Laboration report in Computational Statistics

Laboration 3

732A90

Duc Tran
William Wiik

Division of Statistics and Machine Learning
Department of Computer Science
Linköping University

20 november 2023

Contents

1 Question 1: Sampling algorithms for a triangle distribution

Consider the following density with a triangle-shape:

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

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

1.1 1 a)

Question

Choose an appropriate and simple envelope $e(x)$ for the density and program a random generator for X using rejection sampling.

Answer

We chose to $e(x)$ to $2 \cdot g(x) = 2 \cdot Unif(-1, 1)$ because it covers $f(x)$ as seen in the figure 1.

```
f <- function(x){
  1 - abs(x)
}


ggplot() + xlim(-1,1) + geom_function(fun = f, aes(color = "f(x)")) +
  geom_function(fun = function(x){1}, aes(color = "e(x)")) +
  theme_bw() +
  scale_color_manual(values = c("indianred","steelblue"), name = "Function")
```

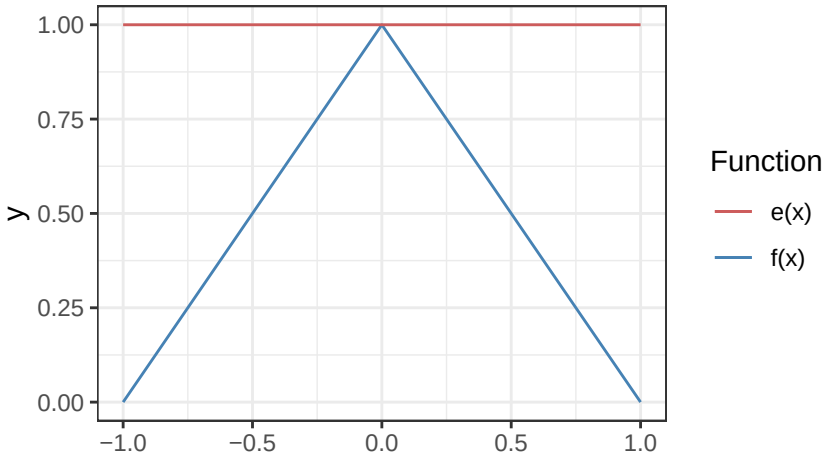

Figure 1: Envelope (e(x)) 2*Unif(-1,1) and the triangle distribution (f(x)).

Rejection sampling algorithm:

1. Sample $Y \sim g(x) = Unif(-1, 1)$

2. Sample $U \sim Unif(0,1)$

3. If $U \leq \frac{f(Y)}{e(Y)}$ accept Y; set $X = Y$; otherwise reject it

4. If more samples desired go to 1.

$$e(x) = 2 \cdot Unif(-1,1) = 2 \cdot \frac{1}{b-a} = \frac{2}{1-(-1)} = 1$$

$$f(x) = \begin{cases} 0 & \text{if } x < -1 \text{ or } x > 1, \\ x+1 & \text{if } -1 \leq x \leq 0, \\ 1-x & \text{if } 0 < x \leq 1 \end{cases} \Rightarrow \begin{cases} 0 & \text{if } x < -1 \text{ or } x > 1, \\ 1-|x| & \text{if } -1 \leq x \leq 1 \end{cases} \tag{2}$$

```
set.seed(13)


random_generator_a <- function(n) {

  g <- runif(n, -1,1)
  e <- 1
  f <- 1 - abs(g)

  U <- runif(n, 0, 1)

  accepted <- g[U<= (f/e)]
  return(accepted)
}
```

1.2 1 b)

Question

Let Y be a random variable following this distribution:

$$F(x) = \begin{cases} 0 & \text{if } x < 0 \text{ or } x > 1, \\ x+1 & \text{if } 0 \leq x \leq 1, \\ 1-x & \text{if } x > 1 \end{cases} \tag{3}$$

The inverse distribution function is

$$F^{-1}(y) = 1 - \sqrt{1-y}$$

$$\text{since } y = 2x - x^2 \iff x^2 - 2x + y = 0 \iff x_{1,2} = 1 \pm \sqrt{1-y} \Longrightarrow 1 - \sqrt{1-y}$$

A random variable -Y has a triangle distribution on the interval [-1, 0]. Program a random generator for X using composition sampling based on Y and -Y . You can use the code from the lecture to generate Y.

Answer

```r
random_generator_b <- function(n) {

  # Sample which distribution to use
  # sign will decide which function to sample from (Y or -Y)
  mu <- c(-1, 1)
  prob <- c(0.5, 0.5)
  sign <- sample(mu, n, replace = TRUE, p = prob)

  U <- runif(n, 0, 1)
  y <- sign * (1 - sqrt(1 - U))

  return(y)

}
```

1.3 1 c)

Question

Sums or differences of two independent uniformly distributed variables can also have some triangle distribution. When U_1, U_2 are two independent $Unif[0, 1]$ random variables, U_1 - U_2 has the same distribution as X. Use this result to program a generator for X.

Answer

```r
random_generator_c <- function(n) {

  U1 <- runif(n,0, 1)
  U2 <- runif(n,0, 1)

  X <- U1-U2
  return(X)

}
```

1.4 1 d)

Question

Check your random generators in each of a. to c. by generating 10000 random variables and plotting a histogram.

Answer

```r
library(ggplot2)
library(cowplot)
set.seed(13)
# Sample 22000 to make sure we get at least 10000 random numbers
data_a <- random_generator_a(22000)
# Remove excess observations so we only have 10000 left
data_a <- data_a[1:10000]
data_b <- random_generator_b(10000)
data_c <- random_generator_c(10000)
```

```r
p1 <- ggplot(as.data.frame(data_a), aes(x = data_a)) +
  geom_histogram(color="black", fill="lightblue") +
  theme_bw()


p2 <- ggplot(as.data.frame(data_b), aes(x = data_b)) +
  geom_histogram(color="black", fill="lightblue") +
  theme_bw()


p3 <- ggplot(as.data.frame(data_c), aes(x = data_c)) +
  geom_histogram(color="black", fill="lightblue") +
  theme_bw()


plot_grid(p1, p2,p3 ,labels=c("a", "b","c"), ncol = 2, nrow = 2)
```

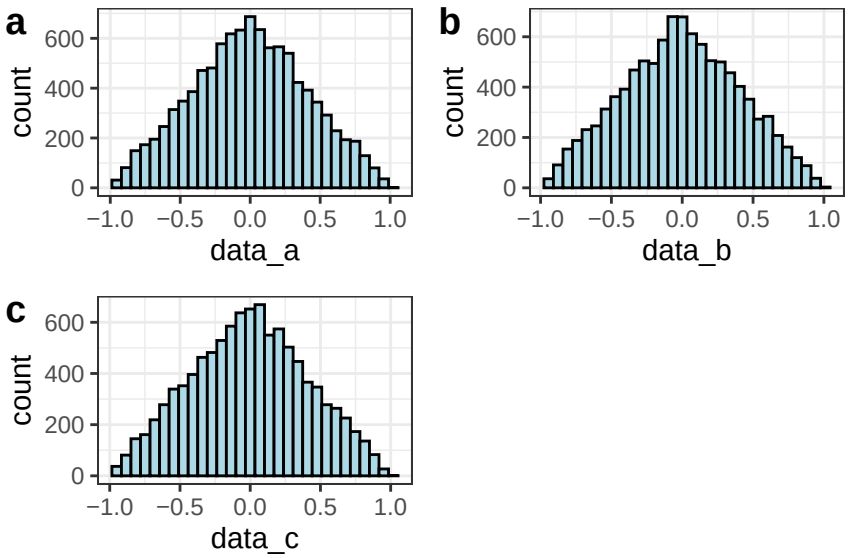

Figure 2: The different ways to generate the triangle distribution from a-c.

Question

Which of the three methods do you prefer if you had to generate samples of X? Use the data from one method to determine the variance of X.

Answer

```r
# Variance from rejection sampling in 1 a)
var(data_a)
```

```
## [1] 0.1655399
```

The variance is 0.17 from the a random generator using rejection sampling in 1 a). If we would choose one

method of the three, we would choose rejection sampling. This is because we find it the most easiest method to understand what is going on. However, we know that the method is slow, due to its dependence on repeated sampling and rejection of points outside the desired distribution.

2 Question 2: Laplace distribution

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

$$f(x) = DE(\mu, \lambda) = \frac{\lambda}{2} exp(-\lambda|x-\mu|) \tag{4}$$

2.1 2.a

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

Answer:

To generate the double exponential distribution by the inverse CDF method we need to find the cumulative distribution function to $f(x)$. This is done by the step as follows:

$$f(x) = \frac{\lambda}{2} exp(-\lambda|x-\mu|) = \begin{cases} \frac{\lambda}{2} \cdot e^{\lambda(x-\mu)} & for x \leq \mu \\ \frac{\lambda}{2} \cdot e^{-\lambda(x-\mu)} & for x > \mu \end{cases} \tag{5}$$

We need to calculate $F(x)$ for $x \leq \mu$ and $x > \mu$.

For $x \leq \mu$ we get:

$$F(u) = \int_{-\infty}^{x} \frac{\lambda}{2} \cdot e^{\lambda(u-\mu)} du \tag{6}$$

$$= \left[\frac{1}{2} e^{\lambda(u-\mu)} \right]_{u=-\infty}^{u=x} \tag{7}$$

$$= \frac{1}{2} e^{\lambda(x-\mu)} - \frac{1}{2} e^{\lambda(-\infty)} \tag{8}$$

$$= \frac{1}{2} e^{\lambda(x-\mu)} \tag{9}$$

For $x > \mu$ we can use the fact that the f(x) is symmetric around μ, which means:

$$\int_{-\infty}^{x} \frac{\lambda}{2} \cdot e^{-\lambda(u-\mu)} du = \frac{1}{2} + \int_{\mu}^{x} \frac{\lambda}{2} \cdot e^{-\lambda(u-\mu)} du \tag{10}$$

The CDF is therefore:

$$F(u) = \frac{1}{2} + \int_{\mu}^{x} \frac{\lambda}{2} \cdot e^{-\lambda(u-\mu)} du \tag{11}$$

$$= \frac{1}{2} + \left[-\frac{1}{2} \cdot e^{-\lambda(u-\mu)} \right]_{u=\mu}^{u=x} \tag{12}$$

$$= \frac{1}{2} - \frac{1}{2} \cdot e^{-\lambda(x-\mu)} + \frac{1}{2} \cdot e^{-\lambda(0)} \tag{13}$$

$$= 1 - \frac{1}{2} \cdot e^{-\lambda(x-\mu)} \tag{14}$$

The inverse for $F(x)$ can be found by calculating the inverse for (9) and (14). $F^{-1}(y)$ for $x \leq \mu$ is as follows:

$$y = \frac{1}{2} \cdot e^{\lambda(x-\mu)} \tag{15}$$

$$\iff ln(2y) = \lambda(x-\mu) \tag{16}$$

$$\iff \frac{ln(2y)}{\lambda} = x - \mu \tag{17}$$

$$\iff x = \mu + \frac{ln(2y)}{\lambda} \tag{18}$$

$F^{-1}(y)$ for $x > \mu$ is as follows:

$$y = 1 - \frac{1}{2} \cdot e^{-\lambda(x-\mu)} \tag{19}$$

$$\iff 1 - y = \frac{1}{2} \cdot e^{-\lambda(x-\mu)} \tag{20}$$

$$\iff 2 - 2y = e^{-\lambda(x-\mu)} \tag{21}$$

$$\iff ln(2-2y) = -\lambda(x-\mu) \tag{22}$$

$$\iff \frac{ln(2-2y)}{-\lambda} = x - \mu \tag{23}$$

$$\iff x = \mu - \frac{ln(2-2y)}{\lambda} \tag{24}$$

The inverse CDF for (5) is as follows:

$$F^{-1}(y) = \begin{cases} \mu + \frac{ln(2y)}{\lambda} & for x \leq \mu \\ \mu - \frac{ln(2-2y)}{\lambda} & for x > \mu \end{cases} \tag{25}$$

With the inverse CDF method we are generating $u \sim Unif(0,1)$ random numbers, with these we will then use different $F^{-1}(y)$. For $u \leq 0.5$ we will use $F^{-1}(y)$ for $x \leq \mu$ and for $u > 0.5$ we will use $F^{-1}(y)$ for $x > \mu$.

The code will therefore be as follows:

```r
laplace <- function(mu, lambda, n=10000){
  u <- runif(n)
  u1 <- u[u<=0.5]
  u2 <- u[u>0.5]
  x1 <- mu + (log(2*u1)/lambda)
  x2 <- mu - (log(2-2*u2)/lambda)
  x <- c(x1,x2)
  return(x)
}
```

We generate 10000 random numbers from our distribution and the result is presented as a histogram in figure 3.

```r
set.seed(13)
mu <- 0
lambda <- 1
x <- laplace(mu, lambda)
plot_data <- data.frame(x=x)
ggplot(plot_data) +
  geom_histogram(aes(x=x), fill="indianred", colour="black") +
  theme_bw() +
  labs(y = "Frequency")
```

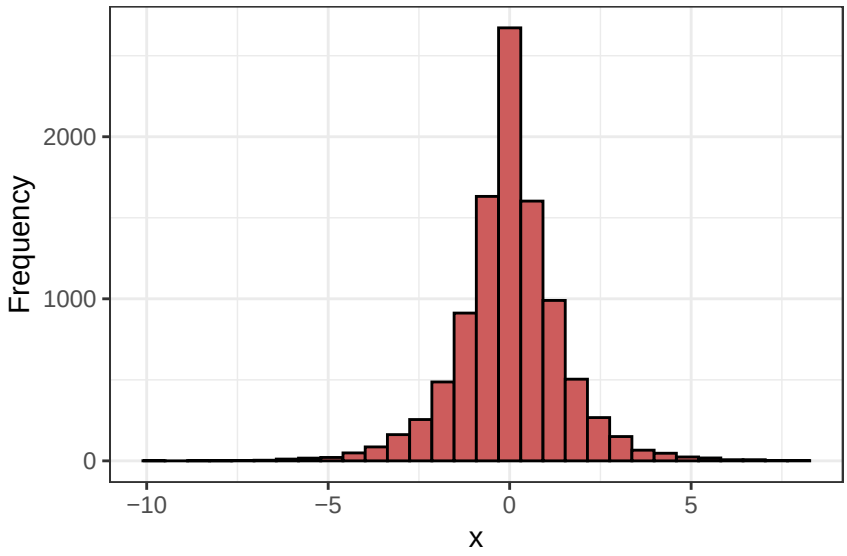

Figure 3: Histogram of 10000 sampled random variable from laplace distribution with $\mu = 0$ and $\lambda = 1$.

From figure 3, the density is similar to a normal distribution $\mathcal{N}(0, 1)$ but with heavier tails. Further investigation about the Laplace distribution[1] is that the Laplace distribution is similar to a normal distribution but it has fatter tails. Therefore, the result from our generated random numbers looks reasonable.

[1] https://en.wikipedia.org/wiki/Laplace_distribution

2.2 2.b

Question: Use rejection sampling with $DE(0,1)$ as envelope to generate $\mathcal{N}(0,1)$ variables. Explain step by step how this was done. How did you choose constant a in this method? Generate 2000 random numbers $\mathcal{N}(0,1)$ using your code and plot the histogram. Compute the average rejection rate R in the rejection sampling procedure. What is the expected rejection rate ER and how close is it to R? Generate 2000 numbers from $\mathcal{N}(0,1)$ using standard rnorm() procedure, plot the histogram and compare the obtained two histograms.

Answer: To find the value of a we tried different values of a, where the aim was to find a laplace distribution that is larger than a normal distribution with $\mu = 0$ and $\mu = 1$ for all values. This was done visually and the result is presented in figure 4.

```
normal_plot <- function(x){
  mu <- 0
  sd <- 1
  pdf <- (1/(sd*sqrt(2*pi)))*exp(-(0.5)*( (x-mu)/sd)^2 )
}
mu <- 0
lambda <- 1
a <- 1.4

laplace_plot <- function(x){
  a * lambda/2 * exp(-lambda*abs(x-mu))
}

ggplot() +
  xlim(-3,3) +
  geom_function(aes(colour="Normal"), fun=normal_plot) +
  geom_function(aes(colour="Laplace"), fun=laplace_plot) +
  scale_colour_manual(name="Distribution", values=c("steelblue", "indianred")) +
  theme_bw()
```

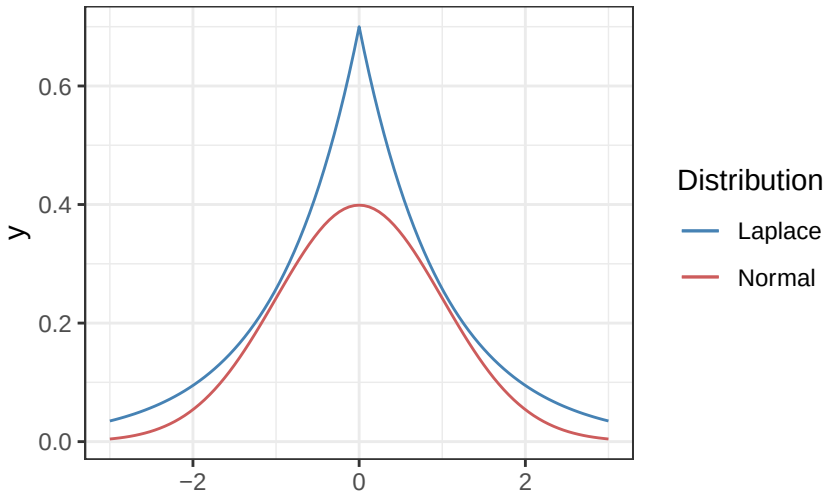

Figure 4: Laplace density function with $\mu = 0$ and $\lambda = 1$ multipled with constant 1.4 and Normal density function with $\mu = 0$ and $\lambda = 1$.

From figure 4, the Laplace distribution with $a = 1.4$ is above the normal distribution between the interval $[-3, 3]$. Since the Laplace distribution have heavier tails than the normal distribution, the Laplace distribution will also be above the normal distribution for the interval $(-\infty, \infty)$.

To generate 2000 random numbers $\mathcal{N}(0, 1)$ with rejection sampling we will have to generate more than 2000 numbers, since some number will be rejected. We chose to generate 3000 numbers from the Laplace distribution. The code to generate the numbers are as follows:

```
set.seed(13)
# Number of generated numbers
n <- 3000
# Generating Laplace distributed numbers
x <- laplace(0, 1, n=n)

# pdf-value of Laplace
g <- lambda/2*exp(-lambda*abs(x-mu))
# pdf-value of a*Laplace (the function that is above normal distribution)
e <- a*g
# pdf-value of the normal distribution we want to sample
f <- 1/(sqrt(2*pi))*exp(-0.5*x^2)
# uniformly distributed variables
u <- runif(n, 0, 1)

# The numbers that are accepted with rejection sampling
accepted <- x[u < f/e]
```

The rate of the rejected numbers are:

```
rate <- length(accepted)/n
rejection <- 1-rate
rejection
```

```
## [1] 0.2913333
```

The average rejection rate is around 29.13%. The expected number of accepted numbers is dependent on the value of a. Since a is 1.4, it is expected that 1 out of 1.4 numbers will be accepted, which is approximately $1/1.4 \approx 0.7143$. The expected rejection rate is therefore $1 - 0.7143 = 0.2857 = 28.57\%$. The average rejection rate from our rejection sampling is close to the expected rejection.

The number of generated random numbers are:

```
length(accepted)
```

```
## [1] 2126
```

Since we only want 2000 generated numbers, we remove the last excess numbers and a histogram of the distribution is presented in figure 5.

```
plot_data <- data.frame(laplace=accepted[1:2000])
ggplot(plot_data) +
  geom_histogram(aes(x=laplace), colour="black",fill="indianred", bins=40) +
  theme_bw() +
  labs(y="Frequency",
       x="x")
```

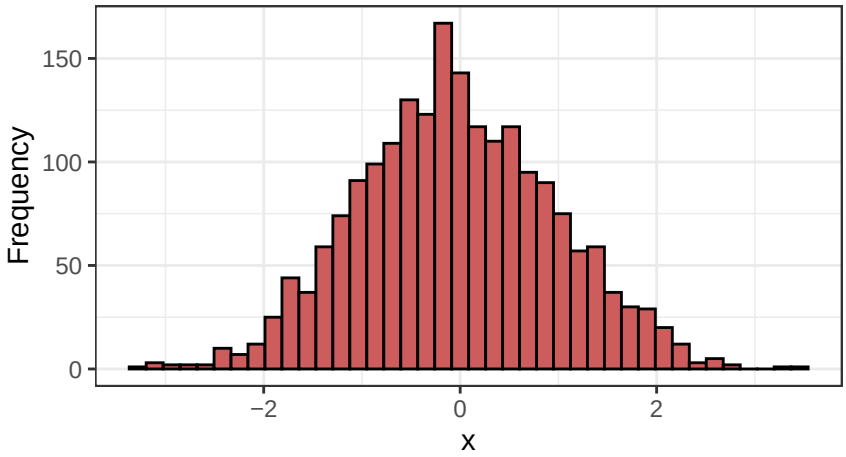

Figure 5: Histogram of 2000 generated $\mathcal{N}(0,1)$ with rejection sampling.

In figure 5, the distribution resembles a normal distribution with $\mathcal{N}(0,1)$.

We generate 2000 numbers with the function rnorm and compare it with our generated numbers from figure 5 The result is presented in figure 6

```
plot_data$normal <- rnorm(2000, 0, 1)
ggplot(plot_data) +
  geom_histogram(aes(x=laplace, fill="Laplace"), alpha=0.5, bins=40) +
  geom_histogram(aes(x=normal, fill="Normal"), alpha =0.5, bins=40) +
  theme_bw() +
  scale_fill_manual(name="Distribution", values=c("steelblue", "indianred"))
```

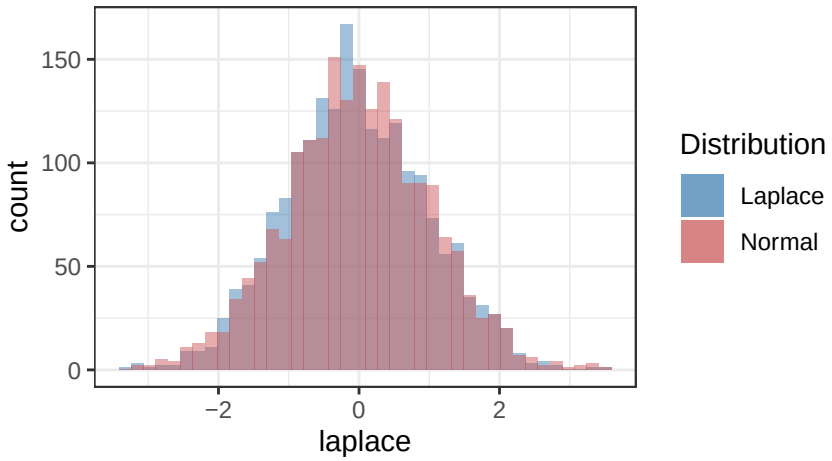

Figure 6: Histogram of 2000 generated $\mathcal{N}(0,1)$ with rejection sampling and 2000 generated $\mathcal{N}(0,1)$ with the function rnorm.

In figure 6 the rejection sampling with our Laplace distribution to generate Normal distributed variables are similar to the values generated from the function rnorm.

3 Statement of Contribution

We both worked on the tasks individually and helped each other when needed. We later compared and discussed our solutions before dividing the task of writing the laboration report.

3.1 Question 1

Text written by William.

3.2 Question 2

Text written by Duc.

4 Appendix

The code used in this laboration report are summarised in the code as follows:

```
library(knitr)
library(ggplot2)
knitr::opts_chunk$set(
  echo = TRUE,
  fig.width = 4.5,
  fig.height = 3)


f <- function(x){
  1 - abs(x)
}


ggplot() + xlim(-1,1) + geom_function(fun = f, aes(color = "f(x)")) +
  geom_function(fun = function(x){1}, aes(color = "e(x)")) +
  theme_bw() +
  scale_color_manual(values = c("indianred","steelblue"), name = "Function")



set.seed(13)


random_generator_a <- function(n) {

  g <- runif(n, -1,1)
  e <- 1
  f <- 1 - abs(g)

  U <- runif(n, 0, 1)

  accepted <- g[U<= (f/e)]
  return(accepted)
```

```r
}

random_generator_b <- function(n) {

  # Sample which distribution to use
  # sign will decide which function to sample from (Y or -Y)
  mu <- c(-1, 1)
  prob <- c(0.5, 0.5)
  sign <- sample(mu, n, replace = TRUE, p = prob)

  U <- runif(n, 0, 1)
  y <- sign * (1 - sqrt(1 - U))

  return(y)

}


random_generator_c <- function(n) {

  U1 <- runif(n,0, 1)
  U2 <- runif(n,0, 1)

  X <- U1-U2
  return(X)

}


library(ggplot2)
library(cowplot)
set.seed(13)
# Sample 22000 to make sure we get at least 10000 random numbers
data_a <- random_generator_a(22000)
# Remove excess observations so we only have 10000 left
data_a <- data_a[1:10000]
data_b <- random_generator_b(10000)
data_c <- random_generator_c(10000)


p1 <- ggplot(as.data.frame(data_a), aes(x = data_a)) +
  geom_histogram(color="black", fill="lightblue") +
  theme_bw()


p2 <- ggplot(as.data.frame(data_b), aes(x = data_b)) +
  geom_histogram(color="black", fill="lightblue") +
  theme_bw()
```

```r
p3 <- ggplot(as.data.frame(data_c), aes(x = data_c)) +
  geom_histogram(color="black", fill="lightblue") +
  theme_bw()


plot_grid(p1, p2,p3 ,labels=c("a", "b","c"), ncol = 2, nrow = 2)
# Variance from rejection sampling in 1 a)
var(data_a)
laplace <- function(mu, lambda, n=10000){
  u <- runif(n)
  u1 <- u[u<=0.5]
  u2 <- u[u>0.5]
  x1 <- mu + (log(2*u1)/lambda)
  x2 <- mu - (log(2-2*u2)/lambda)
  x <- c(x1,x2)
  return(x)
}

set.seed(13)
mu <- 0
lambda <- 1
x <- laplace(mu, lambda)
plot_data <- data.frame(x=x)
ggplot(plot_data) +
  geom_histogram(aes(x=x), fill="indianred", colour="black") +
  theme_bw() +
  labs(y = "Frequency")


normal_plot <- function(x){
  mu <- 0
  sd <- 1
  pdf <- (1/(sd*sqrt(2*pi)))*exp(-(0.5)*( (x-mu)/sd)^2 )
}
mu <- 0
lambda <- 1
a <- 1.4

laplace_plot <- function(x){
  a * lambda/2 * exp(-lambda*abs(x-mu))
}

ggplot() +
  xlim(-3,3) +
  geom_function(aes(colour="Normal"), fun=normal_plot) +
  geom_function(aes(colour="Laplace"), fun=laplace_plot) +
  scale_colour_manual(name="Distribution", values=c("steelblue", "indianred")) +
  theme_bw()
```

```r
set.seed(13)
# Number of generated numbers
n <- 3000
# Generating Laplace distributed numbers
x <- laplace(0, 1, n=n)

# pdf-value of Laplace
g <- lambda/2*exp(-lambda*abs(x-mu))
# pdf-value of a*Laplace (the function that is above normal distribution)
e <- a*g
# pdf-value of the normal distribution we want to sample
f <- 1/(sqrt(2*pi))*exp(-0.5*x^2)
# uniformly distributed variables
u <- runif(n, 0, 1)

# The numbers that are accepted with rejection sampling
accepted <- x[u < f/e]
rate <- length(accepted)/n
rejection <- 1-rate
rejection
length(accepted)
plot_data <- data.frame(laplace=accepted[1:2000])
ggplot(plot_data) +
  geom_histogram(aes(x=laplace), colour="black",fill="indianred", bins=40) +
  theme_bw() +
  labs(y="Frequency",
       x="x")
plot_data$normal <- rnorm(2000, 0, 1)
ggplot(plot_data) +
  geom_histogram(aes(x=laplace, fill="Laplace"), alpha=0.5, bins=40) +
  geom_histogram(aes(x=normal, fill="Normal"), alpha =0.5, bins=40) +
  theme_bw() +
  scale_fill_manual(name="Distribution", values=c("steelblue", "indianred"))
```