Homework 3

Fang, Zheng
Zhao, Qian

2018/2/28

1.

$$Q(\Psi|\Psi^{(k)}) = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)}\{\ln\pi_j + \ln\frac{1}{\sqrt{2\pi}\sigma} + \frac{1}{2\sigma^2}(y_i - x_i^T\beta_j)^2\}$$
$$= \sum_{i=1}^{n}\sum_{j=1}^{m} p^{(k+1)}\ln[\pi_j\cdot(2\pi)^{-\frac{1}{2}}] - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} p^{(k+1)}\ln\sigma^2 - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} p^{(k+1)}(y_i - x_i^T\beta_j)^2\cdot\sigma^2$$

let

$$I_1 = \sum_{i=1}^{n}\sum_{j=1}^{m} p^{(k+1)}\ln[\pi_j\cdot(2\pi)^{-\frac{1}{2}}]$$
$$I_2 = \sum_{i=1}^{n}\sum_{j=1}^{m} p^{(k+1)}\ln\sigma^2$$
$$I_3 = \sum_{i=1}^{n}\sum_{j=1}^{m} p^{(k+1)}(y_i - x_i^T\beta_j)^2\cdot\sigma^2$$

only I_3 contains β_j, find β_j to minimize each I_{3j}

$$I_{3j} = \sum_{i=1}^{n} p^{(k+1)}(y_i - x_i^T\beta_j)^2\cdot\sigma^2$$
$$\frac{dI_{3j}}{d\beta_j} = \sum_{i=1}^{n} p^{(k+1)}\sigma^2(-2x_iy_i + 2x_ix_i^T\beta_j)) = 0$$
$$\sum_{i=1}^{n} p^{(k+1)}x_ix_i^T\beta_j = \sum_{i=1}^{n} p^{(k+1)}x_iy_i$$
$$\beta^{(k+1)} = (\sum_{i=1}^{n} p^{(k+1)}x_ix_i^T)^{-1}(\sum_{i=1}^{n} x_ip^{(k+1)}y_i), j = 1, ..., m$$

only I_2 and I_3 contain σ^2,

$$\frac{d(I_2 + I_3)}{d\sigma^2} = \sum_{i=1}^{n}\sum_{j=1}^{m} p^{(k+1)}\sigma^{-2} - \sum_{i=1}^{n}\sum_{j=1}^{m} p^{(k+1)}(y_i - x_i^T\beta_j)^2(\sigma^{-2})^{-2} = 0$$
$$n\sigma^2 - \sum_{i=1}^{n}\sum_{j=1}^{m} p^{(k+1)}(y_i - x_i^T\beta_j)^2 = 0$$
$$\sigma^{2(k+1)} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} p^{(k+1)}(y_i - x_i^T\beta_j)^2}{n}$$

only I_1 contains π_j to minimize each π_j

$$\frac{dI_1}{d\pi_j} = \sum_{i=1}^{n} p^{(k+1)}ij \frac{1}{\pi_j} = 0$$
$$\pi_j = \frac{1}{n}\sum_{i=1}^{n} p^{(k+1)}$$

2.

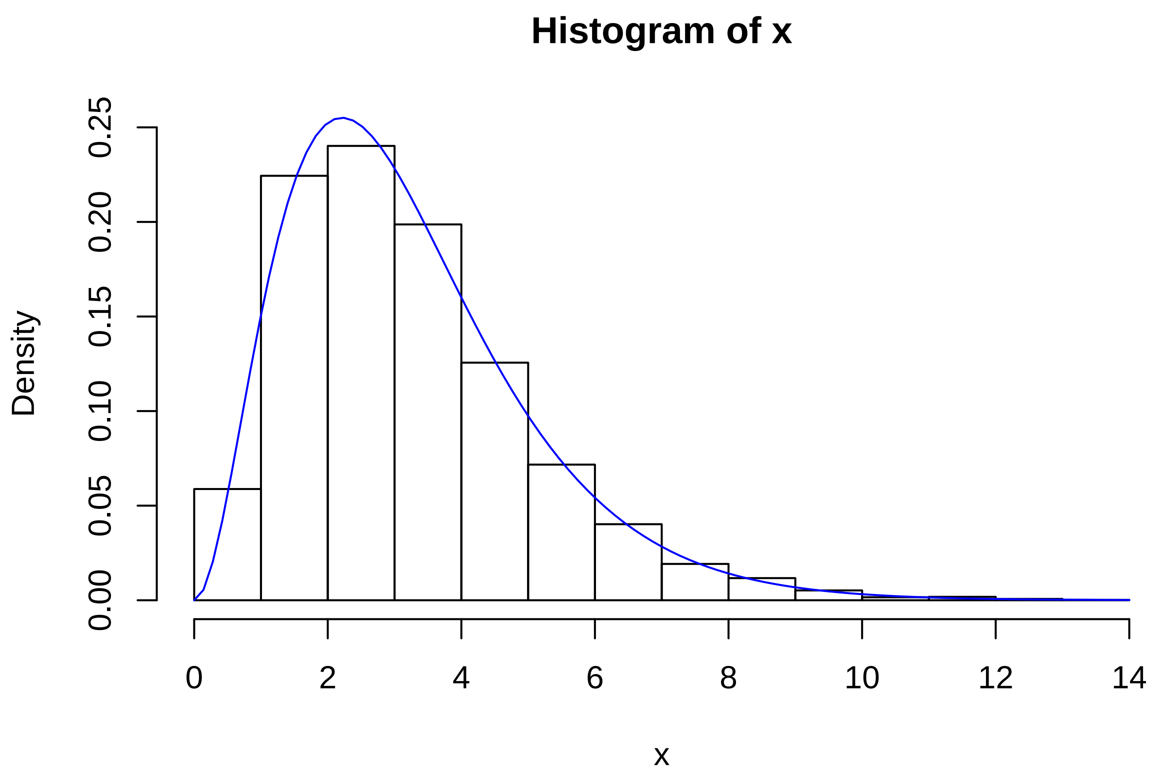
(a)

$$C\int_0^{+\infty}(2x^{\theta-1}+x^{\theta-\frac{1}{2}})e^{-x}dx = 1$$
$$C\int_0^{+\infty}[\Gamma(\theta)\cdot 2\cdot(\frac{e^{-x}x^{\theta-1}}{\Gamma(\theta)}) + \Gamma(\theta+\frac{1}{2})(\frac{e^{-x}x^{\theta-\frac{1}{2}}}{\Gamma(\theta+\frac{1}{2})})]dx = 1$$
$$C[2\Gamma(\theta)\int_0^{+\infty}\frac{e^{-x}x^{\theta-1}}{\Gamma(\theta)}dx + \Gamma(\theta+\frac{1}{2})\int_0^{+\infty}\frac{e^{-x}x^{\theta-\frac{1}{2}}}{\Gamma(\theta+\frac{1}{2})})dx] = 1$$
$$C[2\Gamma(\theta)+\Gamma(\theta+\frac{1}{2})] = 1$$
$$C = \frac{1}{2\Gamma(\theta)+\Gamma(\theta+\frac{1}{2})}$$

(b)

θ can be any positive number, for example $\theta = 3$,

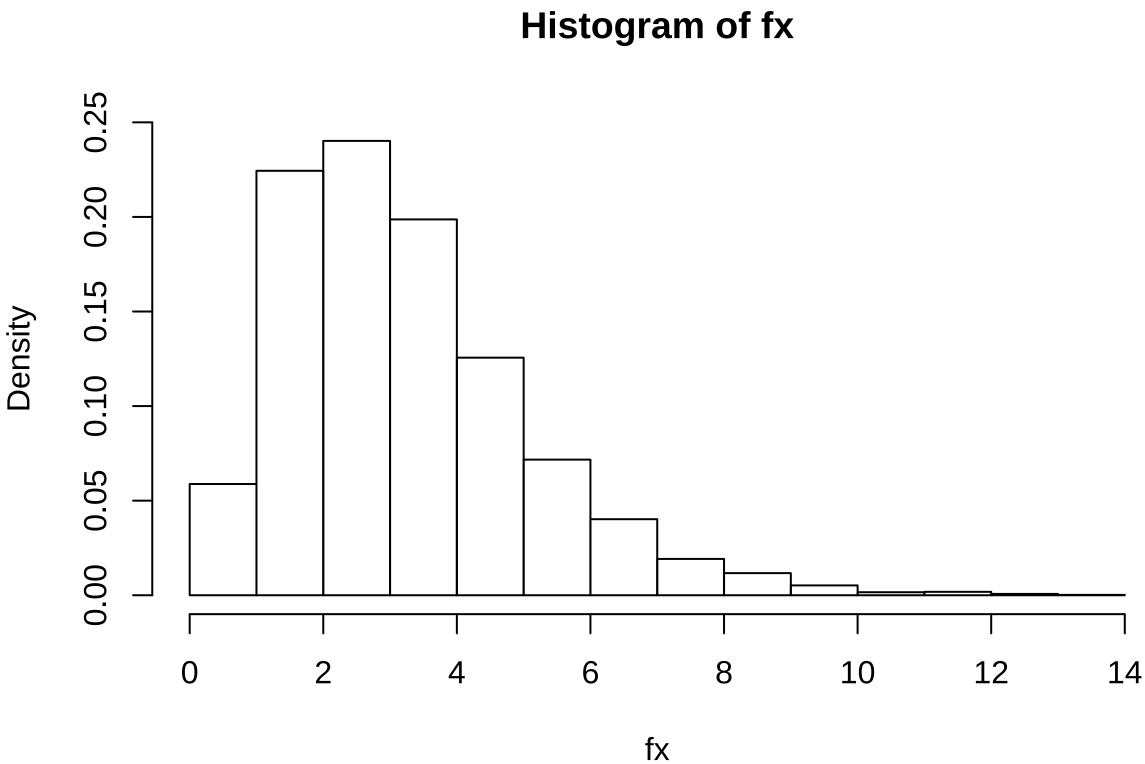
```
# get pdf g(x)
g <- function(x){
  return(C*(2*x^(theta - 1) + x^(theta - 0.5))*exp(-x))
}
# get cdf G(x)
G <- function(x){
  return(integrate(g, 0, x)$value)
}
# get invsese cdf
InverseG <- function(y){
  return(uniroot(function(x) G(x) - y, lower = 0, upper = 20, tol = 1e-9)$root)
}

# let theta = 3
theta <- 3
C <- 1/(2*gamma(theta)+gamma(theta + 0.5))
n <- 10000
#
y <- runif(n, 0, 1)
x <- sapply(y, InverseG)
hist(x, ylim = c(0, 0.25), prob = TRUE)
curve(g(x), add = TRUE, col="blue")
```



(c)

$$
\begin{aligned}
q(x) &= \sqrt{4+x}\cdot x^{\theta-1}e^{-x} \\
g(x) &= C(2x^{\theta-1}+x^{\theta-\frac{1}{2}})e^{-x} \\
\frac{q(x)}{\alpha g(x)} &= \frac{\sqrt{4+x}}{\alpha C(2+\sqrt{x})} \le 1 \\
& \frac{\sqrt{4+x}}{C(2+\sqrt{x})} \le \alpha \\
\alpha &= \sup \frac{\sqrt{4+x}}{C(2+\sqrt{x})} = \frac{1}{C} \\
\therefore \frac{q(x)}{\alpha g(x)} &= \frac{\sqrt{4+x}}{2+\sqrt{x}}
\end{aligned}
$$

```r
# rejection sampling
RejectionS <- function(x){
  frac <- (4 + x)^0.5/(2 + x^0.5)
  if(frac<1){
    return(x)
  }
}
fx <- sapply(x, RejectionS)
hist(fx, ylim = c(0, 0.25),prob = TRUE)
```



3.

(a)

For $0 < x < 1$

$$\frac{1}{1+x^2} \leq 1$$
$$\sqrt{2+x^2} \leq \sqrt{3}$$
$$f(x) = \frac{x^{\theta-1}}{1+x^2} + \sqrt{2+x^2} \cdot (1-x)^{\beta-1} < x^{\theta-1} + \sqrt{3} \cdot (1-x)^{\beta-1}$$

and let U ~ Unif(0.1), Observe that for any $\theta, \beta > 0$,
let $U^{1/a} \sim Beta(a, 1)$, and $1 - U^{1/b} \sim Beta(1, b)$ with the densities of the distributions of

$$g_\theta(x) = \theta x^{\theta-1}$$
$$g_\beta(x) = \beta(1-x)^{\beta-1}$$

let

$$g(x) = (1-\lambda)g_\theta(x) + \lambda g_\beta(x)$$

here we choice $\lambda = \frac{1}{2}$ to simplify our computation since

$$f(x) < x^{\theta-1} + \sqrt{3} \cdot (1-x)^{\beta-1}$$

then to get $f(x) < \alpha g(x)$ we must have

$$\alpha g(x) = \alpha \cdot \frac{1}{2} \cdot \theta \cdot x^{\theta-1} + \alpha \cdot \frac{1}{2} \cdot \beta \cdot (1-x)^{\beta-1} \geq x^{\theta-1} + \sqrt{3} \cdot (1-x)^{\beta-1}$$

which equals to

$$1 \leq \frac{1}{2} \cdot \alpha \cdot \theta$$

$$\sqrt{3} \leq \frac{1}{2} \cdot \alpha \cdot \beta$$

therefore

$$\alpha = max(\frac{2}{\theta}, \frac{2\sqrt{3}}{\beta})$$

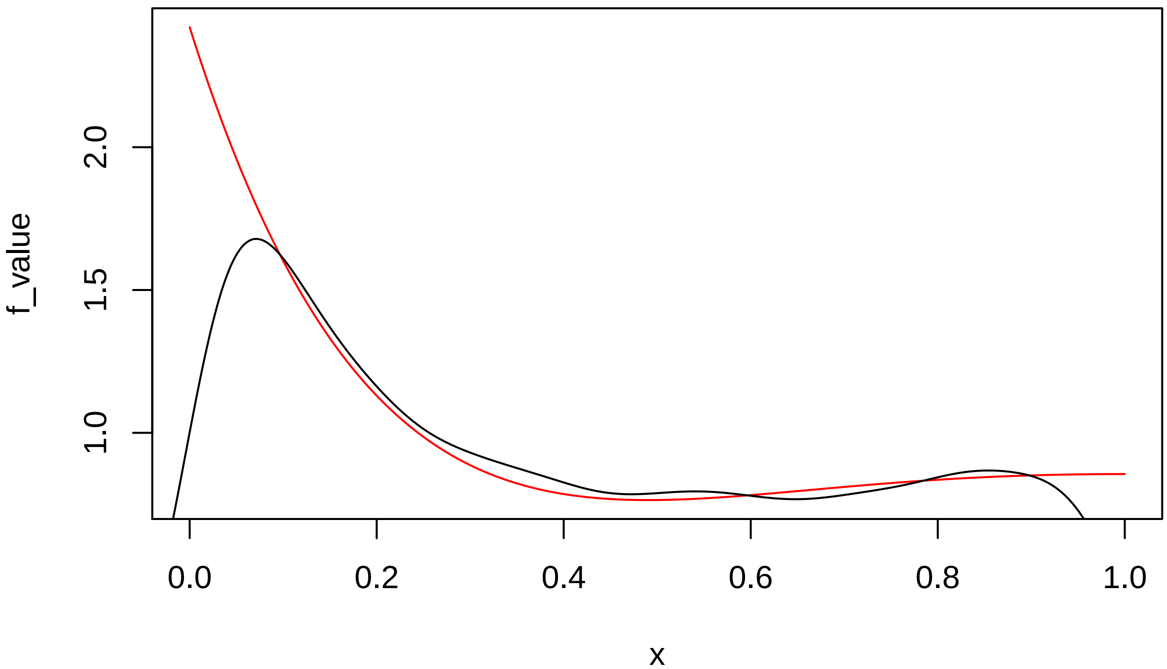
and here we set $(\theta, \beta) = (2, 6)$ so we get $\alpha = 1$

```
#you can change the value of theta and beta
theta_factor <- 2
beta_factor <- 6
# W1 <- (C1^2) / (C1 + C2)#weight for g1(x)
# W2 <- (C2^2) / (C1 + C2)#weight for g2(x)
W1 <- 1/2#weight for g1(x)
W2 <- 1/2#weight for g2(x)
alpha_factor <- max(2/theta_factor, 2/beta_factor)

#pdf of g(x)
func_g1 <- function(x){
  g1 <- theta_factor * (x ^ (theta_factor-1))
  return(g1)
}

func_g2 <- function(x){
  g2 <- (beta_factor) * ((1-x)^(beta_factor-1))
  return(g2)
}

func_g_mix <- function(x){
  g1 <- func_g1(x)
  g2 <- func_g2(x)
  return(W1*g1+W2*g2)
}
# x <- seq(from = 0, to = 1, by = 0.001)
# cdf_g <- c()
# for(i in 1:length(x)){
#   cdf_g <- c(cdf_g, integrate(func_g, lower = 0, upper = x[i])$value)
# }
# #g_value <- sapply(x,func_g)
# plot(cdf_g~x,type = "l", xlab = "x", ylab = "density")
#calculate the inverse of g-unit(1)
inverse_function <- function(x){
  #u <- runif(1)
  cdf_g <- integrate(func_g, lower = 0, upper = x)$value
  inverse_value <- cdf_g - u
  return(inverse_value)
}

#pdf of f
func_f <- function(x){
  f1 <- (x ^ (theta_factor-1)) / (1+x^2)
```

```r
  f2 <- ((2+x^2)^0.5) * ((1-x)^(beta_factor - 1))
  return(f1+f2)
}
func_f_true <- function(x){
  value <- func_f(x) * const
  return(value)
}
#solve for the constant for f_pdf
integrate(func_f, lower = 0, upper = 1)$value
```

```
## [1] 0.5843471
```

```r
const <- 1/integrate(func_f, lower = 0, upper = 1)$value
#true value of f_pdf
x <- seq(from = 0, to = 1, by = 0.001)
f_value <- sapply(x,func_f_true)

f_sample_rejection <-c()
while(length(f_sample_rejection)<10000){
  u_weight <- runif(1)
  if(u_weight < 0.5){
    u <- runif(1)
    func_g <- func_g1
    sample_value <- uniroot(inverse_function,interval = c(0,1))$root
  }
  else{
    u <- runif(1)
    func_g <- func_g2
    sample_value <- uniroot(inverse_function,interval = c(0,1))$root
  }
  sample_value_g <- func_g_mix(sample_value)
  sample_value_f <- func_f(sample_value)
  ratio <- sample_value_f / (alpha_factor * sample_value_g)
  rejection_value <- runif(1)
  if(ratio >= rejection_value){
    f_sample_rejection <- c(f_sample_rejection, sample_value)
  }
}

sample_pdf <- density(f_sample_rejection)

#plot the true density and the sample density in one figure
plot(f_value~x,type = "l",col = "red")
lines(sample_pdf,type = "l", xlab = "x", ylab = "density", main = "using rejection method to get sample
legend(0.6,4.5,c("sample density", "true density of f"), col = c("black","red"),text.col = c("black","r
```




```
# str(f_sample_rejection)
# max(f_sample_rejection)
```

(b)

Now we dealing with the two components of f(x)

$$f_1(x) = \frac{x^{\theta-1}}{1+x^2}$$
$$f_2(x) = \sqrt{2+x^2} \cdot (1-x)^{\beta-1}$$

and still we have

$$\frac{1}{1+x^2} \leq 1\sqrt{2+x^2} \leq \sqrt{3}$$

so when

$$(\alpha_1, \alpha_2) = (\frac{1}{\theta}, \frac{\sqrt{3}}{\beta})$$

$$\alpha_1 g_1(x) \geq f_1(x)\alpha_2 g_2(x) \geq f_2(x)$$

and still we use $(\theta, \beta) = (2, 6)$ so we got $(\alpha_1, \alpha_2) = (\frac{1}{2}, \frac{\sqrt{3}}{6})$ when $Unif(0,1) > \frac{\alpha_1}{\alpha_1+\alpha_2} = \frac{3-\sqrt{3}}{2}$ we sample from $g_2(x)$ otherwise we sample from $g_1(x)$

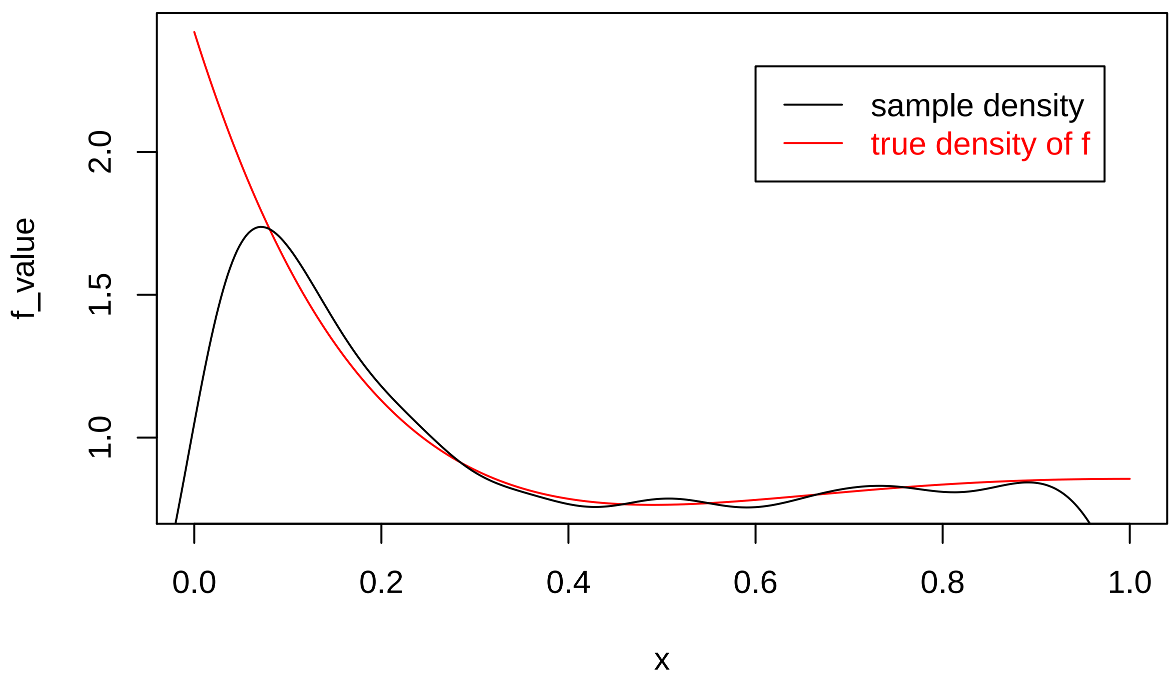
```
#you can change the value of theta and beta
theta_factor <- 2
beta_factor <- 6
alpha_1 <- 1 / theta_factor
alpha_2 <- (3^0.5) / beta_factor

weight_boundary <-alpha_1 / (alpha_1 + alpha_2)
#pdf of g(x)
func_g1 <- function(x){
  g1 <- theta_factor * (x ^ (theta_factor-1))
```

```
  return(g1)
}
func_g2 <- function(x){
  g2 <- (beta_factor) * ((1-x)^(beta_factor-1))
  return(g2)
}

# x <- seq(from = 0, to = 1, by = 0.001)
# cdf_g <- c()
# for(i in 1:length(x)){
#   cdf_g <- c(cdf_g, integrate(func_g, lower = 0, upper = x[i])$value)
# }
# #g_value <- sapply(x,func_g)
# plot(cdf_g~x,type = "l", xlab = "x", ylab = "density")
#calculate the inverse of g-unit(1)
inverse_function <- function(x){
  cdf_g <- integrate(func_g, lower = 0, upper = x)$value
  inverse_value <- cdf_g - u
  return(inverse_value)
}
##--------------Q3-b----------------------
#pdf of f
func_f1 <- function(x){
  f1 <- (x ^ (theta_factor-1)) / (1+x^2)
  return(f1)
}

func_f2 <- function(x){
  f2 <- ((2+x^2)^0.5) * ((1-x)^(beta_factor - 1))
  return(f2)
}
func_f <- function(x){
  f1 <- func_f1(x)
  f2 <- func_f2(x)
  return(f1+f2)
}
integrate(func_f, lower = 0, upper = 1)$value
```

```
## [1] 0.5843471
```

```
const <- 1/integrate(func_f, lower = 0, upper = 1)$value

func_f_true <- function(x){
  value <- func_f(x) * const
  return(value)
}
#solve for the constant for f_pdf

#true value of f_pdf
x <- seq(from = 0, to = 1, by = 0.001)
f_value <- sapply(x,func_f_true)
f_sample_rejection <-c()
while(length(f_sample_rejection)<10000){
  random_weight <- runif(1)
```

```
  if(random_weight > weight_boundary){
    u <- runif(1)
    func_g <- func_g2
    sample_value <- uniroot(inverse_function,interval = c(0,1))$root
    sample_value_g <- func_g2(sample_value)
    sample_value_f <- func_f2(sample_value)
    ratio <- sample_value_f / (alpha_2 * sample_value_g)
  }
  else{
    u <- runif(1)
    func_g <- func_g1
    sample_value <- uniroot(inverse_function,interval = c(0,1))$root
    sample_value_g <- func_g1(sample_value)
    sample_value_f <- func_f1(sample_value)
    ratio <- sample_value_f / (alpha_1 * sample_value_g)
  }
  rejection_value <- runif(1)
  if(ratio >= rejection_value){
    f_sample_rejection <- c(f_sample_rejection, sample_value)
  }
}

sample_pdf <- density(f_sample_rejection)

#plot the true density and the sample density in one figure
plot(f_value~x,type = "l",col = "red")
lines(sample_pdf,type = "l", xlab = "x", ylab = "density", main = "using rejection method to get sample
legend(0.6,2.3,c("sample density", "true density of f"), col = c("black","red"),text.col = c("black","r
```




```
# str(f_sample_rejection)
# max(f_sample_rejection)
```