Population Projection Matrix Models in R

Mario Vallejo-Marin

23/02/2017

Building a projection matrix

Once you have calculated the demographic rates of **survival**, **growth or transition rates**, and **fertility**, you can build a projection matrix.

A typical matrix in which there are three stages (e.g., seedling, juvenile, and adult), looks like this:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

where the a represent transition probabilities. *The transition rates represent the movement of the **j__th column** stage to the **i__th row** stage P_{ij}.

In an organism where there are three pre-reproductive stages, and one reproductive stage, the projection matrix A may look like this:

$$\mathbf{A} = \begin{bmatrix} P_{11} & 0 & 0 & F_4 \\ P_{21} & P_{22} & P_{23} & 0 \\ 0 & P_{32} & P_{33} & 0 \\ 0 & 0 & P_{43} & P_{44} \end{bmatrix}$$

where P represent the transition probabilities between stages P_{ij} or within a stage P_{ii}. The element F is reserved for *reproduction.* A matrix entry of zero indicates that the transition does not occur.

To build the matrix you need to combine the vital rates to generate each matrix element, where:

- s_j is the survival rate of class j
- f_j is the fertility rate for class j
- g_{ij} is the probability that an individual in class j in this census makes the transition to class i

An example with clonal *Mimulus*

Here we will use a recent example of estimating population growth rate in *Mimulus* populations reproducting both sexually and clonally. This study was conducted by Megan Peterson and colleagues:

Peterson ML, Kay KM, Angert AL (2016) The scale of local adaptation in *Mimulus guttatus*: comparing life history races, ecotypes, and populations. *New Phytologist*, 211, 345–356.

Suplementary Material

The general life cycle of *Mimulus* is shown here:

Figure. Life cycle graph of *Mimulus.* Black arrows indicate transitions for annuals, and grey+black are transitions for perennials. From Peterson et al. (2016).

The vital rates for the life cycle graph are given below.

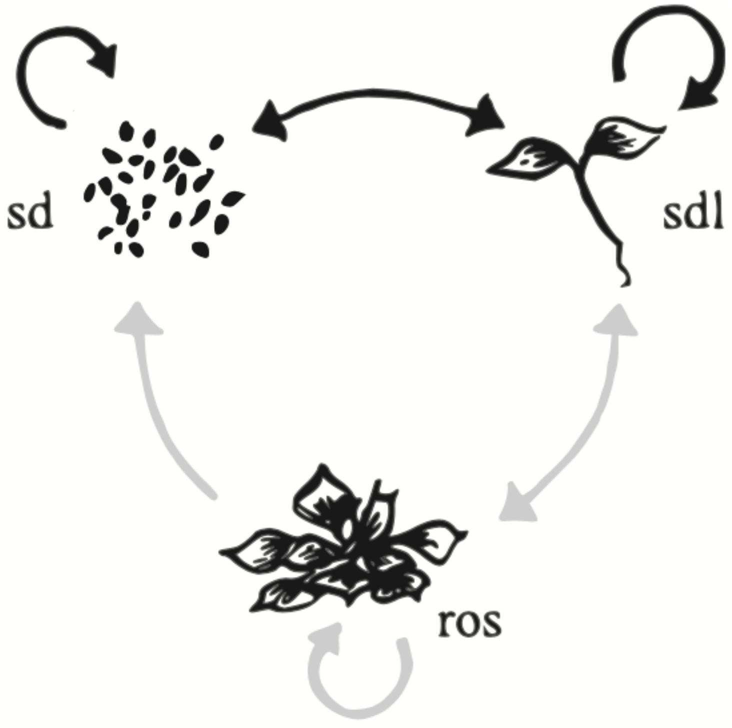


Figure 1: life cycle graph

$$\begin{array}{cccc}
 & Seed_t & Seedling_t & Rosette_t \\
Seed_{t+1} & D(1-G) & FeOA(1-G) & FeOA(1-G) \\
Seedling_{t+1} & DG & FeOAG & FeOAG \\
Rosette_{t+1} & 0 & SR & SR
\end{array}$$

where D is the survival of seeds in the seed bank, G = germination rate, O = ovule number (a proxy for seed production), Fe = flower number, S = overwinter survival, and R = number of additional vegetative rosettes. For details, see Peterson et al. (2016).

This model estimates yearly transitions based ona pre-reproductive census, i.e., right at the start of the growth season, just after the snwomelt. After the census, either seedlings or rosettes can flower and contribute to seed or seedling class, or survive to produce vegetative rosettes in the next time step. Seed to seed transitions represent dormancy in the seed bank.

In an annual population, overwinter survival S is zero, and the number of clonal propagules R is also zero. So the matrix above becomes:

$$\begin{array}{cccc}
 & Seed_t & Seedling_t & Rosette_t \\
Seed_{t+1} & D(1-G) & FeOA(1-G) & FeOA(1-G) \\
Seedling_{t+1} & DG & FeOAG & FeOAG \\
Rosette_{t+1} & 0 & 0 & 0
\end{array}$$

Analysis with ***POPBIO***

Here we will use the `library(popbio)` to analyse population projection models. *popbio* is an package that implements the equations from Caswell (2001), and Morris and Doak (2002) to study matrix populations models.

popbio also contains functions to calculate vital rates and construct projection matrices from raw census datas, as is typical of plant demography studies.

First load the library:

```
library(popbio)
```

Demographic parameters

We will use the the data from Peterson et al. (2016), focusing on three annual and three perennial populations.

In this step, we will create an object that holds the vital rates for a number of annual and perennial populations.

```
par.dat<-data.frame(pop=c("rh","bm","pb","sf","em","sc"), life.history = c(rep("annual",3),rep("per",3)
                    G = c(0.559,0.818,0.740,0.552,0.469,0.410),
                    O = c(286.9,170.9,616.8,803.1,613.9,649.5),
                    Fe = c(5.28,8.60,8.62,0.40,0.64,0.63),
                    S = c(0,0,0,0.1,0.179,0.158),
                    R = c(0,0,0,0.82,1.56,0.66))
par.dat
```

```
##   pop life.history     G     O   Fe     S    R
## 1  rh       annual 0.559 286.9 5.28 0.000 0.00
## 2  bm       annual 0.818 170.9 8.60 0.000 0.00
## 3  pb       annual 0.740 616.8 8.62 0.000 0.00
## 4  sf          per 0.552 803.1 0.40 0.100 0.82
## 5  em          per 0.469 613.9 0.64 0.179 1.56
## 6  sc          per 0.410 649.5 0.63 0.158 0.66
```

Each row represent one population.

The following parameters are the same for all populations:

```
D = 0.534 # Seedbank survival was treated as a constant across all populations
A = 6.7e-4 #Proportional recruitment of ovules relative to rosettes
```

Annual population, no clonality

Now, we need to create the projection matrix. We will begin by analysing one annual population called 'Red Hills'.

```
s1<-subset(par.dat, pop=="rh")   #Subsets only population Red Hills "rh"
s1
```

```
##   pop life.history     G     O   Fe S R
## 1  rh       annual 0.559 286.9 5.28 0 0
```

```
attach(s1) #attaches the object so you can call individual elements
#This next step builds the tranistion matric by multiplying the appropriate vital rates:
pmat<-matrix(c(D*(1-G), Fe*O*A*(1-G), Fe*O*A*(1-G),  D*G, Fe*O*A*G, Fe*O*A*G,   0, S*R, S*R),
             nrow=3, ncol=3, byrow=T,
             dimnames = list(c("seed_t+1","seedling_t+1", "rosette_t+1"),
                             c("seed_t","seedling_t", "rosette_t")))
pmat
```

```
##                 seed_t seedling_t rosette_t
## seed_t+1     0.235494  0.4475874 0.4475874
## seedling_t+1 0.298506  0.5673500 0.5673500
## rosette_t+1  0.000000  0.0000000 0.0000000
```

```
detach(s1)  #Detaches the object s1 from memory
```

The most straightforward calculation once the transition amtrix has been generated is to estimate the population growth rate parameter `lambda`:

```
lambda(pmat)
```

```
## [1] 0.802844
```

Is the population shrinking or growing?

Other analyses

We can also conduct other analysis on this matrix, such as calculating the proportion of individuals in each class at equilibrium, the changes in numbers of individuals through time, and so on. To do that we will use a function called `pop.projection`.

To calculate the projection estimates, we first need to define an initial stage vector (the number of individuals in each of the stage classes). This is often best obtained from census numbers. Here we will use a hypothetical vector given that this was an experimental plot. It is wise to check how sensitive are your findings to different starting conditions.

```
init.vector<-c(rep(100,3))
```

```
pj1<-pop.projection(pmat, init.vector, iterations=20)
pj1
```

```
## $lambda
## [1] 0.802844
##
## $stable.stage
##     seed_t+1 seedling_t+1  rosette_t+1
##        0.441        0.559        0.000
##
## $stage.vectors
##                0        1         2        3        4        5        6
## seed_t+1     100 113.0669  90.77507 72.87822 58.50985 46.97428 37.71302
## seedling_t+1 100 143.3206 115.06409 92.37852 74.16554 59.54336 47.80403
## rosette_t+1  100   0.0000   0.00000  0.00000  0.00000  0.00000  0.00000
##                     7        8        9       10       11       12
## seed_t+1     30.27767 24.30825 19.51573 15.66809 12.57903 10.09900
## seedling_t+1 38.37918 30.81250 24.73763 19.86046 15.94485 12.80123
## rosette_t+1   0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
##                     13       14       15       16       17       18
## seed_t+1      8.107923 6.509397 5.226031 4.195688 3.368483 2.704366
## seedling_t+1 10.277389 8.251141 6.624379 5.318343 4.269800 3.427984
## rosette_t+1   0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
##                    19
## seed_t+1     2.171184
## seedling_t+1 2.752136
## rosette_t+1  0.000000
##
## $pop.sizes
##  [1] 300.000000 256.387488 205.839164 165.256744 132.675390 106.517645
##  [7]  85.517055  68.656857  55.120748  44.253363  35.528548  28.523883
## [13]  22.900229  18.385312  14.760538  11.850410   9.514031   7.638283
## [19]   6.132350   4.923320
##
## $pop.changes
##  [1] 0.854625 0.802844 0.802844 0.802844 0.802844 0.802844 0.802844
##  [8] 0.802844 0.802844 0.802844 0.802844 0.802844 0.802844 0.802844
## [15] 0.802844 0.802844 0.802844 0.802844 0.802844
```

The analysis returns several pieces of information about the population's performance in the future. Let's see each of them in turn.

The eigenvalue, *lambda* is given in:

```
pj1$lambda
```

```
## [1] 0.802844
```

The `stable.stage` element contains the proportion of individuals in the population in each stage, once the equilibrium has been reached.

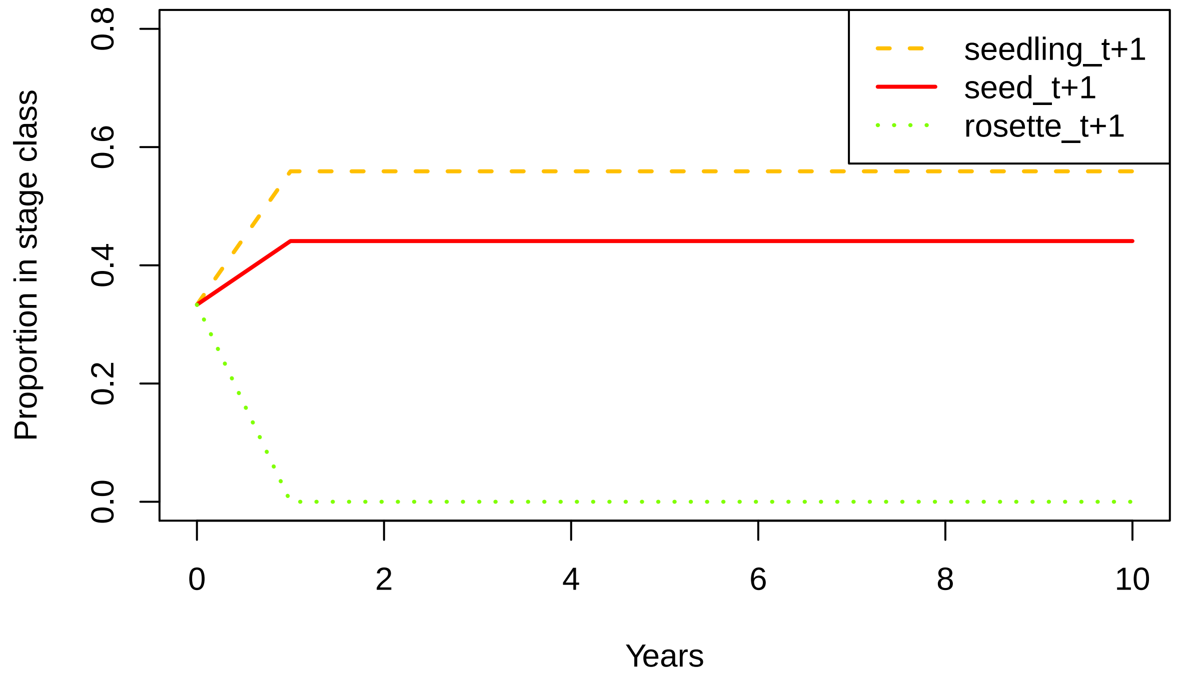
```
pj1$stable.stage
```

```
##     seed_t+1 seedling_t+1  rosette_t+1
##        0.441        0.559        0.000
```

The `stage.vectors` element shows you how the number of individuals in each stage changes through the x number of iterations for which you ran your model (20, in this example).

You can see the changes in the proportion of individuals in each class using the `stage.vector.plot` function.

```
stage.vector.plot(pj1$stage.vectors[,1:11], ylim=c(0,.8))  #Plots initial state and the first ten years
```



The `pop.sizes` element shows you how the population size changes through time, and `pop.changes` give you the proportional change through time.

- **Can you produce population projection analyses for the other two annual populations?**

- **Are these populations also shrinking, or are they growing?**

Perennial population, with clonality

Let's now look at the dynamics of a perennial population. For this we will choose the population 'Eagle Meadows', the local population originary from the same place as the common garden.

For brevity, I will omit the `R` output.

```
s2<-subset(par.dat, pop=="em")   #Subsets only population Red Hills "rh"
s2
```

```
##   pop life.history     G     O   Fe     S    R
## 5  em          per 0.469 613.9 0.64 0.179 1.56
```

```
attach(s2) #attaches the object so you can call individual elements
pmat.s2<-matrix(c(D*(1-G), Fe*O*A*(1-G), Fe*O*A*(1-G),  D*G, Fe*O*A*G, Fe*O*A*G,   0, S*R, S*R),
             nrow=3, ncol=3, byrow=T,
             dimnames = list(c("seed_t+1","seedling_t+1", "rosette_t+1"),
                             c("seed_t","seedling_t", "rosette_t")))
detach(s2)
pmat.s2
```

```
##                  seed_t seedling_t rosette_t
## seed_t+1     0.283554  0.1397806 0.1397806
## seedling_t+1 0.250446  0.1234597 0.1234597
## rosette_t+1  0.000000  0.2792400 0.2792400
```

```
lambda(pmat.s2)
```

```
## [1] 0.5394847
```

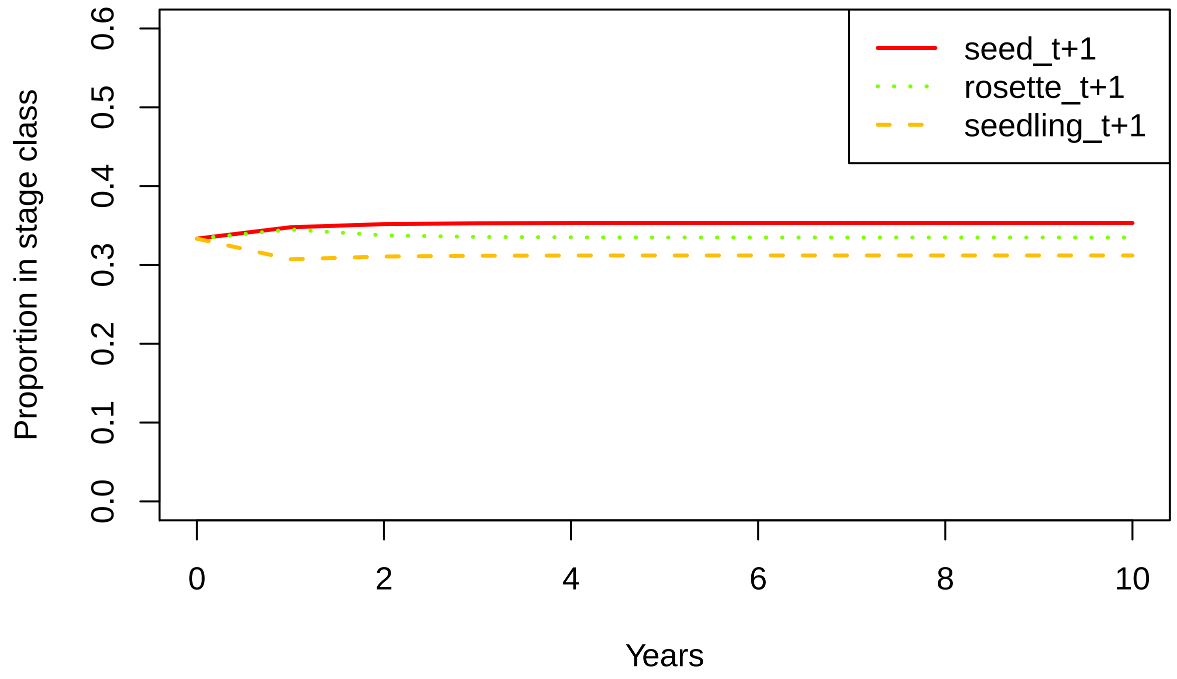
The matrix analysis is done next.

```
pj2<-pop.projection(pmat.s2, init.vector, iterations=20)
pj2
```

```
## $lambda
## [1] 0.5394847
##
## $stable.stage
##     seed_t+1 seedling_t+1  rosette_t+1
##    0.3532388    0.3119944    0.3347668
##
## $stage.vectors
##                0        1        2        3        4        5        6
## seed_t+1     100 56.31152 30.72603 16.62713 8.977552 4.844349 2.613613
## seedling_t+1 100 49.73654 27.13843 14.68573 7.929326 4.278719 2.308446
## rosette_t+1  100 55.84800 29.48343 15.81109 8.515931 4.592174 2.477108
##                     7         8         9        10        11         12
## seed_t+1     1.410028 0.7606921 0.4103823 0.2213951 0.1194393 0.06443566
## seedling_t+1 1.245392 0.6718731 0.3624657 0.1955448 0.1054934 0.05691210
## rosette_t+1  1.336318 0.7209167 0.3889226 0.2098177 0.1131934 0.06106612
##                      13         14          15          16          17
## seed_t+1     0.03476206 0.01875360 0.010117281 0.005458119 0.002944572
## seedling_t+1 0.03070321 0.01656391 0.008935979 0.004820824 0.002600761
## rosette_t+1  0.03294424 0.01777291 0.009588216 0.005172696 0.002790591
##                          18           19
## seed_t+1     0.001588552 0.0008569993
## seedling_t+1 0.001403071 0.0007569354
## rosette_t+1  0.001505481 0.0008121841
##
## $pop.sizes
##  [1] 3.000000e+02 1.618961e+02 8.734789e+01 4.712394e+01 2.542281e+01
##  [6] 1.371524e+01 7.399167e+00 3.991738e+00 2.153482e+00 1.161771e+00
## [11] 6.267575e-01 3.381261e-01 1.824139e-01 9.840951e-02 5.309043e-02
## [16] 2.864148e-02 1.545164e-02 8.335923e-03 4.497103e-03 2.426119e-03
##
## $pop.changes
##  [1] 0.5396535 0.5395307 0.5394972 0.5394881 0.5394857 0.5394850 0.5394848
##  [8] 0.5394848 0.5394847 0.5394847 0.5394847 0.5394847 0.5394847 0.5394847
## [15] 0.5394847 0.5394847 0.5394847 0.5394847 0.5394847
```

For this perennial population, the value of `lambda` is *0.539*.

And the proportion of individuals in each class can be visualised as before

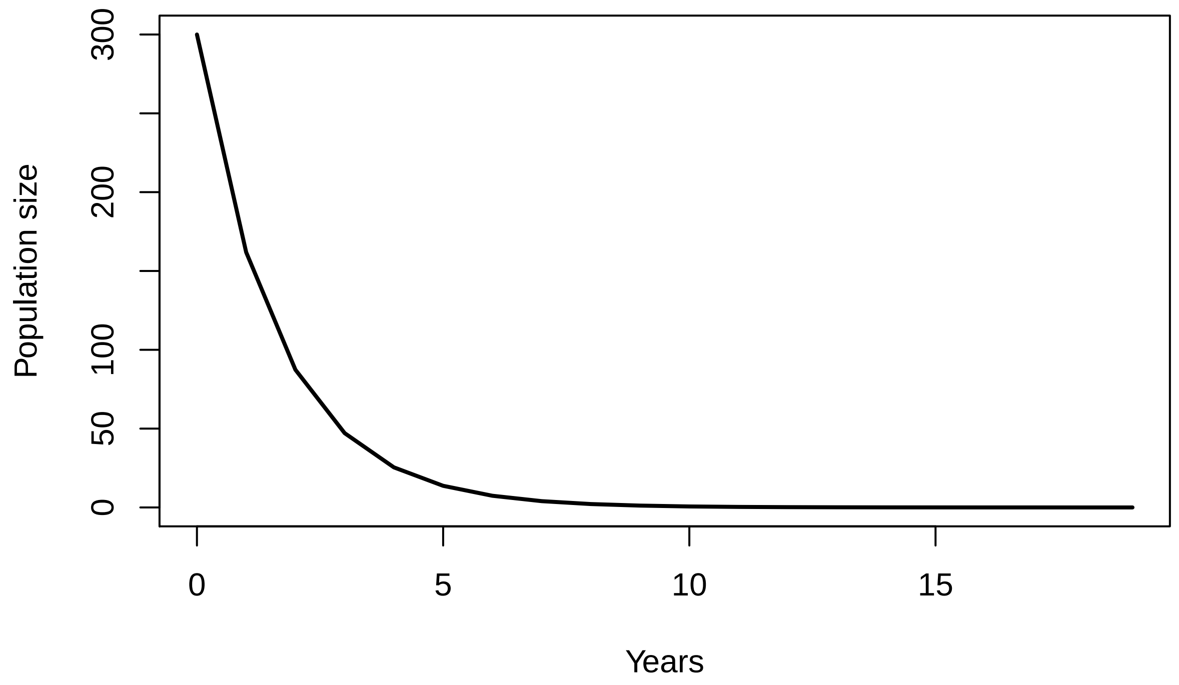


```
##     seed_t+1 seedling_t+1  rosette_t+1
##    0.3532388    0.3119944    0.3347668
```

In this case, about 33.5% of the population will occur as rosettes.

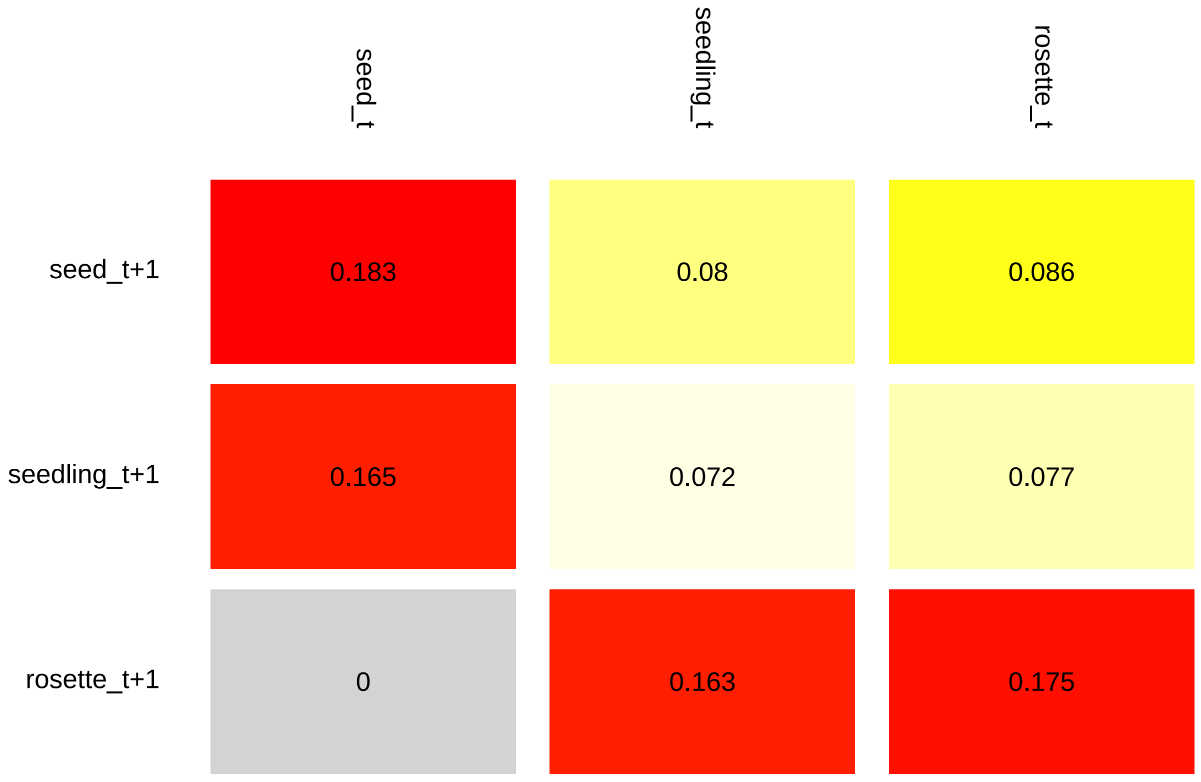
You can visualise the change in population size with time. Because `lambda < 1`, we expect this population to eventually become extinct.

```
plot(pj2$pop.sizes ~ seq(from=0, to=length(pj2$pop.sizes)-1, by=1), xlab="Years", ylab="Population size
```



###Elasticity We can now look at the effect of slightly changing each of the elements of the transition matrix in population growth rate. We do this by calculating the **elasticities** of the transition matrix.

```
elas<-elasticity(pmat.s2)  #Calculates elasticities
#Plot the eleasticity matrix, with warmer colours indicating larger contributions to population growth
image2(elas, mar=c(1,5,5.5,1), cex=0.8, col = c("lightgray", rev(heat.colors(23))))
```



```
sum(elas)  # The sum of all the elasticities should add up to one.
```

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

We can see that the transitions from seedling and rosette to rosettes have high values of elasticity (bottom row of the matrix). In this matrix, such transitions represent the product of plant survivorship and rosette production `S*R`. So, how can we disentangle the contribution of individual vital rates (such as rosette production, `R`)?

What is the effect of clonality?

(1) An approach using simulations

A crude approximation to estimate the effect of changing a given vital rate on lambda is to conduct a simulation. This can be easily done in **R** by repeatedly calculating `lambda` for a range of parameter values. For example, let's build a function to calculate `lambda` in an iterative fashion (if you are unfamiliar with how to build loops in R, I will be happy to explain the general logic and point you to online resources to learn how to do it).

The goal is to obtain a data set containing a value of `lambda` for each value of `R`.

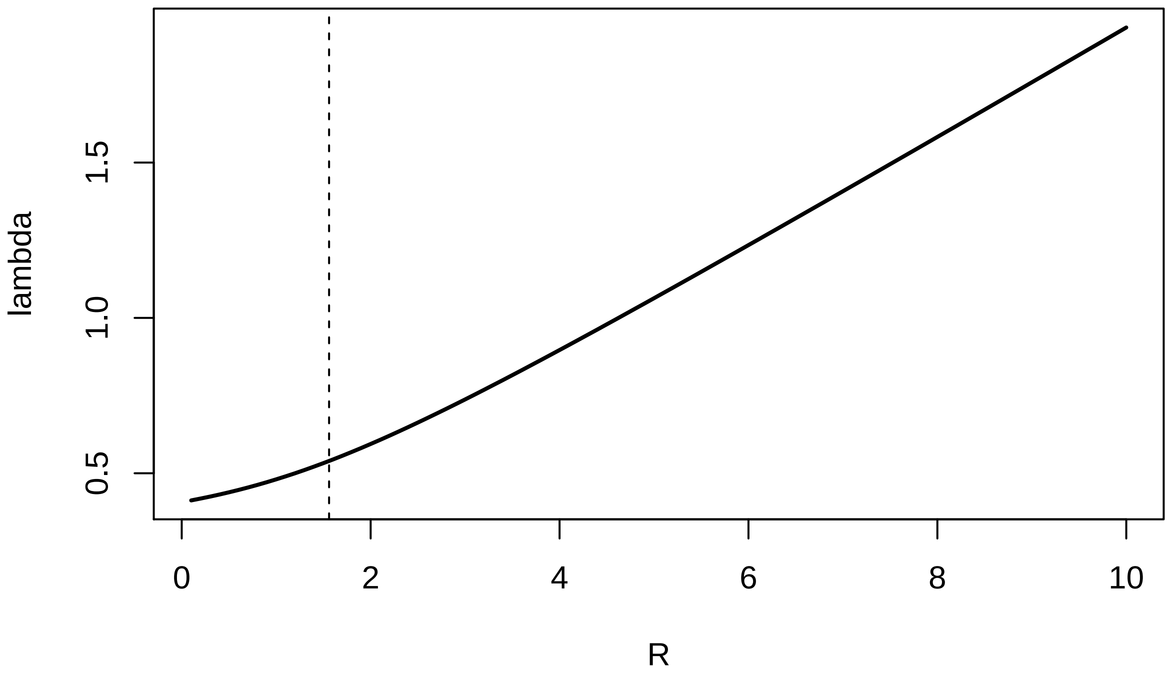
```
#Simulate different rosette production values:
simR<-function(vitrates){
  lambda.results<-data.frame(R=rep(NA,100),lambda=rep(NA,100)) #Creates a data frame to store the resul
  attach(vitrates)
  for (i in 1:100){
    R=i/10  #At every iteration of the loop, the value of R will increase slightly. Change this to test
    pmat<-matrix(c(D*(1-G), Fe*O*A*(1-G), Fe*O*A*(1-G),  D*G, Fe*O*A*G, Fe*O*A*G,   0, S*R, S*R),
                 nrow=3, ncol=3, byrow=T,
                 dimnames = list(c("seed_t+1","seedling_t+1", "rosette_t+1"),
                                 c("seed_t","seedling_t", "rosette_t")))
  lambda.results[i,1]<-R
  lambda.results[i,2]<-lambda(pmat)
  }
  #print(lambda.results) #Uncomment this if you want to see the results after the loop is finished
  return(lambda.results)  #This line tells the program to give you back the dataframe you created so yo
  detach(vitrates)
}
```

And now you can execute your newly created function.

```
sim1<-simR(s2) #The function requires you to provide it with a list of parameter values
sim1[1:20,] #Shows you the first twenty rows of the simulation
```

```
##      R    lambda
## 1  0.1 0.4126125
## 2  0.2 0.4185610
## 3  0.3 0.4248753
## 4  0.4 0.4315705
## 5  0.5 0.4386601
## 6  0.6 0.4461557
## 7  0.7 0.4540668
## 8  0.8 0.4624003
## 9  0.9 0.4711604
## 10 1.0 0.4803484
## 11 1.1 0.4899626
## 12 1.2 0.4999986
## 13 1.3 0.5104491
## 14 1.4 0.5213044
## 15 1.5 0.5325528
## 16 1.6 0.5441803
## 17 1.7 0.5561720
## 18 1.8 0.5685114
## 19 1.9 0.5811815
## 20 2.0 0.5941650
```

```
plot(sim1$lambda~sim1$R, xlab="R", ylab="lambda", type="l", lwd=2) #Shows you the result in a graphical
abline(v=1.56, lty=2)  #Plots the original value of R with a dotted vertical lines
```



(2) An approach using partial derivatives

A more elegant, and powerful approach to estimate the effect of changing individual vital rates on population growth is to calculate the elasticity of vital rates (e.g., `R` or `S`, in contrast to the elasticity of transition probablilities `P_ij`). This approach involves calculating partial derivatives on the vital rates themselves (remember that elasticities are standardised partial derivatives on transition probabilities). This approach is also implemented in `popbio` using the function `vitalsens`, and is easy to do.

To do this we need a list of vital rate. We can get this by combining the fixed vital rates **D** and **A**, with the vital rates of the population of interest, and then creating a matrix with the symbolic representation of the transition rates:

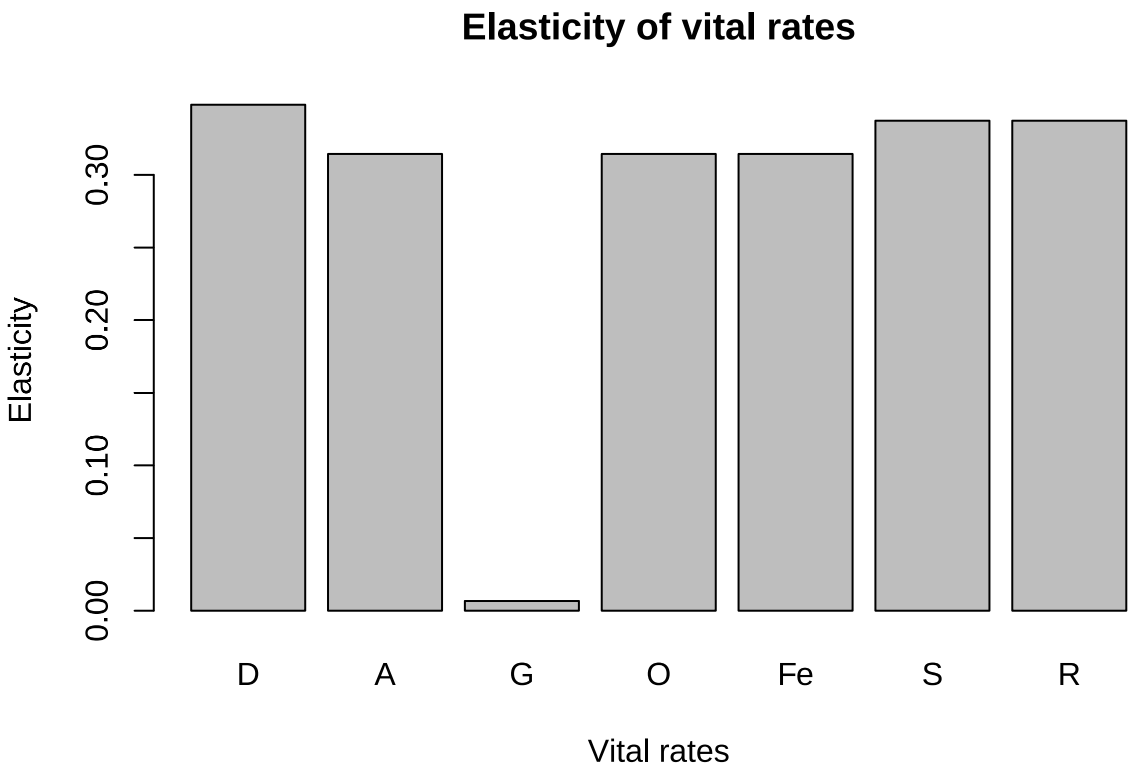
```
em.vr<-list(D=0.534, A=6.7e-4, G=0.469, O=613.9, Fe=0.64, S=0.179, R=1.56 ) #list of vital rates
em.matrix<-expression(
  D*(1-G), Fe*O*A*(1-G), Fe*O*A*(1-G),
  D*G,     Fe*O*A*G,     Fe*O*A*G,
  0,       S*R,          S*R)
```

No you can use the function `vitalsens` to calculate the sensitivity and elasticity of individual rate parameters.

```r
sens.em<-vitalsens(em.matrix, em.vr)
sens.em
```

```
##     estimate  sensitivity  elasticity
## D    0.53400 3.518829e-01 0.348305418
## A    0.00067 2.531345e+02 0.314374208
## G    0.46900 7.749737e-03 0.006737219
## O  613.90000 2.762666e-04 0.314374208
## Fe   0.64000 2.650001e-01 0.314374208
## S    0.17900 1.016644e+00 0.337320374
## R    1.56000 1.166533e-01 0.337320374
```

```r
barplot(t(sens.em)[3,], ylab="Elasticity", xlab="Vital rates", main="Elasticity of vital rates")
```



Note that in this calculation, the sum of all the elasticities does not add up to one. See Casswell (2001), p. 232 for a technical explanation.

You can see that the elasticity of `R` is similar to those of other parameters. The exception is germination rate `G` which seems to contribute proportionally less to population growth rate.

Exercises

1. Calculate lambda for the six populations
2. Compare lambda for annual (non-clonal) and perennial (clonal) populations
3. Calculate the elasticity of `R` for the perennial populations
4. Estimate the effect of changing `R` using a simulation approach