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 μ is $\frac{0.04528}{\sqrt{31200}} \doteq 0.00026$.

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 θ^* 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 **Monte Carlo dataset** (see p. 26 below) consisting of 31,200 rows and one column (the θ_t^*) I add a column monitoring the values of the **derived variable** which is 1 whenever $\theta_t^* \leq q$ 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 θ^* 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 θ 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 θ_j^* may **depend on** $\theta_{j'}^*$ for $j' < j$.

In their pioneering paper Metropolis et al. (1953) allowed for **serial dependence** of the θ_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 **Markov chains**, a subject in the theory of **stochastic processes**.

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 $A \in S$,

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

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 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 `R`, and using the **bounded random walk** as an example:

```
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)

```
> 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

$$9q_1 + 2q_2 = 1 \quad \text{and} \quad q_1 = \frac{3}{2}q_2, \tag{11}$$

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

Based on the run of **1,000,001 iterations** above we would estimate these probabilities **empirically** as $\left[\frac{98535+\ldots+95703}{(9)(1000001)}, \frac{65273+63767}{(2)(1000001)}\right] \doteq (0.096773, 0.064520)$.

Simulation (continued)

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

```
> 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):

```
> 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
```

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 π, characterized by the relation

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

for all states j and times $t \geq 0$, where $P_{ij}(t) = P(\theta_t^* = j | \theta_{t-1}^* = i)$ 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 θ 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 θ,
- 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.

There is a **fourth desirable condition** as well:

- It should not be necessary to work out the **normalizing constant** for $p(\theta|y)$ to implement the algorithm, which is equivalent to saying that $p(\theta|y)$ should appear in the calculations only through **ratios** of the form $\frac{p(\theta|y)}{p(\theta'|y)}$.

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 as $m \to \infty$

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

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

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** θ_0^*: intuitively the **closer** θ_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** θ_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** θ_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**?

The Monte Carlo and MCMC Datasets

The basis of the Monte Carlo approach to obtaining **numerical approximations** to posterior summaries like means and SDs is the (weak) **Law of Large Numbers**: with IID sampling the **Monte Carlo estimates** of the true summaries of $p(\theta|y)$ are **consistent**, meaning that they can be made arbitrarily close to the truth with arbitrarily high probability as the number of monitoring iterations $m \to \infty$.

Before we look at how Metropolis et al. attempted to achieve the same goal with a **non-IID Monte Carlo approach**, let's look at the **practical consequences** of switching from IID to Markovian sampling.

Running the **IID rejection sampler** on the Berkeley PBT example above for a total of m monitoring iterations would produce something that might be called the **Monte Carlo dataset**, with one **row** for each **iteration** and one **column** for each **monitored quantity**; in that example it might look like this (MCSEs in parenthesis):

Iteration	θ	$I(\theta \leq 0.15)$
1	$\theta_1^* = 0.244$	$I_1^* = 0$
2	$\theta_2^* = 0.137$	$I_2^* = 1$
$\vdots$	$\vdots$	$\vdots$
$m = 31,200$	$\theta_m^* = 0.320$	$I_m^* = 0$
Mean	0.2183 (0.003)	0.0556 (0.0013)
SD	0.04528	—
Density Trace	(like the bottom plot on p. 14)	—

Running the **Metropolis sampler** on the same example would produce something that might be called the **MCMC dataset**.

It would have a **similar structure** as far as the **columns** are concerned, but the rows would be divided into **three phases**:

The MCMC Dataset (continued)

- Iteration 0 would be the value(s) used to **initialize** the Markov chain;

- Iterations 1 through b would be the **burn-in** period, during which the chain reaches its **equilibrium** or **stationary** distribution (as mentioned above, iterations 0 through b are generally **discarded**); and

- Iterations $(b+1)$ through $(b+m)$ would be the **monitoring** run, on which **summaries** of the posterior (means, SDs, density traces, ...) will be based.

In the Berkeley PBT example the **MCMC dataset** might look like this:

Iteration	Phase	θ	$I(\theta \leq 0.15)$
0	Initialization	$\theta_0^* = 0.200$	—
1	Burn-in	$\theta_1^* = 0.244$	—
⋮	⋮	⋮	⋮
$b = 500$	Burn-in	$\theta_b^* = 0.098$	—
$(b+1) = 501$	Monitoring	$\theta_{b+1}^* = 0.275$	$I_{b+1}^* = 0$
⋮	⋮	⋮	⋮
$(b+m) = 31,700$	Monitoring	$\theta_{b+m}^* = 0.120$	$I_{b+m}^* = 1$
Mean	(Monitoring Phase Only)	0.2177 (0.009)	0.0538 (0.004)
SD		0.04615	—
Density Trace		(like the bottom plot on p. 14)	—

Think of iteration number i in the Monte Carlo sampling process as a discrete **index of time** t, so that the columns of the MC and MCMC datasets can be viewed as **time series**.

An important concept from time series analysis is **autocorrelation**: the autocorrelation ρ_k of a **stationary** time series θ_t^* at **lag** k (see, e.g., Chatfield (1996)) is $\frac{\gamma_k}{\gamma_0}$, where γ_k is $C(\theta_t^*, \theta_{t-k}^*)$, the covariance of the series with itself k iterations in the past—this measures the degree to which the time series at any given moment **depends on its past history**.

The MCMC Dataset (continued)

IID draws from $p(\theta|y)$ correspond to **white noise**: a time series with **zero autocorrelations** at all lags.

This is the behavior of the columns in the **MC data set** on p. 26, produced by **ordinary rejection sampling**.

Because of the **Markov character** of the columns of the MCMC data set on p. 27, each column, when considered as a time series, will typically have **non-zero autocorrelations**, and because Markov chains use their present values to decide where to go next it shouldn't surprise you to hear that the typical behavior will be **(substantial) positive autocorrelations**—in other words, every time you get another draw from the Markov chain you get some **new information** about the posterior and a rehash of **old information** mixed in.

It's a **marvelous result** from time series analysis (the Ergodic Theorem for Markov chains on p. 25 is an example of this fact) that all of the usual descriptive summaries of the posterior are still **consistent** as long as the columns of the MCMC data set form **stationary** time series.

In other words, provided that you can achieve the **four goals** back on p. 24 which Metropolis et al. set for themselves, and provided that you only do your monitoring **after the Markov chain has reached equilibrium**, the MCMC approach and the IID Monte Carlo approach are **equally valid** (they both get the right answers), but they may well differ on their **efficiency** (the rate per iteration, or per CPU second, at which they learn about the posterior may not be the same); and if, as is typically true, the columns of the MCMC dataset have **positive autocorrelations**, this will translate into **slower learning** (larger MCSEs) than with IID sampling (compare the MCSEs on pages 26 and 27).

The Metropolis Algorithm

Metropolis et al. were able to create what people would now call a **successful MCMC algorithm** by the following means (see the excellent book edited by Gilks et al. (1996) for many more details about the MCMC approach).

Consider the **rejection sampling method** given above in (8) as a mechanism for generating realizations of a time series (where as above time indexes iteration number).

At any time t in this process you make a draw θ^* from the **proposal distribution** $g(\theta|y)$ (the normalized version of the envelope function G) and either accept a **"move"** to θ^* or reject it, according to the **acceptance probability** $\frac{p(\theta^*|y)}{G(\theta^*|y)}$; if accepted the process moves to θ^*, if not you **draw again** and discard the rejected draws until you do make a successful move.

As noted above, the stochastic process thus generated is an **IID (white noise)** series of draws from the **target distribution** $p(\theta|y)$.

Metropolis et al. had the following **beautifully simple idea** for how this may be generalized to situations where IID sampling is difficult: **they allowed the proposal distribution at time t to depend on the current value θ_t of the process**, and then—to get the right stationary distribution—if a proposed move is rejected, instead of discarding it **the process is forced to stay where it is for one iteration before trying again**.

The resulting process is a **Markov chain**, because (a) the draws are now dependent but (b) all you need to know in determining where to go next is **where you are now**.

Metropolis-Hastings

Letting θ_t stand for **where you are now** and θ^* for **where you're thinking of going**, in this approach there is **enormous flexibility** in the choice of the proposal distribution $g(\theta^*|\theta_t, y)$, even more so than in ordinary rejection sampling.

The original Metropolis et al. idea was to work with **symmetric** proposal distributions, in the sense that $g(\theta^*|\theta_t, y) = g(\theta_t|\theta^*, y)$, but Hastings (1970) pointed out that this could easily be **generalized**; the resulting method is the **Metropolis-Hastings** (MH) algorithm.

Building on the Metropolis et al. results, Hastings showed that you'll get the **correct stationary distribution** $p(\theta|y)$ for your Markov chain by making the following choice for the **acceptance probability**:

$$\alpha_{MH}(\theta^*|\theta_t, y) = \min \left\{ 1, \frac{\frac{p(\theta^*|y)}{g(\theta^*|\theta_t, y)}}{\frac{p(\theta_t|y)}{g(\theta_t|\theta^*, y)}} \right\}. \qquad (14)$$

It turns out that the proposal distribution $g(\theta^*|\theta_t, y)$ can be **virtually anything** and you'll get the **right equilibrium distribution** using the acceptance probability (14); see, e.g., Roberts (1996) and Tierney (1996) for the mild regularity conditions necessary to support this statement.

A **summary** of the method is on the next page.

It's instructive to compare (15) with (8) to see how heavily the MH algorithm **borrows from ordinary rejection sampling**, with the key difference that the proposal distribution is allowed to **change over time**.

Notice how (14) **generalizes** von Neumann's acceptance probability ratio $\frac{p(\theta^*|y)}{G(\theta^*|y)}$ for ordinary rejection sampling: the crucial part of the new MH acceptance probability becomes the **ratio of two von-Neumann-like ratios**, one for **where you are now** and one for **where you're thinking of going** (it's equivalent to work with g or G since the normalizing constant **cancels** in the ratio).