Lab 3

Shwetha Vandagadde, Suhani Ariga and Hoda Fakharzadeh

11/21/2020

Question 1: Stable distribution

1

```
f = function(x,c){
  res = 0
  res = c * ((sqrt(2*pi))^-1) * exp(-(c^2)/(2*x)) * x^(-3/2)
  res[x<=0]<-0
  return(res)
}
fp = function(x, a, tmin) {
  return(((a - 1) / tmin) * (x / tmin)^(-a))
  }
c = 2
tmin =  1.33
a = 1.3
x = seq(0,50,length.out = 100)
plot(x, f(x,c), type = "l", ylim = c(0,0.5),xlab = "", ylab = "")
lines(x,fp(x,a,tmin),col="red")
title("Powerlaw and target function")
legend(30,0.5,legend = c("Powerlaw","Target function"),
       lty = 1, col = c("red","black"))
```

Powerlaw and target function
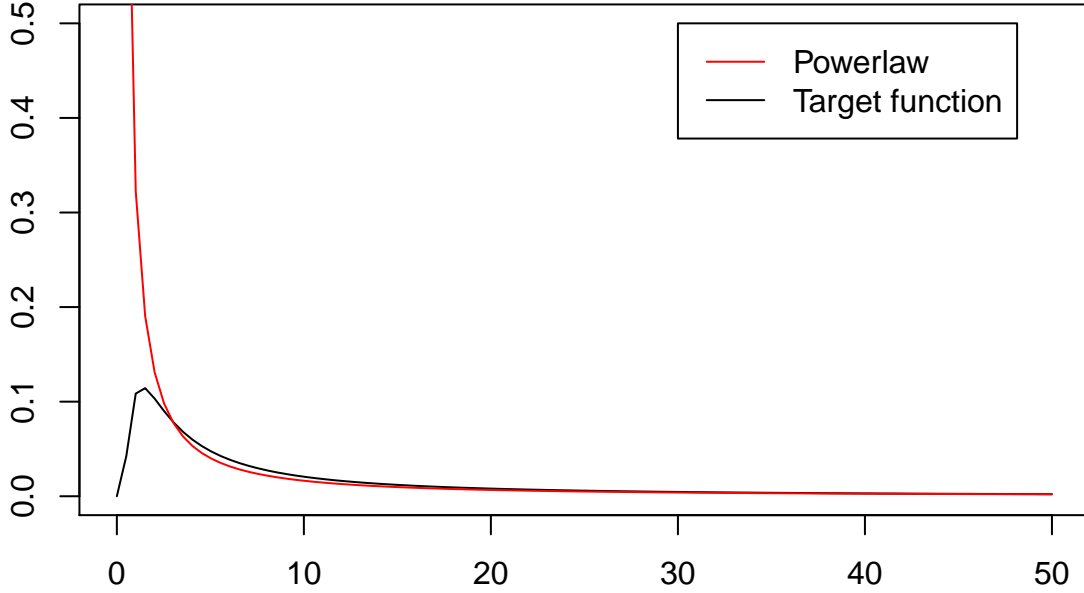
The support for power law is from tmin to infinity , but our target function is on support from 0 to infinity. Hence we cannot use power law by itself. However we can combine it with uniform distribution for 0 to tmin support. Since the max(f(x)) is achieved when the value of x is c^2/3, we have chosen tmin value to be the same, because powerlaw is strictly decreasing function and maximum of it will be at tmin.

Majorizing density function is combination of uniform density and powerlaw density function.

$$Majority density = (probability*uniform density*1[0, tmin]) + (1 - probability*powerlaw density*1[tmin, \inf])$$

$$where probablity = \int_0^{tmin} f(x)dx$$

Then we can find majorizing constant as follows

$$Majoring Constant = max\left(\frac{f(x)}{majority density(x)}\right)$$

We know that our majority density takes 2 parts , 1> When x<tmin , here f(x) = target function and g(x) is uniform density. Here we just need to find max(f(x)) as g(x) would just be a constant here , and max(f(x)) is achieved when x is c^2/3.

$$f(x) = c(\sqrt{2\pi})^{-1}e^{\frac{-c^2}{2x}}x^{\frac{-3}{2}}1_{(0,\infty)}(x)$$

$$logf(xlog(c) - log(\sqrt{2}\pi) - \frac{c^2}{2x} - \frac{3}{2}\log(x)$$

$$\frac{\partial}{\partial x}\log(f(x)) = \frac{c^2}{2x^2} - \frac{3}{2x} = 0$$

$$x_{maj} = \frac{c^2}{3}$$

2> When x > tmin, here g(x) is powerlaw

$$\frac{\partial}{\partial x}\frac{f(x)}{g(x)} = \frac{3^{a-1}c^3 \frac{e^{-c^2}}{2x}\left(\frac{x}{c^2}\right)^a\left((2a-3)x+c^2\right)}{2\sqrt{2\pi}(a-1)x^{7/2}} = 0$$

$$x_{maj} = \frac{c^2}{3-2a}$$

Now we can see the limitation on alpha , a should be in the range from 1 and 1.5. Alpha greater than 1.5 will end up giving negetive majorising constant. Hence we are fixing our alpha to 1.3.

```
p = 0.0832 #for c = 2
max = max(f(x,c))
nd = function(x,tmin,a,p){
  res = c()
  for(i in 1:length(x)){
    if(x[i]>=0 && x[i]<= tmin){
      res[i] = p * dunif(x[i],0,tmin)
    }
    if(x[i]>tmin){
      res[i] = (1-p) * fp(x[i],a,tmin)
    }
  }
  return(res)
}

cmaj = function(c,a) {
  xmaj <- (c^2)/(3-2*a)
  return((f(xmaj,c)/(nd(xmaj,tmin,a,p))))
}
c_maj = cmaj(c,a)

plot(f(x,c),type = "l", col = "blue",ylim = c(0,0.5))
lines(nd(x,tmin,a,p)*c_maj, col = "red")
```
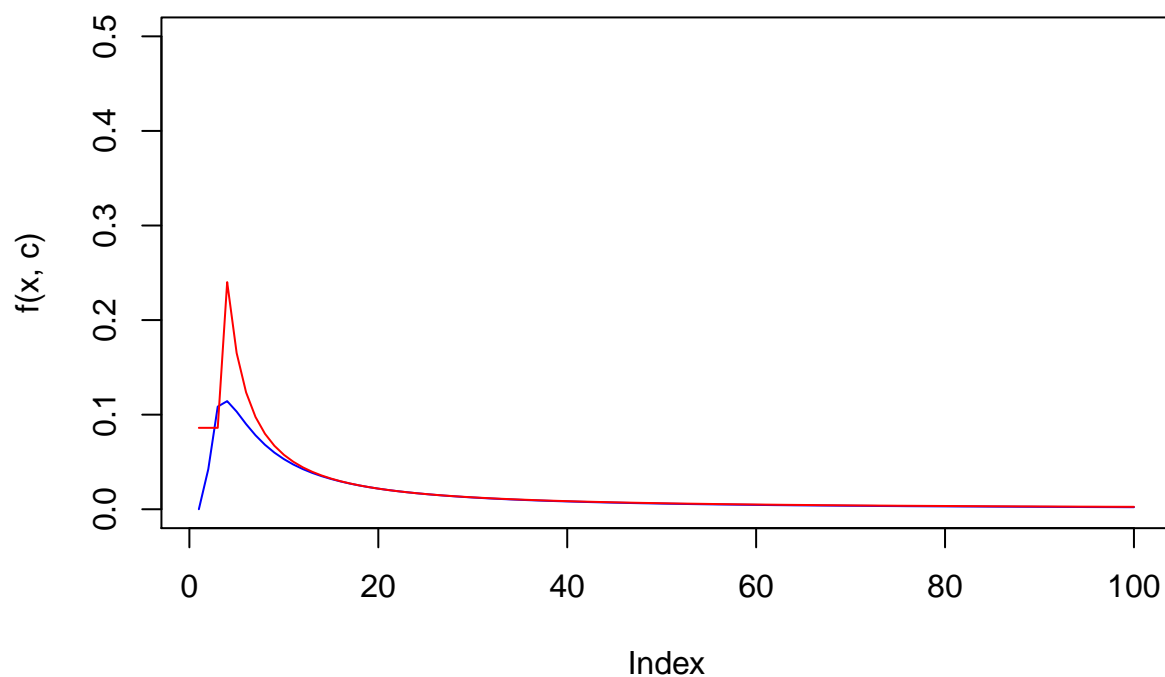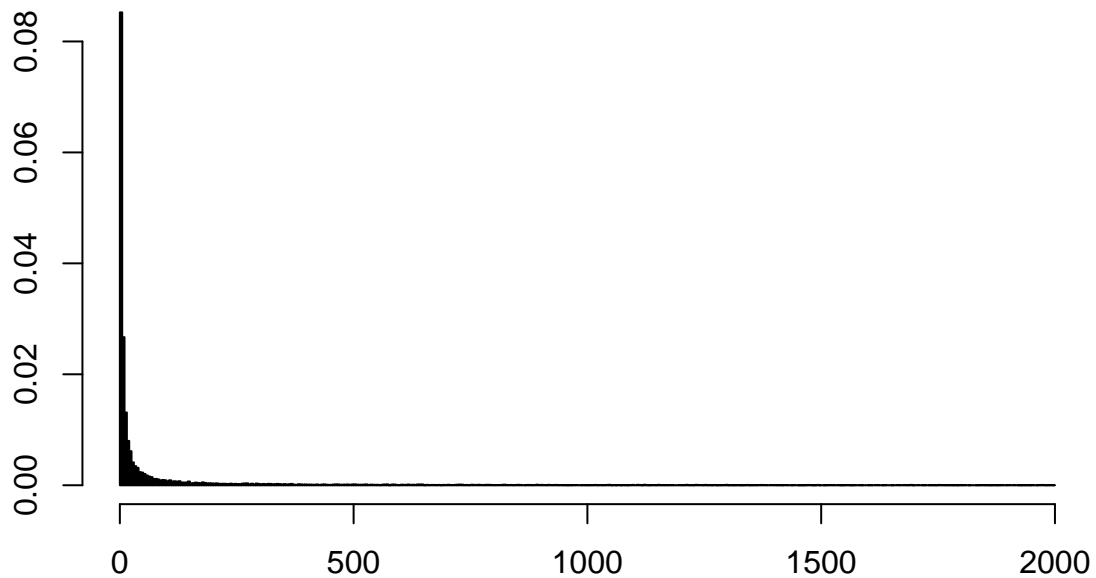
2

Acceptance-rejection algorithm :

```r
rmajorizing<-function(n,tmin,a){
  sapply(1:n,function(i){
    res<-NA
    component<-sample(1:2,1,prob = c(0.0833,1-0.0833))
    if(component==1){res<-runif(1,0,tmin)}
    if(component==2){res<-rplcon(1,tmin,a)}
    res
  })
}
Nsample<-20000
num_histbreaks<-1000
hv = rmajorizing(Nsample,tmin,a)
hist(hv[hv<2000],breaks=300,col="black",xlab="",ylab="",
     main="HistMajorizing",freq=FALSE)
```

HistMajorizing


```r
accept_reject<-function(c,c_maj,t){
  x<-NA
  num_reject<-0
  while (is.na(x)){
    y<-rmajorizing(1,t,a)
    u<-runif(1)
    if (u<=f(y,c)/(c_maj*nd(y,a,tmin = t,p))){
      x<-y
    } else{
      num_reject<-num_reject+1
    }
  }
 res = c(x,num_reject)
 return(res)
}
```

3

Generating large samples from above sampler to different values of c. When we compare the results of f(x) for different

```r
set.seed(12345)
Nsample = 10000
c = c(2,9,15)
p = 0.0833
```

```
t = c(4/3,81/3,15/3)
cm = sapply(c,cmaj,a)
mean=var=rejection_rate=c()
for(i in 1:length(c)){
tmin = t[i]
fx_acceptreject<-sapply(rep(c[i],Nsample),accept_reject,c_maj = cm[i], t = tmin)
histvalue = fx_acceptreject[1,]
histvalue = histvalue[histvalue<2000]
hist(histvalue,col="white",border = "black",breaks=300,xlab="",
     ylab="",freq=FALSE,main="")
title(capture.output(cat("Accept/Reject samples when c is ", c[i])))
mean =cbind(mean, mean(fx_acceptreject[1,]))
var = cbind(var,var(fx_acceptreject[1,]))
rejection_rate = cbind(rejection_rate,sum(fx_acceptreject[2,])/Nsample)
}
```
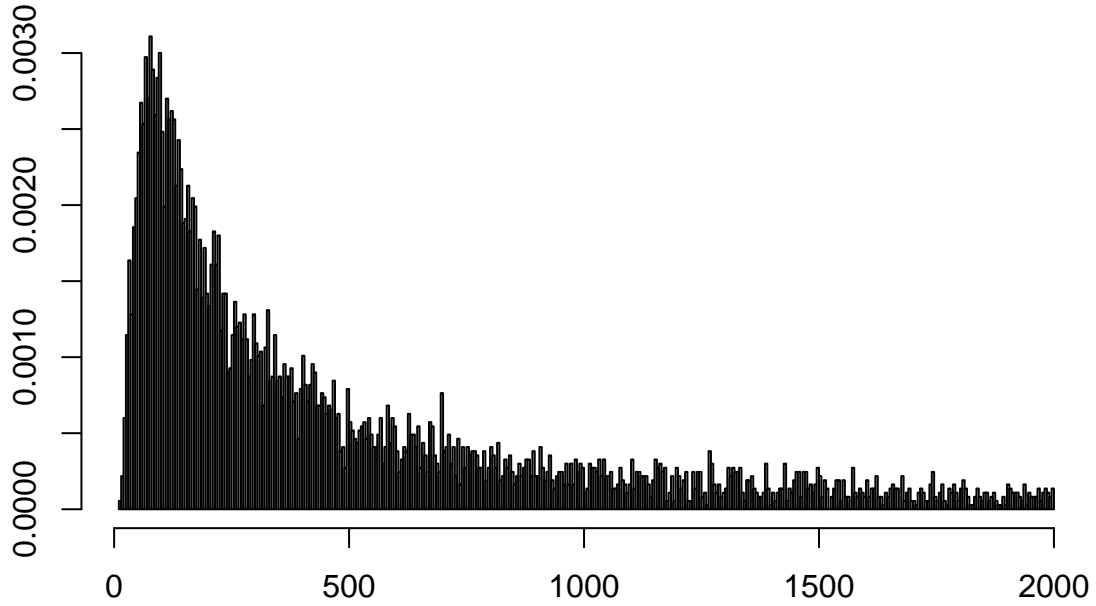
Accept/Reject samples when c is 2
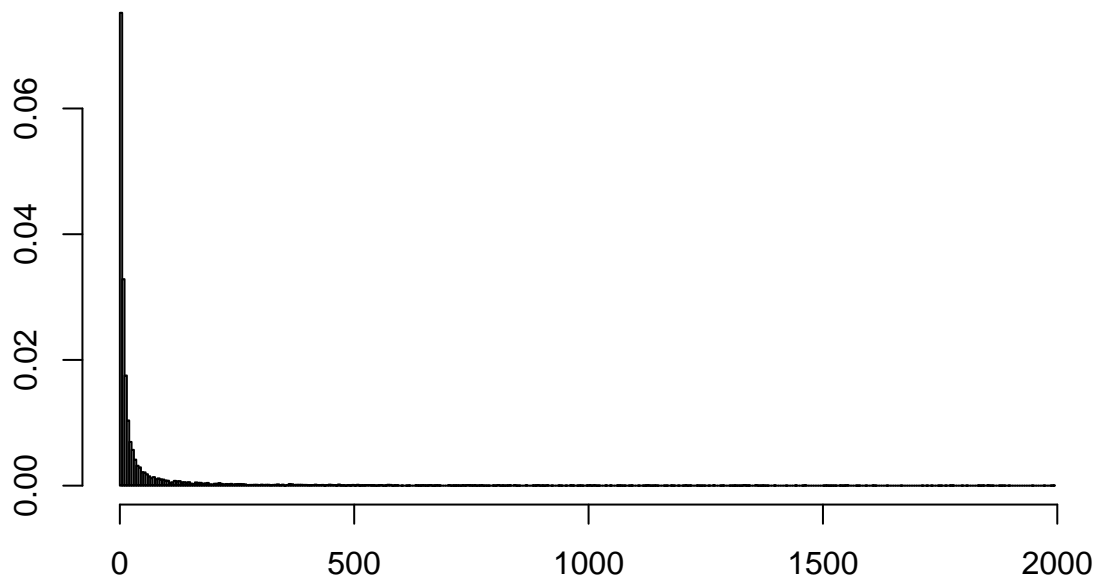
Accept/Reject samples when c is 9
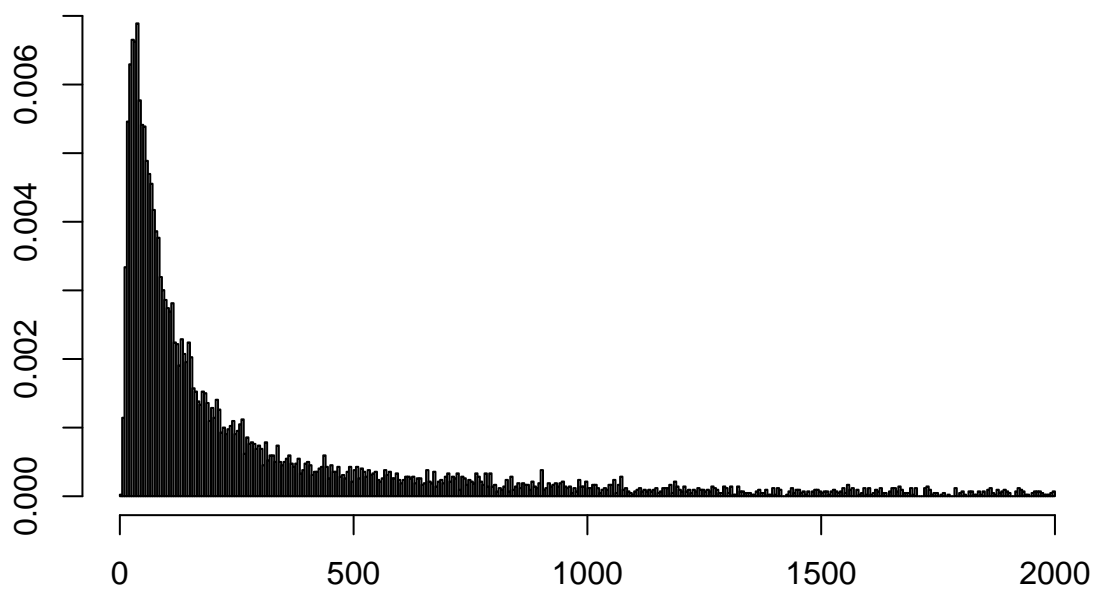
Accept/Reject samples when c is 15

Mean for c = 5 is 5.2121701×10^{5}

Mean for c = 10 is 7.4298507×10^{6}

Mean for c = 15 is 1.1634423×10^{9}

Variance for c = 5 is 2.5106451×10^{15}

Variance for c = 10 is 4.9667656×10^{17}

Variance for c = 15 is 1.3490093×10^{22}

Rejection rate for c = 5 is 0.3896

Rejection rate for c = 10 is 2.4093

Rejection rate for c = 15 is 3.6156

We can see that as value of c increases, the mean and variance also increases. This is because as c increases , target function peak moves further towards right on x as max of target function is achieved when x is c^2 / 3. As c increases , the max of target function decreases, hence reducing the sharp peaks and making it smoother. This inturn increases the values being sampled by accept reject algorithm, thus increasing mean and variance. Rejection rate also increases as the value of c increases.The proposal density goes way higher than our target density. This results in increase in number of rejections.

We have obtained values much larger than 2000 from the sampling, but to nothice the distribution in the beginning where the curve can be observed , we have filtered out the values below 20000.

Question 2: Laplace distribution

```
library(VGAM)
```

```
## Warning: package 'VGAM' was built under R version 4.0.3
```

```
set.seed(12345)
```

The Laplace distribution is the distribution of the difference of two independent random variables with identical exponential distributions.

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

By integrating f(x), CDF is given as :

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

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

$if(x > \mu)$

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

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

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

$if(x <= \mu)$

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

Inverse of CDF

For $if(x > \mu)$ we got

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

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

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

So for $DE(0, 1)$,

$$\frac{\ln(2 - 2y) - 0}{-1} = x$$

For $U \sim U(0, 1)$,

$$-\ln(2 - 2U) = X$$

$$-\ln 2(1 - U) = X$$

$$x \geqslant 0 => \ln 2(1 - U) \leqslant 0 => 0 < 2(1 - U) \leqslant 1 => 0 < (1 - U) \leqslant \frac{1}{2} => -1 < -U \leqslant -\frac{1}{2} => \frac{1}{2} \leqslant U < 1$$

For $if(x <= \mu)$ we got

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

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

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

So for $DE(0,1)$,

$$\frac{\ln(2y)}{1} + 0 = x$$

For $U \sim U(0,1)$,

$$\ln(2U) = X$$

$$x < 0 \Rightarrow 0 < 2U < 1 \Rightarrow 0 < U < \frac{1}{2}$$

Hence inverse CDF equations are

$$F^{-1}(U) = ln(2U)$$

for$0 < U < \frac{1}{2}$

$$F^{-1}(U) = -ln2(1-U)$$

for$\frac{1}{2} \leqslant U < 1$

Generate 10000 random numbers from distribution

```r
#U <- runif(1000,0,1)
inverse_cdf <- function(n){
  U = runif(n,0,1)                          # generate random numbers from the known distribution
  X = c()
  for(i in 1:length(U)){
    if(U[i] < 0.5){
      X[i] = log(2 * U[i])
    }
    else{
      X[i] = -log(2 * (1 - U[i]))
    }
  }
  return(X)
}
X = inverse_cdf(10000)
data = rlaplace(10000,location = 0,scale = 1)
df = data.frame(X,data)
```
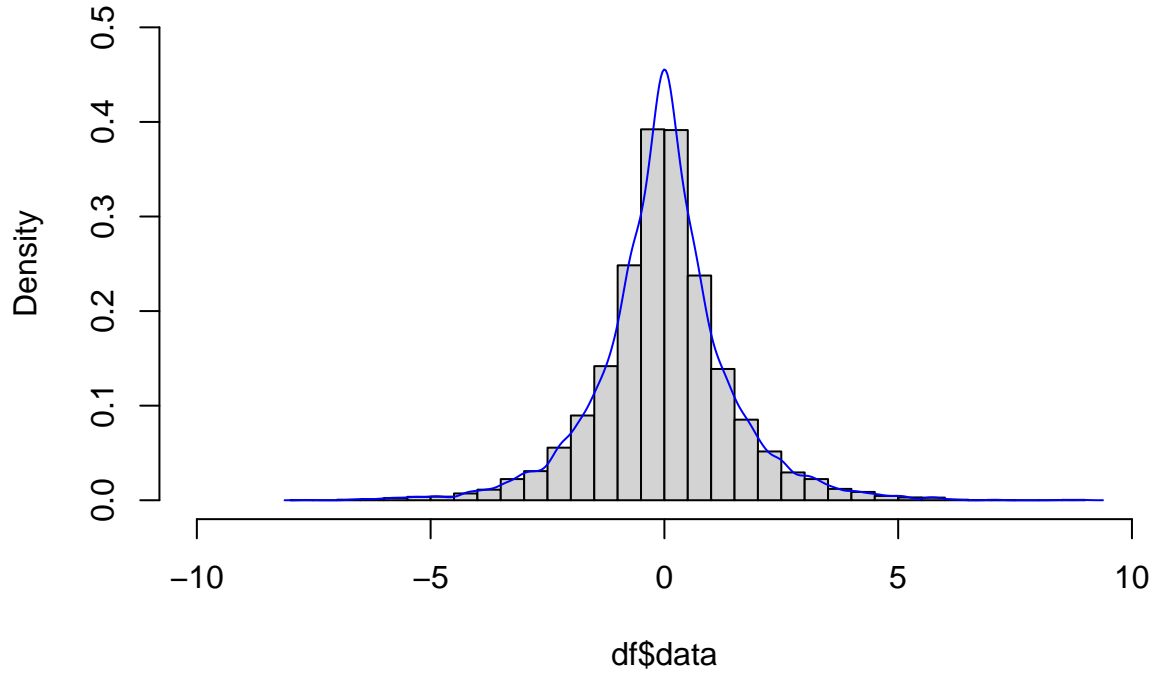
Plot the generated numbers

```r
plot1 = hist(df$X, prob = TRUE, ylim = c(0,0.5), xlim = c(-10,10),
breaks = 30, main = "Calculated Laplace Distribution")
lines(density(df$X), col = "blue")
```

Calculated Laplace Distribution
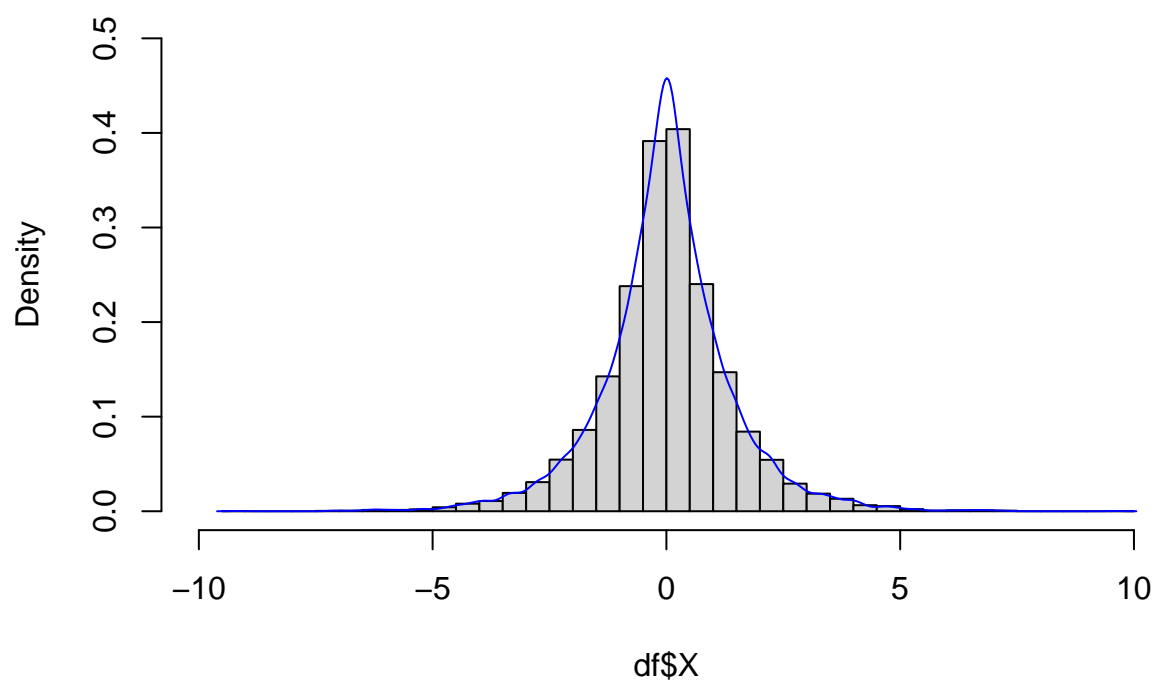

```
plot2 = hist(df$data, prob = TRUE, ylim = c(0,0.5), xlim = c(-10,10),
breaks = 30, main = "Built in Laplace Distribution")
lines(density(df$data), col = "blue")
```

Built in Laplace Distribution

The result looks reasonable. The two plots are similar. Due to mean $= 0$ all the points are close to 0 as shown in the plot. Conclusion, the results obtained using CDF function is close with the result using the rlaplace function.

#2.2 Use the Acceptance/rejection method with DE(0, 1) as a majorizing density to generate N (0, 1) variables.

$$f_X(x) \sim N(0,1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

$$f_Y(x) \sim DE(0,1) = \frac{1}{2} e^{-|x|}$$

If we maximize the f_X(x)/f_Y(x) then it would be the value of c. This can be done with partial derivative of the fraction.

$$\frac{f_X(x)}{f_Y(x)} \leq c$$

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

$$g(x) = -\frac{x^2}{2} + x$$

$$g'(x) = 1 - x$$

$$1 - x = 0 => x = 1$$

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

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

By calculation, constant c $= 1.3154$

Generation of 2000 random numbers from N(0,1)

```r
c = sqrt((2*exp(1))/pi)

DE <- function(x){
  exp(-abs(x)) / 2
}
accept_reject <- function(n){
  R <- 0
  Y <- vector(length = n)
  for(i in 1:n){
    repeat {
      y <- inverse_cdf(1)
      U <- runif(1)
      fy <- dnorm(y, mean = 0, sd = 1)
      gy <- DE(y)
      if(U <= fy / (c * gy)){
        Y[i] <- y
        break
        }
      else{R = R+1}
    }
  }
  return(list(Y=Y, Reject=R))
}
set.seed(12345)
Z <- accept_reject(n = 2000)
```
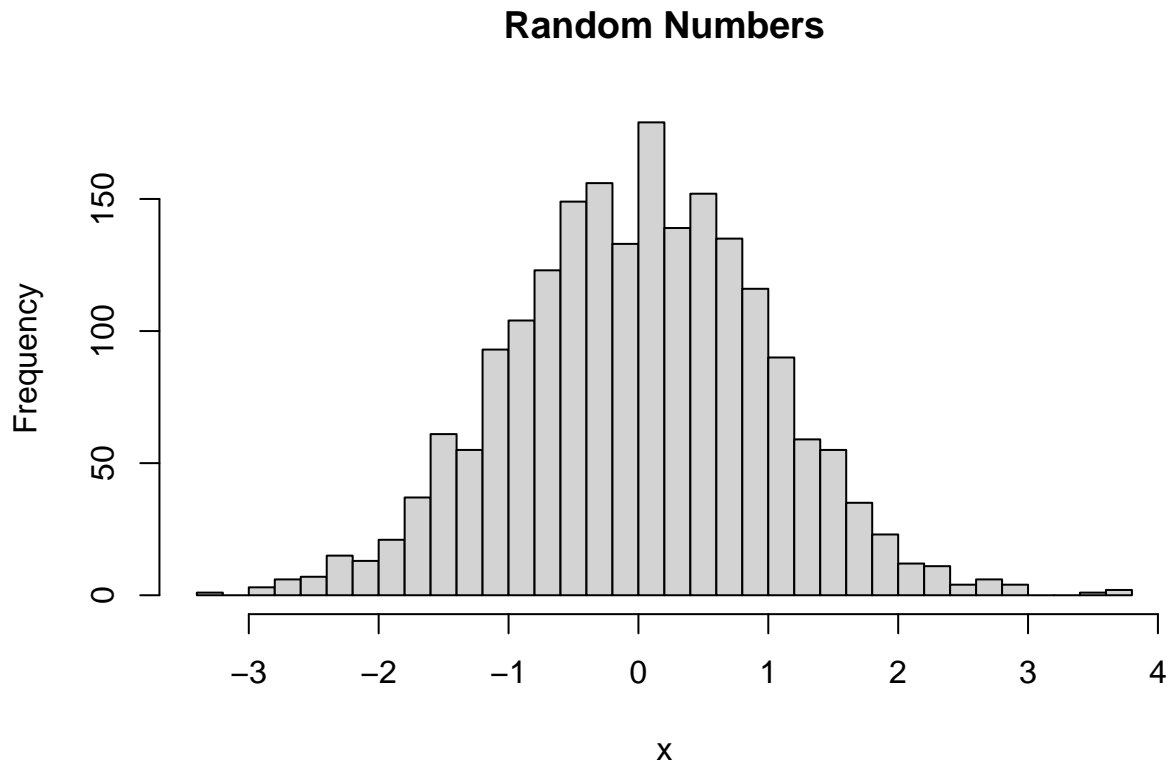
Plot the histogram

```r
hist(Z$Y, xlab = "x",main="Random Numbers", breaks = 50)
```

Random Numbers


```r
cat("Expected Rejection Rate: ER = ", 1 - (1/c))
```

```
## Expected Rejection Rate: ER =  0.2398265
```

```r
cat("Rejection Rate: R = ", sum(Z$Reject)/(2000 +sum(Z$Reject)))
```

```
## Rejection Rate: R =  0.2392545
```

The expected rejection rate is 0.2398265 and the average rejection rate generated in accept_reject function() is 0.2392545. The difference is only 0.000572. Hence expected rejection rate is very close to R. This means that our function for the random sample has satisfying results.

Generate 2000 numbers from N (0, 1) using standard rnorm() procedure and plot the histogram

```r
set.seed(12345)
hist(rnorm(2000,0,1), main = "Normal Random Numbers from rnorm()", xlab = "x", breaks = 50 )
```

Normal Random Numbers from rnorm()
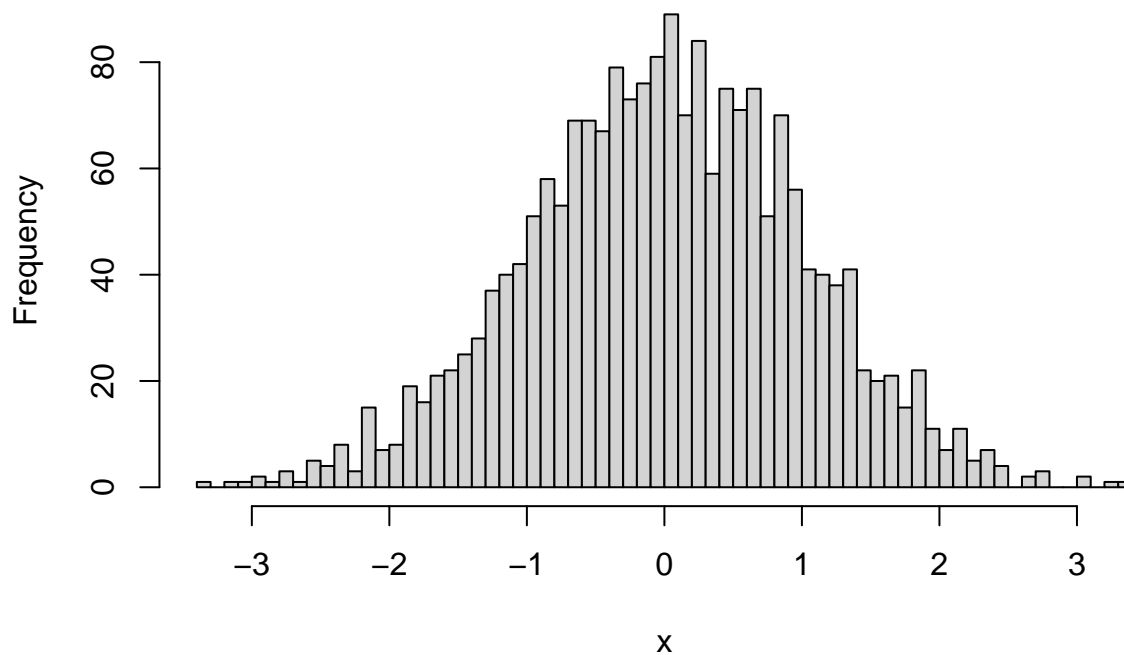
Comparing 2 histograms we can conclude that the 2 results looks similar hence the function is efficient.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(poweRlaw)
f = function(x,c){
  res = 0
  res = c * ((sqrt(2*pi))^-1) * exp(-(c^2)/(2*x)) * x^(-3/2)
  res[x<=0]<-0
  return(res)
}
fp = function(x, a, tmin) {
  return(((a - 1) / tmin) * (x / tmin)^(-a))
  }
c = 2
tmin =  1.33
a = 1.3
x = seq(0,50,length.out = 100)
plot(x, f(x,c), type = "l", ylim = c(0,0.5),xlab = "", ylab = "")
lines(x,fp(x,a,tmin),col="red")
title("Powerlaw and target function")
legend(30,0.5,legend = c("Powerlaw","Target function"),
       lty = 1, col = c("red","black"))
```

```r
p = 0.0832 #for c = 2
max = max(f(x,c))
nd = function(x,tmin,a,p){
  res = c()
  for(i in 1:length(x)){
    if(x[i]>=0 && x[i]<= tmin){
      res[i] = p * dunif(x[i],0,tmin)
    }
    if(x[i]>tmin){
      res[i] = (1-p) * fp(x[i],a,tmin)
    }
  }
  return(res)
}

cmaj = function(c,a) {
  xmaj <- (c^2)/(3-2*a)
  return((f(xmaj,c)/(nd(xmaj,tmin,a,p))))
}
c_maj = cmaj(c,a)

plot(f(x,c),type = "l", col = "blue",ylim = c(0,0.5))
lines(nd(x,tmin,a,p)*c_maj, col = "red")

rmajorizing<-function(n,tmin,a){
  sapply(1:n,function(i){
    res<-NA
    component<-sample(1:2,1,prob = c(0.0833,1-0.0833))
    if(component==1){res<-runif(1,0,tmin)}
    if(component==2){res<-rplcon(1,tmin,a)}
    res
  })
}
Nsample<-20000
num_histbreaks<-1000
hv = rmajorizing(Nsample,tmin,a)
hist(hv[hv<2000],breaks=300,col="black",xlab="",ylab="",
     main="HistMajorizing",freq=FALSE)


accept_reject<-function(c,c_maj,t){
  x<-NA
  num_reject<-0
  while (is.na(x)){
    y<-rmajorizing(1,t,a)
    u<-runif(1)
    if (u<=f(y,c)/(c_maj*nd(y,a,tmin = t,p))){
      x<-y
    } else{
      num_reject<-num_reject+1
    }
  }
```

```r
 res = c(x,num_reject)
 return(res)
}

set.seed(12345)
Nsample = 10000
c = c(2,9,15)
p = 0.0833
t = c(4/3,81/3,15/3)
cm = sapply(c,cmaj,a)
mean=var=rejection_rate=c()
for(i in 1:length(c)){
tmin = t[i]
fx_acceptreject<-sapply(rep(c[i],Nsample),accept_reject,c_maj = cm[i], t = tmin)
histvalue = fx_acceptreject[1,]
histvalue = histvalue[histvalue<2000]
hist(histvalue,col="white",border = "black",breaks=300,xlab="",
     ylab="",freq=FALSE,main="")
title(capture.output(cat("Accept/Reject samples when c is ", c[i])))
mean =cbind(mean, mean(fx_acceptreject[1,]))
var = cbind(var,var(fx_acceptreject[1,]))
rejection_rate = cbind(rejection_rate,sum(fx_acceptreject[2,])/Nsample)
}
library(VGAM)
set.seed(12345)
#U <- runif(1000,0,1)
inverse_cdf <- function(n){
  U = runif(n,0,1)                      # generate random numbers from the known distribution
  X = c()
  for(i in 1:length(U)){
    if(U[i] < 0.5){
      X[i] = log(2 * U[i])
    }
    else{
      X[i] = -log(2 * (1 - U[i]))
    }
  }
  return(X)
}
X = inverse_cdf(10000)
data = rlaplace(10000,location = 0,scale = 1)
df = data.frame(X,data)
plot1 = hist(df$X, prob = TRUE, ylim = c(0,0.5), xlim = c(-10,10),
breaks = 30, main = "Calculated Laplace Distribution")
lines(density(df$X), col = "blue")
plot2 = hist(df$data, prob = TRUE, ylim = c(0,0.5), xlim = c(-10,10),
breaks = 30, main = "Built in Laplace Distribution")
lines(density(df$data), col = "blue")
c = sqrt((2*exp(1))/pi)

DE <- function(x){
  exp(-abs(x)) / 2
}
```

```r
accept_reject <- function(n){
  R <- 0
  Y <- vector(length = n)
  for(i in 1:n){
    repeat {
      y <- inverse_cdf(1)
      U <- runif(1)
      fy <- dnorm(y, mean = 0, sd = 1)
      gy <- DE(y)
      if(U <= fy / (c * gy)){
        Y[i] <- y
        break
        }
      else{R = R+1}
    }
  }
  return(list(Y=Y, Reject=R))
}
set.seed(12345)
Z <- accept_reject(n = 2000)
hist(Z$Y, xlab = "x",main="Random Numbers", breaks = 50)
cat("Expected Rejection Rate: ER = ", 1 - (1/c))
cat("Rejection Rate: R = ", sum(Z$Reject)/(2000 +sum(Z$Reject)))
set.seed(12345)
hist(rnorm(2000,0,1), main = "Normal Random Numbers from rnorm()", xlab = "x", breaks = 50 )
```