MATH 680 Computer Intensive Statistics

Dr David A. Stephens

Department of Mathematics & Statistics
Room 1225, Burnside Hall

d.stephens@math.mcgill.ca
www.math.mcgill.ca/~dstephens/680/

Computational Methods in Statistics

Modern statistics is only possible because of advances in computational power. Virtually all (applied) statistics relies on

- Statistics packages with GUIs: Splus, SPSS, Stata ...
- Command line programming: R, SAS ...
- High-level language programming: C/C++ ...

to implement statistical methods.

Course content

This course will focus on two sorts of numerical computational methods:

- **Deterministic**
  - Optimization
  - Root-finding
- **Stochastic**
  - Random number generation
  - Bootstrap
  - Monte Carlo
  - Markov chain Monte Carlo

In addition, standard computational/numerical methods will be covered (solving linear systems, matrix inversion, Cholesky and other matrix decompositions etc.)

The need for computation

Estimation is a common statistical activity:

- Maximum likelihood (ML)
- Method of Moments (MM)
- (Generalized) Estimating Equations ((G)EE)
- Bayesian estimation

When analytical results are not available, the first three require either **optimization** or **root-finding**; the fourth typically requires **integration**.

Example: ML estimation

For an iid sample $\boldsymbol{x}$ from parametric density $f(x|\theta)$, the ML estimate is

$$\widehat{\theta} = \arg\max_{\theta \in \Theta} L(\theta; \boldsymbol{x})$$

(i.e. obtained by optimization). However if

$$\ell_\theta(\theta; \boldsymbol{x}) = \frac{\partial}{\partial \theta} \log L(\theta; \boldsymbol{x})$$

under regularity conditions, we have that $\widehat{\theta}$ is the solution of the equation

$$\ell_\theta(\theta; \boldsymbol{x}) = \mathbf{0}$$

(i.e. obtained by root-finding).

If θ is a $(p \times 1)$ vector, then this is a p-dimensional optimization or equation system.

Notation

A note on notation: derivatives will be denoted by subscripts, that is, for function g, a function of d-dimensional vector t,

$$g_t(t) = \frac{\partial g(t)}{\partial t} \qquad (d \times 1 \text{ vector})$$

$$g_{tt}(t) = \frac{\partial^2 g(t)}{\partial t \partial t^\intercal} \qquad (d \times d \text{ matrix})$$

etc. The notation

$$\dot{g}(t) \qquad \ddot{g}(t)$$

etc. will also be used.

Example: MM estimation

For MM estimation, if

$$\mu_k(\theta) = \mathbb{E}_{f_{X|\theta}}[X^k]$$

then the MM estimate is the solution of the system

$$\sum_{i=1}^{n}(X_i^k - \mu_k(\theta)) = 0 \qquad k = 1, \ldots, p$$

(i.e. obtained by root-finding).

Example: Logistic regression

Suppose data comprise n data

- Response $Y \in \{0, 1\}$
- Covariate Predictors $X_1, \ldots, X_p$

where

$$\theta_i = \mathbb{P}[Y_i = 1 | \boldsymbol{X}_i = \boldsymbol{x}_i]$$

is modelled via

$$\eta_i = \log\left(\frac{\theta_i}{1-\theta_i}\right) = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip} = \boldsymbol{X}_i^\intercal \beta$$

say; write

$$\theta_i \equiv \theta_i(\beta).$$

Example: Logistic regression

The corresponding likelihood is

$$L(\beta; \boldsymbol{x}) = \prod_{i=1}^{n} \left\{ \frac{\exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}}{1 + \exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}} \right\}^{Y_i} \left\{ \frac{1}{1 + \exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}} \right\}^{1-Y_i}$$

with log-likelihood

$$\ell(\beta; \boldsymbol{x}) = \sum_{i=1}^{n} Y_i \boldsymbol{x}_i^\mathsf{T} \beta - \sum_{i=1}^{n} \log\left(1 + \exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}\right)$$

To compute the ML estimate, take partial derivatives wrt the components of β in turn: for $j = 1, \ldots, p$

$$\ell_{\beta_j}(\beta; \boldsymbol{x}) = \sum_{i=1}^{n} Y_i X_{ij} - \sum_{i=1}^{n} \left\{ X_{ij} \frac{\exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}}{1 + \exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}} \right\}$$

Example: Logistic regression

This simplifies to β: for $j = 1, \ldots, p$

$$\ell_{\beta_j}(\beta; \boldsymbol{x}) = \sum_{i=1}^{n} x_{ij} \left(Y_i - \left\{ \frac{\exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}}{1 + \exp\{\boldsymbol{x}_i^\mathsf{T} \beta\}} \right\} \right)$$

or

$$\ell_{\beta_j}(\beta; \boldsymbol{x}) = \sum_{i=1}^{n} x_{ij} \left(Y_i - \theta_i(\beta) \right)$$

For β_0, the equation is

$$\ell_{\beta_0}(\beta; \boldsymbol{x}) = \sum_{i=1}^{n} \left(Y_i - \theta_i(\beta) \right)$$

Example: Logistic regression

Thus the system of equations that must be solved to find the ML estimate is the $(p+1) \times 1$ system

$$\sum_{i=1}^{n} \eta_{\beta i}(\beta)(Y_i - \theta_i(\beta)) = \mathbf{0}$$

where $\eta_{\beta i}(\beta)$ is the $(p+1) \times 1$ vector with jth component

$$\eta_{\beta ij}(\beta) = \frac{\partial \eta_i(\beta)}{\partial \beta_j}.$$

for $j = 0, 1, \ldots, p$.

The solution is not analytically available, and numerical methods are required.

Example: Logistic regression

Standard errors are typically also required. In regular problems, as $n \longrightarrow \infty$,

$$\sqrt{n}(\widehat{\beta} - \beta_0) \xrightarrow{d} \mathcal{N}(0, \mathcal{I}(\beta_0)^{-1})$$

where β_0 is the true value of β, and

$$\mathcal{I}(\beta_0) = \mathbb{E}\left[S_\beta S_\beta^\intercal\right] = -\mathbb{E}\left[S_{\beta\beta}\right]$$

is the Fisher Information with

$$S_\beta \equiv S_\beta(X;\beta) = \frac{\partial \log f(X|\beta)}{\partial \beta} \qquad (p+1) \times 1$$

$$S_{\beta\beta} \equiv S_{\beta\beta}(X;\beta) = \frac{\partial^2 \log f(X|\beta)}{\partial \beta \partial \beta^\intercal} \qquad (p+1) \times (p+1)$$

and expectations are taken at $\beta = \beta_0$.

Example: Logistic regression

$\mathcal{I}(\beta_0)$ is typically estimated by

$$\widehat{I}(\widehat{\beta}) = \frac{1}{n}\sum_{i=1}^{n} S_\beta(x_i;\widehat{\beta})S_\beta^\intercal(x_i;\widehat{\beta})$$

or

$$\widehat{I}(\widehat{\beta}) = -\frac{1}{n}\sum_{i=1}^{n} S_{\beta\beta}(x_i;\widehat{\beta})$$

Thus the standard errors are available relatively easily once $\widehat{\beta}$ has been obtained.

Taylor approximation

Recall that, by a Taylor expansion of the log-likelihood around $\theta = \widehat{\theta}$

$$\begin{aligned} \ell(\theta) &\simeq \ell(\widehat{\theta}) + \ell_\theta(\widehat{\theta})(\theta - \widehat{\theta}) + \frac{1}{2}(\theta - \widehat{\theta})^\intercal \ell_{\theta\theta}(\widehat{\theta})(\theta - \widehat{\theta}) \\ &= \ell(\widehat{\theta}) + \frac{1}{2}(\theta - \widehat{\theta})^\intercal \ell_{\theta\theta}(\widehat{\theta})(\theta - \widehat{\theta}) \end{aligned}$$

as $\ell_\theta(\widehat{\theta}) = \mathbf{0}$. The matrix

$$\ell_{\theta\theta}(\theta)$$

is the matrix of second partial derivatives of ℓ sometimes termed the **Hessian** matrix for ℓ. We typically need $\ell_{\theta\theta}(\widehat{\theta})$; this is merely $-n\widehat{I}(\widehat{\theta})$.

Taylor approximation

As $L(\theta; \boldsymbol{x}) = \exp\{\ell(\theta)\}$, it follows that for large n

$$L(\theta; \boldsymbol{x}) \simeq c(\widehat{\theta}) \exp\left\{-\frac{n}{2}(\theta - \widehat{\theta})^{\intercal} \widehat{I}(\widehat{\theta})(\theta - \widehat{\theta})\right\}$$

Hence in a Bayesian calculation

$$\pi(\theta|\boldsymbol{x}) \propto L(\theta; \boldsymbol{x})\pi(\theta)$$

and $L(\theta; \boldsymbol{x})$ can be approximated by a Gaussian likelihood.

The Laplace Approximation

Using a similar strategy: for an arbitrary unnormalized multivariate density $f(z) = \exp\{h(z)\}$, say, if $z^* \in \mathbb{R}^d$ corresponds to the mode of f,

$$\begin{aligned} h(z) &\doteq h(z^*) + \frac{1}{2}(z - z^*)^\intercal \ddot{h}(z^*)(z - z^*) \\ &= h(z^*) - \frac{1}{2}(z - z^*)^\intercal H(z^*)(z - z^*) \end{aligned}$$

say. Therefore

$$f(z) \doteq f(z^*) \exp\left\{-\frac{1}{2}(z - z^*)^\intercal H(z^*)(z - z^*)\right\}$$

and

$$\int f(z)\, dz \doteq f(z^*) \frac{(2\pi)^{d/2}}{|\det H(z^*)|^{1/2}}$$

The Laplace Approximation: Example

If $f(z) = z^{\alpha-1}e^{-z}$, then $h(z) = (\alpha-1)\log z - z$ and

$$\begin{aligned} \dot{h}(z) &= \frac{\alpha-1}{z} - 1 \qquad \therefore \qquad z^* = \alpha - 1 \\ H(z) &= -\ddot{h}(z) = \frac{(\alpha-1)}{z^2} \qquad \therefore \qquad H(z^*) = 1/(\alpha-1) \end{aligned}$$

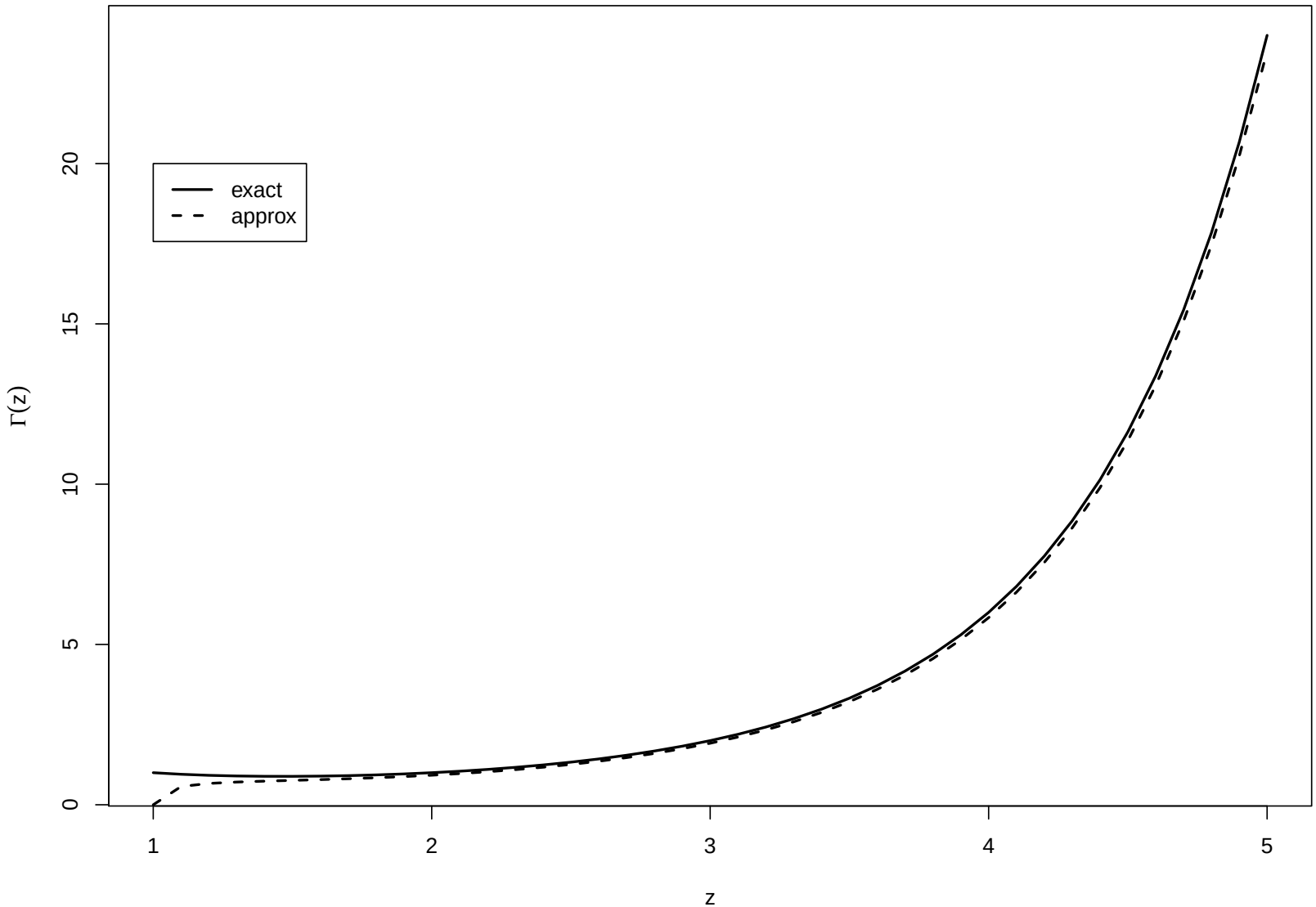
so therefore

$$\int f(\boldsymbol{z})\, d\boldsymbol{z} \doteq \sqrt{(2\pi)}(\alpha-1)^{\alpha-1/2}e^{-(\alpha-1)}$$

Here the normalizing constant is actually $\Gamma(\alpha)$.

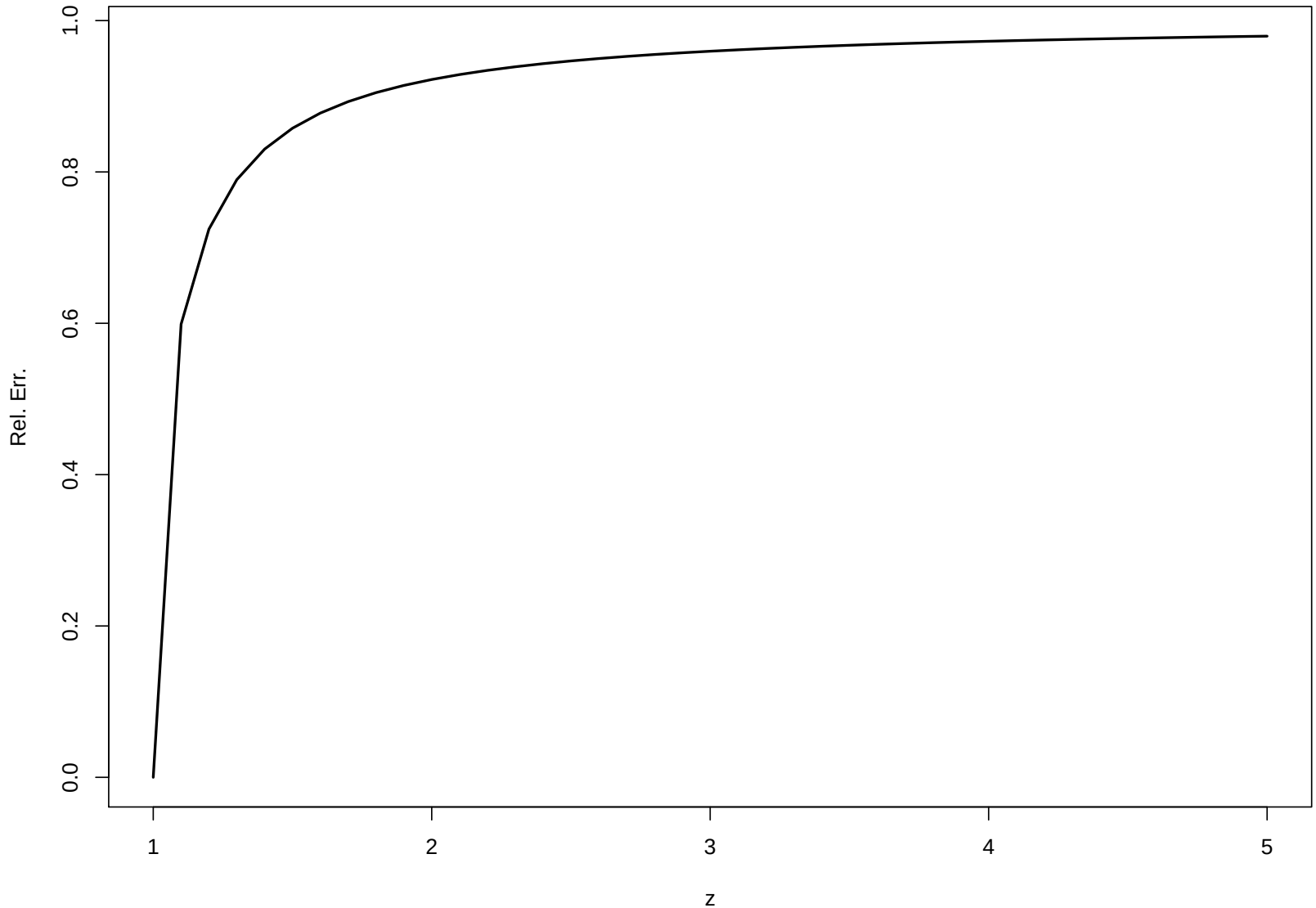
The Laplace Approximation: Example

Approximation:



The Laplace Approximation: Example

Relative error:



Linear Model

In the linear model

$$Y_i = \boldsymbol{X}_i^\intercal \beta + \epsilon_i$$

where $\epsilon_i, i = 1, \ldots, n$ are zero mean, variance σ^2 random variables that are mutually independent (and independent of $\boldsymbol{X}_i$).

Following the same derivation as above, to estimate β,

$$\ell_{\beta_j}(\beta; \boldsymbol{x}) = \sum_{i=1}^{n} x_{ij} \left(Y_i - \boldsymbol{x}_i^\intercal \beta \right) = 0$$

or equivalently, as a $(p+1) \times 1$ system

$$\boldsymbol{X}^\intercal \boldsymbol{X} \beta = \boldsymbol{X}^\intercal \boldsymbol{Y}$$

where $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\intercal$, and $\boldsymbol{X}$ is the $n \times (p+1)$ matrix formed by binding the rows $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$.

Linear Model

Provided $\boldsymbol{X}^\intercal \boldsymbol{X}$ is non-singular (positive definite), this equation has a unique solution

$$\widehat{\beta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{Y}.$$

To compute this, we need

$$(\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}$$

i.e. we need to invert a $(p+1) \times (p+1)$ matrix. This is feasible (usually) using, say

`solve`

in R.

Cholesky decomposition

Alternatively:

- **Cholesky decomposition:** For a positive definite matrix A, write

$$A = LL^\intercal$$

  where L is a lower-triangular matrix. A^{-1} is more readily available compute as L^{-1} is easy to compute.

  We need to apply this to $\boldsymbol{X}^\intercal \boldsymbol{X}$.

Singular value decomposition

Suppose that A is an $m \times n$ real matrix. Then A can be decomposed

$$A = UDV^\intercal$$

where

- U is $m \times m$ such that $U^\intercal U = UU^\intercal = \boldsymbol{I}_m$
- D is $m \times n$ with non-negative elements only on the leading diagonal.
- V is $n \times n$ such that $V^\intercal V = VV^\intercal = \boldsymbol{I}_n$

Singular value decomposition

If A is $m \times m$ and **positive definite**, then D is square and has the m strictly positive **eigenvalues** of A on its diagonal.

In this case U comprises the **eigenvectors** of A as its columns, and

$$U^{-1} = V^{\intercal}$$

and

$$A^{-1} = UD^{-1}V^{\intercal}$$

QR decomposition

- **QR decomposition:** Write

$$\boldsymbol{X} = QR$$

where

  - Q is a $n \times (p+1)$ orthonormal matrix, $Q^\intercal Q = \boldsymbol{I}$,
  - R is a $(p+1) \times (p+1)$ upper-triangular matrix

Then

$$\boldsymbol{X}^\intercal \boldsymbol{X}\beta = \boldsymbol{X}^\intercal \boldsymbol{Y} \qquad \therefore \qquad R^\intercal Q^\intercal QR\beta = R^\intercal Q^\intercal \boldsymbol{Y}$$

and if $Q^\intercal \boldsymbol{Y} = \boldsymbol{Y}^*$, say, then

$$R\beta = \boldsymbol{Y}^*$$

This is easy to solve, as R is triangular.

QR decomposition

Gram-Schmidt orthogonalization: For $n \times p$ matrix A, write

$$A = \left[A_{(1)}, \ldots, A_{(p)}\right]$$

Then set

- At step 1: $B_{(1)} = A_{(1)}$ then $Q_{(1)} = B_{(1)}/||B_{(1)}||$.
- At step 2:

$$B_{(2)} = A_{(2)} - \frac{A_{(2)}^\intercal Q_{(1)}}{Q_{(1)}^\intercal Q_{(1)}} Q_{(1)} \qquad \text{then} \qquad Q_{(2)} = \frac{B_{(2)}}{||B_{(2)}||}$$

- At step $k = 3, \ldots, p$:

$$B_{(k)} = A_{(k)} - \sum_{j=1}^{k-1} \frac{A_{(k)}^\intercal Q_{(j)}}{Q_{(j)}^\intercal Q_{(j)}} Q_{(j)} \qquad \text{then} \qquad Q_{(k)} = \frac{B_{(k)}}{||B_{(k)}||}$$

Then

$$Q = \left[Q_{(1)}, \ldots, Q_{(p)}\right]$$

is the desired matrix.

QR decomposition

The matrix R is the upper-triangular matrix with (j,k)th element

$$Q_{(j)}^{\intercal} A_{(k)} \qquad j \leqslant k.$$

Thus we can compute

$$(\boldsymbol{X}^{\intercal}\boldsymbol{X})^{-1}$$

without matrix inversion. This is often faster than using matrix inversion, and is the default approach used by R in the function `lm`.

Note: Other methods of orthogonalization are available.

Example: Ridge Regression

If $\boldsymbol{X}^\intercal \boldsymbol{X}$ is ill-conditioned, or almost singular, the **ridge regression** estimator is sometimes used:

$$\widetilde{\beta} = (\boldsymbol{X}^\intercal \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\intercal \boldsymbol{Y}.$$

for scalar λ. More generally, could use

$$\widetilde{\beta} = (\boldsymbol{X}^\intercal \boldsymbol{X} + \Lambda^\intercal \Lambda)^{-1} \boldsymbol{X}^\intercal \boldsymbol{Y}.$$

This is the solution to the regularized minimization problem

$$\min_{\beta} \|Y - \boldsymbol{X}\beta\|^2 + \|\Lambda\beta\|^2$$

and is termed **Tikhonov regularization**.

Matrix Inverse Identity

If $\boldsymbol{X}$ is $n \times p$, $\boldsymbol{X}^\intercal \boldsymbol{X}$ is $p \times p$:

- Typically $n \gg p$, and p is reasonably small, and inversion is relatively straightforward.
- Sometimes $p \gg n$, so inverting the large, sparse matrix

$$\boldsymbol{X}^\intercal \boldsymbol{X} + \lambda \boldsymbol{I}$$

  is more challenging.

Matrix Inverse Identity

Suppose

- A is $p \times p$
- U is $p \times n$
- C is $n \times n$
- V is $n \times p$

Then

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

For ridge regression: set

$$A = \lambda \boldsymbol{I} \qquad U = \boldsymbol{X}^\intercal \qquad C = \boldsymbol{I} \qquad V = \boldsymbol{X}$$

so that

$$(\boldsymbol{X}^\intercal \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} = \lambda^{-1}\boldsymbol{I} - \lambda^{-2}\boldsymbol{X}^\intercal(\boldsymbol{I} + \lambda^{-1}\boldsymbol{X}\boldsymbol{X}^\intercal)^{-1}\boldsymbol{X}$$

Matrix Inverse Identity

This identity allows us to exchange an $p \times p$ inversion for an $n \times n$ inversion; it will be useful when p is large and n is small.

- Note that in the general version

$$A = \Lambda^\intercal \Lambda \qquad \text{and} \qquad C$$

must be inverted.

Other forms of regularization

In certain statistical estimation problems, the form of "regularization" is different

- LASSO

$$\min_{\beta} \|\boldsymbol{y} - \boldsymbol{X}\beta\|^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

- L_α

$$\min_{\beta} \|\boldsymbol{y} - \boldsymbol{X}\beta\|^2 + \lambda \sum_{j=1}^{p} |\beta_j|^\alpha$$

- Elastic net

$$\min_{\beta} \|\boldsymbol{y} - \boldsymbol{X}\beta\|^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} |\beta_j|^2$$

Minimization cannot be achieved analytically.

Example: Gaussian Markov Random Field

A (stationary) **Gaussian Markov Random Field** is a probabilistic model describing the co-variation of random variables having different temporal or spatial locations.

At points $x_1, \ldots, x_n$ in 1- or 2-dimensions, the joint distribution of random quantities $Z_1, \ldots, Z_n$ associated with these locations is multivariate Gaussian

$$\mathbb{C}ov[Z_i, Z_j] = \sigma(||x_i - x_j||) = \Sigma_{ij}$$

that is

$$\boldsymbol{Z} \sim \mathcal{N}_n(\boldsymbol{\mu}(\boldsymbol{x}), \Sigma)$$

The covariance function $\sigma(.)$ is often specified parametrically:

$$\sigma(d) = \sigma^2 \exp\{-(d/\lambda)^\alpha\}$$

Example: Gaussian Markov Random Field

This model is used in spatial modelling.

- risk surface analysis; using a binomial logistic model

$$Y_i \sim Binomial(n_i, \theta_i) \qquad \theta_i = \frac{\exp\{Z_i\}}{1 + \exp\{Z_i\}}$$

  or a log-linear Poisson model

$$Y_i \sim Poisson(\theta_i) \qquad \theta_i = \exp\{Z_i\}$$

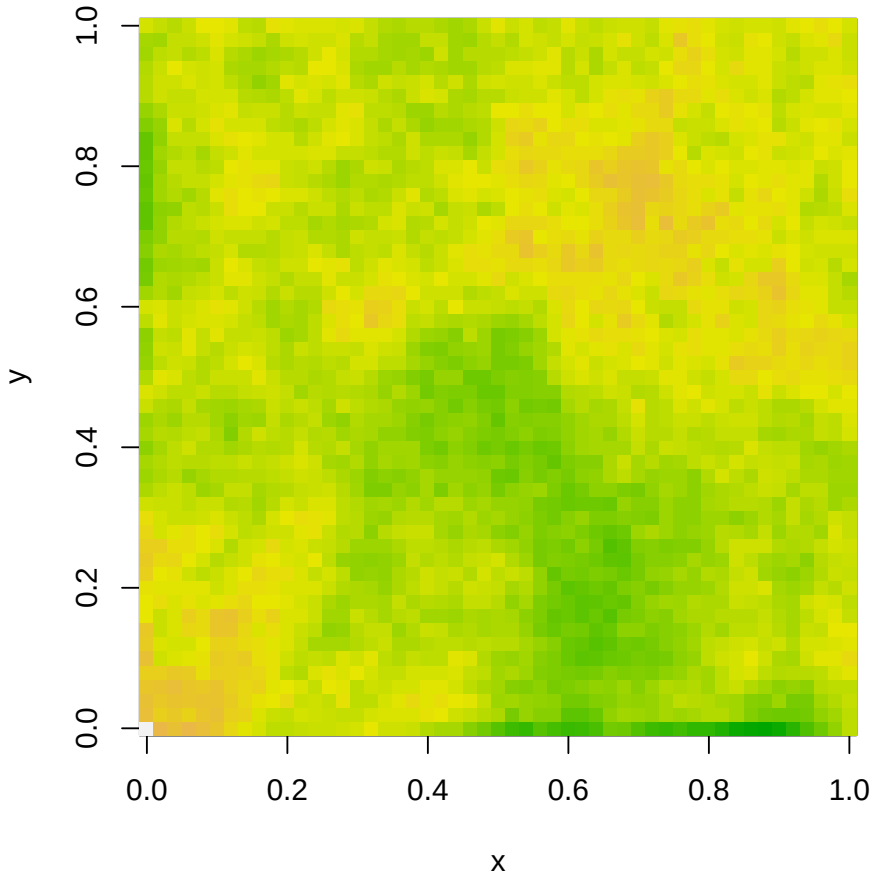
  can build a model for observed response data conditional on $\boldsymbol{Z}$.

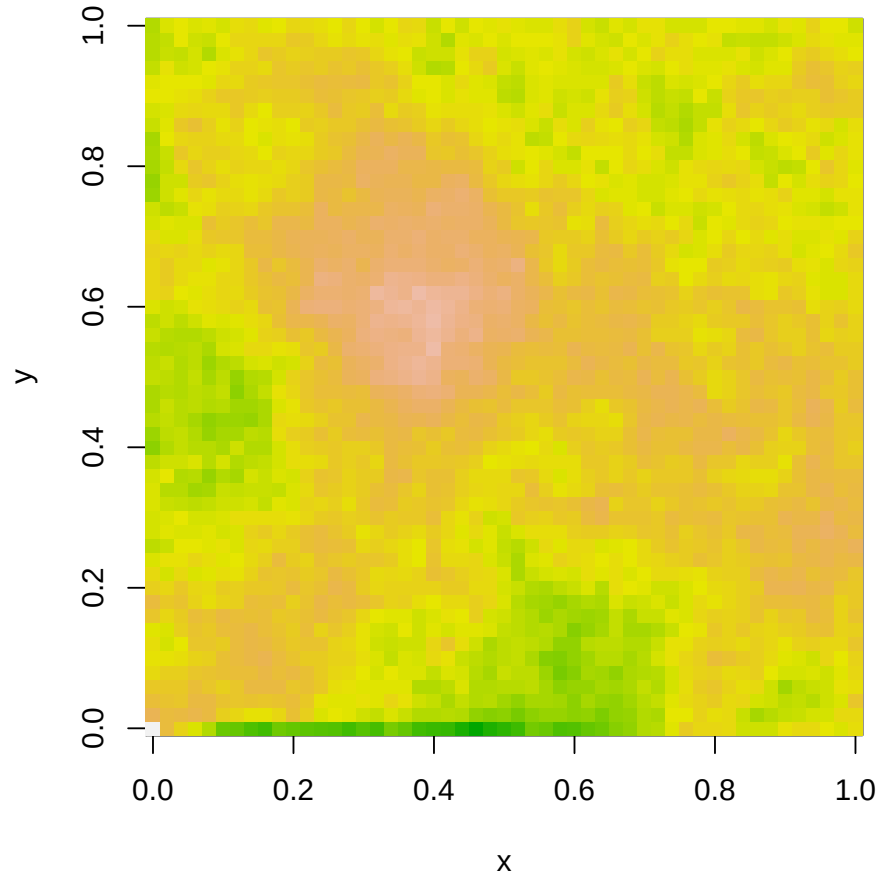
If n is fairly small, and the covariance function is reasonably well-behaved, then it is quite easy to study these models using simulation. However the nature of Σ can pose difficulties

- if n is large, and Σ is not sparse
- if Σ is almost singular

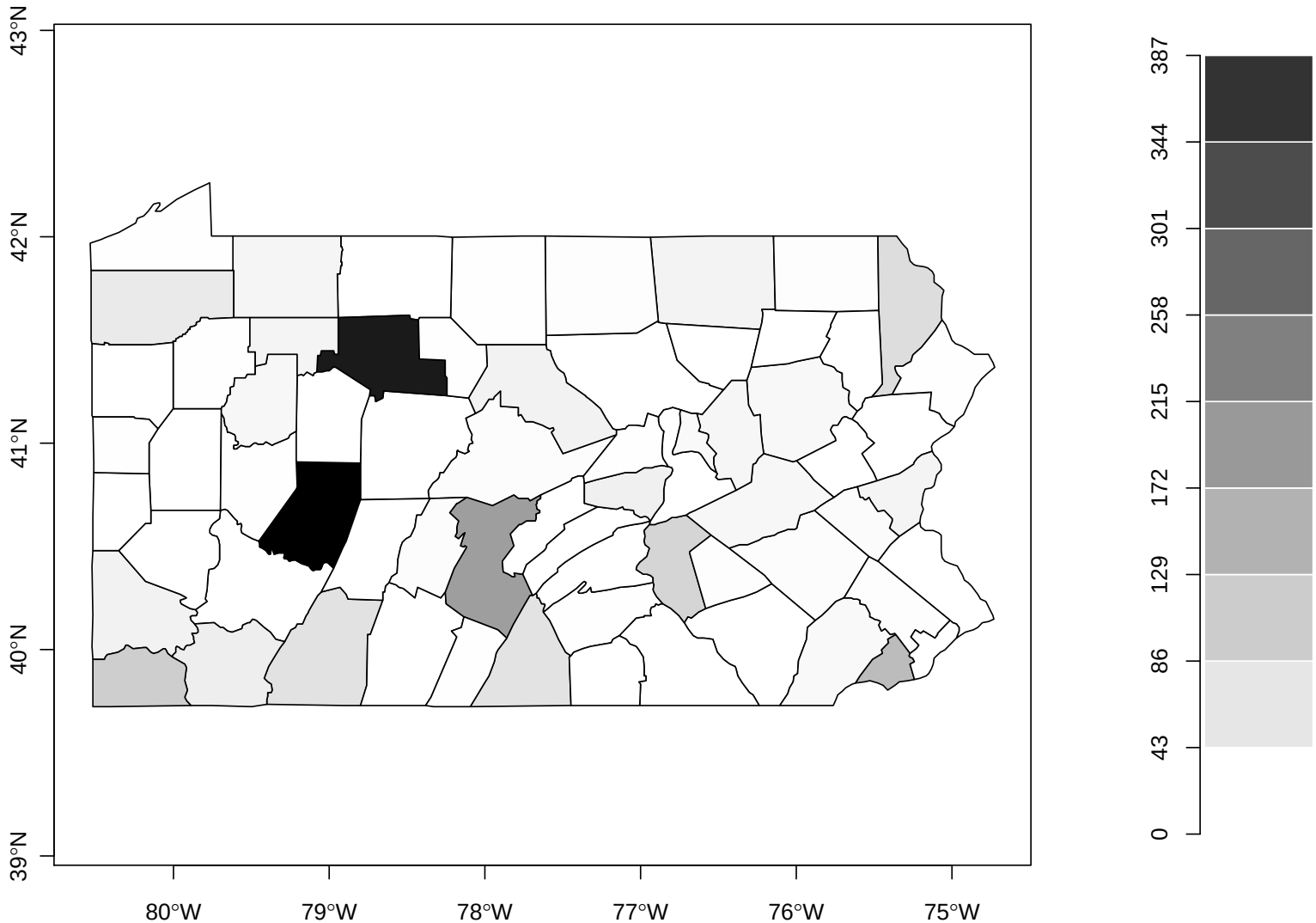
as a Cholesky decomposition of Σ is typically required.

Example: Gaussian Markov Random Field

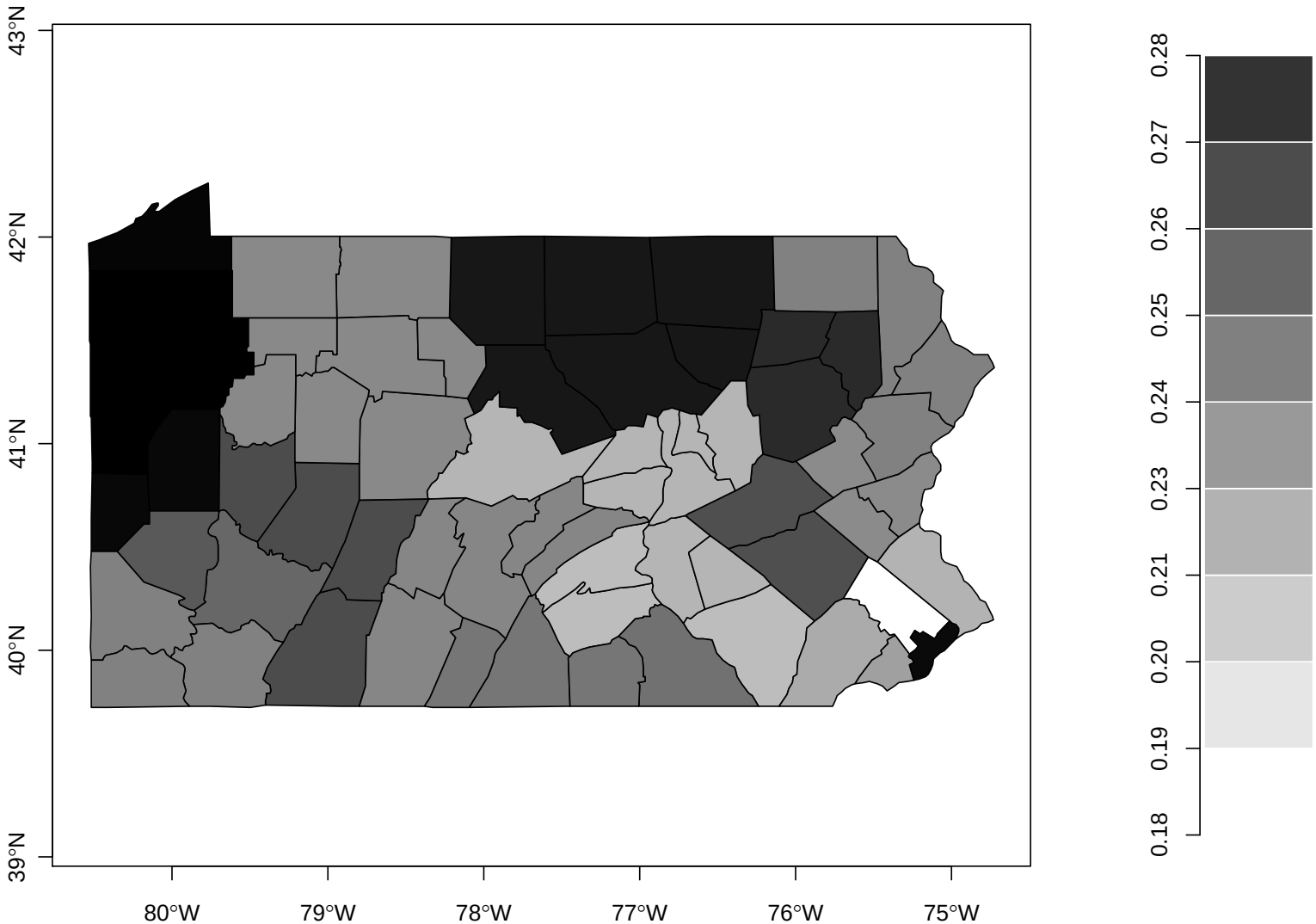




Example: Lung Cancer Cases in Pennsylvania



Example: Smoking rates in Pennsylvania



Example: Ising Model

Consider a collection of binary random variables $\{X_i\}$ placed on a rectangular $(N \times M)$ lattice, with joint mass function

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

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

$$2^{NM}$$

elements. This model is used in statistical physics and image analysis; it is not analytically tractable, but its properties can be studied using **simulation** in a **Monte Carlo** approach.

Example: Ising Model

Simulation of variates from a **discrete** distribution $f(.)$ is in principle straightforward:

1. List the elements in the support of f: $x_1, x_2, \ldots$, with their associated probabilities $p_1, p_2, \ldots$, and cumulative probabilities $P_1, P_2, \ldots$

$$P_k = \sum_{j=1}^{k} p_j \qquad k = 1, 2, \ldots$$

2. Obtain a single random variate u from a $Uniform(0, 1)$ density.
3. Return the value $x = x_k$ as a simulated value from f, where

$$k = \max\{j : P_j \leqslant u\}$$

Example: Ising Model

This method is based on the inverse CDF transformation result

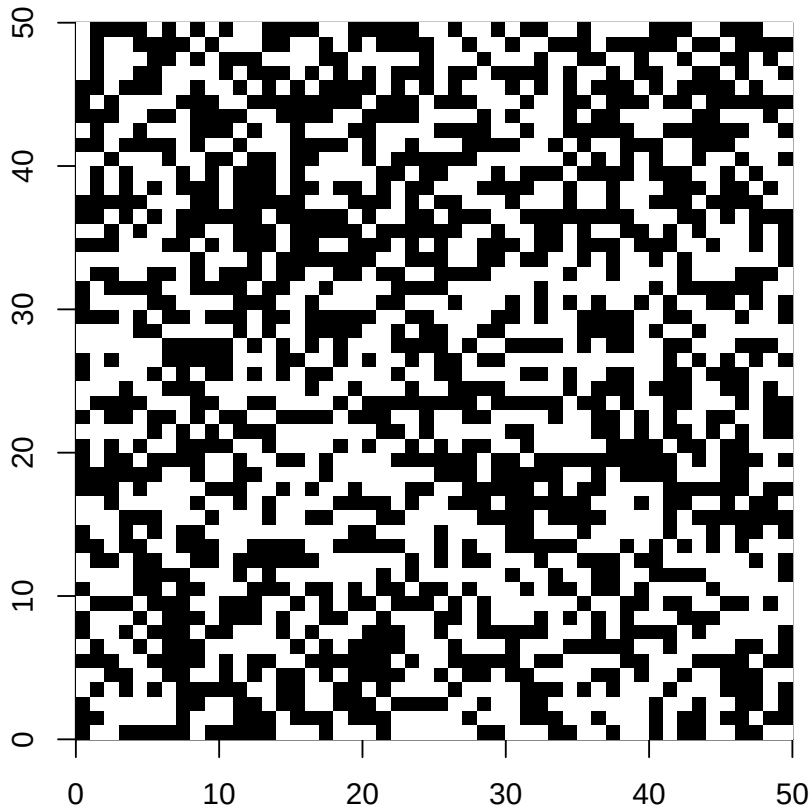
$$\text{If } X \text{ has CDF } F_X, \text{ then } U = F_X(X) \sim \mathit{Uniform}(0,1).$$

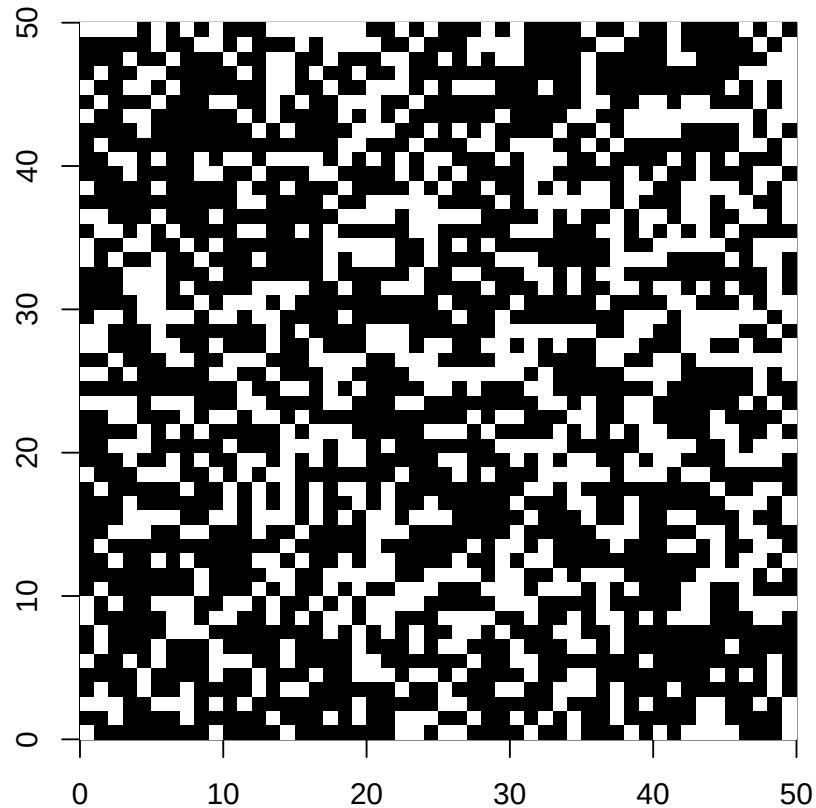
This Ising model mass function has a relatively complicated form; in particular the normalizing constant $Z(\beta)$ is difficult to compute, so that the probabilities $p_1, p_2, \ldots$ in the algorithm are only known up to proportionality.

Therefore, sampling from the Ising model is non-trivial.

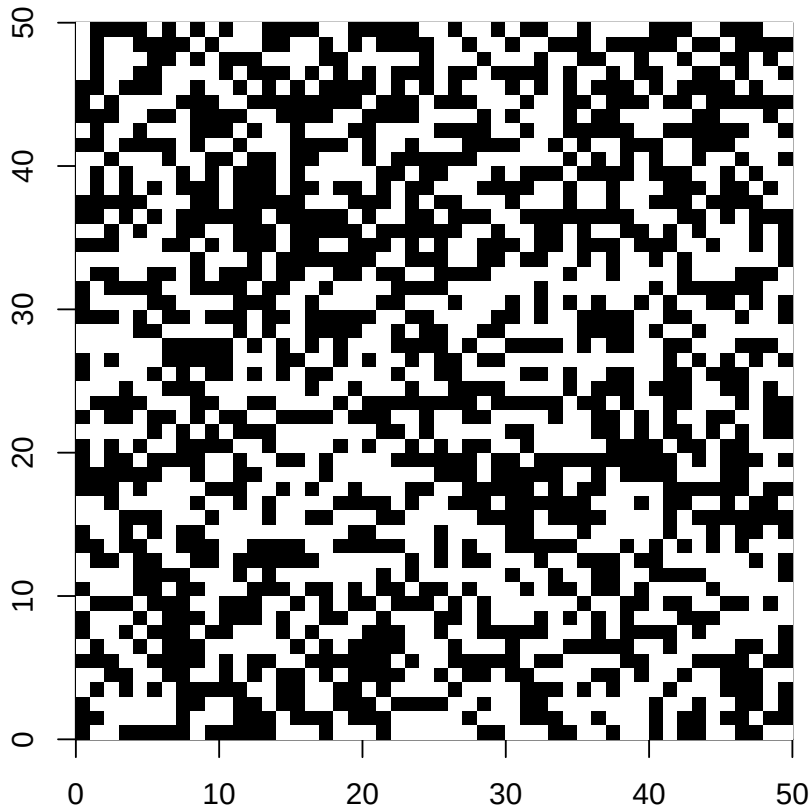
Also: how to achieve 2. ?

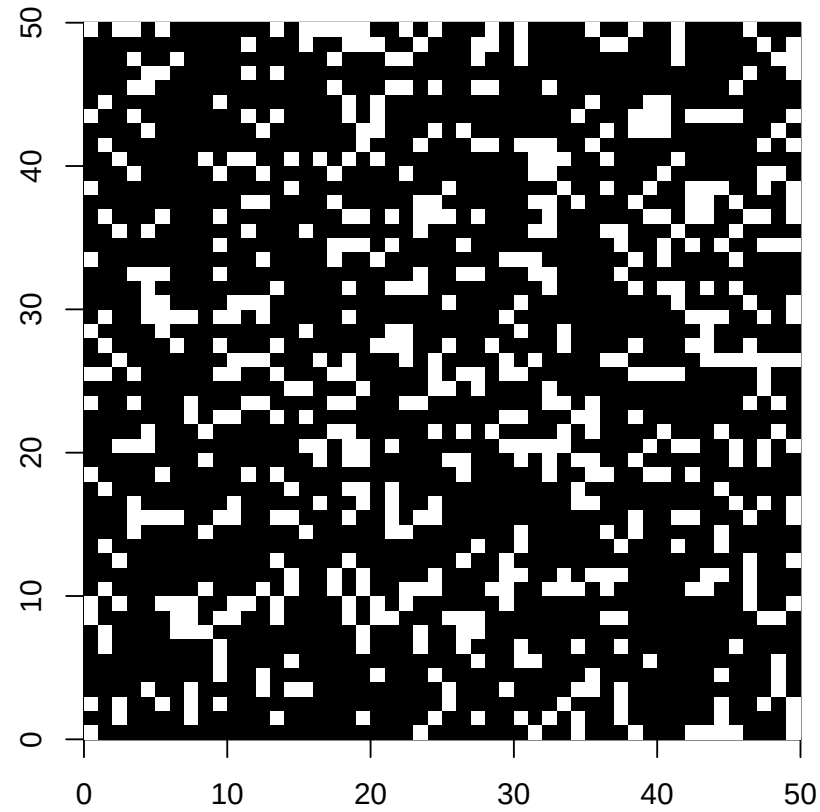
Ising Model: $\beta = 0.1$



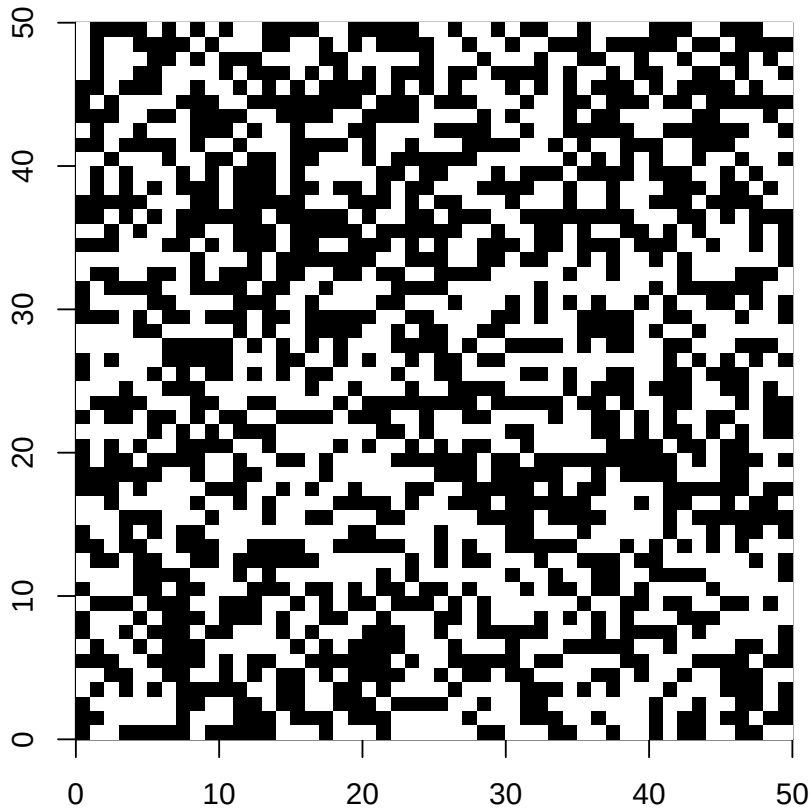


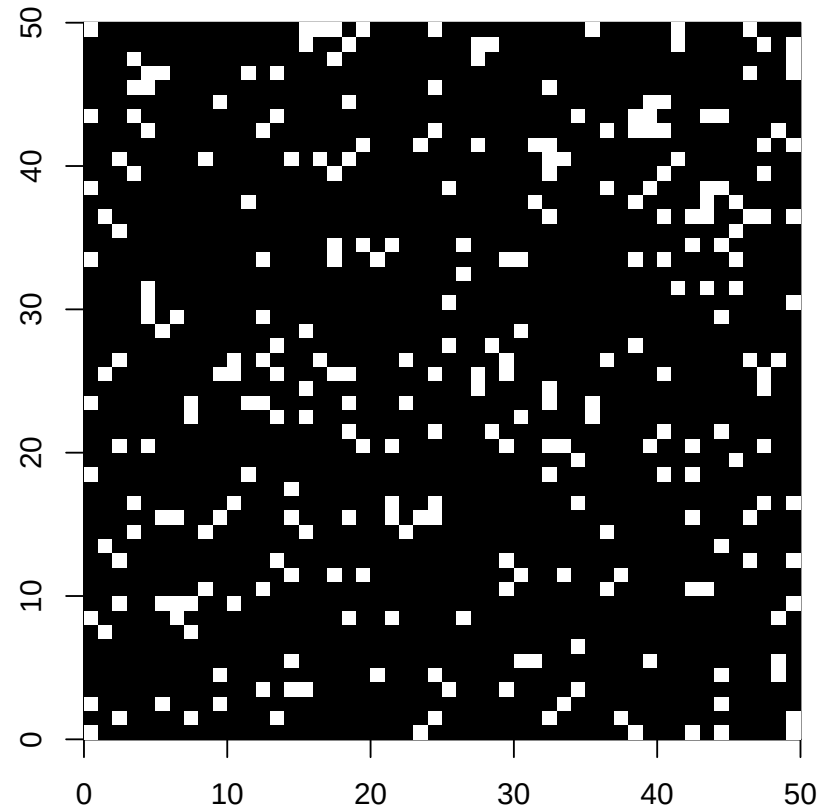
Ising Model: $\beta = 0.2$



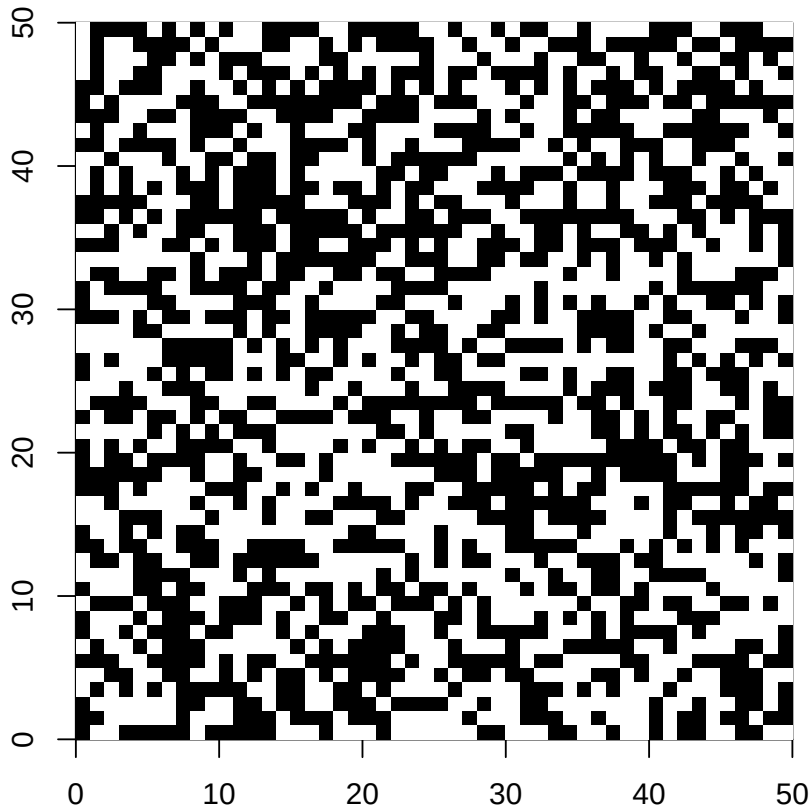


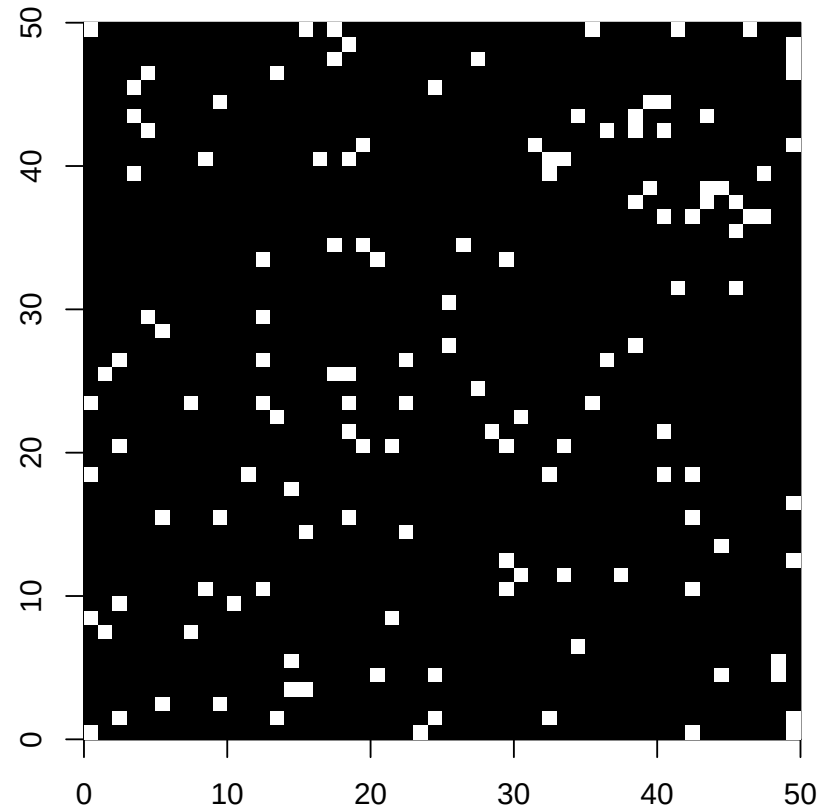
Ising Model: $\beta = 0.3$



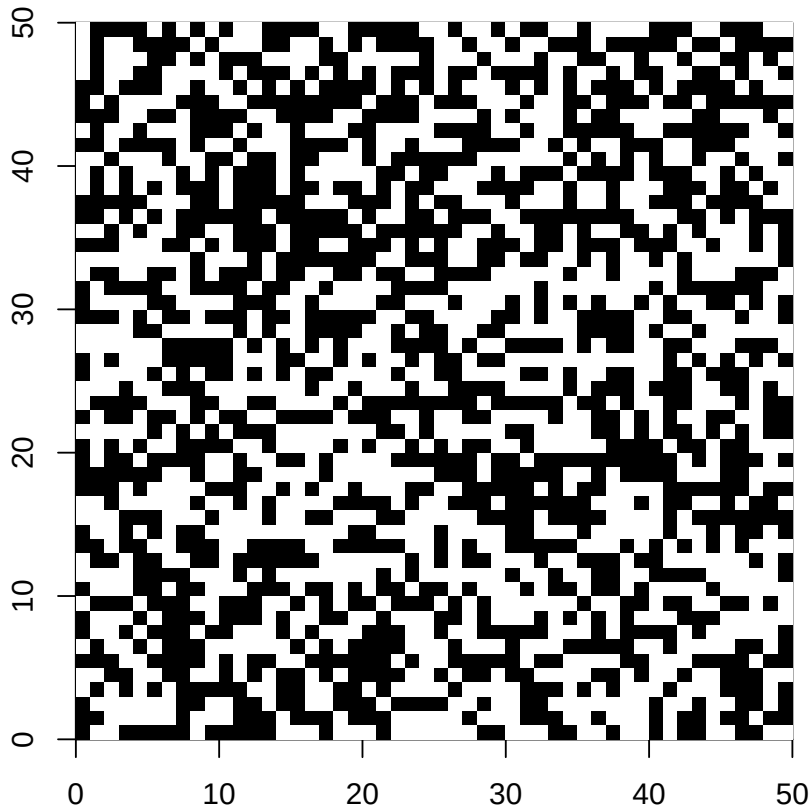


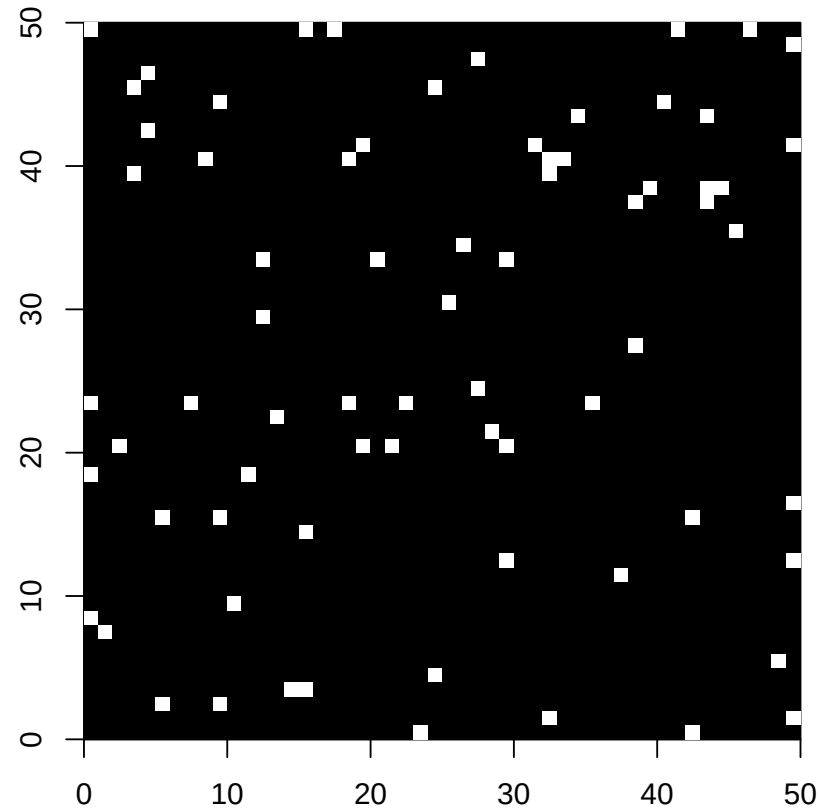
Ising Model: $\beta = 0.4$





Ising Model: $\beta = 0.5$





Example: The sampling distribution of a statistic

Suppose $X_1, \ldots, X_n$ is a random sample from a probability distribution with cdf F_X. Let $T = T(X_1, \ldots, X_n)$ be a statistic used to

- estimate some functional μ_{F_X} of F_X,
- test a hypothesis concerning F_X

The **sampling distribution** of T (perhaps under restrictions imposed by the hypothesis) is required:

- this is merely the probability distribution of T
- in a frequentist sense, this is the distribution of observed values of T over repeated samples of size n in presumed data collection situation.

Example: The sampling distribution of a statistic

If the form of F_X, and the value of the parameters in a parametric representation of F_X are known, we may examine the sampling distribution using **Monte Carlo**

1. Draw a sample $x_1, \ldots, x_n$ from F_X.
2. Compute and store $t = T(x_1, \ldots, x_n)$.
3. Repeat 1. and 2. a large number (N) of times, collecting the observed values
$$t_1, \ldots, t_N$$
4. Approximate f_T by a histogram estimator based on $t_1, \ldots, t_N$.

Example: The sampling distribution of a statistic

If the form of F_X is unknown, we may examine the sampling distribution using the **bootstrap**

1. **Resample** $x_1^{(b)}, \ldots, x_n^{(b)}$ from the original collection $x_1, \ldots, x_n$
   - For $i = 1, \ldots, n$, pick $x_i^{(b)}$ uniformly from $x_1, \ldots, x_n$.
   - Repeat this independently and with replacement.
   - This resampled collection will be similar to, but potentially slightly different from, the original collection; some original values may be repeated in, or omitted from, the resampled collection.
2. Compute the statistic $t^{(b)} = T(x_1^{(b)}, \ldots, x_n^{(b)})$
3. Repeat 1. and 2. a large number (B) of times, collecting the observed values $t_1^{(b)}, \ldots, t_B^{(b)}$.
4. Approximate f_T by a histogram estimator based on $t_1^{(b)}, \ldots, t_B^{(b)}$.

Example: The sampling distribution of a statistic

Example (Bootstrap for Gamma distribution median)

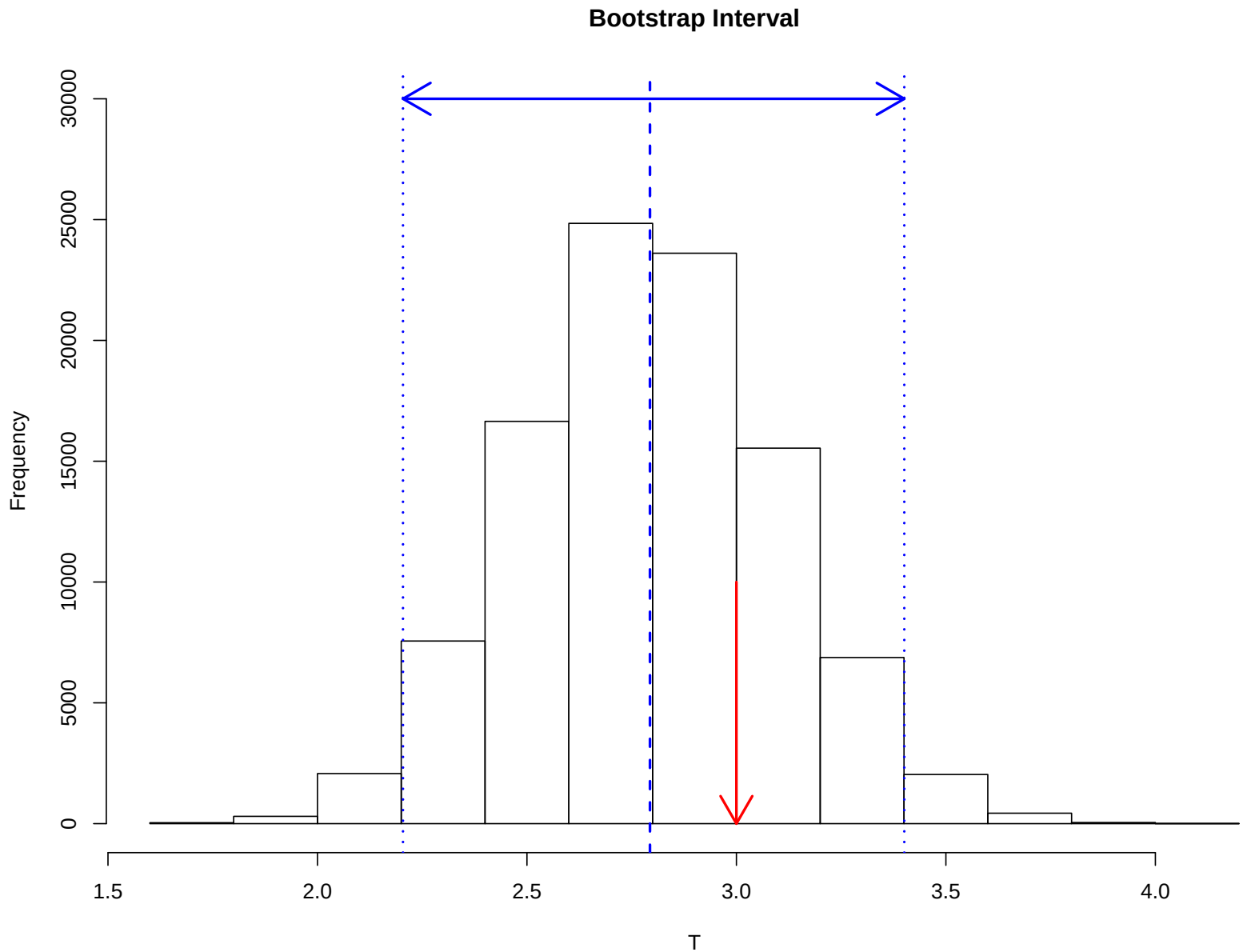
Suppose data

0.931	3.989	2.576	4.084	5.165
1.198	4.859	1.315	2.706	2.223
2.557	2.497	2.768	5.132	4.317
1.499	2.186	1.905	3.371	0.595

are believed to come from a Gamma distribution whose **mean** is the focus of interest. The natural estimator is the sample mean, 2.794.

We can construct a 95% confidence interval using the bootstrap.

Example: Bootstrap interval with $B = 100000$



Example: Bootstrap interval

We can tell if this is a good way to construct such an interval by carrying out a Monte Carlo simulation:

1. For a specific value (or collection of values) for α, generate a sample of size n from $Gamma(\alpha, 1)$.
2. Compute the sample mean.
3. Use the bootstrap to construct a 95% interval for α.
4. Check whether the true value of α lies in the interval.
5. Repeat 1. to 4. a large number of times, and compute the proportion of the repeats in which the interval **does** contain the true value - if the interval construction procedure is good, the proportion should be close to 0.95.

R Code

```
set.seed(423)
n<-20
al<-3
Nrep<-1000
nboot<-10000
tvec<-rep(0,nboot)
ico<-0
for(irep in 1:Nrep){
        X<-rgamma(n,al,1)
        for(i in 1:nboot){
                tvec[i]<-mean(sample(X,size=n,rep=T))
        }
        qv<-quantile(tvec,prob=c(0.025,0.975))
        if(al >= qv[1] & al <= qv[2]){ico<-ico+1}
}
print(ico/Nrep)
```

Example: State-space models

Consider the following model: for $t = 1, \ldots, T$

$$\begin{aligned} y_t &= g(x_t; \theta) + \epsilon_t \\ x_{t+1} &= h(x_t; \theta) + \xi_t \end{aligned}$$

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

This is termed a **state-space model** with state sequence $x_1, \ldots, x_T$. The state space sequence is a focus of interest, along with model parameters θ.

Example: State-space models

Let $\overline{y}_t = (y_1, \ldots, y_t)$, $\overline{x}_t = (x_1, \ldots, x_t)$. For the moment, suppress dependence on θ.

- **Filtering:** Compute $p(x_t|\overline{y}_t)$
- **Prediction:** Compute $p(x_{t+1}|\overline{y}_t)$
- **Smoothing:** Compute $p(x_t|\overline{y}_T)$

The first two steps carry out a sequential calculation **forwards** in time, utilizing basic probability results.

Example: State-space models

Filtering: By Bayes theorem

$$\begin{aligned} p(x_t|\overline{y}_t) &= \frac{p(y_t|x_t)p(x_t|\overline{y}_{t-1})}{p(y_t|\overline{y}_{t-1})} \\ &\propto p(y_t|x_t)p(x_t|\overline{y}_{t-1}) \end{aligned}$$

- The function $p(y_t|x_t)$ is usually a straightforward conditional density.
- The function $p(x_t|\overline{y}_{t-1})$ is the density computed at the previous time point **prediction** step.
- Note that the denominator (normalizing constant) is $p(y_t|\overline{y}_{t-1})$
- Recall that all of these densities potentially depend on θ.

Example: State-space models

Prediction: By marginalization

$$p(x_{t+1}|\overline{y}_t) \quad = \quad \int p(x_{t+1}|x_t)p(x_t|\overline{y}_t)\, dx_t$$

- The function $p(x_{t+1}|x_t)$ is usually a straightforward conditional density derived from the state equation.
- The function $p(x_t|\overline{y}_t)$ is the posterior density computed at the time point **filtering** step.
- This step requires an integration

Example: State-space models

Smoothing: Consider computing **backwards** from $t = T, T-1, \ldots, 1$:

$$\begin{aligned} p(x_t|\overline{y}_T) &= \int p(x_t, x_{t+1}|\overline{y}_T)\, dx_{t+1} \\ &= \int p(x_t|x_{t+1}, \overline{y}_T) p(x_{t+1}|\overline{y}_T)\, dx_{t+1} \\ &= \int p(x_t|x_{t+1}, \overline{y}_t) p(x_{t+1}|\overline{y}_T)\, dx_{t+1} \end{aligned}$$

where

$$p(x_t|x_{t+1}, \overline{y}_t) = \frac{p(x_{t+1}|x_t) p(x_t|\overline{y}_t)}{p(x_{t+1}|\overline{y}_t)}$$

Example: State-space models

- The function $p(x_{t+1}|x_t)$ is usually a straightforward conditional density derived from the state equation.
- The function $p(x_t|\overline{y}_t)$ is the posterior density computed at the relevant *filtering* step in the forward calculation.
- The function $p(x_{t+1}|\overline{y}_t)$ is the the density computed at the relevant *prediction* step in the forward calculation.
- The function $p(x_{t+1}|\overline{y}_T)$ is the density from the previous *smoothing* step in the recursion.
- This step requires an integration.

Example: State-space models

If the error terms ϵ_t and ξ_t are **Gaussian**, and the functions g and h are each **linear** in x_t, then all of the relevant densities

$$p(x_t|\overline{y}_t) \qquad p(x_{t+1}|\overline{y}_t) \qquad p(x_t|\overline{y}_T) \qquad p(x_{t+1}|\overline{y}_T)$$

can be computed analytically, and are Gaussian, so that the distributions are characterized by mean and variance.

If the error terms ϵ_t and ξ_t are **non-Gaussian**, or the functions g and h are **non-linear** in x_t, then the densities can typically only be computed numerically.

A typical approach uses **Sequential Monte Carlo**, in which sample-based approximations to the densities are used.

Example: State-space models

The above calculations presume that θ is known. Note that the likelihood $L(\theta; \overline{y}_T)$ can be expressed

$$L(\theta; \overline{y}_T) = p(y_1; \theta) \prod_{t=1}^{T-1} p(y_{t+1} | \overline{y}_t; \theta)$$

that is, in terms of quantities that are computed using the filtering and prediction steps.

Therefore the filtering approach can be used to facilitate inference about θ.

Part I

Optimization

Optimization in statistical problems

Optimization arises in many statistical problems:

- *minimization* in least-squares, penalized least-squares, model selection etc.
- *maximization* in maximum likelihood estimation, REML, optimal decision making etc.

We will focus on optimization of continuous and differentiable functions of vector parameters. Discrete optimization problems do arise, but are less common.

Optimization in statistical problems

We seek to optimize a scalar function $g(\theta)$ of vector parameter θ

$$\theta = (\theta_1, \ldots, \theta_p)^\intercal.$$

The components of θ may be

- *unconstrained*: all $\theta_j \in \mathbb{R}$
- *constrained*: $\theta_j \in [a_j, b_j]$, or $\varphi(\theta) \in [a, b]$ etc.

Recommendation:

It is usually easier to solve the **unconstrained** version, so where possible try to reparameterize onto $\mathbb{R}^p$.

Some basic facts about optimization

Consider $g(\theta)$ defined on the domain Θ. Usually, g will be continuous and bounded on Θ. Then

- **either** $g(\theta)$ attains its minimium/maximum value at value where

  $$\dot{g}(\theta) = \mathbf{0}$$

  where $\dot{g}(\theta)$ is the $p \times 1$ vector with jth component

  $$\frac{\partial g(\theta)}{\partial \theta_j}$$

- **or** $g(\theta)$ attains its minimium/maximum value on the boundary of Θ.

Some basic facts about optimization

The function $g(\theta)$ is **convex** if, for all θ, the $p \times p$ matrix

$$\ddot{g}(\theta) = \frac{\partial^2 g(\theta)}{\partial\theta\partial\theta^\intercal} = \left[\frac{\partial^2 g(\theta)}{\partial\theta_j\partial\theta_k}\right]_{jk}$$

is positive definite. Another characterization of convexity is that for $\theta_1, \theta_2 \in \Theta$ and $0 \leqslant \lambda \leqslant 1$.

$$g(\lambda\theta_1 + (1-\lambda)\theta_2) \leqslant \lambda g(\theta_1) + (1-\lambda)g(\theta_2)$$

A convex function has a **unique** minimum (if the minimum exists).

Strategies for Numerical Optimization

Basic components of a numerical optimization procedure:

- *approximation*: approximate a complicated function via a simpler and more tractable one.
- *recursion*: refine an initial guess at the optimizing parameter by using repeated adjustment.
- *stopping*: decide when the refinement process can stop.

We focus on estimation of parameter θ via a parametric log likelihood $\ell(\theta)$.

Newton's Method

A second-order Taylor expansion of the log-likelihood around a "current" (at step n) point, $\theta = \theta_n$ yields

$$\ell(\theta) \doteq \ell(\theta_n) + \dot{\ell}(\theta_n)(\theta - \theta_n) + \frac{1}{2}(\theta - \theta_n)^\intercal \ddot{\ell}(\theta_n)(\theta - \theta_n)$$

Using this approximation, consider trying to maximize the right-hand side; we would differentiate wrt θ, and equate to zero. Therefore, to generate the "next" point

$$\dot{\ell}(\theta) \doteq \dot{\ell}(\theta_n) + \ddot{\ell}(\theta_n)(\theta - \theta_n) = \mathbf{0}$$

can be solved for θ to yield the next point in the recursion, θ_{n+1},

$$\theta_{n+1} = \theta_n - \{\ddot{\ell}(\theta_n)\}^{-1}\dot{\ell}(\theta_n)$$

(recall, $\dot{\ell}$ is $p \times 1$, $\ddot{\ell}$ is $p \times p$).

Newton's Method

The recursion continues until $\|\theta_{n+1} - \theta_n\|$ is small enough.

A stationary point, θ^*, of this algorithm, which satisfies

$$\theta^* = \theta^* - \{\ddot{\ell}(\theta^*)\}^{-1}\dot{\ell}(\theta^*)$$

is a local optimum of ℓ. The algorithm requires computation of

$$\{\ddot{\ell}(\theta^*)\}^{-1}$$

and also may be slow to converge (if it converges at all).

In addition, there is no guarantee that

$$\ell(\theta_{n+1}) > \ell(\theta_n)$$

(i.e. it is not an *ascent* algorithm).

Newton's Method

Replacing $-\ddot{\ell}(\theta^*)$ by an approximating non-singular matrix H_n say can produce an ascent algorithm. From the previous expansion

$$\triangle\theta_n = \theta_{n+1} - \theta_n = H_n^{-1}\dot{\ell}(\theta_n)$$

and hence, for $s > 0$, by another Taylor expansion

$$\ell(\theta_n + s\triangle\theta_n) - \ell(\theta_n) \;\doteq\; s\dot{\ell}^{\intercal}(\theta_n)H_n^{-1}\dot{\ell}(\theta_n) \geqslant 0$$

as H_n is positive definite, provided s is made small enough.

Trying $s = 1/2, 1/4, \ldots$ usually can produce an increase in ℓ.

Weighted Least-Squares

Example (Weighted Least-Squares)

Consider estimating parameters θ appearing in the possibly non-linear regression model

$$Y_i = \mu_i(\theta) + \epsilon_i \qquad \mathbb{E}[\epsilon_i] = 0$$

by weighted least-squares. We seek to minimize the function

$$g(\theta) = \frac{1}{2}\sum_{i=1}^{n} w_i(y_i - \mu_i(\theta))^2$$

Weighted Least-Squares

Example (Weighted Least-Squares)

Therefore

$$\begin{aligned}
\dot{g}(\theta) &= -\sum_{i=1}^{n} w_i (y_i - \mu_i(\theta)) \dot{\mu}_i(\theta) \\
\ddot{g}(\theta) &= \sum_{i=1}^{n} w_i \dot{\mu}_i(\theta) \dot{\mu}_i(\theta)^{\mathsf{T}} - \sum_{i=1}^{n} w_i (y_i - \mu_i(\theta)) \ddot{\mu}_i(\theta) \\
&\doteq \sum_{i=1}^{n} w_i \dot{\mu}_i(\theta) \dot{\mu}_i(\theta)^{\mathsf{T}}
\end{aligned}$$

if either

(i) $w_i(y_i - \mu_i(\theta)) \approx 0$ or

(ii) $\ddot{\mu}_i(\theta) \approx \mathbf{0}$

Weighted Least-Squares

Example (Weighted Least-Squares)

In a recursion, given an initial fit, it is quite likely that

$$w_i(y_i - \mu_i(\theta)) \approx 0$$

so (i) holds. Alternatively, (ii) is true if the function $\mu(\theta)$ is approximately linear in θ.

Therefore the approximation

$$\ddot{g}(\theta) \doteq H(\theta) = \sum_{i=1}^{n} w_i \dot{\mu}_i(\theta) \dot{\mu}_i(\theta)^\intercal$$

in the algorithm seems like a good one; note that $H(\theta)$ is positive (semi) definite as it is formed from outer products of the $\dot{\mu}_i(\theta)$.

The Gauss-Newton Algorithm

The approximation method above is a form of the **Gauss-Newton** algorithm. It avoids the calculation of second derivatives of g, and does not rely on specification of a step-size s.

This approximation is useful for estimation in GLMs.

Poisson GLM

Example (Poisson GLM)

Suppose $Y_i \sim Poisson(\mu_i(\theta))$ for $i = 1, \ldots, n$.

$$\ell(\theta) = \sum_{i=1}^{n} \{y_i \log \mu_i(\theta) - \mu_i(\theta)\} + \text{ constant}$$

so that

$$\begin{aligned}
\dot{\ell}(\theta) &= \sum_{i=1}^{n} \left\{ \frac{y_i}{\mu_i(\theta)} \dot{\mu}_i(\theta) - \dot{\mu}_i(\theta) \right\} \\
\ddot{\ell}(\theta) &= \sum_{i=1}^{n} \left\{ -\frac{y_i}{\mu_i(\theta)^2} \dot{\mu}_i(\theta) \dot{\mu}_i(\theta)^\intercal + \frac{y_i}{\mu_i(\theta)} \ddot{\mu}_i(\theta) - \ddot{\mu}_i(\theta) \right\}
\end{aligned}$$

Poisson GLM

Example (Poisson GLM)

As $\mathbb{E}[Y_i] = \mu_i(\theta)$, approximating $y_i/\mu_i(\theta) \approx 1$, we have

$$\ddot{\ell}(\theta) \doteq \sum_{i=1}^{n} \left\{ -\frac{y_i}{\mu_i(\theta)^2} \dot{\mu}_i(\theta) \dot{\mu}_i(\theta)^\intercal \right\} \doteq \sum_{i=1}^{n} \left\{ -\frac{1}{\mu_i(\theta)} \dot{\mu}_i(\theta) \dot{\mu}_i(\theta)^\intercal \right\}$$

which yields the required positive definite approximation to $\ddot{\ell}(\theta)$.

Expected Information

The matrix $\ddot{\ell}(\theta)$ yields $-n\widehat{I}(\theta)$, where $\widehat{I}$ is the observed (unit) information. If available, the Fisher or expected information

$$\mathcal{I}(\theta) = \mathbb{E}[S_\theta(\theta)S_\theta(\theta)^\intercal] = \mathbb{V}ar[S_\theta(\theta)]$$

or

$$\mathcal{I}(\theta) = -\mathbb{E}[S_{\theta\theta}(\theta)]$$

can be utilized in the algorithm instead; this matrix is automatically positive (semi) definite.

Expected Information

In the Exponential Family

$$\ell(\theta) = g(\boldsymbol{y}) + \beta(\theta) + \boldsymbol{\gamma}(\theta)^{\mathsf{T}}\boldsymbol{h}(\boldsymbol{y})$$

so therefore

$$\dot{\ell}(\theta) = \dot{\beta}(\theta) + \dot{\boldsymbol{\gamma}}(\theta)^{\mathsf{T}}\boldsymbol{h}(\boldsymbol{y})$$

and as $\mathbb{E}[\dot{\ell}(\theta)] = \mathbb{E}[S_\theta(\theta)] = \mathbf{0}$, if $\mathbb{E}[\boldsymbol{h}(\boldsymbol{y})] = \boldsymbol{\mu}(\theta)$, so therefore

$$\dot{\beta}(\theta) + \dot{\boldsymbol{\gamma}}(\theta)^{\mathsf{T}}\boldsymbol{\mu}(\theta) = \mathbf{0}$$

Expected Information

Hence

$$\dot{\ell}(\theta) = \dot{\boldsymbol{\gamma}}(\theta)^{\mathsf{T}}(\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu}(\theta))$$

so therefore, by the matrix variance result

$$\mathbb{V}ar[\dot{\ell}(\theta)] = \mathbb{V}ar[\dot{\boldsymbol{\gamma}}(\theta)^{\mathsf{T}}(\boldsymbol{h}(\boldsymbol{Y}) - \boldsymbol{\mu}(\theta))] = \dot{\boldsymbol{\gamma}}(\theta)\Sigma(\theta)\dot{\boldsymbol{\gamma}}(\theta)^{\mathsf{T}}$$

where

$$\Sigma(\theta) = \mathbb{V}ar[\boldsymbol{h}(\boldsymbol{Y})].$$

Here the $p \times p$ matrix $\dot{\boldsymbol{\gamma}}(\theta)$ has (j,k)th element

$$\frac{\partial \gamma_j(\theta)}{\partial \theta_k}$$

Expected Information

Now, by further differentiation

$$\dot{\boldsymbol{\mu}}(\theta) = \Sigma(\theta)\dot{\boldsymbol{\gamma}}(\theta)$$

so that, provided $\Sigma(\theta)$ is non-singular,

$$\dot{\boldsymbol{\gamma}}(\theta) = \Sigma(\theta)^{-1}\dot{\boldsymbol{\mu}}(\theta)$$

and hence

$$\dot{\ell}(\theta) = \dot{\boldsymbol{\mu}}(\theta)^{\mathsf{T}}\Sigma(\theta)^{-1}(\boldsymbol{h}(\boldsymbol{y}) - \boldsymbol{\mu}(\theta))$$

so that, finally

$$\mathcal{J}(\theta) = \mathbb{V}ar[\dot{\ell}(\theta)] = \dot{\boldsymbol{\mu}}(\theta)\Sigma(\theta)\dot{\boldsymbol{\mu}}(\theta)^{\mathsf{T}}$$

Expected Information and Scoring

The matrix $\mathcal{J}(\theta)$ can replace the $\ddot{\ell}(\theta)$ term in the Newton's method formula, for example,

$$\theta_{n+1} = \theta_n + \{\mathcal{J}(\theta_n)\}^{-1}\dot{\ell}(\theta_n)$$

etc. This is termed **scoring** or **Fisher scoring**.

Quasi-Newton methods

The approximating matrix H_n can also be updated during the recursion. Using a second order Taylor approximation

$$\dot{\ell}(\theta_{n+1}) - \dot{\ell}(\theta_n) \simeq \ddot{\ell}(\theta_{n+1})(\theta_n - \theta_{n+1})$$

or

$$g_n \simeq \ddot{\ell}(\theta_{n+1}) s_n$$

say. Thus the positive definite approximating matrix H should satisfy

$$g_n = -H s_n$$

that is, for each step n,

$$g_n = -H_{n+1} s_n$$

Quasi-Newton methods

To update H_n to H_{n+1}, use the recursion

$$H_{n+1} = H_n - c_n v_n v_n^\intercal$$

where

$$c_n = [(g_n + H_n s_n)^\intercal s_n]^{-1}$$

and

$$v_n = g_n + H_n s_n.$$

This is termed a **rank-one update** because of the $v_n v_n^\intercal$ term; higher-rank updates exist (for example, the **BFGS** algorithm).

Note: can consider H_n^{-1} instead of H_n.

Iteratively Reweighted Least Squares

For a model using the one parameter exponential-dispersion family, the log-likelihood for data $y_1, \ldots, y_n$ takes the form

$$\ell(\theta, \phi) = \sum_{i=1}^{n} \frac{w_i (y_i \theta + c(\theta))}{\phi} + \sum_{i=1}^{n} d\left(y_i, \frac{\phi}{w_i}\right) + const. \qquad (1)$$

and the expectation and variance of observable Y_i are

$$\begin{aligned} \mathbb{E}[Y_i] &= -\frac{\partial c(\theta)}{\partial \theta} = -\dot{c}(\theta) = \mu \\ \mathbb{V}ar[Y_i] &= -\frac{\phi}{w_i} \frac{\partial^2 c(\theta)}{\partial \theta^2} = -\frac{\phi}{w_i} \ddot{c}(\theta) = \frac{\phi}{w_i} V(\mu) \qquad (2) \end{aligned}$$

say.

Iteratively Reweighted Least Squares

The GLM with link function l has $\mu = \mu_i$, and sets

$$l(\mu_i) = \boldsymbol{x}_i^{\mathsf{T}} \beta = \eta_i \tag{3}$$

say, for a model with parameters $\beta = \left(\beta_1, ..., \beta_p\right)$. With the canonical link, we have merely that

$$l(\mu_i) = \theta_i$$

so that $\theta_i = \boldsymbol{x}_i^{\mathsf{T}} \beta$.

η_i is the *linear predictor*.

Iteratively Reweighted Least Squares

A set of p **score equations** is obtained by differentiation partially with respect to the elements of β and equating to zero. For example, for β_j, by the chain rule

$$\begin{aligned}\frac{\partial}{\partial \beta_j}\left\{\ell\left(\beta,\phi\right)\right\} &= \sum_{i=1}^{n}\frac{w_i}{\phi}\left(y_i\frac{\partial\theta_i}{\partial\beta_j}+\frac{\partial c(\theta_i)}{\partial\beta_j}\right)\\ &= \sum_{i=1}^{n}\frac{w_i}{\phi}\left(y_i+\frac{\partial c(\theta_i)}{\partial\theta_i}\right)\frac{\partial\theta_i}{\partial\beta_j}=0 \qquad (4)\end{aligned}$$

Iteratively Reweighted Least Squares

However, differentiating the LHS of (3), denoting by $\dot{c}$ and $\ddot{c}$ the first and second derivatives of c wrt θ, we have

$$\frac{\partial}{\partial \beta_j}\left\{l\left(-\dot{c}\left(\theta_i\right)\right)\right\} = \dot{l}\left(-\dot{c}\left(\theta_i\right)\right)\left(-\ddot{c}\left(\theta_i\right)\right)\frac{\partial \theta_i}{\partial \beta_j} = \dot{l}\left(\mu_i\right)V\left(\mu_i\right)\frac{\partial \theta_i}{\partial \beta_j}$$

by (2). Hence, differentiating the RHS of (3) and substituting back

$$\frac{\partial \theta_i}{\partial \beta_j} = \frac{1}{\dot{l}\left(\mu_i\right)V\left(\mu_i\right)}\frac{\partial \eta_i}{\partial \beta_j}.$$

Iteratively Reweighted Least Squares

In addition,

$$\frac{\partial c(\theta_i)}{\partial \theta_i} = \dot{c}(\theta_i) = -\mu_i \qquad\qquad \frac{\partial \eta_i}{\partial \beta_j} = x_{ij}$$

so that, finally, (4) becomes

$$\sum_{i=1}^{n} \frac{w_i}{\phi} \frac{(y_i - \mu_i)\, x_{ij}}{\dot{l}(\mu_i)\, V(\mu_i)} = 0. \tag{5}$$

Removing the dispersion constant ϕ and re-arranging gives the set of score equations

$$\sum_{i=1}^{n} \frac{w_i (y_i - \mu_i)}{V(\mu_i)} \frac{x_{ij}}{\dot{l}(\mu_i)} = 0 \qquad\qquad j = 1, ..., p \tag{6}$$

Iteratively Reweighted Least Squares

If the canonical link is used

$$\theta_i = x_i^\intercal \beta \Longrightarrow \frac{\partial \theta_i}{\partial \beta_k} = x_{ij}$$

and the equations simplify to

$$\sum_{i=1}^{n} w_i \left(y_i - \mu_i\right) x_{ij} = 0 \qquad \therefore \qquad \sum_{i=1}^{n} w_i y_i = \sum_{i=1}^{n} w_i \mu_i x_{ij} \tag{7}$$

for $j = 1, ..., p$, to be solved simultaneously for $\beta_1, ..., \beta_p$ which appear in (7) through $\mu_i = l^{-1}\left(x_i^\intercal \beta\right)$.

Iteratively Reweighted Least Squares

The system (6) cannot be solved analytically, but can be solved recursively. Let

$$\begin{aligned} Z_i &= \eta_i + (Y_i - \mu_i)\,\dot{l}\,(\mu_i) \\ &= x_i^{\mathsf{T}}\beta + (Y_i - \mu_i)\,\dot{l}\,(\mu_i) \end{aligned} \tag{8}$$

then as $\mathbb{E}[Y_i] = \mu_i$ the expected value of Z_i is

$$\mathbb{E}[Z_i] = \eta_i + (\mathbb{E}[Y_i] - \mu_i)\,\dot{l}\,(\mu_i) = \eta_i + 0 \times \dot{l}\,(\mu_i) = \eta_i.$$

and the variance of Z_i

$$\mathbb{V}ar[Z_i] = \mathbb{V}ar[Y_i]\left\{\dot{l}\,(\mu_i)\right\}^2 = \frac{\left\{\dot{l}\,(\mu_i)\right\}^2 \phi V(\mu_i)}{w_i}.$$

Iteratively Reweighted Least Squares

Hence, from (8) the $\{Z_i\}$ form a **linear model** in β conditional on $\{\eta_i\}$ and $\{\mu_i\}$, and if observed values $\{z_i\}$ were available, the parameters in $\mu_i = l^{-1}\left(x_i^{\mathsf{T}}\beta\right)$ could be estimated using a weighted least squares procedure, that is, by $\widehat{\beta}$ where,

$$\widehat{\beta} = \left(\mathbf{X}^{\mathsf{T}}\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{W}z \tag{9}$$

and where, if $x_i = \left(x_{i1}, \ldots x_{ip}\right)^{\mathsf{T}}$ is the row vector of predictors, $\mathbf{X}$ is the $n \times n$ matrix whose n rows are $x_1, \ldots, x_n$, and

$$\mathbf{W} = \operatorname{diag}\left(\frac{w_1}{\left\{\dot{l}\left(\mu_1\right)\right\}^2 V\left(\mu_1\right)}, \ldots, \frac{w_n}{\left\{\dot{l}\left(\mu_n\right)\right\}^2 V\left(\mu_n\right)}\right)$$

Iteratively Reweighted Least Squares

The recursive procedure for finding the estimates of β in the GLM then proceeds as follows:

1. Choose any values for $\widehat{\mu}_i^{(0)}$ and $\widehat{\eta}_i^{(0)} = l\left(\widehat{\mu}_i^{(0)}\right)$ for $i = 1, ..., n$
2. For $i = 1, ..., n$ set

$$\widehat{z}_i^{(1)} = \widehat{\eta}_i^{(0)} + \left(y_i - \widehat{\mu}_i^{(0)}\right) \dot{l}\left(\widehat{\mu}_i^{(0)}\right)$$

and

$$\widehat{w}_i^{(1)} = \frac{w_i}{\left\{\dot{l}\left(\widehat{\mu}_i^{(0)}\right)\right\}^2 V\left(\widehat{\mu}_i^{(0)}\right)} \quad \Longrightarrow \quad \mathbf{W}^{(1)} = \text{diag}\left(\widehat{w}_1^{(1)}, ..., \widehat{w}_n^{(1)}\right)$$

3. Compute estimates

$$\hat{\beta}^{(1)} = \left(\mathbf{X}^\intercal \mathbf{W}^{(1)} \mathbf{X}\right)^{-1} \mathbf{X}^\intercal \mathbf{W}^{(1)} \widehat{z}^{(1)}$$

Iteratively Reweighted Least Squares

4. Define vectors

$$\widehat{\eta}^{(1)} = \mathbf{X}\hat{\beta}^{(1)} \qquad\qquad \widehat{\mu}^{(1)} = l^{-1}\left(\widehat{\eta}^{(1)}\right)$$

5. Iterate through 1. to 4. using $\widehat{\mu}^{(1)}$ and $\widehat{\eta}^{(1)}$ as the updated starting values, eventually to obtain $\widehat{\mu}^{(2)}$ and $\widehat{\eta}^{(2)}$
6. Repeat 1. to 5. until $\widehat{\mu}^{(t)}$ and $\widehat{\eta}^{(t)}$ satisfy

$$\left|\widehat{\mu}^{(t)} - \widehat{\mu}^{(t-1)}\right| < \epsilon_\mu \qquad\qquad \left|\widehat{\eta}^{(t)} - \widehat{\eta}^{(t-1)}\right| < \epsilon_\eta$$

   for tolerances ϵ_μ and ϵ_η.

Iteratively Reweighted Least Squares

This procedure is known as **Iteratively Reweighted Least Squares**. It can be shown that the resulting estimator, $\widetilde{\beta}$, has the correct asymptotic behaviour, that is, as $n \to \infty$,

$$\sqrt{n}(\widetilde{\beta} - \beta) \xrightarrow{d} \mathcal{N}\left(0, \mathcal{I}(\beta)^{-1}\right)$$

where the asymptotic variance may be estimated by the inverse of $\widehat{I}_n$

$$\widehat{I}_n = \frac{1}{\widehat{\phi}} \mathbf{X}^\intercal \mathbf{W} \mathbf{X} \qquad \text{where} \qquad [\mathbf{W}]_{ii} = \frac{w_i}{V(\widetilde{\mu}_i)\left\{\dot{l}(\widetilde{\mu}_i)\right\}^2}$$

and $\widetilde{\mu}_i$ is the estimated μ_i formed by plugging in $\widetilde{\beta}$ to the linear predictor.

Iteratively Reweighted Least Squares

Equation (8) is justified by noting that, by a first order Taylor approximation, expanding around $\mu_i = \mathbb{E}[Y_i]$,

$$\begin{aligned} l(Y_i) &\doteq l(\mu_i) + (Y_i - \mu_i)\dot{l}(\mu_i) \\ &= \eta_i + (Y_i - \mu_i)\dot{l}(\mu_i) \end{aligned}$$

This indicates that the IRLS procedure may work poorly in finite samples when this first order approximation is not adequate.

Conjugate Gradient Methods

A typical minimization problem relates to the quadratic form

$$g(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\intercal A\boldsymbol{x} - \boldsymbol{b}^\intercal \boldsymbol{x} + c$$

If A is positive definite and symmetric, minimizing this function is equivalent to solving the system

$$A\boldsymbol{x} = \boldsymbol{b}$$

as

$$\dot{g}(\boldsymbol{x}) = A\boldsymbol{x} - \boldsymbol{b}$$

This shows the equivalence of certain **optimization** and **root-finding** tasks.

Conjugate Gradient Methods

Steepest descent methods are recursive procedures that utilize the first derivative (**gradient**) to indicate how the next point in the recursion should be generated.

In the quadratic form problem, the direction of steepest descent is

$$-\dot{g}(\boldsymbol{x}) = \boldsymbol{b} - A\boldsymbol{x}$$

At step n, define the *residual*, $\boldsymbol{r}_n$, by

$$\boldsymbol{r}_n = \boldsymbol{b} - A\boldsymbol{x}_n.$$

Denote the true minimizing solution by $\boldsymbol{x}_*$, and the error at step n, $\boldsymbol{e}_n$, by

$$\boldsymbol{e}_n = \boldsymbol{x}_n - \boldsymbol{x}_*.$$

Then $\boldsymbol{r}_n = -A\boldsymbol{e}_n$.

Conjugate Gradient Methods

We need to move in the direction $-\dot{g}(\boldsymbol{x})$ - but how far ? Starting at $\boldsymbol{x}_0$, consider the line

$$\boldsymbol{x} = \boldsymbol{x}_0 + \alpha \boldsymbol{r}_0 = \boldsymbol{x}_0 + \alpha(\boldsymbol{b} - A\boldsymbol{x}_0)$$

Pick α to minimize the function along this line:

$$\alpha_0 = \arg\min_{\alpha} g(\boldsymbol{x}_0 + \alpha \boldsymbol{r}_0)$$

to generate the next point

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \alpha_0 \boldsymbol{r}_0 = \boldsymbol{x}_0 + \alpha_0(\boldsymbol{b} - A\boldsymbol{x}_0)$$

Minimizing wrt α: Note that

$$\frac{dg(\boldsymbol{x}_1)}{d\alpha} = \dot{g}(\boldsymbol{x}_1)^\mathsf{T} \frac{d\boldsymbol{x}_1}{d\alpha} = \dot{g}(\boldsymbol{x}_1)^\mathsf{T} \boldsymbol{r}_0$$

and so the derivative is zero if $\dot{g}(\boldsymbol{x}_1)$ and $\boldsymbol{r}_0$ are orthogonal.

Now, from above,

$$\dot{g}(\boldsymbol{x}_1) = -\boldsymbol{r}_1$$

so therefore at the minimizing α

$$\boldsymbol{r}_1^\mathsf{T} \boldsymbol{r}_0 = \boldsymbol{0}.$$

Conjugate Gradient Methods

Therefore

$$(\boldsymbol{b} - A\boldsymbol{x}_1)^\intercal \boldsymbol{r}_0 = (\boldsymbol{b} - A(\boldsymbol{x}_0 + \alpha \boldsymbol{r}_0))^\intercal \boldsymbol{r}_0 = \boldsymbol{0}$$

Rearranging, this means choosing

$$\alpha_0 = \frac{\boldsymbol{r}_0^\intercal \boldsymbol{r}_0}{\boldsymbol{r}_0^\intercal A \boldsymbol{r}_0}$$

Thus the recursion involves repeatedly computing

- $\boldsymbol{r}_n = \boldsymbol{b} - A\boldsymbol{x}_n$,
- $\alpha_n = \dfrac{\boldsymbol{r}_n^\intercal \boldsymbol{r}_n}{\boldsymbol{r}_n^\intercal A \boldsymbol{r}_n}$,
- $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{r}_n$.

Note that

$$\boldsymbol{r}_{n+1} = (\boldsymbol{I} - \alpha_n A)\boldsymbol{r}_n$$

Conjugate Gradient Methods

A modification to the steepest descent method is as follows: consider a set of **orthogonal** search directions

$$\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{(p-1)}$$

with step sizes $\alpha_0, \ldots, \alpha_{(p-1)}$ chosen to end up precisely at $\boldsymbol{x}_*$ from a given starting point.

In this strategy, we have

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{d}_n$$

Following through in the style of the previous calculations, this implies we need $\boldsymbol{d}_n^\mathsf{T} \boldsymbol{e}_{n+1} = 0$, or

$$\alpha_n = -\frac{\boldsymbol{d}_n^\mathsf{T} \boldsymbol{e}_n}{\boldsymbol{d}_n^\mathsf{T} \boldsymbol{d}_n}$$

Conjugate Gradient Methods

Of course, we cannot compute $\boldsymbol{e}_n = \boldsymbol{x}_n - \boldsymbol{x}_*$. Instead, use the requirement that

$$\boldsymbol{d}_n^\mathsf{T} A \boldsymbol{d}_m = \boldsymbol{0} \qquad \forall m, n$$

that is, the directions are *conjugate* (or *A-orthogonal*). With the new (minimization) requirement that

$$\frac{dg(\boldsymbol{x}_{(n+1)})}{d\alpha} = 0 \qquad \Longrightarrow \qquad \boldsymbol{d}_n^\mathsf{T} A \boldsymbol{e}_{n+1} = 0$$

this means we should set

$$\alpha_n = -\frac{\boldsymbol{d}_n^\mathsf{T} A \boldsymbol{e}_n}{\boldsymbol{d}_n^\mathsf{T} A \boldsymbol{d}_n}$$

which may be calculated, and the algorithm can proceed.

Conjugate Gradient Methods

A Gram-Schmidt type procedure can be used to construct the search direction vectors.

1. Take p linearly independent vectors $\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_p$.
2. Set $\boldsymbol{d}_0 = \boldsymbol{u}_0$.
3. For $n = 1, 2, \ldots$, set

$$\boldsymbol{d}_n = \boldsymbol{u}_n - \sum_{l=1}^{n-1} \frac{\boldsymbol{u}_n^\intercal A \boldsymbol{d}_l}{\boldsymbol{u}_l^\intercal A \boldsymbol{d}_l} \boldsymbol{d}_l$$

A common choice is to set

$$\boldsymbol{u}_n = \boldsymbol{r}_n = \boldsymbol{b} - A\boldsymbol{x}_n$$

that is, based on the successive residuals. This renders the computation more straightforward as Step 3. simplifies to

$$\boldsymbol{d}_n = \boldsymbol{r}_n + \frac{\boldsymbol{r}_n^\intercal \boldsymbol{r}_n}{\boldsymbol{r}_{(n-1)}^\intercal \boldsymbol{r}_{(n-1)}} \boldsymbol{d}_{(n-1)}$$

Conjugate Gradient Methods

Thus the recursion becomes

1. Set $\boldsymbol{d}_0 = \boldsymbol{r}_0 = b - A\boldsymbol{x}_0$.
2. For $n = 1, 2, \ldots,$ set

$$\begin{aligned}
\alpha_n &= \frac{\boldsymbol{r}_n^\mathsf{T}\boldsymbol{r}_n}{\boldsymbol{d}_n^\mathsf{T} A \boldsymbol{d}_n} \\
\boldsymbol{x}_{n+1} &= \boldsymbol{x}_n + \alpha_n \boldsymbol{d}_n \\
\boldsymbol{r}_{n+1} &= \boldsymbol{r}_n - \alpha_n A \boldsymbol{d}_n \\
\boldsymbol{d}_{n+1} &= \boldsymbol{r}_{n+1} + \frac{\boldsymbol{r}_{n+1}^\mathsf{T}\boldsymbol{r}_{n+1}}{\boldsymbol{r}_n^\mathsf{T}\boldsymbol{r}_n}\boldsymbol{d}_n
\end{aligned}$$

Non-linear Conjugate Gradient

To minimize a general function g of θ, we proceed in a similar fashion:

1. Pick a starting value θ_0, and choose α_0 to minimize the function
$$g(\theta_0 - \alpha \dot{g}(\theta_0))$$
over α using a grid-based search. Define
$$\theta_1 = \theta_0 - \alpha_0 \dot{g}(\theta_0).$$
and $d(x_0) = -\dot{g}(\theta_0)$.
2. For the $n+1$st step, starting at θ_n: compute β_n using a standard formula dependent on $\dot{g}(\theta_n)$ and $\dot{g}(\theta_{n-1})$.
3. Set
$$d(\theta_n) = -\dot{g}(\theta_n) + \beta_n d(\theta_{n-1})$$

Non-linear Conjugate Gradient

4. Choose α_n to minimize

$$g(\theta_n + \alpha d(\theta_n))$$

over α.

5. Set

$$\theta_{n+1} = \theta_n + \alpha_n d(\theta_n).$$

and return to 2. Repeat until convergence.

Non-linear Conjugate Gradient

The logic behind this algorithm is that $-\dot{g}(\theta_n)$ gives a good indication of which direction to move in from θ_n in order to achieve minimization.

A typical choice for β_n is

$$\beta_n = \frac{\dot{g}(\theta_n)^\intercal \dot{g}(\theta_n)}{\dot{g}(\theta_{n-1})^\intercal \dot{g}(\theta_{n-1})}$$

Optimization with constraints

Optimization problems of the form

$$\text{minimize } g(\theta) \quad \text{subject to} \qquad \varphi_k(\theta) \leqslant c_k, k = 1, \ldots, K$$

are not straightforward to solve if the constraints, specified by $(\varphi_k, c_k), k = 1, \ldots, K$ are not simple interval constraints.

Lagrange multipliers can be used to reformulate the problem. Define the function $g^*(\theta, \lambda)$

$$g^*(\theta, \lambda) = g(\theta) + \sum_{k=1}^{K} \lambda_k(\varphi_k(\theta) - c_k),$$

differentiate with respect to (θ, λ), and find the zeros of $\dot{g}^*(\theta, \lambda) = 0$. The modification to the usual methods is that the solutions for λ should occur at saddlepoints, not minima.

Nelder-Mead Simplex algorithm

The Nelder-Mead algorithm is another form of recursive optimization procedure. It is **derivative-free** so is simpler to implement, although can be slower to converge.

The basic algorithm proceeds as follows: to minimize a function g in p dimensions

1. Pick $p+1$ points $x_1, \ldots, x_{p+1}$ indexed such that

$$g(x_1) \leqslant g(x_2) \leqslant \cdots \leqslant g(x_p) \leqslant g(x_{p+1})$$

   and compute the centre of mass $\overline{x}_p$ of $x_1, \ldots, x_p$
2. For a suitably chosen $\alpha \geqslant 1$, compute new point, x^*,

$$x^* = \overline{x}_p + \alpha(\overline{x}_p - x_{p+1})$$

Nelder-Mead Simplex algorithm

3. (i) If $g(x_1) \leqslant g(x^*) \leqslant g(x_p)$, replace x_{p+1} by x^*.
   (ii) If $g(x^*) < g(x_1)$, for suitably chosen $\beta \geqslant 1$, compute new point, $x^\dagger$,
   $$x^\dagger = \overline{x}_p + \beta(\overline{x}_p - x_{p+1})$$
   If $g(x^\dagger) < g(x^*)$, replace x_{p+1} by $x^\dagger$ and go to 1., else replace x_{p+1} by x^* and go to 1.
4. For a suitably chosen $\gamma < 1$, compute new point, $x^\ddagger$,
   $$x^\ddagger = x_{p+1} + \gamma(\overline{x}_p - x_{p+1})$$
   If $g(x^\ddagger) < g(x_{p+1})$, replace x_{p+1} by $x^\ddagger$, and go to 1.
5. For $i = 2, \ldots, n+1$, for a suitably chosen $\delta < 1$, replace x_i by
   $$x_1 + \delta(x_i - x_1)$$
   and go to 1.

Nelder-Mead Simplex algorithm

Steps 1-5 are repeated until the function value $f(x_1)$ ceases to change over time. The algorithm is somewhat *ad hoc*

- Need to specify $\alpha, \beta, \gamma, \delta$
- No proof of convergence to the global minimum

and can be slow to reach a stationary value.

However, it is remarkable effective.

R Code for optimization

All of these methods are available via the R function `optim`:

```
optim(par, fn, gr = NULL,  method = c("Nelder-Mead",
    "BFGS", "CG", "L-BFGS-B", "SANN"),
    lower = -Inf, upper = Inf,
    control = list(), hessian = FALSE, ...)
```

`par` provides the parameters, `fn` is the objective function, defined as a function elsewhere, `gr` is the gradient (first derivative) function vector. Arguments to these two functions can be passed via the `...` arguments. `lower` and `upper` specify the parameter constraints.

`hessian` is a logical specifying whether the Hessian (second derivative) matrix evaluated at the minimum value should be returned.

R Code for optimization

`control` is a list that controls the parameters of the chosen algorithm. Key settings are

- `fnscale` sets the scale of the objective function, useful in diagnosing convergence; setting this negative turns the (default) minimization into a maximization
- `maxit` sets the maximum number of iterations
- tolerance parameters `abstol` and `reltol`

Example (Pharmacokinetics Model)

A model typically used in *pharmacokinetics* (the study of how drugs are processed through the body through time), where given a dose D at time $x = 0$, the amount of drug left in the body at time x is

$$\mu(x; \theta_1, \theta_2, \theta_3) = D\theta_1(\exp\{-\theta_2 x\} - \exp\{-(\theta_2 + \theta_3)x\})$$

where $\theta_1, \theta_2, \theta_3 > 0$.

The parameters can be estimated by measuring concentration through time.

Non-Linear Least Squares

Example (Pharmacokinetics Model)

Let $\phi_j = \log \theta_j$. Least squares estimation of finds the minimum of the function

$$g(\phi_1, \phi_2, \phi_3) = \sum_{i=1}^{n} (y_i - \mu(x_i; e^{\phi_1}, e^{\phi_3}, e^{\phi_3}))^2$$

with respect to ϕ_1, ϕ_2, ϕ_3.

We have

$$\dot{g}(\phi) = 2 \sum_{i=1}^{m} (y_i - \mu(x_i; e^{\phi_1}, e^{\phi_3}, e^{\phi_3}))\dot{\mu}(x_i; e^{\phi_1}, e^{\phi_3}, e^{\phi_3})$$

where the computation of $\dot{\mu}_i$ involves partial derivatives with respect to ϕ_1, ϕ_2, ϕ_3.

Example (Pharmacokinetics Model)

For example

$$\mu_{\phi_1}(x;\phi) = De^{\phi_1}(\exp\{-e^{\phi_2}x\} - \exp\{-(e^{\phi_2}+e^{\phi_3})\}) = \mu(x;\phi)$$

and

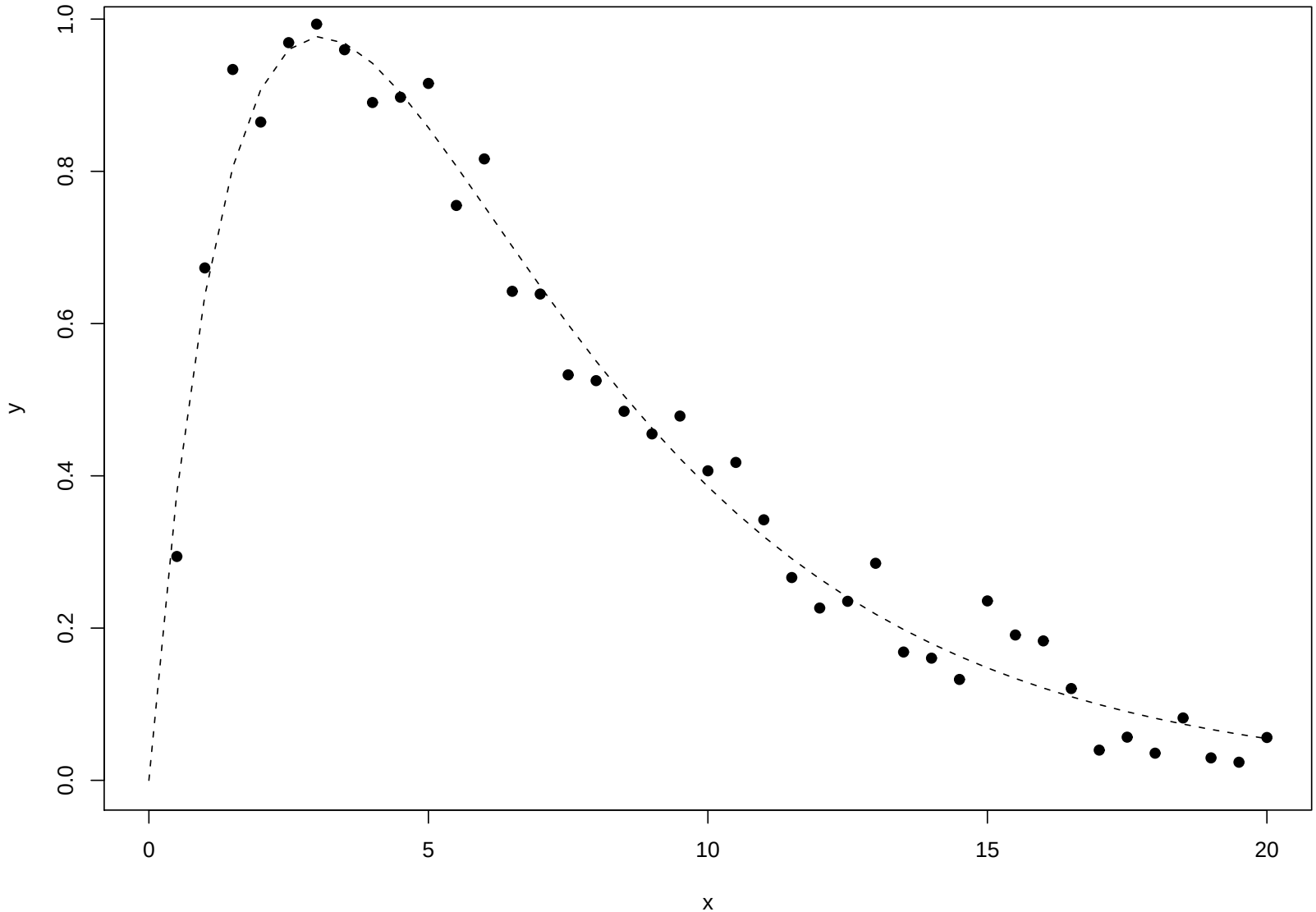
$$\begin{aligned}\mu_{\phi_2}(x;\phi) &= -De^{\phi_1+\phi_2}x(\exp\{-e^{\phi_2}x\} - \exp\{-(e^{\phi_2}+e^{\phi_3})x\}) \\ &= -xe^{\phi_2}\mu(x;\phi)\end{aligned}$$

and

$$\mu_{\phi_3}(x;\phi) = De^{\phi_1+\phi_3}\exp\{-(e^{\phi_2}+e^{\phi_3})x\}.$$

Second derivatives are also available in a straightforward fashion.

Pharmacokinetics data



R Code for Non-Linear LS

```
1  mu.funct<-function(thv,xv){
2
3          aval<-exp(thv[1])
4          lam1<-exp(thv[2])
5          lam2<-exp(thv[3])
6          fvec<-aval*(exp(-lam1*xv)-exp(-(lam1+lam2)*xv))
7          return(fvec)
8  }
9
10 g.funct<-function(thv,xv,yv){
11
12         rv<-yv-mu.funct(thv,xv)
13         return(sum(rv*rv))
14 }
15
16 fit.result<-optim(c(0,0,0),fn=g.funct,xv=X,yv=Y)
```

if the data are in vectors `X` and `Y`.

Using `control=list(trace=1)`

Example

```
  Nelder-Mead direct search function minimizer
function value for initial parameters = 10.485159
  Scaled convergence tolerance is 1.56241e-07
  Stepsize computed as 0.100000

BUILD              4 10.683248 10.379035
EXTENSION          6 10.4851599.809242
LO-REDUCTION       8 10.450614 9.809242
EXTENSION         10 10.379035 8.949165
EXTENSION         12 9.943463 7.170710
```

Using `control=list(trace=1)`

Example

```
LO-REDUCTION     184 0.088658 0.088656
HI-REDUCTION     186 0.088658 0.088656
HI-REDUCTION     188 0.088657 0.088656
HI-REDUCTION     190 0.088656 0.088656
HI-REDUCTION     192 0.088656 0.088656
HI-REDUCTION     194 0.088656 0.088656
Exiting from Nelder Mead minimizer
    196 function evaluations used
```

The EM Algorithm

The EM Algorithm is a method for producing the maximum likelihood estimates in **incomplete data** problems, that is, models formulated for data that are only partially observed.

Suppose that random variables to be modelled can be partitioned $(\boldsymbol{X}, \boldsymbol{Y})$ where

- $\boldsymbol{X} = (X_1, \ldots, X_m)^\intercal$ are **unobserved**, termed the **latent** or **missing** data.
- $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\intercal$ are **observed**, termed the **incomplete data**
- $(\boldsymbol{X}, \boldsymbol{Y})$ are termed the **complete data**

where

$$f_{\boldsymbol{Y}|\theta}(\boldsymbol{y}|\theta) = \int f_{\boldsymbol{X},\boldsymbol{Y}|\theta}(\boldsymbol{x}, \boldsymbol{y}|\theta)\, d\boldsymbol{x}$$

The EM Algorithm: Terminology

$$L_Y(\theta; \boldsymbol{y}) = f_{\boldsymbol{Y}|\theta}(\boldsymbol{y}|\theta)$$

is termed the **incomplete data** likelihood, and

$$L_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) = f_{\boldsymbol{X},\boldsymbol{Y}|\theta}(\boldsymbol{x}, \boldsymbol{y}|\theta)$$

is termed the **complete data** likelihood.

The EM Algorithm facilitates maximization of the incomplete data likelihood $L_Y(\theta; \boldsymbol{y})$ by working with the complete data likelihood $L_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y})$ and the conditional distribution

$$f_{\boldsymbol{X}|\boldsymbol{Y},\theta}(\boldsymbol{x}|\boldsymbol{y}, \theta) = \frac{f_{\boldsymbol{X},\boldsymbol{Y}|\theta}(\boldsymbol{x}, \boldsymbol{y}|\theta)}{f_{\boldsymbol{Y}|\theta}(\boldsymbol{y}|\theta)} = \frac{L_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y})}{L_Y(\theta; \boldsymbol{y})} = K(\theta; \boldsymbol{x}, \boldsymbol{y}) \quad (10)$$

$$\therefore \quad \ell_Y(\theta; \boldsymbol{y}) = \ell_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) - k(\theta; \boldsymbol{x}, \boldsymbol{y}) \quad (11)$$

say.

The EM Algorithm

Consider replacing the right-hand side of equation (11) by the expectations with respect to the conditional density $f_{\boldsymbol{X}|\boldsymbol{Y},\theta}(\boldsymbol{x}|\boldsymbol{y},\theta')$, for some $\theta' \in \Theta$. This yields

$$\begin{aligned}\ell_Y(\theta;\boldsymbol{y}) &= \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}[\ell_{X,Y}(\theta;\boldsymbol{X},\boldsymbol{Y})|\boldsymbol{y},\theta'] - \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}[\log K(\theta;\boldsymbol{X},\boldsymbol{Y})|\boldsymbol{y},\theta'] \\ &= \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}[\ell_{X,Y}(\theta;\boldsymbol{X},\boldsymbol{Y})|\boldsymbol{y},\theta'] - \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}[k(\theta;\boldsymbol{X},\boldsymbol{Y})|\boldsymbol{y},\theta'] \quad (12)\end{aligned}$$

that is, we condition on a specific (but as yet unspecified) value of θ' when computing the expectations of $\ell_{X,Y}(\theta;\boldsymbol{X},\boldsymbol{y})$ and $k(\theta;\boldsymbol{X},\boldsymbol{Y})$ at the θ at which the likelihood on the left-hand side of equation (12) is being computed.

The EM Algorithm

The EM algorithm is an iterative algorithm that produces a sequence of estimates that converges to the (incomplete data) maximum likelihood estimate. Generically, starting from an initial value $\theta = \theta^{(0)}$, the $(r+1)$st value in the sequence, $\theta^{(r+1)}$, is constructed given the rth value , $\theta^{(r)}$,

$$\theta^{(r+1)} = \underset{\theta \in \Theta}{\operatorname{argmax}}\, \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}[\ell_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}, \theta^{(r)}] = \underset{\theta \in \Theta}{\operatorname{argmax}}\, Q(\theta; \theta^{(r)})$$

say. The two components of this calculation are

- **E-step** : compute the expected conditional log-likelihood
- **M-step** : carry out the maximization of the expectation.

We wish to show that the sequence of estimates $\{\theta^{(r+1)}\}$ converges to the maximum likelihood estimate.

The EM Algorithm

For two pdfs f_1 and f_2 for random variable Z, we have by the properties of the *Kullback-Leibler divergence* that

$$\begin{aligned}
\mathbb{E}_{f_1}[\log f_1(Z)] - \mathbb{E}_{f_1}[\log f_2(Z)] &= -\mathbb{E}_{f_1}[\log\{f_2(Z)/f_1(Z)\}] \\
&\geqslant -\log \mathbb{E}_{f_1}[\{f_2(Z)/f_1(Z)\}] \\
&= -\log \int_{\mathcal{Z}} \{f_2(z)/f_1(z)\} f_1(z)\, dz \\
&= -\log \int_{\mathcal{Z}} f_2(z)\, dz = 0
\end{aligned}$$

so therefore $\mathbb{E}_{f_1}[\log f_1(Z)] \geqslant \mathbb{E}_{f_1}[\log f_2(Z)]$, with equality if and only if $f_1 \equiv f_2$.

The EM Algorithm

For $\theta \in \Theta$, as

$$K(\theta; \boldsymbol{X}, \boldsymbol{Y}) = \frac{L_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})}{L_Y(\theta; \boldsymbol{Y})} = f_{\boldsymbol{X}|\boldsymbol{Y},\theta}(\boldsymbol{x}|\boldsymbol{y}, \theta)$$

is itself a (conditional) pdf for all $\theta \in \Theta$, and hence we have, for any θ,

$$\begin{aligned} Q(\theta; \theta^{(r)}) - \ell_Y(\theta; \boldsymbol{y}) &= \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}[\ell_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}, \theta^{(r)}] - \ell_{X,Y}(\theta; \boldsymbol{y}) \\ &= \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}\left[k(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}, \theta^{(r)}\right] \\ &\leqslant \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}\left[k(\theta^{(r)}; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}, \theta^{(r)}\right] \\ &= Q(\theta^{(r)}|\theta^{(r)}) - \ell_Y(\theta^{(r)}|\boldsymbol{y}). \end{aligned}$$

The EM Algorithm

Thus $\ell_Y(\theta; \boldsymbol{y}) - Q(\theta; \theta^{(r)})$ achieves its **minimum** value when $\theta = \theta^{(r)}$. Now suppose that $\theta^{(r+1)}$ is the value that maximizes $Q(\theta; \theta^{(r)})$ over Θ; we have that

$$\begin{aligned}
\ell_Y(\theta^{(r+1)}|\boldsymbol{y}) &\equiv Q(\theta^{(r+1)}|\theta^{(r)}) + \Big(\ell_Y(\theta^{(r+1)}|\boldsymbol{y}) - Q(\theta^{(r+1)}|\theta^{(r)})\Big) \\
&\geqslant Q(\theta^{(r)}|\theta^{(r)}) + \Big(\ell_Y(\theta^{(r)}|\boldsymbol{y}) - Q(\theta^{(r)}|\theta^{(r)})\Big) \\
&= \ell_Y(\theta^{(r)}|\boldsymbol{y})
\end{aligned}$$

and the likelihood attained is **increasing** with the sequence $\theta^{(0)}, \theta^{(1)}, \theta^{(2)}, \ldots$.

Thus, provided $L_Y(\theta; \boldsymbol{y})$ is bounded, we have that the sequence of $\ell_Y(\theta^{(r)}|\boldsymbol{y})$ converges to a (local) maximum of $L_Y(\theta; \boldsymbol{y})$.

Example: Finite Mixture Model

Suppose that $Y_1 \ldots, Y_n$ are a random sample from the K component finite mixture model for $y \in \mathbb{R}$

$$f_{Y|\theta}(y|\theta) = \sum_{k=1}^{K} \omega_k f_k(y|\theta_k) \qquad \sum_{k=1}^{K} \omega_k = 1$$

for $0 < \omega_k < 1$ where $f_1, \ldots, f_K$ are component densities.

Estimation of $\theta = (\theta_1, \ldots, \theta_K)^\intercal$ from the likelihood is in general difficult. However, consider the latent data $X_1, \ldots, X_n$, where $\mathbb{P}[X_i = k] = \omega_k$ are independent random variables. Then

$$\begin{aligned} L_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) &= \prod_{i=1}^{n} \prod_{k=1}^{K} \{\omega_k f_k(y_i|\theta_k)\}^{\mathbb{1}_{\{k\}}(X_i)} \\ \ell_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) &= \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}_{\{k\}}(X_i) \left(\log \omega_k + \log f_k(y_i|\theta_k)\right). \end{aligned}$$

Example: Finite Mixture Model

The conditional distribution $f_{X|Y,\theta}(x|y,\theta)$ is a discrete distribution on the set $\{1,2,\ldots,K\}$ where for each $i = 1,\ldots,n$

$$\mathbb{P}[X_i = k|\boldsymbol{y},\omega,\theta] = \frac{\omega_k f_k(y_i|\theta_k)}{\sum_{j=1}^{K} \omega_j f_j(y_i|\theta_j)} = \varpi_k(y_i,\theta) \qquad k = 1,\ldots,K$$

where $X_1,\ldots,X_n$ are conditionally independent. Thus

$$\mathbb{E}_{f_{X_i|Y_i,\theta,\omega}}[\mathbb{1}_{\{k\}}(X_i)|y_i,\theta,\omega] = \varpi_k(y_i,\theta)$$

Example: Finite Mixture Model

Hence

$$
\begin{aligned}
Q(\theta, \omega | \theta^{(r)}, \omega^{(r)}) &= \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta,\omega}}[\ell_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y}) | \boldsymbol{y}, \theta^{(r)}, \omega^{(r)}] \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} \varpi_k^{(r)}(y_i, \theta^{(r)}) \left(\log \omega_k + \log f_k(y_i|\theta_k)\right) \\
&= \sum_{k=1}^{K} \left\{ \sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)}) \right\} \log \omega_k \qquad (13) \\
&\quad + \sum_{k=1}^{K} \sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)}) \log f_k(y_i|\theta_k) \qquad (14)
\end{aligned}
$$

Example: Finite Mixture Model

We seek to maximize over (θ, ω) to obtain $(\theta^{(r+1)}, \omega^{(r+1)})$ presuming that the values $\varpi_k^{(r)}(y_i, \theta^{(r)})$ are fixed.

From the form of equations (13) and (14), it is evident that the function is sum of two parts, the first only depending on ω, the second only dependent on θ.

We can therefore maximize the two parts separately to obtain $(\theta^{(r+1)}, \omega^{(r+1)})$.

Example: Finite Mixture Model

Equation (13) is of the form of a multinomial likelihood in ω, therefore, by previous results, it follows that

$$\omega_k^{(r+1)} = \frac{\sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)})}{\sum_{j=1}^{K} \sum_{i=1}^{n} \varpi_j^{(r)}(y_i, \theta^{(r)})} \qquad k = 1, \ldots, K$$

Equation (14) is the sum of K log-likelihoods for the K mixture components which can be maximized separately

$$\theta_k^{(r+1)} = \underset{\theta_k}{\operatorname{argmax}} \sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)}) \log f_k(y_i|\theta_k) \tag{15}$$

Example: Finite Mixture Model

For certain choices of the component densities, this maximization can be carried out analytically. For example, if $f_k(y|\theta_k)$ is the normal density with expectation μ_k and variance σ_k^2, it follows that the new maximizing value equals $\theta_k^{(r+1)} = (\mu_k^{(r+1)}, \sigma_k^{(r+1)})$ where

$$\mu_k^{(r+1)} = \frac{\sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)}) y_i}{\sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)})} \qquad \sigma_k^{(r+1)} = \sqrt{\frac{\sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)})(y_i - \mu_k^{(r+1)})^2}{\sum_{i=1}^{n} \varpi_k^{(r)}(y_i, \theta^{(r)})}}$$

In the normal model the terms in (15) correspond to likelihood components of the form

$$\{f_k(y_i|\theta_k)\}^{\varpi_k^{(r)}} = \left(\frac{1}{2\pi\sigma_k^2}\right)^{\varpi_k^{(r)}/2} \exp\left\{-\frac{\varpi_k^{(r)}}{2\sigma_k^2}(y_i - \mu_k^{(r)})^2\right\}$$

so the terms $\varpi_k^{(r)} \equiv \varpi_k^{(r)}(y_i, \theta^{(r)})$ are acting as weighting factors.

Example: Genetics of Human Blood Groups

In human genetics, the **genotype** at a genomic locus is a pair of **alleles** corresponding to small segments of DNA lying on the two chromosomal strands. The **phenotype** is the physical presentation or trait arising from the genotype. At a certain locus that determines the phenotype of blood group, the relationship between genotype and phenotype is somewhat complex; there are

- three alleles (A, B and O) yielding **six** possible genotypes (ordering is not important)
- only **four** phenotypes (A,B,AB and O).

Example: Genetics of Human Blood Groups

The relationship between phenotype and genotype in this case is determined by the following table. The third column, headed X, denotes a label for the genotype class. However, only the phenotype may be observed; let $Y_1, \ldots, Y_n$ denote the recorded phenotype for each of the n data.

Genotype	Phenotype	X	Y
AA	A	1	1
AB	AB	2	3
AO	A	3	1
BB	B	4	2
BO	B	5	2
OO	O	6	4

Example: Genetics of Human Blood Groups

Suppose that inference about the proportions of the three alleles A,B and O, denoted $\theta_A, \theta_B, \theta_O$ is required from a sample of size n of phenotype data.

One independence assumption (based on so-called *Hardy-Weinberg equilibrium*) is needed; we assume that the probability of observing a genotype is the product of the individual allele probabilities. For example

$$\mathrm{P(AA)} = \theta_A \times \theta_A \qquad\qquad \mathrm{P(AB)} = \theta_A \times \theta_B$$

and so on.

Example: Genetics of Human Blood Groups

Define the augmented data $X_1, \ldots, X_m$, where for $i = 1 \ldots, n$,

$$\mathbb{P}[X_i = j] = \mathbb{P}[i\text{th genotype is in class j}] \qquad j = 1, \ldots, 6$$

that is

$$\mathbb{P}[X_i = j] = \begin{cases} \theta_A^2 & j = 1 \\ \theta_A\theta_B & j = 2 \\ \theta_A\theta_O & j = 3 \\ \theta_B^2 & j = 4 \\ \theta_B\theta_O & j = 5 \\ \theta_O^2 & j = 6 \end{cases} \qquad \text{for } i = 1, \ldots, n$$

with $X_1, \ldots, X_m$ a random sample.

Example: Genetics of Human Blood Groups

This simplification yields a complete data likelihood

$$\begin{aligned} L_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) &\equiv L_X(\theta; \boldsymbol{x}) \\ &= \prod_{i=1}^{n} \Big\{ \theta_A^{2\mathbb{1}_{\{1\}}(x_i)+\mathbb{1}_{\{2\}}(x_i)+\mathbb{1}_{\{3\}}(x_i)} \theta_B^{\mathbb{1}_{\{2\}}(x_i)+2\mathbb{1}_{\{4\}}(x_i)+\mathbb{1}_{\{5\}}(x_i)} \\ &\qquad \theta_O^{\mathbb{1}_{\{3\}}(x_i)+\mathbb{1}_{\{5\}}(x_i)+2\mathbb{1}_{\{6\}}(x_i)} \Big\} \\ &= \theta_A^{2n_1+n_2+n_3} \theta_B^{n_2+2n_4+n_5} \theta_O^{n_3+n_5+2n_6} \end{aligned}$$

say, where

$$n_j = \sum_{i=1}^{n} \mathbb{1}_{\{j\}}(x_i) \qquad j = 1, \ldots, 6$$

that is, a multinomial-type likelihood in θ.

Example: Genetics of Human Blood Groups

In the standard notation, for the EM steps, we have to

- E-step: compute

$$Q(\theta; \theta^{(r)}) = \mathbb{E}_{f_{\boldsymbol{X}|\boldsymbol{Y},\theta}}[\ell_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}, \theta^{(r)}]$$

  taking the expectation over $X_1, \ldots, X_m$ etc.

- M-step: maximize $Q(\theta; \theta^{(r)})$ to get $\theta^{(r+1)}$.

Here the M-step is straightforward due to the multinomial likelihood. The E-step is also quite straightforward, but some steps need clarification.

Example: Genetics of Human Blood Groups

The log complete data likelihood takes the form

$$\begin{aligned}\ell_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) &= \sum_{i=1}^{n} \left(2\mathbb{1}_{\{1\}}(x_i) + \mathbb{1}_{\{2\}}(x_i) + \mathbb{1}_{\{3\}}(x_i)\right) \log \theta_A \\ &\quad + \sum_{i=1}^{n} \left(\mathbb{1}_{\{2\}}(x_i) + 2\mathbb{1}_{\{4\}}(x_i) + \mathbb{1}_{\{5\}}(x_i)\right) \log \theta_B \\ &\quad + \sum_{i=1}^{n} \left(\mathbb{1}_{\{3\}}(x_i) + \mathbb{1}_{\{5\}}(x_i) + 2\mathbb{1}_{\{6\}}(x_i)\right) \log \theta_O\end{aligned}$$

which is linear and additive in the indicator functions.

Example: Genetics of Human Blood Groups

Conditional on $\boldsymbol{Y}$ and θ, some expectations can be written down automatically. For example

$$\mathbb{E}_{f_{X_i|Y_i,\theta}}[\mathbb{1}_{\{j\}}(X_i)|Y_i = 3, \theta] = \begin{cases} 1 & j = 2 \\ 0 & j \neq 2 \end{cases}$$

$$\mathbb{E}_{f_{X_i|Y_i,\theta}}[\mathbb{1}_{\{j\}}(X_i)|Y_i = 4, \theta] = \begin{cases} 1 & j = 6 \\ 0 & j \neq 6 \end{cases}$$

as by definition $Y = 3 \Longrightarrow X = 2$ and $Y = 4 \Longrightarrow X = 6$.

Example: Genetics of Human Blood Groups

For the remaining conditional expectations, we have by Bayes theorem

$$\mathbb{E}_{f_{X_i|Y_i,\theta}}[\mathbb{1}_{\{j\}}(X_i)|Y_i = 1, \theta] = \begin{cases} \dfrac{\theta_A^2}{\theta_A^2 + 2\theta_A\theta_O} & j = 1 \\ \dfrac{2\theta_A\theta_O}{\theta_A^2 + 2\theta_A\theta_O} & j = 3 \\ 0 & \text{otherwise} \end{cases}$$

as if $Y = 1$, then either $X = 1$ or $X = 3$, with conditional probability for each determined by noting that

$$\mathbb{P}[X = 1|Y = 1] = \frac{\mathbb{P}[X = 1, Y = 1]}{\mathbb{P}[Y = 1]} = \frac{\mathbb{P}[X = 1, Y = 1]}{\mathbb{P}[X = 1, Y = 1] + \mathbb{P}[X = 3, Y = 1}$$

Example: Genetics of Human Blood Groups

Similarly,

$$\mathbb{E}_{f_{X_i|Y_i,\theta}}[\mathbb{1}_{\{j\}}(X_i)|Y_i = 2, \theta] = \begin{cases} \dfrac{\theta_B^2}{\theta_B^2 + 2\theta_B\theta_O} & j = 4 \\ \dfrac{2\theta_B\theta_O}{\theta_B^2 + 2\theta_B\theta_O} & j = 5 \\ 0 & \text{otherwise} \end{cases}$$

Example: Genetics of Human Blood Groups

Thus $Q(\theta;\theta^{(r)})$ takes the form

$$Q(\theta;\theta^{(r)}) = \alpha_A^{(r)} \log \theta_A + \alpha_B^{(r)} \log \theta_B + \alpha_O^{(r)} \log \theta_O$$

where

$$\begin{aligned}
\alpha_A^{(r)} &= \frac{2n_1\theta_A^{(r)2}}{\theta_A^{(r)2} + 2\theta_A^{(r)}\theta_O^{(r)}} + n_3 + \frac{2n_1\theta_A^{(r)}\theta_O^{(r)}}{\theta_A^{(r)2} + 2\theta_A^{(r)}\theta_O^{(r)}} \\
\alpha_B^{(r)} &= n_3 + \frac{2n_2\theta_B^{(r)2}}{\theta_B^{(r)2} + 2\theta_B^{(r)}\theta_O^{(r)}} + \frac{2n_2\theta_B^{(r)}\theta_O^{(r)}}{\theta_B^{(r)2} + 2\theta_B^{(r)}\theta_O^{(r)}} \\
\alpha_O^{(r)} &= \frac{2n_1\theta_A^{(r)}\theta_O^{(r)}}{\theta_A^{(r)2} + 2\theta_A^{(r)}\theta_O^{(r)}} + \frac{2n_2\theta_B^{(r)}\theta_O^{(r)}}{\theta_B^{(r)2} + 2\theta_B^{(r)}\theta_O^{(r)}} + 2n_4.
\end{aligned}$$

and $n_1, \ldots, n_4$ are the observed counts for phenotypes A,B,AB and O.

Example: Genetics of Human Blood Groups

By the results for the multinomial likelihood, we can maximize $Q(\theta; \theta^{(r)})$ analytically to get

$$\begin{aligned}
\theta_A^{(r+1)} &= \frac{\alpha_A^{(r)}}{\alpha_A^{(r)} + \alpha_B^{(r)} + \alpha_O^{(r)}} \\
\theta_B^{(r+1)} &= \frac{\alpha_B^{(r)}}{\alpha_A^{(r)} + \alpha_B^{(r)} + \alpha_O^{(r)}} \\
\theta_O^{(r+1)} &= \frac{\alpha_O^{(r)}}{\alpha_A^{(r)} + \alpha_B^{(r)} + \alpha_O^{(r)}}
\end{aligned}$$

Example: Genetics of Human Blood Groups

Real Data Example: Data from Clarke *et. al.* (1959)
We have $n_1 = 186$, $n_2 = 38$, $n_3 = 13$ and $n_4 = 284$ for the numbers of A,B,AB and O phenotypes in a sample of $n = 521$. Starting the iterative procedure at $\theta^{(0)} = (1/3, 1/3, 1/3)^\intercal$ yields the following first ten iterations:

r	$\theta_A^{(r)}$	$\theta_B^{(r)}$	$\theta_O^{(r)}$
1	0.25047985	0.06110045	0.68841971
2	0.21845436	0.05049394	0.73105170
3	0.21418233	0.05016173	0.73565593
4	0.21366195	0.05014667	0.73619139
5	0.21359944	0.05014547	0.73625508
6	0.21359196	0.05014535	0.73626270
7	0.21359106	0.05014533	0.73626361
8	0.21359095	0.05014533	0.73626372
9	0.21359094	0.05014533	0.73626373
10	0.21359094	0.05014533	0.73626373

indicating that convergence to the maximum value is fairly rapid.

The EM Algorithm

The EM algorithm

- can often be slow to converge, although speed-up methods exist.
- is only guaranteed to find a local maximum of the likelihood; typically, multiple starting values are used
- does not readily yield standard errors, as it does not utilize (or differentiate) the likelihood directly

The EM Algorithm: Variance Estimates

Let

$$\mathcal{I}_{X|Y}(\theta; \boldsymbol{y}) = -\mathbb{E}_{X|Y,\theta}[\ddot{k}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}]$$

and

$$\mathcal{I}_{X,Y}(\theta) = -\mathbb{E}_{X,Y|\theta}[\ddot{\ell}(\theta; \boldsymbol{X}, \boldsymbol{Y})]$$

denote the **missing** and **complete** information matrices respectively, with corresponding score functions

$$\dot{k}(\theta; \boldsymbol{X}, \boldsymbol{y}) \qquad\qquad \dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})$$

respectively.

The EM Algorithm: Variance Estimates

By definition

$$\dot{\ell}(\theta; \boldsymbol{x}, \boldsymbol{y}) = \frac{\partial}{\partial \theta}\{\log f_{\boldsymbol{X},\boldsymbol{Y}|\theta}(\boldsymbol{x}, \boldsymbol{y}|\theta)\} = \frac{\partial}{\partial \theta}\{\log f_{\boldsymbol{X}|\boldsymbol{Y},\theta}(\boldsymbol{x}|\boldsymbol{y}, \theta) + \log f_{\boldsymbol{Y}|\theta}(\boldsymbol{y}|\theta)\}$$

so therefore

$$\begin{aligned} \dot{\ell}_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) &= \dot{k}(\theta; \boldsymbol{x}, \boldsymbol{y}) + \dot{\ell}_Y(\theta; \boldsymbol{y}) \\ \ddot{\ell}_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) &= \ddot{k}(\theta; \boldsymbol{x}, \boldsymbol{y}) + \ddot{\ell}_Y(\theta; \boldsymbol{y}) \end{aligned}$$

Replacing $\boldsymbol{x}$ by $\boldsymbol{X}$, and taking expectation with respect to the distribution $f_{\boldsymbol{X}|\boldsymbol{Y},\theta}$ at $\boldsymbol{Y} = \boldsymbol{y}$ yields

$$\mathcal{I}_{X,Y}(\theta; \boldsymbol{y}) = \mathcal{I}_{X|Y}(\theta; \boldsymbol{y}) - \ddot{\ell}_Y(\theta; \boldsymbol{y})$$

say, where $\mathcal{I}_{X,Y}(\theta; \boldsymbol{y}) = -\mathbb{E}_{\boldsymbol{X}|\boldsymbol{Y}}[\ddot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}]$.

The EM Algorithm: Variance Estimates

Hence

$$-\ddot{\ell}_Y(\theta; \boldsymbol{y}) = \mathcal{I}_{X,Y}(\theta; \boldsymbol{y}) - \mathcal{I}_{X|Y}(\theta; \boldsymbol{y})$$

This can be interpreted as

Observed data information at $\boldsymbol{y}$

=

Expected complete data information evaluated at $\boldsymbol{y}$

-

Expected latent data information evaluated at $\boldsymbol{y}$.

Replacing $\boldsymbol{y}$ by $\boldsymbol{Y}$, and taking a further expectation over $f_{\boldsymbol{Y}}$, we have

$$\mathcal{I}_Y(\theta) = \mathcal{I}_{X,Y}(\theta) - \mathbb{E}_{f_{\boldsymbol{Y}|\theta}}[\mathcal{I}_{X|Y}(\theta; \boldsymbol{Y})]$$

The EM Algorithm: Variance Estimates

Now

$$\dot{k}(\theta; \boldsymbol{x}, \boldsymbol{y}) = \dot{\ell}_{X,Y}(\theta; \boldsymbol{x}, \boldsymbol{y}) - \dot{\ell}_Y(\theta; \boldsymbol{y})$$

so for any fixed $\boldsymbol{y}$ the final term is a constant, so replacing $\boldsymbol{x}$ by $\boldsymbol{X}$ and taking conditional expectations wrt $f_{\boldsymbol{X}|\boldsymbol{Y},\theta}$, we have

$$\begin{aligned}
\mathcal{I}_{X|Y}(\theta; \boldsymbol{y}) &= \mathbb{V}ar_{f_{X|Y,\theta}}[\dot{k}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}] \\
&= \mathbb{V}ar_{f_{X|Y,\theta}}[\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}] \\
&= \mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})^\intercal|\boldsymbol{y}] \\
&\qquad -\mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}]\mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}]^\intercal
\end{aligned}$$

The EM Algorithm: Variance Estimates

By the usual expansion

$$\mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}] = \mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{X|Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})|\boldsymbol{y}] + \mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{Y}(\theta; \boldsymbol{Y})|\boldsymbol{y}]$$

and in this expression, the first expectation on the RHS is zero by the usual properties of (conditional) score functions. Therefore

$$\mathcal{I}_{X|Y}(\theta; \boldsymbol{y}) = \mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})\dot{\ell}_{X,Y}(\theta; \boldsymbol{X}, \boldsymbol{Y})^{\mathsf{T}}|\boldsymbol{y}] - \dot{\ell}_{Y}(\theta; \boldsymbol{y})\dot{\ell}_{Y}(\theta; \boldsymbol{y})^{\mathsf{T}}$$

Now from the previous expression

$$-\ddot{\ell}_{Y}(\theta; \boldsymbol{y}) = \mathcal{I}_{X,Y}(\theta; \boldsymbol{y}) - \mathcal{I}_{X|Y}(\theta; \boldsymbol{y})$$

At the ML estimate, $\dot{\ell}_{Y}(\widehat{\theta}; \boldsymbol{y}) = 0$.

The EM Algorithm: Variance Estimates

Therefore, at the ML estimate

$$-\ddot{\ell}_Y(\widehat{\theta};\boldsymbol{y}) \quad = \quad \mathcal{I}_{X,Y}(\widehat{\theta};\boldsymbol{y}) - \mathbb{E}_{f_{X|Y,\theta}}[\dot{\ell}_{X,Y}(\widehat{\theta};\boldsymbol{X},\boldsymbol{Y})\dot{\ell}_{X,Y}(\widehat{\theta};\boldsymbol{X},\boldsymbol{Y})^\intercal|\boldsymbol{y}]$$

Both quantities on the right hand side are computed during the EM algorithm steps, as functions of the complete data, so the left hand side can be estimated without recourse to other methods.

MM Algorithms

A critical step in the EM algorithm derivation is the realization that the iterations produce a monotone increasing sequence of likelihood values; this guarantees convergence if the likelihood is bounded above.

This motivates a general objective: to produce an iterative scheme to maximize or function $g(\theta)$ such that for the sequence of points generated, $\{\theta_n\}$, for all n,

$$g(\theta_{n+1}) \geqslant g(\theta_n)$$

for maximization, or

$$g(\theta_{n+1}) \leqslant g(\theta_n)$$

for minimization.

MM Algorithms

An MM algorithm

- for minimization: "majorize/minimize"
- for maximization: "minorize/maximize"

produces such a sequence. A function $h(x; x_n)$ *majorizes* g at x_n if $g(x_n) = h(x_n; x_n)$ and

$$g(x) \leqslant h(x; x_n) \qquad \forall x \neq x_n$$

Therefore if next value x_{n+1} is chosen such that

$$x_{n+1} = \operatorname*{argmin}_x h(x; x_n) \qquad \text{or} \qquad h(x_{n+1}; x_n) \leqslant h(x_n; x_n)$$

then $g(x_{n+1}) \leqslant g(x_n)$. The function h might be easier to work with than g.

At an interior point of the support of g, as $g(x_n) = h(x_n; x_n)$,

$$\dot{g}(x_n) = \dot{h}(x_n; x_n)$$

and

$$\dot{h}(x_n; x_n) - \dot{g}(x_n)$$

is positive definite.

MM Algorithms: Convexity methods

For **convex** functions, a method of majorization follows from

$$g(\alpha^\intercal t) \leqslant \sum_{j=1}^{p} \alpha_j g(t_j)$$

which is useful if we choose for any two positive points x and y, and vector of positive constants c,

$$\alpha_j = \frac{c_j y_j}{c^\intercal y} \qquad\qquad t_j = (c^\intercal y)\frac{x_j}{y_j}$$

then

$$g(c^\intercal x) \leqslant \sum_{j=1}^{p} \frac{c_j y_j}{c^\intercal y} g((c^\intercal y)x_j/y_j) = h(x;y)$$

say. This leads to the potential for a *componentwise* majorization.

MM Algorithms: Convexity methods

To relax positivity: choose α_j according to

$$0 \leqslant \alpha_j \leqslant 1 \qquad \sum_{j=1}^{p} \alpha_j = 1$$

such that

$$g(c^\intercal x) \leqslant \sum_{j=1}^{p} \alpha_j g\left(\frac{\alpha_j}{c_j}(x_j - y_j) + c^\intercal y\right) = h(x; y)$$

provided $c_j \neq 0$ for all j. Could choose

$$\alpha_j = \frac{|c_j|^\gamma}{\sum_{k=1}^{p} |c_k|^\gamma}$$

for $\gamma \geqslant 0$.

MM Algorithms: Derivative methods

First order method: for a **concave** function

$$g(x) \leqslant g(y) + \dot{g}(y)^\intercal (x - y) = h(x; y)$$

Second order method: by a Taylor expansion, for intermediate point z

$$\begin{aligned} g(x) &= g(y) + \dot{g}(y)^\intercal (x - y) + \frac{1}{2}(x - y)^\intercal \ddot{g}(z)(x - y) \\ &\leqslant g(y) + \dot{g}(y)^\intercal (x - y) + \frac{1}{2}(x - y)^\intercal A(x - y) \end{aligned}$$

for some matrix A, provided $A - \ddot{g}(z)$ is non-negative definite for all z. The right-hand side of this inequality defines $h(x; y)$.

MM Algorithms: Example

Example (Linear Least Squares)

In the linear least squares problem, we aim to minimize the function

$$g(\theta) = \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^\mathsf{T}\theta)^2$$

for data $(y_i, \boldsymbol{x}_i), i = 1, \ldots, n$. Note that the terms in the summand are convex in (the elements of) θ, so consider the majorization at current point $\theta^{(r)}$

$$g(\theta) \leqslant \sum_{i=1}^{n} \sum_{j=1}^{p} \alpha_{ij} \left(y_i - \frac{x_{ij}}{\alpha_{ij}} (\theta_j - \theta_j^{(r)}) - \boldsymbol{x}_i^\mathsf{T}\theta^{(r)} \right)^2$$

for which the RHS defines $h(\theta; \theta^{(r)})$.

MM Algorithms: Example

Example (Linear Least Squares)

The minimum of this $h(\theta; \theta^{(r)})$ is achieved at

$$\theta_j^{(r+1)} = \theta_j^{(r)} + \frac{\sum_{i=1}^{n} x_{ij}(y_i - \boldsymbol{x}_i^\mathsf{T}\theta^{(r)})}{\sum_{i=1}^{n} \frac{x_{ij}^2}{\alpha_{ij}}}$$

which defines the recursion. Again define

$$\alpha_{ij} = \frac{|x_{ij}|^\gamma}{\sum_{k=1}^{p} |x_{ik}|^\gamma}$$

for some appropriate γ. This procedure avoids matrix inversion.

Block Relaxation and Coordinate Methods

Block relaxation optimization methods update different (possibly disjoint) subsets of the components of the parameter vector at each step of the recursion, cycling through the subsets to complete an update. This strategy can improve convergence, or be easier to implement.

When each block just involves a single parameter, the approach is termed a **cyclic coordinate** method, that is

- *cyclic coordinate descent* for minimization
- *cyclic coordinate ascent* for maximization

Example (Poisson log-linear model)

A Poisson log-linear model is often used to model data in a multi-way contingency table. For a three-way table, defined by three factors A, B and C, it is assumed that $Y_{ijk} \sim Poisson(\mu_{ijk})$ where the saturated model can be written

$$\log \mu_{ijk} = \lambda_0 + \lambda_i^{(A)} + \lambda_j^{(B)} + \lambda_k^{(C)} + \lambda_{ij}^{(AB)} + \lambda_{ik}^{(AC)} + \lambda_{jk}^{(BC)} + \lambda_{ijk}^{(ABC)}$$

and all other log-linear models are nested inside the saturated model. If the factors take I, J and K levels, then there are a maximum of IJK parameters. The log likelihood is

$$\ell(\lambda) = \sum_i \sum_j \sum_k (y_{ijk} \log \mu_{ijk} - \mu_{ijk}) + \text{const.}$$

Example (Poisson log-linear model)

Identifiability: consider the model conventionally written $Y \sim A$, that is, the main effect A only model. In this model, the mean parameter is explained only as a function of the first factor index, that is, for all j, k,

$$\log \mu_{ijk} = \lambda_0 + \lambda_i^{(A)} \qquad i = 1, \ldots, I$$

The model in this form has $I + 1$ parameters, where only I are identifiable. Thus constraints for identifiability must be introduced; typically, these constraints take the form

$$\sum_{i=1}^{I} \lambda_i^{(A)} = 0.$$

Example (Poisson log-linear model)

For interaction terms, the constraints are of the form

$$\sum_{j=1}^{J} \lambda_{ij}^{(AB)} = 0$$

or

$$\sum_{j=1}^{J} \sum_{k=1}^{K} \lambda_{ijk}^{(ABC)} = 0$$

for each i.

Example (Poisson log-linear model)

To find the ML estimates, note that solving $\ell_{\lambda_0}(\lambda) = 0$ reduces to

$$\ell_{\lambda_0}(\lambda) = \sum_i \sum_j \sum_k (y_{ijk} - \mu_{ijk}) = 0$$

But note that μ_{ijk} can be written

$$\mu_{ijk} = \exp\{\lambda_0\}\omega_{ijk}$$

so therefore

$$\exp\{\lambda_0\} = \frac{\sum_i \sum_j \sum_k y_{ijk}}{\sum_i \sum_j \sum_k \omega_{ijk}} = \frac{N}{\omega_{...}}$$

say, where N is the total count, whatever the ω values are.

Example (Poisson log-linear model)

Therefore the log likelihood using this reparameterization is

$$\begin{aligned}\ell(\omega) &= \sum_i \sum_j \sum_k y_{ijk} \log \frac{N\omega_{ijk}}{\omega_{...}} - N \\ &= \sum_i \sum_j \sum_k y_{ijk} \log \frac{\omega_{ijk}}{\omega_{...}} + N \log N - N\end{aligned}$$

This is a complicated function of the ω parameters, so differentiating and solving to compute the ML estimates is not possible analytically (the exception to this is the case of the saturated model).

Example (Poisson log-linear model)

For a non-saturated model, the Poisson log likelihood is not amenable to direct maximization, but a block-based method can be used. Consider the model written in traditional notation as

$$A + B + C + A.B + A.C + B.C$$

where

$$\log \mu_{ijk} = \lambda_0 + \lambda_i^{(A)} + \lambda_j^{(B)} + \lambda_k^{(C)} + \lambda_{ij}^{(AB)} + \lambda_{ik}^{(AC)} + \lambda_{jk}^{(BC)}$$

and the identifiability constraints are

$$\sum_i \lambda_i^{(A)} = 0 \qquad \sum_j \lambda_{ij}^{(AB)} = 0 \qquad \sum_k \lambda_{ik}^{(AC)} = 0$$

and similarly for the other factors.

Example (Poisson log-linear model)

Write

$$\begin{aligned} \mu_{ij} &= \exp\{\lambda_0 + \lambda_i^{(A)} + \lambda_j^{(B)} + \lambda_{ij}^{(AB)}\} \\ \alpha_{ijk} &= \exp\{\lambda_k^{(C)} + \lambda_{ik}^{(AC)} + \lambda_{jk}^{(BC)}\} \end{aligned}$$

and consider the score equation for a single parameter, $\lambda_{ij}^{(AB)}$ say. Note first that for **any** of the parameters ϕ that appear in the form of μ_{ijk},

$$\frac{\partial \mu_{ijk}}{\partial \phi} = \mu_{ijk}$$

so therefore, using the . symbol to indicate summation, we have

$$\frac{\partial \ell(\lambda)}{\partial \lambda_{ij}^{(AB)}} = \sum_k (y_{ijk} - \mu_{ijk}) = y_{ij.} - \mu_{ij.}$$

Example (Poisson log-linear model)

But

$$\mu_{ij.} = \mu_{ij}\alpha_{ij.}$$

so therefore setting the score equal to zero yields

$$\mu_{ij} = \frac{y_{ij.}}{\alpha_{ij.}}$$

Now, if

$$\sum_k (y_{ijk} - \mu_{ijk}) = 0$$

then

$$\frac{\partial \ell(\lambda)}{\partial \lambda_0} = \frac{\partial \ell(\lambda)}{\partial \lambda_i^{(A)}} = \frac{\partial \ell(\lambda)}{\partial \lambda_j^{(B)}} = 0.$$

Example (Poisson log-linear model)

Assuming the α_{ijk} (i.e. $\lambda_k^{(C)}$, $\lambda_{ik}^{(AC)}$, and $\lambda_{jk}^{(BC)}$) are held constant, the remaining score equations can be solved analytically

$$\begin{aligned}
\lambda_0 &= \frac{1}{IJ}\sum_i\sum_j \log\mu_{ij} = \frac{1}{IJ}\sum_i\sum_j \log\left\{\frac{y_{ij.}}{\alpha_{ij.}}\right\} \\
\lambda_i^{(A)} &= \frac{1}{J}\sum_j \log\mu_{ij} - \lambda_0 \\
\lambda_j^{(B)} &= \frac{1}{I}\sum_i \log\mu_{ij} - \lambda_0 \\
\lambda_{ij}^{(AB)} &= \log\mu_{ij} - \lambda_0 - \lambda_i^{(A)} - \lambda_j^{(B)}
\end{aligned}$$

These parameters are thus updated together according to these formulae, and identify the identifiability constraints.

Example (Poisson log-linear model)

The subsequent steps in the iteration update

- $\lambda_0, \lambda_i^{(A)}, \lambda_k^{(C)}, \lambda_{ik}^{(AC)}$
- $\lambda_0, \lambda_j^{(B)}, \lambda_k^{(C)}, \lambda_{ik}^{(BC)}$

and then all three updates are carried out repeatedly until convergence. Note that the parameter blocks here are not disjoint.

Note also that this algorithm leads to a non-decreasing sequence of log likelihood values. The algorithm is termed *Iterative Proportional Fitting*.

Example (The LASSO)

The LASSO (Least Absolute Shrinkage and Selection Operator) is a form of penalized loss estimation procedure, where the minimum of

$$g(\theta) = \mathcal{L}(\boldsymbol{x}; \theta) + \lambda \sum_{j=1}^{p} |\theta_j|$$

is sought, where $\mathcal{L}$ is a data-dependent loss function, and $\lambda > 0$ is a tuning parameter at the control of the user. Most often the LASSO in the context of (L_2) regression

$$g(\beta) = \|\boldsymbol{y} - \boldsymbol{x}\beta\|^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

but $\mathcal{L}$ could be any negative log likelihood.

Example (The LASSO)

The form of the penalty

$$\sum_{j=1}^{p} |\beta_j|$$

renders this a relatively challenging minimization problem, as the penalty is not differentiable at $\beta_j = 0$.

A common method of solution is cyclic coordinate descent, where the parameters are updated in turn to decrease the objective function.

Example (The LASSO)

Assuming $\mathcal{L}$ is differentiable, taking the derivative of g from the right yields

$$g_{\theta_j}^{+}(\theta) = \dot{\mathcal{L}}_{\theta_j}(\boldsymbol{x};\theta) + \begin{cases} \lambda & \theta_j \geqslant 0 \\ -\lambda & \theta_j < 0 \end{cases}$$

and from the left yields

$$g_{\theta_j}^{-}(\theta) = -\dot{\mathcal{L}}_{\theta_j}(\boldsymbol{x};\theta) + \begin{cases} -\lambda & \theta_j > 0 \\ \lambda & \theta_j \leqslant 0 \end{cases}$$

where

$$g_{\theta}^{\pm}(\theta) = \lim_{t \downarrow 0} \frac{g(\theta \pm t\epsilon_\theta) - g(\theta)}{t}$$

Example (The LASSO)

Cyclic coordinate descent proceeds as follows:

1. Select starting values for $\theta_1^{(0)}, \ldots, \theta_p^{(0)}$.
2. For $j = 1, \ldots, p$, evaluate $g_{\theta_j}^{+}(\theta)$ and $g_{\theta_j}^{-}(\theta)$ at the current parameter value $\theta_j^{(0)}$:
   - If both derivatives are **non-negative**, do not update θ_j
   - If one derivative is **negative** (indicating the potential for reducing the value of the objective function), update θ_j by moving in the direction of greatest possible decrease.
3. Complete the cycle to generate $\theta_1^{(1)}, \ldots, \theta_p^{(1)}$.
4. Cycle through 1. to 3. until the updated vector $\theta_1^{(r)}, \ldots, \theta_p^{(r)}$ ceases to decrease the objective function.

Example (The LASSO)

Implementing the update: If $\theta_j^{(r)} = 0$

$$g_{\theta_j}^{+}(\theta) = \dot{\mathcal{L}}_{\theta_j}(\boldsymbol{x};\theta) + \lambda \qquad\qquad g_{\theta_j}^{-}(\theta) = -\dot{\mathcal{L}}_{\theta_j}(\boldsymbol{x};\theta) + \lambda$$

so

- if $\dot{\mathcal{L}}_{\theta_j}(\boldsymbol{x};\theta) < -\lambda$, **increase** θ_j,
- else if $\dot{\mathcal{L}}_{\theta_j}(\boldsymbol{x};\theta) > \lambda$, **decrease** θ_j.
- else do not update θ_j, set $\theta_j^{(r+1)} = \theta_j^{(r)}$

Example (The LASSO)

In penalized linear (L_2) regression with an intercept β_0:

$$g(\beta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \beta_0 - \boldsymbol{x}_i^\intercal\beta)^2 + \lambda\sum_{j=1}^{p}|\beta_j|.$$

The update for β_0 is straightforward

$$\beta_0^{(r+1)} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\intercal\beta^{(r)}) = \beta_0^{(r)} - \frac{1}{n}\dot{\mathcal{L}}_{\beta_0}(\beta^{(r)})$$

say.

Example (The LASSO)

The update for the other β parameters can be computed using the ideas above: either

$$\beta_j^{(r+1)} = \min \left\{ 0, \beta_j^{(r)} - \frac{\dot{\mathcal{L}}_{\beta_j}(\beta^{(r)}) - \lambda}{\sum_{i=1}^{n} x_{ij}^2} \right\}$$

or

$$\beta_j^{(r+1)} = \max \left\{ 0, \beta_j^{(r)} - \frac{\dot{\mathcal{L}}_{\beta_j}(\beta^{(r)}) + \lambda}{\sum_{i=1}^{n} x_{ij}^2} \right\}$$

as only one of these is non-zero.

Part II

Integration

Applications of Integration in Statistics

The most common use of integration methods in statistics is in Bayesian analysis. For the posterior computation

$$\pi(\theta|\boldsymbol{y}) = \frac{L(\theta;\boldsymbol{y})\pi(\theta)}{\int_\Theta L(t;\boldsymbol{y})\pi(t)\,dt}$$

where the denominator is the *marginal likelihood* or *prior predictive distribution*

$$M(\boldsymbol{Y}) = \int_\Theta L(t;\boldsymbol{Y})\pi(t)\,dt$$

This integral is over a region determined by the model and the dimension of the parameter; typically this is some open subset of $\mathbb{R}^p$.

The ratio marginal likelihoods for two models

$$\frac{M_1(\boldsymbol{y})}{M_2(\boldsymbol{y})} = \frac{\int_\Theta L_1(t;\boldsymbol{Y})\pi_1(t)\,dt}{\int_\Theta L_2(t;\boldsymbol{Y})\pi_2(t)\,dt}$$

can be used as the basis of a model selection procedure. Model 1 is preferred to Model 2 if this ratio is greater than 1.

The ratio M_1/M_2 is termed the *Bayes factor*.

Note: the marginal likelihood measures how well the model (likelihood and prior) predict the **observed** data, that is, it is a "within-sample" measure.

Applications of Integration in Statistics

Bayesian estimation typically relies on minimizing some expected posterior loss

$$\widehat{\theta} = \underset{t}{\operatorname{argmin}} \left\{ \int_{\Theta} \mathcal{L}(t, \theta) \pi(\theta | \boldsymbol{y}) \, d\theta \right\}$$

and sometimes this results in having to compute a functional of the posterior of the form

$$\mathbb{E}_{\theta|\boldsymbol{Y}}[h(\theta)] = \int_{\Theta} h(\theta) \pi(\theta | \boldsymbol{y}) \, d\theta$$

The *posterior predictive distribution* is also of interest

$$f(\boldsymbol{y}^* | \boldsymbol{y}) = \int_{\Theta} L(\boldsymbol{y}^*; \theta) \pi(\theta | \boldsymbol{y}) \, d\theta$$

Applications of Integration in Statistics

In a non-Bayesian setting, the marginal likelihood is required for the analysis of certain sorts of longitudinal data. Suppose the following data are available:

- For individual $i = 1, \ldots, N$, responses y_{ij}, $j = 1, \ldots, n_i$, are recorded over different time points
- Covariate values $\boldsymbol{x}_{ij}$ are available at baseline and on subsequent visits.

A *linear mixed model* or *linear random effects model* is commonly used

$$Y_{ij} = \boldsymbol{x}_i^\mathsf{T} \beta + \eta_i + \epsilon_{ij}$$

with, say, $\epsilon_{ij} \sim \text{Normal}(0, \sigma_\epsilon^2)$ are (possibly correlated) residual errors, and where random effects $\eta_1, \ldots, \eta_N$ are presumed to be iid and $\eta_i \sim \text{Normal}(0, \sigma_\eta^2)$.

Applications of Integration in Statistics

Here the marginal likelihood is formed by integrating out the random effects parameters. If $\theta = (\beta, \sigma_\epsilon, \sigma_\eta)$, then for individual i,

$$L_i(\theta; \boldsymbol{y}_i, \boldsymbol{x}_i) = \int f(y_{i1} \ldots, y_{in_i} | \theta, \eta_i) p(\eta_i) \, d\eta_i$$

say, and

$$L(\theta; \boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{N} L_i(\theta; \boldsymbol{y}_i, \boldsymbol{x}_i)$$

Inference can then proceed using ML.

Note: this is not the only method for analyzing mixed models, it is also possible to estimate and condition upon the η_i.

Analytical Approximation Methods

In many large sample situations, the posterior distribution can be well approximated by a multivariate normal distributions.

If the normal approximation is not valid as the sample is too small, perhaps an analytical quadratic approximation along the lines of the Laplace approximation may be used.

See, in particular:

- L. Tierney and J.B. Kadane, Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, **81** (1986), pp. 82-86.
- Casella and Robert, p 107.

Basic Numerical Integration

The first method typically encountered for univariate numerical integration is the *trapezium rule:* with grid $x_0, x_1, \ldots, x_n$,

$$x_j = x_0 + jh \qquad\qquad h = (x_n - x_0)/n$$

say, if $y_j = g(x_j)$, then

$$\int_{x_0}^{x_n} g(x)\, dx \doteq h\left[\frac{(y_0 + y_n)}{2} + (y_1 + \cdots + y_{n-1})\right]$$

that is

$$\int_{x_0}^{x_n} g(x)\, dx \doteq \frac{hg(x_0) + hg(x_n)}{2} + \sum_{j=1}^{n-1}(x_j - x_{j-1})g(x_j)$$

The error in this approximation is $\mathrm{O}(h^2)$

Gaussian Quadrature

For integrals with respect to probability distributions, it is natural to consider approximations of the form

$$\int_{-\infty}^{\infty} g(x)p(x)\,dx \doteq \sum_{j=0}^{k} w_j g(x_j)$$

say, for points x_j and weights w_j chosen in line with the probability distribution. In the trapezium rule, the density p is a uniform density, and the points are uniformly distributed.

This method of approximation works well when the integrand is smooth.

Approximating the integrand well is important in evaluating the integral.

A Basic Introduction to Hilbert Spaces

Let $\mathcal{H}$ be a *vector space* with *inner product* $\langle h_1, h_2 \rangle$

- for $a, b \in \mathbb{R}$, $\langle ah_1 + bh_2, h_3 \rangle = a\langle h_1, h_2 \rangle + b\langle h_1, h_3 \rangle$,
- $\langle h_1, h_2 \rangle = \langle h_2, h_1 \rangle$,
- $\langle h_1, h_1 \rangle \geqslant 0$, with $\langle h_1, h_1 \rangle = 0$ if and only if $h_1 \equiv 0$.

The *norm* of $h \in \mathcal{H}$ is

$$\|h\| = \langle h_1, h_1 \rangle^{1/2}$$

Elements of $\mathcal{H}$ could be vectors in $\mathbb{R}^p$ say, in which case the usual inner product is used

$$\langle h_1, h_2 \rangle = h_1^\intercal h_2 = \sum_{j=1}^{p} h_{1j} h_{2j}$$

A final requirement is for a vector space $\mathcal{H}$ to be a *Hilbert space* is that it is *complete*, that is, for a sequence $h_1, h_2, \ldots \in \mathcal{H}$, for every $\epsilon > 0$, there exist m, n such that

$$\|h_m - h_n\| < \epsilon$$

that is,

$$\lim_{n \longrightarrow \infty} h_n = h$$

exists, for $h \in \mathcal{H}$

However, elements of $\mathcal{H}$ can be **infinite dimensional**.

For example, $\mathcal{H}$ could contain real-valued functions that obey certain integrability conditions: for example, if $h \in \mathcal{H}$, with

$$h : \mathbb{R} \longrightarrow \mathbb{R}$$

we might require

$$\int h(x)^2 \, dx < \infty$$

In this case, an inner product that is often used is

$$\langle h_1, h_2 \rangle = \int h_1(x) h_2(x) \, dx$$

This Hilbert space is denoted L^2.

A fundamental result for the Hilbert space L^2 relates to approximation: essentially, functions in L^2 can be arbitrarily well approximated. That is, if g is a square-integrable continuous function defined on $\mathbb{R}$, then g can be approximated by a function h such that

$$\|g - h\| < \epsilon$$

for ϵ arbitrarily small (that is, L^2 is *dense*).

More precisely: this result relates to functions that have ***compact*** *(bounded) support.*

A sequence $\{\psi_n\}_{n\geqslant 1}$ of elements of Hilbert space $\mathcal{H}$ is *orthogonal* if

$$\langle \psi_m, \psi_n \rangle = 0 \quad \text{for all } m \neq n$$

and *orthonormal* if it is orthogonal and also $\|\psi_n\| = 1$ for each n.

In a Hilbert space, $\{\psi_n\}_{n\geqslant 1}$ is an *orthonormal basis* if for each $g \in \mathcal{H}$

$$g = \sum_n \langle g, \psi_n \rangle \psi_n$$

In this case

$$\|g\| = \sum_n |\langle g, \psi_n \rangle|^2$$

Approximation: For each $g \in \mathcal{H}$ and $\epsilon > 0$, there exists an $m_0 = m_0(g, \epsilon)$ such that

$$\left\| g - \sum_{n=1}^{m} \langle g, \psi_n \rangle \psi_n \right\| \leqslant \epsilon$$

for all $m \geqslant m_0$.

That is, the function g can be arbitrarily well approximated using a **finite** series expansion in the orthonormal basis.

All of these results are valid if the regular integral is replaced by an integral with respect to a probability measure μ.

Square-integrability becomes

$$\int h(x)^2 \, \mu(dx) < \infty$$

and the inner product becomes

$$\langle h_1, h_2 \rangle = \int h_1(x) h_2(x) \, \mu(dx)$$

If μ admits a density $p(x)$, then

$$\langle h_1, h_2 \rangle = \int h_1(x) h_2(x) p(x) \, dx$$

Orthonormal Polynomial Bases

Many probability distributions can be associated with a natural orthonormal *polynomial* basis

$$\psi_n(x) = \sum_{j=0}^{\infty} a_{nj} x^j$$

Polynomials are useful for approximating arbitrary real functions. To construct an orthonormal polynomial representation for a probability density is more complicated, but essentially reduces to choosing the coefficients a_{nj}.

Provided the density has a *moment generating function*, it can be shown that an orthonormal polynomial basis representation exists.

Orthonormal Polynomial Bases

The orthonormal polynomials can be constructed using the Gram-Schmidt procedure: let $g_1, \ldots, g_k$ be a sequence of polynomials, and

$$\Pi(g_2|g_1) = \frac{\langle g_1, g_2 \rangle}{\langle g_1, g_1 \rangle} g_1 = \left\{ \frac{\int g_1(x) g_2(x) p(x)\, dx}{\int g_1^2(x) p(x)\, dx} \right\} g_1(x)$$

Then the sequence

$$\begin{aligned} \varphi_1 &= g_1 \\ \varphi_2 &= g_2 - \Pi(g_2|\varphi_1) \\ \varphi_3 &= g_3 - \Pi(g_3|\varphi_1) - \Pi(g_3|\varphi_2) \\ \vdots &\quad \vdots \quad \vdots \end{aligned}$$

is an orthogonal sequence, and $\psi_j = \varphi_j / \|\varphi_j\|$ is the normalized version.

Orthonormal Polynomial Bases

Moments can also be used to define the orthonormal basis. If

$$\mu_k = \mathbb{E}[X^k] = \int x^k p(x)\, dx$$

Then the polynomials $\{\varphi_n\}$, defined by

$$\varphi_n(x) = \begin{vmatrix} \mu_0 & \mu_1 & \mu_2 & \cdots & \mu_n \\ \mu_1 & \mu_2 & \mu_3 & \cdots & \mu_{n+1} \\ \vdots & \vdots & \vdots & & \vdots \\ \mu_{n-1} & \mu_n & \mu_{n+1} & \cdots & \mu_{2n-1} \\ 1 & x & x^2 & \cdots & x^n \end{vmatrix}$$

are orthogonal with respect to p.

Orthonormal Polynomial Bases

Example (Standard Normal Density)

Suppose

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

We need to find $\{\psi_n\}$ such that

$$\int_{-\infty}^{\infty} \psi_m(x)\psi_n(x) \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\, dx = 0$$

for $m \neq n$.

Orthonormal Polynomial Bases

Example (Standard Normal Density)

Consider the exponential generating function for the $\{\psi_n\}_{n\geqslant 0}$

$$E(x,t) = \sum_{n=0}^{\infty} \frac{\psi_n(x)}{n!} t^n$$

so that

$$\psi_n(x) = \left. \frac{d^n E(x,t)}{dt^n} \right|_{t=0}$$

Suppose that we let

$$E(x,t) = \exp\{xt - t^2/2\}$$

in order to specify the ψ_n.

Example (Standard Normal Density)

Then it follows that

$$\begin{aligned}
\int E(x,s)E(x,t)p(x)\,dx &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{xs-s^2/2}e^{xt-t^2/2}e^{-x^2/2}\,dx \\
&= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{st}e^{-(x-s-t)^2/2}\,dx \\
&= e^{st} \\
&= \sum_{n=0}^{\infty}\frac{s^n t^n}{n!}
\end{aligned}$$

Orthonormal Polynomial Bases

Example (Standard Normal Density)

However, the integral on the left-hand side can be written

$$\int \left\{ \sum_{m=0}^{\infty} \frac{\psi_m(x)}{m!} s^m \right\} \left\{ \sum_{n=0}^{\infty} \frac{\psi_n(x)}{n!} t^n \right\} p(x)\, dx$$

which can be rewritten

$$\sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \left\{ \int \frac{\psi_m(x)}{m!} \frac{\psi_n(x)}{n!} p(x)\, dx \right\} s^m t^n$$

Orthonormal Polynomial Bases

Example (Standard Normal Density)

Equating the two expansions yields that

$$\int \frac{\psi_m(x)}{m!}\frac{\psi_n(x)}{n!}p(x)\,dx = \begin{cases} 0 & m \neq n \\ \dfrac{1}{n!} & m = n \end{cases}$$

and hence that the sequence $\{H_n(x)\}$

$$H_n(x) = \frac{1}{\sqrt{n!}}\psi_n(x)$$

is an orthonormal sequence with respect to this $p(x)$.

$H_n(x)$ is termed the *Hermite polynomial of order n*.

Orthonormal Polynomial Bases

Example (Standard Normal Density)

An explicit form for $H_n(x)$ can be expanding the generating function $E(x,t)$ as

$$H_n(x) = \sum_{j=0}^{\lfloor n/2 \rfloor} \frac{n!(-1)^j x^{n-2j}}{2^j j!(n-2j)!}$$

Recursion formulae can also be derived

$$H_{n+1}(x) = xH_n(x) - nH_{n-1}(x)$$

which is useful for computation.

Orthonormal Polynomial Bases

Example (Gamma Density)

If

$$p(x) = \frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x} \qquad x > 0$$

then define

$$E(x,t) = \frac{1}{(1-t)^{\alpha}} \exp\left\{-\frac{tx}{1-t}\right\} = \sum_{n=0}^{\infty} \frac{\psi_n(x;\alpha)}{n!} t^n$$

to derive the orthonormal polynomial basis $\{L_n(x;\alpha)\}$

$$L_n(x;\alpha) = \frac{1}{n!\alpha^{\overline{n}}} \psi_n(x;\alpha)$$

where $\alpha^{\overline{n}} = \alpha(\alpha+1)\ldots(\alpha+n-1)$.

Orthonormal Polynomial Bases

Example (Gamma Density)

In this case

$$L_n(x;\alpha) = \Gamma(n+\alpha)\sum_{m=0}^{n}\binom{n}{m}\frac{(-1)^m x^m}{\Gamma(m+\alpha)}$$

$L_n(x;\alpha)$ is the (generalized) *Laguerre polynomial of order n*.

If $\alpha = 1$, we have the recursion for $n \geqslant 1$

$$L_{n+1}(x) = \frac{1}{n+1}((2n+1-x)L_n(x) - nL_{n-1}(x))$$

with $L_0(x) = 1, L_1(x) = 1 - x$.

Orthonormal Polynomial Bases

Other orthogonal polynomial bases include

- Chebychev polynomials

$$p(x) = \frac{1}{\pi\sqrt{1-x^2}} \qquad -1 < x < 1$$

- Gegenbauer polynomials

$$p(x) = \frac{\Gamma(\alpha+1)}{\sqrt{\pi}\Gamma(\alpha+1/2)}(1-x^2)^{\alpha-1/2} \qquad -1 < x < 1$$

- Legendre polynomials

$$p(x) = \frac{1}{2} \qquad -1 < x < 1$$

Implication for Quadrature

Recall the quadrature approximation:

$$\int_{-\infty}^{\infty} g(x)p(x)\,dx \doteq \sum_{j=0}^{k} w_j g(x_j)$$

Provided the support of p is not finite, and provided that p has an associated orthonormal polynomial basis $\{\psi_n\}$, then this approximation can be made **exact** for integrands g that are themselves polynomials of order $2k+1$ or lower by choosing the points $x_0, \ldots, x_k$ to solve

$$\psi_{k+1}(x) = 0$$

with weights w_j determined by the $\{\psi_n\}$.

Frequentist example

Example (Variance of Normal median)

If $X_1, \ldots, X_n \sim \mathcal{N}(0,1)$ are a random sample, the sample median $\widetilde{X}$ has (for n odd) the pdf

$$n\binom{n-1}{m}\Phi(x)^m(1-\Phi(x))^m\phi(x)$$

where $m = \lfloor n/2 \rfloor$, and therefore the variance is

$$n\binom{n-1}{m}\int_{-\infty}^{\infty} x^2\Phi(x)^m(1-\Phi(x))^m\phi(x)\,dx = \int_{-\infty}^{\infty} g(x,n)\phi(x)\,dx$$

say. Using the central limit theorem and delta method

$$\sqrt{n}\widetilde{X} \xrightarrow{d} \mathcal{N}(0, \pi/2)$$

so for large n, the variance is approximately $1.5708/n$.

Frequentist example

Example (Variance of Normal median)

Using the Hermite polynomial basis formulation, the definition of the roots $x_0, x_1, \ldots, x_k$ and the fact that we have

$$\int_{-\infty}^{\infty} H_j(x)\phi(x)\,dx = 0$$

for each j, the weights $\boldsymbol{w} = (w_0, \ldots, w_k)^\intercal$ in the quadrature formula can be found by solving

$$\Psi(\boldsymbol{x})\boldsymbol{w} = \begin{pmatrix} \int_{-\infty}^{\infty} H_0(x)\phi(x)\,dx \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

where the (i,j)th element of $\Psi(\boldsymbol{x})$ is $H_i(x_j)$, for $i, j = 0, \ldots, k$.

R implementation

There are functions in R that evaluate the orthonormal polynomials. For (univariate) Gauss-Hermite integration the function `gqz` in the `npmlreg` library.

Example (Variance of Normal median)

For different values of n and k, the following exact values of the variance of the median estimator were found.

	k					
n	10	20	50	100	1000	Exact
11	0.21018	0.14756	0.13717	0.13716	0.13716	0.13716
21	0.13492	0.10954	0.07455	0.07326	0.07326	0.07326
51	0.02228	0.05311	0.04496	0.03204	0.03054	0.03054
101	0.00075	0.01101	0.02858	0.02268	0.01549	0.01549

R Code for Gauss-Hermite integration

```
1  library(npmlreg)
2
3  g.func<-function(xv,nv){
4    mv<-floor(nv/2)
5    gv<-nv*choose(nv-1,mv)*xv^2
6           *(pnorm(xv)^mv)*((1-pnorm(xv))^mv)
7    return(gv)
8  }
9
10 n<-11
11 nk<-20
12 gh<-gqz(nk,minweight=1e-14)
13 x<-gh$location
14 w<-gh$weight
15 g<-g.func(x,n)
16 print(c(n,ik,sum(w*g)))
```

Higher dimensions

Gauss-Hermite integration extends to multivariate integration problems. The simplest extension utilizes the product approximation

$$\int_{-\infty}^{\infty} g(x_1, x_2) p(x_1, x_2)\, dx_1 dx_2 \doteq \sum_{i=0}^{k} \sum_{j=0}^{k} w_i w_j g(x_{1i}, x_{2j})$$

on a rectangular grid, where the weights are determined identically to the univariate case.

However, for general functions, the approximation breaks down in high dimensions

- in p dimensions, need p^{k+1} function evaluations.
- "high" translates as tens.

Part III

Bootstrap and Resampling Methods

Resampling Approaches to Statistical 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 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

$$\begin{aligned} b_{T_n}(F) &= \mathbb{E}_F[T_n] - \theta(F) \\ v_{T_n}(F) &= \mathbb{V}ar_F[T_n] \end{aligned}$$

both of which depend on the true F.

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

$$\sum_{i=1}^{n} m(X_i; \theta) = 0$$

for some m-estimating function. The estimator is not analytically available, so its properties are hard to study.

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 0.7572, yielding the estimated standard error of the mean (s.e.m.)

$$\widehat{\text{se}}(\widehat{\theta}) = \frac{0.7572}{\sqrt{10}} = 0.2395.$$

If the presumption of normality is made, then this is the *plug-in* estimate of the standard error

- the **true** s.e.m. is $\text{se}(\widehat{\theta}) = \sigma/\sqrt{n}$
- we estimate σ by the standard deviation under $\widehat{F}_n$.

Example (Sample mean standard error)

Set $B = 200$. For $b = 1, \ldots, B$, resample, 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

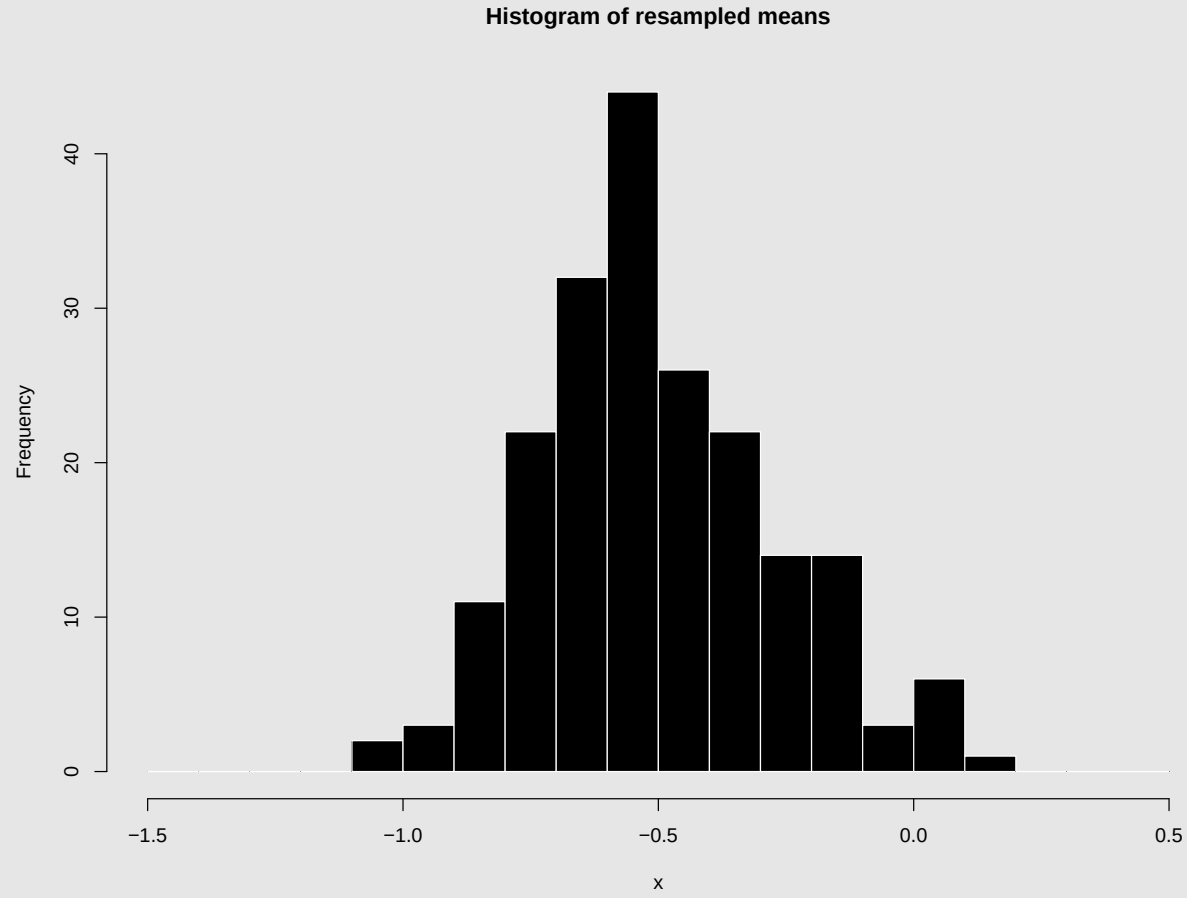
with sample mean -0.2221 and standard deviation 0.9589.

Example (Sample mean standard error)

b	Mean	Variance
1	-0.2221	0.9589
2	-0.7630	0.2068
3	-0.1794	0.7240
4	-0.5606	0.2927
5	-0.8929	0.4093
6	-0.5359	0.6291
7	-0.6969	0.6482
8	-0.4056	0.9639
⋮	⋮	⋮

Example (Sample mean standard error)

We now have 200 (resampled) sample means, with standard deviation $0.2327 \approx s/\sqrt{n}$, where s is the original sample s.d..



Example (Sample kurtosis standard error)

The sample kurtosis

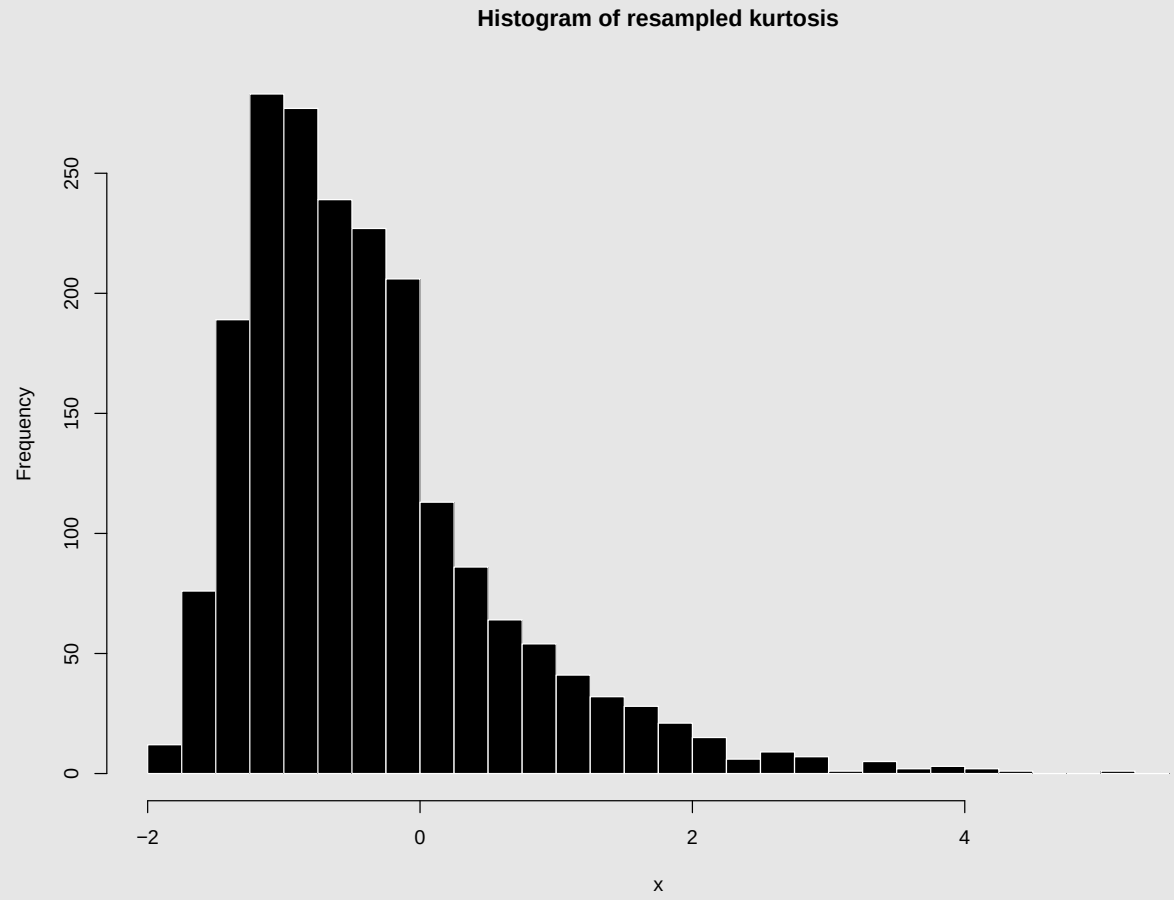
$$t_n = \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2\right)^2} - 3$$

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 $\widehat{\theta}$.

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

$$\overline{y}^{(*)} = \frac{1}{B}\sum_{b=1}^{B}\overline{y}^{(b)} \qquad \overline{y}^{(b)} = \frac{1}{n}\sum_{i=1}^{n} y_i^{(b)}.$$

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 $\text{se}(\widehat{\theta})$ denote the standard error functional for statistic $\widehat{\theta}$, and, suppressing the dependence on $\widehat{\theta}$, let

- $\widehat{\text{se}}_B$ denote the estimate of the standard error from B bootstrap resamples
- $\widehat{\text{se}}_\infty$ denote the ideal estimated bootstrap standard error at $B = \infty$. Recall that $\widehat{\text{se}}_\infty \equiv \widehat{\text{se}}(\widehat{\theta})$ 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

$$\mathbb{E}_F[\widehat{\text{se}}_B^2] = \mathbb{E}_F[\widehat{\text{se}}_\infty^2]$$

and

$$\mathbb{V}ar_F[\widehat{\text{se}}_B^2] \geqslant \mathbb{V}ar_F[\widehat{\text{se}}_\infty^2]$$

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

$$\widehat{\text{se}}(\widehat{\theta})^2 = \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2}{n} = \frac{(n-1)s^2}{n^2}$$

But, by standard theory, under F

$$\frac{(n-1)s^2}{\sigma^2} \sim \chi^2_{n-1}.$$

The Bootstrap: choosing B

Example ($\mathcal{N}(\mu, \sigma^2)$)

Hence, as $\widehat{\text{se}}_\infty \equiv \widehat{\text{se}}$, we have

$$\mathbb{E}_F[\widehat{\text{se}}^2_\infty] = \frac{(n-1)\sigma^2}{n^2} \qquad \mathbb{V}ar_F[\widehat{\text{se}}^2_\infty] = \frac{2(n-1)\sigma^4}{n^4}$$

These results are **not** identical to those obtained taking expectations under $\widehat{F}_n$.

The larger B gets, we expect to see

$$\mathbb{V}ar_F[\widehat{\text{se}}^2_B] \longrightarrow \mathbb{V}ar_F[\widehat{\text{se}}^2_\infty]$$

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

$$\mathbb{E}_F[\widehat{\text{se}}^2_\infty] = \frac{(n-1)}{n^2} \qquad \mathbb{V}ar_F[\widehat{\text{se}}^2_\infty] = \frac{2(n-1)}{n^4}.$$

For $n = 10$, these values are 0.090 and 0.0424^2 respectively.

Using 2000 replicate runs, for different values of B, the estimated following values were obtained:

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.

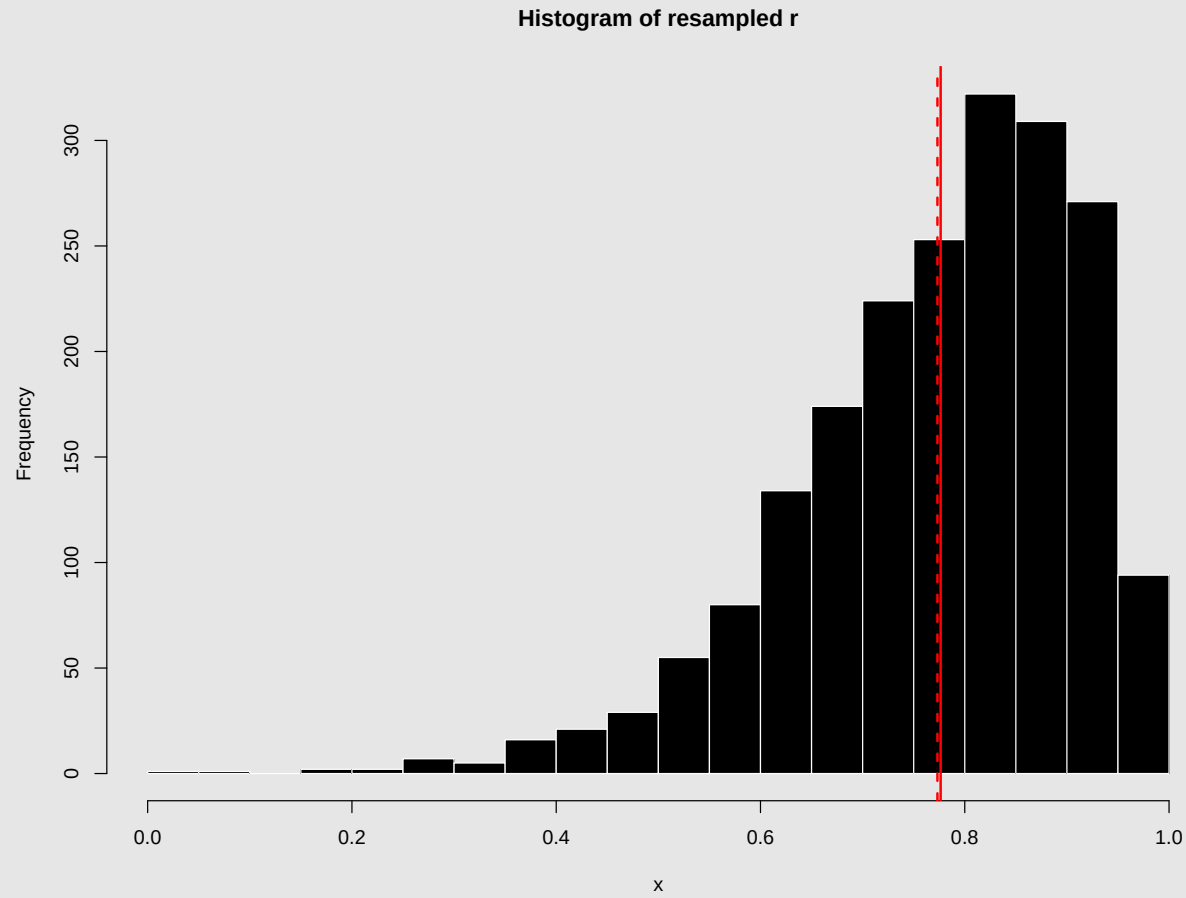
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
GPA	339	330	281	303	344	307	300	343	336	313	312	274	276	288	296
LSAT	576	635	558	578	666	580	555	661	651	605	653	575	545	572	594

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

$$\sqrt{n-3}Z_n \xrightarrow{d} Z \sim \mathcal{N}(\mu, 1)$$

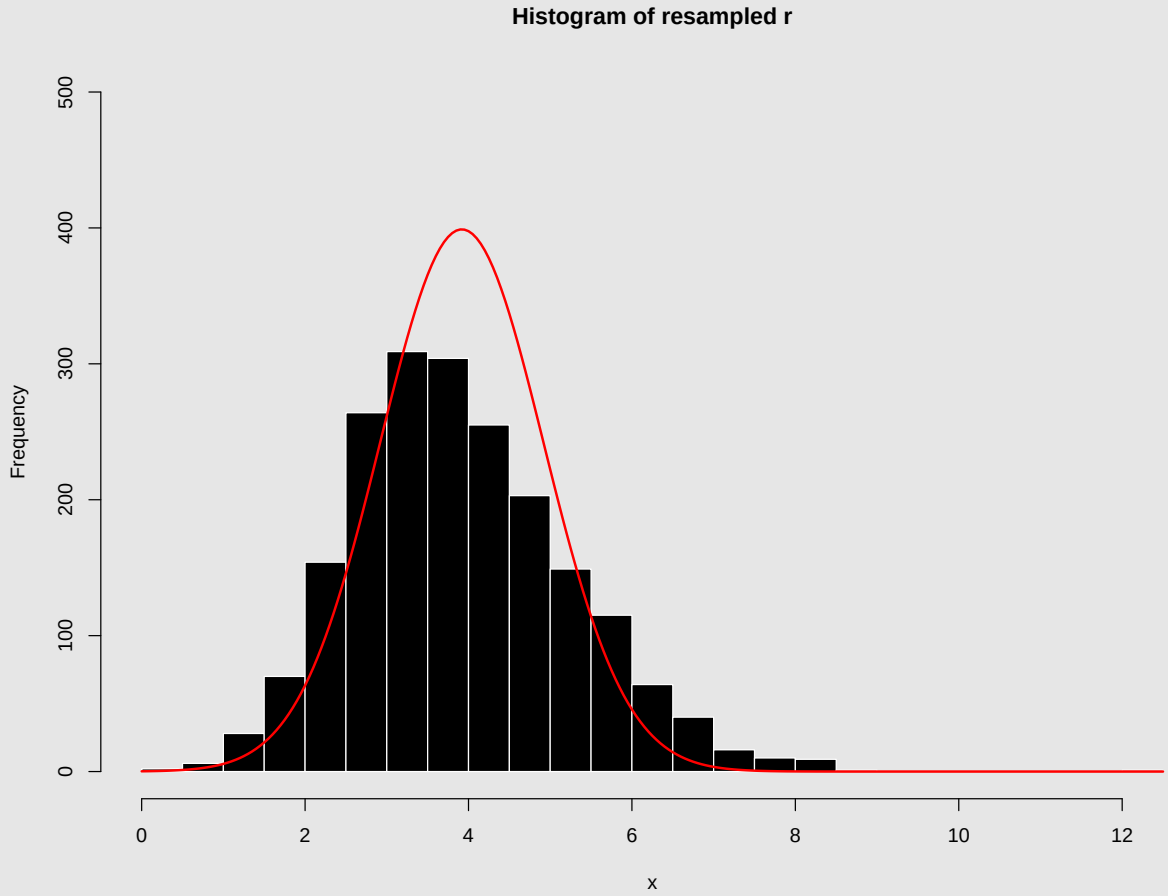
where

$$\mu = \frac{1}{2}\log\left(\frac{1+\rho}{1-\rho}\right)$$

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 p, $\boldsymbol{x}_i, i = 1, \ldots, n$, are available. Let X denote the $n \times p$ matrix formed by row-binding the n i.i.d. vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, after *column standardization*

- each column of X has its mean subtracted.

Then the sample covariance matrix is

$$S = \frac{1}{n} X^\intercal X$$

This is a positive-definite matrix, that can be decomposed using the eigen-decomposition

$$S = VDV^\intercal$$

where D is the diagonal matrix of eigenvalues, and 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 quantities

$$\widehat{\theta}_j = \frac{\widehat{\lambda}_j}{\sum_{l=1}^{p} \widehat{\lambda}_l} \qquad j = 1, \ldots, p.$$

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** $(\boldsymbol{x}_1^{(b)}, \ldots, \boldsymbol{x}_n^{(b)})$ (size n)
- forming the resampled matrix $X^{(b)}$,
- forming the resampled covariance matrix $S^{(b)}$,
- 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.

The Bootstrap: Principal components

Example (Education test scores)

88 students each took five tests on mechanics, vectors, algebra, analysis and statistics. This yielded the S matrix

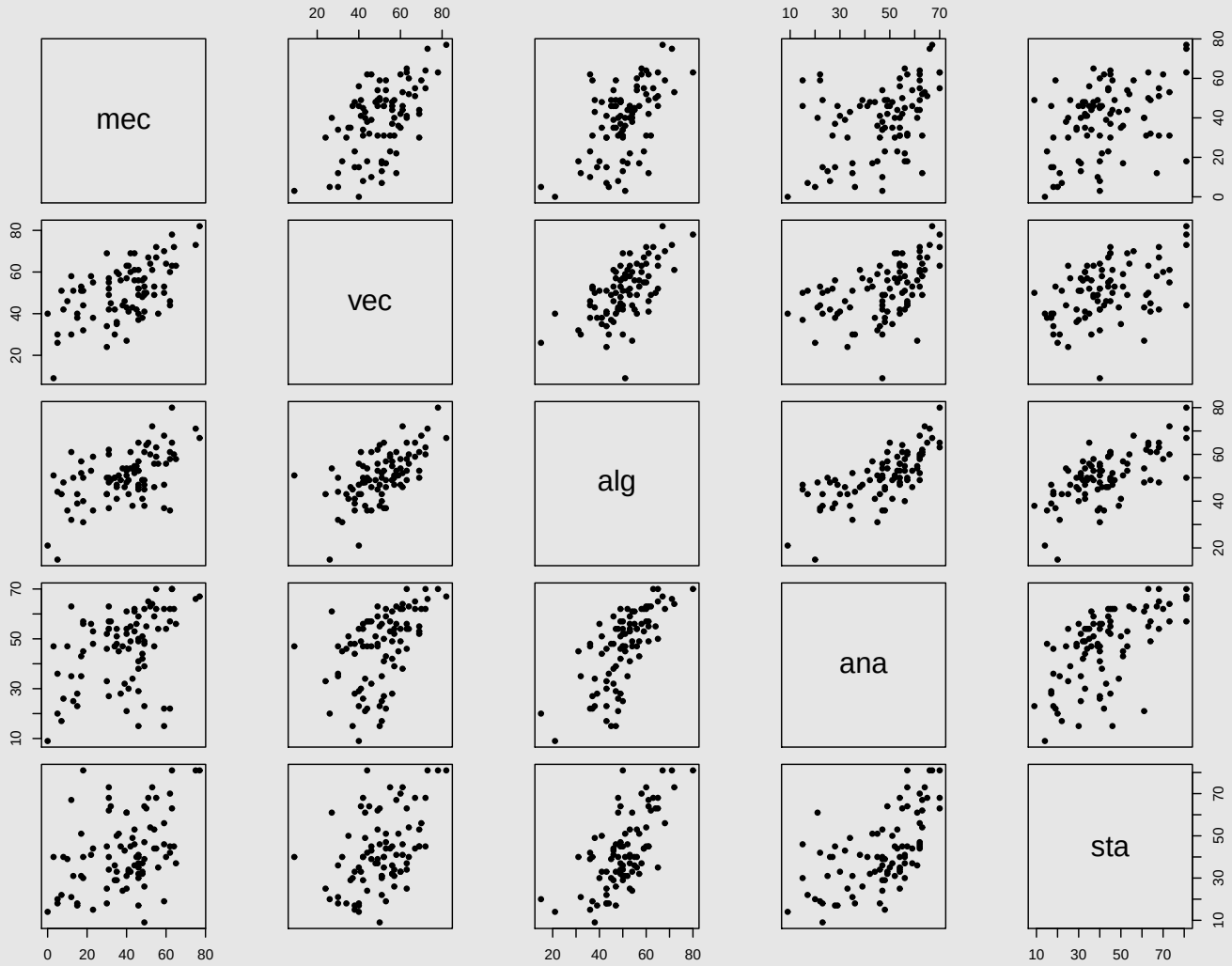
$$S = \begin{bmatrix} 302.29 & 125.78 & 100.43 & 105.07 & 116.07 \\ 125.78 & 170.88 & 84.19 & 93.60 & 97.89 \\ 100.43 & 84.19 & 111.60 & 110.84 & 120.49 \\ 105.07 & 93.60 & 110.84 & 217.88 & 153.77 \\ 116.07 & 97.89 & 120.49 & 153.77 & 294.37 \end{bmatrix}$$

and eigenvalues

$$\hat{\boldsymbol{\lambda}} = (679.18, 199.81, 102.57, 83.67, 31.79)$$

The Bootstrap: Principal components

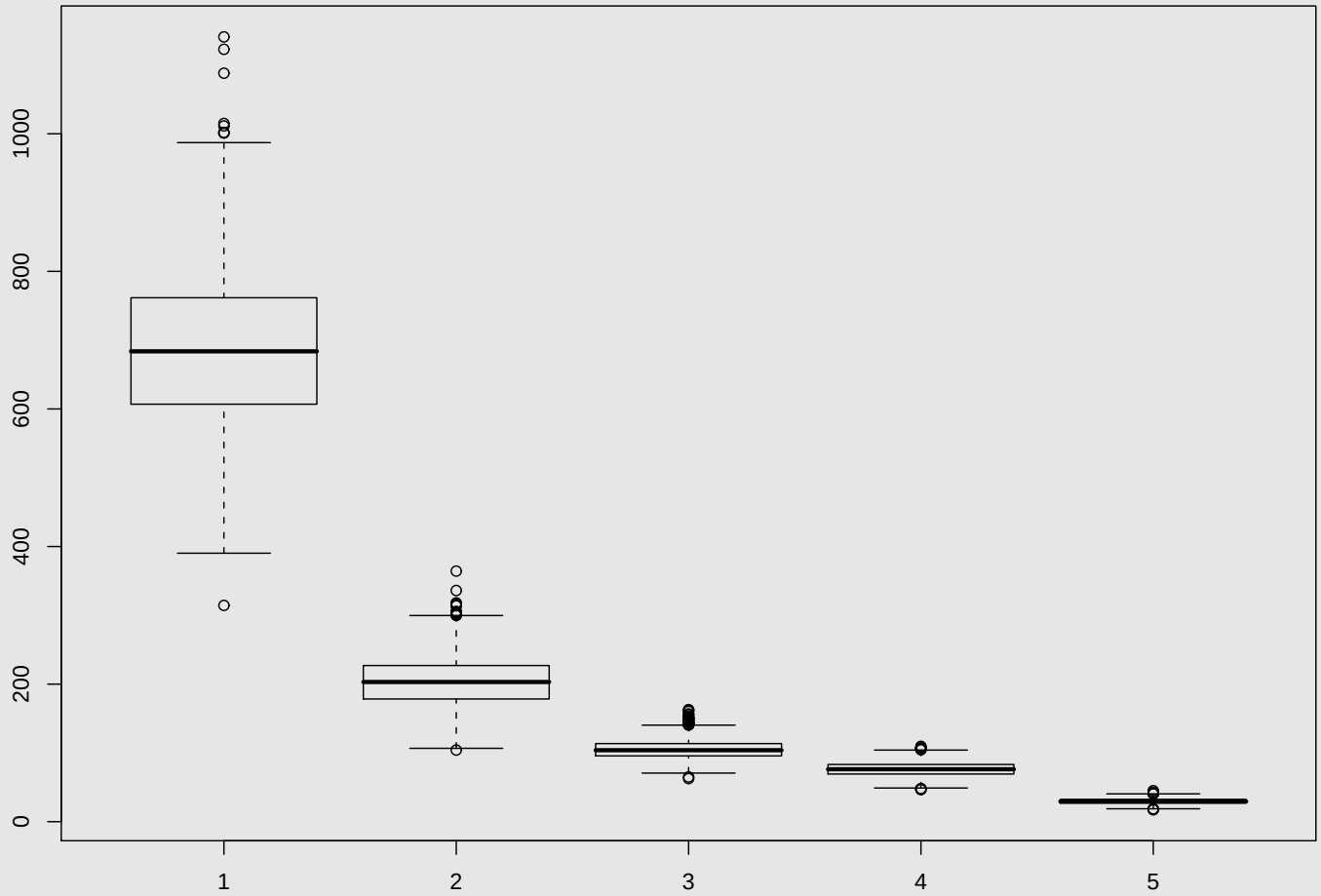
Example (Education test scores)



The Bootstrap: Principal components

Example (Education test scores)

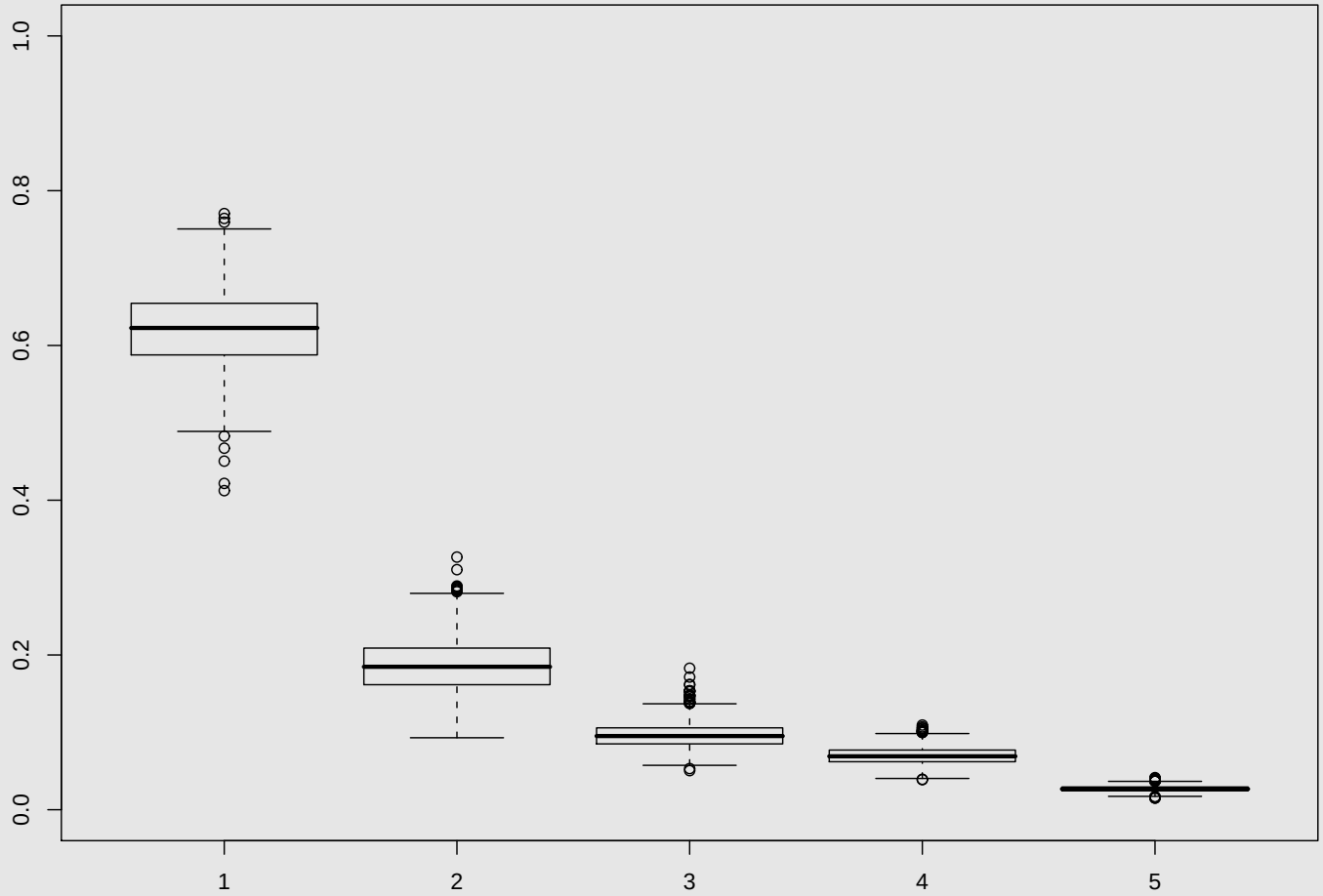
2000 resampled eigenvalues, $\lambda_j, j = 1, \ldots, 5$



The Bootstrap: Principal components

Example (Education test scores)

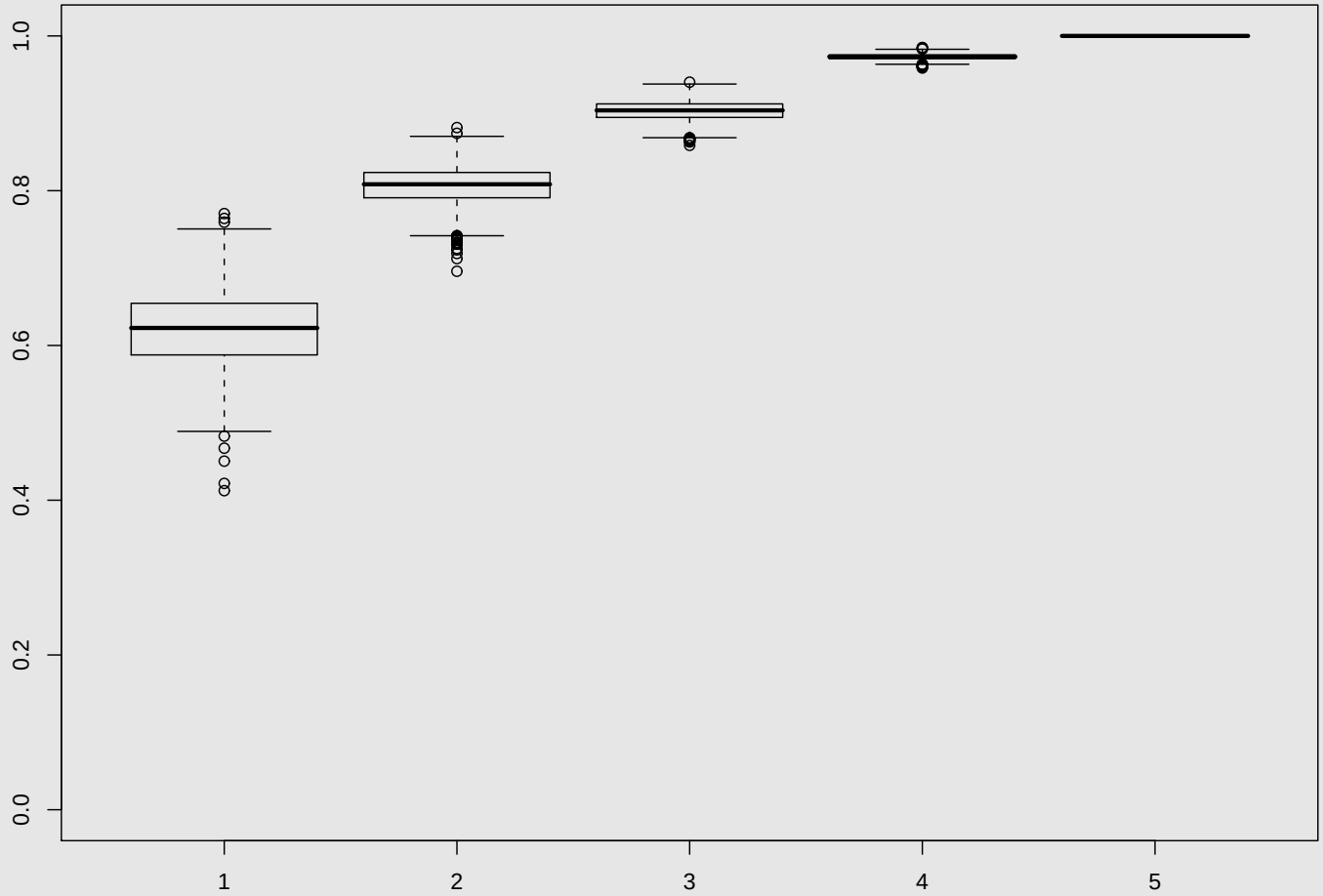
2000 resampled scaled eigenvalues, $\theta_j, j = 1, \ldots, 5$



The Bootstrap: Principal components

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

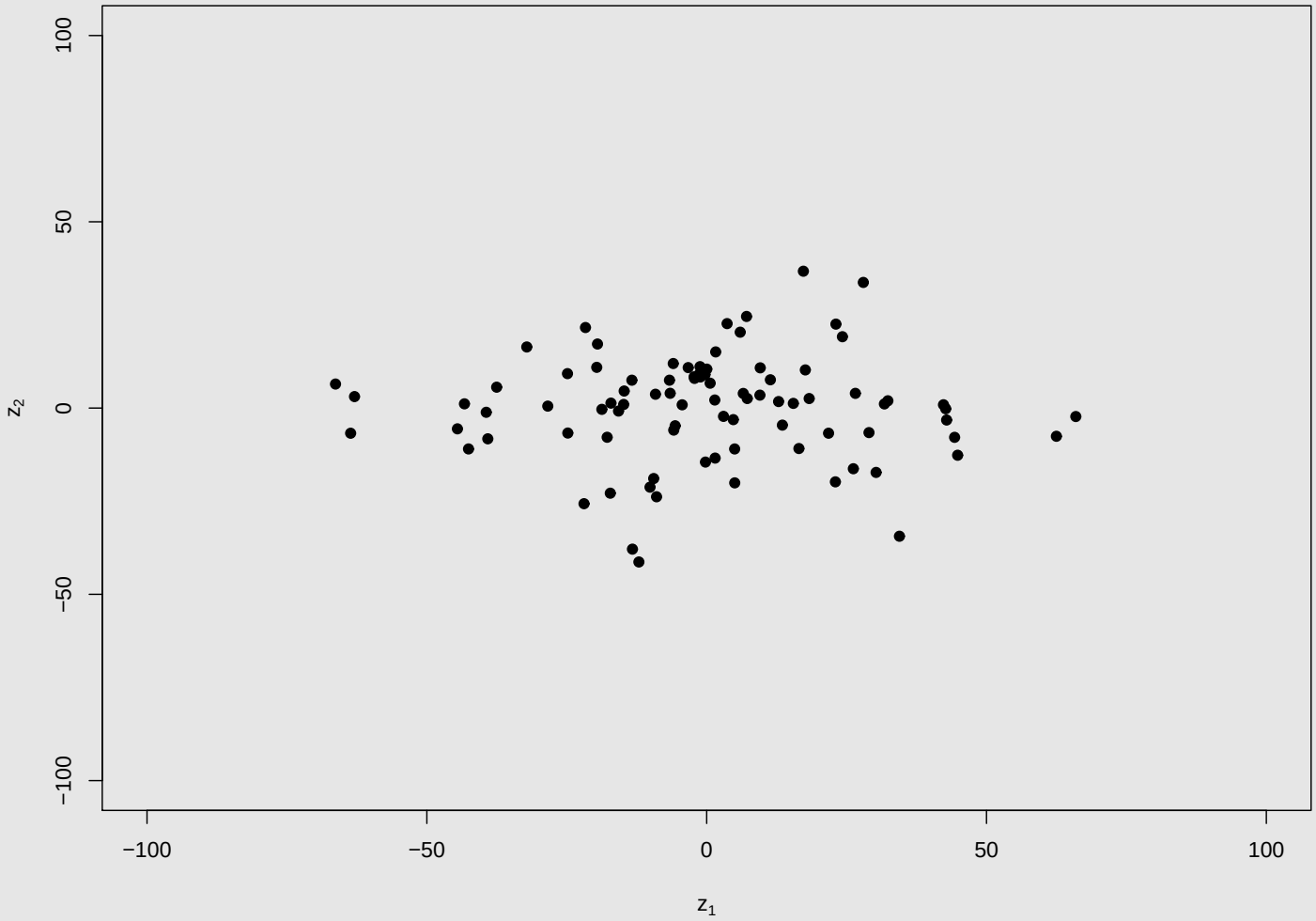
$$z_j = Xv_j \qquad j = 1, \ldots, p$$

where v_j is the jth column of V. The first two principal components carry greater than 80% of the variation observed.

The Bootstrap: Principal components

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 parameters 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.

In this case, *pointwise* confidence intervals are easy to obtain. However, *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 $j = 1, \ldots, p$

$$g_j(x) = \begin{cases} \cos(2\pi x \phi_l) & j = 2l - 1 \\ \sin(2\pi x \phi_l) & j = 2l \end{cases}$$

for known constants $\phi_1, \ldots, \phi_q$.

For example, a typical choice is $q = 2$, with

$$\phi_1 = \frac{1}{2} \qquad \phi_2 = \frac{1}{4}$$

The Bootstrap: Regression

In such a model, the ML estimator is

$$\widehat{\beta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{Y}$$

where $\boldsymbol{X}$ 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

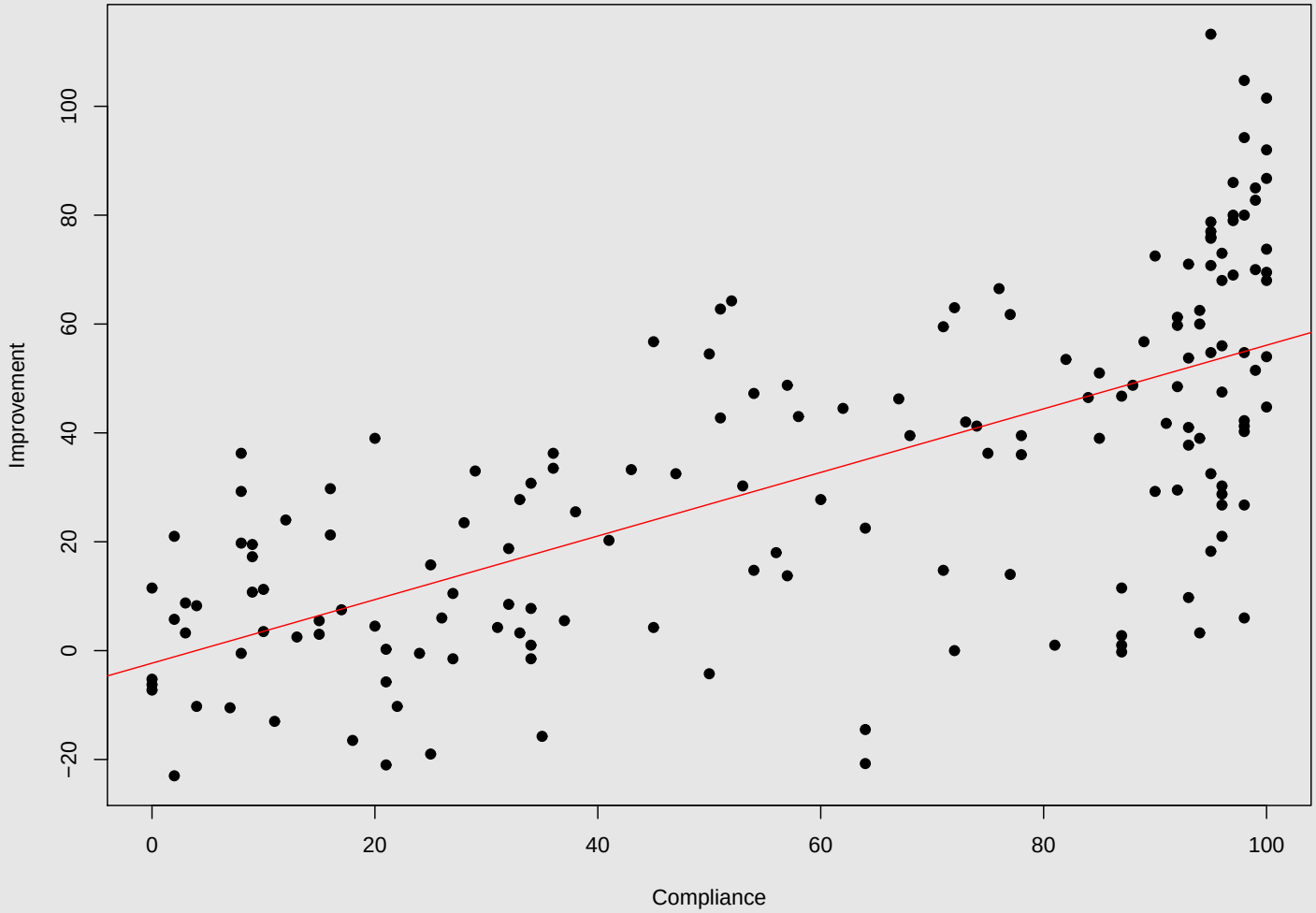
$$\widehat{\sigma}^2 (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}$$

and pointwise confidence intervals can be obtained in the usual way.

The Bootstrap: Regression

Example (Cholostyramine Data)

$n = 164$ men took part in an experiment to see if the drug cholostyramine lowered blood cholesterol levels.



The Bootstrap: Regression

Example (Cholostyramine Data)

Consider the model

$$g(x;\beta) = \beta_{00} + \beta_{01}x + \sum_{j=1}^{q}\left[\beta_{j0}\cos(2\pi x\phi_l) + \beta_{j1}\sin(2\pi x\phi_l)\right]$$

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 $\boldsymbol{X}_{(b)}$.
- fitting the regression to the resampled data obtaining

$$\widehat{\beta}^{(b)} = (\boldsymbol{X}_{(b)}^{\intercal}\boldsymbol{X}_{(b)})^{-1}\boldsymbol{X}_{(b)}^{\intercal}\boldsymbol{y}_{(b)}$$

- recording the fitted values

$$\hat{y}^{(b)} = \boldsymbol{X}_{(b)}\widehat{\beta}^{(b)}$$

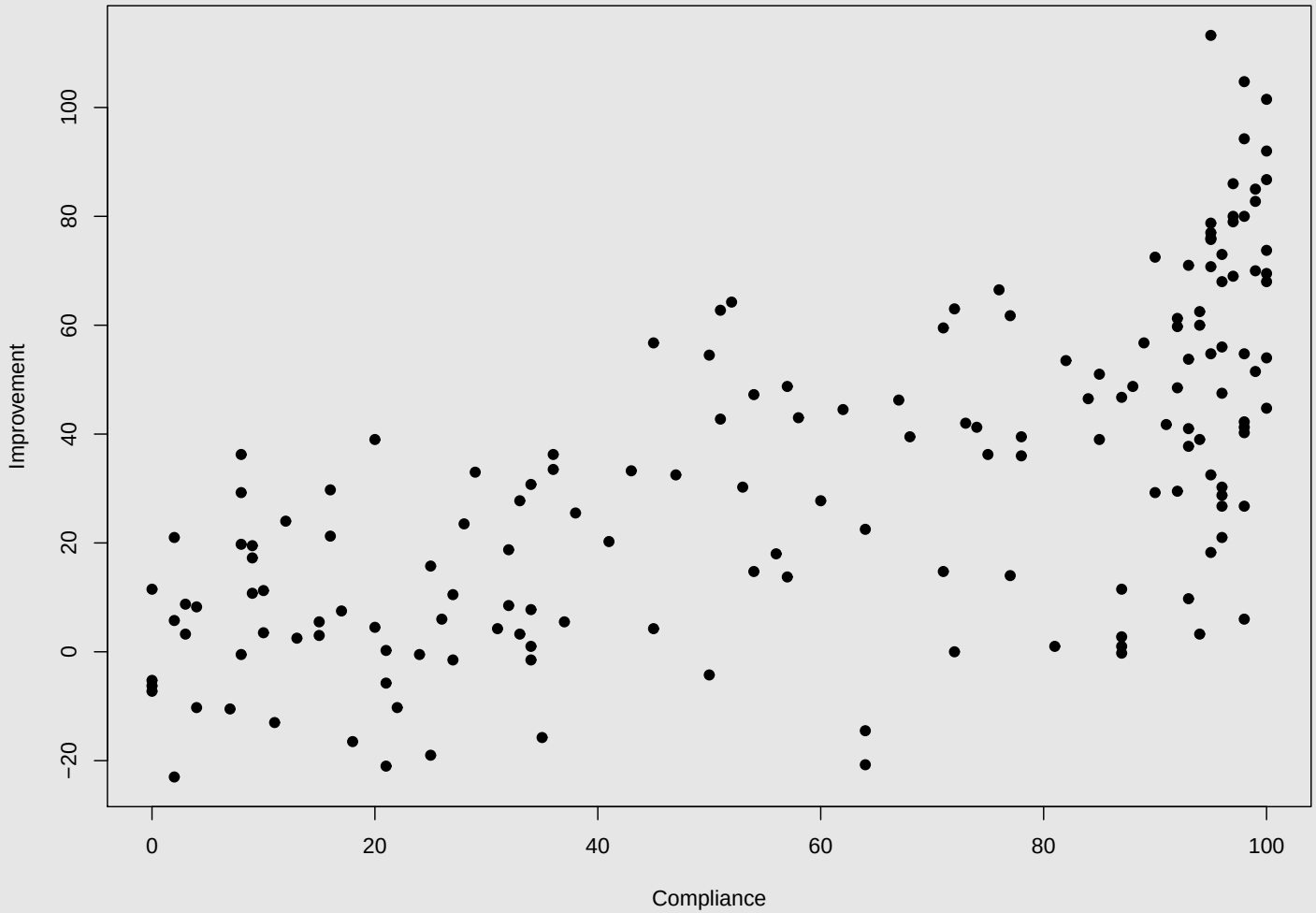
for $b = 1, \ldots, B$.

Again it is the i.i.d. units that are being resampled.

The Bootstrap: Regression

Example (Cholostyramine Data)

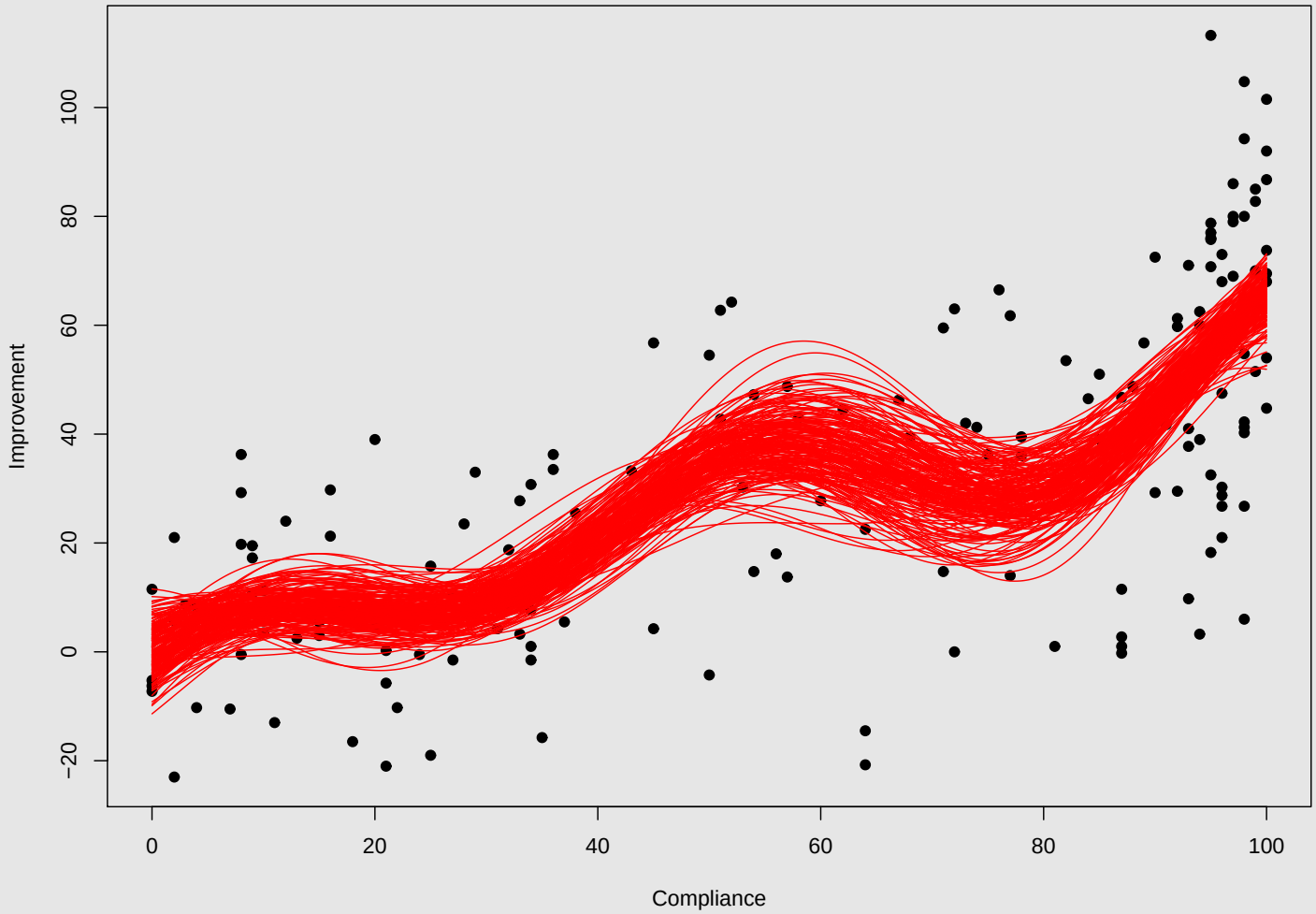
$B = 200$ bootstrapped curves with $q = 2$.



The Bootstrap: Regression

Example (Cholostyramine Data)

$B = 200$ bootstrapped curves with $q = 2$.



The Bootstrap: Loess

A more flexible model is based on *local smoothing* using *loess*.

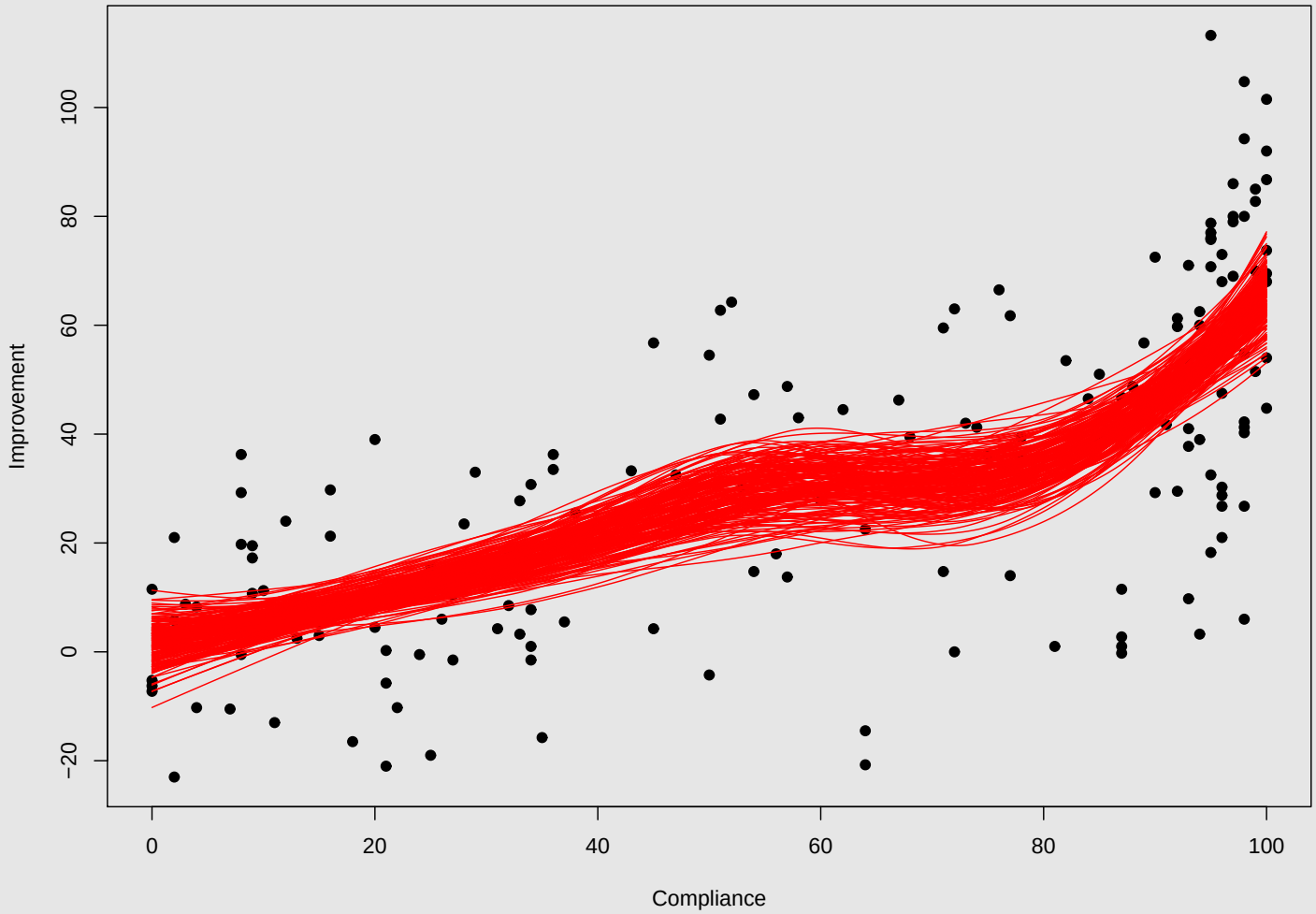
The loess approach fits a series of weighted local linear regression models restricted to a rolling neighbourhood of data points. The size of the neighbourhood (the *span*) determines the smoothness of the resulting fitted curve.

The statistical properties of the resulting fitted curve are hard to study analytically, but can be studied by the bootstrap.

The Bootstrap: Loess

Example (Cholostyramine Data)

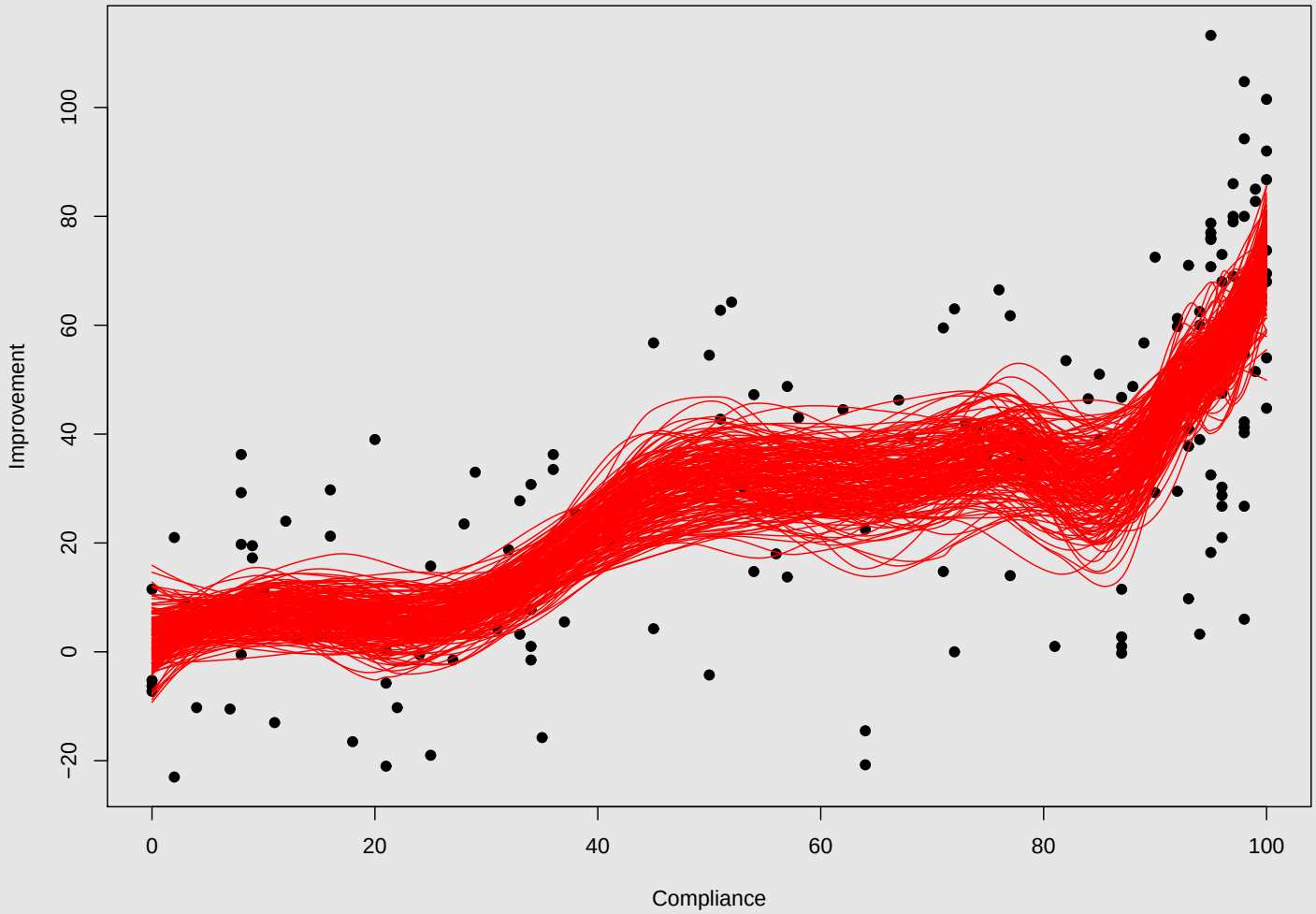
$B = 200$ bootstrapped loess curves: span is 0.75.



The Bootstrap: Loess

Example (Cholostyramine Data)

$B = 200$ bootstrapped loess curves: span is 0.375.



The Bootstrap: Loess

The choice of 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

$$\frac{1}{m}\sum_{i=1}^{m}(\widehat{y}_i - y_i)^2$$

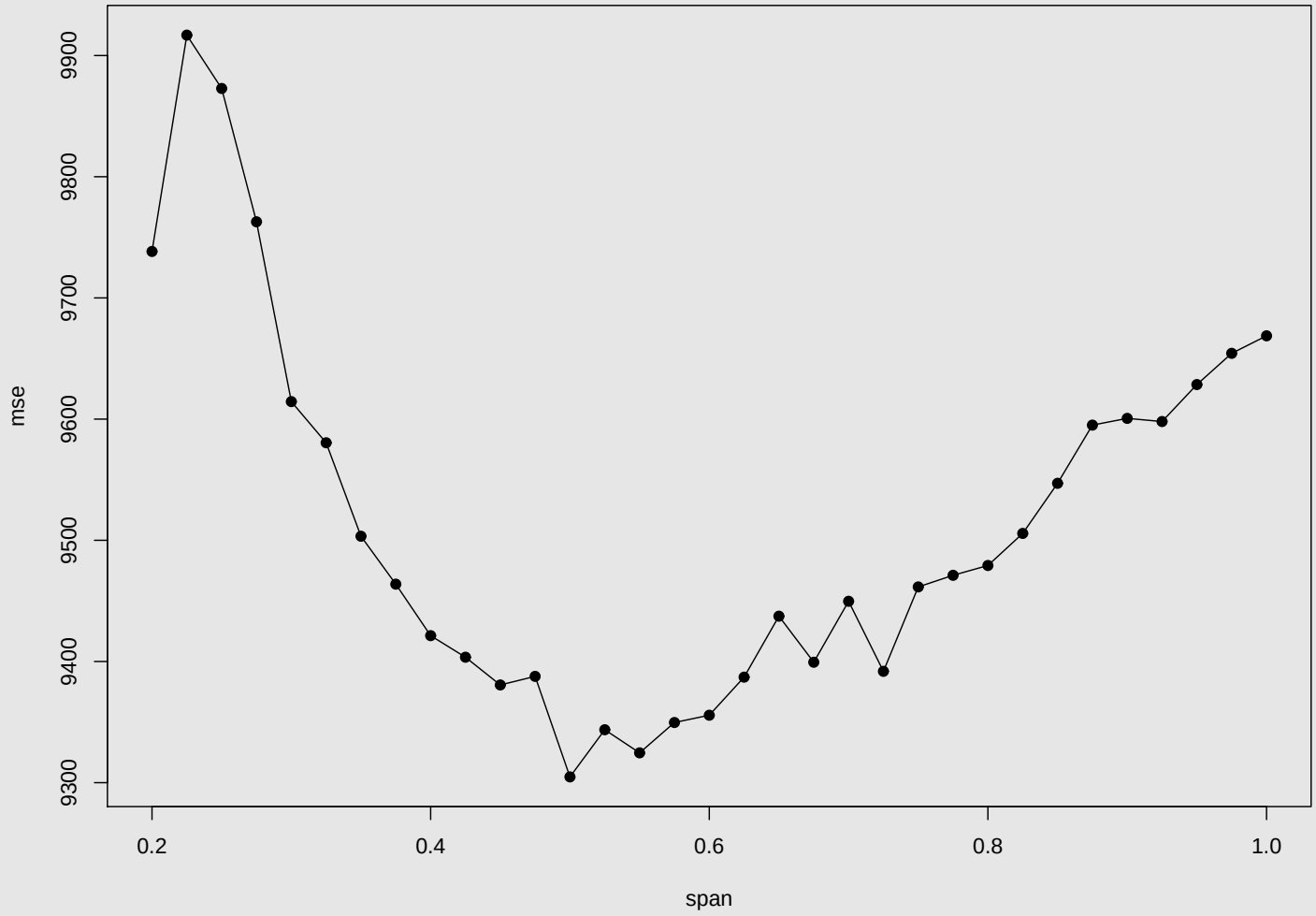
4. repeat 1. - 3. over different random splits, taking the average prediction MSE.

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)

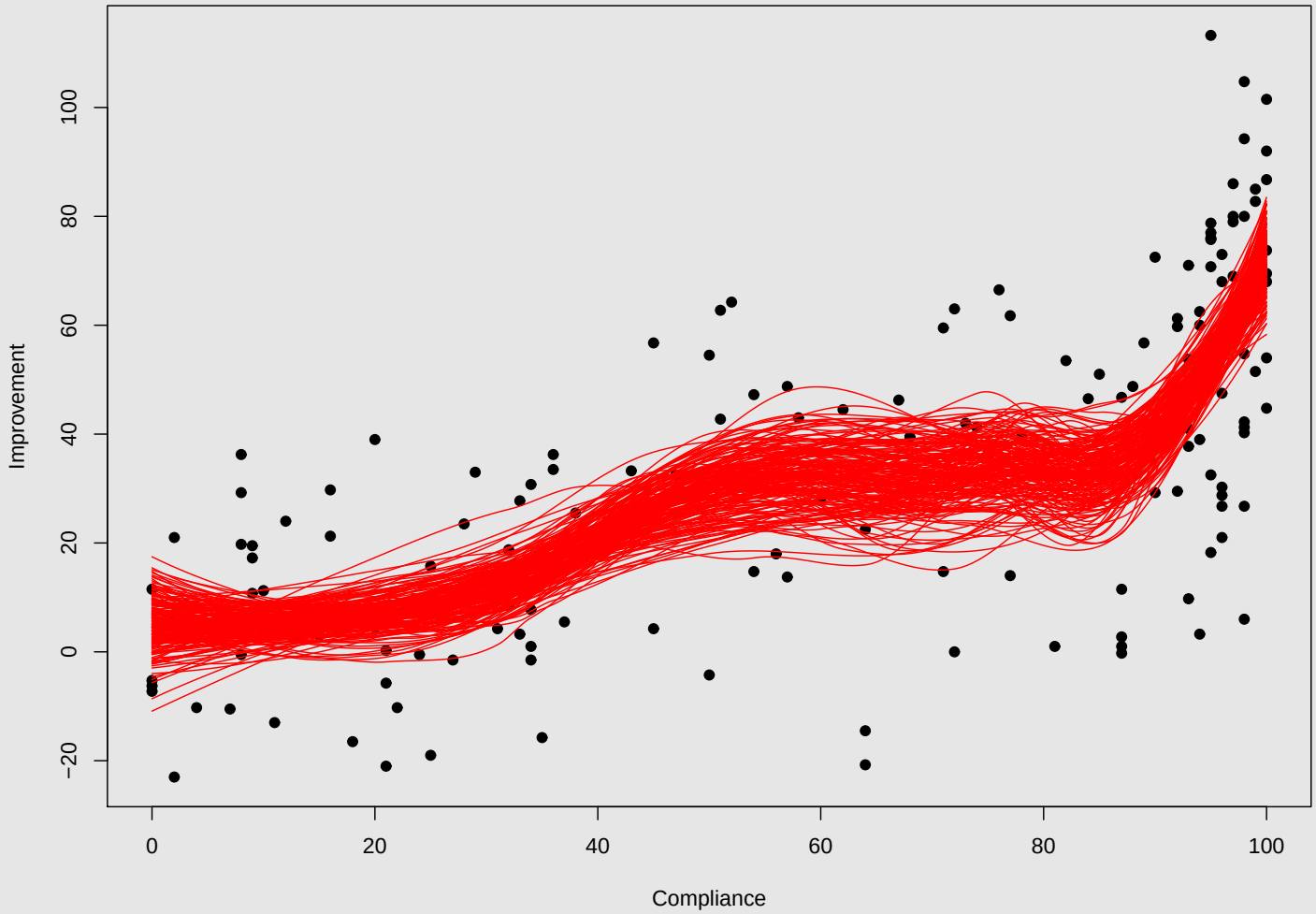
8-fold cross-validation: average prediction MSE from 20000 random splits.



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 $\boldsymbol{X}_{(b)}$ such that

$$\boldsymbol{X}_{(b)}^{\mathsf{T}}\boldsymbol{X}_{(b)}$$

is singular.

The Bootstrap: Problems

Suppose $Y_1, \ldots, Y_n \sim \mathcal{N}(0,1)$. Consider the statistic

$$T_n = \max\{Y_1, \ldots, Y_n\}$$

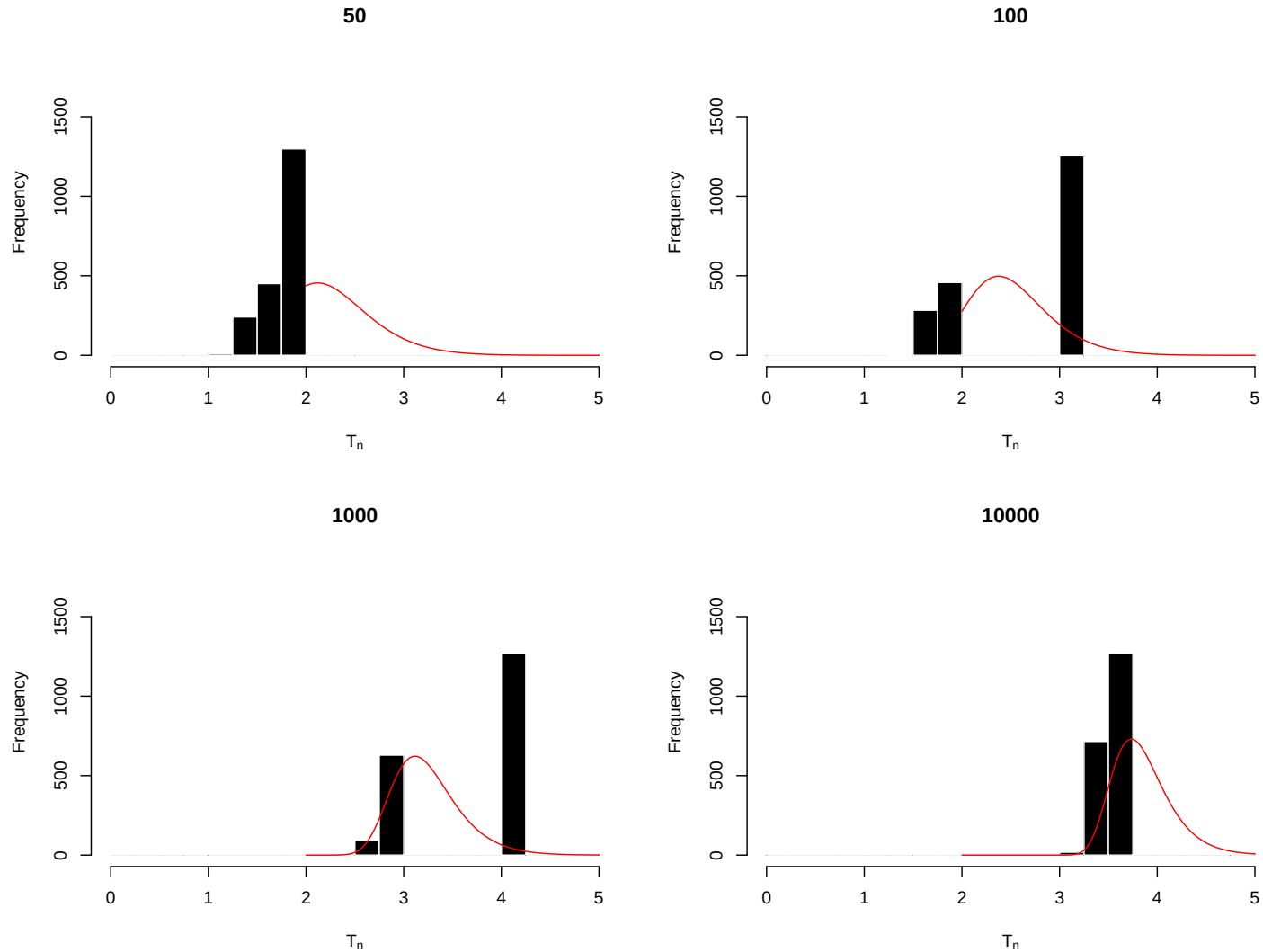
We know that for $t \in \mathbb{R}$

$$F_{T_n}(t) = \{\Phi(t)\}^n \qquad\qquad f_{T_n}(t) = n\phi(t)\{\Phi(t)\}^{n-1}$$

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 statistic T_n, which in this case is 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

$$\begin{aligned}\mathbb{P}[\widehat{\theta}^{(b)} = \widehat{\theta}] &= \mathbb{P}[\max\{Y_1^{(b)}, \ldots, Y_n^{(b)}\} = \max\{Y_1, \ldots, Y_n\}] \\ &= 1 - \left(1 - \frac{1}{n}\right)^n\end{aligned}$$

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 (or 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 $\{y_1, \ldots, y_n\}$ leads to problems as

$$\mathbb{P}[\widehat{\theta}^{(b)} = \widehat{\theta}] > 0$$

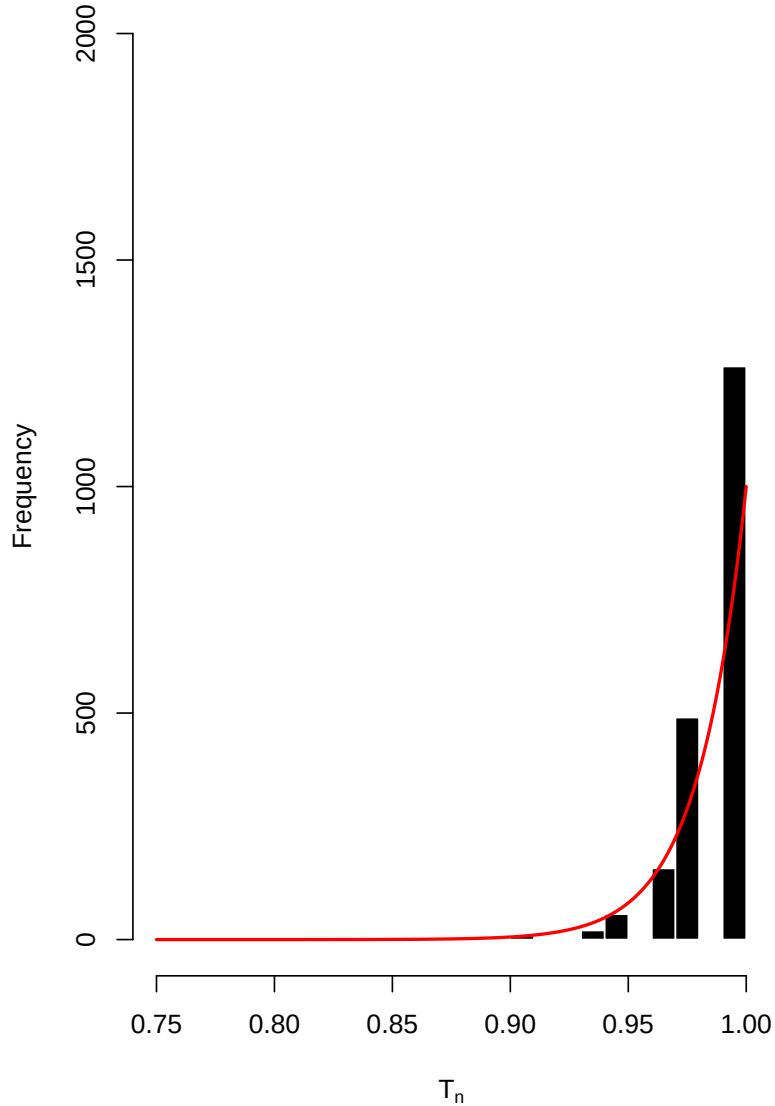
- **Parametric bootstrap:** sampling from $\mathit{Uniform}(0, \widehat{\theta})$ yields

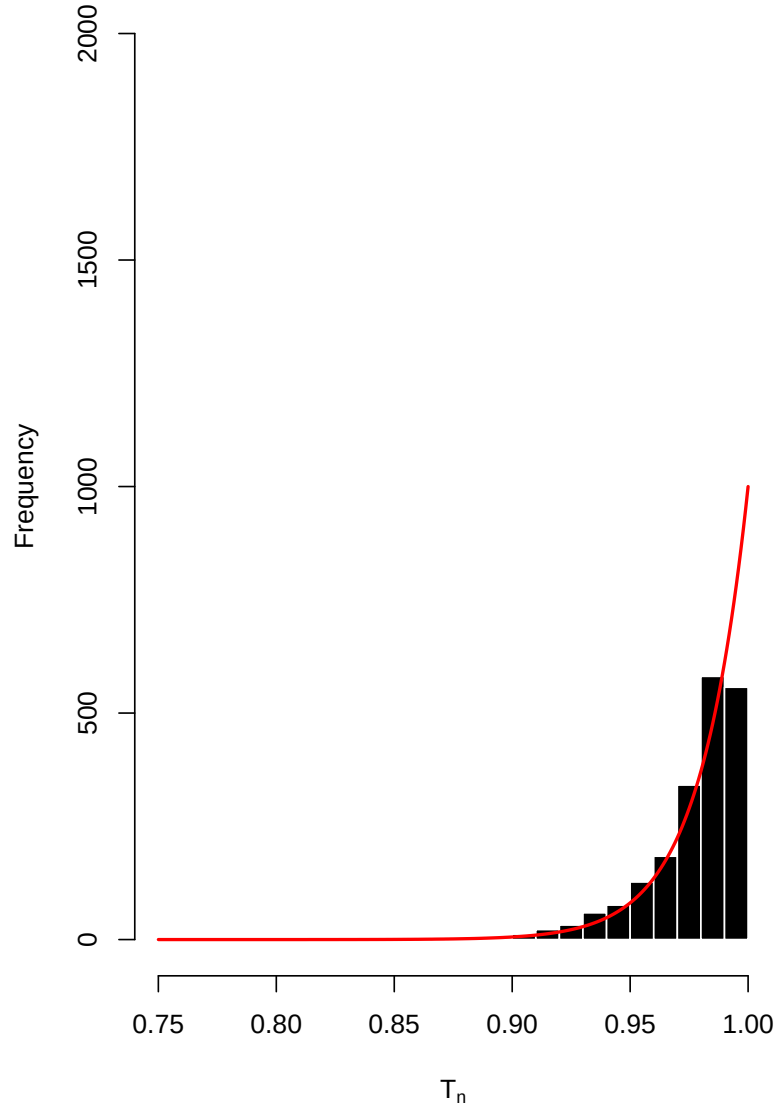
$$\mathbb{P}[\widehat{\theta}^{(b)} = \widehat{\theta}] = 0$$

  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

$$y_i = \boldsymbol{x}_i^\mathsf{T} \beta + \epsilon_i$$

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

$$\widehat{\beta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$$

and then the residual cdf F estimated by the empirical cdf of the residuals

$$e_i = y_i - \boldsymbol{x}_i^\intercal \widehat{\beta} \qquad i = 1, \ldots, n$$

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

$$y_i^{(b)} = \boldsymbol{x}_i^\intercal \beta + e_i^{(b)}$$

Bootstrap regression: alternative approach

This then yields the bootstrap estimate

$$\widehat{\beta}^{(b)} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}^{(b)}.$$

In this case, the idealized bootstrap estimate ($B = \infty$) of the standard error is available analytically via

$$\mathbb{V}ar[\widehat{\beta}^{(b)}] = \widehat{\sigma}^2 (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}$$

where

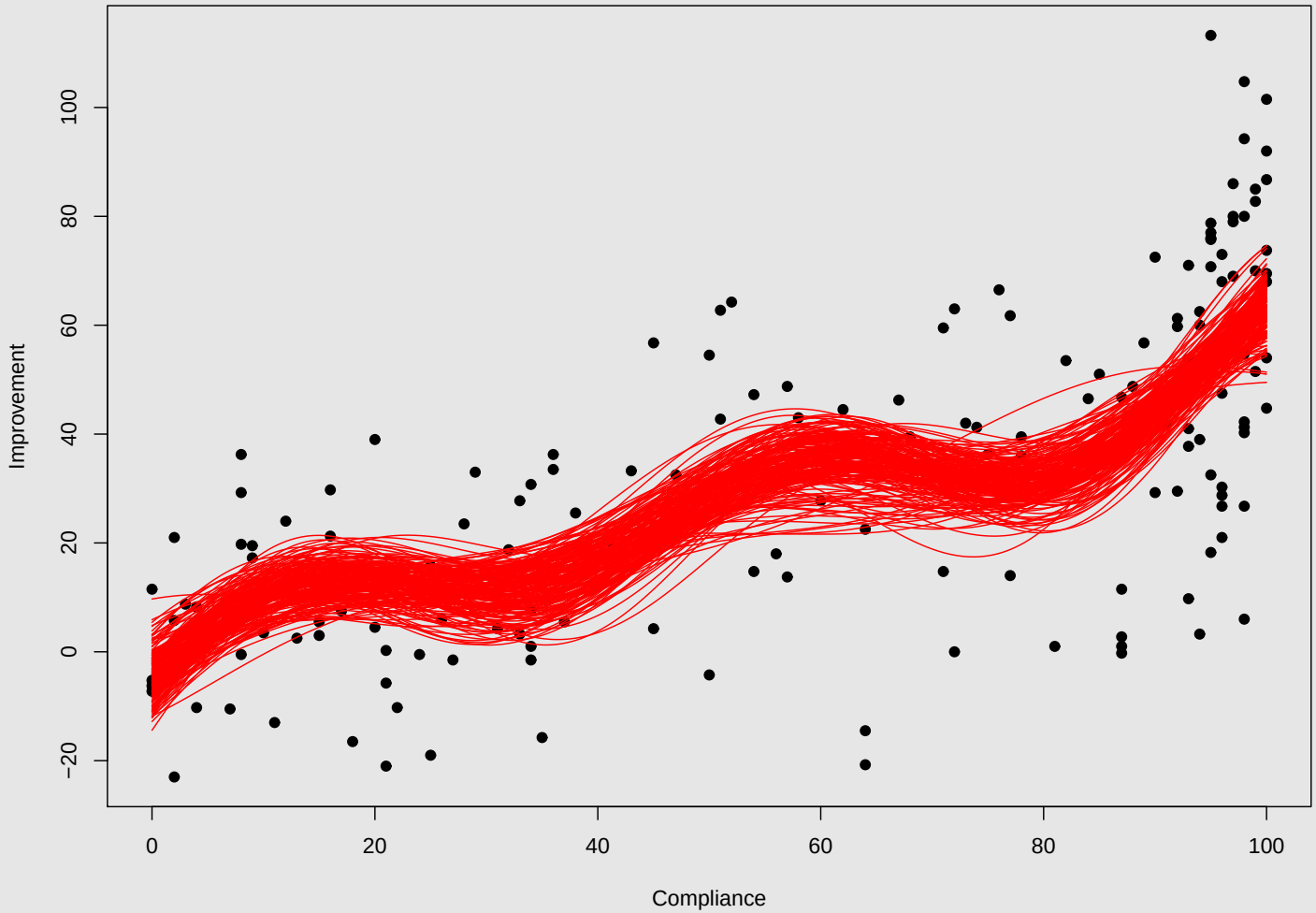
$$\widehat{\sigma}^2 = \frac{1}{n-p} \sum_{i=1}^{n} e_i^2$$

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

$$T_n^{(b)} = T_n(P^{(b)})$$

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 bloodstream measured.

ID	Placebo	Old Patch	New Patch	old-plac. z	new - old 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

$$t_n = \theta(\widehat{F}_n) = \frac{\overline{y}}{\overline{z}} = -0.0713.$$

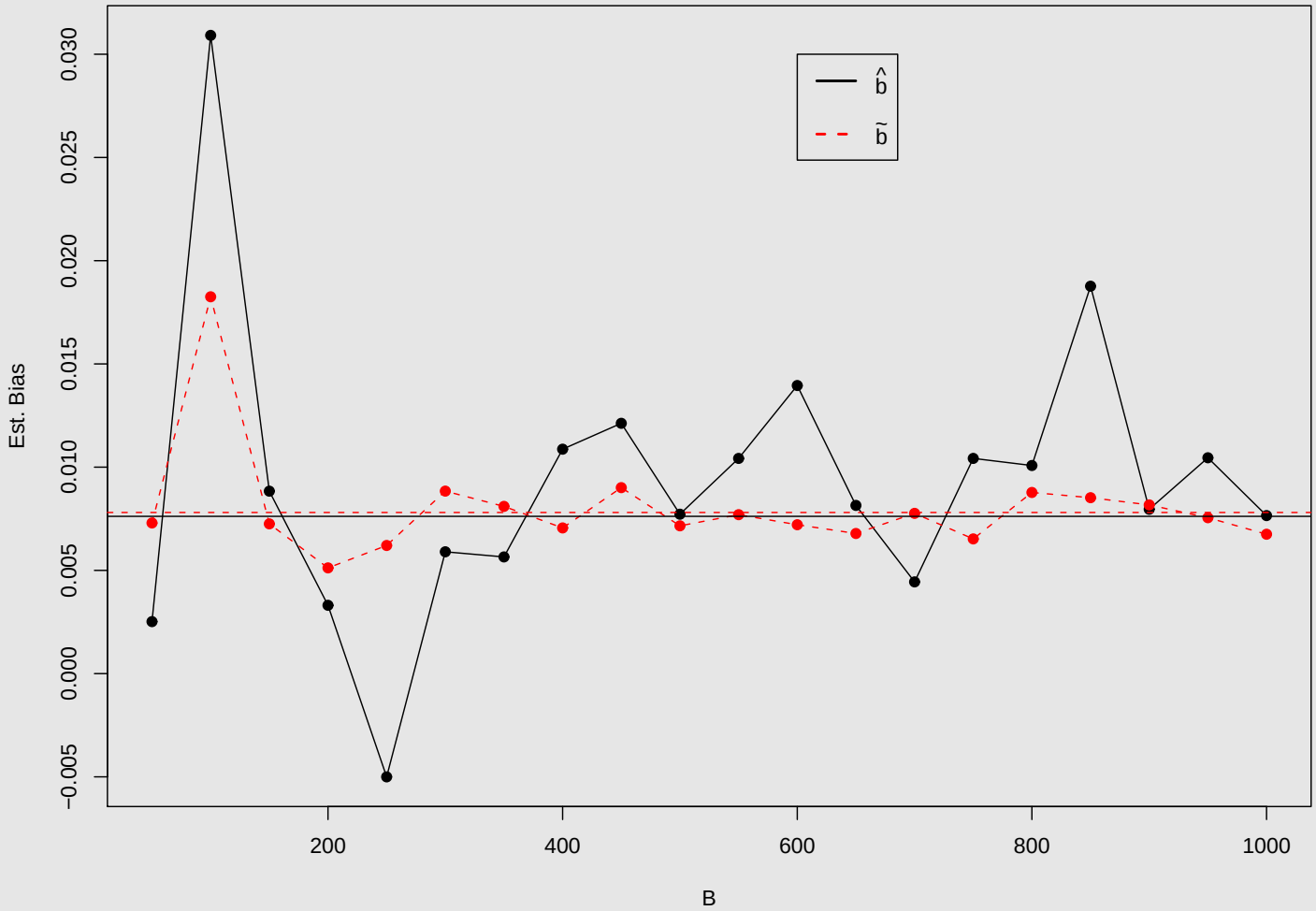
This parameter can be studied using the bootstrap, resampling individual subjects as the i.i.d. units.

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 $B = 100000$

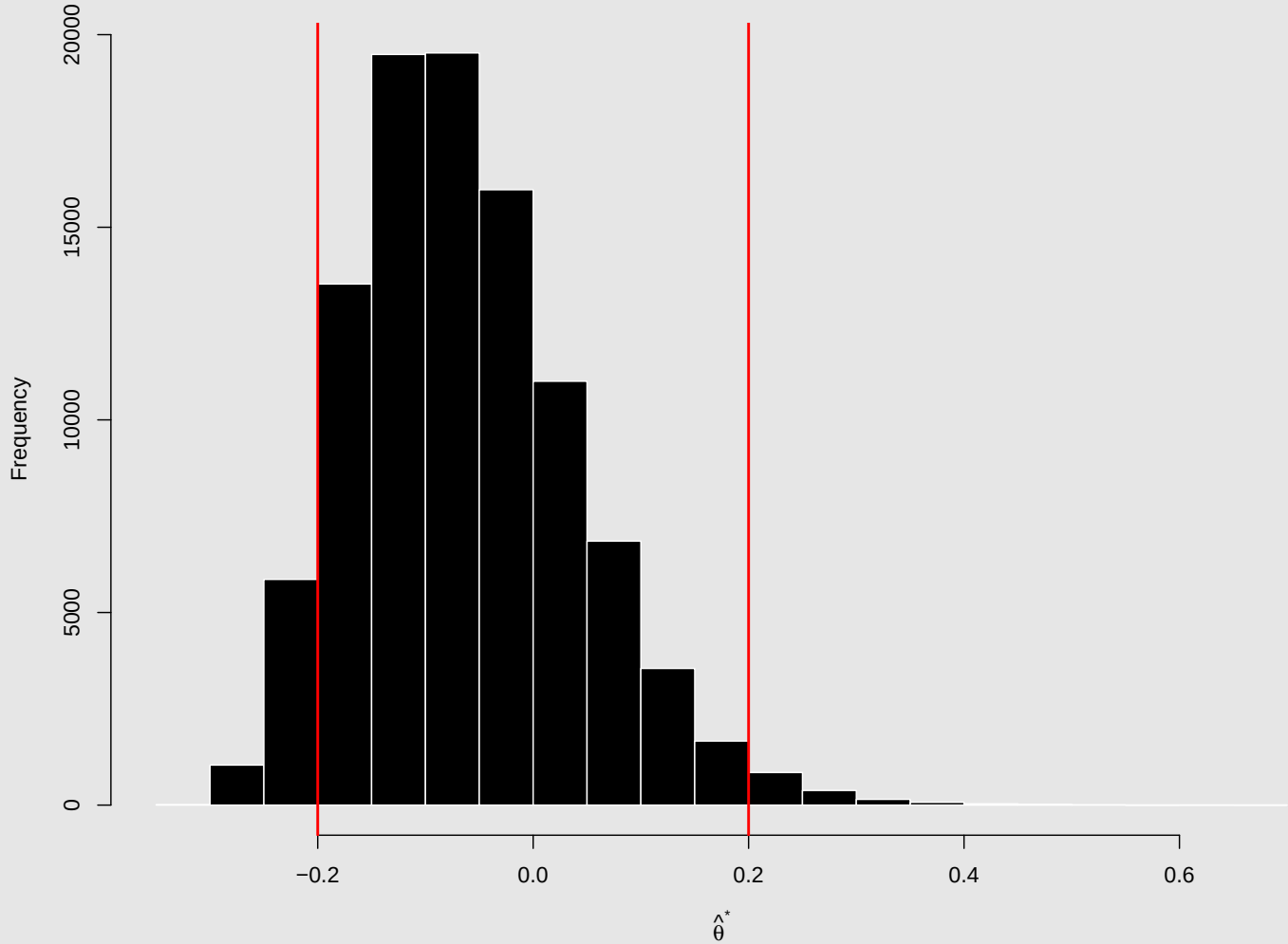
$$\widehat{b}_{T_n}(\widehat{F}_n) = 0.00762 \qquad \widetilde{b}_{T_n}(\widehat{F}_n) = 0.00781$$

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 $\widehat{\theta}$: $\mathbb{P}[|\widehat{\theta}| < 0.2] \simeq 0.9158$.



The jackknife

The *jackknife* was introduced as another bias estimation procedure. For $i = 1, \ldots, n$, let

$$\boldsymbol{y}_{(i)} = (y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n)$$

be the *i-deleted* data set (with the ith datum deleted). Let

$$\widehat{\theta}_{(i)} = t_{n-1}(\boldsymbol{y}_{(i)})$$

be the estimate derived from the i-deleted data, and let

$$\widehat{\theta}_{(.)} = \frac{1}{n} \sum_{i=1}^{n} \widehat{\theta}_{(i)}$$

A bias estimate for a plug-in statistic $\widehat{\theta}$ is then

$$\widehat{b}_{\text{JACK}} = (n-1)(\widehat{\theta}_{(.)} - \widehat{\theta})$$

The jackknife

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.

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.

The jackknife

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 above bias/s.e. formulae are derived under the assumption that the identity

$$\widehat{\theta} = t_n(\boldsymbol{y}) = \mu + \frac{1}{n}\sum_{i=1}^{n} \alpha(y_i)$$

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.

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,

$$\alpha = 0.10 \ : \ \Phi^{-1}(0.95) = 1.645 \qquad \alpha = 0.05 \ : \ \Phi^{-1}(0.975) = 1.960.$$

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

$$\widehat{\theta} \sim \mathcal{N}(\theta, \mathrm{se}^2).$$

Recall that this is a **frequentist** statement, conditional on the **known** true values θ and se. Then

$$Z = \frac{\widehat{\theta} - \theta}{\mathrm{se}} \sim \mathcal{N}(0, 1)$$

so that, for $c > 0$,

$$\mathbb{P}_\theta[-c \leqslant Z \leqslant c] = \Phi(c) - \Phi(-c) = 1 - 2\Phi(-c)$$

Choosing c such that $\Phi(c) = \alpha/2$, say $c = z_{\alpha/2}$, we have

$$\mathbb{P}_\theta \left[-z_{\alpha/2} \leqslant \frac{\widehat{\theta} - \theta}{\text{se}} \leqslant z_{\alpha/2} \right] = 1 - \alpha$$

which can be rewritten

$$\mathbb{P}_\theta \left[\widehat{\theta} - \text{se} \times z_{\alpha/2} \leqslant \theta \leqslant \widehat{\theta} + \text{se} \times z_{\alpha/2} \right] = 1 - \alpha$$

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

$$[\widehat{\theta} - \text{se} \times z_{\alpha/2}, \widehat{\theta} + \text{se} \times z_{\alpha/2}]$$

is justified asymptotically, but for finite n, the *Student-t* interval

$$[\widehat{\theta} - \text{se} \times t_{\alpha/2}(n-1), \widehat{\theta} + \text{se} \times t_{\alpha/2}(n-1)]$$

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 $\widehat{F}_n$.

Confidence Intervals and Hypothesis Testing

The corresponding 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.

Confidence Intervals and Hypothesis Testing

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

$$[\widehat{\theta} - \widehat{\text{se}} \times \widehat{l}_{\alpha/2}, \widehat{\theta} + \widehat{\text{se}} \times \widehat{u}_{\alpha/2}]$$

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 with a control group.

Control C	52	104	146	10	50	31	40	27	46
Treatment T	94	197	16	38	99	141	23		

Example: Mouse data

Example (Mouse data)

Bootstrap percentiles used for reporting CI for $\mathbb{E}[Y_C]$:

B	0.05	0.1	0.5	0.9	0.95
100	-4.6258	-4.1015	0.0382	1.2886	1.4606
200	-4.5560	-2.4595	-0.2170	1.0194	1.4346
500	-4.6962	-3.3398	-0.0873	1.2819	1.5853
1000	-4.4927	-2.3528	-0.0752	1.1252	1.5343
2000	-4.5014	-3.3380	-0.0562	1.1478	1.4993
5000	-4.6618	-3.5709	-0.0495	1.1760	1.5561
10000	-4.5987	-3.4164	-0.0828	1.1706	1.4984
20000	-4.6032	-3.3034	-0.0762	1.1406	1.4817
50000	-4.6337	-3.2448	-0.0710	1.1709	1.5248
100000	-4.6083	-3.2710	-0.0720	1.1603	1.5032
Normal	-1.6449	-1.2816	0.0000	1.2816	1.6449
Student	-1.8595	-1.3968	0.0000	1.3968	1.8595

Example: Mouse data

Example (Mouse data)

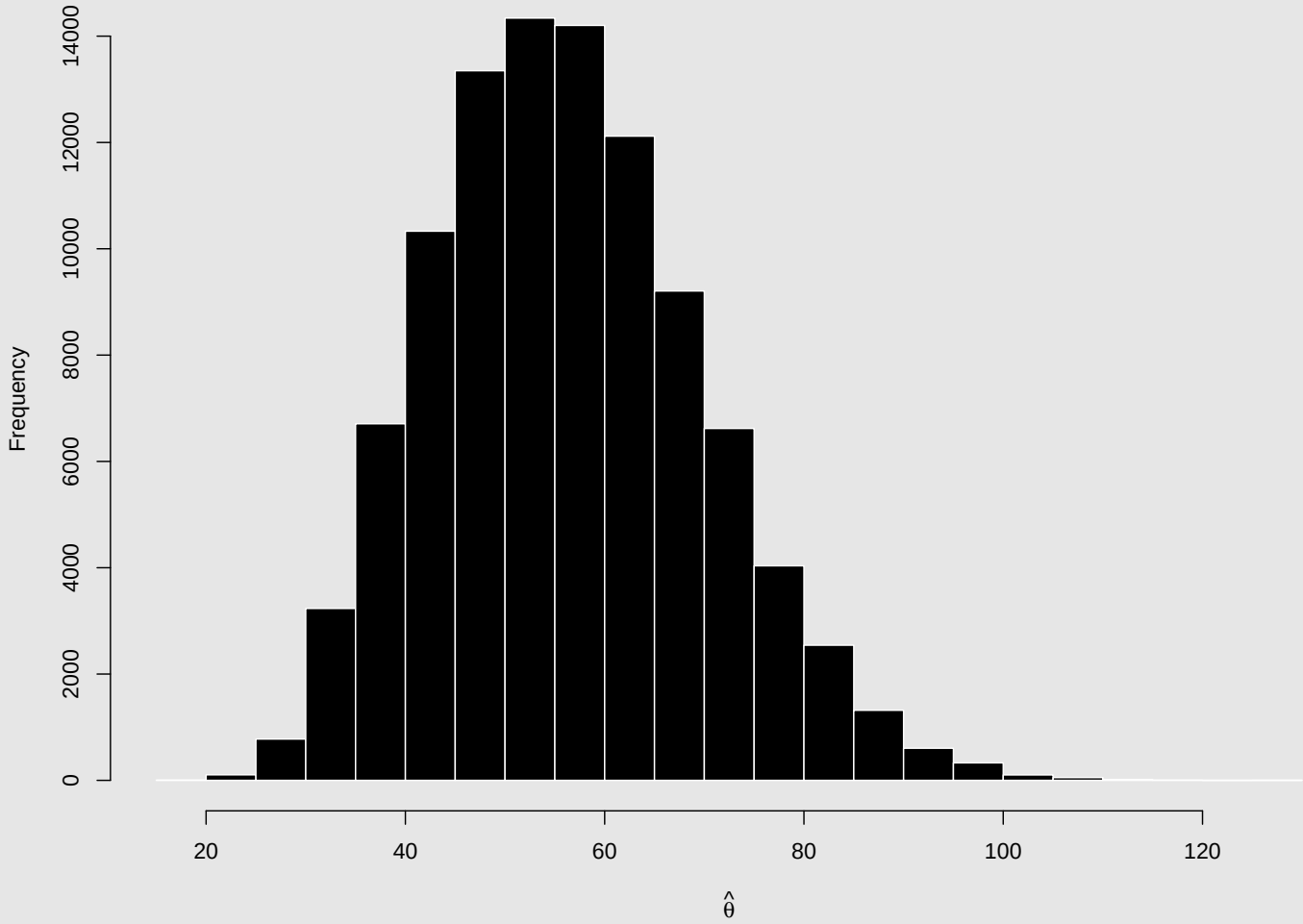
CI for $\mathbb{E}[Y_C]$:

B	Lower	Upper
100	-5.4411	75.6928
200	-4.5104	75.3462
500	-6.3801	77.3555
1000	-3.6664	76.6748
2000	-3.7825	76.2090
5000	-5.9217	76.9659
10000	-5.0803	76.1960
20000	-5.1401	75.9739
50000	-5.5468	76.5478
100000	-5.2086	76.2598
Normal	31.4338	81.0106
Student	34.2957	78.1487

Example: Mouse data

Example (Mouse data)

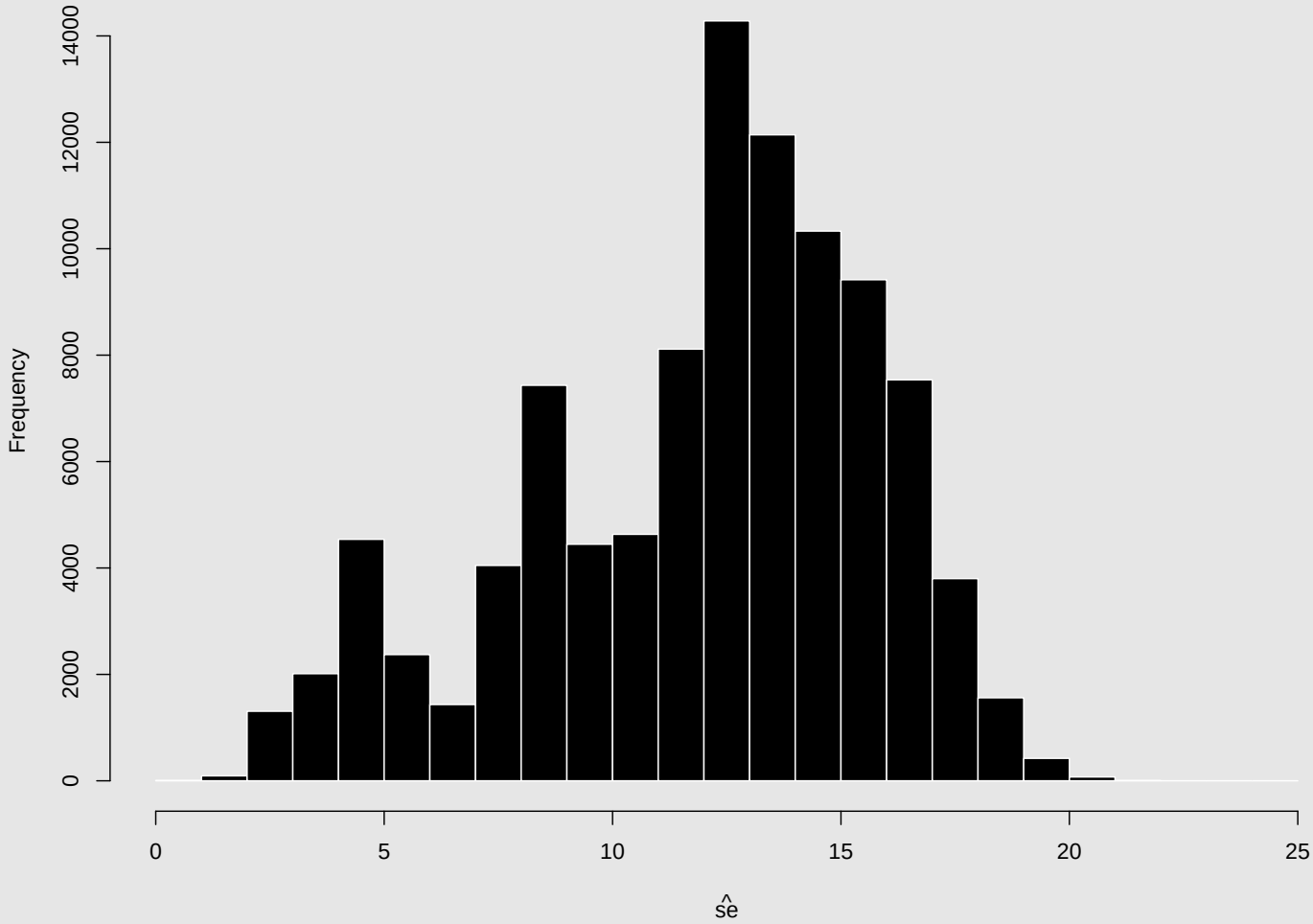
$\widehat{\theta}^{(b)}$:



Example: Mouse data

Example (Mouse data)

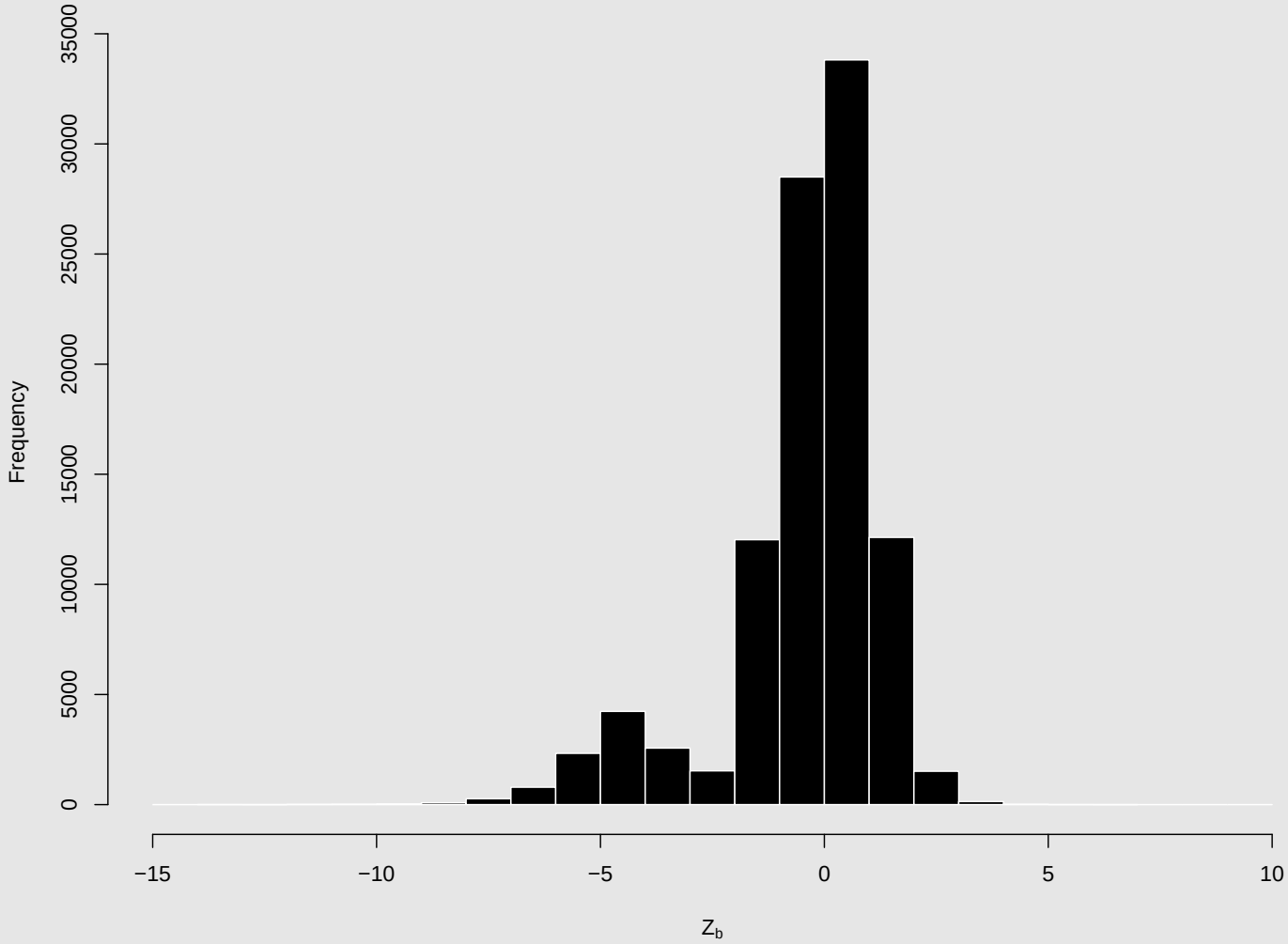
$\widehat{\text{se}}^{(b)}$:



Example: Mouse data

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 (specifically, 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

$$\mathbb{E}[X] = \theta \qquad \mathbb{V}ar[X] = s(\theta)^2$$

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

$$\mathbb{V}ar[Y] = \{\dot{g}(\theta)\}^2 s(\theta)^2 = 1$$

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.

Variance stabilizing transformations: Algorithm

1. Generate B_1 bootstrap samples, and bootstrap estimates $\widehat{\theta}^{(b)}$. For **each** bootstrap sample $\boldsymbol{y}^{(b)}$, 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
4. Use B_3 new bootstrap samples to compute a bootstrap-t interval for $\phi = g(\theta)$ by inspecting

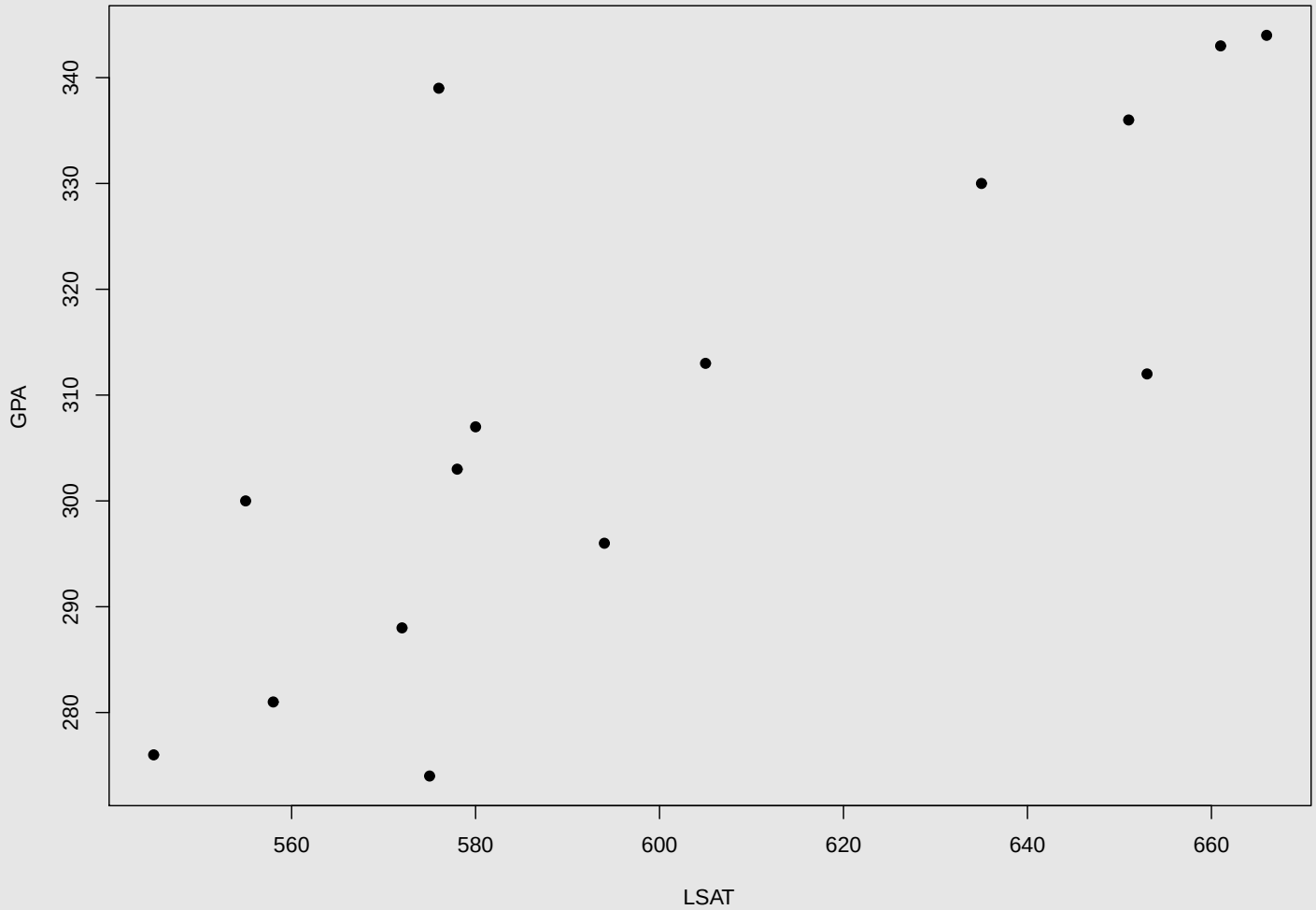
$$\widehat{\phi}^{(b)} - \widehat{\phi}$$

   as the variance has been stabilized to 1.
5. Back-transform to the θ scale using the numerically defined g^{-1}.

Variance stabilizing transformations: Example

Example (LSAT vs GPA data)

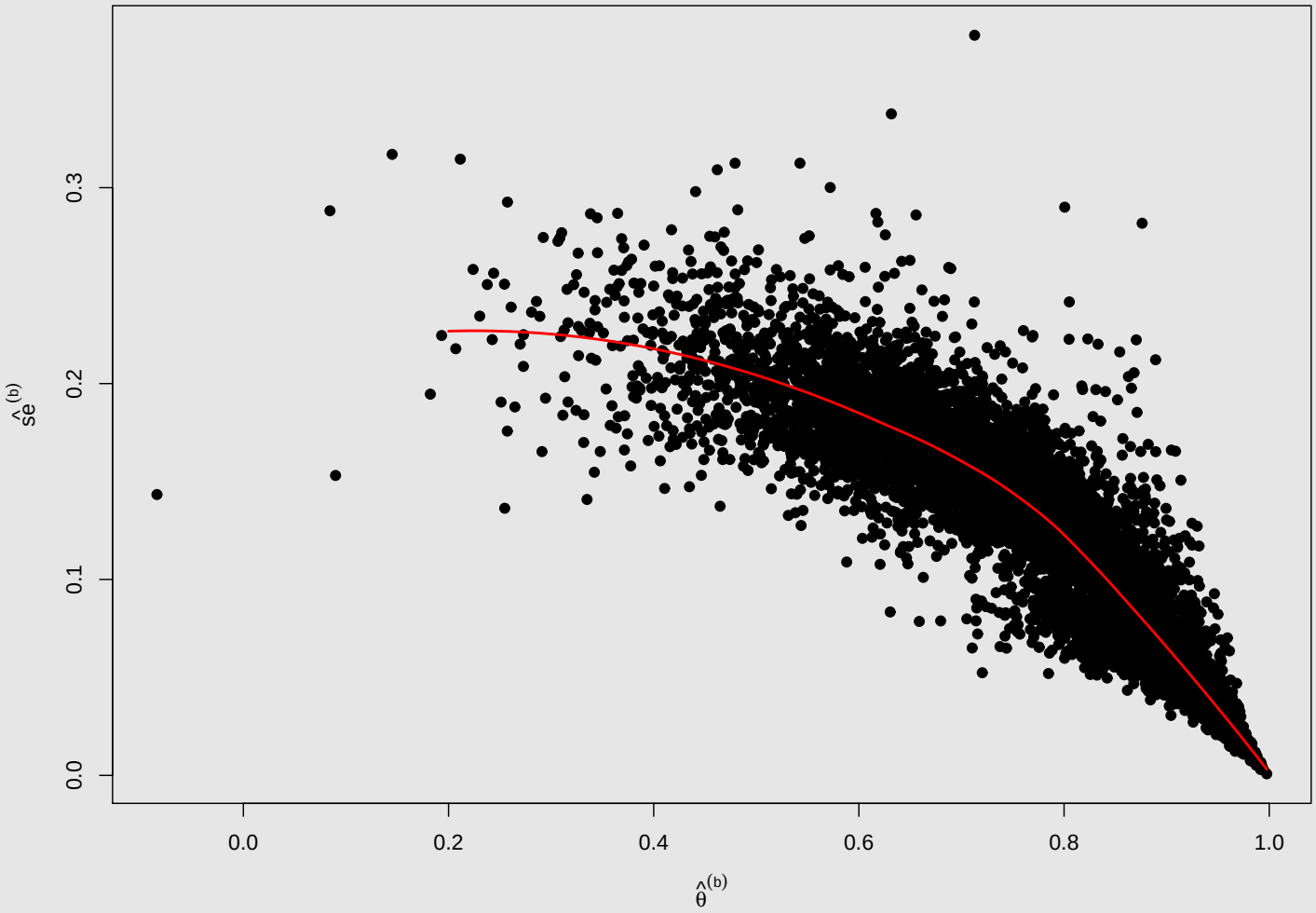
Raw data



Variance stabilizing transformations: Example

Example (LSAT vs GPA data)

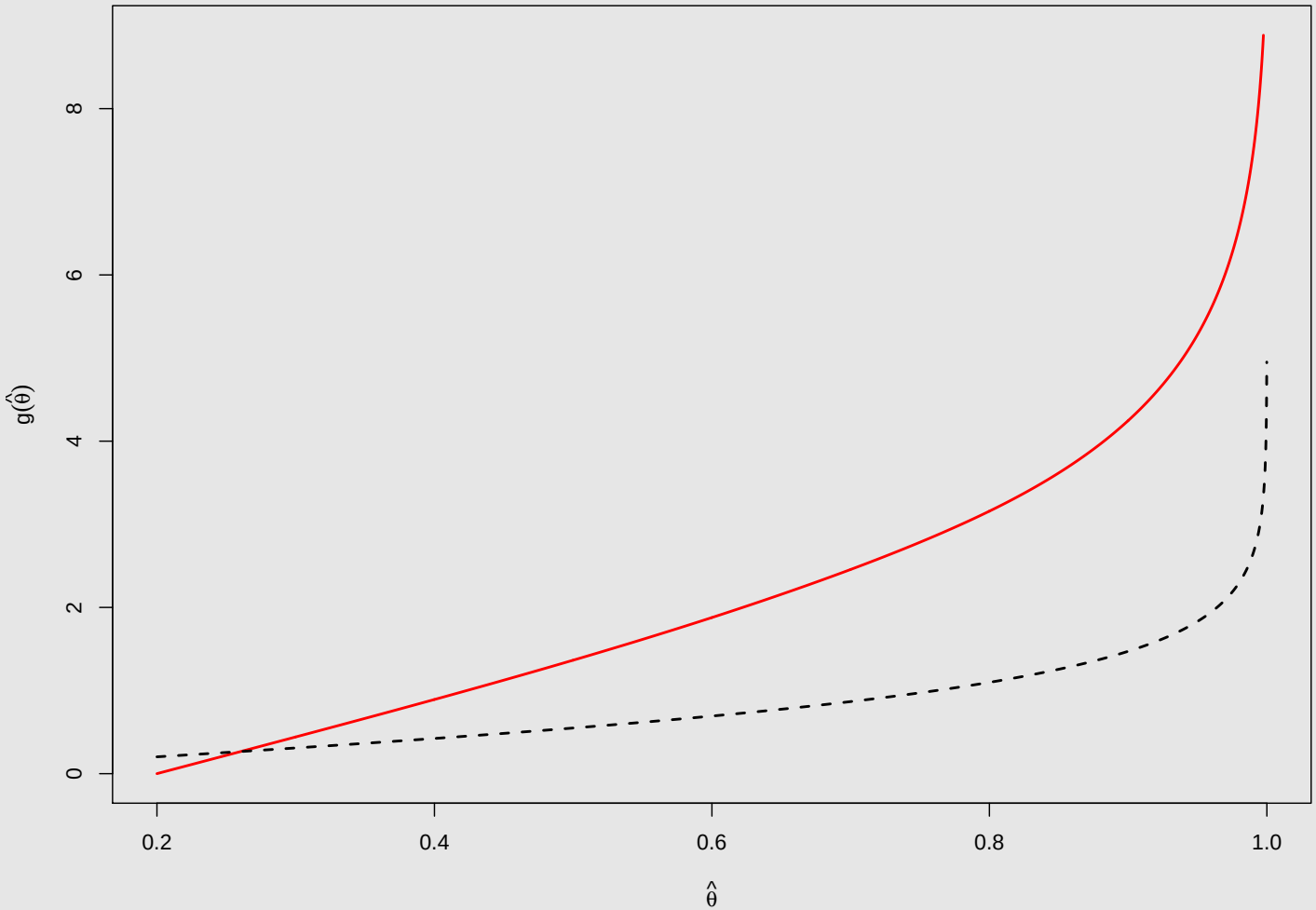
$B_1 = 10000$ bootstrap resamples with $B_2 = 500$ inner resamples.



Variance stabilizing transformations: Example

Example (LSAT vs GPA data)

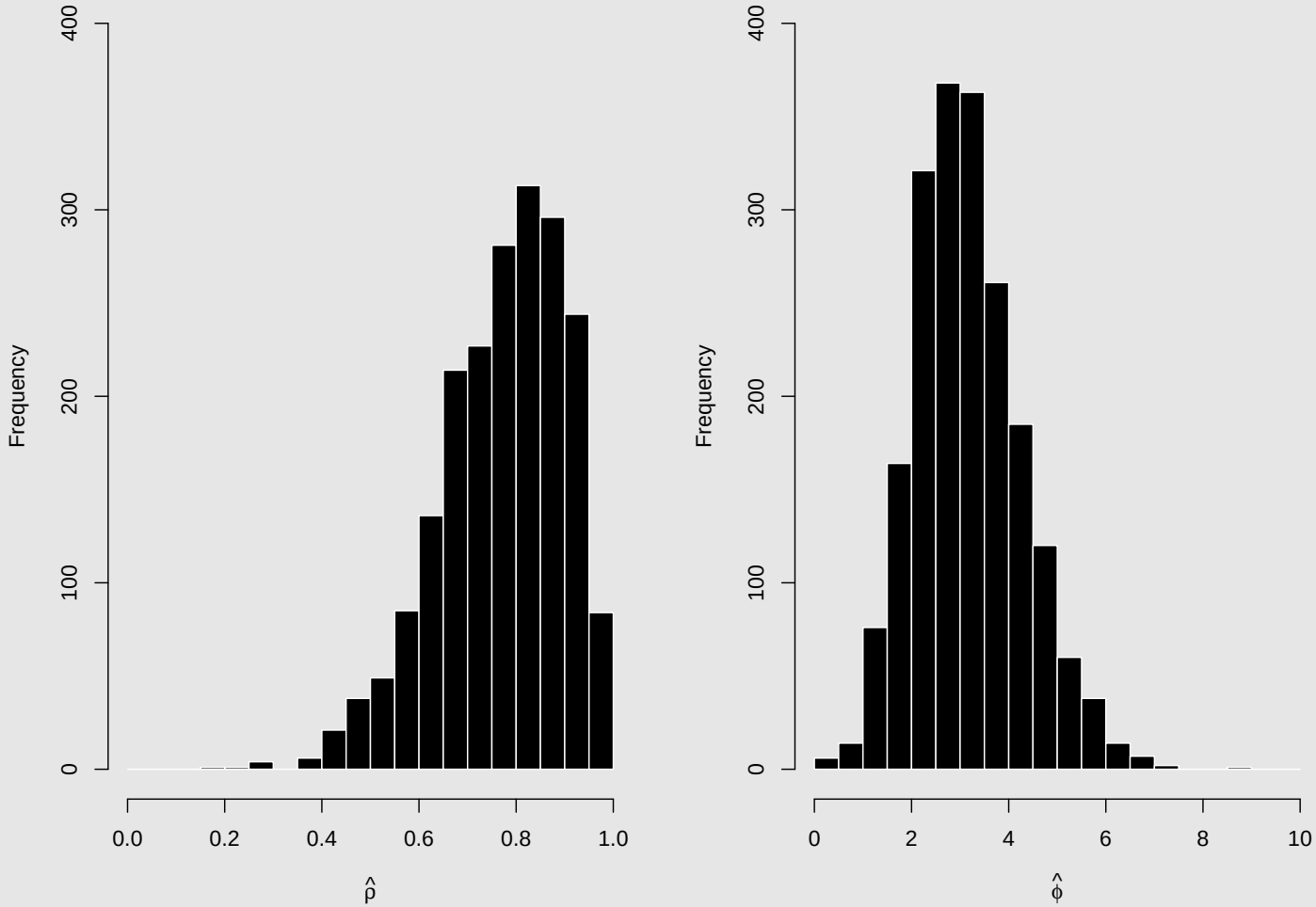
Estimated transform g (solid), and Fisher transform (dashed)



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 of the statistic of interest; it is a non-parametric version of the previous variance-stabilizing transform 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

$$\widehat{l}_{\alpha_1} \quad : \quad \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}_{\{\widehat{\theta}^{(b)} \leqslant \widehat{l}_{\alpha_1}\}} \backsimeq \alpha_1$$

$$\widehat{u}_{\alpha_2} \quad : \quad \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}_{\{\widehat{\theta}^{(b)} \leqslant \widehat{u}_{\alpha_2}\}} \backsimeq \alpha_2$$

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 $\widehat{\theta}^{(b)}$
- $\widehat{a}$ 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 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

Example (Mouse data)

Recall the mouse survival time data:

Control C (y)	52	104	146	10	50	31	40	27	46
Treatment 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.

Permutation tests: Example

Example (Mouse data)

We select a test statistic, $T = T(\boldsymbol{Y}, \boldsymbol{Z})$, 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.

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.

Permutation tests: Example

Example (Mouse data)

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; that is, for relabelling j, we choose **at random without replacement** n_Y measurements and denote them $\boldsymbol{y}_{(j)}$, denote the remaining n_Z measurements $\boldsymbol{z}_{(j)}$, 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.

Permutation tests: Example

Example (Mouse data)

The *critical values* for the test are defined via the empirical distribution of

$$t_{(1)}, \ldots, t_{(N)}$$

If $t = T(\boldsymbol{y}, \boldsymbol{z})$ is the observed test statistic for the original data, then empirical probabilities

$$\mathbb{P}_0[T_{(*)} \leqslant t] = \frac{1}{N} \sum_{j=1}^{N} \mathbb{1}_{[t_{(j)}, \infty)}(t)$$

can be used to define the p-value (*achieved significance level*) in the two-tailed test as

$$\mathbb{P}_0[T_{(*)} \leqslant -|t|] + \mathbb{P}_0[T_{(*)} \geqslant |t|]$$

Permutation tests: Computation

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: Computation

If the permutations are randomly chosen, the procedure is termed a *Monte Carlo permutation* (or *Monte Carlo randomization*) procedure.

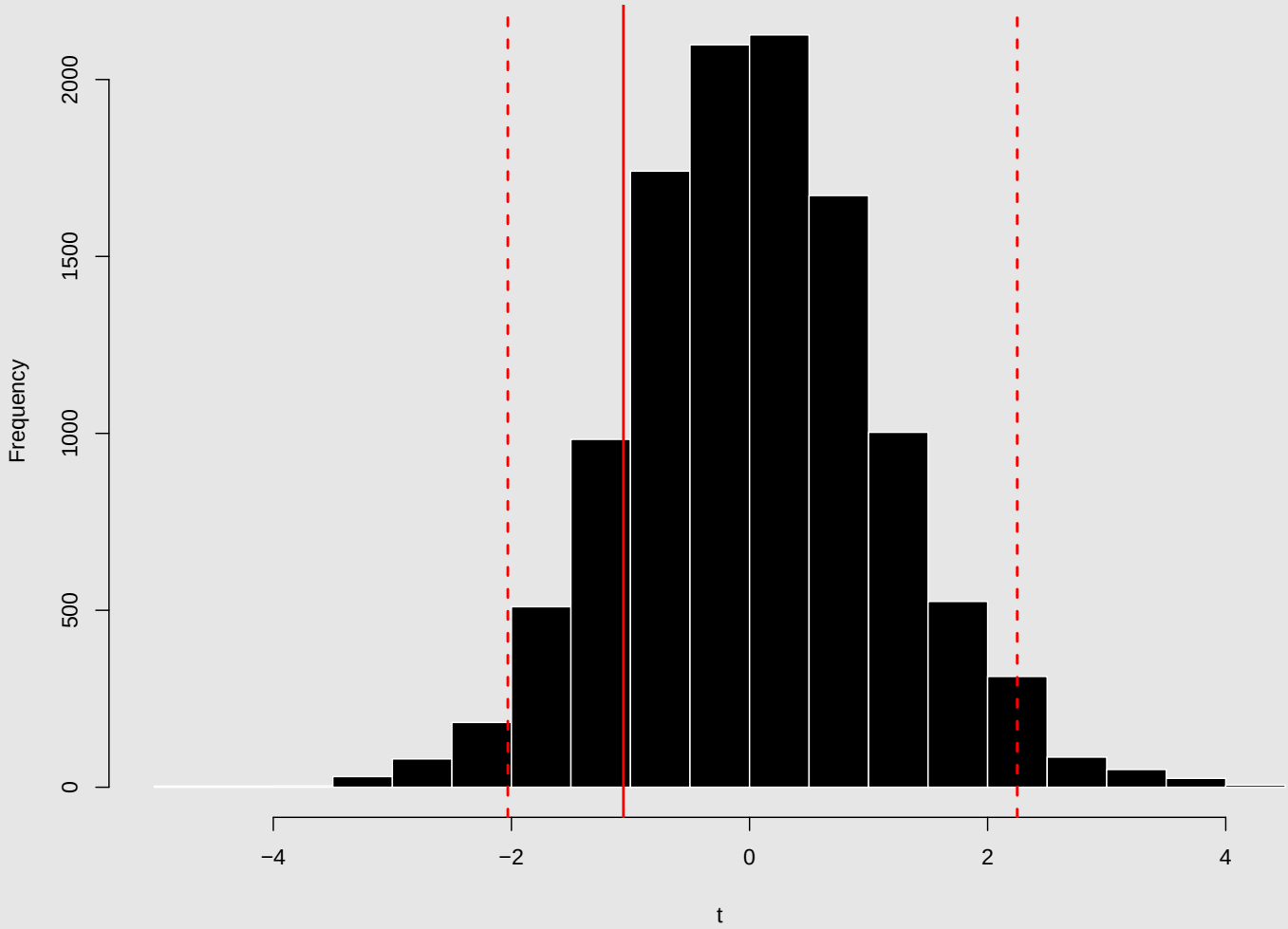
The choice of the Monte Carlo sample size depends on the quantity being estimated.

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

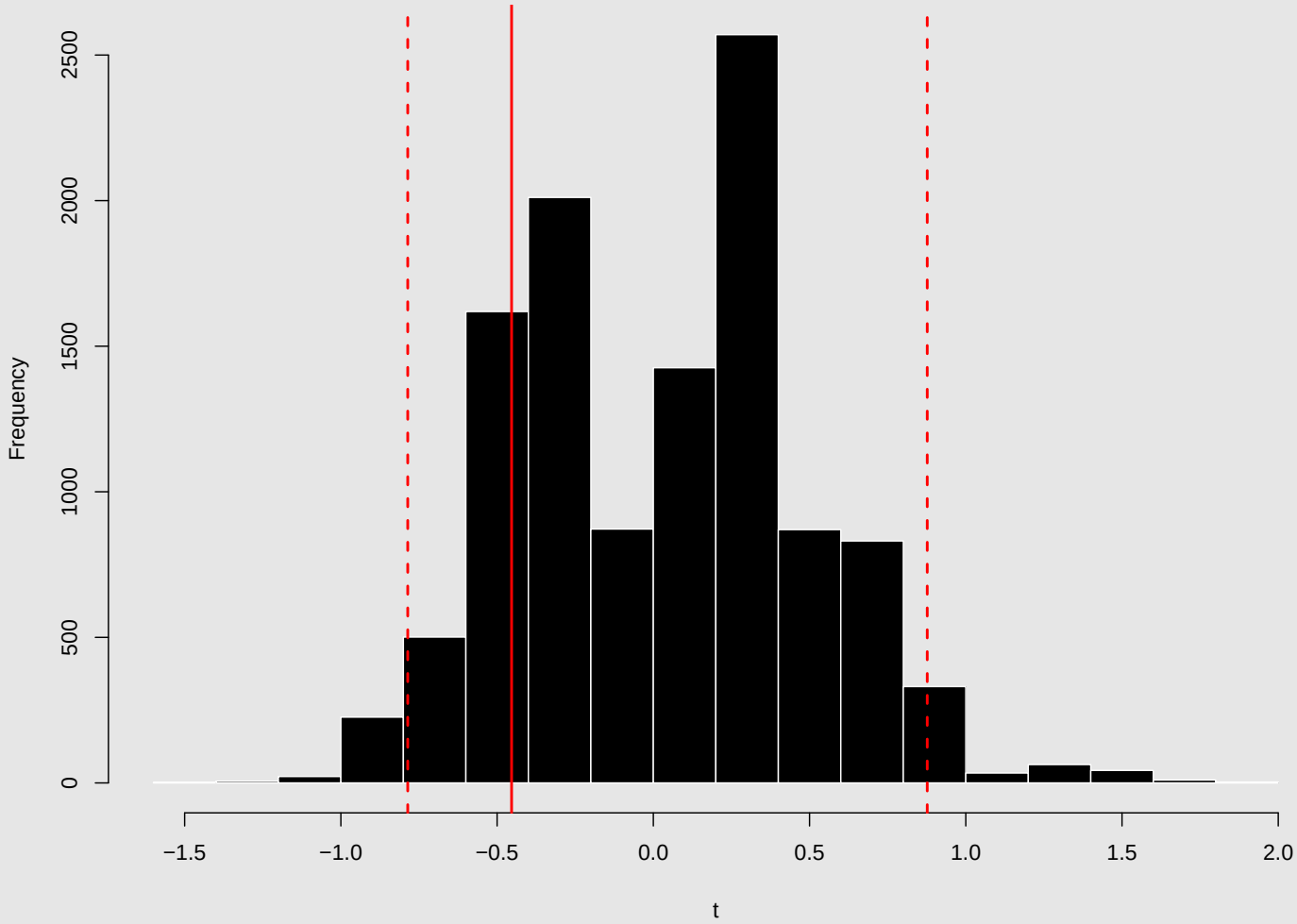
The permutation procedure can be used

- 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)

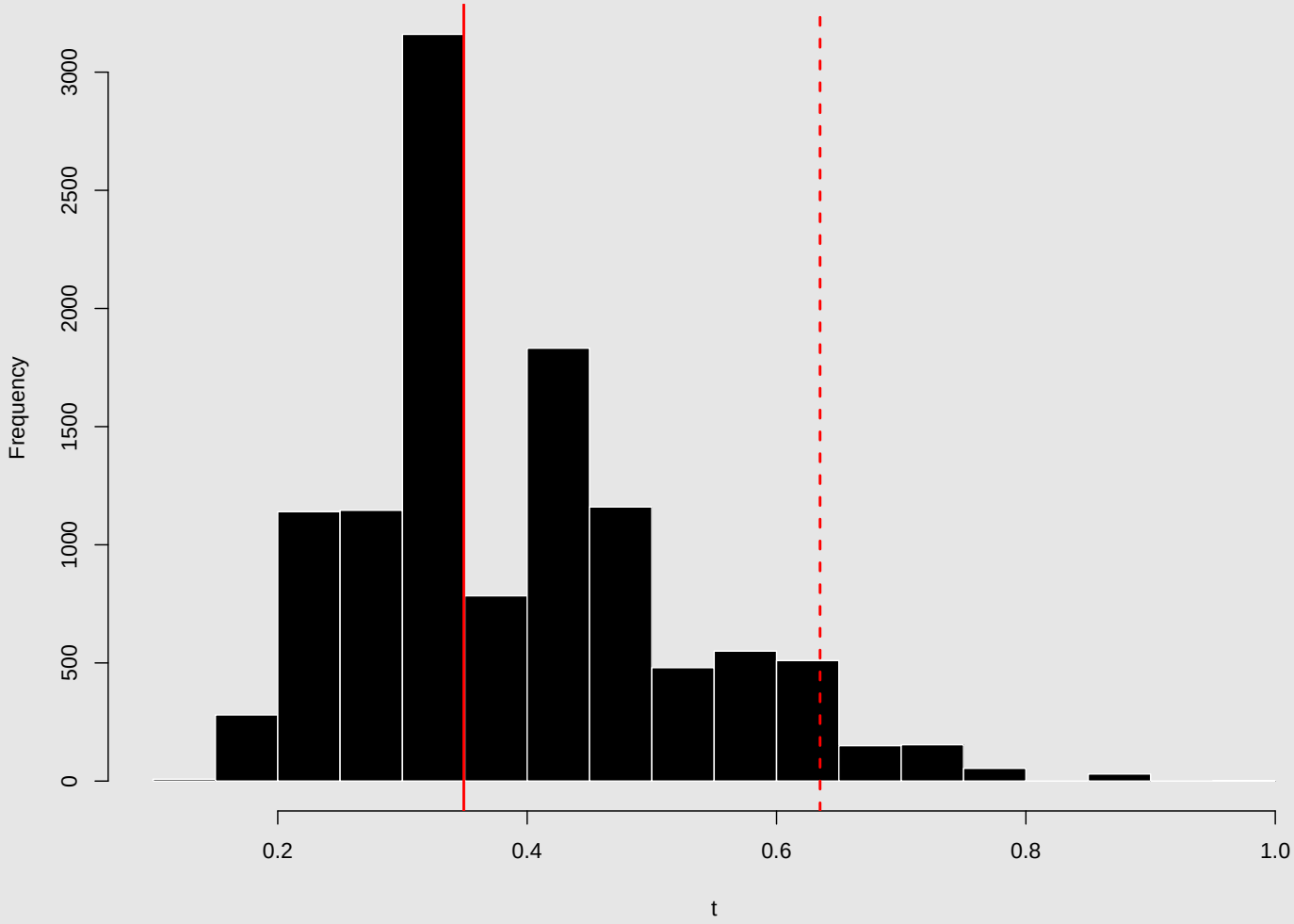
Test statistic $T = \log(s_Y/s_Z)$: two-sided p-value 0.3156



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.

Part IV

Monte Carlo Methods

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 independent, identically distributed random variables. Then, usually, as $n \longrightarrow \infty$,

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

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

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

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 $\{b_n\}$.

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

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

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

Monte Carlo: basic principles

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

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

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

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

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

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

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

for probability density $p(x)$ 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 p.

We need to establish

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

Monte Carlo: Validity

The strong law of large numbers ensures that

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

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

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

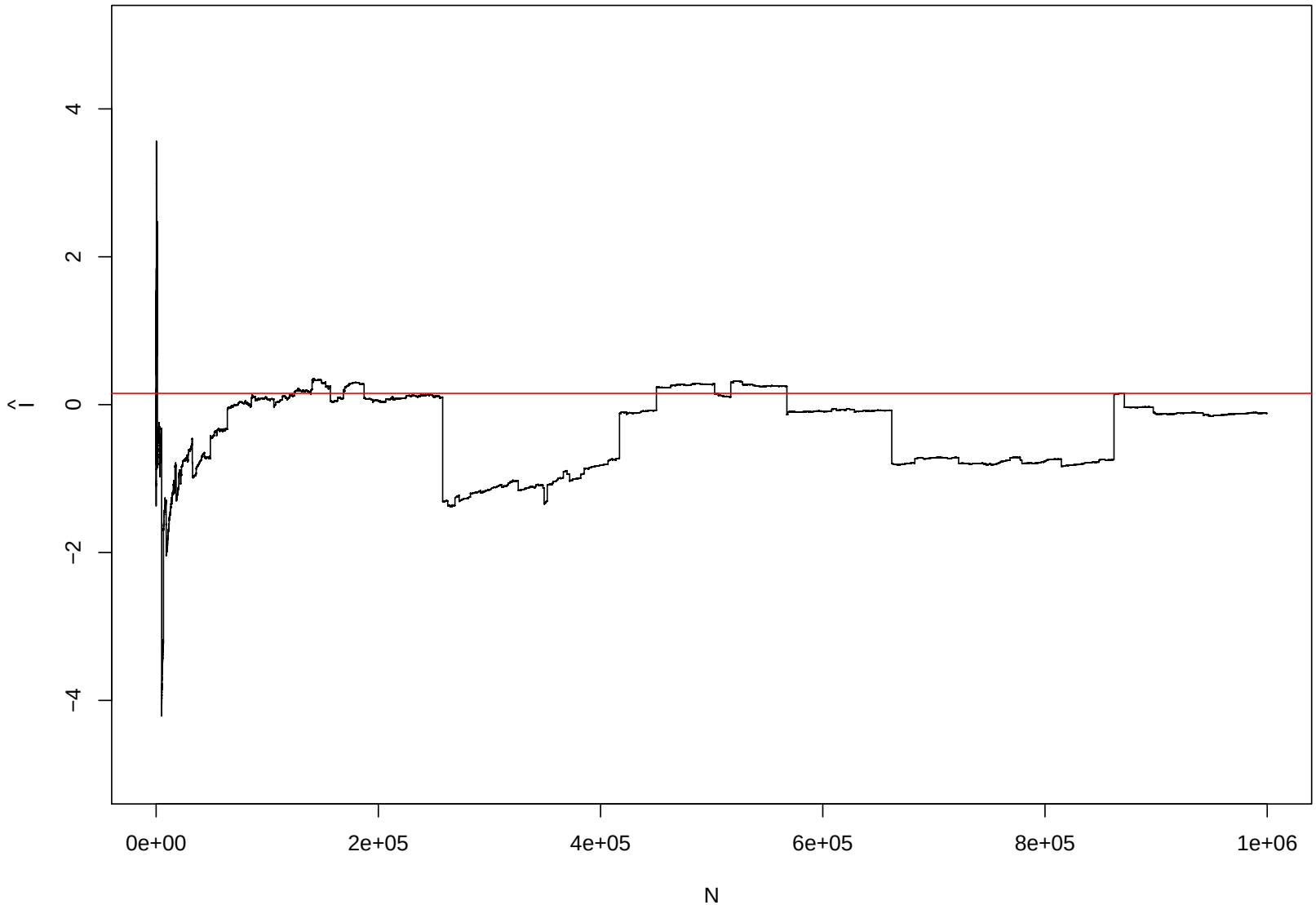
$$\begin{aligned}\int_0^1 \frac{1}{x}\sin(2\pi/x)\,dx &= \int_1^\infty \frac{\sin(2\pi t)}{t}\,dt \\ &= \int_0^\infty \frac{\sin(t)}{t}\,dt - \int_0^{2\pi} \frac{\sin(t)}{t}\,dt \\ &= \mathrm{Si}(\infty) - \mathrm{Si}(2\pi)\end{aligned}$$

where Si(.) is a special function (the *sine integral*) with $\mathrm{Si}(\infty) = \pi/2$.

The numerical value of the integral is 0.1526.

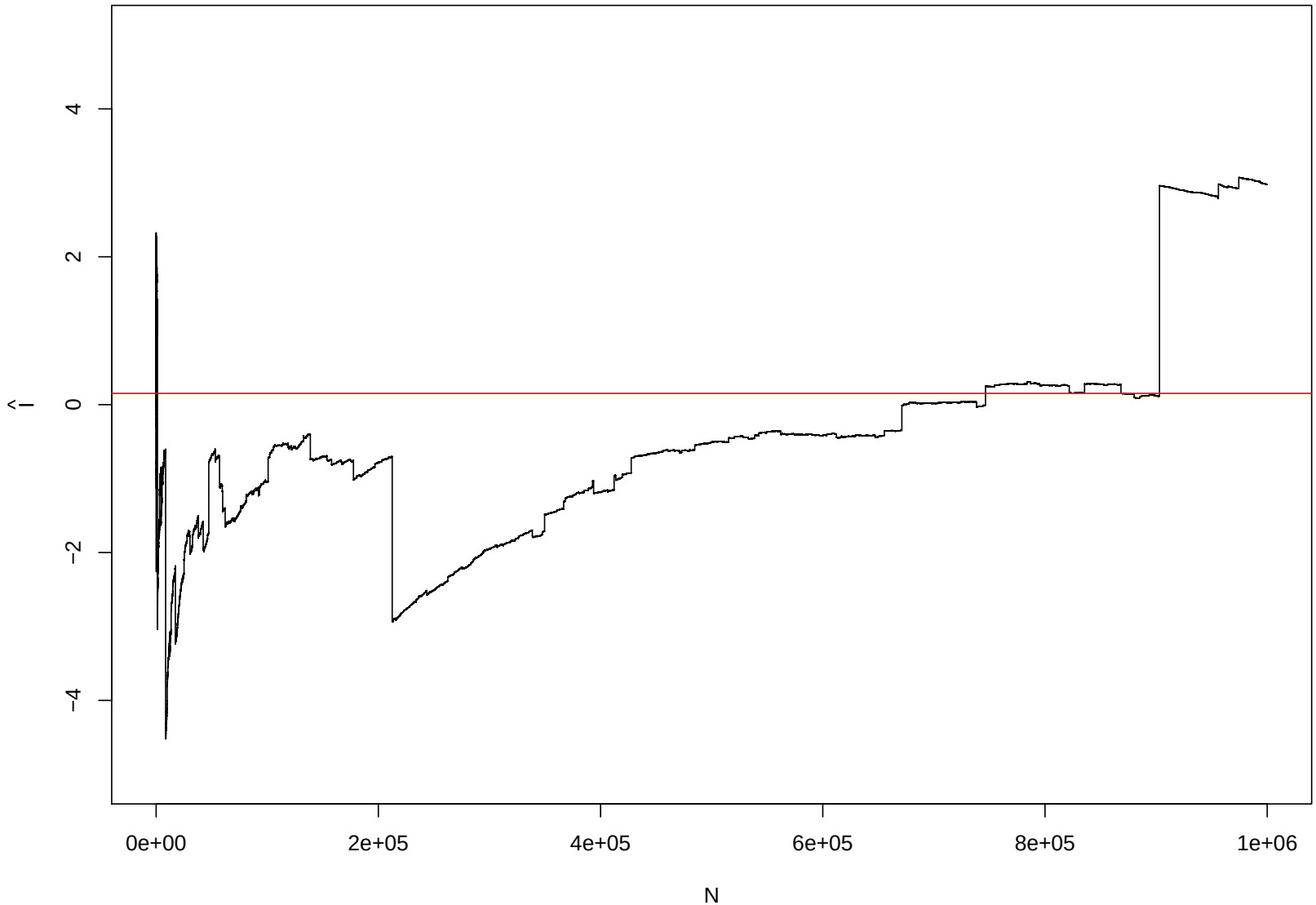
Monte Carlo: Validity

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

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

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

$$V(g) = \mathbb{V}ar[g(X)] = \int (g(x) - \mu(g))^2 \, p(x) \, dx$$

That is, the Monte Carlo estimator exhibits $o_P(\sqrt{N})$ convergence.

Monte Carlo Estimation: Example

Example (Normal Interval Probabilities)

Consider $p(x) = \phi(x)$ (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

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

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 $p_0 = 10^{-6}$:

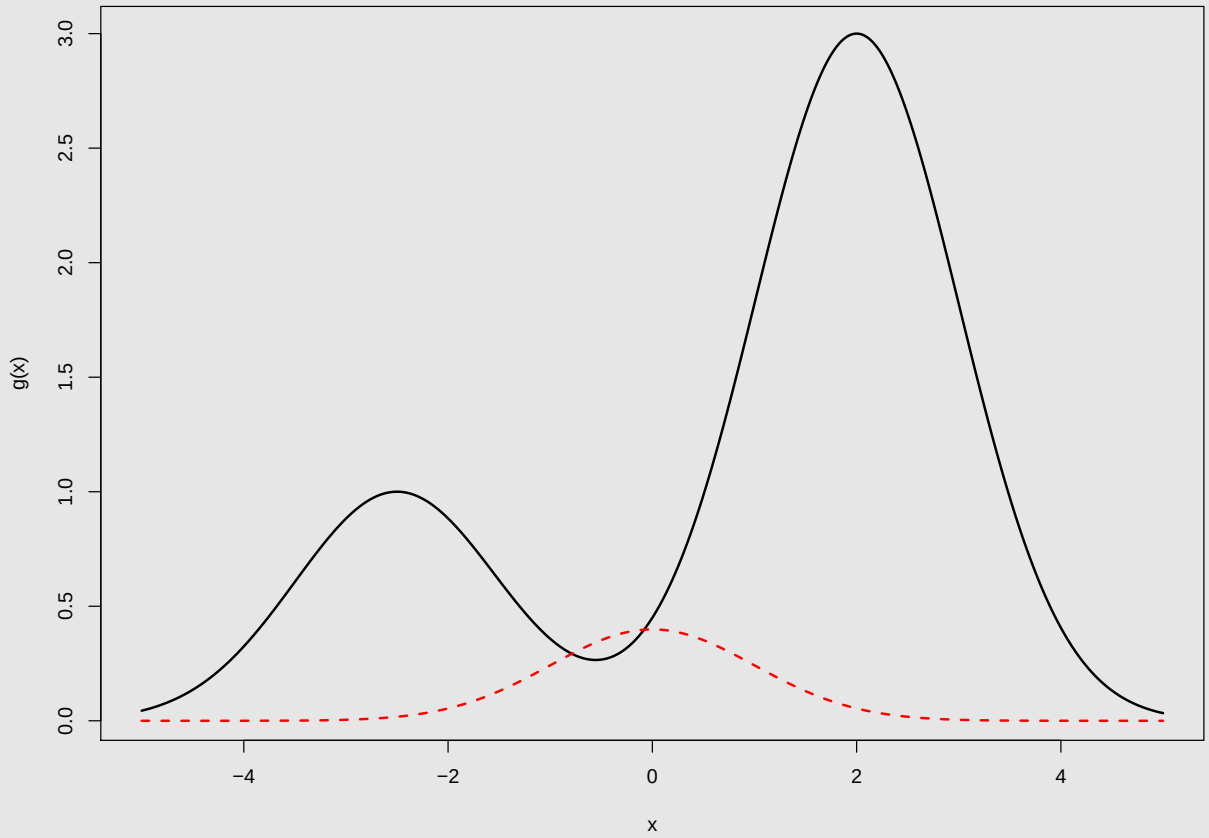
x_0	2	4	6	8	10	12
N_0	6.00×10^2	4.36×10^5	1.40×10^{10}	2.21×10^{16}	1.81×10^{24}	7.77×10^{33}

Monte Carlo Estimation: Example

Example (Mixture Integrand)

Consider

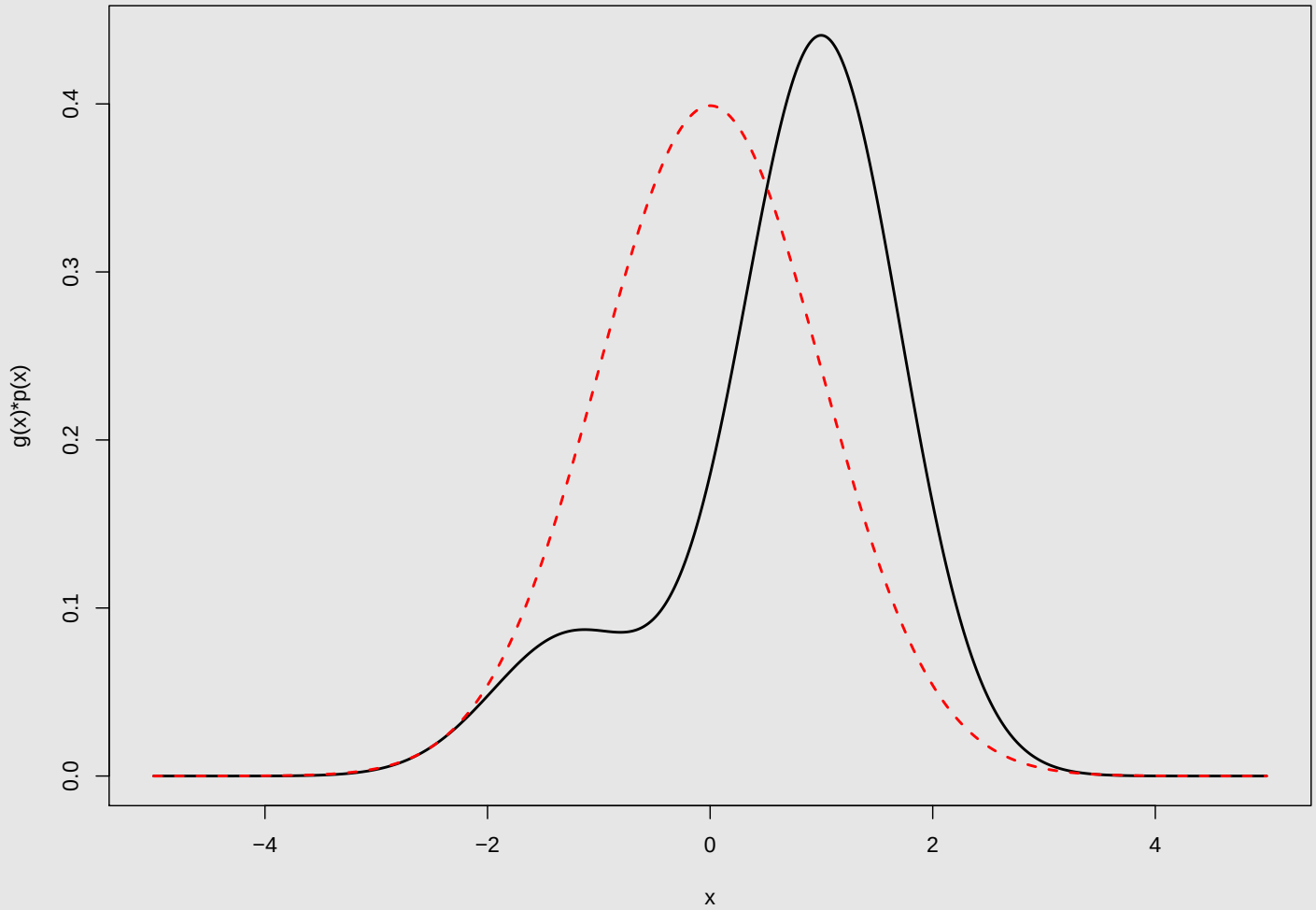
$$g(x) = \exp\left\{-\frac{1}{2}(x+2.5)^2\right\} + 3\exp\left\{-\frac{1}{2}(x-2)^2\right\}$$



Monte Carlo Estimation: Example

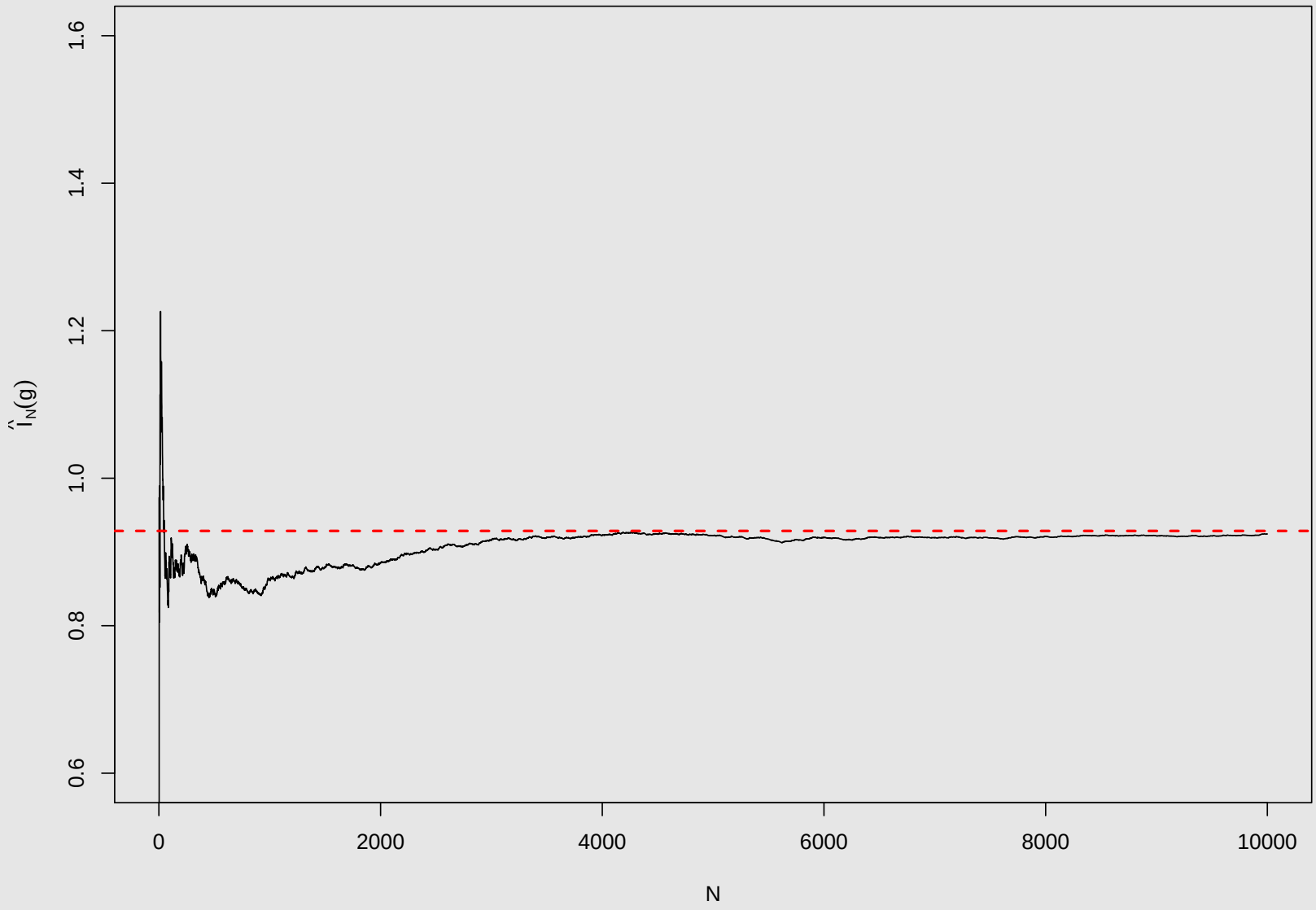
Example (Mixture Integrand)

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



Monte Carlo Estimation: Example

Example (Mixture Integrand)



Monte Carlo Estimation

Motivated by the deterministic approximation

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

where weights are determined by $p(x)$, 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 $p(x)$ generates points in regions where $g(x)$ is large in magnitude.

Importance sampling

The identity

$$\int g(x)p(x)\,dx = \int g(x)p(x)\frac{p_0(x)}{p_0(x)}\,dx = \int \frac{g(x)p(x)}{p_0(x)}p_0(x)\,dx$$

where p_0 is a probability density with support including the support of p, demonstrates that

$$\mathbb{E}_p[g(X)] = \mathbb{E}_{p_0}\left[\frac{g(X)p(X)}{p_0(X)}\right]$$

so that an estimator of the LHS is

$$\widehat{I}_N^{(p_0)}(g) = \frac{1}{N}\sum_{i=1}^{N}\frac{g(X_i)p(X_i)}{p_0(X_i)}$$

where $X_1, \ldots, X_N \sim p_0(.)$.

Importance sampling

$\widehat{I}_N^{(p_0)}$ is termed the *importance sampling* estimator.

p_0 is termed the *importance sampling density*. By careful choice of p_0, the estimator can have better performance than the Monte Carlo estimator in finite samples.

Note that

$$\widehat{I}_N^{(p_0)}(g) = \frac{1}{N}\sum_{i=1}^{N}\frac{p(X_i)}{p_0(X_i)}g(X_i) = \frac{1}{N}\sum_{i=1}^{N} w_0(X_i)g(X_i)$$

say, where

$$w_0(X_i) = \frac{p(X_i)}{p_0(X_i)}$$

is the *importance sampling weight*.

Importance sampling

Note that

$$\mathbb{E}_{p_0}\left[\frac{p(X)}{p_0(X)}\right] = \int p(x)\,dx = 1$$

so

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

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.

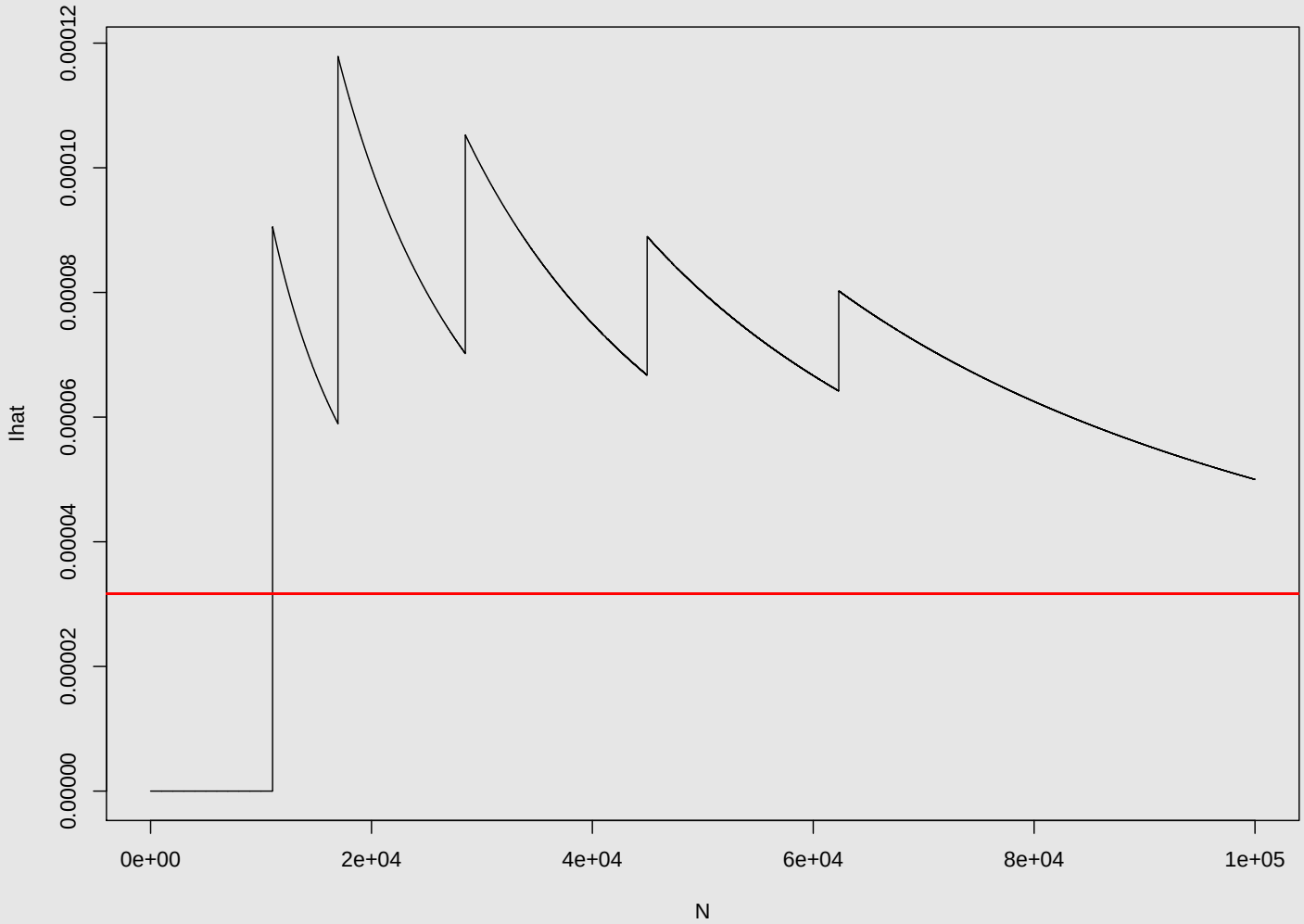
For $x_0 = 4$, $1 - \Phi(x_0) = 3.167 \times 10^{-5}$, and the probability of getting $\widehat{I}_N = 0$ for $N = 100000$ is

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

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

$$p_0(x) = \lambda \exp\{-\lambda(x - x_0)\} \qquad x > x_0$$

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

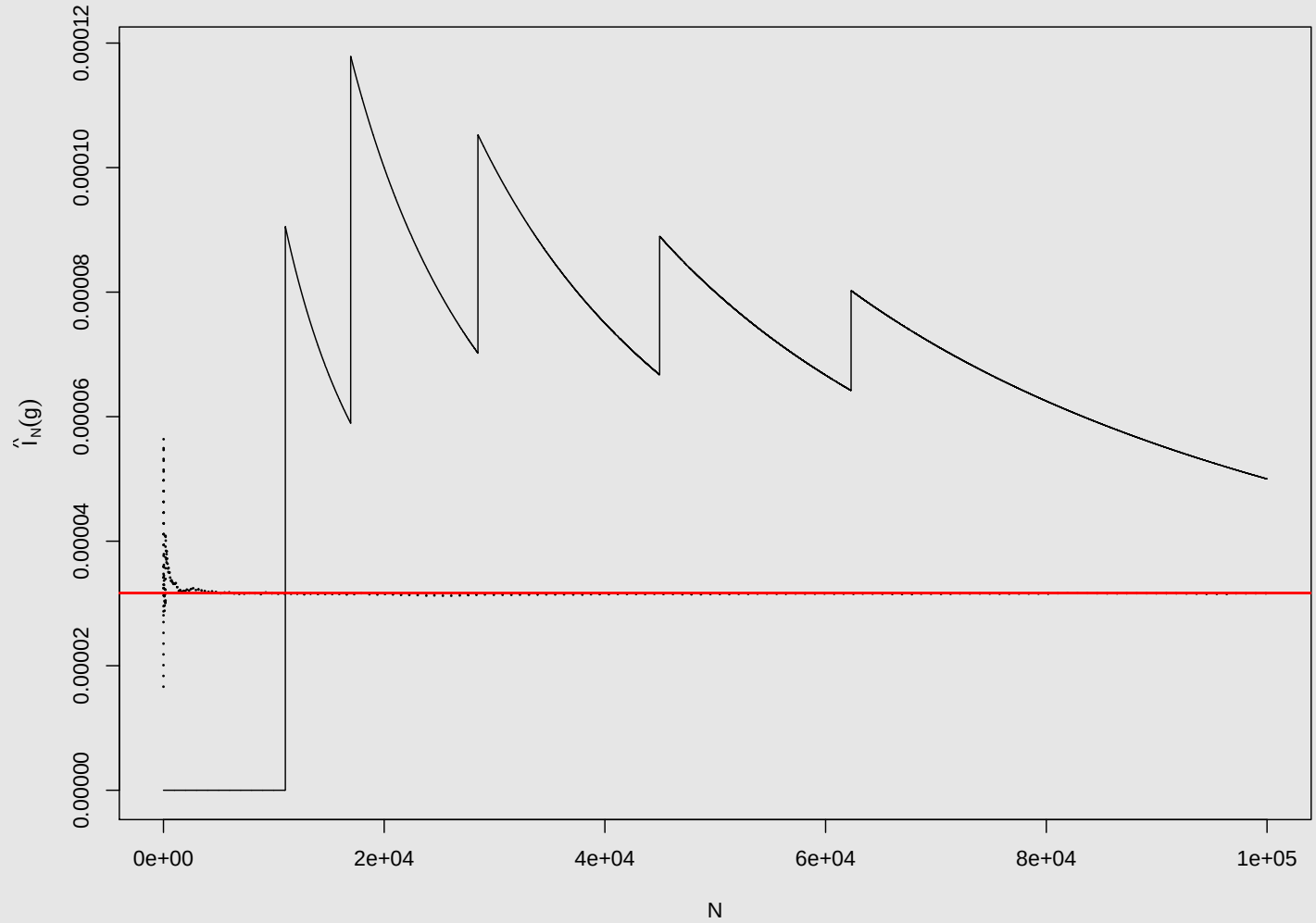
$$\widehat{I}_N^{(p_0)} = \frac{1}{N} \sum_{i=1}^{N} \frac{g(X_i)p(X_i)}{p_0(X_i)} = \frac{1}{N\lambda} \sum_{i=1}^{N} \phi(X_i) \exp\{\lambda(X_i - x_0)\}$$

as $g(X_i) = 1$ for all variates sampled from p_0.

Importance sampling: Example

Example (Normal tail probability)

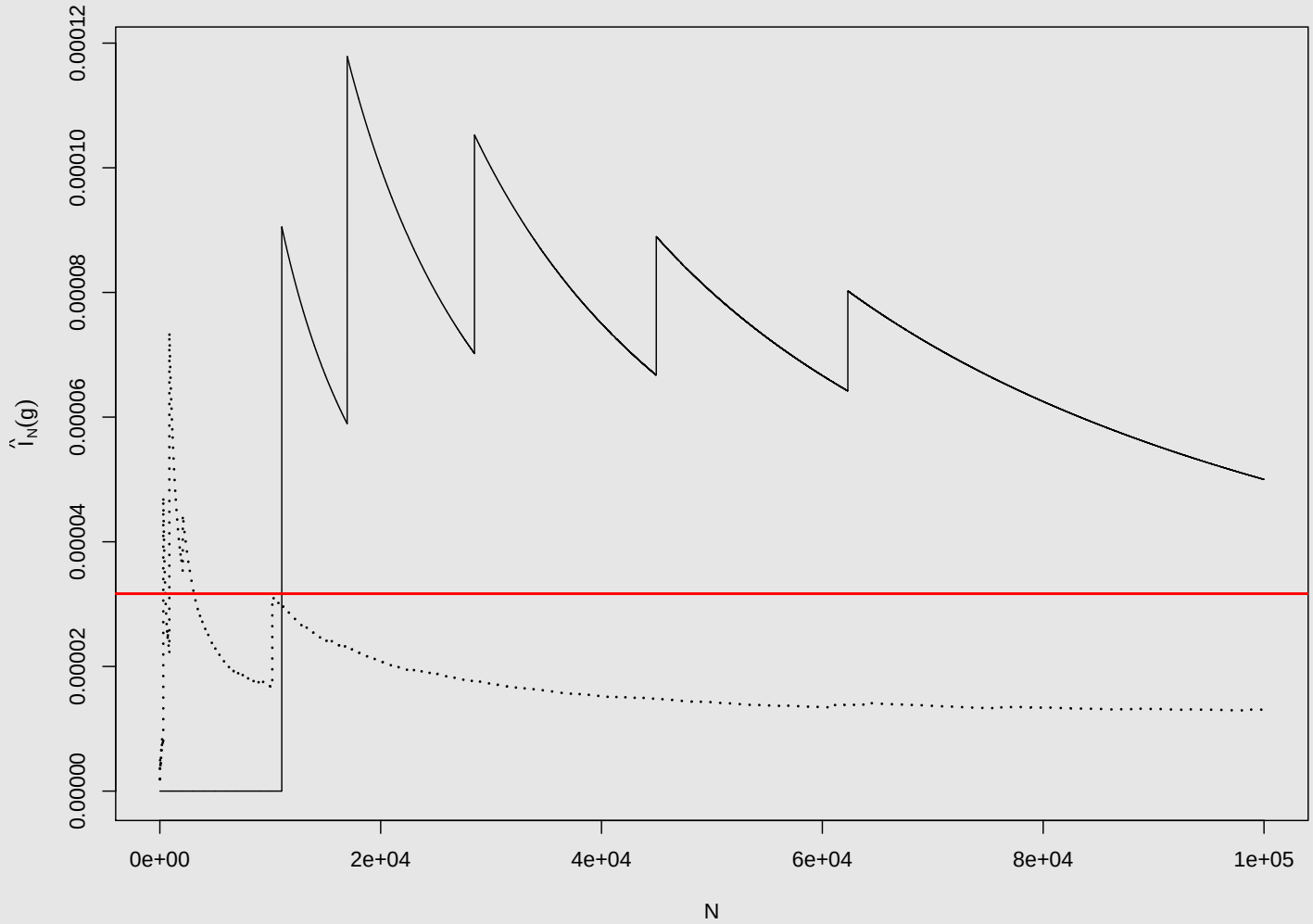
IS estimate $\widehat{I}_N^{(p_0)}(g)$ (dotted) for $\lambda = 10$



Importance sampling: Example

Example (Normal tail probability)

IS estimate $\widehat{I}_N^{(p_0)}(g)$ (dotted) for $\lambda = 100$



Importance sampling: Example

Example (Normal tail probability)

Consider using the importance sampling density $p_0(x) \equiv t(5)$, 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

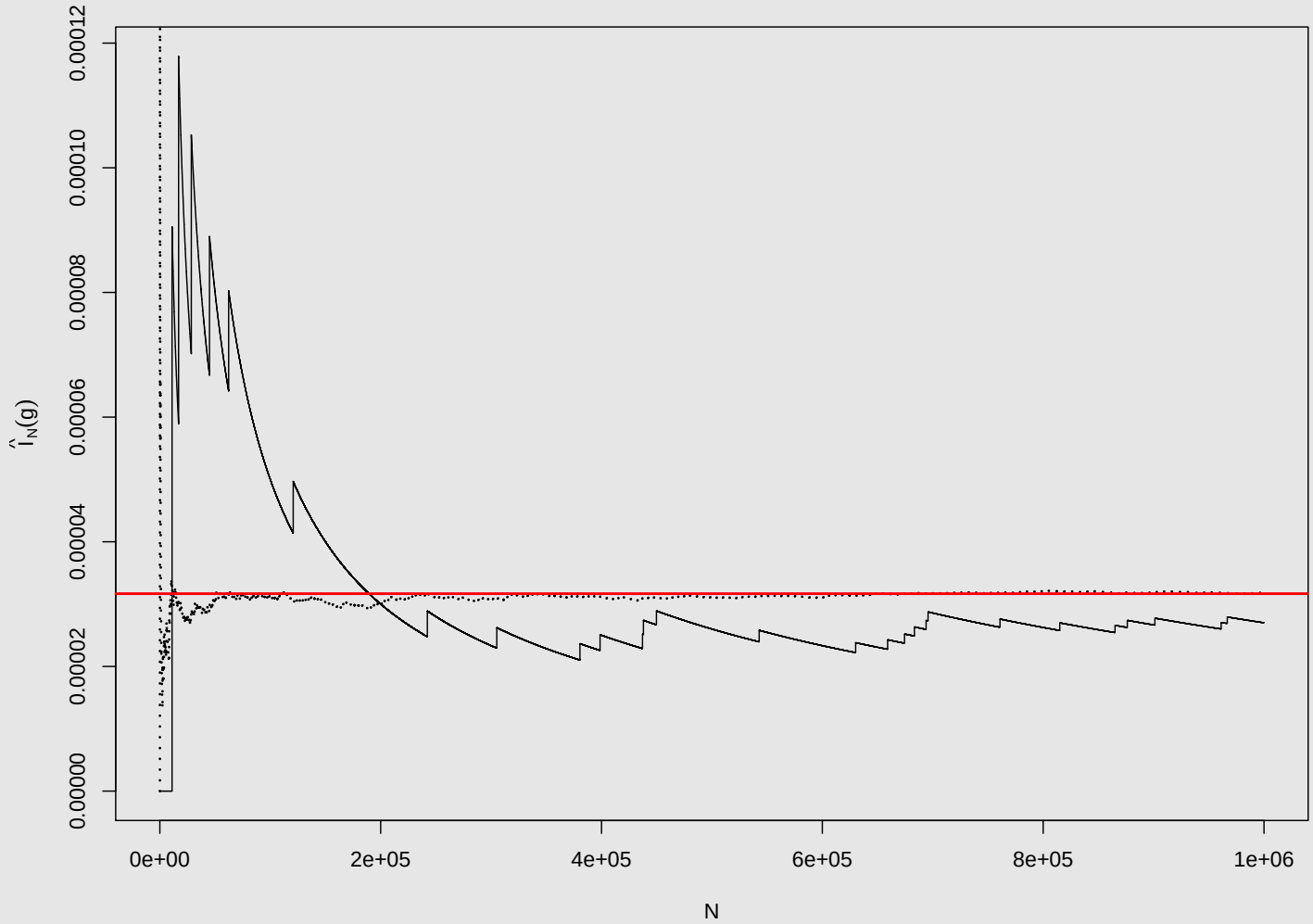
$$g(x) = \begin{cases} 0 & x \leqslant x_0 \\ 1 & x > x_0 \end{cases}$$

must be included in the computation of $\widehat{I}_N^{(p_0)}(g)$.

Importance sampling: Example

Example (Normal tail probability)

IS estimate $\widehat{I}_N^{(p_0)}(g)$ (dotted) for $p_0 \equiv t(5)$



Optimal Importance Sampling

There are many possible choices for the importance sampling density p_0. We now seek guidelines for choosing p_0 optimally.

Note first that the variance of $\widehat{I}_N^{(p_0)}(g)$ is finite if and only if

$$\frac{g(X)p(X)}{p_0(X)}$$

has finite variance, that is, if and only if

$$\mathbb{E}_{p_0}\left[\left\{\frac{g(X)p(X)}{p_0(X)}\right\}^2\right] = \int_{-\infty}^{\infty}\left\{\frac{g(x)p(x)}{p_0(x)}\right\}^2 p_0(x)\,dx$$

is finite.

Optimal Importance Sampling

This equates to

$$\int_{-\infty}^{\infty} \frac{\{g(x)p(x)\}^2}{p_0(x)}\, dx$$

being finite. Note also that

$$\mathbb{E}_{p_0}\left[\left\{\frac{g(X)p(X)}{p_0(X)}\right\}^2\right] = \mathbb{E}_p\left[\{g(X)\}^2\,\frac{p(X)}{p_0(X)}\right]$$

Now, if $p(x)/p_0(x)$ is **unbounded** on the support of p, then these expectations, and the variance of $\widehat{I}_N^{(p_0)}(g)$, are **not finite**.

Therefore, for p with unbounded support, we must ensure that $p(x)/p_0(x)$ 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. We have that

$$\mathbb{V}ar_{p_0}\left[\frac{g(X)p(X)}{p_0(X)}\right] = \mathbb{E}_{p_0}\left[\left\{\frac{g(X)p(X)}{p_0(X)}\right\}^2\right] - \left\{\mathbb{E}_{p_0}\left[\frac{g(X)p(X)}{p_0(X)}\right]\right\}^2.$$

But

$$\mathbb{E}_{p_0}\left[\frac{g(X)p(X)}{p_0(X)}\right] = \mathbb{E}_p\left[g(X)\right]$$

does not depend on p_0, so to minimize the variance, we must minimize the first term. Note that

$$\begin{aligned}\mathbb{E}_{p_0}\left[\left\{\frac{g(X)p(X)}{p_0(X)}\right\}^2\right] &\geqslant \left\{\mathbb{E}_{p_0}\left[\frac{|g(X)|p(X)}{p_0(X)}\right]\right\}^2 \\ &= \left\{\int |g(x)|p(x)\,dx\right\}^2\end{aligned}$$

by Jensen's inequality.

Optimal Importance Sampling

This is a lower bound on the variance of the estimator which is independent of p_0. By the logic of Jensen's inequality, the inequality becomes an equality when

$$\frac{|g(x)|p(x)}{p_0(x)}$$

is a constant, that is, when

$$p_0(x) \propto |g(x)|p(x)$$

that is,

$$p_0(x) = \frac{|g(x)|p(x)}{\int |g(u)|p(u)\, du}.$$

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

Optimal Importance Sampling

Note that, in general, if $X_1, \ldots, X_N, \ldots \sim p_0$, then

$$\frac{1}{N}\sum_{i=1}^{N}\frac{p(X_i)}{p_0(X_i)} \xrightarrow{a.s.} \int_{-\infty}^{\infty}\frac{p(x)}{p_0(x)}p_0(x)\,dx = 1$$

so therefore

$$\widetilde{I}_N^{(p_0)}(g) = \frac{\sum_{i=1}^{N}\frac{g(X_i)p(X_i)}{p_0(X_i)}}{\sum_{i=1}^{N}\frac{p(X_i)}{p_0(X_i)}} \xrightarrow{a.s.} \mathbb{E}[g(X)]$$

also, if the expectation exists. For finite N, this estimator $\widetilde{I}_N^{(p_0)}(g)$ is biased, but asymptotically unbiased. It may have smaller variance than $\widehat{I}_N^{(p_0)}(g)$

Optimal Importance Sampling

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

$$\widetilde{I}_N^{(p_0)}(g) = \frac{\sum_{i=1}^N \frac{g(X_i)p(X_i)}{p_0(X_i)}}{\sum_{i=1}^N \frac{p(X_i)}{p_0(X_i)}} = \frac{\sum_{i=1}^N \frac{g(X_i)}{|g(X_i)|}}{\sum_{i=1}^N \frac{1}{|g(X_i)|}}$$

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

$$\widetilde{I}_N^{(p_0)}(g) = \frac{N}{\sum_{i=1}^N \frac{1}{g(X_i)}}$$

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 p_0 so that

$$\frac{|g(x)|p(x)}{p_0(x)}$$

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

This suggests designing p_0 to have high density whenever the original integrand $|g(x)|p(x)$ is large, subject to the constraint

$$\frac{p(x)}{p_0(x)} < M \qquad \text{or} \qquad \mathbb{E}_p\left[\frac{p(X)}{p_0(X)}\right] < M$$

for some finite bound M.

Monte Carlo and 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.

- $g(\boldsymbol{x})$ is a scalar function of vector argument $\boldsymbol{x}$,
- $p(\boldsymbol{x})$ and $p_0(\boldsymbol{x})$ 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:** The distribution of 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 reasonably challenging. All 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

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

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 also 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.

The most widely used shift generator is the *Mersenne Twister*, which is the default in R, and in its most typical implementation has a cycle length of

$$2^{19937} - 1$$

It produces very reliable uniform pseudo-random variates that are virtually indistinguishable from truly random numbers.

Rejection sampling

Suppose that we wish to sample from an arbitrary density $p(x)$, but this is not straightforward directly. Suppose instead we have easy access to variates from the density $p_0(x)$, where

$$\frac{p(x)}{p_0(x)} < M \qquad \text{for all } x$$

Then the following algorithm produces variates from p:

1. generate x from p_0
2. generate u from $Uniform(0, 1)$
3. if

$$u \leqslant \frac{p(x)}{Mp_0(x)}$$

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

Rejection sampling

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

$$M = \sup_x \frac{p(x)}{p_0(x)}$$

If $p(x)$ is bounded, and has support which is a bounded subset $\mathcal{X}$ of $\mathbb{R}$, then p_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

$$\begin{aligned}
\mathbb{P}[X \text{ is accepted}] &= \mathbb{P}\left[U \leqslant \frac{p(X)}{Mp_0(X)}\right] \\
&= \int_{-\infty}^{\infty} \left\{ \int_0^{p(x)/(Mp_0(x))} du \right\} p_0(x)\, dx \\
&= \int_{-\infty}^{\infty} \frac{p(x)}{Mp_0(x)} p_0(x)\, dx \\
&= \frac{1}{M} \int_{-\infty}^{\infty} p(x)\, dx = \frac{1}{M}
\end{aligned}$$

so ideally M should be as small as possible.

Rejection sampling: Efficiency

Note also neither p nor p_0 need to be normalized for this algorithm to be valid:

- If $p(x) = mg(x)$ and $p_0(x) = m_0 g_0(x)$, then

$$\frac{p(x)}{p_0(x)} = \frac{m}{m_0}\frac{g(x)}{g_0(x)} < M$$

  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

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

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

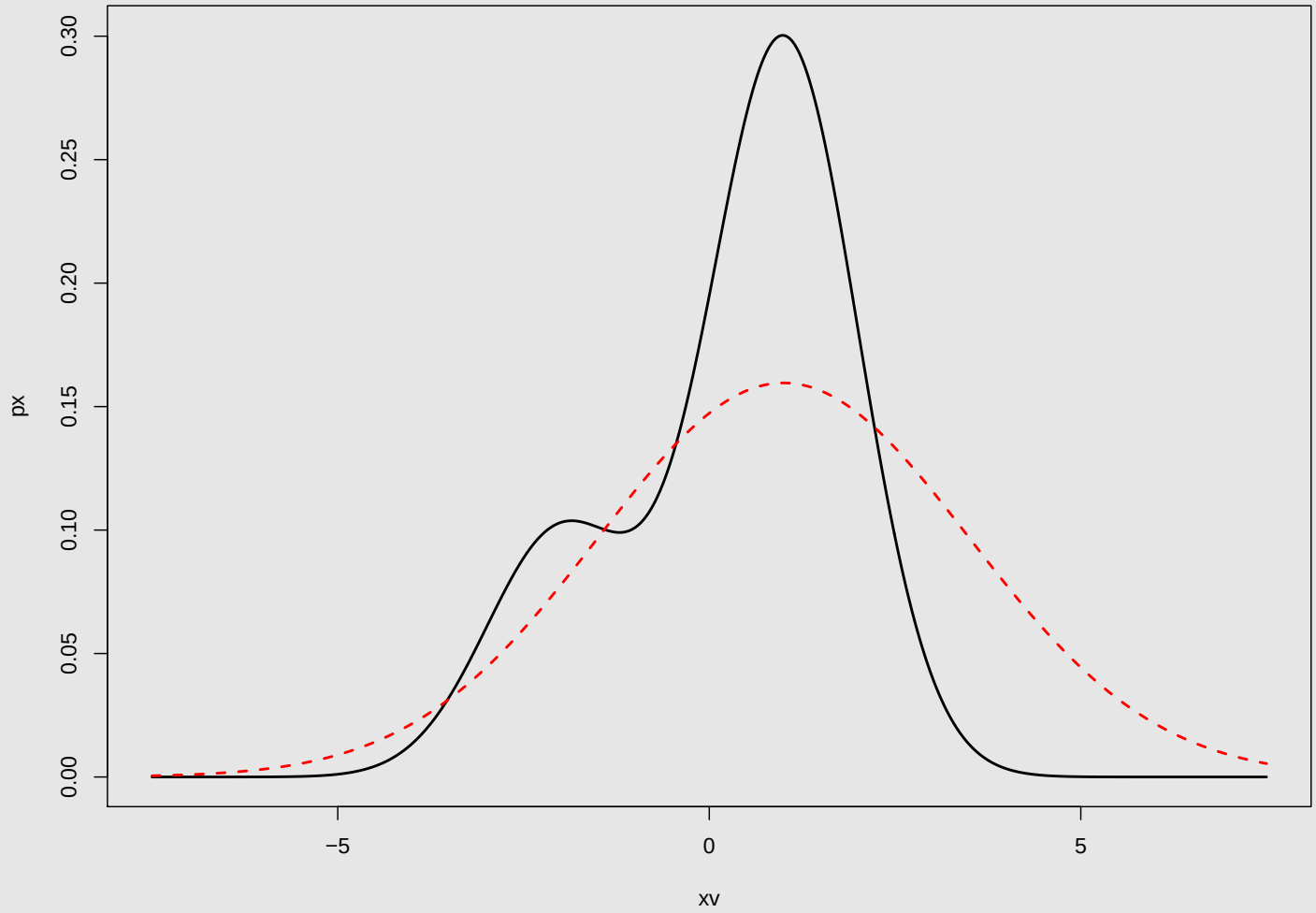
$$p_0(x) = \frac{1}{\sigma}\phi\left(\frac{x-1}{\sigma}\right)$$

for some variance σ^2.

Rejection sampling

Example (Normal mixture)

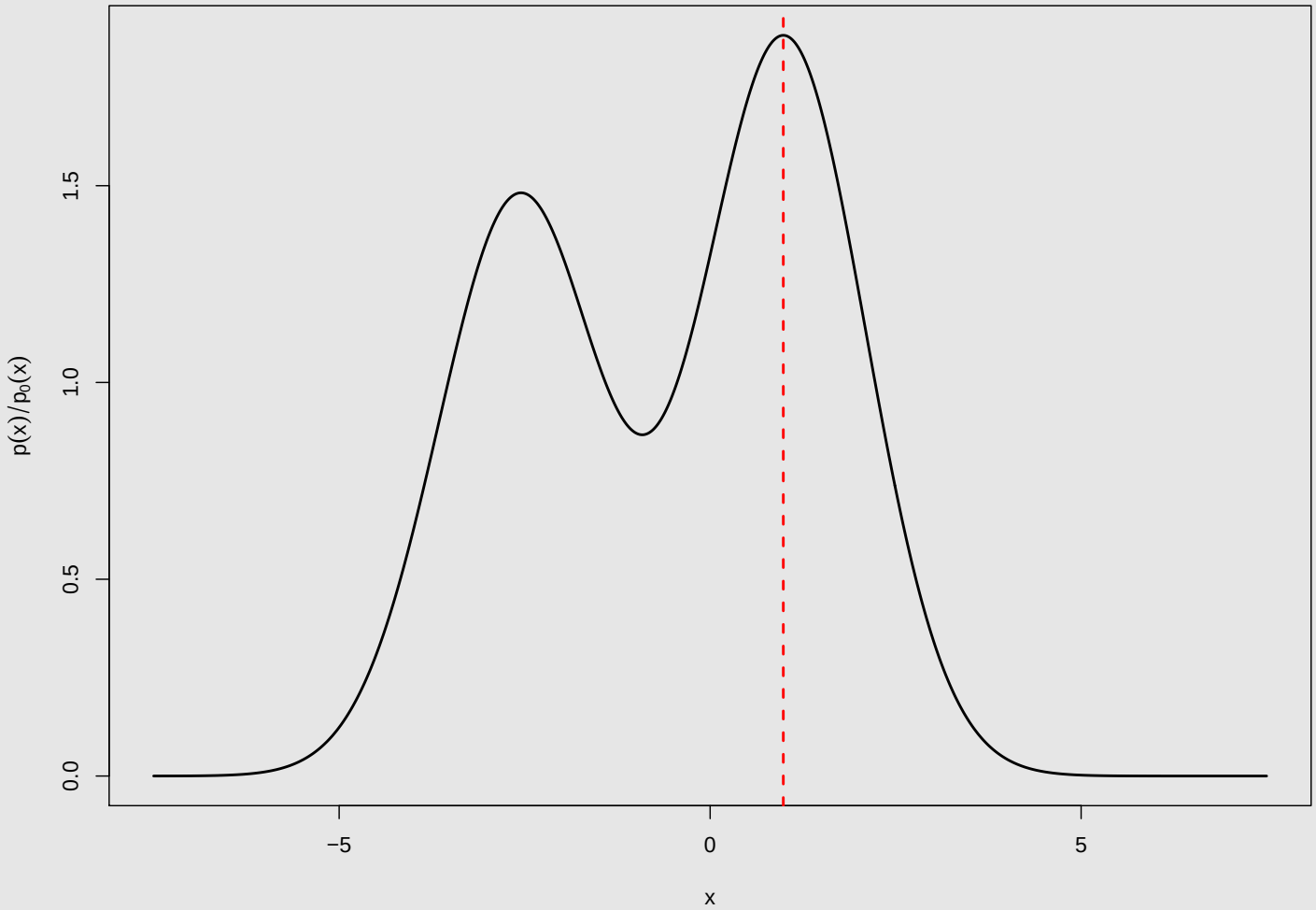
Target $p(x)$ (solid) and $p_0(x)$ (dashed) with $\sigma = 2.5$.



Rejection sampling

Example (Normal mixture)

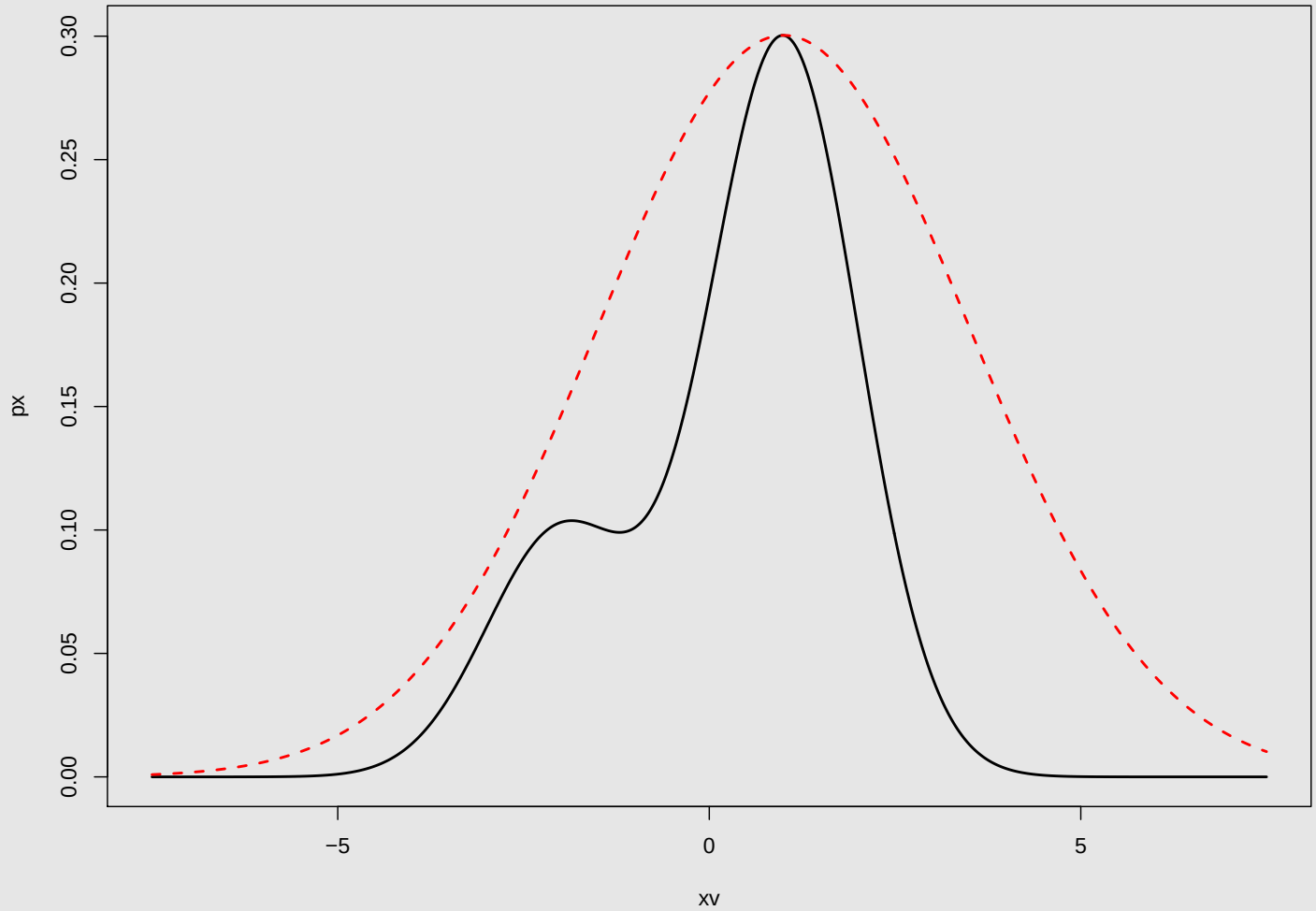
$p(x)/p_0(x)$ maximized at $x = 0.9863$, yielding $M = 1.8821$:



Rejection sampling

Example (Normal mixture)

$p(x)$ bounded by $Mp_0(x)$:



Rejection sampling

If x is a variate from p_0, then $p_0(x)$ is the value of the density at that variate, and $Mp_0(x)$ is the scaled version. Consider a vertical slice at x which is the line segment

$$(x, 0) \longrightarrow (0, Mp_0(x))$$

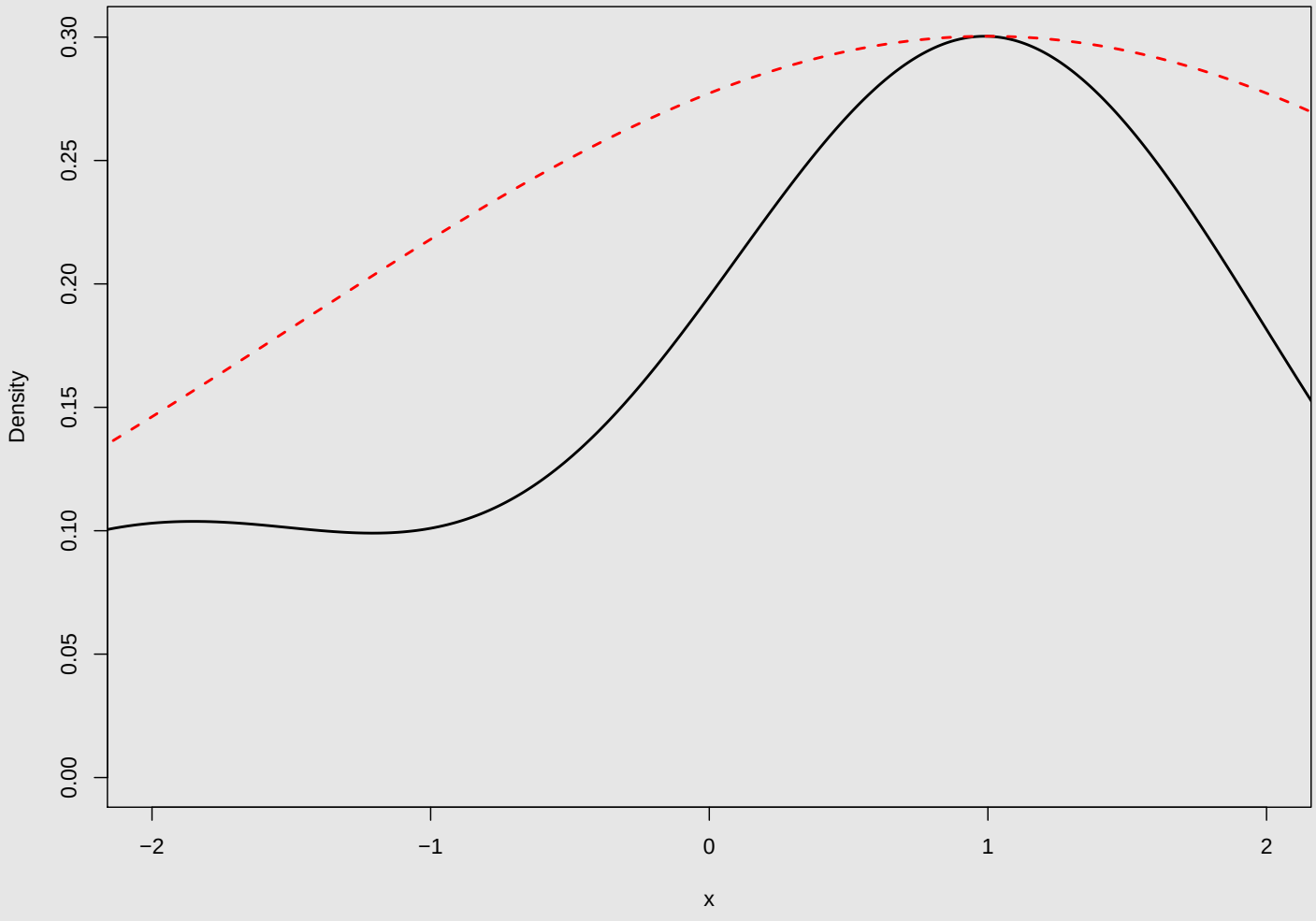
By assumption $p(x) \leqslant Mp_0(x)$. If u is simulated from $Uniform(0, 1)$, then $uMp_0(x)$ is the random portion of the vertical slice, and if

$$p(x) \leqslant uMp_0(x)$$

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

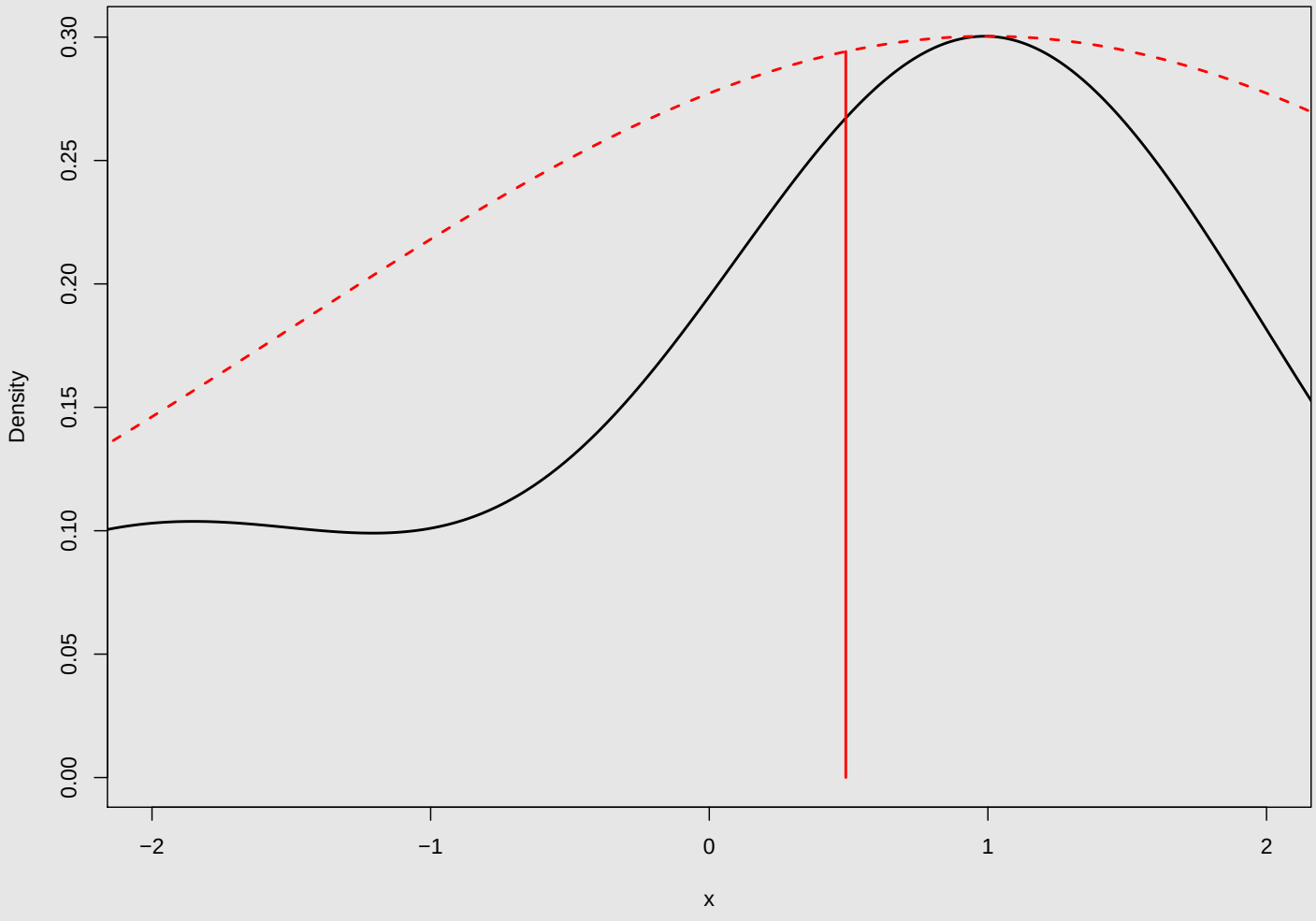
Rejection sampling

Example (Normal mixture)



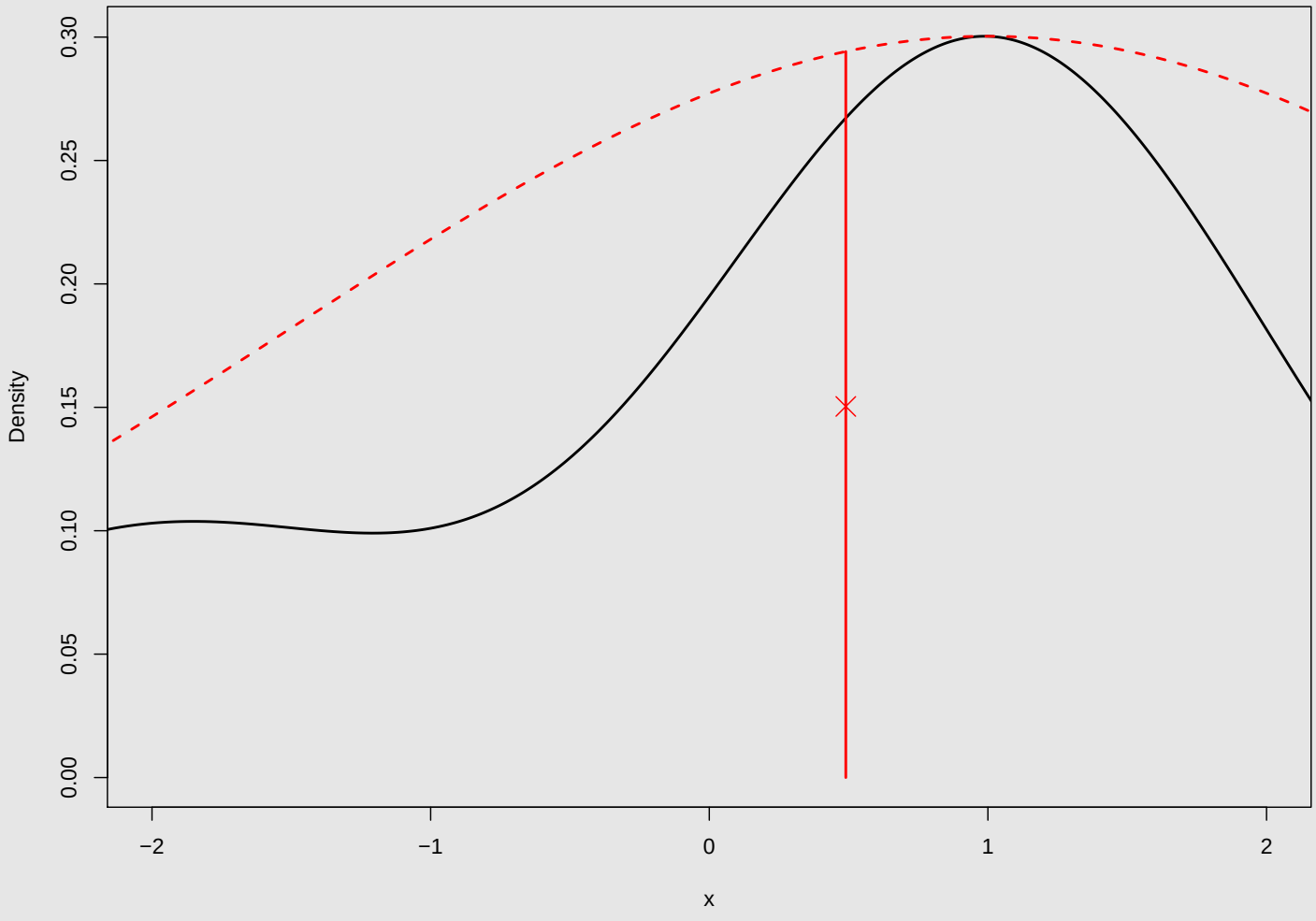
Rejection sampling

Example (Normal mixture)



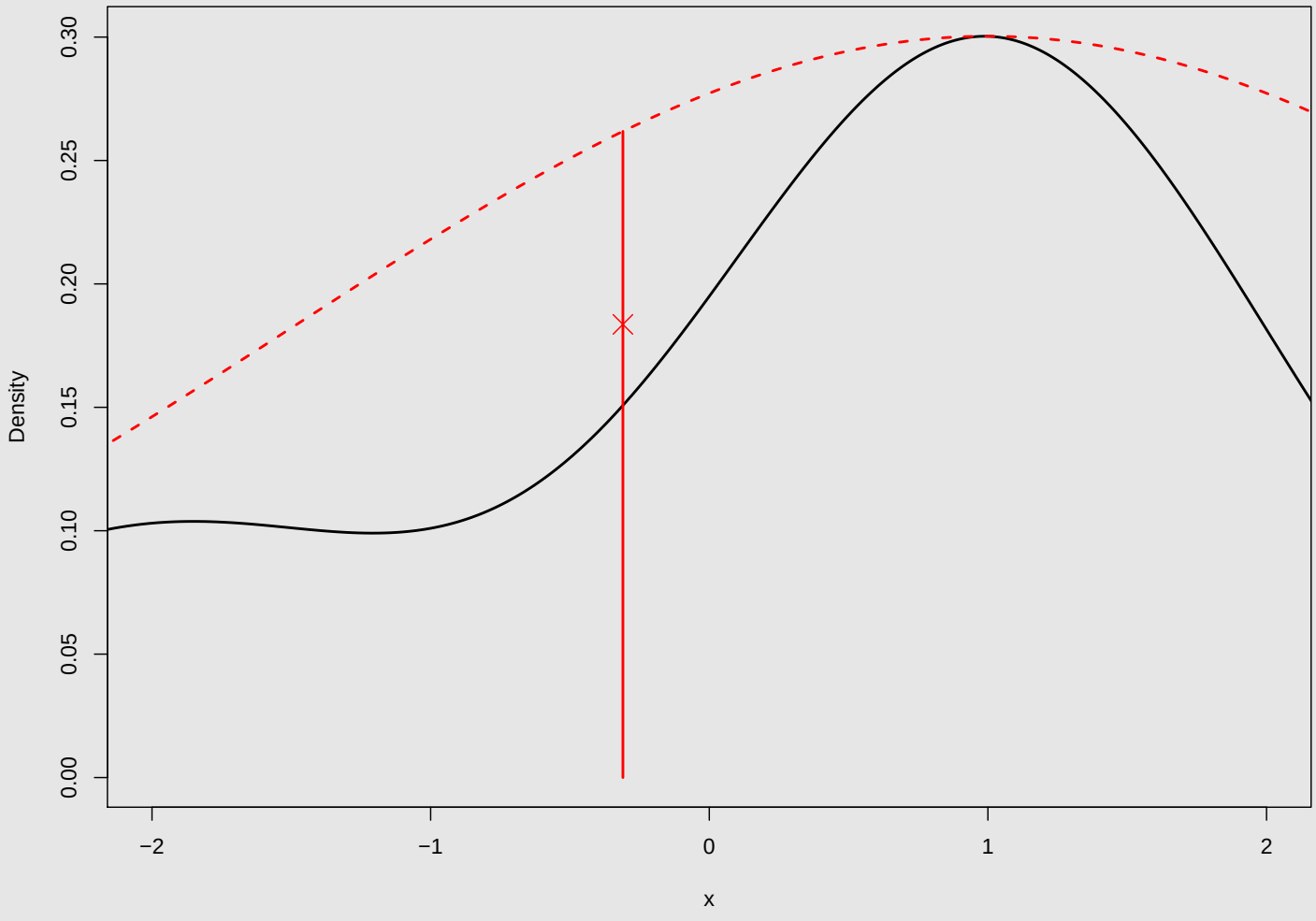
Rejection sampling

Example (Normal mixture)



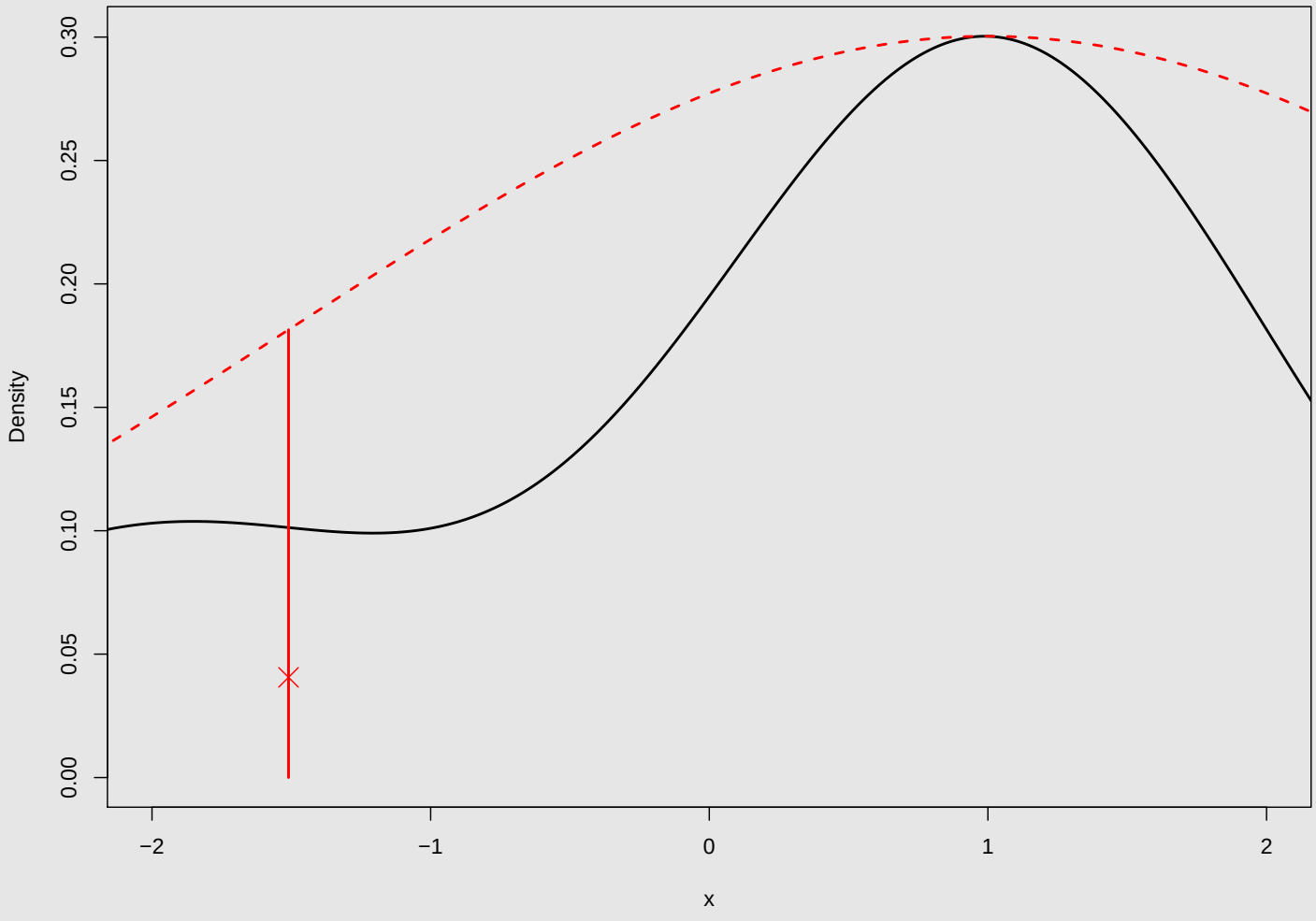
Rejection sampling

Example (Normal mixture)



Rejection sampling

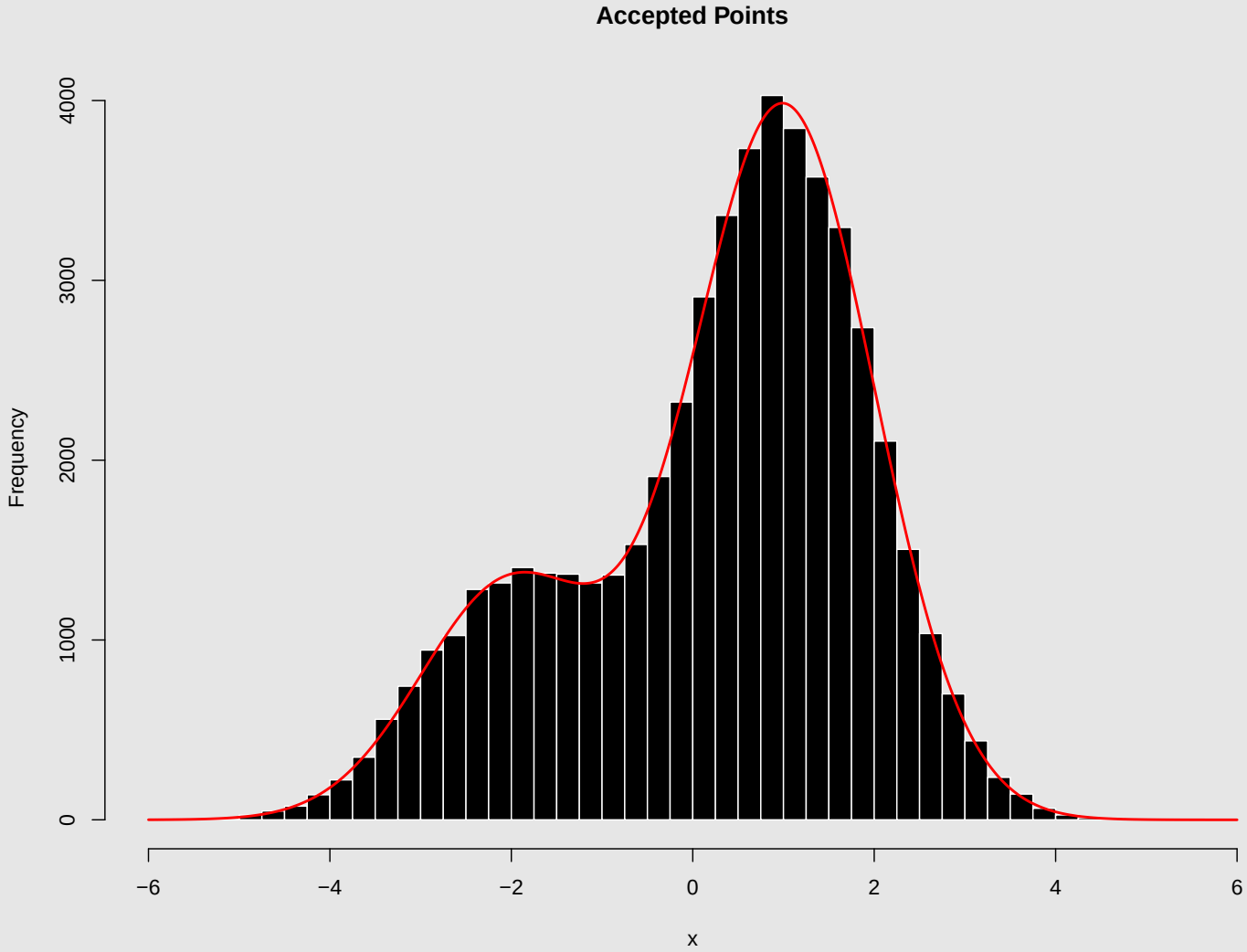
Example (Normal mixture)



Rejection sampling

Example (Normal mixture)

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



Rejection sampling

Notes:

- The acceptance rate is 1 if $p_0(x) \equiv p(x)$.
- If $p_0(x)$ resembles $p(x)$, then the acceptance rate will be high.
- The maximization of $p(x)/p_0(x)$ 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 $p(x)$ is *log-concave* (that is $\log p(x)$ is concave), then *adaptive rejection sampling* constructs an adaptive proposal density using rejected points.

Even when a point x is rejected, $p(x)$ is computed, and this information is useful improving p_0.

For example, consider sampling from $Gamma(\alpha, 1)$ for noninteger $\alpha > 0$. We have

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

which is concave, as the second derivative is negative.

Adaptive Rejection sampling

The proposal function $p_0(x)$ can be taken as $Exponential(1/2)$; in this case

$$\frac{p(x)}{p_0(x)} = \frac{2}{\Gamma(\alpha)} x^{\alpha-1} e^{-x/2}$$

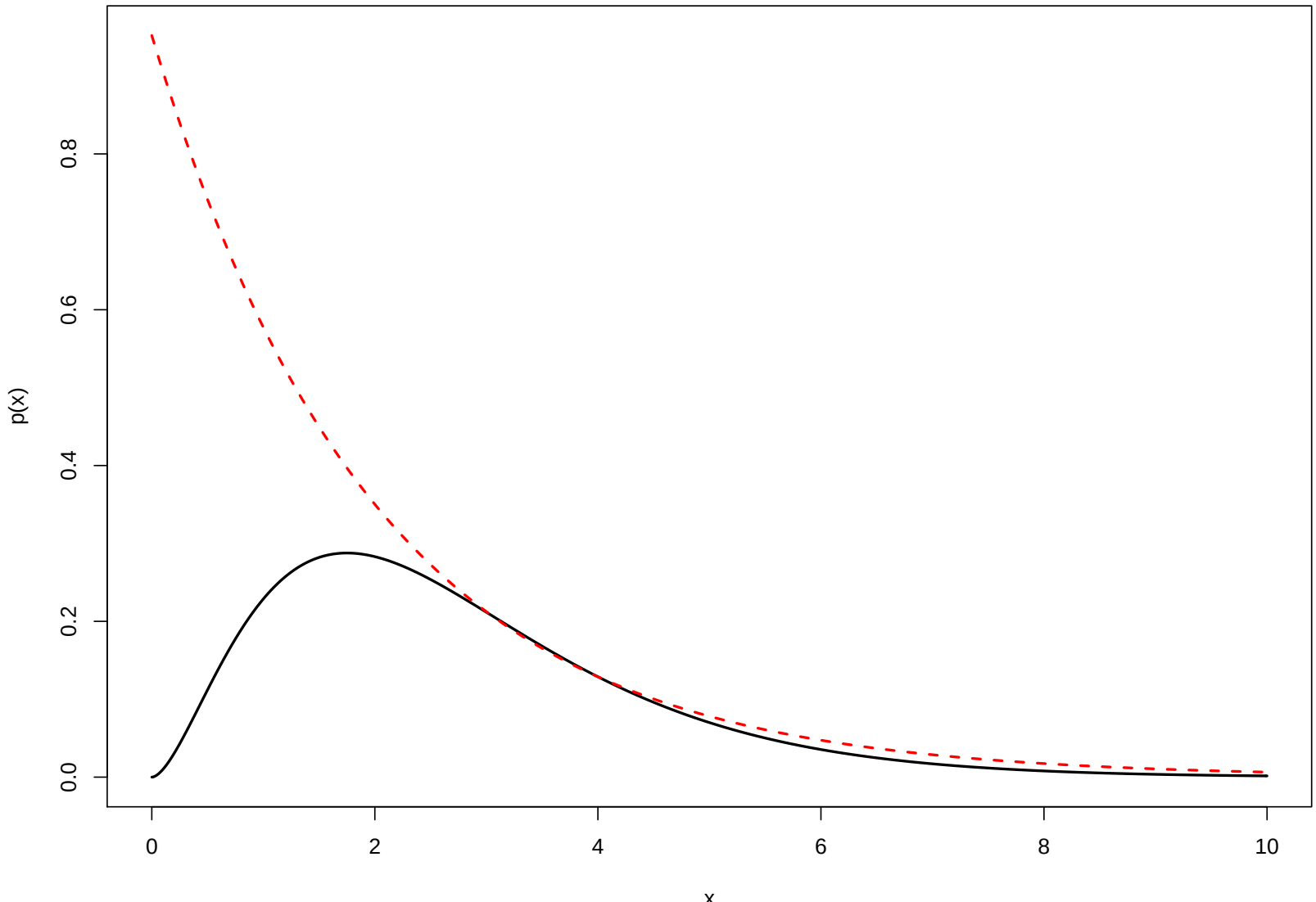
which achieves its unique maximum at

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

Thus rejection sampling can be carried out using this p_0.

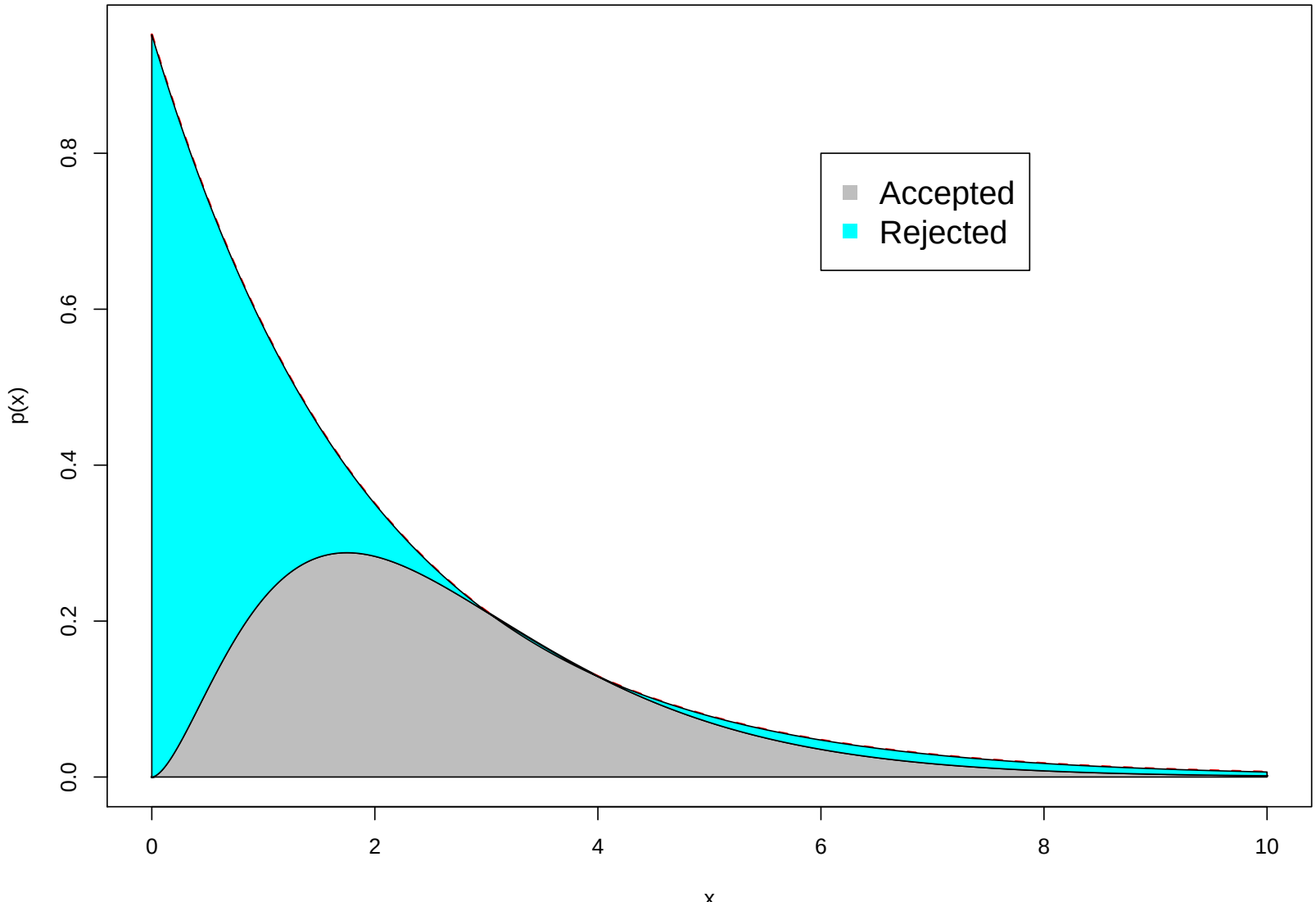
Adaptive Rejection sampling

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

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

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

$$l(x) \leqslant p(x) \leqslant Mp_0(x).$$

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

1. Generate x from p_0, and u from $Uniform(0, 1)$
2. Accept x if $u \leqslant l(x)/(Mp_0(x))$ and stop.
3. Accept x if $u \leqslant p(x)/(Mp_0(x))$ and stop.
4. Return to 1.

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

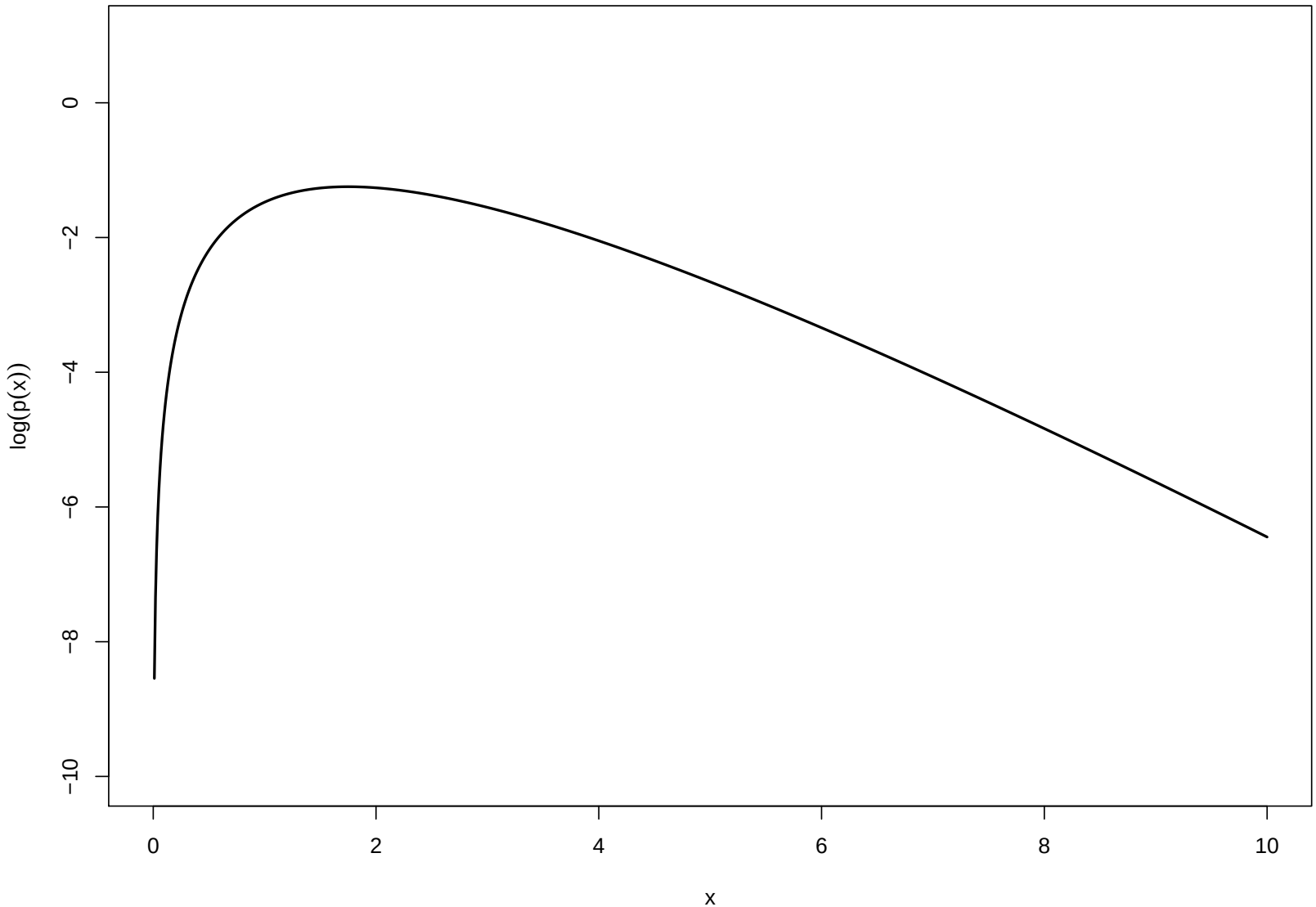
Adaptive Rejection sampling

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

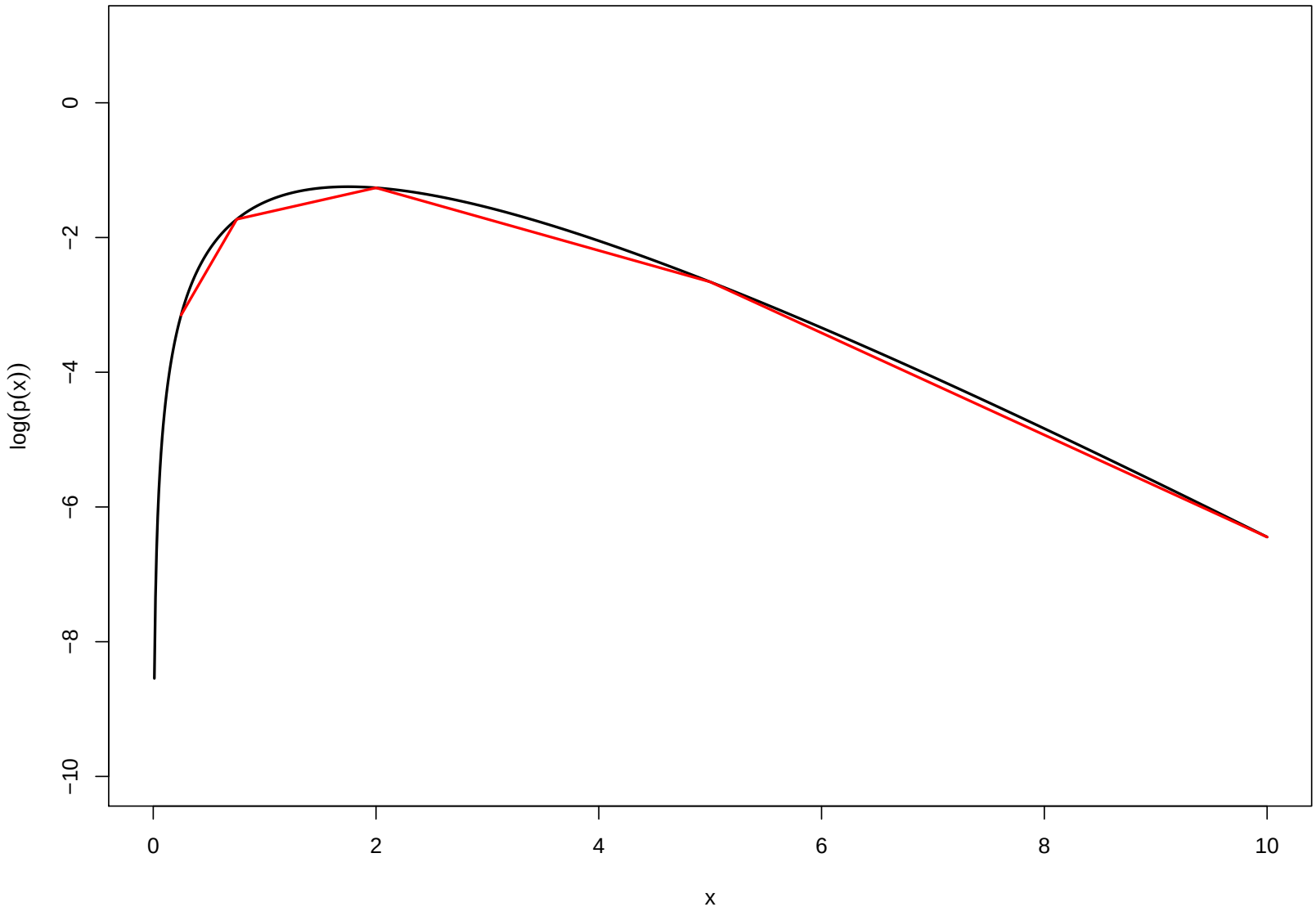
$$L_{i,i+1}(x) : (x_i, y_i) \longrightarrow (x_{i+1}, y_{i+1})$$

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

Adaptive Rejection sampling



Adaptive Rejection sampling



Adaptive Rejection sampling

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

$$\begin{aligned} \text{Lower} \quad &: \quad l_n(x) = L_{i,i+1}(x) \\ \text{Upper} \quad &: \quad u_n(x) = \min\{L_{i-1,i}(x), L_{i+1,i+2}(x)\} \end{aligned}$$

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

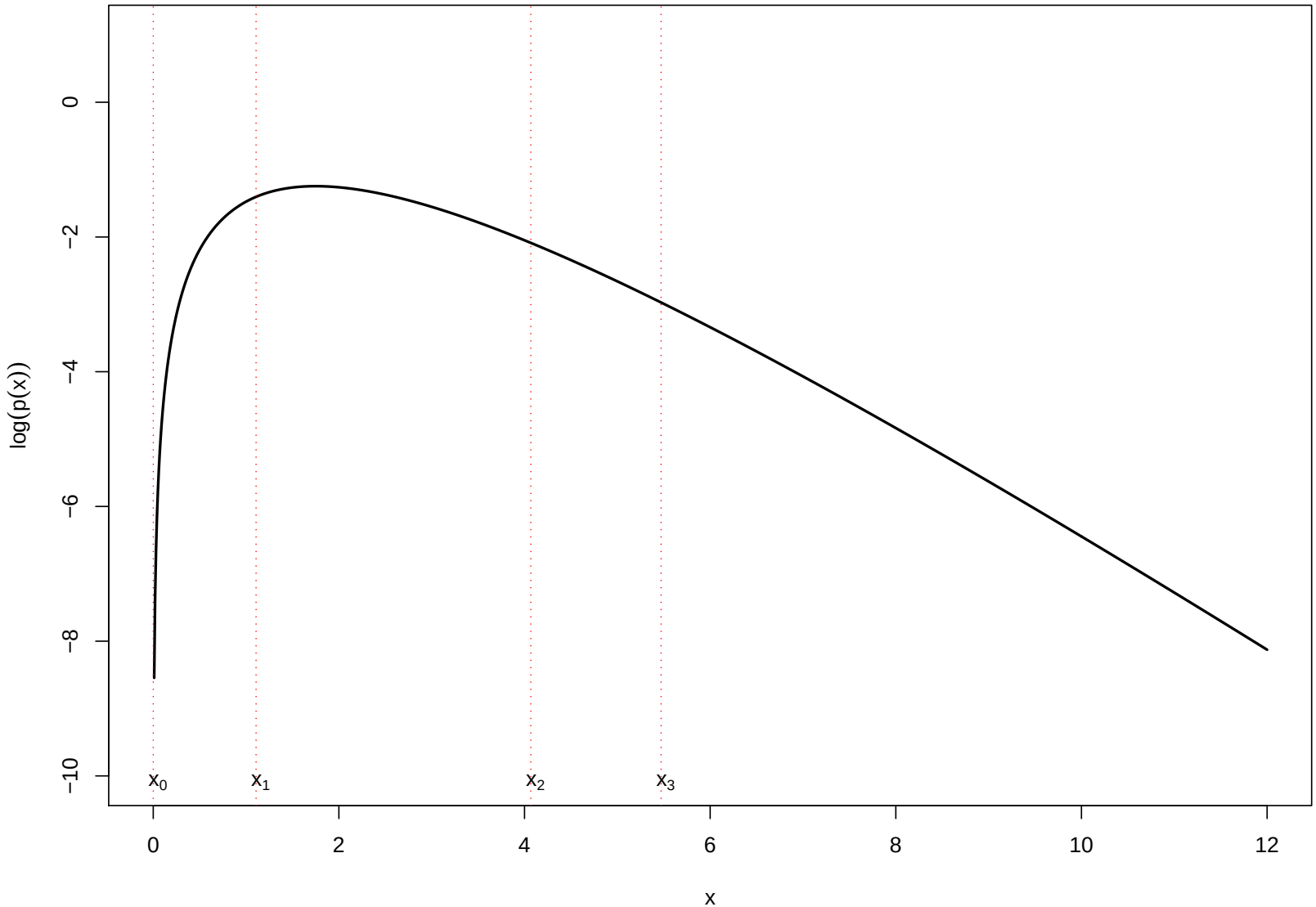
$$\exp\{l_n(x)\} \leqslant p(x) \leqslant \exp\{u_n(x)\} = M_n p_{0n}(x)$$

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

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

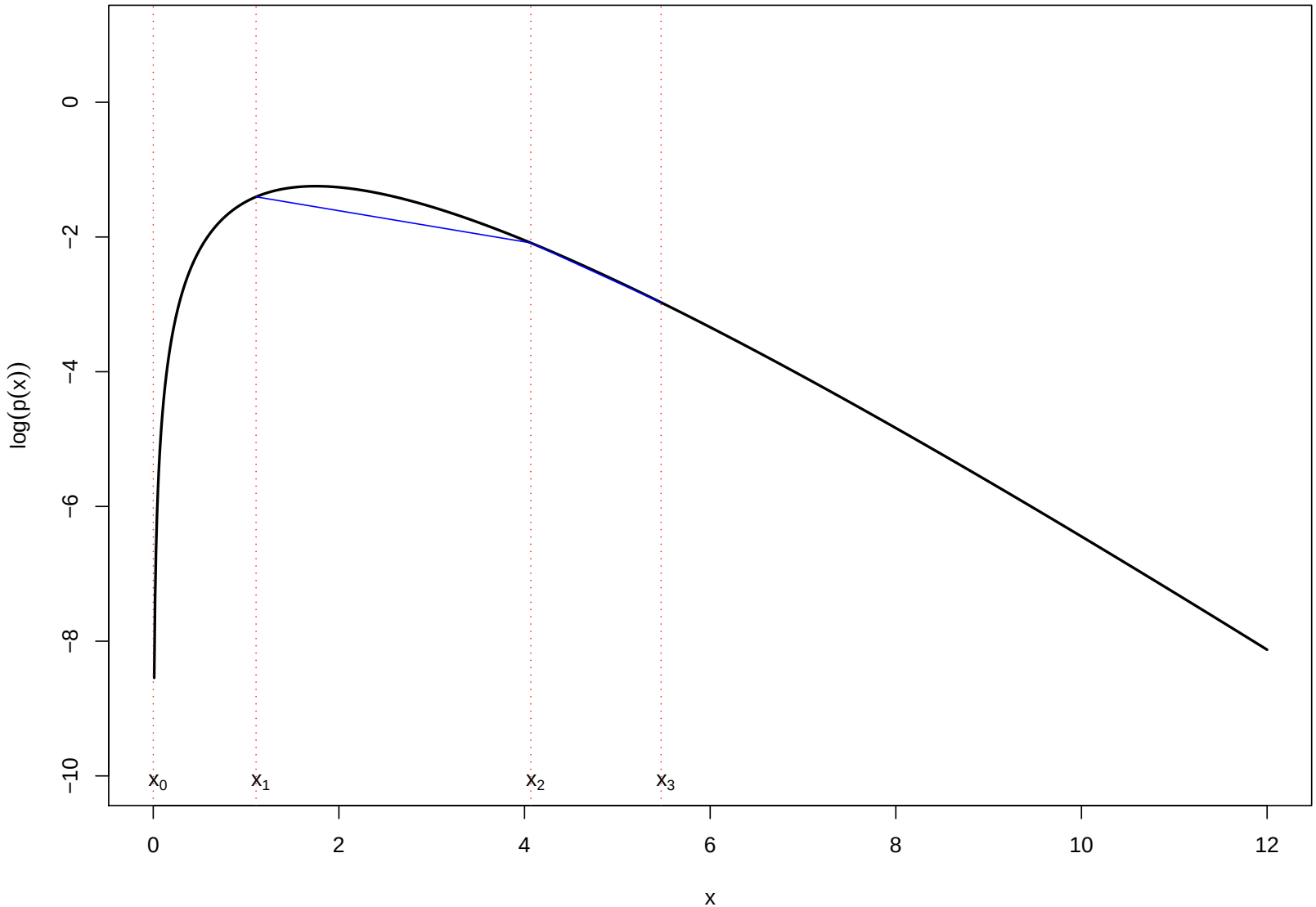
Adaptive Rejection sampling

Log-density:



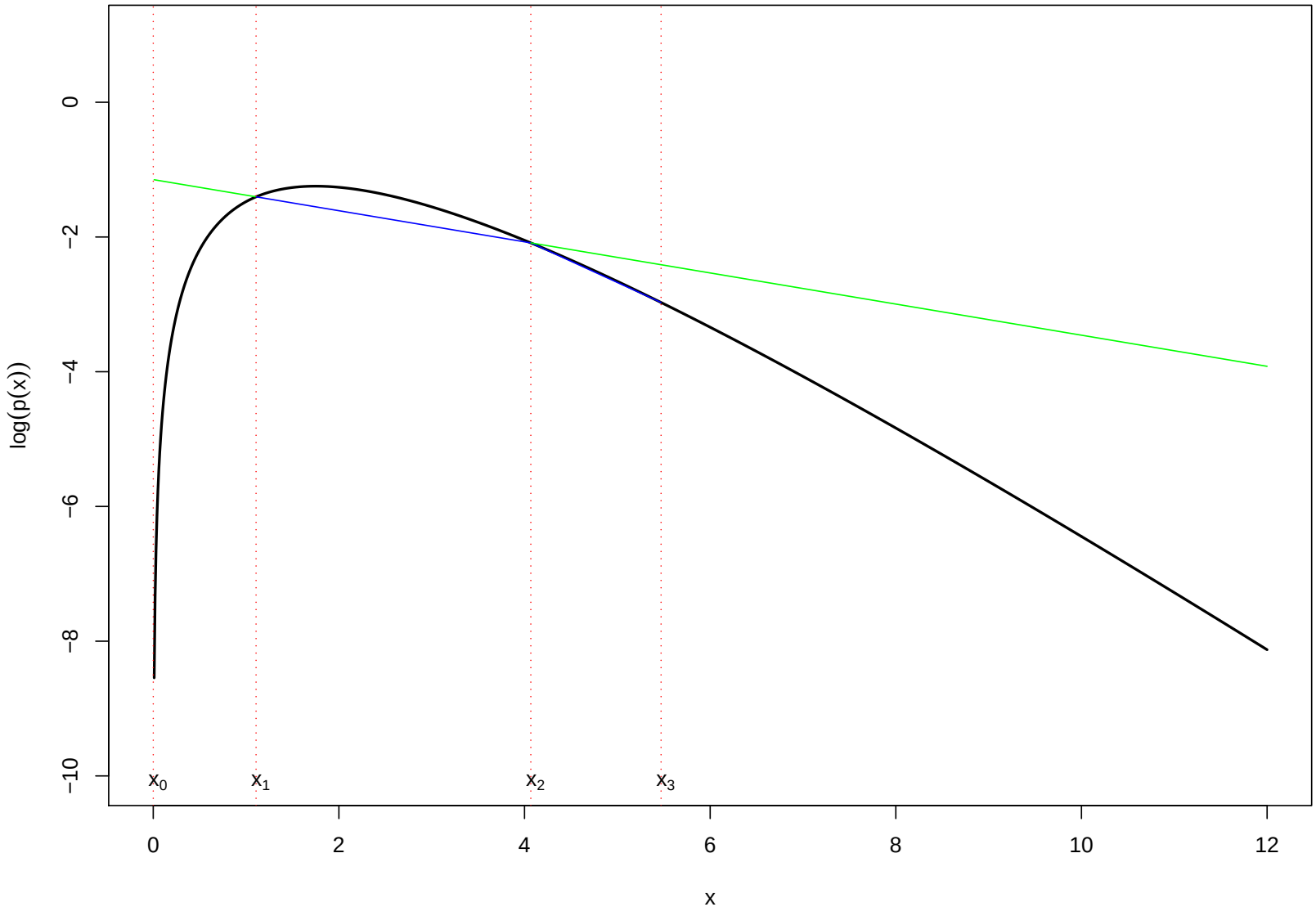
Adaptive Rejection sampling

Lower Envelope



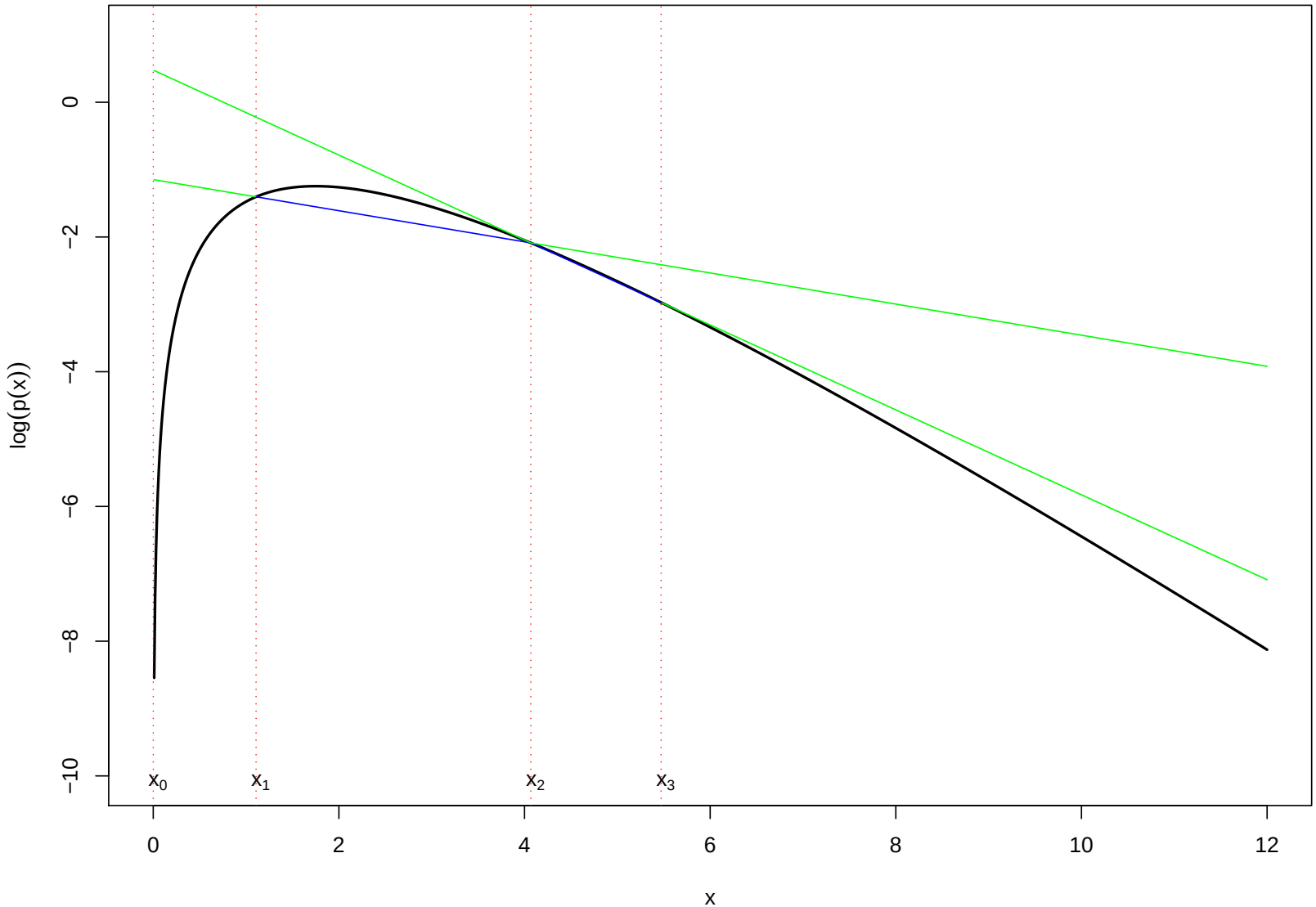
Adaptive Rejection sampling

Upper envelope (part 1):



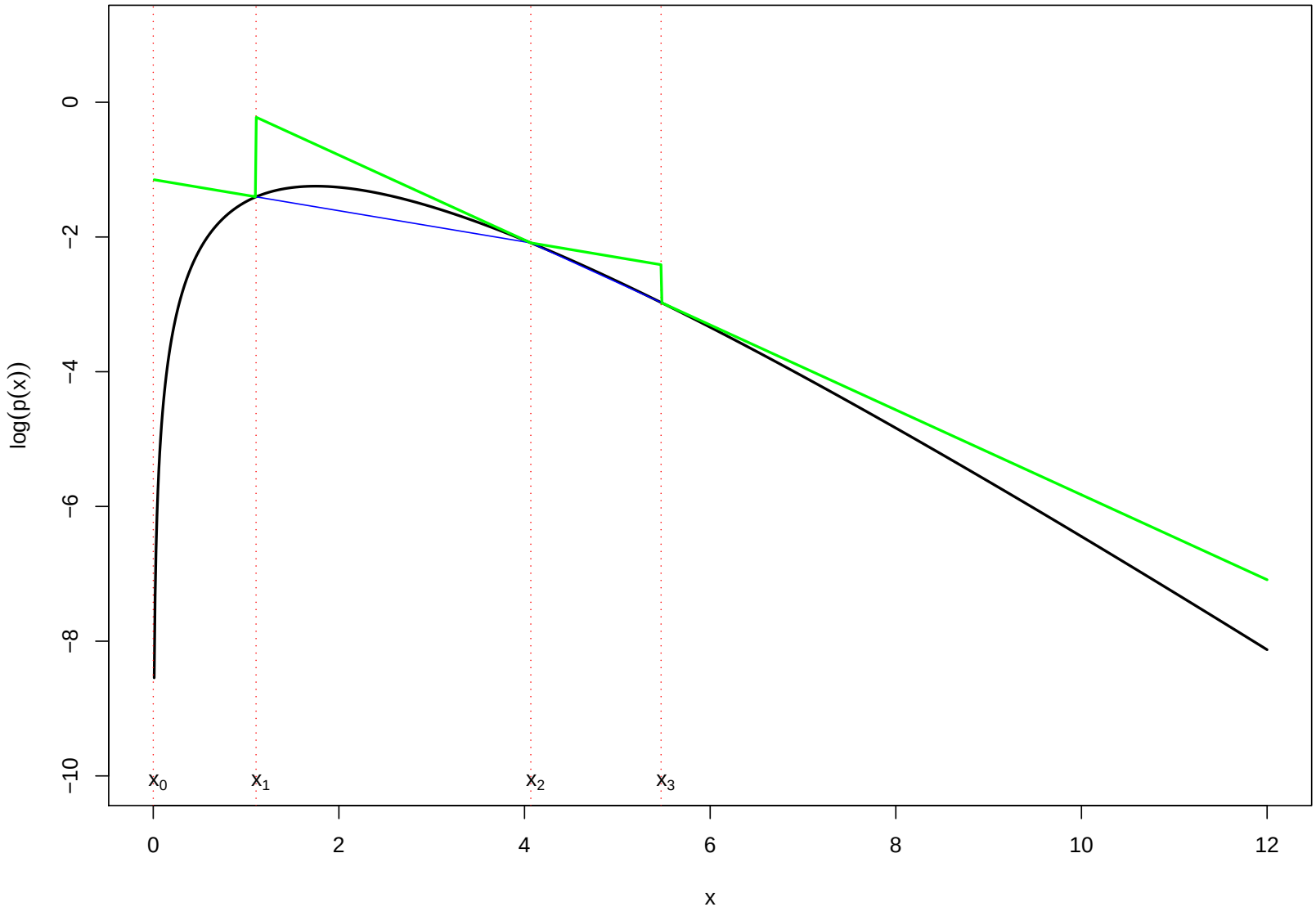
Adaptive Rejection sampling

Upper envelope (part 2):



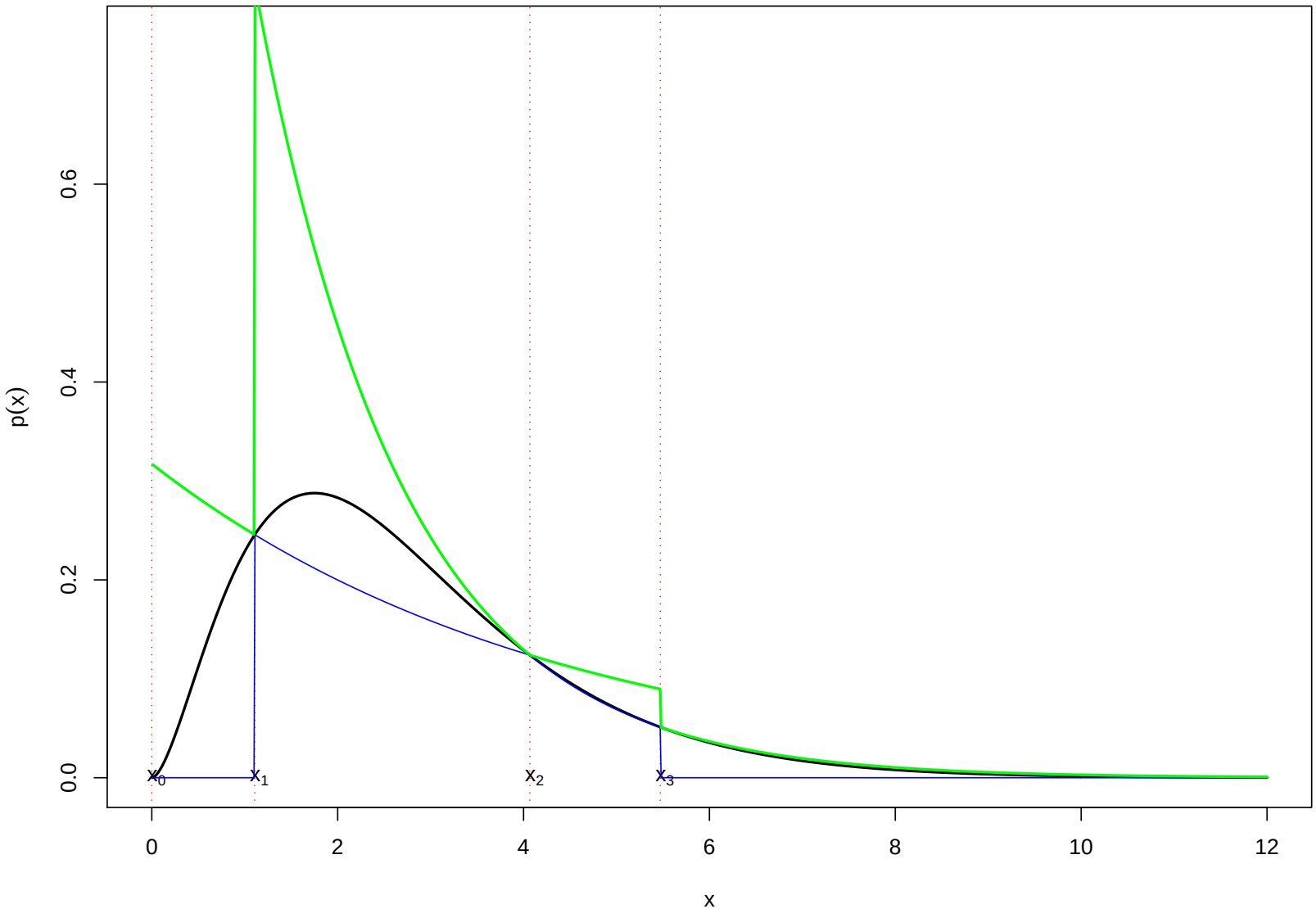
Adaptive Rejection sampling

Lower and Upper envelopes:



Adaptive Rejection sampling

Lower and Upper envelopes: original scale:



Adaptive Rejection sampling

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

1. Generate x from p_{n0}, and u from $Uniform(0, 1)$
2. Accept x if $u \leqslant \exp\{l_n(x)\}/(M_n p_{0n}(x))$, and go to 4.
3. Accept x if $u \leqslant p(x)/(M_n p_{0n}(x))$ and go to 4.
4. Update p_{n0} by updating the lower and upper envelopes taking into account the x and the function value $p(x)$, 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 $p_0(x)$
- the boundedness of the ratio $p(x)/p_0(x)$ is crucial in the construction of the procedure

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

Sampling Importance Resampling

Sampling Importance Resampling (SIR) can be used to sample (approximately) from density $p(x)$ by *re-weighting* and then *resampling* samples from $p_0(x)$:

1. Generate variates $x_1, \ldots, x_N$ from p_0.
2. Compute renormalized weights $w_1, \ldots, w_N$ given by

$$w_i = \frac{p(x_i)/p_0(x_i)}{\sum_{j=1}^{N} (p(x_i)/p_0(x_i))} \qquad i = 1, \ldots, N.$$

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

Sampling Importance Resampling

Returning to the previous example: suppose

$$\begin{aligned} p(x) &= \frac{1}{4}\phi(x+2.5) + \frac{3}{4}\phi(x-1) \\ p_0(x) &= \phi(x/\sigma) \end{aligned}$$

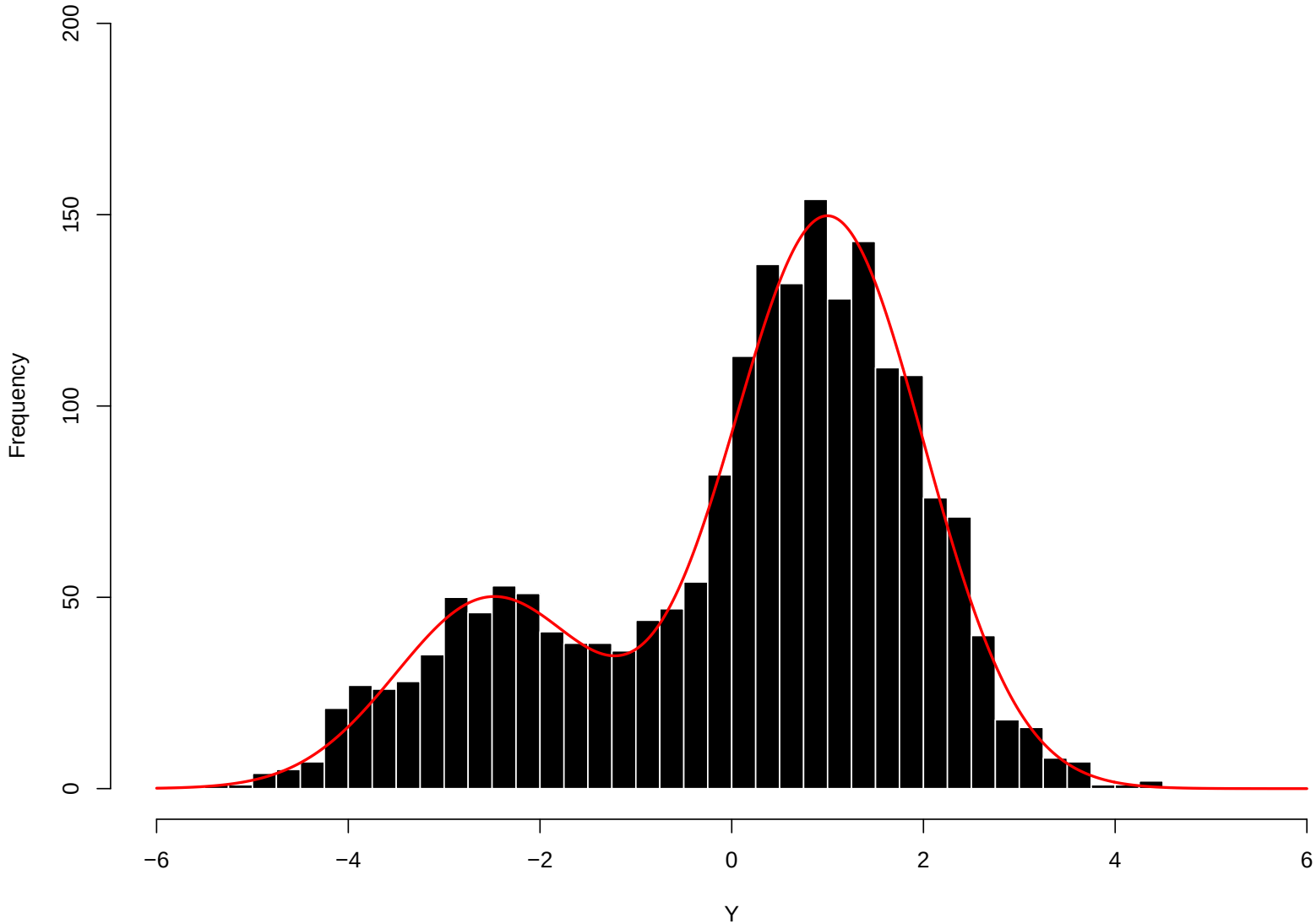
for some $\sigma > 0$.

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

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

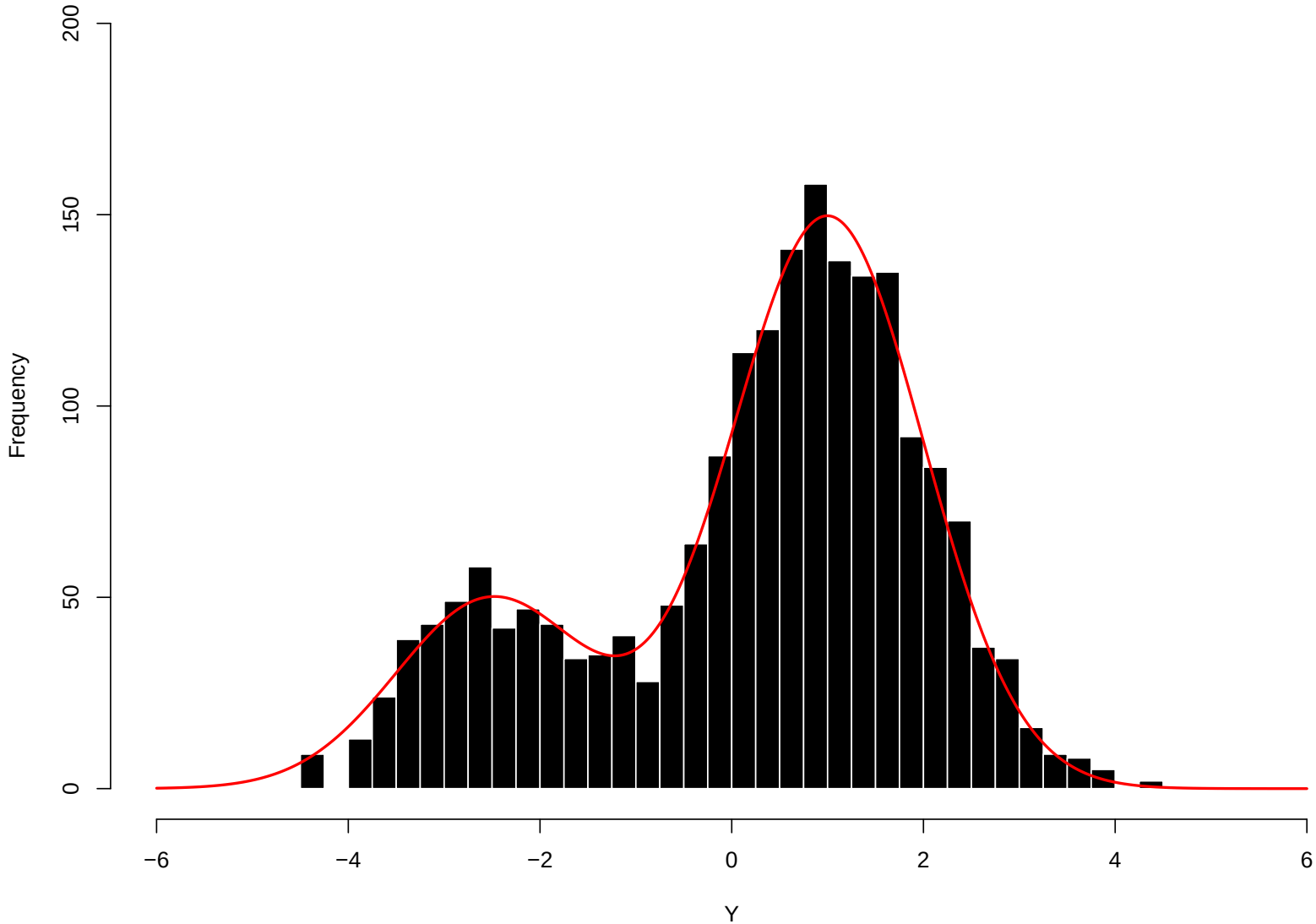
Sampling Importance Resampling

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



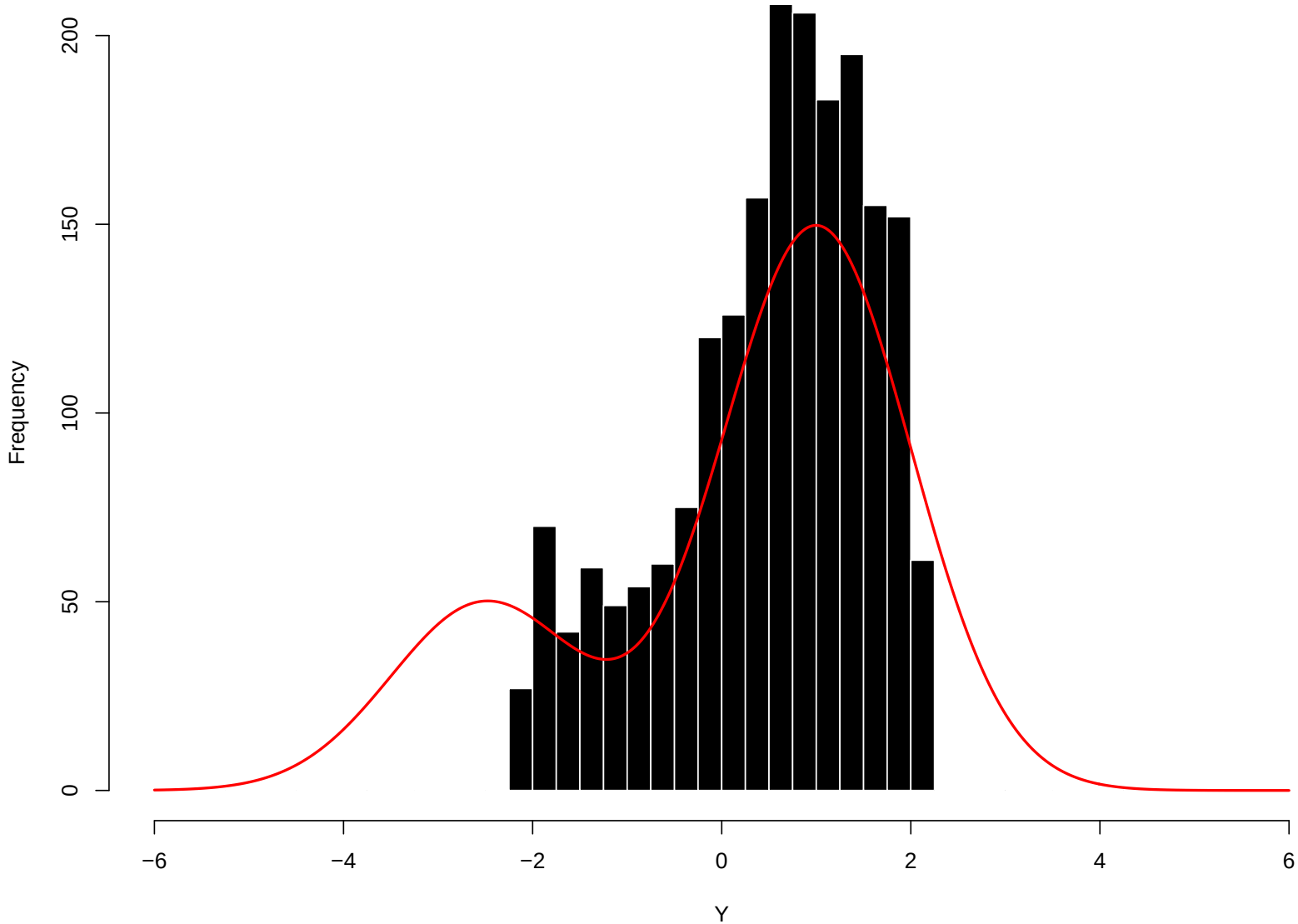
Sampling Importance Resampling

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



Sampling Importance Resampling

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



Variance Reduction: Antithetic Variables

Suppose that the integral

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

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 p, has variance

$$\frac{\mathbb{V}ar[g(X)]}{N}$$

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 p, and $2N_A = N$.

The variance of this estimator is

$$\frac{\mathbb{V}ar[g(X)]}{2N_A} + \frac{\mathbb{C}ov[g(X), g(Y)]}{2N_A}$$

Therefore if $\mathbb{C}ov[g(X), g(Y)]$ is **negative**, 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 $(X, Y)^\intercal \sim \mathcal{N}_2(\mu \mathbf{1}^\intercal, \Sigma)$, with variances 1 and

$$\mathbb{C}ov[X, Y] = \rho$$

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

$$\frac{\mathbb{V}ar[X]}{\mathbb{V}ar[X] + \mathbb{C}ov[X, Y]} = \frac{1}{1 - \rho}$$

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

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

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

$$\mathbb{V}ar[\widehat{I}_N(g)] + \lambda^2\mathbb{V}ar[\widehat{I}_N(g_0)] + 2\lambda\mathbb{C}ov[\widehat{I}_N(g), \widehat{I}_N(g_0)]$$

Variance Reduction: Control Variates

Therefore, choosing

$$\lambda = -\frac{\mathbb{C}ov[\widehat{I}_N(g), \widehat{I}_N(g_0)]}{\mathbb{V}ar[\widehat{I}_N(g_0)]}$$

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

$$(1 - \mathbb{C}orr[\widehat{I}_N(g), \widehat{I}_N(g_0)]^2)\mathbb{V}ar[\widehat{I}_N(g)] \leqslant \mathbb{V}ar[\widehat{I}_N(g)]$$

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

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

Consider

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

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

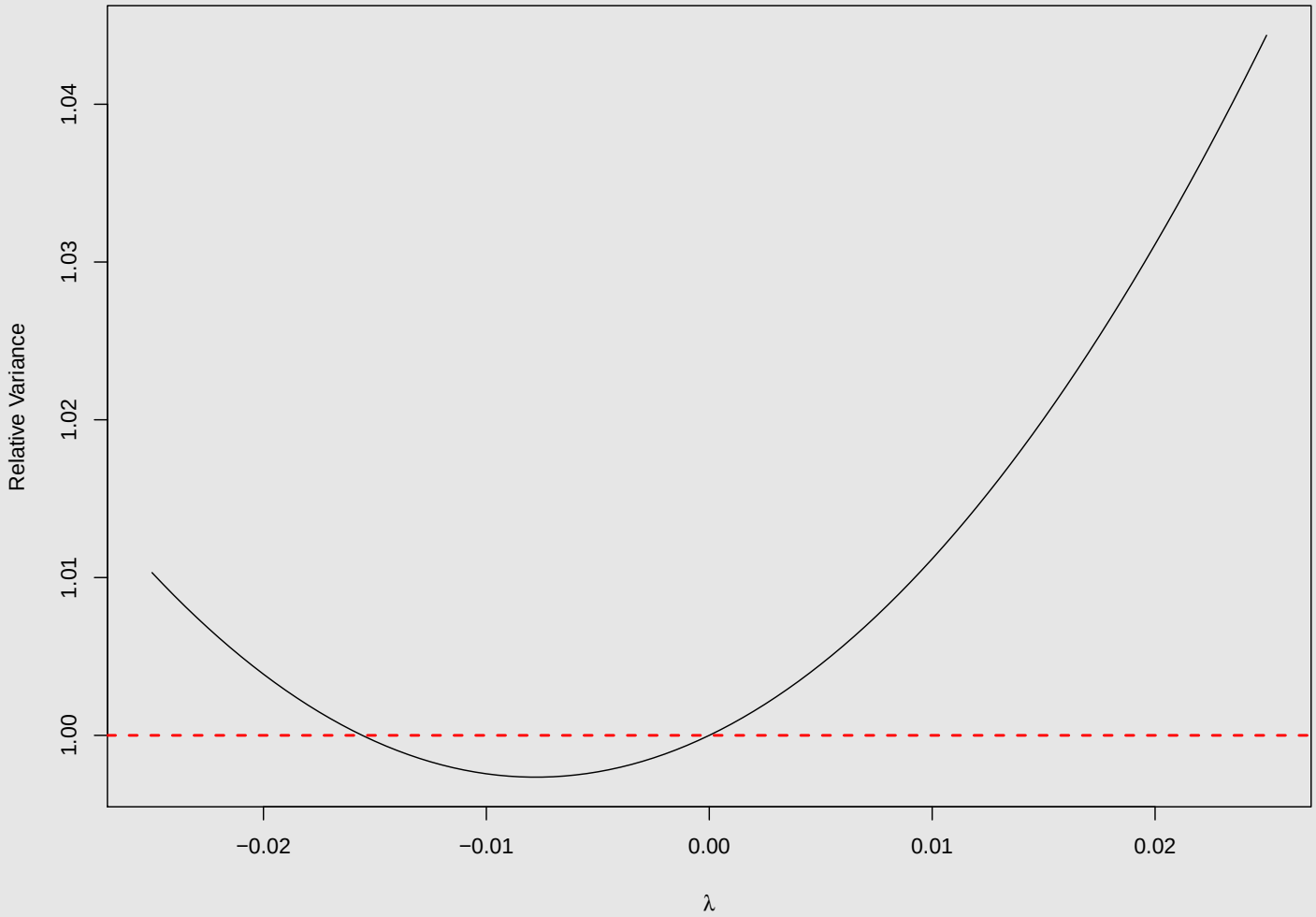
$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{(x_0,\infty)}(X_i) + \lambda\left(\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{(0,\infty)}(X_i) - \frac{1}{2}\right)$$

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)

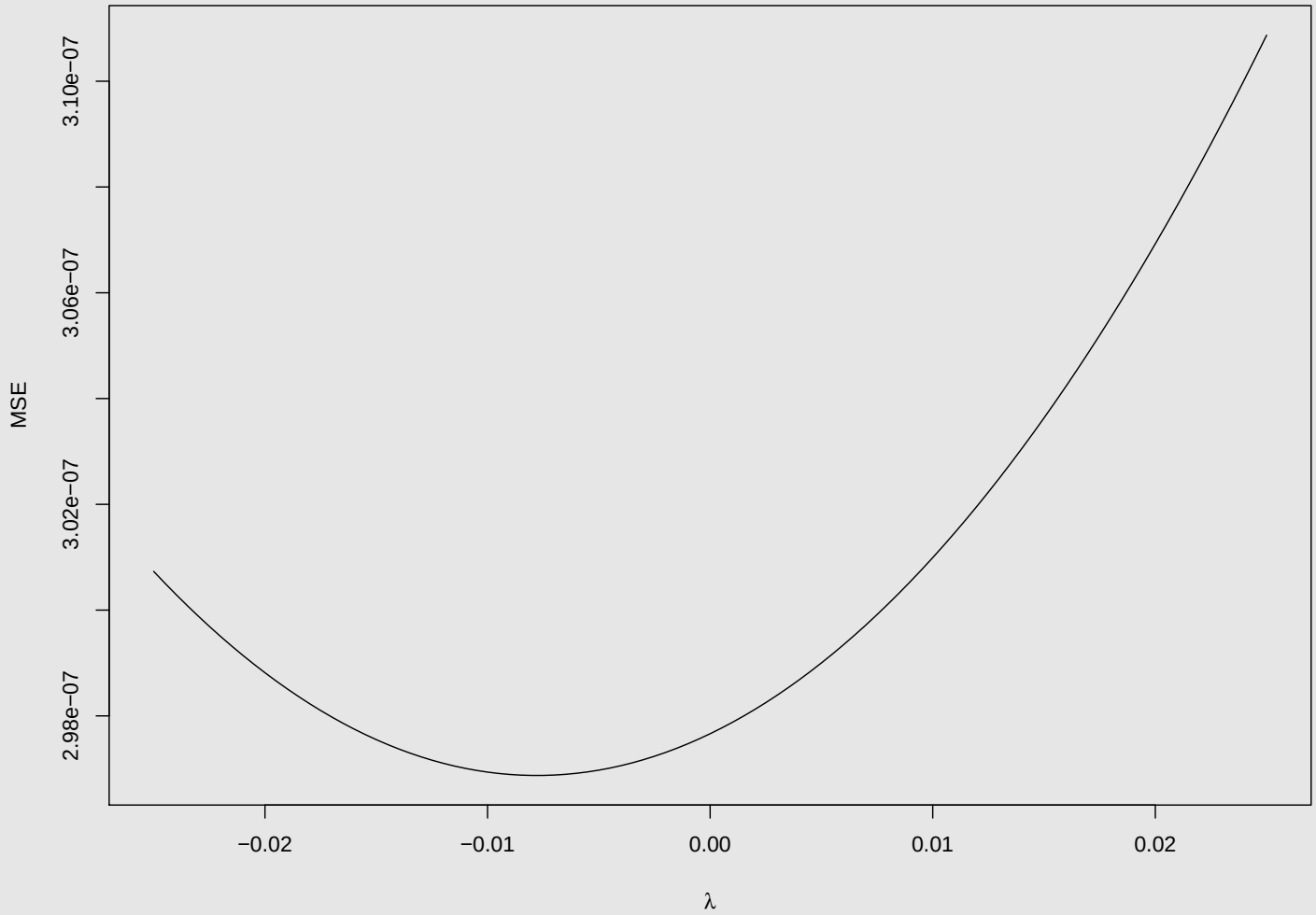
$$\mathrm{MSE}(\lambda) = \mathbb{E}[(I(g) - \widehat{I}_N(g; g_0, \lambda))^2]$$

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: $x_0 = 2.5$, $N = 20000$.



Markov chain Monte Carlo

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

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

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

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

A brief introduction to Markov chains

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

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

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, and its *transition matrix* P, a $d \times d$ *stochastic* matrix whose rows sum to one, such that

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

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

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

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

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

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, $\mathcal{R}$,

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

- *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

$$\mathbb{P}[T_i = \infty] = 0$$

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

The *stationary* or *equilibrium 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,

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

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) Realizations of $\{X_t\}$ exhibit 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 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.

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

$$\sigma^2(g) = \mathbb{V}ar_\pi[g(X_0)] + 2\sum_{t=1}^{\infty} \mathbb{C}ov_\pi[g(X_0), g(X_t)].$$

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

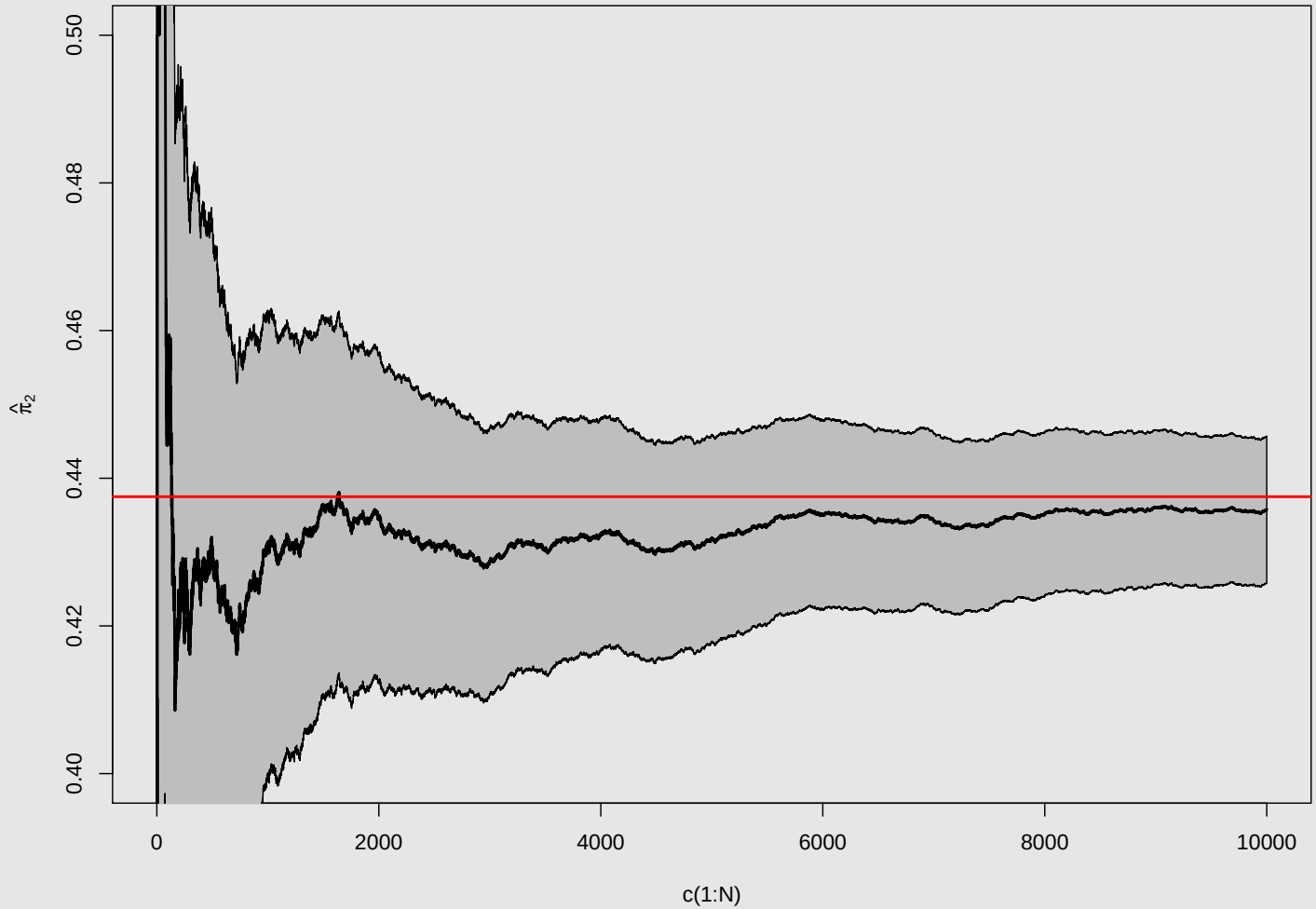
$$\pi_1 P_{12} = \frac{9}{16} \times \frac{7}{10} = \frac{63}{160} \qquad \pi_2 P_{21} = \frac{7}{16} \times \frac{9}{10} = \frac{63}{160}$$

so this chain is reversible.

Discrete Markov chain: example

Example ($d = 2$)

Sample average relative frequency of being in state 2 over 10000 steps.



Constructing reversible chains

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

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

Suppose

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

Then

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

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

In this case, we have the transition probabilities given as follows:

$$\begin{aligned} P_{12} &= \mathbb{P}[X_{t+1} = 2 | X_t = 1] = q_{12}\alpha_{12} \\ P_{21} &= \mathbb{P}[X_{t+1} = 1 | X_t = 2] = q_{21}\alpha_{21} \end{aligned}$$

to therefore

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

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

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*

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

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

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

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

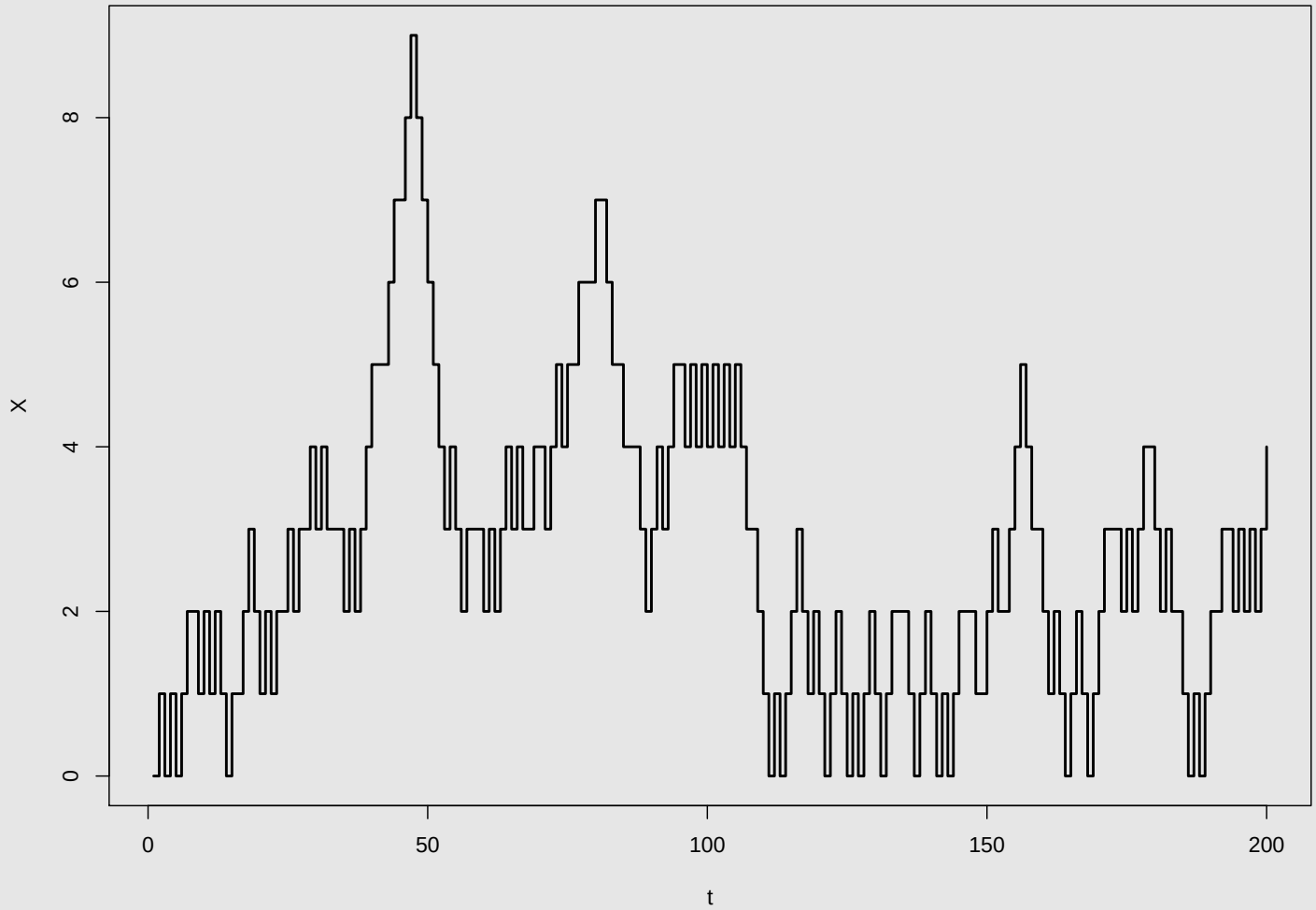
$$q_{ij} = \begin{cases} 1 & i = 0, j = 1 \\ \frac{1}{2} & i \geqslant 1, j = i-1, i+1 \\ 0 & \text{otherwise} \end{cases}$$

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)

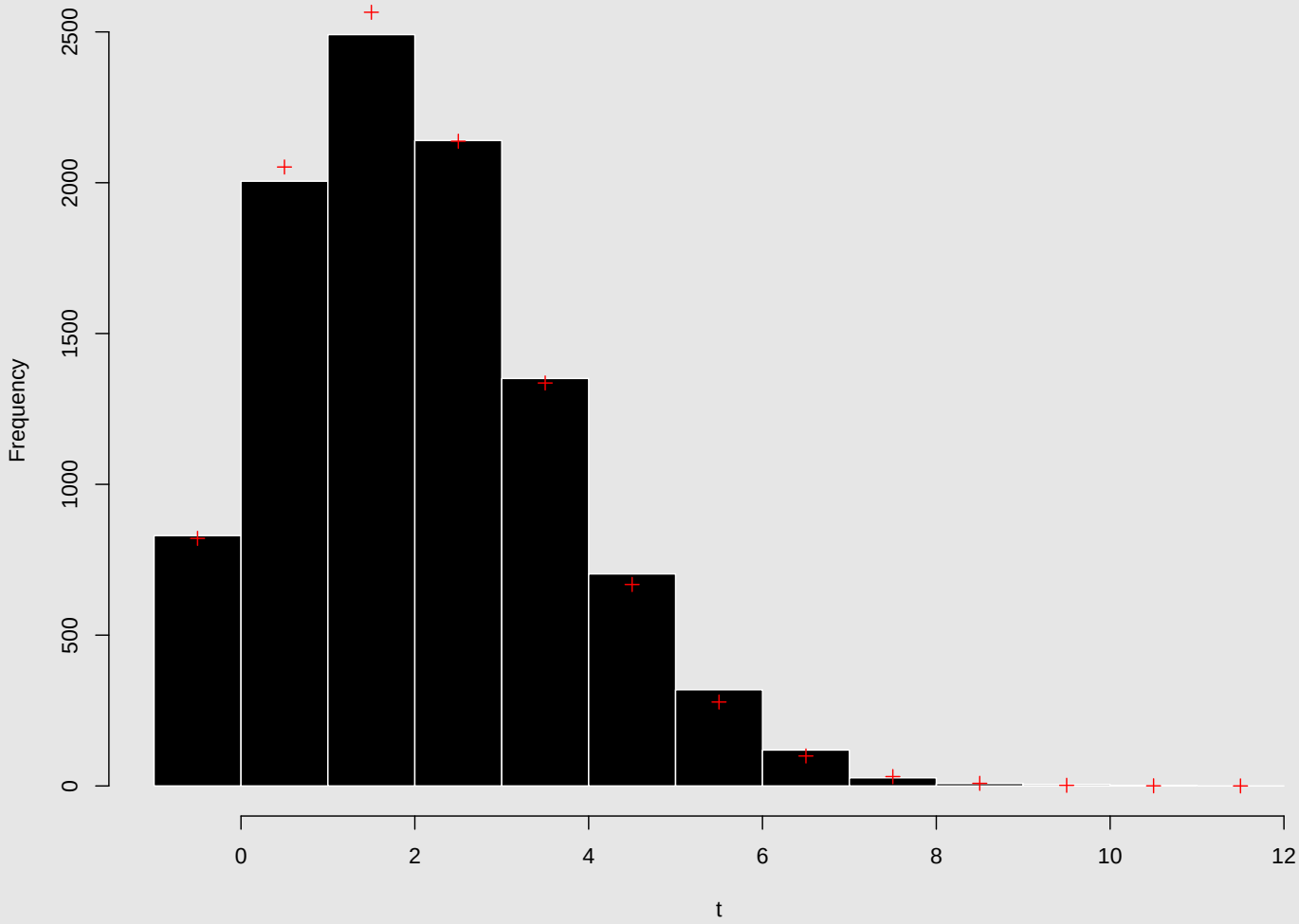
First 200 steps of the chain starting at $X_0 = 0$ with $\lambda = 2.5$.



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

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

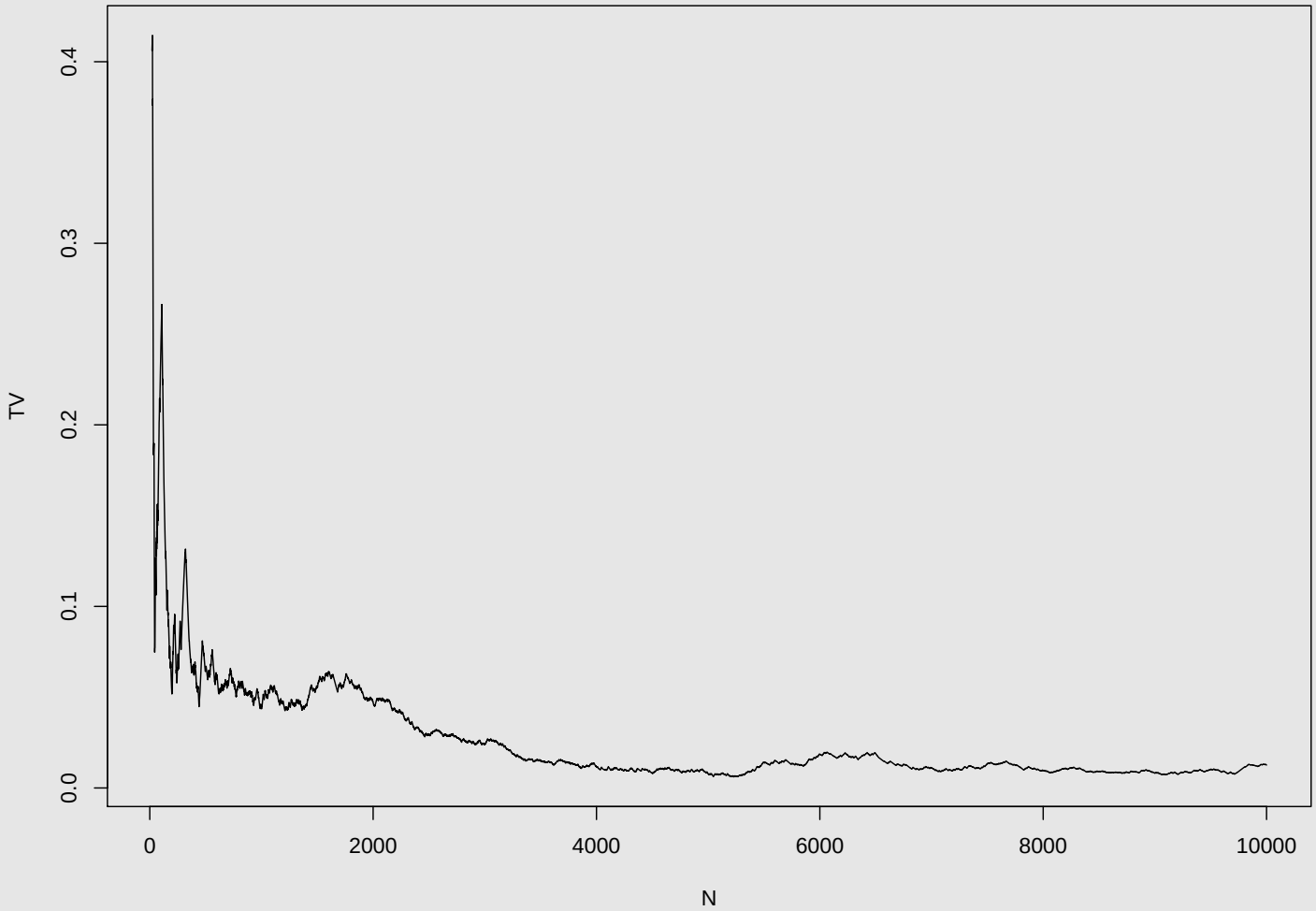
Here we take f_1 to be the true Poisson mass function π, and f_2 to be the estimated mass function $\widehat{\pi}$.

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 (defined with respect to Lebesgue measure) the total variation distance is given by

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

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 $d_H(f_1, f_2)$

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

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

Continuous State-Space Markov Chains

The theory above extends (reasonably straightforwardly) to continuous state spaces, that is, the countable state set $\{s_1, ..., s_d\}$ is replaced by a continuum of possible values, $\mathbb{X}$.

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

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

A reversible chain must satisfy detailed balance

$$\pi(x)P(x, z) = \pi(z)P(z, x)$$

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.

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

$$q(z, x) = q(z \longrightarrow x) > 0$$

for all x, z.

In fact, this can be relaxed to the condition that requires $Q^n(x, z) > 0$ 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

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

so that q is symmetric in its arguments, then

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

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

Another common choice is the *independence Metropolis-Hastings algorithm*, where

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

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

$$\mathbb{P}[Z \leqslant z|X = x] = \mathbb{P}[|Y| \leqslant z|X = x] = \mathbb{P}[-z \leqslant Y \leqslant z|X = x]$$

so therefore

$$\mathbb{P}[Z \leqslant z|X = x] = \Phi((z - x)/\sigma_q) - \Phi((-z - x)/\sigma_q)$$

and, on differentiation wrt z,

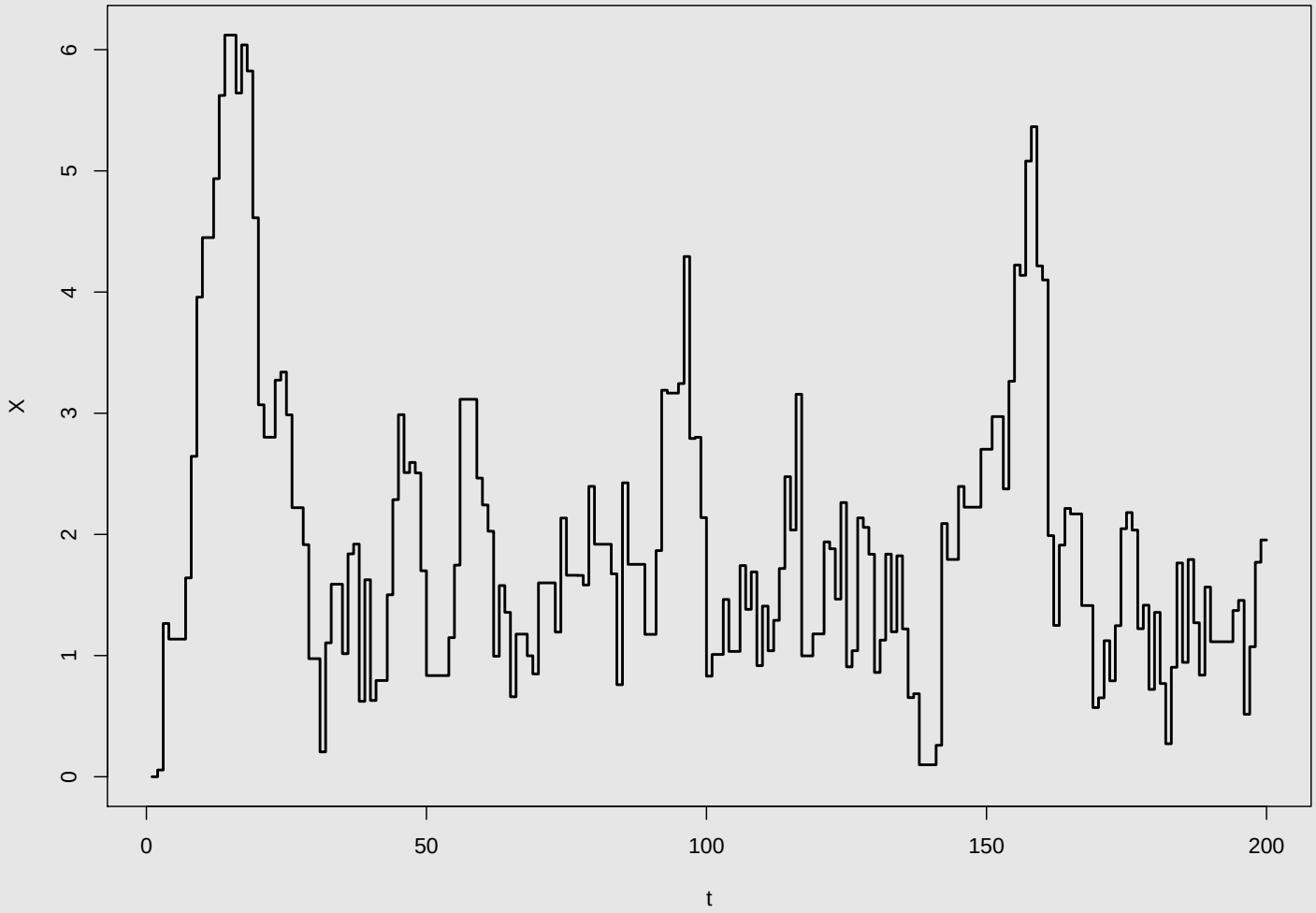
$$q(x, z) = \frac{1}{\sigma_q}(\phi((z - x)/\sigma_q) + \phi((-z - x)/\sigma_q)) = q(z, x)$$

as ϕ is an even function.

Metropolis-Hastings algorithm

Example (Gamma density)

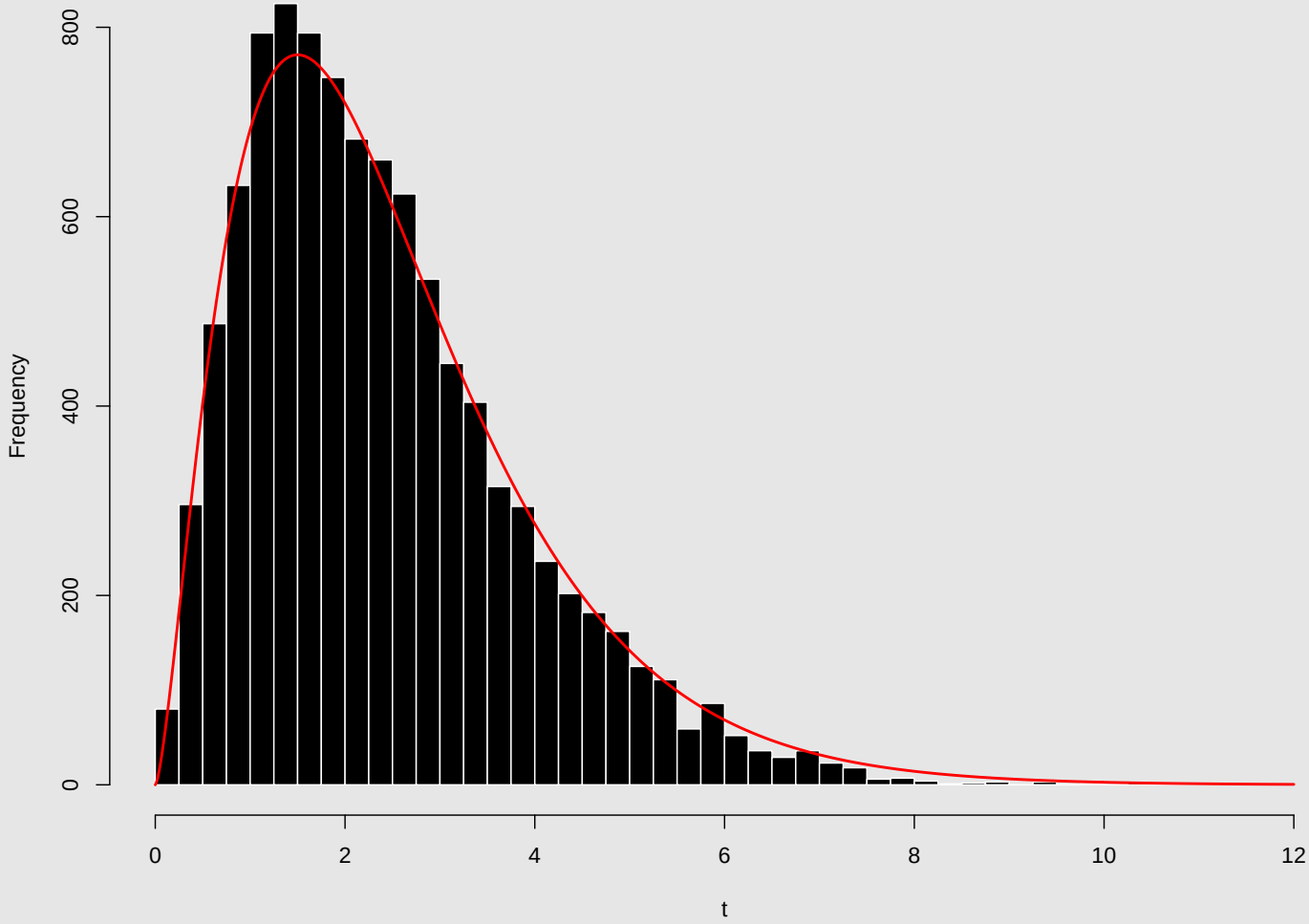
First 200 steps of the chain starting at $X_0 = 0$ with $\gamma = 2.5$, $\sigma_q = 1$.



Metropolis-Hastings algorithm

Example (Gamma density)

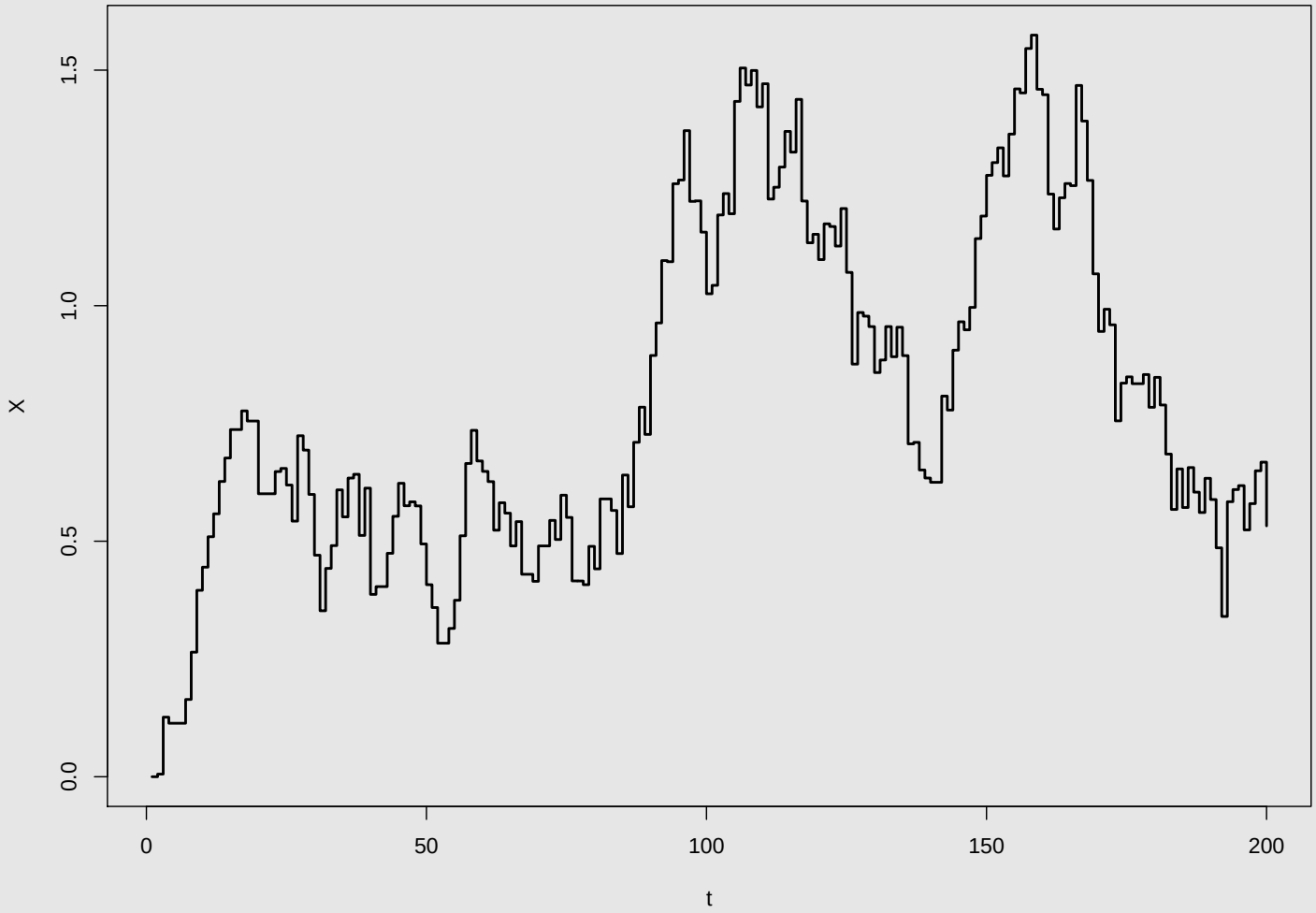
Histogram of states visited over $N = 10000$ steps (with Gamma density (solid))



Metropolis-Hastings algorithm

Example (Gamma density)

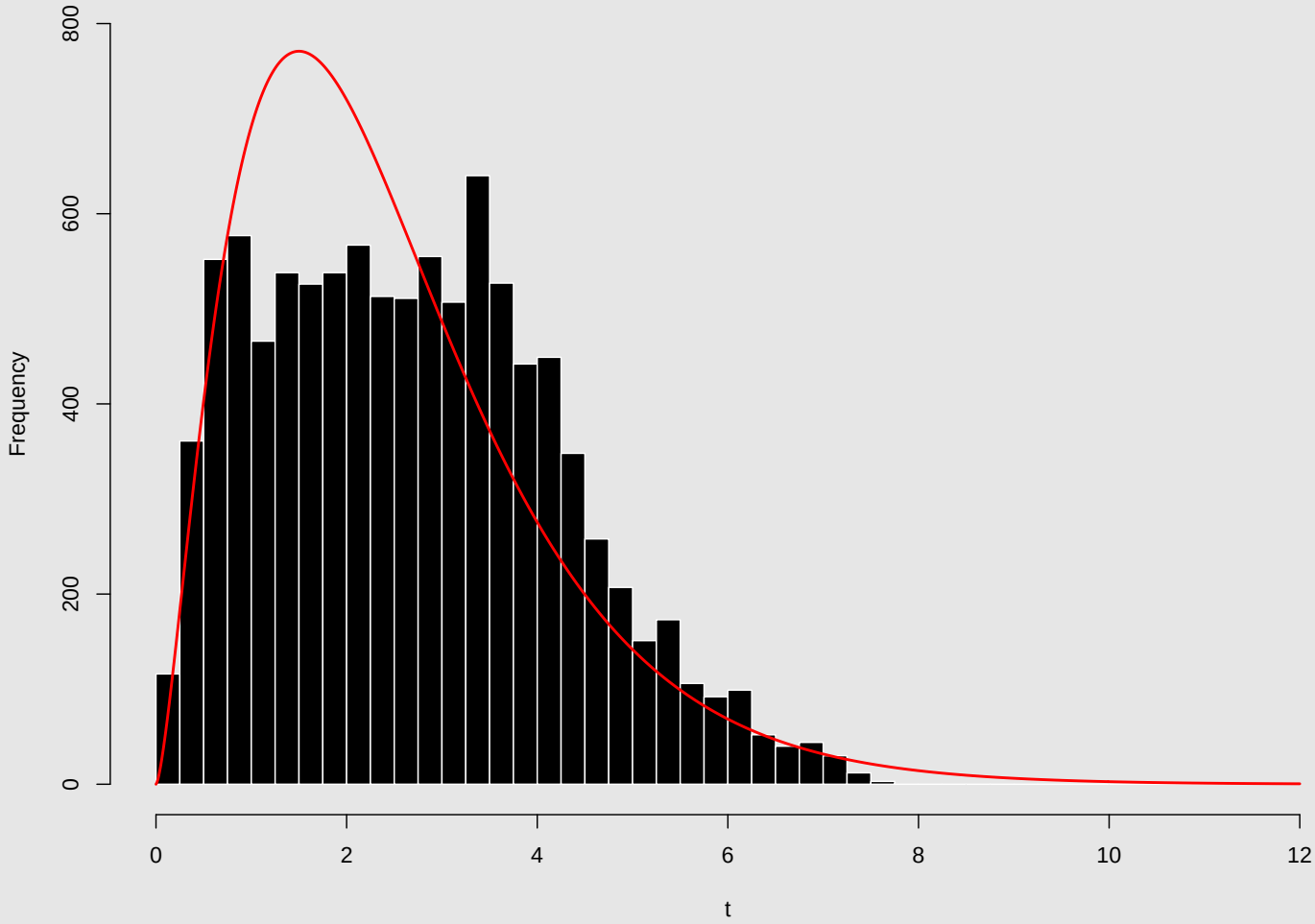
First 200 steps of the chain starting at $X_0 = 0$ with $\gamma = 2.5$, $\sigma_q = 0.1$.



Metropolis-Hastings algorithm

Example (Gamma density)

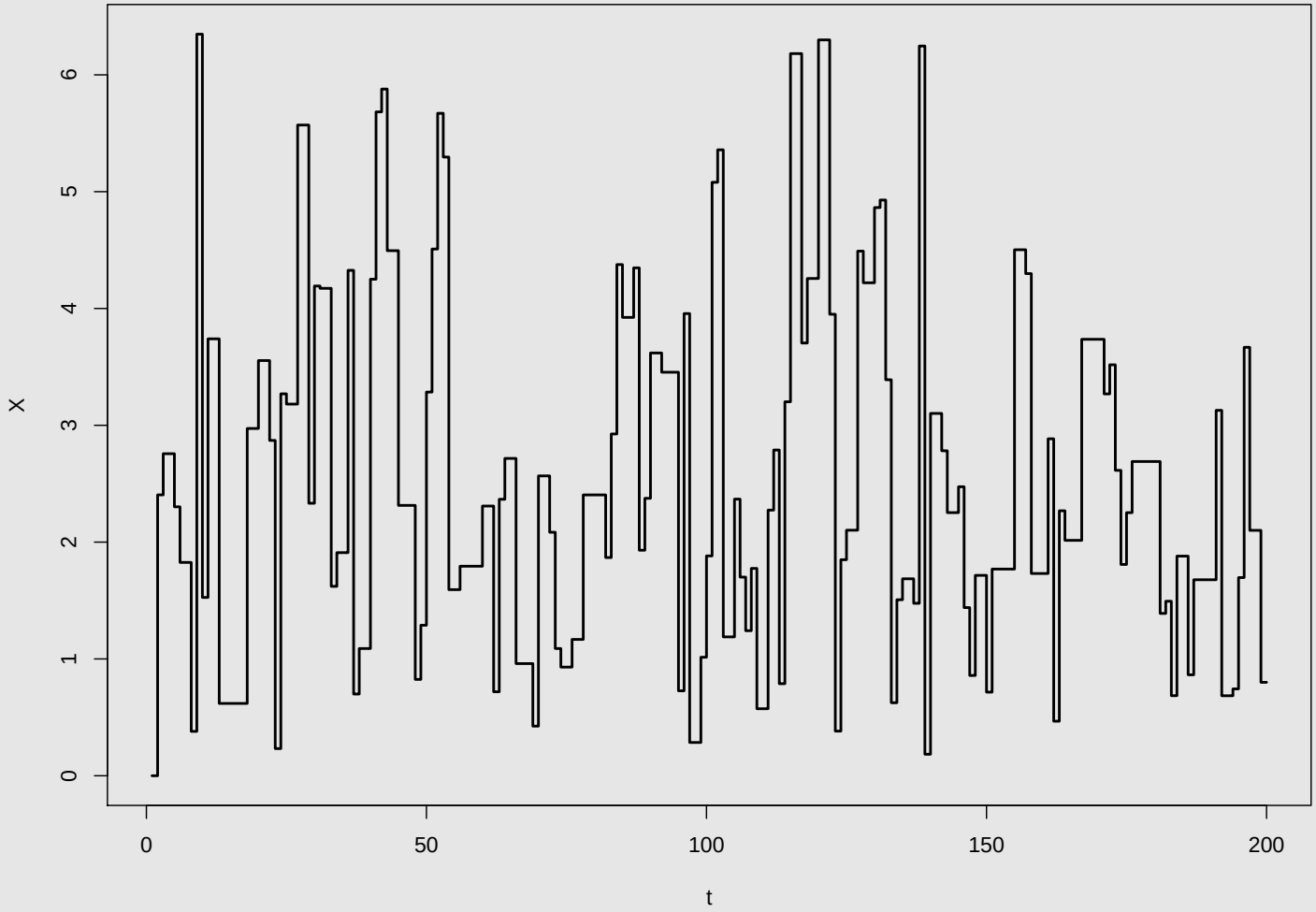
Histogram of states visited over $N = 10000$ steps (with Gamma density (solid))



Metropolis-Hastings algorithm

Example (Gamma density)

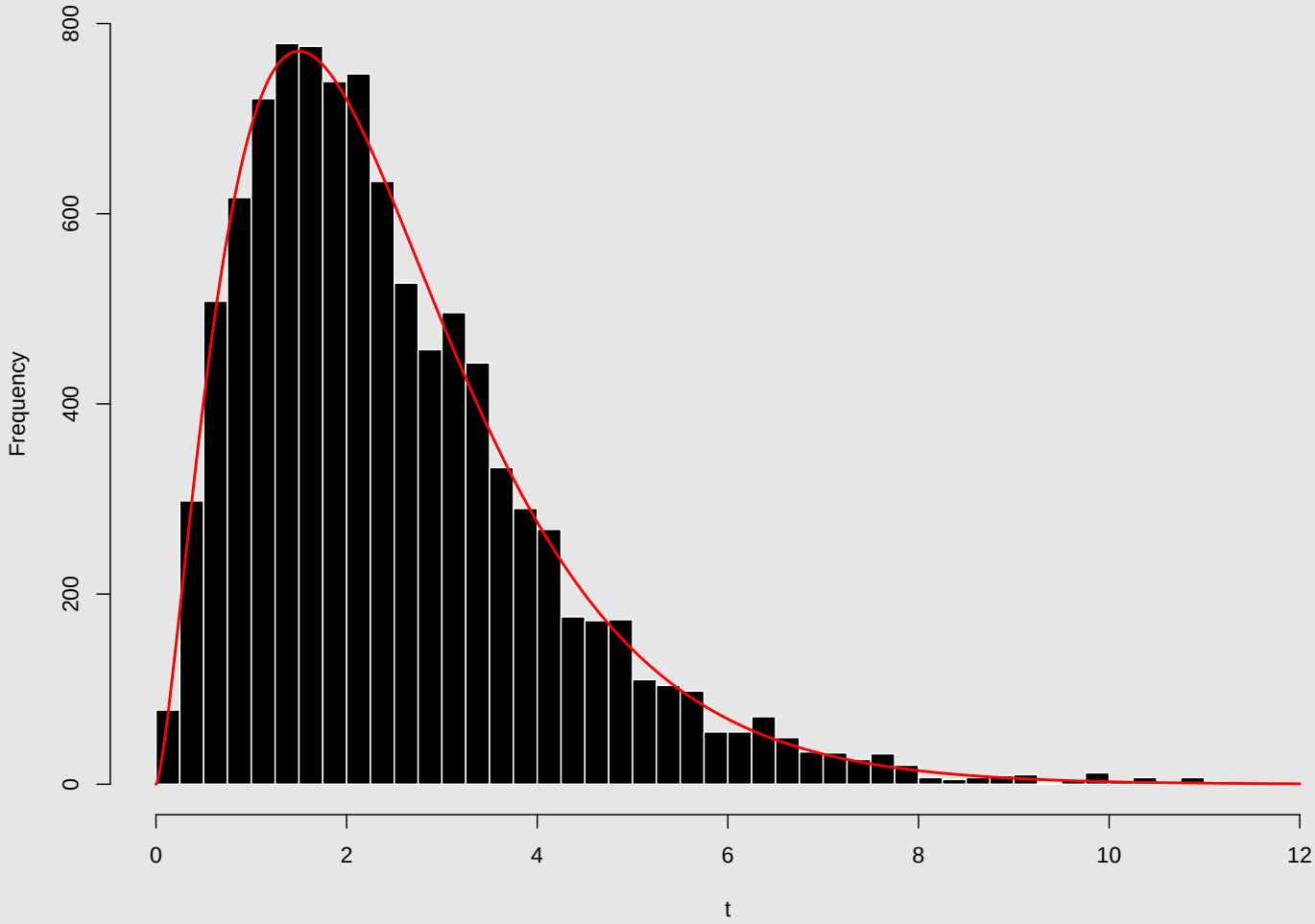
First 200 steps of the chain starting at $X_0 = 0$ with $\gamma = 2.5$, $\sigma_q = 3$.



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

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

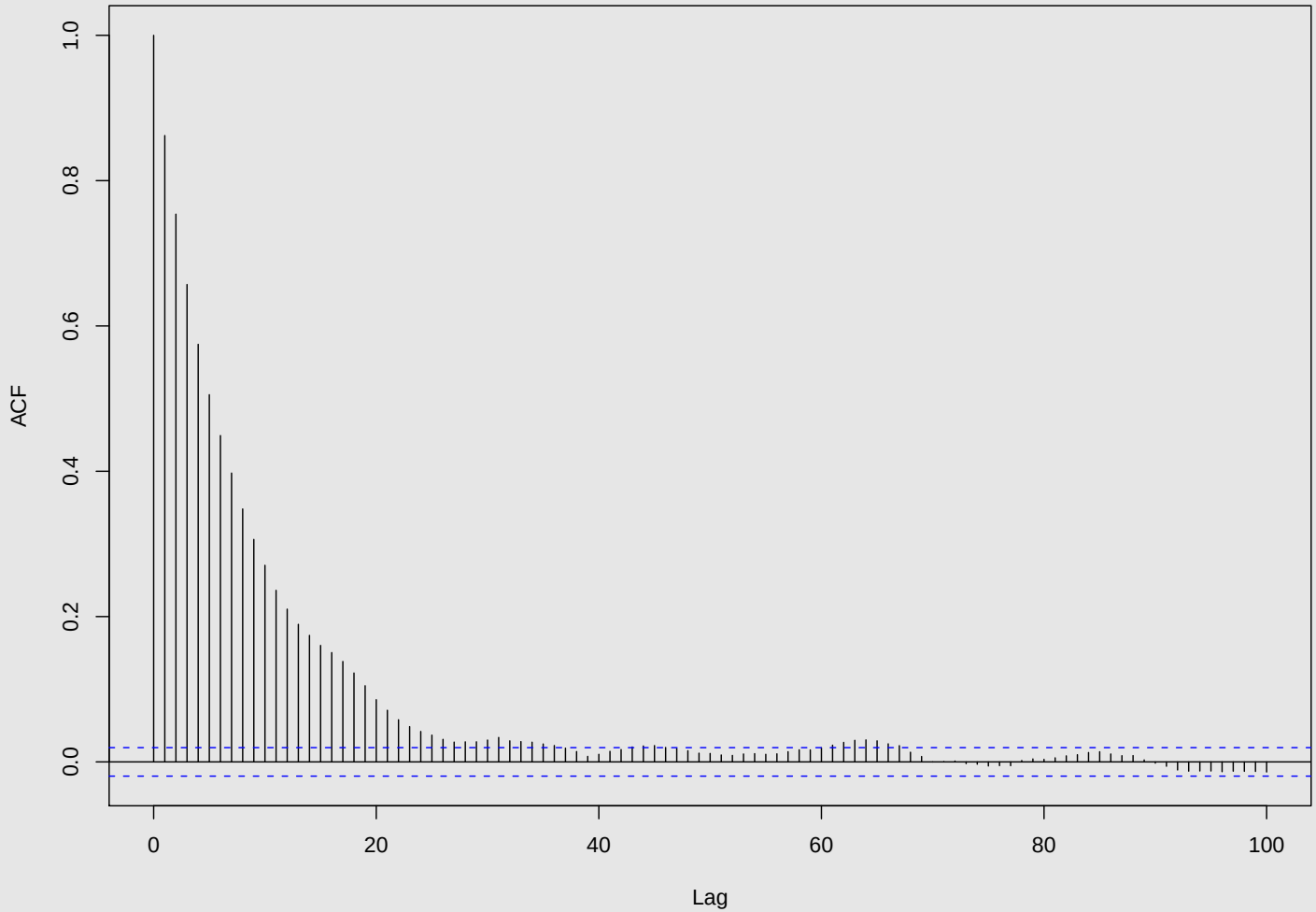
for $k = 0, 1, 2 \ldots$.

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

Metropolis-Hastings algorithm

Example (Gamma density)

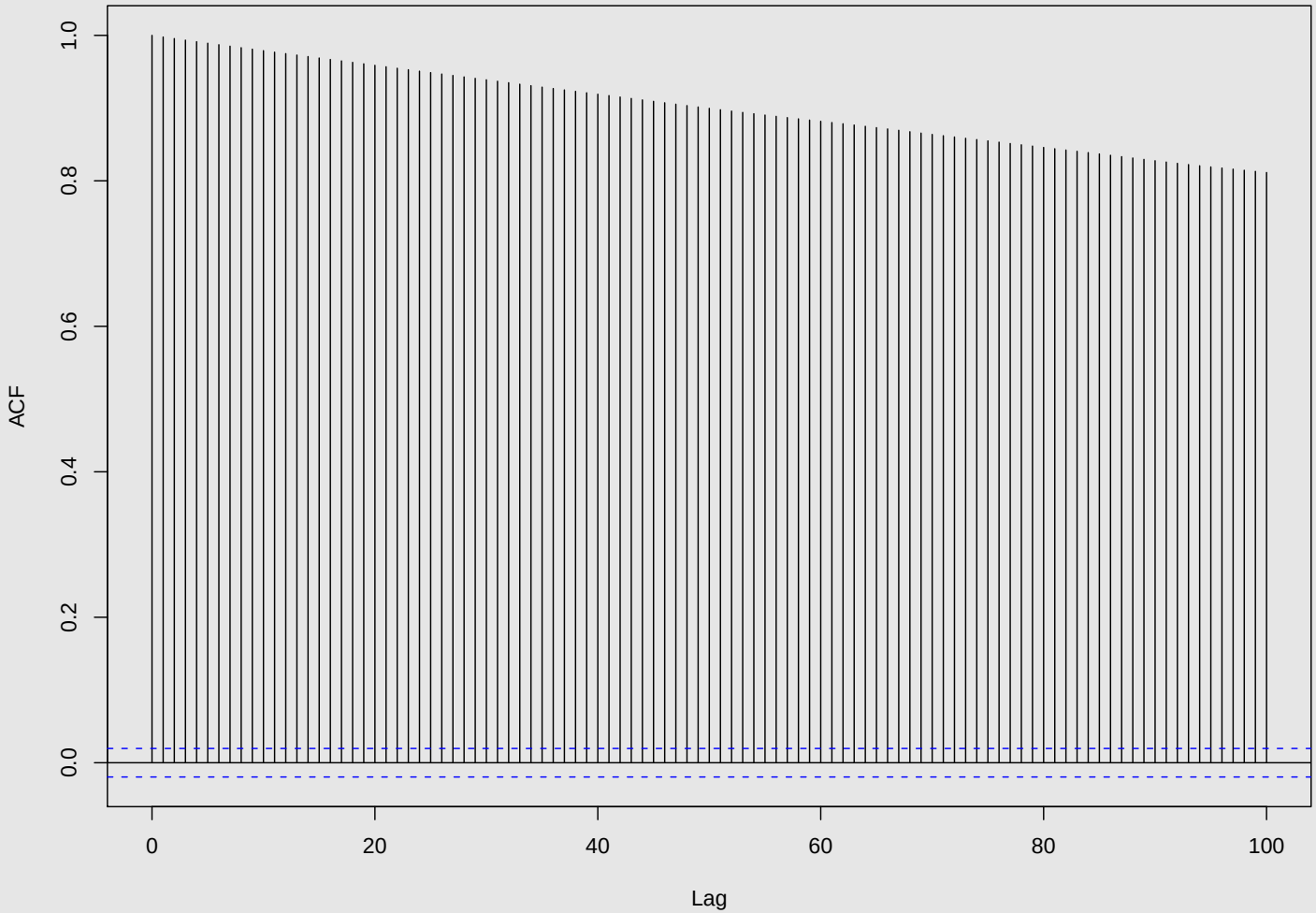
Autocorrelation function for $\sigma_q = 1$.



Metropolis-Hastings algorithm

Example (Gamma density)

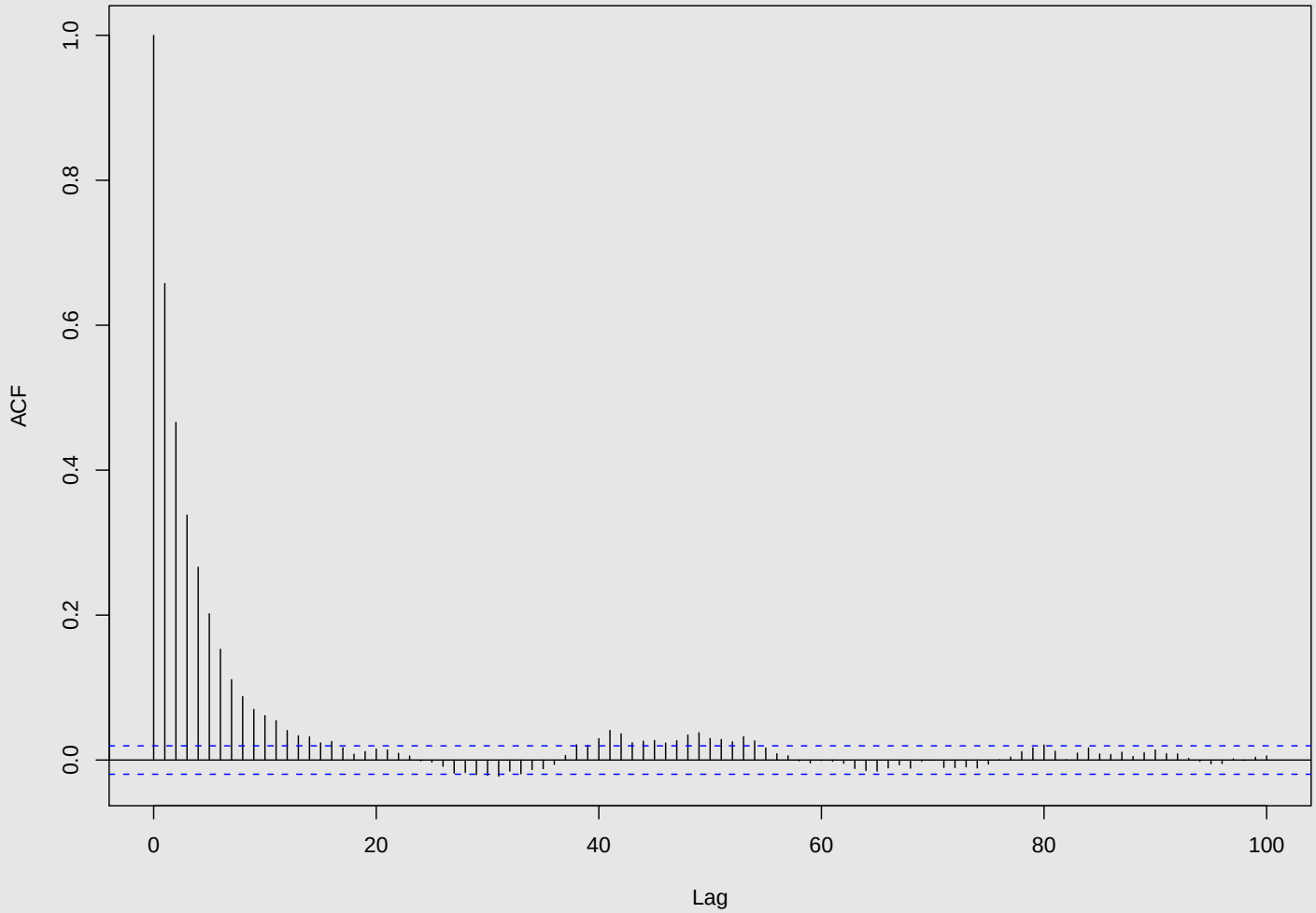
Autocorrelation function for $\sigma_q = 0.1$: inferior performance.



Metropolis-Hastings algorithm

Example (Gamma density)

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



Metropolis-Hastings algorithm

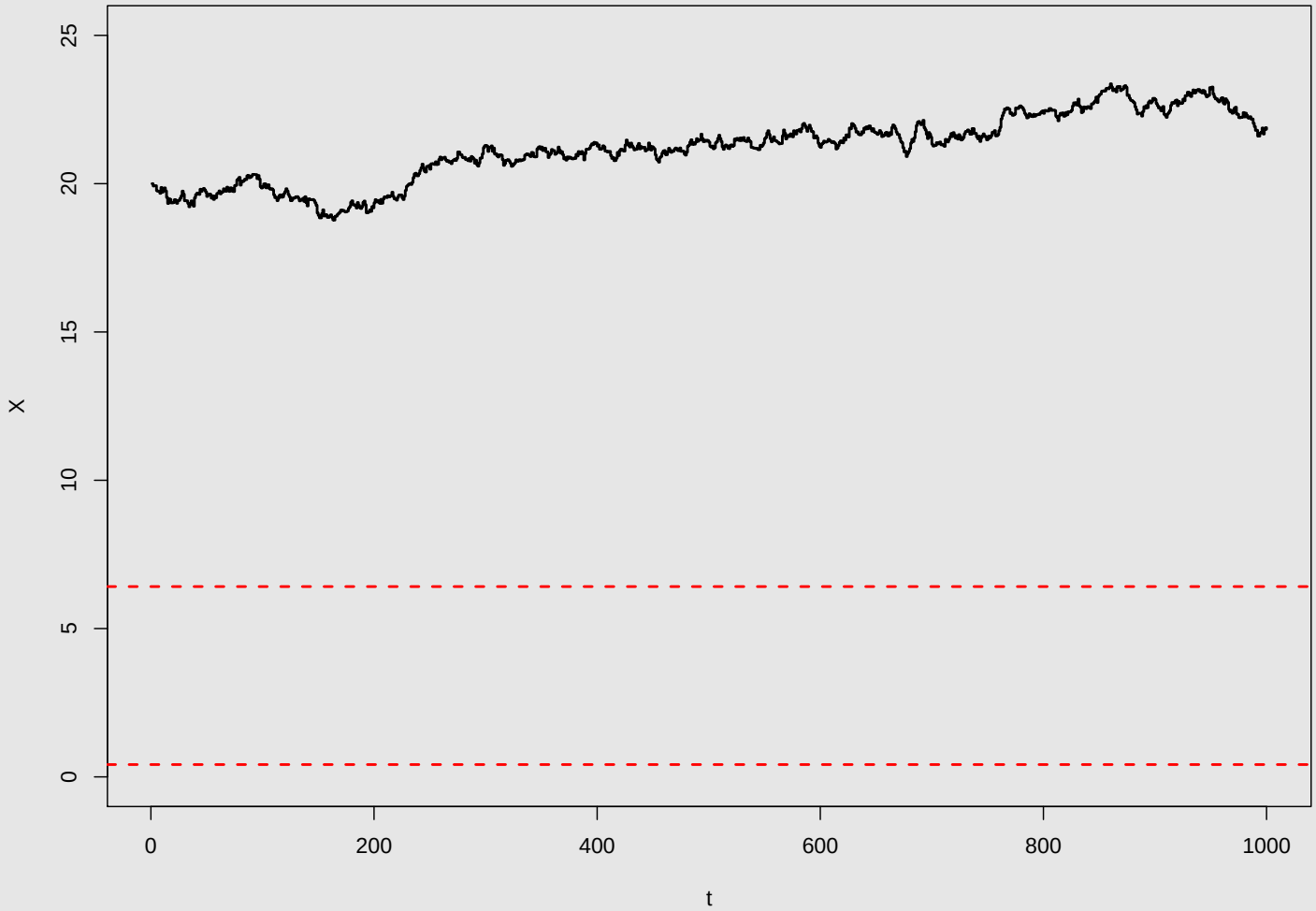
If the chain is started away from the high-probability region of π, then the it can take many steps to return there.

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)

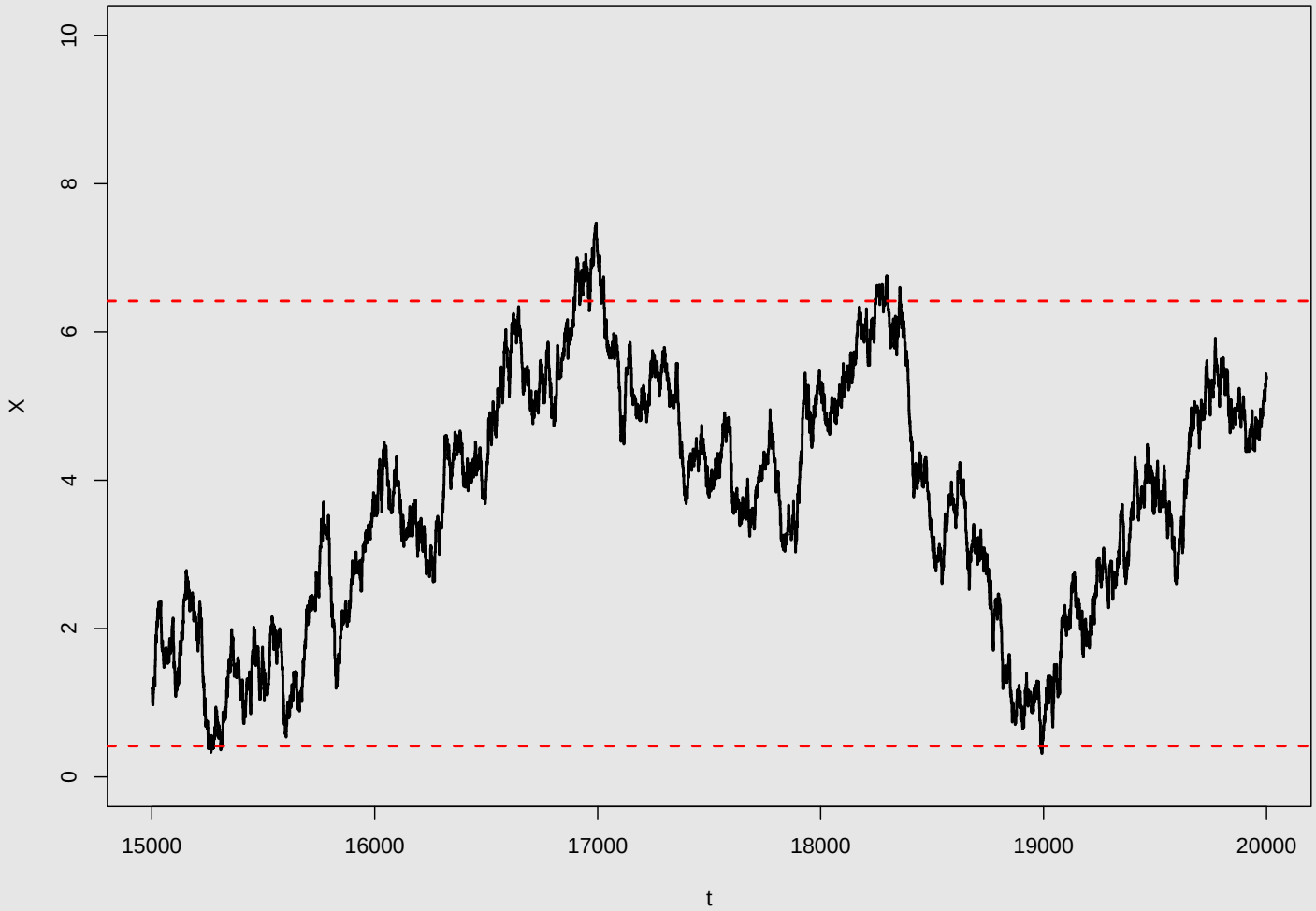
Starting value $X_0 = 20$, $\sigma_q = 0.1$: first 20000 steps.



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.

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 $(x_{10}, ..., x_{K0})$.
2. Sample in turn from the conditional distributions as follows:

   (a) sample x_{11} from $\pi_1 (x_1; x_{20}, x_{30}, ..., x_{K0})$
   (b) sample x_{21} from $\pi_2 (x_2; x_{11}, x_{30}, ..., x_{K0})$
   (c) sample x_{31} from $\pi_3 (x_3; x_{11}, x_{21}, ..., x_{K0})$

   $\vdots$

   (K) sample x_{K1} from $\pi_K (x_K; x_{11}, x_{21}, ..., x_{K-1\ 1})$

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

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 $\boldsymbol{X}$; note that these kernels in isolation yield *reducible* Markov chains.

A more general form of Metropolis-Hastings algorithm is based on a *mixture* transition kernel

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

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

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

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

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

then

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

Gibbs Sampler: example

Example (Bivariate Normal)

Therefore, here $d = 2$, and

$$\begin{aligned} X_1|X_2 = x_2 &\sim \mathcal{N}(\rho x_2, (1-\rho^2)) \\ X_2|X_1 = x_1 &\sim \mathcal{N}(\rho x_1, (1-\rho^2)) \end{aligned}$$

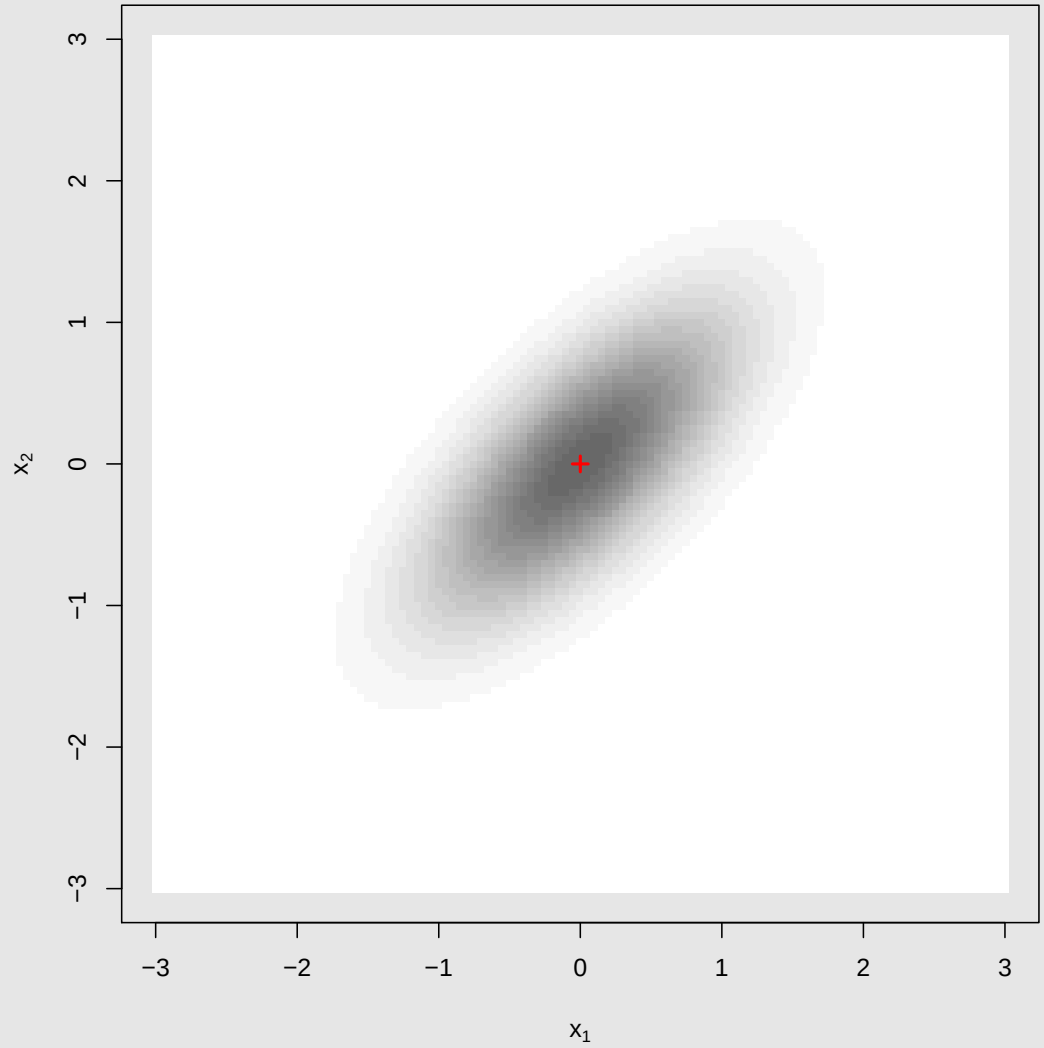
These distributions are sampled repeatedly with updating of the conditioning value after each sampling.

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

Gibbs Sampler: example

Example (Bivariate Normal)

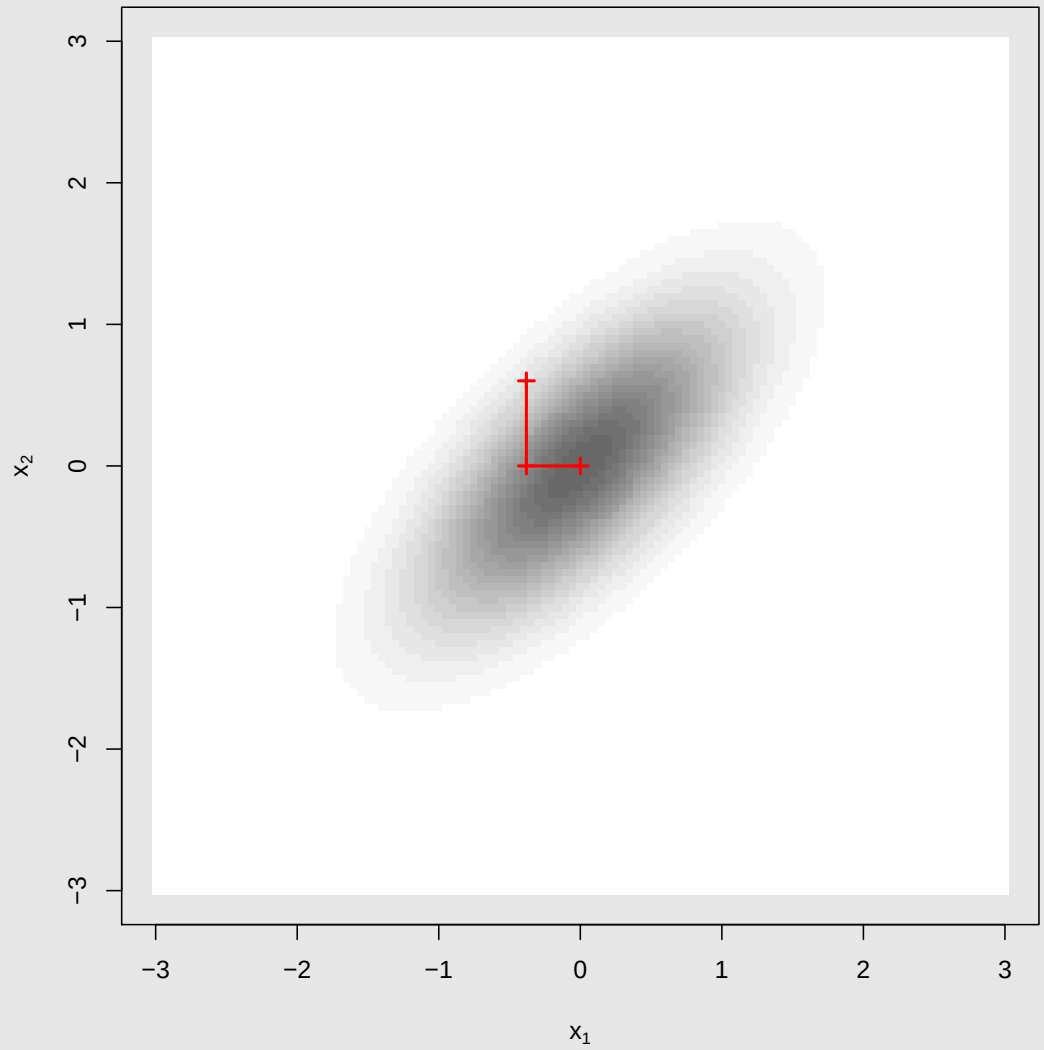
Initial point: (x_{10}, x_{20})



Gibbs Sampler: example

Example (Bivariate Normal)

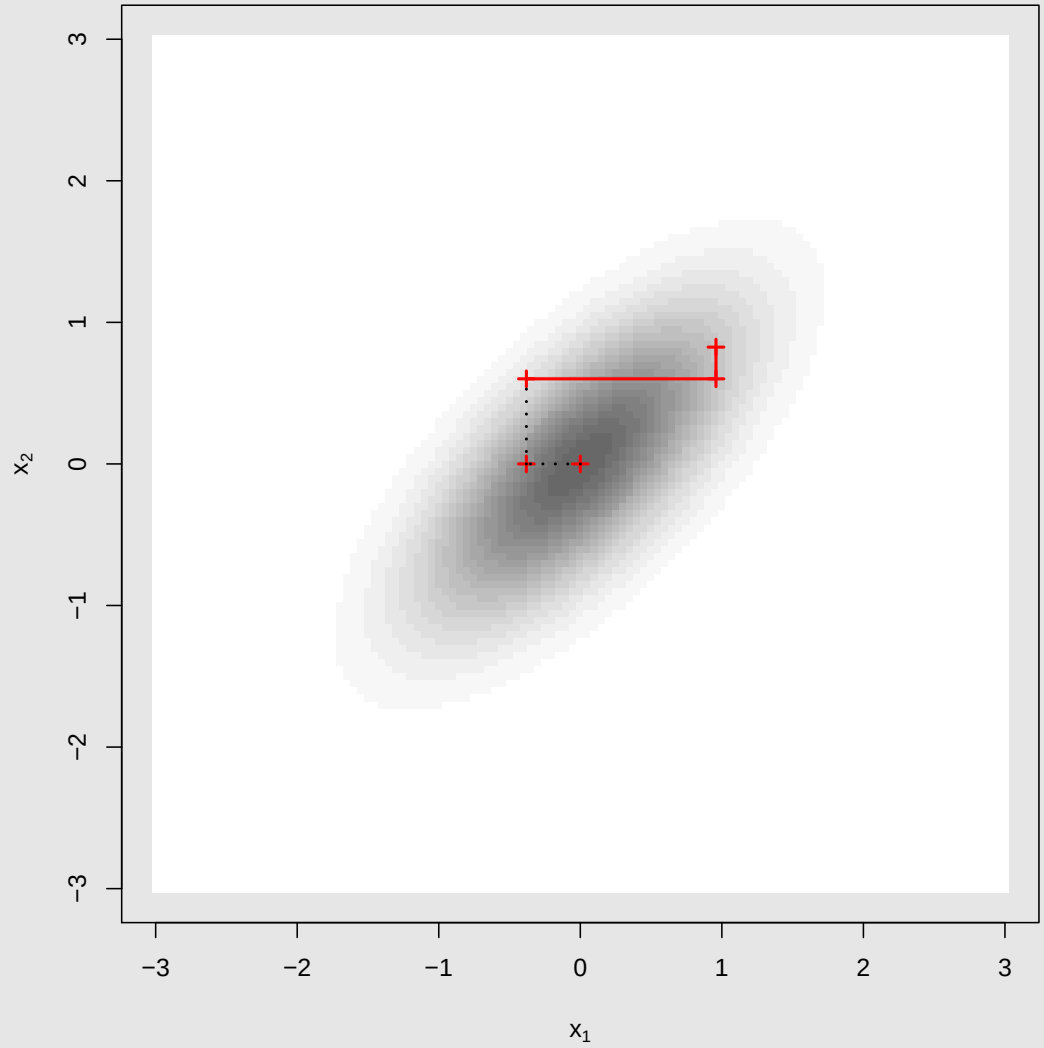
After one update: (x_{11}, x_{21})



Gibbs Sampler: example

Example (Bivariate Normal)

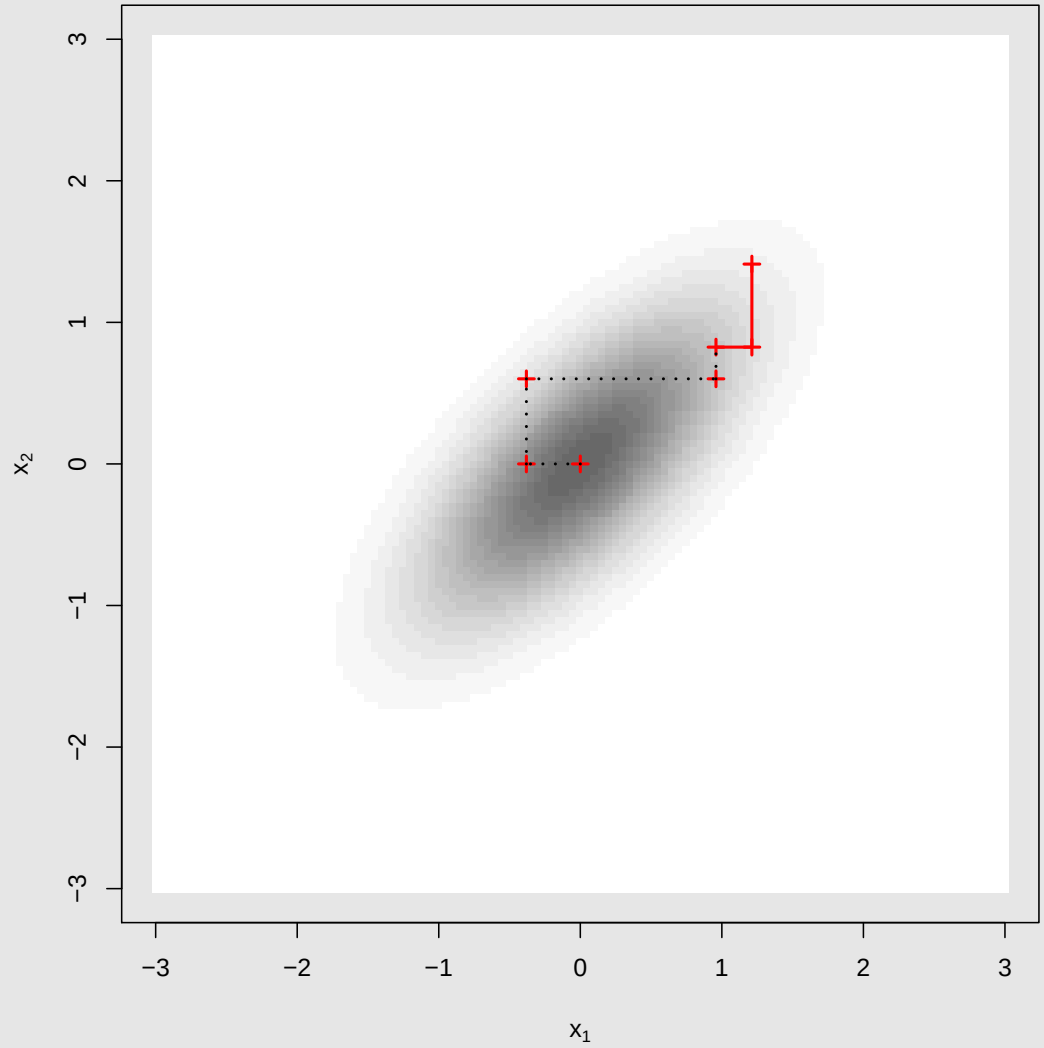
After two updates: (x_{12}, x_{22})



Gibbs Sampler: example

Example (Bivariate Normal)

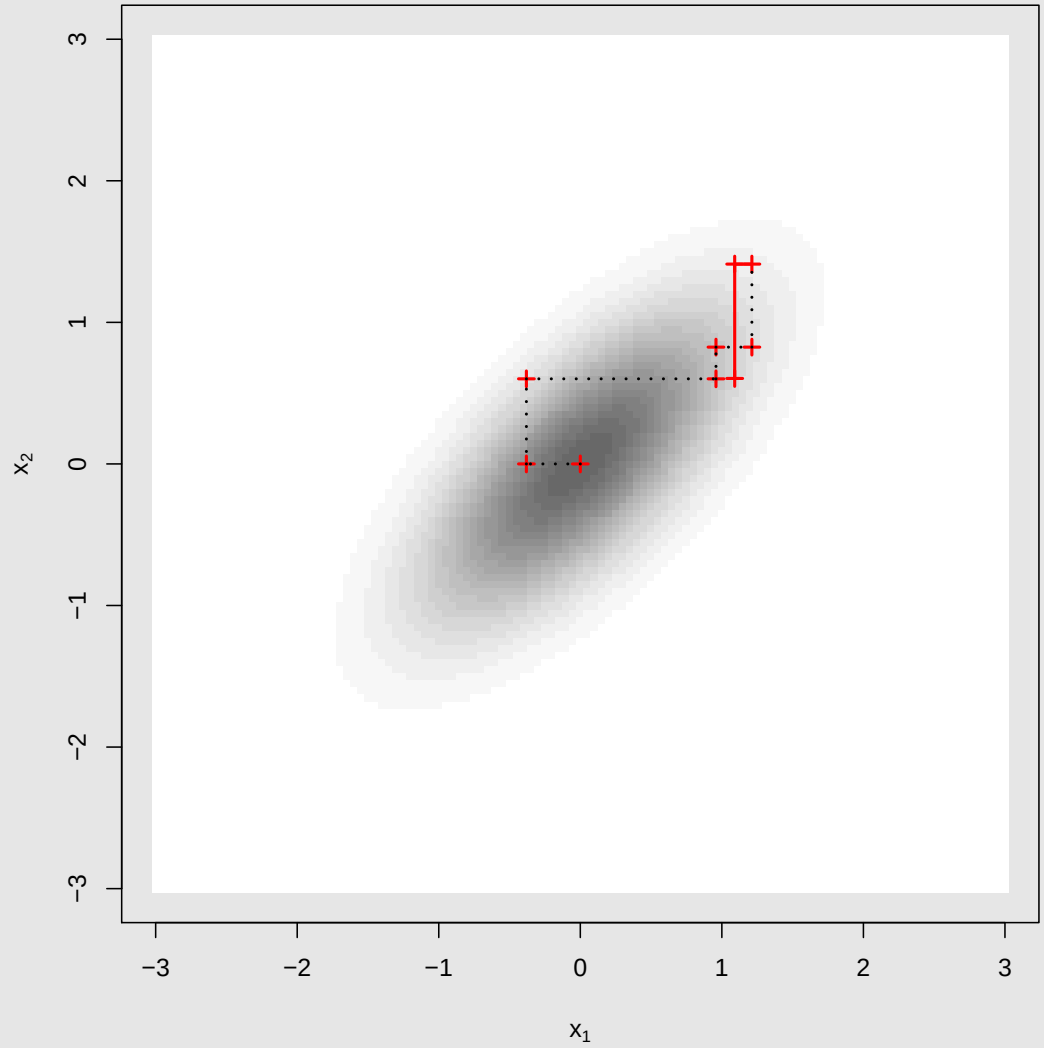
After three updates: (x_{13}, x_{23})



Gibbs Sampler: example

Example (Bivariate Normal)

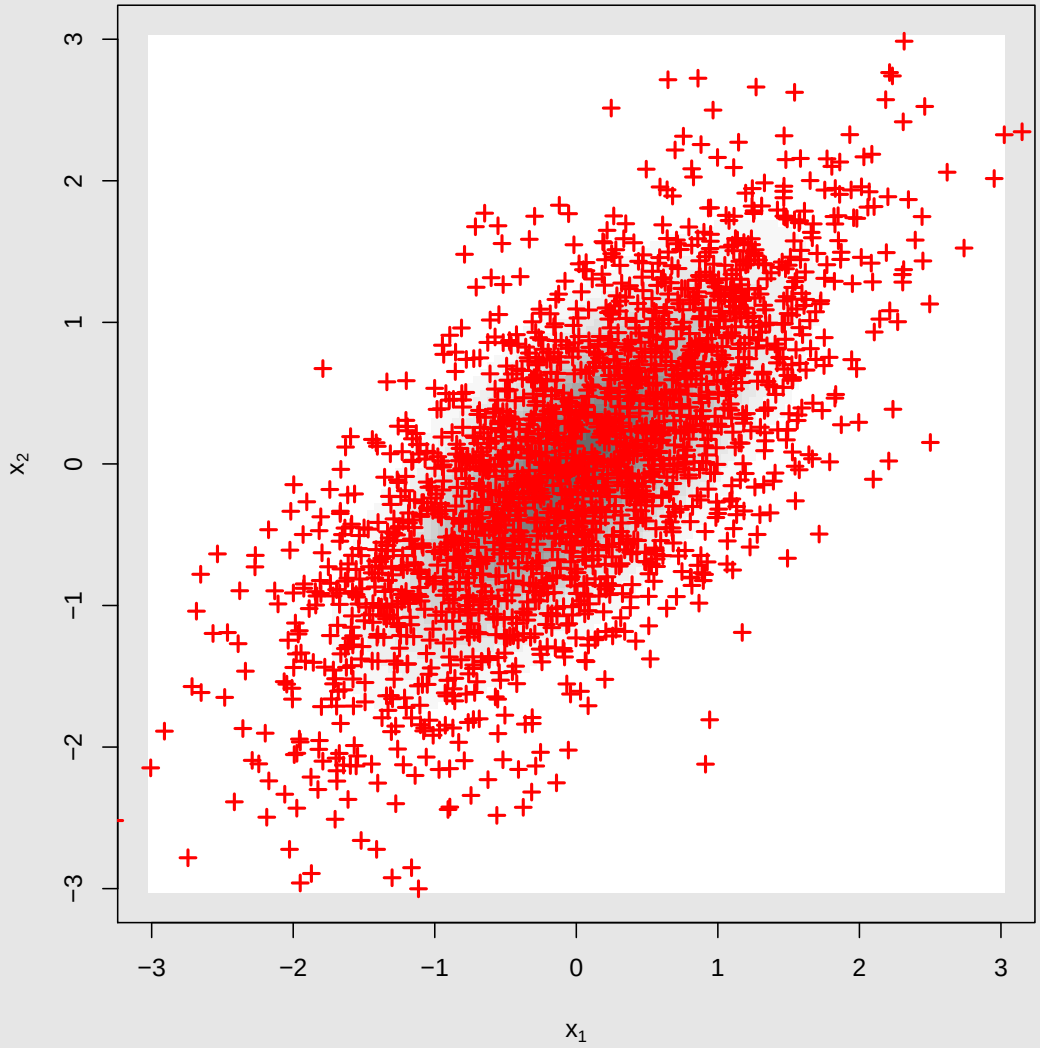
After four updates: (x_{14}, x_{24})



Gibbs Sampler: example

Example (Bivariate Normal)

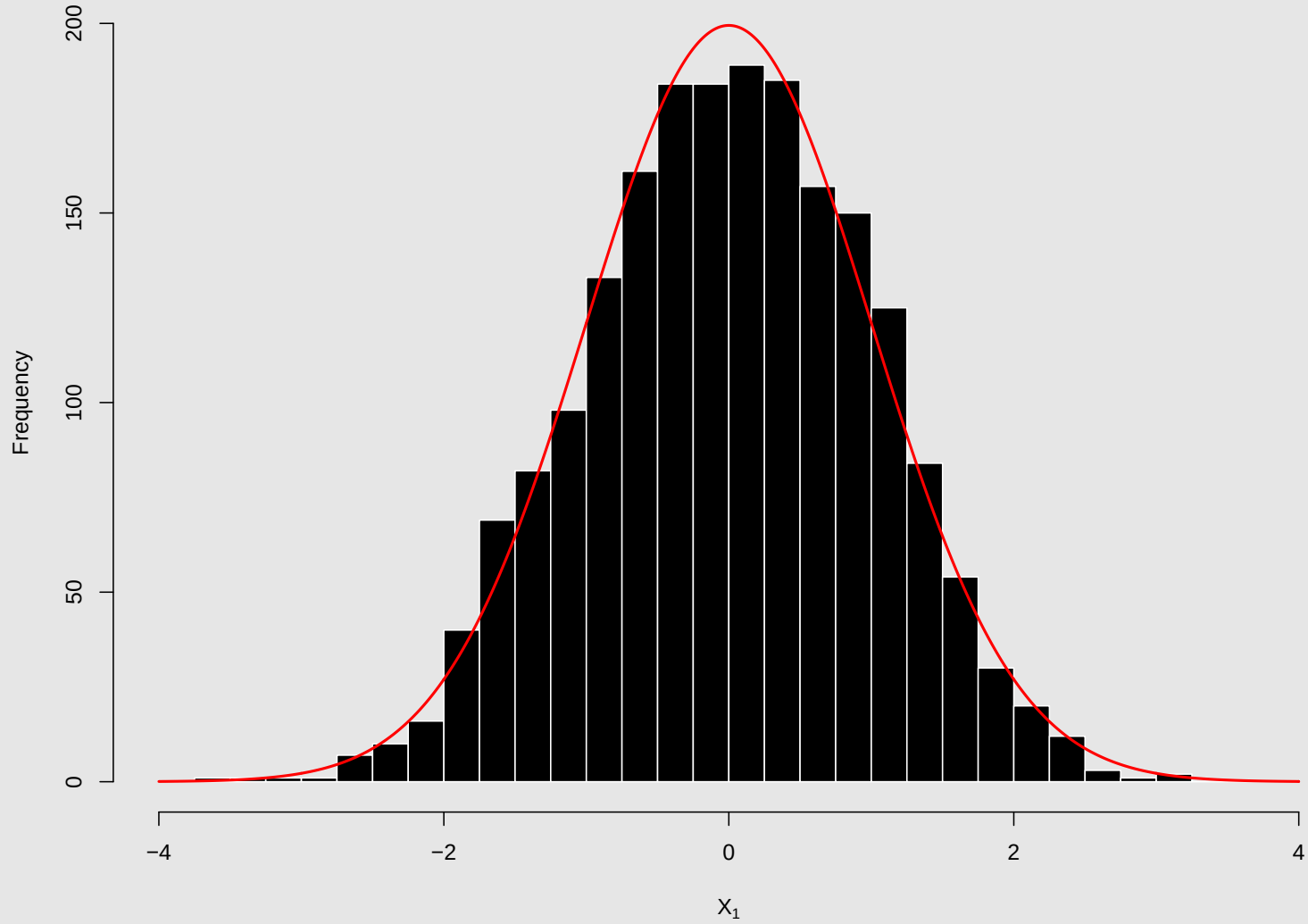
After 2000 updates: entire collected sample



Gibbs Sampler: example

Example (Bivariate Normal)

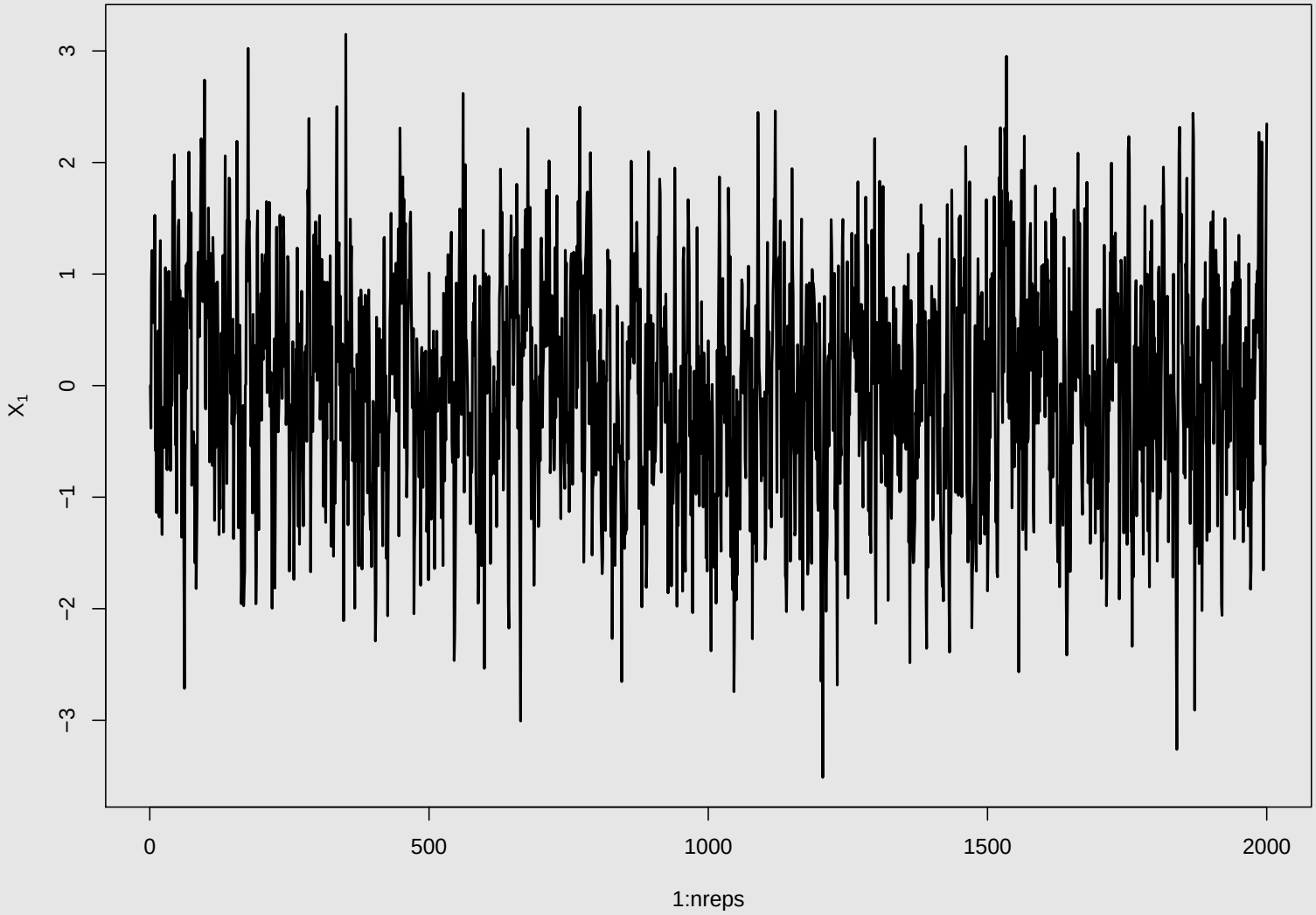
Histogram for X_1 with true marginal density (solid):



Gibbs Sampler: example

Example (Bivariate Normal)

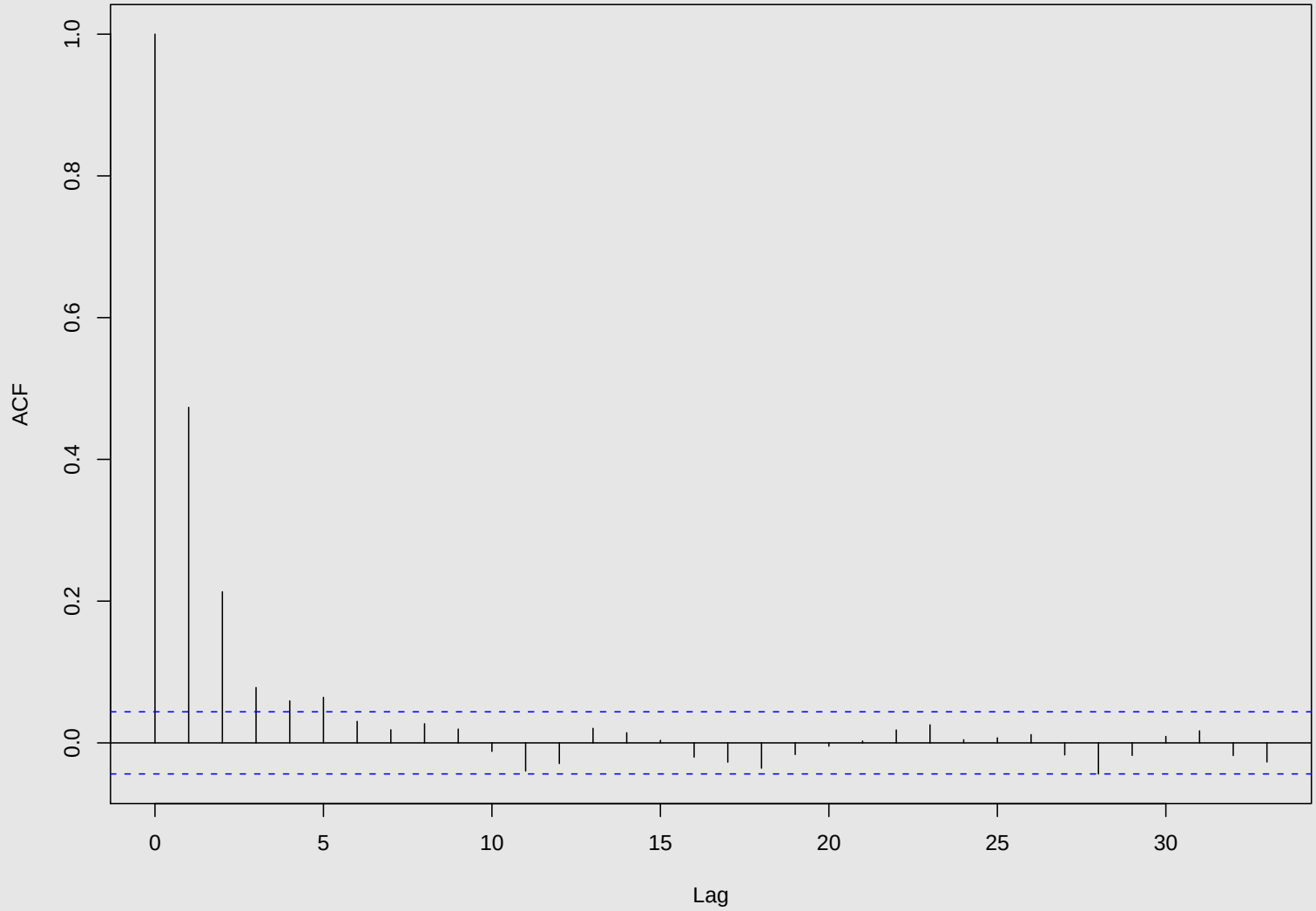
Trace for X_1:



Gibbs Sampler: example

Example (Bivariate Normal)

ACF for X_1:



Gibbs Sampler: example

Example (Bivariate Normal)

We note that

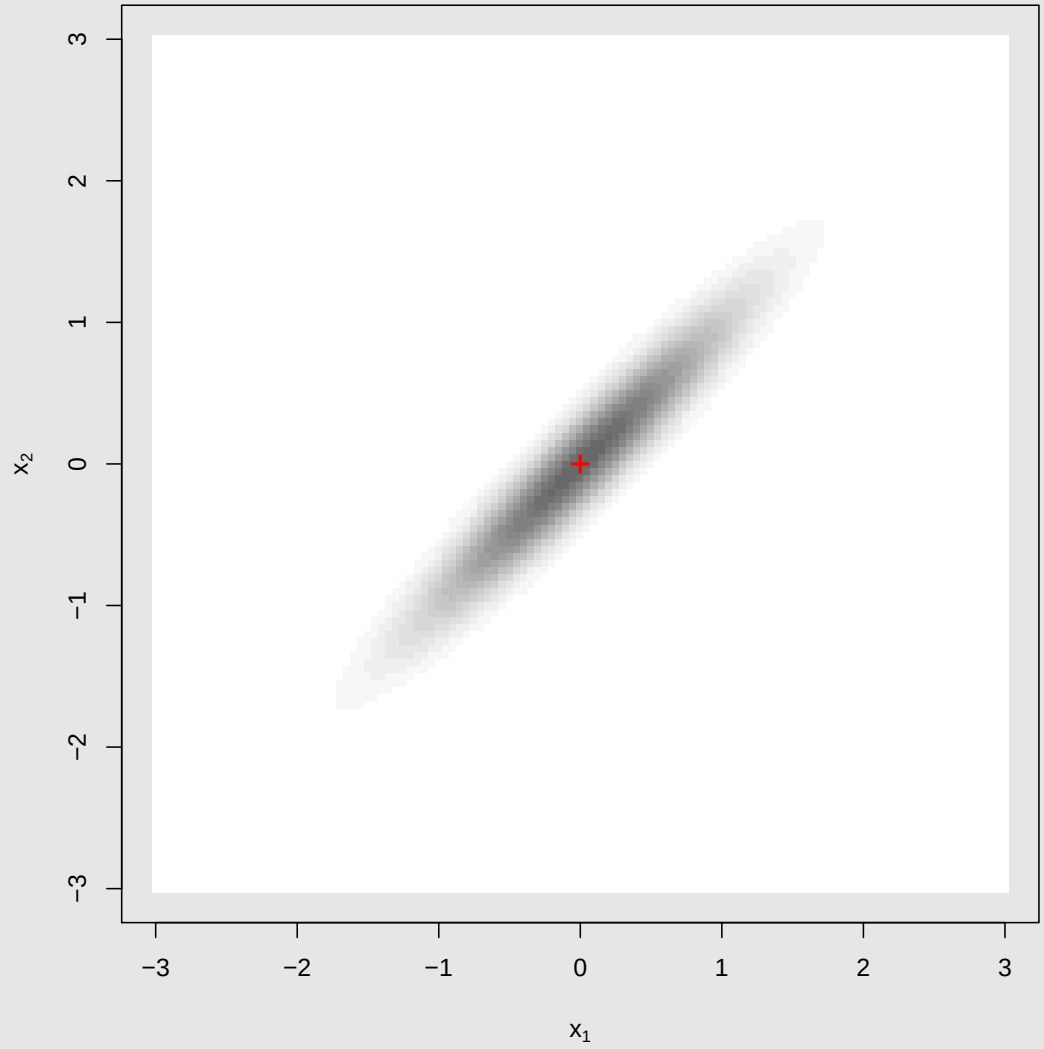
- the Gibbs sampler makes one-step moves along the coordinate axes
- the moves can traverse the support of the joint density fairly well
- there is no "tuning" of a proposal parameter (like σ_q)
- the samples of x_1 that are collected across steps are (dependent) samples from the correct marginal distribution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

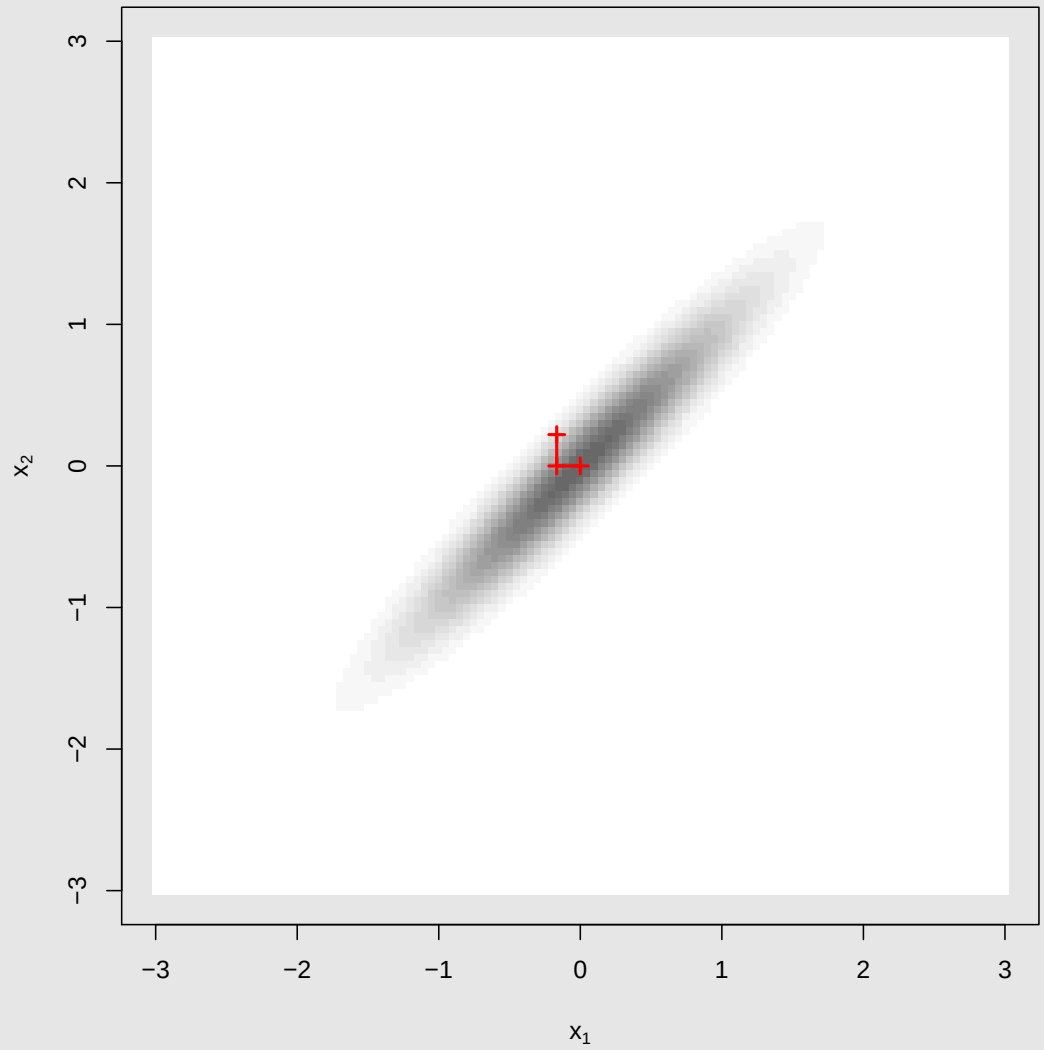
Initial point: (x_{10}, x_{20})



Gibbs Sampler: example

Example (Bivariate Normal)

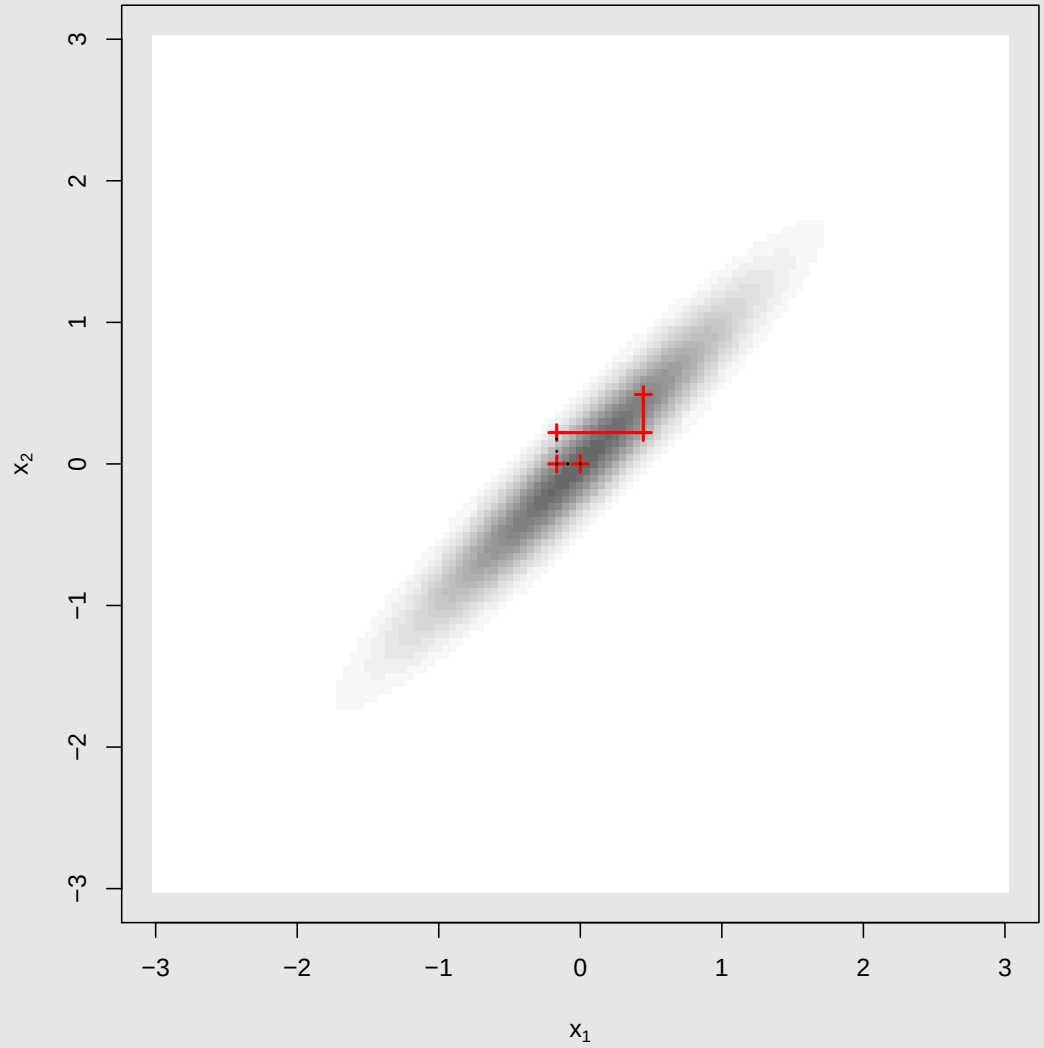
After one update: (x_{11}, x_{21})



Gibbs Sampler: example

Example (Bivariate Normal)

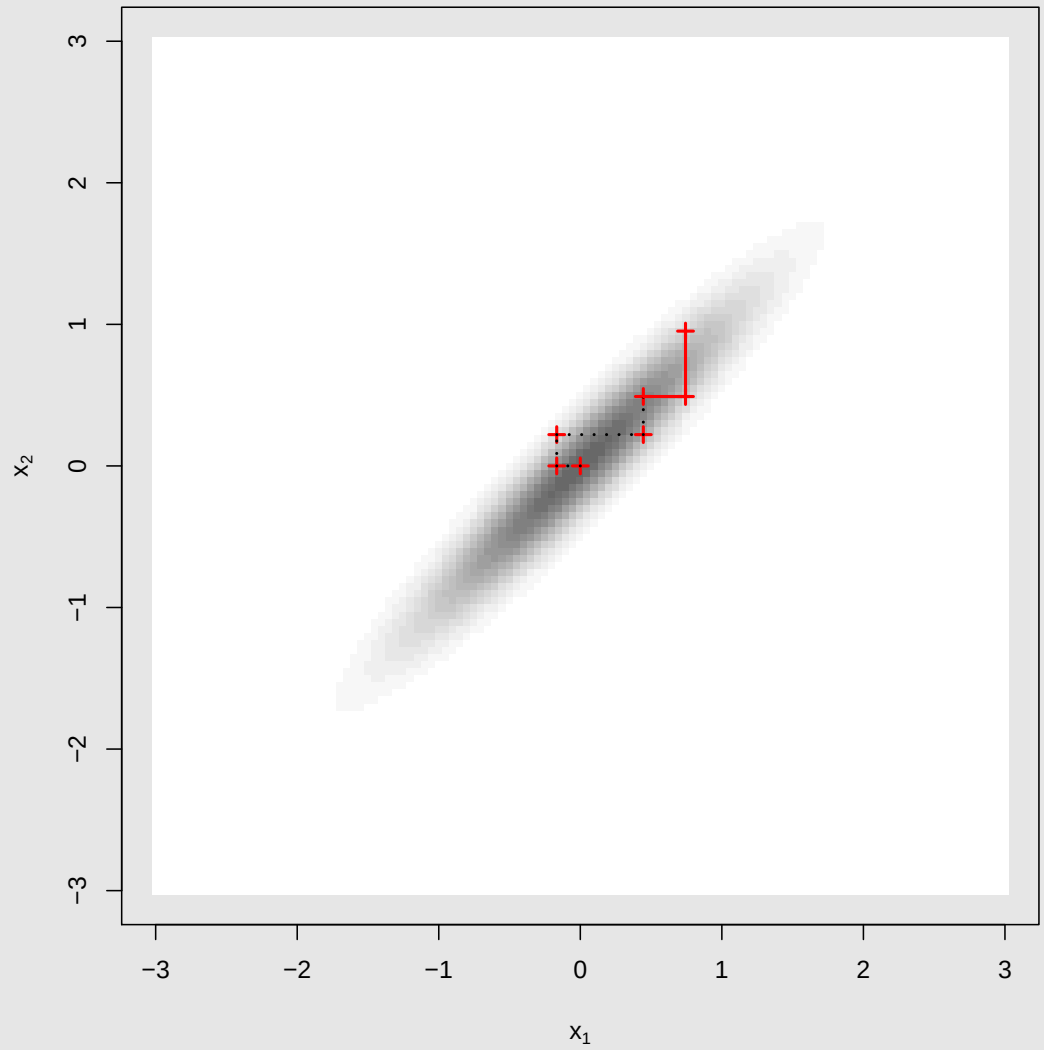
After two updates: (x_{12}, x_{22})



Gibbs Sampler: example

Example (Bivariate Normal)

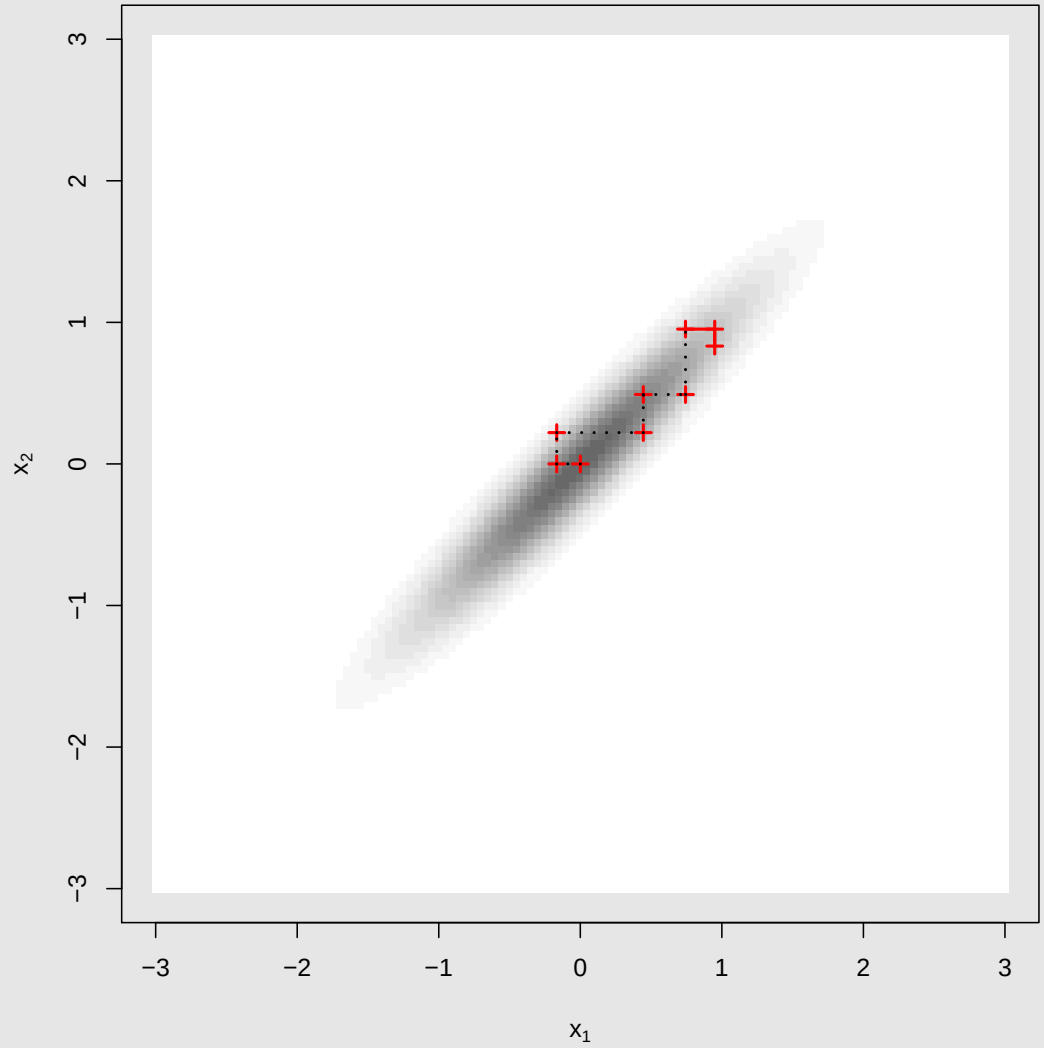
After three updates: (x_{13}, x_{23})



Gibbs Sampler: example

Example (Bivariate Normal)

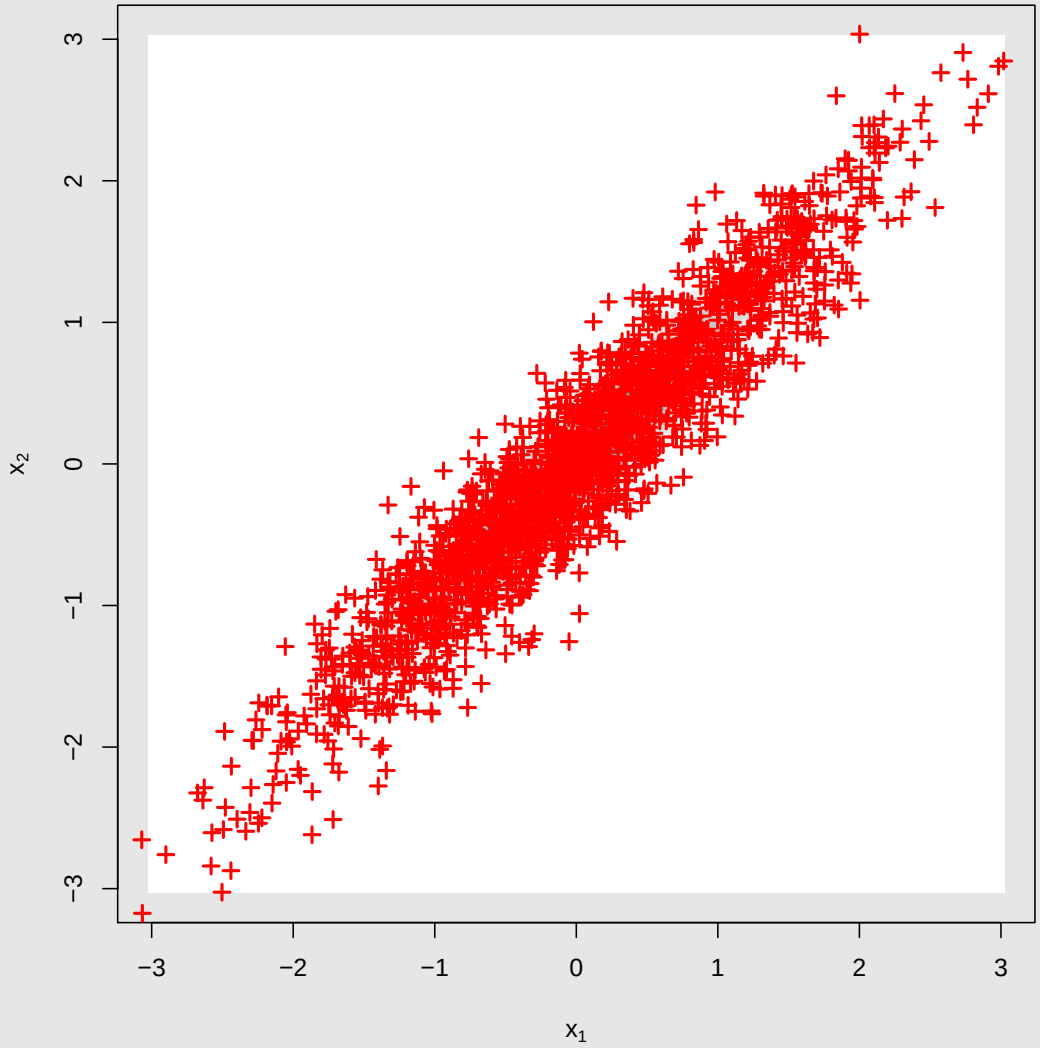
After four updates: (x_{14}, x_{24})



Gibbs Sampler: example

Example (Bivariate Normal)

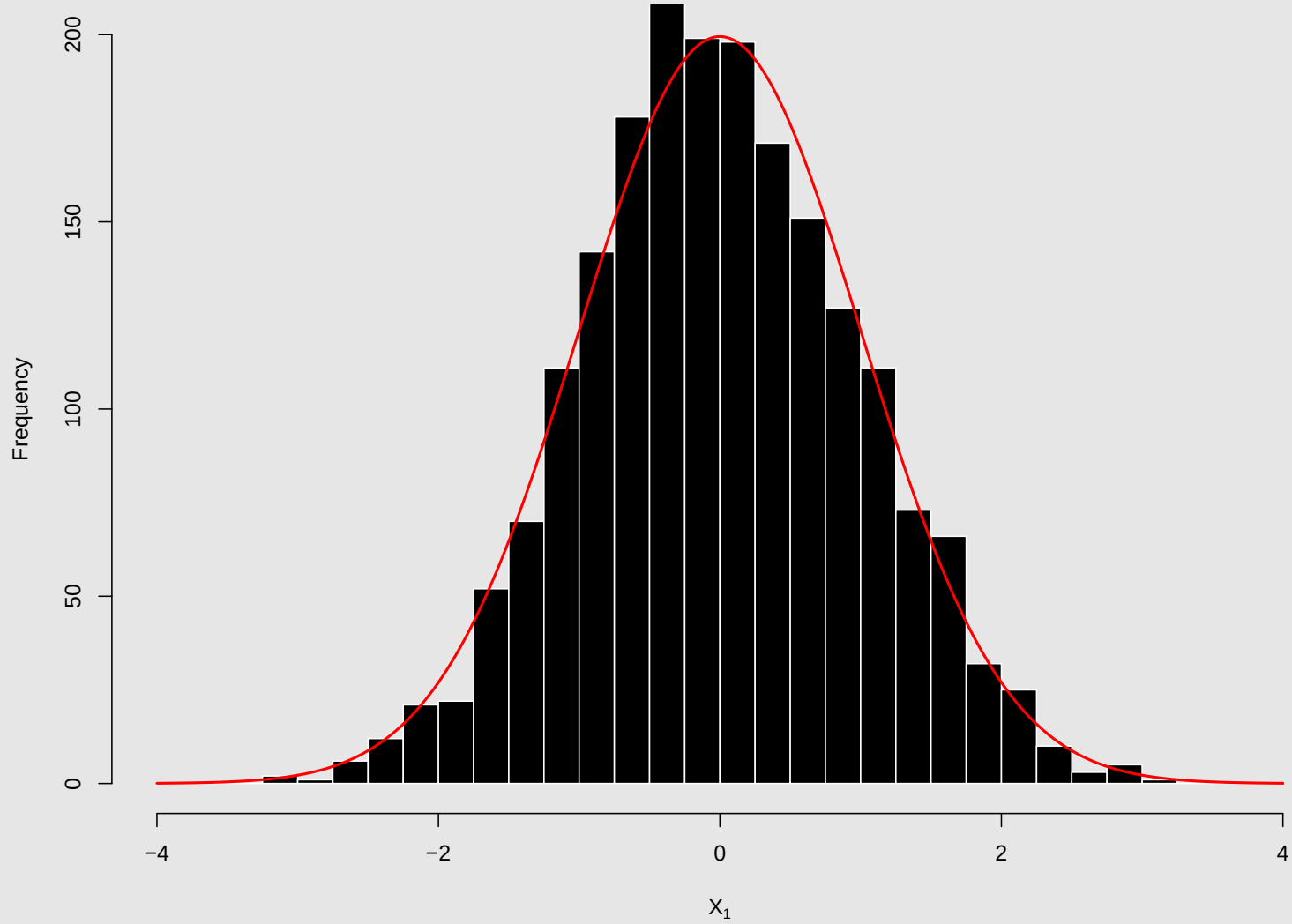
After 2000 updates: entire collected sample



Gibbs Sampler: example

Example (Bivariate Normal)

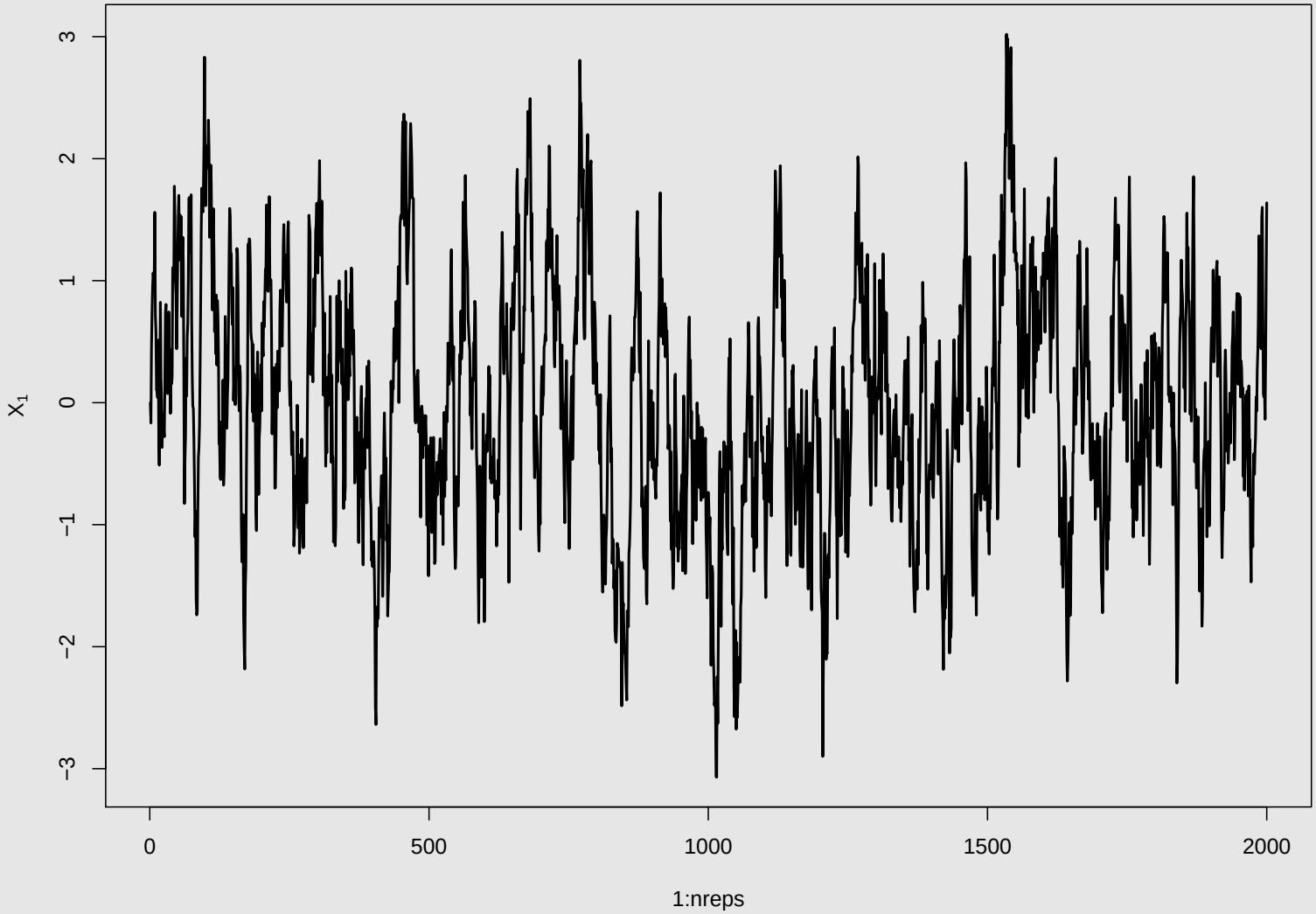
Histogram for X_1 with true marginal density (solid):



Gibbs Sampler: example

Example (Bivariate Normal)

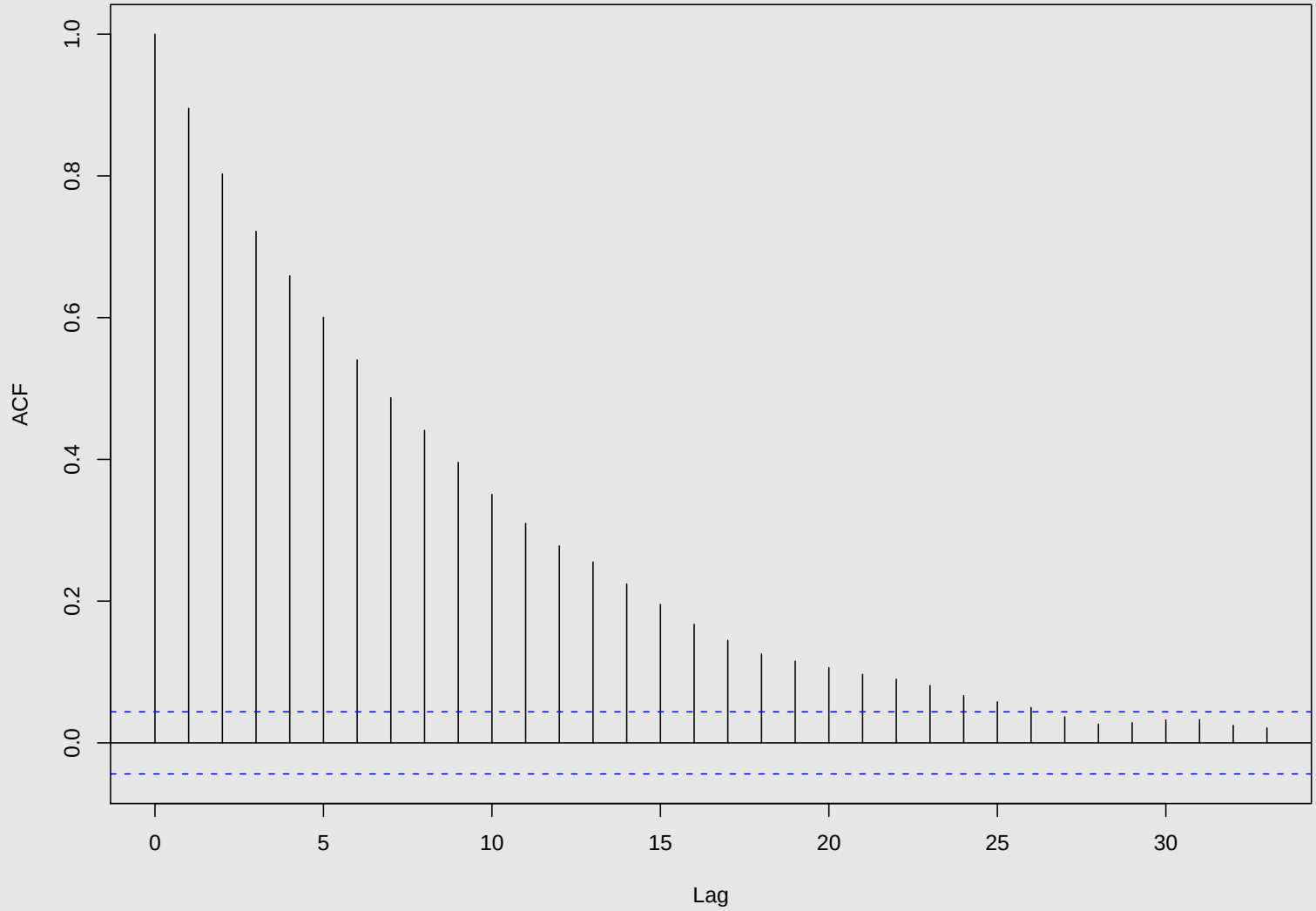
Trace for X_1:



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 i.i.d. from the Weibull distribution with density

$$f_Y(y; \gamma_0, \lambda_0) = \gamma_0 \lambda_0 y^{\gamma_0 - 1} \exp\{-\lambda_0 y^{\gamma_0}\} \qquad y > 0$$

and zero otherwise, for parameters $\gamma_0, \lambda_0 > 0$. 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

$$L(\gamma_0, \lambda_0; \boldsymbol{y}) = \gamma_0^n \lambda_0^n \left(\prod_{i=1}^n y_i \right)^{\gamma_0 - 1} \exp\left\{ -\lambda_0 \sum_{i=1}^n y_i^{\gamma_0} \right\}$$

Gibbs Sampler: example

Example (Weibull posterior distribution)

We assume independent $Exponential(0.1)$ priors for γ_0 and λ_0

$$p(\gamma_0, \lambda_0) = 0.01e^{-0.1(\gamma_0+\lambda_0)} \qquad \gamma_0, \lambda_0 > 0.$$

This yields the posterior distribution up to proportionality as

$$\pi(\gamma_0, \lambda_0|\boldsymbol{y}) \propto L(\gamma_0, \lambda_0; \boldsymbol{y}) p(\gamma_0, \lambda_0) \qquad \gamma_0, \lambda_0 > 0$$

which is a non-standard distribution.

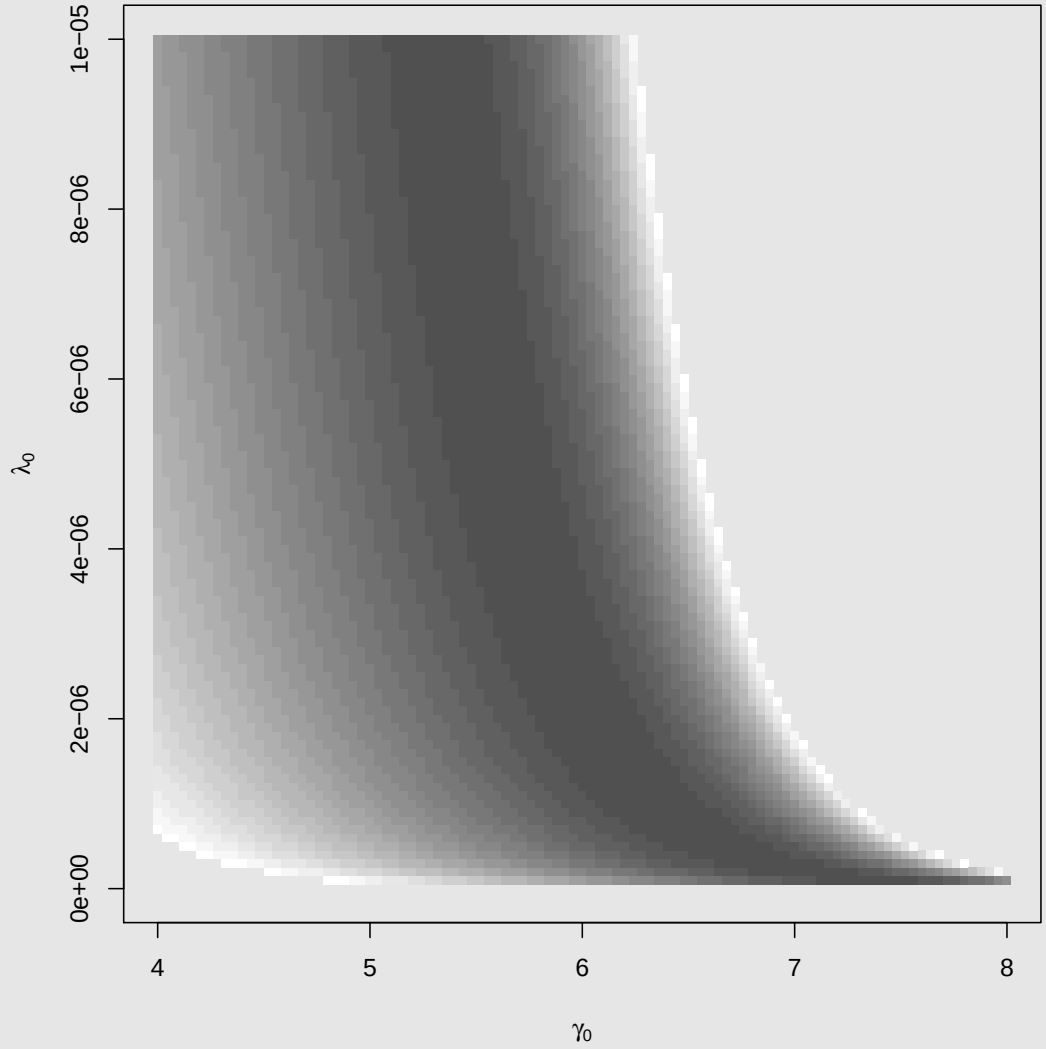
We seek to produce a sample from this joint posterior distribution for data $(n = 15)$.

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

$$\pi(\gamma_0|\lambda_0, \boldsymbol{y}) \qquad \pi(\lambda_0|\gamma_0, \boldsymbol{y})$$

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

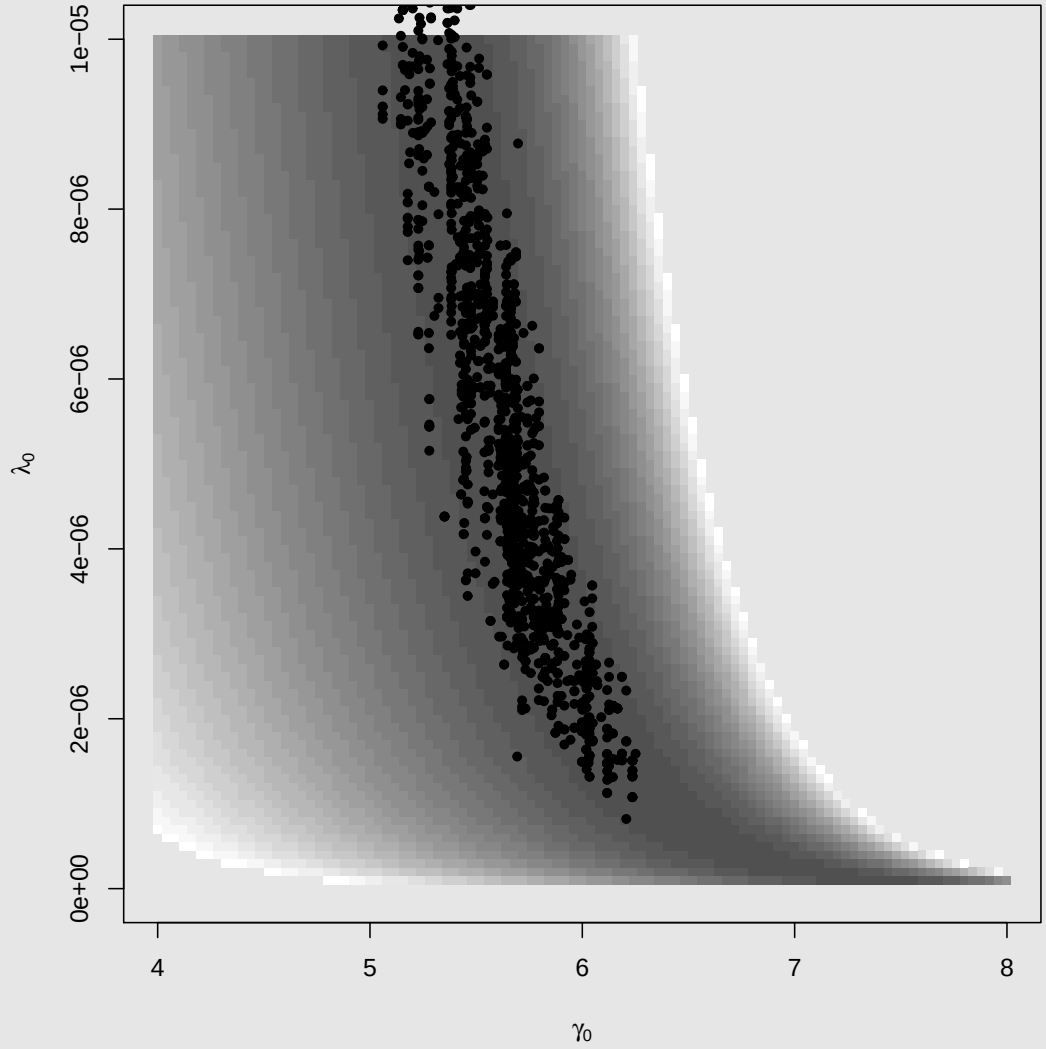
We adopt a *Metropolis-within-Gibbs* strategy, using Metropolis-Hastings accept-reject steps for each parameter and its full conditional.

Specifically, as both parameters are positive, we use the reflected normal proposal distribution from the previous example with $\sigma_q = 1$ for γ_0 and $\sigma_q = 10^{-3}$ for λ_0 proposals.

Gibbs Sampler: example

Example (Weibull example)

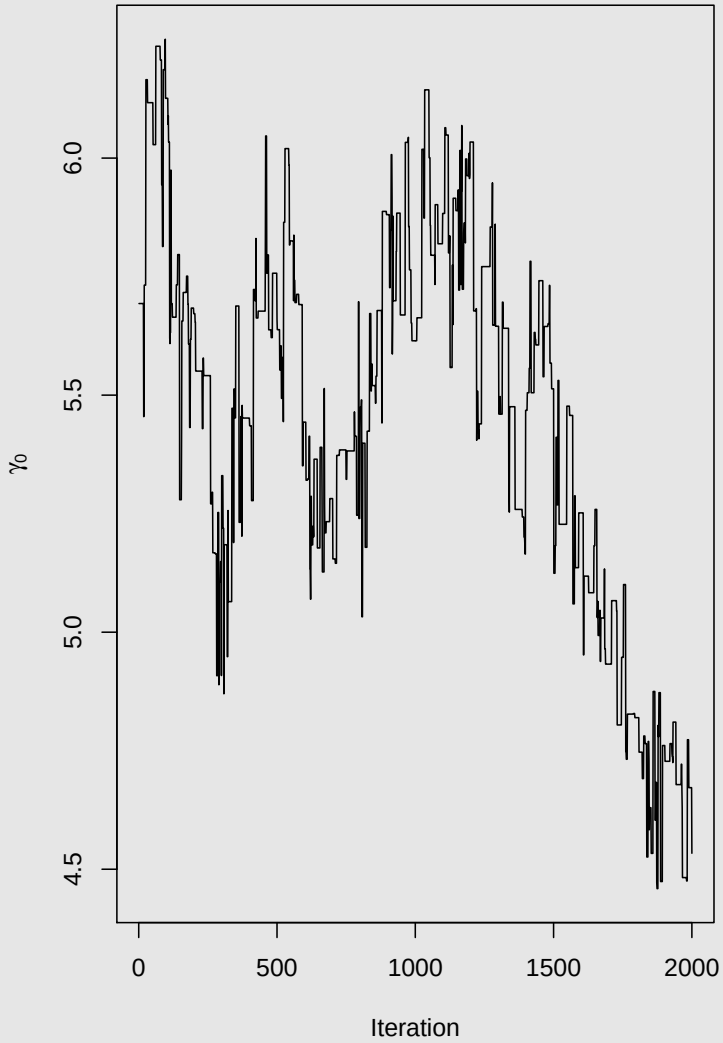
Sample from joint posterior from first 2000 Gibbs sampler steps.

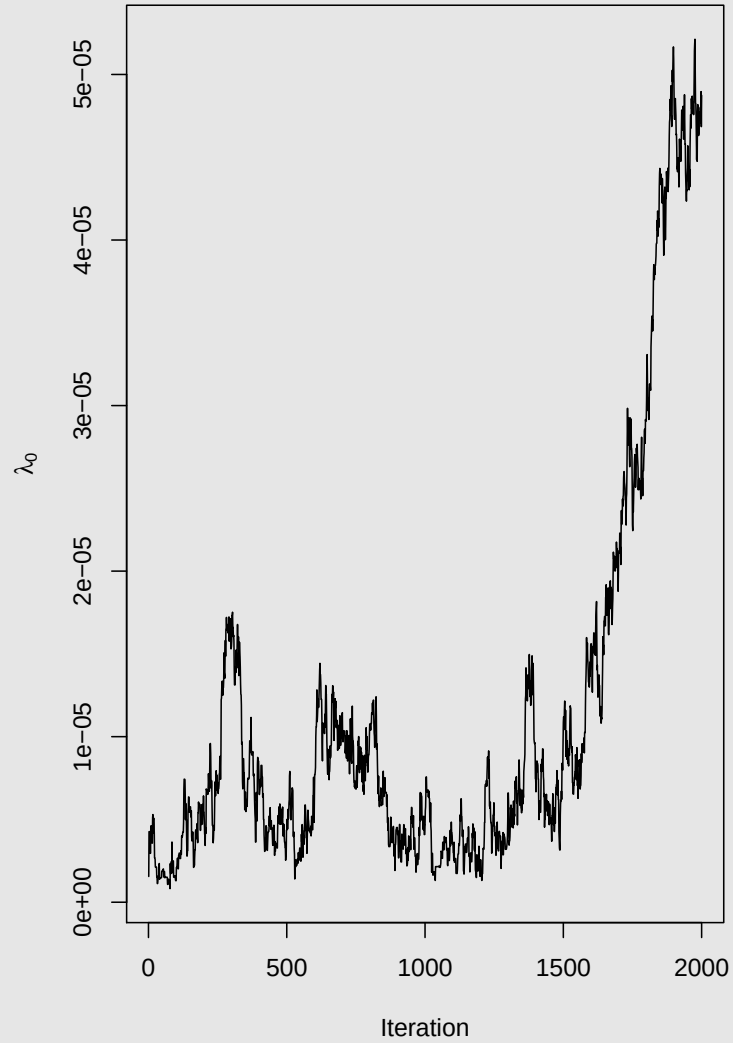


Gibbs Sampler: example

Example (Weibull example)

Trace plots:

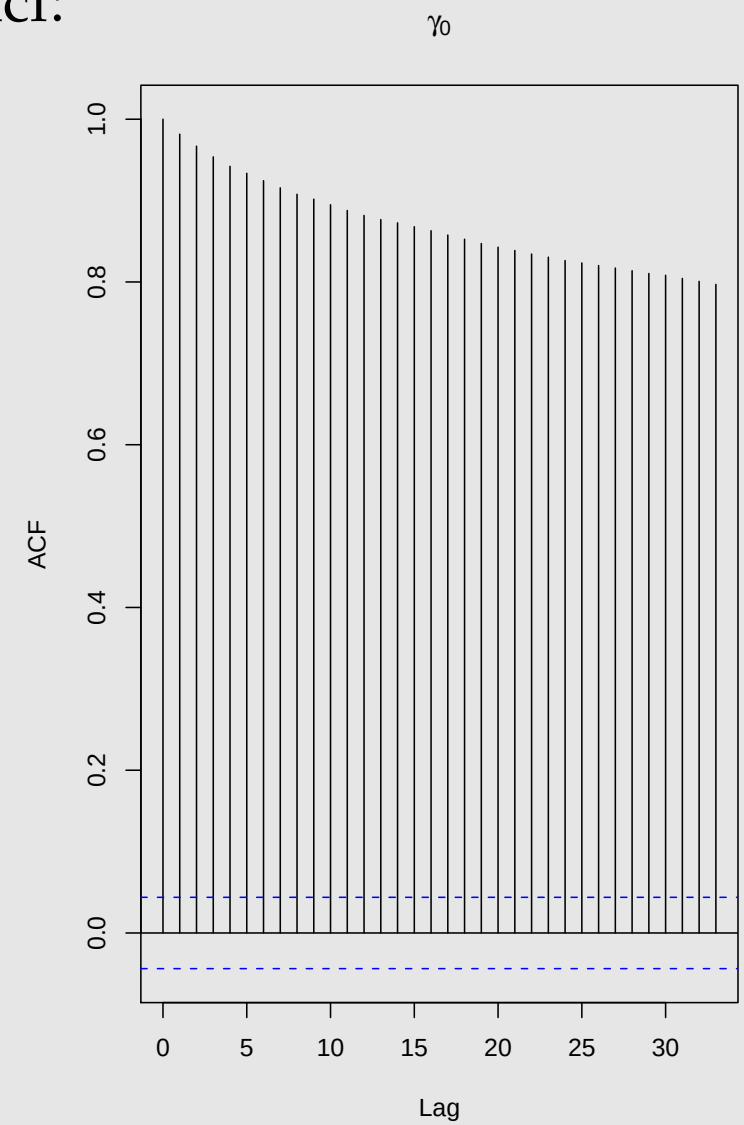


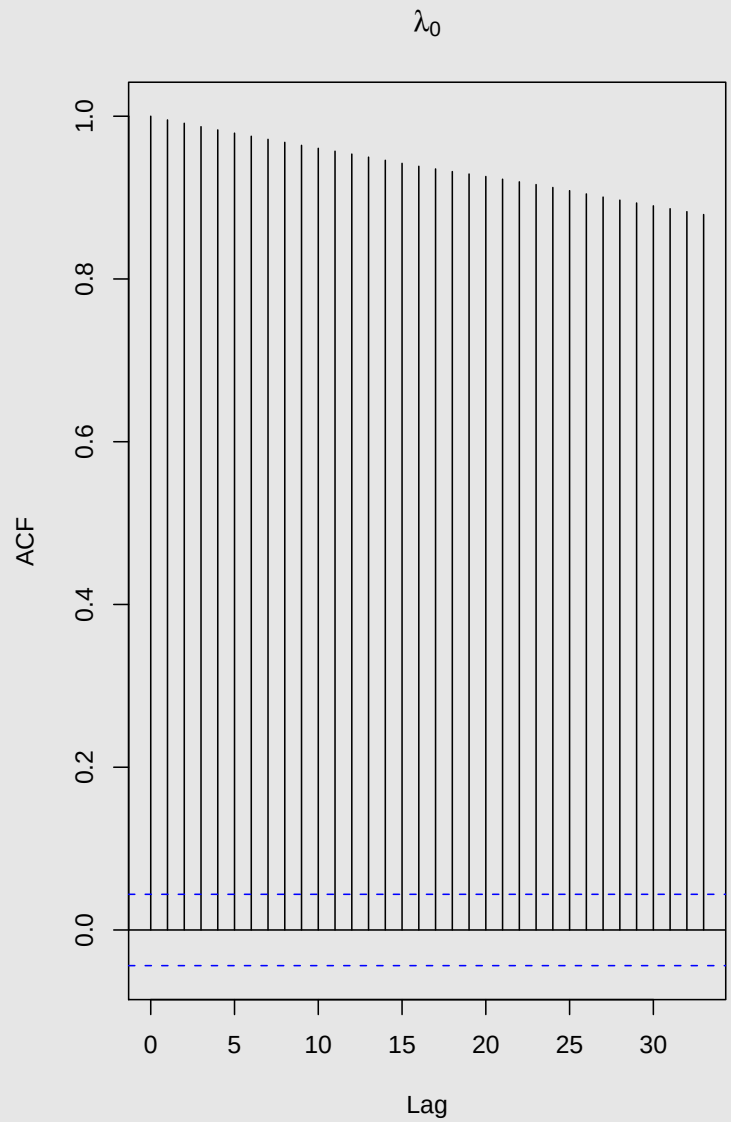


Gibbs Sampler: example

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.

An alternative parameterization partially solves this problem. Let $\gamma = \gamma_0$, but

$$\lambda = \left(\frac{1}{\lambda_0}\right)^{1/\gamma} \qquad \lambda_0 = \left(\frac{1}{\lambda}\right)^{\gamma}$$

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

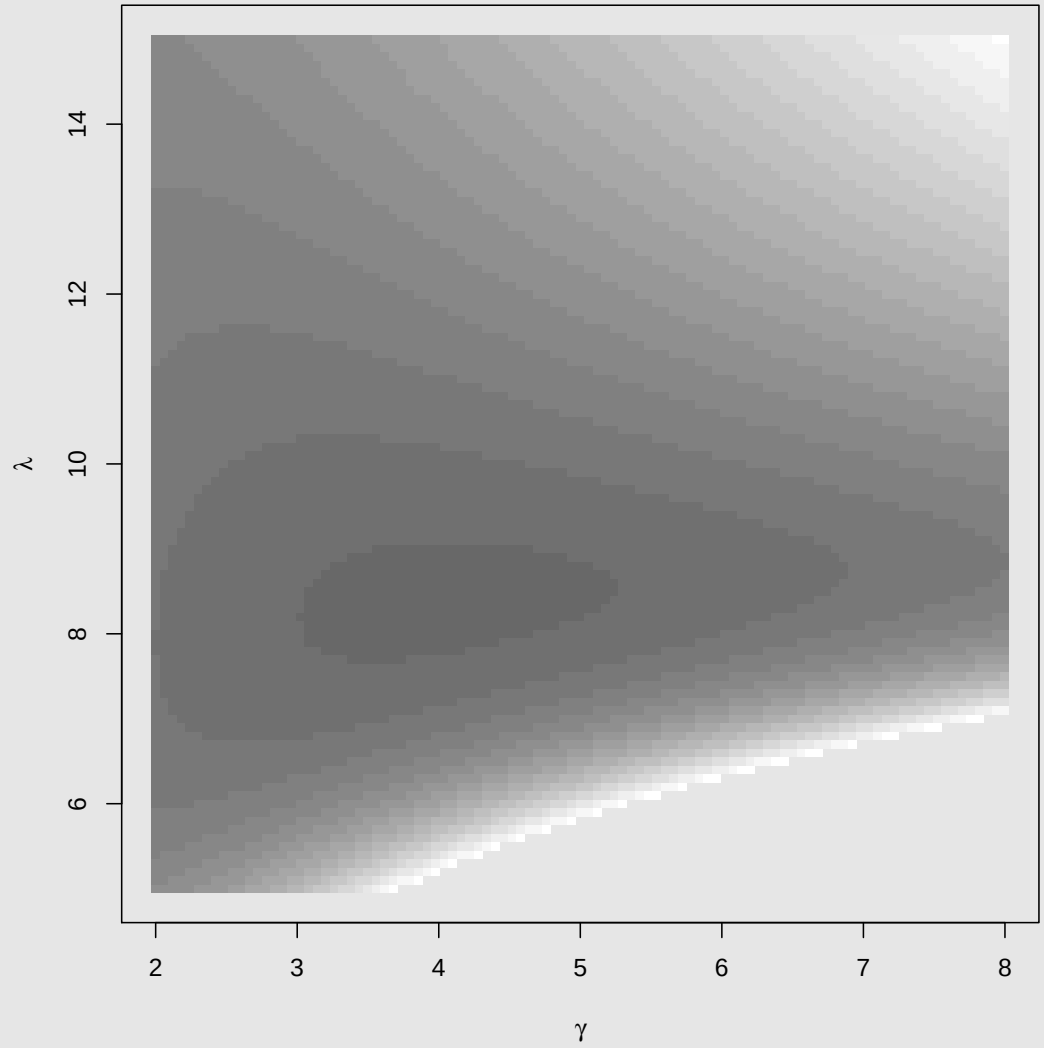
$$p(\gamma, \lambda) = p(\gamma, \lambda_0(\gamma, \lambda))|J(\alpha, \gamma)|$$

We again must use Metropolis-within-Gibbs.

Gibbs Sampler: example

Example (Weibull example)

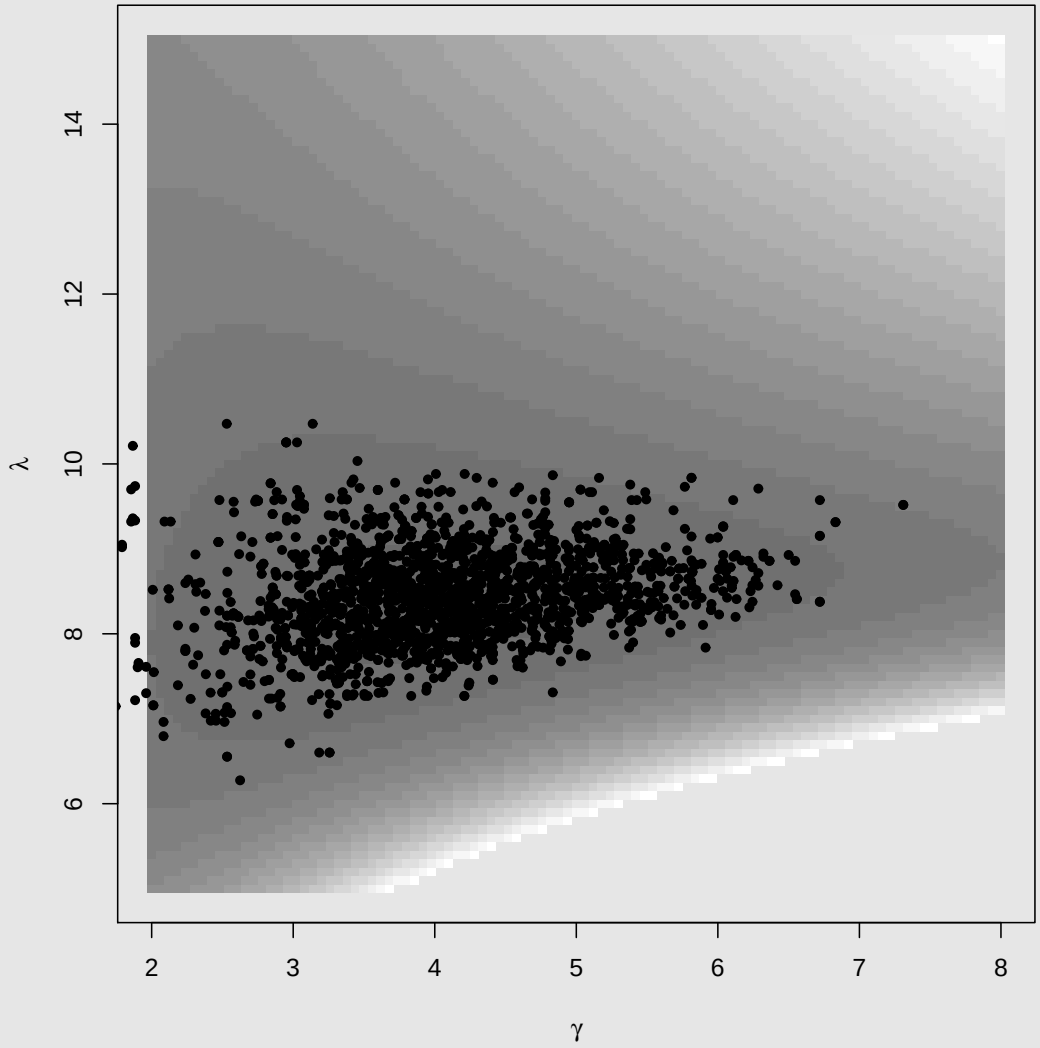
Joint posterior (up to proportionality) for new parameterization



Gibbs Sampler: example

Example (Weibull example)

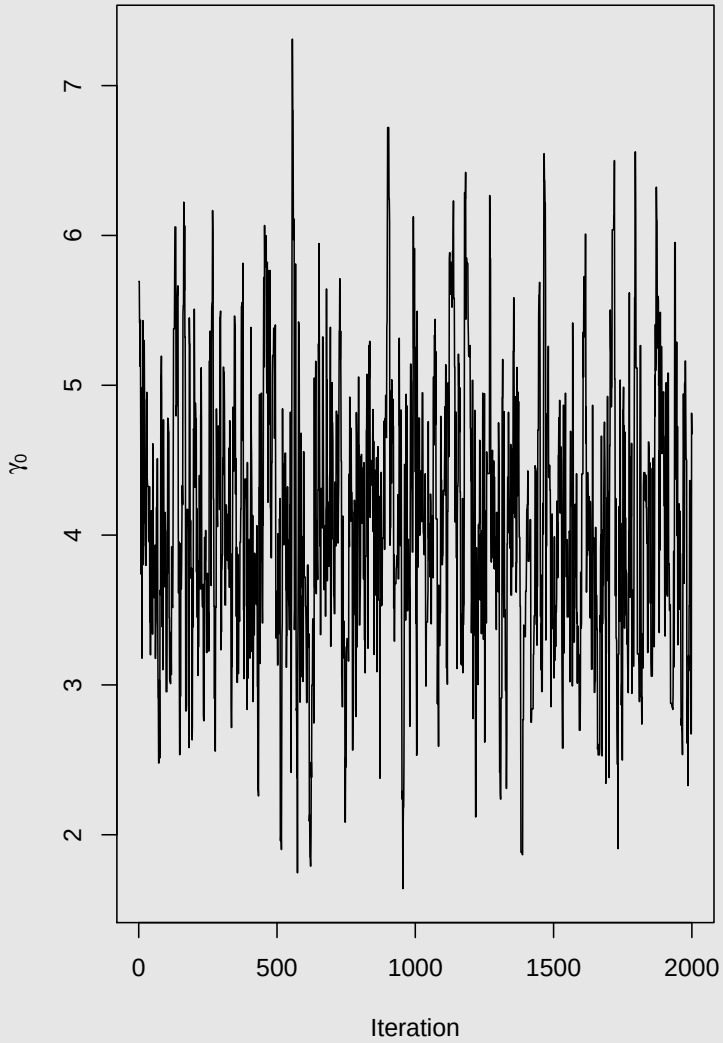
Sample from joint posterior for new parameterization

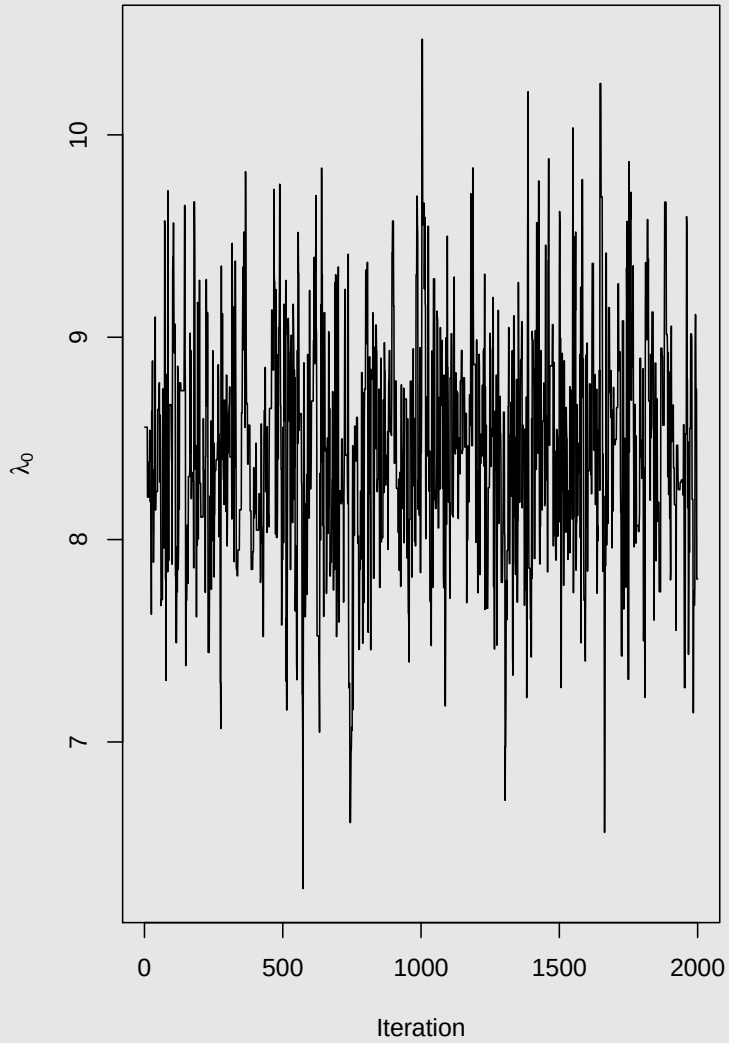


Gibbs Sampler: example

Example (Weibull example)

Trace plots: new parameterization

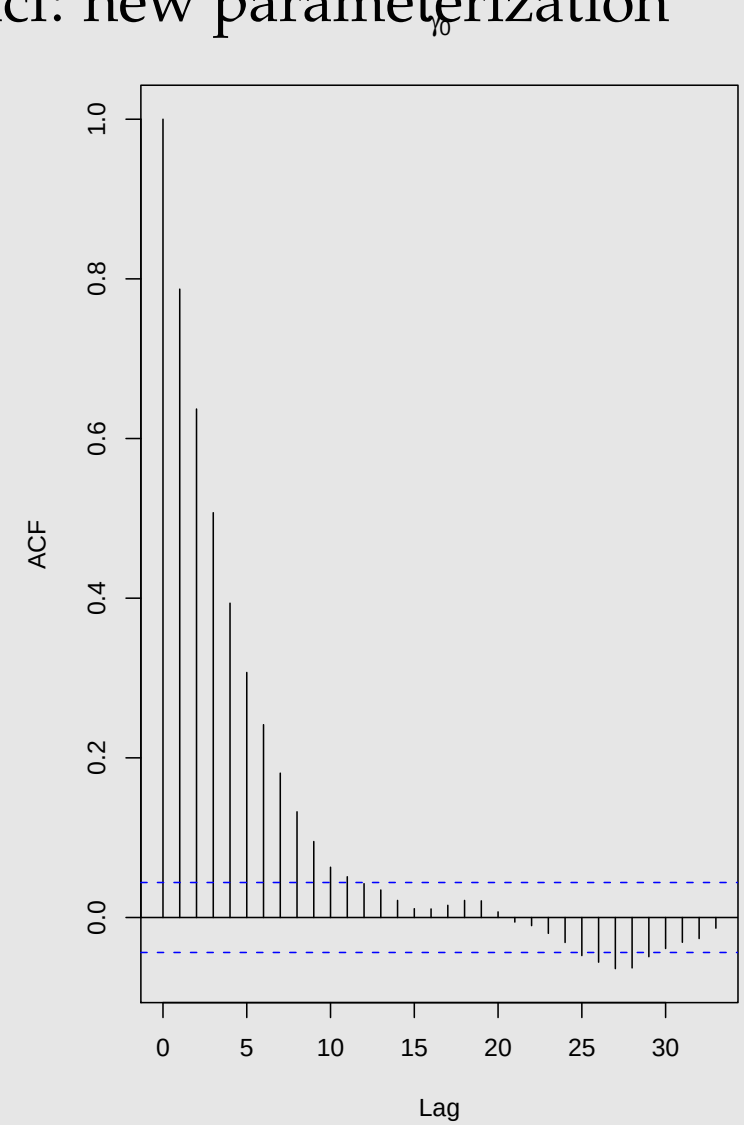


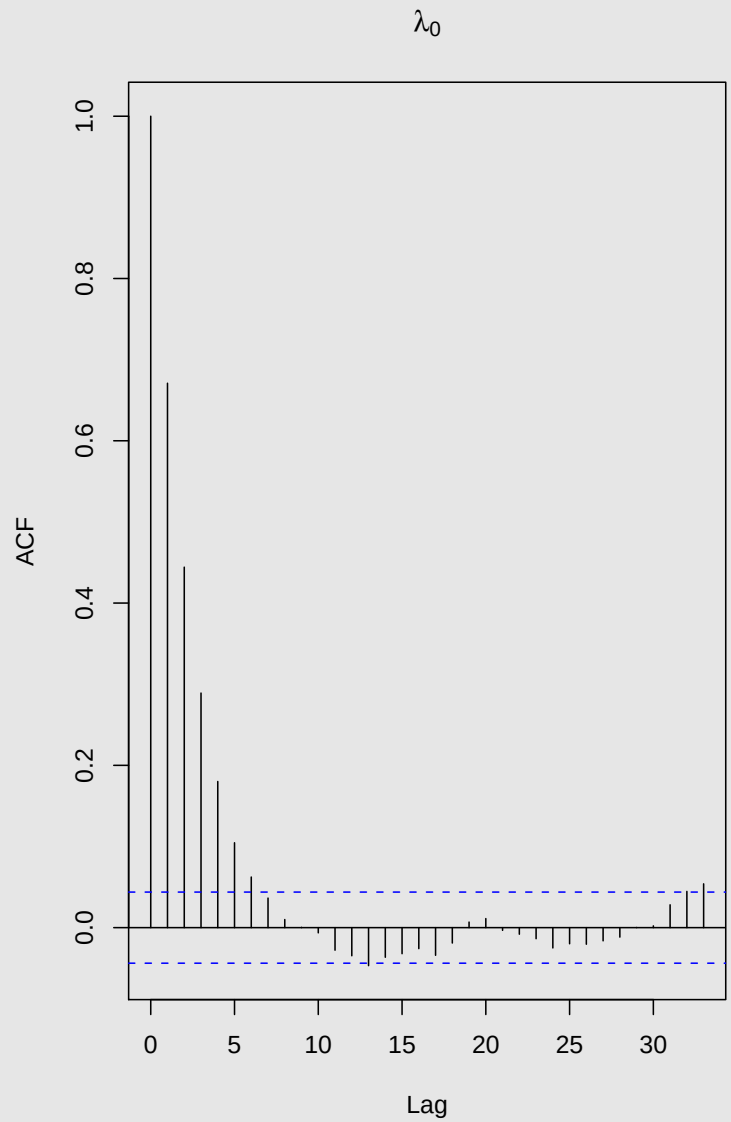


Gibbs Sampler: example

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

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

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

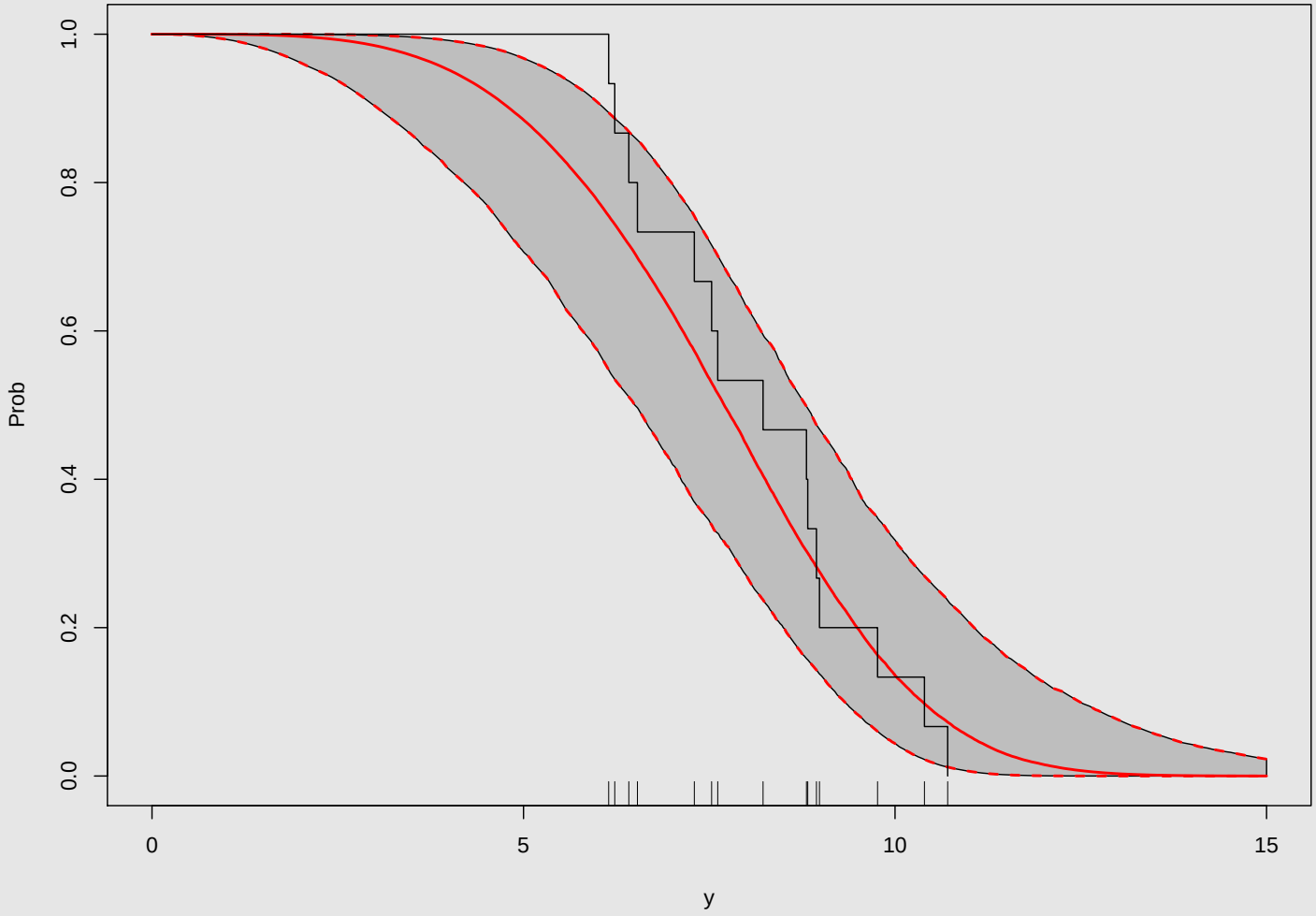
$$S(y) = \mathbb{P}[Y > y] = \exp\{-(y/\lambda)^{\gamma}\}.$$

For each $y \in \mathbb{R}^{+}$, we can compute this function for each sampled pair (γ, λ) 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 i.i.d. sample of size N, the variance of the Monte Carlo estimator $\widehat{I}_N(g)$ is

$$\frac{\mathbb{V}ar[g(X)]}{N}.$$

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

$$\frac{\mathbb{V}ar[g(X)]}{N_{\text{eff}}}$$

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_0, λ_0) parameterization	8.46	0.00
(γ, λ) parameterization	217.20	393.69

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

The value of zero (highlighted) is the result of apparent lack of convergence.

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, λ) parameterization, we have

$$\pi(\gamma, \lambda|\boldsymbol{y}) \propto L(\gamma, \lambda; \boldsymbol{y}) p(\gamma, \lambda_0(\gamma, \lambda)) |J(\alpha, \gamma)|$$

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

We can use proposal function p_0 which is the product of *Gamma* densities; we choose

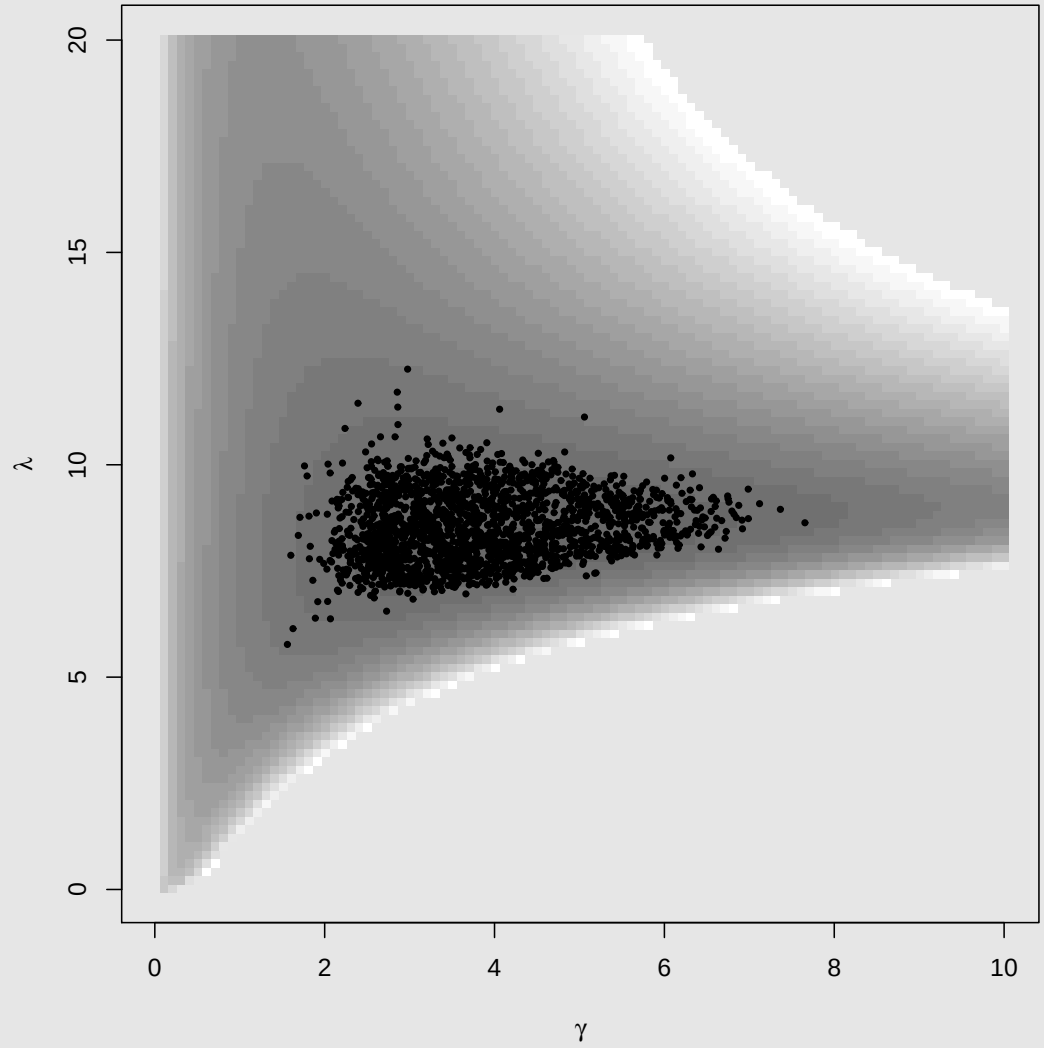
$$Gamma(2, 1/2) \qquad Gamma(4, 2)$$

for proposing (γ, λ), and then use numerical maximization to find M.

Rejection sampling for the Weibull example

Example (Weibull example)

Rejection sampling: acceptance rate is approximately 0.116.



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

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

then

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

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

For Metropolis-within-Gibbs, knowing $\pi(\boldsymbol{x}_1|\boldsymbol{x}_{(1)})$ 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

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

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

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

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.

Auxiliary variable methods

The Gibbs sampler is readily implementable for the Ising model, but can be extremely slow to converge to the stationary distribution. This is due in part to high dependence between the variables.

One method for speeding up simulation of variates from the Ising model is to introduce *auxiliary* or *latent* variables that extend the state-space of the Markov chain in such a way that the full-conditionals remain tractable, but which help to break the dependence.

The most common latent variable method for the Ising model is the *Swendsen-Wang algorithm*.

Auxiliary variable methods

The general principle is as follows. Consider target distribution $\pi_X(x)$ for random variables X, and let U be a collection of random variables such that the joint distribution $\pi_{X,U}(x,u)$ has marginal $\pi_X(x)$

$$\pi_X(x) = \int \pi_{X,U}(x,u)\, du$$

In a Gibbs sampler context, we could consider constructing a Markov chain using the two conditional distributions

$$\pi_{X|U}(x|u) \qquad\qquad \pi_{U|X}(u|x)$$

and then collecting the samples for X, which are necessarily samples from $\pi_X(x)$.

Auxiliary variable methods

Note that

$$\pi_{X,U}(x,u) = \pi_X(x)\pi_{U|X}(u|x)$$

We are free to select the U variables in whichever way is appropriate. We seek to introduce U such that

- the full conditionals are straightforward to sample
- the dependence structure amongst the X variables is broken.

For example, suppose

$$\pi_X(x) \propto \pi_X^{(0)}(x) \prod_{k=1}^{K} b_k(x)$$

where $\pi_X^{(0)}(x)$ can be sampled in straightforward fashion. Let

$$U_k | X = x \sim \mathit{Uniform}(0, b_k(x)) \qquad k = 1, \ldots, K$$

be a collection of conditionally independent random variables.

$$\begin{aligned}
\pi_{X,U}(x,u) &= \pi_X(x)\pi_{U|X}(u|x) \\
&= \left\{\pi_X^{(0)}(x)\prod_{k=1}^{K} b_k(x)\right\}\left\{\prod_{k=1}^{K}\frac{1}{b_k(x)}\mathbb{1}_{[0,b_k(x)]}(u_k)\right\} \\
&= \left\{\pi_X^{(0)}(x)\right\}\left\{\prod_{k=1}^{K}\mathbb{1}_{[0,b_k(x)]}(u_k)\right\}
\end{aligned}$$

Therefore

$$\pi_{X|U}(x|u) \propto \pi_X^{(0)}(x)\prod_{k=1}^{K}\mathbb{1}_{[u_k,\infty)}(b_k(x))$$

that is, proportional to $\pi_X^{(0)}(x)$ with support restricted by

$$b_k(x) \geqslant u_k.$$

for $k = 1, \ldots, K$.

Example (Swendsen-Wang Algorithm)

In the Ising model

$$\pi_X(x) \propto \exp\left\{\beta \sum_l \sum_{m \in \partial l} \mathbb{1}_{\{x_m\}}(x_l)\right\}.$$

where ∂l constitute the neighbours of site l. In the simplest case, if site l is at Cartesian location (i, j), the neighbours are the four sites immediately

- to the left $(i-1, j)$
- to the right $(i+1, j)$
- above $(i, j+1)$
- below $(i, j-1)$

site (i, j).

Example (Swendsen-Wang Algorithm)

The full conditional distributions for the Gibbs sampler are given by

$$X_l | X_{\partial l} = x_{\partial l} \sim \textit{Bernoulli}(\theta(\beta; x_{\partial l}))$$

that is

$$\mathbb{P}[X_l = 1 | X_{\partial l} = x_{\partial l}] = \theta(\beta; x_{\partial l})$$

where

$$\theta(\beta; x_{\partial l}) = \frac{\exp\{\beta N_{l1}\}}{\exp\{\beta N_{l0}\} + \exp\{\beta N_{l1}\}}$$

This algorithm updates one x value at a time, either in a *fixed* or *random* scan fashion.

Example (Swendsen-Wang Algorithm)

The Swendsen-Wang algorithm introduces a collection of auxiliary bond variables, u. Consider two neighbouring sites, l and m, say. Let $\omega = 1 - e^{-\beta}$. Define the binary bond random variable U_{lm}, conditional on X_l and X_m, to have distribution described by

$$\mathbb{P}[U_{lm} = 1 | X_l = x_l, X_m = x_m] = \begin{cases} \omega & x_l = x_m \\ 0 & x_l \neq x_m \end{cases}$$

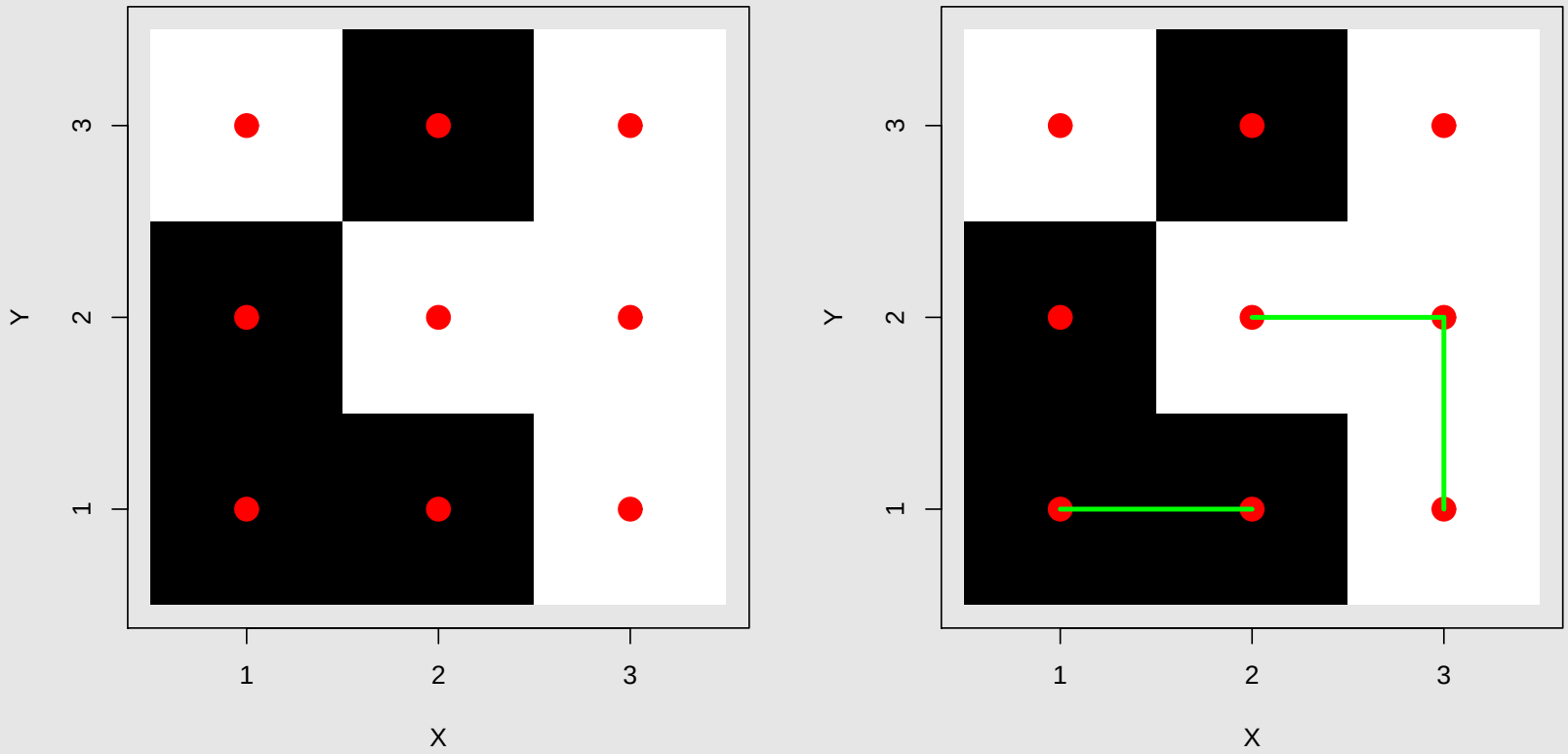
such that the collection of U variables are conditionally independent given X variables.

The bond is *present* if $U_{lm} = 1$, and *absent* otherwise.

Auxiliary variable methods

Example (Swendsen-Wang Algorithm)

Bond variables (green) between adjacent, similar x values.



Example (Swendsen-Wang Algorithm)

Consider the joint specification $\pi_{X,U}$ given by

$$\pi_{X,U}(x,u) \propto \prod_{l,m} \left\{ \omega \delta_{x_l x_m} \delta_{u_{lm} 1} + (1-\omega) \delta_{u_{lm} 0} \right\}$$

where the product is over all pairs of neighbouring sites, and $\delta_{ab} = 1$ if $a = b$, and zero otherwise. We have

$$\pi_{U_{lm}|X}(u|x) = \pi_{U_{lm}|X_l,X_m}(u|x_l,x_m) = \begin{cases} \omega^u (1-\omega)^{1-u} & x_l = x_m \\ \mathbb{1}_{\{0\}}(u) & x_l \neq x_m \end{cases}$$

for $u = 0, 1$.

Example (Swendsen-Wang Algorithm)

Also marginalizing over the u variables by summing out over $u_{lm} = 0, 1$ can be achieved over each term in the product separately. We have

$$\begin{aligned}\sum_u \pi_{X,U}(x,u) &\propto \prod_{l,m} \left\{(1-\omega) + \omega\delta_{x_l x_m}\right\} \\ &= \prod_{l,m} \left\{e^{-\beta} + (1-e^{-\beta})\delta_{x_l x_m}\right\}.\end{aligned}$$

Multiplying through each term by $e^{-\beta}$ yields

$$\sum_u \pi_{X,U}(x,u) \propto \prod_{l,m} \left\{1 + (e^{\beta}-1)\delta_{x_l x_m}\right\}.$$

Example (Swendsen-Wang Algorithm)

For each term in the product, when $x_l \neq x_m$, the term contributes 1; when $x_l = x_m$, the term contributes e^{β}. Hence

$$\sum_u \pi_{X,U}(x,u) \propto \prod_{l,m} \exp\{\beta \delta_{x_l x_m}\} = \exp\left\{\beta \sum_{l,m} \delta_{x_l x_m}\right\} = \pi_X(x)$$

and so the correct marginal model is recovered.

Example (Swendsen-Wang Algorithm)

Marginalizing over the X variables, we have

$$\begin{aligned}\sum_x \pi_{X,U}(x,u) &\propto \sum_x \prod_{l,m} \left\{\omega \delta_{x_l x_m} \delta_{u_{lm}1} + (1-\omega)\delta_{u_{lm}0}\right\} \\ &\propto \omega^{n(u)}(1-\omega)^{n_b - n(u)}\end{aligned}$$

where $n(u)$ is the total number of bonds present

$$n(u) = \sum_{l,m} u_{lm}$$

and n_b is the total number of bonds possible.

Example (Swendsen-Wang Algorithm)

We now consider running a Gibbs sampler for $\pi_{X,U}$.

- The full conditional distribution of $U|X = x$ is determined by the definition above, that is,

$$\pi_{U|X}(u|x) = \prod_{l,m} \pi_{U_{lm}|X_l,X_m}(u_{lm}|x_l, x_m)$$

where

$$\pi_{U_{lm}|X_l,X_m}(u_{lm}|x_l, x_m) = \begin{cases} \omega^u(1-\omega)^{1-u} & x_l = x_m \\ \mathbb{1}_{\{0\}}(u) & x_l \neq x_m \end{cases}$$

for $u = 0, 1$, with the product being over all neighbouring sites. Thus the U variables can be updated independently, conditional on X.

Example (Swendsen-Wang Algorithm)

- The full conditional distribution of $X|U$ is defined by the fact that

$$\pi_{X|U}(x|u) \propto \pi_{X,U}(x,u) \propto \prod_{l,m} \left\{ \omega \delta_{x_l x_m} \delta_{u_{lm} 1} + (1-\omega) \delta_{u_{lm} 0} \right\}$$

  and that bonds link "clusters" of sites that have the same value. The conditional distribution does not depend on the **value** of each x_l, merely whether or not $x_l = x_m$.

  Conditional on u, clusters can assume the label 0 or 1, **independently** with **equal probability**.

Example (Swendsen-Wang Algorithm)

Thus the Gibbs sampler can be implemented by sampling from the two full conditional distributions

$$\pi_{X|U}(x|u) \qquad\qquad \pi_{U|X}(u|x)$$

recursively, both of which involve sampling from trivial distributions.

There is some computational overhead in implementing Swendsen-Wang, in particular, tracing the clusters.

Example (Swendsen-Wang Algorithm)

For a specific computational example: with grid size $N = 128$, $\beta = 0.85$, the Gibbs sampler and Swendsen-Wang (SW) algorithms were run in R for 500 iterations.

- Gibbs sampler: 11 minutes, 30 seconds,
- Swendsen-Wang: 30 minutes, 3 seconds.

that is, roughly 2.5 slower for SW. However, the SW chain converges much more quickly; the Gibbs sampler chain exhibits slower convergence for larger values of β.

Auxiliary variable methods

Example (Swendsen-Wang Algorithm)

With $\beta = 0.85$, the performance of the two methods is not appreciably different; when $\beta = 0.95$, the behaviour is radically different.

The Ising model exhibits *critical behaviour* when $\beta = 1$: for $\beta > 1$, entire configurations of one "colour" occur with high probability.

As $\beta \uparrow 1$, the Gibbs sampler exhibits increasingly poor behaviour ("critical slowing"), whereas Swendsen-Wang does not.

Example (Swendsen-Wang Algorithm)

To demonstrate this, we track the summary statistic

$$T(x) = \sum_{l} \sum_{m \in \partial l} \mathbb{1}_{\{x_m\}}(x_l)$$

across iterations. This is the number of neighbours that are alike in the configuration.

We do this for the three cases

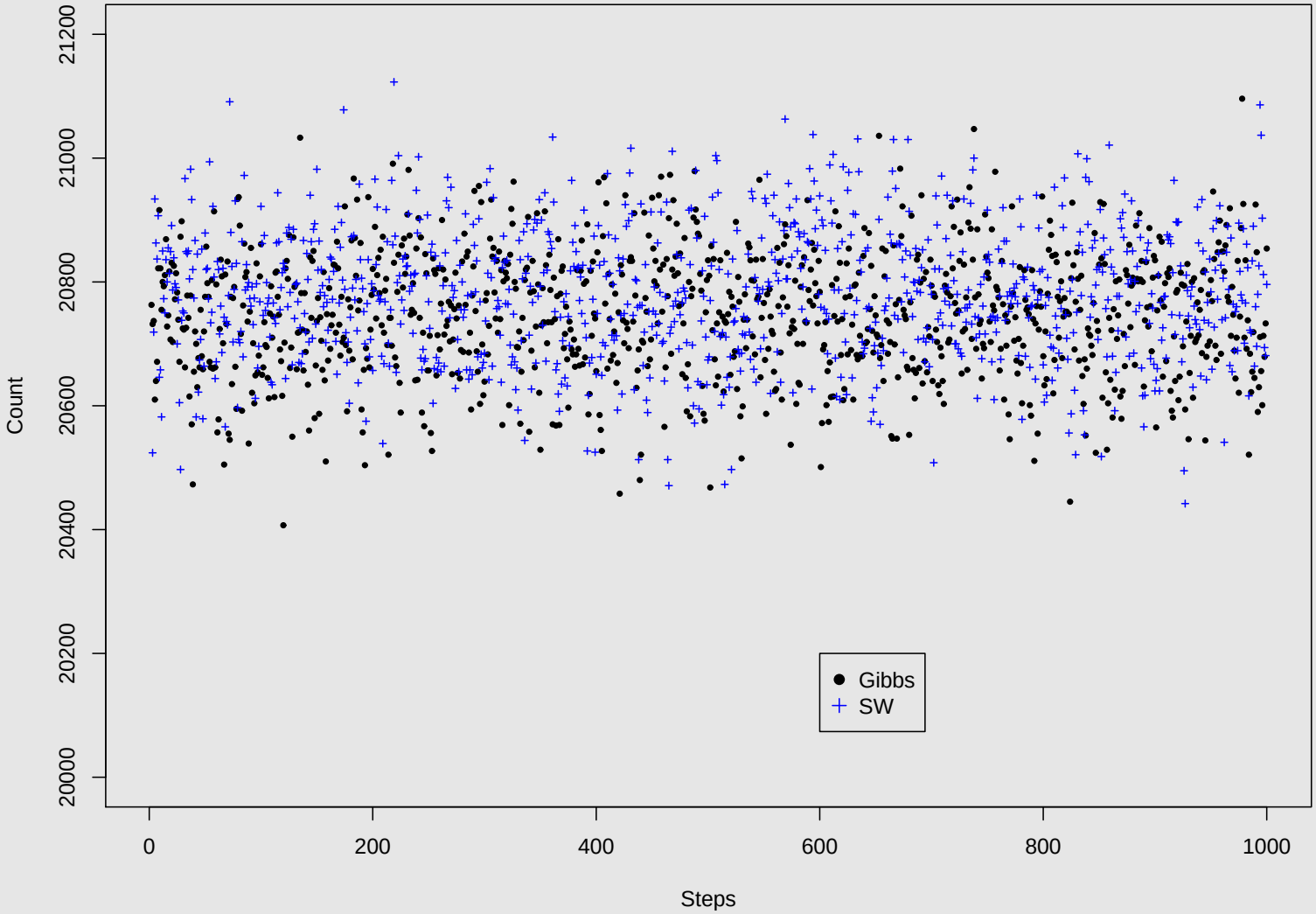
(i) $\beta = 0.5$

(ii) $\beta = 0.85$

(iii) $\beta = 0.95$.

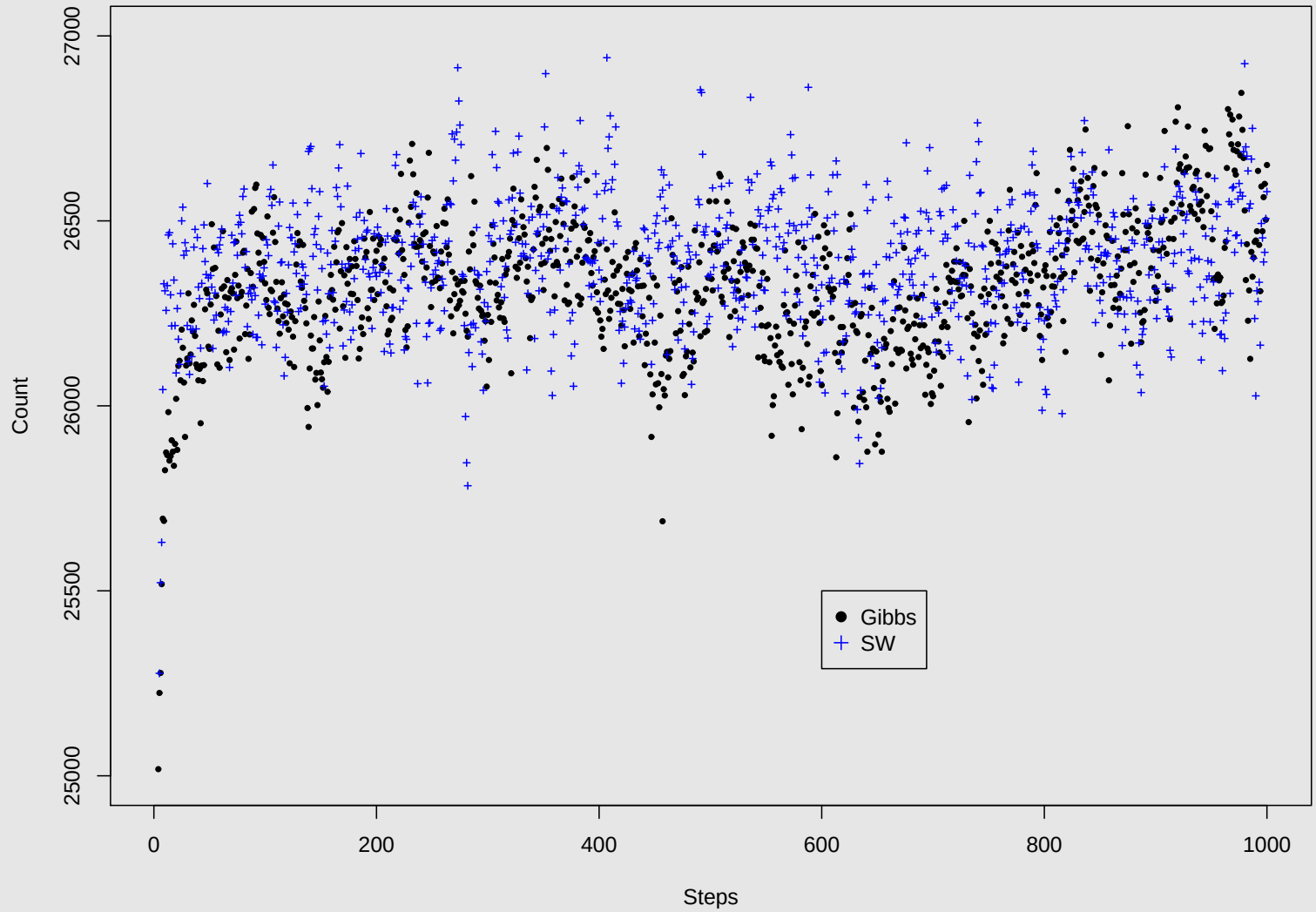
Example (Swendsen-Wang Algorithm)

$\beta = 0.5$



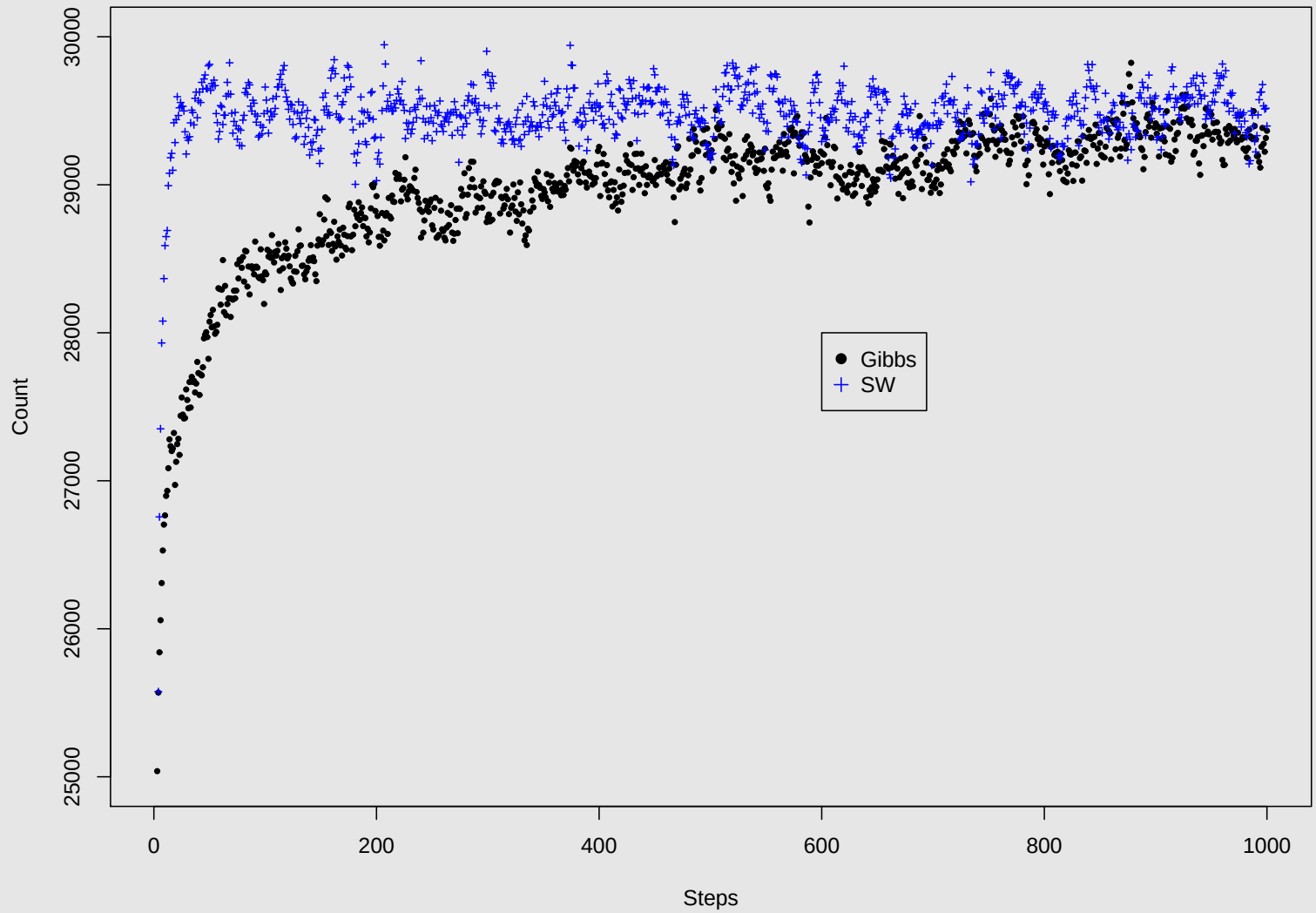
Example (Swendsen-Wang Algorithm)

$\beta = 0.85$



Example (Swendsen-Wang Algorithm)

$\beta = 0.95$



Auxiliary variable methods: Mixture Models

Example (Bayesian analysis of Finite Mixture Models)

Suppose that $Y_1 \ldots, Y_n$ are a random sample from the K component finite mixture model for $y \in \mathbb{R}$

$$f_{Y|\theta}(y|\theta) = \sum_{k=1}^{K} \omega_k f_k(y|\theta_k) \qquad \sum_{k=1}^{K} \omega_k = 1$$

for $0 < \omega_k < 1$ where $f_1, \ldots, f_K$ are component densities. The likelihood is

$$L_Y(\theta; \mathbf{y}) = \prod_{i=1}^{n} \left\{ \sum_{k=1}^{K} \omega_k f_k(y_i|\theta_k) \right\}$$

For a Bayesian analysis, we specify a prior distribution

$$p(\theta) = p(\omega_1, \ldots, \omega_K, \theta_1, \ldots, \theta_K)$$

Auxiliary variable methods: Mixture Models

Example (Bayesian analysis of Finite Mixture Models)

Here

$$\pi(\theta|\boldsymbol{y}) \propto L_Y(\theta;\boldsymbol{y})p(\theta).$$

Consider the discrete auxiliary variables $U_1, \ldots, U_n$ introduced into the likelihood density for each i via the identity

$$f_{Y|\theta}(y|\theta) = \sum_{k=1}^{K} f_{Y|U,\theta}(y|u,\theta) f_U(u)$$

where U has the $Multinomial(1, \omega_1, \ldots, \omega_K)$ distribution

$$f_U(u) = \prod_{k=1}^{K} \omega_k^{\mathbb{1}_{\{k\}}(u)}$$

and $f_{Y|U,\theta}(y|u,\theta) \equiv f_u(y|\theta)$.

Auxiliary variable methods: Mixture Models

Example (Bayesian analysis of Finite Mixture Models)

The U variables are discrete labels indicating which of the component densities, $f_1, \ldots, f_K$, each data point is sampled from, with

$$\mathbb{P}[U = k] = \omega_k \qquad k = 1, \ldots, K$$

Consider the joint posterior distribution on the unknown (unobservable) quantities

$$\pi_{\theta,\boldsymbol{u}}(\theta, \boldsymbol{u}|\boldsymbol{y}) \propto L(\boldsymbol{y}; \theta, \boldsymbol{u}) p(\theta, \boldsymbol{u})$$

Here

$$p(\theta, \boldsymbol{u}) = p(\theta) p(\boldsymbol{u}|\theta) = p(\theta) \prod_{i=1}^{n} \left\{ \prod_{k=1}^{K} \omega_k^{\mathbb{1}_{\{k\}}(u)} \right\}$$

where $p(\theta)$ is a joint prior on the ω and θ variables.

Auxiliary variable methods: Mixture Models

Example (Bayesian analysis of Finite Mixture Models)

Running the Gibbs sampler on the full conditional distributions

$$\pi_{\theta|\boldsymbol{U}}(\theta|\boldsymbol{u},\boldsymbol{y}) \qquad\qquad \pi_{\boldsymbol{U}|\theta}(\boldsymbol{u}|\theta,\boldsymbol{y})$$

yields samples from the joint posterior. Marginally, the Gibbs sampler produces a sample from

$$\pi_{\theta}(\theta|\boldsymbol{y})$$

which is the target posterior of interest.

For the first conditional, given $\boldsymbol{U} = \boldsymbol{u}$, we have a factorization

$$\pi_{\theta|\boldsymbol{U}}(\theta|\boldsymbol{u},\boldsymbol{y}) = \pi_{\theta_1,\ldots,\theta_K|\boldsymbol{U}}(\theta_1,\ldots,\theta_K|\boldsymbol{u},\boldsymbol{y})\pi_{\omega|\boldsymbol{U}}(\omega_1,\ldots,\omega_K|\boldsymbol{u},\boldsymbol{y})$$

and can update these parameters independently in two parts.

Auxiliary variable methods: Mixture Models

Example (Bayesian analysis of Finite Mixture Models)

- $\pi_{\theta|\boldsymbol{u}}(\theta|\boldsymbol{u},\boldsymbol{y})$ (Part I): if the prior on the θ parameters factorizes into independent components, then

$$\pi_{\theta_1,\ldots,\theta_K|\boldsymbol{u}}(\theta_1,\ldots,\theta_K|\boldsymbol{u},\boldsymbol{y}) = \prod_{k=1}^{K} \pi_{\theta_k|\boldsymbol{u}}(\theta_k|\boldsymbol{u},\boldsymbol{y})$$

where

$$\pi_{\theta_k|\boldsymbol{u}}(\theta_k|\boldsymbol{u},\boldsymbol{y}) \propto \left\{ \prod_{i:u_i=k} f_k(y_i|\theta_k) \right\} p(\theta_k)$$

and we can update the θ parameters independently conditional on $\boldsymbol{u}$. This may require a Metropolis-Hastings update.

Auxiliary variable methods: Mixture Models

Example (Bayesian analysis of Finite Mixture Models)

- $\pi_{\theta|\boldsymbol{U}}(\theta|\boldsymbol{u},\boldsymbol{y})$ (Part II): a standard choice for the prior for the ω parameters is the Dirichlet prior

$$p(\omega_1,\ldots,\omega_K) \propto \prod_{k=1}^{K} \omega_k^{\alpha_k-1}$$

$$\Longrightarrow p(\omega_1,\ldots,\omega_K|\boldsymbol{u},\boldsymbol{y}) \propto \prod_{k=1}^{K} \omega_k^{n_k+\alpha_k-1}$$

where, for $k = 1,\ldots,K$, $n_k = \sum_{i=1}^{n} \mathbb{1}_{\{k\}}(u_i)$ is the count of the U variables that are set equal to k. Thus the full conditional distribution is also Dirichlet, and can be sampled directly.

Auxiliary variable methods: Mixture Models

Example (Bayesian analysis of Finite Mixture Models)

- $\pi_{\boldsymbol{U}|\theta}(\boldsymbol{u}|\theta, \boldsymbol{y})$: by conditional independence of the Y given θ, we must have

$$\pi_{\boldsymbol{U}|\theta}(\boldsymbol{u}|\theta, \boldsymbol{y}) = \prod_{i=1}^{n} \pi_{U_i|\theta}(u_i|\theta, y_i)$$

where, by Bayes theorem

$$\pi_{U_i|\theta}(u_i|\theta, y_i) = \frac{f_{Y_i}(y_i|u_i, \theta) f_{U_i}(u_i|\theta)}{\sum_{k=1}^{K} f_{Y_i}(y_i|k, \theta) f_{U_i}(k|\theta)} = \frac{f_{u_i}(y_i|\theta_{u_i})\omega_{u_i}}{\sum_{k=1}^{K} f_k(y_i|\theta_k)\omega_k}$$

which is a discrete distribution on $\{1, \ldots, K\}$. Thus updating the U variables is straightforward.

Auxiliary variable methods: Mixture Models

Example (Bayesian analysis of Finite Mixture Models)

Note: this formulation reveals a connection with the EM algorithm; the full conditional distribution of U_i is identical to the conditional distribution of the missing data, given the observed data (see p. 128).

Indeed, the entire formulation is similar, the only difference is that we replace the Expectation and Maximization steps by sampling steps. The Gibbs sampler does not perform maximization to get the ML estimate, it produces samples from the posterior distribution of the parameters.

Auxiliary variable methods: Mixture Models

Example (Bayesian analysis of Finite Mixture Models)

Some care is needed in interpreting the Gibbs sampler output, as the component labels are, in fact, arbitrarily assigned in the mixture formulation. Therefore, the U variables can exhibit *label-switching*.

A simple solution is to add an additional Metropolis-Hastings step at the end of each Gibbs sampler iteration that permutes the labels on the U variables, that is, for $K = 3$, if the random permutation is $(3, 1, 2)$, then

- All Us labelled 1 are relabelled 3
- All Us labelled 2 are relabelled 1
- All Us labelled 3 are relabelled 2

This Metropolis-Hastings proposal is **always accepted**, as it does not change the posterior density value.

Auxiliary variable methods: Missing data

Auxiliary variable methods can also be used to handle *missing data* problems. Missing data can arise in problems due to

- drop-out
- data handling issues
- censoring/truncation

The missingness can occur

- *completely at random* (independently of the observed and unobserved data)
- *at random* (dependent on the observed data, but independent of the unobserved data)
- *not at random* (dependent on the observed data and unobserved data)

Auxiliary variable methods: Missing data

Missingness can occur in the

- *response* variable Y
- *covariate* X

We will denote the missing values by $X^\star$ and $Y^\star$.

We consider a Bayesian formulation of the missing data inference problem; maximum likelihood estimation can be handled using the EM algorithm. We consider the joint likelihood of the observed and unobserved data, and treat the unobserved values as quantities to be updated during the MCMC.

The parameter vector θ can now also include parameters in the stochastic missingness mechanism.

Auxiliary variable methods: Missing data

In the case of i.i.d. data, where the missing data are considered independent of the observed data, the **complete** data $(\boldsymbol{Y}, \boldsymbol{Y}^\star)$ have likelihood

$$L(\boldsymbol{y}, \boldsymbol{y}^\star; \theta) = \left\{ \prod_{i=1}^{n} f_Y(y_i; \theta) \right\} \left\{ \prod_{i=1}^{n^\star} f_Y(y_i^\star; \theta) \right\}$$

In this case, a Gibbs sampler can be used to update the quantities $(\theta, \boldsymbol{Y}^\star)$ from their full conditionals. For θ,

$$\pi_{\theta|\boldsymbol{Y},\boldsymbol{Y}^\star}(\theta|\boldsymbol{y}, \boldsymbol{y}^\star) \propto L(\boldsymbol{y}, \boldsymbol{y}^\star; \theta) p(\theta)$$

that is, we perform Bayesian inference using the complete data likelihood.

Auxiliary variable methods: Missing data

For elements of $\boldsymbol{Y}^\star$, the full conditional distribution is merely the density $f_Y(y_i^\star;\theta)$. Most typically, however, this kind of missing data occurs due to censoring, that is, we have information that

$$Y_i^\star > c_i$$

say. In this case, the full conditional distribution used in the Gibbs sampler must respect the censoring constraint, that is, the full conditional is

$$f_Y(y_i^\star;\theta,c_i) \propto \mathbb{1}_{(c_i,\infty)}(y_i^\star) f_Y(y_i^\star;\theta)$$

Conditional on $(\boldsymbol{y},\boldsymbol{y}^\star)$, the full conditional distribution of θ is not affected by the censoring, so is identical to above.

Auxiliary variable methods: Missing data

For missing covariate data in a regression setting, sampling the missing covariates involves an extra component. Consider the full conditional distribution for missing covariates $\boldsymbol{X}^\star$

$$\pi_{\boldsymbol{X}^\star|\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Y}^\star,\theta}(\boldsymbol{x}^\star|\boldsymbol{x},\boldsymbol{y},\boldsymbol{y}^\star,\theta)$$

By Bayes theorem, this is proportional to

$$f_{\boldsymbol{Y},\boldsymbol{Y}^\star|\boldsymbol{X},\boldsymbol{X}^\star,\theta}(\boldsymbol{y},\boldsymbol{y}^\star|\boldsymbol{x},\boldsymbol{x}^\star,\theta)\, f_{\boldsymbol{X}^\star|\boldsymbol{X},\theta}(\boldsymbol{x}^\star|\boldsymbol{x},\theta)$$

The first term is the complete data likelihood for a regression problem. The second term most typically simplifies to

$$f_{\boldsymbol{X}^\star|\theta}(\boldsymbol{x}^\star|\theta)$$

under the assumption of random sampling of the response/covariate pairs.

Multiple chain MCMC

The Metropolis-Hastings algorithm is a general purpose algorithm that is extremely useful in producing samples from arbitrary multivariate distributions. The principal problems it encounters in applications relate to

- **Choice of starting values:** Poorly chosen starting values can result in slow convergence; often deterministic numerical optimization methods can be used effectively to determine good starting values.
- **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:** The MH algorithm can sometimes get trapped in localized regions of high probability density; this can sometimes be overcome by using *multiple chain* MCMC methods.

Multi-Mode Distributions

Example (Mixture of Bivariate Normals)

Suppose

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

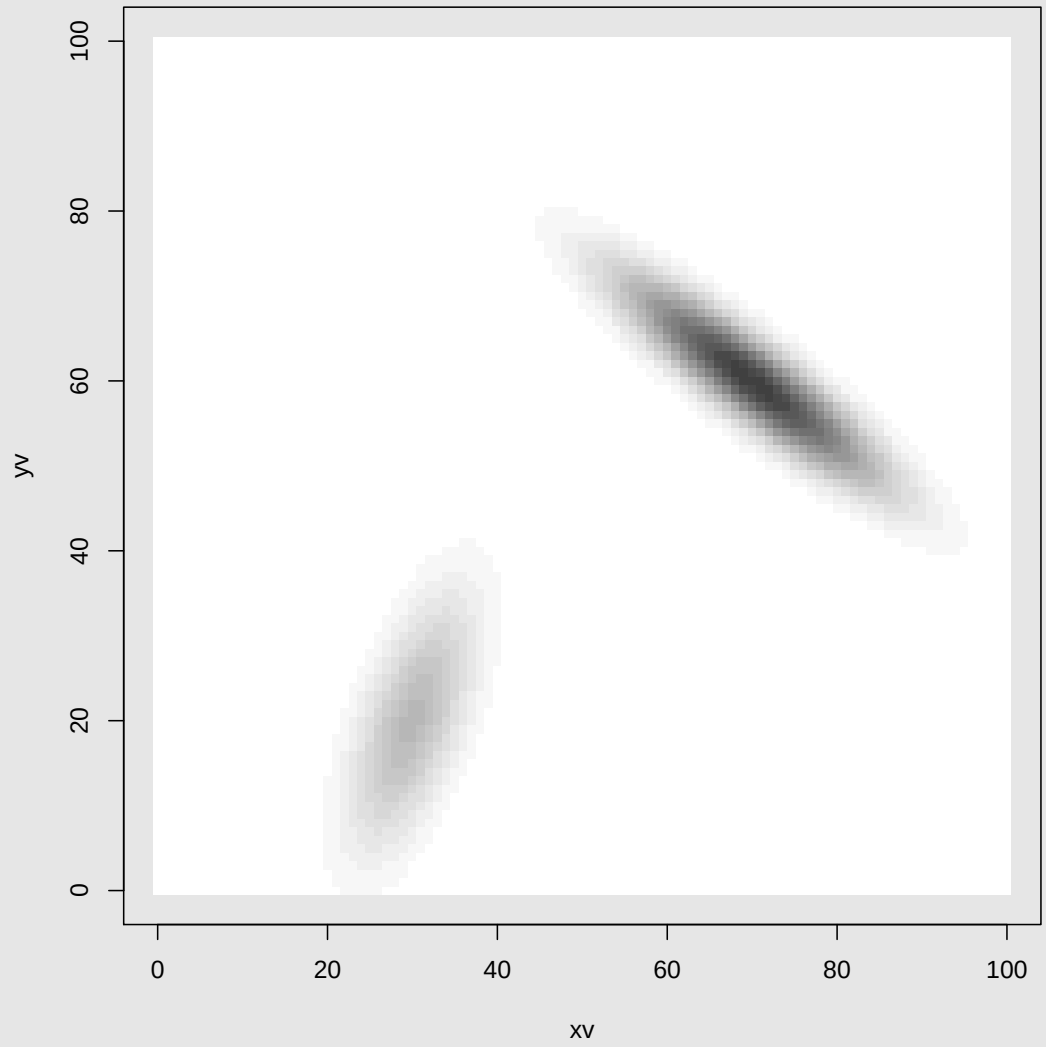
where $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are relatively well separated (in light of the covariance matrices Σ_1 and Σ_2.

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

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

This density has a single mode at $\boldsymbol{\theta}$, 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.

In both of these cases, it is easy to see that the samples produced at stationarity are samples from π.

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

Consider the joint distribution for $\boldsymbol{X} = (X_1, \ldots, X_M)$, $\pi^M(x)$, defined by

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

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.

Multiple chain MCMC: Types of move

1. **Mutation:** Select $m \in \{1, \ldots, M\}$, and perform a Metropolis-Hastings 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.

2. **Exchange:** Select $l, m \in \{1, \ldots, M\}$, and propose the swap

$$x_l \longleftrightarrow x_m$$

   Accept this proposal with probability

$$\alpha = \min \left\{ 1, \frac{\pi_l(x_m) \pi_m(x_l)}{\pi_l(x_l) \pi_m(x_m)} \right\}.$$

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.

Multiple chain MCMC: Types of move

This move can be utilized in the case of **vector** state spaces. If each $\boldsymbol{x}$ is d-dimensional, then the move becomes

3*. **Snooker:** Select $l, m \in \{1, \ldots, M\}$, and, in an attempt to update $\boldsymbol{x}_l$, make the proposal $\boldsymbol{z}$

$$\boldsymbol{z} = \boldsymbol{x}_m + u(\boldsymbol{x}_l - \boldsymbol{x}_m)$$

where u is a scalar random variate sampled from the density

$$f(u) \propto |u|^{d-1} \pi(\boldsymbol{x}_m + u(\boldsymbol{x}_l - \boldsymbol{x}_m))$$

and then update $\boldsymbol{x}_l$ to $\boldsymbol{z}$.

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 $\boldsymbol{x}_l$, proceed as follows: starting at $\boldsymbol{x}_m$, use conjugate gradient or steepest descent to find the local mode of π, and denote this $\widehat{\boldsymbol{x}}_m$.

   Then make the proposal $\boldsymbol{z}$

   $$\boldsymbol{z} = \widehat{\boldsymbol{x}}_m + u(\boldsymbol{x}_l - \widehat{\boldsymbol{x}}_m)$$

   where u is a scalar random variate sampled from the density

   $$f(u) \propto |u|^{d-1} \pi(\widehat{\boldsymbol{x}}_m + u(\boldsymbol{x}_l - \widehat{\boldsymbol{x}}_m))$$

   and then update $\boldsymbol{x}_l$ to $\boldsymbol{z}$.

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 $\boldsymbol{x}_l$ and $\boldsymbol{x}_m$ by

$$\boldsymbol{x}_l^{\text{new}} = (x_{l1}, \ldots, x_{lj}, x_{mj+1}, \ldots, x_{md})$$

$$\boldsymbol{x}_m^{\text{new}} = (x_{m1}, \ldots, x_{mj}, x_{lj+1}, \ldots, x_{ld})$$

that is, exchange the $(j+1, \ldots, d)$ portions of each vector.

Accept this proposal with probability

$$\alpha = \min\left\{1, \frac{\pi_l(\boldsymbol{x}_l^{\text{new}})\pi_m(\boldsymbol{x}_m^{\text{new}})}{\pi_l(\boldsymbol{x}_l)\pi_m(\boldsymbol{x}_m)}\right\}.$$

Parallel Tempering

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

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

and define

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

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

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

where

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

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

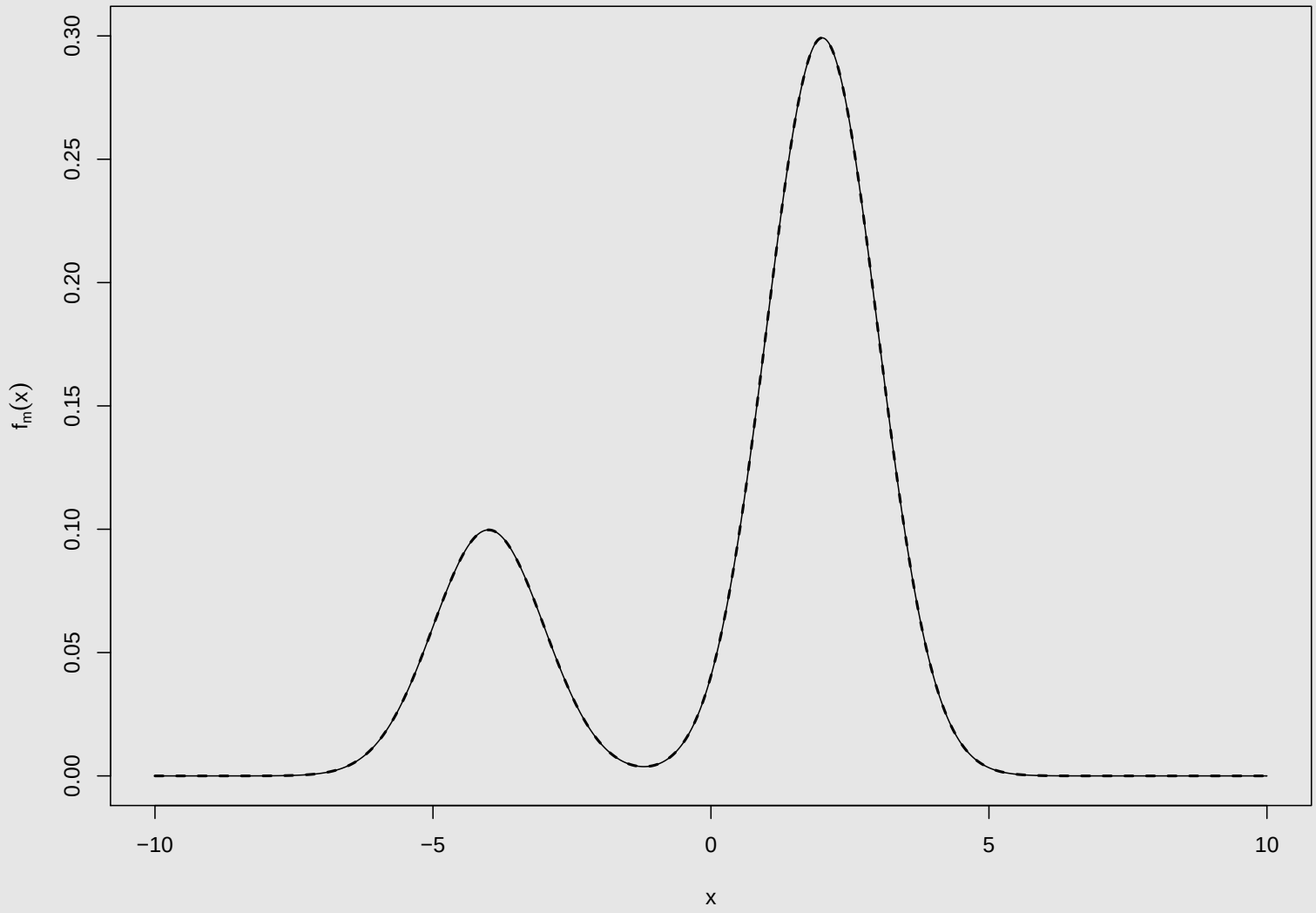
$$\pi(x) = \frac{1}{4}\phi(x+4) + \frac{3}{4}\phi(x-2)$$

the tempered distributions have modes that become increasingly flat.

Parallel Tempering

Example (Two component Normal mixture)

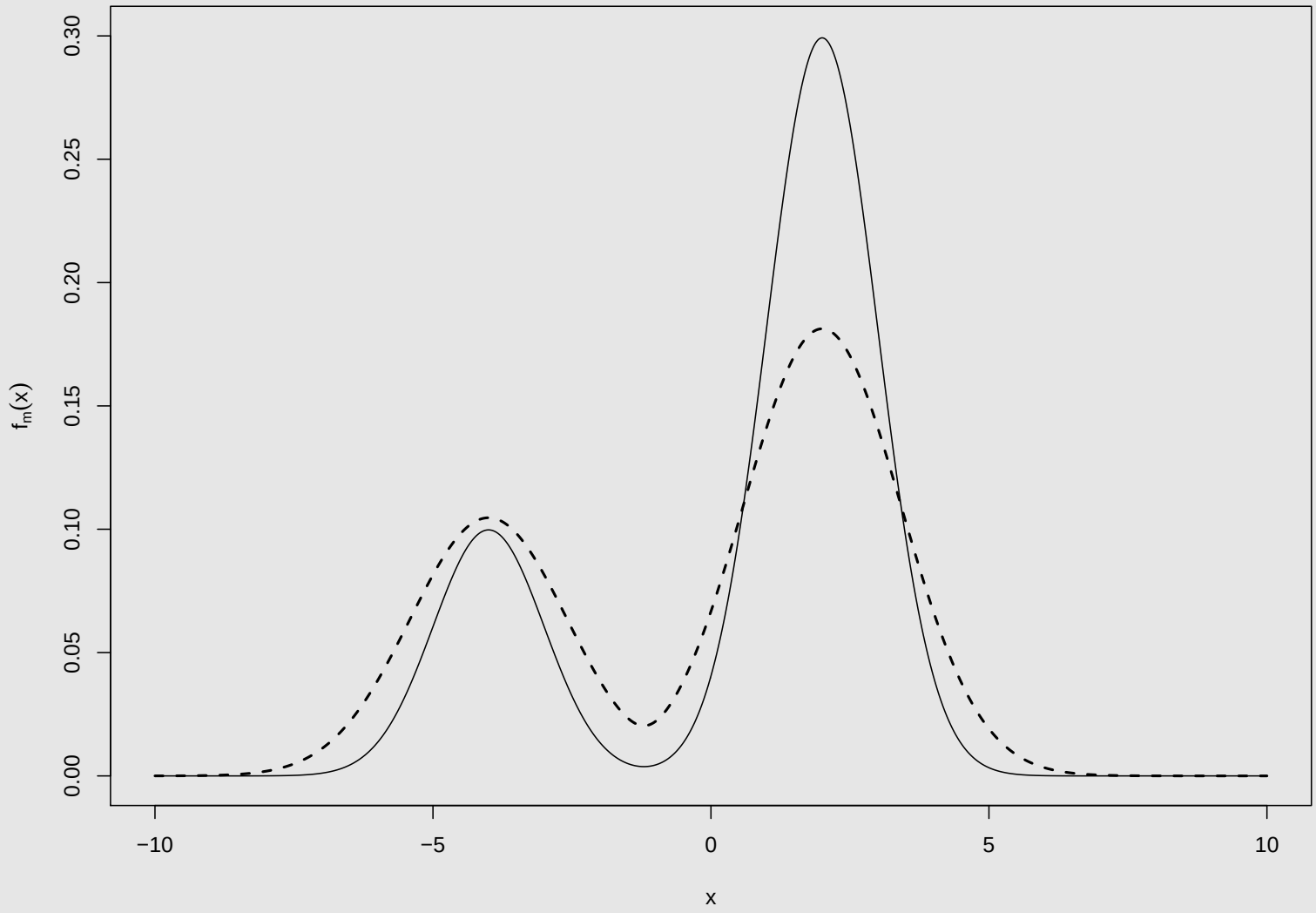
$T_1 = 1$:



Parallel Tempering

Example (Two component Normal mixture)

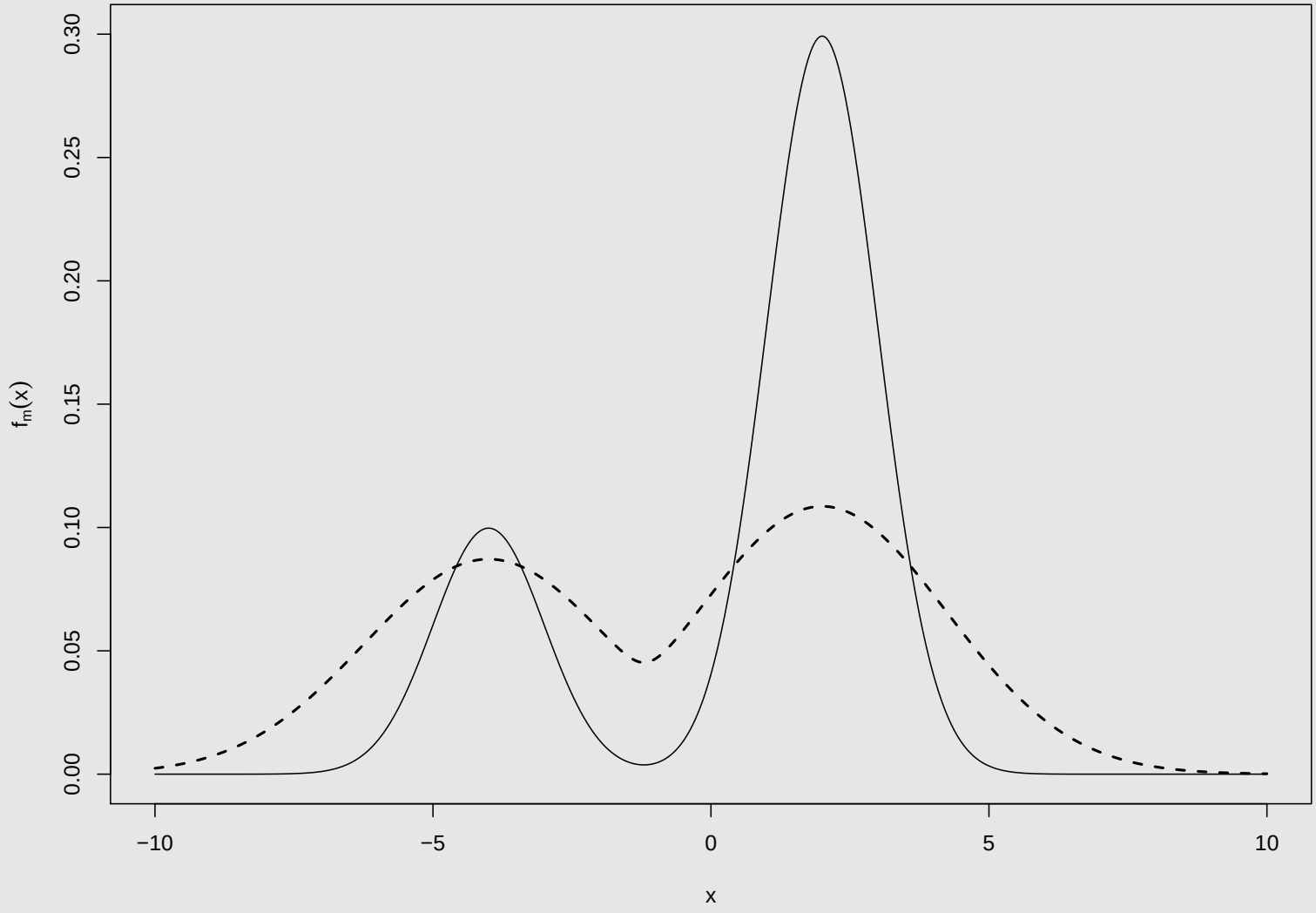
$T_2 = 2$:



Parallel Tempering

Example (Two component Normal mixture)

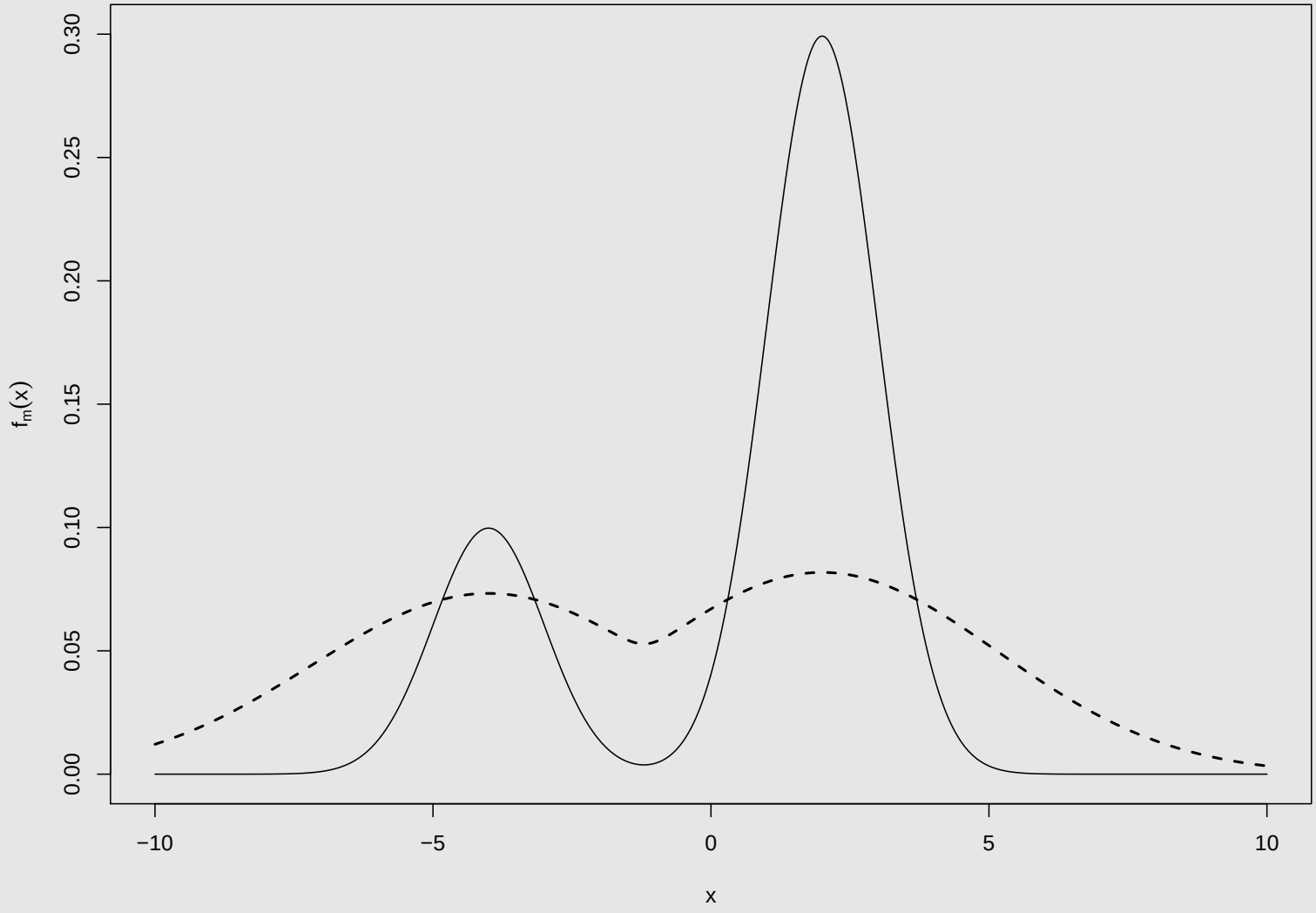
$T_3 = 5$:



Parallel Tempering

Example (Two component Normal mixture)

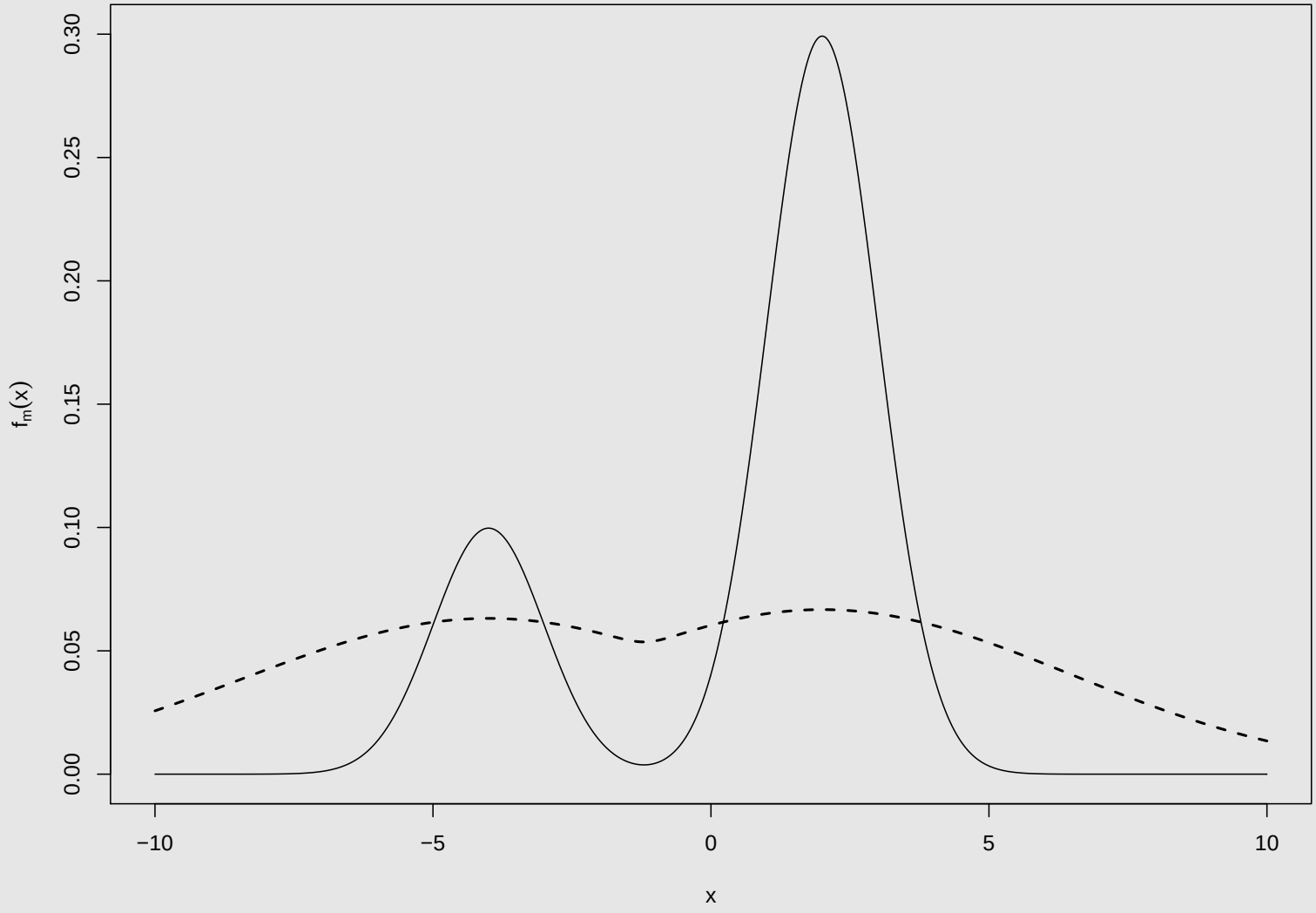
$T_4 = 10$:



Parallel Tempering

Example (Two component Normal mixture)

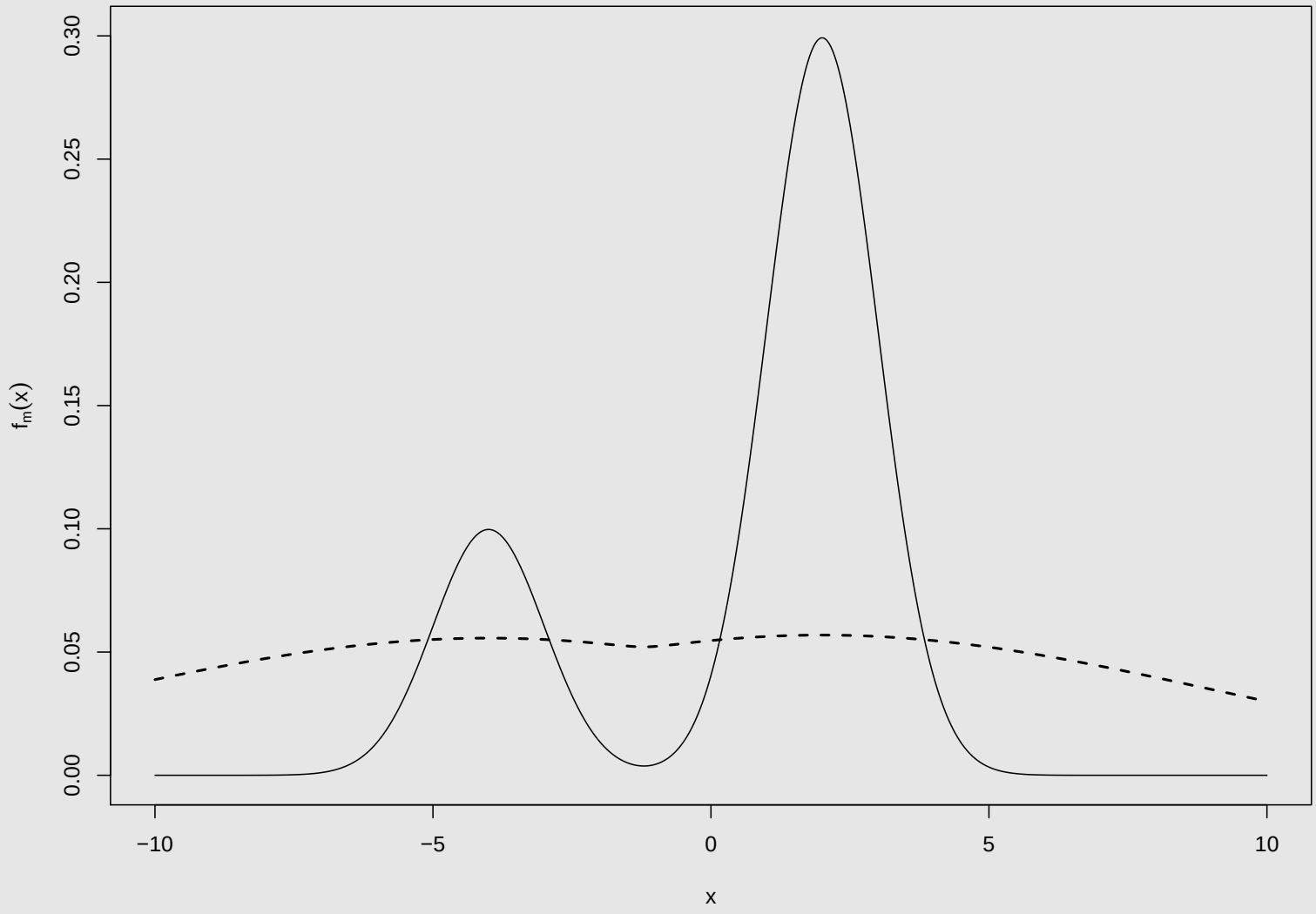
$T_5 = 20$:



Parallel Tempering

Example (Two component Normal mixture)

$T_6 = 50$:



Parallel Tempering

When T_m is large

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

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

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 Metropolis-Hastings 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

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

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

$$
\alpha = \min\left\{1, \frac{\pi_l(x_m)\pi_m(x_l)q_B(m,l)}{\pi_l(x_l)\pi_m(x_m)q_B(l,m)}\right\}
$$

Parallel Tempering

For the tempered distributions

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

or equivalently

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

The temperatures $T_2, \ldots, T_M$ should be chosen such that moves are accepted at an appropriate rate (neither too high nor too low).

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

Parallel Tempering

For the "down phase":

Propose z_M from MH kernel $q_M(y_M, z)$ in chain M
Propose z_{M-1} from MH kernel $q_{M-1}(z_M, z)$ in chain M-1
Propose z_{M-2} from MH kernel $q_{M-2}(z_{M-1}, z)$ in chain M-2
...
Propose z_2 from MH kernel $q_2(z_3, z)$ in chain 2
Propose z_1 from MH 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

$$\begin{aligned}
q_{\overline{U}}(z_1, \ldots, z_M, y_M) &= q_1(z_1, z_2) q_2(z_2, z_3) \cdots q_M(z_M, y_M) \\
q_{\overline{D}}(y_M, \ldots, y_1, x_1) &= q_M(y_M, y_{M-1}) q_{M-1}(y_{M-1}, y_{M-2}) \cdots q_1(y_1, x_1) \\
q_{\underline{U}}(x_1, y_1, \ldots, y_M) &= q_1(x_1, y_1) q_2(y_1, y_2) \cdots q_M(y_{M-1}, y_M) \\
q_{\underline{D}}(y_M, z_M, \ldots, z_1) &= q_M(y_M, z_M) q_{M-1}(z_M, z_{M-1}) \cdots q_1(z_2, z_1)
\end{aligned}$$

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 the move types outlined above (mutation, exchange, crossover, snooker) with tempered distributions in order to make exploration of the state space more effective.

The Metropolis-Hastings acceptance probability is reasonably straightforward to evaluate for each type of move. For a crossover move, a common approach to choosing the two "parent" vectors that participate in the crossover is to examine the current configurations, and to select the first parent to be $\boldsymbol{x}_l$ with probability p_l proportional to

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

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

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

This yields the **numerator** in the Hastings ratio

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

Considering the reverse move yields the denominator.

Simulated Tempering

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

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

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

$$(x_1, \tau_1), (x_2, \tau_2), \ldots, (x_M, \tau_M)$$

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

Simulated Tempering

Mutation and exchange moves can be implemented. For the **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

$$q_m((x_m, \tau_m), (x_m^{\text{new}}, \tau_m^{\text{new}}))$$

with the usual acceptance probability.

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

$$\alpha = \min\left\{1, \frac{\pi_l(x_m)\pi_m(x_l)}{\pi_l(x_l)\pi_m(x_m)}\right\}$$

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

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

Here

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

In this formula, the normalizing constant ratio

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

must be computed. Here

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

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. Define the tempered probability distribution

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

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

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

guarantees convergence of the algorithm 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 available as an option in the `optim` function in R.

Sequential Monte Carlo

Consider a sequence of probability distributions

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

defined on a sequence of state spaces

$$\mathbb{X}_1, \mathbb{X}_2,$$

where, for each n, $\mathbb{X}_n = \mathbb{X}^n$ for some $\mathbb{X}$.

Suppose that

$$\pi_n(\boldsymbol{x}_n) = \frac{g_n(\boldsymbol{x}_n)}{Z_n}$$

where $\boldsymbol{x}_n = (x_1, \ldots, x_n)$, g_n is an integrable non-negative function, and Z_n is a normalizing constant.

Such a series of distributions could arise in

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

$$\begin{aligned} y_t &= h(x_t; \theta) + \epsilon_t \\ x_t &= \mu + \phi(x_{t-1} - \mu) + \varepsilon_t \end{aligned}$$

   where the observed data $\{y_1, \ldots, y_n\}$ are related to the unobserved states $\{x_1, \ldots, x_n\}$.

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

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

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 x_1
   - π_2 being the posterior computed from x_1, x_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

$$\mathbb{E}_{\pi_n}[h(\boldsymbol{X}_n)] = \int h(\boldsymbol{x}_n)\pi_n(\boldsymbol{x}_n)\, d\boldsymbol{x}_n$$

Sequential Monte Carlo

Recall the basic principle of importance sampling:

$$\int h(x)p(x)\,dx = \int h(x)p(x)\frac{p_0(x)}{p_0(x)}\,dx = \int \frac{h(x)p(x)}{p_0(x)}p_0(x)\,dx$$

demonstrates that

$$\mathbb{E}_p[h(X)] = \mathbb{E}_{p_0}\left[\frac{h(X)p(X)}{p_0(X)}\right]$$

so that an estimator of the LHS is

$$\widehat{I}_N^{(p_0)}(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{h(X_i)p(X_i)}{p_0(X_i)}$$

where $X_1, \ldots, X_N \sim p_0(.)$.

Here, write

$$\pi_n(\boldsymbol{x}_n) = \frac{g_n(\boldsymbol{x}_n)}{Z_n} = \frac{g_n(\boldsymbol{x}_n)}{\int g(\boldsymbol{x}_n)\, d\boldsymbol{x}_n}$$

$$= \frac{w_n(\boldsymbol{x}_n) p_n(\boldsymbol{x}_n)}{\int w_n(\boldsymbol{x}_n) p_n(\boldsymbol{x}_n)\, d\boldsymbol{x}_n}$$

where p_n is the importance distribution, and

$$w_n(\boldsymbol{x}_n) = \frac{g_n(\boldsymbol{x}_n)}{p_n(\boldsymbol{x}_n)}$$

is the importance weight.

Sequential Monte Carlo

If a i.i.d. sample of size N

$$\boldsymbol{x}_n^{(1)}, \ldots, \boldsymbol{x}_n^{(N)}$$

can be obtained from p_n, then the empirical distribution

$$\widehat{p}_n(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{\boldsymbol{x}_n^{(i)}}(\boldsymbol{x})$$

can be used to estimate π_n

$$\widehat{\pi}_n(\boldsymbol{x}) = \sum_{i=1}^{N} W_n^{(i)} \delta_{\boldsymbol{x}_n^{(i)}}(\boldsymbol{x})$$

where

$$W_n^{(i)} = \frac{w_n(\boldsymbol{x}_n^{(i)})}{N\widehat{Z}_n} \qquad \widehat{Z}_n = \frac{1}{N} \sum_{i=1}^{N} w_n(\boldsymbol{x}_n^{(i)})$$

Recall that we require the variance of the weights to be finite.

In a "static" Bayesian calculation, the posterior from n data points can be written

$$\pi_n(\theta|\boldsymbol{y}_n) = \frac{L(\theta;\boldsymbol{y}_n)\pi(\theta)}{\int L(\varphi;\boldsymbol{y}_n)\pi(\varphi)\,d\varphi}$$

so could consider setting

$$p_n(\theta) = \pi(\theta)$$

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

Sequential Importance Sampling

Using the chain-rule factorization

$$\pi_n(\boldsymbol{x}_n) = \pi_1(x_1) \prod_{j=2}^{n} \widetilde{\pi}_j(x_j|\boldsymbol{x}_{j-1})$$

where

$$\widetilde{\pi}_j(x_j|\boldsymbol{x}_{j-1}) = \frac{\pi_j(\boldsymbol{x}_j)}{\pi_{j-1}(\boldsymbol{x}_{j-1})}$$

is the conditional distribution of element x_j, given $\boldsymbol{x}_{j-1} = (x_1, \ldots, x_{j-1})$, it is possible to construct a *Sequential Importance Sampling* (SIS) density

$$p_n(\boldsymbol{x}_n) = p_1(x_1) \prod_{j=2}^{n} \widetilde{p}_j(x_j|\boldsymbol{x}_{j-1})$$

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

Sequential Importance Sampling

The importance sampling weight then takes the form

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

so therefore

$$\begin{aligned} w_n(\boldsymbol{x}_n) &= w_{n-1}(\boldsymbol{x}_{n-1}) \frac{\widetilde{\pi}_n(x_n|\boldsymbol{x}_{n-1})}{\widetilde{p}_n(x_n|\boldsymbol{x}_{n-1})} \\ w_{n-1}(\boldsymbol{x}_{n-1}) &= w_{n-2}(\boldsymbol{x}_{n-2}) \frac{\widetilde{\pi}_{n-1}(x_{n-1}|\boldsymbol{x}_{n-2})}{\widetilde{p}_{n-1}(x_{n-1}|\boldsymbol{x}_{n-2})} \end{aligned}$$

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

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

Sequential Importance Sampling

In the numerator of the weights, we have the conditional densities

$$\widetilde{\pi}_j(x_j|\boldsymbol{x}_{j-1}) = \frac{\pi_j(\boldsymbol{x}_j)}{\pi_{j-1}(\boldsymbol{x}_{j-1})}$$

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

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

$$p_{0j}(\boldsymbol{x}_j) \qquad j = 1, \ldots, n$$

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

Combining Importance Sampling and MCMC

Recall that

$$\int h(x)\pi(x)\,dx = \int h(x)\frac{\pi(x)}{p_0(x)}p_0(x)\,dx = \int h(x)w(x)p_0(x)\,dx$$

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

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

then also

$$\int \frac{\pi(z)}{p_0(z)}P(z,x)p_0(z)\,dz = \pi(x)$$

Combining Importance Sampling and MCMC

Thus

$$\int w(z)P(z,x)p_0(z)\,dz = \pi(x)$$

and hence, from the importance sampling identity

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

and the estimator

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

can be constructed, where

- $X_1, \ldots, X_N \sim p_0$
- $Z_1, \ldots, Z_N \sim f(z|x)$

where $f(z|x)$ is the conditional density for Z given X implied by $P(.,.)$.

Annealed Importance Sampling

Annealed Importance Sampling (AIS) uses a SIS approach, but with auxiliary densities **each in n dimensions** defined by

$$p_{0j}(\boldsymbol{x}_j) = \{p_0(\boldsymbol{x}_j)\}^{1-\xi_j}\{\pi_n(\boldsymbol{x}_j)\}^{\xi_j}$$

where p_0 is a "diffuse" distribution, and

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

This defines a "path" from p_0 to π_n. These distributions are utilized by employing a MCMC kernel.

Annealed Importance Sampling

The AIS algorithm proceeds as follows: dropping the dependence on n throughout

1. Sample $\boldsymbol{x}_0$ from p_{00}, set $w_0 = 1$
2. For $j = 1, 2, \ldots, n$,
   (i) Sample $\boldsymbol{x}_j$ from $P_j(\boldsymbol{x}_{j-1}, \boldsymbol{x})$, where P_j is a Markov kernel with invariant distribution p_{0j}.
   (ii) Set
   $$w_j = w_{j-1} \frac{p_{0j}(\boldsymbol{x}_j)}{p_{0\,j-1}(\boldsymbol{x}_{j-1})}$$
3. Return to 1 and repeat N times to produce N samples

   $$\boldsymbol{x}_n^{(1)}, \ldots, \boldsymbol{x}_n^{(N)}$$

   and weights

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

Annealed Importance Sampling

If P_j^R is the "reverse" kernel, for each instance

$$P_j^R(\boldsymbol{x}, \boldsymbol{x}') = \frac{P_j(\boldsymbol{x}', \boldsymbol{x}) p_{0j}(\boldsymbol{x}')}{p_{0j}(\boldsymbol{x})}$$

and

$$\begin{aligned} \pi^\star(\boldsymbol{x}^\star) &= p_{0n}(\boldsymbol{x}_n) \prod_{j=1}^{n} P_j^R(\boldsymbol{x}_{n-j}, \boldsymbol{x}_{n-j+1}) \\ p_0^\star(\boldsymbol{x}^\star) &= p_{00}(\boldsymbol{x}_0) \prod_{j=1}^{n} P_j(\boldsymbol{x}_{j-1}, \boldsymbol{x}_j) \end{aligned}$$

then

$$w^\star(\boldsymbol{x}^\star) = \frac{\pi^\star(\boldsymbol{x}^\star)}{p_0^\star(\boldsymbol{x}^\star)}$$

is the importance sampling weight for the augmented sample $(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ generated from $p_0^\star$. The marginal distribution from $\pi^\star$ of $\boldsymbol{x}_n$ is the **true target** π_n.

Sequential Monte Carlo for State-Space Models

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

$$\begin{aligned} y_t &= g(x_t; \theta) + \epsilon_t \\ x_{t+1} &= h(x_t; \theta) + \varepsilon_t \end{aligned}$$

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

If $\overline{y}_t = (y_1, \ldots, y_t)$, $\overline{x}_t = (x_1, \ldots, x_t)$. we must carry out

- **Filtering:** Compute $p(x_t|\overline{y}_t)$
- **Prediction:** Compute $p(x_{t+1}|\overline{y}_t)$
- **Smoothing:** Compute $p(x_t|\overline{y}_T)$

Filtering: By Bayes theorem

$$\begin{aligned} p(x_t|\overline{y}_t) &= \frac{p(y_t|x_t)p(x_t|\overline{y}_{t-1})}{p(y_t|\overline{y}_{t-1})} \\ &\propto p(y_t|x_t)p(x_t|\overline{y}_{t-1}) \end{aligned}$$

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

Prediction:

$$p(x_{t+1}|\overline{y}_t) = \int p(x_{t+1}|x_t)p(x_t|\overline{y}_t)\, dx_t$$

where $p(x_{t+1}|x_t)$ is again a conditional density derived from the state equation, and $p(x_t|\overline{y}_t)$ 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 $p(x_t|\overline{y}_{t-1})$ obtained at the previous prediction step. Then we can obtain a *particle approximation* to $p(x_t|\overline{y}_t)$ as

$$\widehat{p}(x_t|\overline{y}_t) = \frac{1}{N}\sum_{i=1}^{N} p(y_t|z_t^{(i)})\delta_{z_t^{(i)}}$$

Sequential Monte Carlo for State-Space Models

Note also that the denominator $p(y_t|\overline{y}_{t-1})$ can be approximated by the sum of the weights

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

which is useful in approximating the likelihood by

$$\widehat{L}(\overline{y}_t) = \widehat{p}(y_1)\prod_{t=2}^{T}\widehat{p}(y_t|\overline{y}_{t-1})$$

We can also produce a sample from the particle approximation $\widehat{p}(x_t|\overline{y}_t)$ by performing a *weighted resampling* from the collection $(z_t^{(1)}, \ldots, z_t^{(N)})$ with weights proportional to $p(y_t|z_t^{(i)})$. Denote this resampled vector

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

Sequential Monte Carlo for State-Space Models

For the **prediction** step, the particle approximation to $p(x_{t+1}|\overline{y}_t)$ is obtained using a similar Monte Carlo strategy

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

Also, 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.

Sequential Monte Carlo for State-Space Models

Recap: to perform SIS for $\pi_n(\boldsymbol{x}_n) = g_n(\boldsymbol{x}_n)/Z_n$, use the IS density

$$p_n(\boldsymbol{x}_n) = p_1(x_1) \prod_{j=2}^{n} \widetilde{p}_j(x_j|\boldsymbol{x}_{j-1})$$

with weights given by $w_1(\boldsymbol{x}_1) = \pi_1(x_1)/p_1(x_1)$ and

$$w_j(\boldsymbol{x}_j) = w_{j-1}(\boldsymbol{x}_{j-1}) \frac{\widetilde{\pi}_j(x_j|\boldsymbol{x}_{j-1})}{\widetilde{p}_j(x_j|\boldsymbol{x}_{j-1})} = w_{j-1}(\boldsymbol{x}_{j-1}) \frac{\pi_j(\boldsymbol{x}_j)}{\pi_{j-1}(\boldsymbol{x}_{j-1})\widetilde{p}_j(x_j|\boldsymbol{x}_{j-1})}$$

If the normalizing constants Z_j for $j = 1, \ldots, n$ are unknown, the weights can be replaced by $w_1(\boldsymbol{x}_1) = g_1(x_1)/p_1(x_1)$ and

$$w_j(\boldsymbol{x}_j) = w_{j-1}(\boldsymbol{x}_{j-1}) \frac{g_j(\boldsymbol{x}_j)}{g_{j-1}(\boldsymbol{x}_{j-1})\widetilde{p}_j(x_j|\boldsymbol{x}_{j-1})} = w_{j-1}(\boldsymbol{x}_{j-1})\alpha_j(\boldsymbol{x}_j)$$

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

$$x_j^{(1)}, \ldots, x_j^{(N)} \quad \text{from} \quad \tilde{p}_j(.|\boldsymbol{x}_{j-1}^{(i)})$$

for $i = 1, \ldots, N$. The importance sampling weights are $w_1(x_1^{(i)})$ and, for $j \geqslant 2$,

$$w_j(\boldsymbol{x}_j^{(i)}) = w_{j-1}(\boldsymbol{x}_{j-1}^{(i)})\alpha_j(\boldsymbol{x}_j^{(i)})$$

We then have

$$\widehat{\pi}_n(\boldsymbol{x}) = \sum_{i=1}^{N} W_n^{(i)} \delta_{\boldsymbol{x}_n^{(i)}}(\boldsymbol{x})$$

where

$$W_n^{(i)} = \frac{w_n(\boldsymbol{x}_n^{(i)})}{N\widehat{Z}_n} \qquad \widehat{Z}_n = \frac{1}{N}\sum_{i=1}^{N} w_n(\boldsymbol{x}_n^{(i)})$$

in the usual way.

Sequential Monte Carlo for 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 $\boldsymbol{y}_n = (y_1, \ldots, y_n)$, the posterior of interest is

$$\pi_n(\boldsymbol{x}_n) = p(\boldsymbol{x}_n|\boldsymbol{y}_n) = \frac{p(\boldsymbol{x}_n, \boldsymbol{y}_n)}{p(\boldsymbol{y}_n)} = \frac{g_n(\boldsymbol{x}_n, \boldsymbol{y}_n)}{Z_n(\boldsymbol{y}_n)}$$

We have

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

and

$$Z_n(\boldsymbol{y}_n) = \int g_n(\boldsymbol{x}_n, \boldsymbol{y}_n)\, d\boldsymbol{x}_n$$

Sequential Monte Carlo for State-Space Models

Now, for sequential Monte Carlo, the optimal choice of IS density is

$$\widetilde{p}_j(x_j|\boldsymbol{x}_{j-1})$$

is

$$\widetilde{\pi}_j(x_j|\boldsymbol{x}_{j-1})$$

which in the simple state-space model is given by

$$\begin{aligned}\widetilde{\pi}_j(x_j|\boldsymbol{x}_{j-1}) \equiv p(x_j|\boldsymbol{y}_j, \boldsymbol{x}_{j-1}) &= \frac{p_{Y|X}(y_j|x_j)p_X(x_j|x_{j-1})}{p(y_j|x_{j-1})} \\ &= \frac{p_{Y|X}(y_j|x_j)p_X(x_j|x_{j-1})}{\int p_{Y|X}(y_j|x)p_X(x|x_{j-1})\,dx}\end{aligned}$$

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

$$\widetilde{p}_j(x_j|\boldsymbol{x}_{j-1}) = p(x_j|\boldsymbol{y}_j, \boldsymbol{x}_{j-1}).$$

We might choose instead

$$\widetilde{p}_j(x_j|\boldsymbol{x}_{j-1}) = p(x_j|\boldsymbol{x}_{j-1})$$

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

$$w_j(\boldsymbol{x}_j) = w_{j-1}(\boldsymbol{x}_{j-1}) p_{\mathrm{Y}|\mathrm{X}}(y_j|x_j)$$

Example (Stochastic volatility model)

Suppose

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

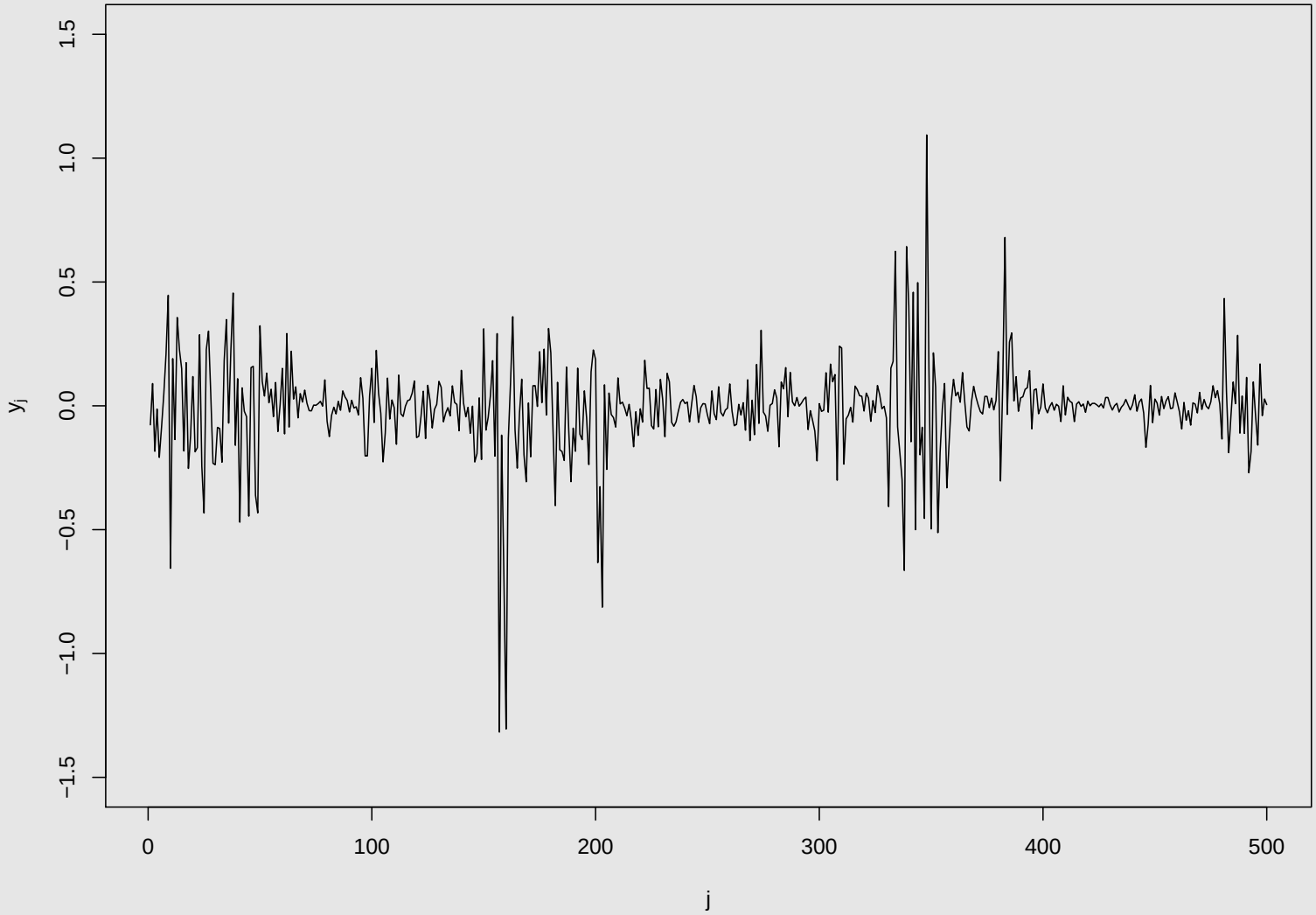
$$\text{State equation} \quad : \quad p_X(x_j|x_{j-1}) \equiv \mathcal{N}(\mu + \phi(x_{j-1} - \mu), \sigma^2)$$

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 $j \geqslant 2$,

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

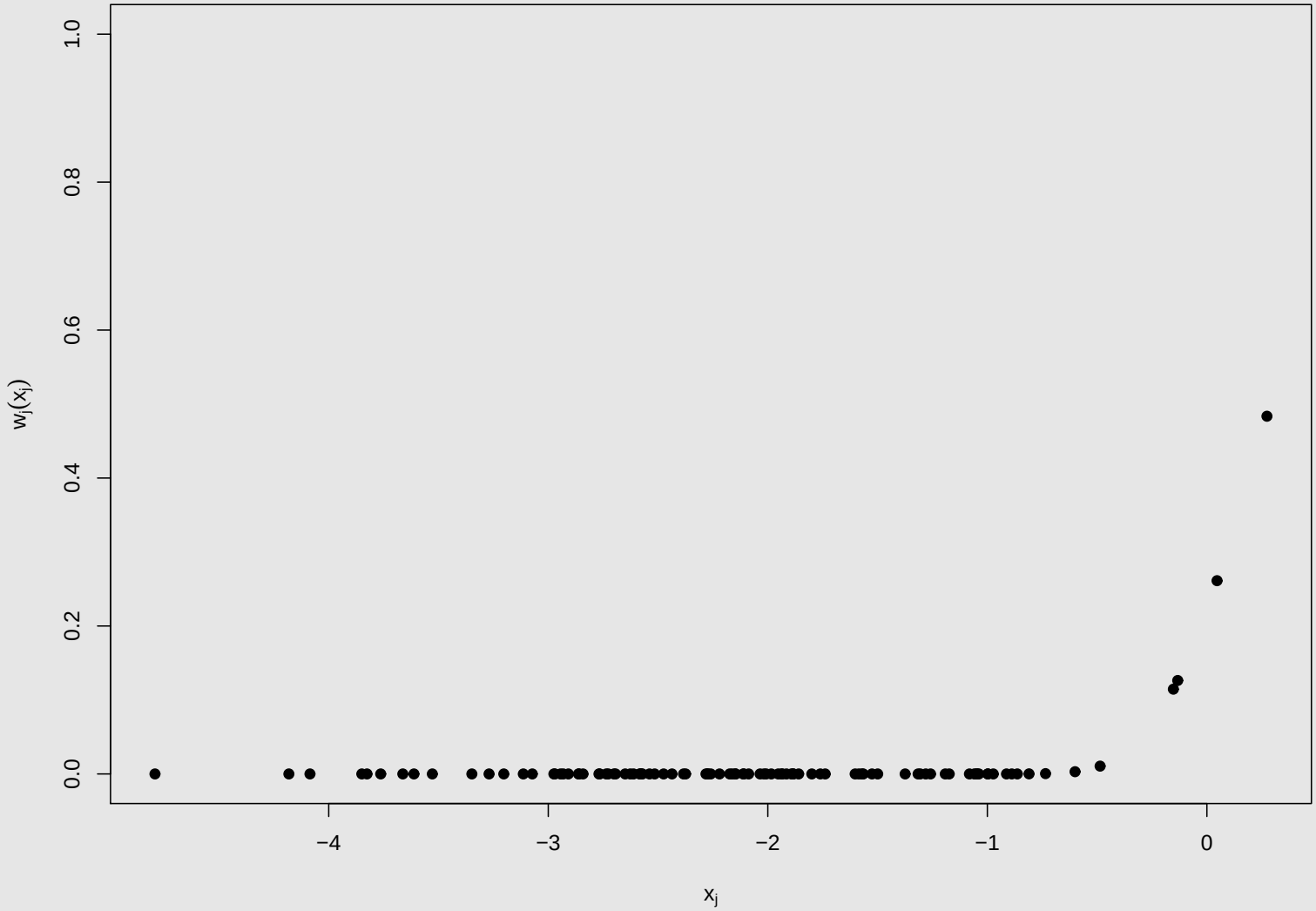
and the recursive weight calculation is based on

$$w_j(\boldsymbol{x}_j) = \prod_{k=1}^{j} p_{Y|X}(y_k|x_k)$$

that is, the likelihood to observation j.

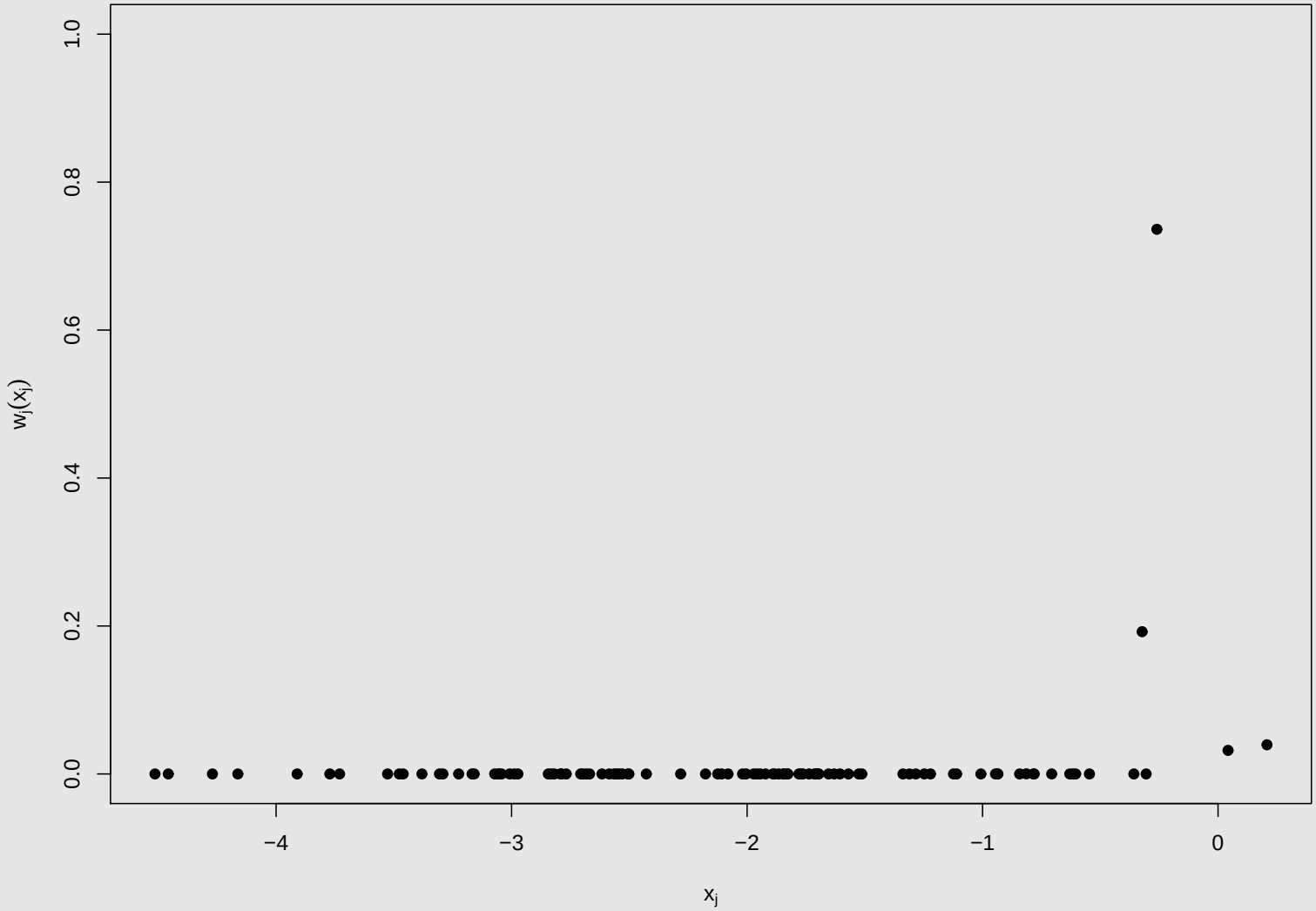
Example (Stochastic volatility model)

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



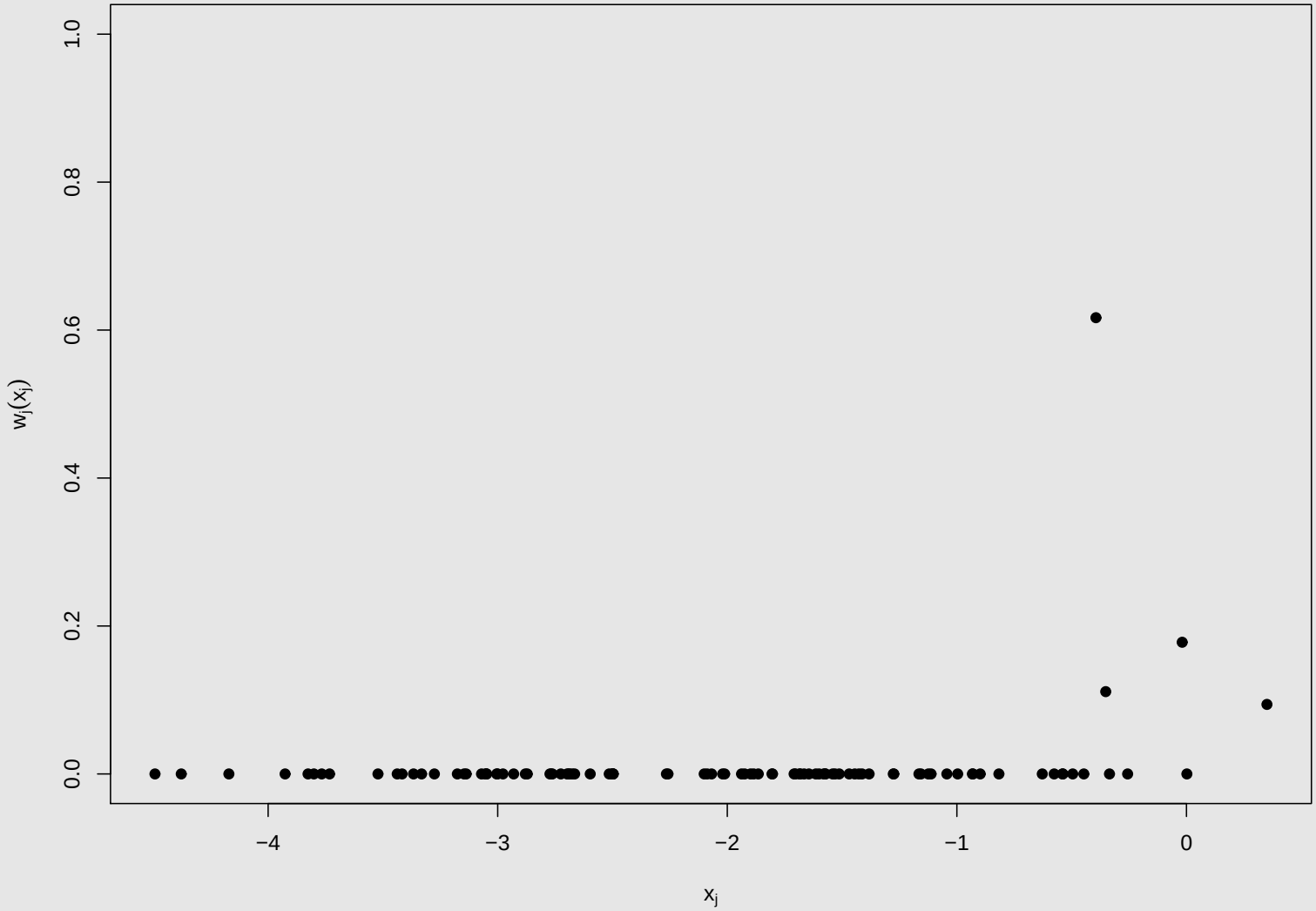
Example (Stochastic volatility model)

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



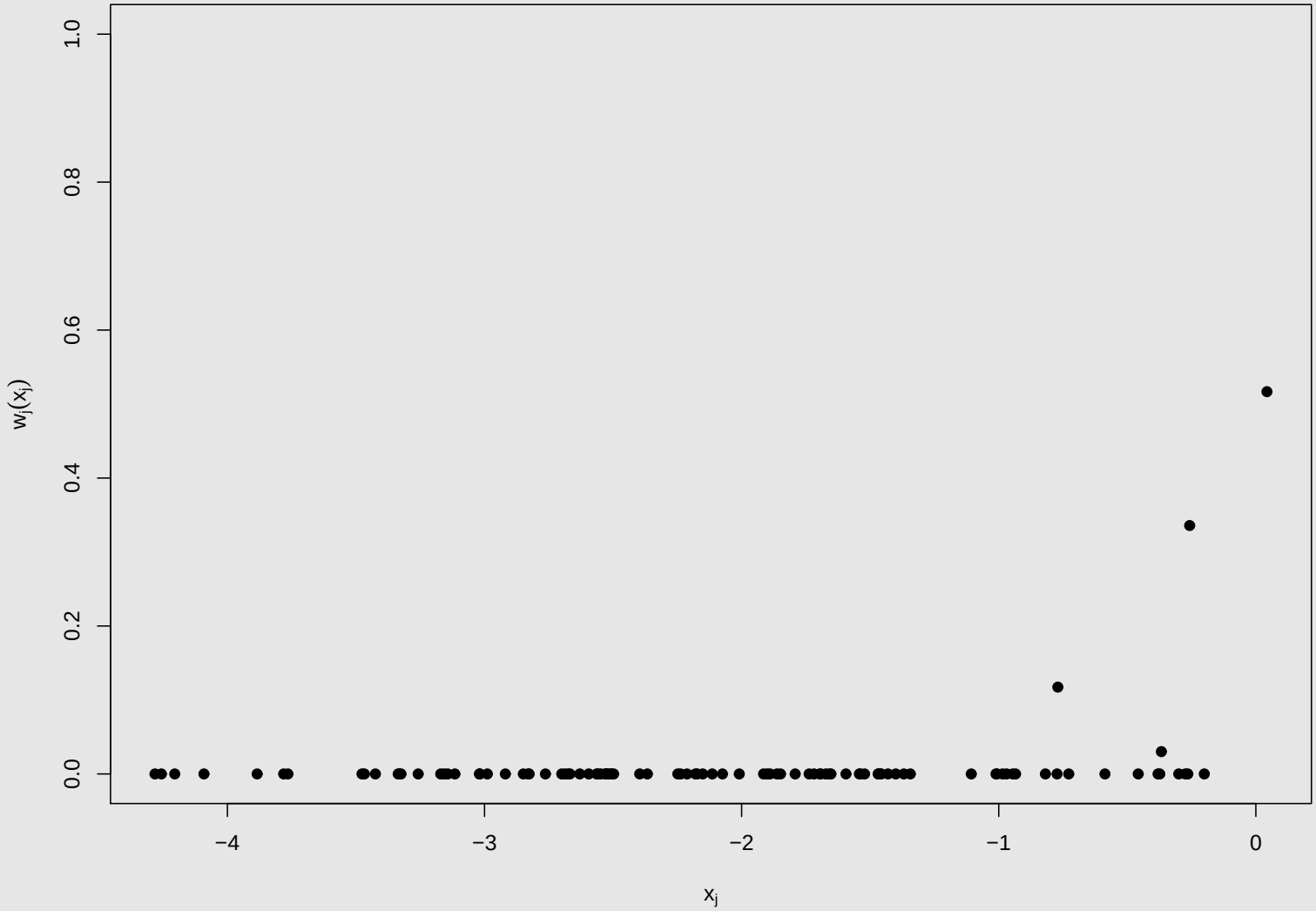
Example (Stochastic volatility model)

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



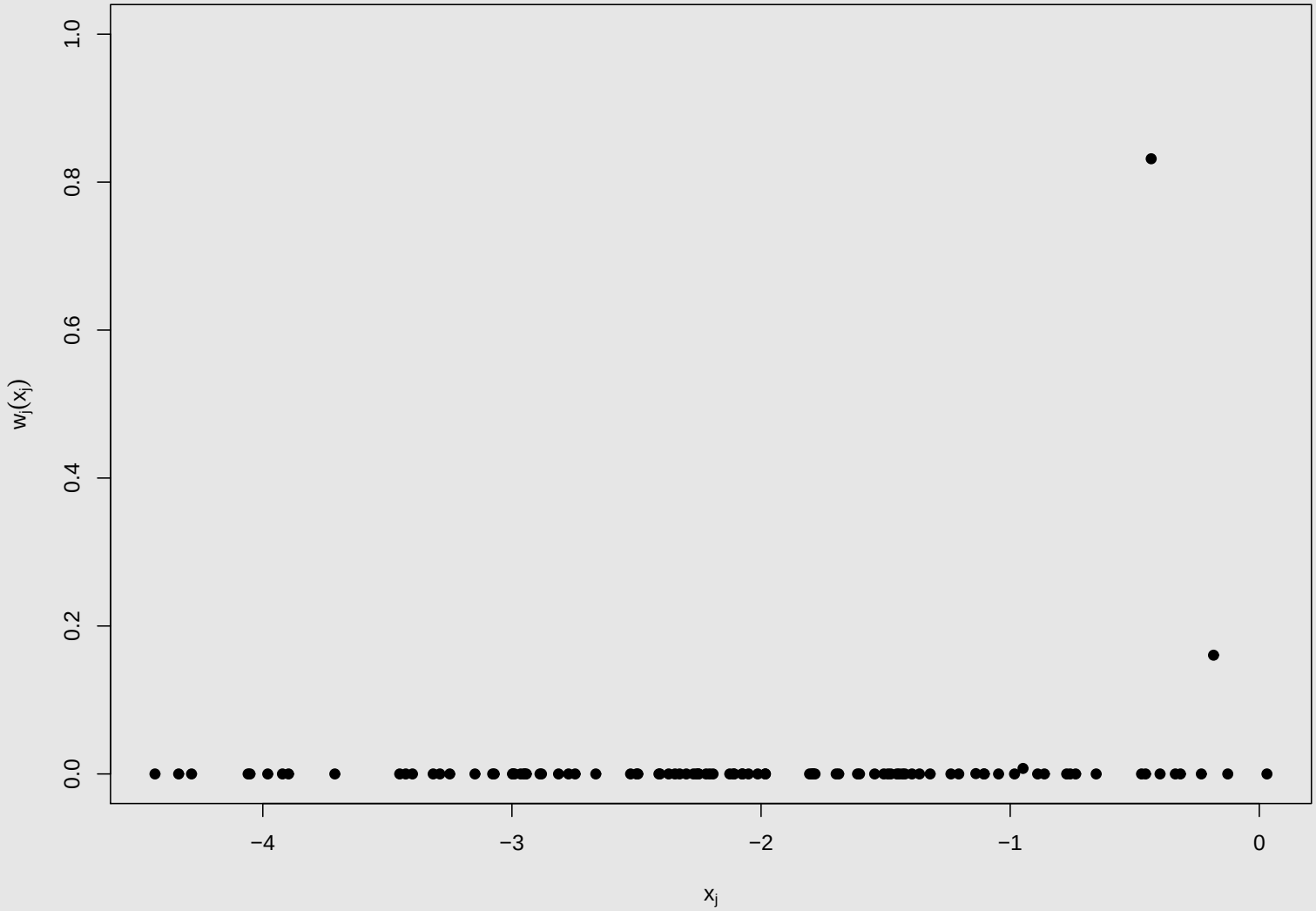
Example (Stochastic volatility model)

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



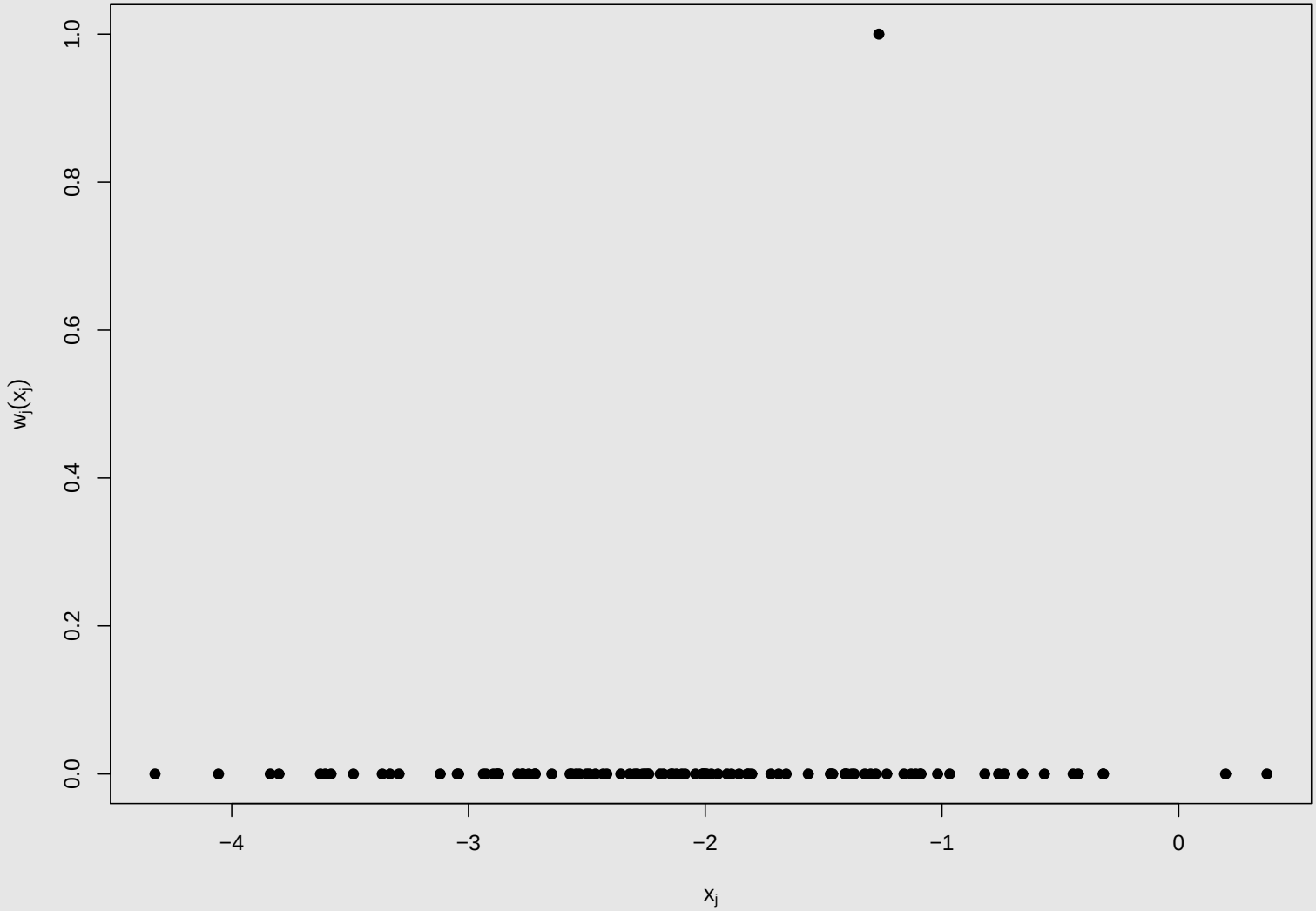
Example (Stochastic volatility model)

$N = 100$ 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 $\widehat{\pi}_n(\boldsymbol{x})$ 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 $\widetilde{p}_j(x_j|\boldsymbol{x}_{j-1})$
- could try a normal approximation of

$$\begin{aligned} p(x_j|\boldsymbol{y}_j, \boldsymbol{x}_{j-1}) &\propto p_{Y|X}(y_j|x_j)p_X(x_j|x_{j-1}) \\ &= \exp\left\{-x_j - \frac{y_j^2}{2e^{2x_j}} - \frac{1}{2\sigma^2}(x_j - \mu_j)^2\right\} \end{aligned}$$

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

Example (Stochastic volatility model)

Here the mode of

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

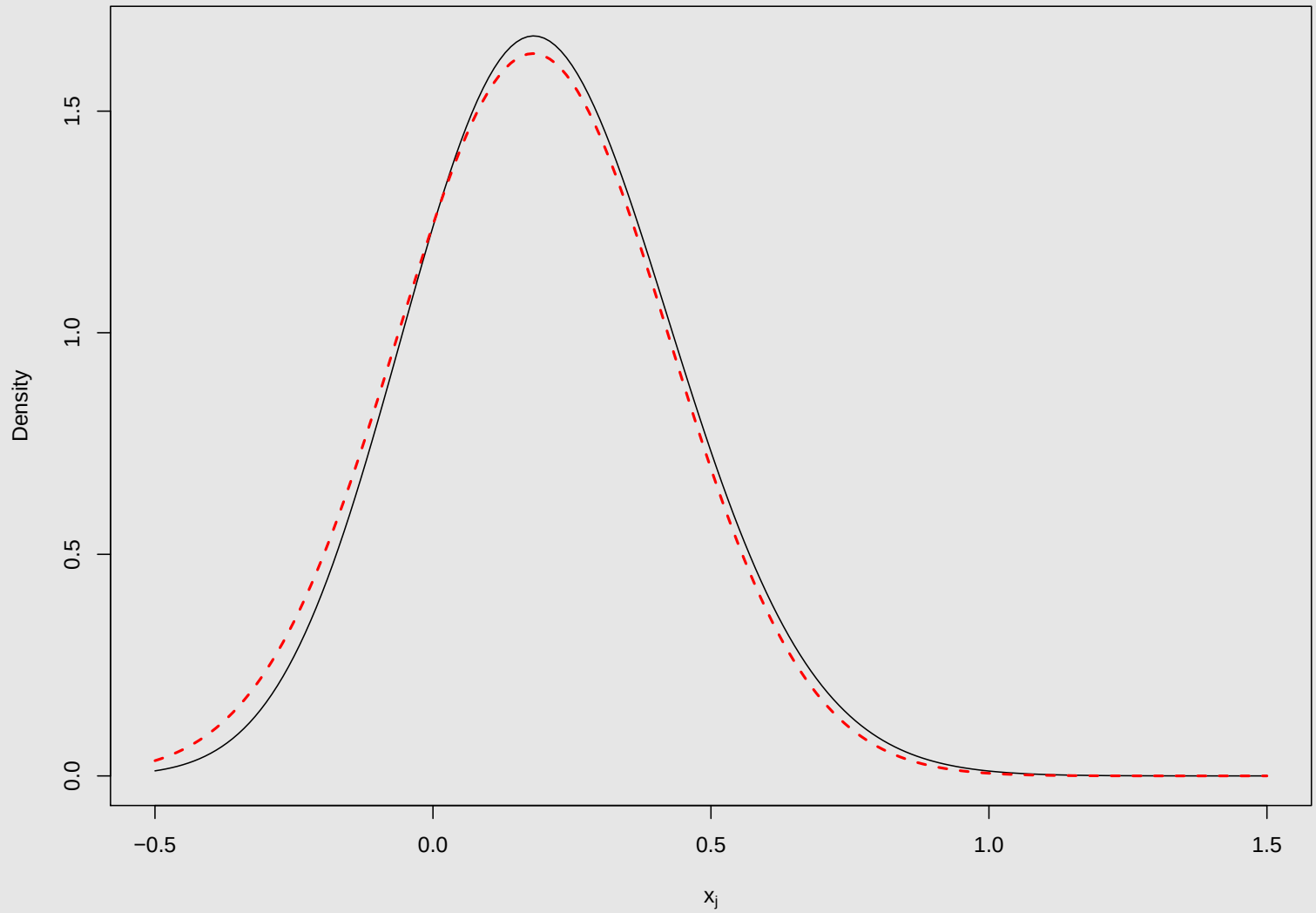
can be computed numerically, and the variance of the normal approximation can be computed by inspecting the curvature at the mode, given by

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

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.

At step j, the Monte Carlo estimate of π_j is

$$\widehat{\pi}_j(\boldsymbol{x}) = \sum_{i=1}^{N} W_j(\boldsymbol{x}_j^{(i)}) \delta_{\boldsymbol{x}_j^{(i)}}(\boldsymbol{x})$$

This discrete distribution on

$$\boldsymbol{x}_j^{(1)}, \ldots, \boldsymbol{x}_j^{(N)}$$

is resampled N times according to the weights

$$W_j(\boldsymbol{x}_j^{(1)}), \ldots, W_j(\boldsymbol{x}_j^{(N)}).$$

Particle Degeneracy

After resampling, the new resampled values replace the

$$\boldsymbol{x}_j^{(1)}, \ldots, \boldsymbol{x}_j^{(N)}$$

collection, and each resampled point carries with it its normalized resampling weight $W_j(\boldsymbol{x}_j^{(i)})$.

Each original value $\boldsymbol{x}_j^{(i)}$ may appear in the resampled collection more than once.

Particle Degeneracy

Let $n_{j1}, \ldots, n_{jN}$ denote the numbers of times that particles $\boldsymbol{x}_j^{(1)}, \ldots, \boldsymbol{x}_j^{(N)}$ are resampled during multinomial sampling. Let

$$\widetilde{\pi}_j(\boldsymbol{x}) = \sum_{i=1}^{N} \frac{n_{ji}}{N} \delta_{\boldsymbol{x}_j^{(i)}}(\boldsymbol{x})$$

Resampling does not disrupt the particle approximation in expectation (that is, it does not introduce bias)

$$\mathbb{E}[\widetilde{\pi}_j(\boldsymbol{x}) | \widehat{\pi}_j(\boldsymbol{x})] = \widehat{\pi}_j(\boldsymbol{x})$$

but it does introduce "noise" - for any suitable h,

$$\mathbb{V}ar_{\widetilde{\pi}_j}[h(\boldsymbol{X}_j)] \geqslant \mathbb{V}ar_{\widehat{\pi}_j}[h(\boldsymbol{X}_j)]$$

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 $0 \leqslant \text{Entropy} \leqslant \log_2 N$

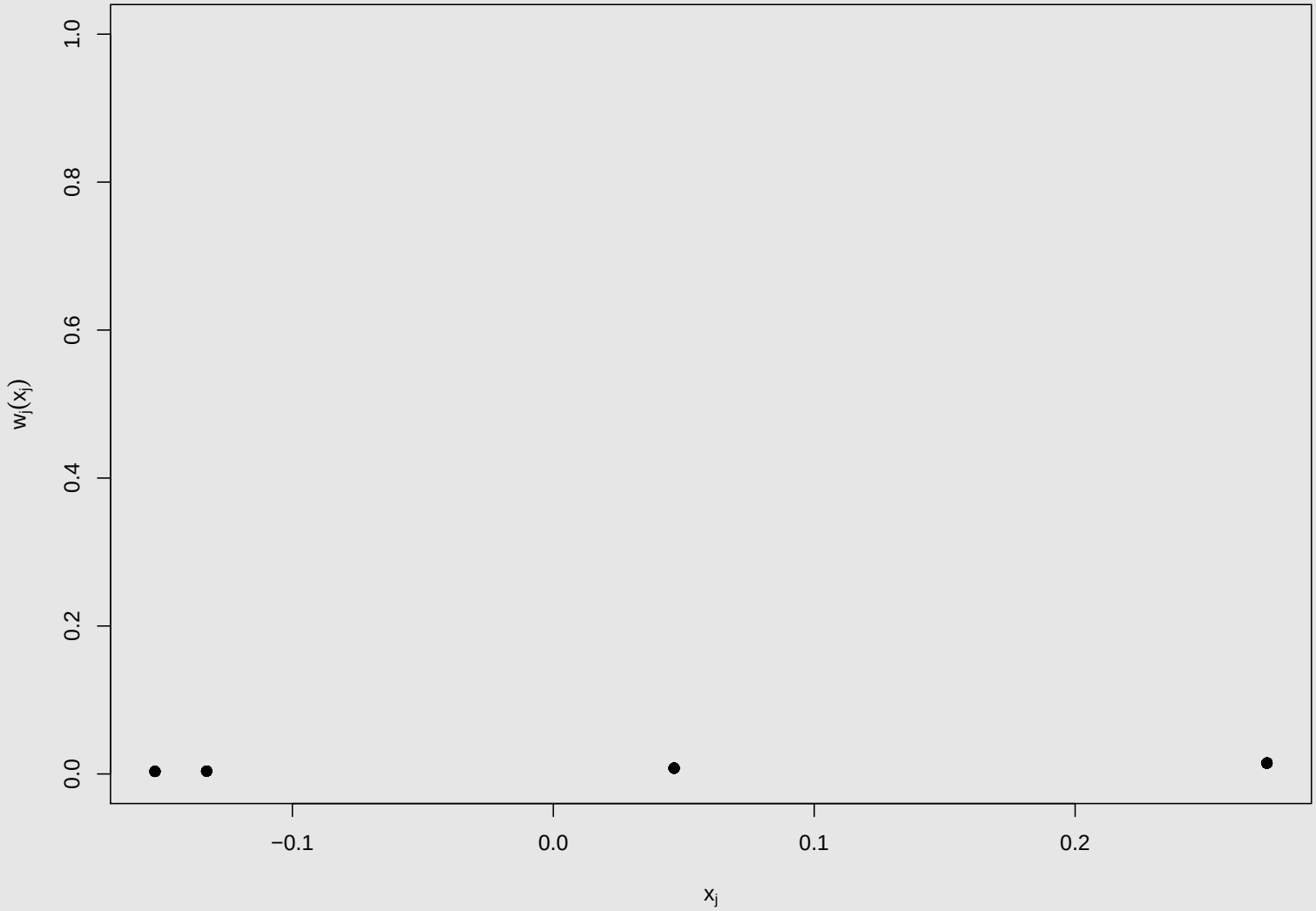
$$\text{Entropy} = 0 \quad \text{if} \quad W_j^{(i)} = 1 \quad \text{for some } i$$

$$\text{Entropy} = \log_2 N \quad \text{if} \quad W_j^{(i)} = \frac{1}{N} \quad \text{for each } i$$

Particle Degeneracy

Example (Stochastic volatility model)

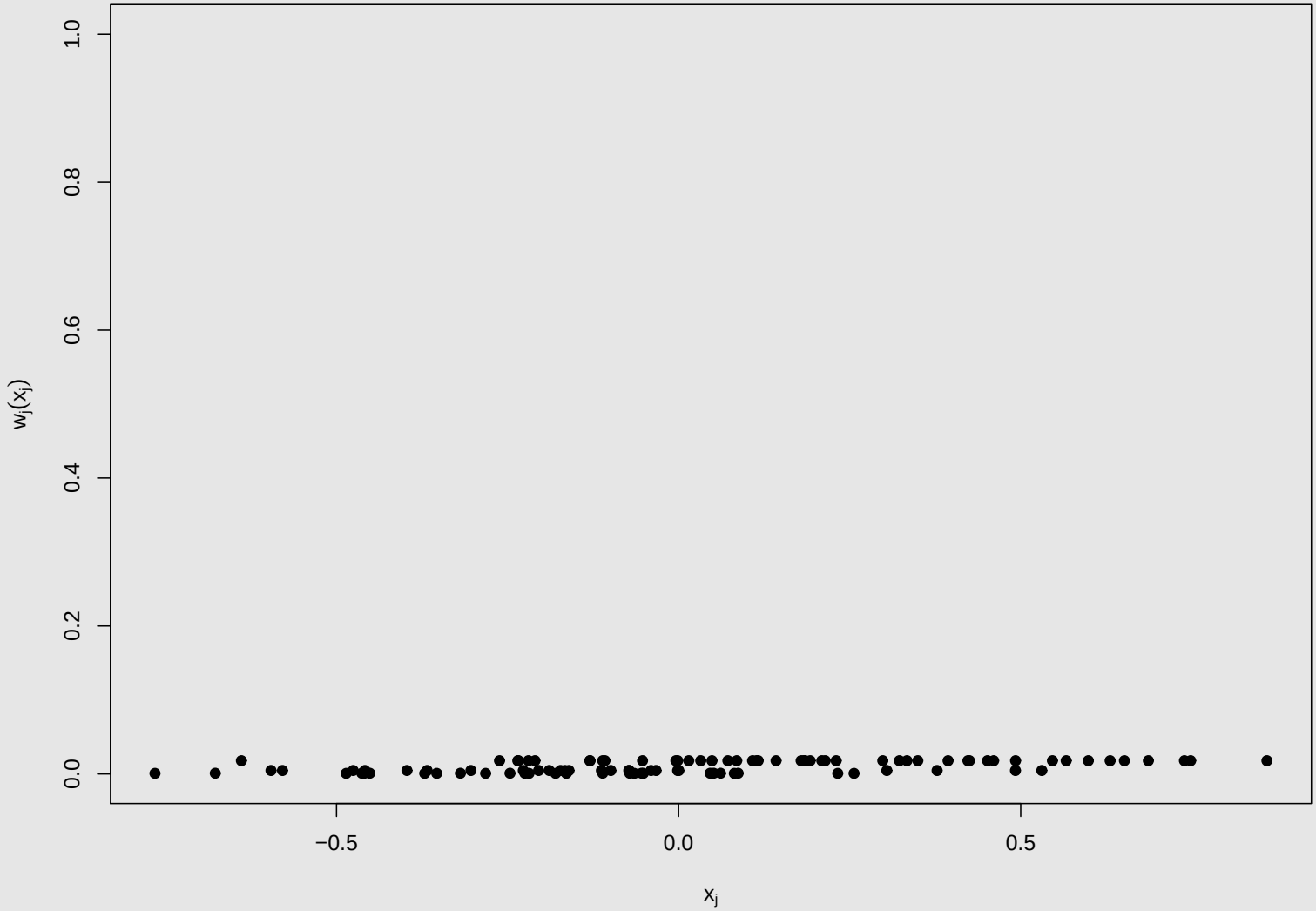
$N = 100$ resampled particles and weights at $j = 1$



Particle Degeneracy

Example (Stochastic volatility model)

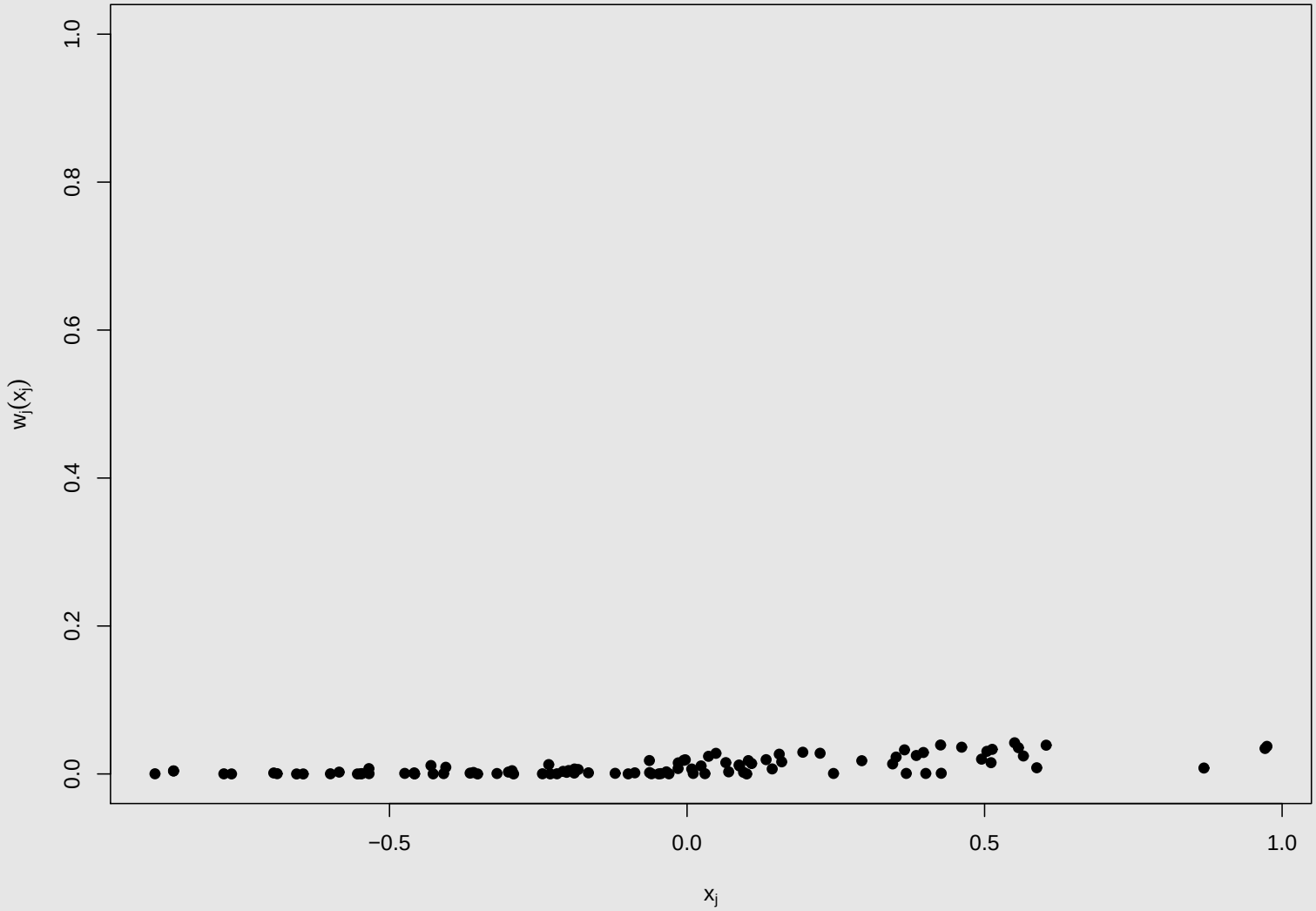
$N = 100$ resampled particles and weights at $j = 2$



Particle Degeneracy

Example (Stochastic volatility model)

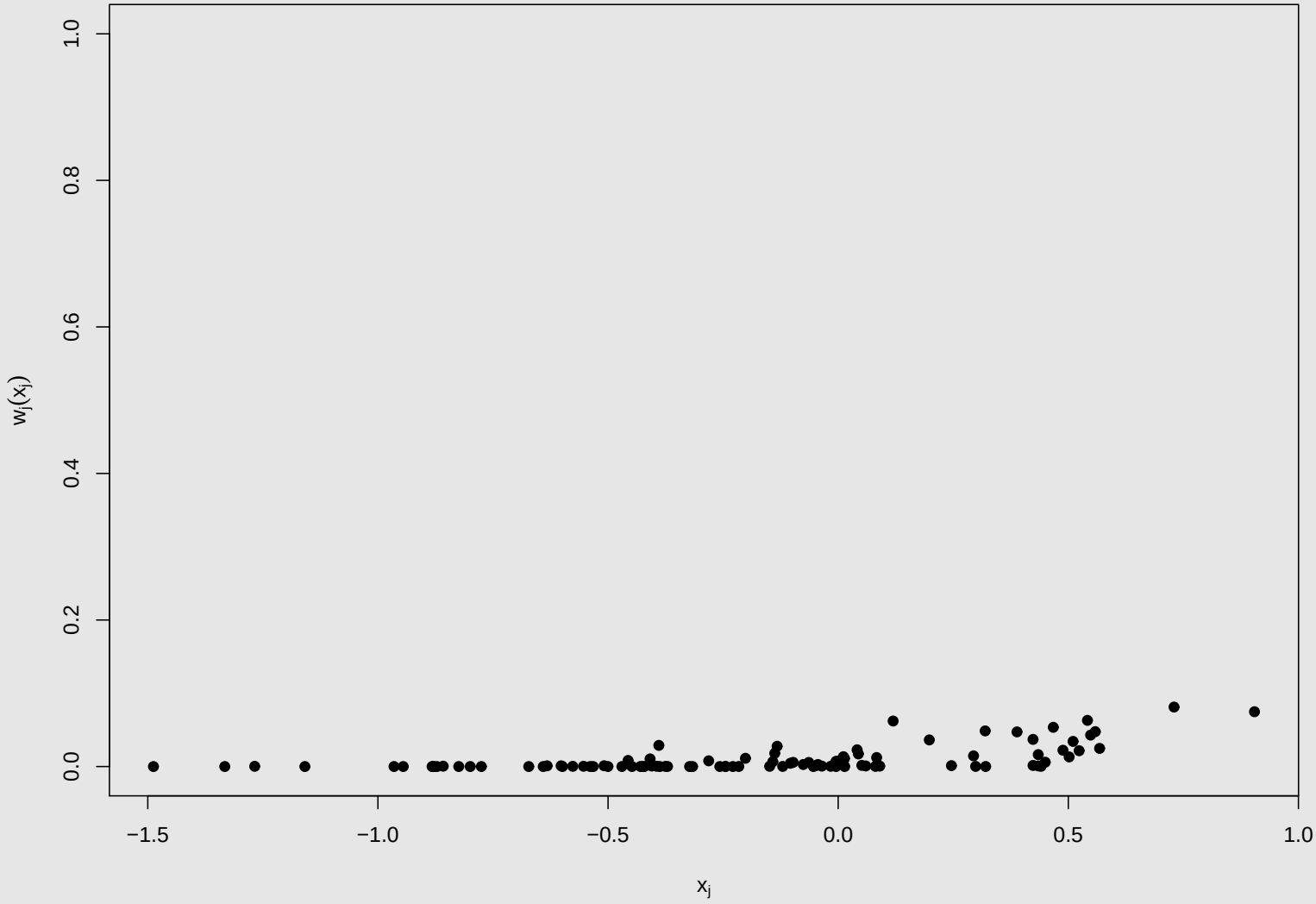
$N = 100$ resampled particles and weights at $j = 3$



Particle Degeneracy

Example (Stochastic volatility model)

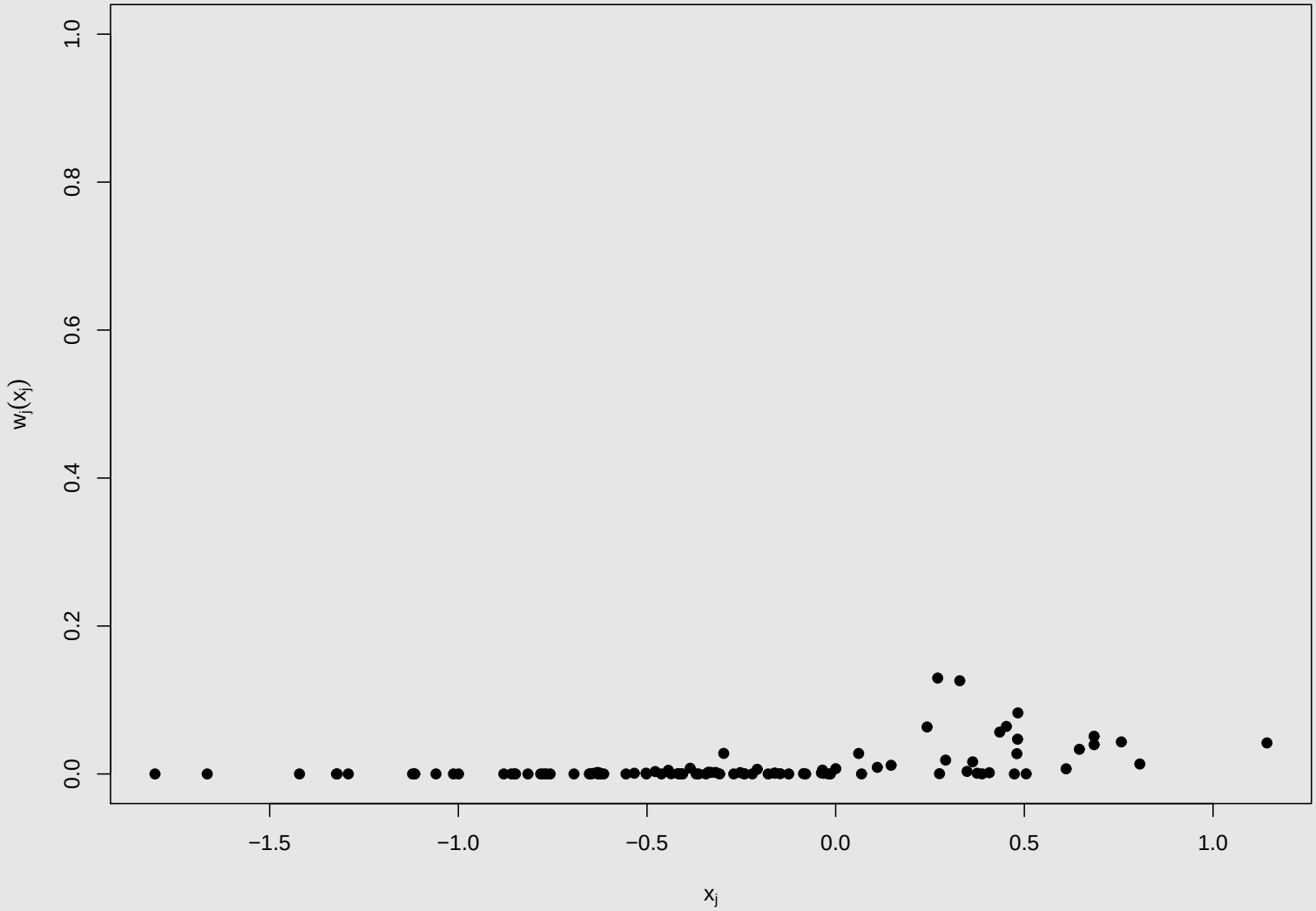
$N = 100$ resampled particles and weights at $j = 4$



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

