Computational Statistics Computer Lab 3 (Group 7)

Qinyuan Qi(qinqi464)   Satya Sai Naga Jaya Koushik Pilla (satpi345)

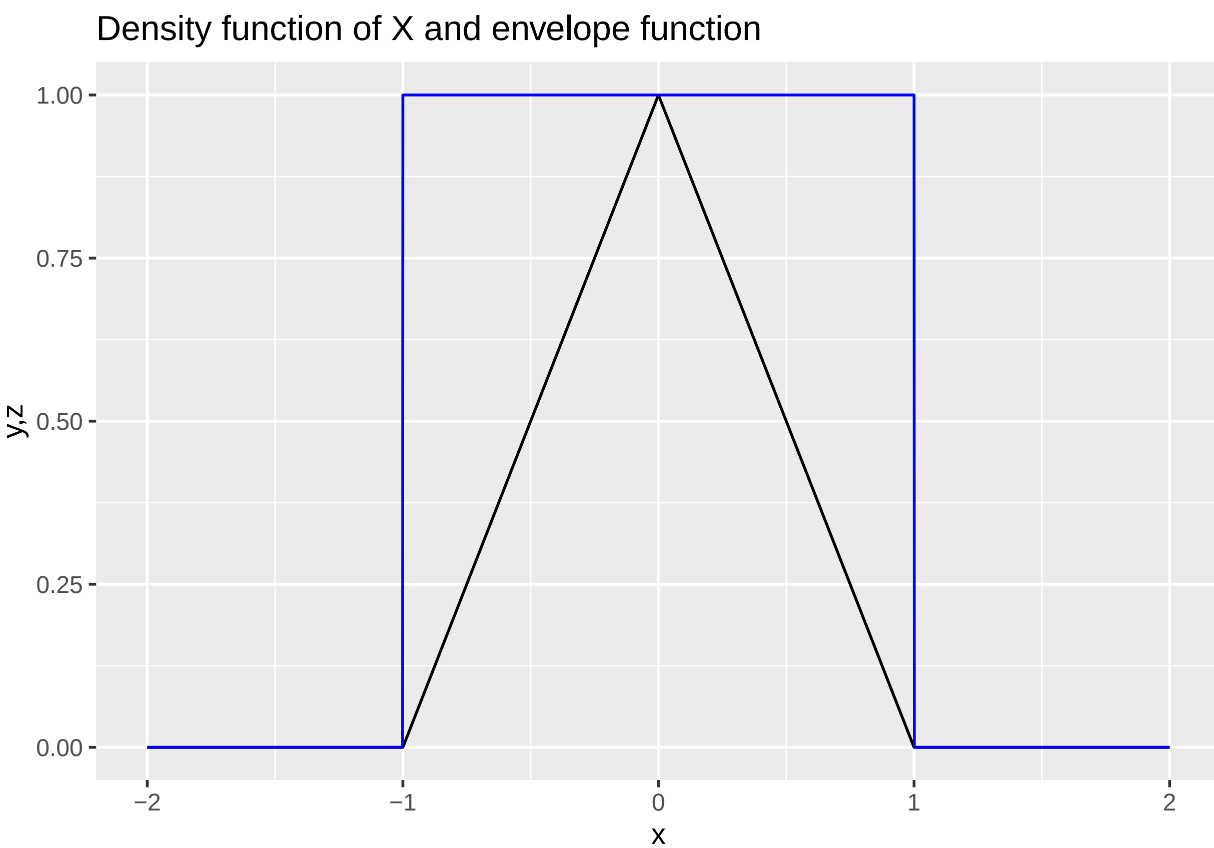
2023-11-21

Question 1: Sampling algorithms for a triangle distribution (Solved by Qinqi Qi)

Answer:

(A)

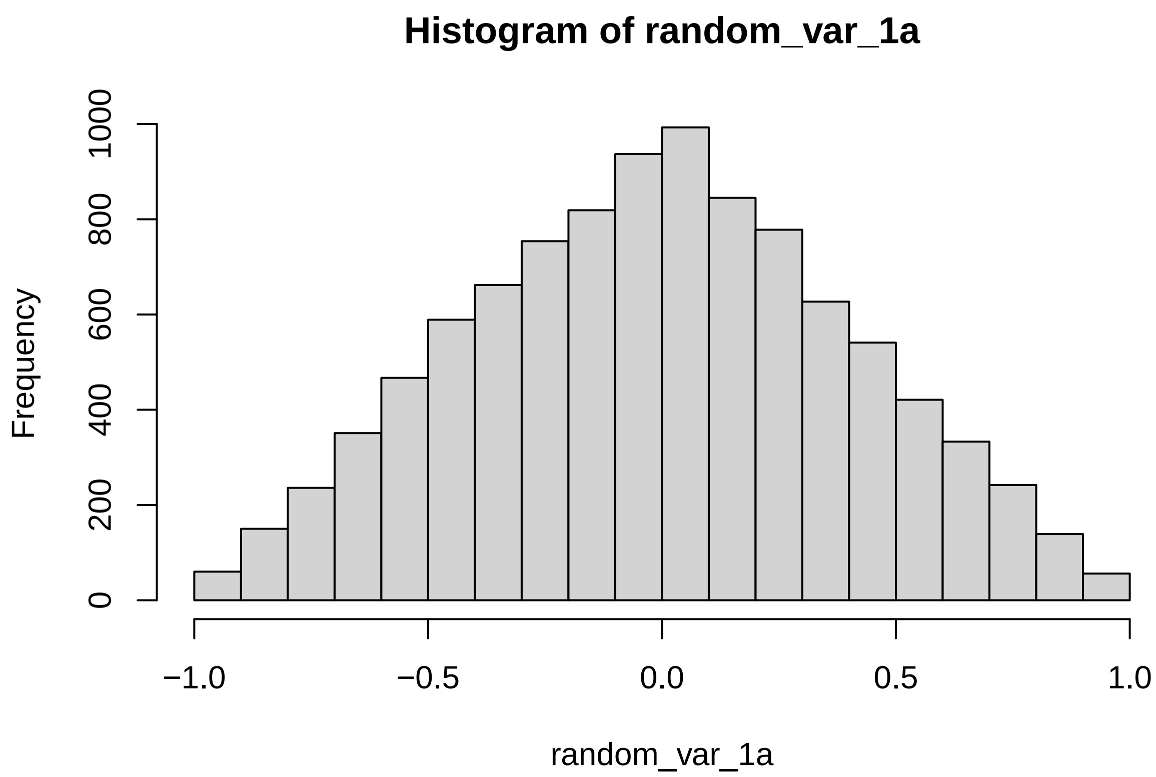
Since we need to find a envelope function for the density function, so we need to plot the density function first. let $x \in [-3, 3]$, and we know it's
triangle sharped function. We also can plot it using the code(Check appendix). the target function $g(x) = 1$ in $[-1, 1]$, otherwise $g(x) = 0$



After we get the plot, we can see that the density function is a triangle we will use an envelope function to cover the density function, and we can use function $ g(x) = 1 $ to cover the density function in the [-1,1] range.

Now we will write a code for the rejection sampling algorithm our g function is $g(x) = 1$ in the range $[-1, 1]$, otherwise $g(x) = 0$ our e function is $e(x) = 1/a$ in the range $[-1, 1]$, otherwise $e(x) = 0$ and we will use $a = 0.5$ here.

```r
generate_random_var_1a <- function(n, a) {
  samples <- numeric(n)
  # Generate a sample from the proposal distribution in [-1,1]
  count <- 1
  e <- 1 / a

  while (count <= n) {

    x <- runif(1, -1, 1)
    # Calculate the acceptance probability
    u <- runif(1)

    # since g will always be 1 in the range [-1,1], so y = 1
    # f1(Y) = 1 - Y and e(Y) = 1 / a = 2
    # f2(Y) = Y + 1 and e(Y) = 1 / a = 2
    # so we have f(Y) / e(Y) = (1 - Y) / 2 or Y+1 / e

    # Calculate the acceptance probability

    if (x > 0 && u <= (1 - x) / e) {
      samples[count] <- x
      count <- count + 1
    }

    if (x <= 0 && u <= (x + 1) / e) {
      samples[count] <- x
      count <- count + 1
    }
  }

  return(samples)
}

a <- 0.5
random_var_1a <- generate_random_var_1a(10000, a)
hist(random_var_1a)
```



(B)

To generate a random variable from the density function using Inverse CDF, we will need to calculate the CDF function first.

CDF function as follows:

$$CDF = \begin{cases} 0, & \text{if } x \in (-\infty, -1) \\ \frac{x^2}{2} + x, & \text{if } x \in (-1, 0) \\ x - \frac{x^2}{2}, & \text{if } x \in (0, 1) \\ 1, & \text{if } x \in (1, \infty) \end{cases}$$

Since -Y has a triangle distribution on $[-1, 0]$, so we have

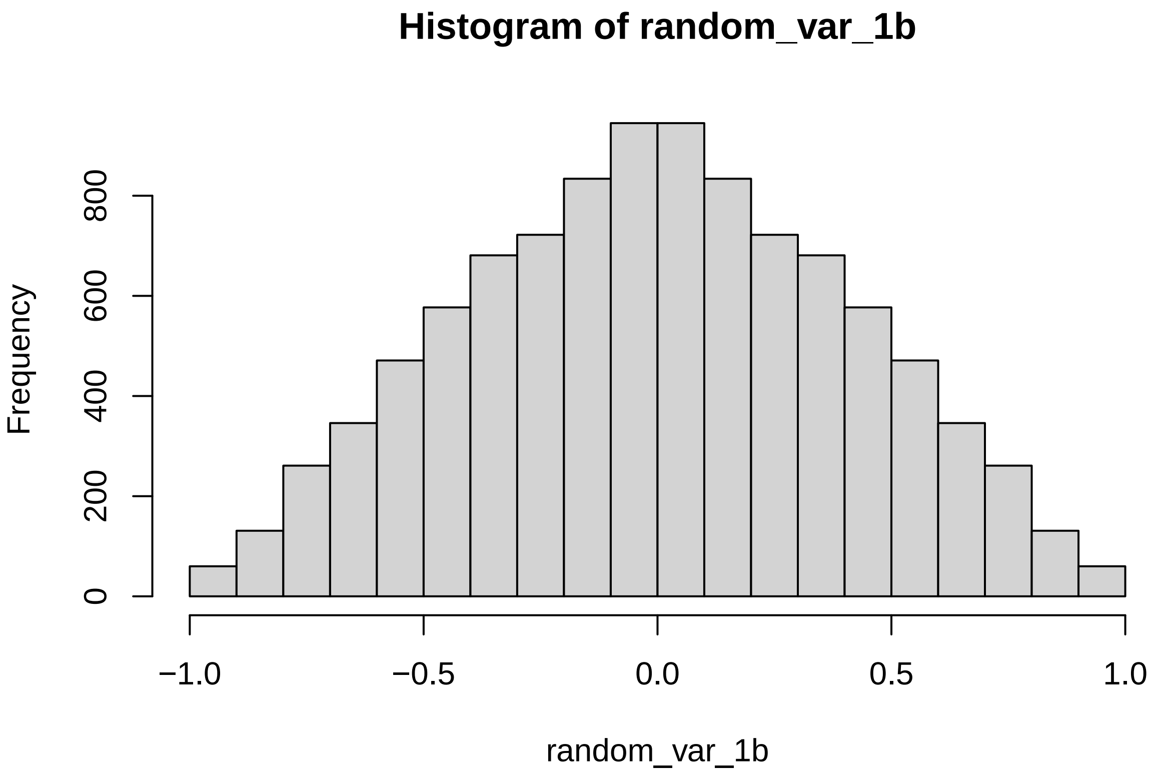
$$F^{-1} = \begin{cases} x = -1 + \sqrt{1 - 2y}, & \text{if } x \in [-1, 0) \\ x = 1 - \sqrt{1 - 2y}, & \text{if } x \in [0, 1] \\ 0, Otherwise \end{cases}$$

The following is the function to generate a random variable from it.

```
########################## [ 1 b ] ##############################################
generate_random_var_1b <- function(n){
  u <- runif(n)
  x1 <-  - 1 + sqrt(1 - 2 * u)
  x2 <-  1 - sqrt(1 - 2 * u)
  return(c(x1, x2))
}

random_var_1b <- generate_random_var_1b(10000)
```

```
## Warning in sqrt(1 - 2 * u): NaNs produced

## Warning in sqrt(1 - 2 * u): NaNs produced
```

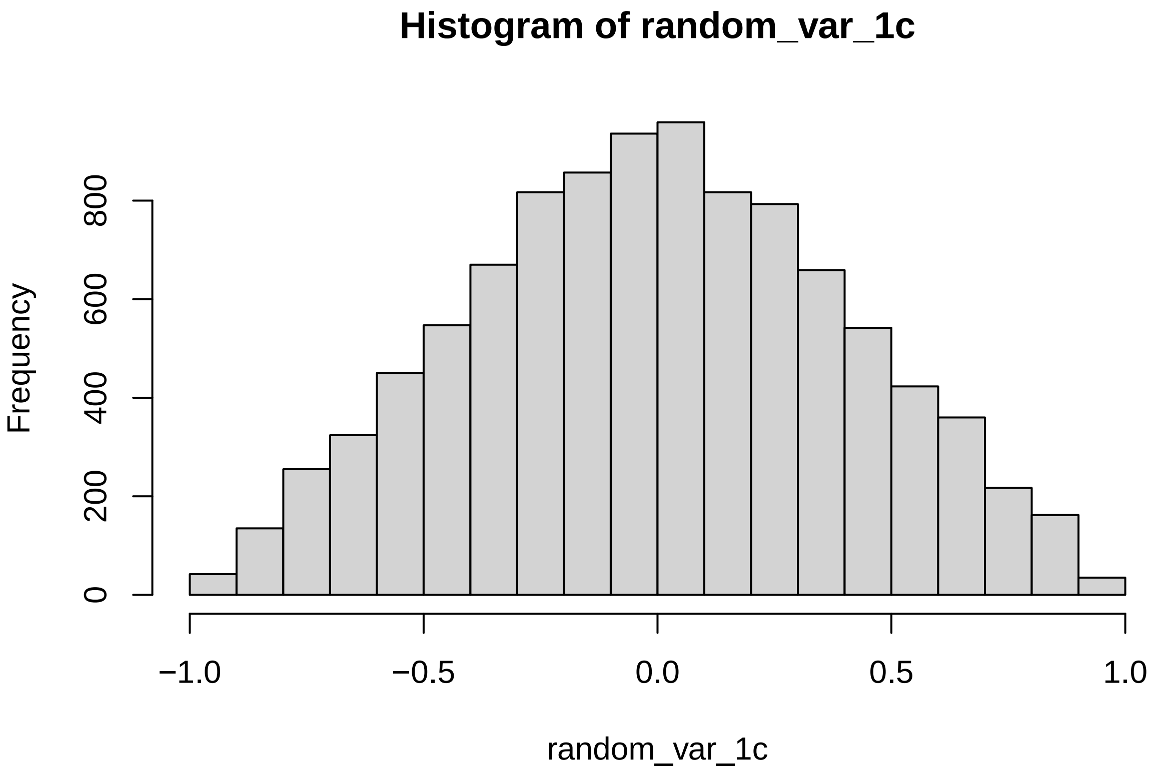
```
hist(random_var_1b)
```



(C)

The code to generate a random variable following a triangle distribution as follows.

```
########################## [ 1 c ] ##############################################
generate_random_var_1c <- function(n){
  u1_1c <- runif(n)
  u2_1c <- runif(n)
  return(u1_1c - u2_1c)
}

random_var_1c <- generate_random_var_1c(10000)
hist(random_var_1c)
```



(D)

Since we already plot the data, we will not plot them here.

since the 1c is the simplest way to generate the random variable, but since it can not adapt to some specific distribution, for 1b , since we need to calculate CDF and inverse function, in some cases, it's hard to do that, so we will use 1a to generate the random variable.

Question 2: Laplace distribution (Solved by Satya Sai Naga Jaya Koushik Pilla)

Answer:

```
########################## [ 2 a ] ######################################
```

(a)

```
########################## [ 2 b ] #############################################
```

(b)

Appendix: Code for this report

```
########################## Init code for question 1 ############################
rm(list = ls())
########################## [ 1 a ] ##############################################

x <- seq(-2, 2, by = 0.001)
y <- rep(0, length(x))
z <- rep(0, length(x))

x1_index <- which(x > 1 | x < -1)
x2_index <- which(x >= -1 & x <= 0)
x3_index <- which(x > 0 & x <= 1)

y[x1_index] <- 0
y[x2_index] <- x[x2_index] + 1
y[x3_index] <- 1 - x[x3_index]

z0_index <- which(x > 1 | x < -1)
z1_index <- which(x >= -1 & x <= 1)

z[z1_index] <- 1
z[-z1_index] <- 0

data <- data.frame(x, y, z)

graph <- ggplot2::ggplot(data) +
  ggplot2::geom_line(mapping=ggplot2::aes(x=x,y=y)) +
  ggplot2::geom_line(mapping=ggplot2::aes(x=x,y=z),color="blue") +
  ggplot2::ggtitle("Density function of X and envelope function") +
  ggplot2::xlab("x") +
  ggplot2::ylab("y,z")

graph
generate_random_var_1a <- function(n, a) {
  samples <- numeric(n)
  # Generate a sample from the proposal distribution in [-1,1]
  count <- 1
  e <- 1 / a

  while (count <= n) {

    x <- runif(1, -1, 1)
    # Calculate the acceptance probability
    u <- runif(1)

    # since g will always be 1 in the range [-1,1], so y = 1
    # f1(Y) = 1 - Y and e(Y) = 1 / a = 2
    # f2(Y) = Y + 1 and e(Y) = 1 / a = 2
    # so we have f(Y) / e(Y) = (1 - Y) / 2 or Y+1 / e

    # Calculate the acceptance probability

    if (x > 0 && u <= (1 - x) / e) {
```

```
      samples[count] <- x
      count <- count + 1
    }

    if (x <= 0 && u <= (x + 1) / e) {
      samples[count] <- x
      count <- count + 1
    }
  }

  return(samples)
}

a <- 0.5
random_var_1a <- generate_random_var_1a(10000, a)
hist(random_var_1a)
########################## [ 1 b ] #############################################
generate_random_var_1b <- function(n){
  u <- runif(n)
  x1 <-  - 1 + sqrt(1 - 2 * u)
  x2 <-  1 - sqrt(1 - 2 * u)
  return(c(x1, x2))
}

random_var_1b <- generate_random_var_1b(10000)
hist(random_var_1b)
########################## [ 1 c ] #############################################
generate_random_var_1c <- function(n){
  u1_1c <- runif(n)
  u2_1c <- runif(n)
  return(u1_1c - u2_1c)
}

random_var_1c <- generate_random_var_1c(10000)
hist(random_var_1c)
########################## Init code for question 2############################
knitr::opts_chunk$set(echo = TRUE)
rm(list = ls())
########################## [ 2 a ] ######################################
########################## [ 2 b ] #############################################
```