- Problem B: Bayesian Image Reconstruction

We study how the ising model can be used as a prior distribution in an image reconstruction setting. We let $y = (y_{ij}, i = 1, \ldots, 89, j = 1, \ldots, 85)$ be the observed "image.txt". We assume this to be a noisy version of an unobserved image $x = (x_{ij}, i = 1, \ldots, 89, j = 1, \ldots, 85)$ with $x_{ij} \in \{0, 1\}$. Our goal in this problem is to use the observed y to estimate x. We assume the elements in y to be conditionally independent given x and

$$y_{ij}|x \sim N(\mu_{x_{ij}}, \sigma^2_{x_{ij}}) \tag{1}$$

where μ_0, μ_1 are the mean values for y_{ij} when x_{ij} is zero and one, respectively and σ_0^2 and σ_1^2 are corresponding variances. Apriori we assume x to be independent of $\varphi = (\mu_0, \mu_1, \sigma_0, \sigma_1)$. As prior for x we assume an Ising model with interaction parameter β, i.e.

$$f(x) \propto \exp\Big\{\beta \sum_{(i,j)\sim(k,l)} I(x_{ij} = x_{kl})\Big\} \tag{2}$$

where the sum is over all pairs of neighbour nodes in the 89×85 lattice and the value of β is assumed to be known. To define a prior for φ we follow a procedure used in Austad and Tjelmeland (2017). We first define a reparametrisation to new parameters (m_0, θ, s, τ) by the relations

$$\sigma_0 = s \cdot \tau,\ \sigma_1 = \frac{s}{\tau},\ \mu_0 = m_0, \text{ and } \mu_1 = m_0 + s\theta$$

The s defines a scale, θ defines the difference between the two mean values in this scale, and τ defines σ_0 and σ_1 using the same scale. We then define a prior for $\varphi = (\mu_0, \mu_1, \sigma_0, \sigma_1)$ implicitly by assigning a prior for (m_0, θ, s, τ)

$$f(\tau) = \begin{cases} \frac{1}{2\tau^2} e^{-(\frac{1}{\tau} - 1)} & \text{for } \tau \in (0, 1], \\ \frac{1}{2} e^{-(\tau - 1)} & \text{for } \tau > 1. \end{cases}$$

We use the transformation formula to find the corresponding prior for $t = 1/\tau$. We let $t = 1/\tau$, such that $\tau = 1/t = w(t)$. The pdf $g(t)$ for t will then be given by

$$g(t) = f(w(t)) \cdot |w'(t)|$$

which gives

$$g(t) = \begin{cases} \frac{1}{2} t^2 e^{-(t-1)} \cdot \frac{1}{t^2} & \text{for } t \geq 1 \\ \frac{1}{2} e^{-(t-1)} \cdot \frac{1}{t^2} & \text{for } t \in (0, 1) \end{cases} = \begin{cases} \frac{1}{2} e^{-(t-1)} & \text{for } t \geq 1 \\ \frac{1}{2t^2} e^{-(\frac{1}{t} - 1)} & \text{for } t \in (0, 1) \end{cases}$$

Notice how the intervals changes and that the expressions for $f(\tau)$ and $g(t)$ are the same. Hence, the priors for τ and t are identical and we can use this to argue that the marginal priors for σ_0 and σ_1 are identical. This is easily shown since $\sigma_0 = s \cdot \tau$ and $\sigma_1 = \frac{s}{\tau}$.

2. We show that the resulting (improper) prior for $\varphi = (\mu_0, \mu_1, \sigma_0, \sigma_1)$ becomes (up to proportionality)

$$f(\varphi) \propto \begin{cases} \frac{(\mu_1 - \mu_0)^3}{\sigma_0^3 \sigma_1^2} \exp\{-\Big[\frac{\mu_1 - \mu_0}{\sqrt{\sigma_0 \sigma_1}} + \sqrt{\frac{\sigma_1}{\sigma_0}}\Big]\} & \text{for } \sigma_0 \leq \sigma_1 \text{ and } \mu_0 < \mu_1 \\ \frac{(\mu_1 - \mu_0)^3}{\sigma_0^2 \sigma_1^3} \exp\{-\Big[\frac{\mu_1 - \mu_0}{\sqrt{\sigma_0 \sigma_1}} + \sqrt{\frac{\sigma_0}{\sigma_1}}\Big]\} & \text{for } \sigma_0 > \sigma_1 \text{ and } \mu_0 < \mu_1 \end{cases}$$

We must use the transformation formula with 4 variables. We find that

$$\begin{aligned} m_0 &= w_1(\mu_0, \mu_1, \sigma_0, \sigma_1) = \mu_0 \\ s &= w_2(\mu_0, \mu_1, \sigma_0, \sigma_1) = \sqrt{\sigma_0 \sigma_1} \\ \tau &= w_3(\mu_0, \mu_1, \sigma_0, \sigma_1) = \sqrt{\frac{\sigma_0}{\sigma_1}} \\ \theta &= w_4(\mu_0, \mu_1, \sigma_0, \sigma_1) = \frac{\mu_1 - \mu_0}{\sqrt{\sigma_0 \sigma_1}} \end{aligned}$$

We thus have

$$\begin{aligned} f(\varphi) = f(\mu_0, \mu_1, \sigma_0, \sigma_1) = g(w_1(\varphi), w_2(\varphi), w_3(\varphi), w_4(\varphi)) \cdot |J| \\ f(\varphi) = g(m_0, s, \tau, \theta) \cdot |J| \\ f(\varphi) = g_M(m_0) g_S(s) g_\tau(\tau) g_\Theta(\theta) \cdot |J| \end{aligned}$$

because of independence, where J is the 4×4 Jacobi determinant shown in (3). Since m_0 and s are assumed to be improper uniform distributed, we have that

$$f(\varphi) \propto g_\tau\Big(\sqrt{\frac{\sigma_0}{\sigma_1}}\Big) g_\Theta\Big(\frac{\mu_1 - \mu_0}{\sqrt{\sigma_0 \sigma_1}}\Big) \cdot |J|$$

The Jacobi determinant is given by

$$J = \begin{vmatrix} \frac{\partial m_0}{\partial \mu_0} & \frac{\partial m_0}{\partial \mu_1} & \frac{\partial m_0}{\partial \sigma_0} & \frac{\partial m_0}{\partial \sigma_1} \\ \frac{\partial s}{\partial \mu_0} & \frac{\partial s}{\partial \mu_1} & \frac{\partial s}{\partial \sigma_0} & \frac{\partial s}{\partial \sigma_1} \\ \frac{\partial \tau}{\partial \mu_0} & \frac{\partial \tau}{\partial \mu_1} & \frac{\partial \tau}{\partial \sigma_0} & \frac{\partial \tau}{\partial \sigma_1} \\ \frac{\partial \theta}{\partial \mu_0} & \frac{\partial \theta}{\partial \mu_1} & \frac{\partial \theta}{\partial \sigma_0} & \frac{\partial \theta}{\partial \sigma_1} \end{vmatrix} \tag{3}$$

which becomes

$$J = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2}\sqrt{\frac{\sigma_1}{\sigma_0}} & \frac{1}{2}\sqrt{\frac{\sigma_0}{\sigma_1}} \\ 0 & 0 & \frac{1}{2\sqrt{\sigma_0 \sigma_1}} & -\frac{1}{2}\sqrt{\frac{\sigma_0}{\sigma_1^3}} \\ -\frac{1}{\sqrt{\sigma_0 \sigma_1}} & \frac{1}{\sqrt{\sigma_0 \sigma_1}} & -\frac{(\mu_1 - \mu_0)}{2\sqrt{\sigma_0^3 \sigma_1}} & -\frac{(\mu_1 - \mu_0)}{2\sqrt{\sigma_0 \sigma_1^3}} \end{vmatrix}$$

This can now be reduced to

$$J = 1 \cdot \begin{vmatrix} 0 & \frac{1}{2}\sqrt{\frac{\sigma_1}{\sigma_0}} & \frac{1}{2}\sqrt{\frac{\sigma_0}{\sigma_1}} \\ 0 & \frac{1}{2\sqrt{\sigma_0 \sigma_1}} & -\frac{1}{2}\sqrt{\frac{\sigma_0}{\sigma_1^3}} \\ \frac{1}{\sqrt{\sigma_0 \sigma_1}} & -\frac{(\mu_1 - \mu_0)}{2\sqrt{\sigma_0^3 \sigma_1}} & -\frac{(\mu_1 - \mu_0)}{2\sqrt{\sigma_0 \sigma_1^3}} \end{vmatrix} = 1 \cdot \frac{1}{\sqrt{\sigma_0 \sigma_1}} \cdot \begin{vmatrix} \frac{1}{2}\sqrt{\frac{\sigma_1}{\sigma_0}} & \frac{1}{2}\sqrt{\frac{\sigma_0}{\sigma_1}} \\ \frac{1}{2\sqrt{\sigma_0 \sigma_1}} & -\frac{1}{2}\sqrt{\frac{\sigma_0}{\sigma_1^3}} \end{vmatrix} = -\frac{1}{2\sqrt{\sigma_0 \sigma_1^3}}$$

Hence, we get

$$f(\varphi) \propto g_\tau\Big(\sqrt{\frac{\sigma_0}{\sigma_1}}\Big) g_\Theta\Big(\frac{\mu_1-\mu_0}{\sqrt{\sigma_0\sigma_1}}\Big) \cdot |J| \propto \begin{cases} \frac{1}{2}\frac{\sigma_1}{\sigma_0} e^{-\left(\sqrt{\frac{\sigma_1}{\sigma_0}}-1\right)} \cdot \left(\frac{\mu_1-\mu_0}{\sqrt{\sigma_0\sigma_1}}\right)^3 e^{-\frac{\mu_1-\mu_0}{\sqrt{\sigma_0\sigma_1}}} \cdot \frac{1}{2\sqrt{\sigma_0\sigma_1^3}} & \text{for } \sigma_0 \le \sigma_1 \text{ and } \mu_0 < \mu_1 \\ \frac{1}{2} e^{-\left(\sqrt{\frac{\sigma_0}{\sigma_1}}-1\right)} \cdot \left(\frac{\mu_1-\mu_0}{\sqrt{\sigma_0\sigma_1}}\right)^3 e^{-\frac{\mu_1-\mu_0}{\sqrt{\sigma_0\sigma_1}}} \cdot \frac{1}{2\sqrt{\sigma_0\sigma_1^3}} & \text{for } \sigma_0 > \sigma_1 \text{ and } \mu_0 < \mu_1 \end{cases}$$

$$f(\varphi) \propto \begin{cases} \frac{(\mu_1-\mu_0)^3}{\sigma_0^3\sigma_1^2} \exp\{-\left[\frac{\mu_1-\mu_0}{\sqrt{\sigma_0\sigma_1}} + \sqrt{\frac{\sigma_1}{\sigma_0}}\right]\} & \text{for } \sigma_0 \le \sigma_1 \text{ and } \mu_0 < \mu_1 \\ \frac{(\mu_1-\mu_0)^3}{\sigma_0^2\sigma_1^3} \exp\{-\left[\frac{\mu_1-\mu_0}{\sqrt{\sigma_0\sigma_1}} + \sqrt{\frac{\sigma_0}{\sigma_1}}\right]\} & \text{for } \sigma_0 > \sigma_1 \text{ and } \mu_0 < \mu_1 \end{cases}$$

which is what we wanted to show.

3. We find (up to proportionality) a formula for the posterior distribution $f(x, \varphi|y)$. We thus have

$$f(x,\varphi|y) \propto f(x,\varphi,y) \propto f(y|x,\varphi)\cdot f(x,\varphi) \propto f(y|x)\cdot f(x,\varphi) \propto f(y|x)\cdot f(x)\cdot f(\varphi)$$

All of these distributions are known to us from equations (1), (2) and ().

$$f(x,\varphi|y) \propto \prod_{i,j} f(y_{ij}|x)\cdot f(x)\cdot f(\varphi) \propto \left(\prod_{i,j} \frac{1}{\sigma_{x_{ij}}} e^{-\frac{1}{2\sigma^2_{x_{ij}}}\left(y_{ij}-\mu_{x_{ij}}\right)^2}\right)\cdot \exp\Big\{\beta \sum_{(i,j)\sim(k,l)} I(x_{ij}=x_{kl})\Big\}\cdot \begin{cases} \frac{(\mu_1-\mu_0)^3}{\sigma_0^3\sigma_1^2} \exp\{-\left[\frac{\mu_1-\mu_0}{\sqrt{\sigma_0\sigma_1}} + \right. \\ \frac{(\mu_1-\mu_0)^3}{\sigma_0^2\sigma_1^3} \exp\{-\left[\frac{\mu_1-\mu_0}{\sqrt{\sigma_0\sigma_1}} + \end{cases}$$

$$\frac{f(\tilde{x},\varphi|y)}{f(x,\varphi|y)} = \frac{\frac{1}{\sqrt{2\pi}\sigma_{\tilde{x}_{ij}}} e^{-\frac{1}{2\sigma^2_{\tilde{x}_{ij}}}\left(y_{ij}-\mu_{\tilde{x}_{ij}}\right)^2} \cdot \exp\{\beta \sum_{(i,j)\sim(k,l)} I(\tilde{x}_{ij}=x_{kl})\}}{\frac{1}{\sqrt{2\pi}\sigma_{x_{ij}}} e^{-\frac{1}{2\sigma^2_{x_{ij}}}\left(y_{ij}-\mu_{x_{ij}}\right)^2} \cdot \exp\{\beta \sum_{(i,j)\sim(k,l)} I(x_{ij}=x_{kl})\}}$$

$$= \frac{\sigma_{x_{ij}}}{\sigma_{\tilde{x}_{ij}}} e^{-\frac{1}{2\sigma^2_{\tilde{x}_{ij}}}\left(y_{ij}-\mu_{\tilde{x}_{ij}}\right)^2 + \frac{1}{2\sigma^2_{x_{ij}}}\left(y_{ij}-\mu_{x_{ij}}\right)^2} \cdot \exp\left\{\beta \sum_{(i,j)\sim(k,l)} (I(\tilde{x}_{ij}=x_{kl}) - I(x_{ij}=x_{kl}))\right\}$$

$$\begin{aligned} s &\sim Uniform(-\infty,\infty) \\ m_0 &\sim Uniform(0,\infty) \\ \theta &\sim Gamma(4,1) \\ \tau &(next\ slide) \end{aligned}$$

4.

c) We propose a potential new value for σ_0 by simulating σ_0^2 from the resulting conditional for σ_0^2. We assume an inverse gamma prior for σ_0^2 with parameters a_0 and b_0. We thus have

$$f(\sigma_0^2|x,y,\mu_0,\mu_1,\sigma_1) \propto f(y|x)f(\sigma_0^2) \propto \prod_{x_{ij}=0} \Big(\frac{1}{\sigma_{x_{ij}}} e^{-\frac{1}{2\sigma^2_{x_{ij}}}(y_{ij}-\mu_{x_{ij}})^2}\Big)\cdot \frac{1}{(\sigma_0^2)^{a_0+1}} e^{-\frac{b_0}{\sigma_0^2}} \propto \Big(\frac{1}{\sigma_0^2}\Big)^{N_0/2} e^{-\sum_{i=1}^{N_0}\frac{1}{2\sigma_0^2}(y_{ij}-\mu_0)^2}\cdot \frac{1}{(\sigma_0^2)^{a_0+1}} e^{-\frac{b_0}{\sigma_0^2}}$$

where N_0 is the number of nodes that satisfy $x_{ij}=0$. Similarly for σ_1^2, we find that

$$f(\sigma_1^2|x,y,\mu_0,\mu_1,\sigma_0) \sim InvGamma\Big(a_1 + N_1/2, b_1 + \sum_{i=1}^{N_1} \frac{(y_{ij}-\mu_1)^2}{2}\Big)$$

where we assume an inverse gamma prior for σ_1^2 with parameters a_1 and b_1.

Acceptance probability for σ_0^2.

We define and implement a Metropolis-Hastings algorithm for simulating from $f(x, \varphi|y)$

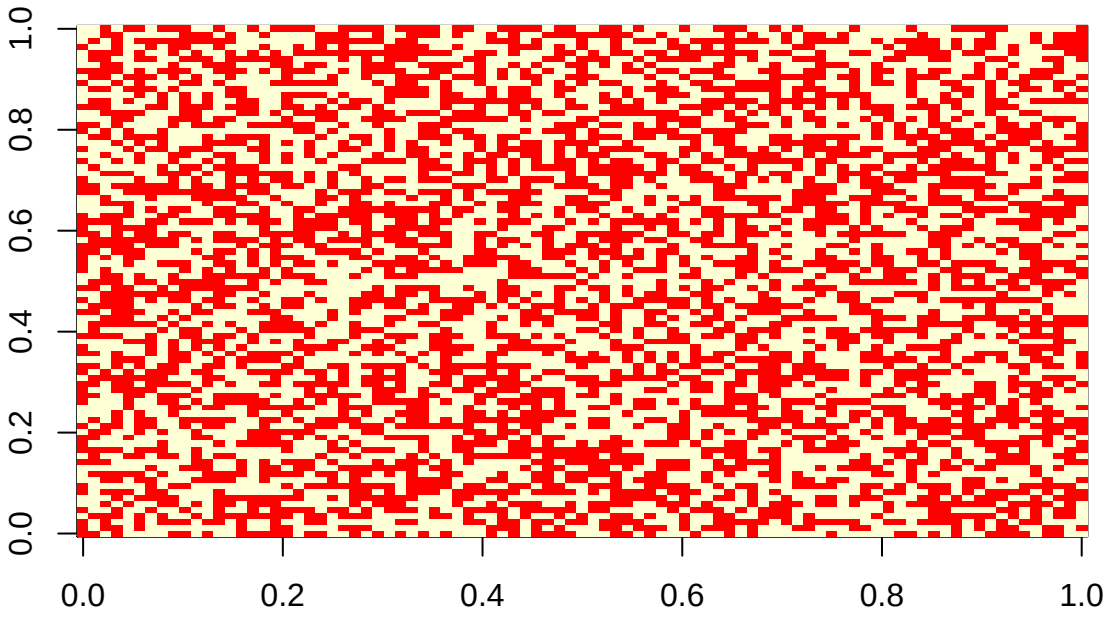
```
x_prior = function(x) {
  nrows = dim(x)[1]
  ncolumns = dim(x)[2]
  indicator_vec = matrix(c(0), nrow = nrows, ncol = ncolumns)
  for (i in 1:nrows) {
    for(j in 1:ncolumns) {
      indicator_vec[i,j] = indicator_x(x,i,j)
    }
  }
  return(sum(indicator_vec))
}

indicator_x = function(x,i,j) {
  nrows = dim(x)[1]
  ncolumns = dim(x)[2]
  if (i > 1 && i < nrows && j > 1 && j < ncolumns) {
    right = x[i,j] == x[i+1,j]
    left = x[i,j] == x[i-1,j]
    up = x[i,j] == x[i,j-1]
    down = x[i,j] == x[i,j+1]
    indicator = right + left + up + down
  }
  else if (i == 1 && j == 1) {
    right = x[i,j] == x[i+1,j]
    down = x[i,j] == x[i,j+1]
    indicator = right + down
  }
  else if (i == 1 && j == ncolumns) {
    right = x[i,j] == x[i+1,j]
    up = x[i,j] == x[i,j-1]
    indicator = right + up
  }
  else if (i == nrows && j == 1) {
    left = x[i,j] == x[i-1,j]
    down = x[i,j] == x[i,j+1]
    indicator = left + down
  }
  else if (i == nrows && j == ncolumns) {
    left = x[i,j] == x[i-1,j]
    up = x[i,j] == x[i,j-1]
    indicator = left + up
  }
  else if (i == 1) {
    right = x[i,j] == x[i+1,j]
    up = x[i,j] == x[i,j-1]
    down = x[i,j] == x[i,j+1]
    indicator = right + up + down
  }
  else if (i == nrows) {
    left = x[i,j] == x[i-1,j]
    up = x[i,j] == x[i,j-1]
    down = x[i,j] == x[i,j+1]
    indicator = left + up + down
```

```
  }
  else if (j == 1) {
    right = x[i,j] == x[i+1,j]
    left = x[i,j] == x[i-1,j]
    down = x[i,j] == x[i,j+1]
    indicator = right + left + down
  }
  else if (j == ncolumns) {
    right = x[i,j] == x[i+1,j]
    left = x[i,j] == x[i-1,j]
    up = x[i,j] == x[i,j-1]
    indicator = right + left + up
  }
return(indicator)
}
```

```
y = read.table("./image.txt", header = FALSE, sep = " ")
nrows = dim(y)[1]
ncolumns = dim(y)[2]
sigma_0 = 0.5
sigma_1 = 0.5
mu_0 = -1
mu_1 = 2
beta = 1

x = matrix(rbinom(nrows * ncolumns, 1, 0.5), ncol = ncolumns, nrow = nrows)
image(z = x)
```

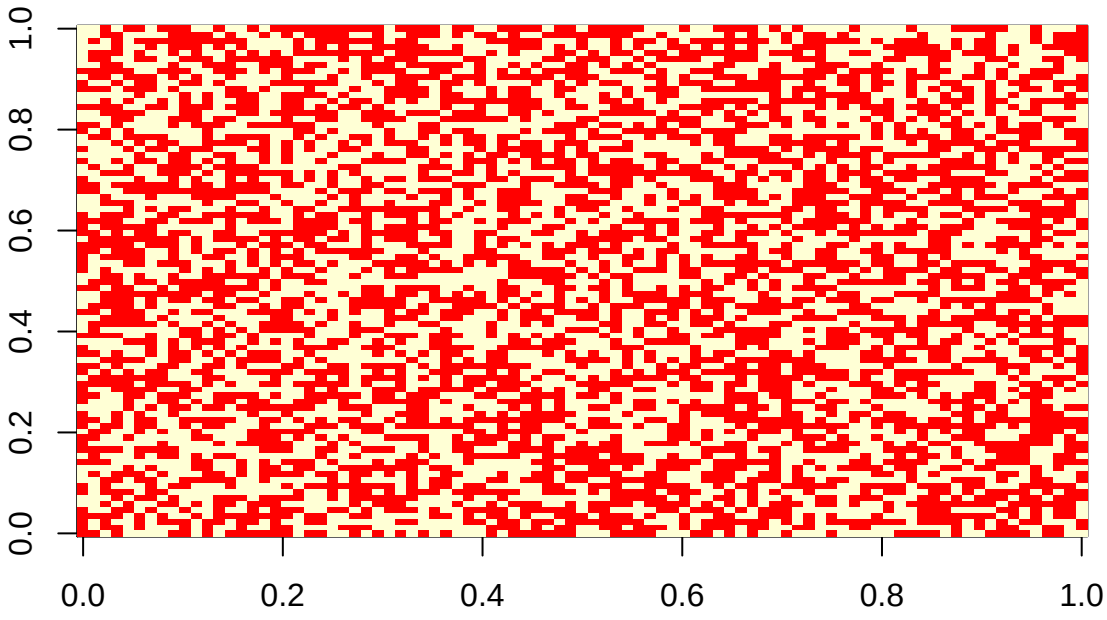



```
x_update <- function(times, x, sigma_0, sigma_1, mu_0, mu_1, beta){
  for (it in 1:times){
    i = ceiling(nrows*runif(1))
    j = ceiling(ncolumns*runif(1))
    x_prop = 1 - x[i,j]
    x_new = x
```

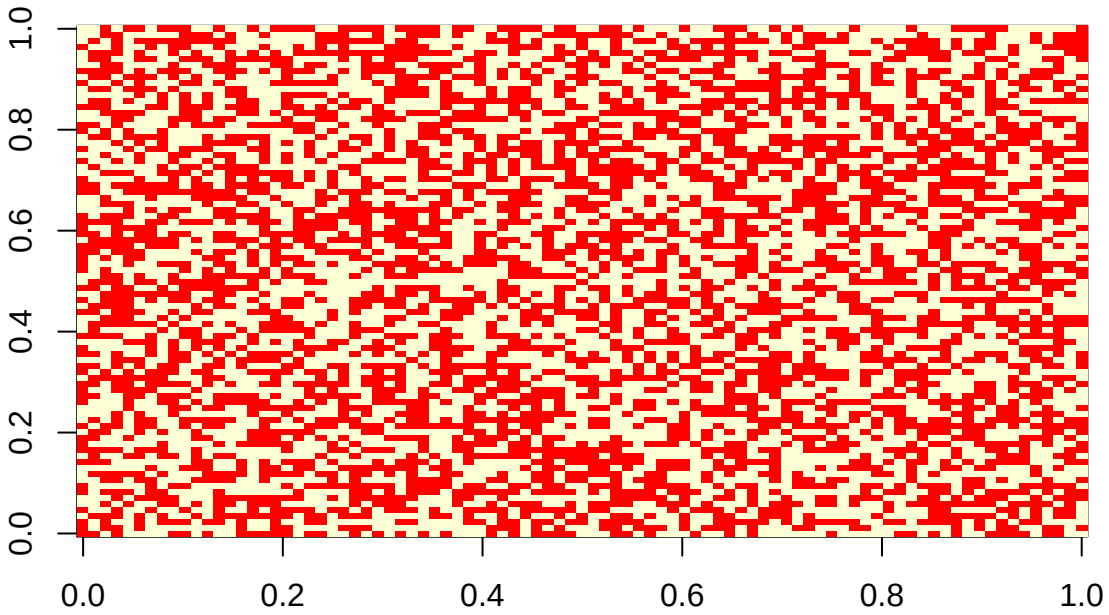
```
    x_new[i,j] = x_prop
    I = x_prior(x) # No. of neeighbours
    I_prop = x_prior(x_new)

    ising = exp(beta*(I_prop-I))
    if (x[i,j] == 0) {
      normal = sigma_0/sigma_1 * exp(-1/(2*sigma_1^2)*(y[i,j]-mu_1)^2 + 1/(2*sigma_0)*(y[i,j]-mu_0)^2)
    }
    if(x[i,j] == 1){
      normal = sigma_1/sigma_0 * exp(-1/(2*sigma_0^2)*(y[i,j]-mu_0)^2 + 1/(2*sigma_1)*(y[i,j]-mu_1)^2)
    }
    fratio = normal * ising
    alpha = min(1,fratio)
    u = runif(1)
    if (u < alpha) {
      x[i,j] = x_prop
    }
  }
  return(x)
}

x_upd <- x_update(1000, x, sigma_0, sigma_1, mu_0, mu_1, beta)
image(z = x_upd)
```




```
image(z = x)
```




```r
# Count number of 0s or ones in x
# and return sum of corresponding values in y
fcount <- function(x, y, bool){
  nrows <- dim(x)[1]
  ncols <- dim(x)[2]
  count <- 0
  yval = 0
  for (i in 1:nrows){
    for(j in 1:ncols){
      if (x[i,j] == bool){
        count <- count + 1
        yval[count] <- y[i,j]
      }
    }
  }
  ysum <- sum(yval)
  res <- data.frame(count, ysum)
  return(res)
}
# b) For mu_0
res0 <- fcount(x,y,0)
N_0 <- res0$count
ysum0 <- res0$ysum
mu0_old <- mu_0

## Acceptance probability
# ---- Utilities --- #
ftheta <- function(mu_0, mu_1, sigma_0, sigma_1){
  if (sigma_0 <= sigma_1 && mu_0 < mu_1){
    res <- (((mu_1-mu_0)^3)/(sigma_0^3*sigma_1^2))*exp(-(((mu_1-mu_0)/(sqrt(sigma_0*sigma_1)))+sqrt(sig
  }
  else if (sigma_0 > sigma_1 && mu_0 < mu_1){
    res <- (((mu_1-mu_0)^3)/(sigma_0^2*sigma_1^3))*exp(-(((mu_1-mu_0)/(sqrt(sigma_0*sigma_1)))+sqrt(sig
  }
```

```
  return(res)
}


fygivenxphi <- function(val, mu, sigma){
  return(dnorm(val, mu, sigma))
}

# --- Main---- #
muzero <- function(N_0, ysum0, sigma_0, mu0_old){
  # Propose random new mu0
  mu0_new <- rnorm(1, ysum0/N_0, sqrt((sigma_0^2)/N_0))
  Q_old <- dnorm(mu0_old, ysum0/N_0, sqrt((sigma_0^2)/N_0), log = TRUE)
  Q_new <- dnorm(mu0_new, ysum0/N_0, sqrt((sigma_0^2)/N_0), log = TRUE)
  f_old <- log(ftheta(mu0_old, mu_1, sigma_0, sigma_1)*fygivenxphi(mu0_old, mu_0, sigma_0))
  f_new <- log(ftheta(mu0_new, mu_1, sigma_0, sigma_1)*fygivenxphi(mu0_new, mu_0, sigma_0))


  # NB: Ratio on log scale
  ratio <- (f_new*Q_old)/(f_old*Q_new)
  acc <- min(0, ratio) # on log scale
  u <- log(runif(1))
  if (u < acc){
    res <- mu0_new
  } else {
    res <- mu0_new
  }
  return(res)
}

test0 <- muzero(N_0, ysum0, sigma_0, mu0_old)
test0
```

```
## [1] -0.877887
```

```
# For mu_1
res <- fcount(x,y,1)
N_1 <- res$count
ysum1 <- res$ysum
mu1_old <- mu_1

muone <- function(N_1, ysum1, sigma_1, mu1_old){
  # Propose random new mu0
  mu1_new <- rnorm(1, ysum1/N_1, sqrt((sigma_1^2)/N_1))
  Q_old <- dnorm(mu1_old, ysum1/N_1, sqrt((sigma_1^2)/N_1), log = TRUE)
  Q_new <-dnorm(mu1_new, ysum1/N_1,sqrt((sigma_1^2)/N_1), log = TRUE)
  f_old <- log(ftheta(mu_0, mu1_old,sigma_0, sigma_1)*fygivenxphi(mu1_old, mu_1, sigma_1))
  f_new <-  log(ftheta(mu_0, mu1_new,sigma_0, sigma_1)*fygivenxphi(mu1_new, mu_1, sigma_1))

  # NB: Ratio on log scale
  ratio <- (f_new*Q_old)/(f_old*Q_new)
  acc <- min(0, ratio) # on log scale
  u <- log(runif(1))
  if (u < acc){
```

```
    res <- mu1_new
  } else {
    res <- mu1_new
  }
  return(res)
}

test1 <- muone(N_1, ysum1, sigma_1, mu1_old)
test1
```

```
## [1] -0.9382593
```

4.

c) We want to find the acceptance probability for σ_0. Since we know that the full conditional for σ_0^2 is inverse gamma distributed, we can use the transformation formula to determine the full conditional distribution for σ_0. We denote this as $Q(\sigma_0|x,y,\varphi^{-1})$. Then $w(\sigma_0) = \sigma_0^2$

$$Q(\sigma_0|x,y,\varphi^{-1}) = f(w(\sigma_0)|x,y,\varphi^{-1})\cdot|w'(\sigma_0)|Q(\sigma_0|x,y,\varphi^{-1}) = ...Q(\sigma_0|x,y,\varphi^{-1}) \propto \frac{1}{(\sigma_0)^{2a_0+N_0+1}}\cdot\exp\Big(-\frac{b_0+\sum_{i=1}^{N_0}(y_i$$

The acceptance probability is given by

$$\alpha = \min\Big(1, \frac{Q(\sigma_0|x,y,\varphi^{-1})}{Q(\tilde{\sigma}_0|x,y,\varphi^{-1})}\frac{f(\tilde{\sigma}_0|x,y,\varphi^{-1})}{f(\sigma_0|x,y,\varphi^{-1})}\Big)$$

After laboursome calculations we define

$$\Omega = \Big(\frac{\tilde{\sigma}_0}{\sigma_0}\Big)^{2a_0+1}\cdot\exp\Big\{b_0\cdot\Big(\frac{1}{\tilde{\sigma}_0^2}-\frac{1}{\sigma_0^2}\Big) - \frac{\mu_1-\mu_0}{\sqrt{\tilde{\sigma}_0\sigma_1}} + \frac{\mu_1-\mu_0}{\sqrt{\sigma_0\sigma_1}}\Big\}$$

and find that

$$\frac{Q(\sigma_0|x,y,\varphi^{-1})}{Q(\tilde{\sigma}_0|x,y,\varphi^{-1})}\frac{f(\tilde{\sigma}_0|x,y,\varphi^{-1})}{f(\sigma_0|x,y,\varphi^{-1})} = \Omega\cdot\begin{cases}\Big(\frac{\sigma_0}{\tilde{\sigma}_0}\Big)^3\cdot\exp\Big(-\sqrt{\frac{\sigma_1}{\tilde{\sigma}_0}}+\sqrt{\frac{\sigma_1}{\sigma_0}}\Big) & \text{for } \sigma_0<\sigma_1 \text{ and } \tilde{\sigma}_0<\sigma_1\\ \frac{\sigma_0^3}{\tilde{\sigma}_0^2\sigma_1}\cdot\exp\Big(-\sqrt{\frac{\tilde{\sigma}_0}{\sigma_1}}+\sqrt{\frac{\sigma_1}{\sigma_0}}\Big) & \text{for } \sigma_0<\sigma_1 \text{ and } \tilde{\sigma}_0>\sigma_1\\ \Big(\frac{\sigma_0}{\tilde{\sigma}_0}\Big)^2\cdot\exp\Big(-\sqrt{\frac{\tilde{\sigma}_0}{\sigma_1}}+\sqrt{\frac{\sigma_0}{\sigma_1}}\Big) & \text{for } \sigma_0>\sigma_1 \text{ and } \tilde{\sigma}_0>\sigma_1\\ \frac{\sigma_0^2\sigma_1}{\tilde{\sigma}_0^3}\cdot\exp\Big(-\sqrt{\frac{\sigma_1}{\tilde{\sigma}_0}}+\sqrt{\frac{\sigma_0}{\sigma_1}}\Big) & \text{for } \sigma_0>\sigma_1 \text{ and } \tilde{\sigma}_0<\sigma_1\end{cases}$$

We implement on log-scale

```
#Acceptance probability for sigma_0
a_0 = 2.5
b_0 = 6

sigma_0squared_prop = rinvgamma(1, shape = a_0 + N_0/2, scale = b_0 + sum((y[x==0]-mu_0)^2)/2)
sigma_0_prop = sqrt(sigma_0squared_prop)
logomega = (2*a_0 + 1)*(log(sigma_0_prop) - log(sigma_0)) + b_0 * (1/sigma_0squared_prop - 1/(sigma_0)^
if (sigma_0 <= sigma_1 && sigma_0_prop < sigma_1) {
  case = 3*(log(sigma_0)-log(sigma_0_prop)) - sqrt(sigma_1/sigma_0_prop) + sqrt(sigma_1/sigma_0)
} else if (sigma_0 <= sigma_1 && sigma_0_prop > sigma_1) {
  case = 3*log(sigma_0) - 2*log(sigma_0_prop) - log(sigma_1) - sqrt(sigma_0_prop/sigma_1) + sqrt(sigma_
} else if (sigma_0 > sigma_1 && sigma_0_prop > sigma_1) {
  case = 2*(log(sigma_0)-sigma_0_prop) -sqrt(sigma_0_prop/sigma_1) + sqrt(sigma_0/sigma_1)
} else if (sigma_0 > sigma_1 && sigma_0_prop < sigma_1) {
```

```
  case = 2*log(sigma_0) + log(sigma_1) - 3*log(sigma_0_prop) - sqrt(sigma_1/sigma_0_prop) + sqrt(sigma_0
}
qfratio = logomega*case
alpha = min(0,qfratio)
u = log(runif(1))
if (u < alpha) {
  sigma_0_prop = sigma_0
}
```

```
#Acceptance probability for sigma_1
a_1 = 2.5
b_1 = 6
sigma_1squared_prop = rinvgamma(1, shape = a_1 + N_1/2, scale = b_1 + sum((y[x==1]-mu_1)^2)/2)
sigma_1_prop = sqrt(sigma_1squared_prop)
logomega = (2*a_1 + 1)*(log(sigma_1_prop) - log(sigma_1)) + b_1 * (1/sigma_1squared_prop - 1/(sigma_0)^2
if (sigma_0 <= sigma_1 && sigma_0_prop < sigma_1) {
  case = 3*(log(sigma_0)-log(sigma_0_prop)) - sqrt(sigma_1/sigma_0_prop) + sqrt(sigma_1/sigma_0)
} else if (sigma_0 <= sigma_1 && sigma_0_prop > sigma_1) {
  case = 3*log(sigma_0) - 2*log(sigma_0_prop) - log(sigma_1) - sqrt(sigma_0_prop/sigma_1) + sqrt(sigma_1
} else if (sigma_0 > sigma_1 && sigma_0_prop > sigma_1) {
  case = 2*(log(sigma_0)-sigma_0_prop) -sqrt(sigma_0_prop/sigma_1) + sqrt(sigma_0/sigma_1)
} else if (sigma_0 > sigma_1 && sigma_0_prop < sigma_1) {
  case = 2*log(sigma_0) + log(sigma_1) - 3*log(sigma_0_prop) - sqrt(sigma_1/sigma_0_prop) + sqrt(sigma_0
}
qfratio = logomega*case
alpha = min(0,qfratio)
u = log(runif(1))
if (u < alpha) {
  sigma_0_prop = sigma_0
}
```