Introduction to Probability

Set Theory

This section goes over some definitions and theory before we go any further.

Set - An unordered collection of distinct elements.

A set is made of of elements, and each of those elements is a **member** of the set. We use the $\in$ symbol to denote membership. The following means *element x is a member of set A*.

$$x \in A$$

Curly braces are usually used to denote sets.

$$\{1, 2, 3\}$$

You could think of this set as variable x, given that x is either 1,2, or 3. We write this mathematically as the following (the pipe | means "given")

$$\{x \mid x = 1 \text{ or } x = 2 \text{ or } x = 3\}$$

A more general form of this for any set can be written as: $\{x|conditions\}$ Where x are the elements of the set, defined by whatever conditions the situation calls for.

Some properties of sets

- **A set is uniquely defined by its members** Two sets are equal only if all of their elements are the same. However sets are also unordered - so $\{1, 2, 3\}$ is equal to $\{3, 1, 2\}$
- The **Empty Set** has no contents. It is represented by $\emptyset$ or $\{\}$
- Set A can be a **Subset** of set B if all the elements in set A are also members of set B. Mathematically, this is written as $A \subset B$. Note that the empty set is the subset of any other set and $A \subset A$ for any set A.
- We say that sets $A_1, A_2, A_3, \ldots$ are **mutually exclusive** or **disjoint** if $A_i \cap A_j = \emptyset$ for any distinct pair $A_i \neq A_j$. For instance, in the coin-toss experiment the set A = {Heads} and B = {Tails} would be mutually exclusive.

Sample Spaces

For an experiment E, the set of all possible outcomes of E is called the **sample space** and is denoted by the letter S. For a coin-toss experiment, S would be the results "Head" or "Tail", which we may represent by $S = \{H, T\}$.

Consider the experiment of dropping an empty Styrofoam cup onto the floor from a height of four feet. The cup hits the ground and eventually comes to rest. It could land upside down, right side up, or it could land on its side. Here, the sample space S is $\{"down", "up", "side"\}$.

With more outcomes, the sample space becomes harder to write out. If we were to flip *two* coins, the result could be two heads, two tails, one head then one tail, or one tail then one head. The sample space S is then $S = \{HH, HT, TH, TT\}$

What about three coins?

Instead of spending the time writing out sample spaces, we can have R do it for us. One of the easiest ways to do this would be to simulate the coin flips a number of times, and then use the `table()` function to show what outcomes were produced. If we run the experiment enough times (10,000 should be more than enough), we can be reasonably sure that all outcomes will appear. Unlike the previous section, the coin flip function here will produce the letters H and T for heads and tails.

```r
flip3 <- function() {
  # A function to flip 3 coins - we'll sample three separately and then use the
  #   paste() function to paste the three together.
      temp <- sample(c("H","T"), size=3, replace=T)
      return(paste(temp, collapse="")) #collapse turns three results into one "word"
}

# Initializing the vector to store all the experiment outcomes (sets of 3 flips)
outcomes <- vector()

# Do the experiment 10,000 times
for (i in 1:10000) {
    outcomes[i] <- flip3()
}

# The unique command prints only the unique outcomes in a vector.
unique(outcomes)
```

```
## [1] "HHT" "HTH" "TTT" "HTT" "HHH" "TTH" "THH" "THT"
```

Now use R to list the Sample Space for the experiment of dropping three styrofoam cups where each one can land down, up, or sideways.

```r
## ANSWER:
drop3 <- function() {
  temp <- sample(c("U","D","S"), size=3, replace=T)
  return(paste(temp, collapse=""))
}

outcomes <- vector()

for (i in 1:10000) {
  outcomes[i] <- drop3()
}

unique(outcomes)
```

```
##  [1] "DUD" "SUU" "UDU" "DUU" "USD" "DDD" "UDD" "UUS" "DSD" "SSD" "UUU"
## [12] "DSS" "USS" "SUS" "DSU" "SDS" "USU" "SSU" "DDS" "SDD" "DDU" "SUD"
## [23] "SDU" "UDS" "DUS" "SSS" "UUD"
```

```r
length(unique(outcomes))
```

```
## [1] 27
```

Sampling from an *urn* is a canonical type of experiment in probability class. The urn contains a bunch of distinguishable objects (i.e. balls) inside. We shake up the urn, reach inside, grab a ball, and take a look. In this simple version, the sample space would just depend on how many types of balls are in the urn. If some are red and some are blue, the sample space for picking one ball would be $S = \{R, B\}$

Suppose you have an urn containing three balls numbered 1 - 3. If we draw two balls from an urn, what are all of the possible outcomes of the experiment? It depends on how we sample. We could select a ball, take a look, put it back, and sample again (sampling **with replacement**). Another way would be to select a ball, take a look – but **not** put it back, and sample again (sampling **without replacement**.)

Use R to build the sample space for sampling 2 balls *with replacement* from an urn containing three balls numbered 1-3.

```
## ANSWER:
pick2 <- function() {
      temp <- sample(c(1,2,3), size=2, replace=T)
      return(paste(temp, collapse=" "))
}

outcomes <- vector()

for (i in 1:10000) {
    outcomes[i] <- pick2()
}

unique(outcomes)
```

```
## [1] "2 3" "3 1" "3 3" "2 1" "1 2" "3 2" "2 2" "1 1" "1 3"
```

How about the same thing, but sampling *without replacement*?

```
## ANSWER:
pick2 <- function() {
      temp <- sample(c(1,2,3), size=2, replace=F)
      return(paste(temp, collapse=" "))
}

outcomes <- vector()

for (i in 1:10000) {
    outcomes[i] <- pick2()
}

unique(outcomes)
```

```
## [1] "2 1" "2 3" "3 2" "1 3" "1 2" "3 1"
```

What is the difference between the two?

> ANSWER: Sampling without replacement didnt have any repeats (e.g. 1 1 or 2 2).

Suppose we do not actually keep track of which ball came first. All we observe are the two balls, and we have no idea about the order in which they were selected. We call this **unordered sampling** (the opposite, and what we were doing before is *ordered sampled*) because the order of the selections does not matter with respect to what we observe.

Challenge question - write R code to simulate picking 2 balls from on urn of 3 numbered balls *with* replacement, if it were *unordered sampling*

```
## ANSWER:
pick2 <- function() {
      temp <- sort(sample(c(1,2,3), size=2, replace=T))
      return(paste(temp, collapse=" "))
}

outcomes <- vector()
```

```
for (i in 1:10000) {
    outcomes[i] <- pick2()
}

unique(outcomes)
```

```
## [1] "2 3" "1 3" "3 3" "1 1" "1 2" "2 2"
```

Events

An event is a specific collection of outcomes, or in other words, a **subset of the sample space**. After the performance of a random experiment, we say that the event A *occurred* if the experiment's outcome *belongs to* A.

Let's create the sample space for the experiment consisting of flipping three coins:

```
## Generate the sample space for flipping a coin 3 times.
## Used in the code below
flip3 <- function() {
      temp <- sample(c("H","T"), size=3, replace=T)
      return(paste(temp, collapse=""))
}
outcomes <- vector()
for (i in 1:10000) {
  outcomes[i] <- flip3()
}
sample.space <- unique(outcomes)
```

Brackets [ ]

R has different ways to find subsets. Square brackets select certain elements of a vector:

```
print(sample.space)
```

```
## [1] "TTH" "HTH" "THT" "HHH" "HHT" "TTT" "THH" "HTT"
```

```
sample.space[2]
```

```
## [1] "HTH"
```

```
sample.space[1:3]
```

```
## [1] "TTH" "HTH" "THT"
```

```
sample.space[c(2,5)]
```

```
## [1] "HTH" "HHT"
```

The `%in%` function

The function %in% helps to learn whether each value of one vector lies somewhere inside another vector. Consider set x, the numbers from 1 to 10, and set y, the numbers from 8 to 12.

```
x <- 1:10
y <- 8:12
y %in% x
```

```
## [1]  TRUE  TRUE  TRUE FALSE FALSE
```

```
y[which(y %in% x)]
```

```
## [1]  8  9 10
```

Notice that the returned value of the first line is a vector of length 5 which tests whether each element of y is in x, in turn. The returned value of the second line are the elements of y that are also elements of x.

Union, Intersection and Difference

Given subsets A and B, it is often useful to manipulate them in an algebraic fashion. We have three set operations at our disposal to accomplish this: **union**, **intersection**, and **difference**. Additionally we can take the **complement** of one of the sets. Below is a table that summarizes the pertinent information about these operations.

Name	Notation	Definition	R Function
Union	$A \cup B$	In either A or B or both	`union(A,B)`
Intersection	$A \cap B$	In both A and B	`intersect(A,B)`
Difference	$A \setminus B$	In A but not in B	`setdiff(A,B)`

Find the union, intersect, and difference of `x` and `y` using R:

```
x <- 1:10
y <- 8:12
```

```
## Answer:
union(x,y)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

```
intersect(x,y)
```

```
## [1]  8  9 10
```

```
setdiff(x,y)
```

```
## [1] 1 2 3 4 5 6 7
```

Switch the order of the variables. For which operations do the results stay the same? For which are they different? Why?

```
## ANSWER:
union(y,x)
```

```
##  [1]  8  9 10 11 12  1  2  3  4  5  6  7
```

```
intersect(y,x)
```

```
## [1]  8  9 10
```

```
setdiff(y,x)
```

```
## [1] 11 12
```

```
## Answer:
# The order doesnt matter in the first two because the operations are symmetrical.
# The difference is not symmetrical because we're taking the area of the second from the area of the fi
```

Complement

If we know the Sample Space, we can also use `setdiff` to take the **complement** of a subset. A complement is the entire part of the sample space *not* in the subset. Set up a new sample space containing the numbers 1 to 15 and find the complement of x in this sample space.

```r
## ANSWER:
s <- 1:15 # A sample space of number 1-15

setdiff(s,x) # The complement of x in sample space s
```

```
## [1] 11 12 13 14 15
```