

**CS395T**  
**Computational Statistics with**  
**Application to Bioinformatics**

Prof. William H. Press  
Spring Term, 2009  
The University of Texas at Austin

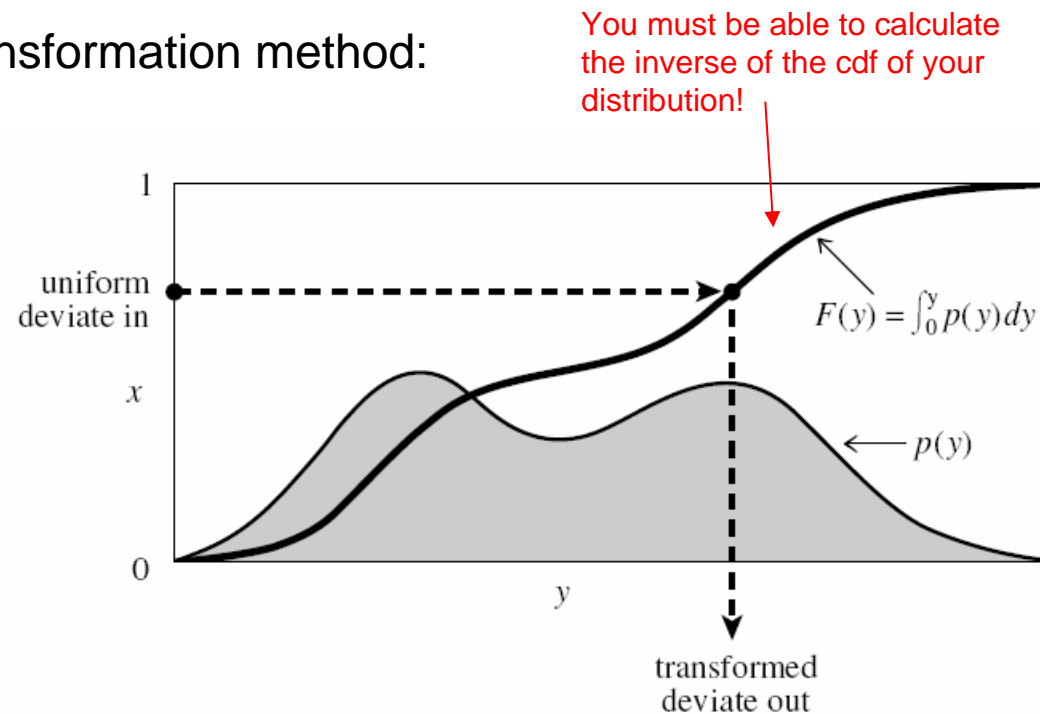
**Unit 7: Random Deviates**

## Random deviates from univariate distributions

You generally have a random generator for  $U(0,1)$ . What do you do for other distributions?

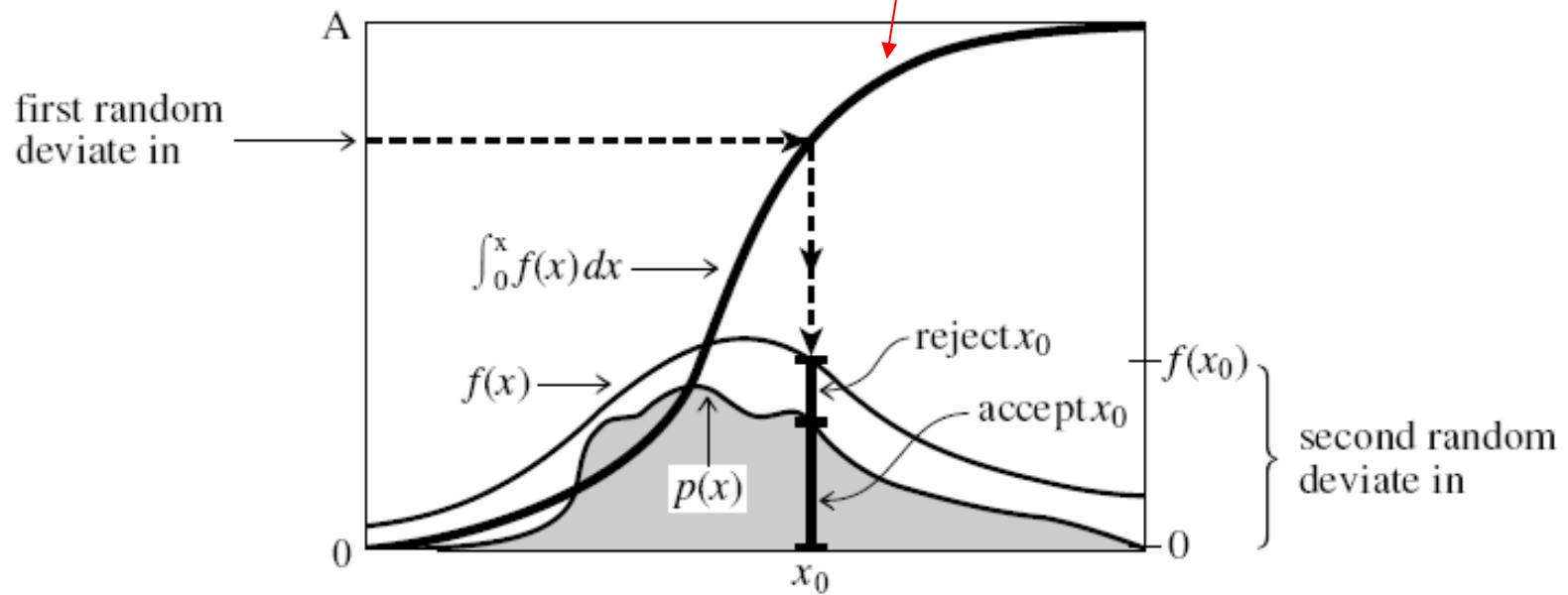
Here are three general methods:

Transformation method:



Rejection method:

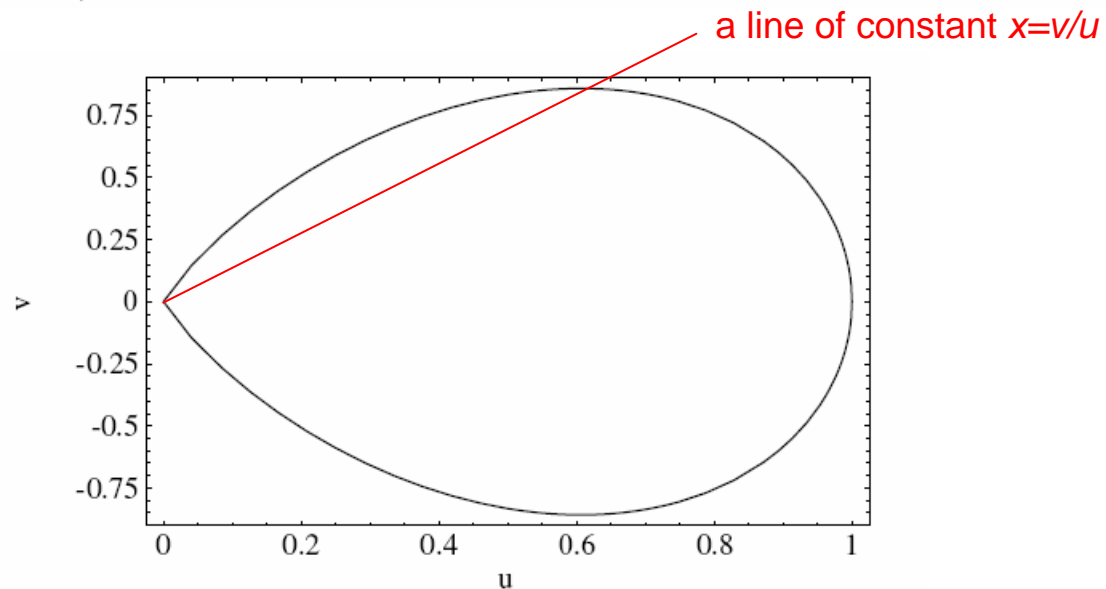
You must have a bounding function for which you are able to calculate the inverse of the cdf!



## Ratio of Uniforms Method

(some of the best features of both xformation and rejection)

- Construct the region in the  $(u, v)$  plane bounded by  $0 \leq u \leq [p(v/u)]^{1/2}$ .
- Choose two deviates,  $u$  and  $v$ , that lie uniformly in this region.
- Return  $v/u$  as the deviate.



Proof: We can represent the ordinary rejection method by the equation in the  $(x, p)$  plane,

$$p(x)dx = \int_{p'=0}^{p'=p(x)} dp' dx \quad (7.3.18)$$

Since the integrand is 1, we are justified in sampling uniformly in  $(x, p')$  as long as  $p'$  is within the limits of the integral (that is,  $0 < p' < p(x)$ ). Now make the change of variable

$$\begin{aligned} \frac{v}{u} &= x \\ u^2 &= p \end{aligned} \quad (7.3.19)$$

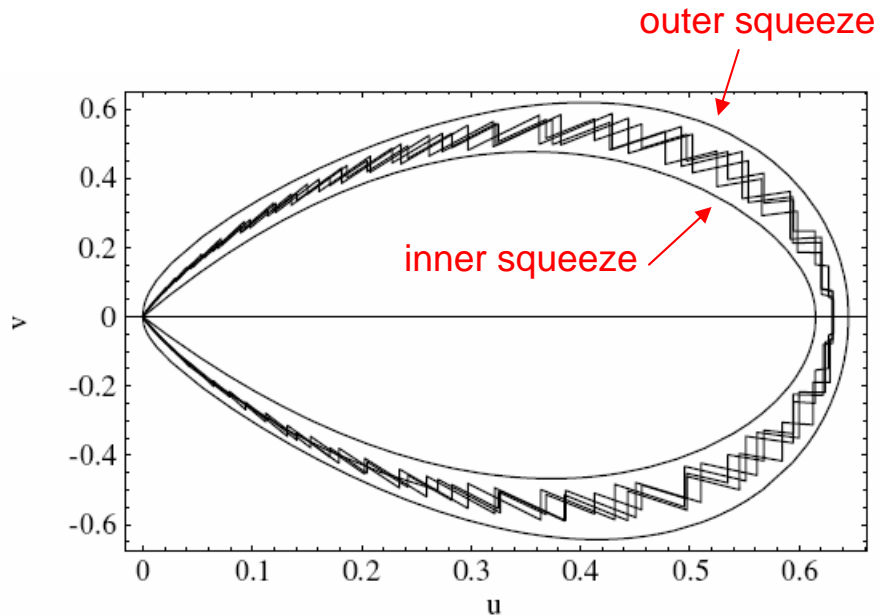
Then equation (7.3.18) becomes

$$p(x)dx = \int_{p'=0}^{p'=p(x)} dp' dx = \int_{u=0}^{u=\sqrt{p(x)}} \frac{\partial(p, x)}{\partial(u, v)} du dv = 2 \int_{u=0}^{u=\sqrt{p(v/u)}} du dv \quad (7.3.20)$$

because (as you can work out) the Jacobian determinant is the constant 2. Since the new integrand is constant, uniform sampling in  $(u, v)$  with the limits indicated for  $u$  is equivalent to the rejection method in  $(x, p)$ .

```
syms u v;
jac = jacobian([v/u u^2], [u v])
jac =
[ -v/u^2, 1/u]
[ 2*u, 0]
abs(det(jac))
ans =
2
```

Ratio of Uniforms is particularly powerful when combined with squeezes



For this particular example the desired distribution is integer valued (binomial deviates), hence the staircases. For a continuous distribution, there would just be a smooth curve between the squeezes (which would typically be too close together to see clearly).

you only compute  $p(v/u)$  when you are between the squeezes!

e.g., Leva's algorithm for normal deviates:

```
struct Normaldev : Ran {
  Structure for normal deviates.
  Doub mu,sig;
  Normaldev(Doub mmu, Doub ssig, Ullong i)
  : Ran(i), mu(mmu), sig(ssig){}
  Constructor arguments are  $\mu$ ,  $\sigma$ , and a random sequence seed.
  Doub dev() {
  Return a normal deviate.
    Doub u,v,x,y,q;
    do {
      u = doub();
      v = 1.7156*(doub()-0.5);
      x = u - 0.449871;
      y = abs(v) + 0.386595;
      q = SQR(x) + y*(0.19600*y-0.25472*x);
    } while (q > 0.27597
      && (q > 0.27846 || SQR(v) > -4.*log(u)*SQR(u)));
    return mu + sig*v/u;
  }
};
```

Just when you were getting comfortable with MATLAB, we're now going to start introducing some C++. This is an NR-style class.

here, only ~1% of the  $(u,v)$  area is between the squeezes, requiring the calculation of the log