STAT 5361 - HW 7

Patrick Toman*

24 October 2020

Problem 5.3.1

Part (i)

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

Therefore, we can re-arrange terms to find that

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

Furthermore, let us denote

$$
\begin{aligned}
h_1(x) &= x^{\theta-1}e^{-x},\ I(x>0) \\
h_2(x) &= x^{\theta-1/2}e^{-x},\ I(x>0)
\end{aligned}
$$

Clearly, these two functions correspond the kernel of two gamma densities, namely $Gamma(\theta,1)$ and $Gamma(\theta+1/2,1)$. Therefore, we derive our full pdf for $g(x)$ as

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

Thus, we conclude that $g(x)$ is a mixture of gammas with the following mixing proportions

$$
\begin{aligned}
\pi_1 &= \frac{2\Gamma(\theta)}{2\Gamma(\theta)+\Gamma(\theta+1/2)} \\
\pi_2 &= \frac{\Gamma(\theta+1/2)}{2\Gamma(\theta)+\Gamma(\theta+1/2)} \\
&\text{Subject to } \pi_1+\pi_2 = 1
\end{aligned}
$$

*patrick.toman@uconn.edu; Ph.D. student at Department of Statistics, University of Connecticut.

Part (ii)

The following pseudo-code details how to draw from our gamma mixture in part(i).

Algorithm 1: Gamma Mixture Simulation

Input: n = sample size,θ = user defined scale parameter
Result: n Simulated Observations from density $g(x)$

Allocate empty vector $X \in \mathcal{R}^{n\times 1}$
for i *in* $1 : n$ **do**
    Draw $U \sim Bernoulli\left(p = \frac{2\Gamma(\theta)}{2\Gamma(\theta)+\Gamma(\theta+1/2)}\right)$
    if $U = 1$ **then**
        $X[i] \sim Gamma(\theta, 1)$
    else
        $X[i] \sim Gamma(\theta + \frac{1}{2}, 1)$
    end
end

Implementation

```
my_gamma_mix_sim <- function(size,theta){

  X <- rep(NA,size)

  p <- 2*gamma(theta)/(2*gamma(theta) + gamma(theta+0.5))

  for(i in 1:size){

    U <- rbinom(1,1,p)

    if(U == 1){

      X[i] <- rgamma(1,shape = theta,rate=1)

    }else{

      X[i] <- rgamma(1,shape = theta+1/2,rate=1)

    }

  }

  return(X)


}
```

Results

The snippet of code below simulates a $n = 10,000$ draws from the gamma mixture densith $g(x|\theta)$ where $\theta = 3.25$

```r
set.seed(1022)
theta <- 3.25
my_simulated_gammamix <- my_gamma_mix_sim(size = 10000,theta=theta)
```

Part (iii) - Rejection Sampling

The folling algorithm presents a rejection sampling scheme to draw samples from target density

$$f(x) \propto \sqrt{4+x}x^{\theta-1}e^{-x}, I(x>0)$$

using the following instrumental density

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

Algorithm 2: Rejection Sampler

Input: $n =$ sample size,$\theta =$ user defined scale parameter,$\alpha \geq 0 =$ user defined constant

Result: simulated sample n of size bservations from target density $f(x)$

1 Allocate empty vector $X \in \mathcal{R}^{n\times 1}$
2 **while** $\dim(X) < n$) **do**
3   Draw $Y^{'} \sim g(x)$ Draw $U \sim Uniform(0,1)$
4   **if** $U < \frac{f(Y^{'})}{\alpha(Y^{'})}$ **then**
5         $X[i] = Y^{'}$
6   **else**
7   Reject $Y^{'}$ and repeat lines (3) and (4)
8   **end**
9 **end**

Implementation and Results

We implement the rejection sampler

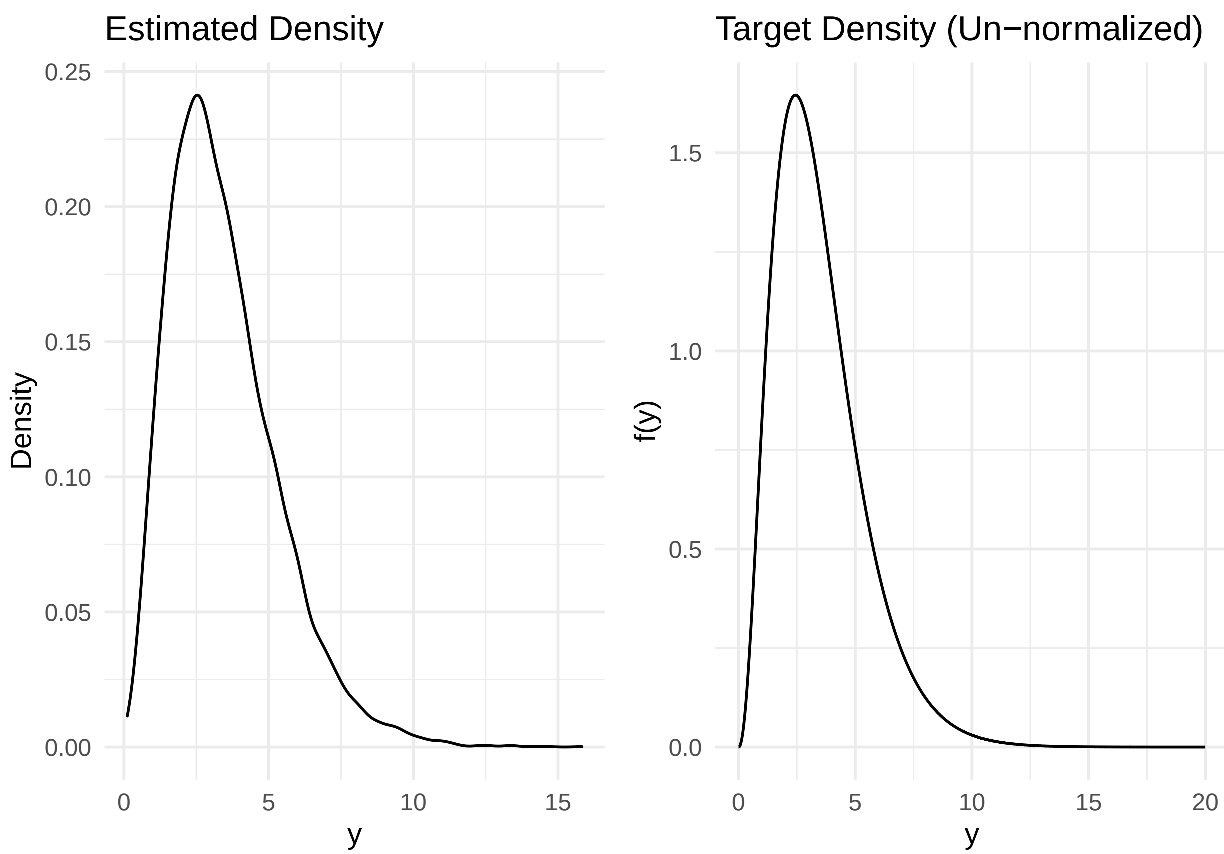
```r
gy <- function(y,theta){(2*y^(theta-1) + y^(theta-1/2))*exp(-y)}

fy <- function(y,theta){sqrt(4+y)*y^(theta-1)*exp(-y)}

rejection_sampler <- function(n,theta,alpha){

  X <- rep(NA,n)

  p <- 2*gamma(theta)/(2*gamma(theta) + gamma(theta+0.5))

  accepted_ct <- 0

  rejected_ct <- 0

  i <- 1

  while(accepted_ct < n){

    y <- my_gamma_mix_sim(size=1,theta=theta)
```

```
    gy <- gy(y=y,theta=theta)

    fy_set <- fy(y=y,theta=theta)

    U <- runif(1,0,1)

    if(U < fy_set/(alpha*gy)){

      X[i] <- y

      i <- i + 1

      accepted_ct <- accepted_ct + 1

    }else{

      rejected_ct <- rejected_ct + 1

    }

  }

  return(list('Sample'=X,'RejectCt'=rejected_ct))
}
```

Results

We run our rejection sampler with $\alpha = 3, \theta = 3.25$ and draw $n = 10,000$ samples from the target density $f(x)$. The plots below are a side-by-side comparison of the estimated kernel density and the un-normalized target density. Clearly, the estimated density and true target density are quite similar.

```
n <- 10000
rejection <- rejection_sampler(n=n,theta = theta,alpha=3)
fy_density <- fy(y=seq(0,20,length.out = 10000),theta=theta)
```



Problem 6.3.1

Setup

Let us denote $\theta = (\lambda, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2)$. Suppose then that we have $X_1, \ldots, X_n \overset{iid}{\sim} f(x|\theta)$ where $f(x|\theta)$ is the gaussian mixture model

$$f(x|\theta) = \lambda\phi(.|\mu_1, \sigma_1^2) + (1-\lambda)\phi(.|\mu_2, \sigma_2^2)$$

Furthermore, let us denote the priors for the components of θ are as follows

- $\pi(\lambda) \sim U(0,1)$
  - If we let p_1 and $(1-\lambda) = p_2$ then we have $\pi(\mathbf{p}) \sim Dirichlet(\gamma_1, \gamma_2)$
- $\pi(\mu_1), \pi(\mu_2) \sim N(0, 10^2)$
- $\pi(\sigma_1^2), \pi(\sigma_2^2) \sim \; InvGamma(a = 0.5, b = 10)$ - (Inverse Gamma with parameters $a = 0.5, b = 10$)

Conditionals

Following the notation of cite here we first define the following

$$n_j = S_j^{(x)} = \sum_{i=1}^{n} I(z_j^{(t)} = j)x_l$$

$$S_j^v = \sum_{i=1}^{n} I(z_i^{(t)} = j)(x_i - \mu_j)^2 \; , j = 1, 2$$

Then it can be shown that we have the following posterior densities

- $(\mu_j|\sigma_j^2, \mathbf{x}, \mathbf{z}) \sim N\left(\frac{\lambda_j\tau_j + S_j^x}{\tau_j + n_j}, \frac{\sigma_j^2}{\tau_j + n_j}\right)$
  - $\lambda_j = 0, j = 1, 2$ - since our prior mean $\mu_j = 0$ for $j = 1, 2$
  - $\tau_j = 10^{-2}, j = 1, 2$ which is the precision for our priors for μ_1, μ_2
- $(\sigma_j^2|\mu_j, \mathbf{x}, \mathbf{z}) \sim \ InvGamma\left(\alpha_j + \frac{n_j+1}{2}, \beta_j + \frac{\tau_j}{2}(\mu_j - \lambda_j)^2 + \frac{S_j^v}{2}\right)$
  - $\alpha_j = a = 0.5$ and $\beta_j = b = 10$ - the prior shape and scale parameters

Gibbs Sampler

In the same spirit as cite here we have

Algorithm 3: Gibbs Sampler

Initialize paramters $\theta^0, \mathbf{p}^0$
for t = *1,....* **do**
    Draw latent assignment variable $z_i^{(t)}$ for $j = 1, 2$ where $P(Z_i^{(t)} = j) \propto \frac{p_j^{(t-1)}}{\sigma_j^{2(t-1)}} exp\left\{-\frac{\left(x_i - \mu_j^{(t-1)}\right)^2}{2\sigma_j^{2(t-1)}}\right\}$
    Calculate $n_j^{(t)}$ for $j = 1, 2$
    Draw $\mathbf{p}^{(t)} \sim Dirichlet(\gamma_1 + n_1, \gamma_2 + n_2)$ - γ_1, γ_2 are known hyperparameters
    Draw $\mu_j^{(t)} \sim N\left(\frac{\lambda_j\tau_j + S_j^{x(t)}}{\tau_j + n_j}, \frac{\sigma_j^{2(t-1)}}{\tau_j + n_j}\right)$
end

Calculate $S_j^{v(t)} = \sum_{i=1}^n I(z_i^{(t)} = j)(x_i - \mu_j)^2$ for $j = 1, 2$
Draw $\sigma_j^{2(t)} \sim \ InvGamma\left(a + \frac{n_j+1}{2}, b + \frac{\tau_j}{2}(S_j^{v(t)})\right)$ for $j = 1, 2$

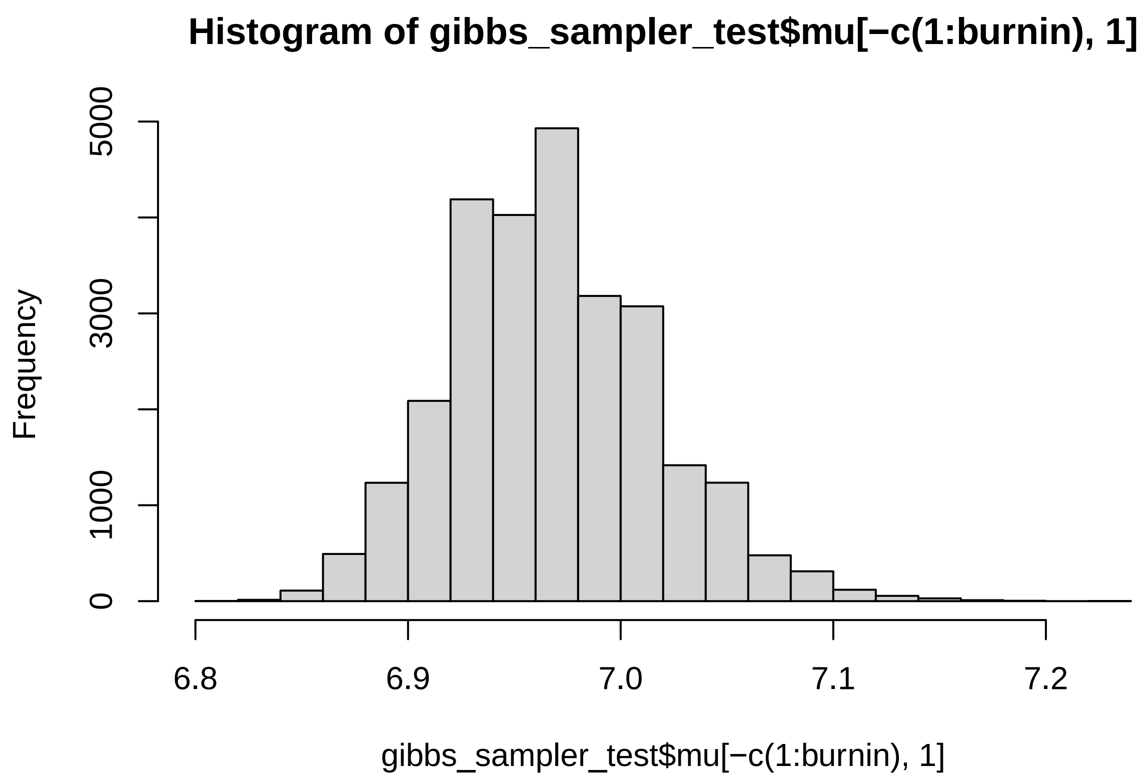
```
normalize <- function(x){return(x/sum(x))}

sample_z <- function(x,pi,mu,sigma_sq){

    dmat <- outer(mu,x,"-") # k by n matrix, d_kj =(mu_k - x_j)

    dmat[,1] <- sqrt(sigma_sq[1])

    dmat[,2] <- sqrt(sigma_sq[2])

    p.z.given.x <- as.vector(pi) * dnorm(dmat,0,1)

    p.z.given.x <- apply(p.z.given.x,2,normalize) # normalize columns

    z <- rep(0, length(x))

    for(i in 1:length(z)){

       z[i] <- sample(1:length(pi), size=1,prob=p.z.given.x[,i],replace=TRUE)

    }

  return(z)
}
```

```
sample_pi <- function(z,k){

    counts <- colSums(outer(z,1:k,FUN="=="))

    pi <- gtools::rdirichlet(1,counts+1)

    return(pi)
}

sample_mu <- function(x, z, k, prior,sigma_sq){

  df <- data.frame(x=x,z=z)

  mu <- rep(0,k)

  for(i in 1:k){

      sample.size <- sum(z==i)

      sample.mean <- ifelse(sample.size==0,0,mean(x[z==i]))

      post.prec <- sample.size+prior$prec

      post.mean <- (prior$mean * prior$prec + sample.mean * sample.size)/post.prec

      mu[i] <- rnorm(1,post.mean,sqrt(sigma_sq[i]/post.prec))

    }

    return(mu)

}

sample_sigmasq <- function(mu,x,z,k,prior){

  smat <- outer(mu,x,"-") # k by n matrix, d_kj =(mu_k - x_j)

  smat <- smat^2 # Comput Sj^v at iteration t

  df <- data.frame(smat=smat,z=z)

  sigma_sq <- rep(NA,k)

  for(i in 1:k){

    sample_size <- sum(z==i)

    sj <- ifelse(sample_size==0,0,sum((smat[z==i]-mu[i])^2))

    shape_param <- prior$shape + (sample_size+1)/2

    scale_param <- prior$scale + (0.5)*prior$prec*(mu[i]-prior$mean)^2 + 0.5*sj
```

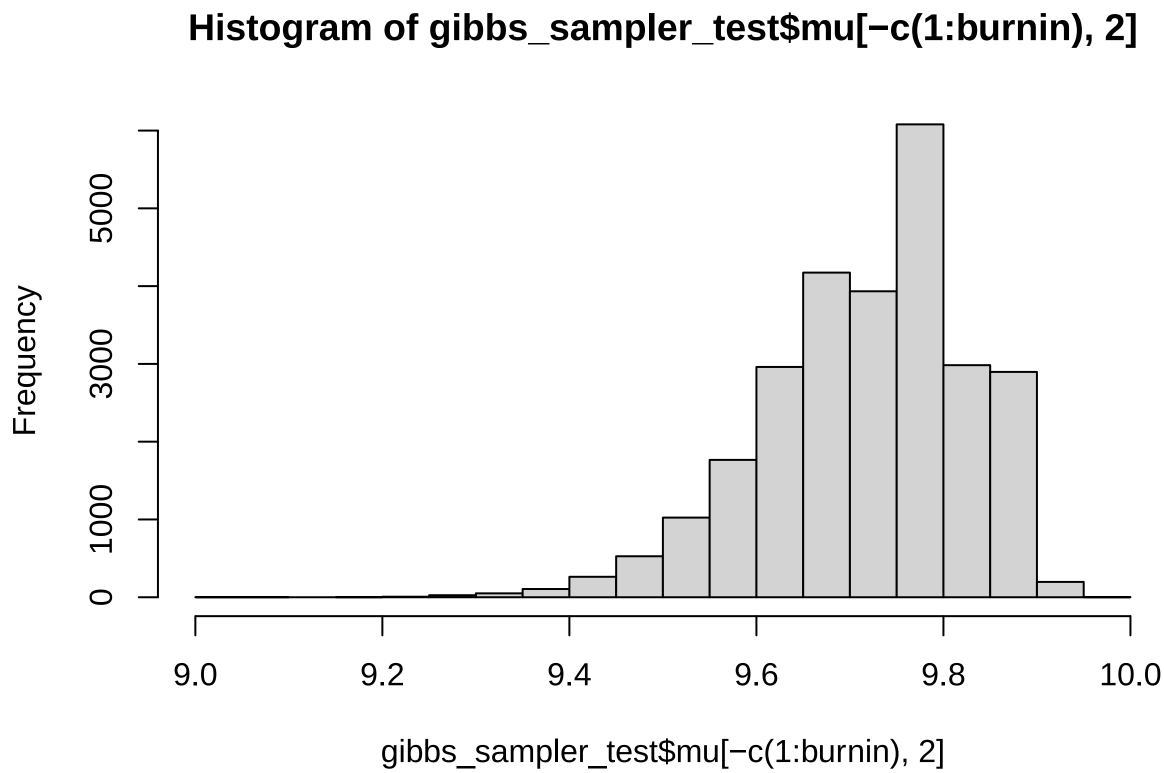
```
    sigma_sq[i] <- 1/(rgamma(1,shape=shape_param,scale=scale_param))

  }

  return(sigma_sq)

}


gibbs <- function(x,k,niter =1000,prior_list = list(mean=0,prec=0.5,shape=0.5,scale=10)){

    pi <- rep(1/k,k) # initialize

    mu <- rnorm(k,0,10)

    sigma_sq <- 1/rgamma(k,shape=0.5,scale=10)

    z <- sample_z(x,pi,mu,sigma_sq)

    res <- list(mu=matrix(nrow=niter, ncol=k), pi = matrix(nrow=niter,ncol=k),
                z = matrix(nrow=niter, ncol=length(x)),sigma_sq = matrix(nrow=niter, ncol=length(x)))

    res$mu[1,] <- mu

    res$pi[1,] <- pi

    res$z[1,] <- z

    res$sigma_sq[1,] <- sigma_sq

    for(i in 2:niter){

      pi <- sample_pi(z,k)

      mu <- sample_mu(x,z,k,prior=prior_list,sigma_sq = sigma_sq)

      sigma_sq <- sample_sigmasq(mu,x,z,k,prior=prior_list)

      z <- sample_z(x,pi,mu,sigma_sq = sigma_sq)

      res$mu[i,] <- mu

      res$pi[i,] <- pi

      res$sigma_sq[i,] <- sigma_sq

      res$z[i,] <- z

    }
    return(res)
  }
```

```
set.seed(1024)
```

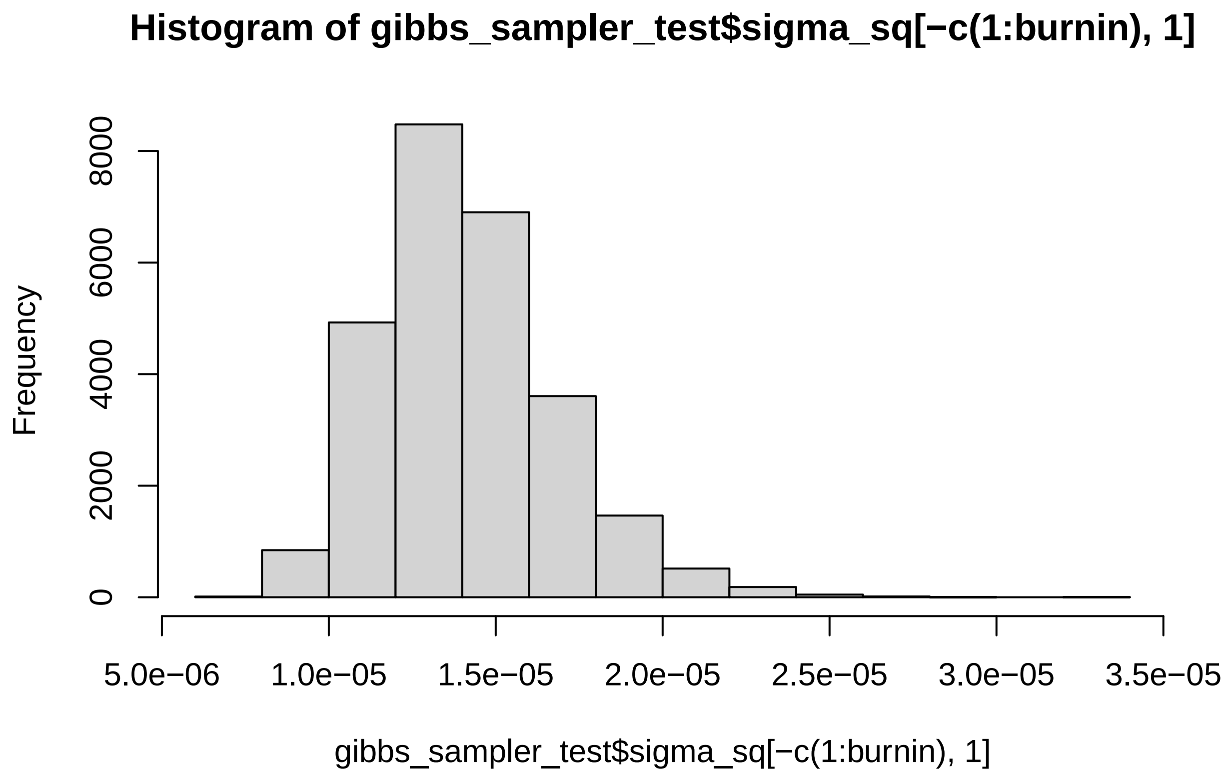
```r
niter_set <- 30000

k_set <- 2

gibbs_sampler_test <- gibbs(x=x,k=k_set,niter=niter_set)

burnin <- ceiling(niter_set/10)


hist(gibbs_sampler_test$mu[-c(1:burnin),1])
```




```r
hist(gibbs_sampler_test$mu[-c(1:burnin),2])
```




```
hist(gibbs_sampler_test$sigma_sq[-c(1:burnin),1])
```


Histogram of gibbs_sampler_test$sigma_sq[−c(1:burnin), 1]
Frequency
0
2000
4000
6000
8000
5.0e−06
1.0e−05
1.5e−05
2.0e−05
2.5e−05
3.0e−05
3.5e−05
gibbs_sampler_test$sigma_sq[−c(1:burnin), 1]