

2 Random Variable Generation

2.1 Basic methods

2.2 Beyond Uniform Distribution

- Rely on the possibility of producing (computer-wise) an endless flow of random variables (usually iid) from well-known distributions
- Given a uniform random number generator, illustration of methods that produce random variables from both standard and nonstandard distributions

2.1 Basic Methods

2.1.1 Introduction

For a function F on \mathbb{R} , *generalized inverse* of F , F^- , defined by

$$F^-(u) = \inf \{x; F(x) \geq u\} .$$

Probability Integral Transform:

If $U \sim \mathcal{U}_{[0,1]}$, then the random variable $F^-(U)$ has the distribution F .

Consequence:

To generate a random variable $X \sim F$, suffices to generate

$$U \sim \mathcal{U}_{[0,1]}$$

and then make the transform

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

2.1.2 Desiderata and Limitations

‘ ‘Any one who considers arithmetical methods of reproducing random digits is, of course, in a state of sin. As has been pointed out several times, there is no such thing as a random number---there are only methods of producing random numbers, and a strict arithmetic procedure of course is not such a method. ”

[John Von Neumann, 1951)]

- Production of a *deterministic* sequence of values in $[0, 1]$ which imitates a sequence of *iid* uniform random variables $\mathcal{U}_{[0,1]}$.
- Can't use the physical imitation of a “random draw” **[no guarantee of uniformity, no reproducibility]**
- *Random* sequence in the sense: Having generated (X_1, \dots, X_n) , knowledge of X_n [or of (X_1, \dots, X_n)] imparts no discernible knowledge of the value of X_{n+1} .

- Deterministic: Given the initial value X_0 , sample (X_1, \dots, X_n) always the same
- Validity of a random number generator based on a single sample X_1, \dots, X_n when n tends to $+\infty$, **not** on replications

$$(X_{11}, \dots, X_{1n}), (X_{21}, \dots, X_{2n}), \dots, (X_{k1}, \dots, X_{kn})$$

where n fixed and k tends to infinity.

2.1.3 Uniform pseudo-random number generator

Algorithm starting from an initial value u_0 and a transformation D , which produces a sequence

$$(u_i) = (D^i(u_0))$$

in $[0, 1]$.

For all n ,

$$(u_1, \dots, u_n)$$

reproduces the behavior of an *iid* $\mathcal{U}_{[0,1]}$ sample (V_1, \dots, V_n) when compared through usual tests

- Validity of the algorithm means that the sequence U_1, \dots, U_n leads to accept the hypothesis

$$H: U_1, \dots, U_n \text{ are iid } \mathcal{U}_{[0,1]}.$$

- The set of tests used is generally of some consequence
 - Kolmogorov–Smirnov
 - Time series methods, for correlation between U_i and $(U_{i-1}, \dots, U_{i-k})$
 - nonparametric tests
 - Marsaglia’s battery of tests called *Die Hard* (!)

2.1.4 The Kiss Generator

A real-life generated random sequence takes values on

$$\{0, 1, \dots, M\}$$

rather than in

$$[0, 1]$$

[M largest integer accepted by the computer]

Period, T_0 , of a generator: smallest integer T such that

$$u_{i+T} = u_i$$

for every i ,

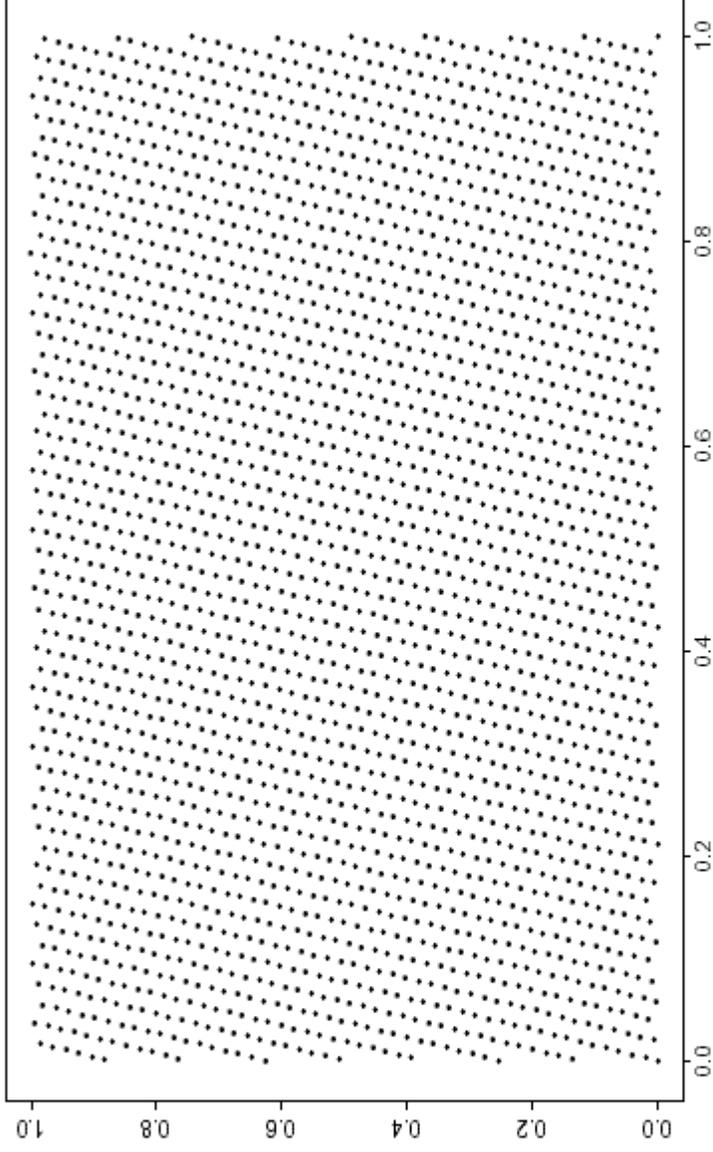
A generator of the form $X_{n+1} = f(X_n)$ has a period no greater than $M + 1$

Warning! A uniform generator on $[0, 1]$ should not never take the values 0 and 1 [\[Gentle, 1998\]](#)

Congruential generator on $\{0, 1, \dots, M\}$: defined by the function

$$D(x) = (ax + b) \bmod (M + 1).$$

- Period and other performance of congruential generators depend heavily on (a, b) .
- With a rational, pairs $(x_n, D(x_n))$ lie on parallel lines.



Representation of the line $y = 69069x \bmod 1$ by uniform sampling with sampling step 3×10^{-4} .

For $k \times k$ matrix T , with entries in $\{0, 1\}$, **shift register generator**:
given by the transformation

$$x_{n+1} = T x_n \pmod{2}$$

where x_n represented as a vector of binary coordinates $e_{ni} \in \{0, 1\}$,

$$x_n = \sum_{i=0}^{k-1} e_{ni} 2^i.$$

To generate a sequence of integers X_1, X_2, \dots , the **Kiss algorithm** generates **three** sequences of integers

- First, a congruential generator

$$I_{n+1} = (69069 \times I_n + 23606797) \pmod{2^{32}},$$

- Then two shift register generators (J_n) and (K_n)
- Overall sequence

$$X_{n+1} = (I_{n+1} + J_{n+1} + K_{n+1}) \pmod{2^{32}}$$

The period of *Kiss* is of order 2^{95}

Kiss has been successfully tested on *Die Hard*

2.2 Beyond Uniform Distributions

- Generation of any sequence of random variables can be formally implemented through a uniform generator
 - For distributions with explicit forms of F^{-1} (for instance, exponential, double-exponential or Weibull distributions), the Probability Integral Transform can be implemented.
 - Case specific methods, which rely on properties of the distribution (for instance, normal distribution, Poisson distribution)

- More general (indirect) methods exist, for example the accept-reject and the ratio-of-uniform methods
- Simulation of the standard distributions is accomplished quite efficiently by many statistical programming packages (for instance, IMSL, Gauss, Mathematica, Matlab/Scilab, Splus/R).

2.2.1 Transformation Methods

Case where a distribution F is linked in a simple way to another distribution easy to simulate.

Example 9 – Exponential variables– If $U \sim \mathcal{U}_{[0,1]}$, the random variable

$$X = -\log U/\lambda$$

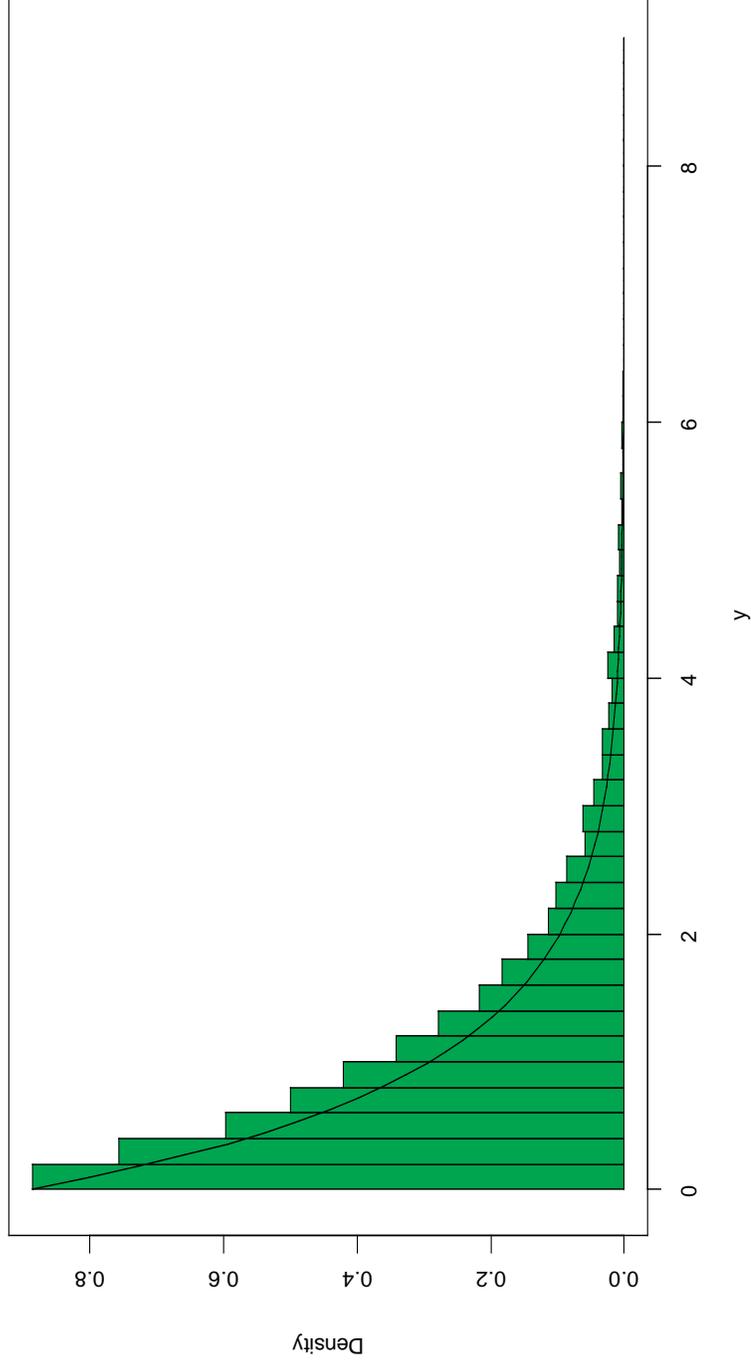
has distribution

$$\begin{aligned} P(X \leq x) &= P(-\log U \leq \lambda x) \\ &= P(U \geq e^{-\lambda x}) = 1 - e^{-\lambda x}, \end{aligned}$$

the exponential distribution $\mathcal{Exp}(\lambda)$.

```
#This generates exponentials from uniforms#  
nsim<-10000;u<-runif(nsim);  
y<--log(u);  
hist(y,main="Exponential",freq=F,col="green",breaks=50)  
par(new=T)  
plot(function(x)dexp(x), 0,10,xlab="",ylab="",xaxt="n",yaxt="n")
```

Exponential



Other random variables that can be generated starting from an exponential include

$$Y = -2 \sum_{j=1}^{\nu} \log(U_j) \sim \chi_{2\nu}^2$$

$$Y = -\beta \sum_{j=1}^a \log(U_j) \sim \mathcal{G}a(a, \beta)$$

$$Y = \frac{\sum_{j=1}^a \log(U_j)}{\sum_{j=1}^{a+b} \log(U_j)} \sim \mathcal{B}e(a, b)$$

Points to note

- Transformation quite simple to use
- There are more efficient algorithms for gamma and beta random variables
- Cannot generate gamma random variables with a non-integer shape parameter
- For instance, cannot get a χ_1^2 variable, which would get us a $\mathcal{N}(0, 1)$ variable.

Example 10 – Normal variables– If r, θ polar coordinates of (X_1, X_2) , then,

$$r^2 = X_1^2 + X_2^2 \sim \chi_2^2 = \text{Exp}(1/2)$$

and

$\theta \sim$ uniform distribution on $[0, 2\pi]$

Consequence: If U_1, U_2 iid $\mathcal{U}_{[0,1]}$,

$$X_1 = \sqrt{-2 \log(U_1)} \cos(2\pi U_2)$$

$$X_2 = \sqrt{-2 \log(U_1)} \sin(2\pi U_2)$$

iid $\mathcal{N}(0, 1)$.

Box-Muller Algorithm:

1. Generate U_1, U_2 iid $\mathcal{U}_{[0,1]}$;

2. Define

$$\begin{cases} x_1 = \sqrt{-2\log(u_1)} \cos(2\pi u_2) , \\ x_2 = \sqrt{-2\log(u_1)} \sin(2\pi u_2) ; \end{cases}$$

3. Take x_1 and x_2 as two independent draws from $\mathcal{N}(0,1)$.

- Unlike algorithms based on the CLT, this algorithm is exact
- Get two normals for the price of two uniforms
- Drawback (in speed) in calculating *log*, *cos* and *sin*.

Example 11 –Poisson generation–

Poisson–exponential connection:

If $N \sim \mathcal{P}(\lambda)$ and $X_i \sim \mathcal{Exp}(\lambda)$, $i \in \mathbb{N}^*$,

$$P_\lambda(N = k) =$$

$$P_\lambda(X_1 + \cdots + X_k \leq 1 < X_1 + \cdots + X_{k+1}) .$$

- A Poisson can be simulated by generating exponentials until their sum exceeds 1.
- This method is simple, but is really practical only for smaller values of λ .
- On average, the number of exponential variables required is λ .
- Other approaches are more suitable for large λ 's.

- A generator of Poisson random variables can produce negative binomial random variables since,

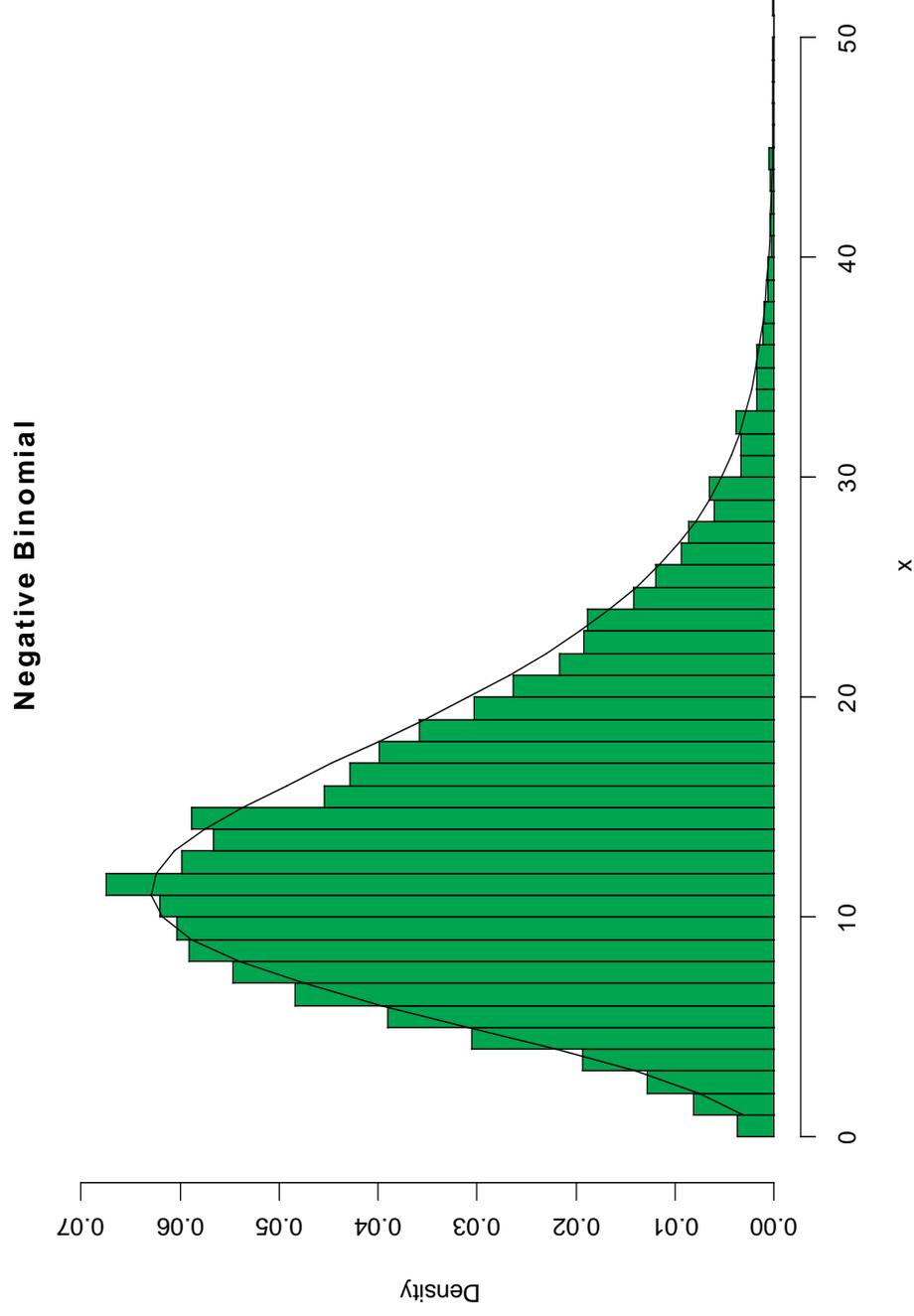
$$Y \sim \mathcal{Ga}(n, (1 - p)/p) \quad X|y \sim \mathcal{P}(y)$$

implies

$$X \sim \mathcal{Neg}(n, p)$$

Negative Binomial

- #This generates negative binomials as a mixture of poissons#
- `nsim<-10000;n<-6;p<-0.3;`
- `y<-rgamma(nsim,6,(1-p)/p);x<-rpois(nsim,y);`
- `hist(x,main="Negative Binomial",freq=F,col="green",breaks=40)`
- `par(new=T)`
- `lines(1:50,dnbinom(1:50,n,p))`



Mixture representation

- The representation of the negative binomial is a particular case of a *mixture distribution*
- The principle of a mixture representation is to represent a density f as the marginal of another distribution, for example

$$f(x) = \sum_{i \in \mathcal{Y}} p_i f_i(x) ,$$

- If the component distributions $f_i(x)$ can be easily generated, X can be obtained by first choosing f_i with probability p_i and then generating an observation from f_i .

2.2.2 Accept-Reject Methods

- Many distributions from which difficult, or even impossible, to **directly** simulate.
- Another class of methods that only require us to know the functional form of the density f of interest **only** up to a multiplicative constant.
- The key to this method is to use a simpler (simulation-wise) density g , the *instrumental density*, from which the simulation from the *target density* f is actually done.

Accept-Reject method

Given a density of interest f , find a density g and a constant M such that

$$f(x) \leq Mg(x)$$

on the support of f .

-
1. Generate $X \sim g$, $U \sim \mathcal{U}_{[0,1]}$;
 2. Accept $Y = X$ if $U \leq f(X)/Mg(X)$;
 3. Return to 1. otherwise.
-

Validation of the Accept-Reject method

This algorithm produces a variable Y distributed according to f

Two interesting properties:

- First, it provides a generic method to simulate from any density f that is known up to a *multiplicative factor*

Property particularly important in Bayesian calculations: there, the posterior distribution

$$\pi(\theta|x) \propto \pi(\theta) f(x|\theta) .$$

- is specified up to a normalizing constant
- Second, the probability of acceptance in the algorithm is $1/M$, e.g., expected number of trials until a variable is accepted is M

Example: Beta Accept-Reject

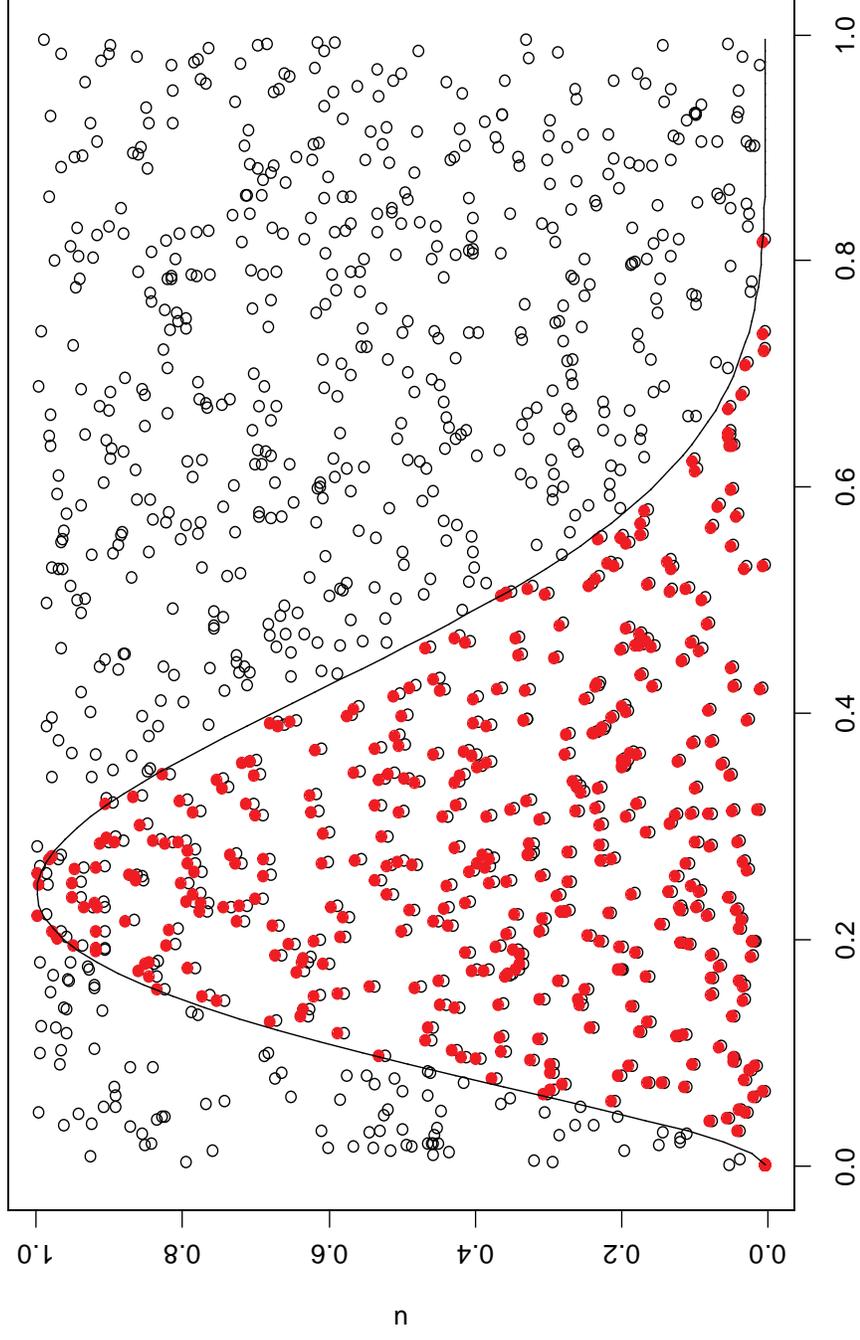
- Generate $Y \sim \text{beta}(a, b)$.
- No direct method if a and b are not integers.
- Set $a = 2.7$ and $b = 6.3$ and put the beta density $f_Y(y)$ inside a box with sides 1 and $c \geq \max_y f_Y(y)$.
- If (U, V) are independent uniform(0, 1) random variables

$$\begin{aligned}
 P(V \leq y | U \leq \frac{1}{c} f_Y(V)) &= \frac{P(V \leq y, U \leq (1/c) f_Y(V))}{P(U \leq \frac{1}{c} f_Y(V))} \\
 &= \frac{(1/c) P(Y \leq y)}{(1/c)} \\
 &= P(Y \leq y)
 \end{aligned}$$

Beta Accept-Reject

```
#This gives an accept-reject algorithm for a beta rv#
a<-2.7; b<-6.3; c<-2.669;nsim<-1000;
#Generate u and v#
u<-runif(nsim);v<-runif(nsim);
#Generate Y, the beta random variable#
test<-dbeta(v, a, b, ncp=0, log = FALSE)/c; ind<-(u<test);
plot(v,u)
par(new=T)
plot(v*(u<test),u*(u<test),xlab="",ylab="",xaxt="n",yaxt="n",col="red",pch=19)
par(new=T)
plot(function(x)(dbeta(x, a, b, ncp=0, log = FALSE)/c),xlab="",ylab="",xaxt="n",yaxt="n")
```

Beta Accept-Reject



The beta distribution with $a=2.7$ and $b=6.3$ with $c=\max_y f_Y(y) = 2.669$.

Beta Accept Reject

- For $c=2.669$ the acceptance probability is $1/2.669 = .37$,
so we accept 37%
- If we simulate from a $\text{beta}(2,6)$, the bound is 1.67,
so we accept 60%

Some intuition

- In cases f and g both probability densities, the constant M is necessarily larger than 1.
- The size of M , and thus the efficiency of the algorithm, functions of how closely g can imitate f , especially in the tails
- For f/g to remain bounded, necessary for g to have tails thicker than those of f .

It is therefore impossible to use the A-R algorithm to simulate a Cauchy distribution f using a normal distribution g , however the reverse works quite well.

Example 12 –Normal from a Cauchy–

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2/2)$$

and

$$g(x) = \frac{1}{\pi} \frac{1}{1+x^2},$$

densities of the normal and Cauchy distributions.

$$\frac{f(x)}{g(x)} = \sqrt{\frac{\pi}{2}} (1+x^2) e^{-x^2/2} \leq \sqrt{\frac{2\pi}{e}} = 1.52$$

attained at $x = \pm 1$.

So probability of acceptance

$$1/1.52 = 0.66,$$

and, on the average, one out of every three simulated Cauchy variables is rejected.

Mean number of trials to success 1.52.

Example 13 Normal/Double Exponential

Generate a $\mathcal{N}(0, 1)$ by using a double-exponential distribution with density

$$g(x|\alpha) = (\alpha/2) \exp(-\alpha|x|)$$
$$\frac{f(x)}{g(x|\alpha)} \leq \sqrt{\frac{2}{\pi}} \alpha^{-1} e^{-\alpha^2/2}$$

and minimum of this bound (in α) attained for $\alpha^* = 1$.

Probability of acceptance $\sqrt{\pi/2}e = .76$: To produce one normal random variable, this Accept-Reject algorithm requires on the average $1/.76 \approx 1.3$ uniform variables.

Compare with the fixed single uniform required by the Box-Muller algorithm.

Example 14 –Gamma with non-integer shape parameter–

Illustrates a real advantage of the Accept-Reject algorithm

The gamma distribution $\mathcal{G}a(\alpha, \beta)$ represented as the sum of α exponential random variables, only if α is an integer

Can use the Accept-Reject algorithm with instrumental distribution

$$G(a, b), \text{ with } a = [\alpha], \quad \alpha \geq 0.$$

(Without loss of generality, $\beta = 1$.)

Up to a normalizing constant,

$$f/g_b = b^{-a} x^{\alpha-a} \exp\{-(1-b)x\} \leq b^{-a} \left(\frac{\alpha-a}{(1-b)e} \right)^{\alpha-a}$$

for $b \leq 1$.

The maximum is attained at $b = a/\alpha$.

Example 15 – Truncated Normal distributions—

Truncated Normals appear in many contexts

Constraints $x \geq \underline{\mu}$ produce densities proportional to

$$e^{-(x-\underline{\mu})^2/2\sigma^2} \mathbf{1}_{x \geq \underline{\mu}}$$

for a bound $\underline{\mu}$ large compared with μ

Alternatives far superior to the naïve method of generating a $\mathcal{N}(\mu, \sigma^2)$ until exceeding $\underline{\mu}$, which requires an average number of $1/\Phi((\mu - \underline{\mu})/\sigma)$ simulations from $\mathcal{N}(\mu, \sigma^2)$ for one acceptance.

Instrumental distribution: translated exponential distribution,
 $\mathcal{Exp}(\alpha, \underline{\mu})$, with density

$$g_\alpha(z) = \alpha e^{-\alpha(z-\underline{\mu})} \mathbf{1}_{z \geq \underline{\mu}}.$$

The ratio f/g_α is bounded by

$$f/g_\alpha \leq \begin{cases} 1/\alpha \exp(\alpha^2/2 - \alpha\underline{\mu}) & \text{if } \alpha > \underline{\mu}, \\ 1/\alpha \exp(-\underline{\mu}^2/2) & \text{otherwise.} \end{cases}$$