Maximum likelihood Lab 2: Functional Responses

answer key

BIOHOPK 143H - Winter 2021

Aiming to characterize the trade-off between growth and survival, Vonesh and Bolker (2005) use experimental manipulations to estimate the effects of prey (tadpole) size and prey density on the rate of consumption by aquatic predators (a species of dragonfly, the "keyhole glider"). Here, we'll model the relationship between tadpole density and predation by fitting two forms of the functional response using their experimental data.

Recall that a **functional response** defines the relationship between prey density or number (the independent variable) and the rate of prey consumption by a predator (the dependent variable) - so, our x variable is the number of tadpole larvae in each tank, and our y variable is the number of tadpoles eaten over the study period. We have a fixed number of predators in each tank (3 dragonflies), and a fixed amount of time (14 days) that we observe our tadpoles for.

We might be tempted to just take the Normal negative log-likelihood functions we made for fitting regression models yesterday, and replace the regression equation with one of the many available functional response equations - but we can do better than this.

Specifically - we should use a binomial likelihood function (instead of a Normal one), for a few reasons. Take a minute to look up the binomial distribution and list a couple reasons why it's a better choice for this data than a Normal distribution:

> **The binomial distribution characterizes the sum of independent Bernoulli trials (trials with binary outcomes) - if we think of each tadpole representing a "trial," and the outcome for each "trial" is whether or not that tadpole was eaten, than the number of tadpoles eaten is Binomially distributed.**

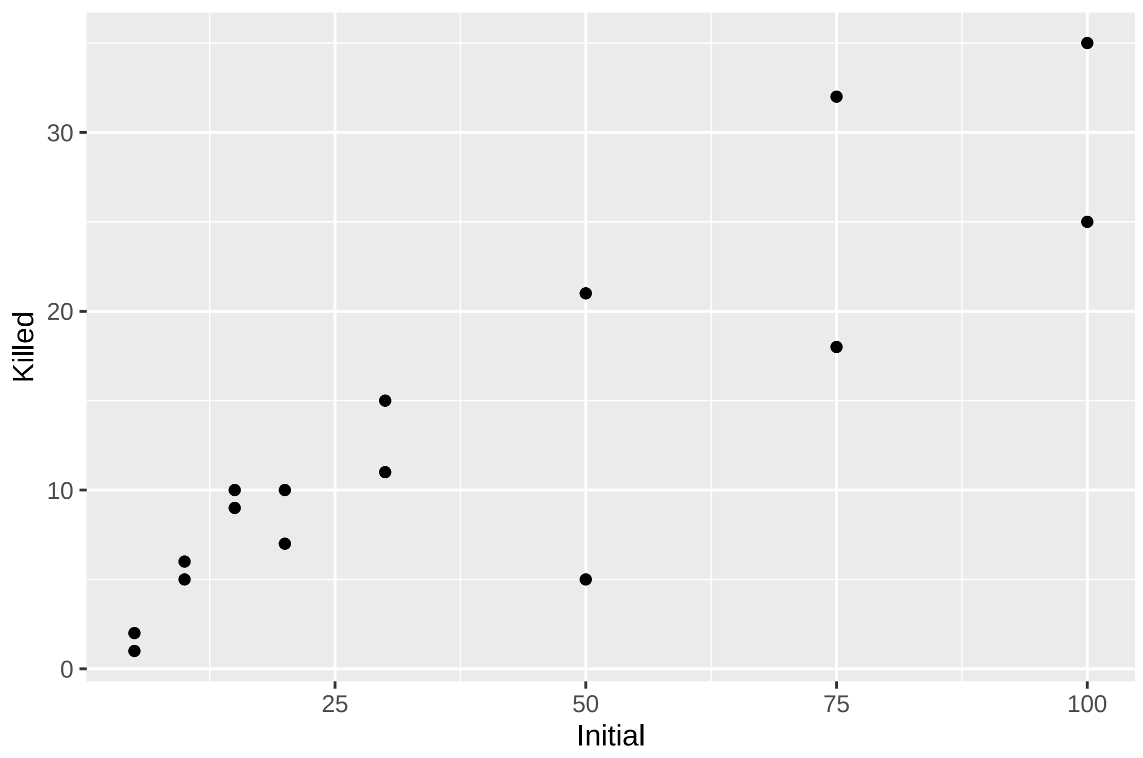
What are the parameters of the binomial distribution? List and describe them here:

> **The parameter n in the Binomial distribution represents the number of Bernoulli trials - here, that's the initial number of tadpoles in the tank. The parameter p is the probability of one of the binary outcomes in any given Bernoulli trial - in this case, if we take the two outcomes to be "eaten" or "not eaten," p is the probability that any individual tadpole is eaten.**

The data we need for this part of the lab is included in the `emdbook` package as `ReedfrogFuncresp` - the `Initial` column indicates the initial number of prey in the tank, and the `Killed` column indicates the number of prey consumed by the predator.

First, let's plot the relationship between the number of tadpoles eaten and the initial number of tadpoles:

```r
data(ReedfrogFuncresp)

ReedfrogFuncresp %>%
  ggplot(aes(Initial, Killed)) +
  geom_point()
```



The Holling Type II response

There are many hypothesized forms of the functional response, but perhaps the most ubiquitous is the Holling Type II response, which models the decline in predation rate at high prey densities as arising from two parameters: the **attack rate** of the predator, and the **handling time**, which is how long it takes a predator to consume an individual prey item. At low prey densities, the attack rate controls how many prey a predator consumes, but as prey density increases, the handling time becomes limiting, and the predation rate approaches as asymptote:

$$N_c = P \times t \times \frac{aN_0}{1 + ahN_0}$$

Where N_c is the number of prey consumed, N_0 is the initial number of prey, a is the attack rate, h is the handling time. P is the number of predators, and t is the amount of time over which predation is observed - these variables just scale the functional response.

It's straightforward to express the Holling Type II response as an R function:

```r
## Function describing Holling Type II functional response
holling2 <- function(N0, a, h, P, t) {
  P * t * (a * N0)/(1 + a * h * N0)
}
```

Now, let's come up with an equation for the likelihood of the data given the likelihood function of choice and our functional response:

$$N_c \sim \text{Binomial}(N_0, p_c)$$
$$p_c = \frac{P \times t \times \frac{aN_0}{1+ahN_0}}{N_0}$$

Here's a function for the negative log-likelihood under this model - take a minute to understand the code. The attack rate a and the handling time h must be positive, so we'll optimize these parameters on the log scale. Also, we know the number of predators P in each tank (3 dragonflies) and the amount of time t for which predation is observed (14 days), so we'll pass these as separate arguments instead of including them in our `par` argument.

Note that our function for p_c is actually only bounded on the lower end - it can't be lower than zero, but if we set the attack rate high enough it can produce a probability of consumption higher than one. As a workaround, we'll return a probability very close to 1 if the expected number of prey consumed exceeds the initial prey density):

```r
## Negative log likelihood
nll_holling2 <- function(par, P, t, N0, Nc) {

  a <- exp(par["log_a"]) ## attack rate
  h <- exp(par["log_h"]) ## handling time

  p_sim = pmin(holling2(N0, a, h, P, t)/N0, 0.999)
  -sum(dbinom(Nc, prob = p_sim, size = N0, log = TRUE))

}
```

Question: Why can't we use a probability of exactly 1 as our ceiling? What other strategies could we try to bound the probability of consumption within (0, 1)?

> If the probability of consumption were estimated as exactly 1 for any of our data points, this would mean that if any prey were not consumed in that trial, the probability of that data point is zero. When we're obtaining the negative log likelihood, that would translate to $-log(0) = \infty$, so we wouldn't be able to evaluate the likelihood. We might try fitting a different function which is naturally bounded within (0,1), or we could try constrained optimization, or we could use a different transformation (like a sigmoid on some interval) to constrain the attack rate to a certain range.

Optional: Come up with a function with a similar shape to the Holling Type II function, and with two parameters a and h which control the slope and asymptote, but which is constrained to the interval (0, 1) so long as a and h are positive:

> The function $(1 - e^{-aN0})/(1 + h)$ satisfies this constraint for a, h > 0

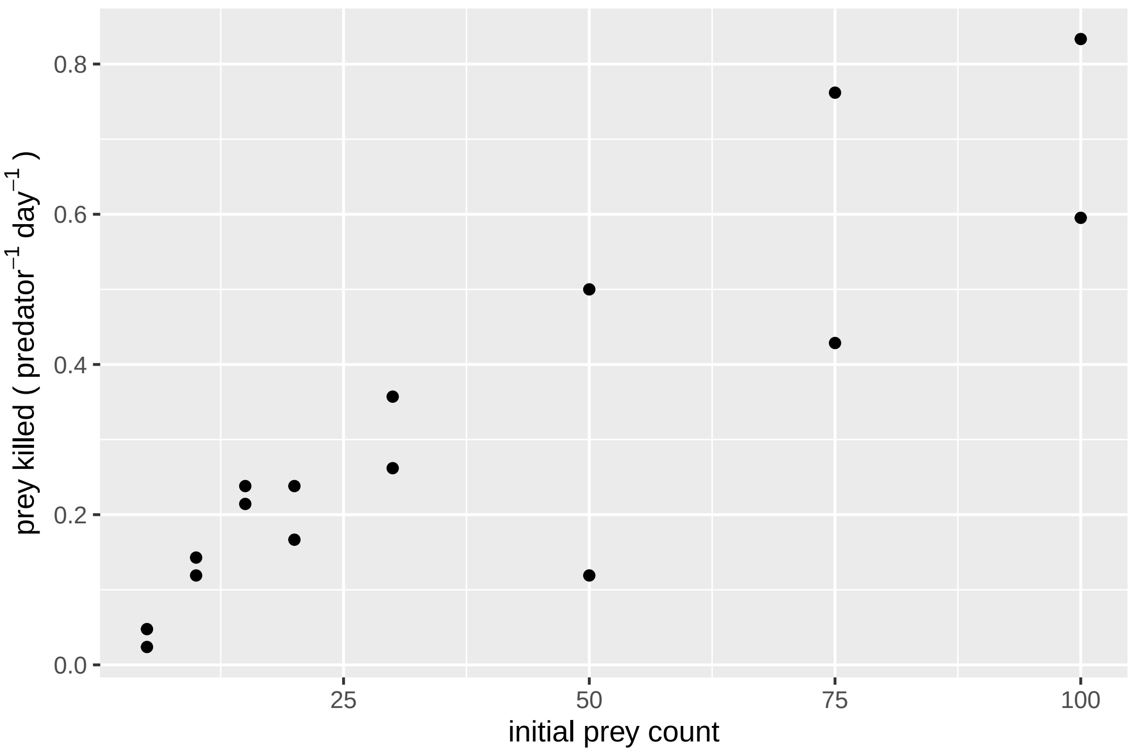
Choosing initial values

When we ran a linear regression on Monday, we didn't pay much attention to the initial parameter values that we passed to `optim`. However, for more complex functions, they can matter quite a bit - if the initial values are too off the mark, the optimization algorithm may never find the maximum likelihood estimates. So, let's try to make a visual guess as to what the initial values should be.

Taking a look at the Hollling Type II equation, we can see that for small values of N_0 (the initial prey density), the number of prey eaten *per predator, and per unit time* (i.e. ignoring the P and t parameters) is close to a (the attack rate) and as N_0 increases, the number of prey eaten by each predator reaches an asymptote of $1/h$.

Let's plot the number of prey eaten, divided by the number of predators P and the study time t, as a function of the initial prey density:

```
ReedfrogFuncresp$killed_norm <- ReedfrogFuncresp$Killed / (3 * 14)

ReedfrogFuncresp %>%
  ggplot(aes(Initial, killed_norm)) + geom_point() +
  labs(x = "initial prey count", y = expression("prey killed ("~predator^{-1}~day^{-1}~")"))
```



The number of prey killed doesn't seem to have reached an asymptote at an initial prey density of 100, which makes it a little difficult to guess an initial value here - but I'd go with an asymptote of maybe 1.2, so we'll use an initial value of $h = 1/1.2 = 0.83$).

Since a controls the slope at low predator densities, we can make a better guess for a by fitting a linear regression on smaller values of the initial population size, holding the intercept at 0 (since predators can't consume prey that aren't there):

```
initial_lm <- lm(
  killed_norm ~ 0 + Initial,
  data = ReedfrogFuncresp %>% filter(Initial < 20)
)

coef(initial_lm)
```

```
##    Initial
## 0.01394558
```

Maximum likelihood estimates

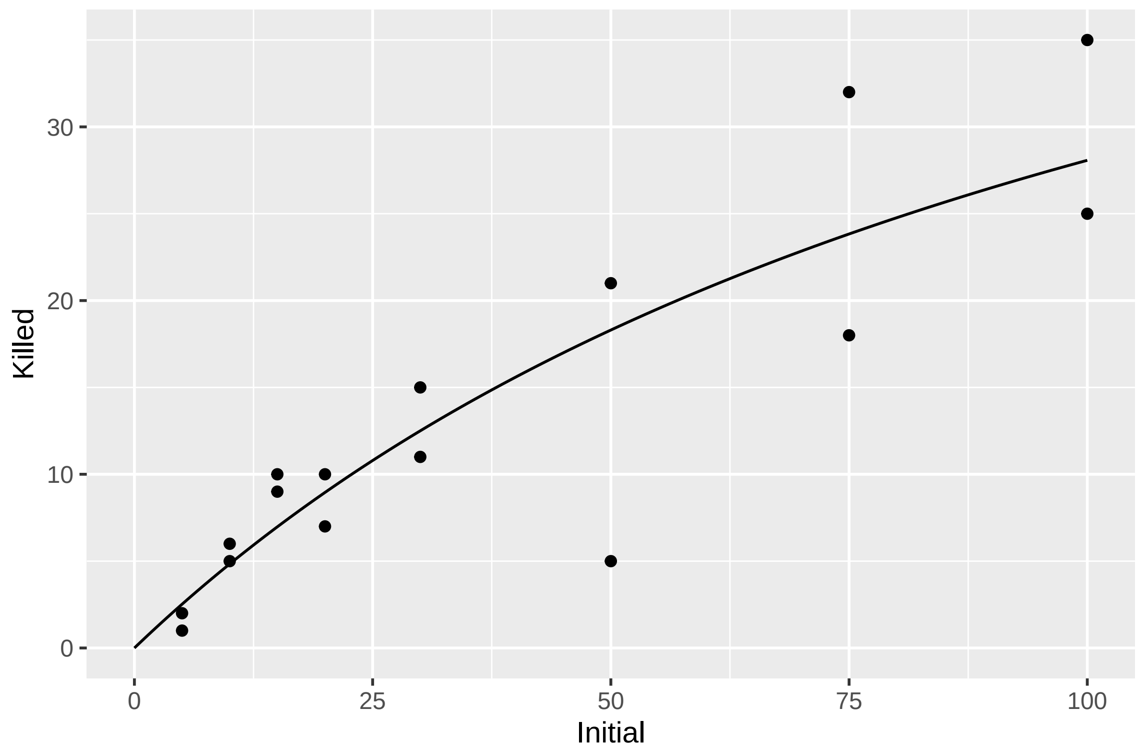
So, our rough initial values are $a = 0.014$ and $h = 0.85$. Let's pass these to `optim`, along with our `nll_holling2` function and our data, to obtain maximum-likelihood estimates for the attack rate and handling time:

```r
holling_mle <- optim(
  par = c(log_a = log(0.014), log_h = log(0.85)),
  fn = nll_holling2,
  N0 = ReedfrogFuncresp$Initial,
  Nc = ReedfrogFuncresp$Killed,
  P = 3,
  t = 14,
  hessian = TRUE
)

# Back-transform estimates
setNames(exp(holling_mle$par), c("a", "h"))
```

```
##          a          h
## 0.01252105 0.69735582
```

Let's superimpose the fitted curve on top of our data:

```r
## obtain fitted values
fitted <- data.frame(initial = seq(0, 100, 0.1)) ## initial prey density
fitted$holling2 <- holling2(fitted$initial, exp(holling_mle$par[1]), exp(holling_mle$par[2]), P = 3, t =

ReedfrogFuncresp %>%
  ggplot(aes(Initial, Killed)) +
  geom_point() +
  geom_line(aes(x = initial, y = holling2), data = fitted) +
  labs(color = "type")
```



We can obtain standard errors for the (log) attack rate and handling time with Fisher information:

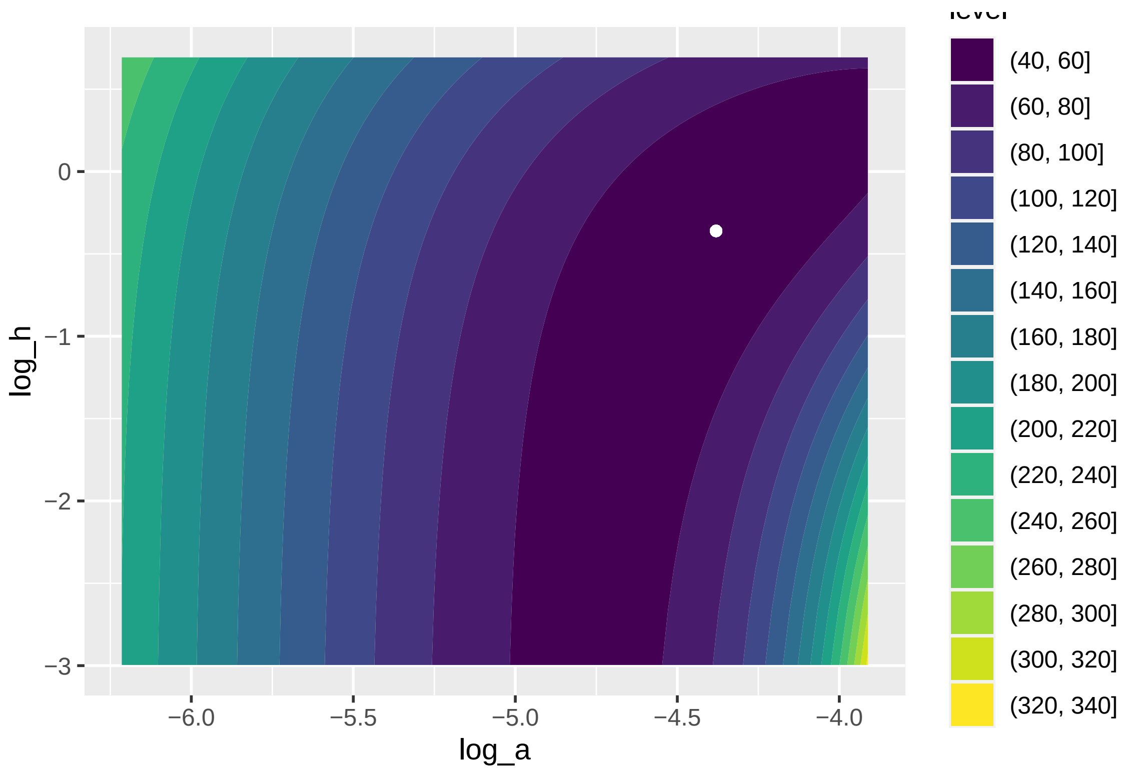
```
var_cov <- solve(holling_mle$hessian)
se <- sqrt(diag(var_cov))
se
```

```
##     log_a     log_h
## 0.1351236 0.2939936
```

Likelihood Surface

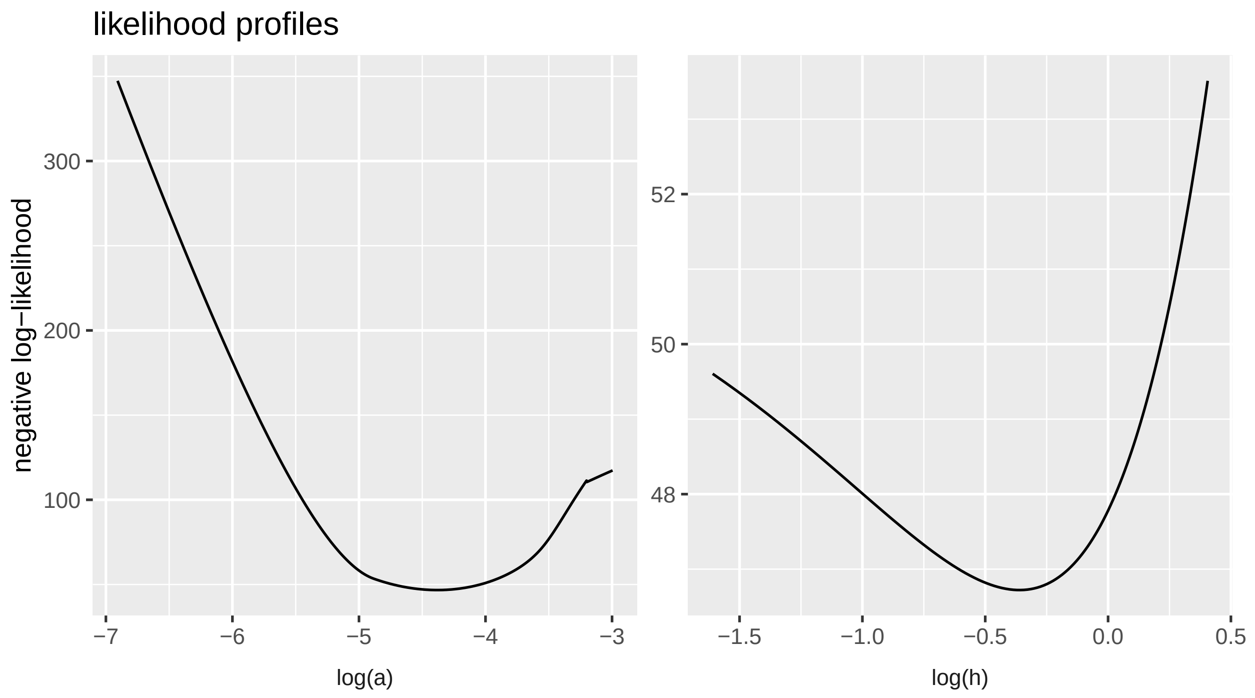
We only have two parameters in this model, so let's visualize the dependency of our likelihood function on the parameters as a **surface**. Fill in the for loop below so that each combination of parameters has a corresponding negative log-likelihood value, and plot the likelihood surface with the parameters / axes on the log scale:

```
a_seq <- seq(0.002, 0.02, length.out = 100)
h_seq <- seq(0.05, 2, length.out = 100)

par_grid <- expand.grid(log_a = log(a_seq), log_h = log(h_seq))

for (i in 1:nrow(par_grid)) {
  par_grid$nll[i] <- nll_holling2(
    par = unlist(par_grid[i,1:2]),
    N0 = ReedfrogFuncresp$Initial,
    Nc = ReedfrogFuncresp$Killed,
    P = 3,
    t = 14
  )
}

par_grid %>%
  ggplot(aes(log_a, log_h, z = nll)) +
  geom_contour_filled() +
  geom_point(aes(x = holling_mle$par[1], y = holling_mle$par[2]), color = "white")
```



Question: Based on the shape of the likelihood surface, do you think a normal approximation for the confidence intervals on a and h would be appropriate? Why or why not?

If a normal approximation were appropriate for both of these paramters, we'd expect the likelihood contours to form ellipses around the maximum likelihood estimate. Because this likelihood surface is more complex than that, it's clear that the likelihood does not drop off symmetrically around the maximum likelihood estimate, which suggests the distribution of the maximum likelihood estimate may not be symmetric either. By plotting the likelihood profiles, we can see that in fact the likelihood is symmetric around the handling time parameter, but is very much not symmetric around the attack rate parameter. Judging from the likelihood surface and profiles, it seems unlikely that a normal approximation would work well. It's also worth noting that the likelihood changes much more quickly as we vary the attack rate than as we vary the handling time - which mirrors the fact that the standard error for the handling time is much larger than the standard error for the attack rate.



Confidence intervals

For this example, let's use the bootstrap to obtain confidence intervals for a and h, and to obtain confidence intervals for the response variable (the number of prey consumed). Since there aren't a ton of observations (16) and the observations are experimental manipulations (not random samples), the bootstrap isn't necessarily the *best* choice, but we'll use it anyway (probably, likelihood profiles would be the best way to compute confidence intervals here).

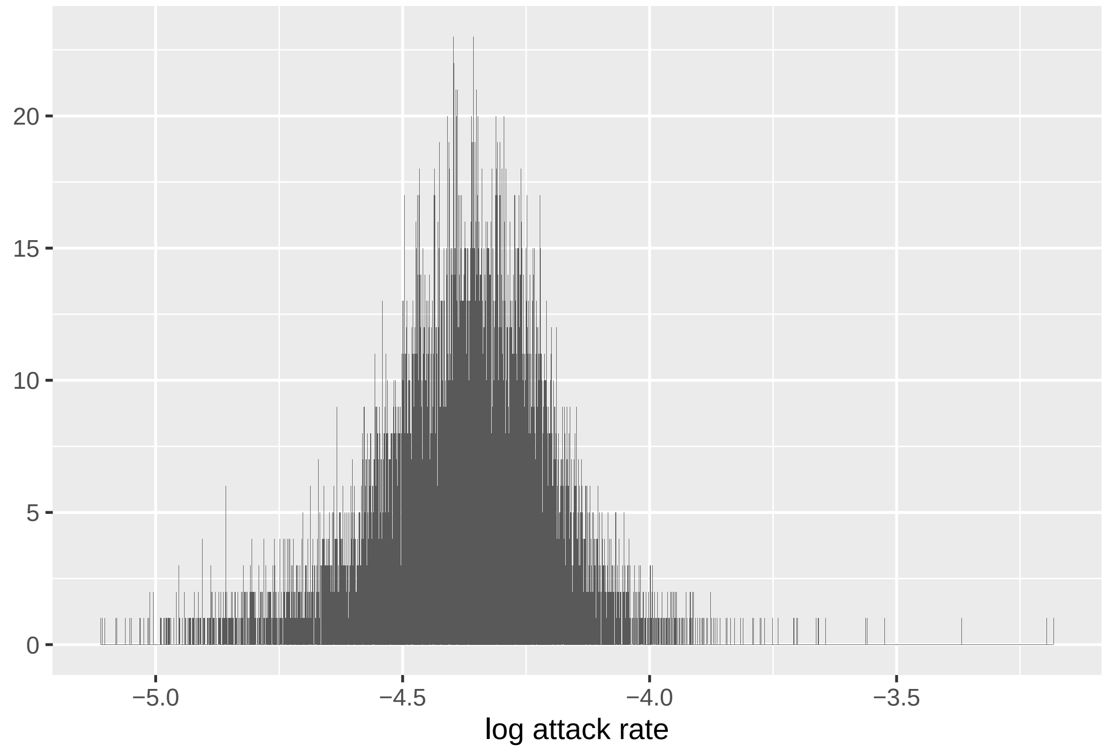
We'll take 10,000 samples of the attack rate and handling time:

```r
n_obs <- nrow(ReedfrogFuncresp) ## number of observations
n_boot <- 10000 ## number of bootstrap samples

## matrix to store attack rate and handling time
holling_boot <- matrix(NA, nrow = n_boot, ncol = 2)

for (i in 1:n_boot) {
  sample_rows <- sample(1:n_obs, n_obs, replace = TRUE) ## rows to sample
  sample_data <- ReedfrogFuncresp[sample_rows,]

  sample_mle <- optim(
    par = holling_mle$par, ## initialize at the overall MLE
    fn = nll_holling2,
    N0 = sample_data$Initial,
    Nc = sample_data$Killed,
    P = 3,
    t = 14
  )
  holling_boot[i,] <- sample_mle$par
}
```

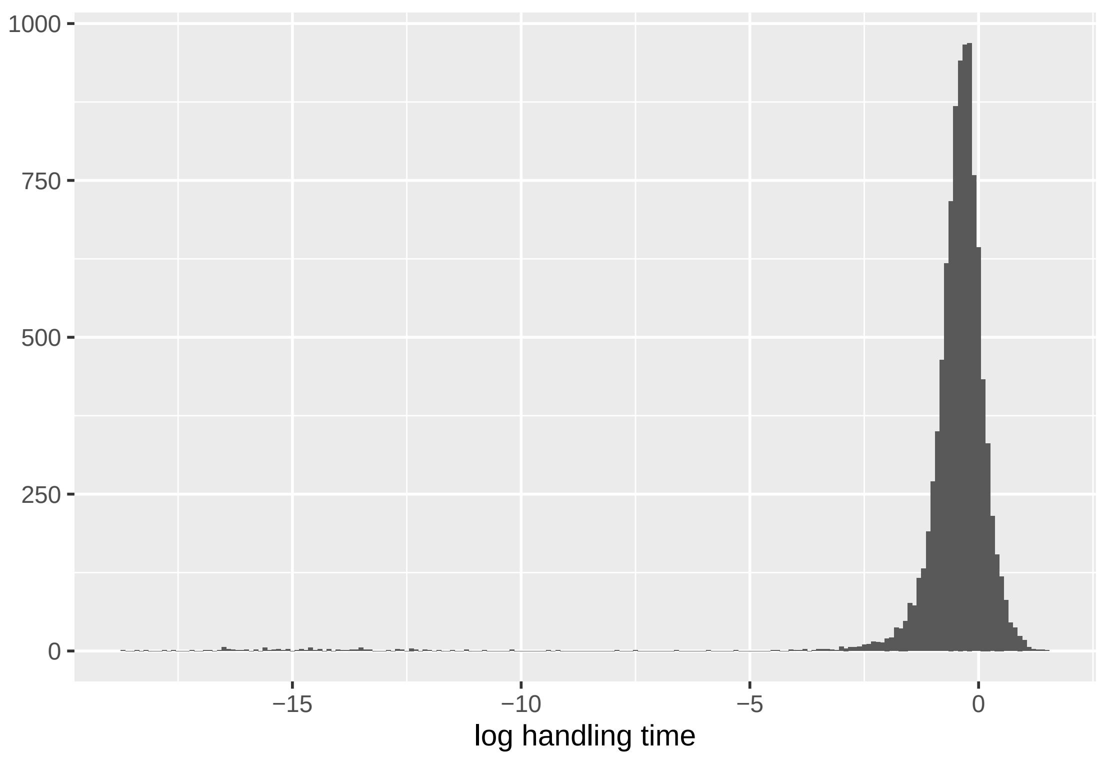
Question: Plot histograms of these samples. The attack rate samples are in the first column, and the handling time samples are in the second. What do you notice about the distributions of each of these parameters?

> **Surprisingly, the bootstrap distribution for the attack rate appears fairly symmetric (and quite narrow), while the bootstrap distribution for the handling time is fairly skewed. Judging from the bootstrap distributions alone, I'd be more inclined to trust the normal approximation for the attack rate than for the handling time.**

```
qplot(holling_boot[,1], binwidth = 0.0005) + xlab("log attack rate")
```



```
qplot(holling_boot[,2], binwidth = 0.1) + xlab("log handling time")
```



Question: Obtain 95% confidence intervals using (1) the bootstrap distribution, and (2) the MLE and Fisher standard errors. Back-transform these intervals so that the attack rate and handling time are interpretable. How do these compare?

> **The bootstrap confidence interval for the attack rate agrees pretty closely with the normal approximation using Fisher information, but the handling time confidence interval differs dramatically between the two methods - the bootstrap method produces a wider confidence interval.*

```
boot_CI <- list(
  a = exp(quantile(holling_boot[,1], c(0.025, 0.975))),
  h = exp(quantile(holling_boot[,2], c(0.025, 0.975)))
)

boot_CI
```

```
## $a
##        2.5%       97.5%
## 0.008358098 0.017270776
##
## $h
##      2.5%     97.5%
## 0.1503015 1.6858798
```

```
fisher_CI <- list(
  a = exp(qnorm(c(0.025, 0.975), mean = holling_mle$par[1], sd = se[1])),
  b = exp(qnorm(c(0.025, 0.975), mean = holling_mle$par[2], sd = se[2]))
)

fisher_CI
```

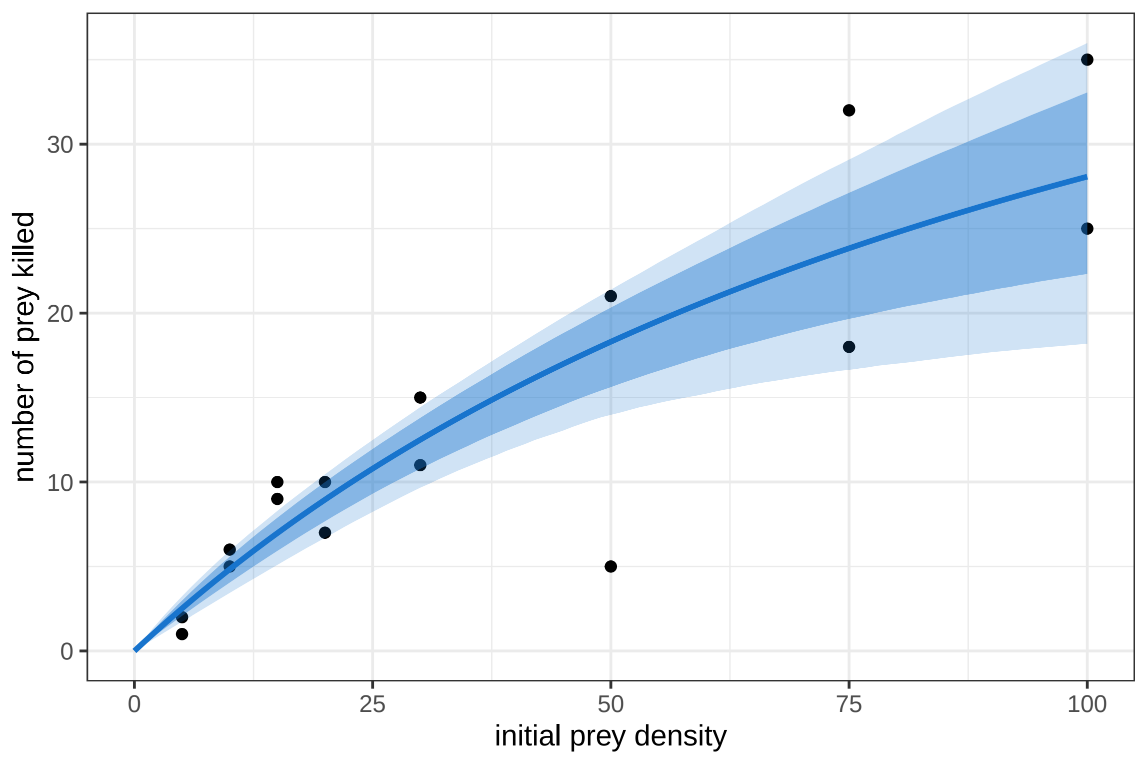
```
## $a
## [1] 0.009607786 0.016317671
##
## $b
## [1] 0.3919283 1.2408013
```

Now, let's use our samples of a and h to draw confidence bands for the number of prey eaten. To do that, we'll loop over the rows of the matrix containing our sample estimates, and for each we'll store a fitted line by applying the `holling2` function we defined earlier on to initial prey densities up to 100:

```
sample_fits <- vector("list", n_boot)

for (i in 1:n_boot) {
  sample_fits[[i]] <- data.frame(
    initial = 0:100,
    killed = holling2(0:100, a = exp(holling_boot[i,1]), h = exp(holling_boot[i,2]), P = 3, t = 14)
  )
}

sample_fits <- do.call("rbind", sample_fits)
```

Now, let's use the `quantile` function on each of the different initial prey densities to obtain 80% and 95% confidence bands and plot them:

```
holling_confint <- sample_fits %>%
  group_by(initial) %>%
  summarize(`2.5%` = quantile(killed, 0.025),
            `10%` = quantile(killed, 0.1),
            `90%` = quantile(killed, 0.9),
            `97.5%` = quantile(killed, 0.975))

ReedfrogFuncresp %>%
  ggplot(aes(Initial, Killed)) +
  geom_point() + ## points for observed data
  geom_ribbon(aes(x = initial, ymin = `2.5%`, ymax = `97.5%`), ## 95% confidence band
              alpha = 0.2, fill = "dodgerblue3", inherit.aes = FALSE,
              data = holling_confint) +
  geom_ribbon(aes(x = initial, ymin = `10%`, ymax = `90%`), ## 80% confidence band
              alpha = 0.4, fill = "dodgerblue3", inherit.aes = FALSE,
              data = holling_confint) +
  geom_line(aes(x = initial, y = holling2), color = "dodgerblue3", ## MLE
            size = 1, data = fitted) +
  labs(x = "initial prey density", y = "number of prey killed") +
  theme_bw()
```



Exercises

1. Likelihood ratio tests

Use a likelihood ratio test to assess the null hypothesis that the handling time h is equal to 1 (i.e., that it takes a dragonfly about 1 day to eat/digest a tadpole). Refer back to the first assignment for how to do this - it involves obtaining a maximum likelihood estimate for a while holding h constant.

> **To do this, I modified the negative log likelihood function so that the handling time was a separate argument, so that the attack rate is the only parameter passed as the first argument (I renamed the first argument to "a" to make this clear). `optim` will only optimize the parameters passed as the first argument, which means it will only optimize `a`, so we can use this function to hold `h` constant and obtain the value of the negative log-likelihood at the restricted maximum likelihood estimate for `a`. This is the negative log likelihood for our reduced model, and we already have the negative log likelihood for the full model (stored in the `holling_mle` variable we created earlier). With these two things, we can compute the likelihood ratio test statistic and a p-value. The p-value I got is 0.15, which means that our maximum likelihood estimate of $\hat{h} \approx 0.7$ is not significantly different from a handling time of 1. This is consistent with the fact that our confidence interval for the handling time h includes 1, regardless of the method we use to calculate it.**

```r
holling_proflik_h <- function(log_a, P, t, N0, Nc, log_h) {
  p_sim = pmin(holling2(N0, exp(log_a), exp(log_h), P, t)/N0, 0.999)
  -sum(dbinom(Nc, prob = p_sim, size = N0, log = TRUE))
}
```

```r
reduced_model <- optim(
  par = holling_mle$par[1],
  fn = holling_proflik_h,
  N0 = ReedfrogFuncresp$Initial,
  Nc = ReedfrogFuncresp$Killed,
  log_h = log(1), P = 3, t = 14
)

full_nll <- holling_mle$value
reduced_nll <- reduced_model$value

chisq <- 2*(reduced_nll - full_nll)

pchisq(chisq, df = 1, lower.tail = FALSE)
```

```
## [1] 0.1454904
```

2. The Rogers functional response

An alternative model, which allows for depletion of the tadpoles over the time during which the study is conducted, is the **Rogers random-predator equation**:

$$N = N_0\left(1 - e^{a(Nh-PT)}\right)$$

From Chapter 8 of Ben Bolker's book:

The Rogers random-predator equation (8.1.4) contains N on both the left- and right-hand sides of the equation; traditionally, one has had to use iterative numerical methods to compute the function (Vonesh and Bolker, 2005). However, the Lambert W function (Corless et al., 1996), which gives the solution to the equation $W(x)e^{W(x)} = x$, can be used to compute the Rogers equation efficiently: in terms of the Lambert W the Rogers equation is:

$$N = N_0 - \frac{W(ahN_0e^{-a(PT-hN_0)})}{ah}$$

The `lambertW` function from the `emdbook` package allows us to compute this version of the Roger's random predator equation. A negative log-likelihood function that uses the Rogers model is:

```r
rogers <- function(N0, a, h, P, t) {
  N0 - lambertW(a * h * N0 * exp(-a*(P * t - h * N0)))/(a * h)
}

nll_rogers <- function(par, P, t, N0, Nc) {

  a <- exp(par["log_a"]); h <- exp(par["log_h"])
  p_sim = pmin(rogers(N0, a, h, P, t)/N0, 0.999)
  -sum(dbinom(Nc, prob = p_sim, size = N0, log = TRUE))

}
```

Obtain maximum-likelihood estimates for a and h under the Rogers model (Use the maximum-likelihood estimates for a and h from the Holling fit as starting parameters). Are the parameter estimates different? Why do you think they are / are not?

The attack rate and handling time here are adjusted for the fact that the number of tadpoles is being depleted throughout the 14 days of the experiment. As the tadpoles become depleted, the attack rate of the predator decreases (imagine climbing down the functional response curve backwards, starting at high prey densities and going down), so we know that the estimates we get without accounting for depletion are probably biased downward. These estimates are higher because they correct that bias.

```r
rogers_mle <- optim(
  par = holling_mle$par,
  fn = nll_rogers,
  P = 3, t = 14,
  N0 = ReedfrogFuncresp$Initial,
  Nc = ReedfrogFuncresp$Killed
)

rogers_mle
```

```
## $par
##      log_a      log_h
## -4.0763077 -0.2118634
##
## $value
## [1] 46.86519
##
## $counts
## function gradient
##       47       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

We can use the Akaike Information Criterion (AIC) to compare these models. We prefer the model with the lowest AIC, but only if that model has an AIC value at least two units smaller than the other models (i.e., if two models are within 2 AIC of each other, neither one is considered to fit the data substantially better). The equation for AIC is:

$$AIC = 2k - 2\ln\mathcal{L}(\hat{\theta})$$

where k is the number of parameters in the model (2, for both models), and $-2\ln\mathcal{L}(\hat{\theta})$ is twice the value of the negative log likelihood at the MLE. Compute the AIC for both models (the negative log likelihood at the MLE is returned from `optim` under the `$value` object).

```r
AIC <- function(opt) 2*length(opt$par) + 2*opt$value

c(holling = AIC(holling_mle), rogers = AIC(rogers_mle))
```

```
##  holling   rogers
## 97.44271 97.73037
```

Does AIC prefer either model? Based on AIC and what you know of the study, which model would you prefer?

> **AIC does not show clear support for either model, and in fact, the Rogers model actually fits slightly worse according to AIC ($\Delta AIC \approx 0.3$). Since they're both very close, and we have a strong reason to suspect that the Holling model is biased because it does not account for the process of depletion which is occurring across our 14 day study, I would choose the Rogers model. The warning that this is giving us, however, is that sometimes we can't use the data to distinguish between two competing models. These data show the same support for both models, so we have to use our knowledge of the study system to choose a superior model. And, the model that best describes the functional response in this study may not even be one of the ones we've considered - it's very possible that if we accounted for depletion across the range of the study in a different way, our parameter estimates would be different still.**