Groupe 03 Projet DANJOU - DUROUSSEAU

Contents

Exercise 1 : Negative weighted mixture

Definition

Question 1

The conditions for a function f to be a probability density are :

- f is defined on $\mathbb{R}$
- f is non-negative, ie $f(x) \geq 0$, $\forall x \in \mathbb{R}$
- f is Lebesgue-integrable
- and $\int_{\mathbb{R}} f(x)\,dx = 1$

The function f, to be a density, needs to be non-negative, ie $f(|x|) \geq 0$ when $|x| \to \infty$

We have, when $|x| \to \infty$, $f_i(|x|) \sim exp(\frac{-x^2}{\sigma_i^2})$ for $i = 1, 2$

Then, $f(|x|) \sim exp(\frac{-x^2}{\sigma_1^2}) - exp(\frac{-x^2}{\sigma_2^2})$

We want $f(|x|) \geq 0$, ie, $exp(\frac{-x^2}{\sigma_1^2}) - exp(\frac{-x^2}{\sigma_2^2}) \geq 0$

$\Leftrightarrow \sigma_1^2 \geq \sigma_2^2$

We can see that f_1 dominates the tail behavior.

Question 2

For given parameters (μ_1, σ_1^2) and (μ_2, σ_2^2), we have $\forall x \in \mathbb{R}$, $f(x) \geq 0$

$\Leftrightarrow \frac{1}{\sigma_1} exp(\frac{-(x-\mu_1)^2}{2\sigma_1^2}) \geq \frac{a}{\sigma_2} exp(\frac{-(x-\mu_2)^2}{2\sigma_2^2})$

$\Leftrightarrow 0 < a \leq a^* = \min_{x \in \mathbb{R}} \frac{f_1(x)}{f_2(x)} = \min_{x \in \mathbb{R}} \frac{\sigma_2}{\sigma_1} exp(\frac{(x-\mu_2)^2}{2\sigma_2^2} - \frac{(x-\mu_1)^2}{2\sigma_1^2})$

To find a^*, we just have to minimize $g(x) := \frac{(x-\mu_2)^2}{2\sigma_2^2} - \frac{(x-\mu_1)^2}{2\sigma_1^2}$

First we derive g: $\forall x \in \mathbb{R}, g'(x) = \frac{x-\mu_2}{\sigma_2^2} - \frac{x-\mu_1}{\sigma_1^2}$

We search x^* such that $g'(x^*) = 0$

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

Then, we compute $a^* = \frac{f_1(x^*)}{f_2(x^*)}$

We call $C \in \mathbb{R}$ the normalization constant such that $f(x) = C(f_1(x) - af_2(x))$

To find C, we know that $1 = \int_{\mathbb{R}} f(x)\,dx = \int_{\mathbb{R}} C(f_1(x) - af_2(x))\,dx = C\int_{\mathbb{R}} f_1(x)\,dx - Ca\int_{\mathbb{R}} f_2(x)\,dx = C(1-a)$ as f_1 and f_2 are density functions and by linearity of the integrals.

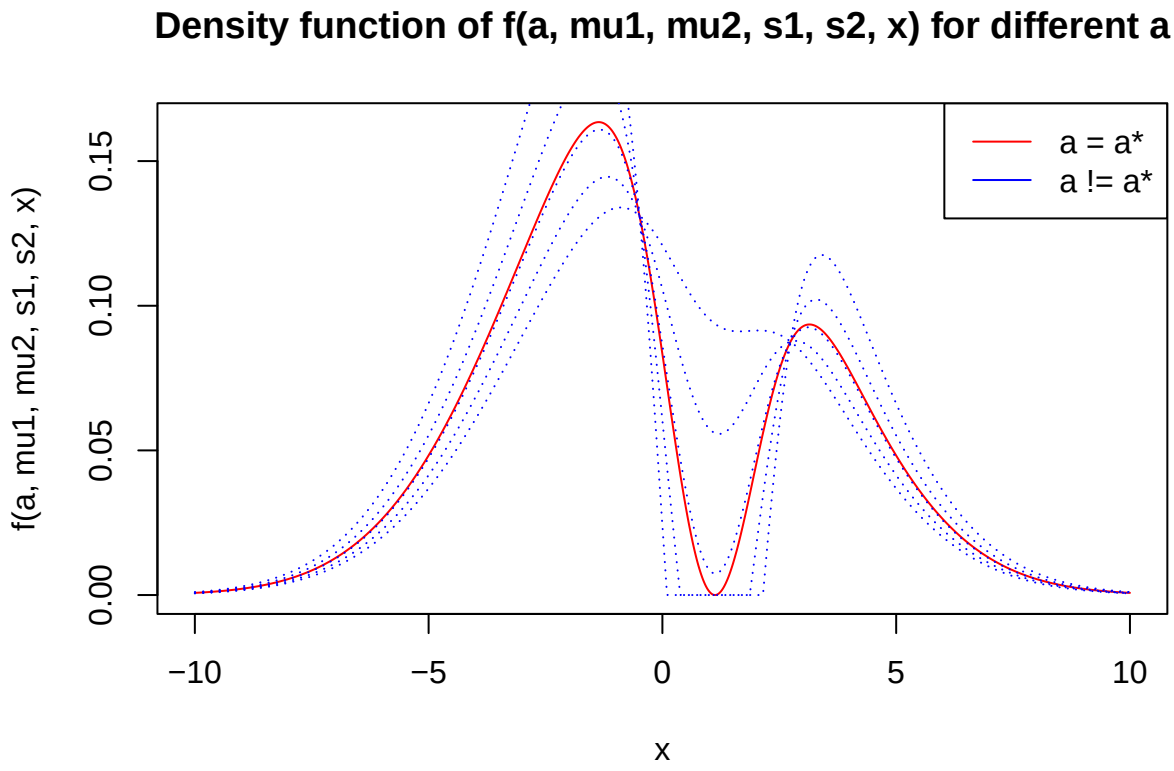
$\Leftrightarrow C = \frac{1}{1-a}$

Question 3

```r
f <- function(a, mu1, mu2, s1, s2, x) {
  fx <- dnorm(x, mu1, s1) - a * dnorm(x, mu2, s2)
  fx[fx < 0] <- 0
  return(fx / (1 - a))
}

a_star <- function(mu1, mu2, s1, s2) {
  x_star <- (mu2 * s1^2 - mu1 * s2^2) / (s1^2 - s2^2)
  return(dnorm(x_star, mu1, s1) / dnorm(x_star, mu2, s2))
}
```

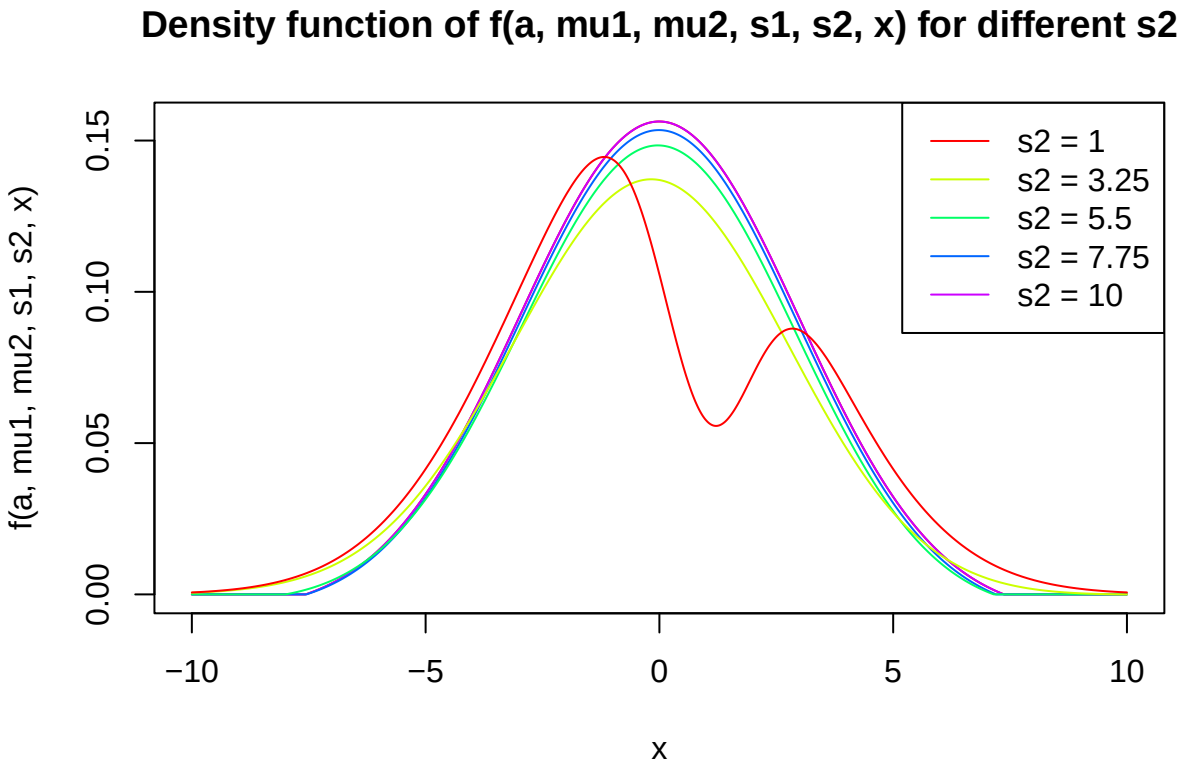
```r
mu1 <- 0
mu2 <- 1
s1 <- 3
s2 <- 1

x <- seq(-10, 10, length.out = 1000)
as <- a_star(mu1, mu2, s1, s2)
a_values <- c(0.1, 0.2, 0.3, 0.4, 0.5, as)

plot(x, f(as, mu1, mu2, s1, s2, x),
     type = "l",
     col = "red",
     xlab = "x",
     ylab = "f(a, mu1, mu2, s1, s2, x)",
     main = "Density function of f(a, mu1, mu2, s1, s2, x) for different a"
)
```

```r
for (i in (length(a_values) - 1):1) {
  lines(x, f(a_values[i], mu1, mu2, s1, s2, x), lty = 3, col = "blue")
}
legend("topright", legend = c("a = a*", "a != a*"), col = c("red", "blue"), lty = 1)
```



We observe that for small values of a, the density f is close to the density of $f_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$. When a increases, the shape of evolves into the combinaison of two normal distributions. We observe that for $a = a^*$, the density the largest value of a for which the density is still a density function, indeed for $a > a^*$, the function f takes negative values so it is no longer a density.

```r
s2_values <- seq(1, 10, length.out = 5)
a <- 0.2

plot(x, f(a, mu1, mu2, s1, max(s2_values), x),
     type = "l",
     xlab = "x",
     ylab = "f(a, mu1, mu2, s1, s2, x)",
     col = "red",
     main = "Density function of f(a, mu1, mu2, s1, s2, x) for different s2"
)

for (i in length(s2_values):1) {
  lines(x, f(a, mu1, mu2, s1, s2_values[i], x), lty = 1, col = rainbow(length(s2_values))[i])
}

legend("topright", legend = paste("s2 =", s2_values), col = rainbow(length(s2_values)), lty = 1)
```



We observe that when $\sigma_2^2 = 1$, the density f has two peaks and when $\sigma_2^1 > 1$, the density f has only one peak.

```
mu1 <- 0
mu2 <- 1
sigma1 <- 3
sigma2 <- 1
a <- 0.2
as <- a_star(mu1, mu2, sigma1, sigma2)

cat(sprintf("a* = %f, a = %f, a <= a* [%s]", as, a, a <= as))
```

```
## a* = 0.313138, a = 0.200000, a <= a* [TRUE]
```

We have $\sigma_1^2 \geq \sigma_2^2$ and $0 < a \leq a^*$, so the numerical values are compatible with the constraints defined above.

Inverse c.d.f Random Variable simulation

Question 4

To prove that the cumulative density function F associated with f is available in closed form, we need to compute $F(x) = \int_{-\infty}^{x} f(t)\,dt = \frac{1}{1-a}(\int_{-\infty}^{x} f_1(t)\,dt - a\int_{-\infty}^{x} f_2(t)\,dt) = \frac{1}{1-a}(F_1(x) - aF_2(x))$ where F_1 and F_2 are the cumulative density functions of $f_1 \sim \mathcal{N}(\mu 1, \sigma_1^2)$ and $f_2 \sim \mathcal{N}(\mu 2, \sigma_2^2)$ respectively.

Then, F is a closed-form as a finite sum of closed forms.

```
F <- function(a, mu1, mu2, s1, s2, x) {
  Fx <- pnorm(x, mu1, s1) - a * pnorm(x, mu2, s2)
  return(Fx / (1 - a))
}
```

To construct an algorithm that returns the value of the inverse function method as a function of $u \in (0,1)$, of the parameters $a, \mu_1, \mu_2, \sigma_1, \sigma_2, x$, and of an approximation precision ϵ, we can use the bisection method.

We fixe $\epsilon > 0$.
We set $u \in (0,1)$.
We define $L = -10$ and $U = -L$, the bounds and $M = \frac{L+U}{2}$, the middle of our interval.
While $U - L > \epsilon$:

- We compute $F(M) = \frac{1}{1-a}(F_1(M) - aF_2(M))$
- If $F(M) < u$, we set $L = M$
- Else, we set $U = M$
- We set $M = \frac{L+U}{2}$

End while

For the generation of random variables from F, we can use the inverse transform sampling method.

We set X as an empty array and n the number of random variables we want to generate.
We fixe $\epsilon > 0$.
For $i = 1, \ldots, n$:

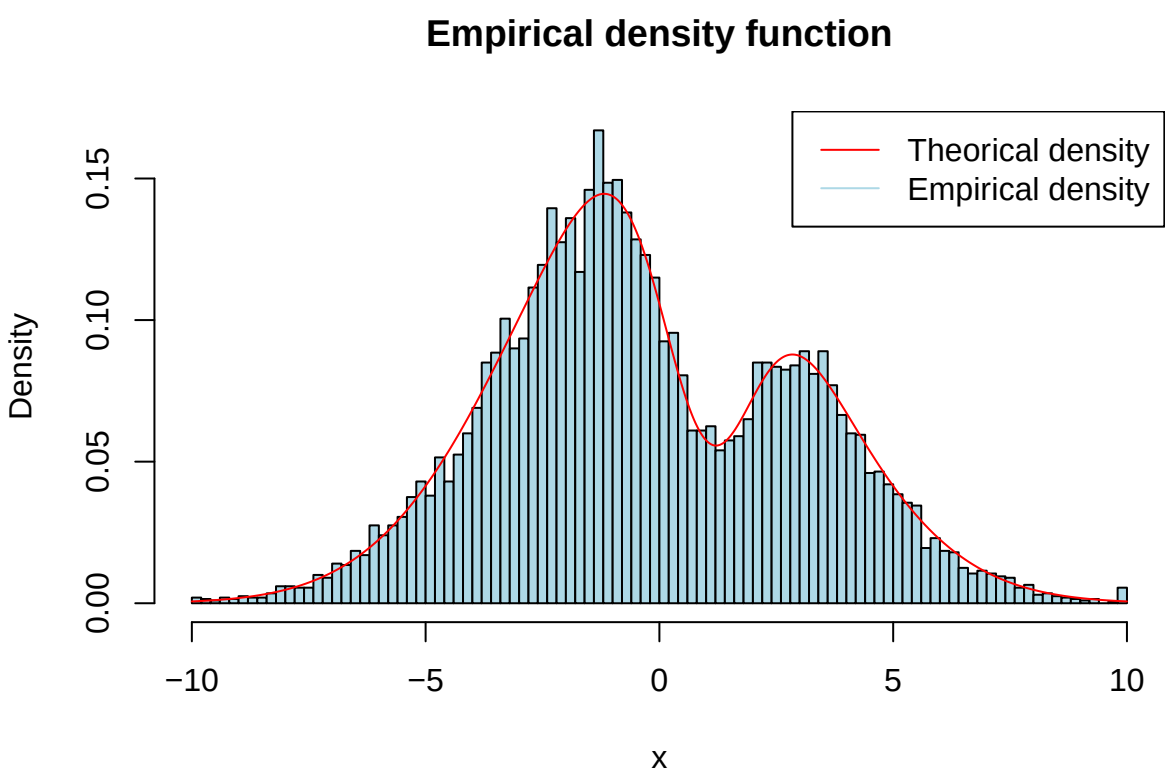
- We set $u \in (0,1)$.
- We define $L = -10$ and $U = -L$, the bounds and $M = \frac{L+U}{2}$, the middle of our interval.
- While $U - L > \epsilon$:
- We compute $F(M) = \frac{1}{1-a}(F_1(M) - aF_2(M))$
- If $F(M) < u$, we set $L = M$
- Else, we set $U = M$
- We set $M = \frac{L+U}{2}$
- End while
- We add M to X

End for We return X

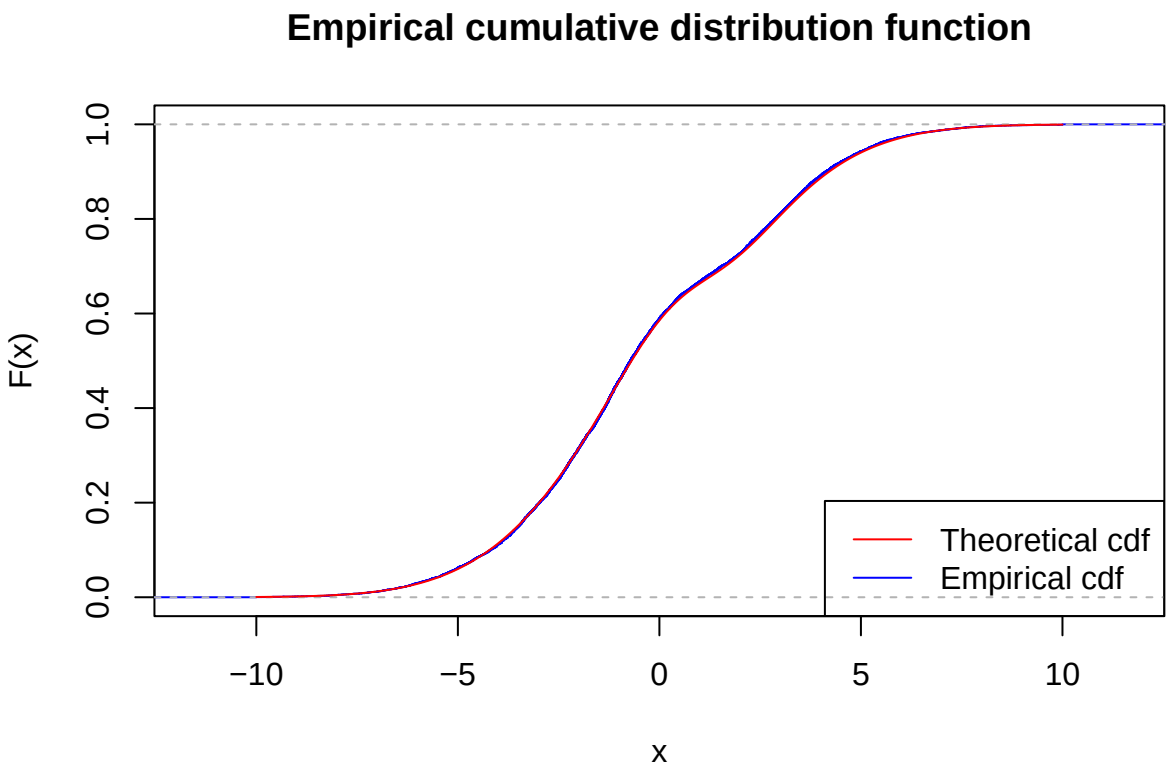
Question 5

```r
inv_cdf <- function(n) {
  X <- numeric(n)
  for (i in 1:n) {
    u <- runif(1)
    L <- -10
    U <- -L
    M <- (L + U) / 2
    while (U - L > 1e-6) {
      FM <- F(a, mu1, mu2, s1, s2, M)
      if (FM < u) {
        L <- M
      } else {
        U <- M
      }
      M <- (L + U) / 2
    }
    X[i] <- M
  }
  return(X)
}
```

```r
set.seed(123)
n <- 10000
X <- inv_cdf(n)
x <- seq(-10, 10, length.out = 1000)

hist(X, breaks = 100, freq = FALSE, col = "lightblue", main = "Empirical density function", xlab = "x")
lines(x, f(a, mu1, mu2, s1, s2, x), col = "red")
legend("topright", legend = c("Theorical density", "Empirical density"), col = c("red", "lightblue"), l
```




```
plot(ecdf(X), col = "blue", main = "Empirical cumulative distribution function", xlab = "x", ylab = "F(
lines(x, F(a, mu1, mu2, s1, s2, x), col = "red")
legend("bottomright", legend = c("Theoretical cdf", "Empirical cdf"), col = c("red", "blue"), lty = 1)
```



We can see of both graphs that the empirical cumulative distribution function is close to the theoretical one.

Accept-Reject Random Variable simulation

Question 6

To simulate under f using the accept-reject algorithm, we need to find a density function g such that $f(x) \leq Mg(x)$ for all $x \in \mathbb{R}$, where M is a constant.

Then, we generate $X \sim g$ and $U \sim \mathcal{U}([0,1])$.
We accept $Y = X$ if $U \leq \frac{f(X)}{Mg(X)}$. Return to 1 otherwise.

The probability of acceptance is $\int_{\mathbb{R}} \frac{f(x)}{Mg(x)} g(x)\,dx = \frac{1}{M}\int_{\mathbb{R}} f(x)\,dx = \frac{1}{M}$

Here we pose $g = f_1$.

Then we have $\frac{f(x)}{g(x)} = \frac{1}{1-a}(1 - a\frac{f_2(x)}{f_1(x)})$

We pose earlier that $a^* = \min_{x\in\mathbb{R}} \frac{f_1(x)}{f_2(x)} \Rightarrow \frac{1}{a^*} = \max_{x\in\mathbb{R}} \frac{f_2(x)}{f_1(x)}$.

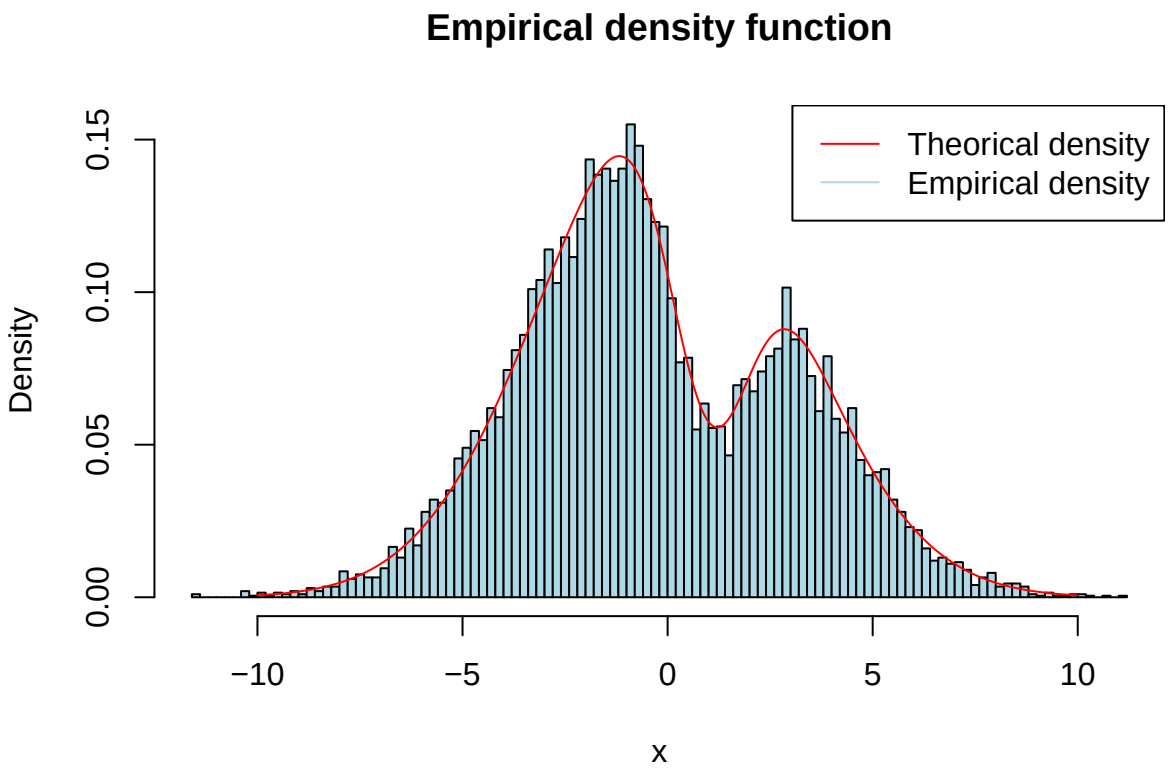
We compute in our fist equation : $\frac{1}{1-a}(1 - a\frac{f_2(x)}{f_1(x)}) \leq \frac{1}{1-a}(1 - \frac{a}{a^*}) \leq \frac{1}{1-a}$ because $a \leq a^* \Rightarrow 1 - \frac{a}{a^*} \leq 0$

To conclude, we have $M = \frac{1}{1-a}$ and the probability of acceptance is $\frac{1}{M} = 1 - a$

Question 7

```
accept_reject <- function(n, a) {
  X <- numeric(0)
  M <- 1 / (1 - a)

  while (length(X) < n) {
    Y <- rnorm(1, mu1, s1)
    U <- runif(1)

    if (U <= f(a, mu1, mu2, s1, s2, Y) / (M * dnorm(Y, mu1, s1))) {
      X <- append(X, Y)
    }
  }
  return(X)
}
```

```
set.seed(123)
n <- 10000
a <- 0.2
X <- accept_reject(n, a)
x <- seq(-10, 10, length.out = 1000)

hist(X, breaks = 100, freq = FALSE, col = "lightblue", main = "Empirical density function", xlab = "x")
lines(x, f(a, mu1, mu2, s1, s2, x), col = "red")
legend("topright", legend = c("Theorical density", "Empirical density"), col = c("red", "lightblue"), l
```



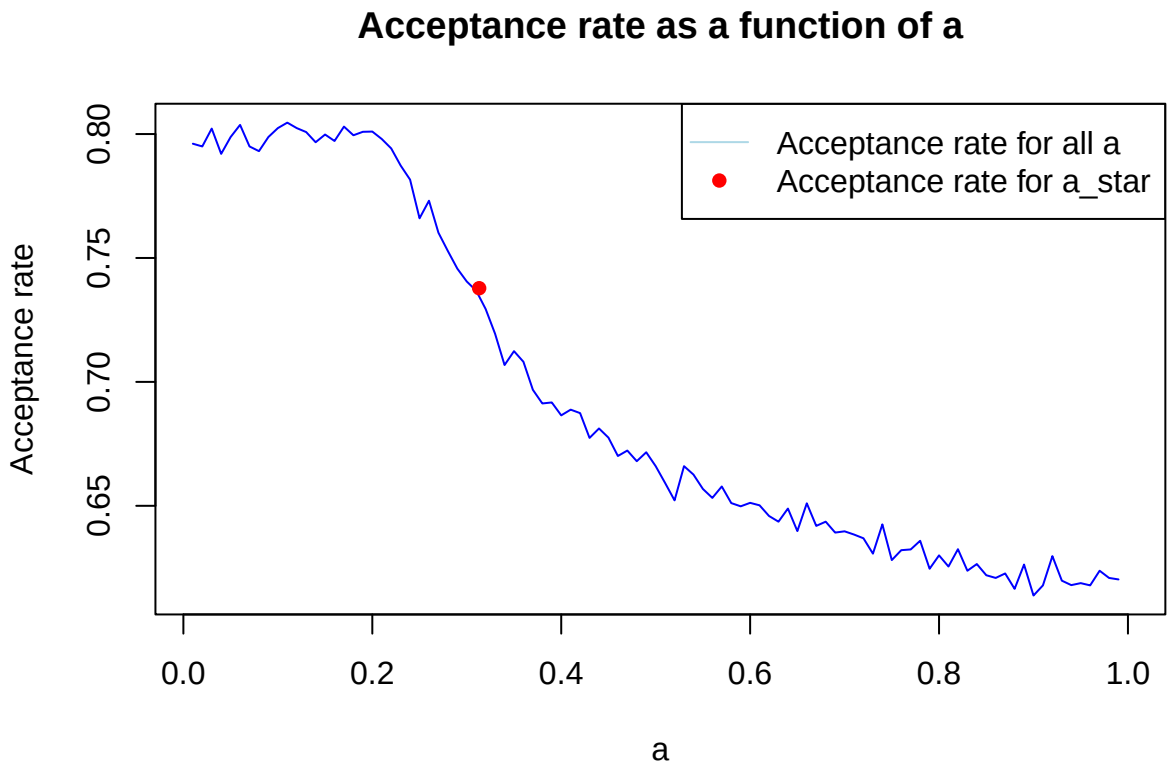

```r
set.seed(123)

acceptance_rate <- function(n, a = 0.2) {
  Y <- rnorm(n, mu1, s1)
  U <- runif(n)
  return(mean(U <= f(a, mu1, mu2, s1, s2, Y) / (M * dnorm(Y, mu1, s1))))
}

M <- 1 / (1 - a)
n <- 10000
cat(sprintf("[M = %.2f] Empirical acceptance rate: %f, Theoretical acceptance rate: %f \n", M, acceptan
```

```
## [M = 1.25] Empirical acceptance rate: 0.802600, Theoretical acceptance rate: 0.800000
```

Question 8

```r
set.seed(123)
a_values <- seq(0.01, 1, length.out = 100)
acceptance_rates <- numeric(length(a_values))
as <- a_star(mu1, mu2, s1, s2)

for (i in seq_along(a_values)) {
  acceptance_rates[i] <- acceptance_rate(n, a_values[i])
```

```r
}

plot(a_values, acceptance_rates, type = "l", col = "blue", xlab = "a", ylab = "Acceptance rate", main =
points(as, acceptance_rate(n, as), col = "red", pch = 16)
legend("topright", legend = c("Acceptance rate for all a", "Acceptance rate for a_star"), col = c("ligh
```



Random Variable simulation with stratification

Question 9

We consider a partition $\mathcal{P} = (D_0, D_1, ..., D_k)$, $k \in \mathbb{N}$ of $\mathbb{R}$ such that D_0 covers the tails of f_1 and f_1 is upper bounded and f_2 lower bounded in $D_1, \dots, D_k$.

To simulate under f using the accept-reject algorithm, we need to find a density function g such that $f(x) \leq Mg(x)$ for all $x \in \mathbb{R}$, where M is a constant.

We generate $X \sim g$ and $U \sim \mathcal{U}([0,1])$ *(1)*.
We accept $Y = X$ if $U \leq \frac{f(X)}{Mg(X)}$. Otherwise, we return to *(1)*.

Here we pose $g(x) = f_1(x)$ if $x \in D_0$ and $g(x) = sup_{D_i} f_1(x)$ if $x \in D_i, i \in \{1, \dots, k\}$.

To conclude, we have $M = \frac{1}{1-a}$ and the probability of acceptance is $r = \frac{1}{M} = 1 - a$

Question 10

Let $P_n = (D_0, \dots D_n)$ a partition of $\mathbb{R}$ for $n \in \mathbb{N}$. We have $\forall x \in P_n$ and $\forall i \in \{0, \dots, n\}$, $\lim_{n\to\infty} sup_{D_i} f_1(x) = f_1(x)$ and $\lim_{x\to\infty} inf_{D_i} f_2(x) = f_2(x)$.

$\Rightarrow \lim_{x\to\infty} g(x) = f(x)$

$\Rightarrow \lim_{x\to\infty} \frac{g(x)}{f(x)} = 1$ as $f(x) > 0$ as f is a density function.

$\Rightarrow \forall \epsilon > 0, \exists n_\epsilon \in \mathbb{N}$ such that $\forall n \geq n_\epsilon$, $M = sup_{x\in P_n} \frac{g(x)}{f(x)} < \epsilon$

$\Rightarrow r = \frac{1}{M} > \frac{1}{\epsilon} := \delta \in]0, 1]$ where r is the acceptance rate defined in the question 10.

Question 11

We recall the parameters and the functions of the problem.

```
mu1 <- 0
mu2 <- 1
s1 <- 3
s2 <- 1
a <- 0.2

f1 <- function(x) {
  dnorm(x, mu1, s1)
}

f2 <- function(x) {
  dnorm(x, mu2, s2)
}

f <- function(x) {
  fx <- f1(x) - a * f2(x)
  fx[fx < 0] <- 0
  return(fx / (1 - a))
}

f1_bounds <- c(mu1 - 3 * s1, mu1 + 3 * s1)
```

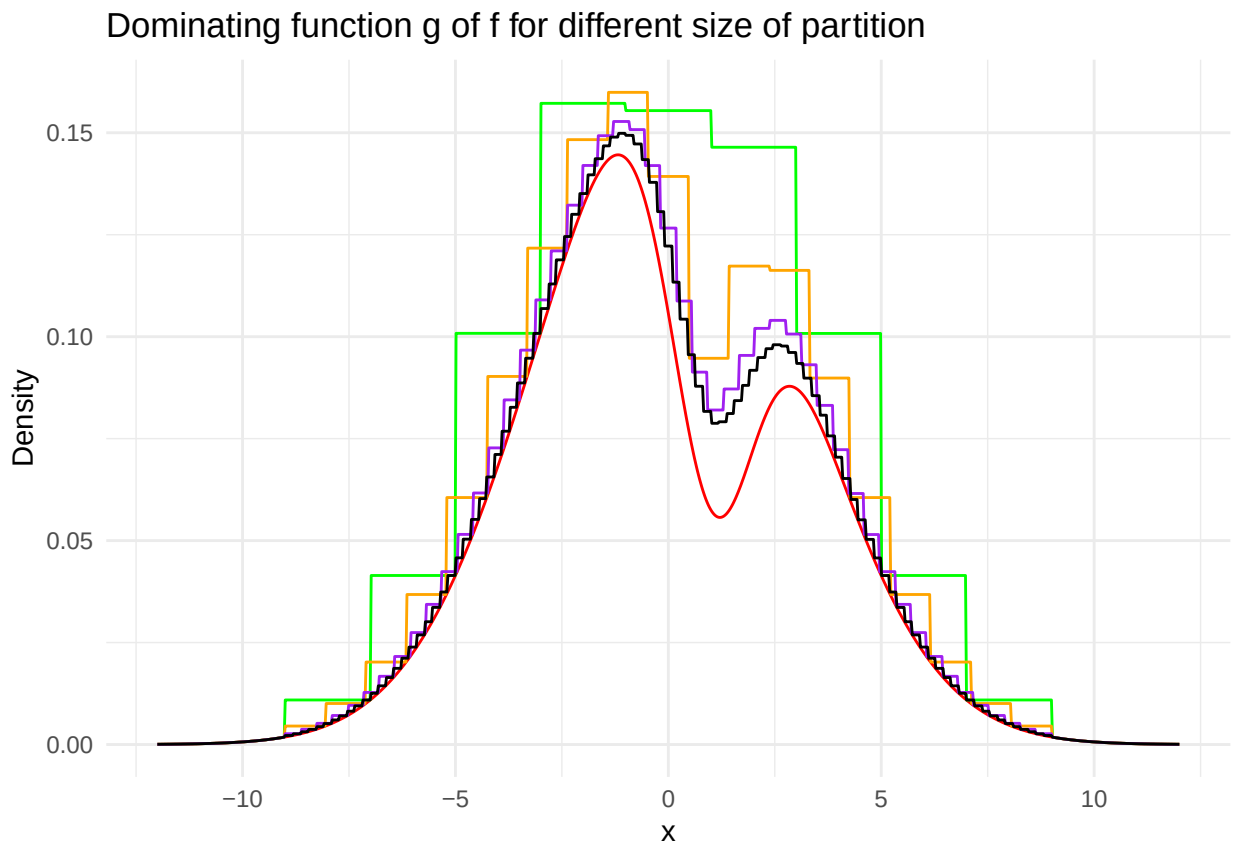
We implement the partition, the given g function to understand the behavior of g compared to f and the computation of the supremum and infimum of f_1 and f_2 on each partition.

```
create_partition <- function(k = 10) {
  return(seq(f1_bounds[1], f1_bounds[2], , length.out = k))
}

sup_inf <- function(f, P, i) {
  x <- seq(P[i], P[i + 1], length.out = 1000)
  f_values <- sapply(x, f)
  return(c(max(f_values), min(f_values)))
}

g <- function(X, P) {
  values <- numeric(0)
```

```
  for (x in X) {
    if (x <= P[1] | x >= P[length(P)]) {
      values <- c(values, 1 / (1 - a) * f1(x))
    } else {
      for (i in 1:(length(P) - 1)) {
        if (x >= P[i] & x <= P[i + 1]) {
          values <- c(values, 1 / (1 - a) * (sup_inf(f1, P, i)[1]) - a * sup_inf(f2, P, i)[2])
        }
      }
    }
  }
  return(values)
}
```

We plot the function f and the dominating function g for different sizes of the partition.

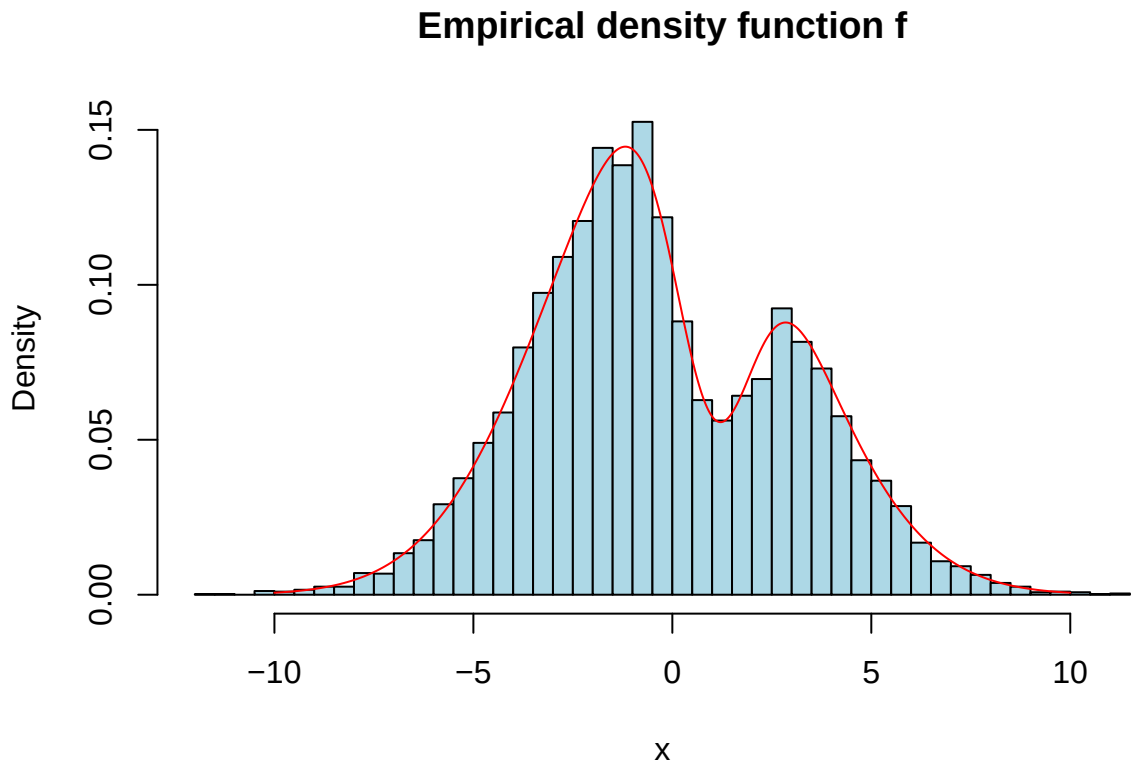
```
library(ggplot2)

X <- seq(-12, 12, length.out = 1000)

# Plot for different k with ggplot on same graph
k_values <- c(10, 20, 50, 100)
P_values <- lapply(k_values, create_partition)
g_values <- lapply(P_values, g, X)

ggplot() +
  geom_line(aes(x = X, y = f(X)), col = "red") +
  geom_line(aes(x = X, y = g(X, P_values[[1]])), col = "green") +
  geom_line(aes(x = X, y = g(X, P_values[[2]])), col = "orange") +
  geom_line(aes(x = X, y = g(X, P_values[[3]])), col = "purple") +
  geom_line(aes(x = X, y = g(X, P_values[[4]])), col = "black") +
  labs(title = "Dominating function g of f for different size of partition", x = "x", y = "Density") +
  theme_minimal()
```



Here, we implement the algorithm of accept-reject with the given partition and an appropriate function g to compute f.

```
set.seed(123)

g_accept_reject <- function(x, P) {
  if (x < P[1] | x >= P[length(P)]) {
    return(f1(x))
  } else {
    for (i in seq_along(P)) {
      if (x >= P[i] & x < P[i + 1]) {
        return(sup_inf(f1, P, i)[1])
      }
    }
  }
}

stratified <- function(n, P) {
  samples <- numeric(0)
  rate <- 0
  while (length(samples) < n) {
    x <- rnorm(1, mu1, s1)
    u <- runif(1)
    if (u <= f(x) * (1 - a) / g_accept_reject(x, P)) {
      samples <- c(samples, x)
    }
```

```
    rate <- rate + 1
  }
  list(samples = samples, acceptance_rate = n / rate)
}

n <- 10000
k <- 100

P <- create_partition(k)
samples <- stratified(n, P)
X <- seq(-10, 10, length.out = 1000)

hist(samples$samples, breaks = 50, freq = FALSE, col = "lightblue", main = "Empirical density function f
lines(X, f(X), col = "red")
```



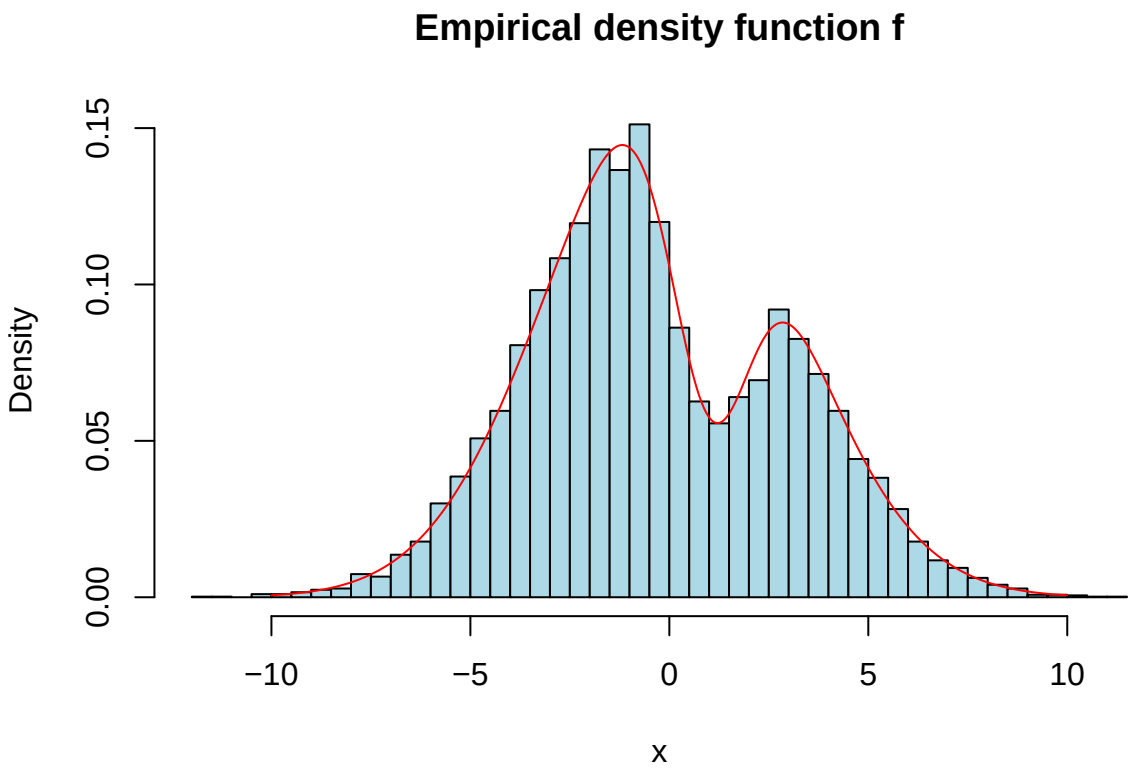
We also compute the acceptance rate of the algorithm.

```
theorical_acceptance_rate <- 1 - a
cat(sprintf("Empirical acceptance rate: %f, Theoretical acceptance rate: %.1f \n", samples$acceptance_ra
```

```
## Empirical acceptance rate: 0.784498, Theoretical acceptance rate: 0.8
```

Question 12

```r
set.seed(123)

stratified_delta <- function(n, delta) {
  samples <- numeric(0)
  P <- create_partition(n * delta)
  rate <- 0
  while (length(samples) < n | rate < delta * n) {
    x <- rnorm(1, mu1, s1)
    u <- runif(1)
    if (u <= f(x) * delta / g_accept_reject(x, P)) {
      samples <- c(samples, x)
    }
    rate <- rate + 1
  }
  list(samples = samples, partition = P, acceptance_rate = n / rate)
}

n <- 10000
delta <- 0.8

samples_delta <- stratified_delta(n, delta)
X <- seq(-10, 10, length.out = 1000)

hist(samples_delta$samples, breaks = 50, freq = FALSE, col = "lightblue", main = "Empirical density fun
lines(X, f(X), col = "red")
```



We also compute the acceptance rate of the algorithm.

```
theorical_acceptance_rate <- 1 - a
cat(sprintf("Empirical acceptance rate: %f, Theoretical acceptance rate: %f \n", samples_delta$acceptanc
```

```
## Empirical acceptance rate: 0.803213, Theoretical acceptance rate: 0.800000
```

Now, we will test the stratified_delta function for different delta:

```
set.seed(123)
n <- 1000
deltas <- seq(0.1, 1, by = 0.1)

for (delta in deltas) {
  samples <- stratified_delta(n, delta)
  cat(sprintf("Delta: %.1f, Empirical acceptance rate: %f \n", delta, samples$acceptance_rate))
}
```

```
## Delta: 0.1, Empirical acceptance rate: 0.097618
## Delta: 0.2, Empirical acceptance rate: 0.199840
## Delta: 0.3, Empirical acceptance rate: 0.304321
## Delta: 0.4, Empirical acceptance rate: 0.392311
## Delta: 0.5, Empirical acceptance rate: 0.503271
## Delta: 0.6, Empirical acceptance rate: 0.594177
## Delta: 0.7, Empirical acceptance rate: 0.692521
```

```
## Delta: 0.8, Empirical acceptance rate: 0.786782
## Delta: 0.9, Empirical acceptance rate: 0.848896
## Delta: 1.0, Empirical acceptance rate: 0.858369
```

Cumulative density function.

Question 13

The cumulative density function $\forall x \in \mathbb{R}\ F_X(x) = \int_{-\infty}^{x} f(t)\,dt = \int_{\mathbb{R}} f(t)h(t),dt$ where $h(x) = \mathbb{1}_{X_n \leq x}$

For a given $x \in \mathbb{R}$, a Monte Carlo estimator $F_n(x) = \frac{1}{n}\sum_{i=1}^{n} h(X_i)$ where h is the same function as above and $(X_i)_{i=1}^{n} \sim^{iid} X$

Question 14

As $X_1, \ldots, X_n$ are iid and follows the law of X, and h is continuous by parts, and positive, we have $h(X_1), \ldots h(X_n)$ are iid and $\mathbb{E}[h(X_i)] < +\infty$. By the law of large numbers, we have $F_n(X) = \frac{1}{n}\sum_{i=1}^{n} h(X_i) \xrightarrow{a.s} \mathbb{E}[h(X_1)] = F_X(x)$.

Moreover, $\forall \epsilon > 0$, $\exists N \in \mathbb{N}$ such that $\forall n \leq N$, $|F_n(x) - F_X(x)| < \epsilon$, ie, $sup_{x\in\mathbb{R}}|F_n(x) - F_X(x)| \xrightarrow{a.s} 0$, by Glivenko-Cantelli theorem.

Hence, F_n is a good estimate of F_X as a function of x.

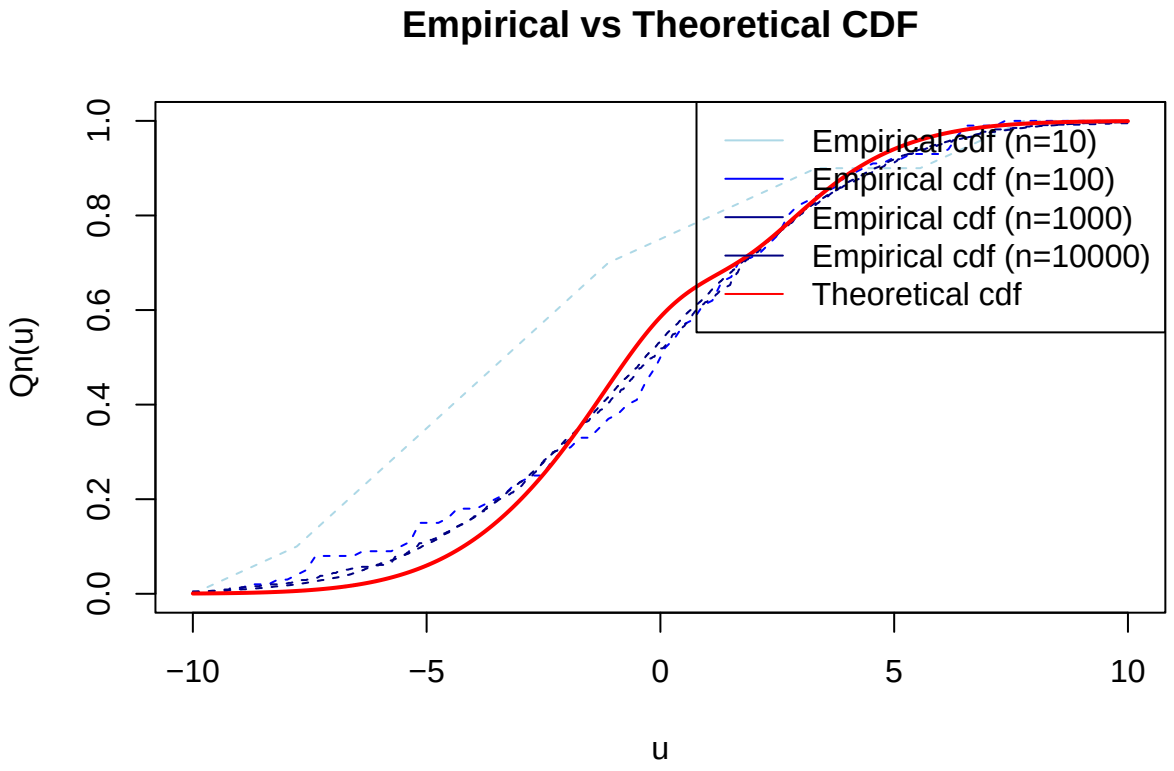
Question 15

```r
set.seed(123)
n <- 10000
Xn <- (rnorm(n, mu1, s1) - a * rnorm(n, mu2, s2)) / (1 - a)
X <- seq(-10, 10, length.out = n)

h <- function(x, Xn) {
  return(Xn <= x)
}

# Fn empirical
empirical_cdf <- function(x, Xn) {
  return(mean(h(x, Xn)))
}

# F theoretical
F <- function(x) {
  Fx <- pnorm(x, mu1, s1) - a * pnorm(x, mu2, s2)
  return(Fx / (1 - a))
}

cat(sprintf("Empirical cdf: %f, Theoretical cdf: %f \n", empirical_cdf(X, Xn), mean(F(Xn))))
```

```
## Empirical cdf: 0.508300, Theoretical cdf: 0.505263
```

Now we plot the empirical and theoretical cumulative density functions for different n.

```r
n_values <- c(10, 100, 1000, 10000)
colors <- c("lightblue", "blue", "darkblue", "navy")

plot(NULL, xlim = c(-10, 10), ylim = c(0, 1), xlab = "u", ylab = "Qn(u)", main = "Empirical vs Theoreti

for (i in seq_along(n_values)) {
  n <- n_values[i]
  X <- seq(-10, 10, length.out = n)
  Xn <- (rnorm(n, mu1, s1) - a * rnorm(n, mu2, s2)) / (1 - a)
  lines(X, sapply(X, empirical_cdf, Xn = Xn), col = colors[i], lt = 2)
}

lines(X, F(X), col = "red", lty = 1, lw = 2)
legend("topright", legend = c("Empirical cdf (n=10)", "Empirical cdf (n=100)", "Empirical cdf (n=1000)"
```



Question 16

As $X_1, \dots, X_n$ are iid and follows the law of X, and h is continuous and positive, we have $h(X_1), \dots h(X_n)$ are iid and $\mathbb{E}[h(X_i)^2] < +\infty$. By the Central Limit Theorem, we have $\sqrt{n}\frac{(F_n(x)-F_X(x))}{\sigma} \xrightarrow{d} \mathcal{N}(0,1)$ where $\sigma^2 = Var(h(X_1))$.

So we have $\lim_{x\to\infty} \mathbb{P}(\sqrt{n}\frac{(F_n(x)-F_X(x))}{\sigma} \leq q_{1-\frac{\alpha}{2}}^{\mathcal{N}(0,1)}) = 1-\alpha$

So by computing the quantile of the normal distribution, we can have a confidence interval for $F_X(x)$: $F_X(x) \in [F_n(x) - \frac{q_{1-\frac{\alpha}{2}}^{\mathcal{N}(0,1)}\sigma}{\sqrt{n}}; F_n(x) + \frac{q_{1-\frac{\alpha}{2}}^{\mathcal{N}(0,1)}\sigma}{\sqrt{n}}]$

```r
set.seed(123)
Fn <- empirical_cdf(X, Xn)
sigma <- sqrt(Fn - Fn^2)
q <- qnorm(0.975)
interval <- c(Fn - q * sigma / sqrt(n), Fn + q * sigma / sqrt(n))
cat(sprintf("Confidence interval: [%f, %f] \n", interval[1], interval[2]))
```

```
## Confidence interval: [0.501203, 0.520797]
```

Question 17

```r
compute_n_cdf <- function(x, interval_length = 0.01) {
  q <- qnorm(0.975)
  ceiling((q^2 * F(x) * (1 - F(x))) / interval_length^2)
}

x_values <- c(-15, -1)
n_values <- sapply(x_values, compute_n_cdf)

data.frame(x = x_values, n = n_values)
```

```
##     x    n
## 1 -15    1
## 2  -1 9530
```

We notice that the size of the sample needed to estimate the cumulative density function is higher for values of x that are close to the mean of the distribution. At $x = -1$, we are on the highest peek of the function and at $x = -15$ we are on the tail of the function.

Empirical quantile function

Question 18

We define the empirical quantile function defined on $(0, 1)$ by : $Q_n(u) = inf\{x \in \mathbb{R} : u \leq F_n(x)\}$. We recall the estimator $F_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{X_i \leq x}$

So we have $Q_n(u) = inf\{x \in \mathbb{R} : u \leq \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{X_i \leq x}\} = inf\{x \in \mathbb{R} : n.u \leq \sum_{i=1}^{n} \mathbb{1}_{X_i \leq x}\}$

We sort the sample $(X_1, \ldots, X_n)$ in increasing order, and we define $X_{(1)} \leq X_{(2)} \leq \cdots \leq X_{(n)}$ the order statistics of the sample.

As $\sum_{i=1}^{n} \mathbb{1}_{X_{(i)} \leq x} = k$ where $X_{(k)} = max\{i = 1, \ldots, n; X_{(i)} \leq x\}$

But as F_n is a step function, we can simplify the expression to $Q_n(u) = X_{(k)}$ where $k = \lceil n \cdot u \rceil$ and $X_{(k)}$ is the k-th order statistic of the sample $(X_1, \ldots, X_n)$.

Question 19

We note $Y_{j,n} := \mathbb{1}_{X_{n,j} < Q(u) + \frac{t}{\sqrt{n}} \frac{\sqrt{u(1-u)}}{f(Q(u))}}$

We know that $(X_{n,j})$ is iid as X is bounded in j and n.

Let $\Delta_n = \frac{t}{\sqrt{n}} \frac{\sqrt{u(1-u)}}{f(Q(u))}$. We have $F_n(X) = \frac{1}{n} \sum_{j=1}^{n} \mathbb{1}_{X_{n,j}} < Q(u) + \Delta_n$

then $\frac{1}{n} \sum_{j=1}^{n} Y_{j,n} = F_n(Q(u) + \Delta_n)$ by definition of the empirical quantile $F_n(Q_n(u)) = u$

By Taylor formula, we got $F_n(Q(u) + \Delta_n) = F_n(Q(u)) + \Delta_n f(Q(u))$

By Lindbergh-Levy Central Limit Theorem, applied to $F_n(Q(u))$ as $\mathbb{E}[F_n(Q(u))] = u < +\infty$ and $Var(F_n(Q(u))) = u(1-u) < +\infty$, we have $\frac{\sqrt{n}(F_n(Q(u)) - u)}{\sqrt{u(1-u)}} \to \mathcal{N}(0,1)$

then $F_n(Q(u)) = u + \frac{1}{\sqrt{n}} Z$ with $Z \sim \mathcal{N}(0, u(1-u))$

Thus $F_n(Q(u) + \Delta_n) = u + \frac{1}{\sqrt{n}} Z + \Delta_n f(Q(u))$

By substituting $Q_n(u) = Q(u) + \Delta_n$, we have

$$F_n(Q_n(u)) = F_n(Q(u) + \Delta_n) \Leftrightarrow u = u + \frac{1}{\sqrt{n}} Z + \Delta_n f(Q(u)) \Leftrightarrow \Delta_n = -\frac{1}{\sqrt{n}} \frac{Z}{f(Q(u))}$$

As $Q_n(u) = Q(u) + \Delta_n \Rightarrow \Delta_n = Q_n(u) - Q(u)$

Then we have

$$Q_n(u) - Q(u) = -\frac{1}{\sqrt{n}} \frac{Z}{f(Q(u))} \Leftrightarrow \sqrt{n}(Q_n(u) - Q(u)) = \frac{Z}{f(Q(u))} \Leftrightarrow \sqrt{n}(Q\_n(u) - Q(u)) \sim \mathcal{N}(0, \frac{u(1-u)}{f(Q(u))^2})$$

Question 20

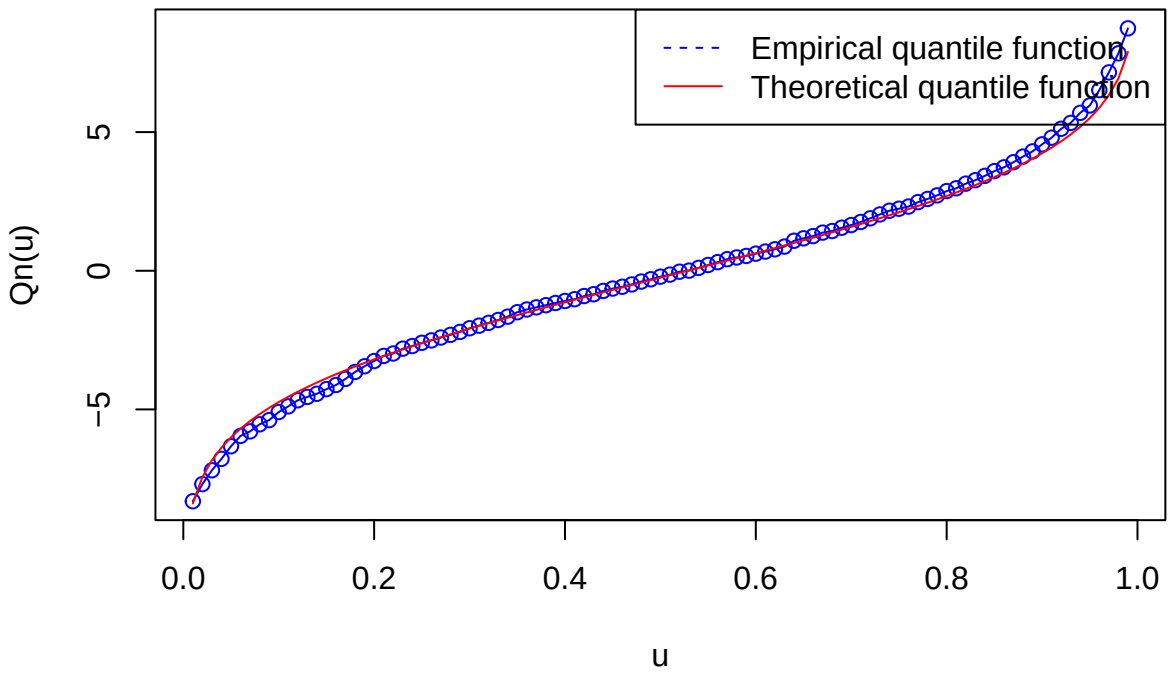
When $u \to 0$, $Q(u)$ corresponds to the lower tail of the distribution, and when $u \to 1$, $Q(u)$ corresponds to the upper tail of the distribution.

So $f(Q(u))$ is higher when u is close to 0 and close to 1, so the variance of the estimator is higher for values of u that are close to 0 and 1. So we need a higher sample size to estimate the quantile function for values of u that are close to 0 and 1.

Question 21

```r
set.seed(123)

empirical_quantile <- function(u, Xn) {
  sorted_Xn <- sort(Xn)
  k <- ceiling(u * length(Xn))
  sorted_Xn[k]
}
```

```r
set.seed(123)
n <- 1000
Xn <- (rnorm(n, mu1, s1) - a * rnorm(n, mu2, s2)) / (1 - a)
u_values <- seq(0.01, 0.99, by = 0.01)
Qn_values <- sapply(u_values, empirical_quantile, Xn = Xn)

plot(u_values, Qn_values, col = "blue", xlab = "u", ylab = "Qn(u)", type = "o")
lines(u_values, (qnorm(u_values, mu1, s1) - a * qnorm(u_values, mu2, s2)) / (1 - a), col = "red")
legend("topright", legend = c("Empirical quantile function", "Theoretical quantile function"), col = c(
```



Question 22

We can compute the confidence interval of the empirical quantile function using the Central Limit Theorem.

We obtain the following formula for the confidence interval of the empirical quantile function:

$$Q(u) \in [Q_n(u) - q_{1-\frac{\alpha}{2}}^{\mathcal{N}(0,1)} \frac{\sqrt{u(1-u)}}{\sqrt{n} f(Q(u))}; Q_n(u) + q_{1-\frac{\alpha}{2}}^{\mathcal{N}(0,1)} \frac{\sqrt{u(1-u)}}{\sqrt{n} f(Q(u))}]$$

```
f_q <- function(u) {
  f(1 / (1 - a) * (qnorm(u, mu1, s1) - a * qnorm(u, mu2, s2)))
}

compute_n_quantile <- function(u, interval_length = 0.01) {
  q <- qnorm(0.975)
  ceiling((q^2 * u * (1 - u)) / (interval_length^2 * f_q(u)^2))
}

u_values <- c(0.5, 0.9, 0.99, 0.999, 0.9999)
n_values <- sapply(u_values, compute_n_quantile)

data.frame(u = u_values, n = n_values)
```

```
##        u          n
## 1 0.5000      667059
## 2 0.9000      934418
```

```
## 3 0.9900    13944215
## 4 0.9990   338588623
## 5 0.9999 10183256354
```

We deduce that the size of the sample needed to estimate the quantile function is higher for values of u that are close to 1. This corresponds to the deduction made in question 20.

Quantile estimation Naïve Reject algorithm

Question 23

We generate a sample $(X_n)_{n\in\mathbb{N}} \sim^{iid} f_X$ the c.d.f. of X

We choose all the $i = 1, \ldots, n$ such that $X_i \in A$ and plug them in a new set $(Y_n)_{n\in\mathbb{N}}$

Then, the mean of Y_n simulate a random variable X conditional to the event $X \in A$ with $A \subset \mathbb{R}$

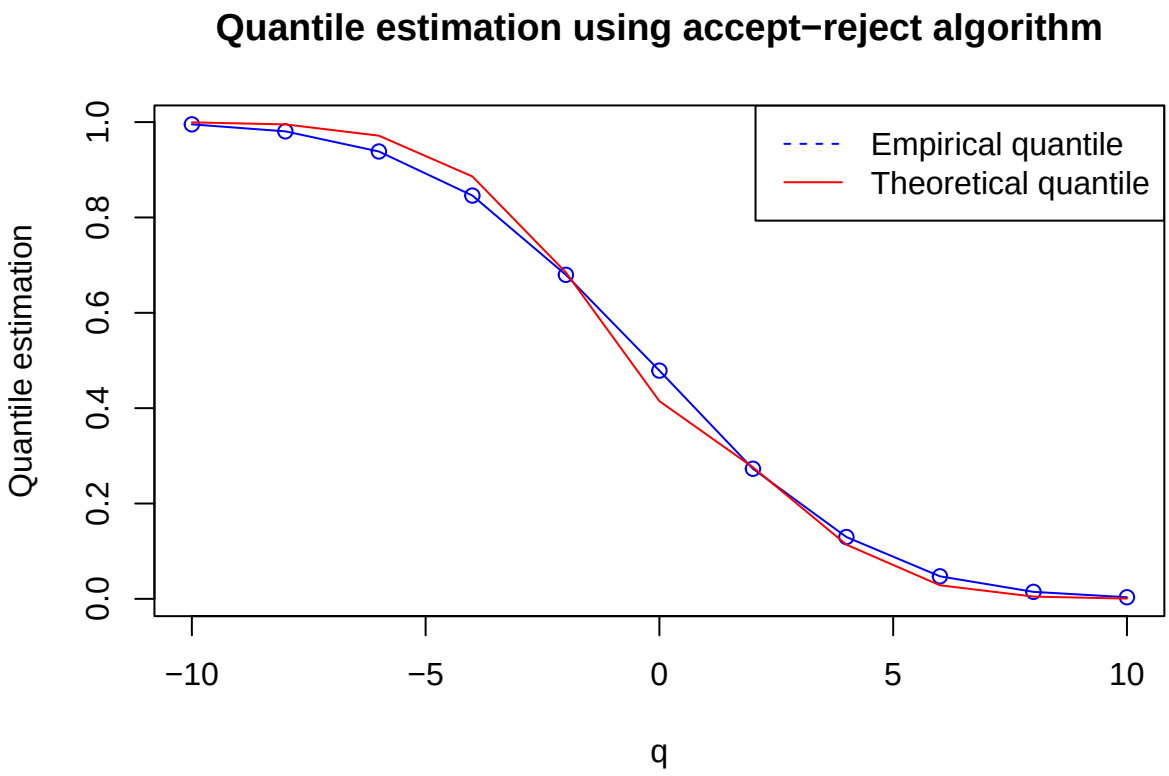
So when n is big enough, we can estimate $\mathbb{P}(X \in A)$ with our sample $(Y_n)_{n\in\mathbb{N}}$ and the mean of Y_n.

Question 24

```r
accept_reject_quantile <- function(q, n) {
  X_samples <- (rnorm(n, mu1, s1) - a * rnorm(n, mu2, s2)) / (1 - a)
  return(mean(X_samples >= q))
}

set.seed(123)
n <- 10000
q_values <- seq(-10, 10, by = 2)
delta_quantile <- sapply(q_values, accept_reject_quantile, n)
data.frame(q = q_values, quantile = delta_quantile)
```

```
##       q quantile
## 1  -10   0.9953
## 2   -8   0.9806
## 3   -6   0.9384
## 4   -4   0.8459
## 5   -2   0.6797
## 6    0   0.4787
## 7    2   0.2729
## 8    4   0.1297
## 9    6   0.0473
## 10   8   0.0147
## 11  10   0.0034
```

```r
plot(q_values, delta_quantile, type = "o", col = "blue", xlab = "q", ylab = "Quantile estimation", main
lines(q_values, 1 - (pnorm(q_values, mu1, s1) - a * pnorm(q_values, mu2, s2)) / (1 - a), col = "red")
legend("topright", legend = c("Empirical quantile", "Theoretical quantile"), col = c("blue", "red"), lt
```



Question 25

Let $X_1, \dots, X_n$ iid such that $\mathbb{E}[X_i] < \infty$, we have $\mathbb{1}_{X_1 \geq q}, \dots, \mathbb{1}_{X_n \geq q}$ and $\mathbb{E}[\mathbb{1}_{X_1 \geq q}] = \mathbb{P}(X_i \geq q) \leq 1 < +\infty$. By TCL, we have $\sqrt{n}\frac{(\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{X_i \geq q} - \mathbb{E}[\mathbb{1}_{X_1 \geq q}])}{\sqrt{Var(\mathbb{1}_{X_1 \geq q})}} \to Z \sim \mathcal{N}(0,1)$ in distribution, with

- $\mathbb{E}[\mathbb{1}_{X_1 \geq q}] = \mathbb{P}(X_i \geq q) = \delta$
- $\sqrt{Var(\mathbb{1}_{X_1 \geq q})} = \mathbb{E}[\mathbb{1}^2_{X_1 \geq q}] - \mathbb{E}[\mathbb{1}_{X_1 \geq q}]^2 = \delta(1-\delta)$

In addition, we have $\hat{\delta}_n^{NR} \to \delta$ a.s. by the LLN, as it is a consistent estimator of δ .

The function $x \mapsto \sqrt{\frac{\delta(1-\delta)}{x(1-x)}}$ is continuous on $(0,1)$, we have $\sqrt{\frac{\delta(1-\delta)}{\hat{\delta}_n^{NR}(1-\hat{\delta}_n^{NR})}} \to \sqrt{\frac{\delta(1-\delta)}{\delta(1-\delta)}} = 1$, then by Slutsky, we have $\sqrt{n}\frac{(\hat{\delta}_n^{NR}-\delta)}{\sqrt{\delta(1-\delta)}}\sqrt{\frac{\delta(1-\delta)}{\hat{\delta}_n^{NR}(1-\hat{\delta}_n^{NR})}} = \sqrt{n}\frac{(\hat{\delta}_n^{NR}-\delta)}{\sqrt{\hat{\delta}_n^{NR}(1-\hat{\delta}_n^{NR}))}} \to Z.1 = Z \sim \mathcal{N}(0,1)$.

Now we can compute the confident interval for δ: $IC_n(\delta) = [\hat{\delta}_n^{NR} - \frac{q_{1-\frac{\alpha}{2}}^{\mathcal{N}(0,1)}}{\sqrt{n}}\hat{\delta}_n^{NR}(1-\hat{\delta}_n^{NR}); \hat{\delta}_n^{NR} + \frac{q_{1-\frac{\alpha}{2}}^{\mathcal{N}(0,1)}}{\sqrt{n}}\hat{\delta}_n^{NR}(1-\hat{\delta}_n^{NR})]$ where $q_{1-\frac{\alpha}{2}}^{\mathcal{N}(0,1)}$ is the quantile of the normal distribution $\mathcal{N}(0,1)$

```r
IC_Naive <- function(delta_hat, n, alpha = 0.05) {
  q <- qnorm(1 - alpha / 2)
  c(lower = delta_hat - q * sqrt(delta_hat * (1 - delta_hat)) / sqrt(n),
    upper = delta_hat + q * sqrt(delta_hat * (1 - delta_hat)) / sqrt(n))
}
```

```r
set.seed(123)
n <- 10000
q_values <- seq(-10, 10, by = 2)
delta_quantile_naive <- sapply(q_values, accept_reject_quantile, n)
IC_values_naive <- sapply(delta_quantile_naive, IC_Naive, n = n)
data.frame(q = q_values, quantile = delta_quantile_naive, IC = t(IC_values_naive))
```

```
##       q quantile    IC.lower    IC.upper
## 1   -10   0.9953 0.993959478 0.996640522
## 2    -8   0.9806 0.977896696 0.983303304
## 3    -6   0.9384 0.933687705 0.943112295
## 4    -4   0.8459 0.838823656 0.852976344
## 5    -2   0.6797 0.670554969 0.688845031
## 6     0   0.4787 0.468909076 0.488490924
## 7     2   0.2729 0.264169343 0.281630657
## 8     4   0.1297 0.123115049 0.136284951
## 9     6   0.0473 0.043139393 0.051460607
## 10    8   0.0147 0.012341201 0.017058799
## 11   10   0.0034 0.002259099 0.004540901
```

Importance Sampling

Question 26

Let $X_1, \ldots, X_n \sim^{iid} g$ where g such that $sup(f) \subseteq sup(g)$ and g density easy to simulate.

We want to determine $\delta = \mathbb{P}(X \geq q) = \int_q^{+\infty} f_X(x)\,dx = \int_{\mathbb{R}} f(x)h(x)\,dx = \mathbb{E}[h(X)]$ for any q, with $h(x) = \mathbb{1}_{x \geq q}$ and f the density function of X.

Given $X_1, \ldots, X_n \sim^{iid} g$, we have $\hat{\delta}_n^{IS} = \frac{1}{n}\sum_{i=1}^n \frac{f(X_i)}{g(X_i)} h(X_i)$

So $\hat{\delta}_n^{IS}$ is an unbiased estimator of δ, since $sup(f) \subseteq sup(g)$.

The importance sampling estimator is preferred to the classical Monte Carlo estimator when the variance of the importance sampling estimator is lower than the variance of the classical Monte Carlo estimator. This is the case when q is located on the tails of the distribution f

Question 27

The density g of a Cauchy distribution for parameters (μ_0, γ) is given by: $g(x; \mu_0, \gamma) = \frac{1}{\pi} \frac{\gamma}{(x-\mu_0)^2+\gamma^2}$, $\forall x \in \mathbb{R}$.

The parameters (μ_0, γ) of the Cauchy distribution used in importance sampling are chosen based on the characteristics of the target density $f(x)$.

- μ_0 is chosen such that $sup(f) \subseteq sup(g)$, where $sup(f)$ is the support of the target density $f(x)$. μ_0 should be chosen to place the center of $g(x)$ in the most likely region of $f(x)$, which can be approximated by the midpoint between μ_1 and μ_2, ie, $\mu_0 = \frac{\mu_1+\mu_2}{2} = \frac{0+1}{2} = \frac{1}{2}$.
- By centering $g(x)$ at μ_0 and setting γ to capture the spread of $f(x)$, we maximize the overlap between $f(x)$ and $g(x)$, reducing the variance of the importance sampling estimator. To cover the broader spread of $f(x)$, γ should reflect the scale of the wider normal component. A reasonable choice is to set γ to the largest standard deviation, ie, $\gamma = max(s_1, s_2) = max(1, 3) = 3$

Question 28

We can compute the confidence interval of the importance sampling estimator using the Central Limit Theorem, with the same arguments as in the question 25, since we have a consistent estimator of δ.

```
mu0 <- 0.5
gamma <- 3
g <- function(x) {
  dcauchy(x, mu0, gamma)
}

IS_quantile <- function(q, n) {
  X <- rcauchy(n, mu0, gamma)
  w <- f(X) / g(X)
  h <- (X >= q)
  return(mean(w * h))
}
```

```
IC_IS <- function(delta_hat, n, alpha = 0.05) {
  q <- qnorm(1 - alpha / 2)
  c(lower = delta_hat - q * sqrt(delta_hat * (1 - delta_hat)) / sqrt(n),
    upper = delta_hat + q * sqrt(delta_hat * (1 - delta_hat)) / sqrt(n))
}

set.seed(123)
q_values <- seq(-10, 10, by = 2)
n <- 10000
delta_quantile_IS <- sapply(q_values, IS_quantile, n = n)
IC_values_IS <- sapply(delta_quantile_IS, IC_IS, n = n)
data.frame(q = q_values, quantile = delta_quantile_IS, IC = t(IC_values_IS))
```

```
##       q     quantile     IC.lower     IC.upper
## 1   -10 0.9947877223 9.933764e-01 0.9961990475
## 2    -8 0.9874385973 9.852558e-01 0.9896214388
## 3    -6 0.9855259473 9.831851e-01 0.9878668168
## 4    -4 0.8659474971 8.592697e-01 0.8726252652
## 5    -2 0.6846463650 6.755393e-01 0.6937534662
## 6     0 0.4197530094 4.100802e-01 0.4294257925
## 7     2 0.2777399916 2.689616e-01 0.2865183657
## 8     4 0.1133957335 1.071812e-01 0.1196103074
## 9     6 0.0268130859 2.364702e-02 0.0299791493
## 10    8 0.0050713958 3.679176e-03 0.0064636156
## 11   10 0.0005414626 8.551521e-05 0.0009974099
```

Control Variate

Question 29

Let $X_1, \ldots, X_n$ iid. For f density with parameter θ, we can compute the score by deriving the partial derivative of the log-likelihood function. $S(\theta) = \frac{\partial l(\theta)}{\partial \theta} = \sum_{i=1}^{n} \frac{\partial log(f(X_i|\theta)}{\partial \theta} = \frac{f_1(x|\theta_1)}{f_1(x|\theta_1) - a f_2(x|\theta_2)} \frac{x-\mu_1}{\sigma_1^2}$

Question 30

We recall $\delta = \mathbb{P}(X \geq q) = \int_q^{+\infty} f_X(x)\,dx = \int_{\mathbb{R}} f_X(x)h(x)\,dx$ where $h(x) = \mathbb{1}_{x \geq q}$. We denote $\hat{\delta}_n^{IC} = \frac{1}{n}\sum_{i=1}^n h(X_i) - b \cdot (S_{\mu_1}(X_i) - m) = \frac{1}{n}\sum_{i=1}^n \mathbb{1}_{X_i \geq q} - b \cdot S_{\mu_1}(X_i)$ where $m = \mathbb{E}[S_{\mu_1}(X_1)] = 0$ and $b = \frac{Cov(\mathbb{1}_{X_1 \geq q}, S_{\mu_1}(X_1))}{Var(S_{\mu_1(X_1)})}$

Question 31

```
CV_quantile <- function(q, n) {
  X <- rnorm(n, mu1, s1)
  S <- (f1(X) / (f1(X) - a * f2(X))) * (X - mu1) / s1^2
  h <- (X >= q)
  b <- cov(S, h) / var(S)
  delta <- mean(h) - b * mean(S)
  return(delta)
}
```

We can compute the confidence interval of the importance sampling estimator using the Central Limit Theorem, with the same arguments as in the question 25, since we have a consistent estimator of δ.

```
IC_CV <- function(delta_hat, n, alpha = 0.05) {
  q <- qnorm(1 - alpha / 2)
  c(lower = delta_hat - q * sqrt(delta_hat * (1 - delta_hat)) / sqrt(n),
    upper = delta_hat + q * sqrt(delta_hat * (1 - delta_hat)) / sqrt(n))
}

set.seed(123)
n <- 10000
q_values <- seq(-10, 10, by = 2)
delta_quantile_CV <- sapply(q_values, CV_quantile, n)
IC_values_CV <- sapply(delta_quantile_CV, IC_CV, n = n)

data.frame(q = q_values, quantile = delta_quantile_CV, IC = t(IC_values_CV))
```

```
##       q    quantile     IC.lower     IC.upper
## 1   -10 0.999434937 9.989692e-01 0.9999007096
## 2    -8 0.995229234 9.938787e-01 0.9965797627
## 3    -6 0.972395901 9.691848e-01 0.9756070203
## 4    -4 0.888190843 8.820144e-01 0.8943673075
## 5    -2 0.704653322 6.957120e-01 0.7135946473
## 6     0 0.444051413 4.343131e-01 0.4537896882
## 7     2 0.215470961 2.074126e-01 0.2235293270
## 8     4 0.074929567 6.976942e-02 0.0800897146
## 9     6 0.017483243 1.491445e-02 0.0200520359
## 10    8 0.002294406 1.356662e-03 0.0032321503
## 11   10 0.000498199 6.083674e-05 0.0009355613
```

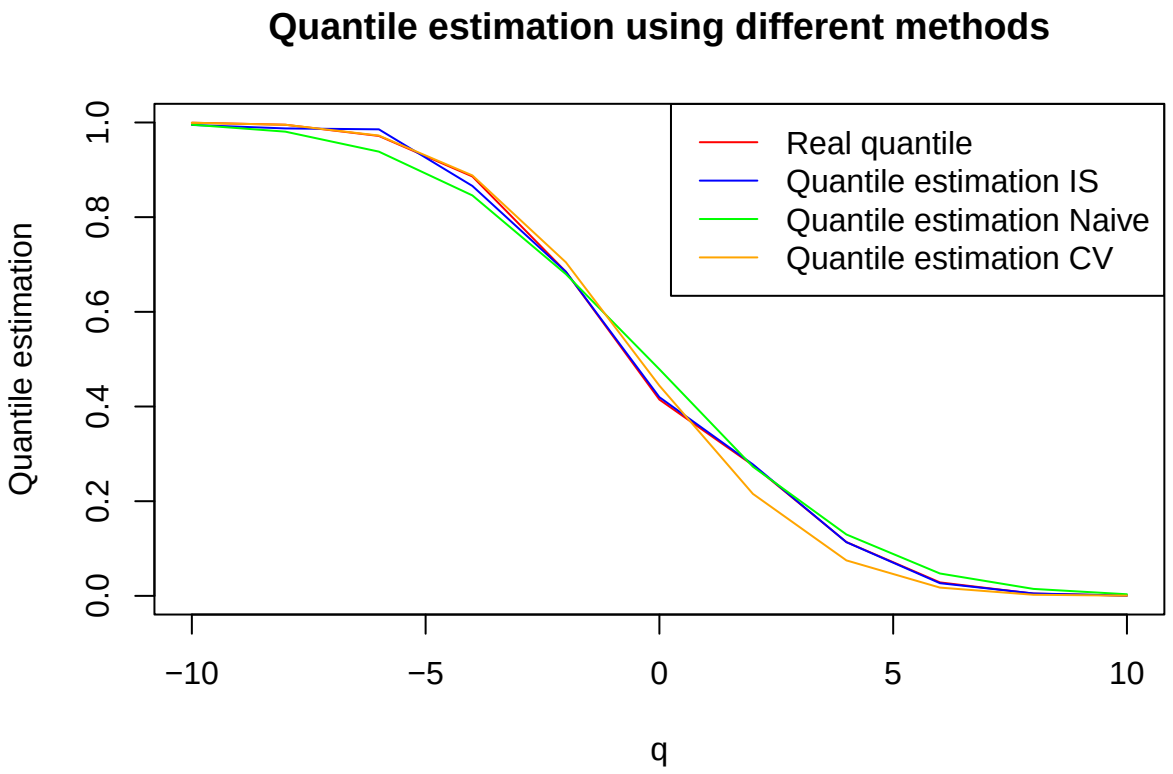
Question 32

```r
set.seed(123)
delta_real <- 1 - (pnorm(q_values, mu1, s1) - a * pnorm(q_values, mu2, s2)) / (1 - a)

IS <- data.frame(IC = t(IC_values_IS), length = IC_values_IS[2] - IC_values_IS[1])
naive <- data.frame(IC = t(IC_values_naive), length = IC_values_naive[2] - IC_values_naive[1])
CV <- data.frame(IC = t(IC_values_CV), length = IC_values_CV[2] - IC_values_CV[1])

data.frame(q = q_values, real_quantile = delta_real, quantile_CV = CV, quantile_IS = IS, quantile_Naive
```

```
##      q real_quantile quantile_CV.IC.lower quantile_CV.IC.upper
## 1  -10  0.9994636746         9.989692e-01         0.9999007096
## 2   -8  0.9952120243         9.938787e-01         0.9965797627
## 3   -6  0.9715623351         9.691848e-01         0.9756070203
## 4   -4  0.8859860470         8.820144e-01         0.8943673075
## 5   -2  0.6847218026         6.957120e-01         0.7135946473
## 6    0  0.4146638135         4.343131e-01         0.4537896882
## 7    2  0.2759518585         2.074126e-01         0.2235293270
## 8    4  0.1136765501         6.976942e-02         0.0800897146
## 9    6  0.0284375933         1.491445e-02         0.0200520359
## 10   8  0.0047879757         1.356662e-03         0.0032321503
## 11  10  0.0005363254         6.083674e-05         0.0009355613
##    quantile_CV.length quantile_IS.IC.lower quantile_IS.IC.upper
## 1        0.0009315447         9.933764e-01         0.9961990475
## 2        0.0009315447         9.852558e-01         0.9896214388
## 3        0.0009315447         9.831851e-01         0.9878668168
## 4        0.0009315447         8.592697e-01         0.8726252652
## 5        0.0009315447         6.755393e-01         0.6937534662
## 6        0.0009315447         4.100802e-01         0.4294257925
## 7        0.0009315447         2.689616e-01         0.2865183657
## 8        0.0009315447         1.071812e-01         0.1196103074
## 9        0.0009315447         2.364702e-02         0.0299791493
## 10       0.0009315447         3.679176e-03         0.0064636156
## 11       0.0009315447         8.551521e-05         0.0009974099
##    quantile_IS.length quantile_Naive.IC.lower quantile_Naive.IC.upper
## 1          0.00282265              0.993959478              0.996640522
## 2          0.00282265              0.977896696              0.983303304
## 3          0.00282265              0.933687705              0.943112295
## 4          0.00282265              0.838823656              0.852976344
## 5          0.00282265              0.670554969              0.688845031
## 6          0.00282265              0.468909076              0.488490924
## 7          0.00282265              0.264169343              0.281630657
## 8          0.00282265              0.123115049              0.136284951
## 9          0.00282265              0.043139393              0.051460607
## 10         0.00282265              0.012341201              0.017058799
## 11         0.00282265              0.002259099              0.004540901
##    quantile_Naive.length
## 1            0.002681044
## 2            0.002681044
## 3            0.002681044
## 4            0.002681044
## 5            0.002681044
## 6            0.002681044
## 7            0.002681044
```

```
## 8            0.002681044
## 9            0.002681044
## 10           0.002681044
## 11           0.002681044
```

```r
plot(q_values, delta_real, type = "l", col = "red", xlab = "q", ylab = "Quantile estimation", main = "Qu
lines(q_values, delta_quantile_IS, col = "blue")
lines(q_values, delta_quantile_naive, col = "green")
lines(q_values, delta_quantile_CV, col = "orange")
legend("topright", legend = c("Real quantile", "Quantile estimation IS", "Quantile estimation Naive", "(
```



Now we compare the three methods: Naive vs Importance Sampling vs Control Variate

The naive method is the easiest to implement but is the least precise for some points. The importance sampling method is more precise than the naive method but requires the choice of a good density g, that can be hard to determine in some case. The control variate method is the most precise but requires the choice of a good control variate, and need to compute the covariance, which require more computation time.

In our case, the control variate method is the most precise, but the importance sampling method is also a good choice.