Sampling via Transformations of the Uniform

Mihir Arjunwadkar

Centre for Modeling and Simulation
Savitribai Phule Pune University

LCG $\to$ Uniform$\langle 0, 1\rangle$

- Linear congruential generator

$$X_{n+1} = (aX_n + c) \mod M$$

All integer arithmetic: fast and efficient.

- Uniform RNs
  - Over $[0, 1)$: $U = X/M$
  - Over $[0, 1]$: $U = X/(M - 1)$
  - Over $(0, 1]$: $U = (X + 1)/M$
  - Over $(0, 1)$: $U = (X + 1)/(M + 1)$

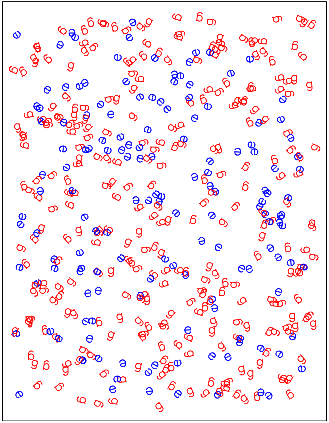
$/ \equiv$ real division.

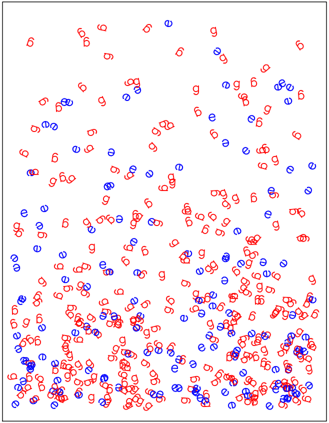
Although we pretend to be dealing with a continuous interval,

in practice, U is represented as a representable floating-point number,

and representable floating-point numbers make a discrete and finite set.

Scrambled eggs





Uniformly scrambled

Under gravity

Uniform$\langle 0, 1\rangle \rightarrow$ Uniform$\langle a, b\rangle$

Define

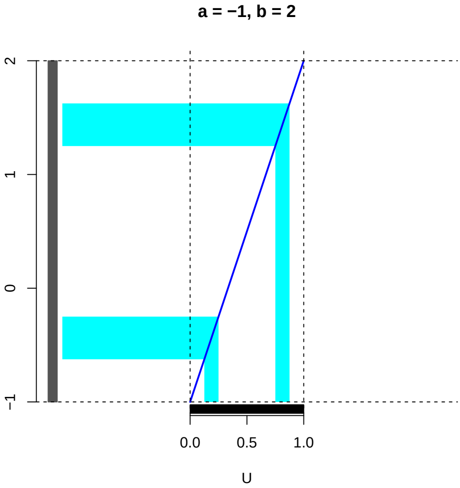
$$U' = (b - a)U + a.$$

If

$$U \sim \text{Uniform}\langle 0, 1\rangle,$$

then

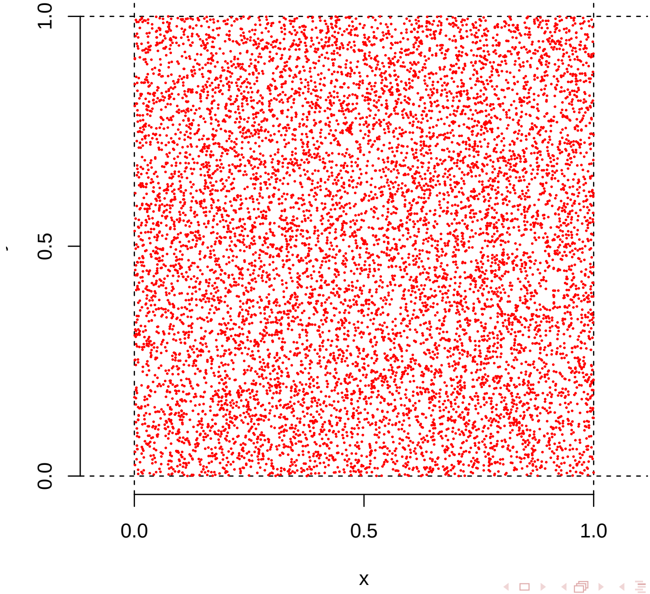
$$U' \sim \text{Uniform}\langle a, b\rangle.$$



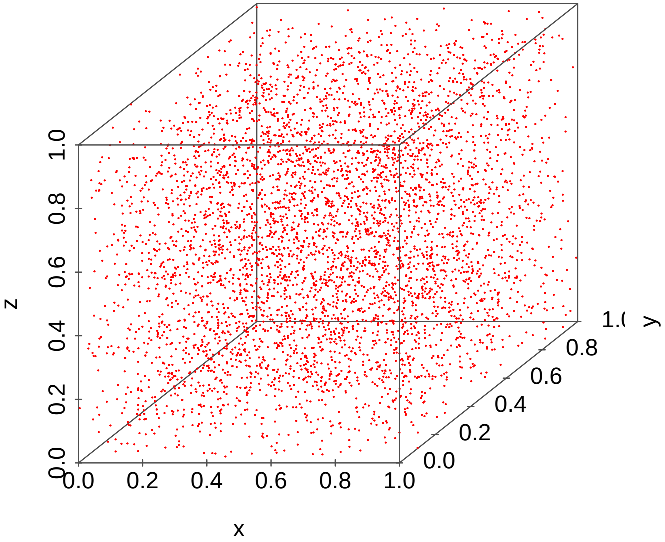
Sampling the d-dimensional hypercube $[0, 1]^d$ uniformly

- Assume independence in the Uniform[0,1] RN stream $U_1, U_2, \ldots$.
- Consider successive d-tuples as points in the d-dimensional space.
- Under assumption of independence, the joint density of d-tuples $(U_1, U_2, \ldots, U_d)$ is uniform over $[0, 1]^d$.

Sampling the d-dimensional hypercube $[0,1]^d$ uniformly



Sampling the d-dimensional hypercube $[0, 1]^d$ uniformly



Uniform$[0,1) \to$ UniformInteger$[0, k-1]$

Let

$$I = \lfloor kU \rfloor.$$

If

$$U \sim \text{Uniform}[0,1),$$

then

I has discrete uniform distribution over the integer set $\{0, \ldots, k-1\}$.

$\lfloor x \rfloor$: *floor* of x, i.e., largest integer $\leq x$.

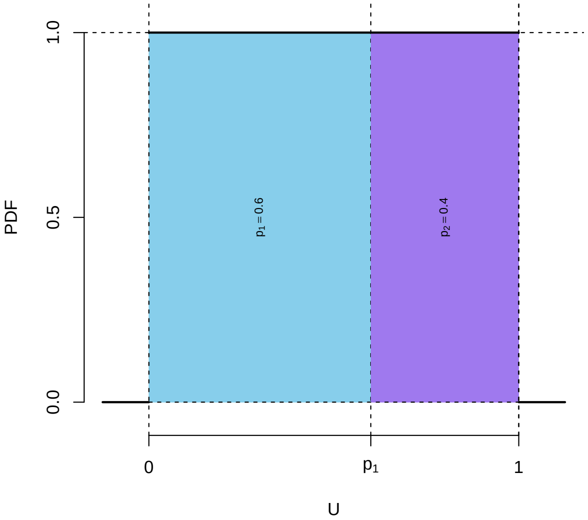
Under the $\lfloor kU \rfloor$ operation, probabilities accumulate at integer values $\{0, \ldots, k-1\}$. Visually,

$$[\longleftarrow)[\longleftarrow)[\longleftarrow)[\longleftarrow)[\longleftarrow)[\longleftarrow)$$

It may be best to do this using an LCG itself, if possible.

Discrete random variable with prespecified PMF

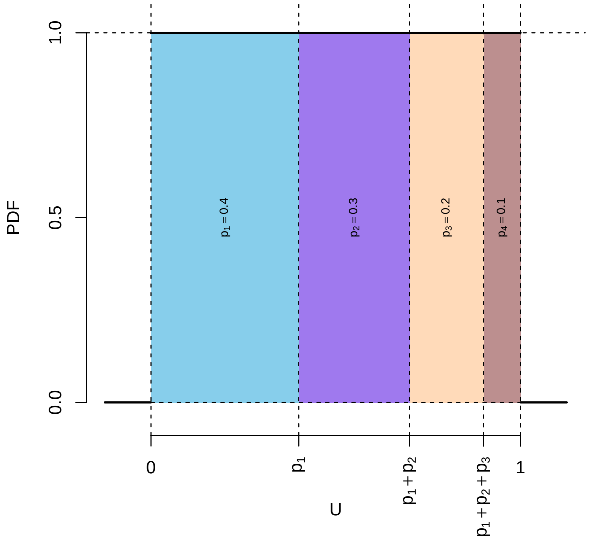
Biased coin toss: Bernoulli(p) with $p = 0.6$



- Partition $[0,1]$ into two events with probabilities 0.6 and 0.4.
- Generate $u \sim \text{Uniform}[0,1]$.
- Outcome: Head if $u \leq 0.6$, and Tail otherwise

Discrete random variable with prespecified PMF

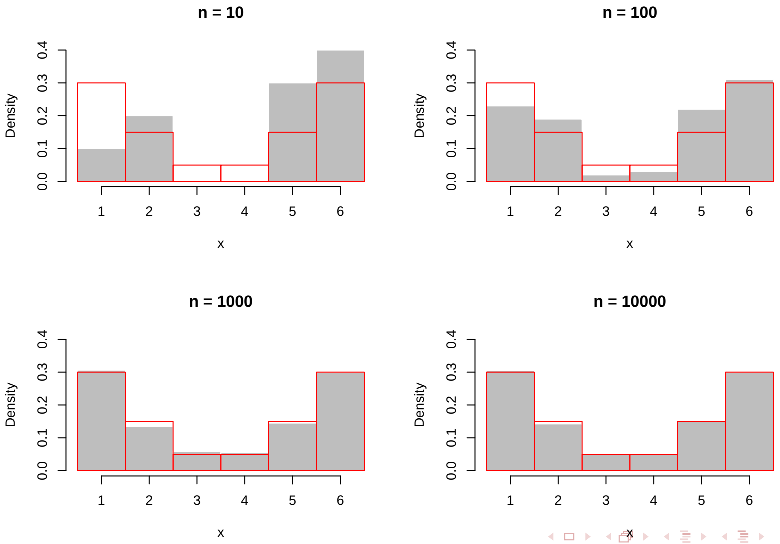
Four states that occur with probabilities $p = (0.4, 0.3, 0.2, 0.1)$



- Partition $[0, 1]$ into 4 events with given probabilities.
- Generate $u \sim \text{Uniform}[0, 1]$.
- Outcome $= 1$ if RN $u \leq p_1$, 2 if $p_1 < u \leq p_1 + p_2$, 3 if $p_1 + p_2 < u \leq p_1 + p_2 + p_3$, and 4 otherwise.
- Use fast look-up methods to ensure efficiency.

Discrete random variable with prespecified PMF

Empirical proportions of $1, \ldots, 6$ for a non-uniform 6-faced die
$p = (0.3, 0.15, 0.05, 0.05, 0.15, 0.3)$



More examples of discrete sampling problems

Specific and general methods for a wide range of sampling problems exist.

- Binomial, Poisson, Geometric, Multinomial, ...
- Sampling without replacement, random permutation, ...

Brian D. Ripley, *Stochastic Simulation*, Wiley (1987)
Luc Devroye, *Non-uniform Random Variate Generation*, Springer (1986)

Uniform$\langle 0, 1 \rangle \rightarrow$ continuous univariate distributions

Given the ability to generate Uniform$\langle 0, 1 \rangle$ RNs,
how do we generate RNs distributed as
some continuous univariate distribution f?

Notation

- $U \sim \text{Uniform}\langle 0, 1\rangle$.
- PDF $f_X(x)$: Target PDF which we wish to sample from.
- CDF $F_X(x) = P(X \leq x) = \int_{-\infty}^{x} f_X(t)dt$. $f_X(x) = \frac{dF_X(x)}{dx}$.
- $F_X(x)$ is assumed 1-to-1 and hence invertible.
- $F_X^{-1}(x)$ is called the *quantile function*.

For invertible F_X:

$F_X^{-1}(F_X(x)) = x$ and $F_X\left(F_X^{-1}(u)\right) = u$.

Sampling prescription

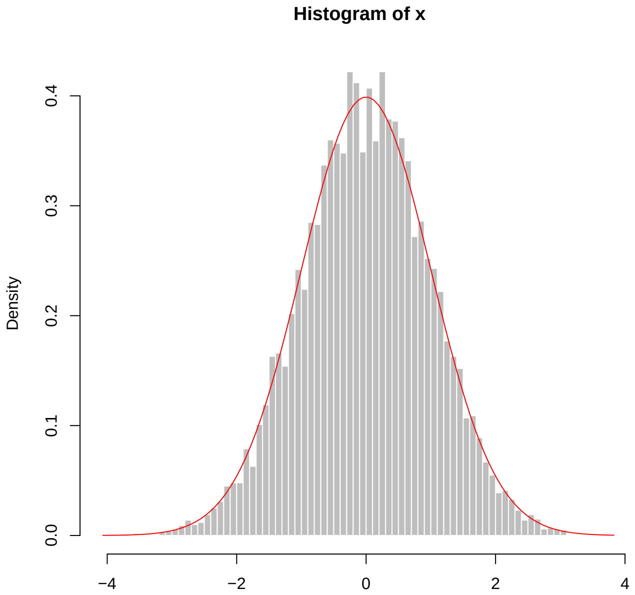
- Generate $U_1, \ldots, U_N \sim \text{Uniform}[0, 1]$
- $X_i = F_X^{-1}(U_i)$

Claim

$X_1, \ldots, X_N \sim f_X$

If this claim is correct, then this prescription amounts to, e.g.,

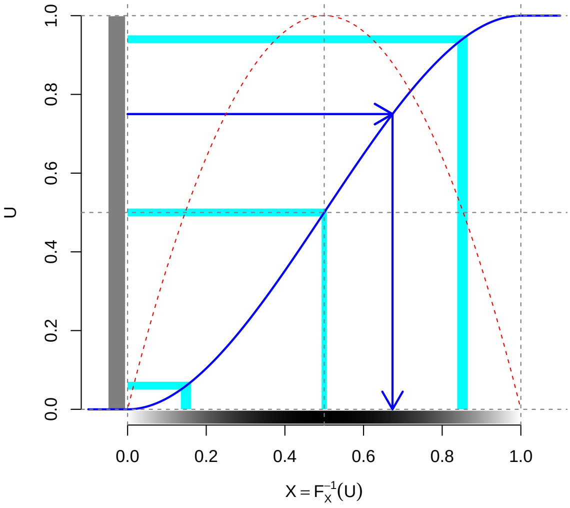
```
# target PDF function
pdf <- 'norm' # any distribution for which dpqr quartet is available

# target {dq} functions
qf <- get( paste( 'q', pdf, sep = '' ) )
df <- get( paste( 'd', pdf, sep = '' ) )

# target RNs
x <- qf( runif( 10000 ) ) # <<<<<<<<<<<<

# visual verification via histogram
pdf( 'inverse-cdf-method.pdf' )
 hist( x, 'FD', freq = FALSE, col = 'gray', border = 'white' )
 curve( df, from = min( x ), to = max( x ), add = TRUE, col = 'red' )
dev.off()
```



If the previous claim is correct, we expect to see something like this

How intervals map under $F_X^{-1}(U)$

- Beta$(2, 2)$: $f_X(x) \propto x(1 - x)$
- Uniform U-density $\rightarrow$ X-density concentrated around $X = 0.5$

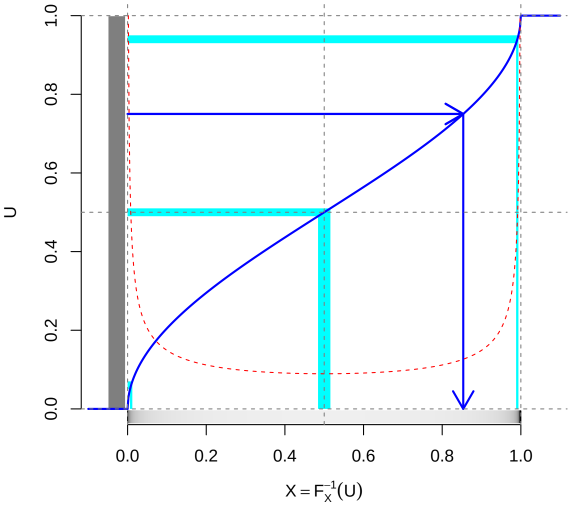


How intervals map under $F_X^{-1}(U)$

- Beta($\frac{1}{2}, \frac{1}{2}$):

  $f_X(x) \propto \frac{1}{\sqrt{x(1-x)}}$

- Uniform U-density $\rightarrow$ X-density concentrated around $X = 0, 1$



Why would this work?

Argument based on the CDF

with some abuse of notation!

Let $X = F_X^{-1}(U)$. Then,

$$\begin{aligned}
\text{CDF of } X \text{ at } x &= P\{X \leq x\} \\
&= P\left\{F_X^{-1}(U) \leq x\right\} \\
&= P\{U \leq F_X(x)\} \\
&= \text{CDF of Uniform[0,1] evaluated at } F_X(x) \\
&= F_X(x)
\end{aligned}$$

Last step follows from
$P(U \leq u) = u$ for $0 \leq u \leq 1$, and that $0 \leq F_X(x) \leq 1$.

Therefore, $X = F_X^{-1}(U)$ has the desired PDF $f_X(x)$ and CDF $F_X(x)$.

Why would this work?

Argument based on RV transformation theory

- Let $X = r(U)$
- If r is strict monotone (increasing or decreasing), hence 1-to-1, then it is invertible; i.e., there exists r^{-1} such that $r^{-1}(r(u)) = u$.
- Then

$$f_X(x) = f_U(r^{-1}(x)) \left| \frac{dr^{-1}(x)}{dx} \right| .$$

- $f_U(\cdot) = 1$ (ignoring $f_U(u) = 0$ when $u < 0$ or $u > 1$)

$$\longrightarrow \quad f_X(x) = \frac{dr^{-1}(x)}{dx} \text{ (ignoring sign)}$$

$$\longrightarrow \quad \int^x f_X(x')dx' = r^{-1}(x) \text{ (notice that LHS} = F_X(x))$$

$$\longrightarrow \quad r(\cdot) = F_X^{-1}(\cdot)$$

Rigorous/detailed proof possible.

Illustration: $\text{Exp}(\lambda)$

- Target PDF: $f_X(x) = \lambda^{-1} \exp\left(-\frac{x}{\lambda}\right) \quad \text{for } x \geq 0$
- Find the CDF: $F_X(x) = 1 - \exp\left(-\frac{x}{\lambda}\right) = r^{-1}(x) = u$
- Solve $u = 1 - \exp\left(-\frac{x}{\lambda}\right)$ for $x \implies x = -\lambda \log(1 - u)$

The required $U \to X$ transformation is

$$X = r(U) = -\lambda \log(1 - U) \equiv -\lambda \log(U)$$

The last $\equiv$ equivalence: U and $1 - U$ are both $\text{Uniform}\langle 0, 1\rangle$.

- Target PDF: $f_X(x) = \frac{1}{\pi(1+x^2)}$ for $-\infty < x < +\infty$
- Find the CDF: $F_X(x) = \frac{1}{2} + \frac{1}{\pi}\tan^{-1}(x) = r^{-1}(x) = u$
- Solve $u = \frac{1}{2} + \frac{1}{\pi}\tan^{-1}(x)$ for $x \implies x = \tan\left(\pi\left(u - \frac{1}{2}\right)\right)$

The required $U \rightarrow X$ transformation is

$$X = r(U) = \tan\left(\pi\left(U - \frac{1}{2}\right)\right)$$

Illustration: Weibull(λ, k)

- Target PDF: $f_X(x) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} \exp\left(-\left(\frac{x}{\lambda}\right)^{k}\right) \quad \text{for } x \geq 0$
- Find the CDF: $F_X(x) = 1 - \exp\left(-\left(\frac{x}{\lambda}\right)^{k}\right) = r^{-1}(x) = u$
- Solve $u = 1 - \exp\left(-\left(\frac{x}{\lambda}\right)^{k}\right)$ for $x \implies x = \lambda\left(-\log(1-u)\right)^{1/k}$

The required $U \to X$ transformation is

$$X = r(U) = \lambda\left(-\log(1-U)\right)^{1/k} \equiv \lambda\left(-\log(U)\right)^{1/k}$$

The last $\equiv$ equivalence: U and $1-U$ are both Uniform$\langle 0, 1\rangle$.

- Target PDF: $f_X(x) = \frac{\alpha}{x}\left(\frac{x_m}{x}\right)^{\alpha}$ for $x \geq x_m$
- Find the CDF: $F_X(x) = 1 - \left(\frac{x_m}{x}\right)^{\alpha} = r^{-1}(x) = u$
- Solve $u = 1 - \left(\frac{x_m}{x}\right)^{\alpha}$ for $x \implies x = x_m\,(1-u)^{-\frac{1}{\alpha}}$

The required $U \to X$ transformation is

$$X = r(U) = x_m\,(1-U)^{-\frac{1}{\alpha}} \equiv x_m U^{-\frac{1}{\alpha}}$$

The last $\equiv$ equivalence: U and $1-U$ are both Uniform$\langle 0, 1\rangle$.

When is inverse-CDF sampling useful?

- For the transformation method / inverse-CDF sampling to work, it should be possible to compute $F_X^{-1}(x)$
  - efficiently; and
  - in a numerically stable fashion.
- Hence, this method is useful when a closed-form expression for the inverse CDF $F_X^{-1}(x)$ is available and is easy to compute.

When is inverse-CDF sampling useful?

This rules out many important distributions that (generally) have no closed-form expressions for CDF or its inverse.

- Normal(0,1)

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{x^2}{2\sigma^2}\right\}, \quad -\infty < x < +\infty$$

- Beta(α, β)

$$f_X(x) \propto x^{\alpha-1}(1-x)^{\beta-1}, \quad 0 \leq x \leq 1$$

Except for special parameters values such as $\alpha = \beta = 2$ or $1/2$.

- Gamma(k, θ)

$$f_X(x) = \frac{\left(\frac{x}{\theta}\right)^{k-1} \exp\left(-\frac{x}{\theta}\right)}{\theta\Gamma(k)}, \quad x \geq 0; k, \theta > 0$$

Univariate Normal(0,1) via the Box-Müller transformation

- Inverse-CDF method cannot be applied directly to the univariate case.

- Fortunately, there is a 2-dimensional transformation for sampling a pair of Normal(0,1) RVs.

- Given $U_1, U_2 \sim \text{Uniform}(0,1]$, consider the forward transformation $(U_1, U_2) \rightarrow (X_1, X_2)$:

$$\begin{aligned} X_1 \equiv X_1(U_1, U_2) &= \sqrt{-2\log(U_1)}\cos(2\pi U_2) \\ X_2 \equiv X_2(U_1, U_2) &= \sqrt{-2\log(U_1)}\sin(2\pi U_2) \end{aligned}$$

- The reverse transformation $(X_1, X_2) \rightarrow (U_1, U_2)$ is

$$\begin{aligned} U_1 \equiv U_1(X_1, X_2) &= \exp\left(-\frac{1}{2}\left(X_1^2 + X_2^2\right)\right) \\ U_2 \equiv U_2(X_1, X_2) &= \frac{1}{2\pi}\tan^{-1}\left(\frac{X_2}{X_1}\right) \end{aligned}$$

Normal(0,1) via the Box-Müller transformation

- Because the transformation is 1-1 and invertible, the joint PDF of (X_1, X_2) is

$$f_{X_1,X_2}(x_1,x_2) = f_{U_1,U_2}(U_1(x_1,x_2), U_2(x_1,x_2)) \begin{vmatrix} \frac{\partial U_1}{\partial X_1} & \frac{\partial U_1}{\partial X_2} \\ \frac{\partial U_2}{\partial X_1} & \frac{\partial U_2}{\partial X_2} \end{vmatrix}$$

- $f_{U_1,U_2}(\cdot,\cdot) = 1$ (ignore cases u_1 or $u_2 < 0$, or u_1 or $u_2 > 1$)

- Simplification of the |Jacobian determinant| leads to

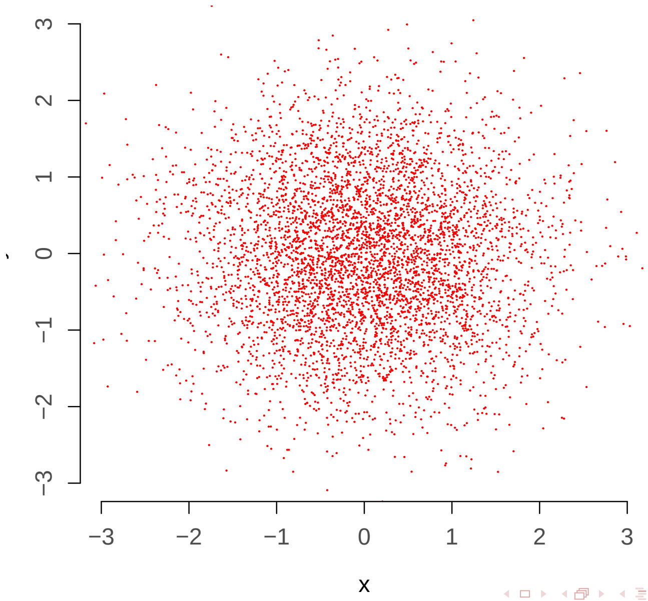
$$f_{X_1,X_2}(x_1,x_2) = \left[\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x_1^2}{2}\right)\right] \times \left[\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x_2^2}{2}\right)\right]$$

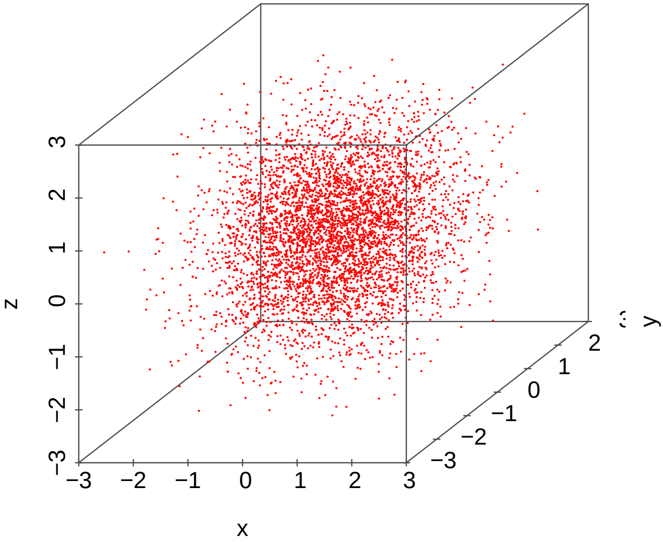
- Hence, $f_{X_1,X_2}(\cdot,\cdot) \equiv$ joint PDF of two *independent* Normal(0,1) RVs.

- Therefore, X_1 and X_2 both have Normal(0,1) PDFs.

Proof above is indicative; subtleties are ignored.

IID bivariate normal scatter




z
3
2
1
0
−1
−2
−3
x
−3
−2
−1
0
1
2
3
y
−3
−2
−1
0
1
2

$$f_X(x) = 0.7 \times \phi(x; 0, 1) + 0.3 \times \phi(x; 2.5, 1)$$

Sampling from a normal mixture

A k-component normal/Gaussian mixture has PDF of the form

$$f_X(x) = \sum_{i=1}^{k} \omega_i \phi(x; \mu_i, \sigma_i)$$

where

- ω_i: weight of the ith component ($\omega_i \geq 0$ and $\sum_{j=1}^{k} \omega_j = 1$).
- μ_i: mean of the ith normal component.
- σ_i: standard deviation of the ith normal component ($\sigma_i > 0$).
- $\phi(x; \mu, \sigma)$: Normal(μ, σ) PDF.

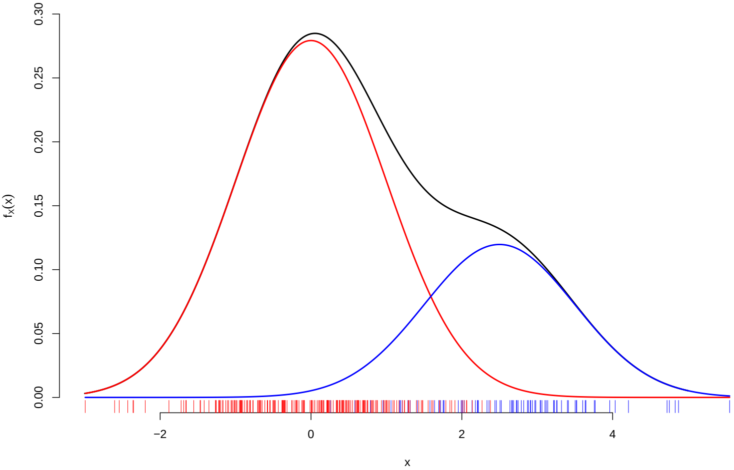
$\{\omega_1, \ldots, \omega_k\}$ can be thought of as a discrete PMF ($\equiv$ categorical distribution).

Algorithm

1. Sample a component from $\{1, \ldots, k\}$ using probabilities $\{\omega_1, \ldots, \omega_k\}$. Suppose that this randomly sampled component is the ith.
2. Sample x from $\phi(\cdot; \mu_i, \sigma_i)$.
3. Repeat steps 1 & 2 as many times as required.

This can be generalized to other mixture PDFs straightforwardly.

$$f_X(x) = 0.7 \times \phi(x; 0, 1) + 0.3 \times \phi(x; 2.5, 1)$$



Sampling from multivariate normal

Multivariate Normal(μ, Σ) PDF

$$f_{\mathbf{X}}(\mathbf{x}) = \left((2\pi)^k \det(\Sigma)\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)\right)$$

where

- $\mathbf{x}$: argument, k-vector
- μ: mean, k-vector
- Σ: $k \times k$ symmetric positive definite variance-covariance matrix

Compare with the univariate normal density
$f_X(x) = (2\pi)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)\frac{1}{\sigma^2}(x-\mu)\right)$

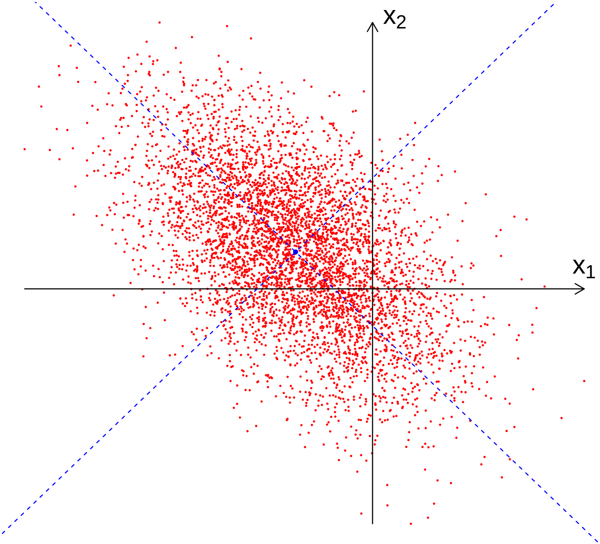
Sampling from multivariate normal

1. Find a matrix A such that $\Sigma = AA^T$
   A common choice: Cholesky factorization of Σ
2. Generate random vector $\mathbf{z} = (z_1, \ldots, z_k)$ of IID N(0,1) random numbers
3. Compute $\mathbf{x} = A\mathbf{z} + \mu$ :: $\mathbf{x} \sim \text{Normal}(\mu, \Sigma)$
4. Repeat steps 2-3 as many times as required

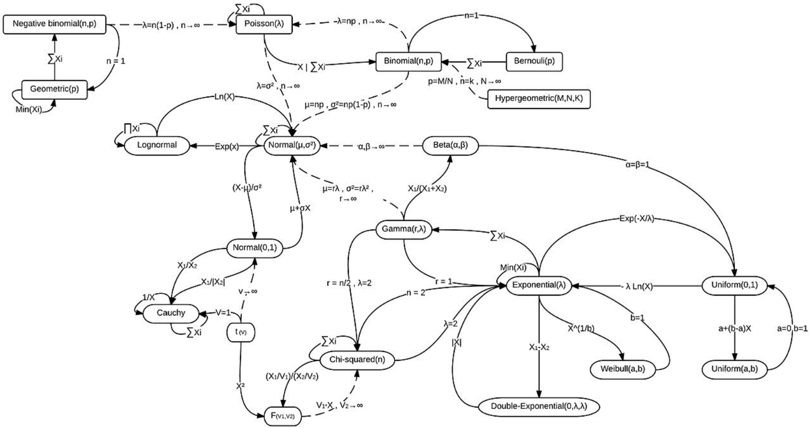
An R implementation

```
mvrnorm <- function( n, mu, Sigma )
 {
  # Multivariate normal random vectors via Cholesky factorization.
  # Assumption: Sigma is a symmetric positive definite matrix.

  stopifnot( is.matrix( Sigma ), is.vector( mu ),
             ncol( Sigma ) == nrow( Sigma ),
             length( mu )  == nrow( Sigma ) )

  k <- length( mu )
  A <- t( chol( Sigma ) ) # assume symmetric positive definite Sigma

  t( replicate( n, c( A %*% rnorm( k ) + mu ) ) ) # n X k matrix
 }
```

A random sample from Normal$\left(\left(-1, +\frac{1}{2}\right), \begin{bmatrix} +1 & -\frac{1}{2} \\ -\frac{1}{2} & +1 \end{bmatrix} \right)$



In principle, one can try to exploit connections between probability distributions to devise RNGs



In practice, computational effort can be too much, and better alternatives may be available (or need to be devised).

Examples

- From Normal(0,1) to χ_p^2 (bad idea)
  $X = \sum_{i=1}^{p} Z_i^2 \sim \chi_p^2$ if $Z_1, \ldots, Z_p \sim \text{Normal}(0, 1)$.
- From Normal(0,1) to t_p (bad idea)
  $X = \frac{Z}{\sqrt{Y/p}} \sim t_p$ if $Z \sim \text{Normal}(0, 1)$ and $Y \sim \chi_p^2$.
- From Normal(0,1) to Log-Normal(μ, σ)
  $X = e^{\mu + \sigma Z} \sim \text{Log-Normal}(\mu, \sigma)$ if $Z \sim \text{Normal}(0, 1)$.

References

- James E. Gentle, *Random Number Generation and Monte Carlo Methods*, Springer (2003).
- Brian D. Ripley, *Stochastic Simulation*, Wiley (1987).
- Luc Devroye, *Non-uniform Random Variate Generation*, Springer (1986). http://luc.devroye.org/rnbookindex.html
- Donald Knuth, *The Art of Computer Programming, Vol 2: Seminumerical Algorithms*, Addison-Wesley (1981).