lefko3: a gentle introduction

Creating and analyzing matrix projection models in R

Richard P. Shefferson

2022-04-16

Chapter 9

Population Projection II: Temporal Environmental Stochasticity

Each of us a cell of awareness
imperfect and incomplete
Genetic blends with uncertain ends
on a fortune hunt that's far too fleet

— Rush, *Freewill* (1980)

Stochasticity refers to randomness. In population ecology, stochasticity is studied in two dominant forms: environmental and demographic (Caswell, 2001). The former refers to temporal randomness in the environment leading to random changes in the projection matrix over time via changes affecting the entire population. This randomness can be temporal or spatial, or both. The latter refers to random divergence in vital rates caused by the differing fates of individuals within populations of finite size. In this chapter, we will concern ourselves only with the former kind of stochasticity.

Most of the literature on environmental stochasticity in matrix projection analysis is focused on randomness with respect to time. Temporal randomness implies that the choice of matrix for a particular point in time, or the changes that occur in the matrix in that time, do not depend on any other time. Thus, the matrix used for projection at some point in time is drawn at random from a distribution of such matrices, which are all independent and identically distributed (Caswell, 2001). Alternatively, the kernel used to populate the matrices has a set of terms that vary across time, and these terms are shuffled randomly at each time step. Thus, the choice of matrix is not predictable - knowing the composition of the matrix at one point in time tells you nothing of its composition at any other point in time.

How exactly does such randomness change the projection? In comparison to equation 8.1, which shows a deterministic projection, an environmental stochastic projection is given as:

$$\mathbf{n_t} = \mathbf{A_t A_{t-1}} \cdots \mathbf{A_1 n_0} \tag{9.1}$$

where each $\mathbf{A_i}$ is a randomly sampled matrix, each of which has the same dimensions. If we project the population forward a large number of time steps, then we generally find certain stable conditions forming in the long run. These stable conditions are predicted by the strong and weak ergodic theorems (Caswell, 2001).

Package `lefko3` includes a number of functions to facilitate matrix projection analysis assuming temporal stochasticity (we shall refer to this in general as *stochastic analysis* from now on). Currently, we include functions to estimate stochastic population growth rate, long-run stage structure, long-run reproductive value, stochastic sensitivity and elasticity, and stochastic life table response experiments. Two projection functions are also included that can be used to assess more interesting questions.

Let's utilize the raw MPMs developed in chapter 4 to illustrate the sorts of stochastic analyses that we can perform. These are the same matrices utilized in the last chapter (chapter 8). Particularly, we will use the lefkoMat objects named `cypmatrix2r`, `cypmatrix2rp`, `cypmatrix3r`, and `cypmatrix3rp`. However, the function-based MPMs, IPMs, and age-by-stage MPMs that we have created can also be used in all cases.

Let's start off by taking a look at summaries of these objects.

```
summary(cypmatrix2r)
>
> This ahistorical lefkoMat object contains 5 matrices.
>
> Each matrix is square with 11 rows and columns, and a total of 121 elements.
> A total of 120 survival transitions were estimated, with 24 per matrix.
> A total of 40 fecundity transitions were estimated, with 8 per matrix.
> This lefkoMat object covers 1 population, 1 patch, and 5 time steps.
>
> The dataset contains a total of 74 unique individuals and 320 unique transitions.
>
> Survival probability sum check (each matrix represented by column in order):
>          [,1]  [,2]  [,3]  [,4]  [,5]
> Min.    0.000 0.050 0.050 0.000 0.050
> 1st Qu. 0.100 0.140 0.140 0.100 0.140
> Median  0.689 0.870 0.864 0.610 0.882
> Mean    0.552 0.629 0.629 0.528 0.627
> 3rd Qu. 1.000 1.000 1.000 0.960 1.000
> Max.    1.000 1.000 1.000 1.000 1.000
summary(cypmatrix2rp)
>
> This ahistorical lefkoMat object contains 15 matrices.
>
> Each matrix is square with 11 rows and columns, and a total of 121 elements.
> A total of 266 survival transitions were estimated, with 17.733 per matrix.
> A total of 70 fecundity transitions were estimated, with 4.667 per matrix.
> This lefkoMat object covers 1 population, 3 patches, and 5 time steps.
>
> The dataset contains a total of 74 unique individuals and 320 unique transitions.
>
> Survival probability sum check (each matrix represented by column in order):
>          [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
> Min.    0.000 0.000 0.000 0.000 0.000 0.000 0.050 0.050 0.000 0.050 0.000 0.000
> 1st Qu. 0.075 0.025 0.075 0.025 0.075 0.075 0.140 0.140 0.100 0.140 0.100 0.100
> Median  0.180 0.100 0.180 0.100 0.180 0.180 0.909 0.778 0.686 0.857 0.750 0.575
> Mean    0.457 0.361 0.471 0.328 0.417 0.464 0.631 0.611 0.530 0.631 0.562 0.523
> 3rd Qu. 0.955 0.769 1.000 0.592 0.781 1.000 1.000 1.000 0.955 1.000 1.000 1.000
> Max.    1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
>         [,13] [,14] [,15]
> Min.    0.000 0.000 0.000
> 1st Qu. 0.075 0.075 0.100
> Median  0.180 0.180 0.750
> Mean    0.432 0.450 0.562
> 3rd Qu. 0.875 1.000 1.000
> Max.    1.000 1.000 1.000
```

```
summary(cypmatrix3r)
>
> This historical lefkoMat object contains 4 matrices.
>
> Each matrix is square with 121 rows and columns, and a total of 14641 elements.
> A total of 242 survival transitions were estimated, with 60.5 per matrix.
> A total of 52 fecundity transitions were estimated, with 13 per matrix.
> This lefkoMat object covers 1 population, 1 patch, and 4 time steps.
>
> The dataset contains a total of 74 unique individuals and 320 unique transitions.
>
> Survival probability sum check (each matrix represented by column in order):
>          [,1]  [,2]  [,3]  [,4]
> Min.    0.000 0.000 0.000 0.000
> 1st Qu. 0.000 0.000 0.000 0.000
> Median  0.000 0.000 0.000 0.000
> Mean    0.173 0.179 0.166 0.198
> 3rd Qu. 0.100 0.100 0.100 0.100
> Max.    1.000 1.000 1.000 1.000
summary(cypmatrix3rp)
>
> This historical lefkoMat object contains 12 matrices.
>
> Each matrix is square with 121 rows and columns, and a total of 14641 elements.
> A total of 516 survival transitions were estimated, with 43 per matrix.
> A total of 68 fecundity transitions were estimated, with 5.667 per matrix.
> This lefkoMat object covers 1 population, 3 patches, and 4 time steps.
>
> The dataset contains a total of 74 unique individuals and 320 unique transitions.
>
> Survival probability sum check (each matrix represented by column in order):
>          [,1]   [,2]   [,3]  [,4]  [,5]  [,6] [,7]  [,8]  [,9]  [,10] [,11]
> Min.    0.000 0.0000 0.0000 0.000 0.000 0.000 0.00 0.000 0.000 0.0000 0.000
> 1st Qu. 0.000 0.0000 0.0000 0.000 0.000 0.000 0.00 0.000 0.000 0.0000 0.000
> Median  0.000 0.0000 0.0000 0.000 0.000 0.000 0.00 0.000 0.000 0.0000 0.000
> Mean    0.107 0.0945 0.0851 0.101 0.158 0.158 0.14 0.169 0.119 0.0851 0.119
> 3rd Qu. 0.000 0.0000 0.0000 0.000 0.100 0.100 0.05 0.100 0.000 0.0000 0.000
> Max.    1.000 1.0000 1.0000 1.000 1.000 1.000 1.00 1.000 1.000 1.0000 1.000
>         [,12]
> Min.    0.000
> 1st Qu. 0.000
> Median  0.000
> Mean    0.144
> 3rd Qu. 0.000
> Max.    1.000
```

We see that these four MPMs cover three patches of one population, and that `cypmatrix2r` and `cypmatrix3r` do not distinguish the patches while `cypmatrix2rp` and `cypmatrix3rp` do. Objects `cypmatrix2r` and `cypmatrix2rp` are ahistorical, while `cypmatrix3r` and `cypmatrix3rp` are historical. Because these are raw MPMs and there are six total monitoring occasions covered, we see five matrices estimated in the ahistorical population-level set and four matrices estimated in the historical population-level set. The patch-level MPMs create five and four matrices per patch for the ahistorical

and historical cases, respectively. Finally, there are 11 life history stages, which we see below.

```
cypmatrix2r$ahstages
>    stage_id stage original_size original_size_b original_size_c min_age max_age
> 1         1    SD           0.0              NA              NA       0      NA
> 2         2    P1           0.0              NA              NA       0      NA
> 3         3    P2           0.0              NA              NA       0      NA
> 4         4    P3           0.0              NA              NA       0      NA
> 5         5    SL           0.0              NA              NA       0      NA
> 6         6     D           0.0              NA              NA       0      NA
> 7         7   XSm           1.0              NA              NA       0      NA
> 8         8    Sm           3.0              NA              NA       0      NA
> 9         9    Md           6.0              NA              NA       0      NA
> 10       10    Lg          11.0              NA              NA       0      NA
> 11       11   XLg          19.5              NA              NA       0      NA
>    repstatus obsstatus propstatus immstatus matstatus entrystage indataset
> 1          0         0          1         0         0          1         0
> 2          0         0          0         1         0          1         0
> 3          0         0          0         1         0          0         0
> 4          0         0          0         1         0          0         0
> 5          0         0          0         1         0          0         0
> 6          0         0          0         0         1          0         1
> 7          1         1          0         0         1          0         1
> 8          1         1          0         0         1          0         1
> 9          1         1          0         0         1          0         1
> 10         1         1          0         0         1          0         1
> 11         1         1          0         0         1          0         1
>    binhalfwidth_raw sizebin_min sizebin_max sizebin_center sizebin_width
> 1               0.0         0.0         0.0            0.0             0
> 2               0.0         0.0         0.0            0.0             0
> 3               0.0         0.0         0.0            0.0             0
> 4               0.0         0.0         0.0            0.0             0
> 5               0.0         0.0         0.0            0.0             0
> 6               0.5        -0.5         0.5            0.0             1
> 7               0.5         0.5         1.5            1.0             1
> 8               1.5         1.5         4.5            3.0             3
> 9               1.5         4.5         7.5            6.0             3
> 10              3.5         7.5        14.5           11.0             7
> 11              5.0        14.5        24.5           19.5            10
>    binhalfwidthb_raw sizebinb_min sizebinb_max sizebinb_center sizebinb_width
> 1                 NA           NA           NA              NA             NA
> 2                 NA           NA           NA              NA             NA
> 3                 NA           NA           NA              NA             NA
> 4                 NA           NA           NA              NA             NA
> 5                 NA           NA           NA              NA             NA
> 6                 NA           NA           NA              NA             NA
> 7                 NA           NA           NA              NA             NA
> 8                 NA           NA           NA              NA             NA
> 9                 NA           NA           NA              NA             NA
> 10                NA           NA           NA              NA             NA
> 11                NA           NA           NA              NA             NA
>    binhalfwidthc_raw sizebinc_min sizebinc_max sizebinc_center sizebinc_width
```

```
> 1                NA           NA           NA               NA               NA
> 2                NA           NA           NA               NA               NA
> 3                NA           NA           NA               NA               NA
> 4                NA           NA           NA               NA               NA
> 5                NA           NA           NA               NA               NA
> 6                NA           NA           NA               NA               NA
> 7                NA           NA           NA               NA               NA
> 8                NA           NA           NA               NA               NA
> 9                NA           NA           NA               NA               NA
> 10               NA           NA           NA               NA               NA
> 11               NA           NA           NA               NA               NA
>    group                        comments alive almostborn
> 1      0                    Dormant seed     1          0
> 2      0                1st yr protocorm     1          0
> 3      0                2nd yr protocorm     1          0
> 4      0                3rd yr protocorm     1          0
> 5      0                         Seedling     1          0
> 6      0                   Dormant adult     1          0
> 7      0    Extra small adult (1 shoot)     1          0
> 8      0       Small adult (2-4 shoots)     1          0
> 9      0      Medium adult (5-7 shoots)     1          0
> 10     0      Large adult (8-14 shoots)     1          0
> 11     0 Extra large adult (>14 shoots)     1          0
```

Let's move on now to the first of the stochastic analyses: the estimation of the stochastic population growth rate, a.

9.1 Stochastic population growth rate

Function `slambda3()` estimates the log stochastic population growth rate in its instantaneous form ($a = \log\lambda_S$). This is estimated as the mean log discrete population growth rate across a user-specified number of random draws of the annual matrices provided as input in the function, as in

$$a = \log\lambda_S = \frac{1}{T}\sum_{t=1}^{T}\log\frac{N_t}{N_{t-1}} \tag{9.2}$$

where N_t is the population size in occasion t, and T is the number of occasions projected forward, set to 10,000 by default. Function `slambda3()` does not shuffle across patches or populations, instead shuffling within patches, or shuffling annual matrices calculated as element-wise means of patch matrices within the same population and the same time. The methodology is based on Morris and Doak (2002), though accounts for spatial averaging of patches and can easily handle large and sparse matrices.

Let's estimate the stochastic population growth rate. We will use `set.seed()` prior to each run to make sure that we get the same result as what you will see (function `set.seed()` allows us to set the seed for the pseudo-random number generator in this function, thus yielding the same "random" sequence of matrix draws).

```
writeLines("Raw ahistorical stochastic log lambda:")
> Raw ahistorical stochastic log lambda:
set.seed(42)
cypmatrix2rp_slam <- slambda3(cypmatrix2rp)
cypmatrix2rp_slam
```

```
>   pop patch           a       var        sd          se
> 1   1     A -0.03349084 4.2171941 2.0535808 0.020535808
> 2   1     B  0.09728146 1.3439679 1.1592963 0.011592963
> 3   1     C  0.06638145 1.1461344 1.0705766 0.010705766
> 4   1     0  0.04221328 0.7354604 0.8575899 0.008575899
writeLines("\nRaw historical stochastic log lambda:")
>
> Raw historical stochastic log lambda:
set.seed(42)
cypmatrix3rp_slam <- slambda3(cypmatrix3rp)
cypmatrix3rp_slam
>   pop patch           a      var        sd          se
> 1   1     A -0.18490427 6.302481 2.5104742 0.025104742
> 2   1     B -0.07213781 2.409698 1.5523202 0.015523202
> 3   1     C -0.05559008 2.806816 1.6753555 0.016753555
> 4   1     0 -0.12650801 0.847868 0.9207975 0.009207975
```

The output above is for the ahistorical and historical patch-level MPMs. The final entry in each case, labeled as patch `0`, corresponds to the population (where the population MPM is one in which the matrix for each year is an equally weighted mean of each patch's matrix for that year). Unlike in deterministic analysis, population growth rate is not estimated for each matrix, but rather for each patch or population. This is because the growth rate is estimated by a random shuffle of the matrices, with the stochastic growth rate equal to the average log change in projected population size in the latter portion of the shuffled set. Since we did not change any of the defaults, R shuffled the matrices 10,000 times, and the resulting mean log stochastic growth rate and its associated variance, standard deviation, and standard error are calculated over these 10,000 times in the projection.

Users will notice differences between ahistorical and historical $\log\lambda_S$ values. Ahistorical population growth rate estimates are often higher than those for historical MPMs. The reasons are not yet clear, but they may have to do with the life histories that have been used in analysis so far - population growth in long-lived herbaceous perennial plants is typically driven by survival transitions in adults, and the impact of fecundity on growth rate appears even lower when individual history is incorporated into analysis. This may also be due to long-term trade-offs captured within the historical models but not within ahistorical models. Of note is the growth trade-off - we have found that large individuals growing to large size in time t from small size in time t-1 have lower survival probability to time t+1 (Shefferson et al., 2014). Finally, the data are cut more finely to determine elements in the historical case, leading to a greater influence of artificial zeros (some zeros are artificial in the sense that they are zero simply because the dataset is too small to yield individuals moving through particular transitions).

It may be interesting in this case to compare our estimates of the population growth rate to the deterministic case. To compare them, we must first estimate the deterministic population growth rate λ for the temporal mean matrices. Then, we need either to take the natural logarithm of λ, or to raise the natural number e to the power of a given in the `slambda3()` output. Let's do the latter. We will put our results together into a data frame using the `cbind.data.frame()` function, as below.

```
cyp2rp_mean <- lmean(cypmatrix2rp)
cyp3rp_mean <- lmean(cypmatrix3rp)
cyp2rp_mean_lam <- lambda3(cyp2rp_mean)
cyp3rp_mean_lam <- lambda3(cyp3rp_mean)
cypmatrix2rp_slam$expa <- exp(cypmatrix2rp_slam$a)
cypmatrix3rp_slam$expa <- exp(cypmatrix3rp_slam$a)

lambda_comparison <- cbind.data.frame(pop = cyp2rp_mean_lam$pop,
  patch = cyp2rp_mean_lam$patch, lambda_ah = cyp2rp_mean_lam$lambda,
```

```
  lambda_h = cyp3rp_mean_lam$lambda, expa_ah = cypmatrix2rp_slam$expa,
  expa_h = cypmatrix3rp_slam$expa)
lambda_comparison
>   pop patch lambda_ah  lambda_h   expa_ah    expa_h
> 1   1     A 0.9835717 0.8510445 0.9670638 0.8311838
> 2   1     B 1.1051470 0.9481901 1.1021705 0.9304027
> 3   1     C 1.0717513 0.9671030 1.0686343 0.9459268
> 4   1     0 1.0450267 0.8937376 1.0431169 0.8811671
```

Temporal variation in fitness leads to lower long-term fitness, and we see this in the general pattern of lower stochastic population growth rate than deterministic growth rate. There also appears to be a general trend of historical matrices yielding lower deterministic and stochastic population growth rates.

9.2 Long-run average stage distribution and reproductive value

Package `lefko3` also handles the estimation of stochastic long-run stage distribution and long-run reproductive value vectors, which are the stochastic equivalents of the deterministic stable stage equilibrium and asymptotic reproductive value vector. It handles this through the projection of shuffled temporal matrices, typically 10,000 occasions forward though the exact number can be set as needed. According to the stochastic strong and weak ergodic theorems, this random shuffling should eventually yield a roughly stationary distribution of stage proportions and stage reproductive values. Thus, the stochastic stable structure is estimated as the arithmetic mean vector of the predicted stage proportion vectors across the final portion of such a projection (in a 10,000 time step projection, this is typically the final 1000 or so time steps).

The stochastic long-run reproductive value is more complicated. Imagine a vector, $\mathbf{x}(t)$, which includes the expected number of offspring produced by each stage in occasion t+1. This is referred to as the *undiscounted reproductive value vector* because it is not standardized (Caswell, 2001), and so is likely to eventually move to an extreme value such as infinity. Vector $\mathbf{x}(t)$ is related to other occasions in this projection, as in:

$$\mathbf{x}^{\top}(t) = \mathbf{x}^{\top}(t+1)\mathbf{A}_t \tag{9.3}$$

Standardizing leads to the *discounted reproductive value vector*, given as

$$\mathbf{v}(t) = \frac{\mathbf{x}(t)}{\|\mathbf{x}(t)\|} \tag{9.4}$$

This reproductive value vector can then be projected as well, as in:

$$\mathbf{v}(t) = \frac{\mathbf{v}^{\top}(t+1)\mathbf{A}_t}{\|\mathbf{v}^{\top}(t+1)\mathbf{A}_t\|} \tag{9.5}$$

The reproductive value vector is projected backwards in time, and under the default 10,000 time steps, package `lefko3` takes its average in the final 1000 steps of the backward projection (smaller numbers of steps are used in cases where the projection is limited to less than 2000 steps).

Now let's take a look at the long-run stage distributions at the population level, comparing the deterministic to the stochastic, and the ahistorical to the historical. We will focus only on the population-level MPMs (note that we need to create the mean matrices for the deterministic analysis of the population-level MPMs, and do so within the first two lines). We will first create the stable stage vectors, and then put them all into a common data frame, as below.

```
cyp2r_mean <- lmean(cypmatrix2r)
cyp3r_mean <- lmean(cypmatrix3r)

tm2ss_r <- stablestage3(cyp2r_mean)
tm3ss_r <- stablestage3(cyp3r_mean)
tm2ss_rs <- stablestage3(cypmatrix2r, stochastic = TRUE, seed = 42)
tm3ss_rs <- stablestage3(cypmatrix3r, stochastic = TRUE, seed = 42)

ss_put_together <- cbind.data.frame(tm2ss_r$ss_prop, tm3ss_r$ahist$ss_prop,
  tm2ss_rs$ss_prop, tm3ss_rs$ahist$ss_prop)
names(ss_put_together) <- c("det ahist", "det hist", "sto ahist", "sto hist")
rownames(ss_put_together) <- tm2ss_r$stage

ss_put_together
>        det ahist     det hist    sto ahist     sto hist
> SD  4.709506e-01 4.730767e-01 4.568960e-01 4.559082e-01
> P1  4.796543e-01 4.739860e-01 4.696325e-01 4.577532e-01
> P2  4.432261e-02 4.692223e-02 6.502990e-02 7.404003e-02
> P3  4.095645e-03 4.645066e-03 6.863520e-03 9.448352e-03
> SL  1.983961e-04 2.418922e-04 3.354917e-04 5.469420e-04
> D   4.309986e-05 8.748103e-05 9.244503e-05 2.550369e-04
> XSm 2.759814e-04 4.850711e-04 4.249543e-04 9.431349e-04
> Sm  3.049081e-04 4.222049e-04 4.652088e-04 8.011253e-04
> Md  9.705559e-05 1.142297e-04 1.714340e-04 2.648440e-04
> Lg  4.938104e-05 1.806557e-05 7.269498e-05 3.680005e-05
> XLg 8.078062e-06 1.053848e-06 1.583885e-05 2.255676e-06
```

Now we will compare the stage distributions via a plot (figure 9.1).

```
ss_put_together <- cbind.data.frame(tm2ss_r$ss_prop, tm3ss_r$ahist$ss_prop,
  tm2ss_rs$ss_prop, tm3ss_rs$ahist$ss_prop)
names(ss_put_together) <- c("det ahist", "det hist", "sto ahist", "sto hist")
rownames(ss_put_together) <- tm2ss_r$stage

barplot(t(ss_put_together), beside=T, ylab = "Proportion", xlab = "Stage",
  ylim = c(0, 0.95), col = c("black", "orangered", "grey", "darkred"),
  bty = "n")
legend("topright", c("det ahist", "det hist", "sto ahist", "sto hist"),
  col = c("black", "orangered", "grey", "darkred"), pch = 15, cex = 0.9,
  bty = "n")
```

Overall, these are generally similar patterns. However, deterministic analyses suggest a greater share of the population should be composed of dormant seeds and 1st year protocorms, and a lower share should be composed of 2nd year protocorms protocorms.

Let's look at stochastic vs. deterministic reproductive values from ahistorical and historical approaches. First let's calculate the reproductive values under each scenario.

```
tm2rv_r <- repvalue3(cyp2r_mean)
tm3rv_r <- repvalue3(cyp3r_mean)
tm2rv_rs <- repvalue3(cypmatrix2r, stochastic = TRUE, seed = 42)
tm3rv_rs <- repvalue3(cypmatrix3r, stochastic = TRUE, seed = 42)
```

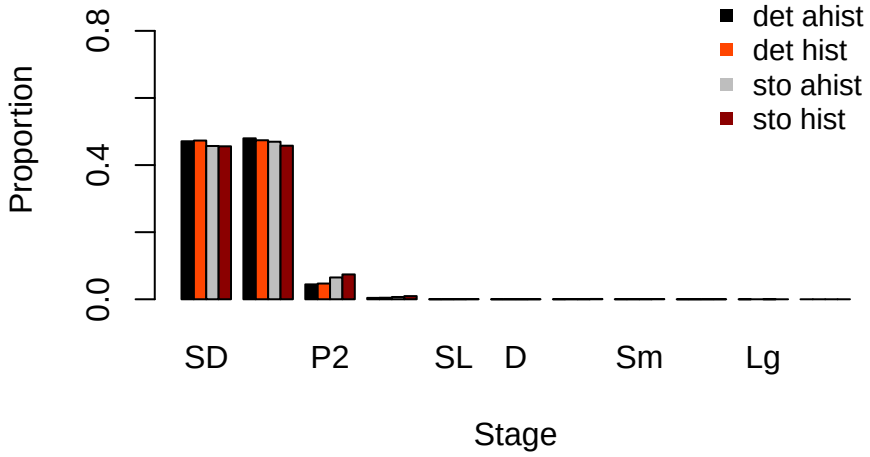

Figure 9.1: Ahistorical vs. historically-corrected stable and long-run mean stage distribution

```
tm2rv_rplot <- as.matrix(tm2rv_r$rep_value)
rownames(tm2rv_rplot) <- tm2rv_r$stage
tm2rv_rsplot <- as.matrix(tm2rv_rs$rep_value)
rownames(tm2rv_rsplot) <- tm2rv_rs$stage
tm3rv_rplot <- as.matrix(tm3rv_r$ahist$rep_value)
rownames(tm3rv_rplot) <- tm3rv_r$ahist$stage
tm3rv_rsplot <- as.matrix(tm3rv_rs$ahist$rep_value)
rownames(tm3rv_rsplot) <- tm3rv_rs$ahist$stage
```

Now let's compare them (figure 9.2).

```
par(mfrow=c(2,2))
barplot(t(tm2rv_rplot), ylab = "Rep value", xlab = "", col = "black", bty = "n")
title("a)", adj = 0)
legend("topleft", c("det ahist", "det hist", "sto ahist", "sto hist"), cex =0.8,
  col = c("black", "orangered", "grey", "darkred"), pch = 15, bty = "n")
barplot(t(tm3rv_rplot), ylab = "", xlab = "", col = "orangered", bty = "n")
title("b)", adj = 0)
barplot(t(tm2rv_rsplot), ylab = "Rep value", xlab = "Stage", col = "grey",
  bty = "n")
title("c)", adj = 0)
barplot(t(tm3rv_rsplot), ylab = "", xlab = "Stage", col = "darkred", bty = "n")
title("d)", adj = 0)
```

We see some big differences here, particularly between ahistorical and historical analyses. Indeed, in the ahistorical case, reproductive value increases with size, reaching its peak in the largest adults. In contrast, in the historical case, the greatest reproductive values are associated with medium adults. So, history appears to have large effects here, while temporal stochasticity does not seem to have much influence.

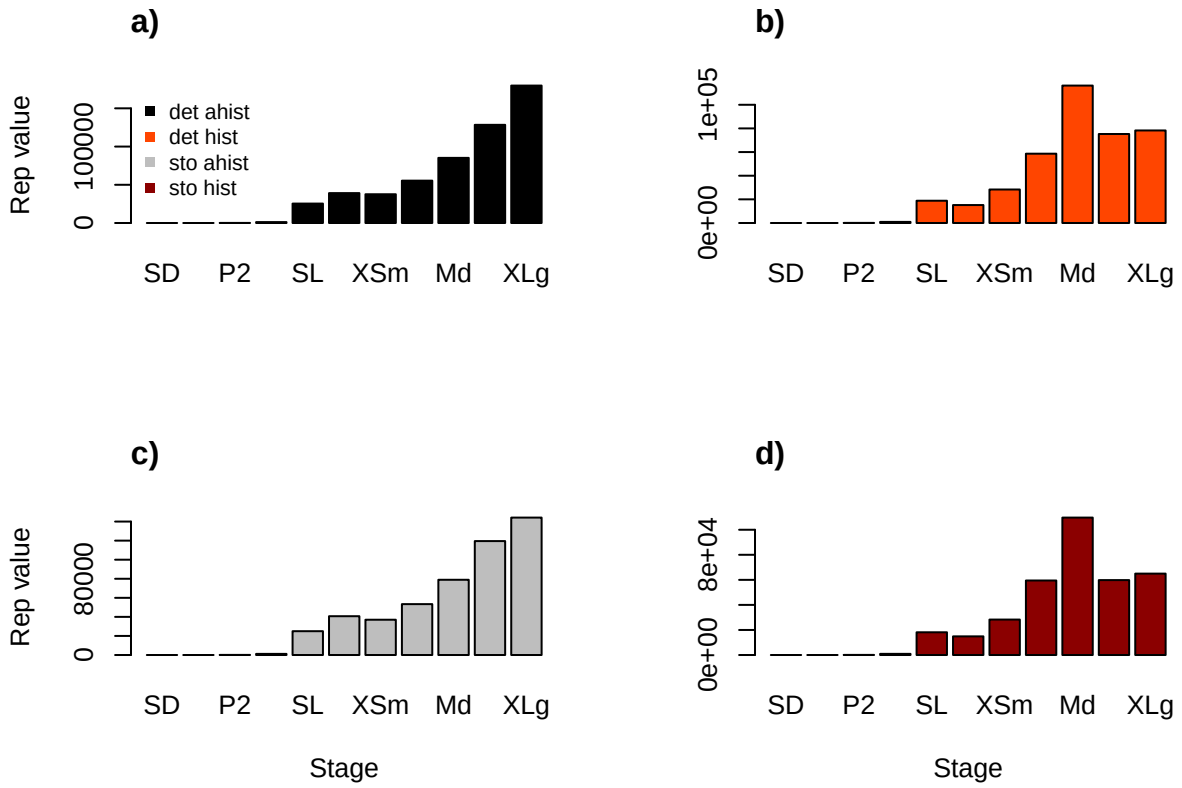

Figure 9.2: Ahistorical (a, c) vs. historically-corrected (b, d) deterministic (a, b) and stochastic (c, d) reproductive values

9.3 Stochastic sensitivity analysis

Package `lefko3` contains functions allowing users to conduct deterministic and stochastic sensitivity and elasticity analyses. Sensitivity and elasticity analysis are forms of perturbation analysis, and we urge readers to consult Caswell (2001) and Caswell (2019) to become fully acquainted with the topic. Here, we discuss just the most important aspects to understand these analyses as conducted using `lefko3`.

Stochastic sensitivity analysis assesses the impact of small, absolute changes in matrix elements on the log stochastic growth rate, $a = \log\lambda_S$ (Caswell, 2001). The stochastic sensitivity of a to an element a_{kj} is

$$\frac{\partial \log\lambda_S}{\partial a_{kj}} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \frac{\mathbf{v}(t+1)\mathbf{w}^\top(t)}{\mathbf{v}^\top(t+1)\mathbf{w}(t+1)} \tag{9.6}$$

where t refers to a specific occasion and T refers to the number of occasions projected. Stochastic sensitivities for hMPMs may be converted to historically-corrected format as in the deterministic case.

Let's now try a stochastic sensitivity analysis. We'll start off by running the ahistorical MPM.

```
set.seed(42)
tm2sens_rs <- sensitivity3(cypmatrix2r, stochastic = TRUE)

writeLines("\nThe highest stochastic sensitivity value: ")
>
> The highest stochastic sensitivity value:
max(tm2sens_rs$ah_sensmats[[1]][which(cyp2r_mean$A[[1]] > 0)])
> [1] 1.468996
writeLines("\nThis value is associated with element: ")
>
> This value is associated with element:
intersect(which(tm2sens_rs$ah_sensmats[[1]] ==
    max(tm2sens_rs$ah_sensmats[[1]][which(cyp2r_mean$A[[1]] > 0)])),
    which(cyp2r_mean$A[[1]] > 0))
> [1] 38
```

The highest sensitivity value is associated with element 38. Since there are 11 stages, and the element number refers to the position of the element within a vectorized form of the matrix, we can infer that $log\lambda$ is most sensitive to the transition from 3rd year protocorm to seedling. Check this by typing `ceiling(38/11)` to get the column number and `38 %% 11` to get the row number for element 38.

```
writeLines(paste0("From stage: ",
  cypmatrix2r$ahstages[ceiling(38/dim(cypmatrix2r$ahstages)[1]),"stage"], "\n"))
> From stage: P3
writeLines(paste0("To stage: ",
  cypmatrix2r$ahstages[38 %% dim(cypmatrix2r$ahstages)[1],"stage"], "\n"))
> To stage: SL
```

Let's compare this to the historical case.

```
set.seed(42)
tm3sens_rs <- sensitivity3(cypmatrix3r, stochastic = TRUE)

writeLines("\nThe highest stochastic sensitivity value: ")
```

```
>
> The highest stochastic sensitivity value:
max(tm3sens_rs$h_sensmats[[1]][which(cyp3r_mean$A[[1]] > 0)])
> [1] 1.169812
writeLines("\nThis value is associated with element: ")
>
> This value is associated with element:
intersect(which(tm3sens_rs$h_sensmats[[1]] ==
    max(tm3sens_rs$h_sensmats[[1]][which(cyp3r_mean$A[[1]] > 0)])),
    which(cyp3r_mean$A[[1]] > 0))
> [1] 3063
```

Here we find that $log\lambda$ is most sensitive to element 3,063. We can use the same trick as with the ahistorical case to determine which transition this is, but we need to use the `hstages` portion of the historical MPM rather than the `ahstages` element. The `hstages` element outlines the 121 stage-pairs corresponding to the rows and column in these matrices. So, let's figure this out, as below.

```
writeLines(paste0("From stage in t-1: ",
  cypmatrix3r$hstages[ceiling(3063/dim(cypmatrix3r$hstages)[1]),
    "stage_1"], "\n"))
> From stage in t-1: P2
writeLines(paste0("From stage in t: ",
  cypmatrix3r$hstages[ceiling(3063/dim(cypmatrix3r$hstages)[1]), "stage_2"],
  "\n"))
> From stage in t: P3
writeLines(paste0("To stage in t: ",
  cypmatrix3r$hstages[3063 %% dim(cypmatrix3r$hstages)[1],"stage_1"], "\n"))
> To stage in t: P3
writeLines(paste0("To stage in t+1: ",
  cypmatrix3r$hstages[3063 %% dim(cypmatrix3r$hstages)[1],"stage_2"], "\n"))
> To stage in t+1: SL
```

This transition corresponds to the transition from the 26th stage pair (2nd year and 3rd year protocorms in times t-1 and t, respectively), to the 38th stage pair (3rd year protocorm and seedling in times t and t+1, respectively). This is similar to our results from the deterministic sensitivity analysis in the last chapter (chapter 8).

9.4 Stochastic elasticity analysis

Elasticity analyses assess the impacts of small, proportional changes in matrix elements on population growth rate. In stochastic elasticity analysis, the population growth rate used is the log stochastic growth rate, $a = \log\lambda_S$ (Caswell, 2001). The stochastic elasticity of $a = \log\lambda_S$ to changes in an element is given as

$$\frac{\partial \log\lambda_S}{\partial \log a_{kj}} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \frac{(\mathbf{v}(t+1)\mathbf{w}^\top(t)) \circ \mathbf{A_t}}{\mathbf{v}^\top(t+1)\mathbf{w}(t+1)} \tag{9.7}$$

where $\mathbf{A_t}$ refers to the A matrix corresponding to occasion t. Stochastic elasticity values for hMPMs may be converted to historically-corrected format as in the deterministic case, and may also be summed as before. Elements estimated to be exactly zero must also have elasticity values equal to zero, making their interpretation even more different than sensitivity values.

Let's assess the elasticity of $a = \log\lambda_S$ to matrix elements, comparing the ahistorical to the historically-corrected case in stochastic analyses.

```
set.seed(42)
tm2elas_rs <- elasticity3(cypmatrix2r, stochastic = TRUE)
set.seed(42)
tm3elas_rs <- elasticity3(cypmatrix3r, stochastic = TRUE)

writeLines("\nMax ahistorical stoch elasticity occurs in element: ")
>
> Max ahistorical stoch elasticity occurs in element:
which(tm2elas_rs$ah_elasmats[[1]] == max(tm2elas_rs$ah_elasmats[[1]]))
> [1] 85
writeLines("\nMax historically-corrected stoch elasticity occurs in element: ")
>
> Max historically-corrected stoch elasticity occurs in element:
which(tm3elas_rs$ah_elasmats[[1]] == max(tm3elas_rs$ah_elasmats[[1]]))
> [1] 85
writeLines("\nMax historical stoch elasticity occurs in element: ")
>
> Max historical stoch elasticity occurs in element:
which(tm3elas_rs$h_elasmats[[1]] == max(tm3elas_rs$h_elasmats[[1]]))
> [1] 10249
```

The ahistorical and historical analyses generally agree that $\log\lambda_S$ is most elastic to the ahistorical stasis transition from Small adult in time t to Small adult in time t+1, and the historical stasis transition as Small adult in times t-1, t, and t+1. This is a different result from sensitivity analysis, where the ahistorical and historical stochastic sensitivity analyses suggested that population growth rate was most sensitivity to the transition from 3rd year protocorm to seedling (with 2nd year protocorm in time t-1 in the historical case).

Let's compare the elasticity of population growth to life history stages (figure 9.3).

```
elas_put_together <- cbind.data.frame(colSums(tm2elas_rs$ah_elasmats[[1]]),
  colSums(tm3elas_rs$ah_elasmats[[1]]))
names(elas_put_together) <- c("sto ahist", "sto hist")
rownames(elas_put_together) <- tm2elas_rs$ah_stages$stage

barplot(t(elas_put_together), beside=T, ylab = "Elasticity", xlab = "Stage",
  col = c("grey", "darkred"), ylim = c(0, 0.50), bty = "n")
legend("topright", c("sto ahist", "sto hist"), col = c("grey", "darkred"),
  pch = 15, bty = "n")
```

Elasticity analyses in these plots look rather similar, though with some key differences. While both ahistorical and historical analyses show that $\log\lambda_S$ should be most elastic in response to shifts in transitions associated with small adults, historical analyses suggest a much stronger elasticity of small and medium adults and a much lower contribution of large and extra large adults.

Finally, let's look at the elasticity sums of different transition types (figure 9.4).

```
tm2elas_rs_sums <- summary(tm2elas_rs)
tm3elas_rs_sums <- summary(tm3elas_rs)

elas_sums_together <- cbind.data.frame(tm2elas_rs_sums$ahist[,2],
```

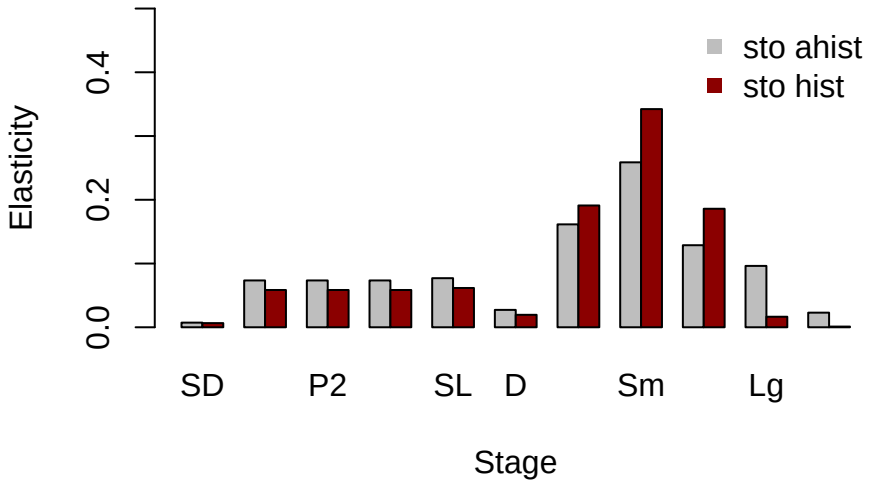

Figure 9.3: Ahistorical vs. historically-corrected stochastic elasticity to stage

```
  tm3elas_rs_sums$ahist[,2])
names(elas_sums_together) <- c("sto ahist", "sto hist")
rownames(elas_sums_together) <- tm2elas_rs_sums$ahist$category

barplot(t(elas_sums_together), beside=T, ylab = "Elasticity",
  xlab = "Transition", col = c("grey", "darkred"), ylim = c(0, 0.60), bty = "n")
legend("topright", c("sto ahist", "sto hist"), col = c("grey", "darkred"),
  pch = 15, bty = "n")
```

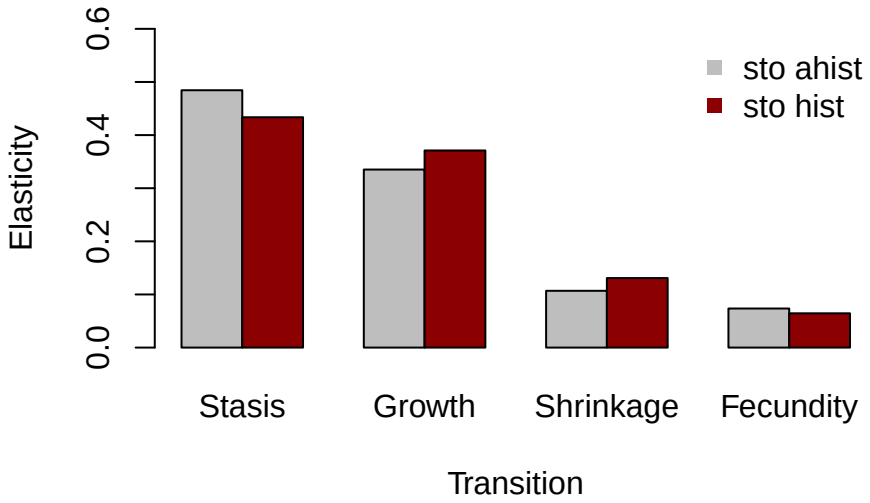

Figure 9.4: Ahistorical vs. historically-corrected elasticity of a to transitions

Fecundity makes least difference in all cases, although it is nearly as influential as shrinkage. Stasis is among the most important transition types across the board. Next, let's see the elasticity values for historical transitions (figure 9.5).

```
elas_hist2plot <- as.matrix(tm3elas_rs_sums$hist[,2])
rownames(elas_hist2plot) <- tm3elas_rs_sums$hist$category

par(mar = c(7, 4, 2, 2) + 0.2)
barplot(t(elas_hist2plot), ylab = "Elasticity", xlab = "", xaxt = "n",
  col = c("orangered", "darkred"), bty = "n")
text(cex=0.6, x=seq(from = 0, to = 1.1*length(tm3elas_rs_sums$hist$category),
  by = 1.15), y=-0.08, tm3elas_rs_sums$hist$category, xpd=TRUE, srt=45)
```

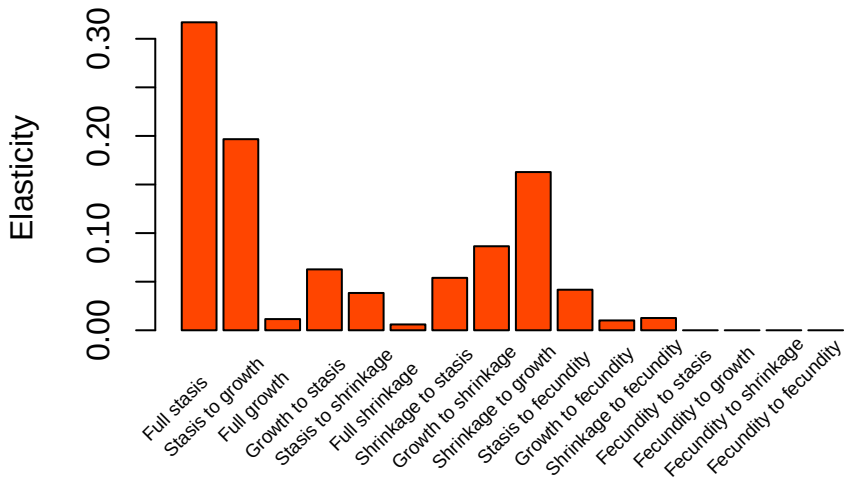

Figure 9.5: Elasticity of a to historical transitions

We can see that full stasis across occasions t-1, t, and t+1 is associated with the greatest summed elasticity, followed by stasis to growth and shrinkage to growth. Transitions associated with fecundity are associated with the lowest summed elasticity values.

Package `lefko3` also includes two functions to conduct general projection simulations, including stochastic simulations. Please see chapter 10 for more.

9.5 Points to remember

1. Stochastic analysis typically refers to matrix projection in which matrices are shuffled randomly a large number of times. The properties of the projection in the long-term have a tendency to be approximately stationary. These analyses assess the impacts of random environmental shifts across time on population dynamics. Although stochasticity may also be demographic or spatial, we concern ourselves only with temporal stochasticity in this chapter because of its dominance in the literature.
2. Package `lefko3` can conduct all major stochastic analyses even for large numbers of massive matrices, such as historical IPMs and age-by-stage MPMs. This includes even sensitivity and elasticity analyses.
3. In most cases, stochastic analysis in `lefko3` is handled with the same functions as deterministic analysis, but using the option `stochastic = TRUE`.