Matt Dickenson `mcd31`

STA561/CS571 — Fall 2013 **Homework 6** Due: 28 October, 2013

Homework Notes: I did not work with anyone else on this homework or refer to resources other than the course notes, textbook, and course Piazza page.

Problem 1

A To update the sample of μ_k at iteration $m+1$, we can sample $\mu_{k,m+1} \sim Unif(l,u)$ where $l = min(0, \mu_k - \epsilon)$ and $u = max(5, \mu_k + \epsilon)$. The prevents us from sampling outside the range defined by the prior. In my samples, I set the 'step' $\epsilon = 0.1$.

The MH acceptance probability is the ratio of the likelihood of the new and old samples: $p(keep) = \frac{\mathcal{L}(X|\mu_{m+1})}{\mathcal{L}(X|\mu_m)}$.

B We chose to perform Metropolis-Hastings rather than a Gibbs sample step because we need to sample μ_k for all k simultaneously. Moreover, we cannot sample from the the conditional probability $p(\mu_k|x, z, \Sigma)$, which would be needed for the Gibbs step.

C I ran 500 rounds of burn-in and 3,000 iterations of sampling. Parameters were initialized as:

Listing 1: R Code for Initial Parameters

```
# settings
BURN = 500
M = 3000+BURN
STEP = 0.1
MIN = 0
MAX = 5
K = 2

# initialize
pis = mus = matrix(NA, nrow=M, ncol=K)
mus[1, ] = sample(X, 2)
mus[1,]
Z = matrix(NA, nrow=nrow(X), ncol=K)
Z[1,] = pis[1, ] = rep(1/K, K)
N = x.bar = rep(0, K)
z = rep(0, nrow(X))
alpha = rep(1, K)
S_0 = rep(3, K)
v_0 = rep(5, K)
v = rep(0, K)
S = rep(0, K)
Sigma.inv = rgamma(K, shape=v_0, rate=S_0)
new.mu = rep(NA, K)
```

The following code runs the burn-in and sampling iterations:

Listing 2: R Code for MH

```r
for(m in 2:M){
  # step 1 - simulate proportion vector from Dirichlet
  pisamp = rdirichlet(1, c(alpha[1]+N[1], alpha[2]+N[2]) )
  pis[m, ] = pisamp[1, ]

  # step 2 - sample latent indicators
  for(i in 1:nrow(Z)){
    for(j in 1:ncol(Z)){
      Z[i, j] = pis[m, j] * dnorm(X[i], mus[m-1, j], 1/Sigma.inv[j])
    }
    Z[i,] = Z[i,]/sum(Z[i,])

    # step 3 - calculate new z's based on Z draws
    z[i] = sample(seq(1:K), size=1, prob=Z[i,])
  }

  # step 4 - update mean and variance
  for(k in 1:K){
    want = which(z==k)
    N[k] = length(want)
    subset = X[want]
    x.bar[k] = mean(subset)
    v[k] = v_0[k] + N[k]
    S[k] = S_0[k] + sum.sq.err(subset, mus[m-1,k]) # test this
    Sigma.inv[k] = rgamma(1, shape=v[k], rate=S[k])

    lower = max(c(MIN, mus[m-1, k]-STEP))
    upper = min(c(MAX, mus[m-1, k]+STEP))
    new.mu[k] = runif(1, lower, upper)

  }

  ll.new.mu = log.lik(X, z, new.mu)
  ll.old.mu = log.lik(X, z, mus[m-1,])

  acceptance.prob = ll.new.mu / ll.old.mu
  acceptance.prob = min(1, acceptance.prob, na.rm=TRUE)
  if(runif(1,0,1)<acceptance.prob){
    mus[m, ] = new.mu
    likelihood.trace[m, 1] = ll.new.mu
  } else {
    mus[m, ] = mus[m-1, ]
    likelihood.trace[m, 1] = ll.old.mu
  }

  if(m%%10==0){ print(m) }
}
```

D Figure 1 shows the log likelihood trace for three different runs of the sampler using three different (randomly chosen) starting points.

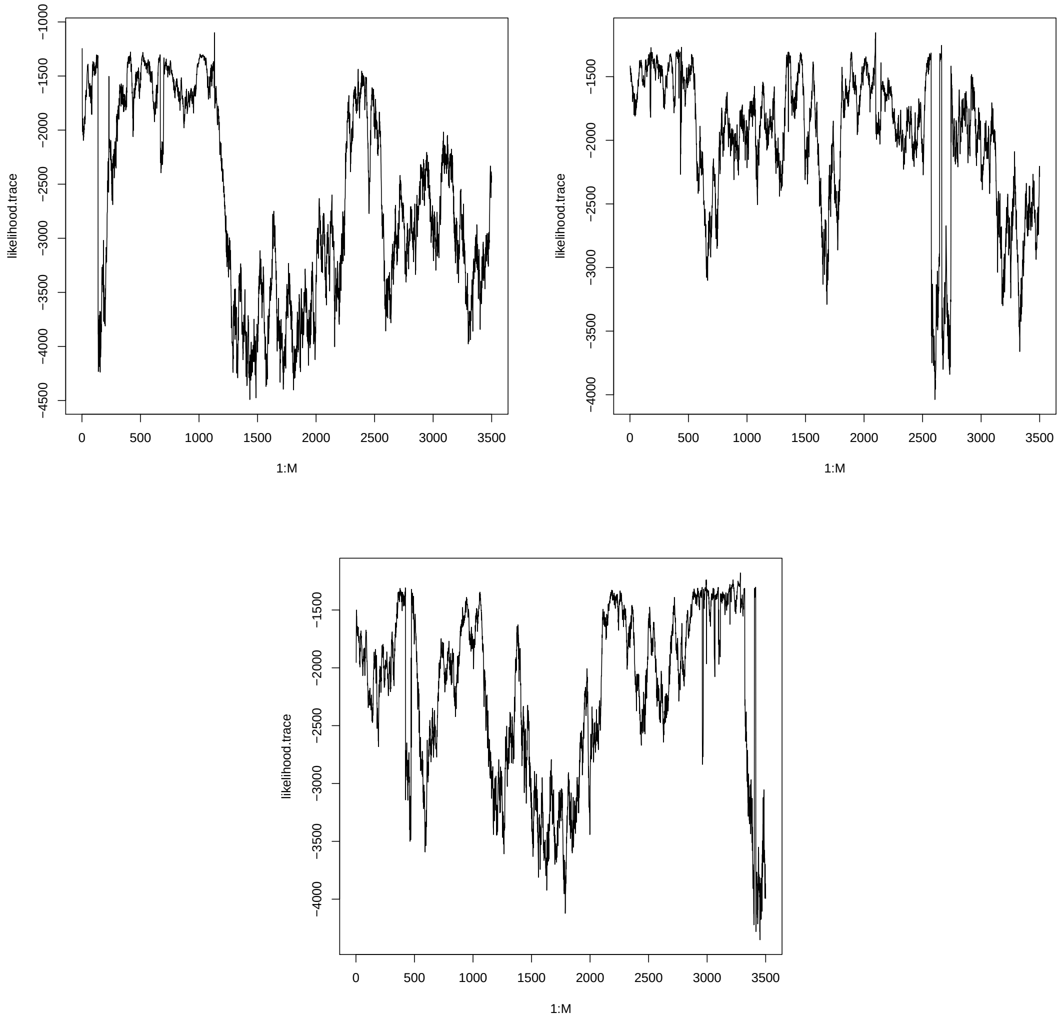

Figure 1: Traceplot of Likelihood for Three Starting Points

E Figure 2 presents histograms of the posterior samples for each mean parameter from a single run (after burn-in). It appears that $K = 2$ is a reasonable fit for the data, with $\mu \approx (1.5, 3)$. Label switching did not occur in these samples.

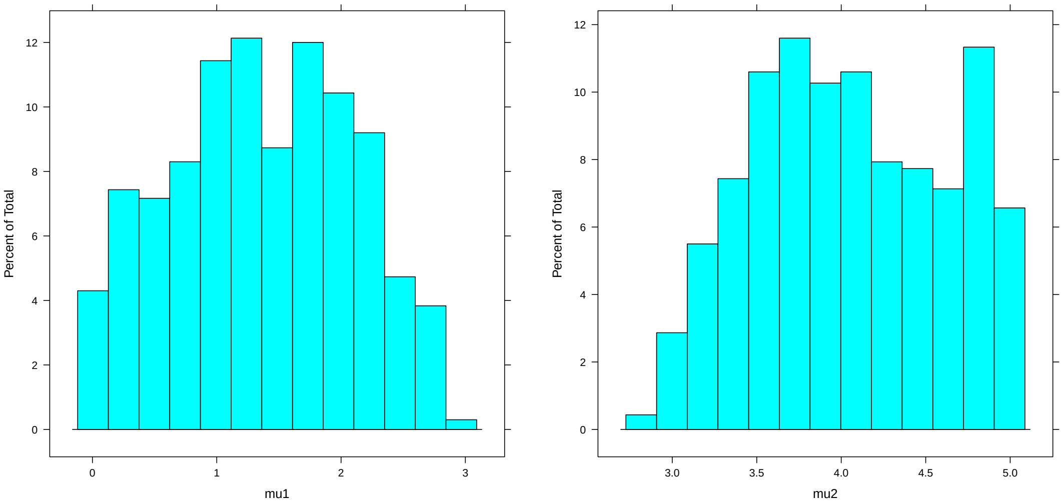

Figure 2: Histogram of Posterior Samples

F I would use a composite from all three samples, weighted by the likelihood of each.