

Teaching computing in statistical theory courses

David R. Hunter
Department of Statistics
The Pennsylvania State University
Technical Report 04-03

November 23, 2004

Abstract

Instruction in statistical computing need not be relegated to courses devoted to statistical computing. Rather, computing can be a component of nearly every statistics course. As one example, Penn State recently rewrote the syllabus for its graduate-level stochastic processes course so that it includes a substantial introduction to Markov chain Monte Carlo techniques. However, wholesale changes to course syllabi are not necessary; even theoretical courses can include considerable computational content through well-designed exercises. Such exercises should be chosen so that they force students to learn essential computing skills without impeding instruction of theory; indeed, theoretical studies are often enhanced by well-designed computing-related exercises. A series of examples used in a course on large-sample theory illustrates this thesis.

Key Words: Large-sample theory, Statistical computing

1 Introduction

The February 2004 issue of *The American Statistician* includes a special section on teaching computational statistics in which three experts — Jim Gentle, John Monahan, and Ken Lange — discuss this important topic (Gentle, 2004; Monahan, 2004; Lange, 2004). These articles led to an invited session on the same topic at the August 2004 Joint Statistical Meetings. Ken Lange was unable to make it to Toronto for the Meetings, so Jim Albert, the organizer, graciously invited me to participate in the session. This article represents an expanded version of the talk I gave in Toronto.

There is little question that computing is an essential aspect of statistics, and education in statistics must adapt to this reality. (I focus on graduate-level education in this article, though the same philosophy I write about here can be applied to undergraduate education.) Probably the most obvious way to instruct graduate students in computational statistics is by developing at least one graduate-level course in computational statistics. However, while I agree with Gentle (2004) that "...all graduate programs in statistics should offer at least one course in statistical computing..." such a course is not the topic of this article.

Instead, this article proposes and illustrates a more general philosophy of computational statistics education:

Computing topics should be sprinkled throughout the statistics curriculum. In particular, computing should be used to verify or amplify theoretical results whenever doing so requires useful computing skills.

There are several reasons, both practical and pedagogical, for this philosophy. First, such an approach kills two birds with one stone: Students learn computation as well as theoretical subjects when they are forced to use the former in pursuit of the latter. This particular advantage can be blunted if too much attention is distracted by the computing at the expense of the theory; thus, the instructor should carefully choose examples that will maintain an appropriate balance and be cautious about allowing too many computing digressions. Second, I believe in the pedagogical advantages of what Gentle (2004) calls the "just-in-time" training model, in which certain skills — computing skills, in this case — are taught only when they are needed for some other purpose. I find that students are often more receptive to new material when it is presented as a means to some greater end rather than as an end in itself. Third, students are exposed to a wider variety of perspectives on computing when they learn from several different instructors than when their computing instruction comes primarily from a single source.

Finally, the general philosophy above has an important practical advantage: It makes offering, even requiring, computing education within the statistics curriculum much easier than

it would be if computing were taught solely in computing courses. This practical advantage is illustrated well by the recent experience of the graduate faculty at Penn State University. We on the faculty decided, a couple years ago, to review our required graduate curriculum from top to bottom. One goal was to reduce slightly the number of required credits for our students so as to allow them more latitude in choosing elective courses. Although Penn State offers a graduate-level course in statistical computing, we did not require this course prior to the curriculum overhaul. And because we really wanted to *reduce* the number of required courses, we did not end up requiring this course after the overhaul either. This is not to say that the faculty did not value computing in the curriculum — exactly the opposite is true — but we faced the reality that there simply wasn't enough room in students' schedules for an entire required course in computing if we wished to be true to our original goal of increasing students' flexibility.

Nevertheless, we did insert computing into the required curriculum in several ways, as instructors of various theoretical courses maintained a commitment to present computing topics in those courses. For instance, the mathematical statistics course requires students to implement an EM algorithm, and the asymptotics course requires many computing-related exercises, specific examples of which will be discussed later in this article. Perhaps the biggest curricular change was the rewriting of the syllabus for our required course in stochastic processes so as to include roughly half a semester on Markov chain Monte Carlo methods. We did this at the expense of topics like Poisson processes and properties of the exponential distribution — not because we do not view those topics as important, but because we recognize them as less important than MCMC to most statisticians today.

Our revised stochastic processes course was taught for the first time in the spring 2004 semester by Bruce Lindsay. The fact that Bruce taught this course illustrates how the general philosophy of computational statistics education can broaden the participation of the faculty in the teaching of computational statistics. For while Bruce is a fantastic statistician, he is (or was) not an expert in MCMC. Indeed, he had to learn much in order to teach the MCMC

section of the course, which he modeled closely on the first chapter of *Markov Chain Monte Carlo in Practice* (Gilks et al., 1996). Judging from students' comments, the MCMC part of the course was very well received. Despite the amount of additional work required to modify a course in this way, I hope that Bruce's example serves as an inspiration for others who might like to add topics in statistical computing to their existing theory courses but do not feel ready to teach an entire course on the subject.

Overhauling a whole course, like we did to stochastic processes at Penn State, is not necessary to employ the general philosophy of sprinkling computing topics throughout the curriculum. The remainder of this article discusses embedding computing content into the large-sample theory course I've taught at Penn State for the last few years. An old saying goes "A picture is worth a thousand words," and I think the pedagogical equivalent would be something like "an example is worth a thousand explanations." With this in mind, each of the next three sections presents and discusses a different example of a homework exercise based on assignments I've written in the asymptotics course.

2 An inconsistent MLE

Exercise: Define $\delta \equiv \delta(\theta) = \exp\{-1/(1-\theta)^4\}$. For $\theta = .2$, generate a simple random sample of size $n = 50$ from the density function

$$f_{\theta}(x) = \theta g(x) + (1-\theta) \frac{1}{\delta} h\left(\frac{x-\theta}{\delta}\right), \quad (1)$$

where

$$\begin{aligned} g(x) &= 1/2 \quad \text{for } -1 < x < 1 \\ h(x) &= 3(1-x^2)/4 \quad \text{for } -1 < x < 1. \end{aligned}$$

Next, graph the log-likelihood function and find the MLE.

This exercise is based on an example due to Ferguson (1982) in which the maximum likelihood estimator is inconsistent. In fact, in this example, the MLE converges in probability to

1 as $n \rightarrow \infty$ no matter what the true value of θ is (students are asked to prove this fact as a separate exercise that involves no computing).

Computing knowledge is required even to begin the problem, since students must be able to generate from a mixture distribution and also to generate from the density $h(x)$. This latter task may be accomplished using a rejection method or by direct inversion of the cumulative distribution function $H(x) = (2 + 3x - x^3)/4$. Inverting the cdf to obtain

$$H^{-1}(x) = 2 \cos \left\{ \frac{4\pi}{3} + \frac{1}{3} \cos^{-1}(1 - 2x) \right\}$$

is an interesting activity in and of itself, though this is a bit too much of a digression for this course and I would typically offer the closed-form $H^{-1}(x)$ as a hint.

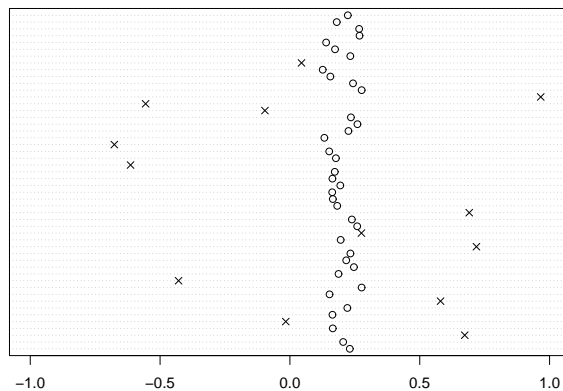


Figure 1: Random sample of size $n = 50$ from the density $f_\theta(x)$ of equation (1). The x's are uniform on $(-1, 1)$, whereas the o's are from $h[(x - \theta)/\delta]/\delta$.

Figure 1 depicts a random sample generated according to the specifications in the exercise. The points selected from the $h[(x - \theta)/\delta]/\delta$ density, marked by o's, must all fall in the range $\theta \pm \delta(\theta)$, which is $(.113, .287)$ in this case. The points selected from the uniform $(-1, 1)$ density, marked by x's, must include at least one point larger than about 0.7 in order for the exercise to work well. The probability that this fails, or $(1 - .15\theta)^n = .218$ in this case, is a bit too large for

my taste; thus, in an actual assignment, I would use a larger value of n and/or a larger value of θ so that each student is nearly assured of obtaining an interesting plot.

Once the sample is selected, plotting the loglikelihood function seems like a trivial exercise: The loglikelihood is

$$\ell(\theta) = \sum_{i=1}^{50} \log \left\{ \frac{\theta}{2} + \frac{I_i(1-\theta)}{\delta} h[(x_i - \theta)/\delta] \right\}, \quad (2)$$

where $I_i = I\{|x_i - \theta| < \delta\}$ denotes the indicator that θ lies within δ of x_i . Unfortunately, naively plotting equation (2) for $0 < \theta < 1$ results in Figure 2(a). What happened to the graph for $\theta > .8$? The answer lies in the fact that $\delta(\theta)$ is specifically constructed to be extremely small for θ near 1. Thus, $(1 - \theta)/\delta$ is extremely large — so large, in fact, that it results in an overflow, treated as $+\infty$ by the software used to create figure 2(a). Thus, even when $I_i = 0$ and the product $I_i(1 - \theta)/\delta$ should be zero, the computer doesn't know how to handle $0 \times \infty$. Diagnosing this problem correctly requires a student to think about how computer software represents numbers, and it reinforces the fact that a mathematical formula like equation (2) is not the same as a numerical algorithm.

The error of Figure 2(a) is easily corrected by the simple device shown in Figure 2(b): Instead of calculating the loglikelihood directly as in equation (2), the value of I_i is tested for each i and the term involving $(1 - \theta)/\delta$ is only calculated when $I_i = 1$; in case $I_i = 0$, the i th summand of equation (2) is replaced by $\theta/2$. Theoretically, the two methods are identical, but the second method never runs into the $0 \times \infty$ problem because $(1 - \theta)/\delta$ is never computed when $I_i = 0$.

Having thus sidestepped one thorny computational issue, we are faced with a second. The computer is unable to graph a smooth curve as truly smooth; rather, it must “connect the dots” between points that lie on the curve. By choosing these points to be very close together, the computer fools our eyes into interpreting a graph as a smooth curve. Ordinarily, there is no harm in this artifice. However, in this example, the true loglikelihood has features that are so incredibly narrow that they are missed by connecting even the most fine-meshed grid of points.

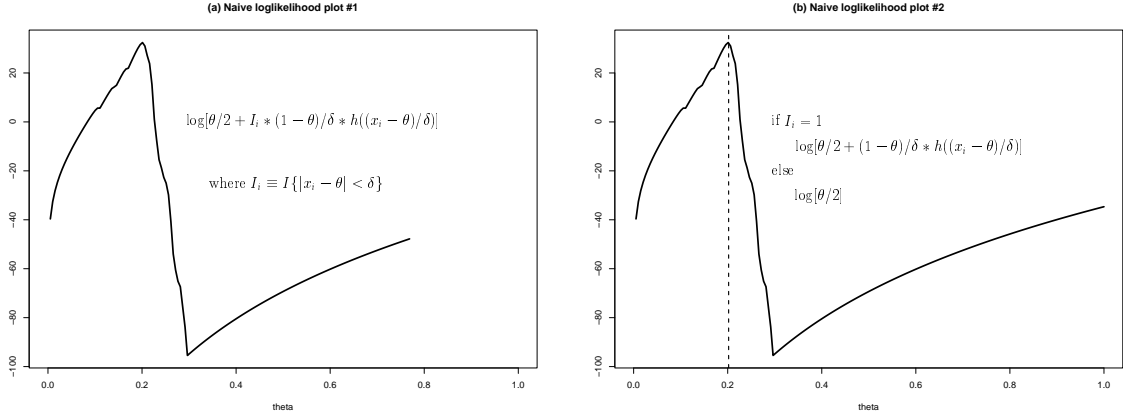


Figure 2: Plot (a) fails, essentially due to the fact that the computer has not been told what to do with a number like $0 \times \infty$. The problem is corrected in plot (b), which uses a different algorithm that is mathematically but not computationally equivalent. Unfortunately, even plot (b) misses important features of the loglikelihood surface; in particular, the correct MLE is *not* 0.202 as it appears to be.

Bringing these features into view requires a slightly more detailed analysis.

Note that if $\theta = x_i$ for some sample point x_i , then $I_i = 1$ and $h[(x_i - \theta)/\delta] = 1$, so this point contributes

$$\log \left[\frac{\theta}{2} + \frac{1 - \theta}{\delta} \right] \quad (3)$$

to the log-likelihood. As noted previously, $1/\delta$ can be enormous for θ close to 1, in which case expression (3) results in an overflow error. This fact illustrates the valuable lesson that $\log[\exp(x)]$ is not always equal to x in computer arithmetic. When θ is near 1, expression (3) is approximately

$$\log(1 - \theta) - \log \delta = \log(1 - \theta) + \frac{1}{(1 - \theta)^4}; \quad (4)$$

in fact, in double-precision computer arithmetic, this “approximation” is exact for $\theta > .6$. Therefore, to correctly represent the loglikelihood and find the MLE, it is necessary to evaluate the loglikelihood at all $\theta = x_i$ specifically, and to use expression (4) when the naive approach

in expression (3) results in overflow. A graph that correctly does this is given in Figure 3. We see that the true MLE is equal to the largest sample value.

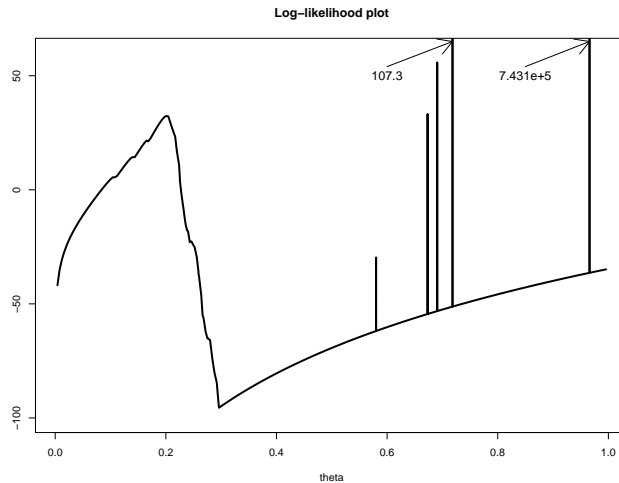


Figure 3: This is the correct plot, showing that the MLE is the largest sample value, about 0.96.

This exercise is really about an example in which the MLE is not consistent — not a subject typically encountered in a computational statistics course! Nevertheless, the exercise as stated forces students to confront several computing issues, including how to sample from a mixture distribution; how to sample from a distribution like $h(x)$; what can happen when we mean 0 but code it as $0 \times \infty$; the effect of overflow error and the fact that $\log[\exp(x)]$ is not always x in computer arithmetic; and how to produce a graph of a function.

3 Hardy-Weinberg equilibrium and Pearson's chi-square

Exercise: *There is a particular characteristic of human blood (the so-called MN blood group) that occurs in three distinct types: M, MN, and N. Under idealized circumstances called Hardy-Weinberg equilibrium, or HWE, these three types occur in a population with relative frequencies $p_1 = \pi_M^2$, $p_2 = 2\pi_M\pi_N$, and $p_3 = \pi_N^2$, respectively, where π_M is the relative frequency of the M allele in the population and $\pi_N = 1 - \pi_M$ is the frequency of the N allele. In a given sample,*

let n_1 , n_2 , and n_3 denote the number of individuals with the M , MN , and N types, respectively. If the value of π_M is not known, it may be estimated by the maximum likelihood estimator $\hat{\pi}_M = (2n_1 + n_2)/2n$, where $n = n_1 + n_2 + n_3$.

Simulate 1000 samples of size $n = 50$ from a population in which $\pi_M = 0.3$ and HWE holds. For each sample, compute both

$$W^2 = \sum_{j=1}^3 \frac{(n_j - np_j)^2}{np_j} \quad \text{and} \quad X^2 = \sum_{j=1}^3 \frac{(n_j - n\hat{p}_j)^2}{n\hat{p}_j}.$$

(The statistic W^2 tests the hypothesis $H_0 : \pi_M = .3$ and HWE holds, and X^2 tests $H_0 : \text{HWE holds.}$) Plot the empirical cumulative distribution functions for the 1000 values of W^2 and the 1000 values of X^2 . Compare each graph with chi-squared distribution functions on 1 and 2 degrees of freedom. What do you conclude?

This is an example of a general type of computational exercise that is relatively easy to generate in an asymptotics course: A simulation study that checks an asymptotic result for a fixed sample size. This method of generating exercises is appealing since an elegant theoretical result can be all the more satisfying when it checks out empirically. However, because such exercises are easy to generate, there is a temptation to overproduce them, asking students to verify every new asymptotic result via simulation. Such a temptation should probably be avoided if only to preserve the students' sanity, though several carefully thought-out exercises of this type are definitely appropriate. It is particularly helpful if the different exercises are designed so that they cover a broad range of computing skills, rather than relying on the same limited set of skills again and again.

Figure 4 depicts 1000 simulated data points as called for in the exercise. Simulating the data amounts to simulating realizations from a multinomial distribution. Although many software packages will do this automatically, it is still worthwhile for students to know how to use the software to do it. And even if multinomial samples can be generated automatically, students might be encouraged to write their own multinomial-generating methods based on nothing but uniform pseudorandom numbers. Note that if true multinomial probabilities were used in place

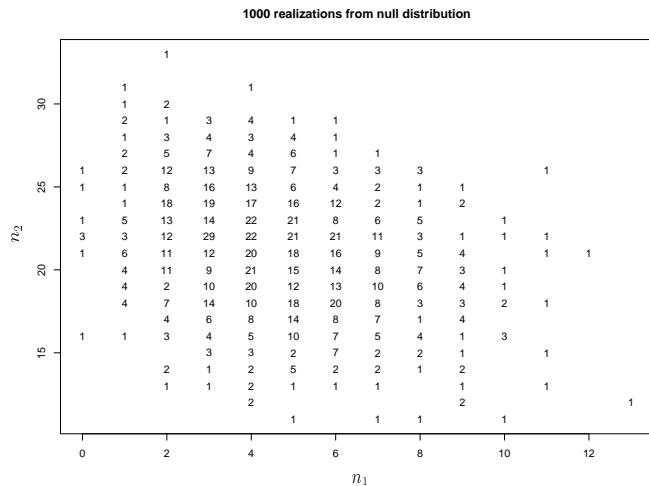


Figure 4: Shown are 1000 samples from $\text{multinomial}\{50, (.3^2, 2 \times .3 \times .7, .7^2)\}$. Only the values of n_1 and n_2 are shown; n_3 is always $50 - n_1 - n_2$. The plotted values are frequencies. For instance, there were 15 occurrences of $(5, 20, 25)$. Since $(5, 20, 25)$ results in $\hat{\pi}_M = .3$, the values of W^2 and X^2 coincide in this case and $W^2 = X^2 = 0.113$.

of simulated frequencies in Figure 4, the *exact* distributions of W^2 and X^2 could be computed; this too would be a useful computing exercise.

Once the multinomials are generated, the task remains to calculate 2000 chi-squared statistics. This clearly requires a bit of programming ability, though not a lot — after all, this is an asymptotics course, not a programming course. Doing this won't be difficult for students who already have some programming skill, and for those who don't it offers the chance to learn some simple programming techniques such as how to implement a loop, how to translate mathematical expressions into computer code, and how to store the results of multiple calculations for later use. I find that in the case of a student who has no idea how to begin such a task, offering a piece of example code performing a similar job is always sufficient to catalyze the student's successful completion of the task.

Once the W^2 and X^2 statistics are computed, students must grapple with the task of how to plot an empirical cdf, then how best to compare the result with chi-square distribution functions.

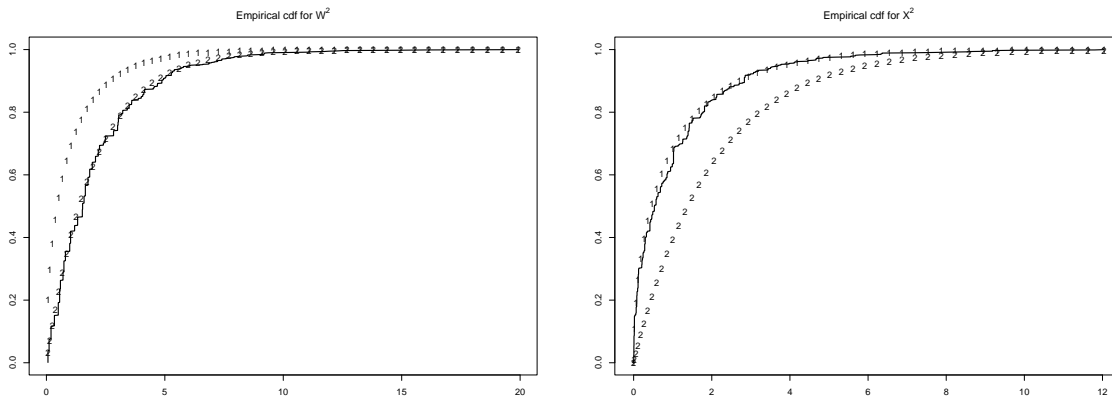


Figure 5: Shown in the left and right plots are the empirical cumulative distribution functions of 1000 realizations of W^2 and X^2 , respectively. Superimposed on the graphs are the true cumulative distribution functions for χ_1^2 and χ_2^2 random variables, marked with 1's and 2's.

Even obtaining the chi-square distribution functions is not a completely trivial exercise since this capability of statistical software is something with which students need to be familiar if they are not already. Figures 5(a) and 5(b) depict the empirical distribution functions of the W^2 and X^2 statistics, respectively. These figures powerfully illustrate the difference between the distribution of W^2 and the distribution of X^2 and how closely the asymptotic theory matches the finite-sample reality in this example.

It should be stressed that carrying out this exercise does not in any way serve as proof of the asymptotic distributions of W^2 and X^2 , nor does it make such a proof less important in an asymptotics course. In fact, I would typically assign students this exercise in addition to theoretical proofs that W^2 and X^2 converge in distribution to χ_2^2 and χ_1^2 as $n \rightarrow \infty$. Computing should aid the instruction of theory, not supplant it.

4 Efficient one-step estimators

Exercise: Generate 1000 random samples of size $n = 15$ from a logistic distribution with distribution function $F_\theta(x) = 1/(1 + \exp\{\theta - x\})$. Use $\theta = 2$. For each sample, calculate

the median $\tilde{\theta}$, the maximum likelihood estimator $\hat{\theta}$, and the one-step Newton-method estimator based on $\tilde{\theta}$, namely

$$\delta = \tilde{\theta} - \frac{\ell'(\tilde{\theta})}{\ell''(\tilde{\theta})}, \quad (5)$$

where $\ell(\theta)$ is the loglikelihood function and $\ell'(\theta)$ and $\ell''(\theta)$ are its first and second derivatives. Find the sample variances of $\tilde{\theta}$, $\hat{\theta}$, and δ ; how well do they match the asymptotic theory?

Like the previous exercise, this one is of the “check asymptotic results for finite samples” type. In this case, however, the theory itself is very suggestive of a computational topic: The asymptotic efficiency of so-called one-step estimators, like δ above, necessitates a discussion of Newton’s method at the very least. And since Newton’s method is such a commonly used computational tool, it is only natural to devise an exercise so that students are required to implement it.

Before tackling the problem, a bit of asymptotics background might be helpful. The maximum likelihood estimator in this problem (and in many other problems) has the desirable characteristic that it is asymptotically efficient, which is to say that its asymptotic variance is in some sense as small as possible. On the other hand, the sample median, which is also a consistent estimator in this problem, has a larger asymptotic variance than the MLE. In particular, if n denotes the sample size, then an asymptotic approximation tells us that the MLE $\hat{\theta}$ is roughly normally distributed with variance $3/n$, whereas the median $\tilde{\theta}$ is roughly normally distributed with variance $4/n$. Incidentally, this problem could just as easily have been written using the sample mean \bar{X} in place of (or in addition to) the sample median; we may verify using the central limit theorem that \bar{X} is approximately normally distributed with variance $\pi^2/3n \approx 3.29/n$.

In this exercise, there is no closed-form solution to the likelihood equation; therefore, finding the maximum likelihood estimator requires some kind of numerical method. Newton’s method is a logical choice: Letting $\ell(\theta)$ denote the loglikelihood function and starting from some value

θ_0 , let

$$\theta_{i+1} = \theta_i - \frac{\ell'(\theta_i)}{\ell''(\theta_i)} \quad (6)$$

for $i = 0, 1, \dots$ until the value of θ_i stops changing. An interesting asymptotic result is that if we take θ_0 to be, say, the median $\tilde{\theta}$, then θ_1 itself is an asymptotically efficient estimator, just like the MLE $\hat{\theta}$. This is how δ is defined in equation (5), which is why δ may be called a one-step estimator.

It would be instructive to supplement this exercise in order to introduce students to the technique of Fisher scoring, in which $\theta_{i+1} = \theta_i + 3\ell'(\theta_i)/n$ for this example. This formula arises when $\ell''(\theta_i)$ of equation (6) is replaced by $-nI(\theta)$, where $I(\theta)$, the Fisher information, equals $1/3$ in this case. Like the one-step Newton-method estimator of equation (5), the one-step scoring estimator is asymptotically efficient.

The main computing benefit of this exercise is that it forces students to implement Newton's method. Because the same basic algorithm used to compute δ may be iterated to find the MLE, students should be encouraged to write a program that could be used for either purpose: Newton's method could be iterated only once or repeatedly until convergence, depending on a user-settable option. This subtle point encourages good programming practice as opposed to a "brute-force" approach in which students write two separate, inflexible programs with substantial overlap.

Table 1 exhibits three of the 1000 samples I generated to complete this exercise, along with the corresponding values of $\tilde{\theta}$, $\hat{\theta}$, and δ . We see that the sample variances are a bit lower than the $4/15 = 0.267$ for the median and $3/15 = 0.200$ for the MLE and one-step estimator predicted by the asymptotic approximation. Nonetheless, the values are pretty close; as a related activity, students could be asked to report approximate confidence intervals for the true variances to see whether the asymptotically predicted values fall within the intervals.

Glancing at the last two columns of Table 1 reveals an insight that is not implied by the asymptotic theory: When Newton's method is started at a reasonable estimate of θ , such as

Sample number	Simulated Data ($n = 15$)					Median $\tilde{\theta}$	MLE $\hat{\theta}$	One-step δ
1	-1.456	0.257	0.376	0.356	2.699	2.364	1.961	1.964
	4.461	1.728	3.479	4.312	3.883			
	2.863	-0.640	2.364	-0.317	3.092			
2	0.451	-3.310	0.104	2.431	0.077	2.175	1.831	1.831
	1.567	-0.111	1.981	2.360	2.223			
	2.175	4.002	4.807	3.096	2.800			
⋮	⋮					⋮	⋮	⋮
1000	1.452	5.630	0.623	1.351	-3.126	1.351	1.545	1.545
	4.681	0.924	0.702	0.976	3.137			
	2.083	2.158	-0.814	2.691	1.205			
Sample variances:						0.2347	0.1870	0.1871

Table 1: Three of the 1000 samples are shown, along with three different estimators of θ for each sample.

the median in this case, a single iteration of Newton’s method winds up very close to the MLE. In fact, the sample mean of $|\delta - \hat{\theta}|$ for these data is only 0.00077. Coupled with the asymptotic theory showing that δ is essentially just as good an estimator as $\hat{\theta}$, this bit of empirical evidence helps the student understand that the one-step estimator is an attractive alternative to the MLE from a computational standpoint. Because the asymptotic theory in this example blends so nicely with the computing, each informing the other, a golden opportunity is missed if students are asked to learn the theory of these one-step estimators without performing any computation.

5 Discussion

It is certainly important to offer a course or courses dedicated to statistical computing. Yet it is also important — perhaps even more so because it is not always logistically feasible to require such courses — to incorporate computing into the existing curriculum. Thus, teaching statistical computing is not merely a topic for instructors of statistical computing courses to worry about

by themselves. This point is illustrated in this article by reference to a revamped stochastic processes course that now includes a lengthy introduction to Markov chain Monte Carlo methods and by three specific exercises designed for use in a theoretical course in asymptotics.

I have intentionally avoided mention of any particular software package; naturally, different instructors may have different software preferences. All of the exercises here can be completed using almost any statistical software and a bit of ingenuity. Even low-level languages like C and FORTRAN may be used, though in those cases it may be necessary to rely on already-written software to handle tasks such as uniform random number generation and graphical plotting.

The three example exercises presented in this article demonstrate the useful symbiosis that occurs when computing is thoughtfully incorporated into a theoretical statistics course. There are several means by which such exercises may be created: As instructors, we should watch for things like theoretical results that lend themselves to simulation; large-sample results that may be checked for fixed small sample sizes; problems with closed-form solutions that may be checked numerically, or problems without closed form solutions that may be solved numerically; and statistical methods that may be illustrated using data. However, computing exercises should not merely be created haphazardly; careful thought should go into their pedagogical value. Such a focus on pedagogy, applied to the implementation of computing assignments, might even spread to all aspects of the course — not a bad outcome!

References

- Ferguson, T. S. (1982), An inconsistent maximum likelihood estimator, *Journal of the American Statistical Association*, **77**: 831–834.
- Gentle, J. E. (2004), Courses in statistical computing and computational statistics, *The American Statistician*, **58**: 2–5.
- Gilks, W. R., Richardson, S., and Spiegelhalter, D. J. (1996), “Introducing Markov Chain

Monte Carlo,” in *Markov Chain Monte Carlo in Practice*, W. R Gilks, S. Richardson, and D. J. Spiegelhalter, editors. Boca Raton: Chapman & Hall/CRC press.

Lange, K. (2004), Computational statistics and optimization theory at UCLA, *The American Statistician*, **58**: 9–11.

Monahan, J. (2004), Teaching statistical computing at North Carolina State University, *The American Statistician*, **58**: 6–8.