Monte Carlo Simulation : Random Number Generators

A. Burnetas

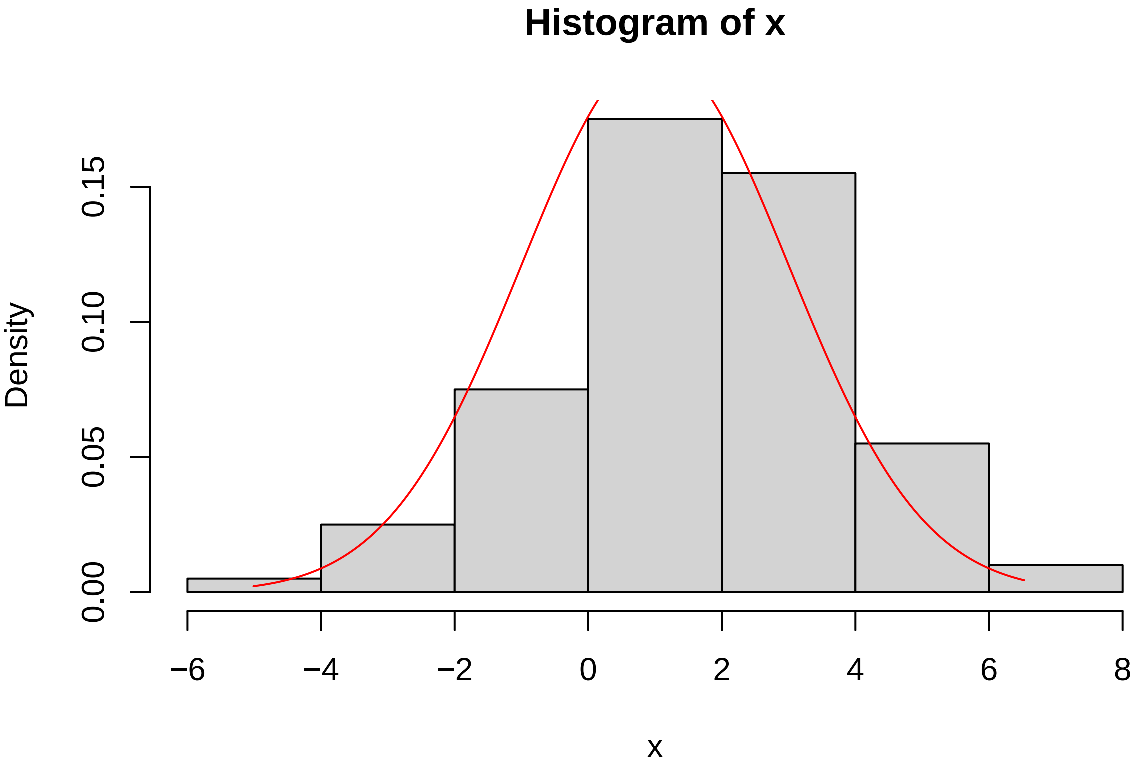
In this note we will create random number generators for several examples of continuous distributions. Before we start, it will be useful to create a function that generates and plots the empirical cumulative distribution function of a vector of random numbers. This function can be used to check how close the random numbers we create fit to the desired distribution.

First, note that if we want to see the approximation in terms of the probability density function, we can do this directly using the histogram of the vector, together with the plot of the density. For example, assume we have a sample of size 100 from the normal distribution $N(1, 2)$.

```
x=rnorm(100, 1,2)
```

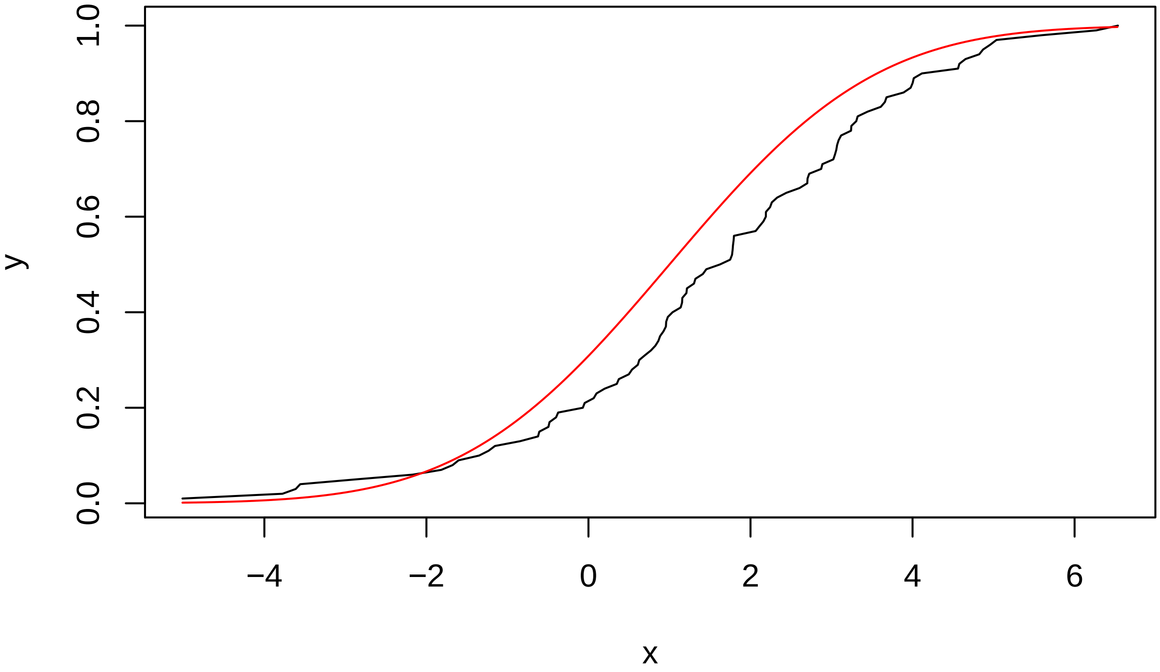
We can create a histogram of the sample using the option probability=TRUE to create relative frequencies. Then we plot the density of the normal distribution on the same graph. We create a sequence of points between the minimum and maximum of x to use for the second plot

```
hist(x, probability=T)
z=seq(min(x), max(x), 0.01)
lines(dnorm(z,1,2)~z, col="red")
```



To do the same with the cumulative distribution is not possible, because the hist function in R does not compute cumulative frequencies. However we can do this easily by programming it directly. The idea is that for a vector x of length n sorted in increasing order, the j-th element is the j/n-th empirical quantile (i.e., the proportion of vector elements that are less than or equal to x_j is equal to j/n, for $j = 1, \ldots, n$.) Therefore, the vector $y = (1/n, 2/n, \ldots, n/n)$ is the empirical distribution vector of x. Plotting y against x gives a plot of the empirical cumulative distribution function, which can be compared with the desired distribution. In the previous example from the normal we create the empirical distribution plot of vector x and the plot of the cumulative distribution function:

```r
x=sort(x)
n=length(x)
y=(1:n)/n
plot(y~x, type="l")
lines(pnorm(z, 1,2)~z, col="red")
```



We will use these methods in the following examples to check the quality of the random number generators we will create. Note that the simple method we used above to create the empirical distribution applies when all values of x are distinct. This is always the case when x is a sample from a continuous distribution. If the distribution is discrete then there are generally repeated values in x and some adjustment is needed. For example, for vector $x = (2, 4, 4, 5, 5, 7)$ the empirical cumulative frequencies are $(1/6, 3/6, 3/6, 5/6, 5/6, 6/6)$.

Example 1: Inverse Transform

Let X be a continuous random variable with probability distribution function

$$F(x) = \begin{cases} 0, & x \leq 1 \\ 1 - \frac{1}{x^3} & x > 1 \end{cases}$$

We first create a function that generates random numbers from the distribution of X using only random numbers from the uniform distribution $U(0, 1)$. The generator will use the inverse transform method.

We know that $F(X)\ U(0, 1$, therefore, if we take a random number $U \sim U(0, 1)$ and solve the equation $F(X) = U$ then $X \sim F(x)$. The equation $F(X) = U$ can be solved analytically.

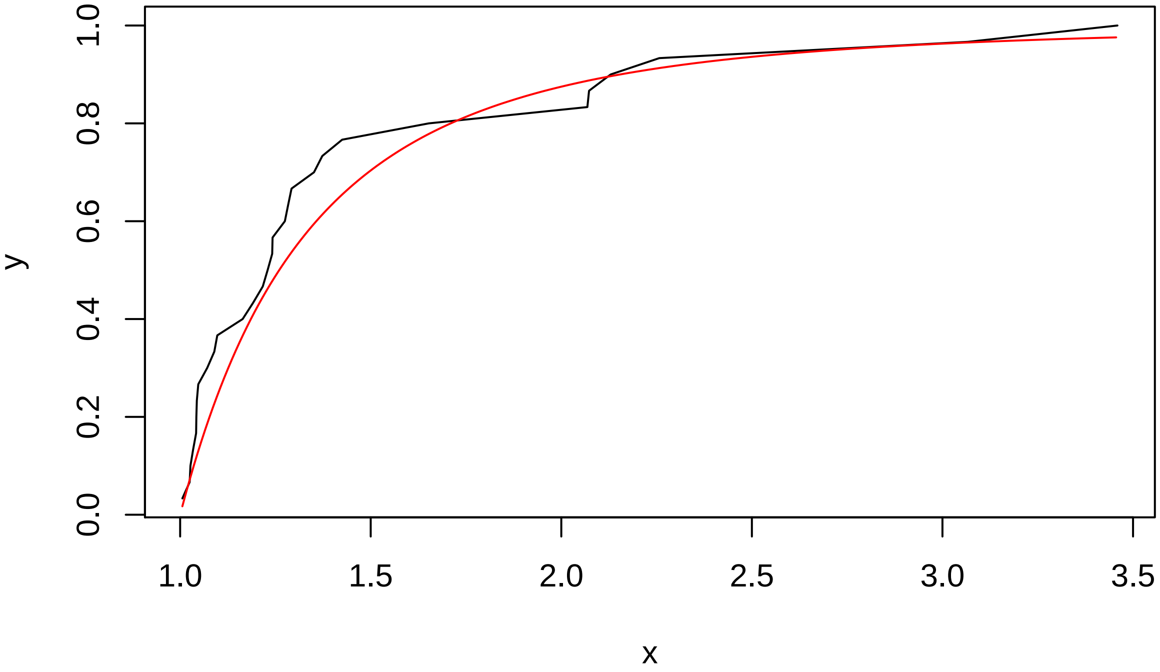
$$1 - \frac{1}{X^3} = U \Leftrightarrow X^3 = \frac{1}{1-U} \Leftrightarrow X = \left(\frac{1}{1-U}\right)^{1/3}.$$

The next function creates a sample of size N from distribution F.

```
Fgen1=function(N)
{
  u=runif(N,0,1)
  x=(1/(1-u))^(1/3)
  return(x)
}
```

We can now use the method we discussed in the beginning to generate a random vector of random numbers using this generator, create the empirical distribution plot and check the proximity with distribution function F:

```
N=30
x=Fgen1(N)
x=sort(x)
y=(1:N)/N
plot(y~x, type="l")
z=seq(min(x), max(x), 0.01)
F=1-1/z^3
lines(F~z, col="red")
```



Example 2: Inverse Transform

Let X be a random variable following Weibull distribution $W(k, \lambda)$. The Weibull distribution is a generalization of the exponential distribution and provides a more flexible method for modeling time durations. The probability density function is

$$f(x) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}, x \geq 0.$$

The cumulative distribution function is

$$F(x) = \begin{cases} 0, & x < 0 \\ 1 - e^{-(x/\lambda)^k}, & x \geq 0 \end{cases}$$

Note that for $k = 1$ we get the exponential distibution with mean λ, i.e., with rate $1/\lambda$.

To create the generator with the inverse transform method, we solve the equation $F(X) = U$, where $U \sim U(0,1)$:

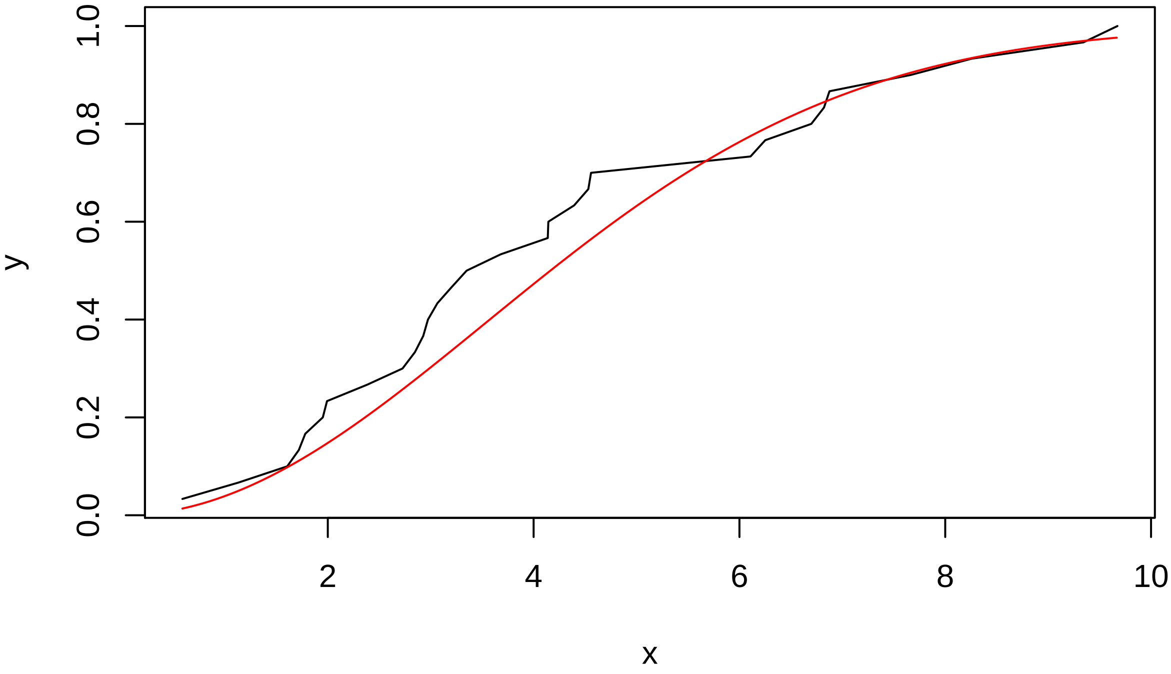
$$1 - e^{-(X/\lambda)^k} = U \Leftrightarrow \frac{X}{\lambda} = (-\log(1-U))^{1/k} \Leftrightarrow X = \lambda\,(-\log(1-U))^{1/k}.$$

Similarly to the previous example, we can now create the generator:

```
Fgen2=function(N,k,l)
{
  u=runif(N,0,1)
  x=l*(-log(1-u))^(1/k)
  return(x)
}
```

We next generate a random vector of random numbers from Weibull(2,5), create the empirical distribution plot and check the proximity with distribution function pweibull:

```
N=30
k=2
l=5
x=Fgen2(N,k,l)
x=sort(x)
y=(1:N)/N
plot(y~x, type="l")
z=seq(min(x), max(x), 0.01)
F=pweibull(z, k, l)
lines(F~z, col="red")
```



Example 3: Accept-Reject Method

We consider a continuous random variable X with probability density function

$$f(x) = \frac{c\sqrt{x}}{x+1},\ 0 \le x \le 3.$$

We will create a random number generator from X using the accept-reject method.

Since the random variable is bounded between 0 and 3, we will use the uniform distribution between 0 and 3 as the auxiliary distribution $g(x) = 1/3, 0 \leq x \leq 3$. First we must find a constant M such that

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

for all $0 \leq x \leq 3$. Since we do not know the value of the normalizing constant c, we rewrite the inequality as

$$\frac{\frac{c\sqrt{x}}{x+1}}{\frac{1}{3}} \leq M \Leftrightarrow \frac{\sqrt{x}}{x+1} \leq \frac{M}{3c}.$$

Let $b = \frac{M}{3c}$. We thus need to find a number b such that the above inequality holds for all x in the interval $[0, 3]$. One possibility is to set b to the maximum possible value of the ratio in $[0, 3]$. (In fact the theory of the algorithm guarantees that this corresponds to the best value of M for fast performance.) To do this, we use the R function optimize. The option maximum sets the objective to maximization (the default is minimization). The result of optimize is a list where maximum is the value of x that maximizes the function and objective is the maximum value.

```
h=function(x) {sqrt(x)/(x+1)}
s=optimize(h, interval=c(0,3), maximum = T)
s
```

```
## $maximum
## [1] 0.9999965
##
## $objective
## [1] 0.5
```

```
b=s$objective
b
```

```
## [1] 0.5
```

The algorithm for the accept-reject generator works as follows:

1. Create a random number Y from distribution $g(x)$.
2. Create a random number U from the uniform distribution $U(0, 1)$.
3. If $U \leq f(Y)/Mg(Y)$ then accept Y and set $X = Y$. Otherwise reject Y and return to step 1.

To adjust the algorithm to our example, we consider the inequality $U \leq f(Y)/Mg(Y)$. Substituting $f(Y) = \frac{c\sqrt{Y}}{Y+1}$, $g(Y) = 1/3$ and $\frac{M}{3c} = b$, we obtain

$$U \leq \frac{\frac{c\sqrt{Y}}{Y+1}}{M\frac{1}{3}} \Leftrightarrow U \leq \frac{\frac{\sqrt{Y}}{Y+1}}{b}.$$

Therefore, the accept-reject algorithm for the specific distribution of X is expressed as follows:

1. Create a random number Y from $U(0, 3)$.

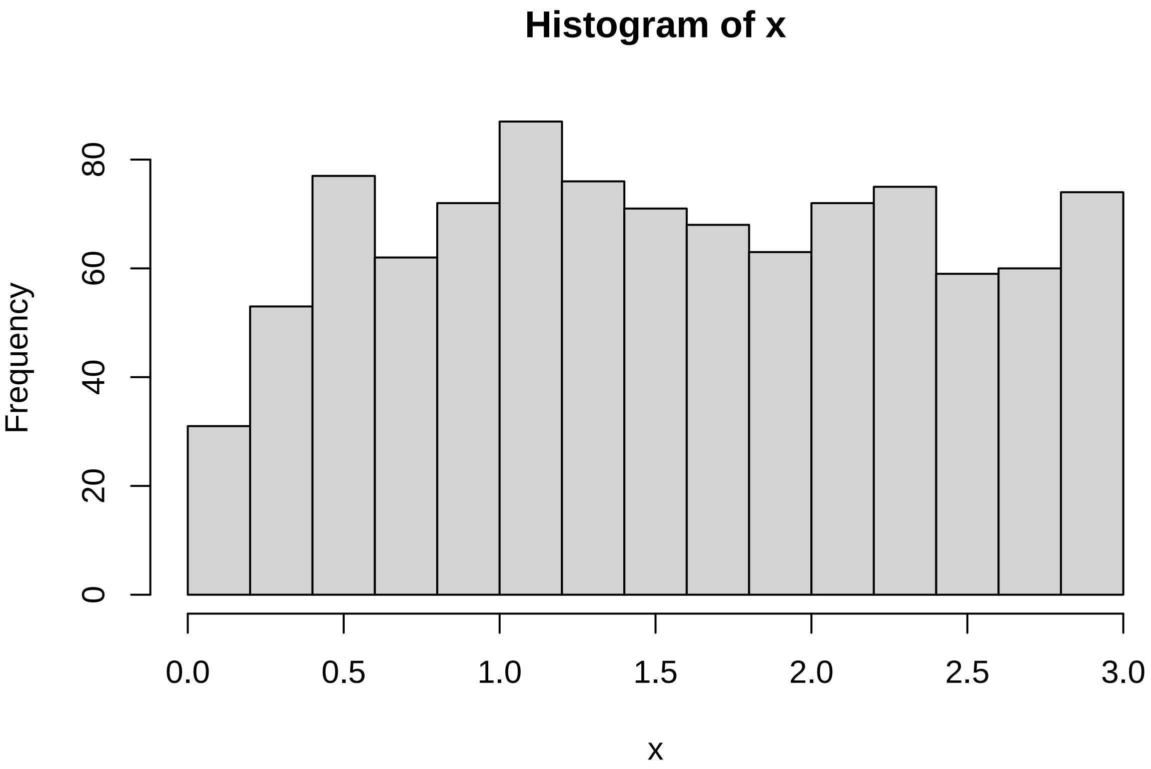
2. Create a random number U from the uniform distribution $U(0, 1)$.
3. If $U \leq \frac{\sqrt{Y}}{b(Y+1)}$ then accept Y and set $X = Y$. Otherwise reject Y and return to step 1.

The following function creates a single observation from X:

```
Fgen3=function()
{
  y=runif(1,0,3)
  u=runif(1,0,1)
  while(u>sqrt(y)/(b*(y+1)))
  {
    y=runif(1,0,3)
    u=runif(1,0,1)
  }
  x=y
  return(x)
}
```

We use the function to generate a sample of size 100:

```
N=1000
x=rep(0,N)
for (i in 1:N) x[i]=Fgen3()
hist(x)
```



Example 4: Accept-Reject

Let X be a continuous random variable with probability density function $f(x) = \frac{c\sqrt{x}e^{-x}}{x+1}, x \geq 0$. We will create a random number generator from X using the accept-reject method.

Since the random variable is not bounded, we cannot use the uniform distribution as auxiliary. We must use a distribution with support at least at $[0, \infty]$. We will use the exponential with rate 1, i.e., $g(x) = e^{-}x, x \geq 0$. First we must find a constant M such that

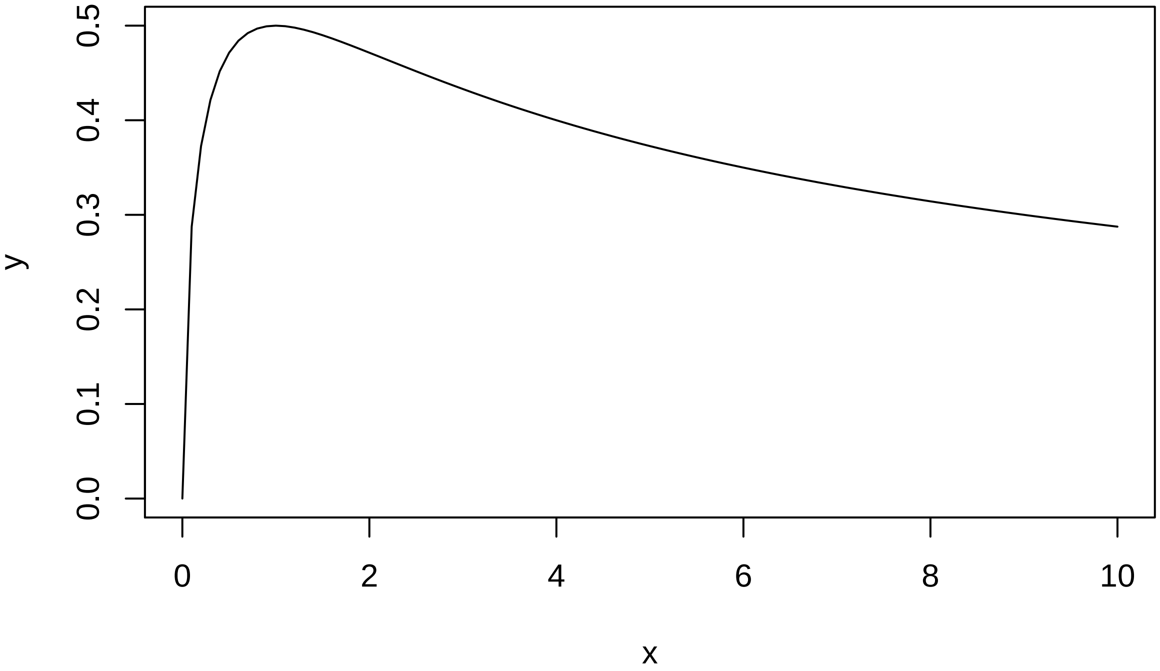
$$\frac{f(x)}{g(x)} \leq M$$

for all $0 \leq x < \infty$. Substituting $f(x), g(x)$ we obtain

$$\frac{\frac{ce^{-x}\sqrt{x}}{x+1}}{e^{-x}} \leq M \Leftrightarrow \frac{\sqrt{x}}{x+1} \leq \frac{M}{c}.$$

Let $b = \frac{M}{c}$. We thus need to find a number b such that the above inequality holds for all $x \geq 0$. One possibility is to set b to the maximum possible value of the ratio in $[0, \infty)$. In this case we cannot use the optimize function directly, because it needs a finite interval in which to search for the maximum. However if we make a plot of function $\frac{\sqrt{x}}{x+1}$,

```
x=seq(0,10,0.1)
y=sqrt(x)/(x+1)
plot(y~x, type="l")
```



we see that it is enough to search for the maximizing value in the interval $[0, 10]$ (this is not a mathematical proof, it can be proved rigorously using calculus). Thus

```
h=function(x) {sqrt(x)/(x+1)}
s=optimize(h, interval=c(0,1), maximum = T)
s
```

```
## $maximum
## [1] 0.9999243
##
## $objective
## [1] 0.5
```

```
b=s$objective
b
```

```
## [1] 0.5
```

The algorithm for the accept-reject generator works as follows:

1. Create a random number Y from distribution $g(x)$.
2. Create a random number U from the uniform distribution $U(0, 1)$.
3. If $U \leq f(Y)/Mg(Y)$ then accept Y and set $X = Y$. Otherwise reject Y and return to step 1.

To adjust the algorithm to our example, we consider the inequality $U \leq f(Y)/Mg(Y)$. Substituting

$f(Y) = \frac{cexp{-Y}\sqrt{Y}}{Y+1}$, $g(Y) = e^{-Y}$ and $\frac{M}{c} = b$, we obtain

$$U \leq \frac{\frac{ce^{-Y}\sqrt{Y}}{Y+1}}{Me^{-Y}} \Leftrightarrow U \leq \frac{\frac{\sqrt{Y}}{Y+1}}{b}.$$

Therefore, the accept-reject algorithm for the specific distribution of X is expressed as follows:

1. Create a random number Y from Exp(1).

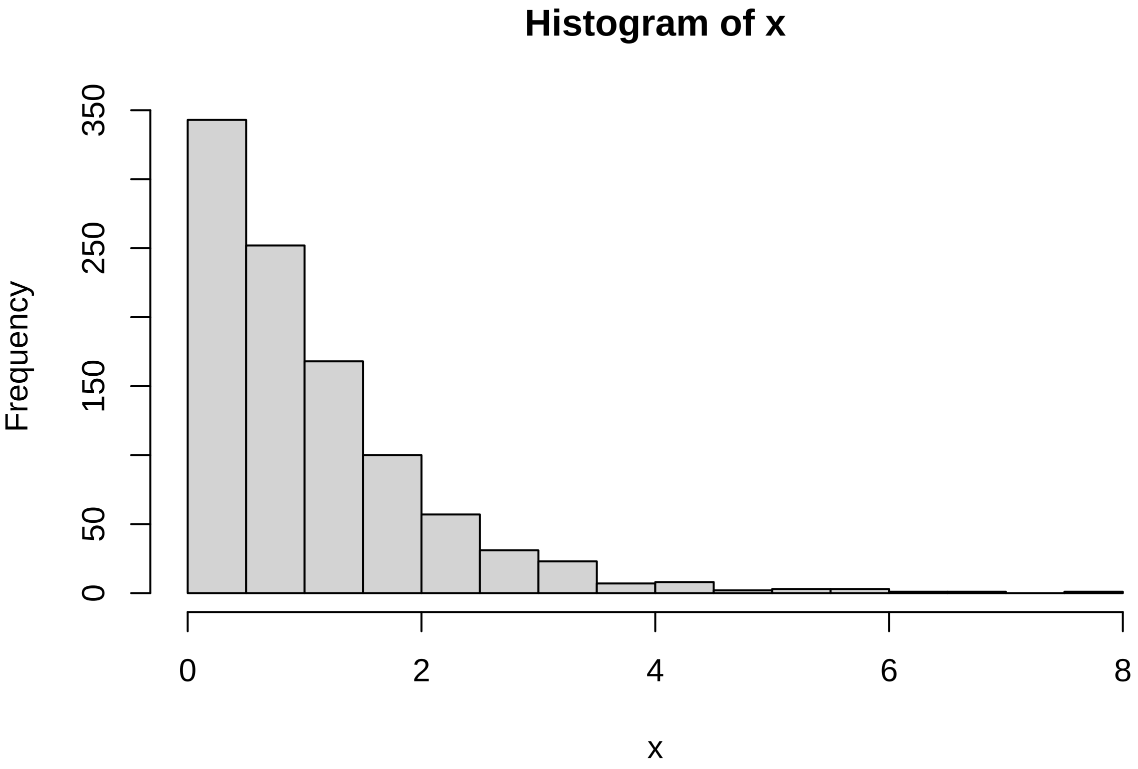
2. Create a random number U from the uniform distribution $U(0, 1)$.
3. If $U \leq \frac{\sqrt{Y}}{b(Y+1)}$ then accept Y and set $X = Y$. Otherwise reject Y and return to step 1.

The following function creates a single observation from X:

```
Fgen4=function()
{
  y=rexp(1,1)
  u=runif(1,0,1)
  while(u>sqrt(y)/(b*(y+1)))
  {
    y=rexp(1,1)
    u=runif(1,0,1)
  }
  x=y
  return(x)
}
```

We use the function to generate a sample of size 100:

```
N=1000
x=rep(0,N)
for (i in 1:N) x[i]=Fgen4()
hist(x)
```


Histogram of x
Frequency
x