BIO311: Population Ecology
Prac 9: Population Matrices & LTRE

Koen van Benthem & Tina Cornioley

koen.vanbenthem@ieu.uzh.ch
tina.cornioley@ieu.uzh.ch

Spring 2014

Contents

1 Hypothetical Dataset **2**

2 Life Table Response Experiment **13**

3 Rotifer data analysis **17**

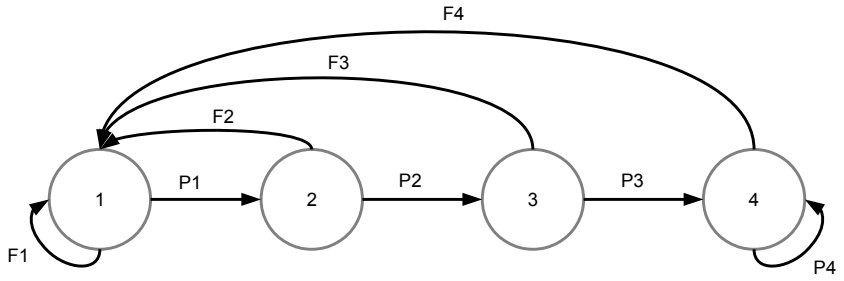
The last practical before the Easter break contains important concept for the analysis of the rotifer data. Therefore the beginning of this practical is the same as the ending of the previous practical to make sure that everybody has the time to finish these exercises.

The completely new material starts at section 1.1.4. In the other sections some minor revisions have been made though.

1 Hypothetical Dataset

1.1 Age-structured Matrix Analysis

In the former practical, we have followed the fate of a hypothetical population of barn owls. The life cycle of this popualtion is given below.



For this population we found the following leslie matrix:

$$\begin{pmatrix} 1.60 & 1.48 & 0.25 & 0 \\ 0.80 & 0 & 0 & 0 \\ 0 & 0.52 & 0 & 0 \\ 0 & 0 & 0.25 & 0 \end{pmatrix} \tag{1}$$

1.1.1 Asymptotic behaviour

Asymptotic rate of increase

From the matrix, we can find the asymptotic growth rate of that population. For this we need to look at the dominant eigenvalue of the population matrix. Remember from the lecture that the dominant eigenvalue of a population matrix is the long term asymptotic rate of increase, λ. An eigenvalue of a matrix is a **scalar that when multiplied by a specific vector, gives the same result as when that vector is multiplied by the matrix**. Mathematically this means:

$$\boldsymbol{A}\vec{w} = \lambda\vec{w} \tag{2}$$

the vector $\vec{w}$ is not any vector but an eigenvector, that is a vector such that equation 2 holds. We will come back to it later. If you do not feel comfortable with the concept of eigenvectors/matrices, you might want to return to the last exercise of the mathematical tools script.

In `R` we do not have to just see the eigenvectors, instead they can be found using the `eigen()` function.

```
eigens.A<-eigen(A)
eigens.A
```

Let us for the moment only focus on the first set of outputted values. These values $(2.168, -0.465, -0.103, 0)$ are the eigenvalues of the matrix $\boldsymbol{A}$. A matrix has as many eigenvalues as it has dimensions. The only one we are interested in is the dominant eigenvalue, i.e. the largest eigenvalue. From looking at the eigenvalues, we see that this is the first one. In general the dominant eigenvalue can be found by using the function `which.max()` that returns the position of the maximum value of an object. Next we extract the value in that position.

```
position<-which.max(eigens.A$values)
position

lambda2<-eigens.A$values[position]
lambda2
```

Alternatively, λ can be found directly by using the `lambda()` funtion of the `popbio` package in `R`. This will give exactly the same result.

```
library(popbio)

## Loading required package:  quadprog

lambda(A)

## [1] 2.168198
```

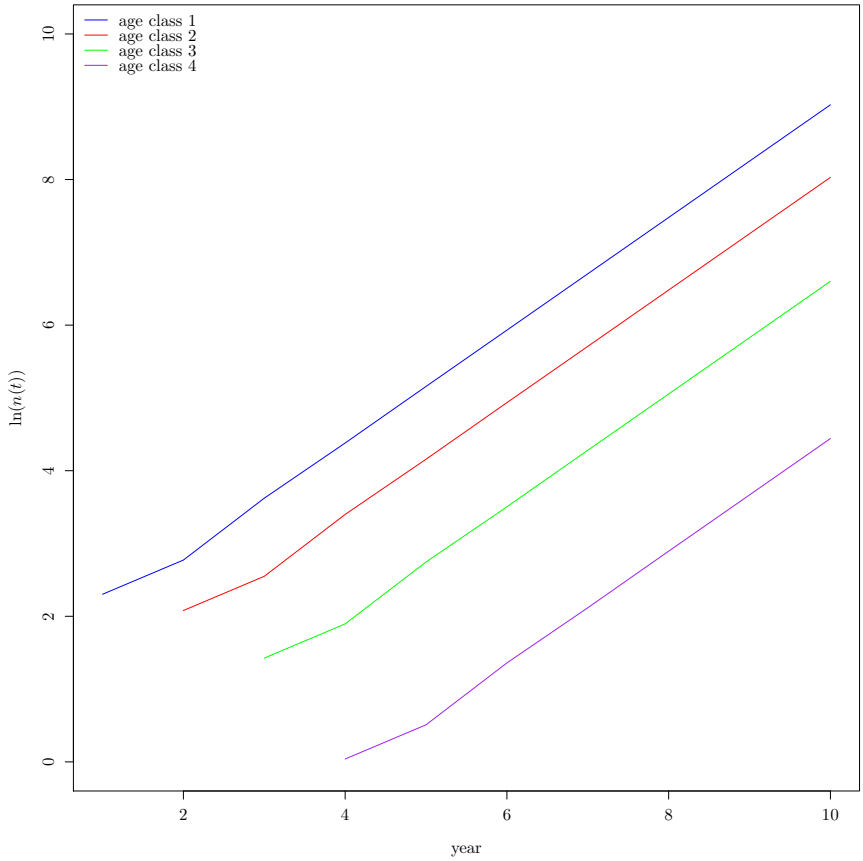
Thus our asymptotic finite rate of increase λ is 2.1681981. This is what we expect the population growth rate to look like on the long-term.

Stable-age distribution

In the previous section, we looked at the growth of the whole population. It is also informative to examine what the population structure will look like on the long-term. We already calculated these numbers in the previous practical, let us now plot them.

```
plot(log(n[1,]), type="l", col="blue", ylim=c(0,5),
     xlab="year", ylab="ln(n(t))")
```

```
lines(log(n[2,]), col="red")
lines(log(n[3,]), col="green")
lines(log(n[4,]), col="purple")
legend("topright",
       c("age class 1 ","age class 2", "age class 3",
         "age class 4"),
       bty = "n",
lty=c(1,1,1,1),
col=c("blue", "red", "green", "purple"))
```



When reading the graph, keep in mind that the y-axis is logarithmic. We see that after five years, the age class trajectories are parallel to each other and growing with a constant rate. If a population is growing with constant birth and death rate, as is the case of our owl population, then the population converges to a stable age distribution. Once a population has reached the stable age distribution, it grows exponentially at the asymptotic finite rate λ. A population at the stable age distribution always has the same relative number of individuals

in each age class every time step. However, the absolute number of indiviuals will change. The stationary age distribution is a special case of stable age distribution where the absolute number of individuals does neither increase nor decrease. The population is not growing. What is the asymptotic growth rate (λ) of such a stationary population?

The stable age distibution is given by the right eigenvector corresponding to the dominant eigenvalue. Let us now look at the vector elements of the `eigen` function. The columns of `eigens.A$ vector` are the right eigenvectors of the matrix $\boldsymbol{A}$. When applying the matrix to them, it is the same as multiplying them by the corresponding eigenvalues, which are scalars. This means that the length but not the direction of this vector is changed by the matrix. Check this for the eigenvector corresponding to the dominant eigenvalue.

There are as many right eigenvectors as there are eigenvalues. To each eigenvalue one eigenvector corresponds. Its position is the same as the one of the dominant eigenvalue so we can extract it with the following lines of code.

```
w<-eigens.A$vector[,position]
w
```

The right eigenvector as such does not give us the proportion found in each age class. This is because in the vector that `R` returns, the length of the eigenvectors is set to 1 instead of the sum of its elements (what is the difference between these two? Which is biologically more relevant? Which is mathematically more relevant?). However, to find the proportion of each age class, we want to rescale it (remember that if you find an eigenvector all vectors that point in the same direction are also eigenvectors. If we multiply an eigenvector by a scalar it thus remains an eigenvector.):

```
ssd<-w/sum(w)
ssd
```

Therefore we know now that in the stable age structure a proportion of 0.68 of the population is in age class 1, 0.25 in age class 2, 0.06 in age class 3, and 0.01 in age class 4.

1.1.2 Sensitivity and elasticity analysis

Sensitivities and elasticities of λ evaluate the relative importance of each matrix element for the asymptotic growth rate. Sensitivity is the effect of additive change in a matrix element on λ and the elasticity is the proportional effect of a proportional change in the parameter to λ.

Sensitivity analysis

The sensitivity tells us how an increase in absolute value of one of the parameters affects λ. How much does λ increase if the fertility of the age class 1 increases by a very small amount (for example 0.01)? Let us play around with the projection matrix and change each element of this matrix by 0.01 one by one to see how if affects λ and store these different values of λ.

```
# Build the Leslie Matrix for the barn owl population

F11<-    + 0.01 #insert your value here
F21<- #insert your value here
F31<- #insert your value here
F41<- #insert your value here

P11<- #insert your value here
P21<- #insert your value here
P31<- #insert your value here
P41<- #insert your value here

A1<-matrix(c(F11,P11,0,0,F21,0,P21,0,F31,0,0,P31,F41,0,0,P41),
           nr=4)
A1
lambdaF1<- lambda(A1)
```

You can see that by changing the parameter values, the value of λ changes. The sensitivity of λ to one element of the matrix is given by:

$$s_{ij} = \frac{\partial \lambda}{\partial a_{ij}} \tag{3}$$

Remember the first practical on mathematical tools, the section on the derivatives. Instead of taking the derivative, equation 3 can be approximated by taking the difference between λ of the original matrix and λ the modified matrix, and dividing it by the change we apply to the element of the matrix.

- $\partial \lambda$ can be approximated by the difference between the original λ and the new λ (after changing one of the elements).
- ∂a_{ij} can be approximated by the difference between the original a_{ij} and the new a_{ij}. In our `R` code, this is 0.01.

If the change to the element of the matrix is very small, the sensitivities calucated so will be close to the actual sensitivities. Try to approximate the sensitivities of λ to a few of the matrix elements in `R`. Then compare your results with the actual sensititvities presented in the table below. The sensitivities in the table have been calculated with the function `sensitivty` from the `popbio` library.

n_1	n_2	n_3	n_4
0.7860	0.2900	0.0696	0.0080
0.5583	0.2060	0.0494	0.0057
0.0906	0.0334	0.0080	0.0009
0.0000	0.0000	0.0000	0.0000

If you want to calculate all sensitivities using the approximation presented above, it is probably best to use a `for` loop (or actually two, why two?). Do not forget to only change one matrix element at a time. Also investigate how small the change in the matrix element has to be to make sure that the approximation returns reasonable numbers. Change the values of a and compare your sensitivities with the one in the table above. What do you see?

```
a<-0.01
Sen<-matrix(NA,4,4)
lamA<-lambda(A)
for (i in ...){
  for (j in ...){
  Ap<-A
  Ap[i,j]<-... # Create the matrix where only one
               # element of the matrix (the element
               #i,j) is changed at a time.
  Sen[i,j]<-... # Find the sensitivity to the
               # element of the matrix modified in
               # Ap one by one from the differences
               # in lambda over the change in aij.
}
}
Sen
```

Elasiticity analysis

Elasticities are proportional sensitivities. Before we asked the question: if a matrix element increases by a small amount, how much does λ increase? Elasticities ask a different question: with what factor does λ change if a matrix element is changed by a certain factor. They thus examine the effect of a proportional change of transition elements on λ. For example, you may be interested in knowing by how much does λ change if the fertility of age class 2 is increased by 25%.

The elasticity of λ to a proporitonal change of one element of the matrix is given by:

$$e_{i,j} = \frac{a_{i,j}}{\lambda}\frac{\partial\lambda}{\partial a_{i,j}} = \frac{a_{i,j}}{\partial a_{i,j}}\frac{\partial\lambda}{\lambda} = \frac{\frac{\partial\lambda}{\lambda}}{\frac{\partial a_{i,j}}{a_{i,j}}} \tag{4}$$

You can thus calculate the elasticities from your approximations of the sensitivities. To do so, focus on the first part of the equation: $e_{i,j} = \frac{a_{i,j}}{\lambda}\frac{\partial\lambda}{\partial a_{i,j}}$. Compare your values with the values below, which were calculated with the `elasticity()` function of the `popbio` package.

n_1	n_2	n_3	n_4
0.5800	0.1980	0.0080	0.0000
0.2060	0.0000	0.0000	0.0000
0.0000	0.0080	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000

1.1.3 Transient dynamics

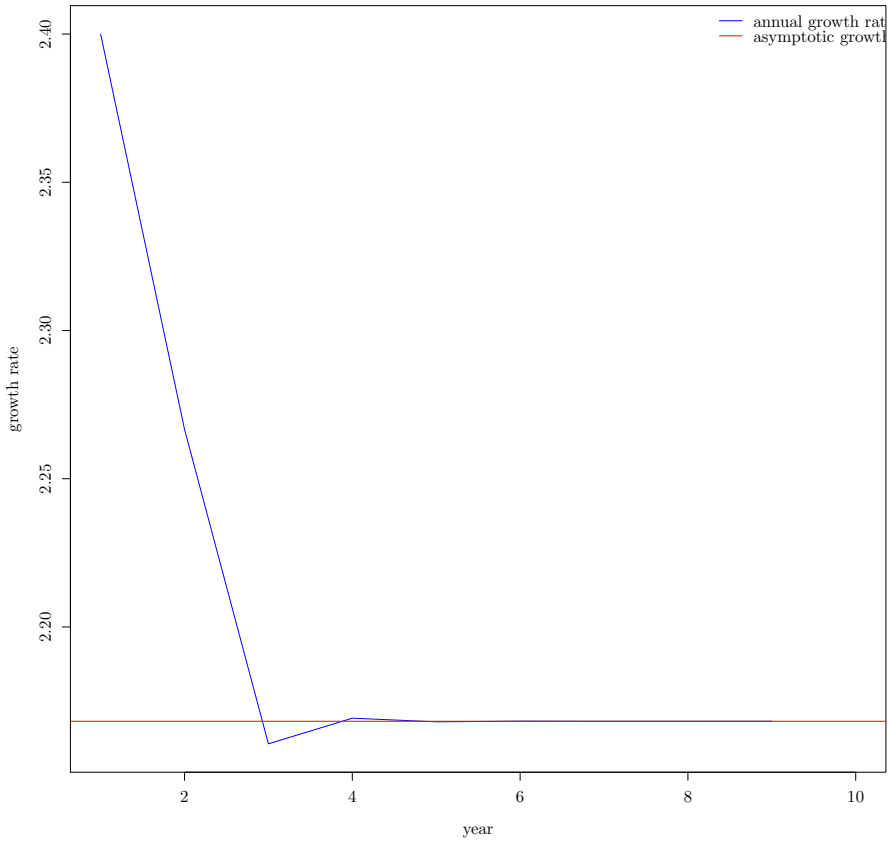
Most of what we have described above focuses on the asymptotic, (i.e. long-term) behaviour of the system. However, the short-term dynamics can be very different. In nature, a population may never be observed in the asymptotic regime. Transient dynamics focus on short-term responses.

There are year-to-year variations in growth rate before the asymptotic growth rate is reached. The code below calculates the annual finite growth rates of the barn owl population.

```
Ntot # the total population size vector.
     # Each element is the population size
     # at one time step. As calculated before
Ntott<-Ntot[2:(length(Ntot)+1)]
# the total popoluation size at t+1

R<-Ntott/Ntot
# the annual finite growth rates
#vector
```

We can plot this in a graph to see how much time it takes this population to reach the asymptotic finite growth rate. To this plot we add a straight line using the function `abline` to represent the value of λ.

```
lam<-lambda(A)

plot(R, type="l", col="blue", xlab="year", ylab="growth rate")
abline(h=lam, col="red")
legend("topright",
       c("annual growth rate","asymptotic growth rate"),
       bty = "n",
lty=c(1,1),
col=c("blue", "red"))
```



It thus takes approximately five years before the population reaches its asymptotic growth rate.

1.1.4 Reproductive values

The reproductive value vector represents the contribution of each individual to present and future reproduction. In other words, it gives how much an individual in a given age class contributes to the population on the long-term, including offspring of its offspring. That is: if we have a population and its matrix at time $t = 1$, what proportion of individuals at a much later time will be descendants of individuals in age class 1 at time $t = 1$ and how many from individuals in age class 2 etc. Mathematically, the reproductive value vector is the left eigenvector of the population matrix.

$$\vec{v}A = \lambda\vec{v} \tag{5}$$

Please note that this time the vector is on the left side instead of on the right side of the matrix. This is why this is called a left eigenvector. Multiplication

with the vector on the left side of the matrix looks as follows in 2 dimensions:

$$\begin{pmatrix} v_1 & v_2 \end{pmatrix} \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} = \begin{pmatrix} v_1 a_{1,1} + v_2 a_{2,1} & v_1 a_{1,2} + v_2 a_{2,2} \end{pmatrix} \tag{6}$$

Again, we take the row from the first element (in this case the vector) and multiply its elements by the elements of the first column of the second element (in these case a matrix). Adding up these contributions gives us the value for the first entry of the final vector. Compare this to what you would get if the order of the matrix and the vector would be the other way around.

To get the left eigenvector of a matrix, we need to perform the eigen analysis on the transpose of the projection matrix. (The transpose of a matrix is the matrix we get by exchanging the rows and the columns of that matrix. For example for the matrix $\boldsymbol{B} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, $\boldsymbol{B}^T$ is the transpose of matrix $\boldsymbol{B}$ and $\boldsymbol{B}^T = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$). More visually: when you transpose a matrix, you mirror its elements in the diagonal of that matrix. `t()` is the `R` function that transposes a matrix. If you do not remember what the function `eigen` does, reread the subsection *asymptotic rate of increase* or aks `R` (`?eigen`).

```
At<-t(A)
eigens.At<-eigen(At)
eigens.At
```

We apply the same proceedure as for the right eigenvector to find the left eigenvector and the eigenvalue. Write the code to extract the eigenvalue and the left eigenvector. Do you notice anything about the eigenvalue? Check that the found dominant left eigenvector is indeed a left eigenvector of the matrix with the dominant eigenvalue as its eigenvalue.

Additional information: Reproductive values more in depth
If you want to understand the interpretation of the reproductive values more closely, take a look at the following (not too elegant) code:

```r
B<-matrix(4*runif(9),nrow=3) # We generate a random matrix

B

##          [,1]     [,2]     [,3]
## [1,] 1.939894 1.657795 3.731020
## [2,] 2.315226 3.360205 1.176907
## [3,] 2.556621 2.080008 2.551491

# We output B, in case you want to repeat the
# simulation yourself

eigen(t(B))$vectors[,1]/sum(eigen(t(B))$vectors[,1])

## [1] 0.3200096 0.3329669 0.3470236

# We calculate the dominant left eigenvector of this matrix
# and normalise it with respect to the sum of its elements

# Now we want to investigate what would happen to the
# following 3 initial populations:
# 1) (1,0,0)  2) (0,1,0)  3) (0,0,1)

N1<-rep(NA,3)
# Create an empty vector for storing the final population
# size if we start with any of the three initial populations
for(i in 1:3){
  N<-rep(0,3)
  N[i]<-1
  # Set the initial population vector
  # the first time it will be (1,0,0)
  # then (0,1,0) and finally (0,0,1)

  for(t in 1:40){
    # we project the population for 40 timesteps
    N<-B%*%N
  }
  N1[i]<-sum(N)
  # We store the total population size after 40 timesteps
  # in the vector N1
}
N1/sum(N1)

## [1] 0.3200096 0.3329669 0.3470236

# We output the normalised version of N1, which should be
# equal to the left eigenvector, do check this!
```

You can interpret this as follows: if we for example start with a population $\begin{pmatrix}1\\1\\1\end{pmatrix}$, after 40 years, 32% of the total population will have originated from the individual that was initially in age class 1, 33% from the individual in age class 2 and the remaining 35% of the total population at timestep 40 from the individual that was in age class 3 in the initial population. *Please not that since we took completely random numbers to construct the matrix, the matrix is unlikely to have biological relevance. But the concept of course remains the same for more relevant matrices.*

However, what would happen if the initial population would look as follows: $\begin{pmatrix}2\\1\\1\end{pmatrix}$? If we project this initial population in the future we find (the code is omitted this time):

```
## [1] 0.4848595 0.2522458 0.2628947
```

This means that after 40 years, 48% of the total population will have originated from the two individuals that were initially in age class 1, 25% from the individual in age class 2 and the remaining 26% of the total population at timestep 40 from the individual that was in age class 3 in the initial population. These numbers do not correspond to the left eigenvector, yet they do relate to the left eigenvalue. First we need to realize that if 48% of the final population originated from the two individuals in age class 1 in the initial population, this means that each of these two individuals is responsible for 24% of the final population. It is of course fairer to compare these numbers with each other (so after correcting for the number of individuals that we started with):

$$\begin{pmatrix}24\\25\\26\end{pmatrix}$$

However, now the numbers no longer add up to 100%. Since all we care about is the ratio between the different contributions, we can just divide all of them by the sum of the three contributions ($\approx 24 + 25 + 26$) and multiply the total by 100 (just to make sure that all the percentages add up to 100). What we get now is:

$$\begin{pmatrix}32\\33.3\\34.7\end{pmatrix}$$

This is indeed again the left eigenvector! (Hopefully. Every time we save this document new random numbers are generated, so actually we can only hope that the theory is general enough to work for all random matrices `B` that are accidentally generated...). We see thus that the left eigenvector actually mainly tell you how many more offspring an individual that starts in a certain age class will have compared to individuals in other age classes on the long term. – This is indeed a relatively difficult concept, but maybe if you think about it and go over this simulation yourself, it might sink in.

2 Life Table Response Experiment

Life table response experiments (LTREs) are used to assess which differences in vital rates lead to a change in λ in an experimental design. The experiment you performed on the rotifers is well suited to be analysed by an LTRE. More information on the LTREs can be find in Caswell (2001) *Matrix population models*, this is also the book that we based the theoretical background of this practical on.

2.1 One-way fixed designs

A one-way fixed effect is applied on data with one treatment of several levels. For example, let us examine two populations of yellow-necked mice.

Figure 1: yellow-necked mouse, from wikipedia.

This species lives mostly in woodlands and it is suspected that its distribution is limited by altitude. Let us compare a population living in the mountain (population $\boldsymbol{M}$) with a population living in the plain (population $\boldsymbol{P}$) to see if the altitude is a limiting distribution factor. This species can be described by a life-cylce in two stages; juveniles and adults. Thus the matrices describing those populations are given by:

$$\boldsymbol{P} = \begin{pmatrix} 0 & 2 \\ 0.25 & 0.5 \end{pmatrix} \tag{7}$$

and

$$\boldsymbol{M} = \begin{pmatrix} 0 & 1.9 \\ 0.2 & 0.45 \end{pmatrix} \tag{8}$$

Find the asymptotic growth rates of these populations. You can see that they differ. We would now like to investigate which matrix elements contribute the

most to the difference in the asymptotic growth rates. For this we describe the asymptotic growth rate of the mountain population matrix, $\boldsymbol{M}$, as a function of the asymptotic growth rate of the plain population matrix, $\boldsymbol{P}$, our reference population plus a treatment effect:

$$\lambda^{(M)} \approx \lambda^{(P)} + \sum_{i,j} (a_{ij}^{(M)} - a_{ij}^{(P)}) \frac{\partial \lambda^A}{\partial a_{ij}^A} \tag{9}$$

Let us work through the right part of this equation together.

- $\lambda^{(P)}$ is the asymptotic growth rate of the $\boldsymbol{P}$ matrix.
- The term $(a_{ij}^{(M)} - a_{ij}^{(P)})$, is the change in the elements of the matrix due to the treatment effect, here the moutain habitat. It tells us how different an element in matrix $\boldsymbol{M}$ is from the element at the same position in matrix $\boldsymbol{P}$.
- The last part, $\frac{\partial \lambda^A}{\partial a_{ij}^A}$, is the sensitivities of the asymptotic growth rate of a "mid-way" matrix to elements of that "mid-way" matrix. This matrix is the mean between $\boldsymbol{P}$ and $\boldsymbol{M}$ and is thus given by

  $$\boldsymbol{A} = \frac{\boldsymbol{M} + \boldsymbol{P}}{2} \tag{10}$$

  The matrix $\boldsymbol{A}$ is used because we need a matrix to compare matrices $\boldsymbol{P}$ and $\boldsymbol{M}$ against. It is possible to use either matrix $\boldsymbol{P}$ or $\boldsymbol{M}$ instead but this would give more weight to the selected matrix. Therefore we use the matrix that lies just in between.
- The multiplication of the sensitivities with the summation term defines how much the change in each elements of the matrix due to the treatment affects the asymptotic growth rate. In other word they are the contributions of the a_{ij} to the effect of the habitat on the growth. It is necessary to do this because for example, a large difference between the elements in the same position may in fact have little effect on the growth if the senstitivity for this position is low.

You may have recognized that equation 9 is a linear equation ($y = b + ax$). This method makes the assumption that the relationship between the matrices is linear and that the slope of this equation is given by $\frac{\partial \lambda^A}{\partial a_{ij}^A}$.

With equation 9, find the value of $\lambda^{(\boldsymbol{M})}$ using `R` and compare it to the value you found earlier. How different are they? More interestingly: which difference in the matrix elements is mainly responsible for this difference? Is it the juvenile survival, the adult survival or the reproductive value?

Hint Find the "mid-way" matrix. For the sensitivities, either approximate them yourself or use the `sensitivity()` function from the `popbio` package

2.2 Fixed Factorial Designs (Theory)

Factorial LTRE allows the examination of the effects of several treatments and their interactions. This is the case for your rotifer data where you have a "layer" treatment, a "pollution" treatment and a "species" treatment. Not all the combinations were investigated though; the layers "recovery" and "pollution" contained only the species BU whereas the layers "commercial" and "postpollution" contained only species BC. What difficulty would this pose if you want to compare the species to each other? Because of this, we cannot make a full factorial LTRE on the rotifer data, but we will examine subsets of the full dataset.

For example let us consider the species BU which has two treatments of the type "layer": the layers "recovery" (r) and "pollution" (p). For each of these layers it has 3 possible levels for the copper treatment: "low" (l), "medium" (m) and "high" (h). We thus have two factors, one with two levels, the second with three levels.

A factorial LTRE is similar to a one-way LTRE; we want to find which differences in matrix elements between a focal matrix and a reference matrix contribute the most to the difference in λ between the two matrices. Let us take the matrix "recovery" and "low" which we call $\boldsymbol{M}^{r,l}$ as the focal matrix, for this we average the rates over all the 6 replicates that were subject to this treatment (r, l). We compare it to the matrix naive to any treatment, which we call $\boldsymbol{M}^{\cdot\cdot}$. This matrix is obtained from taking the average rates for all the matrices that we are comparing to each other (that is in this case the average of $\boldsymbol{M}^{r,l}$,$\boldsymbol{M}^{r,m}$,$\boldsymbol{M}^{r,h}$,$\boldsymbol{M}^{p,l}$,$\boldsymbol{M}^{p,m}$ and $\boldsymbol{M}^{p,h}$). Again we are interested in understanding the effect of the different treatments on the asymptotic growth rates.

$$\lambda^{r,l} = \lambda^{\cdot\cdot} + \alpha^{r} + \beta^{l} + (\alpha\beta)^{r,l} \tag{11}$$

Equation 11 tells us that $\lambda^{r,l}$ can be found from $\lambda^{\cdot\cdot}$ plus an effect from the layer "recovery", an effect from the pollution "low" and an interaction between the two treatments.

To isolate the effect of each treatment, we need to look at them separately. So we examine a matrix $\boldsymbol{M}^{r\cdot}$, where the effect of pollution is ignored. The rates of this matrix are calculated as the average of all the matrices with the layer "recovery".

$$\alpha^{r} = \sum_{ij} (a_{ij}^{r\cdot} - a_{ij}^{\cdot\cdot}) \frac{\partial \lambda^{A}}{\partial a_{ij}^{A}} \tag{12}$$

The structure of equation 12 is the same as the last part of equation 9. The matrix $\boldsymbol{A}$ is again a "mid-way" matrix between $\boldsymbol{M}^{r\cdot}$ and $\boldsymbol{M}^{\cdot\cdot}$. α^{r} tells us how large the effect of the treatment "layer" is on the asymptotic growth rate. If we want to know which matrix elements are responsible for this effect, we need to look at the separate contributions to α^{r} $((a_{ij}^{r\cdot} - a_{ij}^{\cdot\cdot}) \frac{\partial \lambda^{A}}{\partial a_{ij}^{A}})$. Analogously we can also calculate α^{p}.

We do the same for the second factor. We examine a matrix $\boldsymbol{M}^{\cdot l}$, where the

effect of the layer is ignored:

$$\beta^l = \sum_{ij}(a_{ij}^{\cdot l} - a_{ij}^{\cdot\cdot})\frac{\partial \lambda^B}{\partial a_{ij}^B} \tag{13}$$

For the interaction effect, we apply the same logic. We examine a matrix $\boldsymbol{M}^{rl}$ where both the effects of the layer and the pollution are taken into account so as to capture the effect of the interaction between the two factors. Because we want to isolate the interaction effect, we need to remove the effects of the layer and copper treatment.

$$(\alpha\beta)^{rl} = \sum_{ij}(a_{ij}^{rl} - a_{ij}^{\cdot\cdot})\frac{\partial \lambda^C}{\partial a_{ij}^C} - \alpha^r - \beta^l \tag{14}$$

So far we have considered only the effect of the layer "recovery" and the pollution "low". Of course this can be extended to all other layers of all treatments. So in general the equations become for k the level of treatment 1 and m the level of treatment 2.

$$\lambda^{k,m} = \lambda^{\cdot\cdot} + \alpha^k + \beta^m + (\alpha\beta)^{k,m} \tag{15}$$

$$\alpha^k = \sum_{ij}(a_{ij}^{k\cdot} - a_{ij}^{\cdot\cdot})\frac{\partial \lambda^A}{\partial a_{ij}^A} \tag{16}$$

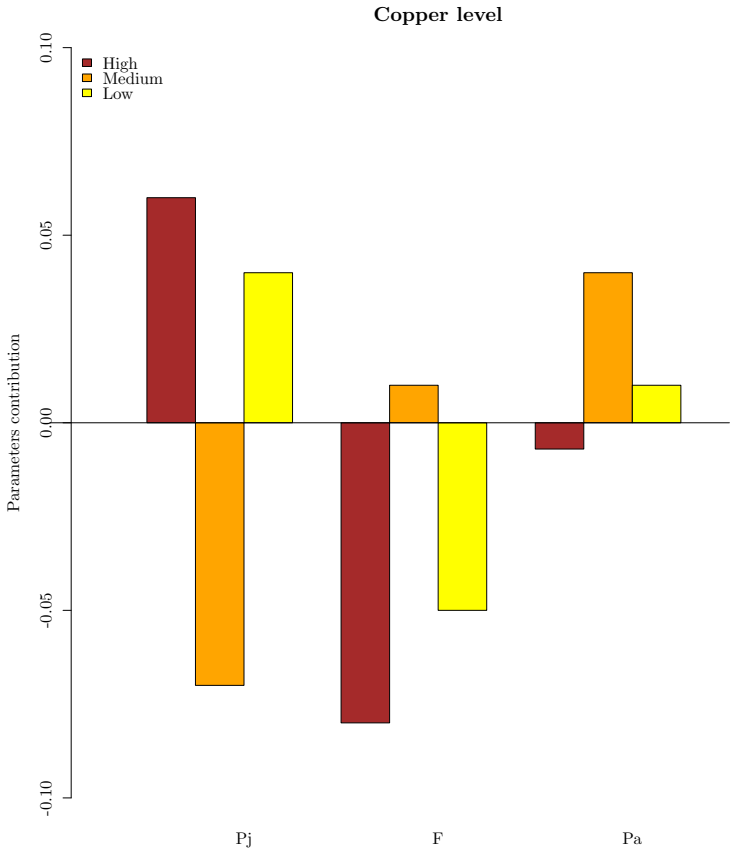
$$\beta^m = \sum_{ij}(a_{ij}^{\cdot m} - a_{ij}^{\cdot\cdot})\frac{\partial \lambda^B}{\partial a_{ij}^B} \tag{17}$$

$$(\alpha\beta)^{km} = \sum_{ij}(a_{ij}^{km} - a_{ij}^{\cdot\cdot})\frac{\partial \lambda^C}{\partial a_{ij}^C} - \alpha^k - \beta^m \tag{18}$$

In these equations:

$$A = \frac{\boldsymbol{M^{k\cdot}} + \boldsymbol{M^{\cdot\cdot}}}{2} \qquad B = \frac{\boldsymbol{M^{\cdot m}} + \boldsymbol{M^{\cdot\cdot}}}{2} \qquad C = \frac{\boldsymbol{M^{km}} + \boldsymbol{M^{\cdot\cdot}}}{2} \tag{19}$$

LTREs provide additional information to the growth rate as it describes how the different treatments affect the growth rate. In particular, LTRE allow a detailed comparison of the effect of a treatment on growth rate by examining the contribution of each survival and fertility rate to the gorwth rates of populations subjected to different treatment levels. For example, the graph below depicts the contribution of each element of populations under different copper levels to the growth rate. Each bar describes what is the contribution of an element to the growth rate when subjected to a given treatment level. What conclusions can you draw from this graph?



3 Rotifer data analysis

In this section, you will apply the factorial LTRE to the rotifer data to compare the layers "pollution" and "recovery". The second factor is the copper treatment with the levels "low", "high", "medium".

1. Load the file that contains the estimated transition rates in `R`, do not forget to set the working directory first.

2. Because in the lab, you have made several replicates for each treatment combination, you now have a rate per replicate. We are however interested in the average rates per treatment. We want one juvenile survival rate, one adult survival rate and one fertility rate per population, per copper level and per species. For this you can use the `aggregate()` function that we have seen in practical 6 and find the mean. Use the function three times seperately, once for each of the rates that we are interested in. Store the

results in a variable. Use `?aggregate` to see how to use the function. You will need to use one special argument: `na.rm=TRUE`. This is to make sure that when there are NAs in the dataset, the function ignores them and still returns a value.

3. Now you have three separate lists. You want to group them. For this you can adapt the following lines of code. Use the names of the variables that you specified in the previous step.

```
younewdata1<-merge(SurvJuv, SurvAdu, by=c(1,2,3))
#by set the column by which to merge the two elements

younewdata2<-merge(younewdata1, Fertility, by=c(1,2,3))
```

You can in addition give a more meaningful name to the columns by using:

```
colnames(younewdata2)<-c("name of col1",
"name of col2", "name of col3",
"name of col4", "name of col5", "name of col6")
```

4. We mentioned that we are only interested at the moment in the layer "Recovery" and "Pollution". Both contained the species BU. Thus, select only the BU species with the `subset` function that we also used in practical 6.

5. Below is a useful function for the LTRE. It allows you to extract a specific matrix per treatment by using `get_matrix`. You need to give it the following arguments 1) the dataset from which you want to extract the matrix, 2) the name of the layer that you are interested in (for example "Pollution"), and 3) the name of the copper level (for example "high"). You do not need to understand the details of this code. If you run this code once during your `R`-session, `R` will remember it and every time you type `get_matrix(data,layer,copper)` it will execute that code.

```
#code for function to extract matrices from the dataset#
get_matrix<-function(roti,pop,cop){
  if(!pop %in% c(levels(roti$Population),"mean")){
    warning("Something went wrong, give the instructors
            a cookie and they may help you out:
            \n------------------------\n ",pop,
        " is not a valid entry\n------------------------\n")

    return()
  }
```

```
  if(!cop %in% c(levels(roti$Copper),"mean")){
    warning("Something went wrong, give the instructors
            a cookie and they may help you out:
            \n-----------------------\n ",cop,"
            is not a valid entry\n-----------------------\n")

    return()
  }

  if(pop=="mean" & cop=="mean"){
    i<-1:length(roti$Copper)
  }else if(pop=="mean"){
    i<-which(roti$Copper==cop)
  }else if(cop=="mean"){
    i<-which(roti$Population==pop)
  }else{
    i<-which(roti$Population==pop & roti$Copper==cop)
  }

  A<-matrix(c(0, mean(roti$Pj[i]),
        mean(roti$F[i]), mean(roti$Pa[i])), nrow=2, ncol=2)
  return(A)
  }
#end of the function#
```

6. With the `get_matrix` function, extract the matrix for each treatment combinations (6 matrices in total). You need to specify the dataset in the first position, in the second position you need to specify the layer and in the third position the copper level.

7. Get the growth rate λ of each of these matrix.

8. Now we get to the LTRE. Let us isolate the effect of the first treatment which is the layer. This factor has two levels; pollution and recovery. What we need to do is to implement equation 16 in R. Let us find the α for the level "pollution" first; α^p.

   (a) Find the matrix $\boldsymbol{M}^{\cdot\cdot}$. The rates of this matrix are the averages of the rates of all the matrices of the BU species. For this you can use the `get_matrix` function. The second argument concerns the layer, but here we want the mean rate over all the layers, we have written the function such that you can achieve this by typing "mean" as an argument for the layer. The same holds for the third argument.

   (b) Next you need the mean pollution matrix, $\boldsymbol{M}^{p\cdot}$. Again, use the `get_matrix` function.

small tip Here you can do a little trick to be able to use the code later on a different subset of your dataset. Before using the `get_matrix` function, set a new name to your focal layers like in the code below. Then in `get_matrix`, instead of calling for "Pollution", call for `yourname1`.

```
yourname1<-'Pollution'
yourname2<-'Recovery'
```

(c) Find the λ for this matrix.

(d) Find the "mid-way" matrix between $\boldsymbol{M}^{\cdot\cdot}$ and $\boldsymbol{M}^{p\cdot}$. This is the mean matrix between $\boldsymbol{M}^{\cdot\cdot}$ and $\boldsymbol{M}^{p\cdot}$.

(e) For equation 16, you need the summation term and the sensitivity of the "mid-way" matrix. Let us look first at the summation term. You need the difference between the elements of matrix $\boldsymbol{M}^{p\cdot}$ and of matrix $\boldsymbol{M}^{\cdot\cdot}$. Thus take $(\boldsymbol{M}^{p\cdot} - \boldsymbol{M}^{\cdot\cdot})$.

(f) To find the sensitivity of the "mid-way" matrix, use the function `sensitivity` from the package `popbio` (see subsection *sensitivity*).

(g) Finally, multiply the substraction matrix by the sensitivity matrix (not a matrix multiplication!) and save this in a matrix.

(h) The sum of this matrix is your α^p, the effect of the layer "Pollution" on the asymptotic growth rate. The separate entries of this matrix show how much each element of the population matrices contributes to the difference in λ between them.

(i) Store this matrix because the different elements of the matrix are what we are ultimately interested in. Check whether the effect you found indeed explains the difference between $\boldsymbol{M}^{\cdot\cdot}$ and $\boldsymbol{M}^{p\cdot}$.

9. Repeat the same proceedure to find the second α, α^r and store the separate contributions $(a_{ij}^{k} - a_{ij}^{\cdot\cdot})\frac{\partial \lambda^A}{\partial a_{ij}^A}$. DO the same for the three values for β and finally for the interactions. For the separate contribution of the interactions $(a_{ij}^{km} - a_{ij}^{\cdot\cdot})\frac{\partial \lambda^C}{\partial a_{ij}^C}$, do not forget to substract the corresponding contributions from the α and β.

10. Once you have all the contributions of all the treatments, you may want to compare them. A possible graphs is a bar plot representing on the same graph the contribution to each parameter of each level of a factor, see for an example the graph above. If you agree that these sort of graphs are interesting, you can follow the next instructions. Feel free however to follow a different logic to produce the same graphs or to present your results in a different way. We do not claim to provide the most efficient code; any comments on how to improve the code are welcome.

    (a) Let us first look at the effect of the layer on the rates. One way of doing this is first to transform your matrices into vectors, using the

`as.vector()` function and then combine the vector of each matrix into an array with `cbind()`. (you can actually ignore the first element of the matrices, the transition from juvenile to juvenile because it is always zero by selecting the vector elements 2 to 4). The columns are the layers and the row the rates.

(b) You can now use the `barplot()` command to plot bar graphs. Specify `beside=TRUE` to see the bars next to each other rather than on top of each other.

(c) You may see that each bar (color) represents a rate although it would be more informative if each bar would be a layer. To correct for this, you can take the transpose of the array by using the function `t()` (we have seen this earlier this practical) and do the bar plot of the transpose.

(d) There are a few arguments you may want to use in the `barplot()` command to improve the visual appearance of the graph such as `xlab=`, `ylab=`, `xlim=c()`, `ylim=c()`, `col=c("colorname")`, `main="yourtitle"`.

(e) Additionally, you may want to specify a legend and an axis. This you can do with the commands `legend()` and `axis()`. Look them up in the help section to see how to use them.

(f) Don't forget to save your figures! A detailed description is given at the end of practical 6.

11. Repeat all of the steps above for the combinations of the layers "Commercial" & "Postpollution" and of the layers "Postpollution" & "Pollution".

If you manage to go through all the steps describe above, you should now know the growth rate of your populations, have performed three LTRE analyses on your rotifer data and have plots to include in your report. The interpretation of the results is up to you.

Good luck!

```
#Open data
rm(list = ls())
rot<-read.csv("BIO311_with_rates.csv", sep=",")
str(rot)

## 'data.frame': 72 obs. of  13 variables:
##  $ Group.ID   : Factor w/ 6 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Population : Factor w/ 4 levels "Commercial","Pollution",..: 1 1 1 2 2 3 3 3 3 4 ...
##  $ Species    : Factor w/ 2 levels "BC","BU": 1 1 1 2 2 1 1 1 1 2 ...
##  $ Copper     : Factor w/ 3 levels "high","low","medium": 2 3 1 2 1 2 3 3 1 2 ...
##  $ Replicate  : int  1 6 4 2 3 6 2 4 2 1 ...
##  $ Day        : int  2 2 2 2 2 2 2 2 2 2 ...
```

Figure 2: random picture, from wikipedia.

```
##  $ Alive_Juv  : int  6 5 5 1 1 0 1 3 1 2 ...
##  $ Alive_Adult: int  7 8 7 5 3 2 3 4 2 6 ...
##  $ Dead_Juv   : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ Dead_Adult : int  2 0 4 0 2 4 2 1 1 0 ...
##  $ Pj         : num  0.833 1 1 0 0 0 1 0 0 0.5 ...
##  $ Pa         : num  0.714 1 0.714 0 0.667 0 1 0.75 0.5 0.833 ...
##  $ F          : num  8 1.625 2 0 0.333 ...

#Find the avg rate per treatment
Pj<-aggregate(rot$Pj, list(rot$Population, rot$Copper, rot$Species), na.rm=TRUE, FUN="mean")
Pj

##          Group.1 Group.2 Group.3         x
## 1     Commercial    high      BC 0.8916667
## 2  Postpollution    high      BC 0.4166667
## 3     Commercial     low      BC 0.8555000
## 4  Postpollution     low      BC 0.2500000
## 5     Commercial  medium      BC 0.9206667
## 6  Postpollution  medium      BC 0.4000000
## 7      Pollution    high      BU 0.5832500
## 8       Recovery    high      BU 0.7221667
## 9      Pollution     low      BU 0.5833333
## 10      Recovery     low      BU 0.5833333
## 11     Pollution  medium      BU 0.3195000
## 12      Recovery  medium      BU 0.2000000

Pa<-aggregate(rot$Pa, list(rot$Population, rot$Copper, rot$Species), na.rm=TRUE, FUN="mean")
Pa

##          Group.1 Group.2 Group.3         x
## 1     Commercial    high      BC 0.9523333
## 2  Postpollution    high      BC 0.5833333
```

```
## 3     Commercial     low      BC 0.9523333
## 4  Postpollution     low      BC 0.5832500
## 5     Commercial  medium      BC 0.9583333
## 6  Postpollution  medium      BC 0.5833333
## 7      Pollution    high      BU 0.8334000
## 8       Recovery    high      BU 0.9583333
## 9      Pollution     low      BU 0.6695000
## 10      Recovery     low      BU 0.8888333
## 11     Pollution  medium      BU 0.8790000
## 12      Recovery  medium      BU 0.7111667
```

```
F<-aggregate(rot$F, list(rot$Population, rot$Copper, rot$Species), na.rm=TRUE, FUN="mean")
F
```

```
##          Group.1 Group.2 Group.3         x
## 1     Commercial    high      BC 2.8556667
## 2  Postpollution    high      BC 0.5750000
## 3     Commercial     low      BC 3.3523333
## 4  Postpollution     low      BC 1.0000000
## 5     Commercial  medium      BC 2.2175000
## 6  Postpollution  medium      BC 1.1527778
## 7      Pollution    high      BU 0.5126000
## 8       Recovery    high      BU 0.5973333
## 9      Pollution     low      BU 0.8451667
## 10      Recovery     low      BU 0.8888333
## 11     Pollution  medium      BU 1.2013333
## 12      Recovery  medium      BU 0.5911667
```

```
Surv<-merge(Pj, Pa,  by=c(1,2,3))
Surv
```

```
##          Group.1 Group.2 Group.3       x.x       x.y
## 1     Commercial    high      BC 0.8916667 0.9523333
## 2     Commercial     low      BC 0.8555000 0.9523333
## 3     Commercial  medium      BC 0.9206667 0.9583333
## 4      Pollution    high      BU 0.5832500 0.8334000
## 5      Pollution     low      BU 0.5833333 0.6695000
## 6      Pollution  medium      BU 0.3195000 0.8790000
## 7  Postpollution    high      BC 0.4166667 0.5833333
## 8  Postpollution     low      BC 0.2500000 0.5832500
## 9  Postpollution  medium      BC 0.4000000 0.5833333
## 10      Recovery    high      BU 0.7221667 0.9583333
## 11      Recovery     low      BU 0.5833333 0.8888333
## 12      Recovery  medium      BU 0.2000000 0.7111667
```

```
# Merge it into one dataset
```

```r
Roti<-merge(Surv, F, by=c(1,2,3))

colnames(Roti)<- c("Population", "Copper", "Species", "Pj", "Pa", "F")
Roti
```

```
##       Population Copper Species        Pj        Pa
## 1     Commercial   high      BC 0.8916667 0.9523333
## 2     Commercial    low      BC 0.8555000 0.9523333
## 3     Commercial medium      BC 0.9206667 0.9583333
## 4      Pollution   high      BU 0.5832500 0.8334000
## 5      Pollution    low      BU 0.5833333 0.6695000
## 6      Pollution medium      BU 0.3195000 0.8790000
## 7  Postpollution   high      BC 0.4166667 0.5833333
## 8  Postpollution    low      BC 0.2500000 0.5832500
## 9  Postpollution medium      BC 0.4000000 0.5833333
## 10      Recovery   high      BU 0.7221667 0.9583333
## 11      Recovery    low      BU 0.5833333 0.8888333
## 12      Recovery medium      BU 0.2000000 0.7111667
##            F
## 1  2.8556667
## 2  3.3523333
## 3  2.2175000
## 4  0.5126000
## 5  0.8451667
## 6  1.2013333
## 7  0.5750000
## 8  1.0000000
## 9  1.1527778
## 10 0.5973333
## 11 0.8888333
## 12 0.5911667
```

```r
#Finding the lambda for each matrix (optional)
Roti$lambda<-NA

library('popbio')
for (i in 1:length(Roti$Pj)){
  M<-matrix(c(0, Roti$Pj[i], Roti$F[i], Roti$Pa[i]), nrow=2, ncol=2)

  Roti$lambda[i]<-lambda(M)
}
 Roti
```

```
##       Population Copper Species        Pj        Pa
## 1     Commercial   high      BC 0.8916667 0.9523333
## 2     Commercial    low      BC 0.8555000 0.9523333
```

```
## 3     Commercial medium      BC 0.9206667 0.9583333
## 4      Pollution   high      BU 0.5832500 0.8334000
## 5      Pollution    low      BU 0.5833333 0.6695000
## 6      Pollution medium      BU 0.3195000 0.8790000
## 7  Postpollution   high      BC 0.4166667 0.5833333
## 8  Postpollution    low      BC 0.2500000 0.5832500
## 9  Postpollution medium      BC 0.4000000 0.5833333
## 10      Recovery   high      BU 0.7221667 0.9583333
## 11      Recovery    low      BU 0.5833333 0.8888333
## 12      Recovery medium      BU 0.2000000 0.7111667
##            F    lambda
## 1  2.8556667 2.1414106
## 2  3.3523333 2.2353301
## 3  2.2175000 1.9862098
## 4  0.5126000 1.1041684
## 5  0.8451667 1.1126134
## 6  1.2013333 1.1990961
## 7  0.5750000 0.8614498
## 8  1.0000000 0.8704558
## 9  1.1527778 1.0307070
## 10 0.5973333 1.2921703
## 11 0.8888333 1.2905799
## 12 0.5911667 0.8502275
```

```
#----------- code for function to extract matrices from the dataset-----------#
get_matrix<-function(roti,pop,cop){
  if(!pop %in% c(levels(roti$Population),"mean")){
    warning("Something went wrong, give the instructors a cookie and they may help you out:\

    return()
  }

  if(!cop %in% c(levels(roti$Copper),"mean")){
    warning("Something went wrong, give the instructors a cookie and they may help you out:\

    return()
  }

  if(pop=="mean" & cop=="mean"){
    i<-1:length(roti$Copper)
  }else if(pop=="mean"){
    i<-which(roti$Copper==cop)
  }else if(cop=="mean"){
    i<-which(roti$Population==pop)
  }else{
    i<-which(roti$Population==pop & roti$Copper==cop)
```

```
  }

  A<-matrix(c(0, mean(roti$Pj[i]), mean(roti$F[i]), mean(roti$Pa[i])), nrow=2, ncol=2)
  return(A)
  }
#--------------------------end of the function-----------------#

#Get the matrices we are interested in (for Species BC)
library(popbio)
a<-'Pollution'
ash<-'Poll' # Short name for layer a
b<-'Recovery'
bsh<-'Rec' # Short name for layer b

Rotisub <- subset(Roti, Roti$Population==a | Roti$Population==b)
Rotisub
```

```
##    Population Copper Species        Pj        Pa         F
## 4   Pollution   high      BU 0.5832500 0.8334000 0.5126000
## 5   Pollution    low      BU 0.5833333 0.6695000 0.8451667
## 6   Pollution medium      BU 0.3195000 0.8790000 1.2013333
## 10   Recovery   high      BU 0.7221667 0.9583333 0.5973333
## 11   Recovery    low      BU 0.5833333 0.8888333 0.8888333
## 12   Recovery medium      BU 0.2000000 0.7111667 0.5911667
##       lambda
## 4  1.1041684
## 5  1.1126134
## 6  1.1990961
## 10 1.2921703
## 11 1.2905799
## 12 0.8502275
```

```
#Here let us do the LTRE
#two factors: population (1) and Copper (2)
#The average matrix:
M<-get_matrix(Rotisub, "mean", "mean")

#First the effect of population (1)-> alpha
Pop1<-get_matrix(Rotisub, a, "mean")
#Population level 3->Postpollution
#Population level 1->Pollution
     #Pollution
       #The mid-way matrix
A<-((M+Pop1)/2)
alpha1<-(Pop1-M)*sensitivity(A)
alpha1
```

```
##               [,1]        [,2]
## [1,]  0.000000000  0.02648626
## [2,] -0.001746024 -0.02259442

#Recovery
Pop2<-get_matrix(Rotisub, b, "mean")
       #The mid-way matrix
B<-((M+Pop2)/2)
alpha2<-(Pop2-M)*sensitivity(B)
alpha2
```

```
##              [,1]        [,2]
## [1,] 0.000000000 -0.02727686
## [2,] 0.001610043  0.02307102

#Second the effect of Copper (2)-> beta
hig<-get_matrix(Rotisub, "mean", "high")
med<-get_matrix(Rotisub, "mean", "medium")
low<-get_matrix(Rotisub, "mean", "low")
#Copper level 1->High
#Copper leve 2->Medium
#Copper leve 3->Low
     #High
       #The mid-way matrix
#for beta:
C<-((M+hig)/2)
beta1<-(hig-M)*sensitivity(C)
beta1
```

```
##             [,1]        [,2]
## [1,] 0.00000000 -0.08325026
## [2,] 0.06794042  0.05693927

  #Medium
       #The mid-way matrix
#for beta:
D<-((M+med)/2)

beta2<-(med-M)*sensitivity(D)
beta2
```

```
##             [,1]        [,2]
## [1,]  0.0000000  0.03379360
## [2,] -0.1438248 -0.02240382

  #Low
       #The mid-way matrix
```

```r
#for beta:
E<-((M+low)/2)

beta3<-(low-M)*sensitivity(E)
beta3
```

```
##             [,1]        [,2]
## [1,] 0.00000000  0.03280523
## [2,] 0.04469452 -0.03349652
```

```r
#Interactions

#Here let us extract the 6 matrices we need:
Pop1h<-get_matrix(Rotisub, a, "high")
Pop1m<-get_matrix(Rotisub, a, "medium")
Pop1l<-get_matrix(Rotisub, a, "low")

Pop2h<-get_matrix(Rotisub, b, "high")
Pop2m<-get_matrix(Rotisub, b, "medium")
Pop2l<-get_matrix(Rotisub, b, "low")

       #The mid-way matrix
#for alpha:beta:
F<-((M+Pop1h)/2)

inter1<-((Pop1h-M)*sensitivity(F))-alpha1-beta1
inter1
```

```
##             [,1]        [,2]
## [1,]  0.0000000 -0.04088048
## [2,] -0.0284428 -0.02644883
```

```r
       #The mid-way matrix
#for alpha:beta:
G<-((M+Pop1m)/2)
#The loop

inter2<-((Pop1m-M)*sensitivity(G))-alpha1-beta2
inter2
```

```
##             [,1]       [,2]
## [1,] 0.00000000 0.05434000
## [2,] 0.02999683 0.08829047
```

```r
#The mid-way matrix
#for alpha:beta:
H<-((M+Pop1l)/2)
```

```r
inter3<-((Pop1l-M)*sensitivity(H))-alpha1-beta3
inter3
```

```
##              [,1]        [,2]
## [1,] 0.000000000 -0.03349910
## [2,] 0.002175623 -0.05864936
```

```r
#The mid-way matrix
#for alpha:beta:

I<-((M+Pop2h)/2)
inter4<-((Pop2h-M)*sensitivity(I))-alpha2-beta1
inter4
```

```
##             [,1]       [,2]
## [1,] 0.00000000 0.04235057
## [2,] 0.02797452 0.02575050
```

```r
#The mid-way matrix
#for alpha:beta:
J<-((M+Pop2m)/2)
inter5<-((Pop2m-M)*sensitivity(J))-alpha2-beta2
inter5
```

```
##             [,1]        [,2]
## [1,]  0.0000000 -0.05759913
## [2,] -0.0217932 -0.09144026
```

```r
#The mid-way matrix
#for alpha:beta:
K<-((M+Pop2l)/2)
inter6<-((Pop2l-M)*sensitivity(K))-alpha2-beta3
inter6
```

```
##               [,1]       [,2]
## [1,]  0.000000000 0.03395076
## [2,] -0.002051386 0.06077038
```

```r
######### ARRANGING THE DATA FOR PLOTTING ###############
#Copper
Copper <- cbind(as.vector(beta1)[2:4], as.vector(beta2)[2:4], as.vector(beta3)[2:4])
colnames(Copper) <- c("High", "Medium", "Low")
rownames(Copper) <- c("Pj", "F", "Pa")
Copper
```

```
##           High      Medium         Low
## Pj  0.06794042 -0.14382485  0.04469452
## F  -0.08325026  0.03379360  0.03280523
## Pa  0.05693927 -0.02240382 -0.03349652
```

```r
Coppert<-t(Copper)

#Layer
Layer <- cbind(as.vector(alpha1)[2:4], as.vector(alpha2)[2:4])
colnames(Layer) <- c(a, b)
rownames(Layer) <- c("Pj", "F", "Pa")
Layer
```

```
##       Pollution     Recovery
## Pj -0.001746024  0.001610043
## F   0.026486263 -0.027276861
## Pa -0.022594421  0.023071020
```

```r
Layert<-t(Layer)

#Interactions
Interactions <- cbind(as.vector(inter1)[2:4], as.vector(inter2)[2:4], as.vector(inter3)[2:4]
colnames(Interactions) <- axislabels
```

```
## Error in eval(expr, envir, enclos):  object 'axislabels' not found
```

```r
rownames(Interactions) <- c("Pj", "F", "Pa")
Interactions
```

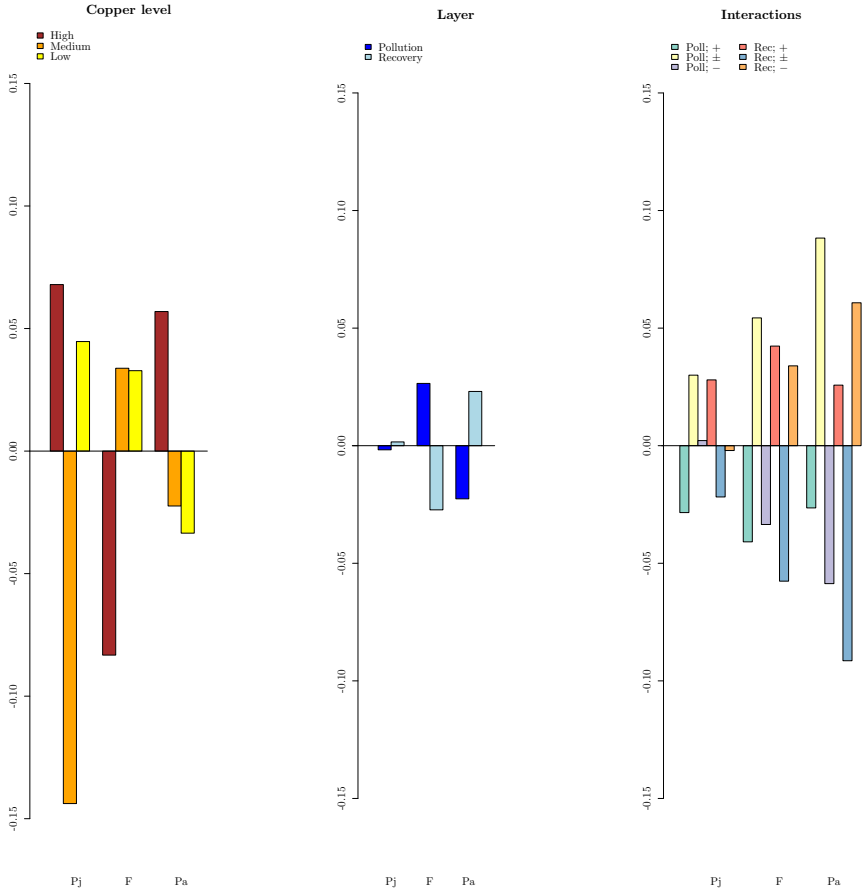
```
##           [,1]       [,2]         [,3]       [,4]
## Pj -0.02844280 0.02999683  0.002175623 0.02797452
## F  -0.04088048 0.05434000 -0.033499097 0.04235057
## Pa -0.02644883 0.08829047 -0.058649357 0.02575050
##           [,5]         [,6]
## Pj -0.02179320 -0.002051386
## F  -0.05759913  0.033950764
## Pa -0.09144026  0.060770385
```

```r
Interactionst<-t(Interactions)

# Plotting
ymin<-min(c(min(Layer),min(Copper),min(Interactions)))

ymax<-max(c(max(Layer),max(Copper),max(Interactions)))

ylim<-0.03+max(c(ymax,abs(ymin)))

limits<-c(-ylim,ylim)

#graphs
par(mfrow=c(1,3),mar=c(3,3,3,7))
```

```r
barplot(Coppert,xaxt='n', xlab="", ylab="Parameters contribution", ylim=limits, xlim=c(0,14)
segments(-1,0, 13,0)
par(mar=c(5.1, 5.1, 4.1, 5.1), xpd=TRUE)
legend("topleft",
     c("High", "Medium", "Low"), fill=c("brown","orange", "yellow"),  bty="n")
axis(side=1, at=c(3,7,11), labels=c( "Pj", "F", "Pa"), line=1, tick=FALSE)

barplot(Layert,xaxt='n', xlab="", ylab="", ylim=limits, xlim=c(0,14), col=c("blue","lightblu
segments(-1,0, 10,0)
par(mar=c(5.1, 5.1, 4.1, 5.1), xpd=TRUE)
legend("topleft",
     c(a,b), fill=c("blue","lightblue"), bty="n")
axis(side=1, at=c(2,5,8), labels=c( "Pj", "F", "Pa"), line=1, tick=FALSE)



#Interaction
#Layer
axislabels<-c()
for(i in c(ash,bsh)){
  for(j in c("H","M","L")){
    axislabels<-c(axislabels,paste(i,j,sep=""))
  }
}

axislabels2<-c()
for(i in c(ash,bsh)){
  for(j in c("$+$","$\\pm$","$-$")){
    axislabels2<-c(axislabels2,paste(i,j,sep="; "))
  }
}


rainbow(6,start = 0, end = 0.5, alpha = 0.7)
```

```
## [1] "#FF0000B3" "#FF9900B3" "#CCFF00B3" "#33FF00B3"
## [5] "#00FF66B3" "#00FFFFB3"
```

```r
library("RColorBrewer")

barplot(Interactionst, xaxt='n', xlab="", ylab="", ylim=limits, xlim=c(0,20), col=brewer.pal
segments(-1,0, 20,0)
par(mar=c(5.1, 5.1, 4.1, 7.1), xpd=TRUE)
legend("topleft",
     axislabels2, fill=brewer.pal(6, "Set3"), bty="n", ncol=2,text.width=5)
axis(side=1, at=c(5,12,18), labels=c( "Pj", "F", "Pa"), line=1, tick=FALSE)
```


Copper level
High
Medium
Low
0.15
0.10
0.05
0.00
-0.05
-0.10
-0.15
Pj
F
Pa
Layer
Pollution
Recovery
0.15
0.10
0.05
0.00
-0.05
-0.10
-0.15
Pj
F
Pa
Interactions
Poll; +
Poll; ±
Poll; −
Rec; +
Rec; ±
Rec; −
0.15
0.10
0.05
0.00
-0.05
-0.10
-0.15
Pj
F
Pa