Lab 6: Simulating functions of continuous random variables

Quang Trinh

04.11.18

## Task 1: Simulating a triangle

(Dobrow 6.57/6.8) An isosceles right triangle has side length uniformly distributed on the unit interval . Simulate 500 hypotenuses for such an isosceles right triangle.

### Code set-up

runif(500,0,1) will generate 500 variates from a distribution. Recall that for an isosceles right triangle, the hypotenuse is times the length of one of the sides. You can use this formula to simulate the hypotenuse length from a simulated side legnth.

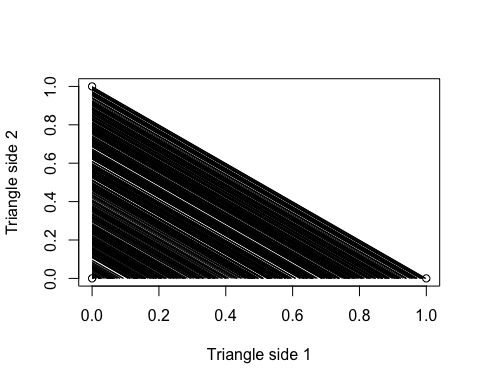
To plot the triangles, you need merely to plot the hypotenuse on a line using the lines function. If we stored our uniform random variates in a vector x, the following code chunk will plot the th hypotenuse generated. This single hypotenuse is not very exciting. But if you wrap this code in a for-loop over the 500 simulation experiments, it will fill in the “triangle space”.

**Note that this code has the eval=FALSE option just to present the code without output. Your code will not use this option.**

# set up the plot by outlining the border from 0 to 1 on each of the triangle side 1 and side 2  
plot(c(0,1,0),c(1,0,0), type="p", xlab="Triangle side 1", ylab="Triangle side 2")  
i = 17 # plot the 17th hypotenuse; remove this line once you set-up the for-loop over i  
# draw the hypotenuse  
lines(c(0,x[i]),c(x[i],0))

### The problem: Simulate 500 hypotenuses for the isosceles right triangle

# Dobrow Exercise 6.57, based on 6.8  
# True expected value and variance are sqrt(2)/2 and 1/6  
  
# set up the plot by outlining the border from 0 to 1 on each of the triangle side 1 and side 2  
plot(c(0,1,0),c(1,0,0), type="p", xlab="Triangle side 1", ylab="Triangle side 2")  
  
# number of simulation  
simnum = 500  
  
# set up x ~ U(0,1)   
x = runif(500,0,1)  
# calculate hypotenuses  
h = sqrt(2) \* x  
  
for(i in 1:simnum) {  
 # draw the hypotenuse  
 lines(c(0,x[i]),c(x[i],0))  
}



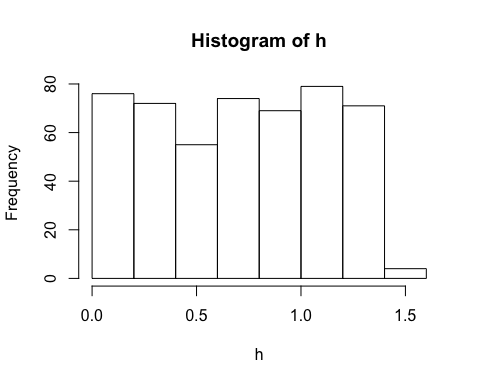
mean(h) # Expected value of hypotenuse

## [1] 0.7084241

var(h) # Variance value of hypotenuse

## [1] 0.1701083

hist(h) # Histogram of hypotenuses.



### Report the following:

* The empirical (simulated) expected length and variance of the length of the hypotenuse.

*The empirical expected length of the hypotenuse is 0.73. The empirical variance of the length of the hypotenuse is 0.17.*

* Since a side has length , the hypotenuse has expected value . . Compare the truth with your empirical values in the previous bullet.

*After running 500 simulations, the empirical expected length of hypothenuse is 0.73 while true value is sqrt(2)/2 = 0.707. The empirical variance value is 0.17 while true value is 1/6 = 0.166. I believe these results are very close to true value. That is, if we would have generated more experiments, we would have had results even closer to the true values.*

* Present a histogram of the simulated hypotenuse lengths. What distribution does it seem to follow and why?

*Because X is uniformly distributed on (0,1) and H is sqrt(2) times X, thus, H is uniformly distributed on (0,sqrt(2)).*

* Present, on a single set of axes, the simulated hypotenuses using the triangle\_plot code chunk above. That is, plot a diagonal line representing the hypotenuse for each simulated value. What do you find?

*I found that the collection of hypotenuse for each simulation will eventually fill in the area of the triangle. That is, if we generate infinite number of simulation, we will fill the entire space (area) of that triangle.*

## Task 2: Simulating Gaussian probabilities

Finding areas under the normal curve (integrating with respect to the normal pdf) is analytically quite difficult due to the form of the pdf. However, Monte Carlo estimates via simulation are quick and accurate.

The Box-Muller algorithm is a popular Gaussian simulator, applying a transformation method to convert uniform variates into normal variates. The algorithm for simulating two random variates and from a distribution is as follow.

**Box-Muller Gaussian simulator**

1. Generate , two uniform random variates.
2. Compute and .

Recall that if we want a variate , we can compute for both and .

### Code set-up

For those who want to try, the Box-Muller Gaussian simulator can be simulated without for-loops by using a matrix of generated uniform variates. Otherwise, code each of the items in the algorithm to generate two variates. Then wrap that in a for-loop to repeat and generate the desired number of Gaussian variates.

The following code will draw a histogram of your variates, , along with a density plot for evaluation.

hist(x, xlab="N(2,2) variates", main="", freq=FALSE)

lines(seq(-4,8,0.05), dnorm(seq(-4,8,0.05),2,sqrt(2)), col="blue")

### The problem

Use the Box-Muller algorithm to simulate 1000 random variates of .

# Generate u1,u2  
u1 = runif(500,0,1)  
u2 = runif(500,0,1)  
  
# Calculate z1,z2  
z1 = sqrt(-2\*log(u1))\*cos(2\*pi\*u2)  
z2 = sqrt(-2\*log(u1))\*sin(2\*pi\*u2)  
  
# Merge z1,z2 into z  
z = c(z1,z2)  
  
# Calculate x base on z  
x = sqrt(2)\*z + 2  
  
mean(x) # Empirical mean

## [1] 2.08668

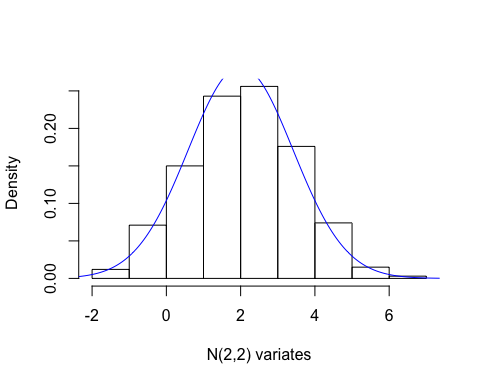
sd(x) # Empirical standard deviation

## [1] 1.45148

median(x) # Empirical median

## [1] 2.080627

hist(x, xlab="N(2,2) variates", main="", freq=FALSE)  
lines(seq(-4,8,0.05), dnorm(seq(-4,8,0.05),2,sqrt(2)), col="blue")



mean( (x > mean(x)-sd(x)) & (x < mean(x)+sd(x)) ) # 1 sd

## [1] 0.68

mean( (x > mean(x)-2\*sd(x)) & (x < mean(x)+2\*sd(x)) ) # 2 sd

## [1] 0.963

mean( (x > mean(x)-3\*sd(x)) & (x < mean(x)+3\*sd(x)) ) # 3 sd

## [1] 0.999

### Report the following:

1. Mean, standard deviation, and median of the variates.

*The empirical mean is 2.01. The empirical standard deviation is 1.44. The empirical median is 2.01.*

1. Histogram of the variates and an overlay with the true density.
2. The proportion of simulated values out of the 1000 that lie one, two, and three standard deviations from the mean (that is, within , , and ). These percentages are estimates of for .

*The proportion of simulated values out of 1000 that lie on 1 standard deviation is 0.69.* *The proportion of simulated values out of 1000 that lie on 2 standard deviation is 0.948.* *The proportion of simulated values out of 1000 that lie on 3 standard deviation is 0.999.*

1. The true values of the three probabilities in (c) are 0.68, 0.95, and 0.997 respectively. Compare your estimates from (c) with the true values. This is called the ``68-95-99.7" rule for the Gaussian/Normal distribution.

*Compare true values with values from part c, we can conclude that our empirical values are very close to the true value. In theory, if we generate infinte number of simulation, we would get exactly 68-95-99.7 porpotion (or empirical rule).*