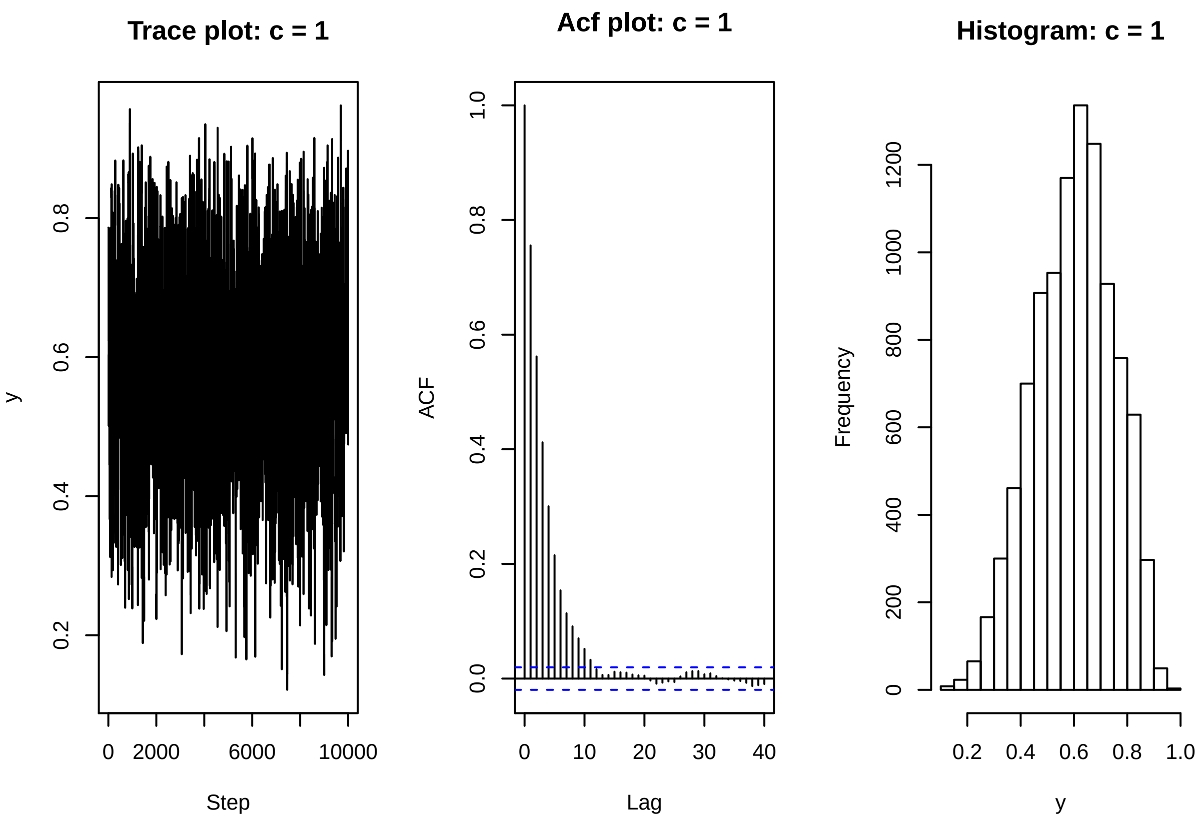
Final 37810

Shiting Zhu, Lijing Wang, Jingting Li

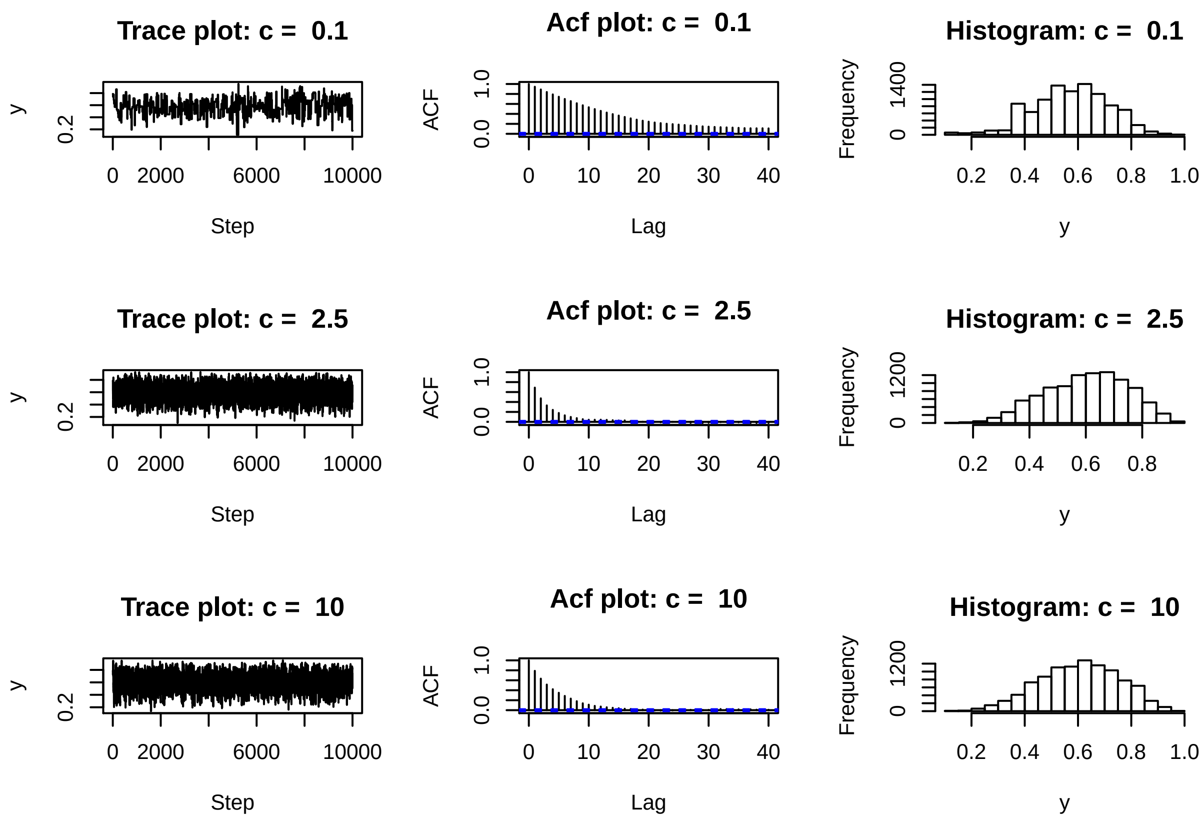
November 1, 2015

Question 1

```r
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)
}

#
run_metropolis_MCMC <- function(startvalue,c,iterations){
  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 = posterior/proposal
    if (runif(1) < probab){
      chain[i+1] = phi
    }else{
      chain[i+1] = chain[i]
    }
  }
  return(chain)
}

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

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")
```




```
Cs <- c(0.1, 2.5, 10)
result <- sapply(Cs, run_metropolis_MCMC, startvalue = startvalue,iterations = 10000)


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)
```



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 $c = \frac{1}{\int_0^B ye^{-yx}\,\mathrm{d}x} = \frac{1}{1-e^{-By}}$.