Lab3

Ayesha Gamage - ayega981, N Muditha Cherangani-mudch175

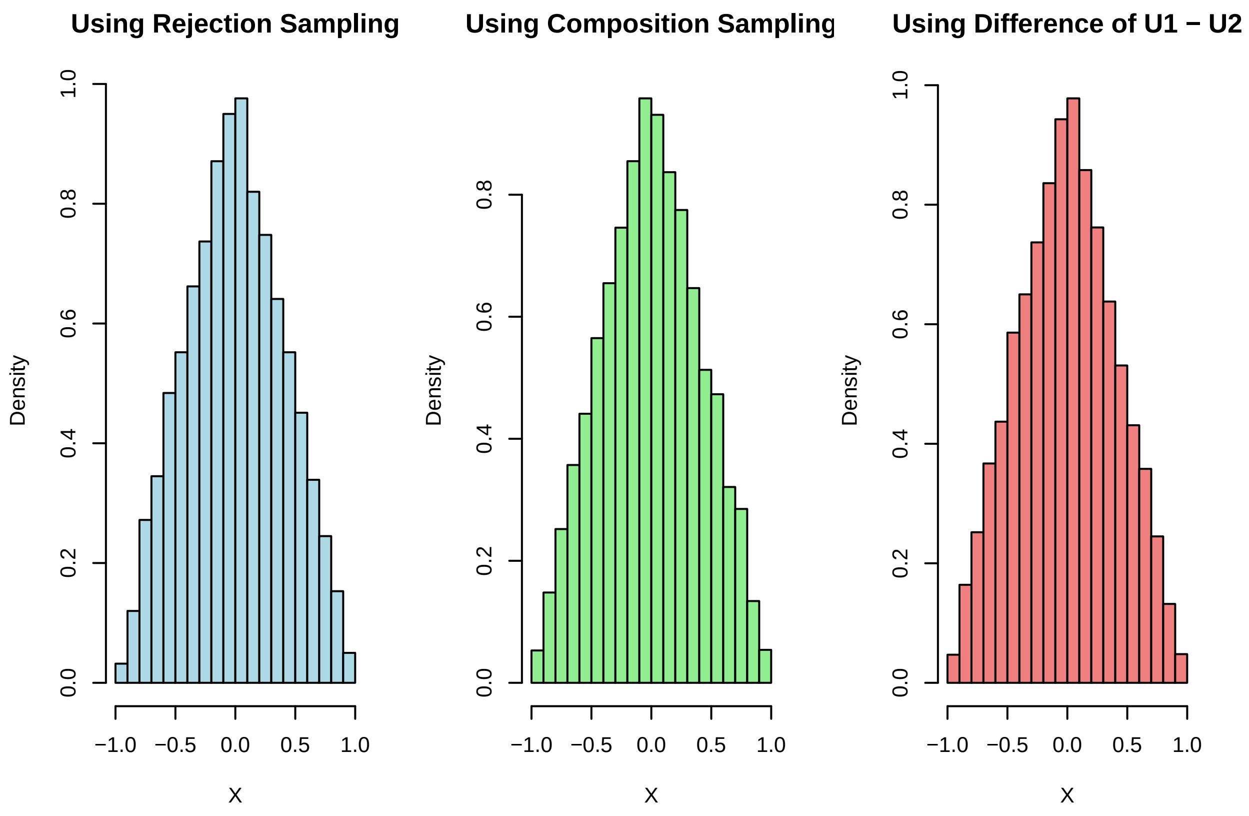
2023-11-21

Question 1

$$f(x) = \begin{cases} 0 & \text{if } x < -1 \text{ or } x > 1 \\ x+1 & \text{if } -1 \leq x \leq 0 \\ 1-x & \text{if } 0 < x \leq 1 \end{cases}$$

"Envelope" function range I have chooses x range -1.5 to 1.5

Using generated method compute 10000 random variables and plotted histograms.



All histograms are closely resemble a triangle. We can suggest all method effectively generating samples.

```
## Variance (Rejection Sampling): 0.1646433
```

```
## Variance (Composition Sampling): 0.1670282
## Variance (Difference of U1 - U2): 0.1656255
```

When compute the variance of sample from each method, variances are nearly equal. When consider about the generated values and plots its difficult to find a preferred method to generate random numbers. Then we need to consider about complexity ,simplicity and specific requirement of the application.The difference of uniforms method is very simple and is often efficient due to the known properties of the difference of uniform random variables.

Question 2

a. The double exponential distribution (Laplace distribution), has the probability density function (PDF) given by,

$$DE(\mu, \lambda) = \frac{\lambda}{2} exp(-lambda|x - \mu|)$$

u is a random number from Unif(0, 1),

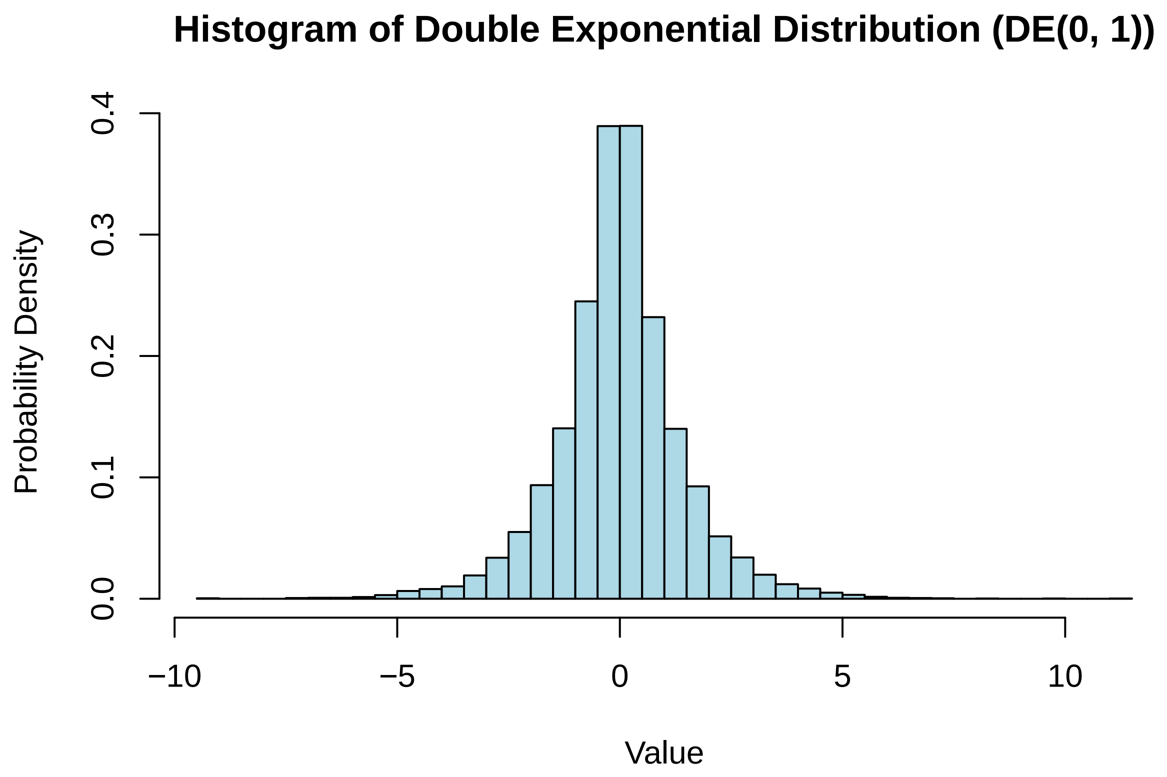
1. If $u < 0.5$:
$$x = \mu - \frac{1}{\lambda} ln(2u)$$
2. If $u \geq 0.5$:
$$x = \mu \ + \frac{1}{\lambda} ln(2(1 - u))$$

```r
# Function to generate random numbers from DE(0, 1) using inverse CDF method
generate_double_exponential <- function(n) {
  u <- runif(n)
  mu <- 0
  lam <- 1

  # Using inverse CDF to generate double exponential random numbers
  x <- ifelse(u < 0.5, mu - log(2 * u) / lam, mu + log(2 * (1 - u)) / lam)
  return(x)
}

random_numbers <- generate_double_exponential(10000)

hist(random_numbers, breaks = 50, prob = TRUE, col = 'lightblue', border = 'black', main = 'Histogram o
```



The double exponential distribution has heavier tails than the normal distribution, and it is symmetric around its mean. The tails of the histogram should extend further than a normal distribution.The histogram aligns with the expected shape, it suggests that the inverse CDF method has been successfully used to generate random numbers from the desired distribution.

b. implementing rejection sampling with DE (0,1) as the envelope to generate N(0,1)

1.Choose an Envelope Distribution: The envelope distribution is DE(0,1), a double exponential distribution with a mean of 0 and a scale parameter of 1. This distribution will serve as an envelope to sample from, and we will accept or reject samples based on their relation to the target distribution, N(0,1).

2.Choose a Constant a: For the standard normal distribution N(0,1) as the target distribution and DE(0,1) as the envelope distribution, the correct choice for a is typically derived from the ratio of their maximum probability density function (PDF) values. a is chosen as

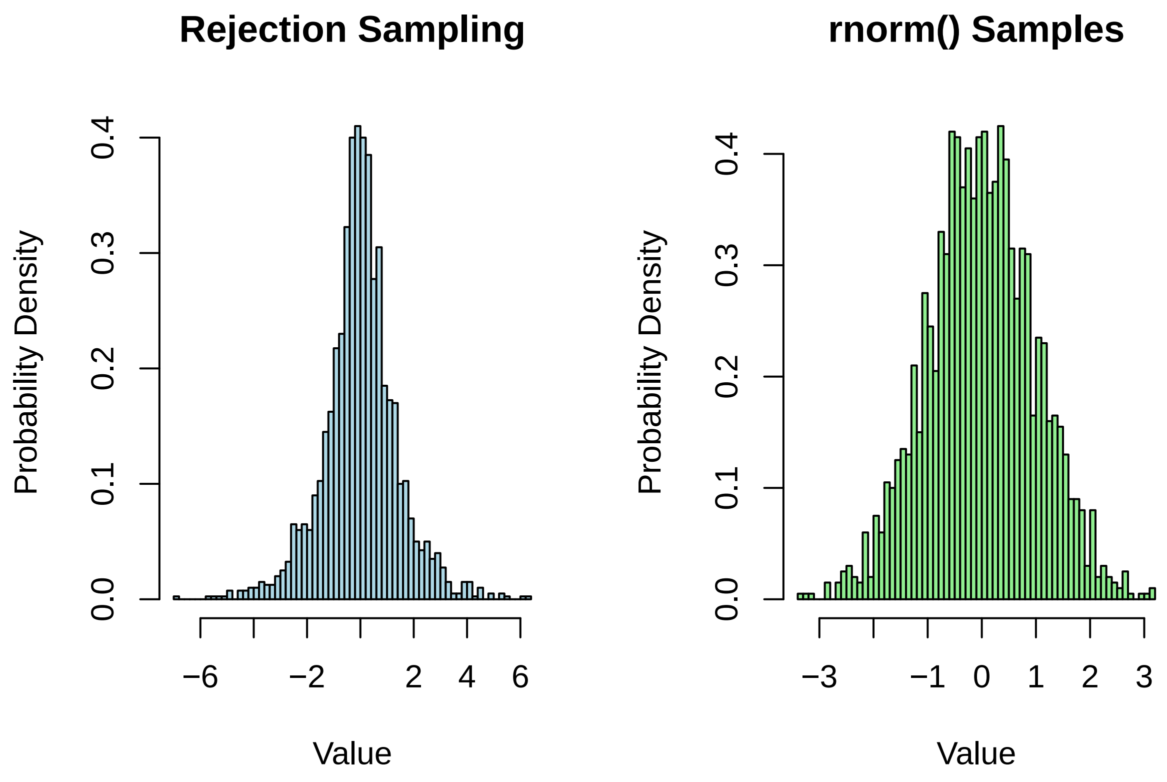
$$a = \sqrt{2/\pi}$$

3.Generate Random Samples: For each sample, generate a random number u from the envelope distribution DE(0,1) and another random number v from a uniform distribution on [0,1). If

$$v \leq \frac{f_{target(u)}}{a * f_{envelope(u)}}$$

u as a sample from the target distribution. Otherwise, reject u and repeat the process until an acceptable sample is obtained.

```
## Rejection Rate (R): 0.1895
## Expected Rejection Rate (ER): 0.5
```



Rejection Sampling Histogram (Left):

This histogram represents the distribution obtained from rejection sampling using DE(0,1) as the envelope to generate N(0,1) variables.It should resemble the standard normal distribution (N(0,1)).

rnorm() Samples Histogram (Right):

This histogram represents the distribution obtained directly from the rnorm() function, which generates random numbers from N(0,1).It serves as a reference for the true standard normal distribution.

The central tendency, spread, and overall symmetry of the histograms are close in shape, which suggests that the rejection sampling method is successful in approximating the target distribution. However, there are overall symmetry differences, which might be due to the randomness of the sampling process.

Appendix: All r code for this report

```
knitr::opts_chunk$set(echo = TRUE)
##############density function###########
density_function <- function(x) {
  if (x < -1 || x > 1) {
    return(0)
  } else if (-1 <= x && x <= 0) {
    return(x + 1)
```

```r
  } else if (0 < x && x <= 1) {
    return(1 - x)
  }
  return(0)
}

##############constant envelope function##########
envelope_function <- function(x) {
  if (-1.5 <= x && x <= 1.5) {
    return(1.5)
  } else {
    return(0)
  }
}

######random generator for X using rejection sampling############
generate_using_rejection <- function(n) {
  samples <- numeric(n)
  for (i in 1:n) {
    while (TRUE) {
      x <- runif(1, -1.5, 1.5)
      u <- runif(1)
      if (u <= (density_function(x) /  envelope_function(x))) {
        samples[i] <- x
        break
      }
    }
  }
  return(samples)
}
######Generate Y using inverse cumulative distribution function method#########
generate_Y <- function(n) {
  u <- runif(n)
  Y <- 1-sqrt(1-u)
  return(Y)
}

#######Composition sampling###########################
generate_X_by_composition <- function(n) {
  Y_positive <- generate_Y(n)
  Y_negative <- -generate_Y(n)
  X <- ifelse(runif(n) < 0.5, Y_positive, Y_negative)
  return(X)
}


####random variables, U1-U2#######################
generate_X_with_difference <- function(n) {
  U1 <- runif(n)
  U2 <- runif(n)
  X <- U1-U2
  return(X)
}
```

```r
set.seed(123)  # Setting seed for reproducibility
samples_rejection <- generate_using_rejection(10000)
samples_composition <- generate_X_by_composition(10000)
samples_difference <- generate_X_with_difference(10000)


par(mfrow = c(1, 3))
hist(samples_rejection, main = "Using Rejection Sampling", col = "lightblue", xlab = "X", freq = FALSE)
hist(samples_composition, main = "Using Composition Sampling", col = "lightgreen", xlab = "X", freq = FA
hist(samples_difference, main = "Using Difference of U1 - U2", col = "lightcoral", xlab = "X", freq = FA

# Calculate variance for each method
var_rejection <- var(samples_rejection)
var_composition <- var(samples_composition)
var_difference <- var(samples_difference)

cat("Variance (Rejection Sampling):", var_rejection, "\n")
cat("Variance (Composition Sampling):", var_composition, "\n")
cat("Variance (Difference of U1 - U2):", var_difference, "\n")
# Function to generate random numbers from DE(0, 1) using inverse CDF method
generate_double_exponential <- function(n) {
  u <- runif(n)
  mu <- 0
  lam <- 1

  # Using inverse CDF to generate double exponential random numbers
  x <- ifelse(u < 0.5, mu - log(2 * u) / lam, mu + log(2 * (1 - u)) / lam)
  return(x)
}

random_numbers <- generate_double_exponential(10000)

hist(random_numbers, breaks = 50, prob = TRUE, col = 'lightblue', border = 'black', main = 'Histogram of

# Function to compute rejection rate for a single trial
compute_rejection_rate <- function(samples, a) {
  accepted_samples <- sum(runif(length(samples)) <= dnorm(samples) / (a * dexp(samples)))
  rejection_rate <- 1 - (accepted_samples / length(samples))
  return(rejection_rate)
}

# Function to generate random numbers from DE(0, 1)
generate_double_exponential <- function(n) {
  u <- runif(n)
  x <- ifelse(u < 0.5, log(2 * u),-log(2 * (1 - u)))
  return(x)
}

# Constants
a <- sqrt(2 / pi)
n_samples <- 2000

# Generate 2000 random numbers N(0, 1) using rejection sampling
```

```r
rejection_samples <- generate_double_exponential(n_samples)

# Generate 2000 random numbers N(0, 1) using rnorm()
standard_normal_samples <- rnorm(n_samples)

# Compute rejection rate (R)
rejection_rate <- compute_rejection_rate(rejection_samples, a)

# Compute expected rejection rate (ER)
expected_rejection_rate <- 1 - (dnorm(0) / (a * dexp(0)))


cat("Rejection Rate (R):", rejection_rate, "\n")
cat("Expected Rejection Rate (ER):", expected_rejection_rate, "\n")


par(mfrow = c(1, 2))
hist(rejection_samples, breaks = 50, prob = TRUE, col = 'lightblue', border = 'black', main = 'Rejectio
hist(standard_normal_samples, breaks = 50, prob = TRUE, col = 'lightgreen', border = 'black', main = 'r
par(mfrow = c(1, 1))
```