Metropolis-Hastings part

Fangzhou Yang, Kim Ting Li, Adrian Diaz

11/3/2017

MAIN FUNCTION

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

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

, which also can be written as

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

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

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

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

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

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

ANd we create the posterior funtion which is

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

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 writhe the Metropolis-Hastings algorithm which is sampled from a Beta distribution. and set the acceptance probability as:

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

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

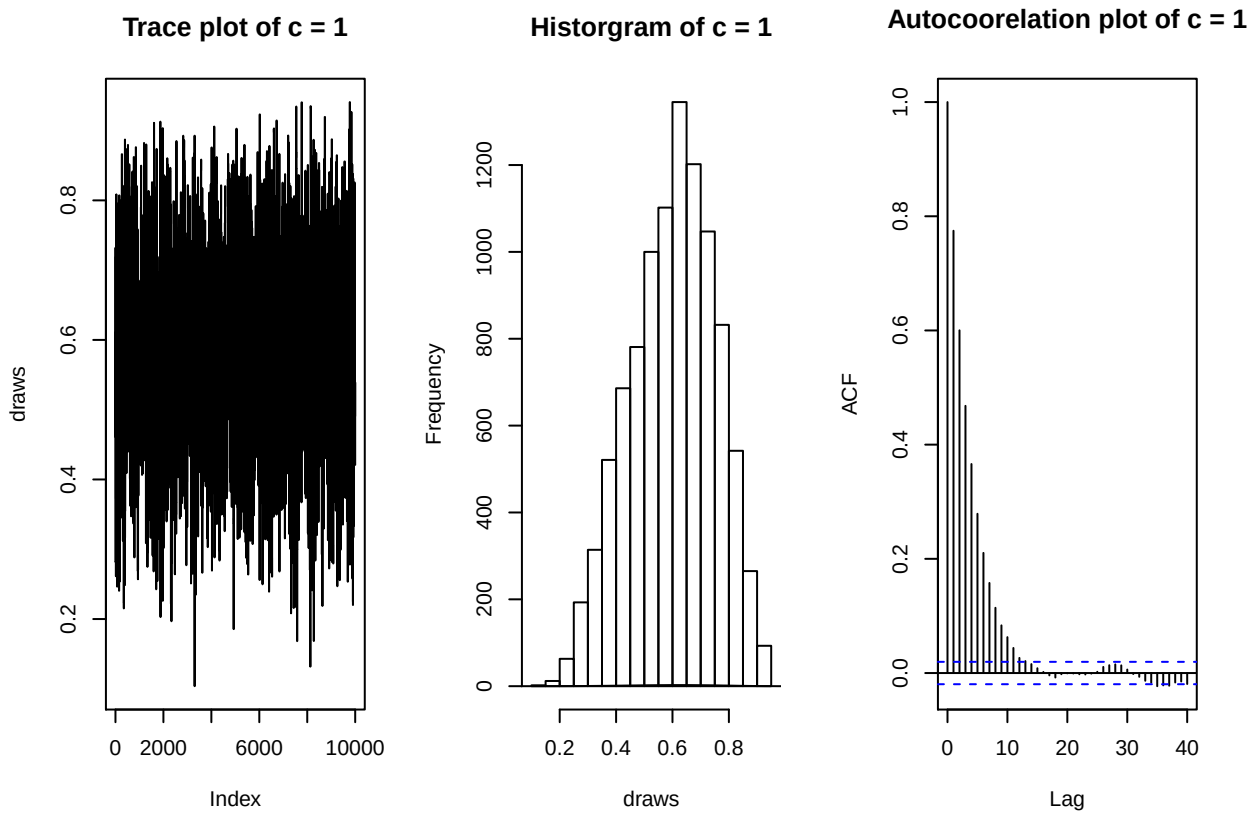
*Final, we repeat this process until generate the required number of samples we need.

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

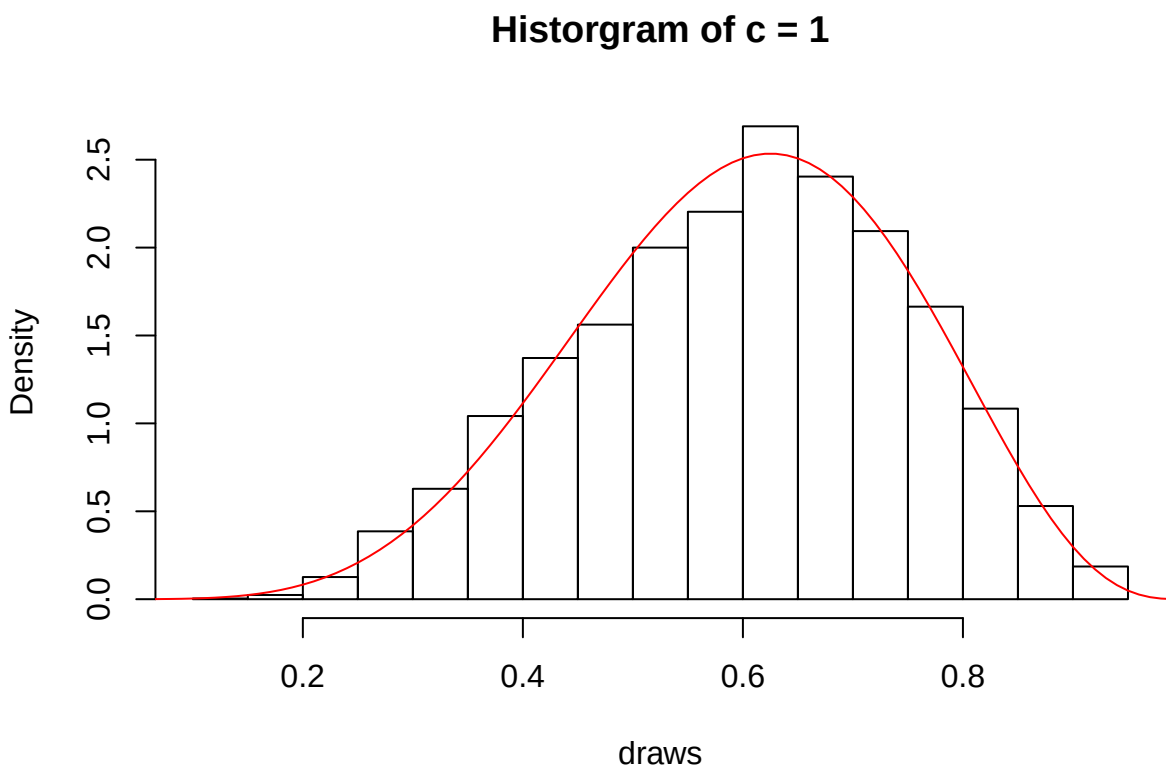
```
  for (i in 1:iterations){
    phi <- proposalfunction(c, startvalue)
    proposal <- dbeta(startvalue, c*phi, c*(1-phi))/dbeta(phi, c*startvalue, c*(1-startvalue))
    #post <- posterior(c,startvalue,trueA,trueB)
    post <- dbeta(phi,trueA,trueB)/dbeta(startvalue,trueA,trueB)
    probab <- min(1,post*proposal)

    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 = "Historgram of c = 1")
x <- seq(0,1,0.01)
lines(x,dbeta(x,6,4))
acf(draws, main = "Autocoorelation plot of c = 1")
```



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




```r
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 = 0.0201, p-value = 0.03519
## alternative hypothesis: two-sided
```

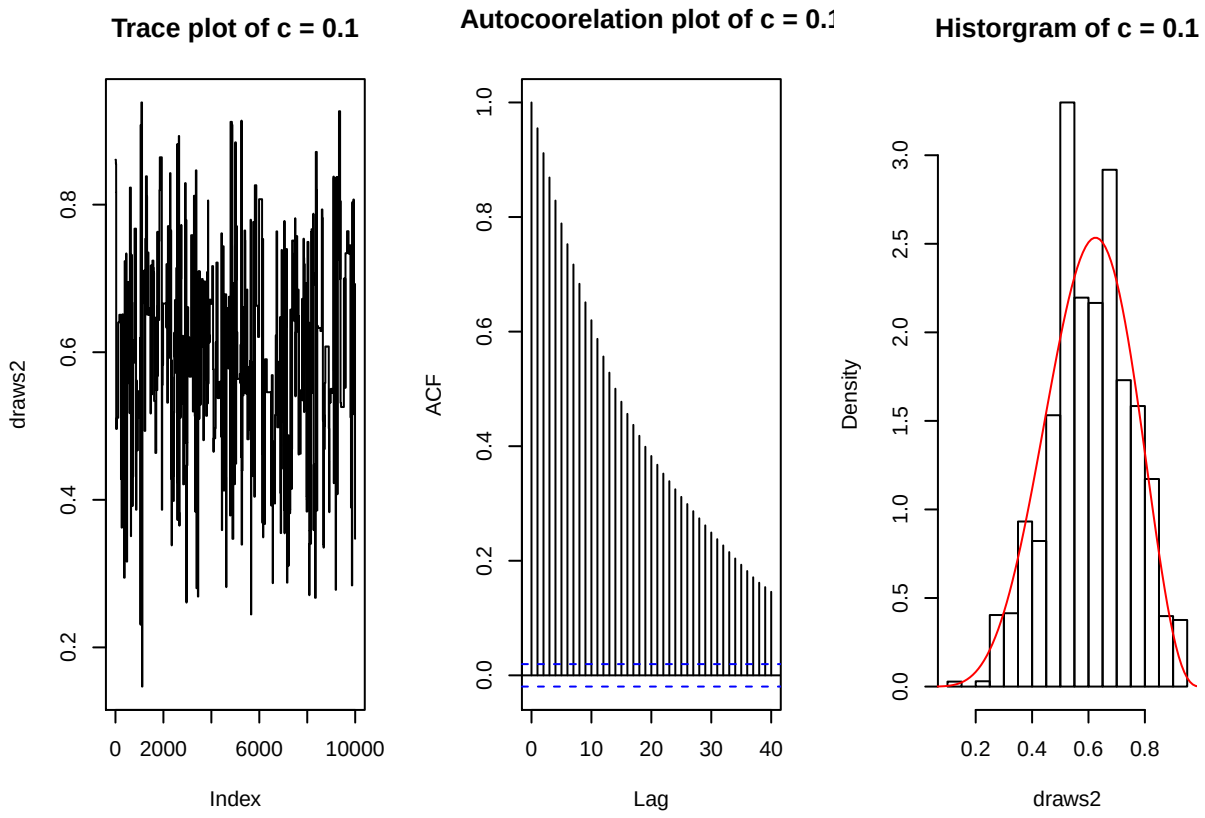
From the Kolmogorov¨CSmirnov statistic test, we can see the p-value equals 0.0009305,which is much smaller than p = 0.01, then we reject the null hypothesis, the data does not follow the beta distribution.

Re-run

c = 0.1:

```r
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 = "Historgram of c = 0.1",freq = FALSE)
```

```
x <- seq(0,1,0.01)
lines(x,dbeta(x,6,4),col = "red")
```



From the acf plot, we could see there are lots of lags, so it should be tested for 50,000 draws before thining and burn-in. After we thin abouth 50 times, test for 10,000 draws, the p-value is much better but still small, so it does not work.

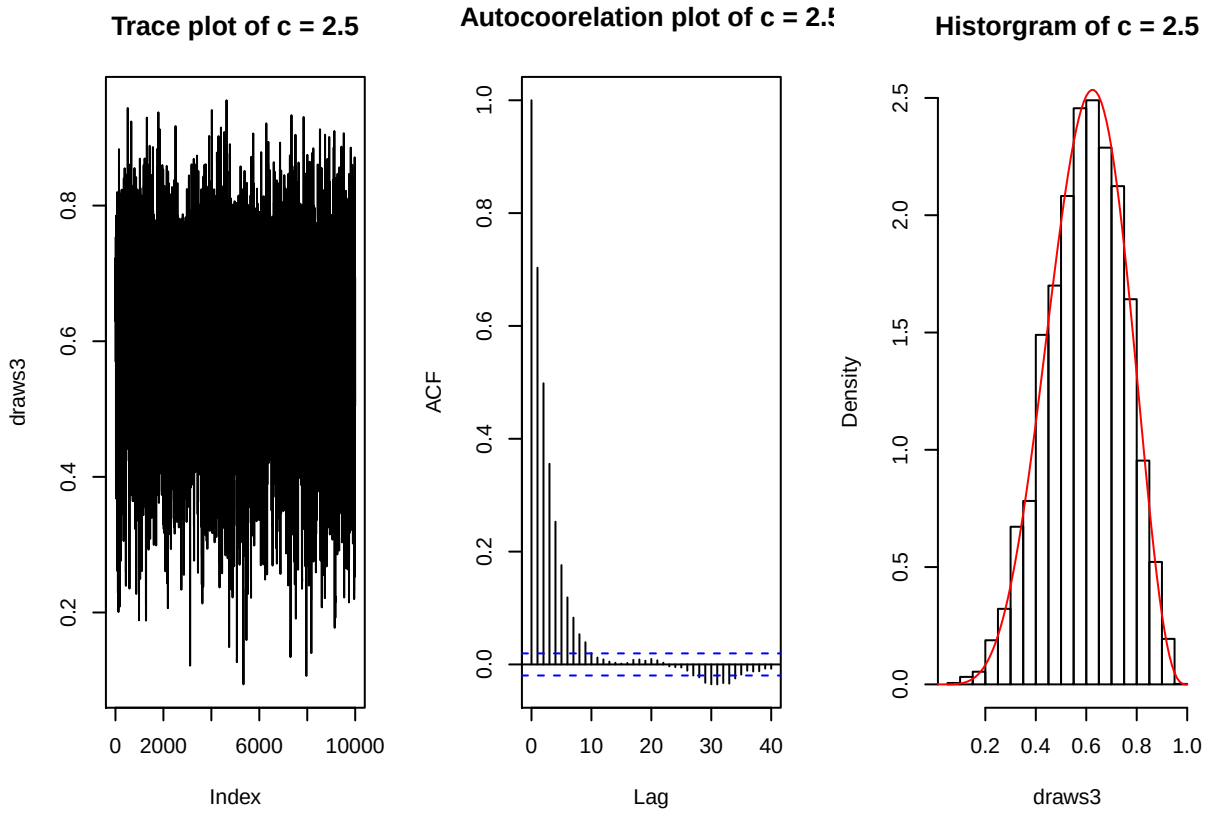
```
ks.test(jitter(draws2,0.0001),rbeta(10000,6,4))
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  jitter(draws2, 1e-04) and rbeta(10000, 6, 4)
## D = 0.0619, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

From the Kolmogorov¨CSmirnov statistic test, we can see the p-value equals 6.142e-11,which is much smaller than p = 0.01, then we reject the null hypothesis, the data does not follow the beta distribution.

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 = "Historgram of c = 2.5",freq = FALSE)
x <- seq(0,1,0.01)
lines(x,dbeta(x,6,4),col = "red")
```



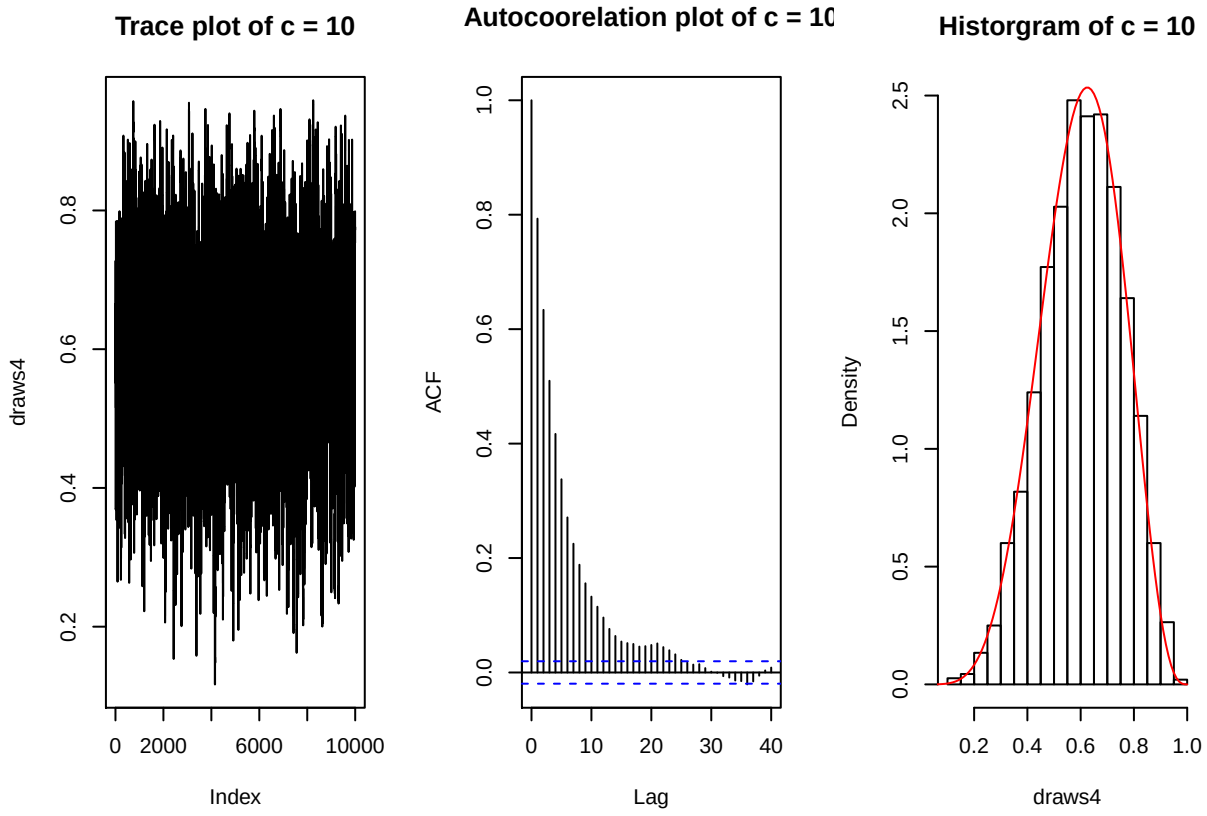
Test for 50,000 draws before thining and burn-in. After we thin 5 times,test for 10,000 draws, the p-value of k-s test is much better.

```
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 = 0.0149, p-value = 0.2169
## alternative hypothesis: two-sided
```

c = 10:

```
draws4 <- beta_metropolis(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 = "Historgram of c = 10",freq = FALSE)
x <- seq(0,1,0.01)
lines(x,dbeta(x,6,4),col = "red")
```



Test for 50,000 draws before thining and burn-in. After we thin 5 times,test for 10,000 draws, the p-value of k-s test is much better.

```
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 = 0.0281, p-value = 0.0007444
## alternative hypothesis: two-sided
```

From the Kolmogorov¨CSmirnov statistic test, we can see the p-value equals 0.08414,which is larger than p = 0.05, then we cannot reject the null hypothesis, the data follows the beta distribution.

Conclusion:

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 historgram, we can see as c becomes larger, it meets the target distribution more.
4. Through the Kolmogorov¨CSmirnov 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.