**EEB 581, Problem Set Nine**

**Solutions**

Recall the joint density given in Example 4 of the MCMC notes,

$$p(x,y) = \frac{n!}{(n-x)!\,x!}\, y^{x+\alpha-1}\,(1-y)^{n-x+\beta-1}$$

where we will take $\alpha = 1$, $\beta = 2$, and $n = 10$.

For this distribution, construct a Gibbs sample (Example 4 gives the marginal distributions). After burning in the sampler for 100 interations, generate a vector of 5000 draws of $(x,y)$ pairs from this distribution. From this sampler estimate $E(x)$, $E(y)$, $\sigma^2(x)$, $\sigma^2(y)$ and $\sigma(x,y)$.

You might want to look over the **R** notes on Metropolis-Hastings sampler.

**Solution**

Recall from Example 4 in the MCMC notes that the conditional distributions are given by

$$x\,|\,y \sim \text{Bin}(n,y) = \text{Bin}(10,y)$$

In **R**, we generate a single draw from a Bin(10,$y$) with the command **`rbinom(1,10,y)`**

$$y\,|\,x \sim \text{Beta}(x+\alpha, n-x+\beta) = \text{Beta}(x+1, 12-x)$$

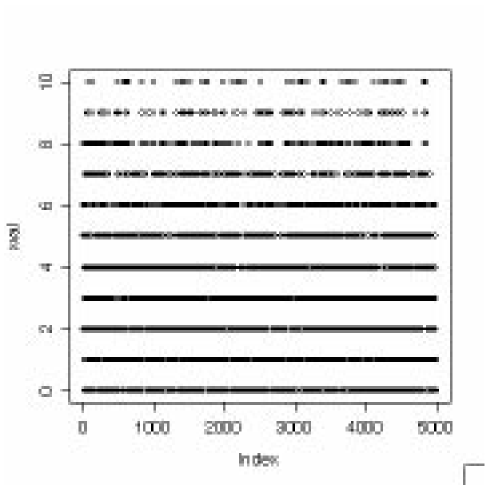
In **R**, we generate a single draw from a Beta$(x+1, 12-x)$ with the command **`rbeta(1,x+1,12-x)`**

The resulting **R** code for 5000 samples after a burn-in of 100 is as follows: (recall that **#** denotes a line being comment. These are included simply for clarity

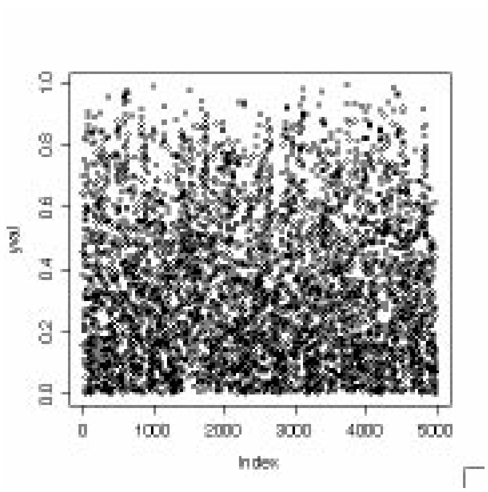
```
# create vectors of lenght 5000 for the x and y values
> xval <- numeric(5000)
> yval <- numeric(5000)
# Declare the intial X value.
> xint <- 5
# sample 100 draws for the burn-in
> xlast <- xval
> for (i in 1:100) {
    ylast <- rbeta(1,xlast+1,12-xlast)
    xlast <- rbinom(1,10,ylast) }
# Now sample and store 5000 values
> for (i in 1:5000) {
    xval[i] <- xlast
    ylast <- rbeta(1,xlast+1,12-xlast)
    yval[i] <- ylast
    xlast <- rbinom(1,10,ylast) }
```

Let's look at the time series traces for both x and y,

**`plot(xval)`**




**`plot(xval)`**




Both samplers this appear to be mixxing well.

We could also use **`hist(xval)`** (or **`hist(xval)`**) to look at the marginal distributions of $x$ and $y$, if so desired.

# compute E(x)

> mean(xval)

[1]   3.355

# compute E(y)

> mean(yval)

[1]   0.3354854

# compute Var(x)

> var(xval)

[1]   7.280031

```
# compute Var(y)
> var(yval)
[1]    0.05620134

# compute Cov(x,y)
> cov(xval, yval)
[1]    0.5622495
```