MwG, and MCMC diagnostics

Example

Suppose

$$X_1, \ldots, X_N \mid (\lambda_1, \lambda_2, T) \sim \text{Exponential}(\lambda_1 + \lambda_2 T),$$

so the likelihood depends on the combined rate

$$\lambda_1 + \lambda_2 T.$$

We assume independent Gamma-type priors:

$$\lambda_1 \sim \text{Gamma}(\alpha_1, \beta_1), \qquad \lambda_2 \sim \text{Gamma}(\alpha_2, \beta_2),$$

with densities proportional to

$$\pi(\lambda_1) \propto \lambda_1^{\alpha_1 - 1} e^{-\beta_1 \lambda_1}, \qquad \pi(\lambda_2) \propto \lambda_2^{\alpha_2 - 1} e^{-\beta_2 \lambda_2}.$$

Our aim is to investigate the posterior distribution of (λ_1, λ_2) given T and the observed data $\mathbf{x} = (x_1, \ldots, x_N)$.

Bayes' theorem for this model

By Bayes' theorem,

$$\pi(\lambda_1, \lambda_2 \mid T, \mathbf{x}) \propto \pi(\mathbf{x} \mid \lambda_1, \lambda_2, T)\, \pi(\lambda_1)\, \pi(\lambda_2).$$

Bayes' theorem for this model

By Bayes' theorem,

$$\pi(\lambda_1, \lambda_2 \mid T, \mathbf{x}) \propto \pi(\mathbf{x} \mid \lambda_1, \lambda_2, T)\, \pi(\lambda_1)\, \pi(\lambda_2).$$

For exponential data with rate $\lambda_1 + \lambda_2 T$,

$$\pi(\mathbf{x} \mid \lambda_1, \lambda_2, T) \propto (\lambda_1 + \lambda_2 T)^N \exp\left[-(\lambda_1 + \lambda_2 T) \sum_{i=1}^{N} x_i\right].$$

Bayes' theorem for this model

By Bayes' theorem,

$$\pi(\lambda_1, \lambda_2 \mid T, \mathbf{x}) \propto \pi(\mathbf{x} \mid \lambda_1, \lambda_2, T)\, \pi(\lambda_1)\, \pi(\lambda_2).$$

For exponential data with rate $\lambda_1 + \lambda_2 T$,

$$\pi(\mathbf{x} \mid \lambda_1, \lambda_2, T) \propto (\lambda_1 + \lambda_2 T)^N \exp\left[-(\lambda_1 + \lambda_2 T) \sum_{i=1}^{N} x_i\right].$$

Combining with the priors gives

$$\pi(\lambda_1, \lambda_2 \mid T, \mathbf{x}) \propto (\lambda_1 + \lambda_2 T)^N \exp\left[-(\lambda_1 + \lambda_2 T) \sum_{i=1}^{N} x_i\right] \lambda_1^{\alpha_1 - 1} e^{-\beta_1 \lambda_1} \lambda_2^{\alpha_2 - 1} e^{-\beta_2 \lambda_2}.$$

This is the posterior we want to sample from.

Why do we need sampling at all?

In principle, we might want quantities such as

$$\mathbb{E}[\lambda_1 \mid T, \mathbf{x}] = \iint \lambda_1 \, \pi(\lambda_1, \lambda_2 \mid T, \mathbf{x}) \, d\lambda_1 \, d\lambda_2.$$

Why do we need sampling at all?

In principle, we might want quantities such as

$$\mathbb{E}[\lambda_1 \mid T, \mathbf{x}] = \iint \lambda_1 \, \pi(\lambda_1, \lambda_2 \mid T, \mathbf{x}) \, d\lambda_1 \, d\lambda_2.$$

But this requires a potentially nasty two-dimensional integral involving the posterior normalising constant.

Why do we need sampling at all?

In principle, we might want quantities such as

$$\mathbb{E}[\lambda_1 \mid T, \mathbf{x}] = \iint \lambda_1 \, \pi(\lambda_1, \lambda_2 \mid T, \mathbf{x}) \, d\lambda_1 \, d\lambda_2.$$

But this requires a potentially nasty two-dimensional integral involving the posterior normalising constant.

Similarly, posterior variances, credible intervals, marginal densities, and posterior probabilities all require difficult integrals.

Why do we need sampling at all?

In principle, we might want quantities such as

$$\mathbb{E}[\lambda_1 \mid T, \mathbf{x}] = \iint \lambda_1 \, \pi(\lambda_1, \lambda_2 \mid T, \mathbf{x}) \, d\lambda_1 \, d\lambda_2.$$

But this requires a potentially nasty two-dimensional integral involving the posterior normalising constant.

Similarly, posterior variances, credible intervals, marginal densities, and posterior probabilities all require difficult integrals.

MCMC replaces these integrals by simulation:

$$\mathbb{E}[\lambda_1 \mid T, \mathbf{x}] \approx \frac{1}{M} \sum_{m=1}^{M} \lambda_1^{(m)},$$

where $\lambda_1^{(m)}$ are draws from the posterior distribution.

Why do we need sampling at all?

In principle, we might want quantities such as

$$\mathbb{E}[\lambda_1 \mid T, \mathbf{x}] = \iint \lambda_1 \, \pi(\lambda_1, \lambda_2 \mid T, \mathbf{x}) \, d\lambda_1 \, d\lambda_2.$$

But this requires a potentially nasty two-dimensional integral involving the posterior normalising constant.

Similarly, posterior variances, credible intervals, marginal densities, and posterior probabilities all require difficult integrals.

MCMC replaces these integrals by simulation:

$$\mathbb{E}[\lambda_1 \mid T, \mathbf{x}] \approx \frac{1}{M} \sum_{m=1}^{M} \lambda_1^{(m)},$$

where $\lambda_1^{(m)}$ are draws from the posterior distribution.

So the problem becomes:

How do we generate approximate samples from the posterior when it has no closed form?

Full conditional for λ_1

To update λ_1, we look at the posterior and keep only the terms involving λ_1:

$$\pi(\lambda_1 \mid \lambda_2, T, \mathbf{x}) \propto (\lambda_1 + \lambda_2 T)^N \exp(-\lambda_1 \sum_{i=1}^{N} x_i)\, e^{-\beta_1 \lambda_1}\, \lambda_1^{\alpha_1 - 1}.$$

Full conditional for λ_1

To update λ_1, we look at the posterior and keep only the terms involving λ_1:

$$\pi(\lambda_1 \mid \lambda_2, T, \mathbf{x}) \propto (\lambda_1 + \lambda_2 T)^N \exp(-\lambda_1 \sum_{i=1}^{N} x_i)\, e^{-\beta_1 \lambda_1}\, \lambda_1^{\alpha_1 - 1}.$$

Equivalently,

$$\pi(\lambda_1 \mid \lambda_2, T, \mathbf{x}) \propto (\lambda_1 + \lambda_2 T)^N \exp\big[- (\sum_{i=1}^{N} x_i + \beta_1)\lambda_1 \big] \lambda_1^{\alpha_1 - 1}.$$

Full conditional for λ_1

To update λ_1, we look at the posterior and keep only the terms involving λ_1:

$$\pi(\lambda_1 \mid \lambda_2, T, \mathbf{x}) \propto (\lambda_1 + \lambda_2 T)^N \exp(-\lambda_1 \sum_{i=1}^{N} x_i)\, e^{-\beta_1 \lambda_1}\, \lambda_1^{\alpha_1 - 1}.$$

Equivalently,

$$\pi(\lambda_1 \mid \lambda_2, T, \mathbf{x}) \propto (\lambda_1 + \lambda_2 T)^N \exp\big[- (\sum_{i=1}^{N} x_i + \beta_1)\lambda_1 \big] \lambda_1^{\alpha_1 - 1}.$$

Important point:

- this is the *kernel* of the full conditional;
- it is not obviously a standard Gamma, Normal, or other easy distribution.

So we cannot directly use a Gibbs step for λ_1.

Full conditional for λ_2

Similarly, keeping only the terms involving λ_2 gives

$$\pi(\lambda_2 \mid \lambda_1, T, \mathbf{x}) \propto (\lambda_1 + \lambda_2 T)^N \exp\left[-\lambda_2 T \sum_{i=1}^{N} x_i\right] e^{-\beta_2 \lambda_2} \lambda_2^{\alpha_2 - 1}.$$

Full conditional for λ_2

Similarly, keeping only the terms involving λ_2 gives

$$\pi(\lambda_2 \mid \lambda_1, T, \mathbf{x}) \propto (\lambda_1 + \lambda_2 T)^N \exp\left[-\lambda_2 T \sum_{i=1}^{N} x_i\right] e^{-\beta_2 \lambda_2} \lambda_2^{\alpha_2 - 1}.$$

Equivalently,

$$\pi(\lambda_2 \mid \lambda_1, T, \mathbf{x}) \propto (\lambda_1 + \lambda_2 T)^N \exp\big[-(T \sum_{i=1}^{N} x_i + \beta_2)\lambda_2\big] \lambda_2^{\alpha_2 - 1}.$$

Full conditional for λ_2

Similarly, keeping only the terms involving λ_2 gives

$$\pi(\lambda_2 \mid \lambda_1, T, \mathbf{x}) \propto (\lambda_1 + \lambda_2 T)^N \exp\left[-\lambda_2 T \sum_{i=1}^{N} x_i\right] e^{-\beta_2 \lambda_2} \lambda_2^{\alpha_2 - 1}.$$

Equivalently,

$$\pi(\lambda_2 \mid \lambda_1, T, \mathbf{x}) \propto (\lambda_1 + \lambda_2 T)^N \exp\big[-(T \sum_{i=1}^{N} x_i + \beta_2)\lambda_2\big] \lambda_2^{\alpha_2 - 1}.$$

Again, this is not a standard distribution from which we can sample directly.

Full conditional for λ_2

Similarly, keeping only the terms involving λ_2 gives

$$\pi(\lambda_2 \mid \lambda_1, T, \mathbf{x}) \propto (\lambda_1 + \lambda_2 T)^N \exp\left[-\lambda_2 T \sum_{i=1}^{N} x_i\right] e^{-\beta_2 \lambda_2} \lambda_2^{\alpha_2 - 1}.$$

Equivalently,

$$\pi(\lambda_2 \mid \lambda_1, T, \mathbf{x}) \propto (\lambda_1 + \lambda_2 T)^N \exp\big[-(T \sum_{i=1}^{N} x_i + \beta_2)\lambda_2\big] \lambda_2^{\alpha_2 - 1}.$$

Again, this is not a standard distribution from which we can sample directly.
Therefore:

- no Gibbs update is available for λ_1,
- no Gibbs update is available for λ_2.

So we use Metropolis–Hastings updates within a Gibbs-style coordinate update scheme.

What kind of sampler do we need?

This is a classic **Metropolis-within-Gibbs** situation:

- We update one parameter at a time, as in Gibbs sampling.
- But each update is done using a Metropolis–Hastings accept/reject step.

What kind of sampler do we need?

This is a classic **Metropolis-within-Gibbs** situation:

- We update one parameter at a time, as in Gibbs sampling.
- But each update is done using a Metropolis–Hastings accept/reject step.

The idea is:

1. Start at some initial point $(\lambda_1^{(0)}, \lambda_2^{(0)})$.
2. Propose a new value for λ_1, keeping λ_2 fixed.
3. Accept or reject it.
4. Then propose a new value for λ_2, using the updated λ_1.
5. Accept or reject it.
6. Repeat many times.

What kind of sampler do we need?

This is a classic **Metropolis-within-Gibbs** situation:

- We update one parameter at a time, as in Gibbs sampling.
- But each update is done using a Metropolis–Hastings accept/reject step.

The idea is:

1. Start at some initial point $(\lambda_1^{(0)}, \lambda_2^{(0)})$.
2. Propose a new value for λ_1, keeping λ_2 fixed.
3. Accept or reject it.
4. Then propose a new value for λ_2, using the updated λ_1.
5. Accept or reject it.
6. Repeat many times.

This gives a Markov chain

$$(\lambda_1^{(0)}, \lambda_2^{(0)}), (\lambda_1^{(1)}, \lambda_2^{(1)}), (\lambda_1^{(2)}, \lambda_2^{(2)}), \ldots$$

whose stationary distribution is the target posterior.

Random-walk proposals

A simple and very common choice is a random-walk proposal:

$$\lambda_1' \sim N(\lambda_1^{(i-1)}, \sigma_1^2), \qquad \lambda_2' \sim N(\lambda_2^{(i-1)}, \sigma_2^2).$$

Random-walk proposals

A simple and very common choice is a random-walk proposal:

$$\lambda_1' \sim N(\lambda_1^{(i-1)}, \sigma_1^2), \qquad \lambda_2' \sim N(\lambda_2^{(i-1)}, \sigma_2^2).$$

Interpretation:

- we start at the current value,
- add random noise,
- then decide whether to accept the proposed move.

Random-walk proposals

A simple and very common choice is a random-walk proposal:

$$\lambda_1' \sim N(\lambda_1^{(i-1)}, \sigma_1^2), \qquad \lambda_2' \sim N(\lambda_2^{(i-1)}, \sigma_2^2).$$

Interpretation:

- we start at the current value,
- add random noise,
- then decide whether to accept the proposed move.

Advantages:

- easy to implement,
- symmetric proposal density,
- proposal ratio cancels in the MH acceptance probability.

Random-walk proposals

A simple and very common choice is a random-walk proposal:

$$\lambda_1' \sim N(\lambda_1^{(i-1)}, \sigma_1^2), \qquad \lambda_2' \sim N(\lambda_2^{(i-1)}, \sigma_2^2).$$

Interpretation:

- we start at the current value,
- add random noise,
- then decide whether to accept the proposed move.

Advantages:

- easy to implement,
- symmetric proposal density,
- proposal ratio cancels in the MH acceptance probability.

Potential issue:

- if we allow proposals outside the valid support (for example negative λ_1 or λ_2), then those proposals should be rejected immediately.

Algorithm overview

At iteration i:

1. Start from the current state $(\lambda_1^{(i-1)}, \lambda_2^{(i-1)})$.
2. Propose λ_1' and accept/reject it to produce $\lambda_1^{(i)}$.
3. Using this updated $\lambda_1^{(i)}$, propose λ_2' and accept/reject it to produce $\lambda_2^{(i)}$.
4. Repeat for many iterations.

Algorithm overview

At iteration i:

1. Start from the current state $(\lambda_1^{(i-1)}, \lambda_2^{(i-1)})$.
2. Propose λ_1' and accept/reject it to produce $\lambda_1^{(i)}$.
3. Using this updated $\lambda_1^{(i)}$, propose λ_2' and accept/reject it to produce $\lambda_2^{(i)}$.
4. Repeat for many iterations.

So each iteration has two sub-updates:

$$(\lambda_1^{(i-1)}, \lambda_2^{(i-1)}) \to (\lambda_1^{(i)}, \lambda_2^{(i-1)}) \to (\lambda_1^{(i)}, \lambda_2^{(i)}).$$

Algorithm overview

At iteration i:

1. Start from the current state $(\lambda_1^{(i-1)}, \lambda_2^{(i-1)})$.
2. Propose λ_1' and accept/reject it to produce $\lambda_1^{(i)}$.
3. Using this updated $\lambda_1^{(i)}$, propose λ_2' and accept/reject it to produce $\lambda_2^{(i)}$.
4. Repeat for many iterations.

So each iteration has two sub-updates:

$$(\lambda_1^{(i-1)}, \lambda_2^{(i-1)}) \to (\lambda_1^{(i)}, \lambda_2^{(i-1)}) \to (\lambda_1^{(i)}, \lambda_2^{(i)}).$$

This ordering matters when writing down the acceptance probabilities:

- the λ_1 update uses $\lambda_2^{(i-1)}$;
- the λ_2 update uses $\lambda_1^{(i)}$, not $\lambda_1^{(i-1)}$.

Metropolis-within-Gibbs algorithm

1. Choose starting values $\lambda_1^{(0)}$ and $\lambda_2^{(0)}$.
2. For $i = 1, 2, \ldots, M$:
   1. Propose
   $$\lambda_1' \sim N(\lambda_1^{(i-1)}, \sigma_1^2).$$
   2. Accept λ_1' with probability $p_{\text{acc},1}$; otherwise set
   $$\lambda_1^{(i)} = \lambda_1^{(i-1)}.$$
   3. Propose
   $$\lambda_2' \sim N(\lambda_2^{(i-1)}, \sigma_2^2).$$
   4. Accept λ_2' with probability $p_{\text{acc},2}$; otherwise set
   $$\lambda_2^{(i)} = \lambda_2^{(i-1)}.$$

Metropolis-within-Gibbs algorithm

1. Choose starting values $\lambda_1^{(0)}$ and $\lambda_2^{(0)}$.
2. For $i = 1, 2, \ldots, M$:
   1. Propose
   $$\lambda_1' \sim N(\lambda_1^{(i-1)}, \sigma_1^2).$$
   2. Accept λ_1' with probability $p_{\text{acc},1}$; otherwise set
   $$\lambda_1^{(i)} = \lambda_1^{(i-1)}.$$
   3. Propose
   $$\lambda_2' \sim N(\lambda_2^{(i-1)}, \sigma_2^2).$$
   4. Accept λ_2' with probability $p_{\text{acc},2}$; otherwise set
   $$\lambda_2^{(i)} = \lambda_2^{(i-1)}.$$

In practice:

- store the chain,
- discard burn-in,

Acceptance probability for the λ_1 update

Because the proposal is symmetric,

$$q(\lambda_1' \mid \lambda_1^{(i-1)}) = q(\lambda_1^{(i-1)} \mid \lambda_1'),$$

so the proposal ratio cancels.

Acceptance probability for the λ_1 update

Because the proposal is symmetric,

$$q(\lambda_1' \mid \lambda_1^{(i-1)}) = q(\lambda_1^{(i-1)} \mid \lambda_1'),$$

so the proposal ratio cancels.
Hence

$$p_{\text{acc},1} = \min\left\{1, \frac{\pi(\lambda_1' \mid \lambda_2^{(i-1)}, T, \mathbf{x})}{\pi(\lambda_1^{(i-1)} \mid \lambda_2^{(i-1)}, T, \mathbf{x})}\right\}.$$

Acceptance probability for the λ_1 update

Because the proposal is symmetric,

$$q(\lambda_1' \mid \lambda_1^{(i-1)}) = q(\lambda_1^{(i-1)} \mid \lambda_1'),$$

so the proposal ratio cancels.
Hence

$$p_{\text{acc},1} = \min\left\{1, \frac{\pi(\lambda_1' \mid \lambda_2^{(i-1)}, T, \mathbf{x})}{\pi(\lambda_1^{(i-1)} \mid \lambda_2^{(i-1)}, T, \mathbf{x})}\right\}.$$

Substituting the full conditional kernel:

$$p_{\text{acc},1} = \min\left\{1, \frac{(\lambda_1' + \lambda_2^{(i-1)}T)^N \exp[-\lambda_1' \sum_{i=1}^N x_i] e^{-\beta_1 \lambda_1'} (\lambda_1')^{\alpha_1 - 1}}{(\lambda_1^{(i-1)} + \lambda_2^{(i-1)}T)^N \exp[-\lambda_1^{(i-1)} \sum_{i=1}^N x_i] e^{-\beta_1 \lambda_1^{(i-1)}} (\lambda_1^{(i-1)})^{\alpha_1 - 1}}\right\}.$$

Simplifying the λ_1 acceptance probability

We can collect the exponential terms:

$$\exp[-\lambda_1' \sum_{i=1}^{N} x_i] e^{-\beta_1 \lambda_1'} = \exp[-(\sum_{i=1}^{N} x_i + \beta_1)\lambda_1'].$$

Similarly in the denominator,

$$\exp[-\lambda_1^{(i-1)} \sum_{i=1}^{N} x_i] e^{-\beta_1 \lambda_1^{(i-1)}} = \exp[-(\sum_{i=1}^{N} x_i + \beta_1)\lambda_1^{(i-1)}].$$

Simplifying the λ_1 acceptance probability

We can collect the exponential terms:

$$\exp[-\lambda_1' \sum_{i=1}^{N} x_i] e^{-\beta_1 \lambda_1'} = \exp[-(\sum_{i=1}^{N} x_i + \beta_1)\lambda_1'].$$

Similarly in the denominator,

$$\exp[-\lambda_1^{(i-1)} \sum_{i=1}^{N} x_i] e^{-\beta_1 \lambda_1^{(i-1)}} = \exp[-(\sum_{i=1}^{N} x_i + \beta_1)\lambda_1^{(i-1)}].$$

So a cleaner version is

$$p_{\text{acc},1} = \min \left\{ 1, \frac{(\lambda_1' + \lambda_2^{(i-1)} T)^N \exp[-(\sum_{i=1}^{N} x_i + \beta_1)\lambda_1'](\lambda_1')^{\alpha_1 - 1}}{(\lambda_1^{(i-1)} + \lambda_2^{(i-1)} T)^N \exp[-(\sum_{i=1}^{N} x_i + \beta_1)\lambda_1^{(i-1)}](\lambda_1^{(i-1)})^{\alpha_1 - 1}} \right\}.$$

Simplifying the λ_1 acceptance probability

We can collect the exponential terms:

$$\exp[-\lambda_1' \sum_{i=1}^{N} x_i] e^{-\beta_1 \lambda_1'} = \exp[-(\sum_{i=1}^{N} x_i + \beta_1)\lambda_1'].$$

Similarly in the denominator,

$$\exp[-\lambda_1^{(i-1)} \sum_{i=1}^{N} x_i] e^{-\beta_1 \lambda_1^{(i-1)}} = \exp[-(\sum_{i=1}^{N} x_i + \beta_1)\lambda_1^{(i-1)}].$$

So a cleaner version is

$$p_{\text{acc},1} = \min \left\{ 1, \frac{(\lambda_1' + \lambda_2^{(i-1)} T)^N \exp[-(\sum_{i=1}^{N} x_i + \beta_1)\lambda_1'](\lambda_1')^{\alpha_1 - 1}}{(\lambda_1^{(i-1)} + \lambda_2^{(i-1)} T)^N \exp[-(\sum_{i=1}^{N} x_i + \beta_1)\lambda_1^{(i-1)}](\lambda_1^{(i-1)})^{\alpha_1 - 1}} \right\}.$$

Key message:

- we only need terms that depend on λ_1;
- everything else cancels.

Acceptance probability for the λ_2 update

Now we update λ_2 *after* updating λ_1.

Acceptance probability for the λ_2 update

Now we update λ_2 *after* updating λ_1.
Therefore the correct acceptance probability is

$$p_{\text{acc},2} = \min\left\{1, \frac{\pi(\lambda_2' \mid \lambda_1^{(i)}, T, \mathbf{x})}{\pi(\lambda_2^{(i-1)} \mid \lambda_1^{(i)}, T, \mathbf{x})}\right\}.$$

Acceptance probability for the λ_2 update

Now we update λ_2 *after* updating λ_1.
Therefore the correct acceptance probability is

$$p_{\text{acc},2} = \min\left\{1, \frac{\pi(\lambda_2' \mid \lambda_1^{(i)}, T, \mathbf{x})}{\pi(\lambda_2^{(i-1)} \mid \lambda_1^{(i)}, T, \mathbf{x})}\right\}.$$

Substituting the full conditional kernel:

$$p_{\text{acc},2} = \min\left\{1, \frac{(\lambda_1^{(i)} + \lambda_2' T)^N \exp[-\lambda_2' T \sum_{i=1}^N x_i] e^{-\beta_2 \lambda_2'} (\lambda_2')^{\alpha_2 - 1}}{(\lambda_1^{(i)} + \lambda_2^{(i-1)} T)^N \exp[-\lambda_2^{(i-1)} T \sum_{i=1}^N x_i] e^{-\beta_2 \lambda_2^{(i-1)}} (\lambda_2^{(i-1)})^{\alpha_2 - 1}}\right\}.$$

Simplifying the λ_2 acceptance probability

Collecting exponential terms again,

$$\exp[-\lambda_2' T \sum_{i=1}^{N} x_i] e^{-\beta_2 \lambda_2'} = \exp[-(T \sum_{i=1}^{N} x_i + \beta_2)\lambda_2'].$$

Simplifying the λ_2 acceptance probability

Collecting exponential terms again,

$$\exp[-\lambda_2' T \sum_{i=1}^{N} x_i] e^{-\beta_2 \lambda_2'} = \exp[-(T \sum_{i=1}^{N} x_i + \beta_2)\lambda_2'].$$

Hence

$$p_{\text{acc},2} = \min \left\{ 1, \frac{(\lambda_1^{(i)} + \lambda_2' T)^N \exp[-(T \sum_{i=1}^{N} x_i + \beta_2)\lambda_2'](\lambda_2')^{\alpha_2 - 1}}{(\lambda_1^{(i)} + \lambda_2^{(i-1)} T)^N \exp[-(T \sum_{i=1}^{N} x_i + \beta_2)\lambda_2^{(i-1)}](\lambda_2^{(i-1)})^{\alpha_2 - 1}} \right\}.$$

Simplifying the λ_2 acceptance probability

Collecting exponential terms again,

$$\exp[-\lambda_2' T \sum_{i=1}^{N} x_i] e^{-\beta_2 \lambda_2'} = \exp[-(T \sum_{i=1}^{N} x_i + \beta_2)\lambda_2'].$$

Hence

$$p_{\text{acc},2} = \min \left\{ 1, \frac{(\lambda_1^{(i)} + \lambda_2' T)^N \exp[-(T \sum_{i=1}^{N} x_i + \beta_2)\lambda_2'](\lambda_2')^{\alpha_2 - 1}}{(\lambda_1^{(i)} + \lambda_2^{(i-1)} T)^N \exp[-(T \sum_{i=1}^{N} x_i + \beta_2)\lambda_2^{(i-1)}](\lambda_2^{(i-1)})^{\alpha_2 - 1}} \right\}.$$

Notice the important bookkeeping detail:

$$\lambda_1^{(i)} \text{ appears here, not } \lambda_1^{(i-1)}.$$

That is because the λ_1 move has already been completed inside iteration i.

How acceptance works in practice

At each Metropolis step:

1. Compute the acceptance probability p_{acc}.
2. Draw $U \sim U(0, 1)$.
3. If $U < p_{\text{acc}}$, accept the proposed value.
4. Otherwise reject and keep the current value.

How acceptance works in practice

At each Metropolis step:

1. Compute the acceptance probability p_{acc}.
2. Draw $U \sim U(0,1)$.
3. If $U < p_{\text{acc}}$, accept the proposed value.
4. Otherwise reject and keep the current value.

So for the λ_1 move:

$$\lambda_1^{(i)} = \begin{cases} \lambda_1', & U_1 < p_{\text{acc},1}, \\ \lambda_1^{(i-1)}, & \text{otherwise.} \end{cases}$$

How acceptance works in practice

At each Metropolis step:

1. Compute the acceptance probability p_{acc}.
2. Draw $U \sim U(0,1)$.
3. If $U < p_{\text{acc}}$, accept the proposed value.
4. Otherwise reject and keep the current value.

So for the λ_1 move:

$$\lambda_1^{(i)} = \begin{cases} \lambda_1', & U_1 < p_{\text{acc},1}, \\ \lambda_1^{(i-1)}, & \text{otherwise.} \end{cases}$$

And for the λ_2 move:

$$\lambda_2^{(i)} = \begin{cases} \lambda_2', & U_2 < p_{\text{acc},2}, \\ \lambda_2^{(i-1)}, & \text{otherwise.} \end{cases}$$

How acceptance works in practice

At each Metropolis step:

1. Compute the acceptance probability p_{acc}.
2. Draw $U \sim U(0,1)$.
3. If $U < p_{\text{acc}}$, accept the proposed value.
4. Otherwise reject and keep the current value.

So for the λ_1 move:

$$\lambda_1^{(i)} = \begin{cases} \lambda_1', & U_1 < p_{\text{acc},1}, \\ \lambda_1^{(i-1)}, & \text{otherwise.} \end{cases}$$

And for the λ_2 move:

$$\lambda_2^{(i)} = \begin{cases} \lambda_2', & U_2 < p_{\text{acc},2}, \\ \lambda_2^{(i-1)}, & \text{otherwise.} \end{cases}$$

If a proposal is rejected, the chain does not move in that coordinate. That is a normal feature of Metropolis–Hastings.

Why can we use the full conditionals instead of the full posterior?

A common source of confusion is:

Should the MH ratio use the full posterior or the full conditional?

Why can we use the full conditionals instead of the full posterior?

A common source of confusion is:

Should the MH ratio use the full posterior or the full conditional?

Answer: either viewpoint is fine, because for a coordinate-wise update the terms not involving that coordinate cancel.

Why can we use the full conditionals instead of the full posterior?

A common source of confusion is:

Should the MH ratio use the full posterior or the full conditional?

Answer: either viewpoint is fine, because for a coordinate-wise update the terms not involving that coordinate cancel.

For example, in the λ_1 update:

$$\frac{\pi(\lambda_1', \lambda_2^{(i-1)} \mid T, \mathbf{x})}{\pi(\lambda_1^{(i-1)}, \lambda_2^{(i-1)} \mid T, \mathbf{x})}$$

contains many terms involving $\lambda_2^{(i-1)}$ only.

Why can we use the full conditionals instead of the full posterior?

A common source of confusion is:

Should the MH ratio use the full posterior or the full conditional?

Answer: either viewpoint is fine, because for a coordinate-wise update the terms not involving that coordinate cancel.

For example, in the λ_1 update:

$$\frac{\pi(\lambda_1', \lambda_2^{(i-1)} \mid T, \mathbf{x})}{\pi(\lambda_1^{(i-1)}, \lambda_2^{(i-1)} \mid T, \mathbf{x})}$$

contains many terms involving $\lambda_2^{(i-1)}$ only.

But $\lambda_2^{(i-1)}$ is fixed during this step, so those terms are identical in numerator and denominator and cancel.

Why can we use the full conditionals instead of the full posterior?

A common source of confusion is:

Should the MH ratio use the full posterior or the full conditional?

Answer: either viewpoint is fine, because for a coordinate-wise update the terms not involving that coordinate cancel.

For example, in the λ_1 update:

$$\frac{\pi(\lambda_1', \lambda_2^{(i-1)} \mid T, \mathbf{x})}{\pi(\lambda_1^{(i-1)}, \lambda_2^{(i-1)} \mid T, \mathbf{x})}$$

contains many terms involving $\lambda_2^{(i-1)}$ only.

But $\lambda_2^{(i-1)}$ is fixed during this step, so those terms are identical in numerator and denominator and cancel.

Therefore:

$$\frac{\pi(\lambda_1', \lambda_2^{(i-1)} \mid T, \mathbf{x})}{\pi(\lambda_1^{(i-1)}, \lambda_2^{(i-1)} \mid T, \mathbf{x})} = \frac{\pi(\lambda_1' \mid \lambda_2^{(i-1)}, T, \mathbf{x})}{\pi(\lambda_1^{(i-1)} \mid \lambda_2^{(i-1)}, T, \mathbf{x})}.$$

So using the full conditional kernel is just a convenient simplification.

There is not a unique correct implementation

The algorithm we wrote down is a valid MCMC algorithm, but it is not the only one.

There is not a unique correct implementation

The algorithm we wrote down is a valid MCMC algorithm, but it is not the only one.
For example, we could:

- update λ_2 before λ_1;
- use different proposal variances for the two coordinates;
- use Uniform random-walk proposals instead of Normal ones;
- use a more sophisticated adaptive proposal;
- transform variables to enforce positivity.

There is not a unique correct implementation

The algorithm we wrote down is a valid MCMC algorithm, but it is not the only one.
For example, we could:

- update λ_2 before λ_1;
- use different proposal variances for the two coordinates;
- use Uniform random-walk proposals instead of Normal ones;
- use a more sophisticated adaptive proposal;
- transform variables to enforce positivity.

What changes?

- the order of updates changes which parameter values appear in each acceptance ratio;
- if the proposal is no longer symmetric, the proposal ratio must be included:

$$\frac{q(\text{current} \mid \text{proposed})}{q(\text{proposed} \mid \text{current})}.$$

There is not a unique correct implementation

The algorithm we wrote down is a valid MCMC algorithm, but it is not the only one.
For example, we could:

- update λ_2 before λ_1;
- use different proposal variances for the two coordinates;
- use Uniform random-walk proposals instead of Normal ones;
- use a more sophisticated adaptive proposal;
- transform variables to enforce positivity.

What changes?

- the order of updates changes which parameter values appear in each acceptance ratio;
- if the proposal is no longer symmetric, the proposal ratio must be included:

$$\frac{q(\text{current} \mid \text{proposed})}{q(\text{proposed} \mid \text{current})}.$$

What does not change?

- the target posterior distribution,
- the basic accept/reject logic,

Two practical questions

Once the algorithm is written down, two practical questions immediately arise:

1. Where should we initialise the chain?
2. How should we choose the proposal step sizes σ_1^2 and σ_2^2?

Two practical questions

Once the algorithm is written down, two practical questions immediately arise:

1. Where should we initialise the chain?
2. How should we choose the proposal step sizes σ_1^2 and σ_2^2?

Unfortunately:

- there is no universal rule that always works;
- much of practical MCMC tuning is trial and error;
- diagnostics such as trace plots are essential.

Two practical questions

Once the algorithm is written down, two practical questions immediately arise:

1. Where should we initialise the chain?
2. How should we choose the proposal step sizes σ_1^2 and σ_2^2?

Unfortunately:

- there is no universal rule that always works;
- much of practical MCMC tuning is trial and error;
- diagnostics such as trace plots are essential.

This leads to two key definitions:

- **burn-in** (or warm-up),
- **thinning**.

Definition: burn-in

Burn-in (or **warm-up**) is the number of initial iterations discarded before the chain is judged to have reached the stationary distribution.

Definition: burn-in

Burn-in (or **warm-up**) is the number of initial iterations discarded before the chain is judged to have reached the stationary distribution.
In our setting, the stationary distribution is the posterior distribution.

Definition: burn-in

Burn-in (or **warm-up**) is the number of initial iterations discarded before the chain is judged to have reached the stationary distribution.
In our setting, the stationary distribution is the posterior distribution.
So if the chain starts far away from the high posterior density region, the first part of the chain may mainly reflect:

- the arbitrary starting value,
- the transient movement towards the posterior.

Definition: burn-in

Burn-in (or **warm-up**) is the number of initial iterations discarded before the chain is judged to have reached the stationary distribution.
In our setting, the stationary distribution is the posterior distribution.
So if the chain starts far away from the high posterior density region, the first part of the chain may mainly reflect:

- the arbitrary starting value,
- the transient movement towards the posterior.

These early iterations are often not representative of the target distribution and are therefore discarded.

Definition: burn-in

Burn-in (or **warm-up**) is the number of initial iterations discarded before the chain is judged to have reached the stationary distribution.
In our setting, the stationary distribution is the posterior distribution.
So if the chain starts far away from the high posterior density region, the first part of the chain may mainly reflect:

- the arbitrary starting value,
- the transient movement towards the posterior.

These early iterations are often not representative of the target distribution and are therefore discarded.
Symbolically, if the full chain is

$$\theta^{(0)}, \theta^{(1)}, \ldots, \theta^{(M)},$$

and the burn-in is B, then we keep only

$$\theta^{(B+1)}, \ldots, \theta^{(M)}.$$

Definition: thinning

Thinning means storing only every kth iteration of the chain.

Definition: thinning

Thinning means storing only every kth iteration of the chain.
For example:

- if $k = 10$, we keep iterations

$$10, 20, 30, \ldots$$

  rather than every single one;
- this reduces storage and can reduce the visible autocorrelation in the retained sample.

Definition: thinning

Thinning means storing only every kth iteration of the chain.

For example:

- if $k = 10$, we keep iterations

$$10, 20, 30, \ldots$$

  rather than every single one;
- this reduces storage and can reduce the visible autocorrelation in the retained sample.

Formally, after burn-in, instead of storing

$$\theta^{(B+1)}, \theta^{(B+2)}, \theta^{(B+3)}, \ldots,$$

we store

$$\theta^{(B+k)}, \theta^{(B+2k)}, \theta^{(B+3k)}, \ldots$$

Definition: thinning

Thinning means storing only every kth iteration of the chain.

For example:

- if $k = 10$, we keep iterations

$$10, 20, 30, \ldots$$

  rather than every single one;
- this reduces storage and can reduce the visible autocorrelation in the retained sample.

Formally, after burn-in, instead of storing

$$\theta^{(B+1)}, \theta^{(B+2)}, \theta^{(B+3)}, \ldots,$$

we store

$$\theta^{(B+k)}, \theta^{(B+2k)}, \theta^{(B+3k)}, \ldots$$

Important caveat:

- thinning does *not* fix a badly mixing chain;
- it is mainly a practical storage device, or a way to reduce redundancy in a highly correlated chain.

Intuition for burn-in

Suppose the target posterior is concentrated around a certain region of parameter space.

Intuition for burn-in

Suppose the target posterior is concentrated around a certain region of parameter space. If we start far away, the chain must first travel towards that region:

$$\theta^{(0)} \to \theta^{(1)} \to \theta^{(2)} \to \cdots$$

before it begins to look like genuine posterior sampling.

Intuition for burn-in

Suppose the target posterior is concentrated around a certain region of parameter space.
If we start far away, the chain must first travel towards that region:

$$\theta^{(0)} \to \theta^{(1)} \to \theta^{(2)} \to \cdots$$

before it begins to look like genuine posterior sampling.
So the first part of the chain is not really telling us about the posterior itself. It is telling us about the *convergence process*.

Intuition for burn-in

Suppose the target posterior is concentrated around a certain region of parameter space.
If we start far away, the chain must first travel towards that region:

$$\theta^{(0)} \to \theta^{(1)} \to \theta^{(2)} \to \cdots$$

before it begins to look like genuine posterior sampling.
So the first part of the chain is not really telling us about the posterior itself. It is telling us about the *convergence process*.
That is why these early values are usually discarded.

Intuition for burn-in

Suppose the target posterior is concentrated around a certain region of parameter space.
If we start far away, the chain must first travel towards that region:

$$\theta^{(0)} \to \theta^{(1)} \to \theta^{(2)} \to \cdots$$

before it begins to look like genuine posterior sampling.
So the first part of the chain is not really telling us about the posterior itself. It is telling us about the *convergence process*.
That is why these early values are usually discarded.
This is also why poor initialisation can be expensive:

- if the starting point is bad,
- and the proposal step size is poor,
- then burn-in can be very long.

What should a good trace plot look like?

A good trace plot should look roughly like a *hairy caterpillar*:

- no obvious trend,
- no long flat sticky segments,
- no slow drifting across the state space,
- repeated movement around the high-density region.

What should a good trace plot look like?

A good trace plot should look roughly like a *hairy caterpillar*:

- no obvious trend,
- no long flat sticky segments,
- no slow drifting across the state space,
- repeated movement around the high-density region.

In other words, after burn-in, the chain should behave approximately like stationary noise around the target region.

What should a good trace plot look like?

A good trace plot should look roughly like a *hairy caterpillar*:

- no obvious trend,
- no long flat sticky segments,
- no slow drifting across the state space,
- repeated movement around the high-density region.

In other words, after burn-in, the chain should behave approximately like stationary noise around the target region.
This does not mean independent samples. MCMC samples are usually autocorrelated.

What should a good trace plot look like?

A good trace plot should look roughly like a *hairy caterpillar*:

- no obvious trend,
- no long flat sticky segments,
- no slow drifting across the state space,
- repeated movement around the high-density region.

In other words, after burn-in, the chain should behave approximately like stationary noise around the target region.
This does not mean independent samples. MCMC samples are usually autocorrelated.
But it should mean:

- stable centre,
- stable spread,
- frequent movement,
- no obvious non-stationary transient behaviour.

When the proposal step size is too large

Suppose the proposal distribution makes huge jumps.

When the proposal step size is too large

Suppose the proposal distribution makes huge jumps.
Then from the current state, proposed values are often very far from the region of high posterior density.

When the proposal step size is too large

Suppose the proposal distribution makes huge jumps.
Then from the current state, proposed values are often very far from the region of high posterior density.
Consequence:

- most proposals are rejected;
- the chain remains stuck at the same value for many iterations;
- occasionally it makes a large accepted jump.

When the proposal step size is too large

Suppose the proposal distribution makes huge jumps.
Then from the current state, proposed values are often very far from the region of high posterior density.
Consequence:

- most proposals are rejected;
- the chain remains stuck at the same value for many iterations;
- occasionally it makes a large accepted jump.

In a trace plot this often looks like:

- long flat stretches,
- sudden jumps,
- poor exploration.

When the proposal step size is too large

Suppose the proposal distribution makes huge jumps.
Then from the current state, proposed values are often very far from the region of high posterior density.
Consequence:

- most proposals are rejected;
- the chain remains stuck at the same value for many iterations;
- occasionally it makes a large accepted jump.

In a trace plot this often looks like:

- long flat stretches,
- sudden jumps,
- poor exploration.

Interpretation:

$$\text{proposal scale too large} \quad \Rightarrow \quad \text{acceptance rate too low.}$$

When the proposal step size is too small

Now suppose the proposal distribution makes only tiny moves.

When the proposal step size is too small

Now suppose the proposal distribution makes only tiny moves.
Then most proposed values are close to the current value and are often accepted.

When the proposal step size is too small

Now suppose the proposal distribution makes only tiny moves.
Then most proposed values are close to the current value and are often accepted.
That sounds good, but there is a problem:

- the chain moves only very slowly through the posterior;
- nearby successive samples are highly correlated;
- it takes a long time to explore the whole distribution.

When the proposal step size is too small

Now suppose the proposal distribution makes only tiny moves.
Then most proposed values are close to the current value and are often accepted.
That sounds good, but there is a problem:

- the chain moves only very slowly through the posterior;
- nearby successive samples are highly correlated;
- it takes a long time to explore the whole distribution.

In a trace plot this often looks like:

- smooth, sticky local wandering,
- slow drift,
- poor long-range exploration.

When the proposal step size is too small

Now suppose the proposal distribution makes only tiny moves.
Then most proposed values are close to the current value and are often accepted.
That sounds good, but there is a problem:

- the chain moves only very slowly through the posterior;
- nearby successive samples are highly correlated;
- it takes a long time to explore the whole distribution.

In a trace plot this often looks like:

- smooth, sticky local wandering,
- slow drift,
- poor long-range exploration.

Interpretation:

$$\text{proposal scale too small} \quad \Rightarrow \quad \text{high acceptance, but inefficient mixing.}$$

The tuning problem

So proposal tuning is a balancing act.

The tuning problem

So proposal tuning is a balancing act.
We want moves that are:

- not so large that almost everything is rejected,
- not so small that the chain crawls through the space.

The tuning problem

So proposal tuning is a balancing act.
We want moves that are:

- not so large that almost everything is rejected,
- not so small that the chain crawls through the space.

This is often described as looking for a *Goldilocks* proposal scale:

not too big, not too small, but about right.

The tuning problem

So proposal tuning is a balancing act.
We want moves that are:

- not so large that almost everything is rejected,
- not so small that the chain crawls through the space.

This is often described as looking for a *Goldilocks* proposal scale:

not too big, not too small, but about right.

Unfortunately, there is no universal closed-form answer.

The tuning problem

So proposal tuning is a balancing act.
We want moves that are:

- not so large that almost everything is rejected,
- not so small that the chain crawls through the space.

This is often described as looking for a *Goldilocks* proposal scale:

not too big, not too small, but about right.

Unfortunately, there is no universal closed-form answer.
In practice we often:

- guess a proposal scale,
- run the chain,
- inspect trace plots and acceptance behaviour,
- adjust the tuning parameter,
- run again.

Initialisation is also hard

A second difficult practical choice is the starting value.

Initialisation is also hard

A second difficult practical choice is the starting value.
Sometimes we can use:

- a sensible guess from the data,
- a maximum likelihood estimate,
- a method-of-moments estimate,
- a previously fitted value from a related model.

Initialisation is also hard

A second difficult practical choice is the starting value.
Sometimes we can use:

- a sensible guess from the data,
- a maximum likelihood estimate,
- a method-of-moments estimate,
- a previously fitted value from a related model.

But often we are largely guessing.

Initialisation is also hard

A second difficult practical choice is the starting value.
Sometimes we can use:

- a sensible guess from the data,
- a maximum likelihood estimate,
- a method-of-moments estimate,
- a previously fitted value from a related model.

But often we are largely guessing.
Poor initialisation can lead to:

- long burn-in,
- delayed entry into the high posterior density region,
- wasted computation.

Initialisation is also hard

A second difficult practical choice is the starting value.
Sometimes we can use:

- a sensible guess from the data,
- a maximum likelihood estimate,
- a method-of-moments estimate,
- a previously fitted value from a related model.

But often we are largely guessing.
Poor initialisation can lead to:

- long burn-in,
- delayed entry into the high posterior density region,
- wasted computation.

So while burn-in can remove the initial transient, it does not make a poor initialisation free. It only mitigates the damage.

Burn-in and tuning can interact badly

Burn-in and proposal scale are not separate issues.

Burn-in and tuning can interact badly

Burn-in and proposal scale are not separate issues.
If the chain starts far from the target region and the proposal step size is poor, then convergence can be extremely slow.

Burn-in and tuning can interact badly

Burn-in and proposal scale are not separate issues.
If the chain starts far from the target region and the proposal step size is poor, then convergence can be extremely slow.
For example:

- a bad initial value plus a tiny proposal scale means the chain inches slowly toward the posterior;
- a bad initial value plus an enormous proposal scale means many huge moves are rejected before a useful proposal is ever accepted.

Burn-in and tuning can interact badly

Burn-in and proposal scale are not separate issues.
If the chain starts far from the target region and the proposal step size is poor, then convergence can be extremely slow.
For example:

- a bad initial value plus a tiny proposal scale means the chain inches slowly toward the posterior;
- a bad initial value plus an enormous proposal scale means many huge moves are rejected before a useful proposal is ever accepted.

So a badly tuned chain can spend a very long time in burn-in.

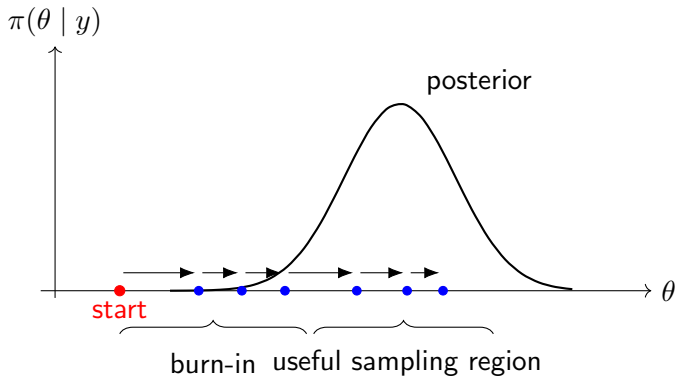
Burn-in and tuning can interact badly

Burn-in and proposal scale are not separate issues.
If the chain starts far from the target region and the proposal step size is poor, then convergence can be extremely slow.
For example:

- a bad initial value plus a tiny proposal scale means the chain inches slowly toward the posterior;
- a bad initial value plus an enormous proposal scale means many huge moves are rejected before a useful proposal is ever accepted.

So a badly tuned chain can spend a very long time in burn-in.
This is one reason MCMC can be computationally expensive in practice.

A conceptual picture of burn-in



The early part is mostly travel toward the posterior; only later does the chain behave like posterior sampling.

What is the curse of dimensionality?

So far we have sounded optimistic:

MCMC lets us sample from difficult posterior distributions.

What is the curse of dimensionality?

So far we have sounded optimistic:

MCMC lets us sample from difficult posterior distributions.

That is true.

What is the curse of dimensionality?

So far we have sounded optimistic:

MCMC lets us sample from difficult posterior distributions.

That is true.
But there is a serious limitation:

MCMC becomes much harder as dimension increases.

What is the curse of dimensionality?

So far we have sounded optimistic:

MCMC lets us sample from difficult posterior distributions.

That is true.
But there is a serious limitation:

MCMC becomes much harder as dimension increases.

In high dimensions:

- the geometry of the target distribution becomes unintuitive,
- most of the mass is often far from the mode,
- it becomes difficult to design proposals that explore efficiently,
- rejection rates can become very high.

What is the curse of dimensionality?

So far we have sounded optimistic:

MCMC lets us sample from difficult posterior distributions.

That is true.
But there is a serious limitation:

MCMC becomes much harder as dimension increases.

In high dimensions:

- the geometry of the target distribution becomes unintuitive,
- most of the mass is often far from the mode,
- it becomes difficult to design proposals that explore efficiently,
- rejection rates can become very high.

This phenomenon is part of what is called the **curse of dimensionality**.

A geometric example: mass in a high-dimensional sphere

Consider a uniform distribution on an n-dimensional ball of radius R.

A geometric example: mass in a high-dimensional sphere

Consider a uniform distribution on an n-dimensional ball of radius R.
Its volume is

$$V_n(R) = \frac{\pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)} R^n.$$

A geometric example: mass in a high-dimensional sphere

Consider a uniform distribution on an n-dimensional ball of radius R.
Its volume is

$$V_n(R) = \frac{\pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)} R^n.$$

Now consider a smaller concentric ball of radius $R_1 < R$. Its volume is

$$V_n(R_1) = \frac{\pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)} R_1^n.$$

A geometric example: mass in a high-dimensional sphere

Consider a uniform distribution on an n-dimensional ball of radius R.
Its volume is

$$V_n(R) = \frac{\pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)} R^n.$$

Now consider a smaller concentric ball of radius $R_1 < R$. Its volume is

$$V_n(R_1) = \frac{\pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)} R_1^n.$$

Therefore the proportion of volume in the *smaller* ball is

$$\frac{V_n(R_1)}{V_n(R)} = \left(\frac{R_1}{R}\right)^n.$$

A geometric example: mass in a high-dimensional sphere

Consider a uniform distribution on an n-dimensional ball of radius R.
Its volume is

$$V_n(R) = \frac{\pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)} R^n.$$

Now consider a smaller concentric ball of radius $R_1 < R$. Its volume is

$$V_n(R_1) = \frac{\pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)} R_1^n.$$

Therefore the proportion of volume in the *smaller* ball is

$$\frac{V_n(R_1)}{V_n(R)} = \left(\frac{R_1}{R}\right)^n.$$

So the proportion of volume in the *outer shell* is

$$1 - \left(\frac{R_1}{R}\right)^n.$$

A numerical illustration

Take:

$$n = 10, \qquad R = 1, \qquad R_1 = \frac{1}{2}.$$

Then the proportion of mass in the outer shell is

$$1 - \left(\frac{1}{2}\right)^{10} = 1 - \frac{1}{1024} \approx 0.9990.$$

A numerical illustration

Take:

$$n = 10, \qquad R = 1, \qquad R_1 = \frac{1}{2}.$$

Then the proportion of mass in the outer shell is

$$1 - \left(\frac{1}{2}\right)^{10} = 1 - \frac{1}{1024} \approx 0.9990.$$

So in 10 dimensions:

- only about 0.1% of the volume lies inside the smaller radius-$1/2$ ball,
- about 99.9% lies in the outer shell.

A numerical illustration

Take:

$$n = 10, \qquad R = 1, \qquad R_1 = \frac{1}{2}.$$

Then the proportion of mass in the outer shell is

$$1 - \left(\frac{1}{2}\right)^{10} = 1 - \frac{1}{1024} \approx 0.9990.$$

So in 10 dimensions:

- only about 0.1% of the volume lies inside the smaller radius-$1/2$ ball,
- about 99.9% lies in the outer shell.

This is already striking at dimension 10.

A numerical illustration

Take:

$$n = 10, \qquad R = 1, \qquad R_1 = \frac{1}{2}.$$

Then the proportion of mass in the outer shell is

$$1 - \left(\frac{1}{2}\right)^{10} = 1 - \frac{1}{1024} \approx 0.9990.$$

So in 10 dimensions:

- only about 0.1% of the volume lies inside the smaller radius-$1/2$ ball,
- about 99.9% lies in the outer shell.

This is already striking at dimension 10.
In even higher dimensions, this effect is stronger still.

A numerical illustration

Take:

$$n = 10, \qquad R = 1, \qquad R_1 = \frac{1}{2}.$$

Then the proportion of mass in the outer shell is

$$1 - \left(\frac{1}{2}\right)^{10} = 1 - \frac{1}{1024} \approx 0.9990.$$

So in 10 dimensions:

- only about 0.1% of the volume lies inside the smaller radius-$1/2$ ball,
- about 99.9% lies in the outer shell.

This is already striking at dimension 10.
In even higher dimensions, this effect is stronger still.
The message is that geometric intuition from 1D, 2D, or 3D can be very misleading in high-dimensional problems.

Why this matters for MCMC

In many problems:

- the posterior mode may be near the "centre",
- but much of the probability mass may lie in a surrounding shell.

Why this matters for MCMC

In many problems:

- the posterior mode may be near the "centre",
- but much of the probability mass may lie in a surrounding shell.

This creates a difficult tension for proposal design.

Why this matters for MCMC

In many problems:

- the posterior mode may be near the "centre",
- but much of the probability mass may lie in a surrounding shell.

This creates a difficult tension for proposal design.
A proposal that stays too close to the mode may miss much of the mass.

Why this matters for MCMC

In many problems:

- the posterior mode may be near the "centre",
- but much of the probability mass may lie in a surrounding shell.

This creates a difficult tension for proposal design.
A proposal that stays too close to the mode may miss much of the mass.
A proposal that jumps too aggressively into the tails may be rejected too often.

Why this matters for MCMC

In many problems:

- the posterior mode may be near the "centre",
- but much of the probability mass may lie in a surrounding shell.

This creates a difficult tension for proposal design.
A proposal that stays too close to the mode may miss much of the mass.
A proposal that jumps too aggressively into the tails may be rejected too often.
So as dimension grows, it becomes increasingly hard to design simple random-walk proposals that:

- move enough to explore,
- but not so much that acceptance collapses.

Why this matters for MCMC

In many problems:

- the posterior mode may be near the "centre",
- but much of the probability mass may lie in a surrounding shell.

This creates a difficult tension for proposal design.
A proposal that stays too close to the mode may miss much of the mass.
A proposal that jumps too aggressively into the tails may be rejected too often.
So as dimension grows, it becomes increasingly hard to design simple random-walk proposals that:

- move enough to explore,
- but not so much that acceptance collapses.

This is one reason naive Metropolis–Hastings can perform very poorly in high dimensions.

Why even moderate dimensions can be hard

High dimension does not necessarily mean hundreds or thousands of parameters.

Why even moderate dimensions can be hard

High dimension does not necessarily mean hundreds or thousands of parameters.
Even a model with around 10 parameters can already exhibit serious difficulty.

Why even moderate dimensions can be hard

High dimension does not necessarily mean hundreds or thousands of parameters.
Even a model with around 10 parameters can already exhibit serious difficulty.
And in practical statistical models it is easy to reach that scale:

- several covariates,
- interaction terms,
- hierarchical effects,
- nuisance parameters,
- latent variables.

Why even moderate dimensions can be hard

High dimension does not necessarily mean hundreds or thousands of parameters.
Even a model with around 10 parameters can already exhibit serious difficulty.
And in practical statistical models it is easy to reach that scale:

- several covariates,
- interaction terms,
- hierarchical effects,
- nuisance parameters,
- latent variables.

So the curse of dimensionality is not only a theoretical issue. It affects realistic data analysis very quickly.

What have we learned about MCMC so far?

Good news

MCMC gives us a general way to approximate difficult posterior distributions by simulation.

What have we learned about MCMC so far?

Good news

MCMC gives us a general way to approximate difficult posterior distributions by simulation.

Bad news

MCMC can be:

- computationally expensive,
- sensitive to tuning,
- slow to converge,
- highly inefficient in high dimensions.

What have we learned about MCMC so far?

Good news

MCMC gives us a general way to approximate difficult posterior distributions by simulation.

Bad news

MCMC can be:

- computationally expensive,
- sensitive to tuning,
- slow to converge,
- highly inefficient in high dimensions.

So MCMC is powerful, but it is not magic.

What have we learned about MCMC so far?

Good news

MCMC gives us a general way to approximate difficult posterior distributions by simulation.

Bad news

MCMC can be:

- computationally expensive,
- sensitive to tuning,
- slow to converge,
- highly inefficient in high dimensions.

So MCMC is powerful, but it is not magic.
It solves a difficult problem by replacing difficult integration with potentially difficult computation.

Beyond basic MCMC

There are other Monte Carlo methods beyond the simple random-walk MCMC framework we have focused on.

Beyond basic MCMC

There are other Monte Carlo methods beyond the simple random-walk MCMC framework we have focused on.
Two important examples are:

1. **Sequential Monte Carlo (SMC)**
2. **Hamiltonian Monte Carlo (HMC)**

Beyond basic MCMC

There are other Monte Carlo methods beyond the simple random-walk MCMC framework we have focused on.
Two important examples are:

1. **Sequential Monte Carlo (SMC)**
2. **Hamiltonian Monte Carlo (HMC)**

These methods aim to address some limitations of basic MCMC:

- poor efficiency,
- difficulty in high dimensions,
- inability to reuse previous computation when data arrive sequentially.

Sequential Monte Carlo: the basic idea

A weakness of standard MCMC is that the posterior changes whenever new data arrive.

Sequential Monte Carlo: the basic idea

A weakness of standard MCMC is that the posterior changes whenever new data arrive.
If we fit a posterior using data $y_{1:t}$ and then receive a new observation y_{t+1}, the target becomes

$$\pi(\theta \mid y_{1:t+1}),$$

which is different from the old posterior.

Sequential Monte Carlo: the basic idea

A weakness of standard MCMC is that the posterior changes whenever new data arrive.
If we fit a posterior using data $y_{1:t}$ and then receive a new observation y_{t+1}, the target becomes

$$\pi(\theta \mid y_{1:t+1}),$$

which is different from the old posterior.
In a basic MCMC approach, we may need to start again.

Sequential Monte Carlo: the basic idea

A weakness of standard MCMC is that the posterior changes whenever new data arrive.
If we fit a posterior using data $y_{1:t}$ and then receive a new observation y_{t+1}, the target becomes

$$\pi(\theta \mid y_{1:t+1}),$$

which is different from the old posterior.
In a basic MCMC approach, we may need to start again.
Sequential Monte Carlo tries to update a collection of particles as the target evolves.

Sequential Monte Carlo: the basic idea

A weakness of standard MCMC is that the posterior changes whenever new data arrive.
If we fit a posterior using data $y_{1:t}$ and then receive a new observation y_{t+1}, the target becomes

$$\pi(\theta \mid y_{1:t+1}),$$

which is different from the old posterior.
In a basic MCMC approach, we may need to start again.
Sequential Monte Carlo tries to update a collection of particles as the target evolves.
So rather than repeatedly restarting from scratch, it propagates and reweights samples over time.

Sequential Monte Carlo: the basic idea

A weakness of standard MCMC is that the posterior changes whenever new data arrive.
If we fit a posterior using data $y_{1:t}$ and then receive a new observation y_{t+1}, the target becomes

$$\pi(\theta \mid y_{1:t+1}),$$

which is different from the old posterior.
In a basic MCMC approach, we may need to start again.
Sequential Monte Carlo tries to update a collection of particles as the target evolves.
So rather than repeatedly restarting from scratch, it propagates and reweights samples over time.
This is particularly useful in settings such as:

- time series,
- epidemic models,
- streaming data,
- online Bayesian updating.

Hamiltonian Monte Carlo: the basic idea

Hamiltonian Monte Carlo uses gradient information from the log-posterior.

Hamiltonian Monte Carlo: the basic idea

Hamiltonian Monte Carlo uses gradient information from the log-posterior.
Instead of making a blind random walk, HMC tries to move through parameter space in a more informed direction.

Hamiltonian Monte Carlo: the basic idea

Hamiltonian Monte Carlo uses gradient information from the log-posterior.
Instead of making a blind random walk, HMC tries to move through parameter space in a more informed direction.
Very roughly:

- the posterior landscape is treated like an energy surface,
- an auxiliary momentum variable is introduced,
- the algorithm follows approximate Hamiltonian dynamics,
- this allows long-distance proposals with high acceptance.

Hamiltonian Monte Carlo: the basic idea

Hamiltonian Monte Carlo uses gradient information from the log-posterior.
Instead of making a blind random walk, HMC tries to move through parameter space in a more informed direction.
Very roughly:

- the posterior landscape is treated like an energy surface,
- an auxiliary momentum variable is introduced,
- the algorithm follows approximate Hamiltonian dynamics,
- this allows long-distance proposals with high acceptance.

Main advantage:

- much better exploration than simple random-walk MH in many high-dimensional problems.

Hamiltonian Monte Carlo: the basic idea

Hamiltonian Monte Carlo uses gradient information from the log-posterior.
Instead of making a blind random walk, HMC tries to move through parameter space in a more informed direction.
Very roughly:

- the posterior landscape is treated like an energy surface,
- an auxiliary momentum variable is introduced,
- the algorithm follows approximate Hamiltonian dynamics,
- this allows long-distance proposals with high acceptance.

Main advantage:

- much better exploration than simple random-walk MH in many high-dimensional problems.

Main cost:

- the mathematics and implementation are more sophisticated.