# Matrix Math Notation

Cole Brookson

10/20/2021

In this lecture we'll go over how to compute our various matrix diagnostic values in R, and also discuss how to write matrices like these in RMarkdown.

Let's start by going back to our whale example. We want to have our equations in Markdown. Let's do that by writing some fancy Latex.

So first things first, I'm actually going to change our equations SLIGHTLY from when we did this on the board, just to model a different assumption for simplicity. We no longer are going to have this group of reproducing females returning to the female and calf group, you ONLY go back to the mature females in between. So that's just represented like this:

$$N_c(t+1) = N_R(t)b$$

$$N_I(t+1) = N_C(t)s_{IC} + N_I(t)s_{II}$$

$$N_M(t+1) = N_I(t)s_{MI} + N_M(t)s_{MM} + N_R(t)s_{MR}$$

$$N_R(t+1) = N_I(t)s_{RI} + N_M(t)s_{RM}$$

but we'll also want this to be

$$\begin{pmatrix} N_C(t+1) \\ N_I(t+1) \\ N_M(t+1) \\ N_R(t+1) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & b \\ S_{IC} & S_{II} & 0 & 0 \\ 0 & S_{MI} & S_{MM} & S_{MR} \\ 0 & S_{RI} & S_{RM} & 0 \end{pmatrix} \begin{pmatrix} N_C(t) \\ N_I(t) \\ N_M(t) \\ N_R(t) \end{pmatrix}$$

How fun!

Ok so let's FINALLY get to our whale example!!! So we have our equations above, so recall what we need is the transition matrix filled with values.

$$\begin{pmatrix} 0 & 0 & 0 & b \\ 0.92 & 0.86 & 0 & 0 \\ 0 & 0.08 & 0.8 & 0.75 \\ 0 & 0.02 & 0.19 & 0 \end{pmatrix}$$

We still need to fill in this birth term term. So that means we need some assumptions. The fecundity assumes a 50% sex ratio, where there is only one calf per mother, and the calf survives to independence due

to the mother surviving gestation ($s_{RM}$) and the mother surviving for at least half of the first year of the calf's life ($s_{MR}^{0.5}$)

$$\begin{pmatrix} 0 & 0 & 0 & b \\ 0.92 & 0.86 & 0 & 0 \\ 0 & 0.08 & 0.8 & 0.75 \\ 0 & 0.02 & 0.19 & 0 \end{pmatrix}$$

So just a quick note that these are coming from the a paper from Fujiwara and Caswell (*Nature*, 2001), but I am changing their model a little bit and that's okay, it's just for teaching purposes. So I actually want to say that the birth term above is

$$b = 0.5 \times s_{RM} \times s_{MR}^{0.5}$$

and that's just formulating what our assumptions above were.

Okay so let's get to our question with this this, with just the information we have right now!

```
# load packages we'll need
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(popbio)

# create matrix
whale_matrix = matrix(data = c(c(0, 0, 0, 0),
                       c(0.92, 0.86, 0, 0),
                       c(0, 0.08, 0.8, 0.75),
                       c(0, 0.02, 0.19, 0)),
                       nrow = 4, ncol = 4,
                       byrow = TRUE,
                       dimnames = list(c("calf", "immature",
                                         "matrue", "adult"),
                                       c("calf", "immature",
                                         "matrue", "adult")))

# create a function to update our matrix for the birth rate
update_birth_rate = function(matrix) {

  # get values for s_RM and s_MR
  # this one is in the matrix in row 4, col 3
  s_RM = matrix[4,3]
  s_MR = matrix[3,4]
```

```
  # now get the birth rate
  b = 0.5 * s_RM * (s_MR^(0.5))

  # put it in the matrix
  matrix[1,4]

  return(matrix)
}

# now update birth rate from zero to the new value
whale_matrix = update_birth_rate(whale_matrix)
```

Now that we've updated our birth rate, we can get re-run that function any time that our parameter values for the survival rates change, we can re-do our birth rate and have an up to date matrix.

```
# so first things first, let's get out sensitivity & elasticity:
sensitivity = sensitivity(whale_matrix, zero = FALSE)
elasticity = elasticity(whale_matrix)

# now let's get our growth rate
growth_rate = lambda(whale_matrix)

# print all these essential values
print("Sensitivity matrix");print(sensitivity)
```

```
## [1] "Sensitivity matrix"

##          calf immature    matrue     adult
## calf        0        0 0.8901648 0.1780330
## immature    0        0 0.9191919 0.1838384
## matrue      0        0 0.8636364 0.1727273
## adult       0        0 0.6818182 0.1363636
```

```
print("Elasticity matrix");print(elasticity)
```

```
## [1] "Elasticity matrix"

##          calf immature    matrue     adult
## calf        0        0 0.0000000 0.0000000
## immature    0        0 0.0000000 0.0000000
## matrue      0        0 0.7272727 0.1363636
## adult       0        0 0.1363636 0.0000000
```

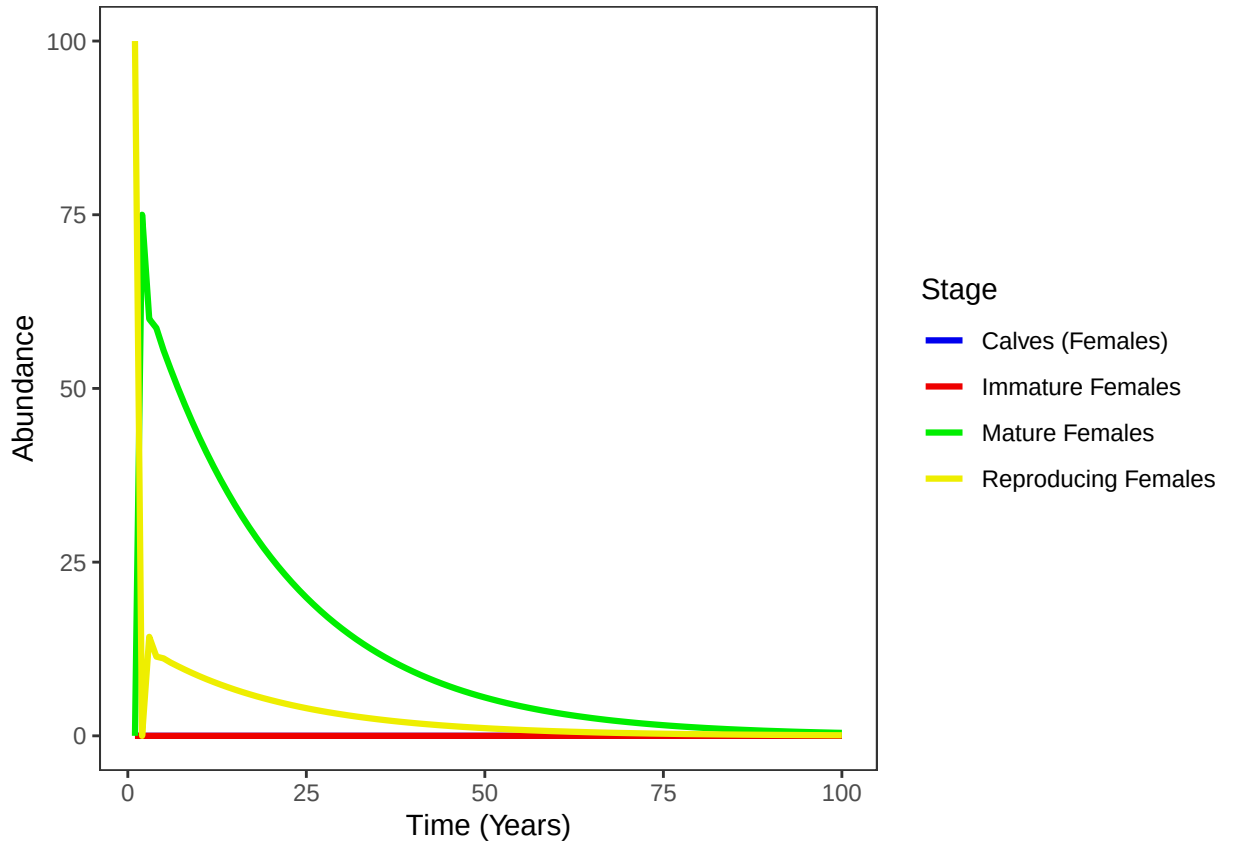
```
print("growth rate");print(growth_rate)
```

```
## [1] "growth rate"

## [1] 0.95
```

So above we got our most important values. Let's now simulate this a little bit to see what the population might project to in the future.

```
library(tidyverse)

# we'll do this by making a function, so we can repeat it multiple times if we
# want to

# make matrix of abundance over 100 years
abundance_df = function(transition_matrix,
                        initial_pop,
                        timesteps){

  # set up an empty matrix where the matrix is 4 rows (because our example
  # has 4 stages) and we fill the matrix with zeros to start off
  pop1 = matrix(rep(0,4*timesteps), 4, timesteps)

  # set the first column (i.e. the first time step) to the initial abundances
  pop1[,1] = initial_pop

  # use our for loop to iterate through the time steps
  for(i in 2:timesteps) {
    # %*% is matrix multiplication in R
    # here, we're filling in each column (i) based on the transition matrix
    # multiplied by the abundance at the previous time step (i-1)
    pop1[,i] = (transition_matrix %*% pop1[,i-1])
  }

  # now let's take our matrix pop1 and put it into a data frame
  pop1_for_plot = data.frame(
    # pop will be our first column name, with the all the abundances of our
    # different stages (recall each stage was a single row in the matrix
    # pop1) that we're pasting together
    pop = c((pop1[1,]), (pop1[2,]), (pop1[3,]), (pop1[4,])),
    # next column is "gen" for the generations, and we want to name them
    # accordingly by using the rep() function to repeat the first value
    # (e.g. "Calves (Females)") by the number of timesteps in our function
    gen = c(rep('Calves (Females)', timesteps),
            rep('Immature Females', timesteps),
            rep('Mature Females', timesteps),
            rep('Reproducing Females', timesteps)),
    # now we want t a column of the timesteps so we can plot it as a timeseries
    ts = rep(c(1:timesteps), 4))
}

# now we use the function we just made, abundance_df() to make our new data
# frame, using the vector c(0,0,0,100) (i.e. 100 reproducing adults and that's
# it) as our initial abundances, and 100 as the number of timesteps
pop_100_yrs = abundance_df(whale_matrix,
                           c(0,0,0,100),
                           100)
# now let's make a plot!
ggplot(data = pop_100_yrs) +
  geom_line(aes(x = ts, y = pop, colour = gen), size = 1.1) +
  theme_bw() +
  theme(
```

```
    # get rid of inner grid
    panel.grid = element_blank()
  ) +
  # specify colours
  scale_colour_manual("Stage", values = c("blue2", "red2",
                                          "green2", "yellow2")) +
  labs(x = "Time (Years)", y = "Abundance")
```




So we can see here that with these values, this whale population declines quite quickly to near extinction by 100 years. Sad! So let's see what parameters we might be able to change to change our growth rate. Recall that $\lambda > 1$ means the population will grow. If we look back at our elasticity matrix:

```
print(elasticity)
```

```
##           calf immature    matrue     adult
## calf         0        0 0.0000000 0.0000000
## immature     0        0 0.0000000 0.0000000
## matrue       0        0 0.7272727 0.1363636
## adult        0        0 0.1363636 0.0000000
```

We can see that the survival rate for mature females is easily the most elastic parameter. So let's see how our lambda value will change if we change our parameter value for $s_{MM}$.

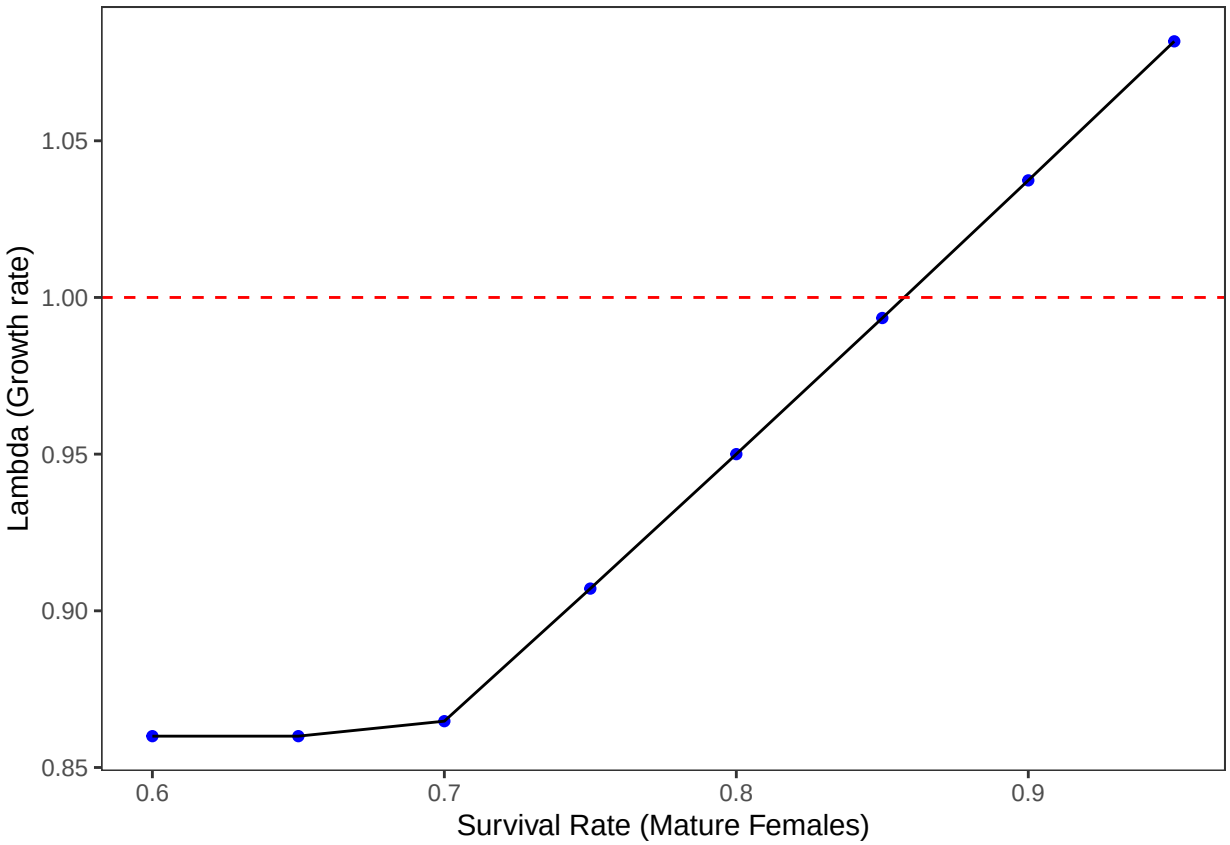
```
# update matrix function - we'll write a function to take our matrix, and our
# new value for our s_MM value, and put the new value in the matrix
```

```
update_survival_rate = function(matrix, new_value) {
  # value of s_MM is [3,3] in our matrix
  matrix[3,3] = new_value
  return(matrix)
}

# now let's make an empty dataframe to track our changed parameter values and
# our changed lambda values
s_mm_lambda = data.frame(s_mm = c(0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95),
                         # set all the values of lambda to NA for now
                         lambda = rep(NA, 8))

s_mm_values = c(0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95)
for(i in 1:length(s_mm_values)) {
  # run the update_survival_rate function to get a new matrix
  temp_matrix = update_survival_rate(whale_matrix, s_mm_values[i])
  # get our temporary lambda for that matrix
  temp_lambda = lambda(temp_matrix)
  # now put that temporary lambda in the dataframe
  s_mm_lambda[i,"lambda"] = temp_lambda
}
print(s_mm_lambda)
```

```
##   s_mm    lambda
## 1 0.60 0.8600000
## 2 0.65 0.8600000
## 3 0.70 0.8647815
## 4 0.75 0.9070949
## 5 0.80 0.9500000
## 6 0.85 0.9934409
## 7 0.90 1.0373670
## 8 0.95 1.0817331
```

Now that we've done all this we can see visually that there is a value just over 0.85 that will result in a positive growth rate (i.e. $\lambda > 1$). But it might be easier to see that if we plot it.

```
ggplot(data = s_mm_lambda) +
  geom_point(aes(x = s_mm, y = lambda), size = 1.5, colour = "blue") +
  geom_line(aes(x = s_mm, y = lambda)) +
  geom_hline(yintercept = 1, linetype = "dashed",
             colour = "red") +
  theme_bw() +
  theme(
    panel.grid = element_blank()
  ) +
  labs(x = "Survival Rate (Mature Females)", y = "Lambda (Growth rate)")
```




There are a number of other fun simulations we could do, but this is enough for this intentionally brief demo!