

Metropolis-Hastings part

Fangzhou Yang, Kim Ting Li, adrian

10/30/2017

MAIN FUNCTION

Here we write a Metropolis-Hastings algorithm which is sampled from a Beta distribution. we should use the Detail Balance equation here:

$$\pi(\phi)T(\phi \rightarrow \phi') = \pi(\phi')T(\phi' \rightarrow \phi)$$

, which also can be written as

$$\frac{T(\phi \rightarrow \phi')}{T(\phi' \rightarrow \phi)} = \frac{\pi(\phi')}{\pi(\phi)}$$

- Firstly, we choose a start value: ϕ at random.
- Then, we separate the transition in two sub-steps: the proposal distribution ($Q(\phi \rightarrow \phi')$) and the acceptance probability ($A(\phi \rightarrow \phi')$).

$$T(\phi \rightarrow \phi') = Q(\phi \rightarrow \phi')A(\phi \rightarrow \phi')$$

in the part below, we write the proposal function of the form

$$\phi_{prop}|\phi_{old} \sim \text{Beta}(c\phi_{old}, c(1 - \phi_{old}))$$

, which is the probability of proposing a state $\phi'|\phi$.

And we create the posterior function which is

$$\frac{P(\phi')}{P(\phi_{old})} = \frac{\text{dbeta}(\phi', a, b)}{\text{dbeta}(\phi_{old}, a, b)}$$

where P follows the Beta distribution, and (a,b) = (6,4).

```
proposalfunction <- function(c, phi_old){
  return (rbeta(1, c*phi_old, c*(1-phi_old)))
}

posterior <- function(c, phi_old, trueA, trueB){
  return(dbeta(proposalfunction(c, phi_old), trueA, trueB)/dbeta(phi_old, trueA, trueB))
}
```

- Next, we write the Metropolis-Hastings algorithm which is sampled from a Beta distribution. and set the acceptance probability as:

$$A(\phi \rightarrow \phi') = \min(1, \frac{\pi(\phi')Q(\phi' \rightarrow \phi)}{\pi(\phi)Q(\phi \rightarrow \phi')})$$

If state is accepted, set the current value to ϕ' ; otherwise, set it to the startvalue ϕ .

*Final, we repeat this process until generate the required number of samples we need.

```
beta_metropolis <- function(c, startvalue, iterations){
  trueA = 6
  trueB = 4
  chain <- rep(0, iterations)
  chain[1] = startvalue
```

```

for (i in 1:iterations){
  phi <- proposalfunction(c, startvalue)
  proposal <- dbeta(startvalue, c*phi, c*(1-phi))/dbeta(phi, c*startvalue, c*(1-startvalue))
  post <- dbeta(phi,trueA,trueB)/dbeta(startvalue,trueA,trueB)
  probab <- min(1,post*proposal)

  if(runif(1) < probab){
    startvalue <- phi
    chain[i] <- phi
  } else {
    chain[i] <- startvalue
  }
}
return(chain)
}

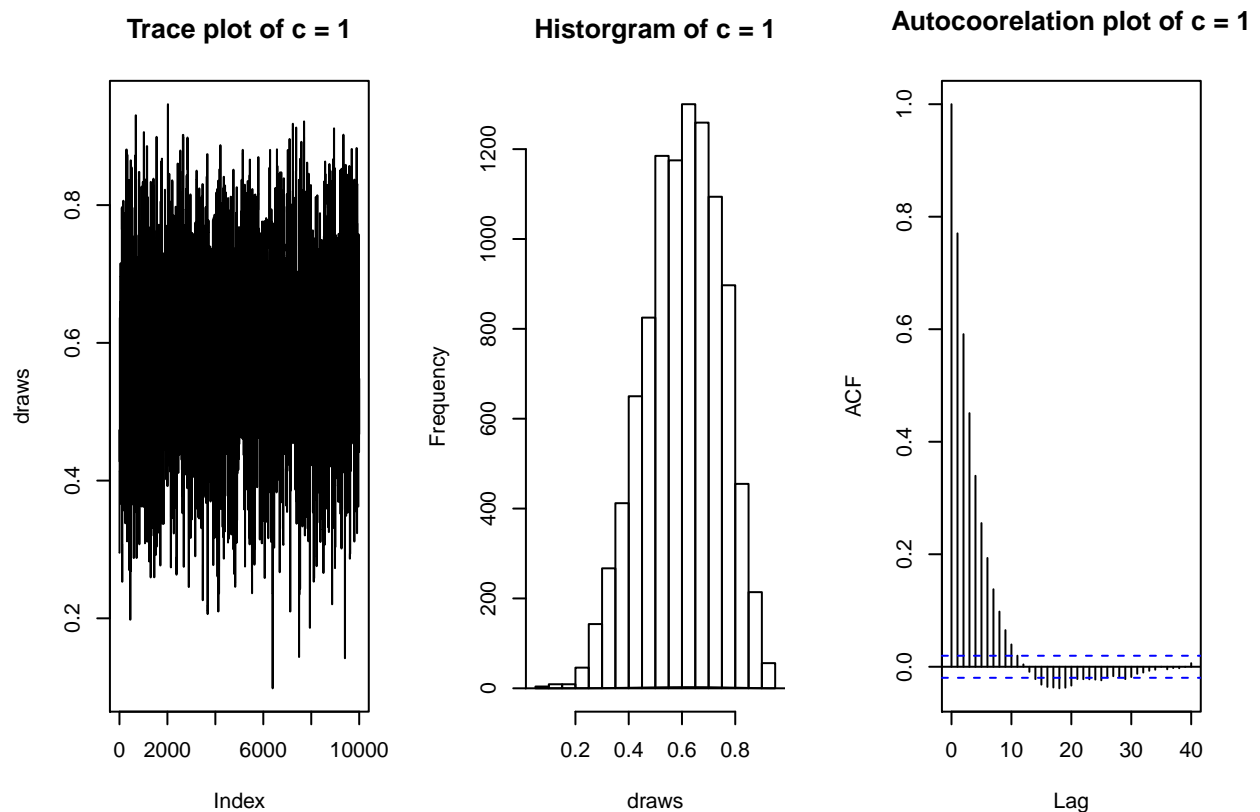
```

by test, we give a sample with $c = 1$, iterations = 10000 and start value = random uniform(1):

```

draws <- beta_metropolis(1,runif(1),10000)
par(mfrow = c(1,3))
plot(draws,type = "l",main = "Trace plot of c = 1")
hist(draws, main = "Histogram of c = 1")
x <- seq(0,1,0.01)
lines(x,dbeta(x,6,4))
acf(draws, main = "Autocoorelation plot of c = 1")

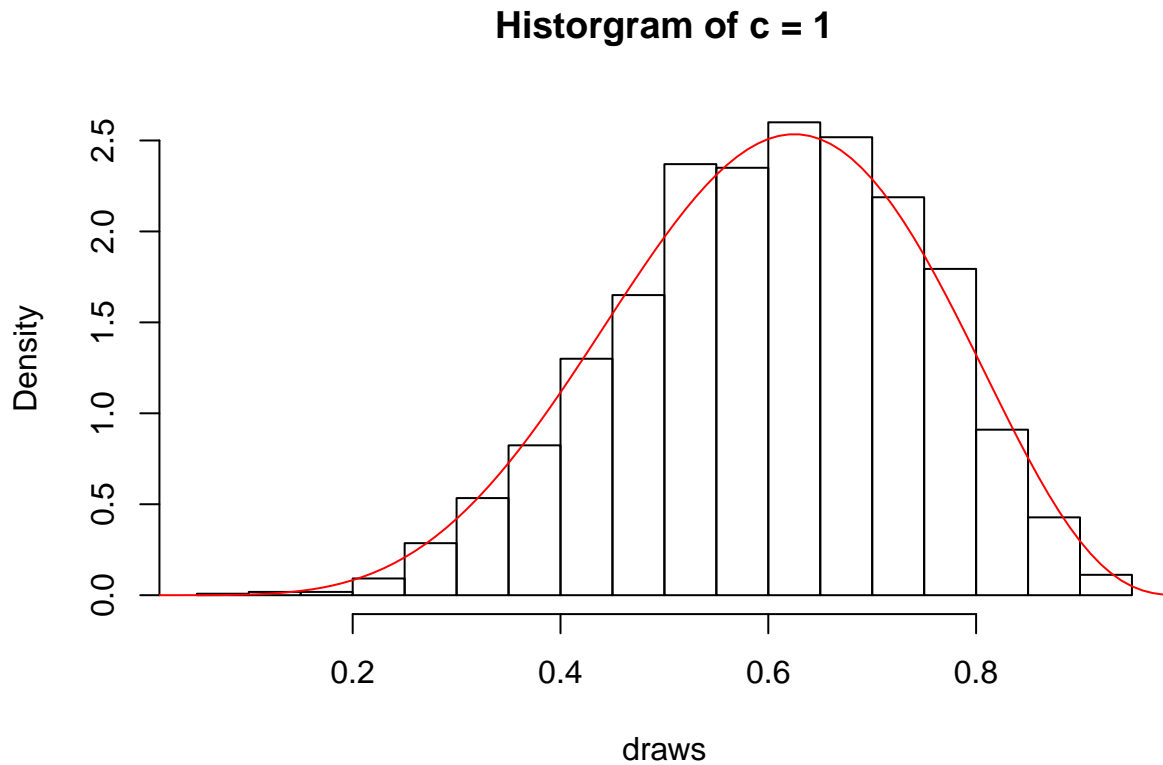
```



```

par(mfrow = c(1,1))
x <- seq(0,1,0.01)
hist(draws, main = "Histogram of c = 1",freq = FALSE)
lines(x,dbeta(x,6,4),col = "red")

```



```

ks.test(jitter(draws,0.0001),rbeta(10000,6,4))

```

```

##
## Two-sample Kolmogorov-Smirnov test
##
## data: jitter(draws, 1e-04) and rbeta(10000, 6, 4)
## D = 0.0268, p-value = 0.00152
## alternative hypothesis: two-sided

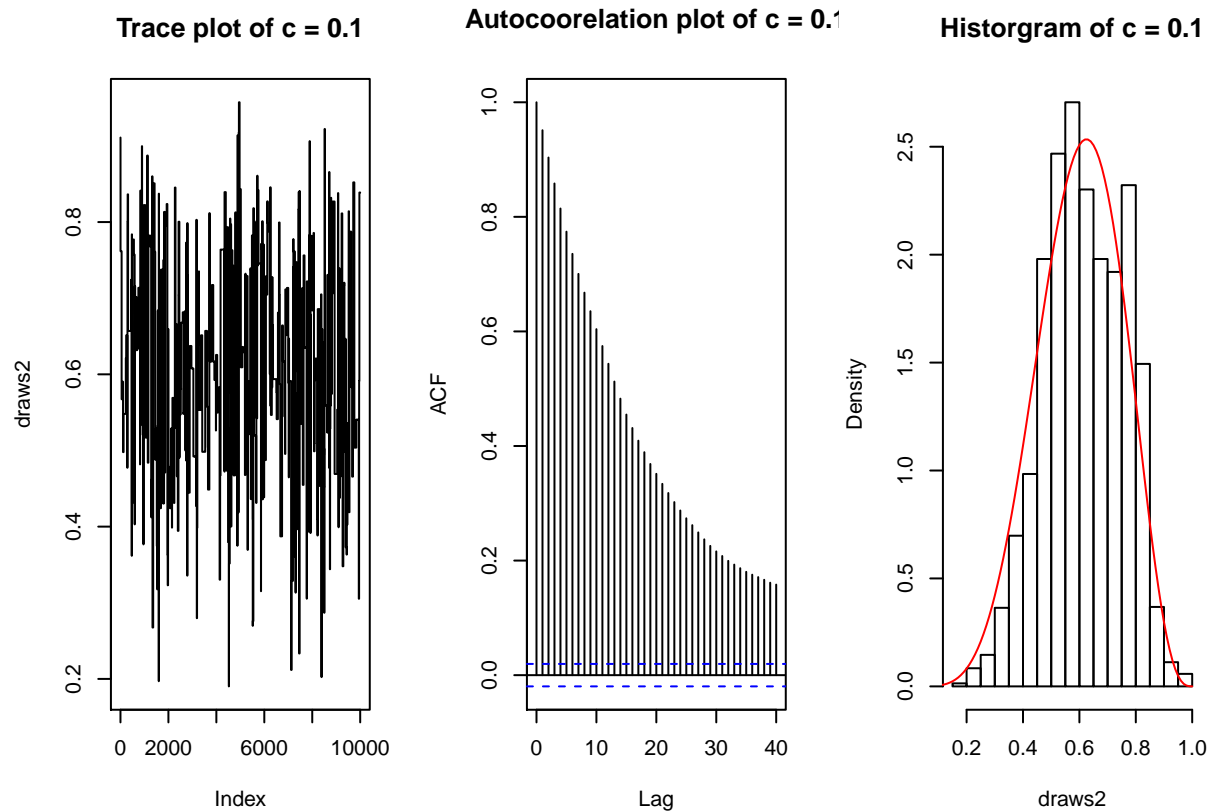
```

c = 0.1:

```

draws2 <- beta_metropolis(0.1,runif(1),10000)
par(mfrow = c(1,3))
plot(draws2,type = "l",main = "Trace plot of c = 0.1")
acf(draws2, main = "Autocoorelation plot of c = 0.1")
hist(draws2, main = "Histogram of c = 0.1",freq = FALSE)
x <- seq(0,1,0.01)
lines(x,dbeta(x,6,4),col = "red")

```

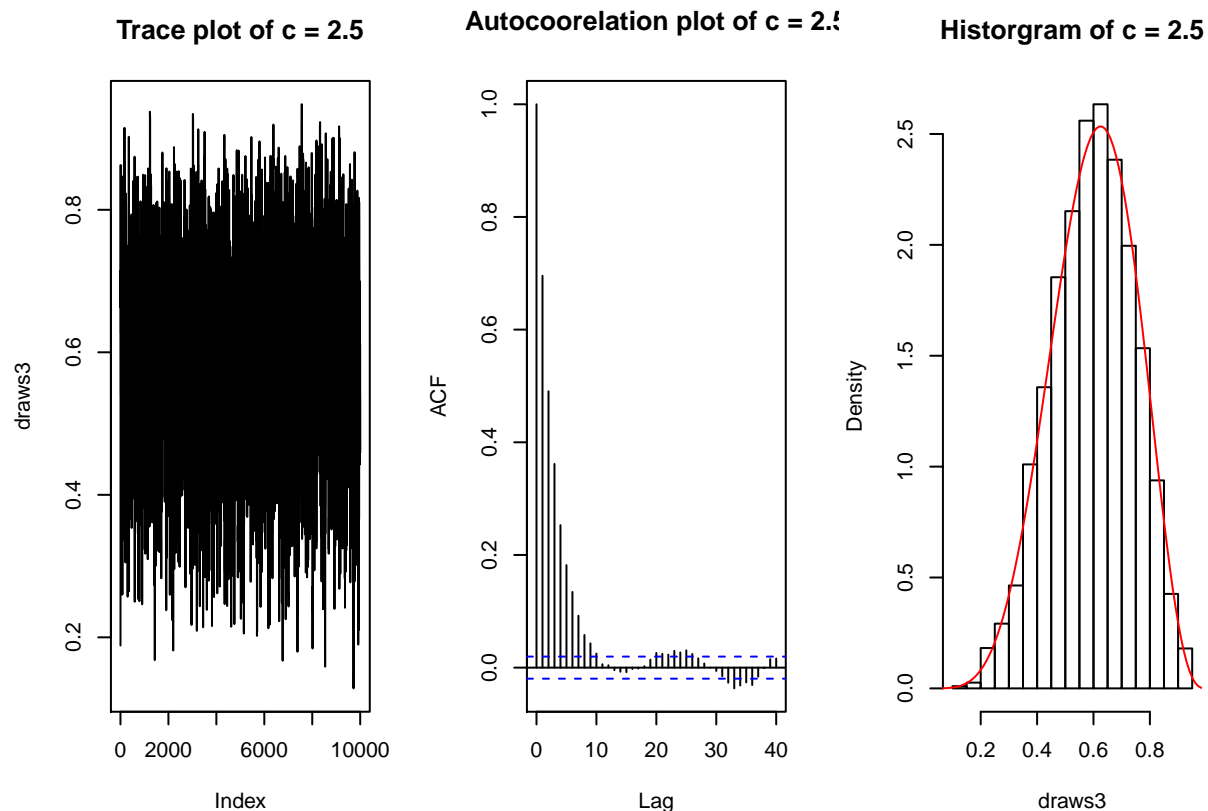


```
ks.test(jitter(draws2,0.0001),rbeta(10000,6,4))
```

```
##
## Two-sample Kolmogorov-Smirnov test
##
## data: jitter(draws2, 1e-04) and rbeta(10000, 6, 4)
## D = 0.066, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
c = 2.5:
```

```
draws3 <- beta_metropolis(2.5,runif(1),10000)
par(mfrow = c(1,3))
plot(draws3,type = "l",main = "Trace plot of c = 2.5")
acf(draws3, main = "Autocoorelation plot of c = 2.5")
hist(draws3, main = "Histogram of c = 2.5",freq = FALSE)
x <- seq(0,1,0.01)
lines(x,dbeta(x,6,4),col = "red")
```

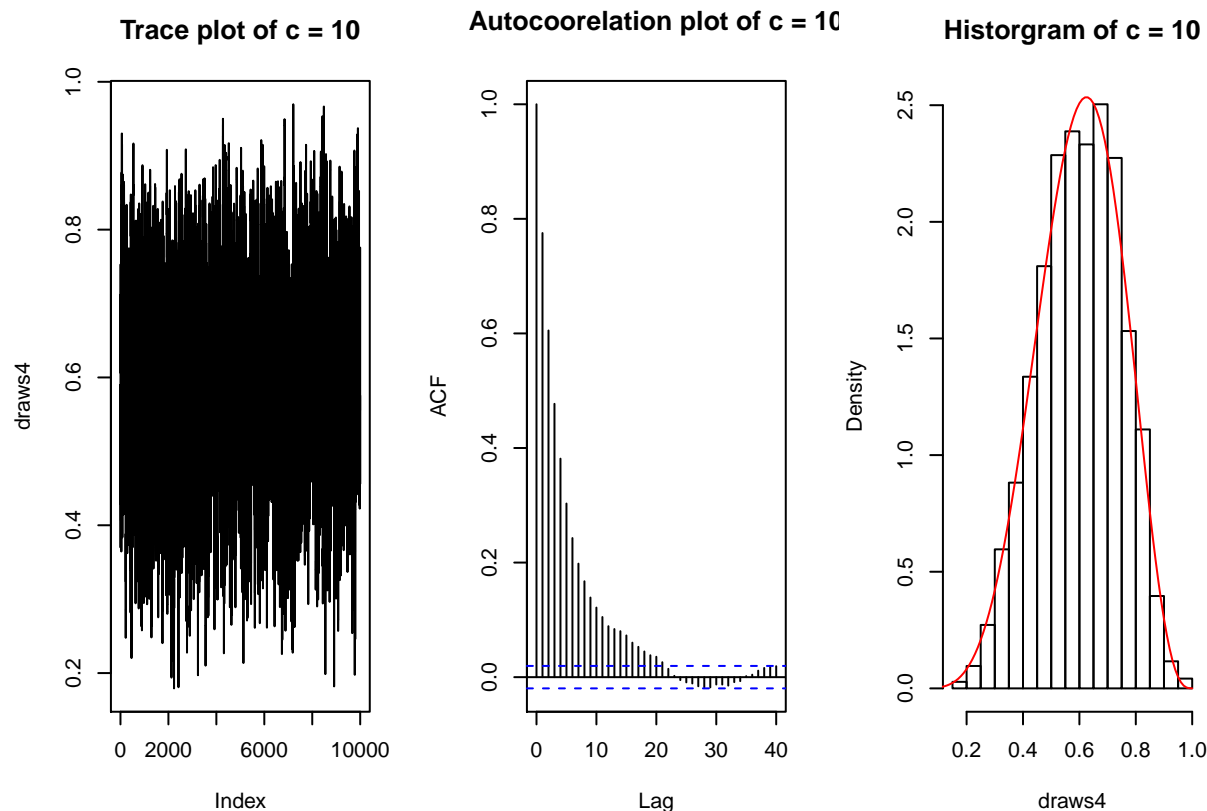


```
ks.test(jitter(draws3,0.0001),rbeta(10000,6,4))
```

```
##
## Two-sample Kolmogorov-Smirnov test
##
## data: jitter(draws3, 1e-04) and rbeta(10000, 6, 4)
## D = 0.0227, p-value = 0.01157
## alternative hypothesis: two-sided
```

```
c = 10:
```

```
draws4 <- beta_metroplis(10,runif(1),10000)
par(mfrow = c(1,3))
plot(draws4,type = "l",main = "Trace plot of c = 10")
acf(draws4, main = "Autocoorelation plot of c = 10")
hist(draws4, main = "Histogram of c = 10",freq = FALSE)
x <- seq(0,1,0.01)
lines(x,dbeta(x,6,4),col = "red")
```



```
ks.test(jitter(draws4,0.0001),rbeta(10000,6,4))
```

```
##
## Two-sample Kolmogorov-Smirnov test
##
## data: jitter(draws4, 1e-04) and rbeta(10000, 6, 4)
## D = 0.0135, p-value = 0.3219
## alternative hypothesis: two-sided
```

By re-run the sampler with $c = 0.1$, $c = 2.5$ and $c = 1$, we can see when c is getting larger, it meets the target distribution better and have larger p-value, but when c is too large, the autocorrelation would be worse.

1. Compare their plot, as c becomes larger, the more efficient it meets target distribution.
2. Through the acf plot, we can see that when $c = 10$, it is better than $c = 0.1$ but worse than $c = 2.5$. So we should choose a fitable c which cannot be too large or too small.
3. By comparing their histogram, we can see as c becomes larger, it meets the target distribution more.
4. Through the Kolmogorov-Smirnov statistic test, the p-value is getting larger as c gets larger.

So by comparing this values of c , $c = 2.5$ is most effective at drawing from the target distribution since it could meets the target distribution enough well and has a better autocorrelation plot than $c = 10$.