HW7

John Rothen

10/24/2020

5.3.1

A

$g(x) = (2x^{\theta-1} + x^{t-.5})e^{-x}$. Note here that a gamma distribution $\Gamma(\alpha, \beta)$ can be written as $\beta^\alpha x^{\alpha-1}e^{-\beta x}/\Gamma(\alpha)$. Thus, we can write g(x) as two gamma function: The first is $2x^{\theta-1}e^{-x}$, which is $Gamma(\theta, 1)$ where the weight is found using $2 = w\beta^\alpha/\Gamma(\alpha)$. Here the weight is w. Thus the weight for this function is $2\Gamma(\theta)$.

For the second part $x^{2\theta-1}e^{-x}$, we know this is $\Gamma(2\theta, 1)$. $1 = w\beta^\alpha/\Gamma(\alpha) = w/\Gamma(2\theta)$. Thus, the weight for this mixture is $\Gamma(2\theta)$.

```
tempfuncG <-function(x,t=2.5){
  return((2*(x^(t-1)) + (x^(t-.5)))*exp(-x))
}
c<-integrate(tempfuncG,lower= 0,upper= Inf)[1]
C <- 1/c$value
C
```

```
## [1] 0.214653
```

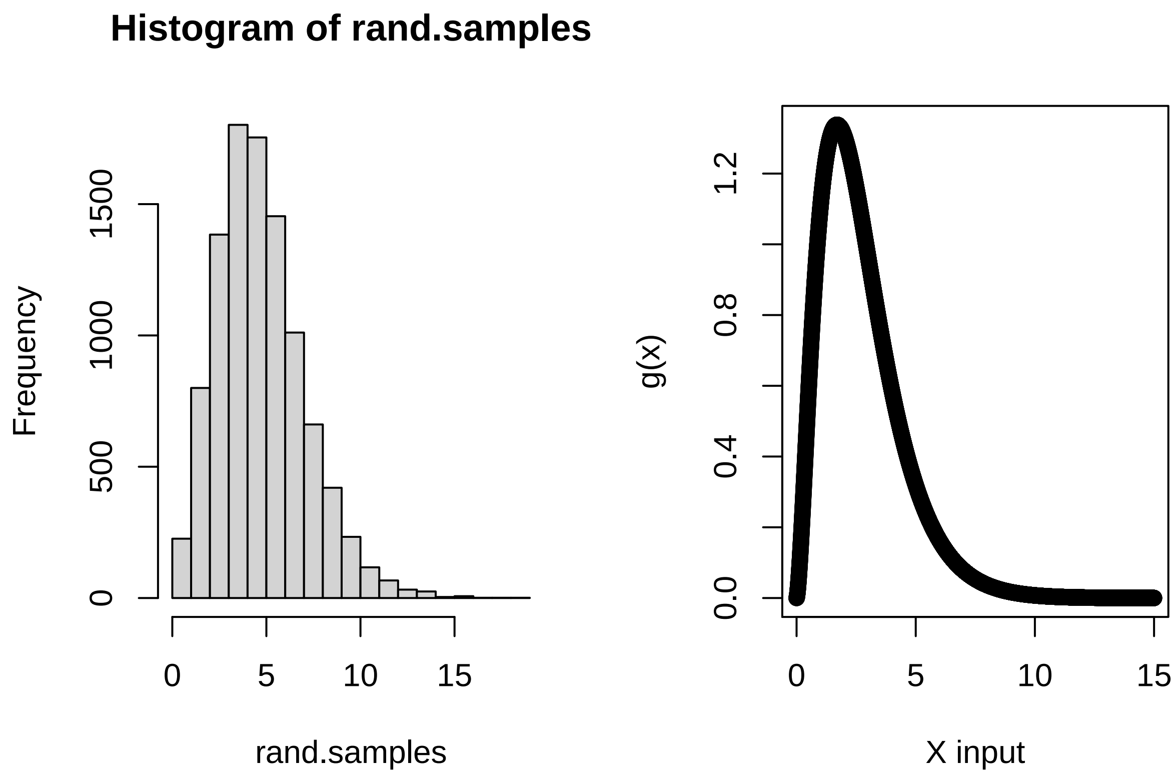
C = .214653

B

```
tempfuncF <- function(x,t=2.5){
  return(sqrt(4+x)*(x^(t-1))*(exp(-x)))
}
n=10000
theta=2.5
w1 = 2*gamma(theta)
w2 = gamma(2*theta)
U = runif(n)
rand.samples = rep(NA,n)
for(i in 1:n){
  if(U[i]<w1/w2){
    rand.samples[i] = rgamma(1,shape=theta,rate=1)
  }
  else{
    rand.samples[i] = rgamma(1,shape=2*theta,rate=1)
  }
}
par(mfrow=c(1,2))

hist(rand.samples)
```

```r
x= seq(0,15, length.out = n)
plot(x,tempfuncG(x), xlab= "X input", ylab= "g(x)")
```

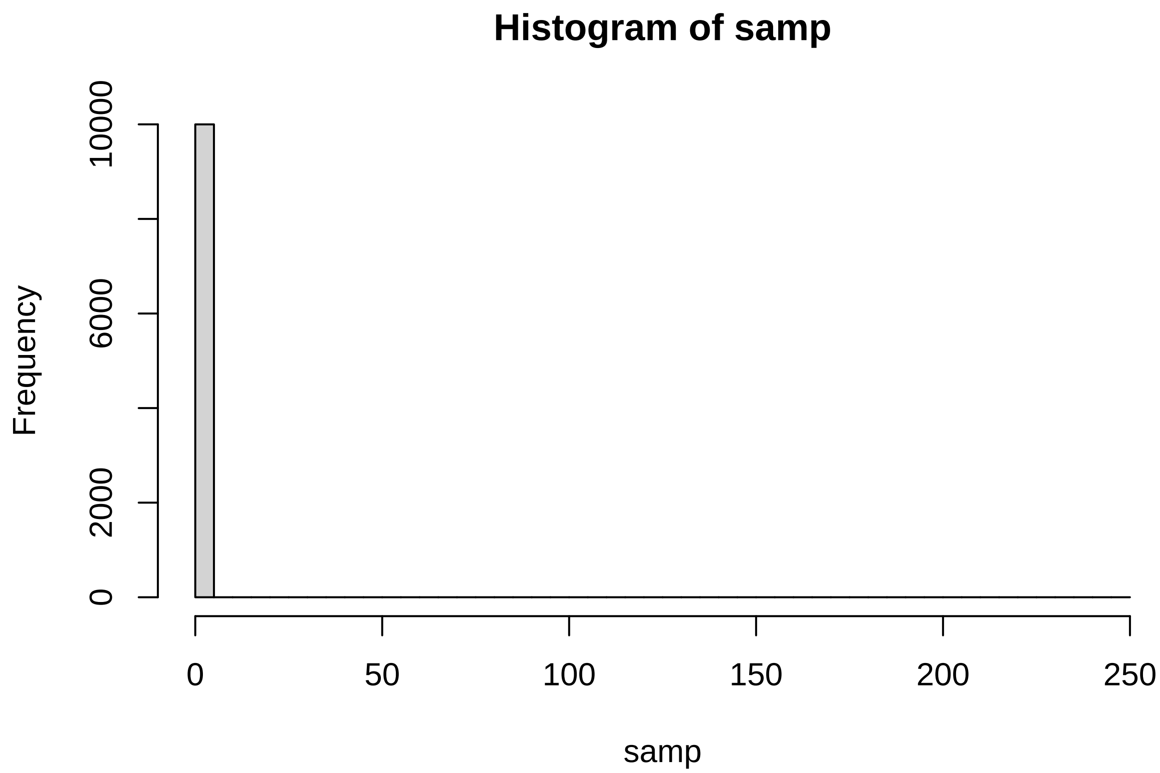


C

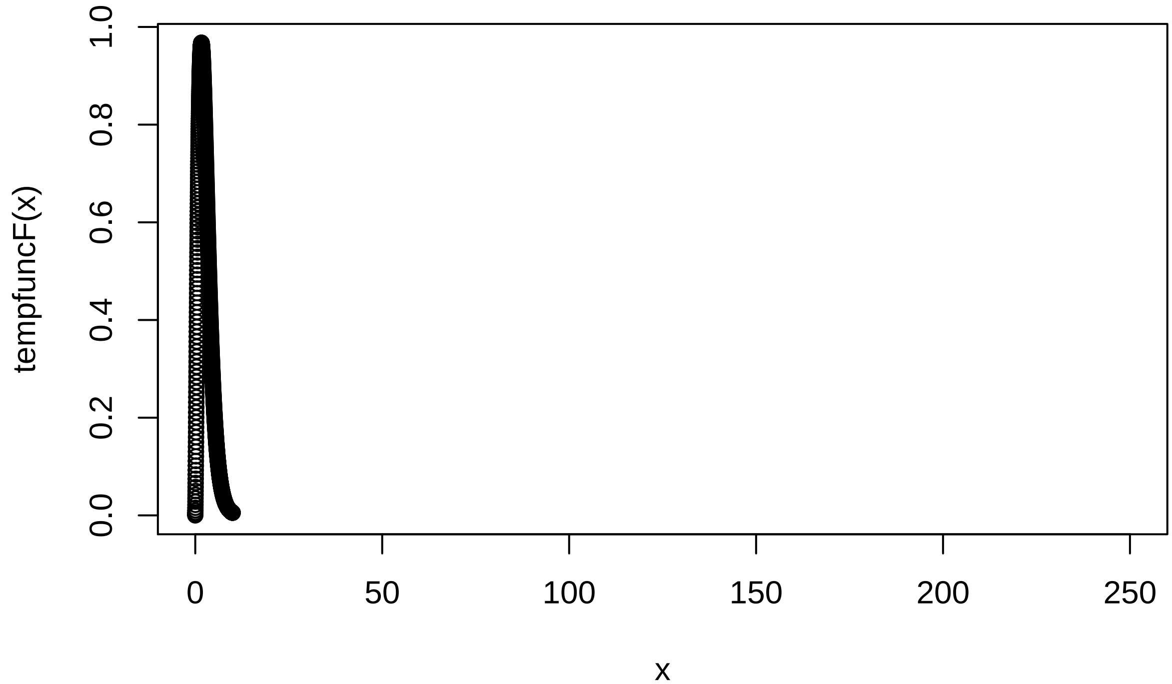
```r
set.seed(14)
x <- seq(0.0001, 10, by = 0.01)
y <- rep(0,n)
rsampling <- function(M){

 for(i in 1:10000){
    u = runif(1, 0, 1)
    y = tempfuncG(u)

    if(tempfuncG(y)*u*M <= tempfuncF(y)){
      return(y)
    }
 }
}
M = 1/(2+sqrt(pi)*2*factorial(theta)/2^(2*theta)/factorial(theta))
samp = replicate(10000, rsampling(M))
hist(samp, breaks=seq(0,250,5) )
```




```
plot(x,tempfuncF(x), xlim=c(0,250))
```



6.3.1

With the prior distributions given, it was determined that the distribution of each normal mixture component was

```
library(HI)
#posterior distribution
normalInvGamma <- function(x,mu,sigma2,a,b,lambda=1){
  l <- ((lambda^.5)/((sigma2*2*pi)^.5))*((b^a)/gamma(a))*((1/sigma2)^(a+1))*exp(-1*(2*b+(lambda*((x-mu)
  return(l)
}
deltalike <- function(delta ,x, mu, sigma2, a,b,lambda=1) {
    prod(delta * normalInvGamma(x, mu, sigma2,a,b,lambda) + (1 - delta) * normalInvGamma(x, mu, sigma2,a
}


#what are we prediciting mu1, mu2, s1, s2
#mu1/mu2 are N(0,100)
#s1/s2 are G(.5,10)


#theta are the mixture mu1/2, sigma1/2


thetaInit = c(.5,1,1,1,1) #delta,mu1,mu2,s1,s2
```

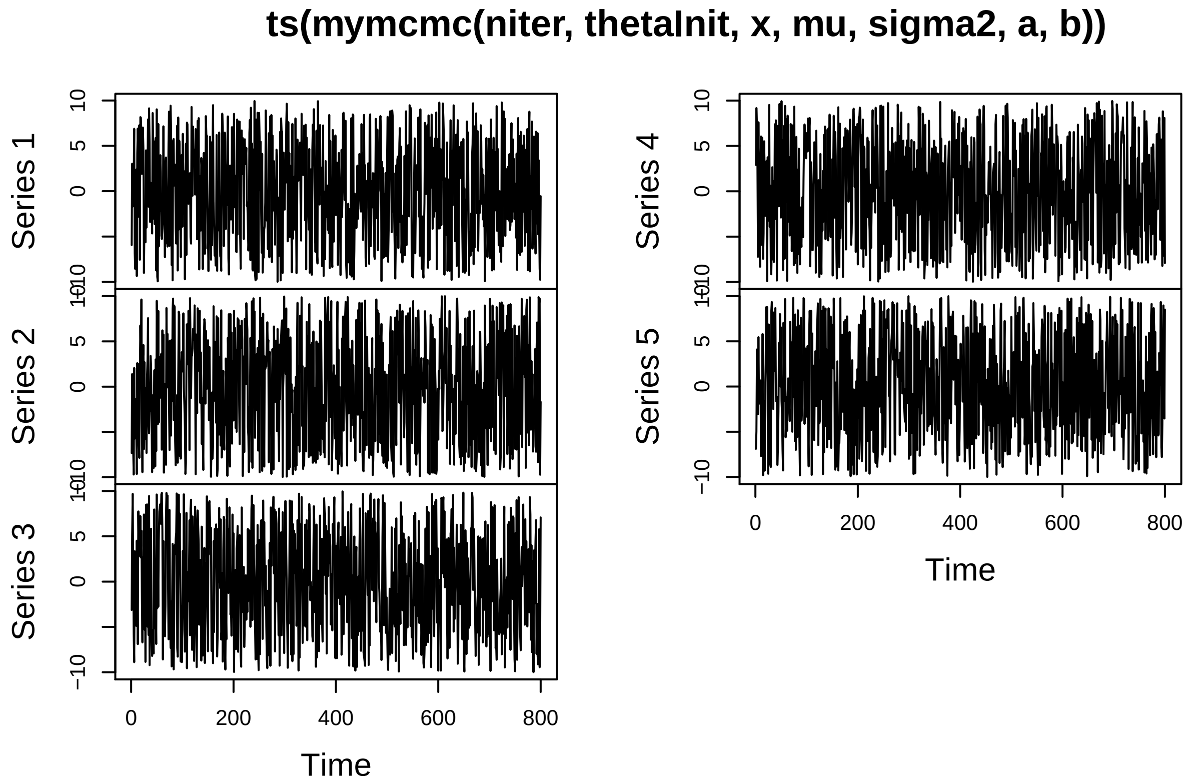
```
niter <- 1000


mymcmc <- function(niter, thetaInit, data, mu, sigma2, a,b, nburn= 200) {
  p <- length(thetaInit)
  thetaCurrent <- thetaInit
  ## define a function for full conditional sampling
  logFC <- function(th, idx) {
    theta <- thetaCurrent
    delta <- thetaCurrent[1]
    theta[idx] <- th
    deltalike(delta, x, mu, sigma2, a,b, 1)
  }
  out <- matrix(thetaInit, niter, p, byrow = TRUE)
  ## Gibbs sampling
  for (i in 2:niter) {
    for (j in 1:p) {
      ## general-purpose arms algorithm
      out[i, j] <- thetaCurrent[j] <-
          HI::arms(thetaCurrent[j], logFC,
                   function(x, idx)((x > -10) * (x < 10)),
                   1, idx = j)
    }
  }
  out[-(1:nburn), ]
}


set.seed(4321)
n=100
mu1 <- rnorm(1,0,100)
mu2 <- rnorm(1,0,100)
s1 <- rgamma(1,.5,scale=10)
s2 <- rgamma(1,.5,scale=10)
s1inv <- s1^-1 #gamma
s2inv <- s2^-1 #gamma
x <- rnorm(n/2, mu1, s1)
y <- rnorm(n/2, mu2, s2)
x <- c(x,y)
mu=0
sigma2=100
a=.5
b=10
par(c(1,1))
```

```
## NULL
```

```
plot(ts(mymcmc(niter, thetaInit, x, mu,sigma2,a,b)))
```



ts(mymcmc(niter, thetaInit, x, mu, sigma2, a, b))
Series 1
Series 2
Series 3
Series 4
Series 5
10
5
0
−10
0
200
400
600
800
Time