

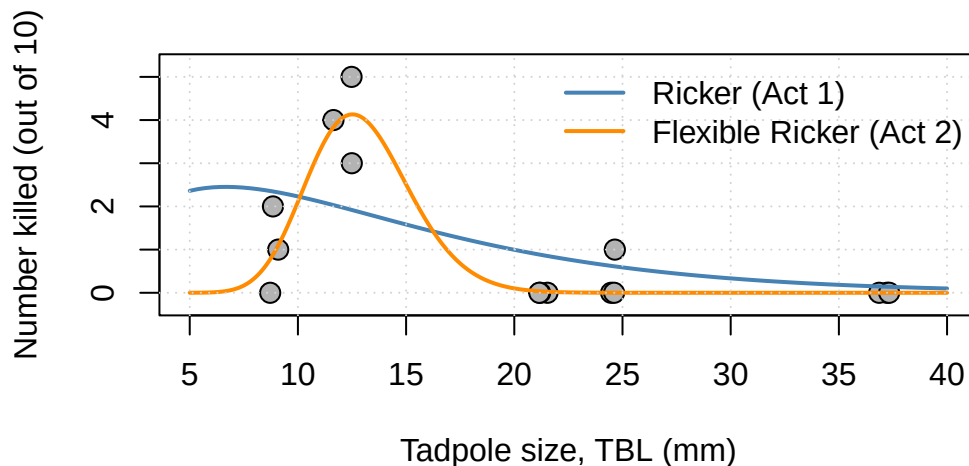
Lab 2, Act 3

How much should you trust the ‘best’ fit?

NRES 746 – name: _____ group: _____

Picking up where we left off

Across Act 1 and Act 2 you found several “best” curves: Act 1’s grid search gave something close to the best possible 2-parameter Ricker fit ($a = 1$, $b = 0.15$); Act 2’s flexible Ricker, once truly optimized, fit far better on paper – but its winning parameters ($a \approx 4 \times 10^{-19}$, $p = 12.52$, $c \approx 29$) don’t look anything like a natural, hand-picked guess (we haven’t talked about overfitting yet, but this kind of result ‘smells’ like overfitting!). So far, every SSR you’ve computed has been calculated against the 5 tank-size means. From here on we’ll work directly with the 15 raw tank observations instead, and compute SSR at the individual-tank level – the more standard approach, and one that will matter once we start resampling.



Today’s question: how much should you actually trust either of these “best” parameter sets? A single fit only tells you what happened to work for the data you happened to collect. Bootstrapping asks a sharper question: if you’d collected slightly different data (which you would, by chance, if you repeated the experiment), how much would your best-fit parameters move around? We’ll compare the plain Ricker against the flexible Ricker – not Holling IV, since its obviously over-fitted just from looking at the curve (that dramatic spike). Flexible Ricker’s problem is sneakier: the curve itself looked perfectly reasonable, and nothing seems terribly off unless you scrutinize the actual parameter values. That’s the kind of model fragility bootstrapping is good at catching.

Try this on paper first, before you open a laptop or start typing code. Part 1 and the checkpoint that follows work best this way – resist the urge to jump straight to R or an AI assistant.

Part 1: Plan the bootstrap (individual, ~15 min)

1a. Act 1's grid search checked about 2,300 combinations of (a, b) for the plain Ricker. If you wanted similar resolution (roughly 50-60 values per parameter) for the flexible Ricker's 3 parameters, about how many combinations would a comparable grid need?

Is a grid search still a practical way to find the best fit here? (the answer is probably obvious, but it's an important concept so let's spell it out here!) _____

For situations like this, R has a general-purpose numerical optimizer, `optim()`: give it a starting guess and a function to minimize (like your SSR function), and it searches nearby values and adjusts step by step until it can't improve further. It's much faster than a grid, but it can settle into a bad valley depending on where it starts. Below we'll use two ready-made, AI-generated functions, `fit_ricker()` and `fit_flex_ricker()`. Both take two arguments, in this order: a vector of TBL values (**x**) and a vector of Kill counts (**y**), the same length as **x** and paired with it position-by-position (the i -th entry of **y** is the observed kill count for the i -th entry of **x**) – the same shape as `frogs$TBL` and `frogs$Kill`, or a resampled version of them. Both functions return the best-fit parameters for whatever **x/y** pair you hand them; `fit_flex_ricker()` tries a few different starting points and keeps the best result, to guard against that same problem.

1b. Design the bootstrap. Working from the two fitting functions described above, write out – in plain English, step by step – the algorithm for bootstrapping the fitted parameters of each model, resampling from all 15 raw tank observations pooled together (not grouped by size). Think about: what you're resampling from, what has to happen at each replicate, what you need to store, and how many times you repeat the whole thing.

1c. Predict before you run anything. Which model do you expect to show a tighter (more consistent) spread of fitted parameters across bootstrap replicates: the plain Ricker, or the flexible Ricker? Why?

Checkpoint: finalize your plan (~5 min)

Reread your algorithm from 1b and your prediction from 1c. Would your plan actually work if you started typing it up right now? Revise anything that's incomplete or unclear before you move on – this is your last checkpoint before writing any code.

Part 2: Run the bootstrap (~20-25 min)

Type the two fitting functions below into your R script, then implement your bootstrap plan.

```
# x: vector of TBL values. y: vector of Kill counts, same length as x,
# paired position-by-position (e.g. frogs$TBL / frogs$Kill, or a resampled
# version of them with the same length). Both functions return a vector
# of best-fit parameters: fit_ricker() -> c(a, b); fit_flex_ricker() -> c(a, p, c)
fit_ricker <- function(x, y) {
  ssr <- function(par) sum((y - par[1]*x*exp(-par[2]*x))^2)
  optim(c(1, 0.15), ssr)$par
}

fit_flex_ricker <- function(x, y) {
  ssr <- function(par) {
    a <- par[1]; p <- par[2]; c <- par[3]
    if (p <= 0 || c <= 0 || a <= 0) return(1e10)
    pred <- a * x^c * exp(-c * x / p)
    if (any(!is.finite(pred))) return(1e10)
    sum((y - pred)^2)
  }
  starts <- list(c(1e-6, 12, 8), c(0.01, 12, 3), c(1, 15, 2))
  best <- NULL
  for (s in starts) {
    fit <- optim(s, ssr, control = list(maxit = 2000))
    if (is.null(best) || fit$value < best$value) best <- fit
  }
  best$par
}
```

2a. Implement your bootstrap: resample all 15 raw observations with replacement, refit both models to each resample using the functions above, and store the results. Run at least 200 replicates (should take well under a minute).

2b. Summarize the spread. For each parameter below, record the median and the interquartile range (IQR) – the span from the 25th to the 75th percentile, covering the middle half of your bootstrap values (R's `quantile()` or `IQR()` functions will get you there): the plain Ricker's peak location ($1/b$), and the flexible Ricker's peak location (p) and shape parameter (c).

Parameter	Median	IQR
Ricker peak ($1/b$)		
Flexible Ricker peak (p)		
Flexible Ricker shape (c)		

2c. Make two histograms: one of the plain Ricker’s bootstrapped peak locations ($1/b$, one value per replicate – about 200 numbers total), and a separate one of the flexible Ricker’s bootstrapped peak locations (p). `hist()` works fine for either; just hand it the vector of peak-location values you stored during the bootstrap.

Each histogram shows the full *distribution* of “best” answers your bootstrap turned up, not just the single median you recorded in 2b. A narrow, tall histogram means the model landed on nearly the same peak location almost every time, regardless of which 15 observations happened to get resampled – a stable, trustworthy estimate. A wide, spread-out histogram means the answer depended heavily on the luck of the resample – an estimate you should be more skeptical of.

Did the plain Ricker’s bootstrapped peak locations cluster anywhere near 12 mm, where the data actually peaks?

2d. Did the results match your prediction in 1c? Which model would you actually trust more, and for what – predicting the peak location, or predicting the overall shape of the curve? Is it possible for a model to be reliable about one of those and not the other? Is one of these clearly the better model to use going forward, or does it depend on what you actually want to do with it?

Reflection (~5 min)

Please take a minute to reflect on any areas of confusion (“muddiest points”) or topics you want to review or dust off going forward.
