Statistical Computing HW 9

Mario Ibanez

April 9, 2016

Question

Suppose that the random variable X and Y both take on values in the interval $(0, B)$. Suppose that the conditional density of X given that $Y = y$ is:

$$f(x|y) = c(y)e^{-xy} \ 0 < x < B$$

and the conditional density of Y given that $X=x:

$$f(y|x) = c(x)e^{-xy} \ 0 < y < B$$

Give a method for approximately simulating the vector X, Y. Run a simulation to estimate (a) $E(X)$ and (b) $E(XY)$.

Answer

Gibbs sampling can be used to simulate the vector X, Y. First, in order for the respective integral to equal one, it must be the case that:

$$c(x) = \frac{x}{1 - e^{-Bx}}$$

and

$$c(y) = \frac{y}{1 - e^{-By}}$$

Rewriting the conditional distributions:

$$f(x|y) = \frac{ye^{-yx}}{1 - e^{-By}}$$

and

$$f(y|x) = \frac{xe^{-xy}}{1 - e^{-Bx}}$$

It is easy to see that these are each exponential distributions conditioned on the values being less than B.

For this assignment, let's let $B = 1000$. The task is to generate values (x, y) from the joint distribution and to estimate $E(X)$ and $E(XY)$. First, let's generate values from the joint distribution. To generate these values from $f(x|y)$ and $f(y|x)$, the inverse transform method will be used. With y and B given, random values from $f(x|y)$ can be generated using the following equation:

$$X = \frac{-ln\left(1 - U * (1 - e^{-By})\right)}{y}$$

where U is a standard uniform random variable. Similarly, to generate values from $f(y|x)$, the following equation is used:

$$Y = \frac{-ln\left(1 - U * (1 - e^{-Bx})\right)}{x}$$

The R functions below are used to calculate these values:

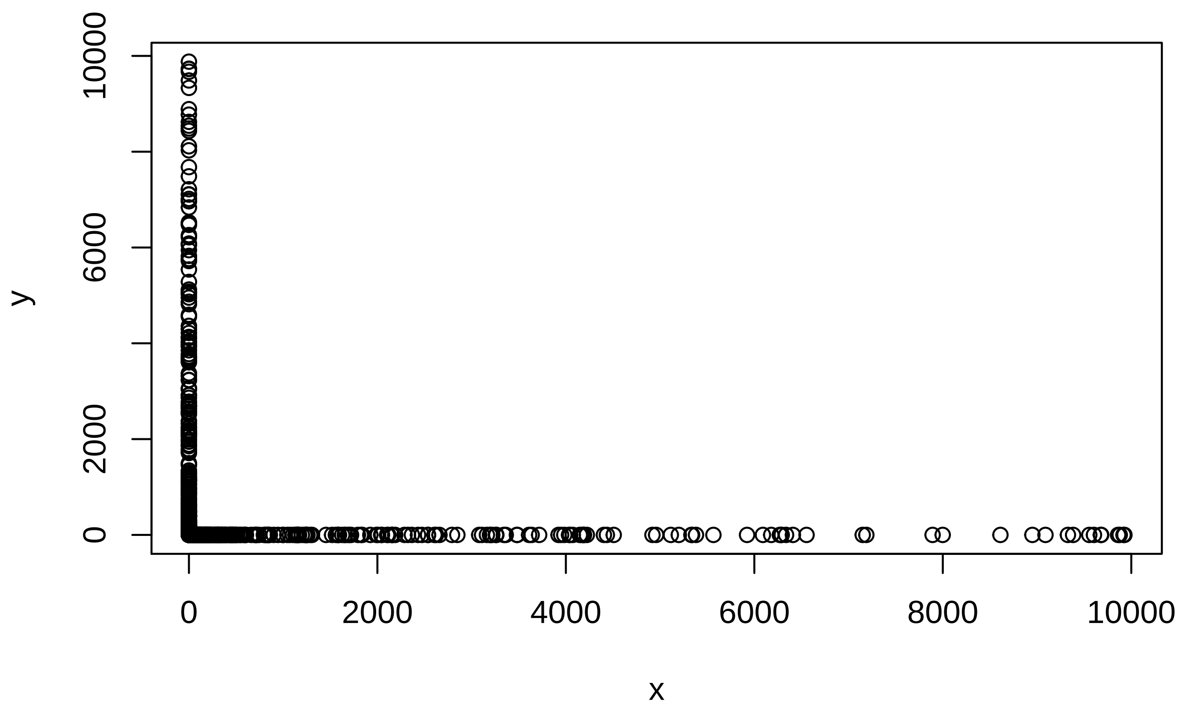
```
# Random values from f(x|y)
fx_given_y <- function(y, B){
  return( -log(1 - runif(1)*(1 - exp(-B*y))) / y)
}

# Random values from f(y|x)
fy_given_x <- function(x, B){
  return( -log(1 - runif(1)*(1 - exp(-B*x))) / x)
}
```

The algorithm begins with an initial value for one of the variables. Let $y_0 = 1$. Then x_0 is calculated be generating a random value from $f(x|y = 1)$. In general, y_i is a random number from $f(y|x = x_{i-1})$ and x_i is a random number from $f(x|y = y_i)$. In order to allow the initial value for y_0 to be "forgotten", the value sampled from the joint distribution will be equal to the 1000th term generated by the algorithm. In otherwords, (x, y) will be the value (x_{1000}, y_{1000}). This entire process is repeated each time we want to generate a single random value from $f(x, y)$. The code below generates 100 values from $f(x, y)$.

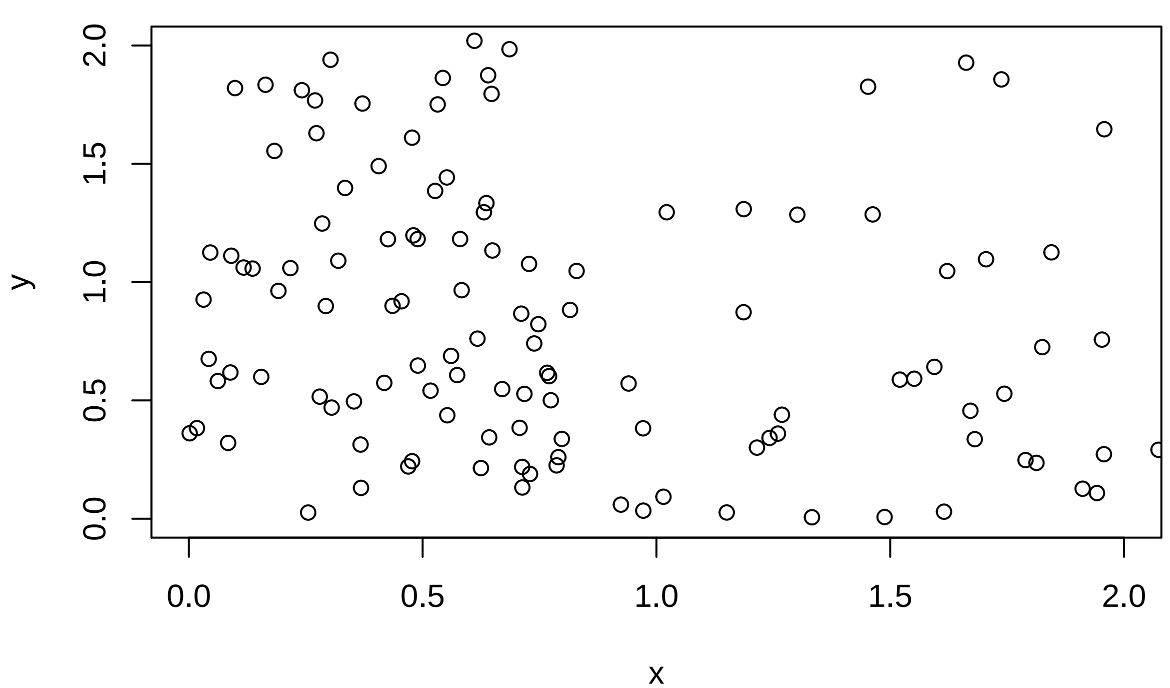
```
# B value
B <- 10000

# Total iterations of algorithm
N <- 1000

# store values
results <- data.frame(x = rep(0, N),
                      y = rep(0, N))

# Loop for results
for(n in 1:N){

  # Loop for generating a single value
  x_burn <- rep(0, 1000)
  y_burn <- rep(0, 1000)
  y_burn[1] <- 10
  x_burn[1] <- fx_given_y(y = y_burn[1], B = B)

  for(i in 2:1000){
    y_burn[i] <- fy_given_x(x = x_burn[i-1], B = B)
    x_burn[i] <- fx_given_y(y = y_burn[i], B = B)
  }

  # use 1000th value
```

```
  results[n, 1] <- x_burn[1000]
  results[n, 2] <- y_burn[1000]
}
```

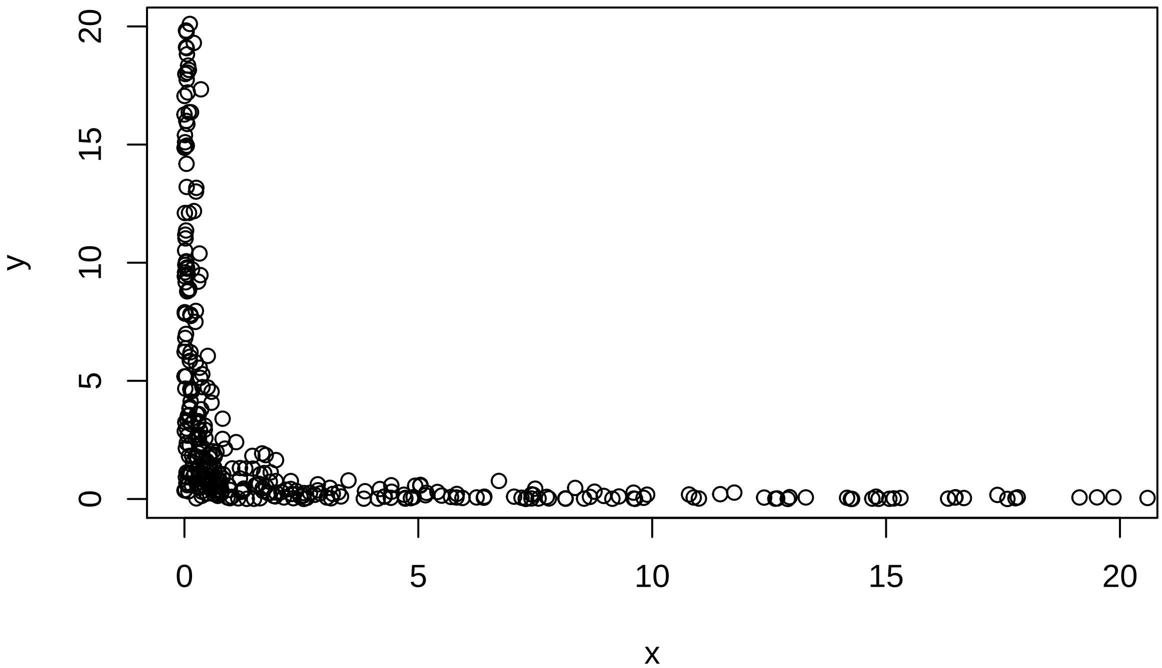
```
plot(y ~ x, data = results)
```



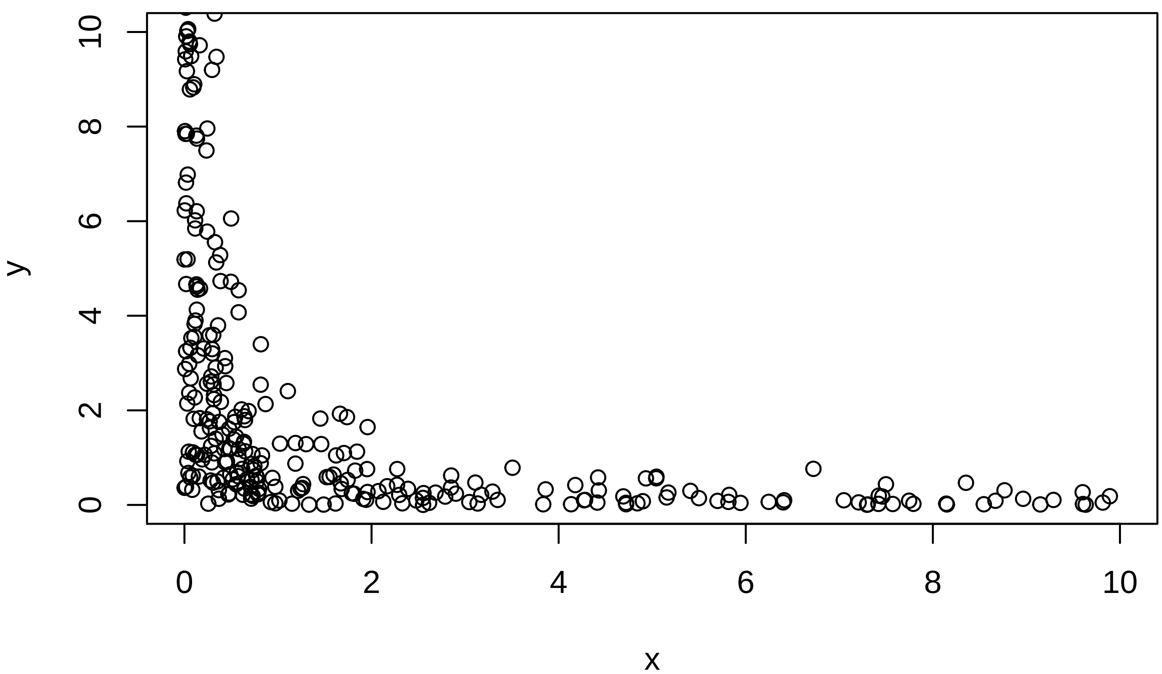

```
plot(y ~ x, data = results, xlim = c(0, 2), ylim = c(0, 2))
```



```r
plot(y ~ x, data = results, xlim = c(0, 20), ylim = c(0, 20))
```




```r
plot(y ~ x, data = results, xlim = c(0, 10), ylim = c(0, 10))
```



more details to be added later but this looks correct. for example as y goes to zero, f(x|y) becomes uniform on (0, B). that explains why there are so many values along the x and y axis.

the estimates are:

```r
mean(results$x)
```

```
## [1] 511.6413
```

```r
mean(results$y*results$x)
```

```
## [1] 0.938753
```