STAT243-PS8

Jinhui Xu

Dec. 2017

I discuss with Shan Gao, Junyuan Gao, Ming Qiu, Xiao Li

1 Question 1

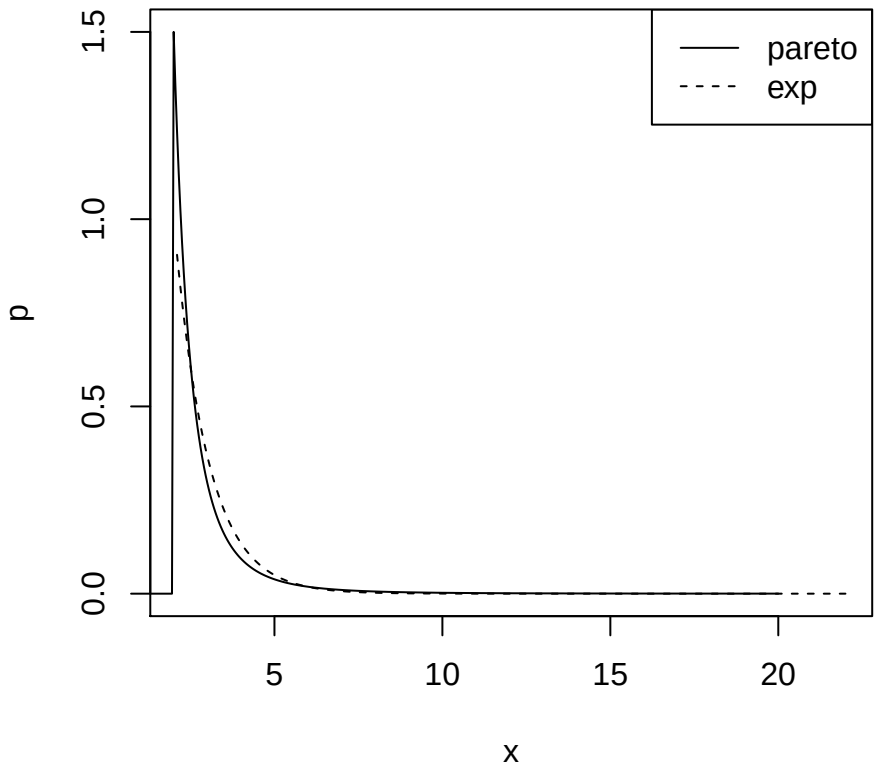
1.1 a)

$$p(x) = \frac{\beta\alpha^{\beta}}{x^{\beta+1}}$$

$$f(x) = \lambda e^{-\lambda x}$$

In this way $\frac{p(x)}{f(x)} = \frac{\beta\alpha^{\beta}}{\lambda}\frac{e^{\lambda x}}{x^{\beta+1}}$. According to L'Hospital's rule, we know $\lim_{x\to\infty}\frac{p(x)}{f(x)} = \infty$. It shows that $p(x)$ has heavier tail, so the pareto decay more slowly.

```
###set parameters
beta<-3;alpha<-2;lambda<-1

###sample from two distributions
x<-seq(0.1,20,0.05)
pareto<-dpareto(x,alpha,beta)
exp<-dexp(x,lambda)

###plot two distributions
plot(x+2,exp,type='l',lty=2,ylim=range(0,1.5),xlab='x',ylab='p')
lines(x,pareto)
legend('topright',lty=c(1,2),legend=c('pareto','exp'))
```



1.2 b)

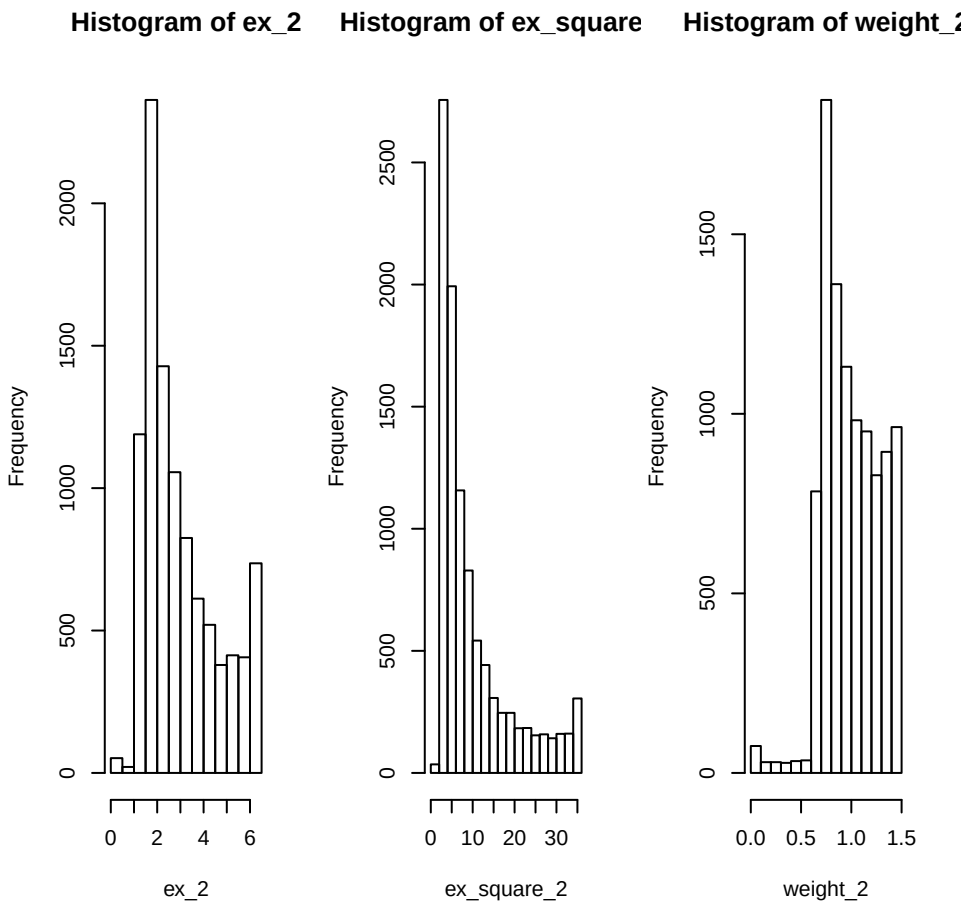
Firstly, do some basic setting and basic calculation

```
####basic setting
beta<-3;alpha<-2;lambda<-1;m<-10000

####get sample from pareto distribution
sample_pareto<-rpareto(m,scale=alpha,shape=beta)

####calcualte the expection of x and square of x
ex_2<-sample_pareto^5*exp(-sample_pareto+2)/24
ex_square_2<-sample_pareto^6*exp(-sample_pareto+2)/24

####calculate the weight
weight_2<-sample_pareto^4*exp(-sample_pareto+2)/24
```

Then show the required expectation and give histogram plots

```
####calculate the expectation,variance of x and square of x
mean(ex_2);mean(ex_square_2)

## [1] 2.998046
## [1] 10.05176
```

```r
var(ex_2);var(ex_square_2)

## [1] 2.406171
## [1] 77.09884

##########plot histogram
par(mfrow=c(1,3))
hist(ex_2);hist(ex_square_2);hist(weight_2)
```



From the plot, we can find that $h(x)f(x)/g(x)$ and $f(x)/g(x)$ is distributed relatively even. And there is no extreme value. It is beacuse $g(x)$ has heavier tail than $f(x)$.

1.3 c)

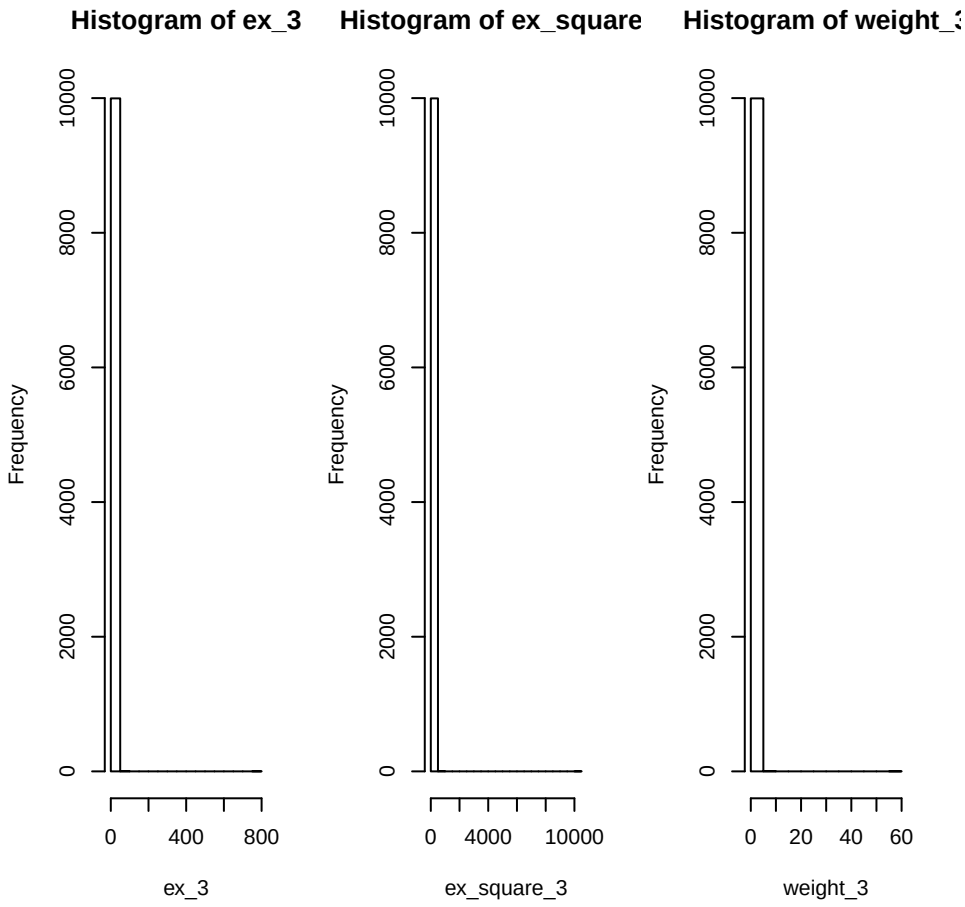
Do some basic setting and basic calculation the same as section_b

```r
####sample from exponential distribution
sample_exp<-rexp(m)+2

####calcualte the expection of x and square of x
ex_3<-24*sample_exp^(-3)*exp(sample_exp-2)
ex_square_3<-24*sample_exp^(-2)*exp(sample_exp-2)

####calculate the weight
```

```
weight_3<-24*sample_exp^(-4)*exp(sample_exp-2)
```

show the result.

```
####calculate the expectation,variance of x and square of x
mean(ex_3);mean(ex_square_3)

## [1] 2.943074
## [1] 10.48022

var(ex_3);var(ex_square_3)

## [1] 60.22603
## [1] 10295.85

##########plot histogram
par(mfrow=c(1,3))
hist(ex_3);hist(ex_square_3);hist(weight_3)
```



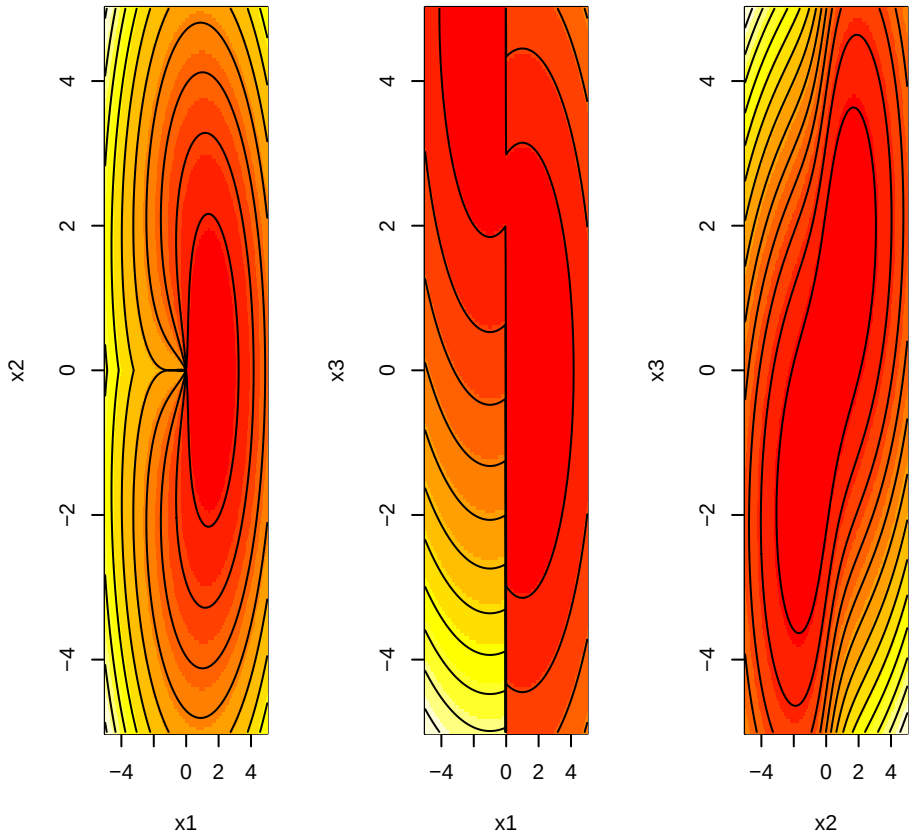
Each plot shows that there are several extreme values which mean that the variance of estimators would be relatively large. It is beacuse $f(x)$ has heavier tail than $g(x)$.

2 Question2

```r
theta <- function(x1,x2) atan2(x2, x1)/(2*pi)

f <- function(x) {
  f1 <- 10*(x[3] - 10*theta(x[1],x[2]))
  f2 <- 10*(sqrt(x[1]^2 + x[2]^2) - 1)
  f3 <- x[3]
  return(f1^2 + f2^2 + f3^2)
}
```

```r
####fix x3 and calculate the value of f(c(x1,x2,0))

###set x from -5 to 5 and get 201 values
x1<-seq(-5,5,0.05)
x2<-seq(-5,5,0.05)
x3<-seq(-5,5,0.05)
f_value12<-matrix(nrow=201,ncol=201)
for(i in 1:201){
  for(j in 1:201){
    f_value12[i,j]<-f(c(x1[i],x2[j],0))
  }
}
####fix x2 as 0
f_value13<-matrix(nrow=201,ncol=201)
for(i in 1:201){
  for(j in 1:201){
    f_value13[i,j]<-f(c(x1[i],0,x3[j]))
  }
}

####fix x1 as 1
f_value23<-matrix(nrow=201,ncol=201)
for(i in 1:201){
  for(j in 1:201){
    f_value23[i,j]<-f(c(1,x2[i],x3[j]))
  }
}
```

```r
########compare the plot
par(mfrow=c(1,3))
image(x1,x2,f_value12)
contour(x1,x2,f_value12,add=TRUE,drawlabels = FALSE)
image(x1,x3,f_value13)
contour(x1,x3,f_value13,add=TRUE,drawlabels = FALSE)
image(x2,x3,f_value23)
contour(x2,x3,f_value23,add=TRUE,drawlabels = FALSE)
```




```
############calculate the optimum of f function
start<-c(100,-10,-10)
op<-optim(par=start,fn=f)
nl<-nlm(f=f,p=start)
op$par;op$value;nl$estimate;nl$minimum

## [1]  1.000634580 -0.001903183 -0.001726437
## [1] 0.0002126484
## [1]  1.000000e+00 -2.142877e-10  3.478953e-10
## [1] 7.203601e-17

start<-c(20,100,1)
op<-optim(par=start,fn=f)
op$par;op$value

## [1]  0.9884603 -0.1617380 -0.2637370
## [1] 0.07295685

start<-c(1,1,1)
op<-optim(par=start,fn=f)
op$par;op$value

## [1]  0.9999779414 -0.0001349269 -0.0001927127
## [1] 1.343098e-07
```

```
start<-c(1,0,0)
op<-optim(par=start,fn=f)
nl<-nlm(f=f,p=start)
op$par;op$value;nl$estimate;nl$minimum

## [1] 1 0 0
## [1] 0
## [1] 1 0 0
## [1] 0
```

From the result, we find all optimum is around 0, and the coordinates are around $(1, 0, 0)$. And when we set the start as $(1, 0, 0)$, the optimum equals to 0. And explore the function f, it returns the sum of three squares, so 0 is the global optimum. And I find nlm get better results.

3 Question3

3.1 a)

Algorithm:

step1: first write the $Q(\theta|\theta_t)$

$$Q(\theta|\theta_t) = E(logL(\theta|Y)|x, \theta_t)$$
$$= -\frac{n}{2}log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=c+1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2 - \frac{1}{2\sigma^2}E([\sum_{i=1}^{c}Z_i^2 - 2(\beta_0 + \beta_1 x_i)\sum_{i=1}^{c}Z_i + \sum_{i=1}^{c}(\beta_0 + \beta_1 x_i)^2]|(\theta_t, x))$$

step2: we want to maximize the $Q(\theta|\theta_t)$, which means minimizes $-Q(\theta|\theta_t)$;let $A_i = E(Z_i^2|(\theta_t, x)), B_i = E(Z_i|\theta_t, x)$, A and B are both constants given θ_t, x.

$$-Q(\theta|\theta_t) = -E(logL(\theta|Y)|x, \theta_t)$$
$$= \frac{1}{2\sigma^2}(n\sigma^2 log(2\pi\sigma^2) + \sum_{i=c+1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2 + [\sum_{i=1}^{c}A_i - 2\sum_{i=1}^{c}(\beta_0 + \beta_1 x_i)B_i + \sum_{i=1}^{c}(\beta_0 + \beta_1 x_i)^2])$$
$$= \frac{1}{2\sigma^2}(n\sigma^2 log(2\pi\sigma^2) + \sum_{i=c+1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2 + \sum_{i=1}^{c}(B_i - \beta_0 - \beta_1 x_i)^2 - \sum_{i=1}^{c}B_i^2 + \sum_{i=1}^{c}A_i)$$

step3:As A,B,c are all constants, if we want to minimize $-Q(\theta|\theta_t)$, we first should minimize $\sum_{i=c+1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2 + \sum_{i=1}^{c}(B_i - \beta_0 - \beta_1 x_i)^2$. It is exactly the RSS of regression of (y, x), where the first c elements in y_i equals to B_i, and the latter are y_i. In this way, we can get the $\hat{\beta_0}, \hat{\beta_1}$ from linear regression

After get the RSS, we then do derivation to σ^2 and get $\hat{\sigma}^2$

step4:let $\theta_{t+1} = (\hat{\beta_0}, \hat{\beta_1}, \hat{\sigma}^2)$ and return to step2, do iteration until $||\theta_t - \theta_{t+1}|| < m$, m=0.00001 is a small constant we set to stop the iteration, or if the iteration times over 1000, I also stop the loop.

3.2 b)

As we only have n-c complete data (x, y), use lm function to $(y\ x)$ and get the coefficients of the linear regression. Then I take β_0 as the first coefficient and take β_1 as the second coefficient, also take σ_0^2 as RSS/n.

3.3 c)

```
######## write ez function to get the expectation of Z and Z^2;
######## which in my algorithm in (a) equals to Bi and Ai
ez<-function(mu,sigma,k){
  result<-c()

  ###calculate the Ez and Ez^2
  k_star<-(k-mu)/sigma
  r_kstar<-dnorm(k_star)/(1-pnorm(k_star))

  ###store the value in a vector
  result[1]<-mu+sigma*r_kstar
  result[2]<-result[1]^2+sigma^2*(1+k_star*r_kstar-r_kstar^2)
  return(result)
}

###write the function to get censored y according to the given proportion
censored_y<-function(x,y,proportion,beta){
  b<-rep(0,100);a<-rep(0,100)
  tau<-sort(yComplete)[n-n*proportion+1]

  ###find the y needed to be censored and then calaulate the corresponding Bi and Ai
  for(i in 1:n){
    if (y[i]>=tau) {
      y[i]<-ez(beta[1]+beta[2]*x[i],sqrt(beta[3]),tau)[1]
      b[i]<-y[i]
      a[i]<-ez(beta[1]+beta[2]*x[i],sqrt(beta[3]),tau)[2]
    }
  }
  ###store the result in a matrix
  data<-cbind(y,a,b)
  return(data)
}

####write the final function to get estimated theta
em<-function(proportion){

  ###get x and y
  set.seed(1)
  n <- 100; beta0 <- 1; beta1 <- 2; sigma2 <- 6
  x <- runif(n)
  b<-rep(0,100);a<-rep(0,100)
  yComplete <- rnorm(n, beta0 + beta1*x, sqrt(sigma2))

  ###initiate some settings
  itera<-0;beta_0<-c();beta_1<-c();error<-100

  ########to calculate the initial beta, first calculate the complete x and y
  tau<-sort(yComplete)[n-n*proportion+1]
  y0<-c();x0<-c()
  y0<-yComplete[which(yComplete<tau)]
  x0<-x[which(yComplete<tau)]
```

```
  ######calculate the beta_0
  lm0<-lm(y0~x0)
  beta_0[1]<-summary(lm0)$coef[1]
  beta_0[2]<-summary(lm0)$coef[2]
  beta_0[3]<-sum(lm0$residuals^2)/length(y0)

  ###do iteration until the number of iteration is over 1000 times or the error is smaller than 0.00001
  while(error>=0.00001&itera<=1000){
    itera<-itera+1
    yab<-censored_y(x,yComplete,proportion,beta_0)
    y<-yab[,1]
    a<-yab[,2]
    b<-yab[,3]
    lm<-lm(y~x)

    ###get a new beta
    beta_1[1]<-summary(lm)$coef[1]
    beta_1[2]<-summary(lm)$coef[2]
    beta_1[3]<-(sum(lm$residuals^2)-crossprod(b)+sum(a))/n

    ####calculate the error between new beta and old beta
    error<-crossprod(beta_0-beta_1)
    beta_0<-beta_1
  }
  return(c(beta_0,error,itera))
}
em(0.2);em(0.8)

## [1] 4.580421e-01 2.832377e+00 4.654111e+00 2.669040e-06 7.000000e+00
## [1] 3.290495e-01 2.899787e+00 3.899489e+00 9.252086e-06 5.800000e+01
```

The first three values are estimated parameters. From the result, we can find that the estimated parameters converge to the true value. It is also obvious that when we lose 80 percent data, the precise of estimation decreases.

3.4 d)

```
###function to calculate the log likelihood value
f<-function(input,proportion=0.2){
  beta00<-input[1];beta01<-input[2];sigma_square<-input[3]

  set.seed(1)
  n <- 100; beta0 <- 1; beta1 <- 2; sigma2 <- 6
  x <- runif(n); yComplete <- rnorm(n, beta0 + beta1*x, sqrt(sigma2))

  ###get two parts of data(x,y)
  tau<-sort(yComplete)[n-n*proportion+1]
  y_censored<-yComplete[which(yComplete>=tau)]
  x_censored<-x[which(yComplete>=tau)]
  y_complete<-yComplete[which(yComplete<tau)]
  x_complete<-x[which(yComplete<tau)]

  ###calculate the log likelihood value of censored data and complete data
```

```
  l_censored<-sum(log(1-pnorm((tau-beta00-beta01*x_censored)/sqrt(sigma_square))))
  l_complete=-n*(1-proportion)/2*log(2*pi*sigma_square)-1/(2*sigma_square)*
  crossprod(y_complete-beta00-beta01*x_complete)

  ###return the negative of sum of two log likelihood value
  return(-(l_censored+l_complete))

}

###get the result and compare with my EM algorithm
op<-optim(par=c(1,1,1),fn=f,lower=c(0,0,0),method = 'BFGS')
op$par;op$counts
```

```
## [1] 0.4580453 2.8325579 4.6548216
## function gradient
##       15       15
```

```
em(0.2)[1:3];em(0.2)[5]
```

```
## [1] 0.4580421 2.8323768 4.6541108
## [1] 7
```

```
############################################################
####try the situation of proportion equals to 0.8
f<-function(input,proportion=0.8){
  beta00<-input[1];beta01<-input[2];sigma_square<-input[3]

  set.seed(1)
  n <- 100; beta0 <- 1; beta1 <- 2; sigma2 <- 6
  x <- runif(n); yComplete <- rnorm(n, beta0 + beta1*x, sqrt(sigma2))

  ###get two parts of data(x,y)
  tau<-sort(yComplete)[n-n*proportion+1]
  y_censored<-yComplete[which(yComplete>=tau)]
  x_censored<-x[which(yComplete>=tau)]
  y_complete<-yComplete[which(yComplete<tau)]
  x_complete<-x[which(yComplete<tau)]

  ###calculate the log likelihood value of censored data and complete data
  l_censored<-sum(log(1-pnorm((tau-beta00-beta01*x_censored)/sqrt(sigma_square))))
  l_complete=-n*(1-proportion)/2*log(2*pi*sigma_square)-1/(2*sigma_square)*
  crossprod(y_complete-beta00-beta01*x_complete)

  ###return the negative of sum of two log likelihood value
  return(-(l_censored+l_complete))

}

###get the result and compare with my EM algorithm
op<-optim(par=c(1,1,1),fn=f,lower=c(0,0,0),method = 'BFGS')
op$par;op$counts
```

```
## [1] 0.3332969 2.9135182 3.9322390
## function gradient
##       14       14
```

```r
em(0.8)[1:3];em(0.8)[5]
```

```
## [1] 0.3290495 2.8997866 3.8994888
## [1] 58
```

I set proportion$= 0.2, 0.8$ and compare two algorithms. We can see that the results are both similar. But when proportion equals to 0.2, my EM algorithm costs fewer iterations while when proportion equals to 0.8, my EM algorithm costs much more iterations.