Final 37810

Shiting Zhu, Lijing Wang, Jingting Li

November 1, 2015

Question 1

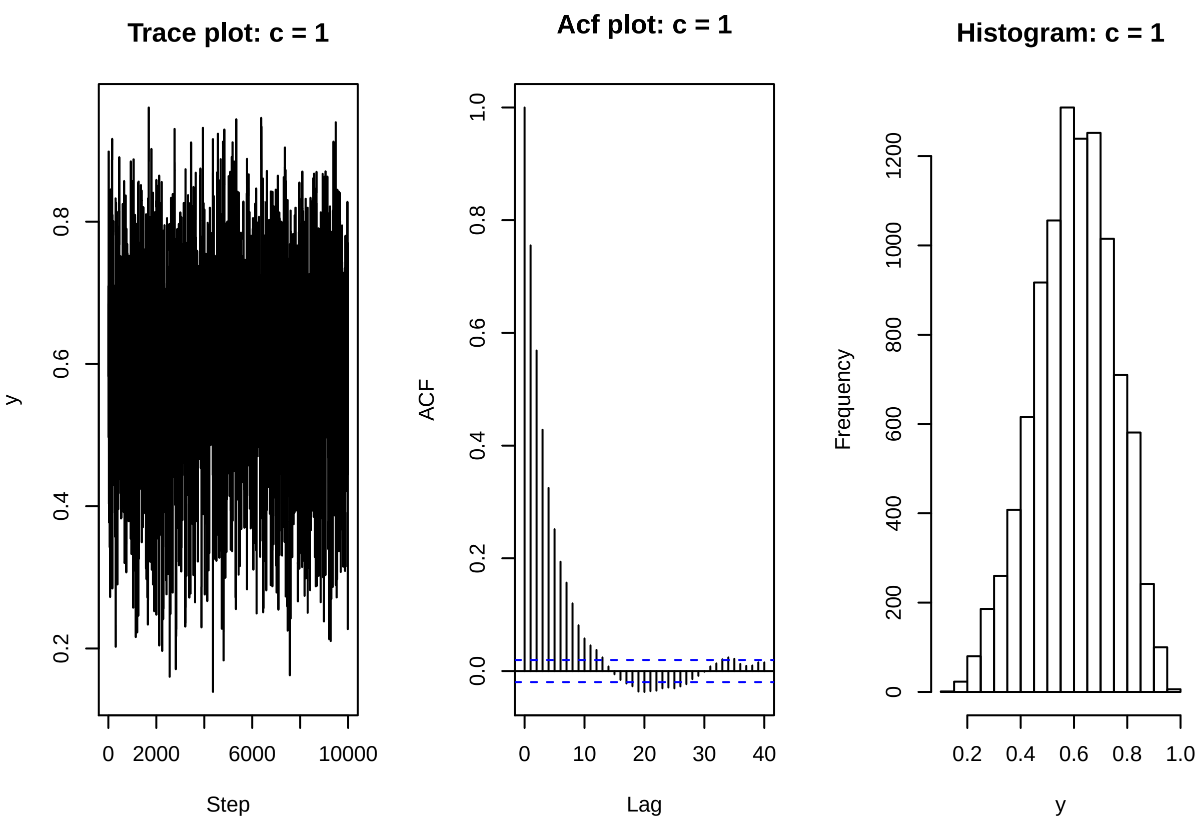
Step 3: Sample a number from uniform(0,1).If it is more than A(ϕ_n->ϕ_{new}), we accept ϕ_{new} and set $\phi_{n+1} = \phi_{new}$. Otherwise, $\phi_{n+1} = \phi_n$.

Step 4 : Repeat Step 1 to Step 3 until we get length n chain.

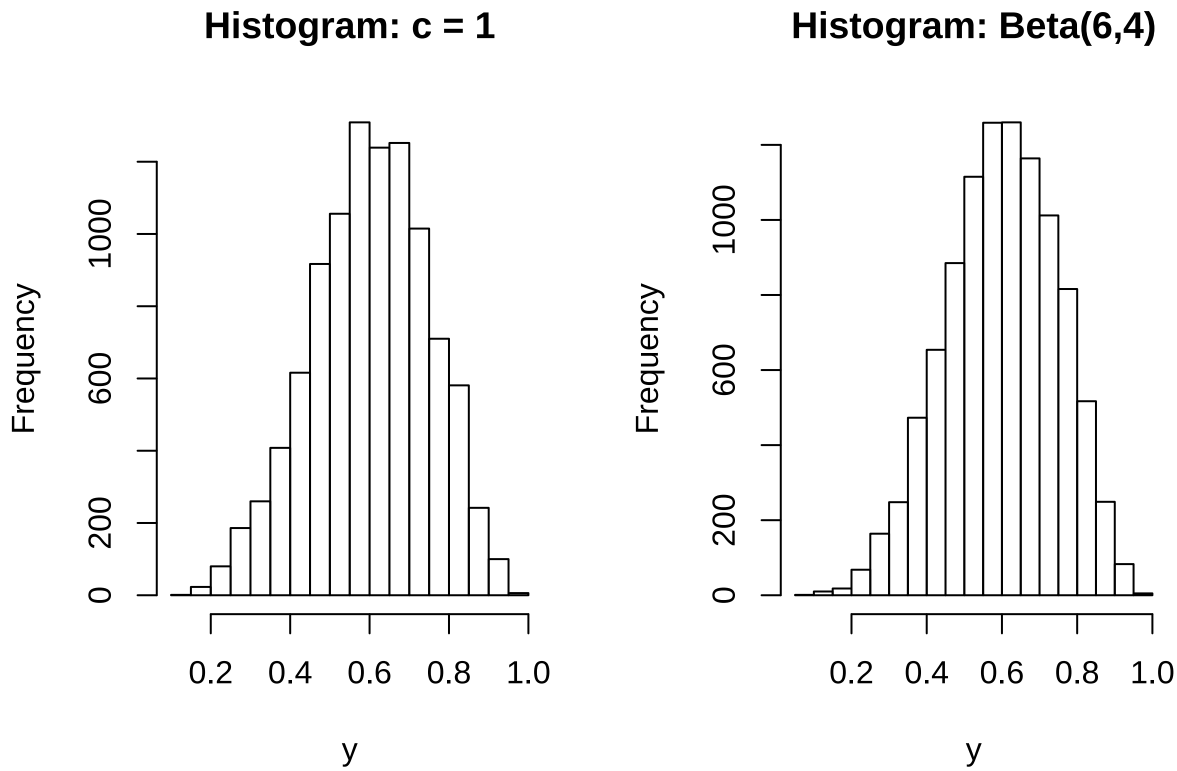
```
library(coda)
Alpha <- 6
Beta <- 4

# Use the proposal function to get new phi, noticeing phi cannot be 0 or 1
phi <- function(c,oldphi){
  newphi = 0
  # Making sure new phi is neither 0 nor 1
  while (newphi == 0 || newphi == 1){
  newphi = rbeta(1,c*oldphi,c*(1-oldphi))
  }
  return(newphi)
}

# Using similar strcuture of assignment 3 to build MCMC chain
run_metropolis_MCMC <- function(startvalue,c,iterations){
  # set chain
  chain = rep(0, iterations+1)
  chain[1] = startvalue
  for (i in 1:iterations){
    phi = phi(c,chain[i])
    posterior = dbeta(phi,Alpha,Beta)/dbeta(chain[i],Alpha,Beta)
    proposal = dbeta(phi,c*chain[i],c*(1-chain[i]))/dbeta(chain[i],c*phi,c*(1-phi))
    probab = min(1,posterior/proposal)
    # accept new value if random number uniform (0,1) is less than
    # acceptance probability
    if (runif(1) < probab){
      chain[i+1] = phi
    # reject new value if random number uniform (0,1) is greater
    # or equal than acceptance probability
    }else{
      chain[i+1] = chain[i]
    }
  }
  return(chain)
}

startvalue = runif(1)
chain = run_metropolis_MCMC(startvalue,1, 10000)
acceptance = 1-mean(duplicated(chain))
```

2. Based on the plots, we think the performance of the sampler when c = 1 is ok. The autocorrection is good and the Kolmogorov-Smirnov Static is showing good result.

```
par(mfrow=c(1,3))   #1 row, 3 columns

  traceplot(as.mcmc(chain), type="l", main = "Trace plot: c = 1", xlab="Step", ylab="y")
  acf(chain, main = "Acf plot: c = 1")
  hist(chain, main = "Histogram: c = 1", xlab="y")
```




```
  target = rbeta(10001,6,4)
 par(mfrow=c(1,2))
  hist(chain, main = "Histogram: c = 1", xlab="y")
  hist(target, main = "Histogram: Beta(6,4)", xlab="y")
```



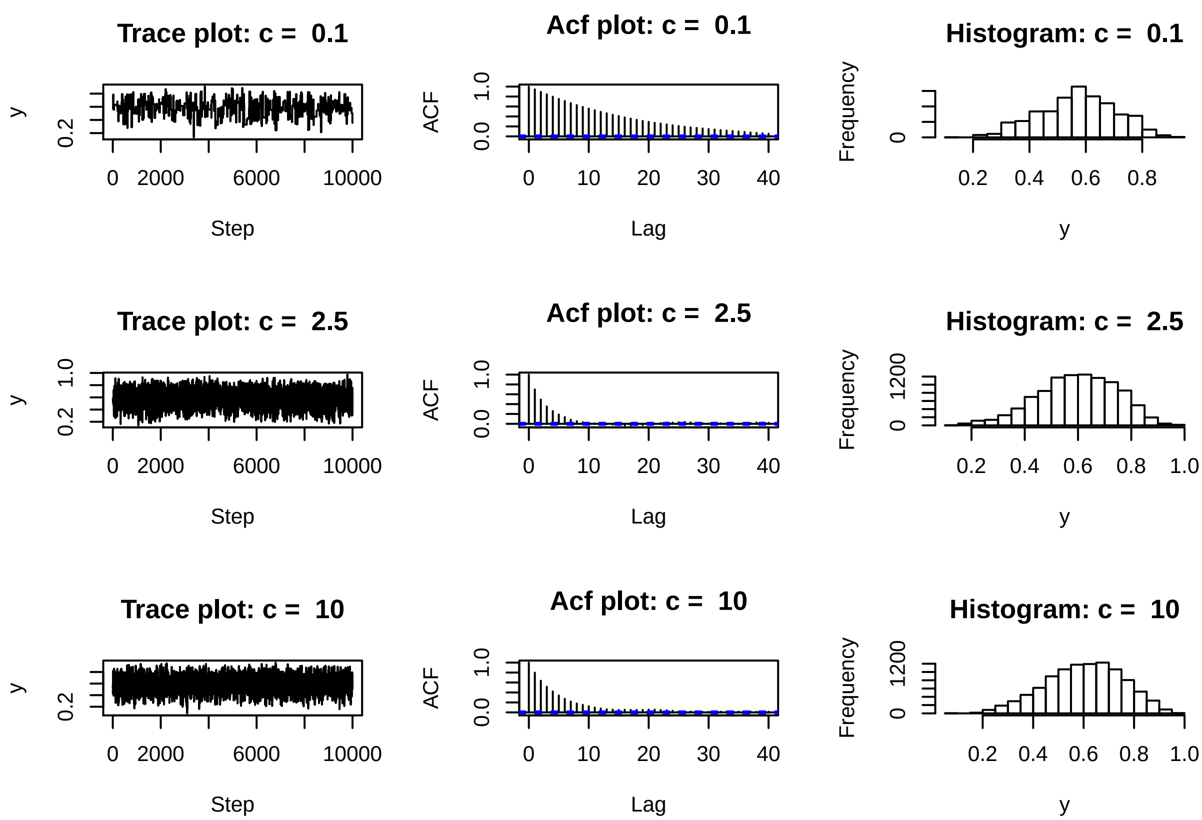

```
ks.test(chain,target)
```

```
## Warning in ks.test(chain, target): p-value will be approximate in the
## presence of ties

##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  chain and target
## D = 0.012899, p-value = 0.3762
## alternative hypothesis: two-sided
```

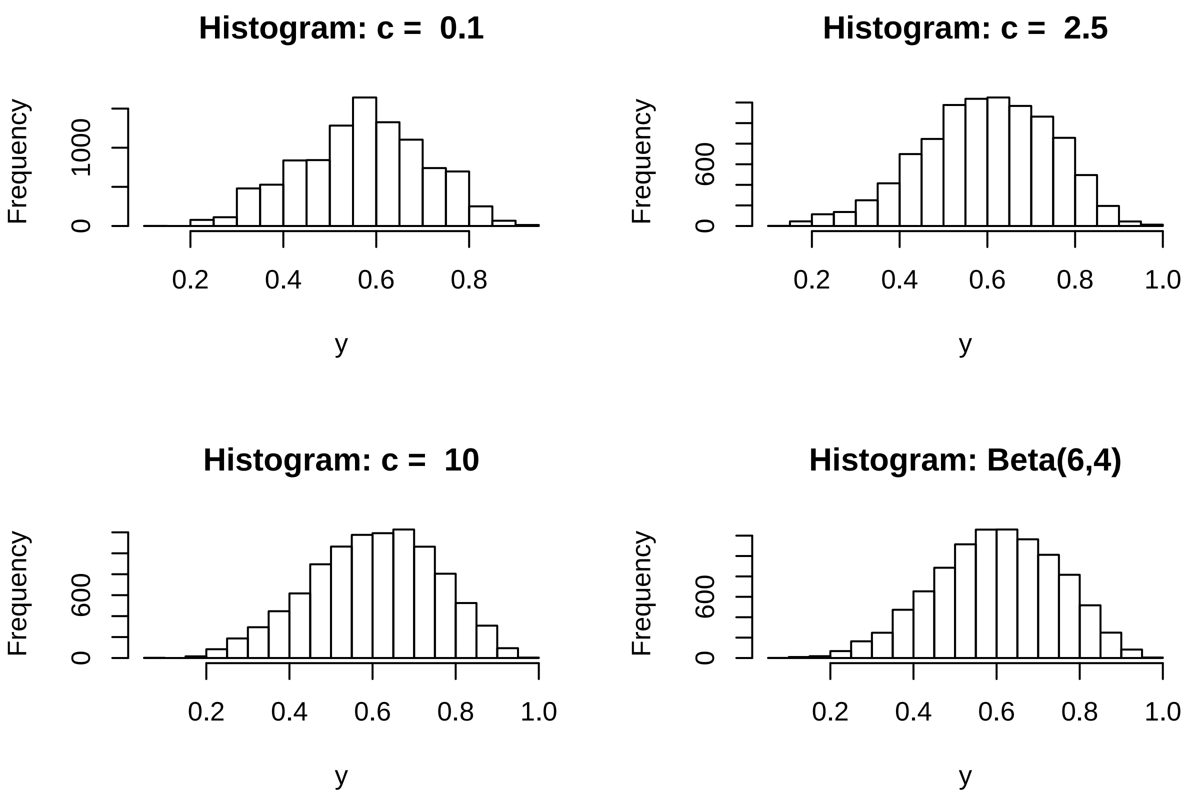
3. It is clear from the result that c = 2.5 gives the best result. Based on the autocorrection plots, c = 0.1 is pretty bad since it requires serious correction. c = 10 is much better but still a little bit worse than c = 2.5. The comparison of histograms of all c and the beta(6,4) also supports our conclusion.

```
Cs <- c(0.1, 2.5, 10)
  # calculate MCMC chian for all the c
result <- sapply(Cs, run_metropolis_MCMC, startvalue = startvalue,iterations = 10000)

# set a graph function which will automatically have the trace plot,
# cf plot and histograms of all the c in vector Cs
graph <- function(Cs){
  n = length(Cs)
  par(mfrow = c(n, 3))
  for( i in 1:n ) {
  traceplot(as.mcmc(result[,i]), type="l", main = paste("Trace plot: c = ", Cs[i]), xlab="Step", ylab="
  acf(result[,i], main = paste("Acf plot: c = ", Cs[i]))
```

```
  hist(result[,i], main = paste("Histogram: c = ", Cs[i]), xlab="y")
  }
}

graph(Cs)
```




```
par(mfrow=c(2,2))
for( i in 1:length(Cs) ) {
hist(result[,i], main = paste("Histogram: c = ", Cs[i]), xlab="y")
}
hist(target, main = "Histogram: Beta(6,4)", xlab="y")
```




```
ks.test(result[,1],target)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  result[, 1] and target
## D = 0.10939, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(result[,2],target)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  result[, 2] and target
## D = 0.013699, p-value = 0.3051
## alternative hypothesis: two-sided
```

```
ks.test(result[,3],target)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  result[, 3] and target
## D = 0.021598, p-value = 0.01884
## alternative hypothesis: two-sided
```

Question 2

Gibbs Sampling

```
Gibbs_<-function(x0,y0,iterations,B=5,burnIn=0)
## 5 parameters are needed in this function, with B=5 and burnIn=0 by default
{
  if(x0>0&&x0<B&&y0>0&&y0<B)
## to check whether the starting values are in the domain.
  {
    x<-c(x0,rep(NA,iterations-1))
## Initialize the Markov chain
    y<-c(y0,rep(NA,iterations-1))
## Initialize the Markov chain
    for(i in 1:(iterations-1))
    {
      x[i+1]<-(-log(1-runif(1)*(1-exp(-y[i]*B)))/y[i])
## use inverse transform sampling to draw sample from conditional distribution
## p(x^{i+1}|y^{i})
      y[i+1]<-(-log(1-runif(1)*(1-exp(-x[i+1]*B)))/x[i+1])
## use inverse transform sampling to draw sample from conditional distribution
## p(y^{i+1}|x^{i+1})
    }
    if(burnIn>0)
    {
        x<-x[-(1:burnIn)]
        y<-y[-(1:burnIn)]
        print(length(x))
## discard the first bunch of draws for the burn-in process
    }
    return(data.frame(x,y))
  }
 else
   stop("Initial values incorrect")
## print the information for incorrect starting values
}
```

To estimate the marginal distribution generated from the conditional distributions using Gibbs sampling, we first pick up the starting values for X and Y. Denote them as `x0` and `y0`. `iterations` stands for the number of draws we want to get from Gibbs sampling. `B` is the number given in the exercise, which is 5 here. `burnIn` is the number of draws we want to discard for the burn-in process, the default for `burnIn` is 0.

In this case, both the conditional distribution of $X|Y$ and $Y|X$ are truncated exponential distribution. The domains of both conditional pdf's are $[0, B]$. So the starting values should satisfy $0 <$`x0`$< B$ and $0 <$`y0`$< B$. Therefore I put a restriction `if(x0>0&&x0<B&&y0>0&&y0<B)` on the input of starting values to check if they satisfy the requirement.

Since

$$p(x|y) \propto ye^{-yx}, 0 < x < B$$

$$p(y|x) \propto xe^{-yx}, 0 < y < B$$

Consider $p(x|y)$ only. To get samples using inverse transform sampling, first we need to figure out the normalizing constant for the conditional pdf and then the inverse function of cdf.

The normalizing constant can be calculated using the formula $c = \frac{1}{\int f(x|y)\,\mathrm{d}x}$, if $p(x|y) \propto f(x|y)$. In this case, $f(x|y) = ye^{-yx}, 0 < x < B$. Therefore $c = \frac{1}{\int_0^B ye^{-yx}\,\mathrm{d}x} = \frac{1}{1-e^{-By}}$.

Then we start to calculate the cdf $H(x|y)$ for $X|Y$.
We have

$$p(x|y) = \frac{ye^{-yx}}{1-e^{-By}}, 0 < x < B$$

Using the formula $F(x) = \int_{-\infty}^{x} p(z)\,\mathrm{d}z$, we get the cdf of $X|Y$:

$$F(x|y) = \frac{1-e^{-xy}}{1-e^{-By}}, 0 < x < B$$

Then we can write down the inverse function of $F(x|y)$, Denote as $F^{-1}(u|y), where 0 \leq u \leq 1$.
We have:

$$F^{-1}(u|y) = -\frac{log(1-u(1-e^{-By}))}{y}, 0 \leq u \leq 1$$

According to inverse transform sampling, to draw a sample from $p(x|y)$, we first generate u from $Unif[0,1]$, then $x = F^{-1}(u|y)$ is from the conditional distribution of X|Y.

We draw samples from $p(y|x)$ using the same method.

Now we start generating samples from p(x,y).

Here I start Gibbs sampling with `x0` and `y0`, draw a value $x^{(1)}$ from the full conditional $p(x|$ `y0` $)$ using inverse transform sampling. Then use the updated $x^{(1)}$ to draw a sample from $p(y|x^{(1)})$.

To get more samples using Gibbs sampling, continually use the most updated values of x and y when generating samples from conditional distribution. To be more specific, generate $x^{(i+1)}$ from $p(x|y^{(i)})$ and $y^{(i+1)}$ from $p(y|x^{(i+1)})$. Repeat it for $n =$ `iterations` times.

For the burn-in process, discard the first $m =$ `burnIn` samples from the Markov chain to reduce the influence of starting values on the Markov chain.