Random number generation

Quantitative Risk Management project work

Silvia Baldisserotto, Maysa Jahanbani,
Claudia Pesci, Phan Ho Tan Phat,
Andrea Venuta

Università degli Studi di Firenze - Finance and Risk Management

Random numbers

- Computer-generated numbers are *pseudo-random*: deterministic and predictable
- *Quasi-random* numbers prevent potential lack of equidistributedness
- **Definition.** *(Sample).* A sequence of number is called a *sample from the distribution F* if the numbers are independent realizations of a random variable with distribution function F
- Uniform deviates: samples from $\sim \mathcal{U}\left[0,1\right]$
- Normal deviates: samples from $\sim \mathcal{N}\left(0,1\right)$
- Drawing uniform deviates is the basis of random number generation

Linear congruential generators

- N_0 is chosen arbitrarily (called the *seed*)
- $N_i = (aN_{i-1} + b) \bmod M$ for $i > 0$, then

$$U_i = \frac{N_i}{M}, \quad U_i \in [0, 1)$$

- Suitability of the numbers U_i depends on how a, b, M are chosen

Linear congruential generators: properties

- Numbers N_i are periodic, with period $\leq M$:
  there are at most M different numbers in the class modulo M
- Examples:
  - If $N = 0$, b can't be 0, otherwise $N_i = 0$ will repeat itself
  - If $a = 0$, generator settles down on $N_n = N_0 + nb$
- Numbers are distributed "evenly" if we have exactly M different numbers in a generator with modulo M, or
- Each grid point on a *mesh* on $[0, 1]$ with size $\frac{1}{M}$ is occupied once

Quality of generators

Requirements:

1. Large period: small set of numbers makes the outcome easier to predict (choose M as large as possible)
2. Statistical tests to verify that the distribution is the intended one
    - Comparison of sample mean and variance μ, σ^2 with desired values
    - Correlation between sample values
    - Quality of approximation of the distribution
3. Distribution in higher dimensional spaces: lattice structure

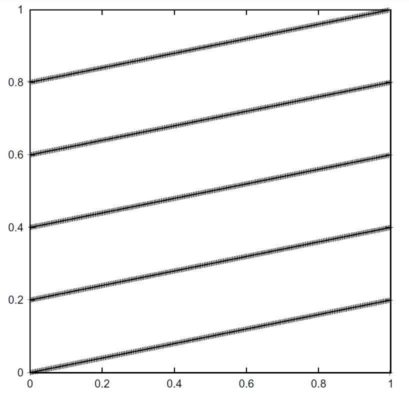
Random vectors and lattice structure

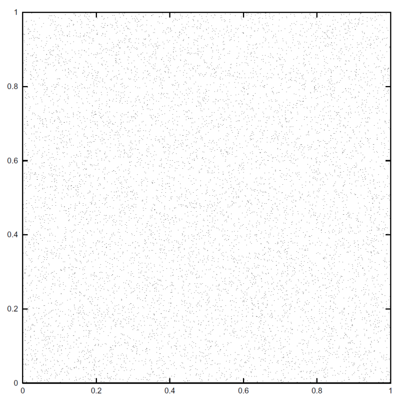
- Sequences of random numbers can be arranged in m-dimensional vectors
- The vectors lie on a number of parallel $(m-1)$-dimensional hyperplanes
- The ideal condition is that the number of parallel hyperplanes is maximized: number of hyperplanes is a measure of equidistributedness
- Family of parallel lines in the (U_{i-1}, U_i)-plane

$$z_0 U_{i-1} + z_1 U_i = c + \frac{z_1 b}{M} \quad \text{where} \quad c := N_{i-1} \frac{z_0 + a z_1}{M} - z_1 k$$

for each tuple (z_0, z_1) and for all cs.

Random vectors and lattice structure





Inversion and transformation methods

Inversion and transformation methods generate numbers distributed according to an arbitrary distribution from uniformly distributed samples.

Inversion method

Theorem. *(inversion)* Suppose $U \sim \mathcal{U}[0,1]$, and F continuous strictly increasing distribution. Then, $F^{-1}(U)$ is a sample from F.

Proof.

$$\mathbb{P}(U \leq \xi) = \xi,\ 0 < \xi < 1$$
$$\mathbb{P}(F^{-1}(U) \leq x) = \mathbb{P}(U \leq F(x)) = F(x).$$

Exponential distribution:

$$F(x) = 1 - e^{-\lambda x}$$
$$F^{-1}(x) = -\frac{1}{\lambda}\log(x)$$

Cauchy distribution:

$$F(x) = \frac{1}{\pi}\arctan(x) + \frac{1}{2}$$
$$F^{-1}(x) = \tan\left(\pi\left(x - \frac{1}{2}\right)\right)$$

Transformation method

Theorem. If X is a $r.v. \sim F(x)$, and $h : S \to B,\ S, B \cup \mathbb{R}$ strictly monotonous, then:

- $Y := h(X)$ is a $r.v.$ with distribution

$$
\begin{aligned}
F_Y(y) &= F(h^{-1}(y)) \quad h' > 0 \\
F_Y(y) &= 1 - F(h^{-1}(y)) \quad h' < 0
\end{aligned}
$$

- If h^{-1} absolutely continuous for almost all y, density of h is

$$
f(h^{-1}(y)) \left| \frac{dh^{-1}(y)}{dy} \right|
$$

Transformation method: Exponential distribution

$$
\begin{aligned}
F(x) &= 1 - e^{-\lambda y} \\
F^{-1}(y) &= \frac{\ln(y-1)}{\lambda} \\
F^{-1}(U) &= h(U) = \frac{\ln(U)}{\lambda} \\
h^{-1}(U) &= e^{-\lambda U}
\end{aligned}
$$

$$
f(h^{-1}(U))\left|\frac{dh^{-1}(y)}{dy}\right| = 1 \cdot \left|-\lambda e^{-\lambda y}\right| = \lambda e^{-\lambda y}
$$

Transformation method: Cauchy distribution

$$F(x) = \frac{1}{\pi}\arctan(x) + \frac{1}{2}$$
$$F^{-1}(y) = \tan\left(\pi\left(y - \frac{1}{2}\right)\right)$$
$$F^{-1}(U) = h(U) = \tan(\pi U)$$
$$h^{-1}(U) = \frac{\arctan(U)}{\pi}$$

$$f(h^{-1}(U))\left|\frac{dh^{-1}(y)}{dy}\right| = 1 \cdot \left|\frac{1}{\pi(1+x^2)}\right| = \frac{1}{\pi(1+x^2)}$$

Box-Muller method

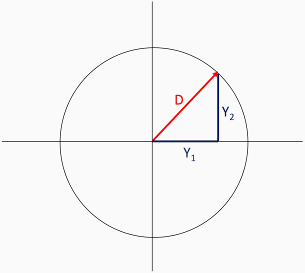
- Generate $x_1, x_2 \sim \mathcal{U}(0,1)$ random numbers
- Derive

$$h_1(x_1, x_2) := y_1 = \sqrt{-2\log x_1}\cos 2\pi x_2$$

$$h_2(x_1, x_2) := y_2 = \sqrt{-2\log x_1}\sin 2\pi x_2$$

- y_1 and y_2 will be i.i.d. $\sim \mathcal{N}(0,1)$

Box-Muller method



$$y_1 = D\cos\omega, \quad y_2 = D\sin\omega$$

where $D = \sqrt{-2\log x_1}, \quad \omega = 2\pi x_2$

$$h^{-1}(x_1, x_2) = \begin{cases} x_1 = \exp\left\{-\frac{y_1^2+y_2^2}{2}\right\} \\ x_2 = \frac{1}{2\pi}\arctan\frac{y_2}{y_1} \end{cases}$$

$$|\text{Jacobian}| = \det\begin{pmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{pmatrix} = \left[\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{y_1^2}{2}\right)\right]\cdot\left[\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{y_2^2}{2}\right)\right]$$

is the density of the bivariate standard normal distribution because it's the product of two univariate standard normal densities.

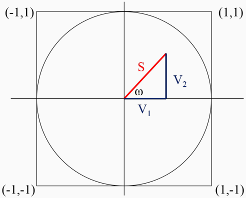
Polar method

1. Let $U_1, U_2 \sim \mathcal{U}(0,1)$
2. Define $V_i = 2U_i - 1$: $V_i \sim \mathcal{U}(-1,1)$
3. Define $S = V_1^2 + V_2^2$
4. If and only if $S \leq 1$, then define

$$Y = \sqrt{\frac{-2\ln S}{S}}$$

5. $\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} V_1 Y \\ V_2 Y \end{pmatrix}$,

   X_1, X_2 i.i.d. $\sim \mathcal{N}(0,1)$



$$\begin{aligned} x_2 &= \frac{1}{2\pi}\arg(V_1, V_2) \\ &= \frac{1}{2\pi}\arctan\left(\frac{V_2}{V_1}\right) \end{aligned}$$

$$\cos 2\pi x_2 = \frac{V_1}{\sqrt{V_1^2 + V_2^2}}$$

$$\sin 2\pi x_2 = \frac{V_2}{\sqrt{V_1^2 + V_2^2}}$$

Correlated bivariate random variables

$$\bar{Z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}, \; z_1, z_2 \sim \mathcal{N}(0,1) \quad \bar{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \quad \Sigma = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$$

1. Calculate the Cholesky decomposition $AA^T = \Sigma$

$$\begin{pmatrix} a & 0 \\ b & c \end{pmatrix} \begin{pmatrix} a & b \\ 0 & c \end{pmatrix} = \begin{pmatrix} a^2 & ab \\ ab & b^2 + c^2 \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$$

$$\rightarrow A = \begin{pmatrix} \sigma_1 & 0 \\ \rho\sigma_2 & \sigma_2(1-\rho^2)^{\frac{1}{2}} \end{pmatrix}$$

2. Calculate $\bar{Z} \sim \mathcal{N}(0, \mathbb{I}_2)$
3. $\mu + A\bar{Z} \sim \mathcal{N}(\mu, \Sigma)$ has the desired distribution.

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \bar{\mu} + \begin{pmatrix} \sigma_1 & 0 \\ \rho\sigma_2 & \sigma_2(1-\rho^2)^{\frac{1}{2}} \end{pmatrix} \bar{Z} = \bar{\mu} + \begin{pmatrix} \sigma_1 z_1 \\ \rho\sigma_2 z_1 + \sigma_2(1-\rho^2)^{\frac{1}{2}} z_2 \end{pmatrix}$$

Implementations - Linear Congruential Generator

```
function [ rn ] = LCG( x )

  if(nargin == 0)
    x = 1;
  end

  rn = zeros(x,1);

  for i = 1:x
    rn(i) = LCGstep();
  end

end
```

```
function [ rnStep ] = LCGstep()

  persistent seed;
  M = 244944;
  a = 1597;
  b = 51749;

  if(isempty(seed))
    seed = 0;
  end

  seed = mod(seed * a + b, M);

  rnStep = seed / M;

end
```

Implementations - Box-Muller method

```
function [ Z ] = BoxMuller( x )

  if(nargin == 0)
    x = 1;
  end

  U = rand(x, 2);

  theta = 2 .* pi .* U(:, 2);
  rho   = sqrt( -2 .* log( U(:, 1) ) );

  Z = [ rho .* cos(theta), rho .* sin(theta) ];

end
```

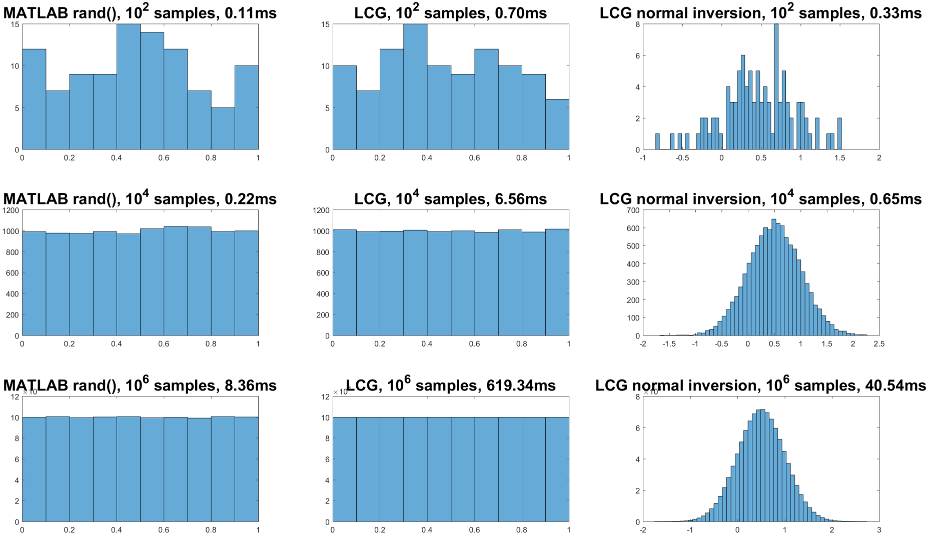
Implementations - Marsaglia polar algorithm

```
function [ Z ] = Marsaglia( x )

  if(nargin == 0)
    x = 1;
  end

  Z = zeros(x,2);

  for i = 1 : x
    W = 1;  V = [ 1, 1 ];
    while not (W < 1)
      V = 2 * rand(1, 2) - 1;
      W = V(1) .^ 2 + V(2) .^ 2;
    end

    Z(i, :) = V .* sqrt(-2 * log(W) / W);
  end
end
```

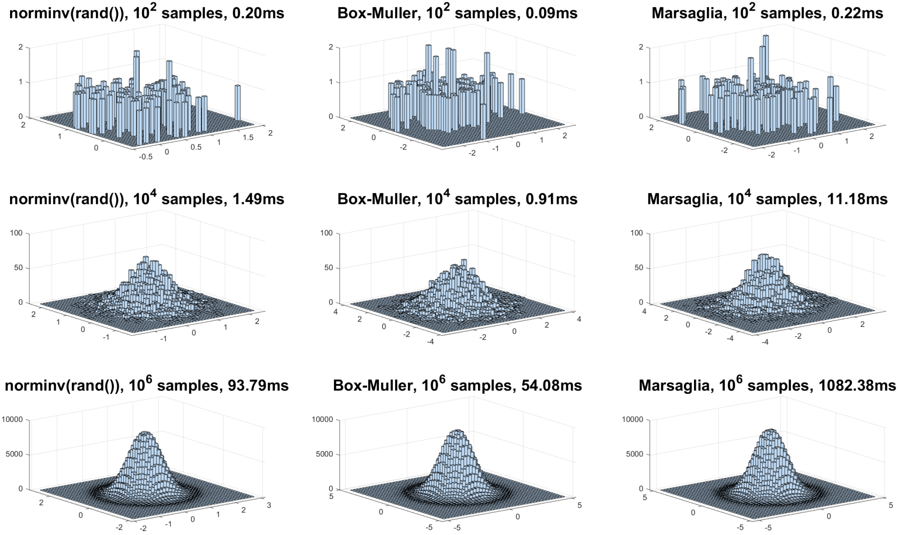
Implementations - Correlated r.v. algorithm

```
function [ Zc ] = CorrelatedRV( x, mu, Sigma )

  if(nargin == 0)
    x = 1;
  end

  A = chol(Sigma);
  Z = BoxMuller(x);

  mu = mu(:);
  Zc = zeros(x,2);

  for i = 1:x
    Zc(i,:) = mu + (A * Z(i,:)');
  end

end
```

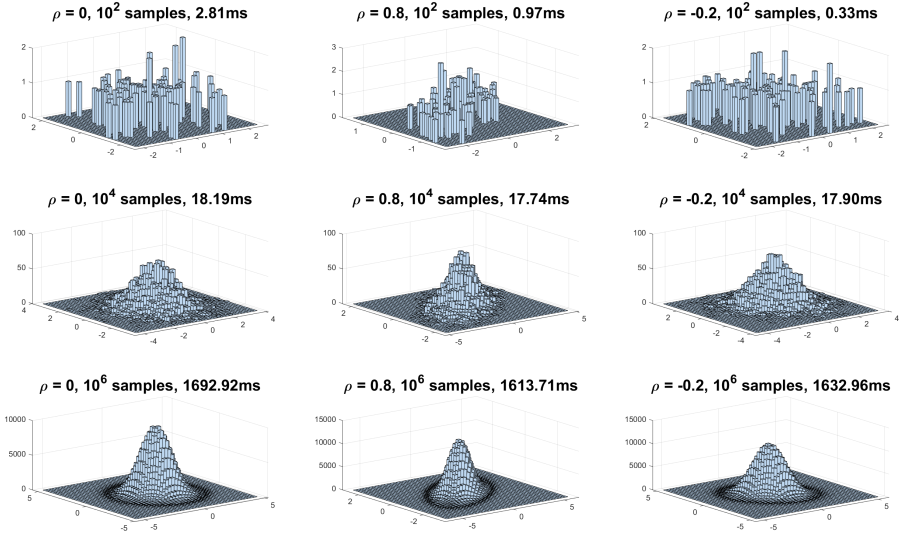
Plots - Univariate methods



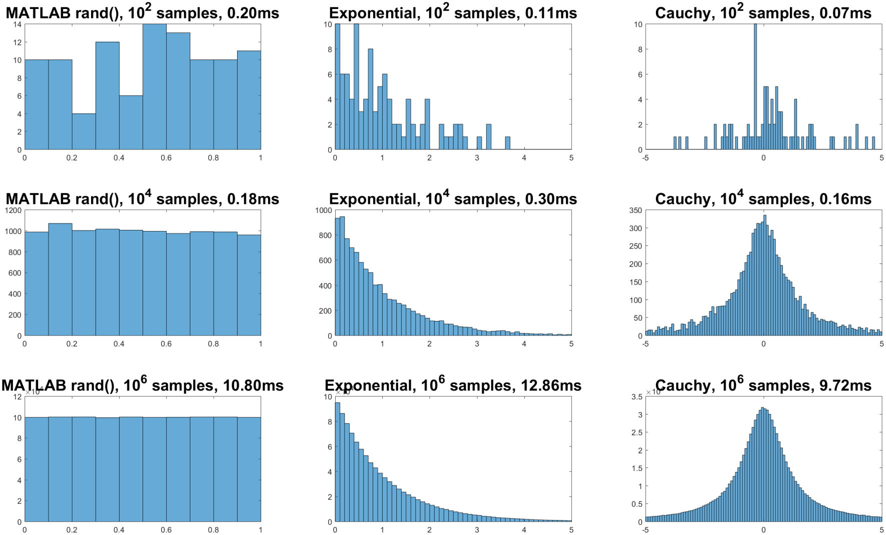
Plots - Bivariate methods



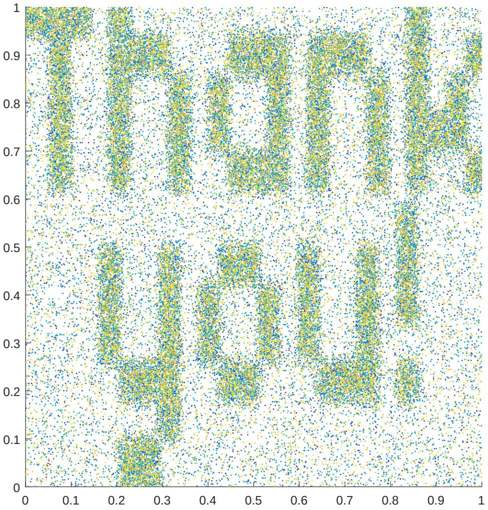
Plots - Correlated normal r.v.



Plots - Inversion method on Exponential and Cauchy



...




```
h=25;w=46;I=cumsum([1,24;0,1;1,-1;0,1;1,-9;0,1;0,1;0,1;
0,1;0,1;0,1;0,1;0,1;0,1;1,-9;0,1;0,1;0,1;0,1;0,1;0,1;0,
1;0,1;0,1;1,-1;0,1;1,-1;0,1;2,-18;0,1;0,1;0,1;0,1;0,1;1
,-5;0,1;0,1;0,1;0,1;0,1;0,4;0,1;0,1;0,1;0,1;0,1;0,1;0,1
;0,1;0,1;1,-24;0,1;0,3;0,1;0,10;0,1;0,1;0,1;0,1;0,1;0,1
;0,1;0,1;0,1;1,-24;0,1;0,3;0,1;0,16;0,1;1,-22;0,1;0,3;0
,1;0,16;0,1;1,-22;0,1;0,3;0,1;0,16;0,1;1,-20;0,1;0,1;0,
1;0,1;0,1;0,1;0,1;0,1;0,10;0,1;1,-20;0,1;0,1;0,1;0,
1;0,1;0,1;0,1;0,1;0,4;0,1;0,1;0,1;0,1;0,1;1,-5;0,1;
0,1;0,1;0,1;0,1;2,-14;0,1;0,1;0,1;1,-3;0,1;0,1;0,1;0,8;
0,1;0,1;0,1;1,-16;0,1;0,5;0,1;0,6;0,1;0,1;0,1;1,-16;0,1
;0,5;0,1;0,4;0,1;0,5;0,1;1,-18;0,1;0,5;0,1;0,4;0,1;0,5;
0,1;1,-18;0,1;0,5;0,1;0,4;0,1;0,5;0,1;1,-16;0,1;0,1;0,1
;0,6;0,1;0,5;0,1;1,-16;0,1;0,1;0,1;0,6;0,1;0,1;0,1;0,1;
0,1;0,1;0,1;1,-7;0,1;0,1;0,1;0,1;0,1;0,1;0,1;2,-16;0,1;
0,1;0,1;0,1;0,1;1,-5;0,1;0,1;0,1;0,1;0,1;0,4;0,1;0,1;0,
1;0,1;0,1;0,1;0,1;1,-18;0,1;0,10;0,1;0,1;0,1;0,1;0,1;0,
1;0,1;1,-18;0,1;0,16;0,1;1,-18;0,1;0,16;0,1;1,-18;0,1;0
,16;0,1;1,-18;0,1;0,1;0,1;0,1;0,1;0,1;0,1;0,10;0,1;1,-1
8;0,1;0,1;0,1;0,1;0,1;0,1;0,1;0,4;0,1;0,1;0,1;0,1;0,1;1
,-5;0,1;0,1;0,1;0,1;0,1;2,-16;0,1;0,3;0,1;0,1;0,1;0,1;0
,1;1,-9;0,1;0,3;0,1;0,1;0,1;0,1;0,1;0,2;0,1;0,1;0,1;0,1
;0,1;0,1;0,1;0,1;0,1;1,-9;0,1;0,1;0,1;0,1;0,1;0,1;0,1;0
,1;0,1;1,-7;0,1;1,-1;0,1;1,-1;0,1;0,1;0,1;1,-3;0,1;0,1;
0,1;1,-5;0,1;0,5;0,1;1,-7;0,1;0,5;0,1]);J=zeros(w,h,1);
for(i=1:316);J(I(i,1),I(i,2))=.9;end;Z=zeros(w,h);J=max
(J,Z+0.1);P=@(x,y,tx,ty)([1-tx,tx]*J(x:x+1,y:y+1)*[1-ty
;ty]);L=@(x,y)P(floor(x),floor(y),x-floor(x),y-floor(y)
);D=@(x,y)L(min(w-1,max(1,x*w)),min(h-1,max(1,y*h)));M=
65536;v=zeros(M,2);for(i=1:M);p=-1;while(rand()>p);r=ra
nd(2,1);p=D(r(1),r(2));end;v(i,:)=r;end;scatter(v(:,1),
v(:,2),2,linspace(1,10,length(v)),'filled');
```