Interactions between life history and the environment

Jason Matthiopoulos 2011

Overview: The complex interactions between life history traits and the environment may favour or disadvantage single individuals or entire populations. These effects are difficult to intuit and their investigation may require the construction of stochastic, individual-based models. Here, we construct such a simulation, loosely based around the life history of female killer whales. We examine the implications on inclusive fitness of four different scenarios of resource richness and environmental variability.

1. the Study System

The Killer Whale or Orca (*Orcinus orca*), is the largest species of the dolphin family. It is found in all the world's oceans, from the frigid Arctic and Antarctic regions to warm, tropical seas. Some killer whale populations feed mostly on fish while others hunt sharks, marine mammals, including sea lions, seals, walruses and even large whales and Great white sharks.

There are up to five distinct killer whale types distinguished by geographical range, preferred prey items and physical appearance. Some of these may be separate races, subspecies or even species. Killer whales are highly social; some populations are composed of matrilineal family groups, which are the most stable of any animal species. Their sophisticated social behaviour, hunting techniques, and vocal behavior have been described as manifestations of culture.

Females become reproductively active at the age of 15 and breed until they are 40. They have periods of polyestrous cycling with non-cycling periods of between three and sixteen months. The gestation period varies from fifteen to 18 months. Calves nurse for about 2 years after birth, but remain closely associated with their mother for about four years. Mothers maintain a level of parental investment for that period, leading to an inter-birth interval of five years. During those years, calves suffer from an annual mortality of 20%. After their fifth year, female calves survive to recruitment with a probability of 0.98.

Females' life spans average 50 but may survive well into their 70-80s in exceptional cases. Males become sexually mature at the age of 15 but do not typically reproduce until age 21. Male killer whales generally do not live as long as females. In the wild, males average 30 years, with a maximum of 50–60 years in exceptional cases.

Killer whale (Orca orcinus)

2. ecological theory

The interplay between life history strategies and environmental variation has the potential for rich dynamics. Life history traits evolve as a result of persistent environmental drivers but, in the short-term, the fitness of individuals and the fate of populations depends on environmental trends and the stochasticity around them. As scientists, we will rarely be able to predict a-priori which aspects of a life history strategy will be beneficial for an animal living in a particular environmental regime. Simulation is an invaluable tool for achieving this.

3. Mathematical theory

■ *Formalising dependence on covariates:* Quantifying the effects of particular causes is the stock-in-trade of most empirical scientists. Coming up with mathematical expressions for these relationships can be achieved either by **mechanistic** or **empirical** means. A mathematical model is called mechanistic when it has been derived from first principles. So, for example, a model for the global human population that takes into account births (by country) and deaths (by cause) is mechanistic at the level of demography, because it acknowledges the proximate causes of population change. In contrast, when we fit an exponential curve through the population data, we are using an empirical model, i.e. one that has the "right sort of form".

At their very simplest, empirical models can be defined on the basis of a function's domain, range and **monotonicity**. In other words, what kind of values a function takes as its input, what sort of values it produces as its output and whether it is purely increasing or decreasing.

For example, if we know that the amount of weight (w) gained/lost by an individual organism is a simple, increasing function of the availability of two types of food (x_1 and x_2), we might express this with the empirical relationship

$$w = a_0 + a_1 x_1 + a_2 x_2 \quad (1)$$

Here, we have used a linear relationship in which w can take both negative and positive values, the intercept a_0 could be thought of as a negative value that has some association with basal metabolic costs and the coefficients a_1 and a_2 give the relationship between weight gain and food availability for each of the two types of food.

In certain cases, the response variable can only take non-negative values. For example, we might be interested in the average number of offspring (b) that can be produced by an animal as a function of food availability. Then, we might consider a **log-linear model**, in which the linear predictor $a_0 + a_1 x_1 + a_2 x_2$ is exponentiated,

$$b = \exp(a_0 + a_1 x_1 + a_2 x_2) \quad (2)$$

In other cases yet, we might be interested in modelling a probability or a proportion. For example, what is the annual probability of survival (s) of an animal as a function of food availability? This may be formalised by a so-called **logit model** which looks like this

$$s = \frac{\exp(a_0 + a_1 x_1 + a_2 x_2)}{1 + \exp(a_0 + a_1 x_1 + a_2 x_2)} \quad (3)$$

If you try very large and very small values for the variables x_1 and x_2, you will notice that this function is constrained to take values between 0 and 1, which is what we require of a probability.

4. R ingredients

■ *Generating random numbers*: This is really easy. You first need to decide which distribution you want these random numbers to come from (uniform, normal, binomial, Poisson etc). Then, you need to specify how many numbers you want out of that distribution and finally, you need to specify the parameters of the distribution. Here are some useful examples:

R command	*Explanation*
`runif(2, 1,10)`	Generates two random numbers between 1 and 2 from the continuous uniform distribution.
`rnorm(100, 0, 10)`	Generates 100 draws from a normal distribution with mean zero and standard deviation 10.
`rpois(10, 5.3)`	Generates ten random numbers from a Poisson distribution with rate 5.3
`rbinom(10,1,0.5)`	Draws ten random numbers from a binomial distribution with 1 trial and a probability of success 0.5.
`rbinom(1,10,0.5)`	Draws one random number from a binomial distribution with 10 trials and a probability of success 0.5.

Can you tell the difference between the last two examples?

■ *Testing facts*: Some times it is useful to test whether a fact is true or false. You may put the question to R as a mathematical statement. For example, you may ask if 2 is bigger than 3,

```
> 2>3
[1] FALSE
```

Equality is tested with the symbol ==,

```
> 2==2
[1] TRUE
```

We can test the validity of combined statements by bolting together different questions with the AND, OR operators. In R, these are written && and || respectively. Let's have a look at some examples:

Plain-language question	*R-version*	*R-response*
Is either of the following true? 1+1=2, 2+2=3	`1+1==2 || 2+2==3`	**TRUE**
Are both of the following true? 1+1=2, 2+2=3	`1+1==2 && 2+2=3`	**FALSE**

■ *Conditional statements:* Conditional statements are useful when you want to execute a set of commands, but only when a certain set of conditions hold. So, the sort of thing you might like to say in plain language is "*If a set of conditions is satisfied then do something* ". The programming structure `if` can be used for this purpose. Here is what it looks like in general

```
If (conditions)
      {
      Commands
      }
```

The following example will first examine if the random number `r` is positive and then print a statement to say so.

```
r<-rnorm(1,0,1)
if (r>0)
  {
  print("positive")
  }
```

There is an extension to this conditional structure leading to **compound conditional statements** which loosely translates to "*If a set of conditions is satisfied then do something, if they are not, then do something else*". The following example will report whether the random number `r` is positive or negative.

```
r<-rnorm(1,0,1)
if (r>0)
  {
  print("positive")
  } else {
  print("negative")
  }
```

■ *Iteration and loops:* **Iteration** is the repeated application of a set of commands. In most occasions we can perform tasks iteratively by using the vectorisation capabilities of R. Sometimes, iterative tasks require a different programming device, called a **loop**, which comprises of 1) a loop declaration and 2) the main body of the loop. Here, we will introduce two types of loops, the `for` loop, performs a set of tasks for a predefined number of iterations. The `while` loop performs a set of tasks while a set of conditions hold.

In R, a typical `for` loop will look a bit like this:

```
for (counter in min:max) #  This line declares the loop counter and range.
      {
      #  The main body of loop - the commands to be repeated.
      }
```

For example, the following will produce the first 21 terms of the sequence $a_{t+1} = 100 + 35a_t$

```
1 a<-c()
2 for (n in 0:20)
3     {
4     a<-c(a, 100+n*35)
5     }
```

Here, line 1 creates an empty list `a`. Line 2 declares the loop i.e. it defines a counter `n` that will successively take the values 0 (min) to 20 (max). The brackets in lines 3 and 5 enclose the main body of the loop, the parts that are to be iterated. Several lines of code can be enclosed in the main body but, in this example, we only have one, line 4, which uses the concatenation command `c()`, to add to the current list `a`, the new value `100+n*35`. Since this value contains the counter `n`, it will be calculated anew with each iteration of the loop. To see how the loop operates, try inserting the command `print(a)` as a new line in-between lines 4 and 5. Also, note that the brackets and main body of the loop is typeset towards the right. This makes it easier to identify on the screen or printed page where the loop begins and where it ends.

In R, a typical `while` loop will look a bit like this:

```
while (conditions) #  This line declares the loop conditions.
      {
      #  The main body of loop - the commands to be repeated.
      }
```

We can achieve the same result with a `while` loop as with a `for` loop. Here is an example

```
1   n<-0
2   a<-c()
3   while (n<=20)
4      {
5      a<-c(a, 100+n*35)
6      n<-n+1
7      }
```

Can you see how this works? A counter is initialised outside of the loop and incremented by one inside the loop. When the counter reaches a pre-determined size, then the loop ends. In this example, the for implementation was more economical than the while version. However, the while loop, really comes into its own when the number of iterations is unknown. In the following example, we toss a fair coin as many times as are required to get 20 heads. At the end, the program reports the number of trials that were conducted. You will notice that this number may change if you re-run the program.

```
1   heads<-0
2   trials<-0
3   while (heads<20)
4      {
5      heads<-heads+rbinom(1,1,0.5)
6      trials<-trials+1
7      }
8   trials
```

■ *Setting out initial conditions and parameters:* It is generally good practice to represent parameters by symbols whose numerical assignments are collected together at the beginning of the code, separately from its main body. This is for three reasons: 1) it gives an overview of the kind of numerical information required for the model, 2) the statement of the model in the main body of the code is more reminiscent of its mathematical description which makes it easier to find mistakes and 3) in many models the same parameter is repeated several times. Rather than having to trawl through the entire code and change the parameter's numerical value in all these instances, it is only necessary to change its numerical assignment at the beginning.

5. Practical tasks

We will gradually build up to a simulation of the reproductive life histories of thousands of female killer whales, to investigate how the availability and variability of resources affects individual fitness, total reproductive output and the age of females at death.

■1■ Read carefully the material in sections 1-4 above.
■2■ Read through the tasks in this section.
■3■ Read through the assessment components in the next section before you carry out any tasks

Modelling a single female without a calf:
■4■ We will use the variable `co`, measured in some arbitrary unit, to track the condition of each individual female. At recruitment (i.e. when they become reproductively active), females have a normal distribution of conditions with mean 100 and standard deviation 10. Write a line of code that selects a random value for `co`, from such a distribution.
■5■ Define the variable `age` to track the age of a single female. Set `age` to the appropriate value for a female that has just become recruited.
■6■ The variable `alive` will take the value 1 if a female is still alive in any given year, and zero if it has died. Set it to 1, to begin with.
■7■ Write a loop that runs while a female is alive and of a reproductive age. Within each iteration of the loop, do the following two things: (i) allow the whale to survive with a probability `sr`=0.8 and (ii) if it does, increment its age by 1. Run the loop and check the age-at-end-of-reproduction.
■8■ The quality and variability of the environment will determine the ability of an individual to improve its condition during any given year. We assume that these **annual condition increments** are normally distributed and independent between successive years. We will represent the influence of the environment by the mean (`enMu`) and standard deviation (`enSD`) of that distribution. Hence, a high value of `enMu` indicates a rich environment and a high value of `enSD` indicates a variable environment. To begin with, set these values to 0 and 20 respectively.
■9■ Write a variation of the loop in task 7 in which the condition of an individual female is changed by a random condition increment each year. Make the probability of survival `sr` depend on the following linear predictor: `s0+s1*co` where `s0`=-2 and `s1`=0.05. Run the loop to check the age at death.

Modelling a single female with a calf:
■10■ Define the variable `calf` that can take the value 1, if a female currently has a calf and 0 if she doesn't. Also, define the variables `calfage` to track the age of a calf and `offspring` to track the number of calves that become independent of their mothers. Initialise all of these at zero, outside your main loop.
■11■ Within your loop, write a compound conditional statement that does the following:

- If the female doesn't have a calf, then
    - calculate its breeding probability `b` as a function that depends on the linear predictor `b0+b1*co` (where `b0`=-10 and `b1`=0.1).
    - use this probability as part of a Bernoulli trial, to simulate the event of birth
    - if a calf is born, then set `calf` to 1, otherwise leave it at zero

- If the female has a calf, then
    - decrement the female's condition by the amount `inv`=10, corresponding to annual maternal investment.
    - decide randomly whether the calf is going to survive the year according to a fixed probability
    - set `calf` to zero if calf has died
    - set `calfage` to zero if calf has died, or increment it by one otherwise.

■12■ Within your loop, after the statement produced from task 11, write a simple conditional statement that does the following:

- If the calf has reached the age of independence
    - Increment the number of the mothers offspring
    - reset `calf` and `calfage` to zero.

Run the entire code to make sure it works. Check out critical outputs such as age or offspring. For example, if your program generates more offspring than the final age of the female, then something is wrong. Go back over the code to find where the problem is.

Modelling several females:
■13■ We will need to store the total number of offspring that reach recruitment from each female whale and the age at which females stop breeding. Create the empty lists `recruits` and `ages` for this purpose.
■14■ Use a loop to simulate 1000 female life histories, each time storing the female's final age and its total number of recruited offspring into the lists `ages` and `recruits` respectively. Note that the total number of recruited offspring is not the same as the number of offspring that become weaned.

Scenario 1: Rich and stable environment
■16■ We will interpret the number of recruits produced by each female as its **inclusive fitness**. Set the parameters of the environment to `enMu`=20 and `enSD`=1.
■17■ Generate a histogram of ages at end of reproduction
■18■ Generate a histogram of inclusive fitness
■19■ Calculate the average number of recruits produced by each female

Scenario 2: Poor and stable environment
■20■ Set `enMu`=2 and `enSD`=1 and repeat tasks 17-19.

Scenario 3: Rich and variable environment
■21■ Set `enMu`=20 and `enSD`=30 and repeat tasks 17-19.

Scenario 4: Poor and variable environment
■22■ Set `enMu`=2 and `enSD`=30 and repeat tasks 17-19.

■23■ Comment on the effect of the environment on the fitness of particular individuals and the reproductive ability of an entire population.

6. Assessment

Write a report, no longer than 3 sides of A4, that contains the following:
■1■ Histograms and value from scenario 1.
■2■ Histograms and value from scenario 2.
■3■ Histograms and value from scenario 3.
■4■ Histograms and value from scenario 4.
■5■ A paragraph with your response to task 23.
■6■ Fully commented Tinn-R code only for the complete simulation that you used to answer parts 1-4 above.