

One-Day Short Course on Bayesian Modeling, Inference and Prediction

5: Simulation-Based Computation

David Draper

Department of Applied Mathematics and Statistics
University of California, Santa Cruz

`draper@ams.ucsc.edu`

`http://www.ams.ucsc.edu/~draper`

*Sponsored by the Boston Chapter
of the American Statistical Association*

10 December 2004, 8am–5.30pm
Hotel@MIT, 20 Sidney Street, Cambridge MA

© 2004 David Draper (all rights reserved)

5.1 Introduction to Markov Chain Monte Carlo (MCMC) methods

Computation via conjugate analysis (parts 2–4) produces **closed-form results** (good) but is **limited in scope** to a fairly small set of models for which straightforward conjugate results are possible (bad).

This was a **severe limitation** for Bayesians for almost 250 years (from the 1750s to the 1980s).

Over the past 10 years the Bayesian community has “discovered” and developed an entirely new computing method, **Markov chain Monte Carlo (MCMC)** (“discovered” because the physicists first figured it out about 50 years ago: Metropolis and Ulam, 1949; Metropolis et al., 1953).

We’ve seen that the **central Bayesian practical challenge** is the **computation of high-dimensional integrals**.

People working on the first atom bomb in World War II faced a **similar challenge**, and noticed that **digital computers** (which were then passing from theory (Turing 1943) to reality) offered an **entirely new approach** to solving the problem.

The idea (Metropolis and Ulam, 1949) was based on the observation that **anything you want to know about a probability distribution** can be learned to arbitrary accuracy by **sampling from it**.

Suppose, for example, that you’re interested in a posterior distribution $p(\theta|y)$ which **cannot be worked with (easily) in closed form**, and initially (to keep things simple) think of θ as a **scalar** (real number) rather than vector.

Simulation-Based Computation

Four things of direct interest to you about $p(\theta|y)$ would be

- its **mean** $\mu = E(\theta|y)$ and **standard deviation** $\sigma = \sqrt{V(\theta|y)}$,
- its **shape** (basically you'd like to be able to trace out (an estimate of) the entire **density curve**), and
- one or more of its **quantiles** (e.g., to construct a 95% central posterior interval for θ you need to know the **2.5% and 97.5% quantiles**, and sometimes the **posterior median** (the **50th percentile**) is of interest too).

Suppose you could take an **arbitrarily large random sample** from $p(\theta|y)$, say $\theta_1^*, \dots, \theta_m^*$.

Then each of the above four aspects of $p(\theta|y)$ can be **estimated** from the θ^* sample:

- $\hat{E}(\theta|y) = \bar{\theta}^* = \frac{1}{m} \sum_{j=1}^m \theta_j^*$,
- $\sqrt{\hat{V}(\theta|y)} = \sqrt{\frac{1}{m-1} \sum_{j=1}^m (\theta_j^* - \bar{\theta}^*)^2}$,
- the density curve can be estimated by a **histogram** or **kernel density estimate**, and
- percentiles can be estimated by **counting** how many of the θ^* values fall below a series of specified points—e.g., to find an estimate of the 2.5% quantile you solve the equation

$$\hat{F}_\theta(t) = \frac{1}{m} \sum_{j=1}^m I(\theta_j^* \leq t) = 0.025 \quad (1)$$

for t , where $I(A)$ is the **indicator function** (1 if A is true, otherwise 0).

5.2 IID Sampling; Rejection Sampling

These are called **Monte Carlo** estimates of the true summaries of $p(\theta|y)$ because they're based on the **controlled use of chance**.

Theory shows that with large enough m , each of the Monte Carlo (or **simulation-based**) estimates can be made arbitrarily close to the truth with arbitrarily high probability, under some reasonable assumptions about the **nature of the random sampling**.

One way to achieve this, of course, is to make the sampling **IID** (this is **sufficient** but **not necessary**—see below).

If, for example, $\bar{\theta}^* = \frac{1}{m} \sum_{j=1}^m \theta_j^*$ is based on an IID sample of size m from $p(\theta|y)$, we can use the **frequentist fact** that in repeated sampling $V(\bar{\theta}^*) = \frac{\sigma^2}{m}$, where (as above) σ^2 is the variance of $p(\theta|y)$, to construct a **Monte Carlo standard error (MCSE)** for $\bar{\theta}^*$:

$$\widehat{SE}(\bar{\theta}^*) = \frac{\hat{\sigma}}{\sqrt{m}}, \quad (2)$$

where $\hat{\sigma}$ is the **sample SD** of the θ^* values.

This can be used, possibly after some **preliminary experimentation**, to decide on m , the Monte Carlo **sample size**, which later we'll call the length of the **monitoring run**.

An IID example. Consider the posterior distribution $p(\lambda|y) = \Gamma(29.001, 14.001)$ in the **LOS example** in part 3.

We already know that the **posterior mean** of λ in this example is $\frac{29.001}{14.001} \doteq 2.071$; let's see how well the Monte Carlo method does in estimating this **known truth**.

IID Example (continued)

Here's an R function to construct **Monte Carlo estimates** of the **posterior mean** and **MCSE values** for these estimates.

```
gamma.sim <- function( m, alpha, beta, n.sim, seed ) {  
  set.seed( seed )  
  theta.out <- matrix( 0, n.sim, 2 )  
  for ( i in 1:n.sim ) {  
    theta.sample <- rgamma( m, alpha, 1 / beta )  
    theta.out[ i, 1 ] <- mean( theta.sample )  
    theta.out[ i, 2 ] <- sqrt( var( theta.sample ) / m )  
  }  
  return( theta.out )  
}
```

This function simulates, `n.sim` times, the process of **taking** an IID sample of size m from the $\Gamma(\alpha, \beta)$ distribution and **calculating** $\bar{\theta}^*$ and $\widehat{SE}(\bar{\theta}^*)$.

```
rosalind 296> R
```

```
R : Copyright 2001, The R Development Core Team  
Version 1.2.1 (2001-01-15)
```

```
> m <- 1000
```

```
> alpha <- 29.001
```

```
> beta <- 14.001
```

```
> n.sim <- 500
```

```
> seed <- c( 6425451, 9626954 )
```

IID Example (continued)

```
> theta.out <- gamma.sim( m, alpha, beta, n.sim, seed )
```

```
# This took about 1 second at 550 Unix MHz.
```

```
> theta.out[ 1:10, ]
```

```
          [,1]      [,2]
[1,] 2.082105 0.01166379
[2,] 2.072183 0.01200723
[3,] 2.066756 0.01247277
[4,] 2.060785 0.01200449
[5,] 2.078591 0.01212440
[6,] 2.050640 0.01228875
[7,] 2.071706 0.01182579
[8,] 2.063158 0.01176577
[9,] 2.058440 0.01186379
[10,] 2.068976 0.01220723
```

The $\bar{\theta}^*$ values fluctuate around the truth with a **give-or-take** of about 0.012, which agrees well with the **theoretical SE**

$$\frac{\sigma}{\sqrt{m}} = \frac{\sqrt{\alpha}}{\beta\sqrt{m}} \doteq 0.01216 \text{ (recall that the variance of a Gamma distribution is } \frac{\alpha}{\beta^2}\text{).}$$

```
> postscript( "gamma-sim1.ps" )
```

```
> theta.bar <- theta.out[ , 1 ]
```

```
> qqnorm( ( theta.bar - mean( theta.bar ) ) /
          sqrt( var( theta.bar ) ) )
```

```
> abline( 0, 1 )
```

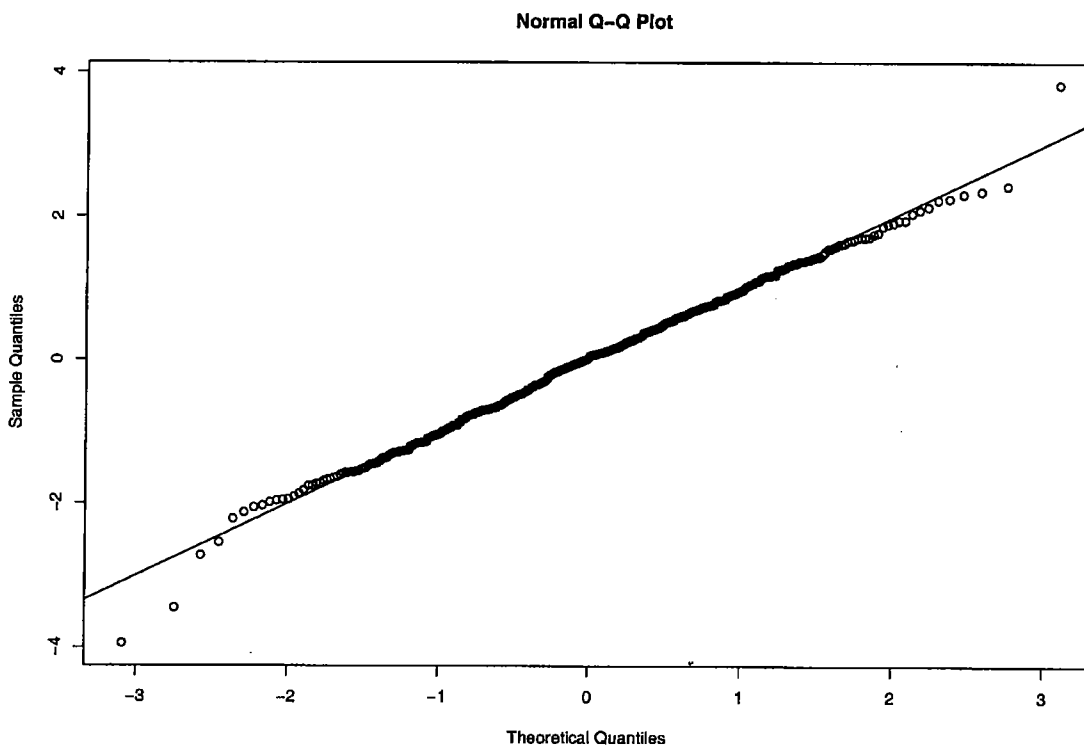
```
> dev.off( )
```

```
null device
```

```
1
```

Each of the $\bar{\theta}^*$ values is the mean of $m = 1,000$ IID draws, so (by the CLT) the **distribution** of the random variable $\bar{\theta}^*$ should be **closely approximated by a Gaussian**.

IID Example (continued)



```
> truth <- alpha / beta

> theta.bar.SE <- theta.out[ , 2 ]

> qnorm( 0.025 )
[1] -1.959964

> sum( ( theta.bar - 1.96 * theta.bar.SE < truth ) *
      ( truth < theta.bar + 1.96 * theta.bar.SE ) ) / n.sim

[1] 0.972
```

Thus we can use **frequentist ideas** to work out how big m needs to be to have **any desired Monte Carlo accuracy** for $\bar{\theta}^*$ as an estimate of the posterior mean $E(\theta|y)$.

In practice, with $p(\theta|y)$ unknown, you would probably take an **initial sample** (of size $m = 1,000$, say) and look at the MCSE to decide **how big m really needs to be**.

IID Example (continued)

```
> theta.bar <- gamma.sim( m, alpha, beta, 1, seed )
```

```
> theta.bar
      [,1]      [,2]
[1,] 2.082105 0.01166379
```

(1) Suppose you wanted the MCSE of $\bar{\theta}^*$ to be (say) $\epsilon = 0.001$. Then you could **solve the equation**

$$\frac{\hat{\sigma}}{\sqrt{m}} = \epsilon \quad \leftrightarrow \quad m = \frac{\sigma^2}{\epsilon^2}, \quad (3)$$

which says (unhappily) that the required m goes up as the **square** of the posterior SD and as the **inverse square** of ϵ .

The previous calculation shows that $\frac{\hat{\sigma}}{\sqrt{1000}} \doteq 0.01166379$, from which $\hat{\sigma} \doteq 0.3688414$, meaning that **to get** $\epsilon = 0.001$ **you need a sample of size** $\frac{0.3688414^2}{0.001^2} \doteq 136,044 \doteq 136\text{k}$ (!).

(2) Suppose instead that you wanted $\bar{\theta}^*$ to **differ** from the true posterior mean μ by **no more than** ϵ_1 with Monte Carlo probability **at least** $(1 - \epsilon_2)$:

$$P(|\bar{\theta}^* - \mu| \leq \epsilon_1) \geq 1 - \epsilon_2, \quad (4)$$

where $P(\cdot)$ here is based on the (frequentist) **Monte Carlo randomness** inherent in $\bar{\theta}^*$.

We know from the CLT and the calculations above that **in repeated sampling** $\bar{\theta}^*$ is approximately **normal** with mean μ and variance $\frac{\sigma^2}{m}$; this leads to the inequality

$$m \geq \frac{\sigma^2 \left[\Phi^{-1}\left(1 - \frac{\epsilon_2}{2}\right) \right]^2}{\epsilon_1^2}, \quad (5)$$

where $\Phi^{-1}(q)$ is the place on the standard normal curve where $100q\%$ of the area is to the left of that place (the q th **quantile** of the standard normal distribution).

A Closer Look at IID Sampling

(5) is like (3) except that the value of m from (3) has to be multiplied by $[\Phi^{-1}(1 - \frac{\epsilon_2}{2})]^2$, which typically makes the required sample sizes **even bigger**.

For example, with $\epsilon_1 = 0.001$ and $\epsilon_2 = 0.05$ —i.e., to have at least 95% Monte Carlo confidence that reporting the posterior mean as 2.071 will be correct to about **four significant figures**—(5) says that you would need a monitoring run of at least $136,044(1.959964)^2 \doteq 522,608 \doteq 523\text{k}$ (!).

(On the other hand, this sounds like a long monitoring run but only takes about **2.5 seconds** at 550 Unix MHz on a SunBlade 100, yielding $[\bar{\theta}^*, \widehat{SE}(\bar{\theta}^*)] = (2.0709, 0.00053)$.)

It's evident from calculations like these that people often report simulation-based answers with numbers of significant figures **far in excess of what is justified** by the actual accuracy of the Monte Carlo estimates.

A Closer Look at IID Sampling. I was able to easily perform the above **simulation study** because R has a large variety of built-in functions like `rgamma` for **pseudo-random-number generation**.

How would you go about **writing** such functions **yourself**?

There are a number of **general-purpose** methods for generating random numbers (I won't attempt a survey here); the one we need to look closely at, to understand the algorithms that arise later in this section, is **rejection sampling** (von Neumann 1951), which is often one of the most **computationally efficient** ways to make IID draws from a distribution.

Rejection Sampling

Example. In the spring of 1993 a survey was taken of **bicycle** and other **traffic** in the vicinity of the University of California, Berkeley, campus (Gelman et al. 1995).

As part of this survey 10 city blocks on **residential** streets with **bike routes** were chosen at random from all such blocks at Berkeley; on one of those blocks n vehicles were observed on a randomly chosen **Tuesday afternoon from 3 to 4pm**, and s of them were bicycles.

To draw inferences about the underlying **proportion θ of bicycle traffic** (PBT) on blocks similar to this one at times similar to Tuesday afternoons from 3 to 4pm, it's natural (as in the AMI mortality case study) to employ the **model**

$$\left\{ \begin{array}{l} \theta \sim \text{Beta}(\alpha_0, \beta_0) \\ (S|\theta) \sim \text{Binomial}(n, \theta) \end{array} \right\} \rightarrow (\theta|s) \sim \text{Beta}(\alpha_0 + s, \beta_0 + n - s), \quad (6)$$

provided that whatever **prior information** I have about θ can be meaningfully captured in the **Beta** family.

After **reflection** I realize that I'd be quite surprised if the PBT in residential city blocks with bike routes in Berkeley on Tuesday afternoons from 3 to 4pm was **less than 5%** or **greater than 50%**.

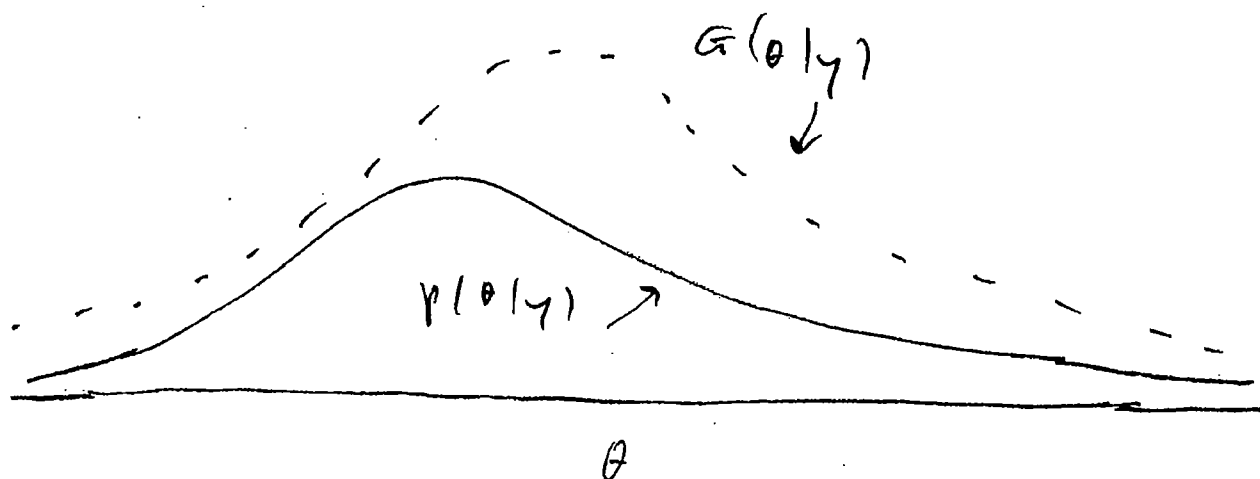
Making this operational by assuming that in the prior $p(0.05 \leq \theta \leq 0.5) = 0.9$, and putting **half** of the remaining prior probability in each of the **left** and **right tails** of the Beta distributions, yields (via numerical methods similar to those in the AMI case study) $(\alpha_0, \beta_0) = (2.0, 6.4)$ (this Beta distribution has prior mean and SD **0.24** and **0.14**, respectively).

In the city block in question the **data** came out $(n, s) = (74, 16)$, so that the data mean was **0.216**, and the posterior is then $\text{Beta}(\alpha_0 + s, \beta_0 + n - s) = \text{Beta}(18.0, 64.4)$.

Rejection Sampling (continued)

Pretend for the sake of illustration of **rejection sampling** that you didn't know the formulas for the mean and SD of a Beta distribution, and suppose that you wanted to use **IID Monte Carlo sampling** from the $\text{Beta}(\alpha_0 + s, \beta_0 + n - s)$ posterior to estimate the **posterior mean**.

Here's von Neumann's basic idea: suppose the target density $p(\theta|y)$ is **difficult** to sample from, but you can find an integrable **envelope function** $G(\theta|y)$ such that (a) G **dominates** p in the sense that $G(\theta|y) \geq p(\theta|y) \geq 0$ for all θ and (b) the density g obtained by normalizing G —later to be called the **proposal distribution**—is easy and fast to sample from.



Then to get a **random draw** from p , make a draw θ^* from g instead and **accept** or **reject** it according to an **acceptance probability** $\alpha_R(\theta^*|y)$; if you **reject** the draw, **repeat** this process until you accept.

von Neumann showed that the **choice**

$$\alpha_R(\theta^*|y) = \frac{p(\theta^*|y)}{G(\theta^*|y)} \quad (7)$$

correctly produces IID draws from p , and you can **intuitively** see that he's right by the following argument.

Rejection Sampling (continued)

Making a **draw** from the posterior distribution of interest is like choosing a point **at random** (in two dimensions) under the density curve $p(\theta|y)$ in such a way that **all possible points are equally likely**, and then writing down its θ value.

If you instead draw from G so that all points under G are equally likely, to get **correct** draws from p you'll need to throw away any point that falls between p and G , and this can be accomplished by **accepting** each sampled point θ^* with probability $\frac{p(\theta^*|y)}{G(\theta^*|y)}$, as von Neumann said.

A **summary** of this method is as follows.

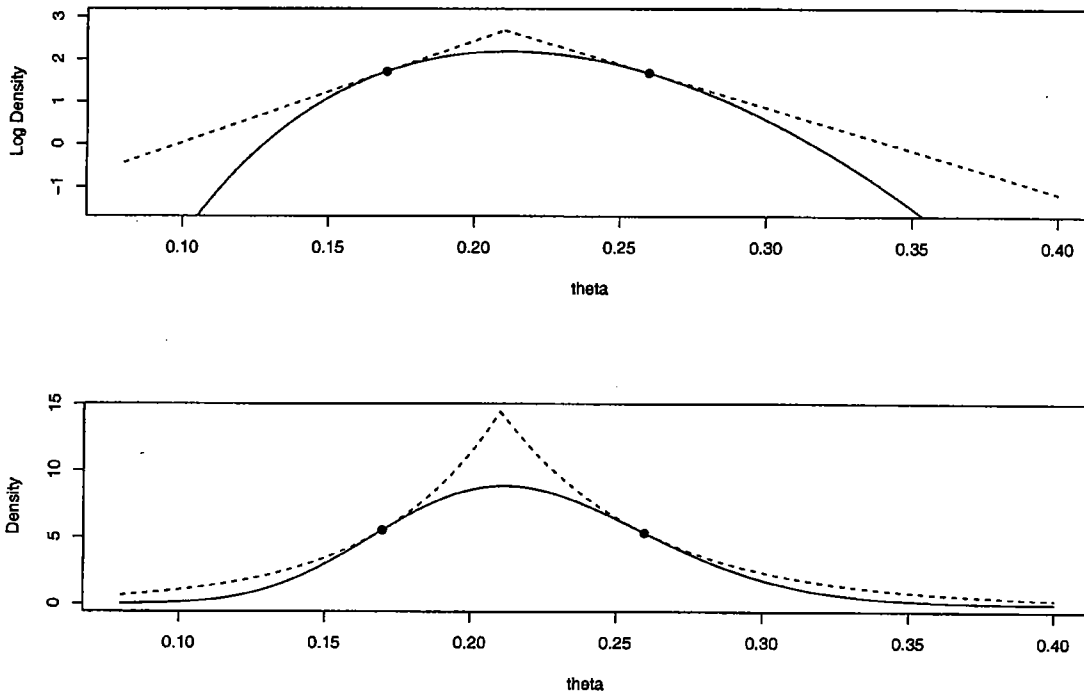
Algorithm (rejection sampling). To make m draws at random from the density $p(\theta|y)$ for real-valued θ , select an integrable **envelope function** G —which when normalized to integrate to 1 is the **proposal distribution** g —such that $G(\theta|y) \geq p(\theta|y) \geq 0$ for all θ ; define the acceptance probability $\alpha_R(\theta^*|y) = \frac{p(\theta^*|y)}{G(\theta^*|y)}$; and

```
Initialize  $t \leftarrow 0$ 
Repeat {
  Sample  $\theta^* \sim g(\theta|y)$ 
  Sample  $u \sim \text{Uniform}(0, 1)$ 
  If  $u \leq \alpha_R(\theta^*|y)$  then
    {  $\theta_{t+1} \leftarrow \theta^*$ ;  $t \leftarrow (t + 1)$  }
}
until  $t = m$ .
```

(8)

The **figure** below demonstrates this method on the Beta(18.0, 64.4) density arising in the **Beta-Bernoulli example** above.

Rejection Sampling (continued)



Rejection sampling permits considerable **flexibility** in the choice of **envelope function**; here, borrowing an idea from Gilks and Wild (1992), I've noted that the relevant Beta density is **log concave** (a real-valued function is log concave if its **second derivative** on the log scale is **everywhere non-positive**), meaning that it's easy to construct an envelope on that scale in a **piecewise linear** fashion, by choosing points on the log density and constructing **tangents** to the curve at those points.

The **simplest** possible such envelope involves **two line segments**, one on either side of the **mode**.

The **optimal** choice of the tangent points would maximize the marginal **probability of acceptance** of a draw in the rejection algorithm, which can be shown to be

$$\left[\int G(\theta) d\theta \right]^{-1}; \quad (9)$$

Rejection Sampling (continued)

in other words, you should **minimize** the area under the (un-normalized) envelope function subject to the constraint that it **dominates** the target density $p(\theta|y)$ (which makes eminently good sense).

Here this optimum turns out to be attained by locating the two tangent points at about **0.17** and **0.26**, as in the figure above; the resulting acceptance probability of about **0.75** could clearly be **improved** by adding more tangents.

Piecewise linear envelope functions on the log scale are a **good choice** because the resulting envelope density on the raw scale is a piecewise set of **scaled exponential distributions** (see the bottom panel in the figure above), from which random samples can be taken **quickly**.

A **preliminary** sample of $m_0 = 500$ IID draws from the Beta(18.0, 64.4) distribution using the above rejection sampling method yields $\bar{\theta}^* = \mathbf{0.2197}$ and $\hat{\sigma} = \mathbf{0.04505}$, meaning that the posterior mean has already been estimated with an **MCSE** of only $\frac{\hat{\sigma}}{\sqrt{m_0}} = 0.002$ even with just **500** draws.

Suppose, however, that—as in equation (4) above—I want $\bar{\theta}^*$ to **differ** from the true posterior mean μ by no more than some (perhaps even smaller) **tolerance** ϵ_1 with Monte Carlo probability at least $(1 - \epsilon_2)$; then equation (5) tells me how long to **monitor** the simulation output.

For instance, to pin down **three significant figures** (sigfigs) in the posterior mean in this example with high Monte Carlo accuracy I might take $\epsilon_1 = 0.0005$ and $\epsilon_2 = 0.05$, which yields a **recommended IID sample size** of

$$\frac{(0.04505^2)(1.96)^2}{0.0005^2} \doteq 31,200.$$

