Inverse Transform method

Introduction to Simulation

- Simulation plays an increasingly large role in statistics.
- It is central to modern Bayesian data analysis, has long been utilized to study properties of statistical procedures that can't be easily derived analytically, and has numerous other applications.
- We will see many applications of simulation throughout the course, and they all require some basics in generating observations from specified probability distributions.
- We begin by studying some of the most important and widely used techniques.

Outline

- Introduction
- Methods for transforming unifrom random variables into other distributions
  - Inverse Transform Method
  - Acceptance-Rejection Method
- R: Built-in functions for random variables

Goal

Use $U(0,1)$ numbers to generate observations (variates) from other distributions.

- Discrete distributions, like Bernoulli, Binomial, Poisson, an empirical
- Continuous distributions like exponential, normal (many ways), and empirical
- Multivariate normal

How do we transform uniformly distributed random variables into random variables from distribution with CDF $F(\cdot)$?

Inverse Transform (or Inverse CDF) Method

Motivation

- Suppose that we wish to obtain draws from a continuous probability distribution with probability density function $f(x)$ and cumulative distribution function $F(x)$.
- Assume that the density function $f(x) > 0$ on an interval (a, b) and is 0 outside of this interval, allowing for the possibility that (a, b) could be infinite ($a = -\infty$, $b = \infty$ or both).
- Then $F(x)$ is a strictly increasing function on (a, b), which implies that F^{-1} is strictly increasing and is a one-to-one map from (0,1) to (a,b).

Inverse CDF Method

- Assume we know how to compute the CDF $F^{-1}(x)$ (aka quantile function)
- Inverse CDF :

If U is a uniform random variable, then $X = F^{-1}(U)$ is a r.v. with CDF $F(\cdot)$.

Proof (for Continuous Case)

- Now consider a random variable U that is uniform on (0,1) and define the random variable $X = F^{-1}(U)$. What is the distribution of X?

Let $a < x < b$.

$$P(X \le x) = P(F^{-1}(U) \le x) = P(U \le F(x)) = F(x)$$

So, X has cdf F.

This implies that if we can obtain a sample $U_1, U_2, ..., U_n$ from a uniform distribution, we can construct a sample drawn from F by $F^{-1}(U_1), F^{-1}(U_2), \ldots, F^{-1}(U_n)$.

Procedures

Here is the inverse transform method for generating a r.v. X having cdf $F(x)$:

(1) Sample U from $\mathcal{U}(0, 1)$.

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

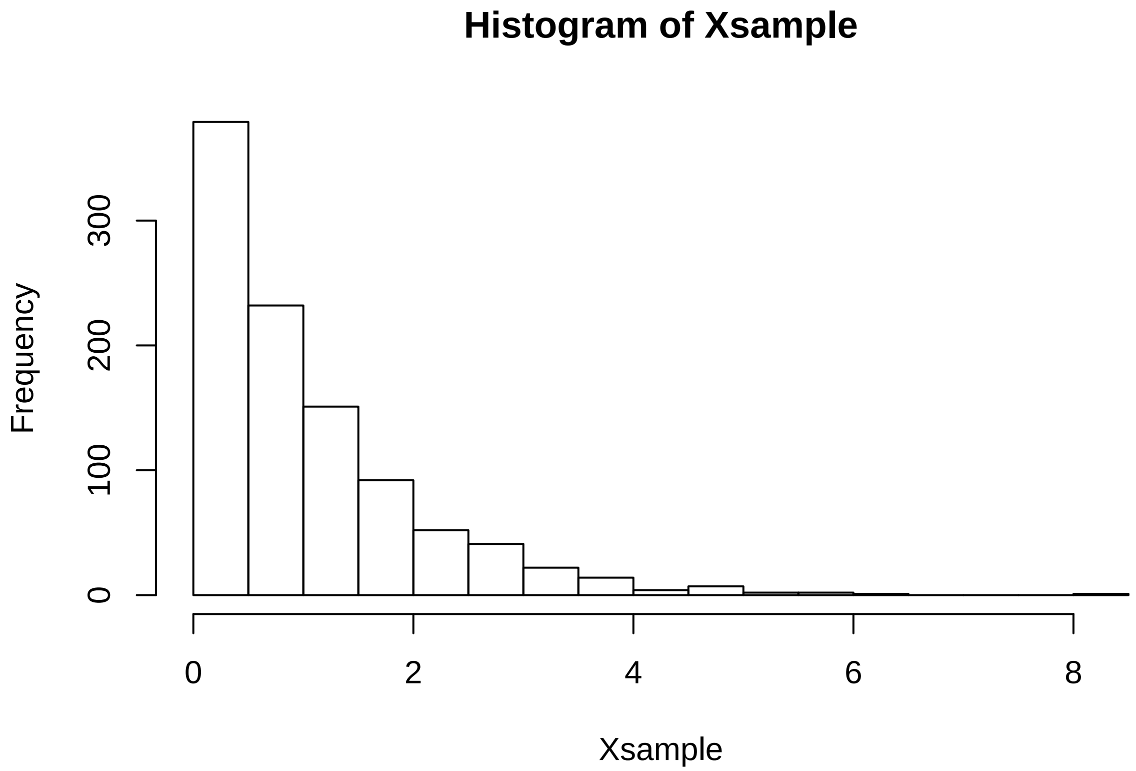
Example 1: The exponential distribution

Consider the density $f(x) = \lambda e^{-\lambda x}$ for $x > 0$.

$F(x) = \int_0^x \lambda e^{-\lambda t} dt = 1 - e^{-\lambda x}$

$x = F^{-1}(u) = -\frac{1}{\lambda} \ln(1 - u)$ □

```r
#draw 1000 from unit exponential and plot histogram
Usample=runif(1000,0,1);Xsample=-log(1-Usample)
hist(Xsample,nclass=25)
```



```
#Alternatively, could use
#Xsample=rexp(1000,1)
```

Example 2: The Weibull distribution

$F(x) = 1 - e^{-(\lambda x)^{\beta}}, \quad x > 0$

Solving $F(X) = U$ for X,

$X = \frac{1}{\lambda}[-\ln(1-U)]^{1/\beta}$ or $X = \frac{1}{\lambda}[-\ln(U)]^{1/\beta}$. $\square$

Example 3: The triangular distribution

The triangular $(0, 1, 2)$ distribution has pdf.

$$f(x) = \begin{cases} x & \text{if } 0 \le x < 1 \\ 2 - x & \text{if } 1 \le x \le 2. \end{cases}$$

The cdf is

$$F(x) = \begin{cases} x^2/2 & \text{if } 0 \le x < 1 \\ 1 - (x-2)^2/2 & \text{if } 1 \le x \le 2. \end{cases}$$

Continue: Caution!

(a) If $U < 1/2$, we solve $X^2/2 = U$ to get $X = \sqrt{2U}$.

(b) If $U \geq 1/2$, the only root of $1 - (X-2)^2/2 = U$ in $[1, 2]$ is

$$X = 2 - \sqrt{2(1-U)}.$$

Thus, for example, if $U = 0.4$, we take $X = \sqrt{0.8}$. □

Do not replace U by $1 - U$ here!

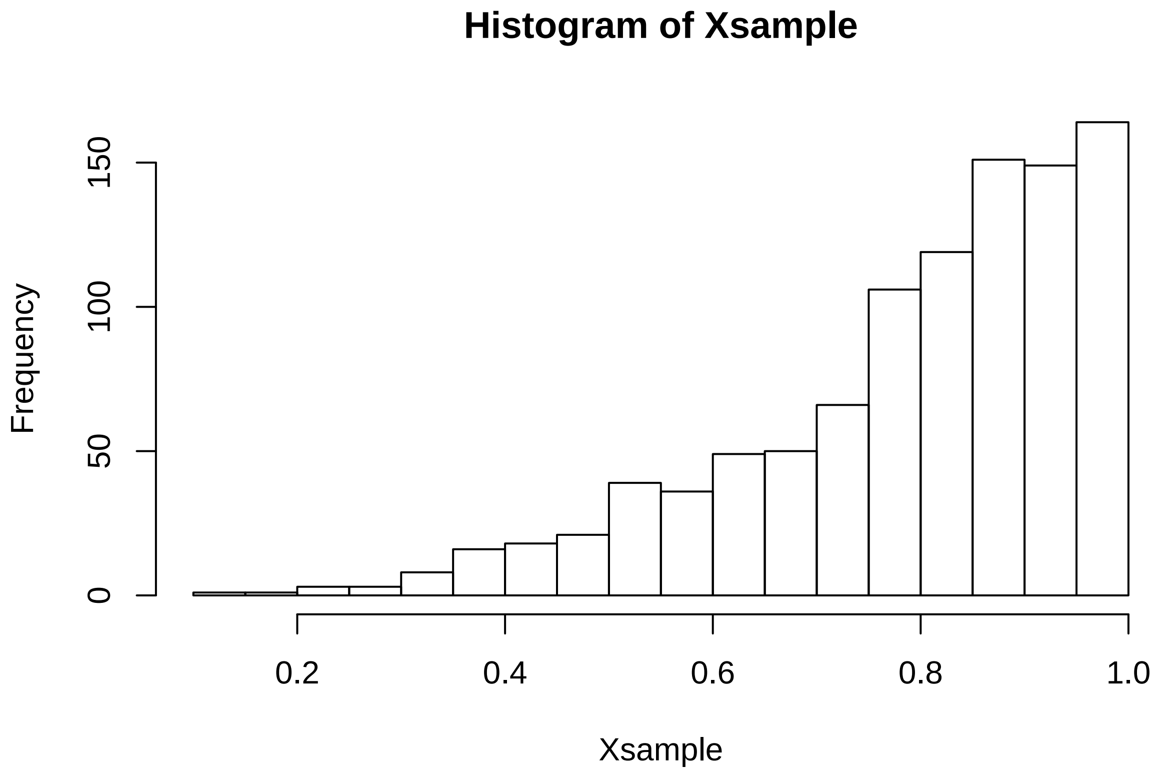
Example 4

Now consider the density $f(x) = 4x^3$ for $0 < x < 1$.

$$F(x) = \int_0^x 4t^3 dt = x^4$$

$$x = F^{-1}(u) = u^{1/4}$$

```
Usample=runif(1000,0,1)
Xsample=Usample^.25
hist(Xsample,nclass=25)
```



Inverse CDF Method in Discrete Case

Suppose a random variable X takes values in an ordered sample space of numbers:

$\Omega = \{x_1, x_2, ..., x_m\}$

We will assume the sample space is finite, but it may be extended to the countably infinite case.

Because of the discreteness of the sample space, the cdf is discontinuous because it jumps at each point in the sample space by a value equal to the probability of each point.

$F(x_1) < F(x_2) < \cdots < F(x_m) = 1.$

Of course $P[X = x_i] = F(x_i) - F(x_{i-1})$.

Procedures

Define the inverse cdf for $u \in (0, 1)$ according to

$F^{-1}(u) = x_i$ when $F(x_{i-1}) < u \le F(x_i)$

If $U \sim Unif(0, 1)$ and we let $X = F^{-1}(U)$,

$P[X = x_i] = P[F^{-1}(U) = x_i]$

$= P[F(x_{i-1}) < U \le F(x_i)] = F(x_i) - F(x_{i-1})$

Example 1:

Suppose

$$x = \begin{cases} -1 & \text{with } p = 0.6 \\ 2.5 & \text{with } p = 0.3 \\ 4 & \text{with } p = 0.1. \end{cases}$$

x	$P(X = x)$	$F(x)$	$\mathcal{U}(0, 1)$'s
-1	0.6	0.6	[0.0,0.6]
2.5	0.3	0.9	(0.6,0.9]
4	0.1	1.0	(0.9,1.0]

Thus, if $U = 0.63$, we take $X = 2.5$. $\square$

Example 2: Multinomial distribution

Suppose X takes values in categories $\{1, 2, 3, 4, 5\}$.

let $\pi_i = P[X = i]$ and assume

$\pi = (\pi_1, \pi_2, ..., \pi_5)^T = (.4, .1, .2, .05, .25)^T$

$F(1) = .4$, $F(2) = .5$, $F(3) = .7$, $F(4) = .75$, $F(5) = 1$.

Now generate 1000 observation of X.

```
Fx=cumsum(c(.4,.1,.2,.05,.25))
Usample=runif(1000,0,1)
Xsample=rep(1,1000)
for(i in 1:5){
Xsample=Xsample+(Usample > Fx[i])
}
```

```
# Let's see if this returns roughly
#the correct probabilities
table(Xsample)/1000
```

```
## Xsample
##     1     2     3     4     5
## 0.410 0.102 0.190 0.038 0.260
```

Another approach is to use sample()

```
probs=c(.4,.1,.2,.05,.25)
Xsample=sample(1:5,1000,replace=TRUE,probs)
table(Xsample)/1000
```

```
## Xsample
##     1     2     3     4     5
## 0.399 0.093 0.199 0.047 0.262
```

Drawbacks of the Inverse CDF Method

Inverse CDF often don't have closed form, and don't have nice numerical solutions

in many cases F^{-1} difficult to compute

Built-in random number generation functions

Built-in random number generation functions in R

- `R` already has functions for many of these distributions, and it's a good idea to explore them.
- Running **help(Distributions)** yields the following information.
- Keep in mind the **d** before a distribution refers to a function for evaluating the density of the distribution.
- Using **r** instead of **d** refers to a function for randomly generating observations from the distribution.

Basic random variable functions in R

- runif(), rnorm(), rbinom(), rpois(), rexp(), rt(), etc…
- First argument is always n, number of variables to generate
- Subsequent arguments are parameters to the distribution

Examples

For the beta distribution see dbeta.

For the binomial (including Bernoulli) distribution see dbinom.

For the Cauchy distribution see dcauchy.

For the chi-squared distribution see dchisq.

For the exponential distribution see dexp.

For the F distribution see df.

For the gamma distribution see dgamma.

For the geometric distribution see dgeom.

For the hypergeometric distribution see dhyper.

For the log-normal distribution see dlnorm.

For the multinomial distribution see dmultinom.

For the negative binomial distribution see dnbinom.

For the normal distribution see dnorm.

For the Poisson distribution see dpois.

For the Student's t distribution see dt.

For the uniform distribution see dunif.

For the Weibull distribution see dweibull.

R: Sample from a finite population

```
sample(x,size,replace=FALSE, prob=NULL)
```

- Built-in function to draw a random sample of **size** points from **x**, optionally with replacement and/or weights
- sample(x)-random permuation of **x** if **x** is a vector

Wrap-up

Summary

- Two basic methods for transforming uniform random variables into other distributions
- Inverse CDF method- often not possible because F^{-1} does not have nice form
- Rejection sampling - efficiency depends on having a good proposal distribution
- Built-in random number generation functions:
  - Common distributions - `runif()`, `rnorm()`, `rbinom()`,…
  - Random sample of the elements of a vector -`sample()`
  - These are building blocks