Monte Carlo Project Part 3

Wardakhan Kévin, Erwan Ouabdesselam, Matteo Casati

2024-12-24

Definition

Question 1:

We want to ensure that f is positive for all $x \in \mathbb{R}$. i.e, $f_1(x) \geq a f_2(x)$ for all $x \in \mathbb{R}$.

$$\frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right) \geq a \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{(x-\mu_2)^2}{2\sigma_2^2}\right)$$

This simplifies to:

$$\exp\left(-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right) \geq a \frac{\sigma_1^2}{\sigma_2^2} \exp\left(-\frac{(x-\mu_2)^2}{2\sigma_2^2}\right)$$

Taking the natural logarithm on both sides, we get:

$$-\frac{(x-\mu_1)^2}{2\sigma_1^2} \geq \ln\left(a\frac{\sigma_1^2}{\sigma_2^2}\right) - \frac{(x-\mu_2)^2}{2\sigma_2^2}$$

Expanding the quadratic terms:

$$-\frac{x^2 - 2\mu_1 x + \mu_1^2}{2\sigma_1^2} \geq \ln\left(a\frac{\sigma_1^2}{\sigma_2^2}\right) - \frac{x^2 - 2\mu_2 x + \mu_2^2}{2\sigma_2^2}$$

Simplifying this expression:

$$\left(\frac{1}{2\sigma_2^2} - \frac{1}{2\sigma_1^2}\right) x^2 + \left(\frac{\mu_2}{\sigma_2^2} - \frac{\mu_1}{\sigma_1^2}\right) x + \left(\frac{\mu_1^2}{2\sigma_1^2} - \frac{\mu_2^2}{2\sigma_2^2}\right) - \ln\left(a\frac{\sigma_1^2}{\sigma_2^2}\right) \geq 0$$

For this inequality to hold for all $x \in \mathbb{R}$, and more specifically for x large enough, the term in x^2 must be positive, otherwise $f(x)$ could become negative for large values of x.

Thus, this implies:

$$\frac{1}{2\sigma_2^2} - \frac{1}{2\sigma_1^2} > 0$$

To ensure that this expression is non-positive, we must have:

$$\frac{1}{2\sigma_2^2} < \frac{1}{2\sigma_1^2} \implies \sigma_2^2 < \sigma_1^2$$

This gives us a necessary condition on (σ_1^2,σ_2^2) considering the tail behavior of f.

Question 2:

Moreover, the integral of f must be equal to 1. Hence, we add a normalization factor $c \in \mathbb{R}$, then the density $f(x)$ is defined as:

$$f(x) = c\left(f_1(x) - af_2(x)\right).$$

For the integral to be equal to 1, we must have:

$$\int_{-\infty}^{\infty} f(x)\,dx = c\left(\int_{-\infty}^{\infty} f_1(x)\,dx - a\int_{-\infty}^{\infty} f_2(x)\,dx\right).$$

This simplifies to:

$$\int_{-\infty}^{\infty} f(x)\,dx = c(1-a).$$

To ensure that $\int_{-\infty}^{\infty} f(x)\,dx = 1$, we must have:

$$c(1-a) = 1.$$

Thus, the normalization factor c is given by:

$$c = \frac{1}{1-a}.$$

Conclusion: The condition for $f(x)$ to be a probability density (i.e., for the integral to equal 1) is that the density is defined as:

$$f(x) = \frac{1}{1-a}\left(f_1(x) - af_2(x)\right),$$

where the normalization factor $c = \frac{1}{1-a}$ ensures that:

$$\int_{-\infty}^{\infty} f(x)\,dx = 1.$$

We also want f to be positive but this time for all $x \in \mathbb{R}$
.

$$\frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right) \geq a\frac{1}{\sqrt{2\pi\sigma_2^2}}\exp\left(-\frac{(x-\mu_2)^2}{2\sigma_2^2}\right)$$

This simplifies to:

$$a \leq \frac{\sigma_2}{\sigma_1}\frac{\exp\left(-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right)}{\exp\left(-\frac{(x-\mu_2)^2}{2\sigma_2^2}\right)}$$

We want to minimize the left member with respect to x, to find an upper bound for a, which will be denoted a^*. We introduce the function $g(x)$, which corresponds to the right-hand side without the quotient of $\frac{\sigma_2}{\sigma_1}$. Next, we differentiate g and find that the derivative vanishes only for:

$$x^* = \frac{\mu_1\sigma_2^2 - \mu_2\sigma_1^2}{\sigma_1^2 - \sigma_2^2}.$$

We know that this is a minimizer of g because g' is decreasing on $(-\infty, x^*)$ and increasing on $(x^*, +\infty)$. Thus, we define:

$$a^* = \frac{\sigma_2}{\sigma_1} g(x^*) = \frac{\sigma_2}{\sigma_1} \exp\left(-\frac{(\mu_1 - \mu_2)^2}{2(\sigma_1^2 - \sigma_2^2)}\right).$$

Hence, $0 < a \leq a^*$ is a sufficient condition for f to be a density function.

Question 3:

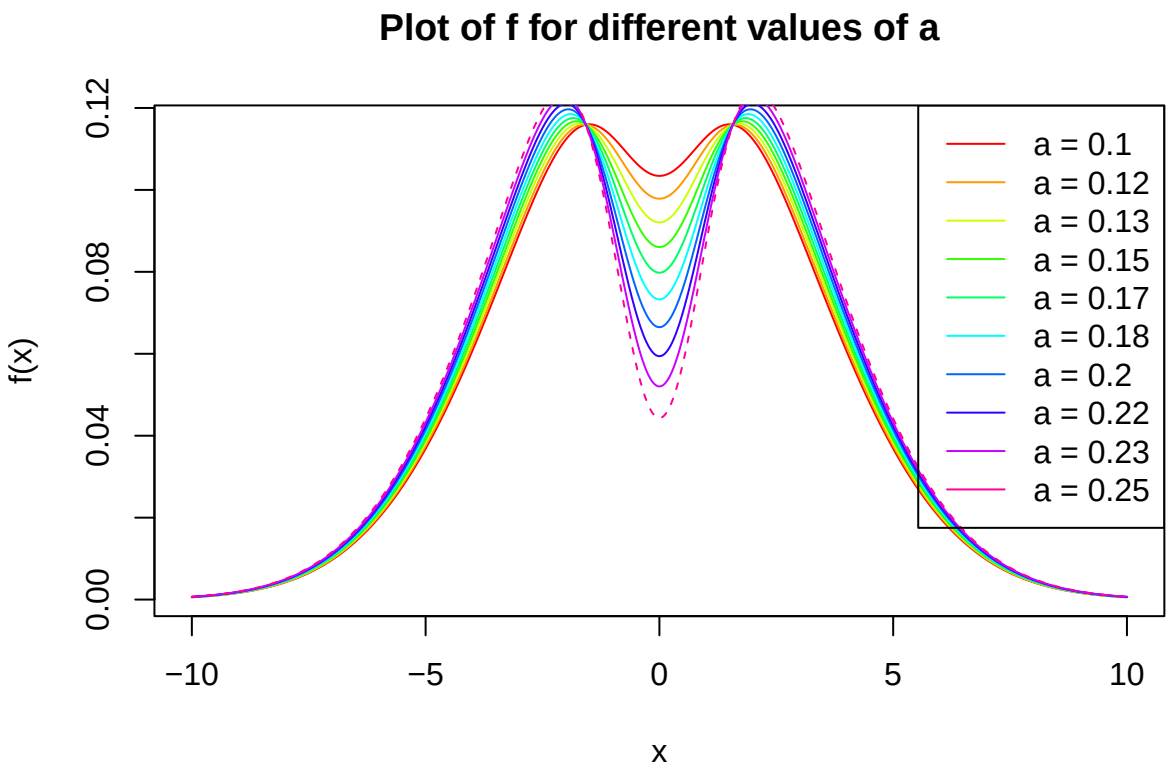
We plot the function f for different values of a. The pink dotted curve corresponds to $a = a^*$.

```
#plot of f for different values of a
#The dotted curve corresponds to a=a_star

a_values <- seq(0.1, a_star, length.out = 10)
colors_a <- rainbow(length(a_values))
x <- seq(-10, 10, length.out = 1000)

plot(x, f(a_values[1], mu_1, mu_2, 3, s_2, x), type = "l",
col = colors_a[1], xlab = "x", ylab = "f(x)",
main = "Plot of f for different values of a")

for (i in 2:length(a_values)) {
  if(a_values[i]==a_star){
    lines(x, f(a_values[i], mu_1, mu_2, 3, s_2, x), col = colors_a[i], lty = 2)
  }
  else{
    lines(x, f(a_values[i], mu_1, mu_2, 3, s_2, x), col = colors_a[i])
  }
}
legend("topright", legend = paste("a =", round(a_values, 2)), col = colors_a, lty = 1)
```



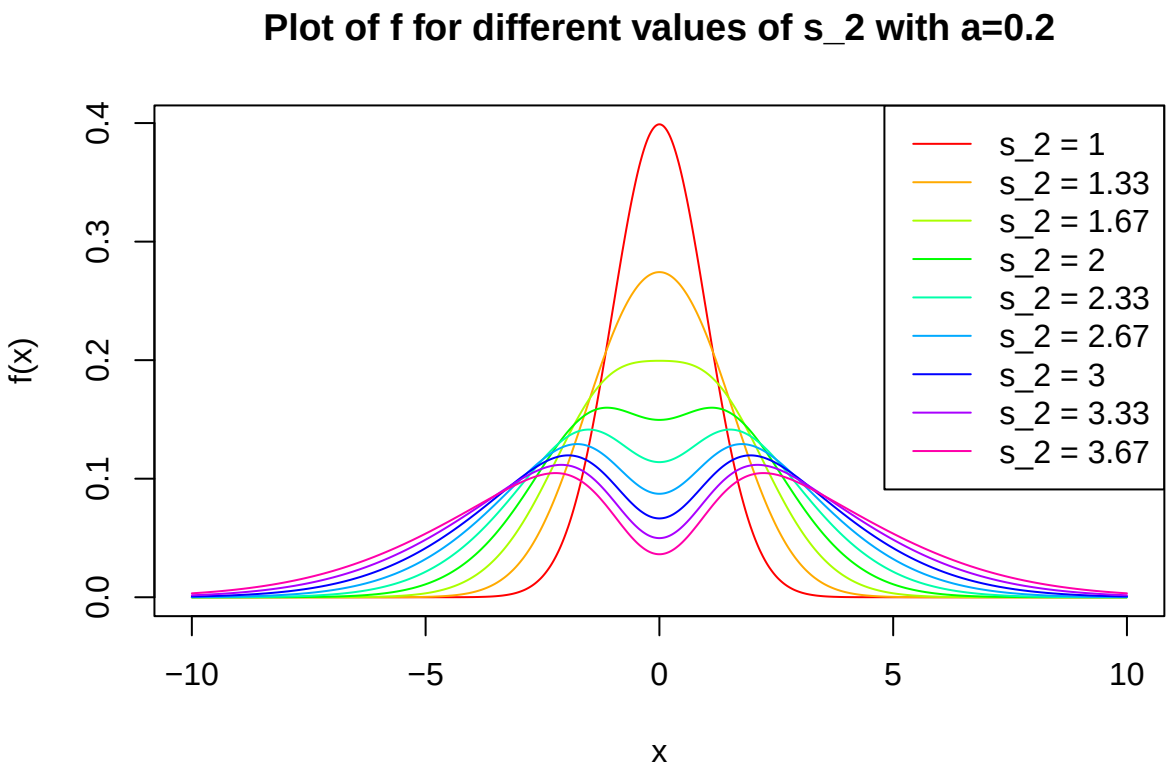
We remark on the previous plot that the more a is close to a^*, the more the curve forms a second slope, which comes from an increase of the weight of f_2 in the function f.

We now plot the function f for s_1 fixed and s_2 varying between 0 (excluded) and s_1.

```
#plot of f for different values of s_2
a=0.2
s_2_values <- seq(1, s_1, length.out = 10)
s_2_values<-s_2_values[1:length(s_2_values)-1]
#We remove the last value because it is equal to s_1 and we need s_1>s_2
colors_s2 <- rainbow(length(s_2_values))

plot(x, f(a, mu_1, mu_2, s_2_values[1], s_2, x), type = "l", col = colors_s2[1],
xlab = "x", ylab = "f(x)", main = "Plot of f for different values of s_2 with a=0.2")

for (i in 2:length(s_2_values)) {
  lines(x, f(a, mu_1, mu_2, s_2_values[i], s_2, x), col = colors_s2[i])
}
legend("topright", legend = paste("s_2 =", round(s_2_values,2)), col = colors_s2, lty = 1)
```



We let from now on $\mu_1 = 0$, $\mu_2 = 1$, $\sigma_1 = 3$, $\sigma_2 = 1$, and $a = 0.2$.

Inverse c.d.f Random Variable simulation

Question 4:

To approach this question, we proceeded as follows:

Step 1: we defined an R function `F` to compute the cumulative distribution function $F(x)$ using the CDFs of the associated normals f_1 and f_2. The CDF is given by

$$F(x) = F_1(x) - a \cdot F_2(x),$$

where F_1 and F_2 are the CDFs of the normal distributions of f_1 and f_2. This closed-form expression enables easy calculation of $F(x)$ for different points.

Step 2: We then implemented an algorithm that uses the inverse CDF method to generate samples from $f(x)$. The process involves:

- Generating uniform values u in $(0, 1)$,
- Using linear interpolation to approximate the inverse of the CDF, which allows us to transform each u into a value x following the distribution $f(x)$.

Step 3: We used interpolation to invert the CDF. After constructing a table of values for $F(x)$, we interpolated x-values to match uniform values u. This method ensures good accuracy in sampling. We first tried to use the dichotomic method, but we encountered many problems. Therefore, we chose another approach to tackle the problem.

```r
# Function CDF F(x)
F <- function(x, a, mu_1, mu_2, s_1, s_2) {
  F1 <- pnorm(x, mean = mu_1, sd = s_1)  # CDF for the first normale
  F2 <- pnorm(x, mean = mu_2, sd = s_2)  # CDF for the second normale
  return(F1 - a * F2/(1-a))
}

# Step 1: Compute the CDF over a grid of values.
create_cdf_table <- function(a, mu_1, mu_2, s_1, s_2, grid_size = 1000) {
  x_vals <- seq(-10, 10, length.out = grid_size)  # Grille of x
  cdf_vals <- F(x_vals, a, mu_1, mu_2, s_1, s_2)  # Values of the CDF
  return(list(x = x_vals, cdf = cdf_vals))  # Return a table of CDF
}

# Step 2: Approximate the inverse of the CDF using interpolation.
inv_cdf_approx <- function(u, cdf_table) {
  # Interpolation of the inverse of the CDF
  return(approx(cdf_table$cdf, cdf_table$x, xout = u)$y)
}

# Step 3: Generate random variables using the approximated inverse of the CDF.
inv_cdf<- function(n, a, mu_1, mu_2, s_1, s_2) {
  cdf_table <- create_cdf_table(a, mu_1, mu_2, s_1, s_2)  # Create the table of CDF
  u_vals <- runif(n)  # generate n uniform values in (0, 1)
  samples <- sapply(u_vals, inv_cdf_approx, cdf_table = cdf_table)  # Generate the samples
  return(samples)
}
```

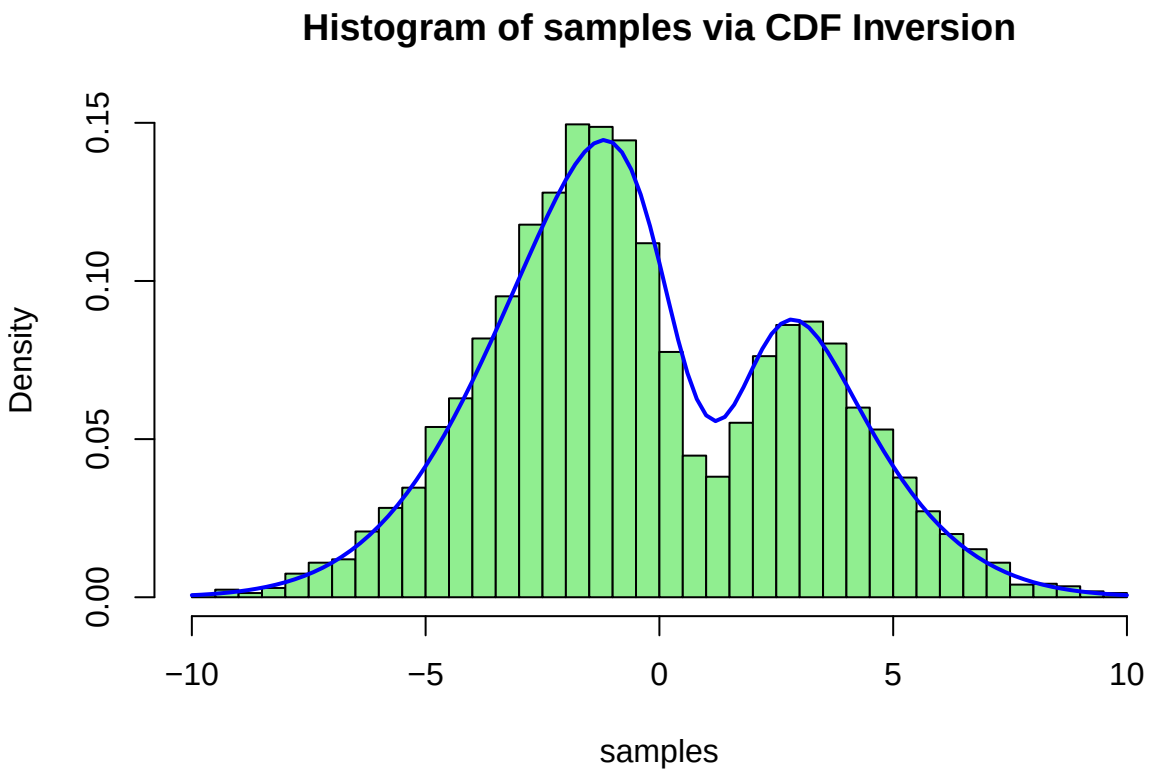
Question 5:

We check here if our estimation of the inverse of the CDF is correct by plotting the histogram of the samples generated by the function *inv_cdf()* and superposing the theoretical density function.

```r
# Generate 10,000 samples
n <- 10000
samples <- inv_cdf(n, a, mu_1, mu_2, s_1, s_2)

# Visualize the histogram of the samples
hist(samples, probability = TRUE, breaks = 50,
     main = "Histogram of samples via CDF Inversion", col = "lightgreen")

# Superimpose the theoretical density function
curve(f(a, mu_1, mu_2, s_1, s_2, x), add = TRUE,
      col = "blue", lwd = 2)
```



Verification of Accuracy: The alignment between the histogram of samples and the theoretical density curve confirms that the inverse CDF method correctly generates samples from f(x).

Accept-Reject Random Variable simulation

Question 6:

In order to simulate under $f(x) := \frac{1}{1-a}(f_1(x) - af_2(x))$ via the accept reject method we have to find a density $g(x)$ such that:

$$Mg(x) \geq f(x) \ \ \forall x \in \mathbb{R}$$

where M is a constant. If we set $g(x) = f_1(x)$ as our proposal distribution we have to now find a value for M such that the previous equation is satisfied, that is:

$$Mf_1(x) \geq \frac{1}{1-a}(f_1(x) - af_2(x))$$

We also know that:

$$\frac{1}{1-a}f_1(x) \geq \frac{1}{1-a}(f_1(x) - af_2(x))$$

thanks to non-negativity property of density function f_2. Thus, it suffices to find a value for M that satisfies

$$Mf_1(x) \geq \frac{1}{1-a}f_1(x) \implies M \geq \frac{1}{1-a}$$

It follows that $M = \frac{1}{1-a}$After having derived the value of M we can approximate the theoretical acceptance rate:

$$acc\_rate := \frac{1}{M} = 1 - a$$

Therefore the higher the impact of f_2 on the distribution function f, the lower the acceptance rate. The algorithm proceeds as follows:

- Generate values $Y \sim f_1$ and $\mathcal{U} \sim U(0,1)$.
- Accept $X = Y$ if $\mathcal{U} \leq \frac{f(Y)}{Mf_1(Y)} = \frac{(1-a)f(Y)}{f_1(Y)}$.

Question 7:

We set $g(x) = f_1(x)$ as the proposal distribution, as it generally covers $f(x)$ well.In order to simulate under $f(x)$, we must find a value for M such that $Mg(x) \geq f(x)$ for all $x \in \mathbb{R}$. In the previous question, we thus found that $Mg(x) = \frac{1}{1-a} f_1(x)$.

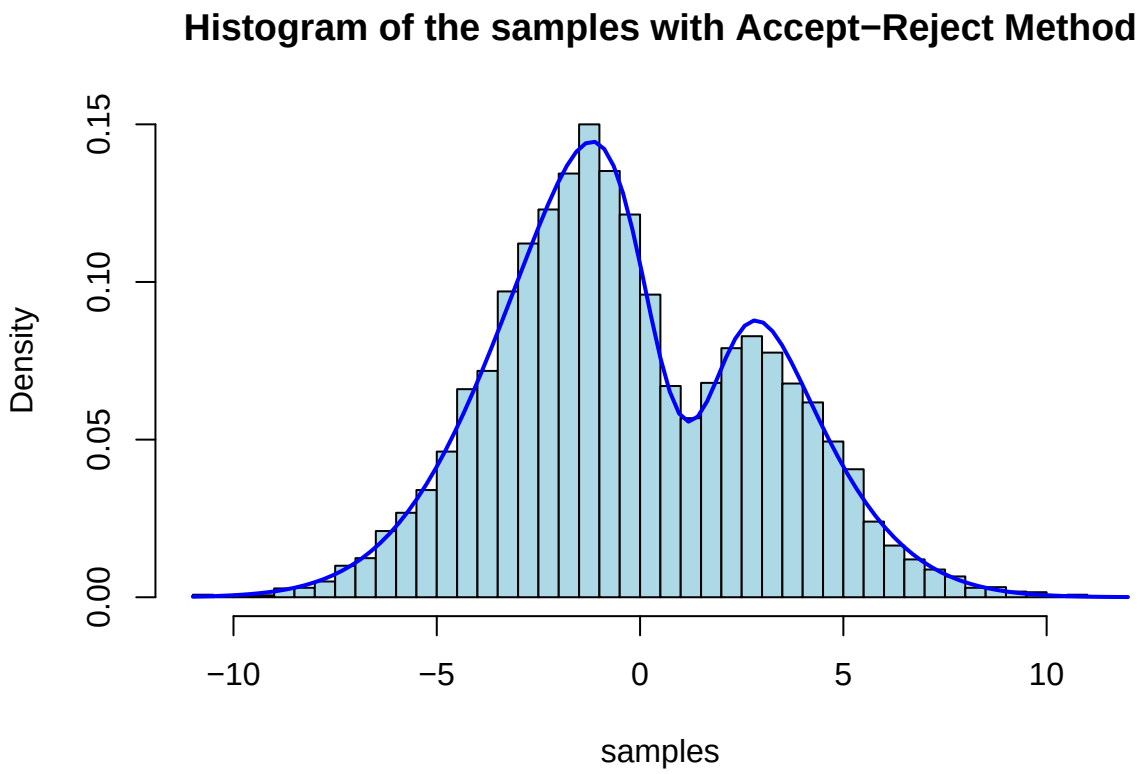
```
###Accept-Reject Random Variable simulation###
M=1/(1-a) #theoretical acceptance rate
n=10000

accept_reject<-function(n){
  samples <- numeric(n)
  count <- 1
  while (count <= n) {
    U <- runif(1)
    X <- rnorm(1, mu_1, s_1)
    R <- f(a, mu_1, mu_2, s_1, s_2, X) / (M * dnorm(X, mu_1, s_1))
    if (U <= R) {
      samples[count] <- X
      count <- count + 1
    }
  }
  return(samples)
}


samples <- accept_reject(n)
```

Validation with Histogram: We generated 10,000 samples and plotted a histogram, superposing the theoretical curve to visually verify that the samples align with $f(x)$. The match between the histogram and the density curve validates the method.

```
hist(samples, probability = TRUE, breaks = 50,
     main = "Histogram of the samples with Accept-Reject Method", col = "lightblue")
curve(f(a, mu_1, mu_2, s_1, s_2, x), add = TRUE, col = "blue", lwd = 2)
```



We now want to compute the empirical acceptance rate of the accept-reject method. We do it by counting the number of accepted samples and dividing it by the total number of attempts. This is done in the *calculate_acceptance_rate* function, which is similar to the *accept_reject* function but only returns the acceptance rate.

```r
# We will plot the acceptance rate as a function of a

# Initialize parameters
mu_1 <- 0
mu_2 <- 1
s_1 <- 3
s_2 <- 1

# Define a function to calculate acceptance rate for a specific value of a
calculate_acceptance_rate <- function(a, n = 10000) {
  M <- 1 / (1 - a)  # Theoretical acceptance rate
  samples <- numeric(n)
  count <- 1
  attempts <- 0  # Count attempts

  while (count <= n) {
    U <- runif(1)
    X <- rnorm(1, mu_1, sqrt(s_1))
    R <- f(a, mu_1, mu_2, s_1, s_2, X) / (M * dnorm(X, mu_1, sqrt(s_1)))
    attempts <- attempts + 1  # Increment attempts
    if (U <= R) {
```

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

  # Calculate acceptance rate
  acceptance_rate <- n / attempts
  return(acceptance_rate)
}
```

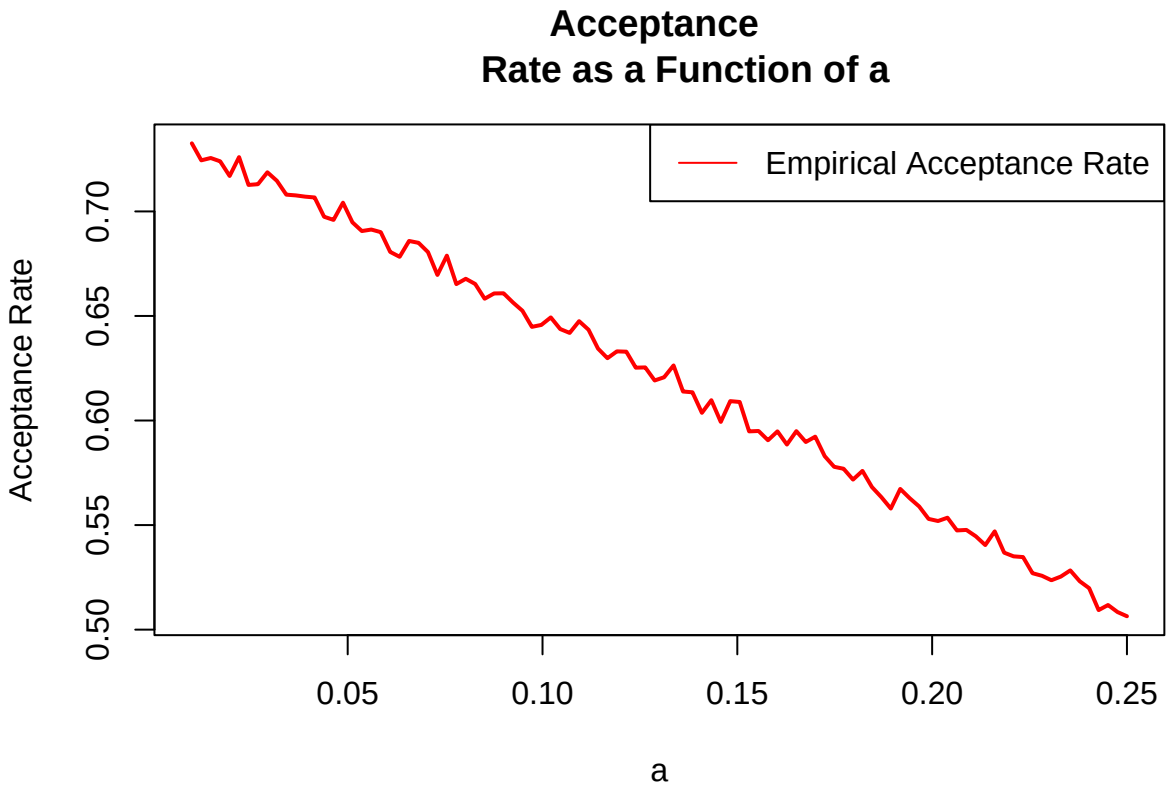
Question 8:

We plotted the empirical and theoretical acceptance rates as a function of a to observe the trend. The graph shows that both acceptance rates decreases as a approaches a^*, due to the increasing influence of $f_2(x)$, which causes more rejections.

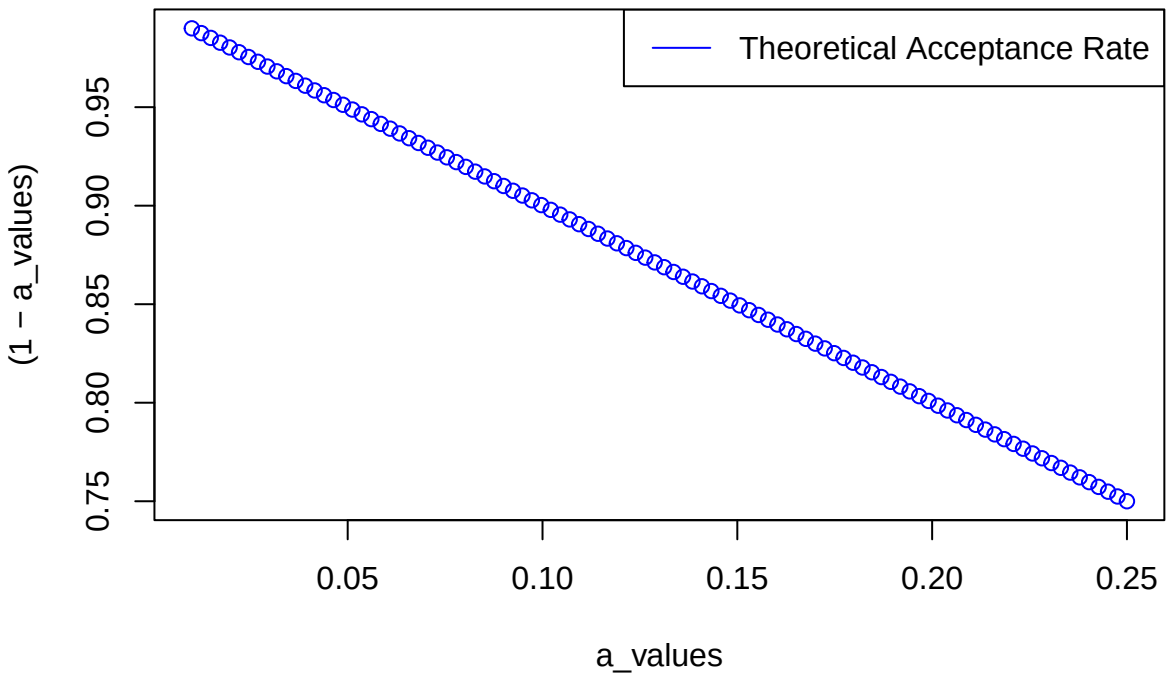
```
# Values of a from 0 (excluded) to a_star
a_values <- seq(0.01, a_star, length.out = 100)  # Avoiding 0 because division by zero

# Store the acceptance rates
acceptance_rates <- sapply(a_values, calculate_acceptance_rate)

# Plot the acceptance rate
plot(a_values, acceptance_rates, type = "l", col = "red", lwd = 2, main = "Acceptance
     Rate as a Function of a", xlab = "a", ylab = "Acceptance Rate")
legend("topright", legend = c("Empirical Acceptance Rate"), col = c("red"), lty = c(1, 1))
```




```r
plot(a_values, (1 - a_values), col = "blue")
legend("topright", legend = c("Theoretical Acceptance Rate"), col = c("blue"), lty = c(1, 1))
```



Interpretation: This result demonstrates that higher values of a reduce the method's efficiency, as more rejections are needed to obtain valid samples. This confirms the importance of keeping a relatively low to improve the acceptance rate.

Random Variable simulation with stratification

Question 9

Let's denote,

$$\tilde{f}_i(x) := f(x)\mathbf{1}_{D_i}(x), \quad \forall x \in \mathbb{R}, \quad \forall i \in 0, \ldots, k.$$

For all $x \in \mathbb{R}$,

$$\begin{aligned}\tilde{f}_0(x) &\leq (1-a) f_1(x) \mathbf{1}_{Di}(x) \\ &= (1-a)\Big(\int_{D_0} f_1(x))\, dx\Big) \frac{f_1(x)\mathbf{1}_{D_0}(x)}{\int_{D_0} f_1(x)} \\ &= M_0\, g_0(x)\end{aligned}$$

where $M_0 = (1-a)\int_{D_0} f_1(x)\, dx$ and $g_0(x) = \frac{f_1(x)\mathbf{1}_{D_0}(x)}{\int_{D_0} f_1(x)\, dx}$ is a density function.

For all $i \in \{1, \ldots, k\}$

$$\begin{aligned}\tilde{f}_i(x) &:= f(x)\, \mathbf{1}_{D_i}(x) \\ &\leq (1-a)\left(\sup_{u \in D_i} f(u) - a \inf_{u \in D_i} f(u)\right) \mathbf{1}_{D_i}(x)\end{aligned}$$

$$= (1-a)|D_i| \left(\sup_{u \in D_i} f(u) - a \inf_{u \in D_i} f(u) \right) \frac{\mathbf{1}_{D_i}(x)}{|D_i|}$$
$$= M_i \, g_i(x)$$

where $M_i = (1-a)|D_i| \left(\sup_{u \in D_i} f(u) - a \inf_{u \in D_i} f(u)\right)$, $g_i(x) = \frac{\mathbf{1}_{D_i}(x)}{|D_i|}$ the density function of a $\mathcal{U}(D_i)$, and $|D_i|$ denotes the Lebesgue measure of D_i.

We denote, for all $x \in \mathbb{R}$,

$$g_{\max}(x) = \sum_{i=0}^{k} g_i(x).$$

Moreover, let

$$M_{\max} = \max(M_0, M_1, \ldots, M_k).$$

We thus have that, for all $x \in \mathbb{R}$,

$$f(x) \leq M_{\max} g_{\max}(x).$$

Algorithm: Let $n \in \mathbb{N}$,

- **Step 1:** We draw $U \sim \mathcal{U}([0,1])$
- **Step 2:** We draw $Y \sim g_{\max}$
- **Step 3:** If $U \leq \frac{f(Y)}{M_{\max} \, g_{\max}(Y)}$, we accept Y, otherwise we go back to step 1.

The theoretical acceptance rate of this algorithm is $M_{\max}$.

Question 11

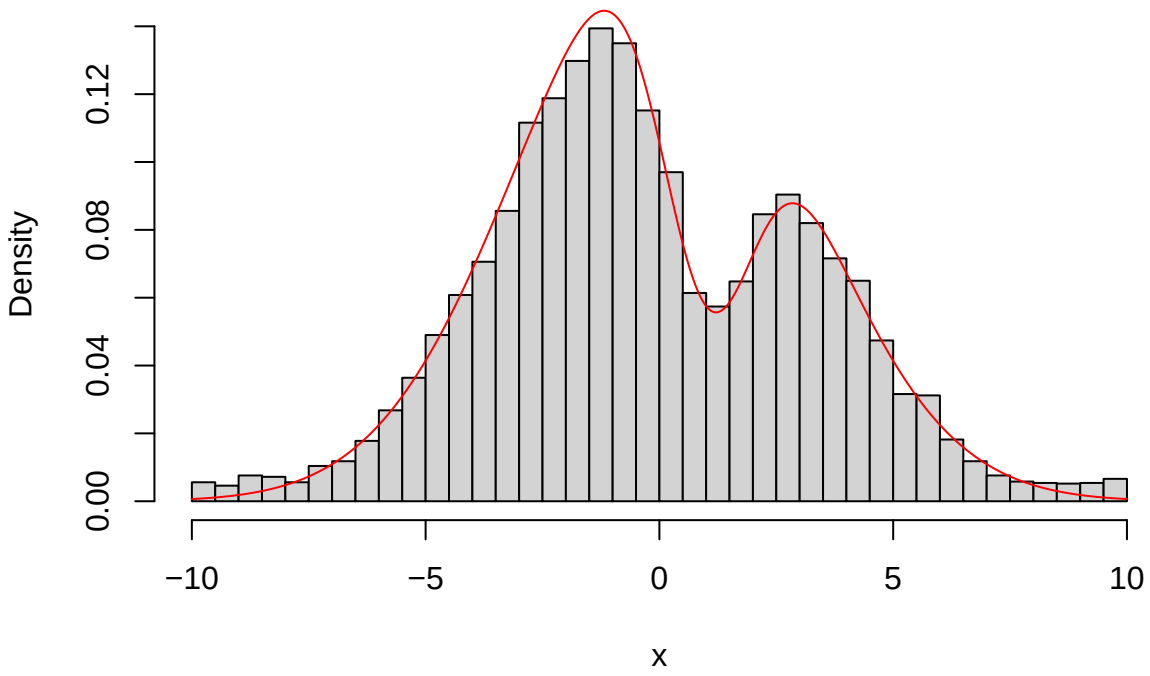
We get the following plot for $k = 10$:

```
n=10000
simulation = stratified(n)
```

```
## Empirical acceptance rate: 0.002189725
## Theorical acceptance rate: 0.02732484
```

```
hist(simulation, breaks = 50, probability = TRUE,
     main = "Histogram of the simulation by stratified accept-reject method", xlab = "x")

x = seq(-10, 10, 0.01)
y = f(a, mu_1, mu_2, s_1, s_2, x)
lines(x, y, col = "red")
```



Cumulative density function.

Question 13

For all $x \in \mathbb{R}$, on a :

$$F(x) = \frac{1}{1-a}\int_{-\infty}^{x} \big(f_1(t) - a\, f_2(t)\big)\, dt$$

$$= \frac{1}{1-a}\big(F_1(x) - aF_2(x)\big)$$

$$= \frac{1}{1-a}\Big[\int_{-\infty}^{x} \frac{e^{-\frac{(t-\mu_1)^2}{2\sigma_1^2}}}{\sqrt{2\pi}\,\sigma_1}\, dt - a\int_{-\infty}^{x} \frac{e^{-\frac{(t-\mu_2)^2}{2\sigma_2^2}}}{\sqrt{2\pi}\,\sigma_2}\, dt\Big]$$

$$= \frac{1}{1-a}\Big[\int_{-\infty}^{\frac{x-\mu_1}{\sigma_1}} \frac{e^{-\frac{t^2}{2}}}{\sqrt{2\pi}}\, dt - a\int_{-\infty}^{\frac{x-\mu_2}{\sigma_2}} \frac{e^{-\frac{t^2}{2}}}{\sqrt{2\pi}}\, dt\Big]$$

Let $x \in \mathbb{R}$, and $(X_i)_{i\geq 1}$ be i.i.d. random variables following the law of X.

The naive Monte Carlo estimator of $F(x)$ is given by:

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{X_i \leq x\}},$$

Question 14

For all $n \in \mathbb{N}^*$, we have :

$$0 \leq \left|F_n(x) - F(x)\right| \leq \sup_{x \in \mathbb{R}} \left|F_n(x) - F(x)\right|.$$

By Glivenko-Cantelli theorem, we know that :

$$\sup_{x \in \mathbb{R}} \left|F_n(x) - F(x)\right| \to 0 \quad \text{when } n \to +\infty.$$

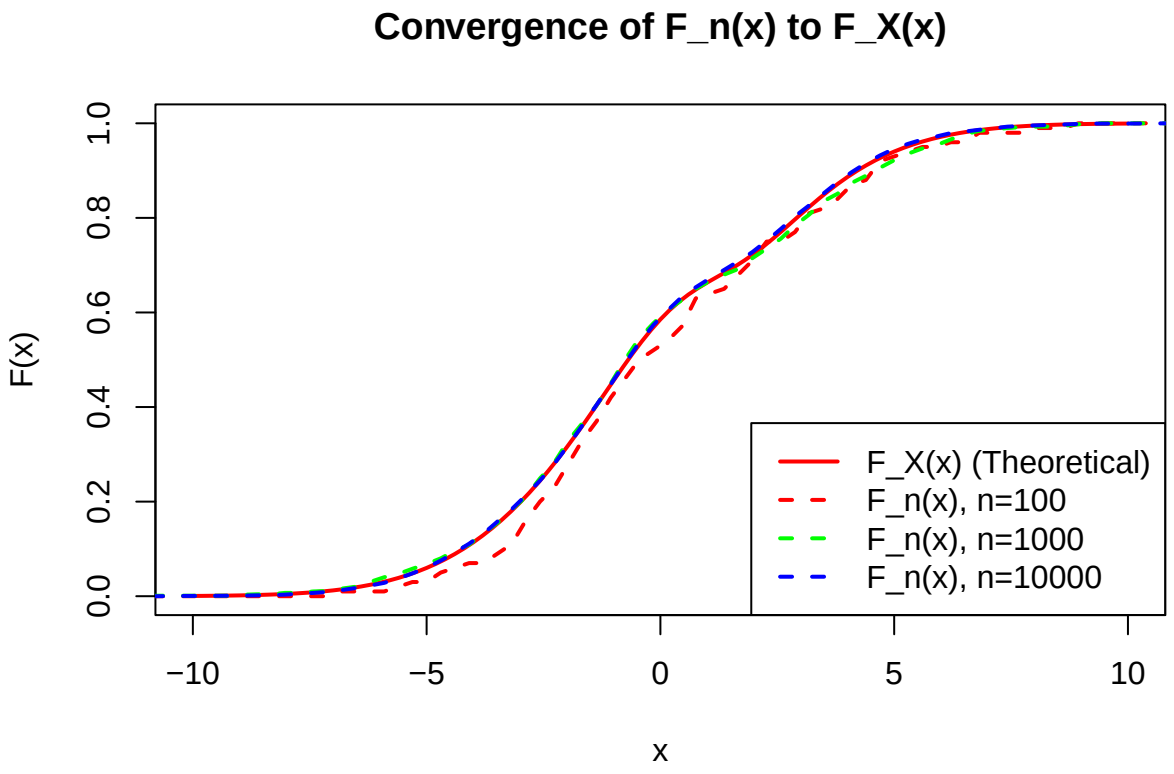
Thus, this implies that :

$$F_n(x) \to F(x) \quad \text{a.s..}$$

and the estimator $F_n(x)$ is strongly consistent. Indeed, we always have that a uniform convergence, implies the simple one (for x fixed).

Question 15

```
plot_cdf_convergence()
```



We observe that the estimator converges to the theoretical CDF as the sample size increases.

Question 16

The empirical Cumulative Distribution Function (CDF) $F_n(x)$ is defined as:

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{X_i \leq x\}},$$

where $\mathbf{1}_{\{X_i \leq x\}}$ is the indicator function, which equals 1 if $X_i \leq x$, and 0 otherwise.

Since the $(\mathbf{1}_{X_i \leq x})_{i \in [|1,n|]}$ are i.i.d. as a measurable transform of the i.i.d. sample $(X_i)_{i \in [|1,n|]}$, and

$$\mathrm{Var}(\mathbf{1}_{X_1 \leq x}) = F(x)(1 - F(x)) < +\infty,$$

the Central Limit Theorem (CLT) gives:

$$\sqrt{n}\big(F_n(x) - F(x)\big) \xrightarrow[n \to +\infty]{\mathcal{L}} \mathcal{N}\big(0, F(x)(1 - F(x))\big),$$

We can estimate the variance through the empirical cumulative distribution $F_n(x)$ once again.

Using the asymptotic normality of $F_n(x)$ along with Slutsky's theorem and the continuity of the inverse function on $\mathbb{R}^*$, we have:

$$\sqrt{\frac{n}{F_n(x)(1 - F_n(x))}}\big(F_n(x) - F(x)\big) \xrightarrow[n \to +\infty]{\mathcal{L}} \mathcal{N}(0, 1)$$

Thus, a 95% confidence interval for $F(x)$ can be constructed as:

$$F_n(x) \pm z_{0.975}\sqrt{\frac{F_n(x)(1 - F_n(x))}{n}},$$

where $z_{0.975}$ is the critical value for a 95% confidence level.

We get the following confidence interval for $x = 1$, $x = -15$ and $n = 1000$:

```
x=1
show_confidence_interval(x,n)
```

```
## Confidence interval at level 95% for x= 1 and n= 1000 : [ 0.6275776 , 0.6805428 ]
## Theorical value of F_X( 1 ): 0.6631983
```

```
x=-15
show_confidence_interval(x,n)
```

```
## Confidence interval at level 95% for x= -15 and n= 1000 : [ 0 , 0 ]
## Theorical value of F_X( -15 ): 3.583145e-07
```

Question 17

In this question, we aim to find the number of simulations n required to achieve a 95% confidence interval for $F(x)$ with a precision $\epsilon > 0$.

We thus must have:

$$z_{0.975}\sqrt{\frac{F_n(x)(1 - F_n(x))}{n}} \leq \epsilon.$$

Rearranging the inequality to isolate n:

$$\sqrt{\frac{F_n(x)(1-F_n(x))}{n}} \leq \frac{\epsilon}{z_{0.975}}.$$

Squaring both sides:

$$\frac{F_n(x)(1-F_n(x))}{n} \leq \frac{\epsilon^2}{z_{0.975}^2}.$$

Finally, solving for n:

$$n \geq \frac{z_{0.975}^2 F_n(x)(1-F_n(x))}{\epsilon^2}.$$

Since we computed the value of $F(x)$, we can replace the value of $F_n(x)$ by it in the formula to get the value of n. We get that

$$N = \left\lceil \frac{z_{0.975}^2 F(x)(1-F(x))}{\epsilon^2} \right\rceil$$

```
eps=0.01
number_of_simulations<-function(x,eps){
  N=floor(q^2*F_X(x)*(1-F_X(x))/(eps^2))+1
  return(N)
}

cat("We need",number_of_simulations(1,eps),
    "simulations to get a confidence interval of width",
    eps, "for x=1\n")
```

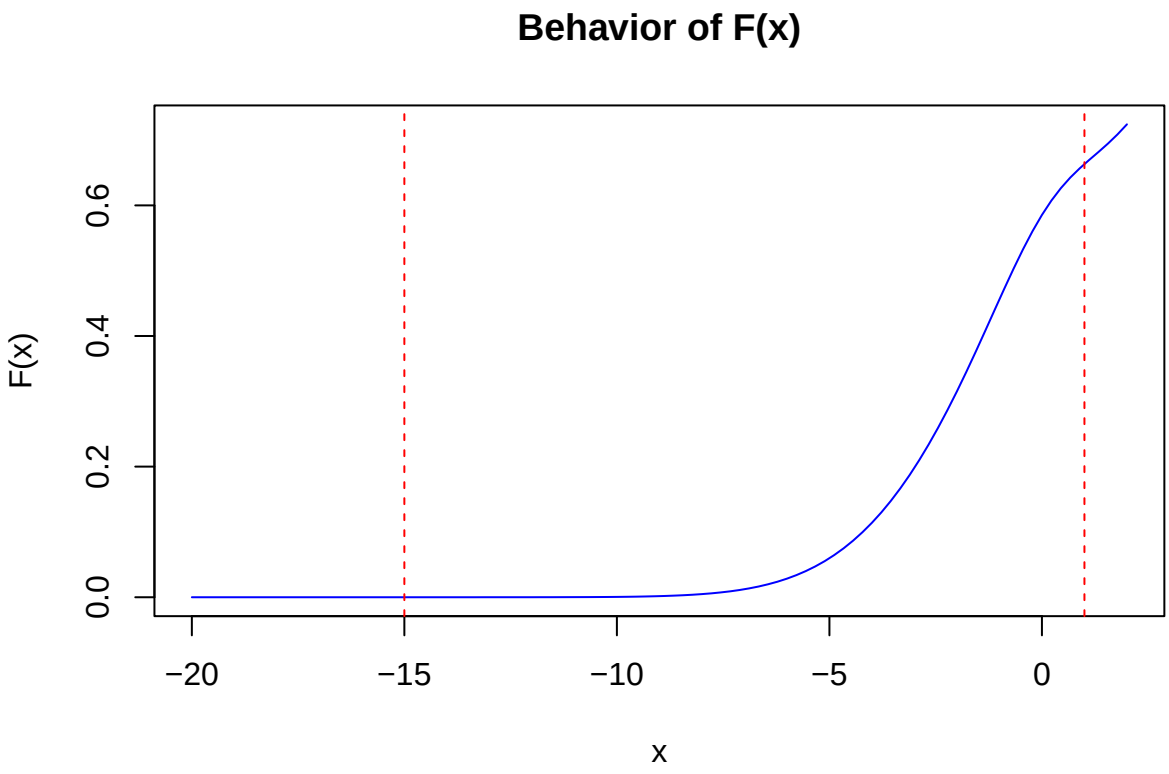
```
## We need 8581 simulations to get a confidence interval of width 0.01 for x=1
```

```
cat("We need",number_of_simulations(-15,eps),
    "simulations to get a confidence interval of width",
    eps ,"for x=-15\n")
```

```
## We need 1 simulations to get a confidence interval of width 0.01 for x=-15
```

We plot the curve of F to understand these values:

```
plot_F_behavior()
```



We remark that since $F(-15)$ is nearly 0, we need only one simulation to get a confidence interval of width 0.01. On the other hand, for $F(1)$, we need much more simulations to have the same precision. This amount can explode if we take a too small precision.

Empirical quantile function

Question 18

Let $u \in (0,1)$, the empirical quantile function is defined as:

$$Q_n(u) = \inf\{x \in \mathbb{R} : F_n(x) \geq u\}.$$

It is the generalized inverse of F_n, and to compute it, we can use order statistics. We denote

$$X_{(1)} \leq X_{(2)} \leq \cdots \leq X_{(n)}$$

the sorted values of a given sample $X_1, \ldots, X_n$. We have that $\forall i \in \{1, \ldots, n\}$,

$$F_n(X_{(i)}) = \frac{i}{n}.$$

Therefore, the smallest i such that $F_n(X_{(i)}) \geq u$ corresponds to $i = \lceil nu \rceil$.

We finally get that:

$$Q_n(u) = X_{(\lceil nu \rceil)}.$$

Question 20

By the result of the previous question, we have:

$$\sqrt{n}\big(Q_n(u) - Q(u)\big) \xrightarrow[n\to+\infty]{\mathcal{L}} N\left(0, \frac{u(1-u)}{f(Q(u))^2}\right)$$

Now, consider the behavior as $u \to 1$ or $u \to 0$:

- When $u \to 1$, $Q(u)$ approaches approximately 10.
- When $u \to 0$, $Q(u)$ approaches approximately -10.

Since the support of f is bounded, the quantity $\frac{u(1-u)}{f(Q(u))^2}$ remains finite and does not explode.

Furthermore, as $u(1-u) \to 0$ when $u \to 0$ or $u \to 1$, we have:

$$\frac{u(1-u)}{f(Q(u))^2} \to 0 \quad \text{as } u \to 0 \text{ or } u \to 1.$$

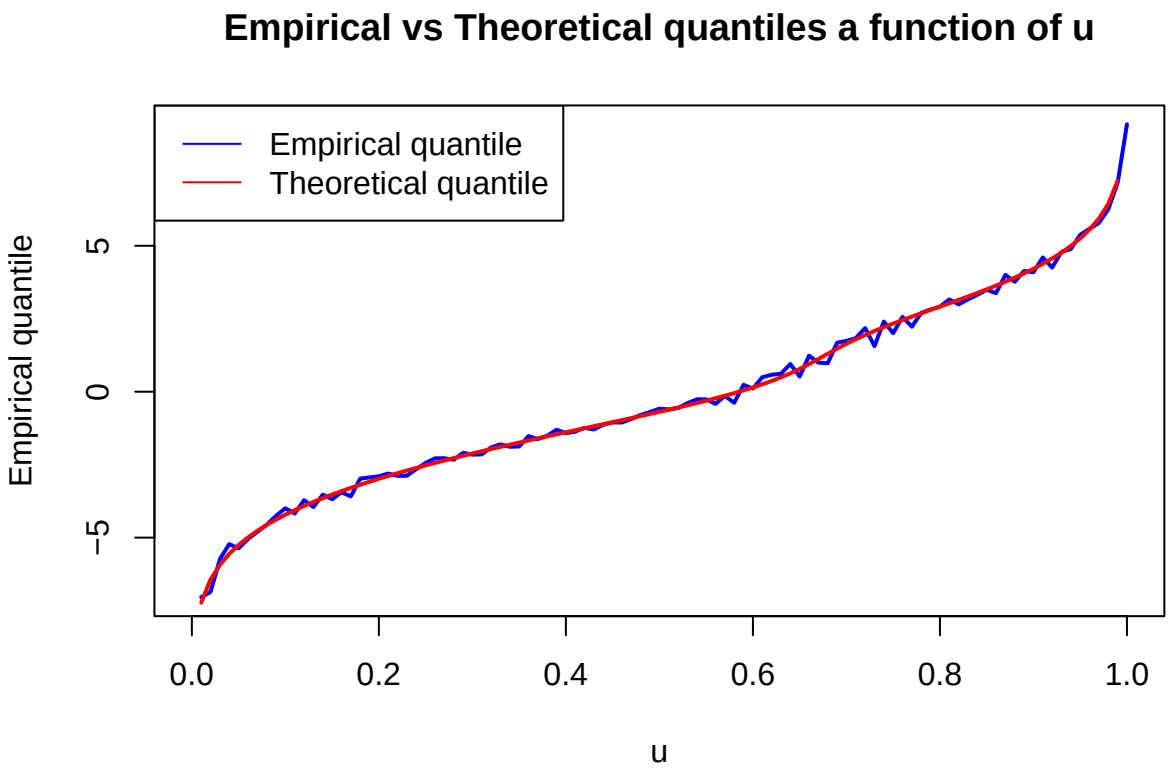
This result implies that the variance of $Q_n(u)$ decreases near the tails (u close to 0 or 1), making the estimator $Q_n(u)$ increasingly precise in those regions.

Question 21

We plot the empirical quantile function $Q_n(u)$ for $u \in (0,1)$ with $n = 1000$ simulations. We also superimpose the theoretical quantile function $Q(u)$ to visually compare the two.

```
u_values=seq(0,1,0.01)
#empirical quantile for n=1000
y=sapply(u_values, function(u){empirical_quantile(u,accept_reject(1000))})
#We use the inverse CDF method to get the quantile
y1=sapply(u_values, function(u){
return(inv_cdf_approx(u,                    create_cdf_table(a,mu_1,mu_2,s_1,s_2)))})

plot(u_values,y,type="l",col="blue",xlab="u",
ylab="Empirical quantile",
main="Empirical vs Theoretical quantiles a function of u",lwd=2)
lines(u_values,y1 ,col="red",lwd=2)
legend("topleft",
legend=c("Empirical quantile","Theoretical quantile"),col=c("blue","red"),lty=1)
```



We notice that the curve is steeper close to 0 and 1, which confirms our intuition stated in the last question.

Question 22

First, using the asymptotic normality of $Q_n(u)$ along with Slutsky's theorem and the continuity of the inverse function on $\mathbb{R}^*$, we have:

$$\left|f(Q(u))^2)\right| \sqrt{\frac{n}{u(1-u)}} \big(Q_n(u) - Q(u)\big) \xrightarrow[n\to+\infty]{\mathcal{L}} \mathcal{N}(0,1)$$

Thus, a 95% confidence interval for $Q(u)$ can be constructed as:

$$Q_u(x) \pm z_{0.975} \sqrt{\frac{u(1-u)}{n \cdot f(Q_n(u))^2}},$$

Moreover, as in Question 17, we aim to find the number of simulations n required to achieve a 95% confidence interval for $Q(u)$ with a precision $\epsilon > 0$, with $u \in (0,1)$ given.

Using the result of Question 19, we must have that:

$$z_{0.975} \sqrt{\frac{u(1-u)}{n \cdot f(Q_n(u))^2}} \leq \epsilon.$$

Rearranging the inequality to isolate n:

$$\sqrt{\frac{u(1-u)}{n \cdot f(Q_n(u))^2}} \leq \frac{\epsilon}{z_{0.975}}.$$

Squaring both sides:

$$\frac{u(1-u)}{n \cdot f(Q_n(u))^2} \leq \frac{\epsilon^2}{z^2_{0.975}}.$$

Finally, solving for n:

$$n \geq \frac{z^2_{0.975}}{\epsilon^2} \frac{u(1-u)}{f(Q_n(u))^2}.$$

Since we can compute the value of $Q(u)$, we can replace the value of $Q_n(u)$ by it in the formula to get the value of n. We get that

$$N = \left\lceil \frac{z^2_{0.975}}{\epsilon^2} \frac{u(1-u)}{f(Q(u))^2} \right\rceil$$

We compute the number of simulations needed for the different values of u given.

```
show_simulations_quantile<-function(U,eps){
  for(u in U){
    cat("We need",number_of_simulations_quantile(u,eps),"simulations to get
a confidence interval of width", eps, "for u=",u,"\n")
  }
}

show_simulations_quantile(u_values,0.01)
```

```
## We need 901032 simulations to get
## a confidence interval of width 0.01 for u= 0.1
## We need 598412 simulations to get
## a confidence interval of width 0.01 for u= 0.2
## We need 447199 simulations to get
## a confidence interval of width 0.01 for u= 0.4
## We need 504593 simulations to get
## a confidence interval of width 0.01 for u= 0.5
## We need 915620 simulations to get
## a confidence interval of width 0.01 for u= 0.9
## We need 4559508 simulations to get
## a confidence interval of width 0.01 for u= 0.99
## We need 29388276 simulations to get
## a confidence interval of width 0.01 for u= 0.999
## We need NA simulations to get
## a confidence interval of width 0.01 for u= 0.9999
```

We notice that the amount of simulations needed increases as u approaches 0 or 1, which is consistent with the results of Question 20. A NA appears in for $u = 0.9999$ because the amount of simulations needed is too high to be computed.

Quantile estimation Naïve Reject algorithm

Question 23

Let $A \subset \mathbb{R}$, X a random variable, and denote $\mathcal{L}(X \mid X \in A)$, the law of X conditional to the event $\{X \in A\}$. We first suppose that $\mathbb{P}(X \in A) > 0$, otherwise the law conditional to that event is not well defined. The density function of a random variable following this law is given by:

$$\forall x \in \mathbb{R}, \quad f_{X|X\in A}(x) = f_X(x)\frac{\mathbf{1}_{\{x\in A\}}}{\mathbb{P}(X \in A)} \leq \frac{f_X(x)}{\mathbb{P}(X \in A)}$$

Here we can set $M = \frac{1}{\mathbb{P}(X\in A)}$, our normalizing constant.

Denoting

$$T = \inf\{n \in \mathbb{N} \mid U_n \leq \frac{f_{X|X\in A}(X_n)}{M f_X(X_n)}\}$$

where $(X_n) \overset{\text{i.i.d.}}{\sim} X$ and $(U_n) \overset{\text{i.i.d.}}{\sim} \mathcal{U}([0,1])$,

We get by accept-reject method that $X_T \sim \mathcal{L}(X \mid X \in A)$. This can also be seen as the fact that we accept the first draw of X that is in A, i.e, $T = \inf\{n \in \mathbb{N} \mid X_n \in A\}$ because :

- If $X \in A$: $\frac{f_{X|X\in A}(X_n)}{M f_X(X_n)} = 1$ so what ever is the value of U_n, we will accept X_n.
- If $X \notin A$: $\frac{f_{X|X\in A}(X_n)}{M f_X(X_n)} = 0$, so whatever is the value of U_n, we will reject X_n.

We remark that the more $\mathbb{P}(X \in A)$ is small, the more M is high, and thus the more this simulation is inefficient.

Question 24

Following the method implemented in the last question, and by taking $A = [q, +\infty[$, we can estimate $\delta = P(X \geq q)$ with the following code:

```
accept_reject_quantile<-function(q,n){
  Xn=accept_reject(n)
  delta_hat=mean(Xn>=q)
  return(delta_hat)
}
```

Question 25

Taking $u = 0.3$, we get the following confidence interval for δ at level 95% for a required precision $\epsilon = 0.01$:

```
show_confidence_interval_delta(u,n,eps)
```

```
## Confidence interval at level 95% for u= 0.3 and n= 10000  for
## the accept-reject method : [ 0.3820333 , 0.4011667 ]
## Number of simulations needed to get a confidence interval of width 0.01  and for u= 0.3  :  9153
```

Importance Sampling

Question 26:

To choose a good function $g(\cdot)$, we must ensure:

(i) Support inclusion, i.e. $\text{supp}(\varphi f) \subseteq \text{supp}(g)$,
that is $\varphi(x)f(x) > 0\} \subseteq \{x : g(x) > 0\}$,

(ii) The variance must be finite.

Theoretically, the optimal choice of g with minimal variance is:

$$g^*(y) = \frac{|f(y)|\, g(y)}{\displaystyle\int |f(y')|\, g(y')\, dy'}.$$

However, this distribution is generally unknown because the integral in the denominator is not explicitly computable.But we cannot calculate this integral [] In our case we propose using a **truncated exponential distribution** $g(x)$, defined over an interval $[a, b]$. Its density is given by:

$$g(x) = \begin{cases} \frac{\lambda e^{-\lambda(x-a)}}{1-e^{-\lambda(b-a)}} & \text{if } x \in [a, b], \\ 0 & \text{otherwise,} \end{cases}$$

where $\lambda > 0$ is the rate parameter, and a and b are the truncation bounds.

To adapt g to the target density $f(x)$, we select the parameters as follows:

- The **rate parameter** λ is set to $\lambda = 0.5$, which provides a balance between the rapid decay of the exponential function and sufficient coverage of the tails.
- The **lower bound** a is chosen as -5, ensuring that the truncated exponential includes the regions where $f(x)$ is significant.
- The **upper bound** b is set to 10, ensuring coverage of the tail regions where rare events may occur.

The truncated exponential distribution $g(x)$ offers several advantages:

- It ensures finite variance of the importance weights $w(x) = f(x)/g(x)$.
- It is computationally simple to sample from and evaluate.
- By adjusting the truncation bounds and the rate parameter λ, $g(x)$ can be tailored to the regions of interest in $f(x)$, such as the tails where rare events occur.

The importance sampling (IS) estimator is given by:

$$\hat{\delta}_n^{IS} = \frac{1}{n} \sum_{i=1}^{n} \varphi(X_i) \frac{f(X_i)}{g(X_i)},$$

where X_i are i.i.d. with density g.

In our case case, the IS estimator is better than the standard Monte Carlo estimator ($\hat{\delta}_n^{MC}$) when:

- We are studying a rare event.
- The function to be integrated has sharp variations.
- The function g is well-chosen.

Question 27:

For $X \sim \mathcal{C}(\mu_0, \gamma)$ (Cauchy distribution with parameters μ_0 and γ), the density is:

$$g(x) = \frac{1}{\pi\gamma} \left[1 + \left(\frac{x - \mu_0}{\gamma} \right)^2 \right]^{-1}.$$

where μ_0 is the location parameter and $\gamma > 0$ is the scale parameter.

The parameters of g were chosen following the steps below:

1. **Choosing μ_0:** The target density $f(x)$ is a mixture of two normal distributions centered at $\mu_1 = 0$ and $\mu_2 = 1$. The second peak, located around $\mu_2 = 1$, dominates for positive values of x. Therefore, we set $\mu_0 = 1$ to center the Cauchy distribution around this region of interest.

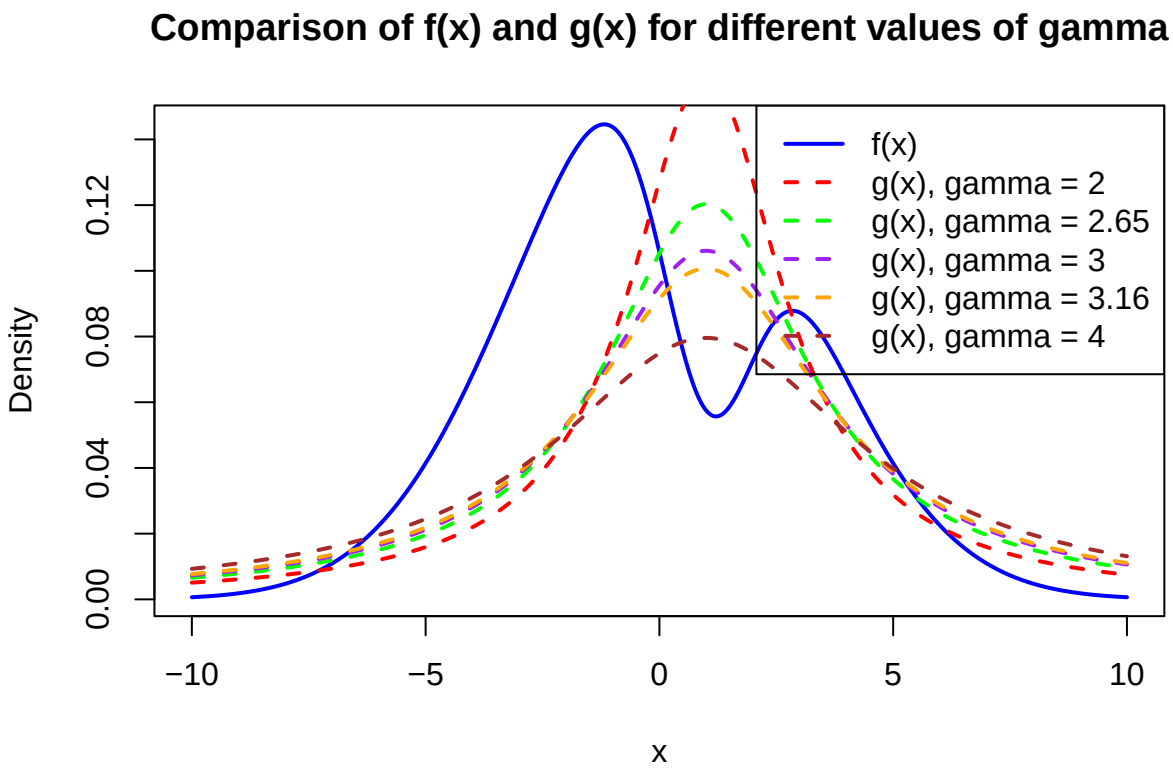
2. **Choosing γ:** The value of γ must allow $g(x)$ to effectively cover both peaks of $f(x)$ as well as its tails. After testing various values of γ (including 2, $\sqrt{7}$, 3, $\sqrt{10}$, and 4), we observed that $\gamma = \sqrt{10}$ provided the best compromise. This value ensures sufficient coverage of the regions where $f(x)$ is significant while maintaining an acceptable variance of the importance sampling weights $w(x) = f(x)/g(x)$.

The figure below illustrates the behavior of $g(x)$ for different values of γ, overlaid on the target density $f(x)$. It can be seen that the final choice of parameters $\mu_0 = 1$ and $\gamma = \sqrt{10}$ offers a good approximation of $f(x)$, thus reducing the variance of the importance sampling estimator.

```r
plot(x_vals, f_vals, type = "l", col = "blue", lwd = 2,
     ylab = "Density", xlab = "x",
     main = "Comparison of f(x) and g(x) for different values of gamma")

# Overlay g(x) for each gamma
for (i in 1:length(gamma_values)) {
  gamma <- gamma_values[i]
  g_vals <- g(x_vals, mu0, gamma)  # Calculate g(x) values
  lines(x_vals, g_vals, col = colors[i], lwd = 2, lty = 2)
}

legend("topright",
       legend = c("f(x)", paste("g(x), gamma =", round(gamma_values, 2))),
       col = c("blue", colors),
       lty = c(1, rep(2, length(gamma_values))),
       lwd = 2)
```



Question 28

Objective

The aim of this question is to estimate $\delta = P(X \geq q)$ using importance sampling. The method uses a Cauchy proposal density $g(x)$ with parameters $\mu_0 = q$ and $\gamma = \sqrt{10}$ to approximate the target density $f(x)$ efficiently.

Approach

The R function below implements the importance sampling procedure:

```
gamma <- sqrt(10)
IS_quantile <- function(u, n, epsilon = NULL) {
  Xn <- rcauchy(n, u, gamma)
  weights <- (f(a, mu_1, mu_2, s_1, s_2, Xn) / dcauchy(Xn, u, gamma)) * (Xn >= u)
  delta_IS <- mean(weights)
  var_delta_IS <- (mean(weights^2) - delta_IS^2) / n
  IC_inf <- delta_IS - 1.96 * sqrt(var_delta_IS / n)
  IC_sup <- delta_IS + 1.96 * sqrt(var_delta_IS / n)

  if (!is.null(epsilon)) {
    repeat {
```

```
      if (0.5 * (IC_sup - IC_inf) <= epsilon) break
      n <- n + 1
      Xn <- c(Xn, rcauchy(1, u, gamma))
      weights <- (f(a, mu_1, mu_2, s_1, s_2, Xn) / dcauchy(Xn, u, gamma)) * (Xn >= u)
      delta_IS <- mean(weights)
      var_delta_IS <- (mean(weights^2) - delta_IS^2) / n
      IC_inf <- delta_IS - 1.96 * sqrt(var_delta_IS / n)
      IC_sup <- delta_IS + 1.96 * sqrt(var_delta_IS / n)
    }
  }
  return(list(delta_IS = delta_IS, CI = c(IC_inf, IC_sup), n = n))
}
```

Key Steps

We generate n samples from $g(x) \sim \mathcal{C}(\mu_0, \gamma)$.

We compute the importance weights:

$$w(x) = \frac{f(x)}{g(x)} \cdot \mathbf{1}_{\{x \geq u\}}.$$

We calculate the estimate δ_{IS} and the 95% confidence interval:

$$\delta_{IS} = \frac{1}{n} \sum_{i=1}^{n} w(X_i), \quad \text{CI} = \left[\delta_{IS} \pm 1.96 \sqrt{\frac{\text{Var}(\delta_{IS})}{n}} \right].$$

Results

With $u = 0.3$, $n = 10,000$, and $\gamma = \sqrt{10}$:

```
print("Estimation of delta_IS : 0.3903642")
```

```
## [1] "Estimation of delta_IS : 0.3903642"
```

```
print("95% Confidence Interval : 0.3902576 to 0.3904707")
```

```
## [1] "95% Confidence Interval : 0.3902576 to 0.3904707"
```

The results show that importance sampling is effective for estimating δ. The choice of $\gamma = \sqrt{10}$ allows $g(x)$ to cover the significant regions of $f(x)$, ensuring low variance in the weights. The iterative feature ensures precision if needed. Overall, the method provides accurate results with a manageable computational cost.

Control Variate

Question 29

The score function, denoted by $s(\theta)$, is defined as the gradient of the log-likelihood function with respect to the parameter θ:

$$s(\theta) = \nabla_\theta \log \mathcal{L}(\theta) = \left(\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta_i} \right)_{i=1,2}$$

where $\mathcal{L}(\theta)$ is the likelihood function.

For $x \in \mathbb{R}$, $\quad f(x \mid \theta_1, \theta_2) = \frac{1}{1-a}\left[f_1(x) - a f_2(x)\right]$

Since only f_1 depends on μ_1, we get that:

$$\frac{\partial f(x \mid \theta_1, \theta_2)}{\partial \mu_1} = \frac{\partial f_1(x \mid \theta_1)}{(1-a)\partial \mu_1}$$

$$= \frac{(x-\mu_1)}{(1-a)\sigma_1^2} f_1(x \mid \theta_1)$$

So,

$$s_{\mu_1}(x|\theta_1, \theta_2) = \frac{\partial \log(f(x|\theta_1, \theta_2))}{\partial \mu_1}$$

$$= \frac{\partial f(x|\theta)}{\partial \mu_1} \frac{1}{f(x|\theta)}$$

$$= \frac{\partial f_1(x|\theta_1)}{\partial \mu_1} \frac{1}{f(x|\theta)(1-a)}$$

$$= \frac{(x-\mu_1)}{\sigma_1^2(1-a)} \frac{f_1(x|\theta_1)}{f(x|\theta)}$$

Question 30

The control variate Monte Carlo estimator $\hat{\delta}_n^{\mathrm{CV}}$ is defined as:

$$\hat{\delta}_n^{\mathrm{CV}} = \frac{1}{n}\left(\sum_{i=1}^{n} \mathbf{1}_{\{X_i \geq q\}} - b^* \cdot (s_{\mu_1}(X_i \mid \theta_1, \theta_2) - \mathbb{E}[s_{\mu_1}(X_1 \mid \theta_1, \theta_2)])\right),$$

where $X_1, \ldots, X_n \overset{\text{iid}}{\sim} f_X$, and

$$b^* = -\frac{\mathrm{Cov}(\varphi(X), Y)}{\mathrm{Var}(Y)}.$$

Question 31

The implemation of the control variate Monte Carlo estimator is given by the following R code:

```
CV_quantile<-function(q,n){
  Xn=accept_reject(n)
  b_star=-cov(Xn,s_mu_1(Xn,mu_1,s_1,mu_2,s_2))/var(s_mu_1(Xn,mu_1,s_1,mu_2,s_2))
  phi_1=sapply(Xn,function(x){phi(x,q)})
  s_mu_1=sapply(Xn,function(x){s_mu_1(x,mu_1,s_1,mu_2,s_2)})
  return(mean(phi_1+b_star*(s_mu_1-expectance_s_mu_1(mu_1,s_1,mu_2,s_2))))
}
```

A confidence interval for δ at level 95% with the control variate estimator is computed for $u = 0.3$, $n = 10000$, and $\epsilon = 0.01$:

```
show_confidence_interval_delta_CV(u,n,eps)
```

```
## Confidence interval at level 95% for u= 0.3 and
##   n= 10000  with the CV estimator: [ 0.3512813 , 0.3701049 ]
## Number of simulations needed to get a confidence interval of width 0.01  and for u= 0.3  :  8859
```

Question 32

The computational cost and precision of the three methods—Naive Estimator, Importance Sampling (IS), and Control Variate (CV)—are compared below. This includes their algorithmic complexity, number of simulations required for a 95% confidence interval, and estimated values.

Method	**Estimated Value (δ)**	**Number of Simulations (N)**
Control of Variate	0.3978	9495
Importance Sampling	0.39	10000
Naive Estimator	0.3833	9118
Theoretical value	0.3857	/

Advantages and Disadvantages

Naive Estimator

- **Advantages:** Straightforward implementation; no additional requirements on the sampling distribution.
- **Disadvantages:** High variance and limited precision compared to more advanced methods.

Importance Sampling (IS)

- **Advantages:** High efficiency for estimating rare probabilities; reduces variance significantly if $g(x)$ is well-chosen.
- **Disadvantages:** Requires careful selection of $g(x)$; performance degrades if $g(x)$ poorly approximates $f(x)$.

Control Variate (CV)

- **Advantages:** Reduces variance significantly; improves precision with fewer simulations by using correlated control variables.
- **Disadvantages:** Requires additional computations for selecting an appropriate control variable.

Algorithmic Complexity and Precision

- **Naive Estimator:** Moderate computational cost; average precision.
- **Importance Sampling:** Higher computational cost; excellent precision when $g(x)$ is optimal.
- **Control Variate:** Moderate to high computational cost; excellent precision when a good control variable is used.

Conclusion

The choice of method depends on the problem requirements:

- **Naive Estimator:** Suitable for simple problems where implementation ease is critical.
- **Importance Sampling:** Best for estimating rare events or tail probabilities.
- **Control Variate:** Ideal for improving precision with limited computational resources when a suitable control variable is available.