An MPM with recurrent removal (Hastings *et al.* 2006) - eq. 3

Bram D'hondt

2025-12-19

Introduction

$$N_T = L^T N_0 - \sum_{i=1}^{T} L^{T+1-i} H_i$$

This is a preparatory document that develops the above formula step by step, using a simple example. This not only serves as an illustration, but it also acts as a reference to check whether the developed function, the scripting of which is way more abstract, is correct.

First, I avoid using *popdemo* functions, to stay closer to the core formulation. Then, I provide the *popdemo* alternative, which is heavily used in further development. Note how the *popdemo::project()* function has the huge advantage that, for a given time span, all intermediate values are provided. Also, by using the option *return.vec = FALSE/TRUE*, one can switch between output per stage or summed over stages.

As a time span, we opt for five steps (years), i.e. $T = 5$.

L, $N0$ and H

L is $\begin{bmatrix} 0 & 0.1 & 0.1 \\ 1 & 1 & 1 \\ 0 & 0.3 & 1.2 \end{bmatrix}$

N_0 is $\begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$

The area removed is H. Removal is performed at the **start of the year**. This is important to understand, because for one thing, it means that you will nowhere see the value of H in the 'ghost matrix' (see further); the first value you will encounter directly is LH, i.e. the end-of-year 'ghost' of what was newly removed at the start of the year.

H is

```
##       [,1]
## [1,]   0.0
## [2,]   0.1
## [3,]   0.0
```

Without removal

If there would be **no removal**, an MPM's base formula gives the population extent.

$$N_T = L^T N_0$$

Without *popdemo*:

```
N1 <- sum(L%^%1 %*% N0) # expm package
N2 <- sum(L%^%2 %*% N0) # matrix power (L^2 = element-wise, L%^%2 = L x L)
N3 <- sum(L%^%3 %*% N0)
N4 <- sum(L%^%4 %*% N0)
N5 <- sum(L%^%5 %*% N0)

print(c(sum(N0), N1, N2, N3, N4, N5))
```

```
## [1] 0.30000 0.47000 0.78500 1.32400 2.24250 3.80365
```

With *popdemo*. Note how $N0$ is automatically included in the output.

```
Nt <- project(L, vector = N0, time = 5, return.vec = T)

print(Nt)        # across stages
```

```
## 1 deterministic population projection over 5 time intervals.
##
## [1] 0.30000 0.47000 0.78500 1.32400 2.24250 3.80365
```

```
print(t(Nt@vec)) # per stage
```

```
##     [,1] [,2]  [,3]  [,4]   [,5]    [,6]
## S1   0.1 0.02 0.045 0.074 0.1250 0.21175
## S2   0.1 0.30 0.470 0.785 1.3240 2.24250
## S3   0.1 0.15 0.270 0.465 0.7935 1.34940
```

With removal

Let's now include **removal**. The equation on top states that from this total projection, we now have to substract a contingent of "what could have been", i.e. that which would have resulted from the removed portion ($-L^T H$), for each of the time steps that have passed ($sum_{i=1}^{T}$).

I refer to the removed cohorts and how they would have propagated as 'ghosts', their yearly numbers collected in the *ghost matrix*.

Without *popdemo*:

```
total  = sum(L%^%5 %*% N0)         # total (actual + removed) - as above

ghost1 = sum(L%^%(5+1-1) %*% H)    # removed in year 1 (five years ago)
ghost2 = sum(L%^%(5+1-2) %*% H)    # removed in year 2 (four years ago)
```

```
ghost3 = sum(L%^%(5+1-3) %*% H)    # removed in year 3 (three years ago)
ghost4 = sum(L%^%(5+1-4) %*% H)    # removed in year 4 (two years ago)
ghost5 = sum(L%^%(5+1-5) %*% H)    # removed in year 5 (previous year)

total-sum(ghost1, ghost2, ghost3, ghost4, ghost5)
```

```
## [1] 1.453128
```

With *popdemo*:

```
total.  = project(L, vector = N0, time = 5, return.vec = F)
ghost.1 = project(L, vector = H,  time = 5+1-1, return.vec = F) # removed in year 1 (five years ago)
ghost.2 = project(L, vector = H,  time = 5+1-2, return.vec = F) # removed in year 2 (four years ago)
ghost.3 = project(L, vector = H,  time = 5+1-3, return.vec = F) # removed in year 3 (three years ago)
ghost.4 = project(L, vector = H,  time = 5+1-4, return.vec = F) # removed in year 4 (two years ago)
ghost.5 = project(L, vector = H,  time = 5+1-5, return.vec = F) # removed in year 5 (previous year)

tail(total.,1)-(tail(ghost.1,1)+tail(ghost.2,1)+tail(ghost.3,1)+tail(ghost.4,1)+tail(ghost.5,1))
```

```
## [1] 1.453128
```

This now opens the way to the generic formulation, in the next script (2_eq3_…).