1

Introducing Point and Interval Estimation

> The discussions of theoretical statistics may be regarded as alternating between problems of estimation and problems of distribution. In the first place a method of calculating one of the population parameters is devised from common-sense considerations: we next require to know its probable error, and therefore an approximate solution of the distribution, in samples, of the statistics calculated.
>
> *(R. A. Fisher, 1922, reproduced in Kotz and Johnson, 1992)*

This chapter and the next two introduce the primary tools and concepts underlying most all problems in statistical inference. We restrict ourselves herein to the independent, identically distributed (i.i.d.) framework, in order to emphasize the fundamental concepts without the need for addressing the additional issues and complexities associated with the workhorse models of statistics, such as linear models, analysis of variance, design of experiments, and time series. The overriding goal is to extract relevant information from the available sample in order to learn about the underlying population from which it was drawn.

We begin with the basic definitions associated with point estimation, and introduce the maximum likelihood estimator (m.l.e.). We will have more to say about point estimation and m.l.e.s in Chapters 3 and 5. The remainder of the chapter is dedicated to individual parameter confidence intervals (c.i.s), restricting attention to the intuitive use of computer-intensive methods for their construction, as they are generally applicable and, for more complex problems, often the only available choice. In particular, a natural progression is made from simulation to the parametric bootstrap, to the nonparametric bootstrap, to the double nonparametric bootstrap, and finally to the double bootstrap with analytic inner loop, the latter using techniques from Chapter 8.

Fundamental Statistical Inference: A Computational Approach, First Edition. Marc S. Paolella.

1.1 POINT ESTIMATION

To introduce the notion of parameter estimation from a sample of data, we make use of two simple models, the Bernoulli and geometric.

1.1.1 Bernoulli Model

Consider an idealized experiment that consists of randomly drawing a marble from an urn containing R red and W white marbles; its color is noted and it is then placed back into the urn. This is repeated n times, whereby n is a known, finite constant, *but R and W are unknown*. This corresponds to a sequence of Bernoulli trials with unknown probability $p = R/(R+W)$ or $p = W/(R+W)$, depending on what one wants to consider a "success." Assuming the former, let X_i, $i = 1, \ldots, n$, denote the outcomes of the experiment, with $X_i \overset{\text{i.i.d.}}{\sim} \text{Bern}(p)$, each with support $\mathcal{S} = \{0, 1\}$. The ultimate goal is to determine the value of p. If n is finite (as reality often dictates), this will be an impossible task. Instead, we content ourselves with attempting to infer as much information as possible about the value of p.

As a starting point, in line with Fisher's "common-sense considerations" in the opening quote, it seems reasonable to examine the proportion of successes. That is, we would compute s/n, where s is the observed number of successes. The value s/n is referred to as a **point estimate** of p and denoted $\hat{p}$, pronounced "p hat." Sometimes it is advantageous to write $\hat{p}_n$, where the subscript indicates the sample size. From the way in which the experiment is defined, it should be clear that s is a realization from the binomial random variable (r.v.) $S = \sum_{i=1}^{n} X_i \sim \text{Bin}(n, p)$. To emphasize this, we also write $\hat{p} = S/n$ and call this a **point estimator** of p, the distinction being that a point estimator is a random variable, while a point estimate is a realization of this random variable resulting from the outcome of a particular experiment.

Note that the same notation of adding a "hat" to the parameter of interest is used to denote both estimate and estimator, as this is the common standard. However, the distinction between estimate and estimator is crucial when attempting to assess the properties of $\hat{p}$ (e.g., is it correct on average?) and compare its performance to other possible estimators (e.g., is one point estimator more likely to be correct than the other?). For instance, $\mathbb{E}[s/n] = s/n$, that is, s/n is a post-experiment constant, while $\mathbb{E}[S/n] = (np)/n = p$ can be computed before or after the experiment takes place. In this case, estimator S/n is said to be **(mean) unbiased**.

More formally, let $\hat{\theta}$ be a point estimator of the finite, fixed, unknown parameter $\theta \in \Theta \subset \mathbb{R}$ such that $\mathbb{E}[\hat{\theta}]$ exists. Then:

> The point estimator $\hat{\theta}$ is **(mean) unbiased** (with respect to the set Θ) if its expected value is θ (for all $\theta \in \Theta$); otherwise it is **(mean) biased** with $\text{bias}(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \theta$.

Generally speaking, mean unbiasedness is a desirable property because it implies that we are "correct on average," where the "average" refers to the hypothetical idea of repeating the experiment infinitely often – something that of course does not actually happen in reality. An impressive theoretical framework in mathematical statistics was developed, starting in the 1950s, for the derivation and study of unbiased estimators with minimum variance; see Chapter 7, especially Section 7.2. It is often the case, however, that estimators can be found

that are biased, but, by virtue of having a lower variance, wind up having a lower mean squared error, as seen from (1.2) directly below. This concept is well known, and reflected, for example, in Shao and Tu (1995, p. 67), stating "We need to balance the advantage of unbiasedness against the drawbacks of a large mean squared error." Another type of unbiasedness involves using the median instead of the mean. See Section 7.4.2 for details on median-unbiased estimators.

For the binomial example, the variance of estimator $\hat{p}$ is

$$\mathbb{V}(\hat{p}) = \mathbb{V}\left(\frac{\sum_{i=1}^{n} X_i}{n}\right) = n\frac{p(1-p)}{n^2} = \frac{p(1-p)}{n}, \tag{1.1}$$

and it clearly goes to zero as the sample size increases. This is also desirable, because, as more samples are collected, the amount of information from which p is to be inferred is growing. This concept is referred to as consistency; recalling the definition of convergence in probability from (A.254) and the weak law of large numbers (A.255), the following definition should seem natural:

> An estimator $\hat{\theta}_n$ based on a sample of n observations is **weakly consistent** (with respect to Θ) if, as $n \to \infty$, $\Pr(|\hat{\theta}_n - \theta| > \epsilon) \to 0$ for any $\epsilon > 0$ (and all $\theta \in \Theta$).

Observe that an estimator can be (mean) unbiased but not consistent: if $X_i \overset{\text{i.i.d.}}{\sim} \mathrm{N}(\mu, 1)$, $i = 1, \ldots, n$, then the estimator $\hat{\mu} = X_1$ is unbiased, but it does not converge to μ as the sample size increases.

Another popular measure of the quality of an estimator is its expected squared deviation from the true value, called *mean squared error*, or *m.s.e.*:

> The **mean squared error** of the estimator $\hat{\theta}$ is defined as $\mathbb{E}[(\hat{\theta} - \theta)^2]$.

An important decomposition of the m.s.e. is as follows. With $\Xi = \mathbb{E}[\hat{\theta}]$,

$$\begin{aligned}
\text{m.s.e.}\,(\hat{\theta}) &= \mathbb{E}[(\hat{\theta} - \theta)^2] = \mathbb{E}[(\hat{\theta} - \Xi + \Xi - \theta)^2] \\
&= \mathbb{E}[(\hat{\theta} - \Xi)^2] + \mathbb{E}[(\Xi - \theta)^2] + \text{cross-term, which is zero} \\
&= \mathbb{E}[(\hat{\theta} - \Xi)^2] + (\Xi - \theta)^2 = \mathbb{V}(\hat{\theta}) + [\text{bias}\,(\hat{\theta})]^2.
\end{aligned} \tag{1.2}$$

The reader should quickly verify that the cross-term is indeed zero. Note that, for an unbiased estimator, its m.s.e. and variance are equal. As the estimator $\hat{\theta}$ is a function of the data, it is itself a random variable. With $f = f_{\hat{\theta}}$ the p.d.f. of $\hat{\theta}$, we can write $\Pr(|\hat{\theta} - \theta| > \epsilon)$ for any $\epsilon > 0$ as

$$\int_{|t-\theta|>\epsilon} f(t)\,\mathrm{d}t \le \int_{|t-\theta|>\epsilon} \frac{(t-\theta)^2}{\epsilon^2} f(t)\,\mathrm{d}t \le \int_{-\infty}^{\infty} \frac{(t-\theta)^2}{\epsilon^2} f(t)\,\mathrm{d}t = \frac{\mathbb{E}[(\hat{\theta} - \theta)^2]}{\epsilon^2},$$

so that $\hat{\theta}$ is weakly consistent if m.s.e. $(\hat{\theta}) \to 0$.

The estimator $\hat{p} = S/n$ for the Bernoulli model is rather intuitive and virtually presents itself as being a good estimator of p. It turns out that this $\hat{p}$ coincides with the estimator we obtain when applying a very general and powerful method of obtaining an estimator for an

unknown parameter of a statistical model. We briefly introduce this method now, and will have more to say about it in Section 3.1.

The **likelihood function** $\mathcal{L}(\theta; \mathbf{x})$ is the joint density of a sample $\mathbf{X} = (X_1, \ldots, X_n)$ as a function of the (for now, scalar) parameter θ, for fixed sample values $\mathbf{X} = \mathbf{x}$. That is, $\mathcal{L}(\theta; \mathbf{x}) = f_{\mathbf{X}}(\mathbf{x}; \theta)$, where $f_{\mathbf{X}}$ is the p.m.f. or p.d.f. of $\mathbf{X}$. Let $\ell(\theta; \mathbf{x}) = \log \mathcal{L}(\theta; \mathbf{x})$,[1] and write just $\ell(\theta)$ when the data are clear from the context. Denote the first and second derivatives of $\ell(\theta)$ with respect to θ by $\dot{\ell}(\theta)$ and $\ddot{\ell}(\theta)$, respectively. The **maximum likelihood estimate**, abbreviated m.l.e. and denoted by $\hat{\theta}$ (or, to distinguish it from other estimates, $\hat{\theta}_{\text{ML}}$), is that value of θ that maximizes the likelihood function for a given data set $\mathbf{x}$. The **maximum likelihood estimator** (as opposed to *estimate*) is the function of the X_i, also denoted $\hat{\theta}_{\text{ML}}$, that yields the m.l.e. for an observed data set $\mathbf{x}$.

In many cases of interest (including the Bernoulli and geometric examples in this chapter), the m.l.e. satisfies $\dot{\ell}(\hat{\theta}) = 0$ and $\ddot{\ell}(\hat{\theta}) < 0$. For example, with $X_i \overset{\text{i.i.d.}}{\sim} \text{Bern}(\theta)$, $i = 1, \ldots, n$, the likelihood is

$$\mathcal{L}(\theta; \mathbf{x}) = \prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{1-x_i} \mathbb{I}_{\{0,1\}}(x_i) = \theta^s (1-\theta)^{n-s} \mathbb{I}_{\{0,1,\ldots,n\}}(s),$$

where $s = \sum_{i=1}^{n} x_i$. Then

$$\dot{\ell}(\theta) = \frac{s}{\theta} - \frac{n-s}{1-\theta} \quad \text{and} \quad \ddot{\ell}(\theta) = -\frac{s}{\theta^2} - \frac{n-s}{(1-\theta)^2},$$

from which it follows (by setting $\dot{\ell}(\hat{\theta}) = 0$ and confirming $\ddot{\ell}(\hat{\theta}) < 0$) that $\hat{\theta}_{\text{ML}} = S/n$ is the m.l.e. It is easy to see that $\hat{\theta}_{\text{ML}}$ is unbiased.

1.1.2 Geometric Model

As in the binomial case, independent draws with replacement are conducted from an urn with R red and W white marbles. However, now the number of trials is not fixed in advance; sampling continues until r red marbles have been drawn. What can be said about $p = R/(R+W)$? Let the r.v. X be the number of necessary trials. From the sampling structure, X follows a negative binomial distribution, $X \sim \text{NBin}(r, p)$, with p.m.f.

$$f_X(x; r, p) = \binom{x-1}{r-1} p^r (1-p)^{x-r} \mathbb{I}_{\{r, r+1, \ldots\}}(x). \tag{1.3}$$

Recall that X can be expressed as the sum of r i.i.d. geometric r.v.s, say $X = \sum_{i=1}^{r} G_i$, where $G_i \overset{\text{i.i.d.}}{\sim} \text{Geo}(p)$, each with support $\{1, 2, \ldots\}$.

This decomposition is important because it allows us to imagine that sampling occurs not necessarily consecutively in time until r successes occur, but rather as r independent (and possibly concurrent) geometric trials using urns with the same red to white ratio, that is, the same p. For example, interest might center on how long it takes a woman to become pregnant using a particular method of assistance (e.g., temperature measurements or hormone treatment). This is worth making an example, as we will refer to it more than once.

[1] Throughout this book, log refers to base e unless otherwise specified.

Example 1.1 (Geometric) *Let* $G_i \overset{\text{i.i.d.}}{\sim} \text{Geo}(\theta)$, $i = 1, \ldots, n$, *with typical p.m.f.*

$$f_G(x;\theta) = \theta(1-\theta)^{x-1}\mathbb{I}_{\{1,2,\ldots\}}(x), \quad \theta \in \Theta = (0,1).$$

Then $\ell(\theta;\mathbf{x})$, *the log-likelihood of the sample* $\mathbf{x} = (x_1, \ldots, x_n)$, *and its first derivative,* $\dot{\ell}(\theta;\mathbf{x})$, *are, with* $s = \sum_{i=1}^n x_i$,

$$\ell(\theta;\mathbf{x}) = n\log(\theta) + \log(1-\theta)\sum_{i=1}^n (x_i - 1), \quad \dot{\ell}(\theta;\mathbf{x}) = \frac{n}{\theta} - \frac{s-n}{1-\theta}.$$

Solving the equation $\dot{\ell}(\theta;\mathbf{x}) = 0$ *and confirming* $\ddot{\ell}(\hat{\theta}) < 0$ *gives* $\hat{\theta}_{ML} = n/S = 1/\bar{G}$. *We will see below and in Section 7.3 that the m.l.e. is not unbiased.*[2] ■

Imagine a study in which each of r couples (independently of each other) attempts to conceive each month until they succeed. In the $r = 1$ case, $X = G_1 \sim \text{Geo}(p)$ and, recalling that $\mathbb{E}[G_1] = 1/p$, an intuitive point estimator of p is $1/G_1$. Interest centers on developing a point estimator for the $r > 1$ case. Of course, in this simple structure, one would just compute the m.l.e. However, we use this easy case to illustrate how one might proceed when simple answers are not immediately available, and some thinking and creativity are required.

Based on the result for $r = 1$, one idea for the $r > 1$ case would be to use the average of the $1/G_i$ values, $r^{-1}\sum_{i=1}^r G_i^{-1}$, which we denote by $\hat{p}_1$. Another candidate is $\hat{p}_2 = 1/\bar{G} = r/\sum_{i=1}^r G_i = r/X$. This happens to be the m.l.e. from Example 1.2. Note that both of these estimators reduce to $1/G_1$ when $r = 1$. We also consider the nonobvious point estimator $\hat{p}_3 = (r-1)/(X-1)$. It will be derived in Section 7.3, and is only useful for $r > 1$.

Instead of algebraically determining the mean and variance of the $\hat{p}_i$, $i = 1, 2, 3$, we will begin our practice of letting the computer do the work. The program in Listing 1.1 computes the three point estimators for a simulated set of G_i; it repeats this `sim` = 10,000 times, and the resulting sample mean and variance of these simulated estimates approximate the true mean and variance.

To illustrate, Figure 1.1 shows the histograms of the simulated point estimators for the case with $p = 0.3$ and $r = 5$. From these, the large upward bias of $\hat{p}_1$ is particularly clear.

```
1  function [p1vec, p2vec, p3vec]=geometricparameterestimate(p,r,sim)
2  p1vec = zeros(sim,1); p2vec = p1vec; p3vec = p1vec;
3  for s=1:sim
4    gvec=geornd(p,[r 1]) +1;
5    p1 = mean(1./gvec); p2 = 1/mean(gvec); p3 = (r-1) / (sum(gvec)-1);
6    p1vec(s) = p1; p2vec(s) = p2; p3vec(s) = p3;
7  end
8  bias1 = mean(p1vec)-p, bias2 = mean(p2vec)-p, bias3 = mean(p3vec)-p
9  var1 = var(p1vec), var2 = var(p2vec), var3 = var(p3vec)
10 mse1 = var1+bias1^2, mse2 = var2+bias2^2, mse3 = var3+bias3^2
```

Program Listing 1.1: Simulates three point estimators for p in the i.i.d. geometric model. Calling the function with $p = 0.3$ and $r = 5$ corresponds to the true probability of success being 0.3 and using five couples in the experiment.

[2] We use the symbol ■ to denote the end of proofs of theorems, as well as examples and remarks, acknowledging that it is traditionally only used for the former, as popularized by Paul Halmos.

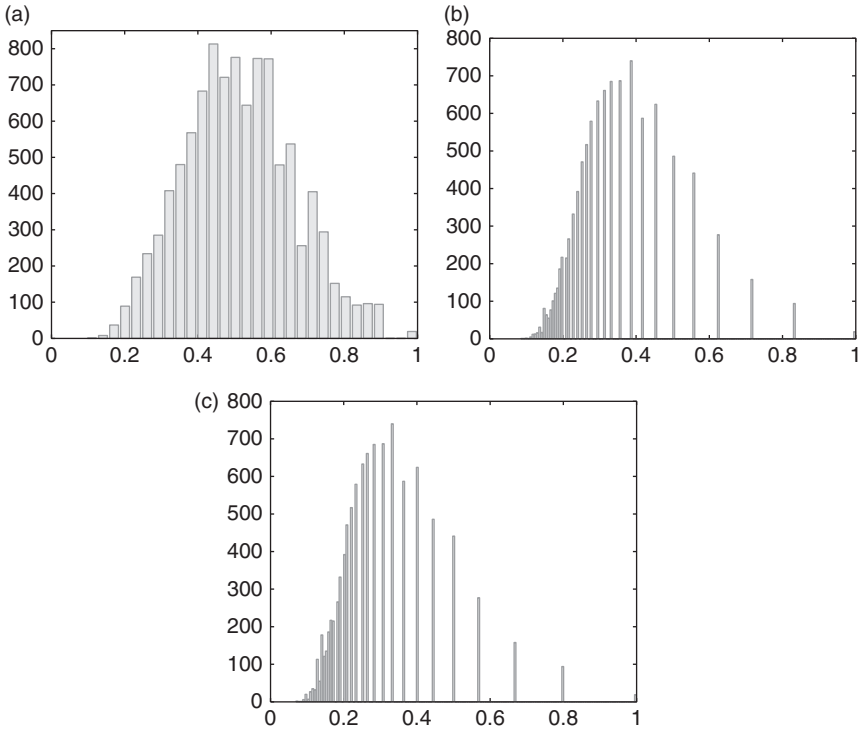

Figure 1.1 *Distribution of point estimators $\hat{p}_1$ (a), $\hat{p}_2$ (b), and $\hat{p}_3$ (c) using output from the program in Listing 1.1 with $p = 0.3$ and $r = 5$, based on simulation with 10,000 replications.*

The discrete nature of $\hat{p}_2$ and $\hat{p}_3$ arises because these two estimators first compute the sum of the observations and then take reciprocals, so that computation of their p.m.f.s is easy. As an example, $\hat{p}_3 = 0.4 \Leftrightarrow X = 11$, which, from (1.3), has probability 0.06, so that approximately 600 of the simulated values depicted in the histogram of $\hat{p}_3$ should be 0.4; there are 624 in the histogram. Similarly, $\hat{p}_3 = 0.8 \Leftrightarrow X = 6$, with probability 0.008505, and 94 in the histogram.

As p increases towards one, the number of points in the supports of $\hat{p}_2$ and $\hat{p}_3$ decreases. This is illustrated in Figure 1.2, showing histograms of $\hat{p}_2$ for $r = 10$ and four different values of p. The code used to make the plots is given in Listing 1.2. Observe how we avoid use of the FOR loop (as was used in Listing 1.1) for generating the 1 million replications, thus providing a significant speed increase. (The use of the `eval` command with concatenated text strings is also demonstrated.)

For the simulation of $\hat{p}_1$, $\hat{p}_2$, and $\hat{p}_3$ from Listing 1.1, with $p = 0.3$ and $r = 5$, the results are shown in the first numeric row of Table 1.1. We see that $\hat{p}_1$ has almost five times the bias of $\hat{p}_2$, while $\hat{p}_2$ has over 100 times the bias of $\hat{p}_3$. The variance of $\hat{p}_1$ is slightly larger than those of $\hat{p}_2$ and $\hat{p}_3$, which are nearly the same. By combining these according to (1.2), it is clear that the m.s.e. will be smallest for $\hat{p}_3$, as also shown in the table. The next row shows the results using a larger sample of 15 couples. While the bias of $\hat{p}_1$ stays the same, those of $\hat{p}_2$ and $\hat{p}_3$ decrease. For all point estimators, the variance decreases.

It turns out that, as the number of couples, r, tends towards infinity, the variance of all the estimators goes to zero, while the bias of $\hat{p}_1$ stays at 0.22 and that of $\hat{p}_2$ goes to zero.

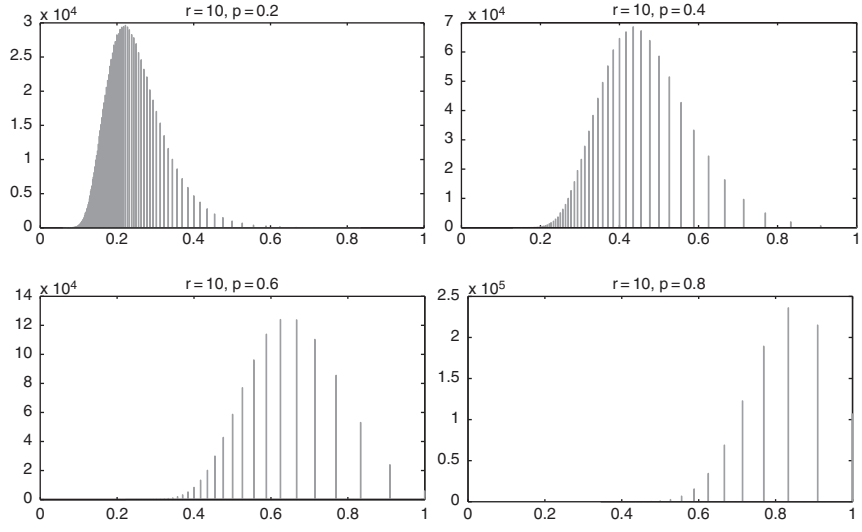

Figure 1.2 *Histogram of point estimator $\hat{p}_2$ for $r = 10$ and four values of p, based on simulation with 1 million replications.*

```
B=1e6; r=10;
for p=0.2:0.2:0.8
  phatvec = 1./mean(geornd(p,[r B])+1); % the MLE
  [histcount, histgrd] =hist(phatvec,1000);
  figure, h1=bar(histgrd,histcount); set(gca,'fontsize',16), xlim([0 1])
  title(['r=',int2str(r),', p=',num2str(p)])
  set(h1,'facecolor',[0.94 0.94 0.94],'edgecolor',[0.9 0.7 1])
  eval(['print -depsc phatforgeogetsmorediscretep',int2str(10*p)])
end
```

Program Listing 1.2: Generates the graphs in Figure 1.2.

Hence, we say that $\hat{p}_2$ is **asymptotically unbiased**. We will see in Section 7.3 that $\hat{p}_3$ is unbiased – not just asymptotically, but for all $0 < p \leq 1$ and any $r > 1$. This implies that the value 0.0004 in the $\hat{p}_3$ bias column of the table just reflects **sampling error** resulting from using only 10,000 replications in the simulation. In comparison, then, point estimator $\hat{p}_3$ seems to be preferred with respect to all three criteria.

The lower portion of Table 1.1 shows similar results using $p = 0.7$. Again, $\hat{p}_1$ is highly biased, while, comparatively speaking, the bias of $\hat{p}_2$ is much smaller and diminishes with growing sample size r. The bias of $\hat{p}_3$ appears very small and, as already mentioned, is theoretically zero. The interesting thing about this choice of p is that the variance of $\hat{p}_2$ is smaller than that of $\hat{p}_3$. In fact, this reduction in variance causes the m.s.e. of $\hat{p}_2$ to be smaller than that of $\hat{p}_3$ even though the bias of $\hat{p}_3$ is essentially zero. This demonstrates two important points:

(i) An unbiased point estimator need not have the smallest m.s.e.

(ii) The relative properties of point estimators may change with the unknown parameter of interest.

TABLE 1.1 Comparison of three point estimators for the geometric model

		bias			variance			m.s.e.		
p	r	$\hat{p}_1$	$\hat{p}_2$	$\hat{p}_3$	$\hat{p}_1$	$\hat{p}_2$	$\hat{p}_3$	$\hat{p}_1$	$\hat{p}_2$	$\hat{p}_3$
0.3	5	0.22	0.045	0.00040	0.023	0.018	0.017	0.070	0.020	0.017
0.3	15	0.22	0.015	0.00041	0.0077	0.0048	0.0045	0.055	0.0050	0.0045
0.7	5	0.13	0.040	0.00057	0.014	0.027	0.033	0.031	0.028	0.033
0.7	15	0.13	0.013	–0.00078	0.0046	0.0096	0.0102	0.022	0.0098	0.010

Having demonstrated these two facts using just the values in Table 1.1, it would be desirable to graphically depict the m.s.e. of estimators $\hat{p}_2$ and $\hat{p}_3$ as a function of p, for several sample sizes. This is shown in Figure 1.3, from which we see that m.s.e. $(\hat{p}_2) <$ m.s.e. $(\hat{p}_3)$ for (roughly) $p > 0.5$, but as the sample size increases, the difference in m.s.e. of the two estimators becomes negligible.

Facts (i) and (ii) mentioned above complicate the comparison of estimators. Some structure can be put on the problem if we restrict attention to unbiased estimators. Then, minimizing the m.s.e. is the same as minimizing the variance; this gives rise to the following concepts:

> An unbiased estimator, say $\hat{\theta}_{\text{eff}}$, is **efficient** (with respect to Θ) if it has the smallest possible variance of all unbiased estimators (for all $\theta \in \Theta$).
>
> The **efficiency** of an unbiased estimator $\hat{\theta}$ is $\text{Eff}(\hat{\theta}, \theta) = \mathbb{V}(\hat{\theta}_{\text{eff}})/\mathbb{V}(\hat{\theta})$.

We will see later (Chapter 7) that the estimator $\hat{p}_3$ used above is efficient.

In many realistic problems, there may be no unbiased estimators, or no efficient one; and if there is, like $\hat{p}_3$ above, it might not have the smallest m.s.e. over all or parts of Θ. This somewhat diminishes the value of the efficiency concept defined above. All is not lost, however. In many cases of interest, the m.l.e. has the property that, asymptotically, it is (unbiased and) efficient. As such, it serves as a natural benchmark with which to compare competing estimators. We expect that, with increasing sample size, the m.l.e. will eventually be as good as, or better than, all other estimators, with respect to m.s.e., for all $\theta \in \Theta$. This certainly does not imply that the m.l.e. is the best estimator in finite samples, as we see in Figure 1.3 comparing the m.l.e. $\hat{p}_2$ to the efficient estimator $\hat{p}_3$. (Other cases in which the

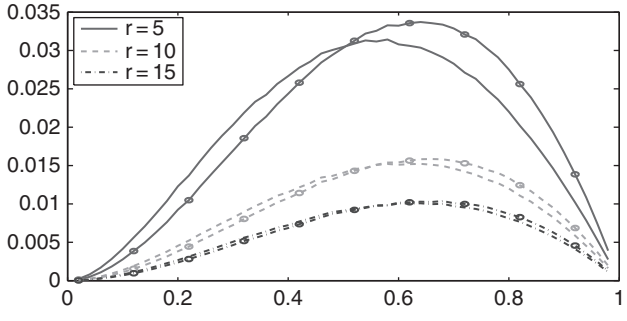

Figure 1.3 *The m.s.e. of estimators $\hat{p}_2$ (lines) and $\hat{p}_3$ (lines with circles) for parameter p in the geometric model, as a function of p, for three sample sizes, obtained by simulation with 100,000 replications.*

m.l.e. is not the best estimator with respect to m.s.e. in finite samples are demonstrated in Example 9.3 and Section 7.4.3.)

Before leaving this section, it is worth commenting on other facts observable from Figure 1.3. Note that, for any $r > 1$, the m.s.e.s for both estimators $\hat{p}_2$ and $\hat{p}_3$ approach zero as $p \to 0$ and $p \to 1$. This is because, for the former case, as $p \to 0$, $\mathbb{E}[X] = r/p \to \infty$, and $\hat{p} \to 0$. For the latter case, as $p \to 1$, $\Pr(X = r) = p^r \to 1$, so that $\hat{p} \to 1$. Also, the m.s.e. increases monotonically with p as p moves from 0^+ to (roughly) $p = 0.6$, and decreases monotonically back towards zero as $p \to 1$.

1.1.3 Some Remarks on Bias and Consistency

> The most effective way to discourage an applied statistician from using a model or method is to say that it gives asymptotically inconsistent parameter estimates. This is completely irrelevant for a fixed small sample; the interval of plausible values, not a point estimate, is essential. … If the sample were larger, in a properly planned study, the model could be different, so the question of a parameter estimate in some fixed model converging asymptotically to a "true" value does not arise.
>
> (J. K. Lindsey, 1999, p. 20)

Sections 1.1.1 and 1.1.2 introduced the fundamental concepts of point estimation, (mean) unbiasedness, consistency, m.s.e., likelihood, and efficiency. We demonstrated that a biased estimator might be preferred to an unbiased one with respect to m.s.e., such as the m.l.e., which, under certain conditions fulfilled in the vast majority of situations, is asymptotically unbiased and consistent (see Section 3.1.4 for the formalities of this).

While (mean) unbiasedness is an appealing property, in many modern applications, particularly in the context of big data and models with a relatively large number of parameters, unbiasedness is not only no longer a consideration, but biased estimates *are actually preferred*, via use of shrinkage estimation; see Chapter 5. Moreover, starting in the late twentieth century and continuing unabated, the Bayesian approach to inference has gained in attention and usage because of advances in computing power and recognition of some of its inferential benefits. In a sense, unbiasedness is the antithesis, or dual, of the Bayesian approach; see Noorbaloochi and Meeden (1983). So-called empirical Bayes methods form a link between pure Bayesian methods and shrinkage estimation, and yield a formidable approach to inference; see the references in Section 5.4.

As such, most researchers are now comfortable working with biased estimators, but will often still insist on such estimators being consistent. As consistency is an asymptotic notion, but reality deals with finite samples, one might also question its value, as suggested in the above quote from Lindsey (1999, p. 20). As a simple example of interest (particularly for anyone with a pension fund), consider a case from financial portfolio optimization. The basic framework of Markowitz (1952) (which led to him receiving the 1990 Nobel Memorial Prize in Economic Sciences) is still used today in industry, though (as was well known to Markowitz) the method is problematic because it requires estimating the mean vector and covariance matrix of past asset returns. This has been researched in a substantial body of literature, resulting in the established finding that shrinking the optimized portfolio weights towards the equally weighted vector (referred to as "$1/N$," where N is the number of assets

under consideration) not only improves matters substantially (in terms of a risk–reward tradeoff), but *just taking the weights to be the shrinkage target* $1/N$ often results in better performance.[3] Alternatively, one can apply the Markowitz optimization framework, but in conjunction with shrinkage applied to the mean vector and/or the covariance matrix.[4]

The humbling result that one is better off forgoing basic statistical modeling and just putting equal amounts of money in each available asset (roughly equivalent to just buying an exchange traded fund) arises because of (i) the high relevance and applicability of shrinkage estimation in this setting; and (ii) the gross misspecification of the model underlying the multivariate distribution of asset returns, and how it evolves over time. More statistically sophisticated models for asset returns *do exist*, such that portfolio optimization *does* result in substantially better performance than use of $1/N$ (let alone the naive Markowitz framework), though unsurprisingly, these are complicated for people not well versed in statistical theory, and require more mathematical and statistical prowess than usually obtained from a course in introductory statistical methods for aspiring investors.[5] Book IV will discuss some such models.

Clearly, $1/N$ is not a "consistent" estimator of the optimal portfolio (as defined by specifying some desired level of annual return, and then minimizing some well-defined risk measure, such as portfolio variance, in the Markowitz setting). More importantly, this example highlights the fact that the model used (an i.i.d. Gaussian or, more generally, an elliptic distribution, with constant unknown mean vector and covariance matrix throughout time) for the returns is so completely misspecified, that the notion of consistency becomes vacuous in this setting.

Two further, somewhat less trivial cases in which inconsistent estimators are favored (and also in the context of modeling financial asset returns), are given in Krause and Paolella (2014) and Gambacciani and Paolella (2017).

1.2 INTERVAL ESTIMATION VIA SIMULATION

To introduce the concepts associated with interval estimation and how simulating from the true distribution can be used to compute confidence intervals, we use the Bernoulli model from Section 1.1.1. For a fixed sample size n, we observe realizations of $X_1, \ldots, X_n$, where

[3] See, for example, DeMiguel et al. (2009a,b 2013) and the references therein.

[4] See, for example, Jorion (1986), Jagannathan and Ma (2003), Ledoit and Wolf (2003, 2004), Schäfer and Strimmer (2005), Kan and Zhou (2007), Fan et al. (2008); Bickel and Levina (2008), and the references therein.

[5] This result is also anathema to supposedly professional investment consultants and mutual fund managers, with their techniques for "stock picking" and "investment strategies." This was perhaps most forcefully and amusingly addressed by Warren Buffett (who apparently profits enormously from market inefficiency). "The Berkshire chairman has long argued that most investors are better off sticking their money in a low-fee S&P 500 index fund instead of trying to beat the market by employing professional stockpickers" (Holm, 2016). To quote Buffett: "Supposedly sophisticated people, generally richer people, hire consultants, and no consultant in the world is going to tell you 'just buy an S&P index fund and sit for the next 50 years.' You don't get to be a consultant that way. And you certainly don't get an annual fee that way. So the consultant has every motivation in the world to tell you, 'this year I think we should concentrate more on international stocks,' or 'this manager is particularly good on the short side,' and so they come in and they talk for hours, and you pay them a large fee, and they always suggest something other than just sitting on your rear end and participating in the American business without cost. And then, after they get their fees, they in turn recommend to you other people who charge fees, which … cumulatively eat up capital like crazy" (Holm, 2016). See also Sorkin (2017) on (i) Buffett's views; (ii) why many high wealth individuals continue to seek highly paid consultants; and (iii) with respect to the concept of market efficiency, what would happen if most wealth were channeled into exchange traded funds (i.e., the market portfolio).

$X_i \overset{\text{i.i.d.}}{\sim} \text{Bern}(p)$, and compute the mean of the X_i, $\hat{p} = S/n$, as our estimator of the fixed but unknown p. Depending on n and p, it could be that $\hat{p} = p$, though if, for example, n is odd and $p = 0.5$, then $\hat{p} \neq p$. Even if n is arbitrarily large but finite, if p is an irrational number in $(0, 1)$, then with probability one (w.p.1), $\hat{p} \neq p$.

The point is that, for almost all values of n and p, the probability that $\hat{p} = p$ will be low or zero. As such, it would seem wise to provide a set of values such that, with a high probability, the true p is among them. For a univariate parameter such as p, the most common set is an interval, referred to as a **confidence interval**, or **c.i.**

Notice that a c.i. pertains to a parameter, such as p, and *not* to an estimate or estimator, $\hat{p}$. We might speak of a c.i. *associated with* $\hat{p}$, in which case it is understood that the c.i. refers to parameter p. It does not make sense to speak of a c.i. for $\hat{p}$.

To get an idea of the uncertainty associated with $\hat{p}$ for a fixed n and p, we can use simulation. This is easily done in Matlab, using its built-in routine `binornd` for simulating binomial realizations.

```
1 p=0.3; n=40; sim=1e4; phat=binornd(n,p,[sim,1])/n; hist(phat)
```

The following code is a little fancier. It makes use of Matlab's `tabulate` function, discussed in Section 2.1.4.

```
1 p=0.3; n=40; sim=1e4; phat=binornd(n,p,[sim,1])/n;
2 nbins=length(tabulate(phat)); [histcount, histgrd] =hist(phat,nbins);
3 h1=bar(histgrd,histcount); xlim([0 0.8])
4 set(h1,'facecolor',[0.64 0 0.24],'edgecolor',[0 0 0],'linewidth',2)
5 set(gca,'fontsize',16), title(['Using n=',int2str(n)])
```

Doing this with $p = 0.3$ and for sample sizes $n = 20$ and $n = 40$ yields the histograms shown in Figure 1.4. We see that, while the mode of $\hat{p}$ is at 0.3 in both cases, there is quite some variation around this value and, particularly for $n = 20$, a small but nonnegligible chance (the exact probability of which you can easily calculate) that $\hat{p}$ is zero or higher than 0.6.

We first state some useful definitions, and then, based on our ability to easily simulate values of $\hat{p}$, determine how to form a c.i. for p.

Consider a distribution (or statistical model) with unknown but fixed k-dimensional parameter $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subseteq \mathbb{R}^k$. A **confidence set** $M(\mathbf{X}) \subset \boldsymbol{\Theta}$ for $\boldsymbol{\theta}$ with **confidence level** $1 - \alpha$ is any set such that

$$\Pr(\boldsymbol{\theta} \in M(\mathbf{X})) \geq 1 - \alpha, \quad \forall\, \boldsymbol{\theta} \in \boldsymbol{\Theta}, \quad 0 < \alpha < 1, \tag{1.4}$$

where $M(\mathbf{X})$ depends on the r.v. $\mathbf{X}$, a realization of which will be observed, but does not depend on the unknown parameter $\boldsymbol{\theta}$. Typical values of α are 0.01, 0.05 and 0.10. The quantity $\Pr(\boldsymbol{\theta} \in M(\mathbf{X}))$ is called the **coverage probability** and can depend on $\boldsymbol{\theta}$; its greatest lower bound

$$\inf_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \Pr(\boldsymbol{\theta} \in M(\mathbf{X}))$$

is referred to as the **confidence coefficient** of $M(\mathbf{X})$.

It is imperative to keep in mind how (1.4) is to be understood: As $\boldsymbol{\theta}$ is fixed and $M(\mathbf{X})$ is random, we say that, before the sample is collected, the set $M(\mathbf{X})$ will contain (or capture)

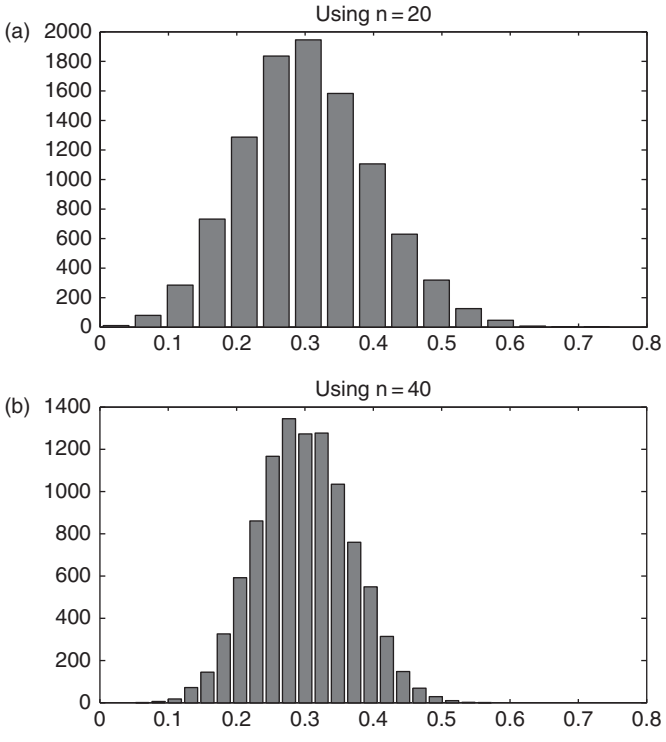

Figure 1.4 *Simulations of $\hat{p} = S/n$ for $S = \sum_{i=1}^{n} X_i$, $X_i \overset{\text{i.i.d.}}{\sim} \text{Bern}(p)$, for $p = 0.3$ and $n = 20$ (a) and $n = 40$ (b), based on 10,000 replications.*

the true $\boldsymbol{\theta}$ with probability at least $1 - \alpha$. Once sample $\mathbf{x}$ is observed and $M(\mathbf{x})$ is computed, it either contains $\boldsymbol{\theta}$ or not. For small values of α, with 0.10, 0.05 and 0.01 being typical in practice, we might be quite confident that $\boldsymbol{\theta} \in M(\mathbf{x})$, but it no longer makes sense to speak of the probability of it being so.

Let the dimension of $\boldsymbol{\theta}$ be $k = 1$, as in the Bernoulli case. In most (but not all) situations with $k = 1$, $M(\mathbf{X})$ will be an interval: denoting the left and right endpoints as $\underline{\theta} = \underline{\theta}(\mathbf{X})$ and $\bar{\theta} = \bar{\theta}(\mathbf{X})$, respectively, $M(\mathbf{X}) = (\underline{\theta}, \bar{\theta})$ is referred to a **confidence interval**, or **c.i.**, for θ with **confidence level** $1 - \alpha$ or, more commonly, a $100(1 - \alpha)\%$ c.i. for θ. It also makes sense to refer to a c.i. as an **interval estimator** of θ, which draws attention to its purpose in comparison to that of a point estimator.

To compute (say) a 95% c.i. for p in the i.i.d. Bernoulli model, a starting point would be to consider the 2.5% and 97.5% quantiles of the simulated $\hat{p}$-Values shown in Figure 1.4. These give us an interval in which $\hat{p}$ falls with 95% probability when the true p is 0.3. This is related, but not equivalent, to what we want: an interval $M(\mathbf{X}) = (\underline{p}, \bar{p})$ such that $\Pr(p \in M) = 0.95$ for all $p \in (0, 1)$. The first problem we have is that the aforementioned quantiles cannot be computed from data because we do not know the true value of p.

To address this problem, let us consider doing the following: From our data set of n i.i.d. Bern (p) realizations, we compute $\hat{p}_{\text{data}} = s/n$, where s is the sum of the observations, and, using it as a best guess for the unknown parameter p, simulate realizations of $\hat{p}_i = S_i/n$, $i = 1, \ldots, B$, each based on n i.i.d. Bern$(\hat{p}_{\text{data}})$ realizations, where B is a large number, say

```
n=20; p=0.3; B=1e4; alpha=0.05; sim=1e5; bool=zeros(sim,1);
for i=1:sim
  phat0=binornd(n,p,[1,1])/n; % the estimate of p from Bin(n,p) data
  phatvec=binornd(n,phat0,[B,1])/n; % B samples of S/n, S~Bin(n,phat0)
  ci=quantile(phatvec,[alpha/2 1-alpha/2]); low=ci(1); high=ci(2);
  bool(i) = (p>low) & (p<high); % is the true p in the interval?
end
actualcoverage=mean(bool)
```

Program Listing 1.3: Determines the actual coverage of the nominal 95% parametric single bootstrap c.i. for the Bernoulli model parameter p.

10,000. The 2.5th and 97.5th sample percentiles of these $\hat{p}_i$ are then computed. (This is referred to as a *percentile (single parametric) bootstrap c.i. method*, as will be explained soon below.)

While this is indeed some kind of c.i. for p, we have our second problem: It is likely, if not nearly certain, that this interval does not have the correct (in this case, 95%) coverage probability, because $\hat{p} \neq p$. To determine the **actual coverage probability** corresponding to the **nominal confidence level** of (in this case) $\alpha = 0.05$, we can "simulate the simulation," that is, repeat the aforementioned simulation of the B values of $\hat{p}_i$ for a given value of $\hat{p}_{\text{data}}$, for many draws of $\hat{p}_{\text{data}}$, all based on the same underlying value of p. The code in Listing 1.3 illustrates how to do this for $n = 20$, $p = 0.3$, and using `sim` = 100,000 replications.

The output is 0.844. Again, this is the **actual coverage probability** corresponding to the **nominal coverage probability** (confidence level) of 0.95. We can envision a function $s : (0, 1) \to (0, 1)$ mapping the nominal to actual coverage, one point of which is $0.844 = s(0.95)$. By repeating this exercise over a grid of nominal coverage probabilities, we obtain an approximation of function s (via, say, linear interpolation), and compute $\alpha_{\text{act}} = 1 - s(1 - \alpha_{\text{nom}})$. The code required for computing and plotting several values of s is shown in Listing 1.4. Based on this, we can approximate the nominal coverage probability that yields an actual one of 0.95, that is, we want $s^{-1}(0.95)$.

This mapping s, computed for $p = 0.3$ and $n = 20$, but also for $n = 40$, 80, and 1000, is shown in Figure 1.5(a). As n increases, the actual level approaches the nominal level.

```
n=20; p=0.3; B=1e4; nominal=0.90:0.002:0.998; sim=1e5;
nomlen=length(nominal); bool=zeros(sim,nomlen);
for i=1:sim
  phat=binornd(n,p,[1,1])/n; art=binornd(n,phat,[B,1])/n;
  for j=1:nomlen
    alpha=1-nominal(j); ci=quantile(art,[alpha/2 1-alpha/2]);
    bool(i,j)=(p>ci(1)) & (p<ci(2));
  end
end
actual20=mean(bool); plot(nominal,actual20,'r-','linewidth',2)
set(gca,'fontsize',16), grid, xlabel('Nominal'), ylabel('Actual')
title(['Actual Coverage Probability for p=',num2str(p)])
```

Program Listing 1.4: Computes and plots the mapping s of the actual coverage probability in the Bernoulli model, as a function of the nominal coverage probability, based on single parametric bootstrap c.i.s.

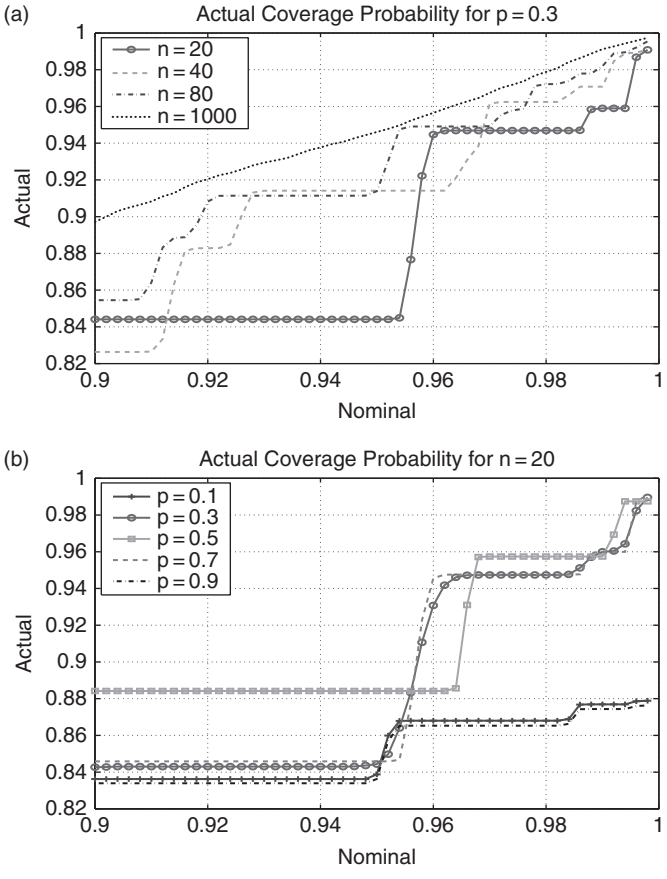

Figure 1.5 *Mapping s between nominal and actual coverage probabilities for c.i.s of the success parameter in the i.i.d. Bernoulli model, based on the (single) parametric bootstrap, each computed via simulation with 100,000 replications.*

From the graph for $p = 0.3$ and $n = 20$, we see that a nominal coverage level of $\alpha_{\text{nom}} \approx 0.015$ corresponds to an actual coverage level of $\alpha_{\text{act}} = 0.05$.

Note that, because of the discreteness of $\hat{p}$, the graphs for the smaller sample sizes are constant over certain ranges of the nominal values, and so s is not a bijection. For a specific value of α_{act}, say 0.05, we would choose α_{nom} to be the leftmost point along the graph such that the actual coverage probability is at least 0.95, to ensure that the resulting c.i. is the shortest possible one that still maintains the desired coverage. While this is graphically simple to see, such results are often expressed algebraically; for a given value of α_{act}, some thought shows that we should take

$$\alpha_{\text{nom}} = 1 - \inf(p \in (0,1) \mid s(p) \geq 1 - \alpha_{\text{act}}) = \sup(p \in (0,1) \mid s(1-p) \geq 1 - \alpha_{\text{act}}).$$

From Figure 1.5(a), we see that the correct nominal value depends on the sample size n. Ideally, for a fixed sample size, the correct value of α_{nom} would not depend on the true p, so that the set of values $\alpha_{\text{nom}}(n)$ could be computed for various sample sizes "once and for all" for a given α_{act}. Figure 1.5(b) shows that this is unfortunately not the case; for a fixed value

of n (here 20), we see that they depend strongly on the true p (though it appears that the behavior for p and $1-p$ is the same). Thus, for a given data set of length n and observed $\hat{p}$, the simulation exercise above would need to be conducted to get the correct nominal coverage level, and then the c.i. for p could be delivered.

The method just described is an example of a very general and powerful technique for constructing c.i.s, called the **parametric bootstrap**. It serves as an introduction to the more popular, but somewhat less obvious, *nonparametric* bootstrap, which is discussed in the next section.

Before proceeding, we make two important remarks.

Remarks

(a) For this simple model, there exists an analytic method for constructing a c.i. for p (see Section 1.3.4 for demonstration and comparison, and Section 8.4.2 for development of the theory), thus obviating the need for simulation. The analytic method also delivers c.i.s that are shorter, on average, than the simulation method just described. For example, with $n = 20$, $p = 0.3$, and using $\alpha_{act} = 0.05$ (i.e., for 95% c.i.s with correct coverage probability), the simulation method yields an average c.i. length of 0.46, while the method developed in Section 8.4.2 yields an average c.i. length of 0.38. Thus, the above simulation method should not actually be used in practice for c.i.s for p in the Bernoulli model, and just serves as an introduction to the method in a simple setting. The real value of the parametric and nonparametric bootstrap methods arises in more complicated model settings for which analytical results are not available.

(b) The reader who has taken an introductory undergraduate course in statistics surely recalls the usual, simple, asymptotically valid c.i. formula for p given by

$$\hat{p} \pm z\widehat{V}^{1/2}, \quad \widehat{V} = \frac{\hat{p}(1-\hat{p})}{n},$$

recalling (1.1), where z is the upper $1-\alpha/2$ quantile of the standard normal distribution, such as 1.96 when $\alpha = 0.05$. This is referred to as a Wald interval, more details of which are given later; see (3.45). What might come as a shock and surprise is that this ubiquitous result turns out to behave rather poorly and erratically, even for reasonably large sample sizes, with the actual coverage potentially changing rather substantially when, for example, n is increased or decreased by one. A detailed discussion, and several alternative (non-bootstrap-based) intervals are provided in Brown et al. (2001). As such, the subsequent development of a bootstrap-based confidence interval for p, while designed for teaching the underlying concepts of bootstrap methodology using a simple example, could also be used in practice.

There are other approaches for addressing the problems that arise in the actual coverage of confidence intervals associated with discrete models. In particular, we recommend the use of the so-called **mid-p-values**; see Agresti (1992), Hwang and Yang (2001), and Agresti and Gottard (2005), as well as Butler (2007, Sec. 6.1.4). ■

1.3 INTERVAL ESTIMATION VIA THE BOOTSTRAP

In the previous section, we used simulated Bernoulli r.v.s with success probability taken to be the point estimate of p from the observed data for constructing a confidence interval. A related way is to simulate not from this distribution, but rather, somewhat perversely, from the actual observed data. This is called the **nonparametric bootstrap**. The idea is to *treat the n observed data points as the entire underlying population*, and draw many n-length samples, or **resample** from it. There is a fundamental advantage to using this instead of the parametric one, as we will discuss below. We denote the number of resamples by B.

1.3.1 Computation and Comparison with Parametric Bootstrap

It is important to emphasize that each of the B samples for the nonparametric bootstrap is drawn *with replacement* from the observed data set. The reason is that our observed data set is being treated as the true underlying population, and thus an i.i.d. sample from it entails drawing n values from this population such that each draw is independent of the others and each of the n values in the population has an equal chance of being drawn. (If we draw n observations without replacement, we obtain exactly the original data set, just in permuted order.)

The total number of different unordered samples that can be drawn, with replacement, from a set of n observations, is $\binom{2n-1}{n}$ (see Section I.2.1 for derivation). This number quickly becomes astronomically large as n grows, already being over 90,000 for $n = 10$. If one were actually to use a complete enumeration of all these data sets, weighting them appropriately from their multinomial probabilities $n!/(n_1!n_2!\cdots n_n!\, n^n)$, where n_i denotes the number of times the ith observation occurs in the resample (see Section I.5.3.1), then it would yield what is called the **exact bootstrap distribution**. For n larger than about 20, such an enumeration is neither feasible in practice nor necessary. (See Diaconis and Holmes, 1994, and the references therein for details on the method of enumeration.) By randomly choosing a large number B of resamples, the exact bootstrap distribution can be adequately approximated.

The key to drawing with replacement is to generate n discrete r.v.s that have equal probability $1/n$ on the integers $1, 2, \ldots, n$. This is conveniently built into Matlab as function `unidrnd` (or `randi`). These then serve as indices into the array of original data. For example, the following code takes an i.i.d. Bernoulli data set and generates a single bootstrap sample `bsamp`:

```
1 n=100; p=0.3; data=binornd(1,p,[n,1]);
2 ind=unidrnd(n,[n 1]); bsamp=data(ind);
```

To help clarify the technique further, imagine you obtain a data set of $n = 100$ i.i.d. Bern(p) observations, with, say, 32 ones and 68 zeros, so that $\hat{p} = 0.32$. The parametric bootstrap can then be used to approximate the sampling distribution of $\hat{p}$ by computing B samples of $\hat{p}$, each based on n observations from a Bern(0.32) distribution. Likewise, the nonparametric bootstrap can be used to approximate the distribution of $\hat{p}$ by computing B samples of $\hat{p}$, each based on a resample of the actual data set. It should be clear from the simple structure of the model that, *in this case, the parametric and nonparametric distributions are theoretically identical*. (Of course, for finite B, they will not be numerically equal.)

```
n=100; p=0.3; data=binornd(1,p,[n 1]); % true data are Bern(p)
phat=mean(data) % point estimator of p, for use with parametric boot:
B=1e5; phatpara=sum(binornd(1,phat,[n B]))/n;
figure, hist(phatpara,5000), xlim([0.1 0.5])
phatnonpara=zeros(B,1);
for i=1:B % compute the nonparametric bootstrap distribution
  ind=unidrnd(n,[n 1]); bsamp=data(ind); phatnonpara(i)=mean(bsamp);
end
figure, hist(phatnonpara,5000), xlim([0.1 0.5])
```

Program Listing 1.5: Compares the parametric and nonparametric bootstraps.

As further practice in programming the bootstrap, and also serving to confirm the equality of the parametric and nonparametric bootstraps in the Bernoulli model case, the code in Listing 1.5 should be studied and run, and the resulting histograms compared.

This implies, for example, that c.i.s based on the parametric and nonparametric bootstraps will have identical nominal coverage properties, and thus there is no need to perform the simulations for generating the plots in Figure 1.5 with the nonparametric bootstrap in order to compare their performance.

The equality of the parametric and nonparametric bootstrap distributions in this example is special for the Bernoulli (and, more generally, for the multinomial) distribution. The result does not hold for distributions with infinite countable support (geometric, Poisson, etc.) or uncountable support. Using the geometric as an example, one might imagine that the parametric bootstrap should be superior to the nonparametric bootstrap, because the point estimator of the geometric success probability parameter p contains all the information in the actual data set (we will qualify this notion in Section 7.3 with the concept of **sufficiency**) and thus samples drawn from a Geo($\hat{p}$) distribution will be of more value than resampled ones associated with the nonparametric bootstrap, which have their support limited to what happened to have been observed in the actual data set.

This conjecture is indeed true, and is demonstrated below in Section 1.4; though for large sample sizes (where "large" will depend on the model), the difference will be negligible. More importantly, however, the above reasoning is only valid *if you are sure about the parametric model that generated the data*. Rarely in real applications can one make such strong assumptions, and this is the reason why the nonparametric bootstrap is more often used; this point is illustrated in Section 2.2.

Remarks

(a) Two of the necessary conditions such that the nonparametric bootstrap leads to asymptotically correct inferential procedures are stated in Section 2.1.

(b) The term *bootstrap* was coined by one of the pioneers of the method, Bradley Efron, in the late 1970s. Via its analogy to a literary reference in which the main character, after falling into a lake, pulls himself out by his own shoelaces (bootstraps), the name indicates the self-referencing nature of the method. There are several textbooks that detail the theory, importance, and wide applicability of bootstrap, resampling, and so-called subsampling methods (as well as situations in which they do not work, and some possible solutions); an excellent starting point is Efron and Tibshirani

(1993), while a more advanced but still accessible and highly regarded treatment is given in Davison and Hinkley (1997). For emphasis on subsampling, see Politis et al. (1999).

(c) The method we show for computing bootstrap c.i.s is just one of several, and is referred to as the **percentile bootstrap method**. It is among the most intuitive methods, though not necessarily the most accurate or fastest (in terms of number of resamples required). In addition to the aforementioned references, see Efron (2003), Davison et al. (2003) and Efron and Hastie (2016, Ch. 11) for details on the other methods of bootstrap c.i. construction. ■

1.3.2 Application to Bernoulli Model and Modification

Listing 1.6 shows how to simulate the actual coverage properties of the method, using a grid of values of parameter p. Observe that, by changing one line in the code, we can switch between the parametric and nonparametric bootstrap; though in this case, as discussed above, they are equivalent (and the faster method should then be chosen).

For each of four sample sizes n, we use it with 10,000 replications (passed as `sim`) and $B = 10{,}000$ (called `B1` in the program) bootstrap resamples, to determine the actual coverage of the nominal 90% c.i.s, as a function of p. The results are shown in Figure 1.6(a). The actual coverage approaches 90% as the sample size increases, and is far below it for small sample sizes and extreme (close to zero or one) values of p. This latter artifact is easily explained: For small n and extreme, say small, p, there is a substantial chance that the data

```
1  function actual90 = bernoulliCIsingleboot(n,sim)
2  pvec=0.05:0.05:0.95; plen=length(pvec); actual90=zeros(plen,1);
3  bool0=zeros(sim,1); alpha=0.10; B1=1e4; bootphat=zeros(B1,1);
4  for ploop=1:plen
5    p=pvec(ploop), n
6    for i=1:sim
7      rand('twister',i) % data sets change 'smoothly' w.r.t. p
8      data=binornd(1,p,[n,1]); phat=mean(data);
9      for b1=1:B1
10       ind=unidrnd(n,[n,1]); bootsamp=data(ind); % nonparametric bootstrap
11     % bootsamp = binornd(1,phat,[n,1]);       %    parametric bootstrap
12       bootphat(b1)=mean(bootsamp);
13     end
14     ci=quantile(bootphat,[alpha/2 1-alpha/2]);
15     bool0(i) = (p>ci(1)) & (p<ci(2));
16   end
17   actual90(ploop)=mean(bool0);
18 end
```

Program Listing 1.6: Implementation of (single) parametric and nonparametric bootstrap for computing a c.i. in the Bernoulli model. As shown here, line 10 generates a resample from the data, and is thus using the nonparametric bootstrap. Line 11 (commented out) can be invoked instead to use the parametric bootstrap, in which case the code accomplishes the same thing as done in Listing 1.4 (with `B1` here set to 10,000, and using `nominal` equal to 0.90).

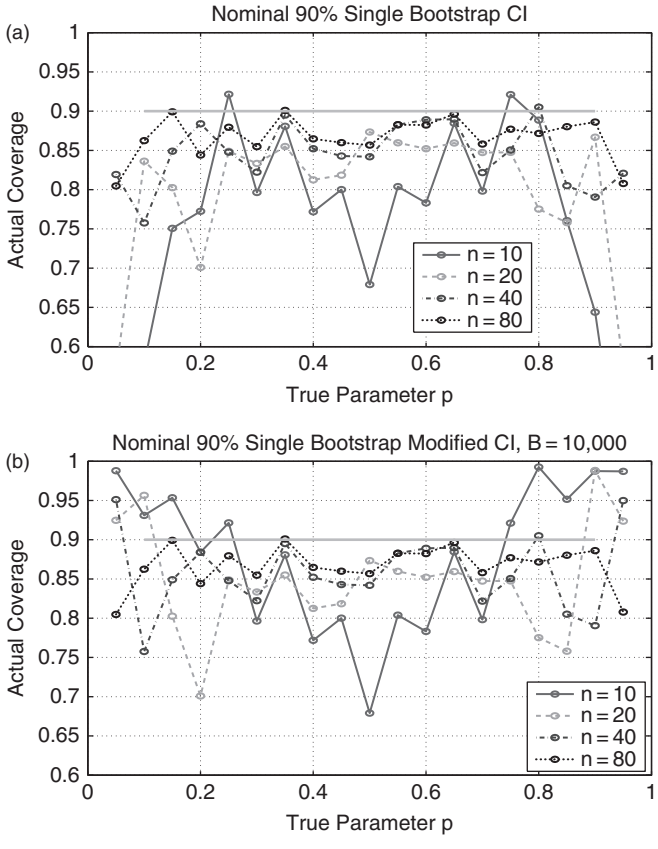

Figure 1.6 *(a) Actual coverage, based on simulation with 10,000 replications, of nominal 90% c.i.s using the (single) nonparametric bootstrap (with* `B1` *= 10,000). Graph is truncated at 0.6, with the actual coverage for* $n = 10$ *and* $p = 0.05$ *and* $p = 0.95$ *being about 0.4. (b) Same but using the modified c.i. in (1.5) and (1.6).*

set will consist of all zeros. If this happens, then clearly, both the parametric and nonparametric method will deliver a degenerate c.i. of the single point zero. This is unsatisfactory, given that, for small n and p, getting all zeros is not improbable.

A simple and appealing solution is to take a c.i. with lower bound zero and upper bound given by the smallest value of p such that the probability of getting n out of n zeros is less than or equal to the chosen value α_{act}. That is, with $S_n = \sum_{i=1}^{n} X_i \sim \text{Bin}(n, p)$, we suggest taking

$$\underline{p} = 0, \quad \bar{p} = \inf(p \in (0,1) \mid \Pr(S_n = 0) \leq \alpha). \tag{1.5}$$

As this probability is a continuous function of p for a given n, it can be computed in Matlab as a solution of one equation in one unknown. Impressively, this can be accomplished with just one line of code (see the Matlab help file):

```
1 fzero(@(p) binopdf(0,n,p)-alpha,[1e-6 1-1e-6])
```

provided `alpha` is defined. A similar procedure yields the c.i. in the event that all the n observations are ones as

$$\underline{p} = \sup(p \in (0,1) \mid \Pr(S_n = n) \leq \alpha), \quad \bar{p} = 1. \tag{1.6}$$

When the data set does not consist of all zeros or ones, we proceed as before, with either the nonparametric or parametric bootstrap, and refer to the resulting c.i. as the **modified** c.i. for the Bernoulli parameter p.

Listing 1.7 gives the code for its implementation, and Figure 1.6(b) shows the actual coverage results with its use, again based on `sim` = 10,000 replications and $B = 10{,}000$ bootstrap samples. We see that the problem in the extreme cases has indeed been solved, with the resulting intervals for small n and extreme p being a bit too conservative (with higher actual than nominal coverage). For example, with $n = 10$, if all the observations in the data sample are zero, then we obtain, for $\alpha = 0.10$, $\bar{p} = 0.206$. Thus, the 90% c.i. will contain the true value of p whenever p is less than this value, for example, for $p = 0.05, 0.10, 0.15$ and 0.20, which are precisely those values in the graph that previously (Figure 1.6a) had lower actual coverage and now (Figure 1.6b) have higher actual coverage.

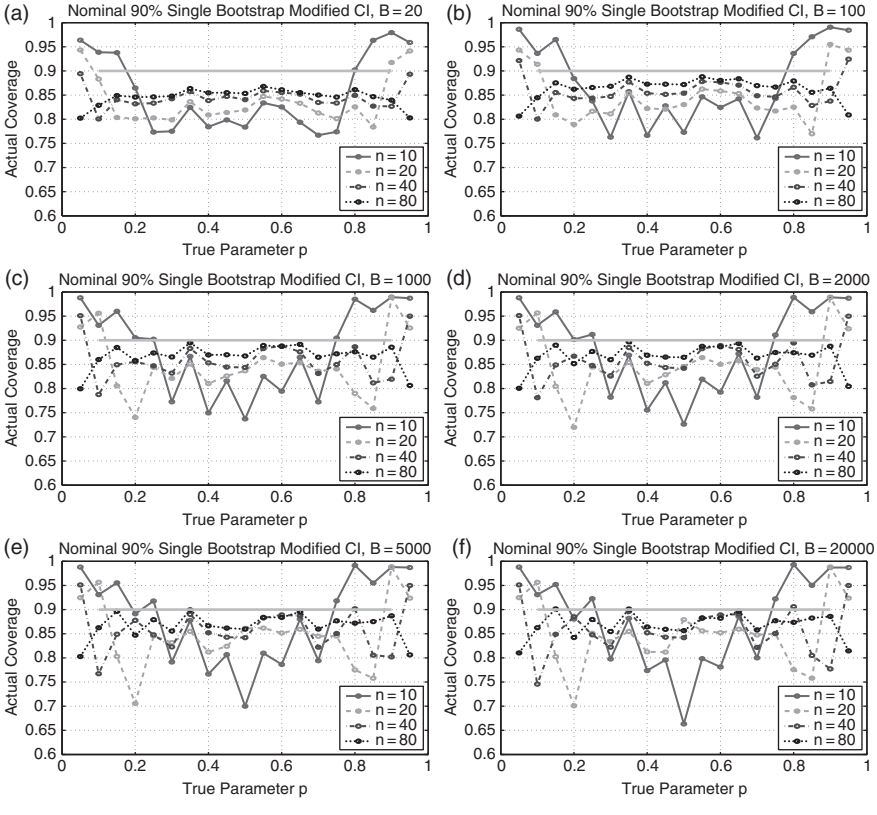

Figure 1.7 *Same as Figure 1.6(b), but using different numbers of bootstrap replications.*

Up to this point, we have used $B = 10{,}000$ resamples, this being a rather arbitrary choice that we hope is high enough such that the results are close to what would be obtained from the (unattainable) exact bootstrap distribution. A simple and intuitive way to heuristically determine if the choice of B is adequate is to use different choices of B, increasing it until the results no longer substantially change, where "substantial" is a relative term reflecting the desired precision. It is imperative here that each time the bootstrap is used (with the same or different B), the seed value is changed, so that a different sequence of draws with replacement is conducted. If this is not done, then, when changing B from, say, 900 to 1000, 90% of the draws will be the same in each set, so that the resulting object of interest (c.i. coverage probability, a standard error, a histogram of the approximate sampling distribution, etc.) will indeed look similar across both bootstrap runs, but not imply that $B = 1000$ is adequate.

As an illustration, Figure 1.7(a–f) is the same as in Figure 1.6(b), but based on different numbers of bootstrap replications (but still with `sim` = 10,000). Figure 1.7(a) shows the case with only $B = 20$, and we see, perhaps surprisingly, that the actual coverage compared to the $B = 10{,}000$ case does not suffer much (and in a couple of cases, it is actually better). Figure 1.7(f) uses $B = 20{,}000$. The only discernable difference with the $B = 10{,}000$ case appears to be for $n = 10$, $p = 0.5$, and it is small; otherwise, they are identical, indicating that $B = 10{,}000$ is enough. In fact, if we can ignore the $n = 10$, $p = 0.5$ case, $B = 2000$ appears to be adequate.

```
function actual90 = bernoulliCIsingleboot(n,sim)
pvec=0.05:0.05:0.95; plen=length(pvec); actual90=zeros(plen,1);
bool0=zeros(sim,1); alpha=0.10; B1=1e4; bootphat=zeros(B1,1);
for ploop=1:plen
  p=pvec(ploop), n
  for i=1:sim
    rand('twister',i) % data sets change 'smoothly' w.r.t. p
    data=binornd(1,p,[n,1]); phat=mean(data);
    if phat==1
      ci(1)=fzero(@(p) binopdf(n,n,p)-alpha,[1e-6 1-1e-6]); ci(2)=1;
    elseif phat==0
      ci(1)=0; ci(2)=fzero(@(p) binopdf(0,n,p)-alpha,[1e-6 1-1e-6]);
    else
      for b1=1:B1
        ind=unidrnd(n,[n,1]); bootsamp=data(ind); % nonpara boot
        % bootsamp = binornd(1,phat,[n,1]);       %    para boot
        bootphat(b1)=mean(bootsamp);
      end
      ci=quantile(bootphat,[alpha/2 1-alpha/2]);
    end
    bool0(i) = (p>ci(1)) & (p<ci(2));
  end
  actual90(ploop)=mean(bool0);
end
```

Program Listing 1.7: Same as the program in Listing 1.6, but treats the cases for which the sample data are all zeros, or all ones, in the modified fashion from (1.5) and (1.6).

1.3.3 Double Bootstrap

> The widespread availability of fast cheap computers has made [the bootstrap] a practical alternative to analytical calculation in many problems, because computer time is increasingly plentiful relative to the number of hours in a researcher's day.
>
> (Davison and Hinkley, 1997, p. 59)

Recall from Figure 1.5 that the actual coverage of the bootstrap (both parametric and nonparametric; they are identical in this case) can deviate substantially from the nominal coverage probability, becoming more acute as the sample size n decreases, and depending on the true parameter p. If we knew the true p, then we could just use simulation, as done to obtain those plots, to determine the mapping from nominal to actual coverage, and deliver a more accurate c.i. Of course, if we knew the true p, we would not have to bother with this exercise at all! The interesting question is how we can optimally choose the nominal coverage of the bootstrap c.i. *given only our data set*, and not the true p.

The answer is analogous to what we did above when we wished to assess the actual coverage: Just as we "simulated the simulation" there, we will *apply the bootstrap to the bootstrap* here. That is, onto each resampled bootstrap data set (referred to as an iteration of the *outer bootstrap loop*), we conduct the bootstrap procedure (called the *inner bootstrap loop*), for a range of nominal coverage probabilities, and keep track of the actual coverage. Then, for the actual data set, we use the nominal coverage that gives rise to the desired actual coverage. This is referred to as the **nested bootstrap** or **double bootstrap**. We use the convention that the outer bootstrap uses B_1 resamples, and each inner bootstrap uses B_2 resamples. Thus, $B_1 B_2$ resamples are needed in total. Pseudo-code Listing 1.1 gives the algorithm for the double bootstrap for parameter θ.

(1) From the data set under study, $\mathbf{y}^{\text{obs}}$, compute the estimate of parameter θ, say $\hat{\theta}^{\text{obs}}$ with a chosen method of estimation that is consistent, e.g., maximum likelihood, and where obs stands for "observed."
(2) FOR $b_1 = 1, \ldots, B_1$ DO
  a. Generate a resample of $\mathbf{y}^{\text{obs}}$, say $\mathbf{y}^{(b_1)}$ (or, for the parametric bootstrap, simulate data set $\mathbf{y}^{(b_1)}$ according to the presumed model and with parameter $\hat{\theta}^{\text{obs}}$).
  b. Compute $\hat{\theta}^{(b_1)}$ for data set $\mathbf{y}^{(b_1)}$ (using the chosen method of estimation).
  c. FOR $b_2 = 1, \ldots, B_2$ DO
    (i) Generate a resample of $\mathbf{y}^{(b_1)}$, say $\mathbf{y}^{(b_1,b_2)}$ (for the parametric bootstrap, simulate data set $\mathbf{y}^{(b_1,b_2)}$ according to the presumed model and with parameter $\hat{\theta}^{(b_1)}$).
    (ii) Compute $\hat{\theta}^{(b_1,b_2)}$ for data set $\mathbf{y}^{(b_1,b_2)}$ (using the same chosen method of estimation).
  d. FOR j over each nominal coverage probability in a grid of values, say $0.799, 0.801, \ldots, 0.999$, DO

(i) Compute the c.i. $\text{ci}^{(b_1,j)}$ as the corresponding lower and upper quantiles from the $(\hat{\theta}^{(b_1,1)}, \ldots, \hat{\theta}^{(b_1,B_2)})$ values.

(ii) Record a one in the b_1th row and jth column of matrix `bool` if $\text{ci}^{(b_1,j)}$ contains $\hat{\theta}^{\text{obs}}$.

(3) Compute the average of each column of matrix `bool` to give a vector of actual coverage probabilities corresponding to the vector of nominal coverage probabilities $0.799, 0.801, \ldots, 0.999$.

(4) Use the previous two vectors and linear interpolation to get the nominal level of coverage, say $1 - \alpha^*$, corresponding to an actual coverage probability of 90%.

(5) Deliver the c.i. for the actual data set $\mathbf{y}^{\text{obs}}$ as the $\alpha^*/2$ and $1 - \alpha^*/2$ quantiles of the outer bootstrap parameter values $(\hat{\theta}^{(1)}, \ldots, \hat{\theta}^{(B_1)})$

Pseudo-code Listing 1.1: Algorithm for the double bootstrap. See Listing 1.8 for the associated Matlab program.

Listing 1.8 gives a program that implements the pseudo-code, and also simulates the double bootstrap to determine the actual coverage for a nominal coverage of 90%. It contains four nested FOR loops: The first is over a grid of none values $(0.1, 0.2, \ldots, 0.9)$ of the parameter p, and the second conducts, for each given value of p, a simulation of `sim` data sets, and keeps track of whether or not its c.i. covers the true value of p. The inner two FOR loops conduct the double nonparametric bootstrap for each given data set.

The execution time for computing a double bootstrap c.i. for a given data set will clearly be far longer than that of a single bootstrap. For the Bernoulli model, we require about 50 seconds (roughly irrespective of n for $20 \leq n \leq 80$) on (one core of) a 3 GHz PC when using (only) $B_1 = B_2 = 1000$. Based on these values for B_1 and B_2, the simulation study using nine values of p and `sim` $= 1000$ simulations for each value of p requires over 5 days of computing, for each sample size n.

The results are shown in Figure 1.8(a) . The actual coverage for $n = 80$ and $p \in [0.1, 0.9]$ (and, in general, for larger n, and p close to 0.5) is quite close to the nominal value of 0.90 and better than the single bootstrap. However, the actual coverage breaks down as the sample size decreases and p moves away from 0.5, worse in fact than occurs for the single bootstrap. The reason, and the solution, are the same as discussed above.

Implementing (1.5) and (1.6) for each of the `sim` data sets, and also for the draws in the outer bootstrap (see Problem 1.2), yields the modified double bootstrap c.i.s for p. The results are shown in Figure 1.8(b). The modification via (1.5) and (1.6) clearly has helped, though compared to the coverage of the modified c.i.s using just the single bootstrap (Figure 1.6(b)), the improvement is not spectacular. This is presumably due in part to having used only $B_1 = B_2 = 1000$ bootstrap resamples (and `sim` $= 1000$ replications), whereas we used $B = 10{,}000$ (and `sim` $= 10{,}000$) for the single bootstrap. To be sure, we would have to rerun the calculations using larger values; though with $B_1 = B_2 = 10{,}000$, the calculation will take about 100 times longer, or 83 minutes per c.i. Doing this for nine values of p and, say, 10,000 replications would take over 14 years using a single core processor – for each sample size n. The next section presents a way around this problem.

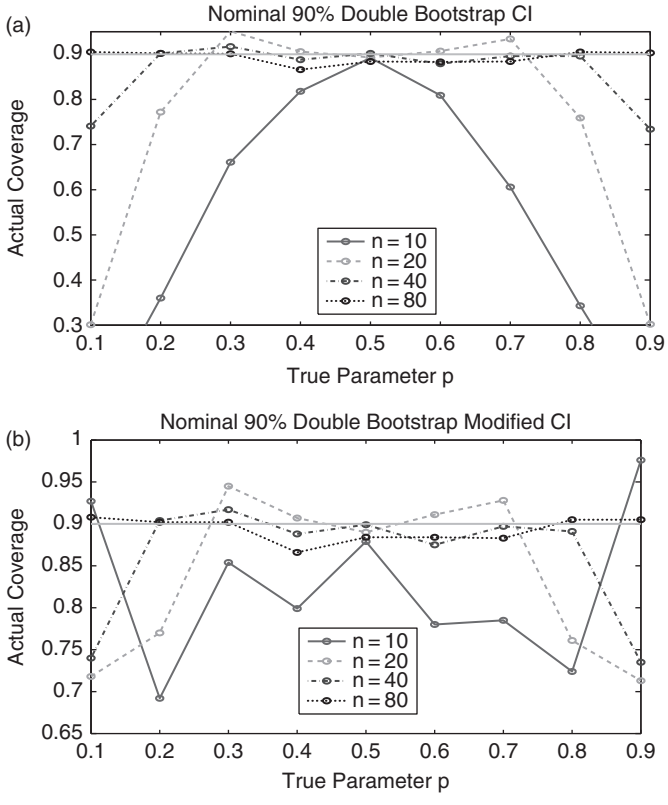

Figure 1.8 *(a) Actual coverage of nominal 90% c.i.s using the double bootstrap (truncated at 0.3), based on 1000 replications. (b) Same but using the modified c.i. in (1.5) and (1.6) applied to each simulated data set and to each bootstrap sample in the outer bootstrap loop.*

1.3.4 Double Bootstrap with Analytic Inner Loop

The previous calculation of computation time shows that simulating the performance of the double bootstrap c.i. over a grid of parameter values p, for several sample sizes, using relatively large values of B_1 and B_2, becomes burdensome, if not infeasible.

As already mentioned at the end of Section 1.2, there exists an analytic method for constructing a c.i. for p. Because of the discreteness of the data, it also does not have exactly correct actual coverage. However, given its speed of calculation, we could use it within a double bootstrap calculation, replacing the inner bootstrap loop with the analytic method. This will yield substantial time savings, because we avoid the B_2 resampling operations for each outer loop iteration, and also avoid the B_2 computations of the parameter estimator. Of course, in our case studied here, the estimator is just the sample mean of n Bernoulli realizations, and so is essentially instantaneously calculated. In general however, we might be using an m.l.e., obtained using multivariate numerical optimization methods, and thus taking vastly longer to compute.

The idea of replacing the inner bootstrap loop with an analytic result or (often a saddlepoint) approximation is very common; see, for example, the discussion in Davison

```
1  function actual90 = bernoulliCIdoubleboot(n,sim)
2  truenominal=0.90; % desired coverage probability;
3   %     change or pass as a parameter to the function
4  pvec=0.1:0.1:0.9; plen=length(pvec); actual90=zeros(plen,1);
5  B1=1e3; b1phat=zeros(B1,1); B2=1e3; b2phat=zeros(B2,1);
6  nominal=0.799:0.002:0.999; nomlen=length(nominal); bool0=zeros(sim,1);
7  bool1=zeros(B1,nomlen);
8  for ploop=1:plen   , p=pvec(ploop);
9    for i=1:sim     , i, p, n
10     rand('twister',i), data=binornd(1,p,[n,1]); % 'actual' data
11     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
12     phatdata=mean(data);
13     for b1=1:B1   %%%%%% outer bootstrap loop
14       ind=unidrnd(n,[n,1]); b1samp=data(ind); b1phat(b1)=mean(b1samp);
15       for b2=1:B2 %%%%%% inner bootstrap loop
16         ind=unidrnd(n,[n,1]); b2samp=b1samp(ind); b2phat(b2)=mean(b2samp);
17       end
18       for j=1:nomlen
19         alpha=1-nominal(j); ci=quantile(b2phat,[alpha/2 1-alpha/2]);
20         bool1(b1,j) = (phatdata>ci(1)) & (phatdata<ci(2));
21       end
22     end
23     bootactual=mean(bool1)+cumsum((1:nomlen)/1e10);
24     boot90=interp1(bootactual,nominal,truenominal);
25     alphanom=1-boot90; ci=quantile(b1phat,[alphanom/2,1-alphanom/2]);
26     % 'actual' CI
27     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28     bool0(i) = (p>ci(1)) & (p<ci(2));
29   end
30   actual90(ploop)=mean(bool0); eval(['save c:\bernoulliCIdoublebootn ',
31   int2str(n)])
32 end
```

Program Listing 1.8: Simulates, over a grid of values of p, the double nonparametric bootstrap for computing a 90% c.i. for parameter p in the Bernoulli model, to determine the actual coverage corresponding to a nominal coverage of 90%. Change variable `truenominal` to choose a different nominal coverage value. The algorithm for the double bootstrap is just the code between the two long comment lines; the rest is for the simulation. In line 21, observe that we add `cumsum((1:nomlen)/1e10)` to the empirical coverage values to force them to be increasing, so that the `interp1` command in the next line does not fail.

and Hinkley (1997) and the references therein, and Butler and Paolella (2002) for an application to computing c.i.s for certain quantities of interest associated with random effects models.

In what follows, we wish to treat the analytic method as a "black box," waiting until Section 8.4.2 to discuss why it works. The Matlab function implementing the method is called `binomCI.m`, and its contents are given in Listing 8.3. It inputs the sample size n, the number of successes s, and the value of α corresponding to the desired confidence level; it outputs the lower and upper limits of the c.i. The computation entails root searching over a function that involves the incomplete beta function (A.13), and so the method is not instantaneously calculated. In fact, it turns out to be considerably slower than using the inner bootstrap loop with $B_2 = 1000$. We will show a way of circumventing this issue below; but

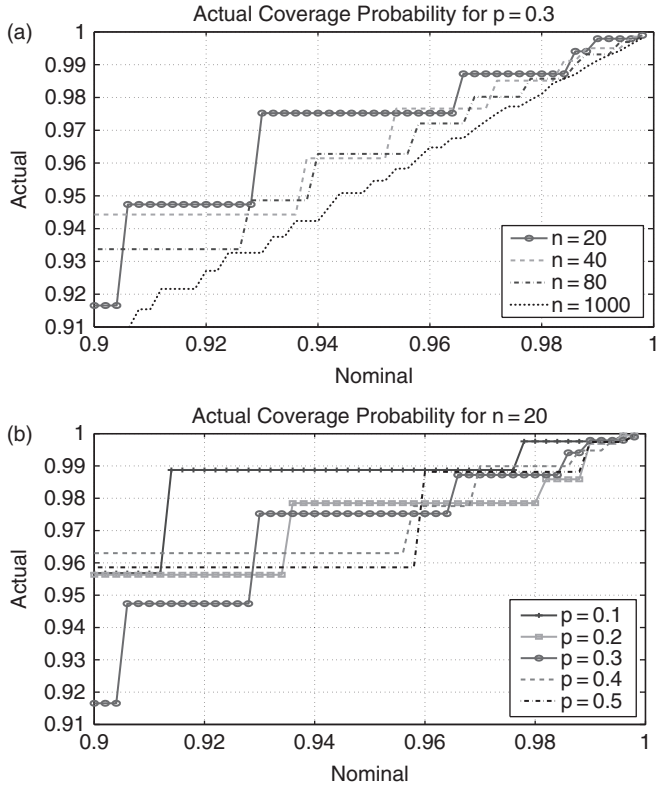

Figure 1.9 *Similar to Figure 1.5 (mapping between nominal and actual coverage probabilities for c.i.s of the success parameter in the i.i.d. Bernoulli model) except that, instead of using the single bootstrap for the c.i.s, this uses the analytic method. In figure b) actual coverage for a given p is identical to that for $1 - p$.*

first, we begin by computing the actual coverage of the analytic method itself, paralleling what we did in Figure 1.5 for the single bootstrap c.i.

We could use computer code as in Listing 1.4 and simulate `binomCI` as we did with the bootstrap, but this a waste of time and also less accurate than computing the *exact* nominal coverage values, which we can easily do in this case because, unlike the bootstrap, the analytic method is not stochastic. In other words, for a given n, s and α, the c.i. is always the same. Thus, for a given n and α, we can simply compute the actual coverage by using the law of total probability (A.30): with C the event that the c.i. covers the true value of p, $\Pr(C) = \sum_{s=0}^{n} \Pr(C \mid S = s) \Pr(S = s)$, where $S \sim \text{Bin}(n, p)$.

The code for this is shown in Listing 1.9, with output (now computed in a matter of seconds) shown in Figure 1.9. We see that, analogous to the bootstrap c.i., as the sample size increases, the nominal and actual coverage values converge, and are otherwise step functions. For a given sample size n and nominal coverage level $1 - \alpha$, the actual coverage for a specific p is identical to that for $1 - p$, as the reader can easily verify.

We now use the analytic method in place of the inner bootstrap loop of the double bootstrap. The procedure is the same as that outlined in Pseudo-code Listing 1.1, except for two changes. First, we delete step 2(c) and replace step 2(d) with the following:

```
function [nominal actual] = binomCIcheck (p,n)
nominal=0.9:0.002:0.998; nomlen=length(nominal); actual=zeros(nomlen,1);
for S=0:n, S
  for j=1:nomlen
    alpha=1-nominal(j); [lb,ub] = binomCI(n,S,alpha);
    if (p>lb) && (p<ub), actual(j)=actual(j)+binopdf(S,n,p); end
  end
end
```

Program Listing 1.9: Computes the mapping between the nominal and actual coverage probabilities for analytic c.i.s of the success parameter in the i.i.d. Bernoulli model. The function `binomCI` is given in Listing 8.3.

FOR j over each nominal coverage probability in a grid of values DO

(i) Compute the c.i. $\text{ci}^{(b_1,j)}$ from the analytic method using data $\mathbf{y}^{(b_1)}$.

(ii) Record a one in the b_1th row and jth column of matrix `bool` if $\text{ci}^{(b_1,j)}$ contains $\hat{\theta}^{\text{obs}}$.

Second, we replace step 5 with the following:

5. Deliver the c.i. for the actual data set $\mathbf{y}^{\text{obs}}$ computed using the analytic method (i.e., passing to `binomCI` the number of successes in the actual data set), and based on a confidence level of $1 - \alpha^*$.

Doing this, we discover that, instead of taking about 50 seconds to produce a c.i., as was the case with the regular double bootstrap (with $B_1 = B_2 = 1000$), we need about three times as long. This is because, as mentioned above, the analytic method is not of closed form. Thus, what was supposed to bestow upon us a massive time saving turns out to take longer!

This issue is easily resolved, however. Recall the crucial point that, for a given n, s and α, the c.i. is always the same. So, an idea that presents itself is to form a table in computer memory with $n - 1$ rows corresponding to the values that s can assume (recall that, with the modified c.i.s, we do not have to worry about the $s = 0$ and $s = n$ cases) and number of columns corresponding to the number of nominal α-values we entertain. Then, if a new s, α combination arises, we compute it with the `binomCI` function and add it to the table, and otherwise just read it from the table. This is a typical example of making a tradeoff between computer computation and memory, and we wish to emphasize this point:

> In many realistic statistical applications, a substantial amount of computation is required to obtain reliable inference. While computers are becoming ever faster, the amount of available memory is also increasing, and one can use a judicious tradeoff between the two in order to save time.

We will see another example of this idea in Section 9.3.3. In the case here, note that the required amount of memory, relative to what a modern desktop PC has, is trivial.

The implementation is done by modifying the code in Listing 1.8 as follows. First, the loop over `b2` is of course removed, and the loop after that, `for j=1:nomlen`, is replaced with the following code:

```
for j=1:nomlen
  alpha=1-nominal(j); [lb,ub]=localbinomCI(n,b1sum,alpha,j,nomlen);
  bool1(b1,j) = (phatdata>lb) & (phatdata<ub);
end
```

The function `localbinomCI` is given in Listing 1.10, and should be appended to the bottom of the `bernoulliCIdoubleboot` program. Our only problem is how to instruct Matlab to keep the tables `tablo` and `tabhi` in memory, given that they are part of a function and thus temporal. The answer is to use global variables – these being typical in many programming languages. They are not reset at every function call, but rather stay in memory, and are available to any function, no matter what its scope. While that would work, newer versions of Matlab offer an even better solution, via use of so called *persistent* variables: these are not global, as they can only be "seen" by the function in which they are defined, but still do not get reset or lost when the evoked function is finished. That is precisely what we want.

We can do a similar tabulation for the calls to the routines that compute the c.i. using the modification via (1.5) and (1.6) in the outer bootstrap loop. This will be especially relevant when n is small, in which case the probability is relatively high that a resample can have all zeros or all ones. The reader is encouraged to implement this; the required code is, of course, very similar to that given in Listing 1.10.

Using this method, a c.i. with inner loop computed analytically, and using $B_1 = 1000$ outer loops, takes only 4 seconds. With this massive speed increase, we have the luxury of using `sim` = 10,000 replications over our grid of parameter values of p, and yielding the plot in Figure 1.10(a). We see that the actual coverage is now very close to the nominal value for all cases except for some values of p when $n = 10$, though even in this case the performance is admirable compared to the analogous results shown in Figure 1.8.

To assess the influence of the value of B_1 in this case, the method was run again, but having used 10 times the number of outer bootstrap iterations (i.e., $B_1 = 10{,}000$), though only `sim` = 2500 replications. Over 2 weeks of computing later, we have the results in Figure 1.10(b). The differences are extremely small, indicating that $B_1 = 1000$ is (possibly more than) enough.

```
function [lb,ub]=localbinomCI(n,s,alpha,j,maxj)

persistent tablo tabhi
if isempty(tablo), tablo=zeros(n-1,maxj)-1; end
if isempty(tabhi), tabhi=zeros(n-1,maxj)-1; end

if tablo(s,j)<0
  [lb,ub]=binomCI(n,s,alpha); tablo(s,j)=lb; tabhi(s,j)=ub;
else
  lb=tablo(s,j); ub=tabhi(s,j);
end
```

Program Listing 1.10: Used to store the c.i.s in memory once they are computed.

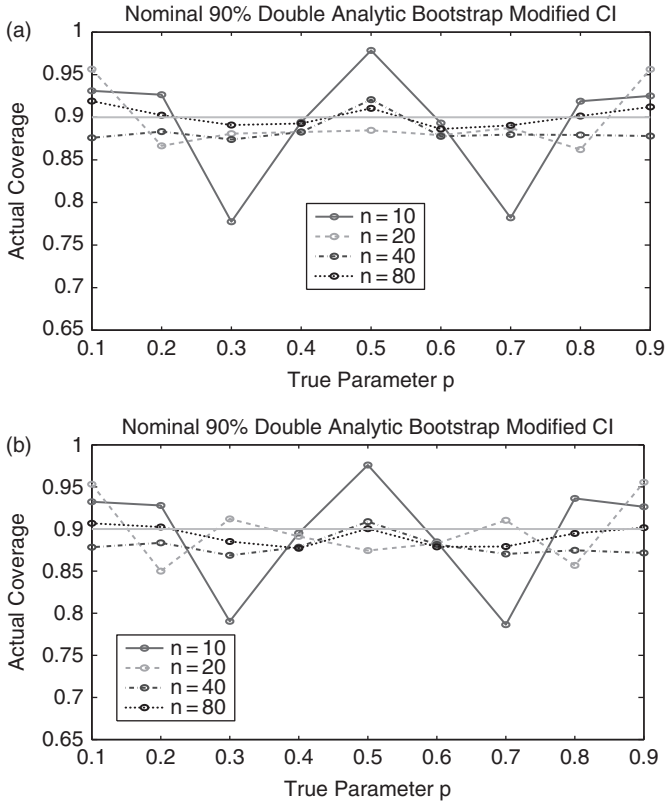

Figure 1.10 *Actual coverage of nominal 90% c.i.s using the double bootstrap with inner loop replaced by the analytic c.i., and having used the modification (1.5) and (1.6) in the outer bootstrap loop. (a) uses $B_1 = 1000$; (b) uses $B_1 = 10{,}000$.*

Remark. Observe that the analytic method for computing the c.i. of the Bernoulli parameter p is specifically designed for this sampling scheme, thus rendering it an *analytic parametric* c.i. As such, the parametric bootstrap can be viewed as an approximation to it; though for this particular model, the parametric and nonparametric bootstrap methods happen to be equivalent, as discussed above.

Problem 1.3 provides an example in which we have a method for computing an *analytic nonparametric* c.i. (for an order statistic of i.i.d. continuous data) and demonstrates that its performance is virtually identical to that of the nonparametric, but not the parametric, bootstrap c.i. ■

1.4 BOOTSTRAP CONFIDENCE INTERVALS IN THE GEOMETRIC MODEL

In this setting, we observe $G_1, \ldots, G_r$, where $G_i \overset{\text{i.i.d.}}{\sim} \text{Geo}(p)$, each with support $\{1, 2, \ldots\}$, and desire a c.i. for p.

Having already laid the groundwork for constructing c.i.s in the Bernoulli case, we just need to decide on a point estimator for p in the geometric model and, by changing a couple

```
r=10; p=0.3; B=1e4; alpha=0.05; sim=1e5; bool=zeros(sim,1);
for i=1:sim
  if mod(i,1e3)==0, i, end
  data=geornd(p,[r 1])+1; phat0=1/mean(data);
  phatvec = 1./mean(geornd(phat0,[r B])+1);
  % could instead use: r ./ ( nbinrnd(r,phat0,[B,1])+r )
  ci=quantile(phatvec,[alpha/2 1-alpha/2]); low=ci(1); high=ci(2);
  bool(i) = (p>low) & (p<high); % is the true p in the interval?
end
actualcoverage=mean(bool)
```

Program Listing 1.11: Determines the actual coverage of the nominal 95% parametric single bootstrap c.i. for the geometric model parameter p, using the m.l.e. as the point estimator of p. Compare with the code in Listing 1.3.

lines of code in Listing 1.3 above, we will have a program to compute a parametric bootstrap c.i. for p. We use the m.l.e. $\hat{p}_2 = 1/\bar{G} = r/\sum_{i=1}^{r} G_i$ from Section 1.1.2. This yields the code in Listing 1.11.

This can be augmented similarly to Listing 1.4 to use a grid of nominal coverage values and thus produce the mapping between the nominal and actual coverage levels. Figure 1.11 shows the results for four different sample sizes r and four choices of true parameter p. We see that, for each value of p, the nominal coverage approaches the actual coverage as r increases, similarly to the finding with the Bernoulli model as the sample size n increases. Observe also that, for a given r, the nominal and actual coverage levels are far closer for smaller values of p. This makes sense because of two facts: First, as indicated in Figure 1.3, m.s.e.$(\hat{p}) \to 0$ as $p \to 0$, so that the point estimates of p of the simulated data sets are very accurate. (This is also true for $p \to 1$.) Second, recall from the distribution of $\hat{p}$ shown in Figure 1.2 that, for a given r, as p increases, the number of points in the support of $\hat{p}$ decreases. This has the effect of causing the quantile function of $\hat{p}$ to become a step function with very few increments, rendering the c.i.s rather coarse.

As discussed in Section 1.3.1 above, we expect the parametric bootstrap to perform better than the nonparametric for the geometric model. The implementation of the nonparametric bootstrap in this context just requires modifying the code in Listing 1.11 to use resamples from `data`, as shown in Section 1.3.1. The results are shown in Figure 1.12, which parallels Figure 1.11. We indeed see that, for a given r and p, the actual coverage is considerably worse than that obtained with the parametric bootstrap.

We end this section with an example of constructing a c.i. not for the success probability p in Bernoulli trials, but rather the number of trials, n, conditional on knowing p.

Example 1.2 *According to the Wikipedia entry, Ibn Saud (1876–1953), the first monarch of Saudi Arabia, is estimated to have had 37 sons by 16 wives, though the actual number is not known, with other estimates stating the number of sons to be as high as 45, from 14 wives.*[6] *Let us say the number of sons is 40. If we assume that the gender of each child is an i.i.d. Bernoulli realization with probability $p = 0.5$ of getting a son, then an obvious and*

[6] The latter estimate was stated in the *Süddeutsche Zeitung*, March 23, 2002, page 3, in an interview with Prince Mamdouh, one of the younger of the 20 sons still alive at the time. Along with the exact number of sons being unknown, the number of daughters is even more of a mystery: With respect to this, the article states "von den Töchtern spricht hier keiner" ("no one here speaks of the daughters").

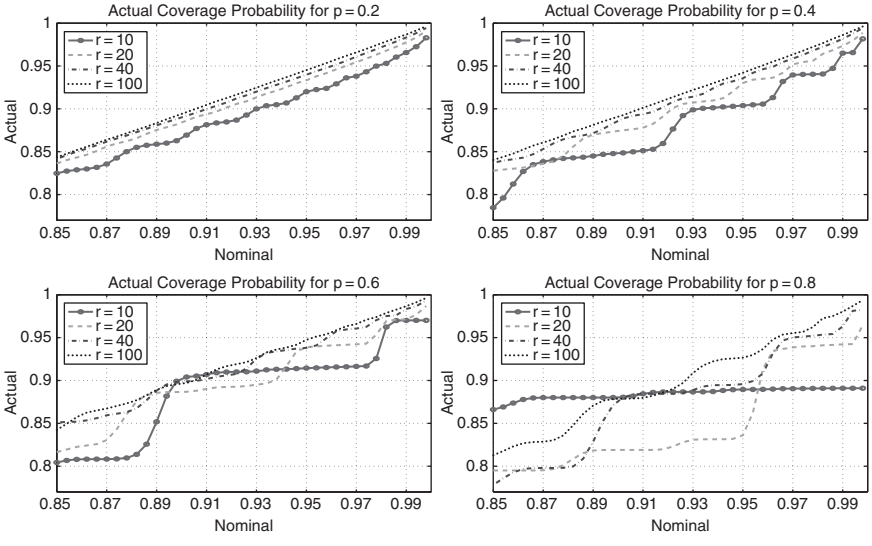

Figure 1.11 *Mapping between nominal and actual coverage probabilities for c.i.s of the success parameter p in the i.i.d. geometric model, using the parametric bootstrap. Based on 100,000 replications and $B = 10{,}000$.*

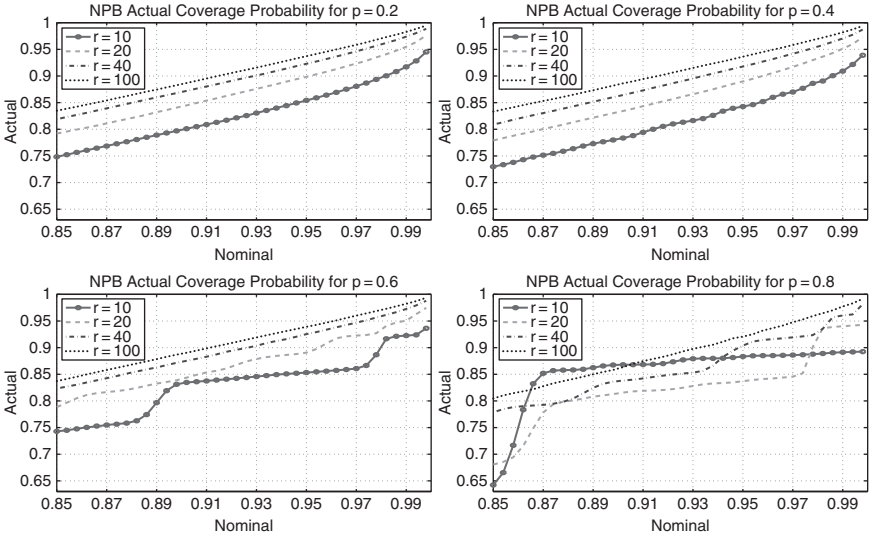

Figure 1.12 *Same as Figure 1.11, also with $B = 10{,}000$ and $\texttt{sim} = 100{,}000$, but with the nonparametric bootstrap (NPB).*

```
truep=0.5; boys=40; alpha=0.01; sim=1e5;
samp = sum( georn d(truep ,[boys sim])+1 ) + round(1/truep)/2;
nstar = round(quantile(samp,1-alpha))
```

Program Listing 1.12: Calculation of n^* via simulation.

```
truep=0.5; boys=40; alpha=0.01; n=round(boys/truep); prob=0;
while prob < (1-alpha)
  n=n+1; prob = 1-binocdf(boys-1,n,truep);
end, nstar = n
```

Program Listing 1.13: A second way of calculating n^*.

intuitive point estimate for the total number of children he fathered is 80. Less obvious is how to derive a value, say n^*, such that we can be, say, 99% confident that he had at most that number of children.

Here is one solution: We repeatedly simulate his reproductive history, recording for each replication the total number of engendered children, and report the 99% quantile of these values. The number of children begotten until a son arrives is a realization of a Geo(0.5) *r.v., so one of his simulated histories consists of summing 40 Geo*(0.5) *r.v.s, and then adding on half of the number of children he would have produced, on average, between his 40th and 41st sons. Recall that the expected value of a Geo*(p) *r.v. is* $1/p$*, so we add on* $1/(2p)$*. This is easily implemented in Matlab, using the code in Listing 1.12.*

For these values, we get $n^* = 104$*. Repeating this numerous times yielded the same value, indicating that 100,000 replications is (more than) enough. If we are satisfied with only 90% confidence, we get* $n^* = 93$*. We can check our obtained value of* $n^* = 104$ *by verifying that* $\Pr(S \geq 40) \geq 0.99$*, where* $S \sim \text{Bin}(104, 0.5)$*. In Matlab, this is computed as* `1-binocdf(boys-1, nstar,truep)`*. In fact, this immediately gives us another way of computing n^* that does not require simulation: we find the smallest value of n such that* $\Pr(S_n \geq 40) \geq 0.99$*, where* $S_n \sim Bin(n, 0.5)$*. Code for this is given in Listing 1.13.*

This results in $n^* = 103$ *and* $n^* = 92$ *for* $\alpha = 0.01$ *and* $\alpha = 0.10$*, respectively, showing that the first algorithm is slightly too conservative.*

Yet another method to compute n^* presents itself after a bit of contemplation: Given that we have the ability to compute a c.i. for p in the Bernoulli model for a fixed n, we can take n^* to be the largest *value of n such that a* $100(1-2\alpha)\%$ *two-sided c.i. for p, given n, still contains 0.5. (We use* 2α *because we are only interested in a one-sided interval for n, but our program for c.i.s for p is two-sided – it inputs* α *and "shares" it in both tails by dividing by 2.) In particular, if we take* $n = 80$ *and have 40 successes, then the c.i. for p will obviously contain 0.5, but by increasing n (and always using 40 successes), eventually it would get so large that* $p = 0.5$ *becomes unlikely and will not be in the c.i. Starting from* $n = 80$*, we continue to increase n and take the last value such that its c.i. for p still contains 0.5. To increase speed, we use the analytic method for constructing a c.i. for p, implemented in function* `binomCI` *and discussed in Section 1.3.4. The code for computing n^* is given in Listing 1.14.*

Running this results in $n^* = 104$ *and* $n^* = 93$ *for* $\alpha = 0.01$ *and* $\alpha = 0.10$*, respectively, which are exactly the same values given by our first method.*

```
truep=0.5; boys=40; alpha=0.01; n=round(boys/truep); upperbound=1;
while upperbound > truep
  n=n+1; [lowerbound, upperbound] = binomCI(n,boys,2*alpha);
end
nstar = n-1 % we find the first value of n such that it is NOT true
            % that 0.5 is in the c.i., so we need to subtract 1
```

Program Listing 1.14: Calculation of n^* using the analytic method for confidence intervals.

Likelihood-based inference on n for both known and unknown p is discussed in Aitkin and Stasinopoulos (1989). See Problem 1.1 for another "application." ■

1.5 PROBLEMS

> Opportunity is missed by most people because it is dressed in overalls and looks like work.
>
> (Thomas Edison)

1.1 The yearly astrology meeting you are organizing takes place soon, but, as the stellar bodies would have it, you can only find the participation list for Sagittarius, which contains 39 members. You wish to be 99% sure of having enough seats (and other relevant paranormal paraphernalia) for you and all listed members. Being skilled in astrology, you find that, based on celestial divination and the Book of Revelation, you will require 666 seats. Confirm this without divination.

1.2 Write a program to compute the modified c.i.s with the double bootstrap and reproduce Figure 1.8(b).

1.3 Recall that the quantile ξ_p of the continuous r.v. X is the value such that $F_X(\xi_p) = p$ for given probability p, $0 < p < 1$. Let $Y_1 < Y_2 < \cdots < Y_n$ be the order statistics of an i.i.d. random sample of length n from a continuous distribution. Example II.6.7 showed that[7]

$$\Pr(Y_i \le \xi_p \le Y_j) = \sum_{k=i}^{j-1} \binom{n}{k} p^k (1-p)^{n-k} = F_B(j-1, n, p) - F_B(i-1, n, p), \quad (1.7)$$

where $B \sim \text{Bin}(n, p)$ and F_B is the c.d.f. of B. This can be used to obtain an analytic, nonparametric (observe that the distribution of X plays no role) c.i. for the quantile. If we attempt to get, say, a 95% c.i., then we first need to compute the inverse c.d.f. values $i = F_B^{-1}(0.025, n, p) + 1$ and $j = F_B^{-1}(1 - 0.025, n, p) + 1$, and then (because of the discreteness of the distribution), compute the true *nominal* coverage, say $1 - \alpha^*$, from (1.7). Write code to compute i, j, and $1 - \alpha^*$.

Next, and more substantially, let $X_i \overset{\text{i.i.d.}}{\sim} \text{Exp}(\lambda)$, $i = 1, \ldots, n$, each with density function $f_{X_i}(x; \lambda) = \lambda \exp(-\lambda x) \mathbb{I}_{(0,\infty)}(x)$. Take $p = 1/2$, so that we wish to construct a c.i. for the median. Recall from Example I.4.6 (or quickly check that), the median of an $\text{Exp}(\lambda)$ r.v. is $\log(2)/\lambda$. Write a program that determines, via simulation with `sim` replications, the *actual* coverage and average length of the c.i.s based on the order

[7] The equation stated in Example II.6.7 has a typo; it is correct here.

statistics. Likewise, the program should also compute the (single) nonparametric and parametric bootstrap c.i.s, using confidence level $1 - \alpha^*$, and determine their actual coverage and average length. The nonparametric bootstrap c.i. should, as usual, use resampling from the actual data, and also the nonparametric estimator of the median (that being just the sample median); whereas the parametric bootstrap should, as usual, draw samples from the exponential distribution using the m.l.e. for $\hat{\lambda}$ (this being $1/\bar{X}$; see Example 3.3), and use the m.l.e. as the parametric estimate of the median, $\log(2)/\hat{\lambda}$.

The results, computed over a grid of n-values, are shown in Figure 1.13. In Figure 1.13(a), the big dark circles show the true nominal coverage $1 - \alpha^*$. As expected, they approach the desired actual coverage level of 95% as n grows. For $n \geq 20$, the order statistics and nonparametric bootstrap c.i.s have virtually the same actual coverage and lengths. This was to be expected, as the former is just an analytic method for computing a nonparametric c.i. For $n \geq 100$, the actual coverages of the three c.i.s are virtually the same, yet the length of the parametric bootstrap c.i. is blatantly shorter. This is because it incorporates the knowledge that the underlying distribution is exponential.

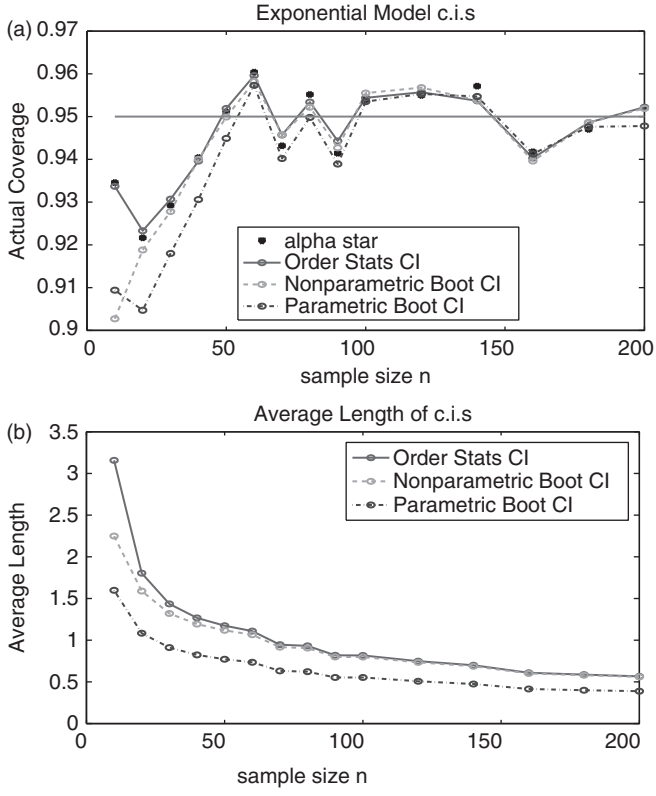

Figure 1.13 *(a) Actual coverage of the three types of c.i.s (lines), along with the true nominal coverage, $1 - \alpha^*$, from (1.7), as dark circles. (b) The average length of the c.i.s.*