C11 Simulation

David Housman

## Introduction

A **random** or **probabilistic** **experiment** is a repeatable activity with a well-defined set of possible outcomes, although the outcome of any particular **run** of the experiment cannot be accurately predicted.  When something is used as a surrogate for a physical experiment, the word **simulate** is often substituted for run.  The set of all possible **outcomes** is called the **sample space** of the experiment.  In a table, each row is typically associated with a run of the experiment.

One good way to think about a random experiment is pulling an outcome randomly from a hat containing all possible (weighted) outcomes.

A **random variable** is a question that can be asked about the outcome of an experiment.  In more formal language, a random variable is a function defined on the sample space of an experiment.  The question usually has numeric answers and is often stated in a declarative, rather than interrogative, form.  The answers are called **values** of the random variable.

A simple example of a random experiment is to roll a fair four-sided die. Its possible outcomes are 1, 2, 3, or 4. The sample space is . A random variable is the remainder when dividing by two. So, the values would be 0 (for an outcome of 2 or 4) and 1 (for an outcome of 1 or 3). The instructor will carry out one run of this random experiment. The next code chunks simulate this random experiment with base R (returning a vector) and the dplyr package from the tidyverse (returning a tibble). The first code chunk shows the way we will be thinking about pseudo-random number generation: a uniformly random number is chosen from the real number interval and then rounded up to the nearest integer to obtain one of the four possible outcomes. The other two code chunks are more readable that hide the underlying mechanism.

ceiling(runif(n = 1, min = 0, max = 4))

sample(1:4, size = 1)

tibble(side = 1:4) |> slice\_sample()

If the die were not fair but the sides are weighted by their number (1 has weight 0.1, 2 has weight 0.2, 3 has weight 0.3, and 4 has weight 0.4, then the following code chunks shows the modification. The first code chunk generates a random number from the real interval and then returns 1 if , 2 if , 3 if and 4 if .

stepfun(c(0.1, 0.3, 0.6), 1:4)(runif(n = 1))

sample(1:4, prob = c(0.1, 0.2, 0.3, 0.4), size = 1)

tibble(side = 1:4, weight = c(0.1, 0.2, 0.3, 0.4)) |>   
 slice\_sample(weight\_by = weight) |> select(side)

## 

## Ten Coin Flips Wrangling

Consider the random experiment involving flipping a fair coin ten times. Students in various Math 323 and 233 classes have been asked to simulate this experiment four times and then run this experiment four times.

1. Read the results from *C11 Coin Flips Human Simulation.csv* and *C11 Coin Flips Actual.csv*.
2. Define a function that simulates the experiment and obtain one simulation.
3. Run the computer simulation 310 times and store in a tibble.
4. Place the three separate tibbles into a single tibble with an identifying categorical variable.
5. Add the variable *number of heads* to the tenflips tibble.
6. Create a function that returns the length of the longest sequence of the same number in a vector.
7. Add the variable *maximum run length* to the tenflips tibble.

## 

## Ten Coin Flips Description

1. Obtain a visualization that compares the number of heads for the three types of ten coin flips. What is your conclusion?
2. Obtain a visualization that compares the maximum run length for the three types of ten coin flips. What is your conclusion?