Monte Carlo Methods

Interested in

$$E\left[f(X)\right] = \int f(x)\, d\mu(x)$$

Examples:

- Type I error rate of a hypothesis test
- Mean width of a confidence interval procedure
- Evaluating a likelihood
- Finding posterior mean and variance

Often calculating these will be difficult.

Approximate with

$$\frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

where $x_1, \ldots, x_n$ is sampled from $\mu(X)$.

Under certain regularity conditions,

$$\frac{1}{n}\sum_{i=1}^{n} f(x_i) \to E\left[f(X)\right]$$

Issues:

- probability measure being integrated over
- function being integrated
- sampling scheme
- form of convergence

Focus: Sampling Schemes

- Independently and identically distributed (IID)
- Importance sampling
- Sequential importance samplers (SIS)
- Markov Chain Monte Carlo (MCMC)
    - Gibbs sampling
    - Metropolis – Hastings (M-H)
    - Reversible jump
    - Bridge sampling
    - etc
- and so on

Choice is often driven by $\mu(X)$, e.g.,

IID infeasible leads to use of SIS or MCMC

SIS may work but Gibbs sampler has reducible chain

M-W works when SIS has poorly behaved importance sampling weights.

etc

There is no one Monte Carlo approach that will solve every problem.

Want to develop a set of tools, that can be used, possibly in combination, for a wide range of problems.

Need to recognize when each of them should work and when modifications are needed.

Basic Simulation Methodology

Pseudo-random numbers:

"Random" numbers generated on computers are not random but generated by deterministic algorithms.

Basic problem: generate u_i, $0 < u_i < 1$, that appear to be an iid sample from the $U(0,1)$ distribution.

Once you have these, you can simulate from "any" distribution.

Uniform deviates

Instead of generating $U(0,1)$, most generators actually generate integers ($U(0,m-1)$ or $U(1,m-1)$) and then convert these to the interval $(0,1)$.

Numerically more stable and faster.

Multiplicative Congruential Generators:

Generate integer sequence $\{k_i\}$ by

$$k_{i+1} = ak_i \bmod m$$

for suitably chosen positive integers a and m, where $b \bmod m$ is the remainder from dividing b by m.

If $a^{m-1} = 1 \bmod m$ and $a^l \neq 1 \bmod m$ for $0 < l < m - 1$ and if k_0 is a positive integer that isn't a multiple of m, then it can be shown that $k_1, \ldots, k_{m-1}$ will be a permutation of $\{1, 2, \ldots, m - 1\}$ (a is said to be a primitive root of unity mod m).

The period of this generator is $m - 1$.

In general the period is the number of values until the generator starts to repeat.

Linear Congruential Generators

$$k_{i+1} = (ak_i + b) \bmod m$$

for suitable integers a, b, and m.

Good generators should have

- long periods
- low (near 0) correlations
- give samples that look uniform

This holds for any generator, not just congruential generators.

Choices for m, a, and b

1) $m = 2^{31} - 1$: largest prime integer that can be stored on most computers
   - $a = 7^5$ (IMSL, early versions of Matlab)
   - a = 950,706,376 (IMSL option, shown to be good by Fishman & Moore)
2) $m = 2^{32}$: number of integers that can be represented on most computers.
   - Can't get full period if b = 0 since m is even.
   - Maximum period of 2^{30} can be achieved if $a = 5 + 8l$ for some l. Common choice is a = 69069

- Can get full period of 2^{30} when $b \neq 0$, such as with $a = 69069$ and $b = 23606797$.

Problems with congruential generators:

- Must have some autocorrelation
- n dimensional uniformity (how close do n tuples of consecutive draws achieve uniformity in the n-dimensional unit cube).
- congruential generators tend to give n-vectors that concentrate near hyperplanes in n-dimensional space for some n.

Example: RANDU generator

- IBM SYSTEM/360 generator

$$k_{i+1} = \left(2^{16} + 3\right) k_i \bmod 2^{31}$$

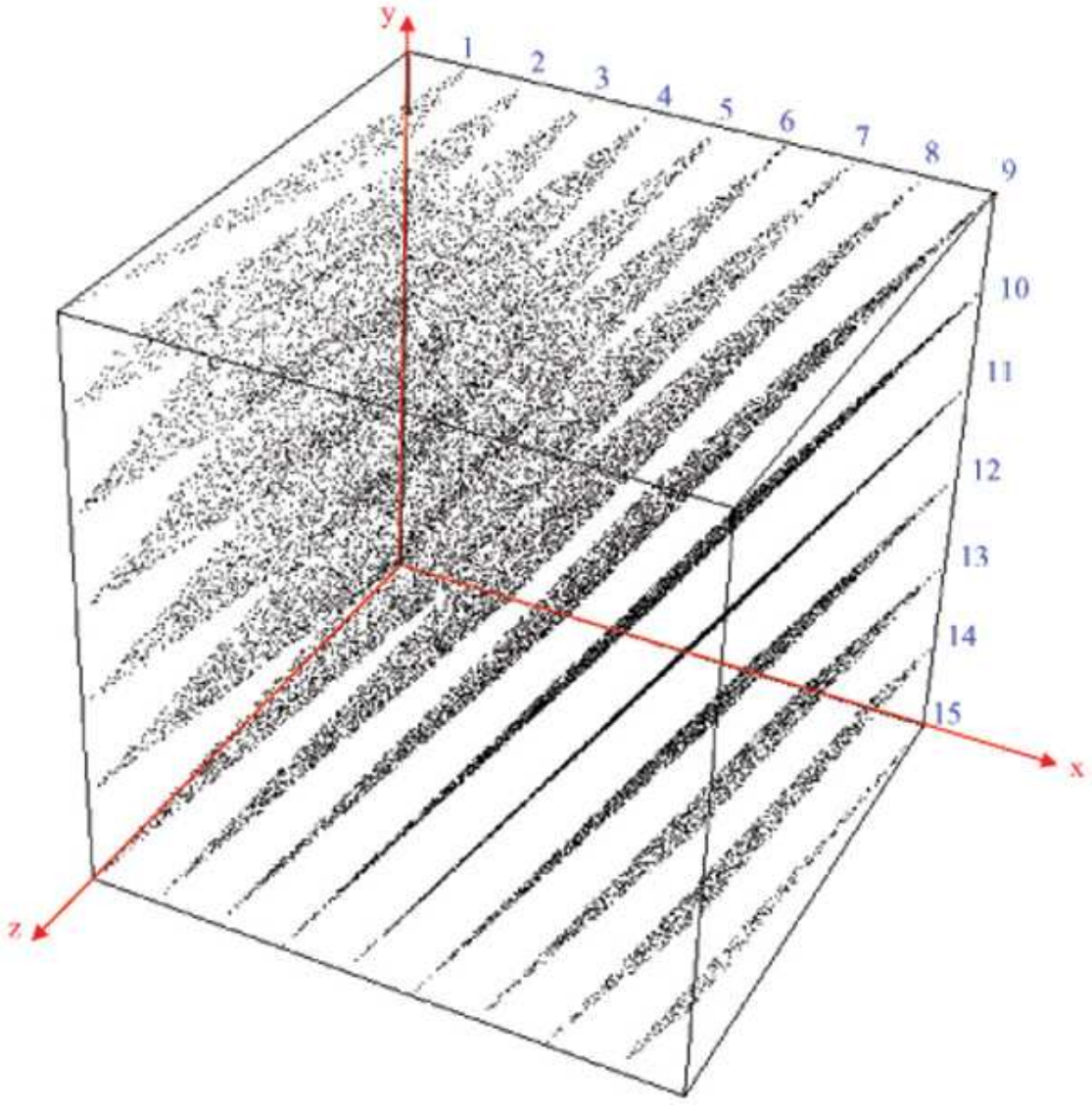
- Has the property that

$$9k_i - 6k_{i+1} + x_{i+2} = 0 \bmod 2^{31}$$

Proof:

$$9x_i - 6ax_i + a^2 x_i = (a-3)^2 x_i$$
$$= 2^{32} x_i = 2^{31} \times 2x_i$$

- Realizations of triples must fall on one of 15 planes, 2^{31} apart



from <http://www.unf.edu/ccec/cis/CIShtml/RANDUinfo.html>

- This generator is still around is system software.

  From HP documentation
  <http://h18009.www1.hp.com/fortran/docs/lrm/lrm0315.htm>

  GSL (Gnu Scientific Library)
  <http://www.gnu.org/software/gsl/manual/gsl-ref_17.html#SEC271>

  In GSL, its there for backward compatibility with old code that people use and historical completeness.

- With congruential generators, the leading bits tend to be more random than the low order bits, so one shouldn't treat subsets of the bits in the representation as separate random numbers
- Standard congruential generators have periods that are too short for some statistical applications.

Shuffling algorithm

- Initialize: $s(i) = u_i; i = 1, \ldots, N$ and set $y = s(N)$.
- Generate a new value u and set $j = \text{int}(yN) + 1$, where $\text{int}(x)$ is the largest integer ≤ x.
- Set $y = s(j)$, $s(j) = u$, and return y as the uniform deviate.

The idea behind this scheme is that a permutation of uniforms is still uniform.

By combining generators with long, but not equal periods, a new generator with a much longer period can be created.

Example: ran2 (Numerical recipies due to L'Ecuyer)

Generator 1 (v_i): a = 40014, m = 2147483563. Uses shuffle algorithm with N = 32.

Generator 2 (w_i): a = 40692, m = 2147483399.

Returns

$$u_i = (v_i - w_i) I(v_i \geq w_i) + (1 - v_i + w_i) I(v_i < w_i)$$

The period of this generator is the product of the periods of the two streams, divided by any common factors.

The period is about $2.3 \times 10^{18} \approx 2^{61}$.

Recursive generators

$$k_{i+1} = a_1 k_i + \ldots + a_l k_{i+1-l} \bmod m$$

Linear combination of the previous l values.

Maximum period: $m^l - 1$

Fibonacci generators

$$u_i = u_{i-17} - u_{i-5}$$
$$u_i = u_{i-97} - u_{i-33}$$

If lagged difference < 0, add 1 to result.

Shift / Tausworthe generators

Based on binary expansion of integers

$$j = \sum_{l=1}^{32} b_l 2^{l-1}$$

The idea is to shift the sequence and then combine it with the original sequence by exclusive or.

As part of the S-Plus generator, they use the following shift generator

```
double ush(j)
  unsigned long *j;
{
  double v = 4294967296;  /* v =
2^32 */
  *j = *j ^ (*j >> 15);
  *j = *j ^ (*j << 17);
  return(*j/v);
}
```

`*j >> 15` shifts the bits right by 15 and replaces bits 1 to 15 with 0. ^ is exclusive or so `*j = *j ^ (*j >> 15)` replaces the vector j with

$$\left(b_{32},\ldots,b_{18},b_{17}+b_{32},\ldots,b_1+b_{16}\right)\bmod 2$$

The S-Plus generator combines this with a congruential generator with $a = 69069$, $m = 2^{32}$ with an exclusive or operation.

R has 6 different uniform generators. The default is the Mersenne Twister, a generalized feedback shift register (GFSR) generator with a period of $2^{19937} - 1 \approx 10^{6000}$. To see the others available in R, see `help(RNGkind)`.

Bottom line: Creating a good generator is an art and a science.

The constants used in congruential generators and the lags used in Fibonacci and Tausworthe generators are not arbitrary. Poor choices can lead to very nonrandom behaviour (such as with RANDU).

Diehard tests

A set of procedures for testing random number generators created by George Marsaglia.

Generating from non-uniform distributions

For a cumulative distribution function (CDF) $P[X \le x] = F(x)$, the inverse CDF is defined by

$$F^{-1}(u) = \inf\{x : F(x) \le u\}$$

For continuous RVs,

$$\begin{aligned} P[F(X) \le u] &= P[X \le F^{-1}(u)] \\ &= F(F^{-1}(u)) = u \end{aligned}$$

so $F(X) \sim U(0,1)$. Conversely, if $U \sim U(0,1)$

$$P[F^{-1}(u) \le x] = F(x)$$

Thus, given an iid $U(0,1)$ sample $\{u_1, \ldots, u_n\}$, an iid sample $\{x_1, \ldots, x_n\}$ from F can be obtained by $x_i = F^{-1}(u_i)$.

Example: Cauchy

$$F(x; \mu, \sigma) = \frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{x - \mu}{\sigma}\right)$$

$$F^{-1}(u; \mu, \sigma) = \mu + \sigma \tan(\pi(u - 1/2))$$

Example: Exponential

$$F(x;\mu) = 1 - \exp(-x/\mu)$$

$$F^{-1}(u;\mu) = -\mu \log(1-u)$$

Sometimes its easier to work with the survivor function $S(x) = 1 - F(x)$. Since U and $1 - U$ both have uniform distributions, $S^{-1}(u)$ will also be a draw from F.

So

$$S^{-1}(u;\mu) = -\mu \log(u)$$

will also give a draw from an exponential distribution.

Not all distributions (e.g. normal or gamma) have nice closed form expressions for F^{-1}.

The density is usually of a nice form, but often the CDF, and thus its inverse aren't. However there are often good analytical approximations to F^{-1}, so these can be used instead.

For example with the standard normal, a rational function approximation could be used (R and Matlab definitely do, S-Plus probably).

Note that the Inverse CDF method isn't commonly used for most distributions as it tends to be slow.

Functions like log, sin, cos, etc tend to be somewhat expensive to calculate.

Though surprisingly in R, it is the default for normals. However there are 4 other methods available (see `help(RNGkind)`). In S-Plus and Matlab, I don't know what they are doing.

Discrete Distributions:

Suppose that the distribution has support points $s_1, s_2, \ldots, s_k$ (k possibly infinite) and set

$$p_j = \sum_{i=1}^{j} P[X = s_i] = P[X \le s_j]$$

Then independent observations x_i can be generated by setting $x_i = s_j$ if $p_{j-1} < u_i \le p_j$ (where $p_0 = 0$).

Essentially this is inverting the CDF.

If k is small, then only a few comparisons need to be made. However if k is big, many comparisons may be needed (if u_i is close to 1).

In this case, other methods are needed.

Relationships with other distributions:

Examples:

- $X \sim N\left(\mu, \sigma^2\right)$ then $Y = e^X$ is lognormal
- $X \sim N(0,1)$ then $Y = X^2$ is χ_1^2
- $X_\alpha \sim Gamma(1, \alpha)$, $X_\beta \sim Gamma(1, \beta)$, then $Y = \dfrac{X_\alpha}{X_\alpha + X_\beta} \sim Beta(\alpha, \beta)$

Polar Coordinates and the Normal Distribution

Suppose that X, Y are independent $N(0,1)$ variables and consider the polar coordinates transformation given by

$$X = R\cos\theta, Y = R\sin\theta; \quad R \geq 0, 0 \leq \theta < 2\pi$$

It is easily shown that $\theta \sim U(0, 2\pi)$, $R^2 \sim \chi^2_2$, and they are independent. Also

$$P[R > r] = \left[R^2 > r^2\right] = \exp\left(-r^2/2\right)$$

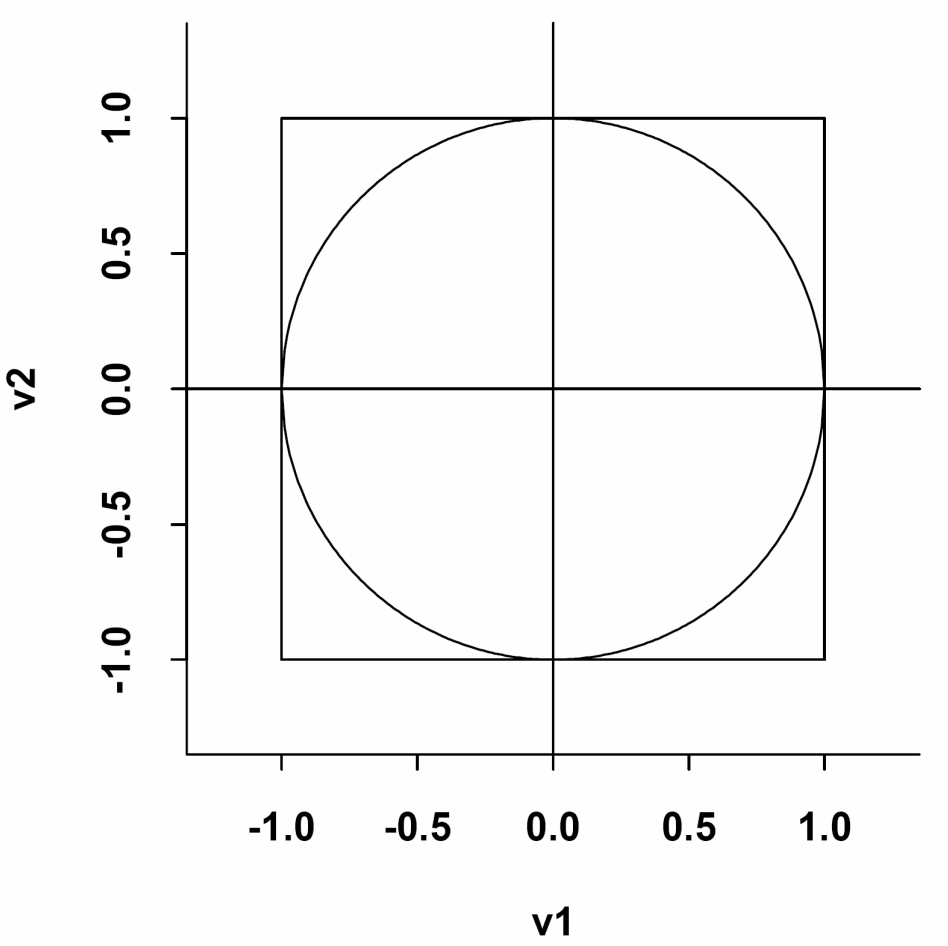
Box-Muller Method

- Generate $u_1, u_2 \sim U(0,1)$
- Set $R = \sqrt{-2\log u_1}$ and $\theta = 2\pi u_2$
- Set $x_1 = R\cos\theta$ and $x_2 = R\sin\theta$

Marsaglia Polar Method

The sin and cos functions (which are slow) can be avoided.

Underlying the Box-Muller method is to pick uniform angle independently of a radius. This can be recast in terms of picking points in the unit circle



Let $v_1, v_2 \sim U(-1,1)$ such that $v_1^2 + v_2^2 \leq 1$

Let θ be the counterclockwise angle from the positive v_1 axis to the point (v_1, v_2). By symmetry, $\theta \sim U(0, 2\pi)$ and

$$\cos\theta = \frac{u_1}{\sqrt{u_1^2 + u_2^2}} \text{ and } \sin\theta = \frac{u_2}{\sqrt{u_1^2 + u_2^2}}$$

Also $P\left[v_1^2 + v_2^2 \leq u\right] = \pi u/\pi = u$, so $v_1^2 + v_2^2 \sim U(0,1)$

Finally θ and $v_1^2 + v_2^2$ are independent (again by symmetry).

Thus, given (v_1, v_2), two normal deviates are given by

$$u = v_1^2 + v_2^2,\ w = \sqrt{-2\log u / u},$$

$$x_1 = v_1 w,\ x_2 = v_2 w$$

This is an example of the Acceptance-Rejection Method.

For this example, the fraction of pairs (v_1, v_2) that are accepted = $\pi/4$ = 0.785, the ratio of the area of the circle to the square.

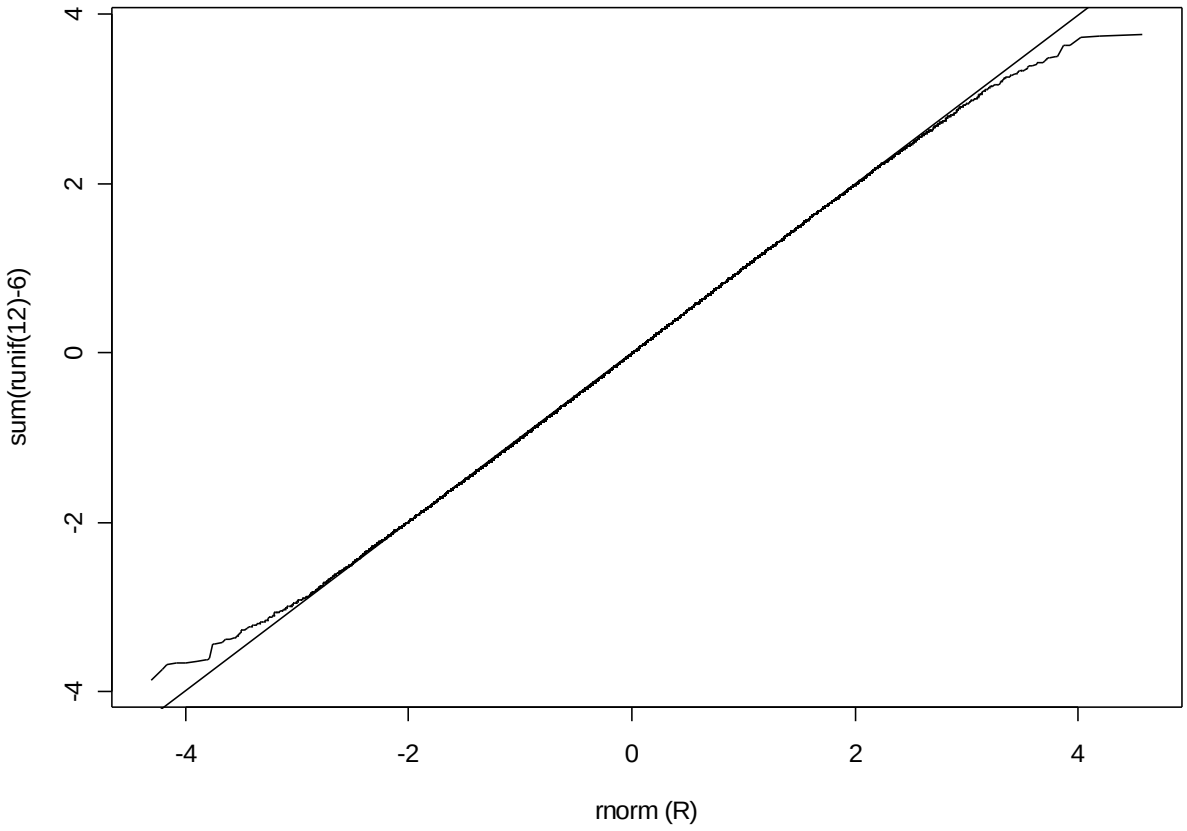
An old generator $N(0,1)$

Let $u_1, \ldots, u_{12} \sim U(0,1)$ and $z = \sum u_i - 6$

$$E[u_i] = \frac{1}{2}; \mathrm{Var}(u_i) = \frac{1}{12}$$

$$E\left[\sum u_i\right] = 6; \mathrm{Var}\left(\sum u_i\right) = 1$$

So z has mean 0 and standard deviation 1 and is approximately normal (very approximately) by the "Central Limit Theorem"



This generator has tails which are too light.

An example of a consequence of this would be to incorrectly estimate the coverage rate of a confidence interval procedure.