# Lab 3 report

Simon Jorstedt & Siddhesh Sreedar

2023-11-20

## Question 1

a)

```
# Question 1.a
custom_density <- function(x) {

  ifelse(x < -1 | x > 1, 0,
         ifelse(-1 <= x & x <= 0, x + 1,
                ifelse(0 < x & x <= 1, 1 - x, 0)))
}

#Based on the function, we have choosen the uniform distribution as the envelop.
uniform_density <- function(x){
  ifelse(x < -1 | x > 1, 0,
         ifelse(-1 <= x & x <= 0, 1,
                ifelse(0 < x & x <= 1, 1, 0)))
}

#Not required to scale since scaling would lead to more wastage.

sampling<- function(amount){
  values<-c()

  i<-1
  while(length(values)<amount){
    U<-runif(1)

    Y<-runif(1,min = -1,max =1)

    if(U <= custom_density(Y)/uniform_density(Y)){
      values[i]<-Y
      i=i+1
    }
  }
  hist(values, main = "Rejection sampling method")
  return(head(values,10))
}
```

b)

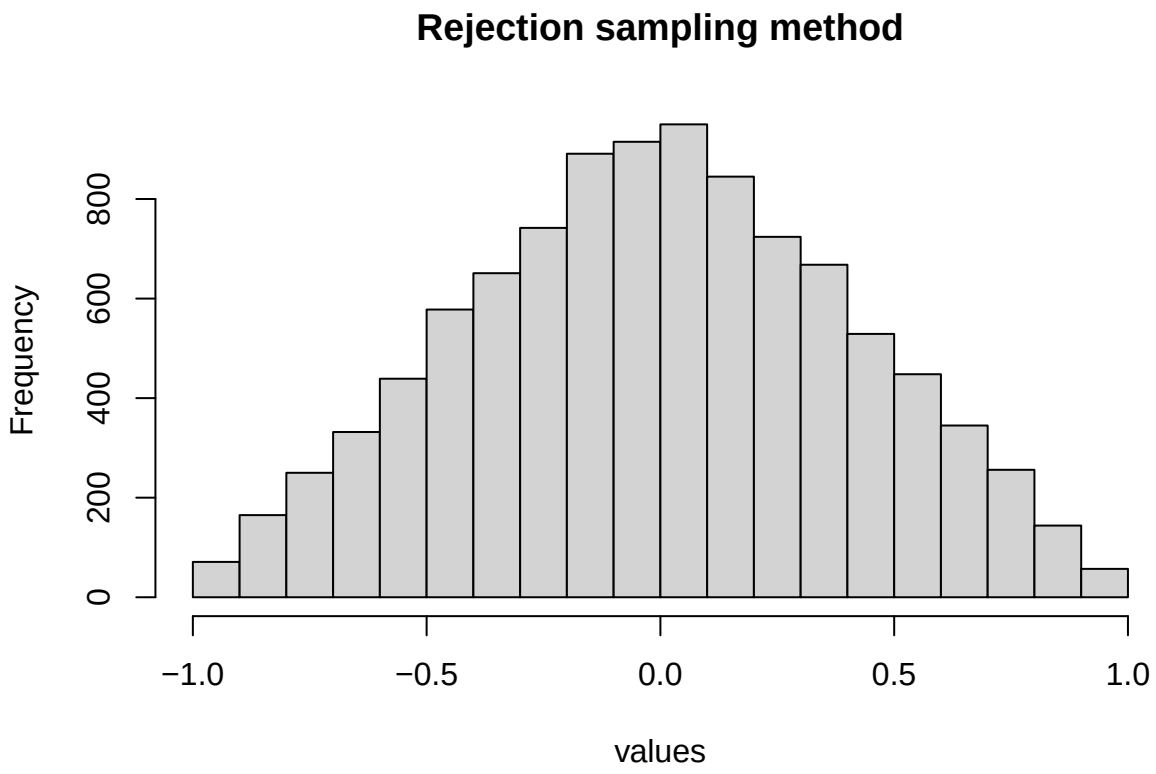
```r
# Question 1.b
x_pos <- function(u){
  a<-1-sqrt(1-u)
  #hist(a)
  return(a)
}

x_neg <- function(u){
  a<-sqrt(1-u)-1
  #hist(a)
  return(a)
}

mixing_para_positive<-0.5
mixing_para_negative<-0.5

#composition sampling
mixture2<- function(n,m_p,m_n){
  u <- runif(1000)
  a<-sample(c(1,2),n,replace = TRUE,prob = c(m_p,m_n))

  final<-ifelse(a==1,x_pos(u),x_neg(u))

  hist(final,breaks=100, main = "composition method")
  return(head(final,10))
}
```

c)

```r
# Question 1.c
diff<- function(n){
  U1<-runif(n)
  U2<-runif(n)

  a<-U1-U2

  hist(a, main = "Difference")
  return(head(a,10))
}
```

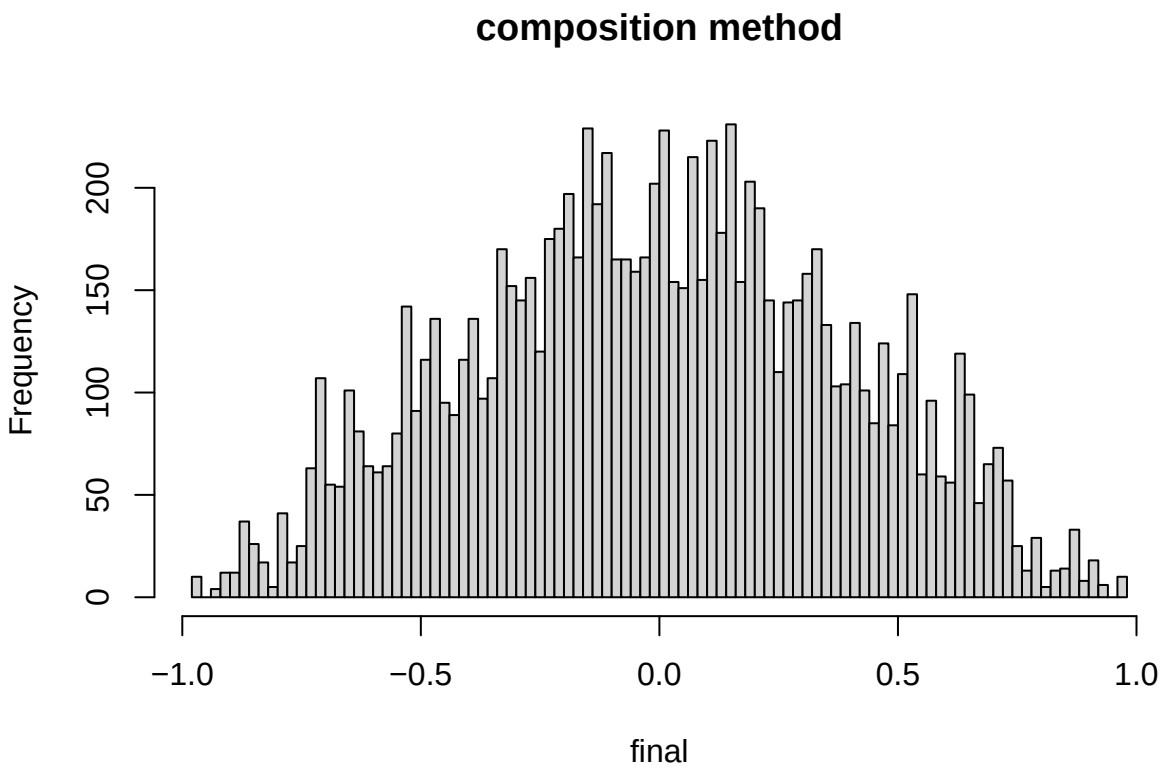
d) Returing 10 samples for each

```r
# Question 1.d
sampling(10000)
```



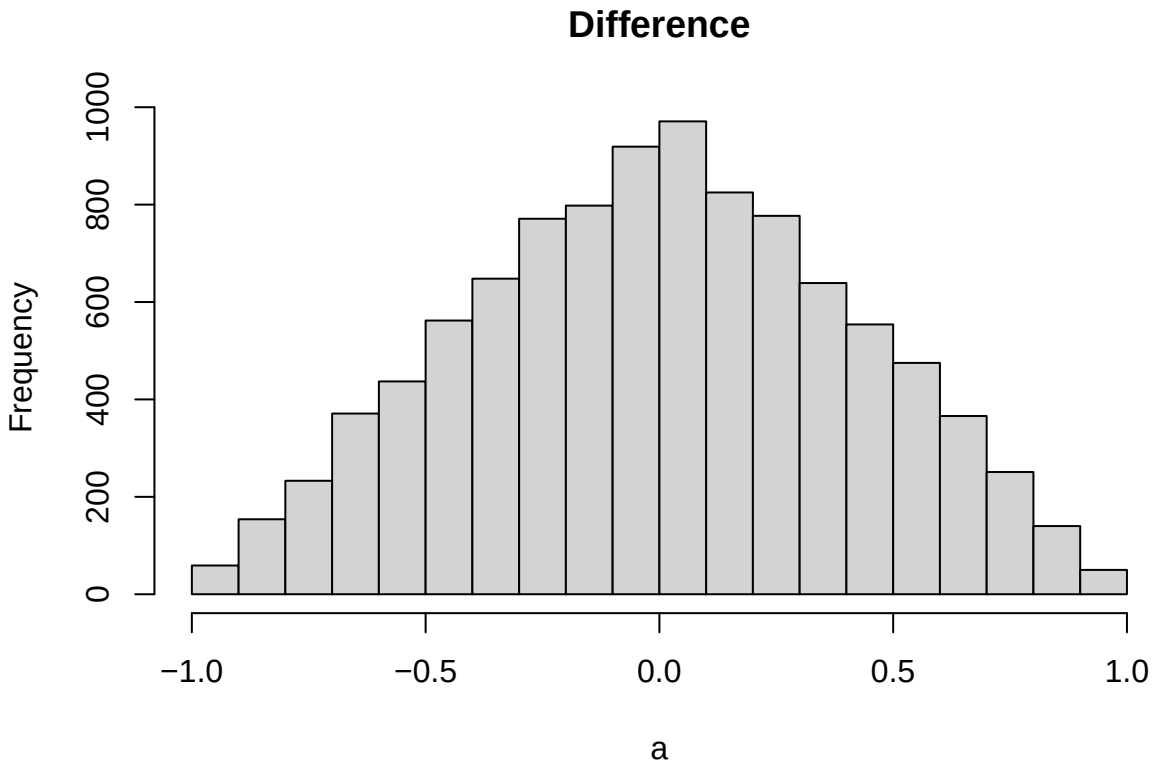

```
##  [1]  0.12783475 -0.53767072  0.05155616 -0.57941790  0.35151944 -0.93725650
##  [7]  0.57823863  0.09696327  0.30282221 -0.24087500
```

```
mixture2(10000,mixing_para_positive,mixing_para_negative)
```




```
##  [1] -0.4790065  0.2490181 -0.3860627 -0.1341828 -0.3003147 -0.1742595
##  [7]  0.1839798  0.4645637  0.2131294  0.1626661
```

```
a<-diff(10000)
```




```
var(a)
```

```
## [1] 0.1814659
```

Based on the 3, I would prefer to generate the samples using c) as it was the most straightforward way to implement.

## Question 2

We will create a function for sampling from the double exponential (Laplace) distribution with location parameter 0, and scale parameter 1, with the density being

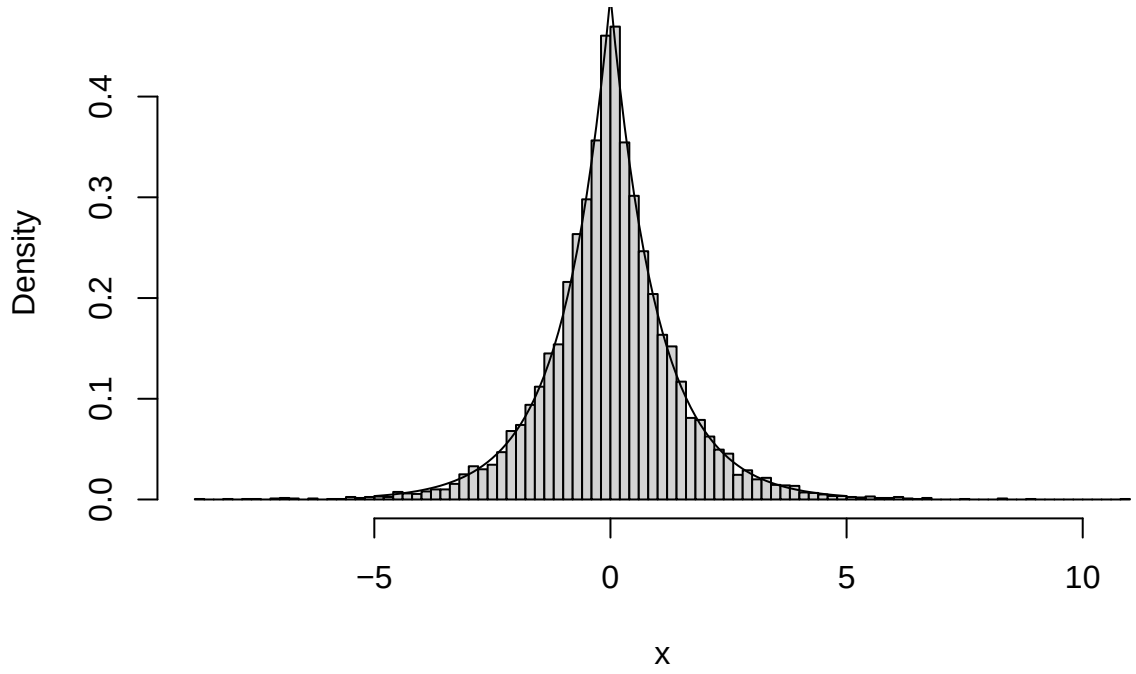
$$g(x) = \frac{1}{2} \cdot \exp\Big( - |x|\Big)$$

Below is our implementation `rlaplace` which samples from the Laplace$(\mu, \lambda)$ distribution using the *inverse distribution function (IDF) method*. The inverse distribution function $G^-1(x)$ was obtained through our reference for the Laplace distribution. As per the IDF method, `rlaplace` first generates a uniformly random number from $U(0, 1)$ for each desired observation. Then the IDF is applied to each such observation, so that the resulting observations are $Laplace(0, 1)$ distributed.

We then proceed to generate $10^4$ approximately independent observations from `rlaplace`, and plot the result in Figure 2.1, along with the true density curve of $Laplace(0, 1)$. The results are quite good, as the distribution of the simulated values closely matches the true density.

```r
# Question 2.a

# Generate laplace variates using Inverse Distribution Function (IDF)
rlaplace <- function(n=1, location=0, scale=1){
  # Generate n observations from U(0,1)
  u <- runif(n)

  # Calculate (and return) corresponding IDF values
  location - scale*sign(u - 0.5)*log(1-2*abs(u-0.5))
}
```

**Fig 2.1. Density of simulated Laplace observations**




We will now proceed to use the Rejection Sampling Method (RSM) (see reference) to generate standard normal variates from $N(0,1)$. For the envelope we will use the $Laplace(0,1)$ distribution, scaled appropriately by a constant $a$. In order to find the optimal $a$, we construct the following expression and then derive the optimal (maximal) $a$ that satisfies the inequality.

$$h(x) = \frac{g(x)}{f(x)} = \frac{1/2 \cdot \exp(-x)}{1/\sqrt{2\pi} \cdot \exp(-\frac{x^2}{2})} = \sqrt{\frac{\pi}{2}} \cdot \exp\left(\frac{x^2}{2} - x\right) \geq a$$

$$h'(x) = \sqrt{\frac{\pi}{2}} \cdot (x-1) \cdot \exp\left(\frac{x^2}{2} - x\right)$$

It is clear that $h(x)$ grows very large as $x$ approaches $\pm\infty$, and thus the lone extrema at $x = 1$ is a global minima. This implies that the optimal $a$ is

$$a = h(1) = \sqrt{\frac{\pi}{2}} \cdot \exp\left(\frac{1^2}{2} - 1\right) \approx 0.76 \text{(rounded down)}$$

Our envelope is thus

$$e(x) = g(x)/a.$$

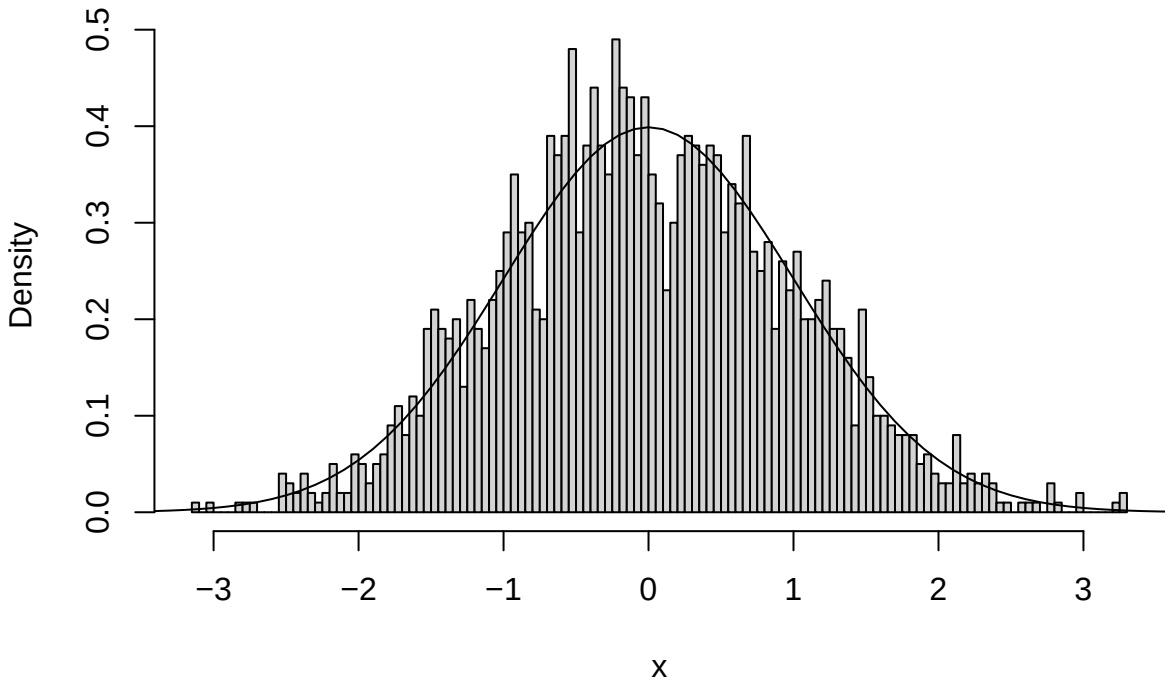
We can now proceed to simulate 2000 random numbers from $N(0, 1)$ using the rejection sampling method. Below we implement this in `rnorm_rejection`. We also measure and output the average rejection rate when the function is run.

```
# Question 2.b

# Our derived optimal a
a <- sqrt(pi/2) * exp(-1/2)

# Generate standard normal variates using rejection sampling
rnorm_rejection <- function(n=1){ #mean=0, sd=1
  output_vector <- c()

  # Keep track of total number of attempts
  n_tries <- 0

  # Run the algorithm once for every n desired values
  for (i in 1:n){
    # Repeat the sampling procedure until a value is accepted
    repeat{
      n_tries <- n_tries + 1
      u <- runif(1, 0, 1)
      Y <- rlaplace(1, 0, 1)
      ratio <- dnorm(Y) / (dlaplace(Y, 0, 1) / a)
      if(u <= ratio){output_vector <- c(output_vector, Y);break}
    }
  }
  cat("The average rejection rate:", (n_tries-n)/n_tries, "\n")
  return(output_vector)
}
```

In Figure 2.2 below, we plot 2000 standard normal variates sampled using `rnorm_rejection`, along with the true standard normal density (instead of values sampled from `rnorm`). The simulated distribution clearly resembles the standard normal, but it is not very smooth, which is desired.

```
## The average rejection rate: 0.2269037
```

**Fig 2.2. Laplace sampled Normal variates**




## Appendix

```
# Question 1.a
custom_density <- function(x) {

  ifelse(x < -1 | x > 1, 0,
         ifelse(-1 <= x & x <= 0, x + 1,
                ifelse(0 < x & x <= 1, 1 - x, 0)))
}

#Based on the function, we have choosen the uniform distribution as the envelop.
uniform_density <- function(x){
  ifelse(x < -1 | x > 1, 0,
         ifelse(-1 <= x & x <= 0, 1,
                ifelse(0 < x & x <= 1, 1, 0)))
}

#Not required to scale since scaling would lead to more wastage.

sampling<- function(amount){
  values<-c()

  i<-1
```

```
  while(length(values)<amount){
    U<-runif(1)

    Y<-runif(1,min = -1,max =1)

    if(U <= custom_density(Y)/uniform_density(Y)){
      values[i]<-Y
      i=i+1
    }
  }
  hist(values, main = "Rejection sampling method")
  return(head(values,10))
}

# Question 1.b
x_pos <- function(u){
  a<-1-sqrt(1-u)
  #hist(a)
  return(a)
}

x_neg <- function(u){
  a<-sqrt(1-u)-1
  #hist(a)
  return(a)
}

mixing_para_positive<-0.5
mixing_para_negative<-0.5

#composition sampling
mixture2<- function(n,m_p,m_n){
  u <- runif(1000)
  a<-sample(c(1,2),n,replace = TRUE,prob = c(m_p,m_n))

  final<-ifelse(a==1,x_pos(u),x_neg(u))

  hist(final,breaks=100, main = "composition method")
  return(head(final,10))
}

# Question 1.c
diff<- function(n){
  U1<-runif(n)
  U2<-runif(n)

  a<-U1-U2

  hist(a, main = "Difference")
  return(head(a,10))
}

# Question 1.d
```

```r
sampling(10000)
mixture2(10000,mixing_para_positive,mixing_para_negative)
a<-diff(10000)

var(a)

# Question 2.a

# Generate laplace variates using Inverse Distribution Function (IDF)
rlaplace <- function(n=1, location=0, scale=1){
  # Generate n observations from U(0,1)
  u <- runif(n)

  # Calculate (and return) corresponding IDF values
  location - scale*sign(u - 0.5)*log(1-2*abs(u-0.5))
}

# Question 2.a

# Laplace density function for use in the histogram below
dlaplace <- function(x, location, scale){
  (scale / 2) * exp(-scale*abs(x-location))
}

# Plot a histogram of 10^4 Laplace observations
set.seed(03737693)
hist(rlaplace(10000, 0, 1), breaks = 100, freq = F,
     main = "Fig 2.1. Density of simulated Laplace observations",
     xlab = "x")

# Add true Laplace density curve
points(x = seq(-5, 5, 0.1),
       y = dlaplace(seq(-5, 5, 0.1), 0, 1),
       type = "l")

# Question 2.b

# Our derived optimal a
a <- sqrt(pi/2) * exp(-1/2)

# Generate standard normal variates using rejection sampling
rnorm_rejection <- function(n=1){ #mean=0, sd=1
  output_vector <- c()

  # Keep track of total number of attempts
  n_tries <- 0

  # Run the algorithm once for every n desired values
  for (i in 1:n){
    # Repeat the sampling procedure until a value is accepted
    repeat{
      n_tries <- n_tries + 1
      u <- runif(1, 0, 1)
```

```
      Y <- rlaplace(1, 0, 1)
      ratio <- dnorm(Y) / (dlaplace(Y, 0, 1) / a)
      if(u <= ratio){output_vector <- c(output_vector, Y);break}
    }
  }
  cat("The average rejection rate:", (n_tries-n)/n_tries, "\n")
  return(output_vector)
}

# Question 2.b

# Plot a histogram of simulated Normal variates
set.seed(9432719)
hist(rnorm_rejection(2000), breaks=100, freq = F,
     main = "Fig 2.2. Laplace sampled Normal variates",
     xlab = "x")

# Add true standard normal density curve
points(x = seq(-5, 5, 0.1), y = dnorm(seq(-5, 5, 0.1), 0, 1), type = "l")
```