dead` fate appearing in the last row for display.

```
R> tf <- table(trans01$fate, trans01$stage)
R> tf[c(stages, "dead"), ]
```

```
              seedling vegetative reproductive
  seedling           0          0            0
  vegetative         3          1            0
  reproductive       0          1            2
  dead               3          0            1
```

Next, the `prop.table` function is used to divide transition counts by the column total to get state transition rates. The `dead` fate must be removed from the final matrix. This can be accomplished by retrieving only those rows and columns listed in the stage vector.

```
R> T.mat <- prop.table(tf, 2)[stages, stages]
R> T.mat
```

```
              seedling vegetative reproductive
  seedling         0.0        0.0        0.000
  vegetative       0.5        0.5        0.000
  reproductive     0.0        0.5        0.667
```

Finally, the `tapply` function is used to calculate the stage-specific fertility rates by averaging individual fertilities by initial stage class to create a row of mean fertilities. The corresponding row is then added to a matrix of zeros. Note that in many stage-structured populations, fertility estimates in a prebreeding census may be included in multiple rows (e.g., corresponding to fates as both seeds and seedlings if some can progress rapidly through the early life cycle stages).

```
R> F.mat <- T.mat * 0
R> F.mat[1, ] <- tapply(trans01$seedling, trans01$stage, mean)
R> F.mat
```

```
              seedling vegetative reproductive
  seedling           0          0         1.67
  vegetative         0          0         0.00
  reproductive       0          0         0.00
```

3.4. Adding estimated transitions directly to matrix

In many species, it is also not possible to directly observe all possible transitions in a population, especially for cryptic transitions such as seed bank survival. These must be added directly into the projection matrix. To illustrate this, consider the formatted stage-fate data frame for *Aquilegia* called `aq.trans` included in the package. The following steps calculate two additional elements to add to the transition matrix, one for seed survivorship within the seed bank and one for recruitment from the seed bank into the seedling class. In this case, both are calculated under the assumptions that the observations were obtained from a prebreeding census and that all seeds are equivalent within the seed bank.

First, an appropriate, single-year subset of the data are obtained.

```
R> data("aq.trans")
R> aq96 <- subset(aq.trans, year == 1996, select = c(plot, plant,
+     stage, fate, fruits))
R> head2(aq96)

     plot plant  stage  fate fruits
1     903     1  small small      0
2     903     2 flower large      1
3     903     3  small large      0
.       .     .      .     .      .
1331  930   106 flower small      5
```

Experiments outside of the scope of the demographic census yielded estimates of the seed survival rate, the initial seed bank size, and the number of seeds per fruit.

```
R> seed.survival <- 0.126
R> seed.bank.size <- 10000
R> seeds.per.fruit <- 120
```

Next, assume that both newly matured seeds and those in the seed bank contribute equally to the recruit pool observed in the next census and therefore to the recruitment rate.

```
R> recruits <- nrow(subset(aq.trans, year == 1997 & stage ==
+     "recruit"))
R> seeds.from.plants <- sum(aq96$fruits) * seeds.per.fruit
R> recruitment.rate <- recruits/(seed.bank.size + seeds.from.plants)
```

Individual fertilities corresponding to reproductive transitions from mature plants to new recruits and into the seed bank are added directly to the stage-fate data frame using the following equations.

```
R> aq96$recruit <- aq96$fruits/sum(aq96$fruits) * seeds.from.plants *
+     recruitment.rate
R> aq96$seed <- aq96$fruits * seeds.per.fruit * seed.survival
R> head2(aq96)

     plot plant  stage  fate fruits recruit seed
1     903     1  small small      0    0.00  0.0
2     903     2 flower large      1    1.02 15.1
3     903     3  small large      0    0.00  0.0
.       .     .      .     .      .       .    .
1331  930   106 flower small      5    5.12 75.6
```

Finally, estimated transitions are added directly as new elements in the matrix with the `add` option to `projection.matrix` by specifying a series of three arguments for each new element: the row and column indices for the new element and the value of that element. In this example, estimates of seed bank survival and recruitment are added directly to the transition matrix as elements $(1, 1)$ and $(2, 1)$, respectively.

```
R> A96 <- projection.matrix(aq96, add = c(1, 1, seed.survival,
+     2, 1, recruitment.rate))
R> A96
```

```
            seed recruit small large  flower
seed    0.12600     0.0 0.000 0.000 48.0426
recruit 0.00853     0.0 0.000 0.000  3.2530
small   0.00000     0.5 0.403 0.176  0.3065
large   0.00000     0.0 0.239 0.471  0.4839
flower  0.00000     0.0 0.000 0.118  0.0806
```

Because these steps are a routine part of our demographic analysis of *Aquilegia*, they are incorporated into the `aq.matrix` function. Its help page also explains these steps in greater detail.

For other species, the additional transitions required to represent the corresponding life cycle may be different than those described above. Consequently, a somewhat different procedure will be required leading up to the final construction of the projection matrix. The mathematical completeness of the R language, however, ensures that all the calculations can be done directly and can be incorporated into a customized function analogous to `aq.matrix` for convenience and to aid the documentation.

3.5. Selecting stage classes

Definition of appropriate age, size, or stage classes requires identifying states with a strong influence on the vital rates. Ideally, a state variable will be highly correlated with all three vital rates—survival, growth, and reproduction—thus enabling prediction of an individual's fate and performance based on information about its state (Morris and Doak 2002, Chapter 6). Graphical and statistical analyses often help to identify those state variables that have the best power to predict differences in vital rates among individuals of different sizes.

In the next example, *Aquilegia* rosette size is examined as a potential explanatory state variable for survival. A logical vector for survival is first added to the data frame. A subset for flowering plants is then selected and the `table` function is used to summarize survival of flowering plants by rosette size class. Because the number of surviving plants is binomially distributed, those data are fit to a logistic regression using the R function `glm` with a logit link function (Crawley 2005). The same procedure is repeated for non-flowering plants. Figure 4 illustrates both the data and the resulting logistic regression functions. In this case it is clear that rosette size is a useful explanatory variable for survivorship, and different relationships hold for flowering and non-flowering individuals.

```
R> aq.trans$survived <- aq.trans$fate != "dead"
R> flwr <- subset(aq.trans, rose > 0 & rose <= 7 & stage ==
+     "flower")
R> x.flwr <- table(flwr$rose, flwr$survived)
R> x.flwr
```

```
  FALSE TRUE
1    10   27
```

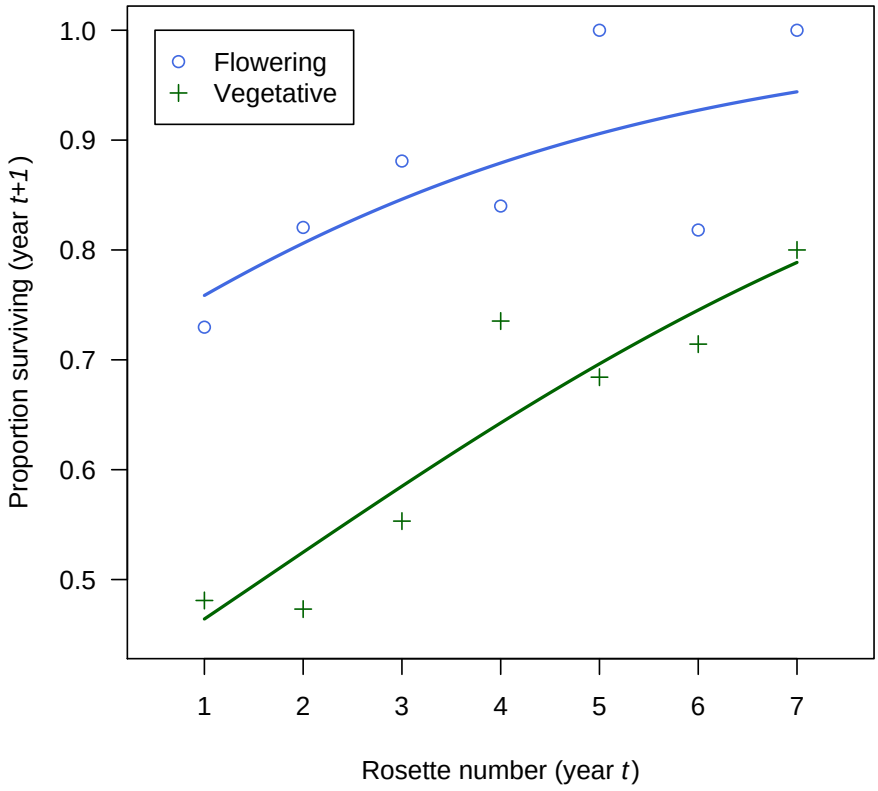

Figure 4: Survival rates in *Aquilegia* depend on flowering state and plant size.

```
  2     7   32
  3     5   37
  4     4   21
  5     0   10
  6     2    9
  7     0    5
```

```
R> rose <- 1:7
R> glm.flwr <- glm(x.flwr ~ rose, binomial)
R> non.flwr <- subset(aq.trans, rose > 0 & rose <= 7 & stage %in%
+     c("large", "small"))
R> x.non.flwr <- table(non.flwr$rose, non.flwr$survived)
R> glm.non.flwr <- glm(x.non.flwr ~ rose, binomial)
```

Some models commonly used to fit relationships between vital rates and potential state variables are listed in Morris and Doak (2002, Table 6.1). These include regression, analysis of variance, and generalized linear models such as logistic regression and loglinear models. In

addition, demographic studies may include life table response experiments involving fixed, random, or regression designs and models to evaluate spatiotemporal variation in transition rates (Caswell 2001). R includes functions for these and many other statistical models (Pinheiro and Bates 2000; Venables and Ripley 2002), which enables researchers to conduct a variety of statistical tests in a demography study.

3.6. Resampling transitions

A detailed description of resampling methods to estimate confidence intervals or other measures of uncertainty for demographic estimates is described in Chapter 12 in Caswell (2001). We illustrate one approach to generate a series of bootstrap projection matrices by resampling from the set of observed transitions in the stage-fate data frames below.

```
R> n <- nrow(trans01)
R> n

[1] 11

R> x <- sample(n, replace = TRUE)
R> x

 [1]  1 10  1  3  1  4  8  2 10  4  3

R> bt <- trans01[x, ]
R> bt

     plant        stage fruits         fate seedling
1        1     seedling      0         dead     0.00
15      10 reproductive      4 reproductive     1.33
1.1      1     seedling      0         dead     0.00
4        3     seedling      0         dead     0.00
1.2      1     seedling      0         dead     0.00
5        4     seedling      0         dead     0.00
13       8   vegetative      0   vegetative     0.00
2        2     seedling      0   vegetative     0.00
15.1    10 reproductive      4 reproductive     1.33
5.1      4     seedling      0         dead     0.00
4.1      3     seedling      0         dead     0.00

R> projection.matrix(bt)

               seedling vegetative reproductive
  seedling        0.000          0         1.33
  vegetative      0.125          1         0.00
  reproductive    0.000          0         1.00
```

Repeating the procedure above will generate additional bootstrap matrices. The function `boot.transitions` simplifies the procedure and returns projection matrices, stage vectors and population growth rates for each bootstrap sample. The function also includes the option to resample by subsets of initial class counts. As shown in Figure 5, the resampled values of population growth rate or other demographic parameters can easily be plotted using `hist` with 95% confidence intervals calculated using `quantile` and then added to the histogram using `abline`.

```
R> boot.transitions(trans01, 5)

$lambda
[1] 0.000 1.380 0.829 1.545 1.393

$matrix
  a11   a21 a31 a12   a22   a32   a13 a23  a33
1   0 0.800   0   0 0.000 1.000 0.000   0 0.00
2   0 0.600   0   0 0.000 1.000 2.000   0 0.75
3   0 0.500   0   0 0.667 0.333 0.667   0 0.00
4   0 0.600   0   0 0.000 1.000 2.167   0 1.00
5   0 0.571   0   0 0.000 1.000 1.333   0 1.00

$vector
  seedling vegetative reproductive
1       10          1            0
2        5          2            4
3        4          6            1
4        5          4            2
5        7          2            2

R> x <- boot.transitions(trans01, 200)
R> ci <- quantile(x$lambda, c(0.025, 0.975))
R> hist(x$lambda, col = "grey80", xlab = "Lambda", main = "")
R> abline(v = ci, lty = 3)
```

4. Conclusion

Beyond the empirical challenges associated with demographic studies, complete analysis of the resulting demographic data requires a variety of statistical and mathematical calculations. Fully understanding the dynamics of natural populations with complex life cycles requires great care and errors of interpretation resulting from incorrect analysis are not uncommon (Caswell 2001). Consequently, there is a great need for statistical and graphical methods that ease the initial demographic analysis but do not limit the potential to increase sophistication as the analysis proceeds.

The **popbio** package has served well to organize our own demographic research by helping to formalize and document our analyses. By contributing to the community of R users, we

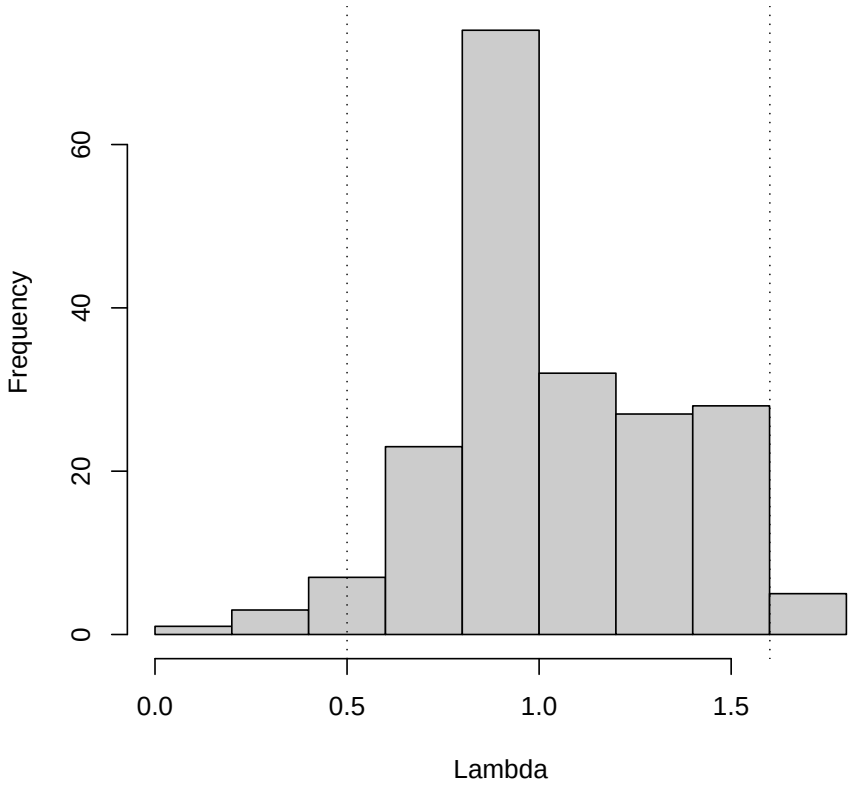

Figure 5: Bootstrap estimates of population growth rates for the hypothetical plant population. Estimates are based on 200 samples from the original stage-fate data frame and 95% confidence intervals are represented by the vertical dotted lines.

hope to foster greater cooperation that results in ongoing improvements in both the formal analysis of demographic data and the availability of demographic datasets for a diversity of natural populations. In the future, we plan to include addtional functions and graphical methods described in both Caswell (2001) and Morris and Doak (2002). The community of demographers can also aid with the improvement of analytical techniques by suggesting improvements to existing methods and by integrating census data from many different studies into the package for comparative analyses.

Acknowledgments

We thank Patrick Nantel for recent contributions to the **popbio** package. New functions include methods to calculate count-based extinction probabilities (Morris and Doak 2002, Chapter 3) and model stochastic growth rates, extinction rates, and demographic stochasticity based on vital rates (Morris and Doak 2002, Chapter 8). This publication was made

possible by support from NIH Grant Number RR-16480 from the BRIN/INBRE Program of the National Center for Research Resources, and from NSF Grant Number 0420407 from the Centers for Research Excellence in Science and Technology Program of the Division of Human Resource Development.

References

Baillargeon S, Rivest L (2007). "The **Rcapture** Package: Loglinear Models for Capture-Recapture Experiments in R." *Journal of Statistical Software*, **19**(5). URL http://www.jstatsoft.org/v19/i05/.

Caswell H (2001). *Matrix Population Models: Construction, Analysis, and Interpretation*. Sinauer, Sunderland, MA, second edition.

Cochran ME, Ellner S (1992). "Simple Methods for Calculating Age-based Life History Parameters for Stage-structured Populations." *Ecological Monographs*, **62**(3), 345–364.

Crawley MJ (2005). *Statistics: An Introduction Using R*. John Wiley & Sons Ltd., West Sussex, UK.

Crouse DT, Crowder LB, Caswell H (1987). "A Stage-based Population Model for Loggerhead Sea Turtles and Implications for Conservation." *Ecology*, **68**(5), 1412–1423.

Dalgaard P (2002). *Introductory Statistics with R*. Statistics and Computing. Springer-Verlag, New York.

de Kroon H, van Groenendael J, Ehrlen J (2000). "Elasticities: A Review of Methods and Model Limitations." *Ecology*, **81**(3), 607–618.

Freville H, Colas B, Riba M, Caswell H, Mignot A, Imbert E, Olivieri I (2004). "Spatial and Temporal Demographic Variability in the Endemic Plant Species *Centaurea corymbosa* (Asteraceae)." *Ecology*, **85**(3), 694–703.

Jones JH (2007). "**demogR**: A Package for Evolutionary Demographic Analysis in R." *Journal of Statistical Software*, **22**(10). URL http://www.jstatsoft.org/v22/i10/.

Kalisz S, McPeek MA (1992). "Demography of an Age-structured Annual: Resampled Projection Matrices, Elasticity Analyses, and Seed Bank Effects." *Ecology*, **73**(3), 1082–1093.

Lefkovitch LP (1965). "The Study of Population Growth in Organisms Grouped by Stages." *Biometrics*, **21**, 1–18.

Leslie PH (1945). "On the Use of Matrices in Population Mathematics." *Biometrika*, **33**, 183–212.

Morris WF, Doak DF (2002). *Quantitative Conservation Biology: Theory and Practice of Population Viability Analysis*. Sinauer, Sunderland, MA.

Pinheiro JC, Bates DM (2000). *Mixed Effects Models in S and S-PLUS*. Springer-Verlag, New York.

R Development Core Team (2007). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

The MathWorks, Inc (2007). *MATLAB – The Language of Technical Computing, Version 7.5*. The MathWorks, Inc., Natick, Massachusetts. URL http://www.MathWorks.com/products/matlab/.

Tuljapurkar S (1990). *Population Dynamics in Variable Environments*. Lecture Notes in Biomathematics 85. Springer-Verlag, New York.

Tuljapurkar S, Horvitz CC (2006). "From Stage to Age in Variable Environments: Life Expectancy and Survivorship." *Ecology*, **87**(6), 1497–1509.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S-PLUS*. Springer-Verlag, New York, fourth edition.

Affiliation:

Chris Stubben
Los Alamos National Lab
BioScience Division
Los Alamos, New Mexico 87545, United States of America
E-mail: stubben@lanl.gov

Brook Milligan
Department of Biology
New Mexico State University
Las Cruces, New Mexico 88003, United States of America
E-mail: brook@nmsu.edu

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 22, Issue 11 *Submitted:* 2007-02-16
September 2007 *Accepted:* 2007-08-28