STAT 8700 Homework 7

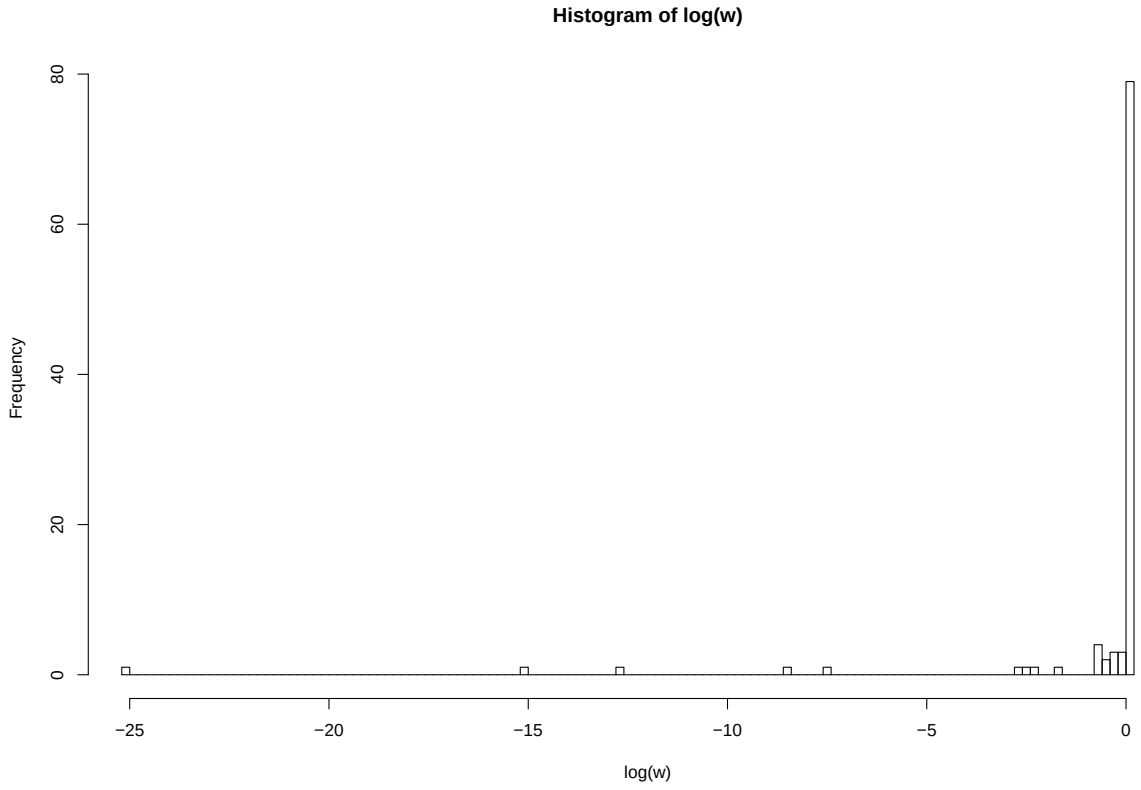
Brian Detweiler

Friday, October 14, 2016

1. Suppose our target distribution $p(\theta|y)$ is a standard Normal, and we choose $g(\theta)$ to be a t-distribution with 3 degrees of freedom.

(a) Draw a sample of size $S = 100$ from $g(\theta)$ and compute the importance ratios. Plot a histogram of the log importance ratios.

```
S <- 100
df <- 3
theta <- rt(S, df)

w <- dnorm(theta, 0, 1) / dt(theta, df = df)

hist(log(w), breaks = S, xlim = c(-25, 1))
```



(b) Estimate $E[\theta|y]$ and $Var(\theta|y)$ using importance sampling. Compare to the true values.

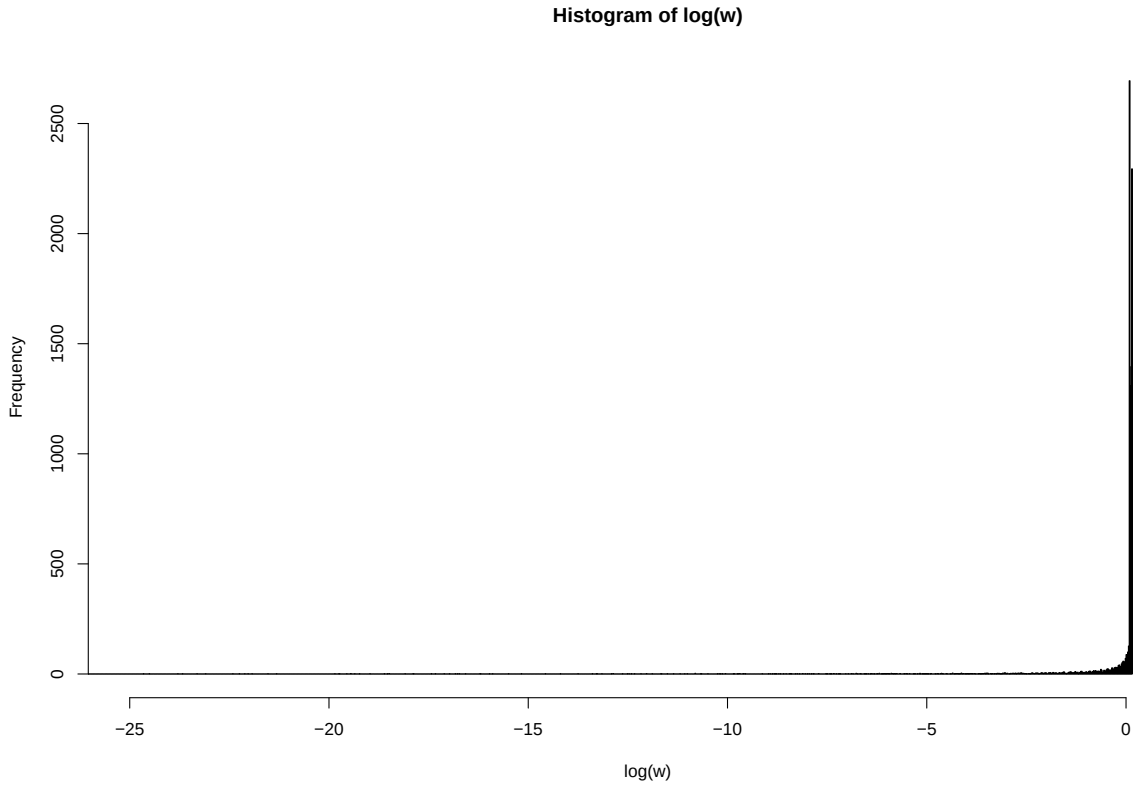
```
theta.expected <- sum(theta * w) / sum(w)
theta.variance <- sum(((theta - theta.expected)^2) * w) / sum(w)
```

$$E[\theta|y] = -0.05708$$
$$Var(\theta|y) = 0.9755562$$

■

(c) Repeat the first two parts of the question using $S = 10000$.

```
S <- 10000
df <- 3
theta <- rt(S, df)

w <- dnorm(theta, 0, 1) / dt(theta, df = df)

hist(log(w), breaks = S, xlim = c(-25, 1))
```




```
theta.expected <- sum(theta * w) / sum(w)
theta.variance <- sum(((theta - theta.expected)^2) * w) / sum(w)
```

$$E[\theta|y] = -0.0015704$$
$$Var(\theta|y) = 0.987901$$

■

(d) Using the sample obtained in the previous part, compute an estimate of the effective sample size.

```
w_tilde <- w / sum(w)
Seff <- 1 / sum(w_tilde^2)
```

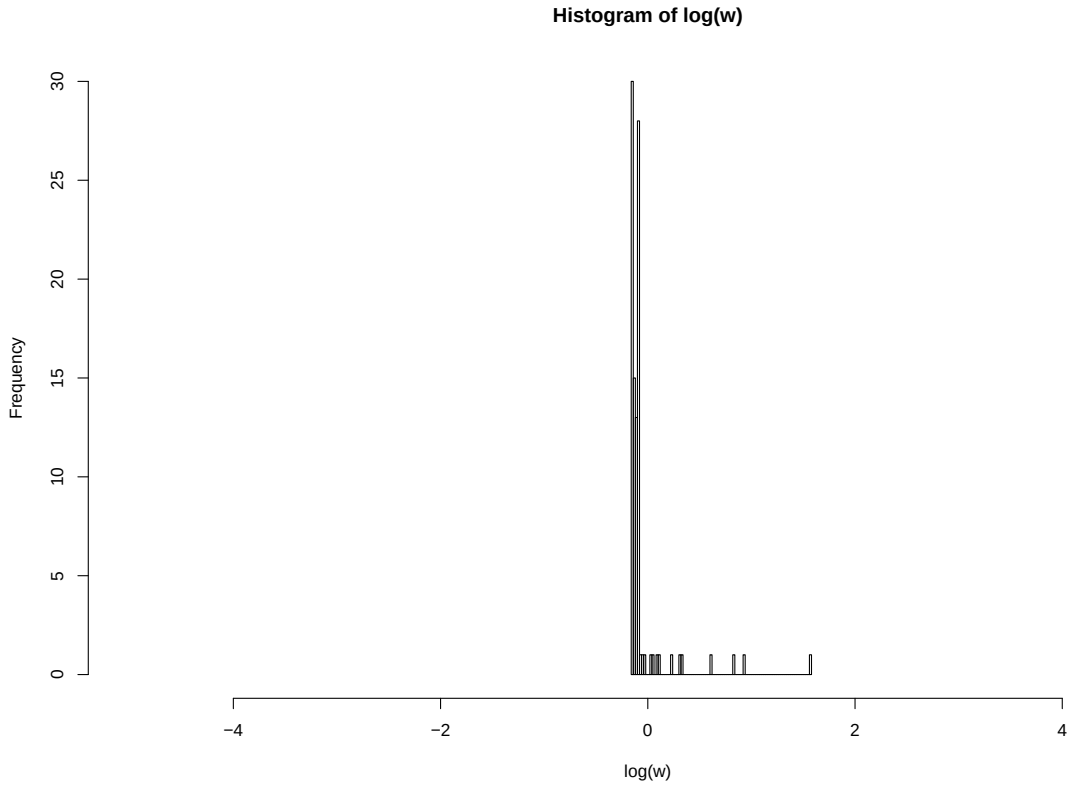
The effective sample size is 9188.0067449.

■

2. Repeat Question 1, swapping the distributions used for $p(\theta|y)$ and $g(\theta)$.

(a) Draw a sample of size $S = 100$ from $g(\theta)$ and compute the importance ratios. Plot a histogram of the log importance ratios.

```
S <- 100
df <- 3
theta <- rnorm(S, 0, 1)

w <- dt(theta, df = df) / dnorm(theta, 0, 1)

hist(log(w), breaks = S, xlim = c(-5, 5))
```



■

(b) Estimate $E[\theta|y]$ and $Var(\theta|y)$ using importance sampling. Compare to the true values.

```
theta.expected <- sum(theta * w) / sum(w)
theta.variance <- sum(((theta - theta.expected)^2) * w) / sum(w)
```

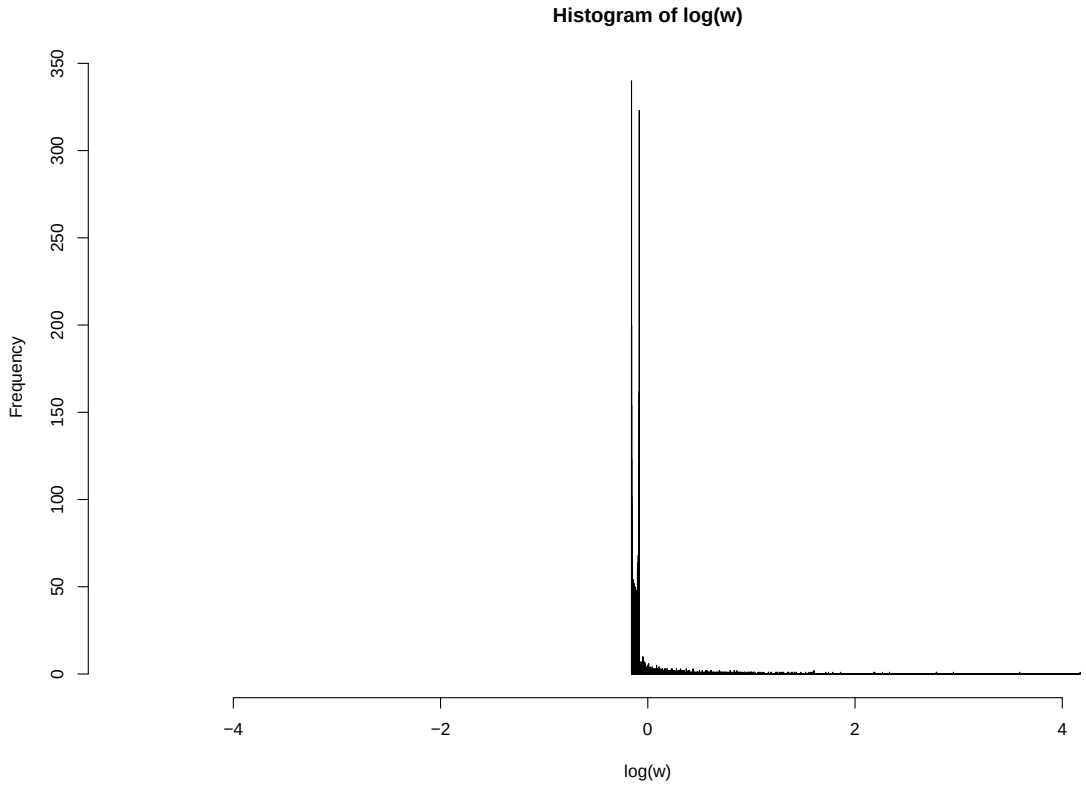
$$E[\theta|y] = 0.0633615$$
$$Var(\theta|y) = 1.7076221$$

The true mean and variance are $\mu = 0$ and $\sigma^2 = 1$. The estimates are off by 0.0633615 and -0.7076221, respectively, which are reasonably close.

■

(c) Repeat the first two parts of the question using $S = 10000$.

```
S <- 10000
df <- 3
theta <- rnorm(S, 0, 1)

w <- dt(theta, df = df) / dnorm(theta, 0, 1)

hist(log(w), breaks = S, xlim = c(-5, 5))
```




```
theta.expected <- sum(theta * w) / sum(w)
theta.variance <- sum(((theta - theta.expected)^2) * w) / sum(w)
```

$$E[\theta|y] = 0.0454561$$
$$Var(\theta|y) = 1.4882434$$

The true mean and variance are $\mu = 0$ and $\sigma^2 = 1$. The estimates are off by 0.0454561 and -0.4882434, respectively, which are reasonably close.

■

(d) Using the sample obtained in the previous part, compute an estimate of the effective sample size.

```
w_tilde <- w / sum(w)
Seff <- 1 / sum(w_tilde^2)
```

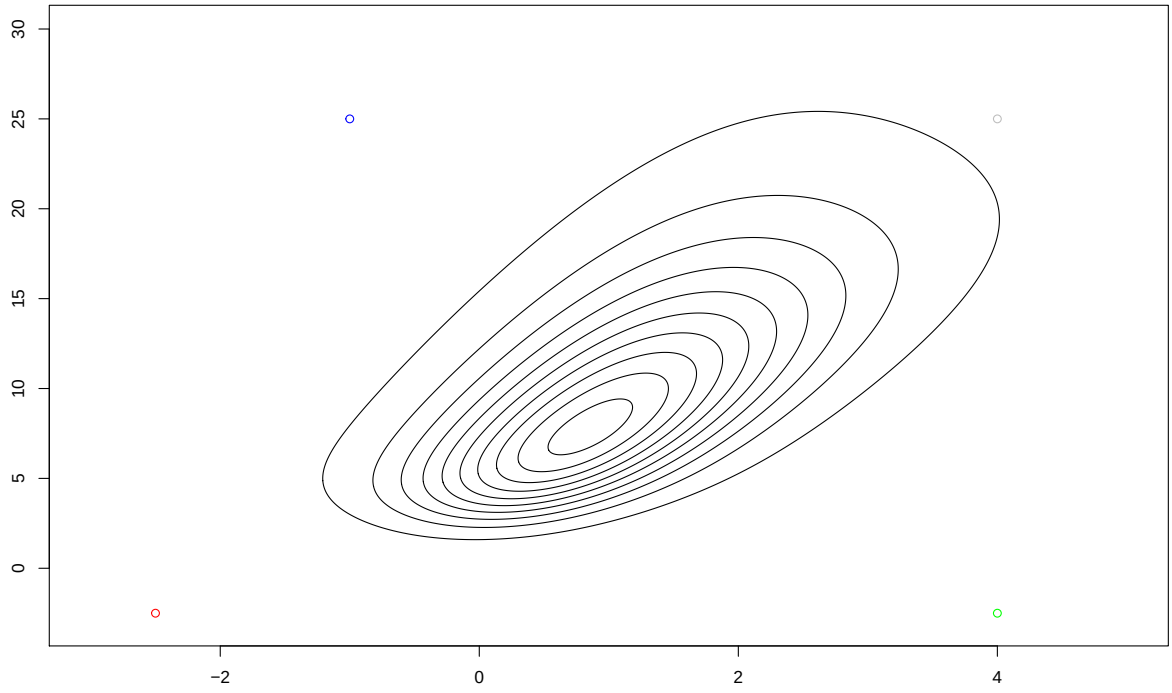
The effective sample size is 5648.8268147.

■

3. Use the Metropolis algorithm to obtain simulated values of *alpha* and *beta* from the Bioassay example in Section 3.7. Be sure to define your starting points and your jumping rule. Compute with log-densities (see page 261). Run your simulations long enough for approximate convergence. Include a plot of the last 25% of your simulated values overlaying the contour plot.

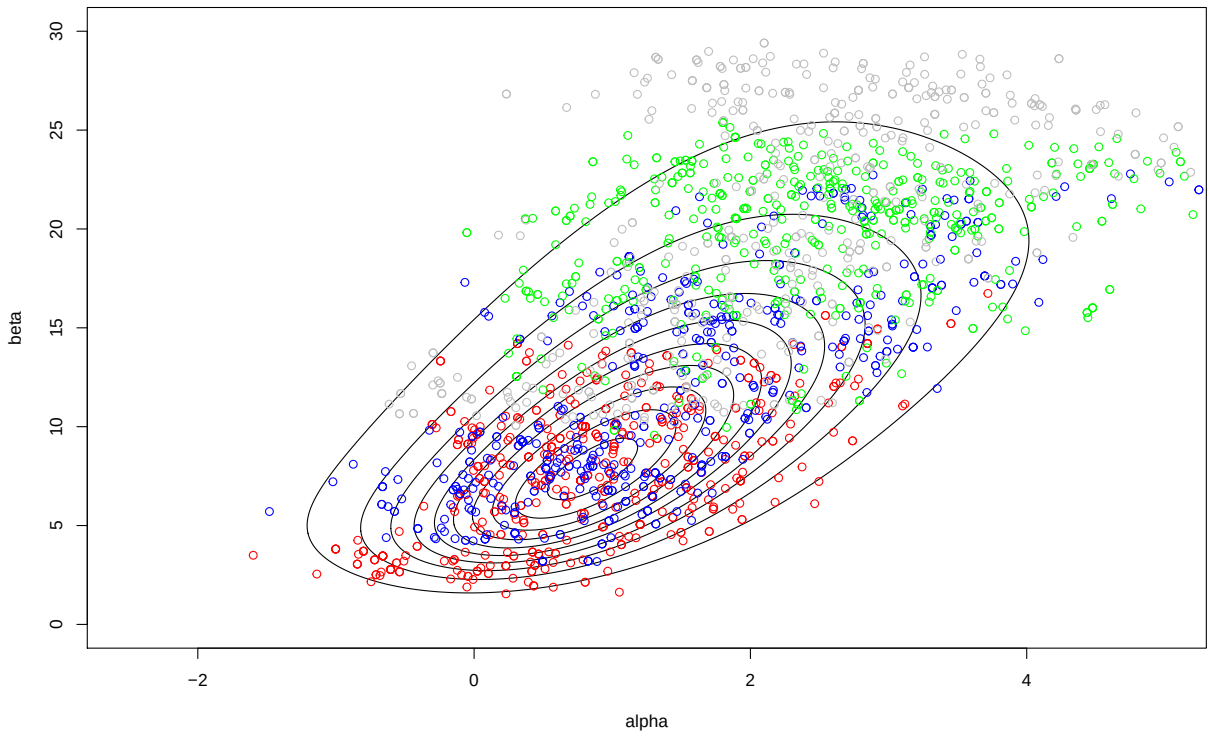
```
bioassay <- data.frame(cbind(x = c(-0.86, -0.30, -0.05, 0.73),
                             n = c(5, 5, 5, 5),
                             y = c(0, 1, 3, 5),
                             y.mod = c(0.5, 1, 3, 4.5)))


logit <- function(x) {
  log(x / (1 - x))
}

inv_logit <- function(x) {
  (exp(x)) / (1 + exp(x))
}

posterior <- function(a, b) {
    temp <- 1
    x <- bioassay$x
    y <- bioassay$y
    n <- bioassay$n
    for (i in 1: length(x)) {
      temp <- temp * (inv_logit(a + (b * x[i]))^y[i]) * ((1 - inv_logit(a + (b * x[i])))^(n[i] - y[i]))
  }
    temp
}


posterior_contour <- function(alpha_min,
                              alpha_max,
                              grid_size_alpha,
                              beta_min,
                              beta_max,
                              grid_size_beta,
                              drawlabels = TRUE) {

  # Generate a list of alpha values
  alpha <- seq(alpha_min, alpha_max, length = grid_size_alpha)

  # Generate a list of beta values
  beta <- seq(beta_min, beta_max, length = grid_size_beta)

  # Evaluate the posterior density and all possible combinations of alpha and beta values
  post.dens <- outer(alpha, beta, "posterior")
```

```
  # Rescale the posterior so values are now relative to height of posterior mode
  scaled.dens <- post.dens / max(post.dens)

  # Draw contour plot
  contour(alpha,
          beta,
          scaled.dens,
          levels = c(0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95),
          xlab = "alpha",
          ylab = "beta",
          drawlabels = drawlabels)
}


# Metropolis Algorithm
require(mvtnorm, quietly=TRUE)
g <- function(a, b, m, s) {
  dmvnorm(x = cbind(a, b), mean = m, sigma = s)
}

x <- seq(-3, 5, length = 1001)
y <- seq(-3, 30, length = 1001)
z <- outer(x, y, "posterior")
z <- z / max(z)
contours <- seq(0.05, 0.95, 0.1)
contour(x, y, z, levels = contours, drawlabels = FALSE)

iterations <- 2000

theta1 <- matrix(NA, iterations + 1, 2)
theta2 <- matrix(NA, iterations + 1, 2)
theta3 <- matrix(NA, iterations + 1, 2)
theta4 <- matrix(NA, iterations + 1, 2)

theta1[1,] <- c(-2.5, -2.5)
theta2[1,] <- c(-1, 25)
theta3[1,] <- c(4, -2.5)
theta4[1,] <- c(4, 25)

# 1. Choose a starting value for theta*
mu <- c(0, 0)
Sigma <- matrix(c(1, 0, 0, 1), 2, 2)
points(theta1, col = "red")
points(theta2, col = "blue")
points(theta3, col = "green")
points(theta4, col = "gray")
```




```
proposal <- posterior

for (i in 1:iterations) {
  thetastar1 <- NULL
  thetastar2 <- NULL
  thetastar3 <- NULL
  thetastar4 <- NULL
  #require(mvtnorm)

  # Sample a proposal value theta*
  thetastar1 <- rmvnorm(n = 1, theta1[i,], 0.6 * Sigma)
  thetastar2 <- rmvnorm(n = 1, theta2[i,], 0.6 * Sigma)
  thetastar3 <- rmvnorm(n = 1, theta3[i,], 0.6 * Sigma)
  thetastar4 <- rmvnorm(n = 1, theta4[i,], 0.6 * Sigma)

  tau1 <- ((proposal(thetastar1[1,1], thetastar1[1,2])) / (proposal(theta1[i,1], theta1[i,2])))
  tau2 <- ((proposal(thetastar2[1,1], thetastar2[1,2])) / (proposal(theta2[i,1], theta2[i,2])))
  tau3 <- ((proposal(thetastar3[1,1], thetastar3[1,2])) / (proposal(theta3[i,1], theta3[i,2])))
  tau4 <- ((proposal(thetastar4[1,1], thetastar4[1,2])) / (proposal(theta4[i,1], theta4[i,2])))

  temp1 <- runif(1, 0, 1)
  temp2 <- runif(1, 0, 1)
  temp3 <- runif(1, 0, 1)
  temp4 <- runif(1, 0, 1)

  if (temp1 < tau1) {
    theta1[i + 1,] <- thetastar1
```

```
  } else {
    theta1[i + 1,] <- theta1[i,]
  }

  if (temp2 < tau2){
    theta2[i + 1,] <- thetastar2
  } else {
    theta2[i + 1,] <- theta2[i,]
  }

  if (temp3 < tau3) {
    theta3[i + 1,] <- thetastar3
  } else {
    theta3[i + 1,] <- theta3[i,]
  }

  if (temp4 < tau4) {
    theta4[i + 1,] <- thetastar4
  } else {
    theta4[i + 1,] <- theta4[i,]
  }

  #lines(theta1, col = "red")
  #lines(theta2, col = "blue")
  #lines(theta3, col = "green")
  #lines(theta4, col = "gray")
}

conv1 <- matrix(NA, iterations / 4, 2)
conv2 <- matrix(NA, iterations / 4, 2)
conv3 <- matrix(NA, iterations / 4, 2)
conv4 <- matrix(NA, iterations / 4, 2)

for (j in 1:(iterations / 4)) {
  conv1[j, ] <- theta1[(3 * iterations / 4) + 1 + j, ]
  conv2[j, ] <- theta2[(3 * iterations / 4) + 1 + j, ]
  conv3[j, ] <- theta3[(3 * iterations / 4) + 1 + j, ]
  conv4[j, ] <- theta4[(3 * iterations / 4) + 1 + j, ]
}

# apply(conv1, 2, "mean")
# apply(conv2, 2, "mean")
# apply(conv3, 2, "mean")
# apply(conv4, 2, "mean")

#plot(conv1, xlim=c(-2.5, 4), ylim=c(-2.5, 25),col="red")

posterior_contour(-2.5, 5, 1001, 0, 30, 1001, FALSE)

points(conv1, col="red")
points(conv2, col="blue")
points(conv3, col="green")
points(conv4, col="gray")
```


beta
alpha
0
5
10
15
20
25
30
−2
0
2
4

■