Lab Report: Lab3 - Computational Statistics

Dhanush Kumar Reddy Narayana Reddy (dhana004), Udaya Shanker Mohanan Nair (udamo524)

2025-02-11

Introduction

Implementation of 2 Assignment questions of Computational Statistics Lab 3 .

Contributions

Member: Dhanush Kumar Reddy Narayana Reddy, Liu Id: dhana004, Contribution: Report writing and coding of question 1.

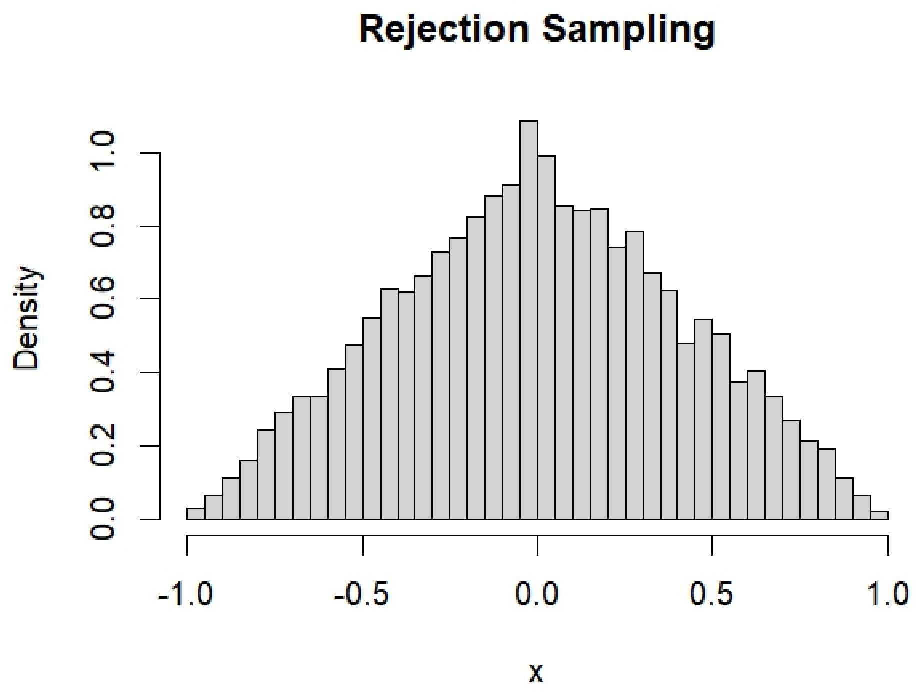
Member: Udaya Shanker Mohanan Nair, Liu Id: udamo524, Contribution: Report writing and coding of question 2.

Question 1

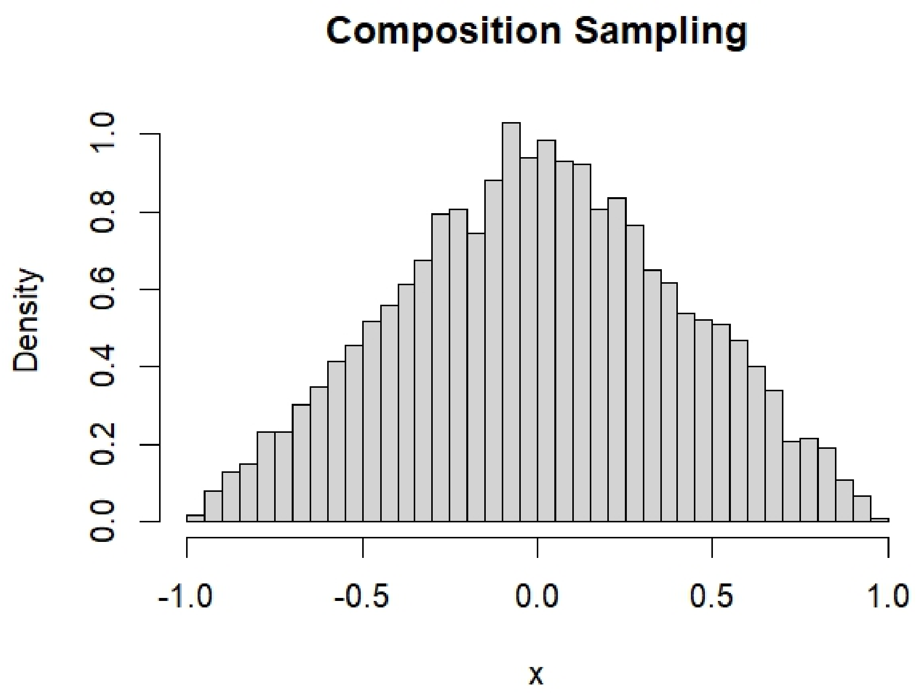
The function f(x) is defined as:

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

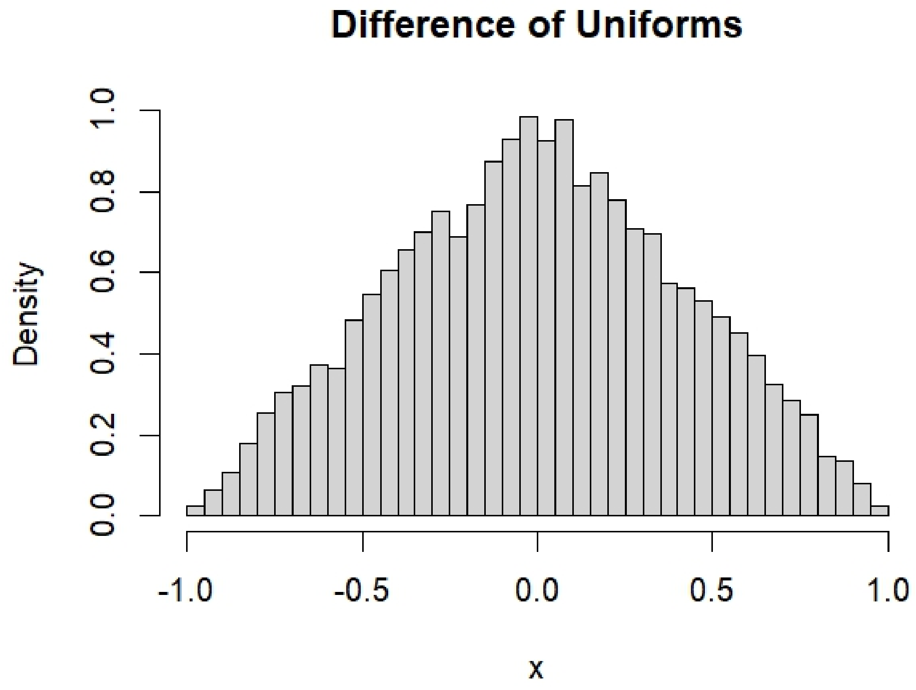
Part A



Part B



Part C



Part D

Comparision:

Rejection Sampling Time:

user system elapsed

0.00 0.00 0.11

Composition Sampling Time:

user system elapsed

0.00 0.00 0.02

Difference of Uniforms Time:

user system elapsed

0 0 0

Choosing the Best Method:

Rejection Sampling is slower compared to the other methods because it involves an accept-reject step. Many proposed samples are discarded, leading to inefficiencies and a longer execution time. However, it is a flexible method that can be applied to more complex distributions where direct sampling is difficult.

Composition Sampling is efficient and provides accurate results since it relies on the inverse cumulative distribution function. However, it requires prior knowledge of the inverse CDF, which may not always be easy to derive for complex distributions. Once the inverse CDF is known, the method is straightforward and computationally efficient.

Difference of Uniforms is the simplest and fastest method since it only requires generating two independent uniform random numbers and taking their difference. It has minimal computational overhead and does not require any accept-reject steps or knowledge of the inverse CDF, making it the preferred choice for efficiency.

The difference of uniforms method is the most efficient and easiest to implement. Based on execution time and simplicity, the difference of uniforms method is the preferred choice for generating samples of X.

Variance for Difference uniform sampling is 0.1682124.

Question 2

part a

In this part, we generated a two dimensional random vector $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ that has 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 Box-Muller method and using runif package of r.

```
## Time measured for generating 10 000 000 random vectors is given below
```

```
##    user  system elapsed
##   0.696   0.092   0.813
```

part b

In this part we gone generate again 10 000 000 random vector. Given two choices either one-dimensional function rnorm or the package mvtnorm. I chose to use mvtnorm because it directly uses mean and variance for generating multivariate normal distributions, which can handle multiple computations inside it more efficiently and give more precise values, removing the chance of potential errors.

```
## Time measured for generating 10 000 000 random vectors is given below
```

```
##    user  system elapsed
##   0.653   0.088   0.751
```

Time comparison between part a and part b is given below

```
##         user system elapsed
## part-a 0.696  0.092   0.813
## part-b 0.653  0.088   0.751
```

part c

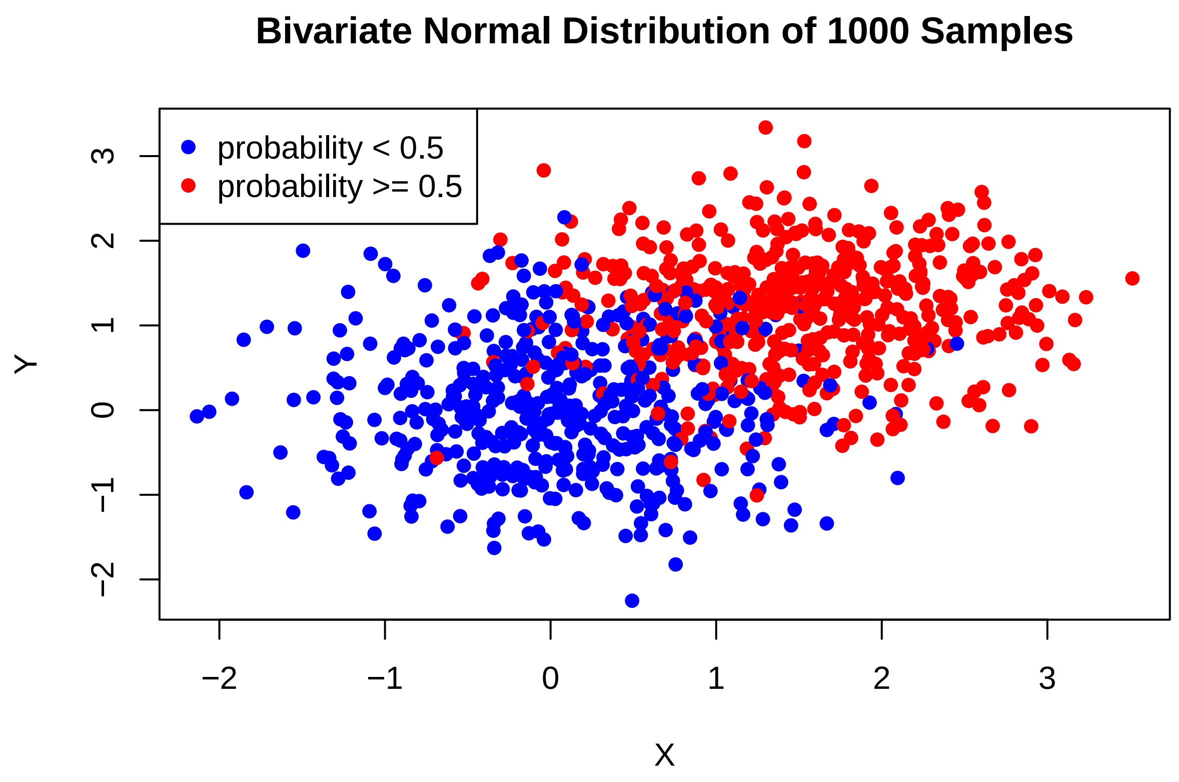
Generated 1000 randoms samples to a two random distributions with 50% probability distribution each for which one distribution is having with mean vector $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and covariance matrix

$$\begin{pmatrix} 0.6 & 0 \\ 0 & 0.6 \end{pmatrix}$$

and the second distribution with with mean vector $\begin{pmatrix} 1.5 \\ 1.2 \end{pmatrix}$ and covariance matrix

$$\begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}$$

Now plotting these 1000 random samples.



The above plot looks satisfactory as it clearly differentiate between two distribution with two different colors. The plot shows less overlapping and more distinguishable clusters, making it easy to identify the data points from two distributions.

Appendix

Question 1 Code

```r
# Question 1: Sampling triangular distribution
# Part A
# Rejection sampling
rejection_sampling <- function(n) {
  f <- function(x) {
    ifelse(x < -1 | x > 1, 0, ifelse(x <= 0, x + 1, 1 - x))
  }
  samples <- numeric(n)
  count <- 0
  while (count < n) {
    x <- runif(1, -1, 1)
    u <- runif(1)
    if (u <= f(x)) {
      count <- count + 1
      samples[count] <- x
    }
  }
  return(samples)
}
# Histogram Plot for Rejection sampling
set.seed(123)
samples_rs <- rejection_sampling(10000)
hist(samples_rs, breaks = 50, main = "Rejection Sampling",
     xlab = "x", freq = FALSE)

# Part B
# Composition sampling
composition_sampling <- function(n) {
  generate_Y <- function(n) {
    u <- runif(n)
    return(1 - sqrt(1 - u))
  }
  samples <- numeric(n)
  for (i in 1:n) {
    if (runif(1) < 0.5) {
      samples[i] <- generate_Y(1)
    } else {
      samples[i] <- -generate_Y(1)
    }
  }
  return(samples)
}
# Histogram Plot for Composition sampling
set.seed(123)
samples_cs <- composition_sampling(10000)
hist(samples_cs, breaks = 50, main = "Composition Sampling",
     xlab = "x", freq = FALSE)

# Part C
# Difference uniform sampling
difference_uniform_sampling <- function(n) {
  u1 <- runif(n)
  u2 <- runif(n)
  x <- u1 - u2
  return(x)
}
# Histogram Plot for Difference uniform sampling
set.seed(123)
samples_dus <- difference_uniform_sampling(10000)
hist(samples_dus, breaks = 50, main = "Difference of Uniforms",
     xlab = "x", freq = FALSE)

# Part D
# Executing time for rejection sampling
time_rs <- system.time(rejection_sampling(10000))
cat("Rejection Sampling Time: \n")
time_rs

# Executing time for composition sampling
time_cs <- system.time(composition_sampling(10000))
cat("Composition Sampling Time: \n")
time_cs

# Executing time for difference of uniforms
time_dus <- system.time(difference_uniform_sampling(10000))
cat("Difference of Uniforms Time: \n")
time_dus

# Comparision
#
# Rejection Sampling Time:
# user  system elapsed
# 0.00    0.00    0.11
#
#
# Composition Sampling Time:
# user  system elapsed
# 0.00    0.00    0.02
#
#
# Difference of Uniforms Time:
# user  system elapsed
# 0       0       0
#
# Choosing the Best Method:
# Rejection Sampling is slower compared to the other methods because
# it involvesan accept-reject step. Many proposed samples are discarded,
# leading to inefficiencies and a longer execution time. However, it is a
# flexible method that can be applied to more complex distributions where
# direct sampling is difficult.
#
# Composition Sampling is efficient and provides accurate results since
# it relies on the inverse cumulative distribution function. However, it
# requires prior knowledge of the inverse CDF, which may not always be easy to
# derive for complex distributions. Once the inverse CDF is known, the method
# is straightforward and computationally efficient.
```

```r
#
# Difference of Uniforms is the simplest and fastest method since it only
# requires generating two independent uniform random numbers and taking their
# difference. It has minimal computational overhead and does not require any
# accept-reject steps or knowledge of the inverse CDF, making it the preferred
# choice for efficiency.
#
# The difference of uniforms method is the most efficient and easiest to
# implement. Based on execution time and simplicity, the difference of uniforms
# method is the preferred choice for generating samples of X.

# Variance for Difference uniform sampling
variance_dus <- var(samples_dus)
variance_dus
# [1] 0.1682124
```

Question 2 Code

```r
library(MASS)
library(mvtnorm)

mean1 <- c(0, 0)
cov1 <- matrix(c(0.6, 0, 0, 0.6), nrow=2)

bivariate_function <- function(mean, variance) {
  set.seed(12345678)
  n1 <- runif(10000000)
  n2 <- runif(10000000)
  g1 <- sqrt(-2 * log(n1)) * cos(2 * pi * n2)
  g2 <- sqrt(-2 * log(n1)) * sin(2 * pi * n2)
  # response <- t(mean + t(chol(variance)) %*% rbind(g1, g2))
  X1 <- sqrt(0.6) * g1
  X2 <- sqrt(0.6) * g2
}

time1 <- proc.time()
response1 <- bivariate_function(mean1, cov1)
time2 <- proc.time()
time_taken1 <- time2 - time1

cat("Time measured for generating 10 000 000 random vectors is given below")
print(time_taken1)

set.seed(123)
n <- 10000000
time3 <- proc.time()
response2 <- rmvnorm(n, mean = mean1, sigma = cov1)
time4 <- proc.time()
time_taken2 <- time4 - time3
cat("Time measured for generating 10 000 000 random vectors is given below")
print(time_taken2)
```

```r
comp_mat <- matrix(c(time_taken1[1],time_taken1[2],time_taken1[3],time_taken2[1],time_taken2[2],time_ta
colnames(comp_mat) <- c("user","system","elapsed")
rownames(comp_mat) <- c("part-a","part-b")
print(comp_mat)

set.seed(1234)
c_n <- 1000
p<- runif(c_n)
c_mean1 <- c(0, 0)
c_variance1 <- matrix(c(0.6, 0, 0, 0.6), nrow = 2)
c_mean2 <- c(1.5, 1.2)
c_variance2 <- matrix(c(0.5, 0, 0, 0.5), nrow = 2)

nd <- matrix(0, nrow = c_n, ncol = 2)
for (i in 1:c_n) {
  if (p[i] < 0.5) {
    nd[i, ] <- rmvnorm(1, mean = c_mean1, sigma = c_variance1)
  } else {
    nd[i, ] <- rmvnorm(1, mean = c_mean2, sigma = c_variance2)
  }
}

colors <- ifelse(p < 0.5, "blue", "red")
plot(nd, col = colors, pch = 16, xlab = "X", ylab = "Y",
     main = "Bivariate Normal Distribution of 1000 Samples")
legend("topleft", legend = c("probability < 0.5", "probability   0.5"),
       col = c("blue", "red"), pch = 16)
```