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 \( \theta \) 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

\[
\begin{align*}
\theta & \sim \text{Beta}(\alpha_0, \beta_0) \\
(S|\theta) & \sim \text{Binomial}(n, \theta)
\end{align*}
\]

\( \rightarrow (\theta|s) \sim \text{Beta}(\alpha_0 + s, \beta_0 + n - s), 
\]

provided that whatever prior information I have about \( \theta \) 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 Beta($\alpha_0 + s$, $\beta_0 + n - s$) = posterior to estimate the posterior mean.

Here is 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 $\theta$ and (b) the density $g$ obtained by normalizing $G$—later to be called the proposal distribution—is easy and fast to sample from.

![Diagram showing $G(\theta|y)$ and $g(\theta|y)$ with annotations]

Then to get a random draw from $p$, make a draw $\theta^*$ 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

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

(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 \( \theta \) 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 \( \theta^* \) 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 \( \theta \), 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 \( \theta \); define the acceptance probability \( \alpha_R(\theta^*|y) = \frac{p(\theta^*|y)}{G(\theta^*|y)} \); and

\[
\begin{align*}
\text{Initialize } t &\leftarrow 0 \\
\text{Repeat } &
\begin{cases}
\text{Sample } \theta^* &\sim g(\theta|y) \\
\text{Sample } u &\sim \text{Uniform}(0,1) \\
\text{If } u \leq \alpha_R(\theta^*|y) &\text{ then} \\
\{ \quad \theta_{t+1} &\leftarrow \theta^*; \\ &t \leftarrow (t + 1) \} \\
\end{cases}
\end{align*}
\]

until \( t = m \).

The figure below demonstrates this method on the Beta(18.0, 64.4) density arising in the Beta-Bernoulli example above.
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 \( p(\theta|y) \).

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 \( \hat{\theta}^* = 0.2197 \) and \( \hat{\sigma} = 0.04505 \), 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 \( \hat{\theta}^* \) to **differ** from the true posterior mean \( \mu \) by no more than some (perhaps even smaller) **tolerance** \( \epsilon_1 \) 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

\[
\frac{(0.04505^2)(1.96)^2}{0.0005^2} = 31,200.
\]
Rejection Sampling (continued)

So I take another sample of 30,700 (which is virtually instantaneous at 550 Unix MHz) and merge it with the 500 draws I already have; this yields $\bar{\theta}^* = 0.21827$ and $\hat{\sigma} = 0.04528$, meaning that the MCSE of this estimate of $\mu$ is $\frac{0.04528}{\sqrt{31200}} = 0.00026$.

I might announce that I think $E(\theta|y)$ is about 0.2183, give or take about 0.0003, which accords well with the true value 0.2184.

Of course, other aspects of $p(\theta|y)$ are equally easy to monitor; for example, if I want a Monte Carlo estimate of $p(\theta \leq q | y)$ for some $q$, as noted above I just work out the proportion of the sampled $\theta^*$ values that are no larger than $q$.

Or, even better, I recall that $P(A) = E[I(A)]$ for any event or proposition $A$, so to the MC data set consisting of 31,200 rows and one column (the $\theta^*_i$) I add a column monitoring the values of the derived variable which is 1 whenever $\theta^*_i \leq q$ and 0 otherwise; the mean of this derived variable is the Monte Carlo estimate of $p(\theta \leq q | y)$, and I can attach an MCSE to it in the same way I did with $\bar{\theta}^*$.

By this approach, for instance, the Monte Carlo estimate of $p(\theta \leq 0.15 | y)$ based on the 31,200 draws examined above comes out $\hat{p} = 0.0556$ with an MCSE of 0.0013.

Percentiles are typically harder to pin down with equal Monte Carlo accuracy (in terms of sigfigs) than means or SDs, because the 0/1 scale on which they're based is less information-rich than the $\theta^*$ scale itself; if I wanted an MCSE for $\hat{p}$ of 0.0001 I would need an IID sample of more than 5 million draws (which would still only take a few seconds at contemporary workstation speeds).
Beyond Rejection Sampling

IID sampling is not necessary. Nothing in the Metropolis-Ulam idea of Monte Carlo estimates of posterior summaries requires that these estimates be based on IID samples from the posterior.

This is lucky, because in practice it’s often difficult, particularly when \( \theta \) is a vector of high dimension (say \( k \)), to figure out how to make such an IID sample, via rejection sampling or other methods (e.g., imagine trying to find an envelope function for \( p(\theta | y) \) when \( k \) is 10 or 100 or 1,000).

Thus it’s necessary to relax the assumption that \( \theta_j^* \overset{\text{IID}}{\sim} p(\theta | y) \), and to consider samples \( \theta_1^*, \ldots, \theta_m^* \) that form a time series: a series of draws from \( p(\theta | y) \) in which \( \theta_j^* \) may depend on \( \theta_{j'}^* \) for \( j' < j \).

In their pioneering paper Metropolis et al. (1953) allowed for serial dependence of the \( \theta_j^* \) by combining von Neumann’s idea of rejection sampling (which had itself only been published a few years earlier in 1951) with concepts from a subject in the theory of stochastic processes called Markov chains.

Combining Monte Carlo sampling with Markov chains gives rise to the name now used for this technique for solving the Bayesian high-dimensional integration problem: Markov chain Monte Carlo (MCMC).
Markov Chains

**Markov chains.** A stochastic process is just a collection of random variables \( \{\theta_t^*, t \in T\} \) for some index set \( T \), usually meant to stand for time.

In practice \( T \) can be either discrete, e.g., \( \{0, 1, \ldots\} \), or continuous, e.g., \([0, \infty)\).

**Markov chains** are a special kind of stochastic process that can either unfold in discrete or continuous time—we’ll talk here about **discrete-time Markov chains**, which is all you need for MCMC.

The **possible values** that a stochastic process can take on are collectively called the **state space** \( S \) of the process—in the simplest case \( S \) is **real-valued** and can also either be discrete or continuous.

Intuitively speaking, a Markov chain (e.g., Feller, 1968; Roberts, 1996; Gamerman, 1997) is a stochastic process unfolding in time in such a way that the past and future states of the process are independent given the present state—in other words, to figure out where the chain is likely to go next you don’t need to pay attention to where it’s been, you just need to consider where it is now.

More formally, a stochastic process \( \{\theta_t^*, t \in T\}, \ T = \{0, 1, \ldots\} \), with state space \( S \) is a **Markov chain** if, for any set \( A \subseteq S \),

\[
P(\theta_{t+1}^* \in A | \theta_0^*, \ldots, \theta_t^*) = P(\theta_{t+1}^* \in A | \theta_t^*). \tag{10}
\]

The theory of Markov chains is **harder mathematically** if \( S \) is continuous (e.g., Tierney, 1996), which is what we need for MCMC with real-valued parameters, but most of the main ideas emerge with discrete state spaces, and I’ll assume discrete \( S \) in the intuitive discussion here.
Markov Chains (continued)

Example. For a simple example of a discrete-time Markov chain with a discrete state space, imagine a particle that moves around on the integers \{\ldots, -2, -1, 0, 1, 2, \ldots\}, starting at 0 (say).

Wherever it is at time \( t \)—say at \( i \)—it tosses a (3-sided) coin and moves to \( (i - 1) \) with probability \( p_1 \), stays at \( i \) with probability \( p_2 \), and moves to \( (i + 1) \) with probability \( p_3 \), for some \( 0 < p_1, p_2, p_3 < 1 \) with \( p_1 + p_2 + p_3 = 1 \)—these are the transition probabilities for the process.

This is called a random walk (on the integers), and it's clearly a Markov chain.

Nice behavior. The most nicely-behaved Markov chains satisfy three properties:

- They're irreducible, which basically means that no matter where it starts the chain has to be able to reach any other state in a finite number of iterations with positive probability;

- They're aperiodic, meaning that for all states \( i \) the set of possible sojourn times, to get back to \( i \) having just left it, can have no divisor bigger than 1 (this is a technical condition; periodic chains still have some nice properties, but the nicest chains are aperiodic).

- They're positive recurrent, meaning that (a) for all states \( i \), if the process starts at \( i \) it will return to \( i \) with probability 1, and (b) the expected length of waiting time til the first return to \( i \) is finite.

Notice that this is a bit delicate: wherever the chain is now, we insist that it must certainly come back here, but we don't expect to have to wait forever for this to happen.
Markov Chains (continued)

The random walk defined above is clearly irreducible and aperiodic, but it may not be positive recurrent (depending on the $p_i$): it's true that it has positive probability of returning to wherever it started, but (because $S$ is unbounded) this probability may not be 1, and on average you may have to wait forever for it to return.

We can fix this by bounding $S$: suppose instead that $S = \{-k, -(k - 1), \ldots, -1, 0, 1, \ldots, k\}$, keeping the same transition probabilities except rejecting any moves outside the boundaries of $S$.

This bounded random walk now satisfies all three of the nice properties.

**The value of nice behavior.** Imagine running the bounded random walk for a long time, and look at the distribution of the states it visits—over time this distribution should settle down (converge) to a kind of limiting, steady-state behavior.

This can be demonstrated by simulation, for instance in $\mathbb{R}$, and using the bounded random walk as an example:

```r
rw.sim <- function( k, p, theta.start, n.sim, seed ) {
    set.seed( seed )

    theta <- rep( 0, n.sim + 1 )

    theta[ 1 ] <- theta.start

    for ( i in 1:n.sim ) {
        theta[ i + 1 ] <- move( k, p, theta[ i ] )
    }

    return( table( theta ) )
}
```
Markov Chain Simulation

move <- function( k, p, theta ) {

    repeat {
        increment <- sample( x = c( -1, 0, 1 ), size = 1, prob = p )

        theta.next <- theta + increment

        if ( abs( theta.next ) <= k ) {

            return( theta.next )

            break

        }
    }
}

rosalind 17> R

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

> p <- c( 1, 1, 1 ) / 3
> k <- 5
> theta.start <- 0
> seed <- c( 6425451, 9626954 )
> rw.sim( k, p, theta.start, 10, seed )

theta
0 1 2
5 5 1

> rw.sim( k, p, theta.start, 100, seed )

-2 -1 0 1 2 3 4 5
7 9 16 17 23 14 8 7
Simulation (continued)

```r
> rw.sim( k, p, theta.start, 1000, seed )
-5  -4  -3  -2  -1   0   1   2   3   4   5
  65 115 123 157 148 123 106  82  46  21  15

> rw.sim( k, p, theta.start, 10000, seed )
-5  -4  -3  -2  -1   0   1   2   3   4   5
  581  877  941  976  959 1034 1009  982 1002  959  681

> rw.sim( k, p, theta.start, 100000, seed )
-5  -4  -3  -2  -1   0   1   2   3   4   5
  6515 9879 9876 9631 9376 9712 9965 9749 9672 9352 6274

> rw.sim( k, p, theta.start, 1000000, seed )
-5  -4  -3  -2  -1   0   1   2   3   4   5
  65273 98535 97715 96708 95777 96607 96719 96361 96836 95703 63767
```

You can see that the distribution of where the chain has visited is **converging** to something close to **uniform** on \{-5,-4,\ldots,4,5\}, except for the effects of the **boundaries**.

Letting \(q_1\) denote the **limiting** probability of being in one of the 9 **non-boundary** states (-4, -3, \ldots, 3, 4) and \(q_2\) be the **long-run** probability of being in one of the 2 **boundary** states (-5, 5), on grounds of **symmetry** you can guess that \(q_1\) and \(q_2\) should satisfy

\[
9q_1 + 2q_2 = 1 \quad \text{and} \quad q_1 = \frac{3}{2}q_2,
\]

from which \((q_1, q_2) = \left(\frac{3}{31}, \frac{2}{31}\right) \approx (0.096774, 0.064516)\).

Based on the run of **1,000,001 iterations** above we would estimate these probabilities **empirically** as

\[
\begin{bmatrix}
98535+\ldots+95703 \\
(9)(1000001)
\end{bmatrix},
\begin{bmatrix}
65273+63767 \\
(2)(1000001)
\end{bmatrix}
\approx (0.096773, 0.064520).
\]
Simulation (continued)

It should also be clear that the limiting distribution does not depend on the initial value of the chain:

```r
> rw.sim( k, p, 5, 100000, seed )

-5 -4 -3 -2 -1 0 1 2 3 4 5
6515 9879 9876 9624 9374 9705 9959 9738 9678 9365 6288
```

Of course, you get a different limiting distribution with a different choice of \((p_1, p_2, p_3)\):

```r
> p <- c( 0.2, 0.3, 0.5 )

> rw.sim( k, p, 0, 10, seed )

0 1 2 3
1 3 4 3

> rw.sim( k, p, 0, 100, seed )

0 1 2 3 4 5
1 3 6 13 30 48

> rw.sim( k, p, 0, 1000, seed )

0 1 2 3 4 5
1 18 71 157 336 418

> rw.sim( k, p, 0, 10000, seed )

-5 -4 -3 -2 -1 0 1 2 3 4 5
5 16 19 30 28 74 215 583 1344 3470 4217

> rw.sim( k, p, 0, 100000, seed )

-5 -4 -3 -2 -1 0 1 2 3 4 5
5 22 53 132 302 834 2204 5502 13489 34460 42998

> rw.sim( k, p, 0, 1000000, seed )

-5 -4 -3 -2 -1 0 1 2 3 4 5
61 198 511 1380 3398 8591 22117 54872 137209 343228 428436

23
Stationary Distributions

A positive recurrent and aperiodic chain is called **ergodic**, and it turns out that such chains possess a unique **stationary** (or **equilibrium**, or **invariant**) distribution $\pi$, characterized by the relation

$$\pi(j) = \sum_i \pi(i) P_{ij}(t) \quad (12)$$

for all states $j$ and times $t \geq 0$, where $P_{ij}(t) = P(\theta^*_t = j | \theta^*_0 = i)$ is the **transition matrix** of the chain.

Informally, the stationary distribution characterizes the **behavior that the chain will settle into** after it's been run for a long time, regardless of its initial state.

**The point of all of this.** Given a parameter vector $\theta$ and a data vector $y$, the Metropolis et al. (1953) idea is to simulate random draws from the posterior distribution $p(\theta|y)$, by constructing a **Markov chain** with the following three properties:

- It should have the **same state space** as $\theta$,
- It should be **easy to simulate from**, and
- Its **equilibrium distribution** should be $p(\theta|y)$.

If you can do this, you can run the Markov chain for a long time, generating a huge sample from the posterior, and then use **simple descriptive summaries** (means, SDs, correlations, histograms or kernel density estimates) to extract any features of the posterior you want.
The Ergodic Theorem

The mathematical fact that underpins this strategy is the **ergodic theorem**: if the Markov chain \( \{\theta^*_t\} \) is ergodic and \( f \) is any real-valued function for which \( E_{\pi} |f(\theta)| \) is finite, then with probability 1

\[
\frac{1}{M} \sum_{t=1}^{M} f(\theta^*_t) \to E_{\pi}[f(\theta)] = \sum_i f(i) \pi(i), \tag{13}
\]

in which the right side is just the expectation of \( f(\theta) \) under the stationary distribution \( \pi \).

In plain English this means that—as long as the stationary distribution is \( p(\theta|y) \)—you can learn (to arbitrary accuracy) about things like posterior means, SDs, and so on just by **waiting for stationarity to kick in** and **monitoring** thereafter for a **long enough period**.

Of course, as Roberts (1996) notes, the theorem is **silent** on the two key practical questions it raises: **how long you have to wait** for stationarity, and **how long to monitor** after that.

A third practical issue is what to use for the **initial value** \( \theta^*_0 \): intuitively the **closer** \( \theta^*_0 \) is to the **center** of \( p(\theta|y) \) the **less time** you should have to wait for stationarity.

The standard way to deal with **waiting for stationarity** is to (a) run the chain from a **good starting value** \( \theta^*_0 \) for \( B \) iterations, until **equilibrium** has been reached, and (b) **discard** this initial **burn-in** period.

All of this motivates the topic of **MCMC diagnostics**, which are intended to answer the following questions:

- What should I use for the **initial value** \( \theta^*_0 \)?
- How do I know when I've reached **equilibrium**? (This is equivalent to asking **how big** \( B \) should be.)
- Once I've reached equilibrium, how big should \( M \) be, i.e., how long should I **monitor the chain** to get posterior summaries with **decent accuracy**?