computational statistic lab3

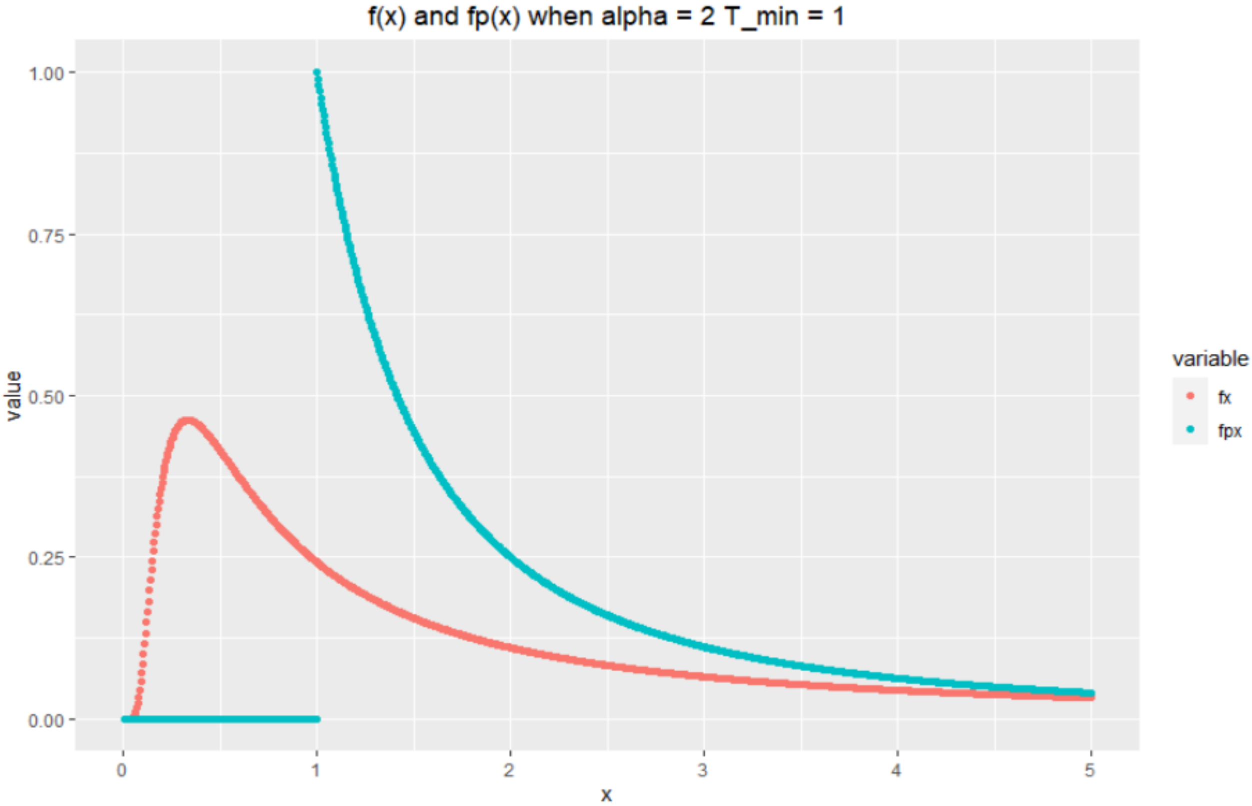
Group 12

11/16/2021

Question 1 **Stable distribution**

q1: plot f(x) and fp(x)

In the following plot, we use c = 1 in $f(x)$,and $\alpha = 2, T_{min} = 1 \ in \ f_p(x)$. The x is start from 0 to 10.



From the plot, we can see that there is a problem happened when x < 1($which \ is \ T_{min}$), so only $f_p(x)$ can not be applied to generate samples. So, we decided to use another function to replace power-low function in the interval(0, T_{min}).

we have $$\int_{T_{min}}^{\infty} f_p(x)\,dx = \int_{T_{min}}^{\infty} \frac{\alpha-1}{T_{min}}\left(\frac{x}{T_{min}}\right)^{-\alpha} dx = 1$$

and when $$x = T_{min}, \quad f_p(x) = \frac{\alpha-1}{T_{min}}$$

we consider $$f_q(x) = \frac{\alpha-1}{T_{min}}$$

$$\int_0^{T_{min}} f_q(x)\,dx = \int_0^{T_{min}} \frac{\alpha-1}{T_{min}} dx = \alpha - 1$$

and when $$x = T_{min}, \ f_q(x) = \frac{\alpha-1}{T_{min}}$$

If we set $f_m(x) = f_p(x) + f_q(x)$, then:

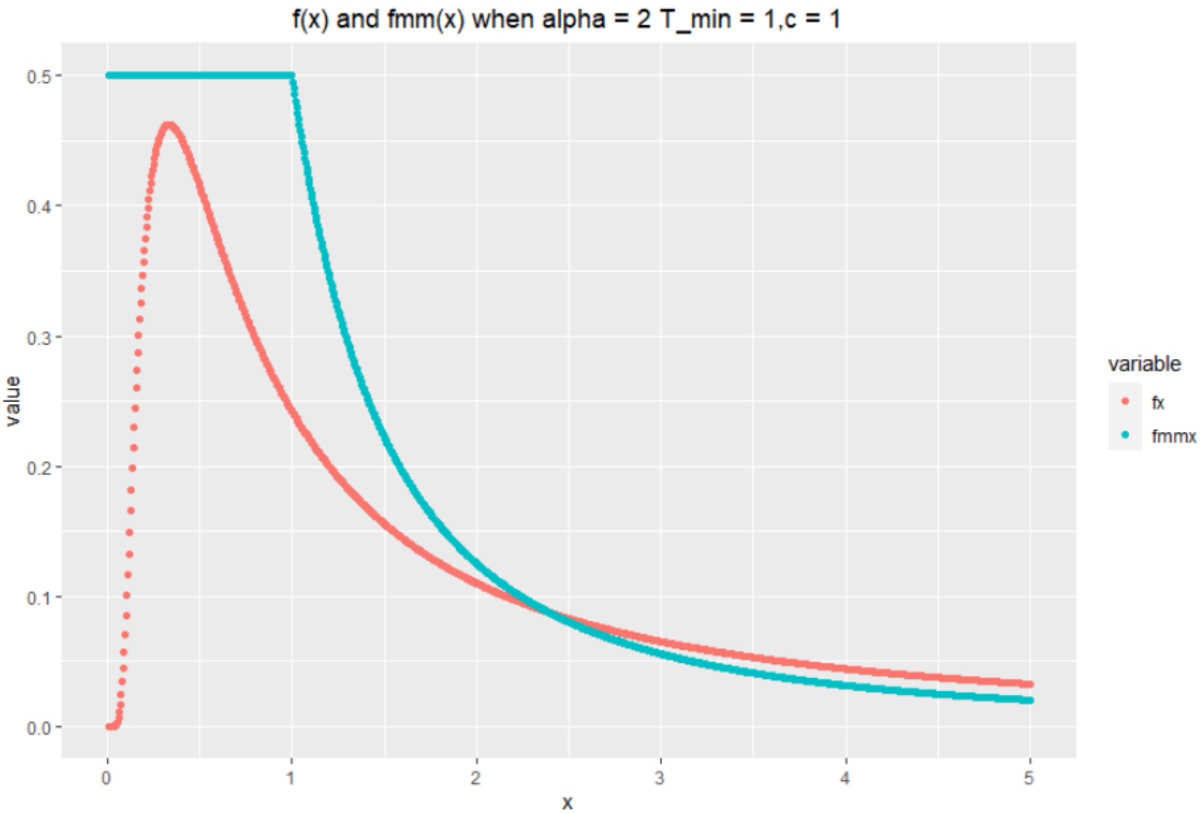
$$M = \int_0^\infty f_m(x)\,dx = (\alpha - 1) + 1 = \alpha$$

To make $f_m(x)$ a density function, we can use $f_{mm}(x) = \frac{f_m(x)}{\alpha}$.

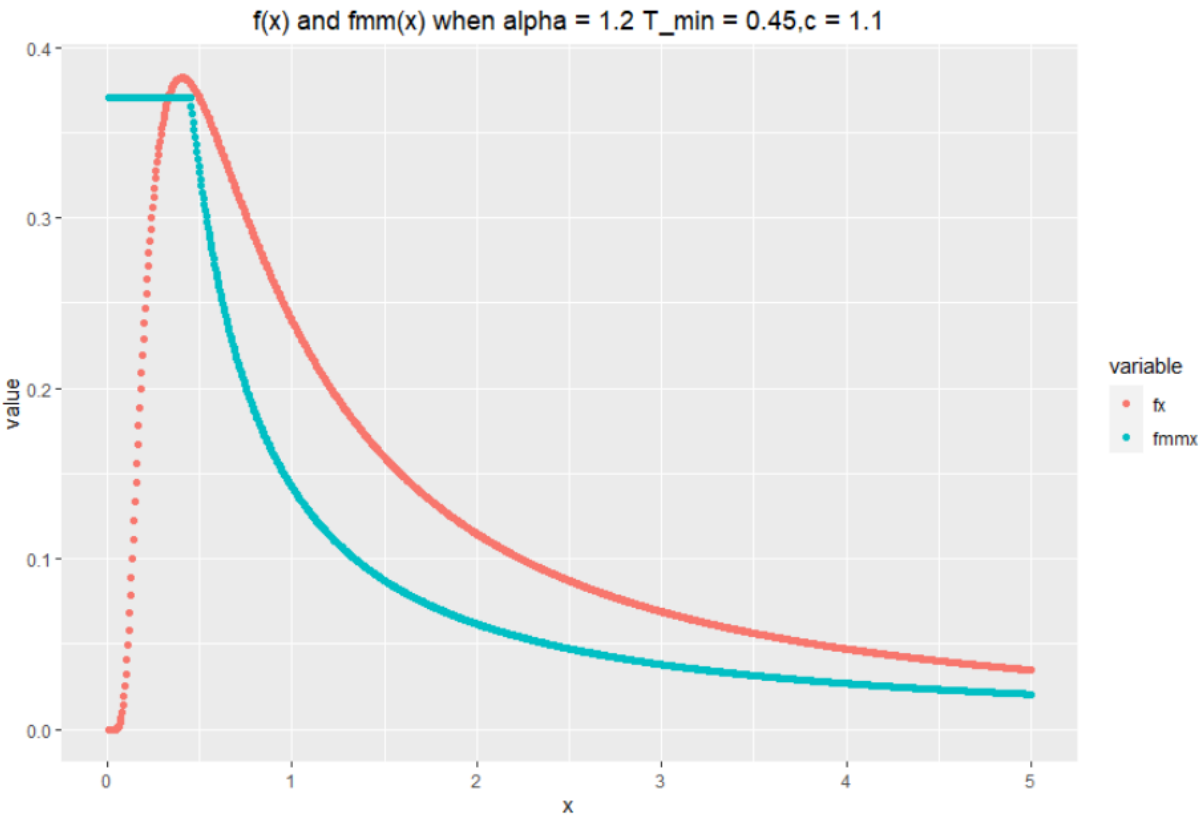
So, the $majorizing\ density$ is :

$$f_{mm}(x) = \begin{cases} \dfrac{(\alpha-1)}{\alpha T_{min}} & x \in (0, T_{min}] \\ \dfrac{(\alpha-1)}{\alpha T_{min}} * \left(\dfrac{x}{T_{min}}\right)^{-\alpha} & x \in (T_{min}, \infty) \end{cases}$$

Plot below shows the plot of $f_{mm}(x)$ when $\alpha = 2, T_{min} = 1$ and c=1 in $f(x)$.



If we want to use $f_{mm}(x)$ as majorizing density function, we have to make these two curves be similar to each other. So, we pick c = 1.1 $\alpha = 1.2$, $T_{min} = 0.45$. The plot below shows two curves with these parameters



q2: sampling

From question1, we use c = 1.1 $\alpha = 1.2$, $T_{min} = 0.45$.

From previous steps, we know function fmm(x) is the combination of two weighted density function. So if we want to sample from this function, we have to sample from these two density function respectively.

When x ∈ $(0, Tmin)$, we need to sample from $U\sim(0, T_{min})$

When x ∈ $(0, Tmin)$, we need to sample from $\frac{(\alpha-1)}{T_{min}} * \left(\frac{x}{T_{min}}\right)^{-\alpha}$

As for the proportion, we need to consider the area between curve and x-axis. In fmm(x), when c = 1.1 $\alpha = 1.2$, $T_{min} = 0.45$, the area of left part is:

$$\int_0^{T_{min}} \frac{\alpha - 1}{\alpha T_{min}} = \frac{\alpha - 1}{\alpha}$$

the area of right part is

$$\int_{T_{min}}^{\infty} \frac{(\alpha - 1)}{\alpha T_{min}} * \left(\frac{x}{T_{min}}\right)^{-\alpha} = \frac{1}{\alpha}$$

So, if we decide to generate **N** samples, then we should have $N * \frac{\alpha-1}{\alpha}$ from left part, and $\frac{N}{\alpha}$ from right part. Then, we can use functions(runif() and poweRlaw::rplcon()) to get samples.

To apply acceptance/rejection method, we need to find majorizing constant (will be called mc in the following text)first.

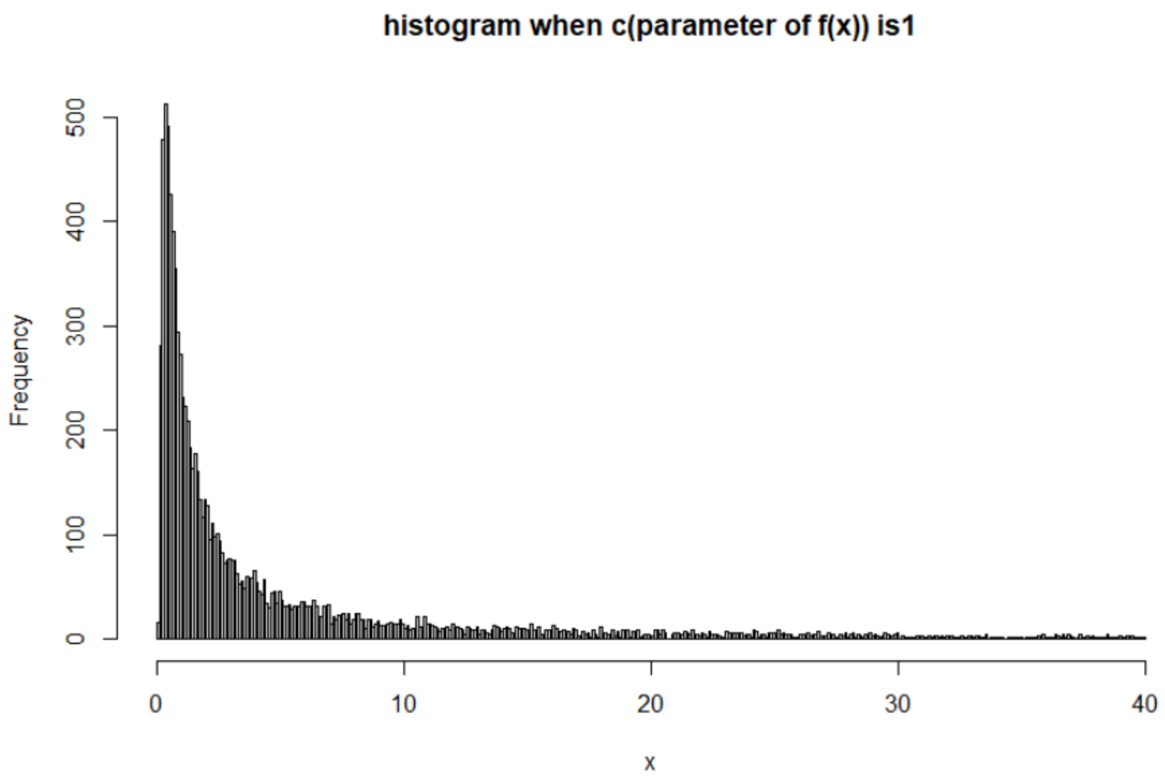
1) when $x \leq T_{min}$, $\left(\frac{f(x)}{f_{mm(x)}}\right)' \propto \exp\left(\frac{c^2}{-2x}\right) * (\frac{1}{2}x^{-3.5} - \frac{3}{2}x^{-2.5})$, so when x=$\frac{1}{3}$, we have mc = 1.003

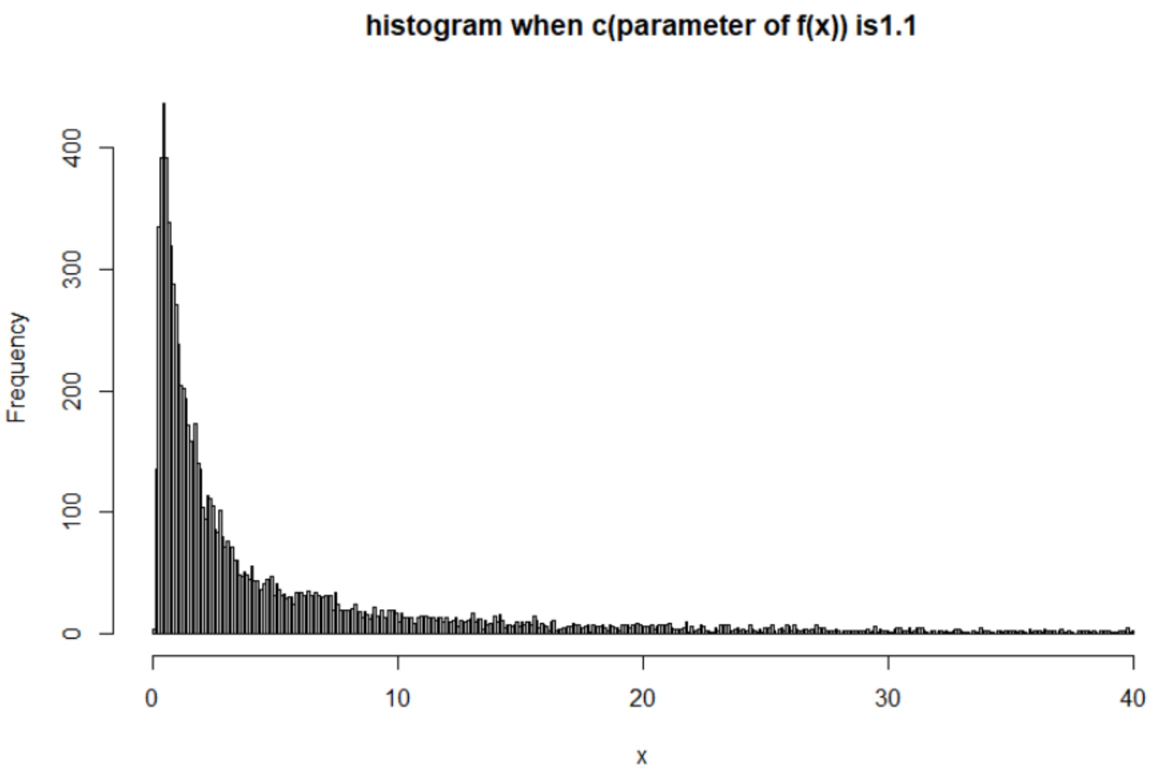
2) when $x \geq T_{min}$, $\left(\frac{f(x)}{f_{mm(x)}}\right)' \propto \exp\left(\frac{c^2}{-2x}\right) * (\frac{1}{2}x^{-3.5+\alpha} - (\frac{3}{2}-\alpha)x^{-2.5+\alpha})$, so when x=$\frac{5}{3}$, we have mc = 1.843

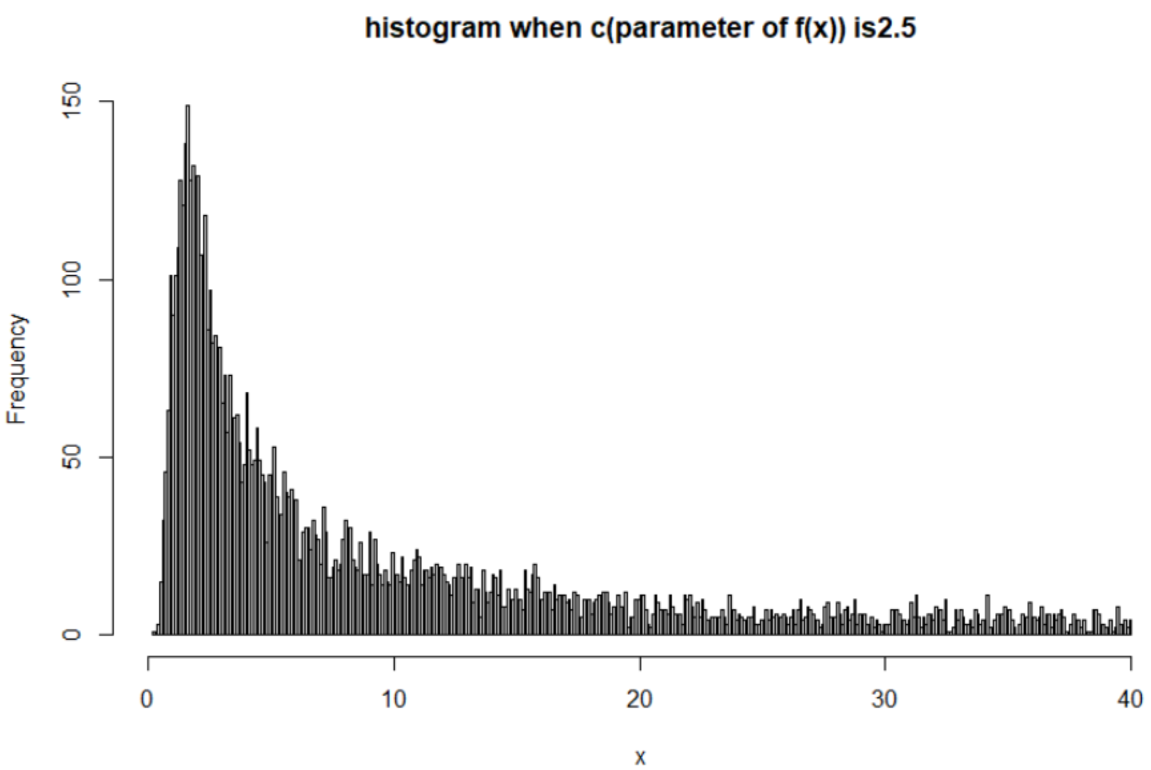
q3: sampling for different majorizing constant.

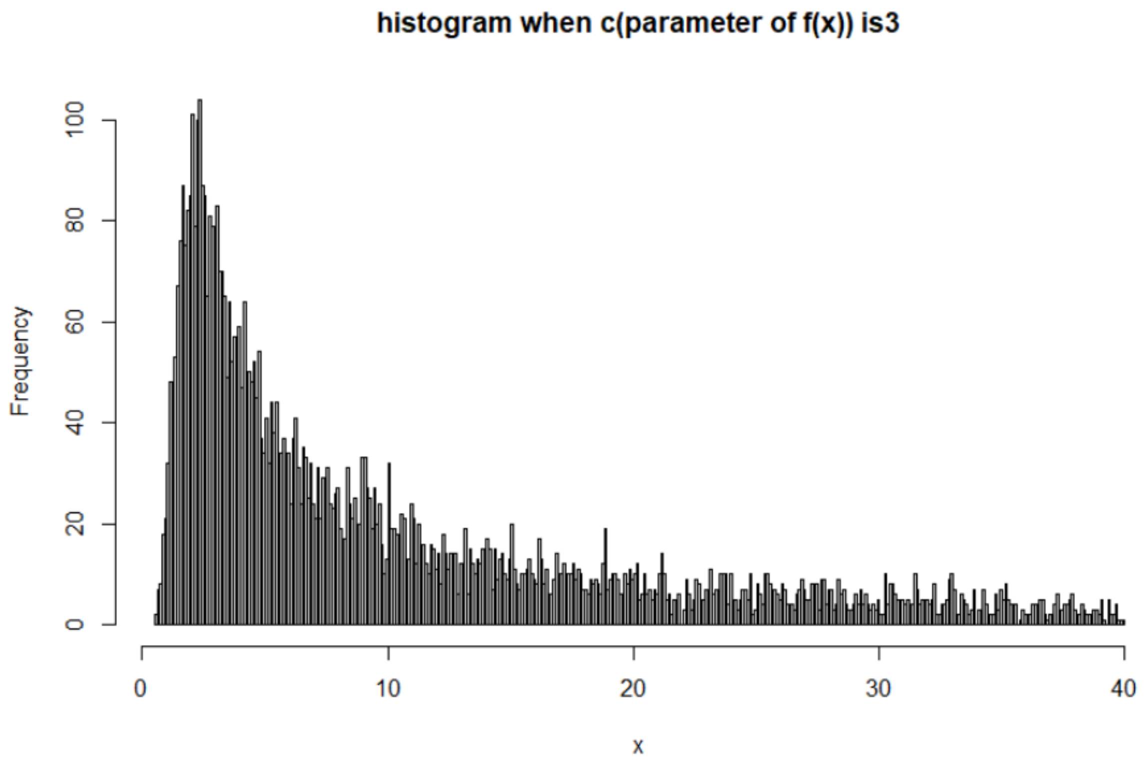
In this section, we have a list for c, 1, 1.1, 2.5 ,3.

Since the data is very large, the result will be scattered in (0,∞) (e.g. some data is like 0,01 and some data will reach 10^6 scale), we set a boundary for plotting (0,40).




histogram when c(parameter of f(x)) is1.1
Frequency
0
100
200
300
400
0
10
20
30
40
x


histogram when c(parameter of f(x)) is2.5
Frequency
0
50
100
150
0
10
20
30
40
x



The table below shows the variance, mean and reject rate regarding different parameter c in f(x).

	C =1	C =1.1	C =2.5	C =3
Mean	79863.653	5974.723	167184.433	360583.365
Variance	6.055782e+13	3.128015e+10	1.530487e+14	4.656850e+14
Reject rate	0.4395720	0.4578229	0.4224211	0.4550728

Mean value will increase with c increase.

Variance shares the similar trend with mean.

Reject rate, I think, will not only depends on c but also many aspects. All the parameters in the power-law function and the replace function will have great impact on reject rate.

Question 2 **Laplace distribution**

q1: Generate DE distribution

step 1: get the formula

Since the target distribution is DE(0,1), so the formula is:

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

Let Y from Unif(0,1), according to the inverse CDF method, we set that :

$$y = \frac{1}{2}e^{-|x|}$$

Solve this formula, we can get:

$$2y = e^{-|x|}$$

$$In(2y) = -|x|$$

$$|x| = -In(2y)$$

We need x both negative and positive, so we remove absolute sign in this way:

$$x = -In(2y) \qquad if\ y \leq 0.5$$

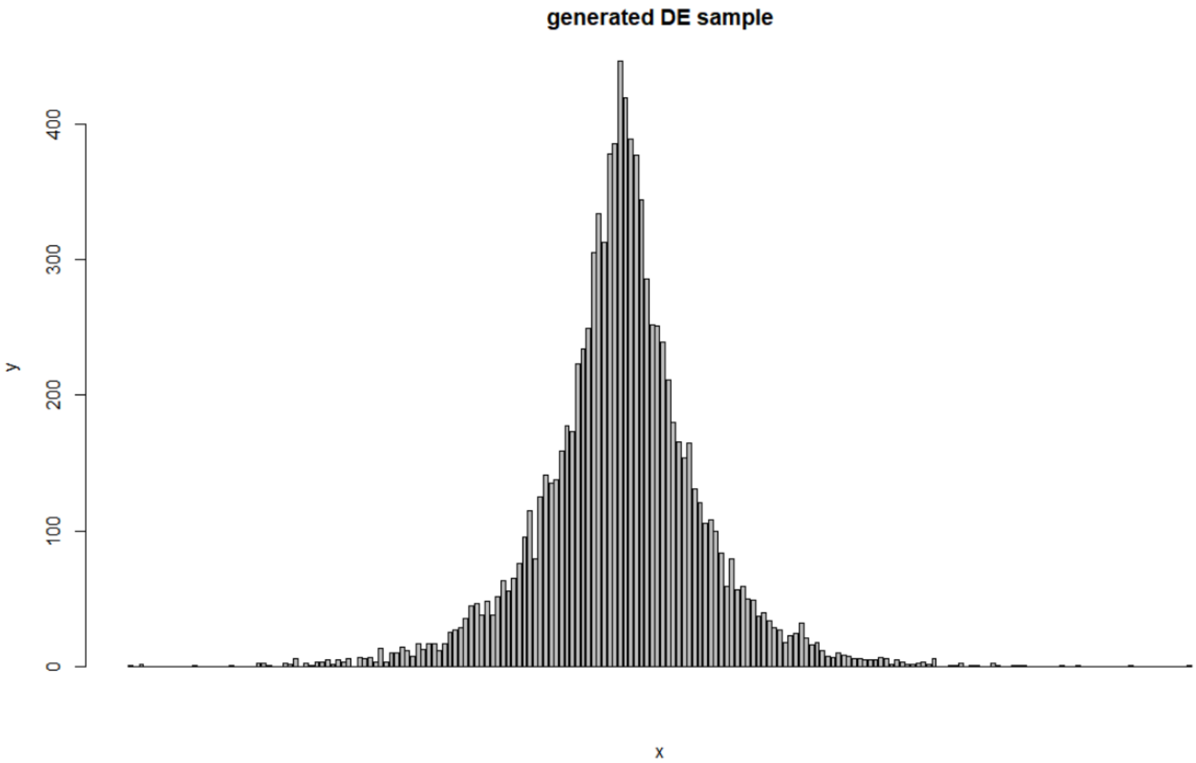
$$x = In(2-2y) \qquad if\ y > 0.5$$

step2: get x

Using function runif() to generate 10000 samples, we can take these samples as y, then get the generated samples.

After generating the samples, we can draw the histogram. Since these samples are randomly generated and computed from function Unif(), the results differ each time. The Laplace density function LA(0,1) should have mean value 0 and variance 2.

The histogram is like below(10000 samples with mean value -0.00934368, variance 1.989046)



q2: Generate normal distribution

step 1: get constant C

Since we must make sure that for every valid x, c*DE(x) >N(x).

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

we move x to right side and remain only constant c in the left side:

$$In\left(\frac{\sqrt{2\pi} * C}{2}\right) \geq \frac{-(X-1)^2 + 1}{2}$$

X = 1 is valid, and when x = 1, the right side reach the maximum value $\frac{1}{2}$. So, we will get:

$$In\left(\frac{\sqrt{2\pi} * C}{2}\right) \geq \frac{1}{2}$$

$$C \geq \frac{2e^{\frac{1}{2}}}{\sqrt{2\pi}}$$

So, we will choose C = 1.3155.

step 2: get sample from DE (0,1)

We already get good result in q1, so we just pick 2000 samples from vector ‘x’ in q1.

step3: apply Acceptance/Rejection method

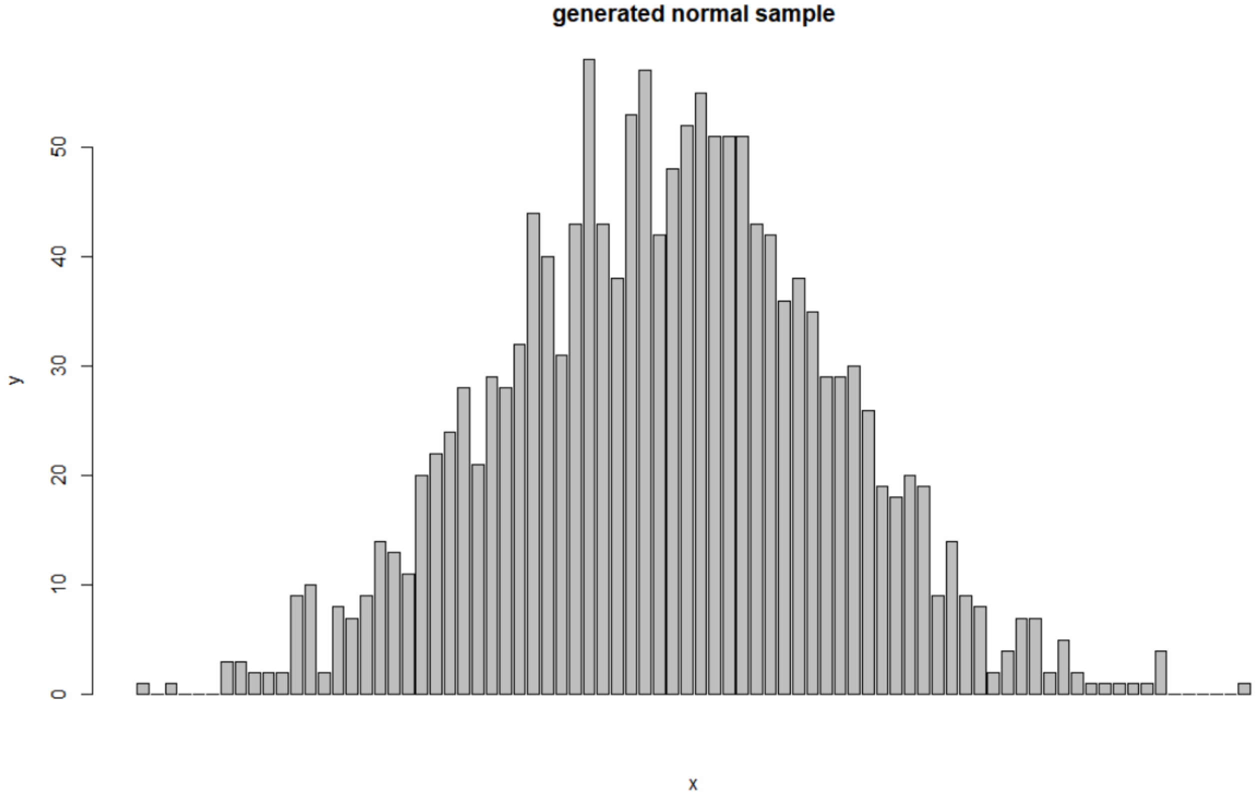
In this step, we will have a loop runs 2000 times. Each loop will pick a sample from what we have in step2(we will call this sample x in the following text), a random sample from function runif (like **runif(1)**) and a random value generated by **dnorm(x)**. Then:

$$p = \frac{dnorm(x)}{c * DE_{(0,1)}(x)} \quad and \quad U = runif(1)$$

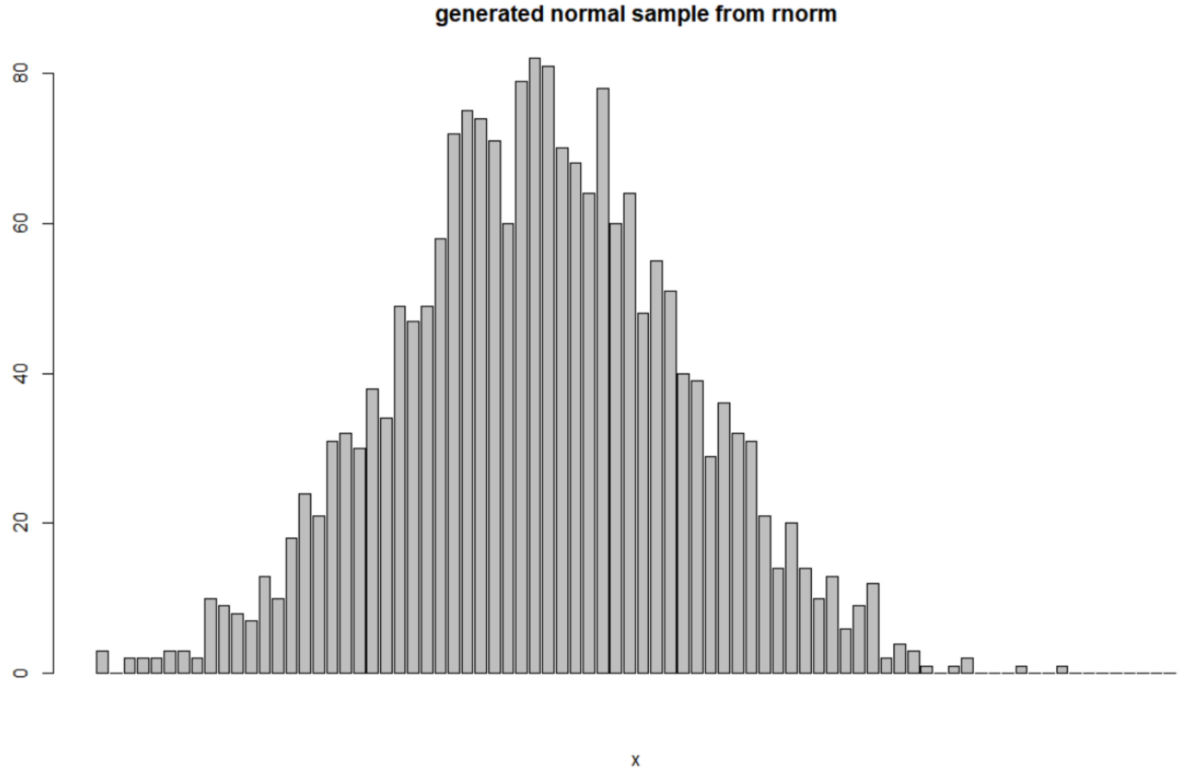
$$if\ p \geq U, accept\ x; \quad else, reject\ x$$

Applying the data from **q1** in these three steps, we have the reject number: 478. So, the reject rate is 0.239.By generating the sample 300 times, we learn that average reject rate is 0.242 The estimated reject rate is 0.2398 (1/C), which is very close to our result.

The histogram of these 2000 samples is:



The histogram of samples from function rnorm() is:



These two histograms show that samples from Acceptance/Rejection method has similar distribution as samples from normal distribution.

Appendix:

Question 1

```
#q1
library(ggplot2)

library(reshape2)

library(poweRlaw)

get_fpx_origin <- function(alpha,t_min,x)
{
  x <- x[order(x)]
  s <- x[which(x<t_min)]
  x <- x[-which(x<t_min)]
  res <- rep(0,length(s))
  res <- c(res,(alpha-1)/t_min*(x/t_min)^(-alpha))
  return(res)
}
get_fpx <- function(alpha,t_min,x)
{
  x <- x[order(x)]
  s <- x[which(x<t_min)]
  x <- x[-which(x<t_min)]
  res <- rep((alpha-1)/t_min,length(s))
  res <- c(res,(alpha-1)/t_min*(x/t_min)^(-alpha))
  res <- res/alpha
  return(res)
}
# plot for f(x) and fp(x)
c <- 1
t_min <- 1
alpha <- 2
x_f <- 1:1000/200
fx <- c/sqrt(2*pi)*exp(-c*c/2/x_f)*x_f^(-3/2)
fpx <- get_fpx_origin(alpha,t_min,x_f)
df <- data.frame(x = x_f,fx =fx,fpx=fpx)
df1 <- melt(df,id.vars='x')

p1 <-ggplot(df1,aes(x=x,y=value))+
  geom_point(aes(color=variable))+
  ggtitle('f(x) and fp(x) when alpha = 2 T_min = 1')+
  theme(plot.title = ggplot2::element_text(hjust=0.5))
print(p1)

# plot for fmm(x) and f(x)
c <- 1
t_min <- 1
alpha <- 2
x_f <- 1:1000/200
fx <- c/sqrt(2*pi)*exp(-c*c/2/x_f)*x_f^(-3/2)
fpx2 <- get_fpx(alpha,t_min,x_f)
df <- data.frame(x = x_f,fx =fx,fmmx=fpx2)
df2 <- melt(df,id.vars='x')
```

```r
# plot
p2 <-ggplot(df2,aes(x=x,y=value))+
  geom_point(aes(color=variable))+
  ggtitle('f(x) and fmm(x) when alpha = 2 T_min = 1,c = 1')+
  theme(plot.title = ggplot2::element_text(hjust=0.5))
print(p2)



# plot for fmm(x) and f(x)
c <- 1.1
t_min <- 0.45
alpha <- 1.2
x_f <- 1:1000/200
fx <- c/sqrt(2*pi)*exp(-c*c/2/x_f)*x_f^(-3/2)
fpx2 <- get_fpx(alpha,t_min,x_f)
df <- data.frame(x = x_f,fx =fx,fmmx=fpx2)
df2 <- melt(df,id.vars='x')
# plot
p2 <-ggplot(df2,aes(x=x,y=value))+
  geom_point(aes(color=variable))+
  ggtitle('f(x) and fmm(x) when alpha = 1.2 T_min = 0.45,c = 1.1')+
  theme(plot.title = ggplot2::element_text(hjust=0.5))
print(p2)

##########################################################################
#q2
new_powerLaw_sampling <- function(nsamples,alpha,t_min)
{

  n1 <- nsamples*(alpha-1)/(alpha)
  n2 <- nsamples - n1
  first_part <- runif(n1,0,t_min)
  second_part <- rplcon(n2,t_min,alpha)
  return(c(first_part,second_part))
}
get_sample <- function(mc,new_samples,alpha,t_min,c_in_fx){
  accept_sample <- c()
  for (i in 1:length(new_samples))
  {
    y <- new_samples[i]
    u <- runif(1)
    fxy <- c_in_fx/sqrt(2*pi)*exp(-c_in_fx*c_in_fx/2/y)*y^(-3/2)
    if(y <= t_min)
    {
      fyy <- (alpha-1)/t_min/alpha
    }else
    {
      fyy <- ((alpha-1)/alpha/t_min)*(y/t_min)^-alpha
    }
    if(u <= fxy/(fyy*mc))
    {accept_sample <- c(accept_sample,y)}
  }
  return(accept_sample)
```

```
}
new_samples <- new_powerLaw_sampling(20000,alpha,t_min)
res <- get_sample(1.8433,new_samples,alpha,t_min,c)
print(1-length(res)/length(new_samples))

##########################################################################
######
#q3
c_in_fx_list <- c(1,1.1,2.5,3)
t_min <- 0.45
alpha <- 1.2
reject_rate <- c()
mean <- c()
variance <- c()
for (i in c_in_fx_list) {
  # get samples
  new_samples <- new_powerLaw_sampling(20000,alpha,t_min)
  x1 <- 1/(3-2*alpha)
  mc1 <- (i/sqrt(2*pi)*exp(-i*i/2/(x1))*(x1)^(-3/2))/((alpha-1)/t_min/alph
a*(x1/t_min)^(-alpha))
  mc2 <- (i/sqrt(2*pi)*exp(-i*i/2/(1/3))*(1/3)^(-3/2))/((alpha-1)/t_min/al
pha)
  res <- get_sample(max(mc1,mc2,1),new_samples,alpha,t_min,i)
  # get some data
  reject_rate <- c(reject_rate,(length(new_samples)-length(res))/length(ne
w_samples))
  mean <- c(mean,mean(res))
  variance <- c(variance,var(res))
  # plot histogram
  hist(res[which(res<=40)],xlab = 'x',main =  paste0('histogram when c(par
ameter of f(x)) is',i),breaks = 400)
}

print(mean)

print(variance)

print(reject_rate)
```

Question 2

```
library(ggplot2)
library(reshape2)

library(poweRlaw)

# q1
unif_sample <- runif(10000,0,1)
DE_sample  <- c()
for (i in 1:10000)
{
  if(unif_sample[i] > 0.5)
    DE_sample <-c(DE_sample,log(2-2*unif_sample[i]))
  else
    DE_sample <-c(DE_sample,-log(2*unif_sample[i]))
}

# interval [n*gap,(n+1)*gap] will be used in the histogram
gap <- (max(DE_sample)-min(DE_sample))/200
# count will be vector used to make histogram
count <-c()
start <- min(DE_sample)
end <- min(DE_sample) + gap
for(i in 1:200)
{
   temp <- DE_sample[which(DE_sample >= start)]
   temp <- temp[which(temp < end)]
   count <- c(count,length(temp))
   start <- start + gap
   end <- end + gap
}
barplot(count,xlab = 'x',ylab = 'y',main = 'generated DE sample')
```

```
# q2
#
c = 1.3155
index <- 1:10000
DE_2000 <- DE_sample[which(index%%5==0)]
reject_num <- 0
accept_sample <- c()
#for(j in 1:300){
for (i in 1:2000)
{
  y <- DE_2000[i]
  # fxx is normal density at x
  u <- runif(1)
  fxy <- dnorm(y)
  fyy <- 1/2 *exp(-abs(y))
  if(u <= fxy/(fyy*c))
  {accept_sample <- c(accept_sample,y)}
  else
  {
    reject_num <- reject_num + 1
  }
}
#}
#reject_num/300/2000
# ER  is 1-1/c
gap <- (max(accept_sample)-min(accept_sample))/80
# count will be vector used to make histogram
count <-c()
start <- min(accept_sample)
end <- min(accept_sample) + gap
for(i in 1:80)
{
  temp <- accept_sample[which(accept_sample >= start)]
  temp <- temp[which(temp < end)]
  count <- c(count,length(temp))
  start <- start + gap
  end <- end + gap
}
barplot(count,xlab = 'x',ylab = 'y',main = 'generated normal sample')

r<-rnorm(2000)
# ER  is 1-1/c
gap <- (max(r)-min(r))/80
# count will be vector used to make histogram
count <-c()
start <- min(accept_sample)
end <- min(accept_sample) + gap
for(i in 1:80)
{
  temp <- r[which(r >= start)]
  temp <- temp[which(temp < end)]
  count <- c(count,length(temp))
  start <- start + gap
  end <- end + gap
```

```
}
barplot(count,xlab = 'x',ylab = 'y',main = 'generated normal sample from r
norm')
```