BIO311: Population Ecology
Prac 9: Population Matrices & LTRE

Timothée Bonnet &* Koen van Benthem

timothee.bonnet@ieu.uzh.ch
koen.vanbenthem@ieu.uzh.ch

Spring 2016

Contents

*This document was co-authored by Tina Cornioley

1 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. Before analysing the rotifer data, we will first perform a simple LTRE analysis on different data. During this part of the practical, we will write some of the functions that we will use later to analyse the rotifer data. In practical 6 (last tuesday) we had a short introduction on functions. We repeat (and slightly extend) it here, to refresh your memory.

Functions so far we have worked with variables that represent a value or set of values, for example `rot` is a variable that we have defined that contains our data. A function is in a sense a variable that does not contain values, but instead contains a set of tasks. When you call a variable, `R` will show the value of that variable. When you call a function, `R` will perform the set of tasks in that function. In general when you want to define a function, it will look like this:

```
name_of_new_function <- function(input_1,input_2,...){

  do stuff

  return(output)
}
```

Let's try to make that less abstract with an example:

```
x_sq <- function(x){
  return(x^2)
}

x_sq(3)

## [1] 9
```

Here we have first defined a function (`x_sq`) that takes one input variable (`x`). This function returns the squared value of the input. We can use the function by typing `x_sq(x)`. The advantage of a function as that we don't have to rewrite the code all the time. If we now want to square a different number, we can just type:

```
x_sq(5)

## [1] 25

x_sq(7)

## [1] 49
```

Of course, this is not very usefull when the operation that we perform is just simply squaring a number, but when the operations (or set of operations) become more complicated, functions are a great way to save time.

Similarly we can also make a function that adds up two numbers and returns the squared sum:

```
x_y_sq <- function(x,y){
  ans <- x + y
  return(ans^2)
}
x_y_sq(3, 4)

## [1] 49
```

One important thing to realise is that things that happen in a function, usually stay in a function (there are ways around this, but in general it is good practice to keep it that way). This means that the tasks in the function are performed in their own 'world' and as soon as the function is executed, the changes are gone.

```
x_y_sq <- function(x,y){
  ans <- x + y
  return(ans^2)
}
x_y_sq(3, 4)

## [1] 49

ans

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

We now get an error, because `ans` only exists in the function and is deleted as soon as the function is completed.

```
ans <- 4
x_y_sq <- function(x,y){
  ans <- x + y
  return(ans^2)
}
x_y_sq(3, 4)

## [1] 49
```

What will now be the value of `ans`?

We have seen that we can not access the version of `ans` that 'lives' inside the function anymore, once the function has run. If you change anything in a function, you best return these changes to `R` at the end of the function using `return()`. Please note that when a function executes `return(x)`, `x` is given back to `R` and anything that comes after it will not be executed.

We will now write a function that calculates the sensitivities of λ to the matrix elements. Writing a general function, will make it easier for us to calculate sensitivities for matrices later (which we will do frequently). We provide two options for doing so: by numerical perturbation as we did in the previous practical, or by using the exact solution. The latter is conceptually more difficult, but more precise and less code. You may choose yourself which one you want to implement. Do try to write your function such that it can handle matrices with different sizes.

A: Sensitivities using perturbations

Yesterday we have calculated the sensitivities for a barn owl population. We can now re-use that code and put it in a function, so we can use it to from now on easily calculate sensitivities for new matrices. Below the outline is given.

```r
# We are going to make a function with the name
# sens, that takes 1 input variable: a matrix. We call this
# matrix A. Everything between { and } is the code
# of the function. At the end the function uses return()
# to give back the calculated values of the sensitivities
sens <- function(A){

  da <- 1e-4 # define a small perturbation
  lam <- ...  # extract the dominant eigenvalue of A
  dim <- ... # store the size of A. Use ncol()
             # use ?ncol to see what it does

  # finally, create an empty matrix to store your results in
  output <- matrix(0,ncol=dim,nrow=dim)

  # loop over all indices [i,j] in A, rows and columns
  # to calculate the sensitivity of each and every index
  for(i in ...){
    for(j in ...){
      TempA <- A # we copy the matrix
      TempA[i,j] <- ... # we perturb one element
      TemLam <- ... # get lambda of the perturbed matrix
      output[i,j] <- ... # calculate the sensitivity of
                         #lambda to a[i,j] (dlambda/da)
    }
  }

  # In higher dimensions, some eigenvalues may have complex parts
  # These are not relevant for the sensitivity analysis
  # (because the sensitivities will not have complex parts)
  # we still add the following line to R to make sure that R does
  # not annoy us constantly by adding a +0i (+ an imaginary part of 0)
  # to the outcomes
  output <- Re(output)

  return(output) # The end of the function is reached,
                 # we now return the value of output
}
```

B: Sensitivities using the exact solution

Here we will use the formula that was shown in the lecture:

$$\begin{pmatrix} \frac{\partial\lambda}{\partial a_{1,1}} & \frac{\partial\lambda}{\partial a_{1,2}} & \cdots & \frac{\partial\lambda}{\partial a_{1,N}} \\ \frac{\partial\lambda}{\partial a_{2,1}} & \frac{\partial\lambda}{\partial a_{2,2}} & \cdots & \frac{\partial\lambda}{\partial a_{2,N}} \\ \vdots & \vdots & & \vdots \\ \frac{\partial\lambda}{\partial a_{N,1}} & \frac{\partial\lambda}{\partial a_{N,2}} & \cdots & \frac{\partial\lambda}{\partial a_{N,N}} \end{pmatrix} = \frac{\vec{v}\vec{w}^T}{\vec{v}^T\vec{w}} \tag{1}$$

Here, $\vec{w}$ and $\vec{v}$ are the dominant right and left eigenvector respectively. Furthermore $\vec{v}^T$ responds to the transpose of $\vec{v}$ (try `?t` in `R`). The product of two vectors can be performed in `R` by using the `%*%` operator. Using this we can now wite a function to calculate the sensitivities:

```
sens <- function(A){
  w <- ...
  v <- ...
  outcome <- ... / as.numeric(...)
  # here we have to add the as.numeric argument.
  # this is just a problem of R:
  # it sees that the number comes from a vector
  # multiplication and therefore stores it as a matrix
  # with dimension 1x1. However, it then has difficulties
  # to divide a matrix by another matrix. We therefore
  # explicitly have to tell R that this 1x1 matrix is just
  # a number.

  # Next: we delete the imaginary part:
  outcome <- Re(outcome)
  return(outcome)
}
```

Now test your function on the following two matrices, to make sure that it works.

```
A <- matrix(c(0.5,0.3,0.75,0.9),nrow=2)
sens(A)

##           [,1]      [,2]
## [1,] 0.3057428 0.2913858
## [2,] 0.7284644 0.6942572

B <- matrix(c(0.5,0.3,0.7,0.9,0.3,0.2,0.4,0.8,0),nrow=3)
sens(B)

##           [,1]      [,2]      [,3]
## [1,] 0.4497062 0.3238265 0.2729042
```

```
## [2,] 0.4412927 0.3177681 0.2677985
## [3,] 0.3831683 0.2759136 0.2325257
```

This is a very good start! We now have a function that easily provides us with the sensitivity values for any matrix. From here, we can finally think about more biological problems!

1.1 Comparison of two populations of mice

We will now continue with the more biological part of the practical and 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:

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

and

$$\boldsymbol{M} = \begin{pmatrix} 0 & 1.9 \\ 0.2 & 0.45 \end{pmatrix} \tag{3}$$

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 between the two populations. As before, we will start by looking at the sensitivities: which parameter seems to have the biggest effect on λ?

The sensitivites give us an idea of what might happen in the future. It shows how much λ would increase if a matrix elemant increases with a certain amount. For this, we look at the equations. As we have shown before:

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

from this it also follows that:

$$s_{i,j} \partial a_{i,j} = \partial \lambda \tag{5}$$

So what we see is that if a matrix element increases with an amount of $\partial a_{i,j}$, λ will increase with an amount of $\partial\lambda$. For our population however, we know how big the difference in $a_{i,j}$ are from looking at the differences between the matrices (for example, there is a difference of 0.25-0.2=SexprM[2,1]-P[2,1] in the maturation rate between the two populations). If we want to know how much each of these differences contributes to differences in λ, all we have to do with these differences is multiply them with the sensitivities of the corresponding elements.

There is one problem however, which sensitivities do we take? The ones from $\boldsymbol{M}$ or the ones from $\boldsymbol{P}$? In practice what we do is that we use the sensitivities of the average matrix, to make sure that our sensitivities are not biased towards one of the two matrices.

Using these values, we can 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:

$$\lambda^{(M)} \approx \lambda^{(P)} + \sum_{i,j} (a_{ij}^{(M)} - a_{ij}^{(P)}) \frac{\partial \lambda^A}{\partial a_{ij}^A} \tag{6}$$

Below we explain the different terms:

- $\lambda^{(P)}$ 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

  $$\boldsymbol{A} = \frac{\boldsymbol{M} + \boldsymbol{P}}{2} \tag{7}$$

  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 6 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 6, 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 rate?

Hint Find the "mid-way" matrix. For the sensitivities.

2 Rotifer data analysis

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

2.1 Rate estimation

Before we can start with the analysis of the population matrices, we first need... the matrices and thus the rates (survival and reproduction). Here, we have only two stages: juveniles and adults, so we consider a 2×2 matrix. First, we load in the data:

```
Rot <- read.csv("rotifer_data.csv")
```

Next, we use the function `reshape()` to change the format of the data. This will later make it easier to calculate the rates. Make sure that you understand how this function changes the data!

```
Roti<-reshape(Rot,timevar="Day",idvar=c("Clone","Copper","Layer"),direction="wide")
```

We will now continue and calculte the rates, for this, we first create a new data frame to store the rates in. This data frame is basically

```
Rates <- Roti[,1:3]
```

Before we start, draw a 2 stage life cycle for the rotifers, this will make it easier to keep track of what is happening. We shall now first calculate the adult survival rate for the period from day 1 to day 2. We start by looking at adult survival. We assume that we counted all dead individuals. Now, the survival rate can easily be calculated, it should be the number of alive adults on the first day, minus the number of dead adults observed on the second day, divided by the number of alive adults in the first time step.

```
Rates$Sa1 <- (Roti$Live_Adult.1 - Roti$Dead_Adult.2)/ Roti$Live_Adult.1
```

Next, we look at the maturation rate. We have already calcuated how many adults survived from day 1, to day 2. The total number of adults on day 2 will however be higher than this number. The additional adults should have arrived due to maturation of juveniles. Calculate the maturation rates (the following expression is incomplete):

```
Rates$M1 <- (Roti$Live_Adult.2 - ....) / ...
```

When we look into the calculated maturation rates, we will see that sometimes it is sometimes higher than 1. The reason is that we have uncertainty in our count data, maybe an adult too many was counted on day 2, or maybe a juvenile too few on day 1. An alternative explanation is that we assume only 1 transition to take place, it is however theoretically possible that within 1 day a juvenile is born and already grows up to the adult stage. Since we have only very few entries that display such problems, we will cut off the maturation rate at 1: we know that it should not be higher than 1. Uncertainty in the data should not change that theoretical assumption.

```
Rates$M1[Rates$M1 > 1] <- 1
```

The next rate is the probability for a juvenile to survive but stay a juvenile. Alive juveniles at day 1 that did not mature and that were not observed dead on day 2, should have survived and stayed in the same stage:

```
Rates$Sj1 <- (Roti$Live_Juv.1 - ... - ...)/...
```

This time we see a few cases were survival is found to be negative. This time we account for it by setting these rate (again there are not many cases) to zero.

```
Rates$Sj1[Rates$Sj1 < 0] <- 0
```

Finally we arrive at the last rate: fecundity. Any juveniles observed on day 2, that are not surviving juveniles from day 1, must be new offspring. Calculate the fecundity rates from this concept:

```
Rates$R1 <- ...
```

Fortunately, for these rates, no weird values are found. Now continue and calculate the rates for the transition from day 2 to day 3. Check for every rate if some illogical values are calculated and correct them before continuing with the calculation of the next rate.

```
Rates$Sa2 <- ...
Rates$M2 <- ...
Rates$Sj2 <- ...
Rates$R2 <- ...
```

We can now plot what these rates look like. For this it is easiest to first reshape the rates. The following two commands allow you to generate a boxplot of the rates for the transition of day 1 to day 2:

```
library(tidyr)
Rates_plot <- gather_(Rates,"Rate","Value",c("Sj1","Sa1","M1","R1"))
boxplot(Value~Copper+Rate,Rates_plot)
boxplot(Value~Layer+Rate,Rates_plot)
```

Here, we use `gather_`, try to understand what it does. Then we use `boxplot`. Here, the first argument specifies what is plotted (in this case the column `Value` is plotted, but it is split out by the categories `Copper` and `Rate` for the first plot. This is parsed to the function by the code `Value` $\sim$ `Copper + Rate`). Adapt this code to also plot the rates for the second transition.

Alternatively you may consider using `ggplot()` to obtain – depending on your taste – nicer looking plots.

```
library(ggplot2)
ggplot(Rates_plot,aes(factor(Copper),Value,fill=Rate)) + geom_boxplot()
```

We will now continue and start creating the matrices. For this, first we calculate the average rates for the transition from day 1 to 2 and the one from day 2 to 3. This will allow us to make on matrix and thus one value of λ per clone

```
# Averages
Rates$SaM <- (Rates$Sa1 + Rates$Sa2)/2
Rates$MM <- ...
Rates$SjM <- ...
Rates$RM <- ...
```

Now we create a new column with the name `lambda` and using a for-loop, we can now for each replicate calculate the value for lambda, by constructing the appropriate matrix.

```
Rates$lambda <- 0 # create empty column
for(i in 1:nrow(Rates)){ # for each row in Rates
  Mat <- ...
  Rates$lambda[i] <- ...
}
```

We can now again use the boxplot commands that we used before to see how lambda varies among matrices.

2.2 Factorial LTRE

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 and a "copper" treatment. We can no longer just compare two treatments with each other, instead we have a whole suite of treatments (9 in total, 3 layers times 3 copper levels). 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 copper" which we call $\boldsymbol{M}^{r,l}$ as the focal matrix, for this we average the rates over all the 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 9 matrices (one for each treatment) that we are comparing to each other. Again we are interested in understanding the effect of the different treatments on the asymptotic growth rates.

$$\lambda^{r,l} = \lambda^{\cdot\cdot} + \alpha^{r} + \beta^{l} + (\alpha\beta)^{r,l} \tag{8}$$

Equation 8 tells us that $\lambda^{r,l}$ can be found from $\lambda^{\cdot\cdot}$ plus an effect from the layer "recovery", an effect from the copper level "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 copper is ignored. The rates of this matrix are calculated as the average of all the matrices with from the layer "recovery".

$$\alpha^{r} = \sum_{ij} (a_{ij}^{r\cdot} - a_{ij}^{\cdot\cdot}) \frac{\partial \lambda^{A}}{\partial a_{ij}^{A}} \tag{9}$$

The structure of equation 9 is the same as the last part of equation 6. 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 the effects for the other layers, e.g. α^{peak}.

We do the same for the second factor. We examine a matrix $\boldsymbol{M}^{\cdot l}$, where the effect of the layer is ignored:

$$\beta^{l} = \sum_{ij} (a_{ij}^{\cdot l} - a_{ij}^{\cdot\cdot}) \frac{\partial \lambda^{B}}{\partial a_{ij}^{B}} \tag{10}$$

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.

$$(\alpha\beta)^{rl} = \sum_{ij}(a_{ij}^{rl} - a_{ij}^{\cdot\cdot})\frac{\partial\lambda^C}{\partial a_{ij}^C} - \alpha^r - \beta^l \tag{11}$$

So far we have considered only the effect of the layer "recovery" and the copper level "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.

$$\lambda^{k,m} = \lambda^{\cdot\cdot} + \alpha^k + \beta^m + (\alpha\beta)^{k,m} \tag{12}$$

$$\alpha^k = \sum_{ij}(a_{ij}^{k\cdot} - a_{ij}^{\cdot\cdot})\frac{\partial\lambda^A}{\partial a_{ij}^A} \tag{13}$$

$$\beta^m = \sum_{ij}(a_{ij}^{\cdot m} - a_{ij}^{\cdot\cdot})\frac{\partial\lambda^B}{\partial a_{ij}^B} \tag{14}$$

$$(\alpha\beta)^{km} = \sum_{ij}(a_{ij}^{km} - a_{ij}^{\cdot\cdot})\frac{\partial\lambda^C}{\partial a_{ij}^C} - \alpha^k - \beta^m \tag{15}$$

In these equations:

$$A = \frac{\boldsymbol{M^{k\cdot}} + \boldsymbol{M^{\cdot\cdot}}}{2} \qquad B = \frac{\boldsymbol{M^{\cdot m}} + \boldsymbol{M^{\cdot\cdot}}}{2} \qquad C = \frac{\boldsymbol{M^{km}} + \boldsymbol{M^{\cdot\cdot}}}{2} \tag{16}$$

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.

We are now going to apply this theory on the rotifer data. First write a function that performs an `LTRE`. That is, if we give it two matrices as input, it calculates the sensitivities of the average matrix (you can call the function you made earlier) and it multiplies this with the difference between the matrices. You don't have to use a loop here, if you use `A*B`, this is not matrix multiplication, but element wise multiplication between the matrices. It simply multiplies the first matrix entry from A, with the first one from B. Then it multiplies the second from A with the second one from B.

Next, we will have to create the mean matrix, to which we will compare all other matrices. Simply take the means of all the rates (`mean`) and put these rates together in a matrix.

```
MatMean <- ...
```

Now we proceed and calculate the effect of copper on lambda. For this we have to calculate an average matrix for each copper treatment and compare these with the overall average matrix. To obtain the average matrix, we again use `aggregate`, but this time we supply a formule: `.~Copper`, this means that `aggregate` will apply the function (in this case `mean`) on all columns, per coppr level. To make sure that we don't add up with lots of columns, we therefore first take a subset of `Rates` that only contains the relevant columns.

```
### Copper treatment
temp  <- Rates[,c('Copper','SaM','MM','SjM','RM')]
tempc <- aggregate(.~Copper,temp,mean)
```

Next, we have to build a matrix for each row in `tempc` and perform an LTRE on each of these matrices compared to the overall mean matrix. To make sure that we can store the result afterwards, we first create an empty list with the right length. Complete the code to obtain the effects of copper through the different matrix elements on λ.

```
copper_effect <- list(3)
for(i in 1:3){
  mat <- ...
  copper[[i]] <- ...
}
```

Finally, we add the names of the copper levels to this list, so we can easily find the right matrix back:

```
names(copper_effect) <- tempc$Copper

# now we can obtain specific copper levels
# by naming them
copper_effect[['100']]
```

What are the total effects of the different copper levels on λ? Repeat the same procedure for the layer effect.

Finally, we have to calculate the interactions. Now we can use that we names the lists with the copper and layer effects to efficiently access them. We will have to access these in order to be able to substract the correct fixed effects from each interaction, as described in the theory at the beginning of this section.

```
temp2  <- Rates[,c('Copper','Layer','SaM','MM','SjM','RM')]
tempi <- aggregate(.~Copper+Layer,temp2,mean)
interaction <- list(9)
```

```
for(i in 1:9){
  mat <- ...
  interaction[[i]] <- ... - ...
}
```

What is left now is to plot the values and to examine the summed effect of each copper level, each layer level and each interaction. One way to plot for example the copper effects is by using `barplot(matrix(unlist(copper),nrow=4))`.

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!

Figure 2: random picture, from wikipedia.