14.3: Generating Random Numbers

Running a Monte Carlo simulation requires that we generate random numbers. Generating truly random numbers (i.e. numbers that are completely unpredictable) is only possible through physical processes, such as the decay of atoms or the rolling of dice, which are difficult to obtain and/or too slow to be useful for computer simulation (though they can be obtained from the NIST Randomness Beacon).

In general, instead of truly random numbers we use *pseudo-random* numbers generated using a computer algorithm; these numbers will seem random in the sense that they are difficult to predict, but the series of numbers will actually repeat at some point. For example, the random number generator used in R will repeate after $2^{19937} - 1$ numbers. That's far more than the number of seconds in the history of the universe, and we generally think that this is fine for most purposes in statistical analysis.

In R, there is a function to generate random numbers for each of the major probability distributions, such as:

- `runif()` - uniform distribution (all values between 0 and 1 equally)
- `rnorm()` - normal distribution
- `rbinom()` - binomial distribution (e.g. rolling the dice, coin flips)

Figure 14.1 shows examples of numbers generated using the `runif()` and `rnorm()` functions, generated using the following code:

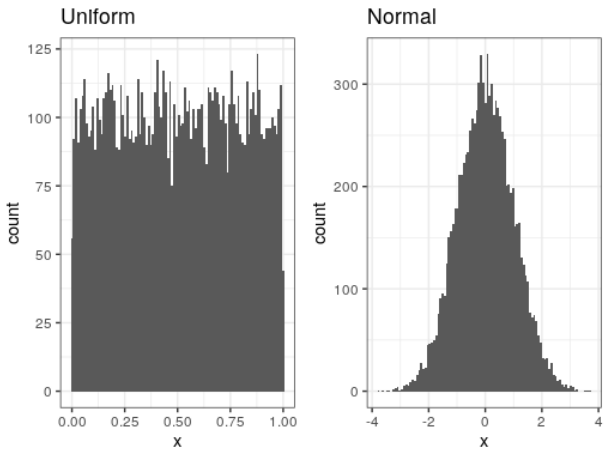

Figure 14.1: Examples of random numbers generated from a uniform (left) or normal (right) distribution.

You can also generate random numbers for any distribution if you have a *quantile* function for the distribution. This is the inverse of the cumulative distribution function; instead of identifying the cumulative probabilities for a set of values, the quantile function identifies the values for a set of cumulative probabilities. Using the quantile function, we can generate random numbers from a uniform distribution, and then map those into the distribution of interest via its quantile function.

By default, R will generate a different set of random numbers every time you run one of the random number generator functions described above. However, it is also possible to generate exactly the same set of random numbers, by setting what is called the *random seed* to a specific value. We will do this in many of the examples in this book, in order to make sure that the examples are reproducible.

If we run the `rnorm()` function twice, it will give us different sets of pseudorandom numbers each time:

```
print(rnorm(n = 5))
```

```
## [1]  1.48  0.18  0.21 -0.15 -1.72
```

```
print(rnorm(n = 5))
```

```
## [1] -0.691 -2.231  0.391  0.029 -0.647
```

However, if we set the random seed to the same value each time using the `set.seed()` function, then it will give us the same series of pseudorandom numbers each time:

```
set.seed(12345)
print(rnorm(n = 5))
```

```
## [1]  0.59  0.71 -0.11 -0.45  0.61
```

```
set.seed(12345)
print(rnorm(n = 5))
```

```
## [1]  0.59  0.71 -0.11 -0.45  0.61
```