Random Number Generation

732A90
Computational Statistics

Krzysztof Bartoszek
(krzysztof.bartoszek@liu.se)

7 II 2017

Department of Computer and Information Science
Linköping University

Pseudorandom numbers

- **A computer is a deterministic machine**

- *Congruential generators*
- Functions of time

- Be careful with respect to application

First step: Generating Unif$[0, 1]$

Linear congruential generator

Define a sequence of integers according to

$$x_{k+1} = (a \cdot x_k + c) \mod m, \quad k \geq 0$$

x_0 is **seed**, e.g. based on time

mod m: remainder after division by m

- $x_k \in \{0, \ldots, m-1\}$ and integer
- $x_k/m \sim \text{Unif}[0, 1]$
- $a, c \in [0, m)$ need to be carefully selected

First step: Generating Unif$[0, 1]$

Generated numbers will get into a loop with a certain **period**

$$x_{k+1} = (a \cdot x_k + c) \mod m, \quad k \geq 0$$

$x_0 = a = c = 7$, $m = 10$

1. $x_1 = (7 \cdot 7 + 7) \mod 10 = 56 \mod 10 = 6$
2. $x_1 = (7 \cdot 6 + 7) \mod 10 = 49 \mod 10 = 9$
3. $x_1 = (7 \cdot 9 + 7) \mod 10 = 70 \mod 10 = 0$
4. $x_1 = (7 \cdot 0 + 7) \mod 10 = 7 \mod 10 = 7$
5. $x_1 = (7 \cdot 7 + 7) \mod 10 = 56 \mod 10 = 6$
6. $\ldots$

First step: Generating Unif[0, 1]

```
fthreebits<-function(k,s,L,N){
    X0<-4*s+1;a<-8*k+5;m<-2^L;X<-X0
    for (i in 1:N){
        print(c(X,rev(intToBits(X)[1:5])))
        X<-(a*X)%%m ##c=0
    }
}
```

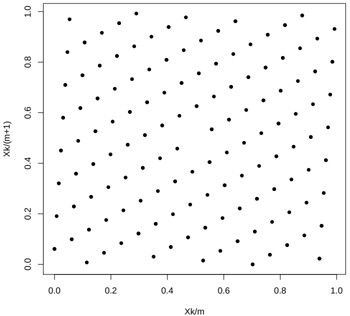
```
> source("CongGen.R");fthreebits(k=2,s=3,L=8,N=10)
[1] 13  0  1  1  0  1
[1] 17  1  0  0  0  1
[1] 101   0   0   1   0   1
[1] 73  0  1  0  0  1
[1] 253   1   1   1   0   1
[1] 193   0   0   0   0   1
[1] 213   1   0   1   0   1
[1] 121   1   1   0   0   1
[1] 237   0   1   1   0   1
[1] 113   1   0   0   0   1
```

Last three bits change between 001 and 101

Discard less significant bits

First step: Generating Unif[0, 1]

```
fCongGenGrid<-function(a,m,c,X0,N){
    X<-X0;vU<-rep(X0,N)
    for (i in 1:N){vU[i]<-X;X<-(a*X+c)%%m}
    vU<-vU/m; plot(vU[1:(N-1)],vU[2:N],pch=19,
        cex=0.8,main="",xlab="Xk/m",ylab="Xk/(m
        +1)")
};fCongGenGrid(a=17,m=131,c=8,X0=4,N=200)
```



First step: Generating Unif$[0, 1]$

- Period has to be smaller than m
- a, c, m (**large**) have to be chosen carefully
  1. c and m have to be relatively prime (no common divisors bar 1)
  2. $a = 1 \mod p$ for every prime divisor p of m
  3. $a = 1 \mod 4$ if 4 divides m
  4. Then full period m reached
- Seed defines the random sequence — same seed, same sequence
  Be careful when re–opening an R workspace
- Other methods (not in this course)

Second step: Generating Unif$[a, b]$

- $U \sim \text{Unif}[0,1]$ can be transformed into $X \sim \text{Unif}[a,b]$ as

$$X = a + U \cdot (b - a)$$

- U can also be transformed into **discrete** uniform distribution on integers $\in \{1, \ldots, n\}$ as ($[\cdot]$, integer part)

$$X = [nU] + 1$$

Questions

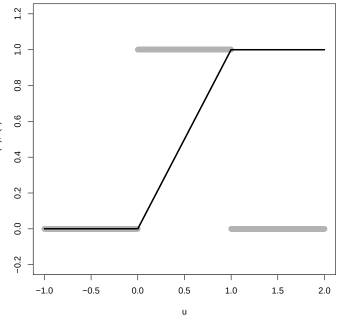
1. Why $+1$?
2. How can U be transformed into Y, where Y is discrete uniform on integers $(50, 55, 60)$?

Second step: Generating nonuniform random numbers

- $U \sim \text{Unif}(0,1)$
- Let F_U be the *cumulative distribution function* (CDF) of U

$$F_U(u) = P(U \leq u) = \begin{cases} 0 & u \leq 0 \\ u & 0 < u \leq 1 \\ 1 & 1 < u \end{cases}$$

- The *probability distribution function* (PDF) of U



$$f_U(u) = \begin{cases} 1 & 0 < u < 1 \\ 0 & u \notin (0,1) \end{cases}$$

Inverse CDF method

Let X be a random variable with CDF $X \sim F_X$ (F_X strictly increasing)

Consider $Y = F_X^{-1}(U)$, where $U \sim \text{Unif}(0,1)$

$$
\begin{aligned}
&F_Y(y) = P(Y \leq y) = P(F_X^{-1}(U) \leq y) \\
&= P(F_X(F_X^{-1}(U)) \leq F_X(y)) \\
&= P(U \leq F_X(y)) = F_U(F_X(y)) = F_X(y)
\end{aligned}
$$

Y **has same probability distribution as** X

Inverse CDF method

If we can generate $U \sim \text{Unif}(0, 1)$, then

we can generate $X \sim F_X$ as

$$X = F_X^{-1}(U)$$

Provided we can calculate F_X^{-1} ...

Inverse CDF method: Example

Let $X \sim \exp(\lambda)$, i.e. with pdf

$$f_X(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

implying (**SHOW THIS**)

$$F_X(x) = \int_{-\infty}^{x} f_X(s)\mathrm{d}s = 1 - e^{-\lambda x}, \quad x \geq 0$$

QUESTIONS:
What is $F_X(x)$ for $x < 0$?
What is $\mathrm{E}[X]$?

Inverse CDF method: Example

Find F_X^{-1}

$$
\begin{aligned}
&y = 1 - e^{-\lambda x} \\
&e^{-\lambda x} = 1 - y \\
&x = -\tfrac{1}{\lambda}\ln(1-y) \\
&F_X^{-1}(y) = -\tfrac{1}{\lambda}\ln(1-y)
\end{aligned}
$$

Hence, if $U \sim \mathrm{U}(0,1)$, then

$$-\frac{1}{\lambda}\ln(1-U) = X \sim \exp(\lambda)$$

Inverse CDF method

1. When F_X^{-1} can be derived: **EASY**

2. When **NOT**: numerical solution

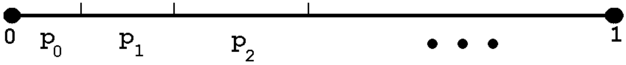
   time–consuming

   numerical errors ?

Situation 2 is common … e.g. $\mathcal{N}(0,1)$

Generating discrete RVs

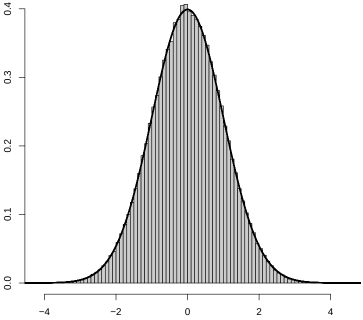
1. Define distribution $P(X = x_i) = p_i$
2. Generate $U \sim \text{Unif}(0, 1)$
3. If $U \leq p_0$, set $X = x_0$
4. Else if $U \leq p_0 + p_1$, set $X = x_1$
5. ...



Generating $\mathcal{N}(0,1)$

Assume

- $\theta \in \text{Unif}(0, 2\pi)$
- $D \in \text{Unif}(0, 1)$



1: Generate θ, D
2: Generate X_1 and X_2 as

$$X_1 = \sqrt{-2 \ln D} \cos \theta$$

$$X_2 = \sqrt{-2 \ln D} \sin \theta$$

X_1 and X_2 are independent and normally distributed

But finding such transformations is not easy

Acceptance/rejection methods

- IDEA: generate $Y \sim f_Y$ similar to some known PDF f_X
- IDEA: f_Y is easy to generate from
- REQUIREMENT: there exists a constant c

$$\forall_x c f_Y(x) \geq f_X(x)$$

- f_Y: majorizing density, proposal density
- f_X: target density
- c: majorizing constant

Acceptance/rejection methods

while X not generated **do**
    Generate $Y \sim f_Y$
    Generate $U \sim \text{Unif}(0, 1)$
    if $U \leq f_X(Y)/(cf_Y(Y))$ **then**
        $X = Y$
        Set X is generated
    end if
end while

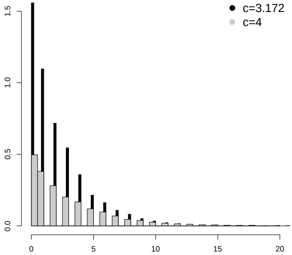
- $X \sim f_X$ **CHECK THIS**
- Larger c: larger rejection rates—c as small as possible
- Can work in higher dimensions—**but** high rejection rate

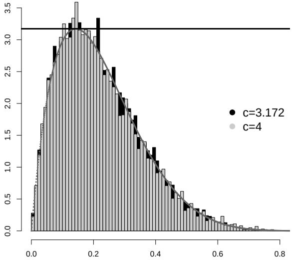
Acceptance/rejection methods: Example

Generate beta(2,7)

```
y<-dbeta(seq(0,2,0.0001),2,7)
c<-max(y);c
[1] 3.172554
```

1: **while** X not generated **do**
2: Generate $Y \sim \text{Unif}(0, 1)$
3: Generate $U \sim \text{Unif}(0, 1)$
4: **if** $U \leq dbeta(Y, 2, 7)/(c \cdot 1)$ **then**
5: $X = Y$
6: Set X is generated
7: **end if**
8: **end while**





QUESTION:

Compare acceptance and rejection regions (of Y) for different c.

Acceptance/rejection methods:

- Acceptance/rejection is difficult to apply
- Difficult to find majorizing density
  - can always take $\sup(f_X) \cdot \mathrm{Unif}(0,1)$
  - but what is the problem?

Generating multivariate normal

Generate $\mathcal{N}(\vec{\mu}, \mathbf{\Sigma}) \in \mathbb{R}^n$

1: Generate n i.i.d. $\mathcal{N}(0,1)$ r.vs. $\vec{X} = (X_1, \ldots, X_n)$
{We know how to do this, see slide 16}
2: Compute Cholesky decomposition (a.k.a. matrix square root) of $\mathbf{\Sigma}$, i.e. find $\mathbf{A}$, lower triangular s.t. $\mathbf{A}\mathbf{A}^T = \mathbf{\Sigma}$,
{in `R`: `chol()` }
3: $\vec{Y} = \mu + \mathbf{A}\vec{X}$

QUESTION:
what is the expectation and variance–covariance of $\vec{Y}$?

Random numbers in R

1. `d`distribution name(): density of distribution
2. `p`distribution name(): CDF of distribution
3. `q`distribution name(): quantiles of distribution
4. `r`distribution name(): simulate from distribution

Summary

- Computers generate pseudo–random numbers
- We draw from pseudo–uniform and transform to desired distribution
- Analytical methods for transforming exist but are distribution specific