Markov Chain Monte Carlo

5.1 Motivation and Background

Markov Chain Monte Carlo, or MCMC, is a method that allows us to generate samples from a posterior distribution when we cannot sample from that posterior directly. Up until this point, most of the posterior distributions we have worked with have been convenient. For example, if we have a binomial likelihood and a beta prior, then we know the posterior is beta. If we have a poisson likelihood and a gamma prior, then we know the posterior is gamma. This is lovely. This is tidy. This is the Bayesian equivalent of a drawer where all of the socks are already paired.

Unfortunately, posterior distributions are not always so cooperative.

In many problems, particularly when working with hierarchical models, the posterior distribution will not have a familiar form. We may still know the posterior up to proportionality, but we will not be able to identify it as a beta, gamma, normal, inverse-gamma, or any other distribution that R can conveniently sample from.

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{\int p(y|\theta)p(\theta)d\theta}$$

$$\propto p(y|\theta)p(\theta)$$

The denominator is the normalizing constant. If we can calculate this integral, then we can normalize the posterior and sample from it in the usual way. If we cannot calculate the integral, then we are left with an unnormalized posterior. This is exactly where MCMC becomes useful.

Important Idea: MCMC lets us sample from a posterior distribution even when we only know the posterior up to proportionality.

$$p(\theta|y) \propto p(y|\theta)p(\theta)$$

Why do we need samples? Remember that the goal of Bayesian inference is not usually to write down a posterior distribution and then lovingly stare at it. We want to summarize the posterior distribution. This typically means calculating things like:

1. Posterior means
2. Posterior standard deviations
3. Credible intervals

4. Posterior probabilities
5. Posterior predictive quantities

If we can generate a large number of draws from the posterior distribution, then we can approximate all of these quantities using the samples.

$$\theta^{(1)}, \theta^{(2)}, ..., \theta^{(S)} \sim p(\theta|y)$$

$$\mathbb{E}[\theta|y] \approx \frac{1}{S}\sum_{s=1}^{S}\theta^{(s)}$$

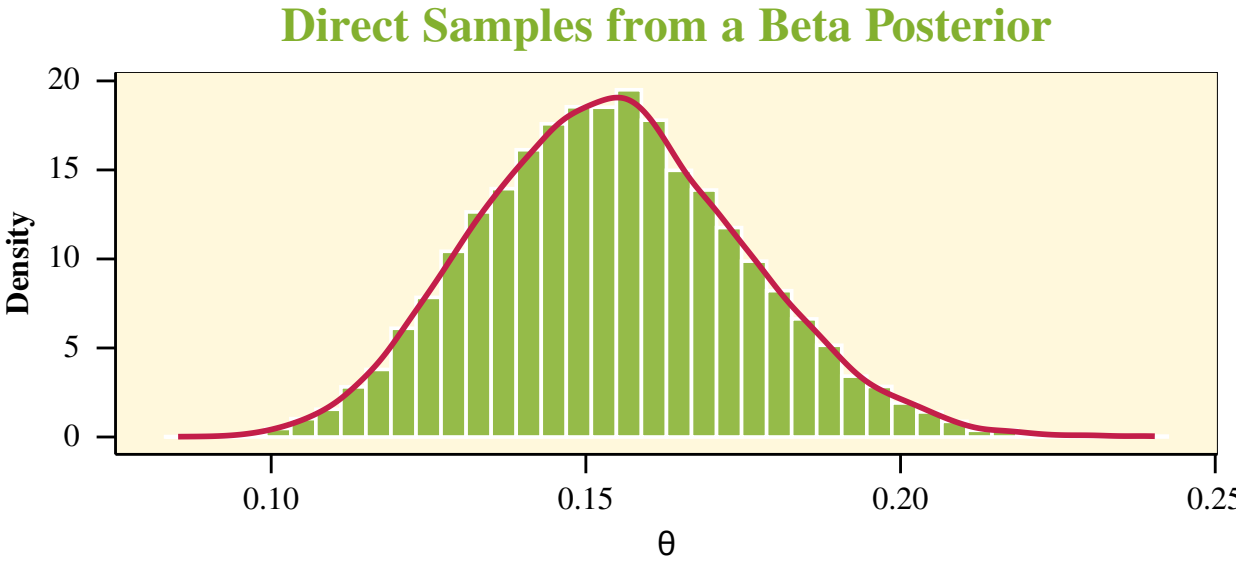
$$95\% \text{ Credible Interval} \approx \left(Q_{0.025}(\theta^{(s)}),\ Q_{0.975}(\theta^{(s)})\right)$$

This is the same basic idea we have already been using in R. If we can sample from a distribution, then we can summarize the distribution by summarizing the samples. MCMC simply gives us a way to get those samples when the posterior is not directly available.

Direct Sampling Suppose we know that:

$$\theta|y \sim \text{Beta}(45, 247)$$

Then there is no need to be clever. We can simply use `rbeta()` to sample from the posterior distribution.



In this case, R already knows how to generate samples from the posterior. The issue is that, for many realistic posterior distributions, we do not have a nice `rposterior()` function sitting around waiting for us.

Rejection Sampling Before getting to MCMC, it is useful to think about one other possible strategy: rejection sampling. In rejection sampling, we choose an envelope distribution $q(\theta)$ that is easy to sample from and that sits above the posterior distribution after being multiplied by some constant c.

$$cq(\theta) \geq p(\theta|y) \quad \text{for all } \theta$$

The General Procedure

1. Draw a proposed value $\theta^\star$ from $q(\theta)$.
2. Draw $u \sim \text{Uniform}(0, 1)$.
3. Accept $\theta^\star$ if:
$$u \leq \frac{p(\theta^\star|y)}{cq(\theta^\star)}$$
4. If the value is accepted, keep it. If the value is rejected, throw it away.
5. Repeat until we have enough accepted samples.

This works beautifully when we can find a good envelope distribution. The problem is that finding a good envelope distribution can be extremely difficult, especially in multiple dimensions. In the hierarchical models from Chapter 4, for example, we would need an envelope for the posterior distribution of (α, β). This is possible in theory, but not especially fun in practice.

A second problem with rejection sampling is efficiency. Even with a well-chosen envelope, many proposed values will be rejected and thrown away. If the envelope fits the posterior loosely, the acceptance rate can become so low that the sampler produces almost no usable draws per unit of computation. In high-dimensional settings, this problem compounds rapidly: as the number of parameters grows, the volume of the parameter space expands in ways that make it nearly impossible to construct an envelope tight enough to maintain a reasonable acceptance rate. This phenomenon is sometimes called the curse of dimensionality, and it is one of the main reasons rejection sampling does not scale to the kinds of models we encounter in practice.

Markov Chains MCMC takes a different approach. Rather than trying to generate independent samples from the posterior, MCMC generates a sequence of dependent samples. The samples are dependent because each new value is generated using the previous value.

A Markov chain is a sequence of random variables where the distribution of the current value depends only on the previous value:

$$p(\theta_t|\theta_{t-1}, \theta_{t-2}, ..., \theta_1) = p(\theta_t|\theta_{t-1})$$

In other words, the chain has a very short memory. It knows where it currently is, but it does not care how it got there.

$$\theta_1 \to \theta_2 \to \theta_3 \to \cdots \to \theta_t$$

For MCMC, we design the Markov chain so that, after it has run for long enough, its stationary distribution is the posterior distribution we care about.

$$\text{Long-run distribution of the chain} \quad = \quad p(\theta|y)$$

This means that the early values of the chain may not be especially useful, because the chain may still be wandering away from its starting point. We call this early portion the burn-in. Once the chain has settled into the high probability region of the posterior, we can use the remaining samples for posterior inference.

One important consequence of using dependent samples is that MCMC samples contain less information per draw than independent samples would. Two consecutive draws from a Markov chain tend to be similar to one another, which means they are not contributing two fully independent pieces of information about the posterior. The degree of dependence between consecutive draws is measured by autocorrelation. High autocorrelation means the chain is moving slowly through the parameter space, and we will need many more draws to get reliable posterior summaries than we would with independent samples. The effective sample size (ESS) formalizes this idea: it estimates the number of independent samples that would carry the same amount of information as our correlated chain.

$$\text{ESS} = \frac{S}{1 + 2\sum_{k=1}^{\infty} \rho_k}$$

where S is the total number of post-burn-in draws and ρ_k is the autocorrelation at lag k. An ESS much smaller than S is a warning sign that the chain is mixing poorly. We will return to diagnosing and addressing these issues in Section 5.3.

MCMC Workflow:

1. Build a Markov chain whose stationary distribution is the posterior distribution.
2. Run the chain for many iterations.
3. Remove the early burn-in samples.
4. Treat the remaining draws as approximate samples from the posterior.
5. Summarize those samples.

5.2 Metropolis-Hastings Algorithm

The main MCMC algorithm we will use is the Metropolis-Hastings Algorithm. The basic idea is that, at each iteration, we propose a new value for the parameter and then decide whether or not to accept it.

For a reusable R framework, see the Metropolis-Hastings template in the appendix.

t	= current iteration
θ_t	= current value of the chain
$\theta^\star$	= proposed value
$p(\theta\|y)$	= unnormalized posterior density
$Q(\theta^\star\|\theta_t)$	= proposal distribution
$q(\theta^\star\|\theta_t)$	= proposal density
$\alpha(\theta^\star\|\theta_t)$	= acceptance probability

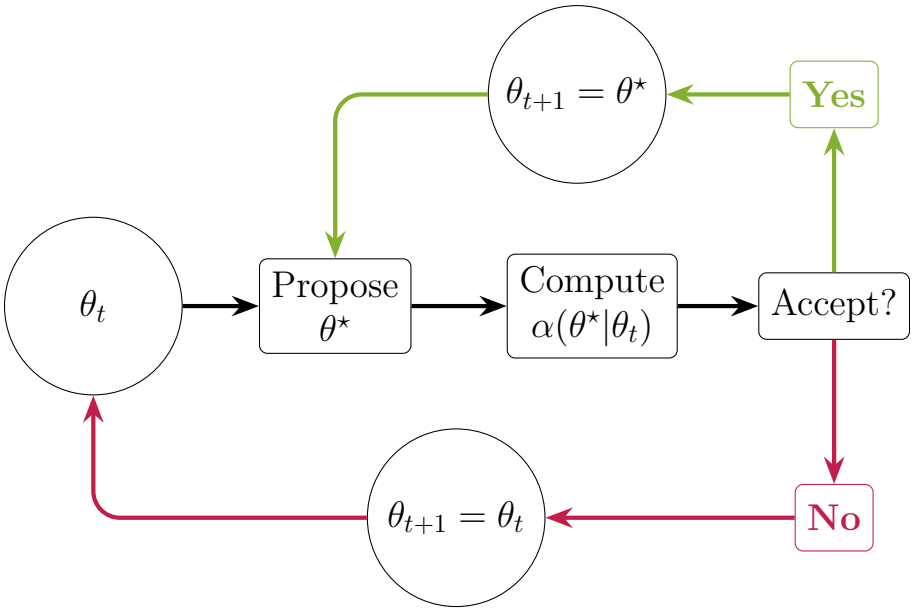
The proposal distribution is the distribution we use to suggest a possible next value. The acceptance probability tells us whether the proposed value should be accepted.

The Procedure The Metropolis-Hastings algorithm can be written as follows:

1. Choose a starting value θ_1.
2. For iteration $t = 1, 2, ..., T$:
   (a) Draw a proposed value: $\theta^\star \sim Q(\theta^\star|\theta_t)$
   (b) Calculate the acceptance probability:
   $$\alpha(\theta^\star|\theta_t) = \min\left(1, \frac{p(\theta^\star|y)q(\theta_t|\theta^\star)}{p(\theta_t|y)q(\theta^\star|\theta_t)}\right)$$
   (c) Draw: $u \sim \text{Uniform}(0, 1)$
   (d) Set the next value using the acceptance rule:
   $$\theta_{t+1} = \begin{cases} \theta^\star, & \text{if } u \leq \alpha(\theta^\star|\theta_t), \\ \theta_t, & \text{if } u > \alpha(\theta^\star|\theta_t). \end{cases}$$
3. After many iterations, summarize the values of θ_t.

Notice that if we reject a proposed value, the chain does not disappear or skip the iteration. Instead, the chain stays at the previous value. This is why traceplots sometimes have flat spots.

The basic movement of the algorithm looks like this:



Starting from the current value θ_t, the algorithm proposes a candidate $\theta^\star$ from the proposal distribution, then computes the acceptance probability $\alpha(\theta^\star|\theta_t)$ by comparing the posterior densities (and proposal densities, if asymmetric) at $\theta^\star$ and θ_t. If the proposal is accepted (green path), $\theta^\star$ becomes the new current value θ_{t+1}, and the next proposal is generated from this updated value. If rejected (red path), the chain remains at θ_t, so $\theta_{t+1} = \theta_t$, and a new proposal is again drawn from this unchanged value. This loop repeats for every iteration, building up the Markov chain one step at a time.

Why does the normalizing constant not matter? One of the nicest things about Metropolis-Hastings is that the acceptance probability involves a ratio of posterior densities. Suppose:

$$p(\theta|y) = \frac{g(\theta)}{C}$$

where $g(\theta)$ is the unnormalized posterior and C is the normalizing constant. Then:

$$\frac{p(\theta^\star|y)}{p(\theta_t|y)} = \frac{\frac{g(\theta^\star)}{C}}{\frac{g(\theta_t)}{C}} = \frac{g(\theta^\star)}{g(\theta_t)}$$

The normalizing constant cancels. This is the magic trick. It is not actually magic, of course, but it is very considerate algebra.

Symmetric Proposals and the Metropolis Algorithm A common choice is to use a normal proposal distribution centered at the current value:

$$\theta^\star|\theta_t \sim N(\theta_t, \sigma_q^2)$$

This is a random walk proposal. If θ_t is the current value, the proposal distribution suggests values near θ_t. Since the normal distribution is symmetric:

$$q(\theta^\star|\theta_t) = q(\theta_t|\theta^\star)$$

Therefore, the proposal densities cancel in the acceptance probability:

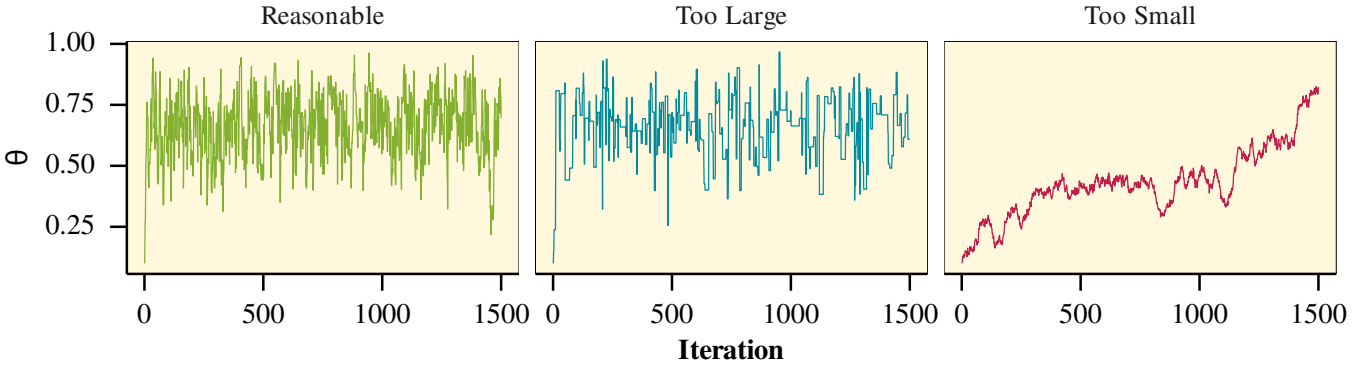
$$\alpha(\theta^\star|\theta_t) = \min\left(1, \frac{p(\theta^\star|y)\cancel{q(\theta_t|\theta^\star)}}{p(\theta_t|y)\cancel{q(\theta^\star|\theta_t)}}\right) = \min\left(1, \frac{p(\theta^\star|y)}{p(\theta_t|y)}\right)$$

When Metropolis-Hastings uses a symmetric proposal distribution, we call it the Metropolis algorithm.

How should we choose the proposal variance? The proposal variance determines how large the proposed moves are. This matters a lot.

σ_q^2 too small $\implies$ The chain accepts often, but crawls slowly
σ_q^2 too large $\implies$ The chain proposes wild moves and rejects often
σ_q^2 reasonable $\implies$ The chain moves around the posterior efficiently

Effect of Proposal Variance on the Chain



The first chain technically moves, but it moves painfully slowly. The third chain also technically moves, but it spends too much time rejecting proposed values. The middle chain is much healthier because it explores the posterior without getting stuck in place.

What do we need in practice? To use Metropolis-Hastings, we need three things:

1. We need to be able to evaluate the unnormalized posterior density $p(\theta|y)$.
2. We need to be able to sample proposed values from $Q(\theta^\star|\theta_t)$.
3. We need to be able to evaluate the proposal density $q(\theta^\star|\theta_t)$, unless the proposal distribution is symmetric and cancels.

The difference between Q and q is subtle but important:

$$Q(\theta^\star|\theta_t) = \text{the distribution we sample from}$$
$$q(\theta^\star|\theta_t) = \text{the density of that proposal distribution}$$

Example: Good Luck Advertisement Data In Chapter 4, we ended with the Restaurant Good Luck advertisement example. The data and model are:

Area	n_j	y_j
Rochester	73	20
Henrietta	32	5
Pittsford	52	18
Brighton	21	8
Fairport	24	6
Penfield	19	4
Webster	34	10
Irondequoit	28	7

$$y_j|\theta_j \sim \text{Bin}(n_j, \theta_j)$$
$$\theta_j|\alpha,\beta \sim \text{Beta}(\alpha,\beta)$$
$$p(\alpha,\beta) \propto (\alpha+\beta)^{-5/2}$$

First, the binomial likelihood and beta prior give:

$$p(\vec{y}|\vec{\theta}) = \prod_{j=1}^{J} p(y_j|\theta_j) \propto \prod_{j=1}^{J} \theta_j^{y_j}(1-\theta_j)^{n_j-y_j}$$

$$p(\vec{\theta}|\alpha,\beta) = \prod_{j=1}^{J} p(\theta_j|\alpha,\beta) \;=\; \prod_{j=1}^{J} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta_j^{\alpha-1}(1-\theta_j)^{\beta-1}$$

Therefore, the joint posterior is:

$$\begin{aligned} p(\theta_1,...,\theta_J,\alpha,\beta|y) &\propto p(\vec{y}|\vec{\theta})p(\vec{\theta}|\alpha,\beta)p(\alpha,\beta) \\ &\propto \left[\prod_{j=1}^{J} \theta_j^{y_j}(1-\theta_j)^{n_j-y_j}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta_j^{\alpha-1}(1-\theta_j)^{\beta-1}\right](\alpha+\beta)^{-5/2} \\ &\propto (\alpha+\beta)^{-5/2}\left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\right)^{J}\prod_{j=1}^{J}\theta_j^{y_j+\alpha-1}(1-\theta_j)^{n_j-y_j+\beta-1} \end{aligned}$$

Now, if we condition on α and β, anything that does not involve θ_j can be treated as constant:

$$\begin{aligned} p(\theta_1, ..., \theta_J|\alpha, \beta, y) &\propto \prod_{j=1}^{J} \theta_j^{y_j+\alpha-1}(1-\theta_j)^{n_j-y_j+\beta-1} \\ &\propto \prod_{j=1}^{J} p(\theta_j|\alpha, \beta, y_j) \\ \theta_j|\alpha, \beta, y_j &\sim \text{Beta}(y_j+\alpha,\ n_j-y_j+\beta) \quad \textbf{Conditional Posterior} \end{aligned}$$

So the conditional posterior for each θ_j is simple. However, the posterior for (α, β) is not a familiar distribution. To get this marginal posterior, we integrate the θ_j values out of the joint posterior:

$$\begin{aligned} p(\alpha, \beta|y) &\propto \int \cdots \int p(\theta_1, ..., \theta_J, \alpha, \beta|y) d\theta_1 \cdots d\theta_J \\ &\propto (\alpha+\beta)^{-5/2} \left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\right)^J \prod_{j=1}^{J} \int_0^1 \theta_j^{y_j+\alpha-1}(1-\theta_j)^{n_j-y_j+\beta-1} d\theta_j \\ &\propto (\alpha+\beta)^{-5/2} \prod_{j=1}^{J} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \frac{\Gamma(y_j+\alpha)\Gamma(n_j-y_j+\beta)}{\Gamma(n_j+\alpha+\beta)} \quad \textbf{Marginal Posterior} \end{aligned}$$

This is exactly the kind of distribution where MCMC is useful. We can evaluate it, but we do not know how to sample directly from it.

For the Metropolis-Hastings step, we propose a new pair $(\alpha^\star, \beta^\star)$ from a symmetric normal random walk:

$$\begin{aligned} \alpha^\star &\sim N(\alpha_t, \sigma_\alpha^2) \\ \beta^\star &\sim N(\beta_t, \sigma_\beta^2) \end{aligned}$$

Because this proposal is symmetric, the proposal densities cancel:

$$\begin{aligned} \alpha\big((\alpha^\star, \beta^\star)|(\alpha_t, \beta_t)\big) &= \min\left(1,\ \frac{p(\alpha^\star, \beta^\star|y)\cancel{q((\alpha_t, \beta_t)|(\alpha^\star, \beta^\star))}}{p(\alpha_t, \beta_t|y)\cancel{q((\alpha^\star, \beta^\star)|(\alpha_t, \beta_t))}}\right) \\ &= \min\left(1,\ \frac{p(\alpha^\star, \beta^\star|y)}{p(\alpha_t, \beta_t|y)}\right) \end{aligned}$$

In practice, the code uses the log posterior because products of many small numbers can underflow:

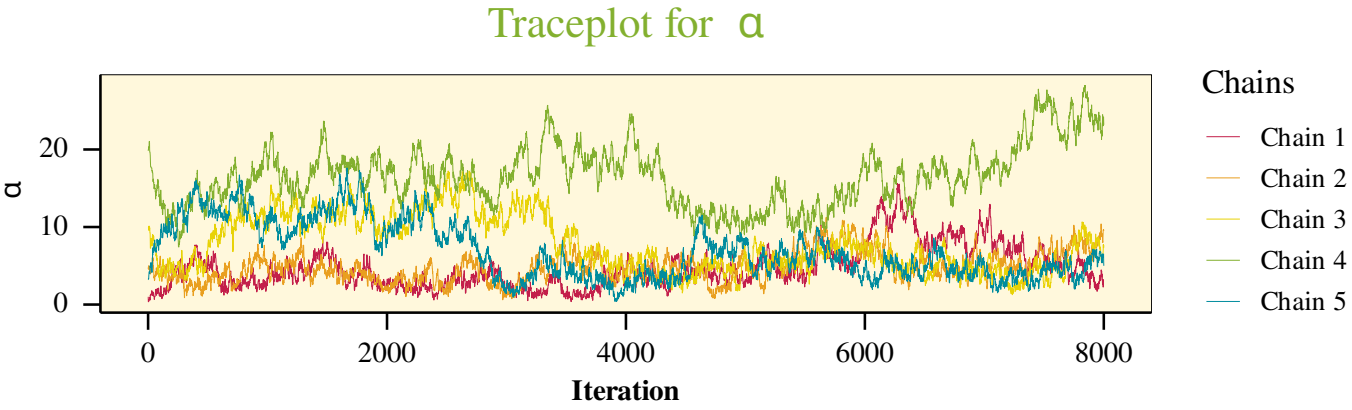
$$\begin{aligned} \log p(\alpha, \beta|y) = &-\frac{5}{2}\log(\alpha+\beta) + J\left[\log\Gamma(\alpha+\beta) - \log\Gamma(\alpha) - \log\Gamma(\beta)\right] \\ &+ \sum_{j=1}^{J}\left[\log\Gamma(y_j+\alpha) + \log\Gamma(n_j-y_j+\beta) - \log\Gamma(n_j+\alpha+\beta)\right] + c \end{aligned}$$

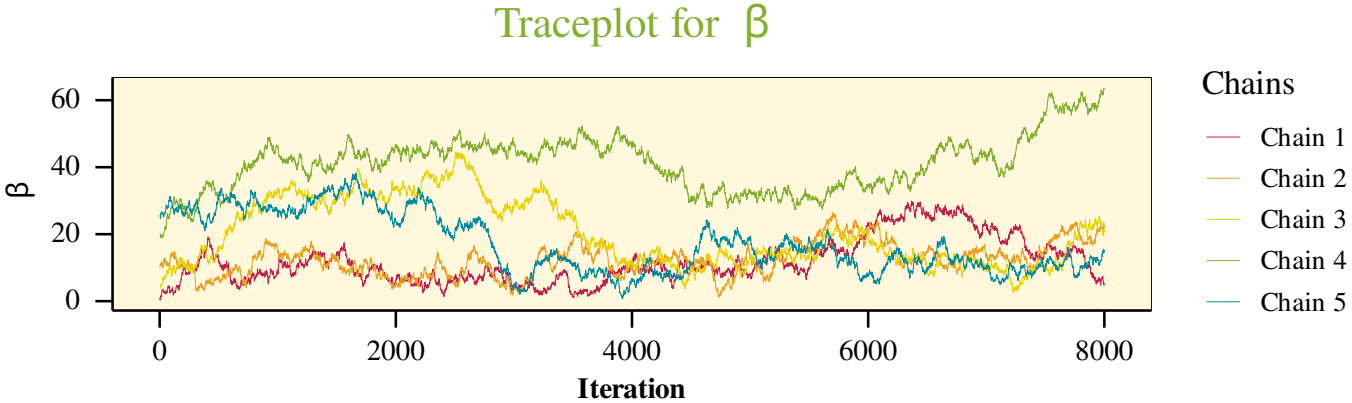
The constant c does not matter because the acceptance rule only uses differences in log posterior values:

$$\log\left(\frac{p(\alpha^\star, \beta^\star|y)}{p(\alpha_t, \beta_t|y)}\right) = \log p(\alpha^\star, \beta^\star|y) - \log p(\alpha_t, \beta_t|y)$$

Procedure for this example

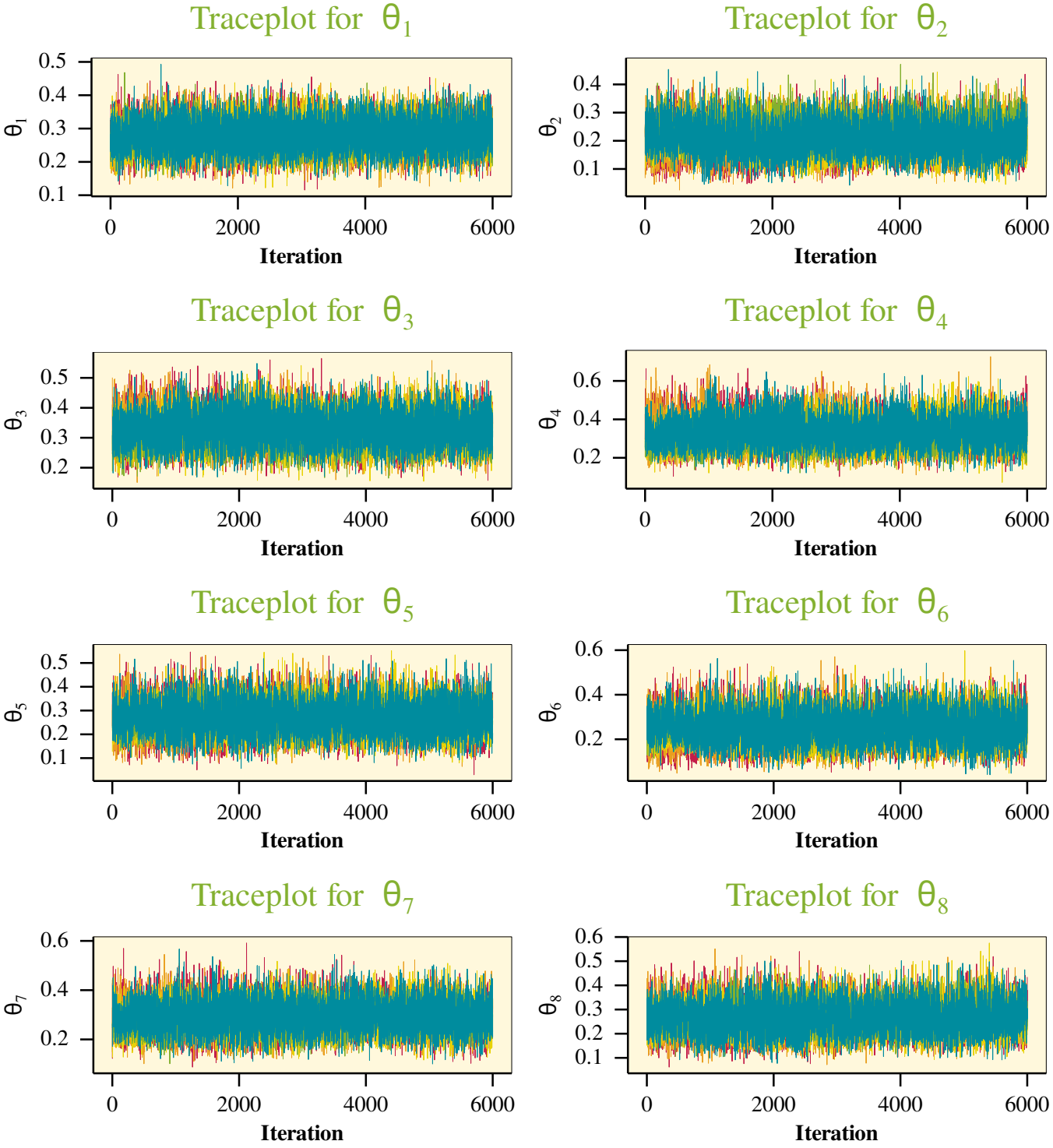
1. Choose starting values for α and β.
2. Propose a new pair $(\alpha^\star, \beta^\star)$ from a proposal distribution.
3. Calculate the posterior density at the proposed and current values:
$$p(\alpha^\star, \beta^\star|y) \qquad \text{and} \qquad p(\alpha_t, \beta_t|y)$$
4. Use the Metropolis-Hastings acceptance probability to decide whether to accept the proposed pair; repeat this many times to get samples of α and β.
5. For each sampled pair $(\alpha^{(s)}, \beta^{(s)})$, sample:
$$\theta_j^{(s)} \sim \text{Beta}(y_j + \alpha^{(s)},\ n_j - y_j + \beta^{(s)})$$
6. Summarize the sampled values of α, β, and each θ_j.





These traceplots show five chains started from different initial values. The point of using multiple chains is that we do not want convergence to be an artifact of one

lucky starting point. If chains that start in different places all wander into the same region and mix together, that is a very good sign. Ideally, traceplots should look like a "Bushy Hedge" rather than a line with a strong trend.



For the θ_j values, the chains mix much more quickly. This makes sense because once we have values for α and β, each θ_j can be sampled directly from its full conditional beta distribution. The hard part of the example is really the posterior for the hyperparameters.

5.3 Gibbs Sampling

The second MCMC method we will use is Gibbs sampling. Gibbs sampling is used when we have multiple unknown parameters and can sample from each full conditional posterior distribution.

For a reusable R framework, see the Gibbs sampling template in the appendix.

Suppose our posterior distribution is:

$$p(\theta_1, \theta_2, \theta_3|y)$$

Sampling from this joint posterior directly may be difficult. However, it may be much easier to sample from:

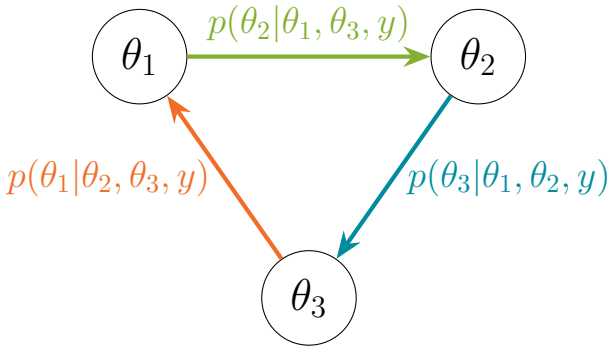
$$p(\theta_1|\theta_2, \theta_3, y)$$
$$p(\theta_2|\theta_1, \theta_3, y)$$
$$p(\theta_3|\theta_1, \theta_2, y)$$

These are called the full conditional posterior distributions. Each one is the posterior distribution of one parameter, conditional on all of the other parameters.

The Procedure The Gibbs sampler works by cycling through the full conditionals:

1. Choose starting values $\theta_1^{(1)}, \theta_2^{(1)}, \theta_3^{(1)}$.
2. For iteration $t = 2, 3, ..., T$:
   (a) Sample: $$\theta_1^{(t)} \sim p(\theta_1|\theta_2^{(t-1)}, \theta_3^{(t-1)}, y)$$
   (b) Sample: $$\theta_2^{(t)} \sim p(\theta_2|\theta_1^{(t)}, \theta_3^{(t-1)}, y)$$
   (c) Sample: $$\theta_3^{(t)} \sim p(\theta_3|\theta_1^{(t)}, \theta_2^{(t)}, y)$$
3. Repeat this many many many times.
4. Remove burn-in and summarize the sampled values.

The diagram for Gibbs sampling is a cleaner cycle since it does not have to split into cases based on if the proposal is accepted or not:



Each arrow represents drawing a new value for one parameter from its full conditional posterior, using the most recently updated values of the other two parameters. Starting from θ_1, a new θ_2 is drawn conditional on the current θ_1 and θ_3; then a new θ_3 is drawn conditional on the updated θ_1 and θ_2; then a new θ_1 is drawn conditional on the updated θ_2 and θ_3. This cycle repeats every iteration, with each parameter update immediately incorporating the latest sampled values of the others.

Notice that after we sample $\theta_1^{(t)}$, we immediately use that new value when sampling $\theta_2^{(t)}$. The sampler always uses the most recent available value of each parameter.

Why are Gibbs proposals always accepted? Gibbs sampling is actually a special case of Metropolis-Hastings. In Metropolis-Hastings, we propose a new value and then accept it with some probability. In Gibbs sampling, the proposal distribution is the full conditional posterior itself.

Suppose we are updating θ_1. Then:

$$q(\theta_1^\star|\theta_1^{(t)}) = p(\theta_1^\star|\theta_2, \theta_3, y)$$

The Metropolis-Hastings acceptance probability becomes:

$$\alpha(\theta_1^\star|\theta_1^{(t)}) = \min\left(1,\ \frac{p(\theta_1^\star, \theta_2, \theta_3|y)q(\theta_1^{(t)}|\theta_1^\star)}{p(\theta_1^{(t)}, \theta_2, \theta_3|y)q(\theta_1^\star|\theta_1^{(t)})}\right) \quad \text{using} \quad p(\theta_1|\theta_2, \theta_3, y) = \frac{p(\theta_1, \theta_2, \theta_3|y)}{p(\theta_2, \theta_3|y)}$$

the full conditional terms cancel out perfectly, leaving:

$$\alpha(\theta_1^\star|\theta_1^{(t)}) = \min(1, 1) = 1$$

Thus, in Gibbs sampling, every proposed value is accepted. This is extremely convenient, but it comes with one major requirement:

We must be able to sample from all of the full conditional posterior distributions.

Example: Normal Model with Unknown Mean and Variance

Recall from Chapter 3 that if:

$$y_1, ..., y_n | \mu, \sigma^2 \sim N(\mu, \sigma^2)$$

with the uninformative prior:

$$p(\mu, \sigma^2) \propto \frac{1}{\sigma^2}$$

Let's derive the full conditional posterior distributions from the joint posterior. The likelihood and prior give:

$$\begin{aligned} p(\mu, \sigma^2 | y) &\propto p(y | \mu, \sigma^2) p(\mu, \sigma^2) \\ &\propto \left[\prod_{i=1}^{n} (\sigma^2)^{-1/2} \exp\left\{ -\frac{(y_i - \mu)^2}{2\sigma^2} \right\} \right] \frac{1}{\sigma^2} \\ &\propto (\sigma^2)^{-(\frac{n}{2}+1)} \exp\left\{ -\frac{\sum_{i=1}^{n} (y_i - \mu)^2}{2\sigma^2} \right\} \end{aligned}$$

To find the full conditional for μ, hold σ^2 fixed and keep only terms involving μ:

$$\begin{aligned} p(\mu | \sigma^2, y) &\propto \exp\left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mu)^2 \right\} \\ &\propto \exp\left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left[(y_i - \bar{y}) + (\bar{y} - \mu) \right]^2 \right\} \\ &\propto \exp\left\{ -\frac{1}{2\sigma^2} \left[\sum_{i=1}^{n} (y_i - \bar{y})^2 + 2(\bar{y} - \mu) \sum_{i=1}^{n} (y_i - \bar{y}) + n(\bar{y} - \mu)^2 \right] \right\} \\ &\propto \exp\left\{ -\frac{(\mu - \bar{y})^2}{2\sigma^2/n} \right\} \quad \Longrightarrow \quad \mu | \sigma^2, y \sim N\left(\bar{y}, \frac{\sigma^2}{n} \right) \end{aligned}$$

To find the full conditional for σ^2, hold μ fixed and keep only terms involving σ^2:

$$\begin{aligned} p(\sigma^2 | \mu, y) &\propto (\sigma^2)^{-(\frac{n}{2}+1)} \exp\left\{ -\frac{\sum_{i=1}^{n} (y_i - \mu)^2}{2\sigma^2} \right\} \\ &\Longrightarrow \quad \sigma^2 | \mu, y \sim \text{Inv-Gamma}\left(\frac{n}{2}, \frac{\sum_{i=1}^{n} (y_i - \mu)^2}{2} \right) \end{aligned}$$

Thus, the full conditional posterior distributions are:

$$\mu | \sigma^2, y \sim N\left(\bar{y}, \frac{\sigma^2}{n} \right)$$

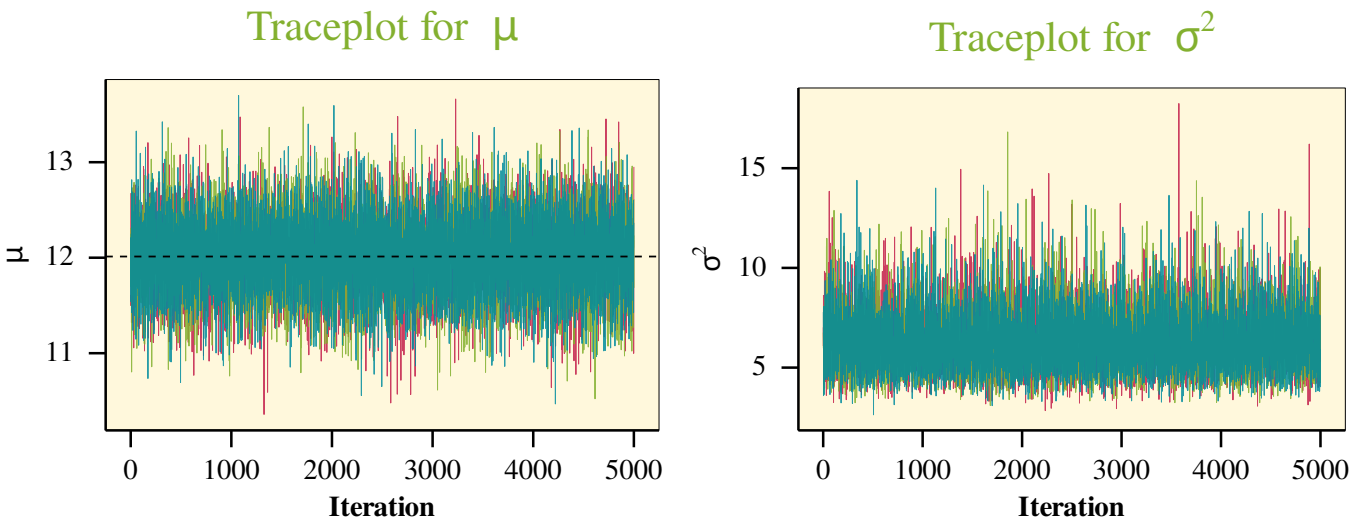
$$\sigma^2 | \mu, y \sim \text{Inv-Gamma}\left(\frac{n}{2}, \frac{\sum_{i=1}^{n} (y_i - \mu)^2}{2} \right)$$

This gives us an immediate Gibbs sampler:

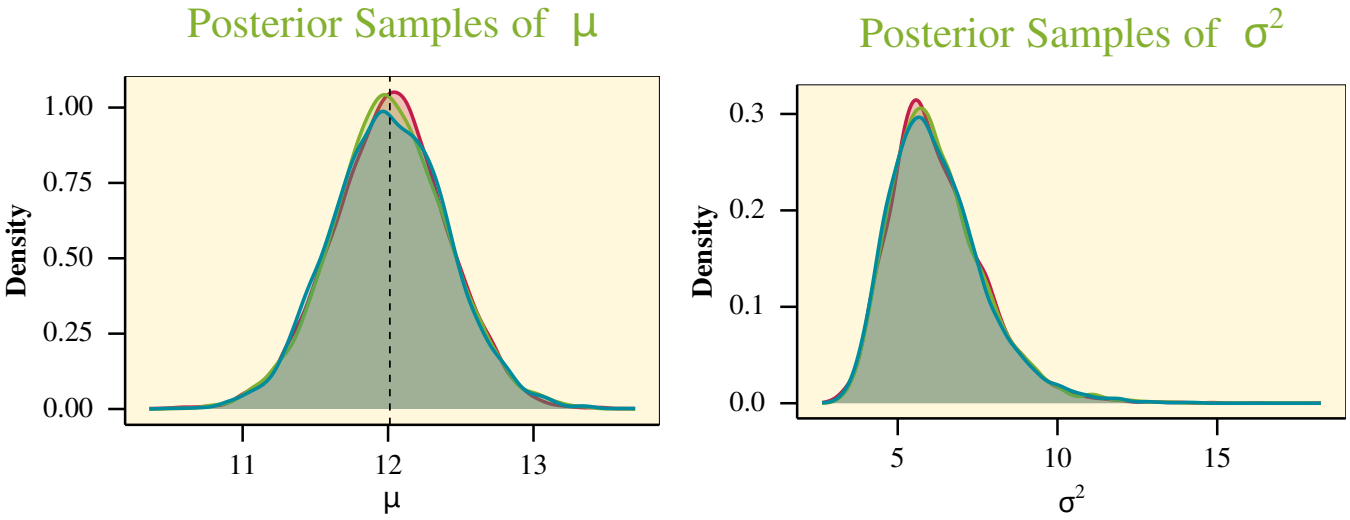
1. Choose starting values $\mu^{(1)}$ and $(\sigma^2)^{(1)}$.
2. Sample:
$$\mu^{(t)} \sim N\left(\bar{y}, \frac{(\sigma^2)^{(t-1)}}{n}\right)$$
3. Sample:
$$(\sigma^2)^{(t)} \sim \text{Inv-Gamma}\left(\frac{n}{2}, \frac{\sum_{i=1}^{n}(y_i - \mu^{(t)})^2}{2}\right)$$
4. Repeat steps 2 and 3.

This is a nice example because we can see exactly what Gibbs sampling is doing: it alternates between sampling μ while pretending σ^2 is known, and sampling σ^2 while pretending μ is known. Neither parameter is actually known, of course, but the sampler keeps updating them until the joint collection of samples approximates the joint posterior distribution.

To see what this looks like, suppose we observe $n = 40$ values from a normal population. The Gibbs sampler below starts three chains from different initial values and then alternates between sampling μ and σ^2 from the two full conditionals above.



The traceplots show the chains quickly moving away from their different starting values and then mixing in the same stable region. This is our first convergence check: after burn-in, the chains should look like stable bands with no strong upward or downward trend. Once we are comfortable that the chains are sampling from a common region, we can ignore the order of the draws and look at their distribution using density plots.



The density plots show the posterior uncertainty after burn-in. The posterior for μ is concentrated around the observed sample mean, while the posterior for σ^2 remains more right-skewed because variance parameters are positive and often have asymmetric posterior distributions. The overlap among the chains is also reassuring: it suggests the posterior summaries are not being driven by one particular starting value.

Example: Regression Gibbs Sampler In the MCMC examples, we also considered a regression model:

$$y_i = \alpha + \beta x_i + \epsilon_i \qquad \text{with:} \qquad y_i|\alpha, \beta, \sigma^2 \sim N(\alpha + \beta x_i, \sigma^2)$$

We will use the same uninformative prior style as before:

$$p(\alpha, \beta, \sigma^2|x) \propto \frac{1}{\sigma^2}$$

The joint posterior is:

$$\begin{aligned} p(\alpha, \beta, \sigma^2|x, y) &\propto p(y|\alpha, \beta, \sigma^2, x)p(\alpha, \beta, \sigma^2|x) \\ &\propto \left[\prod_{i=1}^{n} (\sigma^2)^{-1/2} \exp\left\{-\frac{(y_i - \alpha - \beta x_i)^2}{2\sigma^2}\right\}\right] \frac{1}{\sigma^2} \\ &\propto (\sigma^2)^{-(\frac{n}{2}+1)} \exp\left\{-\frac{\sum_{i=1}^{n}(y_i - \alpha - \beta x_i)^2}{2\sigma^2}\right\} \end{aligned}$$

Now we find each full conditional by holding the other parameters fixed.

For σ^2, hold α and β fixed:

$$\begin{aligned} p(\sigma^2|\alpha, \beta, x, y) &\propto (\sigma^2)^{-(\frac{n}{2}+1)} \exp\left\{-\frac{\sum_{i=1}^{n}(y_i - \alpha - \beta x_i)^2}{2\sigma^2}\right\} \\ \implies \quad \sigma^2|\alpha, \beta, x, y &\sim \text{Inv-Gamma}\left(\frac{n}{2}, \frac{\sum_{i=1}^{n}(y_i - \alpha - \beta x_i)^2}{2}\right) \end{aligned}$$

For α, only the residual sum of squares matters:

$$\begin{aligned} p(\alpha|\beta, \sigma^2, x, y) &\propto \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \alpha - \beta x_i)^2\right\} \\ &= \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left[(y_i - \beta x_i) - \alpha\right]^2\right\} \\ &\propto \exp\left\{-\frac{1}{2\sigma^2}\left[n\alpha^2 - 2\alpha\sum_{i=1}^{n}(y_i - \beta x_i)\right]\right\} \\ &\propto \exp\left\{-\frac{n}{2\sigma^2}\left[\alpha^2 - 2\alpha(\bar{y} - \beta\bar{x})\right]\right\} \\ &\propto \exp\left\{-\frac{n}{2\sigma^2}\left[\alpha - (\bar{y} - \beta\bar{x})\right]^2\right\} \\ \implies \quad \alpha|\beta, \sigma^2, x, y &\sim N\left(\bar{y} - \beta\bar{x}, \frac{\sigma^2}{n}\right) \end{aligned}$$

For β, again keep only terms involving β:

$$\begin{aligned}
p(\beta|\alpha,\sigma^2,x,y) &\propto \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\alpha-\beta x_i)^2\right\} \\
&\propto \exp\left\{-\frac{1}{2\sigma^2}\left[\beta^2\sum_{i=1}^{n}x_i^2-2\beta\sum_{i=1}^{n}x_i(y_i-\alpha)\right]\right\} \\
&\propto \exp\left\{-\frac{\sum_{i=1}^n x_i^2}{2\sigma^2}\left[\beta-\frac{\sum_{i=1}^n x_i(y_i-\alpha)}{\sum_{i=1}^n x_i^2}\right]^2\right\} \\
&= \exp\left\{-\frac{\sum_{i=1}^n x_i^2}{2\sigma^2}\left[\beta-\frac{\sum_{i=1}^n x_iy_i-n\alpha\bar{x}}{\sum_{i=1}^n x_i^2}\right]^2\right\} \\
&\implies \quad \beta|\alpha,\sigma^2,x,y \sim N\left(\frac{\sum_{i=1}^n x_iy_i-n\alpha\bar{x}}{\sum_{i=1}^n x_i^2},\frac{\sigma^2}{\sum_{i=1}^n x_i^2}\right)
\end{aligned}$$

Putting these together, the full conditional posterior distributions are:

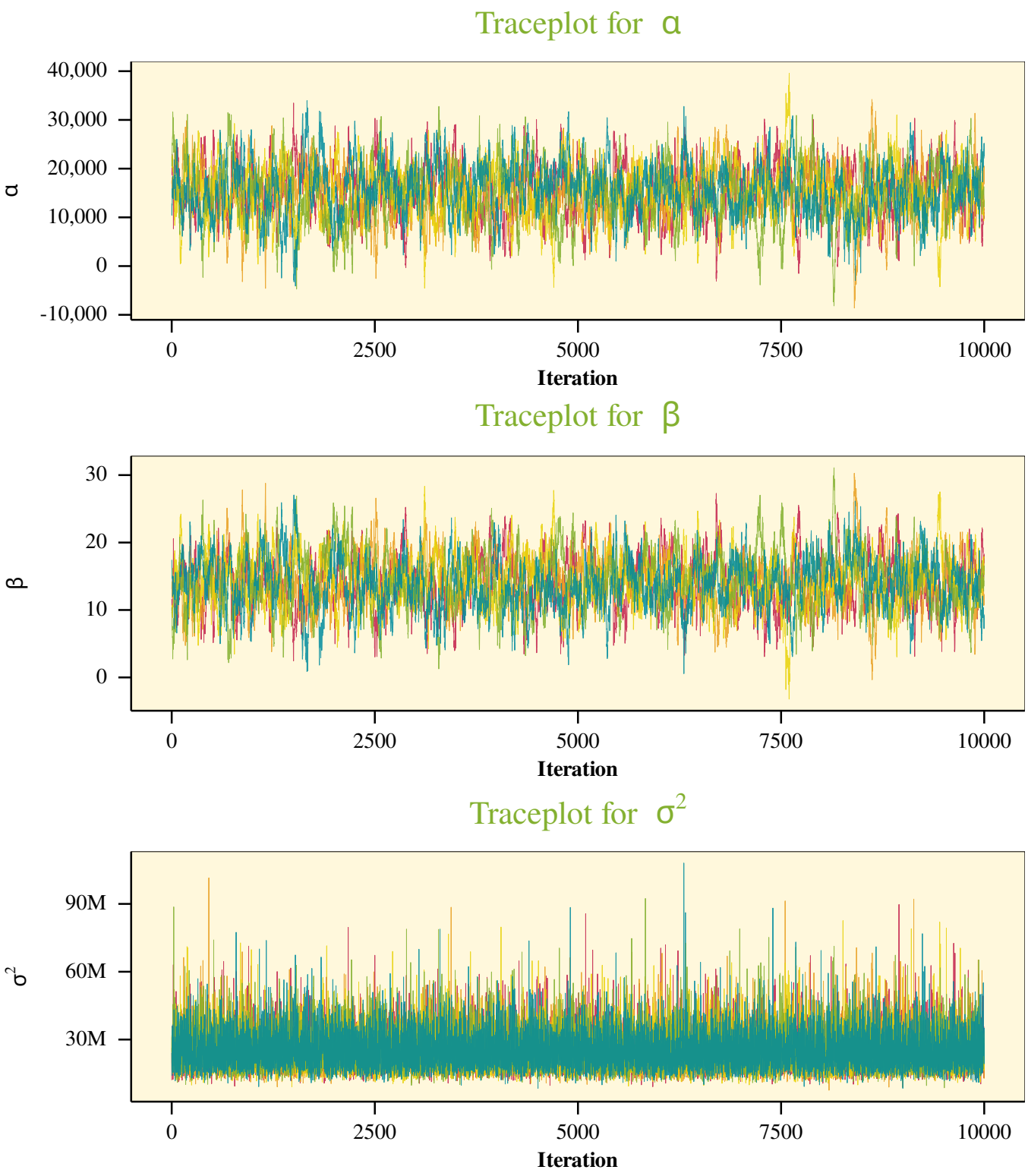
$$\alpha|\beta,\sigma^2,x,y \sim N\left(\bar{y}-\beta\bar{x},\frac{\sigma^2}{n}\right)$$

$$\beta|\alpha,\sigma^2,x,y \sim N\left(\frac{\sum_{i=1}^n x_iy_i-n\alpha\bar{x}}{\sum_{i=1}^n x_i^2},\frac{\sigma^2}{\sum_{i=1}^n x_i^2}\right)$$

$$\sigma^2|\alpha,\beta,x,y \sim \text{Inv-Gamma}\left(\frac{n}{2},\frac{\sum_{i=1}^n(y_i-\alpha-\beta x_i)^2}{2}\right)$$

So the procedure is:

1. Start with initial values for α, β, and σ^2.
2. Sample α from its normal full conditional using the current values of β and σ^2.
3. Sample β from its normal full conditional using the newest value of α and the current value of σ^2.
4. Sample σ^2 from its inverse-gamma full conditional using the newest values of α and β.
5. Repeat until we have a large sample from the joint posterior.



Here, each parameter is moving around a stable region rather than drifting upward or downward across the whole chain. The five chains also overlap after burn-in, which is what we want to see. If the traceplot had a strong trend, or if different chains were stuck in noticeably different regions, then we would be much more suspicious.

Metropolis-Hastings vs. Gibbs It is helpful to keep the two methods straight:

Method	What we need	Acceptance Step?
Metropolis-Hastings	Unnormalized posterior and proposal distribution	Yes
Gibbs Sampling	Full conditional posteriors we can sample from	No; always accept

We can also combine these methods. If one parameter has a convenient full conditional, we can update it using Gibbs. If another parameter does not have a convenient full conditional, we can update it using Metropolis-Hastings. This is common in hierarchical models.

Overall Summary:

1. MCMC is used when the posterior distribution is not easy to sample from directly.
2. Metropolis-Hastings proposes a candidate value and accepts or rejects it.
3. The acceptance probability only needs the unnormalized posterior.
4. Gibbs sampling cycles through full conditional posterior distributions.
5. Once we have posterior samples, we summarize them the same way we have summarized posterior samples all semester.

Appendix: R Code Templates

Metropolis-Hastings Template

```r
# 1. Define the log unnormalized posterior.
log_posterior <- function(theta, data) {
  # Replace this with: log_likelihood(theta, data) + log_prior(theta)
  log_lik <- 0
  log_prior <- 0
  log_lik + log_prior
}

n_iter <- 10000 # 2. Choose MCMC settings.
burn_in <- 1000
proposal_sd <- 0.25

theta <- numeric(n_iter)
accepted <- logical(n_iter)
theta[1] <- 0

for (t in 2:n_iter) { # 3. Run the Metropolis-Hastings chain.
  current <- theta[t - 1]

  # Symmetric random-walk proposal:
  # theta_star | theta_current ~ Normal(theta_current, proposal_sd^2)
  theta_star <- rnorm(1, mean = current, sd = proposal_sd)

  log_alpha <- log_posterior(theta_star, data) -
    log_posterior(current, data)

  if (log(runif(1)) <= log_alpha) {
    theta[t] <- theta_star
    accepted[t] <- TRUE
  } else {
    theta[t] <- current
  }
}

# 4. Remove burn-in and summarize the posterior samples.
posterior_draws <- theta[(burn_in + 1):n_iter]
acceptance_rate <- mean(accepted)

mean(posterior_draws)
quantile(posterior_draws, c(0.025, 0.5, 0.975))
acceptance_rate
```

Use this template when you can evaluate the posterior density up to proportionality, but cannot sample from the posterior directly. The function `log_posterior()` should

return the log of the unnormalized posterior density. Working on the log scale is usually more stable than multiplying many small likelihood and prior values directly.

The proposal step above uses a symmetric normal random walk, so the proposal densities cancel in the acceptance probability. If you use an asymmetric proposal, add the forward and reverse proposal log densities to the acceptance ratio.

To use the template, start by writing `log_posterior()` for your model. Then choose a starting value, number of iterations, burn-in, and proposal standard deviation. After running the chain, check the acceptance rate and a traceplot. If the chain barely moves, the proposal standard deviation may be too small. If it rejects almost everything, the proposal standard deviation may be too large.

Gibbs Sampling Template

```r
# 1. Write one sampler for each full conditional distribution.
sample_theta1 <- function(theta2, theta3, data) {
  rnorm(1, mean = 0, sd = 1)
} # Replace this with a draw from:  p(theta1 | theta2, theta3, data)

sample_theta2 <- function(theta1, theta3, data) {
  rnorm(1, mean = 0, sd = 1)
} # Replace this with a draw from: p(theta2 | theta1, theta3, data)

sample_theta3 <- function(theta1, theta2, data) {
  rgamma(1, shape = 1, rate = 1)
} # Replace this with a draw from: p(theta3 | theta1, theta2, data)

n_iter <- 10000 # 2. Choose MCMC settings.
burn_in <- 1000

# Initialize matrix:
# (rows: 1 = theta1, 2 = theta2, 3 = theta3; columns: iterations)
theta <- matrix(0, nrow = 3, ncol = n_iter)

theta[1, 1] <- 0
theta[2, 1] <- 0
theta[3, 1] <- 1

for (t in 2:n_iter) { # 3. Run the Gibbs sampler.
  theta[1, t] <- sample_theta1(theta[2, t - 1], theta[3, t - 1], data)
  theta[2, t] <- sample_theta2(theta[1, t],     theta[3, t - 1], data)
  theta[3, t] <- sample_theta3(theta[1, t],     theta[2, t],     data)
}

# 4. Remove burn-in and collect the posterior samples.
posterior_draws <- as.data.frame(t(theta[, (burn_in + 1):n_iter]))
colnames(posterior_draws) <- c("theta1", "theta2", "theta3")

colMeans(posterior_draws)
apply(posterior_draws, 2, quantile, probs = c(0.025, 0.5, 0.975))
```

Use this template when every unknown parameter has a full conditional posterior distribution that you can sample from directly. Each helper function should return one new draw from its full conditional distribution, using the most recently available values of the other parameters.

To use the template, derive the full conditional posterior for each unknown parameter and replace the placeholder sampler functions. The order matters: after sampling `theta1[t]`, the update for `theta2[t]` uses that new value immediately. After burn-in, summarize the columns of `posterior_draws` and inspect traceplots for each parameter.