Monte Carlo Simulation - Introduction

Computer simulation essentially does in actual code what one does conceptually in our "notebook" view of probability.

Example: (Rolling Dice) If we roll three dice, what is the probability that their total is 8?

Simulating Rolling Dice

```
# roll d dice; find P(total = k)
probtotk <- function(d, k, nreps) {
   count <- 0
   # do the experiment nreps times --
   # like doing nreps notebook lines
   for (rep in 1:nreps) {
      sum <- 0
      # roll d dice and find their sum
      for (j in 1:d) sum <- sum + roll()
      if (sum == k) count <- count + 1
   }
   return(count/nreps)
}
# simulate roll of one die; the possible return values are
# 1,2,3,4,5,6, all equally likely
roll <- function() return(sample(1:6,1))
# example
probtotk(3,8,1000)
```

First Improvement

Since applications of R often use large amounts of computer time, good R programmers are always looking for ways to speed things up.

Improved Simulation Code

```
# roll d dice; find P(total = k)

probtotk <- function(d, k, nreps) {
    count <- 0
    # do the experiment nreps times
    for (rep in 1:nreps) {
        total <- sum(sample(1:6, d, replace=TRUE))
        if (total == k) count <- count + 1
    }
    return(count/nreps)
}
```

Second Improvement

Further improvements are possible.

Vectorized Simulation Code

```
# roll d dice; find P(total = k)

# simulate roll of nd dice; the possible return values are
# 1,2,3,4,5,6, all equally likely
roll <- function(nd) return(sample(1:6, nd, replace=TRUE))

probtotk <- function(d, k, nreps) {
   sums <- vector(length=nreps)
   # do the experiment nreps times
   for (rep in 1:nreps) sums[rep] <- sum(roll(d))
   return(mean(sums == k))
}
```

Third Improvement

Even better. **Improved Simulation Code with Replicate**

```
roll <- function(nd) return(sample(1:6, nd, replace=TRUE))

probtotk <- function(d, k, nreps) {
   # do the experiment nreps times
   sums <- replicate(nreps, sum(roll(d)))
   return(mean(sums == k))
}
```

Example: Dice Problem

- Suppose three fair dice are rolled.
- We wish to find the approximate probability that the first die shows fewer than 3 dots, given that the total number of dots for the 3 dice is more than 8, using simulation.
- Simulation is writing code that implements our "notebook" view of probability.

Simulating the Dice Problem

```
 1  dicesim <- function(nreps) {
 2      count1 <- 0
 3      count2 <- 0
 4      for (i in 1:nreps) {
 5          d <- sample(1:6,3,replace=T)
 6          if (sum(d) > 8) {
 7              count1 <- count1 + 1
 8              if (d[1] < 3) count2 <- count2 + 1
 9          }
10      }
11      return(count2 / count1)
12  }
```

Example: Bus Ridership - Introduction

Consider the following analysis of bus ridership. In this oversimplified model, we'll explore various probabilities related to # of bus passengers and stops.

Model Assumptions

- At each stop, each passenger alights from the bus, independently, with probability 0.2 each.
- Either 0, 1, or 2 new passengers get on the bus with probabilities 0.5, 0.4, and 0.1, respectively.
- Passengers at successive stops act independently.
- Assume the bus is so large that it never becomes full, so new passengers can always get on.
- Suppose the bus is empty when it arrives at its first stop.

Bus Ridership - Notation

Let's define some notation for the analysis:

- N_i : Number of passengers on the bus as it leaves the i^{th} stop, $i = 1, 2, 3, \ldots$
- B_i : Number of new passengers who board the bus at the i^{th} stop.
- A_i : Number of passengers who alights from the bus at the i^{th} stop.

(i) Probability that No Passengers Boarding in First Three Stops?

$$P(B_1 = 0 \text{ and } B_2 = 0 \text{ and } B_3 = 0) = 0.5^3$$

Bus Ridership - cont'd

(ii) Probability that Bus Leaves Second Stop Empty, i.e., $P(N_2 = 0)$?

To calculate this, we employ a very common approach:

- Ask, "How can this event happen?"
- Break big events into smaller events.
- Apply the mailing tubes.

$$
\begin{aligned}
P(N_2 = 0) &= P(B_1 = 0 \text{ and } B_2 = 0 \\
&\quad \text{or } B_1 = 1 \text{ and } B_2 = 0 \text{ and } A_2 = 1 \\
&\quad \text{or } B_1 = 2 \text{ and } B_2 = 0 \text{ and } A_2 = 2) \\
&= P(B_1 = 0 \text{ and } B_2 = 0) + P(B_1 = 1 \text{ and } B_2 = 0 \text{ and } A_2 = 1) \\
&\quad + P(B_1 = 2 \text{ and } B_2 = 0 \text{ and } A_2 = 2) \\
&= 0.5^2 + (0.4)(0.5)(0.2) + (0.1)(0.5)(0.2^2) \\
&= 0.292 \qquad (1)
\end{aligned}
$$

Bus Ridership - cont'd

(iii) Suppose we are told that the bus arrives empty at the third stop. What is the probability that exactly two people boarded the bus at the first stop?.

Note first that we want to find: $P(B_1 = 2 | N_2 = 0)$.

$$
\begin{aligned}
P(B_1 = 2 \mid N_2 = 0) &= \frac{P(B_1 = 2 \text{ and } N_2 = 0)}{P(N_2 = 0)} \\
&= P(B_1 = 2)\ P(N_2 = 0 \mid B_1 = 2)\ /\ 0.292 \\
&= P(B_1 = 2)\ P(B_2 = 0 \text{ and } A_2 = 2)\ /\ 0.292 \\
&= (0.1)(0.5)(0.2^2)/0.292
\end{aligned}
$$

(the 0.292 had been previously calculated in Equation (1)).

Bus Ridership - cont'd

(v) Probability that fewer people board at the second stop than at the first?

$$P(B_2 < B_1) = 0.4 \cdot 0.5 + 0.1 \cdot (0.5 + 0.4)$$

(vi) Probability that Only One Passenger on Bus When the Bus Left First Stop?

$$P(N_1 = 1 | N_2 = 0 \text{ and } N_1 > 0) = \frac{0.4 \cdot 0.2 \cdot 0.5}{0.4 \cdot 0.2 \cdot 0.5 + 0.1 \cdot 0.2^2 \cdot 0.5}$$

(vii) An observer at the second stop notices that no one alights there. Find the probability that there was already one passenger on the bus when it came to the second stop?

$$P(N_1 = 1 | A_2 = 0) = \frac{0.4 \cdot 0.8}{0.5 \cdot 1 + 0.4 \cdot 0.8 + 0.1 \cdot 0.8^2}$$

Use of runif() for Simulating Events

- To simulate whether a simple event occurs or not, we typically use R function `runif()`.
- This function generates random numbers from the interval (0,1), with all the points inside being equally likely.

Ex: Simulating a Coin Toss

```
if (runif(1) < 0.5) heads <- TRUE else
  heads <- FALSE
```

Example: Simulating ALOHA Network

Following is a computation via simulation of the approximate values of $P(X_1 = 2)$, $P(X_2 = 2)$ and $P(X_2 = 2|X_1 = 1)$.

```
1  # finds P(X1 = 2), P(X2 = 2) and P(X2 = 2|X1 = 1) in ALOHA example
2  sim <- function(p,q,nreps) {
3     countx2eq2 <- 0; countx1eq1 <- 0; countx1eq2 <- 0; countx2eq2givx1eq1 <- 0
4     # simulate nreps repetitions of the experiment
5     for (i in 1:nreps) {
6        numsend <- 0
7        for (j in 1:2)
8           if (runif(1) < p) numsend <- numsend + 1
9        if (numsend == 1)  X1 <- 1
10       else X1 <- 2
11       if (X1 == 2) countx1eq2 <- countx1eq2 + 1
12       numactive <- X1
13       if (X1 == 1 && runif(1) < q) numactive <- numactive + 1
14       if (numactive == 1)
15          if (runif(1) < p) X2 <- 0
16          else X2 <- 1
17       else {
18          numsend <- 0
19          for (i in 1:2)
20             if (runif(1) < p) numsend <- numsend + 1
21          if (numsend == 1) X2 <- 1
22          else X2 <- 2
23       }
24       if (X2 == 2) countx2eq2 <- countx2eq2 + 1
25       if (X1 == 1) {
26          countx1eq1 <- countx1eq1 + 1
27          if (X2 == 2) countx2eq2givx1eq1 <- countx2eq2givx1eq1 + 1
28       }
29    }
30    # print results
31    cat("P(X1 = 2):",countx1eq2/nreps,"\n")
32    cat("P(X2 = 2):",countx2eq2/nreps,"\n")
33    cat("P(X2 = 2 | X1 = 1):",countx2eq2givx1eq1/countx1eq1,"\n")
```

Example: Bus Ridership (cont'd.)

- Consider Example 1.1 (in Matloff).
- Let's find the probability that after visiting the tenth stop, the bus is empty. This is too complicated to solve analytically but can easily be simulated.

Simulating Bus Ridership

```
nreps <- 10000
nstops <- 10
count <- 0
for (i in 1:nreps) {
   passengers <- 0
   for (j in 1:nstops) {
      if (passengers > 0)
         for (k in 1:passengers)
            if (runif(1) < 0.2)
               passengers <- passengers - 1
      newpass <- sample(0:2,1,prob=c(0.5,0.4,0.1))
      passengers <- passengers + newpass
   }
   if (passengers == 0) count <- count + 1
}
print(count/nreps)
```

Example: Board Game

- Recall the board game example in Section 1.8 in Matloff.
- Below is simulation code to find the probability in (2) below.

$$
\begin{aligned}
P(B > 0 \mid R + B = 4) &= \frac{P(R + B = 4, B > 0)}{P(R + B = 4)} \qquad (2) \\
&= \frac{P(R + B = 4, B > 0)}{P(R + B = 4, B > 0 \text{ or } R + B = 4, B = 0)} \\
&= \frac{P(R + B = 4, B > 0)}{P(R + B = 4, B > 0) + P(R + B = 4, B = 0)} \\
&= \frac{P(R = 3, B = 1)}{P(R = 3, B = 1) + P(R = 4)} \\
&= \frac{\frac{1}{6} \cdot \frac{1}{6}}{\frac{1}{6} \cdot \frac{1}{6} + \frac{1}{6}} \\
&= \frac{1}{7}
\end{aligned}
$$

Simulating the Board Game

```
 1  boardsim <- function(nreps) {
 2      count4 <- 0
 3      countbonusgiven4 <- 0
 4      for (i in 1:nreps) {
 5          position <- sample(1:6,1)
 6          if (position == 3) {
 7              bonus <- TRUE
 8              position <- (position + sample(1:6,1)) %% 8
 9          } else bonus <- FALSE
10          if (position == 4) {
11              count4 <- count4 + 1
12              if (bonus)
13              {countbonusgiven4 <- countbonusgiven4 + 1}
14          }
15      }
16      return(countbonusgiven4/count4)
17  }
```

Example: Broken Rod

- Say a glass rod drops and breaks into 5 random pieces.
- Let's find the probability that the smallest piece has a length below 0.02.

Simulating the Broken Rod

```
1  # random breaks the rod into k pieces, returning the
2  #length of the shortest one
3  minpiece <- function(k) {
4      breakpts <- sort(runif(k-1))
5      lengths <- diff(c(0,breakpts,1))
6      min(lengths)
7  }
8
9  # returns the approximate probability that the smallest
10 #of k pieces will have a length less than q
11 bkrod <- function(nreps,k,q) {
12     minpieces <- replicate(nreps,minpiece(k))
13     mean(minpieces < q)
14 }
```

Example: Toss a Coin Until k Consecutive Heads

- We toss a coin until we get k heads in a row.
- Let N denote the number of tosses needed.
- Here is code that finds the approximate probability that $N > m$.

```
 1  ngtm <- function(k,m,nreps) {
 2      count <- 0
 3      for (rep in 1:nreps) {
 4          consech <- 0
 5          for (i in 1:m) {
 6              toss <- sample(0:1,1)
 7              if (toss) {
 8                  consech <- consech + 1
 9                  if (consech == k) break
10              } else consech <- 0
11          }
12          if (consech < k) count <- count + 1
13      }
14      return(count/nreps)
```

How Long Should We Run the Simulation?

- Clearly, the larger the value of **nreps** in our examples above, the more accurate our simulation results are likely to be.
- But how large should this value be?
- What measure is there for the degree of accuracy one can expect for a given value of **nreps**?
- These questions will be addressed in Chapter 10 (of Matloff's book).