732A90 Computational Statistics

Lab 3

Johannes Hedström & Mikael Montén

STIMA
Institutionen för datavetenskap
Linköpings universitet

2024-01-25

Contents

```r
library(ggplot2)
```

1 Statement of contribution

Mikael did assignment 1 and Johannes did assignment 2, we discussed our results afterwards.

2 Question 1: Sampling algorithms for a triangle distribution

Consider the following density with a triangle-shape (another triangle distribution than considered in Lecture 3):

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

We are interested to generate draws of a random variable X with this density. Below is the density function plotted.

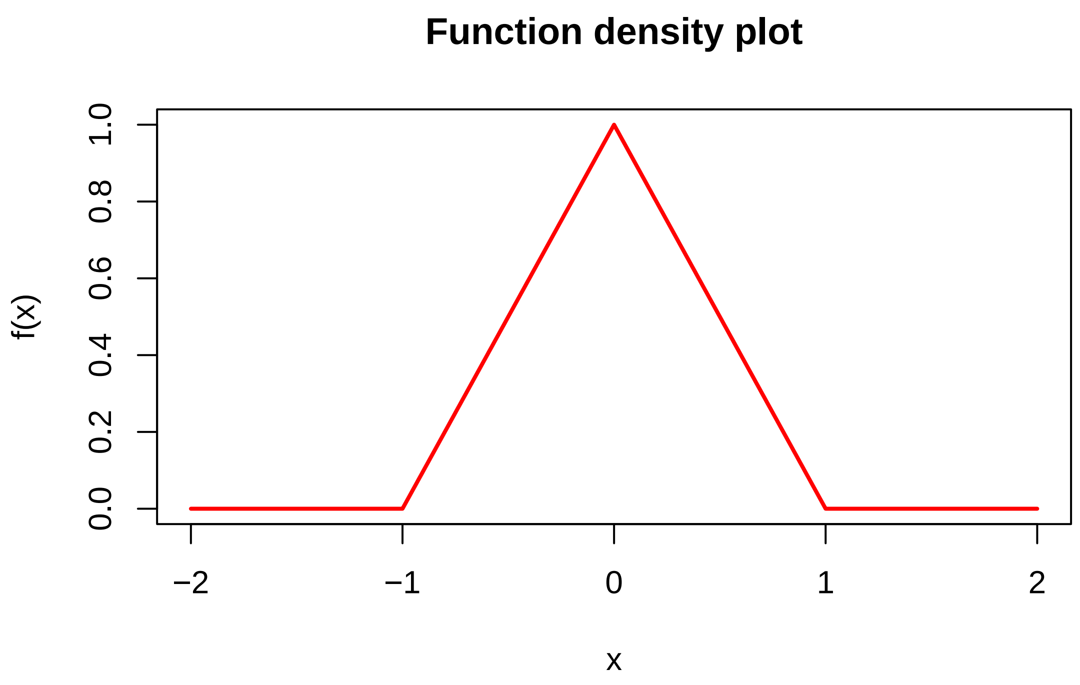


Figure 1: Density of triangle distribution

The minimum is -1, maximum is 1 and mode is 0.

2.0.1 a)

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

A uniform distribution between -1 and 1 would suffice as an envelope, albeit it could probably be optimized to have less area.

```r
# envelope (uniform)
e <- function(x) {
  ifelse(x < -1 | x > 1, 0, 1)
}
```

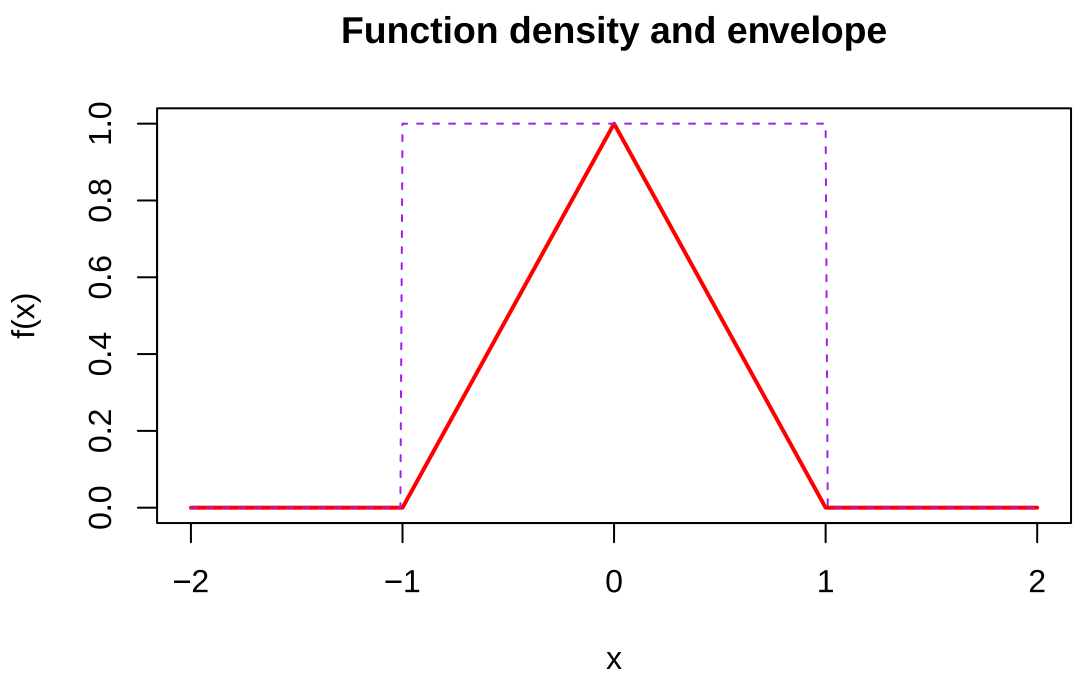


Figure 2: Envelope function on top triangle density function

Rejection sampling is done by

1. Sample $Y \sim g$
2. Sample $U \sim Unif(0,1)$
3. If $U \leq f(Y)/e(Y)$, accept Y; set X = Y; otherwise reject it

```r
rejection_sampling <- function(n) {
  stopifnot(is.numeric(n))
  x <- c()

  for(i in 1:n){
    y <- runif(1, min = -1, max = 1)  # sample Y~g
    u <- runif(1)  # sample U~Unif(0,1)

    if(u <= f(y)/e(y)){
      x[i] <- y
    }
  }
  return(x)
}
```

TOLKNING

2.0.2 b)

In Lecture 3, another triangle distribution was generated using the inverse cumulative distribution function method, see page 9-10 of the lecture notes. Let Y be a random variable following this distribution. 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

```r
composition_sampling <- function(Y, n, prob){
  negY <- -Y

  mu <- c(mean(Y),mean(negY))
  sigma <- c(sd(Y), sd(negY))
  prob <- c(prob, 1-prob)
  n <- n

  g <- sample(length(mu), n, replace = TRUE, p = prob)
  x <- rnorm(n, mean = mu[g], sd = sigma[g])

  return(x)
}
```

TOLKNING

2.0.3 c)

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.

```r
iud_sampling <- function(n) {
  u1 <- runif(n)
  u2 <- runif(n)

  x <- u1 - u2

  return(x)
}
```

TOLKNING

2.0.4 d)

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

Rejection sampling,

```
set.seed(123)

# rejection sampling
sample_rej <- rejection_sampling(10000)
hist(sample_rej, freq = FALSE, main = "Rejection sampling", col = "lightblue", xlim = c(-1.5, 1.5), xlab
curve(f(x), add = TRUE, col = "red", lwd = 2) # add the true distribution
```

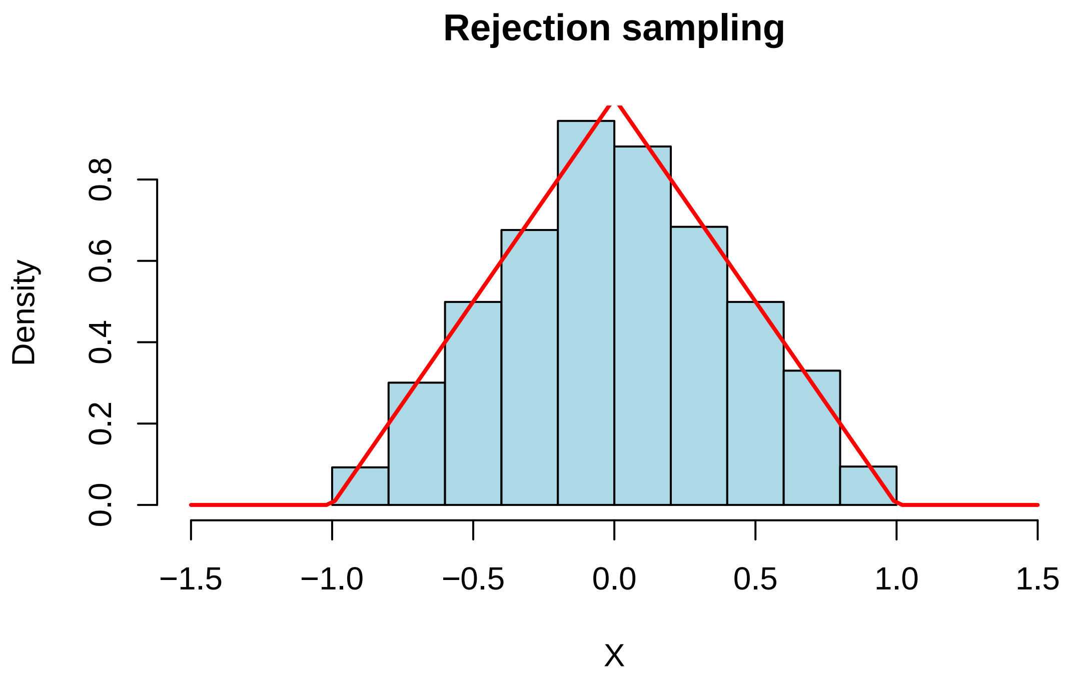


Figure 3: Rejection sampling with true distribution on top

TOLKNING

Individually uniformly distributed sampling,

```
# individually uniformly distributed sampling
sample_iud <- iud_sampling(10000)
hist(sample_iud, freq = FALSE, main = "I.U.D sampling", col = "lightblue", xlim = c(-1.5, 1.5), xlab = "
curve(f(x), add = TRUE, col = "red", lwd = 2) # add the true distribution
```

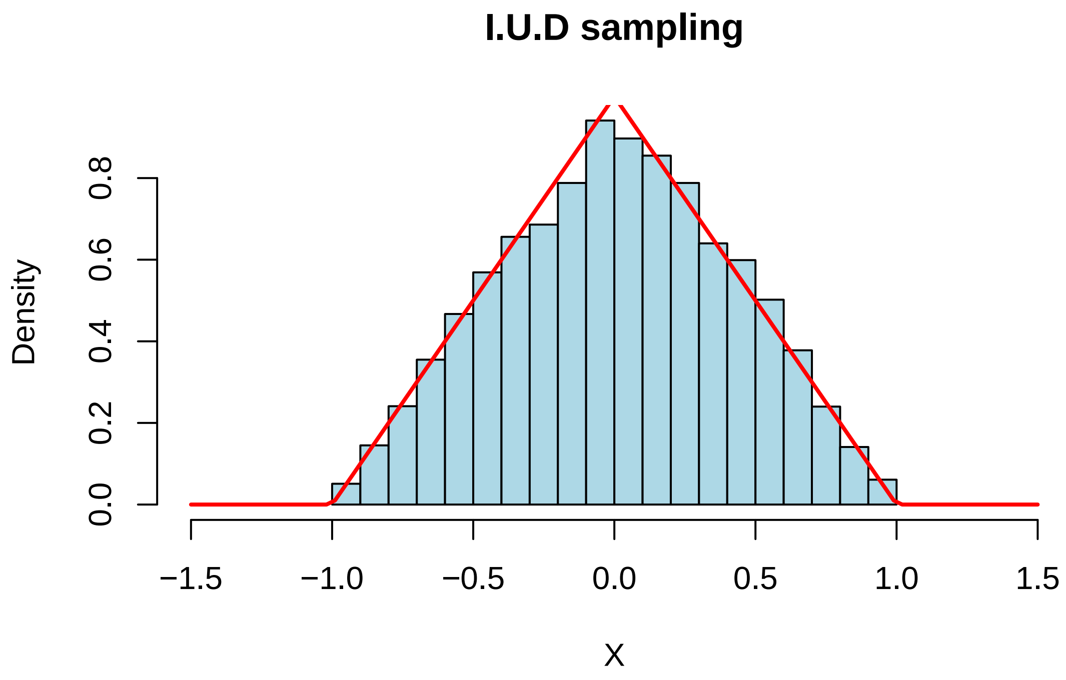
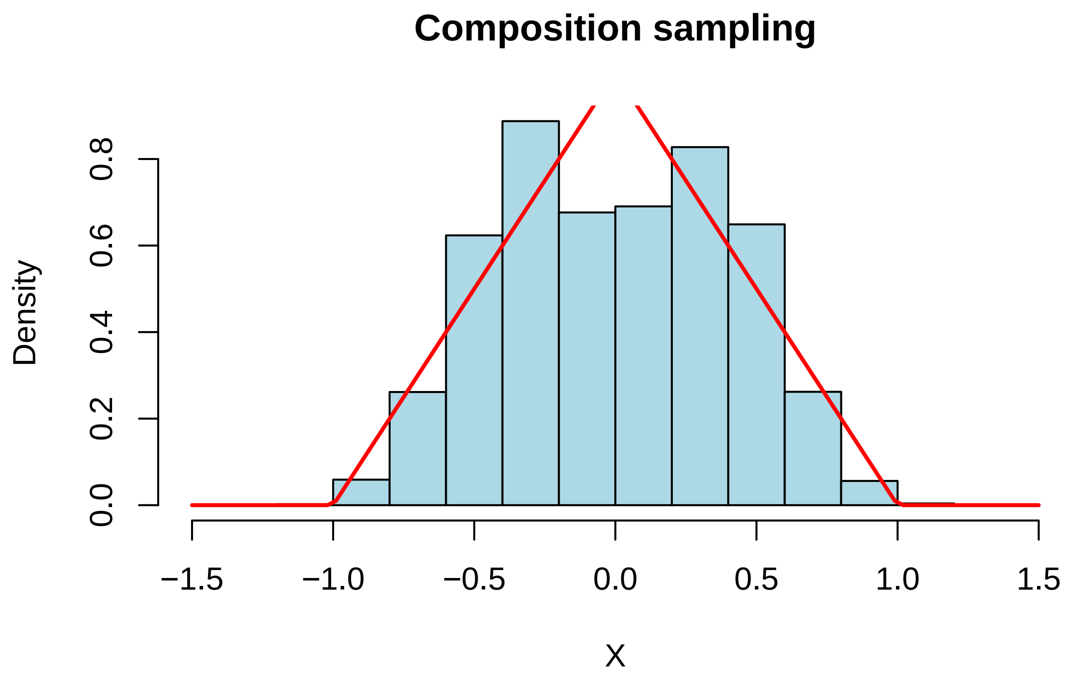


Figure 4: Individually unfirmoly distributed sampling with true distribution on top

TOLKNING

Composition sampling

```r
# composition sampling
u <- runif(10000)
y <- 1-sqrt(1-u)
sample_comp <- composition_sampling(y, 10000, 0.5)
hist(sample_comp, freq = FALSE, main = "Composition sampling", col = "lightblue", xlim = c(-1.5, 1.5), x
curve(f(x), add = TRUE, col = "red", lwd = 2) # add the true distribution
```



TOLKNING

The IUD sampling provides the best fit according to the distribution. The variance of X is

```r
var(sample_iud)
```

```
## [1] 0.1699904
```

3 Question 2: Laplace distribution

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

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

3.1 a

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.

Integrate PDF to CDF:

$$F(x) = \int_{-\inf}^{x} \frac{-\lambda}{2} exp(-\lambda|x-\mu|)dx$$

Breaking it to two parts, $x < \mu$ and $x \geq \mu$:

$$\frac{\lambda}{2}\int_{-\inf}^{x} \exp(\lambda(\mu - x))dx \text{ for } x < \mu$$
$$\frac{\lambda}{2}\int_{-\inf}^{x} \exp(-\lambda(x - \mu))dx \text{ for } x \geq \mu$$

Then solving the integrals :

$$x < \mu :$$
$$F(x) = \Big[-\frac{1}{2}\exp(\lambda(\mu - x)\Big]_{-\infty}^{x} = -\frac{1}{2}\exp(\lambda(\mu - x))$$

$$x \geq \mu :$$
$$F(x) = \Big[-\frac{1}{2}\exp(-\lambda(x - \mu))\Big]_{-\infty}^{x} = 1 - \frac{1}{2}\exp(-\lambda(x - \mu))$$

Then combining the results above:

$$F(x) = \begin{cases} -\frac{1}{2}\exp(\lambda(\mu - x)) & \text{if } x < \mu \\ 1 - \frac{1}{2}\exp(-\lambda(x - \mu)) & \text{if } x \geq \mu \end{cases}$$

As lambda = 1 and mu = 0 the cumulative distribution function of the laplace distribution is :

$$F(x) = \begin{cases} -\frac{1}{2}\exp(-x) & \text{if } x < \mu \\ 1 - \frac{1}{2}\exp(x) & \text{if } x \geq \mu \end{cases}$$

Now we need to look for the inverse CDF:

Solve (x):

For

$$x < 0 :$$

Setting y = the Function

$$-\frac{1}{2}\exp(-x) = y$$

Multiplying by -2

$$\exp(x) = -2y$$

taking the logarithm to remove exp

$$-\frac{x}{2} = \ln(-2 * y)$$

Multiplying by minus 1

$$x = -\ln(2 * y)$$

For x $\geq$ 0:

$$\frac{1}{2}\exp(x) = 1 - y$$

Multiplying by two

$$\exp(x) = 2(1 - y)$$

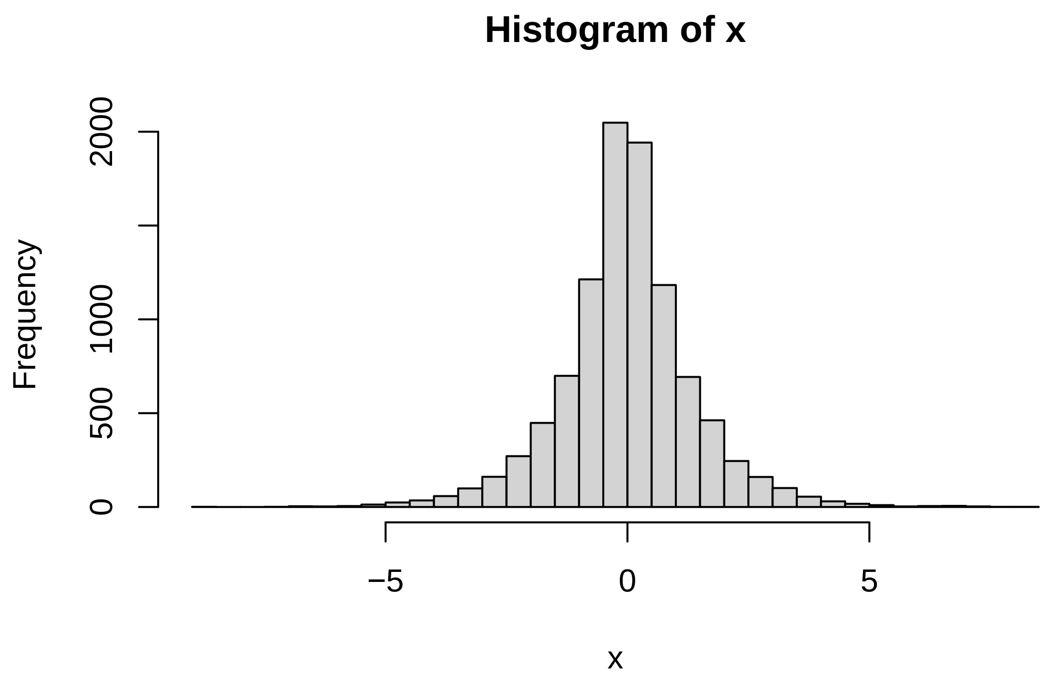
Logarithm to get rid of exp

$$x = \ln(2(1 - y))$$

And as y will be generated from the uniform distrubution(0,1) we can set the inverse CDF to:

$$F^{-1}(y) = \begin{cases} -\ln(2y) & \text{if } y < \frac{1}{2} \\ \ln(2(1-y)) & \text{if } y \geq \frac{1}{2} \end{cases}$$

```r
# this code was inspired wikipedia

laplace = function(n,lambda=1){

  y = runif(n,0,1) # draw from the uniform dist
  x <- c()

  # checking if the drawn values are bigger or smaller then 0.5 then making the calculations
  for(i in 1:n){

  if(y[i] >= 0.5){ # the condition in the inverse CFD
    x[i] <- lambda* log(2 * (1 - y[i]))
  }else{
    x[i] <- lambda* -log(2 * y[i])
  }

  }
  x # returning the generated CDF

}

# number of draws
n = 10000

# generated numbers
x <- laplace(10000)


# histogram
hist(x,breaks=30 )
```



The histogram looks reasonable as its symmetric and looks to align with the mean around 0, it drops of quickly which is expected with lambda = 1 and have heavy tails which is normal for the Laplace distribution.

3.2 b

Use rejection sampling with DE(0, 1) as envelope to generate 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 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 N (0, 1) using standard rnorm() procedure, plot the histogram and compare the obtained two histograms.

Scaling my envelope to be bigger than the density of $N(0,1)$

$$e(x) = \frac{g(x)}{\alpha} \geq f(x)$$

Choosing alpha with a loop(instead of solving from the formula) through testing values from 1 to 0 and stopping when i find an alpha value that for all x in g(x)/alpha is greater or equal to f(x) where x is going to be a sequence from -100 to 100(hopefully a big enough interval for x instead of testing all x values).

```r
# scaling DE(0,1)

# Define Laplace density function g(x)
laplace_density <- function(x) {

  (1/2) * exp(-abs(x))

}

# Define standard normal density function f(x)
normal_density <- function(x) {
 1/(sqrt(2*pi)) * exp(-x^2/2)
}

# looping to find alpha, starting with 1 and going downwards to find optimal alpha
for(i in seq(1,0, by=-0.001)){
  if(all(laplace_density(seq(-100, 100, 0.01)) / i >= normal_density(seq(-100, 100, 0.01)))){
  alpha = i
  print(i)
  break
  }
}
```

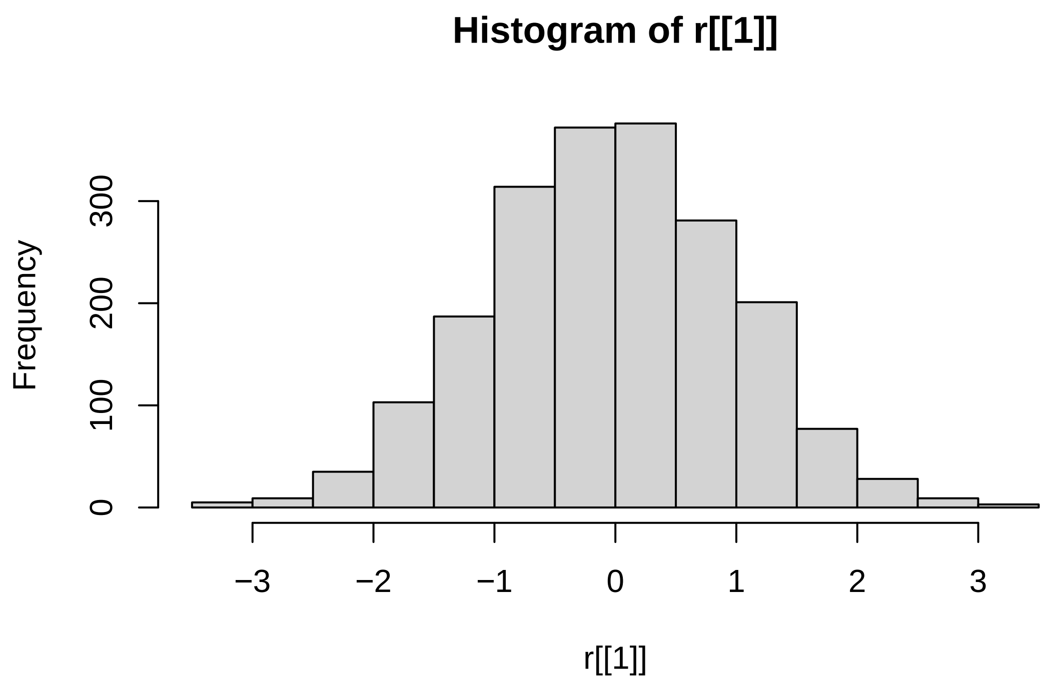
```
## [1] 0.76
```

```r
cat('my alpha is', alpha)
```

```
## my alpha is 0.76
```

```r
e <- function(y,alpha){

laplace_density(y)/alpha
}
```

```r
# reject function
reject <- function(n,alpha){
  vec <- c()
  # to generate n number of values and calculating all loops
  i <- 1
  nr_loops <- 0

  while (i<=n) {

    nr_loops <- nr_loops +1

    y <- laplace(1) # draw a number from the laplace dist

    u <- runif(1,0,1) # sampling a number between 0 and 1 to check the condition with

    if(u <= (dnorm(y,0,1)/e(y,alpha))){ # checking if to accept Y

      vec[i] <- y
      i <- i+1
    }




  }

  return(list(vec,nr_loops))
}

r <- reject(2000,alpha)

hist(r[[1]])
```



```
cat('The avarage rejection rate is: ',(r[[2]]- 2000)/r[[2]])
```

```
## The avarage rejection rate is:  0.2441421
```

The expected rejection rate

$$E\left[U \geq \frac{f(Y)}{e(Y)}\right] = 1 - E\left[U \leq \frac{f(Y)}{\frac{g(Y)}{\alpha}}\right]$$

The expected rejection rate is: $1 - \alpha = 0.24$ which mean that our average rejection is really close to the expected rejection rate.

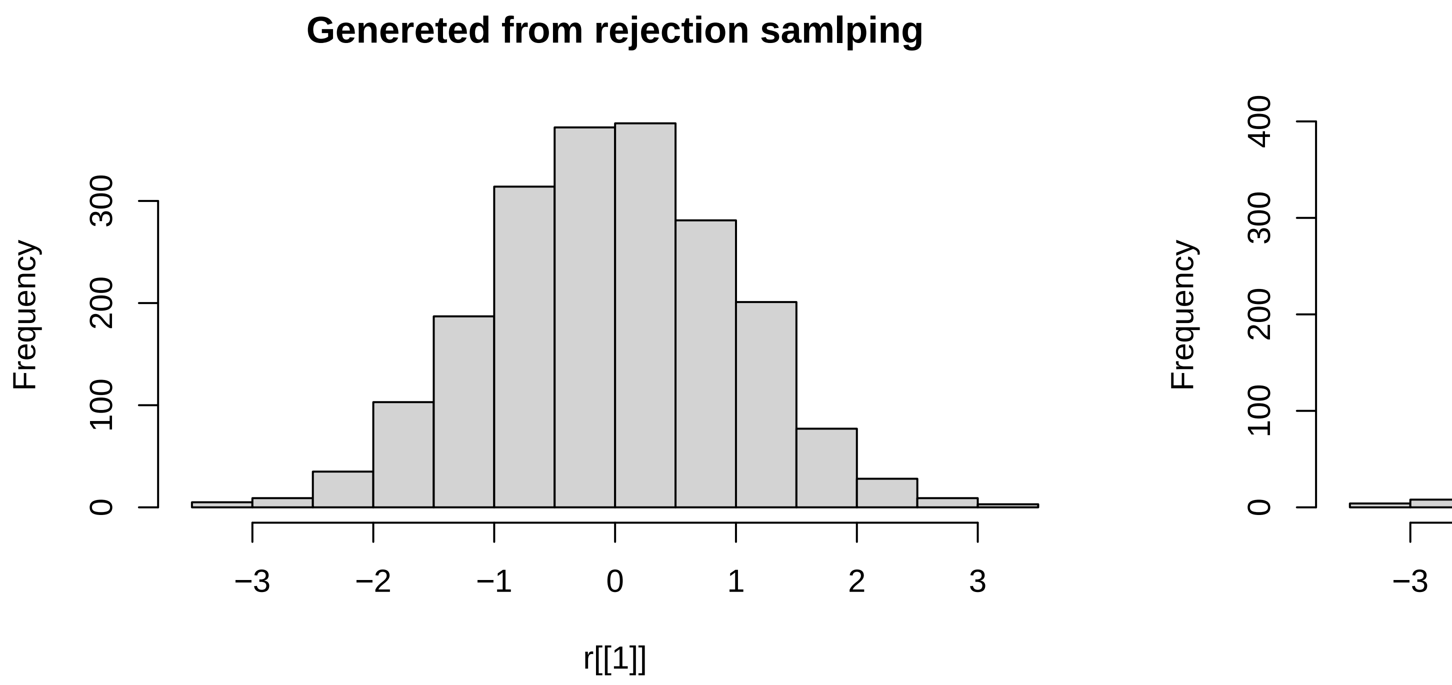
as:

```
# Integrate f(x) from -Inf to Inf
af <- integrate(normal_density, lower = -Inf, upper = Inf)
area_f <- af$value

# Integrate g(x) from -Inf to Inf
ag <- integrate(laplace_density, lower = -Inf, upper = Inf)
area_g <- ag$value

e_rate <- area_f/(area_g/alpha)
```

```
1-e_rate
```

```
## [1] 0.24
```

```
library(cowplot)

plot_grid(hist(r[[1]], main="Genereted from rejection samlping"),hist(rnorm(2000,0,1), main='Normal dist
```



The histograms looks quite similar, there is a difference in the x-axis between the plots which might create a bigger difference.

4 References

5 Appendix

```r
knitr::opts_chunk$set(echo = TRUE, message = FALSE, warning=FALSE, fig.width = 6, fig.height = 4)
library(ggplot2)
# define the function
f <- function(x) {
  ifelse(x < -1 | x > 1, 0,
         ifelse(-1 <= x & x <= 0, x + 1, 1 - x))
}


# generate x-grid and apply function
x_values <- seq(-2, 2, by = 0.01)
f_values <- sapply(x_values, f)
```

```r
# plot function
plot(x_values, f_values, type = 'l', col = 'red', lwd = 2,
     xlab = 'x', ylab = 'f(x)', main = 'Function density plot')
# envelope (uniform)
e <- function(x) {
  ifelse(x < -1 | x > 1, 0, 1)
}
# plot envelope
e_values <- sapply(x_values, e)
plot(x_values, f_values, type = 'l', col = 'red', lwd = 2,
     xlab = 'x', ylab = 'f(x)', main = 'Function density and envelope')
lines(x_values,e_values, col = "purple", lty = 2)
rejection_sampling <- function(n) {
  stopifnot(is.numeric(n))
  x <- c()

  for(i in 1:n){
    y <- runif(1, min = -1, max = 1)  # sample Y~g
    u <- runif(1)  # sample U~Unif(0,1)

    if(u <= f(y)/e(y)){
      x[i] <- y
    }
  }
  return(x)
}
composition_sampling <- function(Y, n, prob){
  negY <- -Y

  mu <- c(mean(Y),mean(negY))
  sigma <- c(sd(Y), sd(negY))
  prob <- c(prob, 1-prob)
  n <- n

  g <- sample(length(mu), n, replace = TRUE, p = prob)
  x <- rnorm(n, mean = mu[g], sd = sigma[g])

  return(x)
}
iud_sampling <- function(n) {
  u1 <- runif(n)
  u2 <- runif(n)

  x <- u1 - u2

  return(x)
}
set.seed(123)
```

```
# rejection sampling
sample_rej <- rejection_sampling(10000)
hist(sample_rej, freq = FALSE, main = "Rejection sampling", col = "lightblue", xlim = c(-1.5, 1.5), xlab
curve(f(x), add = TRUE, col = "red", lwd = 2) # add the true distribution
# individually uniformly distributed sampling
sample_iud <- iud_sampling(10000)
hist(sample_iud, freq = FALSE, main = "I.U.D sampling", col = "lightblue", xlim = c(-1.5, 1.5), xlab = "
curve(f(x), add = TRUE, col = "red", lwd = 2) # add the true distribution
# composition sampling
u <- runif(10000)
y <- 1-sqrt(1-u)
sample_comp <- composition_sampling(y, 10000, 0.5)
hist(sample_comp, freq = FALSE, main = "Composition sampling", col = "lightblue", xlim = c(-1.5, 1.5), x
curve(f(x), add = TRUE, col = "red", lwd = 2) # add the true distribution
var(sample_iud)

# this code was inspired wikipedia

laplace = function(n,lambda=1){

  y = runif(n,0,1) # draw from the uniform dist
  x <- c()

  # checking if the drawn values are bigger or smaller then 0.5 then making the calculations
  for(i in 1:n){

  if(y[i] >= 0.5){ # the condition in the inverse CFD
    x[i] <- lambda* log(2 * (1 - y[i]))
  }else{
    x[i] <- lambda* -log(2 * y[i])
  }

  }
  x # returning the generated CDF

}

# number of draws
n = 10000

# generated numbers
x <- laplace(10000)


# histogram
hist(x,breaks=30 )


# scaling DE(0,1)
```

```r
# Define Laplace density function g(x)
laplace_density <- function(x) {

  (1/2) * exp(-abs(x))

}

# Define standard normal density function f(x)
normal_density <- function(x) {
 1/(sqrt(2*pi)) * exp(-x^2/2)
}

# looping to find alpha, starting with 1 and going downwards to find optimal alpha
for(i in seq(1,0, by=-0.001)){
  if(all(laplace_density(seq(-100, 100, 0.01)) / i >= normal_density(seq(-100, 100, 0.01)))){
  alpha = i
  print(i)
  break
  }
}

cat('my alpha is', alpha)

e <- function(y,alpha){

laplace_density(y)/alpha


}




# reject function
reject <- function(n,alpha){
  vec <- c()
  # to generate n number of values and calculating all loops
  i <- 1
  nr_loops <- 0

  while (i<=n) {

    nr_loops <- nr_loops +1

    y <- laplace(1) # draw a number from the laplace dist

    u <- runif(1,0,1) # sampling a number between 0 and 1 to check the condition with
    if(u <= (dnorm(y,0,1)/e(y,alpha))){ # checking if to accept Y

      vec[i] <- y
      i <- i+1
    }




  }

  return(list(vec,nr_loops))
}

r <- reject(2000,alpha)

hist(r[[1]])

cat('The avarage rejection rate is: ',(r[[2]]- 2000)/r[[2]])

# Integrate f(x) from -Inf to Inf
af <- integrate(normal_density, lower = -Inf, upper = Inf)
area_f <- af$value

# Integrate g(x) from -Inf to Inf
ag <- integrate(laplace_density, lower = -Inf, upper = Inf)
area_g <- ag$value

e_rate <- area_f/(area_g/alpha)

1-e_rate

library(cowplot)

plot_grid(hist(r[[1]], main="Genereted from rejection samlping"),hist(rnorm(2000,0,1), main='Normal dist
```