4

# Implementing MLE

In this chapter, we discuss the methods and challenges of actually implementing likelihood methods in modern computers. Understanding the likelihood surface is a good place to start. We outline the most commonly used numerical algorithms for finding the MLE. While these algorithms often work with no problem, there can be challenges. We provide suggestions for trouble-shooting computational problems. We discuss these initially as if they were purely computational problems, but often such problems are linked with substantive modeling issues like collinearity of predictors and perfect separation in categorical data.

## 4.1 THE LIKELIHOOD SURFACE

The likelihood surface is simply the likelihood (or, more commonly, the log-likelihood) displayed as a function of parameter values. Visualizing the likelihood surface is the most basic tool for understanding whether a particular likelihood function is regular and, if not, where problems might arise in finding the MLE.

We have already examined the regular likelihood surface for the Bernoulli model in Figure 1.1. To see how visualizing the likelihood surface can highlight problems or challenges in estimation, we consider two irregular likelihoods, each relying on the uniform distribution.

**In case you were wondering ... 4.1 Uniform distribution**

Suppose $X$ is a sample from the closed interval $[a, b]$ where all values in this interval have equal probability of being drawn. We say that $X \sim \text{Unif}[a, b]$ where the distribution function, $f(x)$, is

https://doi.org/10.1017/9781316888544.007 Published online by Cambridge University Press

$$f(x) = \begin{cases} \frac{1}{b-a} & x \in [a,b] \\ 0 & \text{otherwise} \end{cases},$$

with $\mathrm{E}[X] = \frac{a+b}{2}$ and $\mathrm{var}(X) = \frac{(b-a)^2}{12}$.

### 4.1.1 Examples

***Irregular Likelihood Surface***

Let $\mathbf{x} = (x_1, \ldots, x_n)$ be $n$ independent draws from Unif$[-\theta, \theta]$. In this example the parameter, $\theta$, determines the support for the probability model, something we recognized as a violation of regularity conditions. Nevertheless, we can construct a likelihood using our standard procedure:

$$\begin{aligned} \mathcal{L}(\theta \mid \mathbf{x}) &= \prod_{i=1}^{n} \frac{1}{2\theta} \\ &= (2\theta)^{-n}. \end{aligned}$$

This likelihood assigns positive probability for any $\mathbf{x}$ such that $x_i \in [-\theta, \theta], i \in \{1, \ldots, n\}$ and 0 otherwise. Figure 4.1 plots the likelihood function for $\theta$

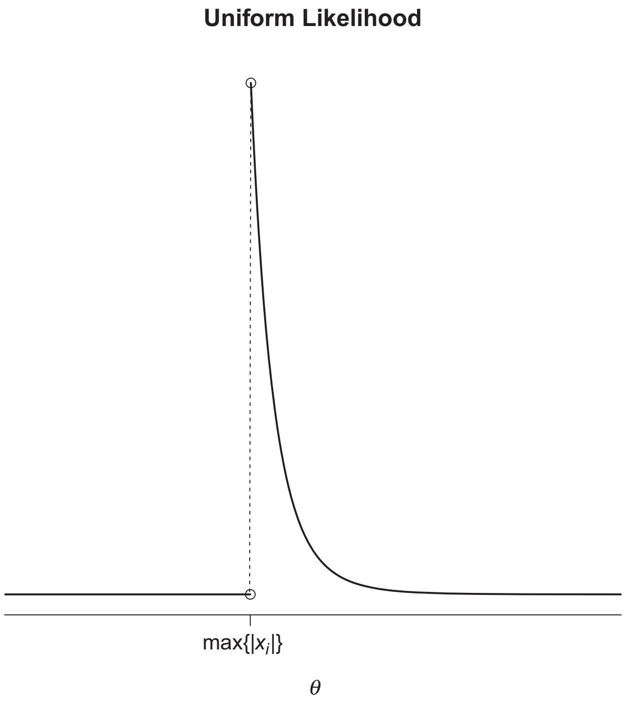

FIGURE 4.1 The likelihood surface for $\mathcal{L}(\theta \mid \mathbf{x})$ where the probability model is $X \sim \text{Unif}(-\theta, \theta)$.

given fixed $\mathbf{x}$. The likelihood surface is clearly irregular, displaying sharp discontinuities at the MLE.[1] Standard properties of the MLE will not hold in this case.

***Flat Likelihood Surface***

Suppose our data, $\mathbf{x}$, are now hypothesized to be drawn from Unif$[\theta - 2, \theta + 2]$. The likelihood is now $\mathcal{L}(\theta \mid \mathbf{x}) = 4^{-n}$, i.e., the likelihood is a constant. Figure 4.2 plots this likelihood surface for fixed and observed data, $\mathbf{x}$.

A flat likelihood implies that many values of $\theta$ are consistent with the observed data.[2] Obviously such cases are irregular, but, more problematically, flat likelihoods indicate a model where the parameter is *not identified*.

In this toy example we constructed a perfectly flat likelihood surface. But in some real-world applications, the curvature of the likelihood surface is sufficiently slight that the computer cannot tell the difference between the flat likelihood and a nearly flat one. This may happen when, for example, two covariates in a regression model are almost perfectly collinear. Most statistical programs generate an error when this happens, but some will still present

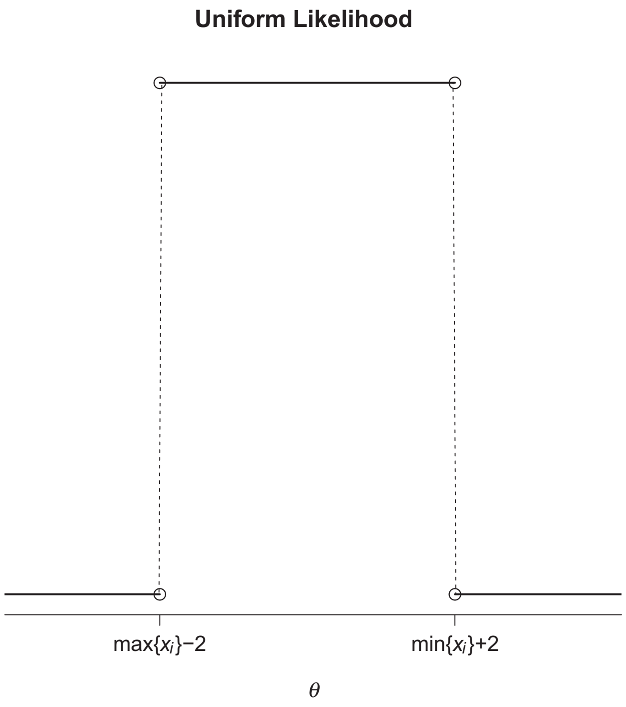

FIGURE 4.2 The likelihood surface for $\mathcal{L}(\theta \mid \mathbf{x})$ where the probability model is $X \sim \text{Unif}(\theta - 2, \theta + 2)$.

[1] The MLE is given as $\hat{\theta} = \max\{|x_1|, \ldots, |x_n|\}$.
[2] In fact the MLE is any $\theta \in [x_* - 2, x^* + 2]$ where $x_* = \min\{x_1, \ldots, x_n\}$ and $x^* = \max\{x_1, \ldots, x_n\}$.

results. Applying the theoretical results from Chapter 2 to such output can lead to erroneous conclusions.

### 4.1.2 Profile Likelihood

Likelihood functions with many parameters (i.e., most of them) become difficult to conceptualize. In some applications, we may only care about a subset of model parameters, viewing the rest as *nuisance parameters* – those that must be estimated for completeness but in which we have no interest. Even when we are interested in all the estimated parameters, it is difficult to visualize the likelihood surface for all parameters jointly. We examine the surface in one dimension at a time. To do this effectively we need to focus on the parameters' functional interdependence. *Profile* or *concentrated* likelihood is one of the most common ways of doing this.

**Definition 4.1** (Profile Likelihood). Suppose we have likelihood function $\mathcal{L}(\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is a vector. We partition $\boldsymbol{\theta}$ into $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$, where interest centers on $\boldsymbol{\theta}_1$. The *profile likelihood*, $\mathcal{L}_p(\boldsymbol{\theta}_1)$ is defined as

$$\begin{aligned}\mathcal{L}_p(\boldsymbol{\theta}_1) &\equiv \max_{\boldsymbol{\theta}_2} \mathcal{L}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)\\ &\equiv \mathcal{L}(\boldsymbol{\theta}_1, \hat{\boldsymbol{\theta}}_2(\boldsymbol{\theta}_1)).\end{aligned}$$

In other words, the profile likelihood function returns, for each value of $\boldsymbol{\theta}_1$, the maximum value of the likelihood function for the remaining parameters, $\boldsymbol{\theta}_2$. Maximizing the profile likelihood will return the MLE for $\boldsymbol{\theta}_1$. We can plot profile likelihoods, construct likelihood ratios, and even build likelihood-based confidence intervals. We have already seen a profile likelihood once: Figure 1.4 displays the profile likelihood for a regression coefficient treating the other coefficients and the variance as nuisance parameters.

## 4.2 NUMERICAL OPTIMIZATION

Notwithstanding some of the examples presented in earlier chapters, finding the MLE and taking advantage of all its various properties is almost always accomplished using a computer. In calculating the MLE, most computer programs iterate some numerical optimization procedure. These algorithms update and recalculate until the change in the current value falls below some threshold or *tolerance*. At this point the algorithm is said to have *converged*. In many of the standard models discussed in this book, finding extrema is not too difficult. But in more complicated analyses, especially using highly customized likelihoods, a unique maximum and the ability of a particular algorithm to find it is not guaranteed.

The workhorse for finding the extrema of log-likelihood functions are *hill-climbing* algorithms. Such algorithms use the information contained in the derivative of a function to "climb" to the maximum (or descend to the minimum). Although there are now many hill-climbing algorithms, most are based on the logic of Newton's method, also known as the Newton-Raphson algorithm.

### 4.2.1 Newton-Raphson

The Newton-Raphson (N-R) algorithm is a general procedure for finding the root(s) of equations of the form $g(\theta) = 0$. For scalar $\theta$, given an initial guess $\theta_0$, the algorithm updates by relying on a linear (Taylor series) expansion of the target function:

$$g(\theta) \approx g(\theta_0) + g'(\theta_0)(\theta - \theta_0) = 0$$

solving for $\theta$ yields the next step in the algorithm

$$\theta_1 = \theta_0 - \frac{g(\theta_0)}{g'(\theta_0)},$$

where $g'(\theta_0) = \frac{d}{d\theta} g(\theta)\Big|_{\theta=\theta_0}$. The algorithm stops when $\theta_n = \theta_{n-1}$.

For vector $\boldsymbol{\theta}$, recall that the MLE is the solution to the score equation, $S(\boldsymbol{\theta}) = 0$. Given an initial $\boldsymbol{\theta}_0$, the Taylor expansion is

$$\begin{aligned} S(\boldsymbol{\theta}) &\approx S(\boldsymbol{\theta}_0) - \frac{\partial}{\partial \boldsymbol{\theta}} S(\boldsymbol{\theta}_0)(\boldsymbol{\theta} - \boldsymbol{\theta}_0) = 0 \\ &\approx S(\boldsymbol{\theta}_0) - H(\boldsymbol{\theta}_0)(\boldsymbol{\theta} - \boldsymbol{\theta}_0) = 0. \end{aligned}$$

The updated value, $\boldsymbol{\theta}_1$, is therefore

$$\boldsymbol{\theta}_1 = \boldsymbol{\theta}_0 - H(\boldsymbol{\theta}_0)^{-1} S(\boldsymbol{\theta}_0).$$

Intuitively, the N-R algorithm climbs to the MLE by moving in the direction indicated by the gradient, with the step size weighted by the curvature of the likelihood surface at our current point. The algorithm will iterate until $|\boldsymbol{\theta}_n - \boldsymbol{\theta}_{n-1}| < \varepsilon$, where $\varepsilon$ sets the tolerance.

#### *Stabilization, Fisher Scoring, & Iteratively (Re)weighted Least Squares (IWLS)*

Newton-Raphson will converge so long as the Hessian matrix is invertible. When $\boldsymbol{\theta}_0$ starts near $\hat{\boldsymbol{\theta}}$, this typically happens quickly and with few problems. But if our starting values are far away from where we want to end up, then we may encounter issues. Given a starting value, $\boldsymbol{\theta}_0$, the next step in the Newton-Raphson algorithm is determined by $S(\boldsymbol{\theta}_0)$. If we are far from the top of the hill, then $|S(\boldsymbol{\theta}_0)|$ could be quite large, leading to a large step. This step could be sufficiently large in magnitude that the estimate jumps over the MLE. One way to stabilize convergence is by adjusting the size of the step by some factor, $\delta \in (0, 1]$. The algorithm is now:

$$\boldsymbol{\theta}_1 = \boldsymbol{\theta}_0 - \delta H(\boldsymbol{\theta}_0)^{-1} S(\boldsymbol{\theta}_0).$$

Other stabilization procedures can be used in place of or in addition to adjusting the step size. Most software packages allow users to adjust the step size in addition to supplying different starting values and setting the tolerance.

Given a regular likelihood there should be no problem inverting the Hessian matrix, but in some situations $H(\boldsymbol{\theta}_0)$ may turn out to be negative or difficult to invert, especially if we are far from the MLE. Replacing the observed Fisher information, $-H(\boldsymbol{\theta}) = I(\boldsymbol{\theta})$, in the updating step with the expected Fisher information, $\mathcal{I}(\boldsymbol{\theta})$, can solve this problem, in addition to speeding up convergence. Optimizers that replace observed with expected Fisher information are said to follow *Fisher scoring*. In the context of the Generalized Linear Model (see Chapter 7), where observed and expected Fisher information are the same, Fisher scoring is automatic. The default optimizer for $\mathcal{R}$'s `glm` function, Iteratively (re)Weighted Least Squares (IWLS), uses the Fisher scoring approach. Standard summary output from `glm` reports the number of Fisher scoring (i.e., N-R) iterations.

### 4.2.2 Other Algorithms

The N-R and related methods have the notable drawback of requiring recalculation of the first and second derivatives of the log-likelihood at every iteration. Moreover, N-R is only useful for problems where the likelihood is regular in the neighborhood of the MLE. Derivative-based hill-climbing approach may not be available for nonstandard problems. "Quasi-Newton" algorithms such as BFGS (implemented in $\mathcal{R}$ as part of the `optim` function) approximate the Hessian using information in the gradient. This can accelerate convergence.

Complex or highly-sculpted likelihoods may have several local optima that can trap hill-climbing algorithms. A variety of other algorithms are available. Among the most interesting (and computationally intensive) are *simulated annealing* and *Markov Chain Monte Carlo-MLE* (MCMC-MLE). While they differ in many important ways, both involve procedures that iteratively and stochastically propose new values of $\boldsymbol{\theta}$. The new value is probabilistically accepted based on a particular rule and information about the current value of the likelihood function. Both methods intentionally allow the algorithm to sample points in the parameter space that are "worse" (i.e., of lower likelihood) than the previous iteration. While this makes the algorithms more computationally expensive it also allows them to jump across the parameter space and reduces the chances of becoming stuck at a local maximum or saddle point.

### 4.2.3 Expectation Maximization (EM)

In a famous paper – cited over 46,000 times as of this writing – Dempster et al. (1977) develop a general optimization algorithm with application to a variety of more complicated likelihood-based estimation problems, including missing data (Chapter 12) and mixture models. This algorithm goes by the name of *Expectation Maximization* (EM). EM appears frequently in custom and nonstandard estimation problems, so we outline the basic logic of EM here.

Suppose we are interested in a model that can be expressed as $\mathcal{L}(\boldsymbol{\theta} \mid \mathbf{X}_c)$, but we are missing some information: the *complete data*, $\mathbf{X}_c$, is only partially observed. We define the *observed data* as $\mathbf{X}_o \subset \mathbf{X}_c$. This might happen because some covariates were not measured or reported for some of the units in $\mathbf{X}_c$. We call $\log \mathcal{L}(\boldsymbol{\theta}; \mathbf{X}_o)$ the *observed data log-likelihood*, while $\log \mathcal{L}(\boldsymbol{\theta}; \mathbf{X}_c)$ is the *complete data log-likelihood*. Starting from an initial guess, $\boldsymbol{\theta}_0$, EM arrives at $\hat{\boldsymbol{\theta}} = \arg\max \log \mathcal{L}(\boldsymbol{\theta}; \mathbf{X}_c)$ by iterating the following steps:

E-step Given the observed data and a particular iteration, $\boldsymbol{\theta}_n$, find the expected value of the complete data log-likelihood,

$$Q(\boldsymbol{\theta} \mid \mathbf{X}_o, \boldsymbol{\theta}_n) \equiv \mathrm{E}\left[\log \mathcal{L}(\boldsymbol{\theta}; \mathbf{X}_c) \mid \mathbf{X}_o, \boldsymbol{\theta}_n\right].$$

M-step Find the maximizer of the expected complete data log-likelihood, given the observed data and $\boldsymbol{\theta}_n$,

$$\boldsymbol{\theta}_{n+1} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}).$$

EM's "trick" is to use the value of $\boldsymbol{\theta}$ at the current iteration, combined with the likelihood function, to fill in the missing values in the observed data. EM's tremendous popularity is testament to its flexibility and the ubiquity of missing data problems. EM has drawbacks, however: It can be slow to converge; it does not readily produce uncertainty estimates around $\hat{\boldsymbol{\theta}}$; and it can get trapped at local maxima.

**In case you were wondering ... 4.2 Mixture distribution/mixture model**

A *mixture distribution* is a distribution or mass function that is composed of multiple subpopulations described by different probability distributions. For example, let $g_j(x; \boldsymbol{\theta}_j)$ be a distribution function with parameter vector $\boldsymbol{\theta}_j$ and assume that there are $J$ total subpopulations. We can express the mixture distribution as

$$f(x; w_j, \boldsymbol{\theta}_j) = \sum_{j=1}^{J} w_j g_j(x; \boldsymbol{\theta}_j),$$

where the parameters $w_j$ are the *mixing parameters* or weights such that $\sum_{j=1}^{J} w_j = 1$. When $J$ is known (or assumed), we have a *finite mixture model*. In such a case we can construct a likelihood:

$$\mathcal{L}(w_j, \boldsymbol{\theta}_j \mid \mathbf{x}) = \prod_{i=1}^{n} \left[ \sum_{j=1}^{J} w_j g_j(x_i; \boldsymbol{\theta}_j) \right].$$

Treating the $w_j$ as "missing data," we can use EM to maximize the likelihood (although there are other tools as well).

Mixture distributions and models are widely applied in categorization and machine learning problems, notably text analysis.

## 4.3 ESTIMATION CHALLENGES

Real-world applications of likelihood-based models can encounter a variety of challenges in estimation. We describe some of the most common and offer suggestions for troubleshooting.

### 4.3.1 Optimizer Fails to Converge

Numerical optimization techniques can run into problems depending on the data, model estimated, and optimizing algorithm. When the algorithm fails to converge it can be a simple computational issue or a symptom of a bigger problem. Here we highlight the most common problems and offer some guidance toward solutions, should you encounter them.

### 4.3.2 Starting Values

Most numerical optimizers need jumping off points in the form of starting values for the parameters. Unfortunately, if you had ideal starting values, you might not need the algorithm at all. Practically, most algorithms will start pretty well if least squares estimates are used to generate a set of starting values, but some problems may be too complex for this simple solution. If you are totally in the dark, you might try a grid search or sample from a variety of starting values to ensure that the solution is not dependent on where the algorithm starts.

### 4.3.3 Algorithm Choice

The choice of optimizer seems esoteric, but it might actually be important in some problems. There are two major types of algorithms: those that allow you to specify the first and second derivatives, and those that try to figure them out without your intervention. The former generally are much faster because they have more information to deal with and can straightforwardly work out where the maxima (minima) will be located. If you can supply only first-order conditions, then the BHHH algorithm is a good choice; BFGS and DFP are good algorithms when you don't have good starting values and don't know the formulas for the Hessians. One strategy is to start with the coarse algorithms, and use the results from these to serially move to better and better algorithms, carrying along the final estimates from one run as starting values for the next, and also supplying the calculated Hessian from the preceding run into the call for the current run.

### 4.3.4 Scaling of Variables

The scale of the variables should not matter in principle, but it can matter for numerical calculations, and especially for minimization (maximization) problems. If you have variables with vastly different scales (e.g., GDP in dollars and population in billions of individuals), it is quite plausible that coefficients for some variables will disappear into machine rounding during iterative fitting. As a result, the first- and second-order conditions will deliver incorrect or misleading information to the optimizing function, since for some changes in parameter estimates it will appear that the log likelihood will not change. This is because the actual change is being masked by the scale of an estimated coefficient. In general, variables should be within about three orders of magnitude. Standardizing variables to have a mean of 0 and a standard deviation of 1 can be particularly helpful in more complicated models with covariates on vastly different original scales. Gelman et al. (2008) suggest scaling of variables in logistic regression to have a mean of 0 and a standard deviation of 0.5; see also Gelman (2008).

### 4.3.5 Step Sizes

Most optimizing algorithms internally calculate the size of the change for the next iteration, but if the step is too big the optimizer can get stuck in a loop, continually skipping over the maximum (minimum), or go in the wrong direction, whether off to $\pm\infty$ or simply to a lower "hill" (and the wrong answer). Fortunately, most computer programs allow you the flexibility of specifying the step size. Shrinking the step size is one possible solution to this problem, albeit at the cost of more computational cycles.

### 4.3.6 Flat Likelihoods

A "flat likelihood" is generally a failure of model specification, implying that there is no unique solution to the maximization problem. This is most commonly a problem of perfect collinearity in covariates (which some programs identify and drop) or complete separation (see Section 4.5.1), but it could also be a failure to program a completely identified statistical model, implying that different combinations of values for covariates are equally likely, given the data. Flat likelihoods also arise in more complex likelihood functions, especially ones with multiple modes. The flat area of the likelihood could trap the optimizer even when there is a unique and identifiable maximum. The best solution is to plot the likelihood surface in a variety of directions, if at all possible. From here it may make sense to give the optimizer different starting values (away from the plateau) or increase the step size (so the optimizer can jump off the flat area).

### 4.3.7 Absence of Single Maximum

Similar to flat likelihoods, some likelihood surfaces are multimodal. Multimodality can trap the optimizer at a local optimum when a superior alternative exists. Multimodality is again more common in complicated, highly sculpted likelihoods. It can be hard to diagnose without visualizing the likelihood in some way. Solutions involve combinations of those given above: give the optimizer several sets of starting values, increase the step size, and try different algorithms.

## 4.4 DIFFERENT SOFTWARE YIELDS "DIFFERENT" RESULTS

Suppose you are interested in replicating a statistical finding reported in published research. The author has been kind enough to post or share her data with you. She may have even shared her STATA .do file. You estimate the model in $\mathcal{R}$ and the table of numbers displayed differs from what you see in the published result. What gives? Is STATA wrong? Is there a coding error in $\mathcal{R}$? Both? Neither?

Very often the answer is "neither." Different results from different software packages can be the result of different parameterizations for the same model. For example, STATA's `ologit` command, which estimates the ordered logit model discussed in Chapter 8, uses a parameterization which omits the constant (intercept), whereas the commonly used $\mathcal{R}$ functions do estimate an intercept. The models are mathematically identical and will yield the same interpretations on the scale of the response variable – another reason to focus interpretation there rather than on complicated transformations of specific coefficients. Similarly, some software packages may make different default decisions about how to treat categorical variables (e.g., which level is the "reference category"). Looking into the documentation is the only way to sort out how this works.

If serious differences remain across statistical platforms, then the next place to look is at the default settings for dealing with missing data, perfect separation, and similar issues. Some software packages will fit models with little or no warning that there is a problem. For example, the Rauchhaus (2009) example later in this chapter was initially fit in STATA. STATA, unlike $\mathcal{R}$, automatically drops covariates for which there is perfect separation. Furthermore, Rauchhaus's model was fit using STATA's `xtgee` function; this function does not generate a warning that perfect separation was detected, leading to material differences in estimates across computing platforms.

A third place to look for the explanation is in the default choice of optimization algorithm, especially for more complicated or customized likelihood problems. Different optimizers with different default tolerances, etc., can generate different answers. Whether this is an indication of a problem or simple rounding differences requires further investigation by the analyst. If different optimization algorithms are giving wildly different answers, then this is a symptom of problems described above (multiple modes, etc.). A closer inspection of the likelihood surface is in order.

## 4.5 MODELING CHALLENGES APPEAR AS ESTIMATION PROBLEMS

### 4.5.1 (Near-)Perfect Separation

Perfect separation occurs when some covariate in a model (or, less commonly, linear combination of covariates) perfectly predicts an outcome, or so nearly so that the computer can't tell the difference. Perfect separation occurs most commonly in models for categorical data (such as the binary logit), but it illustrates a more general issue. This problem is easiest to see when considering a single covariate, $X$, that can take on two values, "high" or "low." In our data there are no observed successes when $X$ is "high." Knowing that observation $i$ has $x_i = \text{high}$ is sufficient to tell us that $y_i = 1$.

Perfect separation can also happen with covariates that are continuous if there is some threshold that cleanly divides all the 1s and 0s. For example, if we are using age to predict whether someone voted in the last US presidential election (which was about a year ago at the time of publication), then (almost?) all survey respondents reporting an age less than 19 will not have voted. The consequence of perfect separation is that the optimizer will never converge, instead attempting to estimate an infinite coefficient (and standard error).

A recent example received some attention in the literature on nuclear deterrence and stability. Rauchhaus (2009) looks at the incidence of conflict at the country dyad-year level, finding that if both countries in the dyad are nuclear-capable, then the risk of escalation to war is substantially lower than if only one or neither of the countries is armed with nuclear weapons. Bell and Miller (2015) criticize this article on the basis of its handling of perfect separation, among other issues.

We fit the Rauchhaus (2009) model in $\mathcal{R}$ as a standard logit using `glm`, with $\mathcal{R}$ issuing the following warning:

```
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
```

This is a generic warning, indicating that the optimizer called by `glm()` is spinning off towards $\pm\infty$, forcing the predicted probability to one of the boundaries. This happens most commonly under situations of perfect separation. In particular, this means there is probably no unique set of parameters that maximizes the likelihood. If a model with a single covariate, for example, gives perfect separation with an estimate $\hat{\beta} = 0.3$, that model will also give the same perfect separation with $\hat{\beta} = 0.3 \times 100$.

The results presented in Table 4.1 give an additional hint as to the problem: the `twonukedyad` variable, representing an indicator variable for dyads in which both countries are nuclear-capable, has an enormous standard error relative to the estimated coefficient. But this standard error is not infinite.

Table 4.2 looks at this variable more directly. The contingency table shows that being a nuclear dyad is a perfect predictor of the absence of war. Zorn (2005) refers to such patterns as "quasi-complete separation" since `twonukedyad` is only a perfect predictor of nonwar.

What is to be done here? The most common solution – the one built in to STATA – is to simply drop the variable on which there is perfect separation. Many have noted that this is clearly suboptimal, since the offending variable

TABLE 4.1 *Estimation of the Rauchhaus (2009) logit model. Note the enormous standard error for* `twonukedyad`.

| | $\hat{\beta}$ | $\sigma_{\hat{\beta}}$ | *t*-Ratio | *p*-Value |
|---|---|---|---|---|
| Constant | −3.85 | 1.13 | −3.42 | 0.00 |
| onenukedyad | 0.91 | 0.37 | 2.47 | 0.01 |
| twonukedyad | −13.28 | 522.99 | −0.03 | 0.98 |
| logCapabilityRatio | −0.65 | 0.12 | −5.23 | 0.00 |
| Ally | −0.44 | 0.35 | −1.25 | 0.21 |
| SmlDemocracy | −0.07 | 0.03 | −2.35 | 0.02 |
| SmlDependence | −119.86 | 48.97 | −2.45 | 0.01 |
| logDistance | −0.69 | 0.13 | −5.19 | 0.00 |
| Contiguity | 2.95 | 0.38 | 7.69 | 0.00 |
| MajorPower | 2.36 | 0.39 | 5.97 | 0.00 |
| NIGOs | −0.03 | 0.01 | −2.52 | 0.01 |
| *n* | 455,619 | | | |
| AIC | 932 | | | |

TABLE 4.2 *Contingency table showing the incidence of war between dyad-years of different levels of nuclear capacity from Rauchhaus (2009).*

| | < 2-nuke dyad | 2-nuke dyad |
|---|---|---|
| no war | 610,402 | 806 |
| war | 102 | 0 |

is, in essence, *too good* of a predictor. One estimation strategy is the so-called Firth logistic regression (Firth, 1993), which is a penalized likelihood approach. Bell and Miller (2015) use Firth regression and show that Rauchhaus's finding disappears.

### 4.5.2 Rare Events (and Small Samples)

The example above also serves to illustrate another challenge in model fitting for categorical data, the so-called *rare events* problem, in which the frequency of observations in a particular category is dwarfed by the number of trials. The problem is most extensively studied in the context of binary data where the number of 1s is quite small relative to $n$. In the conflict data just considered, there were 102 dyad-years of observed wars against 611,402 nonwars, or about one war for every 6,000 nonwars. Data with rare events are, ironically, seen in political science with some regularity. The number of major protests are small relative to the number of daily or yearly opportunities; the number of revolutions or regime changes is even less frequent. The number of people who choose to run for public office is small relative to the population, etc.

The consistency of the MLE is an *asymptotic* property. In small samples, the MLE, including the logit model, can be biased (Firth, 1993; McCullagh and Nelder, 1989). King and Zeng (2001) show that data with rare events are analogous to small samples, since a dataset with only 102 events gives relatively little information about event occurrence even when there are over 600,000 observed trials. But there is the issue of absolute versus relative infrequency of events. The issue of small-sample bias is one of absolute infrequency. If we had a dataset with 2,000 observed wars, we would no longer face a small-sample problem, even if that would still represent less than 0.3% of the observed dyad-years. That said, extremes in relative infrequency can also induce computational problems in some circumstances.

Rare events can be approached from several directions. One way is to alter the assumed link function, using an asymmetric one, such as the *cloglog* approach mentioned earlier or even the generalized extreme value distribution. Another approach, advocated by King and Zeng (2001), imposes a direct bias correction on the intercept. Penalized likelihood such as Firth regression can also models for binary data in the face of rare events. Finally, there are various Bayesian estimation frameworks beyond the scope of this text.

## 4.6 CONCLUSION

Estimating models using maximum likelihood is often fast and painless. But one of the strengths of the likelihood approach is its flexibility in accommodating more complicated data structures and custom-designed models. As a result, some understanding of the topography of a likelihood function and of details around numerical optimization is a good thing. This chapter introduced the basics of profile likelihood and outlined some of the more common optimization procedures.

Understanding what is happening inside your statistical software is important not just for understanding the output on your screen. Some problems that initially appear to be computational in nature are, in fact, features of the data at hand, something the should be explored rather than ignored or swept under the rug. We highlighted perfect separation and rare events as frequently encountered examples in the context of model for binary or other categorical data.

## 4.7 FURTHER READING

### Applications

Perfect separation is a common problem in political science data; recent examples include Ahlquist (2010a); Barrilleaux and Rainey (2014); and Mares (2015, ch. 9). Chalmers (2017) uses both rare events and Firth logit in modeling banks' decisions to lobby the Basel Committee.

### Past Work

Heinze and Schemper (2002) provide an applied discussion in support of Firth regression. Zorn (2005) provides an applied, political-science-focused discussion of perfect separation problems.

### Advanced Study

Pawitan (2013) provides an extended discussion of profile likelihoods and likelihood-based confidence intervals and inference.

Mebane and Sekhon (1998, 2011) develop and implement a flexible optimization algorithm that has seen some use in the numerical optimization of more complicated likelihood functions.

When looking at separation and rare events, Gelman et al. (2008) take a Bayesian approach and propose a proper but uninformative Cauchy prior over the regression coefficients. See Rainey (2016) for more on the limitations of Firth regression and the importance of priors when using Bayesian methods to address perfect separation.

Kordas (2006) uses a binary quantile regression framework for modeling unbalanced and rare events binary data. Wang and Dey (2010) discuss the use of the Generalized Extreme Value distribution for modeling rare events using a more flexible, asymmetric link function.

**Software Notes**

The $\mathcal{R}$ library `ProfileLikelihood` (Choi, 2011) calculates, plots, and constructs likelihood-based confidence intervals from profile likelihoods for specified parameters for many commonly used models.

The $\mathcal{R}$ libraries `maxLik` (Henningsen and Toomet, 2011) and `bbmle` (Bolker and R Development Core Team, 2016) provide wrapper functions and easier access to a number of $\mathcal{R}$'s numerical optimization algorithms for maximum likelihood estimation. `rgenoud` implements the Mebane and Sekhon GENOUD algorithm.

Penalized and Firth regression are implemented in `logistf` (Heinze and Ploner, 2016) and `brglm` (Kosmidis, 2017). King and Zeng's rare events logit model is implemented within the `Zelig` library (Imai et al., 2009). Gelman et al.'s Bayesian approach to separation and rare events are implemented in the `bayesglm()` function in the `arm` library (Gelman and Su, 2016).