PLSC 503: "Multivariate Analysis for Political Research"

Estimation / Optimization
April 4, 2017

Optimization

Things We Won't Cover:

- Grid search / "hill climbing"
- Genetic algorithms
- Annealing methods
- Local search methods (tabu, etc.)
- many others...

Numerical Optimization Methods

First, some truth in advertising: The numerical approaches I discuss here are *only one way of getting parameter estimates that are MLEs*. Others include Bayesian approaches (e.g., Markov-Chain Monte Carlo methods), derivative-free methods (such as simulated annealing), and many others. A discussion of the relative merits of these approaches is a bit beyond what I care to go into here. To the extent that most, if not all, widely-used software programs use the numerical methods that follow, knowing something about what they are and how they work is highly recommended.

The Intuition

- Start with some guess of $\hat{\boldsymbol{\beta}}$ (call this $\hat{\boldsymbol{\beta}}_{\mathbf{0}}$),
- Then adjust that guess depending on what value of the (log-)likelihood it gives us.
- If we call this adjustment $\mathbf{A}$, we get:

$$\hat{\boldsymbol{\beta}}_{\mathbf{1}} = \hat{\boldsymbol{\beta}}_{\mathbf{0}} + \mathbf{A}_{\mathbf{0}}$$

and, more generally,

$$\hat{\boldsymbol{\beta}}_{\ell} = \hat{\boldsymbol{\beta}}_{\ell-\mathbf{1}} + \mathbf{A}_{\ell-\mathbf{1}} \tag{1}$$

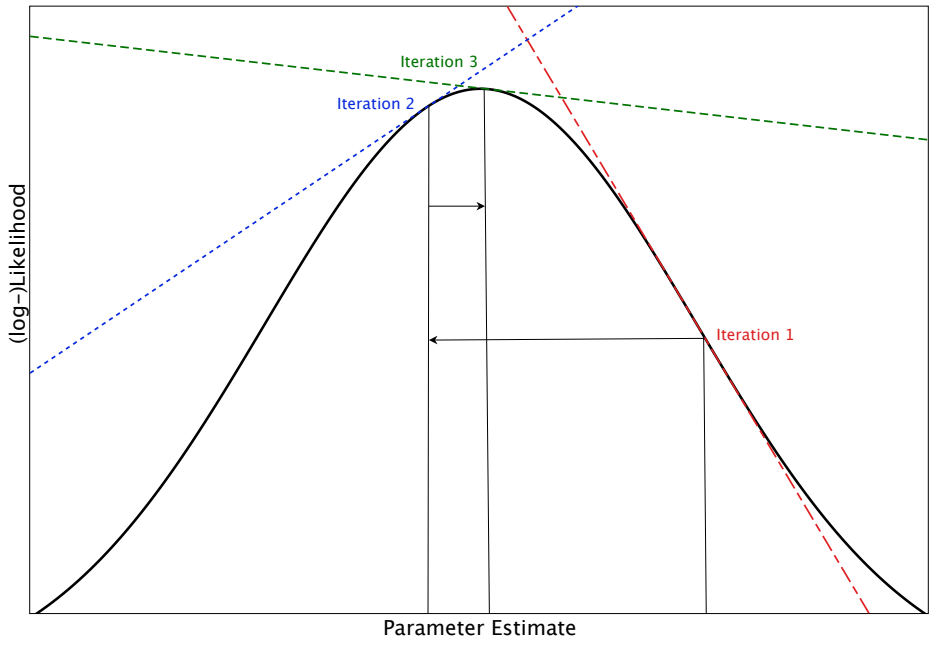
We want to move the vector $\hat{\boldsymbol{\beta}}$ to the point at which the value of the likelihood is at its highest...

- This means we need to take into account the slope of the likelihood function at each point...
- Intuitively, we do this by incorporating information from the *gradient matrix* (the matrix of first derivatives of the (log-)likelihood w.r.t. the $\boldsymbol{\beta}$s)
  - If the gradient is positive, then:
    - the likelihood is increasing in $\hat{\boldsymbol{\beta}}$, and so
    - we should increase our estimate even more...
  - If the gradient is negative, then:
    - the likelihood is decreasing in $\hat{\boldsymbol{\beta}}$, and so
    - we should decrease our estimate of $\hat{\boldsymbol{\beta}}$.

In this way, we "climb to the top" of the likelihood function...

- Once we get near the top, the gradient gets very close to zero (b/c the likelihood is near its maximum).
- At this point, when the changes get small enough from one iteration to the next, we simply stop, and evaluate our estimates.

Figure 1: MLE: Graphical Intuition



The Gradient Matrix and "Steepest Ascent"

How do we use the information from the gradient matrix?

- Again, start with some guess of $\hat{\boldsymbol{\beta}}$ (call this $\hat{\boldsymbol{\beta}}_{\mathbf{0}}$).
- Then adjust that guess depending on what value of the (log-)likelihood it gives us.

The simplest way to update the parameter estimates is to specify $\mathbf{A_k} = \frac{\partial lnL}{\partial \boldsymbol{\beta}_{\mathbf{k}}}$.

- This yields:

$$\hat{\boldsymbol{\beta}}_{\ell} = \hat{\boldsymbol{\beta}}_{\boldsymbol{\ell-1}} + \frac{\partial \mathbf{lnL}}{\partial \boldsymbol{\beta}_{\boldsymbol{\ell-1}}} \tag{2}$$

- That is, we adjust the parameter by a factor equal to the first derivative of the function w.r.t. the parameters.
- Review Q: What does a first derivative do? (A: Tells us the slope of the function at that point.)
- So:
  - If the slope of the function is positive, then the likelihood is increasing in $\hat{\boldsymbol{\beta}}$, and we should increase $\hat{\boldsymbol{\beta}}$ even more the next time around...
  - If the derivative is negative, then the likelihood is decreasing in $\hat{\boldsymbol{\beta}}$, and we should "back up" (make $\hat{\boldsymbol{\beta}}$ smaller) to increase the likelihood.
  - This general approach is called the *method of steepest ascent* (sometimes also called steepest descent).

Using the Second Derivatives

Steepest ascent seems attractive, but has a problem... the method doesn't consider *how fast the slope is changing*. You can think of the matrix $\frac{\partial lnL}{\partial \boldsymbol{\beta}_k}$ as the *direction* matrix – it tells us which direction to go in to reach the maximum. We could generalize (1), so that we changed our estimates not only in terms of their "direction," but also by a factor determining how far we changed the estimate (Greene calls this the "step size"):

$$\hat{\boldsymbol{\beta}}_{\mathbf{k}} = \hat{\boldsymbol{\beta}}_{\mathbf{k-1}} + \lambda_{\mathbf{k-1}} \boldsymbol{\Delta}_{\mathbf{k-1}} \tag{3}$$

Here, we've decomposed $\mathbf{A}$ into two parts:

- $\boldsymbol{\Delta}$ tells us the *direction* we want to take a step in, while
- λ tells us the *amount* by which we want to change our estimates (the "length" of the step size).

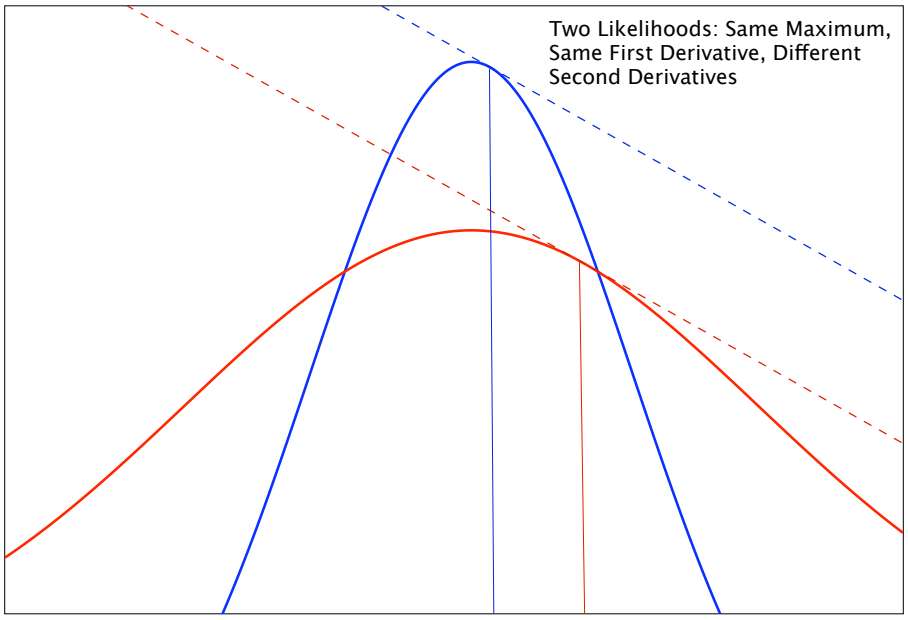
To get at this, we need to figure out a way of determining *how fast the slope of the likelihood is changing at that value of* $\hat{\boldsymbol{\beta}}$...

First, let's consider the usual approach to maximization...

- Once we've solved for the maximum (first derivative), we use the second derivative to determine if the value we get is a minimum or maximum...
- We do this, because the 2nd derivative tells us the "slope of the line tangent to the first derivative;" i.e., the *direction in which the slope of the function is changing.*
- So, if the second derivative is negative, then the slope of the first derivative is becoming less positive in the variable, indicating a maximum.
- It also tells us the *rate* at which that slope is changing:
  - E.g., if its BIG, then the slope is increasing or decreasing quickly.
  - This, in turn, tells us that the slope of the function at that point is very steep.

Now think about our likelihood example of the mean again, but compare it to another function with the same maximum...

Figure 2: Two Likelihood Functions, with Differing Rates of Change



- For red line, the function is flatter; the first derivative is negative, while the second will be quite small.
- For the blue line, the second derivative will be quite a bit larger, because the value of the likelihood is changing rapidly depending on the values of $\hat{\boldsymbol{\beta}}$.

This shows that the second derivative illustrates the speed with which the slope of the function is changing.

In likelihood terms, we want to somehow incorporate this second derivative into the maximization routine:

- If the function is relatively "steep", we don't want to change the parameters very much from one iteration to the next.
- OTOH, if it is flat, we can adjust the coefficients more from one step to the next.

To incorporate this, we can use three possibilities:

- The *Hessian*:

$$\frac{\partial^2 \ln L}{\partial \boldsymbol{\beta}^2} \tag{4}$$

  - This is the second derivative of the likelihood function with respect to the parameters.
  - This $k \times k$ matrix contains the second derivatives along the main diagonal, and the cross-partials of the elements of $\hat{\boldsymbol{\beta}}$ in the off-diagonal elements.

In some cases, figuring out this second derivative can be really hard, to the point of being impracticable. Two alternatives to this are:

- The *information matrix*:

$$-\,\mathrm{E}\left[\frac{\partial^2 \ln L}{\partial \boldsymbol{\beta}^2}\right] \tag{5}$$

  - This is the negative of the expected value of the Hessian.
  - It can be easier to compute than the Hessian itself, in some cases. (In fact, in some cases, there may be no analytical second derivative at all).
  - Yields a similar kind of result (tells how quickly the slope is changing...)

- If even the information matrix is too hard to compute, we can also use the *outer product approximation* to the information matrix:

$$\sum_{i=1}^{N} \frac{\partial \ln L_i}{\partial \boldsymbol{\beta}} \frac{\partial \ln L_i}{\partial \boldsymbol{\beta}}' \tag{6}$$

  - That is, we sum over the "squares" (outer products) of the first derivatives of the log-likelihoods.
  - This is nice because we don't have to deal with the second derivatives at all; we have only to evaluate the gradient.
  - This can be really useful when we have very complex likelihoods

Optimization Methods

Each of these options for incorporating the "rate of change" has an associated maximization algorithm associated with it...

1. **Newton-Raphson** uses the (inverse of the) Hessian:

$$\hat{\boldsymbol{\beta}}_\ell = \hat{\boldsymbol{\beta}}_{\ell-1} + \left[\left(\frac{\partial^2 \ln L}{\partial \hat{\boldsymbol{\beta}}_{\ell-1} \partial \hat{\boldsymbol{\beta}}'_{\ell-1}} \right)^{-1} \frac{\partial \ln L}{\partial \hat{\boldsymbol{\beta}}_{\ell-1}} \right] \tag{7}$$

   - I.e., the new parameter estimates are equal to the old ones, adjusted in the *direction* of the first derivative (a la steepest descent), BUT
   - The *amount* of change is inversely related to the size of the second derivative.
     - If the second derivative is big, the function is steep, and we don't want to change very much.
     - The opposite is true if the second derivative is small and the slope is relatively flat.

2. **Method of Scoring** uses the (inverse of the) information matrix:

$$\hat{\boldsymbol{\beta}}_\ell = \hat{\boldsymbol{\beta}}_{\ell-1} - \left[\mathrm{E} \left(\frac{\partial^2 \ln L}{\partial \hat{\boldsymbol{\beta}}_{\ell-1} \partial \hat{\boldsymbol{\beta}}'_{\ell-1}} \right) \right]^{-1} \frac{\partial \ln L}{\partial \hat{\boldsymbol{\beta}}_{\ell-1}} \tag{8}$$

3. **Berndt, Hall, Hall and Hausman (BHHH)** uses the inverse of the outer product approximation to the information matrix:

$$\hat{\boldsymbol{\beta}}_\ell = \hat{\boldsymbol{\beta}}_{\ell-1} - \left(\sum_{i=1}^{N} \frac{\partial \ln L_i}{\partial \boldsymbol{\beta}_{\ell-1}} \frac{\partial \ln L_i}{\partial \boldsymbol{\beta}_{\ell-1}}' \right)^{-1} \frac{\partial \ln L}{\partial \hat{\boldsymbol{\beta}}_{\ell-1}} \tag{9}$$

There are others...

- E.g., the "Davidson-Fletcher-Powell" (DFP) algorithm, in which the "step length" is chosen in such a way as to always make the updated matrix of parameter estimates positive definite.
- See Judge et al. (Appendix B) or Greene (2003, Appendix E.6) for more discussion...

MLE, Uncertainty and Inference

Since the matrix of second derivatives tells us the rate at which the slope of the function is changing (i.e., its "steepness" or "flatness" at a given point), it makes sense that we can use this information to determine the variance ("dispersion") of our estimate...

- If the likelihood is very steep,
  - We can be quite sure that the maximum we've reached is the "true" maximum.
  - I.e., the variance of our estimate of $\hat{\boldsymbol{\beta}}$ will be small.
- If the likelihood is relatively "flat," then
  - We can't be as sure of our "maximum," and...
  - so the variance around our estimate will be larger.

Maximum likelihood theory tells us that, asymptotically, the variance-covariance matrix of our estimated parameters is equal to the inverse of the negative of the information matrix:

$$\text{Var}(\hat{\boldsymbol{\beta}}) = -\left[\text{E}\left(\frac{\partial^2 \ln L}{\partial \hat{\boldsymbol{\beta}}^2}\right)\right]^{-1} \tag{10}$$

As above, we can, if need be, substitute the outer product approximation for this...

Typically, whatever "slope" (i.e., second derivative) matrix a method uses is also used for estimating the variance of the estimated parameters:

Method	"Step size" (∂^2) matrix	Variance Estimate
Newton	Inverse of the observed second derivative (Hessian)	Inverse of the negative Hessian
Scoring	Inverse of the expected value of the Hessian (information matrix)	Inverse of the negative information matrix
BHHH	Outer product approximation of the information matrix	Inverse of the outer product approximation

There are a few other alternatives as well (e.g., "robust" / "Huber-White" standard errors); we'll talk more about those a bit later on.

General comments on Numerical Optimization

- Newton works very well and quickly for simple functions with global maxima.
- Scoring and BHHH can be better alternatives when the likelihood is complex or the data are ill-conditioned (e.g. lots of collinearity).

Practical Tips

- One can occasionally converge to a local minimum or maximum, or to a *saddle point*.
- Estimates may not converge at all...
- If convergence is difficult, try the following:
  - Make sure that the model is properly- (or, at least, well-)*specified*.
  - Delete all *missing data* explicitly.
  - *Rescale* the variables so that they're all more-or-less on the same scale.
  - If possible, try another optimization *algorithm*.

Software

- Stata uses a modified version of Newton ("Quasi-Newton") for the `-ml-` routine that makes up most of its "canned" routines.
  - Requires that it have the gradient and Hessian at each iteration...
  - Uses numeric derivatives when analytic ones aren't available.
  - This can be slow, since at actually calculates the inverse Hessian at each step, BUT
  - ...it also tends to be very reliable (more so than some other packages).
  - Stata *doesn't* allow for scoring or BHHH options.
- S-Plus / R have `-glm-` routines which default to the method of scoring (or IRLS – more on that later).
- Some programs (e.g. `LIMDEP`) also allow you to start out with steepest ascent, to get "close" to a maximum, then switch to Newton or BHHH for the final iterations.

Software: R

Lots of optimizers:

- `maxLik` package: options for Newton-Raphson, BHHH, BFGS, others

- `optim` (in `stats`) – quasi-Newton, plus others
- `nlm` (in `stats`) – nonlinear minimization "using a Newton-type algorithm"
- `newton` (in `Bhat`) – Newton-Raphson solver
- `solveLP` (in `linprog`) – linear programming optimizer

R : Using `maxLik`

- *Must* provide log-likelihood function
- Can provide $\mathbf{H}(\hat{\boldsymbol{\beta}})$, $\mathbf{g}(\hat{\boldsymbol{\beta}})$, both, or neither
- Choose optimizer (Newton, BHHH, BFGS, etc.)
- Returns an object of class `maxLik`

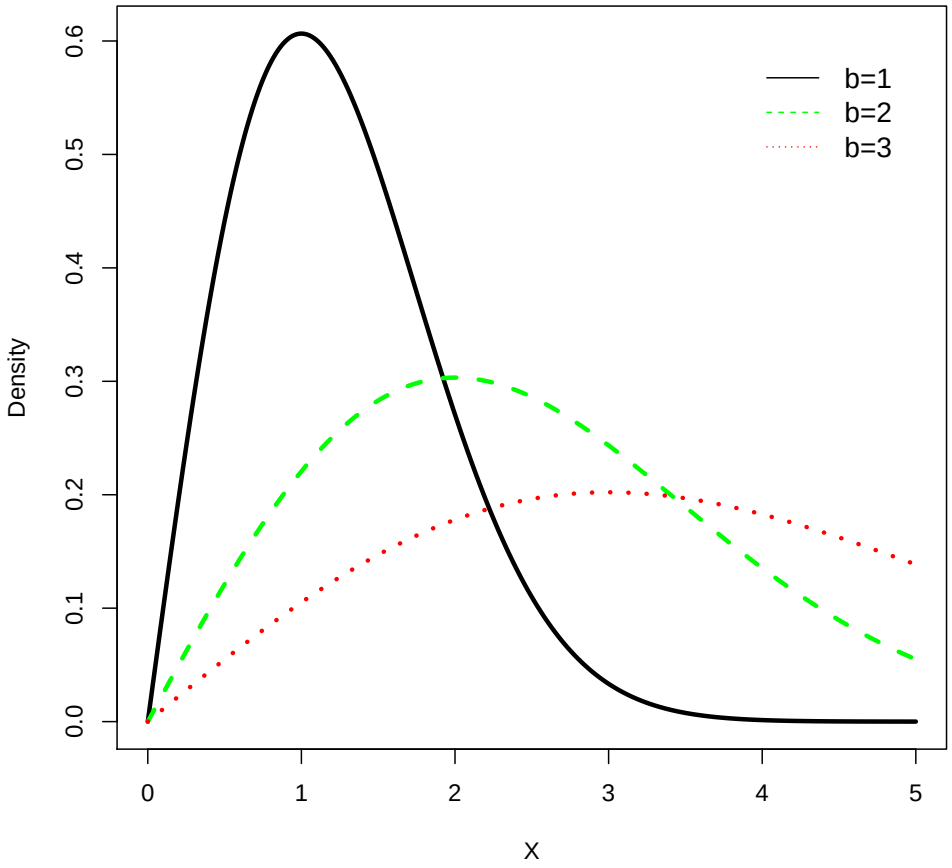
R : Example

The Rayleigh distribution:

- One-parameter distribution ($b > 0$)
- Support on $\mathcal{R}^+$
- PDF is:

$$\Pr(X) = \frac{x}{b^2} \exp\left[\frac{-x^2}{2b^2}\right]$$

- A picture:


b=1
b=2
b=3
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
1
2
3
4
5
X

R : Generate 100 draws from a Rayleigh distribution with $b = 3$

```
> library(maxLik,distr)
> set.seed(7222009)
> U<-runif(100)
> rayleigh<-3*sqrt(-2*log(1-U))
> loglike <- function(param) {
+    b <- param[1]
+    ll <- (log(x)-log(b^2)) + ((-x^2)/(2*b^2))
+    ll
+  }
```

R : What We Like To See

```
> x<-rayleigh
> hats <- maxLik(loglike, start=c(1))
> summary(hats)
--------------------------------------------
Maximum Likelihood estimation
Newton-Raphson maximisation, 8 iterations
Return code 2: successive function values within tolerance limit
Log-Likelihood: -195.7921
1  free parameters
Estimates:
     Estimate Std. error t value Pr(> t)
[1,]   2.9168     0.1459      20  <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
--------------------------------------------
```

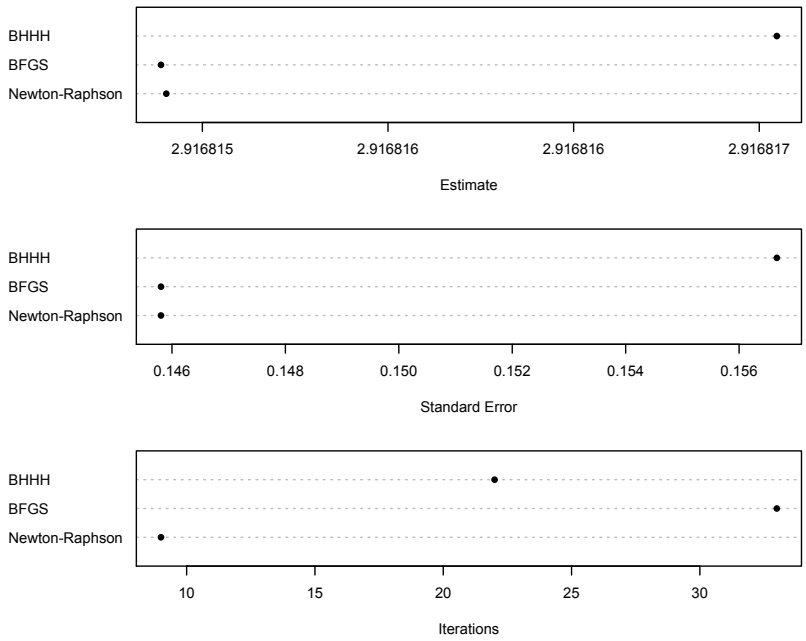
R : Comparing Different Optimizers

```
> hatsBFGS <- maxLik(loglike, start=c(1),method="BFGS") # BFGS
> hatsBHHH <- maxLik(loglike, start=c(1),method="BHHH") # BHHH
> labels<-c("Newton-Raphson","BFGS","BHHH")

hats<-c(hatsNR$estimate,hatsBFGS$estimate,hatsBHHH$estimate) # Estimates
ses<-c(sqrt(-(1/(hatsNR$hessian))),sqrt(-(1/(hatsBFGS$hessian))),sqrt(-(1/(hatsBHHH$hessian))))
its<-c(hatsNR$iterations,hatsBFGS$iterations,hatsBHHH$iterations) # Iterations

par(mfrow=c(3,1))
dotchart(hats,labels=labels,groups=NULL,pch=16,xlab="Estimate")
dotchart(ses,labels=labels,pch=16,xlab="Standard Error")
dotchart(its,labels=labels,pch=16,xlab="Iterations")
```

Different Estimates, S.E.s, and Iterations for $\hat{b}$



R : What We *Don't* Like To See

```
> Y<-c(0,0,0,0,0,1,1,1,1,1)
> X<-c(0,1,0,1,0,1,1,1,1,1) # No obs. where Y=1 and X=0
> logL <- function(param) {
+   b0<-param[1]
+   b1<-param[2]
+   ll<-Y*log(exp(b0+b1*X)/(1+exp(b0+b1*X))) +
+       (1-Y)*log(1-(exp(b0+b1*X)/(1+exp(b0+b1*X)))) # lnL for binary logit
+   ll
+ } # Logit regression function

> Bhat<-maxLik(logL,start=c(0,0))
> summary(Bhat)
--------------------------------------------
Maximum Likelihood estimation
Newton-Raphson maximisation, 9 iterations
Return code 1: gradient close to zero
Log-Likelihood: -4.187887
2  free parameters
Estimates:
     Estimate Std. error t value Pr(> t)
[1,]   -104.3        Inf       0       1
[2,]    105.2        Inf       0       1
--------------------------------------------
```

Practical Optimization...

- Potential Problems
- Likely Causes
- Tips

Problems

Enemy # 1: Noninvertable $\mathbf{H}(\hat{\boldsymbol{\beta}})$

- "Non-concavity," "non-invertability," etc.
- (Some part of) the likelihood is "flat"
- Why? (Bob Dole...)

Identification

- Possible due to functional form alone...
- "Fragile"
- Manifestation: parameter instability

Poor Conditioning

- Numerical issues
- Potentially:
  - Collinearity
  - Other weirdnesses (nonlinearities)

Potential Causes

- Bad specification!
- Missing data
- Variable scaling
- Typical $\Pr(Y)$

Hints

- T-h-i-n-k!
- Know thy data
- Keep an eye on your iteration logs...
- Don't overreach