Lab 1 Group 8

Liuxi Mei Xiaochen Liu

2025-02-10

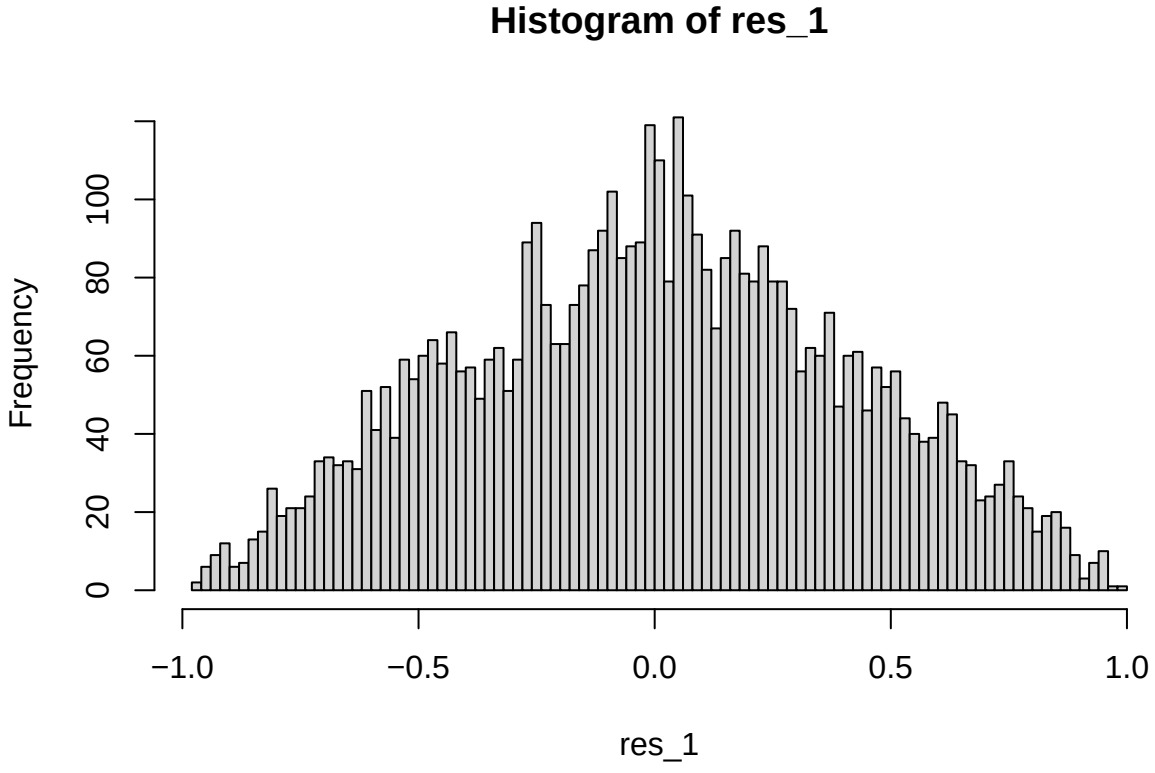
Question 1: Sampling algorithms for a triangle distribution

Consider the following density with a triangle-shape (another triangle distribution than considered in Lecture 3):

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

We are interested to generate draws of a random variable X with this density.

a. Choose an appropriate and simple envelope e(x) for density and program a random generator for X using rejection sampling. Generate 10000 random variables and plot a histogram.



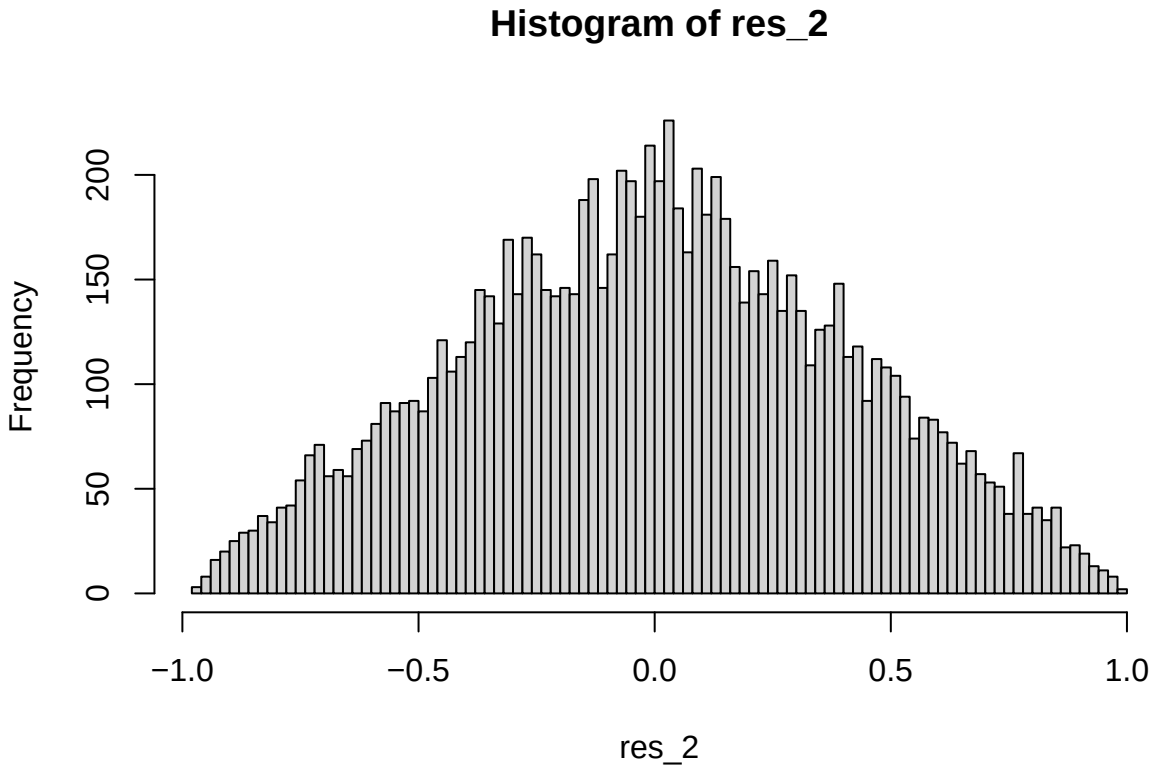
b. In Lecture 3, another triangle distribution was generated using the inverse cumulative distribution function method; see pages 9-10 of the lecture notes. Let Y be a random variable following this distribution. A random variable −Y has a triangle distribution in the interval [−1, 0]. Program a random generator for X using composition sampling based on Y and −Y . You can use the code from the lecture to generate Y . Generate 10000 random variables and plot a histogram.

The cumulative distribution function (CDF) is:

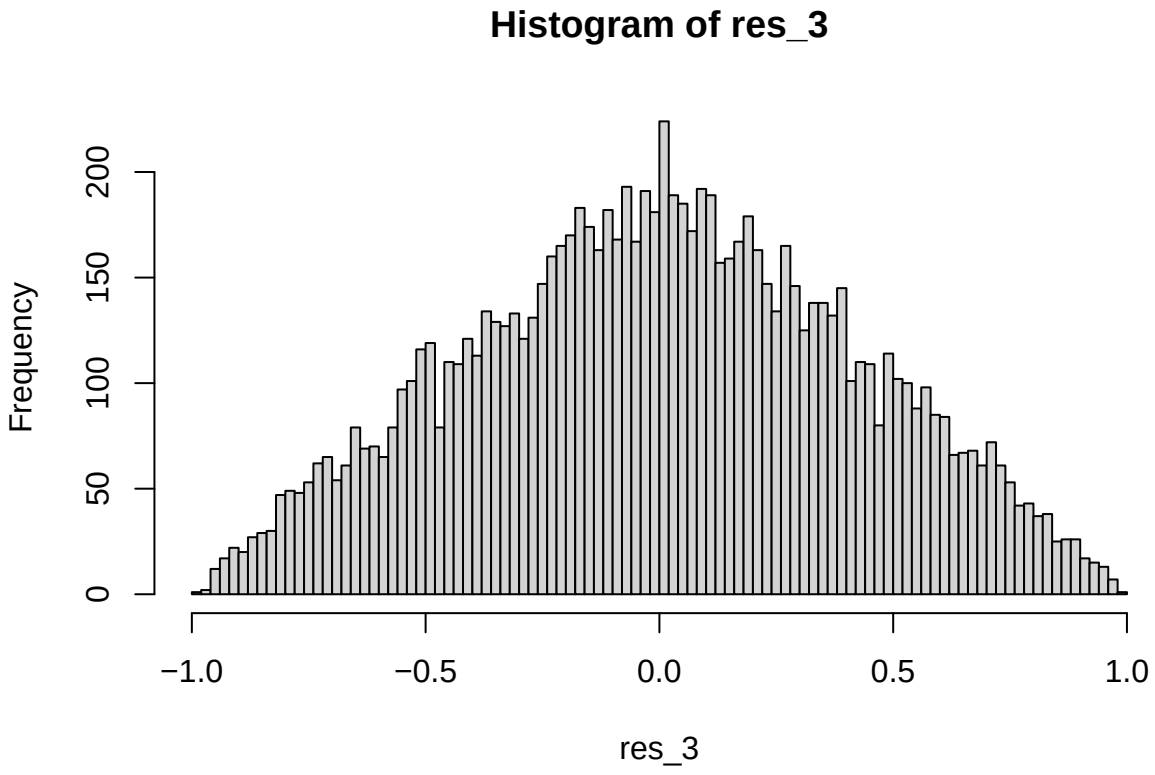
$$F(x) = \begin{cases} 0, & x < -1, \\ \frac{x^2}{2} + x + \frac{1}{2}, & -1 \le x \le 0, \\ \frac{1}{2} + x - \frac{x^2}{2}, & 0 < x \le 1, \\ 1, & x > 1. \end{cases}$$

The inverse cumulative distribution function (quantile function) is:

$$F^{-1}(p) = \begin{cases} -1 + \sqrt{2p}, & 0 \le p \le \frac{1}{2}, \\ 1 - \sqrt{2(1-p)}, & \frac{1}{2} < p \le 1. \end{cases}$$



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]-distributed random variables, U1 − U2 has the same distribution as X. Use this result to program a generator for X. Generate 10000 random variables and plot a histogram.



d. Consider the amount of programming for the three methods and compare the number of random value generation and other time consuming operations needed by the methods to judge expected running time. Which of the three methods do you prefer if you had to generate samples of X? Use data from one method to determine the variance of X.

Method 3 is obviously the easiest and the most preferable regarding to programming difficulty and time consuming. For method 1, a envelope has to be chosen and logical judgement must be done between U and f(Y)/e(Y). And in method 2, the CDF and inverse CDF must be calculated out, which can be time consuming.

```
## [1] 0.1693053
```

##Question 2: Bivariate normal and normal mixture distribution In parts a. and b., we generate random variables with bivariate normal distributions with different methods. Then, we generate bivariate normal mixtures.

a. Generate a two-dimensional random vector $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ that has a two-dimensional normal distribution with mean vector $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and covariance matrix $\begin{pmatrix} 0.6 & 0 \\ 0 & 0.6 \end{pmatrix}$ using the Box-Muller method and using only `runif` as random number generator. Measure the time for generating 10,000,000 random vectors (note: one possibility to measure time is with the function `proc.time()`).

With covariance matrix $\begin{pmatrix} 0.6 & 0 \\ 0 & 0.6 \end{pmatrix}$, we can know that those two vector both have variance as 0.6 and they are independent of each other.

```
## Time used for calculation is  1.9
```

b. Generate again 10,000,000 random vectors with this distribution, but now using either the one-dimensional function `rnorm` or the package `mvtnorm`. Explain why you have chosen to generate it in the way you did. Measure the computation time and compare it to the result in a.

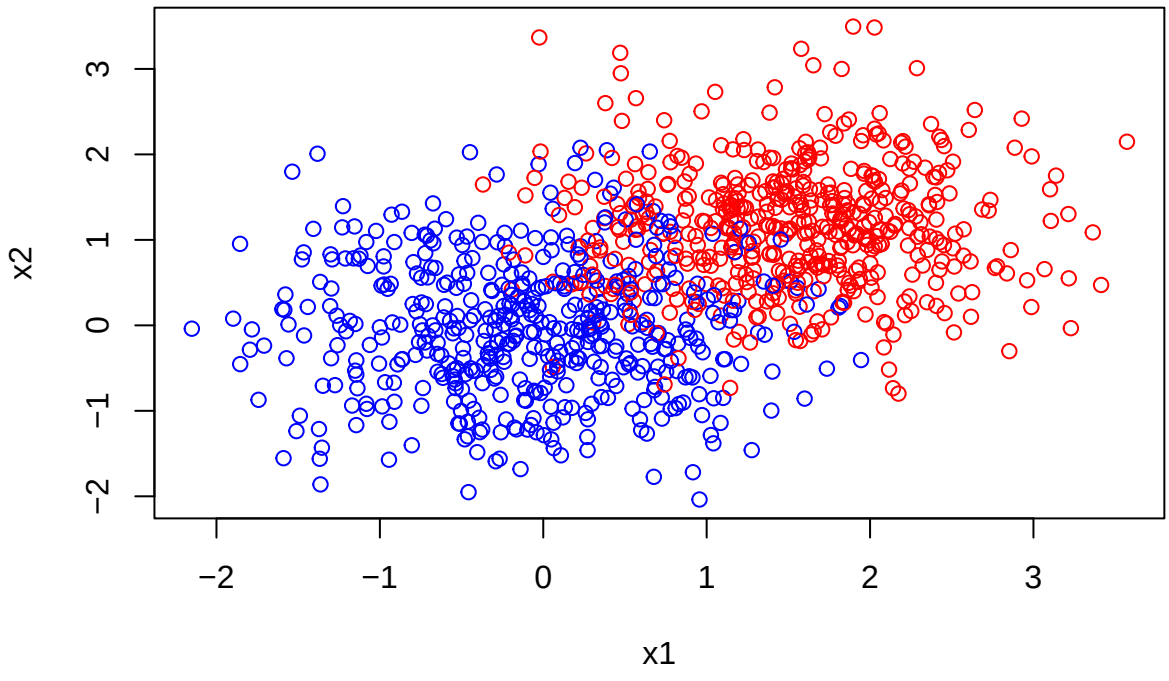
```
## Time used for calculation is  1.01
```

Since the variance is 0.6, the standard devation is sqrt(0.6). No seed is set here since x2 and x3 are independent of each other. As is shown, Box-Muller method is significantly slower than rnorm method.

c. Now, we consider the density of the bivariate normal mixture in Lecture 2. There, observations follow with 50% probability each one of the following two distributions:

- normal distribution with mean vector $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and covariance matrix $\begin{pmatrix} 0.6 & 0 \\ 0 & 0.6 \end{pmatrix}$
- normal distribution with mean vector $\begin{pmatrix} 1.5 \\ 1.2 \end{pmatrix}$ and covariance matrix $\begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}$

Generate 1000 random samples according to that distribution and plot these 1000 samples $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$. Does the plot look satisfactory, and why do you conclude that?



The graph looks satisfactory with two cluster can be celarly seen around (0,0) and (1.5, 1.2), where most data points were generated.

##Appendix

```
# Q1, a)
fx <- function (x) {
  ifelse (x >= -1 & x < 0, x + 1, ifelse (x <= 1,  1 - x, 0))
}

ex <- function (x) {
  ifelse (x >= -1 & x <= 1, 1, 0)
}
set.seed (2025)
u <- runif(10000, -1 , 1)

accept_prob <- fx(u) / ex(u)
uni_rn <- runif(10000,0,1)

res_1 <- u[uni_rn < accept_prob]
hist(res_1, breaks = 100)

# Q1, b)
set.seed (2025)
v <- runif(10000)

part_1 <- 1 - sqrt(2 * (1 - v[v > 0.5]))
part_2 <- -1 + sqrt(2 * v[v < 0.5])
res_2 <- append(part_1, part_2)
hist(res_2, breaks = 100)

#Q1,c)
u1<- runif (10000,0,1)
u2<- runif(10000,0,1)
res_3 <- u1-u2
hist(res_3, breaks = 100)

#Q2, a)
ptm <- proc.time()
u1<- runif(10000000)  # the r of the circle
u2<- runif (10000000)  # the theta
x1<- sqrt(-2* log(u1))*cos(2*pi*u2)*sqrt(0.6)
y1<- sqrt(-2* log(u1))*sin(2*pi*u2)*sqrt(0.6)
time1<- proc.time() - ptm
cat('Time used for calculation is ', time1[3],"\n")

#Q2, b)
ptm <- proc.time()
x2 <- rnorm(10000000,0,sqrt(0.6))
y2 <- rnorm(10000000,0,sqrt(0.6))
time2<- proc.time() - ptm
cat('Time used for calculation is ', time2[3],"\n")

#Q3, c)
g <- rbinom(1000, size = 1, prob = 0.5)

vector_1 <- rnorm(1000, mean=0 , sd = sqrt(0.6))
```

```r
vector_2 <- rnorm(1000, mean = 0, sd = sqrt(0.6))
vector_3 <- rnorm(1000, mean = 1.5, sd = sqrt(0.5))
vector_4<- rnorm(1000, mean = 1.2, sd = sqrt(0.5))
x1 <- ifelse(g==0, vector_1,vector_3)
x2<- ifelse(g==0, vector_2,vector_4)
plot(x=x1,y =x2,col = ifelse(g == 0, "blue", "red"))
```