

Math 5040, Fall 2009  
Assignment 1 on simulation with solutions  
Sept. 28, 2009

Theoretical problems

1. (a) Simulating a standard normal is tricky, but consider the following idea. Let  $X$  and  $Y$  be independent  $N(0,1)$  random variables. Let  $(R, \Theta)$  be the polar coordinates corresponding to the point  $(X, Y)$ . Then  $R^2 = X^2 + Y^2$  has the chi-squared distribution with 2 degrees of freedom, which is exponential with mean 2, and  $\Theta$  is uniform on  $[0, 2\pi)$  and independent of  $R^2$ . (This can be proved from the transformation of densities formula—you need not provide the details.) Use these facts to describe an algorithm for simulating a standard normal random variable  $Z$ .

Solution. We know that  $X = R \cos \Theta$  and  $Y = R \sin \Theta$ , so since  $R^2$  is exponential with parameter  $1/2$ , we can write  $R^2 = -2 \ln U_1$  or  $R = \sqrt{-2 \ln U_1}$ . Also,  $\Theta = 2\pi U_2$ . Thus, by calling the random number generator twice to get  $U_1$  and  $U_2$ , we can get two independent standard normal random variable, namely

$$X = \sqrt{-2 \ln U_1} \cos(2\pi U_2), \quad Y = \sqrt{-2 \ln U_1} \sin(2\pi U_2).$$

We can use either one but might as well use both to make the simulation as efficient as possible.

2. Suppose we want to simulate a gamma random variable (density  $f(x) = \lambda^n x^{n-1} e^{-\lambda x} / (n-1)!$ ) using the rejection method and exponential distribution with density  $g(y) = \beta e^{-\beta y}$  with  $0 < \beta < \lambda$ , which we know how to simulate. Choose  $\beta$  to make your algorithm as efficient as possible ( $\beta$  may depend on  $\lambda$  and  $n$ ), minimizing  $c$  of the rejection method. Then describe your algorithm as simply as possible.

Solution. Let  $c = \sup f(x)/g(x)$ . For  $c$  to be finite, we need  $0 < \beta < \lambda$ . Now

$$\frac{f(x)}{g(x)} = K x^{n-1} e^{-(\lambda-\beta)x}$$

for some constant  $K$ . We set the derivative to 0 and solve for  $x$  to maximize:

$$\frac{n-1}{x} - (\lambda - \beta) = 0, \quad \text{or} \quad x_0 = \frac{n-1}{\lambda - \beta}.$$

We conclude that

$$c = \frac{f(x_0)}{g(x_0)} = \frac{\lambda^n x_0^{n-1} e^{-\lambda x_0} / (n-1)!}{\beta e^{-\beta x_0}} = \frac{\lambda^n}{\beta(n-1)!} \left( \frac{n-1}{\lambda - \beta} \right)^{n-1} e^{-(n-1)}.$$

To minimize this, we must minimize

$$\frac{1}{\beta} \left( \frac{n-1}{\lambda - \beta} \right)^{n-1}$$

or equivalently maximize  $h(\beta) = \beta(\lambda - \beta)^{n-1}$ . We find the optimal  $\beta$  by solving  $h'(\beta) = 0$ , which implies  $\beta = \lambda/n$ . It follows that the optimal  $c$  is

$$c = \frac{\lambda^n}{(\lambda/n)(n-1)!} \left(\frac{n}{\lambda}\right)^{n-1} e^{-(n-1)} = \frac{n^n}{(n-1)!} e^{-(n-1)}$$

Now the second part is to describe the algorithm.

1. Generate independent  $U_1$  and  $U_2$  and put  $Y = -(n/\lambda) \ln U_1$ , which is our exponential( $\lambda/n$ ).

2. Now

$$\frac{f(x)}{cg(x)} = \frac{(\lambda^n/(n-1)!)x^{n-1}e^{-\lambda x}}{(\lambda/n)e^{-(\lambda/n)x}} \frac{(n-1)!}{n^n} e^{n-1} = \left(\frac{\lambda e x}{n}\right)^{n-1} e^{-\lambda(n-1)x/n}.$$

Therefore the rejection method accepts when

$$U_2 \leq \frac{f(Y)}{cg(Y)}$$

and sets  $X = Y$ , otherwise we return to step 1.

3.  $X$  is gamma( $n, \lambda$ ).

Computer problems

3. Write a program in your favorite programming environment to simulate  $E[|Z|]$ , using the algorithm of problem 1. Choose your sample size to ensure accuracy to within 0.001 with confidence level 0.95. Write up your results so that they can be understood by someone who can't necessarily read your code.

This is part (b) of problem 1. Solution. Here is a true BASIC program with sample size 100,000.

```

RANDOMIZE
LET M=50000          !      2*M is the sample size
LET sum=0
LET sum2=0
FOR n=1 to M
  LET u1=rnd
  LET u2=rnd
  LET Z1=abs(sqr(-2*log(u1))*cos(2*pi*u2))
  LET Z2=abs(sqr(-2*log(u1))*sin(2*pi*u2))
  LET sum=sum+Z1+Z2
  LET sum2=sum2+Z1^2+Z2^2
NEXT n
LET mean=sum/(2*M)
LET var=sum2/(2*M)-mean^2
PRINT mean,var
END
.799175             .365193

```

So we must choose the sample size to exceed  $(1.96)^2(0.365193)/(0.001)^2 \approx 1.4$  million. We round up and take it to be 1,500,000.

```

RANDOMIZE
LET M=750000          !      2*M is the sample size
LET sum=0
LET sum2=0
FOR n=1 to M
  LET u1=rnd
  LET u2=rnd
  LET Z1=abs(sqr(-2*log(u1))*cos(2*pi*u2))
  LET Z2=abs(sqr(-2*log(u1))*sin(2*pi*u2))
  LET sum=sum+Z1+Z2
  LET sum2=sum2+Z1^2+Z2^2
NEXT n
LET mean=sum/(2*M)
LET var=sum2/(2*M)-mean^2
PRINT mean,var
END
.797227          .363186

```

Our final estimate is 0.797227. Of course we can compute the exact answer in this case, and it is  $\sqrt{2/\pi} \approx 0.797885$ , so our estimate is indeed within 0.001.

4. Write a program to simulate the probability that a well-shuffled standard deck has at least one adjacent ace and jack, in either order. Choose your sample size to ensure accuracy to within 0.001 with confidence level 0.95. (If you are good at combinatorics, you may be able to find the exact answer. Even so, the problem is to simulate this probability.)

Here we don't have to run a test program for the variance. We can use the fact that the variance of a Bernoulli( $p$ ) random variable is  $p(1-p)$ , which is at most  $1/4$ . So our sample size should be at least  $(1.96)^2(1/4)/(0.001)^2 = 960,400$ . We round up to 1 million.

```

LET n=52
LET sample=1000000
DIM x(52)
FOR i = 1 to n
  LET x(i) = i
NEXT i
LET total=0
RANDOMIZE
FOR run=1 to sample
  FOR k = n to 2 step -1
    LET I = int(k*rnd) + 1
    LET temp = x(I)

```

```

        LET x(I) = x(k)
        LET x(k) = temp
    NEXT k
    LET adj=0
    FOR k=1 to n-1
        IF mod(x(k)-1,13)=0 and mod(x(k+1)-1,13)=10
            or mod(x(k)-1,13)=10 and mod(x(k+1)-1,13)=0 then LET adj=1
    NEXT k
    LET total=total+adj
NEXT run
PRINT "sample size";
PRINT using "#####": sample
PRINT "estimate";
PRINT using " .#####": total/sample
END
sample size 1000000
estimate .486418

```

We have computed that the exact probability is 0.486279043603 (rounded to 12 places). Our estimate is indeed within the required tolerance.

5. Let  $(X_1, X_2, \dots, X_{36})$  be multinomial( $900, 1/36, 1/36, \dots, 1/36$ ), and put  $Y = \max_{1 \leq i \leq 36} X_i$ . (To make this more concrete, think of a roulette wheel with no zeros, which is spun 900 times, and let  $Y$  be the frequency of the most frequent number.) Use a computer to simulate the distribution of  $Y$ , that is, simulate  $P(Y = k)$  for  $k = 0, 1, 2, \dots, 60$ , say. Run your program long enough to get reasonable accuracy (at least 3, maybe 4, decimal places).

First thing to notice is that, since  $900/36 = 25$ , we must have  $Y \geq 25$ . So we will simulate  $P(Y = k)$  for  $k = 25, 26, \dots, 60$ . For each simulated value of  $Y$  we need 900 simulated values of a discrete uniform random variable on  $\{1, 2, \dots, 36\}$ .

```

DIM x(36), est(25:100), cumest(25:100)
LET sample=1000000
FOR n=1 to sample
    FOR k=1 to 36
        LET x(k)=0
    NEXT k
    FOR m=1 to 900
        LET u=int(36*rnd)+1
        LET x(u)=x(u)+1
    NEXT m
    LET max=x(1)
    FOR k=2 to 36
        IF x(k)>max then LET max=x(k)
    NEXT k

```

```

        LET est(max)=est(max)+1
NEXT n
FOR k=25 to 60
  FOR j=k to 60
    LET cumest(k)=cumest(k)+est(j)
  NEXT j
  PRINT using "##":k;
  PRINT using " #.#####":est(k)/sample,cumest(k)/sample
NEXT k
END
25 .000000 1.000000
26 .000000 1.000000
27 .000000 1.000000
28 .000000 1.000000
29 .000021 1.000000
30 .000775 .999979
31 .008314 .999204
32 .036619 .990890
33 .088916 .954271
34 .141000 .865355
35 .166437 .724355
36 .159133 .557918
37 .131560 .398785
38 .096672 .267225
39 .066078 .170553
40 .042480 .104475
41 .026509 .061995
42 .015666 .035486
43 .008954 .019820
44 .005004 .010866
45 .002900 .005862
46 .001477 .002962
47 .000740 .001485
48 .000383 .000745
49 .000186 .000362
50 .000088 .000176
51 .000048 .000088
52 .000024 .000040
53 .000009 .000016
54 .000005 .000007
55 .000001 .000002
56 .000001 .000001
57 .000000 .000000
58 .000000 .000000
59 .000000 .000000
60 .000000 .000000

```

We did only a million runs, but 10,000,000 would have been better.

6. Compare two methods for simulating a roll of a pair of dice. Method 1 simulates each die separately, then adds the results. Method 2 uses the distribution of the sum of two dice and the discrete inverse transformation method. Simulate 10 million dice rolls using each method, and time your programs. Which method is more efficient?

```
DIM dice(10000000)
LET sample=10000000
PRINT time$
FOR n=1 to sample
    LET dice(n)=int(6*rnd)+int(6*rnd)+2
NEXT n
PRINT time$
PRINT
PRINT time$
FOR n=1 to sample
    LET U=int(36*rnd)+1
    SELECT CASE U
    CASE 1
        LET dice(n)=2
    CASE 2,3
        LET dice(n)=3
    CASE 4,5,6
        LET dice(n)=4
    CASE 7,8,9,10
        LET dice(n)=5
    CASE 11,12,13,14,15
        LET dice(n)=6
    CASE 16,17,18,19,20,21
        LET dice(n)=7
    CASE 22,23,24,25,26
        LET dice(n)=8
    CASE 27,28,29,30
        LET dice(n)=9
    CASE 31,32,33
        LET dice(n)=10
    CASE 34,35
        LET dice(n)=11
    CASE 36
        LET dice(n)=12
    END SELECT
NEXT n
PRINT time$
```

END  
17:35:33  
17:36:20  
  
17:36:20  
17:37:34

The approach using two random numbers took 47 seconds in True BASIC.  
The inverse transformation method took 74 seconds.