MLE basics, with pictures

Ben Bolker

June 11, 2008

1 One-parameter likelihood model: binomial

Examples of looking at likelihoods: we'll start with a basic binomial model of a single population — say that k, the number of susceptibles infected in a specified time period, out of a total of S susceptible individuals. We'll suppose that (1) all susceptible individuals have the same *per-individual* probability of infection p, and (2) all individuals are infected (or not infected) independently. Then the distribution of the number infected will be binomial:

$$k \sim \text{Binomial}(p, S) \tag{1}$$

or

$$\text{Prob}(k|p, S) = \binom{S}{n} p^k (1-p)^{S-k}. \tag{2}$$

Since the joint probability of independent events equals the product of their probabilities, you can think of this as (probability of k independent "successes" (infections), each with probability p) $\times$ (probability of $S-k$ independent "failures" (non-infections), 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 k add up to 1.0 as they should.

For concreteness, let's say that $N = 100$ and $k = 20$. To calculate the probability of this outcome for a particular infection probability p, use `dbinom(20,size=100,prob=p)`; for example, `dbinom(20,size=100,prob=0)` is 0 (there is no chance of getting 20 successes if the per-trial probability is zero), while `dbinom(20,size=100,prob=0.2)` is 0.099.

There are two important ways to use `dbinom()` to look at the distribution. In the first, we can generate a distribution of possible outcomes (from 0 to 100 in this case) and see what their probabilities are, for a fixed p:

```
> kvec = 0:100
> plot(kvec, dbinom(kvec, size = 100, prob = 0.2))
> abline(v = 20)
```

Note that while $k = 20$ is the most likely outcome (with a probability of 0.099, getting 19 infections is almost equally likely (0.098).

Extreme outcomes (say, $k < 5$ or $k > 40$) are very unlikely indeed. We can see just how extreme by plotting the log-probability:

```
> plot(kvec, dbinom(kvec, size = 100, prob = 0.2, log = TRUE))
> abline(v = 20)
```

(This is almost the same as `plot(kvec,log(dbinom(kvec,size=100,prob=0.2)))`, but slightly more accurate for small probabilities. R uses natural logarithms by default: use `log10` to get base-10 logarithms.) Alternatively, we could

```
> plot(kvec, dbinom(kvec, size = 100, prob = 0.2), 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 100% infection with these parameters, even if it is on the order of 10^{-72} ...

Most maximum likelihood procedures search for the minimum negative log-likelihood, rather than the maximum log-likelihood.

```
> plot(kvec, -dbinom(kvec, size = 100, prob = 0.2, log = TRUE))
> abline(v = 20)
```

Now turn things around and suppose we have data (i.e. $k = 20$), and want to calculate the *likelihood curve* — the probability of a fixed k for different values of p. p must be between 0 and 1:

```
> pvec = seq(0, 1, by = 0.01)
> plot(pvec, dbinom(20, size = 100, prob = pvec))
> abline(v = 0.2)
```

Now the peak is at $\hat{p} = 0.2$ (i.e., the maximum likelihood estimate is equal to the observed proportion of successes), but once again the likelihood (probability of the data) is almost as high with nearby values (0.096 for $p = 0.19$ or 0.096 for $p = 0.21$. Note also that the maximum likelihood is still pretty low: the probability of getting exactly the expected number of success (20 out of 100) is only 0.099 — precisely because you might easily get a value just below or above the expected value.

You can also take a bit of a shortcut and try

```
> curve(dbinom(20, size = 100, prob = x), from = 0, to = 1)
```

to save the trouble of setting up the vector of probabilities. (`curve()` is a "magic" function — you *must* use `x` to denote the x variable. It doesn't work very well for functions like `dbinom(x,100,0.2)` which only make sense for integer values of `x`.)

Exercises. 1. Try increasing or decreasing the number of trials (say $N = 10$ or $N = 1000$) and see what happens to the shapes of the probability density for a fixed p (`dbinom(x,prob=0.2,N)`) or the likelihood curve (`dbinom(20,prob=x,N)`). Can you explain the results?

2. `rbinom(n,prob,size)` will pick `n` binomial random numbers, and `table()` will tabulate the number of outcomes of each k. Use these two functions to generate a plot of outcomes for 10,000 binomial samples each with $p = 0.2$, $N = 100$, and compare the results to the expected distribution. (Hint: you will either have to scale the results of your "experiment" from frequencies to proportions of the outcome, or multiply the probability density by the number of samples to get frequencies.) (In general, sampling random numbers is a good way to familiarize yourself with the properties of distributions.)

Finding maximum likelihood estimates automatically, with `mle2`:

```
> minuslogl = function(p, N, k) {
+     -dbinom(k, prob = p, size = N, log = TRUE)
+ }
> library(bbmle)
> fit1 = mle2(minuslogl, start = list(p = 0.9), data = list(N = 100,
+     k = 20))
```

This leads to warnings, but the answer seems OK. To get rid of the warnings, you could use `method="L-BFGS-B"`:

```
> mle2(minuslogl, start = list(p = 0.9), method = "L-BFGS-B", lower = 0.001,
+     upper = 0.999, data = list(N = 100, k = 20))

Call:
mle2(minuslogl = minuslogl, start = list(p = 0.9), method = "L-BFGS-B",
    data = list(N = 100, k = 20), lower = 0.001, upper = 0.999)

Coefficients:
        p
0.2000012
```

```
Log-likelihood: -2.31
```

(I have to set the limits slightly in from 0 and 1, because R tries to evaluate the function *at* the limits and fails.)

Exercise: convince yourself that in this case, starting value is irrelevant (as long as it's between 0 and 1).

Exercise: try finding the quadratic and profile confidence limits (using `confint()` or `confint(...,method="quad")`).

Challenge: Use `as.data.frame(profile(fit1))` to extract a data frame from the profile (you may need to get the latest version of `bbmle` from the development site: `install.packages("bbmle",repos="http://r-forge.r-project.org")`). Then draw a graph to compare the quadratic approximation to the actual profile. (The `z` value in the data frame is the *signed square root* of the difference from the minimum deviance; to get the negative log-likelihood, you will need to calculate $z^2/2$.)

2 Two-parameter likelihood model: binomial

Now let's make things slightly more complicated and suppose that we have multiple binomial samples (all with $N = 50$) along a transect. The x variables range from 0 to 1, and the y values (numbers of infected individuals?) range from 0 to 50.

```
> bdat <- data.frame(x = c(0.006, 0.056, 0.081, 0.138, 0.182, 0.267,
+     0.289, 0.363, 0.413, 0.419, 0.427, 0.43, 0.44, 0.443, 0.765,
+     0.775, 0.862, 0.881, 0.888, 0.986), y = c(0, 0, 1, 0, 0,
+     1, 1, 0, 4, 1, 2, 5, 5, 2, 31, 32, 42, 38, 38, 48))
> plot(y ~ x, data = bdat)
```

We'll use a *logistic function* to describe the trend in the infection probability — $p = 1/(1+e^{b*(x-a)})$. You can code this directly in R (`1/(1+exp(b*(x-a)))`), or you can use `plogis(b*(x-a))`. (A slightly more standard parameterization would be `plogis(a+b*x)`, but our version is easier because it separates the slope and intercept terms — we can change the rate at which the probability goes from 0 to 1 (b) and the point at which it equals 0.5 (a) independently.

A basic likelihood function:

```
> minuslogl2 = function(a, b, N, y, x) {
+     -sum(dbinom(y, prob = plogis(b * (x - a)), size = N, log = TRUE))
+ }
```

Since we have multiple value we have to use `sum()` to combine the likelihoods (assuming as usual that they're all independent).

We need starting values for a and b. We can just see that the half-maximum is somewhere around 0.6 ($a \approx 0.6$). How about the slope? We could use

```
> plot(qlogis(y/50) ~ x, data = bdat)
```

to get a rough idea of the slope (`qlogis` is the logit transform, the inverse of the logistic function: on this scale, `qlogis(y/50)` should equal $b(x-a)$, or

```
> plot(log(y/50) ~ x, data = bdat)
```

and look at the *initial* slope ($\log(p) \propto bx$ for small x), or look at the doubling time:

$$T_{1/2} = \log(1/2)/b \approx 0.7/b \to b \approx 0.7/T_{1/2} \tag{3}$$

The doubling time is *approximately* $\Delta x = 0.2$, so we'll use $b = 3.5$ to start.

```
> fit2 = mle2(minuslogl2, start = list(a = 0.6, b = 3.5), data = c(list(N = 50),
+     bdat))
```

We use `c()` to combine the N value with the x and y data.

```
> confint(fit2)
```

```
      2.5 %     97.5 %
a 0.6968835  0.7466484
b 8.2910295 10.5198601
```

```
> plot(y ~ x, data = bdat)
> curve(50 * plogis(coef(fit2)["b"] * (x - coef(fit2)["a"])), add = TRUE)
```

Pretty nice fit!

```
> library(emdbook)
> c1 = curve3d(minuslogl2(a = x, b = y, N = 50, x = bdat$x, y = bdat$y),
+     from = c(0.65, 7), to = c(0.8, 12), sys3d = "image", col = gray((50:0)/50),
+     xlab = "a", ylab = "b")
> contour(c1$x, c1$y, c1$z, add = TRUE)
> contour(c1$x, c1$y, c1$z, level = -logLik(fit2) + qchisq(0.95,
+     2)/2, add = TRUE, col = 2, labels = "95%")
```

```
> library(ellipse)
> lines(ellipse(vcov(fit2), centre = coef(fit2)), col = "blue")
> points(coef(fit2)["a"], coef(fit2)["b"], pch = 16)
> points(0.75, 10)
```

I've added the true values of the parameters for which I simulated the data ($a = 0.75$, $b = 10$). They lie just *outside* the 95% confidence limits — this does happen sometimes.

Since we have a fairly large data set and have set up the model so that the parameters are nearly independent, the picture is very nice and elliptical.

Exercise: redraw the surface with the lower a limit at 0.65 instead of 0.5: on this scale, the outer contours are slightly avocado-shaped (to continue the fruit & vegetable analogies), but not nearly as "banana-like" as the chain binomial example (analyzed elsewhere).

Challenge: add the 99.9% confidence region and (using the `level` command to the `ellipse` function) the quadratic approximation for it to the graph. Also try to the 50% confidence region. How does the accuracy of the quadratic approximation change?