PROBLEM CORNER

Md S. Warasi
Department of Mathematics and Statistics
Radford University, VA 24142
email: msarker@radford.edu

Problem 1

Random number generators are widely used in statistical applications for simulating data and validating statistical methods. Nowadays, nearly all programming languages are equipped with built-in programs for generating random numbers. The main goal of this problem is to introduce students with a basic algorithm for generating random numbers from a normal (bell-shaped) distribution without implementing any existing programs. This procedure works in two steps.

- Step 1: Generate two numbers U_1 and U_2 from a uniform distribution that is defined over $[0, 1]$. Note that a uniform distribution over $[0, 1]$ is a unit constant function, $f(y) = 1$ for $0 \leq y \leq 1$.
- Step 2: Calculate $X_1 = \sqrt{-2 \ln U_1} \cos(2\pi U_2)$ and $X_2 = \sqrt{-2 \ln U_1} \sin(2\pi U_2)$.

Upon completion, these steps will result in two independent observations, X_1 and X_2, that follow a normal distribution with mean = 0 and variance = 1. This algorithm was introduced by Box and Muller (1958).

The uniform numbers, U_1 and U_2, required for this algorithm can be generated as follows. Select two numbers at random from $0, 1, 2, ..., 200$ with replacement. If either 0 or 200 is selected, discard it and select another number. Divide both numbers by 200 (the maximum possible value). This will yield a pair of independent uniform numbers, U_1 and U_2, that fall between 0 and 1. Next, complete step 2 to obtain X_1 and X_2. Repeat the entire process to simulate multiple pairs of normal variates. For example, to simulate 100 numbers, one needs to repeat these steps 50 times. Note that 200 is a reasonable maximum value for the simple random sampling. However, the performance would be better if a larger value, such as 500, is used.

Do the following using the algorithm described above.

(a) Sample 100 random numbers from a normal distribution that has mean = 0 and variance = 1.

(b) Construct a histogram using the 100 sample points. Is the distribution bell shaped?

(c) Write your own program to implement the algorithm. Generate 1000 random numbers using your program.

Solution: We first generate U_1 and U_2 using the given instructions and then complete step 2. After repeating both steps 50 times, we obtain 100 data points that are observed from a

normal distribution with mean = 0 and variance = 1. A histogram of the generated data is shown in Figure 1 (left). This shows a bell-shaped distribution as expected. To verify whether data are generated from a normal distribution, we provide an additional graph (Q-Q plot) in Figure 1 (right). The Q-Q plot shows a straight line pattern, indicating that the data indeed follow a normal distribution. We do the entire calculation in R; see the R codes below. A general R function that implements the Box-Muller algorithm is also provided for users.

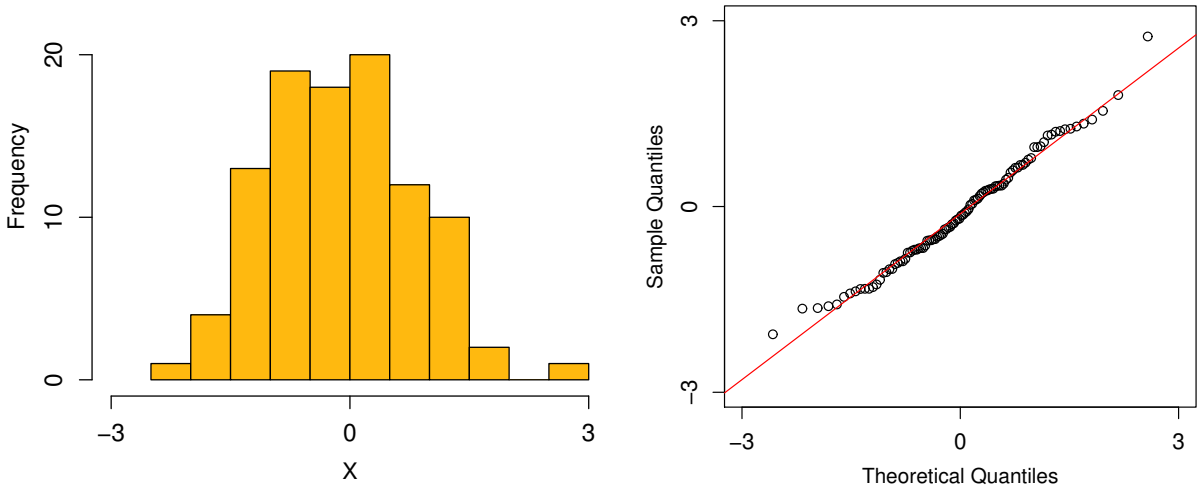

Figure 1: Histogram (left) and normal Q-Q plot (right) of the 100 normal random numbers generated based on the Box-Muller algorithm.

Problem 2

Random numbers can be used to easily approximate areas under any (complicated) functions. For example, the area under the quadratic function $f(y) = y^2$ over $[0, 1]$ is $\int_0^1 f(y)dy = \int_0^1 y^2 dy = 1/3$ (exact result). This can be approximated as follows. Generate 100 random numbers from a uniform distribution that is defined over $[0, 1]$ by simple random sampling as described in Problem 1. Substitute each of the 100 numbers for y in $f(y) = y^2$ to get 100 values of the function and then take their average. This average should be a reasonable approximate for $\int_0^1 y^2 dy$ and should be close to $1/3$. For a more accurate result, use at least 1000 uniform numbers rather than 100.

Use the numerical technique to approximate $\int_0^1 f(y)dy$ for each of the following functions.

(a) $f(y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}}$;

(b) $f(y) = y^2 e^{-y^2}$;

(c) $f(y) = 10^y e^{-e^{-y}}$.

The approximate answers for 2(a), 2(b), and 2(c) are 0.341, 0.189, and 2.336, respectively. Note that the approximation technique described above is valid only for the unit interval $[0, 1]$. This can, however, be generalized easily.

Solution: The functions in 2(a)-2(c) are depicted in Figure 2, where the areas that we are interested in approximating are shaded. Following the given instructions, we first generate 1000 random numbers from the uniform distribution and then approximate the integrals. To illustrate how this has been accomplished, we now describe our calculation for 2(a). Let $U_1, U_2, ..., U_{1000}$ be 1000 uniform variates that we have generated. First, evaluate $f(y) = \frac{1}{\sqrt{2\pi}} e^{-y^2/2}$ at $y = U_1$; that is, calculate $f(U_1) = \frac{1}{\sqrt{2\pi}} e^{-U_1^2/2}$. Continue this to calculate $f(U_2), f(U_3), ..., f(U_{1000})$. Finally, take average of $f(U_1), f(U_2), ..., f(U_{1000})$. This average should be close to 0.341. R codes for this implementation are provided below.

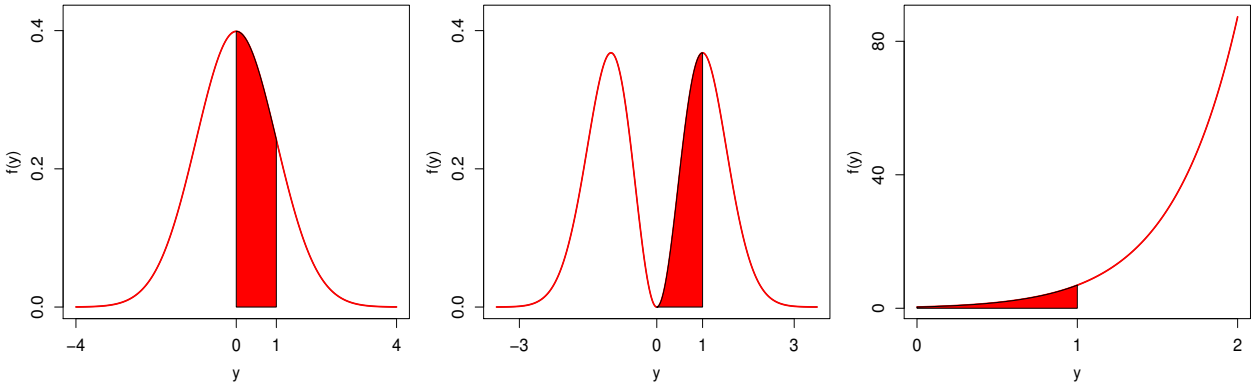

Figure 2: Graphs of the functions in Problem 2. The functions in (a), (b), and (c) are shown in the left, middle, and right, respectively. Areas under the curve for $y \in [0, 1]$ are also shown.

References

Box, G. and Muller, M. (1958). A note on the generation of random normal variates. *Annals of Mathematical Statistics* **29**, 610–611.

R codes:

```
#************************************************#
#            R codes for Problem 1               #
#************************************************#
# 1(a)
#
X <- NULL
for(i in 1:50){
   U1 <- sample(0:200, 1)/200
   U2 <- sample(0:200, 1)/200
   X1 <- sqrt( -2*log(U1) )*cos(2*pi*U2)
   X2 <- sqrt( -2*log(U1) )*sin(2*pi*U2)
   X <- c( X, c(X1,X2) )
}
print( X )

# 1(b):
#
hist(X); qqnorm(X); qqline(X,col=2)

# 1(c): Function to implement the Box-Muller algorithm.
#
rand.norm <- function(n,mx=1000){
   X <- NULL
   for(i in 1:n){
      U1 <- sample(0:mx, 1)/mx
      U2 <- sample(0:mx, 1)/mx
      X1 <- sqrt( -2*log(U1) )*cos(2*pi*U2)
      X2 <- sqrt( -2*log(U1) )*sin(2*pi*U2)
      X <- c( X, c(X1,X2) )
    }
return(X)
}
# Input: n, mx
#  n    = Number of repeatitions, each of which generates two numbers.
# mx    = Maximum of the simple random sampling for uniform variates.

# Output: X
# X     = A vector of two normal random numbers.

# Run this to generate 1000 normal variates:
res <- rand.norm(n=500)
hist(res)
```

```
#************************************************#
#            R codes for Problem 2               #
#************************************************#

# Function to generate uniform numbers.
#
unif <- function(n,mx=1000){
   U <- NULL
   for(i in 1:n){
      U1 <- sample(0:mx, 1)/mx
      U2 <- sample(0:mx, 1)/mx
  U <- c(U, c(U1,U2))
   }
   return(U)
}
U <- unif(500); hist(U)

# 2(a):
#
y <- seq(-4,4,.01)
y0 <- seq(0,1,.01)
f <- function(y) exp(-y^2/2)/sqrt(2*pi)
plot(y, f(y),type="l")
polygon(c(0,y0,1),c(0,f(y0),0),col="red")
U <- unif(500); mean( f(U) )

# 2(b):
#
y <- seq(-3.5,3.5,.01)
f <- function(y) (y^2)*exp(-y^2)
plot(y,f(y),type="l")
polygon(c(0,y0,1),c(0,f(y0),0),col="red")
U <- unif(500); mean( f(U) )

# 2(c):
#
y <- seq(0,2,.01)
f <- function(y) (10^y)*exp(-exp(-y))
plot(y,f(y),type="l")
polygon(c(0,y0,1),c(0,f(y0),0),col="red")
U <- unif(500); mean( f(U) )
```

Note: R is a free programming language widely used for research and applications. R is available for downloaded at `https://www.r-project.org`.