Computer Lab 3

Group-19 - Dinesh and Umamaheswarababu Maddela

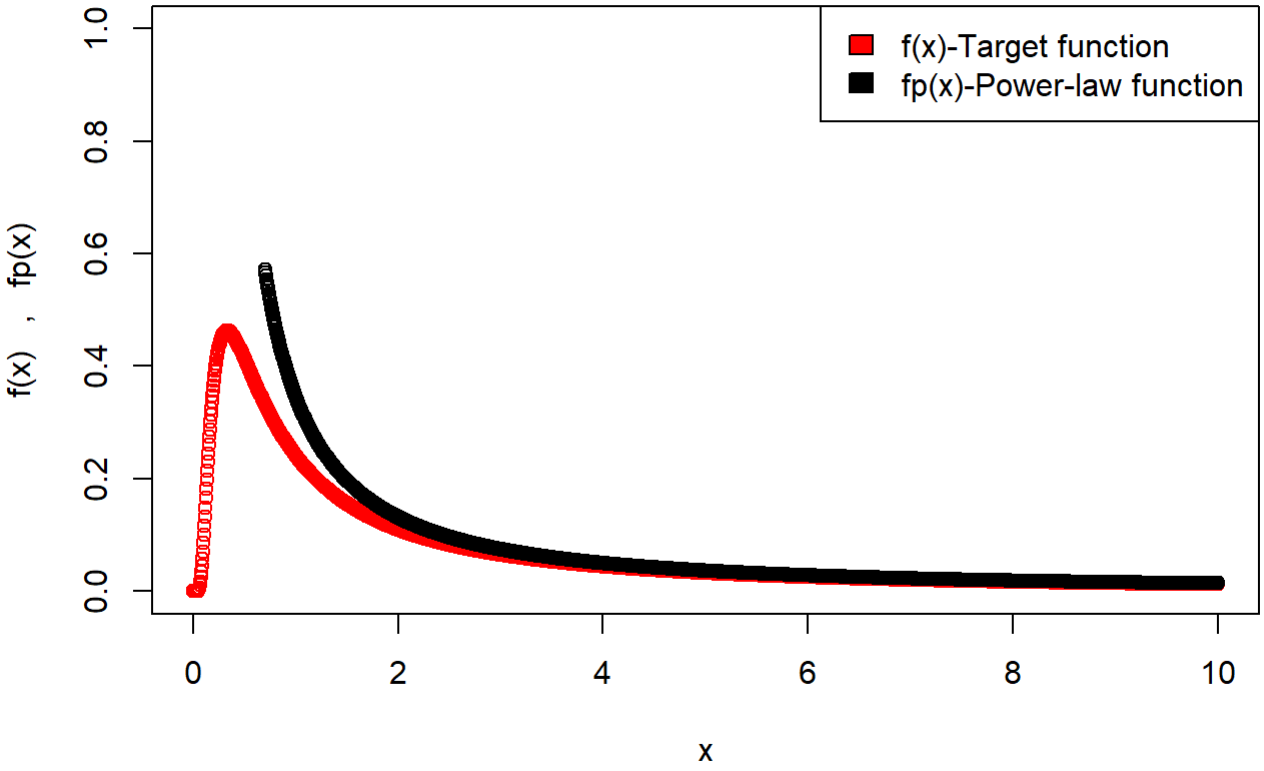
2023-02-7

Statement of Contribution : Question 1 was mostly done by Umamaheswarababu and Question 2 was mostly done by Dinesh

Question 1: Stable distribution

Question 1.1

Plottiing $f(x)$ and $f_p(x)$



The support of power-law distribution function $f_p(x)$ must include the support of target distribution function $f(x)$. From the plot we can say that this condition is satisfied but not for the entire interval of x. So we can use power-law distribution as majorizing density to sample the target distribution only for the interval $x \in (T_{min}, \infty)$.

The meaningful values of α and T_{min} are taken as 1.4 and 0.7 respectively such that the power-law distribution envelopes the target distribution for the period $[T_{min}, \infty]$

Since the power-law distribution does not include entire support of target distribution, we go for mixture density. Here we use Uniform distribution to include the target distribution for the period $[0, T_{min}]$.

The majorizing density for the entire period $[0, \infty]$, $g(x)$ is given by

$$g(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x)$$

Where,

$$f_1(x) = Unif(0, T_{min}) = \frac{1}{T_{min}} I_{(0, T_{min})}(x)$$

$$f_2(x) = f_p(x) = \frac{\alpha - 1}{T_{min}} \left(\frac{x}{T_{min}} \right)^{-\alpha} I_{(T_{min}, \infty)}(x)$$

α_1 and α_2 are the weights given to each distribution.

α_1 is given by

$$\alpha_1 = \int_0^{T_{min}} f(x) dx$$

$$\alpha_1 = \int_0^{T_{min}} c(\sqrt{2\pi})^{-1} e^{\frac{-c^2}{2x}} x^{\frac{-3}{2}} dx$$

α_2 is given by

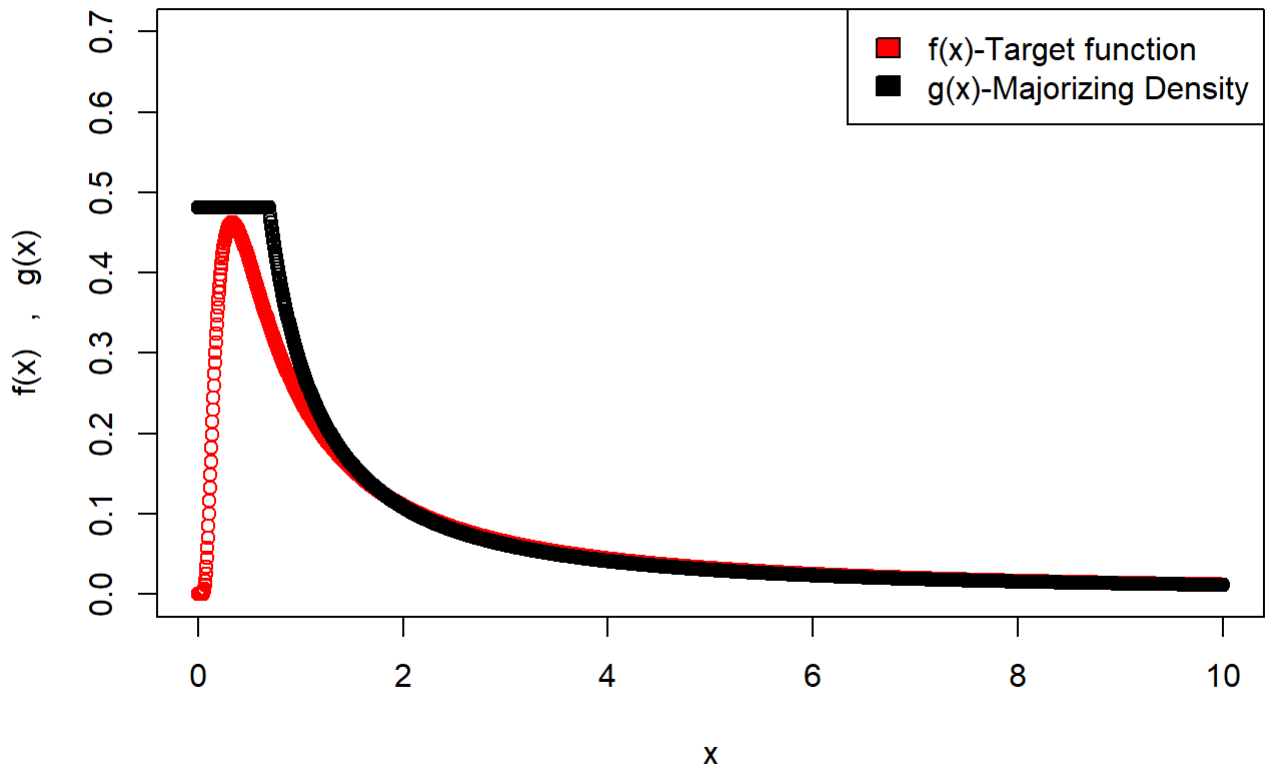
$$\alpha_2 = \int_{T_{min}}^{\infty} f(x) dx$$

$$\alpha_2 = \int_{T_{min}}^{\infty} c(\sqrt{2\pi})^{-1} e^{\frac{-c^2}{2x}} x^{\frac{-3}{2}} dx$$

Solving the above integrals we get $\alpha_1 = 0.337$ and $\alpha_2 = 0.825$ for $T_{min} = 0.7$ and $\alpha = 1.4$

Now majorizing density $g(x)$ is given by

$$g(x) = (0.337) \frac{1}{T_{min}} + (0.825) \frac{\alpha - 1}{T_{min}} \left(\frac{x}{T_{min}} \right)^{-\alpha}$$



From the above figure, we can say that majorizing density g(x) includes the support of target distribution f(x) for the entire period $[0, \infty]$ so we can use this majorizing density for implementation of acceptance-rejection algorithm.

Question 1.2

Implementation of Acceptance-rejection algorithm

From lecture slides the algorithm is as follows

1: while X not generated do

2: Generate Y ~ fY

3: Generate U ~ Unif(0, 1)

4: if U ≤ fX(Y)/(cfY (Y)) then

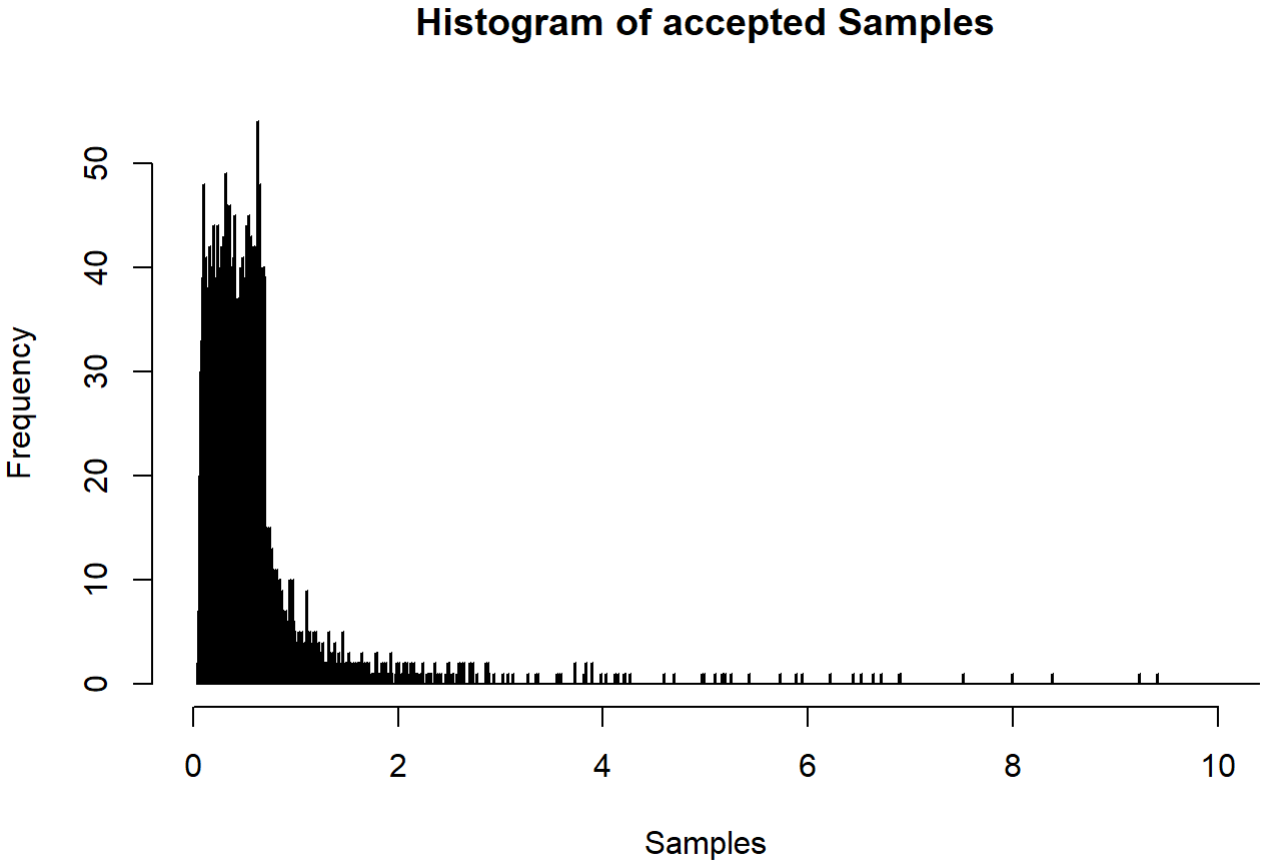
5: X = Y

6: Set X is generated

7: end if

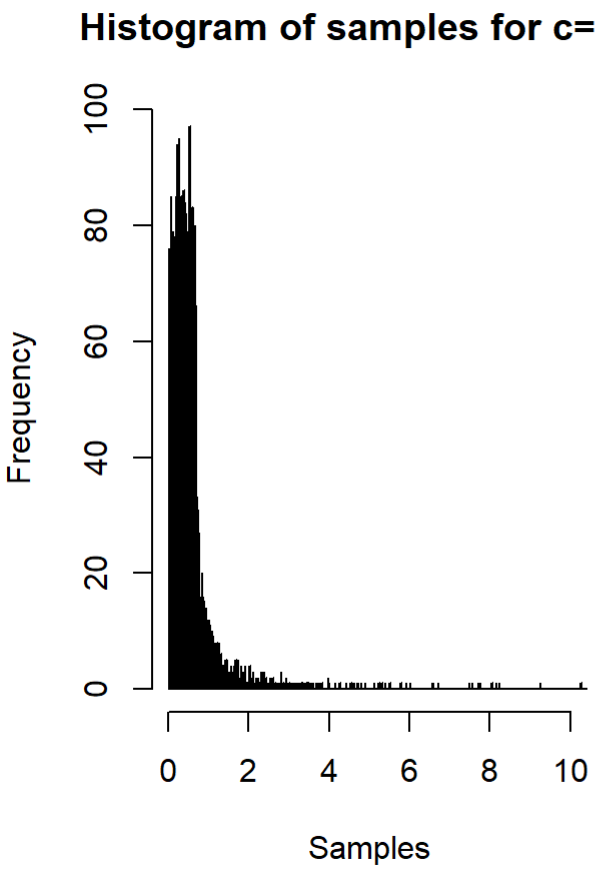
8: end while

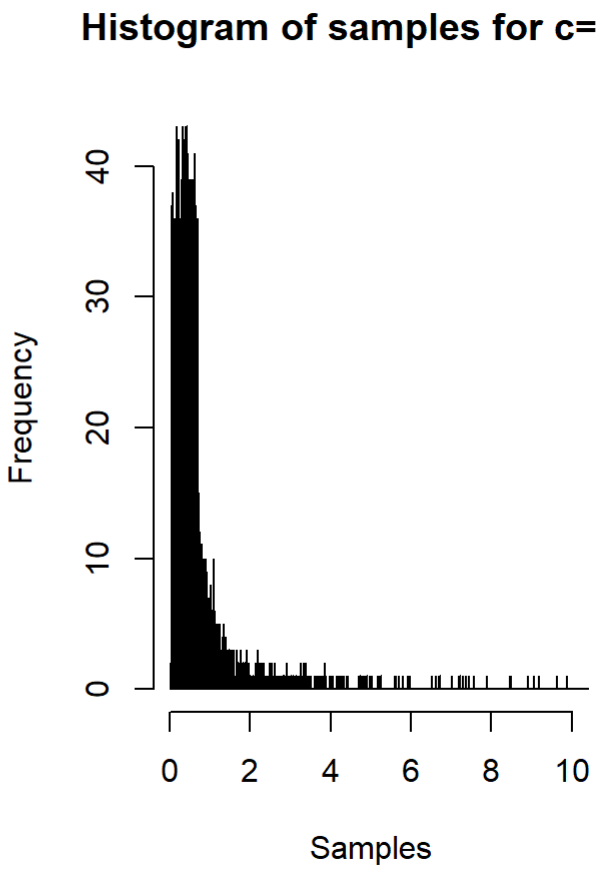
The histogram of generated samples using the above algorithm is given below

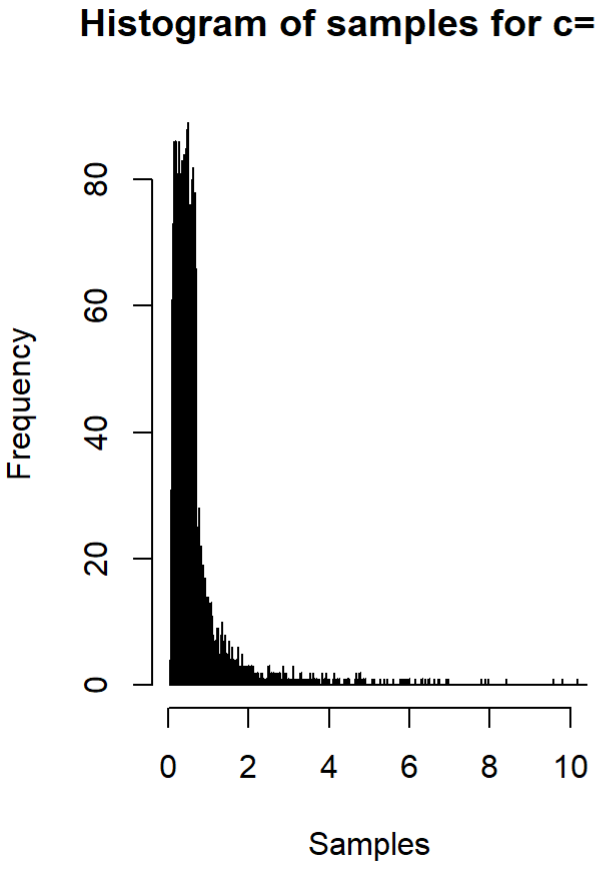


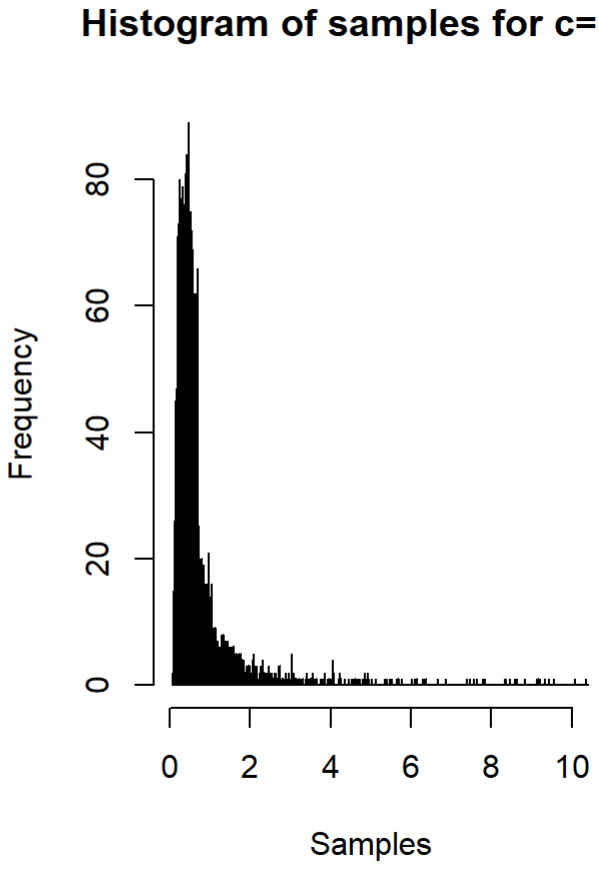
Question 1.3

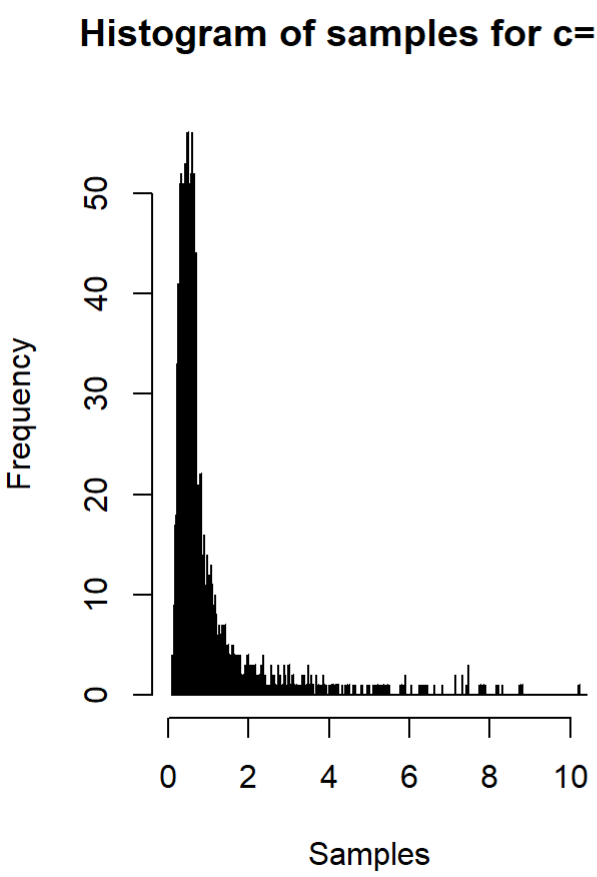
Generating a large sample using your implemented sampler, for different choices of c

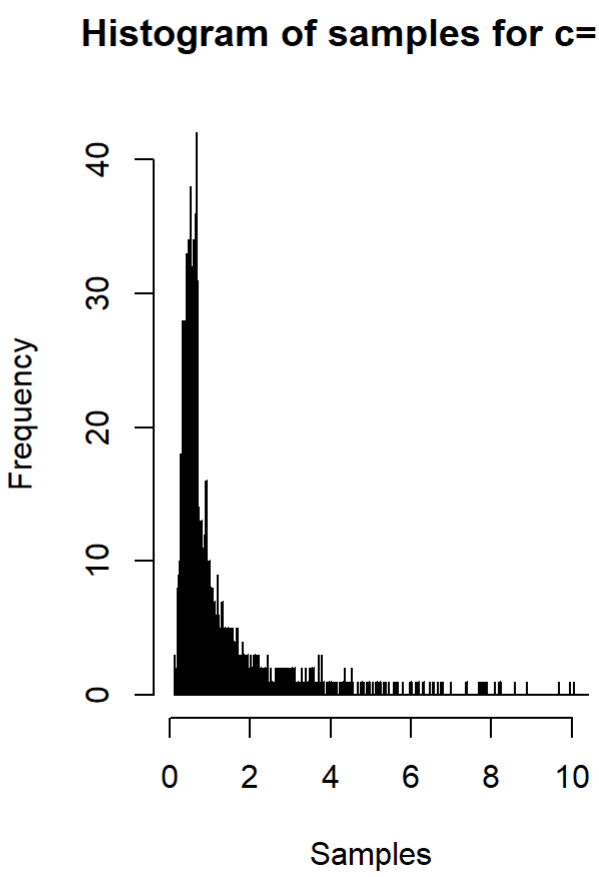

Histogram of samples for c= 0.7
Frequency
0
20
40
60
80
100
0
2
4
6
8
10
Samples

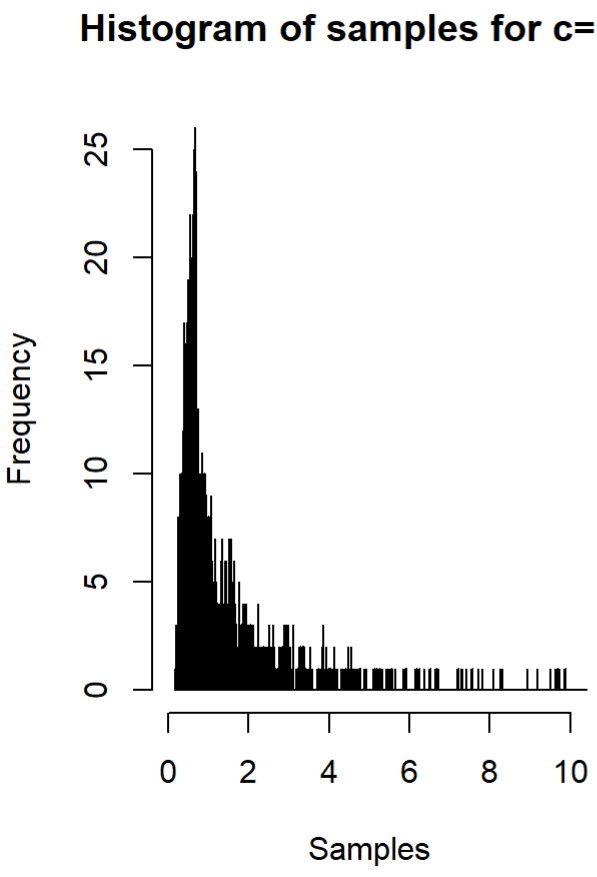

Histogram of samples for c= 0.9
Frequency
0
10
20
30
40
0
2
4
6
8
10
Samples

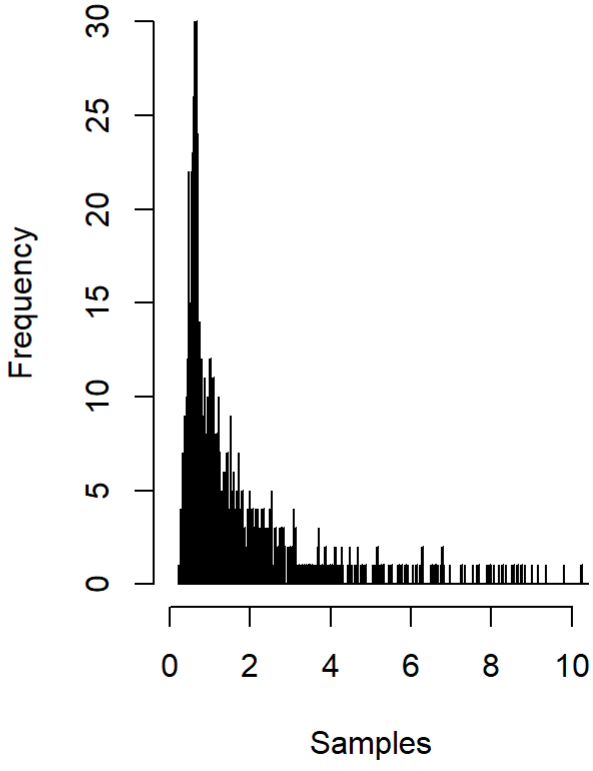

Histogram of samples for c= 1.1
Frequency
0
20
40
60
80
0
2
4
6
8
10
Samples

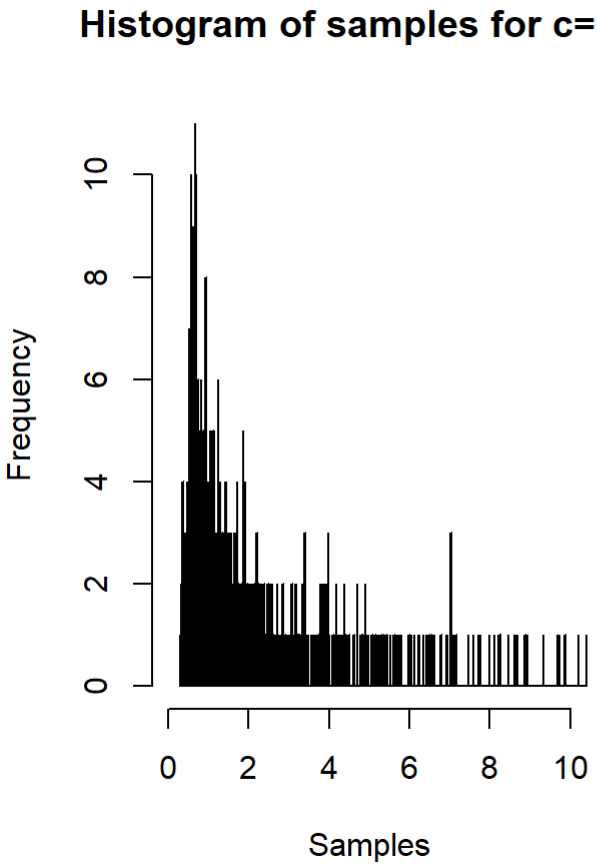

Histogram of samples for c= 1.3
Frequency
0
20
40
60
80
0
2
4
6
8
10
Samples

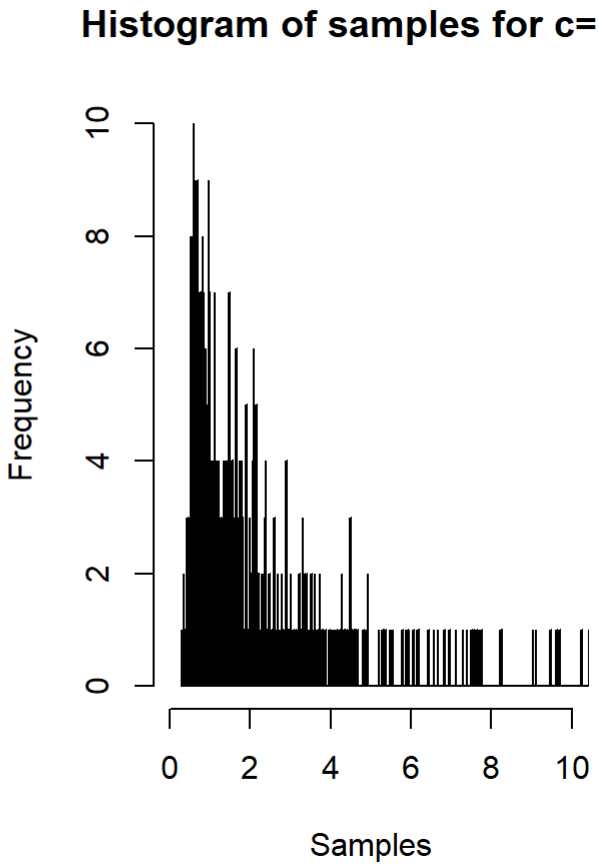

Histogram of samples for c= 1.5
Frequency
0
10
20
30
40
50
0
2
4
6
8
10
Samples


Histogram of samples for c= 1.7
Frequency
0
10
20
30
40
0
2
4
6
8
10
Samples


Histogram of samples for c= 1.9
Frequency
0
5
10
15
20
25
0
2
4
6
8
10
Samples


Histogram of samples for c= 2.1
Frequency
0
5
10
15
20
25
30
0
2
4
6
8
10
Samples





Mean, variance and rejection rate.

```
##      c      Mean   Variance Rejection.rate
## 1  0.7 0.5011434  0.4357771         43.495
## 2  0.9 0.5387661  0.4965916         44.365
## 3  1.1 0.5813847  0.8998412         47.365
## 4  1.3 0.6647274  1.9839962         55.395
## 5  1.5 0.8146414  1.8393212         71.390
## 6  1.7 1.1080032  6.8646746         82.170
## 7  1.9 1.3223611  4.9128676         88.575
## 8  2.1 1.7633317 10.6115831         92.455
## 9  2.3 2.2881006 13.9360978         94.520
## 10 2.5 2.4293209 22.6877446         96.255
```

We observe that rejection rate increases with increase in value of c

Question 2 : Laplace distribution

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.

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

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

where, x is a real number. μ is a location parameter. α is a real positive number.

The cumulative density function (CDF) of a continuous random variable X is defined as:

$$F(x) = \int_{-\infty}^{x} f(x)dx, \qquad -\infty < x < \infty$$

The cumulative distribution for the double exponential distribution:

For $x > \mu$:

$$F(x) = \int_{-\infty}^{x} \frac{\alpha}{2} e^{-\alpha(x-\mu)} dx, \quad x > \mu$$

$$F(x) = 1 - \int_{x}^{\infty} \frac{\alpha}{2} e^{-\alpha(x-\mu)} dx$$

Integrating with respect to x,

$$F(x) = 1 - \frac{1}{2} e^{-\alpha(x-\mu)}$$

For $x \leq \mu$:

$$F(x) = \int_{-\infty}^{x} \frac{\alpha}{2} e^{\alpha(x-\mu)} dx, \quad x \leq \mu$$

Integrate with respect to x, we have:

$$F(x) = \frac{1}{2} e^{\alpha(x-\mu)}$$

The inverse CDF technique:

The following steps demonstrates how to solve the equation F(x)=U for x to obtain a random number from a double-exponential distribution given a uniform random number in U between 0 and 1(i.e $U \sim U(0,1)$):

For $U > 1/2$, we have:

$$U = 1 - \frac{1}{2} e^{-\alpha(x-\mu)}$$

Solving for x, we will get:

$$x = \mu - \frac{ln(2-2U)}{\alpha}$$

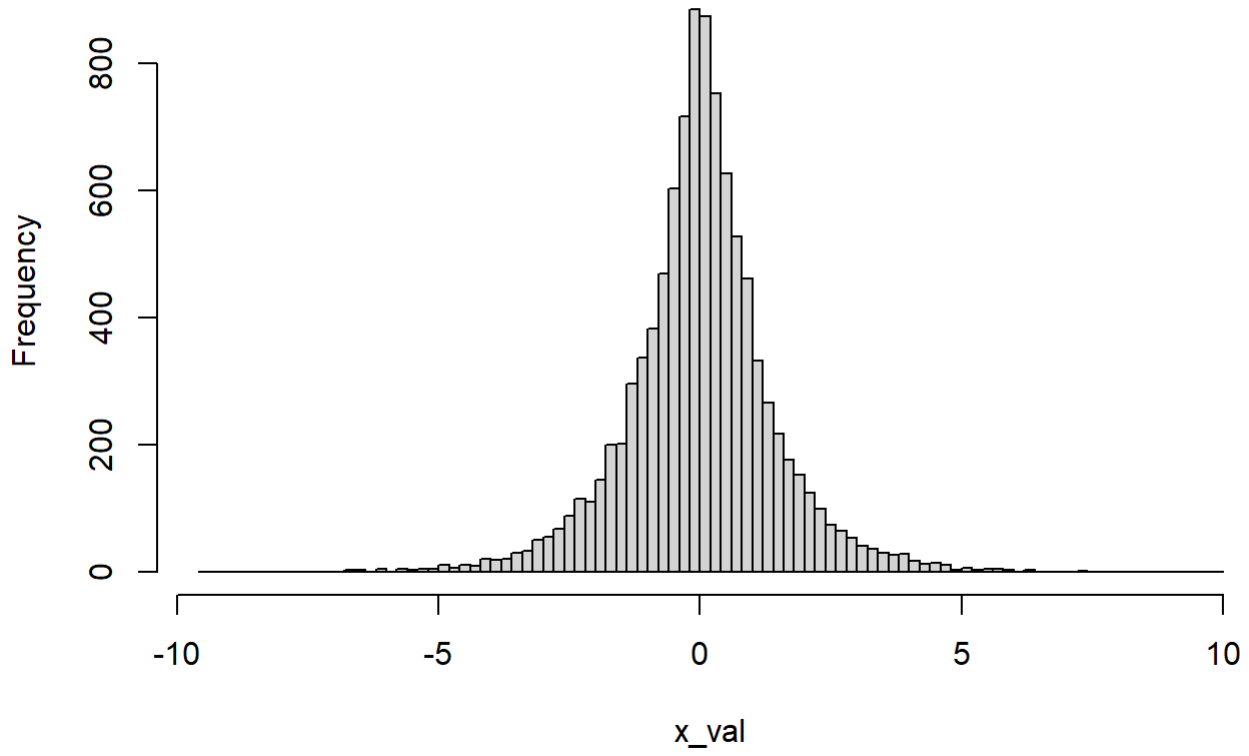
For $U \leq 1/2$, we have:

$$U = \frac{1}{2}e^{\alpha(x-\mu)}$$

Solving for x, we will get:

$$x = \mu + \frac{ln(2U)}{\alpha}$$

```
# Generating CDF function
inv_cdf = function(mu,alpha){
  u = runif(1,0,1)
  if (u <= 0.5){
    x=mu+(log(2*u))/alpha
  }else{
    x=mu-(log(2-2*u))/alpha
  }
  return(x)
}

x_val <- c()
for (i in 1:10000){
  x_val[i] <-inv_cdf(0,1)
  x_val
}

## Plotting the histogram
hist(x_val, breaks = 100,main = "Plotting 10000 random numbers using double exponential(Laplace)
distribution")
```



The histogram makes the result appear reasonable because it almost exactly matches the plot of the Laplace distribution. The random numbers produced are legitimate.

Use the Acceptance/rejection method with DE(0,1) as a majorizing density to generate N(0,1) variables. Explain step by step how this was done. How did you choose constant c 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 acceptance/rejection 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.

$$c * g_y(x) \geq f_x(x) \qquad for\ \ all\ \ x$$

Where, the density $g_y(x)$ is *majorizing* density or the *proposal* density $c * g_y(x)$ is the majorizing function. $f_x(x)$ is the *target* density.

Majorizing density $f_y(x) \sim DE(0,1)$ is,

$$g_y(x) = \frac{1}{2}e^{-|x|}$$

Target density $f_x(x) \sim N(0,1)$ is,

$$f_x(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

Evaluating the majorizing constant c,

$$c \geq \frac{f_x(x)}{g_y(x)}$$

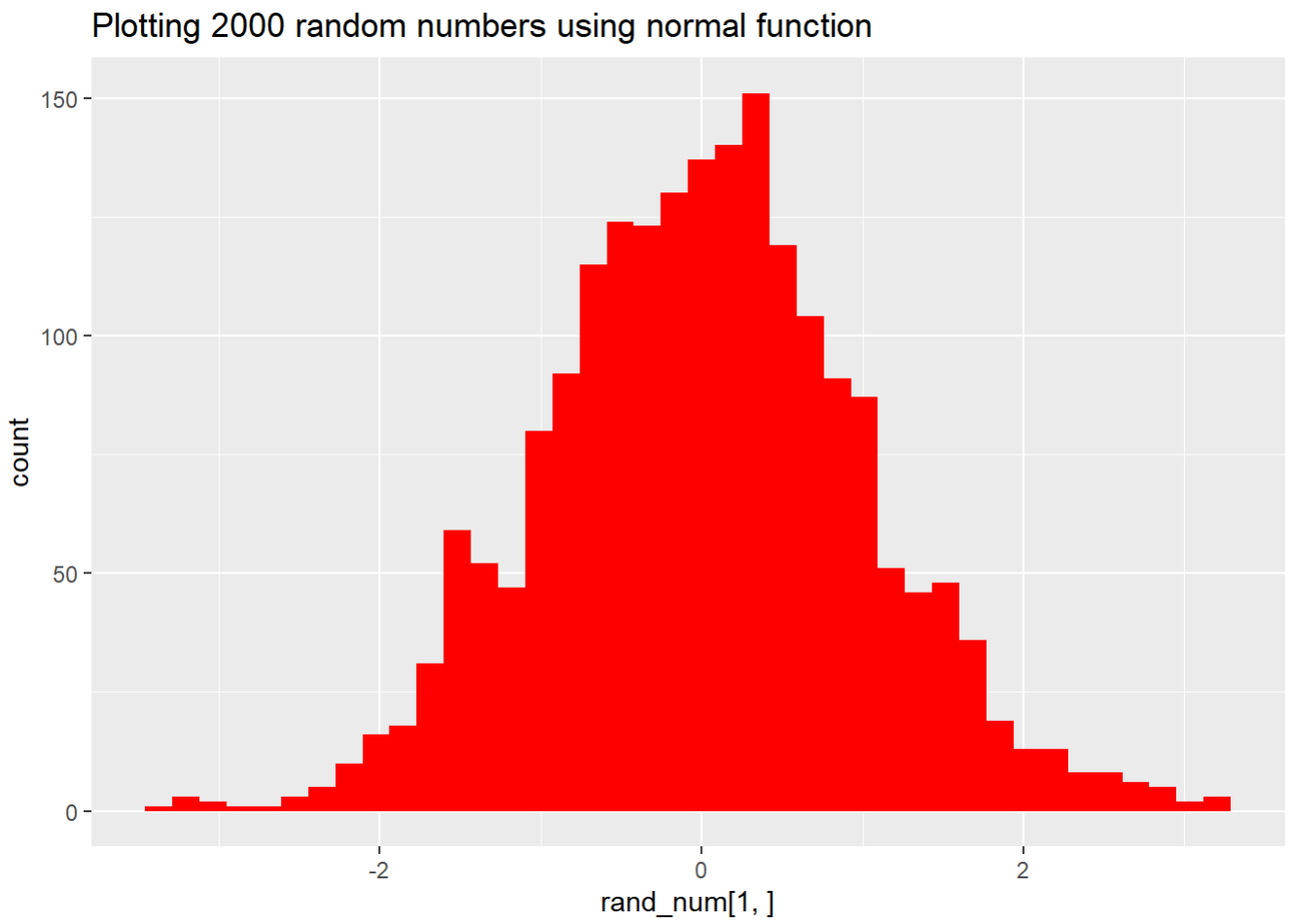
Which is,

$$c \geq \sqrt{\frac{2}{\pi}} e^{-\frac{x^2}{2}+|x|}$$

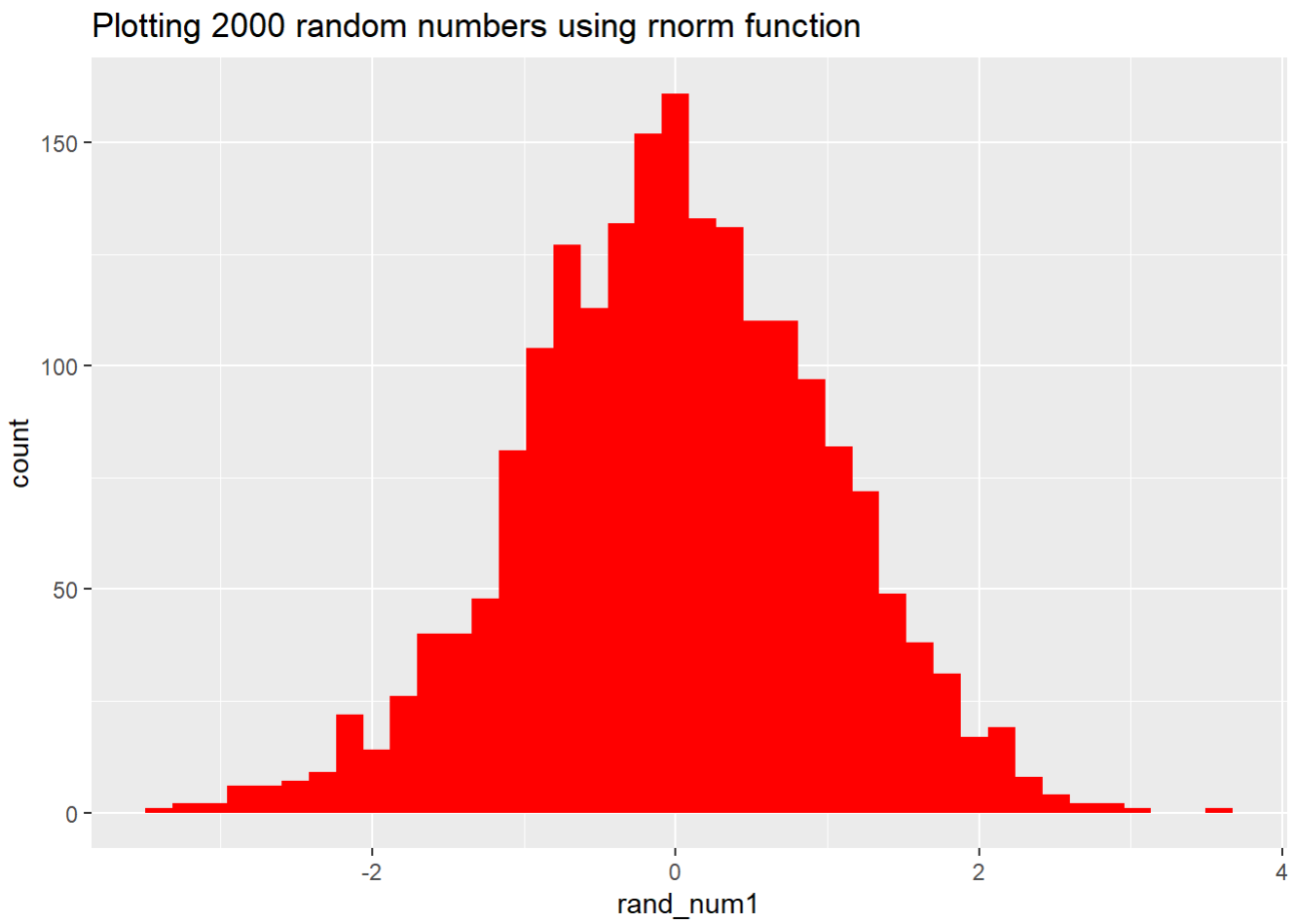
We obtained the maximum at x=1 by differentiating the above mentioned formula and setting it to zero. The c value will therefore be,

$$\sqrt{\frac{2}{\pi}} e^{\frac{1}{2}} = \sqrt{\frac{2e}{\pi}}$$

```
# generate normal distribution function from laplace distribution
gen_normal<-function(c){
  x<-NA
  num_reject<-0

  while (is.na(x)){
    y<-inv_cdf(0,1)
    u<-runif(1)
    f_y = 2/sqrt(2*pi) * exp(-y^2/2)
    g_y = exp(-abs(y))

    if (u <= f_y / (c* g_y))
    {
      x<-y
    }
    else
    {
      num_reject<-num_reject+1
    }
  }
  return(c(x,num_reject))
}

# constant c
c <- sqrt(2 *exp(1)/pi)

# generate numbers using normal function
library(ggplot2)
rand_num <- sapply(rep(c,2000),gen_normal)
ggplot() + geom_histogram(aes(rand_num[1,]),fill="red",bins = 40)+
  ggtitle("Plotting 2000 random numbers using normal function")
```




```
# generate numbers using rnorm() function
rand_num1 <- rnorm(2000)
ggplot() + geom_histogram(aes(rand_num1),fill="red",bins = 40)+
  ggtitle("Plotting 2000 random numbers using rnorm function")
```




```
# average rejection rate (R)
num_reject <- sum(rand_num[2,])
average_rejection <- num_reject / (num_reject+ncol(rand_num))
paste("Average rejection rate R=", average_rejection)
```

```
## [1] "Average rejection rate R= 0.240410178503608"
```

```
# expected rejection rate (ER)
expected_rejection <- (c-1) / c
paste("Expected rejection rate ER=", expected_rejection)
```

```
## [1] "Expected rejection rate ER= 0.23982654946686"
```

The values of the expected rejection rate and average rejection rate appear to be nearly identical with a small differences. And by seeing the histogram plot of acceptance/rejection method and the the method using rnorm() function, we can say that moreover it looks similar and they are symmetric.

Reference

1. http://www.columbia.edu/~ks20/4703-Sigman/4703-07-Notes-ARM.pdf (http://www.columbia.edu/~ks20/4703-Sigman/4703-07-Notes-ARM.pdf)

2. https://math.stackexchange.com/questions/2021342/how-is-this-inverse-function-calculated-laplace-distribution (https://math.stackexchange.com/questions/2021342/how-is-this-inverse-function-calculated-laplace-distribution)
3. https://en.wikipedia.org/wiki/Laplace_distribution#Cumulative_distribution_function (https://en.wikipedia.org/wiki/Laplace_distribution#Cumulative_distribution_function)

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
#one-sided strictly stable distribution
f_x <- function(x, c){
  res <- (c/sqrt(2 * pi)) * (exp(- (c^2/(2*x)))) *( x^(-3/2))
  return(res)
}
#power-law distribution
f_p_x <- function(x, alpha, t_min){
  res <- ((alpha-1)/t_min) * (x/t_min)^-alpha
  return(res)
}
x<-seq(0,10,0.005) #taking sequence of x values
f_x_values<- f_x(x,c=1)  #choosing c=1
plot(x,f_x_values, ylim=c(0,1), col="red", ylab="f(x)   ,   fp(x)")
f_p_x_values<-f_p_x(x=seq(0.7,10,0.005),alpha = 1.4, t_min = 0.7) #choosing alpha=1.4,t_min=0.7
points(seq(0.7,10,0.005),f_p_x_values)
legend("topright", legend=c("f(x)-Target function", "fp(x)-Power-law function"), fill = c("re
d","black"))
g_x <- function(x, alpha, t_min){
  a<-0.337/t_min
  b<-0.825*(alpha-1)/t_min
  c<-(x/t_min)^(-alpha)
  res <- a+b*c
  return(res)
}
x<-seq(0,10,0.005) #taking sequence of x values
f_x_values<- f_x(x,c=1)  #choosing c=1
plot(x,f_x_values, ylim=c(0,0.7), col="red", ylab="f(x)   ,   g(x)")

g_x_values=NULL
for (i in x){
  if ( i<=0.7){
    g_x_values<-c(g_x_values,0.337/0.7)
  }
  else{
    g_x_values<-c(g_x_values,0.825*f_p_x(x=i,alpha=1.4,t_min=0.7))
  }
}

points(x,g_x_values)
legend("topright", legend=c("f(x)-Target function", "g(x)-Majorizing Density"), fill = c("re
d","black"))

#Acceptance - rejection algorithm
rand_samples<-function(c,n){  #n-> no. of samples
  x<-seq(0.1,10,0.005)
  C<- max(f_x(x,c=1)/g_x(x,alpha=1.4,t_min = 0.7)) #majorizing constant
  rand<-NULL
  rejection<-0
  for (i in 1:n){
    u<-runif(1,0,1)
    toss<-sample(c(0,1),1) #coin toss to choose a sample either from uniform part or powerlaw pa
```

```
rt
    y<- ifelse(toss==0,runif(1,0,0.7),poweRlaw::rplcon(1,xmin=0.7, alpha = 1.4))#uniform if toss
=0; powerlaw if toss=1
    if(u<=f_x(y,c)/C*g_x(x=y,alpha=1.4,t_min=0.7)){
      rand<-c(rand,y)
    }else{ rejection<-rejection+1}
  }
  return(list(rand,rejection))
}

samples<- rand_samples(c=1,n=10000)
hist(samples[[1]],breaks=10000,xlim = c(0,10),xlab="Samples",main="Histogram of accepted Sample
s")
#generating a large sample for different choices of c={0.7,0.9,....,2.5}
par(mfrow=c(1,2))
Mean<-NULL
Variance<-NULL
Rejection_rate<-NULL
for (c in seq(0.7,2.5,0.2)){
  sample<-rand_samples(c,n=20000) # generating 20000 samples
  hist(sample[[1]],breaks=10000,xlim = c(0,10),xlab="Samples",main=paste("Histogram of samples f
or c=",c))
  Mean<-c(Mean,mean(sample[[1]]))
  Variance<-c(Variance,var(sample[[1]]))
  Rejection_rate<-c(Rejection_rate,100*sample[[2]]/20000)
}

data<-data.frame("c"=seq(0.7,2.5,0.2),"Mean"=Mean,"Variance"=Variance,"Rejection rate"=Rejection
_rate)
data
# Generating CDF function
inv_cdf = function(mu,alpha){
  u = runif(1,0,1)
  if (u <= 0.5){
    x=mu+(log(2*u))/alpha
  }else{
    x=mu-(log(2-2*u))/alpha
  }
  return(x)
}

x_val <- c()
for (i in 1:10000){
  x_val[i] <-inv_cdf(0,1)
  x_val
}

## Plotting the histogram
hist(x_val, breaks = 100,main = "Plotting 10000 random numbers using double exponential(Laplace)
distribution")
# generate normal distribution function from laplace distribution
gen_normal<-function(c){
```

```
  x<-NA
  num_reject<-0

  while (is.na(x)){
    y<-inv_cdf(0,1)
    u<-runif(1)
    f_y = 2/sqrt(2*pi) * exp(-y^2/2)
    g_y = exp(-abs(y))

    if (u <= f_y / (c* g_y))
    {
      x<-y
    }
    else
    {
      num_reject<-num_reject+1
    }
  }
  return(c(x,num_reject))
}

# constant c
c <- sqrt(2 *exp(1)/pi)

# generate numbers using normal function
library(ggplot2)
rand_num <- sapply(rep(c,2000),gen_normal)
ggplot() + geom_histogram(aes(rand_num[1,]),fill="red",bins = 40)+
  ggtitle("Plotting 2000 random numbers using normal function")

# generate numbers using rnorm() function
rand_num1 <- rnorm(2000)
ggplot() + geom_histogram(aes(rand_num1),fill="red",bins = 40)+
  ggtitle("Plotting 2000 random numbers using rnorm function")


# average rejection rate (R)
num_reject <- sum(rand_num[2,])
average_rejection <- num_reject / (num_reject+ncol(rand_num))
paste("Average rejection rate R=", average_rejection)

# expected rejection rate (ER)
expected_rejection <- (c-1) / c
paste("Expected rejection rate ER=", expected_rejection)
```