COMP 50 Lab: Probability

December 4–5, 2013

This lab delves a little bit deeper into probability. The goal of the lab is for you to write code that can tell you what pair of dice were chosen at random from a bag.

At the start of the lab, you and your partner should each close your eyes and *choose two dice from the bag* of dice. *Don't let anyone see which dice you've chosen.*

Data Definitions

A `(weighted X)` is a structure

- `(make-weighted weight value)` where `weight` is a number and `value` is an X

A `(dist X)`, aka "probability distribution of `X`" is a `(listof (weighted X))` where the weights in the list add up to 1

Functions to start with

The function `d` produces a distribution of numbers for a die. It works *only* for d4, d6, d8, d12, and d20. The d10 is numbered from 0 to 9 and requires a special case or some other adjustment.[1]

```
;; d : number -> (dist number)
;; compute the probability distribution of rolls for a die with N sides
(define (d n)
  (build-list n (lambda (i) (make-weighted (/ 1 n) (add1 i)))))

(check-expect (d 4) (list (make-weighted 1/4 1)
                          (make-weighted 1/4 2)
                          (make-weighted 1/4 3)
                          (make-weighted 1/4 4)))
```

The `expectation` function computes the expected value of a function

```
;; expectation : (dist X) (X -> number) -> number
;; compute the average result from applying `f` to values drawn from `dist`
(define (expectation dist f)
  (foldl
   (lambda (wv sum) (+ sum (* (weighted-weight wv) (f (weighted-value wv)))))
   0
```

[1] The d10 is in fact nearly impossible to tell apart from the d8.

```
   dist))

(check-expect (expectation (d 4) (lambda (n) (remainder n 2)))
              1/2)
```

The probability of an event is the expectation of a function that returns 1 when the event is true and 0 otherwise. Such a function is called a *characteristic* function.

```
;; probability : (dist X) (X -> boolean) -> number
;; compute the probability of `(p? x)` when `x` drawn randomly from `dist`
(define (probability dist p?)
  (expectation dist (lambda (x) (characteristic (p? x)))))

(check-expect (probability (d 6) (lambda (n) (< n 5))) 2/3)

;; characteristic : boolean -> number
;; convert the given Boolean to 0 or 1
(define (characteristic b)
  (cond [b 1]
        [else 0]))

(check-expect (characteristic (< 2 3)) 1)
(check-expect (characteristic (< 3 3)) 0)
```

Problems to solve

The problems are divided into two parts. In the first part, you compute what dice you think your partner has rolled. In the second part, you compute how confident you are.

The most likely dice

1. *Design a function* which takes as inputs two distributions of numbers and a predicate on numbers. Its output is the probability that the sum of the two numbers satisfies the predicate.
2. *Make a list* of all possible pairs of dice. Individual dice are available with 4, 6, 8, 10, 12, and 20 sides. You can make a list of all pairs by hand, or you can define a function to do it.

   ```
   (check-expect (all-pairs '(1 2 3)) '((1 1) (1 2) (1 3) (2 2) (2 3) (3 3)))
   ```

3. *Define a function* that is given a pair of dice and computes the log probability of rolling 7 or less with that pair of dice. Use decibans.

   ```
   ;; log-decibans : number -> number
   ;; return the 10 times the base-10 logarithm of the given number
   (define (log-decibans x)
     (/ (log x) log10/10))

   (check-within (log-decibans 1)      0 EPSILON)
   (check-within (log-decibans 10)    10 EPSILON)
   (check-within (log-decibans 100)   20 EPSILON)
   (check-within (log-decibans 1/10) -10 EPSILON)
   ```

4. *Define a function* that is given a pair of dice and computes the log probability of rolling 8 or more with that pair of dice. Use decibans.

5. Have your partner roll his or her *hidden* dice for 25 to 50 trials, reporting how many trials are 7 or less versus how many are 8 or more.

   Each trial adds a little bit of evidence; because trials are independent, probabilities multiply and log probabilities add.

6. *Define a function* that takes as arguments (a) a pair of dice, (b) the number of rolls of 7 or less, and (c) the number of rolls of 8 or more, and returns the log probability of seeing exactly the given rolls with the given dice.

7. Use your results to list all possible pairs of dice from most likely to least likely. What dice do you think were rolled? How confident are you?

Your confidence in the dice

We can compute confidence using Bayes's theorem.

8. *The easy case*. Let's suppose there are only two possibilities for the dice that your partner holds, which I'll call *pair1* and *pair2*. Let's further suppose that the choices are equally likely, then the odds that your partner has *pair1* are

$$\frac{P(observation|pair1)}{P(observation|pair2)}$$

   If you take the two most likely choices from your list, you can estimate the log odds in favor of *pair1*—but it will be an overestimate.

The true odds of your partner having a given pair of dice $pair_i$ account for the possibility that your partner might have *any* other pair. They are

$$\frac{P(pair_i)}{P(\neg pair_i)} \cdot \frac{P(observation|pair_i)}{P(observation|\neg pair_i)}$$

Where $P(pair_i)$ is the "prior" probability of having chosen $pair_i$ from the bag.

Probabilities based on *not* having $pair_i$ include

$$P(\neg pair_i) = 1 - P(pair_i)$$

$$P(observation|\neg pair_i) = \sum_{j \neq i} P(observation|pair_j) P(pair_j|\neg pair_i)$$

Note that the equation uses the sum of the *probabilities*, **not** the sum of the logarithms! To sum the probabilities you will need to exponentiate the logarithms, then sum—and if the probabilities are too small, you may need to scale them first. If this happens, ask for help from Norman.

I believe that the probability $P(pair_j|\neg pair_i)$ of having pair j given that your partner does not have pair i is

$$P(pair_j|\neg pair_i) = \frac{P(pair_j)}{1 - P(pair_i)}$$

The prior probability of choosing a given pair can be estimated by knowing the contents of the bag:

Number of sides	Number of those dice in the bag
d4	12
d6	12
d8	12
d10	16
d12	17
d20	17

9. Using the table, *define a function* that computes the prior probability of choosing a given pair of dice.
10. Using the prior probabilities, compute the log odds in favor of each possible pair of dice (as compared with all the others by summing).

Submitting the lab

Five minutes before the end of the lab, put the following text at the beginning of a DrRacket file. You may use an empty file or the source code you have been working on:

```
#|
What I did during this lab:
   (you fill in this part)

What I learned during this lab:
   (you fill in this part)

|#
```

Finally, *submit this file through the handin server* as `lab-final`. **Submit it as lab, not as homework.** You will need to submit it using *two* usernames connected by a + sign, as in

```
Jane.Doe+Richard.Roe
```

You submit using Jane Doe's password.