Study guide

The purpose of today’s session is to learn to relate basic properties from probability theory to the functioning of Monte Carlo simulations. In class, we use the classic Monty Hall problem to practice framing a problem as a simulation in order to clear up confusing aspects of the problem and make it easier to find an answer.

The two key questions that we focus on in this session are—

1. What are the uses of and consequences of using random numbers to drive our simulations?
2. How can we use “simulation thinking” to think clearly through difficult modeling scenarios involving probabilities?

To address these questions, and others to come in the Monte Carlo simulation unit, you need to be comfortable with basic probability theory. In particular, make sure that your memory of the following topics is fresh—

- what probabilities are,
- what conditional probabilities are,
- how probability density functions and cumulative distribution functions work,
- what the expected value (also known as mean or average) under a probability distribution is,
- what the variance under a probability distribution is.

The code in the main reading for today (Shonkwiler & Mendivil, Explorations in Monte Carlo methods) is written in Matlab but we keep using Python in class. Matlab code is easy to read and in most instances, it will be straightforward to translate the code from Matlab to Python.

The key method for generating numbers we have used so far is to generate uniformly distributed values to select between different random events in a simulation. For example, using

```
if random.uniform(0, 1) < p:
```

to simulate something happening with probability p, or using

```
index = random.randint(0, len(values) - 1)
```

to select a random value (for example node or edge) from a list of values. We start with uniformly distributed random numbers in this session and will build on them to generate ever more complex simulations and random numbers from other, non-uniform distributions.

You will soon see that we can generate random numbers from <u>any</u> probability distribution, no matter how complex, using simulations.