Gibbs sampling for clustering genetic data

Matthew Stephens
University of Chicago
January 24, 2026

See here for a PDF version of this vignette.

Prerequisites

Be familiar with Bayesian inference for the two class problem and conjugate Bayesian analysis for a binomial proportion.

Overview

Suppose we observe genetic data on a sample of n elephants at R locations ("loci") in the genome. For simplicity, we will assume the elephants are haploid; that is, they have just one copy of their genome. And we will assume that there are just two genetic types ("alleles") at each locus, which we will label as 0 and 1.

We will further assume that there are two type of elephant: forest elephants and savanna elephants, and that the allele frequencies in forest elephants are different from those in savanna elephants, but that the allele frequencies for each of these two groups are unknown. Also, we do not know which samples are forest elephants and which are savanna elehants. Our goal is to infer both these sets of unknowns: (i) which elephants are forest and which are savanna; (ii) what are the allele frequencies in each group of elephants.

Notation

Let x_i denote the genetic data for individual i ($i = 1, \ldots, n$). Thus, x_i is a binary vector (a vector of zeros and ones) of length R. Let X denote the combined genetic data, $X = (x_1, \ldots, x_n)$.

Let $z_i \in \{0, 1\}$ denote the group (forest vs. savanna) of individual i, and let Z denote the vector $Z = (z_1, \ldots, z_n)$.

Let P_{kj} denote the frequency of the 1 allele at locus j in group k ($j = 1, \ldots, R$, $k = 0, 1$). (Here, 0 = forest and 1 = savanna.) Let P_k denote the vector $(P_{k1}, \ldots, P_{kR})$, and let P denote all the (unknown) allele frequencies $P = (P_0, P_1)$.

With this notation in place, we can state the problem, which is to infer the unknowns P and Z from the genetic data X.

Model

To perform Bayesian inference for Z and P, we need to specify the likelihood, $p(X \mid Z, P)$, and a prior distribution, $p(Z, P)$.

Likelihood

For each individual, we will assume that if we knew its group of origin, and we knew the allele frequencies in each group, then the genetic data at the different markers are independent draws from the relevant allele frequencies. This is exactly the model assumed here. In mathematical notation, we assume

$$p(x_i \mid z_i, P) = \prod_{j=1}^{R} P_{z_i j}^{x_{ij}} (1 - P_{z_i j})^{1 - x_{ij}}.$$

All the subscripts here make this difficult to read. To make things easier to read, we can replace z_i with k:

$$p(x_i \mid z_i = k, P) = \prod_{j=1}^{R} P_{kj}^{x_{ij}} (1 - P_{kj})^{1 - x_{ij}}.$$

We will further assume that the different individuals are independent:

$$p(X \mid Z, P) = \prod_{i=1}^{n} p(x_i \mid z_i, P).$$

This completes specification of the likelihood.

Prior

We will assume that P and Z are *a priori* independent, so $p(P, Z) = p(P)\, p(Z)$. This assumption seems reasonable: before seeing the genetic data (X), telling you the allele frequencies in the two groups would not tell you anything about the group membership of the elephants. (Of course, after seeing the genetic data X, the allele frequencies would help classify the individuals, so P and Z are not going to be *a posteriori* independent. However, here we are concerned with the prior, not the posterior.)

For the prior on P, we will further assume that the allele frequencies in each group at each locus are independent, so $p(P) = \prod_{k=1}^{2} \prod_{j=1}^{R} p(P_{kj})$. This assumption could be improved, but at the cost of considerable extra complexity, and so we stick with independence for now. Also, for simplicity we will assume a uniform prior distribution for P_{kj}, so $p(P_{kj}) = 1$.

For Z, we will assume that the origin of each individual is independent, with an equal probability (0.5) of arising from each of the two groups. So

$$p(Z) = \prod_{i=1}^{n} p(z_i),$$

and $p(z_i = k) = 0.5$. Again, this assumption could be improved, but we start here for simplicity.

Computation

Our goal is to compute (or sample from) the posterior distribution $p(Z, P \mid X)$, which by Bayes Theorem is

$$p(Z, P \mid X) \propto p(X \mid Z, P)\, p(Z, P).$$

One way to sample from this distribution is to implement a Gibbs sampler. This requires us to be able to do two things:

1. Sample from
$$p(Z \mid P, X)$$
.
2. Sample from
$$p(P \mid Z, X)$$
.

These are called the "full conditional distributions" for Z and P respectively. The use of the word "full" here indicates that they are conditional on *everything else* (the data and all the other parameters).

Full conditional for Z

We know that

$$p(Z \mid P, X) \propto p(Z, P, X) = \prod_{i=1}^{n} p(x_i \mid z_i, P)\, p(z_i)\, p(P).$$

So we see that the full conditional for $Z = (z_1, \ldots, z_n)$ factorizes over i into terms that depend only on z_i and not the other elements of Z. That is,

$$p(Z \mid P, X) \propto \prod_{i=1}^{n} f_i(z_i; x_i, P)$$

for some (yet to be determined) functions f_i.

This implies that the z_i are *conditionally independent* given X, P, which is very convenient as it means we can compute their conditional distribution just by computing the marginals:

$$p(Z_i = k \mid P, X) \propto p(x_i \mid z_i = k, P)$$

Exercise: Derive the full conditional for P.

Simulate some data

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

```r
set.seed(33)
# Generate data from a mixture of normals.
#' @param n The number of samples to simulate.
#' @param P A 2 x R matrix of allele frequencies.
r_simplemix <- function (n, P) {
  R <- ncol(P)
  z <- sample(2,prob = c(0.5,0.5),size = n,replace = TRUE)
  x <- matrix(0,n,R)
  for (i in 1:n)
    x[i,] <- rbinom(R,rep(1,R),P[z[i],])
  return(list(x = x,z = z))
}
P <- rbind(c(0.500,0.500,0.500,0.500,0.500,0.500),
           c(0.001,0.999,0.001,0.999,0.001,0.999))
sim <- r_simplemix(n = 50,P)
x <- sim$x
```

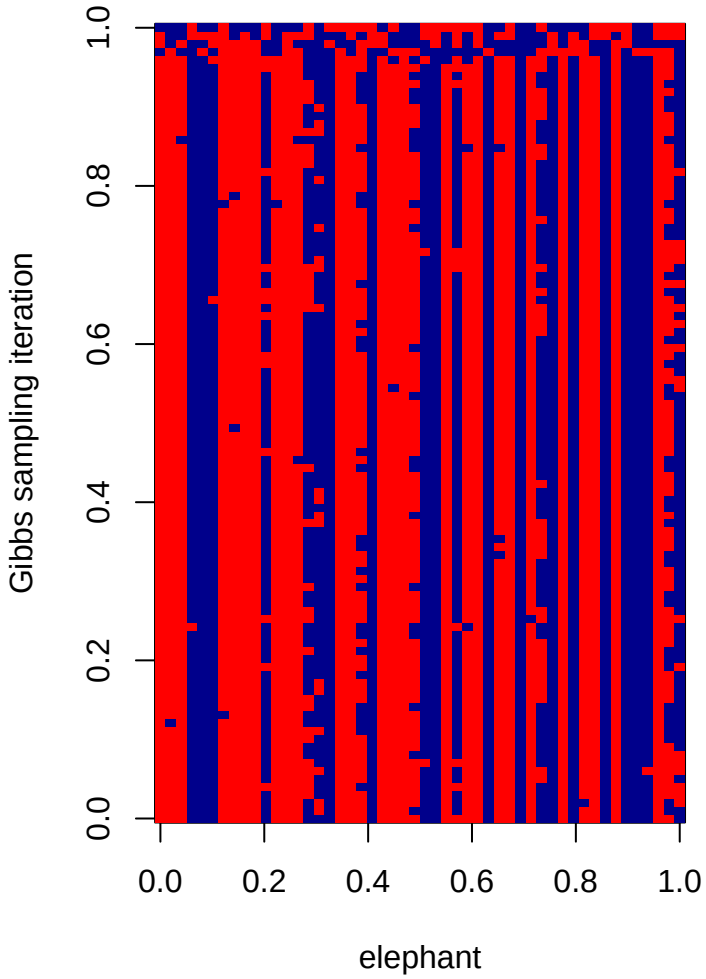
Gibbs sampler code

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

#' @param x Data vector (length R).
#' @param P 2 x R matrix of allele frequencies.
#' @return The log-likelihood for each of the K populations.
log_pr_x_given_P <- function (x, P)
  colSums(x*log(t(P)) + (1-x)*log(1-t(P)))

#' @param x n x R data matrix.
#' @param P 2 x R matrix of allele frequencies.
#' @return Group memberships (vector of length n).
sample_z <- function (x, P) {
  K <- nrow(P)
  loglik_matrix <- apply(x,1,log_pr_x_given_P,P = P)
  lik_matrix <- exp(loglik_matrix)
  p.z.given.x <- apply(lik_matrix,2,normalize)
  z <- rep(0,nrow(x))
  for (i in 1:length(z))
    z[i] <- sample(K,size = 1,prob = p.z.given.x[,i],replace = TRUE)
  return(z)
}

#' @param x n x R data matrix.
#' @param z Cluster allocations (vector of length n).
#' @return 2 x R matrix of allele frequencies.
```

```
sample_P <- function (x, z) {
  R <- ncol(x)
  P <- matrix(0,2,R)
  for (i in 1:2) {
    sample_size <- sum(z == i)
    if (sample_size == 0)
      number_of_ones <- rep(0,R)
    else
      number_of_ones <- colSums(x[z == i,])
    P[i,] <- rbeta(R,number_of_ones + 1,sample_size - number_of_ones + 1)
  }
  return(P)
}

gibbs <- function (x, niter = 100) {
  n <- nrow(x)
  z <- sample(2,n,replace = TRUE)
  out <- matrix(0,niter,n)
  out[1,] <- z
  for(i in 2:niter) {
    P <- sample_P(x,z)
    z <- sample_z(x,P)
    out[i,] <- z
  }
  return(out)
}
```

Try the Gibbs sampler on the data simulated above:

```
z <- gibbs(x,niter = 100)
table(mcmc = z[1,],true = sim$z)
#     true
# mcmc  1  2
#    1 14 11
#    2 10 15
table(mcmc = z[100,],true = sim$z)
#     true
# mcmc  1  2
#    1  4 26
#    2 20  0
image(t(z[100:1,]),xlab = "elephant",
      ylab = "Gibbs sampling iteration",
      col = c("red","darkblue"))
```


Gibbs sampling iteration
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
elephant