Computer Lab 3

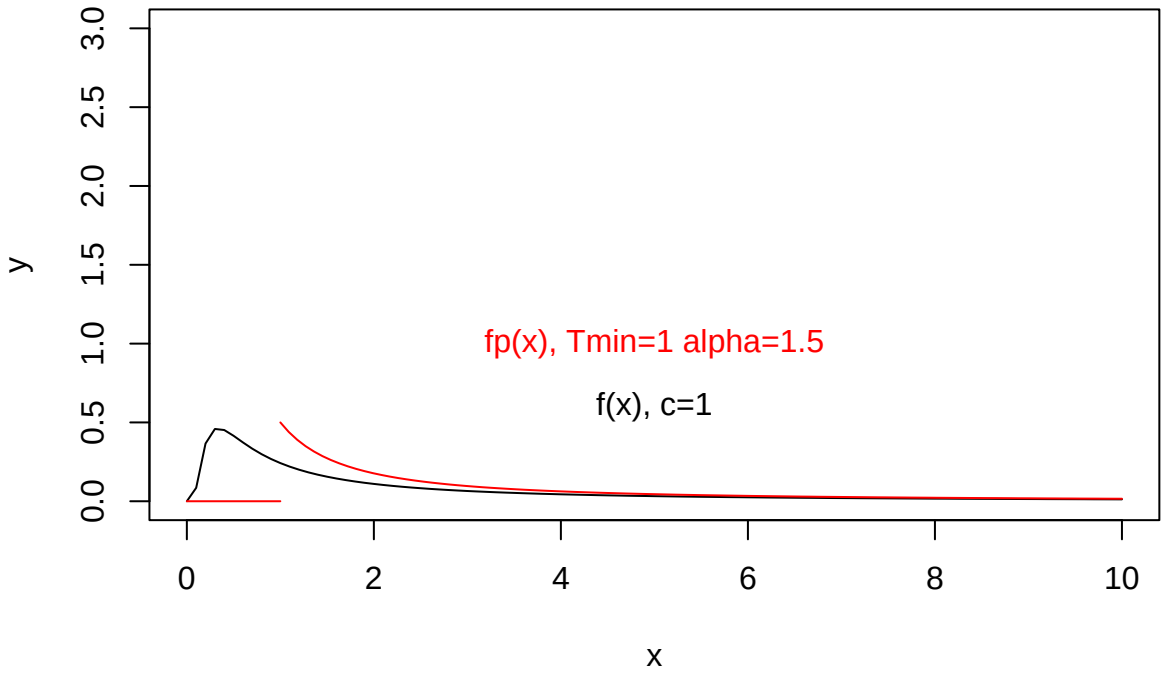
Shipeng Liu (shili506) Dongwei Ni (donni508)

Question 1: Stable distribution

Task 1

```
fx <- function(x, c){
  if (x > 0) {
    f <- c * (sqrt(2*pi)^-1) * exp(1)^(-c^2 / (2*x)) * x^(-3 / 2)
  } else {
    f <- 0
  }
  return(f)
}

fpx <- function(x, Tmin, alpha){
  if (x > Tmin) {
    fp <- ((alpha - 1) / Tmin) * (x / Tmin)^(-alpha)
  } else {
    fp <- 0
  }
  return(fp)
}
```

```
fx_c <- 1
fpx_Tmin <- 1
fpx_alpha <- 1.5

fx1 <- \(x) sapply(x, fx, c = fx_c)
# f(x) in a sapply form
fpx1 <- \(x) sapply(x, fpx, Tmin = fpx_Tmin, alpha = fpx_alpha)
# fp(x) in a sapply form
curve(fx1, xlim = c(0,10), ylim = c(0,3), xlab = "x", ylab = "y")
curve(fpx1, from = 0, to = fpx_Tmin , add = TRUE, col = "red")
curve(fpx1, from = fpx_Tmin+ exp(-15) ,  add = TRUE, col = "red")
text(x = 5, y  = c(0.6, 1),
     # labels = c("f(x), c=1", "fp(x), Tmin=1, alpha=2"),
     labels = c(paste0("f(x), c=", fx_c), paste0("fp(x), Tmin=",fpx_Tmin, " alpha=", fpx_alpha)),
     col = c("black", "red")
     )
```



The power–law distribution can not be used just by itself,because the two distribution don't have the same support. In $(0, t_{min})$,the density of the power–law distribution is 0,but for one–sided strictly stable distribution of order 1/2 it isn't.The power–law distribution can't generate the proper samples in accept-reject method in this interval. We can apply a uniform distribution at least on the interval $(0, t_{min})$ as a majorizing function.

if we want $\forall_x n f_p(x) \geq f(x)$, then it has to be

$$\lim_{x\to\infty} \frac{n f_p(x)}{f(x)} \geq 1$$

to guarantee this requirement ,we can make that

$$\lim_{x\to\infty} \frac{x^{-\alpha}}{x^{-3/2}} = +\infty$$
$$\alpha < \frac{3}{2}$$

given that we choose $\alpha = 1.2$
take the derivative of target density function

$$\frac{df(x)}{dx} = (\sqrt{2\pi}^{-1} e^{-\frac{c^2}{2x}} (\frac{c^2}{2x} x^{-3/2} - \frac{3}{2} x^{-5/2}))$$

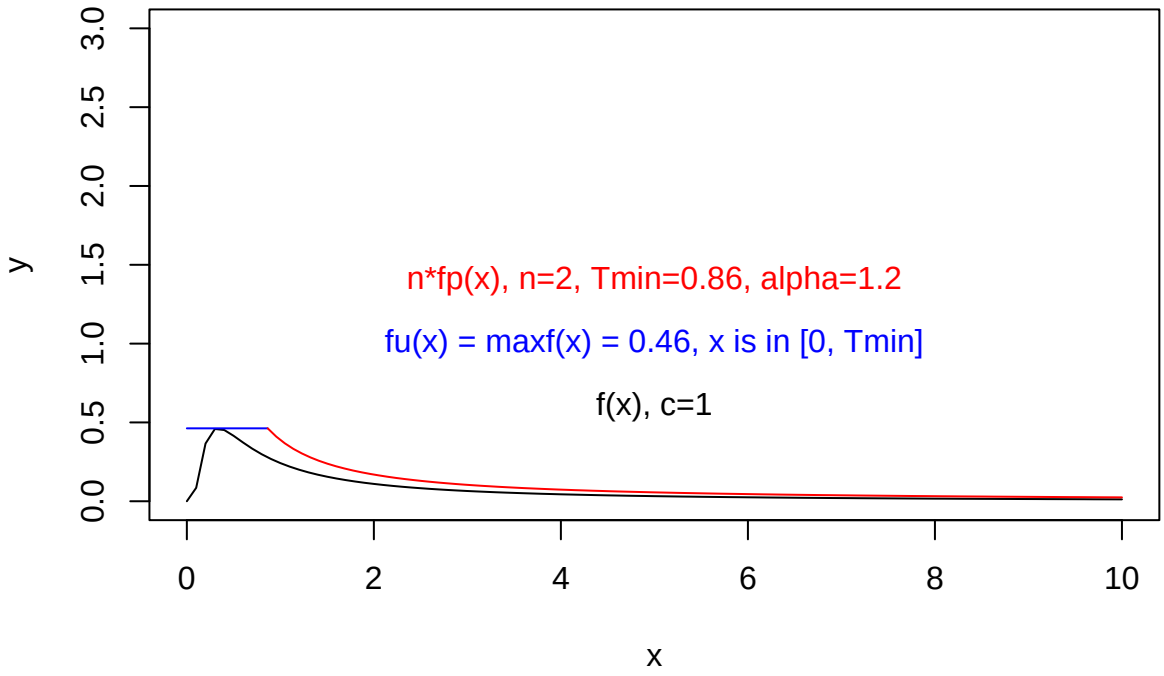
solve

$$\frac{df(x)}{dx} = 0$$
$$x = \frac{1}{3} c^2$$

now we know that $maxf(x) = f(\frac{1}{3}c^2)$ we want n rises with c but with a negative 2nd-derivative, so arbitrarily we choose $n = 1.5 + 0.5\sqrt{c}$ then we let

$$\lim_{x \to T_{min}} nf_p(x) = n\frac{\alpha - 1}{T_{min}} = f(\frac{1}{3}c^2)$$
$$T_{min} = n\frac{\alpha - 1}{f(\frac{1}{3}c^2)}$$

```
fx_c <- 1
fpx_n <- 1.5+0.5*sqrt(fx_c)
fpx_alpha <- 1.2
fx1 <- \(x) sapply(x, fx, c = fx_c)
# f(x) in a sapply form
fx1_max <- fx1(fx_c^2/3)

fpx_Tmin <- fpx_n*(fpx_alpha - 1) / fx1_max

fpxn <- function(x, Tmin, alpha, n){
  if (x > Tmin) {
    fp <- n *((alpha - 1) / Tmin) * (x / Tmin)^(-alpha)
    #n * fpx
  } else {
    fp <- 0
  }
  return(fp)
}
fpx2 <- \(x) sapply(x, fpxn, Tmin = fpx_Tmin, alpha = fpx_alpha, n = fpx_n)
# n*fp(x) in a sapply form

curve(fx1, xlim = c(0,10), ylim = c(0,3), xlab = "x", ylab = "y")
curve(fx1_max*x^0, from = 0, to = fpx_Tmin , add = TRUE, col = "blue")
curve(fpx2, from = fpx_Tmin+ exp(-15) ,  add = TRUE, col = "red")
text(x = 5, y  = c(0.6, 1, 1.4),
     # labels = c("f(x), c=1", "fp(x), Tmin=1, alpha=2"),
     labels = c(paste0("f(x), c=", fx_c),
                paste0("fu(x) = maxf(x) = ", round(fx1_max, 2),", x is in [0, Tmin]" ),
                paste0("n*fp(x)", ", n=", fpx_n, ", Tmin=", round(fpx_Tmin,2), ", alpha=", fpx_alpha)),
     col = c("black", "blue", "red")
     )
```



Task 2

```
int1 <- fx1_max * fpx_Tmin
int1
```

```
## [1] 0.4
```

```
int2 <- integrate(fpx2, fpx_Tmin, Inf)
int2$value
```

```
## [1] 2
```

we have $\int_0^{T_{min}} f_{unif}(x) : \int_{T_{min}}^{+\infty} f_p(x) = 1 : 5$

```
rmajor <- function(n){
  res <- c()
  for (i in 1:n) {
    div <- sample(c(1,2), 1, prob = c(int1, int2$value))
    if (div == 1) {
      res[i] <- runif(1,0, fpx_Tmin)
    } else {
      res[i] <- rplcon(1,fpx_Tmin, fpx_alpha)
    }
  }
  return(res)
}

fmajor <- function(x){
```

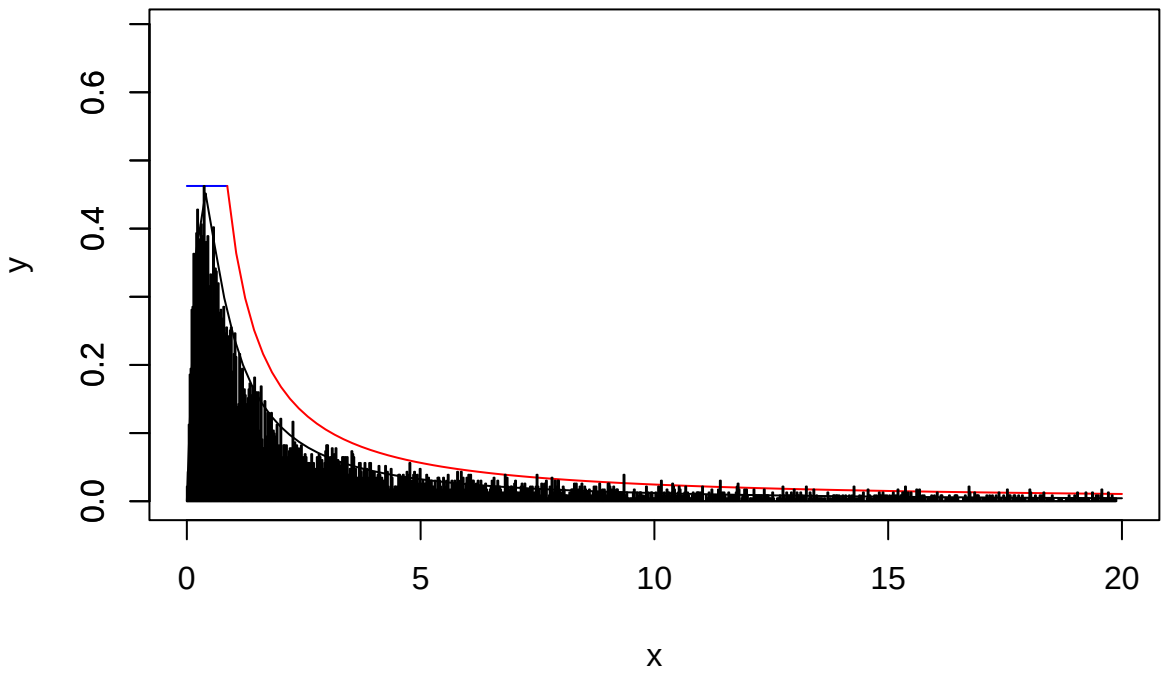
```
  if (x <= 0) return(0)
  sapply(x, function(x, Tmin){
    if (x > Tmin) {
      y <- fpx2(x)
    }else {
      y <- fx1_max
    }
    return(y)
  },
  Tmin = fpx_Tmin
    )
}

generate <- function(n){
    x <- c()
    reject <- 0
  for (i in 1:n) {
    x[i] <- NA
    while (is.na(x[i])) {
      y <- rmajor(1)
      u <- runif(1,0, 1)
      accept <- u <= (fx1(y) / fmajor(y))
      if (accept) {
        x[i] <- y
      } else {
        reject <- reject + 1
      }
    }
  }
  return(list(x, reject))
}
```

Task 3

make a sample which n == 10000

for c = 1:

```
generate_list <- generate(10000)
rnumber <- generate_list[[1]]
reject_num <- generate_list[[2]]


hist_rnum <- hist(rnumber[rnumber<20],breaks = 1000,plot = FALSE)
hist_scale <- hist_rnum$counts * (fx1_max / max(hist_rnum$counts))

curve(fx1, xlim = c(0,20), ylim = c(0,1.5*fx1_max), xlab = "x", ylab = "y")
curve(fx1_max*x^0, from = 0, to = fpx_Tmin , add = TRUE, col = "blue")
curve(fpx2, from = fpx_Tmin+ exp(-15) ,  add = TRUE, col = "red")
barplot(hist_scale, width = 20 / 1000, add = TRUE, space = 0)
```




```
mean(rnumber)
```

```
## [1] 38595.98
```

```
var(rnumber)
```

```
## [1] 5.790519e+12
```

```
reject_num / (reject_num + length(rnumber))
```

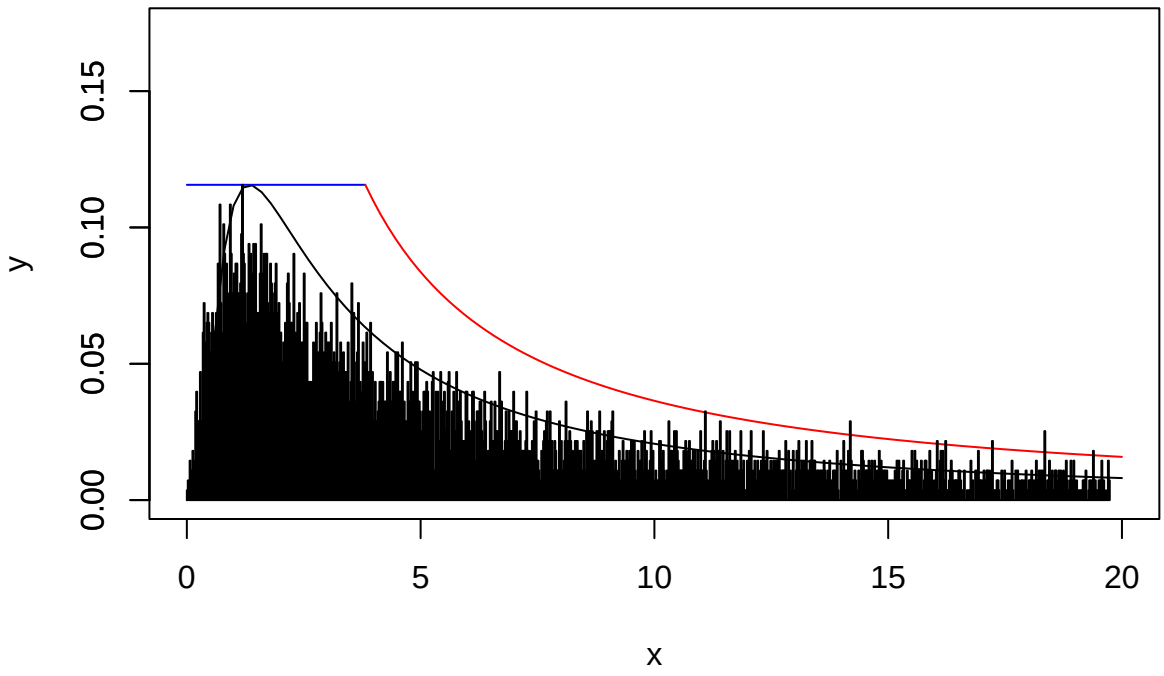
```
## [1] 0.5827074
```

```
# reject rate
```

for c = 2:

```
fx_c <- 2
fpx_n <- 1.5+0.5*sqrt(fx_c)
fpx_alpha <- 1.2
fx1 <- \(x) sapply(x, fx, c = fx_c)
fx1_max <- fx1(fx_c^2/3)
fpx_Tmin <- fpx_n*(fpx_alpha - 1) / fx1_max

fpxn <- function(x, Tmin, alpha, n){
  if (x > Tmin) {
    fp <- n *((alpha - 1) / Tmin) * (x / Tmin)^(-alpha)
  } else {
    fp <- 0
  }
```

```
  return(fp)
}
fpx2 <- \(x) sapply(x, fpxn, Tmin = fpx_Tmin, alpha = fpx_alpha, n = fpx_n)

generate_list <- generate(10000)
rnumber <- generate_list[[1]]
reject_num <- generate_list[[2]]

hist_rnum <- hist(rnumber[rnumber<20],breaks = 1000,plot = FALSE)
hist_scale <- hist_rnum$counts * (fx1_max / max(hist_rnum$counts))

curve(fx1, xlim = c(0,20), ylim = c(0,1.5*fx1_max), xlab = "x", ylab = "y")
curve(fx1_max*x^0, from = 0, to = fpx_Tmin , add = TRUE, col = "blue")
curve(fpx2, from = fpx_Tmin+ exp(-15) ,  add = TRUE, col = "red")
barplot(hist_scale, width = 20 / 1000, add = TRUE, space = 0)
```




```
mean(rnumber)
```

```
## [1] 3936051
```

```
var(rnumber)
```

```
## [1] 1.535814e+17
```

```
reject_num / (reject_num + length(rnumber))
```

```
## [1] 0.6227127
```

```
# reject rate
```

both the mean and the variance falls when c rises, the proposal function we chose yields more rejection rate than what we can see from the plot when x nears 0. So it might be more important to consider optimal the proposal function when $x \to +\infty$ than when x nears 0. $\alpha = 1.2$ don't seems to be a best choice. Seems it's better to consider a greater α, such as 1.5.

Question 2: Laplace distribution

Task 1

Write a code generating double exponential distribution DE(0, 1) from Unif(0, 1) by using the inverse CDF method.

The PDF of the Laplace distribution is given:

$$f(x) = \frac{\alpha}{2} exp(-\alpha \mid x - \mu \mid)$$

And we calculate the CDF:

$$F(x) = \int_{-\infty}^{x} f(u)du = \begin{cases} \frac{1}{2}\mathrm{e}^{(\frac{\mathrm{x}-\mu}{\alpha})} & \text{if } \mathrm{x} < \mu \\ 1 - \frac{1}{2}\mathrm{e}^{(\frac{\mathrm{x}-\mu}{\alpha})} & \text{if } \mathrm{x} \geq \mu \end{cases}$$

The inverse of CDF:

$$F^{-1}(x) = \begin{cases} -\frac{1}{\alpha}\ln 2(1 - F(x)) + \mu & \text{if } \mathrm{x} \geq \mu \\ \frac{ln(2F(x))}{\alpha} + \mu & \text{if } \mathrm{x} < \mu \end{cases}$$

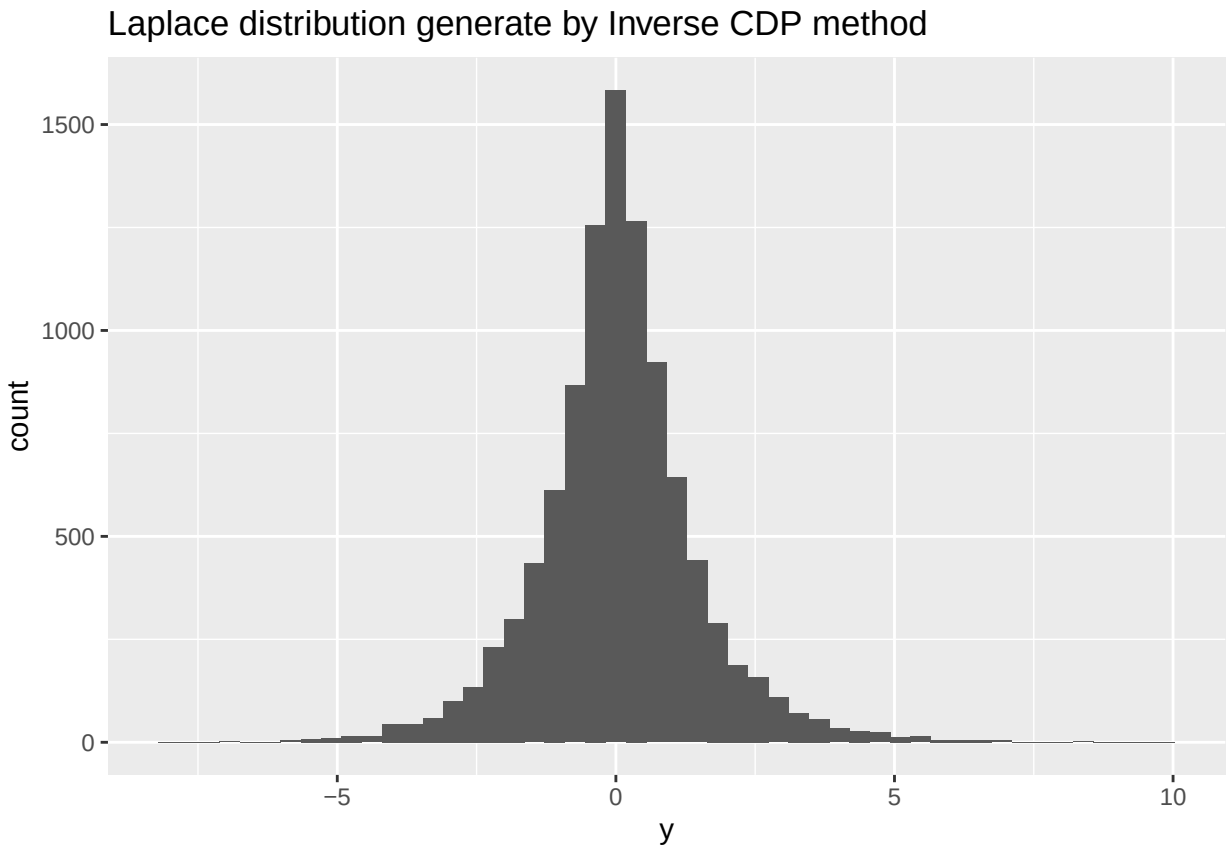
when $x = \mu$,$F(x) = \frac{1}{2}$ hence The inverse of CDF:

$$F^{-1}(x) = \begin{cases} -\frac{1}{\alpha}\ln 2(1 - F(x)) + \mu & \text{if } \mathrm{F(x)} \geq \frac{1}{2} \\ \frac{ln(2F(x))}{\alpha} + \mu & \text{if } \mathrm{F(x)} < \frac{1}{2} \end{cases}$$

```
r_laplace=function(gen_num,mu,alpha){
  unif=runif(gen_num,0,1)
  temp=sapply(unif,function(unif){
    if(unif>=0.5){
      return((-1/alpha)*log(2*(1-unif))+mu)
    }else{
      return((log(2*unif)/alpha)+mu)
    }
  })
  return(temp)
}
```

Generate 10000 random numbers from this distribution, plot the histogram and comment whether the result looks reasonable.

```
p4=ggplot(data=data.frame("y"=r_laplace(10000,0,1)),aes(x=y))+
  geom_histogram(bins=50)+
  labs(title="Laplace distribution generate by Inverse CDP method")
p4
```



Compare to the plot of Laplace distribution probability density function(it can be thought of as two exponential distributions spliced together along the abscissa),the result seems resonable.

Task 2

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

```
ar_norm<-function(c){
  x<-NA
  num.reject<-0
  while (is.na(x)){
    y<-r_laplace(1,0,1)
    u<-runif(1)
    if (u<=dnorm(y,0,1)/(c*(exp(-1*abs(y))/2))){x<-y}
    else{num.reject<-num.reject+1}
  }
  c(x,num.reject)
}
```

We sample Y from the majorizing distribution, sample U from the uniform distribution, and then filter out samples that satisfy $U \leq \frac{f_X(Y)}{cf_Y(Y)}$, and resample if not satisfied , until a sample that satisfies the condition is obtained.

majorizing constant c

We need to choose majorizing constant c such that:

$$\forall_x c f_Y(x) \geq f_X(x)$$

where $f_Y(x)$ is PDF of Laplace distribution,$f_X(x)$ is PDF of Normal distribution. c mush be large enough,but too large c may cause large rejection rates,we need to choose carefully.

$$\frac{c}{2}e^{-|x|} \geq \frac{1}{\sqrt{2\pi}}e^{|x|-\frac{x^2}{2}}$$

solve it and we'll get:

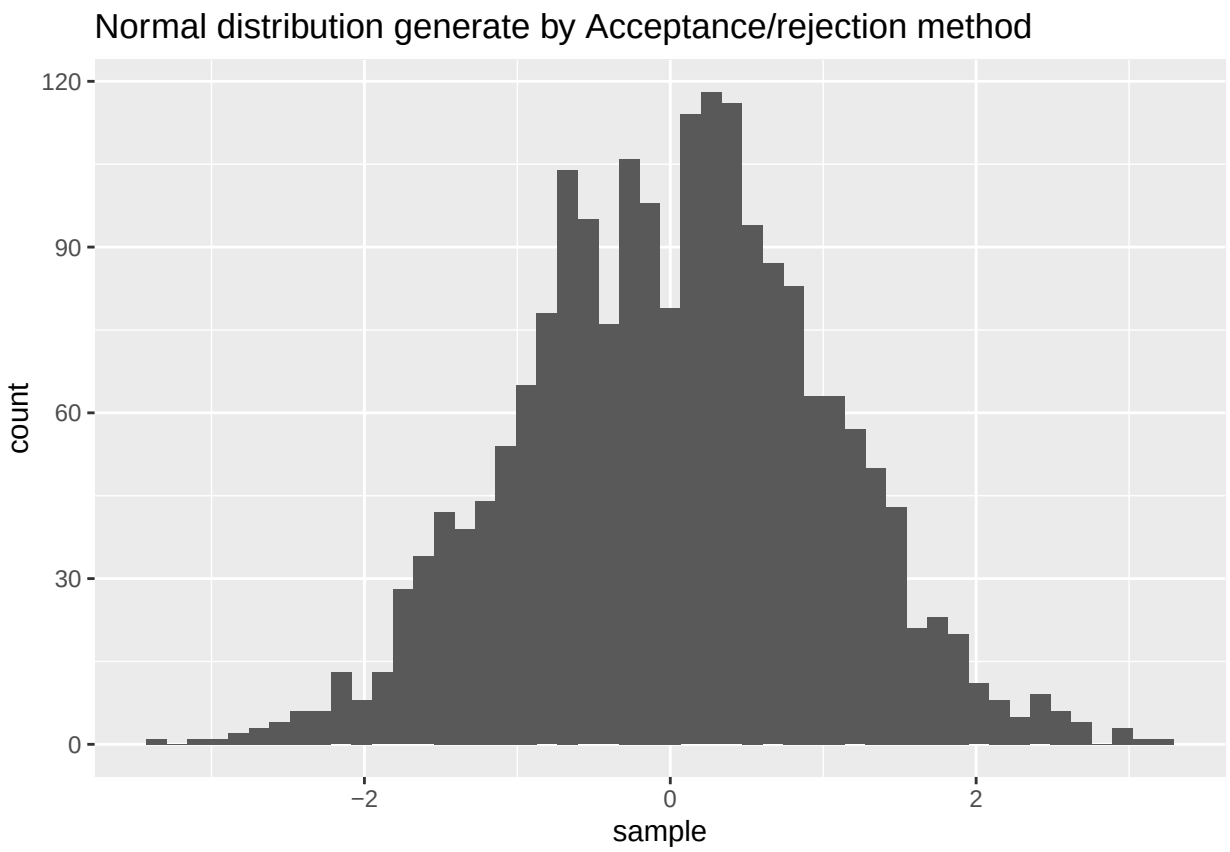
$$c \geq \sqrt{\frac{2}{\pi}}e^{(|x|-\frac{x^2}{2})}$$

when$x = 1$,we get the maximum c:

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

Generate 2000 random numbers N (0, 1) using your code and plot the histogram.

```
df_norm=data.frame(t(data.frame(sapply(rep(sqrt((2*exp(1))/pi),2000),ar_norm))))
colnames(df_norm)=c("sample","reject")
p4=ggplot(data=df_norm,aes(x=sample))+
  geom_histogram(bins=50)+
  labs(title="Normal distribution generate by Acceptance/rejection method")
p4
```



Rejection rate

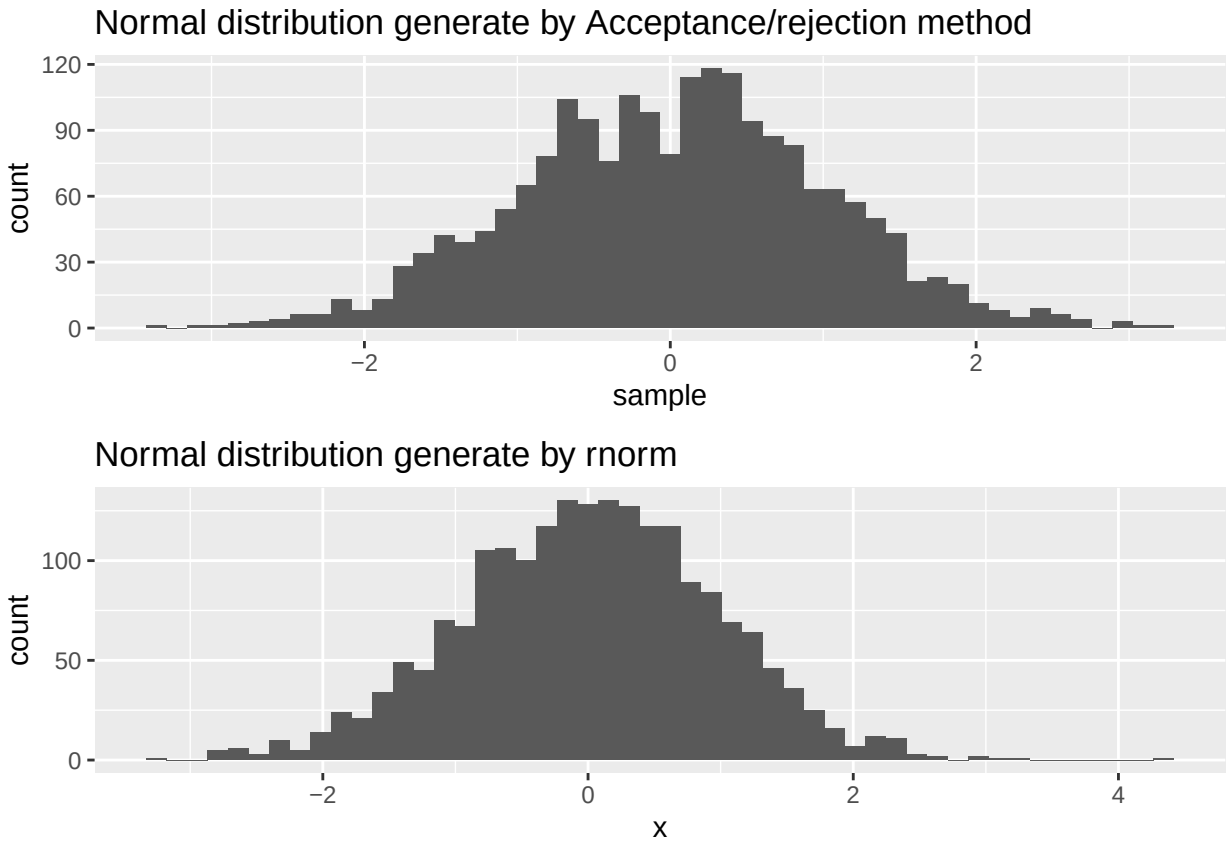
```
mean_r=sum(df_norm[,2])/(2000+sum(df_norm[,2]))
ER=1-(1/sqrt((2*exp(1))/pi))

cat("The average rejection rate R: ",mean_r,"\nThe expected rejection rate ER: ",
    ER,"\nThe difference:",abs(ER-mean_r))
```

```
## The average rejection rate R:  0.2406986
## The expected rejection rate ER:  0.2398265
## The difference: 0.0008720079
```

The R and ER are very close to each other.

Generate 2000 numbers from N (0, 1) using standard rnorm() procedure

```
p5=ggplot(data=data.frame("x"=rnorm(2000,0,1)),aes(x=x))+
  geom_histogram(bins=50)+
  labs(title="Normal distribution generate by rnorm")
plot_comb1=grid.arrange(p4,p5)
```



The obtained two histograms has similar distribution.The number of samples concentrated around 0 is the largest, and the number of samples decreases as the absolute value of the sample increases.

Code appendix

```
library(dplyr)
library(ggplot2)
library(gridExtra)
library(poweRlaw)
fx <- function(x, c){
  if (x > 0) {
    f <- c * (sqrt(2*pi)^-1) * exp(1)^(-c^2 / (2*x)) * x^(-3 / 2)
  } else {
    f <- 0
  }
  return(f)
}

fpx <- function(x, Tmin, alpha){
  if (x > Tmin) {
    fp <- ((alpha - 1) / Tmin) * (x / Tmin)^(-alpha)
  } else {
    fp <- 0
  }
  return(fp)
}

fx_c <- 1
fpx_Tmin <- 1
fpx_alpha <- 1.5

fx1 <- \(x) sapply(x, fx, c = fx_c)
# f(x) in a sapply form
fpx1 <- \(x) sapply(x, fpx, Tmin = fpx_Tmin, alpha = fpx_alpha)
# fp(x) in a sapply form
curve(fx1, xlim = c(0,10), ylim = c(0,3), xlab = "x", ylab = "y")
curve(fpx1, from = 0, to = fpx_Tmin , add = TRUE, col = "red")
curve(fpx1, from = fpx_Tmin+ exp(-15) ,  add = TRUE, col = "red")
text(x = 5, y  = c(0.6, 1),
     # labels = c("f(x), c=1", "fp(x), Tmin=1, alpha=2"),
     labels = c(paste0("f(x), c=", fx_c), paste0("fp(x), Tmin=",fpx_Tmin, " alpha=", fpx_alpha)),
     col = c("black", "red")
     )


fx_c <- 1
fpx_n <- 1.5+0.5*sqrt(fx_c)
fpx_alpha <- 1.2
fx1 <- \(x) sapply(x, fx, c = fx_c)
# f(x) in a sapply form
fx1_max <- fx1(fx_c^2/3)

fpx_Tmin <- fpx_n*(fpx_alpha - 1) / fx1_max

fpxn <- function(x, Tmin, alpha, n){
  if (x > Tmin) {
```

```
    fp <- n *((alpha - 1) / Tmin) * (x / Tmin)^(-alpha)
    #n * fpx
  } else {
    fp <- 0
  }
  return(fp)
}
fpx2 <- \(x) sapply(x, fpxn, Tmin = fpx_Tmin, alpha = fpx_alpha, n = fpx_n)
# n*fp(x) in a sapply form

curve(fx1, xlim = c(0,10), ylim = c(0,3), xlab = "x", ylab = "y")
curve(fx1_max*x^0, from = 0, to = fpx_Tmin , add = TRUE, col = "blue")
curve(fpx2, from = fpx_Tmin+ exp(-15) ,  add = TRUE, col = "red")
text(x = 5, y  = c(0.6, 1, 1.4),
     # labels = c("f(x), c=1", "fp(x), Tmin=1, alpha=2"),
     labels = c(paste0("f(x), c=", fx_c),
                paste0("fu(x) = maxf(x) = ", round(fx1_max, 2),", x is in [0, Tmin]" ),
                paste0("n*fp(x)", ", n=", fpx_n, ", Tmin=", round(fpx_Tmin,2), ", alpha=", fpx_alpha)),
     col = c("black", "blue", "red")
     )


int1 <- fx1_max * fpx_Tmin
int1
int2 <- integrate(fpx2, fpx_Tmin, Inf)
int2$value
rmajor <- function(n){
  res <- c()
  for (i in 1:n) {
    div <- sample(c(1,2), 1, prob = c(int1, int2$value))
    if (div == 1) {
      res[i] <- runif(1,0, fpx_Tmin)
    } else {
      res[i] <- rplcon(1,fpx_Tmin, fpx_alpha)
    }
  }
  return(res)
}

fmajor <- function(x){
  if (x <= 0) return(0)
  sapply(x, function(x, Tmin){
    if (x > Tmin) {
      y <- fpx2(x)
    }else {
      y <- fx1_max
    }
    return(y)
  },
  Tmin = fpx_Tmin
    )
}
```

```r
generate <- function(n){
    x <- c()
    reject <- 0
  for (i in 1:n) {
    x[i] <- NA
    while (is.na(x[i])) {
      y <- rmajor(1)
      u <- runif(1,0, 1)
      accept <- u <= (fx1(y) / fmajor(y))
      if (accept) {
        x[i] <- y
      } else {
        reject <- reject + 1
      }
    }
  }
  return(list(x, reject))
}
generate_list <- generate(10000)
rnumber <- generate_list[[1]]
reject_num <- generate_list[[2]]


hist_rnum <- hist(rnumber[rnumber<20],breaks = 1000,plot = FALSE)
hist_scale <- hist_rnum$counts * (fx1_max / max(hist_rnum$counts))

curve(fx1, xlim = c(0,20), ylim = c(0,1.5*fx1_max), xlab = "x", ylab = "y")
curve(fx1_max*x^0, from = 0, to = fpx_Tmin , add = TRUE, col = "blue")
curve(fpx2, from = fpx_Tmin+ exp(-15) ,  add = TRUE, col = "red")
barplot(hist_scale, width = 20 / 1000, add = TRUE, space = 0)


mean(rnumber)
var(rnumber)
reject_num / (reject_num + length(rnumber))
fx_c <- 2
fpx_n <- 1.5+0.5*sqrt(fx_c)
fpx_alpha <- 1.2
fx1 <- \(x) sapply(x, fx, c = fx_c)
fx1_max <- fx1(fx_c^2/3)
fpx_Tmin <- fpx_n*(fpx_alpha - 1) / fx1_max

fpxn <- function(x, Tmin, alpha, n){
  if (x > Tmin) {
    fp <- n *((alpha - 1) / Tmin) * (x / Tmin)^(-alpha)
  } else {
    fp <- 0
  }
  return(fp)
}
fpx2 <- \(x) sapply(x, fpxn, Tmin = fpx_Tmin, alpha = fpx_alpha, n = fpx_n)

generate_list <- generate(10000)
```

```r
rnumber <- generate_list[[1]]
reject_num <- generate_list[[2]]

hist_rnum <- hist(rnumber[rnumber<20],breaks = 1000,plot = FALSE)
hist_scale <- hist_rnum$counts * (fx1_max / max(hist_rnum$counts))

curve(fx1, xlim = c(0,20), ylim = c(0,1.5*fx1_max), xlab = "x", ylab = "y")
curve(fx1_max*x^0, from = 0, to = fpx_Tmin , add = TRUE, col = "blue")
curve(fpx2, from = fpx_Tmin+ exp(-15) ,  add = TRUE, col = "red")
barplot(hist_scale, width = 20 / 1000, add = TRUE, space = 0)

mean(rnumber)
var(rnumber)
reject_num / (reject_num + length(rnumber))
# reject rate
r_laplace=function(gen_num,mu,alpha){
  unif=runif(gen_num,0,1)
  temp=sapply(unif,function(unif){
    if(unif>=0.5){
      return((-1/alpha)*log(2*(1-unif))+mu)
    }else{
      return((log(2*unif)/alpha)+mu)
    }
  })
  return(temp)
}
p4=ggplot(data=data.frame("y"=r_laplace(10000,0,1)),aes(x=y))+
  geom_histogram(bins=50)+
  labs(title="Laplace distribution generate by Inverse CDP method")
p4
ar_norm<-function(c){
  x<-NA
  num.reject<-0
  while (is.na(x)){
    y<-r_laplace(1,0,1)
    u<-runif(1)
    if (u<=dnorm(y,0,1)/(c*(exp(-1*abs(y))/2))){x<-y}
    else{num.reject<-num.reject+1}
  }
  c(x,num.reject)
}
df_norm=data.frame(t(data.frame(sapply(rep(sqrt((2*exp(1))/pi),2000),ar_norm))))
colnames(df_norm)=c("sample","reject")
p4=ggplot(data=df_norm,aes(x=sample))+
  geom_histogram(bins=50)+
  labs(title="Normal distribution generate by Acceptance/rejection method")
p4
mean_r=sum(df_norm[,2])/(2000+sum(df_norm[,2]))
ER=1-(1/sqrt((2*exp(1))/pi))

cat("The average rejection rate R: ",mean_r,"\nThe expected rejection rate ER: ",
    ER,"\nThe difference:",abs(ER-mean_r))
p5=ggplot(data=data.frame("x"=rnorm(2000,0,1)),aes(x=x))+
```

```
  geom_histogram(bins=50)+
  labs(title="Normal distribution generate by rnorm")
plot_comb1=grid.arrange(p4,p5)
```