A short note on stochastic simulations: a glimpse on the rejection method

*R. Monteiro**

10/2/2018

Generating some random variables using inverse function method

To begin with, I would like to simulate a simple exponential variable. This is quite easy if we use what some people call the "inverse method": using the cumulative distribution function, we know that

$$F(x) = \int_0^x \lambda e^{-\lambda s} \mathrm{d}s = 1e^{\lambda x}. \tag{1}$$

Furthermore, we know that

$$X = F^{-1}(U)$$

is an exponential distribution, we only need to simulate a uniform on the interval [0,1]. Hence, we first generate an uniform random variable U

```
U = runif(1,min=0,max=1);
```

Now we use the inverse function in (1). It is not hard to show that

$$F^{-1}(u) = -\frac{1}{\lambda} log(1-u). \tag{2}$$

which can be simplified to

$$F^{-1}(u) = -\frac{1}{\lambda} log(u). \tag{3}$$

if you note that 1-U and U both have uniform distribution in $[0, 1]$. Ok, now we are ready to generate our exponential distribution:

```
generate_exponential <- function(lambda){
U = runif(1,min=0,max=1);
  X = -(1/lambda)*log(U);
return (X);  }
```

We are going to generate 1000 exponentially distributed random variables[1].

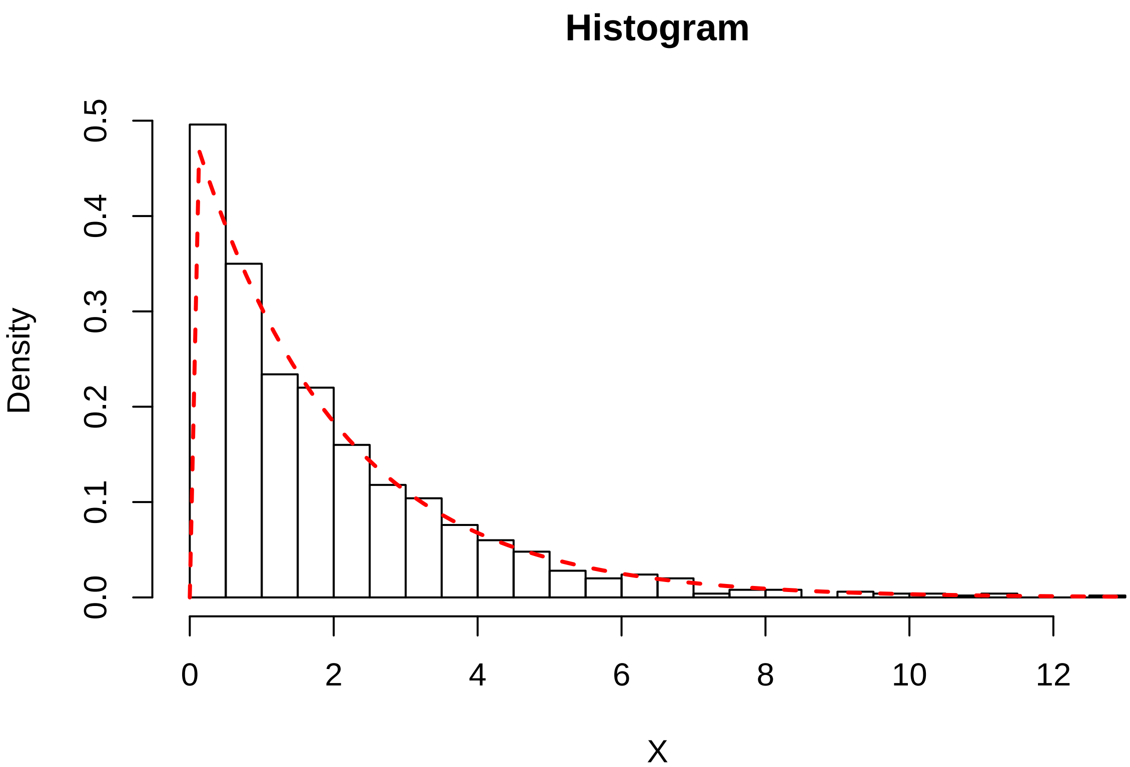
```
N = 1000;
X <- matrix(0,N,1);
lambda =.5;
for (i in 1:N ){
  X[i] = generate_exponential(lambda);
}
```

*Mathematics for Advanced Materials - Open Innovation Lab, Japan. Email: rafael.a.monteiro.math@gmail.com, monteirodasilva-rafael@aist.go.jp

[1] There is a simpler way to write this function without a loop. One could use for instance runif(1000,min=0,max=1)

Just for verification, I will plot a histogram with these values. Note how close to the density they are (a good sign,right?).

```
hist(X, breaks=25,main="Histogram", prob =TRUE)
curve(dexp(x,rate=lambda), add=TRUE, col = 2, lty = 2, lwd = 2)
```



A case that is not contemplated by the previous method

Now we shall consider a case that does not fit the previous technique. To begin with, imagine that you want to simulate a random variable X on the interval $[0, 1]$ that has the following density function:

$$f_X(x) = cx^2(1 - x)^2. \quad (4)$$

I'll first create the function f:

```
f <- function(x) {
  result= 30*x^2*(1-x)^2;
  return(result)
}
```

The constant c is so that $\int_0^1 f(s)\mathrm{d}s = 1$. This can be found using integration (or any symbolic package, like SAGE): in fact, $c = 30$.

```
var('x c')
f(x) = c*x^2*(1-x)^2
integral(f,x,0,1)
```

Good luck if you want to integrate and find a full expression for this inverse. You will probably end yo in a 3rd order polynomial, whose expression for roots will be nothing but horrible. But there is a way to circumvent that, which is the main reason for us to introduce the rejection method.

The idea is that, if you generate another distribution Y with density function $g(\cdot)$ and so that

$$\frac{f(x)}{g(x)} \leq M$$

(be careful, this is a pointwise bound! In other words, this is a severely tying/constraining both distributions X and Y!). In our case, $g(\cdot) = 1$, and it is not hard to show that $M = \frac{30}{16}$ (a crude bound would be $M = 30$, but smaller values are better (the reason why is that the acceptance rate of the method below behaves as a geometric distribution with rate $\frac{1}{M}$). Now we do the following:

- Step 1: Generate a random variable with distribution Y;
- Step 2: Generate an Uniform r.v. U;
- Step 3: If $U \leq \frac{f(Y)}{cg(Y)}$ then do X =Y, otherwise go back to step 1

Ok, now we are ready to go. Notice that we must simulate until we get a valid result[2]

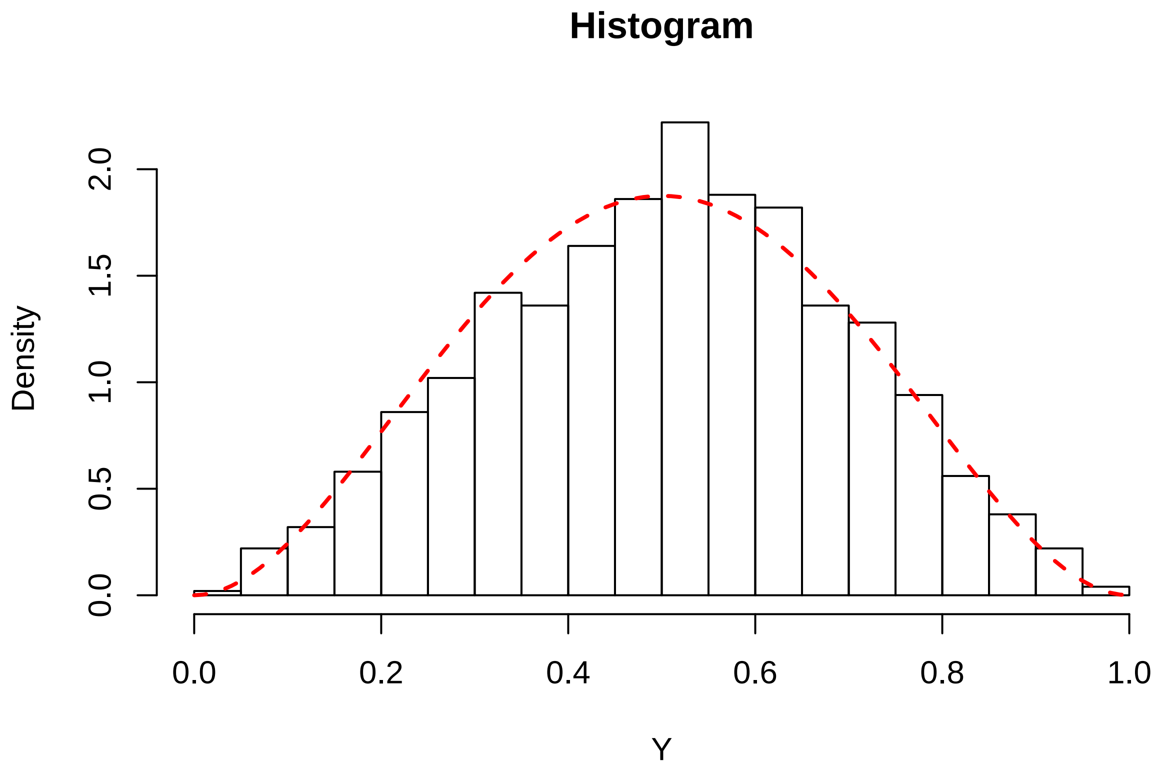
```
M = 30/16;
Y = matrix(0,N,1);
for (i in 1:N){
  while (TRUE){
    U = runif(2, min=0,max=1);
    if (U[2] <f(U[1])/M){
      Y[i]= U[1];
      break;
    }
  }

}
```

Just for curiosity, we can plot the histogram for Y. The fitting is stunning.

```
hist(Y, breaks=25,main="Histogram", prob =TRUE)
curve(f(x), add=TRUE, col = 2, lty = 2, lwd = 2)
```

[2] I will not explain it here, but the distribution of accepted simulated values also follow a probability distribution, called geometric distribution.


Histogram
Density
0.0
0.5
1.0
1.5
2.0
0.0
0.2
0.4
0.6
0.8
1.0
Y