Final Project - Metropolis-Hastings

Boxin Zhao

10/27/2018

1. Algorithm description

My algorithm for M-H sampling is as below:

(1) Generate $x_1 \sim Uniform[0,1]$

(2) For t = 1 to (n.iteration - 1):

   (a) Generate $x^* \sim Beta(x|cx_t, c(1-x_t))$
   (b) Generate $u \sim Uniform[0,1]$
   (c) Let $\alpha(x_t, x^*) = min\{\frac{Beta(x^*|6,4)Beta(x_t|cx^*,c(1-x^*))}{Beta(x_t|6,4)Beta(x^*|cx_t,c(1-x_t))}, 1\}$ if $u < \alpha(x_t, x^*)$: $x_{t+1} = x^*$ else: $x_{t+1} = x_t$

Below is the implementation:

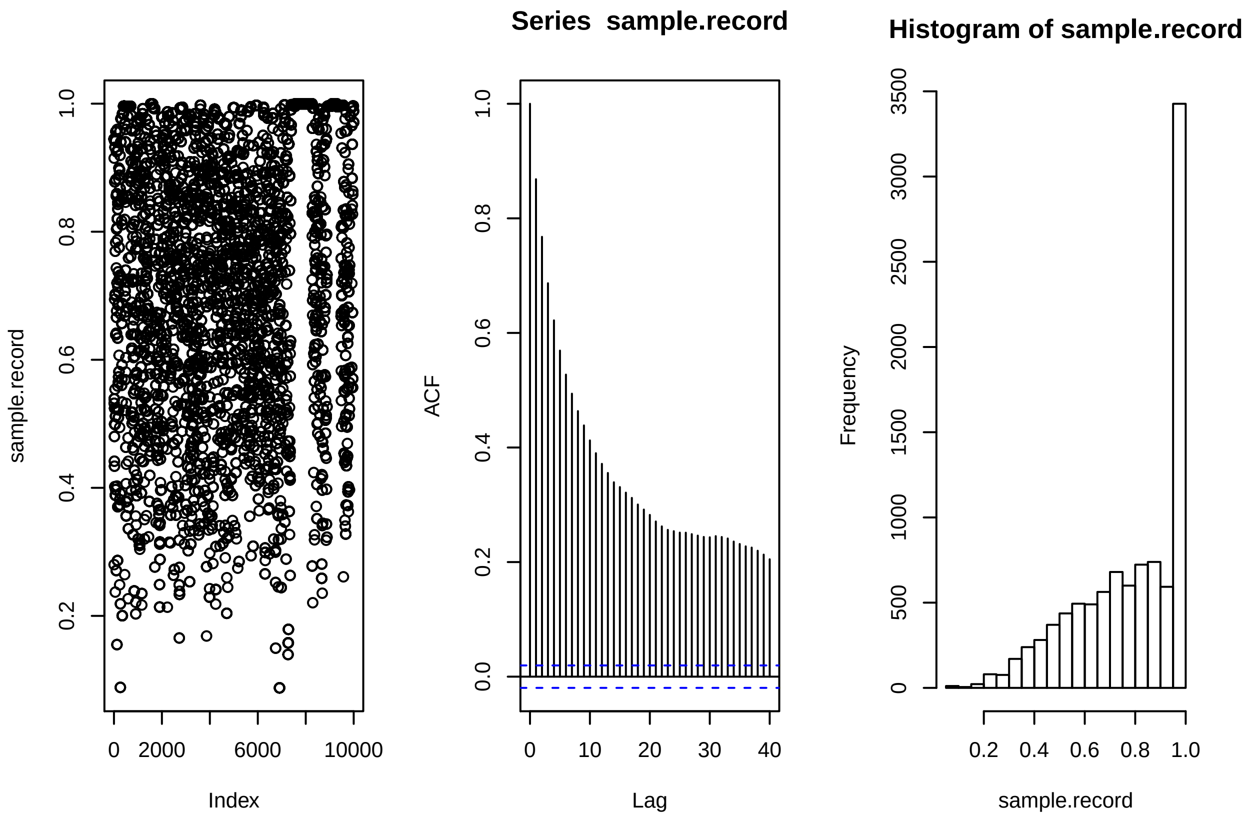
```
c <- 1
n.iteration <- 10000

sample.record <- numeric(n.iteration)

sample.record[1] <- runif(1)
for (t in 1:(n.iteration - 1)){
  xt <- sample.record[t]
  x.prop <- rbeta(1, c * xt, c * (1 - xt))
  u <- runif(1)
  alpha <- min(pbeta(x.prop, 6, 4) * pbeta(xt, c * x.prop, c * (1- x.prop)) /
               (pbeta(xt, 6, 4) * pbeta(x.prop, c * xt, c * (1- xt))), 1)
  if (u < alpha){
    sample.record[t + 1] <- x.prop
  } else {
    sample.record[t + 1] <- xt
  }
}
```

2. Evaluate the performance of the sampler.

Let's now analysis the result.

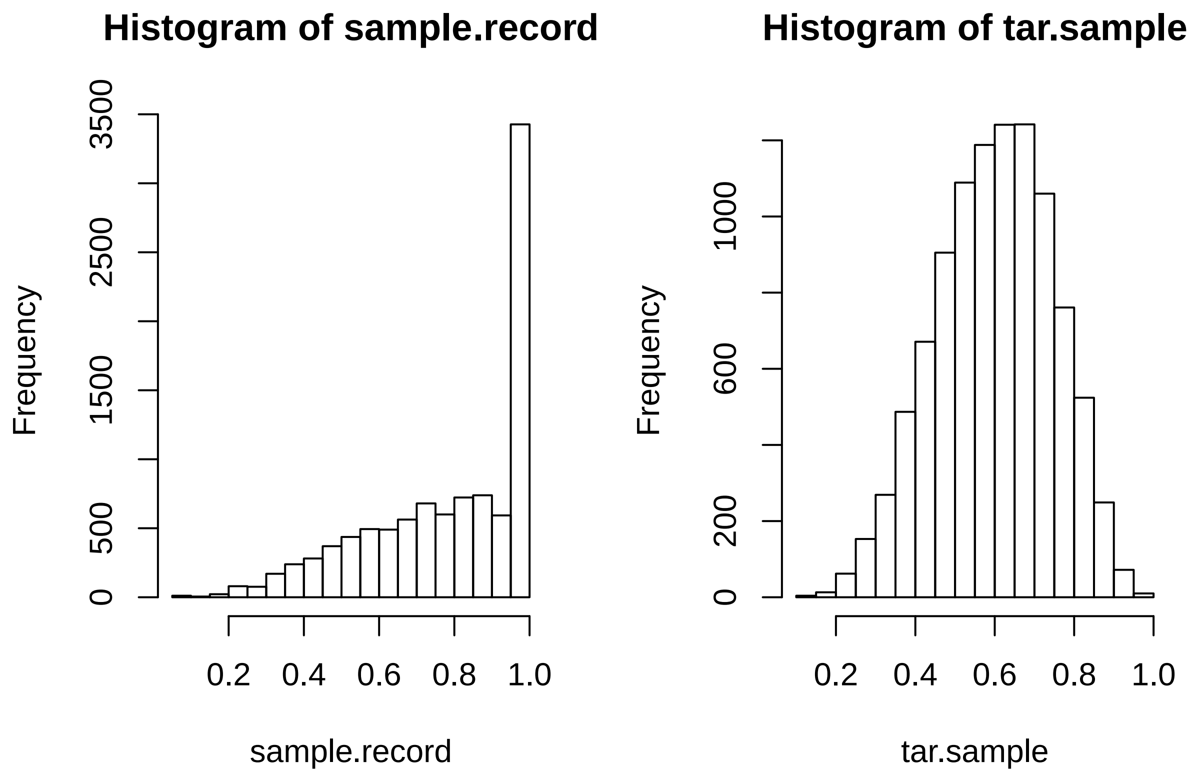
First, a trace plot of this sampler and an autocorrelation plot, as well as a histogram of the draws.

```
par(mfrow=c(1,3))   #1 row, 3 columns
plot(sample.record); acf(sample.record); hist(sample.record)   #plot commands
```



Compare the histogram of the draws with the target distribution

```
tar.sample <- rbeta(n.iteration, 6, 4)

par(mfrow=c(1,2))
hist(sample.record); hist(tar.sample)
```



And the Kolmogorov–Smirnov test

```
ks.test(sample.record, pbeta, 6, 4)
```

```
## Warning in ks.test(sample.record, pbeta, 6, 4): ties should not be present
## for the Kolmogorov-Smirnov test
```

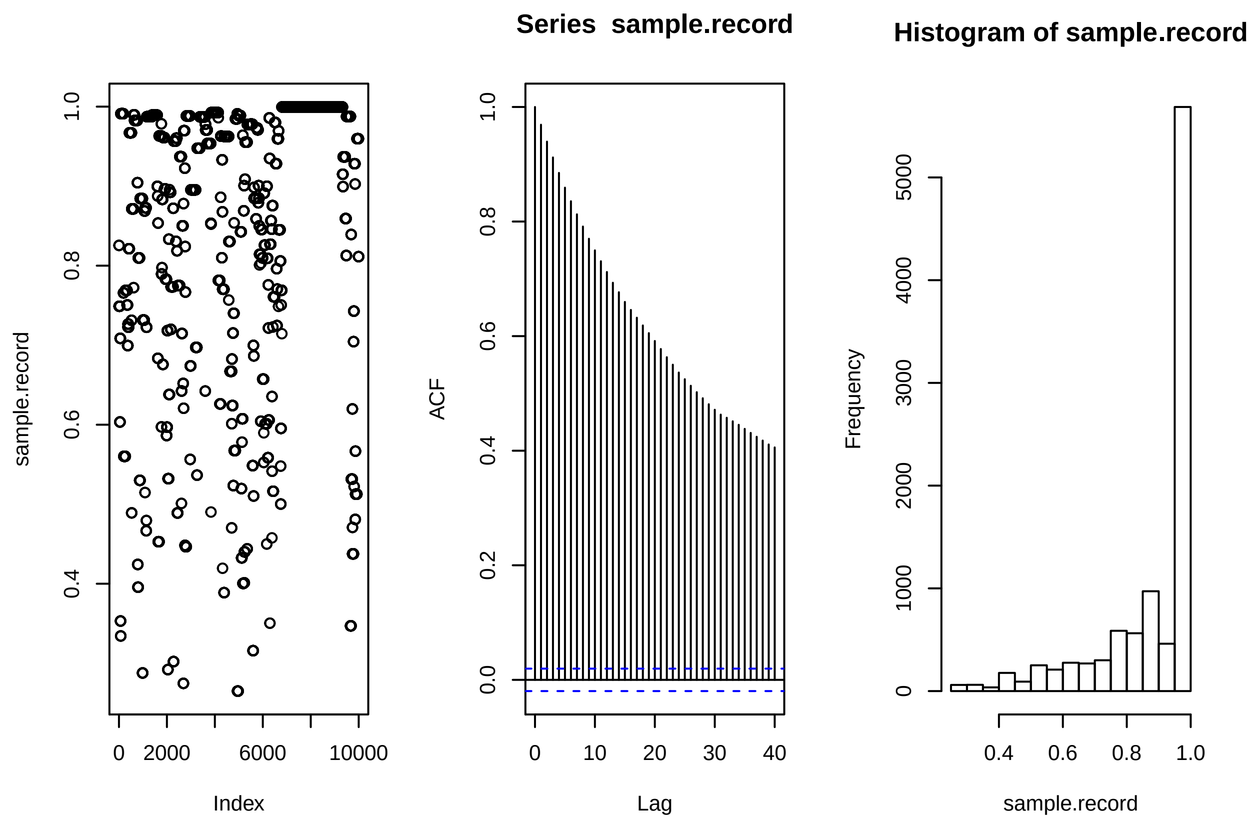
```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sample.record
## D = 0.46433, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

3. Re-run this sampler with $c = 0.1$, $c = 2.5$ and $c = 10$.

When $c = 0.1$

```
c <- 0.1
n.iteration <- 10000

sample.record <- numeric(n.iteration)

sample.record[1] <- runif(1)
for (t in 1:(n.iteration - 1)){
  xt <- sample.record[t]
  x.prop <- rbeta(1, c * xt, c * (1 - xt))
```

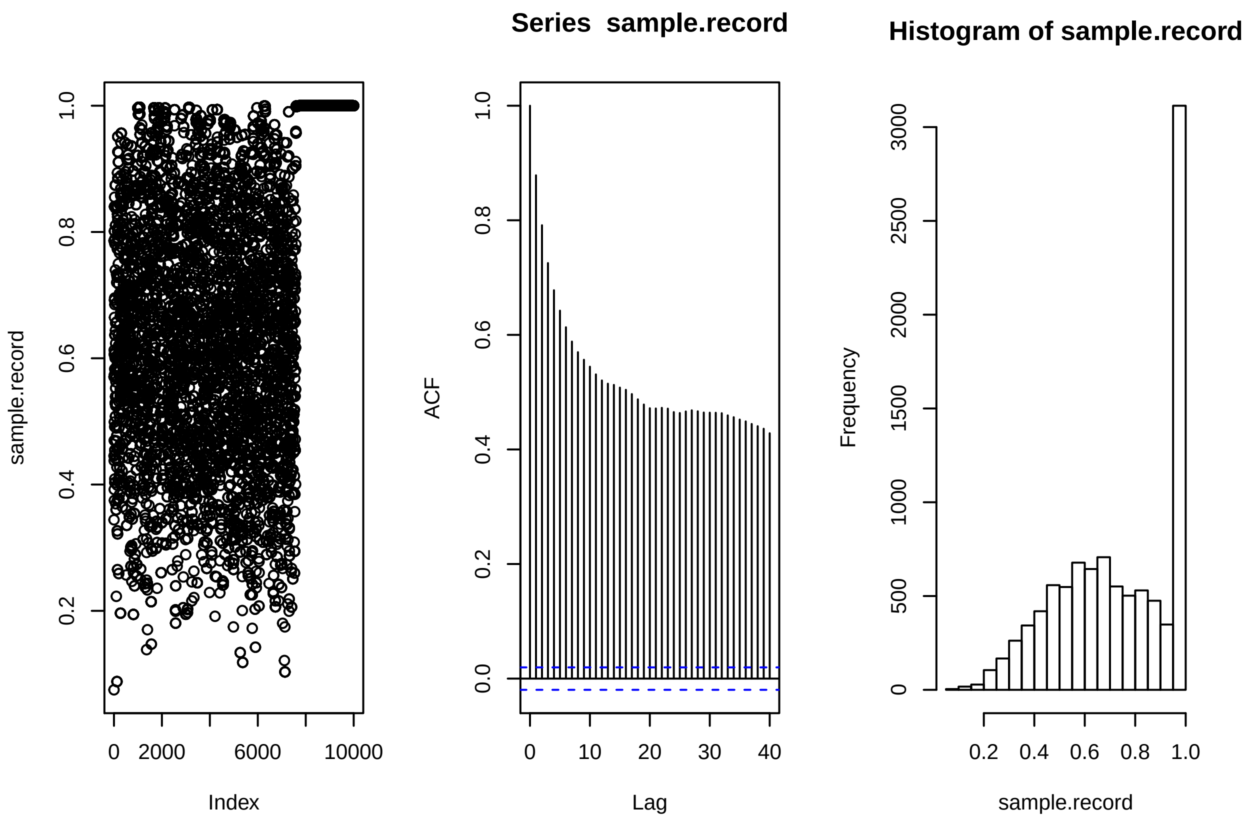
```
  u <- runif(1)
  alpha <- min(pbeta(x.prop, 6, 4) * pbeta(xt, c * x.prop, c * (1- x.prop)) /
                (pbeta(xt, 6, 4) * pbeta(x.prop, c * xt, c * (1- xt))), 1)
  if (u < alpha){
    sample.record[t + 1] <- x.prop
  } else {
    sample.record[t + 1] <- xt
  }
}

par(mfrow=c(1,3))  #1 row, 3 columns
plot(sample.record); acf(sample.record); hist(sample.record)  #plot commands
```



When $c = 2.5$

```
c <- 2.5
n.iteration <- 10000

sample.record <- numeric(n.iteration)

sample.record[1] <- runif(1)
for (t in 1:(n.iteration - 1)){
  xt <- sample.record[t]
  x.prop <- rbeta(1, c * xt, c * (1 - xt))
  u <- runif(1)
  alpha <- min(pbeta(x.prop, 6, 4) * pbeta(xt, c * x.prop, c * (1- x.prop)) /
```

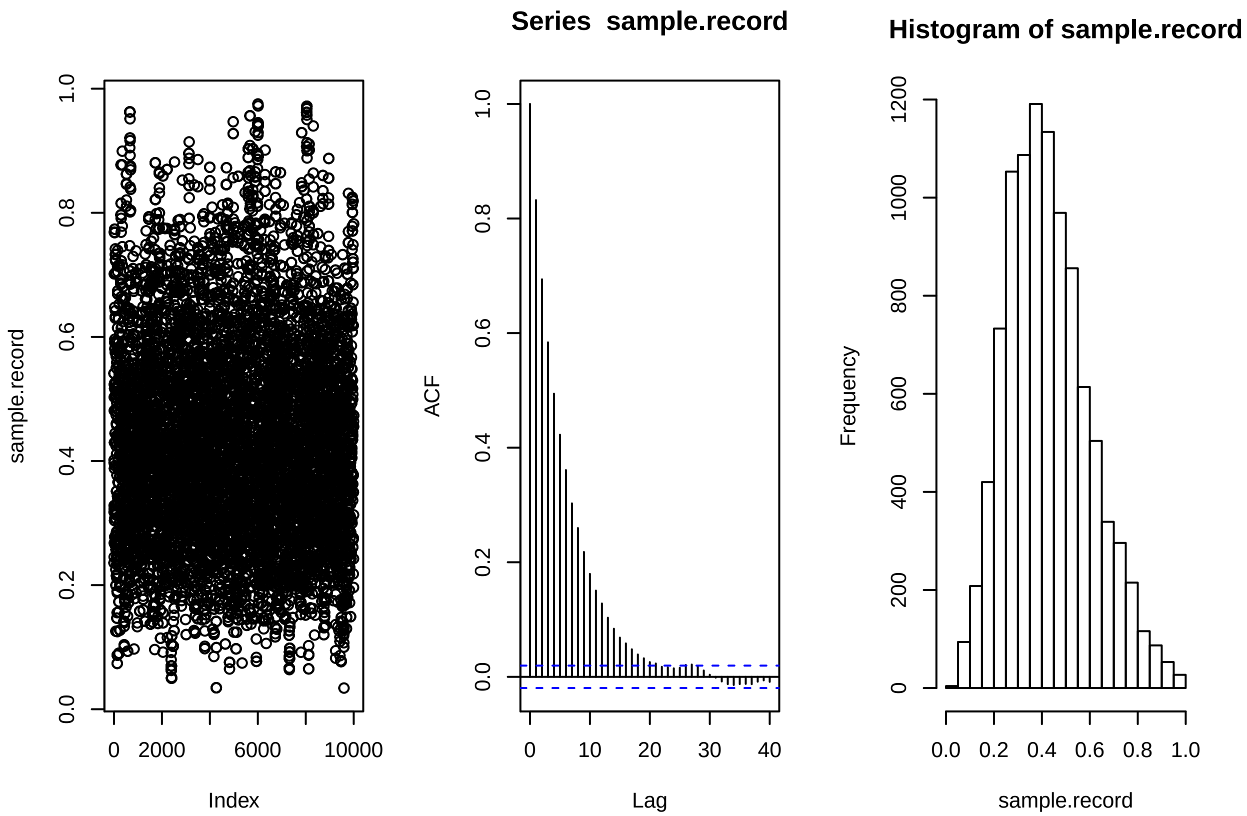
```
                 (pbeta(xt, 6, 4) * pbeta(x.prop, c * xt, c * (1- xt))), 1)
  if (u < alpha){
    sample.record[t + 1] <- x.prop
  } else {
    sample.record[t + 1] <- xt
  }
}

par(mfrow=c(1,3))  #1 row, 3 columns
plot(sample.record); acf(sample.record); hist(sample.record)  #plot commands
```



When c = 10

```
c <- 10
n.iteration <- 10000

sample.record <- numeric(n.iteration)

sample.record[1] <- runif(1)
for (t in 1:(n.iteration - 1)){
  xt <- sample.record[t]
  x.prop <- rbeta(1, c * xt, c * (1 - xt))
  u <- runif(1)
  alpha <- min(pbeta(x.prop, 6, 4) * pbeta(xt, c * x.prop, c * (1- x.prop)) /
                 (pbeta(xt, 6, 4) * pbeta(x.prop, c * xt, c * (1- xt))), 1)
  if (u < alpha){
```

```r
    sample.record[t + 1] <- x.prop
  } else {
    sample.record[t + 1] <- xt
  }
}

par(mfrow=c(1,3))   #1 row, 3 columns
plot(sample.record); acf(sample.record); hist(sample.record)   #plot commands
```



I think when c = 10, the sampler is most effective. Because the shape of the histogram is most similar to the hist of target distribution, which indicates we need to discard less samples (or a early burn-in). If we analyze the properties of target distribution, we can draw the same conclusion. Because for a x drawn from $Beta(6,4)$, the shape of $Beta(cx, c(1-x))$ is most similar to $Beta(6,4)$ when c = 10 (among c = 0.1, 1, 2.5 or 10). In this way, we will have a higher acceptance probability and more efficient drawing process.