Stat5361-Homework 3

Xueying Li & Huiqiong Liao

2/25/2018

Question1

Derive the updating rules in the given algorithm based on the construction of an EM algorithm.

$$Q[\Psi|\Psi(k)] = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)}[log(\pi_i) + log(\phi(y_i - x_i^T\beta_j; 0, \sigma^2))]$$

$$\sum_{j=1}^{m}\pi_j = 1$$

$$L(\pi_1, ..., \pi_m, \lambda) = Q[\Psi|\Psi(k)] - \lambda(\sum_{j=1}^{m}\pi_j - 1) = 0$$

Calculate the first derivative of $L(\pi_1, ..., \pi_m, \lambda)$ respect to π_j and λ, for $j = 1, ..., m$:

$$L(\pi_1, ..., \pi_m, \lambda)'_{\pi_j} = 0; L(\pi_1, ..., \pi_m, \lambda)'_{\lambda} = 0$$

$$\sum_{i=1}^{n} p_{ij}^{(k+1)}(\frac{1}{\pi_j}) - \lambda = 0$$

Slove for π_j

$$\pi_j = \frac{\sum_{i=1}^{n} p_{ij}^{(k+1)}}{\lambda}$$

$$\sum_{j=1}^{m}\pi_j = \frac{\sum_{j=1}^{m}\sum_{i=1}^{n} p_{ij}^{(k+1)}}{\lambda} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)}}{\lambda} = \frac{n}{\lambda} = 1; \lambda = n$$

$$\pi_j = \frac{\sum_{i=1}^{n} p_{ij}^{(k+1)}}{n}$$

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

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

$$= \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)} log(\pi_j) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)} log(2\pi\sigma^2) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)}\frac{(y_i - x_i^T\beta_j)^2}{\sigma^2} = I_1 - \frac{1}{2}I_2 - \frac{1}{2}I_3$$

Where $I_1 = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)} log(\pi_j)$;$I_2 = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)} log(2\pi\sigma^2)$; and $I_3 = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)} \frac{(y_i - x_i^T\beta_j)^2}{\sigma^2}$

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

Which is a qudratic form. It is the sum of m quadratic forms each involving a single β_j for every certain j.

$$\sum_{i=1}^{n} p_{ij}^{(k+1)} (\frac{(y_i - x_i^T\beta_j)^2}{2\sigma^2}) = \sum_{i=1}^{n} p_{ij}^{(k+1)} [x_i^T (\frac{y_i}{x_i^T} - \beta_j)^2]$$

From the property of sample mean, β_j must be the mean of a weighted sample of $\frac{y_1}{x_1}, ..., \frac{y_n}{x_n^T}$. And for each element, it has the weight: $p_{ij} \cdot x_i \cdot x_i^T$. It can be represented in the following form:

$$\sum_{i=1}^{n} p_{ij}^{(k+1)} x_i x_i^T (\frac{y_i}{x_i^T} - \beta_j)^2$$

Solve for β_j:

$$\beta_j = \frac{\sum_{i=1}^{n} p_{ij}^{(k+1)} x_i x_i^T (\frac{y_i}{x_i^T})}{\sum_{i=1}^{n} p_{ij}^{(k+1)} x_i x_i^T} = \frac{\sum_{i=1}^{n} p_{ij}^{(k+1)} x_i y_i}{\sum_{i=1}^{n} p_{ij}^{(k+1)} x_i x_i^T}$$

Only I_2 and I_3 contain σ

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

Minimizing $I_2 + I_3$ to maximaze $Q[\Psi|\Psi(k)]$. Now we knew that β_j is the weighted mean of $\frac{y_i}{x_i^T}$, therefore, σ must be the sample variance of all data for every $j = 1, ..., m$.

$$\sigma^{2(k+1)} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)} (y_i - x_i^T\beta_j^{(k+1)})^2}{\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)}}$$

$$= \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)} (y_i - x_i^T\beta_j^{(k+1)})^2}{n}$$

Question2

(a)

$$g(x) \propto (2x^{\theta-1} + x^{\theta-\frac{1}{2}} e^{-x})$$

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

$$2C\gamma_{(\theta)} + C\gamma_{(\theta+\frac{1}{2})} = 1$$

$$C = \frac{1}{2\gamma_{(\theta)} + C\gamma_{(\theta+\frac{1}{2})}}$$

$$\therefore g(x) = \frac{2\gamma_{(\theta)}}{2\gamma_{(\theta)} + \gamma_{(\theta+\frac{1}{2})}} \frac{1}{\gamma_{(\theta)}} x^{\theta-1} e^{-x} + \frac{\gamma_{(\theta+\frac{1}{2})}}{2\gamma_{(\theta)} + \gamma_{(\theta+\frac{1}{2})}} * \frac{1}{\gamma_{(\theta+\frac{1}{2})}} x^{\theta-\frac{1}{2}} e^{-x}$$

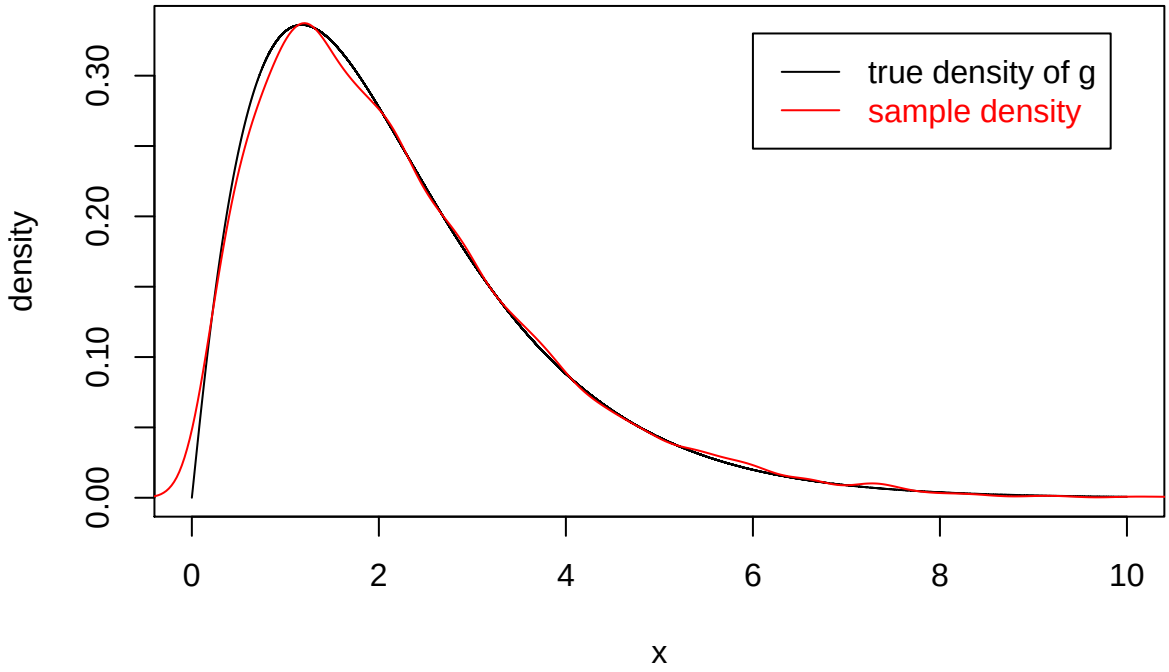
It is the mixture of $\gamma(\theta, 1)$ and $\gamma(\theta + \frac{1}{2}, 1)$, where the weights are $\frac{2\gamma_{(\theta)}}{2\gamma_{(\theta)}+\gamma_{(\theta+\frac{1}{2})}}$ and $\frac{\gamma_{(\theta+\frac{1}{2})}}{2\gamma_{(\theta)}+\gamma_{(\theta+\frac{1}{2})}}$

(b)

Repeat the following step 1 and step 2 for 10,000 times:
Step 1: $u \sim unif(0, 1)$
Step 2: find x, where $Cg(x) = u$
$sample_g \leftarrow c(sample_g, x)$

```
#you can change the value of theta, we set it equals to 2 in order to calculate conveniently
theta <- 2
C <- 1 / (2 * gamma(theta) + gamma(theta + 1/2))

#pdf of g(x)
func_g <- function(x){
  return(C * (2 * x ** (theta - 1) + x ** (theta - 1/2)) * exp(-x))
}

#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)
  #return()
}

#generate 10000 samples
sample_g <- c()
for(i in 1:10000){
  u <- runif(1)
  inverse_method <- uniroot(inverse_function,interval = c(0,20))$root
  sample_g <- c(sample_g,inverse_method)
}

sampleg_pdf <- density(sample_g)

#calculate the true value of g_pdf
x <- seq(from = 0, to = 10, by = 0.0001)
g_value <- sapply(x,func_g)

##--------------Q2-b----------------------
```

```r
#plot the true density and the sample density in one figure
plot(g_value~x,type = "l", xlab = "x", ylab = "density")
lines(sampleg_pdf, col = "red")
#add legend the fist two parameters denote the location, lty =1 denotes the type of legend is line
legend(6,0.33,c("true density of g", "sample density"), col = c("black","red"),text.col = c("black","red
```



(c)

$\sqrt{4+x} < 2+\sqrt{x}$ for all $x > 0$ because $(2+\sqrt{x})^2 - (\sqrt{4+x})^2 = 4 \cdot \sqrt{x} > 0$.
Therefore, $\dfrac{f(x)}{g(x)} \leq 1$, we choose $\alpha = \frac{1}{C}$.

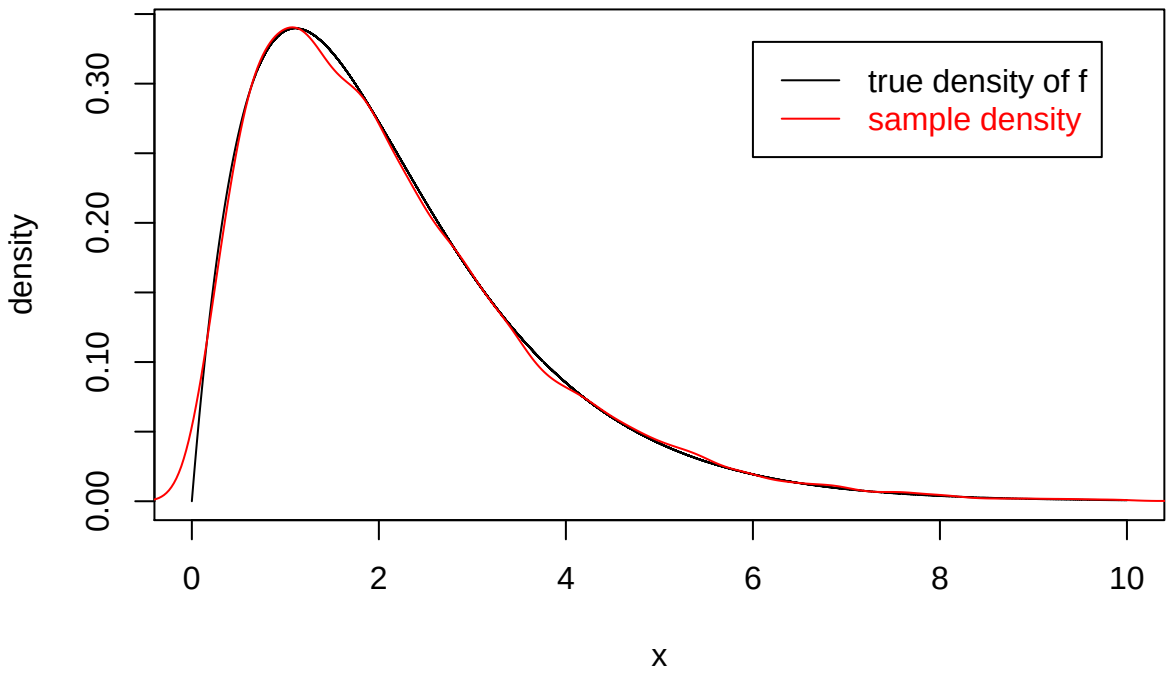
Step 1: $u \sim unif(0,1)$
Step 2: find x^*, where $Cg(x) = u$
Calculate $f(x^*)$ and $g(x^*)$
$u^* \sim unif(0,1)$
if $u^* < \dfrac{f(x^*)}{g(x^*)}$
$sample \leftarrow c(sample, x^*)$
Repeat Step 1 and Step 2 until the length of sample is 10,000

```r
#you can change the value of theta, we set it equals to 2 in order to calculate conveniently
theta <- 2
C <- 1 / (2 * gamma(theta) + gamma(theta + 1/2))

#pdf of g(x)
```

```r
func_g <- function(x){
  return(C * (2 * x ** (theta - 1) + x ** (theta - 1/2)) * exp(-x))
}

#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)
  #return()
}
##--------------Q2-c----------------------
```

```r
#pdf of f
func_f <- function(x){
  return(((4 + x) ** 0.5) * (x ** (theta - 1)) * exp(- x))
}

const <- 1 / integrate(func_f,lower = 0, upper = Inf)$value
func_f_true <- function(x){
  value <- func_f(x) * const
  return(value)
}
#calculate the true value of f_pdf
x <- seq(from = 0, to = 10, by = 0.0001)
f_value <- sapply(x,func_f_true)

f_sample_rejection <-c()
while(length(f_sample_rejection)<10000){
  u <- runif(1)
  sample_value <- uniroot(inverse_function,interval = c(0,20))$root
  sample_value_g <- func_g(sample_value) / C
  sample_value_f <- func_f(sample_value)
  ratio <- sample_value_f / sample_value_g
  rejection_value <- runif(1)
  if(ratio >= rejection_value){
    f_sample_rejection <- c(f_sample_rejection, sample_value)
  }
}

sampleg_pdf <- density(f_sample_rejection)
##--------------Q2-c----------------------
```

```r
#plot the true density and the sample density in one figure
plot(f_value~x,type = "l", xlab = "x", ylab = "density")
lines(sampleg_pdf, col = "red")
#add legend the fist two parameters denote the location, lty =1 denotes the type of legend is line
legend(6,0.33,c("true density of f", "sample density"), col = c("black","red"),text.col = c("black","red
```



Question 3

(a)

For

$$0 < x < 1$$

,

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

$$g_1^*(x) = x^{\theta-1}$$

$$\int_0^1 g_1^*(x)dx = \frac{x^\theta}{\theta}|_0^1 = \frac{1}{\theta} = \frac{1}{C_1} \Rightarrow g_1(x) = \theta \cdot x^{\theta-1}$$

$$g_2^*(x) = \sqrt{3} \cdot (1-x)^{\beta-1}$$

$$\int_0^1 g_2^*(x)dx = -\sqrt{3} \cdot \frac{(1-x)^\beta}{\beta}|_0^1 = \frac{\sqrt{3}}{\beta} = \frac{1}{C_2} \Rightarrow g_2(x) = \beta \cdot (1-x)^{\beta-1}$$

$$g(x) = \frac{1}{2}g_1(x) + \frac{1}{2}g_2(x)$$

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

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

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

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

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

$$\frac{2}{\beta} \leq \alpha$$

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

Step 1: u-weight $\sim unif(0,1)$
Step 2: if u-weight $\leq w_1$
$u \sim unif(0,1)$, find x^*, where $g_1(x) = u$
elase,
$u \sim unif(0,1)$, find x^*, where $g_2(x) = u$
Step 3: calculate $f(x^*)$ and $g(x^*)$
$u^* \sim unif(0,1)$
if $u^* \leq \frac{f(x^*)}{g(x^*)}$
$sample \leftarrow c(sample, x^*)$
Repeat Step 1,2 and 3 until the length of sample is 10,000

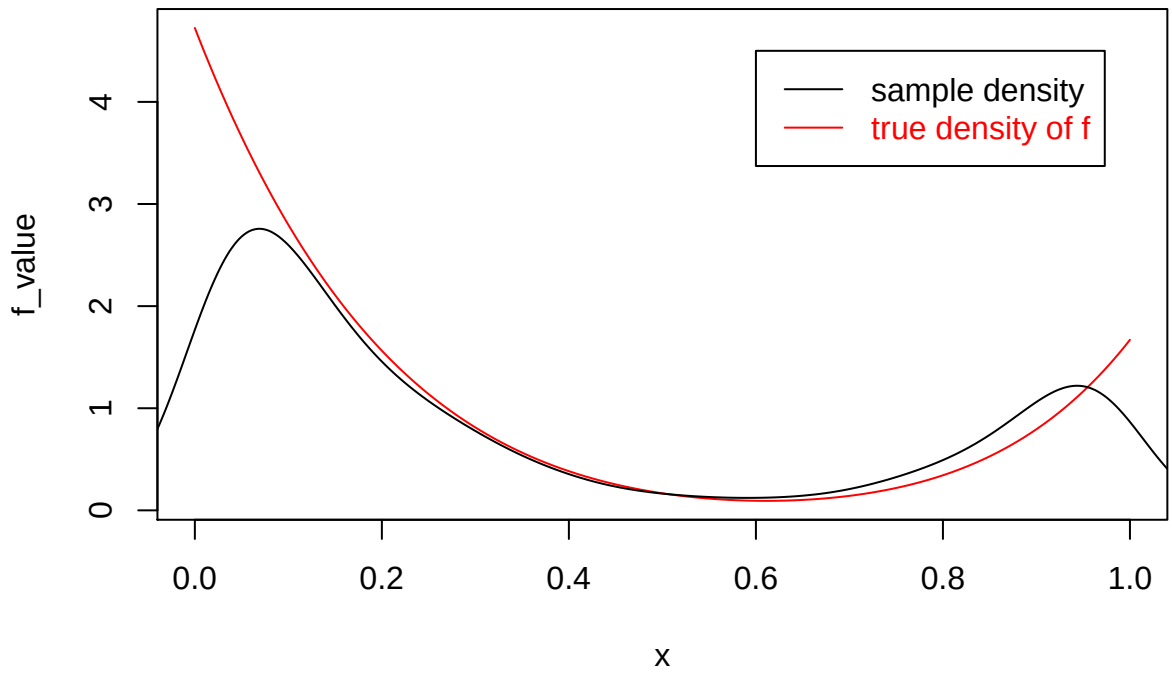
```
#you can change the value of theta and beta
theta_factor <- 9
beta_factor <- 6
# W1 <- (C1^2) / (C1 + C2)#weight for g1(x)
# W2 <- (C2^2) / (C1 + C2)#weight for g2(x)
W1 <- 1/2#weight for g1(x)
W2 <- 1/2#weight for g2(x)
alpha_factor <- max(2/theta_factor, 2/beta_factor)

#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)
}
##--------------Q3-a-----------------------
#pdf of f
func_f <- function(x){
  f1 <- (x ^ (theta_factor-1)) / (1+x^2)
  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] 0.2993621
```

```
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)

$$f(x_1) = \frac{x^{\theta-1}}{1+x^2} \leq x^{\theta-1} = \alpha_1 \cdot g_1(x)$$

$$f(x_2) = \sqrt{2+x^2} \cdot (1-x)^{\beta-1} \leq \sqrt{3} \cdot (1-x)^{\beta-1} = \alpha_2 \cdot g_2(x)$$

Then, $\alpha_1 = \frac{1}{\theta}$, and $\alpha_2 = \frac{\sqrt{3}}{\beta}$

Thus, when $runif > \frac{\frac{1}{\theta}}{\frac{1}{\theta} + \frac{\sqrt{3}}{\beta}}$, sample from $g_2(x)$

Step 1: u-weight $\sim unif(0,1)$

Step 2: if u-weight $>$ weighted-boundary
$u \sim unif(0,1)$, find x^*, where $g_2(x^*) = u$
calculate ratio $= \frac{f_2(x^*)}{\alpha g_2(x^*)}$
else,
$u \sim unif(0,1)$, find x^*, where $g_1(x^*) = u$
calculate ratio $= \frac{f_1(x^*)}{\alpha g_1(x^*)}$

Step 3: $u^* \sim unif(0,1)$
if $u^* < ratio$
$sample \leftarrow c(sample, x^*)$

Repeat Step 1, 2 and 3 until the length of sample is 10,000

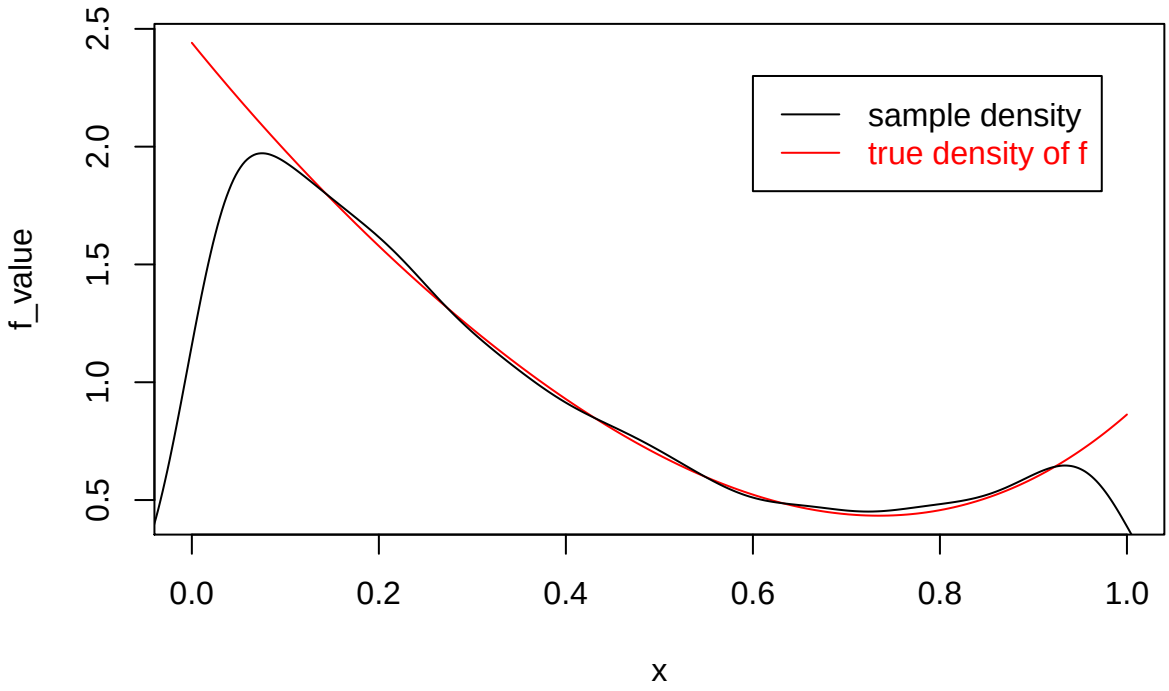
```
#you can change the value of theta and beta
theta_factor <- 6
beta_factor <- 3
alpha_1 <- 1 / theta_factor
alpha_2 <- (3^0.5) / beta_factor

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] 0.5793803
```

```
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)
```