LAB 5 - Occam's Fishhook

ER Deyle

Fall 2024; Marine Semester Block 3

Introduction

The purpose of this lab is to investigate the statistical machinery for comparing model structures while interfacing with the fishery and ecological notions of population crashes or collapses. One of the simplest ways to model population dynamics with crashes are fisheries models that include "depensation"; these are also sometimes called "sigmoidal" population dynamic models (when you plot them, it will be clear why!).

Fisheries context

Depensation in models creates a *minimum viable population* size. In these models, 0 becomes a stable equilibrium point, so below a certain threshold, the population will continue to shrink to extinction. The fishery notion of "collapse" is distinct from the strict notion of a minimum viable population size, but does share some sense of a critical threshold below which the stock cannot easily recover.

The forms for depensation in fisheries models have been described previously, but they are rarely used. There is a notion from previous studies that these models are not useful for fisheries management due to a lack of evidence of suitability in model comparisons. However, "Can we accurately model a population process?" is not the same question as "Does a population process exist?" and the model comparison exercise can be questionable grounds for asserting ecology realities!

Computational approach

The goal of this lab will be to create a series of simulations of population dynamics that are prone to collapse (due to a depensatory model equation) to see if the model comparison exercise is a robust approach to determining the underlying population processes.

The statistical machinery is maximum likelihood estimation, which boils down to writing functions to describe the statistical likelihood of data under a model parameterization and optimization methods to determine the model parameterization that maximizes that likelihood.

Part I: Pieces in Place

Model Simulations

The model we'll begin with is framed with the simplest age structure; current recruits are determined by a function of spawner abundance $R[i] = g(S[i])$, and future spawner abundance $S[i+1]$ is determined by fixed survival rate Sa of current spawners and current recruits. We entertain two recruitment functions: the standard Ricker and a modified Ricker that has a sigmoidal shape to capture Allee affects.

$$g_{Ricker}(S) = S\exp(r(1-S/K))$$
$$g_{S-Ricker}(S) = S^d\exp(r(1-S/K))$$

This function defines the recruitment in the age structured model. For the adults, we also want to include a fishing mortality rate, F_m. If we assume fishing happens at the end of spawning season, we just multiply the equation for next years stock $S[i+1]$ by an additional factor of $(1-F_m)$, i.e. the fraction of adults that survive the fishing with mortality rate F_m.

$$R[i] = g(S[i])$$
$$S[i+1] = (R[i] + S_A S[i])(1-F_m)$$

Function specification

In the R code, we define functions so that they are properly nested, i.e. simpler models can be derived from more complex models by setting one or more parameters in the more complex model to zero. We set up these functions a little different from before, putting all the parameters together in a vector "mu". This is a common syntax for a bunch of R packages and tools involved in model simulation and fitting.

```
# recruit function
gR_SRicker <- function(S,mu){ with(mu,  S^(1+delta)*exp(r*(1-S/K)) ) }
# spawner function
gS_SRicker <- function(S,Fm,mu){with(mu,( gR_SRicker(S,mu) + Sa*S) )}
```

The `with()` wrapper lets us refer to elements of *mu* by their name only, just like the columns of a data.frame once we're in a string of `ggplot` commands. A quick example with two numbers, `a` and `b`:

```
a = 1
b = 2

a+b
```

```
## [1] 3
```

```
remove("a","b")

a+b
```

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

```
v_params <- list(a=1,b=2)

a+b
```

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

```
with(v_params,a+b)
```

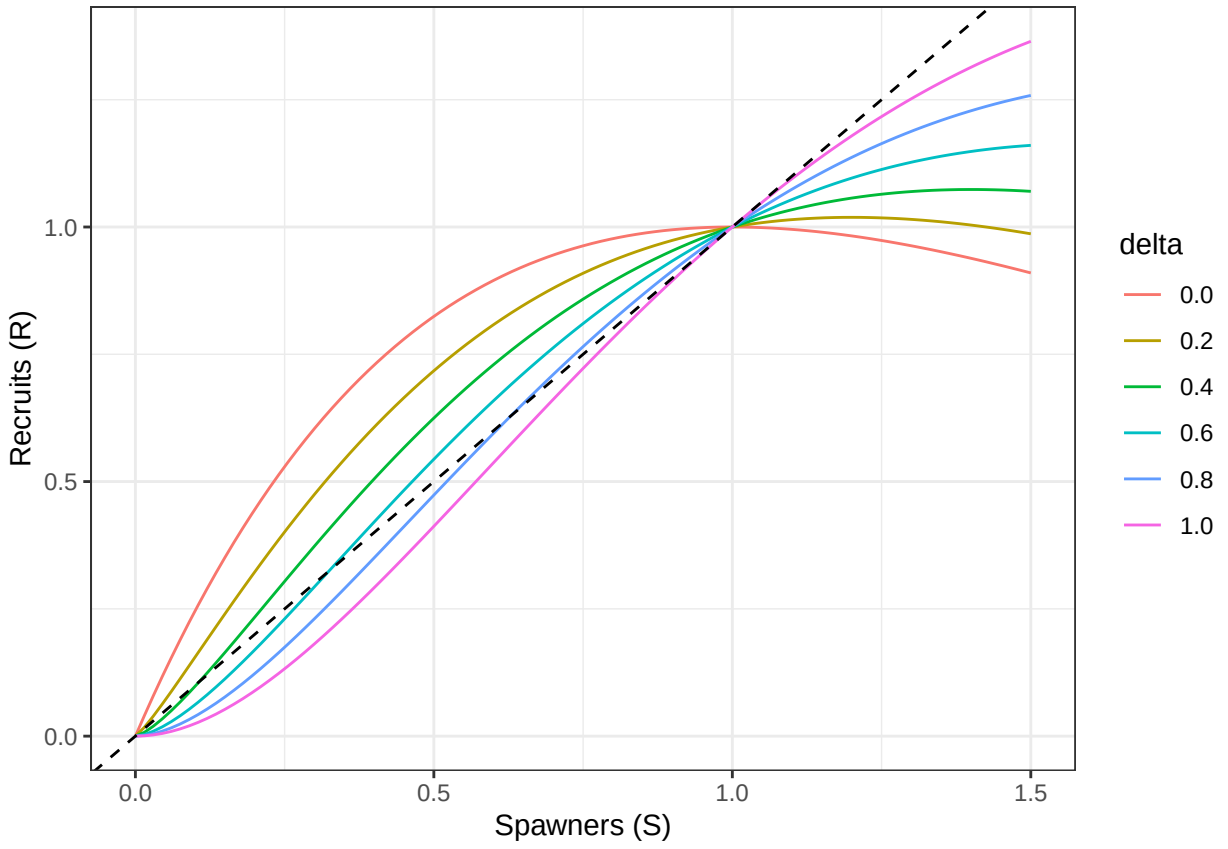
```
## [1] 3
```

Function visualization

Let's look at the effect of tuning the exponent on the sigmoidal "S-Ricker" model.

```
gR_SRicker_fix_r_K <- function(S,delta){ S^(1+delta)*exp( (1 - (S) ) ) }
```

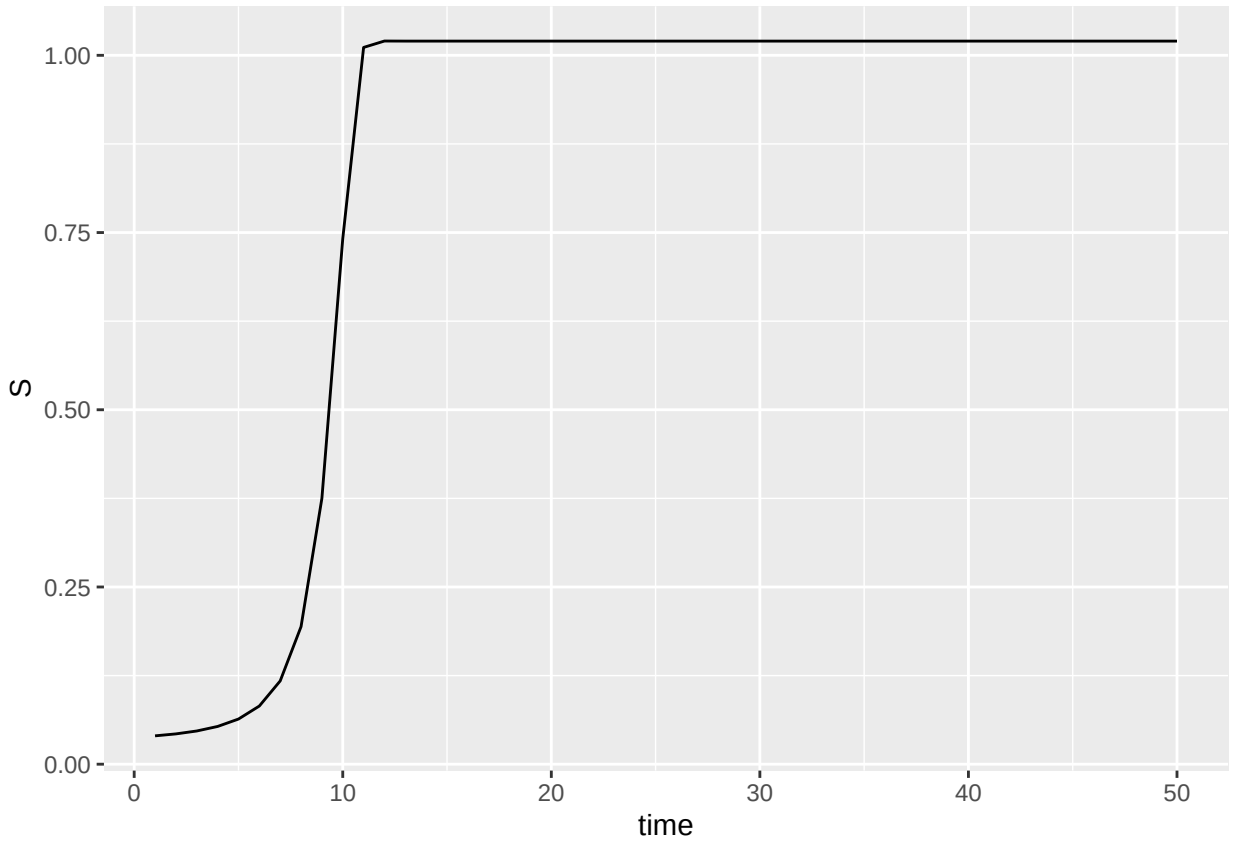
```
ggplot() +
  geom_function(fun= ~ gR_SRicker_fix_r_K(.x,0),aes(color="0.0")) +
  geom_function(fun= ~ gR_SRicker_fix_r_K(.x,0.2),aes(color="0.2")) +
  geom_function(fun= ~ gR_SRicker_fix_r_K(.x,0.4),aes(color="0.4")) +
  geom_function(fun= ~ gR_SRicker_fix_r_K(.x,0.6),aes(color="0.6")) +
  geom_function(fun= ~ gR_SRicker_fix_r_K(.x,0.8),aes(color="0.8")) +
  geom_function(fun= ~ gR_SRicker_fix_r_K(.x,1.0),aes(color="1.0")) +
  geom_abline(slope=1,intercept=0,lty=2) +
  theme_bw() +
  xlim(0,1.5) + labs(x="Spawners (S)",y="Recruits (R)",color="delta")
```



Noise-free Function simulation

Later on, we're going to study the behavior of this model under fishing pressure and stochastic variation. To build up the R code and our intuition, though, we begin with noise free simulation. Below is code to perform a single simulation of the dynamics.

```r
n_total <- 50

mu_test <- list(r=2.6,K=3/4,delta=1,Sa=.6)

data <- data.frame(S = numeric(n_total), R = numeric(n_total))

S0 <- 0.04
R0 <- gR_SRicker(S0,mu_test)

data$S[1] <- S0
data$R[1] <- R0

i_model <- 1

while(i_model < n_total){
  i_model <- i_model + 1

  R <- gR_SRicker(S0,mu_test) # Could add recruit observation error with factor like (1 + .1*rnorm(1))
  S <- with(mu_test,(R + Sa*S0))

  data[i_model,] <- c(S,R)

  S0 <- S
  R0 <- R
}

data <- cbind(data.frame(time=1:n_total),data)

ggplot(data,aes(x=time,y=S))+ geom_line()
```



As in several of the previous labs, it will be easiest to proceed with *in silica* experimentation if you create a function that can reproduce the simulation, but for different model parameters (r, K, δ, Sa) and different simulation parameters (starting conditions, number of iterations).

```
do_SRicker_sim_no_noise <- function(S0,mu,n_iter){

  ## YOUR CODE HERE

  return(SIMULATION DATA IN A DATA FRAME)

}
```

Once you have done that, you can create a `for` loop over a parameter and generate multiple trajectories to compare! For example

```
list_S0_try <- c(0.01,0.1,1.0)

l_simulated_data <- vector(mode="list")

for(S0_i in list_S0_try){

  df_sim_i <- do_SRicker_sim_no_noise(S0_i,mu=list(r=2.6,K=3/4,delta=1.0,Sa=.6),n_iter=50)
  df_sim_i$S0_value <- as.character(S0_i)

  l_simulated_data <- c(l_simulated_data,list(df_sim_i)) # need to wrap the new data.frame in list()
}
```

Now that loop should give you a list where each list element is the resulting simulated trajectory for each of the S_0 values being "tried". `ggplot` really likes data to be all in one data-frame. This can be accomplished a few ways. We can use `rbind` ("row bind") to paste them together end-to-end.

```
df_simulated_data <- rbind(l_simulated_data[[1]],l_simulated_data[[2]],l_simulated_data[[3]])
```

However if we ran more than 3 different values this would be a lot to write out. We can also use `do.call()` to take a list and input each element as an additional arguement to a function. Type `?do.call`. The first argument is the function we want to use; the second argument is the list of arguments we want to plug in. So,

```
df_simulated_data <- do.call(rbind,l_simulated_data)
```

Now that your final, aggregated data.frame has a time column and the `S0_value` stored, we can use ggplot syntax to plot each experiment as a line with a different color for each `S0_value`.

```
ggplot(df_simulated_data,aes(x=time,y=S,color=as.character(S0_value))) +
  geom_line() +
  labs(x="",y="",color="Initial Condition")
```

TASK 1: Simulate a single parameterization of the sigmoidal Ricker over a range of at least 10 starting values, e.g. S_0 = seq(0.01,0.1,by=0.01). Collect the simulations in a single data-frame so that you can use ggplot to visualize all the trajectories at once but use, e.g. different colors. (Remember to describe in text what your plot shows.) You'll need to include a lot of the machinery from the previous few pages in your code in order to make this work.

> Optional: once the function is set up you can also experiment with different parameters of the sigmoidal Ricker. Try decreasing *delta* or increasing R and see how the trajectories of the initial conditions change.

Function simulation with developing fishery

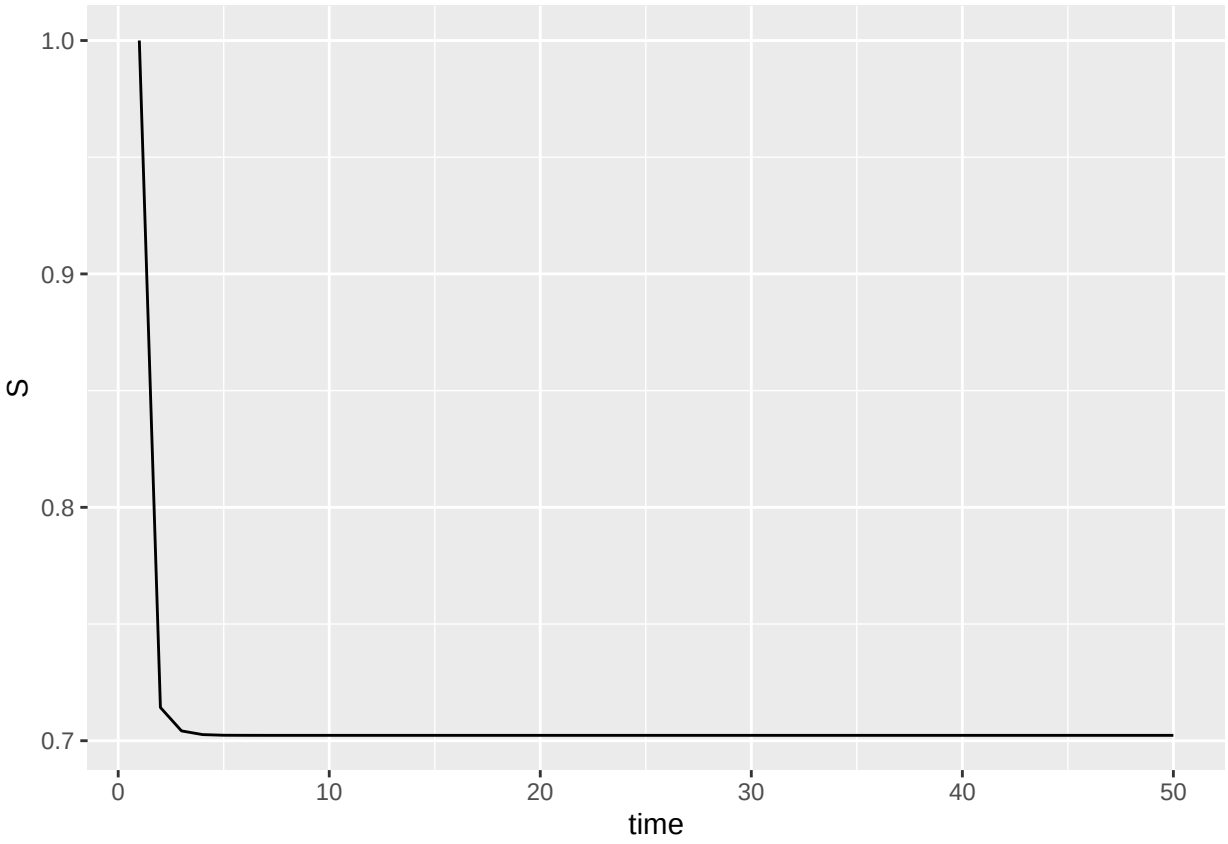
We can additionally include fishing mortality by subtracting off $F_m \times S$ at the end of the equation for S.

$$R[i] = g(S[i])$$
$$S[i+1] = (R[i] + S_A S[i])(1 - F_m)$$

When we do this, code for a single simulation will look something like this.

```
n_total <- 50

mu_test <- list(r=2.6,K=3/4,delta=1.0,Sa=.6)
# mu_test <- list(r=3.5,K=3,delta=1.0,Sa=.45)
Fm <- 0.3

data <- data.frame(S = numeric(n_total), R = numeric(n_total))

S0 <- 1.0
R0 <- gR_SRicker(S0,mu_test)
```

```r
data$S[1] <- S0
data$R[1] <- R0

i_model <- 1

while(i_model < n_total){
  i_model <- i_model + 1

  R <- gR_SRicker(S0,mu_test)
  S <- with(mu_test,(R + Sa*S0)*(1-Fm))

  data[i_model,] <- c(S,R)

  S0 <- S
  R0 <- R
}

data <- cbind(data.frame(time=1:n_total),data)

ggplot(data,aes(x=time,y=S))+ geom_line()
```



Of course to do experiments, we once again should consider writing a custom function that will do the simulation for user-input parameters. Give it a similar but distinct name from the no-noise, no-fishing experiment you did above. Something like

```
do_SRicker_sim_fishing_no_noise <- function(S0,mu,n_iter){

  ## YOUR CODE HERE
  return(SIMULATION DATA IN A DATA FRAME)

}
```

TASK 2: Use your new function to do some simulations with the same starting S_0 but different levels of F_m, like 'list_Fm_values = seq(0,0.8,by=0.1). (Remember to describe in text what your plot shows.)

> This bears similarity to Beddington and May, but recall that in their demonstration, the fishery dynamics were pretty "safe" with a constant effort model of fishing enforcement rather than a constant yield model. Which are we implementing here?

Interpretation Depending on the parameter values, you can actually generate chaotic behavior of this model instead of stable equilibrium dynamics (particularly by cranking up the growth rate R). In that context, fishing in this model can actually act to stabilize the function. However, an issue lies below the surface; the more fishing pressure goes up, the smaller the fluctuations, yes, but just like in Beddington and May's example, the function is becoming more and more sensitive to external forcing. The true danger of collapse lies in a world with (random) fluctuations in the environment. You know, the real world!

Stochastic Function simulation

The above parameterizations just simulate equilibrium dynamics. If you recall the Beddington & May paper, their final ingredient was to show how the logistic model behaved in a "fluctuating environment". This idea of including some random variation or *stochasticity* appears in the later simulation-based papers we've seen so far as well (e.g. Perretti et al. 2013). We think of the above equation as a "deterministic skeleton", where the realized abundances change based on random fluctuations around that *deterministic* rule. In the style of Perretti, we can write e.g.

$$\hat{R}_{S-Ricker} = S\exp(r(1-S/K)+\epsilon)$$

where ϵ $N(0,\sigma_R^2)$. However, we can also model R as a random variable selected from a distribution with mean $\hat{R}_{S-Ricker}$. That is,

$$\hat{R} \sim Lognormal(R,\tau_R^2).$$

This is probably a slightly preferred form, particularly for interfacing with the `bbmle` tools for fitting maximum likelihood. Thus the recruitment function can be rewritten as

```
gR_SRicker <- function(S,mu){
  with(mu, rlnorm(1, meanlog=(1+delta)*log(S)+(r*(1-(S^(1))/K)), sdlog=tau_r) ) }
```

TASK 3: Repeat the simulations done on the noise-free function but with the stochastic version. Set $\tau_R^2 = 0$ to start with (did you get the same figure? you should!), then try a small amount of "noise", like $\tau_R^2 = 0.1$.

> Question: Is this modified version including "*process error*" or "*observation error*"?

Model Comparison

Now as we noted up above, the sigmoidal Ricker is a generalization of the standard Ricker. Or, put differently, the standard Ricker is **nested** in the sigmoidal Ricker. In fact, previous research has use the likelihood ratio test for nested models to ask if there's evidence of stock-collapse dynamics in real populations. However, this sort of analysis is tricky to interpret; we can see why for ourselves!

Approach:

- Generate data with the sigmoidal-Ricker.
- Use the likelihood ratio test to see if model comparison analysis can successfully discern the presence of "depensitory mechanics" (i.e. stock-collapse). *Note that we are just fitting the relationship between* $S[i]$ *and* $R[i+1]$, which will require modifying the data outputs we've been working with so far (to offset by a year).

To interface with the `bbmle` function `mle2`, the stochastic part of the recruitment function needs to get rewritten (at least this is how I get it to work!). The reason (I think) is that the function `rlnorm` generates individual random numbers that following the probability distribution, which is what you need to produce data simulating of the process, while `dlnorm` actually describes the probability density, which is what you need to calculate the **likelihood** of the results of the process.

```
gR_Allee <- function(S,mu){
  with(mu,rlnorm(1, meanlog=(1+delta)*log(S) + ( r * (1 - (S^(1))/K ) ),sdlog=tau_r) ) }
gS_Allee <- function(S,Fm,mu){
  with(mu,( gR_Allee(S,mu) + Sa*S) * ( 1 - Fm))}

gR_Ricker <- function(S,mu){
  with(mu, rlnorm(1, meanlog=(1)*log(S) + ( r * (1 - (S^(1))/K ) ),sdlog=tau_r) ) }
gS_Ricker <- function(S,Fm,mu){
  with(mu,( gR_Ricker(S,mu) + Sa*S) * ( 1 - Fm))}

mu_test <- list(r=2.1,K=1.,delta=0.6,Sa=.65,tau_r=0.15)

n_burn <- 50
n_total <- 150 + n_burn

data <- data.frame(S = numeric(n_total), R = numeric(n_total), Fm = numeric(n_total))

S <- 0.8
R <- gR_Allee(S,mu_test)

Fm <- 0.
dFm <- 0.005

for (i_model in 1:n_total){

  R <- gR_Allee(S,mu_test) # Could add observation error on recruits with factor like (1 + .1*rnorm(1))
  S <- with(mu_test,(R + Sa*S)*(1 - Fm)) # gS_Allee(S,Fm,mu_test)

  data[i_model,] <- c(S,R,Fm)
  Fm <- Fm + dFm
}

data <- cbind(data.frame(time=1:n_total),data)
```

```
# NOTE that to do the fit, we need to fit R[i+1] as a function of S[i]
data_fit_1 <- data.frame(time=61:110,S=data$S[61:110],R=data$R[62:111])
data_fit_2 <- data.frame(time=61:140,S=data$S[61:140],R=data$R[62:141])

mle2_fit.m0 <- bbmle::mle2(R ~ dlnorm((1+0)*log(S) + alpha - beta*(S^(1+0)),sdlog=tau_r),
                           # method = "SANN",
                           method = "BFGS",
                           control = list(maxit = 5000),
                           data = data_fit_1,
                           start = list(alpha=2.5,beta=1/2.2,tau_r=0.1))

mle2_fit.m1 <- bbmle::mle2(R ~ dlnorm((1+delta)*log(S) + alpha - beta*(S^(1)),sdlog=tau_r),
                           # method = "SANN",
                           method = "BFGS",
                           control = list(maxit = 5000),
                           data = data_fit_1,
                           start = list(alpha=2.5,beta=1/2.2,delta=1.0,tau_r=0.1))

out_anova <- bbmle::anova(mle2_fit.m0,mle2_fit.m1)
print(out_anova)
```

```
## Likelihood Ratio Tests
## Model 1: mle2_fit.m0, R~dlnorm((1+0)*log(S)+alpha-beta*(S^(1+
##           0)),sdlog=tau_r)
## Model 2: mle2_fit.m1, R~dlnorm((1+delta)*log(S)+
##           alpha-beta*(S^(1)),sdlog=tau_r)
##   Tot Df Deviance  Chisq Df Pr(>Chisq)
## 1      3  -49.530
## 2      4  -52.503 2.9732  1    0.08465 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
str(summary(out_anova))
```

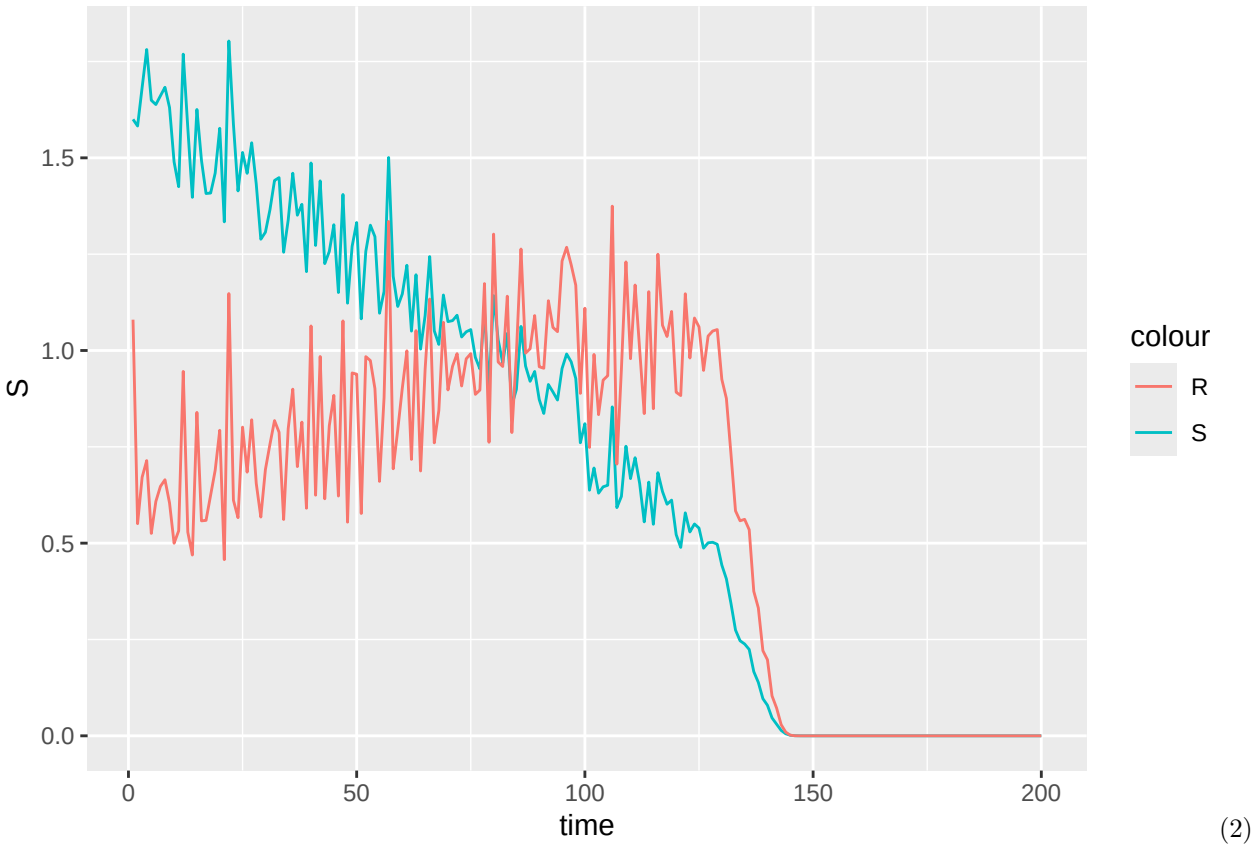
```
##  'table' chr [1:7, 1:5] "Min.   :3.00  " "1st Qu.:3.25  " "Median :3.50  " ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:7] "" "" "" "" ...
##   ..$ : chr [1:5] "    Tot Df" "   Deviance" "    Chisq" "      Df" ...
```

```
p_val <- out_anova[2,5]
```

Some plotting ideas:

(1) Visualize the time series coming out of the simulation

```
ggplot(data,aes(x=time,y=S)) + geom_line(aes(color="S")) + geom_line(aes(y=R,color="R"))
```

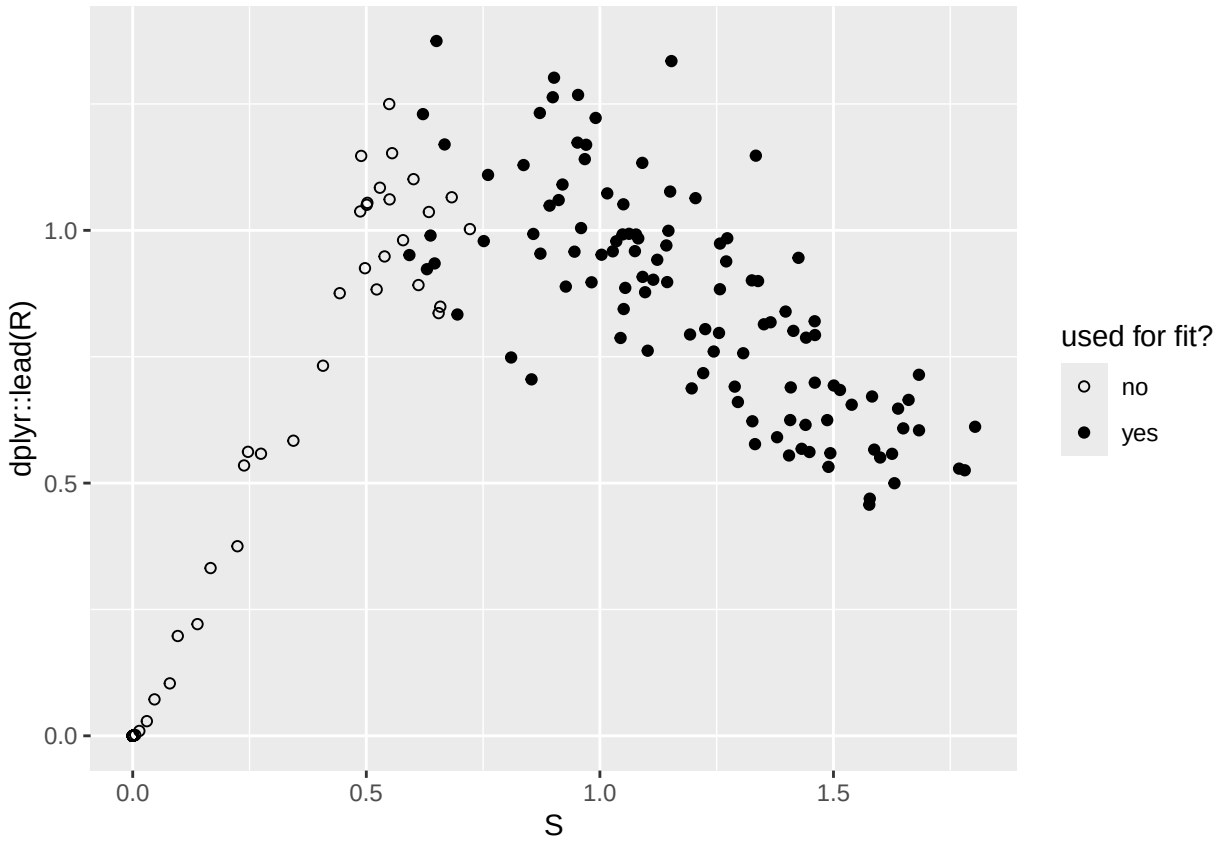


(2)

Examine the Stock-Recruitment relationship from data that is being used as the basis of the model comparison test (the anova bit)

```
data %>%
  mutate(library=time <= 110) %>%
  ggplot(aes(x=S,y=dplyr::lead(R),shape=library)) +
  geom_point() +
  scale_shape_manual(breaks=c(FALSE,TRUE),
                     values=c(1,19),
                     labels=c("no","yes")) +
  labs(shape="used for fit?")
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



TASK 4: Generate population dynamics data with the sigmoidal Ricker and delta = 0.6. Use the likelihood ratio test to see if model comparison can detect that delta > 0 (the anova test will give a Pr(>Chisq)). Repeat this many times (500) to see how the answer varies over different realizations of the stochastic model.

Optional Extension

If you use data closer to the "point of no return" (e.g. see `data_fit_2` in the example code chunk for the model fit and anova) does the result of (D) change?

Reflection questions:

> Think back to the Worm et al. and Branch et al. papers we described in class. What connections do you see between this lab and the debate between those papers over quantifying collapses in real ecosystems.

Final editorialization:

There are emerging machine-learning methods to test for stability (ala eigenvalues!). These offer a potentially minimal-assumptive framework for quantifying collapse potential.