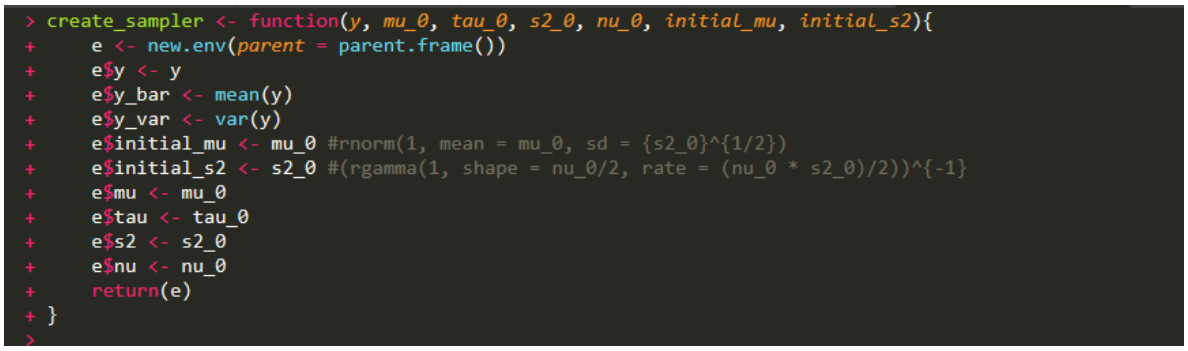
STAT611 HW4 Woojeong Kim

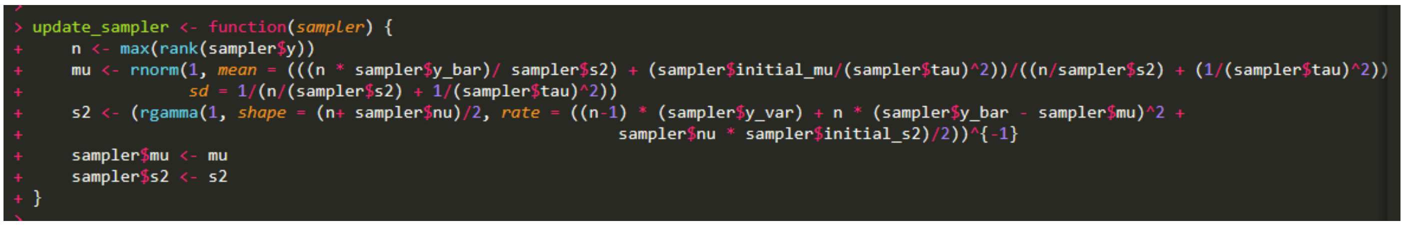
Q1)



```
> create_sampler <- function(y, mu_0, tau_0, s2_0, nu_0, initial_mu, initial_s2){
+     e <- new.env(parent = parent.frame())
+     e$y <- y
+     e$y_bar <- mean(y)
+     e$y_var <- var(y)
+     e$initial_mu <- mu_0 #rnorm(1, mean = mu_0, sd = {s2_0}^{1/2})
+     e$initial_s2 <- s2_0 #(rgamma(1, shape = nu_0/2, rate = (nu_0 * s2_0)/2))^{-1}
+     e$mu <- mu_0
+     e$tau <- tau_0
+     e$s2 <- s2_0
+     e$nu <- nu_0
+     return(e)
+ }
```

1. I set the constructor function "create_sampler" to define a new environment "e". The function uses its variable as y, mu_0, tau_0, s2_0, nu_0, initial_mu and initial_s2. The environment "e" is labeled with these variables. And in the last line, we return e with that labels initialized by variables of our function named as "create_sampler".

This part is necessary for setting starting values to calculate our MCMC inductively.



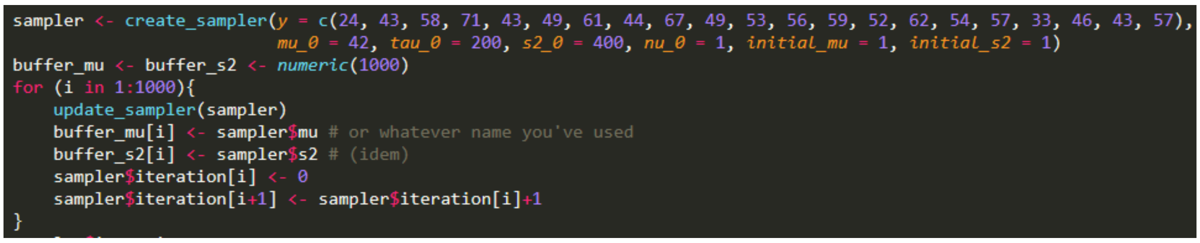
```
> update_sampler <- function(sampler) {
+     n <- max(rank(sampler$y))
+     mu <- rnorm(1, mean = (((n * sampler$y_bar)/ sampler$s2) + (sampler$initial_mu/(sampler$tau)^2))/((n/sampler$s2) + (1/(sampler$tau)^2))
+                sd = 1/(n/(sampler$s2) + 1/(sampler$tau)^2))
+     s2 <- (rgamma(1, shape = (n+ sampler$nu)/2, rate = ((n-1) * (sampler$y_var) + n * (sampler$y_bar - sampler$mu)^2 +
+                                                     sampler$nu * sampler$initial_s2)/2))^{-1}
+     sampler$mu <- mu
+     sampler$s2 <- s2
+ }
```

2. update function is defined by mutable object. From the initial values, we can get the right next "mu" and "s2" from recurrence relations of MCMC. To calculate the next values, we use a variable "sampler" of the function "update_sampler" and the number of components of vector "y" contained in the "create_sampler" function. The recurrence equation for next "mu" and "s2" are stated from 3rd line to 6th line.

In detail, the equations for "n", "mu" and "s2" consist of only one variable "sampler" of "update_sampler" and the variable consists of several labels "y", … , "nu" as stated in the first figure in Q1 to state recurrence equation by using the split labels.

At the last two lines in the statement, calculated "mu" and "s2" is plugged in "sampler$mu" and "sampler$s2" respectively as posteriors. This step means new "mu" and "s2" is defined after calculating the recurrence equation, using the priors.

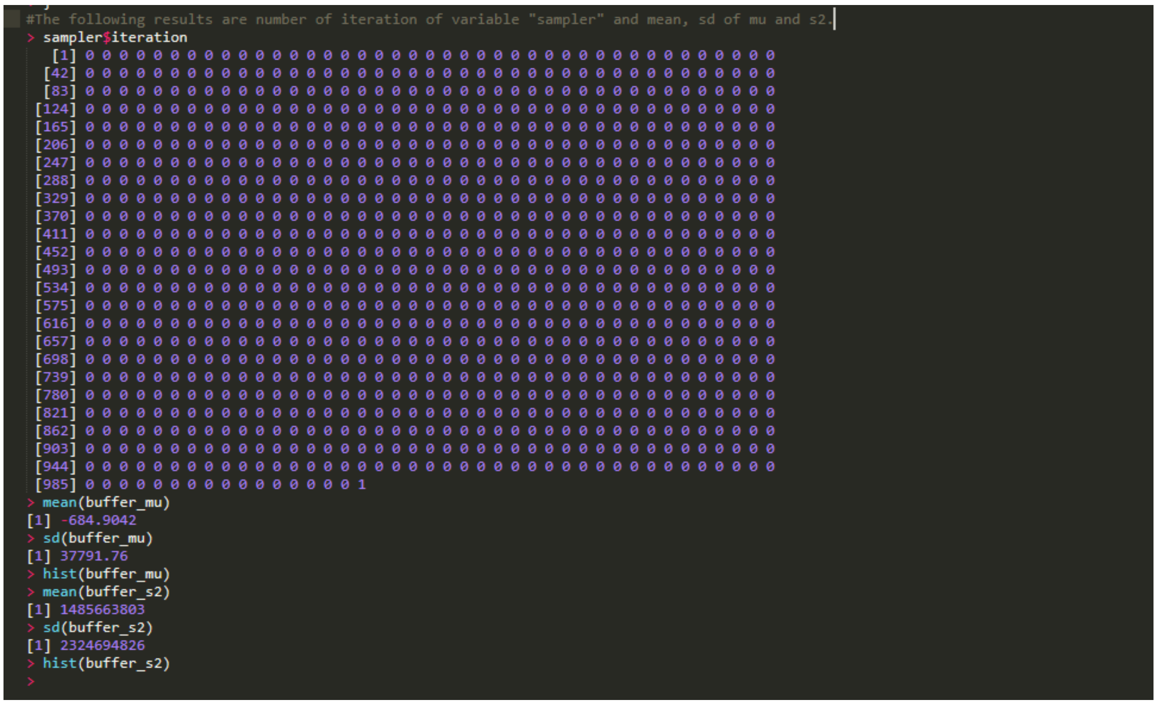
Also, there is no determined value for any variable and thus our "update_sampler" function use only mutable object "sampler".



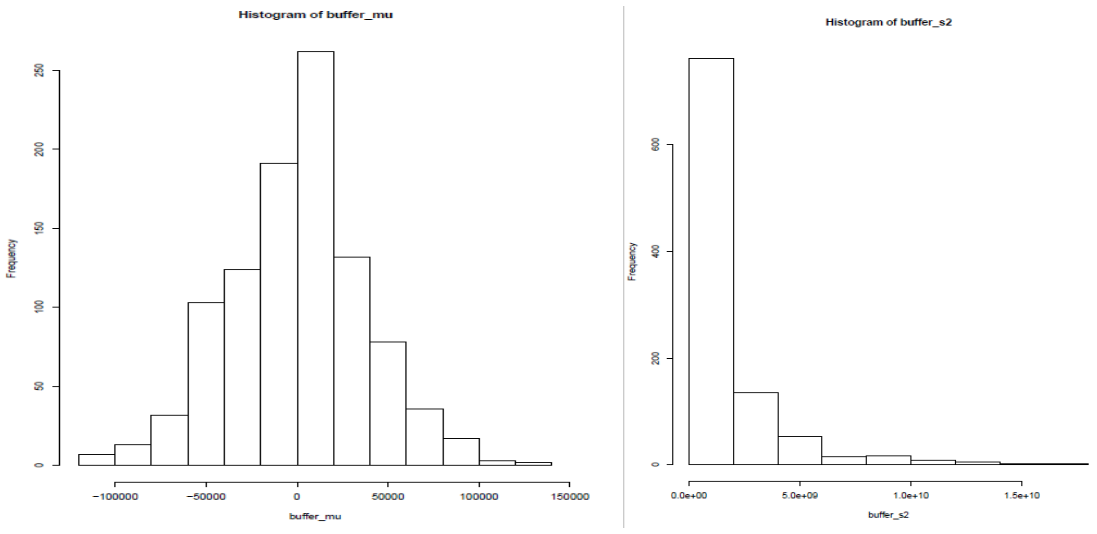
```
sampler <- create_sampler(y = c(24, 43, 58, 71, 43, 49, 61, 44, 67, 49, 53, 56, 59, 52, 62, 54, 57, 33, 46, 43, 57),
                          mu_0 = 42, tau_0 = 200, s2_0 = 400, nu_0 = 1, initial_mu = 1, initial_s2 = 1)
buffer_mu <- buffer_s2 <- numeric(1000)
for (i in 1:1000){
    update_sampler(sampler)
    buffer_mu[i] <- sampler$mu # or whatever name you've used
    buffer_s2[i] <- sampler$s2 # (idem)
    sampler$iteration[i] <- 0
    sampler$iteration[i+1] <- sampler$iteration[i]+1
}
```

3. we can use the function "create_sampler" with determined initial value "y", "mu_0", … , "initial_s2". To observe the trend of variables "mu" and "s2" as going further in inductive chain of MCMC, we use "buffer" as recording variable. For buffer_mu and buffer_s2 with 1000 indexes(Defined in 3rd line), write the "for" statement(4th line). Through 1000 indexes of buffer_mu and buffer_s2, sampler$mu and sampler$s2 are substituted for each index i of buffer_mu and buffer_s2 each time they iterate the "for" statement.

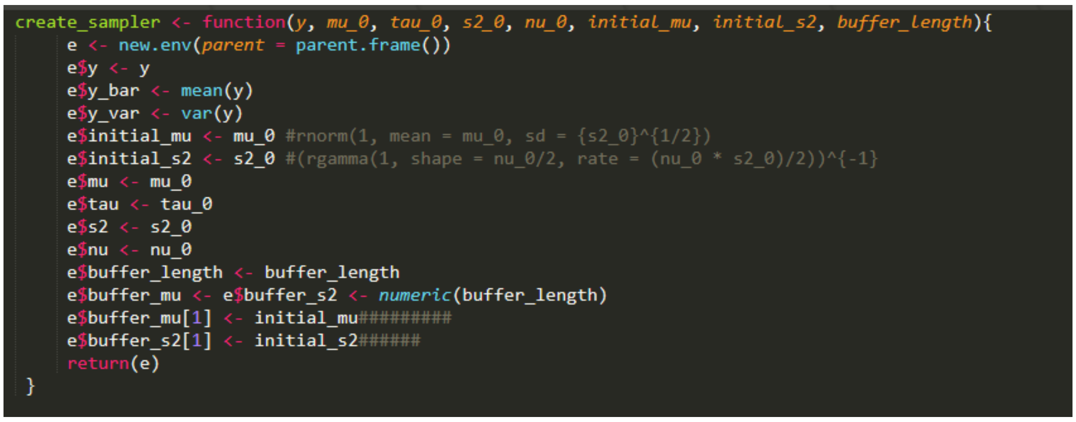
4. The results are as follows.



```
#The following results are number of iteration of variable "sampler" and mean, sd of mu and s2.
> sampler$iteration
   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [42] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [83] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [124] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [165] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [206] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [247] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [288] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [329] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [370] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [411] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [452] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [493] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [534] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [575] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [616] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [657] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [698] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [739] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [780] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [821] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [862] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [903] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [944] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [985] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
> mean(buffer_mu)
[1] -684.9042
> sd(buffer_mu)
[1] 37791.76
> hist(buffer_mu)
> mean(buffer_s2)
[1] 1485663803
> sd(buffer_s2)
[1] 2324694826
> hist(buffer_s2)
>
```



Q2



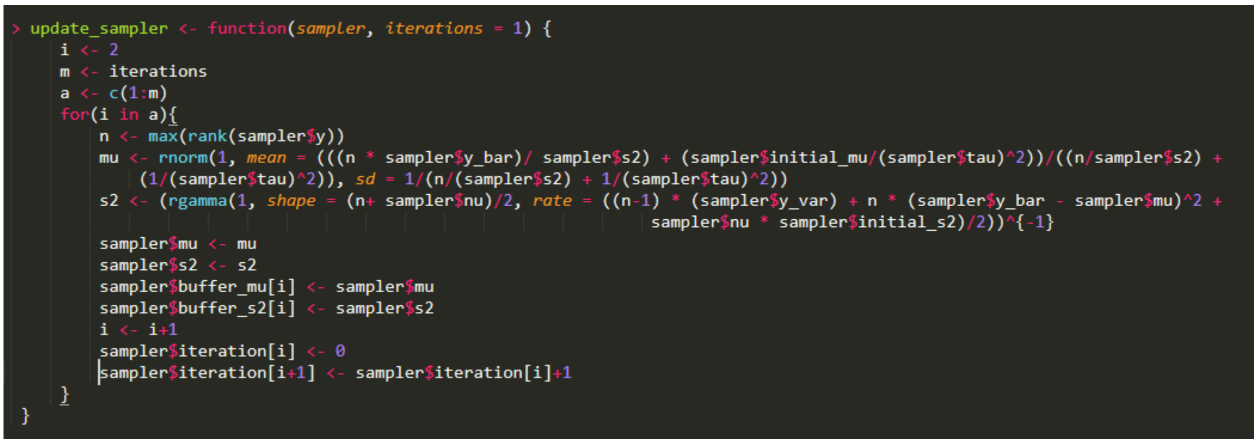
```
create_sampler <- function(y, mu_0, tau_0, s2_0, nu_0, initial_mu, initial_s2, buffer_length){
     e <- new.env(parent = parent.frame())
     e$y <- y
     e$y_bar <- mean(y)
     e$y_var <- var(y)
     e$initial_mu <- mu_0 #rnorm(1, mean = mu_0, sd = {s2_0}^{1/2})
     e$initial_s2 <- s2_0 #(rgamma(1, shape = nu_0/2, rate = (nu_0 * s2_0)/2))^{-1}
     e$mu <- mu_0
     e$tau <- tau_0
     e$s2 <- s2_0
     e$nu <- nu_0
     e$buffer_length <- buffer_length
     e$buffer_mu <- e$buffer_s2 <- numeric(buffer_length)
     e$buffer_mu[1] <- initial_mu#########
     e$buffer_s2[1] <- initial_s2######
     return(e)
 }
```

1. For the second question, we construct the buffer that can store mu and s2 from the iteration when the iteration counting number is big enough as we want. To do this, modification of constructor function "create_sampler" is needed. Also, to store the result

values from each iteration, we should modify the updating function “update_sampler” in Q1.

First, in the constructor function “cerate_sampler”, add the definition of last variable “beffer_length” in the function(1st line). By using this variable, we initialize the new labels e$buffer_length to count the number of interation proceeded. Also, as buffer in Q1, the vectors e$buffer_mu and e$buffer_s2 with indexes of which sizes are same with “buffer_length”(13th line). For 1st indexes of buffer_mu and buffer_s2, initial_mu and initial_s2 is substituted. At the end, we get the “mutable” variable “e”.

From this modification, we can get wanted capacity of the buffer index size only by putting designated number into “buffer_length” of constructor function. This is because the buffer_length is mutable variable, not determined number.



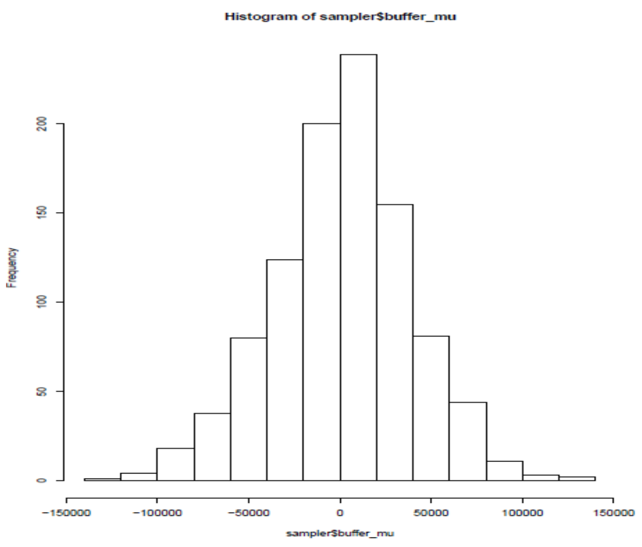
```
> update_sampler <- function(sampler, iterations = 1) {
    i <- 2
    m <- iterations
    a <- c(1:m)
    for(i in a){
        n <- max(rank(sampler$y))
        mu <- rnorm(1, mean = (((n * sampler$y_bar)/ sampler$s2) + (sampler$initial_mu/(sampler$tau)^2))/((n/sampler$s2) +
            (1/(sampler$tau)^2)), sd = 1/(n/(sampler$s2) + 1/(sampler$tau)^2))
        s2 <- (rgamma(1, shape = (n+ sampler$nu)/2, rate = ((n-1) * (sampler$y_var) + n * (sampler$y_bar - sampler$mu)^2 +
                                                                    sampler$nu * sampler$initial_s2)/2))^{-1}
        sampler$mu <- mu
        sampler$s2 <- s2
        sampler$buffer_mu[i] <- sampler$mu
        sampler$buffer_s2[i] <- sampler$s2
        i <- i+1
        sampler$iteration[i] <- 0
        sampler$iteration[i+1] <- sampler$iteration[i]+1
    }
}
```

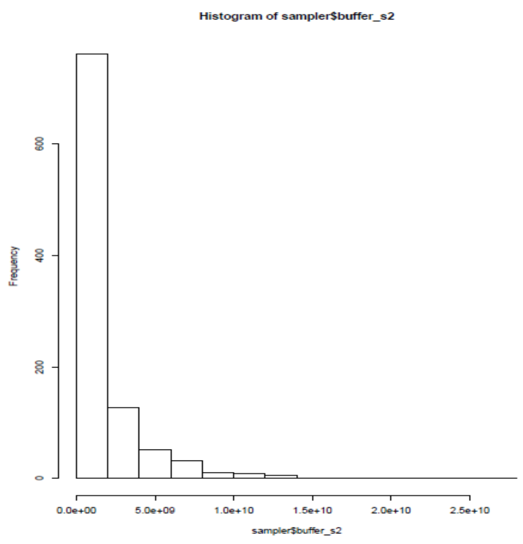
2. Second, let us modify the updating function “update_sampler” in Q1. Likewise constructor function, I added variable “iterations” to the function(1st line). For making the loop to fill the indexes of buffer_mu and buffer_s2 from computing the inductive variables of MCMC, the for statement is used. To put the variables in the for statement, set the starting variable of index for buffer as 1(2nd line). Also putting new variable m as iterations that we want for index size of buffer. From these variables, set the vector a as (1, 2, …, m-1, m) where m = variable “iterations” of update_sampler function. Now, “for” statement is used to get inductive sequence of MCMC during index of buffer is from 1 to variable “iterations”.

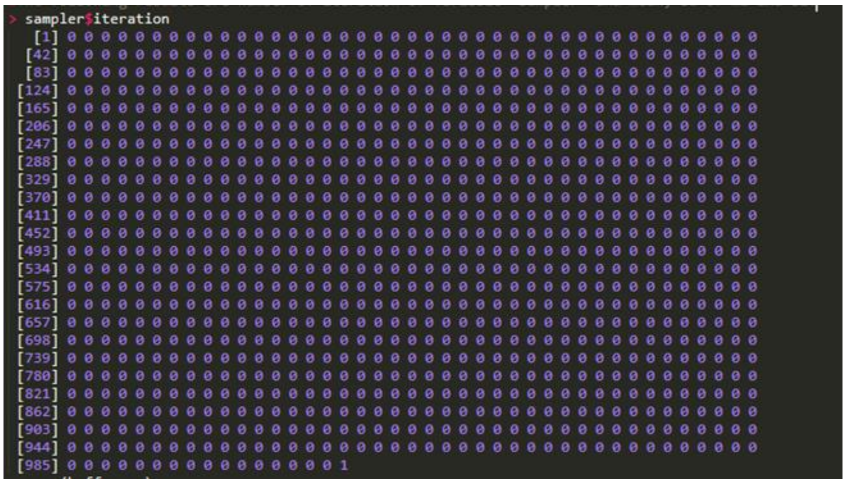
When we observe each iteration of the for statemenr, if index i is in the range 1 to m, then mu and s2 is computed and is substituted to the next mu and s2. Also, as we want, buffer_mu and buffer_s2 contain them as index whenever each new mu and s2 come.

3. For these process, the computing and recording of next mu and s2 are recorded in the buffer_mu of the form of vector objects. Therefore, If we determine sampler variable and iterations variable of function update_sampler, the function make MCMC sequence from determined initial variables mu, nu, tau and s2 from the sampler variable and compute until the sequence display the size we wanted and record them in buffer successively by for statement.

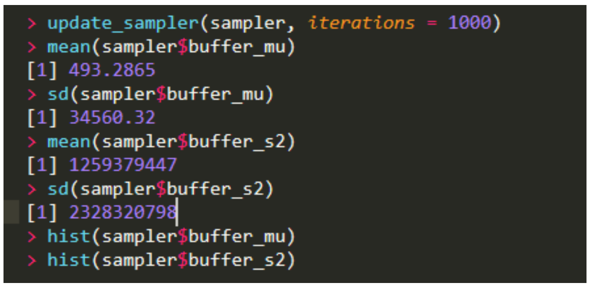
The following are results of sampler$buffer_mu and sampler$buffer_s2.







```
> sampler$iteration
   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [42] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  [83] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [124] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [165] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [206] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [247] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [288] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [329] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [370] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [411] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [452] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [493] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [534] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [575] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [616] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [657] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [698] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [739] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [780] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [821] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [862] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [903] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [944] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [985] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```



```
> update_sampler(sampler, iterations = 1000)
> mean(sampler$buffer_mu)
[1] 493.2865
> sd(sampler$buffer_mu)
[1] 34560.32
> mean(sampler$buffer_s2)
[1] 1259379447
> sd(sampler$buffer_s2)
[1] 2328320798
> hist(sampler$buffer_mu)
> hist(sampler$buffer_s2)
```