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
```

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

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

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

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

```
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

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

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...

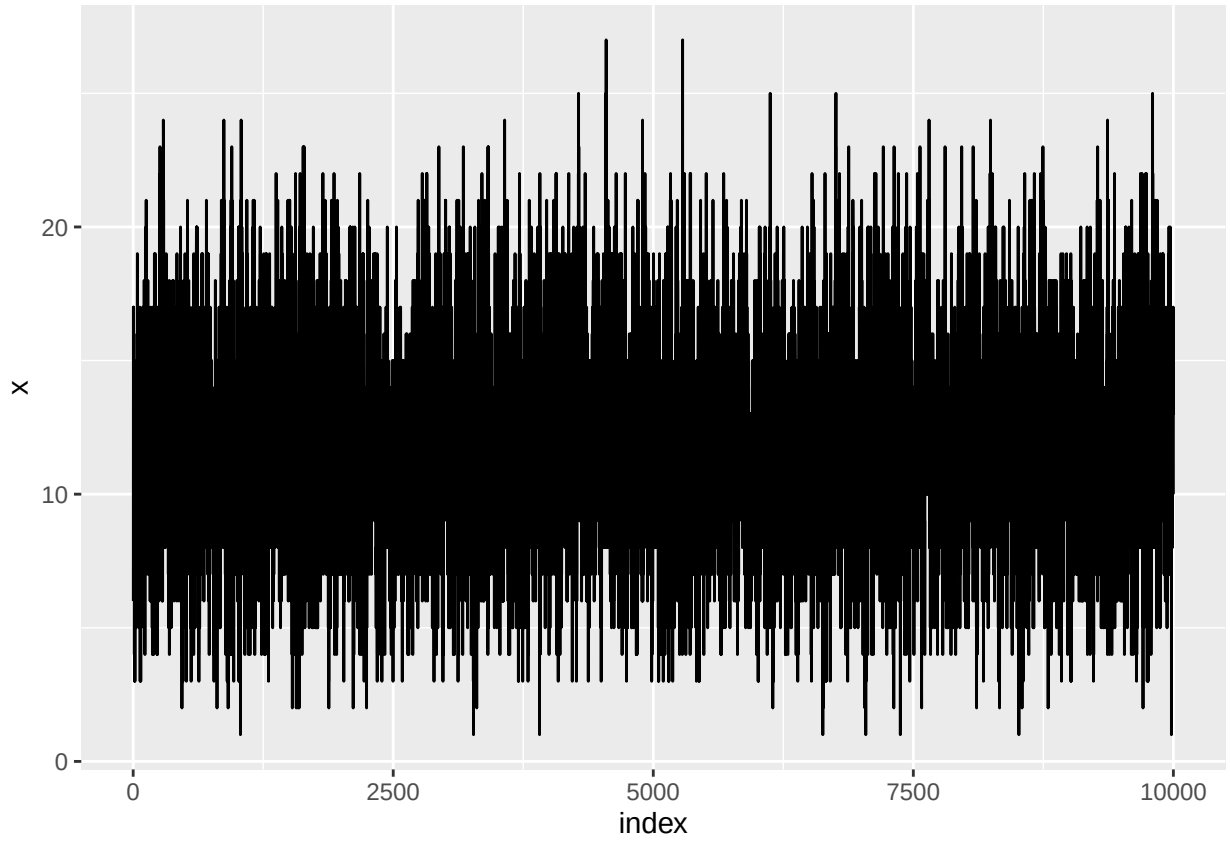
1

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

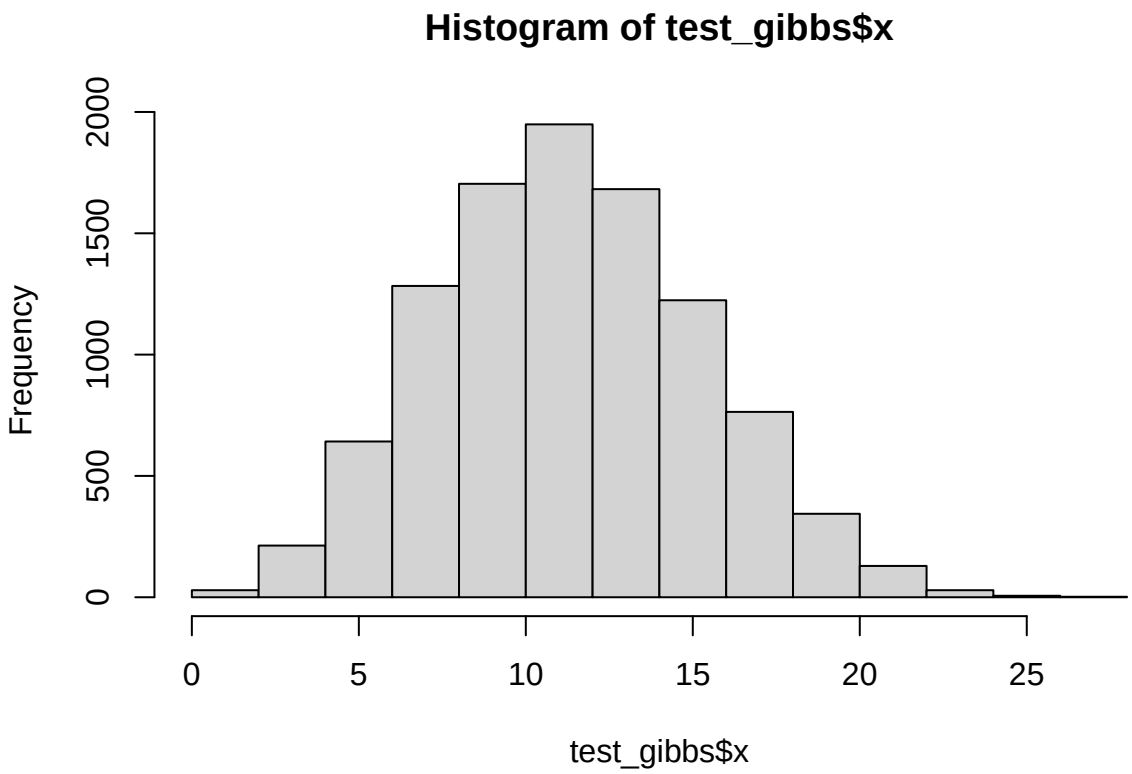
$$
\begin{aligned}
f(y|x) &= \frac{f(x,y)}{f(x)} && (12)\\
&= f(x,y)/\binom{n}{x}\int_y y^{x+a-1}(1-y)^{n-x+b-1} && (13)\\
&= f(x,y)/\binom{n}{x}B(x+a,n-x+b) && (14)\\
&= \frac{\binom{n}{x}y^{x+a-1}(1-y)^{n-x+b-1}}{\binom{n}{x}B(x+a,n-x+b)} && (15)\\
&= Beta(x+a,n-x+b) && (16)
\end{aligned}
$$

```
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)
```

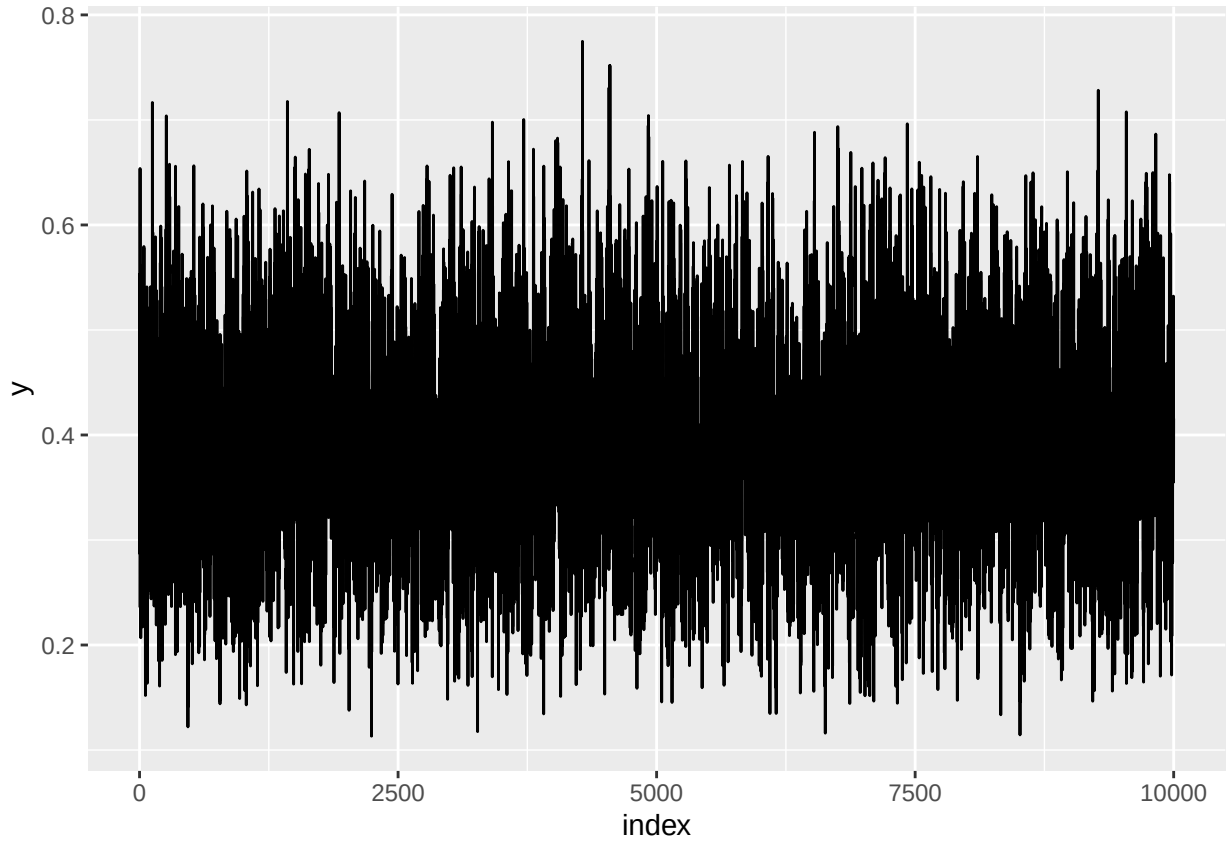
```
test_gibbs = gibbs(30,a = 9,b=14)

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



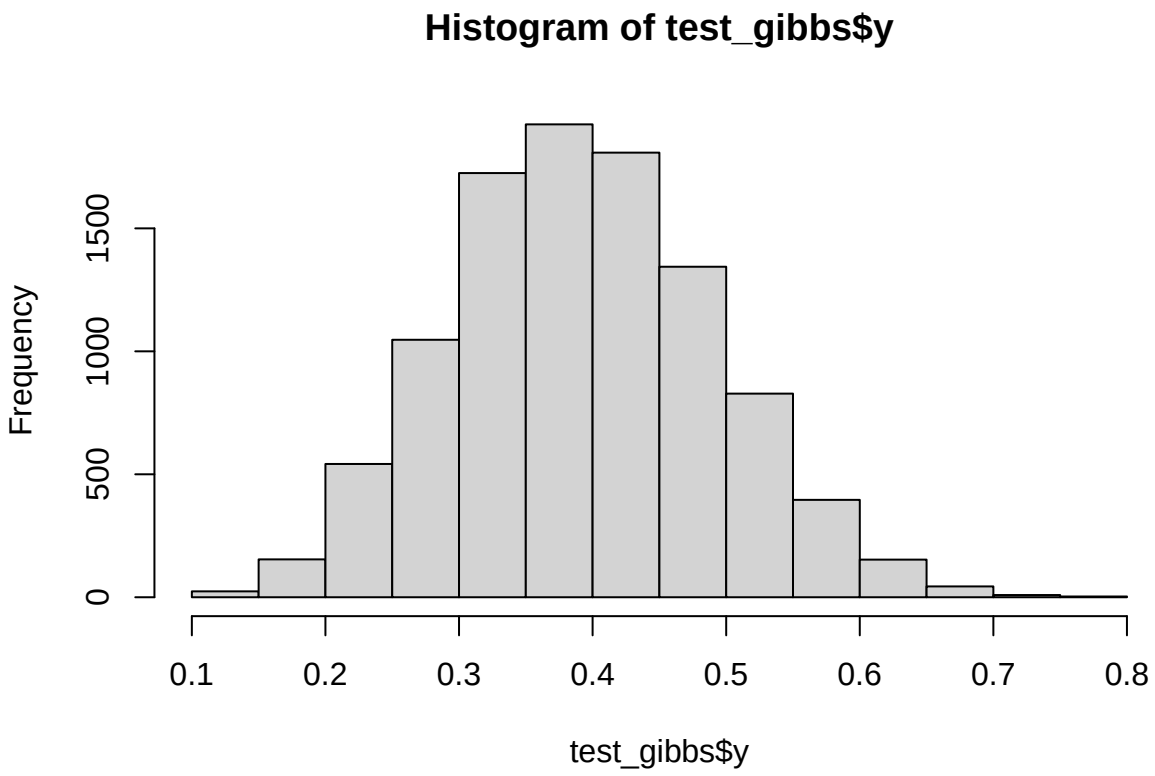

```
hist(test_gibbs$x)
```




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




```
hist(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

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

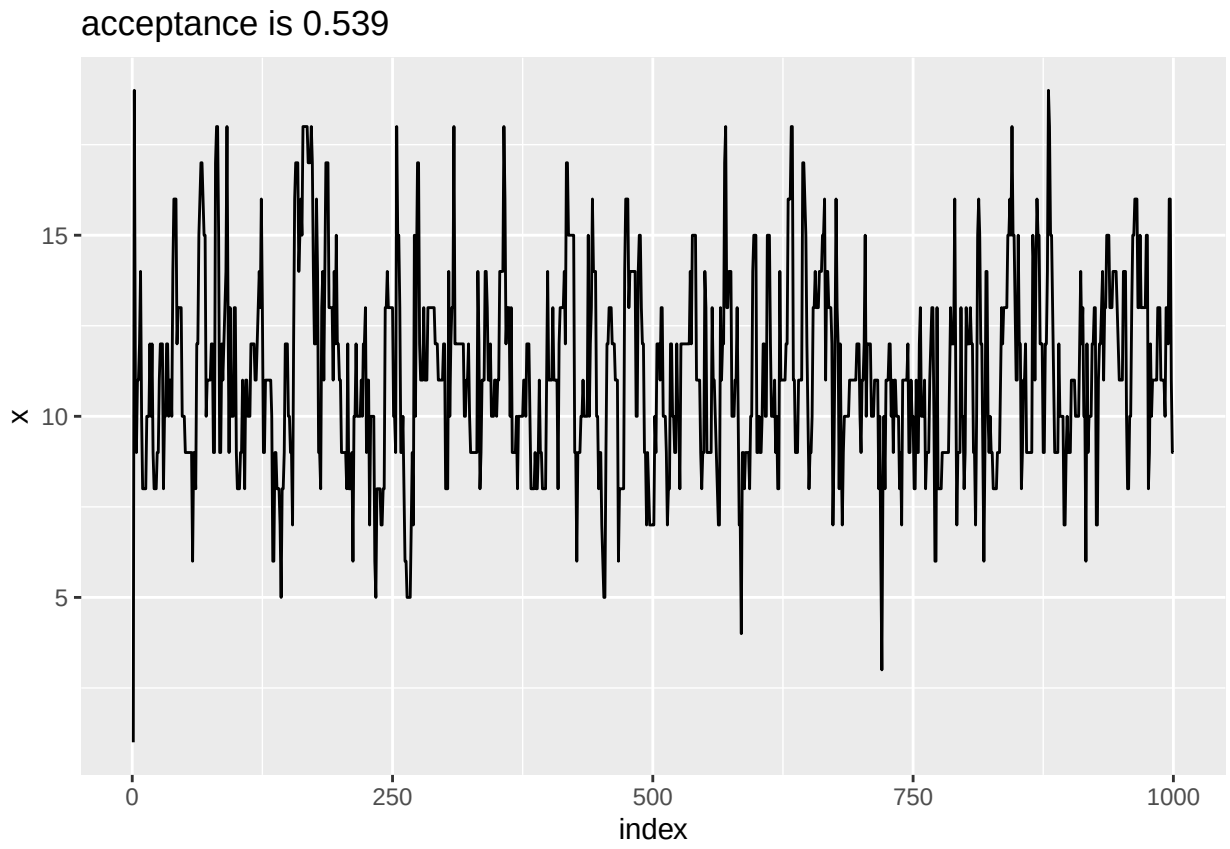
s.t over accept probabilty is:

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

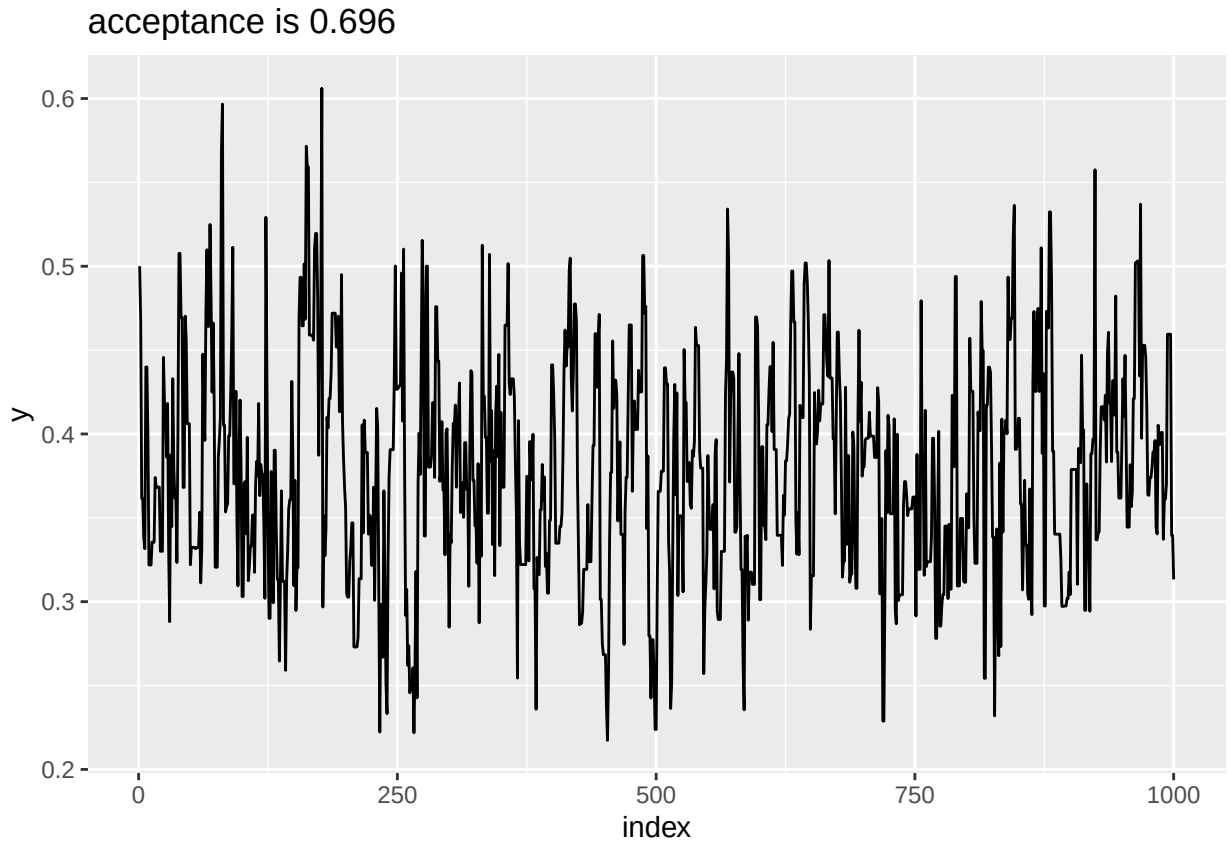
```
logP = function(theta,n,a,b,i,..){
  x = theta[[1]]
  y = theta[[2]]

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

```
    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
```

```
    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))
```




```
# 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	11.72690	0.3907731	11.00125	0.3744719
result.2	4	0.00	11.69791	0.3902345	10.99679	0.3741472
result.3	7	0.00	11.75605	0.3919235	10.97925	0.3736102
result.4	10	0.00	11.73620	0.3913460	10.98660	0.3737238

	x_init	y_init	gibbs_x	gibbs_y	met_x	met_y
result.5	13	0.00	11.71000	0.3901154	10.99779	0.3742624
result.6	17	0.00	11.73902	0.3914173	10.99901	0.3743032
result.7	20	0.00	11.76862	0.3922908	11.01499	0.3744462
result.8	23	0.00	11.72347	0.3908873	10.97490	0.3735634
result.9	26	0.00	11.74396	0.3911396	10.96237	0.3733640
result.10	30	0.00	11.73007	0.3912455	10.98927	0.3737016
result.11	1	0.25	11.71832	0.3909942	10.95942	0.3731909
result.12	4	0.25	11.72137	0.3907566	10.97450	0.3734152
result.13	7	0.25	11.73848	0.3914636	10.97595	0.3736516
result.14	10	0.25	11.73836	0.3910511	10.97679	0.3736718
result.15	13	0.25	11.70855	0.3906583	10.99972	0.3740242
result.16	17	0.25	11.71964	0.3908408	10.94451	0.3729933
result.17	20	0.25	11.72921	0.3910008	10.98449	0.3739219
result.18	23	0.25	11.70971	0.3905833	10.96617	0.3732538
result.19	26	0.25	11.78207	0.3924706	10.97621	0.3737348
result.20	30	0.25	11.75304	0.3917886	10.99811	0.3744136
result.21	1	0.50	11.72205	0.3911910	10.98465	0.3738420
result.22	4	0.50	11.75686	0.3919630	10.96934	0.3732690
result.23	7	0.50	11.74319	0.3913857	10.96667	0.3735604
result.24	10	0.50	11.68741	0.3899074	10.99089	0.3740185
result.25	13	0.50	11.76852	0.3918895	11.01073	0.3744354
result.26	17	0.50	11.67990	0.3898952	10.98771	0.3739288
result.27	20	0.50	11.76976	0.3919899	10.97808	0.3739060
result.28	23	0.50	11.79075	0.3923527	10.99419	0.3743380
result.29	26	0.50	11.76416	0.3916221	11.00125	0.3743189
result.30	30	0.50	11.72963	0.3914002	11.00430	0.3744901
result.31	1	0.75	11.74795	0.3912980	10.98245	0.3738517
result.32	4	0.75	11.74139	0.3915266	10.98972	0.3742626
result.33	7	0.75	11.76164	0.3920102	11.01139	0.3745841
result.34	10	0.75	11.74058	0.3911289	10.97352	0.3737834
result.35	13	0.75	11.74242	0.3911800	10.99190	0.3740506
result.36	17	0.75	11.74746	0.3917311	10.99631	0.3744064
result.37	20	0.75	11.75844	0.3919359	10.95896	0.3731657
result.38	23	0.75	11.73200	0.3911269	10.98831	0.3740630
result.39	26	0.75	11.73667	0.3913982	10.96873	0.3736891
result.40	30	0.75	11.76437	0.3919231	11.00696	0.3744667
result.41	1	1.00	11.76182	0.3918310	11.01969	0.3747083
result.42	4	1.00	11.76764	0.3918322	10.96896	0.3733750
result.43	7	1.00	11.74723	0.3914430	11.00751	0.3744965
result.44	10	1.00	11.71801	0.3908551	10.97183	0.3735912
result.45	13	1.00	11.76648	0.3919556	10.97651	0.3736094
result.46	17	1.00	11.72079	0.3908125	10.96582	0.3731609
result.47	20	1.00	11.72933	0.3911060	10.98947	0.3737554
result.48	23	1.00	11.72657	0.3908755	10.97930	0.3736588
result.49	26	1.00	11.67636	0.3896876	10.98705	0.3740366
result.50	30	1.00	11.72348	0.3910136	10.97207	0.3732992

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.