Homework 7

Mei Jiao

10/23/2020

5.3.1 Rejection sampling

Let f and g be two probability densities on $(0, \infty)$, such that

$$f(x) \propto \sqrt{4+x}\, x^{\theta-1} e^{-x}, \quad g(x) \propto (2x^{\theta-1} + x^{\theta-1/2}) e^{-x}, \quad x > 0.$$

1. Find the value of the normalizing constant for g, i.e., the constant C such that

$$C \int_0^\infty (2x^{\theta-1} + x^{\theta-1/2}) e^{-x} dx = 1.$$

Show that g is a mixture of Gamma distributions. Identify the component distributions and their weights in the mixture.

To calculate the normalizing constant for g, we have

$$\begin{aligned}
&\int_0^\infty (2x^{\theta-1} + x^{\theta-1/2}) e^{-x} dx \\
&= \int_0^\infty 2x^{\theta-1} e^{-x} dx + \int_0^\infty x^{\theta-1/2} e^{-x} dx \\
&= 2\Gamma(\theta) + \Gamma(\theta + \frac{1}{2})
\end{aligned}$$

Thus, we can get,

$$C = \frac{1}{2\Gamma(\theta) + \Gamma(\theta + \frac{1}{2})}$$

Then, the full expression of g will be,

$$g(x) = \frac{1}{2\Gamma(\theta) + \Gamma(\theta + \frac{1}{2})} (2x^{\theta-1} + x^{\theta-1/2}) e^{-x}$$

which can also be expressed as,

$$g(x) = \frac{2\Gamma(\theta)}{2\Gamma(\theta) + \Gamma(\theta + \frac{1}{2})} \frac{x^{\theta-1} e^{-x}}{\Gamma(\theta)} + \frac{\Gamma(\theta + \frac{1}{2})}{2\Gamma(\theta) + \Gamma(\theta + \frac{1}{2})} \frac{x^{(\theta+\frac{1}{2})-1} e^{-x}}{\Gamma(\theta + \frac{1}{2})}$$

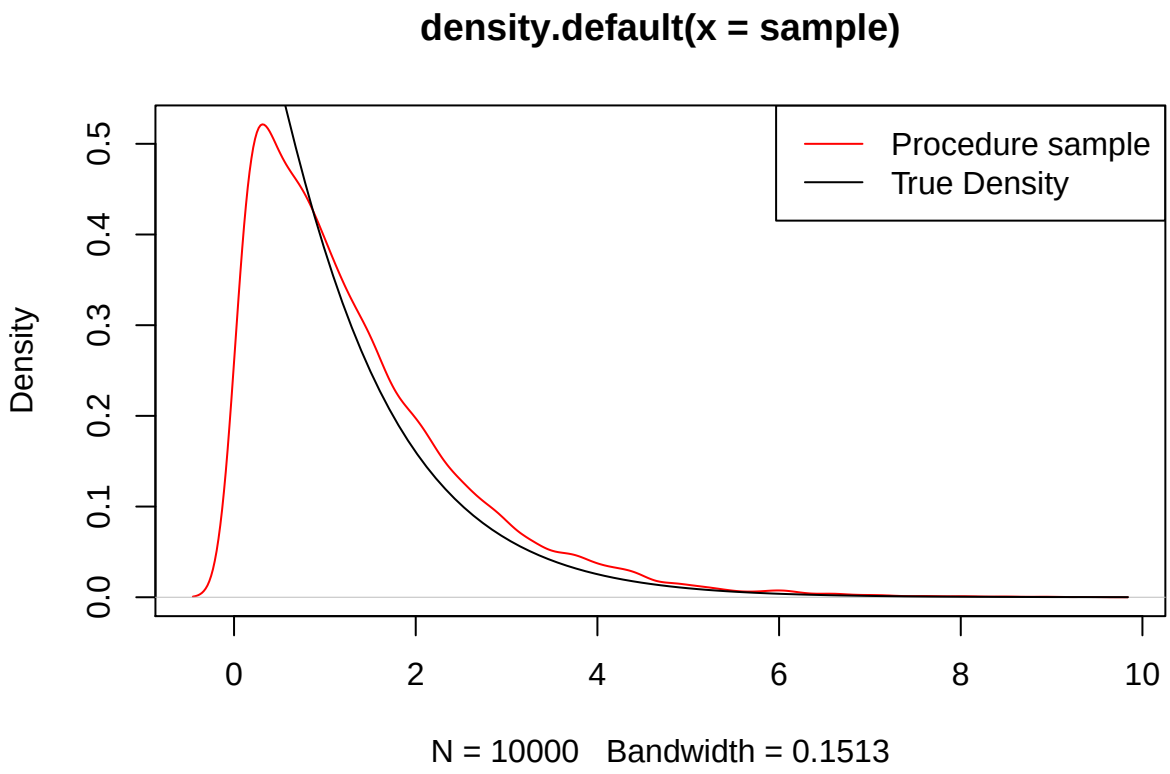
Thus we can have that g is a mixture of Gamma distributions. The component distribution is $\Gamma(\theta, 1)$ with weight $w1$, and $\Gamma(\theta + \frac{1}{2}, 1)$ with weight $w2$, which can be expressed as:

$$w1 = \frac{2\Gamma(\theta)}{2\Gamma(\theta) + \Gamma(\theta + \frac{1}{2})}, w2 = \frac{\Gamma(\theta + \frac{1}{2})}{2\Gamma(\theta) + \Gamma(\theta + \frac{1}{2})}$$

2. Design a procedure (pseudo-code) to sample from g; implement it in an R function; draw a sample of size $n = 10,000$ using your function for at least one θ value; plot the kernel density estimation of g from your sample and the true density in one figure.

Assume size as 10000 and theta as 1.

```
gprocedure <- function(size, theta){
  w1 = 2 * gamma(theta) / (2 * gamma(theta) + gamma(theta + (1/2)))
  sample <- double(size)
  u = runif(size,0,1)

  for (i in 1:size){
    if (u[i] > w1){
      sample[i] <- rgamma(1, shape = theta, rate = 1)
    }
    else{
      sample[i] <- rgamma(1, shape = theta + (1/2), rate = 1)
    }
  }
  return(sample)
}

g <- function(theta, x) {
  g= 1/(2*gamma(theta) + gamma(theta + (1/2)))*(2*x^(theta-1) +  x^(theta-1/2))*exp(-x)
}

theta <-1
size <-10000
sample <- gprocedure(size, theta)
plot(density(sample), col = "red")
curve(g(theta = 1,x), add = TRUE, col = "black")
legend("topright", legend = c("Procedure sample", "True Density"),
       col = c("red","black"), lty = c(1,1))
```



3. Design a procedure (pseudo-code) to use rejection sampling to sample from f using g as the instrumental distribution. Overlay the estimated kernel density of a random sample generated by your procedure and f.

To use g estimate the distribution of f , first let

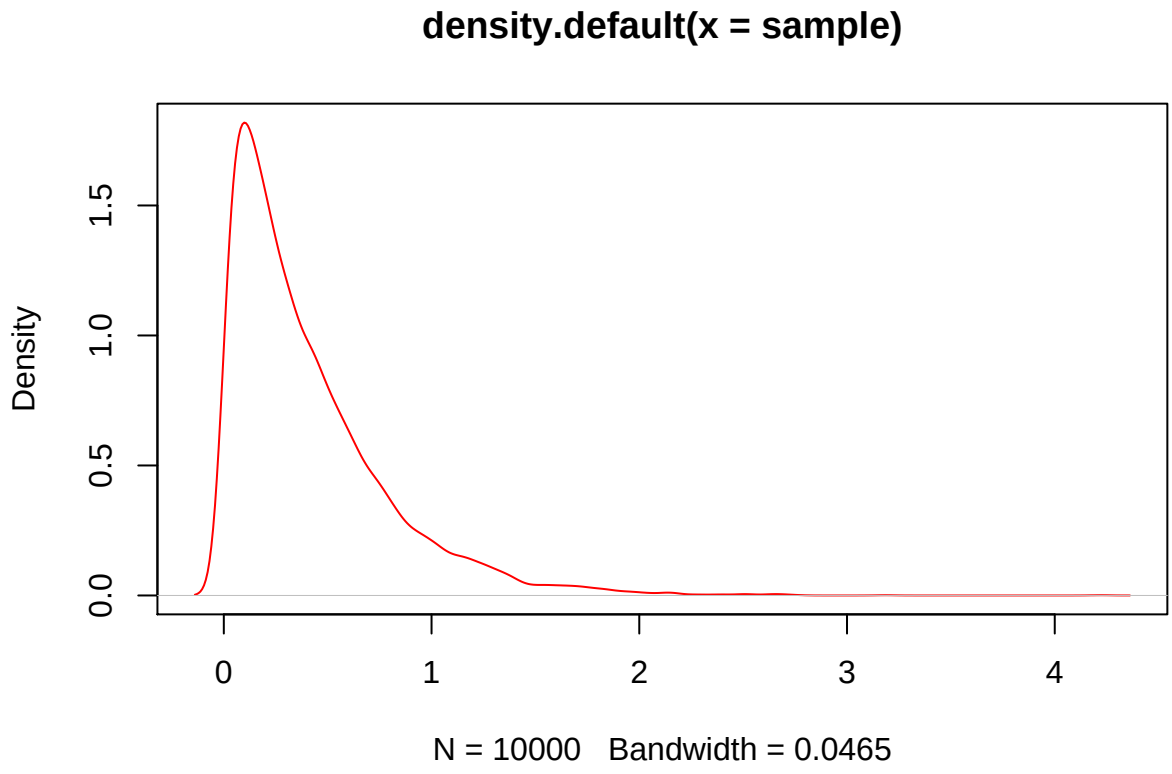
$f(x) \propto q(x) = \sqrt{4+x}x^{\theta-1}e^{-x}$, and

$$\begin{aligned}\alpha &= \sup_{x>0}\frac{q(x)}{g(x)} \\ &= \sup_{x>0}\frac{\sqrt{4+x}x^{\theta-1}e^{-x}}{C(2x^{\theta-1}+x^{\theta-\frac{1}{2}})e^{-x}} \\ &= \sup_{x>0}\frac{\sqrt{4+x}}{C\sqrt{x}} \\ &= \frac{1}{C}\end{aligned}$$

which means $q(x) = \sqrt{4+x}x^{\theta-1}e^{-x} \leqslant \alpha g(x) = \frac{1}{C}g(x) = (2x^{\theta-1}+x^{\theta-\frac{1}{2}})e^{-x}$

```
fprocedure <- function(size, theta) {
  count <- 0
  sample <- double(0)
  while(count < size) {
    gfunc <- gprocedure(1, theta)
    u <- runif(1,0,1)
```

```
    q <- sqrt(4 + gfunc)*((gfunc)^(theta-1))*exp(-1*gfunc)
    r <- q/(2*gfunc^(theta-1) +  gfunc^(theta-1/2))*exp(-gfunc)
    if (u <= r) {
      sample <- append(sample, gfunc)
      count <- count + 1
    }
  }
  return(sample)
}

theta <- 1
size <-10000
sample <- fprocedure(size, theta)
plot(density(sample), col = "red")
```



6.3.1 Consider again the normal mixture example, except that the parameters of the normal distributions are considered unknown. Suppose that prior for μ_1 and μ_2 are $N(0, 10^2)$, that the prior for $1/\sigma_1^2$ and $1/\sigma_2^2$ are $\Gamma(a, b)$ with shape $a = .5$ and scale $ b = 10$. Further, all the priors are independent. Design an MCMC using the Gibbs sampling approach to estimate all 5 parameters. Use the arms() function in package HI. Run your chain for sufficiently long and drop the burn-in period. Plot the histogram of the results for all the parameters.

First, generate some data with mixing rate δ and $1-\delta$. However, as the parameter of the normal distribution are unknown, we need to generate some random sample with $\delta = 0.7, \mu_1 = 7, \sigma_1^2 = 0.5^2, \mu_2 = 10, \sigma_2^2 = 0.5^2$

Thus, the function of likelihood will be:

$$likelihood(x; \delta, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2) = \prod_{i=1}^{n} \left[\delta N(\mu_1, \sigma_1^2) + (1-\delta)N(\mu_2, \sigma_2^2)\right]$$

Also, we can have the log likelihood function as:

$$loglike(x; \delta, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2) = \sum_{i=1}^{n} log(\delta N(\mu_1, \sigma_1^2) + (1-\delta)N(\mu_2, \sigma_2^2))$$

Next, we need to find the posterior density by using the Bayes theory:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

where $p(\theta|x)$ is the posterior density we need, $p(x|\theta)$ is the likelihood function, $p(\theta)$ is prior probability and $p(x)$ is the normalization constant useful for Bayesian model.

As the prior for μ_1 and μ_2 and $1/\sigma_1^2$ and $1/\sigma_2^2$ given in the question, we can get the posterior density we need is:

$$p(\delta, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2|x) = \prod_{i=1}^{n} \left[\delta N(\mu_1, \sigma_1^2) + (1-\delta)N(\mu_2, \sigma_2^2)\right] N(\mu_1, 0, 10)N(\mu_2, 0, 10)IG(\sigma_1^2, 0.5, 10)IG(\sigma_2^2, 0.5, 10)$$

where the prior for σ_1^2 and σ_2^2 are the inverse gamma distribution $IG(0.5, 10)$.

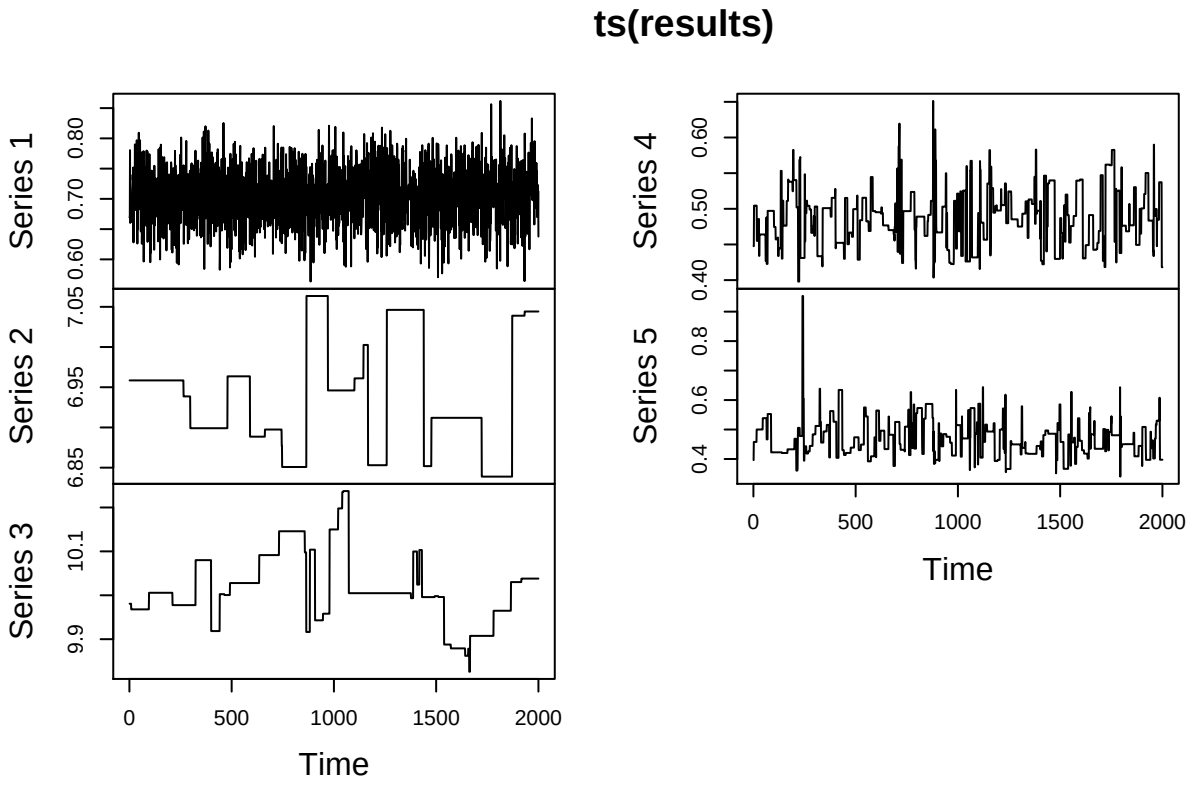
Thus, the log posterior function will be:

$$logposterior(\delta, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2|x) = \sum_{i=1}^{n}(\delta N(\mu_1, \sigma_1^2)+(1-\delta)N(\mu_2, \sigma_2^2))+log(N(\mu_1, 0, 10))+log(N(\mu_2, 0, 10))+log(IG(\sigma_1^2, 0.5, 10)$$

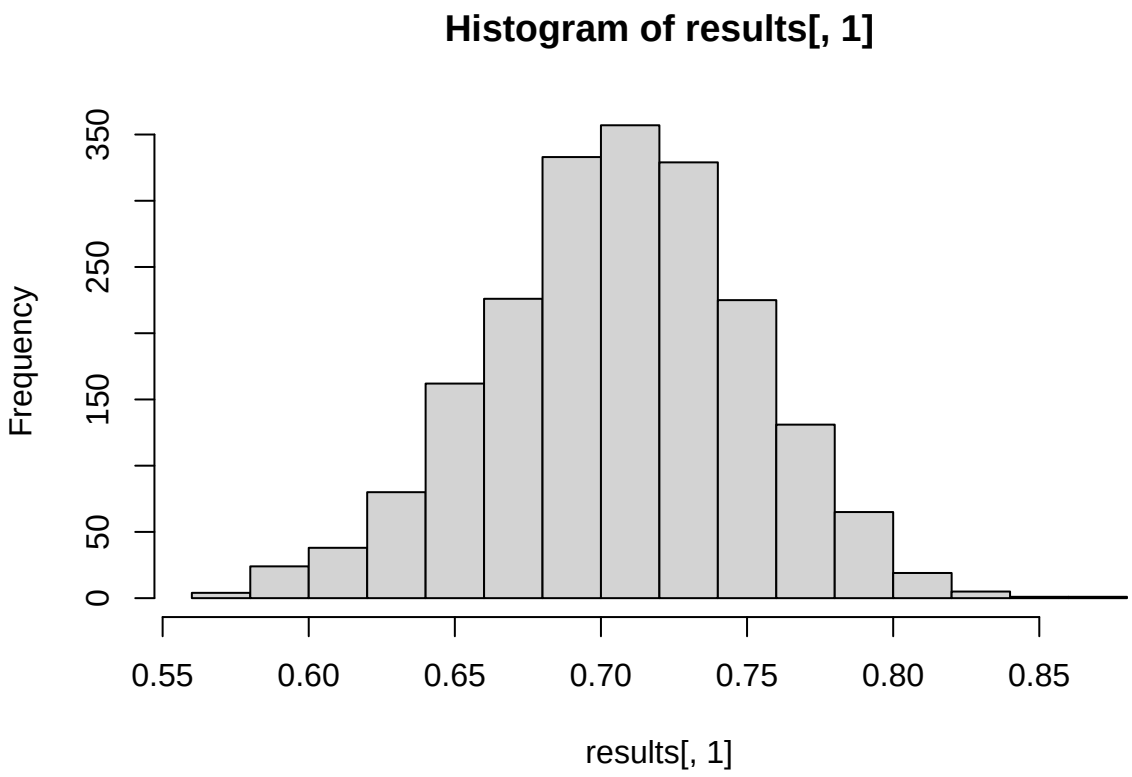
Last, we need to compose the MCMC function based on the Gibbs sampling approach by using the ARMS algorithm with R package HI.By the result, we can see δ is about 0.7, μ_1 is about 7,μ_2 is about 10 and σ_1^2,σ_2^2 is about 0.5^2

```
delta <- 0.7 # true value to be estimated based on the data
n <- 100
set.seed(123)
u <- rbinom(n, prob = delta, size = 1)
```

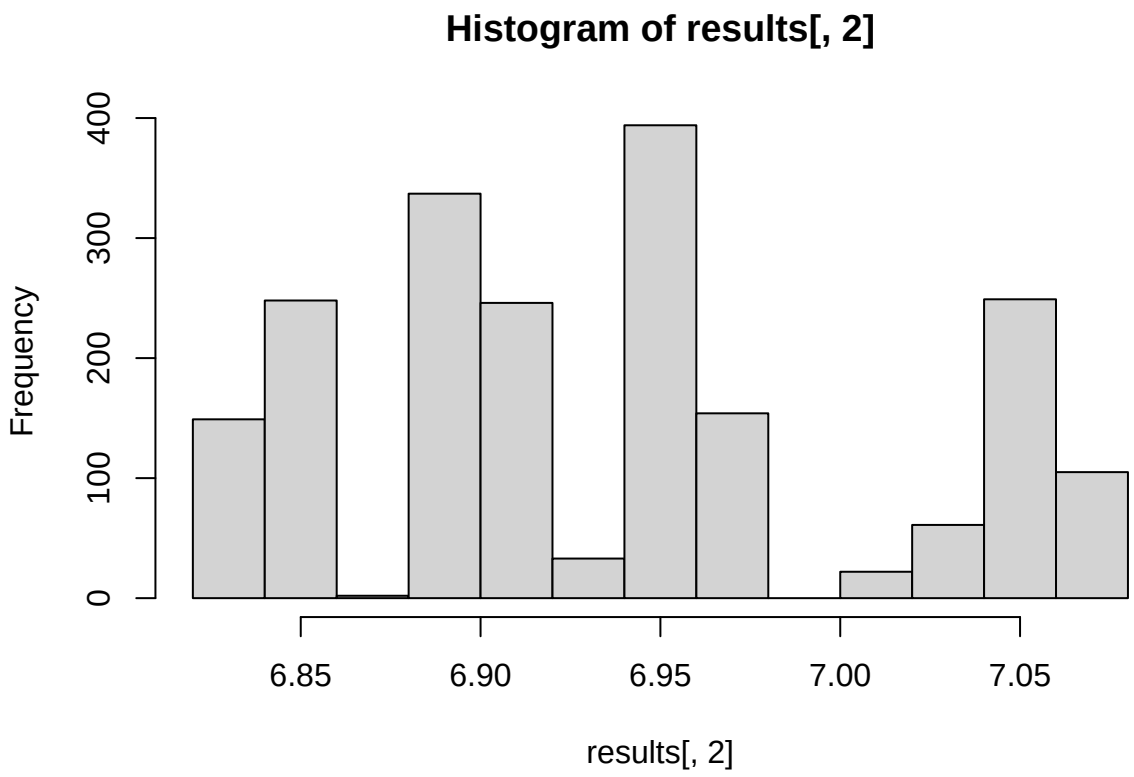
```r
x0 <- rnorm(n, ifelse(u == 1, 7, 10), 0.5)

#Compose the likelihood function
mylike <- function(delta, x, mu1, mu2, sigma1, sigma2) {
    delta * dnorm(x, mu1, sigma1) + (1 - delta) * dnorm(x, mu2, sigma2)
}

#Compose the log likelihood function
loglike <- function(delta, x , mu1, mu2, sigma1, sigma2){
  sum(log(delta * dnorm(x, mu1, sigma1) + (1 - delta) * dnorm(x, mu2, sigma2)))
}

#Compose the log posterior density function
library(invgamma)
logposterior <- function(delta, x, mu1, mu2, sigma1, sigma2){
  mu1.logprior <- dnorm(mu1, 0, 10, log = T)
  mu2.logprior <- dnorm(mu2, 0, 10, log = T)
  sigma1.logprior <- dinvgamma(sigma1^2, shape = 0.5, scale = 10, log = T)
  sigma2.logprior <- dinvgamma(sigma2^2, shape = 0.5, scale = 10, log = T)
  sum(mu1.logprior + mu2.logprior +sigma1.logprior + sigma2.logprior) + loglike(delta, x, mu1, mu2, sigm
}

# Compost MCMC function
library(HI)
mymcmc <- function(niter, delta.init, mu1.init, mu2.init, sigma1.init, sigma2.init,x){
  data <- matrix(nrow = niter, ncol = 5)

  #Gibbs Sampling
  for(i in 1:niter){
    f1 <- function(x1)  logposterior(x1,x0,mu1.init,mu2.init,sigma1.init,sigma2.init)
    delta.init <- data[i,1] <- arms(delta.init, f1, function(x1) (x1 > 0) * (x1< 1), 1)

    f2 <- function(x2)  logposterior(delta.init,x0,x2,mu2.init,sigma1.init,sigma2.init)
    mu1.init <- data[i,2] <- arms(mu1.init, f2, function(x2) (x2 > -100) * (x2 < 100), 1)

    f3 <- function(x3)  logposterior(delta.init,x0,mu1.init,x3,sigma1.init,sigma2.init)
    mu2.init <- data[i,3] <- arms(mu2.init, f3, function(x3) (x3 > -100) * (x3 < 100), 1)

    f4 <- function(x4)  logposterior(delta.init,x0,mu1.init,mu2.init,x4,sigma2.init)
    sigma1.init <- data[i,4] <- arms(sigma1.init, f4, function(x4) (x4 > 0) * (x4 < 200), 1)

    f5 <- function(x5)  logposterior(delta.init,x0,mu1.init,mu2.init,sigma1.init,x5)
    sigma2.init <- data[i,5] <- arms(sigma2.init, f5, function(x5) (x5 > 0) * (x5 < 200), 1)
  }
  data
}

niter <- 2500
results <- mymcmc(niter, 0.5, 1,1,1,1,x0)[-c(1:500),]
plot(ts(results))
```


ts(results)
Series 1
Series 2
Series 3
Series 4
Series 5
Time
0
500
1000
1500
2000

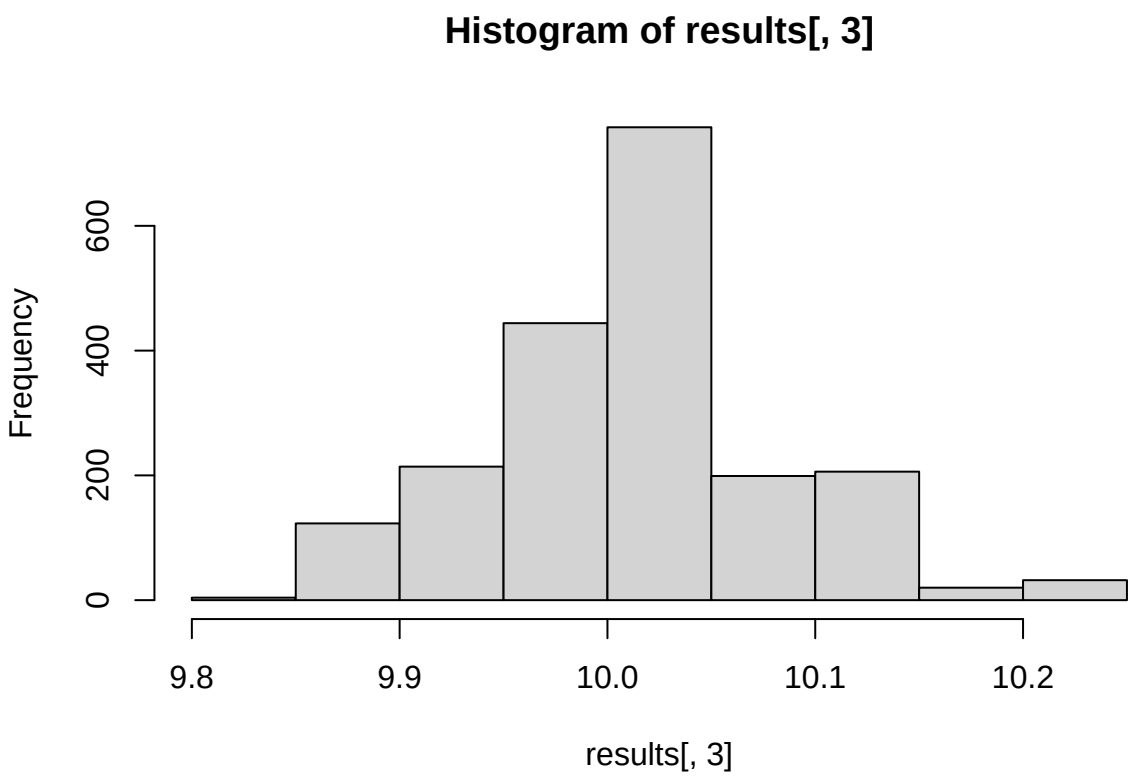
```r
# delta is about 0.7
hist(results[,1])
```



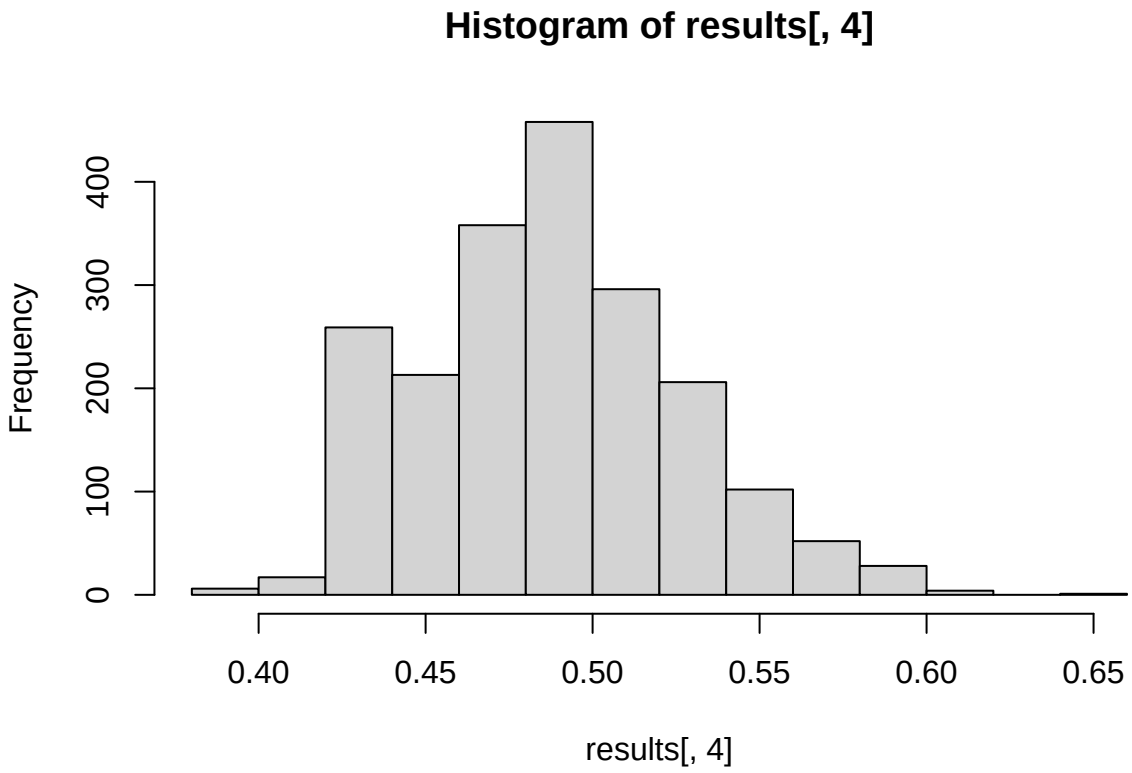
```r
# mu1 is about 7
hist(results[,2])
```



```r
# mu2 is about 10
hist(results[,3])
```



```r
#sigma1 is about 0.5^2
hist(results[,4])
```



```r
#sigma2 is about 0.5^2
hist(results[,5])
```