Prof. David Draper
Department of
Applied Mathematics and Statistics
University of California, Santa Cruz

AMS 132: Discussion Section 6

The plan in this discussion section is to explore the following idea, recently mentioned in class and highly important in contemporary Bayesian computation:

> **Theorem:** Virtually anything you want to know about a p–dimensional probability distribution, even if p is really big, can — as long as p is finite — be learned, to arbitrary accuracy, just by taking large enough random samples from the distribution and summarizing the random draws appropriately.

Examples of random simulation as a method for problem-solving go back all the way to the 1700s, when a guy called Georges-Louis Leclerc, Comte de Buffon (French, 1707–1788) asked a question whose answer leads to simulating the value of π by dropping a needle haphazardly on a floor with parallel floor boards and keeping track of the frequency with which the needle touches one of the lines between the boards. I mentioned another example in class, in which in 1908 the Guinness brewery guy, William Gosset *("Student")* (British, 1876–1937), repeatedly simulated n random draws from the $N(\mu, \sigma^2)$ distribution to approximate the sampling distribution of $\frac{\bar{Y}-\mu}{s/\sqrt{n}}$.

The main people who first made general and pioneering use of the **Theorem** above were two scientists, Nicholas Metropolis (Greek–American physicist, 1915–1999) and Stanislaw Ulam (Polish–American mathematician, 1909–1984), who — in about 1942, while working on the *Manhattan Project* to create the first atomic bomb — proposed the name *Monte Carlo* methods (named after the famous European town where casinos have offered games of chance since the mid 1800s) for random-simulation approaches to mathematical problem-solving. The work of Metropolis and Ulam was classifed as a war secret by the U.S. government, and Metropolis and Ulam were not allowed to publish until 1949; the idea of Monte-Carlo sampling didn't really take off until the 1960s, when the first digital computers made taking large numbers of random draws from a distribution possible, and Monte-Carlo methods have become more and more popular since then. Mathematicians today typically try to solve the problem they're working on directly in closed form, but if they can't find closed-form solutions they often use the Monte Carlo approach to gain insight into what analytic solutions might look like if you could find them.

(1) As our first example of Monte-Carlo approximation, consider the sampling model — $(Y_i \,|\, \theta \, \mathcal{B}) \stackrel{\text{IID}}{\sim} \text{Bernoulli}(\theta)$, for $i = 1, \ldots, n$ — we used in the Kaiser ICU case study, in which we saw that a conjugate $\text{Beta}(\alpha, \beta)$ prior distribution led to a $\text{Beta}(\alpha + s, \beta + n - s)$ posterior for θ, where $s = \sum_{i=1}^{n} y_i$ is a sufficient statistic and $\boldsymbol{y} = (y_1, \ldots, y_n)$ is the vector of observed binary outcomes. In the Kaiser case study, $(s, n) = (4, 112)$ and we found that $\alpha = \beta = 0.5$ led to a well-calibrated low-information prior consistent with the context of the problem: this focuses attention on the $\text{Beta}(4.5, 108.5)$ posterior distribution. We were able to summarize this distribution with the `R` functions `dbeta` (to plot the posterior density) and `qbeta` (to get the 2.5% and 97.5% quantiles of the distribution, to construct a 95% interval for θ, and we

used the closed-form results

$$\text{If } (\theta \,|\, \mathcal{B}) \sim \text{Beta}(\alpha, \beta) \text{ then } E(\theta \,|\, \mathcal{B}) = \frac{\alpha}{\alpha + \beta} \text{ and } SD(\theta \,|\, \mathcal{B}) = \sqrt{\frac{\alpha\,\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}}$$

to compute the posterior mean and SD for θ. But what if you didn't know those closed-form results and you didn't have `dbeta` and `qbeta`? According to the Monte-Carlo **Theorem**, you're supposed to be able to obtain all of the same findings just by taking random samples from the posterior, for example with the `R` function `rbeta` — how does that work? Let's find out.

(a) Download the `R` code for Discussion Section 6 from the course web page. Run the code block between the delimiters

```
# --> code for part 1 (a) of discussion section 6 starts here <--
```

and

```
# --> code for part 1 (a) of discussion section 6 ends here <--
```

Verify that you get precisely the same results I got with the random number generator `seed` set to 1. As the code block suggests, now try running this same code with two new random number seeds of your own choosing, and make a table like the one at the end of the code block.

(b) As noted in the code, you can see that with only $M = 100$ simulated draws, the Monte Carlo method is not very accurate, but we can solve that problem easily, just by increasing M. Rerun the code block with $M = 1{,}000{,}000$ and make a second table with results from the same three random number seeds. How much more accurate are the Monte Carlo results now?

(c) We need a way to quantify the amount of uncertainty in Monte Carlo estimates as a function of M. Fortunately, we're statisticians: consider, for example, the Monte Carlo estimate of the posterior mean. This is just the mean $\bar{\theta}^*$ of the M IID draws $(\theta_1^*, \ldots, \theta_M^*)$ that you got `R` to make for you, and we know from the frequentist part of this course that the estimated standard error of a sample mean under IID sampling is just the sample SD divided by $\sqrt{M}$, so let's invent a quantity called the *Monte Carlo standard error (MCSE)* of the sample mean:

$$\widehat{MCSE}\big(\bar{\theta}^*\big) = \frac{s_{\theta^*}}{\sqrt{M}}\,, \tag{1}$$

in which s_{θ^*} is the sample SD of the M IID draws $(\theta_1^*, \ldots, \theta_M^*)$. Run the code block between the delimiters

```
# --> code for part 1 (c) of discussion section 6 starts here <--
```

and

```
# --> code for part 1 (c) of discussion section 6 ends here <--
```

Verify that you get precisely the same results I got with the random number generator `seed` set to 1, and try rerunning the code block with a random number seed of your choice. You can see that the MCSE of $\bar{\theta}^*$ is dramatically smaller with $M = 1$ million than with $M = 100$, and we can readily compute how much smaller: $\sqrt{\frac{1000000}{100}} = 100$

times smaller. Notice that, right in the middle of a Bayesian calculation, we can use frequentist ideas on the *Monte Carlo data set* $(\theta_1^*, \ldots, \theta_M^*)$ to construct a 95% Monte Carlo confidence interval for the posterior mean, namely

$$\bar{\theta}^* \pm 2\,\widehat{MCSE}\big(\bar{\theta}^*\big) = \bar{\theta}^* \pm 2\,\frac{s_{\theta^*}}{\sqrt{M}}\,. \tag{2}$$

(d) Now suppose you want to plan a Monte Carlo experiment: how big should M be to get the accuracy level you need (not less, and not more, than that level)? One simple idea would be to set the MCSE equal to a target value T and solve for M:

$$\text{If} \quad \widehat{MCSE}\big(\bar{\theta}^*\big) = \frac{s_{\theta^*}}{\sqrt{M}} = T \quad \text{then } M = \frac{s_{\theta^*}^2}{T^2}\,. \tag{3}$$

Easy enough, but there's a slight problem: before running the experiment you don't know how s_{θ^*} will come out. Here's a simple fix:

> In step (1) of your design, run a small pilot experiment with (say) $M = M_0 =$ 1,000 and observe the value of s_{θ^*}; plug this into equation (3) and solve for M. If the resulting M is less than M_0, you're done; otherwise in step (2) of your design, make $(M - M_0)$ additional Monte Carlo draws, append them to your provisional Monte Carlo data set of size M_0 in step (1), and summarize your findings.

Suppose that in the Kaiser case study we had wanted a Monte Carlo standard error of $T = 0.00001$ for the Monte Carlo estimate of the posterior mean (this is unrealistically small, but we're just practicing here). Run a pilot study with $M_0 = 1{,}000$, as described in the paragraph above, and show that the full Monte Carlo run would have needed about $M \doteq 3$ million draws to achieve this level of Monte-Carlo accuracy.

(e) The Monte Carlo method is far more flexible than just approximating posterior means, SDs and densities. For example, in the Kaiser case study above, suppose that in addition to θ we're also interested in $\eta = \log\big(\frac{\theta}{1-\theta}\big)$ and in estimating the posterior probability that θ (the population rate of unplanned transfers to the ICU for heart attack patients) was no more than (say) 7%. Keeping track of a quantity in your Monte Carlo data set is called *monitoring* that quantity: in addition to θ we want to monitor η and $P(\theta \le 0.07 \,|\, \boldsymbol{y}\,\mathcal{B})$. Nothing could be simpler for η: just create a new variable in `R` called `eta.star` whose values are given by

```
eta.star <- log( theta.star / ( 1 - theta.star ) )
```

and summarize `eta.star`. Do this with your choice of random number seed and $M =$ 1 million: use Monte Carlo to approximate the posterior mean and SD of η (you should get values around -3.3 and 0.51, respectively) and make a histogram summarizing the posterior density of η (you should find that it looks roughly Gaussian but with a bit of a left tail).

As for $P(\theta \le 0.07 \,|\, \boldsymbol{y}\,\mathcal{B})$, slightly more cleverness is needed: let's use the fact, from AMS 131, that

> If Y is a random variable and A is a true/false proposition about Y (e.g., $Y > 5$), then you can create an indicator variable $I(A) = 1$ if A is true and 0 if A is false and calculate $P(A)$ by computing the expected value of $I(A)$.

Thus, to use Monte Carlo to estimate $P(\theta \le 0.07 \,|\, \boldsymbol{y}\,\mathcal{B})$, just create in `R` a variable that's 1 whenever $\theta^* \le 0.07$ and 0 otherwise and calculate the mean of that variable (you should get a value of around 0.93 for this posterior probability).