# Gibbs Sampling for a mixture of normals

**Matthew Stephens**
University of Chicago
January 24, 2026

See here for a PDF version of this vignette.

## Prerequisites

You should know about Gibbs sampling and mixture models, and be familiar with Bayesian inference for the normal mean and for the two class problem.

Load the gtools package, which is needed to sample from a Dirichlet distribution.

```
library(gtools)
```

## Overview

We consider using Gibbs sampling to perform inference for a normal mixture model,

$$X_1, \dots, X_n \sim f(x),$$

where

$$f(x) = \sum_{k=1}^{K} \pi_k N(x; \mu_k, 1).$$

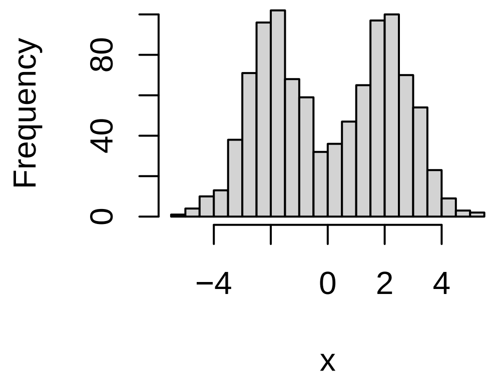
Here, $\pi_1, \dots, \pi_K$ are non-negative and sum to 1, and $N(\cdot\,; \mu, \sigma^2)$ denotes the probability density function of the $N(\mu, \sigma^2)$ distribution.

Recall the latent variable representation of this model:

$$\Pr(Z_j = k) = \pi_k X_j \mid Z_j = k \quad \sim N(\mu_k, 1).$$

To illustrate, let's simulate data from this model:

```
set.seed(33)
# Simulate from a mixture of normals.
#' @param n Number of samples.
#' @param pi Mixture proportions.
#' @param mu Mixture means.
#' @param s Mixture standard deviations.
rmix <- function (n, pi, mu, s) {
  z <- sample(length(pi),prob = pi,size = n,replace = TRUE)
  x <- rnorm(n,mu[z],s[z])
  return(x)
}
x <- rmix(n = 1000,pi = c(0.5,0.5),mu = c(-2,2),s = c(1,1))
hist(x,breaks = 32,main = "")
```



## Gibbs sampler

Suppose we want to perform inference for the parameters $\mu, \pi$. That is, we want to sample from the posterior distribution $p(\mu, \pi \mid x)$. We can use a Gibbs sampler. However, to do this we have to augment the space to sample from $p(z, \mu, \pi \mid x)$, not only $p(\mu, \pi \mid x)$.

Here is the algorithm in outline:

1. Sample $\mu$ from $p(\mu \mid x, z, \pi)$.
2. Sample $\pi$ from $p(\pi \mid x, z, \mu)$.
3. Sample $z$ from $p(z \mid x, \pi, \mu)$.

The point here is that all of these conditionals are easy to sample from.

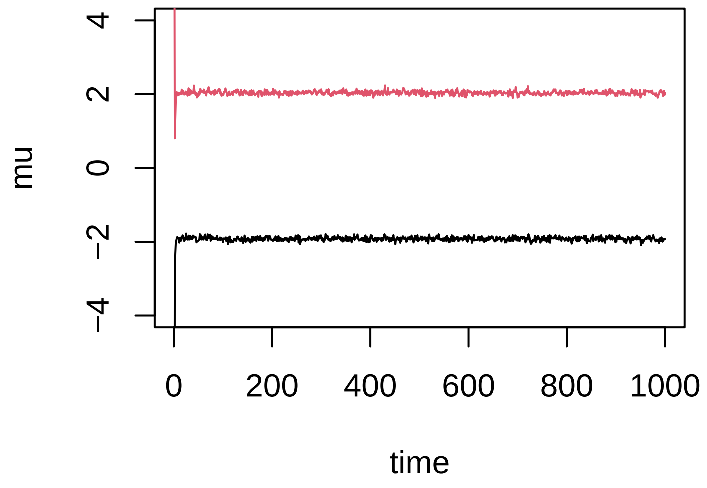
## The code

```
normalize <- function (x)
  return(x/sum(x))

#' @param x Data vector (length n).
#' @param pi Vector (length k).
#' @param mu Vector (length k).
sample_z <- function (x, pi, mu) {
  dmat <- outer(mu,x,"-") # k x n matrix, d[k,j] = mu[k] - x[j]
  p.z.given.x <- as.vector(pi) * dnorm(dmat)
  p.z.given.x <- apply(p.z.given.x,2,normalize) # Normalize columns.
  z <- rep(0,length(x))
  for (i in 1:length(z))
    z[i] <- sample(length(pi),size = 1,prob = p.z.given.x[,i],replace = TRUE)
  return(z)
```

```r
}

#' @param z Vector of cluster allocations (length n).
#' @param k The number of clusters.
sample_pi <- function (z, k) {
  counts <- colSums(outer(z,1:k,FUN = "=="))
  pi <- rdirichlet(1,counts + 1)
  return(pi)
}

#' @param x Data vector (length n).
#' @param z Cluster allocations (length n).
#' @param k The number of clusters.
#' @param prior.mean The prior mean for mu.
#' @param prior.prec The prior precision for mu.
sample_mu <- function (x, z, k, prior) {
  mu <- rep(0,k)
  for (i in 1:k) {
    sample.size <- sum(z == i)
    sample.mean <- ifelse(sample.size == 0,0,mean(x[z == i]))
    post.prec <- sample.size + prior$prec
    post.mean <- (prior$mean * prior$prec + sample.mean * sample.size)/post.prec
    mu[i] <- rnorm(1,post.mean,sqrt(1/post.prec))
  }
  return(mu)
}

gibbs <- function (x, k, niter = 1000, muprior = list(mean = 0,prec = 0.1)) {
  pi <- rep(1/k,k)
  mu <- rnorm(k,0,10)
  z  <- sample_z(x,pi,mu)
  res <- list(mu = matrix(0,niter,k),
              pi = matrix(0,niter,k),
              z  = matrix(0,niter,length(x)))
  res$mu[1,] <- mu
  res$pi[1,] <- pi
  res$z[1,]  <- z
  for (i in 2:niter) {
    pi <- sample_pi(z,k)
    mu <- sample_mu(x,z,k,muprior)
    z  <- sample_z(x,pi,mu)
    res$mu[i,] <- mu
    res$pi[i,] <- pi
    res$z[i,]  <- z
  }
  return(res)
}
```

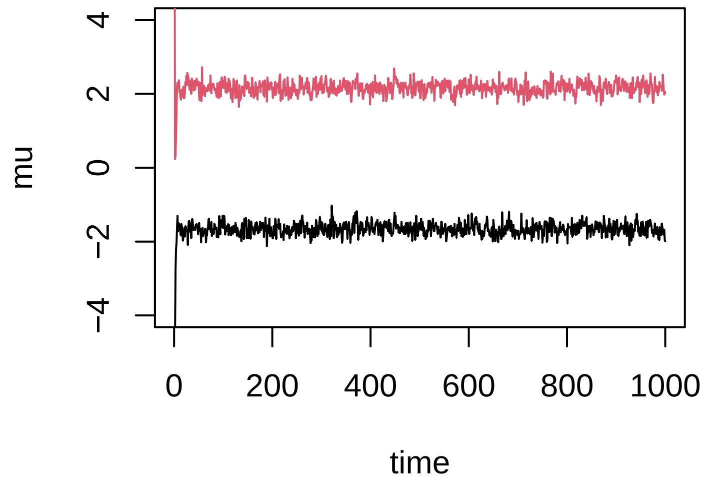
Try the Gibbs sampler on the data set we simulated above. We see it quickly moves to a part of the space where the mean parameters are near their true values (-2 and 2):

```
res <- gibbs(x,2)
plot(res$mu[,1],ylim = c(-4,4),type = "l",xlab = "time",ylab = "mu")
lines(res$mu[,2],col = 2)
```



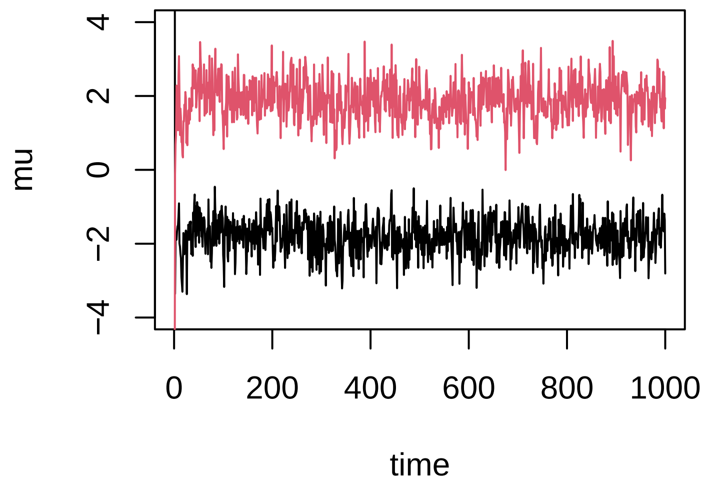
If we simulate data with fewer observations we should see more uncertainty in the parameters:

```
x <- rmix(100,c(0.5,0.5),c(-2,2),c(1,1))
res2 <- gibbs(x,2)
plot(res2$mu[,1],ylim = c(-4,4),type = "l",xlab = "time",ylab = "mu")
lines(res2$mu[,2],col = 2)
```



And with even fewer observations…

```
x <- rmix(10,c(0.5,0.5),c(-2,2),c(1,1))
res3 <- gibbs(x,2)
plot(res3$mu[,1],ylim = c(-4,4),type = "l",xlab = "time",ylab = "mu")
lines(res3$mu[,2],col = 2)
```



We can easily obtain credible intervals (CI) from these samples. For example, to get 90% posterior CIs for the mean parameters:

```
quantile(res3$mu[-(1:10),1],c(0.05,0.95))
#     5%    95%
# -2.645 -1.004
quantile(res3$mu[-(1:10),2],c(0.05,0.95))
#    5%   95%
# 0.940 2.777
```

(Here we discarded the first few samples as “burn-in”.)