Homework 4

Minxuan Chen, Andrew Nguyen, Han Yang

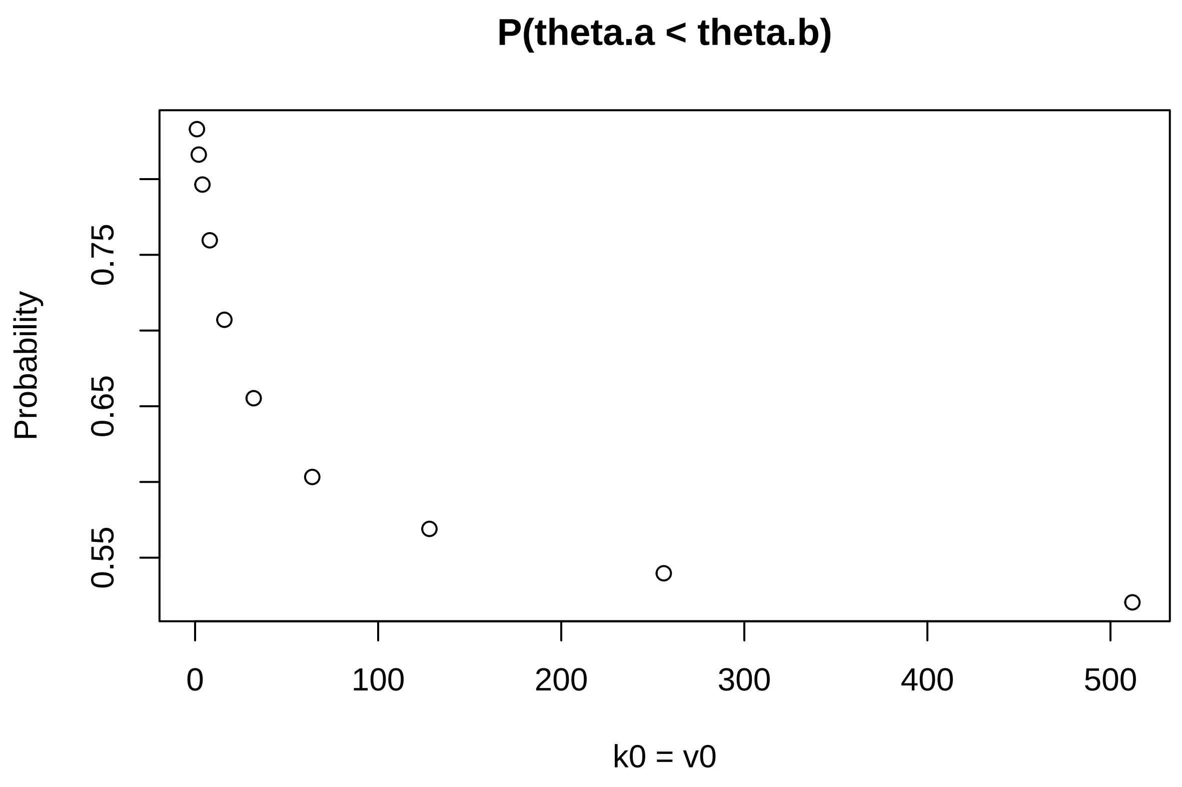
2023-12-10

Problem 1 (Andrew, Han)

part a

```r
ybar.a = 75.2
std.a = 7.3
n.a = 16

ybar.b = 77.5
std.b = 8.1
n.b = 16

prior.mean = 75
prior.variance = 100
```

```r
k0 = v0 = c(1,2,4,8,16,32,64,128,256,512)
probs = vector(length=length(k0))

set.seed(42)
for (i in 1:length(k0)) {

  v.n = v0[i] + n.a

  sigma.n2 = (1/v.n)*(v0[i]*prior.variance + k0[i]*n.a*(prior.mean-ybar.a)^2/(k0[i]+n.a) + (n.a-1)*std.
  sigmas.tilde.2.a = rgamma(10000, v.n/2, v.n*sigma.n2/2)

  post.mean.a = (k0[i]*prior.mean + n.a*ybar.a) / (k0[i] + n.a)
  post.precision.a = (k0[i] +n.a)*sigmas.tilde.2.a
  thetas.a = rnorm(10000, mean=post.mean.a, sd=1/sqrt(post.precision.a))

  v.n = v0 + n.b

  sigma.n2 = (1/v.n)*(v0[i]*prior.variance + k0[i]*n.b*(prior.mean-ybar.b)^2/(k0[i]+n.b) + (n.b-1)*std.
  sigmas.tilde.2.b = rgamma(10000, v.n/2, v.n*sigma.n2/2)

  post.mean.b = (k0[i]*prior.mean + n.b*ybar.b) / (k0[i] + n.b)
  post.precision.b = (k0[i] +n.b)*sigmas.tilde.2.b
  thetas.b = rnorm(10000, mean=post.mean.b, sd=1/sqrt(post.precision.b))
  probs[i] = mean(thetas.a < thetas.b)
}
```



The larger $\kappa_0 = \nu_0$ is, the closer to .50 the probability approaches. However, it never reaches .50. This is evidence that $\theta_A < \theta_B$ across numerous priors.

part b

```
sample.mu = function(n1, n2, ybar1, ybar2, mu0, gamma0.sq, delta, sigma.sq) {
  gamma.n.sq = 1/(1/gamma0.sq + (n1+n2)/sigma.sq)
  mu.n = gamma.n.sq * (mu0/gamma0.sq
                       + (n1/sigma.sq)*(ybar1-delta)
                       + (n2/sigma.sq)*(ybar2-delta)
                       )
  mu = rnorm(1, mean=mu.n, sd=sqrt(gamma.n.sq))
  return(mu)
}

sample.delta = function(n1, n2, ybar1, ybar2, mu0, delta0, tau0.sq, mu, sigma.sq) {
  tau.n.sq = 1/(1/tau0.sq + (n1+n2)/sigma.sq)
  delta.n = tau.n.sq * (delta0/tau0.sq
                       + (n1/sigma.sq)*(ybar1-mu)
                       + (n2/sigma.sq)*(ybar2-mu)
                       )
```

```r
  delta = rnorm(1, mean=delta.n, sd=sqrt(tau.n.sq))
}

sample.sigma.sq = function(n1, n2, ybar1, ybar2, v0, sigma.sq0, mu, delta, std1,std2) {
  v.n = v0 + n1 + n2
  a = std1^2*(n1-1) + n1*ybar1^2 - 2*(mu+delta)*n1*ybar1 + n1*(mu+delta)^2
  b = std2^2*(n2-1) + n2*ybar2^2 - 2*(mu-delta)*n2*ybar2 + n2*(mu-delta)^2
  vn.sigma.nsq = v0*sigma.sq0 + a + b
  sigma.sq = invgamma::rinvgamma(1, v.n/2, vn.sigma.nsq/2)
  return(sigma.sq)
}

gibbs.sample = function(niter =1000, n1, n2, ybar1, ybar2, std1, std2,
                        mu0, gamma.sq0, delta0, tau0.sq, v0, sigma.sq0) {
  res = list(mu = vector(length=niter),
             delta = vector(length=niter),
             sigma.sq = vector(length=niter))
  # initializing
  mu = 75; delta = 10; sigma.sq = 100
  res$mu[1]= mu
  res$delta[1]= delta
  res$sigma.sq[1] = sigma.sq

  for (i in 2:niter) {
    mu = sample.mu(n1, n2, ybar1, ybar2, mu0, gamma0.sq, delta, sigma.sq)
    delta = sample.delta(n1, n2, ybar1, ybar2, mu0, delta0, tau0.sq, mu, sigma.sq)
    sigma.sq = sample.sigma.sq(n1, n2, ybar1, ybar2, v0, sigma.sq0, mu, delta, std1,std2)

    res$mu[i]= mu
    res$delta[i]= delta
    res$sigma.sq[i] = sigma.sq

  }
  return(res)
}
```

```r
ybar.a = 75.2
std.a = 7.3
n.a = 16

ybar.b = 77.5
std.b = 8.1
n.b = 16

mu0 = 75
gamma0.sq = 100
v0 = 2
sigma.sq0 = 200
delta0 = c(-4,-2,0,2,4)
tau0.sq = c(10,50, 100, 500)

res.mat = matrix(NA, nrow=length(delta0)*length(tau0.sq), ncol=7)
colnames(res.mat) = c("delta0","tau0.sq","prior.corr", "Pr(delta < 0)","post.corr","lower","upper")
```

```
res.mat[, c("delta0","tau0.sq")] = as.matrix(expand.grid(delta0,tau0.sq))

set.seed(42)
for (i in 1:nrow(res.mat)) {
  mu = rnorm(10000, 75, 10)
  delta0 = res.mat[i,c("delta0")]; tau.sq0 = res.mat[i,c("tau0.sq")]
  delta = rnorm(10000, delta0, sqrt(tau.sq0))
  prior.thetas.a = mu + delta
  prior.thetas.b = mu - delta
  # prior.corr.matrix[i,j] = round(cor(prior.thetas.a,prior.thetas.b),3)
  res.mat[i,c("prior.corr")] = round(cor(prior.thetas.a,prior.thetas.b),3)
  res = gibbs.sample(niter =10000, n.a, n.b, ybar.a, ybar.b, std.a, std.b,
                     mu0, gamma.sq0, delta0, tau.sq0, v0, sigma.sq0)
  # prob.matrix[i,j] = mean(res$delta < 0)
  res.mat[i,c("Pr(delta < 0)")] = mean(res$delta < 0)
  thetas.a = res$mu + res$delta
  thetas.b = res$mu - res$delta
  # post.corr.matrix[i,j] = round(cor(thetas.a,thetas.b),3)
  res.mat[i,c("post.corr")] = round(cor(thetas.a,thetas.b),3)
  res.mat[i,c("lower")] = round(quantile(res$delta, .025),2)
  res.mat[i,c("upper")] = round(quantile(res$delta, .975),2)
}
```

```
##       delta0 tau0.sq prior.corr Pr(delta < 0) post.corr  lower upper
##  [1,]     -4      10      0.818        0.8633     0.214  -8.58  2.41
##  [2,]     -2      10      0.815        0.7182     0.226  -7.32  3.81
##  [3,]      0      10      0.818        0.5153     0.220  -5.37  5.57
##  [4,]      2      10      0.817        0.3156     0.231  -4.01  6.86
##  [5,]      4      10      0.818        0.1476     0.246  -2.33  8.89
##  [6,]     -4      50      0.328        0.6079     0.028 -10.78  7.40
##  [7,]     -2      50      0.314        0.5670     0.058 -10.17  8.09
##  [8,]      0      50      0.320        0.5315     0.056  -9.32  8.88
##  [9,]      2      50      0.336        0.4526     0.053  -8.06 10.17
## [10,]      4      50      0.333        0.3642     0.047  -6.84 11.75
## [11,]     -4     100     -0.009        0.5722    -0.027 -11.58  9.51
## [12,]     -2     100      0.004        0.5367     0.010 -10.76 10.02
## [13,]      0     100      0.012        0.5016    -0.010 -10.27 10.73
## [14,]      2     100     -0.004        0.4603    -0.013  -9.43 11.60
## [15,]      4     100      0.010        0.4454    -0.001  -9.27 12.03
## [16,]     -4     500     -0.662        0.5209    -0.132 -13.77 13.30
## [17,]     -2     500     -0.665        0.5323    -0.087 -12.69 12.05
## [18,]      0     500     -0.672        0.5134    -0.115 -12.24 12.74
## [19,]      2     500     -0.669        0.4880    -0.131 -12.20 14.44
## [20,]      4     500     -0.667        0.5249    -0.121 -12.45 14.00
```

One can use this information to convey the fact that $\theta_A > \theta_B$ across numerous priors by examining the posterior probability of delta being less than 0 and the confidence intervals. For most combinations of `delta0` and `tau0.sq`, one can see that delta is typically negative, and the credible intervals do cover negative values.

Problem 2 (Minxuan)

part a

First, we condition the whole model on K.

We will specify the parameters and hyperparameters, the likelihood and the full conditional distributions.

We denote the points in dataset by $\mathbf{X}$, which is a $n \times 3$ matrix. Each row of $\mathbf{X}$ corresponds to a point, denoted by X_i.

For each point X_i, we assign a class label to it, denoted by $Z_i, Z_i = 1, 2, \cdots, K$. They are latent variables in this model.

The probability that point X_i belongs to class j is denoted by p_j. Since we have K classes in total, we have a probability vector $(p_1, p_2, \cdots, p_K)$ with $\sum_i p_i = 1$. This vector is in $\mathbb{R}^{K-1}$ space, so for simplicity, we only consider $(p_1, p_2, \cdots, p_{K-1})$ and get p_K by $p_K = 1 - \sum_{i=1}^{K-1} p_i$.

When the class of point X_i is known, for example $Z_i = j$. The distribution of X_i is normal with parameters (μ_j, Σ_j), i.e.

$$X_i | Z_i = j \sim N(\mu_j, \Sigma_j)$$

Thus, we need $2K$ parameters in total for the mixture density, denoted by $(\mu_1, \Sigma_1, \mu_2, \Sigma_2, \cdots, \mu_K, \Sigma_K)$.

In summary, we regard $(p_1, \cdots, p_{K-1}, \mu_1, \Sigma_1, \cdots, \mu_K, \Sigma_K)$ as parameters.

And the class labels $(Z_1, \cdots, Z_n)$ as latent variables.

For priors on parameters, it's reasonable to assume some independence

$$\begin{aligned} &p(p_1, \cdots, p_{K-1}, \mu_1, \Sigma_1, \cdots, \mu_K, \Sigma_K) \\ =&p(p_1, \cdots, p_{K-1}) \prod_i p(\mu_i, \Sigma_i) \end{aligned}$$

We write out the posterior of

$$(\mathbf{Z}, \mathbf{p}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (Z_1, \cdots, Z_n, p_1, \cdots, p_{K-1}, \mu_1, \Sigma_1, , \cdots, \mu_K, \Sigma_K)$$

$$\begin{aligned} &p(\mathbf{Z}, \mathbf{p}, \boldsymbol{\mu}, \boldsymbol{\Sigma} | \mathbf{X}) \\ \propto& p(\mathbf{X} | \mathbf{Z}, \mathbf{p}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(\mathbf{Z}, \mathbf{p}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ \propto& \prod_{i=1}^n p(X_i | \mathbf{Z}, \mathbf{p}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(\mathbf{Z} | \mathbf{p}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(\mathbf{p}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ \propto& \left(\prod_{i=1}^n p(X_i | Z_i, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \right) \left(\prod_{i=1}^n p(Z_i | \mathbf{p}) \right) p(p_1, \cdots, p_{K-1}) \prod_{j=1}^K p(\mu_j, \Sigma_j) \end{aligned}$$

For the first term, let $f(x | \mu_t, \Sigma_t)$ denote the density of normal distribution $N(\mu_t, \Sigma_t)$. We use the indicator $I(Z_i = t)$ to represent whether point X_i belongs to class t. It takes the value 1 if $Z_i = t$, otherwise 0. Then

$$p(X_i | Z_i, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{t=1}^K I(Z_i = t) f(X_i | \mu_t, \Sigma_t)$$

Note that although this expression is complicated, as long as Z_i is given, $p(X_i|Z_i, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ only contains one term, which is some normal density. Therefore, for the product, we can rewrite the summation index to get

$$\begin{aligned}&\prod_{i=1}^{n} p(X_i|Z_i, \boldsymbol{\mu}, \boldsymbol{\Sigma})\\ =&\prod_{t=1}^{K} \prod_{i:Z_i=t} f(X_i|\mu_t, \Sigma_t)\end{aligned}$$

We can next combine this term with the last one

$$\prod_{i=1}^{n} p(X_i|Z_i, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \prod_{j=1}^{K} p(\mu_j, \Sigma_j) = \prod_{t=1}^{K} \left(p(\mu_t, \Sigma_t) \cdot \prod_{i:Z_i=t} f(X_i|\mu_t, \Sigma_t) \right)$$

Note that the inner term describe the prior and likelihood of normal $N(\mu_t, \Sigma_t)$. Therefore, it's reasonable to assign a Jeffreys' prior to (μ_t, Σ_t), which is

$$p(\mu_t, \Sigma_t) \propto |\Sigma_t|^{-5/2}, t = 1, 2, \cdots, K$$

For the second term, by definition of $\mathbf{p}$, we have

$$p(Z_i|\mathbf{p}) = p_1^{I(Z_i=1)} p_2^{I(Z_i=2)} \cdots p_K^{I(Z_i=K)} = \prod_{t=1}^{K} p_t^{I(Z_i=t)}$$

and

$$\begin{aligned}\prod_{i=1}^{n} p(Z_i|\mathbf{p}) &= \prod_{i=1}^{n} \prod_{t=1}^{K} p_t^{I(Z_i=t)}\\ &= \prod_{t=1}^{K} \prod_{i=1}^{n} p_t^{I(Z_i=t)}\\ &= \prod_{t=1}^{K} p_t^{\sum_{i=1}^{n} I(Z_i=t)}\end{aligned}$$

This term is the likelihood of a multinomial distribution. Therefore, it's reasonable to assign a Dirichlet distribution on the parameter $(p_1, \cdots, p_{K-1})$, that is

$$p(p_1, \cdots, p_{K-1}) \propto \left(1 - \sum_{t=1}^{K-1} p_t\right)^{\alpha_K - 1} \prod_{t=1}^{K-1} p_t^{\alpha_t - 1} = \prod_{t=1}^{K} p_t^{\alpha_t - 1}$$

For simplicity, we take $\alpha_t = 1, t = 1, 2, \cdots, K$, then

$$p(p_1, \cdots, p_{K-1}) \propto 1$$

From above, we can get

$$\begin{aligned}&p(\mathbf{Z}, \mathbf{p}, \boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathbf{X})\\ \propto&\prod_{t=1}^{K} \left(p(\mu_t, \Sigma_t) \cdot \prod_{i:Z_i=t} f(X_i|\mu_t, \Sigma_t) \right) \prod_{t=1}^{K} p_t^{\sum_{i=1}^{n} I(Z_i=t)}\\ \propto&\prod_{t=1}^{K} \left(p(\mu_t, \Sigma_t) \cdot \prod_{i:Z_i=t} f(X_i|\mu_t, \Sigma_t) \right) \prod_{t=1}^{K} p_t^{\sum_{i=1}^{n} I(Z_i=t)}\end{aligned}$$

Now, very similar to Homework 3, problem 3, we have

$$p(\mu_t|\cdot) \sim N(\bar{X}_t, \frac{\Sigma_t}{n_t}), t = 1, 2, \cdots, K$$

in which

$$\begin{aligned} n_t &= \sum_{i=1}^{n} I(Z_i = t) = \text{ number of points in class t} \\ \bar{X}_t &= \frac{1}{n_t} \sum_{i:\, Z_i = t} X_i \end{aligned}$$

and (using the notation in the book Bayesian Data Analysis Third edition)

$$p(\Sigma_t|\cdots) \propto \text{Inv-Wishart}_{n_t+1}((S_t^0)^{-1})$$

in which

$$\begin{aligned} &n_t \text{ follows above} \\ &S_t^0 = \sum_{i:\, Z_i=t}^{n} (X_i - \mu_t)(X_i - \mu_t)^T \end{aligned}$$

and

$$\begin{aligned} p(p_1, \cdots, p_{K-1}|\cdot) &\propto \prod_{t=1}^{K} p_t^{\sum_{i=1}^{n} I(Z_i=t)} \\ &\sim \text{Dirichlet}(1 + n_1, 1 + n_2, \cdots, 1 + n_K) \end{aligned}$$

in which

$$n_t \text{ follows above}, t = 1, 2, \cdots, K$$

and

$$p(Z_i = t|\cdot) \propto f(X_i|\mu_t, \Sigma_t)p_t$$

After getting all these, we consider when K is random. We assign a prior to it, for example, a truncated poisson distribution.

$$p(K = k) \propto \frac{\lambda^k}{k!} \mathbb{I}\{k \leq 100\}$$

Here λ is a constant, for simplicity, we take $\lambda = 4$.

Note that we have pointed out, the previous derivation is conditioned on K, in other words, we have got the full conditional distribution of $\theta = (\mathbf{Z}, \mathbf{p}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$, i.e.

$$p(\theta|\mathbf{X}, K)$$

and the prior becomes

$$p(\theta, K) = p(\theta|K)p(K)$$

Now for full conditional distribution of K, we have

$$\begin{aligned} p(K|\theta, \mathbf{X}) &= p(K, \theta|\mathbf{X})/p(\theta|\mathbf{X}) \\ &\propto p(\mathbf{X}|\theta, K)p(\theta|K)p(K) \end{aligned}$$

Note that for the first two terms $p(\mathbf{X}|\theta, K)p(\theta|K)$, we have already calculated them in $p(\theta|\mathbf{X}, K)$, but omited K in conditional set. So

$$p(K|\theta, \mathbf{X}) \propto \prod_{t=1}^{K} \left(p(\mu_t, \Sigma_t) \cdot \prod_{i:Z_i=t} f(X_i|\mu_t, \Sigma_t) \right) \prod_{t=1}^{K} p_t^{\sum_{i=1}^{n} I(Z_i=t)} \frac{\lambda^K}{K!} \mathbb{I}\{K \leq 100\}$$

part b

Sampling from this model using the standard Gibbs sampler is unfeasible. The primary challenge arises from the fact that, even though we possess the complete conditional distribution of all parameters, alterations in the value of K lead to changes in the dimensionality of the parameters.

Consequently, the transition process involves not only traversing between different parameters within a single space but also between two spaces with varying dimensions. Relying solely on transition kernels induced by the full conditional distribution is inadequate due to their fixed dimensions, resulting in a reducible Markov chain. Then there is no guarantee that this chain will converge to true posterior distribution.

To address this issue, we find out a method called Reversible-jump Markov chain Monte Carlo. It is introduced by Peter J.Green in the paper Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. It allows simulation of the posterior distribution on spaces of varying dimensions.

This method is based on the Metropolis-Hastings algorithm and requires the construction of a mapping and then the acceptance probability. However, the derivation and implementation of sampling are too complex. We have few ideas on how to solve these problems.

Therefore, to perform the sampling and determine the optimal K, we would use 5 fixed values of K and standard Gibbs samplers. After sampling, we use posterior means as estimation and calculate AIC value of these 5 models. We choose the model with lowest value as the final result. Below is an example with $K = 3$, which seems to yield the best results.

```r
mixgauss <- read.table("./mixgauss.dat", header=FALSE)
n <- nrow(mixgauss) #mixgauss is n x 3 shape
p <- 3
ttt <- 0

#' Function get class center
#'
#' @param X data matrix (n,3)
#' @param Z vector class label
#' @return mu class center
class_center <- function(X, Z){
  Xnew <- as.data.frame(cbind(X, Z))

  # Group by class label and calculate the mean for each group
  colnames(Xnew) <- c("x", "y", "z", "class_label")

  class_centers <- Xnew %>%
    group_by(class_label) %>%
    summarise(center_x = mean(x),
              center_y = mean(y),
              center_z = mean(z)) %>%
    ungroup() %>%
    arrange(class_label)
  # Return the result as a matrix
  return(as.matrix(class_centers[, c("center_x", "center_y", "center_z")]))
}

#' Gibbs sampler for p, \mu, \Sigma, Z
#' @param X data matrix (n,3)
#' @param Z vector class label
#' @param p vector p1,p2,...pK-1
#' @param Sigma vector of matrix
#' @param mu matrix, row i is mu_i
#' @return sample of step t
Gibbs <- function(X, Z, p, Sigma, mu, K){
  ni <- 1:K
  ni <- sapply(ni, function(x) sum(Z==x))


  #sample p
  p.t <- c(rdirichlet(1, 1+ni))
  #sample mu
  mu.t <- matrix(0, nrow=K, ncol=3)

  #get class center
  class_centers <- class_center(X, Z)
  for (i in 1:K){
    mu.t[i,] <- c(mvtnorm::rmvnorm(1,
                                   class_centers[i,],
                                   Sigma[[i]]/ni[i]))
  }
  #sample Sigma
  Sigma.t <- vector("list", K)
  for (i in 1:K){
    Xi <- t(X[Z==i, ])
    Si <- tcrossprod(Xi - mu[i,])
    tryCatch({
      # Code that might produce an error or warning
      solve(Si)
    }, error = function(err) {
      print(Si)
      print(Z)
      ttt<<-Si
      print(ttt)
      print(i)
      cat("An error occurred:", conditionMessage(err), "\n")
      # You can also access more information about the error using conditionName(err), conditionCall(er
    }, warning = function(warn) {
      # Code to handle a warning
      cat("A warning occurred:", conditionMessage(warn), "\n")
    })

    Sigma.t[[i]] <- MCMCpack::riwish(ni[i]+1, Si)
  }

  #unnormalized prob
  pZi <- 1:K
  pZi <- sapply(pZi,
                function(i) (dmvnorm(X, mean=mu.t[i, ], sigma=Sigma.t[[i]])*p.t[i])
  )
  #print(pZi)
  Z.t <- 1:nrow(X)
  Z.t <- sapply(Z.t, function(i)
    sample(1:K, size=1, replace=TRUE,
           prob=pZi[i,]))
  return(list(Z=Z.t, mu=mu.t, Sigma=Sigma.t, p=p.t))
}

#sampling

sampling_4_fix_K <- function(K, X){
  #we need initial value for  Z_1,\cdots,Z_n, p_1,\cdots,p_{K},
  #\mu_1,\Sigma_1,,\cdots,\mu_K,\Sigma_K,
  #K
  p.0 <- rep(1/K, K) # vector p1,p2,...pK
  Z.0 <- sample(1:K, n, replace=TRUE) # class label Z1...Zn
  mu.0 <- matrix(0, ncol=p, nrow=K) #class center
  Sigma.0 <- vector("list", K)
  for (i in 1:K){
    Sigma.0[[i]] <- diag(p)
  }

  samp.size <- 10000
  samples.out <- vector("list", samp.size+1)
  samples.out[[1]] <- list(Z=Z.0, mu=mu.0,
                         Sigma=Sigma.0, p=p.0)
  for (i in 2:(samp.size+1)){
    samples.out[[i]] <- do.call(Gibbs,
                              args=c(list(X=X, K=K), samples.out[[i-1]]))
  }

  mu.sample <- array(unlist(lapply(samples.out, function(p) p$mu)),
                     dim = c(K, 3, samp.size+1))
  Z.sample <- do.call(rbind, lapply(samples.out, function(p) p$Z))
  print(paste('size:', dim(Z.sample)))
  Sigma.sample <- lapply(samples.out, function(p) p$Sigma)

  #point estimate
  #use last 5000 sample
  mu.hat <- rowMeans(mu.sample[ , , 5000:samp.size+1], dims=2)

  #zhat we use round
  Z.hat <- round(colMeans(Z.sample[5000:samp.size+1, ]))

  Sigma.hat <- vector("list", K)
  for (i in 1:K){
    Sigmai <- lapply(Sigma.sample, function(p) p[[i]])
    Sigmai.hat <- array(unlist(Sigmai),
                        dim=c(3,3, samp.size+1))
    Sigma.hat[[i]] <- rowMeans(Sigmai.hat[ , , 5000:samp.size+1], dims=2)
  }

  #BIC
  #-2log(y|theta)+klog(n)
  #likelihood
  #&\prod_{i=1}^np(X_i|Z_i, \boldsymbol{\mu}, \mathbf{\Sigma}) \\
  #=&\prod_{t=1}^K \prod_{i:Z_i=t} f(X_i|\mu_t,\Sigma_t)
  likelihood <- 1:K
  likelihood <- -2*sum(log(sapply(likelihood,
                                function(t) prod(dmvnorm(X[Z.hat==t,],
                                                         mean=mu.hat[t,],
                                                         sigma=Sigma.hat[[t]])))))
  bic <- likelihood + log(nrow(X))*(
    K*3+K*6+K-1 #mu #sigma #p
  )

  return (list(bic=bic,
               mu.hat=mu.hat,
               Z.hat=Z.hat,
               Sigma.hat=Sigma.hat))
}

res3 = sampling_4_fix_K(X=mixgauss, K=3)
```

```
## [1] "size: 10001" "size: 300"
```

```
# cluster means, each row represents a cluster
res3$mu.hat
```

```
##           [,1]      [,2]      [,3]
## [1,]  3.952613  6.018458  1.951767
## [2,] -1.962100 -1.883564 -4.160951
## [3,] -4.040665 -1.963702  1.992996
```

```
# cluster labels
table(res3$Z.hat)
```

```
##
##   1   2   3
## 123  92  85
```

```
# cluster covariance matrices
res3$Sigma.hat
```

```
## [[1]]
##           [,1]      [,2]      [,3]
## [1,] 0.9557901 0.2625266 0.8407938
## [2,] 0.2625266 0.3768503 0.1656895
## [3,] 0.8407938 0.1656895 0.7539312
##
## [[2]]
##            [,1]        [,2]        [,3]
## [1,]  0.8572315  0.19690899 -0.13154095
## [2,]  0.1969090  0.99241690 -0.01537281
## [3,] -0.1315410 -0.01537281  0.88346523
##
## [[3]]
##           [,1]      [,2]      [,3]
## [1,] 0.4282743 0.4784067 0.3460749
## [2,] 0.4784067 1.1079191 0.5356734
## [3,] 0.3460749 0.5356734 0.3227038
```

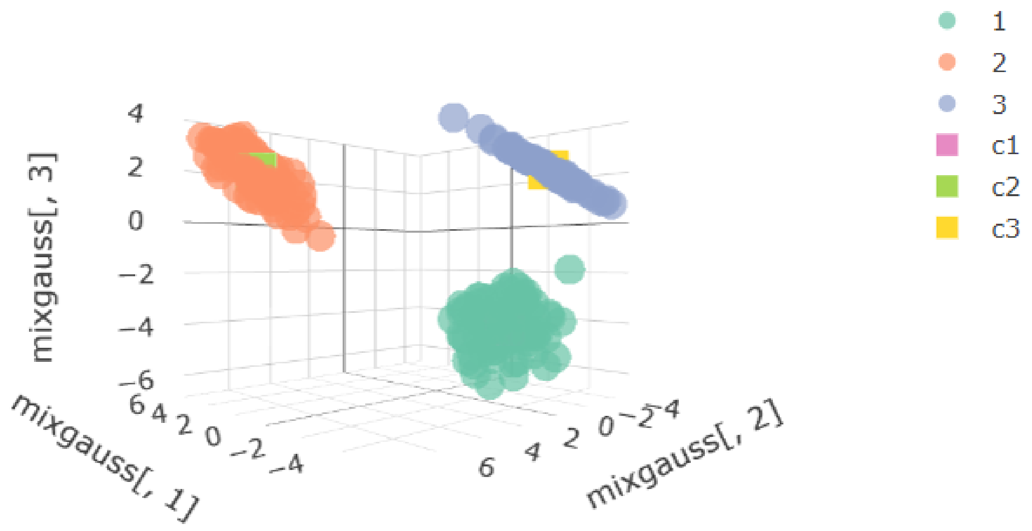


Figure 1: Three clusters

Problem 3 (Andrew)

part a

```r
setwd("~/FALL 2023/STATS 551")
crime <- read.csv("crime.dat", sep="")
```

```r
bayes.regression = function(niter=1000, y, X, n, g, v0, prior.variance) {
  SSR.g = t(y) %*% (diag(n) - g/(g+1) * X %*% solve(t(X)%*%X) %*% t(X)) %*% y
  sigma.sq = invgamma::rinvgamma(niter,(v0+n)/2, (v0*prior.variance+SSR.g)/2)

  betas = matrix(NA, nrow=niter, ncol=ncol(X))
  colnames(betas) = colnames(X)
  for (i in 1:niter) {
    V = (g/(g+1)) * sigma.sq[i] * solve(t(X) %*% X)
    m = (g/(g+1)) * solve(t(X) %*% X) %*% t(X) %*% y
    beta = MASS::mvrnorm(1, mu=m, Sigma=V)
    betas[i,] = beta
  }
  return(betas)
}
```

```r
n = nrow(crime)
g = n
v0 = 2
prior.variance = 1
```

```r
y = crime$y
X = as.matrix(crime[,-1])

niter = 10000

set.seed(42)
betas = bayes.regression(niter=10000, y, X, n, g, v0, prior.variance)
```

```r
colMeans(betas)
```

```
##             M            So            Ed           Po1           Po2 
##  0.2787699021  0.0002266249  0.5333546383  1.4299547462 -0.7555178942 
##            LF           M.F           Pop            NW            U1 
## -0.0638176261  0.1296530294 -0.0695335401  0.1073401064 -0.2655550179 
##            U2           GDP          Ineq          Prob          Time 
##  0.3619633425  0.2323263680  0.7111529918 -0.2796994876 -0.0597217395
```

```r
get.conf.int = function(col,lower,upper) {
  return(quantile(col, c(lower,upper)))
}
t(apply(betas, 2, get.conf.int, lower=.025, upper=.975))
```

```
##             2.5%       97.5%
## M     0.03317834  0.52785656
## So   -0.33230841  0.34063230
## Ed    0.21514881  0.86026222
## Po1  -0.02167809  2.90163628
## Po2  -2.27973640  0.78040962
## LF   -0.33700669  0.21810718
## M.F  -0.15227257  0.41049670
## Pop  -0.29902081  0.16210355
## NW   -0.20320062  0.41769367
## U1   -0.62111133  0.08504625
## U2    0.03347690  0.68954606
## GDP  -0.23206890  0.70368887
## Ineq  0.28558171  1.14446126
## Prob -0.51716969 -0.03840211
## Time -0.29810149  0.17850705
```

The percentage of males aged 14-24, mean years of schooling, unemployment rate of urban males 35-39, income inequality and probability of imprisonment seem strongly predictive of crime rates, as their credible intervals do not contain 0.

```r
summary(lm(y ~ X))
```

```
## 
## Call:
## lm(formula = y ~ X)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.02571 -0.26223 -0.01598  0.29013  1.33038
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0004581  0.0789333  -0.006  0.99541
## XM           0.2865181  0.1357981   2.110  0.04304 *
## XSo         -0.0001140  0.1840530  -0.001  0.99951
## XEd          0.5445141  0.1796106   3.032  0.00488 **
## XPo1         1.4716211  0.8166435   1.802  0.08127 .
## XPo2        -0.7817801  0.8515935  -0.918  0.36570
## XLF         -0.0659646  0.1534476  -0.430  0.67025
## XM.F         0.1312980  0.1551792   0.846  0.40398
## XPop        -0.0702919  0.1269186  -0.554  0.58367
## XNW          0.1090567  0.1719187   0.634  0.53051
## XU1         -0.2705364  0.1966309  -1.376  0.17872
## XU2          0.3687303  0.1798586   2.050  0.04889 *
## XGDP         0.2380595  0.2589534   0.919  0.36503
## XIneq        0.7262920  0.2341502   3.102  0.00408 **
## XProb       -0.2852264  0.1337912  -2.132  0.04105 *
## XTime       -0.0615769  0.1310241  -0.470  0.64167
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5411 on 31 degrees of freedom
## Multiple R-squared:  0.8029, Adjusted R-squared:  0.7075
## F-statistic: 8.418 on 15 and 31 DF,  p-value: 3.59e-07
```

The posterior means also seem to match up to the least squares estimates. The number of significant predictors also checks out.

part b

```r
set.seed(42)
rows = sample(n, .5*n)
data.train = crime[rows,]
data.test = crime[-rows,]

y.train = data.train$y
X.train = as.matrix(data.train[,-1])

y.test = data.test$y
X.test = as.matrix(data.test[,-1])
```

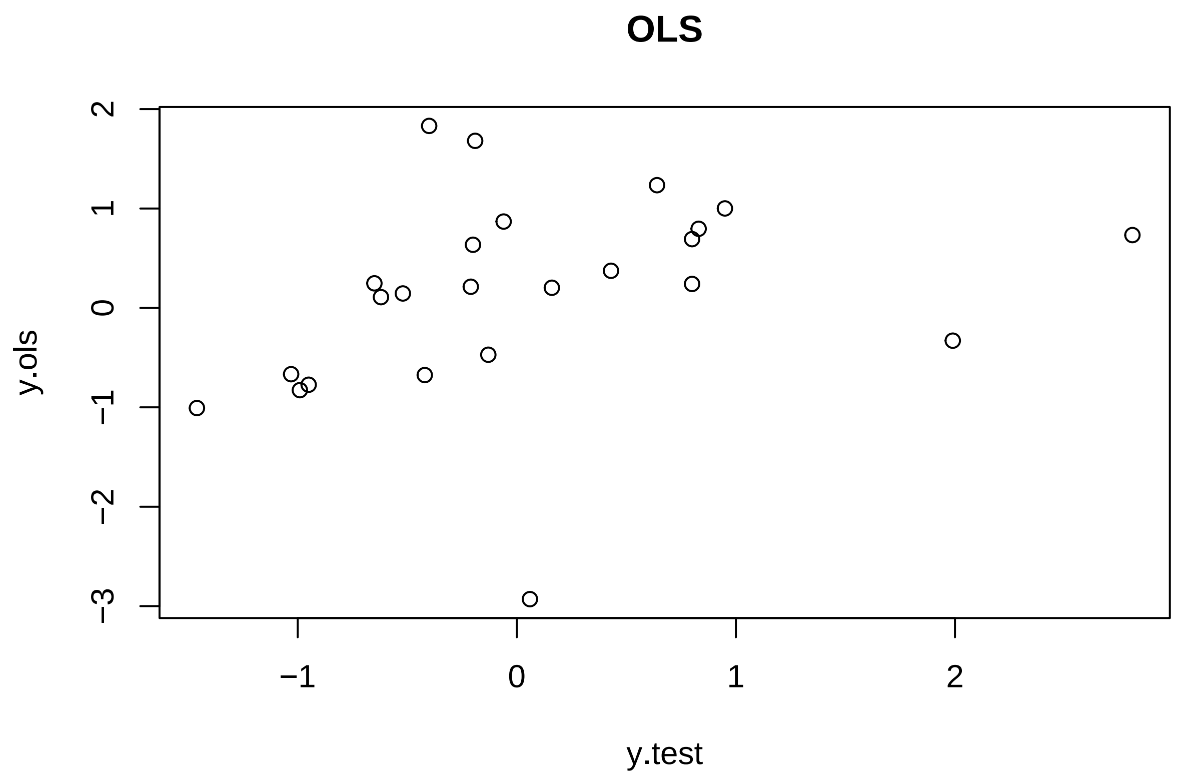
b1

```r
model = lm(y.train ~ X.train)
summary(model)
```

```
##
```

```
## Call:
## lm(formula = y.train ~ X.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.38242 -0.13022 -0.07314  0.09259  0.51747
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.03569    0.09853   0.362  0.72785
## X.trainM     0.68003    0.18955   3.588  0.00889 **
## X.trainSo    0.15909    0.23514   0.677  0.52041
## X.trainEd    0.87760    0.21619   4.059  0.00481 **
## X.trainPo1   1.07009    1.29180   0.828  0.43479
## X.trainPo2  -0.75741    1.21989  -0.621  0.55435
## X.trainLF    0.08488    0.25124   0.338  0.74538
## X.trainM.F  -0.30878    0.24008  -1.286  0.23930
## X.trainPop   0.14949    0.15528   0.963  0.36775
## X.trainNW    0.13220    0.15043   0.879  0.40867
## X.trainU1   -0.23068    0.23349  -0.988  0.35608
## X.trainU2    0.51547    0.20497   2.515  0.04011 *
## X.trainGDP  -0.54340    0.35040  -1.551  0.16488
## X.trainIneq  0.13663    0.36527   0.374  0.71944
## X.trainProb -1.05807    0.29179  -3.626  0.00844 **
## X.trainTime -0.53269    0.15968  -3.336  0.01249 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3659 on 7 degrees of freedom
## Multiple R-squared:  0.9609, Adjusted R-squared:  0.877
## F-statistic: 11.46 on 15 and 7 DF,  p-value: 0.001617
```

```r
y.ols = as.matrix(X.test) %*% model$coefficients[-1] + model$coefficients[1]
plot(y.ols ~ y.test, xlab="y.test",main="OLS")
```



The predictions seem to be roughly positive with a few outliers.

```
mean((y.ols - y.test)^2)
```

```
## [1] 1.328702
```

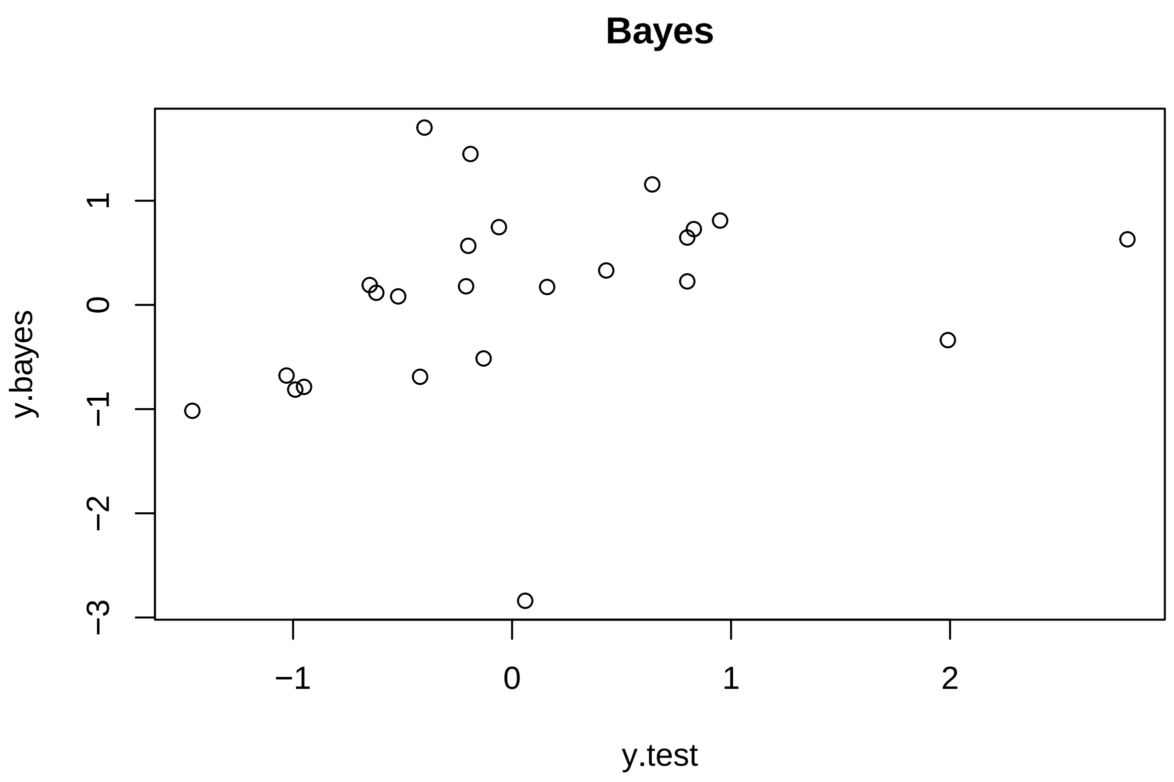
b2

```
n = nrow(X.train)
g = n
v0 = 2
prior.variance = 1

niter = 10000
betas.2 = bayes.regression(niter=10000, y.train, X.train,n,g,v0,prior.variance)
```

```
colMeans(betas.2)
```

```
##           M          So          Ed         Po1         Po2          LF
##  0.62541230  0.13147796  0.86530306  0.96971870 -0.69184924  0.04920529
##         M.F         Pop          NW          U1          U2         GDP
## -0.27948740  0.15784734  0.13781289 -0.23538010  0.49475800 -0.49793695
##        Ineq        Prob        Time
##  0.17702109 -1.00425491 -0.51671710
```

```r
y.bayes = as.matrix(X.test) %*% colMeans(betas.2)
plot(y.bayes ~ y.test, xlab="y.test",main="Bayes")
```



```r
mean((y.bayes - y.test)^2)
```

```
## [1] 1.247651
```

This is smaller than the MSE acquired from the OLS regression. The plot looks almost exactly identical to that from the OLS.

b3

```r
reps = 10
mse.ols = mse.bayes = vector(length=reps)

v0 = 2
prior.variance = 1

for (i in 1:reps) {
  rows = sample(nrow(crime), .5*nrow(crime))
  data.train = crime[rows,]
  data.test = crime[-rows,]
  y.train = data.train$y
  X.train = as.matrix(data.train[,-1])

  y.test = data.test$y
  X.test = as.matrix(data.test[,-1])

  n = nrow(X.train)
  g = n

  model = lm(y.train ~ X.train)
  y.ols = as.matrix(X.test) %*% model$coefficients[-1] + model$coefficients[1]

  betas.2 = bayes.regression(niter=10000, y.train, X.train,n,g,v0,prior.variance)
  y.bayes = as.matrix(X.test) %*% colMeans(betas.2)

  mse.ols[i] = mean((y.ols - y.test)^2)
  mse.bayes[i] = mean((y.bayes - y.test)^2)
}
```

```r
paste0("Average OLS MSE: ", round(mean(mse.ols),2))
```

```
## [1] "Average OLS MSE: 1.33"
```

```r
paste0("Average Bayes MSE: ", round(mean(mse.bayes),2))
```

```
## [1] "Average Bayes MSE: 1.03"
```

Problem 4 (Minxuan, Han)

```r
msparrownest <- read.table("~/FALL 2023/STATS 551/msparrownest.dat", quote="\"", comment.char="")
```

part a

$$\log\left(\frac{p}{1-p}\right) = \alpha + \beta x_i$$
$$e^{\alpha+\beta x_i} = \frac{p}{1-p}$$
$$p = \frac{e^{\alpha+\beta x_i}}{1+e^{\alpha+\beta x_i}}$$

$$\prod_{i=1}^{n} p(y_i|\alpha,\beta,x_i) = p^{\sum_{i=1}^{n} y_i}(1-p)^{n-\sum_{i=1}^{n} y_i}$$
$$\log\prod_{i=1}^{n} p(y_i|\alpha,\beta,x_i) = \log\left[p^{\sum_{i=1}^{n} y_i}(1-p)^{n-\sum_{i=1}^{n} y_i}\right]$$

$$= \sum_{i=1}^{n}[y_i\log p + (1-y_i)\log(1-p)]$$

part b

We will endow the following prior:

$$p(\alpha, \beta) \sim N_2\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}100 & 0\\0 & 1\end{bmatrix}\right)$$

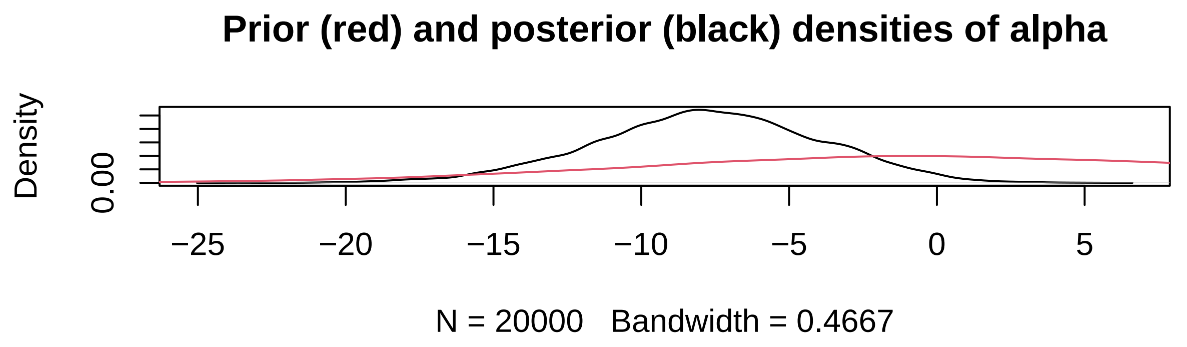
part c

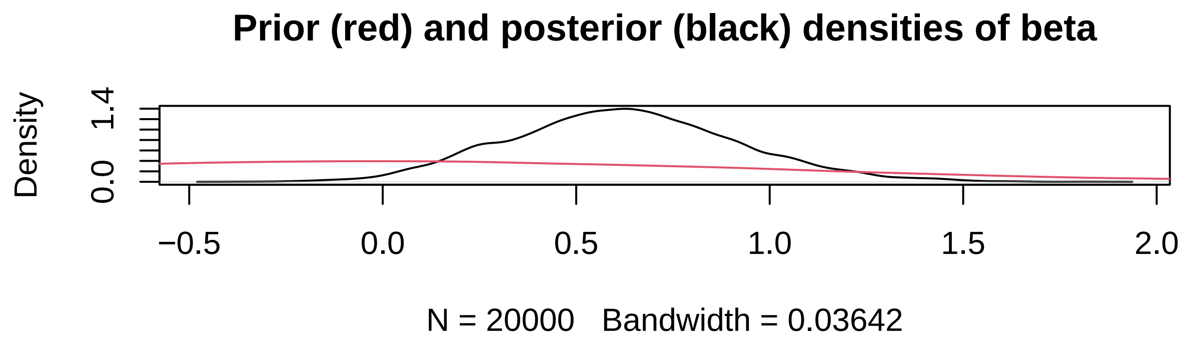
```r
# get.p = function(x, alpha, beta) {
#   return(exp(alpha + beta*x)/(1+exp(alpha + beta*x)))
# }

get.p = function(X, coeffs) {
  return(exp(X %*% coeffs)/(1+exp(X %*% coeffs)))
}

log.likelihood = function(y, X, coeffs) {
  p = get.p(X, coeffs)
  ans = sum(y * log(p) + (1 - y) * log(1 - p))
  return(ans)
}

metropolis.hastings = function(y, x, prior.mean, prior.sigma, niter=1000) {
  #alphas = betas = vector(length=niter)
  X = cbind(1, x)
  var.prop = 15 * solve(t(X) %*% X)

  res = matrix(NA, nrow=niter, ncol=2)
  colnames(res) = c("alpha","beta")
  res[1,] = prior.mean
  accepts = 0
  set.seed(42)
  for (i in 2:niter) {
    current = res[i-1,]
    proposed = mvtnorm::rmvnorm(1, mean=current, sigma=var.prop)

    log.lik.current = log.likelihood(y, X, current)
    log.lik.proposed = log.likelihood(y, X, as.vector(proposed))

    prob.current = mvtnorm::dmvnorm(current, mean=prior.mean, sigma=prior.sigma, log=T)
    prob.proposed = mvtnorm::dmvnorm(proposed, mean=prior.mean, sigma=prior.sigma, log=T)

    R = (log.lik.proposed-log.lik.current) + (prob.proposed-prob.current)

    if (log(runif(1)) < R) {
      res[i,] = proposed
      accepts = accepts + 1
      }
    else {res[i,] = res[i-1,]}
  }
  print(paste("Acceptance rate:", round(accepts/niter,3)))
  return(res)
}
```

```
y = msparrownest$V1
x = msparrownest$V2

# X = cbind(1, x)
# var.prop = 15 * solve(t(X) %*% X)

prior.mean = c(0,0)
prior.sigma = matrix(c(100,0,0,1),nrow=2,ncol=2)
prior.sd = c(10,1)

res.mh = metropolis.hastings(y,x,prior.mean,prior.sigma,niter=20000)
```

```
## [1] "Acceptance rate: 0.313"
```

part d





The posterior densities shift left from the priors due to the data and are not as bell-shaped (a few distinct peaks/modes) as the prior distributions are.

```
colMeans(res.mh)
```

```
##      alpha       beta
## -7.7957491  0.6221408
```

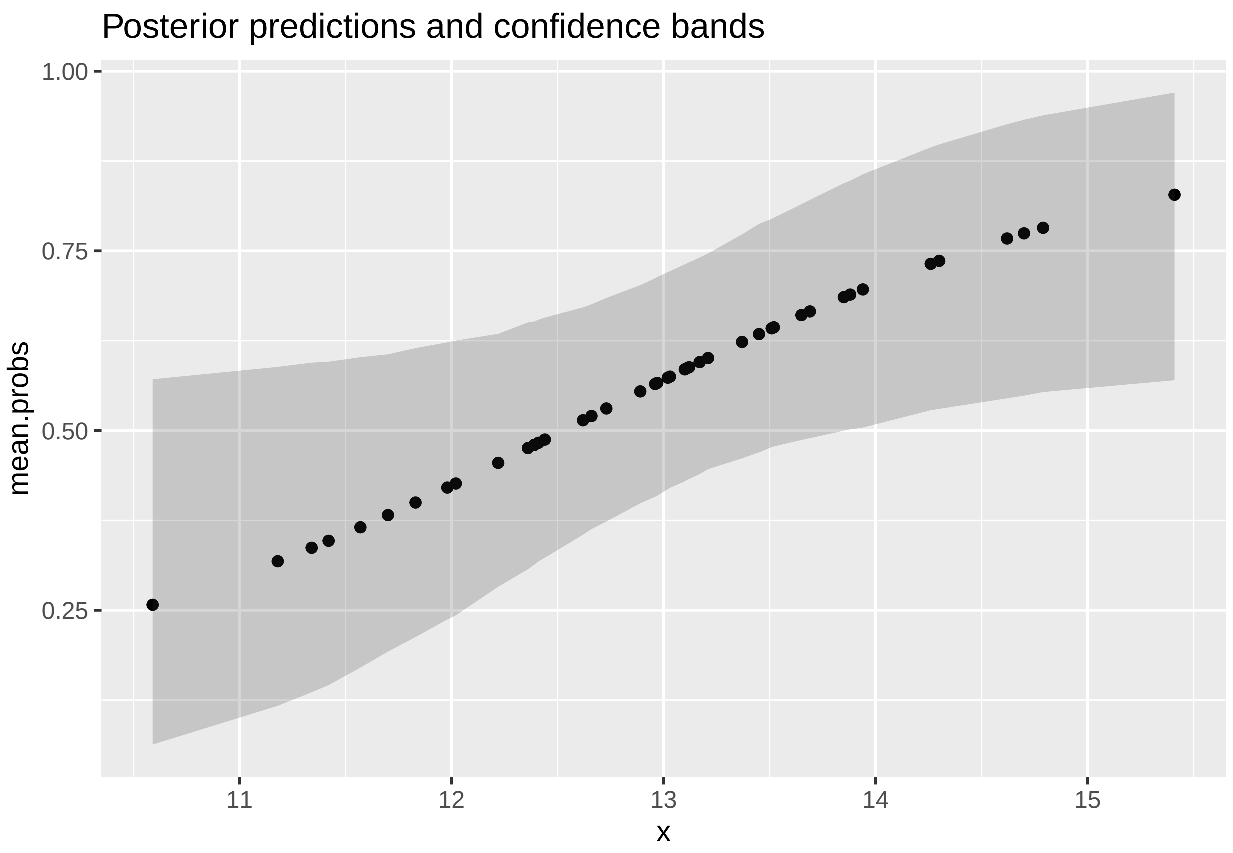
```
coda::effectiveSize(res.mh)
```

```
##    alpha     beta
## 2795.011 2790.193
```

part e

```
# f.ab = function(alpha, beta, x) {
#   return(1/(1+exp(-alpha-beta*x)))
# }

niter = 10000
probs.mat = matrix(NA, nrow=length(x),ncol=niter)
X = cbind(1, x)
for (i in 1:niter) {
  #probs.mat[,i] = f.ab(res.mh[i,1], res.mh[i,2], x)
  probs.mat[,i] = get.p(X,res.mh[i,])
}
```

```
library(ggplot2)

confints = t(apply(probs.mat, 1, get.conf.int, lower=.025, upper=.975))
mean.probs.df = data.frame(x = x, mean.probs = apply(probs.mat,1,mean),
                           lower = confints[,1], upper = confints[,2])

ggplot(mean.probs.df, aes(x, mean.probs)) +
  geom_point()+
  geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) +
  ggtitle(label="Posterior predictions and confidence bands")
```



The predictions in the middle seem to have narrower confidence bands and they roughly follow a logistic function. It would seem that in general, the bigger the wingspan, the more likely they nest.