

Final Project Part 1

Feng Wang

October 25, 2016

Metropolis-Hastings

1. Algorithm and main function

Algorithm

- Randomly generate the start value from a uniform distribution
- Built the proposalfunction given in the problem, and generate a possible sample candidate
- Adjust the acceptance probability and apply into comparision
- For the non symmetric distribution, we have $r = \frac{P(\theta')/J(\theta')}{P(\theta)/J(\theta)}$
- we compute the posterior and proposal probability, Posterior = $\frac{L(\phi')}{L(\phi)}$, Proposal = $\frac{L'(\phi)}{L'(\phi')}$
- We change the equation and multiple these two number and compare to a random number from a uniform distribution
- If it's larger add into the chain, otherwise use original value, and do the iterations

Main function

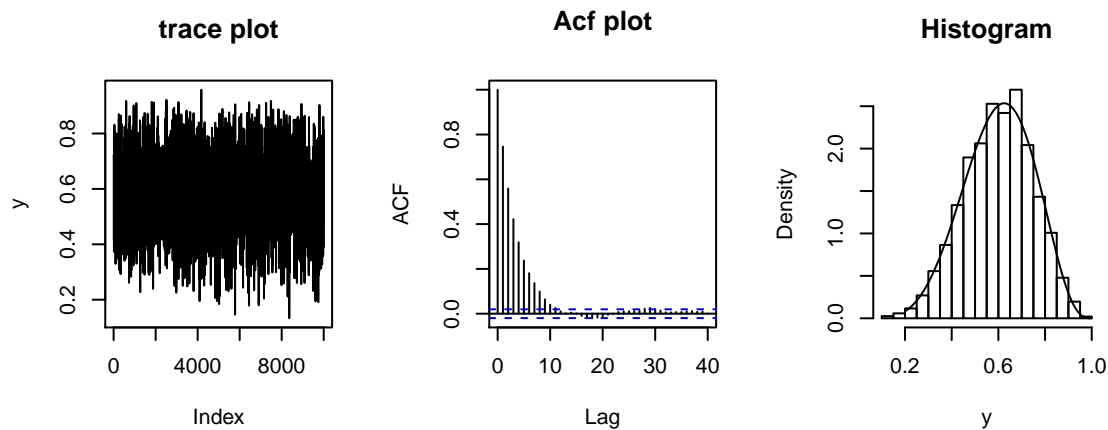
```
Beta_MetropolisHastings = function(true_alpha, true_beta, c, iter){  
  set.seed(1)  
  # start value random from a uniform distribution  
  start_value = runif(1); chain = rep(0, iter)  
  # the proposal function given in the problem  
  proposalfunction = function(c, phi_old){return(rbeta(1, c * phi_old, c * (1 - phi_old)))}  
  for (i in 1 : iter){  
    # generate a bata sample  
    beta = proposalfunction(c, start_value)  
    # compute the posterior and propsal value from generated beta  
    post = dbeta(beta, true_alpha, true_beta) / dbeta(start_value, true_alpha, true_beta)  
    proposal = dbeta(start_value, c * beta, c * (1 - beta)) / dbeta(beta, c * start_value,  
                                                                    c * (1 - start_value))  
  
    # if it's in the accept field, add beta into chain, otherwise,  
    # use the same value with last element  
    if(runif(1) < min(1, post * proposal)){  
      start_value = beta  
    }  
    chain[i] = start_value  
  }  
  return(chain)  
}
```

2. Evaluate the performance of the sampler

Variables:

$\alpha = 6$, $\beta = 4$, $c = 1$, iteration = 10000

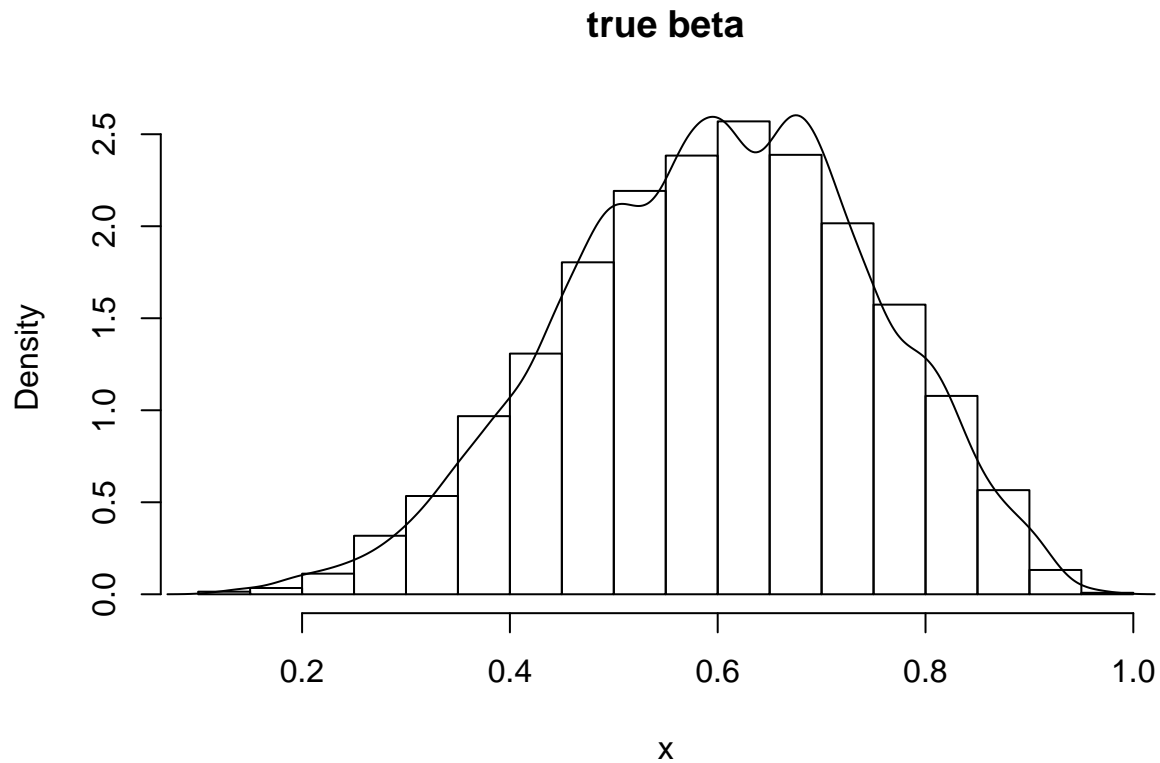
```
set.seed(1)
true_alpha = 6; true_beta = 4; c = 1; iter = 10000
chain = Beta_MetropolisHastings(true_alpha = 6, true_beta = 4, c = 1, iter = 10000)
par(mfrow = c(1, 3)) #1 row, 3 columns
plot(chain, type = "l", main = "trace plot", ylab = "y")
acf(chain, main = "Acf plot")
hist(chain, freq = FALSE, main = "Histogram", xlab = "y")
grid = seq(0.2, 1, length = 300)
beta_density = dbeta(grid, true_alpha, true_beta)
lines(grid, beta_density)
```



Comparison

In the histogram plot, I change the histogram from frequency to density and also add the true beta(6,4) density line in the graph. It seems that the generated samples follow a beta distribution. The following graph shows the true beta distribution with empirical density.

```
x = rbeta(iter, 6, 4)
hist(x, freq = FALSE, main = "true beta")
lines(density(chain))
```



```
ks.test(chain[-(1:5000)], x)
```

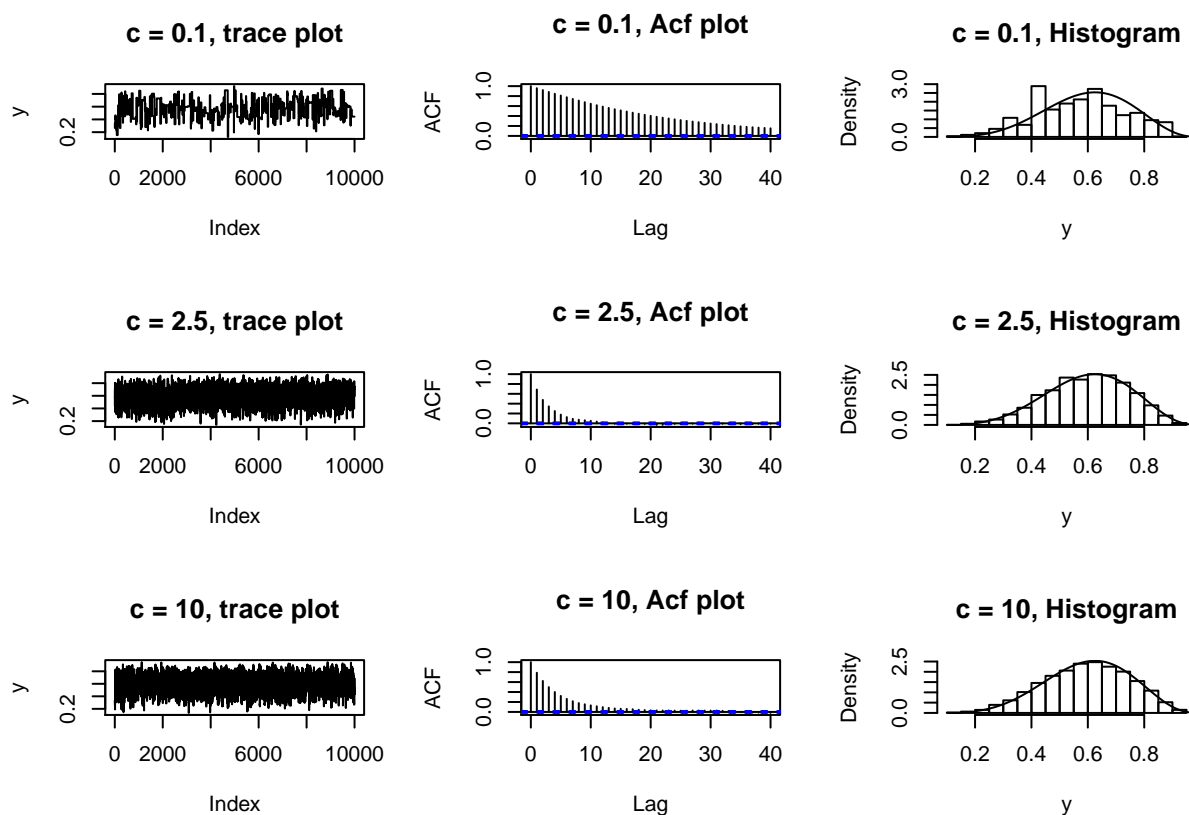
```
##
## Two-sample Kolmogorov-Smirnov test
##
## data: chain[-(1:5000)] and x
## D = 0.0186, p-value = 0.199
## alternative hypothesis: two-sided
```

Burn in the first 5000 samples, and compare the random data from $\text{beta}(6,4)$. The Kolmogorov-Smirnov test shows that the p-value is greater than 0.05, which means that we can NOT reject the null hypothesis. Then we can conclude that data may follow beta distribution.

3. Re-run the sample with more parameter

```
c1 = 0.1; c2 = 2.5; c3 = 10
chain1 = Beta_MetropolisHastings(true_alpha = 6, true_beta = 4, c = c1, iter = 10000)
chain2 = Beta_MetropolisHastings(true_alpha = 6, true_beta = 4, c = c2, iter = 10000)
chain3 = Beta_MetropolisHastings(true_alpha = 6, true_beta = 4, c = c3, iter = 10000)
grid = seq(0.2,1,length = 300)
beta_density = dbeta(grid, true_alpha, true_beta)

par(mfrow = c(3, 3))
plot(chain1, type = "l", main = "c = 0.1, trace plot", ylab = "y")
acf(chain1, main = "c = 0.1, Acf plot")
hist(chain1, freq = FALSE, main = "c = 0.1, Histogram", xlab = "y")
lines(grid, beta_density)
plot(chain2, type = "l", main = "c = 2.5, trace plot", ylab = "y")
acf(chain2, main = "c = 2.5, Acf plot")
hist(chain2, freq = FALSE, main = "c = 2.5, Histogram", xlab = "y")
lines(grid, beta_density)
plot(chain3, type = "l", main = "c = 10, trace plot", ylab = "y")
acf(chain3, main = "c = 10, Acf plot")
hist(chain3, freq = FALSE, main = "c = 10, Histogram", xlab = "y")
lines(grid, beta_density)
```



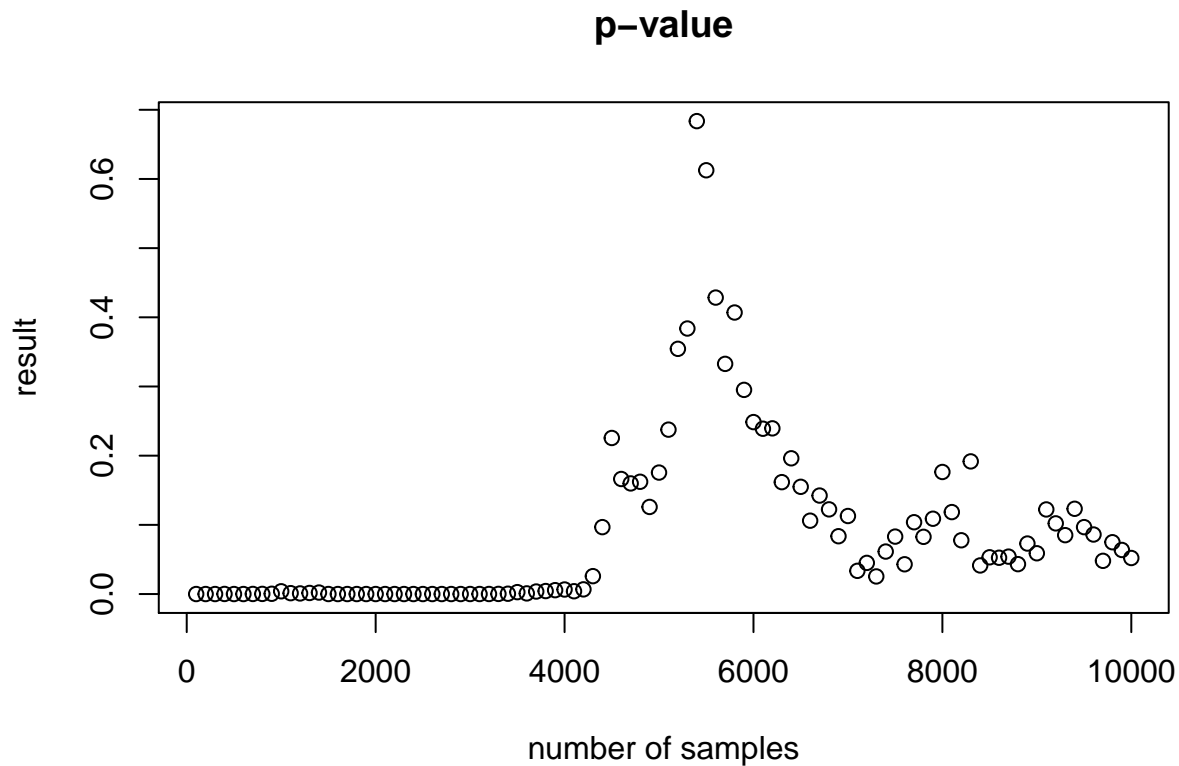
Analysis 1

From the trace, Acf plots and histograms, we find that when $c = 2.5$, it has the best fit for beta distribution and least autocorrelation. For smaller c or larger c , it has high autocorrelation, and the samples should be generated independently. Therefore, $c = 2.5$ would be the most effective at drawing from the target distribution.

Analysis 2

Consider different number of draws with Kolmogorov-Smirnov test:

```
x = rbeta(iter, 6, 4); result = c()
par(mfrow = c(1, 1))
sample = Beta_MetropolisHastings(true_alpha = 6, true_beta = 4, c = 2.5, iter = 10000)
for(i in 1:100){
  result = c(result, ks.test(sample[1:(i*100)],x)$p.value)
}
plot((1:100)*100,result, main = "p-value", xlab = "number of samples")
```



For this certain samples, we could see that when the sample number is greater than 4000, the p-value will be significant not zero and greater than 0.05, and we can NOT reject null hypothesis. The generated samples could be from a beta distribution when the number of draws is greater than 4000.