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 $\mathbf{X}$:

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

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

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...

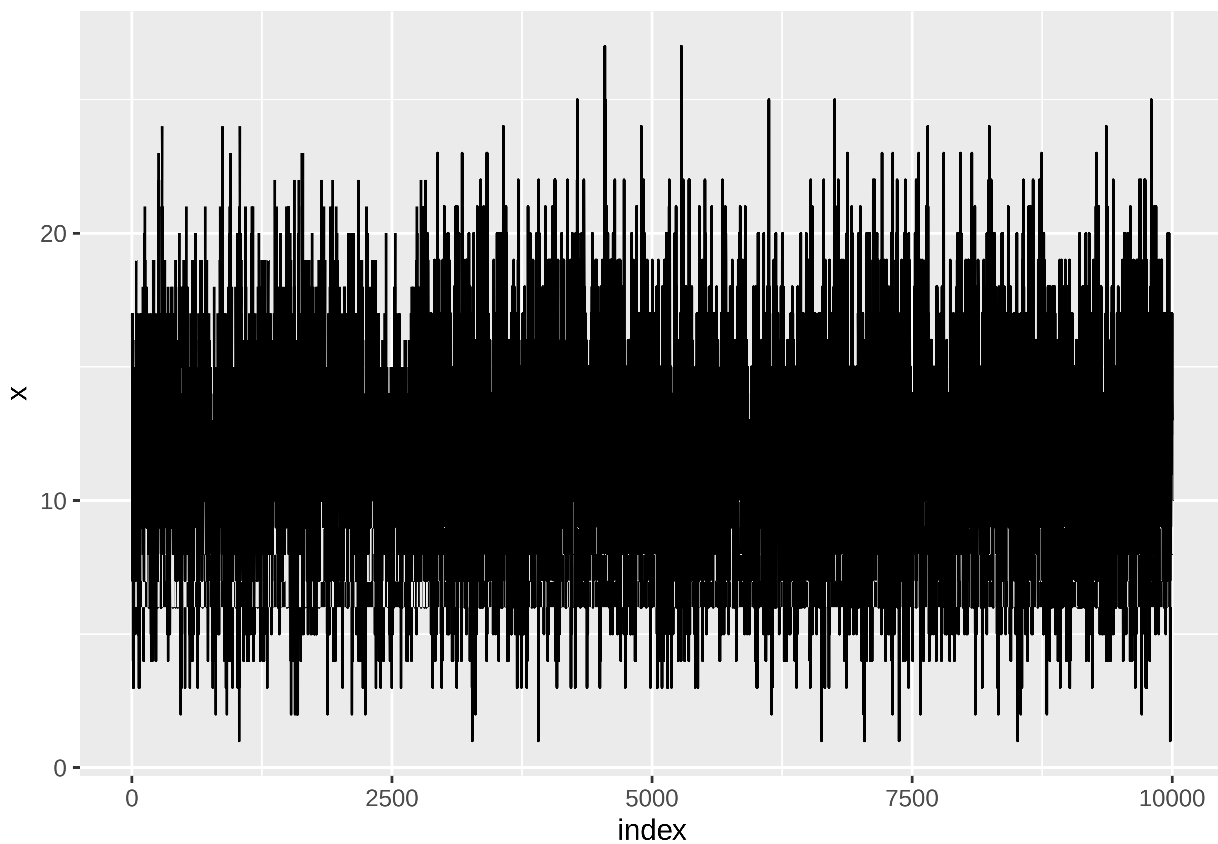
1

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

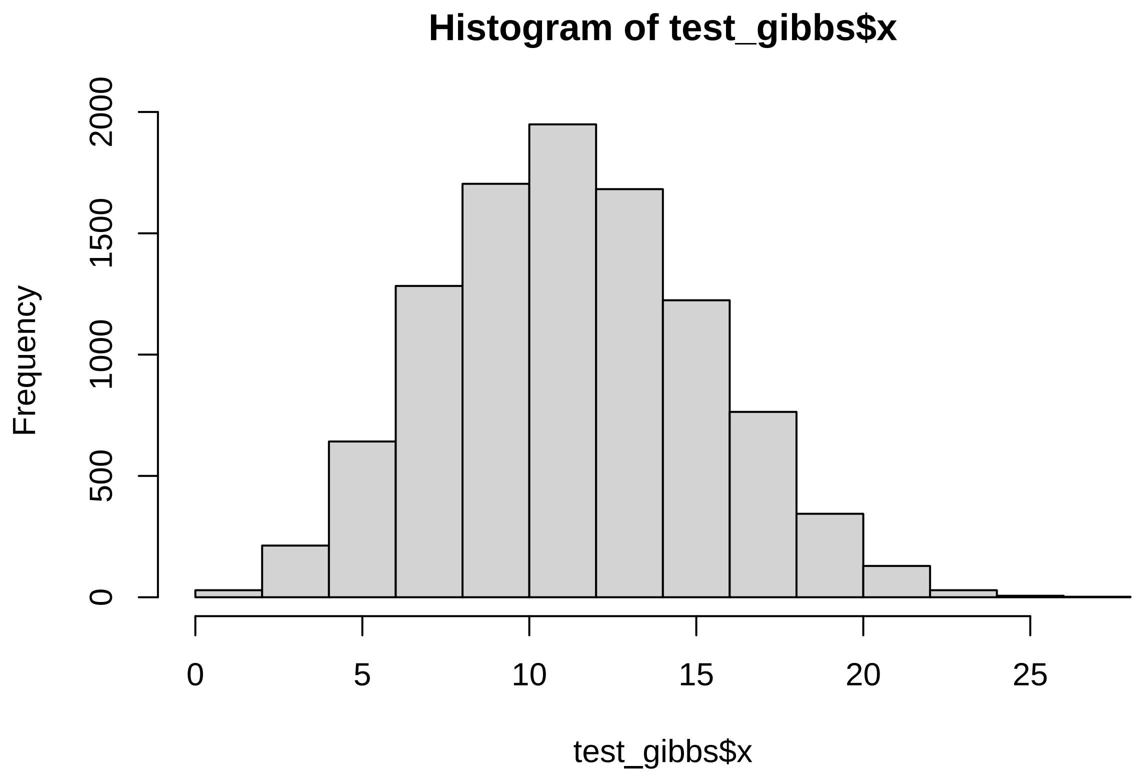
$$
\begin{aligned}
f(y|x) &= \frac{f(x,y)}{f(x)} && (12)\\
&= f(x,y)/\binom{n}{x}\int_y y^{x+a-1}(1-y)^{n-x+b-1} && (13)\\
&= f(x,y)/\binom{n}{x}B(x+a,n-x+b) && (14)\\
&= \frac{\binom{n}{x}y^{x+a-1}(1-y)^{n-x+b-1}}{\binom{n}{x}B(x+a,n-x+b)} && (15)\\
&= Beta(x+a,n-x+b) && (16)
\end{aligned}
$$

```
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)
```

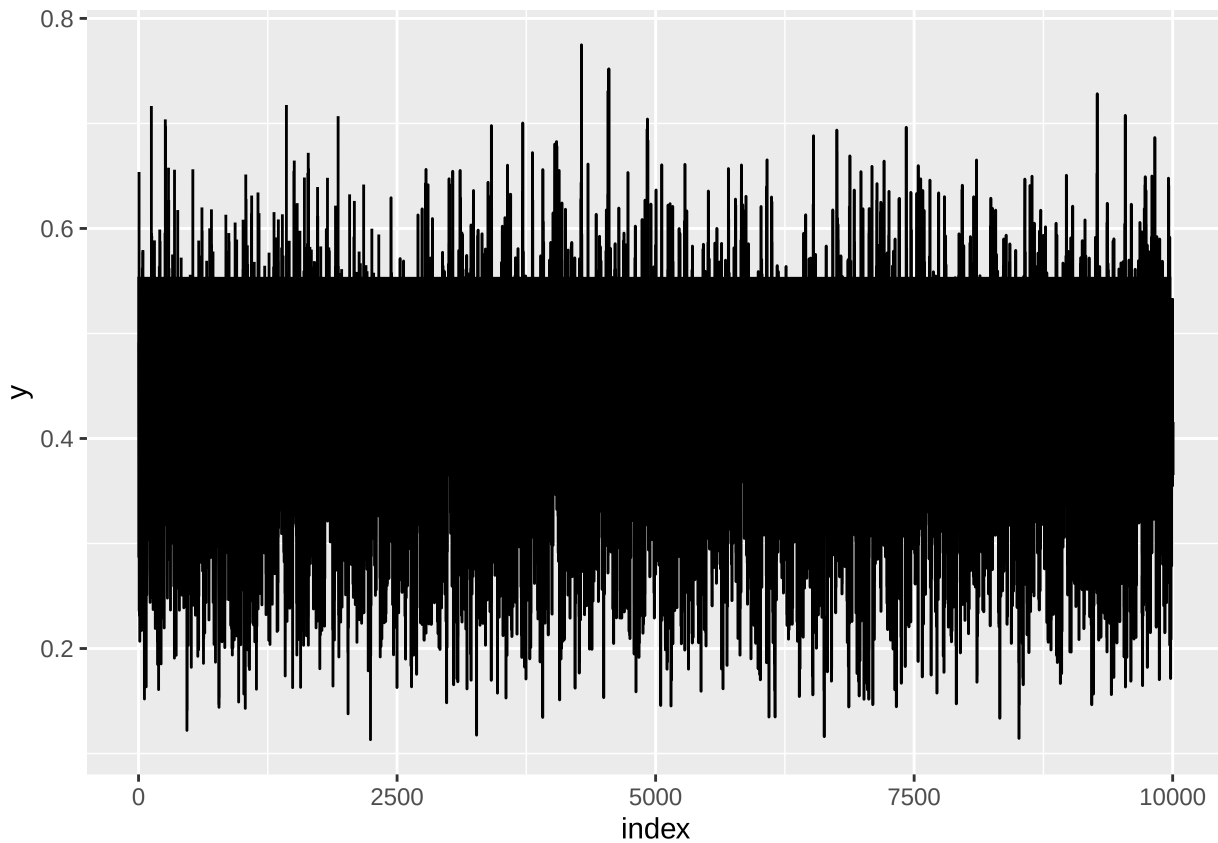
```
test_gibbs = gibbs(30,a = 9,b=14)

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



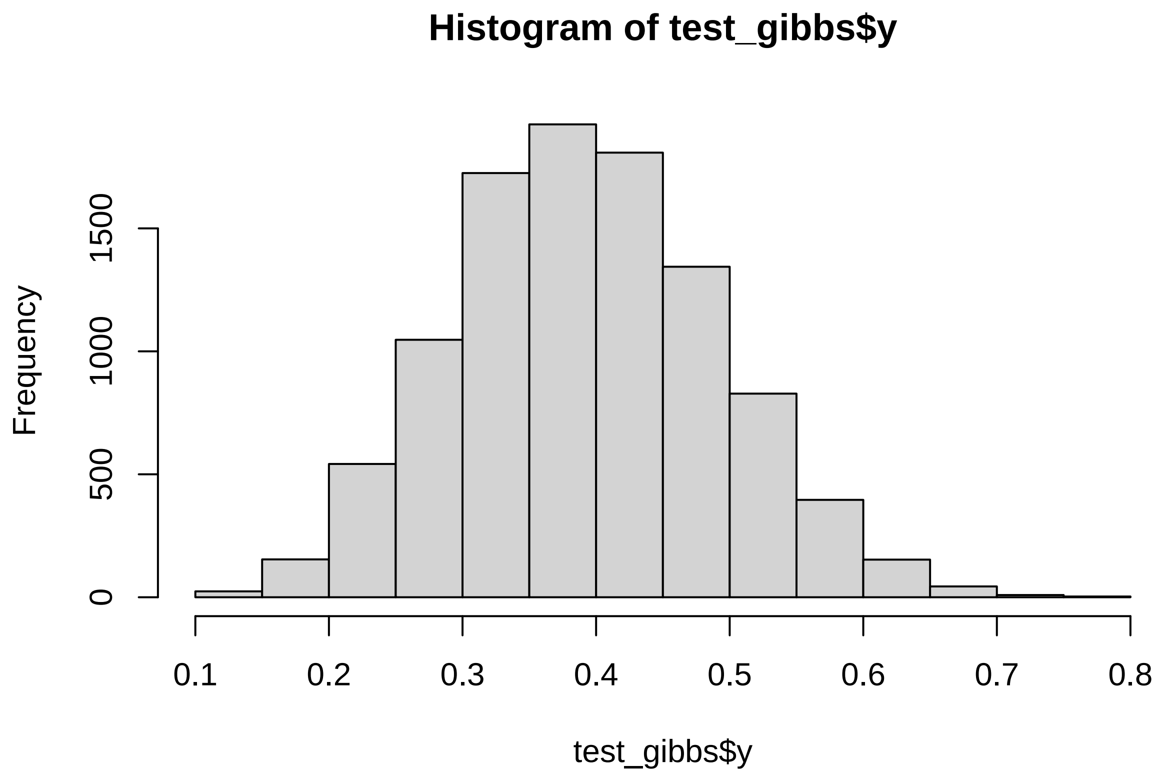

```
hist(test_gibbs$x)
```




```
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,

$$Y|X \sim Beta(X + a, n - X + b)$$

and

$$X|Y \sim Poisson(n * Y)$$

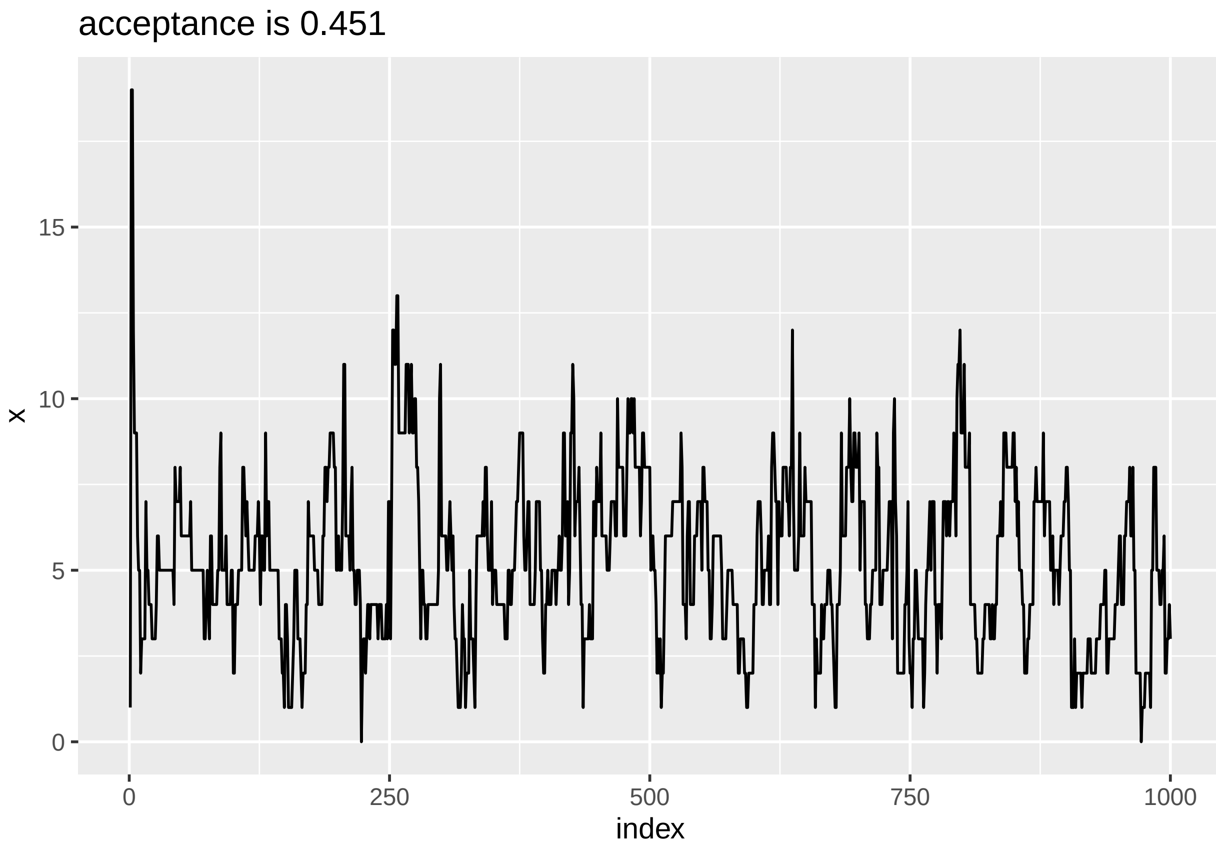
s.t over accept probabilty is:

$$\alpha_i(x_i^k, X_{-i}^k, y_i) = min\{\frac{\pi(poison(y_1; n * y_2)) * \pi(beta(y_2; y_1 + a, n - y_1 + b))\binom{n}{y_1} y_2^{y_1+a-1}(1 - y_2)^{n-y_1+b-1}}{\pi(poison(x_1; n * x_2)) * \pi(beta(x_2; x_1 + a, n - x_1 + b))\binom{n}{x_1} x_2^{x_1+a-1}(1 - x_2)^{n-x_1+b-1}}; 1\})$$

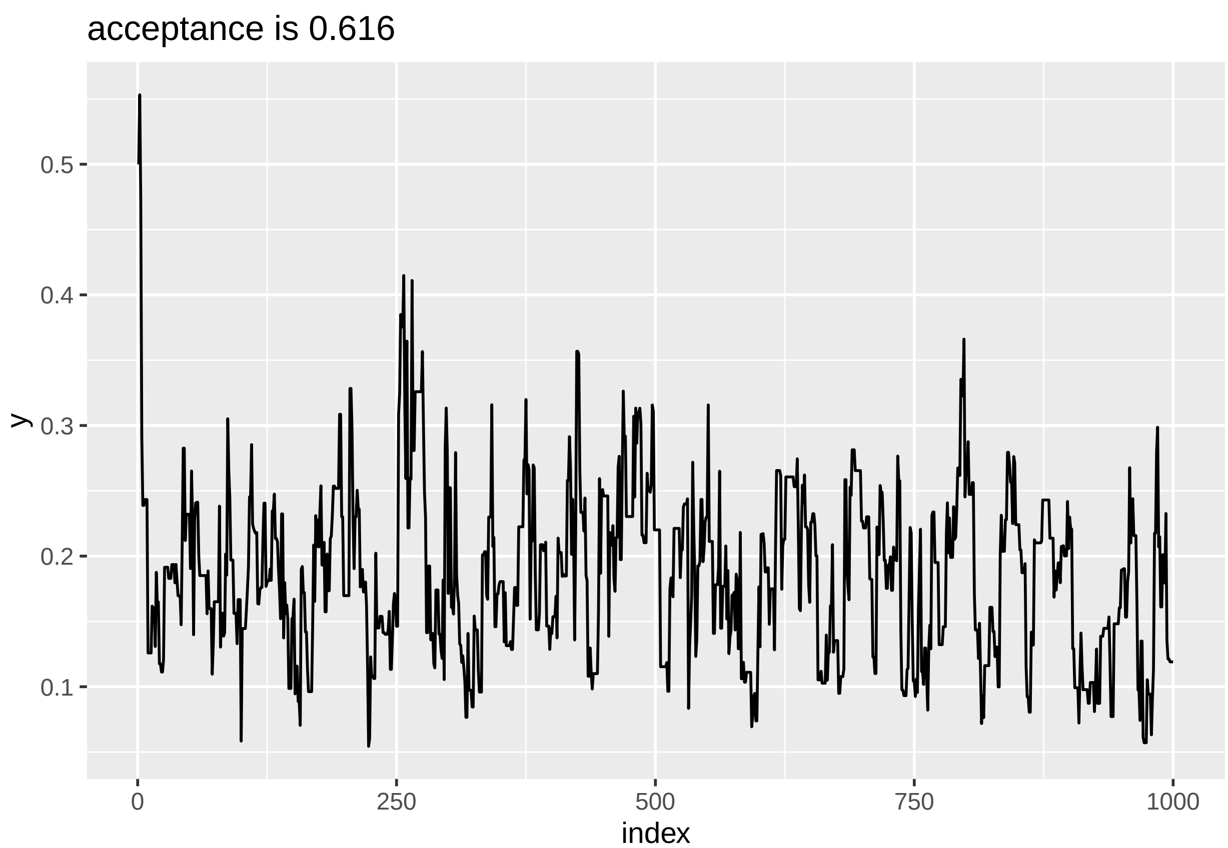
```
logP = function(theta,n,a,b){
  x = theta[[1]]
  y = theta[[2]]

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

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

  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) - logP(theta, n, a, b)

      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(30,4,13,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))
```




```r
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	11.75774	0.3916770	10.85275	0.3699938
result.2	4	0.00	11.77409	0.3919955	10.81447	0.3693705
result.3	7	0.00	11.74642	0.3911175	10.83548	0.3696568
result.4	10	0.00	11.74879	0.3915232	10.84470	0.3698764

	x_init	y_init	gibbs_x	gibbs_y	met_x	met_y
result.5	13	0.00	11.76281	0.3918796	10.86222	0.3704456
result.6	17	0.00	11.70654	0.3905943	10.90585	0.3714791
result.7	20	0.00	11.73804	0.3912717	10.85393	0.3702544
result.8	23	0.00	11.73024	0.3913210	10.81612	0.3691960
result.9	26	0.00	11.72606	0.3908191	10.83316	0.3693884
result.10	30	0.00	11.74091	0.3914495	10.90416	0.3714803
result.11	1	0.25	11.74799	0.3916115	10.90450	0.3711891
result.12	4	0.25	11.76541	0.3920141	10.83089	0.3694642
result.13	7	0.25	11.76462	0.3918480	10.88010	0.3706515
result.14	10	0.25	11.76230	0.3917264	10.88891	0.3709120
result.15	13	0.25	11.75420	0.3916707	10.87214	0.3706449
result.16	17	0.25	11.72446	0.3911818	10.87412	0.3708563
result.17	20	0.25	11.73505	0.3913063	10.85992	0.3701137
result.18	23	0.25	11.74102	0.3914666	10.83250	0.3695225
result.19	26	0.25	11.74734	0.3915290	10.89623	0.3710718
result.20	30	0.25	11.69695	0.3901061	10.85368	0.3703280
result.21	1	0.50	11.73187	0.3912056	10.87452	0.3707199
result.22	4	0.50	11.75145	0.3917772	10.87789	0.3705943
result.23	7	0.50	11.74212	0.3915207	10.91351	0.3716661
result.24	10	0.50	11.75070	0.3914726	10.84838	0.3700277
result.25	13	0.50	11.69587	0.3899829	10.88308	0.3706693
result.26	17	0.50	11.70941	0.3908483	10.88194	0.3708429
result.27	20	0.50	11.74749	0.3912661	10.87836	0.3706922
result.28	23	0.50	11.75321	0.3916275	10.84797	0.3697458
result.29	26	0.50	11.76224	0.3916342	10.89055	0.3709482
result.30	30	0.50	11.75042	0.3917670	10.84395	0.3700564
result.31	1	0.75	11.72424	0.3909153	10.87835	0.3706304
result.32	4	0.75	11.73481	0.3913404	10.89532	0.3712639
result.33	7	0.75	11.73672	0.3912441	10.82105	0.3694075
result.34	10	0.75	11.70386	0.3901614	10.82727	0.3692497
result.35	13	0.75	11.75744	0.3919755	10.83765	0.3697117
result.36	17	0.75	11.68813	0.3901043	10.87084	0.3705341
result.37	20	0.75	11.76489	0.3918498	10.83000	0.3698522
result.38	23	0.75	11.70970	0.3904851	10.90926	0.3714294
result.39	26	0.75	11.71670	0.3907259	10.86979	0.3706005
result.40	30	0.75	11.77095	0.3922068	10.88156	0.3709885
result.41	1	1.00	11.78730	0.3923384	10.89558	0.3710448
result.42	4	1.00	11.73565	0.3909728	10.83072	0.3695123
result.43	7	1.00	11.74626	0.3915734	10.86505	0.3703706
result.44	10	1.00	11.73501	0.3913039	10.91148	0.3716233
result.45	13	1.00	11.73927	0.3912378	10.83033	0.3692520
result.46	17	1.00	11.75493	0.3919707	10.84349	0.3696923
result.47	20	1.00	11.75155	0.3915769	10.90294	0.3713121
result.48	23	1.00	11.72269	0.3910134	10.82492	0.3694473
result.49	26	1.00	11.70538	0.3906602	10.92288	0.3718473
result.50	30	1.00	11.71487	0.3907421	10.88274	0.3706612