Chapter 13

Introduction to Simulation Using R

A. Rakhshan and H. Pishro-Nik

13.1 Analysis versus Computer Simulation

A computer simulation is a computer program which attempts to represent the real world based on a model. The accuracy of the simulation depends on the precision of the model. Suppose that the probability of heads in a coin toss experiment is unknown. We can perform the experiment of tossing the coin n times repetitively to approximate the probability of heads.

$$P(H) = \frac{\text{Number of times heads observed}}{\text{Number of times the experiment executed}}$$

However, for many practical problems it is not possible to determine the probabilities by executing experiments a large number of times. With today's computers processing capabilities, we only need a high-level language, such as R, which can generate random numbers, to deal with these problems.

In this chapter, we present basic methods of generating random variables and simulate probabilistic systems. The provided algorithms are general and can be implemented in any computer language. However, to have concrete examples, we provide the actual codes in R. If you are unfamiliar with R, you should still be able to understand the algorithms.

13.2 Introduction: What is R?

R is a programming language that helps engineers and scientists find solutions for given statistical problems with fewer lines of codes than traditional programming languages, such as C/C++ or Java, by utilizing built-in statistical functions. There are many built-in statistical functions and add-on packages available in R. It also has high quality customizable graphics capabilities. R is available for Unix/Linux, Windows, and Mac. Besides all these features, R is free!

13.3 Discrete and Continuous Random Number Generators

Most of the programming languages can deliver samples from the uniform distribution to us (In reality, the given values are pseudo-random instead of being completely random.) The rest of this section shows how to convert uniform random variables to any other desired random variable. The R code for generating uniform random variables is:

$$\boxed{U = runif(n, min = 0, max = 1)}$$

which returns a pseudorandom value drawn from the standard uniform distribution on the open interval (0,1).

13.3.1 Generating Discrete Probability Distributions from Uniform Distribution

Let's see a few examples of generating certain simple distributions:

Example 1. (Bernoulli) Simulate tossing a coin with probability of heads p.

Solution: Let U be a Uniform(0,1) random variable. We can write Bernoulli random variable X as:

$$X = \begin{cases} 1 & U < p \\ 0 & U \geq p \end{cases}$$

Thus,

$$\begin{aligned} P(H) &= P(X = 1) \\ &= P(U < p) \\ &= p \end{aligned}$$

Therefore, X has $Bernoulli(p)$ distribution. The R code for $Bernoulli(0.5)$ is:

$$\boxed{\begin{aligned} &p = 0.5; \\ &U = runif(1, min = 0, max = 1); \\ &X = (U < p); \end{aligned}}$$

Since the "runif(1, min = 0, max = 1)" command returns a number between 0 and 1, we divided the interval $[0, 1]$ into two parts, p and $1-p$ in length. Then, the value of X is determined based on where the number generated from uniform distribution fell.

Example 2. (Coin Toss Simulation) Write codes to simulate tossing a fair coin to see how the law of large numbers works.

Solution: You can write:

```
n = 1000;
U = runif(n, min = 0, max = 1);
toss = U < 0.5;
a = numeric(n + 1);
avg = numeric(n);
for(i   in   2 : n + 1)
{
a[i] = a[i − 1] + toss[i − 1];
avg[i − 1] = a[i]/(i − 1);
}
plot(1 : n, avg[1 : n], type = ”l”, lwd = 5, col = ”blue”, ylab = ”ProportionofHeads”,
xlab = ”CoinTossNumber”, cex.main = 1.25, cex.lab = 1.5, cex.axis = 1.75)
```

If you run the above codes to compute the proportion of ones in the variable “toss,” the result will look like Figure 13.1. You can also assume the coin is unbiased with probability of heads equal to 0.6 by replacing the third line of the previous code with:

```
toss = U < 0.6;
```

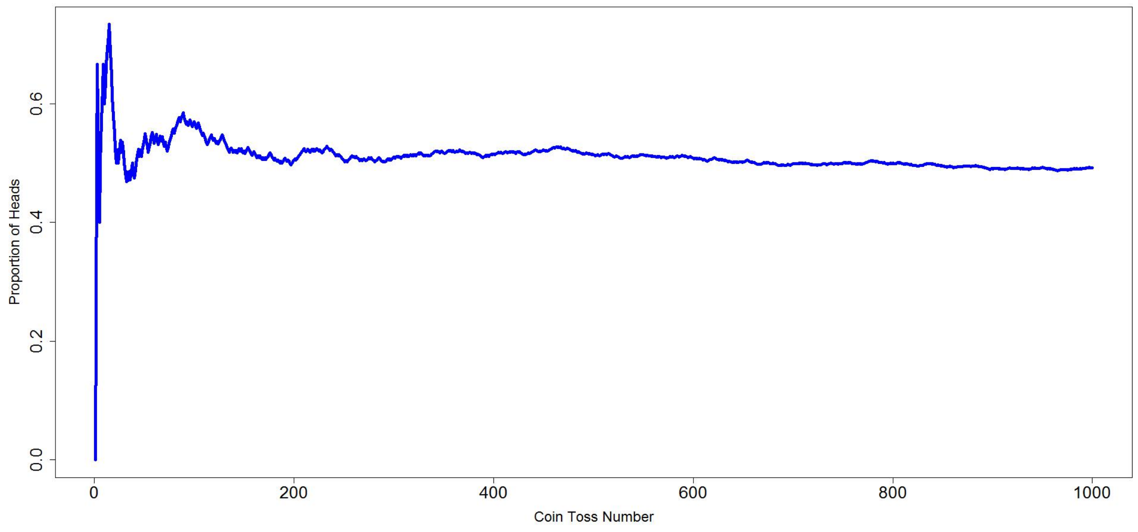

Figure 13.1: R - coin toss simualtion

Example 3. (Binomial) Generate a $Binomial(50, 0.2)$ random variable.

Solution: To solve this problem, we can use the following lemma:

Lemma 1. If $X_1, X_2, ..., X_n$ are independent $Bernoulli(p)$ random variables, then the random variable X defined by $X = X_1 + X_2 + ... + X_n$ has a $Binomial(n, p)$ distribution.

To generate a random variable $X \sim Binomial(n, p)$, we can toss a coin n times and count the number of heads. Counting the number of heads is exactly the same as finding $X_1+X_2+...+X_n$, where each X_i is equal to one if the corresponding coin toss results in heads and zero otherwise.

Since we know how to generate Bernoulli random variables, we can generate a $Binomial(n, p)$ by adding n independent $Bernoulli(p)$ random variables.

```
p = 0.2;
n = 50;
U = runif(n, min = 0, max = 1);
X = sum(U < p);
```

Generating Arbitrary Discrete Distributions

In general, we can generate any discrete random variables similar to the above examples using the following algorithm. Suppose we would like to simulate the discrete random variable X with range $R_X = \{x_1, x_2, ..., x_n\}$ and $P(X = x_j) = p_j$, so $\sum_j p_j = 1$.

To achieve this, first we generate a random number U (i.e., $U \sim Uniform(0, 1)$). Next, we divide the interval $[0, 1]$ into subintervals such that the jth subinterval has length p_j (Figure 13.2). Assume

$$X = \begin{cases} x_0 & \text{if} \quad (U < p_0) \\ x_1 & \text{if} \quad (p_0 \leq U < p_0 + p_1) \\ \vdots & \\ x_j & \text{if} \quad \left(\sum_{k=0}^{j-1} p_k \leq U < \sum_{k=0}^{j} p_k\right) \\ \vdots & \end{cases}$$

In other words

$$X = x_j \quad \text{if} \quad F(x_{j-1}) \leq U < F(x_j),$$

where $F(x)$ is the desired CDF. We have

$$\begin{aligned} P(X = x_j) &= P\left(\sum_{k=0}^{j-1} p_k \leq U < \sum_{k=0}^{j} p_k\right) \\ &= p_j \end{aligned}$$

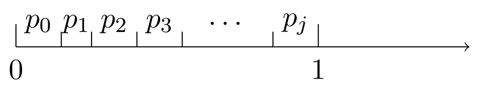

Figure 13.2: Generating discrete random variables

Example 4. Give an algorithm to simulate the value of a random variable X such that

$$
\begin{aligned}
P(X=1) &= 0.35 \\
P(X=2) &= 0.15 \\
P(X=3) &= 0.4 \\
P(X=4) &= 0.1
\end{aligned}
$$

Solution: We divide the interval $[0,1]$ into subintervals as follows:

$$
\begin{aligned}
A_0 &= [0, 0.35) \\
A_1 &= [0.35, 0.5) \\
A_2 &= [0.5, 0.9) \\
A_3 &= [0.9, 1)
\end{aligned}
$$

Subinterval A_i has length p_i. We obtain a uniform number U. If U belongs to A_i, then $X = x_i$.

$$
\begin{aligned}
P(X = x_i) &= P(U \in A_i) \\
&= p_i
\end{aligned}
$$

```
$P = c(0.35, 0.5, 0.9, 1);$
$X = c(1, 2, 3, 4);$
$counter = 1;$
$r = runif(1, min = 0, max = 1);$
$while(r > P[counter])$
  $counter = counter + 1;$
$end$
$X[counter]$
```

13.3.2 Generating Continuous Probability Distributions from the Uniform Distribution- Inverse Transformation Method

At least in principle, there is a way to convert a uniform distribution to any other distribution. Let's see how we can do this. Let $U \sim Uniform(0,1)$ and F be a CDF. Also, assume F is continuous and strictly increasing as a function.

Theorem 1. Let $U \sim Uniform(0,1)$ and F be a CDF which is strictly increasing. Also, consider a random variable X defined as

$$X = F^{-1}(U).$$

Then,

$$X \sim F \quad \text{(The CDF of} \quad X \quad \text{is} \quad F)$$

Proof:

$$\begin{aligned} P(X \leq x) &= P(F^{-1}(U) \leq x) \\ &= P(U \leq F(x)) \quad \text{(increasing function)} \\ &= F(x) \end{aligned}$$

Now, let's see some examples. Note that to generate any continuous random variable X with the continuous cdf F, $F^{-1}(U)$ has to be computed.

Example 5. (Exponential) Generate an Exponential(1) random variable.

Solution: To generate an Exponential random variable with parameter $\lambda = 1$, we proceed as follows

$$\begin{aligned} F(x) &= 1 - e^{-x} \qquad x > 0 \\ U &\sim Uniform(0,1) \\ X &= F^{-1}(U) \\ &= -\ln(1-U) \\ X &\sim F \end{aligned}$$

This formula can be simplified since

$$1 - U \sim \quad Uniform(0,1)$$

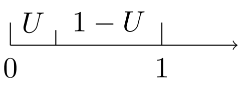

Figure 13.3: Symmetry of Uniform

Hence we can simulate X using

$$X = -\ln(U)$$

$$\boxed{\begin{aligned} &U = runif(1, min = 0, max = 1); \\ &X = -log(U) \end{aligned}}$$

Example 6. (Gamma) Generate a Gamma(20,1) random variable.

Solution: For this example, F^{-1} is even more complicated than the complicated gamma cdf F itself. Instead of inverting the CDF, we generate a gamma random variable as a sum of n independent exponential variables.

Theorem 2. Let $X_1, X_2, \cdots, X_n$ be independent random variables with $X_i \sim \mathit{Exponential}(\lambda)$. Define

$$Y = X_1 + X_2 + \cdots + X_n$$

By the moment generating function method, you can show that Y has a gamma distribution with parameters n and λ, i.e., $Y \sim \mathit{Gamma}(n, \lambda)$.

Having this theorem in mind, we can write:

$$\boxed{\begin{array}{l} n = 20; \\ lambda = 1; \\ X = (-1/lambda) * sum(log(runif(n, min = 0, max = 1))); \end{array}}$$

Example 7. (Poisson) Generate a Poisson random variable. Hint: In this example, use the fact that the number of events in the interval $[0, t]$ has Poisson distribution when the elapsed times between the events are Exponential.

Solution: We want to employ the definition of Poisson processes. Assume N represents the number of events (arrivals) in [0,t]. If the interarrival times are distributed exponentially (with parameter λ) and independently, then the number of arrivals occurred in [0,t], N, has Poisson distribution with parameter λt (Figure 13.4). Therefore, to solve this problem, we can repeat generating $\mathit{Exponential}(\lambda)$ random variables while their sum is not larger than 1 (choosing $t = 1$). More specifically, we generate $\mathit{Exponential}(\lambda)$ random variables

$$T_i = \frac{-1}{\lambda} \ln(U_i)$$

by first generating uniform random variables U_i's. Then we define

$$X = \max\{j : T_1 + \cdots + T_j \leq 1\}$$

The algorithm can be simplified:

$$X = \max\left\{j : \frac{-1}{\lambda} \ln(U_1 \cdots U_j) \leq 1\right\}$$

```
Lambda = 2;
i = 0;
U = runif(1, min = 0, max = 1);
Y = -(1/Lambda) * log(U);
sum = Y;
while(sum < 1)
{U = runif(1, min = 0, max = 1);
Y = -(1/Lambda) * log(U);
sum = sum + Y;
i = i + 1; }
X = i
```

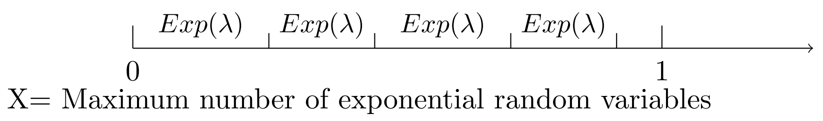

Figure 13.4: Poisson Random Variable

To finish this section, let's see how to convert uniform numbers to normal random variables. Normal distribution is extremely important in science because it is very commonly occuring.

Theorem 3. (Box-Muller transformation) We can generate a pair of independent normal variables (Z_1, Z_2) by transforming a pair of independent $Uniform(0,1)$ random variables (U_1, U_2) [1].

$$\begin{cases} Z_1 = \sqrt{-2\ln U_1}\cos(2\pi U_2) \\ Z_2 = \sqrt{-2\ln U_1}\sin(2\pi U_2) \end{cases}$$

Example 8. (Box-Muller) Generate 5000 pairs of normal random variables and plot both histograms.

Solution: We display the pairs in Matrix form.