LAB 3 Models Over Fishing

ER Deyle

Fall 2024; Marine Semester Block 3

Contents

1 Introduction

The goal of this lab is to numerically explore the 1-dimensional calculus and optimization that is a major foundation for the mathematics of fisheries management. We will also add a few more basic ingredients to those from the previous exercises. By the end of this lab you should be able to:

- Write custom functions within an R script.
- Use `while` to construct an iterated calculation that need to be run until a condition is met.
- Use custom functions in a modular way for doing iterated numerical experiments.

1.1 Fisheries Context

In addition to building the tools described above, you will be reproducing a piece of land-mark analysis in the development of scientific thinking around Quantitative Fisheries Analysis from the 1977 paper by Beddington & May:

Beddington, J. R., & May, R. M. (1977). Harvesting natural populations in a randomly fluctuating environment. Science, 197(4302), 463-465.

The paper should be included in the course OneDrive with the file name "Beddington_et_May_Science_1977.pdf". After you have progressed most of the way through the lab, you will need to locate the Beddington & May 1977 paper and read that as well. It is short but dense.

The essential insight of their paper is as follows. Despite the MSY value for fishing being optimal in the mathematical sense under the assumptions reflected by the model, the authors note that one of these assumptions is that the population exists in a static, unchanging environment. If that assumption is relaxed– that is, the model is adapted to account for even simple random variation in the environment– application of MSY management will lead to over-exploitation. The stock can still recover under these modified assumptions, but the recovery can take a long time. This is especially true depending on how MSY is implemented.

1.2 Computational Approach

1.2.1 Population Dynamics

The paper studies a "deliberately oversimplified" population with a logistic net growth rate.

$$\frac{dN(t)}{dt} = rN(t)(1 - N(t)/k)$$

On top of that, they add a model of how fishing works. The first way is to think of a given level of fishing effort yielding a catch that is proportional to the size of the population. This is `constant effort` fishing.

$$\frac{dN(t)}{dt} = rN(t)(1 - N(t)/k) - EN(t)$$

The second way they model fishing is treating the fishing level determining the total catch, regardless of the population abundance. This is `constant catch` fishing. Presumably, the amount of effort that would be required might change, but the human behavior is being modeled as effort compensating for any change in population.

This model is a differential equation, as opposed to a difference equation, meaning that population growth is being treated as a continuous process in time, not a step-by-step process one generation to the next. In the calculus of differential equations, these problems are solved by integrating both sides through time, given some initial value for $N(t = 0) = N_0$.

$$\int_{t'=0}^{t} \frac{dN(t')}{dt'} dt' = \int_{t'=0}^{t} \left[rN(t')(1 - N(t')/k) - EN(t') \right] dt'$$

The left-hand side is easy, by the fundamental theorem of calculus.

$$\int_{t'=0}^{t} \frac{dN(t)}{dt} dt' = N(t) - N_0$$

The right-hand side is not, generally. However, recall the motivation of the integral as the area-under-the-curve; it came from approximating the function as a series of narrow little steps, and the area below as small

rectangles. Adding up lots of small things by hand, no fun. What about adding up lots of small things by computer...?

This constitues the "Forward Euler method" for solving an Ordinary Differential Equation. We integrate over tiny little snippets of time of length Δt, assuming that the function is basically constant. That is, $N(t + \Delta t) \approx N(t)$ if $\Delta t << 0$. Then, the integral of a constant over the interval is just the value of the constant times the length of th einterval (the area of a small little rectangular slice).

$$\int_{t'=0}^{\Delta t} \frac{dN(t)}{dt'} dt' = N(\Delta t) - N_0 = \int_{t'=0}^{t} \left[rN_0(1 - N_0/k) - EN_0 \right] dt'$$

And

$$N(\Delta t) = N_0 + \left[rN_0(1 - N_0/k) - EN_0 \right] \Delta t$$

More generally, to go from t_k to $t_{k+1} = t_k + Deltat$

$$N(t_{k+1}) = N(t_k) + \left[rN(t_k)(1 - N(t_k)/k) - EN(t_k) \right] \Delta t$$

This is just arithmetic, now, iterated many many times. Perfect for a computer. And since it's just a model of a single population abundance, N, you don't even have to write it as a matrix equation!

1.2.2 Return Time

The paper also invokes a mathematical notion, the "characteristic return time". The precise definition is something we can more easily revisit later in the course once we've talked about mathematical stability and matrices. However, the general behavior the mathematical quantity describes is something we can simulate with computer code, without needing to wade into derivations and what-not. Essentially it is the formal mathematical answer to the question "How long does it take a dynamic (changing-over-time) system to recover from a perturbation?"

Intuitive explanations of system stability often fall back on the mental picture of a ball down in a valley.

```
## Loading required package: ggplot2
```

The steeper valley on the left has a stronger *restoring force*, and corresponds to a more stable equilibrium. If we displace each ball the same amount horizontally, the ball on the left will return to the equilibrium faster than the one on the right. The actual return time will depend on how far the ball was displaced. For small perturbations, the return to equilibrium follows an exponential form; it's basically akin to radioactive decay. The "characteristic return time" is basically just the half-life of that decay (properly 1/e instead of 1/2...).

In population dynamics, we're not thinking about a ball, but a population. The displacements are perturbations; maybe from climatic fluctuations, a random weather event, or a human action... like fishing. **We will capture variation in the return time** (i.e. variation in the stability of the population) **by simulating the dynamics of the population returning from a perturbation, and calculating how long it takes to be halfway back to equilibrium**. (Math side note: there's nothing special about picking halfway back specifically, we would get the same behavior if we tracked return time to 90% or 99% recovered). If exploitation reduces the resilience of the system, this number of time steps will get longer and longer as the system recovers more and more slowly from disturbance.

So, putting it all together, we will use the model from the paper and the approximate integral calculus described above to simulate the population in the model returning to equilibrium. Then, we will calculate how long (how many time steps) it took to return halfway to equilibrium as a function of fishing level.

Figure 1: Physical analogy of stability

The experiment will be easier to set up if you can determine the equilibrium abundance under a particular level of fishing effor, $N^*(E)$. The experiment will have some parameters, including the size of the perturbation, ΔN. Each single iteration of the experiment, then, will consist of simulating the population N using the integration scheme, with an initial abundance of either $N^*(E)+\Delta N$ or $N^*(E)-\Delta N$. This will generate a time-series of population abundances. You don't need to record each time-series for each single iteration, however. The important data to record will be the number of time-steps it took for $|N(t)-N^*(E)| < \Delta N/2$.

1.3 Task list:

For economy, I'm going to refer to the population dynamic equation in Beddington & May 1977, Equation 1 as B&M Eq(1).

- Write a basic script to simulate B&M Eq(1) for specific values of parameters of the model and simulation method.
- Write a more flexible script to simulate the population dynamics using a custom function.
- Simulate return times under constant effort and constant yield management.

2 Part I: Simulate the dynamics of the Beddington & May model

The analysis of Beddington & May is all done without specificying values of r and K. To do numerical simulations, we need to chose. As a starting point, pick $r = 2.5$ and $K = 1$. So effectively we're simulating the more specific model:

$$\frac{dN(t)}{dt} = 2.5\ N(t)(1 - N(t))$$

You are free to fiddle with these later if time allows. The Forward Euler method took our differential equation and approximated it with a difference equation, just with very small intervals.

$$N(t_{k+1}) = N(t_k) + [rN(t_k)(1 - N(t_k)/k) - EN(t_k)]\,\Delta t$$

We just need a little set up and a `for` loop with the right arithmetic and we can solve for the behavior of the population given some initial starting value and fishing effort E. For now, let's set the fishing effort $E = 0$.

With no noise and no fishing, this model has a stable equilibrium at the carrying capacity, K [Question: How do you prove that?]. So if you initialize the model at say, $N_0 = 0.25$ and run it for a long time, you should get something that looks like:

```
r_ <- 2.5
K_ <- 1
N_0_ <- K_*.25

dt <- 0.01

v_N <- vector(length=10/dt)
v_N[1] <- N_0_

for(index in 2:length(v_N)){
  N_i_ <- v_N[index-1]
  v_N[index] <- N_i_ + dt*((r_ * N_i_)*(1 - N_i_ / K_))
}

plot(v_N,type='l',ylim = c(0,1.5),main="r = 2.5; K = 1")
```

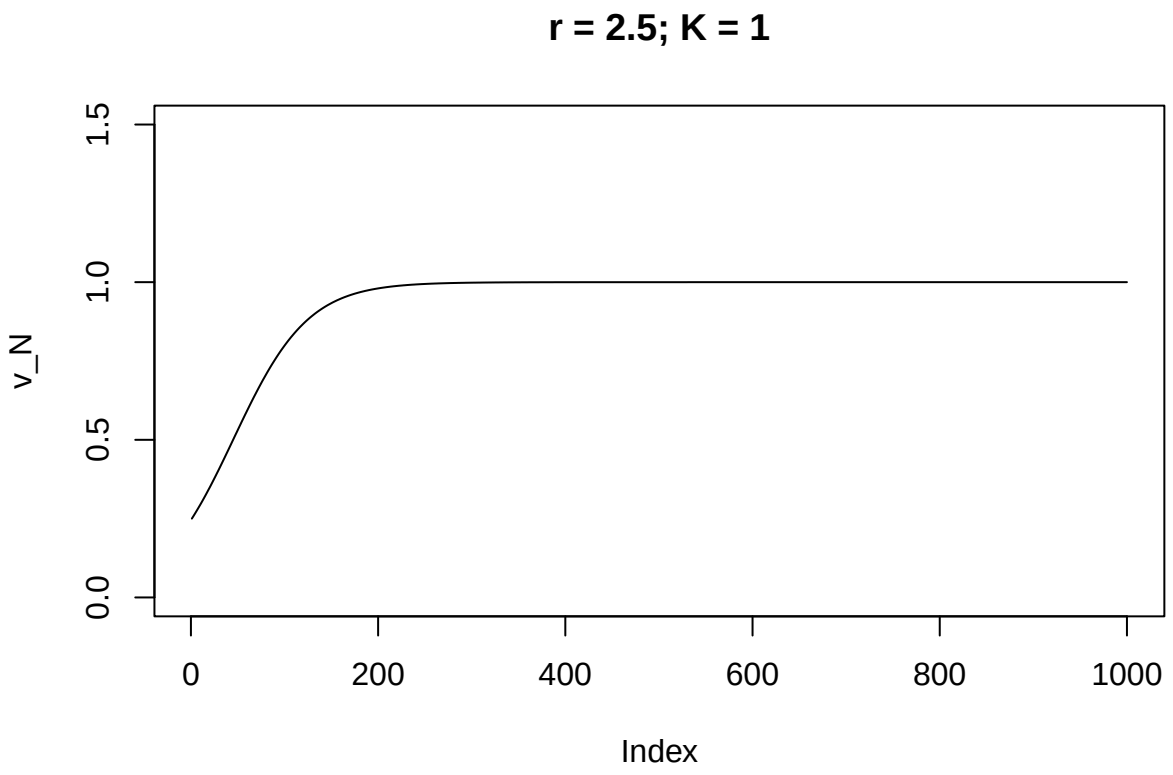


Figure 2: A single simulation of the B&M model relaxing to the stable equilibrium

If you decrease the growth rate, say, to $r = 2$, the rate at which the population approaches equilibrium is slower. This is essentially what the return time, T_R, quantifies.

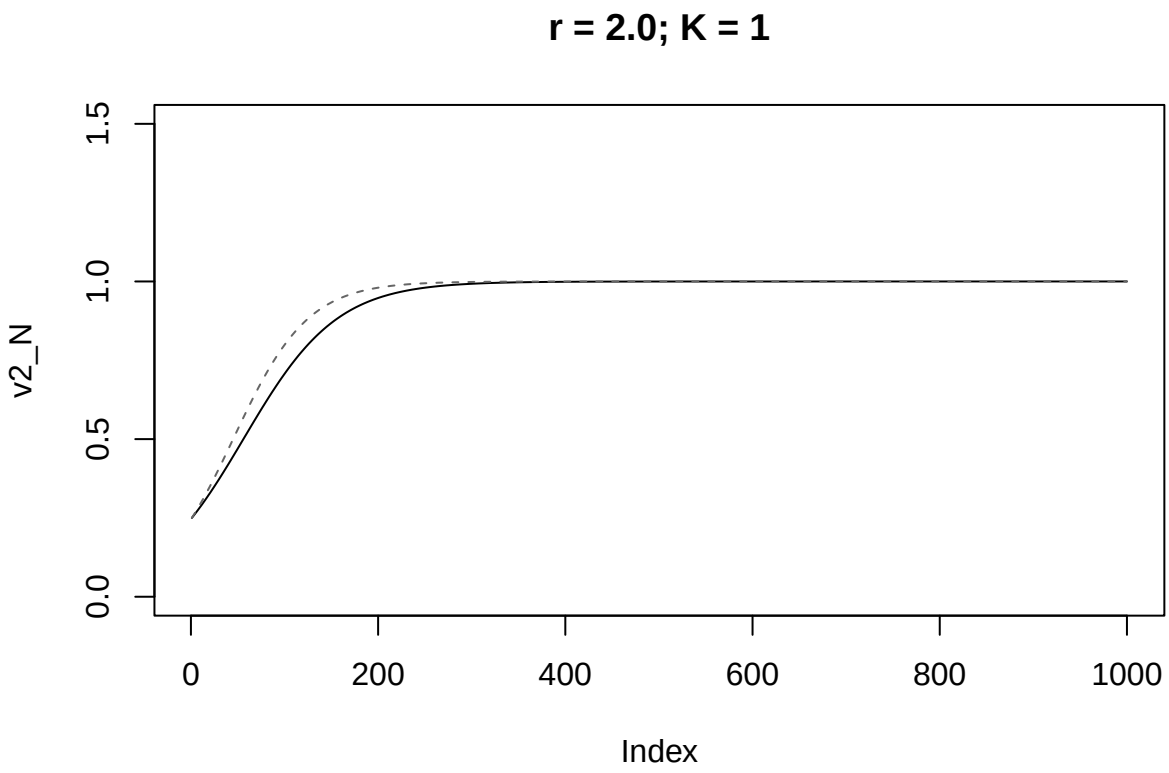


Figure 3: A second simulation of the B&M model with slower growth

2.1 Use a custom function

To start with, we've kept fishing effort at 0. However, the results of the paper pretain to the behavior of the "oversimplified population" as harvesting approaches the theoretical optimum. Thus we want to be able to simulate the population repeatedly with different values of E.

Suppose we wanted to add 1 to a number. Over and over. We write a function that takes a number and returns that number plus 1.

```
f_demo <- function(x){
  y = x+1
  return(y)
}
```

Now if we can ask the value of a few numbers.

```
f_demo(1)
```

```
## [1] 2
```

```
f_demo(5)
```

```
## [1] 6
```

```
f_demo(1.5)
```

```
## [1] 2.5
```

We can even ask it the value for a bunch of numbers at once*

```
f_demo(c(1,5,1.5))
```

```
## [1] 2.0 6.0 2.5
```

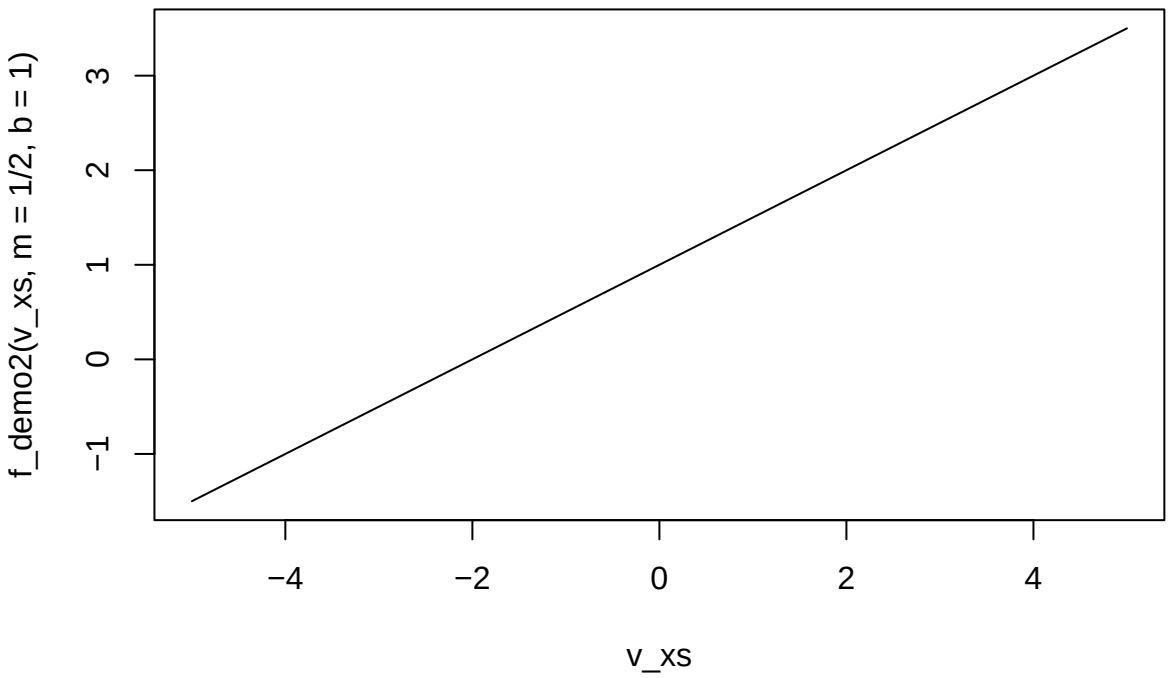
*This is actually something that isn't possible in other languages that is a double edged sword in R. Generally we don't have to tell R what kind of input we're giving a function. R will just try to make sense of it. In some languages, we would have had to specify if the input were going to be an integer, double precision (decimal), a vector, etc. And hey, the answer looks slightly different!

Just like built in R functions aren't limited to one argument, like `cor()`, custom functions are not limited to one argument. If we want to multiply x by a number, then add a number to it, we could write:

```
f_demo2 <- function(x,m,b){
  y = m*x+b
  return(y)
}
```

Look it's a line!

```
v_xs <- seq(-5,5,by=0.01)
plot(v_xs,f_demo2(v_xs,m=1/2,b=1),type='l')
```



TASK 1: Write a custom function to calculate the value of $\frac{dN}{dt} = rN(t)(1 - N(t)/k) - EN(t)$

The function should take as arguements the current stock-size N, level of fishing effort E and model parameters r, k. Use the custom R function to plot the growth rate dN/dt for a few different sets of values for parameters r and K. How do the parameters affect the shape?

TASK 2: Rewrite the `for` loop above to simulate the population dynamics but use the custom function to get the rate of change of the population.

Use this code to reproduce the above Lab Figure 3 comparing model behavior under two different growth rates.

2.2 Creating the experimental simulation as a "module"

We want to build up modular pieces for the numerical experiments in a deliberate way. First, we write script that performs the essential task a single time for a single set of parameters. Once we convince ourselves that code is working as intended (e.g. by plotting the results, by repeating the calculation with different parameters), we take the code for the single task and turn it into a function that repeats the task but allows for parameters to be changed– a little module. Then we convince ourselves that the modularized code works (e.g. by reproducing the initial script produced). There's a number of reasons for this strategy. One is that diagnosing problems in the code (debugging) is easier outside a function or `for` loop than inside. Some of you may have already experienced this!

The formatting style of code can also be strategic but also comes down to personal preference. Below, we're going to use a long descriptor for the function that simulates the Beddington & May 1977 logistic

model with constant effort, `f_sim_BM_logistic_CE()`. In this lab (and the original paper) there are a few different model structures considered and compared in the *in silico* experiments. Thus, we could also name our functions `f_sim_model_1()`, `f_sim_model_2()`, etc. and describe in the markdown text that "model 1" corresponds to constant effort, "model 2" corresponds to constant yield, etc.

```
dNdt_BM_logistic_CE <- function(N,r_,K_,Effort) {((r_ * N)*(1 - N / K_) - N*Effort)}

f_sim_BM_logistic_CE <- function(N_0,Effort=0,r_=2.5,K_=1,time_to_stop=10){

  dt <- 0.01

  v_N <- vector(length=time_to_stop/dt)
  v_N[1] <- N_0

  for(iter_i in 2:length(v_N)){
    N_i <- v_N[iter_i-1]
    v_N[iter_i] <- N_i + dt*dNdt_BM_logistic_CE(N_i,r_,K_,Effort)
  } # for(iter)

  return(v_N)
}
```

TASK 3: Now write a function that performs the `for` loop simulation of the population dynamics.

> Include meta-parameters of the simulation for the time-step Δ_t, the initial value of the stock-size N_0 and the total length of the simulation in the time units of the model. Plot the simulated dynamics (N as a function of time) for three values of fishing effort E.

Important connection to make here to the paper. So far, we have coded one of three harvesting behaviors that are discussed in the paper. We have assumed there is *constant effort*. This corresponds to the (a) curve in Figures 1 and 2. The conclusions of the paper also depend on the model behavior under *constant yield* behavior.

2.3 Exploring "Characteristic Return Time" under constant effort.

We almost have all the ingredients in place now to reproduce the (a) curve of B&M 1977 Fig 2. Currently, we have a function to simulate the population dynamics of the model under different model parameters (like growth rate `r`) and simulation parameters (like initial abundance `N_0`. Recall our strategy for approximating the "characteristic return time".

> Each single iteration of the experiment, then, will consist of simulating the population N using the integration scheme, with an initial abundance of either $N^*(E) + \Delta N$ or $N^*(E) - \Delta N$. This will generate a time-series of population abundances. You don't need to record each time-series for each single iteration, however. The important data to record will be the number of time-steps it took for $|N(t) - N^*(E)| < \Delta N/2$.

VERY IMPORTANT DETAIL. The equilibrium abundance depends on the level of fishing. You need to account for this! You can either write code to find the equilibrium (HINT: where the derivative = 0!)... or you can take the algebraic expression from the paper (Eqn 2).

$$N^*(E) = K\left(1 - E/r\right)$$

```
f_N_equil_E <- function(Effort,r,K){ K * (1 - Effort/r)}

delta_N_test1 <- 0.2
r_test1 <- 2.5
K_test1 <- 1

E_test1 <- 0.2

N_equil_test1 <- f_N_equil_E(E_test1,r_test1,K_test1)
N_0_test1 <- N_equil_test1 - delta_N_test1

v_N_test1 <- f_sim_BM_logistic_CE(N_0_test1,Effort=E_test1,r_ = r_test1,K_ = K_test1)
```

We also need an R expression to give us the number of time steps before $|N(t) - N^*(E)| < \Delta N/2$ for the simulated trajectory. If we have a simulated trajectory, equilibrium abundance, and perturbation size `delta_N` taken from above, then

```
which(abs(v_N_test1 - N_equil_test1) < delta_N_test1 / exp(1))
```

tells us all the indices of `v_N_test1` that have decayed past $1/e$, and the minimum of that list is the first time index that has "returned" to within $1/e$.

```
return_time_test1 <- min(which(abs(v_N_test1 - N_equil_test1) < delta_N_test1 / exp(1)))
```

Does that look right?

```
plot(v_N_test1,xlab = "time step",ylab="N",type='l')
abline(h=N_equil_test1,lty=2)
points(return_time_test1,v_N_test1[return_time_test1],pch=8,cex=2,col="blue")
```

Optionally, you can compare these calculations to the mathematically derived expression of the original Beddington & May paper, which the authors describe (footnote 10) as " T_R is essentially the reciprocal of the real part of the dominant eigenvalue". They give a formula for it, which is based on derivation in a previous paper. The ratio of the return time with a harvest effort E to the return time with no harvest is:

$$\frac{T_R(E)}{T_R(0)} = (1 - E/r)^{-1}$$

To reproduce B&M Figure 2, we need to generate the characteristic return time across effort.

```
f_N_equil_E <- function(Effort,r,K){ K * (1 - Effort/r)}

f_sim_return_time_CE <- function(delta_N,Effort,r_=2.5,K_=1){

  N_equil <- f_N_equil_E(Effort,r_,K_)
  N_0 <- N_equil - delta_N

  v_N <- f_sim_BM_logistic_CE(N_0,Effort=Effort,r_ = r_,K_ = K_)
  return_time <- min(which(abs(v_N - N_equil) < delta_N / exp(1)))
  return(return_time)
}
```

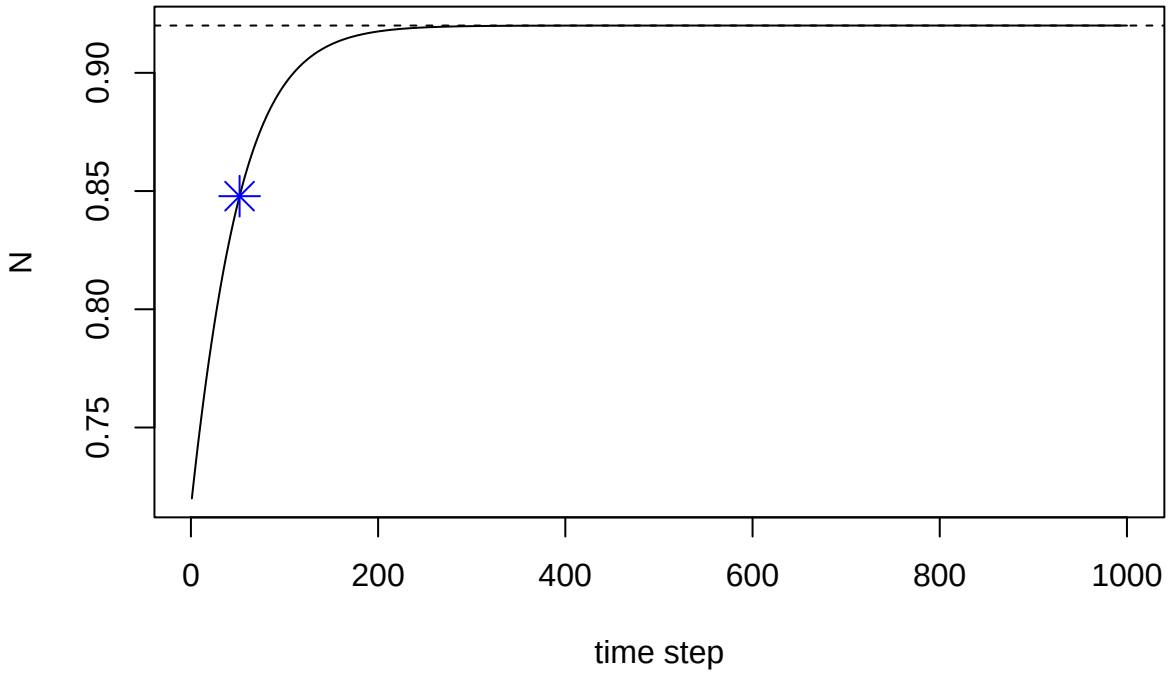


Figure 4: Plot of return time for single experiment iteration.

TASK 4: Make a code module (write a function) to calculate the "characteristic return time" of a simulated perturbation.

Reproduce the location of the blue asterix on Lab Figure 4 above using the modularized code.

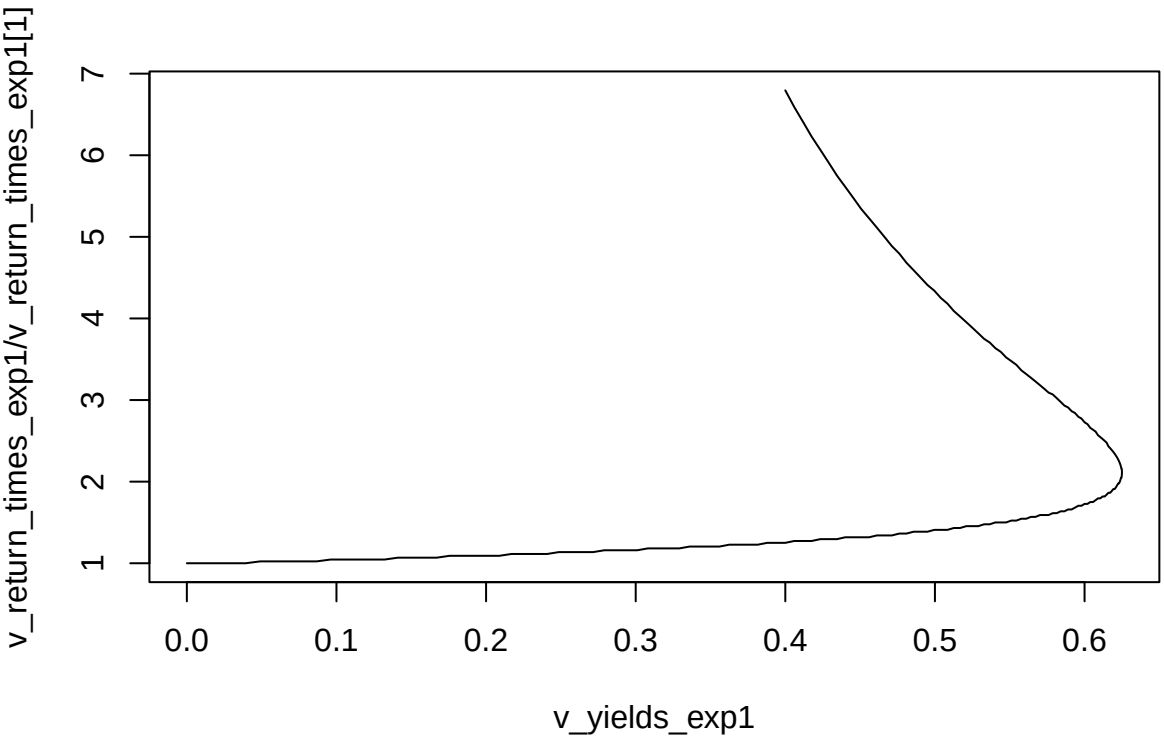
Now that we can write `for` loops around these to numerically generate data across treatments. We will use the function above inside, but need a few other things too. In particular, B&M plot the return time in terms of *yield* not *effort*. The constant effort model of harvest assumes that yield Y is a linear function of abundance N for a given level of constant effort E. So we need to put something like this in the code module:

```
Y_i <- N_equil_E_i * E_i
```

Otherwise, we are now constructing a `for` loop similar to Lab 1, where each time through the experimental treatment is slightly different. We need to create a list of efforts to "loop" over, a vector for storing return times for each experiment, and a vector of storing yields for each treatment. Put it all together, and you have a numerical experiment!

NOTE in previous version, the final run in the loop was producing an error. The equilibrium abundance at `Effort = 2` is $N^* = 0.2$, and we were perturbing the population by a magnitude of 0.2. This is a problem! The two fixes are to either do perturbations in the positive direct, $N_0 = N^* + \Delta N$ or to use a smaller ΔN.

```
r_exp1 <- 2.5
K_exp1 <- 1

list_of_efforts <- seq(0,2,by=0.01)

v_return_times_exp1 <- vector()
v_yields_exp1 <- vector()

for(E_i in list_of_efforts){

  N_equil_E_i <- f_N_equil_E(E_i,r_exp1,K_exp1)
  return_time_E_i <- f_sim_return_time_CE(delta_N = 0.1,Effort = E_i,r_ = r_exp1,K_=K_exp1)

  v_yields_exp1 <- c(v_yields_exp1, N_equil_E_i * E_i)
  v_return_times_exp1 <- c(v_return_times_exp1,return_time_E_i)

}

plot(v_yields_exp1,v_return_times_exp1/v_return_times_exp1[1],type='l')
```



TASK 5: Iterate the return time calculation for effort ranging from 0-1.5.

> This should produce something with the same general shape as curve (a) in B&M Figure 2. Note that at a certain point, R starts complaining about missing arguments. What gives? (Hint: What has happened to the return time relative to the length of the simulation). What do we change in our code to fix that?

3 Part II: Exploring "Characteristic Return Time" under constant yield.

At this point, we have reproduced the behavior of the *constant effort* version of their logistic model. However, key to their argument is the dangerous dynamics produced under a philosophy of Maximum Sustainable Yield when the management is framed in terms of a limit on *yield*.

3.1 Adapt the machinery from Part I

3.1.1 Planning

Up to this point in the lab, the instructions have been quite lengthy. Now it is time to push our understanding of the code a bit. We can reuse much of the above that was spelled out explicitly, but need to keep in mind what needs to be modified for a *constant yield* paradigm of harvest. Before diving in, try to think through all the pieces first.

TASK 6: Make a list of all the pieces (modules and scripts) we built above for *constant effort* that we need to have for *constant yield*.

Note which pieces need to be modified and which do not. (There should be a mix).

Check this list with a neighbor, TF, or instructor.

3.1.2 Execute

Once you are set, proceed!

TASK 7: Reproduce each module in sequence to apply to the *constant yield* version, and check the code is giving you reasonable behavior in the same way we did for the *constant effort* steps.

For functions you need to modify, give them new names but with an obvious correspondence to what you used for *constant effort*. For example, we called the Make sure your code runs after you clear the output (use the little broom in the R-studio "Environment" tab). When you repurpose and modify code it can be easy to miss pieces and accidently reuse something you didn't intend! The equilibrium biomass under the *constant yield* version is a little bit more complicated than under *constant effort*. This is part of the message in B&M Figure 1. If we solve $\frac{dN}{dt} = 0 = rN^*(1 - N^*/k) - Y$ to get $N*(Y)$ we have a quadratic equation with two roots. This corresponds to the two intersections of the horizontal line (b) with the parabola defined by $\frac{dN}{dt}$. The larger root is a stable point so long as yield does not increase too high.

$$N^*(Y) = \frac{K}{2}\left(1 + \sqrt{1 - 4Y/rK}\right)$$

Finally,

TASK 8: Rerun the code you created for Experiment 1 (simulating return time across constant levels of effort), but now simulating across constant levels of yield.

3.2 Put it all together

Now is time to go and look through the original paper.

The key figure as far as we're concerned today is Fig. 2 (Upper Right panel in the plots on page numbered pg 464). Curve (a) corresponds to a constant effort policy and curve (b) corresponds to a constant yield policy.

TASK 9: Reproduce B&M Figure 2 with both curves (a) and (b) shown.

TASK 10: Reproduce the figure for a second set of model parameters.

Beddington and May did all their analysis in relative units, scaling everything based on the values of growth rate and carrying capacity. Therefore, you should get a similar pattern if you change units, although the curves may scale and distort.

4 Optional Extensions

In the constant *effort* version of the model, we go on to calculate the equilibrium yield from the effort and equilibrium population size, $Y^* = EN^*$. In the constant *yield* version of the model, we can similarly derive the effort needed to achieve the yield. What is happening to the fishing effort as yield forces the characteristic return time to infinity?

We have focused on Figure 2, which shows the resilience of the a stock under constant effort and constant yield harvesting. The authors also go on to compute average yield, shown in Figure 3. Reread carefully and see if you can reproduce that as well. Start with the solid line.

4.1 Further Interest

In their analysis, the authors considered environmental fluctuations as modifying the growth rate but not the carrying capacity of the population. What happens if you put environmental variability onto the carrying capacity instead?

5 Reflection Questions

- What does this say about MSY management in natural systems?
- Figure 3 shows the average yield [If you completed the optional extension, it is relevant here]. What point on this graph is a more realistic "maximum sustainable yield"?