# Likelihood and all that, for disease ecologists

Ben Bolker (bolker@mcmaster.ca) and Steve Ellner and Matthew Ferrari?

May 19, 2012

**Last compiled:** Thu May 17 15:37:47 2012



## Contents



## 1 Why likelihood?

Why go to the trouble of defining likelihood etc. rather than using good old fashioned ANOVA, $t$ tests, nonparametric tests, etc. . . . ?

- **Likelihood-based approaches are flexible and can be based on mechanistic models**. Parameters are meaningful, and statistical hypotheses often correspond closely with biological hypotheses. One can often adapt likelihood models to estimate the parameters of specific, theoretically based biological models.

- **Many (most) common statistical approaches represent special cases of likelihood-based approaches**. It is a unifying framework into which most statistical techniques fit. For example, the MLE is equivalent to the least-squares fit for normal, homoscedastic, independent data.
- **Likelihood-based approaches focus on parameter and confidence interval estimation (rather than $p$ values and null hypothesis testing)**. We will see later how to estimate parameters and compute confidence intervals based on likelihood — but we can also compare models and get $p$ values of specified null hypotheses.

## 2 Definition and simple example

Definition: *probability of a given set of data having occurred, given a particular hypothesis*: $\text{Prob}(D|H) = \mathcal{L}(D, H)$[1].

Read in a data set (collected by Caro Perez-Heydrich) that we will use for running examples. For 250 gopher tortoises captured in Florida, it documents various individual characteristics (sex/reproductive status, size, site and year) and the serological status (determined via ELISA test) with respect to mycoplasmal infection.

Download the gopher tortoise data from `http://www.math.mcmaster.ca/bolker/eeid/private/gopherto` and save it in your working directory.

```
> dat <- read.table("gophertortoise.txt",header=TRUE)
```

We'll start with a basic binomial model — say our observations are $x$, the number of seropositive individuals out of a total of $N$ individuals. We'll suppose that (1) all individuals have the same *per-individual* probability of infection $p$, and (2) all individuals are seropositive (or seronegative) independently. Then the distribution of the number seropositive will be binomial:

$$x \sim \text{Binomial}(p, N) \tag{1}$$

or

$$\text{Prob}(x|p, N) = \binom{N}{x} p^x (1-p)^{N-x}. \tag{2}$$

Since the joint probability of independent events equals the product of their probabilities, you can think of this as (probability of $x$ independent "successes" (positive tests), each with probability $p$) $\times$ (probability of $N - x$ independent "failures" (negative tests) each with probability $1 - p$) — times a complicated bit at the beginning which accounts for the order of events and makes the probabilities of any possible $x$ add up to 1.0 as they should[2][3]

[1] Because it can be reasonably said to be the "likelihood of the hypothesis", you will sometimes see the notation $\text{Prob}(D|H) = \mathcal{L}(H|D)$. While this is technically consistent, I don't like it because it makes it easier to slip across the line to thinking that you are calculating $\text{Prob}(H|D)$.

[2] in practise, we can often ignore these *normalization constants* when doing likelihood analysis, for reasons to be explained later

[3] one possible source of confusion here is that we have two probabilities — one "per-trial"

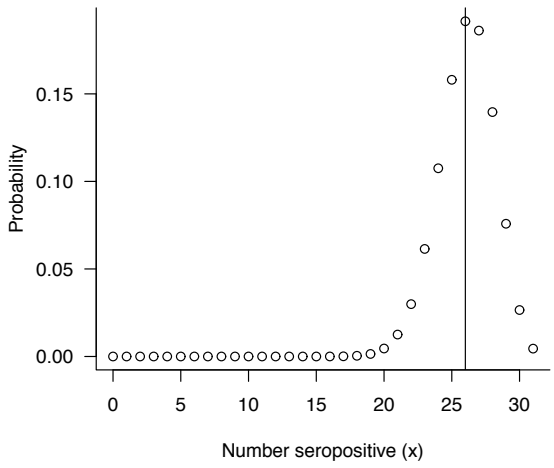
Of the $N = 31$ individuals between 230 and 240 mm carapace length, there are $x = 26$ seropositive and $N - x = 5$ seronegative individuals[4]. In R, I will define these values as $N =$`tot0` (total) and $x =$`pos0` (seropositive). To calculate the probability of this outcome for a particular infection probability $p$, use `dbinom(pos0,size=tot0,prob=p)`; for example, `dbinom(pos0,size=tot0,prob=0)` is 0 (there is no chance of getting 26 successes if the per-trial probability is zero), while `dbinom(pos0,size=tot0,prob=0.84)` is 0.191.

# 3 Probability distributions and likelihood curves

## 3.1 Probability distributions

There are two ways to examine the binomial probability model: in R, both use `dbinom()`. In the first, we can generate the *probability distribution* of possible outcomes (from 0 to 31 in this case), for a fixed $p$[5]:

```
> xvec = 0:tot0
> plot(xvec,dbinom(xvec,size=tot0,prob=0.84),
      xlab="Number seropositive (x)",ylab="Probability")
> abline(v=pos0)
```




Note that while $x = 26$ is the most likely outcome (with a probability of 0.191, getting 27 seropositive individuals is almost equally likely (0.191).

---

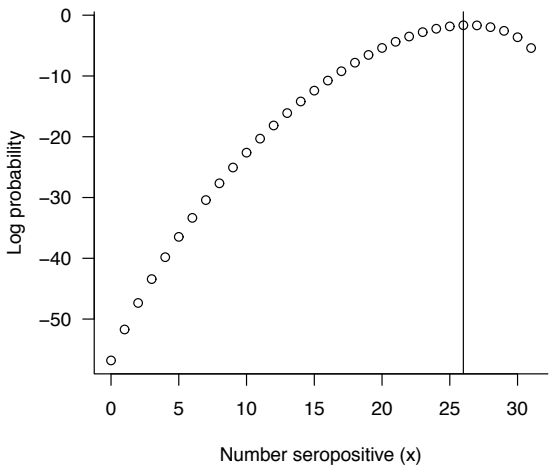
or "per-capita" probability, the probability that any *individual* is seropositive, and the other the probability of the overall data set (i.e., the likelihood).

[4] `with(subset(dat,size>=230 & size<240),table(status))`

[5] we use R's default point-based plot in this case to emphasize that only integer outcomes are possible, although we could use `type="b"` to guide our eye

Extreme outcomes (say, $x < 21$ or $x > 30$) are very unlikely indeed. We can see just how extreme by plotting the log-probability:[6]

```
> plot(xvec,dbinom(xvec,size=tot0,prob=0.84,log=TRUE),
      xlab="Number seropositive (x)",ylab="Log probability")
> abline(v=pos0)
```




(Note that R uses natural logarithms for `log(x)` and for `log=TRUE`: use `log10(x)` to get base-10 logarithms.)

Alternatively, we could

```
> plot(xvec,dbinom(xvec,size=tot0,prob=0.84),log="y")
```

which plots the probabilities on a logarithmic $y$ scale rather than calculating the log-probabilities. There is at least *some* probability of getting no seropositive individuals with these parameters, even if it is on the order of $10^{-25}$ ...

### 3.2 Probability distributions: Examples

An important question at the start of any analysis is "which probability distribution/likelihood should I use?". In some cases, the process itself suggests a distribution; i.e. the binomial distribution describes processes that can be reasonable caricatured as coin-flipping trials – e.g. the death or infection of discrete individuals.

Some special distributions result as the consequence of aggregating many arbitrary random variables. Notably, the Normal and Lognormal distribution result from the sum and product, respectively, of many randome events. The

[6] This plot is almost the same as the one we would get for `plot(xvec,log(dbinom(xvec,size=tot0,prob=0.84)))`, but slightly more accurate for small probabilities.

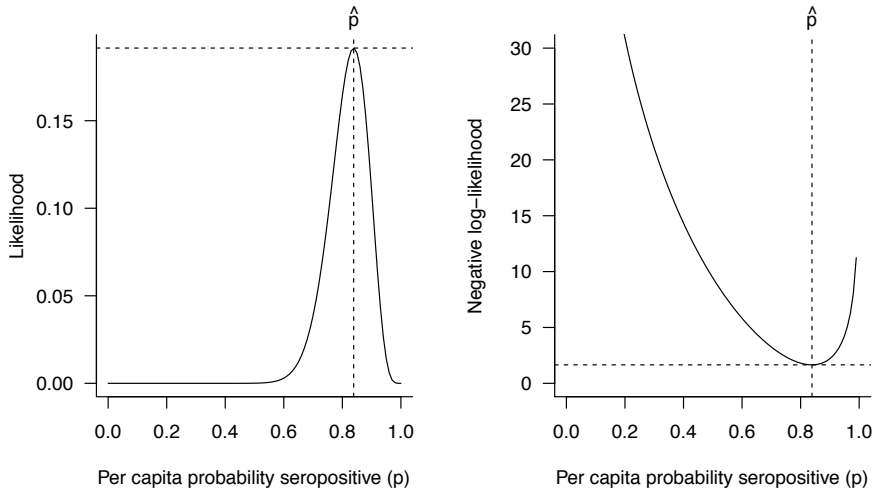
more things in the collection, the better these distributions reflect the distribution of the aggregation.

In general, we choose distributions/likelihoods because their properties (range, skewness) reflect the pattern of variation seen in the data.

| **Distribution** | **Type** | **Range** | **Skew** | **Examples** |
|---|---|---|---|---|
| Binomial | Discrete | 0,N | Any | coin flipping (# of heads), number surviving, cases reported |
| Geometric | Discrete | 0,inf | Right | coin flipping (flips until a head), discrete lifetimes |
| Poisson | Discrete | 0,inf | Right | Counts of rare events, cases per day |
| Negative Binomial | Discrete | 0,inf | Right | Counts of aggregated events, cases per day |
| Uniform | Continuous | -inf,inf | None | |
| Normal | Continuous | -inf,inf | None | Sum of arbitrary random events, size, mass |
| Exponential | Continuous | 0,inf | Right | Survival time, infectious duration |
| Gamma | Continuous | 0,inf | Right | Survival time, infectious duration |
| Beta | Continuous | 0,1 | Any | Random probabilities |
| Lognormal | Continuous | 0,inf | Right | Product of arbitrary random events, size, mass |

### 3.3 Likelihood curves

In the second approach to examining the binomial probability model, rather than using $x$ as the dependent variable, assume that $x$ is known and plot the probability as function of $p$:




- the *maximum likelihood estimate* (MLE) is the value of the parameter that makes the data most likely to have occurred. In the particular case of binomial data, we could do a little bit of calculus to show that the MLE in this case is (number seropositive)/(total), which requires either common sense or a little bit of calculus (i.e. calculate $(d \log \mathcal{L}/dp)$ and solve for $\hat{p}$ such that $(d \log \mathcal{L}/dp)(\hat{p}) = 0$).

- the *maximum likelihood* or *maximum log-likelihood* is the (log) probability of the data given that the parameter is set to the MLE. Perhaps surprisingly, this is usually not a number we care about much (but the ratios or differences between likelihoods are interesting).

- the *likelihood curve* shows the likelihood for a range of possible parameter values. We often draw the negative log-likelihood curve (which has its *minimum* in the same place, at $\hat{p}$), or the curve of $-2 \log \mathcal{L}$ (called the *deviance*), for various computational and historical reasons.

**Exercise**. Use `dbinom` to calculate that the maximum likelihood is 0.191.

## 4 From data points to data sets

If we have more than one data point (we hope so!), and *if* the data points are independent (which we usually hope, or assume ...), then according to basic probability theory the likelihood of the full data set is the product of the likelihoods for the individual points, and the (negative) log-likelihood is the sum of the (negative) log-likelihoods.

Condense the gopher tortoise data set to numbers and numbers seropositive at each size (the `ddply` function in the `plyr` package takes a data frame, splits it into chunks according a specified variable, applies a specified function to

condense each chunk, and sticks the chunks back together; in this case we are just computing the total number of individuals and the total number seropositive in each size category):

```
> library(plyr)
> sizetab <- ddply(dat,"size",
                   function(x) c(tot=nrow(x),pos=sum(x$status)))
```

(When you do this yourself you should sanity-check the results with some combination of `summary()`, `head()`, `tail()`, and `str()` — and even plot some summaries if you have time.)

Calculate the overall fraction seropositive:

```
> (prob0 <- with(sizetab,sum(pos)/sum(tot)))
```

```
[1] 0.744
```

Calculate $-\log \mathcal{L}$ for the whole dataset, if every individual had the same probability $p_0$ of seropositivity: $\sum_{a=1}^{A} \log \mathrm{Binom}(x_a|p_0, N)$:

```
> with(sizetab,-sum(dbinom(pos,size=tot,prob=prob0,log=TRUE)))
```

```
[1] 114.6869
```

(this corresponds to a tiny probability: more on this in a moment).

Define an R function to compute this value for any specified value of `prob`:

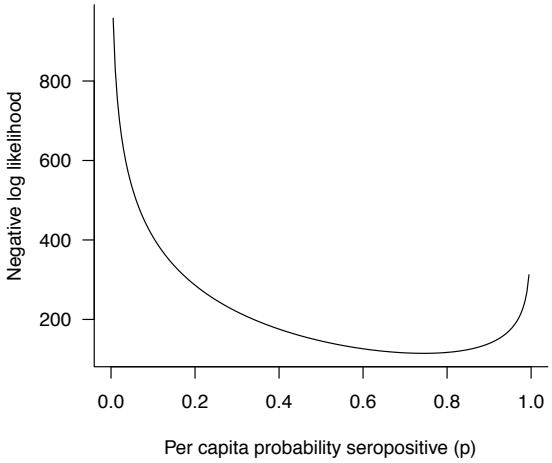
```
> NLLfun_binom0 <- function(prob,dat=sizetab) {
   with(dat,-sum(dbinom(pos,size=tot,prob=prob,log=TRUE)))
 }
```

Define a vector of probabilities from 0 to 1 and compute $-\log \mathcal{L}$ for each (using `sapply` for compactness: we could also use a `for` loop).

```
> pvec <- seq(0,1,length=201)
> NLLvec <- sapply(pvec,NLLfun_binom0)
```

Plot the likelihood curve:

```
> plot(pvec,NLLvec,type="l",
      xlab=xlabstr,ylab="Negative log likelihood")
```




If we compare this plot with the previous curve (based only on the individuals between 230 and 240 mm), we see that:

- the minimum is much higher. Since we are looking at negative log-likelihood this means that the probability of the data is much *lower*, by many orders of magnitude (113 log-likelihood units $\approx$ a factor of $10^{-50}$ (!)));the difference in height is not particularly important; with more data, the probability of any *particular* outcome will decrease (getting 26 out of 31 is more probable than getting 186 out of 250).

- the curve has its minimum (MLE) at a slightly different place. This is not surprising; the overall fraction seropositive (and hence the MLE) differs between the 230–240 mm individuals and the whole population (later on, the inferential tools that we develop will apply only to comparing different models to the same data set, not the same (or different) models to different data sets).

- The curve is much steeper. This is most important, because (as we will see) the width of the curve (inversely related to the steepness) determines the confidence interval.

In general the effect of increasing the size of a data set by a proportion $C$ (say doubling it) will be to (approximately) *multiply* the entire log-likelihood curve by $C$, which increases both the minimum negative log likelihood and the steepness of the curve by a factor of $C$.

**Exercise**. (1) use `NLLfun_binom0` to compute the likelihood curve for the 230–240 mm data (*hint #1*: construct a miniature data frame by taking the subset of `sizetab` that is includes sizes between 230 mm and 240 mm and use it in `NLLfun_binom0`. *hint #2*: use the `ylim=` argument to `plot` to make sure

the $y$-axis range is large enough to accommodate both curves. (2) subtract the minimum (`min()`) from each curve and plot the adjusted curves on the same plot to emphasize the difference in steepness rather than the difference in overall height.

## 5 MLE fitting

In this case we know that the MLE $\hat{p}$ is equal to the overall fraction seropositive, but we can also use R to do minimize the negative log-likelihood for us. We have to specify a reasonable starting guess for the probability: for complex problems this can be tricky. [7]

R has a function, `optim()`, that accesses a range of algorithms for finding the minimum of a given function. Here, we are looking to find the minimum of a function that returns the negative log-likelihood. In lay terms, a call to `optim` will make many repeated evaluations of the function to minimize until it finds the values that give the minimum. There are many possible arguments to `optim`, but the three most crucial are the starting value `par`, the function to minimize `fn`, and the algorithm to use `method`; here we use the `meth="Brent"`, which is specifically designed for optimizing a function with only one parameter (here p). `method="Brent"` additionally requires that we give lower and upper ranges over which to search for the minimum. The key outputs of `optim` are the parameter value that yields the minimum, `$par`, and the value of the function at the minimum, `$value`.

```
> m0<-optim(par=05,fn=NLLfun_binom0,method="Brent",lower=0,upper=1)
> m0

$par
[1] 0.744

$value
[1] 114.6869

$counts
function gradient
      NA       NA

$convergence
[1] 0

$message
NULL
```

[7] The best strategies for guessing reasonable starting values are (1) **know what the parameters of your model mean, and what units they are measured in** so that you can guess at reasonable orders of magnitude and (2) if possible **do some initial graphical exploration** to confirm that your starting guesses are OK. At the very least, you should input your starting guesses into your negative log-likelihood function (e.g. `NLLfun_binom0(startguess)`) to make sure that you get finite values ...

While `optim` is generic, the `bbmle` package finds the MLE if we specify the negative log-likelihood function and a starting value. It does this by making a call to `optim`[8] and formatting the outputs in useful ways (as we'll see):

```
> library(bbmle)
> (m1 <- mle2(NLLfun_binom0,start=list(prob=0.5)))

Call:
mle2(minuslogl = NLLfun_binom0, start = list(prob = 0.5))

Coefficients:
     prob
0.7439994

Log-likelihood: -114.69
```

This command produces a bunch of warning messages. **It is OK to ignore warning messages and proceed if, AND ONLY IF, you know what they mean and you are confident that the condition that is triggering them is not affecting your answers in an important way. Always stop and figure out what warning messages mean before proceeding. Ask for help if necessary!** In this case they are caused because by the minimization function trying values of `prob` that don't make sense ($\leq 0$ or $\geq 1$). It's generally OK for the function to visit bad values on the way to its final answer, so you can usually ignore this particular type of warning provided that the final fit is reasonable. To be 100% sure, you should re-do the fit in a way that eliminates the warnings (see exercise below).

For simple problems like this one, you can also use a formula interface that constructs the negative log-likelihood function automatically. This is quicker than writing a new negative log-likelihood function every time you want to change the model a little bit, and enables some additional functionality (such as calculating predicted values of the responses or simulating from the fitted model, and making parameters vary across groups in a convenient way), but is sometimes impractical for more complex models.

For example:

```
> m1F <- mle2(pos~dbinom(prob=prob,size=tot),
              start=list(prob=0.5),data=sizetab)
```

- the left-hand side of the formula specifies the response variable (`pos` in this case)
- the right-hand side of the formula consists of a probability distribution or density function that R knows (see `?Distributions` for a long list), *not* including the first argument (corresponding to the response) but including all the other parameters required to define the distribution. These can be

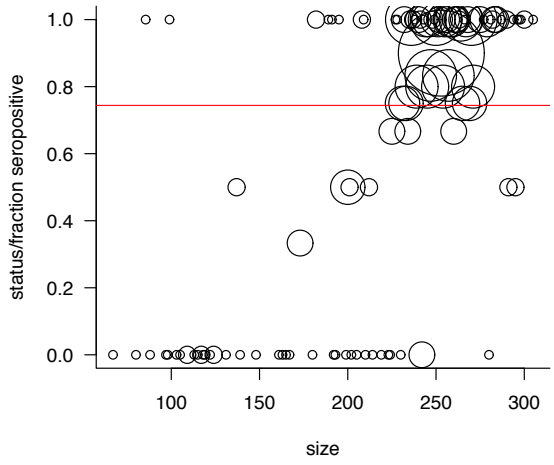
[8] Importantly, this means that you may need to look to the help files for `optim` for some issues do to with implementation of `mle2`

expressed in terms of an arbitrary expression, say `prob=a+b*size` (see examples below)

- an optional (but highly recommended) `data` argument, usually a data frame, contains the variables that R will look for in computing $-\log \mathcal{L}$
- the `start` argument must contain values for all the parameters — any variable in the formula that is not built into R or specified in `data`

This works easily, but it is actually a terrible model for the data:

```
> with(sizetab,plot(size,pos/tot,cex=tot,
                    xlab="size",ylab="status/fraction seropositive"))
> abline(h=prob0,col=2)
```




It does not capture the obvious increase in the probability of seropositivity with size.

A general point here:

**Always look at pictures of your results! (1) Likelihood surface, or profiles (if feasible); (2) Best fit of model to data (with confidence intervals on predictions, if feasible)**

**Exercise**. Redo the `mle2` fit in at least one of the following ways:

- specify a minimization algorithm that allows minimization with *box constraints* (i.e., set lower and/or upper bounds on parameters); one such example is the "L-BFGS-B" option in `mle2`. Specify `method="L-BFGS-B"` and use `lower=`, `upper=` (all as arguments — i.e. within the `mle2` call). It is often useful to set the constraints slightly within the feasible bounds —

say `lower=0.002` and `upper=0.998` — rather than setting them exactly on the boundaries.[9]

- alternatively, you can change the scale on which the parameters are estimated. For example, suppose that rather than using the probability of seropositivity as a parameter (which must be between 0 and 1), we use the *logit* of the probability — the logit, $log(p/(1-p))$ (`qlogis` in R), can range from $-\infty$ to $\infty$. The logistic, $1/(1+\exp(-x))$ (`plogis` in R), is the inverse transformation. Thus, we can use `logitprob` as our parameter and write our formula as `pos~dbinom(prob=plogis(logitprob),size=tot)`. (This approach is often easier than introducing bounds, and may have other nice statistical properties[10], but will behave badly in cases where the MLE is actually on the boundary.) The trickiest part to this is remembering that when you specify the starting value it must be sensible on the *logit* scale.

## 6 Inference I: confidence intervals

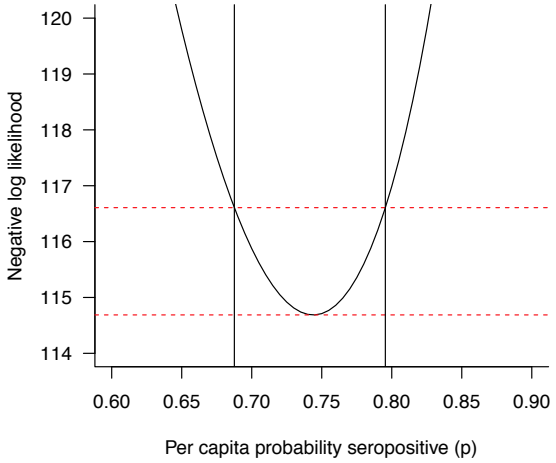
Coming back to the likelihood curve: since height on the negative log-likelihood corresponds to decreasing goodness of fit, we can define confidence intervals by picking a reference level of "badness of fit", drawing a horizontal line at this level corresponding to fits that are a particular amount worse than the best fit, and finding the intersections of the curve with that reference level.

One might pick a cutoff like 2 log-likelihood units (corresponding to fits that make the likelihood $e^2$ ($\approx 8$) times worse than the maximum likelihood); one can also derive a frequentist confidence interval based on half of the 95% upper critical value of the $\chi^2_1$ distribution, which turns out to be $\approx 1.92$. Confidence intervals derived in this way are called *profile confidence intervals*, and you can get them via `confint(m1)`. [11]

[9] Some other options are (a) use `method="Brent"` with `lower0.002` and `upper=0.998` as in the call to `optim` (b) using `optimizer="optimize"` (a specialized, derivative-free minimizer for one-dimensional (single-parameter) problems) or (c) `optimizer="optimx", method="bobyqa"` (you will need to `library("optimx")` first; this is a derivative-free method that allows constraints).

[10] The negative log-likelihood surface may be closer to quadratic (see below for why this matters) for the transformed than for the original parameters

[11] Profile confidence intervals can be difficult to compute for complex models. The *Wald confidence intervals* are an easier-to-compute approximation to profile intervals. They assume the likelihood surface is quadratic, which can sometimes be a bad assumption. You can computed them via `confint(m1,method="quad")` — they are also the basis of the $p$ values you get with `summary()`.




**Exercise**. Add horizontal lines to the (fraction seropositive vs. size) figure above corresponding to the lower and upper profile confidence intervals for $p$ (`abline` may be helpful).

In addition to computing confidence intervals, you can also use the *Likelihood Ratio Test* (LRT) to test a particular null hypothesis. Null hypotheses correspond to null values of parameters; for example, to test the null hypothesis that $p_0 = 0.5$ we can calculate the (log-)likelihood ratio statistic

$$2(\mathcal{L}(\hat{p}) - \mathcal{L}(p_0)) :$$

```
> (s <- 2*(c(logLik(m1))-(-NLLfun_binom0(0.5))))  ## test statistic
[1] 62.15793
```

(the `c` in the command above is used to throw out some extra junk that's associated with the `logLik` function; note also that we use `-NLLfun_binom0` to get the log-likelihood [the *negative* of the negative log likelihood] ...)

Next we compare it to the upper tail of the $\chi^2_1$ distribution:

```
> pchisq(s,df=1,lower.tail=FALSE)
[1] 3.169882e-15
```

We can reject this null hypothesis pretty conclusively ... this is equivalent to seeing that $p_0 = 0.5$ is well outside the 95% confidence interval for $p$.

Remember that the LRT is based only on *differences* between log-likelihoods (i.e. likelihood ratios), not on the absolute value of the log-likelihood. This means that (1) the actual value of the minimum negative log-likelihood, which increases with data set size, is irrelevant; (2) when computing the log-likelihood, you can leave out the normalization constants (which generally depend on the data and not on the parameters), *as long as you consistently exclude them for all the models you consider*.

## 7 Multi-parameter models: likelihood surfaces

Now let's make the model more realistic (i.e. allowing seropositivity to increase with size). We'll make the increase linear, but we have to do something to prevent unrealistic ($< 0$ or $> 1$) probabilities, so we will simply cut off the probabilities at 0.001 and 0.999.

We start by defining a utility function that chops values at 0.001 and 0.999:

```
> cfun <- function(x,min=0.001,max=0.999) {
   ifelse(x<min,min,ifelse(x>max,max,x))
 }
```

The other trick that is generally useful is to center the `size` variable at some reasonable value, so that our intercept is meaningful — we don't need to be predicting seropositivity for individuals of carapace length 0! Centering predictors generally improves both estimation and interpretability.

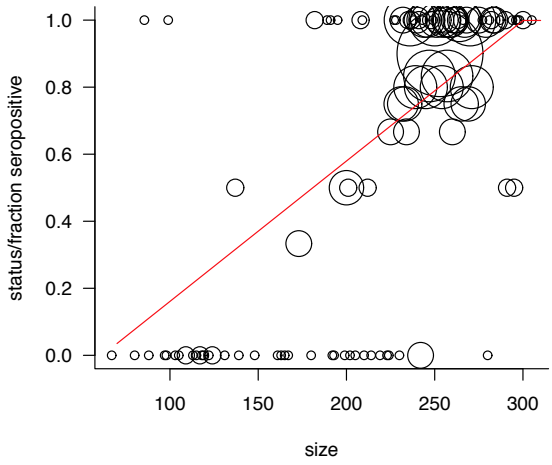
```
> (m1L <- mle2(pos~dbinom(prob=cfun(a+b*(size-200)),
                          size=tot),start=list(a=0.5,b=0.001),data=sizetab))

Call:
mle2(minuslogl = pos ~ dbinom(prob = cfun(a + b * (size - 200)),
    size = tot), start = list(a = 0.5, b = 0.001), data = sizetab)

Coefficients:
          a           b
0.579666662 0.004187887

Log-likelihood: -79.45
```

We can see that these answers are reasonable: 58% seropositivity at size=200, and an increase of about 0.4% in seropositivity per 1 mm increase (4% per 10 mm increase may be easier to think about).

We should plot the predicted graph. We could compute the predictions by hand fairly easily, but `mle2` also offers a `predict` function (but it only works with problems specified via the formula interface). We construct a new data frame specifying the variable ranges for which we want to compute predictions (we have to specify values for all of the variables used in the model, which includes `tot` in this case: we specify `tot`=1 here to get predictions for the probabilities):

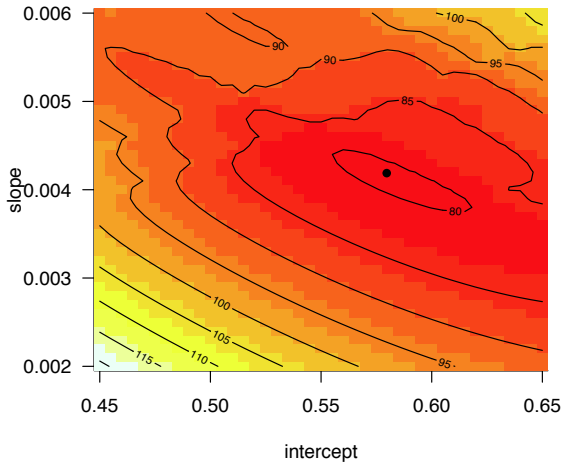
```
> pp1 <- data.frame(size=70:310,tot=1)
> pp1$pred <- predict(m1L,pp1)
```

Plotting these predictions:




Now that we have two parameters (slope and intercept), the likelihood curve becomes a likelihood surface:

We'll use the `curve3d` function in the `emdbook` package as an easy way to construct a plot of the likelihood surface. (The less-magic way to do this would be to construct a matrix; use nested `for` loops to fill in the values; and use `image` and `contour` to draw the plots.) The one other bit of magic we need, because we never defined the negative log-likelihood function explicitly, is to use `m1L@minuslogl` to pull out the function from the fitted `mle2` object. We then use `contour` (with the object returned from `curve3d`, which is a list with elements `x`, `y`, and `z`) and `points` to overlay contours and the MLE on the plot.

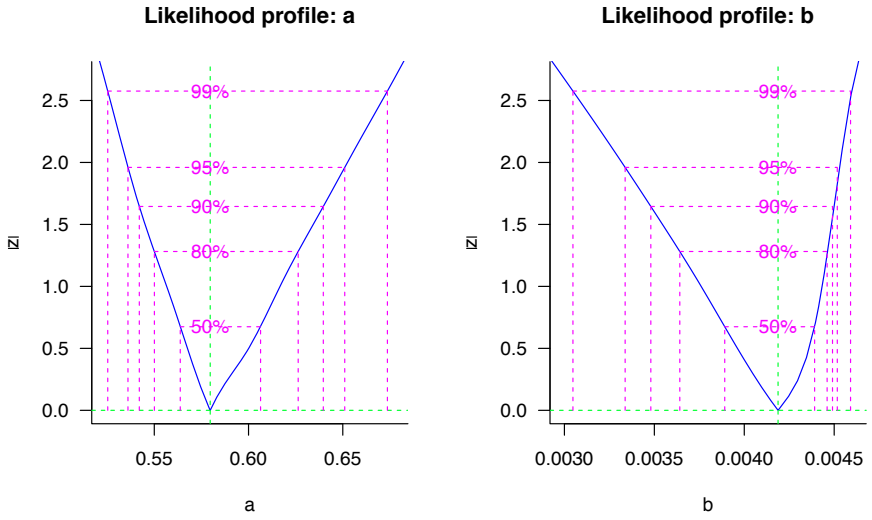



```
> library(emdbook)
> cc <- curve3d(m1L@minuslogl(x,y),
                xlim=c(0.45,0.65),ylim=c(0.002,0.006),
                sys3d="image", xlab="intercept",ylab="slope")
> contour(cc$x,cc$y,cc$z,add=TRUE)
> points(coef(m1L)[1],coef(m1L)[2],pch=16)
```

Remember that every point on this surface represents a separate fit to the data: here $x$ is the intercept, $y$ is the slope, and $z$ (the height of the surface) is the negative log-likelihood (badness of fit).

To get confidence intervals on individual parameters, we can ask R to compute (and plot) the *likelihood profile* (a sort of cross-section of the likelihood surface, too complicated to explain here). R plots the square root of the change in log-likelihood, which will be symmetrical and V-shaped if the surface is quadratic (not absolutely necessary but generally convenient):

```
> plot(profile(m1L))
```




Compute confidence intervals based on these profiles:

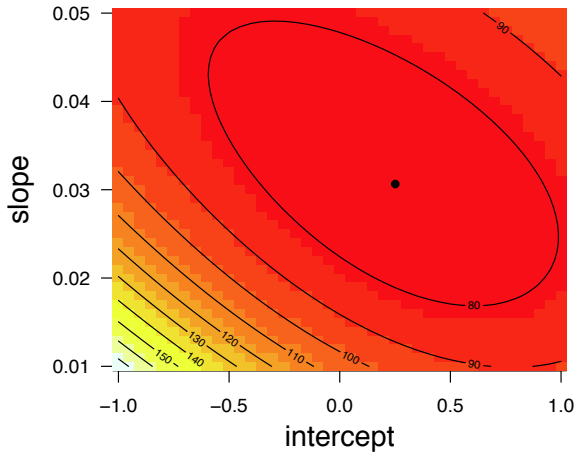
```
> confint(m1L)

Profiling...
        2.5 %      97.5 %
a 0.536052561 0.651043225
b 0.003336967 0.004540122
```

(you may get a warning here: it's OK to ignore it, especially since you've already looked at the profile to see if the answer was reasonable).

The likelihood surface above is a bit wonky because of the sharp cutoffs in our likelihood. A more standard to way to model changes in probability is as a *logistic* function, rather than a linear function — we have already seen the `plogis` function, and it is easy to switch to this alternative:

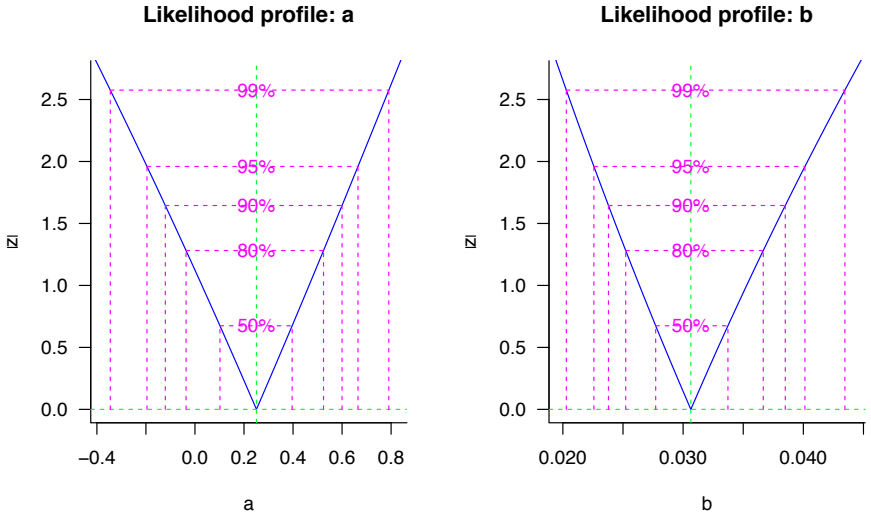
```
> m1L2 <- mle2(pos~dbinom(prob=plogis(a+b*(size-200)),
                          size=tot),start=list(a=0.5,b=0.001),data=sizetab)
```

The likelihood surface is much smoother, with elliptical contours indicating a quadratic surface:




The profile looks better too (more V-shaped, more [although not entirely] symmetrical):

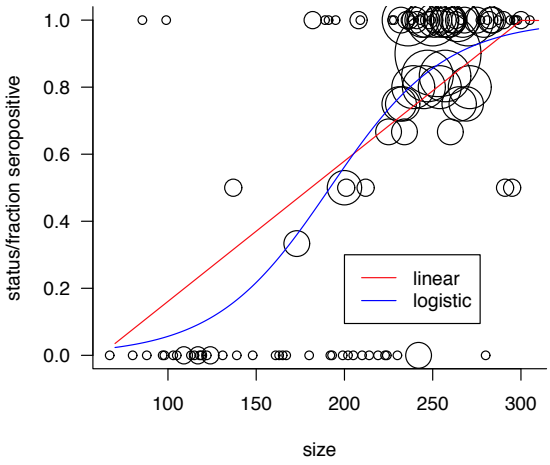
```
> plot(profile(m1L2))
```




Prediction:

```
> pp2 <- pp1
> pp2$pred <- predict(m1L2,pp2)
```

A plot of the results:




## 8 Inference II: multi-variable models

We can use the LRT to do typical sorts of hypothesis testing. In general we can do hypothesis testing for *nested* models, i.e. those where one model is a special case of the other, in which a smaller number of parameters is fitted — often called the "reduced" (simpler) and "full" (more complex) models. For example, consider the logistic model above. A sensible question (although one we really don't need statistics for) is whether seropositivity changes significantly with size. This is equivalent to asking whether $b$ is significantly different from 0, *or* whether the confidence interval of $b$ includes 0, *or* whether the logistic model fits significantly better than the constant model we fitted above.

A particularly simple way to do this comparison is using the `anova` function in R, using the two models as input[12].

```
> anova(m1L2,m1)

Likelihood Ratio Tests
Model 1: m1L2, pos~dbinom(prob=plogis(a+b*(size-200)),size=tot)
Model 2: m1, [NLLfun_binom0]: prob
  Tot Df Deviance  Chisq Df Pr(>Chisq)
1      2   147.36
2      1   229.37 82.017  1  < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

[12] why `anova`? For classical linear models, the analogue of this sort of model comparison is ANOVA. Thus the `anova` function has been co-opted for model comparison in general: analysis of variance for linear models (`lm`), analysis of deviance for generalized linear models (`glm`), and likelihood ratio tests for general maximum likelihood fitting.

If you compare nested models with more than one parameter held constant then we base inference $\chi^2_{\Delta p}$ where $\Delta p$ is the difference in the number of parameters. For example, if we were going to compare the two-parameter logistic model with a fixed seropositivity probability, e.g $p_0 = 0.5$, we would use the $\chi^2_2$ distribution (the `anova` function does this automatically).

Another way to compare models, especially non-nested models, is the AIC (Akaike Information Criterion): $-2 \log \mathcal{L} + 2k$ where $k$ is the number of parameters. We don't have time to cover this in detail, except to say that

- like negative log-likelihoods, smaller is better (even when the AIC values are negative, smaller still means "more negative")
- some general rules of thumb are that $\Delta\text{AIC} < 2$ is "nearly equivalent"; $\Delta\text{AIC} > 10$ is "extremely different"
- rather than the standard null-hypothesis testing framework of LRT (i.e. identifying deviations of a reference statistic that would be unlikely under some null hypothesis), AIC is based on estimating the model with the best expected predictive accuracy for future data sets
- like the LRT, AIC-based inference depends only on differences in AIC and not on the overall value of AIC
- R has the `AIC` function available for most models in R (including `mle2` fits, and the `AICtab` function in the `bbmle` package that provides a convenient way to compare AIC values. By default, `AICtab` reports only $\Delta$ AIC and the number of parameters (`df`).

```
> AICtab(m1,m1L,m1L2)

     dAIC df
m1L2  0.0 2
m1L  11.5 2
m1   80.0 1
```

**Exercise**. Re-do the fit with the function $p = 1 - e^{b(\text{size} - s_0)}$; wrap the whole thing in the `cfun` function defined above to prevent the probability going negative. What are reasonable starting values? Compare the fit to the other models fitted so far.

## 9 Summary of methods for `mle2` fits

| Function | Purpose |
|---|---|
| `coef(m1)` | extract coefficients (MLEs) |
| `summary(m1)` | summary information: coefficients, standard errors, Wald tests |
| `logLik(m1)` | log-likelihood |
| `deviance(m1)` | $-2 \log \mathcal{L}$ |
| `AIC(m1)` | AIC |
| `AICtab(m1,m2,...)` | AIC table |
| `stdEr(m1)` | standard errors of coefficients |
| `vcov(m1)` | variance-covariance matrix of coefficients |
| `anova(m1,m2)` | likelihood ratio test |
| `profile(m1)` | calculate likelihood profiles |
| `confint(m1)` | profile confidence intervals |
| `confint(m1,method="quad")` | Wald confidence intervals |
| * `residuals(m1)` | residuals |
| * `predict(m1)` | predicted (fitted) values |
| * `predict(m1,newdata)` | predicted values for new set of predictors |
| * `simulate(m1)` | simulated values from the fitted model |

* only available for models fitted with the formula interface