MATH 598

Bayesian Inference, Computational Methods and Monte Carlo

Dr David A. Stephens

Department of Mathematics & Statistics
Room 1225, Burnside Hall

david.stephens@mcgill.ca

Part 2
Bayesian Computation

Monte Carlo: basic principles

- Monte Carlo methods
- Importance sampling
- Rejection sampling
- Sampling importance resampling
- Variance reduction techniques

Monte Carlo: basic principles

The basic principles of Monte Carlo are based on simple ideas from frequentist statistics:

- *Laws of large numbers:* Suppose $X_1, \ldots, X_n, \ldots$ are iid random variables. Then, usually,

$$\frac{1}{n}\sum_{i=1}^{n} g(X_i) \xrightarrow[\text{p}]{\text{a.s.}} \mathbb{E}[g(X)]$$

  as $n \longrightarrow \infty$.

- *Ergodic Theorems:* For sequence $X_1, \ldots, X_n, \ldots$, and under mild conditions on the joint distribution of random variables,

$$\frac{1}{n}\sum_{i=1}^{n} g(X_i) \xrightarrow[\text{p}]{\text{a.s.}} \mathbb{E}[g(X)]$$

  as $n \longrightarrow \infty$.

Monte Carlo: basic principles

- *Central Limit Theorems:* Under mild conditions on the joint distribution of random variables $X_1, \ldots, X_n, \ldots$, as $n \longrightarrow \infty$,

$$a_n \left(\frac{1}{n} \sum_{i=1}^{n} g(X_i) - b_n \right) \xrightarrow{d} \mathcal{N}(\mu, \sigma^2)$$

for suitable choices of the sequences $\{a_n\}$ and $\{b_n\}$.

Monte Carlo: basic principles

Essentially, standardized sums of random variables have stable long-run behaviour. For example, for a distribution with finite second moment

$$\overline{X} \xrightarrow{p} \mathbb{E}[X] \qquad\qquad \frac{1}{n}\sum_{i=1}^{n} X_i^2 \xrightarrow{p} \mathbb{E}[X^2]$$

as $n \longrightarrow \infty$, and so on.

Monte Carlo: basic principles

More explicitly, in the i.i.d. case,

$$\frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{p} \int x\, dF(x) = \int x f(x)\, dx$$

where $f(x)$ is the density of the X variables. Recall that

$$\frac{1}{n}\sum_{i=1}^{n} x_i = \int x\, d\widehat{F}_n(x)$$

where $\widehat{F}_n$ is the empirical cdf.

Monte Carlo: basic principles

Note

In essence, Monte Carlo methods replace integrals with respect to F by integrals with respect to $\widehat{F}_n$, and then rely on the convergence of the latter to the former.

Monte Carlo: basic principles

We consider approximating the integral

$$\int g(x)f(x)\,dx$$

for probability density $f(x)$ by the sum

$$\frac{1}{N}\sum_{i=1}^{N} g(x_i)$$

where $x_1, \ldots, x_N$ are an i.i.d sample from f.

We need to establish

- whether this approximation works
- the accuracy of the approximation
- how the samples from $f(x)$ are obtained.

Monte Carlo: Validity

The strong law of large numbers ensures that

$$\frac{1}{N}\sum_{i=1}^{N} g(X_i) \xrightarrow{a.s.} \mathbb{E}[g(X)] = \int g(x)f(x)\,dx$$

provided the expectation $\mathbb{E}[g(X)]$ exists. Does this ever go wrong ?

Monte Carlo: Validity

Consider computing

$$\int_0^1 \frac{1}{x} \sin(2\pi/x)\, dx$$

by sampling $X_i \sim \mathit{Uniform}(0, 1)$, and then computing

$$\frac{1}{N} \sum_{i=1}^{N} \frac{1}{X_i} \sin(2\pi/X_i)$$

Monte Carlo: Validity

The integral can be computed as

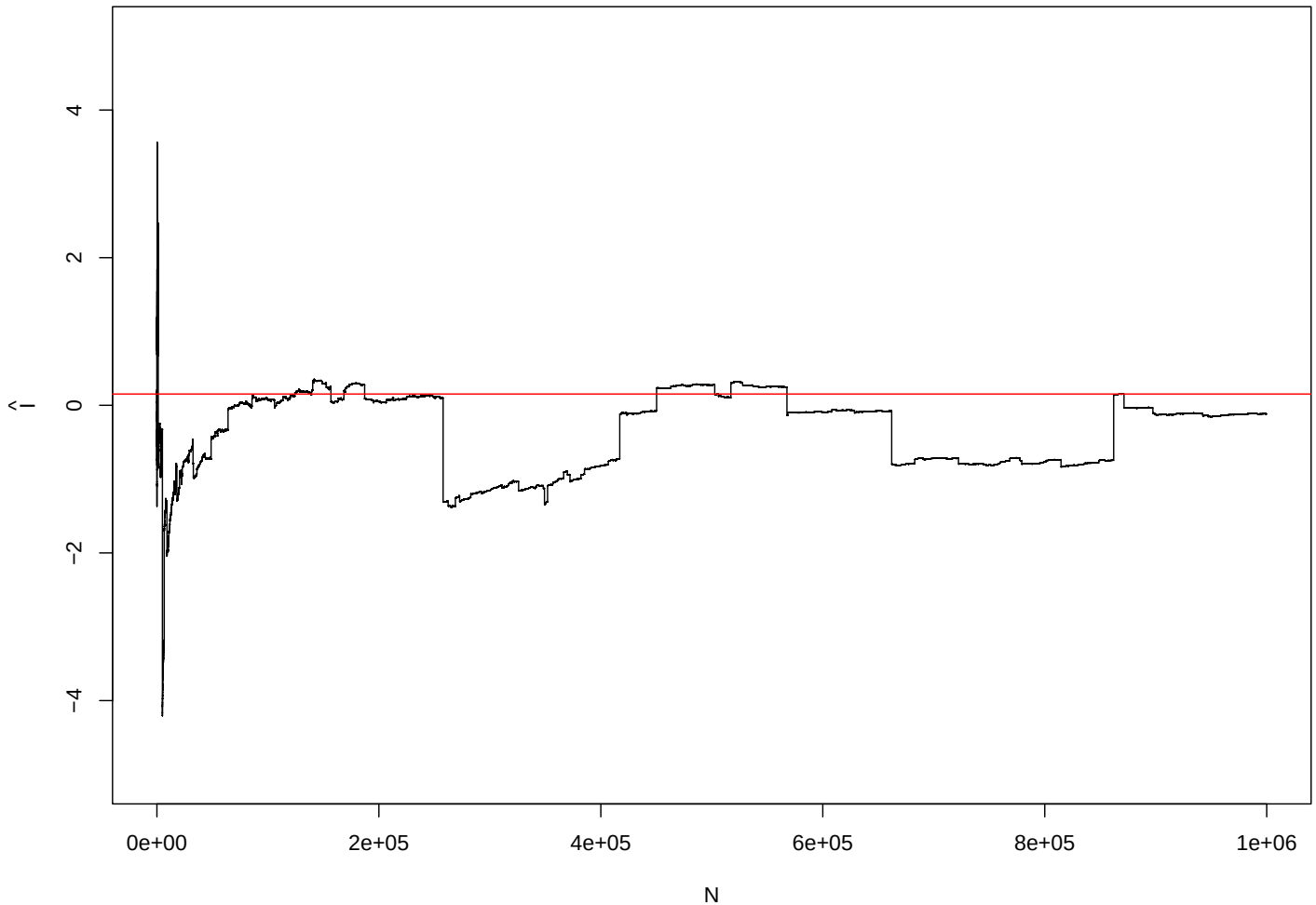
$$\begin{aligned}\int_0^1 \frac{1}{x}\sin(2\pi/x)\,dx &= \int_1^\infty \frac{\sin(2\pi t)}{t}\,dt \\ &= \int_0^\infty \frac{\sin(t)}{t}\,dt - \int_0^{2\pi} \frac{\sin(t)}{t}\,dt \\ &= \mathrm{Si}(\infty) - \mathrm{Si}(2\pi)\end{aligned}$$

where Si(.) is a special function (the *sine integral*).

We have that $\mathrm{Si}(\infty) = \pi/2$; the numerical value of the integral is 0.1526.

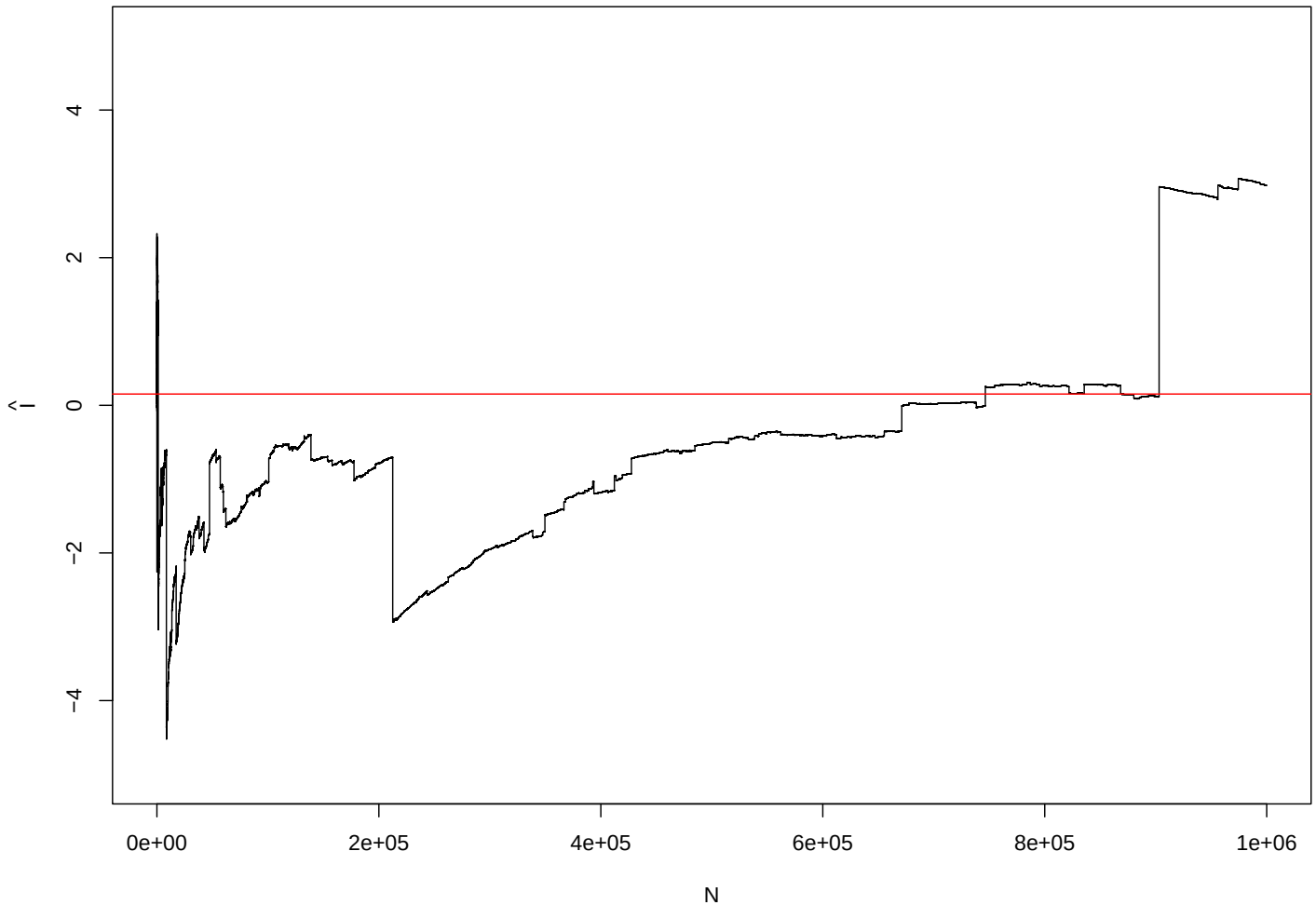
Monte Carlo: Validity

Run 1:



Monte Carlo: Validity

Run 2:



Monte Carlo: Validity

Occasional large values of

$$Y_i = \frac{1}{X_i} \sin(2\pi/X_i)$$

cause the Monte Carlo average to not converge as N gets large.

A sufficient condition for strong convergence is

$$\int |g(x)| f(x)\, dx < \infty$$

which does *not* hold here.

Monte Carlo Estimation: Statistical Properties

In situations where the Monte Carlo estimator

$$\widehat{I}_N(g) = \frac{1}{N} \sum_{i=1}^{N} g(X_i)$$

of $\mu(g) = \mathbb{E}[g(X)]$ does converge satisfactorily, *and* a central limit theorem applies, we have that

$$\sqrt{N}(\widehat{I}_N(g) - \mu(g)) \xrightarrow{d} \mathcal{N}(0, V(g))$$

where

$$V(g) = \mathrm{Var}[g(X)] = \int (g(x) - \mu(g))^2 \, f(x) \, dx$$

The Monte Carlo estimator exhibits $o_P(\sqrt{N})$ convergence.

Monte Carlo Estimation: Example

Example: Normal Interval Probabilities

Consider $f(x) = \phi(x)$ (standard Normal pdf), the function

$$g(x) = \begin{cases} 0 & x \leqslant x_0 \\ 1 & x > x_0 \end{cases}$$

for some fixed $x_0 > 0$, and the integral

$$\int_{-\infty}^{\infty} g(x) f(x)\, dx = 1 - \Phi(x_0).$$

The integral is the tail probability from ordinate x_0.

Monte Carlo Estimation: Example

Example: Normal Interval Probabilities

The Monte Carlo estimator is this case is

$$\widehat{I}_N(g) = \frac{1}{N}\sum_{i=1}^{N} g(X_i) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{(x_0,\infty)}(X_i)$$

where $X_1, \ldots, X_N \sim \mathcal{N}(0,1)$, which has $\mathbb{E}[\widehat{I}_N(g)] = 1 - \Phi(x_0)$, and variance

$$\frac{\Phi(x_0)(1-\Phi(x_0))}{N}$$

which *decreases* with *increasing* x_0. The variance estimate is

$$\frac{\widehat{I}_N(g)(1-\widehat{I}_N(g))}{N}$$

Monte Carlo Estimation: Example

Example: Normal Interval Probabilities

A problem that occurs when x_0 is large is finite sample bias; the probability of obtaining no $X_i > x_0$, and thus an estimate $\widehat{I}_N(g) = 0$, is

$$p_0 = \{\Phi(x_0)\}^N$$

which can remain large even for large N if x_0 is large enough.

In the following table, N_0 is N such that $p_0 = 10^{-6}$:

x_0	2	4	6	8	10	12
N_0	6.00×10^2	4.36×10^5	1.40×10^{10}	2.21×10^{16}	1.81×10^{24}	7.77×10^{33}

Monte Carlo Estimation: Example

Example: Mixture Integrand

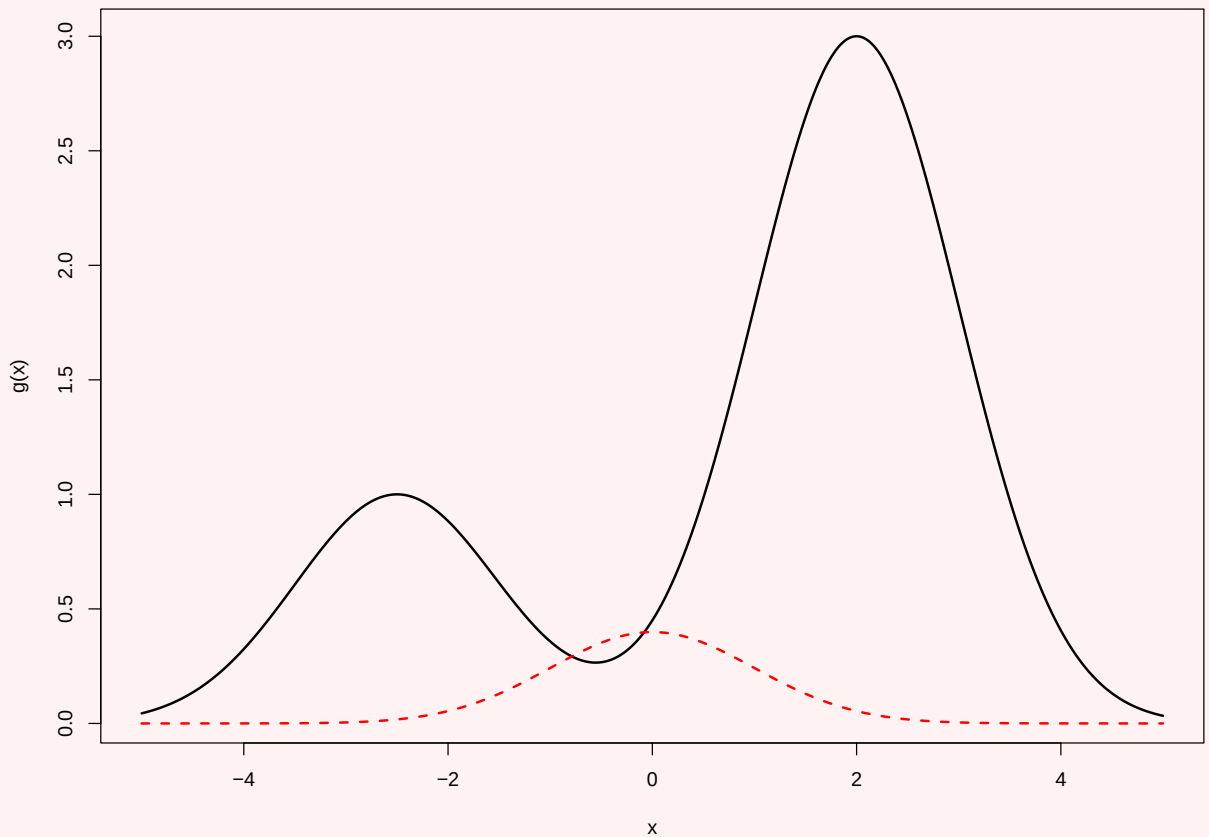
Consider

$$g(x) = \exp\left\{-\frac{1}{2}(x+2.5)^2\right\} + 3\exp\left\{-\frac{1}{2}(x-2)^2\right\}$$

Monte Carlo Estimation: Example

Example: Mixture Integrand

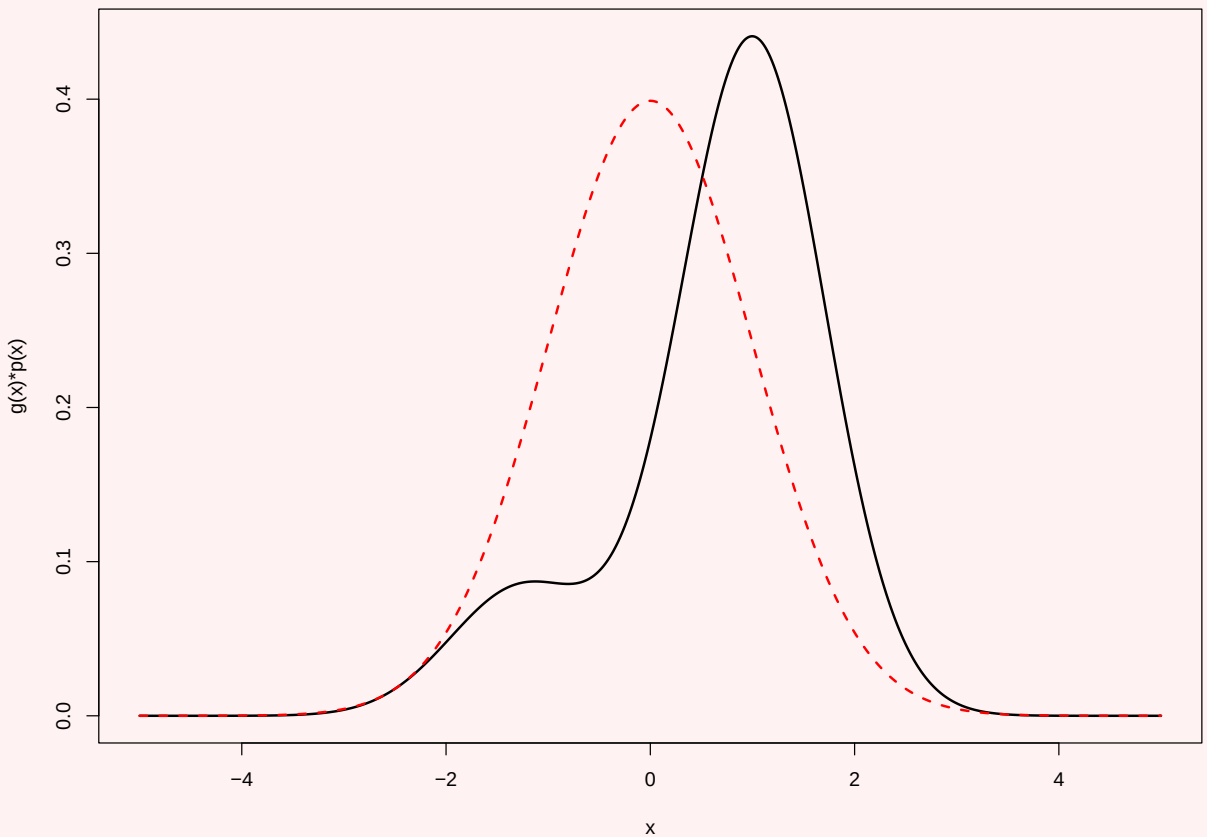
Plot of integrand $g(x)$ (solid) and $f(x)$ (dashed):



Monte Carlo Estimation: Example

Example: Mixture Integrand

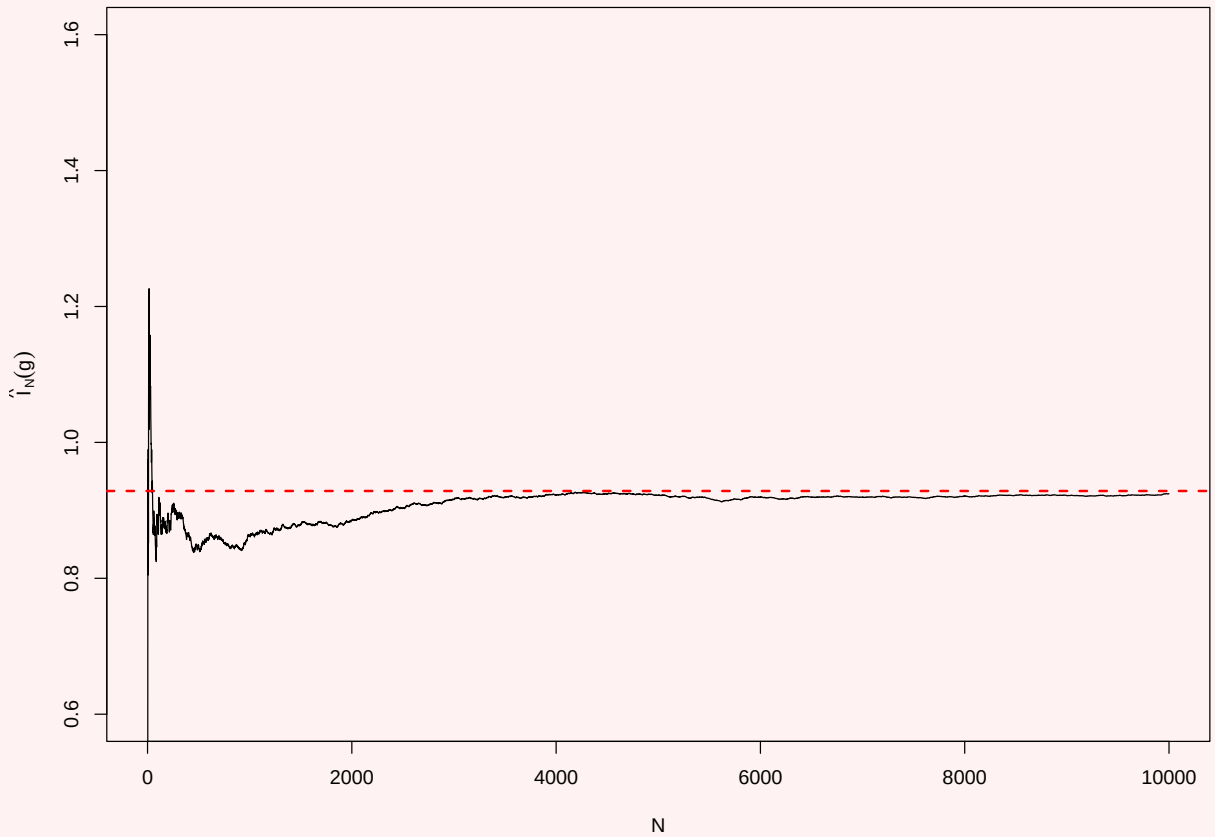
Plot of integrand $g(x)f(x)$ (solid) and $f(x)$ (dashed):



Monte Carlo Estimation: Example

Example: Mixture Integrand

Convergence of estimator for $N =$ up to 10000.



Monte Carlo Estimation

Motivated by the deterministic approximation

$$\int g(x)f(x)\,dx \doteq \sum_{j=0}^{k} w_j g(x_j)$$

where weights are determined by $f(x)$, to minimize the error in the approximation, it should be advantageous to choose design points $x_0, \ldots, x_k$ where g is largest.

In a Monte Carlo setting, it seems clear that the estimator will converge more quickly and have lower variance for finite N when $f(x)$ generates points in regions where $g(x)$ is large in magnitude.

Monte Carlo Estimation

Example:

See `knitr 4`

Importance sampling

The identity

$$\int g(x)f(x)\,dx = \int g(x)f(x)\frac{f_0(x)}{f_0(x)}\,dx = \int \frac{g(x)f(x)}{f_0(x)}f_0(x)\,dx$$

where f_0 is a probability density with support including the support of f, demonstrates that

$$\mathbb{E}_f[g(X)] = \mathbb{E}_{f_0}\left[\frac{g(X)f(X)}{f_0(X)}\right]$$

so that an estimator of the LHS is

$$\widehat{I}_N^{(f_0)}(g) = \frac{1}{N}\sum_{i=1}^{N}\frac{g(X_i)f(X_i)}{f_0(X_i)}$$

where $X_1, \ldots, X_N \sim f_0(.)$.

Importance sampling

$\widehat{I}_N^{(f_0)}$ is termed the *importance sampling* estimator.

f_0 is termed the *importance sampling density*. By careful choice of f_0, the estimator can have better performance than the Monte Carlo estimator in finite samples.

Note that

$$\widehat{I}_N^{(f_0)}(g) = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{f_0(X_i)}g(X_i) = \frac{1}{N}\sum_{i=1}^{N} w_0(X_i)g(X_i)$$

say, where

$$w_0(X_i) = \frac{f(X_i)}{f_0(X_i)}$$

is the *importance sampling weight*.

Importance sampling

Note that

$$\mathbb{E}_{f_0}\left[\frac{f(X)}{f_0(X)}\right] = \int f(x)\,dx = 1$$

so

$$\mathbb{E}_{f_0}\left[\frac{1}{N}\sum_{i=1}^{N} w_0(X_i)\right] = 1$$

although for any realization

$$\frac{1}{N}\sum_{i=1}^{N} w_0(x_i) \neq 1$$

in general.

Importance sampling: Example

Example: Normal tail probability

Consider the Monte Carlo estimation of

$$\int_{x_0}^{\infty} \phi(x)\, dx = 1 - \Phi(x_0)$$

For x_0 large, the Monte Carlo estimator is prone to finite sample bias.

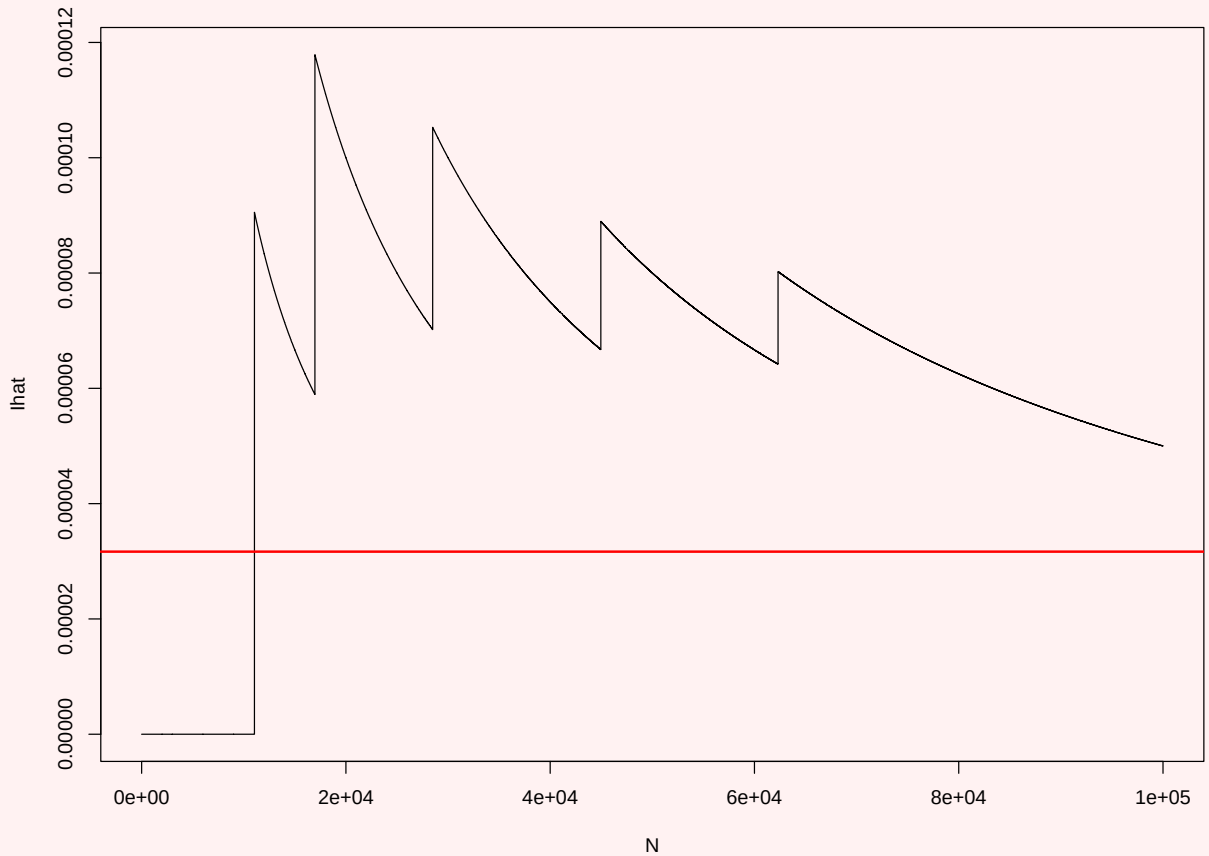
For $x_0 = 4$, $1 - \Phi(x_0) = 3.167 \times 10^{-5}$, and the probability of getting $\widehat{I}_N = 0$ for $N = 100000$ is

$$\{\Phi(x_0)\}^{100000} = 0.0421$$

Importance sampling: Example

Example: Normal tail probability

Monte Carlo estimate $\widehat{I}_N(g)$ of $1 - \Phi(x_0)$, with $x_0 = 4$:



Importance sampling: Example

Example: Normal tail probability

Consider using the importance sampling density

$$f_0(x) = \lambda \exp\{-\lambda(x - x_0)\} \qquad x > x_0$$

This density has support identical to the tail region of interest, so in the importance sampling estimator, we have

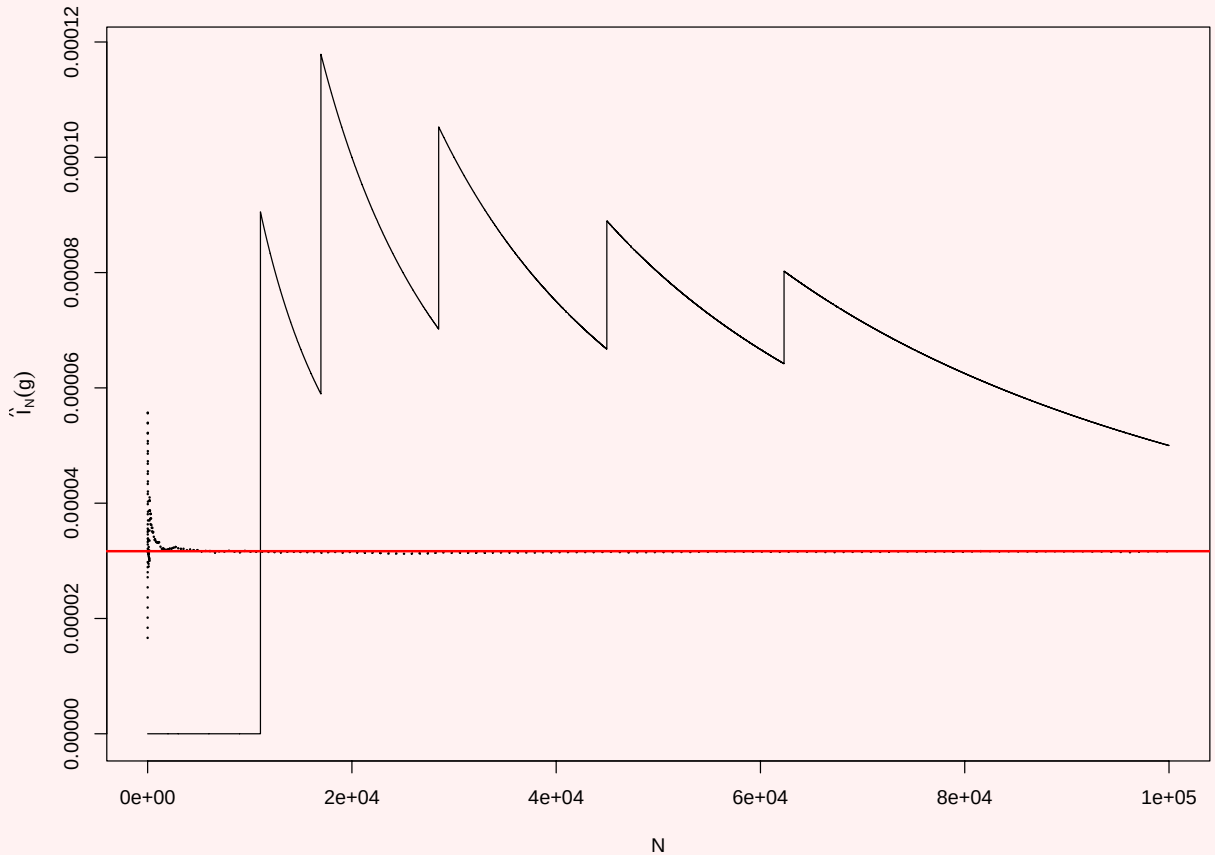
$$\widehat{I}_N^{(f_0)} = \frac{1}{N} \sum_{i=1}^{N} \frac{g(X_i) f(X_i)}{f_0(X_i)} = \frac{1}{N\lambda} \sum_{i=1}^{N} \phi(X_i) \exp\{\lambda(X_i - x_0)\}$$

as $g(X_i) = 1$ for all variates sampled from f_0.

Importance sampling: Example

Example: Normal tail probability

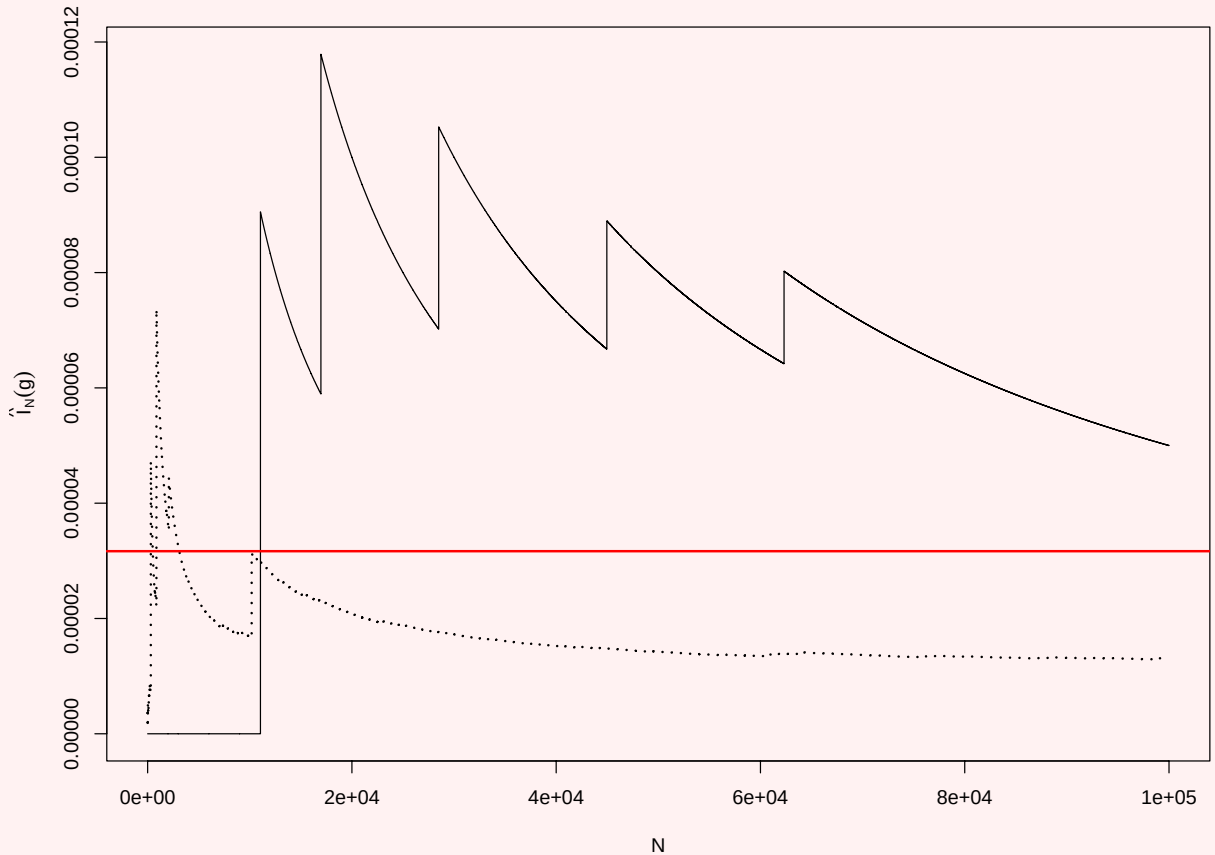
IS estimate $\widehat{I}_N^{(f_0)}(g)$ (dotted) for $\lambda = 10$



Importance sampling: Example

Example: Normal tail probability

IS estimate $\widehat{I}_N^{(f_0)}(g)$ (dotted) for $\lambda = 100$



Importance sampling: Example

Example: Normal tail probability

Consider using the importance sampling density $f_0(x) \equiv t(5)$, the Student-t density with five degrees of freedom.

A similar importance sampling estimator can be defined for this density; in this case, the tail-region indicator function

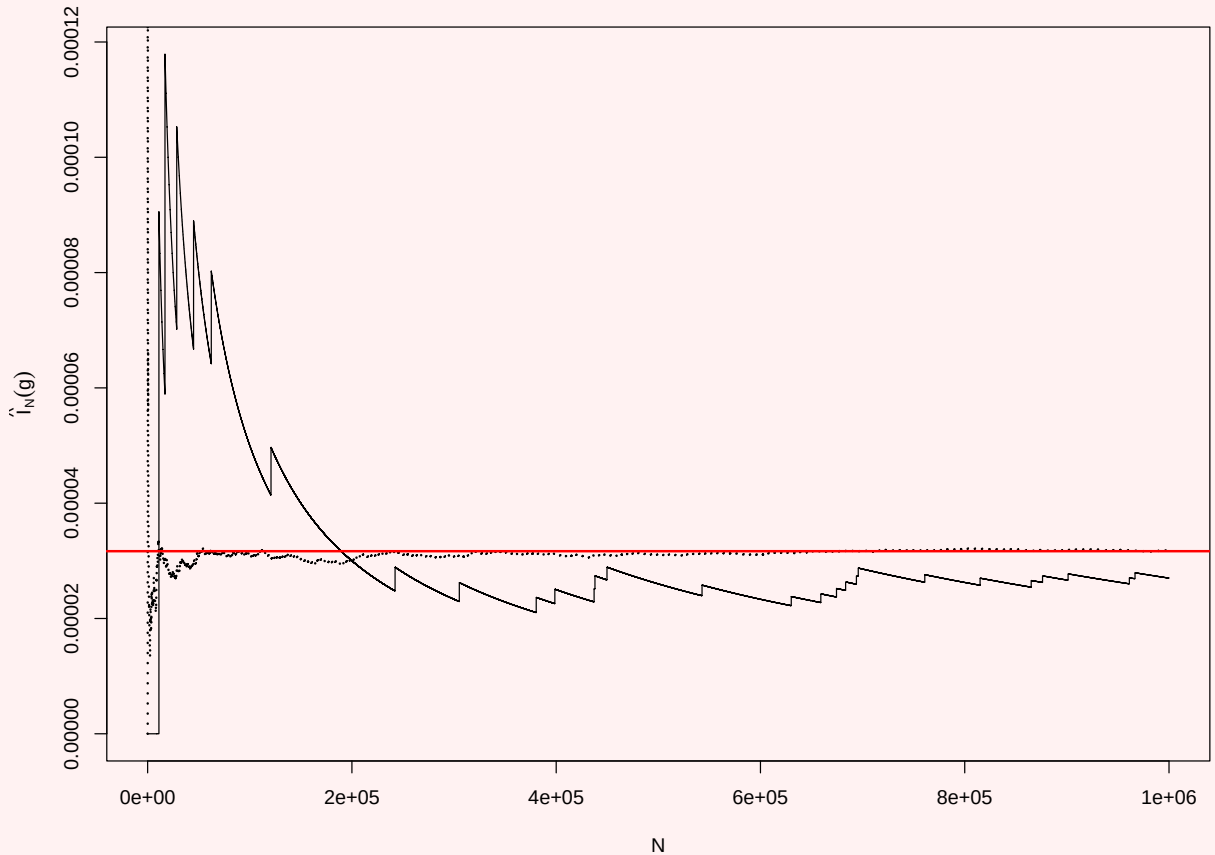
$$g(x) = \begin{cases} 0 & x \leqslant x_0 \\ 1 & x > x_0 \end{cases}$$

must be included in the computation of $\widehat{I}_N^{(f_0)}(g)$.

Importance sampling: Example

Example: Normal tail probability

IS estimate $\widehat{I}_N^{(f_0)}(g)$ (dotted) for $f_0 \equiv t(5)$



Optimal Importance Sampling

There are many possible choices for the importance sampling density f_0. We now seek guidelines for choosing f_0 optimally.

Note first that the variance of $\widehat{I}_N^{(f_0)}(g)$ is finite if and only if

$$\frac{g(X)f(X)}{f_0(X)}$$

has finite variance, that is, if and only if

$$\mathbb{E}_{f_0}\left[\left\{\frac{g(X)f(X)}{f_0(X)}\right\}^2\right] = \int_{-\infty}^{\infty}\left\{\frac{g(x)f(x)}{f_0(x)}\right\}^2 f_0(x)\,dx$$

is finite.

Optimal Importance Sampling

This equates to

$$\int_{-\infty}^{\infty} \frac{\{g(x)f(x)\}^2}{f_0(x)}\, dx$$

being finite. Note also that

$$\mathbb{E}_{f_0}\left[\left\{\frac{g(X)f(X)}{f_0(X)}\right\}^2\right] = \mathbb{E}_f\left[\{g(X)\}^2\,\frac{f(X)}{f_0(X)}\right]$$

Now, if $f(x)/f_0(x)$ is *unbounded* on the support of f, then these expectations, and the variance of $\widehat{I}_N^{(f_0)}(g)$, are *not finite*.

Therefore, for f with unbounded support, we must ensure that $f(x)/f_0(x)$ stays bounded on $\mathbb{R}$, particularly in the *tails*.

Optimal Importance Sampling

To optimize the importance sampling procedure, we inspect the variance of the IS estimator.

$$\mathrm{Var}_{f_0}\left[\frac{g(X)f(X)}{f_0(X)}\right] = \mathbb{E}_{f_0}\left[\left\{\frac{g(X)f(X)}{f_0(X)}\right\}^2\right] - \left\{\mathbb{E}_{f_0}\left[\frac{g(X)f(X)}{f_0(X)}\right]\right\}^2.$$

But

$$\mathbb{E}_{f_0}\left[\frac{g(X)f(X)}{f_0(X)}\right] = \mathbb{E}_f\left[g(X)\right]$$

does not depend on f_0, so we must minimize the first term. Note that

$$\mathbb{E}_{f_0}\left[\left\{\frac{g(X)f(X)}{f_0(X)}\right\}^2\right] \geqslant \left\{\mathbb{E}_{f_0}\left[\frac{|g(X)|f(X)}{f_0(X)}\right]\right\}^2 = \left\{\int |g(x)|f(x)\,dx\right\}^2$$

by Jensen's inequality.

Optimal Importance Sampling

This is a lower bound on the variance of the estimator which is independent of f_0. By the logic of Jensen's inequality, the inequality becomes an equality when

$$\frac{|g(x)|f(x)}{f_0(x)}$$

is a constant, that is, when

$$f_0(x) \propto |g(x)|f(x) \quad \therefore \quad f_0(x) = \frac{|g(x)|f(x)}{\int |g(u)|f(u)\,du}.$$

However, this is an infeasible choice, as we do not know the denominator.

Optimal Importance Sampling

Note that, in general, if $X_1, \ldots, X_N, \ldots \sim f_0$, then

$$\frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{f_0(X_i)} \xrightarrow{a.s.} \int_{-\infty}^{\infty}\frac{f(x)}{f_0(x)}f_0(x)\,dx = 1$$

so therefore

$$\widetilde{I}_N^{(f_0)}(g) = \frac{\sum_{i=1}^{N}\frac{g(X_i)f(X_i)}{f_0(X_i)}}{\sum_{i=1}^{N}\frac{f(X_i)}{f_0(X_i)}} \xrightarrow{a.s.} \mathbb{E}[g(X)]$$

also, if the expectation exists. For finite N, this estimator $\widetilde{I}_N^{(f_0)}(g)$ is biased, but asymptotically unbiased. It may have smaller variance than $\widehat{I}_N^{(f_0)}(g)$

Optimal Importance Sampling

It seems appealing to combine this with the optimality result. The estimator

$$\widetilde{I}_N^{(f_0)}(g) = \frac{\sum_{i=1}^{N} \frac{g(X_i)f(X_i)}{f_0(X_i)}}{\sum_{i=1}^{N} \frac{f(X_i)}{f_0(X_i)}} = \frac{\sum_{i=1}^{N} \frac{g(X_i)}{|g(X_i)|}}{\sum_{i=1}^{N} \frac{1}{|g(X_i)|}}$$

is feasible. When $g(.)$ is positive

$$\widetilde{I}_N^{(f_0)}(g) = \frac{N}{\sum_{i=1}^{N} \frac{1}{g(X_i)}}$$

that is, the *harmonic mean* estimator. Unfortunately, this estimator often has poor properties.

Optimal Importance Sampling

However, to get as close to the variance bound as possible, a sensible objective is to choose f_0 so that

$$\frac{|g(x)|f(x)}{f_0(x)}$$

is almost constant, such that the variance is *finite*.

This suggests designing f_0 to have high density whenever the original integrand $|g(x)|f(x)$ is large, subject to the constraint

$$\frac{f(x)}{f_0(x)} < M \qquad \text{or} \qquad \mathbb{E}_f\left[\frac{f(X)}{f_0(X)}\right] < M$$

for some finite bound M.

Importance Sampling in Higher Dimensions

All of the previous results carry across to the case where the target integral is an integral in dimension higher than one.

- $g(\mathbf{x})$ is a scalar function of vector argument $\mathbf{x}$,
- $f(\mathbf{x})$ and $f_0(\mathbf{x})$ are multivariate densities.

In higher dimensions, in general, many more random samples are needed to obtain sufficient accuracy than in the univariate case.

Random number generation

Monte Carlo and Importance sampling both require ready access to random samples from univariate or multivariate distributions.

There are many straightforward techniques available to obtain random samples from standard distributions once a random sample of Uniform random variables is available:

- *CDF inversion:* If $U \sim Uniform(0,1)$, and F_X is a cdf with inverse F_X^{-1}, then

$$X = F_X^{-1}(U) \sim F_X$$

(the *probability integral transform*). If F_X is discrete, define

$$F_X^{-1}(u) = \max\{x : F_X(x) \leqslant u\}$$

Random number generation

- *Transformation:*
  - Exponential, Gamma, Beta
  - Normal
    - Box-Müller
    - Polar Marsaglia
  - Student, Fisher, Multivariate Normal
- *Summation:* It is possible to generate using sums of independent identically distributed random variables.

R has many random number generation tools for standard distributions.

Uniform random number generation

Generation of truly random uniform variates mechanically is challenging; statistical packages rely on *pseudo-random* uniform generation:

- sequences $\{u_n\}$ that are generated by some *deterministic* mechanism, but that appear to be random even under severe statistical testing.

Pseudo-random number generators are typically built on deterministic equations of the form

$$u_{n+1} = h(u_1, \ldots, u_n)$$

constructed to that u_{n+1} is apparently not predictable from previous elements.

Congruential Generators

A (linear) *congruential generator* takes the form

$$u_{n+1} = au_n + b \qquad \mod c$$

for integers a, b, c, where $0 < a < c$ and $0 \leqslant b < c$. The initialization of the sequence is achieved by setting the seed u_0, $0 \leqslant u_0 < c$. The recursion generates integers on $\{0, 1, \ldots, c-1\}$, which can be transformed into uniform random variates by rescaling.

Care is needed in the choice of the constants, but provided

- a and b are coprime,
- $a - 1$ is divisible by all prime factors of c
- $a - 1$ is a multiple of 4 if c is a multiple of 4.

the generator has full cycle length c. Typically $c = 2^{32}$ is chosen.

Congruential Generators: defects

Congruential generators can perform adequately, but have known defects. They are principally favoured because of simplicity of computation.

Most obviously, the generated points can be shown to exhibit specific forms of serial dependence. For examples, certain subsequences of points can be demonstrated to lie on a hyperplanes (and hence be predictable).

Predictability is an undesirable property for randomness, but may not be too problematic in Monte Carlo for certain integrals.

Recursion/shift Generators

More recently, vector recursion generators have superseded congruential generators. These algorithms take linear (matrix) combinations of current vectors to produce new vectors in the sequence.

A favoured shift generator is the *Mersenne Twister*, which is the default in R, which has a cycle length of

$$2^{19937} - 1.$$

This generator produces very reliable uniform variates that are virtually indistinguishable from truly random numbers.

Rejection sampling

Suppose that we wish to sample from an arbitrary density $f(x)$, but this is not straightforward directly. Suppose instead we have easy access to variates from the density $f_0(x)$, where

$$\frac{f(x)}{f_0(x)} < M \qquad \text{for all } x$$

Rejection sampling

Then the following algorithm produces variates from f:

1. generate x from f_0
2. generate u from $Uniform(0, 1)$
3. if

$$u \leqslant \frac{f(x)}{Mf_0(x)}$$

   accept x as a variate from f; if this inequality is not met, return to 1.

Rejection sampling

This is the *rejection sampling* (or *accept-reject*) algorithm. The density f_0 is the *proposal density*.

It works for *any* bound M, but is most efficient (has fewest rejections) when M is as small as possible, for example, if

$$M = \sup_x \frac{f(x)}{f_0(x)}$$

If $f(x)$ is bounded, and has support which is a bounded subset $\mathcal{X}$ of $\mathbb{R}$, then f_0 can be chosen to be the Uniform density on $\mathcal{X}$, although this is not necessarily the optimal choice.

Rejection sampling: Efficiency

Note that

$$\begin{aligned}
\Pr[X \text{ is accepted}] &= \Pr\left[U \leqslant \frac{f(X)}{Mf_0(X)}\right] \\
&= \int_{-\infty}^{\infty} \left\{ \int_0^{f(x)/(Mf_0(x))} du \right\} f_0(x)\, dx \\
&= \int_{-\infty}^{\infty} \frac{f(x)}{Mf_0(x)} f_0(x)\, dx \\
&= \frac{1}{M} \int_{-\infty}^{\infty} f(x)\, dx = \frac{1}{M}
\end{aligned}$$

so ideally M should be as small as possible.

Rejection sampling: Efficiency

Note also neither f nor f_0 need to be normalized for this algorithm to be valid:

- If $f(x) = mg(x)$ and $f_0(x) = m_0 g_0(x)$, then

$$\frac{f(x)}{f_0(x)} = \frac{m}{m_0}\frac{g(x)}{g_0(x)} < M$$

  or

$$\frac{g(x)}{g_0(x)} < M' = \frac{mM}{m_0}$$

  is the rejection sampling bound.
- We proceed by bounding $g(x)/g_0(x)$.
- The acceptance probability is now indeterminate, however, by monitoring the empirical acceptance rate, an estimate of m/m_0 can be obtained.

Rejection sampling

Example: Normal mixture

Consider sampling from the normal mixture

$$f(x) = \frac{1}{4}\phi(x+2) + \frac{3}{4}\phi(x-1)$$

where $\phi(.)$ is the standard normal pdf. Consider rejection sampling from this density using

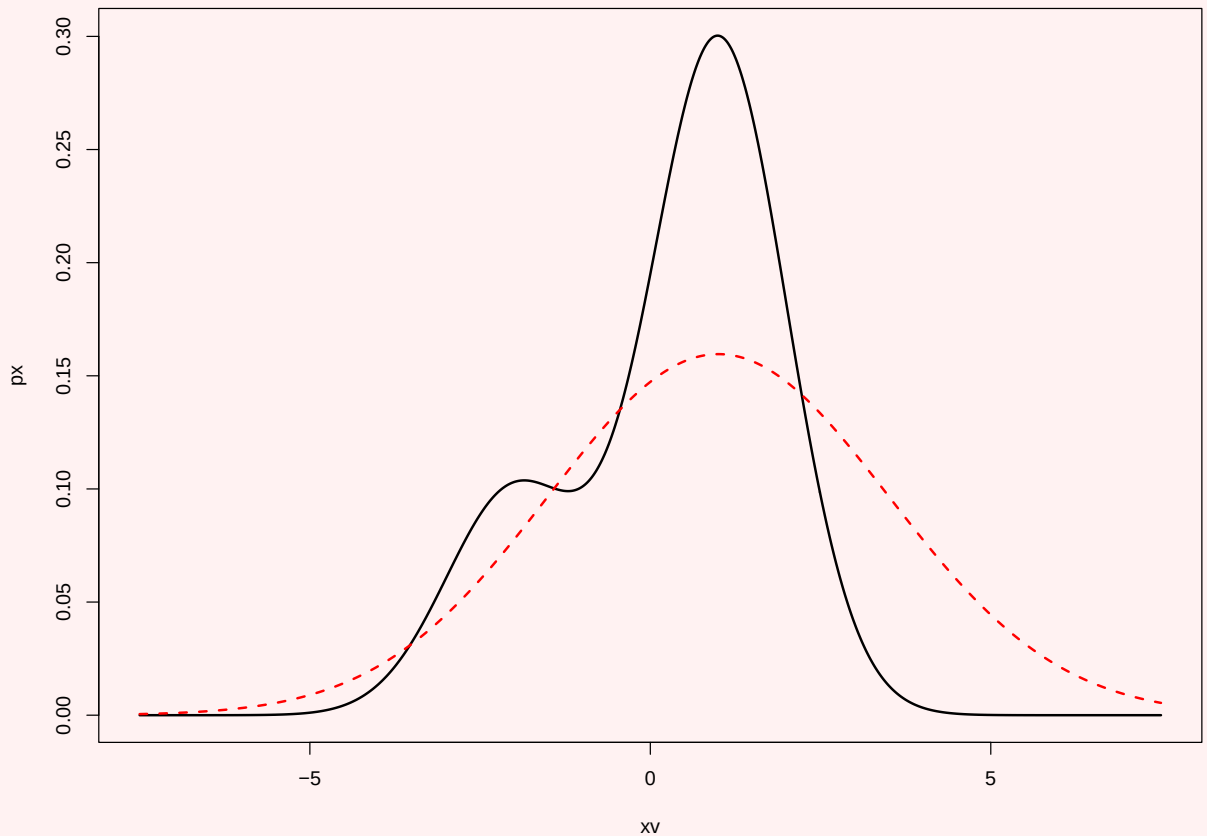
$$f_0(x) = \frac{1}{\sigma}\phi\left(\frac{x-1}{\sigma}\right)$$

for some variance σ^2.

Rejection sampling

Example: Normal mixture

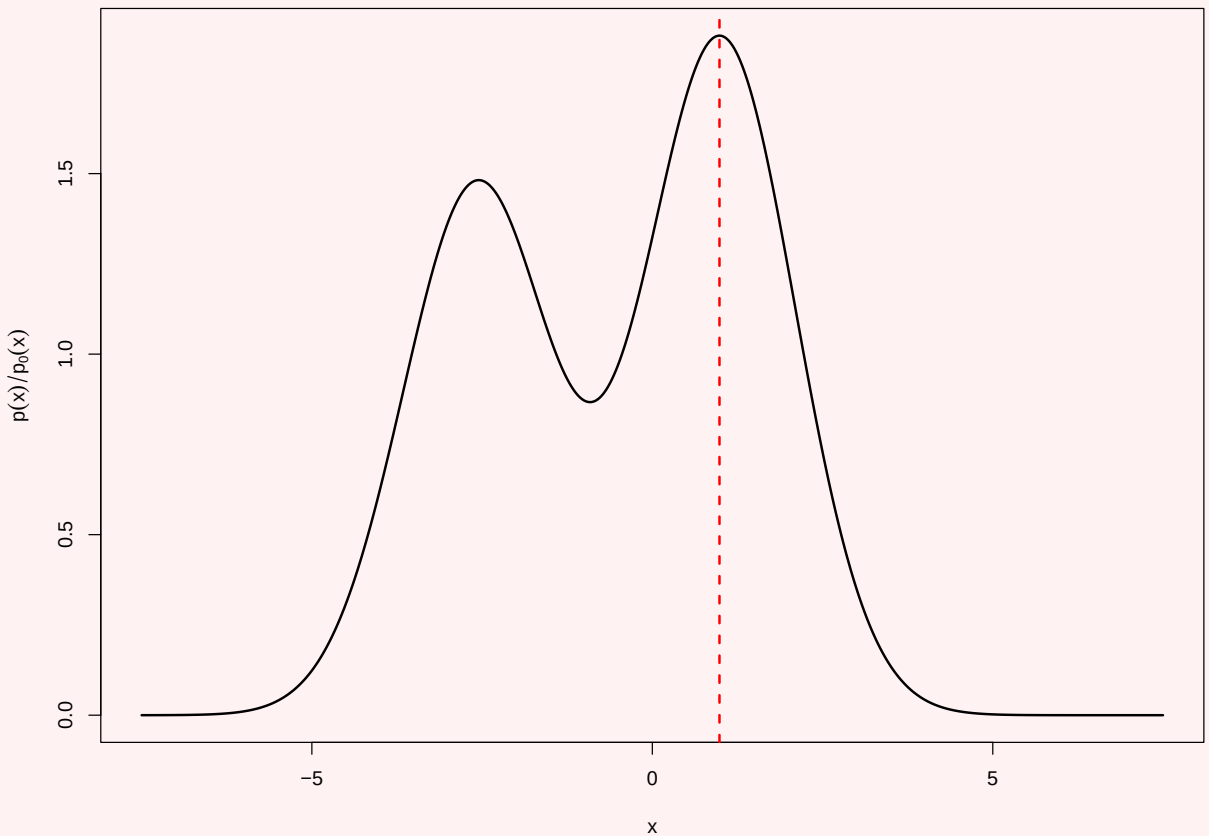
Target $f(x)$ (solid) and $f_0(x)$ (dashed) with $\sigma = 2.5$.



Rejection sampling

Example: Normal mixture

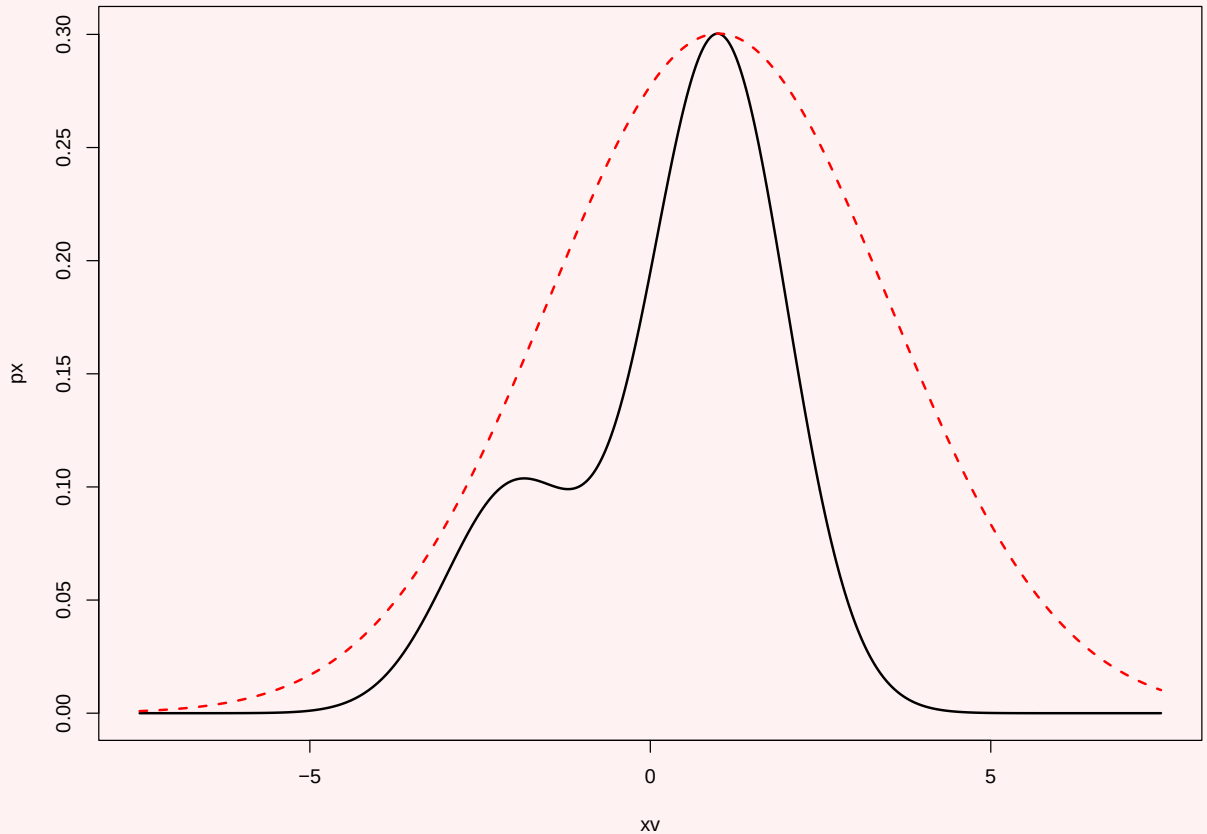
$f(x)/f_0(x)$ maximized at $x = 0.9863$, yielding $M = 1.8821$:



Rejection sampling

Example: Normal mixture

$f(x)$ bounded by $Mf_0(x)$:



Rejection sampling

If x is a variate from f_0, then $f_0(x)$ is the value of the density at that variate, and $Mf_0(x)$ is the scaled version. Consider a vertical slice at x which is the line segment

$$(x, 0) \longrightarrow (x, Mf_0(x))$$

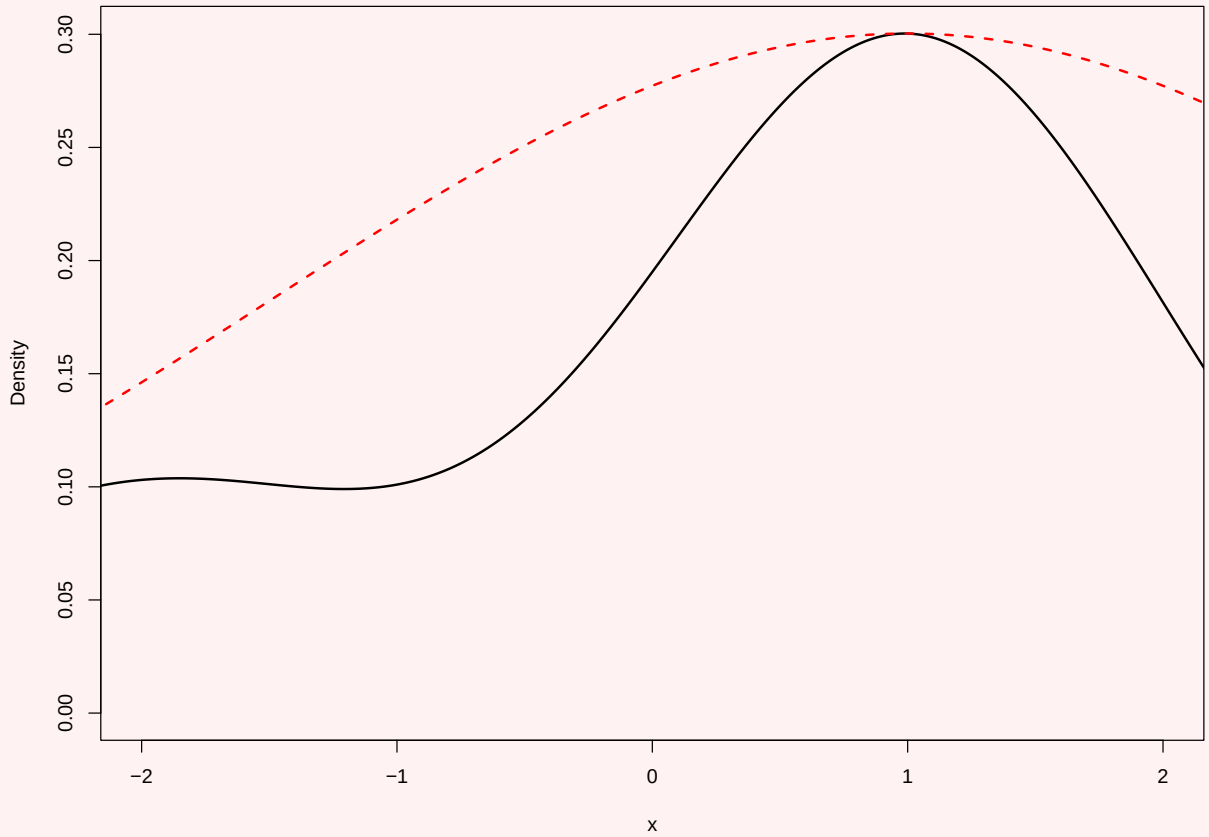
By assumption $f(x) \leqslant Mf_0(x)$. Them, if u is simulated from $Uniform(0, 1)$, then $uMf_0(x)$ is the random portion of the vertical slice, and if

$$f(x) \leqslant uMf_0(x)$$

then $f(x)$ is *below* the random point on the line segment, and hence is rejected.

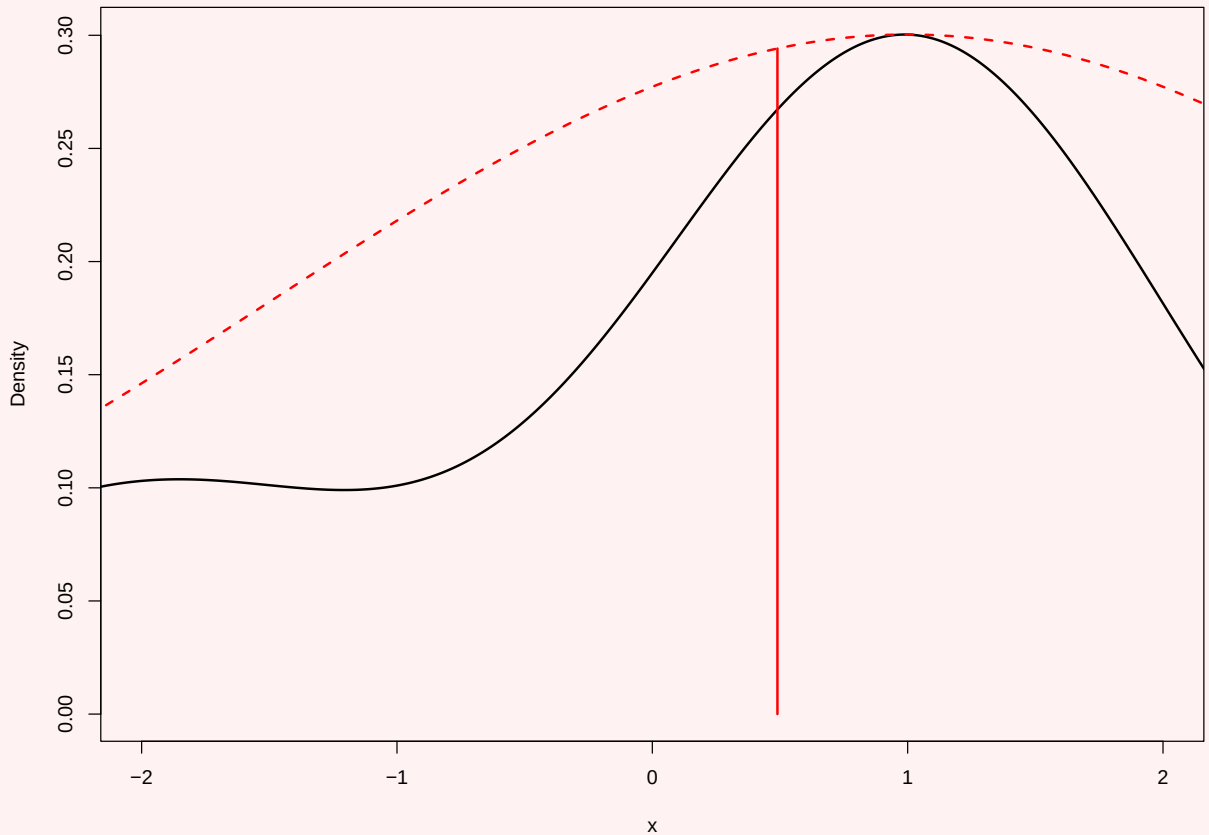
Rejection sampling

Example: Normal mixture



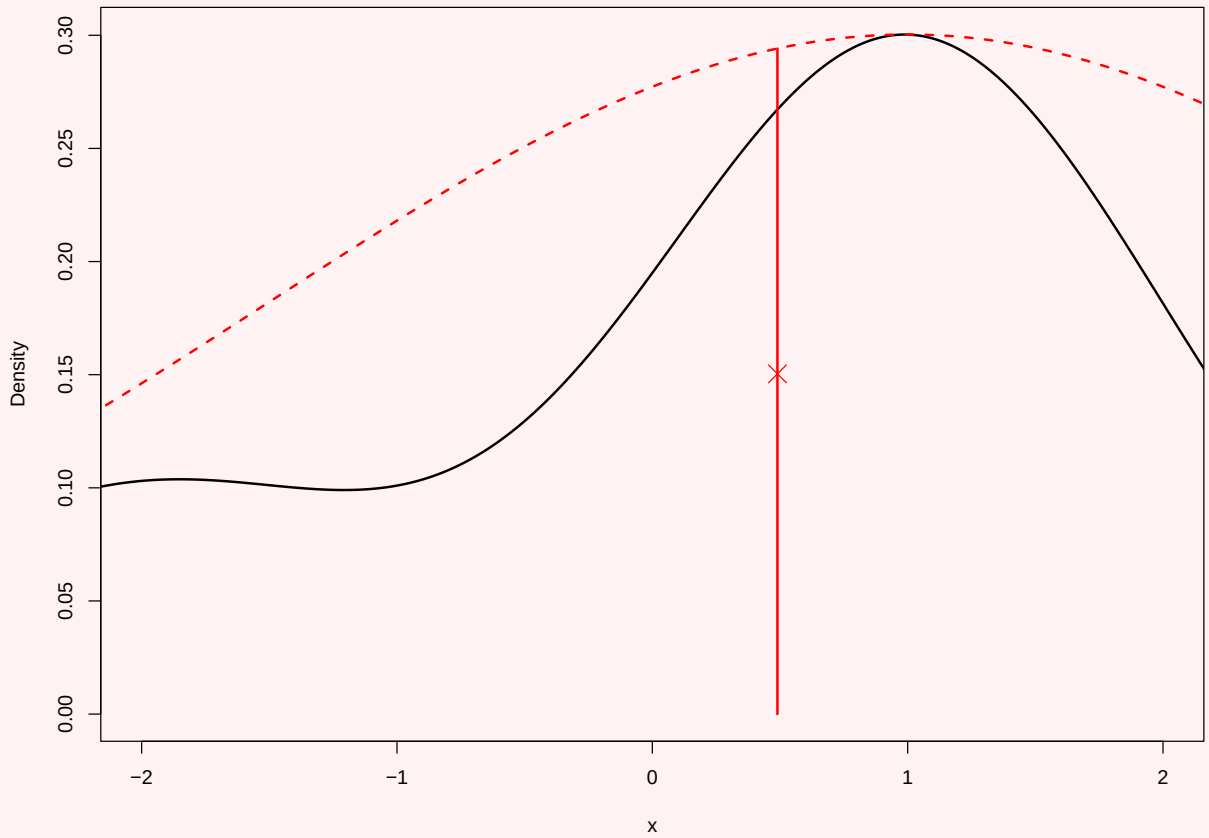
Rejection sampling

Example: Normal mixture



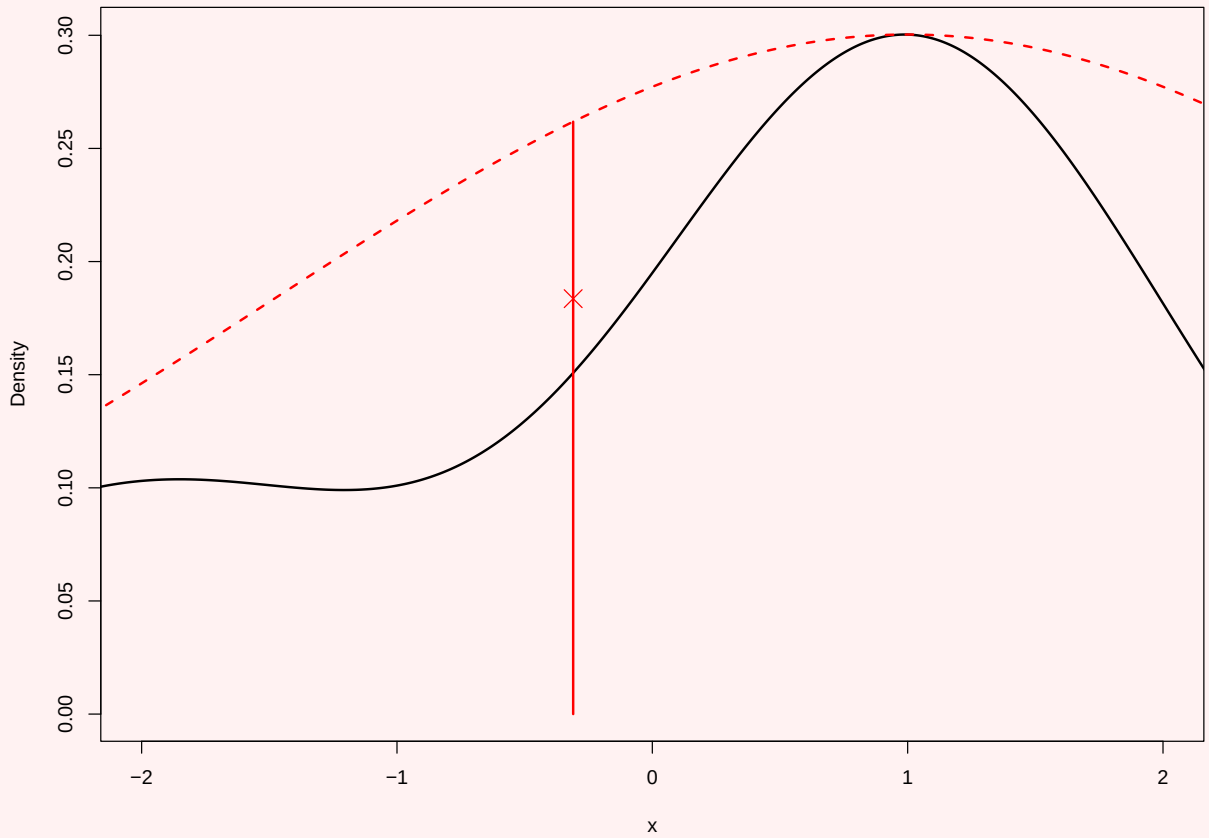
Rejection sampling

Example: Normal mixture



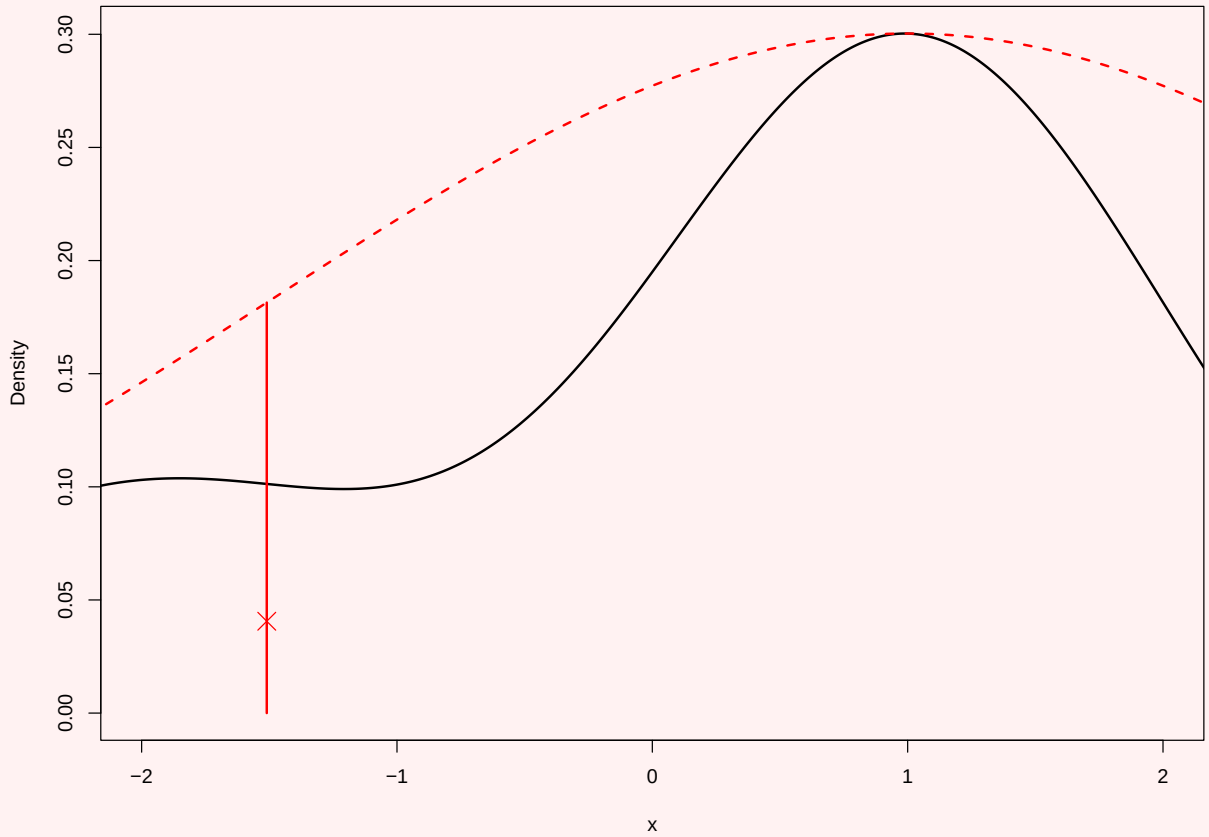
Rejection sampling

Example: Normal mixture



Rejection sampling

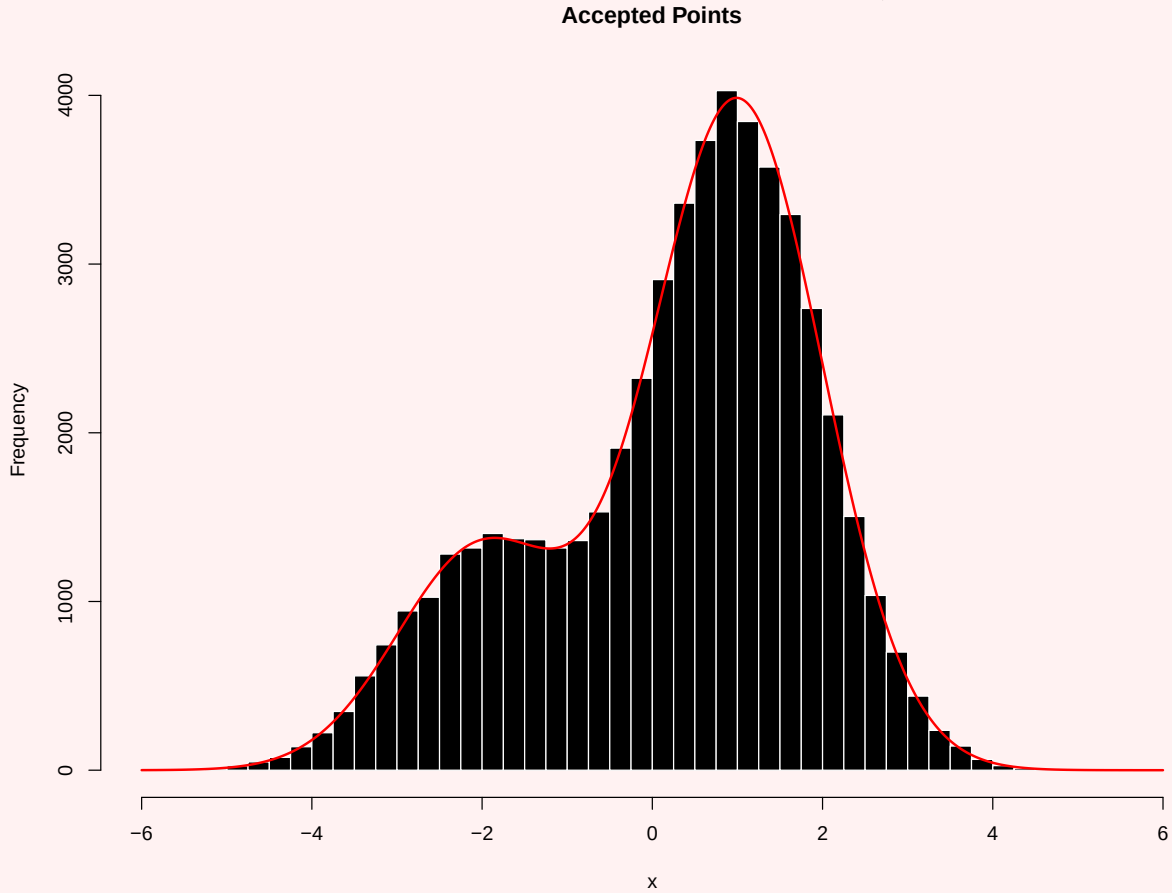
Example: Normal mixture



Rejection sampling

Example: Normal mixture

53078 out of 100000 points accepted ($1/M = 0.5313$).



Rejection sampling

- The acceptance rate is 1 if $f_0(x) \equiv f(x)$.
- If $f_0(x)$ resembles $f(x)$, then the acceptance rate will be high.
- The maximization of $f(x)/f_0(x)$ can be done numerically.
- The extension to the multivariate case is straightforward
- The proposal density can be constructed *adaptively*;
  - whenever a point is rejected, it is used to update the proposal function.

Adaptive Rejection sampling

If $f(x)$ is *log-concave* (that is $\log f(x)$ is concave), then *adaptive rejection sampling* constructs an adaptive proposal density using rejected points.

Even when a point x is rejected, $f(x)$ is computed, and this information is useful improving f_0.

For example, consider sampling from $Gamma(\alpha, 1)$ for noninteger $\alpha > 1$. We have

$$\log f(\mathbf{x}) = \text{const.} + (\alpha - 1)\log \mathbf{x} - \mathbf{x}$$

which is concave, as the second derivative is negative.

Adaptive Rejection sampling

The proposal function $f_0(x)$ can be taken as $Exponential(1/2)$; in this case

$$\frac{f(x)}{f_0(x)} = \frac{2}{\Gamma(\alpha)} x^{\alpha-1} e^{-x/2}$$

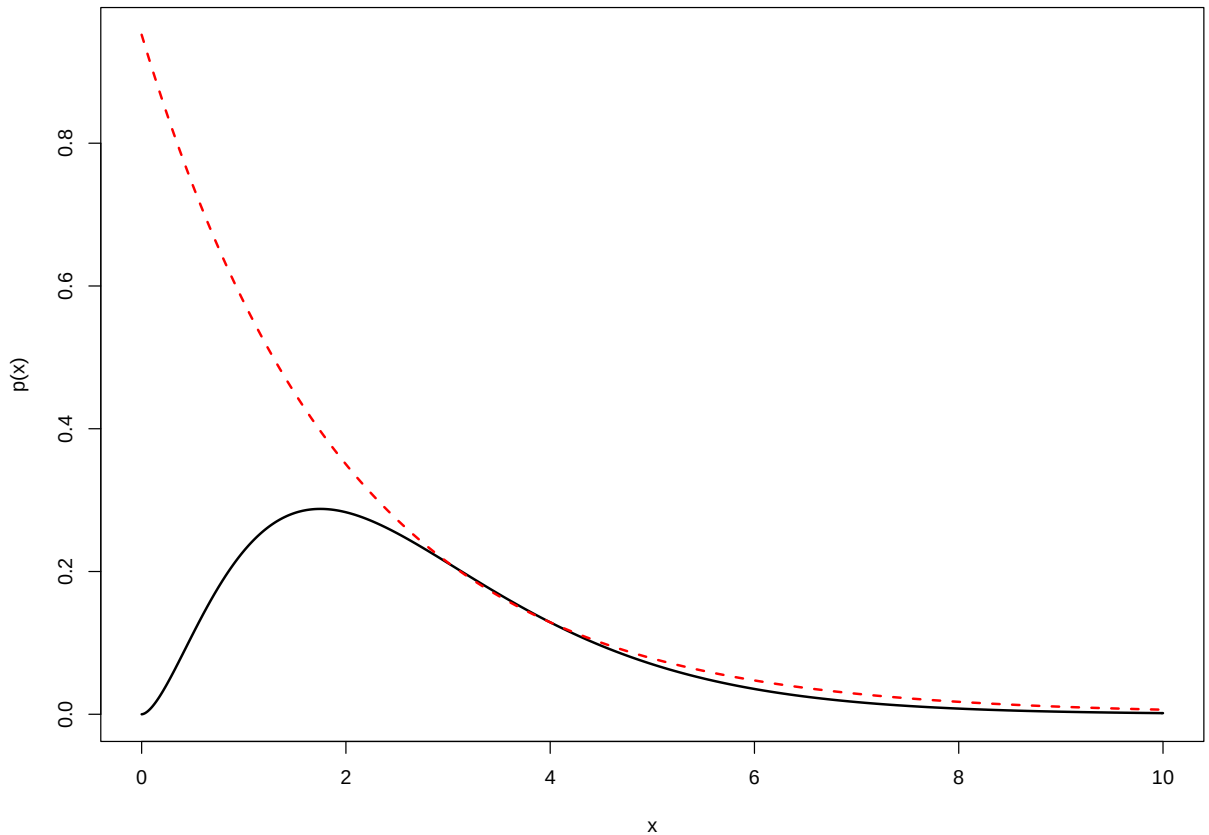
which achieves its unique maximum at

$$x_{\max} = 2(\alpha - 1)$$

Thus rejection sampling can be carried out using this f_0.

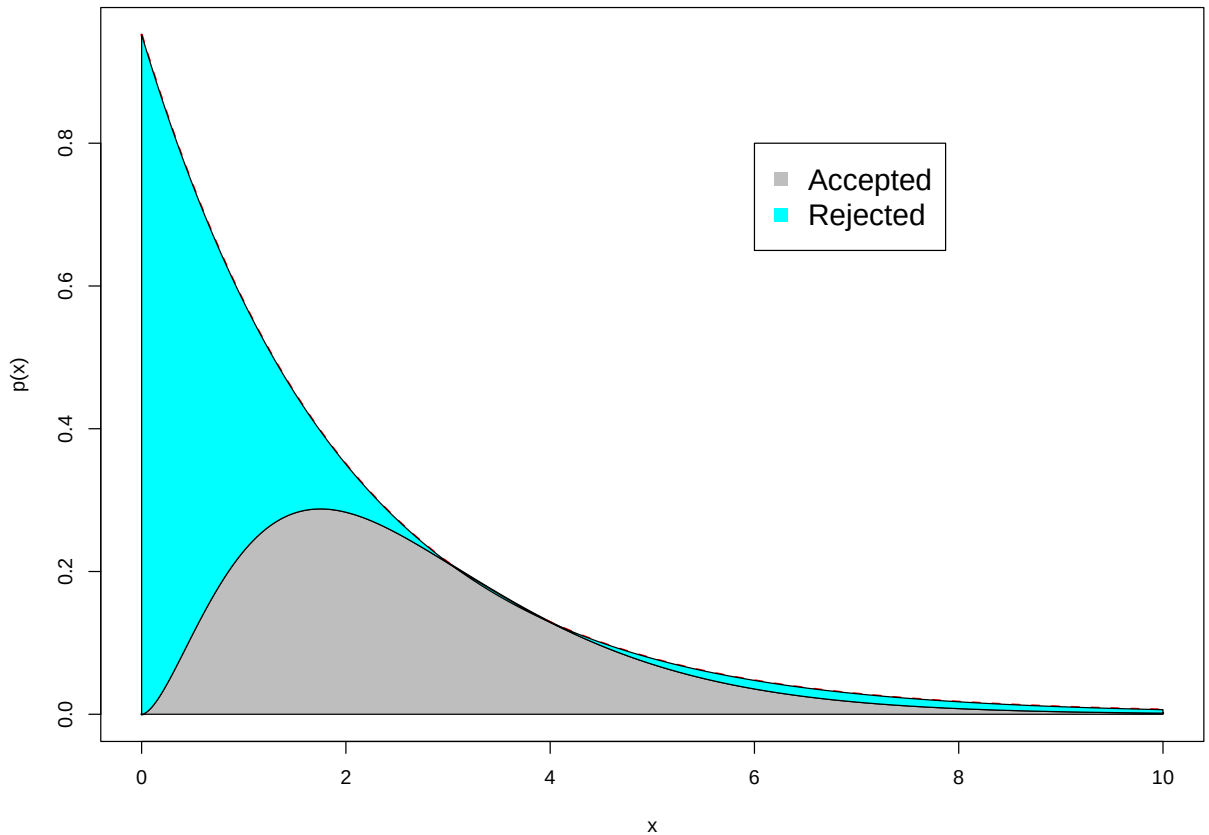
Adaptive Rejection sampling

However, the $Exponential(1/2)$ distribution is potentially inefficient



Adaptive Rejection sampling

However, the $Exponential(1/2)$ distribution is potentially inefficient



Adaptive Rejection sampling

In this problem

$$M = \frac{2}{\Gamma(\alpha)} \mathrm{x}_{\max}^{\alpha-1} e^{-\mathrm{x}_{\max}/2}$$

so acceptance rate is the reciprocal of this

α	1	2	3	4	5	6	7	8
$1/M$	0.5000	0.6796	0.4618	0.2790	0.1600	0.0890	0.0486	0.0262

For large α, the acceptance rate becomes very small.

Adaptive Rejection sampling

Adaptive rejection sampling proceeds by adapting the *squeezing* method. Suppose that there exists a function $l(x)$ such that

$$l(x) \leqslant f(x) \leqslant Mf_0(x).$$

Then the following algorithm generates a variate from $f(x)$:

1. Generate x from f_0, and u from $Uniform(0,1)$
2. Accept x if $u \leqslant l(x)/(Mf_0(x))$ and stop.
3. Accept x if $u \leqslant f(x)/(Mf_0(x))$ and stop.
4. Return to 1.

The function $l(.)$ provides a screening mechanism.

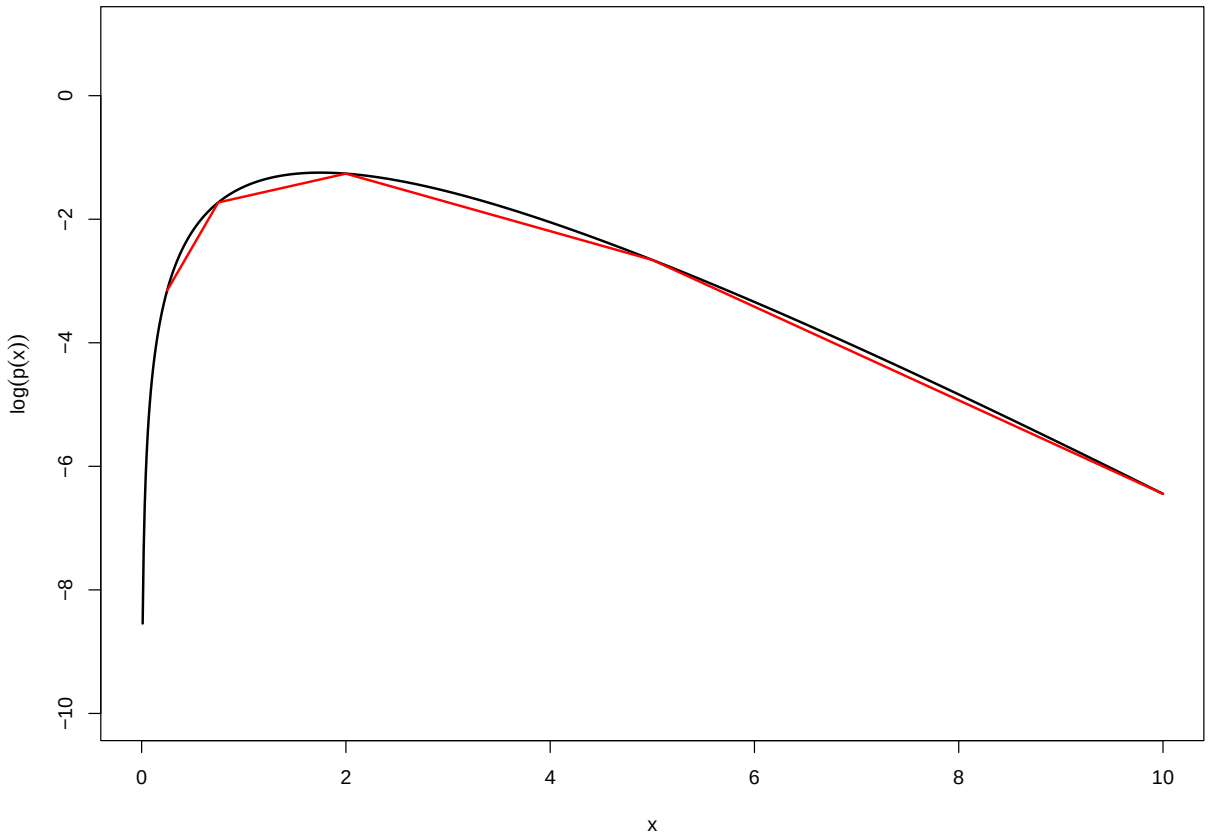
Adaptive Rejection sampling

For an adaptive version for log-concave densities, suppose that $(x_0, \ldots, x_{n+1})$ is an ordered set of points at which $y_i = \log f(x_i)$ has been evaluated for each i. For a log-concave density, the line segment

$$L_{i,i+1}(x) : (x_i, y_i) \longrightarrow (x_{i+1}, y_{i+1})$$

for $x_i \leqslant x \leqslant x_{i+1}$ is *below* $\log f(x)$ on (x_i, x_{i+1}), and above $\log f(x)$ on the rest of the range.

Adaptive Rejection sampling



Adaptive Rejection sampling

Therefore, consider the squeezing envelope defined piecewise on the support of $f(x)$. For $x_i \leqslant x \leqslant x_{i+1}$

$$\text{Lower} \quad : \quad l_n(x) = L_{i,i+1}(x)$$

$$\text{Upper} \quad : \quad u_n(x) = \min\{L_{i-1,i}(x), L_{i+1,i+2}(x)\}$$

where, for each i

$$L_{i,i+1}(x) = y_i + \left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right)(x - x_i)$$

Take

$$l_n(x) = -\infty \qquad\qquad u_n(x) = \min\{L_{0,1}(x), L_{n,n+1}(x)\}$$

for x outside of (x_0, x_{n+1}).

Adaptive Rejection sampling

The scheme needs initialization; usually two function evaluations at x values either side of the mode are sufficient.

The lower bound $l_n(x)$ and upper bound $u_n(x)$ yield the following bounds on exponentiation

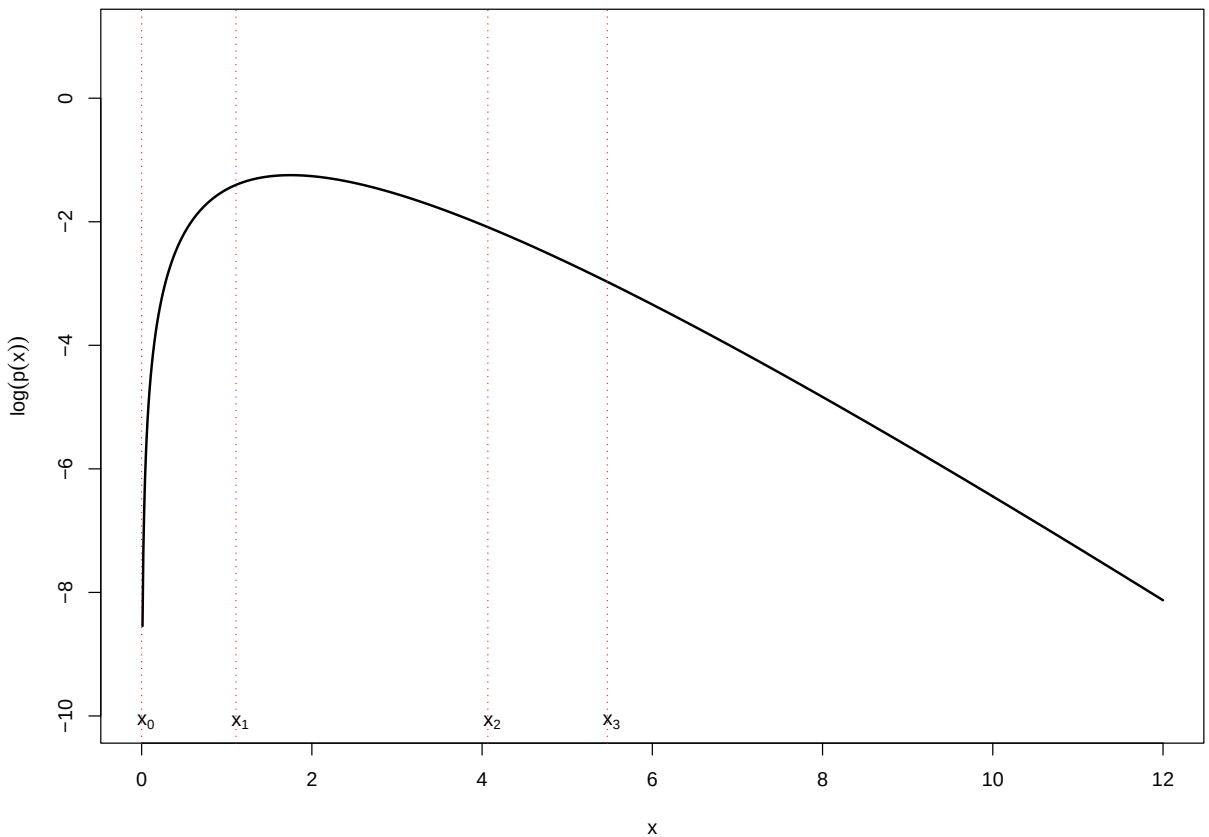
$$\exp\{l_n(x)\} \leqslant f(x) \leqslant \exp\{u_n(x)\} = M_n f_{0n}(x)$$

say, where M_n is a scaling constant such that $f_{0n}(x)$ is a density function

$$f_{0n}(x) = \frac{\sum_{i=1}^{n} \exp\{m_i x + c_i\} \mathbb{1}_{[x_{i-1}, x_i]}(x)}{\sum_{i=1}^{n} \int_{-\infty}^{\infty} \exp\{m_i t + c_i\} \mathbb{1}_{[x_{i-1}, x_i]}(t)\, dt}$$

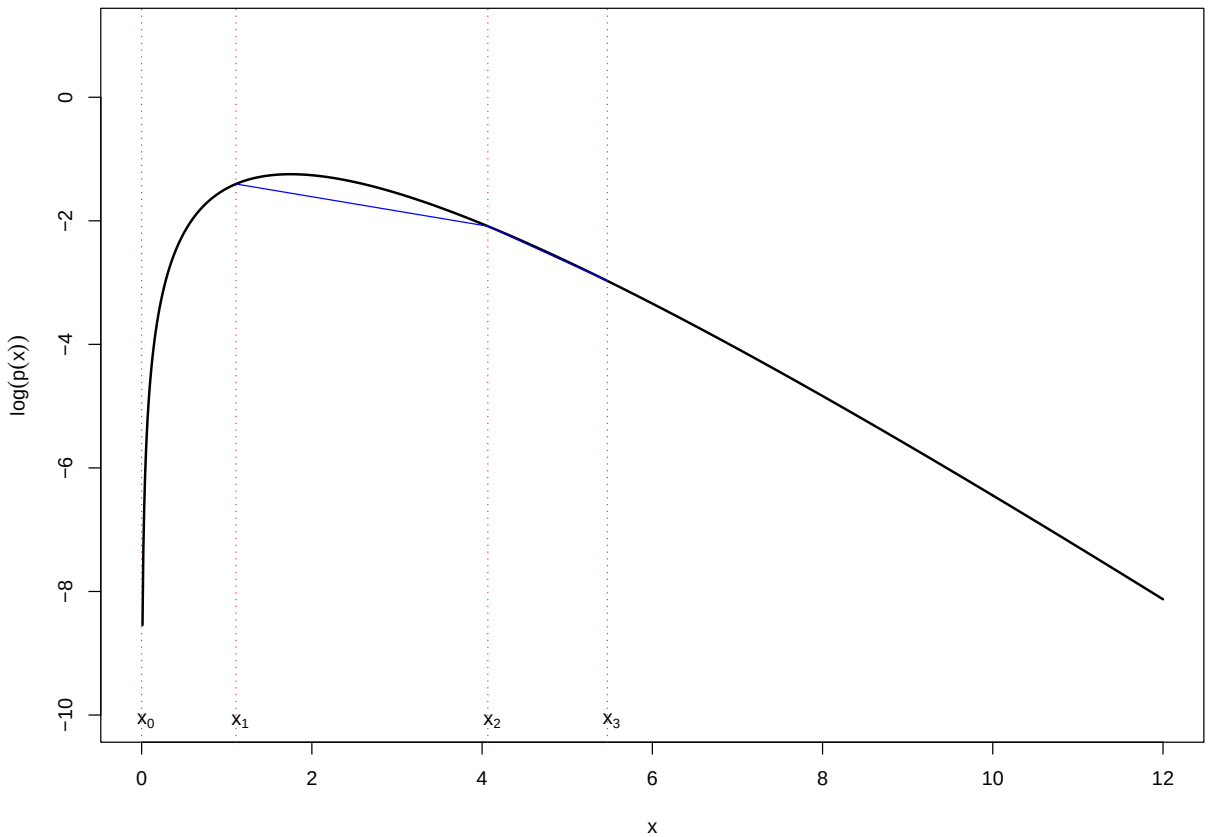
Adaptive Rejection sampling

Log-density:



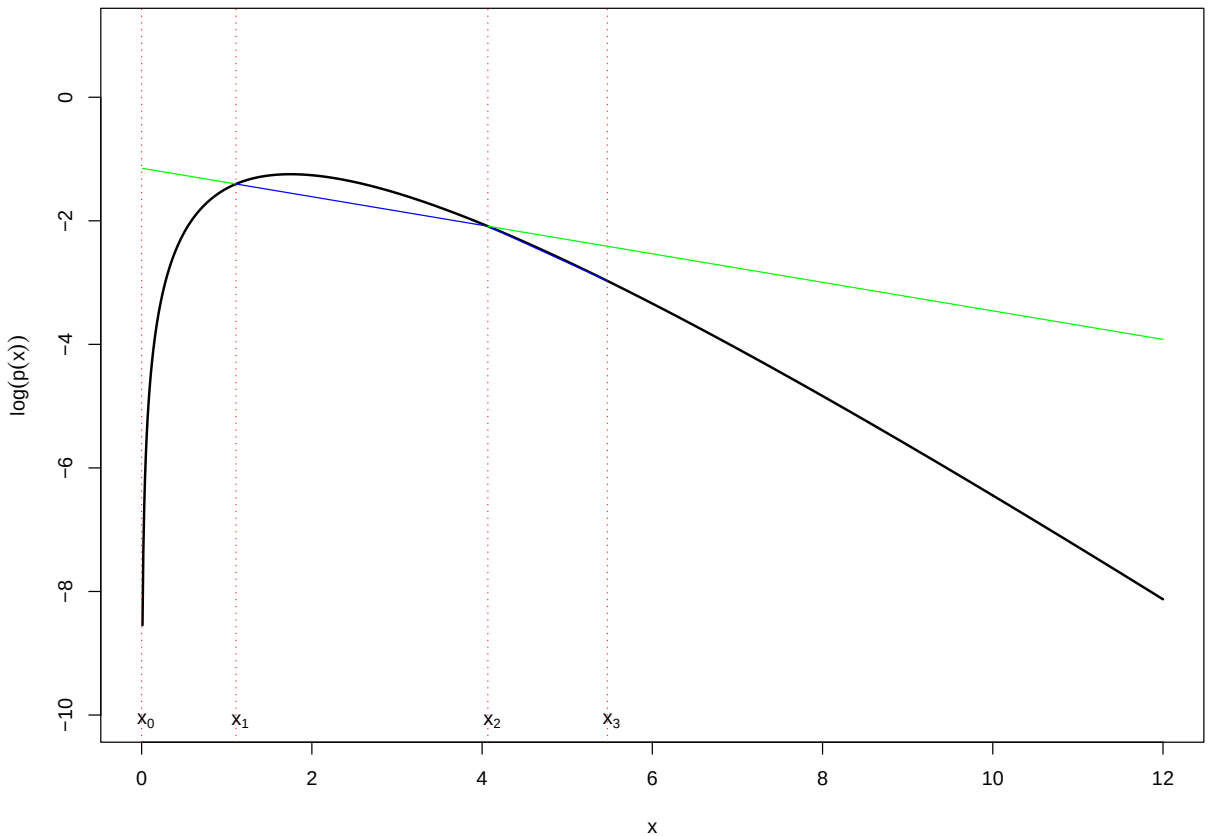
Adaptive Rejection sampling

Lower Envelope



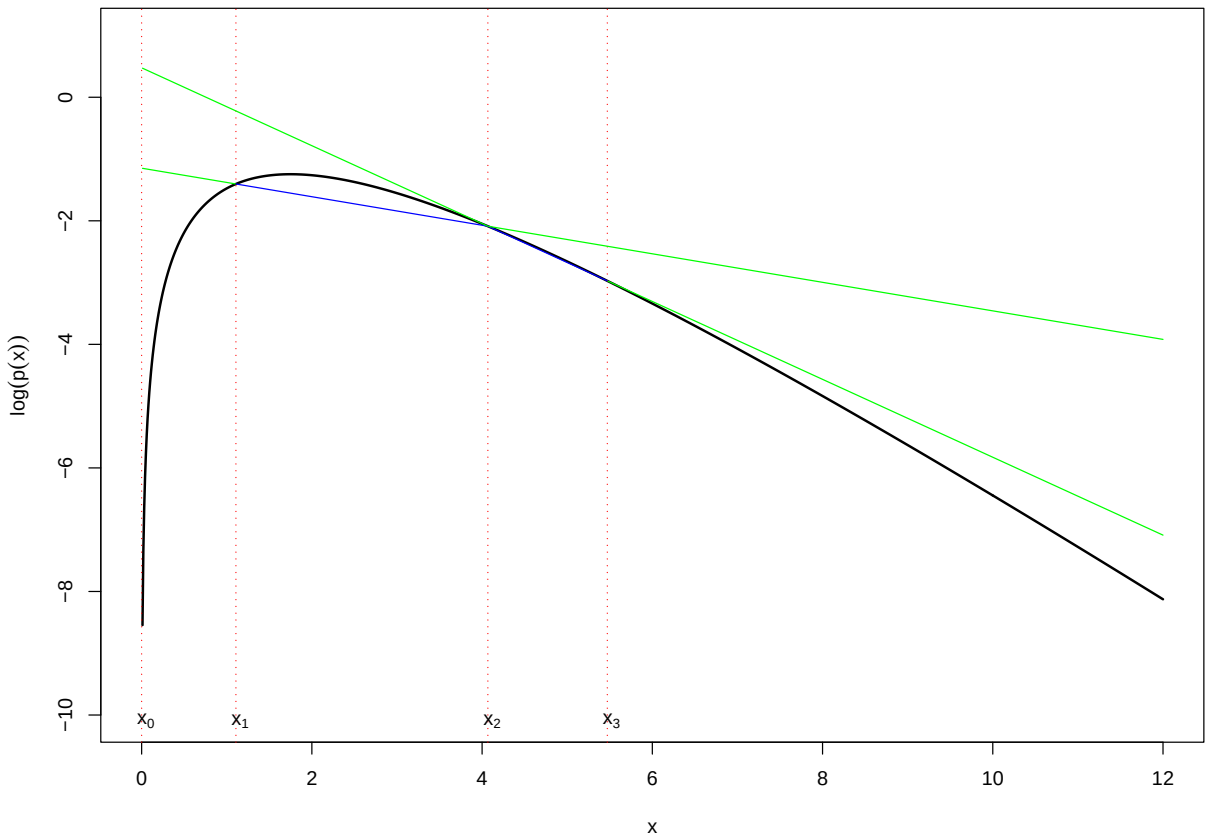
Adaptive Rejection sampling

Upper envelope (part 1):



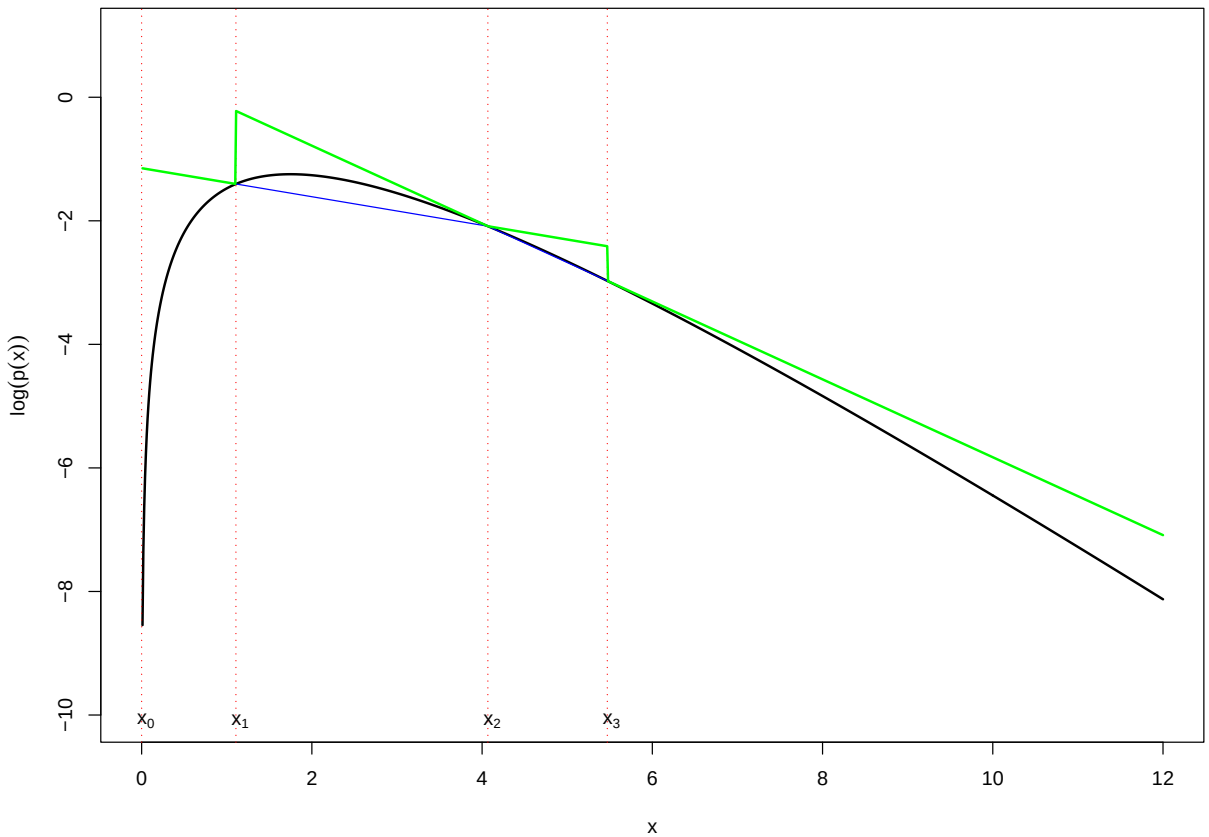
Adaptive Rejection sampling

Upper envelope (part 2):



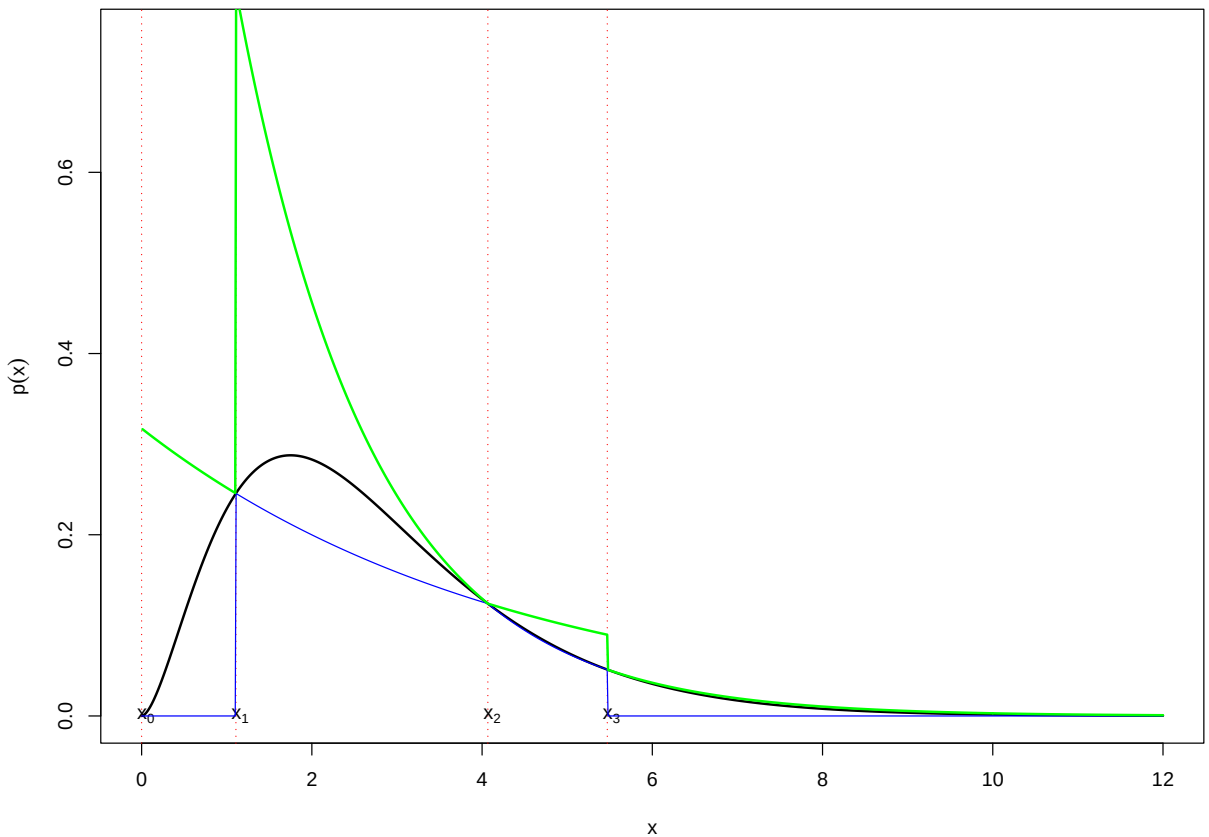
Adaptive Rejection sampling

Lower and Upper envelopes:



Adaptive Rejection sampling

Lower and Upper envelopes: original scale:



Adaptive Rejection sampling

This $f_{0n}(x)$ can be sampled directly by cdf inversion. The algorithm is therefore

1. Generate x from f_{n0}, and u from $Uniform(0,1)$
2. Accept x if $u \leqslant \exp\{l_n(x)\}/(M_n f_{0n}(x))$, and go to 4.
3. Accept x if $u \leqslant f(x)/(M_n f_{0n}(x))$ and go to 4.
4. Update f_{n0} by updating the lower and upper envelopes taking into account the x and the function value $f(x)$, and return to 1.

Every time a point is accepted or rejected, the envelopes are potentially updated; the updating only changes the envelopes locally, the intervals surrounding the location of the rejected points.

Rejection sampling and Importance sampling

There is obviously a clear connection between rejection sampling and importance sampling

- both rely on choosing a suitable $f_0(x)$
- the boundedness of the ratio $f(x)/f_0(x)$ is crucial in the construction of the procedure

The difference is that rejection sampling produces i.i.d. samples from $f(x)$, whereas importance sampling approximates numerical integration with respect to $f(x)$.

Sampling Importance Resampling

Sampling Importance Resampling (SIR) can be used to sample (approximately) from density $f(x)$ by *re-weighting* and then *resampling* samples from $f_0(x)$:

1. Generate variates $x_1, \ldots, x_N$ from f_0.
2. Compute renormalized weights $w_1, \ldots, w_N$ given by

$$w_i = \frac{f(x_i)/f_0(x_i)}{\sum_{j=1}^{N} (f(x_j)/f_0(x_j))} \qquad i = 1, \ldots, N.$$

3. Resample y_i from the discrete distribution on $\{x_1, \ldots, x_N\}$ with masses $\{w_1, \ldots, w_N\}$.

Sampling Importance Resampling

Example: Normal mixture

$$
\begin{aligned}
f(x) &= \frac{1}{4}\phi(x + 2.5) + \frac{3}{4}\phi(x - 1) \\
f_0(x) &= \frac{1}{\sigma}\phi(x/\sigma)
\end{aligned}
$$

for some $\sigma > 0$.

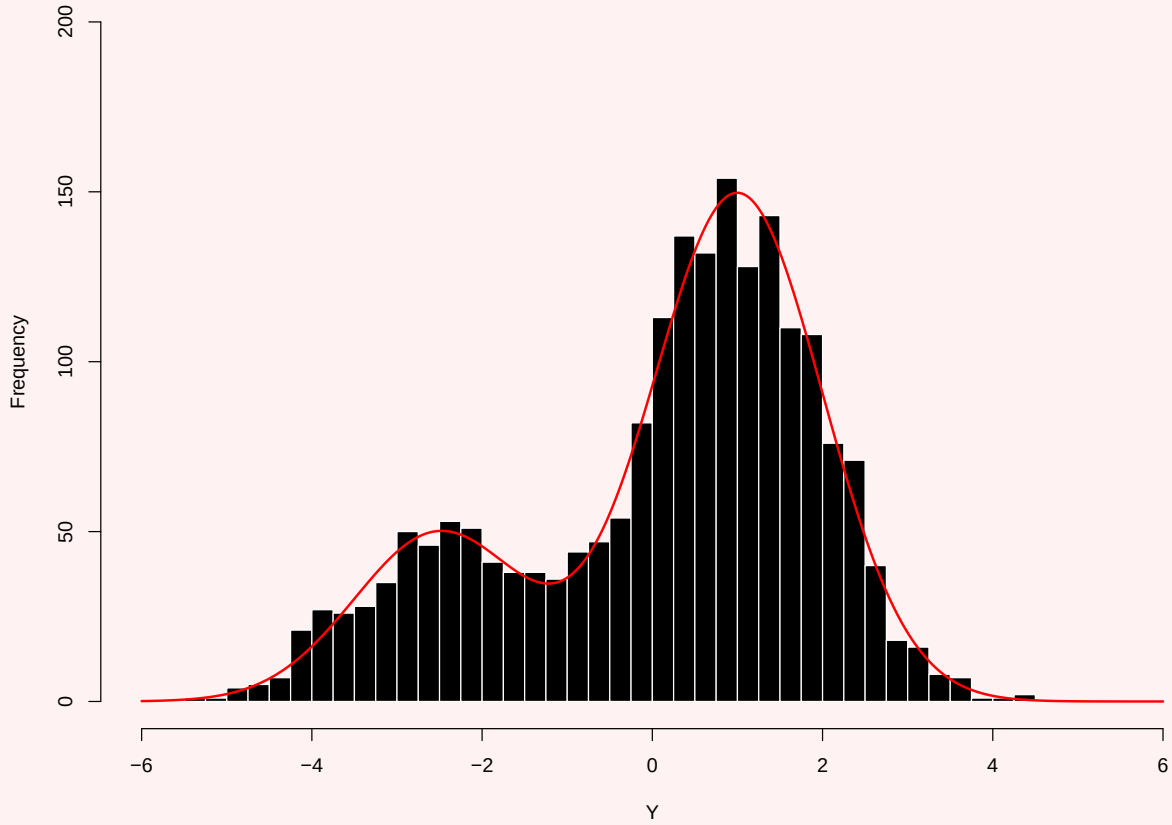
Choosing σ poorly can disrupt the performance of the SIR algorithm.

- need to ensure there are samples in the tails of $f(x)$.

Sampling Importance Resampling

Example: Normal mixture

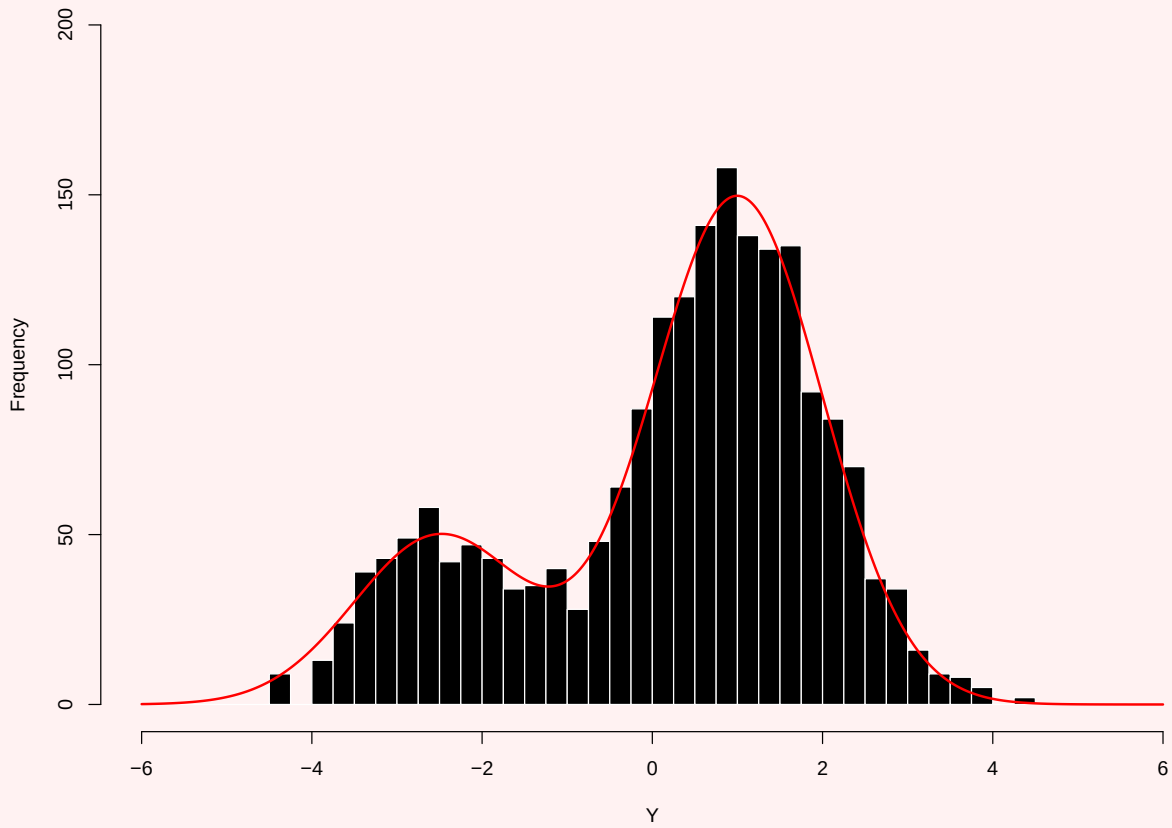
SIR with $\sigma = 2$ ($N = 100000$, 2000 resampled points):



Sampling Importance Resampling

Example: Normal mixture

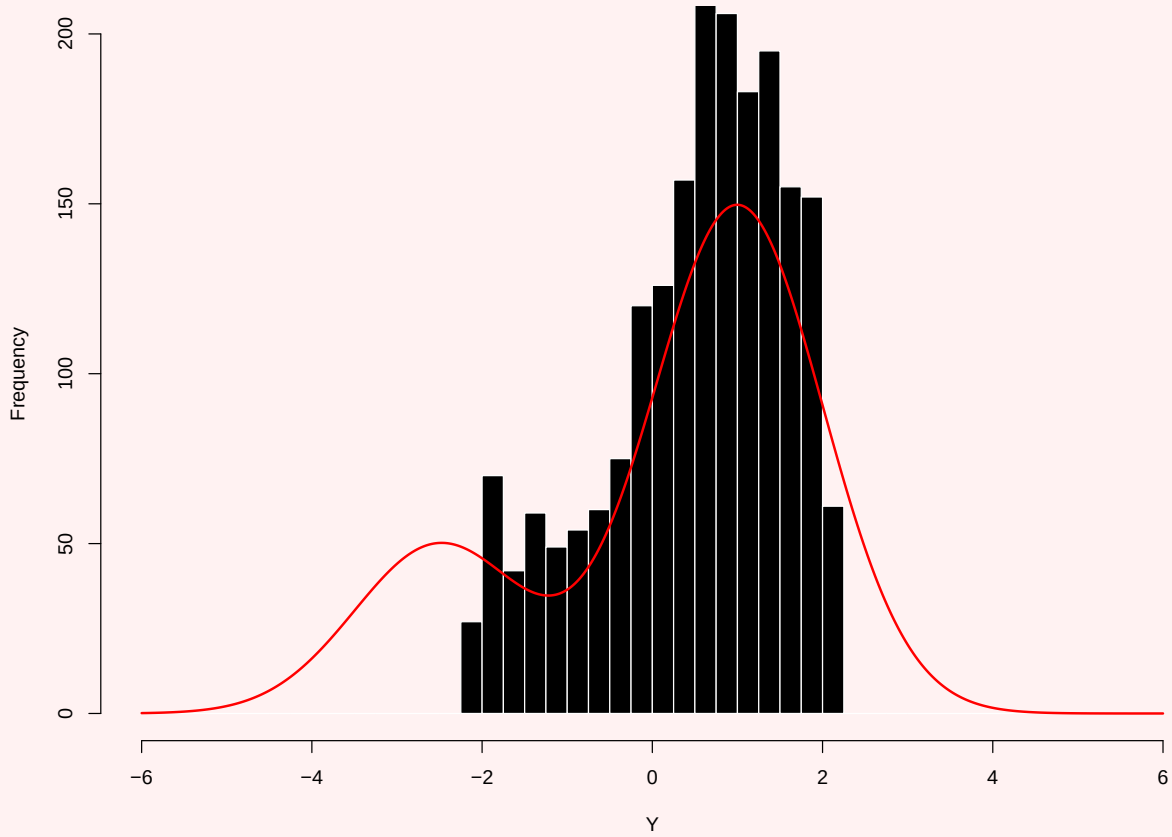
SIR with $\sigma = 1$ ($N = 100000$, 2000 resampled points):



Sampling Importance Resampling

Example: Normal mixture

SIR with $\sigma = 0.5$ ($N = 100000$, 2000 resampled points):



Variance Reduction: Antithetic Variables

Suppose that the integral

$$I = \int g(x) f(x)\, dx$$

is to be estimated via Monte Carlo. The usual estimator

$$\widehat{I}_N(g) = \frac{1}{N} \sum_{i=1}^{N} g(X_i),$$

with $X_1, \ldots, X_N$ i.i.d. from f, has variance

$$\frac{\mathrm{Var}[g(X)]}{N}$$

We aim to reduce this variance.

Variance Reduction: Antithetic Variables

Consider the estimator

$$\breve{I}_{N_A}(g) = \frac{1}{2N_A} \sum_{i=1}^{N_A} \left[g(X_i) + g(Y_i) \right],$$

where pairs $(X_1, Y_1), \ldots, (X_{N_A}, Y_{N_A})$ are i.i.d. from some bivariate distribution with identical marginal distribution f, and $2N_A = N$.

Variance Reduction: Antithetic Variables

The variance of this estimator is

$$\frac{\mathrm{Var}[g(X)]}{2N_A} + \frac{\mathrm{Cov}[g(X), g(Y)]}{2N_A}$$

Therefore if $\mathrm{Cov}[g(X), g(Y)]$ is *negative*, then this variance is smaller than the usual Monte Carlo variance. This is termed the method of *antithetic variables.*

Variance Reduction: Antithetic Variables

Example: Bivariate Normal

Suppose $(X, Y)^\top \sim \mathcal{N}_2(\mu \mathbf{1}^\top, \Sigma)$, with variances 1 and

$$\mathrm{Cov}[X, Y] = \rho$$

In the estimation of μ by Monte Carlo, the relative variances of the Monte Carlo and Antithetic Variable estimators is

$$\frac{\mathrm{Var}[X]}{\mathrm{Var}[X] + \mathrm{Cov}[X, Y]} = \frac{1}{1 - \rho}$$

Thus there is automatic variance reduction for the same total simulation size.

There is a slight computation overhead in computing $\breve{I}_{N_A}(g)$.

Variance Reduction: Control Variates

Suppose

$$I(g) = \int g(x)f(x)\,dx$$

with Monte Carlo estimator $\widehat{I}_N(g)$.

Consider an adjusted Monte Carlo estimator of $I(g)$

$$\widehat{I}_N(g; g_0, \lambda) = \widehat{I}_N(g) + \lambda(\widehat{I}_N(g_0) - I(g_0))$$

for some function g_0, where $\widehat{I}_N(g_0)$ is an unbiased estimator of $I(g_0)$, and λ is some real constant.

Variance Reduction: Control Variates

This estimator is unbiased for $I(g)$, and has variance

$$\mathrm{Var}[\widehat{I}_N(g)] + \lambda^2 \mathrm{Var}[\widehat{I}_N(g_0)] + 2\lambda \mathrm{Cov}[\widehat{I}_N(g), \widehat{I}_N(g_0)]$$

Therefore, choosing

$$\lambda = -\frac{\mathrm{Cov}[\widehat{I}_N(g), \widehat{I}_N(g_0)]}{\mathrm{Var}[\widehat{I}_N(g_0)]}$$

yields the variance of $\widehat{I}_N(g; g_0, \lambda)$ as

$$(1 - \mathrm{Corr}[\widehat{I}_N(g), \widehat{I}_N(g_0)]^2)\mathrm{Var}[\widehat{I}_N(g)] \leqslant \mathrm{Var}[\widehat{I}_N(g)]$$

Variance Reduction: Control Variates

This optimal choice of λ yields the maximum possible reduction for a given g_0, but requires knowledge of the sign of the covariance value.

In practice, the sign of λ can be assessed by regressing $g(x_i)$ on $g_0(x_i)$.

Variance Reduction: Control Variates

Example: Normal tail probability

Consider

$$I(g) = \int_{-\infty}^{\infty} g(x)f(x)\,dx = \int_{x_0}^{\infty} \phi(x)\,dx = \Pr[X > x_0]$$

Consider

$$g_0(x) = \mathbb{1}_{(0,\infty)}(x)$$

so that $I(g_0) = 1/2$. We examine the variance of the estimator

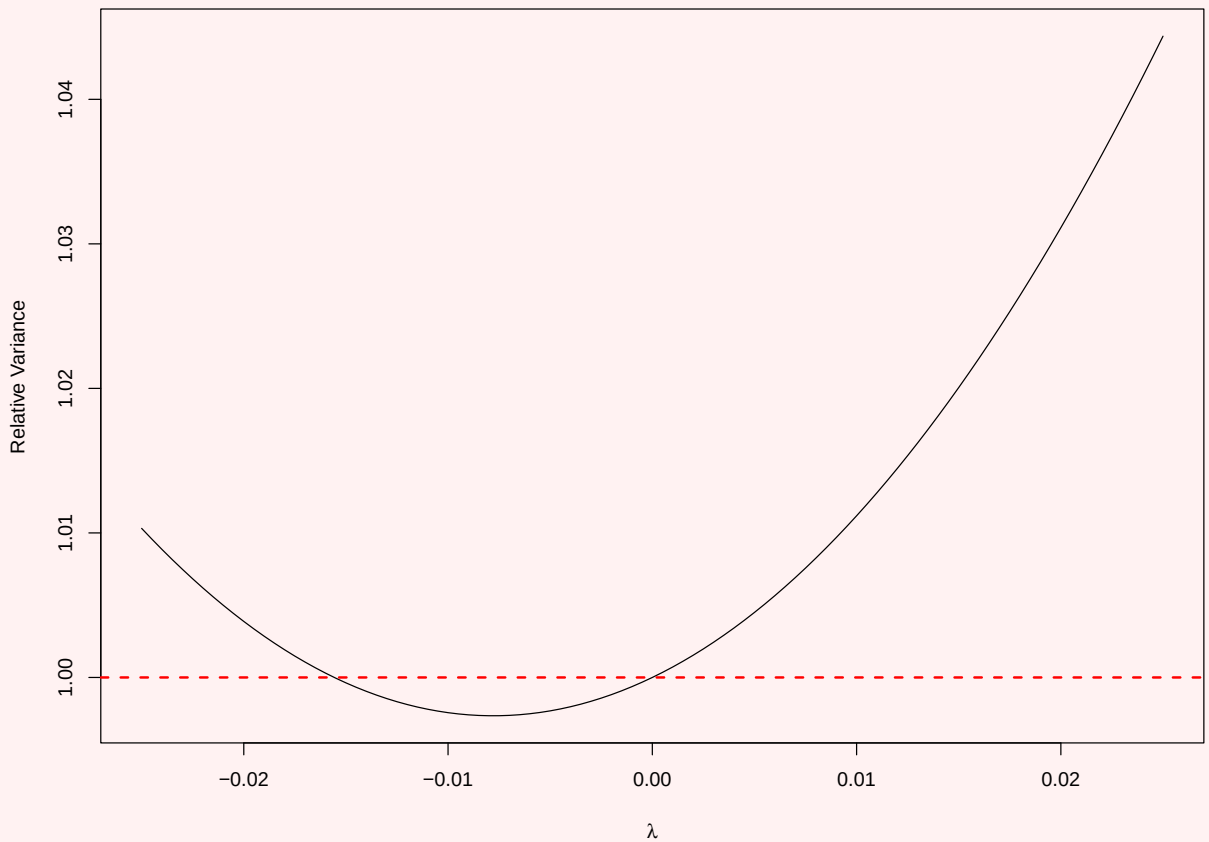
$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{(x_0,\infty)}(X_i) + \lambda\left(\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{(0,\infty)}(X_i) - \frac{1}{2}\right)$$

relative to the Monte Carlo estimator $\widehat{I}_N(g)$.

Variance Reduction: Control Variates

Example: Normal tail probability

Relative variance for varying λ: $x_0 = 2.5$, $N = 20000$.



Variance Reduction: Control Variates

Example: Normal tail probability

The best choice seems to be $\lambda \approx -0.008$.

In this case, finite sample bias might be exhibited. Therefore we also examine mean-square error (MSE)

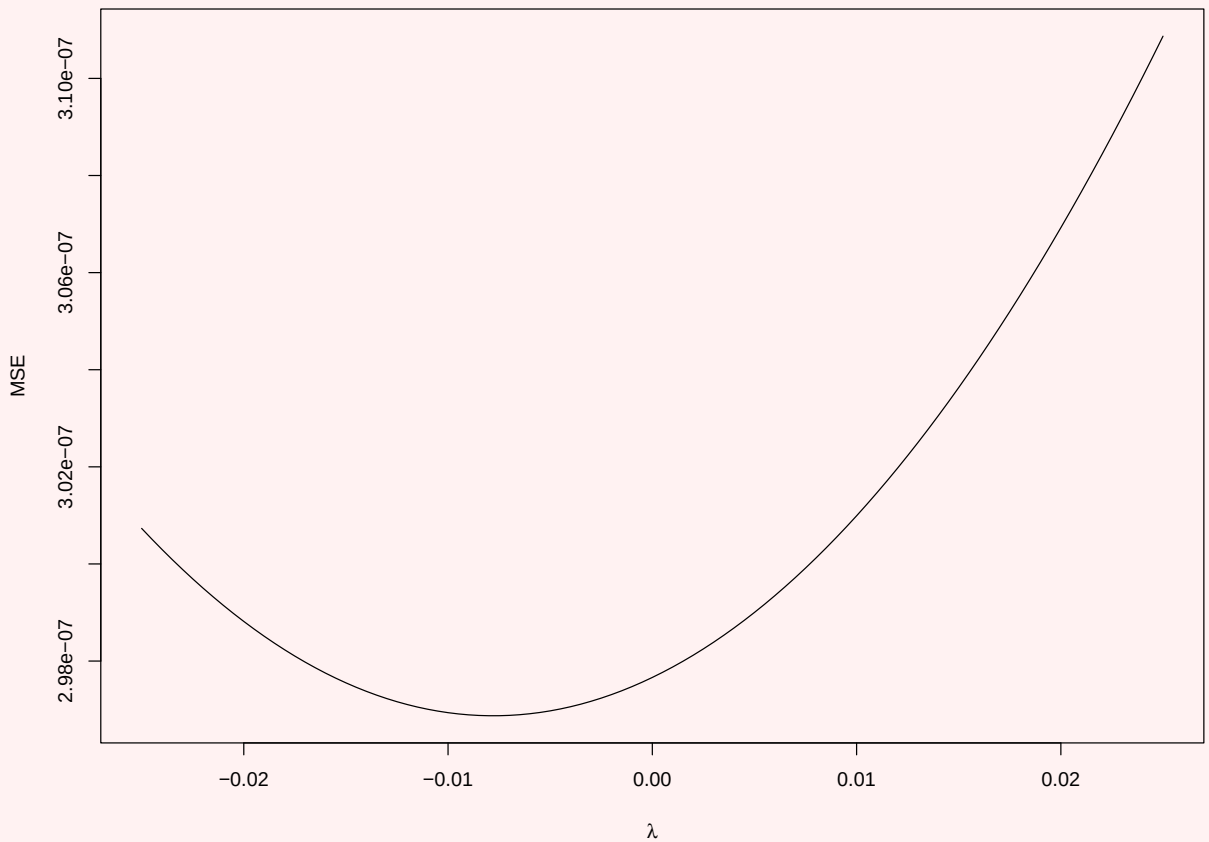
$$\mathrm{MSE}(\lambda) = \mathbb{E}[(I(g) - \widehat{I}_N(g; g_0, \lambda))^2]$$

We see that the MSE is also lowest when $\lambda \approx -0.008$, indicating that if there is finite sample bias, it is relatively insignificant.

Variance Reduction: Control Variates

Example: Normal tail probability

MSE for varying λ: $x_0 = 2.5$, $N = 20000$.



Variance Reduction: Control Variates

Example:

See `knitr 5`

Markov chain Monte Carlo

Generating variates from a probability distribution $f(x)$ is not straightforward when

- x is high-dimensional (but not Gaussian),
- $f(x)$ is of non-standard form.

These difficulties occur often (but not exclusively) in Bayesian inference.

We seek a way of simulating from $f(x)$ in this situation.

A brief introduction to Markov chains

A *Markov chain* is a countable sequence of random variables, $\{X_t\}$, for which

$$\Pr[X_t \in B | X_1 = x_1, \ldots, X_{t-1} = x_{t-1}] = \Pr[X_t \in B | X_{t-1} = x_{t-1}]$$

that is, X_t is conditionally independent of $X_0, \ldots, X_{t-2}$ given X_{t-1}.

In the simplest case, the $\{X_t\}$ takes values on the finite (discrete) state space

$$\mathcal{S}_X = \{s_1, \ldots, s_d\}$$

and the chain is *homogeneous*, that is, the stochastic properties of $\{X_t\}$ do not change with time.

A brief introduction to Markov chains

A homogeneous, discrete state-space Markov chain is characterized by its initial state X_0 or its initial distribution $p^{(0)}$, and its *transition matrix* P, a $d \times d$ *stochastic* matrix whose rows sum to one, such that

$$P_{ij} = \Pr[X_t = s_j | X_{t-1} = s_i]$$

describes the entire set of one-step ahead conditional probabilities. Denote by

$$p_{ij}(k) = \Pr[X_{t_0+k} = s_j | X_{t_0} = s_i]$$

the k-step ahead probabilities.

Note that if $P(k)$ is the matrix of k-step ahead transition probabilities, then

$$P(k) = P^k$$

and then the k-step distribution is

$$p^{(k)} = p^{(0)} P^k$$

A brief introduction to Markov chains

The properties of the chain depend on P. The chain is

- *irreducible* if $p_{ij}(k) > 0$, for all i, j, and at least one k.
- *aperiodic* if all states have period 1: that is, for each i, returns to state i can occur after any number of steps.

  The *period* of state i is defined as the greatest common divisor of the set of possible return times, $\mathcal{R}$,

$$\mathcal{R} = \{r : \Pr[X_r = s_i | X_0 = s_i] > 0\}$$

A brief introduction to Markov chains

- *recurrent* if all states are recurrent, that is, the probability of returning to each state in a finite number of steps is positive. Let $T_i = \inf\{k : X_k = i | X_0 = i\}$. State i is recurrent if and only if

$$\Pr[T_i = \infty] = 0$$

and *transient* otherwise. If $\mathbb{E}[T_i] < \infty$, state i is termed *positive recurrent*, otherwise it is termed *null recurrent*.

A brief introduction to Markov chains

A *stationary* or *invariant distribution*, π^*, of a homogeneous Markov chain is the $1 \times d$ vector of probabilities such that

$$\pi^* = \pi^* P$$

that is, for each i,

$$\pi_i^* = \sum_j \pi_j^* P_{ji}$$

The *equilibrium distribution* of the chain, π, is defined by

$$\pi = \lim_{k \longrightarrow \infty} p^{(k)} = p^{(0)} \lim_{k \longrightarrow \infty} P^k$$

when this limit exists and is independent of $p^{(0)}$. That is, we may compute π as

$$\mathbf{1}\pi = \lim_{k \longrightarrow \infty} P^k$$

if the limit exists. The equilibrium distribution is a stationary distribution.

An irreducible chain has a stationary distribution if and only if all of its states are positive recurrent, in which case π is unique. Given P, π can be computed as

$$\lim_{k \longrightarrow \infty} P^k = \mathbf{1}\pi.$$

where $\mathbf{1}$ is the $d \times 1$ vector of 1s. A matrix P with the stationary distribution π can be computed by solving the system of equations $\pi = \pi P$.

Key aspects of the stationary distribution are that

(a) As the k-step ahead probability matrix P^k converges to a matrix with d identical rows, the Markov chain can eventually "forget" its initial value X_0.

(b) Realized values of $\{X_t\}$ have statistical properties that demonstrate (strong) convergence to the stationary distribution, that is, for $i = 1, \ldots, d$,

$$\lim_{n \longrightarrow \infty} \frac{\sum_{k=1}^{n} \mathbb{1}_{\{s_i\}}(X_k)}{n} = \pi_i$$

A Markov chain is *reversible* if, for every $n \geqslant 1$,

$$X_0, X_1, \ldots, X_{n-1}, X_n$$

and

$$X_n, X_{n-1}, \ldots, X_1, X_0$$

have the same joint distribution.

It follows that the reverse chain is also Markov and that the individual X_k have the same marginal distribution: for arbitrary state sequence $(l_{k+2}, \ldots, l_n)$,

$$\begin{aligned}
&\Pr\left[X_k = i \middle| X_{k+1} = j, X_{k+2} = l_{k+2}, \ldots, X_n = l_n\right] \\
&\qquad = \frac{\Pr\left[X_k = i, X_{k+1} = j, X_{k+2} = l_{k+2}, \ldots, X_n = l_n\right]}{\Pr\left[X_{k+1} = j, X_{k+2} = l_{k+2}, \ldots, X_n = l_n\right]} \\
&\qquad = \frac{\pi_i P_{ij} P_{j,l_{k+2}} \cdots P_{l_{n-1},l_n}}{\pi_j P_{j,l_{k+2}} \cdots P_{l_{n-1},l_n}} \\
&\qquad = \frac{\pi_i P_{ij}}{\pi_j}
\end{aligned}$$

which only depends on i and j.

A homogeneous Markov chain with stationary distribution π is reversible if

$$\pi_i P_{ij} = \pi_j P_{ji}$$

for all states i and j. This is also termed the *detailed balance* condition.

Note that *if* this equation holds for a specified π, *then* this implies that the P has been specified so as to have stationary distribution π.

Discrete Markov chains

Note that the $\{X_t\}$ are *dependent* random variables, so the standard frequentist asymptotic laws do not directly apply.

However, the *ergodic theorem* applies for irreducible, aperiodic and positive recurrent Markov chains, in particular

$$\frac{1}{N}\sum_{t=1}^{N} g(X_t) \xrightarrow{a.s.} \mathbb{E}_\pi[g(X)]$$

for all bounded functions g, provided

$$\mathbb{E}_\pi[|g(X)|] < \infty$$

A Central Limit Theorem result also holds under mild regularity conditions, specifically,

$$\sqrt{N}\left(\frac{1}{N}\sum_{t=1}^{N} g(X_t) - \mathbb{E}_\pi[g(X)]\right) \xrightarrow{d} \mathcal{N}(0, \sigma^2(g))$$

where

$$\sigma^2(g) = \mathrm{Var}_\pi[g(X_0)] + 2\sum_{t=1}^{\infty} \mathrm{Cov}_\pi[g(X_0), g(X_t)].$$

Discrete Markov chain: example

Example: $d = 2$

Consider $d = 2$, with

$$P = \begin{bmatrix} 0.3 & 0.7 \\ 0.9 & 0.1 \end{bmatrix}$$

Then $\pi = (9/16, 7/16)$. Here

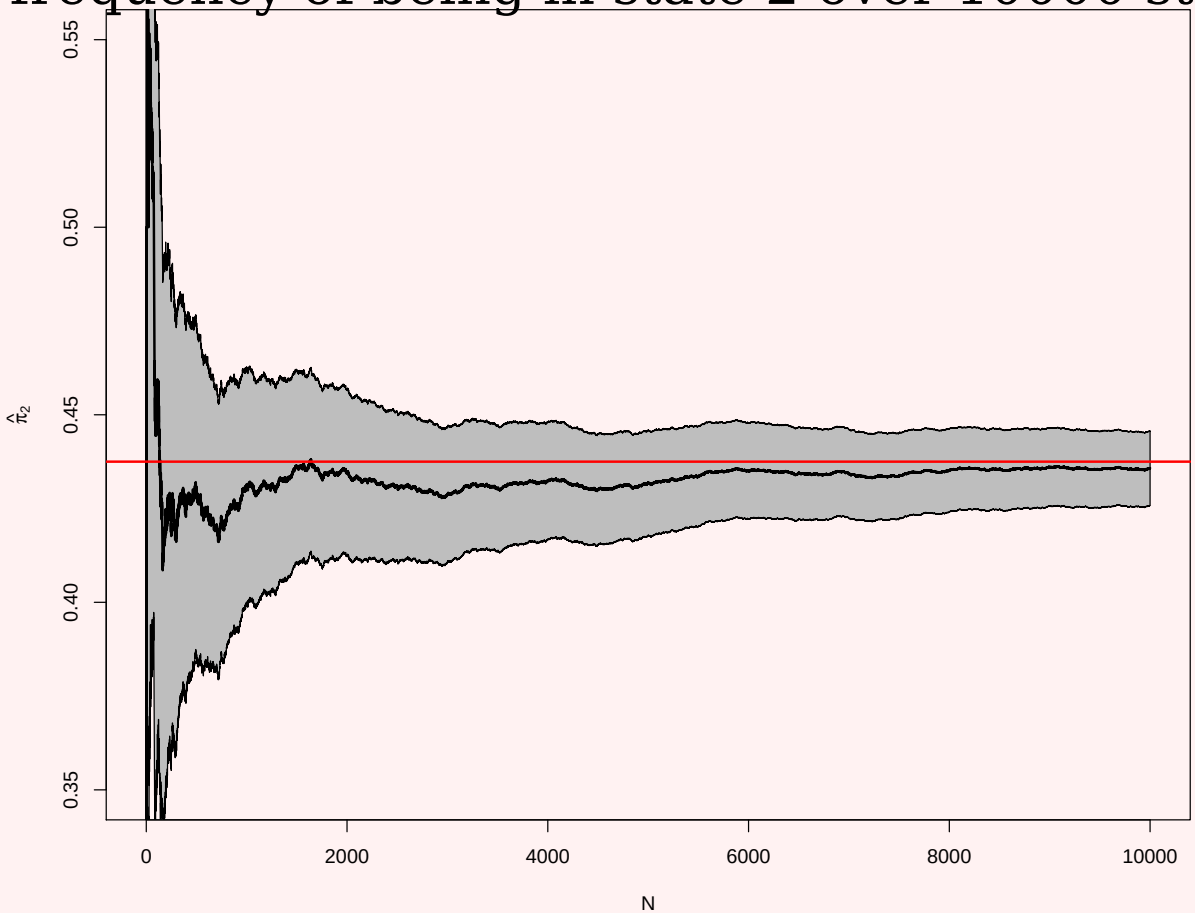
$$\pi_1 P_{12} = \frac{9}{16} \times \frac{7}{10} = \frac{63}{160} \qquad \pi_2 P_{21} = \frac{7}{16} \times \frac{9}{10} = \frac{63}{160}$$

so this chain is reversible.

Discrete Markov chain: example

Example: $d = 2$

Relative frequency of being in state 2 over 10000 steps.



Constructing reversible chains

In the 2×2, for a reversible chain, we require

$$\pi_1 P_{12} = (1 - \pi_1) P_{21}$$

or

$$\frac{\pi_1}{(1 - \pi_1)} = \frac{P_{21}}{P_{12}}$$

Constructing reversible chains

Suppose

$$P_{12} = \min\left\{1, \frac{1-\pi_1}{\pi_1}\right\} \qquad\qquad P_{21} = \min\left\{1, \frac{\pi_1}{1-\pi_1}\right\}$$

Then

$$\begin{aligned}
\pi_1 P_{12} &= \pi_1 \min\left\{1, \frac{1-\pi_1}{\pi_1}\right\} \\
&= \min\{\pi_1, 1-\pi_1\} \\
&= \min\{1-\pi_1, \pi_1\} \\
&= (1-\pi_1)\min\left\{1, \frac{\pi_1}{1-\pi_1}\right\} \\
&= (1-\pi_1)P_{21}
\end{aligned}$$

Constructing reversible chains

The above Markov chain we can think of as acting as follows:

- *If* $X_t = 1$: propose setting $X_{t+1} = 2$, but only accept this move with probability

$$\min\left\{1, \frac{1-\pi_1}{\pi_1}\right\}$$

  otherwise set $X_{t+1} = 1$.
- *If* $X_t = 2$: propose setting $X_{t+1} = 1$, but only accept this move with probability

$$\min\left\{1, \frac{\pi_1}{1-\pi_1}\right\}$$

  otherwise set $X_{t+1} = 2$.

Constructing reversible chains

A generalization of this approach is as follows:

- If $X_t = 1$: simulate Z_t on the set $\{1, 2\}$ with probabilities (q_{11}, q_{12}).
- If $X_t = 2$: simulate Z_t on the set $\{1, 2\}$ with probabilities (q_{21}, q_{22}).

Constructing reversible chains

- If $X_t = 1$: if $Z_t = 2$, set $X_{t+1} = Z_t = 2$ with probability

$$\alpha_{12} = \min\left\{1, \frac{\pi_2}{\pi_1}\frac{q_{21}}{q_{12}}\right\}$$

  otherwise set $X_{t+1} = X_t = 1$.

- If $X_t = 2$: if $Z_t = 1$, set $X_{t+1} = Z_t = 1$ with probability

$$\alpha_{21} = \min\left\{1, \frac{\pi_1}{\pi_2}\frac{q_{12}}{q_{21}}\right\}$$

  otherwise set $X_{t+1} = X_t = 2$.

Constructing reversible chains

The transition probabilities are then

$$\begin{aligned} P_{12} &= \Pr\left[X_{t+1} = 2 \middle| X_t = 1\right] = q_{12}\alpha_{12} \\ P_{21} &= \Pr\left[X_{t+1} = 1 \middle| X_t = 2\right] = q_{21}\alpha_{21} \end{aligned}$$

so therefore

$$\pi_1 P_{12} = \pi_1 q_{12}\alpha_{12} = \pi_1 q_{12} \min\left\{1, \frac{\pi_2}{\pi_1}\frac{q_{21}}{q_{12}}\right\} = \min\left\{\pi_1 q_{12}, \pi_2 q_{21}\right\}$$

$$\pi_2 P_{21} = \pi_2 q_{21}\alpha_{21} = \pi_2 q_{21} \min\left\{1, \frac{\pi_1}{\pi_2}\frac{q_{12}}{q_{21}}\right\} = \min\left\{\pi_2 q_{21}, \pi_1 q_{12}\right\}$$

and

$$\pi_1 P_{12} = \pi_2 P_{21}.$$

Constructing reversible chains

This allows the generalization to the more general discrete state space $\{1, 2, \ldots, d, \ldots\}$. Suppose that π is an arbitrary discrete distribution. Let matrix Q define the *proposal probabilities*

$$[Q]_{ij} = \Pr[Z_t = j | X_t = i]$$

Define the *acceptance probabilities*

$$\alpha_{ij} = \min\left\{1, \frac{\pi_j}{\pi_i}\frac{q_{ji}}{q_{ij}}\right\}$$

If $X_t = i$, set $X_{t+1} = Z_t = j$ with probability α_{ij}. Otherwise set $X_{t+1} = X_t = i$.

Constructing reversible chains

This Markov chain satisfies detailed balance

$$\pi_i P_{ij} = \pi_j P_{ji} \qquad \text{for all } i, j$$

provided it is irreducible, aperiodic and positive recurrent.

Note that the rows of Q must sum to 1 as

$$\sum_{j=1}^{\infty} \Pr[Z_t = j | X_t = i] = 1$$

so Q defines a stochastic proposal (or *transition*) matrix.

Constructing reversible chains: example

Example: Poisson distribution

Suppose, for $\lambda > 0$,

$$\pi_i = \frac{e^{-\lambda}\lambda^i}{i!} \qquad i = 0, 1, 2, \ldots.$$

Suppose

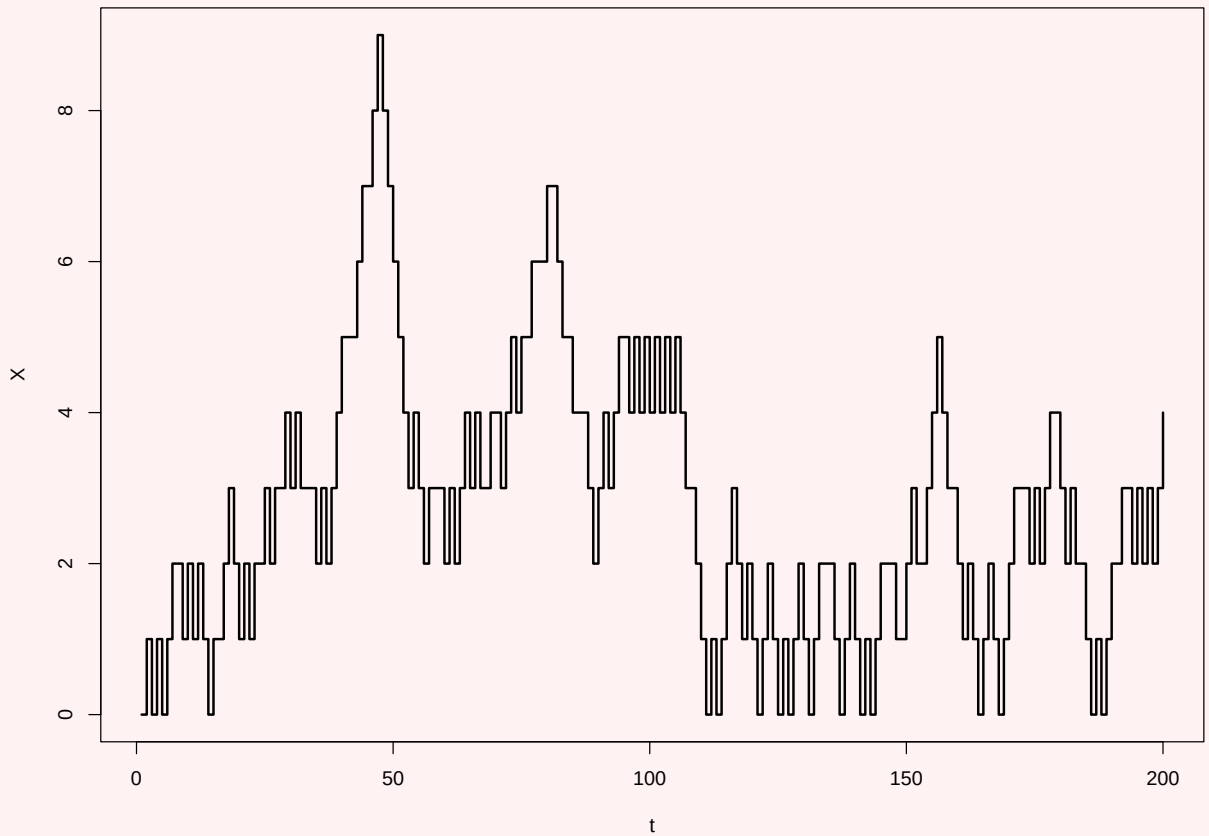
$$q_{ij} = \begin{cases} 1 & i = 0, j = 1 \\ \frac{1}{2} & i \geqslant 1, j = i-1, i+1 \\ 0 & \text{otherwise} \end{cases}$$

That is, Z_t is proposed uniformly on the finite set $\{x_t - 1, x_t + 1\}$, unless $X_t = 0$, in which case $Z_t = 1$ is proposed.

Constructing reversible chains: example

Example: Poisson distribution

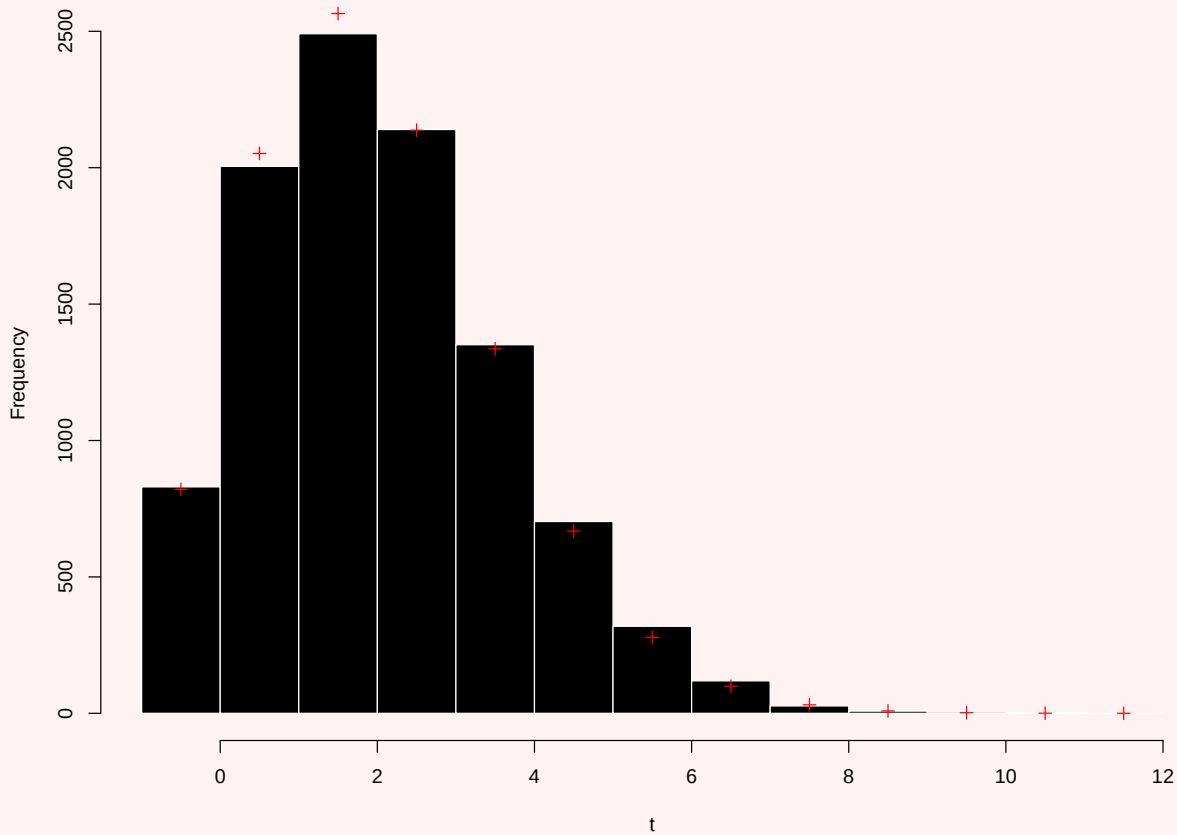
First 200 steps of the chain starting at $X_0 = 0$ with $\lambda = 2.5$.



Constructing reversible chains: example

Example: Poisson distribution

Histogram of states visited over $N = 10000$ steps (with true values (+))



Constructing reversible chains: example

Example: Poisson distribution

Relative frequencies ($\widehat{\pi}_i$) of states visited over $N = 10000$ steps, with true Poisson probabilities (π_i).

i	0	1	2	3	4	5
$\widehat{\pi}_i$	0.0830	0.2005	0.2491	0.2140	0.1351	0.0703
π_i	0.0821	0.2052	0.2565	0.2138	0.1336	0.0668

i	6	7	8	9	10	11
$\widehat{\pi}_i$	0.0319	0.0119	0.0027	0.0008	0.0005	0.0002
π_i	0.0278	0.0099	0.0031	0.0009	0.0002	0.0000

Constructing reversible chains: example

Example: Poisson distribution

To measure convergence, we can use the *Total Variation* distance. For two mass functions $f_1(x)$ and $f_2(x)$, we compute

$$d_{TV}(f_1, f_2) = \frac{1}{2} \sum_x |f_1(x) - f_2(x)|$$

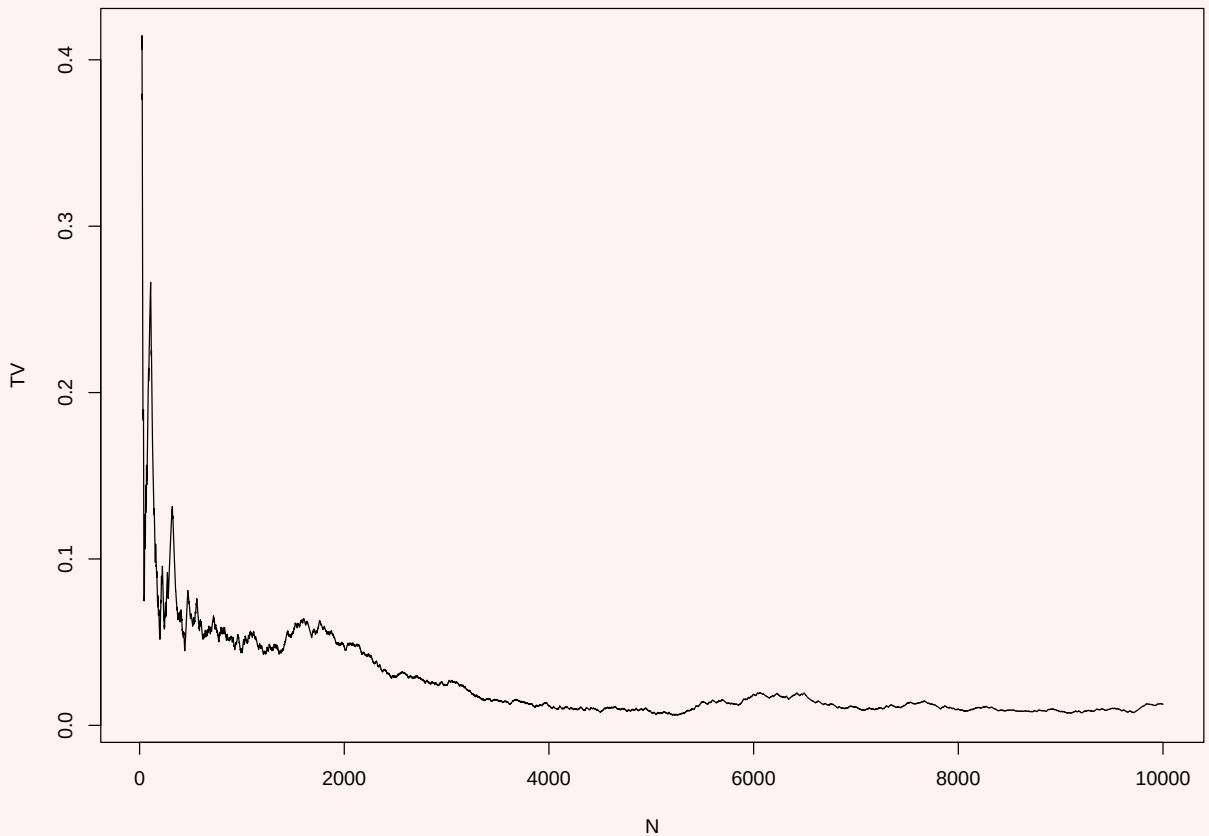
Here we take f_1 to be the true Poisson mass function π, and f_2 to be the estimated mass function $\widehat{\pi}$.

We monitor this distance as N increases.

Constructing reversible chains: example

Example: Poisson distribution

$d_{TV}(\pi, \widehat{\pi})$ as N increases.



Measuring convergence

Note: For continuous densities the total variation distance is given by

$$d_{TV}(f_1, f_2) = \frac{1}{2} \int |f_1(\mathrm{x}) - f_2(\mathrm{x})| \, dx$$

In general, the total variation distance $d_{TV}(f_1, f_2)$ can be hard to compute.

Measuring convergence

Recall the relationship between $d_{TV}(f_1, f_2)$ and the *Hellinger* distance $d_H(f_1, f_2)$

$$d_H(f_1, f_2) = \sqrt{\sum_{\mathrm{x}} \left(\sqrt{f_1(\mathrm{x})} - \sqrt{f_2(\mathrm{x})}\right)^2}$$

that is,

$$\frac{1}{2} d_H^2(f_1, f_2) \leqslant d_{TV}(f_1, f_2) \leqslant d_H(f_1, f_2)$$

These inequalities also hold in the continuous case. Note that

$$d_{TV}(f_1, f_2) \longrightarrow 0 \qquad \Longleftrightarrow \qquad d_H(f_1, f_2) \longrightarrow 0.$$

The Hellinger distance is sometimes easier to compute.

Discrete Markov chains: Recap

For an arbitrary discrete distribution, π, we can now construct a Markov chain that has π as its stationary distribution, and can produce dependent samples from the π by running the Markov chain and collecting the generated states.

- specify the stochastic matrix Q
- initialize the chain by setting X_0
- for each t, if $X_t = i$, use the ith row of Q as a discrete distribution for proposing Z_t
- If $Z_t = j$, accept $X_{t+1} = j$ with probability α_{ij}

$$\alpha_{ij} = \min\left\{1, \frac{\pi_j}{\pi_i}\frac{q_{ji}}{q_{ij}}\right\}$$

otherwise set $X_{t+1} = X_t = i$.

Discrete Markov chains: Recap

Example:

See `knitr 6`

Continuous State Space Markov Chains

The theory above extends (reasonably straightforwardly) to continuous state spaces, that is, the countable state set

$$\{s_1, \ldots, s_d\}$$

is replaced by a continuum of possible values, $\mathbb{X}$.

Continuous State Space Markov Chains

In this case, instead of having a transition matrix, we have a *transition kernel*

$$P(x, B) = \int_B P(x, z)\, dz$$

$P(x, B)$ determines the probability of making the transition from current value x into the set $B \subset \mathbb{X}$ in any given step.

Continuous State Space Markov Chains

We retain the *discrete time* nature of the Markov chain, and again consider outcome sequences $\{x_1, x_2, ..., x_n, ...\}$).

Transitions are implemented using a *transition density*

$$P(x, z) \equiv P(x \to z)$$

which specifies a conditional probability density in z, given the current value x, for $x, z \in \mathbb{X}$.

Continuous State Space Markov Chains

By analogy with the discrete case, the stationary distribution π for the continuous state space chain must satisfy

$$\pi(\mathrm{x}) = \int P(z, \mathrm{x})\pi(z)\, dz$$

A reversible chain must satisfy detailed balance

$$\pi(\mathrm{x})P(\mathrm{x}, z) = \pi(z)P(z, \mathrm{x})$$

for all x and z. Given P, we can in theory solve for π.

In the context of sampling from probability distributions, we wish to specify π, and then find a P such that its equilibrium distribution is π.

Harris recurrence

Suppose $\{X_n\}$ is a homogeneous Markov chain on a general state space $\mathbb{X}$ with transition kernel $P(x, .)$ which represents the one-step transition probability

$$\Pr[X_{n+1} \in B | X_n = x] = P(x, B) \qquad \forall x \in \mathbb{X}, B \in \mathbb{X}.$$

Harris recurrence

Then $\{X_n\}$ is a *Harris chain* if there exists $A \subseteq \mathbb{X}$, $\epsilon > 0$, and probability measure $\nu(.)$ with $\nu(\mathbb{X}) = 1$ such that

(i) If $\tau_A = \inf\{n \geqslant 0 : X_n \in A\}$, then

$$\Pr[\tau_A < \infty | X_0 = x] = 1 \qquad \forall x \in \mathbb{X};$$

(ii) If $x \in \mathbb{X}$ and $B \subseteq \mathbb{X}$, then

$$P(x, B) \geqslant \epsilon \nu(B).$$

Harris recurrence

(i) ensures that the chain returns to the A with probability 1, regardless of where it starts, so that it visits A infinitely often (with probability 1).

(ii) implies that once the chain is in A, its next state can be generated with the help of a Bernoulli draw:

- (ii) ensures that $0 \leqslant \epsilon \leqslant 1$: simply set $B = \mathbb{X}$;
- suppose $x \in A$ and set $X_n = x$;
- to generate X_{n+1}, draw $Z \sim \mathit{Bernoulli}(\epsilon)$: if $Z = 1$, generate X_{n+1} according to ν on $\mathbb{X}$, otherwise generate X_{n+1} according to the distribution

$$\Pr[X_{n+1} \in B | X_n = x] = \frac{\big(P(x, B) - \epsilon\nu(B)\big)}{1 - \epsilon}$$

Harris recurrence

Suppose that $X_0 \sim p^{(0)}$ and define for arbitary A

$$\tau_A = \inf\{n \geqslant 1 : X_n \in A\}$$

that is, τ_A is the first 'entry time' for A.

The chain is *Harris recurrent* if, for all $p^{(0)}$,

$$\Pr[\tau_A < \infty | X_0 \in A] = 1$$

and *aperiodic* if there exists n_A such that for all $n \geqslant n_A$

$$\Pr[X_n \in A | X_0 \in A] > 0.$$

Harris recurrence

If $\{X_n\}$ is an aperiodic, Harris recurrent chain with stationary distribution π, then provided

$$\Pr[\tau_A < \infty | X_0 = x] = 1$$

for $x \in \mathbb{X}$ we have that

$$d_{TV}(p^{(n)}, \pi) \longrightarrow 0$$

as $n \longrightarrow \infty$, where for $B \subseteq \mathbb{X}$,

$$p^{(n)}(B) = \Pr[X_n \in B | X_0 = x].$$

The Metropolis-Hastings Algorithm

We attempt to mimic the construction of a Markov chain with stationary distribution π used in the discrete case.

Let Q be any proposal (transition) kernel suitable for moving (exhaustively) around $\mathbb{X}$, with associated transition density q such that

$$q\left(z, \mathrm{x}\right) = q(z \longrightarrow \mathrm{x}) > 0$$

for all x, z.

In fact, this can be relaxed to the condition that requires $Q^n\left(x, z\right) > 0$ for all $x, z \in \mathbb{X}$, separated by n steps in the chain).

The Metropolis-Hastings Algorithm

Then, for $z \neq x$, define

$$P(x, z) = q(x, z)\,\alpha(x, z)$$

where

$$\alpha(x, z) = \min\left\{1, \frac{\pi(z)}{\pi(x)}\frac{q(z, x)}{q(x, z)}\right\}$$

defines an *acceptance probability* for the move from x to z.

The Metropolis-Hastings Algorithm

Under this transition kernel or density P with transition density if the current state of the chain at step n is $X_n = x$, then the next value of the chain is either

- a *new value* $X_{n+1} = z$, generated from the conditional density $q(x, z)$,
- or the *current value* $X_{n+1} = x$.

The value z the proposed or *candidate* state.

Thus, starting from the n^{th} step when $x_n = x$, we have the following algorithm for implementing the continuous state space Markov chain:

The Metropolis-Hastings Algorithm

1. Generate candidate z from the conditional density $q(.,.)$ given x
2. Compute $\alpha(x, z)$
3. Generate u from $Uniform(0, 1)$
   - if $u \leqslant \alpha(x, z)$, *accept* the move to z and set $X_{n+1} = z$
   - if $u > \alpha(x, z)$, *reject* the move to z and set $X_{n+1} = x$
4. Return to 1 to generate X_{n+2}

and so on.

This is the *Metropolis-Hastings algorithm*

The Metropolis Algorithm

The general algorithm above has some special cases of interest. If q is chosen such that

$$q\left(\mathrm{x}, z\right) = q\left(z, \mathrm{x}\right)$$

so that q is symmetric in its arguments, then

$$\alpha\left(\mathrm{x}, z\right) = \min\left\{1, \frac{\pi\left(z\right)}{\pi\left(\mathrm{x}\right)}\right\}$$

and the move to z is accepted with certainty if the target probability density at z is higher than at x.

The Metropolis Algorithm

A simple symmetric transition density has

$$Z|X_n = x \sim \mathcal{N}\left(x, \sigma_q^2\right)$$

Choosing σ_q^2 small encourages many small moves.

This is the original Markov chain simulation algorithm, known as the *Metropolis Algorithm*.

Many such “local" moves can be proposed. Note that it is important to respect any parameter constraints in the proposal.

Independence Metropolis-Hastings

The *independence Metropolis-Hastings algorithm* uses

$$q(x, z) = q(z)$$

that is, independent of the current value of the chain. This still defines a Markov chain as

$$p(x, z) = q(z)\,\alpha(x, z)$$

still depends on x through $\alpha(x, z)$.

A good independence Markov chain (that traverses $\mathbb{X}$ quickly) is more difficult to construct without knowledge of π.

However, if π can be well-approximated by a density q (as in rejection sampling), then this method can work well.

Metropolis-Hastings algorithm

Example: Gamma density

Suppose, for $\gamma > 0$

$$\pi(x) = \frac{1}{\Gamma(\gamma)} x^{\gamma-1} e^{-x} \qquad x > 0.$$

Suppose, first that $q(x, z)$ is specified as a *reflected* normal density, that is, we propose z by simulating

$$Y|X_t = x \sim \mathcal{N}(x, \sigma_q^2),$$

and setting $Z = |Y|$.

Metropolis-Hastings algorithm

Example: Gamma density

Note that

$$\Pr[Z \leqslant z|X = x] = \Pr[|Y| \leqslant z|X = x] = \Pr[-z \leqslant Y \leqslant z|X = x]$$

so therefore

$$\Pr[Z \leqslant z|X = x] = \Phi((z - x)/\sigma_q) - \Phi((-z - x)/\sigma_q)$$

and, on differentiation wrt z,

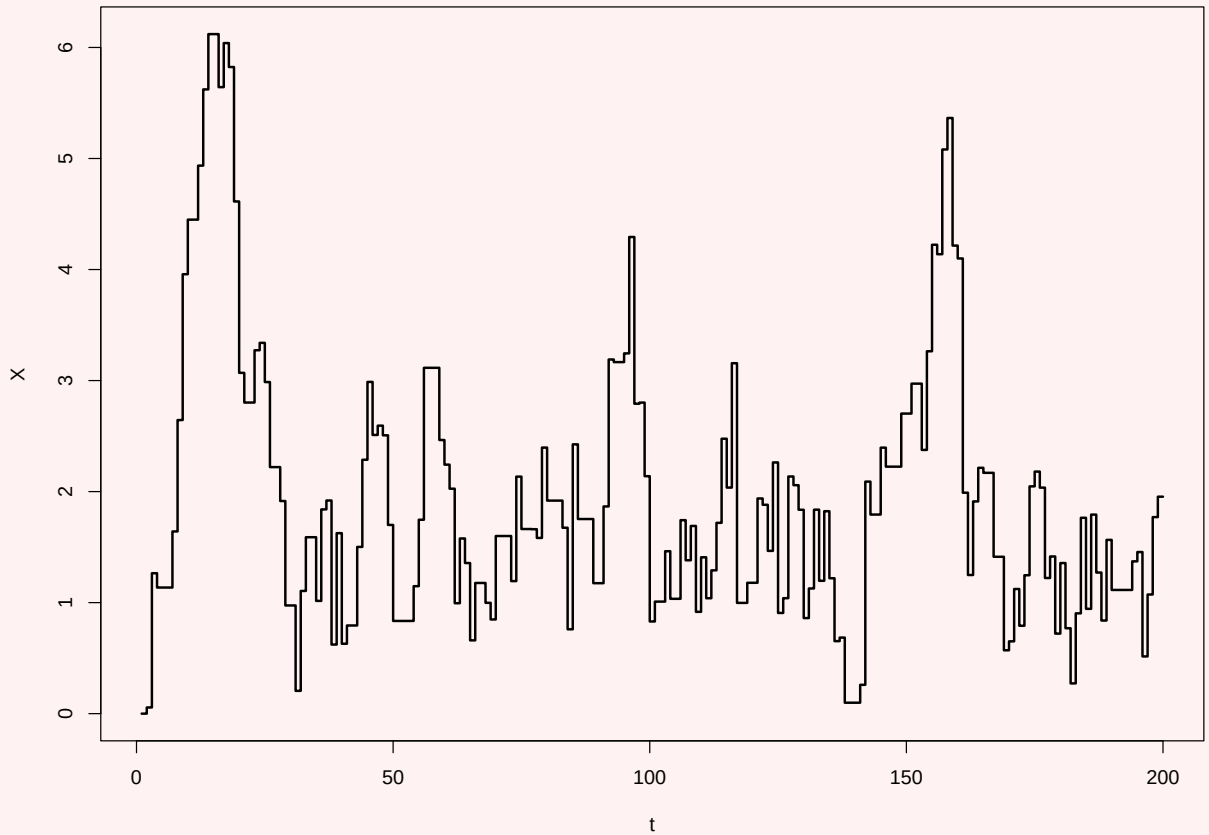
$$q(x, z) = \frac{1}{\sigma_q}(\phi((z - x)/\sigma_q) + \phi((-z - x)/\sigma_q)) = q(z, x)$$

as ϕ is an even function.

Metropolis-Hastings algorithm

Example: Gamma density

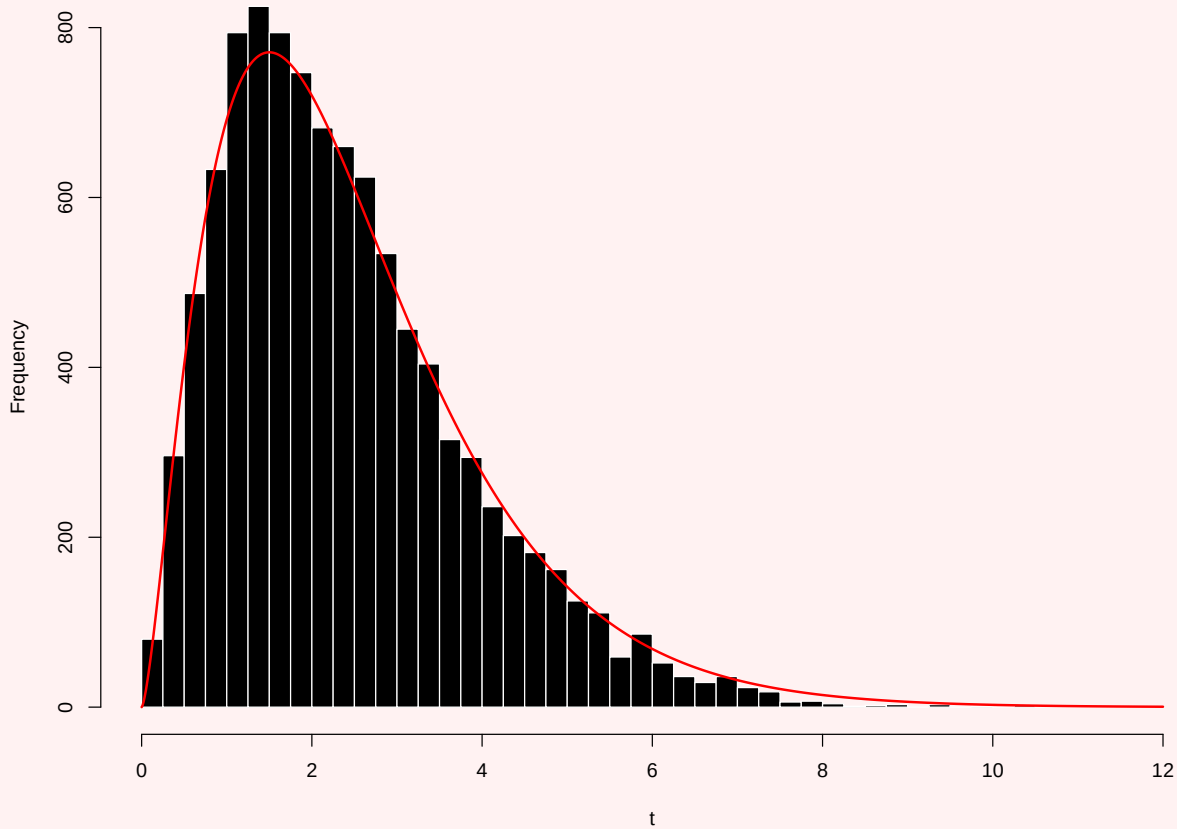
First 200 steps of the chain starting at $X_0 = 0$ with $\gamma = 2.5$, $\sigma_q = 1$.



Metropolis-Hastings algorithm

Example: Gamma density

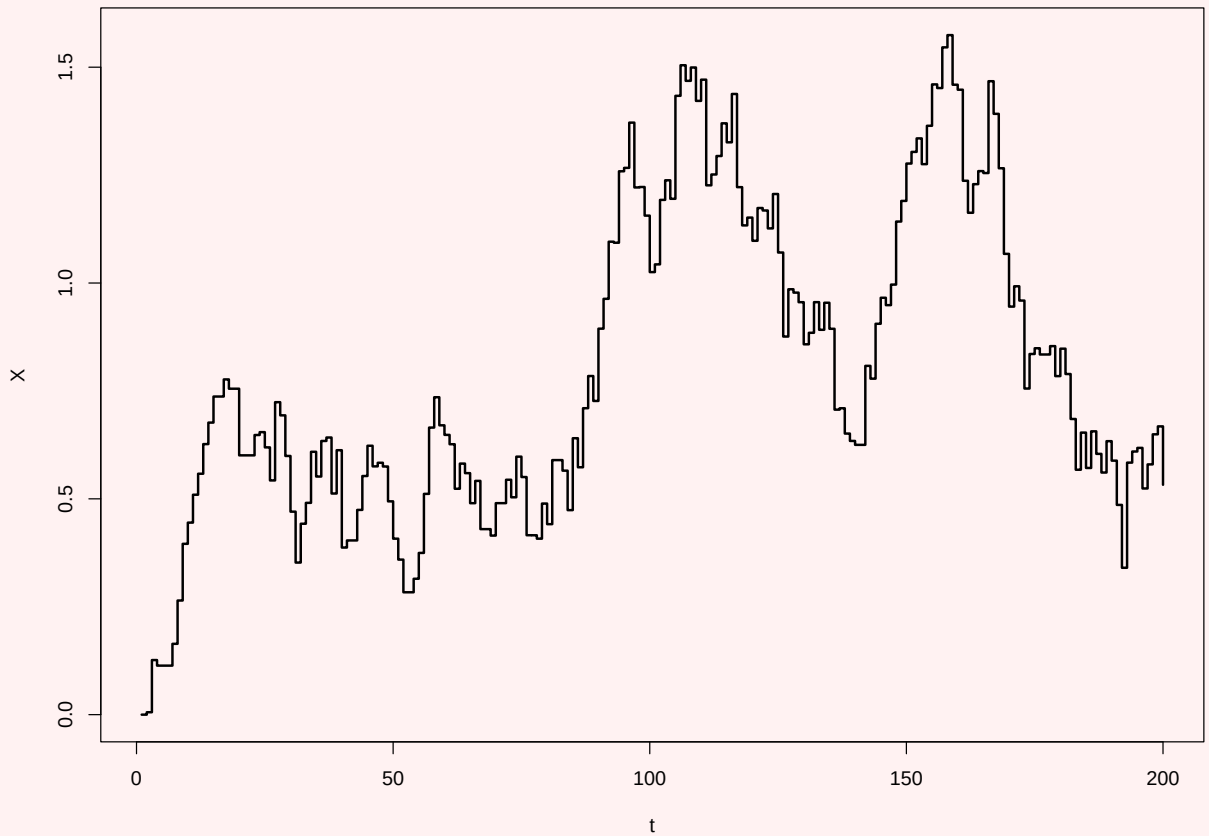
Histogram of states visited over $N = 10000$ steps (with Gamma density (solid))



Metropolis-Hastings algorithm

Example: Gamma density

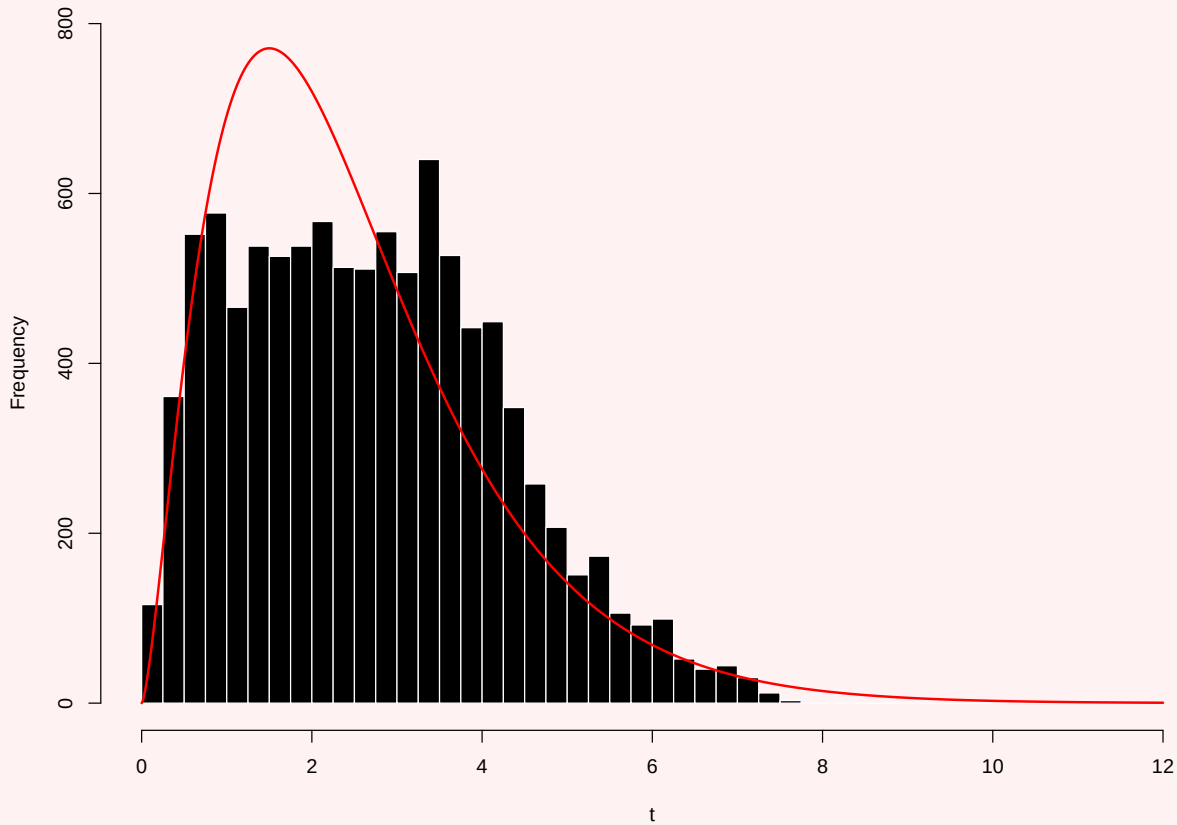
First 200 steps of the chain starting at $X_0 = 0$ with $\gamma = 2.5$, $\sigma_q = 0.1$.



Metropolis-Hastings algorithm

Example: Gamma density

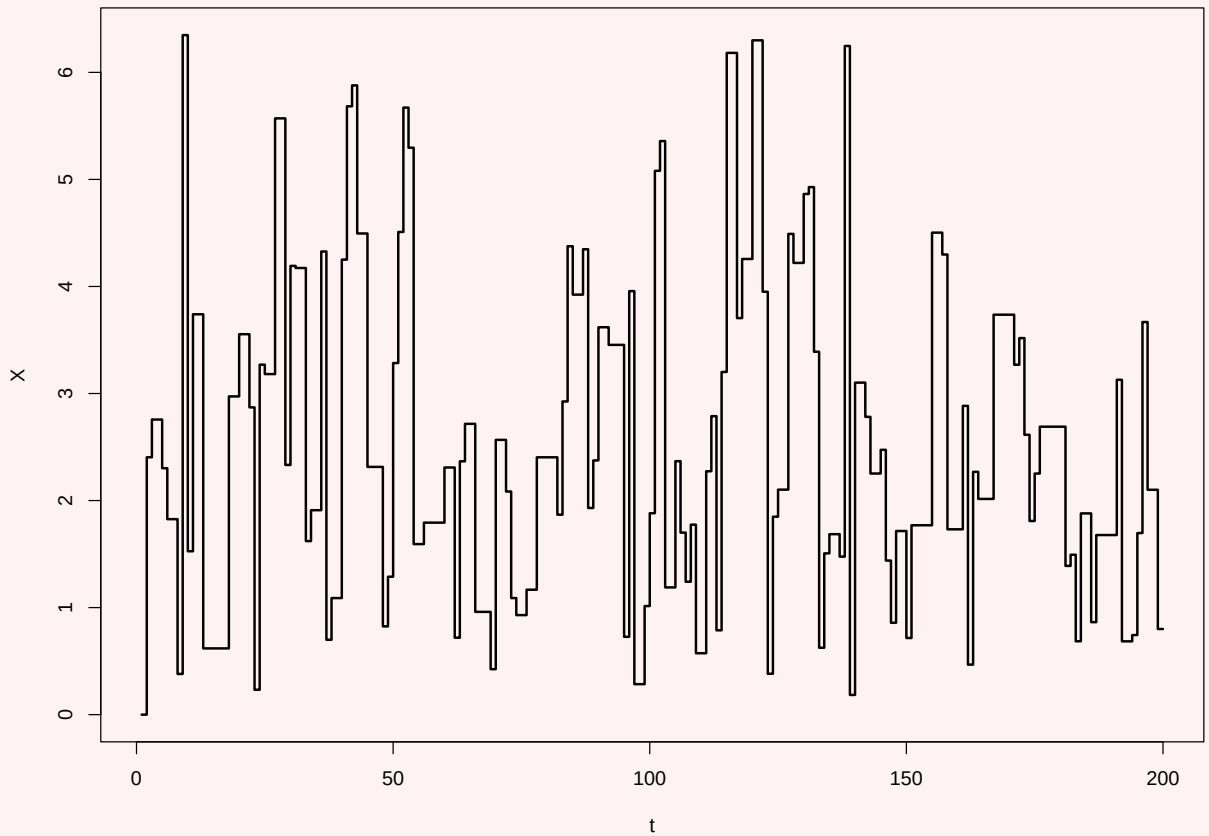
Histogram of states visited over $N = 10000$ steps (with Gamma density (solid))



Metropolis-Hastings algorithm

Example: Gamma density

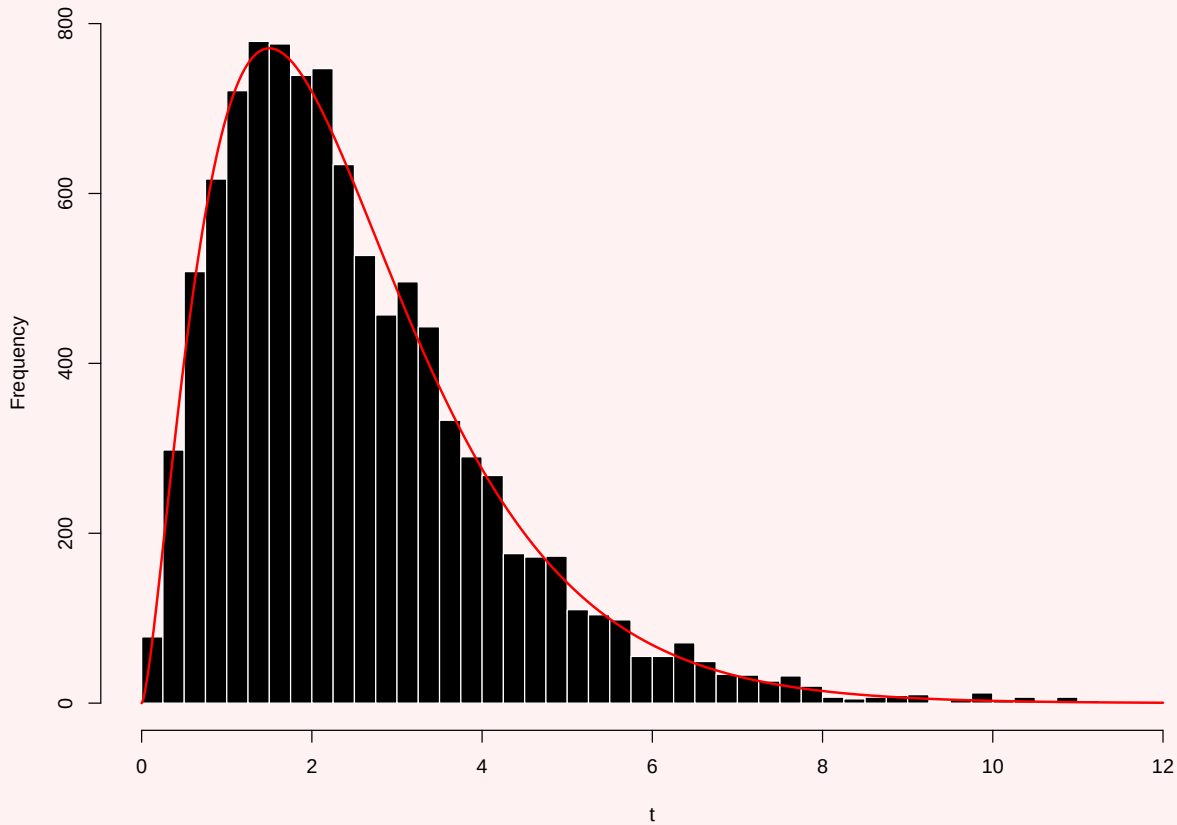
First 200 steps of the chain starting at $X_0 = 0$ with $\gamma = 2.5$, $\sigma_q = 3$.



Metropolis-Hastings algorithm

Example: Gamma density

Histogram of states visited over $N = 10000$ steps (with Gamma density (solid))



Metropolis-Hastings algorithm

The states generated by the Markov chain are correlated; we can assess the performance of the Markov chain by examining the sample autocorrelation function

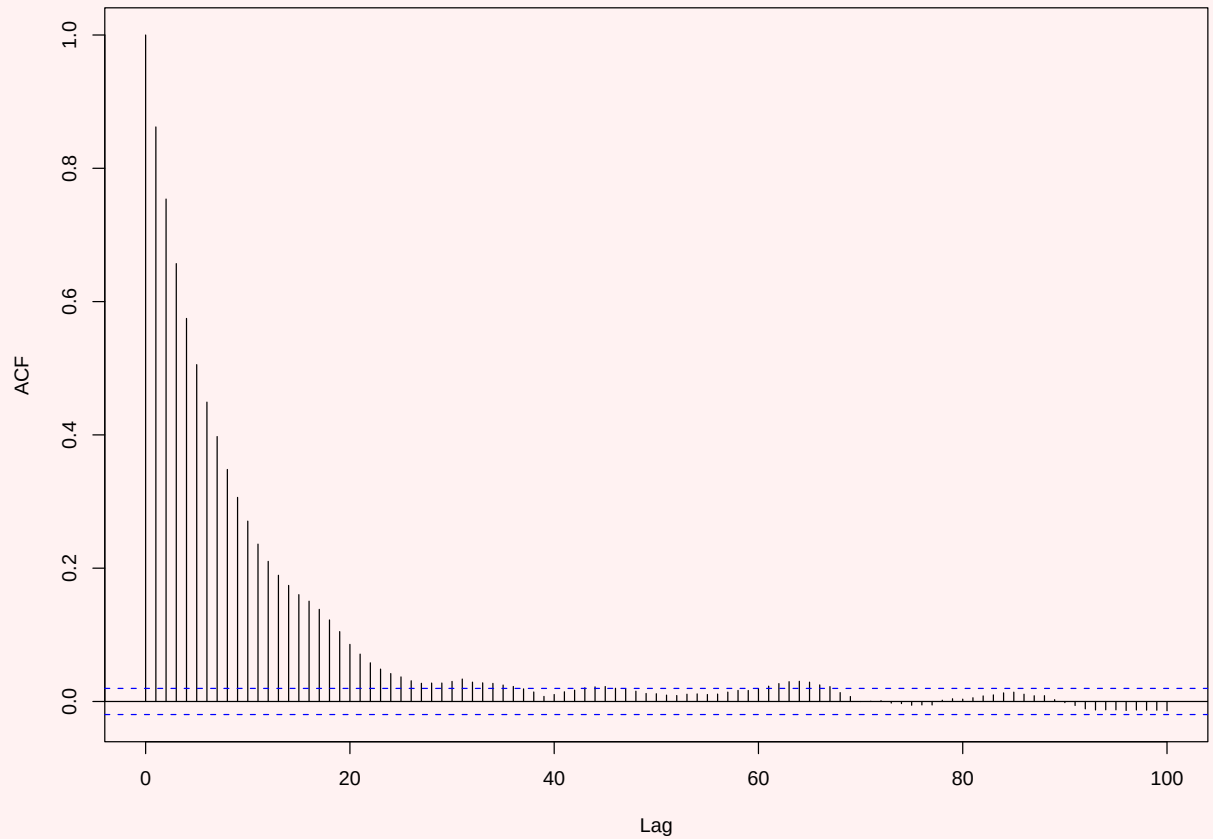
$$r(k) = \left(\frac{N-1}{N-k-1} \right) \frac{\sum_{t=k+1}^{N} (x_t - \overline{x})(x_{t-k} - \overline{x})}{\sum_{t=1}^{N} (x_t - \overline{x})^2}$$

for $k = 0, 1, 2 \ldots$.

A chain with high autocorrelation for large k is typically slow to converge.

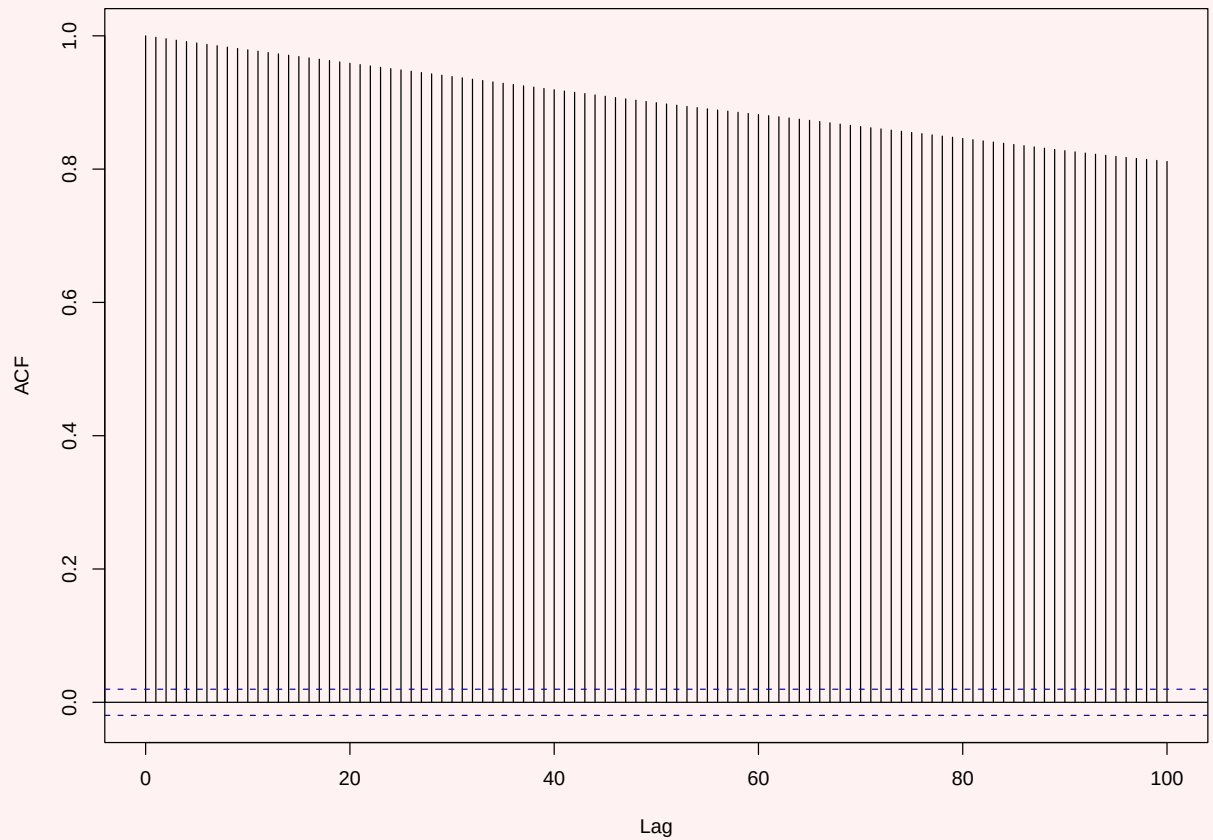
Example: Gamma density

Autocorrelation function for $\sigma_q = 1$.



Example: Gamma density

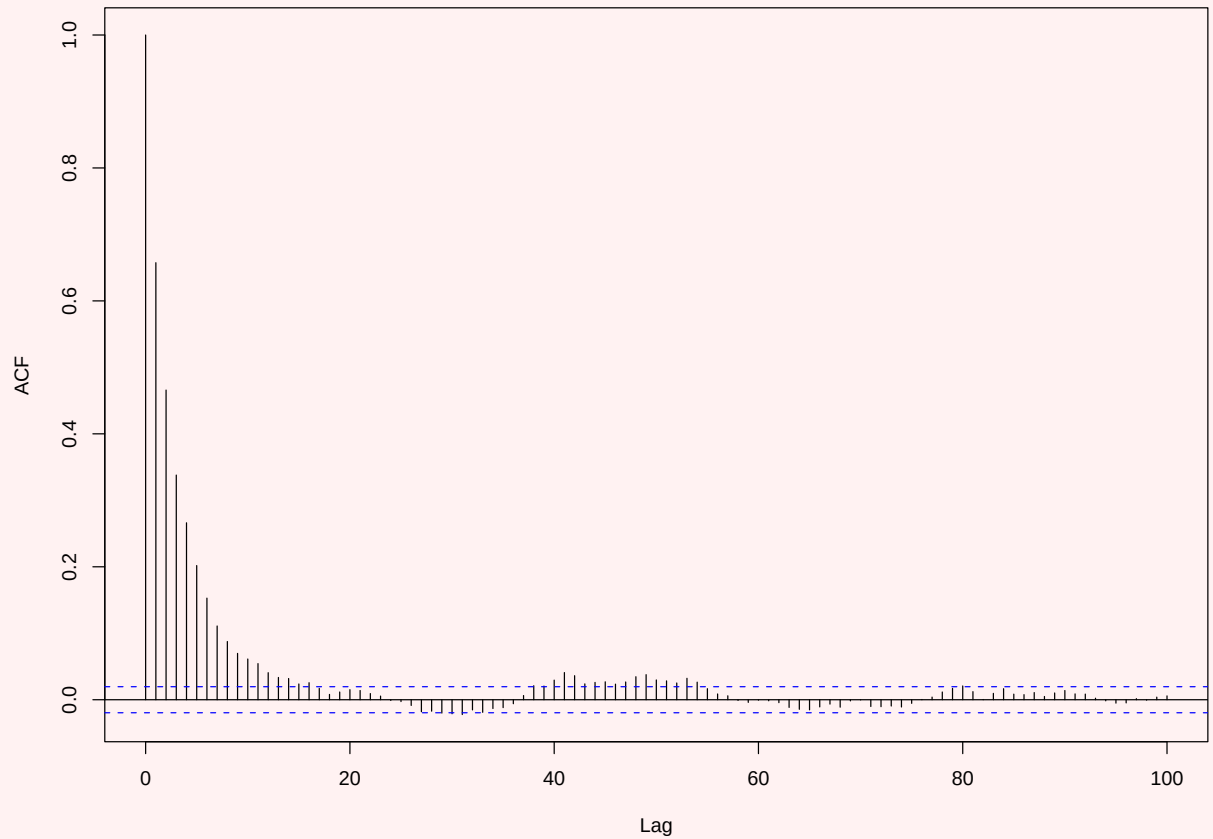
Autocorrelation function for $\sigma_q = 0.1$: inferior performance.



Metropolis-Hastings algorithm

Example: Gamma density

Autocorrelation function for $\sigma_q = 3$: superior performance.



Metropolis-Hastings algorithm

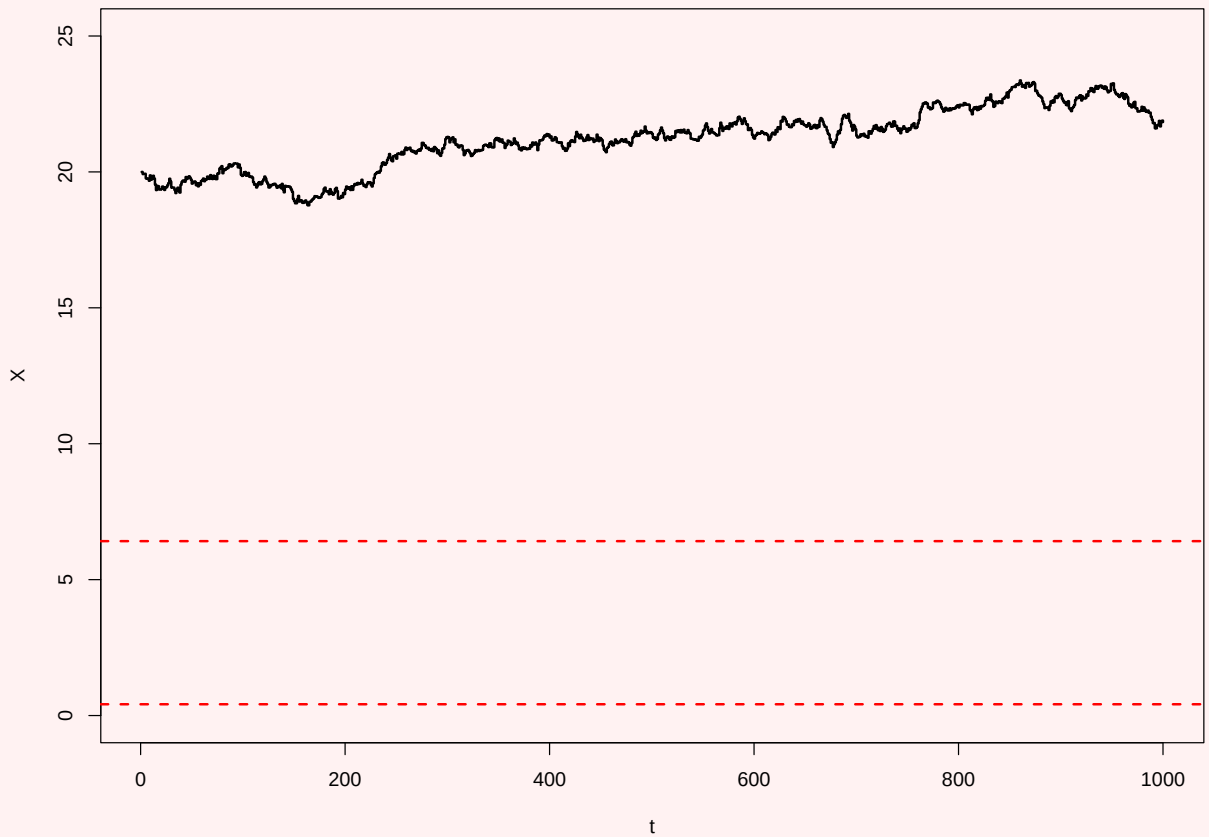
If the chain is started away from the high-probability region of π, then the it can take many steps to return there.

In the following trace plots, the red dashed lines give the 0.025 and 0.975 quantiles of the $Gamma(2.5, 1)$ distribution from the example. The chain is initialized at $X_0 = 20$, and then run for 20000 steps.

Metropolis-Hastings algorithm

Example: Gamma density

Starting value $X_0 = 20$, $\sigma_q = 0.1$: first 1000 steps.



Metropolis-Hastings algorithm

Example: Gamma density

Starting value $X_0 = 20$, $\sigma_q = 0.1$: first 10000 steps.

Metropolis-Hastings algorithm

Example: Gamma density

Starting value $X_0 = 20$, $\sigma_q = 0.1$: first 20000 steps.

Metropolis-Hastings algorithm

Example: Gamma density

Starting value $X_0 = 20$, $\sigma_q = 0.1$: steps 15000 to 20000.

Metropolis-Hastings algorithm

In this final section of the chain, the generated values appear to oscillate, despite the fact that the chain has apparently reached the stationary phase. This oscillation is a result of the high autocorrelation present in the chain.

It is often difficult to distinguish such high autocorrelation from the case where a chain has not converged.

The high autocorrelation results here from the choice $\sigma_q = 0.1$; this value is smaller than is optimal.

Metropolis-Hastings algorithm

Example:

See `knitr 7`

Gibbs Sampler

The Metropolis-Hastings algorithm above is valid for both univariate and multivariate probability distributions, but is more complicated in high dimensions.

The objective is to choose a transition density q that moves around the space $\mathbb{X}$ quickly, which means that we wish to have the acceptance probability reasonably large.

In high dimensions, this is often difficult to achieve. The *Gibbs Sampler* algorithm attempts to solve this problem by breaking down a high-dimensional problem into several lower dimensional problems that are solved iteratively and simultaneously.

Gibbs Sampler

Suppose that π is a probability density in K dimensions, and let the variables be denoted $(X_1, ..., X_K)$. Define the conditional density $\pi_k(.|.)$ for

$$X_k | X_1, ..., X_{k-1}, X_{k+1}, ...X_K$$

by

$$\pi_k\left(x_k; x_{(k)}\right) = \frac{\pi\left(x_1, ..., x_K\right)}{\pi\left(x_1, ..., x_{k-1}, x_{k+1}, ...x_K\right)} \propto \pi\left(x_1, ..., x_K\right)$$

where the denominator is the marginal distribution of $X_{(k)}$, the $K-1$ variables excluding X_k.

The Gibbs Sampler utilizes this set of K *full conditional distributions* to construct a Markov chain on the support of the joint distribution.

Gibbs Sampler

It is implemented using the following algorithm:

1. Set a vector of starting values for the K variables $\left(x_{10}, ..., x_{K0}\right)$.
2. Sample in turn from the conditional distributions as follows:
   (a) sample x_{11} from $\pi_1\left(x_1; x_{20}, x_{30}, ..., x_{K0}\right)$
   (b) sample x_{21} from $\pi_2\left(x_2; x_{11}, x_{30}, ..., x_{K0}\right)$
   (c) sample x_{31} from $\pi_3\left(x_3; x_{11}, x_{21}, ..., x_{K0}\right)$
   $\vdots$
   (K) sample x_{K1} from $\pi_K\left(x_K; x_{11}, x_{21}, ..., x_{K-1\ 1}\right)$

   This completes one step of the Gibbs sampler.
3. Return to 2 (a), and repeat to obtain, at step t, the sampled variates $\left(x_{1t}, x_{2t}, ..., x_{Kt}\right)$

Gibbs Sampler

This Markov chain defines, in steps 2(a)-2(K), a means of updating the vector x_t to vector x_{t+1}. Each of the steps can be achieved using direct sampling from the conditional distribution if that is possible, but can also involve individual Metropolis-Hastings steps, with acceptance probabilities

$$\alpha_k(x, z) = \min\left\{1, \frac{\pi_k\left(z; x_{(k)}\right)}{\pi_k\left(x; x_{(k)}\right)} \frac{q_k(z, x)}{q_k(x, z)}\right\}$$

for $k = 1, ..., K$. In 2., the steps can also be completed in *random order*

Finally, these steps can be achieved with the scalar variables X_1,...,X_K or with these components as vector variables; deciding on which *blocks* of variables to update simultaneously is often a key issue.

Gibbs Sampler as Metropolis-Hastings

The Gibbs sampler is in fact a special case of the Metropolis-Hastings algorithm: we can regard the individual updates in 2. as implementing K separate transition kernels $P_1, \ldots, P_K$ that act on the components of the vector $\mathbf{X}$; note that these kernels in isolation yield *reducible* Markov chains.

A more general form of MH algorithm is based on a *mixture* transition kernel

$$P(\mathrm{x}, B) = \sum_j \omega_j P_j(\mathrm{x}, B)$$

where $0 \leqslant \omega_j \leqslant 1$ and $\sum_j \omega_j = 1$, and the P_j are themselves transition kernels. This allows for the possibility of choosing several proposal densities q_j.

Gibbs Sampler: example

Example: Bivariate Normal

Suppose $\mathbf{X} = (X_1, X_2)^\top \sim \mathcal{N}_2(\mathbf{0}, \Sigma)$ where

$$\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

The general result for multivariate normal distribution conditional distributions is that if $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)^\top \sim \mathcal{N}_d(\boldsymbol{\mu}, \Sigma)$, where $\mathbf{X}_1$ is $(d_1 \times 1)$, and

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}$$

then

$$\mathbf{X}_1 | \mathbf{X}_2 = \mathbf{x}_2 \mathcal{N}_{d_1} \sim \left(\boldsymbol{\mu}_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \right)$$

Gibbs Sampler: example

Example: Bivariate Normal

Therefore, here $d = 2$, and

$$X_1|X_2 = x_2 \sim \mathcal{N}(\rho x_2, (1-\rho^2))$$

$$X_2|X_1 = x_1 \sim \mathcal{N}(\rho x_1, (1-\rho^2))$$

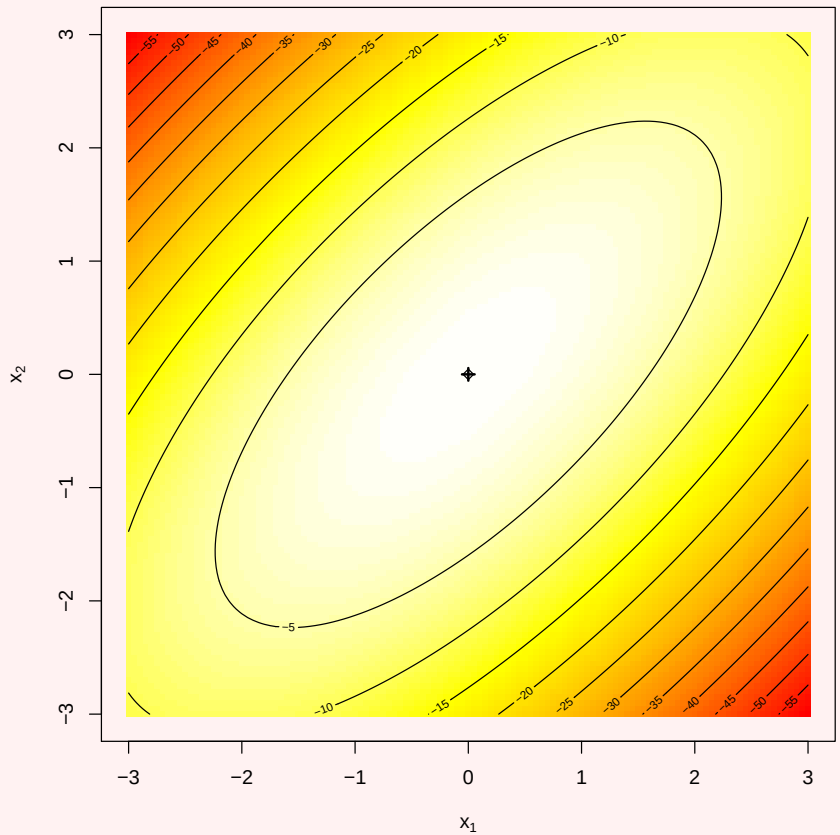
These distributions are sampled repeatedly with updating of the conditioning value after each sampling.

Suppose we start at $(x_{10}, x_{20}) = (0, 0)$.

Gibbs Sampler: example

Example: Bivariate Normal

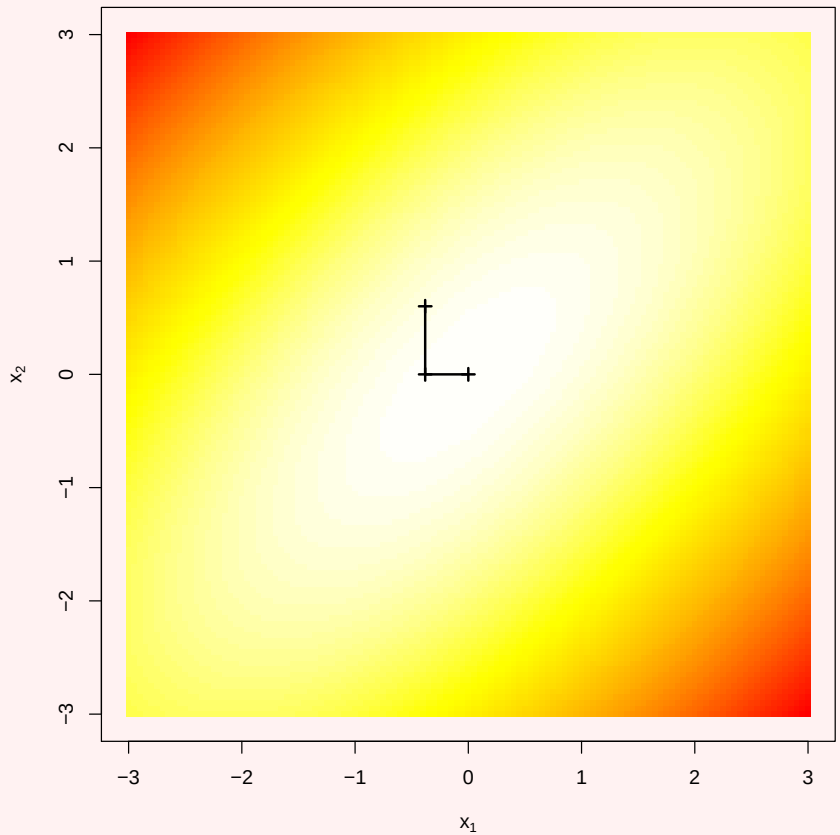
Initial point: (x_{10}, x_{20})



Gibbs Sampler: example

Example: Bivariate Normal

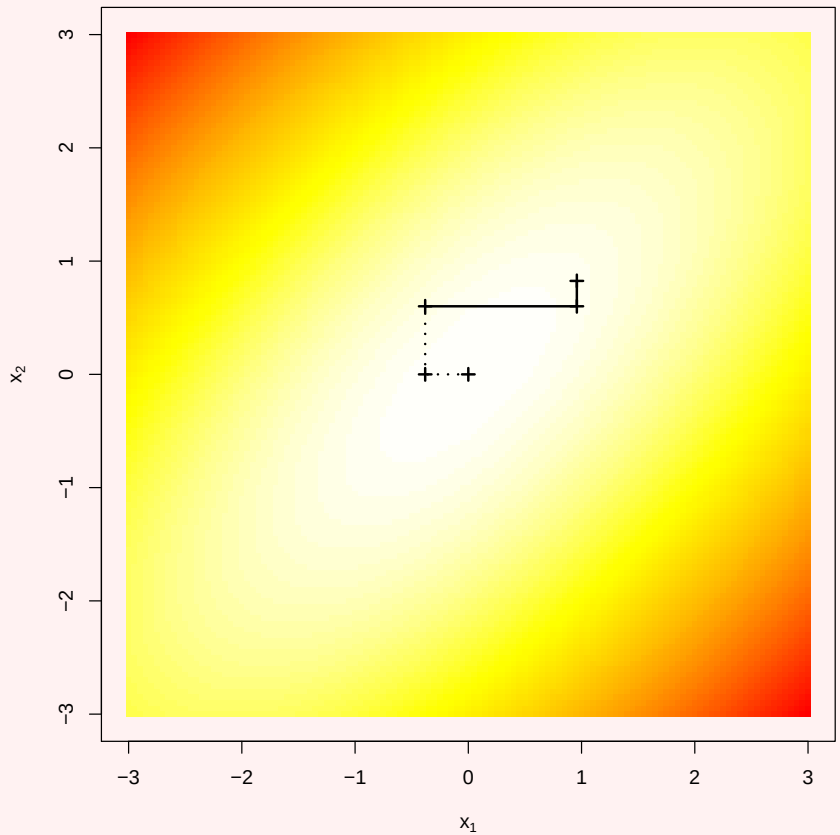
After one update: (x_{11}, x_{21})



Gibbs Sampler: example

Example: Bivariate Normal

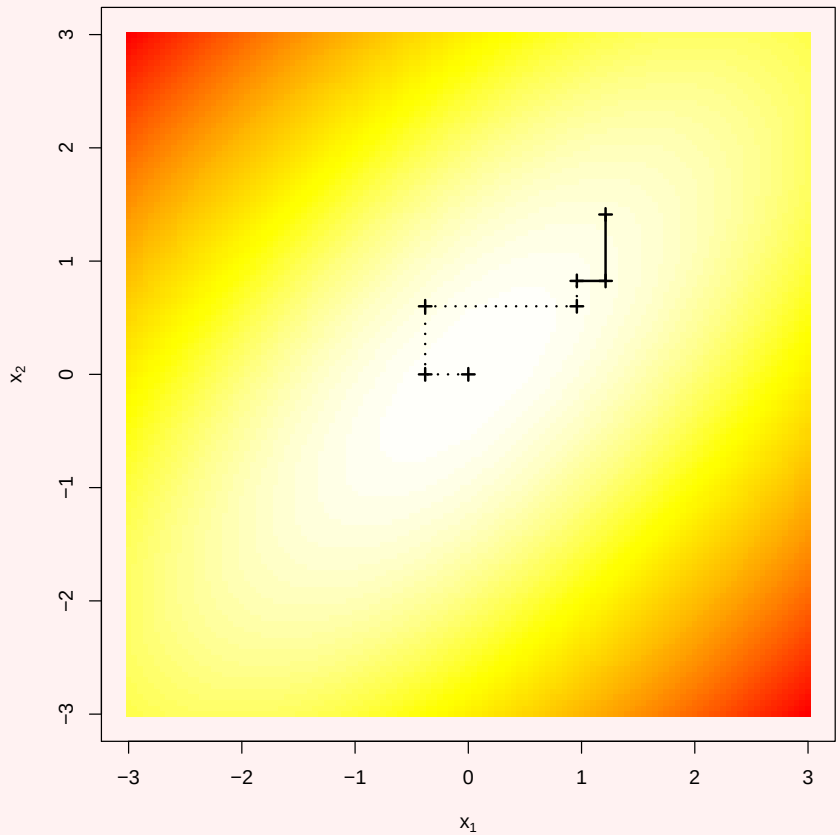
After two updates: (x_{12}, x_{22})



Gibbs Sampler: example

Example: Bivariate Normal

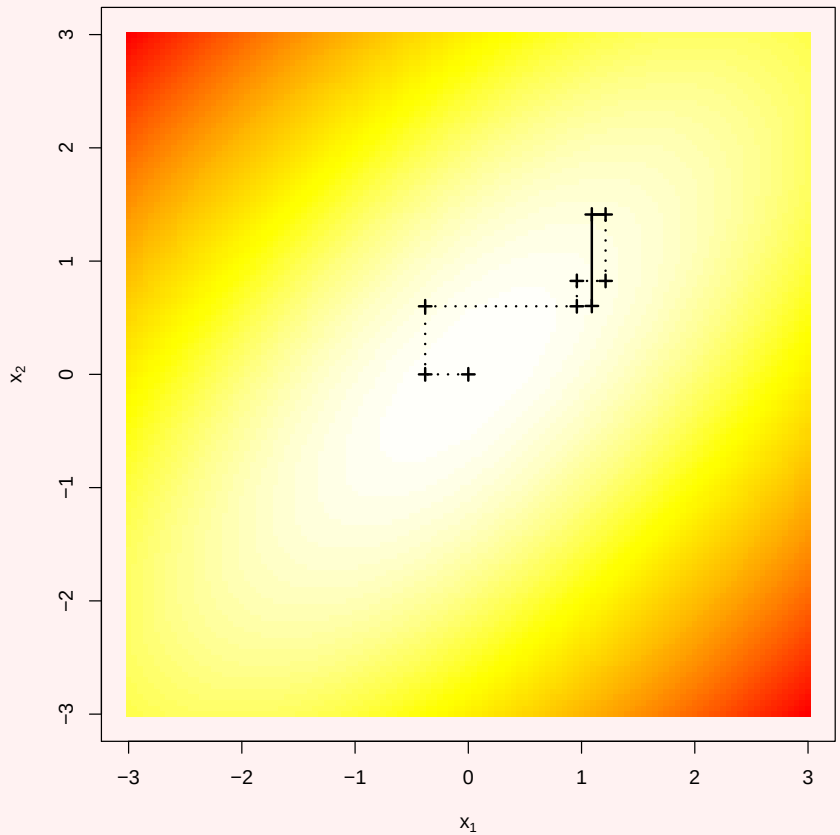
After three updates: (x_{13}, x_{23})



Gibbs Sampler: example

Example: Bivariate Normal

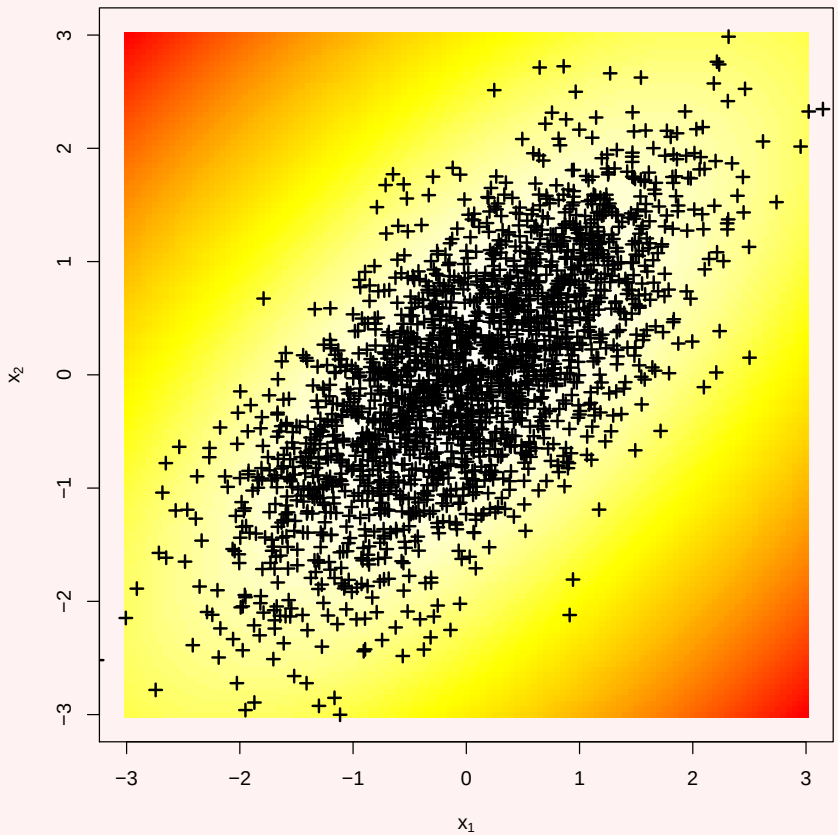
After four updates: (x_{14}, x_{24})



Gibbs Sampler: example

Example: Bivariate Normal

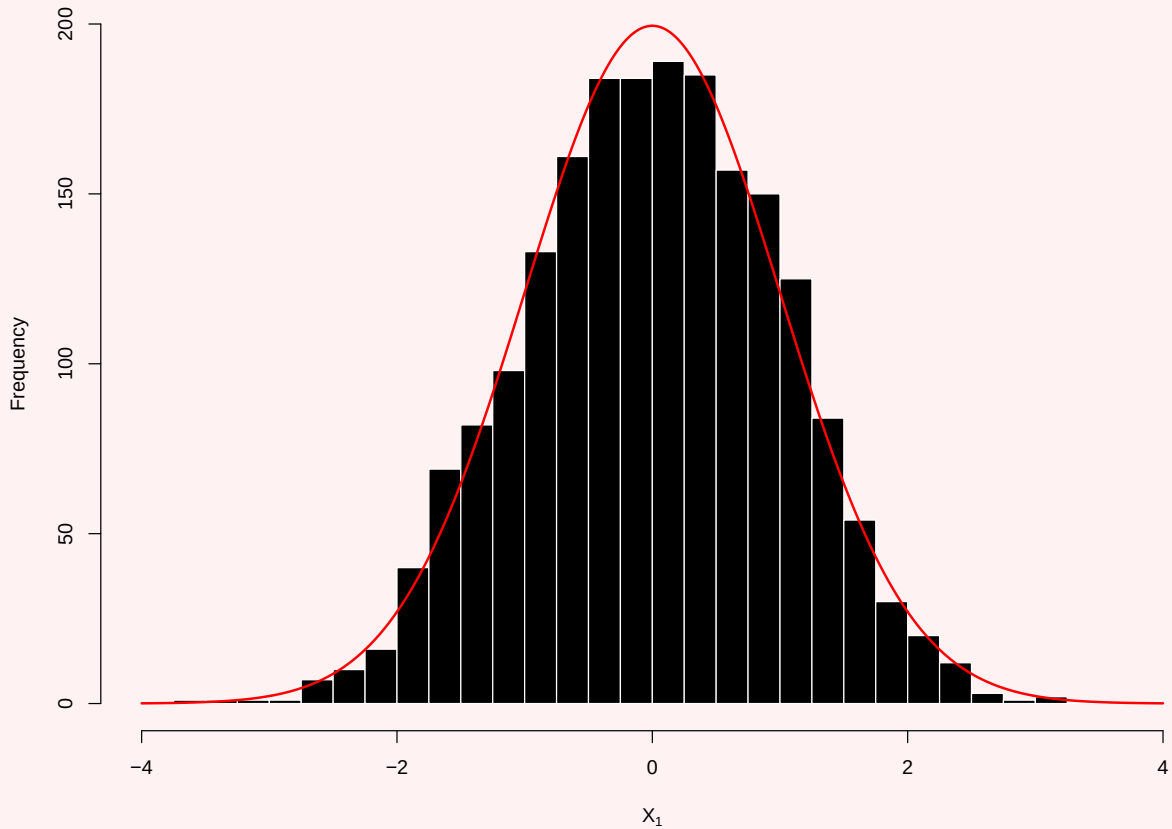
After 2000 updates: entire collected sample



Gibbs Sampler: example

Example: Bivariate Normal

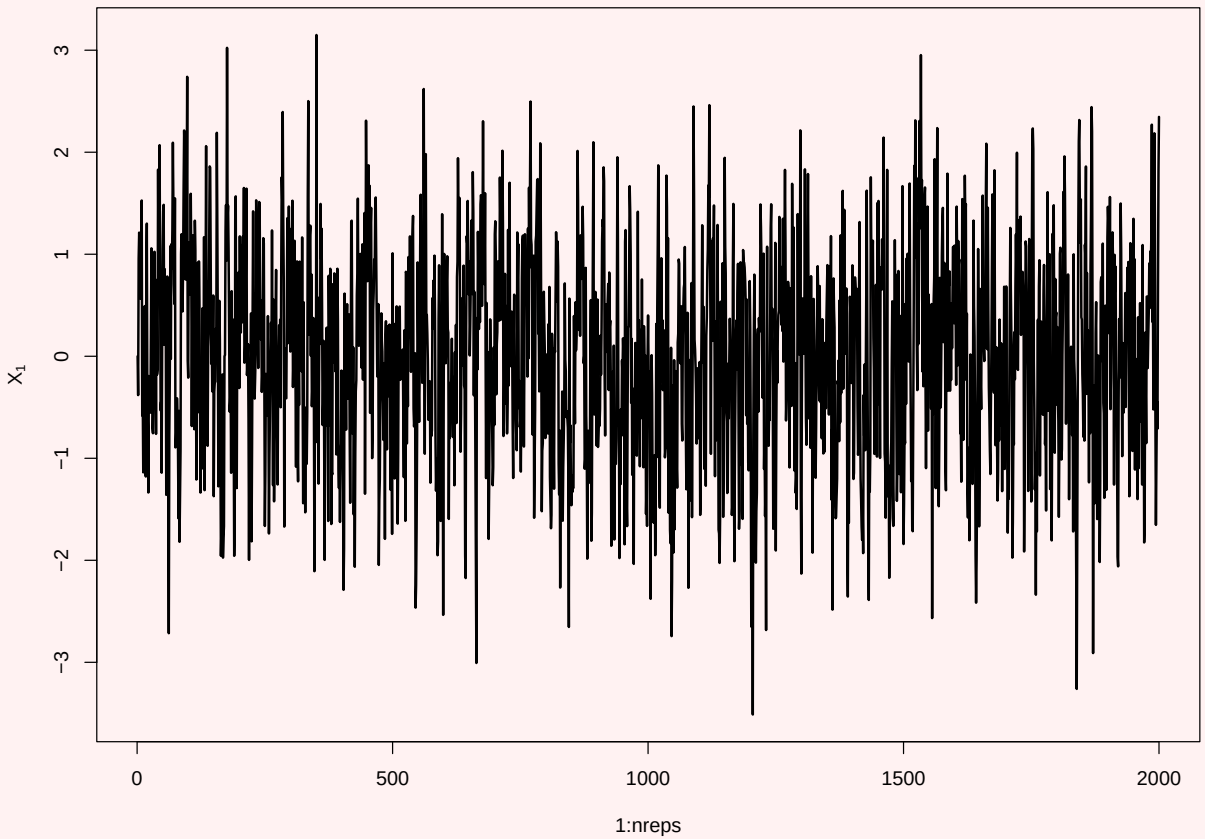
Histogram for X_1 with true marginal density (solid):



Gibbs Sampler: example

Example: Bivariate Normal

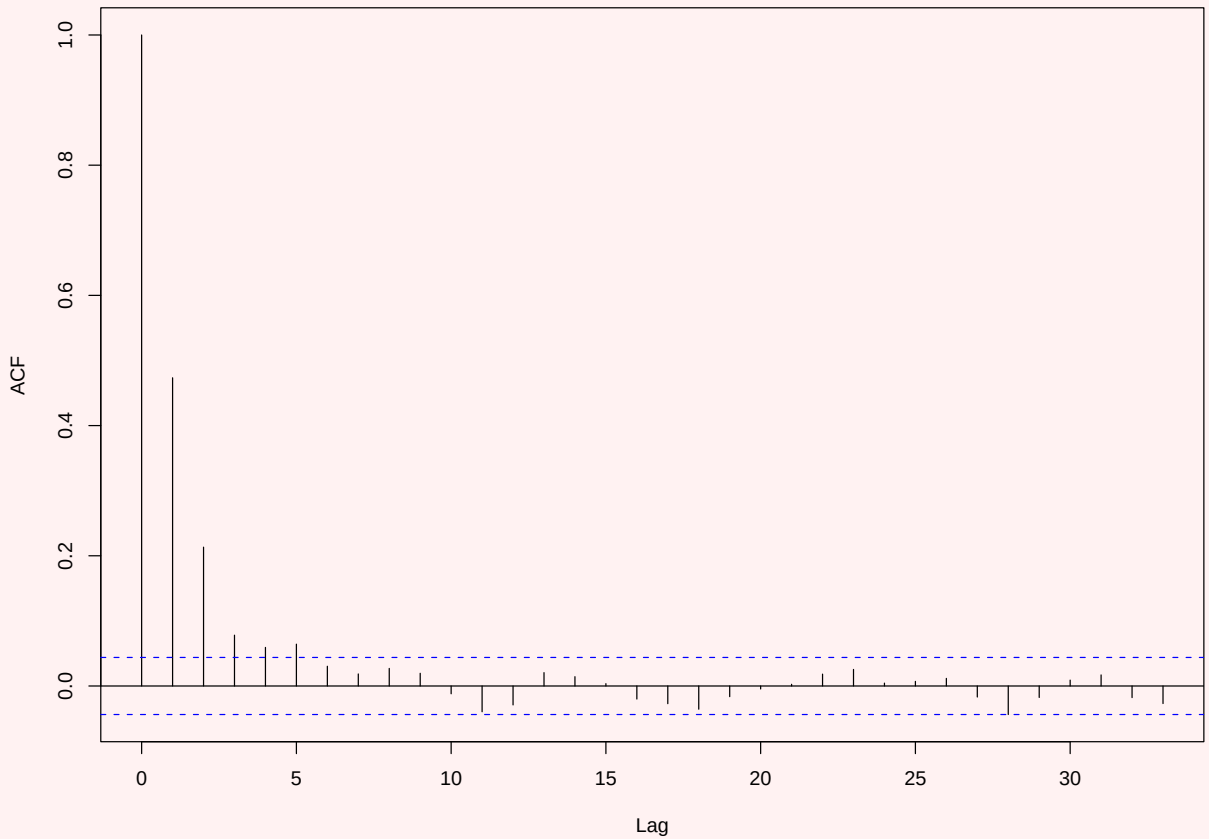
Trace for X_1:



Gibbs Sampler: example

Example: Bivariate Normal

ACF for X_1:



Gibbs Sampler: example

Example: Bivariate Normal

We note that

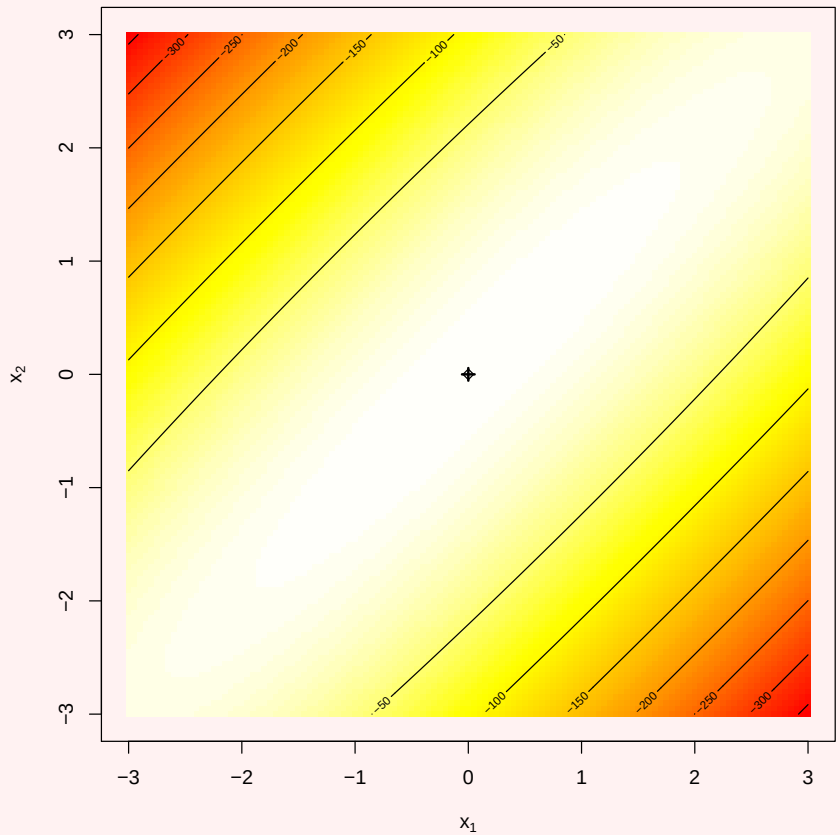
- the Gibbs sampler makes one-step moves along the coordinate axes
- the moves can traverse the support of the joint density fairly well
- there is no “tuning" of a proposal parameter (like σ_q)
- the samples of x_1 that are collected across steps are (dependent) samples from the correct marginal distribution for X_1; the same result holds for X_2

We can re-run the Gibbs sampler from the same starting value, but now with $\rho = 0.95$.

Gibbs Sampler: example

Example: Bivariate Normal

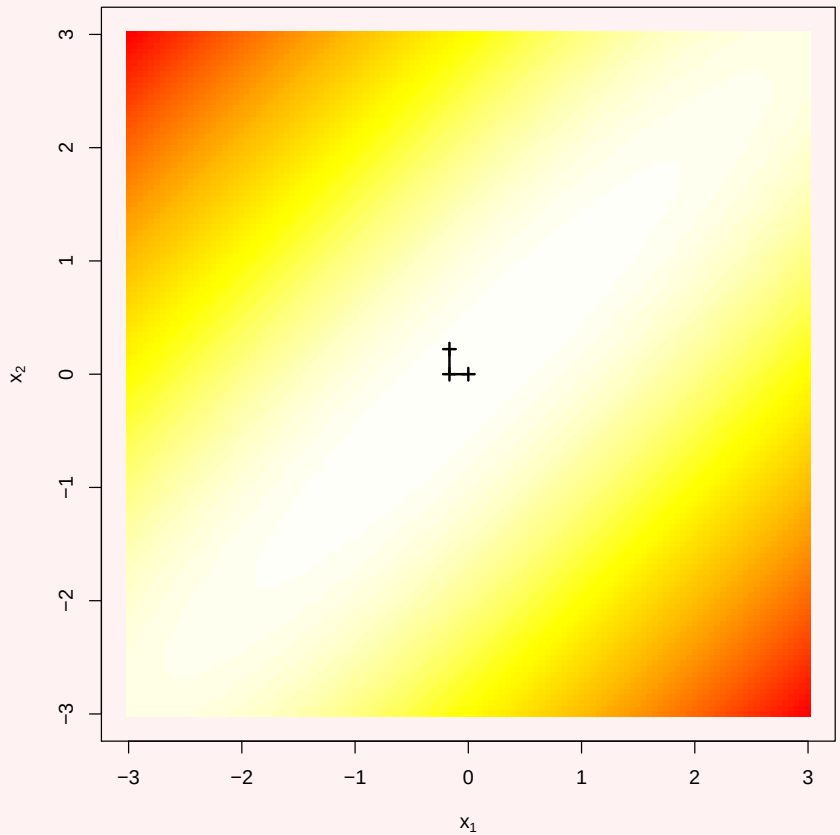
Initial point: (x_{10}, x_{20})



Gibbs Sampler: example

Example: Bivariate Normal

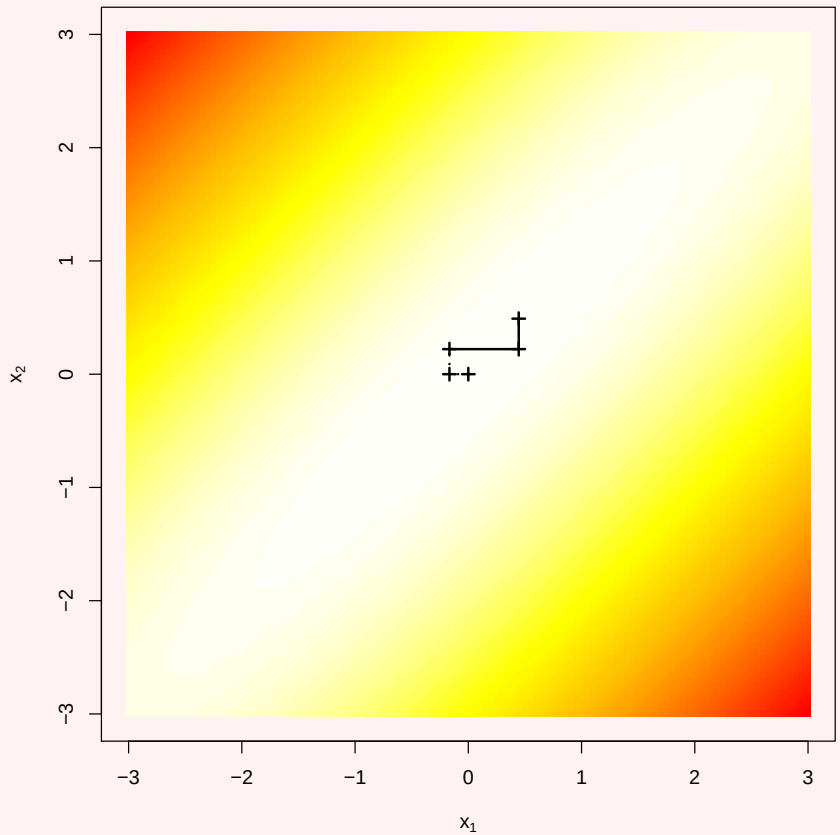
After one update: (x_{11}, x_{21})



Gibbs Sampler: example

Example: Bivariate Normal

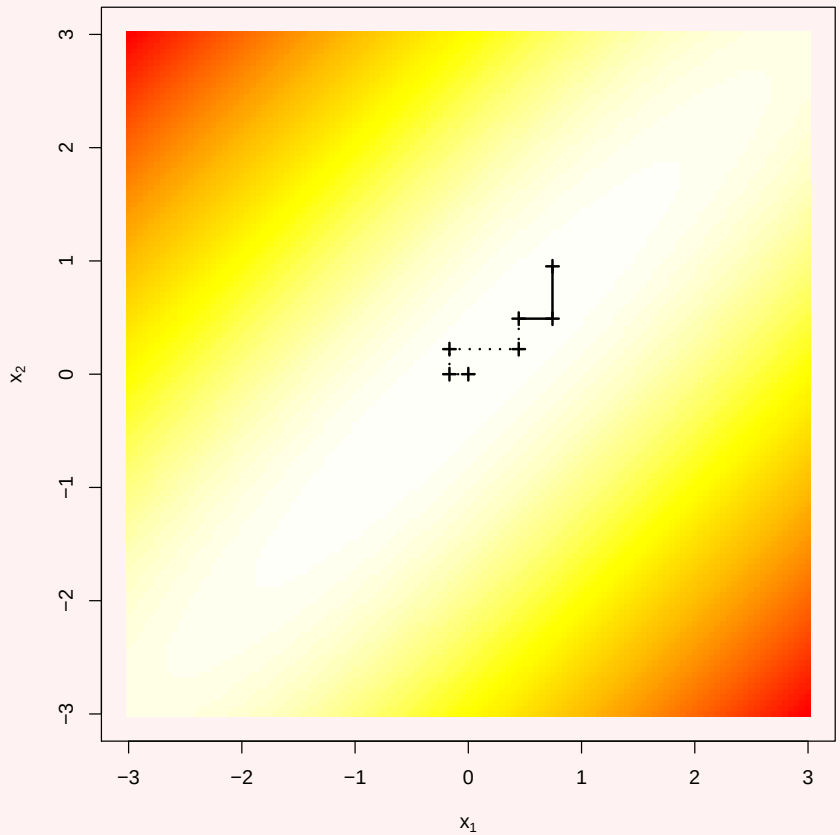
After two updates: (x_{12}, x_{22})



Gibbs Sampler: example

Example: Bivariate Normal

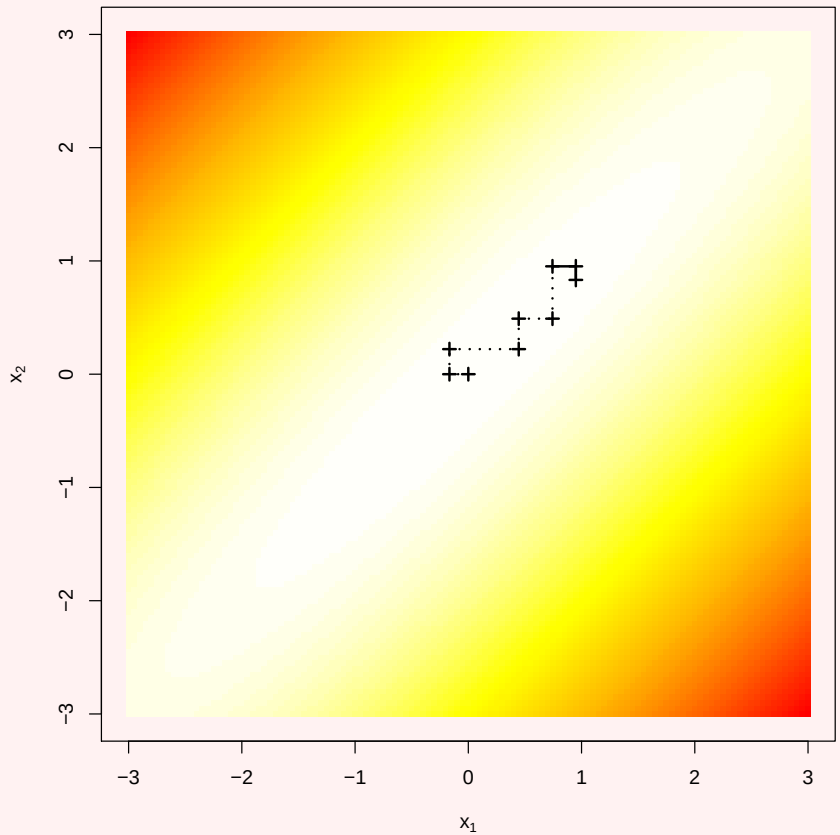
After three updates: (x_{13}, x_{23})



Gibbs Sampler: example

Example: Bivariate Normal

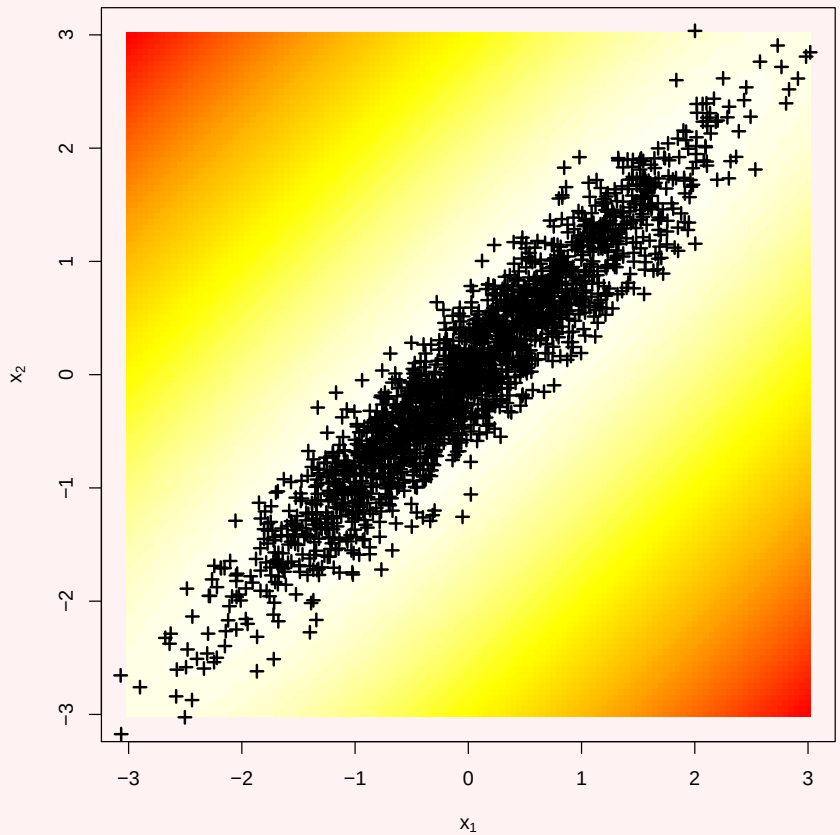
After four updates: (x_{14}, x_{24})



Gibbs Sampler: example

Example: Bivariate Normal

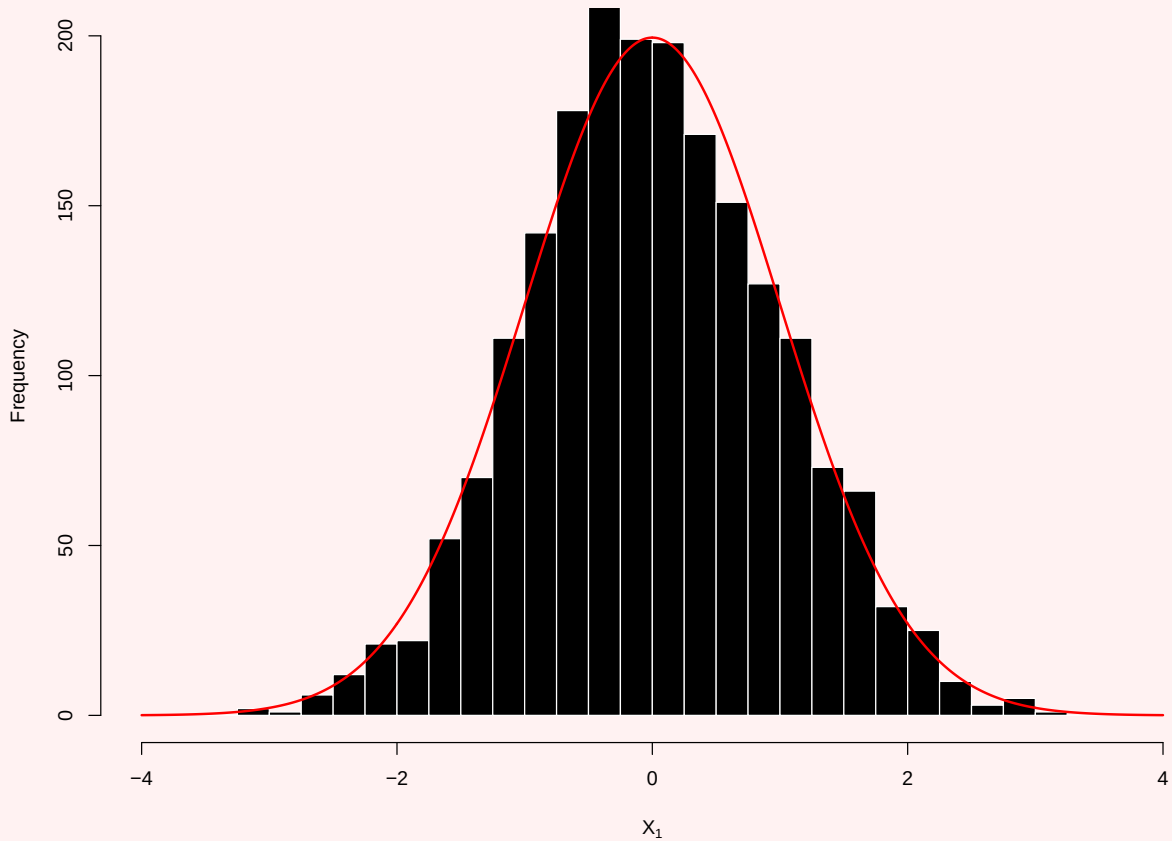
After 2000 updates: entire collected sample



Gibbs Sampler: example

Example: Bivariate Normal

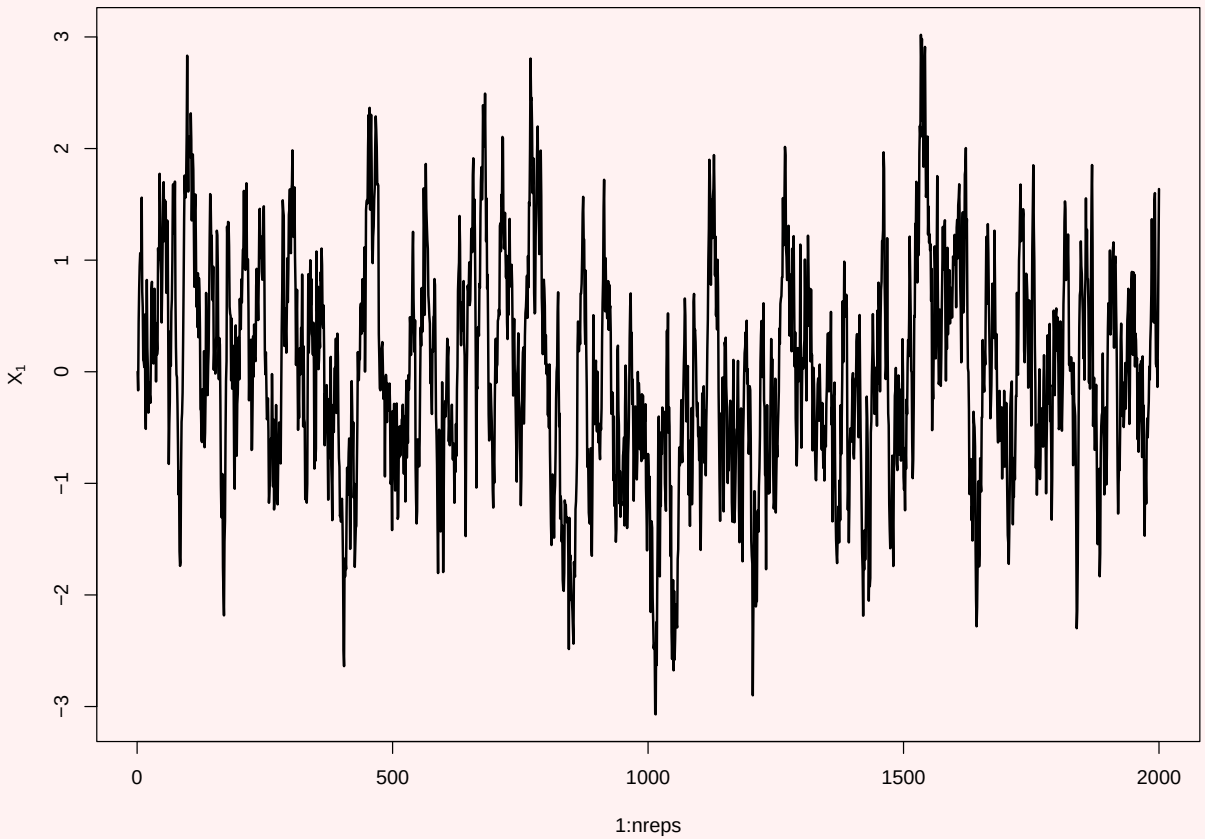
Histogram for X_1 with true marginal density (solid):



Gibbs Sampler: example

Example: Bivariate Normal

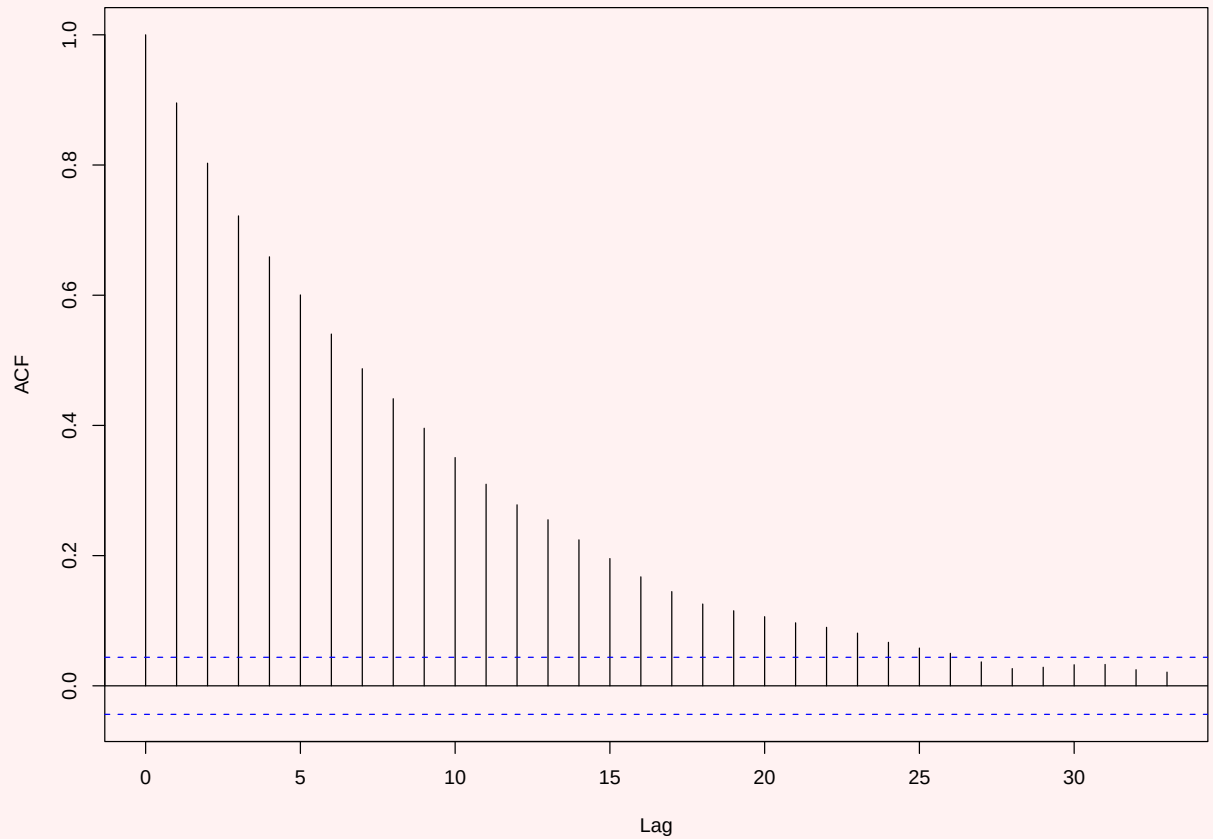
Trace for X_1:



Gibbs Sampler: example

Example: Bivariate Normal

ACF for X_1:



Gibbs Sampler: example

Example: Bivariate Normal

With $\rho = 0.95$, the Gibbs sampler still performs adequately, but the moves made are smaller, and exploring the distribution is much more difficult.

This illustrates a potential general problem with the Gibbs sampler: although it is straightforward to implement as it involves only sampling variates from univariate densities, the restriction to moves along the coordinate axes can cause problems if the variables are *highly correlated*.

Gibbs Sampler: example

Example: Weibull posterior distribution

Suppose that $Y_1, \ldots, Y_n$ are conditionally iid from the Weibull distribution with density

$$f_Y(y; \gamma, \lambda) = \gamma \lambda y^{\gamma - 1} \exp\{-\lambda y^{\gamma}\} \qquad y > 0$$

and zero otherwise, for parameters $\gamma, \lambda > 0$. We seek to perform Bayesian inference for the two unknown parameters.

The likelihood function for observed data $y_1, \ldots, y_n$ takes the form

$$\mathcal{L}_n(\gamma, \lambda) = \gamma^n \lambda^n \left(\prod_{i=1}^{n} y_i \right)^{\gamma - 1} \exp\left\{ -\lambda \sum_{i=1}^{n} y_i^{\gamma} \right\}$$

Gibbs Sampler: example

Example: Weibull posterior distribution

We assume independent $\mathit{Exponential}(0.1)$ priors for γ and λ

$$\pi_0(\gamma, \lambda) = 0.01 e^{-0.1(\gamma+\lambda)} \qquad \gamma, \lambda > 0.$$

This yields the posterior distribution up to proportionality as

$$\pi_n(\gamma, \lambda) \propto \mathcal{L}_n(\gamma, \lambda)\pi_0(\gamma, \lambda) \qquad \gamma, \lambda > 0$$

which is a non-standard distribution.

Gibbs Sampler: example

Example: Weibull posterior distribution

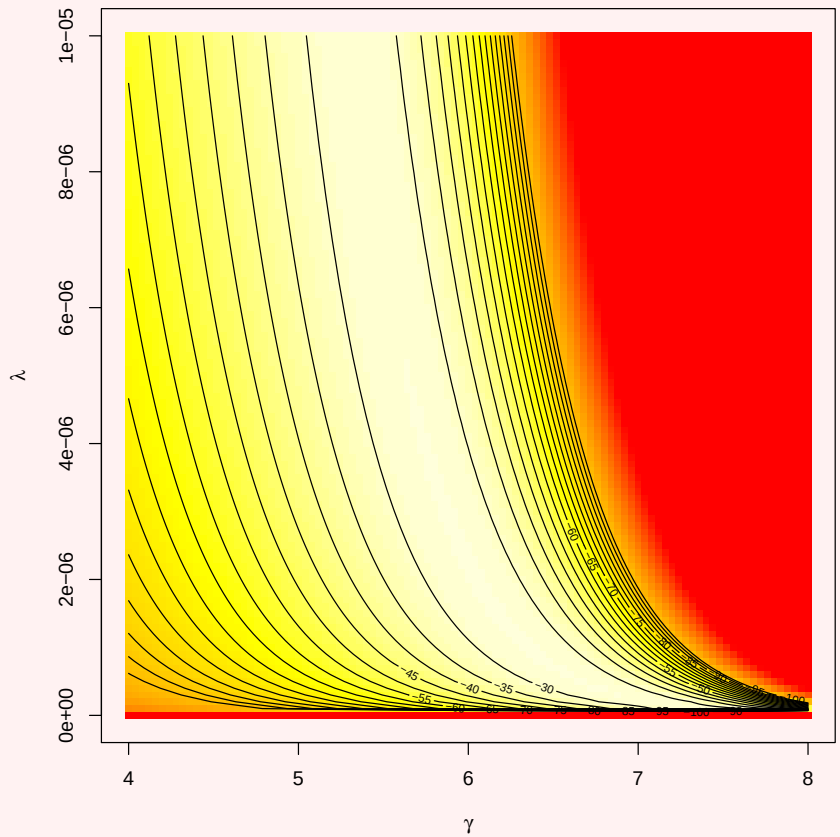
We seek to produce a sample from this joint posterior distribution for data $(n = 15)$.

10.3959	6.2281	6.5331	10.7086	7.6138
8.9423	8.8254	6.1461	7.2988	8.8081
7.5316	8.2238	8.9831	6.4174	9.7648

Gibbs Sampler: example

Example: Weibull example

Joint posterior (up to proportionality)



Gibbs Sampler: example

Example: Weibull example

In this case, neither full conditional posterior

$$\pi_n(\gamma|\lambda) \qquad\qquad \pi_n(\lambda|\gamma)$$

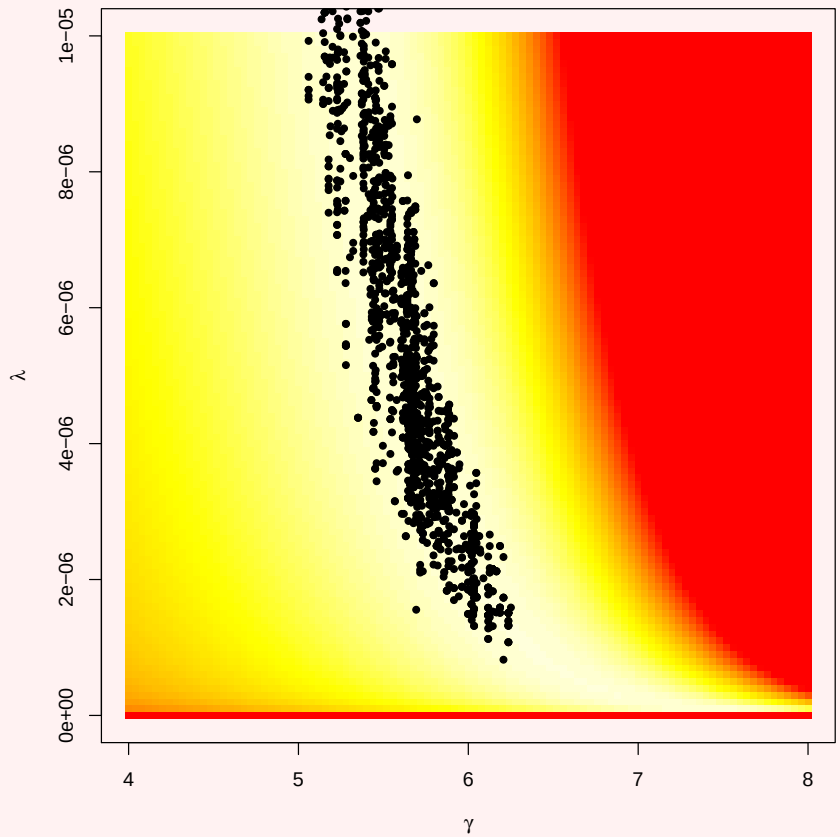
is a standard distribution, and cannot be sampled from easily.

We adopt a *Metropolis-within-Gibbs* strategy; this uses MH accept-reject steps for each parameter and its full conditional.

Specifically, as both parameters are positive, we use the reflected normal proposal distribution from the previous example with $\sigma_q = 1$ for γ and $\sigma_q = 10^{-3}$ for λ proposals.

Example: Weibull example

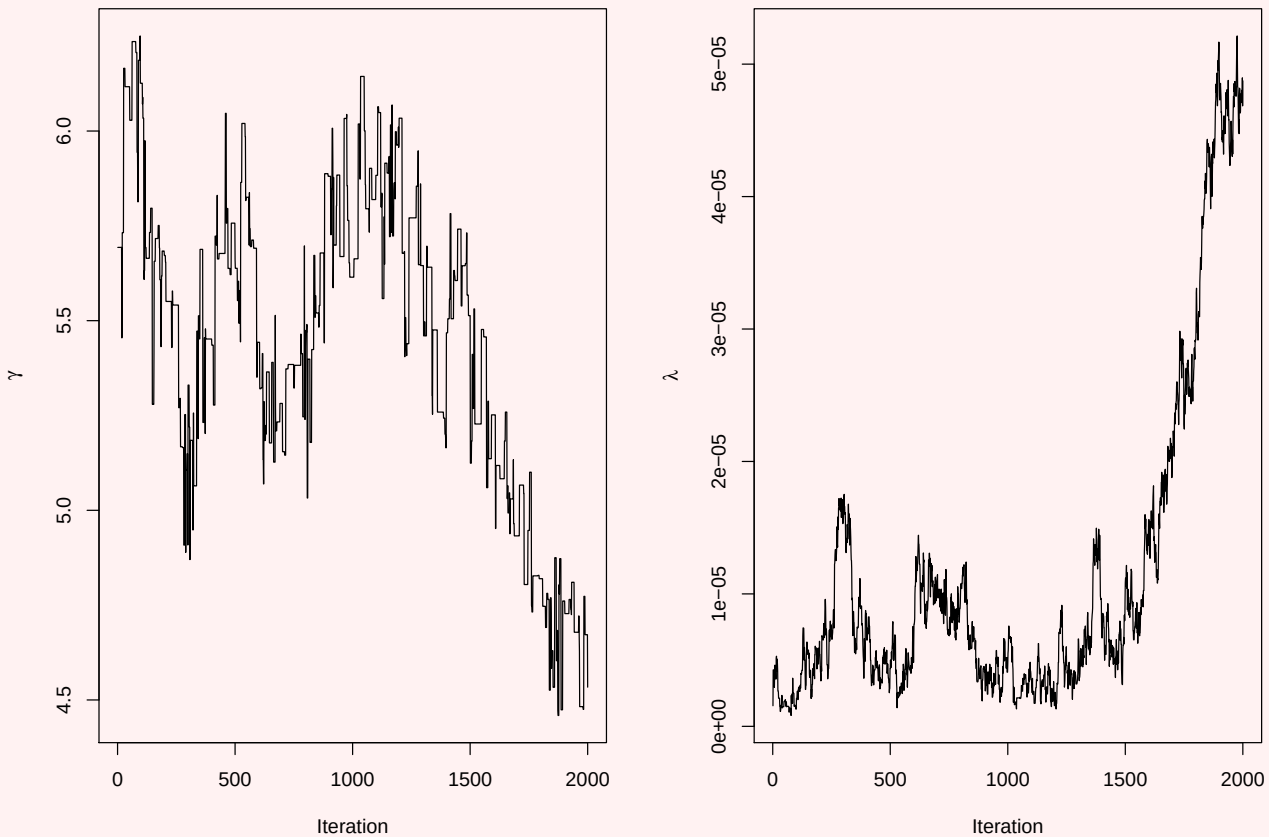
First 2000 Gibbs sampler steps.



Gibbs Sampler: example

Example: Weibull example

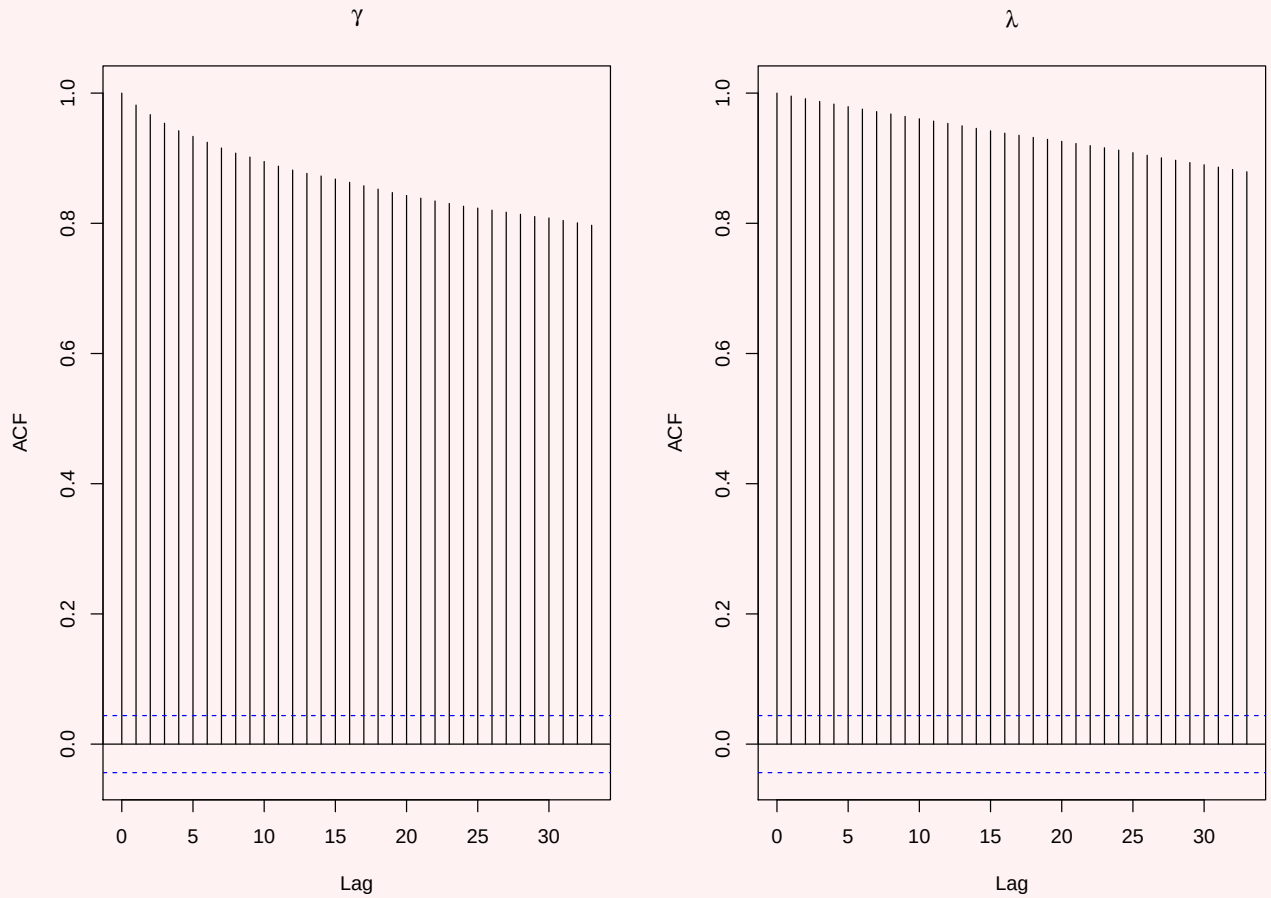
Trace plots:



Gibbs Sampler: example

Example: Weibull example

Acf:



Gibbs Sampler: example

Example: Weibull example

The posterior correlation here is approximately 0.86; this severely affects the performance of the Gibbs sampler.

A reparameterization partially solves this problem. Define ϕ by

$$\phi = \left(\frac{1}{\lambda}\right)^{1/\gamma} \qquad \therefore \qquad \lambda = \left(\frac{1}{\phi}\right)^{\gamma}$$

For this new parameterization, we must remember to include the Jacobian in the prior for the new parameters

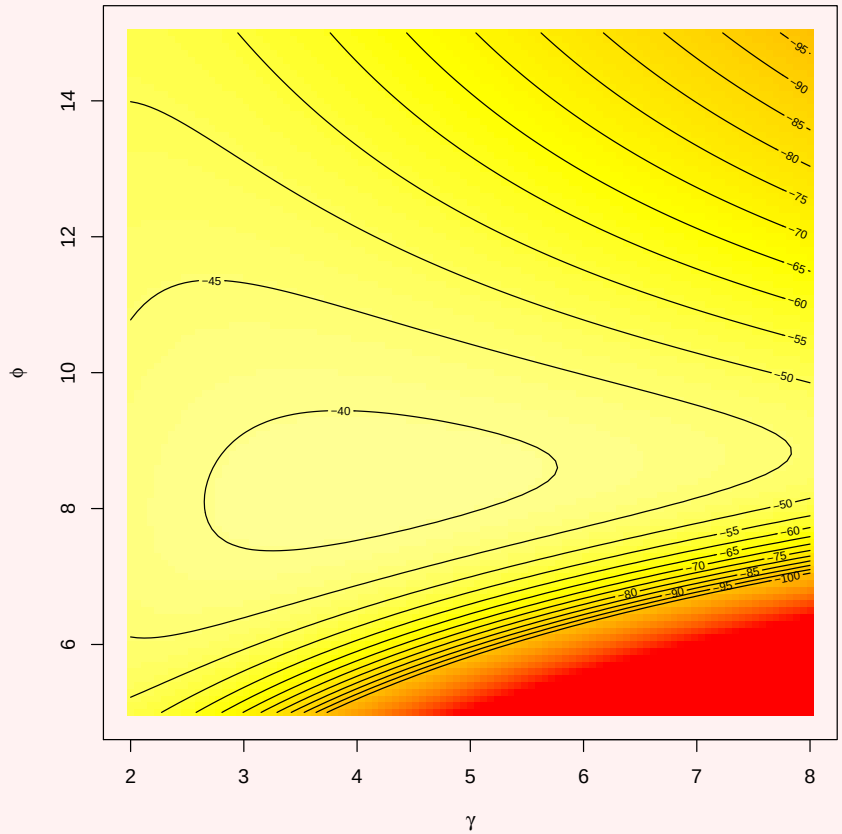
$$\pi_0(\gamma, \phi) = \pi_0(\gamma, \lambda(\gamma, \phi))|J(\gamma, \phi)|$$

We again must use Metropolis-within-Gibbs.

Gibbs Sampler: example

Example: Weibull example

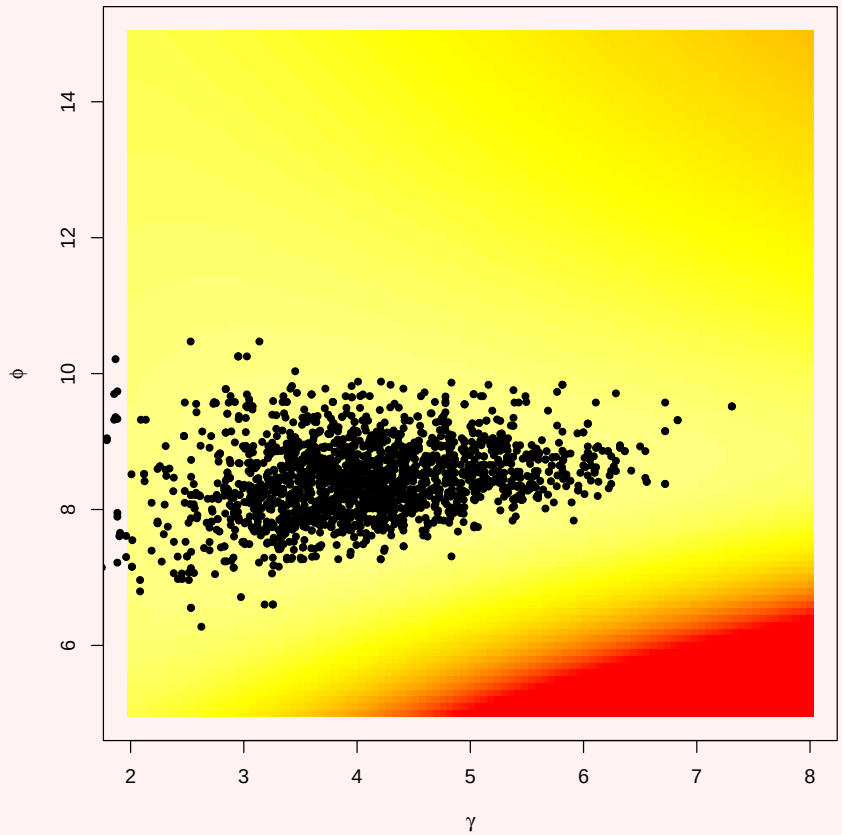
Joint posterior for new parameterization



Gibbs Sampler: example

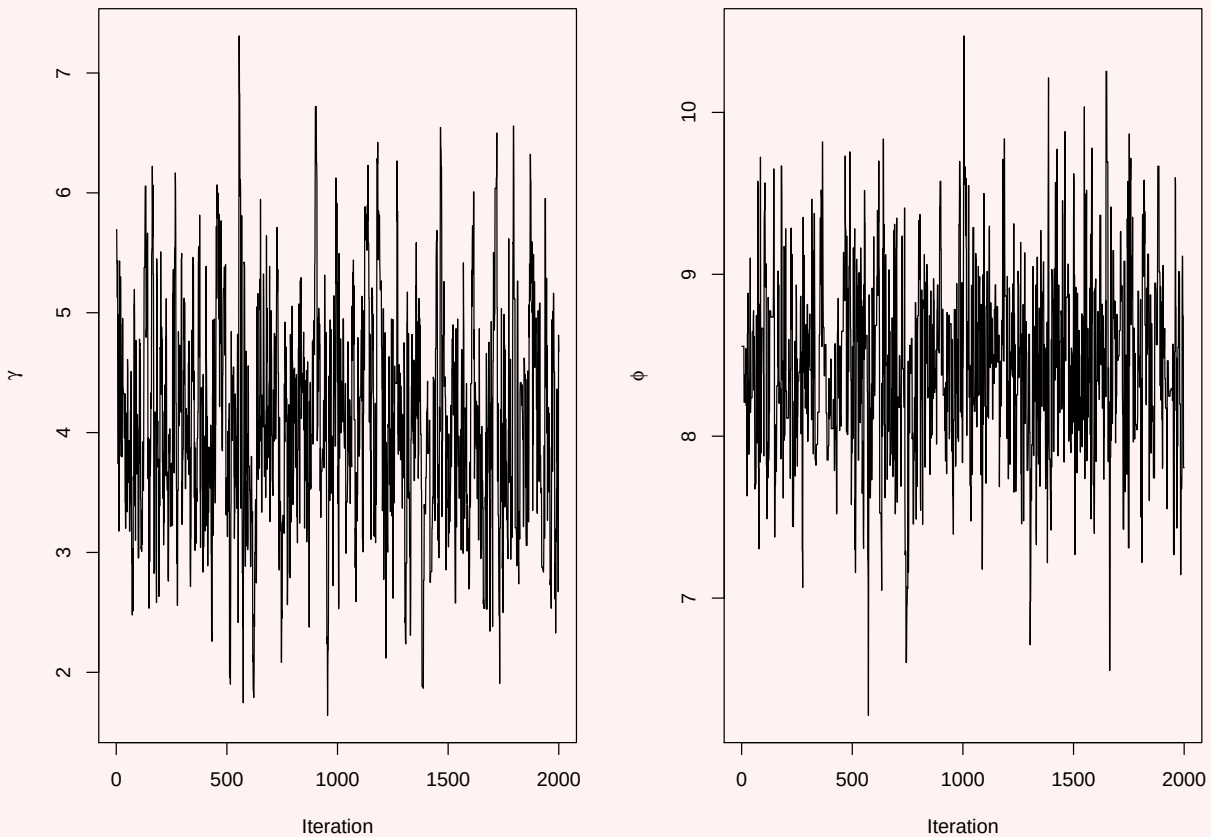
Example: Weibull example

Sample from joint posterior for new parameterization



Example: Weibull example

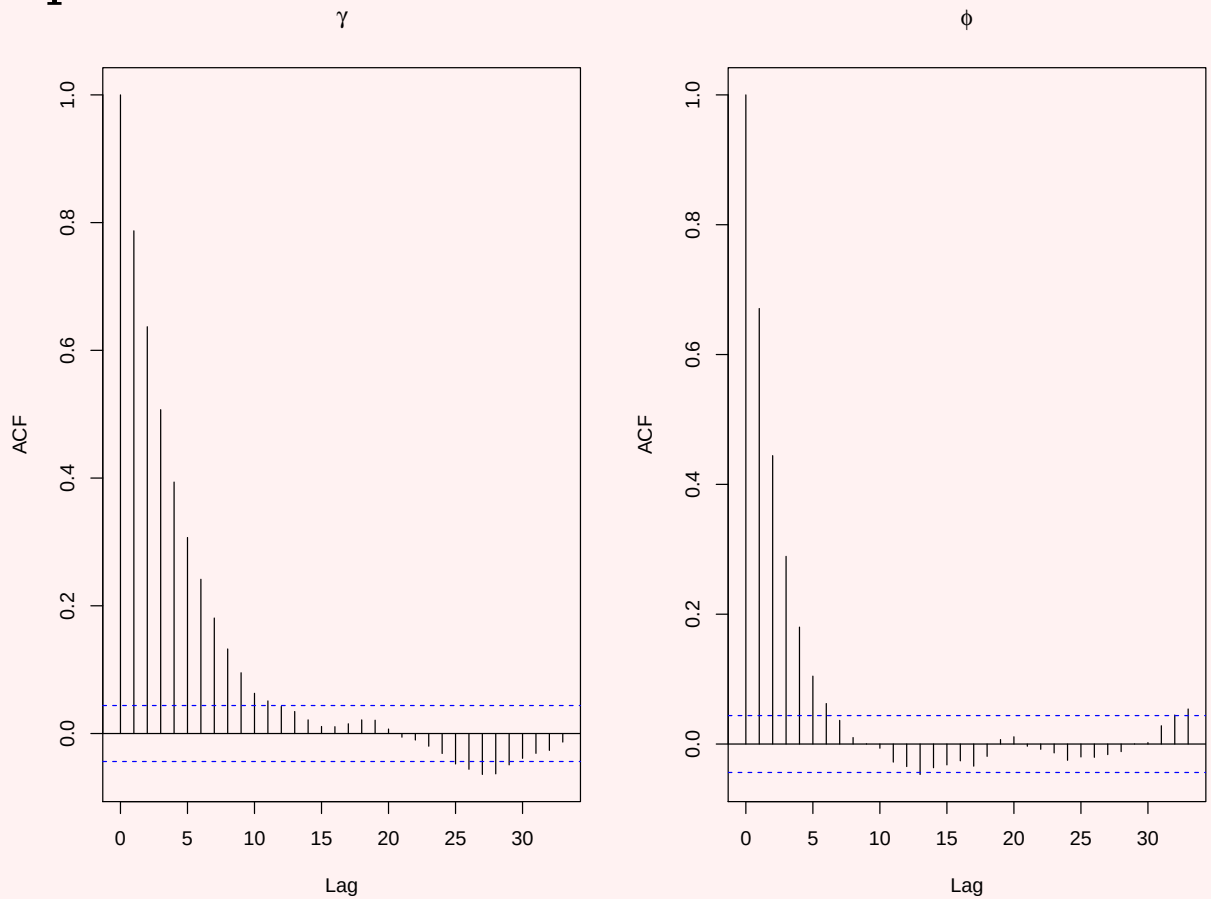
Trace plots: new parameterization



Gibbs Sampler: example

Example: Weibull example

Acf: new parameterization



Gibbs Sampler: example

Example: Weibull example

The posterior correlation here is approximately 0.25, and the Gibbs sampler can effectively traverse the parameter space.

Parameter estimates for the new parameters can be obtained from the posterior samples: the mean and 95% credible interval for each parameter is

$$\begin{aligned} \gamma &: 4.079\,(2.455, 6.038) \\ \phi &: 8.446\,(7.310, 9.593) \end{aligned}$$

It is also possible to obtain posterior summaries for other functions of the posterior parameters.

Gibbs Sampler: example

Example: Weibull example

For example, the survivor function, $S(y)$, is defined by

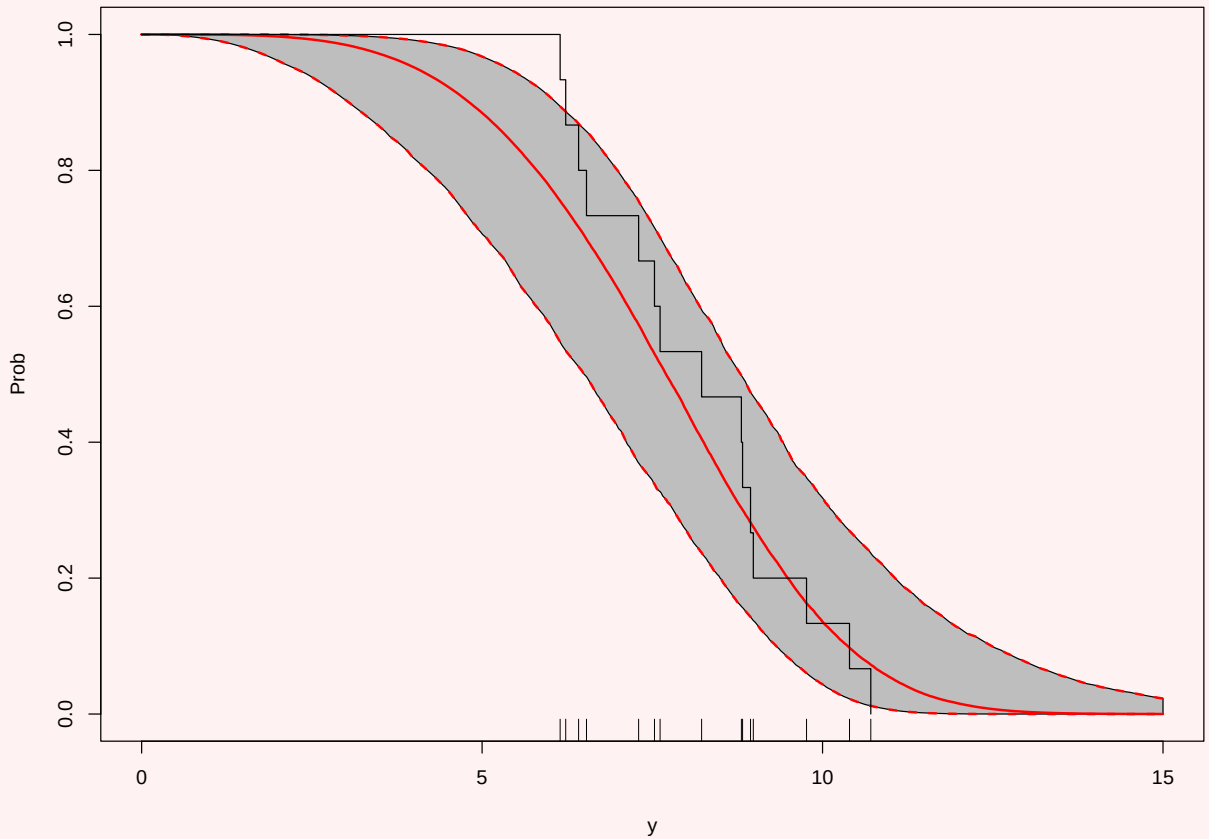
$$S(y) = \Pr[Y > y] = \exp\{-(y/\phi)^{\gamma}\}.$$

For each $y \in \mathbb{R}^{+}$, we can compute this function for each pair of generated points (γ, ϕ) obtained from the Gibbs sampler. We can then compute the pointwise credible intervals.

Gibbs Sampler: example

Example: Weibull example

Posterior survivor function: Bayes estimate and 95 % credible interval (shaded). Solid line is empirical survivor function.



Effective sample size

We seek to measure the adequacy of the collected samples for estimating parameters, in particular, we wish to assess the variance of the estimators.

For an iid sample of size N, the variance of the Monte Carlo estimator $\widehat{I}_N(g)$ is

$$\frac{\mathrm{Var}[g(X)]}{N}.$$

However, for a *dependent* sample, the variance is

$$\frac{\mathrm{Var}[g(X)]}{N_{\mathrm{eff}}}$$

where N_{eff} is the *effective sample size*.

Effective sample size

We have that, for a series of N observations from a dependent stochastic process, the effective sample size is given by

$$N_{\text{eff}} = \frac{N}{1 + 2\sum_{k=1}^{\infty} \rho(k)}$$

where $\rho(k)$ is the true lag-k autocorrelation for the Markov chain. The denominator is termed the *integrated autocorrelation time*.

The true autocorrelations are typically not known, so must be estimated from the data. Most typically this is achieved using spectral methods. The calculation is available in R from the library `coda`, in the function

`effectiveSize`.

Effective sample size

For the Weibull example, with $N = 2000$ in runs shown above

	N_{eff}	
	γ	λ
(γ, λ) parameterization	8.46	4.14
(γ, ϕ) parameterization	217.20	393.69

From this we can tell that the second MCMC run, in the (γ, ϕ) parameterization, has produced much larger effective sample sizes.

Rejection sampling for the Weibull example

Note that we could attempt to address the problem of sampling from the posterior distribution in the Weibull problem above by *rejection sampling*. In the (γ, ϕ) parameterization, we have

$$\pi_n(\gamma, \phi) \propto \mathcal{L}_n(\gamma, \phi)\pi_0(\gamma, \lambda(\gamma, \phi))|J(\gamma, \phi)|$$

where $\pi_0(.,.)$ is the product of independent $Exponential(0.01)$ priors.

Rejection sampling for the Weibull example

We use proposal function f_0 which is the product of *Gamma* densities; we choose

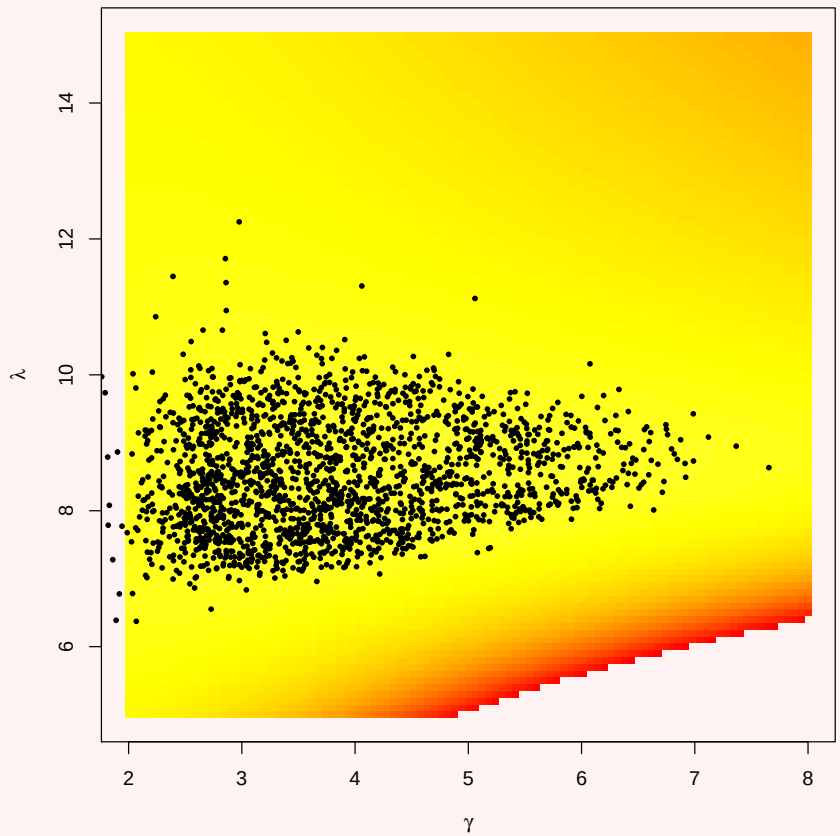
$$Gamma(2, 1/2) \qquad Gamma(4, 2)$$

for proposing (γ, ϕ), and then use numerical maximization to find the bound M.

Rejection sampling for the Weibull example

Example: Weibull example

Rejection sampling: acceptance rate is approximately 0.116.



Examples

Example: Weibull

See `knitr 8`

Example: Non-linear regression

See `knitr 9`

Metropolis-Hastings in higher dimensions

The Metropolis-Hastings algorithm can be used for probability distributions in arbitrary dimension. Note that it is always the case that, for the full conditional distributions, if

- $\mathbf{x}_1$ is a sub-vector of the entire vector $\mathbf{x}$ of variables, and
- $\mathbf{x}_{(1)}$ is $\mathbf{x}$ with the components $\mathbf{x}_1$ removed,

then

$$\pi(\mathbf{x}_1|\mathbf{x}_{(1)}) \propto \pi(\mathbf{x})$$

as the normalizing constant is $\pi(\mathbf{x}_{(1)})$, which does not depend on $\mathbf{x}_1$.

For Metropolis-within-Gibbs, knowing $\pi(\mathbf{x}_1|\mathbf{x}_{(1)})$ up to proportionality is sufficient, as the normalizing constant cancels out in the calculation of the acceptance probability.

Gibbs sampler: example

Example: Ising model

Recall the Ising model: this is a joint distribution for the collection of binary random variables $\{X_i\}$ placed on a rectangular $(N \times M)$ lattice, with joint mass function

$$\pi_{\mathbf{X}}(\mathbf{x}; \beta) = \frac{\exp\left\{\beta \sum_{i=1}^{NM} \sum_{j \neq i} \mathbb{1}_{\{x_i\}}(x_j)\right\}}{Z(\beta)}$$

where $Z(\beta)$ is the normalizing constant. The support of this mass function has

$$2^{NM}$$

elements.

Gibbs sampler: example

Example: Ising model

The full conditional distribution for the X_i is a discrete distribution on $\{0, 1\}$, with

$$\pi_i(x_i|x_{(i)}) \propto \exp\left\{\beta \sum_{j \in \partial i} \mathbb{1}_{x_j}(x_i)\right\}$$

where ∂i is the neighbourhood of i. This distribution reduces to

$$\Pr[X_i = 0|X_{(i)} = x_{(i)}] = \frac{e^{\beta n_{i0}}}{e^{\beta n_{i0}} + e^{\beta n_{i1}}}$$

where n_{i0} and n_{i1} are the numbers of neighbours of i that take the values 0 and 1 respectively. This distribution can be sampled easily as part of a Gibbs sampler.

Examples

Example: Auxiliary variable methods

See `knitr 10`

Example: Missing data problems

See `knitr 11`

Examples

Example: Multi-level models

See `knitr 12`

Example: Hierarchical linear regression

See `knitr 13`

Example: Hierarchical non-linear regression

See `knitr 14`

Multiple chain MCMC

The principal problems the MH algorithm encounters in applications relate to

- **Choice of starting values:** Poorly chosen starting values can result in slow convergence;
- **Posterior correlation:** High posterior correlation leads to inefficiency of Gibbs sampler moves, and slow convergence; this can sometimes be overcome by reparameterization.
- **Failure to explore the state space:** Algorithm can get trapped in localized regions of high probability density; this can sometimes be overcome by using *multiple chain* MCMC.

Multi-Mode Distributions

Example: Mixture of Bivariate Normals

Suppose

$$\pi(\mathbf{x}) \equiv \omega \mathcal{N}_2(\boldsymbol{\mu}_1, \Sigma_1) + (1-\omega)\mathcal{N}_2(\boldsymbol{\mu}_2, \Sigma_2)$$

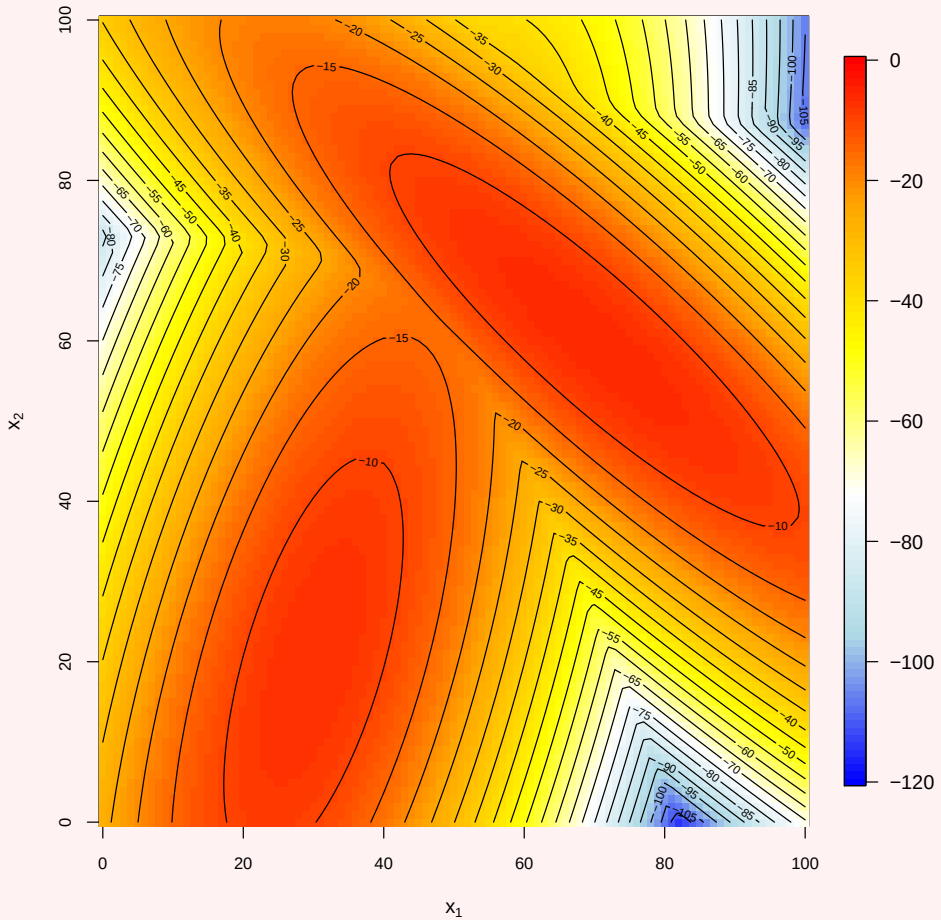
where $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are relatively well separated.

MCMC algorithms to sample from this bivariate density often encounter problems because they get stuck in the modes at $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, and cannot jump between them.

Multi-Mode Distributions

Example: Mixture of Bivariate Normals

Mixture of Bivariate Normal densities.



Dominant-Mode Distribution

Example: The Witch's Hat

In d dimensions, suppose

$$\pi(\mathbf{x}) = \omega \left(\frac{1}{\sigma}\right)^d \prod_{j=1}^{d} \phi\left(\frac{\mathbf{x}_j - \theta_j}{\sigma}\right) + (1-\omega) \prod_{j=1}^{d} \mathbb{1}_{(0,1)}(\mathbf{x}_j)$$

This density has a single mode at θ, but also a diffuse probability on the d-dimensional hypercube given by the second term.

MCMC algorithms can find it difficult to explore the "diffuse" region, because they get stuck in the mode, or *vice versa*.

Multiple chain MCMC

It is possible to use multiple Markov chain algorithms to produce variates from a target distribution π.

- **Replicate chains:** we run m chains with the *same* transition kernel P from *different starting values*, and then combine the chain as independent samples.
- **Parallel chains:** we run M chains each with its *own* transition kernel $P_1, \ldots, P_M$, such that the chain is reversible with respect to π: for $m = 1, \ldots, M$

$$\pi(x)P_m(x, y) = \pi(y)P_m(y, x)$$

and then the variates from the different chains are combined as independent samples.

Multiple chain MCMC

- **Different chains:** we run M chains each with its *own* transition kernel $P_1, \ldots, P_M$, such that each chain is reversible with respect to *its own target* π_m where, say $\pi_1 \equiv \pi$. We then allow *exchange of information* between chains to facilitate sampling from π.

In this case it is not so straightforward to guarantee that the exchange of information does produce samples from π. This does need verification, once the exchange mechanism has been defined.

Such methods are called *multiple chain* or *population* MCMC.

Multiple chain MCMC

The simplest way to think about multiple chain methods in the case of different chains is via *auxiliary variables*, and an augmentation of the state space. For $m = 1, \ldots, M$, consider

- variable X_m
- (posterior) distribution $\pi_m(x)$
- irreducible transition kernel P_m, defined so that P_m and π_m exhibit detailed balance, and P_m defines a recurrent Markov chain

where, without loss of generality, π_1 is the distribution of interest, so that $X_2, \ldots, X_m$ are auxiliary variables.

Multiple chain MCMC

For $\mathbf{X} = (X_1, \ldots, X_M)$, define

$$\pi^M(\mathbf{x}) \propto \prod_{m=1}^{M} \pi_m(x_m).$$

By construction, the marginal distributions are $\pi_1, \ldots, \pi_M$.

Consider constructing a Markov chain on the extended state space, with π^M as the target distribution. The usual rules of Metropolis-Hastings apply - we propose a candidate new value, and accept it with some acceptance probability, otherwise we remain at the current value.

The values of x_1 that we collect across iterations are (dependent) samples from the marginal distribution, π_1.

▸ Move Types

Parallel Tempering

Parallel Tempering is a multiple chain method that gives a specific construction of the distributions $\pi_1, \ldots, \pi_M$. Write density π as

$$\pi(\mathbf{x}) \propto \exp\{-H(\mathbf{x})\}$$

and define

$$\pi_m(\mathbf{x}) \propto \exp\left\{-\frac{H(\mathbf{x})}{T_m}\right\} = \{\pi(\mathbf{x})\}^{1/T_m}$$

where

$$1 = T_1 < T_2 < \cdots < T_M$$

represent a series of “temperatures". The densities $\pi_2, \ldots, \pi_M$ are *tempered* versions of π_1.

Parallel Tempering

Note that

$$\pi_m(\mathbf{x}) = \frac{\exp\left\{-\frac{H(\mathbf{x})}{T_m}\right\}}{Z(T_m)}$$

where

$$Z(T_m) = \int \exp\left\{-\frac{H(\mathbf{x})}{T_m}\right\}\, d\mathbf{x}$$

may not be available analytically.

This integral needs to be finite for π_m to be a proper distribution.

Parallel Tempering

Example: Two component Normal mixture

For the two component mixture

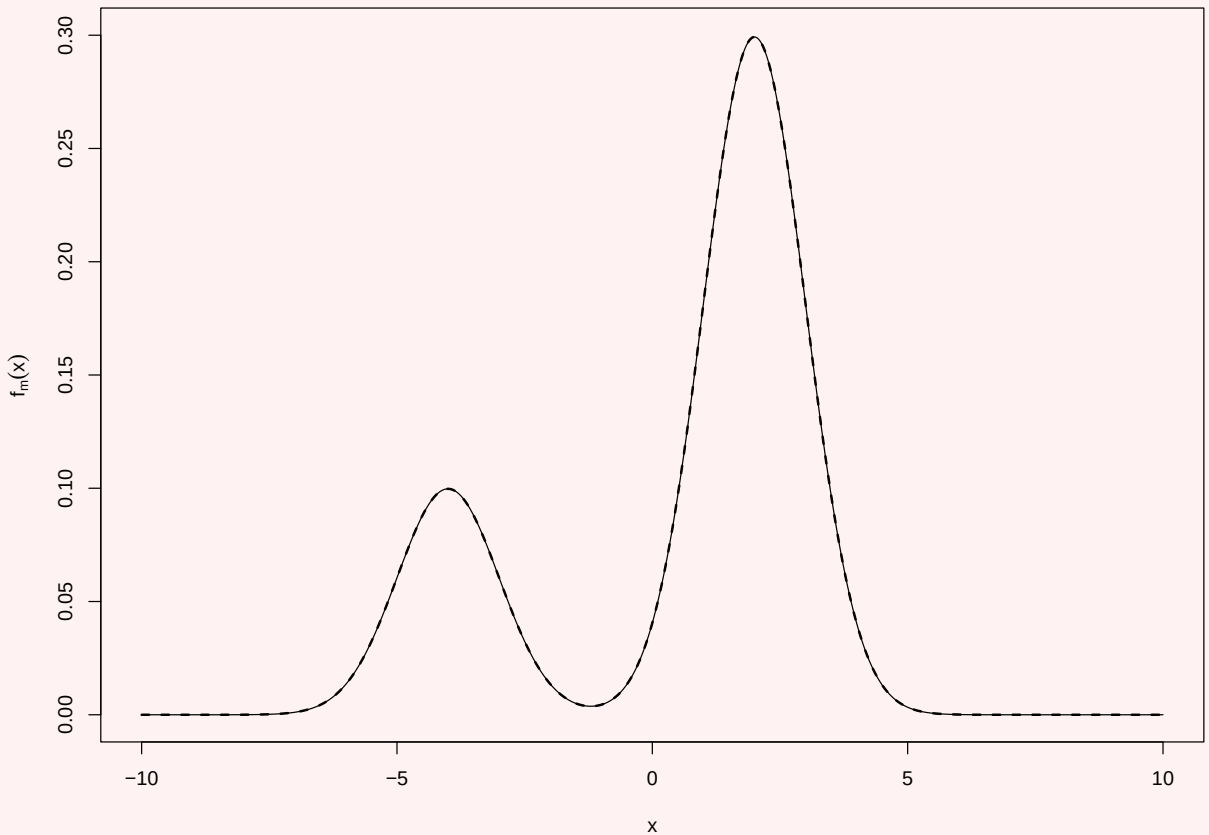
$$\pi(x) = \frac{1}{4}\phi(x+4) + \frac{3}{4}\phi(x-2)$$

the tempered distributions have modes that become increasingly flat.

Parallel Tempering

Example: Two component Normal mixture

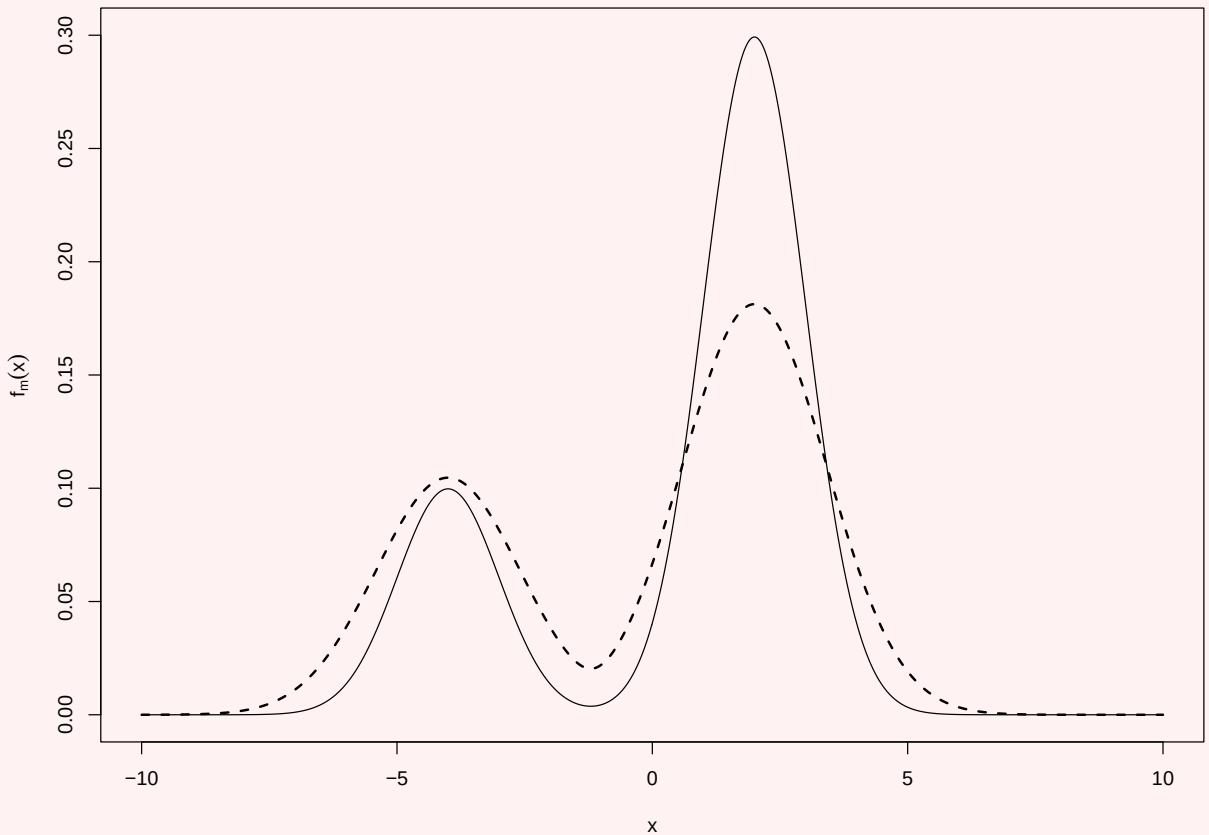
$T_1 = 1$:



Parallel Tempering

Example: Two component Normal mixture

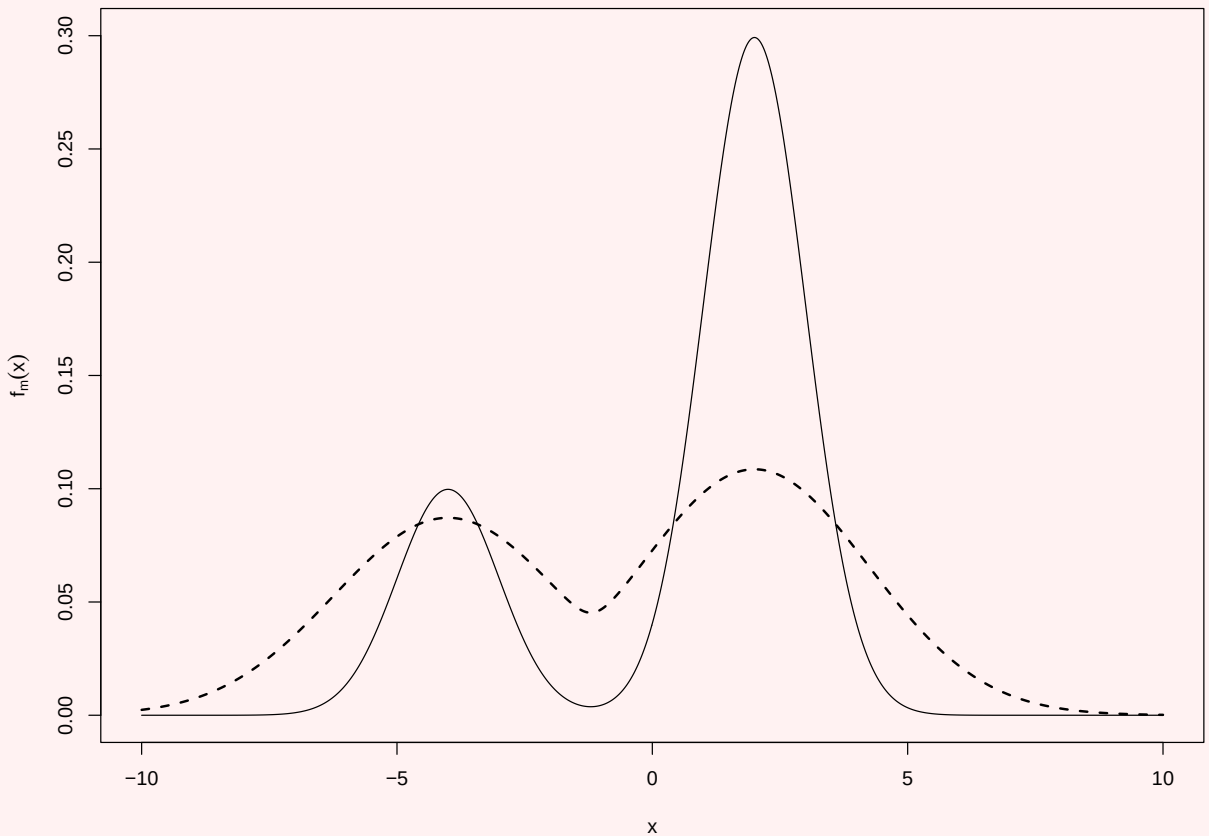
$T_2 = 2$:



Parallel Tempering

Example: Two component Normal mixture

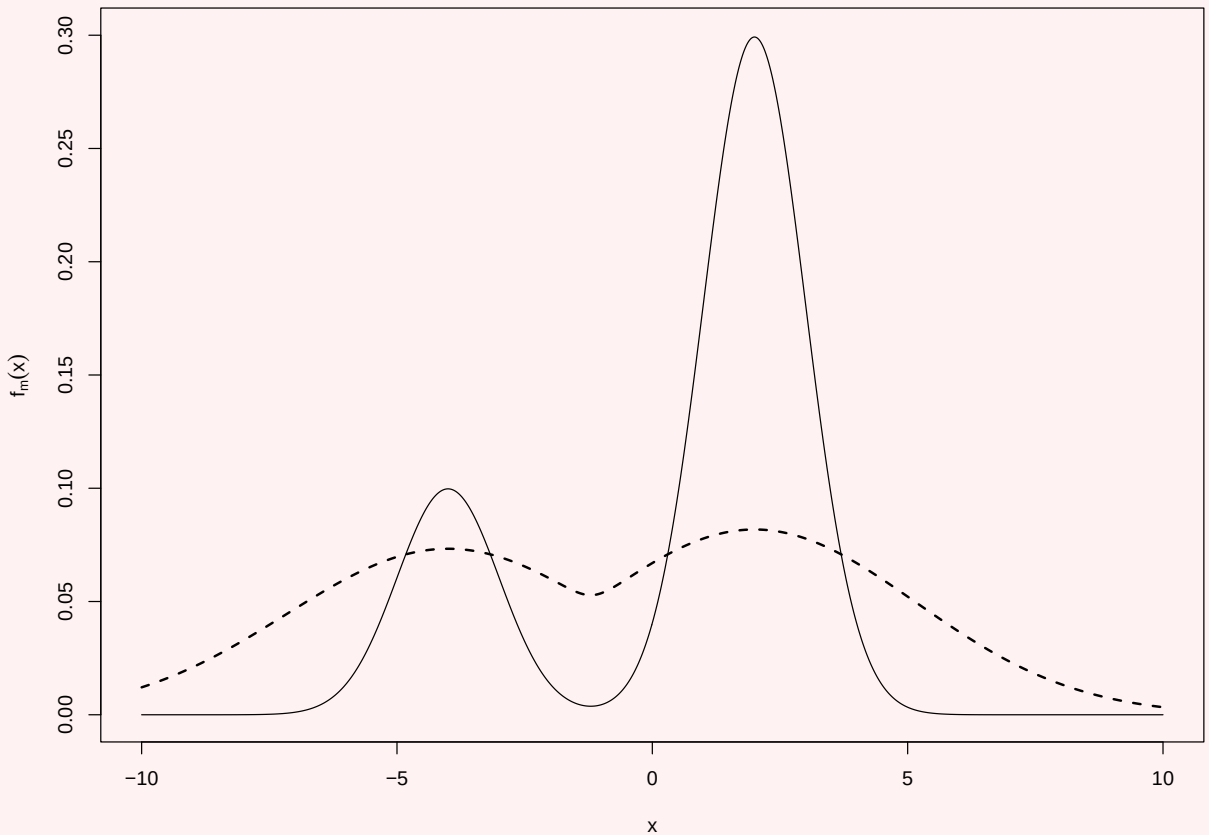
$T_3 = 5$:



Parallel Tempering

Example: Two component Normal mixture

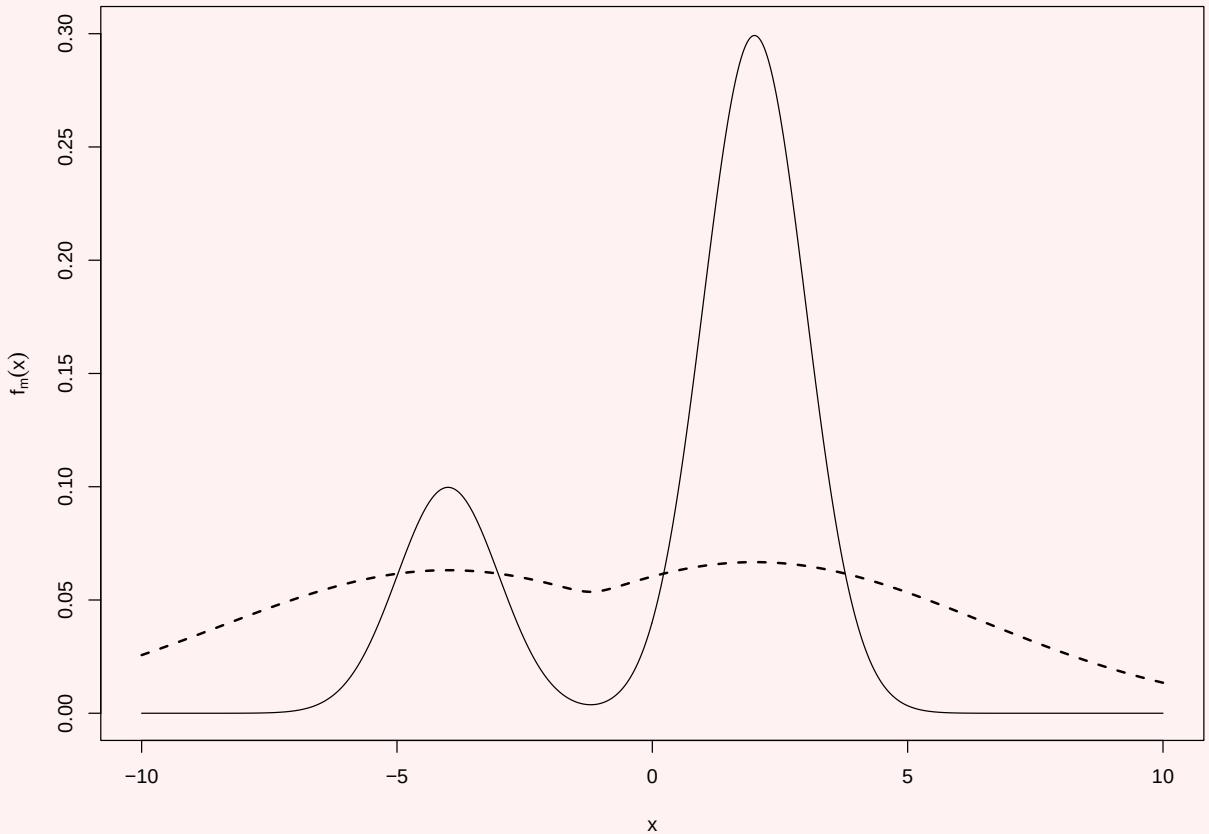
$T_4 = 10$:



Parallel Tempering

Example: Two component Normal mixture

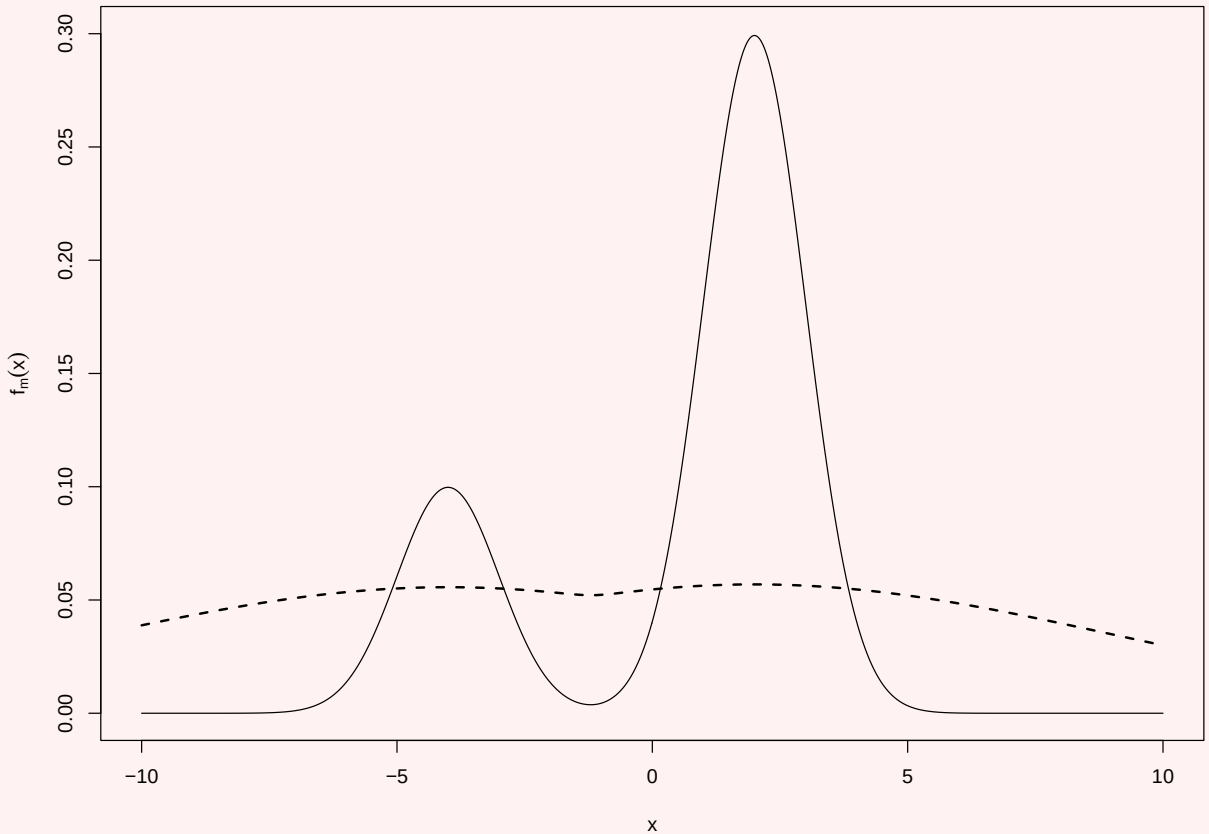
$T_5 = 20$:



Parallel Tempering

Example: Two component Normal mixture

$T_6 = 50$:



Parallel Tempering

When T_m is large

$$\pi_m(\mathbf{x}) \propto \exp\left\{-\frac{H(\mathbf{x})}{T_m}\right\}$$

is relatively “flat", that is, changes in x do not lead to large changes in $\pi_m(\mathbf{x})$.

Parallel tempering utilizes *within* chain and *between* chain updates. In the classical version, each iteration consists of

1. **Within:** an update for *each* chain using the MH kernels $P_1, \ldots, P_M$, then
2. **Between:** an update that attempts to swap x_m (from the chain for π_m at temperature T_m) with one of its temperature neighbours.

Parallel Tempering

Define the between-chain exchange probabilities

$$
\begin{aligned}
q_B(l,m) &= \Pr[\text{Swap chain } l \text{ with chain } m] \\
&= \begin{cases} \dfrac{1}{2} & 2 \leqslant l \leqslant M, m = l-1, l+1 \\ 1 & l = 1 \text{ and } m = 2, \text{ or } l = 1 \text{ and } m = 2 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

Then the between-chain exchange that proposes to swap x_l with x_m is accepted with probability

$$
\alpha = \min\left\{1, \frac{\pi_l(\mathrm{x}_m)\pi_m(\mathrm{x}_l)q_B(m,l)}{\pi_l(\mathrm{x}_l)\pi_m(\mathrm{x}_m)q_B(l,m)}\right\}
$$

Parallel Tempering

For the tempered distributions

$$\frac{\pi_l(x_m)\pi_m(x_l)}{\pi_l(x_l)\pi_m(x_m)} = \exp\left\{-\frac{H(x_m)}{T_l} - \frac{H(x_l)}{T_m} + \frac{H(x_l)}{T_l} + \frac{H(x_m)}{T_m}\right\}$$

or equivalently

$$\exp\left\{(H(x_l) - H(x_m))\left(\frac{1}{T_l} - \frac{1}{T_m}\right)\right\}$$

Temperatures $T_2, \ldots, T_M$ should be chosen such that moves are accepted at neither too high nor too low a rate.

The samples collected for x_1 across iterations represent a sample from the marginal $\pi_1 \equiv \pi$, the target distribution.

Parallel Tempering

Tempered moves can be proposed in an “up-down” fashion. An update for x_1 (in the target chain for $\pi_1 \equiv \pi$) can be proposed as follows.

For the “up phase":

Propose y_1 from kernel $q_1(x_1, y)$ in chain 1
Propose y_2 from kernel $q_2(y_1, y)$ in chain 2
Propose y_3 from kernel $q_3(y_2, y)$ in chain 3
...
Propose y_{M-1} from kernel $q_{M-1}(y_{M-2}, y)$ in chain $M-1$
Propose y_M from kernel $q_M(y_{M-1}, y)$ in chain M

Parallel Tempering

For the “down phase":

Propose z_M from kernel $q_M(y_M, z)$ in chain M

Propose z_{M-1} from kernel $q_{M-1}(z_M, z)$ in chain $M-1$

Propose z_{M-2} from kernel $q_{M-2}(z_{M-1}, z)$ in chain $M-2$

...

Propose z_2 from kernel $q_2(z_3, z)$ in chain 2

Propose z_1 from kernel $q_1(z_2, z)$ in chain 1

This up and down sequence has proposed a move in the π_1 chain $x_1 \longrightarrow z_1$ via the intermediate steps through the other tempered chains.

Parallel Tempering

The acceptance probability for the move is the minimum of 1 and

$$\frac{\pi_1(z_1)}{\pi_1(x_1)} \frac{q_{\overline{U}}(z_1, \ldots, z_M, y_M) q_{\overline{D}}(y_M, \ldots, y_1, x_1)}{q_{\underline{U}}(x_1, y_1, \ldots, y_M) q_{\underline{D}}(y_M, z_M, \ldots, z_1)}$$

where

$$q_{\overline{U}}(z_1, \ldots, z_M, y_M) = q_1(z_1, z_2) q_2(z_2, z_3) \cdots q_M(z_M, y_M)$$

$$q_{\overline{D}}(y_M, \ldots, y_1, x_1) = q_M(y_M, y_{M-1}) q_{M-1}(y_{M-1}, y_{M-2}) \cdots q_1(y_1, x_1)$$

$$q_{\underline{U}}(x_1, y_1, \ldots, y_M) = q_1(x_1, y_1) q_2(y_1, y_2) \cdots q_M(y_{M-1}, y_M)$$

$$q_{\underline{D}}(y_M, z_M, \ldots, z_1) = q_M(y_M, z_M) q_{M-1}(z_M, z_{M-1}) \cdots q_1(z_2, z_1)$$

If all of the proposals are symmetric $q_m(x, y) = q_m(y, x)$, then the ratio simplifies to

$$\frac{\pi_1(z_1)}{\pi_1(x_1)}$$

Evolutionary Monte Carlo

Evolutionary Monte Carlo combines multi-chain moves with tempered distributions in order to make exploration of the state space more effective.

For a "crossover" move, two chains exchange their values. For two "parent" chains that participate in the crossover select the first parent to be chain l with probability p_l proportional to

$$\exp\{-\beta H(\mathbf{x}_l)\}$$

where $\beta > 0$, and the second parent uniformly from the remaining chains.

Evolutionary Monte Carlo

The acceptance probability is computed by considering the various probabilities and densities in the proposal steps. For the crossover move

- Pick parent 1, $\mathbf{x}_l$, with probability p_l
- Pick parent 2, $\mathbf{x}_m$, uniformly, with probability $1/(M-1)$
- Pick the location of the crossover j, uniformly, with probability $1/(d-1)$
- Perform the crossover to obtain $\mathbf{x}_l^{\text{new}}$ and $\mathbf{x}_m^{\text{new}}$.

This yields the *numerator* in the Hastings ratio

$$p_m \times \frac{1}{M-1} \times \frac{1}{d-1} \times \pi_l\left(\mathbf{x}_l^{\text{new}}\right) \times \pi_m\left(\mathbf{x}_m^{\text{new}}\right)$$

Considering the reverse move yields the denominator.

Simulated Tempering

In *simulated tempering*, the temperatures are also treated as quantities that are updated during the MCMC updates. Suppose

$$\pi_m(\mathbf{x}, \tau_m) \propto \frac{\exp\{-H(\mathbf{x})/\tau_m\}}{Z(\tau_m)}$$

where $Z(\tau_m)$ is the normalizing constant for the mth tempered (conditional) distribution. A Metropolis-Hastings algorithm is constructed on the augmented state space

$$(\mathbf{x}_1, \tau_1), (\mathbf{x}_2, \tau_2), \ldots, (\mathbf{x}_M, \tau_M)$$

and the sampled collected from chain 1 are samples from the target $\pi_1 \equiv \pi$.

Simulated Tempering

For a *mutation* move, chain m is selected uniformly from collection of chains, and then the pair (x_m, τ_m) is updated using a transition proposal

$$q_m\big((\mathbf{x}_m, \tau_m), (\mathbf{x}_m^{\text{new}}, \tau_m^{\text{new}})\big)$$

with the usual acceptance probability.

For the *exchange* move between l and m, the proposed exchange is accepted with probability

$$\alpha = \min\left\{1, \frac{\pi_l(\mathbf{x}_m)\pi_m(\mathbf{x}_l)}{\pi_l(\mathbf{x}_l)\pi_m(\mathbf{x}_m)}\right\}$$

where the probabilities of choosing pairs of chains l and m to attempt the exchange are equal.

Simulated Tempering

A Gibbs sampler update, involving proposals for x, and then for τ_l, can also be carried out. Specifically, we may propose an exchange for τ_l whilst holding x_l constant.

If l is selected uniformly from $\{1, \ldots, M\}$, then another index m is selected with probability q_{lm}, and a temperature swap $\tau_l \longleftrightarrow \tau_m$ is proposed. In this case, the acceptance probability is

$$\alpha = \min\left\{1, \frac{\pi_m(\mathbf{x}_l)}{\pi_l(\mathbf{x}_l)} \frac{q_{ml}}{q_{lm}}\right\}$$

Here

$$\frac{\pi_m(\mathbf{x}_l)}{\pi_l(\mathbf{x}_l)} = \frac{Z(\tau_m)}{Z(\tau_l)} \exp\left\{-H(\mathbf{x}_l)\left(\frac{1}{\tau_m} - \frac{1}{\tau_l}\right)\right\}$$

In this formula, the normalizing constant ratio

$$\frac{Z(\tau_m)}{Z(\tau_l)}$$

must be computed. Here

$$Z(\tau) = \int \exp\{-H(\mathbf{x})/\tau\}\, d\mathbf{x}$$

which, in the multivariate case is a d-dimensional integral that needs to be finite for each possible τ.

In some cases, analytical expressions can be obtained, but more generally an estimate $\widehat{Z}(\tau)$ must be used.

Simulated Annealing

Simulated annealing is a related optimization technique that utilizes *MCMC* and a temperature ladder. Suppose that the function $H(x)$ is to be minimized, let

$$\pi_T(\mathbf{x}) \propto \exp\{-H(\mathbf{x})/T\}$$

and consider the sequence of *decreasing* temperatures

$$T_1 > T_2 > \cdots$$

Simulated annealing proceeds by running MCMC at the highest temperature T_1 for a fixed number N_1 of iterations, then changing temperatures to T_2 and running MCMC for N_2 iterations, and so on.

Simulated Annealing

As $i \longrightarrow \infty$ with $T_i \longrightarrow 0$, the MCMC samples become focussed at the mode(s) of $\pi(x)$, that is, the minima of $H(x)$. Theoretically, the logarithmic rate, where

$$T_i = \frac{K}{\log(M_i)} \qquad \text{with} \qquad M_i = \sum_{j=1}^{i} N_i$$

ensures convergence to the global maximum of π.

This rate is extremely slow, and typically geometric or linear cooling schedules are used; they generally have reasonable performance.

Simulated annealing is an option in the `optim` function in `R`.

Examples

Example: Tempering methods

See `knitr 15`

Sequential Monte Carlo

Consider a sequence of probability distributions

$$\pi_1, \pi_2, \ldots,$$

defined on a sequence of state spaces $\mathbb{X}_1, \mathbb{X}_2, \ldots$ where, for each n, we may, for some $\mathbb{X}$, either have

$$\mathbb{X}_n = \mathbb{X} \times \mathbb{X} \cdots \times \mathbb{X} \equiv \mathbb{X}^n$$

that is, dimension *increasing*, or $\mathbb{X}_n \equiv \mathbb{X}$ (i.e. dimension *fixed*). Let

$$\pi_n(\mathbf{x}_{1:n}) = \frac{g_n(\mathbf{x}_{1:n})}{Z_n}$$

where $x_{1:n} = (x_1, \ldots, x_n)$, g_n is an integrable non-negative function, and Z_n is a normalizing constant.

Sequential Monte Carlo: Two cases

1. **Dynamic problems:** where data are collected sequentially though time, as in *state-space models*

$$\begin{aligned} y_t &= H(\mathbf{x}_t;\theta) + \epsilon_t \\ \mathbf{x}_t &= \mu + \phi(\mathbf{x}_{t-1} - \mu) + \varepsilon_t \end{aligned}$$

   where the observed data $\{y_1, \ldots, y_n\}$ are related to the unobserved *states* $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

   Here the parameters $\theta, \mu, \phi, \sigma^2_\epsilon, \sigma^2_\varepsilon$ are also of interest.

   Here π_t is the posterior distribution derived from the data $\{y_1, \ldots, y_t\}$.

Sequential Monte Carlo: Two cases

2. **Static problems:** where data are collected together, but used sequentially.

   For example, in a Bayesian inference problem, we may have

   - π_1 being the posterior computed from y_1
   - π_2 being the posterior computed from y_1, y_2

   and so on.

In both cases, we may wish to perform a Monte Carlo calculation with respect to π_n, such as an expectation of the form

$$\mathbb{E}_{\pi_n}[h(X_{1:n})] = \int h(x_{1:n})\pi_n(x_{1:n})\, dx_{1:n}$$

Recall: **importance sampling** uses the identity

$$\int h(x)f(x)\,dx = \int h(x)f(x)\frac{f_0(x)}{f_0(x)}\,dx = \int \left\{ \frac{h(x)f(x)}{f_0(x)} \right\} f_0(x)\,dx$$

which demonstrates that

$$\mathbb{E}_f[h(X)] = \mathbb{E}_{f_0}\left[\frac{h(X)f(X)}{f_0(X)} \right]$$

so that an estimator of the LHS is

$$\widehat{I}_N^{(f_0)}(h) = \frac{1}{N}\sum_{i=1}^{N} \frac{h(X_i)f(X_i)}{f_0(X_i)}$$

where $X_1, \ldots, X_N \sim f_0(.)$.

Sequential Monte Carlo

Let

$$\pi_n(x_{1:n}) = \frac{g_n(x_{1:n})}{Z_n} \equiv \frac{g_n(x_{1:n})}{\int g(x_{1:n})\, dx_{1:n}}$$

$$= \frac{w_n(x_{1:n}) p_n(x_{1:n})}{\int w_n(x_{1:n}) p_n(x_{1:n})\, dx_{1:n}}$$

where p_n is a chosen importance distribution, and

$$w_n(x_{1:n}) = \frac{g_n(x_{1:n})}{p_n(x_{1:n})}$$

is the importance weight.

Sequential Monte Carlo

If an i.i.d. sample of size N $x_{1:n}^{(1)}, \ldots, x_{1:n}^{(N)}$ can be obtained from p_n, then the *empirical pmf*

$$\widehat{p}_n(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{x_{1:n}^{(i)}}(\mathbf{x})$$

can be used to construct an estimate of π_n:

$$\widehat{\pi}_n(\mathbf{x}) = \sum_{i=1}^{N} W_n^{(i)} \delta_{x_{1:n}^{(i)}}(\mathbf{x})$$

where

$$W_n^{(i)} = \frac{w_n(x_{1:n}^{(i)})}{N\widehat{Z}_n} \qquad \widehat{Z}_n = \frac{1}{N} \sum_{i=1}^{N} w_n(x_{1:n}^{(i)})$$

Sequential Monte Carlo

In a *static* Bayesian calculation, the full posterior is

$$\pi_n(\theta) = \frac{\mathcal{L}_n(\theta)\pi_0(\theta)}{\int \mathcal{L}_n(t)\pi_0(t)\,dt}$$

so we could consider setting

$$p_n(\theta) = \pi_0(\theta)$$

for all n. The problem with this is that $\mathcal{L}_n(\theta)$ is likely to be very peaked compared to $\pi_0(\theta)$ once n gets moderately large.

Sequential Importance Sampling

Using the chain-rule factorization

$$\pi_n(x_{1:n}) = \pi_1(x_1) \prod_{j=2}^{n} \widetilde{\pi}_j(x_j | x_{1:(j-1)})$$

where

$$\widetilde{\pi}_j(x_j | x_{1:(j-1)}) = \frac{\pi_j(x_{1:j})}{\pi_{j-1}(x_{1:(j-1)})}$$

is the conditional distribution of element x_j, given

$$x_{1:(j-1)} = (x_1, \ldots, x_{j-1}).$$

Sequential Importance Sampling

We then can construct a *Sequential Importance Sampling* (SIS) density

$$p_n(x_{1:n}) = p_1(x_1) \prod_{j=2}^{n} \widetilde{p}_j(x_j | x_{1:(j-1)})$$

where $\widetilde{p}_j(x_j | x_{1:(j-1)})$ is again a conditional distribution.

Sequential Importance Sampling

The importance sampling weight then takes the form

$$w_n(x_{1:n}) = \frac{\pi_n(x_{1:n})}{p_n(x_{1:n})} = \frac{\pi_1(x_1)}{p_1(x_1)} \prod_{j=2}^{n} \frac{\widetilde{\pi}_j(x_j|x_{1:(j-1)})}{\widetilde{p}_j(x_j|x_{1:(j-1)})}$$

so therefore

$$w_n(x_{1:n}) = w_{n-1}(x_{1:(n-1)}) \frac{\widetilde{\pi}_n(x_n|x_{1:(n-1)})}{\widetilde{p}_n(x_n|x_{1:(n-1)})}$$

$$w_{n-1}(x_{1:(n-1)}) = w_{n-2}(x_{1:(n-2)}) \frac{\widetilde{\pi}_{n-1}(x_{n-1}|x_{1:(n-2)})}{\widetilde{p}_{n-1}(x_{n-1}|x_{1:(n-2)})}$$

and so on, so there is a *recursive* weight calculation,

$$w_j(x_{1:j}) = w_{j-1}(x_{1:(j-1)}) \frac{\widetilde{\pi}_j(x_j|x_{1:(j-1)})}{\widetilde{p}_j(x_j|x_{1:(j-1)})}.$$

Sequential Importance Sampling

However: in the numerator of the weights, we have

$$\widetilde{\pi}_j\left(x_j \middle| x_{1:(j-1)}\right) = \frac{\pi_j\left(x_{1:j}\right)}{\pi_{j-1}\left(x_{1:(j-1)}\right)}$$

but to compute this we need the marginal distributions π_j and π_{j-1}, which may not be available analytically.

Sequential Importance Sampling

A strategy for the *static* Monte Carlo case involves, for each j, defining the *auxiliary distributions*

$$p_{0j}(x_{1:j}) \qquad j = 1, \ldots, n$$

such that $p_{0n}(x_{1:n}) \equiv \pi_n(x_{1:n})$, and where p_{0j} resembles π_j, but is *directly available*.

Combining Importance Sampling and MCMC

Recall that

$$\int h(x)\pi(x)\,dx = \int h(x)\frac{\pi(x)}{f_0(x)}f_0(x)\,dx = \int h(x)w(x)f_0(x)\,dx$$

If P is a Markov transition kernel with stationary (invariant) distribution π, that is, for all x

$$\int \pi(z)P(z,x)\,dz = \pi(x).$$

Combining Importance Sampling and MCMC

Then

$$\int \frac{\pi(z)}{f_0(z)} P(z, x) f_0(z)\, dz = \pi(x)$$

or

$$\int w(z) P(z, x) f_0(z)\, dz = \pi(x).$$

and so substituting in for $\pi(x)$

$$\begin{aligned}\int h(x)\pi(x)\, dx &= \int h(x) \left\{ \int w(z) P(z, x) f_0(z)\, dz \right\} dx \\ &= \iint h(x) w(z) P(z, x) f_0(z)\, dz\, dx.\end{aligned}$$

Combining Importance Sampling and MCMC

Hence, from the importance sampling identity

$$\mathbb{E}_\pi[h(X)] = \mathbb{E}_{p_0}[w(X)h(X)] = \int\int h(x)w(z)P(z,x)f_0(z)\,dzdx$$

and the estimator

$$\frac{1}{N}\sum_{i=1}^{N} h(X_i)w(Z_i)$$

can be constructed, where

$$Z_1,\ldots,Z_N \sim f_0 \qquad\qquad X_i \sim f_{X|Z}(x|z_i),\ i = 1,\ldots,N$$

where $f_{X|Z}(x|z)$ is the conditional density for X given Z.

Annealed Importance Sampling

Annealed Importance Sampling (AIS) uses a SIS approach, but with auxiliary densities *of the same dimension* defined by

$$p_{0j}(x) = c_j g_{0j}(x) = c_j \{g_0(x)\}^{1-\xi_j} \{g_n(x)\}^{\xi_j}$$

where p_0 is a “diffuse" distribution, and

$$0 = \xi_0 < \xi_1 < \cdots < \xi_n = 1.$$

Annealed Importance Sampling

We have a “path" from $p_{00}(x) \equiv p_0(x)$ to the target distribution

$$p_{0n}(x) \equiv \pi_n(x) = c_n g_n(x).$$

For $j = 1, \ldots, n-1$, let P_j be a Markov kernel with invariant distribution p_{0j}.

Annealed Importance Sampling

The AIS algorithm proceeds as follows:

1. Sample x_1 from p_{00}, set $w_0 = 1/g_0(x_1)$.
2. For $j = 1, 2, \ldots, n-1$,
   (i) Sample x_{j+1} from $P_j(x_j, \cdot)$;
   (ii) Set
   $$w_j = w_{j-1} \frac{p_{0j}(x_j)}{p_{0j}(x_{j+1})} = w_{j-1} \frac{g_{0j}(x_j)}{g_{0j}(x_{j+1})}.$$
3. Return to 1., repeat to produce N samples and weights
   $$x_n^{(1)}, \ldots, x_n^{(N)} \qquad\qquad w_n^{(1)}, \ldots, w_n^{(N)}$$
   where
   $$w_n = \frac{1}{g_0(x_1)} \times \frac{g_{01}(x_1)}{g_{01}(x_2)} \times \frac{g_{02}(x_2)}{g_{02}(x_3)} \times \cdots \times \frac{g_{0\,n-1}(x_{n-1})}{g_{0\,n-1}(x_n)} \times g_n(x_n).$$

Annealed Importance Sampling

Consider the *reverse kernel* P_j^R defined using Bayes Theorem as

$$P_j^R(x_{j+1}, x_j) = \frac{P_j(x_j, x_{j+1}) p_{0j}(x_j)}{p_{0j}(x_{j+1})} = \frac{P_j(x_j, x_{j+1}) g_{0j}(x_j)}{g_{0j}(x_{j+1})},$$

and let

$$g^\star(x_{1:n}) = g_n(x_n) \prod_{j=1}^{n-1} P_j^R(x_{j+1}, x_j)$$

$$g_0^\star(x_{1:n}) = g_0(x_1) \prod_{j=1}^{n-1} P_j(x_j, x_{j+1})$$

where $g_0^\star(x)$ is proportional to the AIS proposal joint density $p_0^\star(x)$.

Annealed Importance Sampling

Define

$$w^\star(x_{1:n}) = \frac{g^\star(x_{1:n})}{g_0^\star(x_{1:n})}$$

as the importance sampling weight for the augmented sample

$$x_{1:n} = (x_1, \ldots, x_n)$$

generated from $p_0^\star$.

Annealed Importance Sampling

We have that

$$w^\star(x_{1:n}) = \frac{g_n(x_n)\prod_{j=1}^{n-1} P_j^R(x_{j+1}, x_j)}{g_0(x_1)\prod_{j=1}^{n-1} P_j(x_j, x_{j+1})}$$

$$= \frac{g_n(x_n)\prod_{j=1}^{n-1} \frac{P_j(x_j, x_{j+1})g_{0j}(x_j)}{g_{0j}(x_{j+1})}}{g_0(x_1)\prod_{j=1}^{n-1} P_j(x_j, x_{j+1})}$$

Annealed Importance Sampling

That is, all the terms in $P_j(\cdot, \cdot)$ cancel, and

$$w^\star(x) = \frac{1}{g_0(x)} \times \frac{g_{01}(x_1)}{g_{01}(x_2)} \times \frac{g_{02}(x_2)}{g_{02}(x_3)} \times \cdots \times \frac{g_{0\,n-1}(x_{n-1})}{g_{0\,n-1}(x_n)} \times g_n(x_n)$$

This is the identical weight to the one computed recursively above.

Annealed Importance Sampling

Note that for each $j = 1, \ldots, n-1$,

$$\int P_j^R(x_{j+1}, x_j)\, dx_j = 1$$

so the marginal distribution of x_n from

$$\pi^\star(x_{1:n}) = c^\star g^\star(x_{1:n})$$

obtained by integrating $g^\star(x_{1:n})$ with respect to

$$x_1, x_2, \ldots, x_{n-1}$$

is precisely the true target $\pi_n(x_n)$.

SMC for State-Space Models

Recall the state-space model: for $t = 1, \ldots, T$

$$\begin{aligned} y_t &= g(\mathbf{x}_t; \theta) + \epsilon_t \\ \mathbf{x}_{t+1} &= h(\mathbf{x}_t; \theta) + \varepsilon_t \end{aligned}$$

where only the series $y_1, \ldots, y_T$, is observed. Sequences $\{\epsilon_t\}$ and $\{\varepsilon_t\}$ are independent zero mean random variables.

If $y_{1:t} = (y_1, \ldots, y_t)$, $\mathbf{x}_{1:t} = (\mathbf{x}_1, \ldots, \mathbf{x}_t)$. we must carry out

- **Filtering:** Compute $p(\mathbf{x}_t | y_{1:t})$
- **Prediction:** Compute $p(\mathbf{x}_{t+1} | y_{1:t})$
- **Smoothing:** Compute $p(\mathbf{x}_t | y_{1:T})$

SMC for State-Space Models

Filtering: By Bayes theorem

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:(t-1)})}{p(y_t|y_{1:(t-1)})} \propto p(y_t|x_t)p(x_t|y_{1:(t-1)})$$

where the function $p(y_t|x_t)$ is usually a straightforward conditional density, but the function $p(x_t|y_{1:(t-1)})$ is the density computed at the previous time point **prediction** step.

Prediction:

$$p(x_{t+1}|y_{1:t}) = \int p(x_{t+1}|x_t)p(x_t|y_{1:t})\, dx_t$$

where $p(x_{t+1}|x_t)$ is again a conditional density derived from the state equation, and $p(x_t|y_{1:t})$ is the posterior density computed at the time point *filtering* step.

We consider simulation-based versions of these calculations.

For example, in the *filtering* step, suppose we have a sample

$$z_t^{(1)}, \ldots, z_t^{(N)}$$

from $p(x_t|y_{1:(t-1)})$ obtained at the previous prediction step. Then we can obtain a *particle approximation* to $p(x_t|y_{1:t})$ as

$$\widehat{p}(x_t|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} p(y_t|z_t^{(i)})\delta_{z_t^{(i)}}$$

Note also that the denominator $p(y_t|y_{1:(t-1)})$ can be approximated by the sum of the weights

$$\widehat{p}(y_t|y_{1:(t-1)}) = \frac{1}{N}\sum_{i=1}^{N} p(y_t|z_t^{(i)})$$

which is useful in approximating the likelihood by

$$\widehat{\mathcal{L}}(y_{1:t}) = \widehat{p}(y_1)\prod_{t=2}^{T}\widehat{p}(y_t|y_{1:(t-1)})$$

We can produce a sample from the particle approximation

$$\widehat{p}(x_t|y_{1:t})$$

by performing a *weighted resampling* from

$$(z_t^{(1)}, \ldots, z_t^{(N)})$$

with weights proportional to $p(y_t|z_t^{(i)})$.

Denote this resampled vector

$$v_t^{(1)}, \ldots, v_t^{(N)}$$

For the *prediction* step, the particle approximation to

$$p(x_{t+1}|y_{1:t})$$

is obtained using a similar Monte Carlo strategy

$$\widehat{p}(x_{t+1}|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} p(x_{t+1}|v_t^{(i)})$$

A sample from this approximation can be obtained by propagating the samples $v_t^{(1)}, \ldots, v_t^{(N)}$ forward by sampling the transition densities $p(x_{t+1}|v_t^{(i)})$ for each i.

This yields the sample

$$z_{t+1}^{(1)}, \ldots, z_{t+1}^{(N)}$$

and the recursion continues.

To perform SIS for $\pi_n(x_{1:n}) = g_n(x_{1:n})/Z_n$, use the IS density

$$p_n(x_{1:n}) = p_1(x_1) \prod_{j=2}^{n} \widetilde{p}_j(x_j | x_{1:(j-1)})$$

with weights given by $w_1(bx_1) = \pi_1(x_1)/p_1(x_1)$ and

$$\begin{aligned} w_j(x_{1:j}) &= w_{j-1}(x_{1:(j-1)}) \frac{\widetilde{\pi}_j(x_j | x_{1:(j-1)})}{\widetilde{p}_j(x_j | x_{1:(j-1)})} \\ &= w_{j-1}(x_{1:(j-1)}) \frac{\pi_j(x_{1:j})}{\pi_{j-1}(x_{1:(j-1)}) \widetilde{p}_j(x_j | x_{1:(j-1)})} \end{aligned}$$

If the normalizing constants Z_j for $j = 1, \ldots, n$ are unknown, the weights can be replaced by $w_1(x_1) = g_1(x_1)/p_1(x_1)$ and

$$\begin{aligned} w_j(x_{1:j}) &= w_{j-1}(x_{1:(j-1)}) \frac{g_j(x_{1:j})}{g_{j-1}(x_{1:(j-1)}) \widetilde{p}_j(x_j | x_{1:(j-1)})} \\ &= w_{j-1}(x_{1:(j-1)}) \alpha_j(x_{1:j}) \end{aligned}$$

say.

The Monte Carlo computation is carried out by sampling

$$x_1^{(1)}, \ldots, x_1^{(N)}$$

from p_1, then at step $j \geqslant 2$ sampling for $i = 1, \ldots, N$

$$x_j^{(1)}, \ldots, x_j^{(N)} \quad \text{from} \quad \widetilde{p}_j(.|x_{1:(j-1)}^{(i)})$$

The importance sampling weights are $w_1(x_1^{(i)})$ and, for $j \geqslant 2$,

$$w_j(x_{1:j}^{(i)}) = w_{j-1}(x_{1:(j-1)}^{(i)})\alpha_j(x_{1:j}^{(i)})$$

We then have

$$\widehat{\pi}_n(\mathbf{x}) = \sum_{i=1}^{N} W_n^{(i)} \delta_{x_{1:n}^{(i)}}(\mathbf{x})$$

where

$$W_n^{(i)} = \frac{w_n(x_{1:n}^{(i)})}{N\widehat{Z}_n} \qquad \widehat{Z}_n = \frac{1}{N}\sum_{i=1}^{N} w_n(x_{1:n}^{(i)})$$

in the usual way.

State-Space Models

In a simple state-space model, write

$$\begin{aligned} \text{Observation equation} &: p_{Y|X}(y_j|x_j) \\ \text{State equation} &: p_X(x_j|x_{j-1}) \end{aligned}$$

with $X_1 \sim p_{X_1}$.

In light of the observed data $y_{1:n} = (y_1, \ldots, y_n)$, the posterior of interest is

$$\pi_n(x_{1:n}) = p(x_{1:n}|y_{1:n}) = \frac{p(x_{1:n}, y_{1:n})}{p(y_{1:n})} = \frac{g_n(x_{1:n}, y_{1:n})}{Z_n(y_{1:n})}$$

We have

$$g_n(x_{1:n}, y_{1:n}) = p_{X_1}(x_1) \left\{ \prod_{j=2}^{n} p_X(x_j|x_{j-1}) \right\} \left\{ \prod_{j=1}^{n} p_{Y|X}(y_j|x_j) \right\}$$

and

$$Z_n(y_{1:n}) = \int g_n(x_{1:n}, y_{1:n})\, dx_{1:n}$$

For SMC, the optimal choice of IS density $\widetilde{p}_j(x_j|x_{1:(j-1)})$ is

$$\widetilde{\pi}_j(x_j|x_{1:(j-1)})$$

which in the simple state-space model is given by

$$\widetilde{\pi}_j(x_j|x_{1:(j-1)}) \equiv \widetilde{\pi}_j(x_j|y_{1:j}, x_{1:(j-1)}) = \frac{p_{Y|X}(y_j|x_j)p_X(x_j|x_{j-1})}{p(y_j|x_{j-1})}$$

$$= \frac{p_{Y|X}(y_j|x_j)p_X(x_j|x_{j-1})}{\int p_{Y|X}(y_j|x)p_X(x|x_{j-1})\,dx}$$

as, conditional on $x_{1:(j-1)}$, x_j is independent of $y_{1:(j-1)}$.

In most cases (specifically, outside of the Gaussian case), the optimal choice is not feasible: recall that we need successive samples from the IS densities

$$\widetilde{p}_j(x_j|x_{1:(j-1)}) = \widetilde{\pi}_j(x_j|y_{1:j}, x_{1:(j-1)}) = \widetilde{\pi}_j(x_j|y_j, x_{(j-1)}).$$

We might choose instead

$$\widetilde{p}_j(x_j|x_{1:(j-1)}) = p_X(x_j|x_{1:(j-1)})$$

which typically is straightforward to sample from, and yields the recursive weight calculation

$$w_j(x_{1:j}) = w_{j-1}(x_{1:(j-1)})p_{Y|X}(y_j|x_j)$$

Example: Stochastic volatility model

Suppose

$$\text{Observation equation} \ : \ p_{Y|X}(y_j|x_j) \equiv \mathcal{N}(0, \exp\{2x_j\})$$

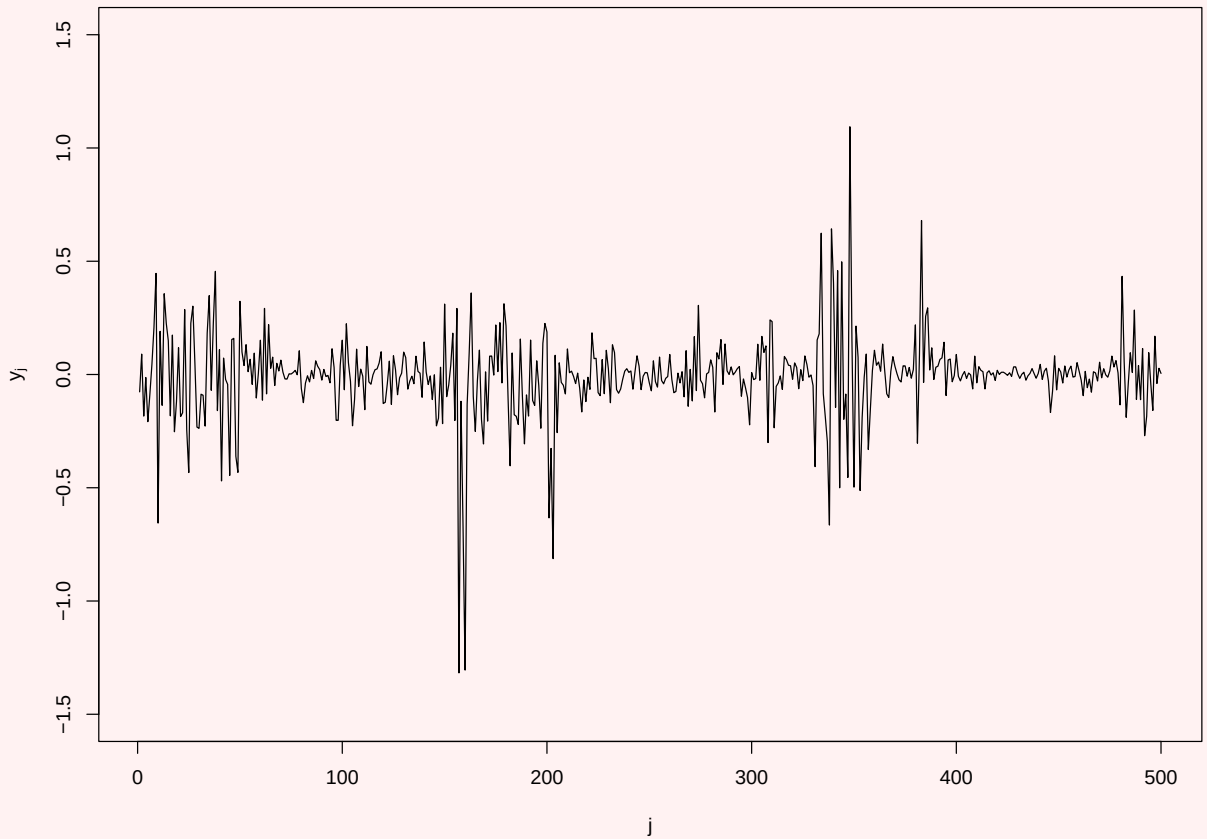
$$\text{State equation} : \ p_X(x_j|x_{j-1}) \equiv \mathcal{N}(\mu + \phi(x_{j-1} - \mu), \sigma^2)$$

with $p_{X_1}(x) \equiv \mathcal{N}(\mu, \sigma^2/(1-\phi^2))$. We regard μ, ϕ and σ as known constants.

This model is used in the analysis of financial time series: it is a non-linear state-space model, due to the non-linear dependence of the distribution of y_j on x_j.

Example: Stochastic volatility model

$n = 500$ realizations from the stochastic volatility model.



Example: Stochastic volatility model

Here, for $j \geqslant 2$,

$$p_{Y|X}(y_j|x_j) = \left(\frac{e^{-2x_j}}{2\pi}\right)^{1/2} \exp\{-e^{-2x_j} y_j^2/2\}$$

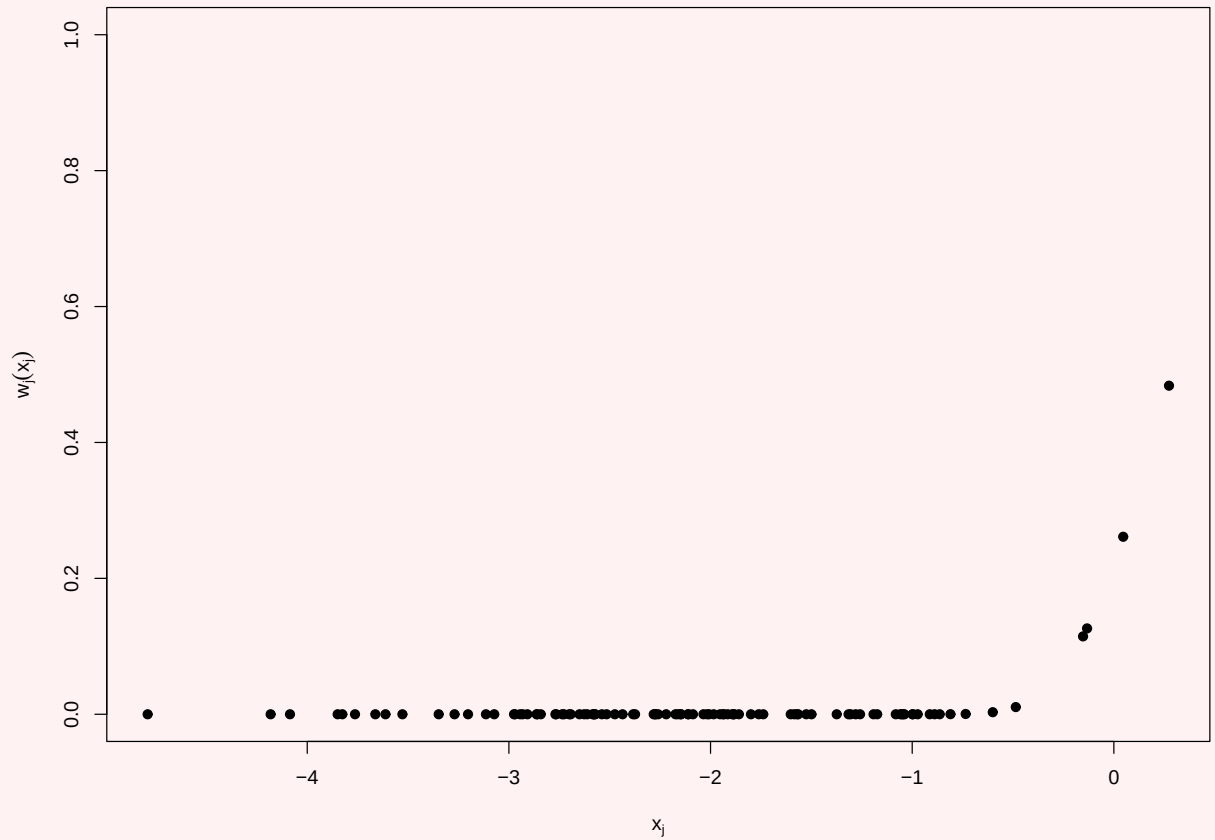
and the recursive weight calculation is based on

$$w_j(x_{1:j}) = \prod_{k=1}^{j} \frac{\widetilde{\pi}_k(x_k|x_{1:(k-1)})}{\widetilde{p}_k(x_k|x_{1:(k-1)})} = \prod_{k=1}^{j} p_{Y|X}(y_k|x_k)$$

that is, the likelihood up to observation j.

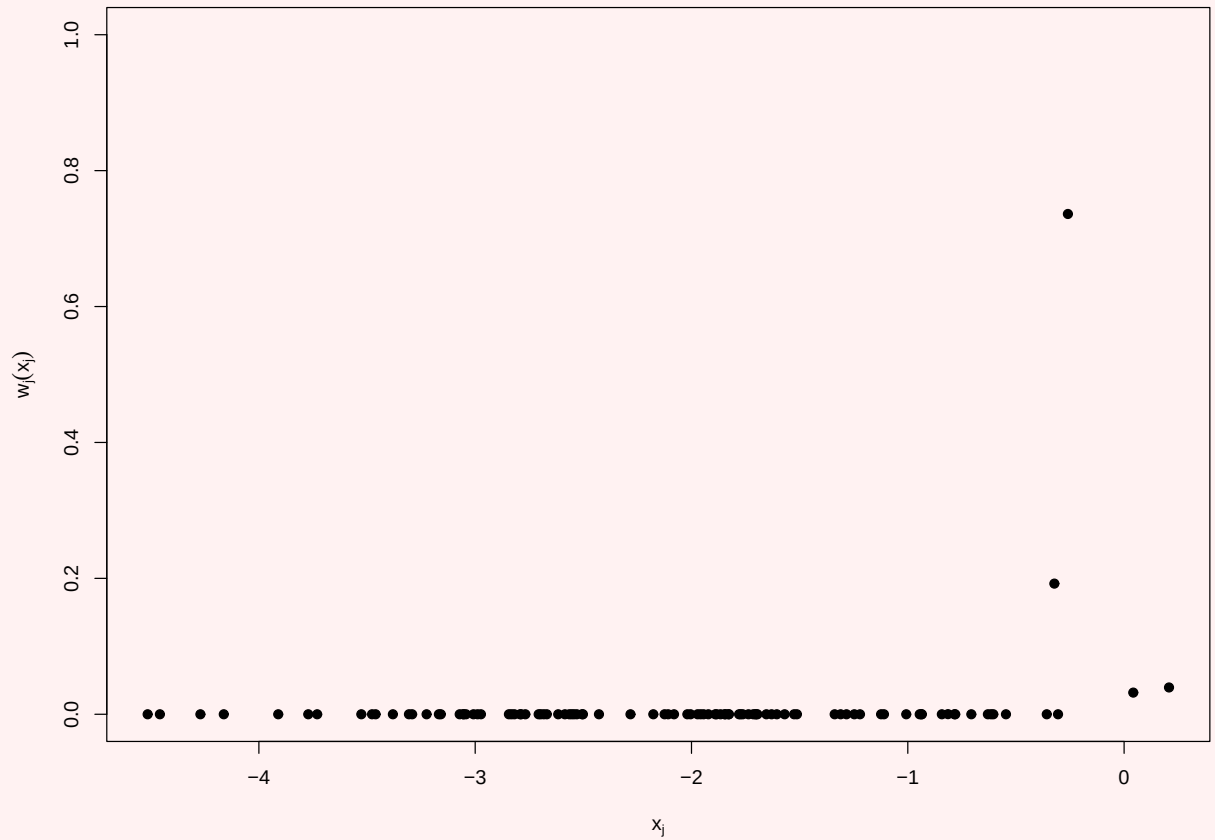
Example: Stochastic volatility model

$N = 100$ particles and weights at $j = 1$



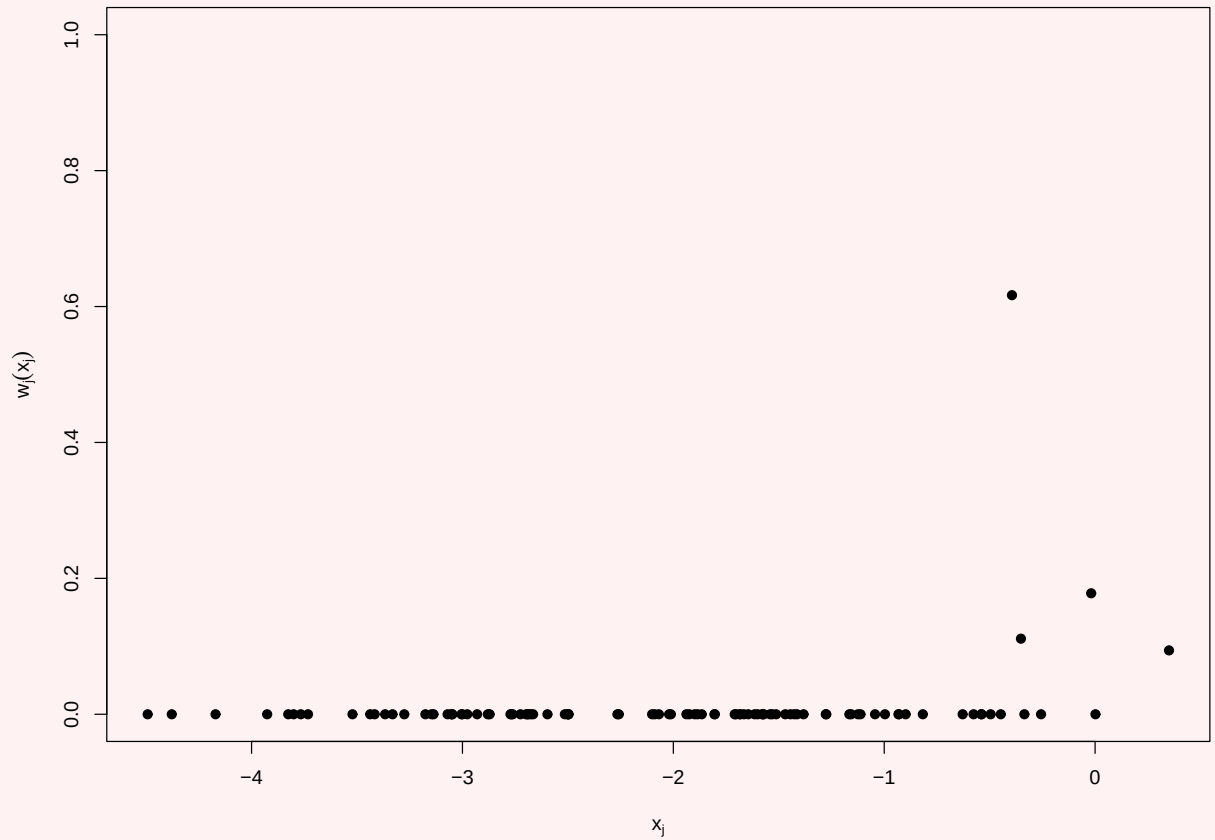
Example: Stochastic volatility model

$N = 100$ particles and weights at $j = 2$



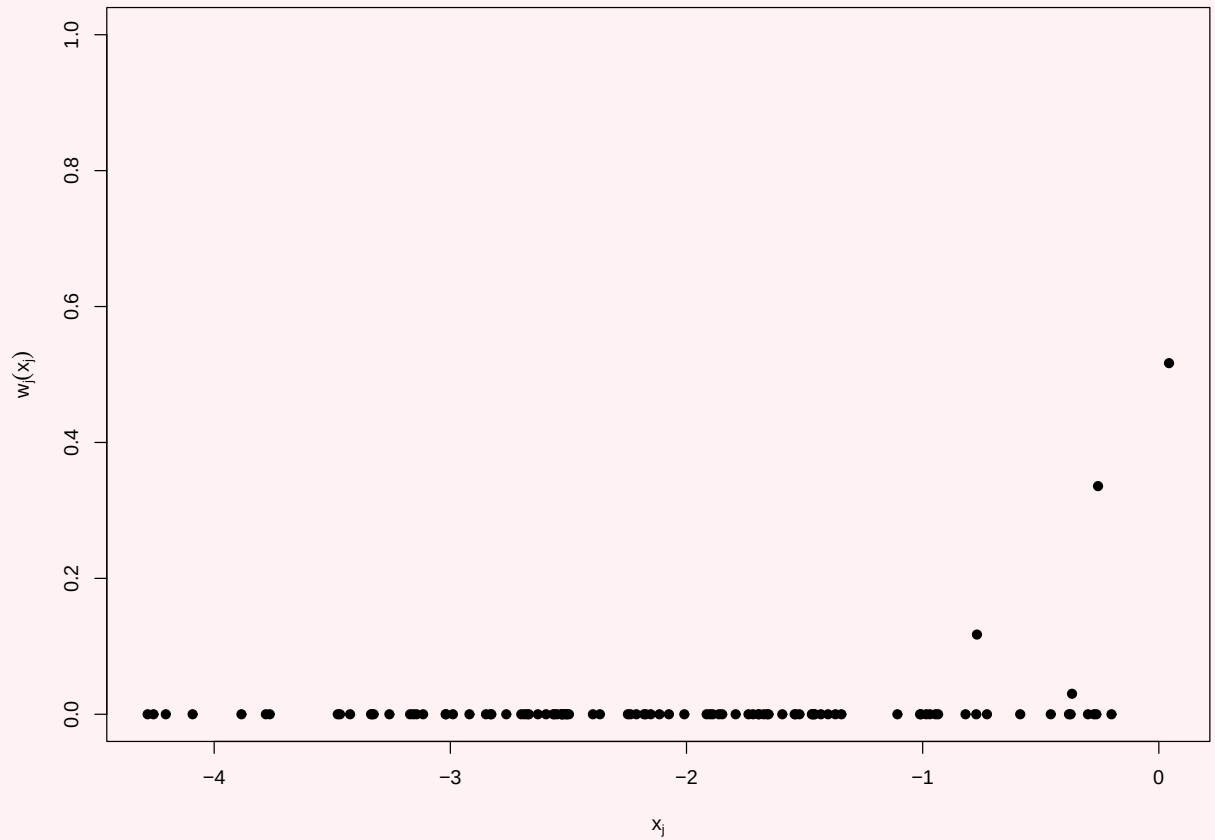
Example: Stochastic volatility model

$N = 100$ particles and weights at $j = 3$



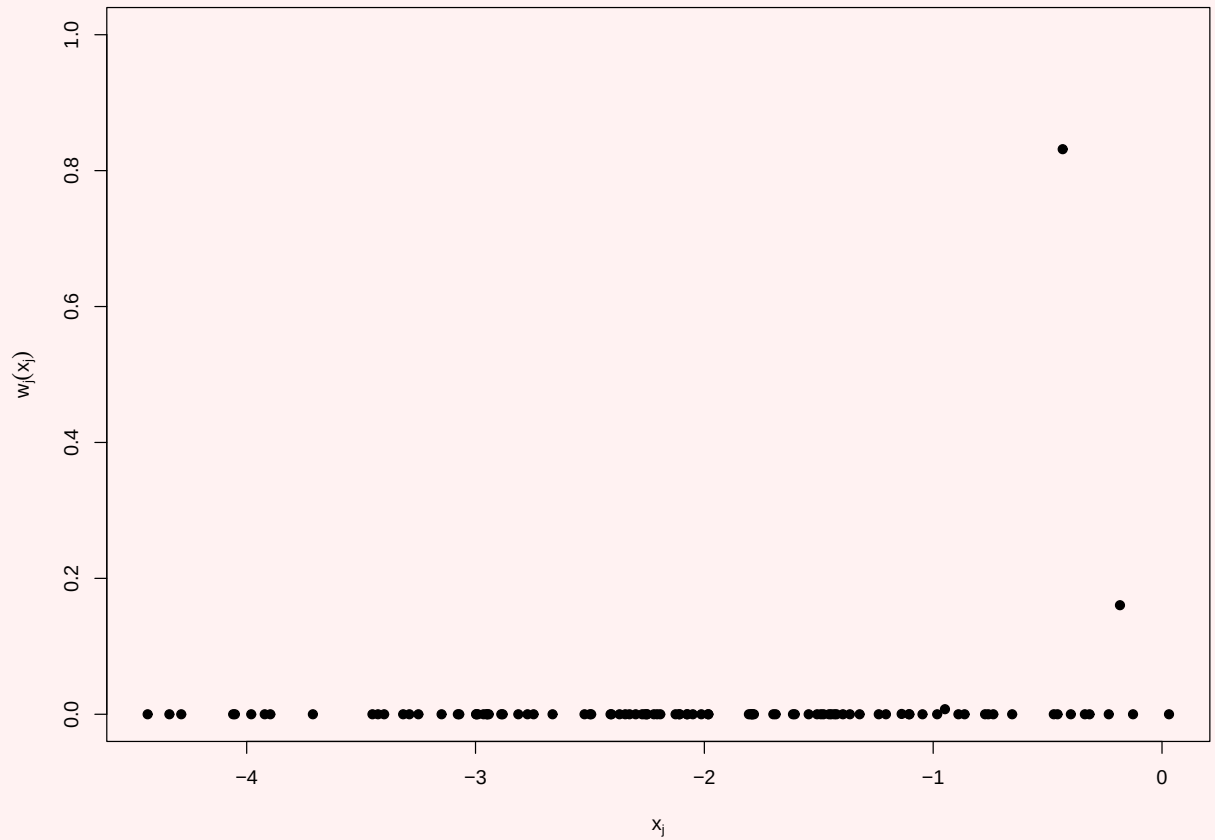
Example: Stochastic volatility model

$N = 100$ particles and weights at $j = 4$



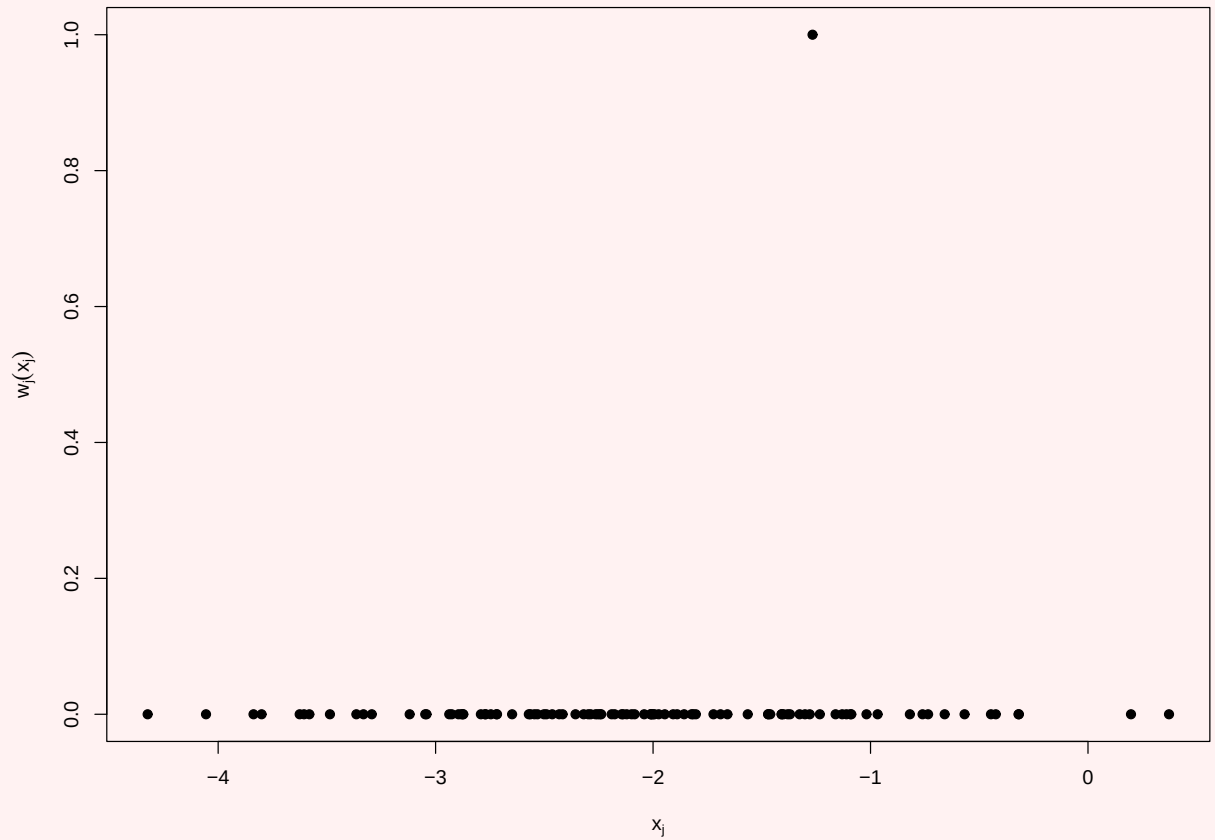
Example: Stochastic volatility model

$N = 100$ particles and weights at $j = 5$



Example: Stochastic volatility model

$N = 100$ particles and weights at $j = 20$



Particle Degeneracy

A problem with this strategy is that the weights start to *degenerate* as j increases;

- the weights that define the Monte Carlo estimate $\widehat{\pi}_n(\mathbf{x})$ are very small for most i, with only a few large weights.
- the empirical variance of the collection of unnormalized weights $\{w_j(x_j^{(i)})\}$ for $i = 1, \ldots, N$ increases
- this is mitigated, but not resolved by a better choice of $\widetilde{p}_j(x_j | x_{1:(j-1)})$

Particle Degeneracy

- could try a normal approximation of

$$p(x_j|y_{1:j}, x_{1:(j-1)}) \propto p_{Y|X}(y_j|x_j)p_X(x_j|x_{j-1})$$

$$= \exp\left\{-x_j - \frac{y_j^2}{2e^{2x_j}} - \frac{1}{2\sigma^2}(x_j - \mu_j)^2\right\}$$

where $\mu_j = \mu + \phi(x_{j-1} - \mu)$.

Example: Stochastic volatility model

Here the mode of

$$p(x|y_j, x_{j-1}) = \exp\left\{-x - \frac{y_j^2}{2e^{2x}} - \frac{1}{2\sigma^2}(x - \mu_j)^2\right\}$$

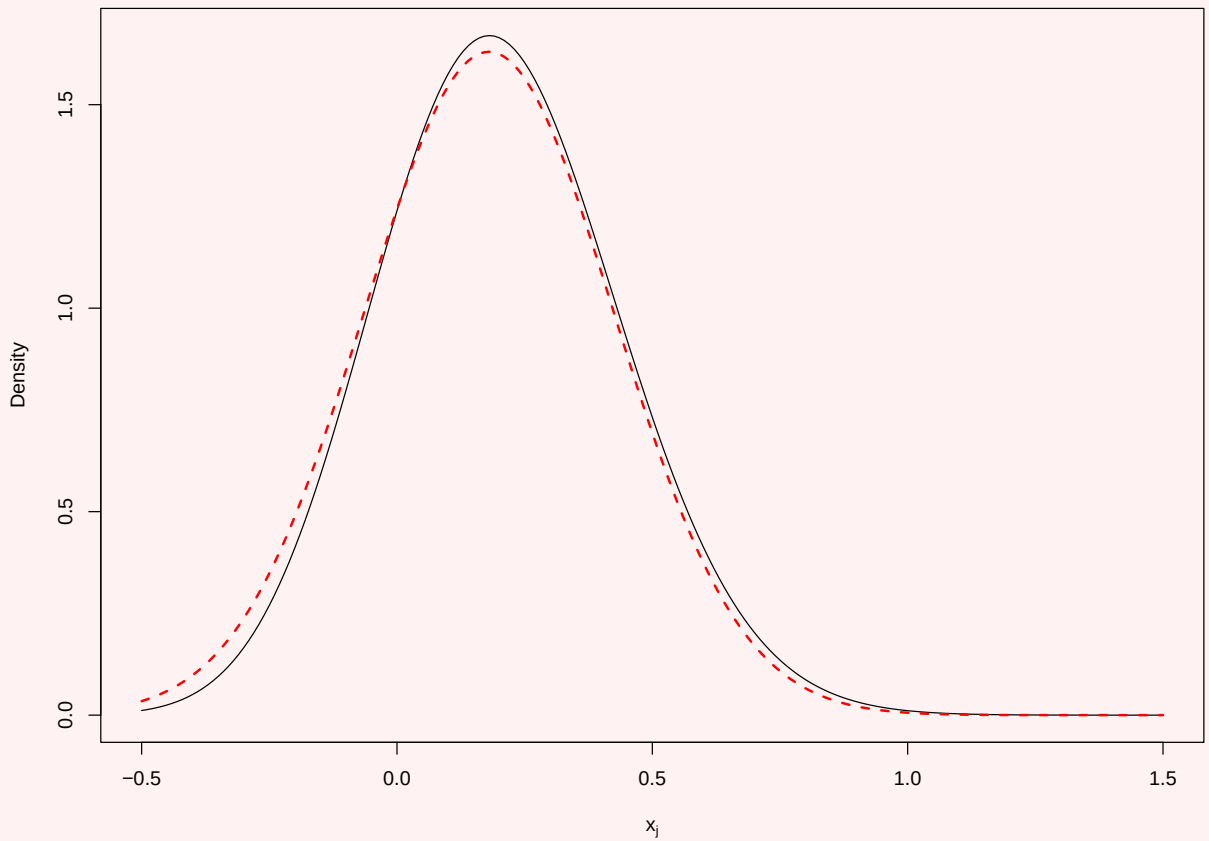
can be computed numerically, and the variance of the normal approximation can be computed by inspecting the curvature at the mode, given by

$$\left[-\frac{\partial^2 \log p(x|y_j, x_{j-1})}{\partial x^2}\right]^{-1} = \left[2y_j^2 e^{-2x} + 1/\sigma^2\right]^{-1}$$

Particle Degeneracy

Example: Stochastic volatility model

Normal approximation to $p(x_j|y_j, x_{j-1})$ for $x_{j-1} = 0, y_j = 2$:



Particle Degeneracy

The main reason for particle degeneracy is that particles with low weight at step $j-1$ tend to have low weight at step j.

Particle degeneracy can be overcome by *resampling* that removes the particles with low weights. The commonest form of resampling is to use a non-parametric bootstrap, with multinomial sampling.

Particle Degeneracy

At step j, the Monte Carlo estimate of π_j is

$$\widehat{\pi}_j(\mathbf{x}) = \sum_{i=1}^{N} W_j(x_{1:j}^{(i)}) \delta_{x_{1:j}^{(i)}}(\mathbf{x})$$

This discrete distribution on

$$x_{1:j}^{(1)}, \ldots, x_{1:j}^{(N)}$$

is resampled N times according to the weights

$$W_j(x_{1:j}^{(1)}), \ldots, W_j(x_{1:j}^{(N)}).$$

Particle Degeneracy

After resampling, the new resampled values replace the

$$x_{1:j}^{(1)}, \ldots, x_{1:j}^{(N)}$$

collection, and each resampled point carries with it its normalized resampling weight $W_j(x_{1:j}^{(i)})$.

Each original value $x_{1:j}^{(i)}$ may appear in the resampled collection more than once.

Particle Degeneracy

Let $n_{j1}, \ldots, n_{jN}$ denote the numbers of times that each of the N particles are resampled during multinomial sampling. Let

$$\widetilde{\pi}_j(\mathbf{x}) = \sum_{i=1}^{N} \frac{n_{ji}}{N} \delta_{x_{1:j}^{(i)}}(\mathbf{x})$$

Resampling does not disrupt the particle approximation in expectation (that is, it does not introduce bias)

$$\mathbb{E}\left[\widetilde{\pi}_j(\mathbf{X}) | \widehat{\pi}_j(x_{1:j})\right] = \widehat{\pi}_j(x_{1:j})$$

but it does introduce “noise" - for any suitable h,

$$\mathrm{Var}_{\widetilde{\pi}_j}\left[h(x_{1:j})\right] \geqslant \mathrm{Var}_{\widehat{\pi}_j}\left[h(x_{1:j})\right]$$

This follows by the law of iterated variance.

Particle Degeneracy

The message from this result is that although you should perform resampling to preserve particle diversity, you should not resample too often, as that introduces variability.

A common strategy is to resample only when the observed variance of the weights exceeds some threshold. A statistic to track is the *effective sample size* (ESS) where

$$\text{ESS} = \left(\sum_{i=1}^{N} \left\{ W_j^{(i)} \right\}^2 \right)^{-1}$$

We have $1 \leqslant \text{ESS} \leqslant N$:

$$\text{ESS} = 1 \quad \text{if} \quad W_j^{(i)} = 1 \quad \text{for some } i$$

$$\text{ESS} = N \quad \text{if} \quad W_j^{(i)} = \frac{1}{N} \quad \text{for each } i$$

Particle Degeneracy

Can also use *entropy* as a measure

$$\text{Entropy} = -\sum_{i=1}^{N} W_j^{(i)} \log W_j^{(i)}$$

where $0 \leqslant \text{Entropy} \leqslant \log_2 N$

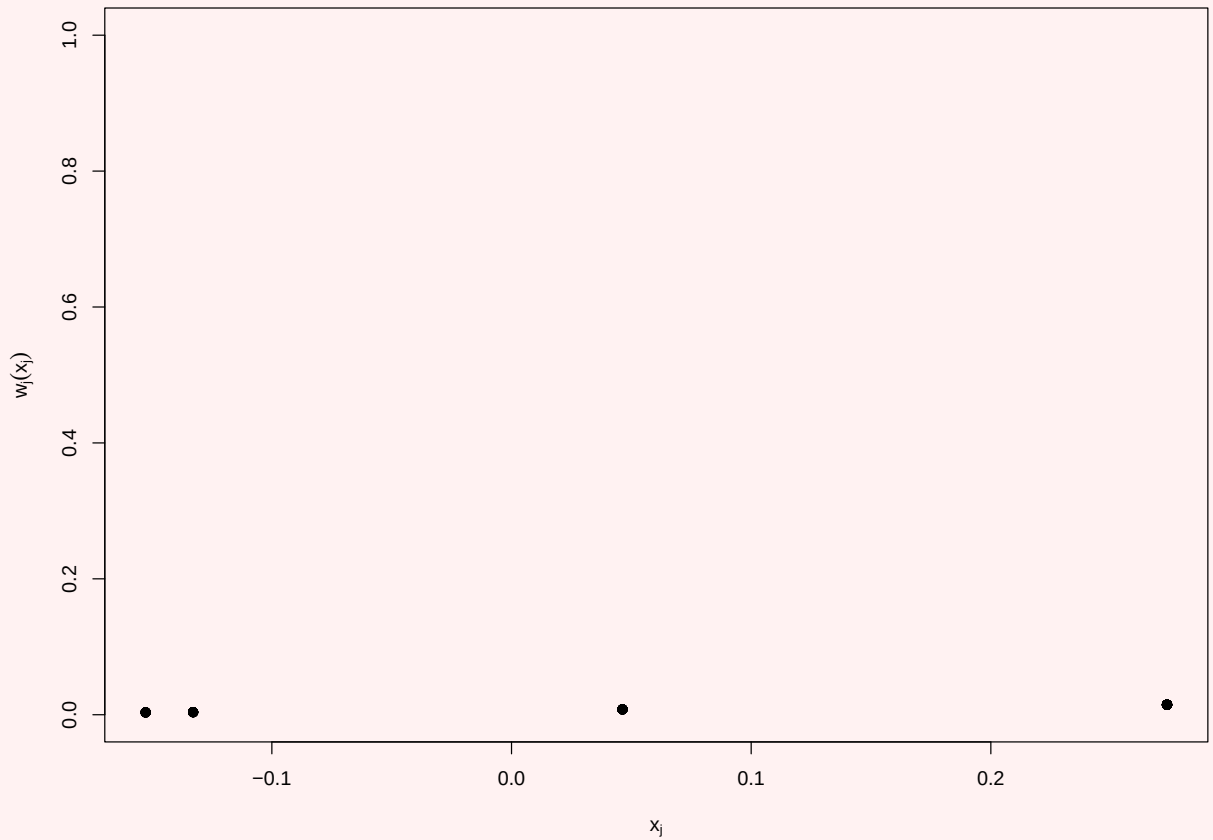
$$\text{Entropy} = 0 \quad \text{if} \quad W_j^{(i)} = 1 \quad \text{for some } i$$

$$\text{Entropy} = \log_2 N \quad \text{if} \quad W_j^{(i)} = \frac{1}{N} \quad \text{for each } i$$

Particle Degeneracy

Example: Stochastic volatility model

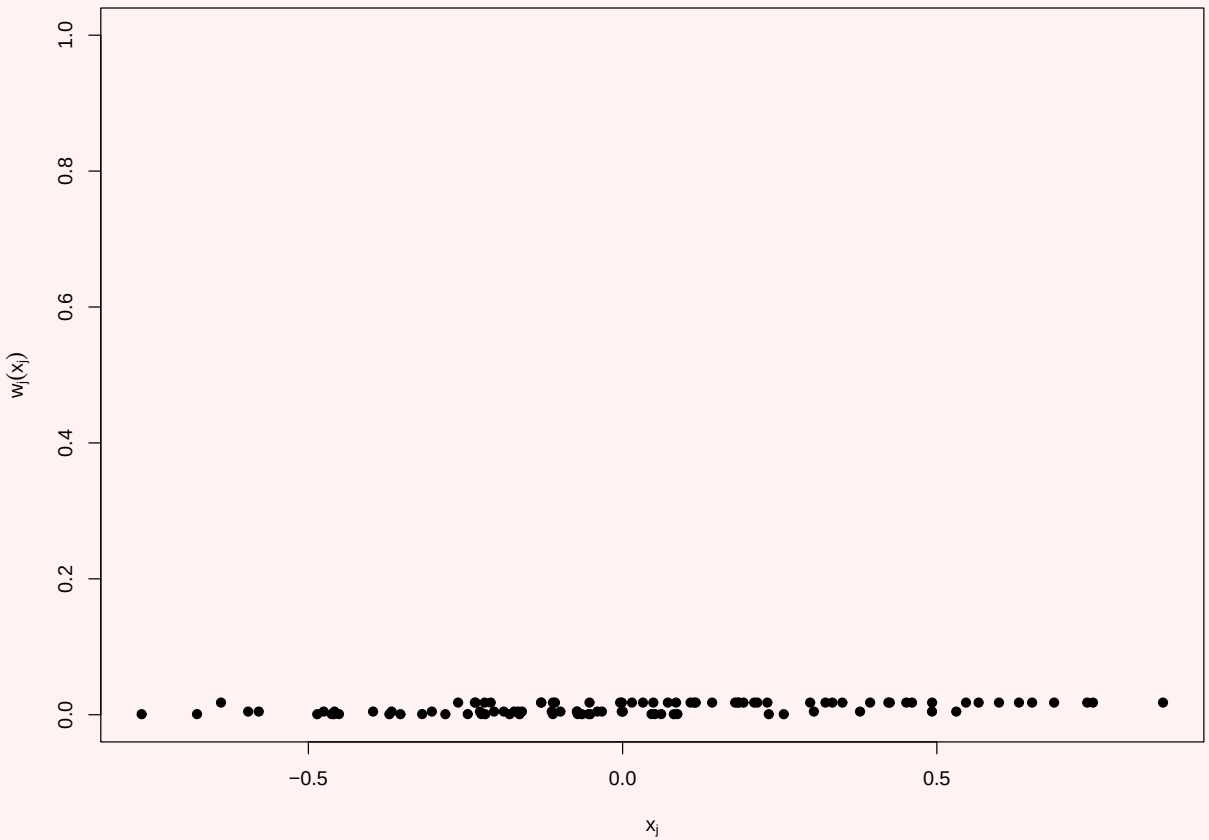
$N = 100$ resampled particles and weights at $j = 1$



Particle Degeneracy

Example: Stochastic volatility model

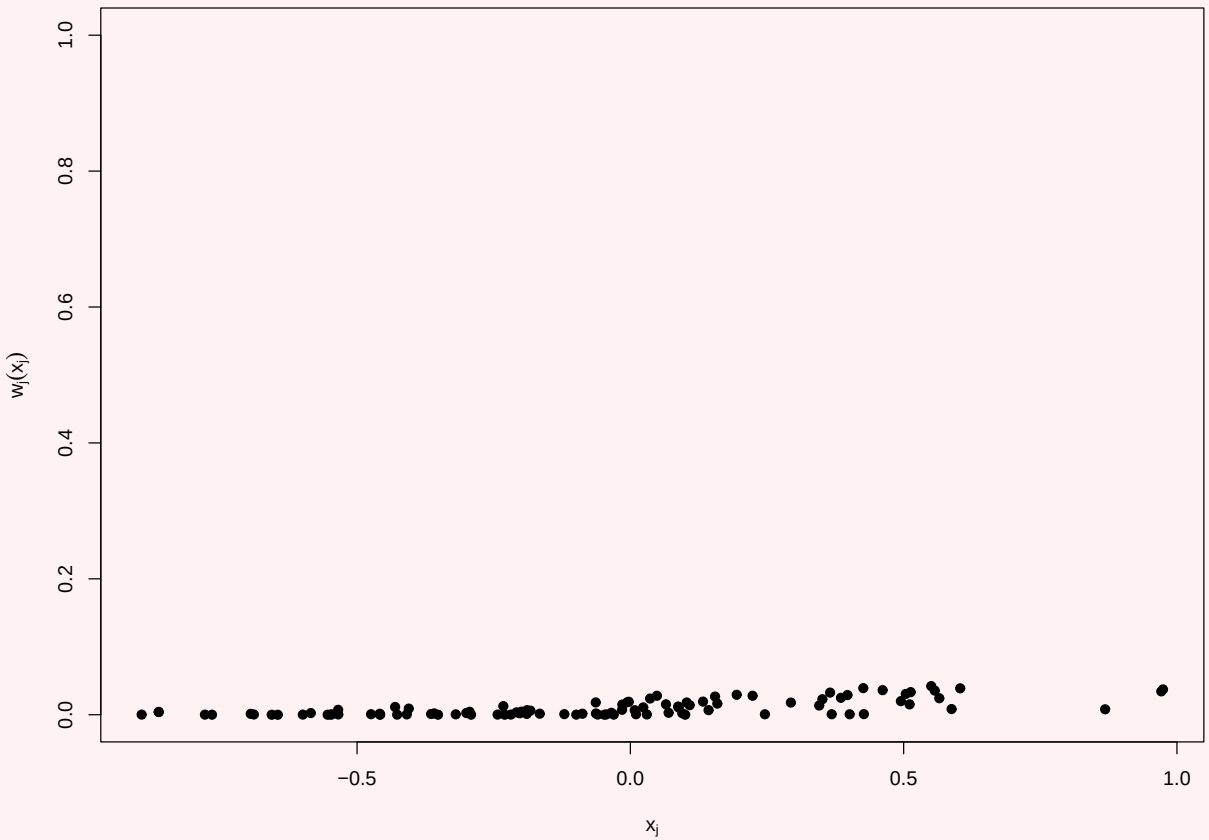
$N = 100$ resampled particles and weights at $j = 2$



Particle Degeneracy

Example: Stochastic volatility model

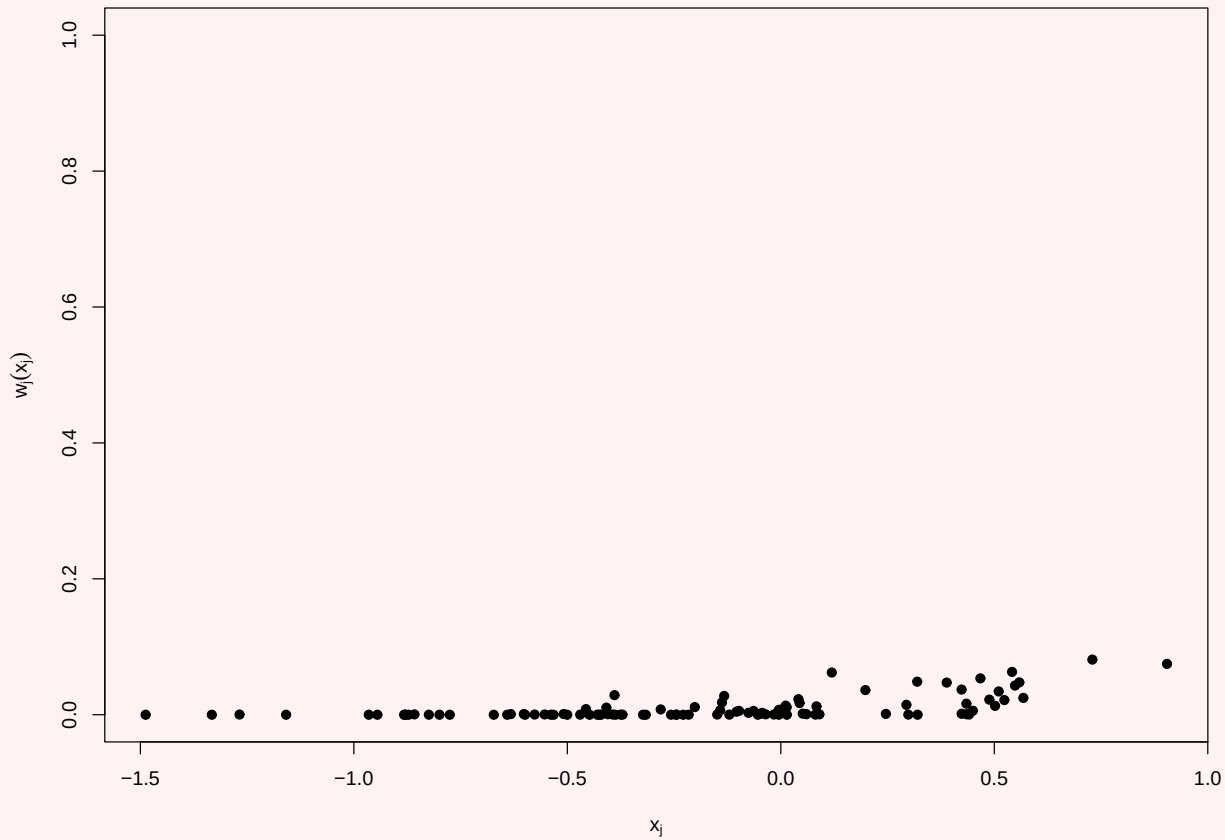
$N = 100$ resampled particles and weights at $j = 3$



Particle Degeneracy

Example: Stochastic volatility model

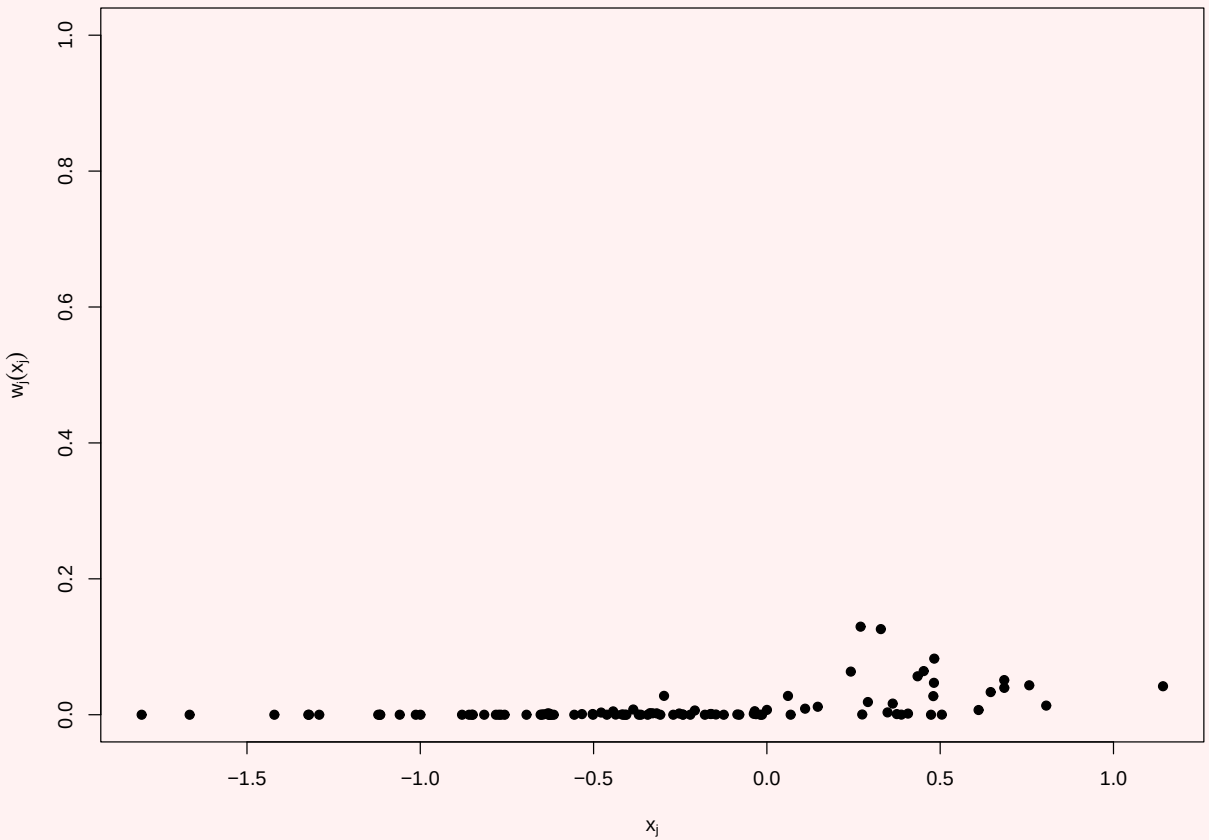
$N = 100$ resampled particles and weights at $j = 4$



Particle Degeneracy

Example: Stochastic volatility model

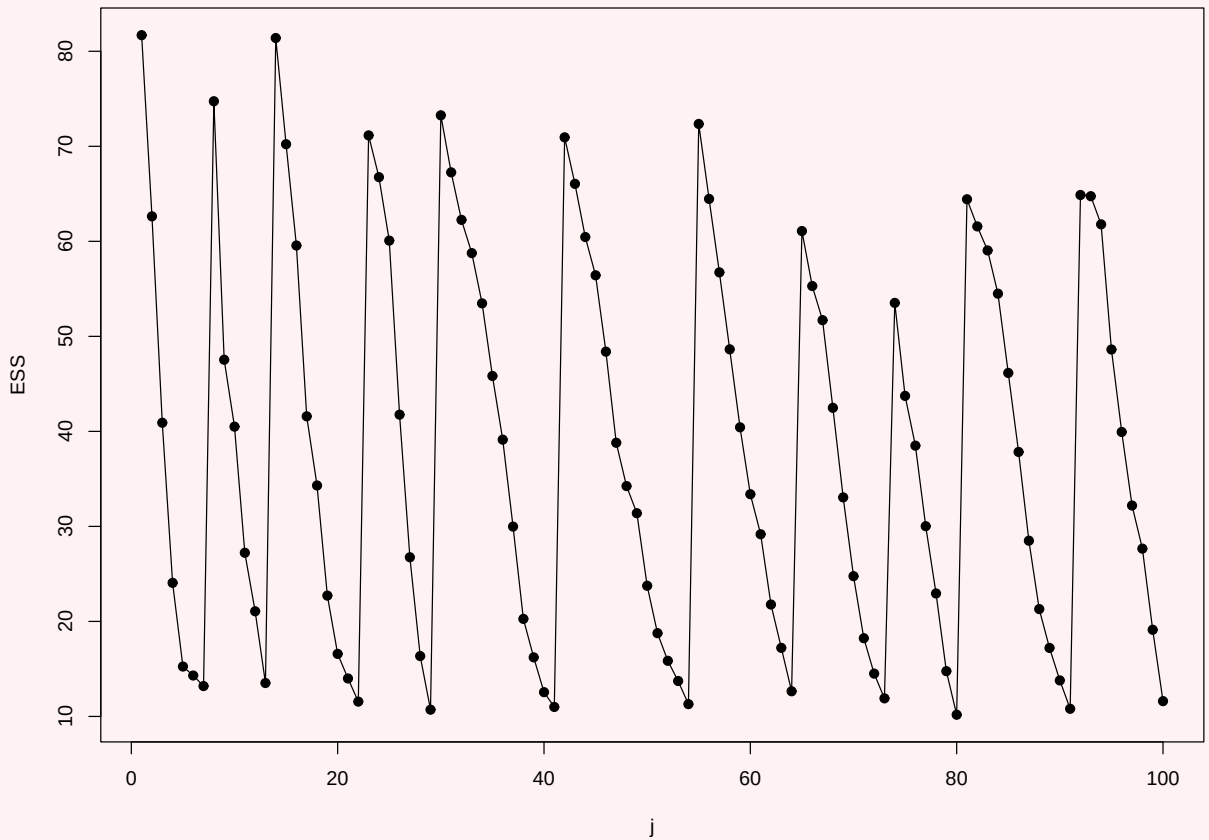
$N = 100$ resampled particles and weights at $j = 5$



Particle Degeneracy

Example: Stochastic volatility model

Effective sample size (ESS) with resampling when ESS < 10



Examples

Example: SMC for the stochastic volatility model

See `knitr 16`

Model-selection via trans-dimensional MCMC

Model selection via MCMC exploits trans-dimensional moves

- Discrete time chains
  - Carlin and Chib (1995): extend the state space to consider all possible models simultaneously
- Continuous time chains
  - Grenander and Miller (1994), Phillips and Smith (1995): jump diffusions.
  - Geyer and Moller (1994), Stephens (M) (2000): birth and death in continuous time

Model-selection via trans-dimensional MCMC

In the contributed Discussion of Grenander and Miller (1994), Green laid out the basic construction of

Reversible Jump MCMC.

- Green pointed out that whereas their Gibbs sampler construction might only be rarely of use, their MH algorithm they also suggested could provide a general mechanism for jumping between subspaces of different dimension.
- The suggested general algorithm retained the reversibility and detailed balance properties of the usual MH algorithm, but required “dimension-matching" terms to account for the differences in dimension of different subspaces.

Reversible Jump MCMC

- target distribution π
- parameter vector $\theta \in \Theta \subseteq \mathcal{R}^d$;
- proposal kernel $Q(\theta, d\theta') = \Pr(\theta' \in d\theta' | \theta)$,
- transition kernel $P(\theta, d\theta')$ with π as its unique stationary distribution.

Let π and q denote the densities of π and Q wrt to Lebesgue measure.

Reversible Jump MCMC

By the usual arguments, the acceptance probability for values generated from q is

$$\alpha(\theta, \theta') = \min\left\{1, \frac{\pi_n(\theta')}{\pi_n(\theta)} \frac{q(\theta', \theta)}{q(\theta, \theta')}\right\}$$

Under mild conditions on $Q(.,.)$, variates generated from this Markov chain with transition kernel $P(.,.)$ form a dependent sample from π, and the ergodicity of the chain permits Monte Carlo estimation of functionals of π.

Reversible Jump MCMC

Consider a countable collection of Bayesian models, $\{M_k, k = 1, 2, \ldots\}$, where model M_k is parameterized via parameter θ_k with parameter space $\Theta_k \subset \mathbb{R}^{d_k}$. We formulate a joint prior on the parameters and the unknown model $\mathcal{M}$ by setting a discrete prior

$$\pi_0(M_k) = \Pr(\mathcal{M} = M_k)$$

say, and then the conditional prior

$$\pi_0(\theta_k | M_k)$$

so that the full joint posterior can be written

$$\pi_n(M_k, \theta_k) = \frac{\mathcal{L}_n^{M_k}(\theta_k)\pi_0(\theta_k|M_k)\pi_0(M_k)}{\sum_j \left\{ \int \mathcal{L}_n^{M_j}(t_j)\pi_0(t_j|M_j)\, dt_j \right\} \pi_0(M_j)}$$

Reversible Jump MCMC

Our objective is to construct a Metropolis-Hastings chain to that is aperiodic and irreducible on the *union* of parameter spaces

$$\Theta = \bigcup_k \Theta_k .$$

This requires the specification of a proposal transition density, $q(.,.)$, but, in contrast to the usual fixed-dimension case, we face the difficulty that the arguments of q are potentially of different dimensions, rendering the reversibility requirement difficult to meet.

Reversible Jump MCMC

At a specific iteration, suppose that

- the chain is in model M with parameter θ having dimension d,
- the proposal is to move to model M' with parameter θ' having dimension d'.

We envisage this move as first selecting a move between models, $M \longrightarrow M'$, and then the proposal of a θ' possibly dependent on the current θ.

Reversible Jump MCMC

For a reversible proposal mechanism, we need equally dimensioned arguments to $q(.,.)$: consider the introduction of collections of latent variables $\mathbf{u}$ and $\mathbf{v}$ of dimension d_u and d_v respectively, so that $d + d_u = d' + d_v$.

The proposal density q is then considered for the extended parameter vectors $(\theta, \mathbf{u})$ and $(\theta', \mathbf{v})$ such that $q((\theta, \mathbf{u}), (\theta', \mathbf{v}))$ is reversible; this is most easily constructed using a 1-1 differentiable mapping, that is

$$(\theta', \mathbf{v}) = \mathbf{g}(\theta, \mathbf{u}) \qquad \Longleftrightarrow \qquad (\theta, \mathbf{u}) = \mathbf{h}(\theta', \mathbf{v}).$$

Standard (fixed-dimension) moves are special cases of this general proposal procedure; for example, for a Metropolis move, we might set

$$\theta' = \theta + \mathbf{u} \qquad \mathbf{u} \sim Normal_d(0, \Sigma)$$

and for Metropolis-Hastings moves, we typically use a conditional generation $q(\theta'|\theta)$ taking the conditioning variable as a constant parameter in a suitably chosen density; the stochastic elements $\mathbf{u}$ in the MH move can be thought of as the uniform variates used to perform basic random number generation from this conditional density.

Reversible Jump MCMC

To establish the acceptance probability for jump move, consider a "hybrid" MH algorithm comprising a mixture of move types, with

- move m
- transition proposal Q_m
- move m selected with probability r_m

for a finite collection of move types, some of which may be trans-dimensional, but each of which retains the detailed balance property.

If ψ and ψ' represent the latent-augmented parameter vectors, and let the augmented posterior density be denoted

$$\widetilde{\pi}_m(\psi) = \widetilde{\pi}_m(M, \theta, \mathbf{u}) = \pi_n(M, \theta) p_U(\mathbf{u})$$

Reversible Jump MCMC

Let $\alpha_m(\psi, \psi')$ denote the acceptance probability for move type m, so that for arbitrary sets A and B

$$\int_A \tilde{\pi}_m(d\psi) \int_B \alpha_m(\psi, \psi') Q_m(\psi, d\psi') = \int_B \tilde{\pi}_m(d\psi') \int_A \alpha_m(\psi', \psi) Q_m(\psi', d\psi)$$

which implies as usual that

$$\alpha_m(\psi, \psi')\tilde{\pi}_m(\psi) q_m(\psi, \psi') = \alpha_m(\psi', \psi)\tilde{\pi}_m(\psi') q_m(\psi', \psi)$$

or,

$$\alpha_m(\psi, \psi') f_m(\psi, \psi') = \alpha_m(\psi', \psi) f_m(\psi', \psi)$$

say where f_m is defined with respect to a common, symmetric measure on the product space.

Reversible Jump MCMC

Consider the forward move; in this case $f_m(\psi, \psi')$ is given by

$$f_m(\psi, \psi') = \widetilde{\pi}_m(\psi) q_m(\psi, \psi') = \pi_n(M, \theta) p_U(\mathbf{u}) \overrightarrow{r}_m.$$

For the reverse move, to preserve symmetry, we must set

$$f_m(\psi', \psi) = \widetilde{\pi}_m(\psi') q_m(\psi', \psi) = \pi_n(M', \theta') p_V(\mathbf{v}) \left| \frac{\partial(\theta', \mathbf{v})}{\partial(\theta, \mathbf{u})} \right| \overleftarrow{r}_m$$

where the term

$$\left| \frac{\partial(\theta', \mathbf{v})}{\partial(\theta, \mathbf{u})} \right| = \left| \frac{\partial \mathbf{g}(\mathbf{t}_1, \mathbf{t}_2)}{\partial(\theta, \mathbf{u})} \right|_{\mathbf{t}_1 = \theta', \mathbf{t}_2 = \mathbf{v}}$$

is the Jacobian for the 1-1 mapping $\mathbf{g} : (\theta, \mathbf{u}) \mapsto (\theta', \mathbf{v})$.

This term arises as in the augmented posterior, we have

$$\widetilde{\pi}(M', \theta', \mathbf{v}) = \widetilde{\pi}(M, \mathbf{h}(\theta', \mathbf{v}))|J(\theta', \mathbf{v})| = \widetilde{\pi}(M, \theta, \mathbf{u}))|J(\theta, \mathbf{u})|^{-1}$$

under the bijection; here

$$|J(\theta', \mathbf{v})| = \left|\frac{\partial(\theta, \mathbf{u})}{\partial(\theta', \mathbf{v})}\right| = \left|\frac{\partial(\theta', \mathbf{v})}{\partial(\theta, \mathbf{u})}\right|^{-1} = |J(\theta, \mathbf{u})|^{-1}$$

is the Jacobian.

In the expressions above, the two terms $\overrightarrow{r}_m$ and $\overleftarrow{r}_m$ represent the probabilities of choosing to make move m (from M to M') and the probability of the reverse move (from M' to M).

Thus, the acceptance probability for move type m is

$$\alpha_m((M,\theta),(M',\theta')) = \min\left\{1, \frac{\pi_n(M',\theta')p_V(\mathbf{v})\overleftarrow{r}_m}{\pi_n(M,\theta)p_U(\mathbf{u})\overrightarrow{r}_m}\left|\frac{\partial(\theta',\mathbf{v})}{\partial(\theta,\mathbf{u})}\right|\right\}.$$

Reversible Jump MCMC

These general calculations simplify if one of $\mathbf{u}$ or $\mathbf{v}$ is a null vector. If $\dim(\theta) < \dim(\theta')$, and the proposed move attempts to increase the dimension of the current model, then only the augmenting variables $\mathbf{u}$ are needed to match dimension, that is, the bijection can be constructed by setting $\theta' = \mathbf{g}(\theta, \mathbf{v})$. In this case

$$\alpha_m((M,\theta),(M',\theta')) = \min\left\{1, \frac{\pi_n(M',\theta')\overleftarrow{r}_m}{\pi_n(M,\theta)p_U(\mathbf{u})\overrightarrow{r}_m}\left|\frac{\partial(\theta')}{\partial(\theta,\mathbf{u})}\right|\right\}.$$

Conversely, if $\dim(\theta) > \dim(\theta')$, then only the augmenting variables $\mathbf{v}$ are needed, and

$$\alpha_m((M,\theta),(M',\theta')) = \min\left\{1, \frac{\pi_n(M',\theta')p_V(\mathbf{v})\overleftarrow{r}_m}{\pi_n(M,\theta)\overrightarrow{r}_m}\left|\frac{\partial(\theta',\mathbf{v})}{\partial(\theta)}\right|\right\}.$$

Notes:

1. The proposal mechanisms can be generalized by allowing the generation for $\mathbf{u}$ or $\mathbf{v}$ to depend on the values of θ or θ' respectively. Under this generalization, $p_U(\mathbf{u})$ and $p_V(\mathbf{v})$ are replaced by $p_U(\mathbf{u}|\theta)$ and $p_V(\mathbf{v}|\theta')$ in the acceptance probabilities. This merely corresponds to an alternative construction of the augmented posterior $\widetilde{\pi}$.

Reversible Jump MCMC

2. Each move type m comprises a pair of moves operating in each direction between models M and M'.
   - if move m is a move from M to M', constructed by generation of augmenting variables $\mathbf{u}$ and transformation, there should exist in our collection of potential moves the reverse move, indexed m', say, which utilizes the augmenting variables $\mathbf{v}$, the two moves being selected with probabilities $\overrightarrow{r}_m$ and $\overleftarrow{r}_m$

Reversible Jump MCMC

3. In some situations, the reversible jump acceptance probability simplifies even further.
   - In a move that increases dimension by d_U through the variables $\mathbf{u}$, it may be feasible to set the new parameters precisely equal to $\mathbf{u}$, that is, the proposed parameter vector θ' is formed by concatenating θ and $\mathbf{u}$. In this case, the Jacobian of the transformation is 1.

4. It may be possible to generate $\mathbf{u}$ from a prior distribution, which facilitates cancellation in the Hastings ratio.

Prior specification

Let $k = 1, 2, \ldots$ index models $M_1, M_2, \ldots$ under consideration, where k represents the dimension of $\theta_k = (\theta_{k1}, \ldots, \theta_{kk})$. Suppose that the model specification is such that identical, independent priors are used for the components of the parameter vector

$$\pi_0(\theta_k) = \prod_{l=1}^{k} \pi_0(\theta_{kl}).$$

For a proposed move from M_j to M_k with $j < k$, suppose that the elements of $\mathbf{u}$ are generated independently from p_0, with

$$p_U(\mathbf{u}) = \prod_{l=1}^{k-j} \pi_0(u_l),$$

say, and suppose that $\theta_k = (\theta_j, \mathbf{u})$.

Prior specification

Then the acceptance probability $\alpha_j \equiv \alpha_j((M_j, \theta_j), (M_k, \theta_k))$ is

$$\begin{aligned}
\alpha_j &= \min\left\{1, \frac{\pi_n(M_k, \theta_k)\overleftarrow{r}_j}{\pi_n(M_j, \theta_j)p_U(\mathbf{u})\overrightarrow{r}_j}\left|\frac{\partial(\theta_k)}{\partial(\theta, \mathbf{u})}\right|\right\} \\
&= \min\left\{1, \frac{\mathcal{L}_n^{M_k}(\theta_k)\pi_0(\theta_k|M_k)\pi_0(M_k)\overleftarrow{r}_j}{\mathcal{L}_n^{M_j}(\theta_j)\pi_0(\theta_j|M_j)\pi_0(M_j)p_U(\mathbf{u})\overrightarrow{r}_j}\left|\frac{\partial(\theta_k)}{\partial(\theta, \mathbf{u})}\right|\right\} \\
&= \min\left\{1, \frac{\mathcal{L}_n^{M_k}(\theta_k)\pi_0(M_k)\overleftarrow{r}_j}{\mathcal{L}_n^{M_j}(\theta_j)\pi_0(M_j)\overrightarrow{r}_j}\right\}
\end{aligned}$$

as $\pi_0(\theta_k|M_k)$ can be written

$$\prod_{l=1}^{k}\pi_0(\theta_{kl}) = \left\{\prod_{l=1}^{j}\pi_0(\theta_{jl})\right\}\left\{\prod_{l=1}^{k-j}\pi_0(u_l)\right\} = \pi_0(\theta_j|M_j)p_U(\mathbf{u})$$

Prior specification

For the reverse move that attempts to decrease the model complexity by moving from model k to model j, the deterministic proposal that sets the last $k - j$ components of θ_k to zero is used.

The approach described in this example can be useful in some applications, but can also lead to low acceptance rates.

Example

Consider two models M_1 and M_2 with parameters $\theta^{(1)} = \theta_1$ and $\theta^{(2)} = (\theta_{21}, \theta_{22})$, with all parameters taking values on $\mathbb{R}$. We consider four move types:

1. $m = 1$: move *within* Model M_1,
2. $m = 2$: move *within* Model M_2,
3. $m = 3$: move *from* Model M_1 *to* Model M_2,
4. $m = 4$: move *from* Model M_2 *to* Model M_1.

For the within model moves, standard MH acceptance calculations proceed as usual; Moves 3 and 4 are a forward/reverse move pair.

Clearly, if the current state of the chain is in M_1, only moves 1 or 3 can be selected, and similarly for M_2.

Example

We only need to consider the relative magnitudes of the reversible jump moves selection probabilities; for example, if the probability of selecting move 3 is 0.3 and the probability of selecting move 4 is 0.1, then

$$\frac{\overleftarrow{r}_3}{\overrightarrow{r}_3} = \frac{0.1}{0.3} \qquad \frac{\overleftarrow{r}_4}{\overrightarrow{r}_4} = \frac{0.3}{0.1}$$

are the ratios that enter into the acceptance probability calculations for move types 3 and 4 respectively.

Example

When the current state of the chain is in M_1, to propose a move to M_2 after selecting move 3, we must introduce a single random variate u; suppose that $u \sim Normal(0,1)$, and let

$$\theta_{21} = \theta_1 + u \qquad \theta_{22} = \theta_1 - u \qquad \Longleftrightarrow \qquad \theta_1 = \frac{\theta_{21} + \theta_{22}}{2}.$$

In this case the acceptance probability takes the form

$$\alpha_3 = \min\left\{1, \frac{\pi_n(M_2, (\theta_{21}, \theta_{22}))}{\pi_n(M_1, \theta_1)\phi(u)} \frac{\overleftarrow{r}_3}{\overrightarrow{r}_3} \left|\frac{\partial(\theta_{21}, \theta_{22})}{\partial(\theta_1, u)}\right|\right\}.$$

where $\phi(.)$ is the standard normal pdf, and

$$\left|\frac{\partial(\theta_{21}, \theta_{22})}{\partial(\theta_1, u)}\right| = \begin{vmatrix} \frac{\partial\theta_{21}}{\partial\theta_1} & \frac{\partial\theta_{21}}{\partial u} \\ \frac{\partial\theta_{22}}{\partial\theta_1} & \frac{\partial\theta_{22}}{\partial u} \end{vmatrix} = \begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix} = 2.$$

Example

For the reverse move 4, we have

$$\alpha_4 = \min\left\{1, \frac{\pi_n(M_1, \theta_1)\phi(u)}{\pi_n(M_2, (\theta_{21}, \theta_{22}))} \frac{\overleftarrow{r}_4}{\overrightarrow{r}_4} \left|\frac{\partial(\theta_1, u)}{\partial(\theta_{21}, \theta_{22})}\right|\right\}.$$

where

$$\left|\frac{\partial(\theta_1, u)}{\partial(\theta_{21}, \theta_{22})}\right| = \frac{1}{2}.$$

Pharmacokinetic example

Consider two competing pharmacokinetic (PK) models for response data $y(t)$ measured at different time points $t_1, \ldots, t_n$:

1. **Model M_1:** One compartment, elimination only;

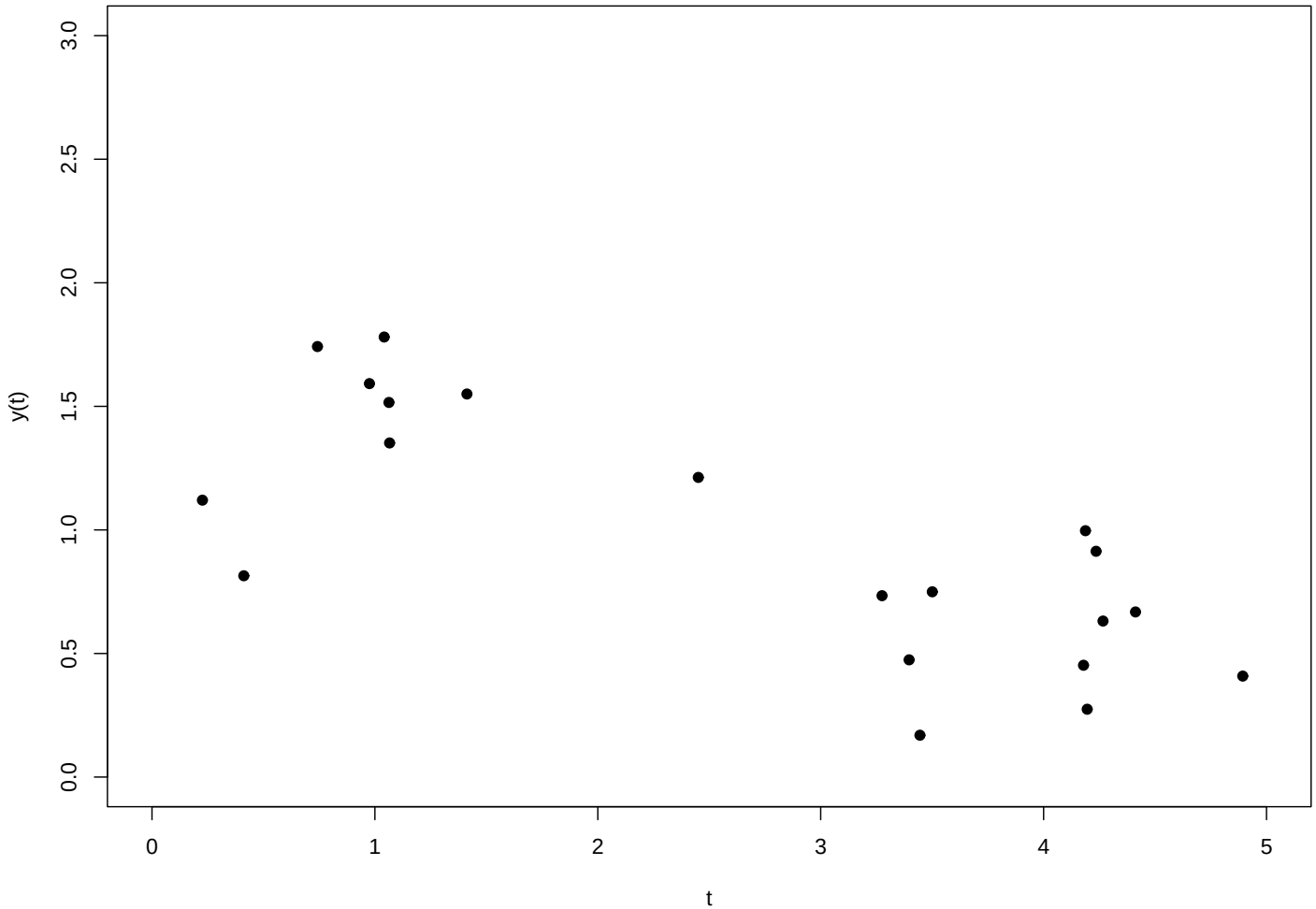
$$\mathbb{E}[Y(t)] = A_1 \exp\{-\lambda_1 t\} \qquad t \geqslant 0$$

2. **Model M_2:** One compartment, absorption and elimination;

$$\mathbb{E}[Y(t)] = A_2 \left(\exp\{-\lambda_{21} t\} - \exp\{-(\lambda_{21} + \lambda_{22})t\}\right) \qquad t \geqslant 0$$

where (A_1, λ_1) and $(A_2, \lambda_{21}, \lambda_{22})$ are positive parameters. Under an assumption of additive, heteroscedastic Normal errors, we have two competing explanations for the observed data; both models can be fitted using ordinary least-square

Pharmacokinetic example



Pharmacokinetic example

These models are nested, but the nesting structure is complicated, as we require $\lambda_{22} \longrightarrow \infty$, that is, a boundary point in the parameter space, which leads to non-regular frequentist asymptotic theory.

Bayes factors can be used:

$$BF(M_1, M_2) = \frac{\int \mathcal{L}_n^{M_1}(\theta_1)\pi_0(\theta_1|M_1)\, d\theta_1}{\int \mathcal{L}_n^{M_2}(\theta_2)\pi_0(\theta_2|M_2)\, d\theta_2}$$

but this requires numerical integration.

Pharmacokinetic example

We consider a reversible jump MCMC solution. First, we use a log-scale parameterization and set

$$\theta_1 = (\log A_1, \log \lambda_1) \qquad \theta_2 = (\log A_2, \log \lambda_{21}, \log \lambda_{22}).$$

We place equal prior probabilities on M_1 and M_2, and then place independent $N(0, \tau^2)$ priors on the components of θ_1 and θ_2.

The prior on the residual error variance σ^2 is Inverse Gamma with parameters 20 and 8.

Pharmacokinetic example

The ML estimates $\widehat{\theta}_1$ and $\widehat{\theta}_2$ can be computed easily, as can the Hessian matrices $\widehat{\mathbf{I}}_1$ and $\widehat{\mathbf{I}}_2$; these likelihood-based results yield reasonable approximations to the posterior distributions to produce independence MH algorithms.

Specifically, at the ML estimates for σ under the two models, we may approximate the conditional posterior for θ by the Normal density

$$p(\theta_k|\widehat{\sigma}, \mathbf{y}) \doteq Normal(\widehat{\theta}_k, n^{-1}\widehat{\sigma}_k^2 \widehat{\mathbf{I}_k}^{-1}) \tag{1}$$

On fitting using ML, the estimates of σ under the two models are found to be quite similar ($M_1 : \widehat{\sigma}_1 = 0.252, M_2 : \widehat{\sigma}_2 = 0.329$).

Pharmacokinetic example

A reversible jump MCMC algorithm can be constructed as follows: we again consider four move types:

1. $m = 1$: move *within* M_1; update θ_1 from $\pi_n(\theta_1|M_1, \sigma)$
2. $m = 2$: move *within* M_1; update θ_2 from $\pi_n(\theta_2|M_2, \sigma)$
3. $m = 3$: move *from* M_1 *to* M_2; propose a new θ_2, and carry out an accept/reject step.
4. $m = 4$: move *from* Model M_2 *to* Model M_1; propose a new θ_1, and carry out an accept/reject step.

with the remaining parameter σ^2 being updated in a Gibbs sampler algorithm at each iteration.

Pharmacokinetic example

Moves $m = 3, 4$ are a forward/reverse move pair. For move 3, several options are available; for example, we could adopt the earlier strategy, and generate a new variate u from the prior for the additional parameter, and then merely use the mapping

$$(\theta_{11}, \theta_{12}, u) \longmapsto (\theta_{21} = \theta_{11}, \theta_{22} = \theta_{12}, \theta_{23} = u)$$

with reverse move setting $\theta_{23} = 0$.

This approach may be adequate, but more probably would not facilitate good mixing across the models.

Pharmacokinetic example

A perhaps better strategy is to consider a different augmentation, where we generate $\mathbf{u} = (u_1, u_2, u_3)$ from the model in (1) for $k = 2$, and map $(\theta_{11}, \theta_{12}, u_1, u_2, u_3)$ to

$$(\theta_{21} = u_1, \theta_{22} = u_2, \theta_{23} = u_3, v_1 = \theta_{11}, v_2 = \theta_{12})$$

with the paired reverse move being to generate $\mathbf{v} = (v_1, v_2)$ from the model in (1) for $k = 1$.

This guarantees that the proposed value θ_2 lies in a region with reasonably high posterior support under model M_2, although it does not guarantee that the move will be accepted with high probability.

Pharmacokinetic example

In the Hastings ratio, the Jacobian of the transformation is 1, and under equal probabilities of forward/reverse moves, we have that

$$\frac{\pi_n(M_2, \theta_2) p_V(v_1, v_2)}{\pi_n(M_1, \theta_1) p_U(u_1, u_2, u_3)}$$

can be written

$$\frac{\mathcal{L}_n^{M_2}(\theta_2) \left\{ \prod_{j=1}^{3} \phi(\theta_{2j}/\tau)/\tau \right\} \phi_2(\theta_{11}, \theta_{12}; \widehat{\theta}_1, \widehat{\mathbf{I}}_1)}{\mathcal{L}_n^{M_1}(\theta_1) \left\{ \prod_{j=1}^{2} \phi(\theta_{1j}/\tau)/\tau \right\} \phi_3(\theta_{21}, \theta_{22}, \theta_{23}; \widehat{\theta}_2, \widehat{\mathbf{I}}_2)}$$

where τ is the prior variance for the regression parameters.

Pharmacokinetic example

The logic of this construction is that numerically

$$\mathcal{L}_n^{M_1}(\theta_1) \doteq Normal_2(\widehat{\theta}_1, \widehat{\mathbf{I}}_1^{-1}) \qquad \mathcal{L}_n^{M_2}(\theta_2) \doteq Normal_3(\widehat{\theta}_2, \widehat{\mathbf{I}}_2^{-1}).$$

An approximation that incorporates the Normal asymptotic likelihood in equation (1) as well as the Normal prior distribution for the parameters can be constructed.

Pharmacokinetic example

The algorithm was run for 100000 iterations. In this run with $\tau = 4$, the chain spent about 66 % of the time in model M_1, indicating the posterior probabilities are

$$\pi_n(M_1) \simeq 0.66 \qquad \pi_n(M_2) \simeq 0.34.$$

The model posterior probabilities vary with the choice of τ; this is as expected, as the model probabilities are closely related to the marginal likelihood, or prior predictive distribution, which is the expected value of the likelihood for the observed data with respect to the prior distribution.

Pharmacokinetic example

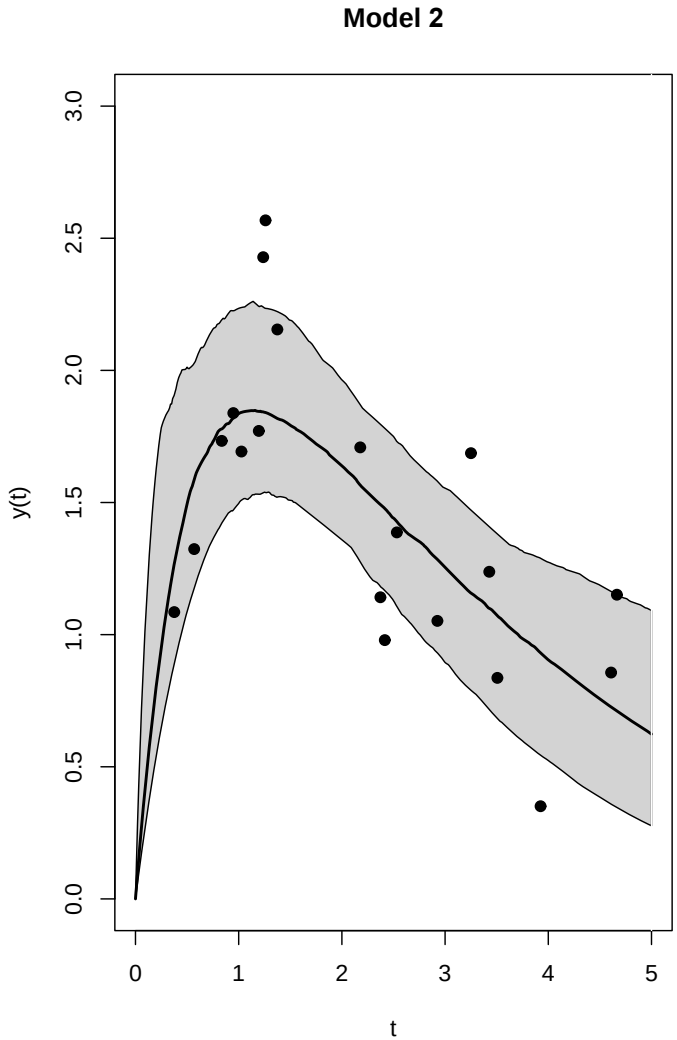
It is evident from the discussion that the prior specification acts as a penalty for complexity. For illustration, if $\tau = 1$, the model probabilities change to $(0.43, 0.57)$; if $\tau = 10$, the model probabilities are approximately $(0.80, 0.20)$.

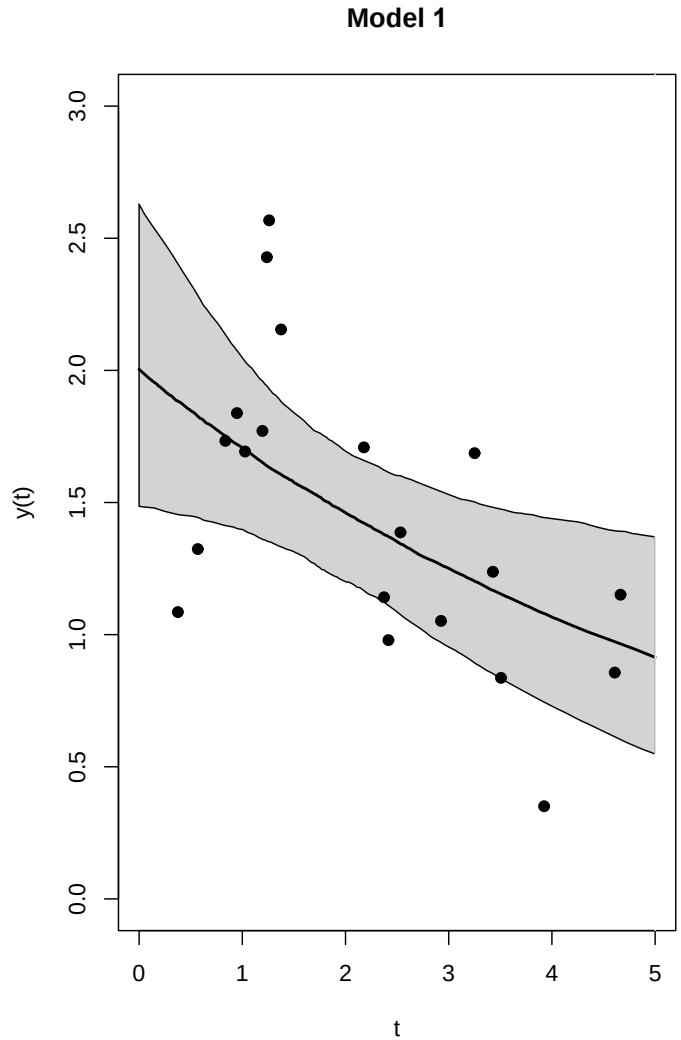
Pharmacokinetic example

Conditional on M_1 or M_2 being true, we can perform inference about the parameters of the two models, and also reconstruct estimates and posterior credible intervals for $\mathbb{E}[Y]$.

The figure below displays the reconstructed posterior intervals for the two models.

Pharmacokinetic example





Pharmacokinetic example

It is evident that both models are plausible explanations of the observed data, but offer potentially different predictions of future responses, especially near $t = 0$. Note that the possible nesting structure, incorporated by the assumption that $\lambda_{22} \longrightarrow \infty$, is not supported by the priors studied.

Finally, note that the BIC values for the two models, computed as

$$\text{BIC}_k = -2 \log \mathcal{L}_n^{M_k}(\widehat{\theta}_k) + d_k \log n$$

where d_k is the total number of parameters fitted, are 21.2953 and 13.5964 for models 1 and 2, indicating strong support for model 2.

Appendix: Trans-dimensional examples

Finite Mixture Model

The finite mixture model with K components has density

$$f(y|\omega, \theta, K) = \sum_{k=1}^{K} \omega_k f_k(y|\theta_k)$$

where θ_k are the parameters for component density k, and

$$0 < \omega_k < 1, \forall k \qquad \& \qquad \sum_{k=1}^{K} \omega_k = 1.$$

Finite Mixture Model

Most typically the component densities are from the same parametric location-scale family (say Gaussian) which differ in location and scale.

For exchangeable data $(y_1, \ldots, y_n)$ from this model, the likelihood arising from this model does not permit analytical calculation of the posterior distribution, so MCMC is commonly used.

Finite Mixture Model

If K is known, MCMC is typically implemented using auxiliary variables $z_1, \ldots, z_n$, and a completed data model

$$f(y, z; \omega, \theta, K) = \omega_z f_z(y|\theta_z) \qquad z \in \{1, 2, \ldots, K\}$$

and a Gibbs sampler strategy updating the $\mathbf{z}$ and (ω, θ) from their full conditional posterior distributions.

Conditional on (ω, θ), the posterior distribution for each z_i is a discrete distribution on $\{1, 2, \ldots, K\}$, whereas conditional on the $\mathbf{z}$, the posterior for the mixture weights ω is a density on the K dimensional simplex, and the posterior for the components of θ_k proceeds using only those y_i for which $z_i = k$, for $k = 1, \ldots, K$.

Finite Mixture Model

If K is also treated as an unknown parameter, then transdimensional MCMC is also needed. A discrete prior on K, typically on some finite set, completes the posterior specification, and yields a posterior distribution $\pi_n\left(K, \theta^{(K)}, \omega^{(K)}\right)$.

Finite Mixture Model

Here, dimension changing moves correspond to changing the value of K, and in order to construct an efficient and tunable MCMC algorithm, moves which peturb K by ± 1 are typically considered.

- For example, moves of type $K \longrightarrow K+1$ are termed “birth" moves, and $K \longrightarrow K-1$ are termed “death" moves.
- The forward/reverse move pairs are then births from K and deaths from $K+1$.

Finite Mixture Model

We consider a Normal mixture model for simplicity.

- If y is one dimensional, each component density has two parameters, so a birth/death pair requires the introduction of three latent variables $\mathbf{u}$, a new mean and variance, and also a parameter to introduce a new component weight.
- If y is d-dimensional, $d + d(d+1)/2 + 1$ new scalar parameters are needed, that is, a new mean and covariance matrix, and a new component weight parameter.

Finite Mixture Model

- **New location/scale parameters:** Either these new parameters are generated independently of the current parameters from proposal distribution, or a subset of the current components are selected and used to generate a new additional component.
- **New weight parameters:** A random split or rescaling of the currently existing weights is used.

Finite Mixture Model

In typical mixture applications, inference about the parameter K may be of primary interest in a given application; perhaps more commonly the mixture model offers flexible modelling in the presence of heterogeneity.

The Normal mixture case is the most commonly studied, but other mixtures of other parametric models have also been used.

Regression

Suppose the regression relationship between a scalar covariate x and scalar covariate y

$$y = g(x) + \epsilon$$

is the focus of interest, for independent homoscedastic Normal errors $\{\epsilon\}$. Typically, we seek the prediction $\widehat{y} = \widehat{g}(x)$.

Suppose we seek to approximate g locally on a finite domain $\mathcal{X}$ by a series of step functions:

$$g_K(\mathbf{x}; \boldsymbol{\beta}_K, \kappa_K) = \sum_{k=1}^{K} \beta_{Kk} \mathbb{1}_{B_k}(\mathbf{x})$$

where $(B_1, \ldots, B_K)$ form a partition of $\mathcal{X}$ defined by a series of knots $\kappa_0, \kappa_1, \ldots, \kappa_{K-1}, \kappa_K$ so that $B_k \equiv [\kappa_{k-1}, \kappa_k)$. Without loss of generality, assume $\mathcal{X} \equiv [0, 1)$, with $\kappa_0 = 0$, $\kappa_K = 1$. In this step-function approximation, $\boldsymbol{\beta}_K = (\beta_{K1}, \ldots, \beta_{KK})$ are the parameters defining the piecewise constant levels of the function.

Regression

The likelihood for data $(x_i, y_i), i = 1, \ldots, n$ is

$$\mathcal{L}_n^{M_K}(\boldsymbol{\beta}_K, \sigma, \kappa_K) = \prod_{i=1}^{n} \phi\left(\frac{y_i - g_K(x_i; \boldsymbol{\beta}_K, \kappa_K)}{\sigma}\right)$$

Typically, within model M_K, a conjugate prior specification on $(\boldsymbol{\beta}_K, \sigma)$ is used so that the marginal likelihood $\mathcal{L}_n^{M_K}(\kappa_K)$ can be computed analytically by integrating out $(\boldsymbol{\beta}_K, \sigma)$. The conjugate prior takes the form

$$\pi_0(\boldsymbol{\beta}_k, \sigma) = \pi_0(\boldsymbol{\beta}_K|\sigma)\pi_0(\sigma)$$

and $\pi_0(\boldsymbol{\beta}_K|\sigma)$ is some multivariate Normal density.

Regression

A reversible jump MCMC algorithm can be constructed to explore the posterior $\pi_n(M_K, \kappa_K)$ which, under the conjugate specification for $(\boldsymbol{\beta}_K, \sigma)$, depends on the marginal likelihood and the prior $\pi_0(M_K, \kappa_K)$.

One approach to the specification of this prior is to assume a Poisson process model for the change-in-level or jump locations of the step function.

- $\pi_0(M_K)$ to be a Poisson distribution with rate parameter λ,
- within model M_K, κ_K are the order statistics derived from a Uniform random sample on $(0, 1)$, so that

$$\pi_0(\kappa_{K1}, \ldots, \kappa_{K,K-1} | M_K) = (K-1)!$$

for $0 < \kappa_1 < \cdots < \kappa_{K,K-1} < 1$.

Regression

Dimension-changing moves in this model correspond to the addition or removal of a knot or knots.

- a new knot can be proposed uniformly on (0,1), or uniformly in an interval selected in light of the current knot positions. In the former case, the reverse move corresponds to removing a knot uniformly at random from the current collection.
- Model complexity in this model is controlled by the Poisson model hyperparameter λ, and the covariance structure in the prior model for β_K.

Examples

Example: Reversible jump MCMC

See `knitr 17`

Example: Galaxy data

See `knitr 18`

Example: Model selection in regression

See `knitr 19`

Bayesian Non-Parametrics: Motivation

- Parametric models are restrictive;
- We often meet data sets that exhibit non-standard heterogeneity;
- Likelihood-based inference is more straoghtforward than moment based estimation (such as GEE, GMM)
- Because we can report a more complete summary of the inference results, and perform prediction.

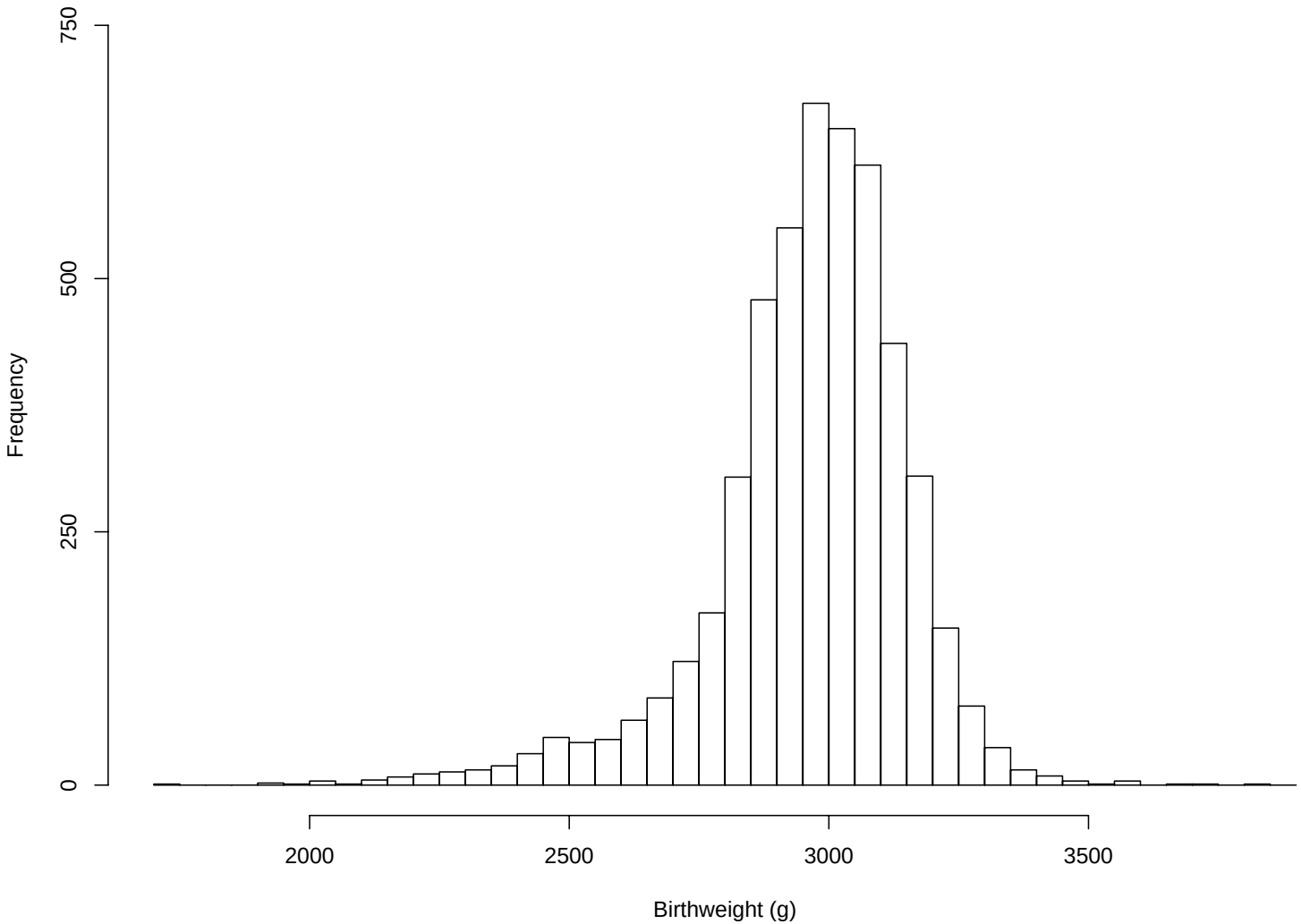
Bayesian Non-Parametrics: Motivation

- Regression Problems
- Survival Analysis
- Clustering
- Measurement Error/Errors-in-Variables Problems
- Density Estimation

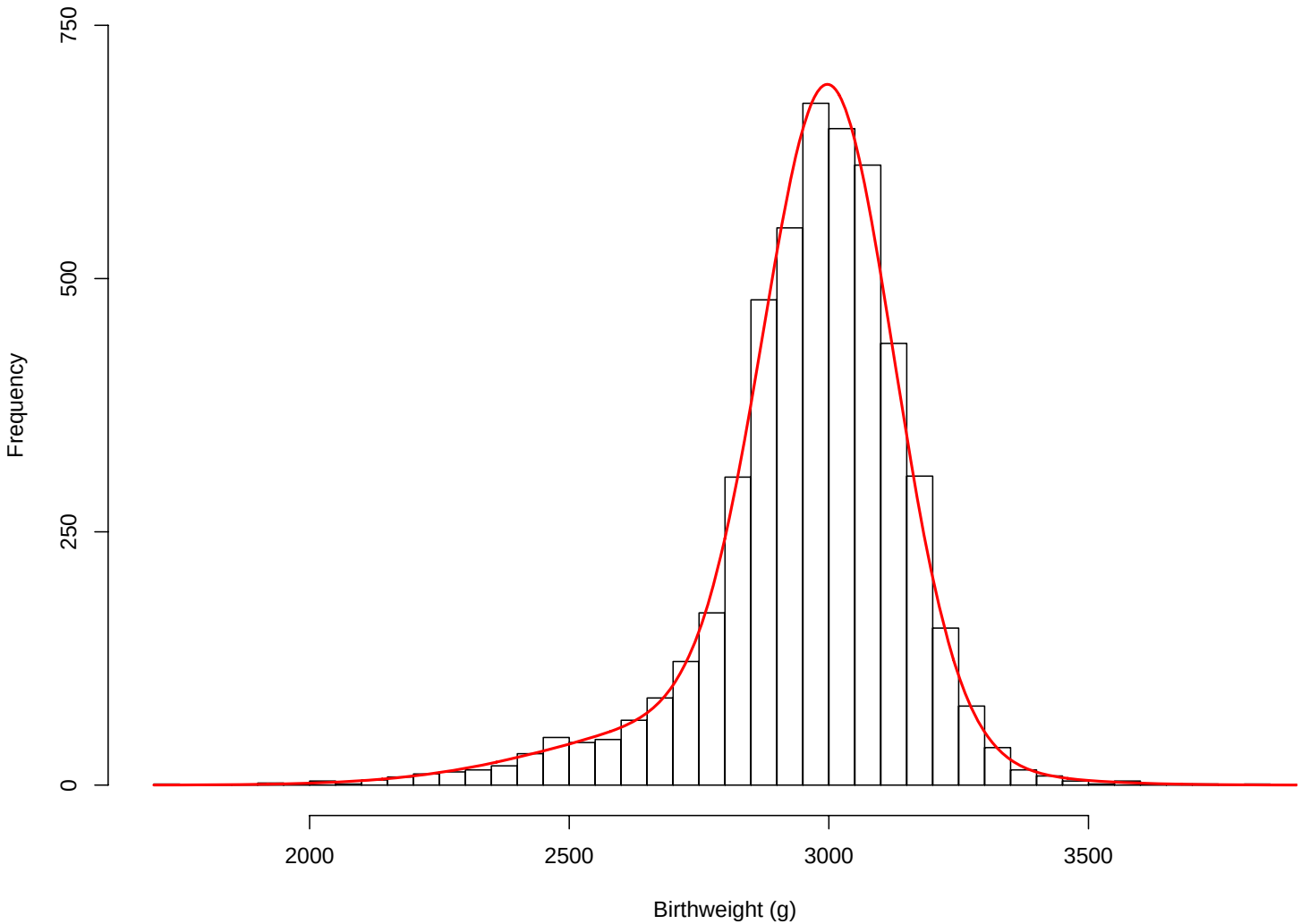
that is, Bayesian nonparametric methods can be used where parametric models, or non-parametric maximum likelihood are used.

Some applications in hypothesis testing.

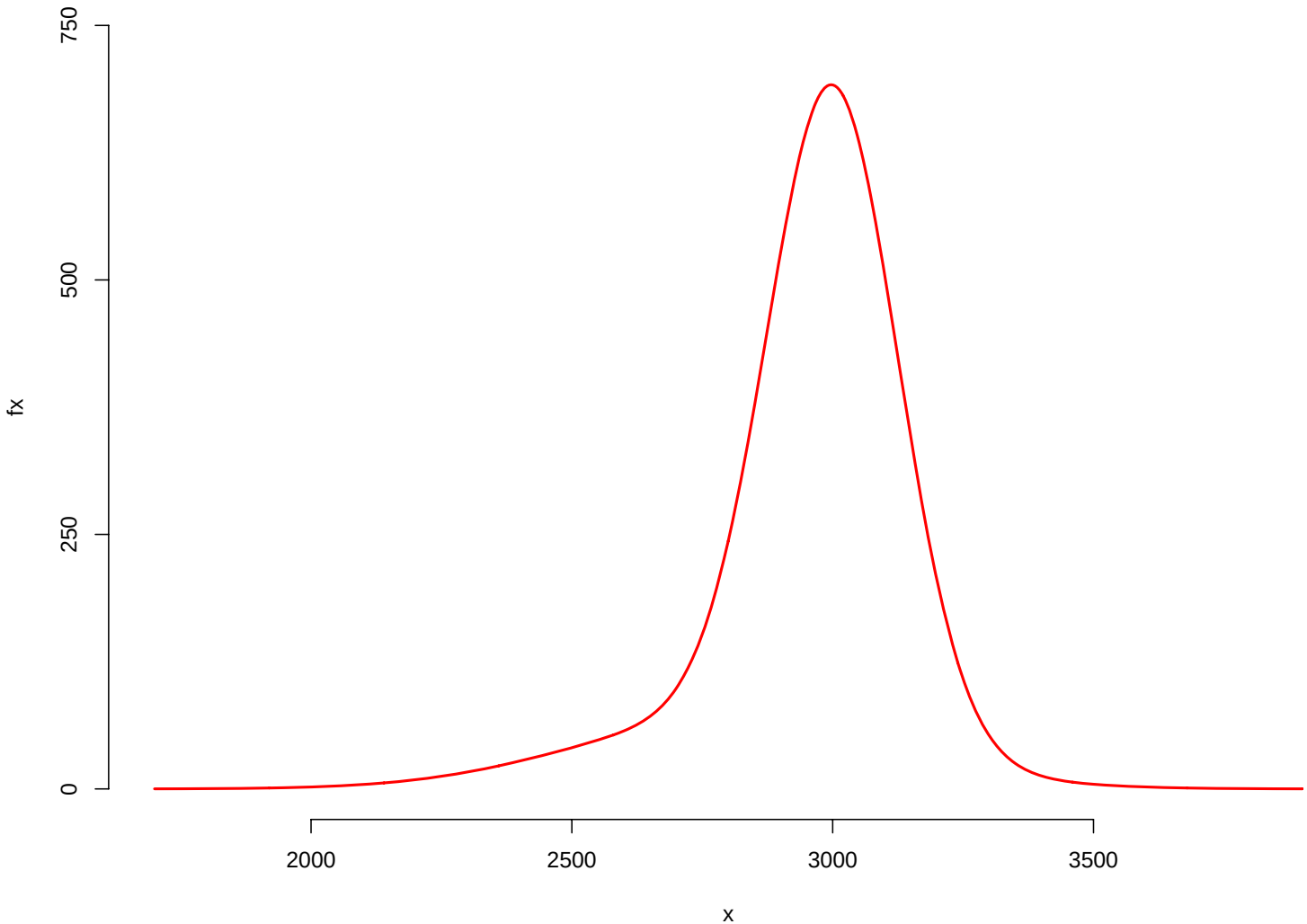
Hypothetical Example: Birthweight data



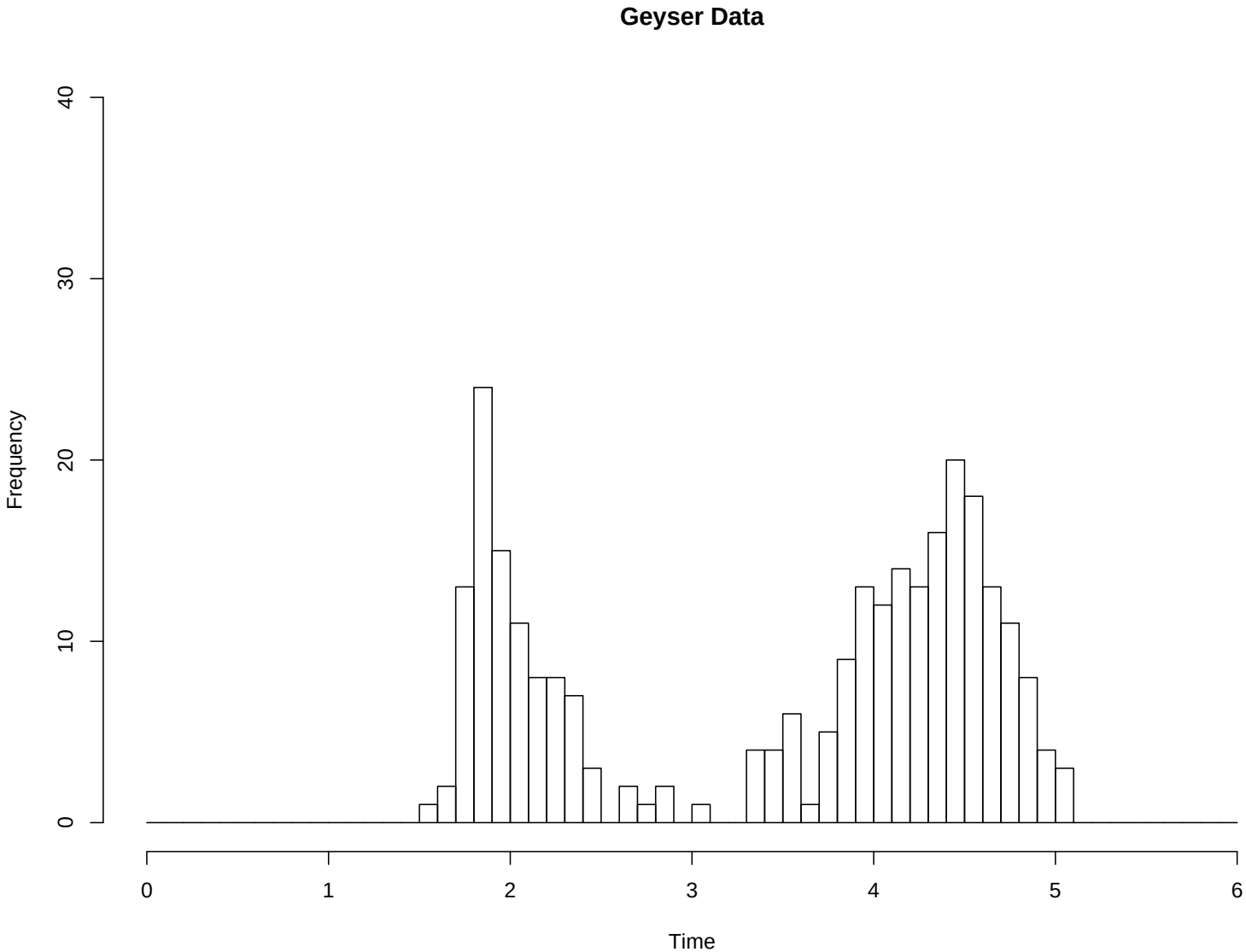
Hypothetical Example: Birthweight data



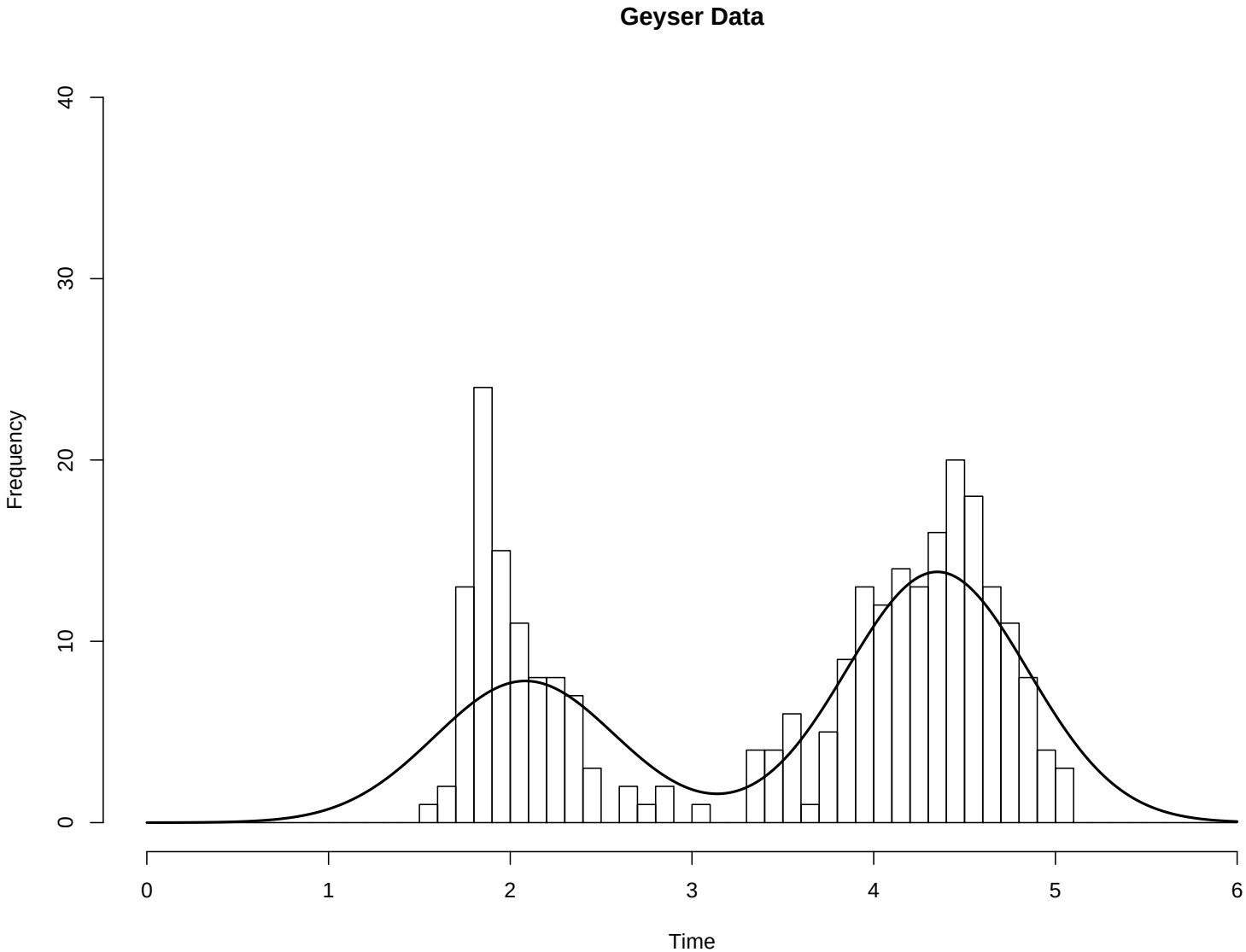
Hypothetical Example: Birthweight data



Example: Old Faithful Data



Example: Old Faithful Data



Bayesian Inference

In Bayesian statistical inference, we compute and summarize the posterior distribution of the unknown parameters in the probability model, in light of observed data.

- In Bayesian *parametric* inference, the parameter is the usual θ, λ, μ say that appears in the presumed (conditional) data generating model.
- In Bayesian *non-parametric* inference, the parameter is the *distribution* from which the data are drawn.

As part of Bayesian inference, we need to specify the *prior* probability distribution for these parameters.

Random Distributions

How do we construct a “random" distribution suitable for non-parametric prior modelling ? In the univariate case:

(I) Choose some locations on the real line

$$x_1, x_2, x_3, \ldots$$

(II) Place a mass of probability at each location

$$p_1, p_2, p_3, \ldots$$

where the probabilities sum to 1.

Then the function

$$f(x) = \sum_{i=1}^{\infty} p_i \delta_{x_i}(x)$$

is a discrete probability distribution on the set $\{x_1, x_2, x_3, \ldots\}$.

Random Distributions

How do we make this random ?

(I) Choose the locations *randomly* (and independently) from some distribution G_X; denote them $x_1, x_2, x_3, \ldots$.

(II) Choose the probabilities *randomly* in such a way such that they sum to 1; denote them $\pi_1, \pi_2, \pi_3, \ldots$.

Then

$$\widetilde{f}(x) = \sum_{i=1}^{\infty} \pi_i \delta_{x_i}(x)$$

is a random mass function.

Random Distributions

If the number of locations is *finite*, equal to K say, then we can generate the probabilities $\pi_1, \ldots, \pi_K$ from a *Dirichlet* distribution

$$Dirichlet(K; \alpha_1, \ldots, \alpha_{K+1})$$

where the αs are fixed constants.

In this case, the x_is are fixed, or the x_is simulated from G_X.

Random Distributions

If the number of locations is *infinite*, it is helpful to order the π_i in descending order

$$\pi_1 \geqslant \pi_2 \geqslant \pi_3 \geqslant \cdots$$

so that eventually the terms in the infinite sum become effectively zero, so that the truncation

$$\widetilde{f}(x) \backsimeq \widetilde{f}_N(x) = \sum_{i=1}^{N} \pi_i \delta_{x_i}(x)$$

can be computed.

Random Distributions

For example, consider for $\alpha > 0$

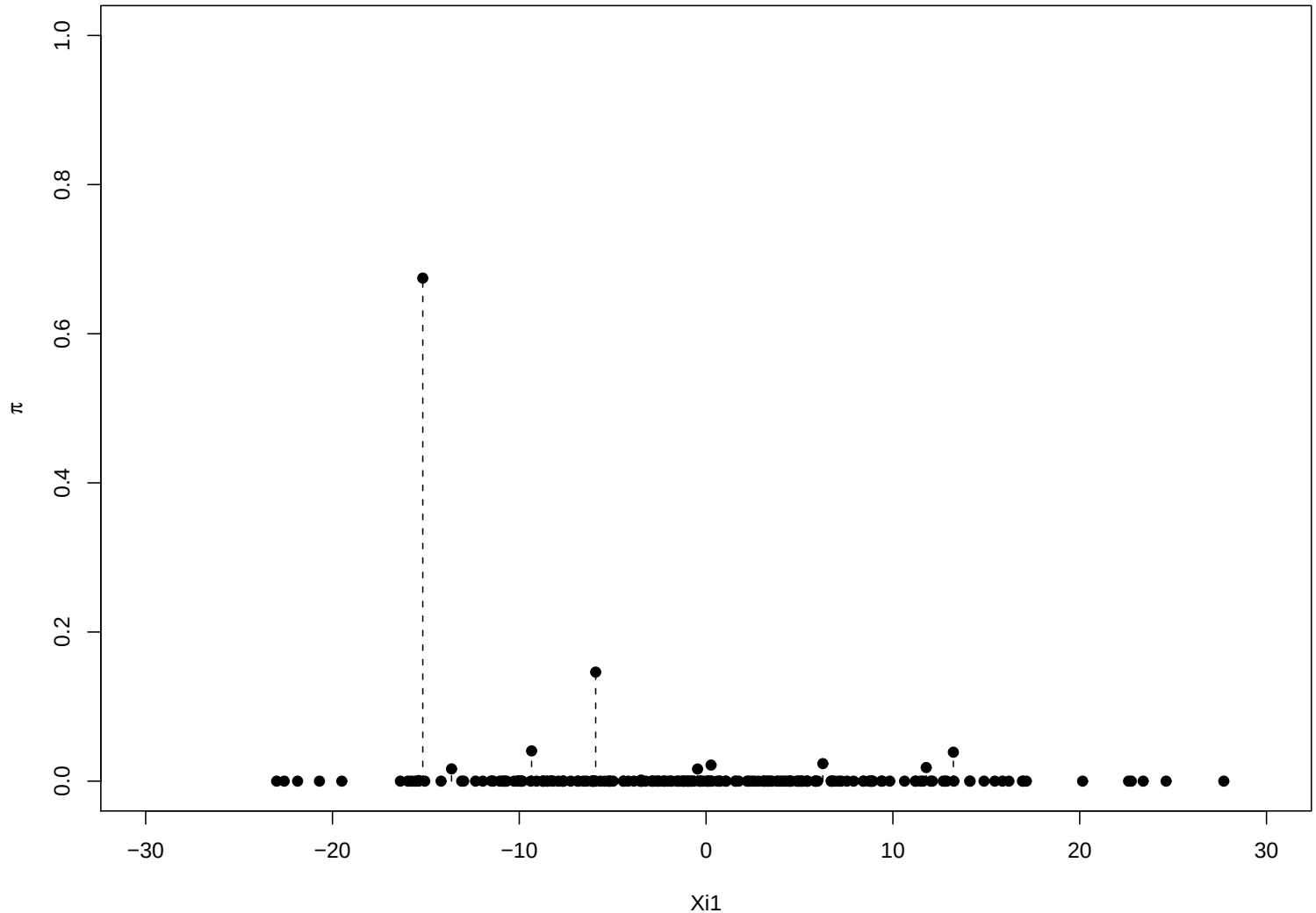
$$V_1, V_2, V_3, \ldots \sim Beta(1, \alpha)$$

independently distributed, and

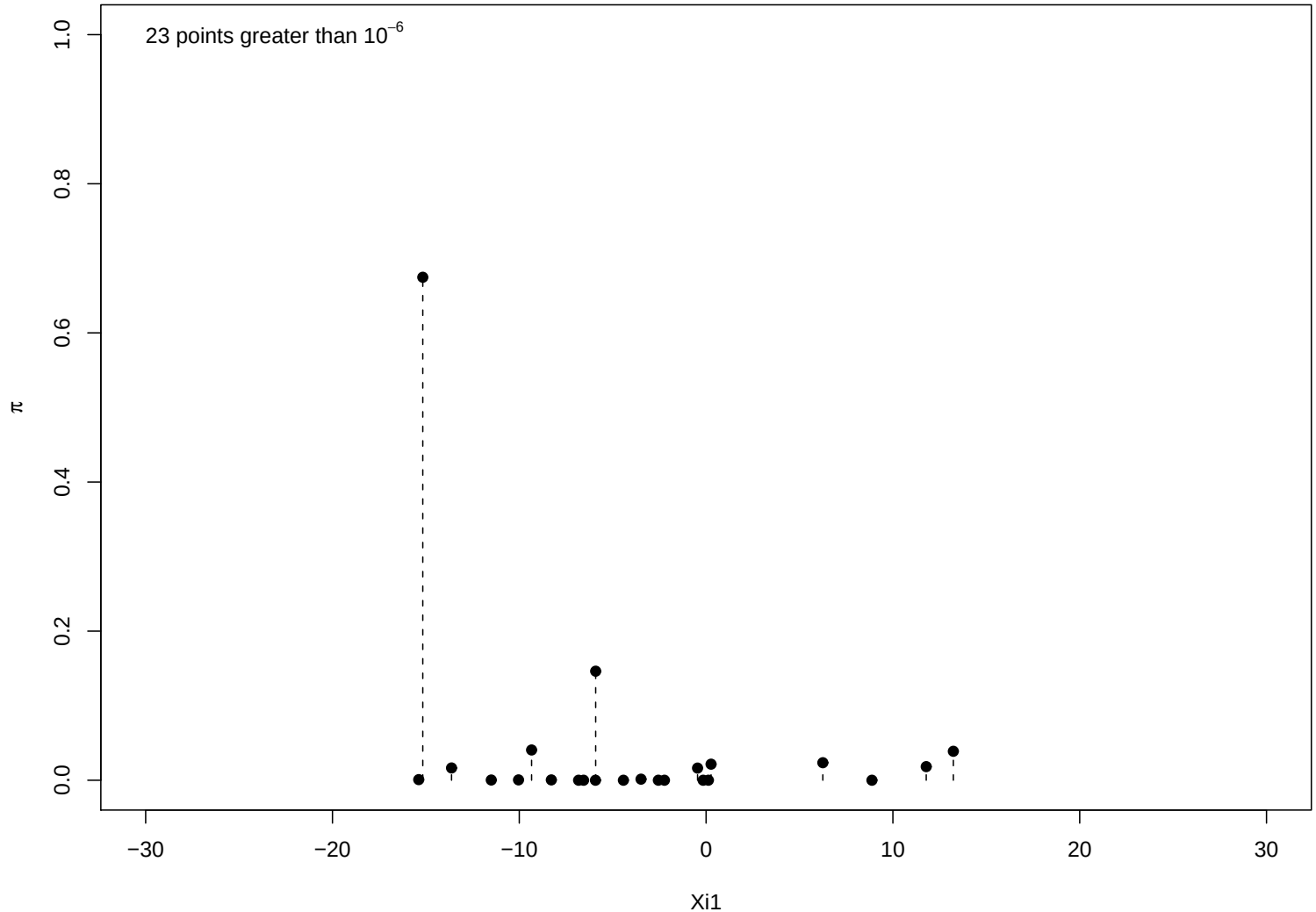
$$\begin{aligned} \pi_1 &= V_1 \\ \pi_2 &= (1 - V_1)V_2 \\ \pi_3 &= (1 - V_1)(1 - V_2)V_3 \\ &\vdots \quad \vdots \quad \vdots \end{aligned}$$

The π_i sequence generated in this way are *decreasing in expectation*.

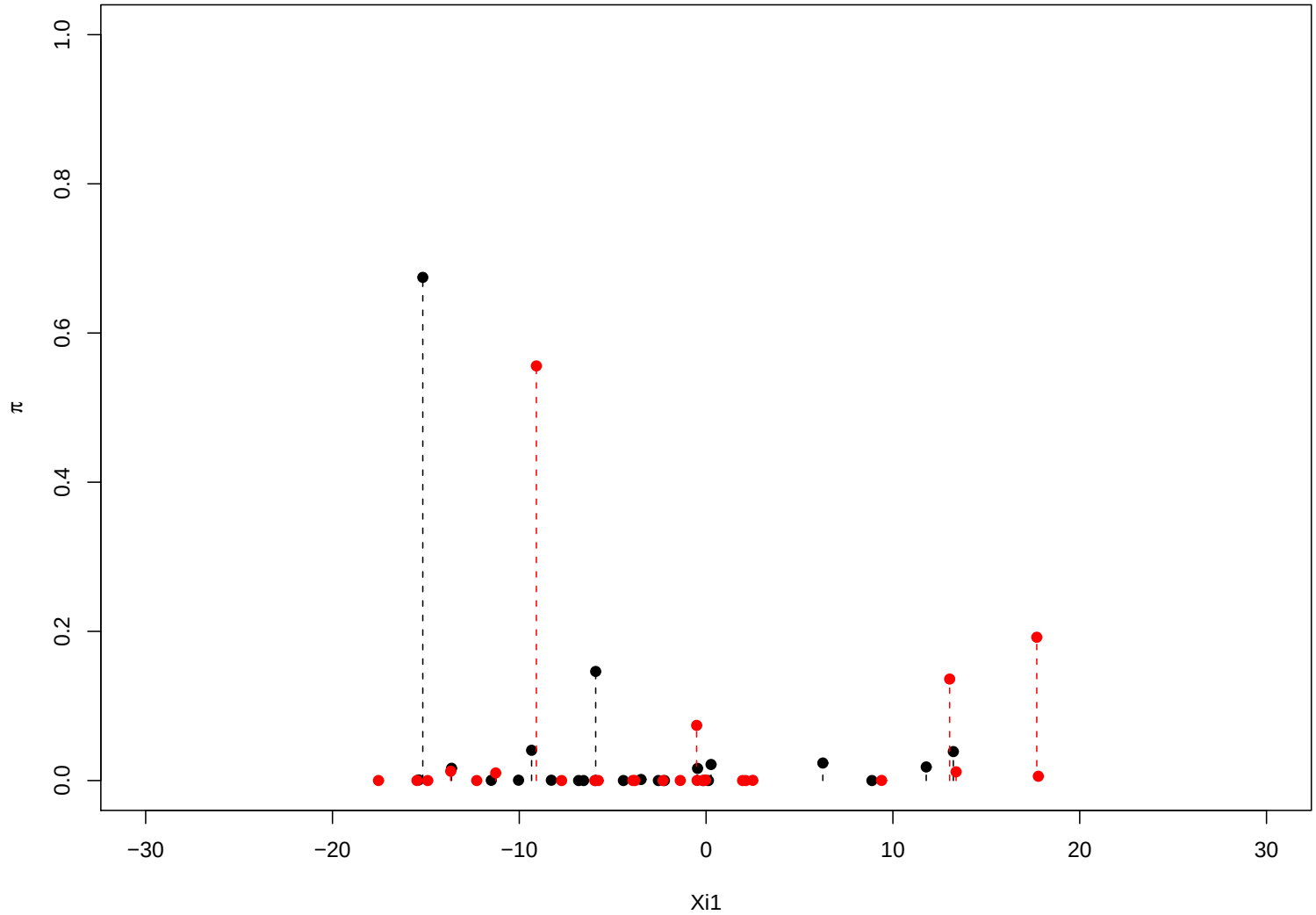
Simulation: $\alpha = 2$



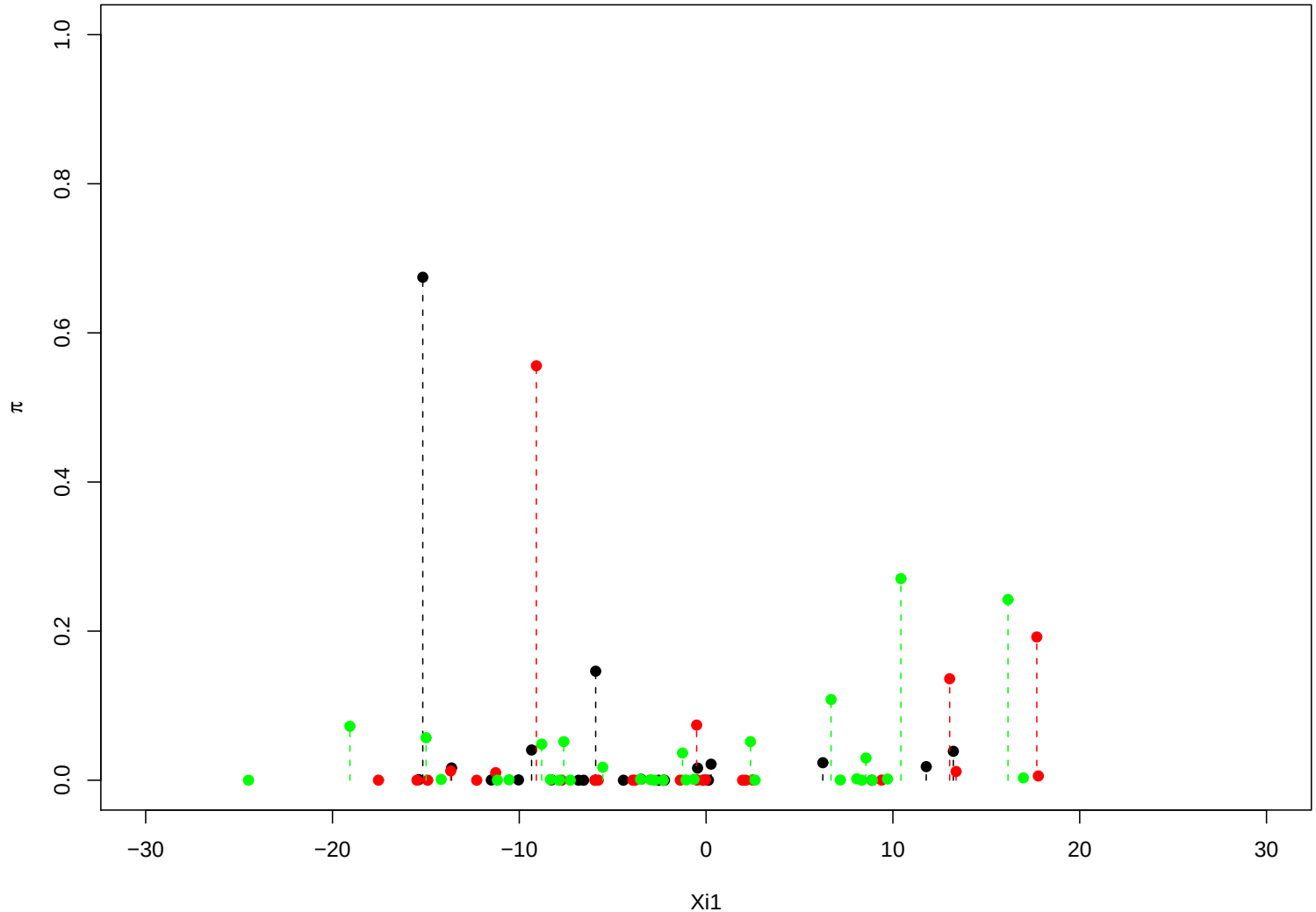
Simulation: $\alpha = 2$, thresholded



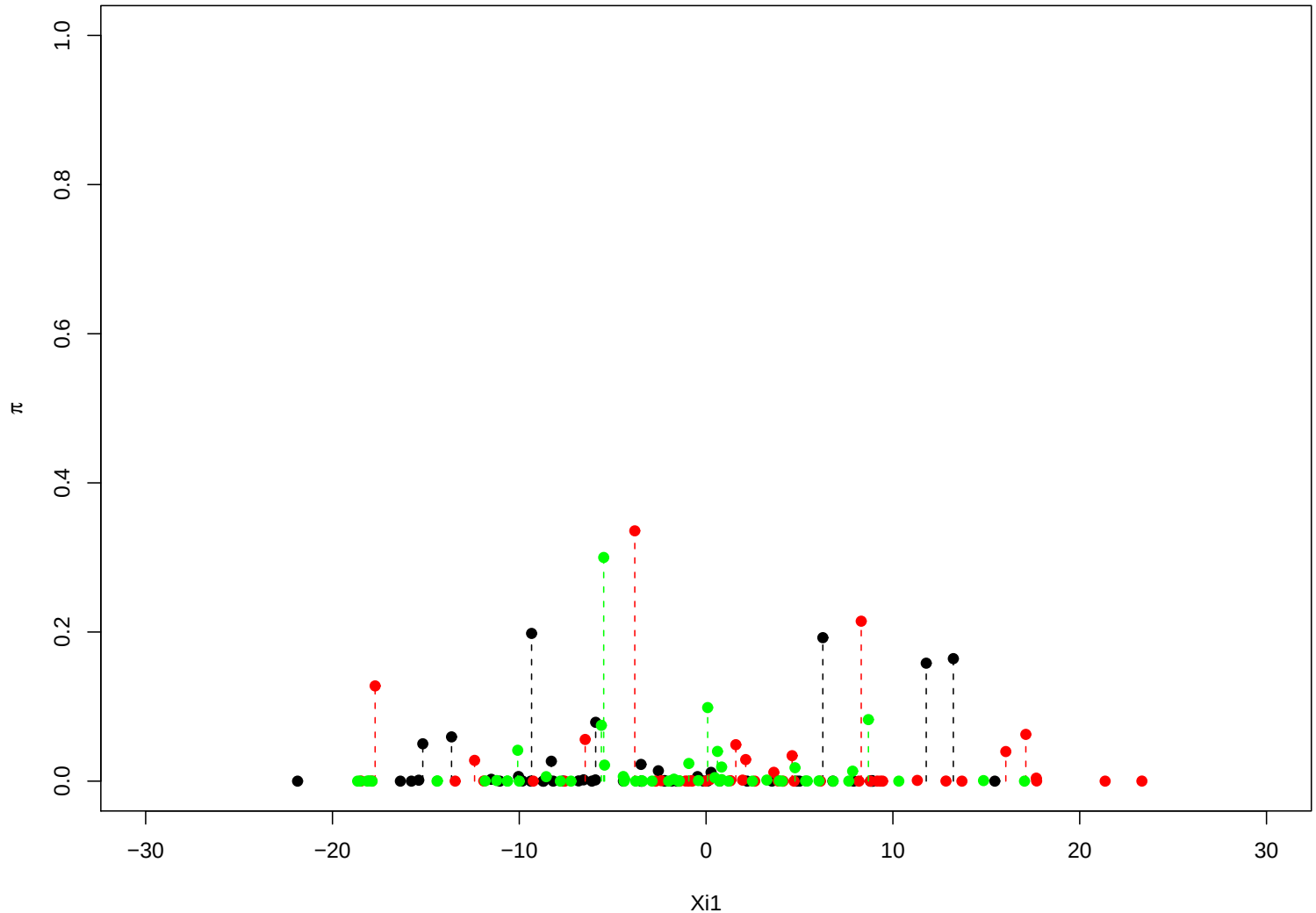
Simulation: $\alpha = 2$, thresholded



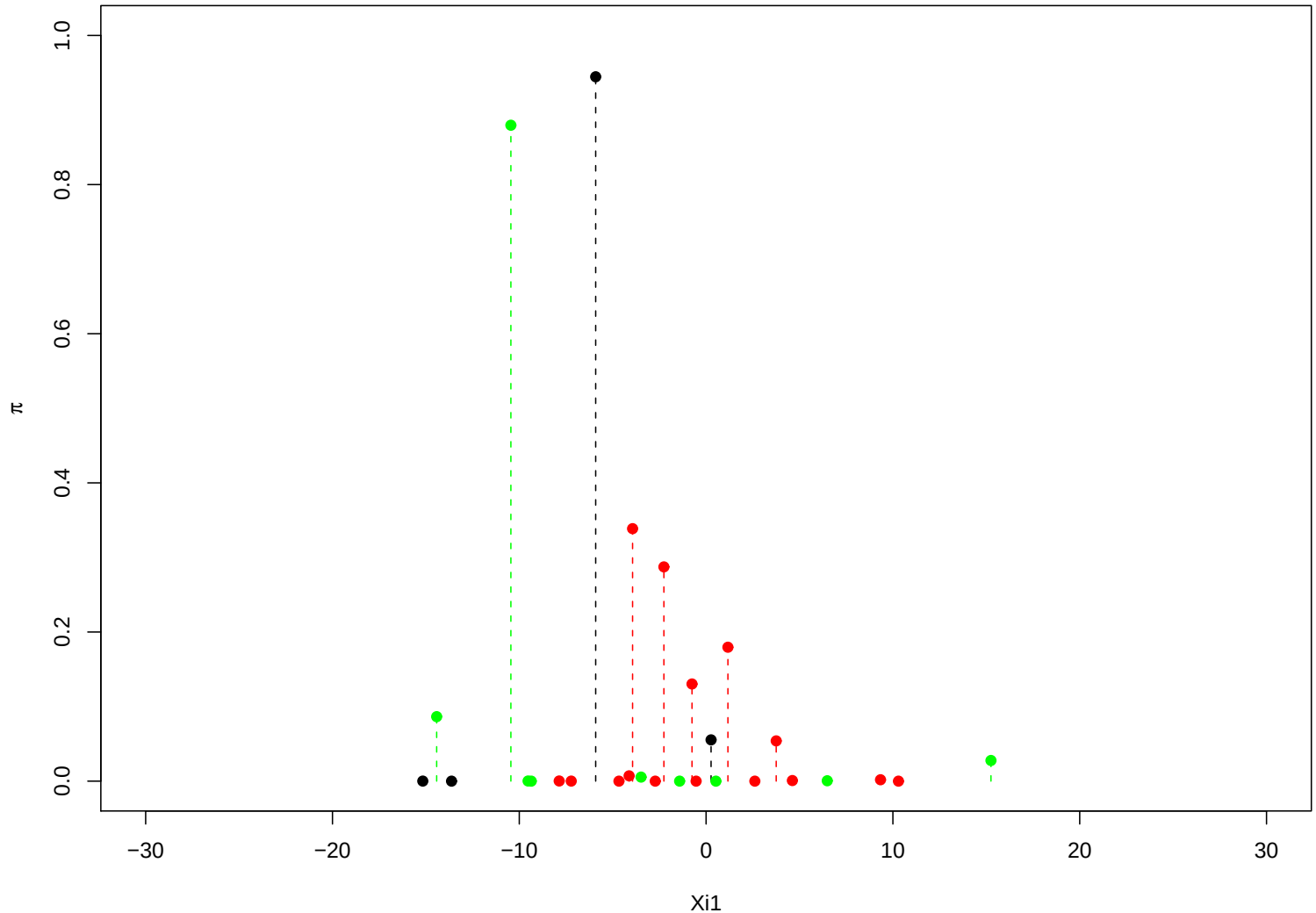
Simulation: $\alpha = 2$, thresholded



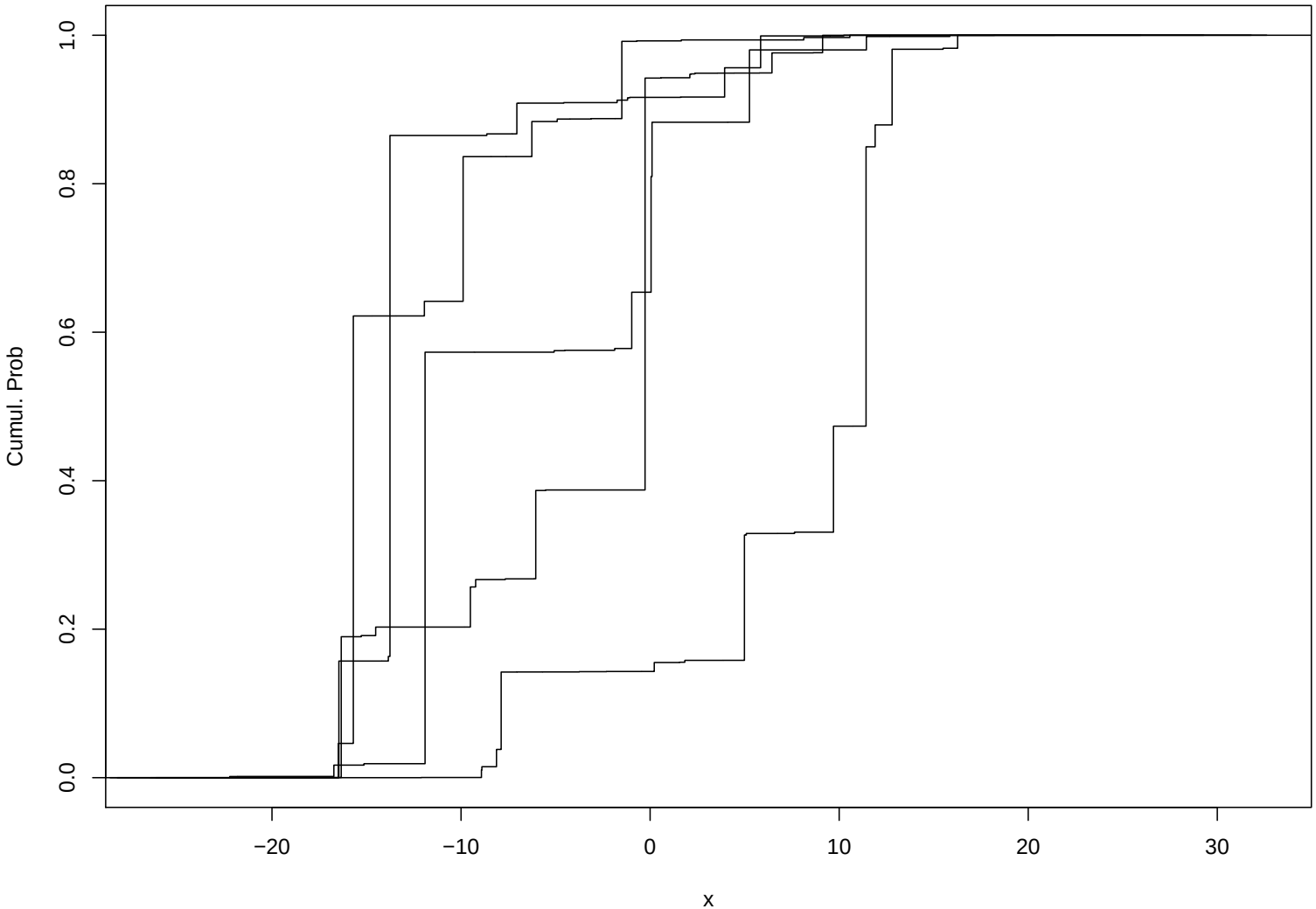
Simulation: $\alpha = 4$



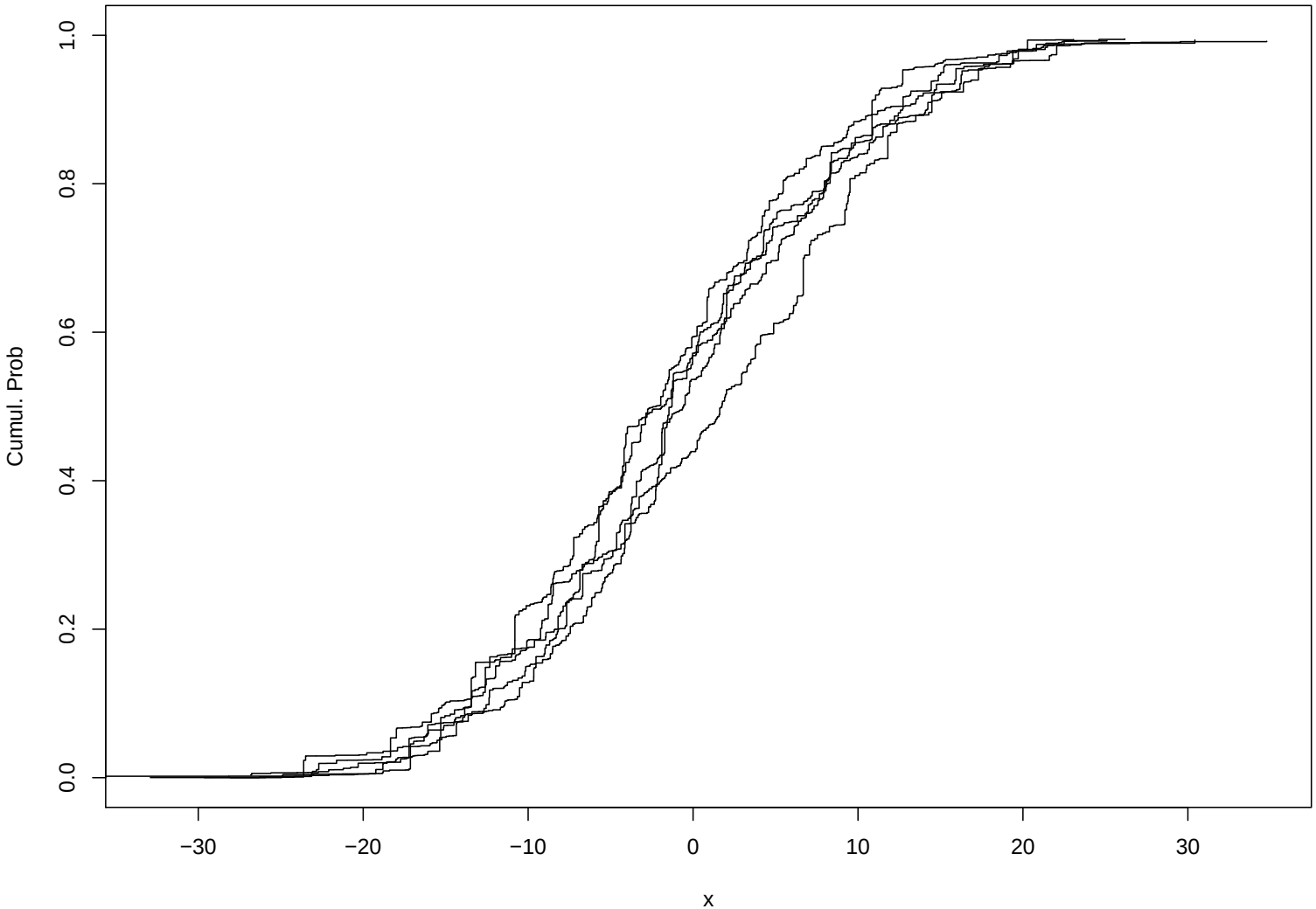
Simulation: $\alpha = 0.5$



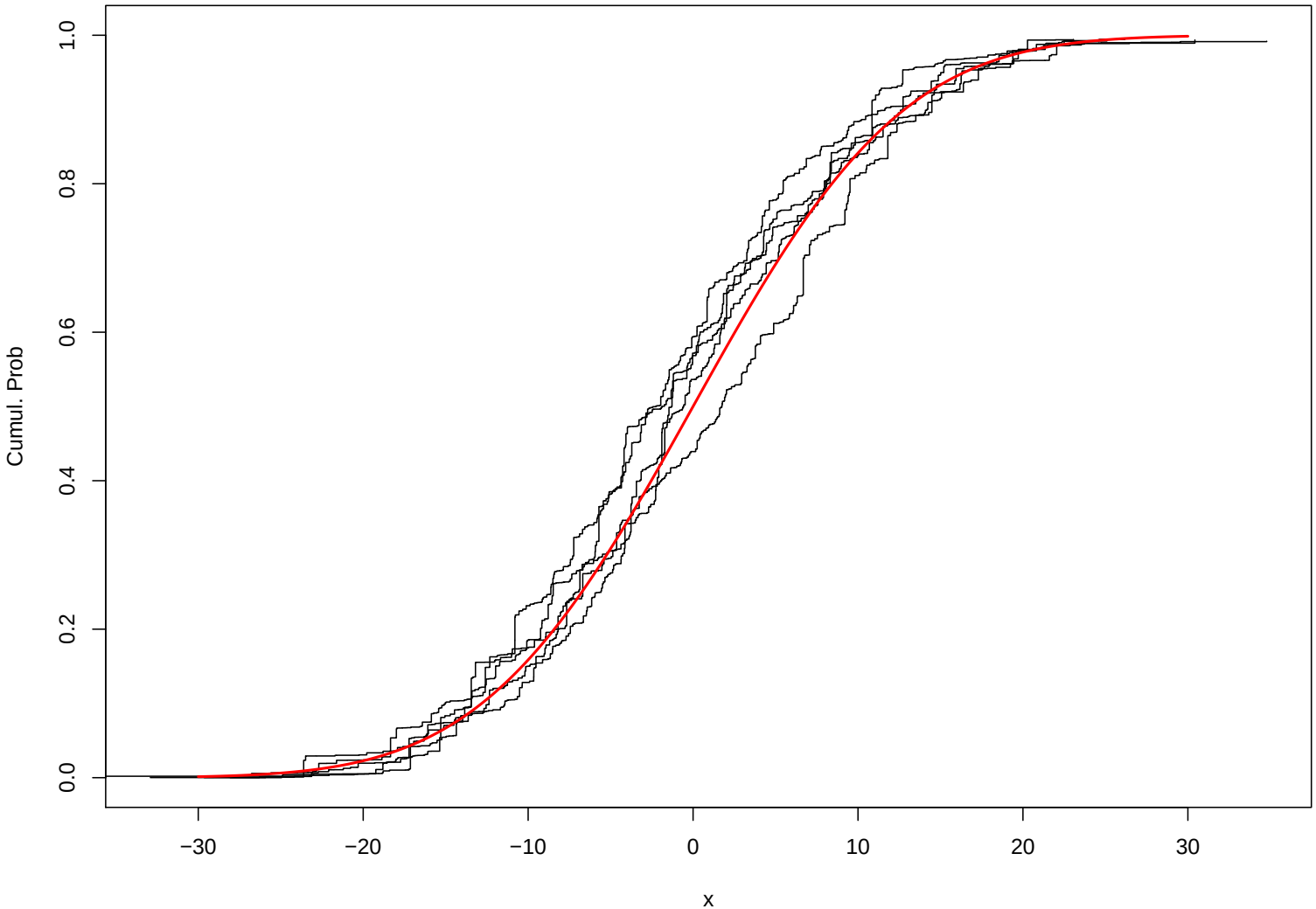
Simulation: $\alpha = 2$ (CDF)



Simulation: $\alpha = 100$ (CDF)



Simulation: $\alpha = 100$ (CDF)



Summary

- Two hyperparameters α, G_X.
- α small gives a few large masses
- α large gives many small masses
- α large reproduces a distribution much like G_X

Thus α is like a *precision* parameter, G_X is like a *location* parameter for the distribution.

Dirichlet Process

The method described above is the *stick-breaking* construction of the *Dirichlet Process* with parameters (α, G_X).

$$\widetilde{f} \sim DP(\alpha, G_X)$$

For any partition of $\mathbb{R}$ into disjoint subsets $B_1, B_2, \ldots, B_K, B_{K+1}$ the Dirichlet process assigns random probabilities

$$\boldsymbol{p} = (p_1, p_2, \ldots, p_K, p_{K+1})^\top$$

to the subsets, where

$$\boldsymbol{p} \sim Dirichlet(K; \alpha_1, \alpha_2, \ldots, \alpha_K, \alpha_{K+1})$$

and $\alpha_k = \alpha G_X(B_k)$ for each k

Posterior Calculation

Suppose *a priori* $\widetilde{f} \sim DP(\alpha, G_X)$. What is the posterior distribution if data $y_1, y_2, \ldots, y_n$ are observed and presumed independent draws from $\widetilde{f}$?

We have a conjugate model: *a posteriori*

$$\widetilde{f} \sim DP(\alpha^\star, G_X^\star)$$

where

$$\begin{aligned} \alpha^\star &= \alpha + n \\ G_X^\star &= \frac{\alpha G_X + \sum_{j=1}^{n} \delta_{y_j}}{\alpha + n} \end{aligned}$$

Dirichlet Process

For any partition of $\mathbb{R}$ into disjoint subsets $B_1, B_2, \ldots, B_K, B_{K+1}$ with associated probabilities

$$\boldsymbol{p} = (p_1, p_2, \ldots, p_K, p_{K+1})^\top$$

- in the *prior*

$$\boldsymbol{p} \sim \mathit{Dirichlet}(K; \alpha_1, \alpha_2, \ldots, \alpha_K, \alpha_{K+1})$$

and

$$\alpha_k = \alpha G_X(B_k)$$

- in the *posterior*

$$\boldsymbol{p} \sim \mathit{Dirichlet}(K; \alpha_1^\star, \alpha_2^\star, \ldots, \alpha_K^\star, \alpha_{K+1}^\star)$$

and

$$\alpha_k^\star = \alpha^\star G_X^\star(B_k)$$

Dirichlet Process

It follows that

$$\alpha_k^\star = \alpha^\star G_X^\star(B_k) = G_X(B_k) + n_k$$

where

$$n_k = \text{Number of } y_j \text{ in the set } B_k.$$

Therefore we have the usual kind of Bayesian updating rule.

Also

- α small: posterior looks like empirical cdf
- α large: posterior looks like G_X.

So, overall, things proceed much like parametric inference !

Example: Bayesian Survival Analysis

For possibly censored survival data, the usual approach is to use non-parametric maximum likelihood and the *Kaplan-Meier* curve.

Suppose we have

- event times

$$t_1 \leqslant \ldots \leqslant t_n$$

  with N_j events at time j.
- censoring indicators z_{jk} where

$$z_{jk} = \begin{cases} 1 & \text{Death} \\ 0 & \text{Censored} \end{cases}$$

Can form likelihood in terms of discrete hazards; leads to usual Kaplan-Meier formulation

Example: Bayesian Survival Analysis

In the Bayesian version, we specify a non-parametric model by considering the probability content for the disjoint intervals

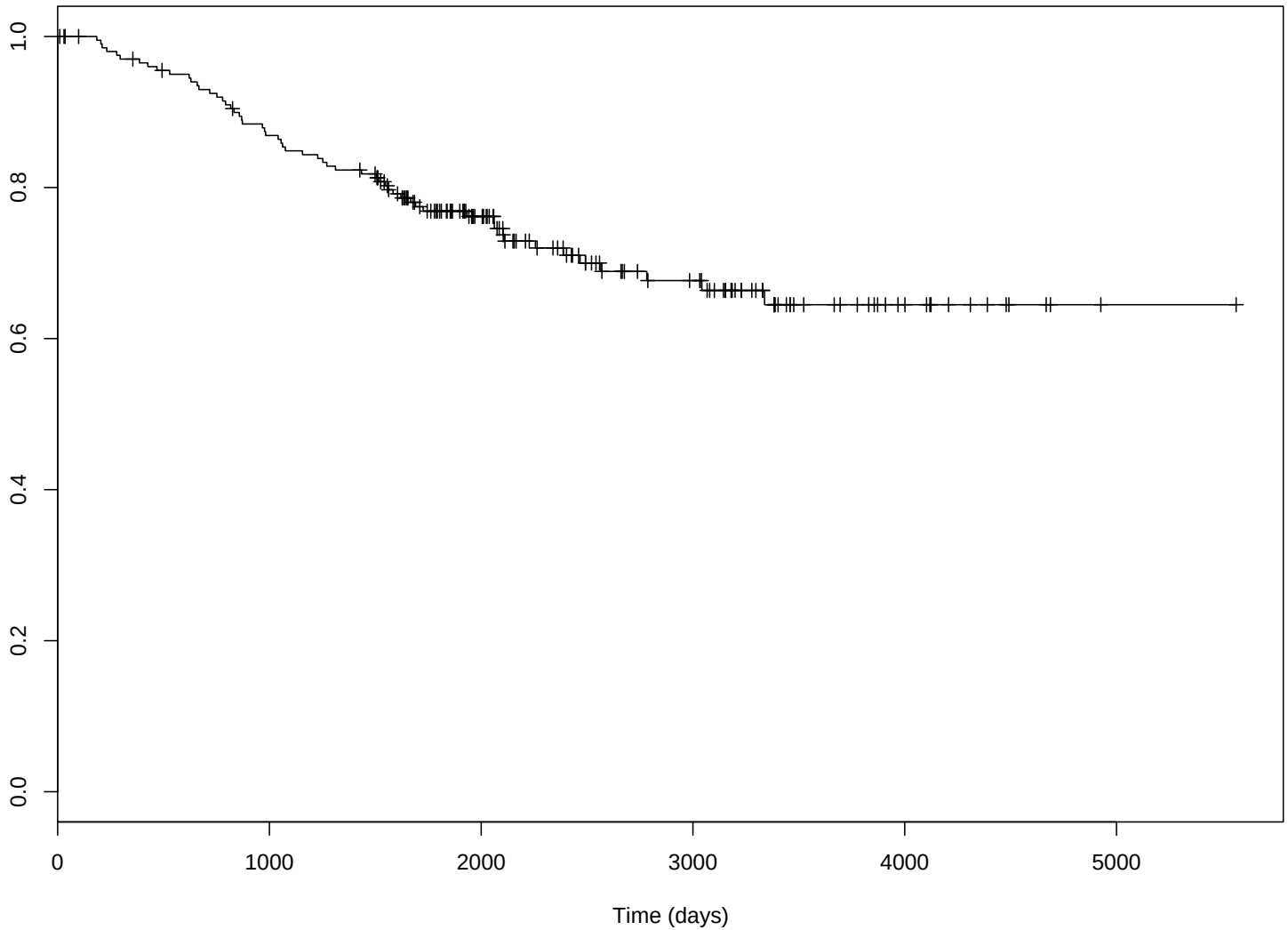
$$(0, t_1], (t_1, t_2], ..., (t_{n-1}, t_n], (t_n, \infty).$$

We may use a Dirichlet distribution to specify the prior; however, need to specify the α parameters.

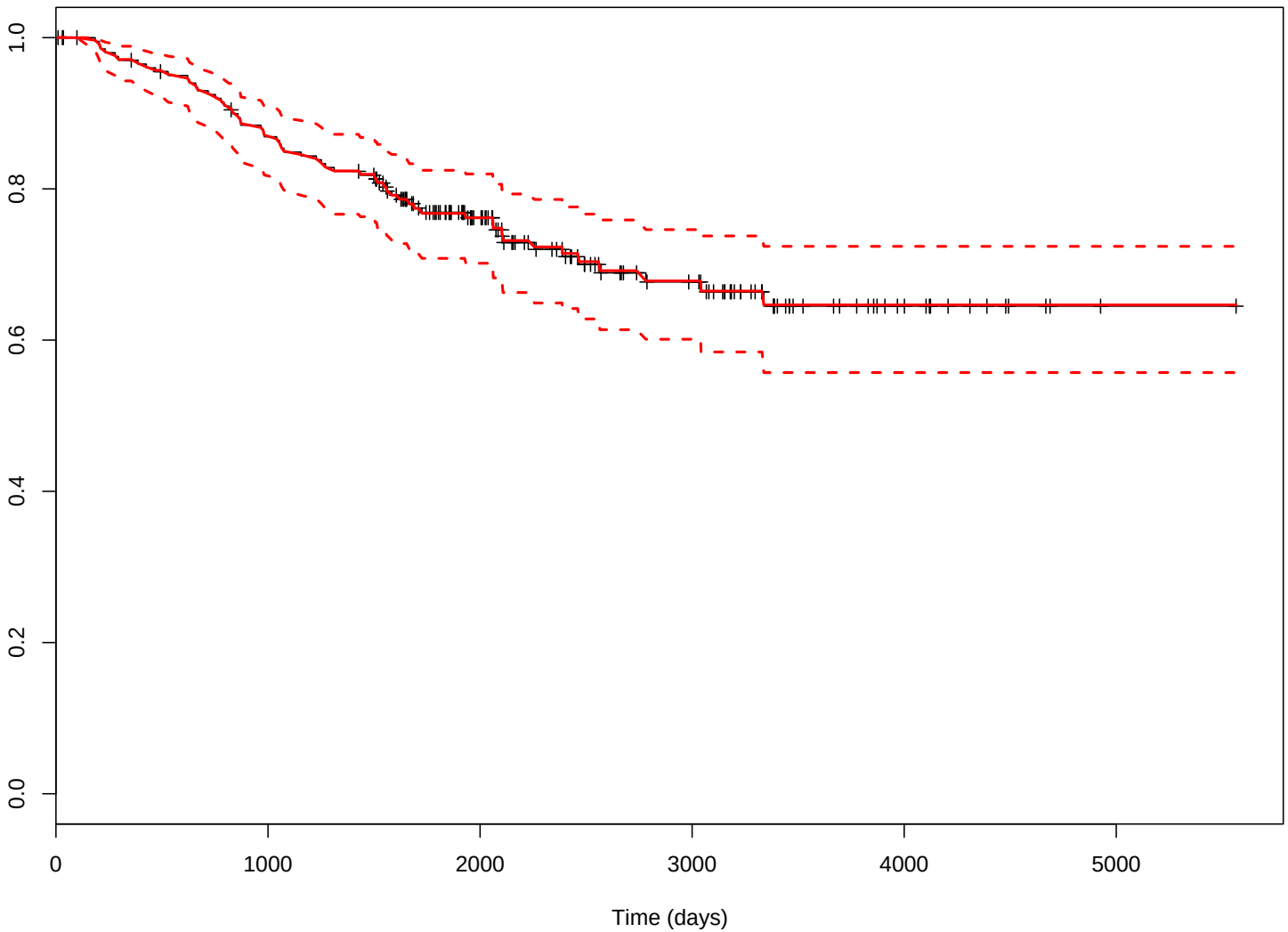
To do this in a *coherent* fashion, we use ideas from the Dirichlet process; we specify

$$\alpha_k = \alpha G_X((t_{k-1}, t_k]) \qquad k = 1, 2, \ldots$$

Example: Melanoma Data - Kaplan-Meier



Example: Melanoma Data - Bayesian interval



Dirichlet Process: Some issues

1. How do we report the inference ?
   - Which partition, that is, which sets $B_1, B_2, \ldots, B_K, B_{K+1}$ should we choose to report the posterior ?
2. In this formulation, the posterior can only support *discrete* distributions
   - that is, any estimate of the true f obtained from the model is almost surely a discrete distribution.

We solve 1. by using *simulation* methods

We solve 2. by *extending the model*.

Simulating from a Dirichlet Process Model

To obtain a sample (of data) of size M from a Dirichlet Process $DP(\alpha, G_X)$:

STEP 1: Sample $Z_1 \sim G_X$.

STEP 2: For $j = 2, \ldots, M$, sample

$$Z_j | Z_1, Z_2, \ldots, Z_{j-1} \sim \frac{\alpha}{\alpha + j - 1} G_X + \frac{1}{\alpha + j - 1} \sum_{l=1}^{j-1} \delta_{Z_l}(Z_j)$$

that is, given $Z_1, Z_2, \ldots, Z_{j-1}$, either sample

Z_j from G_X with probability $\alpha/(\alpha + j - 1)$.

or sample

Z_j from $\{Z_1, \ldots, Z_{j-1}\}$ with probability $1/(\alpha + j - 1)$ each.

The Dirichlet Process and Clustering

This algorithm is a *Polya Urn* scheme.

It demonstrates that the Dirichlet process model induces a *clustering* mechanism: in the simulated Z sample, we have many identical values due to the sampling of Z_j uniformly on

$$\{Z_1, \ldots, Z_{j-1}\}$$

at each j.

In a sample of n items from a $DP(\alpha, G_X)$ model, the probability of having k clusters is

$$\Pr[K = k] = \frac{\alpha^k B(n,k)}{A_n(\alpha)}$$

where $B(n,k)$ is the *Stirling number of the first kind*.

$$A_n(x) = \sum_{j=1}^{n} B(n,j) x^j$$

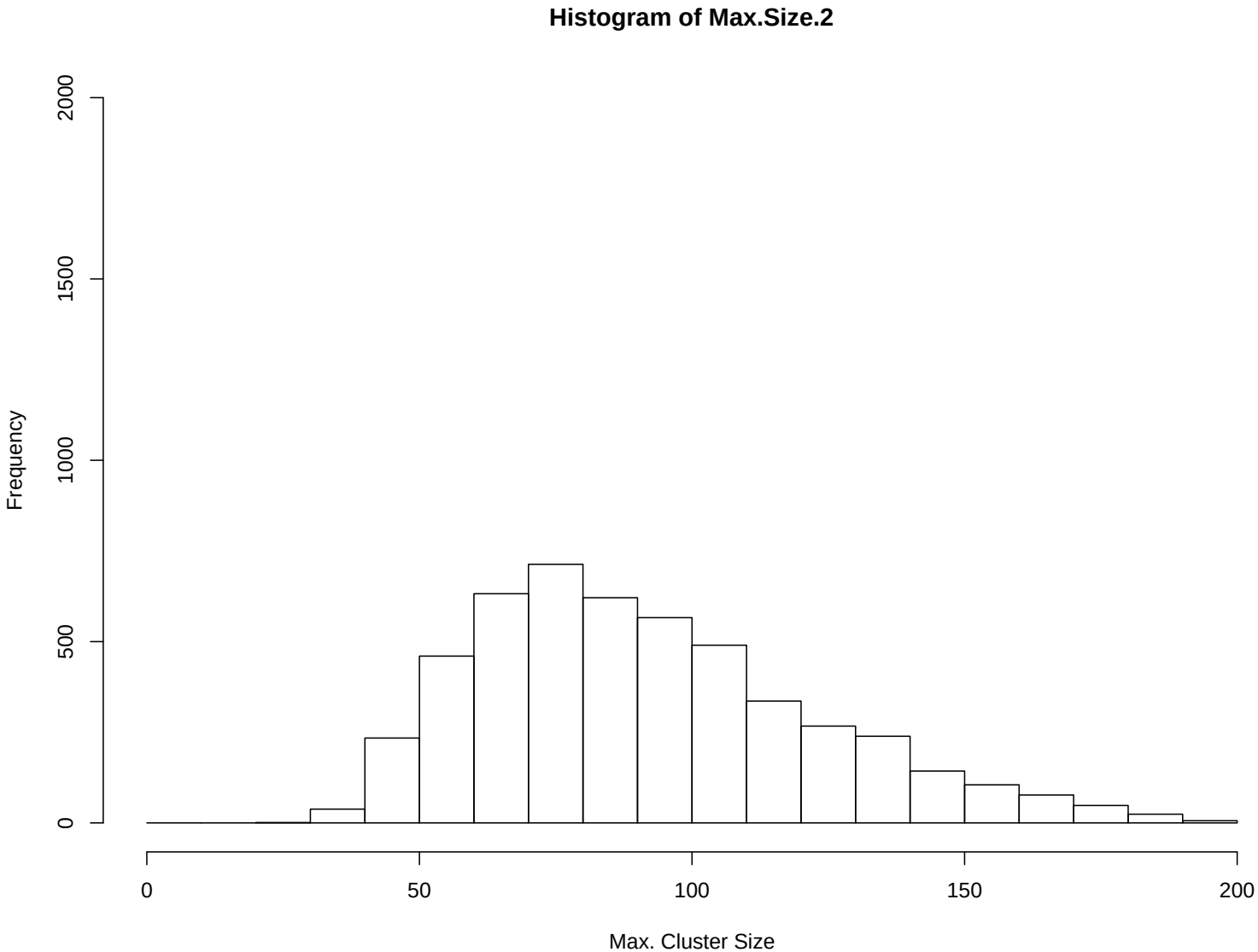
The expected number of clusters *a priori* is

$$\sum_{j=1}^{n} \frac{\alpha}{\alpha + j - 1} = \mathrm{O}(\alpha \log n)$$

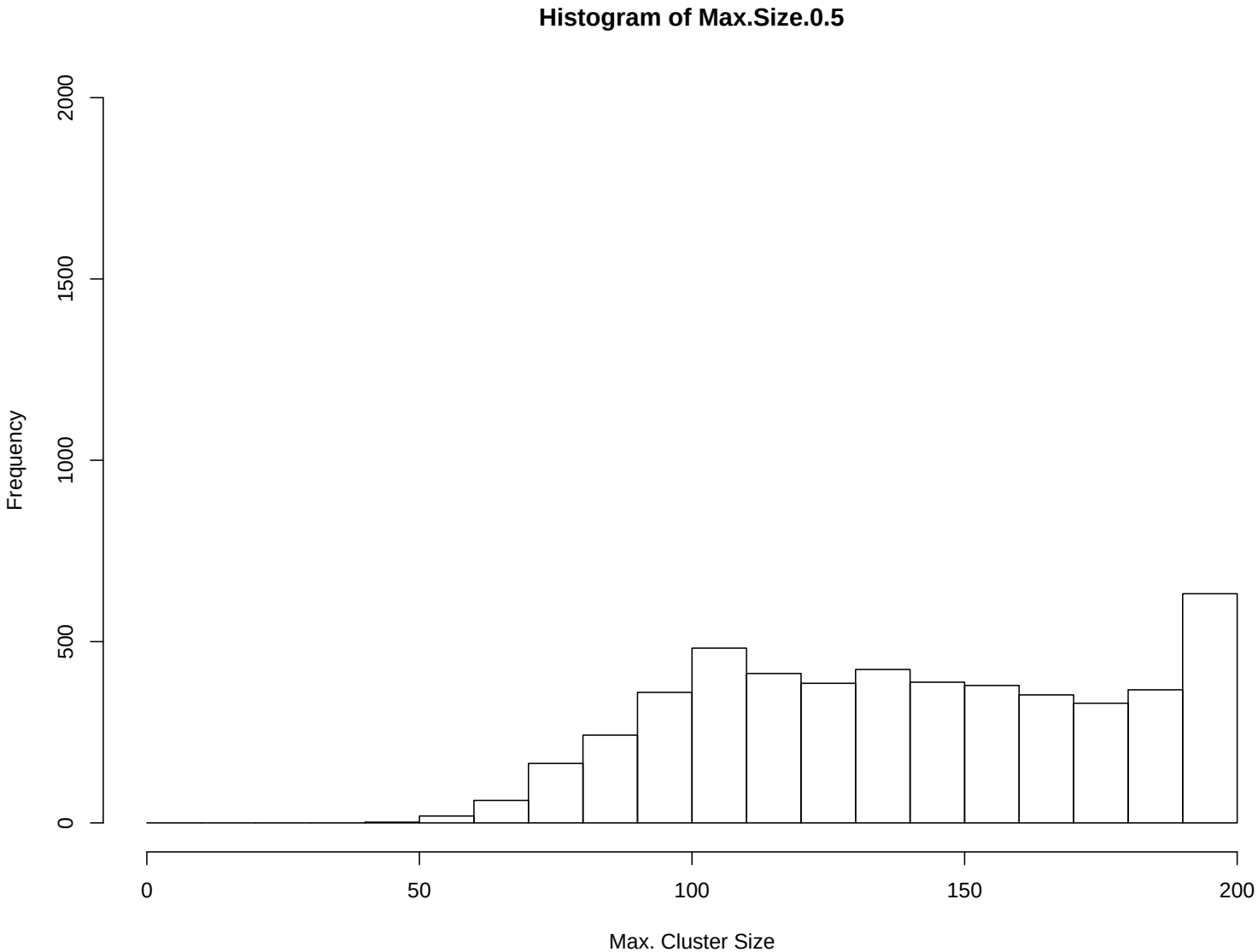
For $n = 200$:

α	$\mathbb{E}[K]$
0.5	3.631
2.0	9.766
4.0	16.238
10.0	30.930

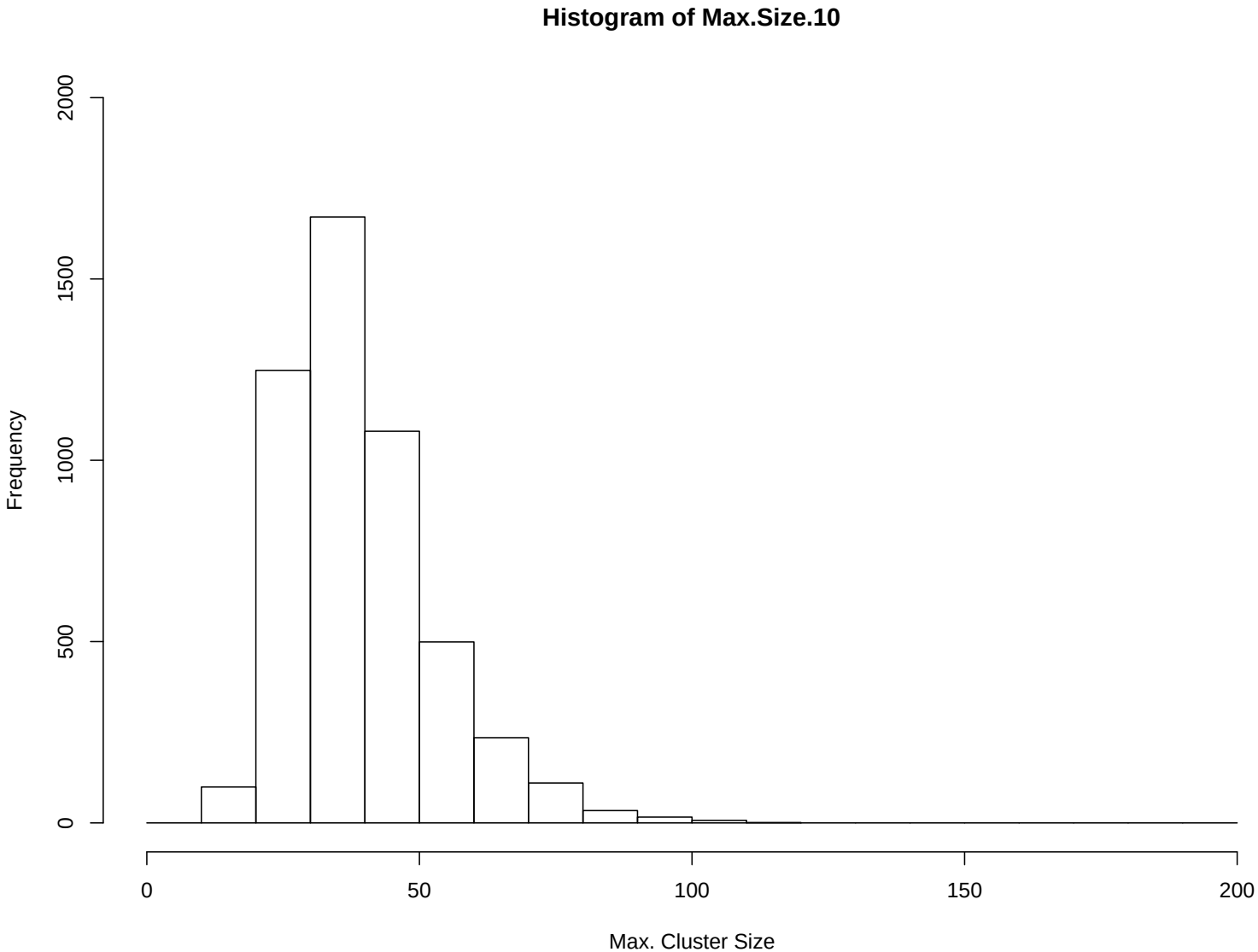
Maximum Cluster size: $\alpha = 2$



Maximum Cluster size: $\alpha = 0.5$

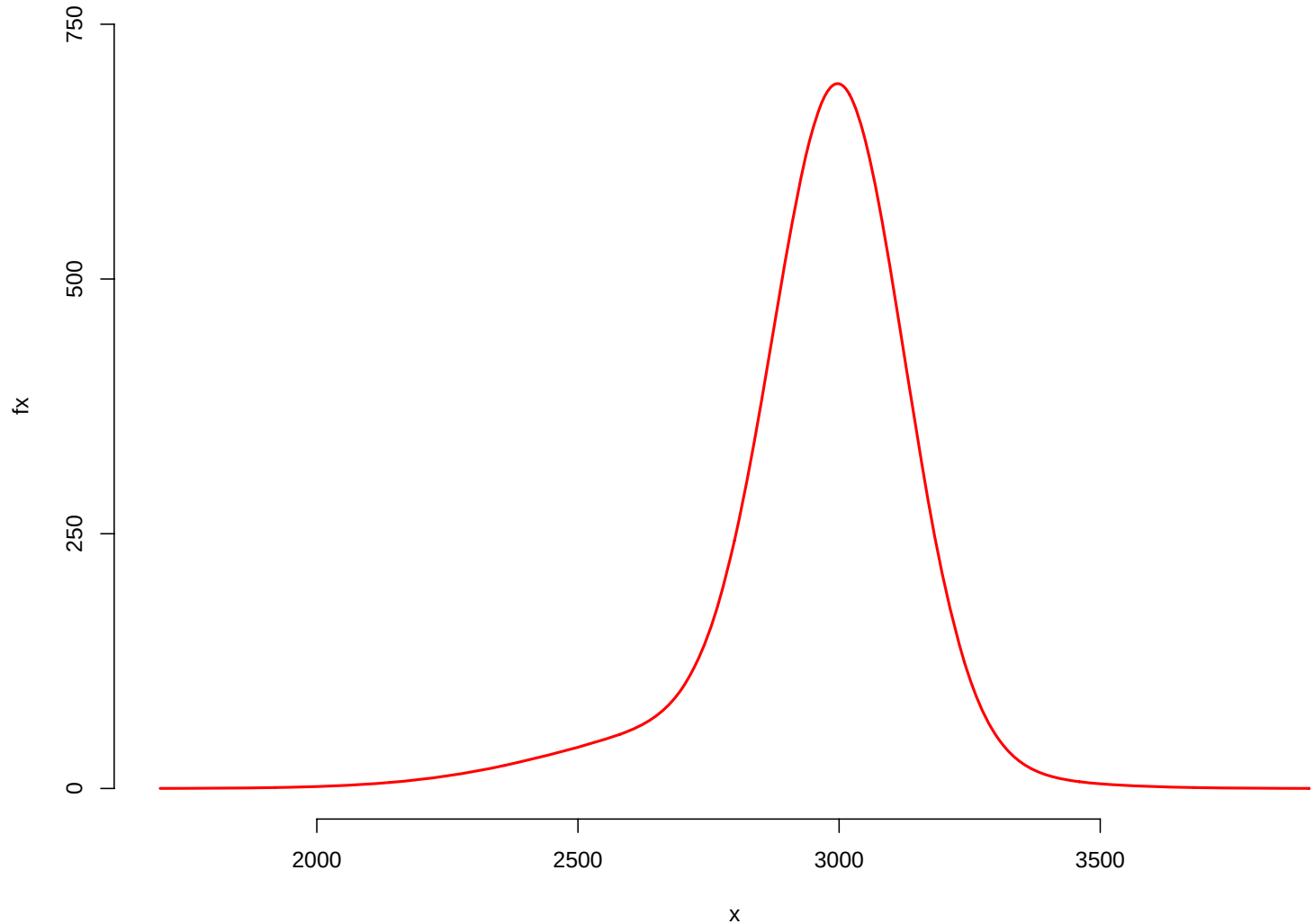


Maximum Cluster size: $\alpha = 10$



Extension to The Continuous Case

We need to model continuous distributions.



Extension to The Continuous Case

We add another stage that brings in a continuous distribution.

For example, could treat each x_i as the location of a normal density, and consider generating a y for each

$$\begin{aligned} x_1, x_2, \ldots &\sim G_X \\ \pi_1, \pi_2, \ldots &\quad \text{generated by stick-breaking.} \\ y &\sim \phi((y - x_i)/\sigma) \qquad i = 1, 2, \ldots \end{aligned}$$

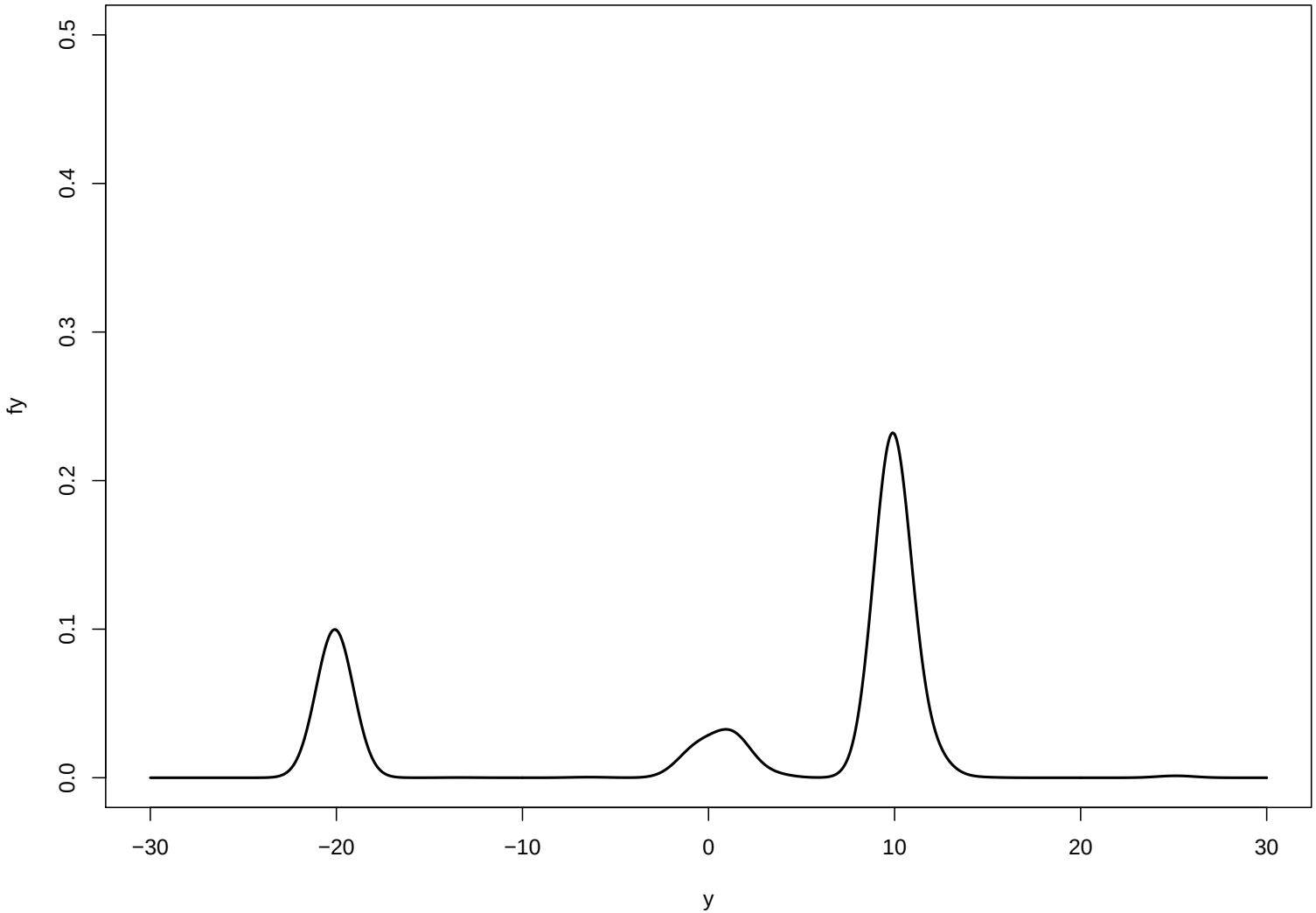
Then,

$$\widetilde{f}(y) = \sum_{i=1}^{n} \pi_i \phi((y - x_i)/\sigma)$$

that is, an *infinite mixture model*.

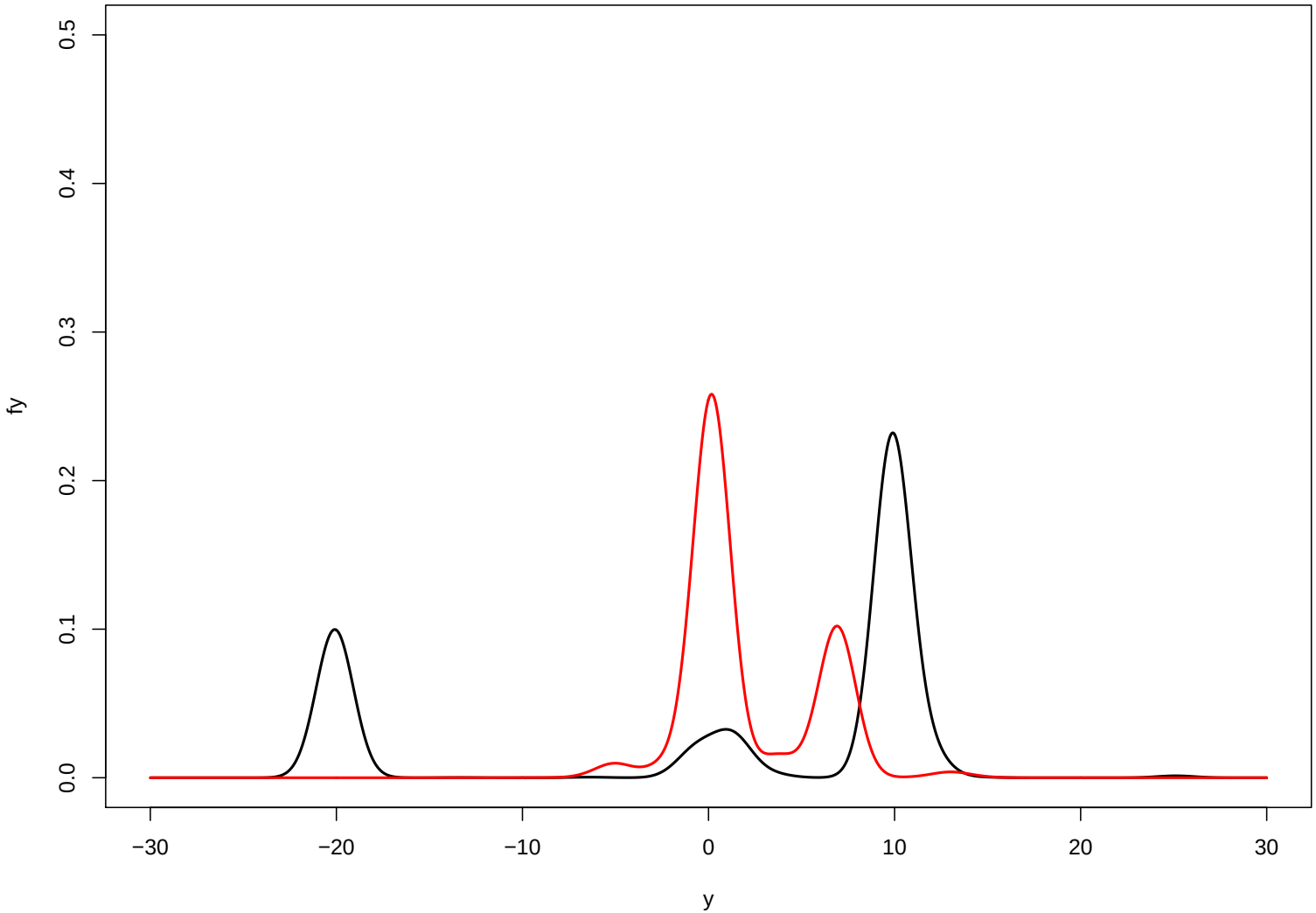
Example: $\alpha = 2, \sigma = 1$

Simulation 1



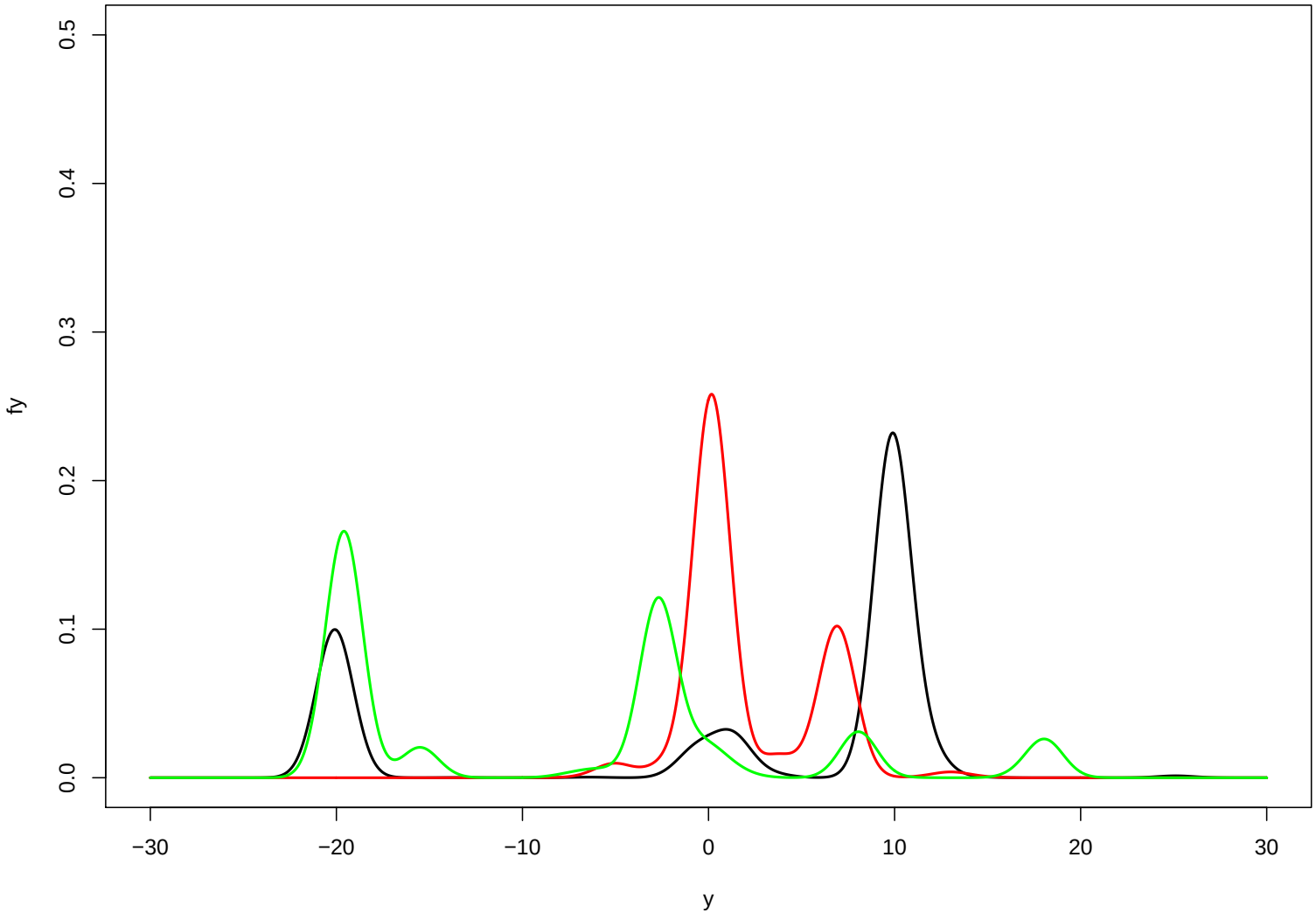
Example: $\alpha = 2, \sigma = 1$

Simulation 2



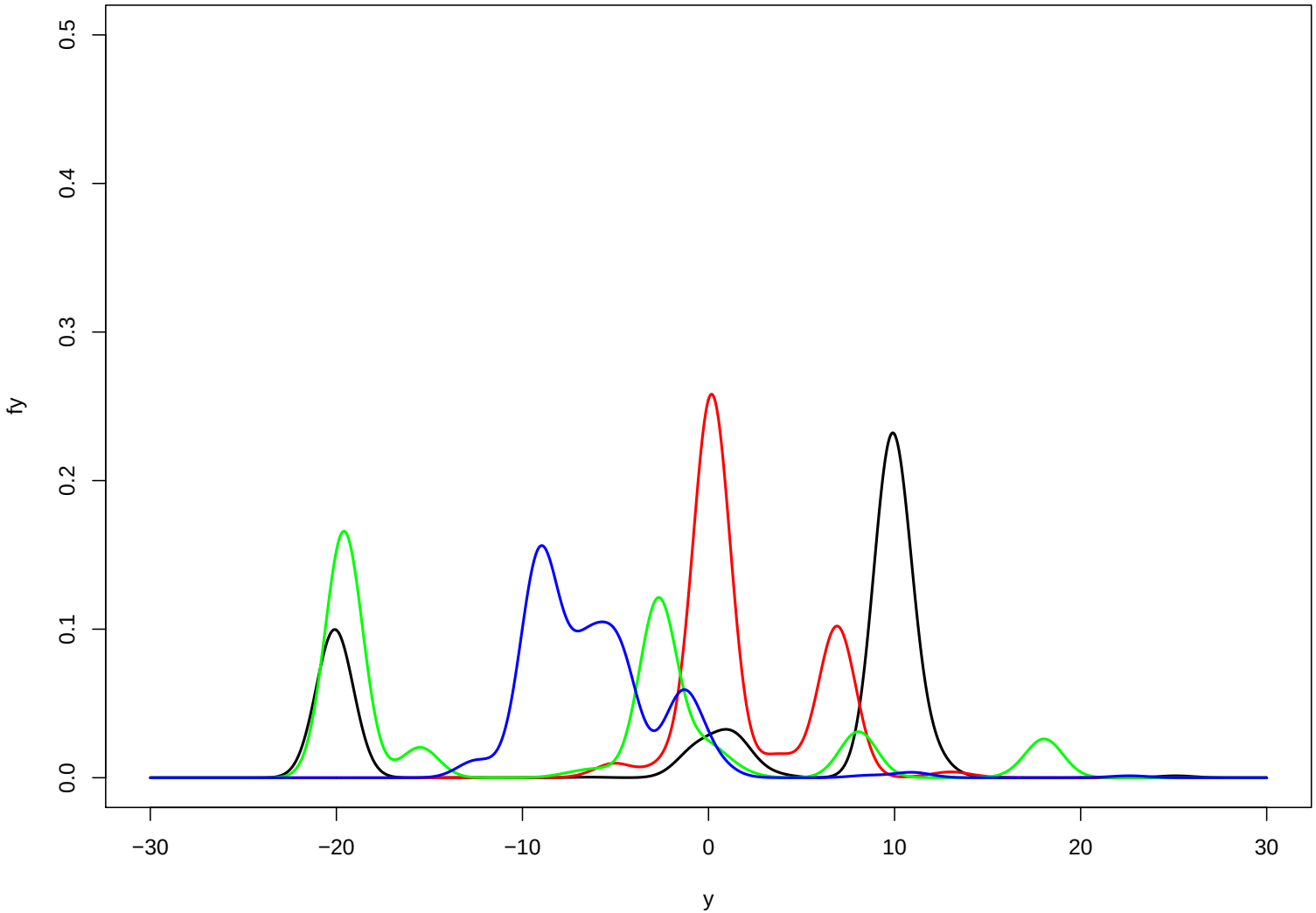
Example: $\alpha = 2, \sigma = 1$

Simulation 3



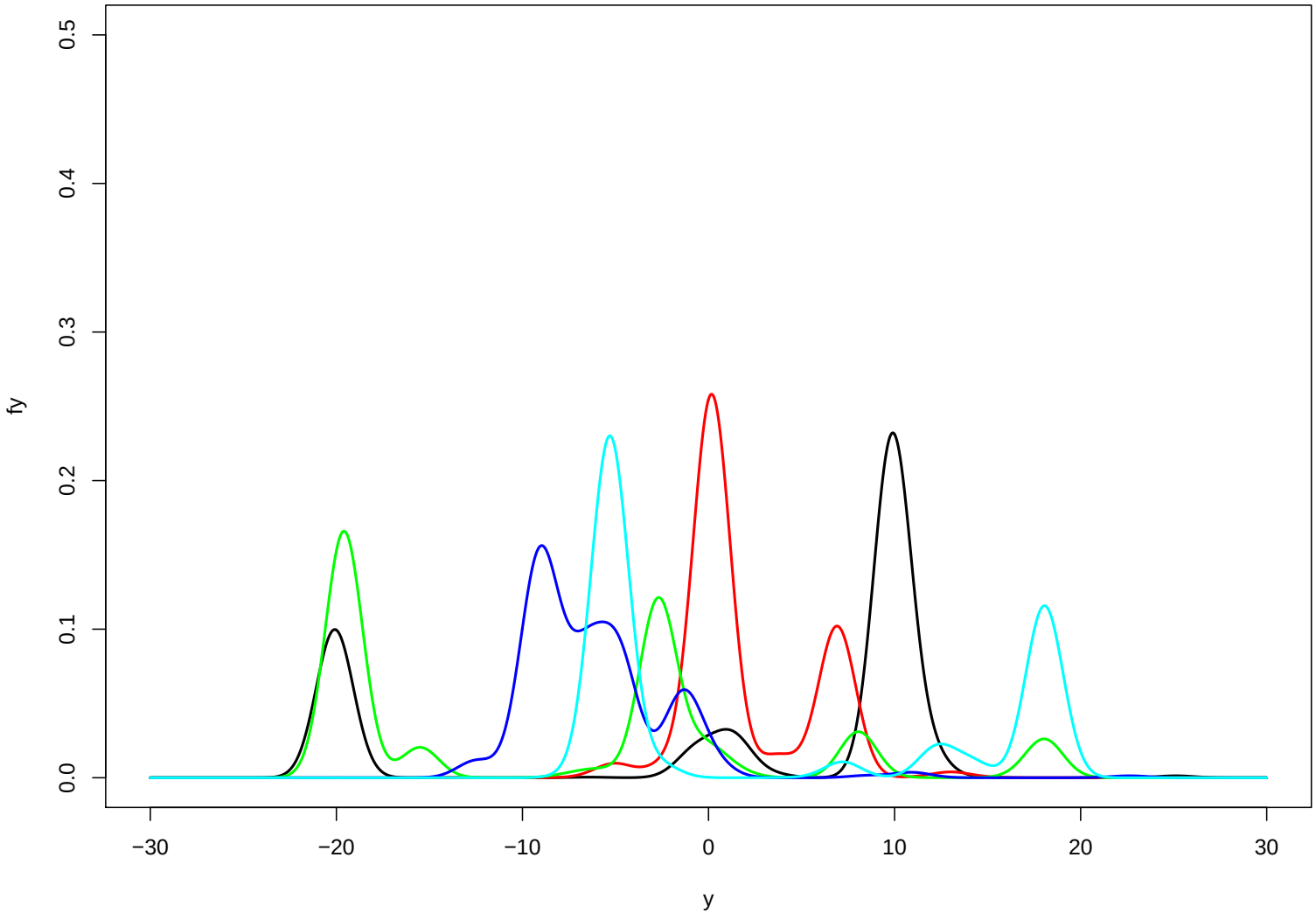
Example: $\alpha = 2, \sigma = 1$

Simulation 4



Example: $\alpha = 2, \sigma = 1$

Simulation 5



Dirichlet Process Mixture

This construction is called the

Dirichlet Process Mixture (DPM)

with a Normal kernel. Any continuous kernel g_Y can be used in place of ϕ.

Under this model, $\widetilde{f}$ is almost surely *continuous*.

- α small implies less bumpy
- α large implies more bumpy.

Bayesian Inference

Suppose we have the usual de Finetti construction

- a prior model for f that is $DPM(\alpha, G_X, g_Y; \theta)$ where θ represents the parameters that appear in G_X and g_y.
- conditional on f, data $y_1, y_2, \ldots, y_n \sim f$

We wish to compute the posterior for f. We use the hierarchical formulation

$$\begin{aligned} y_j|x_j &\overset{\text{ind.}}{\sim} g_Y(y_j|x_j, \theta) \qquad j = 1, \ldots, n \\ x_1, \ldots, x_n &\sim DP(\alpha, G_X; \theta) \\ \theta &\sim p(\theta). \end{aligned}$$

Bayesian Inference: Algorithm 1

The latent variables $x_1, \ldots, x_n$ are also treated as parameters. They can be sampled using an MCMC *Gibbs sampler* scheme.

For $j = 1, \ldots, n$, we sample

$$x_j \mid \boldsymbol{x}_{(j)}, \boldsymbol{y} \sim w_0 p(x_j|y_j) + \sum_{l \neq j} w_l \delta_{x_l}$$

where

- $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$
- $\boldsymbol{x}_{(j)} = (x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)^\top$.
- w_0 is proportional to α times the *prior predictive* of y_j
- w_l is proportional to the *likelihood* of $y_j|x_l$
- $p(x_j|y_j)$ is the *posterior* for x_j given y_j.

Bayesian Inference: Algorithm 2

To be more efficient, we can use the *clustering* property: suppose that at a given iteration of the MCMC, there are K clusters labelled 1 to K, where $K \leqslant n$.

Label the K distinct x values

$$z_1, \ldots, z_K$$

and for each j, define the corresponding cluster label c_j where

$$c_j = k \quad \Longleftrightarrow \quad x_j = z_k$$

We can update the c_js instead of the x_js which will be more computationally efficient; we are clustering xs to the *cluster centres* at the z values.

Bayesian Inference: Algorithm 2

For $i = 1, \ldots, n$, let

- $n_1(i), \ldots, n_K(i)$ denote the number of items in clusters $1, \ldots, K$
- $\boldsymbol{y}_1(i), \ldots, \boldsymbol{y}_K(i)$ denote the vectors of y values currently allocated to the K clusters

if the ith data point is removed.

For $i = 1, \ldots, n$, we sample the cluster labels in a Gibbs sampler with conditional probabilities

$$\Pr[c_i = k \mid c_{(i)}] \propto \frac{n_k(i)}{n - 1 + \alpha} p(y_i | \boldsymbol{y}_k(i)) \qquad k = 1, \ldots, K$$

and

$$\Pr[c_i = K + 1 \mid c_{(i)}] \propto \frac{\alpha}{n - 1 + \alpha} p(y_i)$$

Bayesian Inference: Algorithm 2

In this expression

- $p(y_i|\boldsymbol{y}_k(i))$ is the *posterior predictive* density for y_i in the DPM model, assuming that y_i comes from cluster k.
- $p(y_i)$ is the *prior predictive* density for y_i in the DPM model, assuming that y_i comes from a new cluster not currently represented in the data.

In this formulation, we have *integrated out* the Dirichlet process.

Thus we can simply sample the cluster labels in turn, and then sample the $z_1, \ldots, z_k$ values; this will allow us to do density estimation.

Bayesian Inference: Algorithm 2

By the usual calculation

$$p(y_i|\boldsymbol{y}_k(i)) = \int g_Y(y_i \mid \mathrm{x}) p(\mathrm{x} \mid \boldsymbol{y}_k(i))\, d\mathrm{x}$$

where

$$\begin{aligned} p(\mathrm{x} \mid \boldsymbol{y}_k(i)) &\propto p(\boldsymbol{y}_k(i) \mid \mathrm{x}) p(\mathrm{x}) \\ &= \left\{ \prod_{l \neq i} g_Y(y_l \mid \mathrm{x}) \right\} p(\mathrm{x}) \end{aligned}$$

gives the posterior distribution for the kth cluster centre.

Similarly

$$p(y_i) = \int g_Y(y_i \mid \mathrm{x}) p(\mathrm{x})\, d\mathrm{x}$$

Bayesian Inference: Algorithm 2

In the earlier Gaussian model, suppose for simplicity that G_X is the $N(0, \lambda^2)$ density:

$$\begin{aligned} x_i &\sim N(0, \lambda^2) \\ y_i \mid x_i &\sim N(x_i, \sigma^2) \end{aligned}$$

Then

$$p(y_i | \boldsymbol{y}_k(i)) \equiv N\left(\frac{n_k(i)\overline{y}_k(i)}{n_k(i)/\sigma^2 + 1/\lambda^2}, \frac{(n_k(i)+1)/\sigma^2 + 1/\lambda^2}{n_k(i)/\sigma^2 + 1/\lambda^2}\sigma^2 \right)$$

and

$$p(y_i) \equiv N\left(0, \sigma^2 + \lambda^2\right)$$

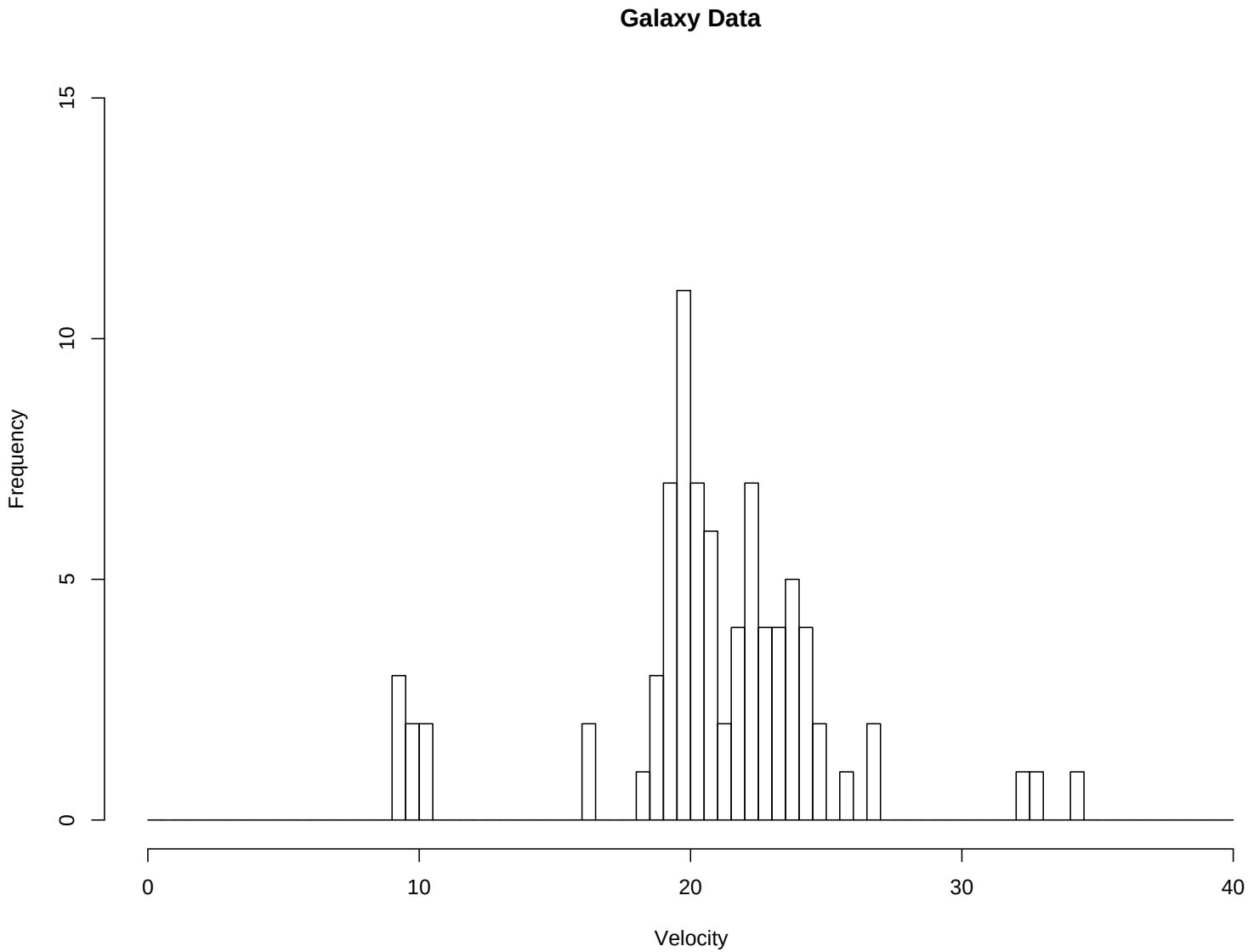
Extensions

Easy to extend to

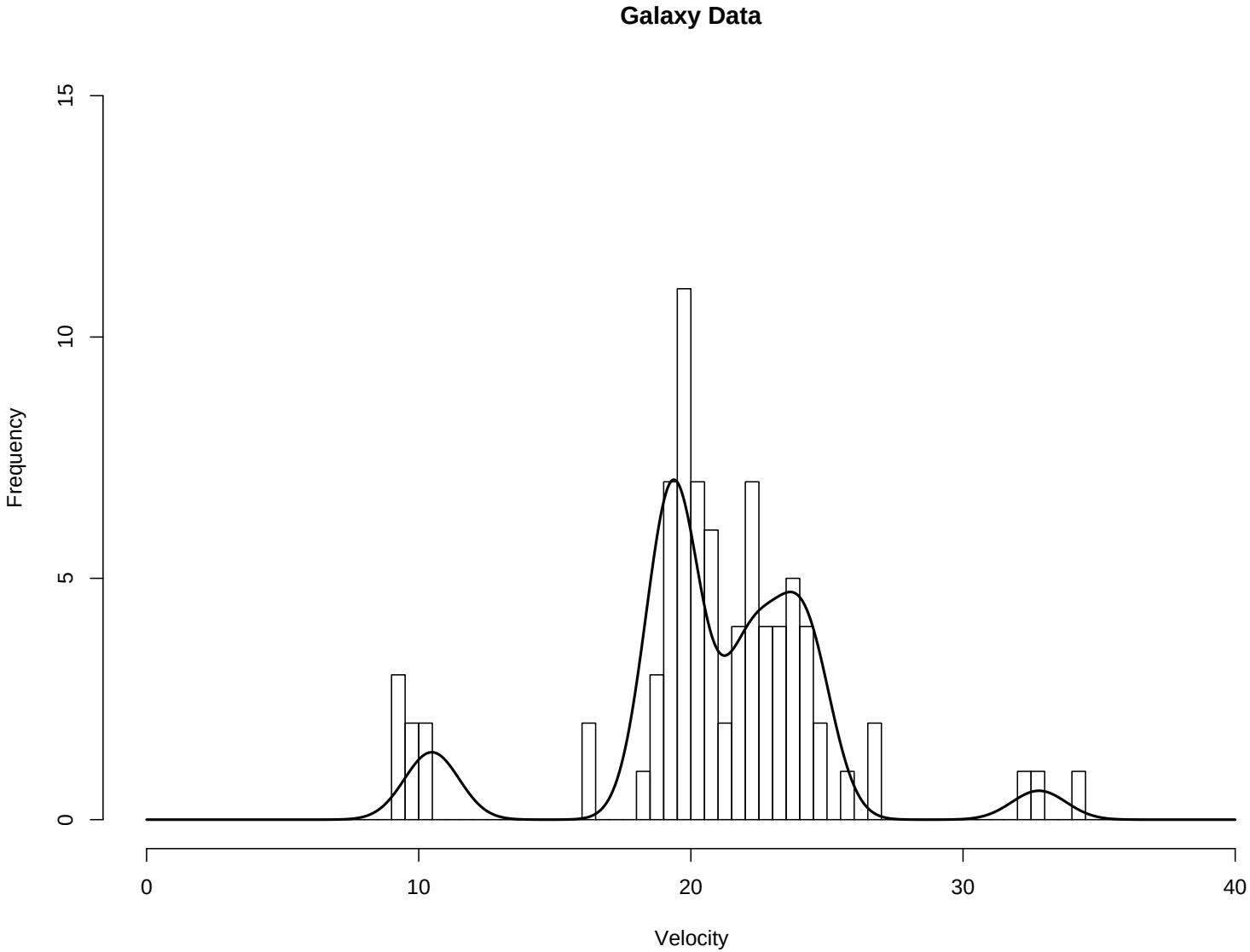
- unknown σ^2
- non-Gaussian conjugate models
- blocked Gibbs sampler
- Metropolis-Hastings MCMC for cluster labels
- multivariate conjugate models

Not so easy to do non-conjugate models.

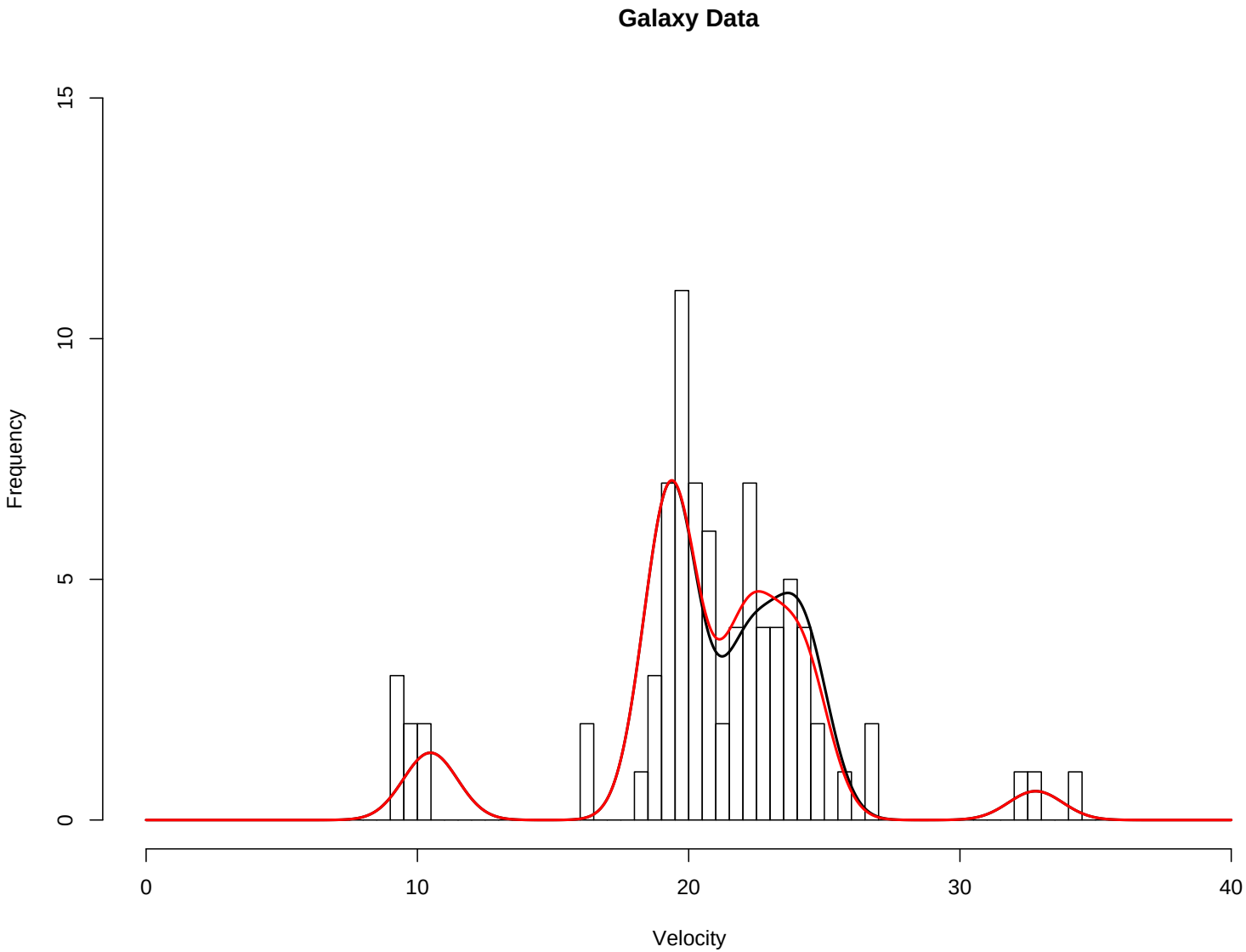
Example: Galaxy Data



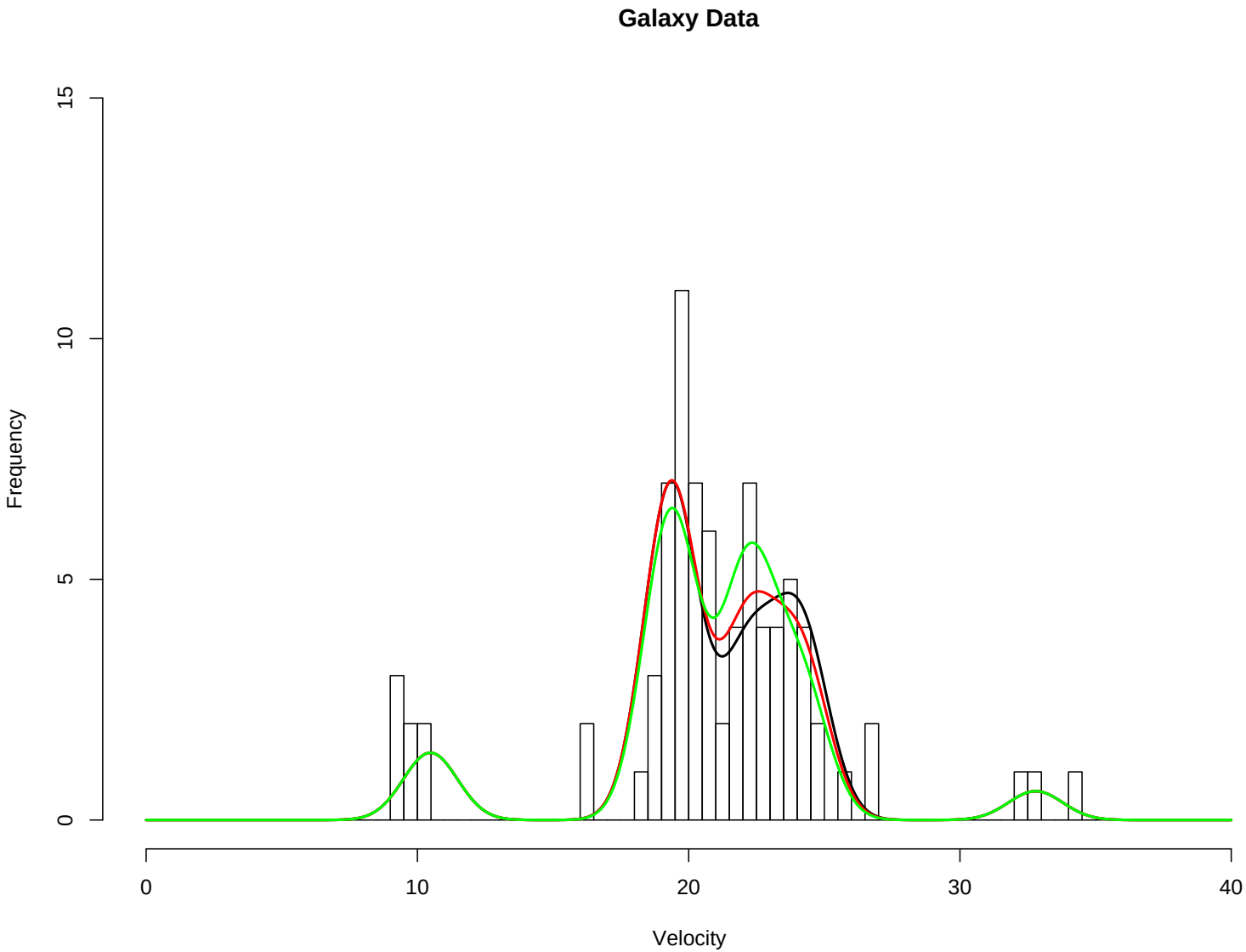
Example: Galaxy Data



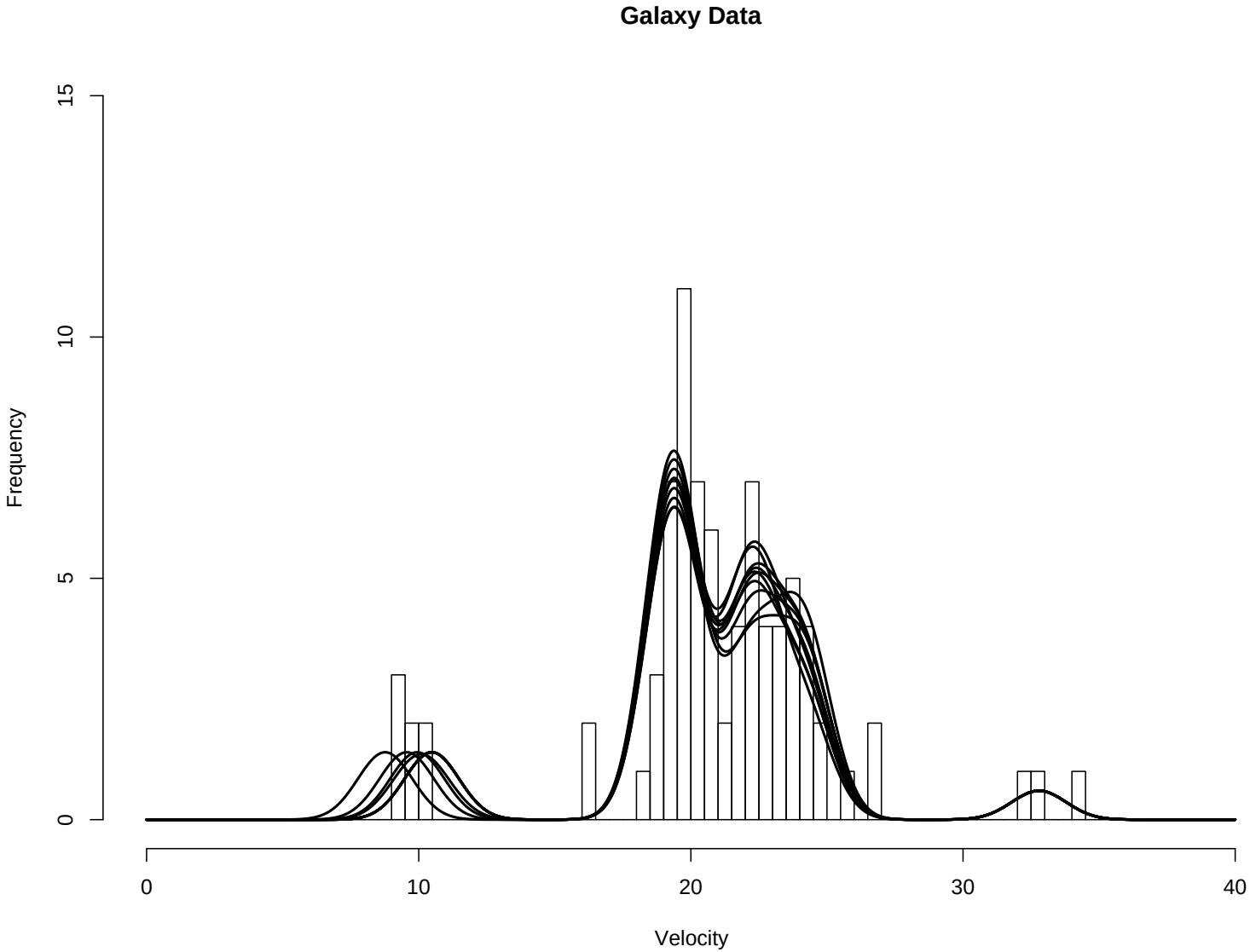
Example: Galaxy Data



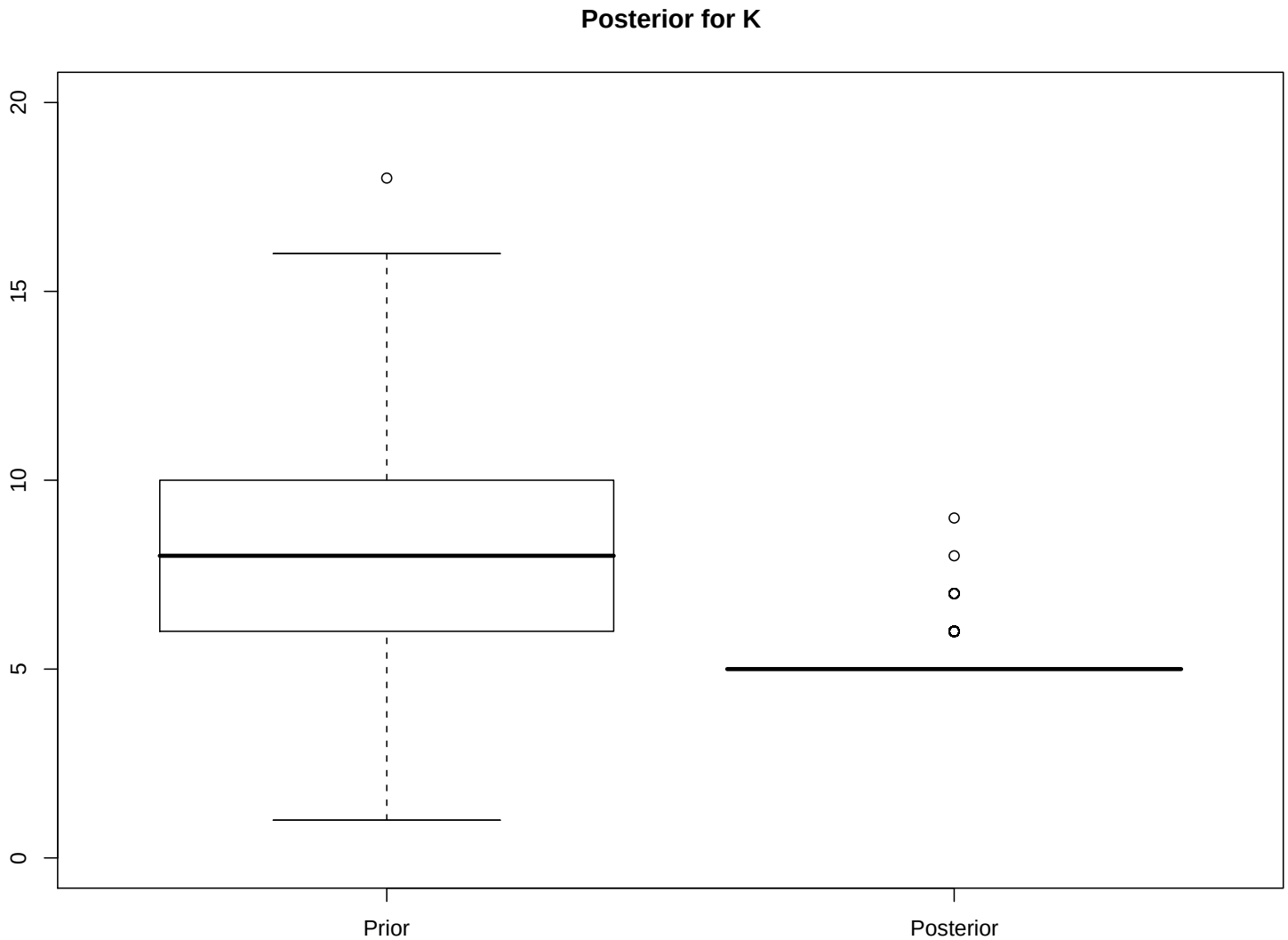
Example: Galaxy Data



Example: Galaxy Data



Example: Galaxy Data



Examples

Example: Galaxy data

See `knitr 20`

Extensions & Open Problems

- Extensions :
  - Polya Tree Models
  - Hypothesis Testing
  - Spatial Problems
  - Normalized Random Measures
  - Connections with Lévy Processes
- Open Problems:
  - Properties of Estimators (consistency etc.)
  - Model Selection
  - Comparison with *Bayesian Semi-Parametrics*

Resampling Approaches to Inference

Frequentist properties of statistical estimators are often difficult to study in a finite sample setting: for example, the finite sample variance of the sample median can only be computed via numerical integration.

Resampling methods allow the study of frequentist properties of statistical quantities by producing pseudo-replicate data sets of the same size as the observed data, and examining the statistical variation across these replicate data sets.

Notation: independence case

Suppose $Y_1, \ldots, Y_n \sim F$ are a random sample, and let $\theta = \theta(F)$ be the focus of inferential interest. For example

$$\theta(F) = \int y \, dF(y) \qquad \text{or} \qquad \theta(F) = \inf_y \{F(y) \geqslant p\}$$

etc. Let $y_1, \ldots, y_n$ be the observed data.

If $\widehat{F}_n$ is the empirical cdf,

$$\widehat{F}_n(y) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{[y_i, \infty)}(y) \qquad\qquad d\widehat{F}_n(y) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{y_i\}}(y)$$

then a natural 'plug-in' estimator of $\theta(F)$ is $T_n = \theta(\widehat{F}_n)$,

Notation: independence case

Recall that, for each y, under mild regularity conditions

$$\widehat{F}_n(y) \xrightarrow{p} F(y)$$

but also

$$\sup_y |\widehat{F}_n(y) - F(y)| \xrightarrow{p} 0$$

as $n \longrightarrow \infty$. Therefore

$$\theta(\widehat{F}_n) \xrightarrow{p} \theta(F)$$

which justifies (asymptotically) the use of the plug-in estimator.

Notation: independence case

The bias and variance of the estimator are

$$\begin{aligned} b_{T_n}(F) &= \mathbb{E}_F[T_n] - \theta(F) \\ v_{T_n}(F) &= \mathrm{Var}_F[T_n] \end{aligned}$$

both of which depend on the true F.

Notation: independence case

We wish to study these properties of the estimator. In some cases, it is possible to study these quantities analytically. Suppose, however, $\widehat{\theta}$ is the solution of

$$\sum_{i=1}^{n} m(y_i; \theta) = 0$$

for some *m-estimating function*.

The corresponding estimator is not analytically available, so its finite sample properties are hard to study.

Suppose we wish to summarize an aspect of the sampling distribution of $T_n = \theta(\widehat{F}_n)$. Let

$$s(F) \equiv s(T_n; F)$$

denote the statistical summary of interest; it is written as a function of F as the statistical properties of T_n are entirely dictated by F.

The quantity $s(F)$ can again usually be expressed in terms of an integral with respect to F

$$s(F) = \int s(t(\mathbf{y}))dF(\mathbf{y}).$$

for function $t(.)$ that defines the estimator. Occasionally, this expression can be computed analytically.

The Bootstrap

The key idea of the bootstrap is to replace calculations wrt F by calculations wrt $\widehat{F}_n$, and to compute

$$s(\widehat{F}_n)$$

numerically, that is

$$s(\widehat{F}_n) = \int s(t(\mathbf{y}))d\widehat{F}_n(\mathbf{y}).$$

The Bootstrap

For a random sample, $y_1, \ldots, y_n$, the bootstrap proceeds as follows:

1. Set B (the number of bootstrap resamples)
2. For $b = 1, \ldots, B$,
   (a) generate a sample of size n $y_1^{(b)}, \ldots, y_n^{(b)}$ at random *with replacement* from $\widehat{F}_n$
   (b) form the statistic of interest $t_n^{(b)}$
3. Summarize the resampled estimates

$$t_n^{(1)}, \ldots, t_n^{(B)}$$

using the desired statistical summary, $s(.)$.

Example: Sample mean standard error

Consider data

-0.1568	0.4439	-0.7865	-1.6531	-0.6037
-0.6231	0.9061	-0.8215	-1.2829	-0.3538

with sample mean -0.4932 and standard deviation

$$S = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2} = 0.7184$$

yielding the estimated standard error of the mean (s.e.m.)

$$\widehat{\text{se}}(\widehat{\theta}) = \frac{0.7184}{\sqrt{10}} = 0.2272.$$

Example: Sample mean standard error

If the presumption of normality is made, then this is the *plug-in* estimate of the standard error

- the *true* s.e.m. is $\mathrm{se}(\widehat{\theta}) = \sigma/\sqrt{n}$;
- we estimate σ by the standard deviation under $\widehat{F}_n$.
  - $\widehat{F}_n$ is the discrete distribution that places mass $1/n$ at each of $y_1, \ldots, y_n$;
  - we then compute the 'theoretical' variance of this discrete distribution.
- this yields $\widehat{\mathrm{se}}(\widehat{\theta}) = S/\sqrt{n}$

Example: Sample mean standard error

The bootstrap version of this calculation replaces an analytic calculation by a numerical one. Proceed as follows: set $B = 200$. For $b = 1, \ldots, B$, resample n values without replacement, and compute the appropriate summary. For example, for $b = 1$,

0.9061	0.4439	0.4439	-1.6531	-1.2829
-0.1568	-0.1568	-1.6531	0.4439	0.4439

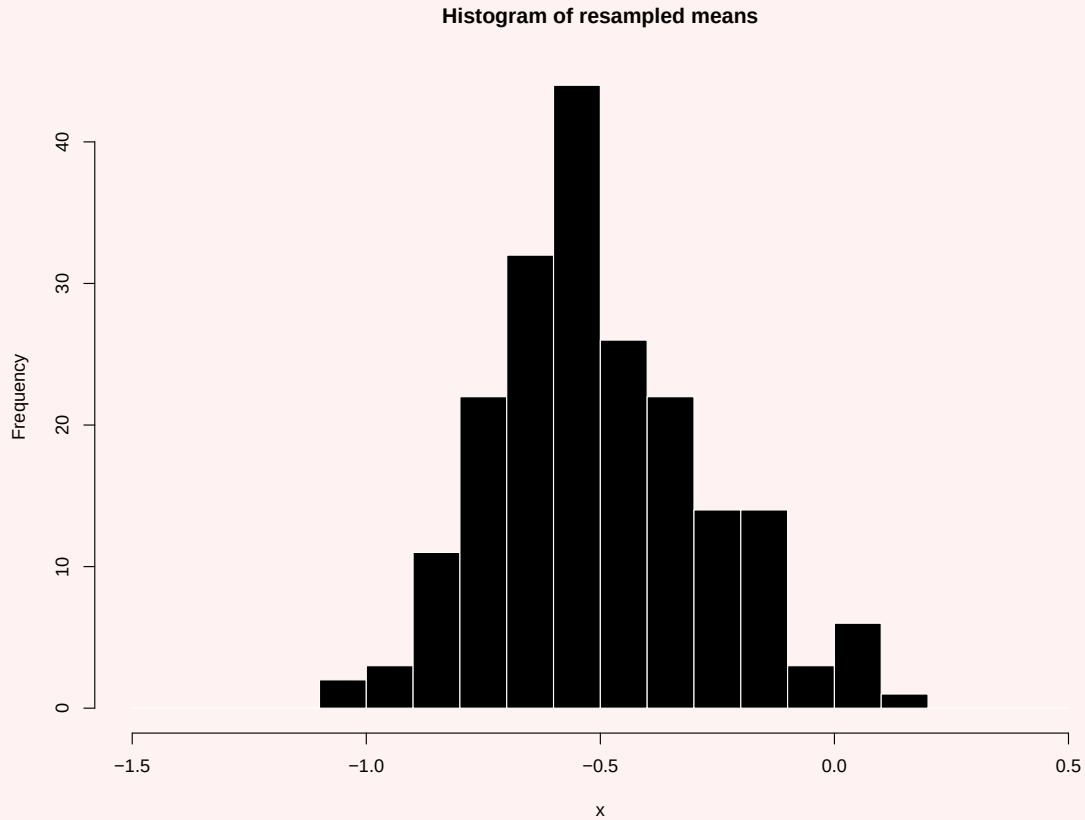
with sample mean -0.2221 and standard deviation 0.9097.

Example: Sample mean standard error

b	Mean	Std. Dev.
1	-0.2221	0.9097
2	-0.7630	0.1962
3	-0.1794	0.6869
4	-0.5606	0.2777
5	-0.8929	0.3883
6	-0.5359	0.5958
7	-0.6969	0.6149
8	-0.4056	0.9144
⋮	⋮	⋮

Example: Sample mean standard error

We now have 200 (resampled) sample means, with standard deviation $0.2327 \doteq S/\sqrt{n}$, where S is the original sample s.d..



Example: Sample kurtosis standard error

The sample kurtosis

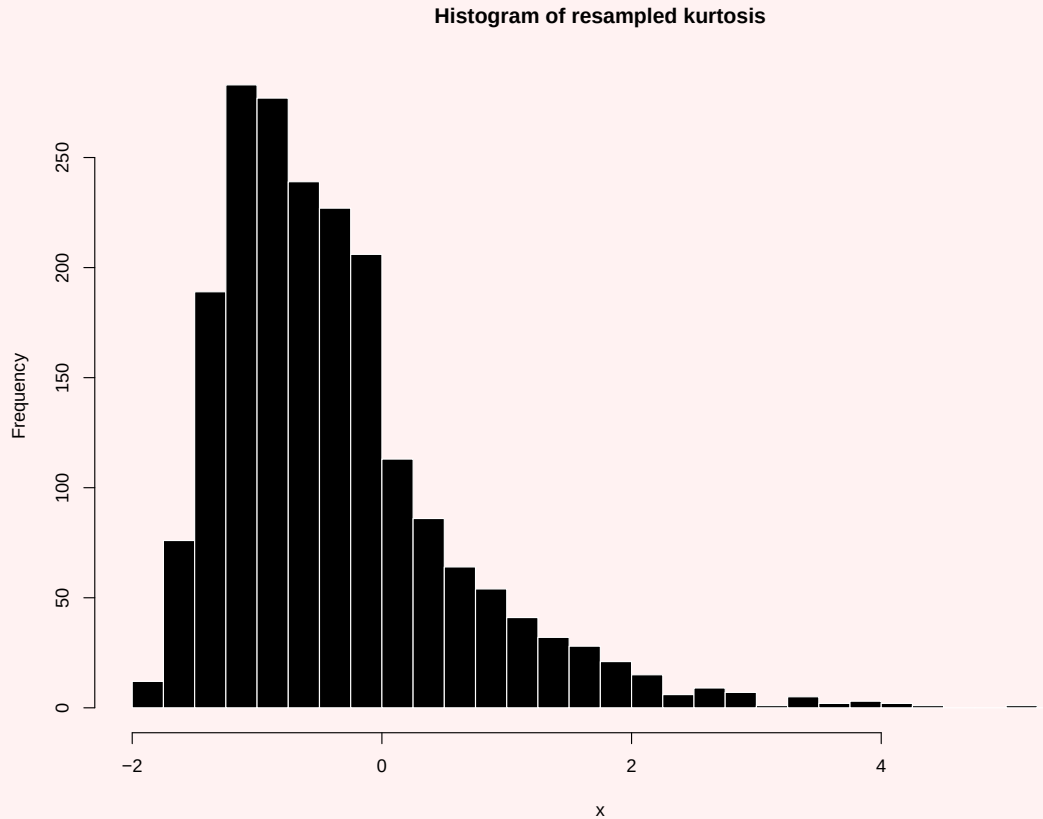
$$t_n = \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2\right)^2} - 3$$

is a statistic that records the heavy-tailed nature of a sample.

There is no simple formula for the standard error for this statistic (although an asymptotic expression can be derived).

Example: Sample kurtosis standard error

2000 (resampled) sample kurtosis values, with standard deviation 0.9689.



The Bootstrap: notation

Let $T_n = \widehat{\theta}$ be the statistic of interest, and let

$$s(\widehat{\theta}) \qquad \widehat{s}_B(\widehat{\theta}^{(*)})$$

respectively denote the true and bootstrap estimated value of summary quantity s of the sampling distribution of $\widehat{\theta}$.

The Bootstrap: notation

For example, for the normal mean standard error, $\widehat{\theta} = \overline{y}$,

$$s(\widehat{\theta}) = \mathrm{se}(\widehat{\theta}) = \frac{\sigma}{\sqrt{n}} \qquad \widehat{\mathrm{se}}(\widehat{\theta}) = \frac{S}{\sqrt{n}}$$

and

$$\widehat{s}_B(\widehat{\theta}^{(*)}) = \widehat{\mathrm{se}}_B(\widehat{\theta}^{(*)}) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\overline{y}^{(b)} - \overline{y}^{(*)})^2}$$

with

$$\overline{y}^{(*)} = \frac{1}{B}\sum_{b=1}^{B}\overline{y}^{(b)} = \widehat{\theta}^{(*)} \qquad \overline{y}^{(b)} = \frac{1}{n}\sum_{i=1}^{n} y_i^{(b)}.$$

Example: Sample mean standard error

For the data set above, the following bootstrap standard errors were obtained for different values of B

B	50	100	200	500	1000	10000	20000
$\hat{\theta}^{(*)}$	-0.5095	-0.5097	-0.4820	-0.4968	-0.5014	-0.4942	-0.4953
$\widehat{\text{se}}_B(\hat{\theta}^{(*)})$	0.2362	0.2353	0.2415	0.2263	0.2309	0.2283	0.2272

where recall that the exact standard error of the mean is

$$\widehat{\text{se}}(\hat{\theta}) = \frac{S}{\sqrt{n}} = 0.2272.$$

▸ Further details and Examples

The Bootstrap is a Bayesian procedure

Recall Bayesian nonparametric inference based on the Dirichlet process:

- $Y_1, \ldots, Y_n \sim \widetilde{f}$ (conditionally independent)
- *a priori* $\widetilde{f} \sim DP(\alpha, G_Y)$.
  - $\alpha > 0$
  - $G_Y(.)$ some distribution on $\mathbb{R}$

The Bootstrap is a Bayesian procedure

- *a posteriori*

$$\widetilde{f} \sim DP(\alpha^\star, G_Y^\star)$$

where

$$\begin{aligned} \alpha^\star &= \alpha + n \\ G_Y^\star &= \frac{\alpha G_Y + \sum_{j=1}^{n} \delta_{y_j}}{\alpha + n} \end{aligned}$$

The Bootstrap is a Bayesian procedure

- For K-partition of $\mathbb{R}$, $\{B_1, B_2, \ldots, B_K, B_{K+1}\}$, with associated probabilities

$$\boldsymbol{p} = (p_1, p_2, \ldots, p_K, p_{K+1})^\top$$

  - in the *prior*

$$\boldsymbol{p} \sim Dirichlet(K; \alpha_1, \alpha_2, \ldots, \alpha_K, \alpha_{K+1})$$

    and

$$\alpha_k = \alpha G_Y(B_k)$$

  - in the *posterior*

$$\boldsymbol{p} \sim Dirichlet(K; \alpha_1^\star, \alpha_2^\star, \ldots, \alpha_K^\star, \alpha_{K+1}^\star)$$

    and

$$\alpha_k^\star = \alpha^\star G_Y^\star(B_k)$$

The Bootstrap is a Bayesian procedure

The Dirichlet Process is a distribution on distributions that are discrete (with probability 1), that is,

- mass function of the form

$$\widetilde{f}(y) = \sum_{i=1}^{\infty} \pi_i \delta_{Y_i}(y)$$

- random locations $Y_1, Y_2, \ldots \sim G_Y$;
- random probabilities $\pi_1, \pi_2, \ldots$ constructed according to the stick-breaking mechanism, that is, defined by the single parameter α.

The Bootstrap is a Bayesian procedure

We can easily produce a i.i.d sample $Y_1, Y_2, \ldots, \sim \widetilde{f}(.)$ as it is merely a discrete distribution: this construction ensures that $\{Y_n\}$ is an *exchangeable* sequence by de Finetti's theorem.

- $\widetilde{f} \sim DP(\alpha, G_Y)$
- $Y_1, Y_2, \ldots, Y_n \mid \widetilde{f} \sim \widetilde{f}$, independently.

The Bootstrap is a Bayesian procedure

In light of observed data $y_1, \ldots, y_n$, if the prior is $DP(\alpha, G_Y)$, then the posterior is

$$DP\left(\alpha + n, \frac{\alpha G_Y + \sum_{i=1}^{n} \delta_{y_i}(.)}{\alpha + n}\right).$$

Denote the posterior parameters where

$$\begin{aligned} \alpha^\star &= \alpha + n \\ G_Y^\star &= \frac{\alpha G_Y + \sum_{j=1}^{n} \delta_{y_j}(.)}{\alpha + n} \end{aligned}$$

The Bootstrap is a Bayesian procedure

Recall the predictive calculation given by de Finetti:

$$p_n(y_{(n+1):(n+m)}) = \int \prod_{i=n+1}^{n+m} f(y_i)\, \pi_n(df).$$

To sample from p_n, we

- sample $f \sim \pi_n$;
- sample $Y_{n+1}, \ldots, Y_{n+m}$ independently from f.

We sample a random f from π_n, and then obtain a sample $Y_{n+1}, \ldots, Y_{n+m}$ from the predictive distribution using the randomly drawn f.

The Bootstrap is a Bayesian procedure

This may be achieved using the Polya urn.

STEP 1: Sample $Y_{n+1} \sim G_Y^\star$.

STEP 2: For $j = 2, \ldots, m$, sample Y_{n+j} from

$$\frac{\alpha^\star}{\alpha^\star + j - 1} G_Y^\star(.) + \frac{1}{\alpha^\star + j - 1} \sum_{l=1}^{j-1} \delta_{Y_{n+l}}(.)$$

that is, given $Y_{n+1}, Y_{n+2}, \ldots, Y_{n+j-1}$, sample

$$Y_{n+j} \sim \begin{cases} G_Y^\star & \text{w.p. } \alpha^\star/(\alpha^\star + j - 1) \\ Y_{n+l} & \text{w.p. } 1/(\alpha^\star + j - 1), l = 1, \ldots, j-1. \end{cases}$$

The Bootstrap is a Bayesian procedure

The posterior mean of the DP is the measure

$$\frac{\alpha G_Y + \sum_{i=1}^{n} \delta_{y_i}(.)}{\alpha + n}.$$

Instead of using the full Polya urn scheme, consider a *plug-in* procedure that replaces a sample of f by this posterior mean.

Independently for $j = n+1, \ldots, n+m$,

- w.p. $\alpha/(\alpha+n)$: draw from G_Y;
- w.p. $1/(\alpha+n)$: draw y_i, for each $i = 1, \ldots, n$.

The Bootstrap is a Bayesian procedure

In the limit as $\alpha \longrightarrow 0$, this procedure becomes

- sample y_i w.p. $1/n$, $i = 1, \ldots, n$

independently for $j = n+1, \ldots, n+m$. This is identical to the bootstrap.

Therefore we can consider bootstrap calculations as Monte Carlo calculations made with respect to the *predictive distribution* computed for a Dirichlet process prior and posterior, *in the limit as* $\alpha \longrightarrow 0$, using a *plug-in* approach.

Predictive distributions

In Bayesian inference the predictive distribution

$$p_n(y_{n+1}) \equiv p(y_{n+1}|y_{1:n})$$

is the natural estimator of the true probability measure of a single element of the infinitely exchangeable sequence.

Predictive distributions

Recall the frequentist justification of maximum likelihood: in a potentially mis-specified model $f(y;\theta)$, we identify the true value of θ, θ_0 as

$$\theta_0 = \arg\min_\theta KL(f_0, f(.;\theta)) = \arg\min_\theta \int \log\left\{\frac{f_0(y)}{f(y;\theta)}\right\} f_0(y)\, dy$$

Predictive distributions

The corresponding estimator is obtained when we replace the integral by a 'Monte Carlo' version based on an i.i.d. sample

$$\widehat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log f(y_i; \theta) = \arg\max_{\theta} \ell_n(\theta)$$

where the (Monte Carlo) sample is the data drawn (by definition) from f_0. In alternative form, $\widehat{\theta}$ is the solution to the estimating equation

$$\dot{\ell}_n(\theta) = \sum_{i=1}^{n} \frac{\partial \log f(y_i; \theta)}{\partial \theta} = 0$$

Predictive distributions

A Bayesian version of the calculation replaces the original sample by a sample from the predictive distribution

$$p_n\big(y_{(n+1):(n+m)}\big).$$

However, we are not restricted to use the 'score' function as the basis of an estimation procedure

Predictive distributions

- we may use any loss function $L(y, \theta)$ say, and define the Bayesian estimator as

$$\arg\min_{\theta} \int L(y, \theta) p_n(y)\, dy = \arg\min_{\theta} \mathbb{E}_{p_n}[L(Y, \theta)]$$

- this is a valid fully Bayesian estimator as it minimizes an expected posterior loss;
- via this route, we may achieve fully Bayesian inference in a semi-parametric fashion.

The Bayesian Bootstrap

The *Bayesian bootstrap* replaces the $1/n$ weights in the bootstrap by repeated draws of $W = (W_1, \ldots, W_n)$

$$W \sim Dirichlet(n, 1, 1, \ldots, 1)$$

where

$$\mathbb{E}[W_i] = \frac{1}{n}$$

and uses this as representation of the predictive distribution $p_n(.)$.

The Bayesian Bootstrap

That is, $p_n(.)$ is the predictive distribution derived from a Dirichlet process model, in the limiting case with $\alpha \longrightarrow 0$, so that, given that we have an observed draw

$$w = (w_1, \ldots, w_n)$$

of $W \sim Dirichlet(n, 1, 1, \ldots, 1)$, the predictive distribution takes the form

$$p_n(y) = \sum_{i=1}^{n} w_i \delta_{\{y_i\}}(y).$$

This yields the calculation

$$\mathbb{E}_{p_n}[L(Y,\theta)] = \sum_{i=1}^{n} w_i L(y_i,\theta)$$

and in the specific case of the log-density loss

$$\mathbb{E}_{p_n}[L(Y,\theta)] = -\sum_{i=1}^{n} w_i \ell(y_i;\theta)$$

Hence, we must perform the calculation of

$$\theta_{\mathrm{OPT}} = \arg\max_{\theta} \sum_{i=1}^{n} w_i \ell(y_i; \theta)$$

to minimize the loss.

The quantity θ_{OPT} is a functional of the Dirichlet process posterior, and so we may build up a posterior distribution for it by repeatedly sampling the Dirichlet weights, and recomputing θ_{OPT} for each sample.

Example

Example: Bayesian bootstrap

See `knitr 21`

Langevin algorithm

The *Langevin algorithm* uses a *diffusion approximation* to generate the a process that has the target posterior distribution as its invariant measure.

- new states are proposed using *Langevin dynamics*, based on the gradient of the target pdf;
- due to the necessary time discretization, proposed values must be subjected to a Metropolis–Hastings accept/reject step in order for the target to be preserved.

Langevin algorithm

Denote the target pdf $\pi(x)$. In the notation of stochastic differential equations (SDEs), consider the diffusion process $\{X_t, t > 0\}$ where

$$dX_t = \frac{1}{2} S(X_t)dt + dW_t$$

where $\{W_t\}$ is Brownian motion, and

$$S(\mathbf{x}) = \frac{d}{d\mathbf{x}} \log \pi(\mathbf{x}) = \nabla \log \pi(\mathbf{x}).$$

This is the *Langevin SDE*, and it can be shown that the invariant measure of this continuous time process is $\pi(\mathbf{x})$.

An initial condition $X_0 = x_0$ is specified.

Langevin algorithm

A standard scheme for simulating this diffusion is given by an approximation based on the time-discretization

$$\{X_{k\delta}, k = 1, 2, \ldots\}$$

for some time-step $\delta > 0$ where $X_0 = x_0$, and then for $k = 0, 1, 2, \ldots$

$$X_{k+1} = X_k + \frac{\delta}{2} S(X_k) + \sqrt{\delta} Z_k$$

where $\{Z_k, k = 1, 2, \ldots\}$ are an iid $Normal_d(0, \mathbf{I}_d)$ sequence.

Langevin algorithm

The approximation induced by time-discretization does not quite preserve $\pi(x)$ as the invariant distribution. However this can be easily corrected using a Metropolis-Hastings step: that is, the *proposal*

$$X^\star_{k+1} \sim Normal_d(X_k + \delta S(X_k)/2, \delta \mathbf{I}_d)$$

is *accepted* with probability

$$\min\left\{1, \frac{\pi(X^\star_{k+1})}{\pi(X_k)} \frac{q(X^\star_{k+1}, X_k)}{q(X_k, X^\star_{k+1})}\right\}$$

where

$$q(\mathbf{x}, \mathbf{y}) \propto \exp\left\{-\frac{1}{2\delta}(\mathbf{y} - \mathbf{x} - \delta S(\mathbf{x})/2)^\top (\mathbf{y} - \mathbf{x} - \delta S(\mathbf{x})/2)\right\}$$

Langevin algorithm

In this algorithm, the time-step δ can be tuned to optimize the effective sample size of the sampler:

- as δ varies the acceptance rate of the Metropolis step varies;
- some theory suggests that the optimal acceptance rate is around 0.5-0.6 in tractable problems, but this is problem-dependent.

Langevin algorithm

This algorithm is known as the *Metropolis-adjusted Langevin algorithm* (MALA)

Example: MALA

See `knitr 22`

Hamiltonian MCMC

Hamiltonian Monte Carlo is another method that appeals to the dynamics of physical systems to perform sampling from a target distribution.

Consider two d-dimensional vectors that describe the motion of an object in $\mathbb{R}^d$.

- $x = (x_1, \ldots, x_d)$ denotes the *position* of the object
- $v = (v_1, \ldots, v_d)$ denotes the *momentum* (proportional to the *velocity*) of the object.

Let $z = (x, v)$ be the 2d concatenated vector.

Hamiltonian MCMC

Hamiltonian dynamics is formulated via the *Hamiltonian*, denoted $H(z) \equiv H(x, v)$, which is a function of $z = (x, v)$ but *not* t, and which describes how z changes in time via the differential form

$$\frac{dz}{dt} = \begin{bmatrix} \mathbf{0}_d & \mathbf{I}_d \\ -\mathbf{I}_d & \mathbf{0}_d \end{bmatrix} \frac{\partial H(z)}{\partial z} = \mathbf{D} \frac{\partial H(z)}{\partial z}$$

say, that is

$$\frac{dx}{dt} = \frac{\partial H(x, v)}{\partial v} \qquad \frac{dv}{dt} = -\frac{\partial H(x, v)}{\partial x}.$$

These are *Hamilton's Equations*.

Hamiltonian MCMC

Attention focusses on Hamiltonians that satisfy

$$H(x, v) = H(x, -v)$$

and are assumed to be separable

$$H(x, v) = U(x) + K(v)$$

so that $K(v) = K(-v)$.

Note that replacing v by $-v$ in the equations yields

$$\frac{dx}{dt} = \frac{\partial H(x, -v)}{\partial(-v)} = -\frac{\partial H(x, v)}{\partial v}$$

as $H(x, -v) = H(x, v)$, by assumption, and

$$-\frac{dv}{dt} = \frac{d(-v)}{dt} = -\frac{\partial H(x, -v)}{\partial x} = -\frac{\partial H(x, v)}{\partial x}.$$

Thus after changing $v \longrightarrow -v$, we have

$$\frac{dx}{dt} = -\frac{\partial H(x, v)}{\partial v}$$

and

$$\frac{dv}{dt} = \frac{\partial H(x, v)}{\partial x}$$

which are identical to the original equations, but with the signs changed on the right hand side.

Therefore changing $v \longrightarrow -v$ is the same as running the dynamics with time *reversed*.

Hamiltonian MCMC

- $U(x)$ is termed the *potential energy*
- $K(v)$ is termed the *kinetic energy*, where typically

$$K(v) = \frac{1}{2} v^\top \mathbf{M}^{-1} v$$

where $\mathbf{M}$ is a positive definite, symmetric matrix.

Hamiltonian MCMC

The Hamiltonian, and Hamilton's equations, define the evolution of the system in continuous time: the objective is to find the solution $z(t) = (x(t), v(t))$ to the equations for all t.

We consider the state of the system at times t and $t + s$

$$z(t) \longrightarrow z(t + s)$$

that is

$$(x(t), v(t)) \longrightarrow (x(t + s), v(t + s)).$$

Hamiltonian MCMC

Denote by $\mathcal{H}_s$ the mapping taking $z(t)$ to $z(t+s)$, that is

$$\mathcal{H}_s(z(t)) = z(t+s).$$

The mapping is 1-1 with inverse mapping $\mathcal{H}_{-s}$, say. That is

$$\mathcal{H}_{-s}(z(t+s)) = z(t)$$

which follows from the *time reversibility* established above.

Hamiltonian MCMC

Conservation: Note that

$$\frac{dH(z)}{dt} = 0$$

as

$$\frac{dH(z)}{dt} = \sum_{j=1}^{d} \frac{\partial H(z)}{\partial x_j} \frac{dx_j}{dt} + \sum_{j=1}^{d} \frac{\partial H(z)}{\partial v_j} \frac{dv_j}{dt}$$

$$= \sum_{j=1}^{d} \frac{\partial H(z)}{\partial x_j} \frac{\partial H(z)}{\partial v_j} - \sum_{j=1}^{d} \frac{\partial H(z)}{\partial v_j} \frac{\partial H(z)}{\partial x_j} = 0$$

This means that $H(z) = H(x, v)$ is *constant over time*.

Preservation of volume: If $\mathcal{B} \subset \mathbb{R}^{2d}$ has volume $V(\mathcal{B})$, and for all s

$$\mathcal{H}_s(\mathcal{B}) = \{\mathcal{H}_s(z) : z \in \mathcal{B}\}$$

is the image of $\mathcal{B}$, then the volume of $\mathcal{H}_s(\mathcal{B})$ is also $V(\mathcal{B})$. This feature is a consequence of the *symplectic* property of Hamiltonian dynamics, specifically that

$$\dot{\mathcal{H}}_s^\top \mathbf{D}^{-1} \dot{\mathcal{H}}_s = \mathbf{D}^{-1}$$

where $\dot{\mathcal{H}}_s$ is the Jacobian associated with the map $\mathcal{H}_s$.

Hamiltonian MCMC

The preservation of volume under the mapping $\mathcal{H}_s$ will be important when it is utilized as a proposal mechanism in MCMC:

- recall that a multivariate transformation that computes the distribution of transforms of continuous random variables require 'preservation of probability' under the transformation.
- 'preservation of volume' corresponds to a transform that has Jacobian 1.

Discrete approximation

Hamiltonian dynamics can be approximated using a discrete time approach as for Langevin dynamics. We consider

$$z_k \equiv z(k\delta), k = 1, 2, \ldots$$

for time-step $\delta > 0$. Then the dynamics equation

$$\frac{dz}{dt} = \mathbf{D}\frac{\partial H(z)}{\partial z}$$

becomes in an Euler approximation

$$z_{k+1} = z_k + \delta \mathbf{D}\frac{\partial H(z)}{\partial z}\bigg|_{z=z_k}.$$

Discrete approximation

That is, if $H(x, v) = U(x) + K(v)$

$$x_{k+1} = x_k + \delta \frac{\partial H(x, v)}{\partial v}\bigg|_{z=(x_k, v_k)} = x_k + \delta \dot{K}(v_k)$$

$$v_{k+1} = v_k - \delta \frac{\partial H(x, v)}{\partial x}\bigg|_{z=(x_k, v_k)} = v_k - \delta \dot{U}(x_k)$$

Discrete approximation

This method can be improved by amending the second equation and considering *sequential* updating:

$$x_{k+1} = x_k + \delta \dot{K}(v_k)$$

$$v_{k+1} = v_k - \delta \dot{U}(x_{k+1})$$

Discrete approximation

A further amendment using half-steps gives further improvement: the *leapfrog* method uses the updates

$$v_k^* = v_k - \frac{\delta}{2}\dot{U}(x_k)$$

$$x_{k+1} = x_k + \delta\dot{K}(v_k^*)$$

$$v_{k+1} = v_k^* - \frac{\delta}{2}\dot{U}(x_{k+1})$$

where we consider

$$v_k^* \equiv v((k + 1/2)\delta).$$

Discrete approximation

Note

These discrete time dynamics are also

- reversible,
- volume-preserving.

Error analysis shows that the leapfrog method provides the closest approximation to the continuous time dynamics.

Hamiltonian MCMC

Assuming the separable case, we consider joint pdf $\pi_{X,V}(x, v)$

$$\pi_{X,V}(\mathbf{x}, v) \propto \exp\{-H(\mathbf{x}, v)\} = \exp\{-U(\mathbf{x}) - K(v)\}$$

corresponding to independence of corresponding random variables X and V

$$\pi_X(\mathbf{x}) \propto \exp\{-U(\mathbf{x})\} \qquad \text{and} \qquad \pi_V(v) \propto \exp\{-K(v)\}\,.$$

If the marginal $\pi_X(x)$ is our true target, then V is merely an *auxiliary variable*.

We are free to choose the distribution $\pi_V(v)$, and a typical choice is the multivariate Normal

$$\pi_V(v) \propto \exp\left\{-\frac{1}{2}v^\top \mathbf{M}^{-1} v\right\}$$

so that as before

$$K(v) = \frac{1}{2}v^\top \mathbf{M}^{-1} v.$$

If $\mathbf{M} = \text{diag}(m_1, \ldots, m_d)$, then

$$K(v) = \frac{1}{2} \sum_{j=1}^{d} \frac{v_j^2}{m_j}$$

which corresponds to an assumption that $V_j \sim Normal(0, m_j)$ for $j = 1, \ldots, d$ are independent. Here

$$\dot{K}(v) = \mathbf{M}^{-1} v = \begin{bmatrix} \frac{v_1}{m_2} \\ \vdots \\ \frac{v_d}{m_d} \end{bmatrix}$$

Hamiltonian MCMC

The basic Hamiltonian MCMC algorithm proceeds using the following Metropolis accept/reject approach: we construct an MCMC move

$$\left(x_k, v_k\right) \longrightarrow \left(x_{k+1}, v_{k+1}\right)$$

as follows:

(I) Generate $v_1' \sim Normal_d\left(\mathbf{0}_d, \mathbf{M}\right)$.

(II) Perform L dynamics updates with time-step δ: for example, for the leapfrog updates,

(i) set $v_1' , x_1' = x_k$;

(ii) for $l = 1, \ldots, L-1$

$$v_l^* = v_l' - \frac{\delta}{2}\dot{U}(x_l')$$

$$x_{l+1}' = x_l' + \delta \dot{K}(v_l^*)$$

$$v_{l+1}' = v_l^* - \frac{\delta}{2}\dot{U}(x_{l+1}');$$

(iii) set $x_{k+1}^* = x_L'$ and $v_{k+1}^* = -v_L'$.

(III) Accept (x^*_{k+1}, v^*_{k+1}) with probability

$$\min\left\{1, \frac{\pi_{X,V}(x^*_{k+1}, v^*_{k+1})}{\pi_{X,V}(x, v)}\right\}$$

where

$$\frac{\pi_{X,V}(x^*_{k+1}, v^*_{k+1})}{\pi_{X,V}(x, v)} = \exp\{-H(x^*_{k+1}, v^*_{k+1}) + H(x_k, v_k)\}.$$

Note that the proposal in step (II) is reversible by construction, so the proposal mechanism does not appear in the acceptance probability as it cancels in numerator and denominator of the ratio.

Note

- As for Langevin dynamics and MALA, the quantities L and δ can be adjusted to obtain the optimal effective sample size.
- In step (II), due to conservation, we would anticipate that

$$H(x_{k+1}^*, v_{k+1}^*) \approx H(x_k, v_k)$$

as the exact dynamics would keep $H(x, v)$ *constant*.

However, the generation in step (I) prevents this by allowing v to be changed independently according to $\pi_V(v)$

Hamiltonian MCMC

Example: Hamiltonian MCMC

See `knitr 23`

Appendix
Multiple chain Moves

Appendix: Multiple chain Moves

Back to slides

Multiple chain MCMC: Types of move

1. **Mutation:** Select $m \in \{1, \ldots, M\}$, and perform a MH move using proposal density $q_m(x, z)$. If the current value for x_m is x, and the proposed value is z, accept the proposed point with probability

$$\alpha = \min\left\{1, \frac{\pi_m(z) q_m(z, x)}{\pi_m(x) q_m(x, z)}\right\}$$

   In the context of the vector $\boldsymbol{x}$, this move is essentially a Gibbs sampler move, as the full conditional for X_m is π_m.

Back to slides

Multiple chain MCMC: Types of move

2. **Exchange:** Select $l, m \in \{1, \ldots, M\}$, and propose the swap

$$\mathbf{x}_l \longleftrightarrow \mathbf{x}_m$$

Accept this proposal with probability

$$\alpha = \min\left\{1, \frac{\pi_l(\mathbf{x}_m)\pi_m(\mathbf{x}_l)}{\pi_l(\mathbf{x}_l)\pi_m(\mathbf{x}_m)}\right\}.$$

Back to slides

Multiple chain MCMC: Types of move

In the case where all the π_m are equal to the target π, we can utilize the following move:

3. **Snooker:** Select $l, m \in \{1, \ldots, M\}$, and, in an attempt to update x_l, make the proposal z

$$z = x_m + u(x_l - x_m)$$

where u is a random variate sampled from the density

$$f(u) \propto \pi(x_m + u(x_l - x_m))$$

and then update x_l to z.

Back to slides

Multiple chain MCMC: Types of move

This move can be utilized in the case of *vector* state spaces. If each x is d-dimensional, then the move becomes

3*. **Snooker:** Select $l, m \in \{1, \ldots, M\}$, and, in an attempt to update x_l, make the proposal z

$$z = x_m + u(x_l - x_m)$$

where u is a scalar random variate sampled from the density

$$f(u) \propto |u|^{d-1} \pi(x_m + u(x_l - x_m))$$

and then update x_l to z.

Back to slides

Multiple chain MCMC: Types of move

A related move returns to basic ideas from optimization:

4. **Local optimum:** Select $l, m \in \{1, \ldots, M\}$, and, in an attempt to update x_l, proceed as follows: starting at x_m, use conjugate gradient or steepest descent to find the local mode of π, and denote this $\widehat{x}_m$.

   Then make the proposal z

   $$z = \widehat{x}_m + u(x_l - \widehat{x}_m)$$

   where u is a scalar random variate sampled from the density

   $$f(u) \propto |u|^{d-1} \pi(\widehat{x}_m + u(x_l - \widehat{x}_m))$$

   and then update x_l to z.

Back to slides

Multiple chain MCMC: Types of move

Again in the d-vector case, we have the final move type, inspired by ideas from genetic algorithms:

5. **Crossover:** Select $l, m \in \{1, \ldots, M\}$, and then select j uniformly from $\{1, \ldots, d-1\}$. Perform a crossover at position j, that is, attempt to replace x_l and x_m by

$$x_l^{\text{new}} = (x_{l1}, \ldots, x_{lj}, x_{mj+1}, \ldots, x_{md})$$

$$x_m^{\text{new}} = (x_{m1}, \ldots, x_{mj}, x_{lj+1}, \ldots, x_{ld})$$

that is, exchange the $(j+1, \ldots, d)$ portions of each vector.

Accept this proposal with probability

$$\alpha = \min\left\{1, \frac{\pi_l(x_l^{\text{new}})\pi_m(x_m^{\text{new}})}{\pi_l(x_l)\pi_m(x_m)}\right\}.$$

Back to slides

Appendix
Bootstrap examples and Extensions

Appendix: Bootstrap examples and Extensions

Back to slides

The Bootstrap: choosing B

For a sample of size n, there are

$$M = \binom{2n-1}{n}$$

possible distinct resampled data sets:

n	Number of distinct samples
10	9.238×10^4
20	6.892×10^{10}
50	5.045×10^{28}
100	4.527×10^{58}

How the value of B is chosen depends on the type of calculation being done, and the nature of the underlying data generating mechanism.

The Bootstrap: choosing B

Note that $B = \infty$ returns the plug-in estimate.

- For the s.e.m., at $B = \infty$

$$\widehat{\text{se}}_\infty(\widehat{\theta}^{(*)}) = \sqrt{\sum_{j=1}^{M} w_j(\overline{y}_j - \overline{y}_\cdot)^2}$$

  where $w_j, j = 1, \ldots, M$ represents the relative frequency of the resampled data sets yielding mean $\overline{y}_j$, and $\overline{y}_\cdot$ denotes the weighted mean

$$\overline{y}_\cdot = \frac{1}{M} \sum_{j=1}^{M} w_j \overline{y}_j.$$

- This quantity arises from the *exact* sampling distribution of the $\widehat{\theta}$, given $\widehat{F}_n$.
- Recall that there may not be a simple analytical formula for the required statistic.

The Bootstrap: choosing B

Let $\mathrm{se}(\widehat{\theta})$ denote the standard error functional for statistic $\widehat{\theta}$, and, suppressing the dependence on $\widehat{\theta}$, let

- $\widehat{\mathrm{se}}_B$ denote the estimate of the standard error from B bootstrap resamples
- $\widehat{\mathrm{se}}_\infty$ denote the ideal estimated bootstrap standard error at $B = \infty$. Recall that $\widehat{\mathrm{se}}_\infty \equiv \widehat{\mathrm{se}}(\widehat{\theta})$ is the plug-in estimated standard error of $\widehat{\theta}$ (that is, the standard deviation of the sampling distribution of $\widehat{\theta}$ under $\widehat{F}_n$).

It can be shown that

$$\mathbb{E}_F[\widehat{\mathrm{se}}_B^2] = \mathbb{E}_F[\widehat{\mathrm{se}}_\infty^2]$$

and

$$\mathrm{Var}_F[\widehat{\mathrm{se}}_B^2] \geqslant \mathrm{Var}_F[\widehat{\mathrm{se}}_\infty^2]$$

The Bootstrap: choosing B

Example: $\mathcal{N}(\mu, \sigma^2)$

The sample variance of $\widehat{\theta} = \overline{Y}$ is

$$\text{se}(\widehat{\theta})^2 = \frac{\sigma^2}{n}$$

estimated by

$$\widehat{\text{se}}(\widehat{\theta})^2 = \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2}{n} = \frac{(n-1)s^2}{n^2} = \frac{S^2}{n}.$$

But, by standard theory, under distribution $F(y) \equiv \mathcal{N}(\mu, \sigma^2)$,

$$\frac{(n-1)s^2}{\sigma^2} \sim \chi^2_{n-1}.$$

The Bootstrap: choosing B

Example: $\mathcal{N}(\mu, \sigma^2)$

Hence, as $\widehat{\text{se}}_\infty \equiv \widehat{\text{se}}$, we have

$$\mathbb{E}_F[\widehat{\text{se}}^2_\infty] = \frac{(n-1)\sigma^2}{n^2} \qquad \text{Var}_F[\widehat{\text{se}}^2_\infty] = \frac{2(n-1)\sigma^4}{n^4}$$

These results are *not* identical to those obtained taking expectations under $\widehat{F}_n$.

The larger B gets, we expect to see

$$\text{Var}_F[\widehat{\text{se}}^2_B] \longrightarrow \text{Var}_F[\widehat{\text{se}}^2_\infty]$$

Note that we cannot compute the limiting value as $B \longrightarrow \infty$ in many cases.

The Bootstrap: choosing B

Example: Simulation study

Suppose F is $\mathcal{N}(0,1)$, so that

$$\mathbb{E}_F[\widehat{\text{se}}_\infty^2] = \frac{(n-1)}{n^2} \qquad \text{Var}_F[\widehat{\text{se}}_\infty^2] = \frac{2(n-1)}{n^4}.$$

For $n = 10$, these values are 0.090 and 0.0424^2 respectively.

For different values of B, the following values were obtained for the mean and standard deviation of $\widehat{\text{se}}_B^2$ computed across 2000 simulated $\mathcal{N}(0,1)$ data sets:

B	50	100	200	500	1000	10000	20000	∞
Mean	0.0910	0.0910	0.0907	0.0906	0.0907	0.0907	0.0907	0.0900
s.d.	0.0469	0.0450	0.0437	0.0432	0.0428	0.0427	0.0427	0.0424

The Bootstrap: choosing B

Example: Simulation study

For $n = 50$, these values are 0.0196 and 0.0040^2 respectively.

B	50	100	200	500	1000	10000	20000	∞
Mean	0.0195	0.0196	0.0197	0.0197	0.0197	0.0197	0.0197	0.0196
s.d.	0.0056	0.0049	0.0044	0.0042	0.0041	0.0040	0.0040	0.0040

The Bootstrap: choosing B

Note: the estimator used here for σ^2 is

$$S^2 = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \overline{Y})^2$$

that is, the *ML* estimator, not the *unbiased* estimator.

This estimator yields the “correct" variance for the bootstrap estimator at $B = \infty$, as S^2 is the correct plug-in estimator, that is, it is the variance under $\widehat{F}_n$.

The unbiased estimator could be used in practice; the difference is negligible for large n.

The Bootstrap: Further Examples

In the case of i.i.d. univariate sampling, the bootstrap extends easily to more general settings. For example, extension to the i.i.d. vector case is straightforward.

1. A two-sample problem.
2. The Pearson correlation coefficient.
3. Principal components analysis.

The Bootstrap: Two-sample Estimation

Suppose that two random samples are available from different distributions

$$\begin{aligned} F_1 &: y_{11}, \ldots, y_{1n_1} \\ F_2 &: y_{21}, \ldots, y_{2n_2} \end{aligned}$$

and the parameter of interest is the difference in means

$$\mu_1 - \mu_2.$$

In the normal case, under equal variances for the two populations, the standard error of the estimator

$$\overline{y}_1 - \overline{y}_2$$

can be studied analytically; more generally, outside of the normal case and for other estimands, it cannot.

The Bootstrap: Two-sample Estimation

The bootstrap operates in this case by

- resampling $y_{11}^{(b)}, \ldots, y_{1n_1}^{(b)}$ (size n_1) from the first sample,
- resampling $y_{21}^{(b)}, \ldots, y_{2n_2}^{(b)}$ (size n_2) from the second sample,
- forming the difference

$$\overline{y}_1^{(b)} - \overline{y}_2^{(b)}$$

for $b = 1, \ldots, B$. Other estimators can be studied in this way.

The Bootstrap: Pearson correlation

Suppose that pairs of observations $(x_i, y_i), i = 1, \ldots, n$ are available. The Pearson product-moment correlation is

$$r = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i y_i - \overline{xy})}{\sqrt{\left\{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2\right\}\left\{\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2\right\}}}$$

which measures the linear correlation between the two variables in the sample.

r is the moment-based estimator of the population correlation ρ.

The Bootstrap: Pearson correlation

The bootstrap operates in this case by

- resampling *pairs* $((x_1^{(b)}, y_1^{(b)}), \ldots, (x_n^{(b)}, y_n^{(b)}))$ (size n)
- forming the resampled value of r

$$r^{(b)}$$

for $b = 1, \ldots, B$.

Note that again it is the i.i.d. units that are being resampled.

The Bootstrap: Pearson correlation

Example: LSAT vs GPA

The relationship between GPA (x) and LSAT score (y) in $n = 15$ law school students is to be studied.

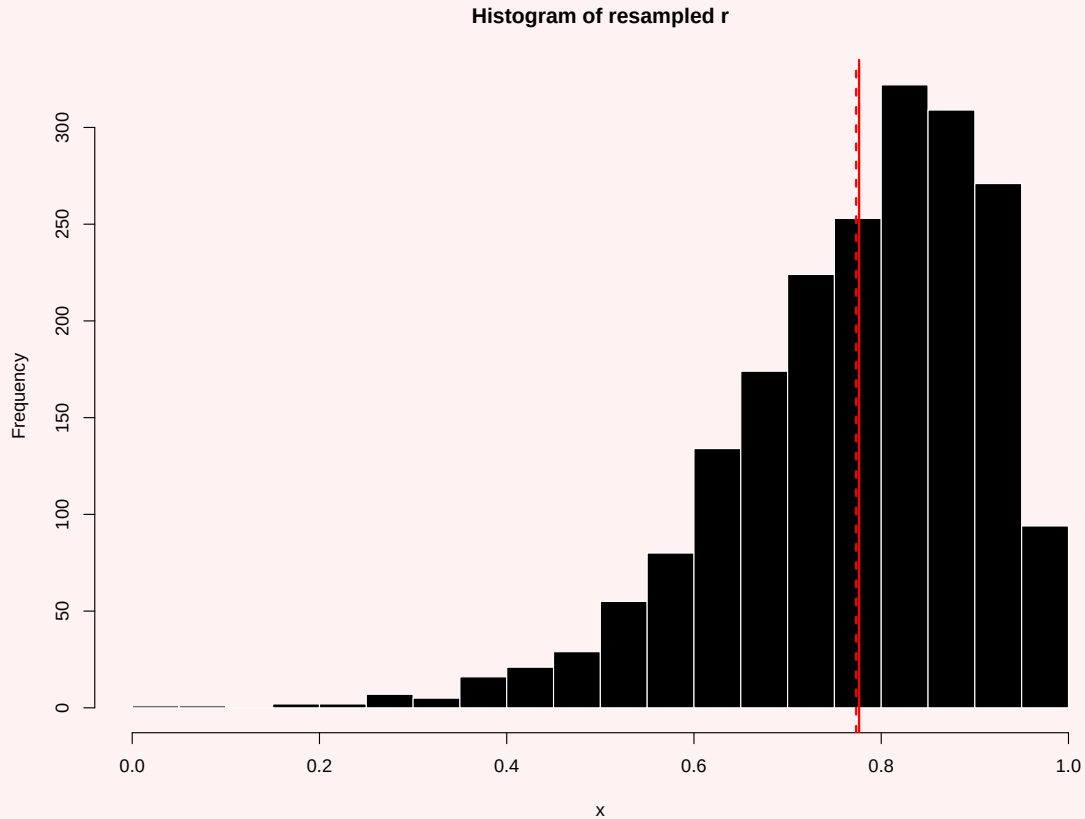
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
GPA	339	330	281	303	344	307	300	343	336	313	312	274	276	288	296
LSAT	576	635	558	578	666	580	555	661	651	605	653	575	545	572	594

For these data $r = 0.776$.

The Bootstrap: Pearson correlation

Example: LSAT vs GPA

2000 (resampled) sample correlation values, with standard deviation 0.1382.



The Bootstrap: Pearson correlation

Example: LSAT vs GPA

A common transformation to approximate normality (for large n) uses

$$Z_n = \frac{1}{2}\log\left(\frac{1+r}{1-r}\right)$$

for which

$$\sqrt{n-3}Z_n \xrightarrow{d} Z \sim \mathcal{N}(\mu, 1)$$

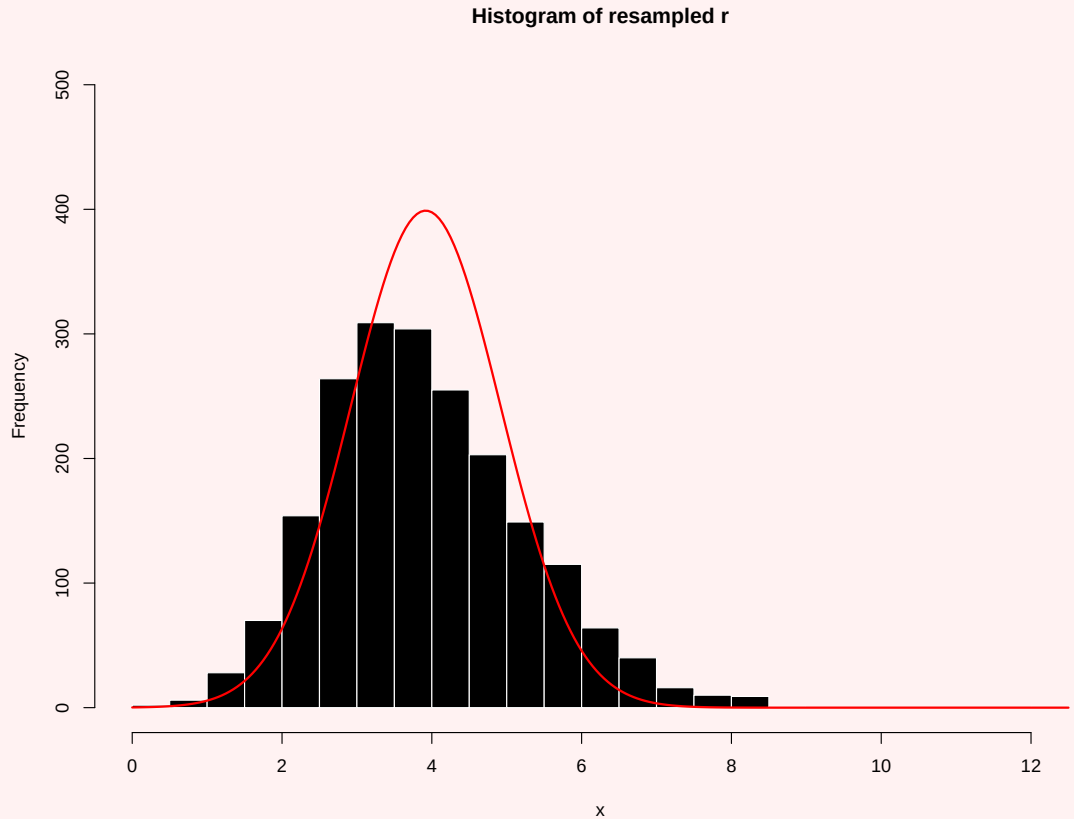
where

$$\mu = \frac{1}{2}\log\left(\frac{1+\rho}{1-\rho}\right)$$

The Bootstrap: Pearson correlation

Example: LSAT vs GPA

2000 Z_n values, and normal approximation: skewness is apparent, and the bootstrap variance is 1.793, rather than 1.



The Bootstrap: Principal components

Suppose that vectors of observations of length d

$$\mathbf{x}_i, i = 1, \ldots, n$$

are available. Let X denote the $n \times p$ matrix formed by row-binding the n i.i.d. vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$, after *column standardization*

- each column of X has its mean subtracted.

Then the sample covariance matrix is

$$\mathbf{S} = \frac{1}{n}\mathbf{X}^\top \mathbb{X}$$

This is a positive-definite matrix, that can be decomposed using the eigen-decomposition

$$\mathbf{S} = \mathbf{V}\mathbf{D}\mathbf{V}^\top$$

where $\mathbf{D}$ is the diagonal matrix of eigenvalues, and $\mathbf{V}$ is the corresponding eigenvector matrix with orthonormal columns.

The Bootstrap: Principal components

The diagonal elements of D are the sample eigenvalues

$$\widehat{\lambda}_1 \geqslant \widehat{\lambda}_2 \geqslant \cdots \geqslant \widehat{\lambda}_p$$

and the scaled quantity

$$\widehat{\theta}_j = \frac{\widehat{\lambda}_j}{\sum_{l=1}^{p} \widehat{\lambda}_l} \qquad j = 1, \ldots, p.$$

is an estimate of the proportion of variation explained by the first eigenvector.

The statistical properties of $\widehat{\theta}_j$ are hard to study analytically.

The Bootstrap: Pearson correlation

The bootstrap operates in this case by

- resampling *vectors* $(\mathbf{x}_1^{(b)}, \ldots, \mathbf{x}_n^{(b)})$ (size n)
- forming the resampled matrix $\mathbf{X}^{(b)}$,
- forming the resampled covariance matrix $\mathbf{S}^{(b)}$,
- computing eigenvalues $\widehat{\lambda}_1^{(b)}, \ldots, \widehat{\lambda}_p^{(b)}$ and scaling

for $b = 1, \ldots, B$.

Note that again it is the i.i.d. units that are being resampled.

Example: Education test scores

88 students each took five tests on mechanics, vectors, algebra, analysis and statistics. This yielded the S matrix

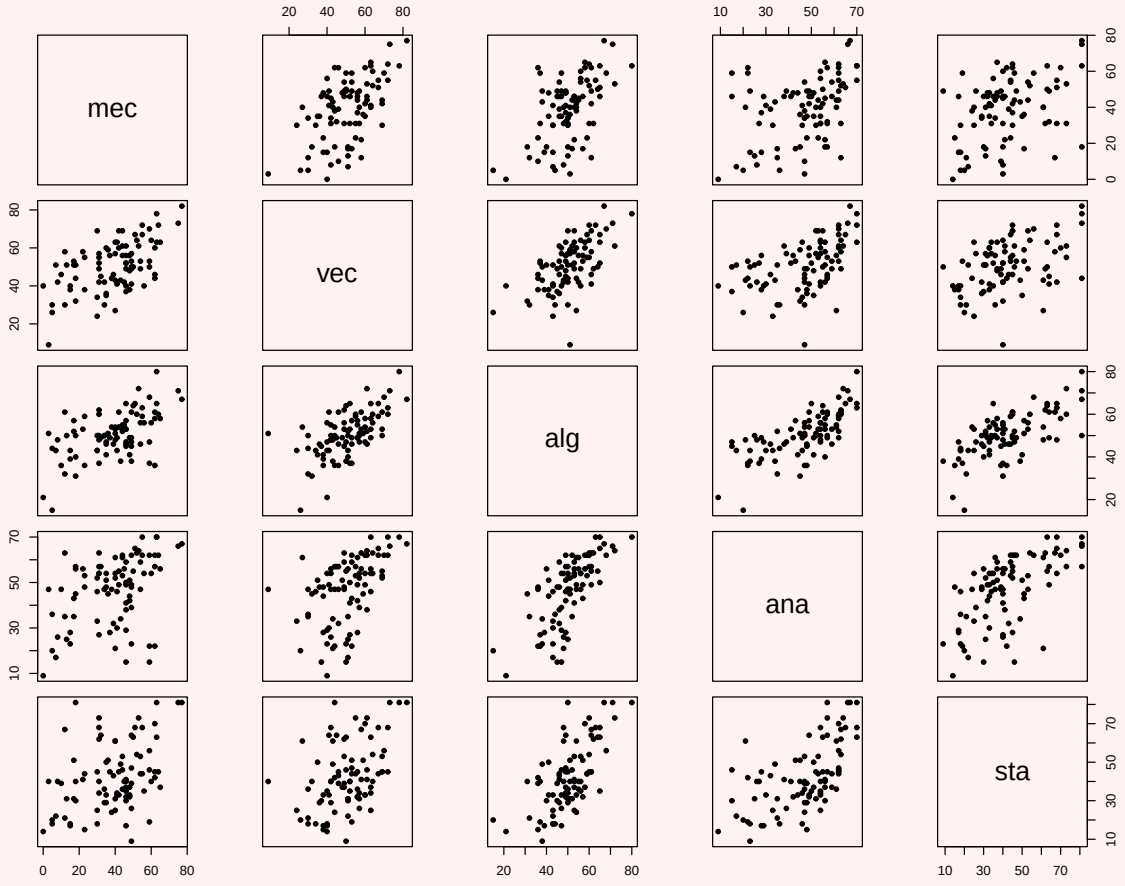
$$S = \begin{bmatrix} 302.29 & 125.78 & 100.43 & 105.07 & 116.07 \\ 125.78 & 170.88 & 84.19 & 93.60 & 97.89 \\ 100.43 & 84.19 & 111.60 & 110.84 & 120.49 \\ 105.07 & 93.60 & 110.84 & 217.88 & 153.77 \\ 116.07 & 97.89 & 120.49 & 153.77 & 294.37 \end{bmatrix}$$

and eigenvalues

$$\hat{\lambda} = (679.18, 199.81, 102.57, 83.67, 31.79)$$

The Bootstrap: Principal components

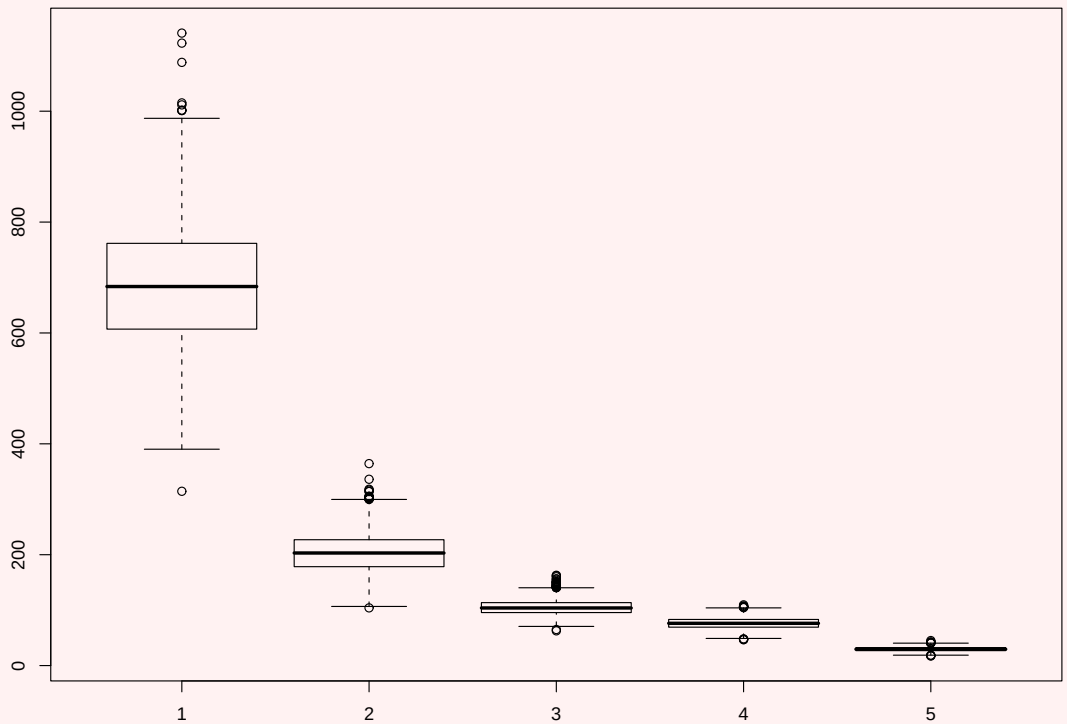
Example: Education test scores



The Bootstrap: Principal components

Example: Education test scores

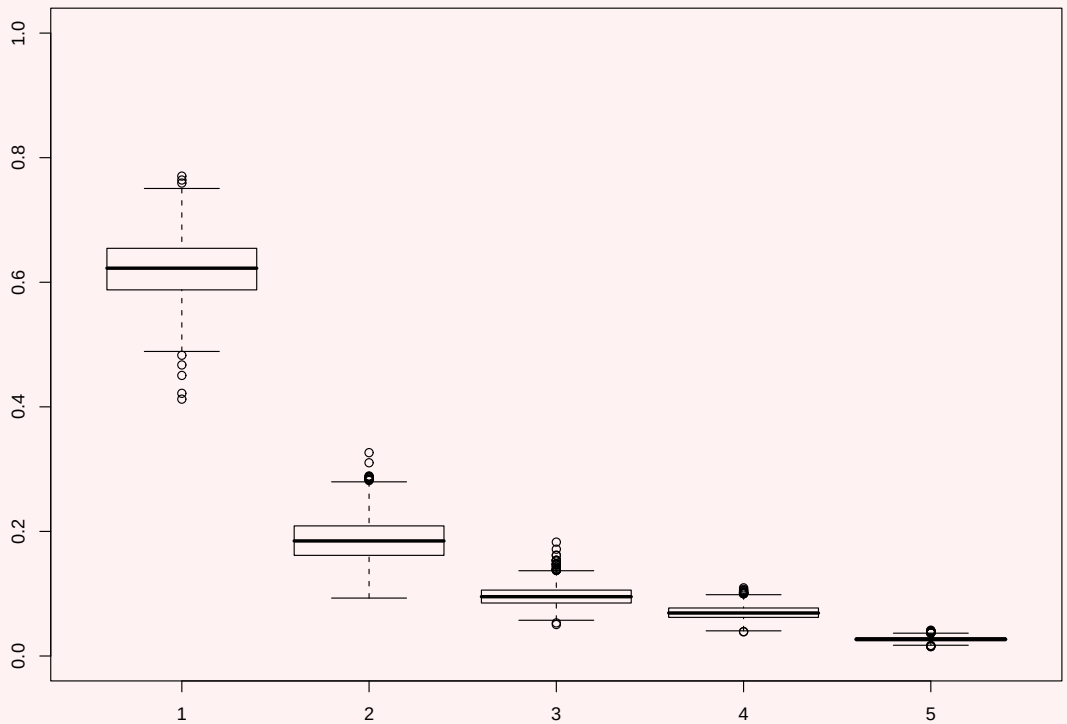
2000 resampled eigenvalues, $\lambda_j, j = 1, \ldots, 5$



The Bootstrap: Principal components

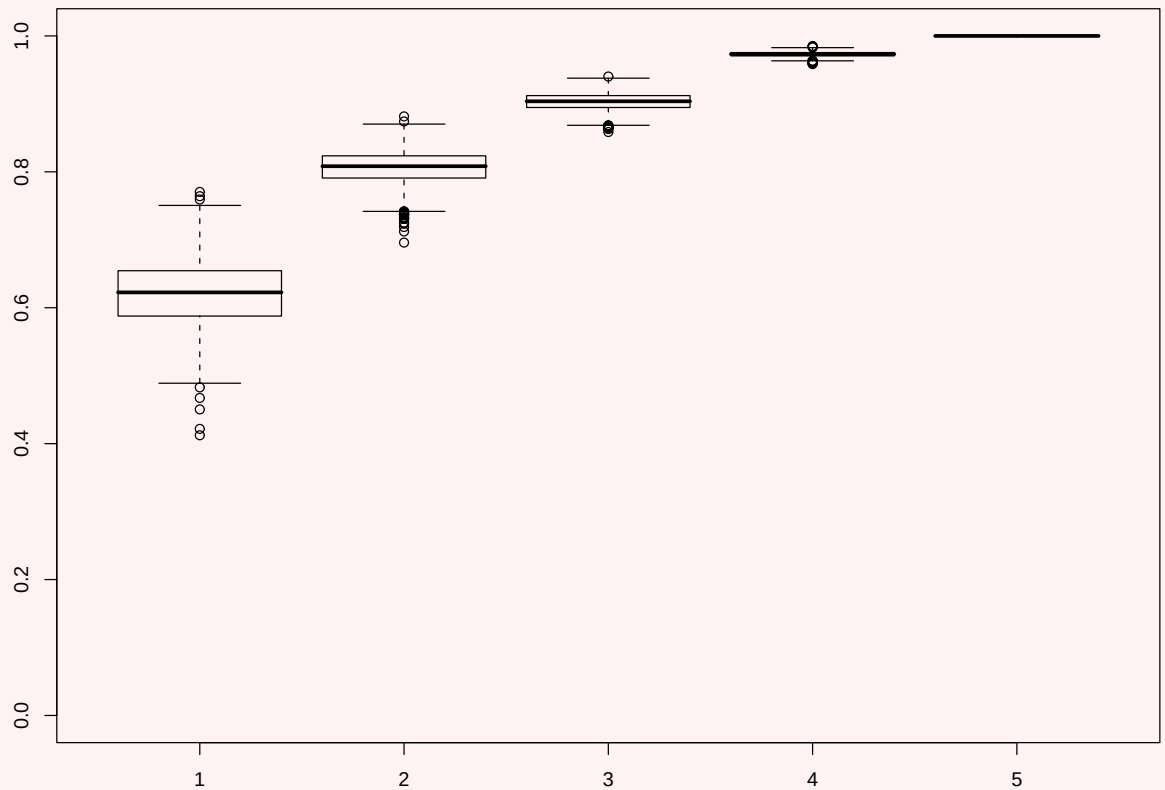
Example: Education test scores

2000 resampled scaled eigenvalues, $\theta_j, j = 1, \ldots, 5$



Example: Education test scores

2000 resampled scaled eigenvalues, cumulative:



The Bootstrap: Principal components

Example: Education test scores

We might conclude from this that really there are only two (orthogonal) dimensions of variation (rather than five) that explain the majority variation in the response.

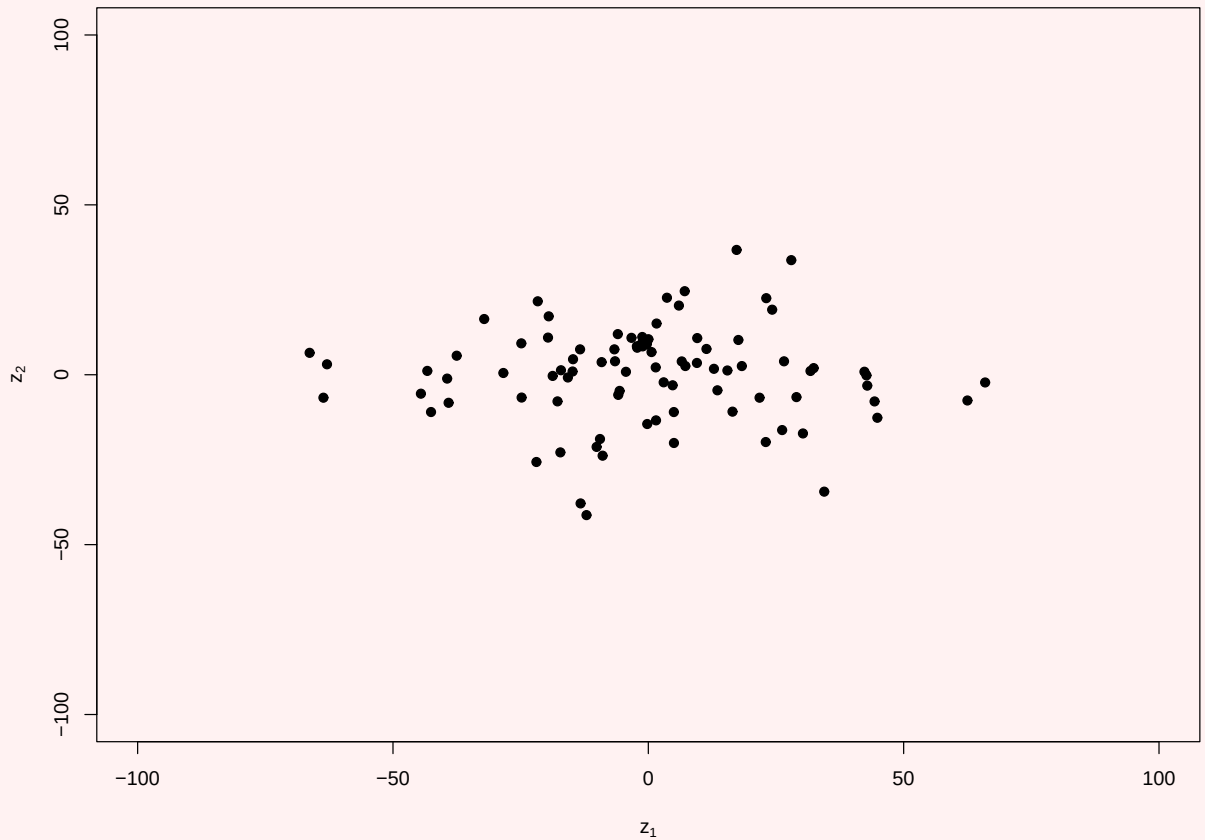
The *principal components* are the vectors

$$z_j = \mathbf{X}v_j \qquad j = 1, \ldots, p$$

where v_j is the jth column of V. The first two principal components carry greater than 80% of the variation observed.

Example: Education test scores

Principal components



The Bootstrap: Regression

The bootstrap can also be applied in a regression setting. Suppose pairs of data $(x_i, y_i), i = 1, \ldots, n$ are available, and the relationship is modelled as

$$y_i = g(x_i; \beta) + \epsilon_i \qquad \mathbb{E}[\epsilon_i | x] = 0$$

for some regression function g. If g is taken to be linear in β,

$$g(x; \beta) = \beta_0 + \sum_{j=1}^{p} \beta_j g_j(x)$$

and the residual errors are presumed Gaussian, then estimation can be carried out using least-squares in the usual way.

Pointwise confidence intervals are easy to obtain, but *simultaneous* confidence intervals are more complicated to compute.

The Bootstrap: Regression

In *harmonic regression*, the g_js are taken to be $\sin$ and $\cos$ functions. Assuming $p = 2q$ say, and $j = 1, \ldots, p$

$$g_j(\mathrm{x}) = \begin{cases} \cos(2\pi \mathrm{x} \phi_l) & j = 2l - 1 \\ \sin(2\pi \mathrm{x} \phi_l) & j = 2l \end{cases}$$

for known constants $\phi_1, \ldots, \phi_q$.

For example, a typical choice is $q = 2$, with

$$\phi_1 = \frac{1}{2} \qquad \phi_2 = \frac{1}{4}$$

In such a model, the ML estimator is

$$\widehat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$$

where $\mathbf{X}$ is the $n \times (p+1)$ matrix with ith row

$$[1, \cos(2\pi x_i \phi_1), \sin(2\pi x_i \phi_1), \cos(2\pi x_i \phi_2), \sin(2\pi x_i \phi_2)]$$

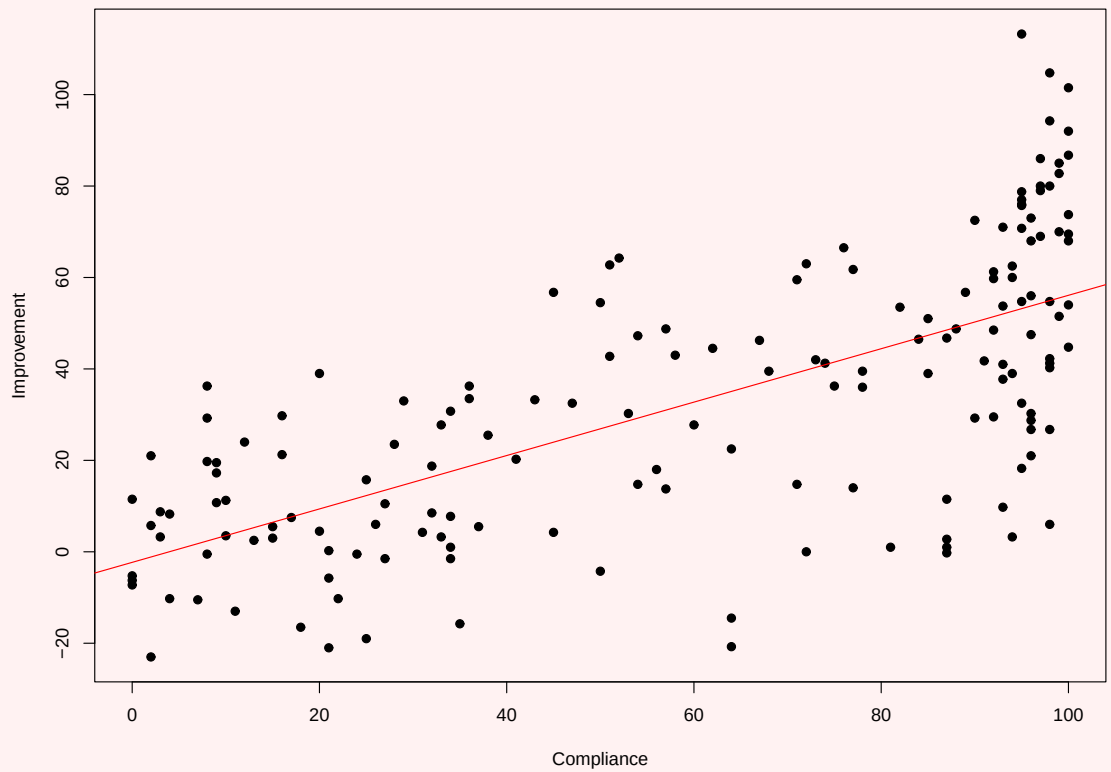
The variance of $\widehat{\beta}$ is

$$\widehat{\sigma}^2 (\mathbf{X}^\top \mathbf{X})^{-1}$$

and pointwise confidence intervals can be obtained in the usual way.

Example: Cholostyramine Data

$n = 164$ men took part in an experiment to see if the drug cholostyramine lowered blood cholesterol levels.



Example: Cholostyramine Data

Consider the model

$$g(x;\beta) = \beta_{00} + \beta_{01}x + \sum_{j=1}^{q} [\beta_{j0}\cos(2\pi x\phi_l) + \beta_{j1}\sin(2\pi x\phi_l)]$$

on the range $[0, 100]$ with

$$\phi_1 = \frac{1}{50} \qquad \phi_2 = \frac{1}{66} \qquad \phi_3 = \frac{1}{75}$$

and the three fits obtained taking

- $q = 0$
- $q = 2$
- $q = 3$

The Bootstrap: Regression

The bootstrap operates in this case by

- resampling *pairs* $((x_1^{(b)}, y_1^{(b)}), \ldots, (x_n^{(b)}, y_n^{(b)}))$ (size n) and forming the boostrapped design matrix $\mathbf{X}_{(b)}$.
- fitting the regression to the resampled data obtaining

$$\widehat{\beta}^{(b)} = (\mathbf{X}_{(b)}^\top \mathbf{X}_{(b)})^{-1} \mathbf{X}_{(b)}^\top \mathbf{y}_{(b)}$$

- recording the fitted values

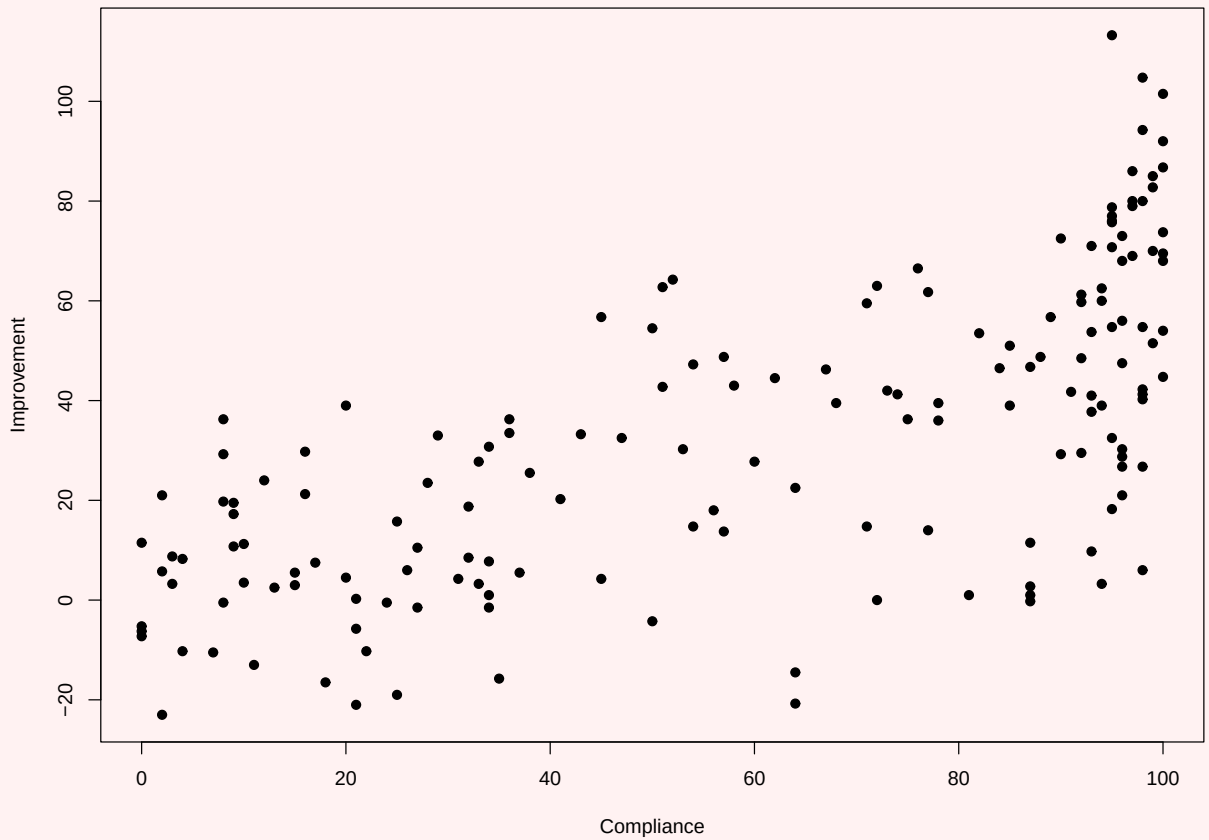
$$\hat{y}^{(b)} = \mathbf{X}_{(b)} \widehat{\beta}^{(b)}$$

for $b = 1, \ldots, B$.

Again it is the i.i.d. units that are being resampled.

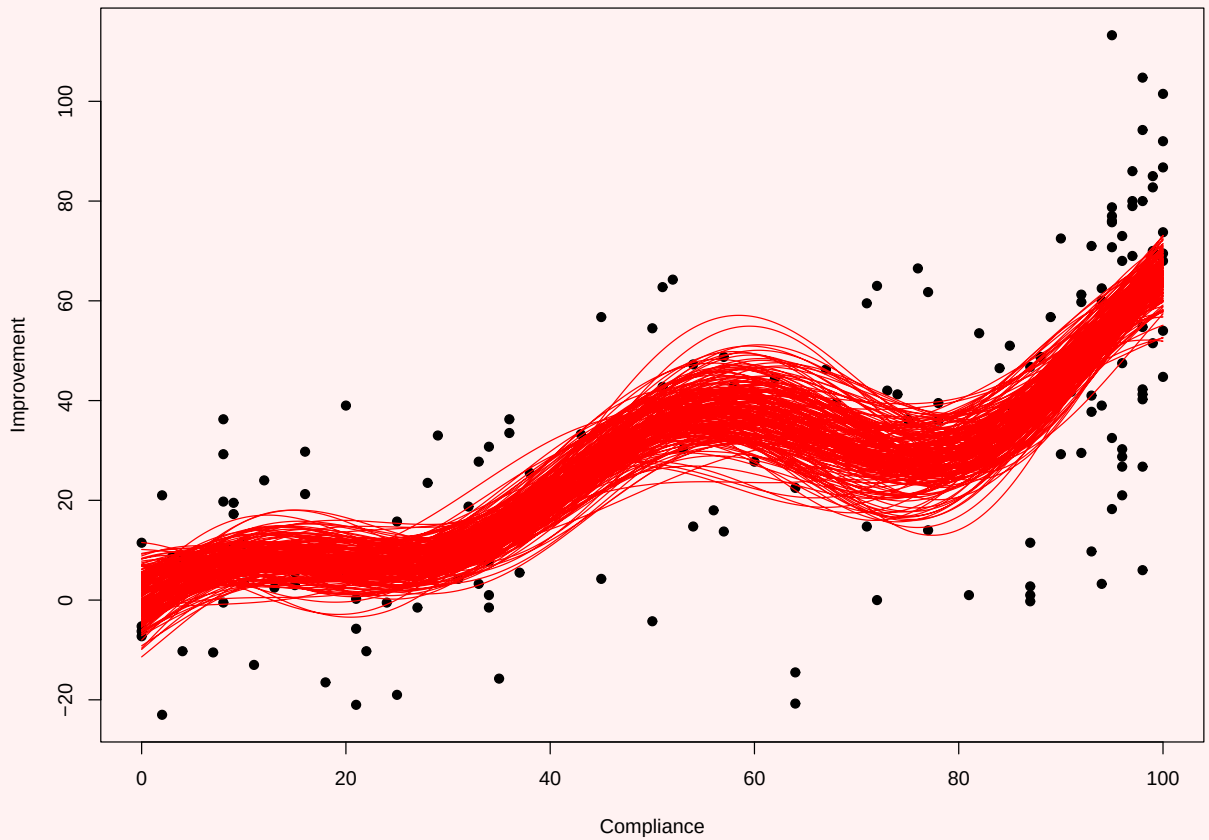
Example: Cholostyramine Data

$B = 200$ bootstrapped curves with $q = 2$.



Example: Cholostyramine Data

$B = 200$ bootstrapped curves with $q = 2$.



The Bootstrap: Loess

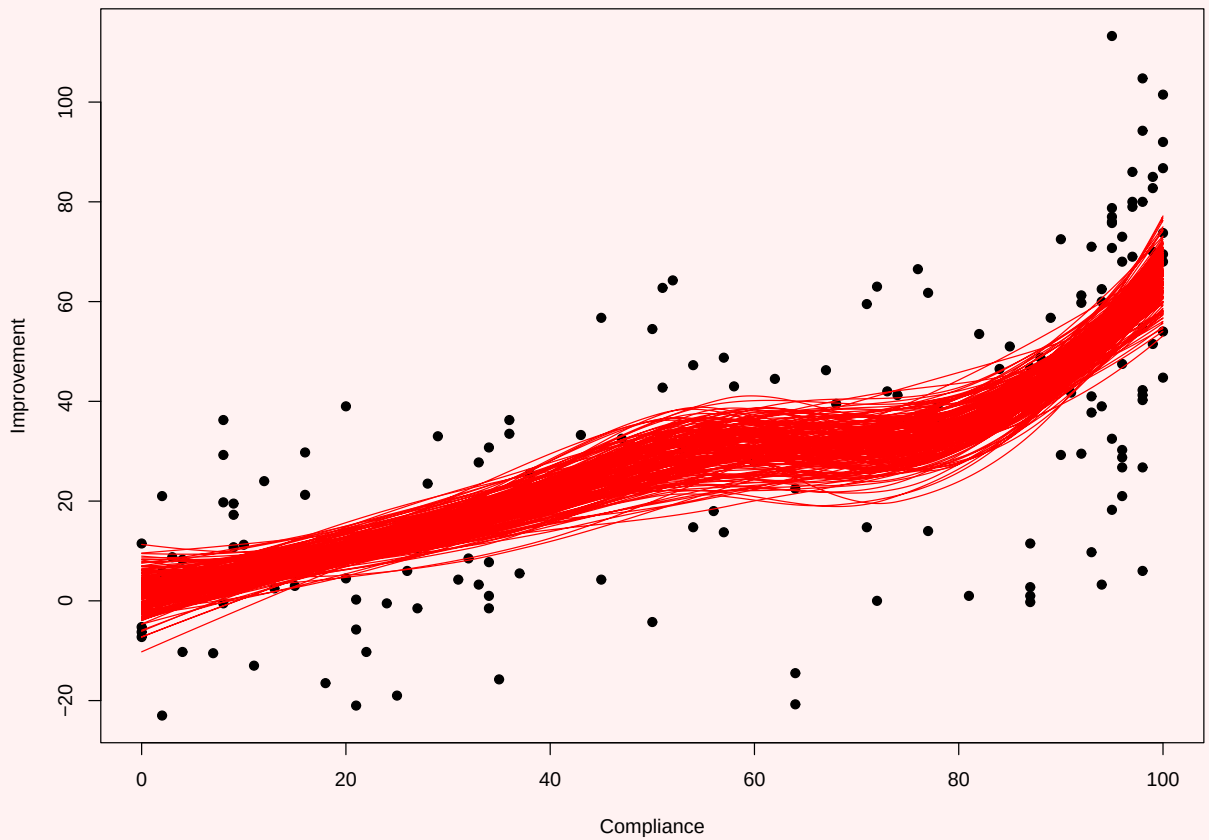
A more flexible model is based on *local smoothing* using *loess*.

The loess approach fits a series of weighted local linear regression models restricted to a rolling neighbourhood of data points. The size of the neighbourhood (the *span*) determines the smoothness of the resulting fitted curve.

The statistical properties of the resulting fitted curve are hard to study analytically, but can be studied by the bootstrap.

Example: Cholostyramine Data

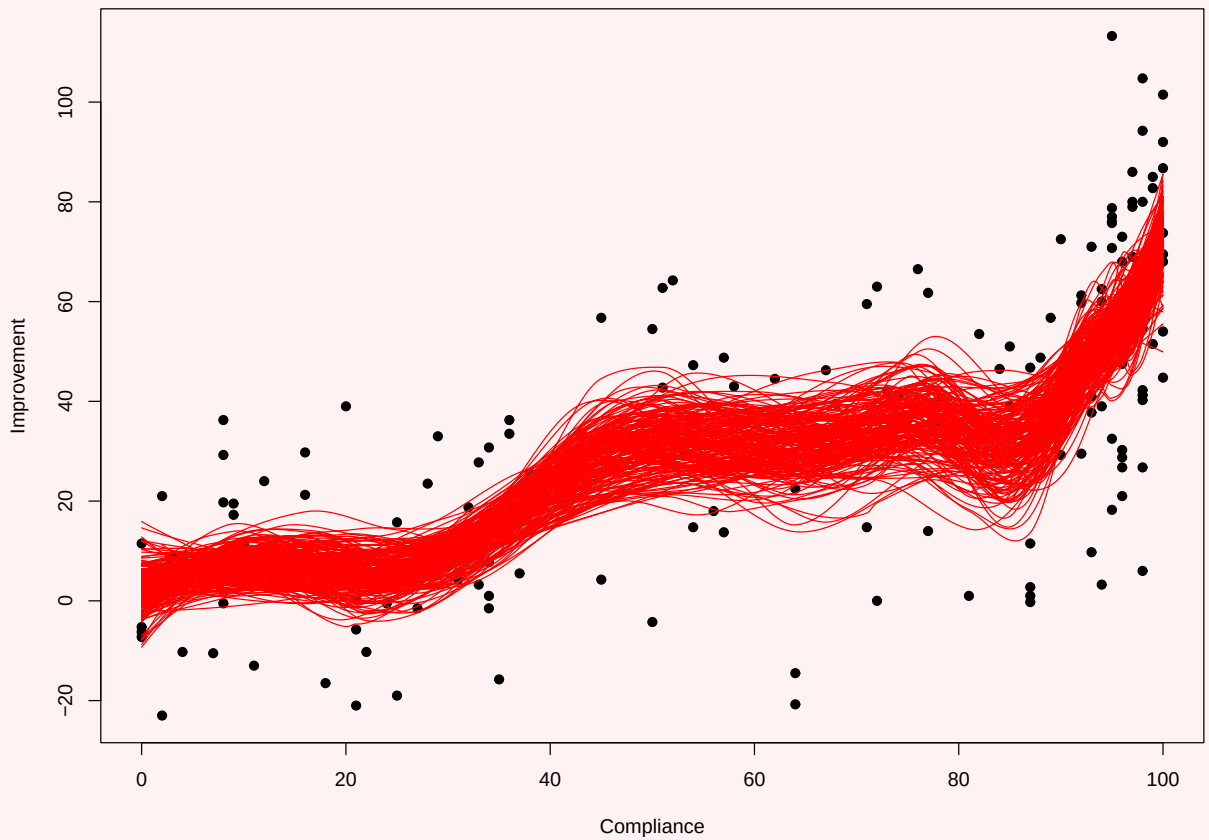
$B = 200$ bootstrapped loess curves: span is 0.75.



The Bootstrap: Loess

Example: Cholostyramine Data

$B = 200$ bootstrapped loess curves: span is 0.375.



The Bootstrap: Loess

The span can be determined by K-fold *cross-validation*:

1. split data at random into K equal portions of size m.
2. fit the loess model using $K-1$ *training* portions using a fixed value of the span.
3. use the fitted model to predict the values in the remaining *test* portion; record the prediction mean square error

$$\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$$

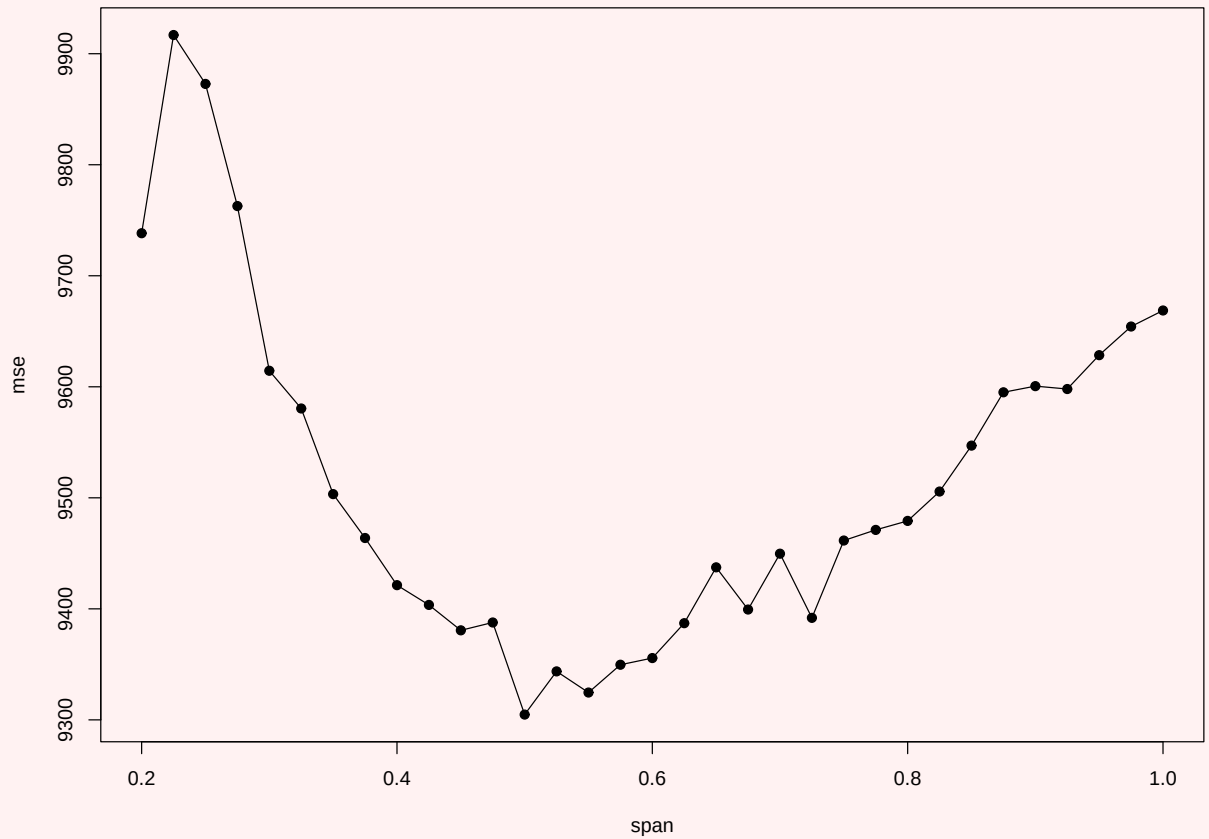
4. repeat 1. to 3. over different random splits, taking the average prediction MSE.

and repeat this whole process for different values of the span, recording the MSE every time. Choose the optimal span to minimize the prediction MSE.

The Bootstrap: Loess

Example: Cholostyramine Data

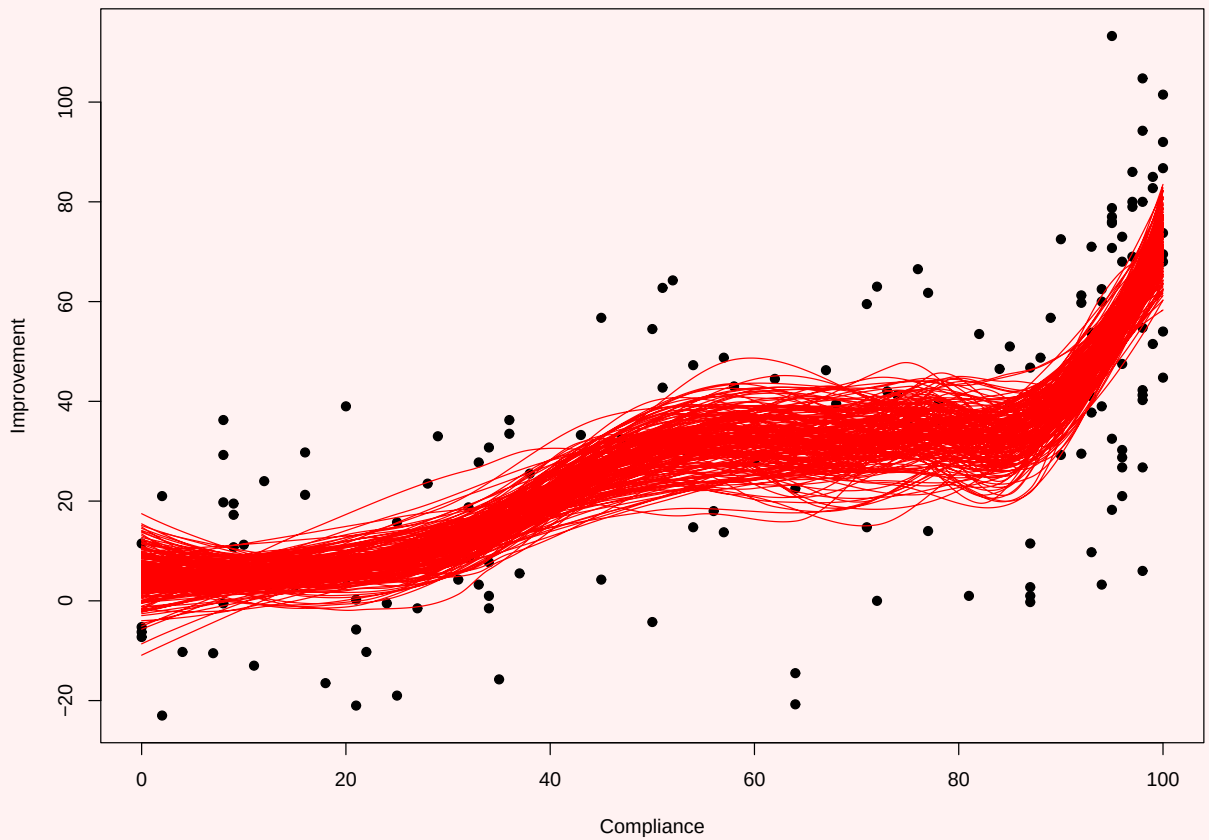
8-fold cross-validation: average prediction MSE from 20000 random splits.



The Bootstrap: Loess

Example: Cholostyramine Data

$B = 200$ bootstrapped loess curves: span is 0.50.



The Bootstrap: Loess instability

Note: in this *semi-parametric* setting, the cross-validation can become unstable. A specific random split may disrupt the local structure of the data so that a local fit at any given x is not possible.

This instability can potentially occur in the parametric setting as well; if the x values contain ties (as for the cholostyramine data) then, by chance we may resample an $\mathbf{X}_{(b)}$ such that

$$\mathbf{X}_{(b)}^{\top}\mathbf{X}_{(b)}$$

is singular.

The Bootstrap: Problems

Suppose $Y_1, \ldots, Y_n \sim \mathcal{N}(0,1)$. Consider the statistic

$$T_n = \max\{Y_1, \ldots, Y_n\}$$

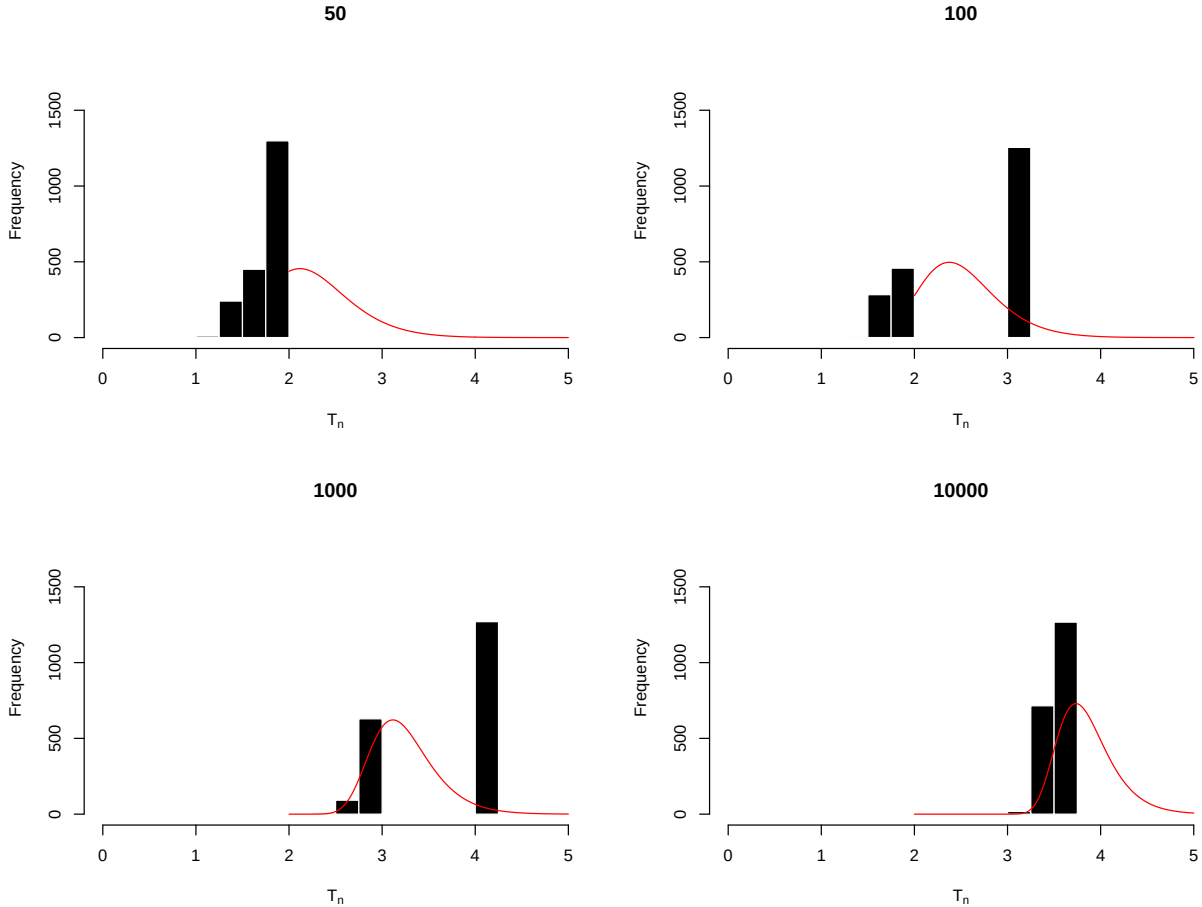
We know that for $t \in \mathbb{R}$

$$F_{T_n}(t) = \{\Phi(t)\}^n \qquad\qquad f_{T_n}(t) = n\phi(t)\{\Phi(t)\}^{n-1}$$

Can we study this statistic via the bootstrap ?

The Bootstrap: Problems

$B = 2000$ bootstrapped maxima for $n = 50, 100, 1000, 10000$.



The Bootstrap: Problems

The maximum order statistic cannot be studied in this setting using the bootstrap.

In this case, T_n has no limiting distribution, that is, as $n \longrightarrow \infty$

$$F_{T_n}(t) \longrightarrow 0$$

for all t. Is this the source of the problem ?

The Bootstrap: Problems

Suppose $Y_1, \ldots, Y_n \sim \mathit{Uniform}(0, \theta)$. Consider again the ML estimator of θ

$$\widehat{\theta}_n = T_n = \max\{Y_1, \ldots, Y_n\}.$$

We know that for $t \in \mathbb{R}$

$$F_{T_n}(t) = \left\{\frac{t}{\theta}\right\}^n \qquad\qquad f_{T_n}(t) = n\frac{1}{\theta}\left\{\frac{t}{\theta}\right\}^{n-1}$$

The same problem arises in this setting.

The Bootstrap: Problems

For the bth bootstrap resampled data set

$$\begin{aligned}\Pr[\widehat{\theta}^{(b)} = \widehat{\theta}] &= \Pr[\max\{Y_1^{(b)}, \ldots, Y_n^{(b)}\} = \max\{Y_1, \ldots, Y_n\}] \\ &= 1 - \left(1 - \frac{1}{n}\right)^n\end{aligned}$$

and as $n \longrightarrow \infty$

$$1 - \left(1 - \frac{1}{n}\right)^n \longrightarrow 1 - e^{-1} \doteq 0.632$$

Therefore, even for large n, there is a probability mass at $\widehat{\theta}$ in the sampling distribution.

The Bootstrap: Problems

Other problems can occur for the bootstrap when resampling can fundamentally change the nature of the data set:

- **Mixture distributions:** if the Y_i follow a mixture distribution with K components, then resampling may only yield observations from $L < K$ of the components. This changes the statistical behaviour of some estimators.
- **Logistic Regression (classification):** For binary data, resampling may eliminate *all* of the data with response value 0 (or 1), in which case the logistic regression coefficients are not defined.

The Parametric Bootstrap

The *parametric* bootstrap replaces *resampling* from $\widehat{F}_n$ with *sampling* from a parametric approximation $F(.;\widehat{\vartheta}_n)$, where $\widehat{\vartheta}_n$ is the parameter estimate derived from the original sample.

Essentially, this is a *Monte Carlo* technique, where $\widehat{s}(\widehat{\theta}^{(*)})$ is approximated using B random samples from $F(.;\widehat{\vartheta}_n)$.

This can be useful if a good (and tractable) parametric approximation exists.

The Parametric Bootstrap

Suppose $Y_1, \ldots, Y_n \sim \mathit{Uniform}(0, \theta)$.

- **Nonparametric bootstrap:** resampling from the values $\{y_1, \ldots, y_n\}$ leads to problems as

$$\Pr[\widehat{\theta}^{(b)} = \widehat{\theta}] > 0$$

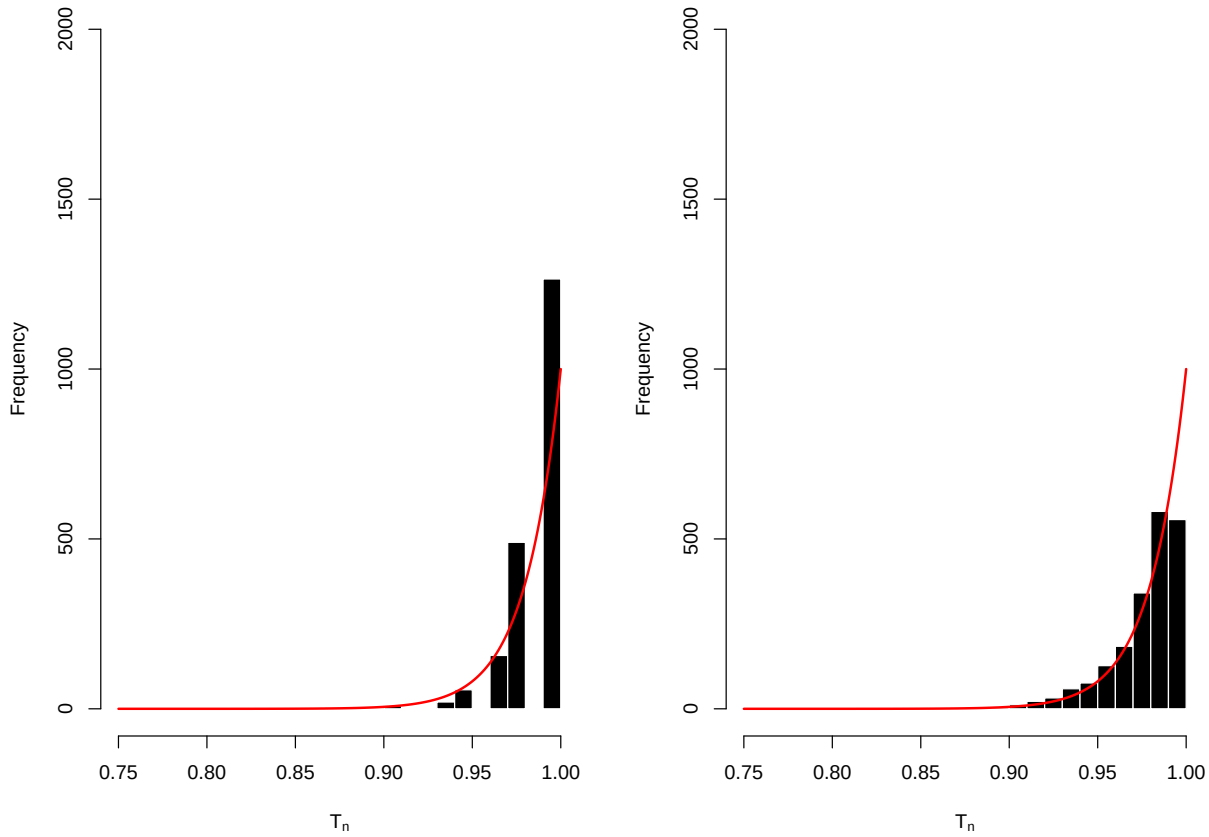
- **Parametric bootstrap:** sampling from the distribution $\mathit{Uniform}(0, \widehat{\theta})$ yields

$$\Pr[\widehat{\theta}^{(b)} = \widehat{\theta}] = 0$$

  and a reasonable approximation to the distribution of $\widehat{\theta}$.

The Parametric Bootstrap

Non-parametric (left) vs Parametric (right)



Bootstrap regression: alternative approach

In an alternative formulation of the bootstrap for linear regression, consider

$$y_i = \mathbf{x}_i \beta + \epsilon_i$$

where $\epsilon_i \sim F(.)$, where F is constrained to have mean zero.

If the distribution F is regarded as unknown, but the form of the linear regression is satisfactory (up to unknown parameters), attention can focus on bootstrapping the residual errors $\epsilon_1, \ldots, \epsilon_n$.

Bootstrap regression: alternative approach

In this β can be estimated by least-squares

$$\widehat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

and then the residual cdf F estimated by the empirical cdf of the residuals

$$e_i = y_i - \mathbf{x}_i \widehat{\beta} \qquad i = 1, \ldots, n$$

denoted $\widehat{F}_n$.

The bootstrap data $y_1^{(b)}, \ldots, y_n^{(b)}$ are then computed by producing a bootstrap resample $e_1^{(b)}, \ldots, e_n^{(b)}$, and setting

$$y_i^{(b)} = \mathbf{x}_i \beta + e_i^{(b)}$$

Bootstrap regression: alternative approach

This then yields the bootstrap estimate

$$\widehat{\beta}^{(b)} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}^{(b)}.$$

In this case, the idealized bootstrap estimate ($B = \infty$) of the standard error is available analytically via

$$\text{Var}[\widehat{\beta}^{(b)}] = \widehat{\sigma}^2 (\mathbf{X}^\top \mathbf{X})^{-1}$$

where

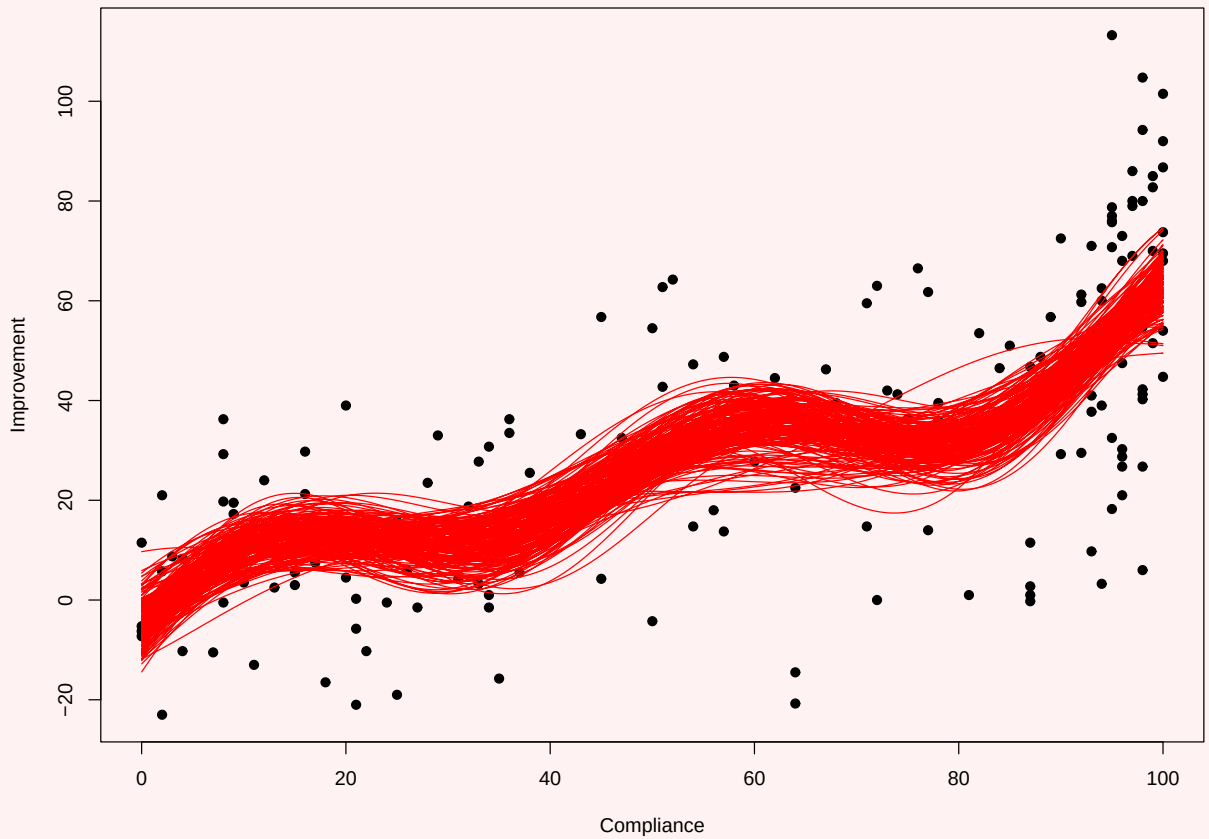
$$\widehat{\sigma}^2 = \frac{1}{n-p} \sum_{i=1}^{n} e_i^2$$

that is, identical to the usual variance in least-squares.

Bootstrap regression: alternative approach

Example: Cholostyramine Data

$B = 200$ harmonic regressions with bootstrapped residuals.



Estimating Bias

The bias of an estimator T_n of $\theta(F)$ is

$$b_{T_n}(F) = \mathbb{E}_F[T_n] - \theta(F)$$

which can be estimated by

$$\widehat{b}_{T_n}(\widehat{F}_n) = \mathbb{E}_{\widehat{F}_n}[T_n^{(*)}] - \theta(\widehat{F}_n)$$

where $T_n^{(*)}$ is the bootstrap estimator, derived from bootstrap resamples

$$t_n^{(1)}, \ldots, t_n^{(B)}$$

and $\theta(\widehat{F}_n)$ is the plug-in estimate. Typically $\mathbb{E}_{\widehat{F}_n}[T_n^{(*)}]$ is approximated by

$$\frac{1}{B}\sum_{b=1}^{B} t_n^{(b)}$$

Estimating Bias

If $T_n = \theta(\widehat{F}_n)$, that is, the estimator is the plug-in estimator, then an alternative estimate of bias can be obtained for $B = \infty$.

Denote by $P_j^{(b)}, j = 1, \ldots, n$ denote the proportion of a bootstrap resampled data set $y_1^{(b)}, \ldots, y_n^{(b)}$ that are equal to y_j.

$$P_j^{(b)} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{y_j\}}(y_i^{(b)})$$

The (random) probability distribution $P^{(b)} = (P_1^{(b)}, \ldots, P_n^{(b)})$ (the *resampling vector*) can be used to construct the bootstrap estimate in terms of the original data; for example,

$$\overline{y}^{(b)} = \sum_{j=1}^{n} P_j^{(b)} y_j.$$

Estimating Bias

The bootstrap resampling vectors are thus

$$P^{(1)}, \ldots, P^{(B)}.$$

In general, denote the bootstrap estimator in terms of the

$$T_n^{(b)} = T_n\big(P^{(b)}\big)$$

Let $P_0 = \frac{1}{n}\mathbf{1}_n$ denote the vector of length n containing identical elements $1/n$. Then, in this notation

$$T_n(P_0) = \theta(\widehat{F}_n)$$

Estimating Bias

Let

$$\overline{P}^{(*)} = \frac{1}{B}\sum_{b=1}^{B} P^{(b)}$$

This allows a new bias estimate

$$\widetilde{b}_{T_n}(\widehat{F}_n) = \frac{1}{B}\sum_{b=1}^{B} t_n^{(b)} - T_n(\overline{P}^{(*)})$$

to be compared with the plug-in estimate

$$\widehat{b}_{T_n}(\widehat{F}_n) = \frac{1}{B}\sum_{b=1}^{B} t_n^{(b)} - T_n(P_0)$$

Estimating Bias

Example: Hormone patch data

Eight subjects were treated with two different hormone patches, and the subsequent level of hormone in the blood-stream measured.

ID	Placebo	Old Patch	New Patch	old-plac.	new - old
				z	y
1	9243	17649	16449	8406	-1200
2	9671	12013	14614	2342	2601
3	11792	19979	17274	8187	-2705
4	13357	21816	23798	8459	1982
5	9055	13850	12560	4795	-1290
6	6290	9806	10157	3516	351
7	12412	17208	16570	4796	-638
8	18806	29044	26325	10238	-2719

Estimating Bias

Example: Hormone patch data

The parameter of interest here is

$$\theta = \frac{\mathbb{E}[Y]}{\mathbb{E}[Z]}$$

which is a measure of bio-equivalence. The plug-in estimate is

$$t_n = \theta(\widehat{F}_n) = \frac{\overline{y}}{\overline{z}} = -0.0713.$$

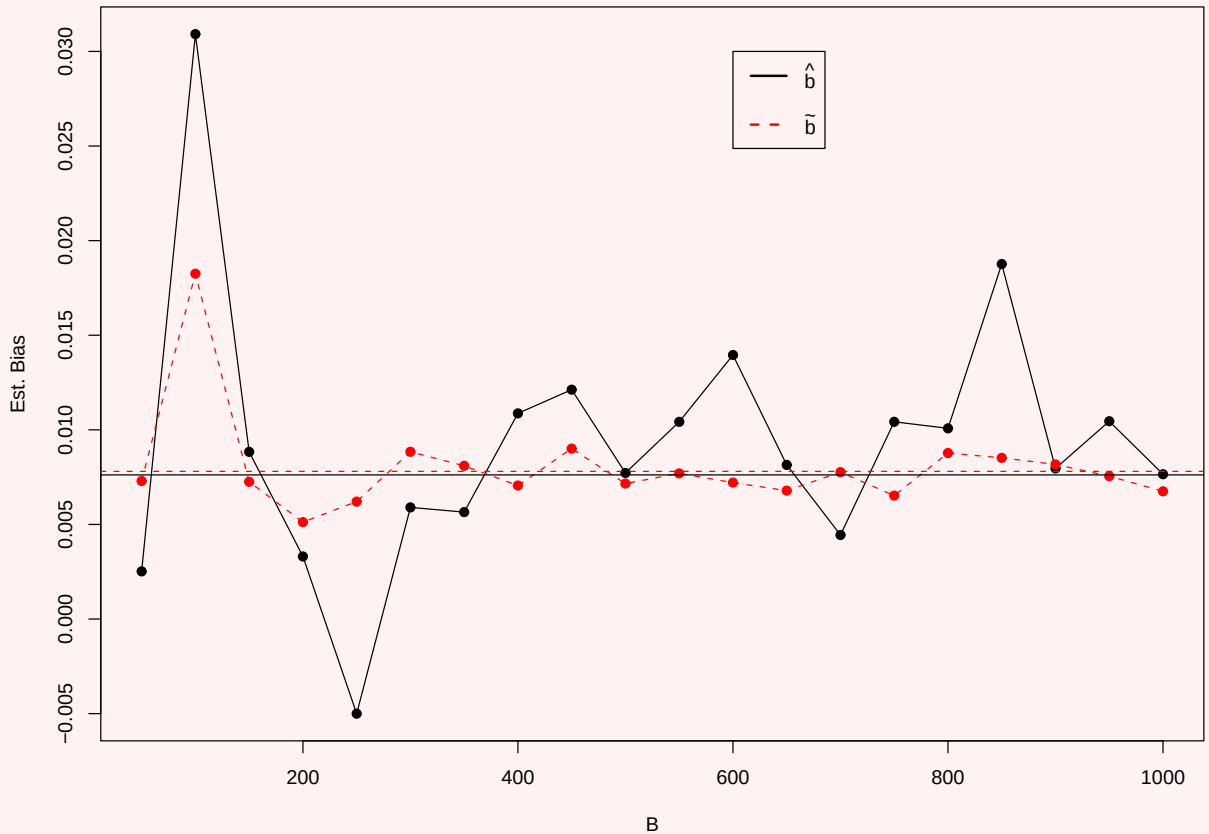
This parameter can be studied using the bootstrap, resampling individual subjects as the i.i.d. units.

For different values of B, we study the bias estimates given by the previous formulae.

Estimating Bias

Example: Hormone patch data

Bias estimates $\widehat{b}$ and $\widetilde{b}$ for B up to 1000.



Estimating Bias

Example: Hormone patch data

The bias estimate $\widetilde{b}$ is much less variable than $\widehat{b}$.

The two bias estimates agree approximately when $B = 100000$

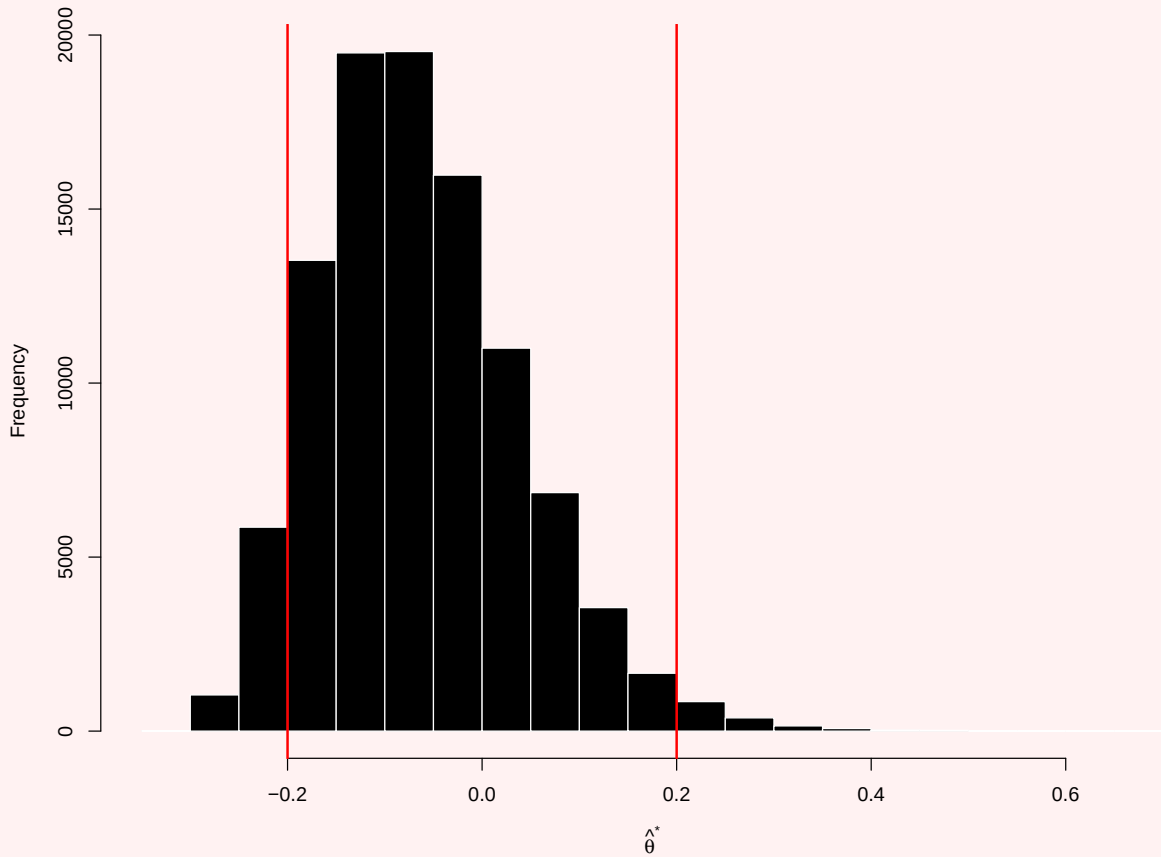
$$\widehat{b}_{T_n}(\widehat{F}_n) = 0.00762 \qquad \widetilde{b}_{T_n}(\widehat{F}_n) = 0.00781$$

but for B up to 1000, $\widetilde{b}$ is very stable, whereas $\widehat{b}$ fluctuates considerably.

Estimating Bias

Example: Hormone patch data

Sampling distribution of $\widehat{\theta}$: $\Pr[|\widehat{\theta}| < 0.2] \doteq 0.9158$.



The jackknife

The *jackknife* was introduced as another bias estimation procedure. For $i = 1, \ldots, n$, let

$$\mathbf{y}_{(i)} = (y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n)$$

be the *i-deleted* data set (with the ith datum deleted). Let

$$\widehat{\theta}_{(i)} = t_{n-1}(\mathbf{y}_{(i)})$$

be the estimate derived from the i-deleted data, and let

$$\widehat{\theta}_{(.)} = \frac{1}{n} \sum_{i=1}^{n} \widehat{\theta}_{(i)}$$

The jackknife

A bias estimate for a plug-in statistic $\widehat{\theta}$ is then

$$\widehat{b}_{\text{JACK}} = (n-1)(\widehat{\theta}_{(.)} - \widehat{\theta})$$

In this calculation

$$\widehat{\theta} = \theta(\widehat{F}_n) \qquad\qquad \widehat{\theta}_{(i)} = \theta(\widehat{F}_{n(i)})$$

are the plug-in estimators derived from the full and i-deleted data respectively.

The jackknife

A jackknife estimate for the standard error of a statistic is

$$\widehat{\text{se}}_{\text{JACK}} = \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}(\widehat{\theta}_{(i)} - \widehat{\theta}_{(.)})^2}$$

This formula applies for *smooth* functions t of the data.

Note: There is no resampling here, so precisely n computations are needed.

Formally, the jackknife is an approximation to the bootstrap that arises from a linear approximation to the statistic of interest, derived from a Taylor expansion.

The jackknife

The above bias/s.e. formulae are derived under the assumption that the identity

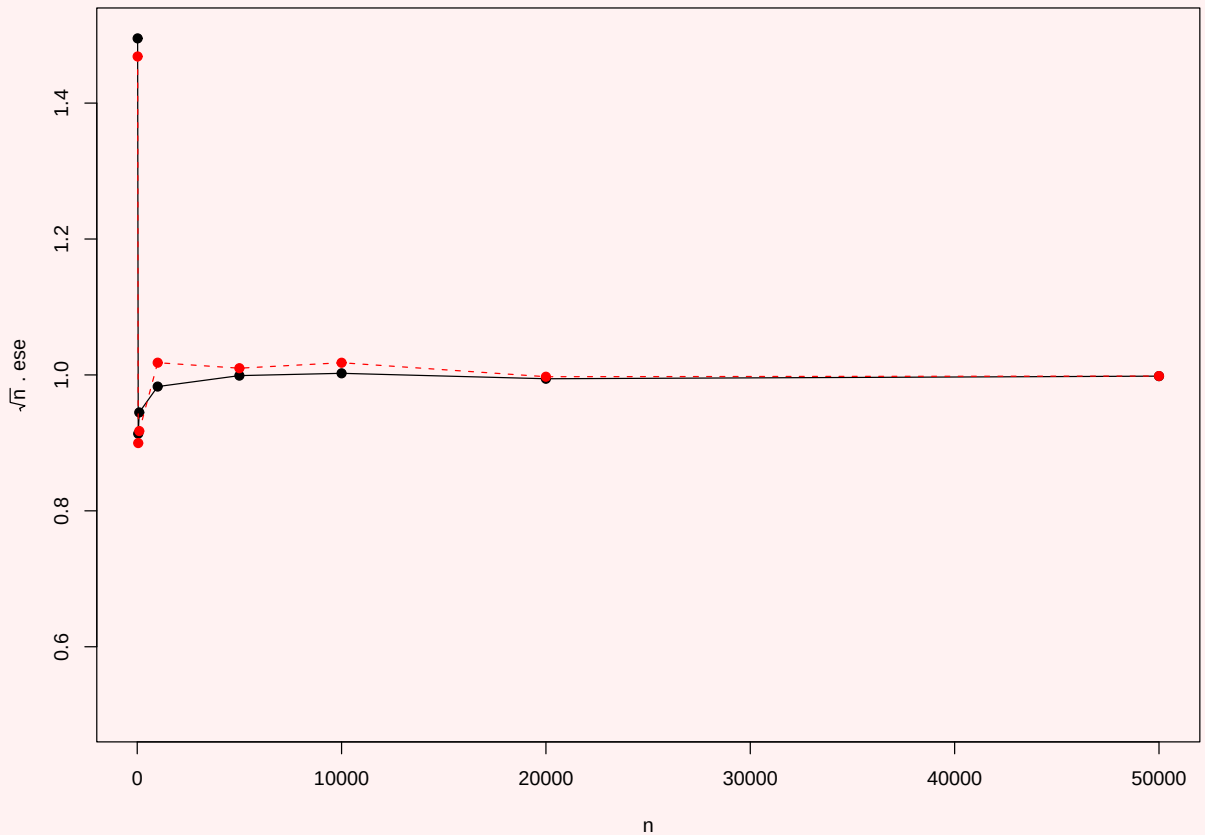
$$\widehat{\theta} = t_n(\mathbf{y}) = \mu + \frac{1}{n}\sum_{i=1}^{n} \alpha(y_i)$$

holds exactly from some function $\alpha(.)$. The jackknife arises from regarding a first-order Taylor approximation for arbitrary t_n.

The jackknife procedure fails when such an expansion fails; for example, for studying the median, the jackknife provides an *inconsistent* estimator.

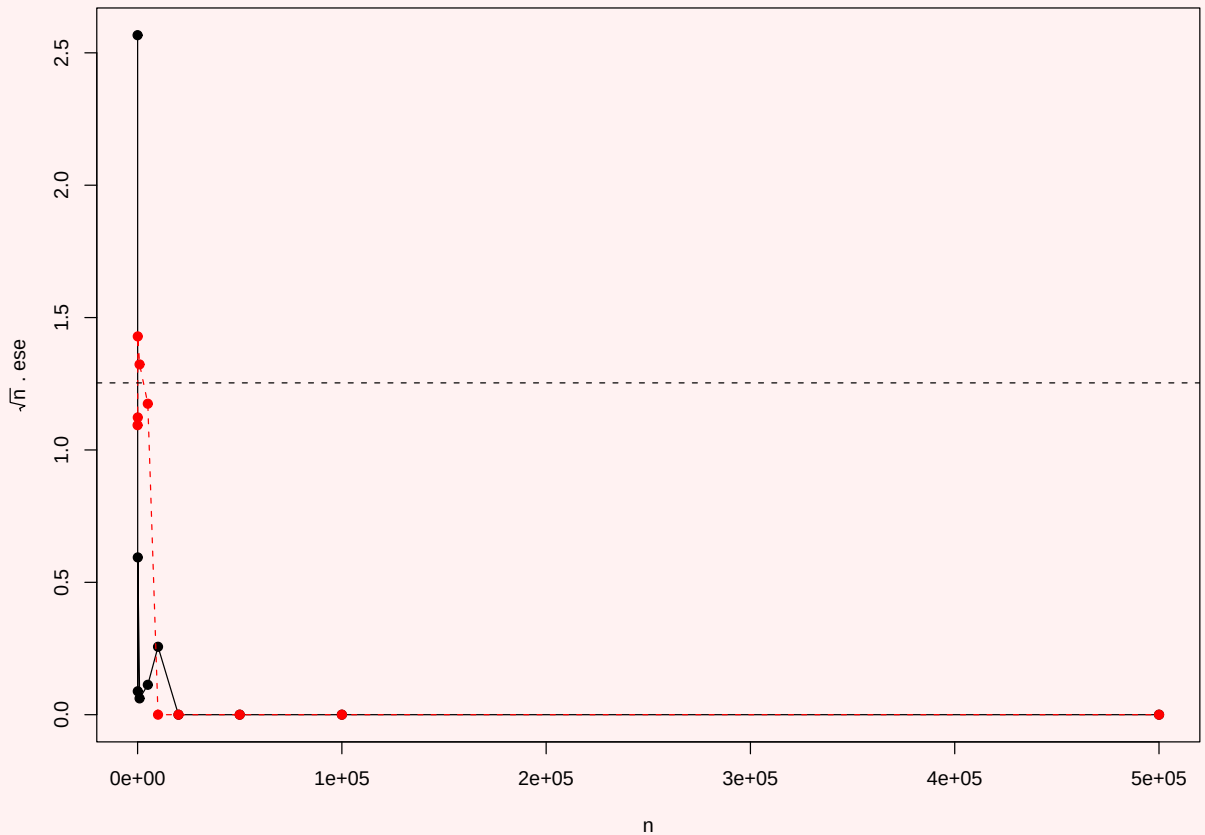
Example: Mean – estimated standard error

$B = 2000$ bootstrap resamples (black) and jackknife (red)



Example: Median – estimated standard error

$B = 2000$ bootstrap resamples (black) and jackknife (red)



Confidence Intervals and Hypothesis Testing

Given an estimator $\widehat{\theta}$ and estimated standard error $\widehat{\text{se}}$, the $(1-\alpha)\%$ *confidence interval*

$$\widehat{\theta} \pm \Phi^{-1}\left(1-\frac{\alpha}{2}\right)\widehat{\text{se}}$$

is commonly used, where Φ^{-1} is the standard normal inverse cdf.

For example,

$$\alpha = 0.10 : \ \Phi^{-1}(0.95) = 1.645$$

$$\alpha = 0.05 : \ \Phi^{-1}(0.975) = 1.960.$$

This formula uses the asymptotic normal approximation to the distribution of $\widehat{\theta}$.

Confidence Intervals and Hypothesis Testing

Recall the definition of a confidence interval in a simple situation where a *pivotal quantity* is available. Suppose the sampling distribution of estimator $\widehat{\theta}$ is given *exactly* by

$$\widehat{\theta} \sim \mathcal{N}(\theta, \text{se}^2).$$

Recall that this is a *frequentist* statement, conditional on the *known* true values θ and se. Then

$$Z = \frac{\widehat{\theta} - \theta}{\text{se}} \sim \mathcal{N}(0, 1)$$

so that, for $c > 0$,

$$\Pr_\theta[-c \leqslant Z \leqslant c] = \Phi(c) - \Phi(-c) = 1 - 2\Phi(-c)$$

Choosing c such that $\Phi(c) = \alpha/2$, say $c = z_{\alpha/2}$, we have

$$\Pr_\theta \left[-z_{\alpha/2} \leqslant \frac{\widehat{\theta} - \theta}{\text{se}} \leqslant z_{\alpha/2} \right] = 1 - \alpha$$

which can be rewritten

$$\Pr_\theta \left[\widehat{\theta} - \text{se} \times z_{\alpha/2} \leqslant \theta \leqslant \widehat{\theta} + \text{se} \times z_{\alpha/2} \right] = 1 - \alpha$$

This is a probability statement concerning the *random interval*

$$[\widehat{\theta} - \text{se} \times z_{\alpha/2}, \widehat{\theta} + \text{se} \times z_{\alpha/2}]$$

given the true value of θ, and the true standard error se.

Confidence Intervals and Hypothesis Testing

Recalling the equivalence with hypothesis testing, the $(1-\alpha)$% confidence interval is the set of values of θ_0 that *do not* lead to a rejection of the null hypothesis

$$H_0 \ : \ \theta = \theta_0$$

at significance level α, in a two-tailed test, in light of the observed data and the estimate $\widehat{\theta}$.

Typically, α is chosen to be 0.05, or 0.01.

The *Normal* interval

$$[\widehat{\theta} - \widehat{\text{se}} \times z_{\alpha/2}, \widehat{\theta} + \widehat{\text{se}} \times z_{\alpha/2}]$$

is justified asymptotically, but for finite n, the *Student-t* interval

$$[\widehat{\theta} - \widehat{\text{se}} \times t_{\alpha/2}(n-1), \widehat{\theta} + \widehat{\text{se}} \times t_{\alpha/2}(n-1)]$$

is preferable, where

$$t_\alpha(\nu)$$

is the α percentile of the Student-t density with ν degrees of freedom.

The standard error se is typically not known, so it is replaced by the usual estimate $\widehat{\text{se}}$ obtained from the plug-in procedure.

For example, for an unknown mean

$$\widehat{\text{se}}(\widehat{\theta}) = \frac{\widehat{\sigma}}{\sqrt{n}} = \frac{1}{\sqrt{n}}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2}$$

that is, using the plug-in estimator of the population variance based on $\widehat{F}_n$.

The corresponding *studentized* bootstrap interval is based on the bootstrap resampled quantities

$$Z^{(b)} = \frac{\widehat{\theta}^{(b)} - \widehat{\theta}}{\widehat{\text{se}}^{(b)}}$$

where

- $\widehat{\theta}$ is the estimator from the original data
- $\widehat{\theta}^{(b)} = t_n^{(b)}$ is the estimator derived from the bth bootstrap resample.
- $\widehat{\text{se}}^{(b)}$ is the estimator of the standard error for $\widehat{\theta}^{(b)}$ from the bth bootstrap resample.

The empirical distribution of the observed bootstrap resampled quantities $z^{(1)}, \ldots, z^{(B)}$ yields the lower and upper percentiles

$$\widehat{l}_{\alpha/2} \quad : \quad \frac{1}{B} \sum_{b=1}^{B} \mathbb{1}_{\{z^{(b)} \leqslant \widehat{l}_{\alpha/2}\}} \doteq \alpha/2$$

$$\widehat{u}_{\alpha/2} \quad : \quad \frac{1}{B} \sum_{b=1}^{B} \mathbb{1}_{\{z^{(b)} \geqslant \widehat{u}_{\alpha/2}\}} \doteq \alpha/2$$

and hence the bootstrap interval

$$[\widehat{\theta} - \widehat{\text{se}} \times \widehat{u}_{\alpha/2}, \widehat{\theta} - \widehat{\text{se}} \times \widehat{l}_{\alpha/2}]$$

Here $\widehat{\theta}$ and $\widehat{\text{se}}$ are estimated from the original data.

Example: Mouse data

Example: Mouse data

The following data are the survival times for mice in a randomized experiment comparing a treated group T with a control group C.

C	52	104	146	10	50	31	40	27	46
T	94	197	16	38	99	141	23		

Example: Mouse data

Bootstrap percentiles: studentized CI for $\mathbb{E}[Y_C]$:

B	0.05	0.1	0.5	0.9	0.95
100	-4.8036	-4.0021	0.0147	1.1167	1.2910
200	-4.8036	-3.4922	-0.1306	1.1167	1.3470
500	-4.8022	-2.2789	0.0000	1.1301	1.5284
1000	-4.5891	-3.0654	-0.0331	1.1705	1.5438
2000	-4.5817	-3.1671	-0.0562	1.1930	1.5290
5000	-4.6010	-3.2014	-0.0411	1.1953	1.5528
10000	-4.6087	-2.9663	-0.0512	1.1739	1.5193
20000	-4.5900	-3.0849	-0.0678	1.1662	1.5072
50000	-4.6083	-3.1837	-0.0735	1.1662	1.5064
100000	-4.5813	-3.0844	-0.0720	1.1677	1.5059
Normal	-1.6449	-1.2816	0.0000	1.2816	1.6449
Student	-1.8595	-1.3968	0.0000	1.3968	1.8595

The observed quantiles $z^{(1)}, \ldots, z^{(B)}$ yields the lower and upper percentiles of the distribution of the pivotal quantities:

$$\widehat{l}_{\alpha/2} \quad : \quad \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}_{\{z^{(b)} \leqslant \widehat{l}_{\alpha/2}\}} \doteq \alpha/2$$

$$\widehat{u}_{\alpha/2} \quad : \quad \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}_{\{z^{(b)} \geqslant \widehat{u}_{\alpha/2}\}} \doteq \alpha/2$$

and hence the bootstrap interval

$$[\widehat{\theta} - \widehat{\text{se}} \times \widehat{u}_{\alpha/2}, \widehat{\theta} - \widehat{\text{se}} \times \widehat{l}_{\alpha/2}]$$

Here $\widehat{\theta}$ and $\widehat{\text{se}}$ are estimated from the original data.

Example: Mouse data

$\alpha = 0.05$, equi-tailed studentized bootstrap interval.

B	Lower	Upper
100	31.6323	123.7727
200	32.6767	128.2670
500	30.6577	128.8923
1000	31.1979	126.5177
2000	31.1686	126.5072
5000	31.1257	127.3363
10000	31.4763	127.8735
20000	31.7005	127.6070
50000	31.8398	127.4720
100000	31.8758	126.9972
Normal	30.0952	82.3492
Student	25.4824	86.9620

R Code

```
> library(boot)
> meancalc<-function(x,i){
+      m <- mean(x[i])
+      n <- length(i)
+      v <- (n-1)*var(x[i])/n^2
+      return(c(m, v))
+ }
> bmean<-boot(y,meancalc,R=100000)
> mouse.ci<-boot.ci(bmean)
> mouse.ci
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 100000 bootstrap replicates

CALL : boot.ci(boot.out = bmean)

Intervals :
Level      Normal              Basic             Studentized
95%   ( 30.09,  82.37 )   ( 27.89,  79.22 )   ( 32.10, 127.22 )

Level     Percentile            BCa
95%   (33.22, 84.56 )   (36.11, 91.67 )
Calculations and Intervals on Original Scale
```

R Code

```
> bmean

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:
boot(data = y, statistic = meancalc, R = 1e+05)


Bootstrap Statistics :
     original        bias    std. error
t1*  56.22222  -0.009346667    13.33840 #For the sample mean
t2* 177.69822 -19.867671001    84.15095 #For the sample variance
```

R Code

The five intervals are:

- `Normal:` The normal interval using a bias-correction constructed using the bootstrap, that is

$$\left[(\widehat{\theta} - \widehat{b}(\widehat{\theta})) - 1.96 \times \widehat{\text{se}}, (\widehat{\theta} - \widehat{b}(\widehat{\theta})) + 1.96 \times \widehat{\text{se}}\right]$$

  where the bias estimate $\widehat{b}(\widehat{\theta})$ is given by

$$\widehat{b}(\widehat{\theta}) = \frac{1}{B}\sum_{b=1}^{B} \widehat{\theta}^{(b)} - \widehat{\theta}$$

```
35 >mean(bmean$t[,1])-thetahat.c
36 [1] -0.009346667
```

R Code

- `Basic:` The Basic interval is constructed by regarding the quantities

$$Z^{(b)} = \widehat{\theta}^{(b)} - \widehat{\theta}$$

as pivotal, that is the interval is $\left[\widehat{\theta} - \widehat{u}_{\alpha/2}, \widehat{\theta} - \widehat{l}_{\alpha/2}\right]$ where

$$\widehat{l}_{\alpha/2} \ : \ \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}_{\{z^{(b)} \leqslant \widehat{l}_{\alpha/2}\}} \backsimeq \alpha/2 \backsimeq \widehat{u}_{\alpha/2} \ : \ \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}_{\{z^{(b)} \geqslant \widehat{u}_{\alpha/2}\}}$$

which implies that the interval may be rewritten

$$\left[\widehat{\theta} - (\widehat{\theta}_u - \widehat{\theta}), \widehat{\theta} - (\widehat{\theta}_l - \widehat{\theta})\right]$$

where $\widehat{\theta}_u$ and $\widehat{\theta}_l$ are the $(1-\alpha/2)$ and $\alpha/2$ bootstrap quantiles.

R Code

- `Studentized:` As described above, the Studentized interval uses the pivots

$$Z^{(b)} = \frac{\widehat{\theta}^{(b)} - \widehat{\theta}}{\widehat{\text{se}}^{(b)}} \qquad b = 1, \ldots, B$$

  and the interval

$$\left[\widehat{\theta} - \widehat{\text{se}} \times \widehat{u}_{\alpha/2}, \widehat{\theta} - \widehat{\text{se}} \times \widehat{l}_{\alpha/2}\right]$$

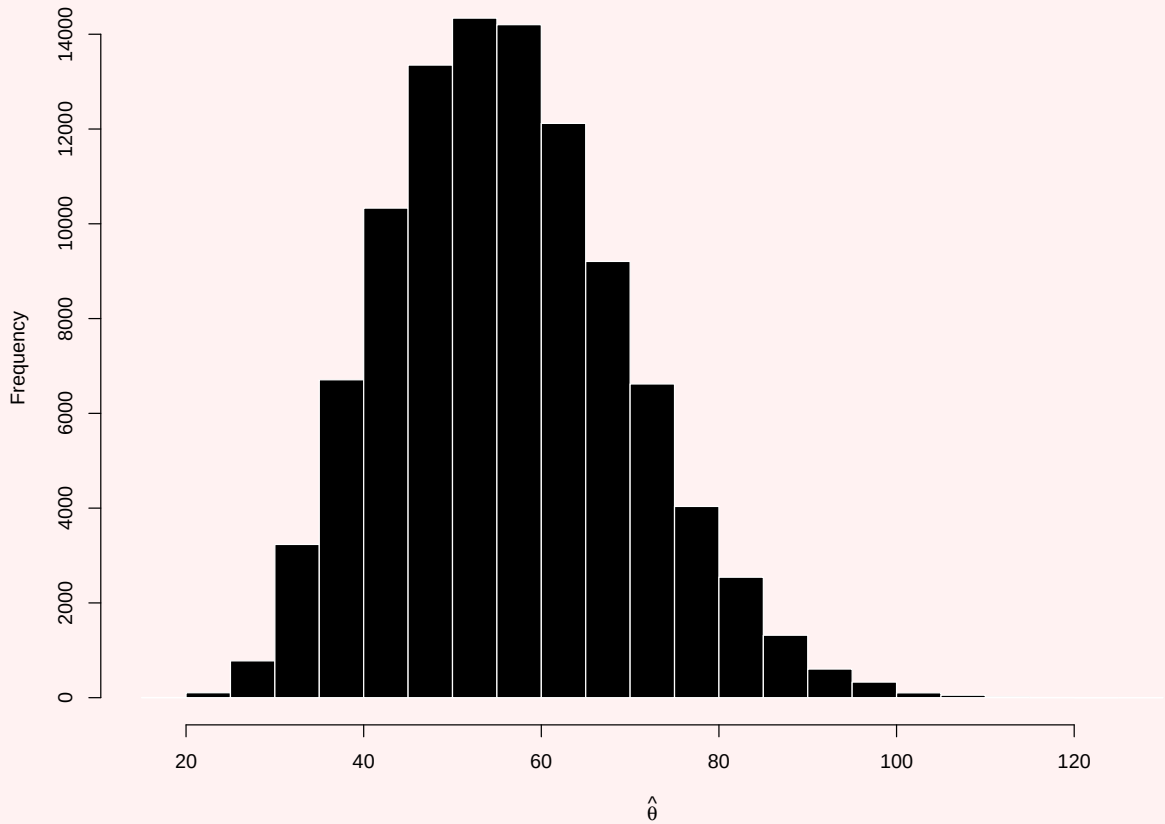
- `Percentile:` The percentile interval is simply

$$[\widehat{\theta}_l, \widehat{\theta}_u]$$

- `BCa:` The BCa interval is the bias-corrected accelerated interval from lectures.

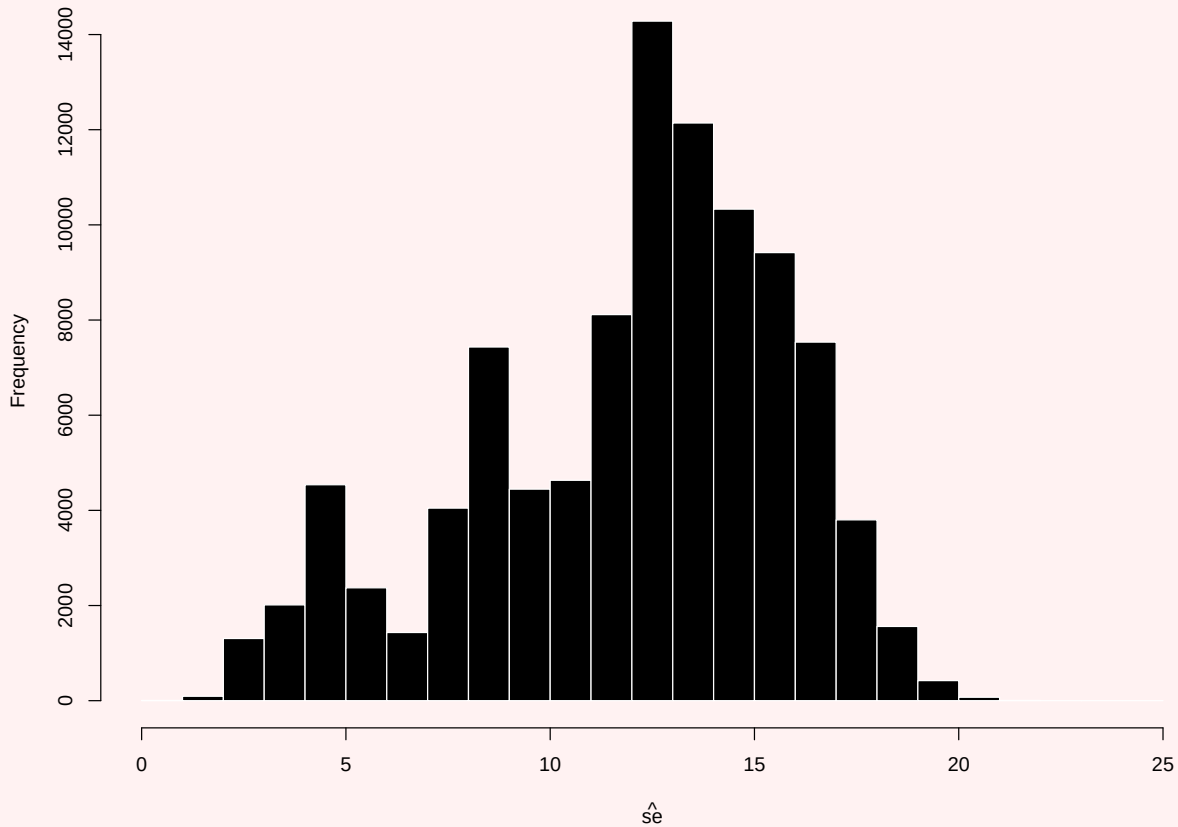
Example: Mouse data

$\widehat{\theta}^{(b)}$:



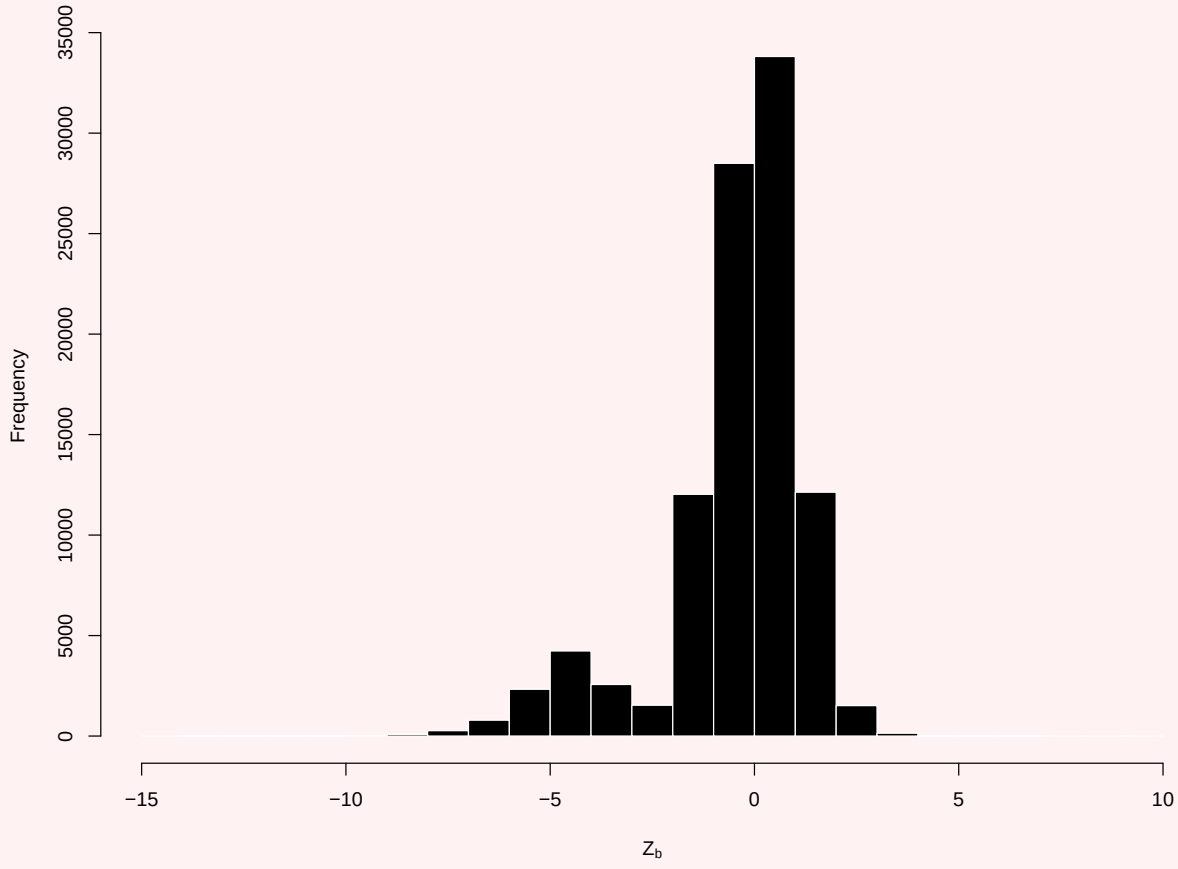
Example: Mouse data

$\widehat{\text{se}}^{(b)}$:



Example: Mouse data

$Z^{(b)} = (\widehat{\theta}^{(b)} - \widehat{\theta})/\widehat{\text{se}}^{(b)}$:



Issues

Some issues:

- In general, a formula for $\widehat{\text{se}}$ is not available
  - Need to do a *nested* bootstrap: within each bootstrap iteration, need an internal bootstrap calculation of the standard error estimate, $\widehat{\text{se}}_B$.
  - Can be computationally demanding
- The symmetric interval of the form

$$\widehat{\theta} \pm \text{se} \times q_{\alpha/2}$$

  may include negative values, even if the quantity of interest is positive.
  - A transformation may be necessary
- The bootstrap interval is not invariant to transformations (the computation of standard errors needs care)

Variance stabilizing transformations

The bootstrap-t interval is constructed by appealing to asymptotic normality, so therefore we seek a transformation $\phi = g(\theta)$ such that the distribution of $\widehat{\phi}$ is better represented by a normal distribution for finite n.

Suppose that, by the Central Limit Theorem, we have that

$$\sqrt{n}(\widehat{\theta} - \theta) \xrightarrow{d} \mathcal{N}(0, \sigma^2(\theta))$$

where the asymptotic variance $\sigma^2(\theta)$ in general depends on the true θ. We seek a transform $\phi = g(\theta)$ such that

$$\sqrt{n}(\widehat{\phi} - \phi) \xrightarrow{d} \mathcal{N}(0, \varsigma^2)$$

where ς^2 does not depend on the true θ (or ϕ).

Variance stabilizing transformations

For example, for the Pearson correlation coefficient R in a bivariate normal problem with true correlation ρ, we saw that if

$$Z = \frac{1}{2}\log\left(\frac{1+R}{1-R}\right) \qquad\qquad \varrho = \frac{1}{2}\log\left(\frac{1+\rho}{1-\rho}\right)$$

then

$$\sqrt{n-3}Z \xrightarrow{d} \mathcal{N}(\varrho, 1)$$

suggesting that we should construct the bootstrap-t interval for Z, and then back-transform to the R scale to get the confidence interval for ρ.

Variance stabilizing transformations

Suppose that X is a random variable with

$$\mathbb{E}[X] = \theta \qquad \text{Var}[X] = s(\theta)^2$$

Let transformation g be defined by the relation

$$\dot{g}(x) = \frac{1}{s(x)} \qquad \text{or, equivalently} \qquad g(x) = \int^x \frac{1}{s(u)}\, du$$

Then if $Y = g(X)$, by a Taylor series approximation

$$\text{Var}[Y] = \{\dot{g}(\theta)\}^2 s(\theta)^2 = 1$$

This *variance-stabilizing transformation* can be used to assist in the construction of the bootstrap-t intervals.

Variance stabilizing transformations

In statistical applications, the function $s(.)$ is the standard error, a function of the true θ; often this is not known. However, it can be estimated from the data.

Let

$$s(u) = \text{se}(\widehat{\theta}|\theta = u).$$

This function can be estimated using a nested bootstrap for a collection of fixed values, then integrated numerically to yield an estimate of the transform function g.

Algorithm

1. Generate B_1 bootstrap samples, and bootstrap estimates $\widehat{\theta}^{(b)}$. For *each* bootstrap sample $\mathbf{y}^{(b)}$, generate B_2 nested resamples, and estimate the standard error $\widehat{\text{se}}(\widehat{\theta}^{(b)})$ for this bootstrap sample.
2. Fit a curve through the collection of points $\{(\widehat{\theta}^{(b)}, \widehat{\text{se}}(\widehat{\theta}^{(b)}),$ to get an estimate of $s(u) = \text{se}(\widehat{\theta}|\theta = u)$.
3. Estimate the variance-stabilizing transform g by integrating $s(u)$ numerically

Algorithm

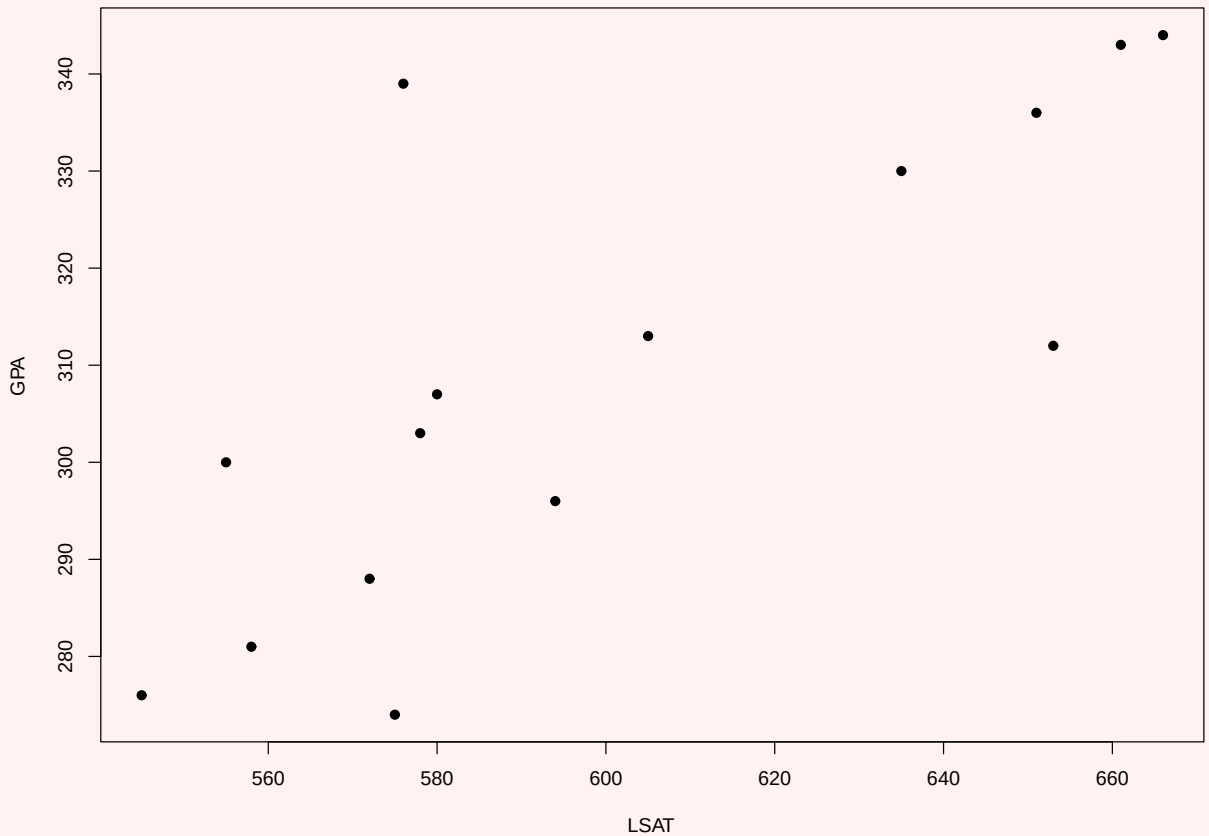
4. Use B_3 new bootstrap samples to compute a bootstrap-t interval for $\phi = g(\theta)$ by inspecting

$$\widehat{\phi}^{(b)} - \widehat{\phi}$$

   as the variance has been stabilized to 1.
5. Back-transform to the θ scale using the numerically defined g^{-1}.

Example: LSAT vs GPA data

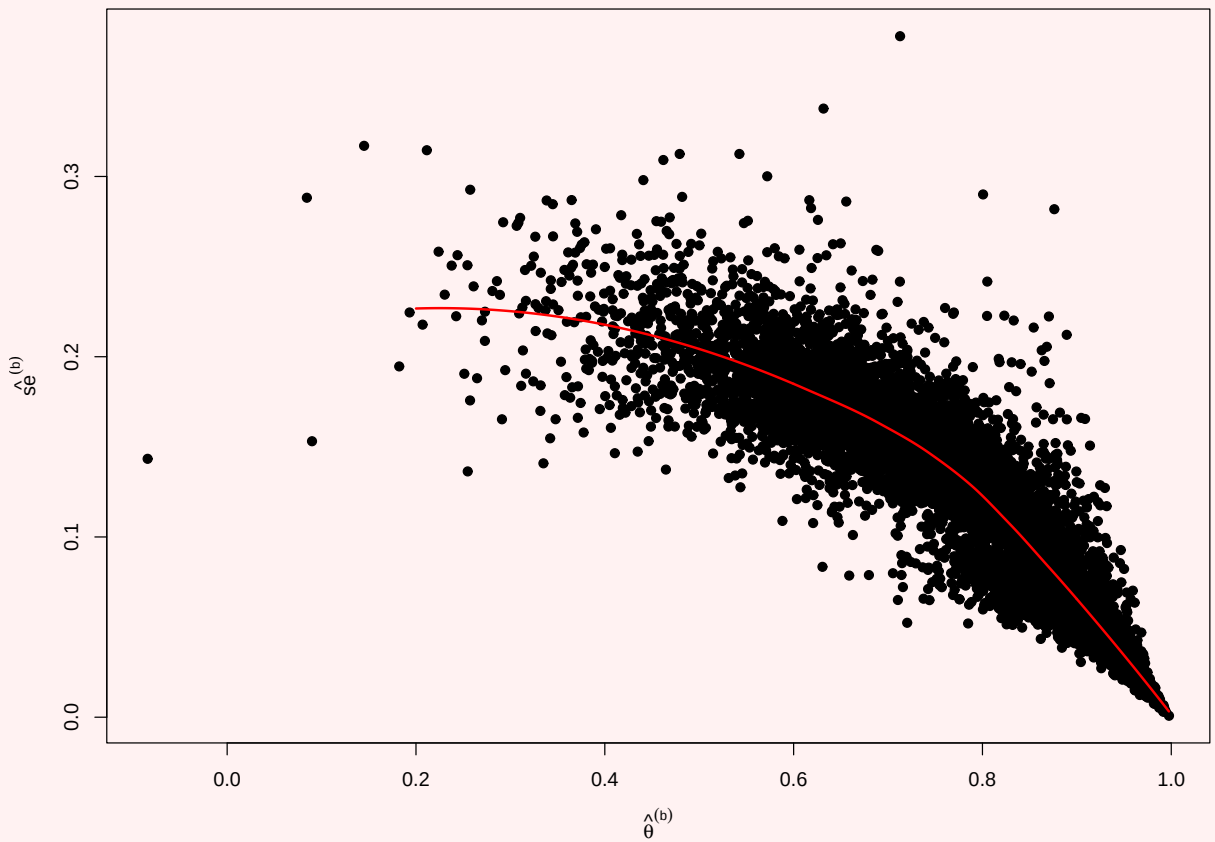
Raw data



Variance stabilizing transformations: Example

Example: LSAT vs GPA data

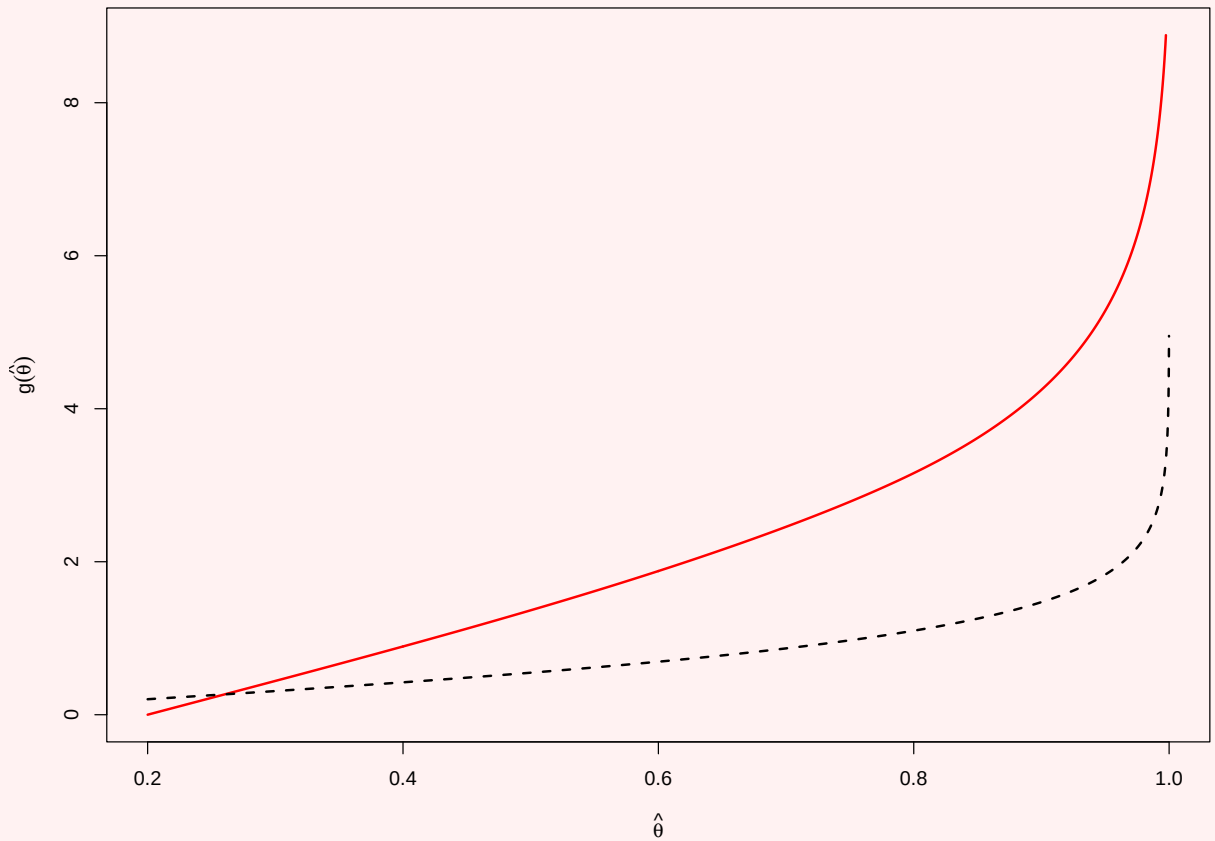
$B_1 = 10000$ bootstrap resamples, $B_2 = 500$ inner resamples.



Variance stabilizing transformations: Example

Example: LSAT vs GPA data

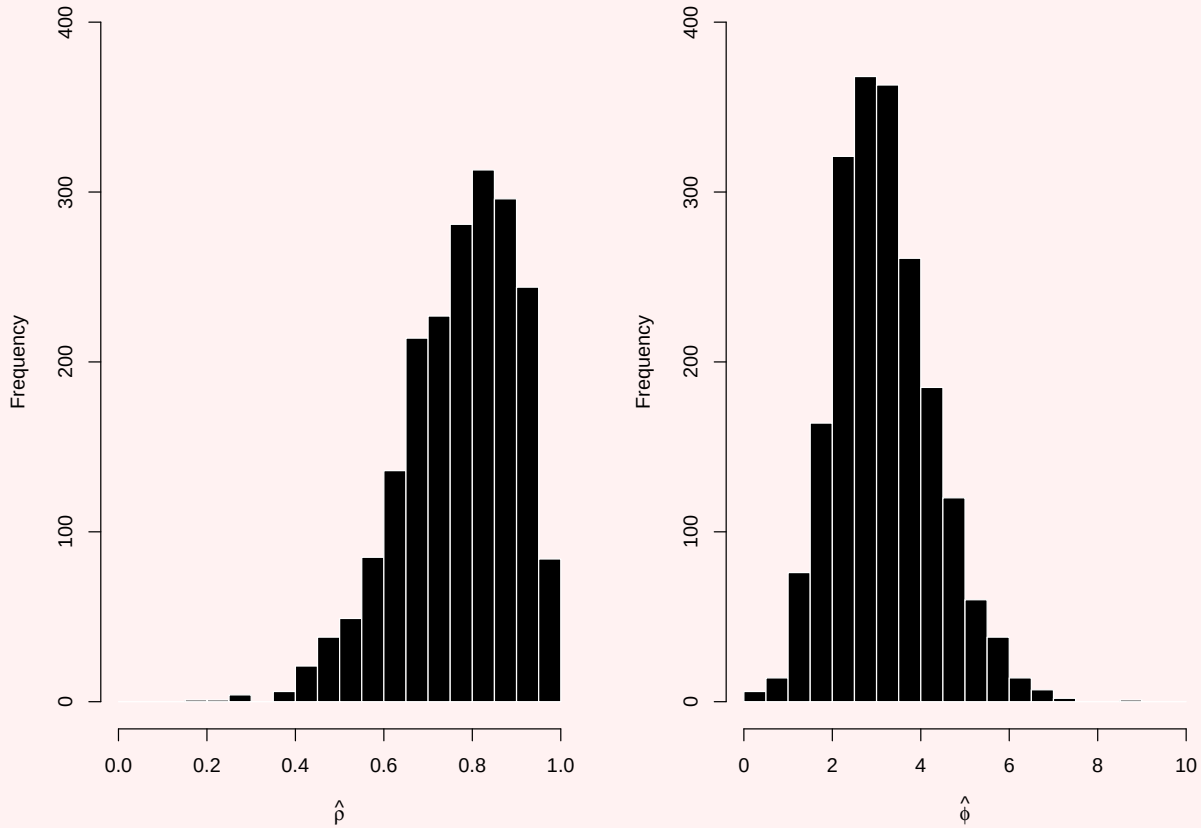
Estimated transform g (solid), and Fisher transform (dashed)



Variance stabilizing transformations: Example

Example: LSAT vs GPA data

$B = 2000$ bootstrap samples of $\widehat{\rho}$ (left) and $\widehat{\phi}$ (right).



Variance stabilizing transformations: Example

Example: LSAT vs GPA data

Comparison of 90 % intervals:

Method	Lower	Upper
Fisher transform	0.5090	0.9071
$\widehat{g}$	0.5299	0.9448
Bootstrap	0.5262	0.9475

The bootstrap interval is computed direct from the percentiles of the initial B_1 bootstrap resampled values.

Bootstrap Percentile Interval

The *bootstrap percentile interval* is computed direct from the percentiles of the bootstrap resampled values.

$$\widehat{l}_{\alpha/2} \quad : \quad \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}_{\{\widehat{\theta}^{(b)} \leqslant \widehat{l}_{\alpha/2}\}} \doteq \alpha/2$$

$$\widehat{u}_{\alpha/2} \quad : \quad \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}_{\{\widehat{\theta}^{(b)} \geqslant \widehat{u}_{\alpha/2}\}} \doteq \alpha/2$$

This method is invariant to transformation; it is a non parametric version of the previous method which focusses on the quantiles of interest.

Typically, B needs to be large to get reliable percentile intervals. They can be more variable when the sampling distribution of the statistic is highly skewed.

Bias correction/Acceleration

An alternative approach uses the interval based on

$$\widehat{l}_{\alpha_1} \quad : \quad \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}_{\{\widehat{\theta}^{(b)} \leqslant \widehat{l}_{\alpha_1}\}} \doteq \alpha_1$$

$$\widehat{u}_{\alpha_2} \quad : \quad \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}_{\{\widehat{\theta}^{(b)} \leqslant \widehat{u}_{\alpha_2}\}} \doteq \alpha_2$$

Bias correction/Acceleration

That is, the empirical α_1 and α_2 quantiles, where

$$\begin{aligned}
\alpha_1 &= \Phi\left(\widehat{z}_0 + \frac{\widehat{z}_0 + z_{\alpha/2}}{1 - \widehat{a}(\widehat{z}_0 + z_{\alpha/2})}\right) \\
\alpha_2 &= \Phi\left(\widehat{z}_0 + \frac{\widehat{z}_0 + z_{1-\alpha/2}}{1 - \widehat{a}(\widehat{z}_0 + z_{1-\alpha/2})}\right)
\end{aligned}$$

where $\widehat{z}_0$ and $\widehat{a}$ are to be defined, and $z_{\alpha/2}$ and $z_{1-\alpha/2}$ are the tail quantiles of the standard normal.

Bias correction/Acceleration

For $\alpha = 0.1$, $z_{\alpha/2} = z_{0.05} = -1.645$, $z_{1-\alpha/2} = z_{0.95} = 1.645$.

- $\widehat{z}_0$ is a *bias-correction* constant defined by

$$\widehat{z}_0 = \Phi^{-1}\left(\frac{1}{B}\sum_{b=1}^{B} \mathbb{1}_{\{\widehat{\theta}^{(b)} \leqslant \widehat{\theta}\}}\right)$$

  that is, it measures the median bias of $\widehat{\theta}^{(b)}$

Bias correction/Acceleration

- $\widehat{a}$ is an *acceleration* factor which defined via *jackknife* quantities

$$\widehat{a} = \frac{\sum_{i=1}^{n} (\widehat{\theta}_{(.)} - \widehat{\theta}_{(i)})^3}{6 \left\{ \sum_{i=1}^{n} (\widehat{\theta}_{(.)} - \widehat{\theta}_{(i)})^2 \right\}^{3/2}}$$

Bias corrected accelerated (*BCa*) percentile intervals have good theoretical properties.

Permutation tests

Permutation procedures utilize resampling to construct tests of hypothesis without using parametric assumptions.

The sampling distribution of a (test) statistic is constructed using resampling from the original data, but assuming the structure implied by the *null hypothesis*.

Thus, unlike the bootstrap which retains the original data structure during resampling, permutation tests impose a hypothesis driven structure.

Permutation tests

Example: Mouse data

Recall the mouse survival time data:

$C\,(y)$	52	104	146	10	50	31	40	27	46
$T\,(z)$	94	197	16	38	99	141	23		

Suppose the null hypothesis is that there is no difference between the survival time distributions in the two groups.

Example: Mouse data

We select a test statistic, $T = T(\mathbf{Y}, \mathbf{Z})$, suitable for testing the hypothesis; could use the two-sample t-statistic assuming *unequal* variances:

$$T = \frac{\overline{Y} - \overline{Z}}{\sqrt{\frac{s_Y}{n_Y} + \frac{s_Z}{n_Z}}}$$

where s_Y and s_Z are the unbiased sample variances. This statistic is derived from the perspective of normal random sampling, but the test itself will not rely on this assumption.

Permutation tests

Example: Mouse data

The sampling distribution of T *under the null hypothesis*, H_0, is computed by noting that, under H_0, the labelling of a mouse as being in the Control or Treatment group makes no difference to their random survival time.

Therefore, we compute the statistic for each of the possible relabellings of the $n = n_Y + n_Z$ measurements consistent with the original group sizes.

Example: Mouse data

For relabelling j, we choose *at random without replacement* n_Y measurements and denote them $\mathbf{y}_{(j)}$, denote the remaining n_Z measurements $\mathbf{z}_{(j)}$, and compute the resulting relabelled statistic $T_{(j)}$.

When all N possible relabellings have been examined, we have computed the statistics

$$t_{(1)}, \ldots, t_{(N)}$$

which define the permutation distribution.

Example: Mouse data

The *critical values* for the test are defined via the empirical distribution of

$$t_{(1)}, \ldots, t_{(N)}$$

If $t = T(\mathbf{y}, \mathbf{z})$ is the observed test statistic for the original data, then empirical probabilities

$$\Pr_0[T_{(*)} \leqslant t] = \frac{1}{N} \sum_{j=1}^{N} \mathbb{1}_{[t_{(j)}, \infty)}(t)$$

can be used to define the p-value *(achieved significance level)* in the two-tailed test as

$$\Pr_0[T_{(*)} \leqslant -|t|] + \Pr_0[T_{(*)} \geqslant |t|]$$

Permutation tests

Evaluating *all* possible relabellings can be a computationally demanding step; for the two-sample problem, the number of relabellings is

$$N = \binom{n}{n_Y}$$

For the mouse data, $n_Y = 9, n_Z = 7$, so

$$N = \binom{16}{9} = 11440$$

which is not too large.

If N gets too large, a fixed number of *randomly chosen* relabellings can be generated in place of the exhaustive calculation.

Permutation tests

If the permutations are randomly chosen, the procedure is termed a *Monte Carlo permutation* (or *Monte Carlo randomization*) procedure.

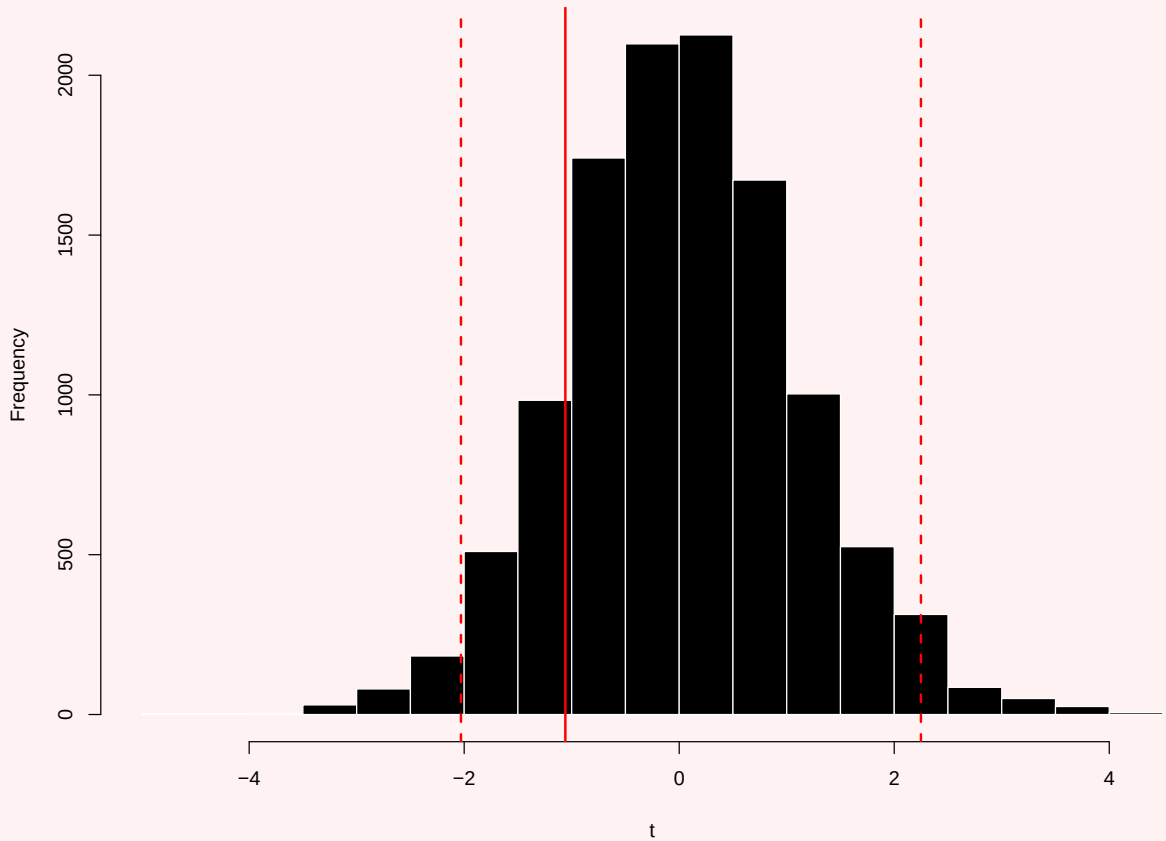
The choice of the Monte Carlo sample size depends on the quantity being estimated.

In any case, the procedure is *distribution-free*, but is reliant on being able to construct a suitable test statistic.

Permutation tests: Example

Example: Mouse data

Permutation p-value = 0.3068.



Permutation tests: Other statistics

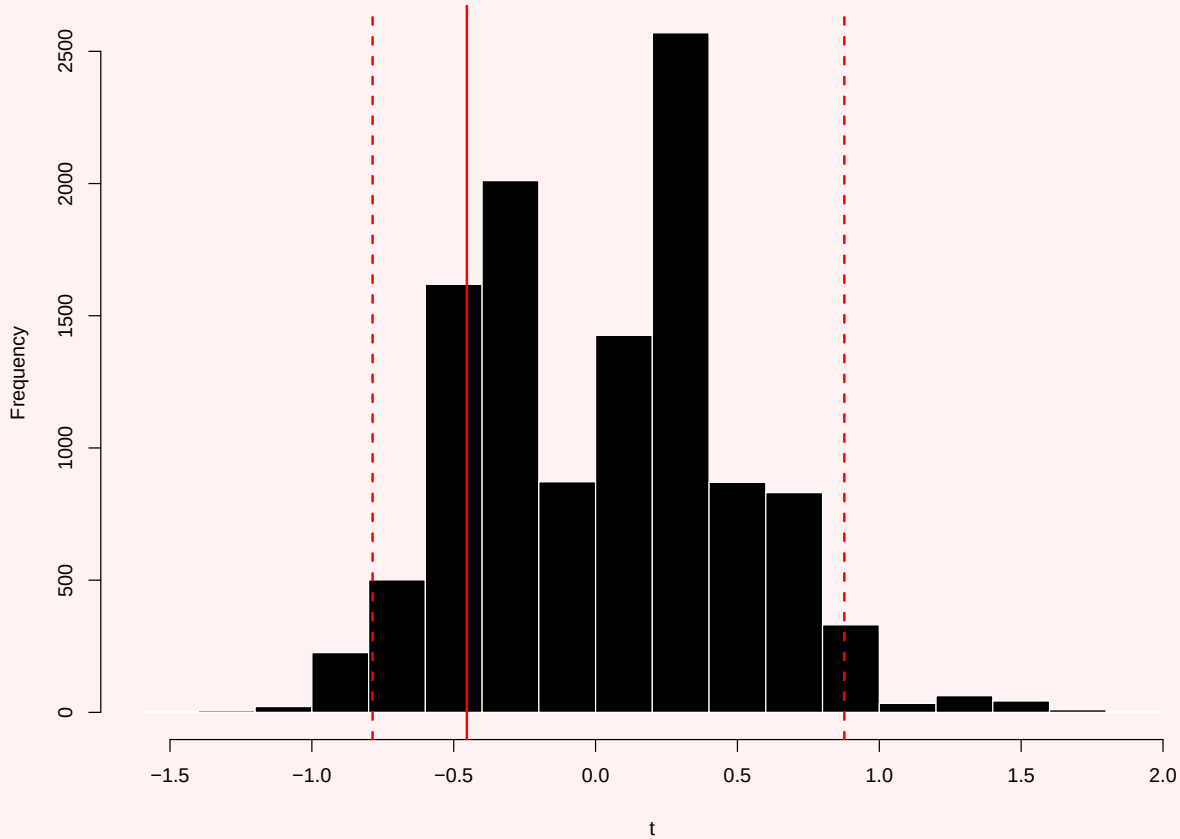
The permutation procedure can be used

- with other test statistics
  - tests for equality of variance
  - Kolmogorov-Smirnov
- in more general data collection situations
  - inference in regression
  - ANOVA
  - discrete data (chi-squared test, Fisher's Exact test)

Permutation tests: Other statistics

Example: Mouse data

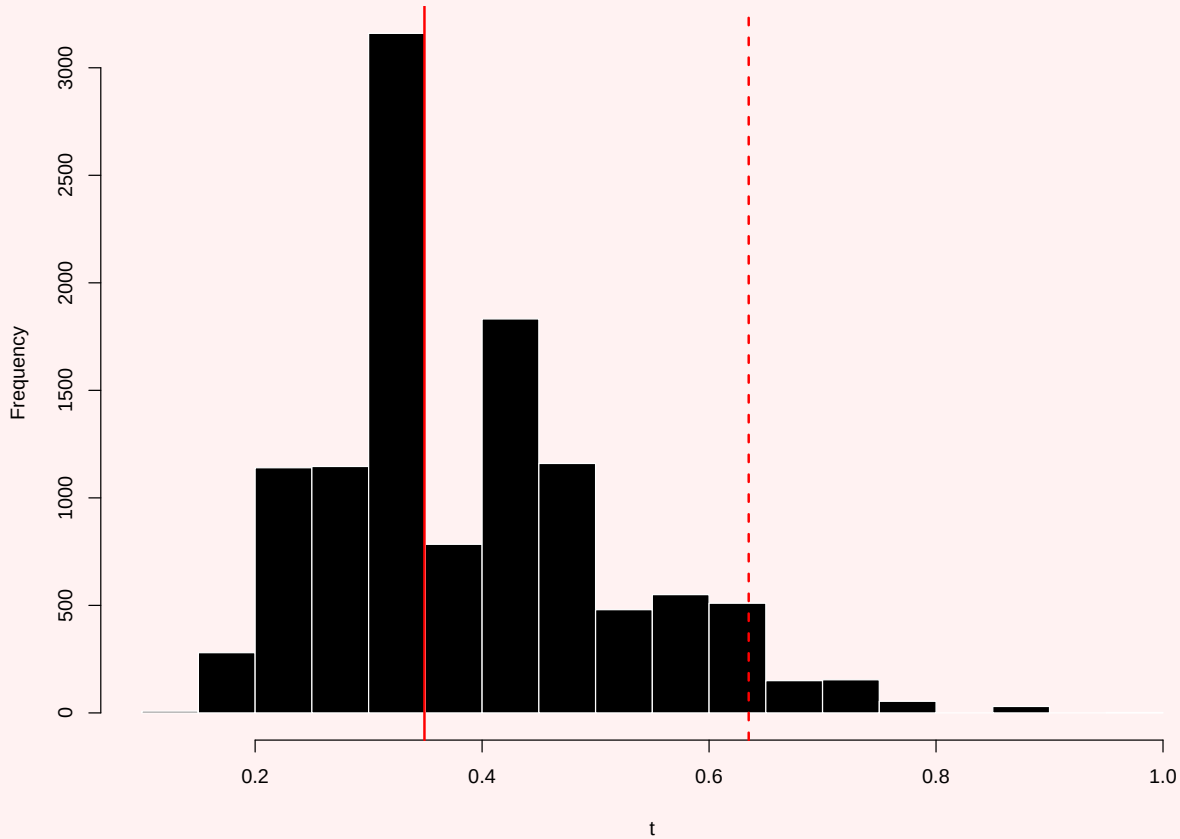
Test statistic $T = \log(s_Y/s_Z)$: two-sided p-value 0.3156



Permutation tests: Other statistics

Example: Mouse data

Kolmogorov-Smirnov test one-sided p-value 0.5727 (32 possible values of $t_{(j)}$.



Hypothesis testing: summary

In general, hypothesis testing can be carried out using the bootstrap due to the connection between hypothesis testing and confidence intervals.

That is, a $(1-\alpha)$% confidence interval for θ that contains those θ_0 values that, in light of the observed data and estimate $\widehat{\theta}$, would NOT lead to the rejection of the null hypothesis

$$H_0 \ : \ \theta = \theta_0$$

at significance level α.