

Coin Flipping

BIOF2014

```
set.seed(1234);
```

Objective

Let's create a simulation model for sampling sequence of coin flips and tallying the number of heads. Through this exercise, we will illustrate how a random variable can be implemented explicitly in R.

We will

1. Generate the sample space
2. Define the random variable
3. Calculate the induced probability function on the random variable
4. Sample from the sample space and output a realization of the random variable

Model parameters

We are sampling a sequence of coin flips. So, the space space consists of all possible coin flip outcomes for a sequence of a defined length.

Let's first define the sequence length, which is the number of coin flips

```
n.trials <- 5;
```

Each coin flip has two possible outcomes: tail or head.

```
outcomes <- c("T", "H");
```

Fair coins are boring! So, let's make the coin flipping more likely to produce a head.

```
probs.outcome <- c(0.3, 0.7);
```

Space space

With the parameters defined, we are now ready to generate all possible elements in the sample space. So, let's define a helper function.

```
# Enumerate the sample space of a sequence of n trials,
# given the outcomes of individual trials,
# returning the sample space
enumerate <- function(outcomes, n) {
  if (n == 1) {
    as.list(outcomes)
  } else {
    branches <- enumerate(outcomes, n - 1);
    unlist(
      lapply(branches,
        function(b) {
          lapply(outcomes, function(o) c(b, o))
        }
      ),
      recursive = FALSE
    )
  }
}
```

Now, we're ready to enumerate the sample space S .

```
S <- enumerate(outcomes, n.trials);
```

We must then determine the probability function P for the sample space. We will implement P as a vector called `probs.S`.

```
probs.S <- unlist(lapply(
  enumerate(probs.outcome, n.trials),
  function(probs) {
    prod(probs)
  }
));
```

Random variable

Recall that a random variable, mathematically, is a mapping from the sample space to real numbers. So, we can explicitly define the random variable X as a function that counts the number of heads in a sample s .

```
X <- function(s) {  
  sum(s == "H")  
}
```

Using our random variable function, let's map each element of the sample space onto the domain of X .

```
values.x <- unlist(lapply(S, X));
```

Now, we can explicitly determine the domain of X .

```
domain.x <- unique(values.x);
```

Induced probability function

In the random variable space \mathcal{X} , we also need to have a probability function, P_X , which is induced by the random variable X , according to

$$P_X(X = x_i) = P(\{s_j \in \mathcal{S} : X(s_j) = x_i\}),$$

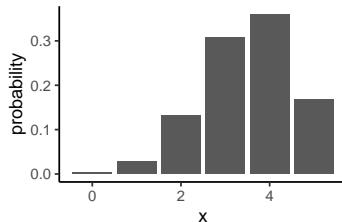
where P is the probability function of the sample space that we have already defined above as `probs.S`.

Therefore, P_X can be implemented as

```
probs.x <- unlist(lapply(domain.x,  
  function(x) {  
    idx <- values.x == x;  
    sum(probs.S[idx])  
  })  
);
```

Let's visualize our induced probability function P_X . (Since our random variable is discrete, $P_X(X = x)$ is equivalent to our probability mass function $f_X(x)$.)

```
library(ggplot2)
ggplot(data.frame(x=domain.x, y=probs.x), aes(x, y)) + theme_classic() +
  geom_col() + ylab("probability")
```



Sampling

Now, we're finally ready to sample a realization $x \in \mathcal{X}$. We first draw a sample from the sample space and map it to \mathcal{X} to obtain the realization.

```
s <- sample(outcomes, n.trials, replace=TRUE, prob=probs.outcome);
X(s)
```

```
[1] 4
```

We determine the probability of sample s using P (`probs.S`) by

```
s.idx <- which(unlist(lapply(S, function(s.i) all(s.i == s))));
probs.S[s.idx]
```

```
[1] 0.07203
```

Remarks

1. Here, we represented some mathematical functions (e.g. P and P_X) as vectors (`probs.S` and `probs.X`). Later in the course, we will derive the mathematical expression for P_X , and we will be able to implement it as a real function.
2. Our sampling implementation is a very explicit, direct implementation of the mathematical framework for a random variable. In practice, we can sample X directly based on the pmf or pdf $f_X(x)$. To emphasize, explicit direct implementation of mathematical models can be computationally inefficient, so this is why we need to do **mathematical derivations** to simplify the model before we implement it!

Questions

1. When we are drawing coin flips, why do we sample with replacement?
2. How does changing the probability function on the sample space change the induced probability function on X ?
3. As the number of trials increases, can we still calculate `prob.x` accurately? If not, modify the code to make the calculation of `prob.x` more accurate.