

# STAT 202C - HW 3

*John Rapp Farnes / 405461225*

*5/29/2020*

## Problem 7.1

(a)

Let  $\pi \in \text{Gamma}(4.3, 6.2)$  and proposal  $g \in \text{Gamma}(4, 7)$  as well as  $g_2 \in \text{Gamma}(5, 6)$ . In order to perform rejection sampling, we need first to calculate an appropriate  $M$  s.t.  $Mg(x) \geq \pi(x) \forall x$ . Below is a plot of  $g(x)$  and  $\pi(x)$ , corresponding to  $M = 1$ .

```
alpha_pi <- 4.3
beta_pi <- 6.2

alpha_g <- 4
beta_g <- 7

pi_distr <- function(x) {
  dgamma(x, shape = alpha_pi, scale = beta_pi)
}

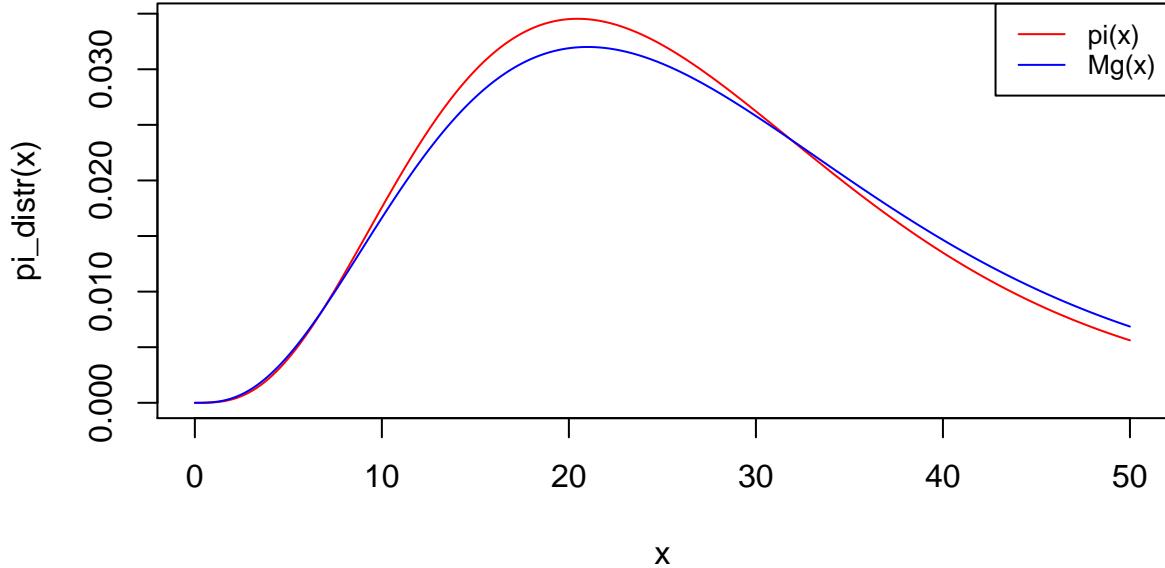
g_distr <- function(x) {
  dgamma(x, shape = alpha_g, scale = beta_g)
}

g_sample <- function(n = 1) {
  rgamma(n, shape = alpha_g, scale = beta_g)
}

pi_div_g_distr <- function(x) {
  pi_distr(x) / g_distr(x)
}

x <- seq(0, 50, 0.01)

plot(x, pi_distr(x), col="red", type="l")
lines(x, g_distr(x), col="blue")
legend("topright", legend=c("pi(x)", "Mg(x)"),
       col=c("red", "blue"), lty=1, cex=0.8)
```



In order to determine  $M$ , we can maximize solve  $Mg(x) \geq \pi(x) \forall x \Leftrightarrow \frac{Mg(x)}{\pi(x)} \geq 1$  for  $M$ . We have

$$\frac{Mg(x)}{\pi(x)} = M \frac{-\beta_g^{\alpha_g} x^{\alpha_g-1} e^{-x/\beta_g} / \Gamma(\alpha_g)}{\beta_\pi^{-\alpha_\pi} x^{\alpha_\pi-1} e^{-x/\beta_\pi} / \Gamma(\alpha_\pi)} = M \frac{\beta_g^{\alpha_\pi} \Gamma(\alpha_\pi)}{\beta_\pi^{\alpha_g} \Gamma(\alpha_g)} x^{\alpha_g - \alpha_\pi} e^{-x(1/\beta_g - 1/\beta_\pi)} \geq 1$$

Finding the minimum by stationary point:

$$\frac{d}{dx} \frac{Mg(x)}{\pi(x)} = 0 \Leftrightarrow e^{-x(1/\beta_g - 1/\beta_\pi)} [(\alpha_g - \alpha_\pi)x^{\alpha_g - \alpha_\pi - 1} - (1/\beta_g - 1/\beta_\pi)x^{\alpha_g - \alpha_\pi}] = 0 \Leftrightarrow x^* = \frac{(\alpha_g - \alpha_\pi)}{1/\beta_g - 1/\beta_\pi} \implies M \geq \frac{\pi(x^*)}{g(x^*)}$$

```
x_star <- (alpha_g-alpha_pi)/(1/beta_g-1/beta_pi)
x_star
```

```
## [1] 16.275
```

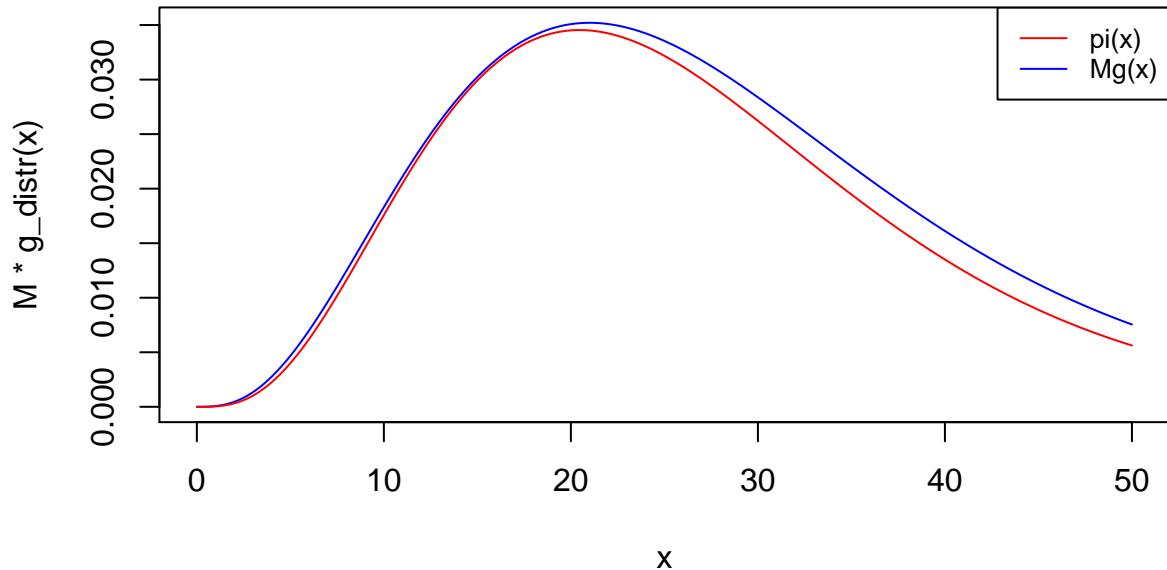
```
pi_distr(x_star) / g_distr(x_star)
```

```
## [1] 1.089486
```

Hence, we can use  $M = 1.1$  to be sure, plot below:

```
M <- 1.1

plot(x, M*g_distr(x), type="l", col="blue")
lines(x, pi_distr(x), col="red")
legend("topright", legend=c("pi(x)", "Mg(x)"),
       col=c("red", "blue"), lty=1, cex=0.8)
```



Below is a plot of the estimated mean, as the number of samples increase, with 100000 in total.

```
N <- 100000
set.seed(2020)

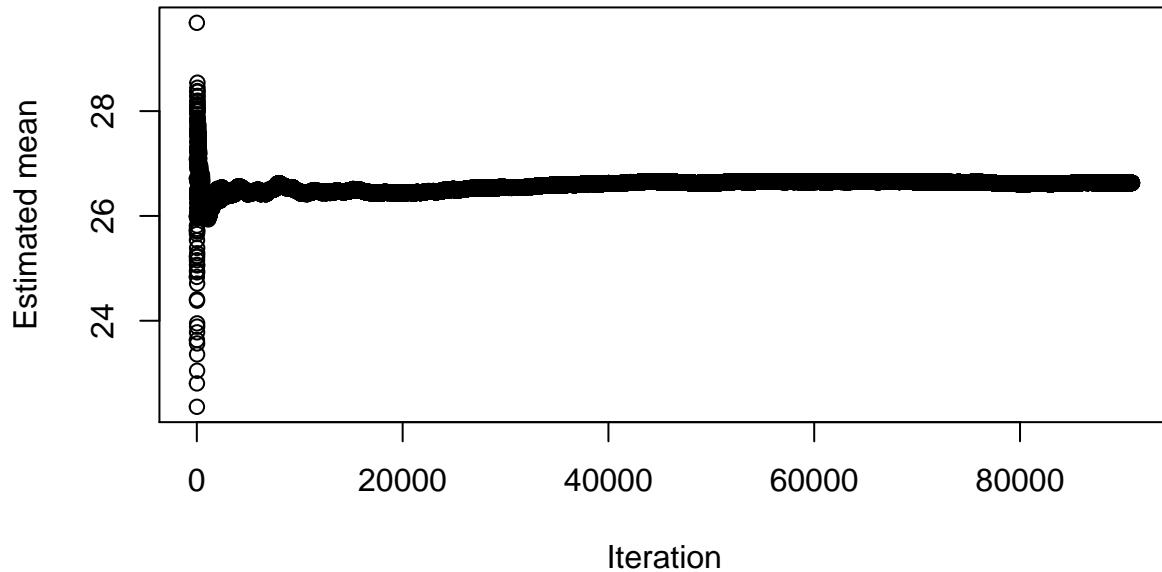
samples <- c()

for (i in 1:N) {
  x <- g_sample(1)
  r <- pi_div_g_distr(x)/M

  u = runif(1)
  if (r <= u) {
    next;
  } else {
    samples <- c(samples, x)
  }
}

mean_sequence <- cumsum(samples) / seq_along(samples)
plot(mean_sequence, main="Estimated mean by rejection sampling with g(x)", xlab = "Iteration", ylab="Es
```

## Estimated mean by rejection sampling with $g(x)$



```
est_mean <- mean_sequence[length(mean_sequence)]
```

The estimated value was 26.6300131, this is fairly close to the theoretical value:

```
theoretical_mean <- alpha_pi * beta_pi  
theoretical_mean
```

```
## [1] 26.66
```

Looking at the plot, we can see that the estimate moves a lot during the first iterations, but settles after less than 10 000 iterations, after which it converges to the estimated value. After 100000 values, it appears to have fully converged.

(b)

Below is an implementation of the Metropolis-Hastings algorithm, and its estimate using  $g$  as a proposal distribution, for 100000 iterations with  $x = 20$  as initial state.

```
metropolis_hastings <- function(q_sample, q_ratio, pi_ratio, N, initial_state) {
  state <- initial_state

  states <- list(state)

  for (i in 1:N) {
    proposed <- q_sample()

    alpha <- q_ratio(state, proposed) * pi_ratio(proposed, state)

    if (alpha >= 1 || runif(1) <= alpha) {
      state <- proposed
    }

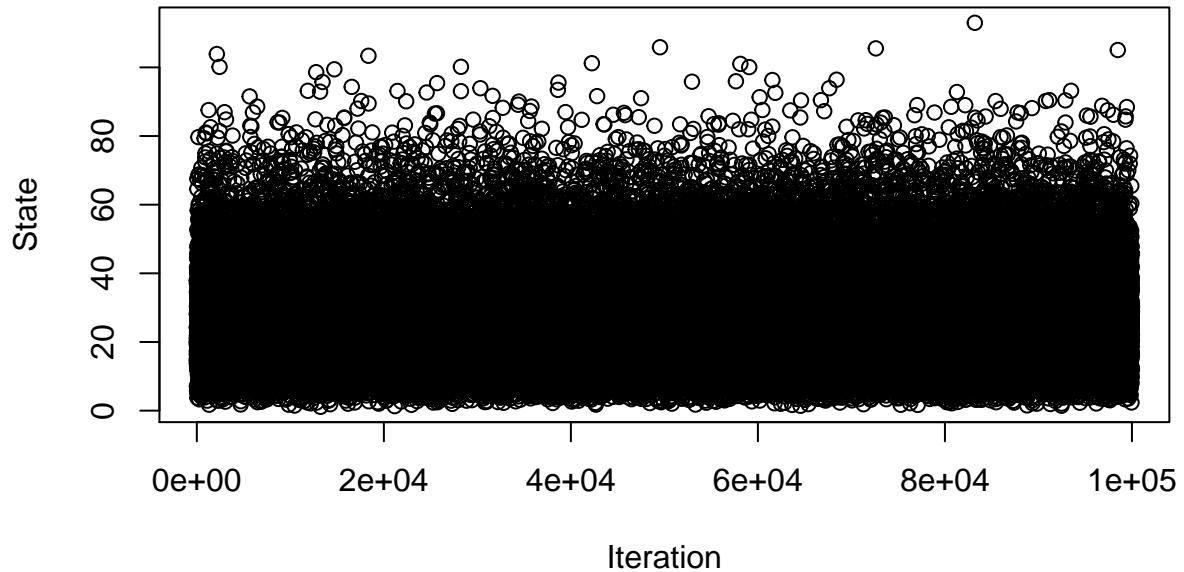
    states <- append(states, list(state))
  }

  return (states)
}

states <- unlist(metropolis_hastings(
  g_sample,
  function(x, y) g_distr(x) / g_distr(y),
  function(x, y) pi_distr(x) / pi_distr(y),
  N,
  20
))

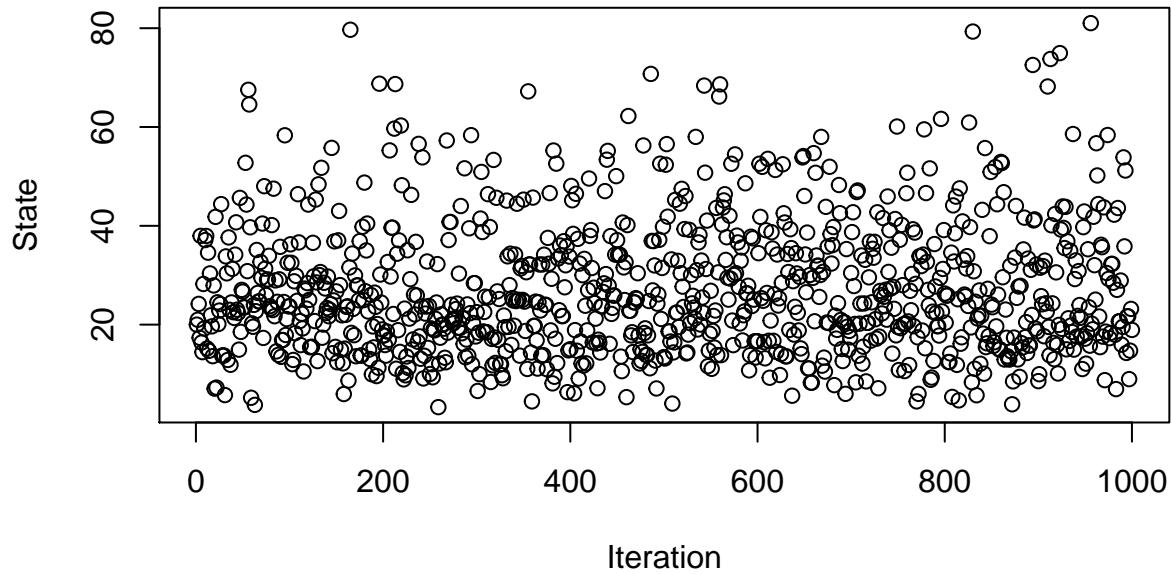
plot(states, main = "MCMC state for g", xlab = "Iteration", ylab = "State")
```

**MCMC state for g**

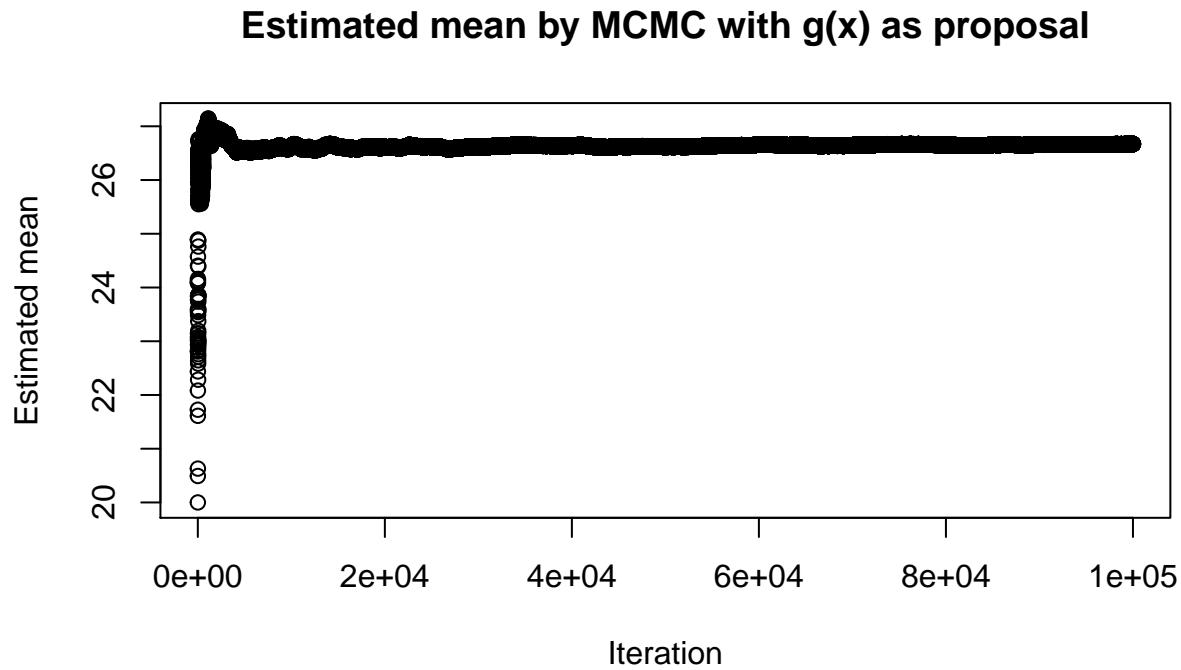


```
plot(states[1:1000], main = "First 1000 MCMC states", xlab = "Iteration", ylab = "State")
```

**First 1000 MCMC states**

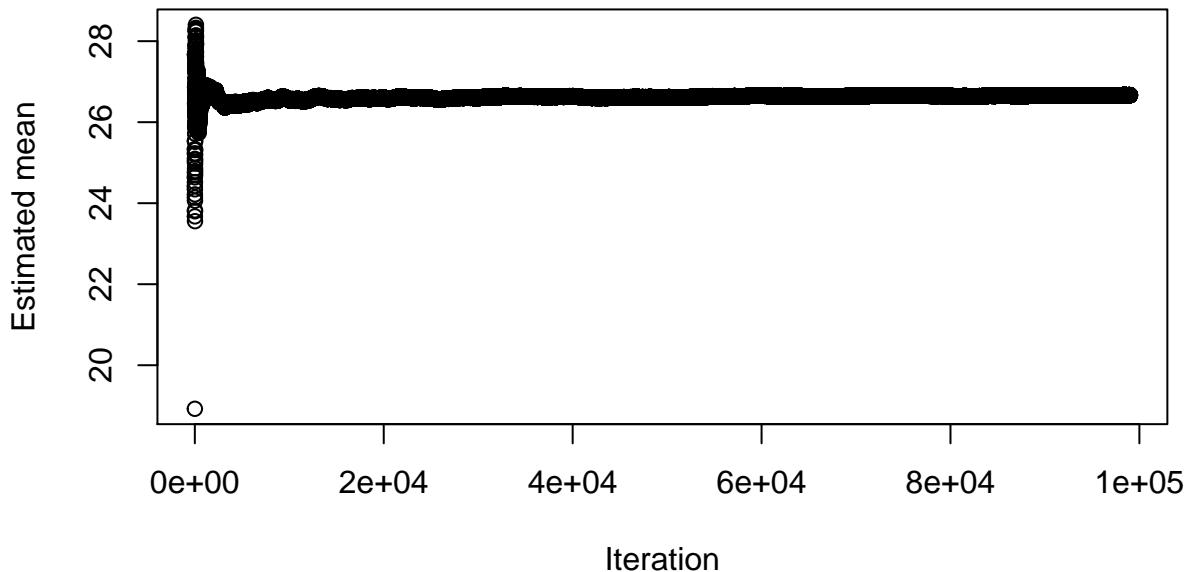


```
mean_sequence <- cumsum(states) / seq_along(states)
plot(mean_sequence, main="Estimated mean by MCMC with g(x) as proposal", xlab = "Iteration", ylab="Estimated mean")
```



```
burn_in_length <- 1000
samples <- states[burn_in_length:length(states)]
mean_sequence <- cumsum(samples) / seq_along(samples)
plot(mean_sequence, main="Estimated mean by MCMC, after the first 1000 iterations", xlab = "Iteration",
```

## Estimated mean by MCMC, after the first 1000 iterations



```
est_mean <- mean_sequence[length(mean_sequence)]
```

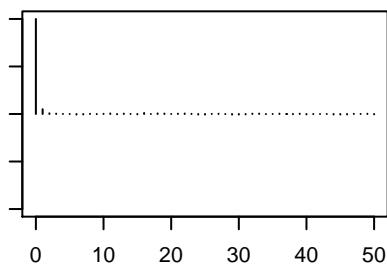
The estimated value was 26.6669955, which again is close to the theoretical value.

Looking at convergence, we can see that the chain quickly appears to move between states consistent with the final distribution, with a short burn-in period. This is however hard to see directly in the plot, as the proposal distribution is very similar to the target. In addition, it quickly moves from neighborhood of the initial state. Using coda, we can analyse the result further. There are many diagnostics of convergence, and following I will show the most relevant ones.

```
library(coda)
effective_percent <- effectiveSize(states) / N
```

The ratio of the effective size and  $N$  was 0.8682331, which is fairly high, meaning most samples are used.

```
par(mar=c(1,1,1,1))
autocorr.plot(states)
```



Looking at the autocorrelation-plot above, it is low for lags greater than 0. Below is the value of the Geweke's Convergence Diagnostic:

```
geweke.diag(states)
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##      var1
## -0.6331
```

We can see that the value is not significant, as its value is much lower than 2, this confirms that the burn in period of the chain is short. Below is the result of the Heidelberger And Welch's Convergence Diagnostic:

```
heidel.diag(states)
```

```
##
##      Stationarity start      p-value
##      test          iteration
## [,1] passed       1        0.514
##
##      Halfwidth Mean Halfwidth
##      test
## [,1] passed     26.7 0.0854
```

The MCMC passed the stationarity test.

(c)

Below is the MCMC estimate using  $g_2$  as a proposal distribution, for 100000 iterations with  $x = 20$  as initial state.

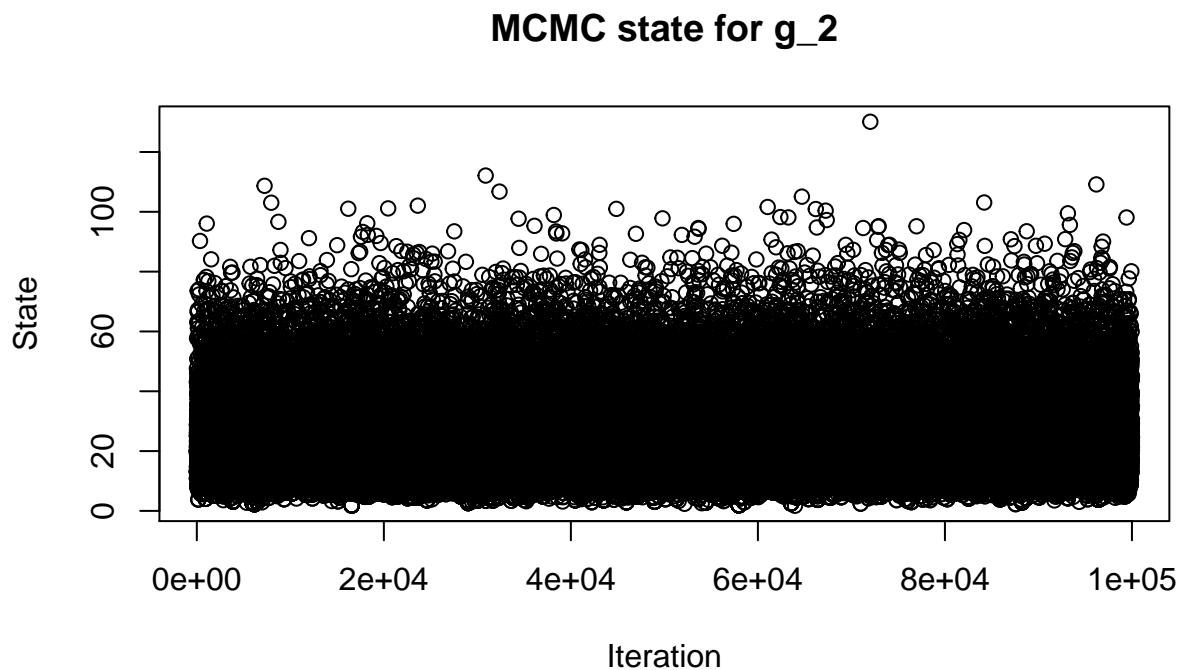
```
alpha_g2 <- 5
beta_g2 <- 6

g2_distr <- function(x) {
  dgamma(x, shape = alpha_g2, scale = beta_g2)
}

g2_sample <- function(n = 1) {
  rgamma(n, shape = alpha_g2, scale = beta_g2)
}

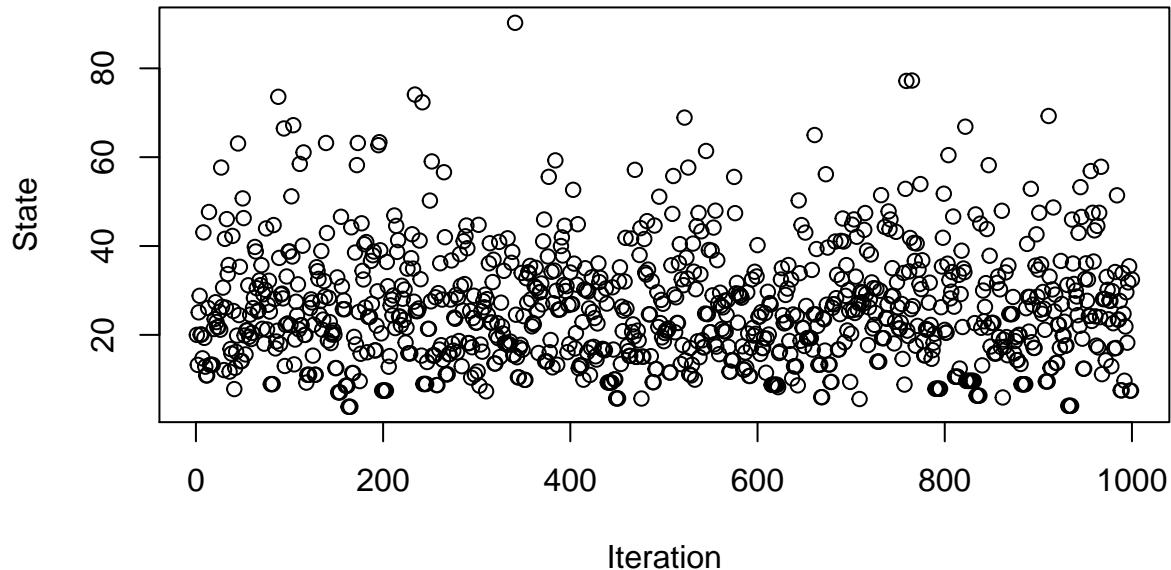
states <- unlist(metropolis_hastings(
  g2_sample,
  function(x, y) g2_distr(x) / g2_distr(y),
  function(x, y) pi_distr(x) / pi_distr(y),
  N,
  20
))

plot(states, main = "MCMC state for g_2", xlab = "Iteration", ylab = "State")
```



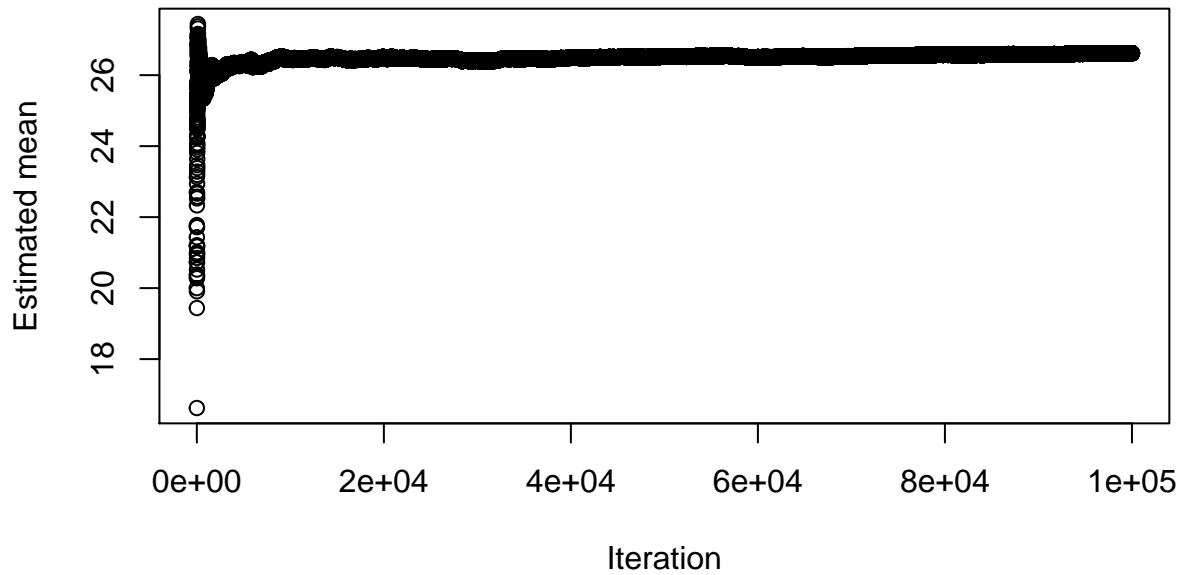
```
plot(states[1:1000], main = "First 1000 MCMC states", xlab = "Iteration", ylab = "State")
```

### First 1000 MCMC states



```
mean_sequence <- cumsum(states) / seq_along(states)
plot(mean_sequence, main="Estimated mean by MCMC with g_2(x) as proposal", xlab = "Iteration", ylab="Est")
```

### Estimated mean by MCMC with $g_2(x)$ as proposal

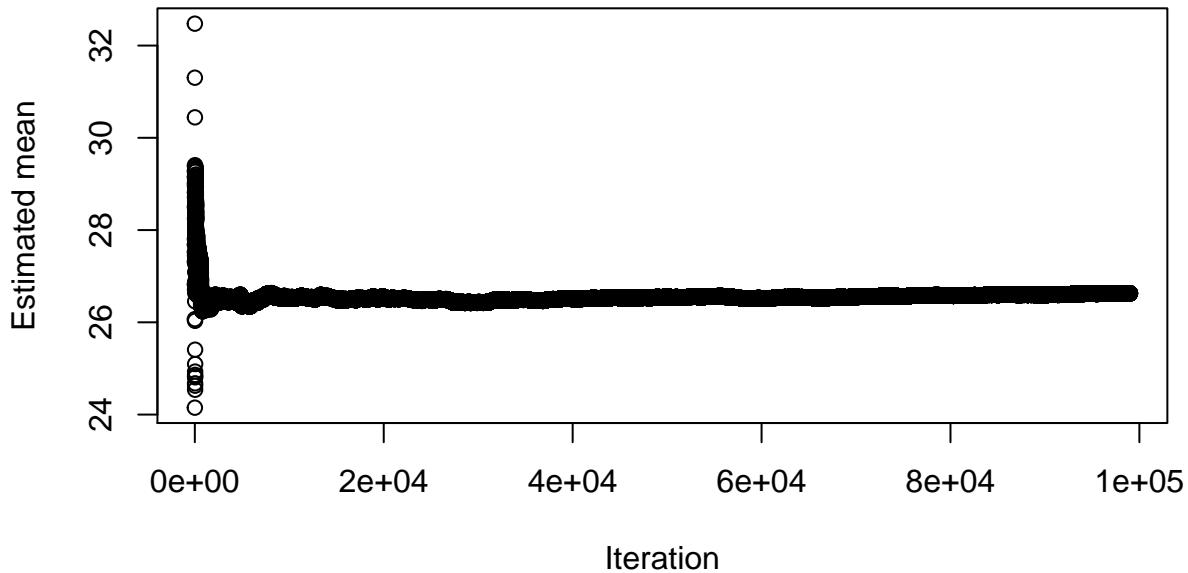


```

burn_in_length <- 1000
samples <- states[burn_in_length:length(states)]
mean_sequence <- cumsum(samples) / seq_along(samples)
plot(mean_sequence, main="Estimated mean by MCMC, after the first 1000 iterations", xlab = "Iteration",

```

## Estimated mean by MCMC, after the first 1000 iterations



```

est_mean <- mean_sequence[length(mean_sequence)]

```

The estimated value was 26.6293824, again close to the theoretical value.

As in (b), the chain quickly appears to move between states consistent with the final distribution, with a short burn-in period.

```

effective_percent <- effectiveSize(states) / N

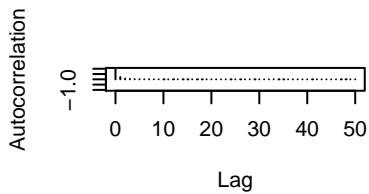
```

The ratio of the effective size and  $N$  was 0.5466949, which is lower than in (b), only about half of the samples are used. As such, this proposal distribution is less appropriate for the estimation of  $\pi(x)$ .

```

autocorr.plot(states)

```



Looking at the autocorrelation-plot above, it is low for lags greater than 0. Below is the value of the Geweke's Convergence Diagnostic:

```
geweke.diag(states)
```

```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   var1
## -1.394
```

We can see that the value is not significant, as its value is lower than 2, which confirms that the burn in time is fairly short. However, this value is greater than in (b). Below is the result of the Heidelberger And Welch's Convergence Diagnostic:

```
heidel.diag(states[1:(length(states) - 1)])
```

```
##
##      Stationarity start      p-value
##      test          iteration
## [1,] passed       30001     0.293
##
##      Halfwidth Mean Halfwidth
##      test
## [1,] passed     26.7 0.125
```

This MCMC passed the stationarity test as well.

## Problem 7.20

```
tab <- read.table("LogisticData.txt", header = TRUE, sep = "", dec = ".")  
  
head(tab)  
  
##   plan np metq age erodd  outcost  
## 1 AVM  1    1  11     0  85.5000  
## 2 AVM  1    1  18     0  0.0000  
## 3 AVM  1    2   6     0  68.3333  
## 4 AVM  1    2   9     1 185.6667  
## 5 AVM  1    3  10     0 114.7143  
## 6 AVM  1    3   9     0  94.7500
```

(a)

Below is a derivation of the likelihood function for the problem:

$$\begin{aligned} \text{logit}(p_{ij}) &= \log \frac{p_{ij}}{1 - p_{ij}} = a + bx_i + cz_{ij} \Leftrightarrow \\ p_{ij} &= \frac{\exp(a + bx_i + cz_{ij})}{1 + \exp(a + bx_i + cz_{ij})} \implies \\ 1 - p_{ij} &= \frac{1 + \exp(a + bx_i + cz_{ij})}{1 + \exp(a + bx_i + cz_{ij})} - \frac{\exp(a + bx_i + cz_{ij})}{1 + \exp(a + bx_i + cz_{ij})} \\ &= \frac{1}{1 + \exp(a + bx_i + cz_{ij})} \\ \therefore P(Y_{ij} = y_{ij} \forall i, j) &= \prod_{i=1}^k P(Y_{ij} = y_{ij} \forall j) \\ &= \prod_{i=1}^k \prod_{j=1}^{n_i} p_{ij}^{y_{ij}} (1 - p_{ij})^{1-y_{ij}} \\ &= \prod_{i=1}^k \prod_{j=1}^{n_i} \left( \frac{\exp(a + bx_i + cz_{ij})}{1 + \exp(a + bx_i + cz_{ij})} \right)^{y_{ij}} \left( \frac{1}{1 + \exp(a + bx_i + cz_{ij})} \right)^{1-y_{ij}} \end{aligned}$$

(b)

Using R, we can make a GLM logistic regression estimation:

```
data <- data.frame(
  emergency = tab$erodd,
  hmo = tab$np,
  health = tab$metq
)

head(data)

##   emergency hmo health
## 1          0    1     1
## 2          0    1     1
## 3          0    1     2
## 4          1    1     2
## 5          0    1     3
## 6          0    1     3

logistic_model <- glm(emergency ~ hmo + health, data = data, family = "binomial")

sum <- summary(logistic_model)
```

The results of the regression are found in the table below:

```
a_mean <- sum$coefficients["(Intercept)", "Estimate"]
a_variance <- sum$coefficients["(Intercept)", "Std. Error"]^2

b_mean <- sum$coefficients["hmo", "Estimate"]
b_variance <- sum$coefficients["hmo", "Std. Error"]^2

c_mean <- sum$coefficients["health", "Estimate"]
c_variance <- sum$coefficients["health", "Std. Error"]^2

tab <- data.frame(
  a = c(a_mean, a_variance),
  b = c(b_mean, b_variance),
  c = c(c_mean, c_variance)
)

row.names(tab) <- c("Mean", "Variance")

tab

##           a         b         c
## Mean -1.97391269 0.162205862 0.284376200
## Variance 0.04889907 0.006357976 0.008600073
```

(c)

Following is an implementation of an MCMC with 100000 iterations, independent normal distributions with sample mean and variance from the logistical model for the parameters, and sample mean as initial state.

```
library(mvtnorm)

likelihood <- function(data, a, b, c) {
  hmos <- unique(data$hmo)
  k <- length(hmos)

  likelihood <- 1

  for (i in 1:k) {
    indices <- which(data$hmo == hmos[i])
    exp_factor <- exp(a + b*hmos[i]+c*data$health[indices])

    factors <- ifelse(data$emergency[indices], exp_factor, 1) / (1 + exp_factor)

    likelihood <- likelihood * prod(factors)
  }

  return (likelihood)
}

mean <- c(a_mean, b_mean, c_mean)
sigma <- diag(c(a_variance, b_variance, c_variance))
sample_proposal <- function() rmvnorm(1, mean, sigma)
proposal_likelihood <- function(x) dmvnorm(x, mean, sigma, log=FALSE)

states <- metropolis_hastings(
  function() c(sample_proposal()),
  function(x, y) proposal_likelihood(x) / proposal_likelihood(y),
  function(x, y) likelihood(data, x[1], x[2], x[3]) / likelihood(data, y[1], y[2], y[3]),
  N,
  mean
)
```

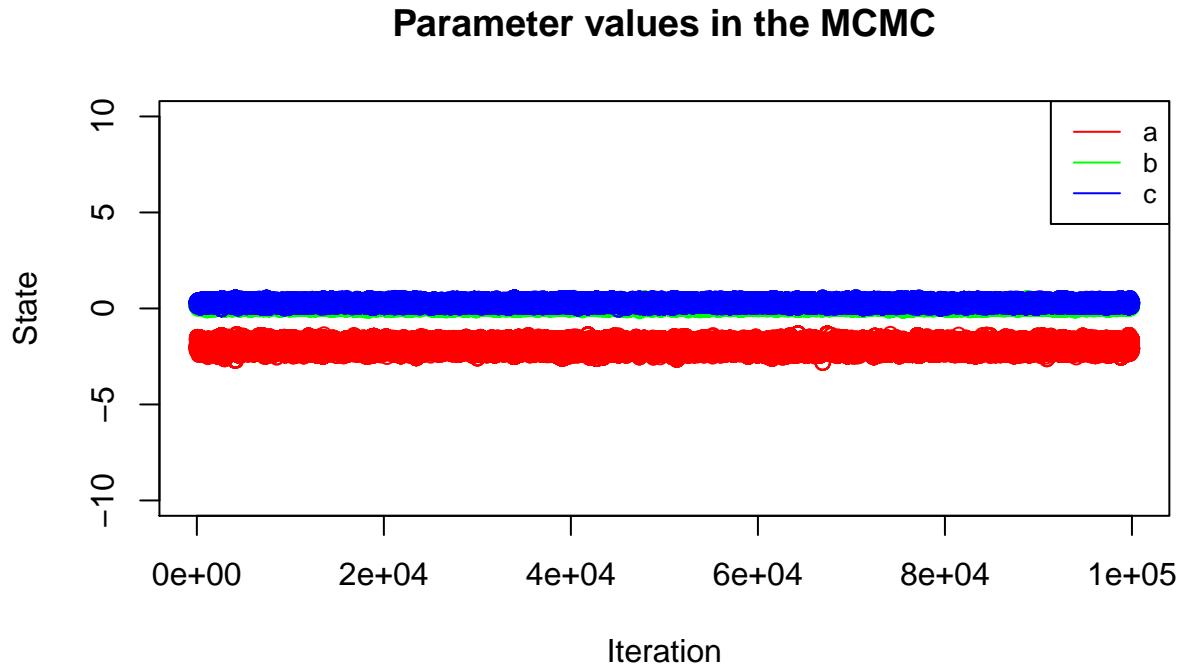
Below are plots of the value of  $a$ ,  $b$  and  $c$  as the MCMC runs:

```
as <- c()
bs <- c()
cs <- c()
for (s in states) {
  as <- c(as, s[1])
  bs <- c(bs, s[2])
  cs <- c(cs, s[3])
}

samples <- data.frame(a=as, b=bs, c=cs)

plot(0:N, samples$a, col="red", ylim=c(-10, 10), main="Parameter values in the MCMC", xlab = "Iteration
points(0:N, samples$b, col="green")
points(0:N, samples$c, col="blue")
```

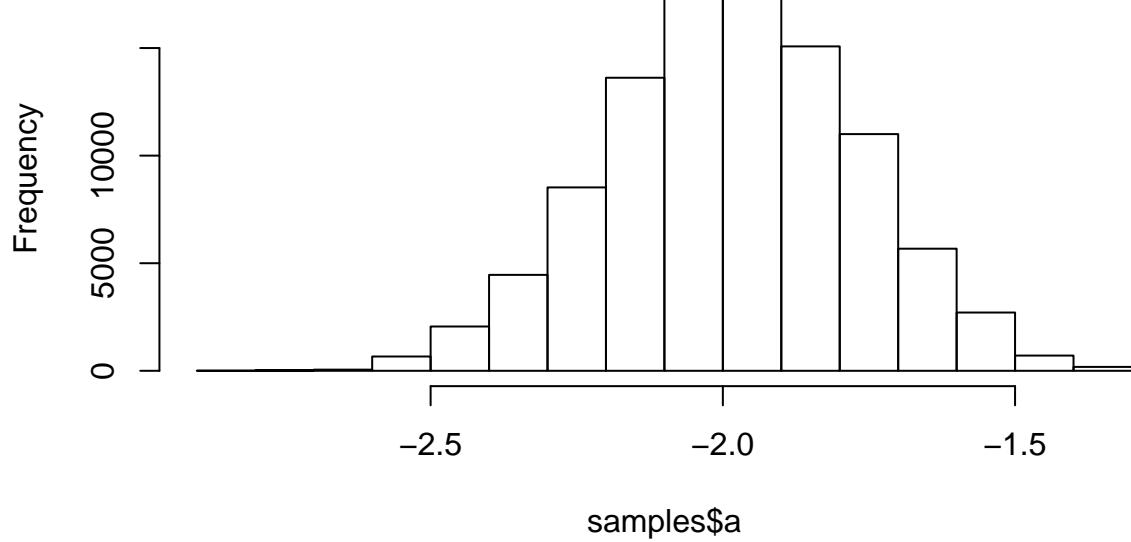
```
legend("topright", legend=c("a", "b", "c"),
       col=c("red", "green", "blue"), lty=1 , cex=0.8)
```



Finally, histograms of the parameter values:

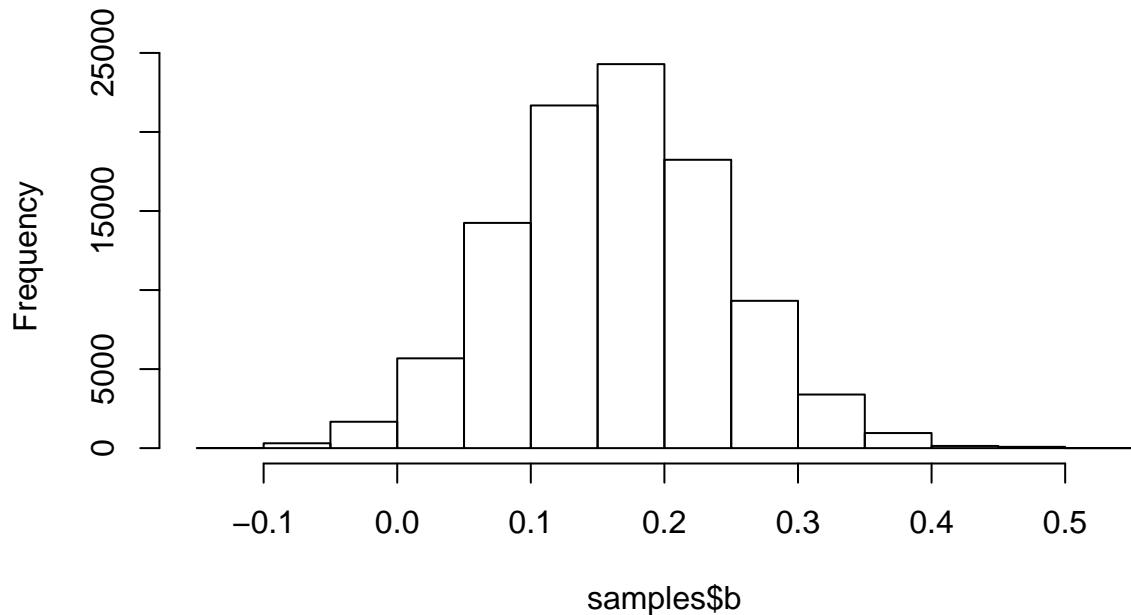
```
hist(samples$a, main="Histogram of a")
```

### Histogram of a



```
hist(samples$b, main="Histogram of b")
```

### Histogram of b



```
hist(samples$c, main="Histogram of c")
```

