Homework 5 on MCMC

ZHUOHUI LIANG zl2074

Due: 04/18/2020, by 11:59pm

Problem 1

Derive the posterior distributions in the following settings:

1. Suppose $X_1, ..., X_n$ iid sample from $N(\theta, \sigma^2)$ distribution, the prior distribution of θ is $N(\mu, \tau^2)$, derive the posterior distribtuion of θ given $\mathbf{X}$:

$$
\begin{aligned}
Pr[\theta|X] = \frac{Pr[\theta] * Pr[X|\theta]}{Pr[X]} &\propto exp(-\frac{(\mu-\theta)^2}{\tau^2}) * exp(-\frac{(\sum X - \theta)^2}{\sigma^2}) & (1)\\
&= exp[\frac{-(\sigma^2(\mu^2 - 2\mu\theta + \theta^2) + \tau^2(\sum X^2 - 2\sum X\theta + \theta^2))}{\tau^2\sigma^2}] & (2)\\
&\propto exp(-[\frac{\theta^2 - 2\theta(\frac{\mu}{\tau^2} + \frac{\sum X}{\sigma^2}) + (\frac{\mu}{\tau^2} + \frac{\sum X}{\sigma^2})^2}{(\frac{1}{\tau^2} + \frac{n}{\sigma})^{-1}}]) & (3)
\end{aligned}
$$

As such, $\theta|X \sim N((\frac{\mu}{\tau^2} + \frac{\sum X}{\sigma^2}), (\frac{1}{\tau^2} + \frac{n}{\sigma})^{-1})$

2.Suppose $X_1, ..., X_n$ iid sample from $U(0, \theta)$ distribution, the prior distribution of θ is Pareto distribution with pdf

$$\pi(\theta) = \frac{\alpha\beta^\alpha}{\theta^{\alpha+1}} I\{\theta \geq \beta\}$$

with known β and α

$$
\begin{aligned}
Pr[\theta|X] &\propto L(X|\theta) * \pi(\theta) & (4)\\
&= \theta^{-n} * \frac{\alpha\beta^\alpha}{\theta^{\alpha+1}} I(\theta \geq \beta) & (5)\\
&= \frac{\alpha\beta^\alpha}{\theta^{n+\alpha+1}} I(\theta \geq \beta) & (6)
\end{aligned}
$$

Answer: your answer starts here...

```
#R codes:
```

Problem 2

Suppose there are three possible weathers in a day: rain, nice, cloudy. The transition probabilities are

rain nice cloudy

rain 0.5 0.5 0.25

nice 0.25 0 0.25

cloudy 0.25 0.5 0.5

where the columns represent the `origin" and the rows represent the`destination" of each step. The initial probabilities of the three states are given by (0.5,0, 0.5) for (rain, nice, cloudy). Answer the following questions

1. Compute the probabilities of the three states on the next step of the chain.
2. Find the stationary distribution of the chain
3. Write an R algorithm for the realization of the chain and illustrate the feature of the chain.

Answer: your answer starts here...

```
K = matrix(c(.5, .5, .25, .25, 0, .25, .25, .5, .5), 3, 3, byrow = T)

init = c(0.5, 0, .5)


# function
easychain = function(init, transit, step = Inf) {

  state = K %*% init

  prev_state = init

  i = 1
  while (any(abs(prev_state - state) > 1e-6 & i < step)) {
    i  = i + 1
    prev_state = state
    state = K %*% state
  }
  return(state)
}
```

```
# first step
easychain(init,K,1)
```

```
##       [,1]
## [1,] 0.375
## [2,] 0.250
## [3,] 0.375
```

```r
# converge value/ stationary
easychain(init,K)
```

```
##           [,1]
## [1,] 0.4000001
## [2,] 0.1999998
## [3,] 0.4000001
```

```r
# proved of stationary
easychain(c(.1,.3,.6),K)
```

```
##           [,1]
## [1,] 0.3999997
## [2,] 0.2000001
## [3,] 0.4000002
```

```r
easychain(diag(3),K)
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.4000001 0.4000001 0.3999999
## [2,] 0.2000000 0.1999998 0.2000000
## [3,] 0.3999999 0.4000001 0.4000001
```

problem 3 Consider the bivariate density

$$f(x,y) \propto \binom{n}{x} y^{x+a-1}(1-y)^{n-x+b-1}, x = 0, 1, \cdots, n, 0 \le y \le 1$$

Complete the following tasks:

1. Write the algorithm of the Gibbs sampler, implement it in R program, and generate a chain with target joint density $f(x,y)$
2. Use a Metropolis sampler to generate a chain with target joint density $f(x;y)$ and implement in R program.
3. Suppose $n = 30, a = 9, b = 14$, use simulations to compare the performance of the above two methods.

Answer: your answer starts here...

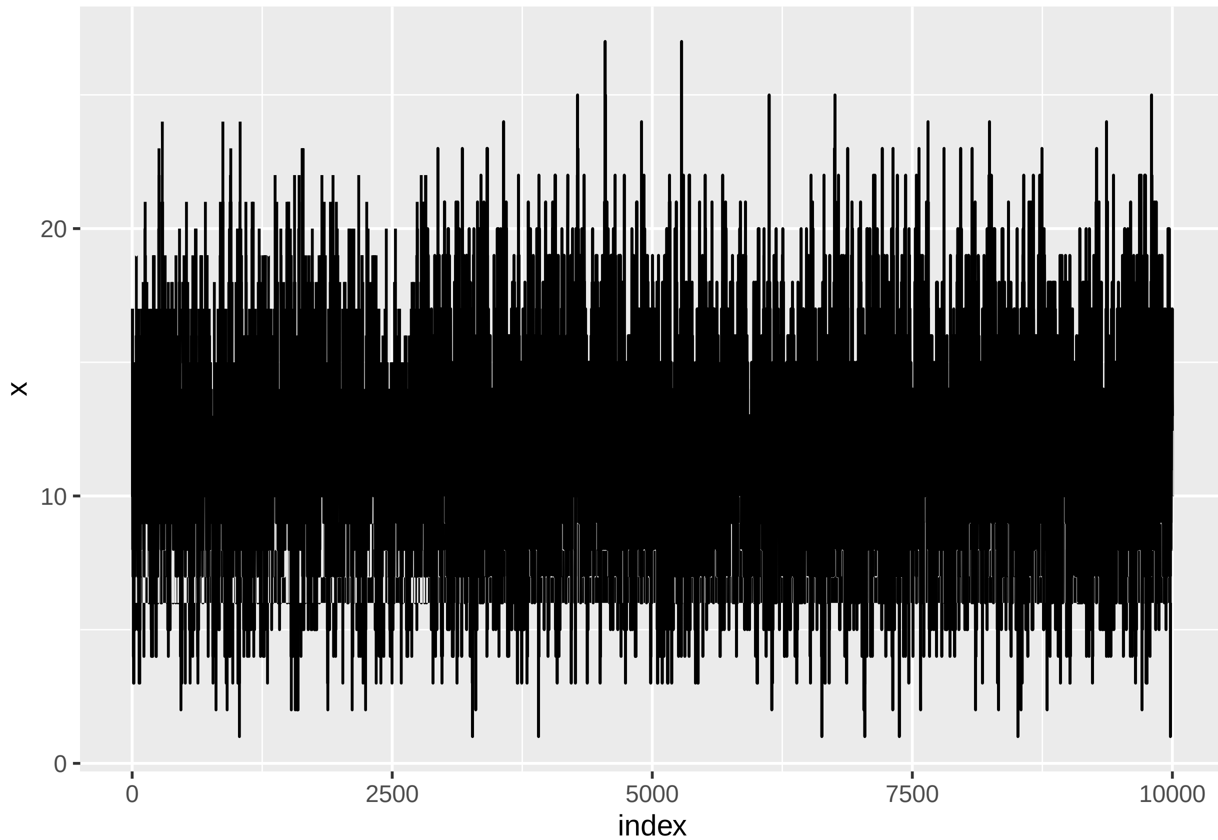
1

$$\begin{aligned}
f(x|y) &= \frac{f(x,y)}{f(y)} && (7)\\
&= f(x,y)/\sum_x f(x,y) && (8)\\
&= f(x,y)/y^{a-1}(1-y)^{b-1}\sum \binom{n}{x} y^x(1-y)^{n-x} && (9)\\
&= \frac{\binom{n}{x} y^{x+a-1}(1-y)^{n-x+b-1}}{y^{a-1}(1-y)^{b-1}} && (10)\\
&= Bin(n,y) && (11)
\end{aligned}$$

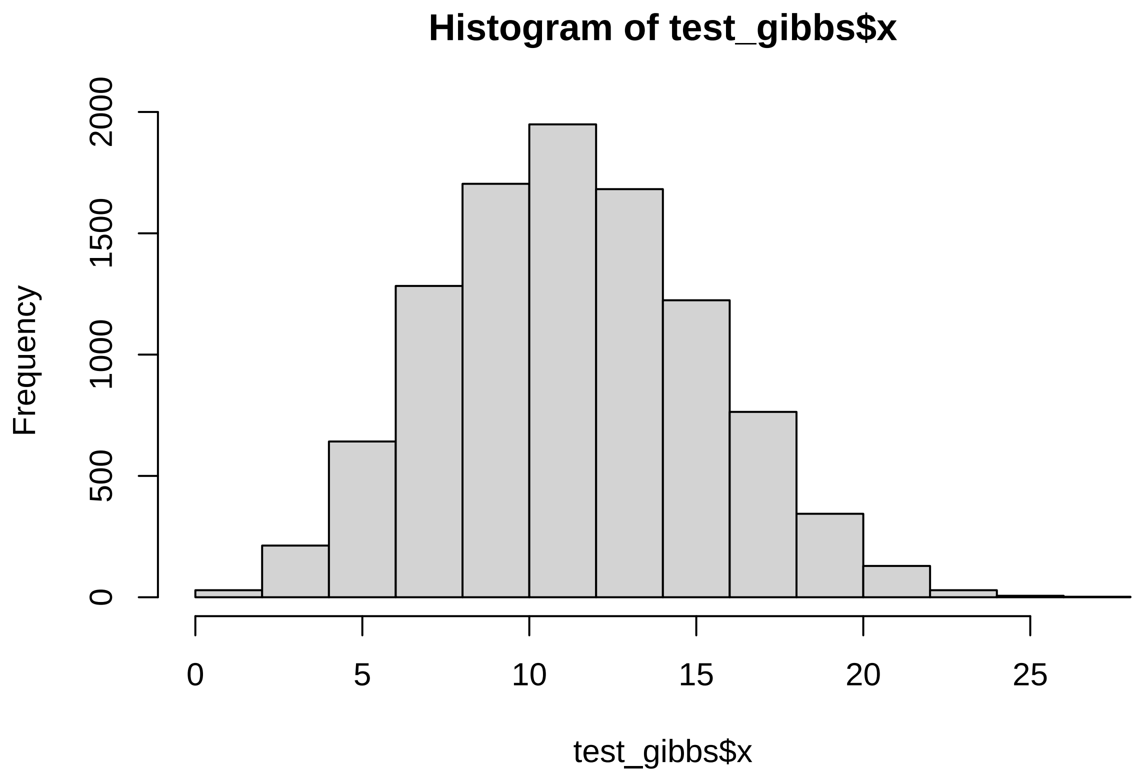
$$
\begin{aligned}
f(y|x) &= \frac{f(x,y)}{f(x)} && (12)\\
&= f(x,y)/\binom{n}{x}\int_y y^{x+a-1}(1-y)^{n-x+b-1} && (13)\\
&= f(x,y)/\binom{n}{x}B(x+a,n-x+b) && (14)\\
&= \frac{\binom{n}{x}y^{x+a-1}(1-y)^{n-x+b-1}}{\binom{n}{x}B(x+a,n-x+b)} && (15)\\
&= Beta(x+a,n-x+b) && (16)
\end{aligned}
$$

```r
gibbs =
  function(n,
           a,
           b,
           step = 1e+4,
           burn = F,
           x_init = NA,
           y_init = NA,
           .tol = 1e-6) {
    if (is.na(x_init))
      x_init = runif(1,1,10)%/%1

    if (is.na(y_init))
      y_init = runif(1)

    x = c(x_init)
    y = c(y_init)
    iter = 1

    while (iter < step) {
      x = c(x, rbinom(1, n, y[iter]))
      y = c(y, rbeta(1, x[iter] + a, n - x[iter] + b))
      iter = iter + 1
    }


    index = 1:step

    if (burn) {
      index = index[-c(1:burn)]
      x = x[-c(1:burn)]
      y = y[-c(1:burn)]
    }

    return(list(x = x,
                y = y,
                index = index))
  }


set.seed(123123)
```

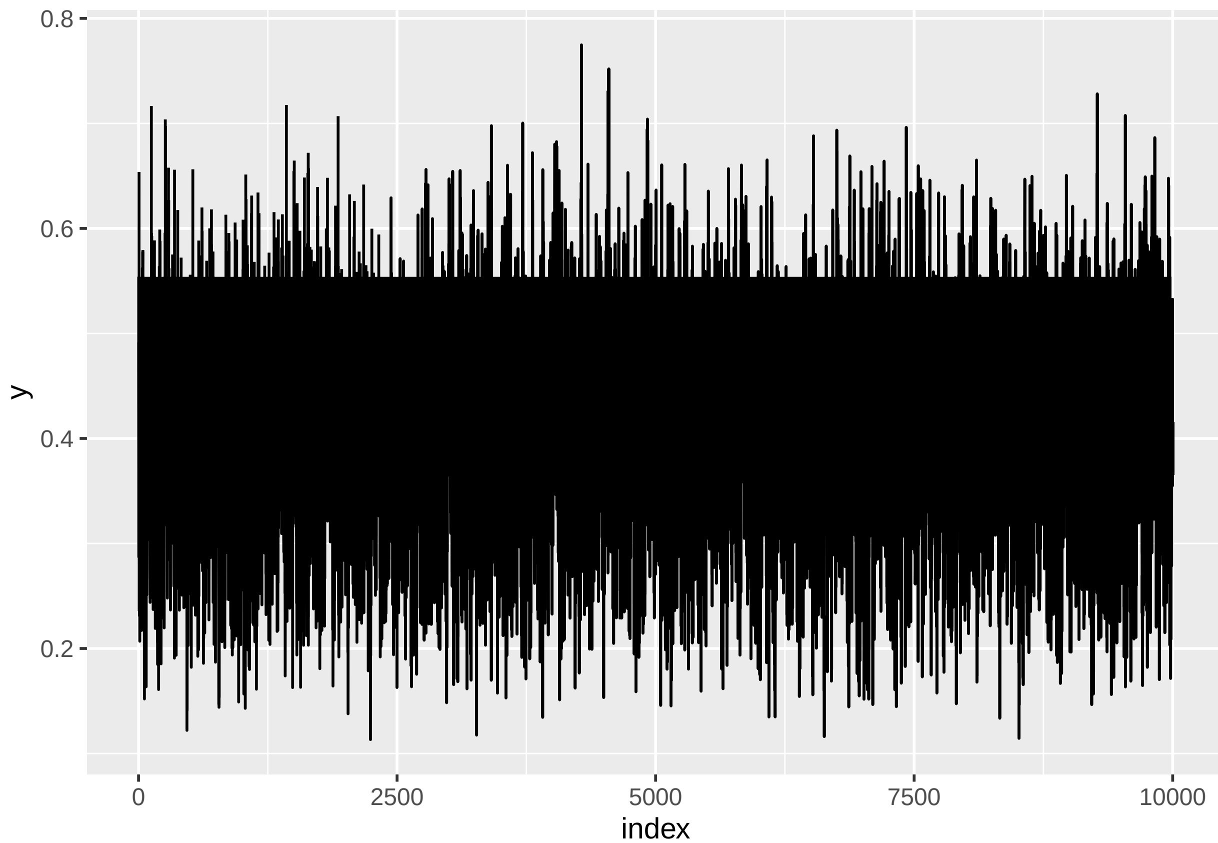
```r
test_gibbs = gibbs(30,a = 9,b=14)

ggplot(as_tibble(test_gibbs))+geom_path(aes(index,x))
```



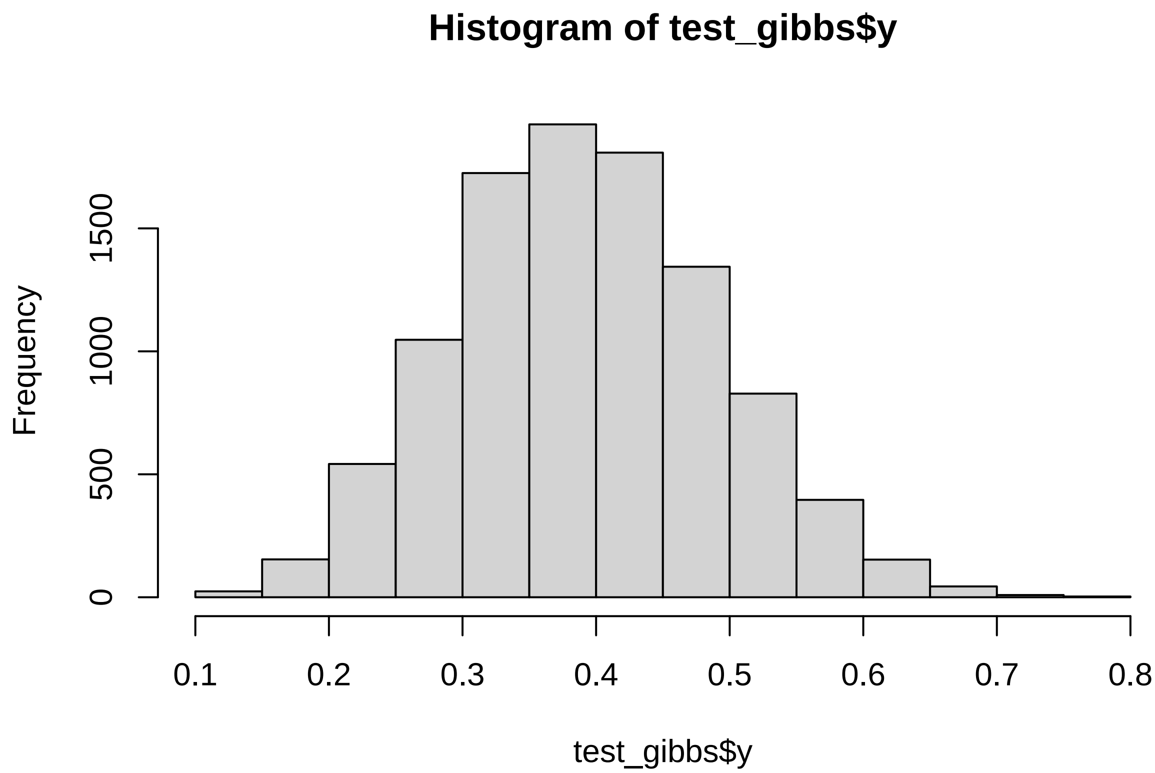

```r
hist(test_gibbs$x)
```



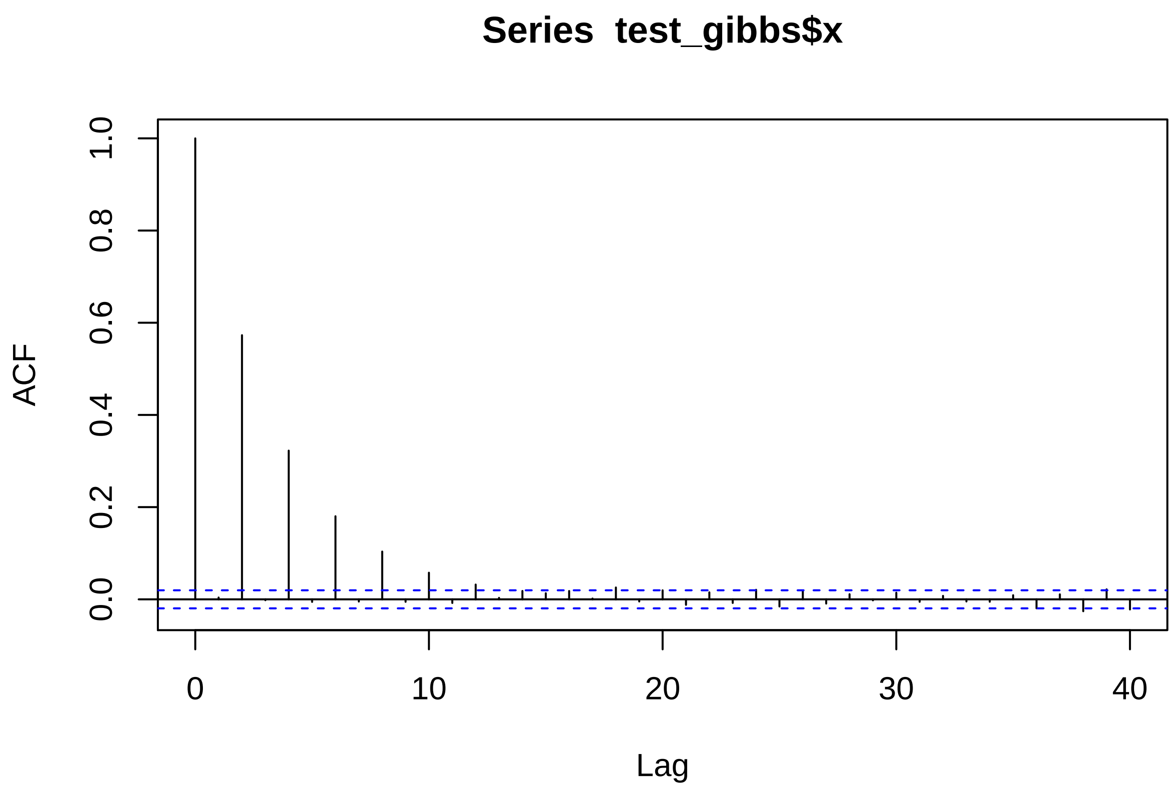

```
ggplot(as_tibble(test_gibbs))+geom_path(aes(index,y))
```



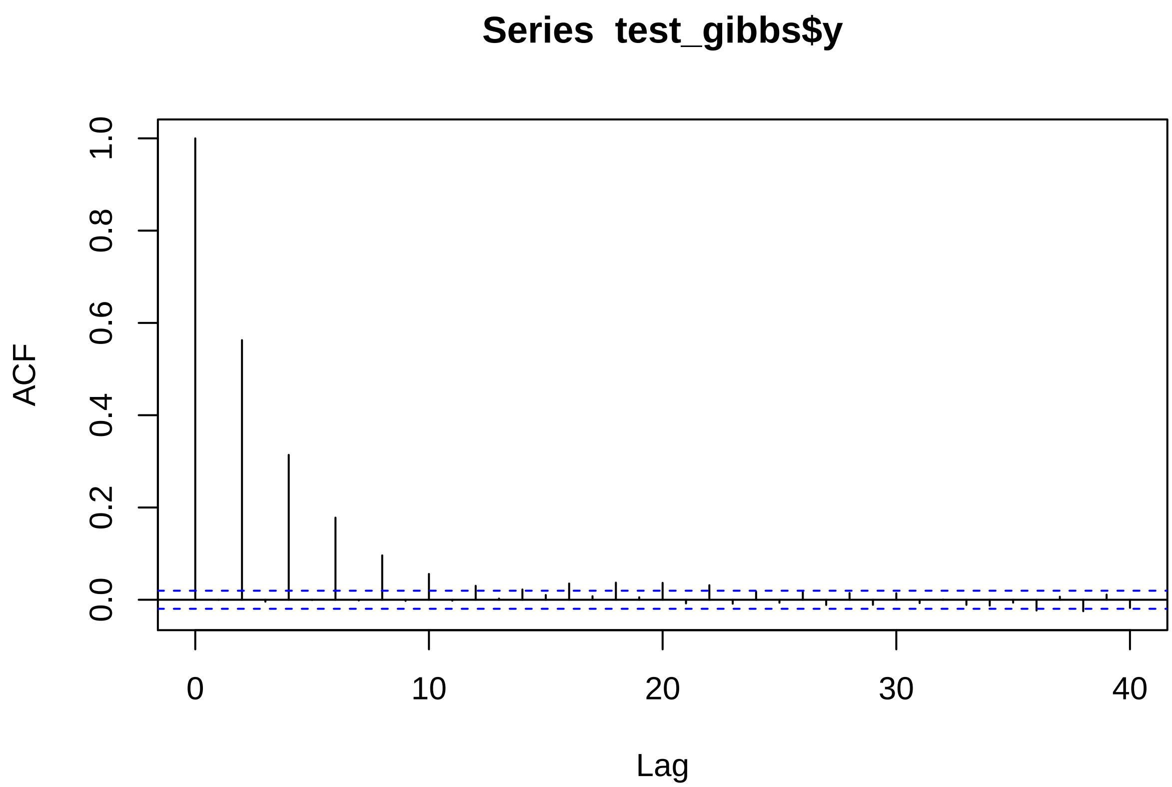

```
hist(test_gibbs$y)
```




```
acf(test_gibbs$x)
```




```
acf(test_gibbs$y)
```



2

We propose two different proposal distribution for x and y,

$$Y_i|X_{i-1}, Y_{i-2} \sim Beta(X_{i-1} + a, n - X_{i-1} + b)$$

and

$$X_i|X_{i-1}, Y_{i-2} \sim Poisson(n * Y_{i-1})$$

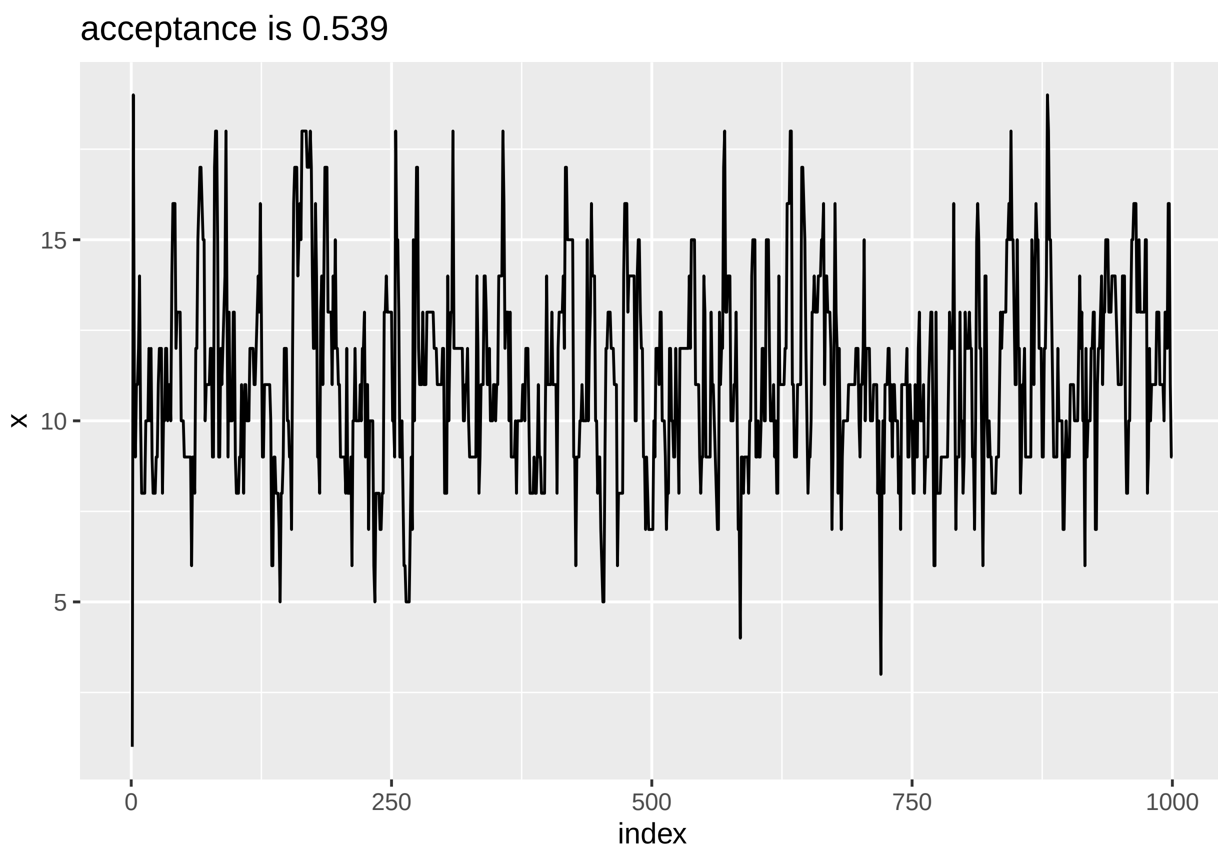
s.t over accept probabilty is:

$$\alpha_i(x_i^k, X_{-i}^k, y_i) = min\{\frac{q(y_i|x_{1,k}, x_{2,k-1})\binom{n}{y_1}y_2^{y_1+a-1}(1-y_2)^{n-y_1+b-1}}{q(x_i|y_{1,k}, y_{2,k-1})\binom{n}{x_1}x_2^{x_1+a-1}(1-x_2)^{n-x_1+b-1}}; 1\})$$

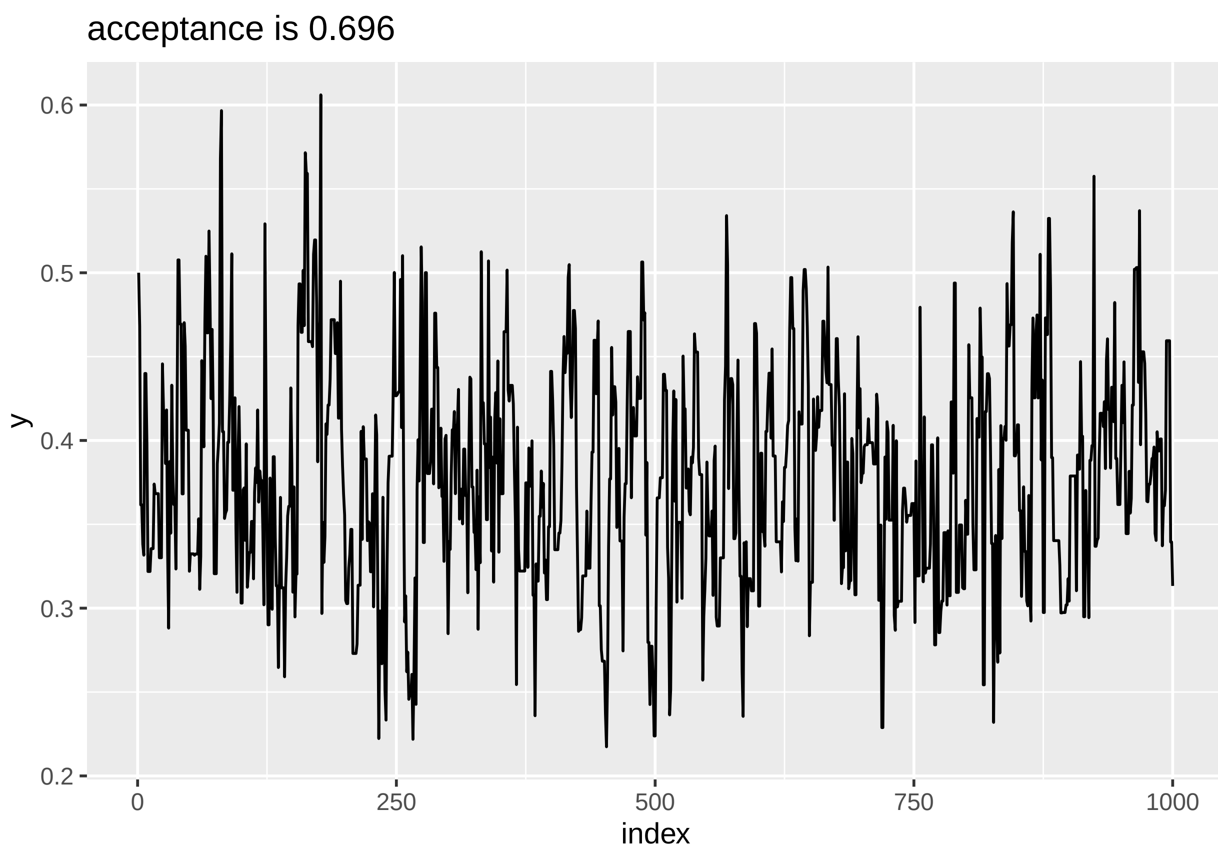
```
logP = function(theta,n,a,b,i,..){
  x = theta[[1]]
  y = theta[[2]]

  res = choose(n, x) *
          y ^ (x + a - 1) *
          (1 - y) ^ (n - x + b - 1)
  if (i == 1)
```

```r
    res = dpois(x, n * y) * res

  if (i == 2)
    res = dbeta(y, x + a, n - x + b) * res

  return(log(res))}

x_update =
  function(x,y,n,a,b,...){
    new_x =
      rpois(1,n*y)
  }

y_update =
  function(x,y,n,a,b,...){
    # make sure that y is in 0,1
    new_y =
      rbeta(1,x+a,n-x+b)
    return(new_y)
  }

M_update =
  function(theta,update_function_list, n, a, b) {
    for (i in 1:length(theta)) {
      # take old parameter
      new = theta

      #update x/y given old
      new[[i]] = update_function_list[[i]](theta[[1]],theta[[2]],n,a,b)

      #calculated the acceptance rate
      accept = logP(new, n, a, b,i) - logP(theta, n, a, b,i)

      if(is.na(accept)) next

      if (log(runif(1)) < accept)
        theta = new
    }

    return(theta)
  }

MET =
  function(n,
           a,
           b,
           step = 1e+4,
           .tol = 1e-6,
           x_init = 1,
           y_init = .5,
           burn = F,
           ...) {
    iter = 1
```

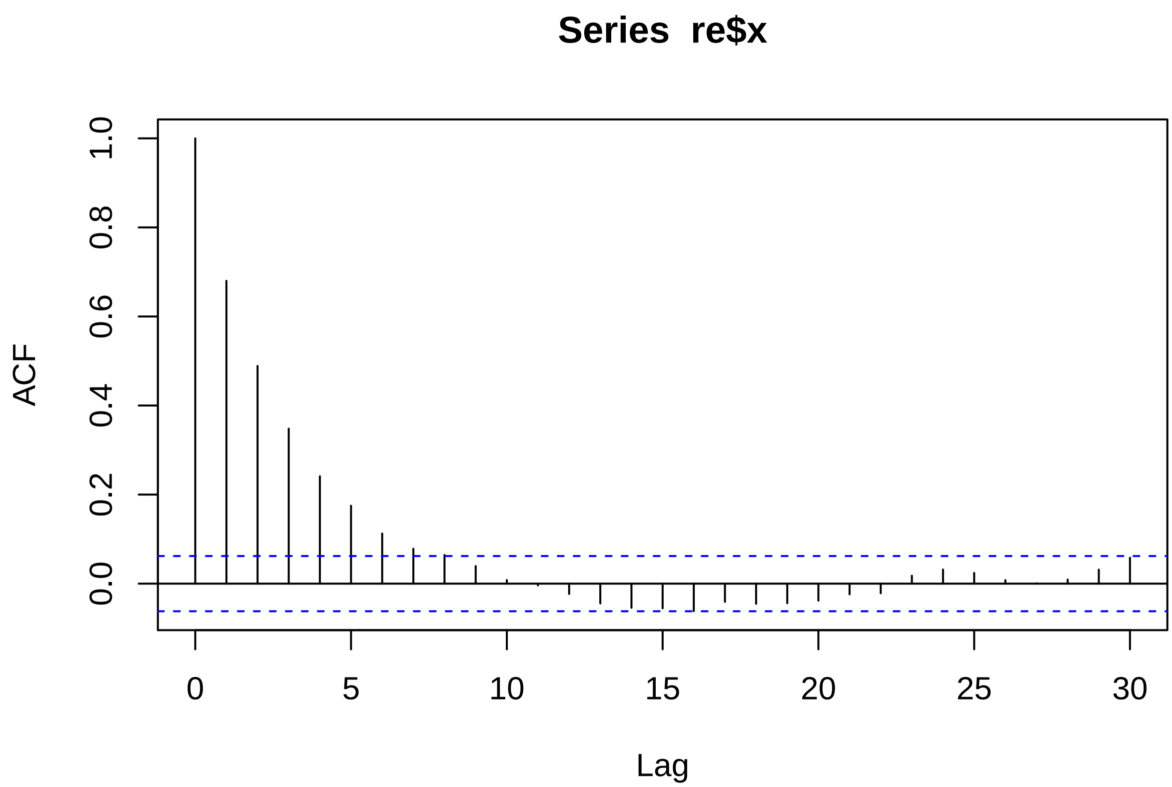
```r
    x = y = xaccept = yaccept = rep(NA, step)

    x[[1]] = x_init

    y[[1]] = y_init


    while (iter < step) {
      new_theta = M_update(c(x[[iter]], y[[iter]]),
                           list(x_update, y_update),
                           n, a, b)
      iter = iter + 1
      x[[iter]] = new_theta[[1]]
      xaccept[[iter]] = x[[iter - 1]] != x[[iter]]
      y[[iter]] = new_theta[[2]]
      yaccept[[iter]] = y[[iter - 1]] != y[[iter]]
    }

    if (burn) {
            x = x[-c(1:burn)]
            y = y[-c(1:burn)]
            xaccept = xaccept[-c(1:burn)]
            yaccept = yaccept[-c(1:burn)]
    }

    accept =
            list(x = xaccept,
                 y = yaccept)

return(list(x = x,
            y = y,
            accept = accept))
  }

re = MET(30,9,14,step = 1000)

ggplot(tibble(index = 1:1000, x = re$x),aes(x =index, y = x))+
  geom_path()+
  labs(title = str_c("acceptance is ",sum(re$accept$x,na.rm = T)/1000))
```



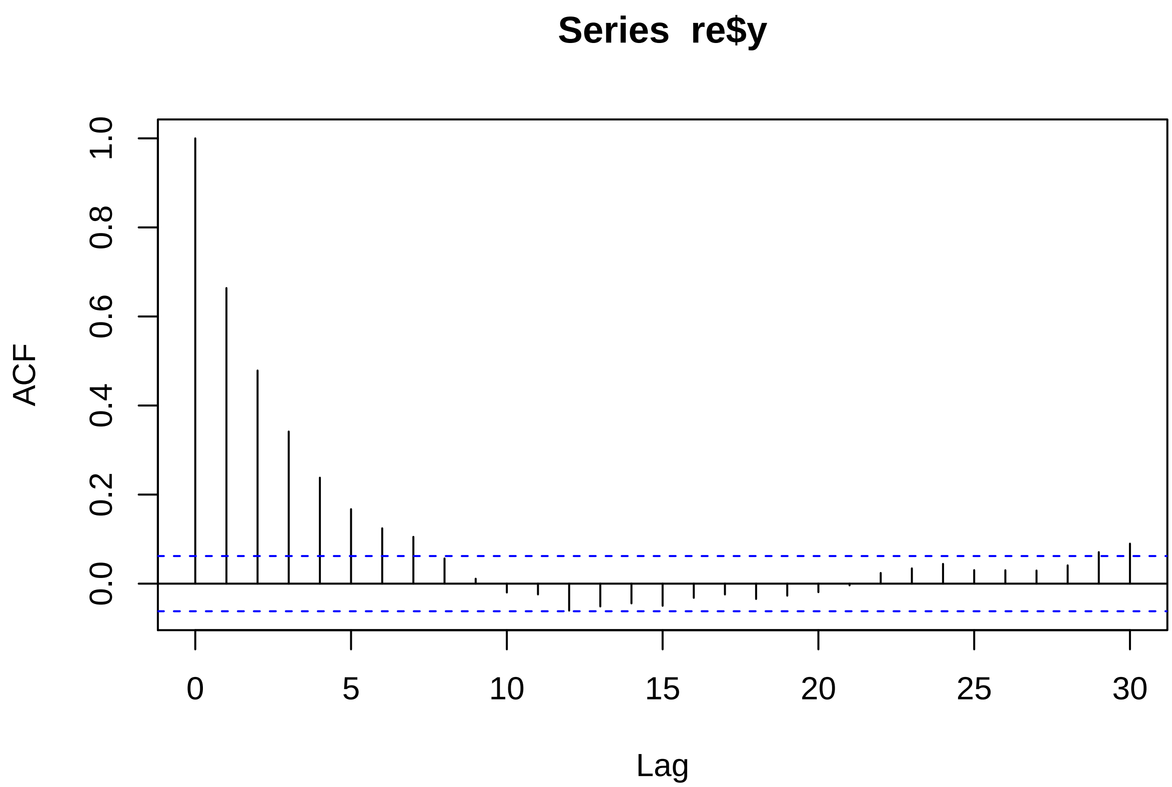

```
ggplot(tibble(index = 1:1000, y = re$y),aes(x =index, y = y))+
  geom_path()+
  labs(title = str_c("acceptance is ",sum(re$accept$y,na.rm = T)/1000))
```




```
acf(re$x)
```




```
acf(re$y)
```




```
# n=30,a = 9, b = 14

#starting value x: 1 to 30
#y 0 to 1

set.seed(123123)
cl = makePSOCKcluster(5)
registerDoParallel(cl)

cond = expand.grid(x_int = seq(1, 30, len = 10) %/% 1,
                   y_int = seq(0, 1, len = 5))

G = foreach(i = 1:nrow(cond),
            .combine = rbind) %dopar% {
              x = cond[i, 1]
              y = cond[i, 2]
              g_mean = list()
              iter = 1
              while(iter<100){
                g = gibbs(
                30,
                9,
                14,
                step = 1000,
                x_init = x,
                y_init = y,
```

```
                  burn = 100
                )
                g_mean[[iter]] = as.numeric(lapply(g, mean)[-3])
                iter = iter + 1
                }
                g_mean = do.call(rbind,g_mean)

                g_mean = colMeans(g_mean)

                g_mean
              }

M = foreach(i = 1:nrow(cond),
              .combine = rbind) %dopar% {
                x = cond[i, 1]
                y = cond[i, 2]
               m_mean = list()
               iter = 1
               while (iter < 100) {
                  m = MET(
                  30,
                  9,
                  14,
                  step = 1000,
                  x_init = x,
                  y_init = y,
                  burn = 100
                )
                m_mean[[iter]] = as.numeric(lapply(m[-3], mean))
                iter = iter + 1
               }
                m_mean = colMeans(do.call(rbind,m_mean))

                m_mean
              }

stopCluster(cl)

sim_data = cbind(cond,G,M)

names(sim_data) = c("x_init","y_init","gibbs_x","gibbs_y","met_x","met_y")

knitr::kable(sim_data,
             caption = "Simulation result based on 100 run on differet start values with 100 burn")
```

Table 1: Simulation result based on 100 run on differet start values with 100 burn

	x_init	y_init	gibbs_x	gibbs_y	met_x	met_y
result.1	1	0.00	11.75278	0.3912468	11.00435	0.3741335
result.2	4	0.00	11.73707	0.3911495	10.96567	0.3735091
result.3	7	0.00	11.70434	0.3908669	10.96640	0.3735068
result.4	10	0.00	11.71430	0.3909699	11.00616	0.3742583

	x_init	y_init	gibbs_x	gibbs_y	met_x	met_y
result.5	13	0.00	11.72996	0.3914803	10.96375	0.3736333
result.6	17	0.00	11.73076	0.3912258	11.01961	0.3746398
result.7	20	0.00	11.72817	0.3907815	11.01811	0.3747866
result.8	23	0.00	11.73038	0.3908368	10.96765	0.3734996
result.9	26	0.00	11.73627	0.3912100	10.97663	0.3736809
result.10	30	0.00	11.73046	0.3910881	10.98553	0.3742803
result.11	1	0.25	11.77440	0.3923670	11.03425	0.3750278
result.12	4	0.25	11.74042	0.3911872	10.99008	0.3740984
result.13	7	0.25	11.69324	0.3903548	10.92217	0.3724546
result.14	10	0.25	11.76320	0.3917337	10.97677	0.3736270
result.15	13	0.25	11.78167	0.3921391	11.01068	0.3747932
result.16	17	0.25	11.74773	0.3915341	10.98079	0.3739023
result.17	20	0.25	11.74691	0.3912330	10.93724	0.3725918
result.18	23	0.25	11.75229	0.3918272	10.96887	0.3735142
result.19	26	0.25	11.74608	0.3912947	10.98838	0.3738287
result.20	30	0.25	11.73672	0.3912832	11.00935	0.3745065
result.21	1	0.50	11.73888	0.3912095	10.99284	0.3740593
result.22	4	0.50	11.77744	0.3922974	10.96369	0.3734679
result.23	7	0.50	11.74196	0.3912345	11.01352	0.3747706
result.24	10	0.50	11.78129	0.3920987	10.95494	0.3732189
result.25	13	0.50	11.75728	0.3916879	10.96400	0.3735654
result.26	17	0.50	11.71871	0.3907759	11.00467	0.3742285
result.27	20	0.50	11.74823	0.3911002	10.98489	0.3742282
result.28	23	0.50	11.69077	0.3903386	10.99401	0.3743501
result.29	26	0.50	11.73715	0.3914853	10.98960	0.3737183
result.30	30	0.50	11.76099	0.3916533	10.96795	0.3733220
result.31	1	0.75	11.74204	0.3917364	11.02438	0.3748286
result.32	4	0.75	11.77217	0.3921399	10.98810	0.3738833
result.33	7	0.75	11.73508	0.3913423	10.96765	0.3732906
result.34	10	0.75	11.74319	0.3915168	10.96979	0.3734650
result.35	13	0.75	11.76405	0.3920141	10.98753	0.3740215
result.36	17	0.75	11.78026	0.3923542	10.97384	0.3735423
result.37	20	0.75	11.72245	0.3907296	10.97520	0.3732499
result.38	23	0.75	11.75045	0.3916638	10.99703	0.3742552
result.39	26	0.75	11.76284	0.3919266	10.97002	0.3733925
result.40	30	0.75	11.72827	0.3907589	10.98532	0.3739931
result.41	1	1.00	11.72455	0.3907915	10.99584	0.3741454
result.42	4	1.00	11.73093	0.3912295	10.98956	0.3739961
result.43	7	1.00	11.73664	0.3911985	10.97683	0.3738307
result.44	10	1.00	11.73808	0.3911339	10.98978	0.3740832
result.45	13	1.00	11.75283	0.3917209	10.99071	0.3741415
result.46	17	1.00	11.71972	0.3908877	10.98844	0.3740322
result.47	20	1.00	11.75376	0.3914740	10.98433	0.3737887
result.48	23	1.00	11.74805	0.3916388	10.95542	0.3732552
result.49	26	1.00	11.76008	0.3918936	10.97221	0.3738985
result.50	30	1.00	11.71669	0.3910038	10.99131	0.3741062

A 100 run simulation is conducted,each senario start from a different starting value of x and y, which include extreme and moderated cases. In the simulation, we can see that regardless of starting values, both methods reach similar conclusion about the posterior value of x and y.