5.3 Gibbs Sampler

Full conditionals, algorithm, and a hierarchical exponential example

When can we use a Gibbs sampler?

Key requirement

We can use a Gibbs sampler when we can **sample directly** from the **full conditional distributions**.

When can we use a Gibbs sampler?

Key requirement

We can use a Gibbs sampler when we can **sample directly** from the **full conditional distributions**.

Suppose our parameter vector is

$$\boldsymbol{\theta} = (\theta_1, \ldots, \theta_N), \qquad \text{target: } \pi(\theta_1, \ldots, \theta_N \,|\, y).$$

When can we use a Gibbs sampler?

Key requirement

We can use a Gibbs sampler when we can **sample directly** from the **full conditional distributions**.

Suppose our parameter vector is

$$\boldsymbol{\theta} = (\theta_1, \ldots, \theta_N), \qquad \text{target: } \pi(\theta_1, \ldots, \theta_N \,|\, y).$$

If we can derive (and sample from) each conditional

$$\pi(\theta_1 \,|\, \theta_2, \ldots, \theta_N, y),\ \pi(\theta_2 \,|\, \theta_1, \theta_3, \ldots, \theta_N, y),\ \ldots,\ \pi(\theta_N \,|\, \theta_1, \ldots, \theta_{N-1}, y),$$

then we can build an MCMC algorithm with no accept/reject step.

Full conditionals: the practical rule

How to derive a full conditional

Start from the joint posterior $\pi(\boldsymbol{\theta} \mid y)$ and **keep only terms that depend on the parameter of interest**.

Everything else is a multiplicative constant and can be absorbed into $\propto$.

Full conditionals: the practical rule

How to derive a full conditional

Start from the joint posterior $\pi(\boldsymbol{\theta} \mid y)$ and **keep only terms that depend on the parameter of interest**.
Everything else is a multiplicative constant and can be absorbed into $\propto$.

- For $\pi(\theta_j \mid \theta_{-j}, y)$, keep terms involving θ_j.
- Then **match the kernel** to a standard distribution (Normal, Gamma, Beta, ...).

Gibbs sampler: algorithm (one sweep)

Step 1 (initialise). Choose initial values

$$\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \ldots, \theta_N^{(0)}).$$

Gibbs sampler: algorithm (one sweep)

Step 1 (initialise). Choose initial values

$$\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \ldots, \theta_N^{(0)}).$$

Step 2. Set $i = 1$.

Gibbs sampler: algorithm (one sweep)

Step 1 (initialise). Choose initial values

$$\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \ldots, \theta_N^{(0)}).$$

Step 2. Set $i = 1$.

One Gibbs sweep at iteration i

1. Draw
$$\theta_1^{(i)} \sim \pi\left(\theta_1 \,\middle|\, \theta_2^{(i-1)}, \ldots, \theta_N^{(i-1)}, y\right).$$
2. Draw
$$\theta_2^{(i)} \sim \pi\left(\theta_2 \,\middle|\, \theta_1^{(i)}, \theta_3^{(i-1)}, \ldots, \theta_N^{(i-1)}, y\right).$$
3. $\vdots$
4. Draw
$$\theta_N^{(i)} \sim \pi\left(\theta_N \,\middle|\, \theta_1^{(i)}, \theta_2^{(i)}, \ldots, \theta_{N-1}^{(i)}, y\right).$$

The key idea: "most recent values"

When updating θ_j at iteration i:

- Use **updated** values $\theta_1^{(i)}, \ldots, \theta_{j-1}^{(i)}$ (already sampled in this sweep),
- and use **previous** values $\theta_{j+1}^{(i-1)}, \ldots, \theta_N^{(i-1)}$ (not yet updated),
- always conditioning on the data y.

The key idea: "most recent values"

When updating θ_j at iteration i:

- Use **updated** values $\theta_1^{(i)}, \ldots, \theta_{j-1}^{(i)}$ (already sampled in this sweep),
- and use **previous** values $\theta_{j+1}^{(i-1)}, \ldots, \theta_N^{(i-1)}$ (not yet updated),
- always conditioning on the data y.

Repeat

Repeat the sweep for $i = 2, \ldots, M$ to obtain M iterations of the Markov chain.

Pseudocode: generic Gibbs sampler

```
M # number of iterations
N # number of parameters
theta.store <- matrix(NA, N, M)
theta <- numeric(N) # current state (theta_1,...,theta_N)

# initialise theta[1:N] somehow
# theta <- ...

for (iter in 1:M) {
  for (i in 1:N) {
    # update theta[i] by sampling from its full conditional
    # theta[i] <- sample_from_full_conditional(i, theta[-i], y)
  }
  theta.store[, iter] <- theta # store current values
}
```

Pseudocode: generic Gibbs sampler

```
M # number of iterations
N # number of parameters
theta.store <- matrix(NA, N, M)
theta <- numeric(N) # current state (theta_1,...,theta_N)

# initialise theta[1:N] somehow
# theta <- ...

for (iter in 1:M) {
  for (i in 1:N) {
    # update theta[i] by sampling from its full conditional
    # theta[i] <- sample_from_full_conditional(i, theta[-i], y)
  }
  theta.store[, iter] <- theta # store current values
}
```

- Outer loop: iterations of the Markov chain (M).
- Inner loop: one sweep through parameters ($1, \ldots, N$).

Example 5.8: hierarchical exponential model

We consider the hierarchical model (from earlier examples):

$$Y_1, \ldots, Y_N \,\big|\, (\lambda, \gamma) \text{ i.i.d. } \sim \text{Exp}(\lambda) \qquad \text{(likelihood)}$$

$$\lambda \,\big|\, \gamma \sim \text{Exp}(\gamma) \qquad \text{(prior)}$$

$$\gamma \sim \text{Exp}(\nu) \qquad \text{(hyperprior)}.$$

Example 5.8: hierarchical exponential model

We consider the hierarchical model (from earlier examples):

$$Y_1, \ldots, Y_N \,\big|\, (\lambda, \gamma) \text{ i.i.d. } \sim \text{Exp}(\lambda) \qquad \text{(likelihood)}$$

$$\lambda \,\big|\, \gamma \sim \text{Exp}(\gamma) \qquad \text{(prior)}$$

$$\gamma \sim \text{Exp}(\nu) \qquad \text{(hyperprior)}.$$

Goal

Use a Gibbs sampler to generate samples from the joint posterior $\pi(\lambda, \gamma \,|\, y)$.

Posterior kernel (up to proportionality)

Using Bayes:

$$\pi(\lambda, \gamma \,|\, y) \propto \pi(y \,|\, \lambda)\, \pi(\lambda \,|\, \gamma)\, \pi(\gamma).$$

Posterior kernel (up to proportionality)

Using Bayes:

$$\pi(\lambda, \gamma \,|\, y) \propto \pi(y \,|\, \lambda)\, \pi(\lambda \,|\, \gamma)\, \pi(\gamma).$$

Likelihood term:

$$\pi(y \,|\, \lambda) = \prod_{i=1}^{N} \lambda e^{-\lambda y_i} = \lambda^N \exp\left(-\lambda \sum_{i=1}^{N} y_i\right).$$

Posterior kernel (up to proportionality)

Using Bayes:

$$\pi(\lambda, \gamma \,|\, y) \propto \pi(y \,|\, \lambda)\, \pi(\lambda \,|\, \gamma)\, \pi(\gamma).$$

Likelihood term:

$$\pi(y \,|\, \lambda) = \prod_{i=1}^{N} \lambda e^{-\lambda y_i} = \lambda^N \exp\left(-\lambda \sum_{i=1}^{N} y_i\right).$$

Prior and hyperprior:

$$\pi(\lambda \,|\, \gamma) = \gamma e^{-\gamma \lambda}, \qquad \pi(\gamma) = \nu e^{-\nu \gamma}.$$

Posterior kernel (up to proportionality)

Using Bayes:

$$\pi(\lambda, \gamma \,|\, y) \propto \pi(y \,|\, \lambda)\, \pi(\lambda \,|\, \gamma)\, \pi(\gamma).$$

Likelihood term:

$$\pi(y \,|\, \lambda) = \prod_{i=1}^{N} \lambda e^{-\lambda y_i} = \lambda^N \exp\left(-\lambda \sum_{i=1}^{N} y_i\right).$$

Prior and hyperprior:

$$\pi(\lambda \,|\, \gamma) = \gamma e^{-\gamma \lambda}, \qquad \pi(\gamma) = \nu e^{-\nu \gamma}.$$

So the posterior kernel is

$$\pi(\lambda, \gamma \,|\, y) \propto \lambda^N e^{-\lambda \sum y_i} \cdot \gamma e^{-\gamma \lambda} \cdot \nu e^{-\nu \gamma}.$$

Full conditional for λ

Keep only terms involving λ:

$$\pi(\lambda \,|\, y, \gamma) \propto \lambda^N \exp\left(-\lambda \sum_{i=1}^{N} y_i\right) \exp(-\gamma\lambda).$$

Full conditional for λ

Keep only terms involving λ:

$$\pi(\lambda \mid y, \gamma) \propto \lambda^N \exp\left(-\lambda \sum_{i=1}^{N} y_i\right) \exp(-\gamma\lambda).$$

Combine the exponent:

$$\pi(\lambda \mid y, \gamma) \propto \lambda^N \exp\left(-\lambda\Big(\gamma + \sum_{i=1}^{N} y_i\Big)\right).$$

Full conditional for λ

Keep only terms involving λ:

$$\pi(\lambda \mid y, \gamma) \propto \lambda^N \exp\left(-\lambda \sum_{i=1}^{N} y_i\right) \exp(-\gamma\lambda).$$

Combine the exponent:

$$\pi(\lambda \mid y, \gamma) \propto \lambda^N \exp\left(-\lambda\Big(\gamma + \sum_{i=1}^{N} y_i\Big)\right).$$

Distribution matching

This is a Gamma kernel:

$$\lambda \mid y, \gamma \sim \text{Gamma}\Big(N+1,\ \gamma + \sum_{i=1}^{N} y_i\Big),$$

(shape–rate parameterisation)

Plugging in the example numbers

For the specific values $N = 10$ and $\sum_{i=1}^{N} y_i = 95$:

$$\pi(\lambda \,|\, y, \gamma) \propto \lambda^{10} \exp\big(- \lambda(95 + \gamma)\big),$$

so

$$\lambda \,|\, y, \gamma \sim \text{Gamma}(11,\ 95 + \gamma).$$

Full conditional for γ

Keep only terms involving γ:

$$\pi(\gamma \mid y, \lambda) \propto \gamma\, e^{-\gamma\lambda}\, e^{-\nu\gamma}.$$

Full conditional for γ

Keep only terms involving γ:

$$\pi(\gamma \,|\, y, \lambda) \propto \gamma\, e^{-\gamma\lambda}\, e^{-\nu\gamma}.$$

Combine exponent:

$$\pi(\gamma \,|\, y, \lambda) \propto \gamma \exp\big(-(\nu+\lambda)\gamma\big).$$

Full conditional for γ

Keep only terms involving γ:

$$\pi(\gamma \,|\, y, \lambda) \propto \gamma\, e^{-\gamma\lambda}\, e^{-\nu\gamma}.$$

Combine exponent:

$$\pi(\gamma \,|\, y, \lambda) \propto \gamma \exp\big(-(\nu+\lambda)\gamma\big).$$

Distribution matching

This is also Gamma:

$$\gamma \,|\, y, \lambda \sim \text{Gamma}(2,\ \lambda+\nu),$$

(shape–rate).

Full conditional for γ

Keep only terms involving γ:

$$\pi(\gamma \mid y, \lambda) \propto \gamma\, e^{-\gamma\lambda}\, e^{-\nu\gamma}.$$

Combine exponent:

$$\pi(\gamma \mid y, \lambda) \propto \gamma \exp\big(-(\nu + \lambda)\gamma\big).$$

Distribution matching

This is also Gamma:

$$\gamma \mid y, \lambda \sim \text{Gamma}(2,\ \lambda + \nu),$$

(shape–rate).

- Note: this is sometimes described informally as "like an exponential", but the leading γ makes it Gamma with shape 2.

Gibbs sampler for (λ, γ)

Algorithm

1. Set initial values $\{\lambda^{(0)}, \gamma^{(0)}\}$.
2. Set $i = 1$.
3. Draw
$$\lambda^{(i)} \,\big|\, y, \gamma^{(i-1)} \sim \text{Gamma}\Big(N + 1,\ \sum_{k=1}^{N} y_k + \gamma^{(i-1)}\Big).$$
4. Draw
$$\gamma^{(i)} \,\big|\, y, \lambda^{(i)} \sim \text{Gamma}\big(2,\ \lambda^{(i)} + \nu\big).$$
5. Repeat steps 3–4 for $i = 2, \ldots, n_{\text{iter}}$.

Gibbs sampler for (λ, γ)

Algorithm

1. Set initial values $\{\lambda^{(0)}, \gamma^{(0)}\}$.
2. Set $i = 1$.
3. Draw
$$\lambda^{(i)} \,\big|\, y, \gamma^{(i-1)} \sim \text{Gamma}\Big(N + 1,\ \sum_{k=1}^{N} y_k + \gamma^{(i-1)}\Big).$$
4. Draw
$$\gamma^{(i)} \,\big|\, y, \lambda^{(i)} \sim \text{Gamma}\big(2,\ \lambda^{(i)} + \nu\big).$$
5. Repeat steps 3–4 for $i = 2, \ldots, n_{\text{iter}}$.

- Again: γ uses the updated value $\lambda^{(i)}$.

R code: set up storage

```
# Set Up MCMC Algorithm ---------------------------------------------------
nu <- 0.01
n.iter <- 10000

lambda.store <- numeric(n.iter) # store lambda at each iteration
gamma.store <- numeric(n.iter) # store gamma at each iteration

# (optional) initial values:
lambda.store[1] <- 0.1
gamma.store[1] <- 1.0
```

R code: set up storage

```
# Set Up MCMC Algorithm ------------------------------------------------
nu <- 0.01
n.iter <- 10000

lambda.store <- numeric(n.iter) # store lambda at each iteration
gamma.store <- numeric(n.iter) # store gamma at each iteration

# (optional) initial values:
lambda.store[1] <- 0.1
gamma.store[1] <- 1.0
```

- You must initialise the chain (even if the example code starts at i=2).
- After that, each iteration updates both parameters once.

R code: run Gibbs updates

Here we use the example-specific values $N = 10$ and $\sum y_i = 95$, so $\lambda \,|\, \gamma, y \sim \Gamma(11, 95 + \gamma)$ and $\gamma \,|\, \lambda, y \sim \Gamma(2, \nu + \lambda)$.

```
# Run MCMC Algorithm ------------------------------------------------------
for (i in 2:n.iter) {

  # 1) update lambda | gamma, y
  lambda.store[i] <- rgamma(1, shape = 11, rate = 95 + gamma.store[i-1])

  # 2) update gamma | lambda, y
  gamma.store[i] <- rgamma(1, shape = 2, rate = nu + lambda.store[i])

}
```

R code: run Gibbs updates

Here we use the example-specific values $N = 10$ and $\sum y_i = 95$, so $\lambda \,|\, \gamma, y \sim \Gamma(11, 95 + \gamma)$ and $\gamma \,|\, \lambda, y \sim \Gamma(2, \nu + \lambda)$.

```
# Run MCMC Algorithm ------------------------------------------------------
for (i in 2:n.iter) {

  # 1) update lambda | gamma, y
  lambda.store[i] <- rgamma(1, shape = 11, rate = 95 + gamma.store[i-1])

  # 2) update gamma | lambda, y
  gamma.store[i] <- rgamma(1, shape = 2, rate = nu + lambda.store[i])

}
```

- In R: `rgamma(n, shape, rate)` uses **rate** (not scale).

Trace plots: checking mixing

Trace plots show the Markov chain values over iterations.

```
# Plot trace plot (Markov chain values)
plot(lambda.store, type = 'l',
     xlab = "iteration", ylab = expression(lambda))

plot(gamma.store, type = 'l',
     xlab = "iteration", ylab = expression(gamma))
```

Trace plots: checking mixing

Trace plots show the Markov chain values over iterations.

```
# Plot trace plot (Markov chain values)
plot(lambda.store, type = 'l',
     xlab = "iteration", ylab = expression(lambda))

plot(gamma.store, type = 'l',
     xlab = "iteration", ylab = expression(gamma))
```

- We want a stable "hairy caterpillar" look (after burn-in).
- Large drifts or long flat regions suggest poor mixing.

Posterior density and summaries

Use histograms (or kernel density estimates) for marginal posteriors:

```
# Plot posterior density
hist(lambda.store, prob = TRUE,
     xlab = expression(lambda),
     main = "Posterior density of lambda")

mean(lambda.store) # posterior mean
quantile(lambda.store, c(0.025, 0.975)) # 95% credible interval

hist(gamma.store, prob = TRUE,
     xlab = expression(gamma),
     main = "Posterior density of gamma")
```

Posterior density and summaries

Use histograms (or kernel density estimates) for marginal posteriors:

```
# Plot posterior density
hist(lambda.store, prob = TRUE,
     xlab = expression(lambda),
     main = "Posterior density of lambda")

mean(lambda.store) # posterior mean
quantile(lambda.store, c(0.025, 0.975)) # 95% credible interval

hist(gamma.store, prob = TRUE,
     xlab = expression(gamma),
     main = "Posterior density of gamma")
```

- Posterior mean $\approx$ average of stored samples.
- Credible interval $\approx$ sample quantiles.

Investigating dependence: scatter plot

We can investigate dependence between parameters using a scatter plot:

```
# Investigate correlation between parameters
plot(lambda.store, gamma.store,
     xlab = expression(lambda), ylab = expression(gamma))
```

Investigating dependence: scatter plot

We can investigate dependence between parameters using a scatter plot:

```
# Investigate correlation between parameters
plot(lambda.store, gamma.store,
     xlab = expression(lambda), ylab = expression(gamma))
```

- This visualises the **joint posterior** relationship between λ and γ.
- Useful for understanding parameter trade-offs and uncertainty.

Recap

- Gibbs sampling is available when full conditionals are easy to sample from.
- Algorithm: initialise, then update each component using its conditional given the latest values.
- Hierarchical exponential model gives Gamma full conditionals for both λ and γ.
- Implementation is simple: store samples, plot trace plots, summarise posteriors, and inspect joint dependence.

Recap

- Gibbs sampling is available when full conditionals are easy to sample from.
- Algorithm: initialise, then update each component using its conditional given the latest values.
- Hierarchical exponential model gives Gamma full conditionals for both λ and γ.
- Implementation is simple: store samples, plot trace plots, summarise posteriors, and inspect joint dependence.

Questions?