Question 1

Wenjing Xu, Yanfei Zhou, Yijia Zhao

Question 1 Metropolis-Hastings

1.

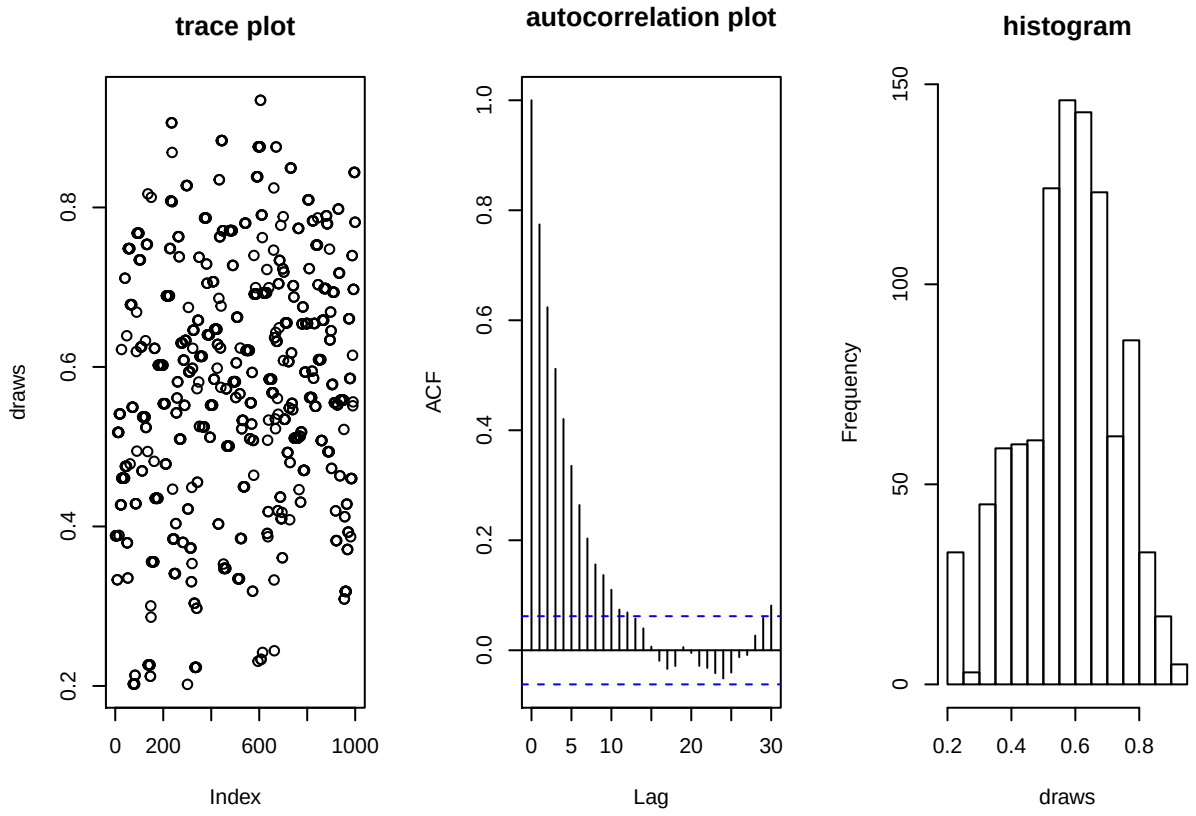
The code for Metropolis-Hastings algorithm is shown as follows.

```
Metropolis_Hastings_sampler <- function(c,chain_length,start_value){
  chain <- numeric(chain_length)
  accept_ratio <- 0
  for (i in 1:chain_length){
    proposed <- rbeta(1,c*start_value,c*(1-start_value))
    accept_ratio <- dbeta(proposed,6,4)*dbeta(start_value,c*proposed,c*(1-proposed))/
      (dbeta(start_value,6,4)*dbeta(proposed,c*start_value,c*(1-start_value)))
    temp <- runif(1,0,1)
    if (temp < accept_ratio){
      chain[i] <- proposed
      start_value <- proposed
    }else{
      chain[i] <- start_value
    }
  }
  return(chain)
}
c <- 1
iterations <- 1000
start <- runif(1)
chain <- Metropolis_Hastings_sampler(c,iterations,start)
```

2.

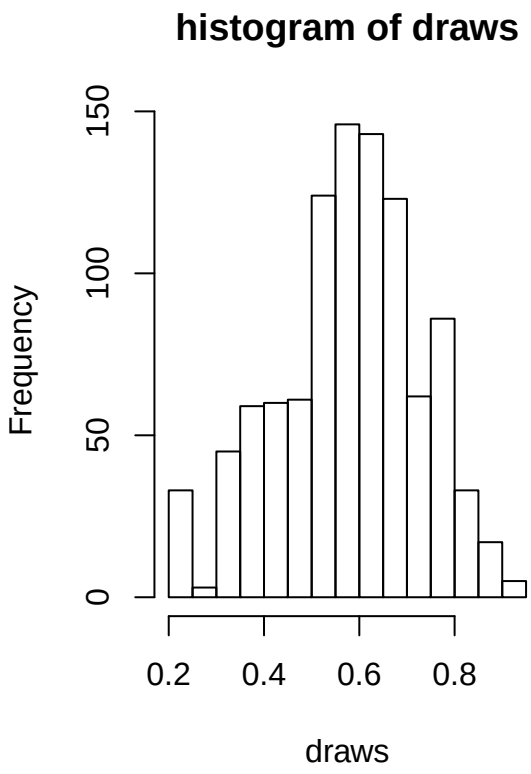
Now we use the following commands to evaluate the performace of this sampler.

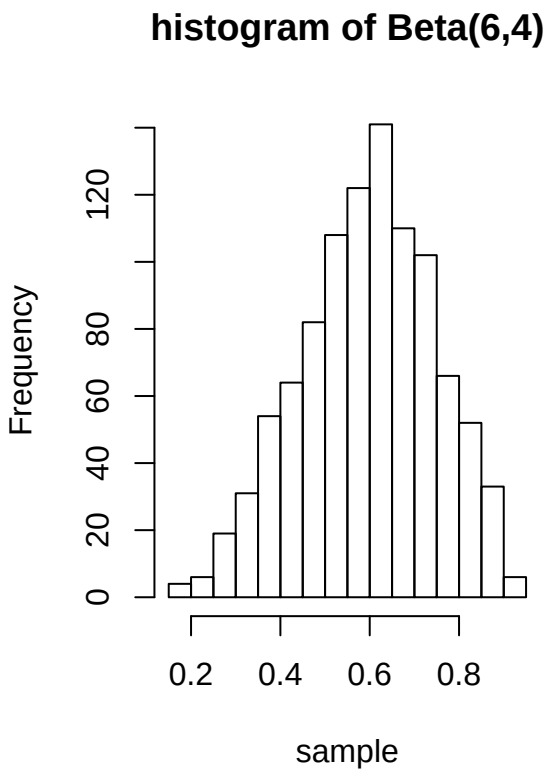
```
par(mfrow=c(1,3))
draws <- chain
plot(draws,main = "trace plot")
acf(draws,main = "autocorrelation plot")
hist(draws, main = "histogram")
```



Compare the histogram of draws with target distribution of Beta(6,4).

```
par(mfrow=c(1,2))
hist(draws, main = "histogram of draws",nclass = 15)
sample <- rbeta(iterations,6,4)
hist(sample, main = "histogram of Beta(6,4)",nclass = 15)
```





Then we calculate the Kolmogorov-Smirnov statistic using ks.test().

```
ks.test(draws,"pbeta",6,4)
```

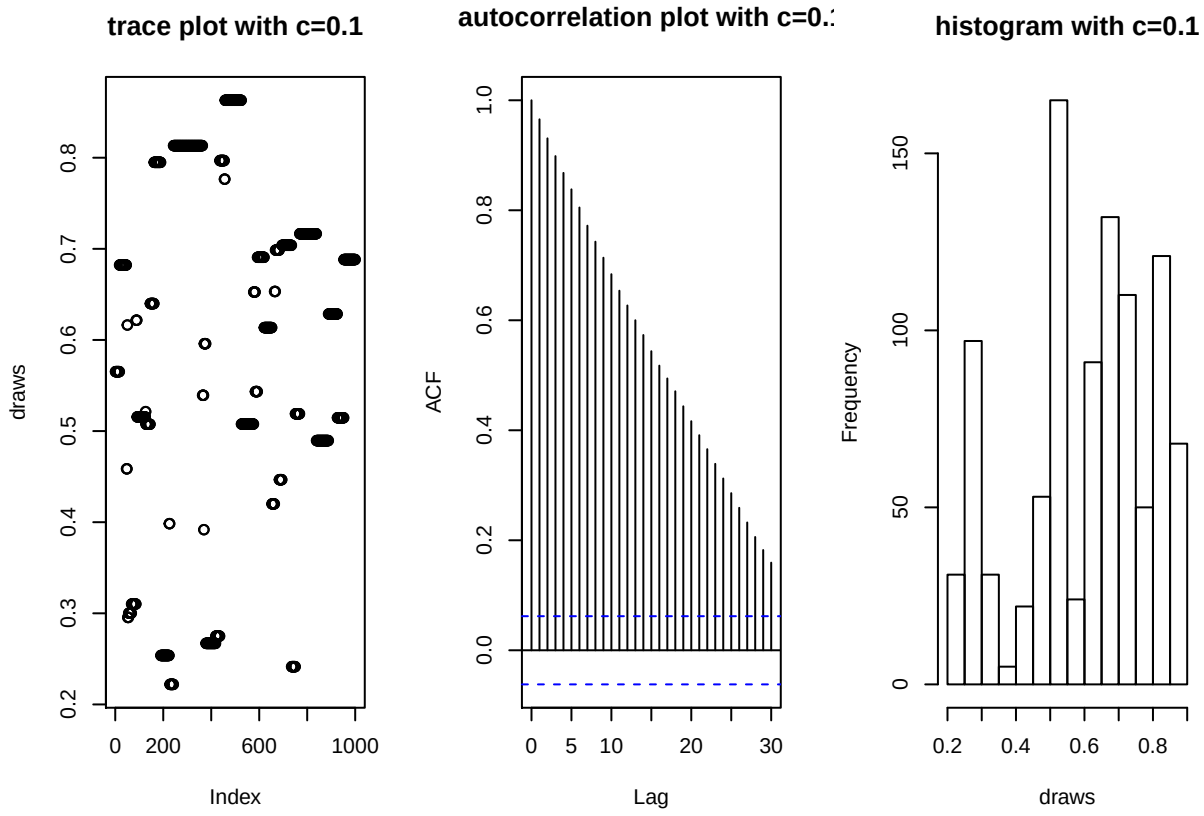
```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.079769, p-value = 5.945e-06
## alternative hypothesis: two-sided
```

3.

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

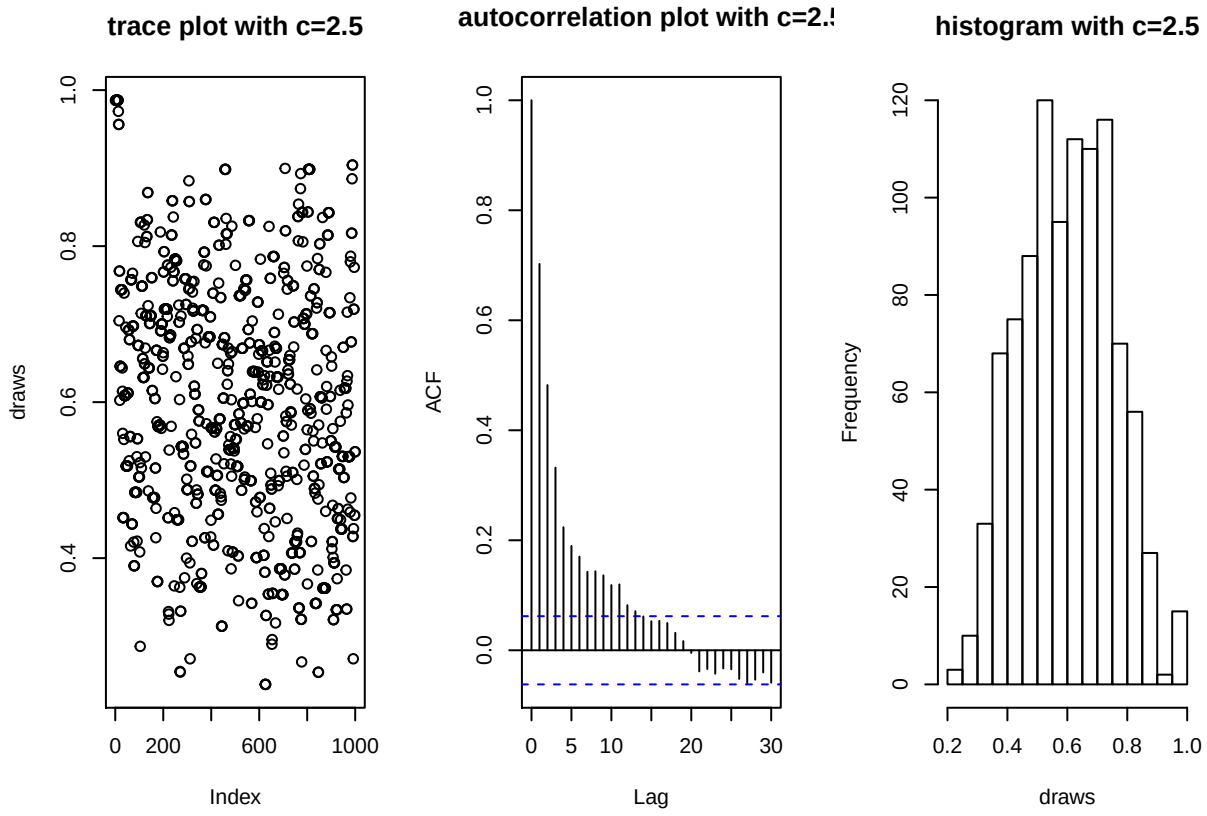
```
c <- 0.1
start <- runif(1)
draws <- Metropolis_Hastings_sampler(c,iterations,start)
par(mfrow=c(1,3))
plot(draws,main = "trace plot with c=0.1")
acf(draws,main = "autocorrelation plot with c=0.1")
hist(draws, main = "histogram with c=0.1")
```




```
print(1-mean(duplicated(draws)))
```

```
## [1] 0.042
```

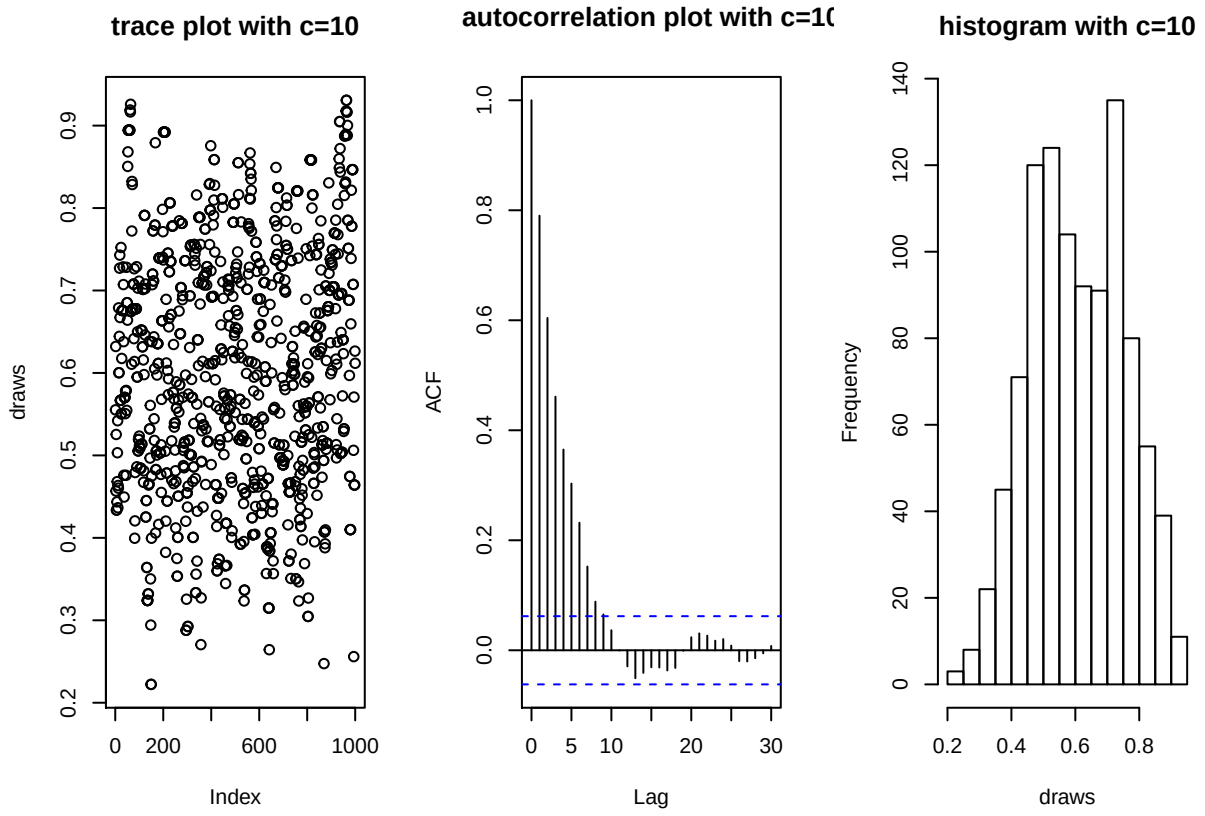
```
c <- 2.5
start <- runif(1)
draws <- Metropolis_Hastings_sampler(c,iterations,start)
par(mfrow=c(1,3))
plot(draws,main = "trace plot with c=2.5")
acf(draws,main = "autocorrelation plot with c=2.5")
hist(draws, main = "histogram with c=2.5")
```




```
print(1-mean(duplicated(draws)))
```

```
## [1] 0.434
```

```
c <- 10
start <- runif(1)
draws <- Metropolis_Hastings_sampler(c,iterations,start)
par(mfrow=c(1,3))
plot(draws,main = "trace plot with c=10")
acf(draws,main = "autocorrelation plot with c=10")
hist(draws, main = "histogram with c=10")
```




```
print(1-mean(duplicated(draws)))
```

```
## [1] 0.652
```

The higher rejection rate is, the less efficient the sampler will be. So the sampler with c=10 is most efficient. Also, the number of draws needed from the sampler before thinning and burn-in is smallest when c=10. We can see this from the autocorrelation plot. Last but not least, the histogram when c=10 is also closest to our target distribution.