

# Generating Random Variables

*Jonathan Navarrete*

*April 15, 2017*

## Introduction

One of the fundamental tools required in monte carlo methods is the ability to generate pseudo-random variables from a specified probability distribution. We explore some of these methods starting with the Inverse Transform Method. One of the most fundamental number generation is the uniform random generation. Pseudo random number genrators rely on the assumption that computational methods can consistently generate uniform random numbers. Though, these notes do not dive deep into the mechanics of random number generation, they do provide an overview of some topics. After having the ability to generate simple random numbers transformations are used to simulate other random processes.

In this section we'll cover the following methods for generating random numbers from a target distribution.

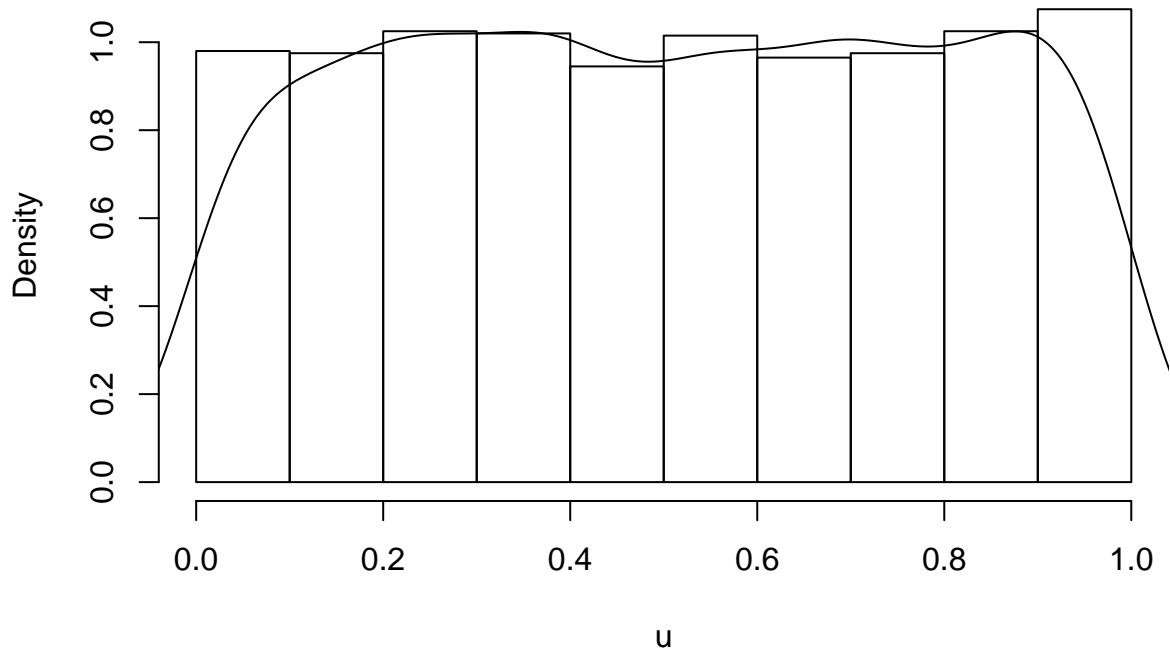
1. Inverse Transform Method
2. Accept-Reject Method
3. Transformation Method
4. Sums and Mixture distributions
5. Stochastic Processes

## Generation uniform samples

```
u = runif(2000)

#par(mfrow=c(1,2))
hist(u, probability=TRUE)
lines(density(u), xlim=c(0,1))
```

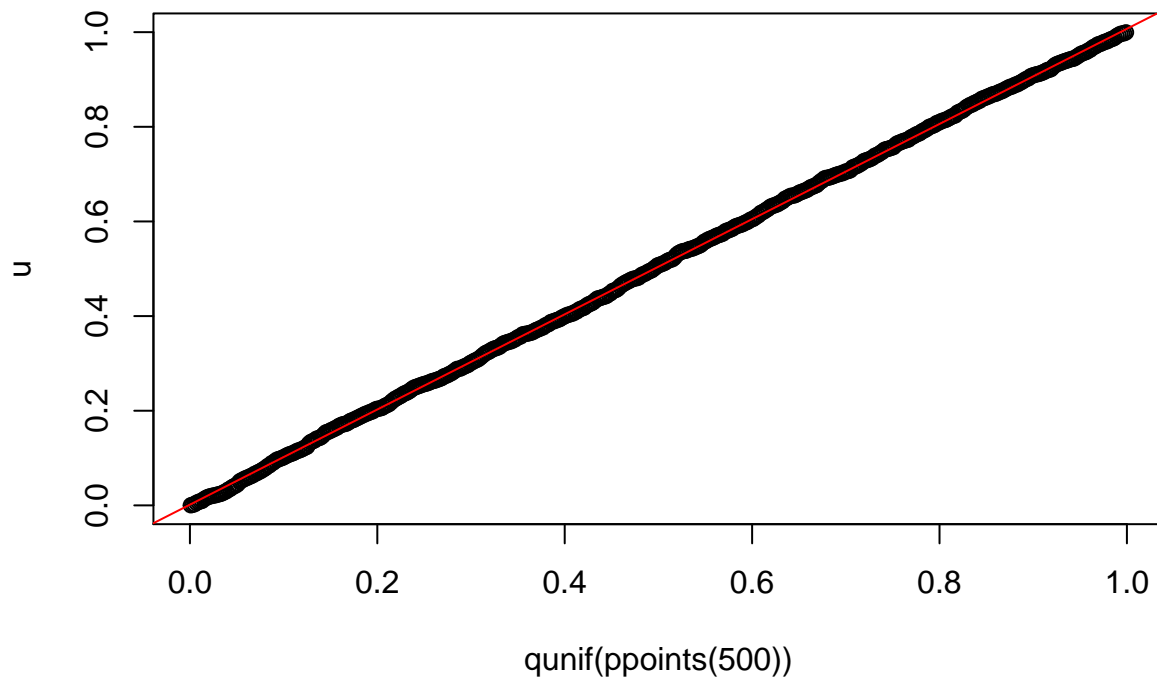
Histogram of u



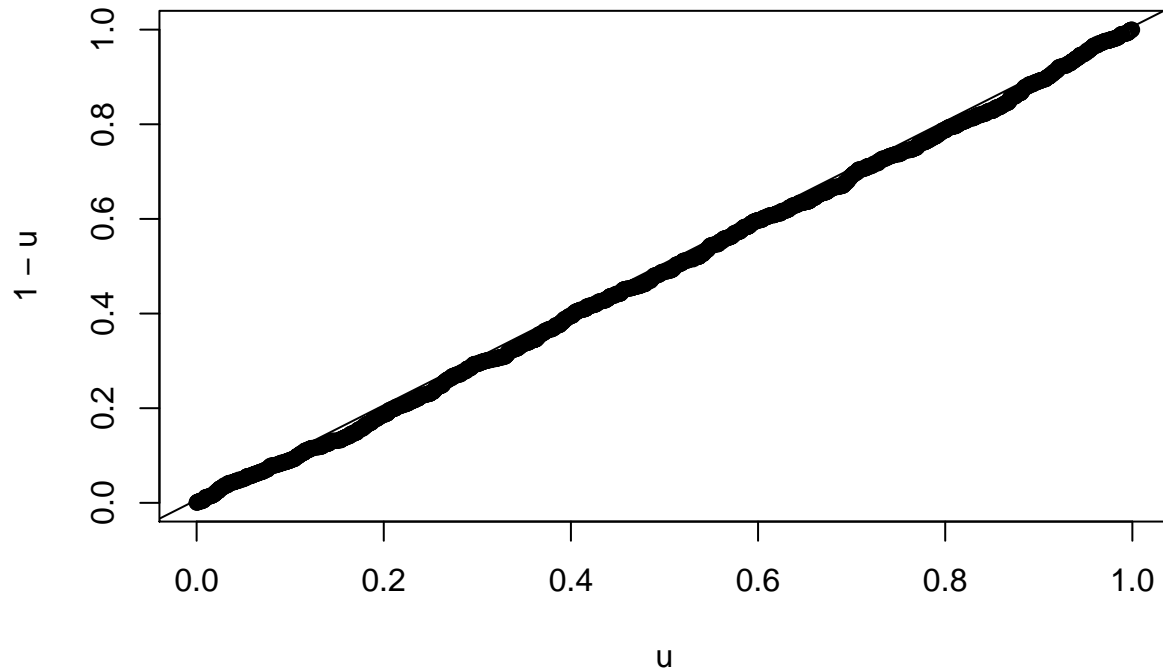
```
#par(mfrow=c(1,1))

## Q-Q plot for `runif` data against true theoretical distribution:
qqplot(qunif(ppoints(500)), u,
       main = expression("Q-Q plot for" ~~ {Unif(0,1)}))
qqline(u, distribution = qunif,
       prob = c(0.1, 0.6), col = 2)
```

Q-Q plot for Unif(0, 1)

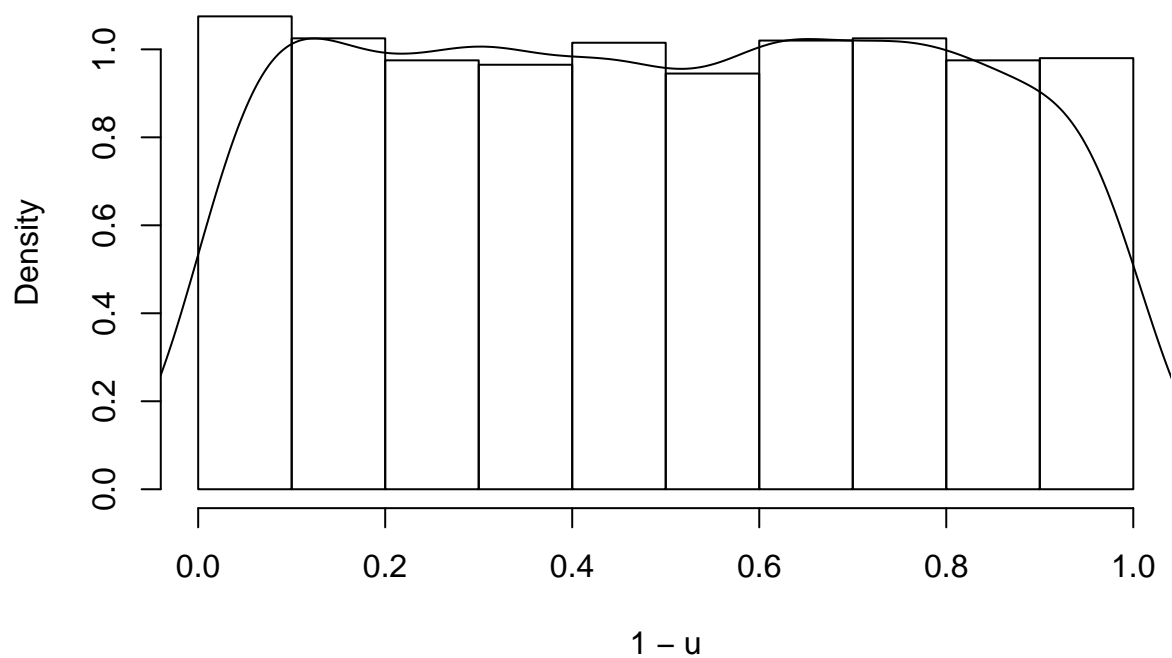


```
qqplot(u, 1-u)
qqline(u, distribution = qunif)
```



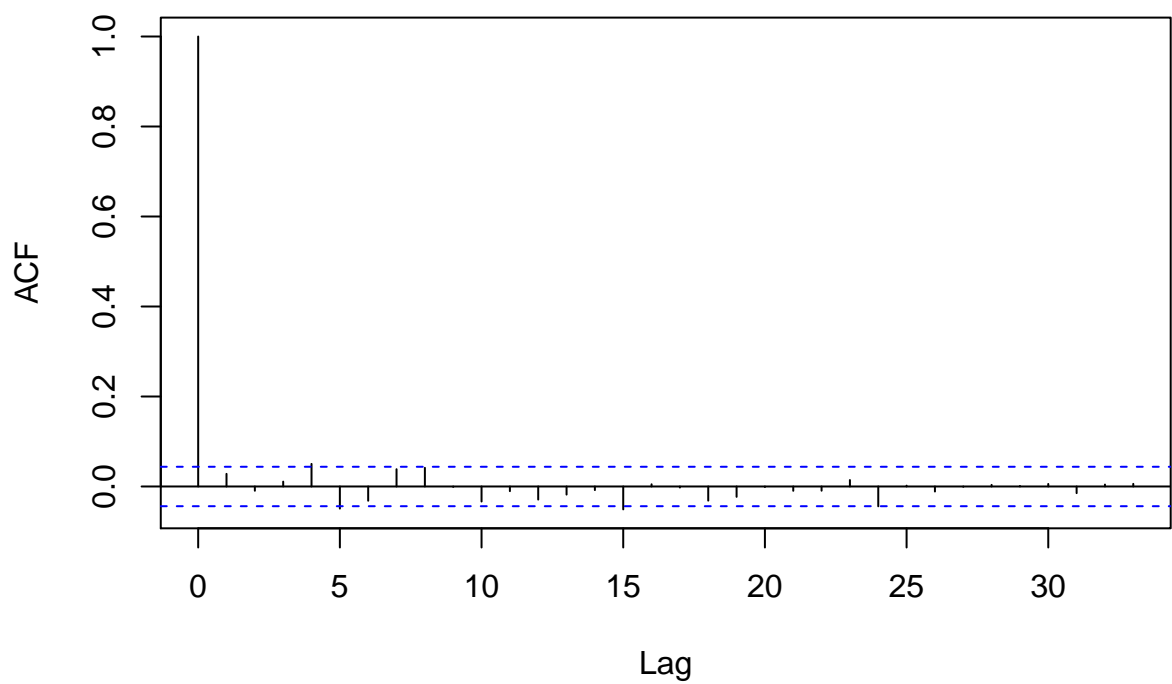
```
hist(1-u, probability = TRUE) ## also follows a uniform dist
lines(density(1-u), xlim=c(0,1))
```

### Histogram of $1 - u$



```
acf(u) ## autocorrelation of random number generation
```

### Series $u$



## Inverse Transform Method

General idea: only using a uniform distribution, generate random values and use an inverse CDF of the target distribution for which you wish to simulate. See the following link for further discussion: [How does the inverse transform method work?](#)

### Theorem (Probability Integral Transformation):

If  $X$  is a continuous random variable with CDF  $F_X(X)$ , then  $U = F_X(X) \sim \text{Uniform}(0, 1)$ . If  $U \sim \text{Uniform}(0, 1)$ , then for all  $x \in \mathbb{R}$

$$P(F_X^{-1}(U) \leq x) = P(\inf\{t : F_X(t) = U\} \leq x) = P(U \leq F_X(x)) = F_U(F_X(x)) = F_X(x)$$

and therefore  $F_X^{-1}(U)$  has the same distribution as  $X$ .

Steps: 1. For target probability distribution function (pdf)  $f(x)$ , calculate the inverse of the CDF by setting  $F(x) = U$ , then solving for  $X$ , for which  $U \sim \text{Unif}(0, 1)$ .

2. Generate  $N$  random numbers from  $U \sim \text{Unif}(0, 1)$
3. Plug in  $u$  observed values in  $F^{-1}(U)$  to obtain  $N$   $x$  values for which  $X \sim f(x)$

### Example: Exponential distribution

Suppose we are interested in generating 10,000 random values from an Exponential distribution

1.  $f(X) = \lambda e^{-\lambda X}$
2.  $F(X) = 1 - e^{-\lambda X} = U$
3.  $F^{-1}(U) = -1/\lambda \log(1 - U)$ ; can use  $(1-u)$  or  $u$ , since both are uniformly distributed.

If we set  $\lambda = 5$ , then

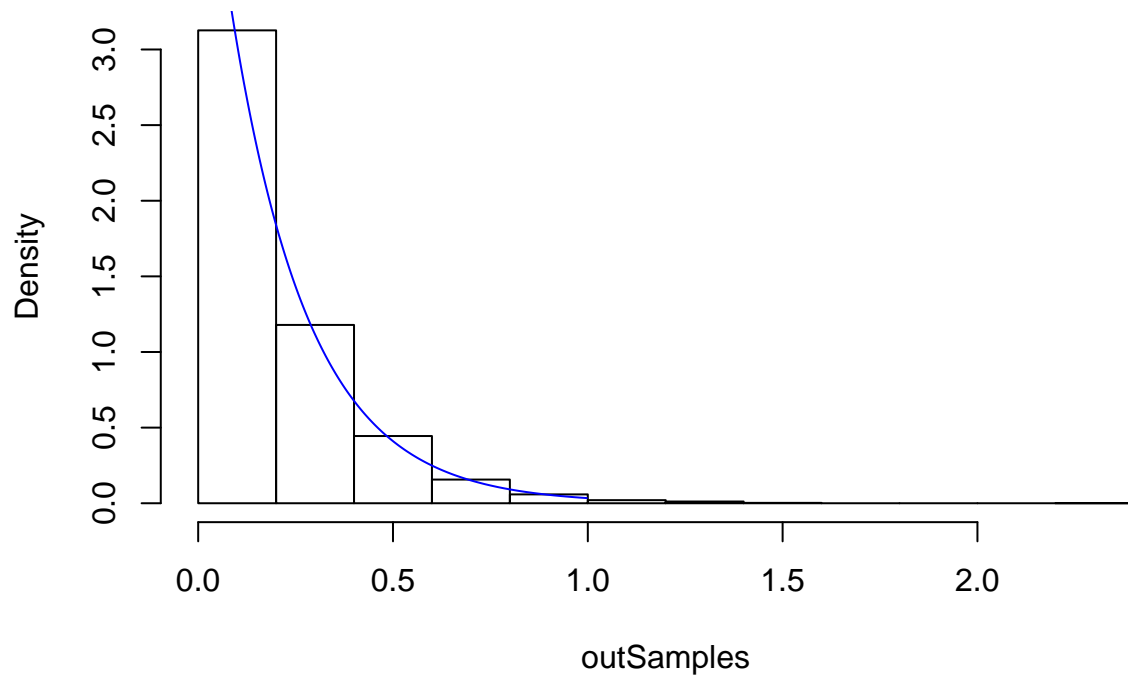
```
N = 10^4
u = runif(N)

fInv = function(u){
  (-1/5) * log(u) ## or log(1-u)
}

outSamples = fInv(u)

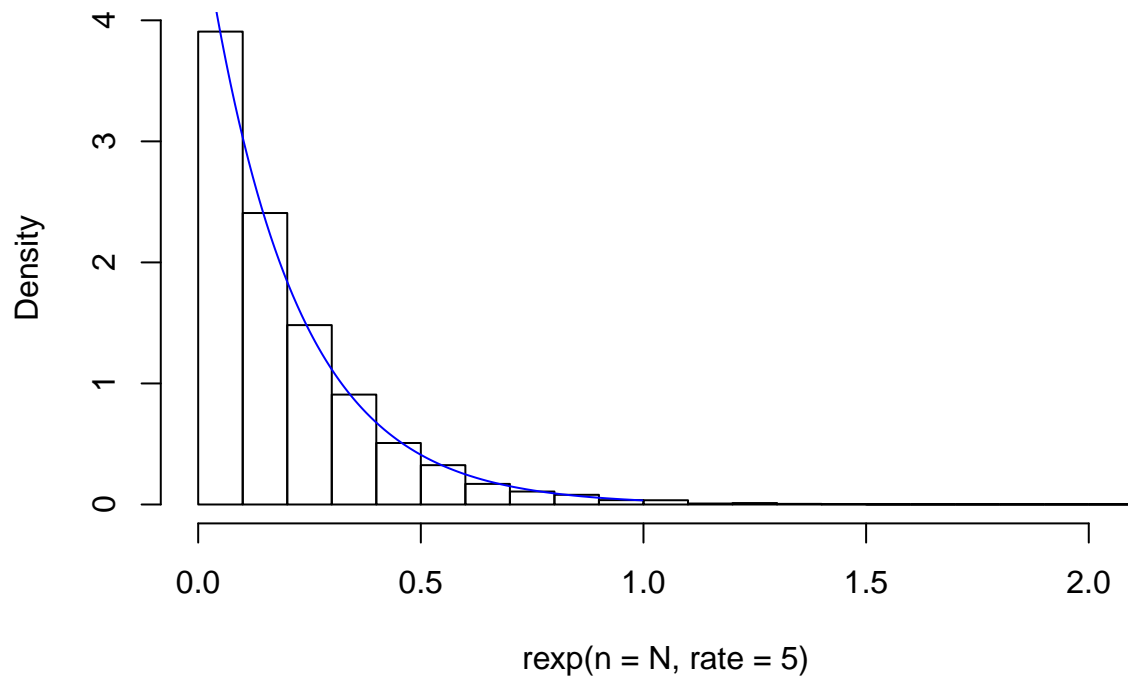
hist(outSamples, probability = TRUE)
lines(x = ppoints(200), y = dexp(x = ppoints(200), rate = 5),
      col = "blue")
```

## Histogram of outSamples



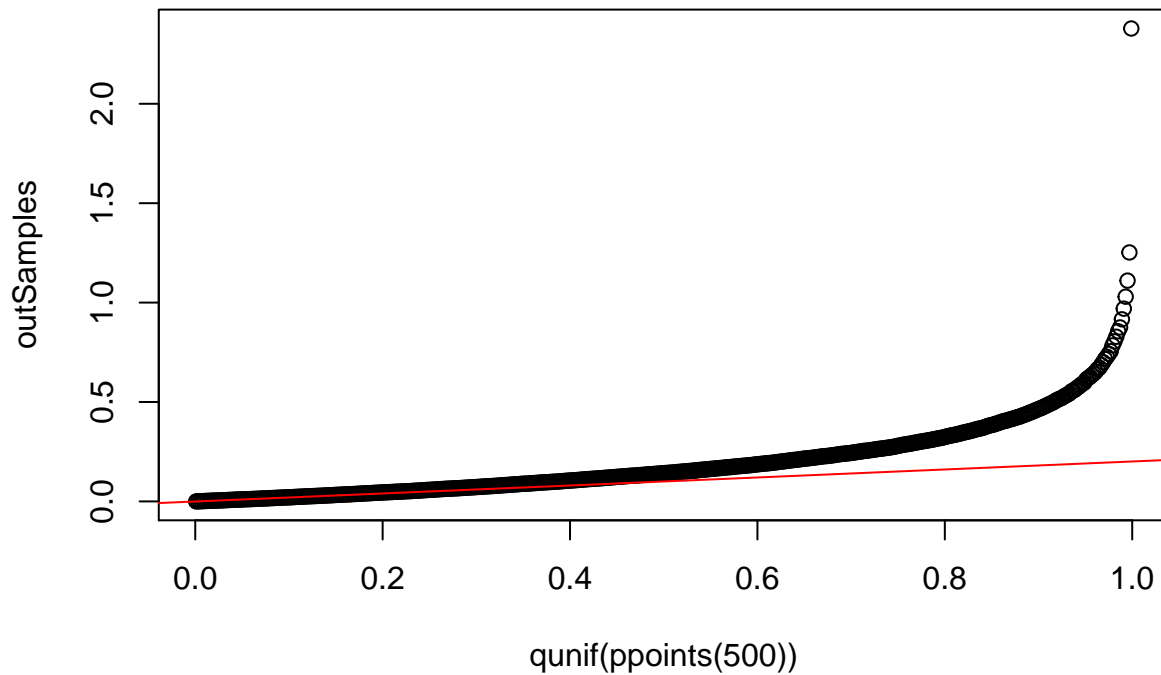
```
hist(rexp(n = N, rate = 5), probability = TRUE)
lines(x = ppoints(200), y = dexp(x = ppoints(200), rate = 5),
      col = "blue")
```

## Histogram of rexp(n = N, rate = 5)



```
qqplot(qunif(ppoints(500)), outSamples,
       main = expression("Q-Q plot for" ~~ {Exp(5)}))
qqline(outSamples, distribution = qexp,
       prob = c(0.1, 0.9), col = 2)
```

Q-Q plot for Exp(5)



### Example: Pareto Distribution

For information on the Pareto distribution, please see: [Pareto Distribution](#)

The  $Pareto(a, b)$  distribution has CDF  $F(X \leq x) = 1 - (\frac{b}{x})^a$  for  $x \geq b > 0$ ,  $a > 0$

1. First set  $F(x) = U$ , where  $U \sim Unif(0, 1)$

$$1 - \left(\frac{b}{x}\right)^a = U$$

2. Solve for X

$$\left(\frac{b}{x}\right)^a = 1 - U$$

- 3.

$$\frac{b}{x} = (1 - U)^{1/a}$$

- 4.

$$x = b \times (1 - U)^{-1/a}$$

```
n = 1000
U =runif(n)
```

```

a = 3
b = 2

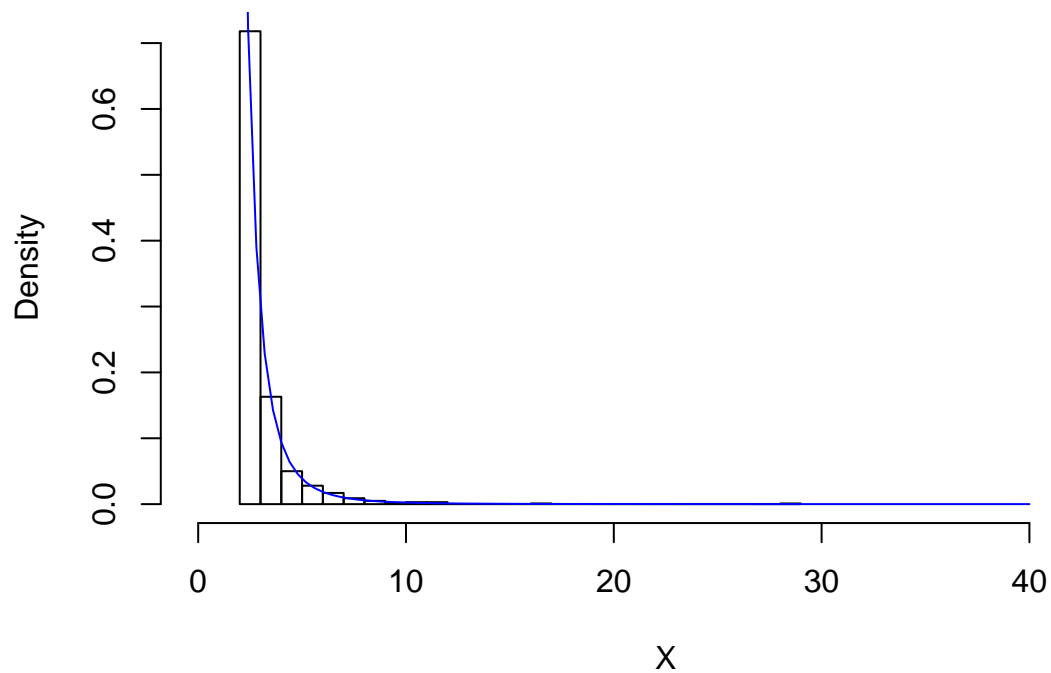
X = b*(1-U)^(-1/a)

hist(X, probability = TRUE, breaks = 25, xlim = c(0, 45), main = "Inverse Transform: Pareto(3,2)")

pareto = function(x){(a*(b^a)/x^(a+1))}
curve(pareto(x), from = 0, to = 40, add = TRUE, col = "blue")

```

### Inverse Transform: Pareto(3,2)



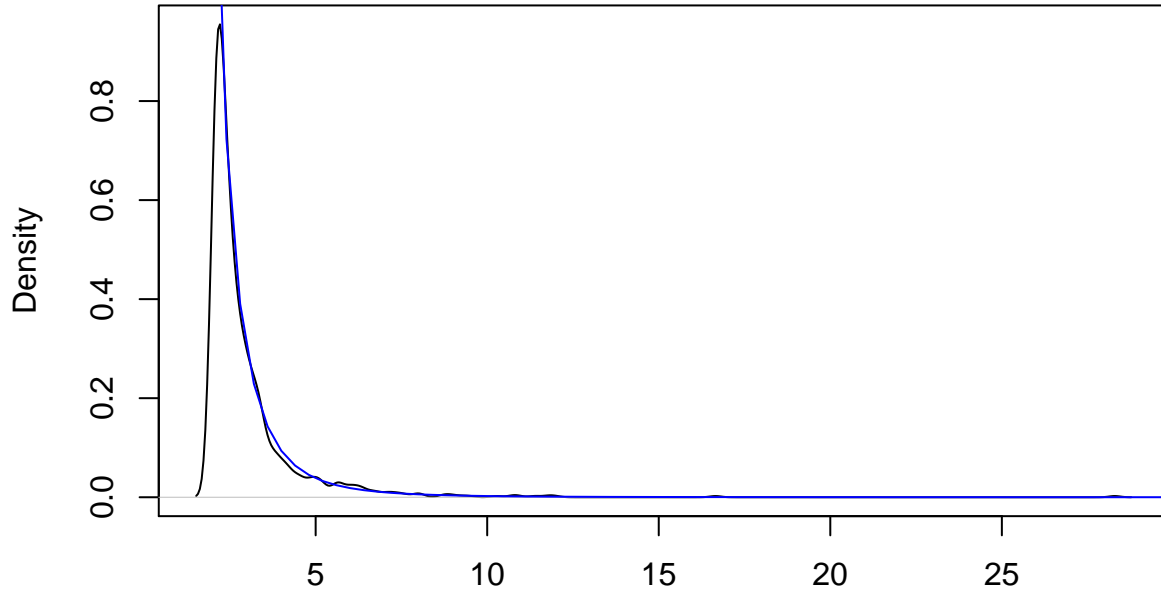
```

plot(density(X), main = "Density Plot of Inverse Transform: Pareto(3, 2)")
curve(pareto(x), from = 0, to = 40, add = TRUE, col = "blue")

```



## Density Plot of Inverse Transform: Pareto(3, 2)



N = 1000 Bandwidth = 0.161

### Inverse Transform Discrete scenario

For a given an ordered discrete random sample  $\dots < x_{i-1} < x_i < x_{i+1} < \dots$  from a distribution  $f(X)$ , with CDF  $F(x)$ . Then, the inverse transformation  $F_X^{-1}(u) = x_i$ , where  $F_X(x_{i-1}) < u \leq F_X(x_i)$ . Then for each random variable desired,

1. Generate a random variable  $u \sim Unif(0, 1)$
2. Deliver  $x_i$  where  $F(x_{i-1}) < u \leq F(x_i)$

### Example

Given the following distribution  $P(X = 0) = 0.1$ ,  $P(X = 1) = 0.2$ ,  $P(X = 2) = 0.2$ ,  $P(X = 3) = 0.2$ , and  $P(X = 4) = 0.3$ , use the inverse transform method to generate a random sample of size 1000 from the distribution.

$$F(X \leq x) = \begin{cases} 0.1 & \text{if } x \leq 0 \\ 0.3 & \text{if } x \leq 1 \\ 0.5 & \text{if } x \leq 2 \\ 0.7 & \text{if } x \leq 3 \\ 1.0 & \text{if } x \leq 4 \end{cases}$$

```
cdf = c(0.0, 0.1, 0.3, 0.5, 0.7, 1.0)
results = numeric(length = 1000) ## creates a vector of zeros
u = runif(1000)
for(i in 2:6){
  ind = (cdf[i-1] < u) & (u <= cdf[i])
```

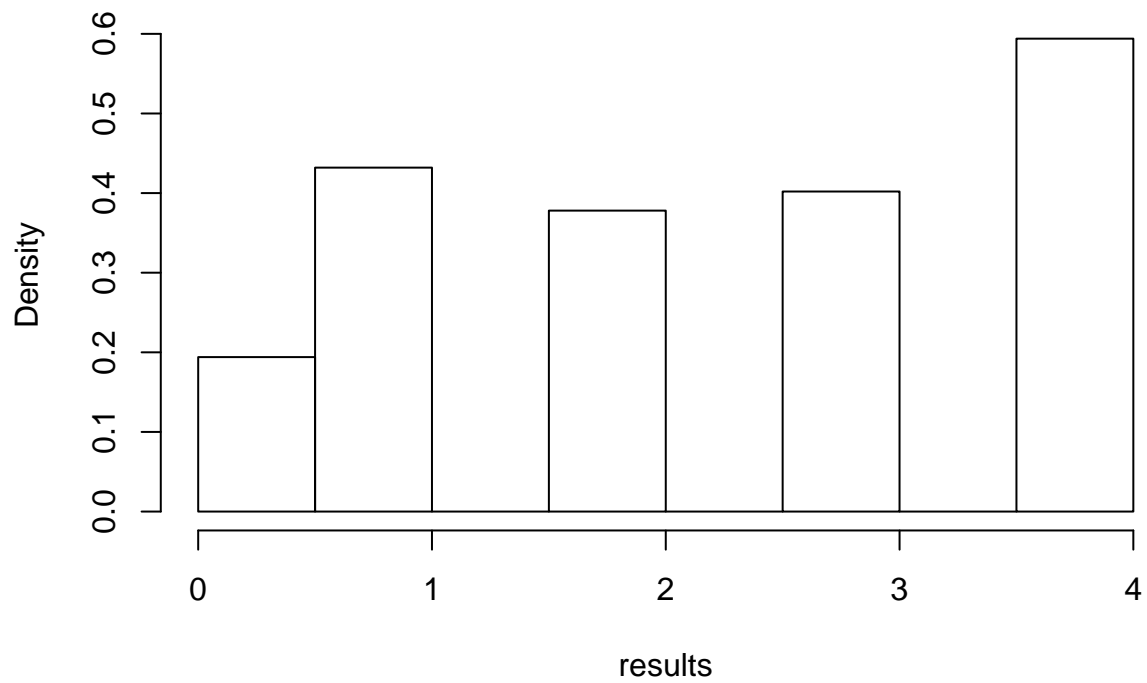
```
    results[ind] <- (i-2)
  }

table(results) / 1000

## results
##      0      1      2      3      4
## 0.097 0.216 0.189 0.201 0.297

hist(results, probability = TRUE)
```

**Histogram of results**



## Transformation Methods

Now we look at transformation methods for generating random variables. Distributions sometimes share relationships, and if the relationship is relatively simple, we can exploit this to generate random variables from a “simple to simulate” distribution and transform those random variables to generate samples from a “harder to simulate” distribution. I’ll give some examples below. First, see the following Wikipedia page for some relationships: Relationships among probability distributions

Now that we’ve covered how to simulate random variables from an  $Exp(\theta)$  distribution (“simple to simulate”), I’ll cover how to generate some random variables from “harder to simulate” distributions.

We’ll cover the following:

1.  $Y = \sum_{i=1}^N X$  where  $Y \sim Chi\ Square(2N)$  ( $2N$  degree of freedom)

See: Chi-Square distribution

2.  $Y = \beta \sum_{i=1}^{\alpha} X$  where  $Y \sim Gamma(\alpha, \beta)$

See: Gamma distribution

3.  $Y = \frac{\sum_{i=1}^a X}{\sum_{i=1}^{a+b} X}$  where  $Y \sim Beta(a, b)$

See: Beta distribution

### Example: Generate $R \sim Rayleigh$

First attempt the Inverse Transform Method and see why it won’t work.

For information on the Rayleigh distribution follow the link: Rayleigh Distribution

PDF:  $f(x|\sigma) = \frac{x}{\sigma^2} \exp(\frac{-x^2}{2\sigma^2})$  for  $x \geq 0$ ,  $\sigma > 0$

CDF:  $F(X \leq x) = 1 - \exp(\frac{-x^2}{2\sigma^2})$

Inverse Transform: Set  $F(x) = U$ , where  $U \sim Unif(0, 1)$ .

1.

$$1 - \exp(\frac{-x^2}{2\sigma^2}) = U$$

2.

$$\exp(\frac{-x^2}{2\sigma^2}) = 1 - U$$

3.

$$\log(\exp(\frac{-x^2}{2\sigma^2})) = \log(1 - U)$$

4.

$$\frac{-x^2}{2\sigma^2} = \log(1 - U)$$

5.

$$-x^2 = 2\sigma^2 \times \log(1 - U)$$

6.

$$x = \sqrt{-2\sigma^2 \times \log(1 - U)}$$

From the last equation, we see that we'd be taking the square root of negative values which would be problematic. Therefore, we need an alternative algorithm.

From information on the Rayleigh distribution, we know that given two i.i.d. random variables  $Z_1, Z_2 \sim N(0, \sigma)$  then  $R = \sqrt{Z_1^2 + Z_2^2} \sim \text{Rayleigh}(\sigma)$ . Therefore, in order to simulate 1 random variable from  $\text{Rayleigh}(\sigma)$ , we first generate 2 random variables from a Normal distribution with mean 0 and standard deviation  $\sigma$ .

To generate  $N$  Rayleigh random variables, our algorithm would be:

1. Generate  $2 \times N$  random variables  $Z_i \sim N(0, \sigma)$  for  $i \in (0, 2N)$
2. For each pair of  $Z_i \sim N(0, \sigma)$  use the transformation  $R = \sqrt{Z_1^2 + Z_2^2}$  to obtain  $N$  random variables from  $\text{Rayleigh}(\sigma)$ .

```
N = 1000

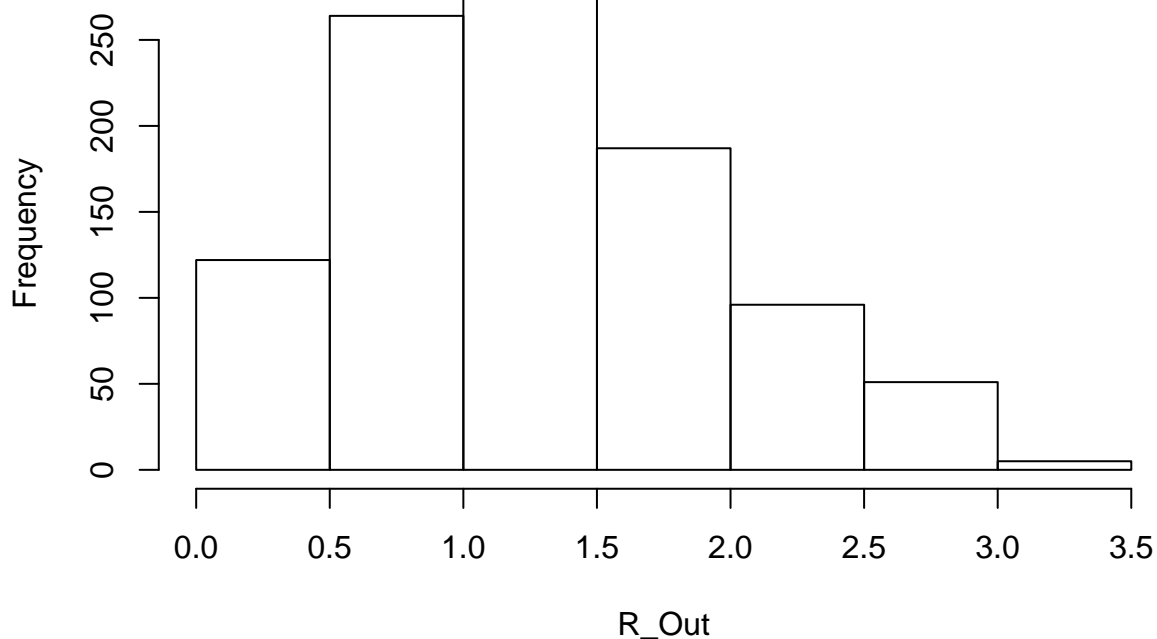
Z = rnorm(n = 2*N, mean = 0, sd = 1)
Z = matrix(data = Z, nrow = N, ncol = 2)

transformation <- function(vec){
  R = sqrt(sum(vec^2))
  #R = sqrt(vec[1]^2 + vec[2]^2)
  return(R)
}

R_Out = apply(X = Z, MARGIN = 1, FUN = transformation)

hist(R_Out)
```

**Histogram of R\_Out**



```
## compare with Rayleigh {VGAM}

sqrt(pi/2) ## theoretical mean
```

```
## [1] 1.253314
mean(R_Out) ## calculated mean
## [1] 1.266352
```