Lab 6: Simulating Functions of Continuous Random Variables

San Diego State University - STAT550

Aiden Jajo

Task 1: Simulating a triangle

(Dobrow 6.57/6.8) An isosceles right triangle has side length uniformly distributed on the unit interval $(0, 1)$. Simulate 500 hypotenuses for such an isosceles right triangle.

Code set-up

`runif(500,0,1)` will generate 500 variates from a $uniform(0, 1)$ distribution. Recall that for an isosceles right triangle, the hypotenuse is $\sqrt{2}$ times the length of one of the sides. You can use this formula to simulate the hypotenuse length from a simulated side length.

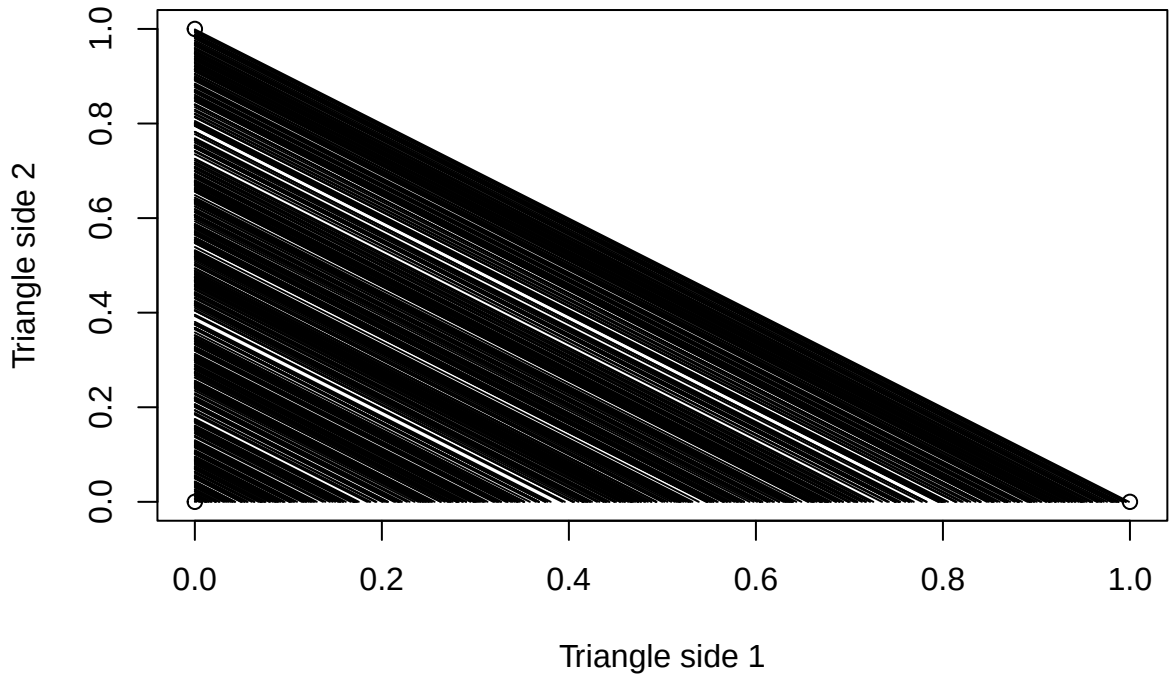
To plot the triangles, you need merely to plot the hypotenuse on a line using the `lines` function. If we stored our uniform random variates in a vector `x`, the following code chunk will plot the ith hypotenuse generated. This single hypotenuse is not very exciting. But if you wrap this code in a for-loop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

```
# 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

# Generate 500 uniform random side lengths
sides <- runif(500, 0, 1)

# Calculate hypotenuses using Pythagorean theorem for isosceles right triangle
# For isosceles right triangle: hypotenuse = sqrt(2) * side
hypotenuses <- sqrt(2) * sides

# Plot all 500 triangles
plot(c(0, 1, 0), c(1, 0, 0), type = "p",
     xlab = "Triangle side 1",
     ylab = "Triangle side 2")

# Draw each hypotenuse as a diagonal line
```

```r
for (i in 1:500) {
  lines(c(0, sides[i]), c(sides[i], 0))
}
```




```r
# Calculate empirical statistics
sample_mean <- mean(hypotenuses)
sample_var <- var(hypotenuses)

cat("Sample mean:", sample_mean, "\n")
```

```
## Sample mean: 0.7032896
```

```r
cat("Sample variance:", sample_var, "\n")
```

```
## Sample variance: 0.1750462
```

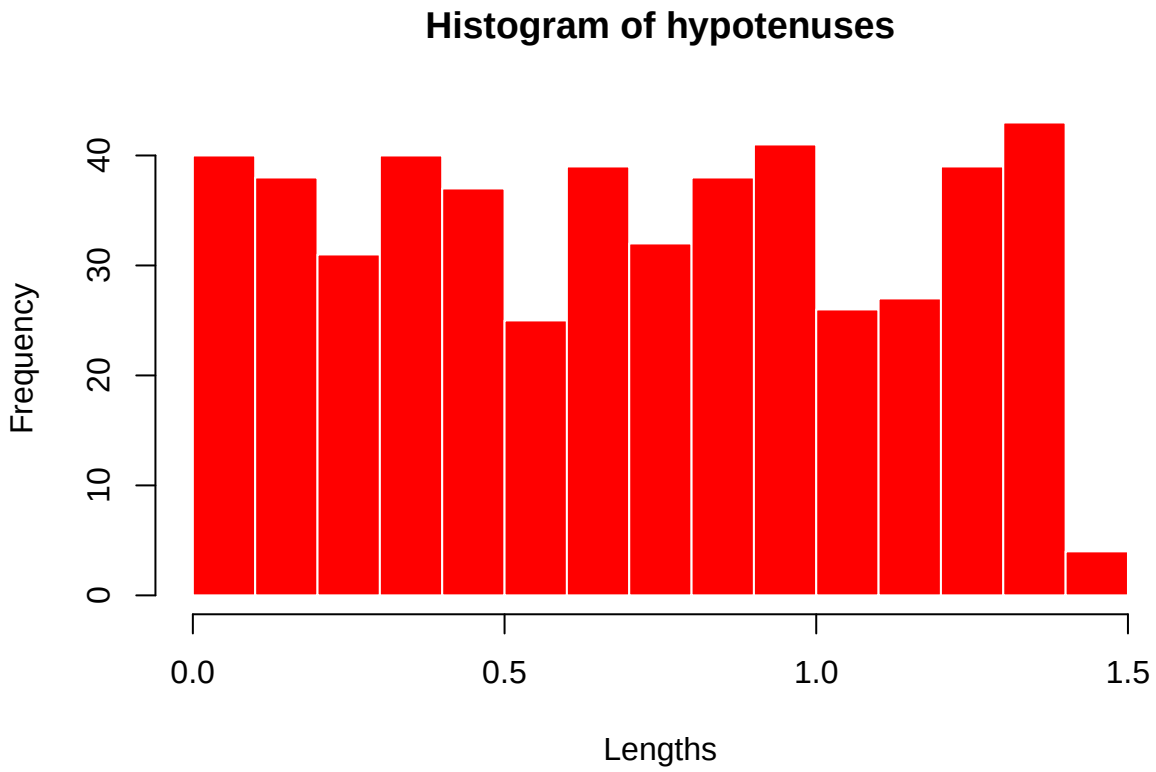
```r
# Calculate theoretical values
true_mean <- sqrt(2)/2
true_var <- 1/6

cat("True mean:", true_mean, "\n")
```

```
## True mean: 0.7071068
```

```r
cat("True variance:", true_var, "\n")
```

```
## True variance: 0.1666667
```

```r
# Create histogram of hypotenuse lengths
hist(hypotenuses,
     breaks = 20,
     col = "red",
     xlab = "Lengths",
     ylab = "Frequency",
     border = "white",
     main = "Histogram of hypotenuses")
```



Report the following:

- The empirical (simulated) expected length and variance of the length of the hypotenuse.
- Since a side has length $X \sim U(0,1)$, the hypotenuse $H = X \cdot \sqrt{2}$ has expected value $E(H) = \sqrt{2} \cdot E(X) = \sqrt{2}/2$. $VAR(H) = 2VAR(X) = 1/6$. Compare the truth with your empirical values in the previous bullet.

The values that I got after running were similar to the truth values. My empirical expected length is approximately 0.7. The empirical length variance is approximately 0.18.

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

The distribution seems to follow a uniform shape. This makes sense because random sampling is being used.

- 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?

What I found was that the plot had a lot of lines that looked to be triangles filled in black. The spots varied in shade as some areas seemed to be darker than others which indicates that some values were being repeated.

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 Z_1 and Z_2 from a $N(0,1)$ distribution is as follow.

Box-Muller Gaussian simulator

i. `Generate` $U_1, U_2 \sim U(0,1)$, two uniform random variates.

ii. `Compute` $Z_1 = \sqrt{-2\ln(U_1)}\cos(2\pi U_2)$ and $Z_2 = \sqrt{-2\ln(U_1)}\sin(2\pi U_2)$.

Recall that if we want a variate $X \sim N(\mu, \sigma^2)$, we can compute $X = \sigma Z + \mu$ for both Z_1 and Z_2.

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, x, 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 $X \sim N(2,2)$.

```
# Box-Muller algorithm implementation
box_muller <- function(n) {
  m <- ceiling(n/2)  # Generate pairs, round up for odd n

  # Generate uniform random variates
  U1 <- runif(m)
  U2 <- runif(m)

  # Box-Muller transformation
```

```r
  Z1 <- sqrt(-2*log(U1)) * cos(2*pi*U2)
  Z2 <- sqrt(-2*log(U1)) * sin(2*pi*U2)

  # Combine and return exactly n values
  Z <- c(Z1, Z2)
  return(Z[1:n])
}

# Generate 1000 standard normal variates
Z <- box_muller(1000)

# Transform to N(2, 2): X = sigma*Z + mu where sigma = sqrt(2), mu = 2
X <- sqrt(2)*Z + 2

# Calculate descriptive statistics
sim_mean <- mean(X)
sim_mean
```

```
## [1] 2.056239
```

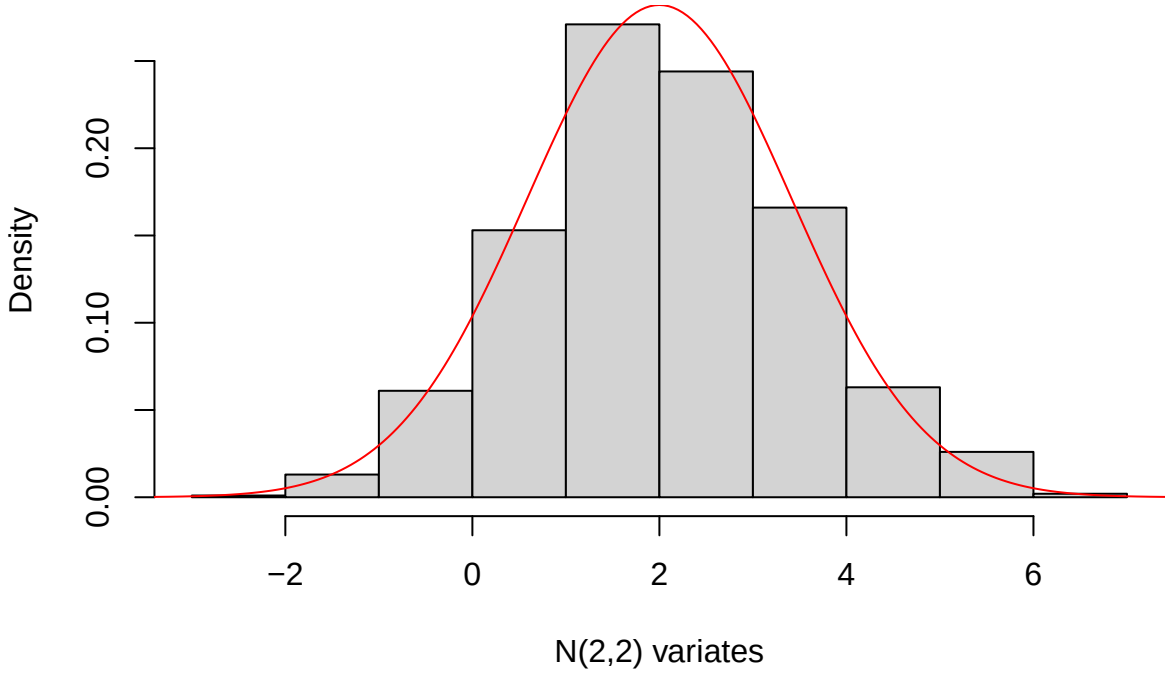
```r
sim_sd <- sd(X)
sim_sd
```

```
## [1] 1.433712
```

```r
sim_median <- median(X)
sim_median
```

```
## [1] 2.016288
```

```r
# Create histogram with theoretical normal density overlay
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 = "red")
```




```r
# Calculate proportions within 1, 2, and 3 standard deviations
mu <- 2
sigma <- sqrt(2)

# Proportion within 1 SD (mu ± sigma)
prop_1sd <- mean(X > (mu - sigma) & X < (mu + sigma))
prop_1sd
```

```
## [1] 0.676
```

```r
# Proportion within 2 SD (mu ± 2*sigma)
prop_2sd <- mean(X > (mu - 2*sigma) & X < (mu + 2*sigma))
prop_2sd
```

```
## [1] 0.949
```

```r
# Proportion within 3 SD (mu ± 3*sigma)
prop_3sd <- mean(X > (mu - 3*sigma) & X < (mu + 3*sigma))
prop_3sd
```

```
## [1] 0.997
```

Report the following:

a) Mean, standard deviation, and median of the variates.

Mean: 2.056, SD: 1.434, Median: 2.016

b) Histogram of the variates and an overlay with the true density.

c) The proportion of simulated values out of the 1000 that lie one, two, and three standard deviations from the mean (that is, within $\mu \pm \sigma$, $\mu \pm 2\sigma$, and $\mu \pm 3\sigma$). These percentages are estimates of $P(\mu - i\sigma < X < \mu + i\sigma)$ for $i = 1, 2, 3$.

1 SD: 67.6%, 2 SD: 94.9%, 3 SD: 99.7%

d) 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.

My estimates from (c) are very similar to the true values, validating the Box-Muller algorithm and demonstrating the empirical rule for normal distributions.