732A90 Lab 3

2023-11-22

Computational Statistics - Fall 23

Computer Lab 3

Hugo Morvan, Daniele Bozzoli

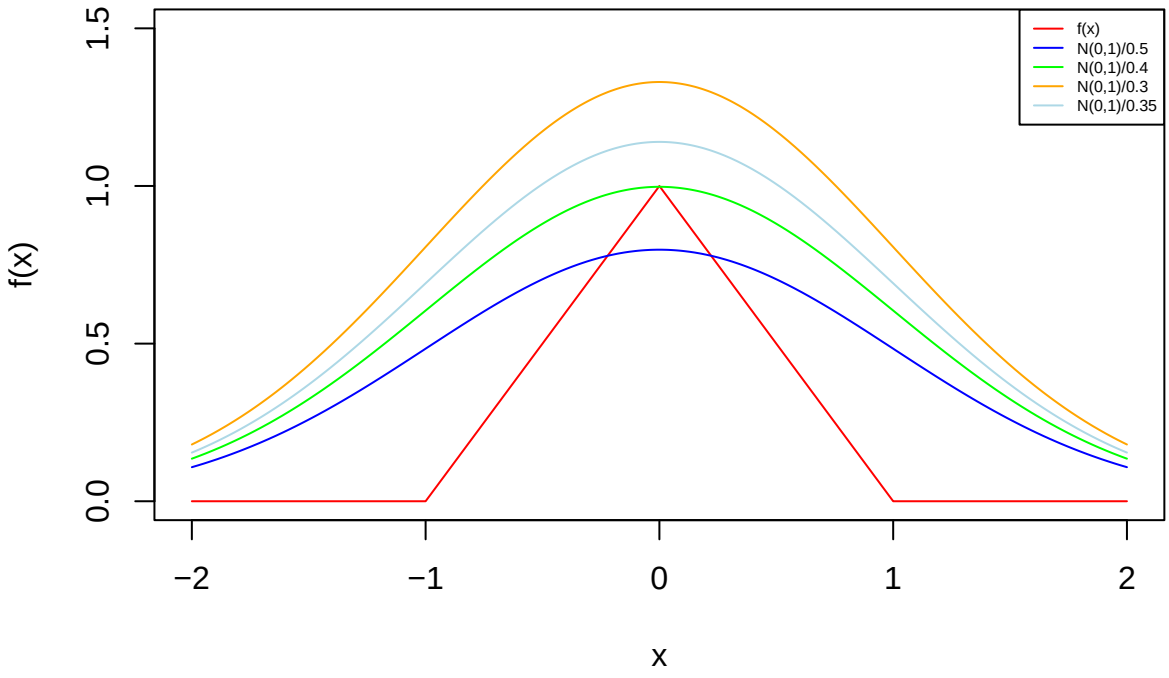
Question 1: Sampling algorithms for a triangle distribution

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

a.

Choose an appropriate and simple envelope e(x) for the density:

```
set.seed(12345)

f <- function(x){
  out <- c()
  for (i in 1:length(x)){
    if (x[i] < -1){out <- append(out, 0)
    }else if(x[i] > 1){ out <- append(out, 0)
    }else if (-1 <= x[i] && x[i] <= 0){out <- append(out, x[i]+1)
    }else if (0 < x[i] && x[i] <= 1){out <- append(out, 1-x[i])}
  }
  return(out)
}

#Determining proper value of a for the envelope
x <- seq(-2, 2, 0.01)
plot(x, f(x), col = "red", type = "l", ylim = c(0, 1.5))
lines(x, dnorm(x)/0.5, col = "blue")
lines(x, dnorm(x)/0.4, col = "green")
lines(x, dnorm(x)/0.3, col = "orange")
lines(x, dnorm(x)/0.35, col = "lightblue")
#add legend
legend("topright", legend = c("f(x)", "N(0,1)/0.5", "N(0,1)/0.4", "N(0,1)/0.3", "N(0,1)/0.35"), col = c
```

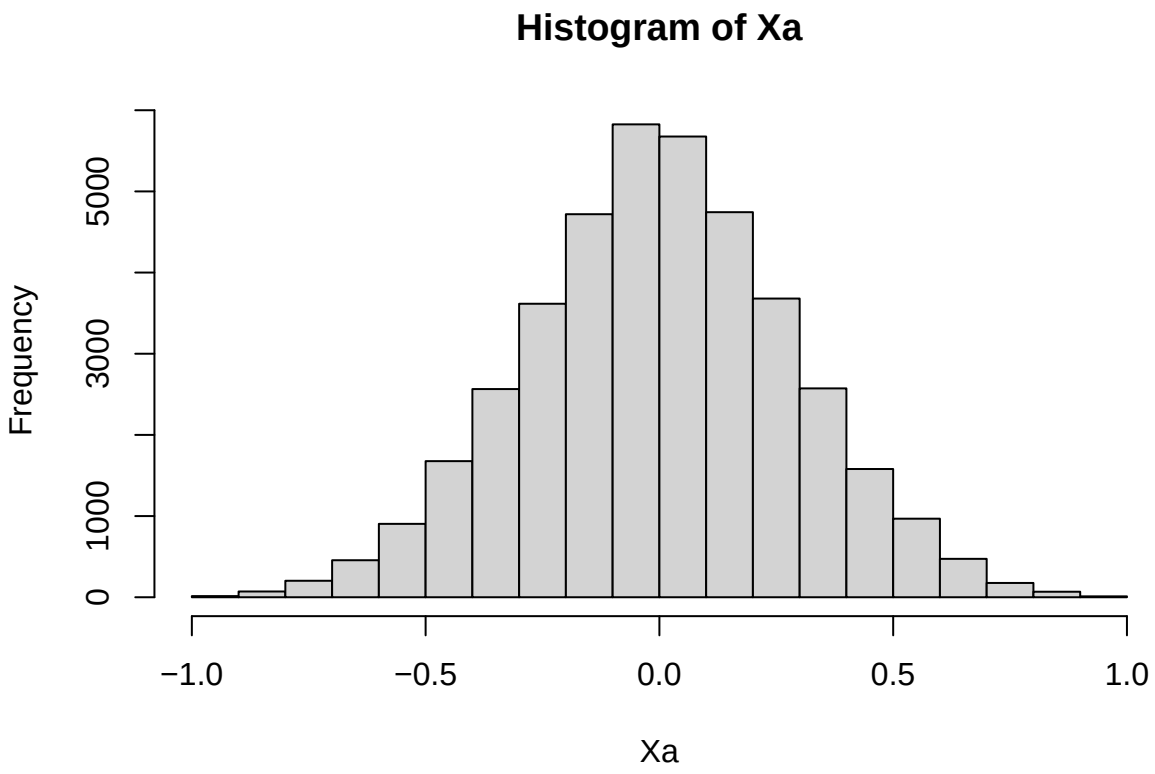



```r
#Based on the three plots, we choose a = 0.35
a <- 0.35

e <- function(x){
  return(dnorm(x) / a)
}
```

We chose a simple normal distribution for the envelope. From this plot it can be seen that a value of a = 0.4 is not quite right therefore a = 0.35 is a good choice for the envelope.

Program a random generator for X using rejection sampling:

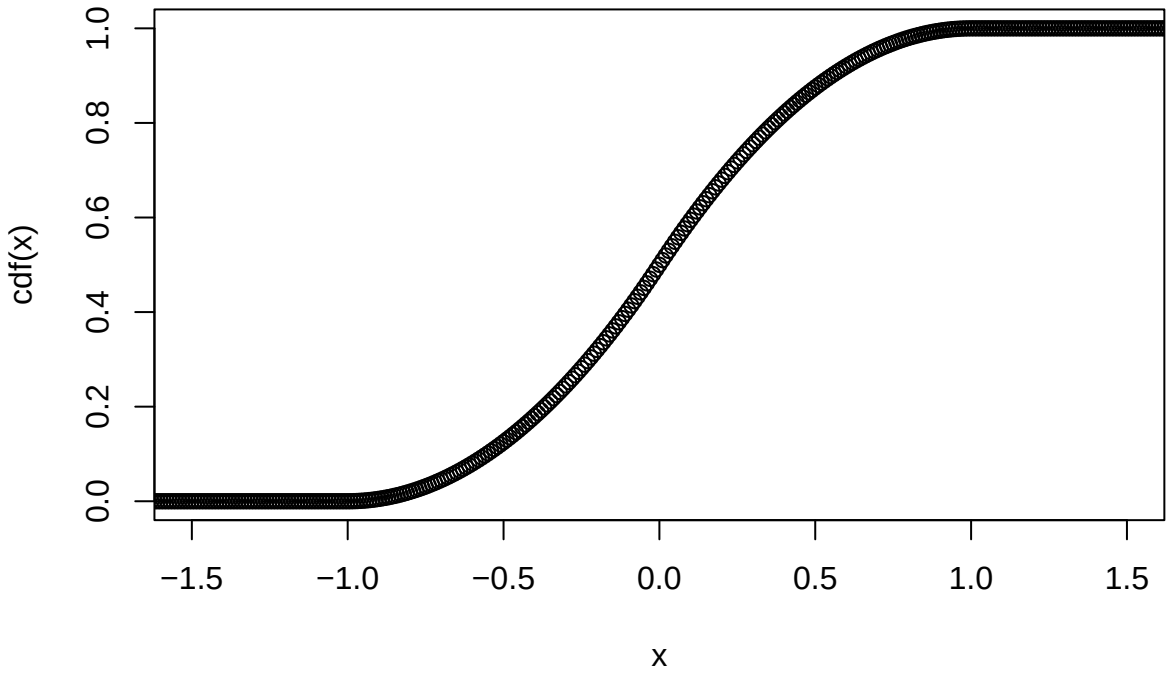
```r
#Rejection sampling:
Xa <- c()

while(length(Xa) < 40000){
   U <- runif(1)
   Y <- rnorm(1) * a            # g(x) = N(0,1)
  if (U <= f(Y)/e(Y)){
    Xa <- c(Xa, Y)
  }
}

#plot the histogram of the generated values
hist(Xa)
```



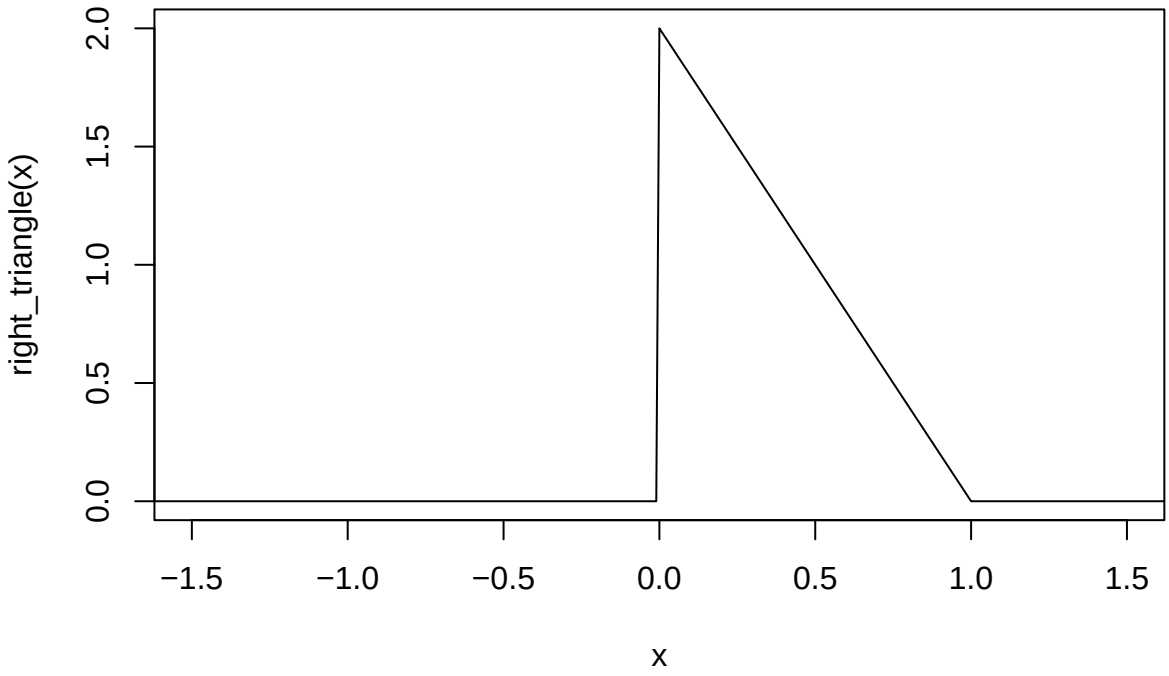
b.

In Lecture 3, another triangle distribution was generated using the inverse cumulative distribution function method, see page 9-10 of the lecture notes. Let Y be a random variable following this distribution. A random variable -Y has a triangle distribution on the interval [-1, 0]. Program a random generator for X using composition sampling based on Y and -Y . You can use the code from the lecture to generate Y .

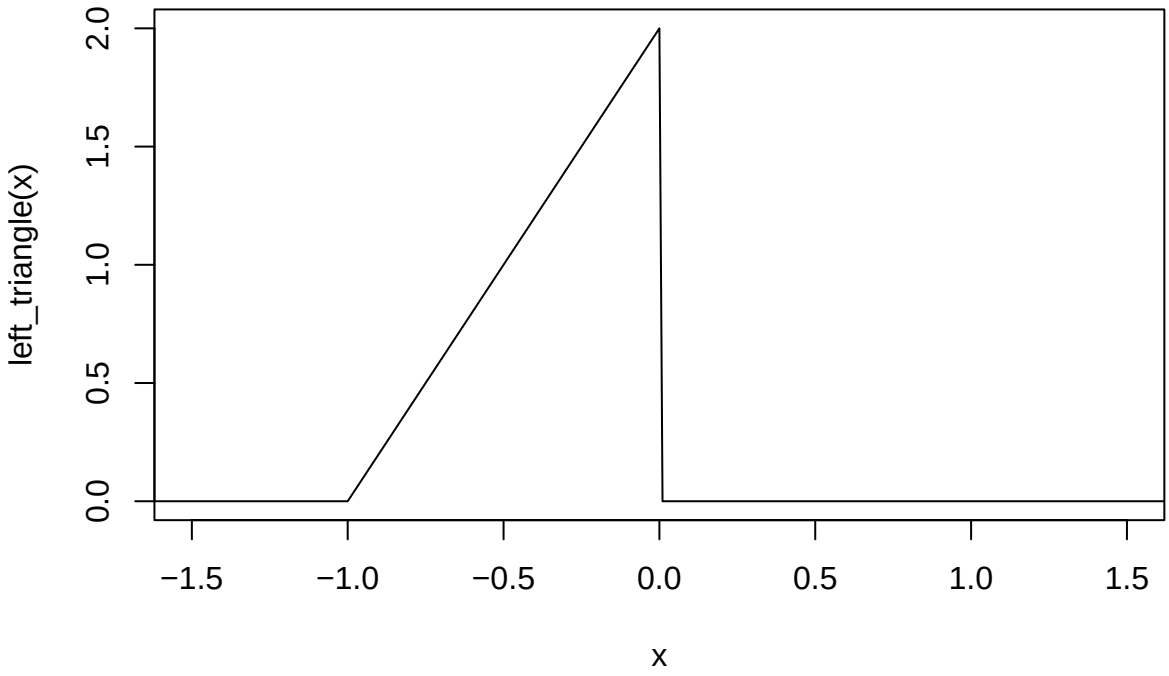
```
x <- seq(-2,2, 0.01)

cdf <- function(x){
  y <- rep(0,length(x))
  for (i in 1:length(x)){
    if (x[i] < -1){
      y[i] <- 0
    }else if (x[i]<=0){
      y[i] <- x[i]^2/2 + x[i] + 0.5
    }else if (x[i]<=1){
      y[i] <- x[i] - (x[i]^2 /2) + .5
    }else y[i] <- 1
  }
  return(y)
}

plot(x,cdf(x), xlim=c(-1.5, 1.5))
```



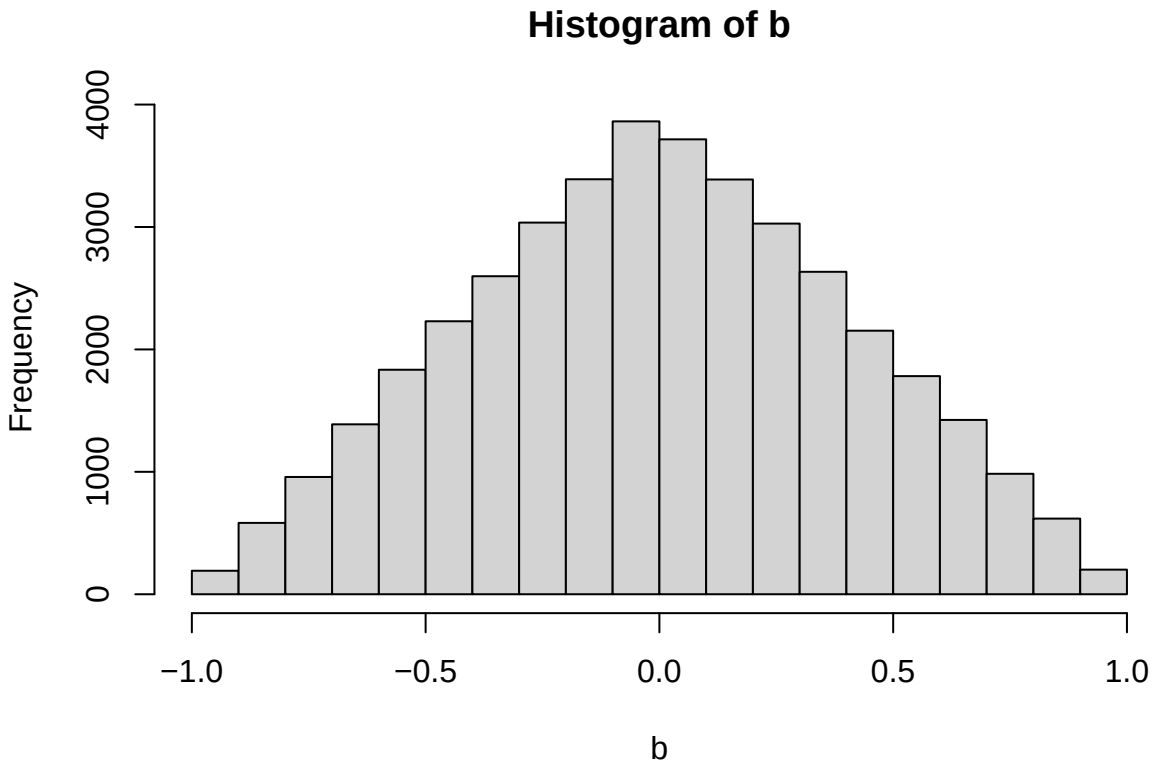

```
right_triangle <- function(x){
  y <- rep(0, length(x))
  for (i in 1:length(x)){
    if (x[i] < 0){
        y[i] <- 0
      }else if(x[i] > 1){
        y[i] <- 0
      }else y[i] <- 2 - 2*x[i]
    }
  return(y)
}

left_triangle <- function(x){
  y <- rep(0, length(x))
  for (i in 1:length(x)){
    if (x[i] < -1){
      y[i] <- 0
    }else if(x[i] > 0){
      y[i] <- 0
    }else y[i] <- 2 + 2*x[i]
  }
  return(y)
}

plot(x,right_triangle(x), xlim=c(-1.5, 1.5), type="l")
```




```
plot(x,left_triangle(x), xlim=c(-1.5, 1.5), type="l")
```



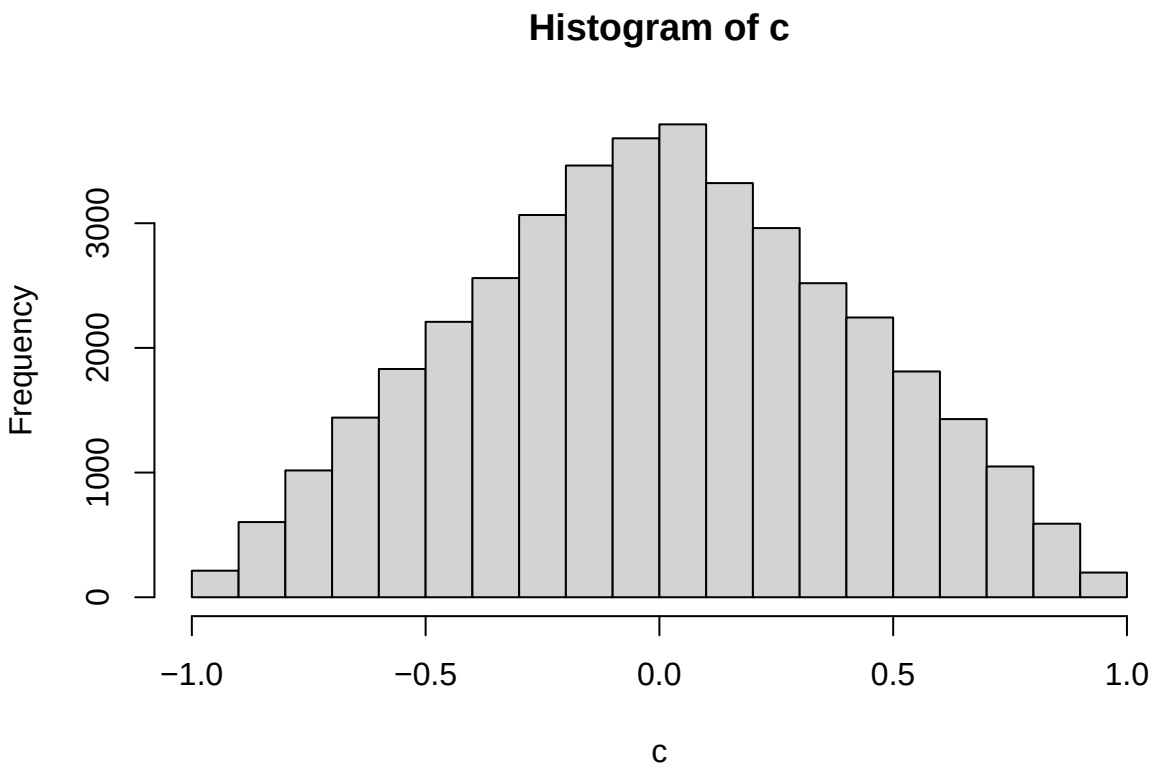

```r
cdf_right <- function(y){
  return(1 - sqrt(1-y))
}

cdf_left <- function(y){
  return(sqrt(y)-1)
}

zerone <- runif(40000, 0,1)
g <- rbinom(40000, 1, 0.5)

genX <- function(x,g){
  res <- rep(0, length(x))
  for (i in 1:length(x)){
    if (g[i] == 0) res[i] <- cdf_right(x[i])
    else res[i] <- cdf_left(x[i])
  }
  return(res)
}
b <- genX(zerone, g)
hist(b)
```



c.

Sums or differences of two independent uniformly distributed variables can also have some triangle distribution. When U1 , U2 are two independent Unif [0, 1] random variables, U1 - U2 has the same distribution as X. Use this result to program a generator for X.

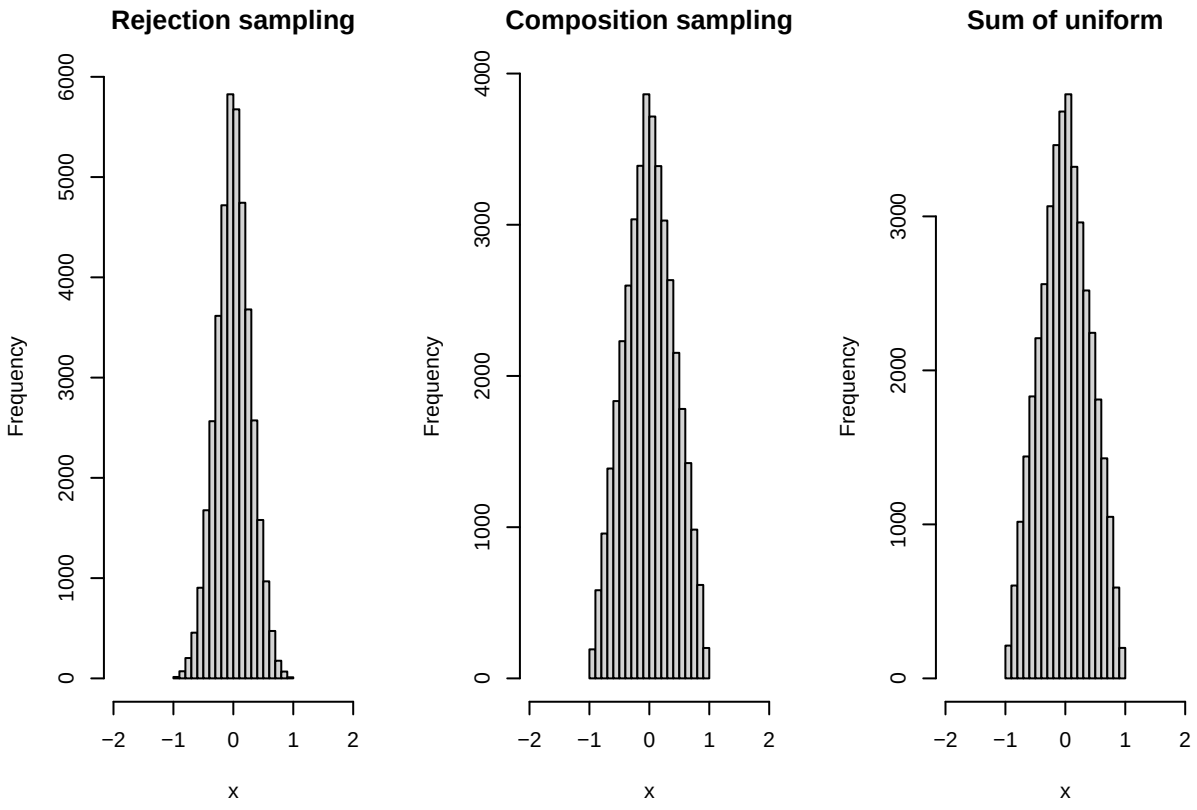
```
genXunif <- function(){
  u1 <- runif(40000)
  u2 <- runif(40000)
  return(u1-u2)
}

c <- genXunif()
hist(c)
```



d.

Check your random generators in each of a. to c. by generating 10000 random variables and plotting a histogram. Which of the three methods do you prefer if you had to generate samples of X? Use the data from one method to determine the variance of X.

```
#Comparaison of the histograms
par(mfrow=c(1,3))
hist(Xa, main="Rejection sampling", xlab="x", xlim=c(-2,2))
hist(b, main="Composition sampling", xlab="x", xlim=c(-2,2))
hist(c, main="Sum of uniform", xlab="x", xlim=c(-2,2))
```



Question 2: Laplace distribution

The double exponential (Laplace) distribution is given by formula:

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

a) Write a code generating double exponential distribution DE(0, 1) from Unif(0,1) by using the inverse CDF method. Explain how you obtained that code step by step. Generate 10000 random numbers from this distribution, plot the histogram and comment whether the result looks reasonable.

The cumulative distribution function of the Laplace distribution is given by:

$$F(x) = \int_{-\infty}^{t} f(t)dt = \begin{cases} \frac{1}{2} exp(\lambda(x - \mu)) & \text{for} \quad x < \mu \\ 1 - \frac{1}{2} exp(-\lambda(x - \mu)) & \text{for} \quad x \geqslant \mu \end{cases}$$

Which can be simplified to the following:

$$\frac{1}{2} + \frac{1}{2} sign(x - \mu)(1 - exp(-\lambda|x - \mu|)$$

where sign(x) is the sign function, which is defined as:

$$sign(x) = \begin{cases} 1 & \text{for} \quad x > 0 \\ 0 & \text{for} \quad x = 0 \\ -1 & \text{for} \quad x < 0 \end{cases}$$

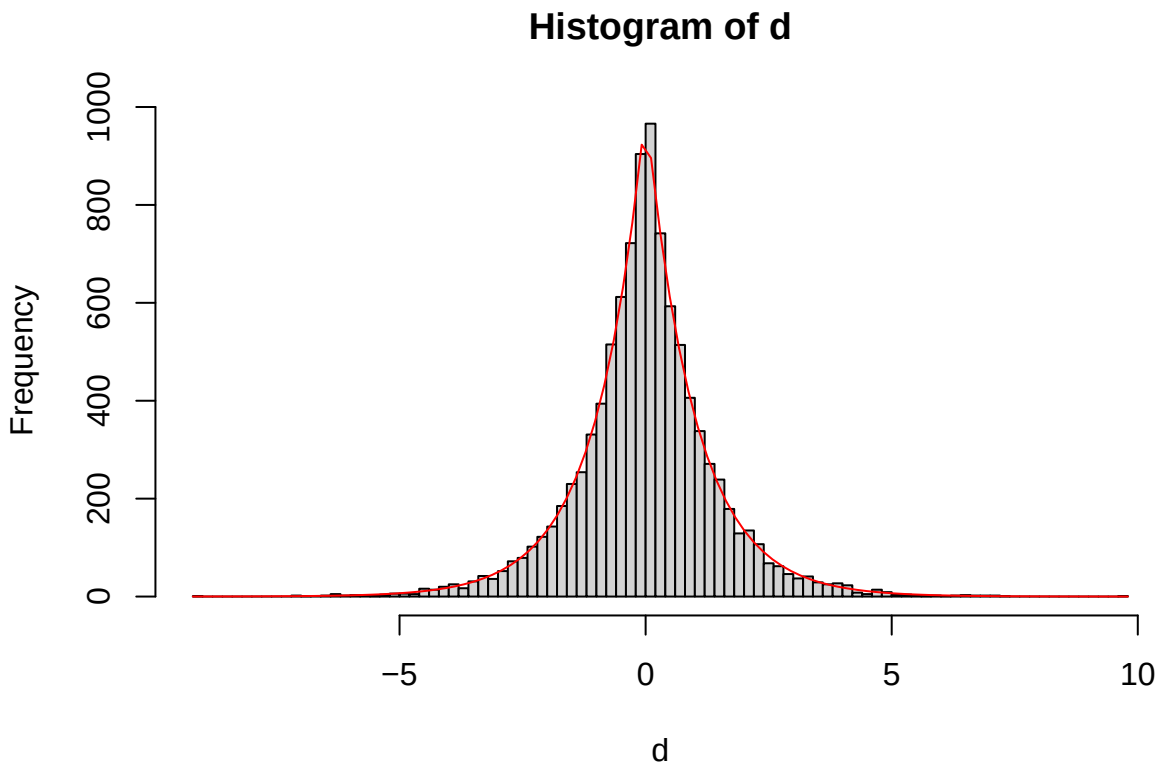
The inverse of the cumulative distribution function is given by:

$$F^{-1}(y) = \mu - \frac{1}{\lambda} sign(y - 0.5) ln(1 - 2|y - 0.5|)$$

(Source : https://en.wikipedia.org/wiki/Laplace_distribution)

This function can be used to generate random numbers from the Laplace distribution. The following code generates 10000 random numbers from the Laplace distribution and plots the histogram:

```
de <- function(x, mu, lambda){
  return(lambda/2 * exp(-lambda * abs(x-mu)))
}
#Inverse CDF method
#Generate 10000 random numbers from Unif(0,1)
set.seed(12345)
u <- runif(10000)
#Generate 10000 random numbers from DE(0,1) using the inverse CDF method
F_inv <- function(p, mu, lambda){
  return(mu - (1/lambda) * sign(p-0.5)*log(1-2*abs(p-0.5)))
}
d <- F_inv(u, 0, 1)
hist(d, breaks = 100)
curve(de(x, 0, 1)*2000, add = TRUE, col = "red")
```



b)

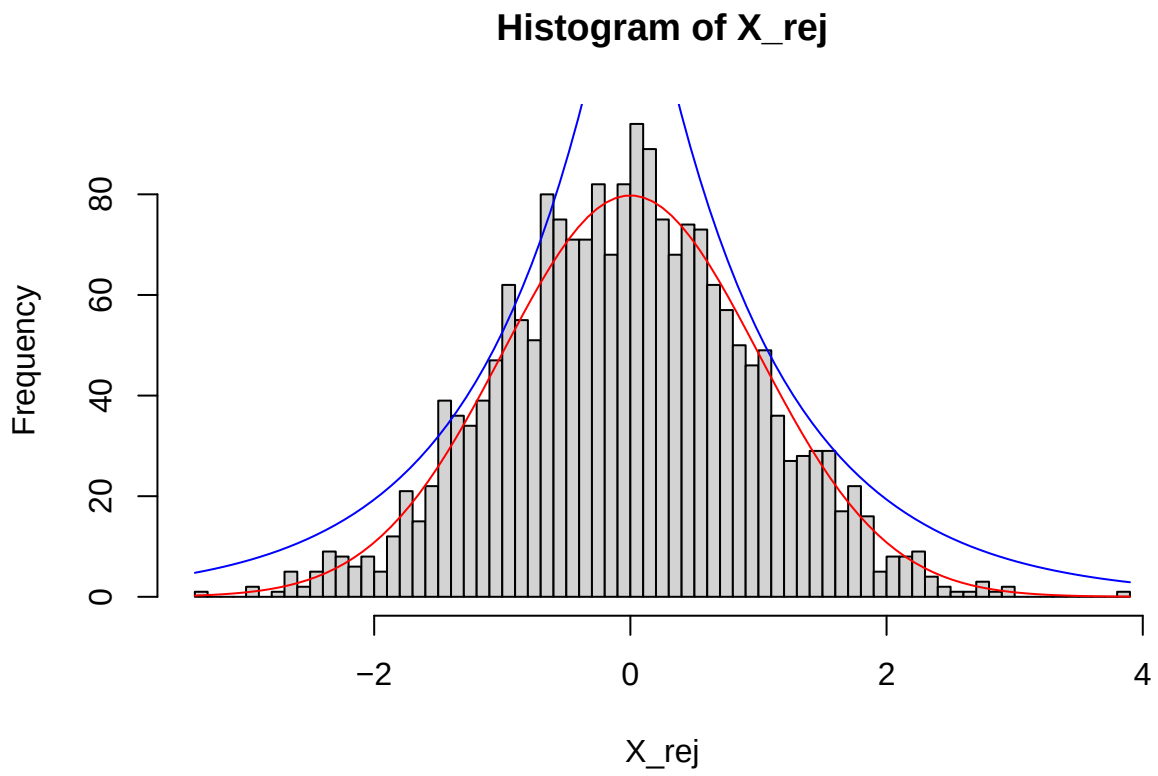
Use rejection sampling with DE (0, 1) as envelope to generate N (0, 1) variables. Explain step by step how this was done. How did you choose constant a in this method? Generate 2000 random numbers N (0, 1) using your code and plot the histogram. Compute the average rejection rate R in the rejection sampling procedure. What is the expected rejection rate ER and how close is it to R?

The rejection sampling method is based on the following theorem: Let f be a probability density function and g be a probability density function such that $f(x) \leqslant ag(x)$ for all x and some constant a. Then the following algorithm generates a random variable X with density f: 1. Generate Y from g 2. Generate U from $U(0,1)$ 3. If $U \leqslant \frac{f(Y)}{ag(Y)}$ then set $X = Y$ otherwise go to step 1.

In this case, f is the standard normal distribution and g is the Laplace distribution. The constant a is chosen such that $f(x) \leqslant ag(x)$ for all x. This is the case when $a = \frac{1}{\sqrt{2\pi}}$. The following code generates 2000 random numbers from the standard normal distribution using the rejection sampling method and plots the histogram:

```
#Rejection sampling method
#Determining proper value of a for the envelope
a <- 0.7
#plot(x, f(x), col = "red", type = "l")
#lines(x, g(x, 0, 1)/a, col = "blue")

#Generate 2000 random numbers from DE(0,1)
set.seed(12345)
u <- runif(2000)
Y <- F_inv(u, 0, 1)*a

f <- function(x){
  return(dnorm(x, 0, 1))
}
g <- function(x, mu, lambda){
  return(de(x, mu, lambda))
}

#After trial and error, a = 0.7 seems to be the best value for a

#Generate 2000 random numbers from N(0,1) using the rejection sampling method
X_rej <- c()
samples <- 0
rejection_rate <- 0
while(length(X_rej) < 2000){
  samples <- samples + 1
  Y <- F_inv(runif(1), 0, 1)
  U <- runif(1, 0,1)
  if(U <= f(Y)/(g(Y, 0, 1)/a)){
    X_rej <- c(X_rej, Y)
  }else{
    rejection_rate <- rejection_rate + 1
  }
}
rejection_rate <- rejection_rate/samples

hist(X_rej, breaks = 100)
curve(dnorm(x, mean = 0, sd = 1)*200, add = TRUE, col = "red")
```

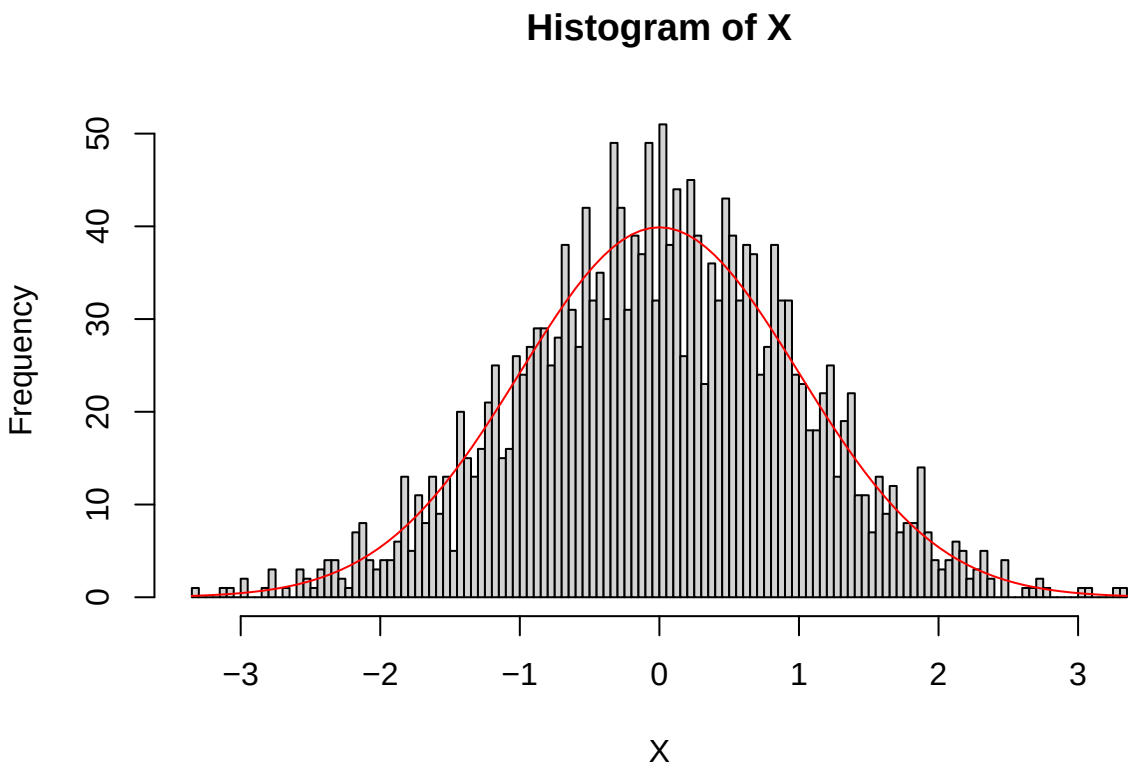
```r
curve(de(x, 0, 1)*200/a, add = TRUE, col = "blue")
```



The rejection rate is 0.2910315.

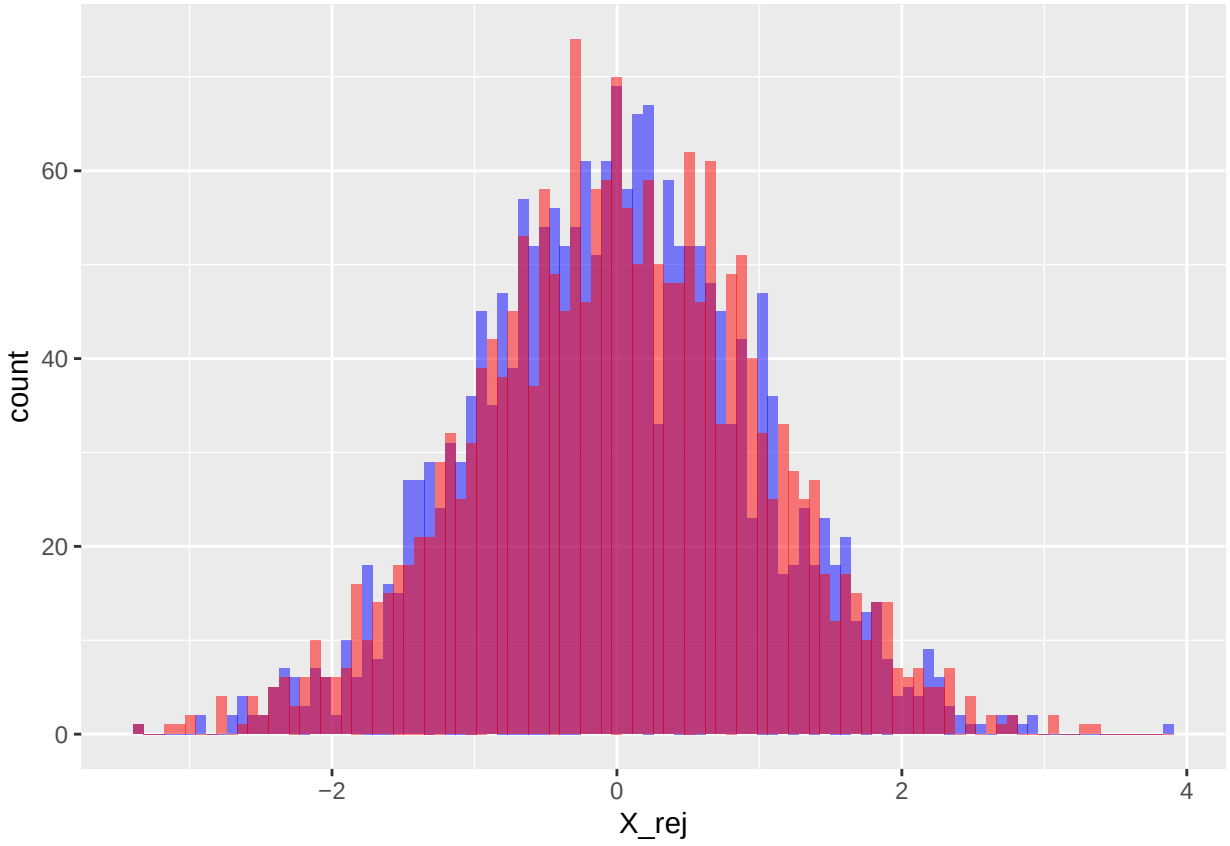
Generate 2000 numbers from N (0, 1) using standard rnorm() procedure, plot the histogram and compare the obtained two histograms.

```r
#Generate 2000 random numbers from N(0,1) using rnorm()
set.seed(12345)
X <- rnorm(2000, mean = 0, sd = 1)
hist(X, breaks = 100)
curve(dnorm(x, mean = 0, sd = 1)*100, add = TRUE, col = "red")
```



Comparaison:

```
library(ggplot2)
df1 <- data.frame(X_rej)
df2 <- data.frame(X)
#Comparaison of the histograms
ggplot(df1, aes(x = X_rej)) + geom_histogram(bins= 100,fill = "blue", alpha = 0.5) + geom_histogram(data
```



The two histograms are very similar.