

STAT 325 – Handout 3 Computer Simulations (1.3)

Computer simulations are very valuable not only for providing insights into understanding concepts of probability, but also for analyzing random process that may be too complex to analyze analytically.

Example 3-1: Mental Coin Tossing

a) Create in your head a sequence of results from tossing a fair coin 100 times. Record your sequence of H's and T's in the grid below:

b) Determine the longest run of consecutive heads in your sequence.

c) Make a guess for the probability that the longest run of heads in a sequence of 100 tosses of a fair coin would exceed 5.

d) Make a guess for the long-term average length of the longest run of heads in a sequence of 100 tosses of a fair coin.

This problem is quite difficult to analyze analytically, so we will simulate this process a large number of times using R. The first step is to simulate tossing a fair coin 100 times, which we can achieve with the following commands:

```
x=c("H","T")
y=sample(x,100,replace=TRUE)
```

Now our strategy will be to go through the sequence of 100 tosses one at a time, keeping track at each step of the length of the current run of heads. Also at each step, we will compare the current run of heads to the longest run attained thus far, replacing the longest run with the current run if it exceeds the previous longest. We can achieve this with these commands, which include a loop that extends through all 100 tosses:

```
for (i in 1:100) {
  if (y[i] == "H") {run = run + 1} else {run = 0}
  if (run > longrun) longrun = run
}
```

Before running through this loop, we need to initialize the values of run and longrun:

```
run = 0; longrun = 0
```

Now, to repeat this process a large number of times (say, 10000), we need an outer loop that will keep track of the longest run of heads for each of the 10000 repetitions. The entire program is:

```
x=c("H","T")
longruns = rep(NA, times = 10000)
for (j in 1:10000) {
  longrun = 0
  run = 0
  y=sample(x,n,replace=TRUE)
  for (i in 1:100) {
    if (y[i] == "H") {run = run + 1} else {run = 0}
    if (run > longrun) longrun = run
  }
  longruns[j] = longrun
}
results=table(longruns); results
mean(longruns)
hist(longruns)
long5 = (longruns > 5)
table(long5)
prob = sum(long5/10000); prob
```

e) Execute this program, and report the (approximate) probability that the longest run of heads in 100 tosses of a fair coin will exceed 5.

f) How would you expect this probability to change if you were to toss a coin 500 times rather than 100 times?

g) Make the appropriate adjustment to the program in order to do 500 tosses, run the program, and report the (approximate) probability. Is this greater or less than the previous probability? Is this what you expected?

h) How would you adjust the program to allow for any number of coin tosses?

i) How would you tweak the program to allow for any number of repetitions?

One thing to remember with any simulation analysis is that the resulting probability is an approximate one. If we let \hat{p} denote the approximate probability, we can determine a margin-

of-error for the approximation using the expression: M-of-E = $1.96 \sqrt{\frac{\hat{p}(1-\hat{p})}{N}}$, where N

represents the number of repetitions. More specifically, taking $\hat{p} \pm 1.96 \sqrt{\frac{\hat{p}(1-\hat{p})}{N}}$ produces an interval that you can be 95% confident to contain the actual probability.

j) Calculate the margin-of-error, and 95% confidence interval, for the two approximate probabilities above.

Example 3-2: Random Babies (cont.)

Recall Example 1-1, in which we considered returning four babies to four mothers at random. We considered the probability that at least one mother would receive the correct baby. We will again approximate this probability with a simulation conducted in R. This time we will learn a clever way to replicate the random process repeatedly without using a loop. We will also write this program generally, without specifying 4 as the number of babies/mothers at the outset.

First we define a vector of integers 1 to n , representing the mothers:

```
x = (1:n)
```

Then we scramble those n integers into a new vector, representing the order in which babies are returned to the mothers:

```
y = sample(x,n,replace=FALSE)
```

Now we check to see which mothers got the correct baby. In other words, we check to see which elements of the vector x equal their counterpart in the vector y :

```
match = (x == y)
```

The vector `match` consists of TRUE and FALSE, but we can count the number of matches by summing this vector, which treats TRUE as 1 and FALSE as 0:

```
nummatches = sum(match)
```

Finally, we check whether at least one mother got the correct baby:

```
event = (nummatches >= 1)
```

a) Execute this series of commands, being sure to first set $n = 4$. Examine and report the values of the vectors x , y , and `match` each time, as well as the numerical value of `nummatches` and the logical value of `event`.

```
x =          y =          match =  
  
nummatches =          event =
```

b) Repeat this series of commands several times.

Now to repeat this process a large number of times, we will define the above commands as a function as follows:

```
match = function(n) {  
  x = (1:n)  
  y = sample(x,n,replace=FALSE)  
  match = (x==y)  
  match  
  nummatches = sum(match)  
  event = (nummatches >= 1)  
}
```

After we define N to be the number of repetitions that we want, we then use an R command called `replicate`, which repeats the above matching function N time. Then we can calculate and print the approximate probability of the event:

```
replicate(N, match(n)) -> result
prob = sum(result)/N; prob
```

c) Do this for $N = 10,000$ repetitions, and report the approximate probability of obtaining at least one match.

d) Repeat for $n = 10$ mothers and babies rather than $n = 4$.

e) What do you notice about these approximate probabilities of obtaining at least one match?

f) Determine 95% confidence intervals for the exact probability in both cases. What do you notice about these intervals?

Example 3-3: Sports Series

Suppose that two sports teams play a series of games. Suppose that team A is better than team B and has probability p of winning any one game, where $p > .5$. The rules are that the first team to take a 2-game lead over the other is declared the winner of the series. What is the probability that team A wins the series?

a) First think about how hard it would be to list sample space of possible outcomes from this random process.

A key component in this program is determining the winner of each game. This is accomplished by generating a random number between 0 and 1. If that number is less than or equal to p , then team A wins the game; otherwise team B wins the game. We also need to keep track of how many games each team has won:

```
x = runif(1,0,1)
if (x <= p) {
  win = "A"
  Awins = Awins + 1
} else {
  win = "B"
  Bwins = Bwins + 1
}
```

A complication with simulating this process is that we do not know in advance how many games will be needed. In principle, there's no limit to how many games might be needed. We'll handle this complication by using a `while` loop, based on the difference in number of games won between the two teams. First we initialize `diff = 0`, then we update `diff` after each game, and the loop continues to run as long as `diff < 2`:

```
while (diff < 2) {
    x = runif(1,0,1)
    if (x <= p) {
        win = "A"
        Awins = Awins + 1
    } else {
        win = "B"
        Bwins = Bwins + 1
    }
    diff = abs(Awins - Bwins)
}
```

The other important components of the program are to initialize various quantities and keep track of results. You can find the complete program on our course webpage.

b) Execute the program with $N = 10,000$ repetitions with $p = .6$. Report the approximate probability that team A wins the series. Also determine a 95% confidence interval for the actual probability.

c) Repeat with $p = .7$.

d) Describe how you would generalize the program so the rules are that a team has to have won k more games than the other in order to be declared the winner of the series.