Lecture 2: Random number generation

Ciaran Evans

Last time

```
mu_x <- 0
n <- 20
x <- rnorm(n, mean=mu_x, sd=1)
```

- The `rnorm` function provides a random sample from a univariate normal distribution with specified mean and standard deviation
- What other functions exist in R for sampling from probability distributions?

runif, rpois, rexp, rchisq, etc...

Our goal for this unit

Goal: Learn how to simulate random variables

Two main steps:

1. Generating "random" (really, *pseudo*-random) numbers
2. Using random numbers to simulate from a specified distribution

Warm-up question

Suppose that someone asked you to generate a random number (e.g. between 0 and 1). Without resorting to existing software, what would you do? (Your answer does not have to involve a computer!)

Example: using a coin to generate a random number

First, note that we can represent integers in binary (base 2):

E.g. 4-bit integer

$\underline{\quad}$ $\underline{\quad}$ $\underline{\quad}$ $\underline{\quad}$

8s 4s 2s 1s

Decimal

0 = 0 0 0 0

1 = 0 0 0 1

2 = 0 0 1 0

3 = 0 0 1 1

etc.

Notice: k-bit binary integer has values between 0 and $2^k - 1$

$\Rightarrow \frac{x}{2^k}$ is between 0 and 1

Example: using a coin to generate a random number

- Flip fair coin k times
- $H = 1$, $T = 0$

$\Rightarrow$ k-bit binary integer

Not very efficient or practical!

“Random” numbers

- The typical way to generate “random” numbers is with a computer
- By themselves, computers can't generate *truly* random numbers
- Instead, computers use a deterministic algorithm to generate *pseudo-random* numbers

set.seed(...) in R starts the algorithm at a specific place

Example: what does it mean to "behave" like a random number?

Consider the following strings of 0s and 1s

$$0 \;\; 1 \;\; 0 \;\; 1 \;\; 0 \;\; 1 \;\; 0 \;\; 1 \;\; 0 \;\; 1$$

$$0 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 1 \;\; 1 \;\; 0 \;\; 0 \;\; 1$$

Questions:

1. If $P(0) = P(1) = 0.5$, what is the probability of each string?
2. Which string do you think was actually randomly generated?

1) $\left(\frac{1}{2}\right)^{10}$

2) The second one!

Linear congruential generator

$x_n \in \{0, 1, \dots, m-1\}$

$0 \le u_n < 1$

One of the oldest (and historically, widely used) generators is the **linear congruential generator**:

- let x_0 be some initial value (x_0 integer, $0 \le x_0 < m$)

- recurrence relation:

$$x_{n+1} = (a x_n + c) \bmod m$$

multiplier $0 < a < m$; shift $0 \le c < m$; modulus

(x_0, a, c, m all integers)

$x_1, x_2, x_3,$ etc. integers between 0 & $m-1$

$u_n = \frac{x_n}{m}$ is then between 0 & 1

Examples:

· want a long period
· want sequence to bounce around (more "random")

- $a = 1,\ c = 1,\ m = 8$ $\quad x_0 = 1$ $\quad 1, 2, 3, 4, \ldots$

$$x_1 = (1 \cdot 1 + 1) \bmod 8 = 2$$
$$x_2 = (1 \cdot 2 + 1) \bmod 8 = 3$$

- $a = 3,\ c = 0,\ m = 16$ $\quad x_0 = 1$

$$x_1 = 3 \bmod 16 = 3 \qquad x_2 = 9 \bmod 16 = 9$$
$$x_3 = 27 \bmod 16 = 11 \qquad x_4 = 33 \bmod 16 = 1$$

$$\underbrace{1, 3, 9, 11,}_{\text{period}}\ 1, 3, 9, 11, \ldots$$

period (length before repeat) = 4

- $a = 5,\ c = 3,\ m = 16$

$$\underbrace{1, 8, 11, 10, 5, 12, 15, 14, 9, 0, 3, 2, 13, 4, 7, 6,}_{\text{period} = 16}\ 1, 8, 11$$

Choosing the parameters

A sufficient condition for a period of length m (for any initial seed) is:

- c and m are coprime (i.e., greatest common divisor is 1)
- $a - 1$ is divisible by all prime factors of m
- $a - 1$ is divisible by 4 if m is divisible by 4

Why is it helpful for m to be a power of 2?

23 mod 8 = 7

10111 01000 = 00111

121 mod 8 = 1

1111001

= 1

A: it makes the math easy!

Your turn

Practice questions on the course website:

https://sta379-s25.github.io/practice_questions/pq_2.html

- Write code to implement a LCG in R, and experiment with different values of m, a, and c
- Start in class. You are welcome to work with others
- Practice questions are to help you practice. They are not submitted and not graded
- Solutions are posted on the course website