Metropolis-Hastings

Seung Ah Ha, Jaymo Kim, Wonbin Song

Metropolis-Hastings

Our target distribution is Beta(6,4) and our proposal distribution is $\phi_{prop}|\phi_{old} \sim Beta(c\phi_{old},\ c(1-\phi_{old}))$. First, we implement several functions which will be used in calculating posterior distribution. Here, we define likelihood function which gives the log-likelihood value of the parameter, and also prior function that gives the log density value of the parameter. The density used in the likelihood function is Beta(6,4) since it is our target distribution, and the density used in the prior function is Uniform(0,1) since our start value is from Uniform(0,1). In addition, since the parameters in Beta distribution have to be larger than 0, for c>0, $\phi < 1$.

```
a <- 6
b <- 4

likelihood <- function(param){
  singlelikelihoods = dbeta(param, a, b, log = T)
}

prior <- function(param){
  pr = dunif(param, min=0, max=1, log = T)
  return(pr)
}

posterior <- function(param){
  return (exp(likelihood(param) + prior(param)))
}
```

Posterior function can be calculated by multiplying likelihood function and prior function. However, we used log in the likelihood function and the prior function, so we need to convert them by putting *exp* in the posterior function.

```
metropolis_MCMC <- function(startvalue, iterations, c){
  chain = array(dim = c(iterations+1,1))
  chain[1,] = startvalue
  for (i in 1:iterations){
    proposal = rbeta(1, c*chain[i,], c*(1-chain[i,]))
    probab = (posterior(proposal)*dbeta(chain[i,],c*proposal,c*(1-proposal)))/
      (posterior(chain[i,])*dbeta(proposal, c*chain[i,], c*(1-chain[i,])))
    if (runif(1) < probab){
      chain[i+1,] = proposal
    }else{
      chain[i+1,] = chain[i,]
    }
  }
  return(chain)
}
```

After implementing all the functions that are required to obtain a chain, then we can finally define metropolis_MCMC function that can yield a chain. Note that

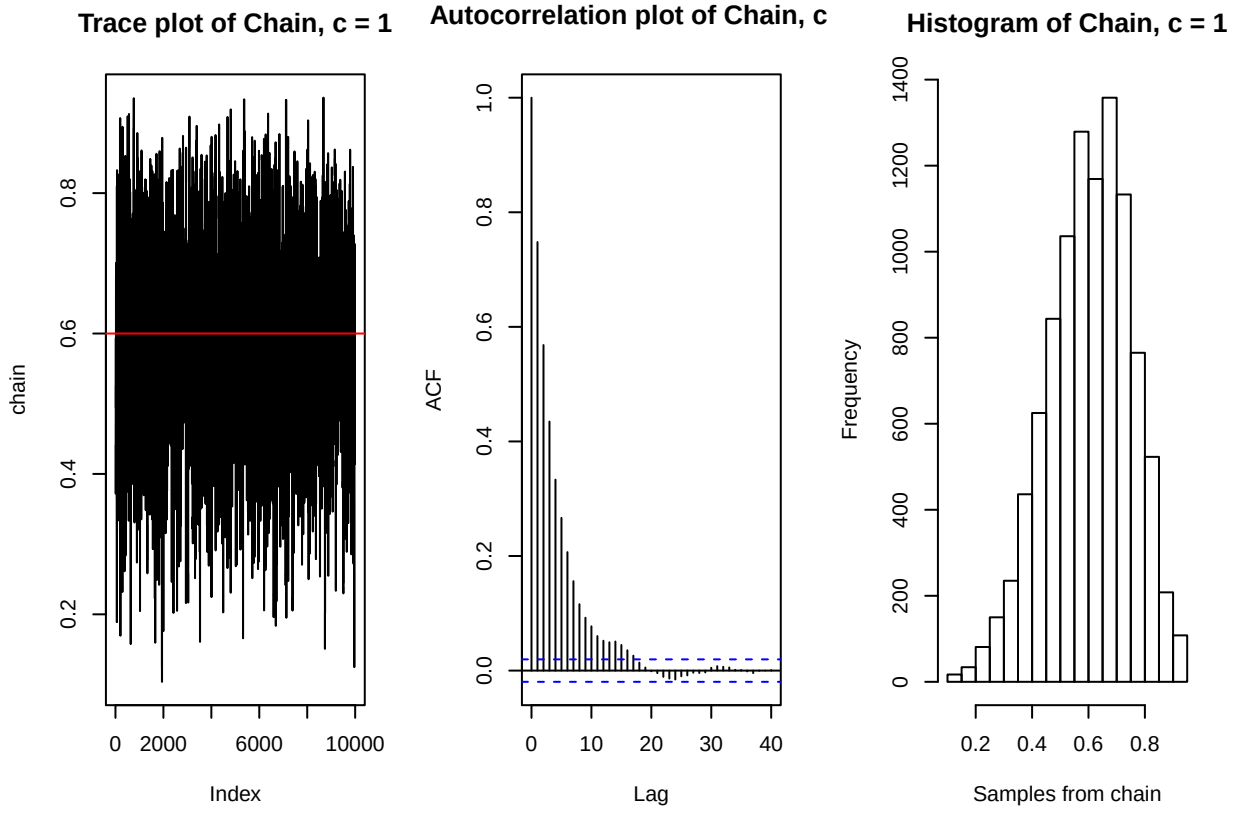
$$\phi_{new} = \phi_{proposal} \ \ with \ \ probability \ \ \alpha(\phi_{old}, \phi_{new})$$
$$= \phi_{old} \ \ with \ \ probaility \ \ 1 - \alpha(\phi_{old}, \phi_{new})$$

where $\alpha(\phi_{old}, \phi_{new}) = min(1, \frac{p(\phi_{new})q(\phi_{old}|\phi_{new})}{p(\phi_{old})q(\phi_{new}|\phi_{old})})$. Here, $p(\phi)$ is the posterior function, and $q(\phi_{new}|\phi_{old})$ is our proposal functions that follows $Beta(c\phi_{old}, \ c(1-\phi_{old}))$. Since Beta distribution is asymmetric, the probability form cannot be simplified to the ratio of posterior functions. Hence, we need to calculate the probabiltiy by using posterior functions and proposal functions.

Next, we can finally get the chain using metropolis_MCMC function with the startvalue generated from Uniform(0,1). We also generate random values from Beta(6,4) for the comparison.

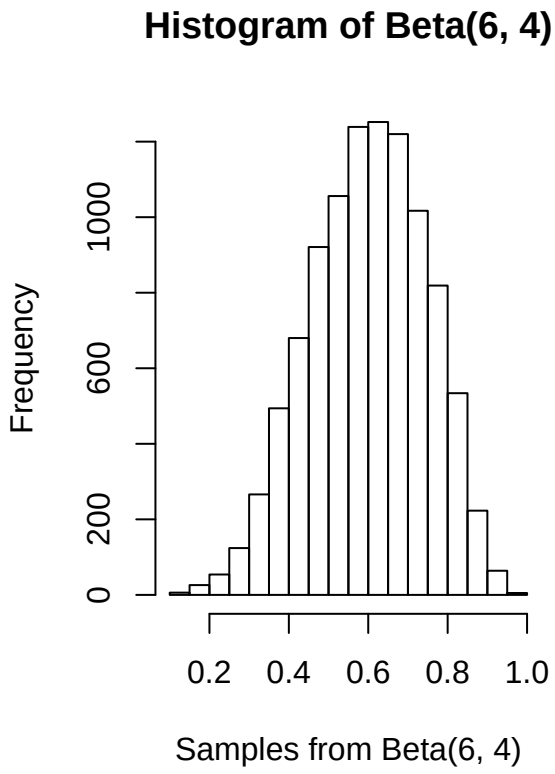
```
startvalue <- runif(1, 0, 1)
chain=metropolis_MCMC(startvalue, 10000, c=1)
test <- rbeta(10000, 6, 4)
```

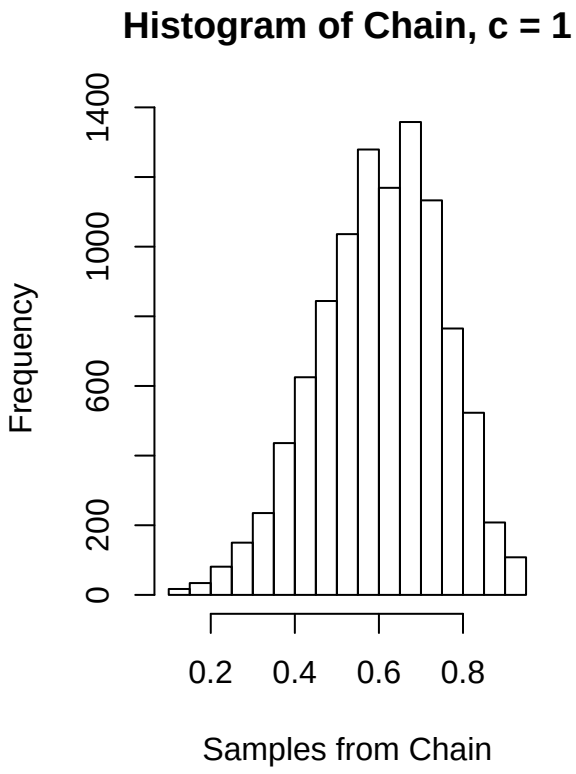
```
# Performance of the sampler: c = 1
par(mfrow=c(1,3))
plot(chain, type='l', main = "Trace plot of Chain, c = 1")
abline(h=0.6, col="red")
acf(chain, main = "Autocorrelation plot of Chain, c = 1")
hist(chain, xlab = "Samples from chain", main = "Histogram of Chain, c = 1")
```



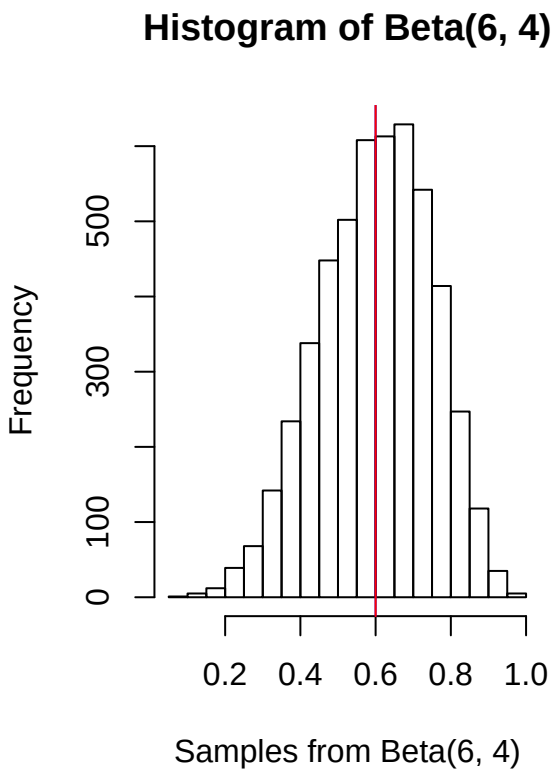

```
# Histogram of the chain vs. the target distribution Beta(6,4)
par(mfrow=c(1,2))
```

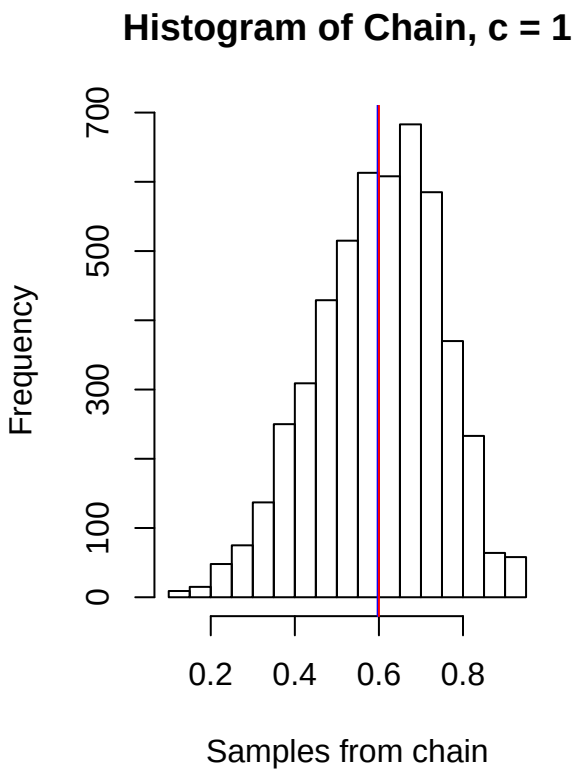
```r
hist(test, xlab = "Samples from Beta(6, 4)", main = "Histogram of Beta(6, 4)")
hist(chain, xlab = "Samples from Chain", main = "Histogram of Chain, c = 1")
```






```r
# Histogram of the target distribution Beta(6,4) vs. the Chain after burnIn
# Putting burnIn in
burnIn = 5000
test2 <- rbeta(5000, 6, 4) # true mean: 6/10 = 0.6
acceptance = 1-mean(duplicated(chain[-(1:burnIn),]))
hist(test2, xlab = "Samples from Beta(6, 4)", main = "Histogram of Beta(6, 4)")
abline(v=mean(test2), col="blue")
abline(v=0.6, col="red")
hist(chain[-(1:burnIn),1], xlab = "Samples from chain", main = "Histogram of Chain, c = 1")
abline(v = mean(chain[-(1:burnIn),1]), col="blue")
abline(v=0.6, col="red")
```



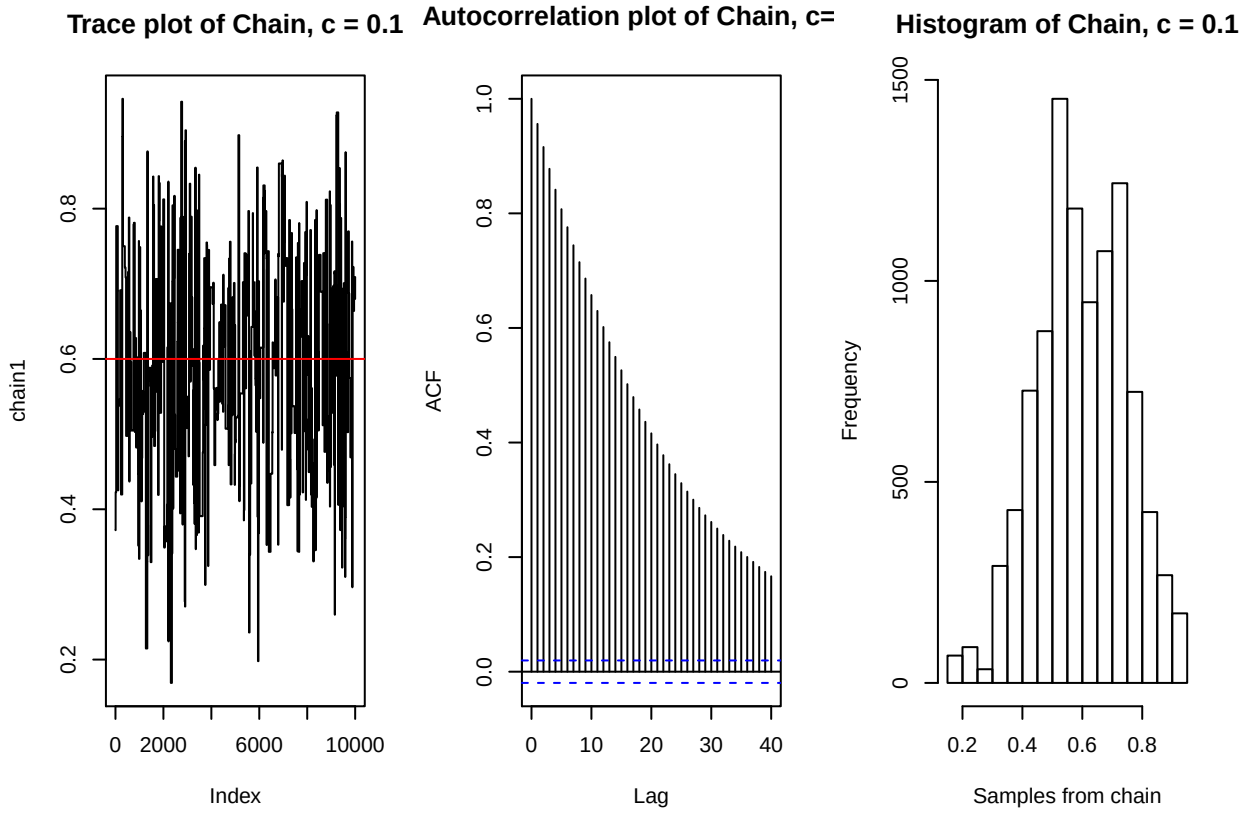



```
# Kolmogorov-Smirnov statistic
ks.test(chain[-(1:burnIn),1], test2)
```

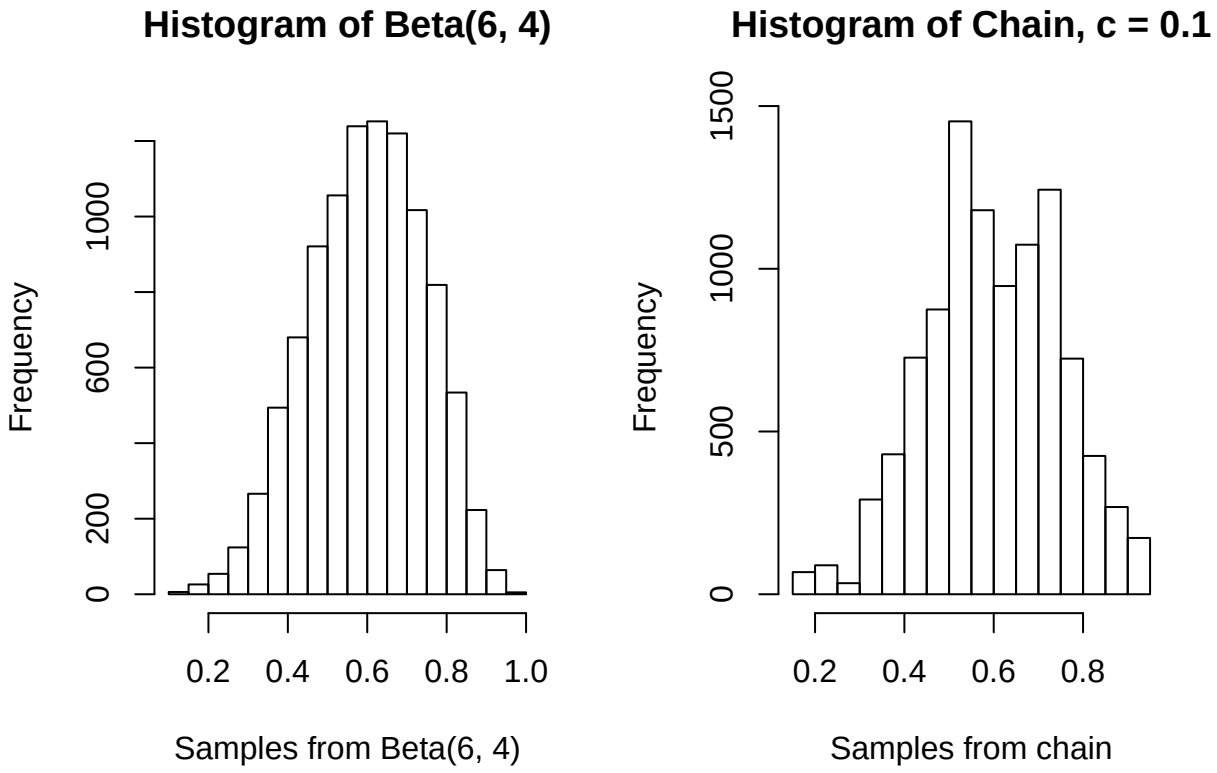
```
## Warning in ks.test(chain[-(1:burnIn), 1], test2): p-value will be
## approximate in the presence of ties

##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  chain[-(1:burnIn), 1] and test2
## D = 0.028031, p-value = 0.03933
## alternative hypothesis: two-sided
```

```
# c = 0.1 / c = 2.5 / c = 10
chain1=metropolis_MCMC(startvalue, 10000, c=0.1)
chain2=metropolis_MCMC(startvalue, 10000, c=2.5)
chain3=metropolis_MCMC(startvalue, 10000, c=10)

acceptance1 = 1-mean(duplicated(chain1[-(1:burnIn),]))
acceptance2 = 1-mean(duplicated(chain2[-(1:burnIn),]))
acceptance3 = 1-mean(duplicated(chain3[-(1:burnIn),]))

# Performance of the sampler: c = 0.1
par(mfrow=c(1,3))
plot(chain1, type='l', main = "Trace plot of Chain, c = 0.1")
```

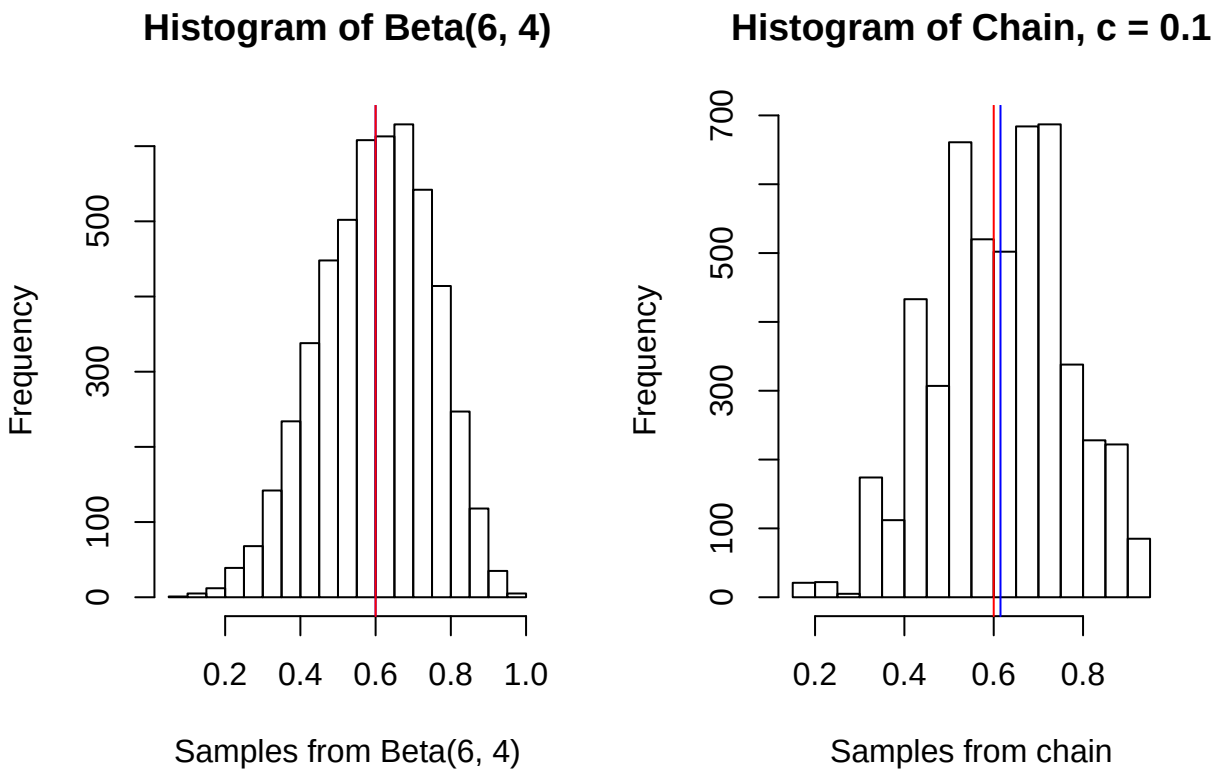
```
abline(h=0.6, col="red")
acf(chain1, main = "Autocorrelation plot of Chain, c= 0.1")
hist(chain1, xlab = "Samples from chain", main = "Histogram of Chain, c = 0.1")
```




```
# Comparing the histogram of the chain with the target distribution Beta(6,4)
par(mfrow=c(1,2))
hist(test, xlab = "Samples from Beta(6, 4)", main = "Histogram of Beta(6, 4)")
hist(chain1, xlab = "Samples from chain", main = "Histogram of Chain, c = 0.1")
```




```r
# Histogram of the target distribution Beta(6,4) vs. the Chain after burnIn
hist(test2, xlab = "Samples from Beta(6, 4)", main = "Histogram of Beta(6, 4)")
abline(v=mean(test2), col="blue")
abline(v=0.6, col="red")
hist(chain1[-(1:burnIn),1], xlab = "Samples from chain", main = "Histogram of Chain, c = 0.1")
abline(v = mean(chain1[-(1:burnIn),1]), col="blue")
abline(v=0.6, col="red")
```

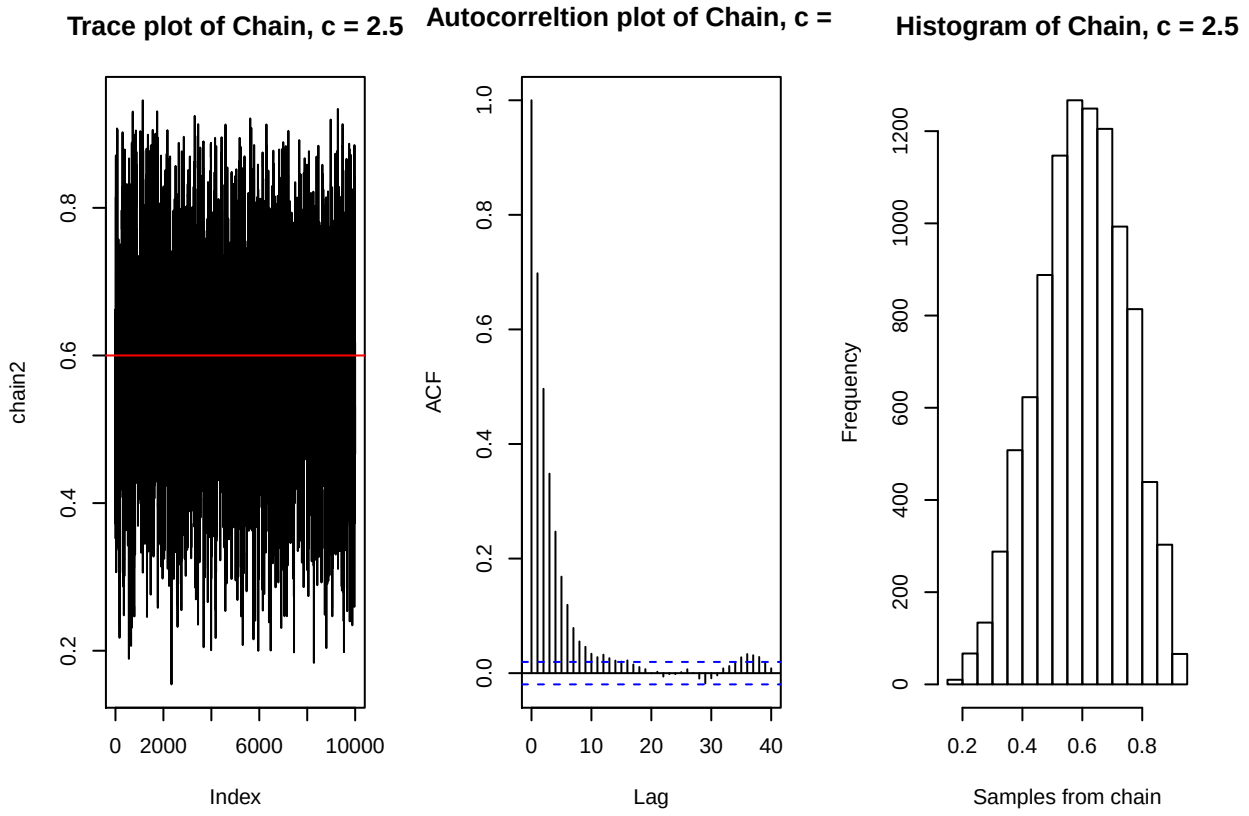



```
# Kolmogorov-Smirnov statistic
ks.test(chain1[-(1:burnIn),1], test2)
```

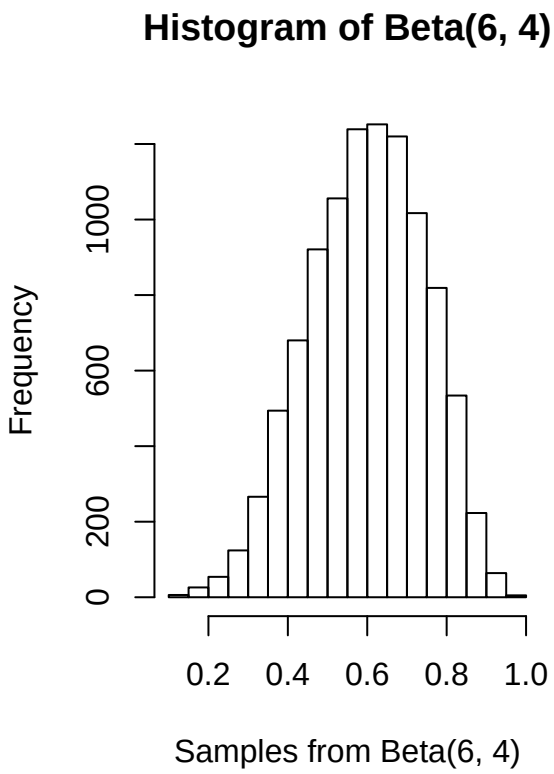
```
## Warning in ks.test(chain1[-(1:burnIn), 1], test2): p-value will be
## approximate in the presence of ties
```

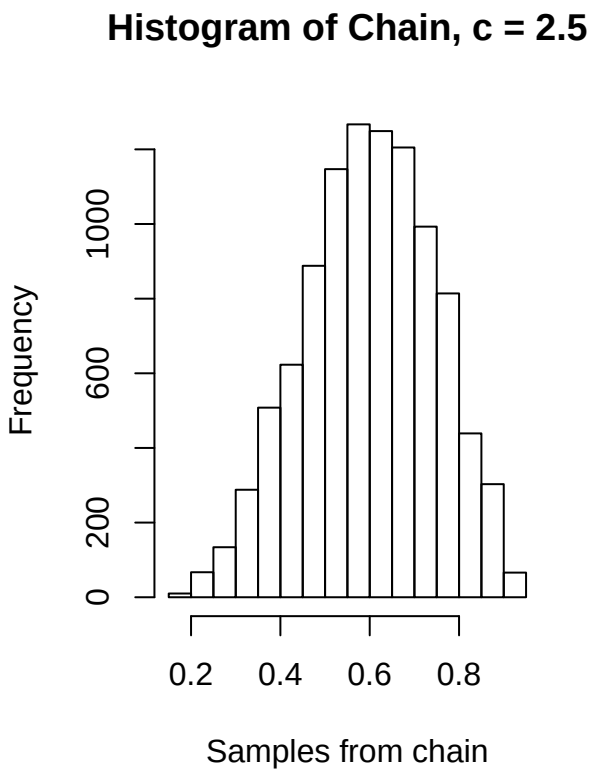
```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  chain1[-(1:burnIn), 1] and test2
## D = 0.07152, p-value = 1.558e-11
## alternative hypothesis: two-sided
```

```
# Performance of the sampler: c = 2.5
par(mfrow=c(1,3))
plot(chain2, type='l', main = "Trace plot of Chain, c = 2.5")
abline(h=0.6, col="red")
acf(chain2, main = "Autocorreltion plot of Chain, c = 2.5")
hist(chain2, xlab = "Samples from chain", main = "Histogram of Chain, c = 2.5")
```

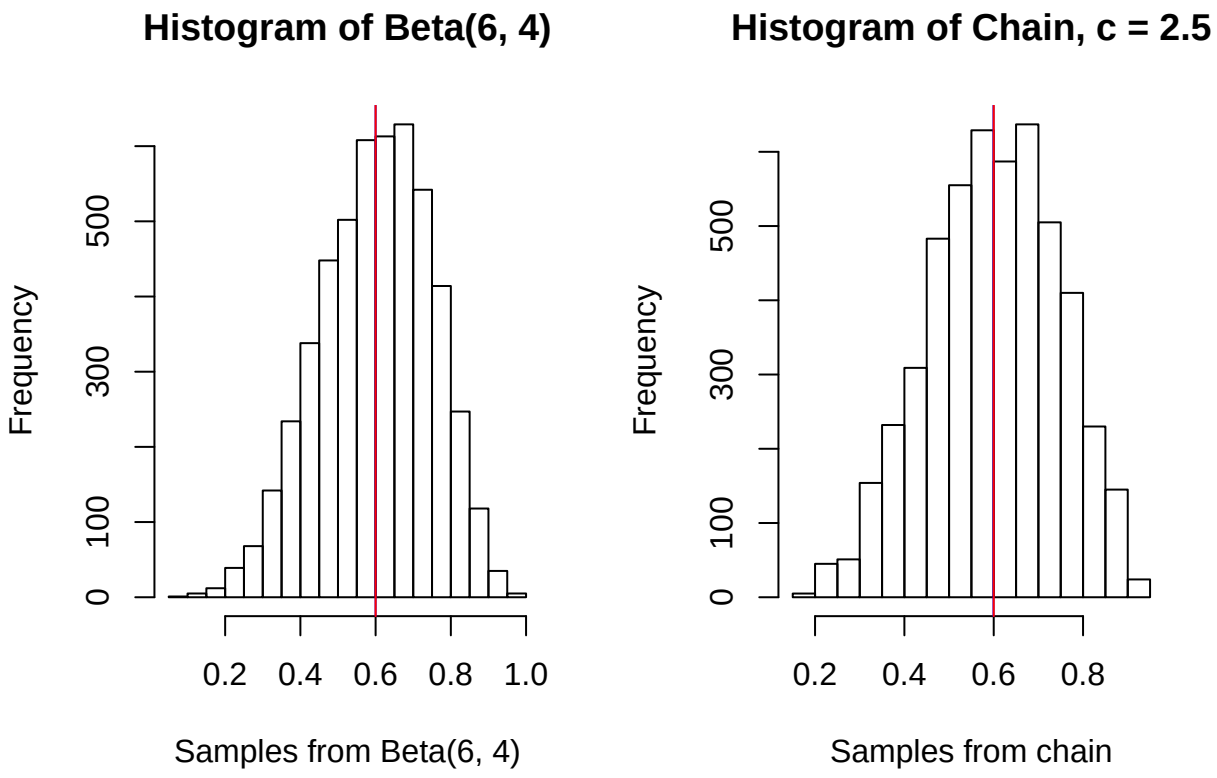



```
# Comparing the histogram of the chain with the target distribution Beta(6,4)
par(mfrow=c(1,2))
hist(test, xlab = "Samples from Beta(6, 4)", main = "Histogram of Beta(6, 4)")
hist(chain2, xlab = "Samples from chain", main = "Histogram of Chain, c = 2.5")
```






```r
# Histogram of the target distribution Beta(6,4) vs. the Chain after burnIn
hist(test2, xlab = "Samples from Beta(6, 4)", main = "Histogram of Beta(6, 4)")
abline(v=mean(test2), col="blue")
abline(v=0.6, col="red")
hist(chain2[-(1:burnIn),1], xlab = "Samples from chain", main = "Histogram of Chain, c = 2.5")
abline(v = mean(chain2[-(1:burnIn),1]), col="blue")
abline(v=0.6, col="red")
```

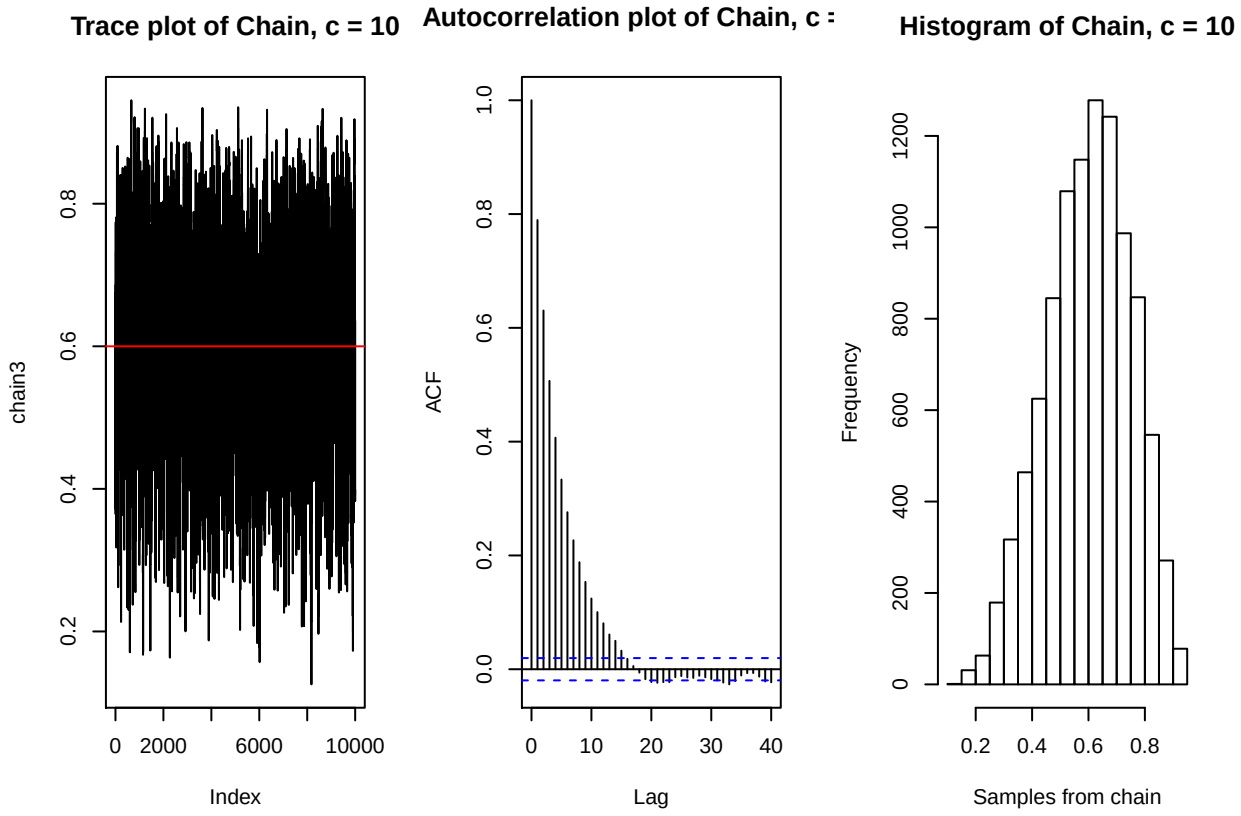



```
# Kolmogorov-Smirnov statistic
ks.test(chain2[-(1:burnIn),1], test2)
```

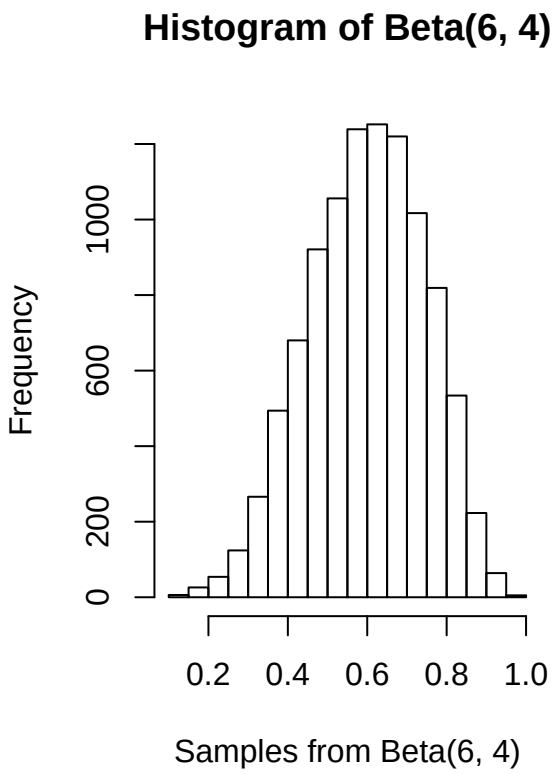
```
## Warning in ks.test(chain2[-(1:burnIn), 1], test2): p-value will be
## approximate in the presence of ties
```

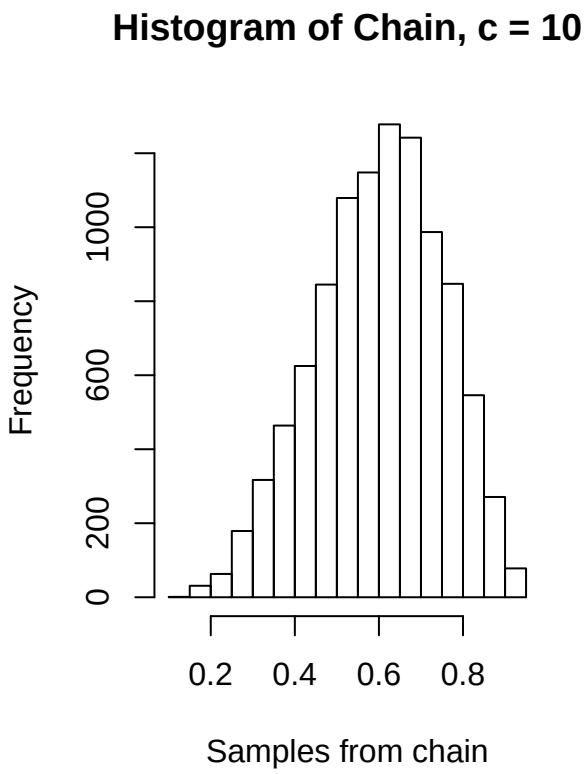
```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  chain2[-(1:burnIn), 1] and test2
## D = 0.018106, p-value = 0.3854
## alternative hypothesis: two-sided
```

```
# Performance of the sampler: c = 10
par(mfrow=c(1,3))
plot(chain3, type='l', main = "Trace plot of Chain, c = 10")
abline(h=0.6, col="red")
acf(chain3, main = "Autocorrelation plot of Chain, c = 10")
hist(chain3, xlab = "Samples from chain", main = "Histogram of Chain, c = 10")
```

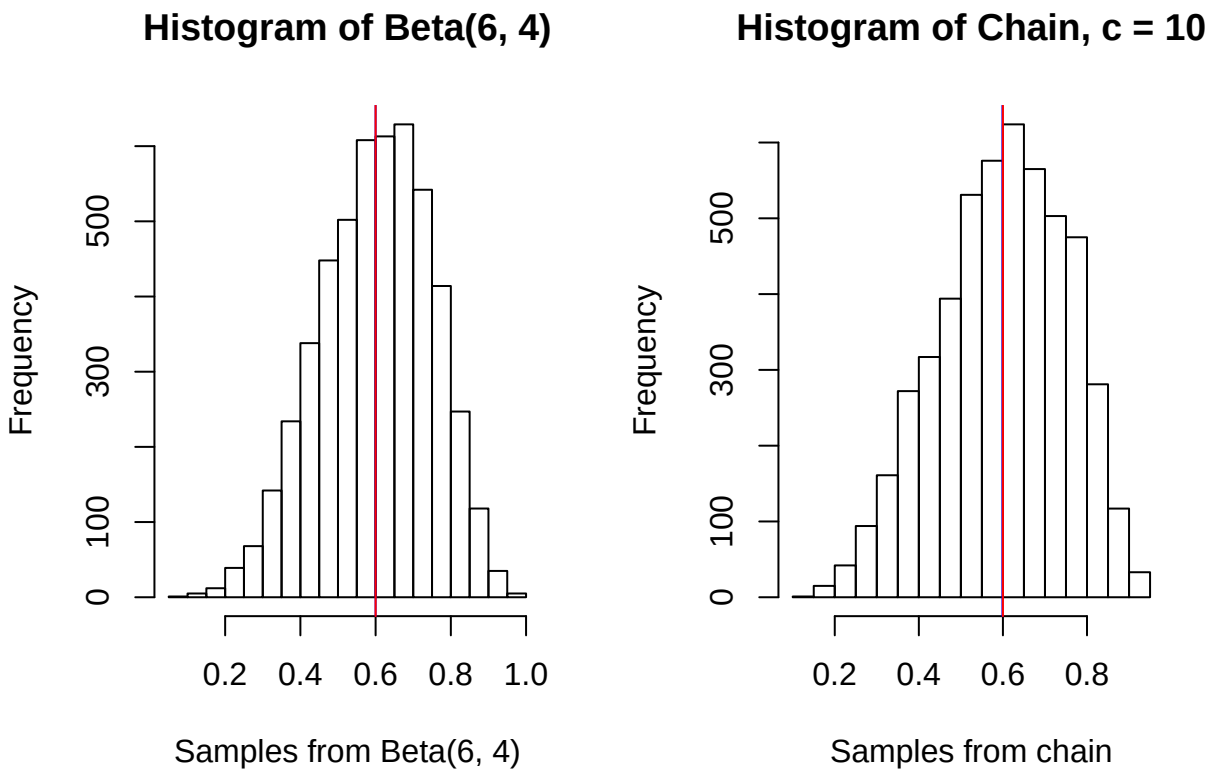



```r
# Comparing the histogram of the chain with the target distribution Beta(6,4)
par(mfrow=c(1,2))
hist(test, xlab = "Samples from Beta(6, 4)", main = "Histogram of Beta(6, 4)")
hist(chain3, xlab = "Samples from chain", main = "Histogram of Chain, c = 10")
```






```r
# Histogram of the target distribution Beta(6,4) vs. the Chain after burnIn
hist(test2, xlab = "Samples from Beta(6, 4)", main = "Histogram of Beta(6, 4)")
abline(v=mean(test2), col="blue")
abline(v=0.6, col="red")
hist(chain3[-(1:burnIn),1], xlab = "Samples from chain", main = "Histogram of Chain, c = 10")
abline(v = mean(chain3[-(1:burnIn),1]), col="blue")
abline(v=0.6, col="red")
```




```
# Kolmogorov-Smirnov statistic
ks.test(chain3[-(1:burnIn),1], test2)
```

```
## Warning in ks.test(chain3[-(1:burnIn), 1], test2): p-value will be
## approximate in the presence of ties
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  chain3[-(1:burnIn), 1] and test2
## D = 0.024354, p-value = 0.103
## alternative hypothesis: two-sided
```