STAT 243 HW8

Lei Zhang

December 2, 2017

1 Problem 1

1.1 Problem a

$$\begin{aligned}\int_x^{+\infty} p(t)dt &= \beta\alpha^\beta \int_x^{+\infty} t^{-\beta-1}dt \\ &= = \beta\alpha^\beta \frac{1}{-\beta} t^{-\beta} \mid_x^{+\infty} \\ &= \frac{\alpha^\beta}{x^\beta}\end{aligned}$$

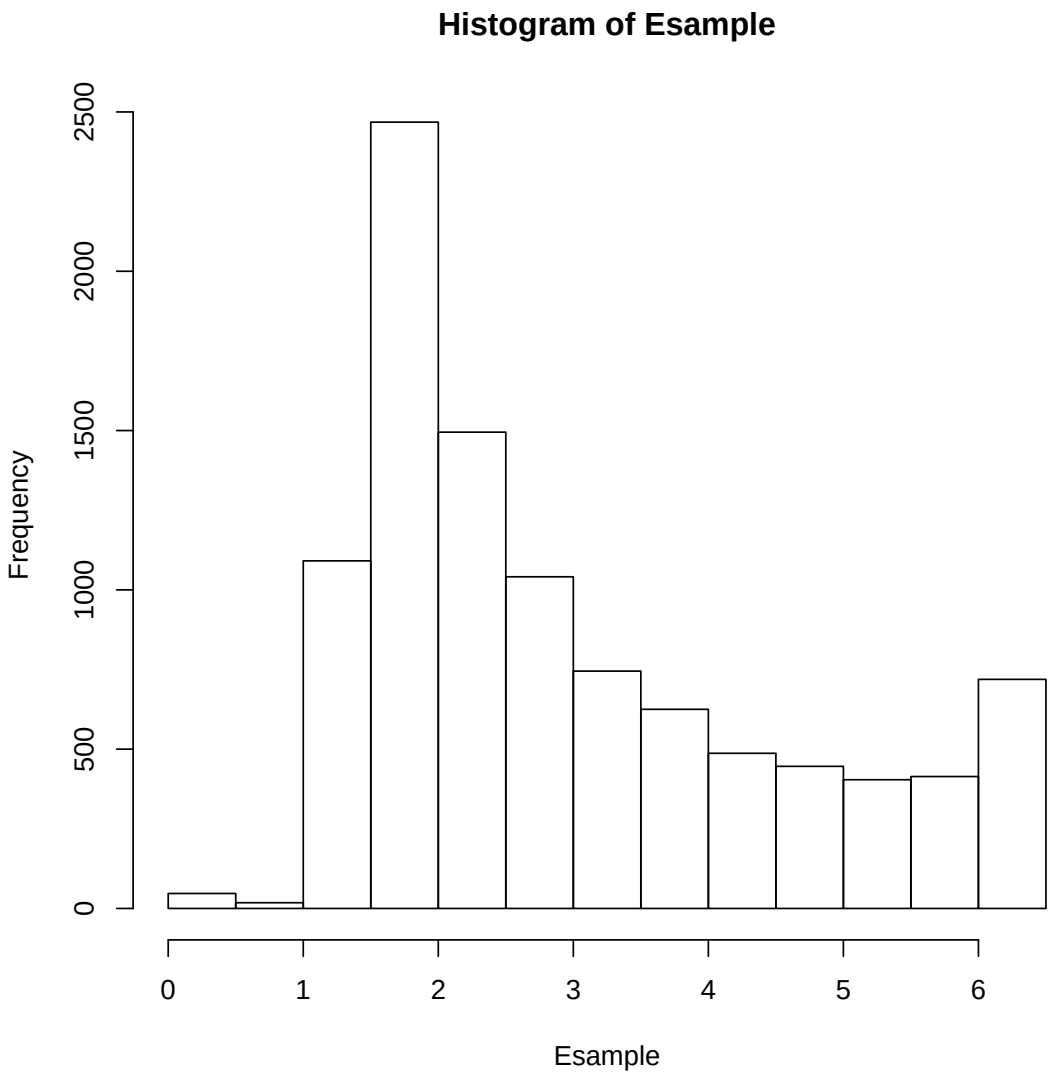
$$\begin{aligned}\int_x^{+\infty} f(t)dt &= \int_x^{+\infty} \lambda e^{-\lambda} t dt \\ &= \frac{1}{\lambda}\lambda e^{-\lambda t} \mid_x^{+\infty} \\ &= e^{-\lambda x}\end{aligned}$$

We know that exponential grows faster than polynomial, so when we take the reciprocal, the tail of the Pareto decay more slowly than that of an exponential distribution.

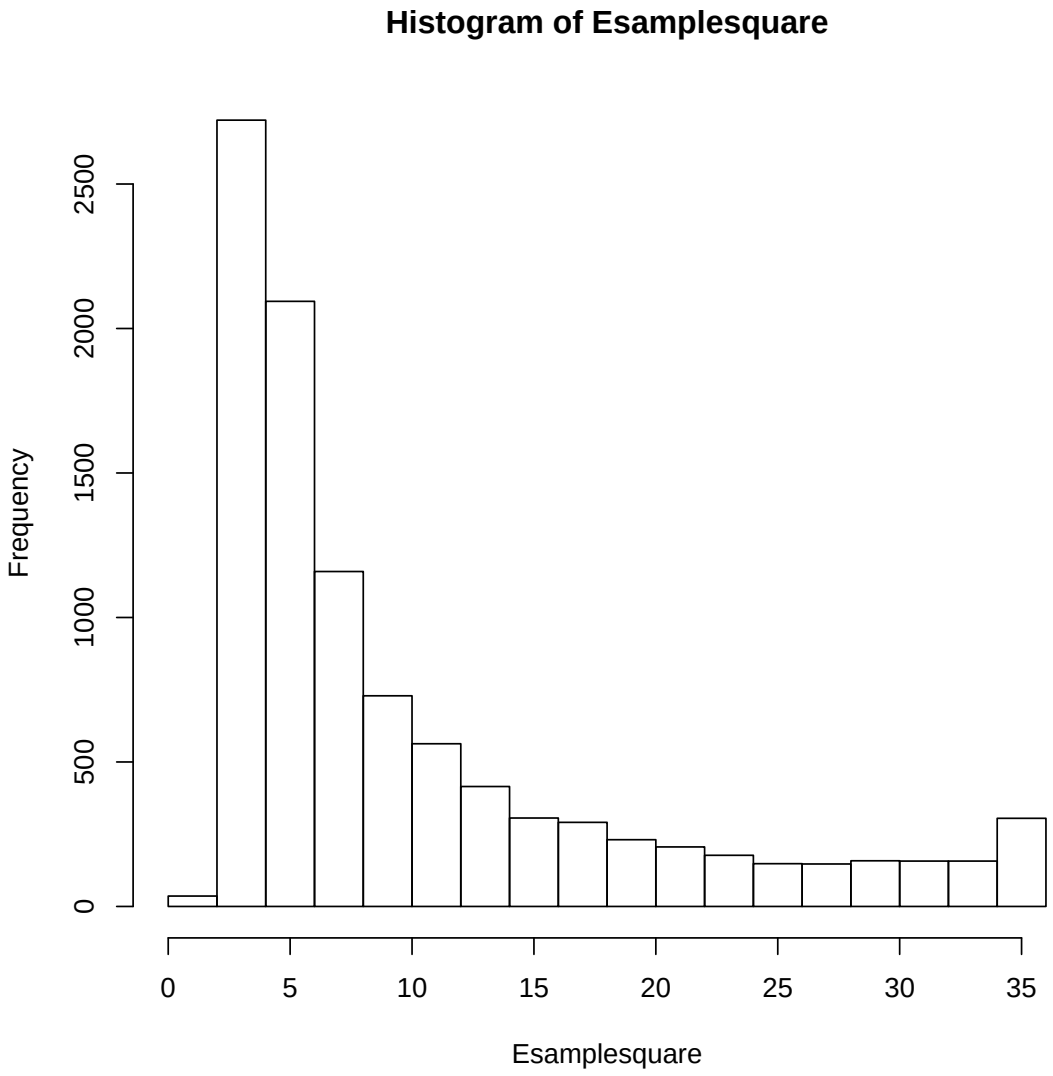
1.2 Problem b

```
library(VGAM)

## Loading required package:  stats4
## Loading required package:  splines
```

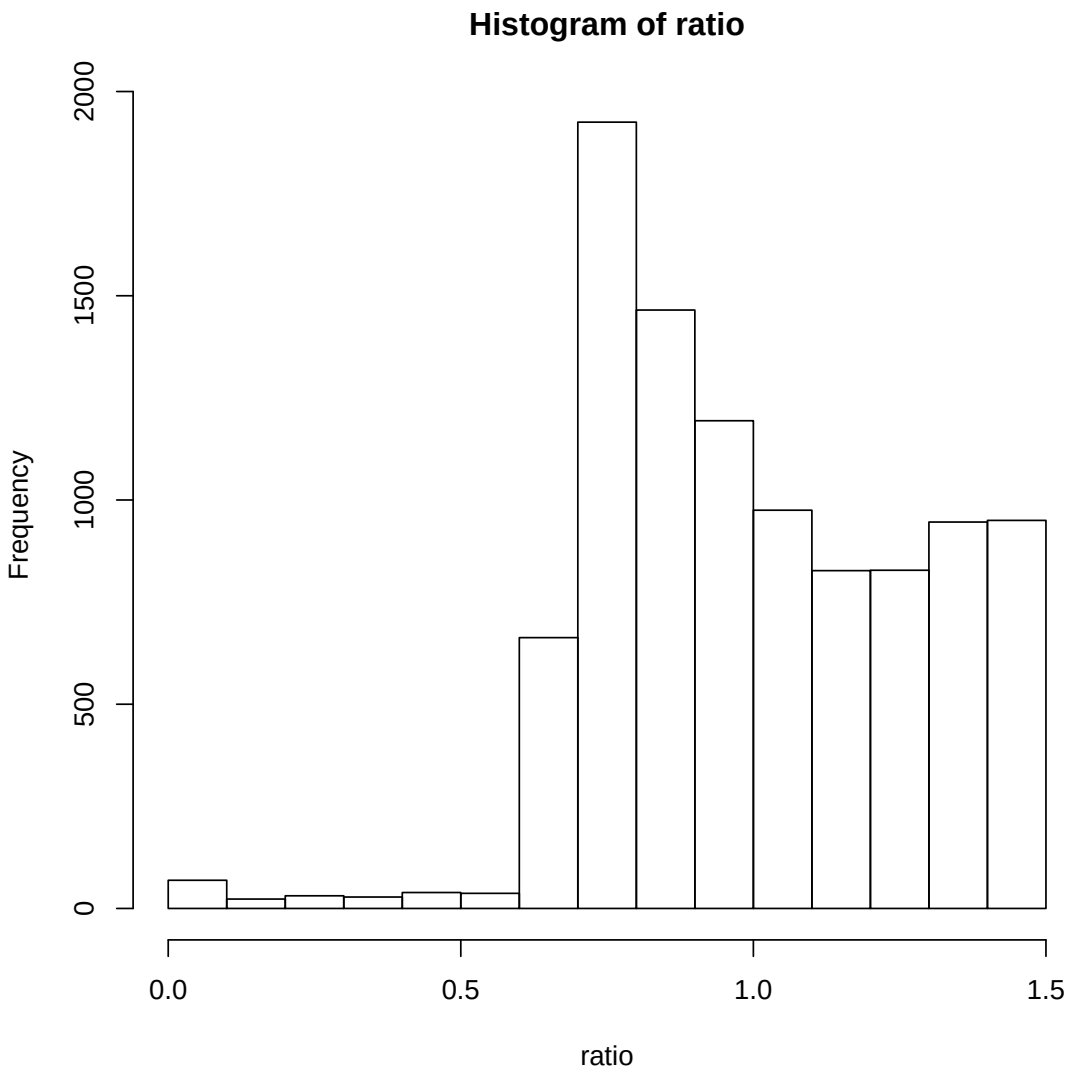
```r
m <- 10000
alpha <- 2
beta <- 3

#sample from Pareto distribution
sample <- rpareto(m, scale = alpha, shape = beta)
samplesquare <- sample^2
fsample <- dexp(sample - alpha, rate = 1)
gsample <- dpareto(sample, scale = alpha, shape = beta)

#Estimate the mean of X and X^2, f/g
Esample <- sample*fsample/gsample
Esamplesquare <- samplesquare*fsample/gsample
ratio <- fsample/gsample

#draw histogram
hist(Esample)
```




```
hist(Esamplesquare)
```


Histogram of Esamplesquare
Frequency
0
500
1000
1500
2000
2500
0
5
10
15
20
25
30
35
Esamplesquare

```
hist(ratio)
```

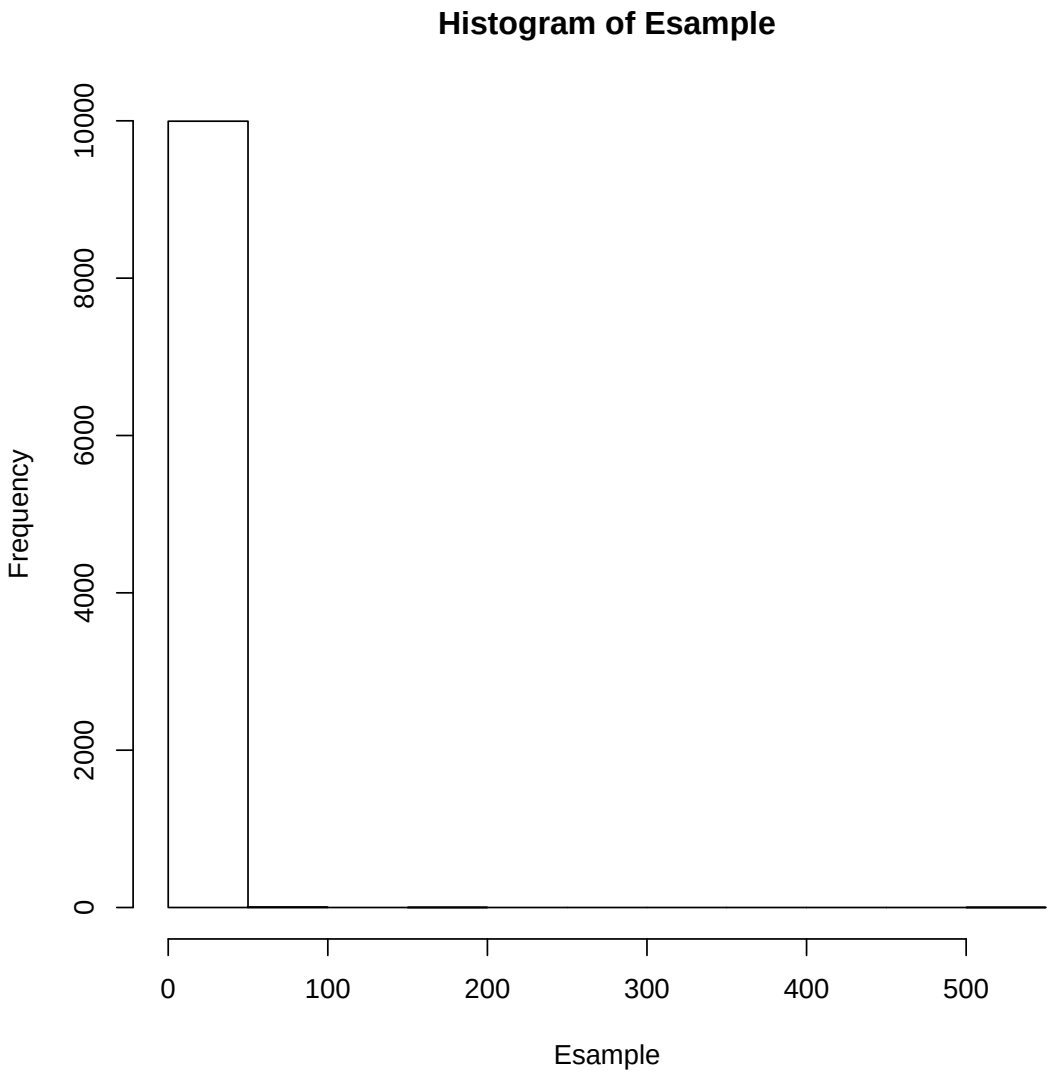


There are no extreme weights that would have a very strong influence on $\hat{\mu}$

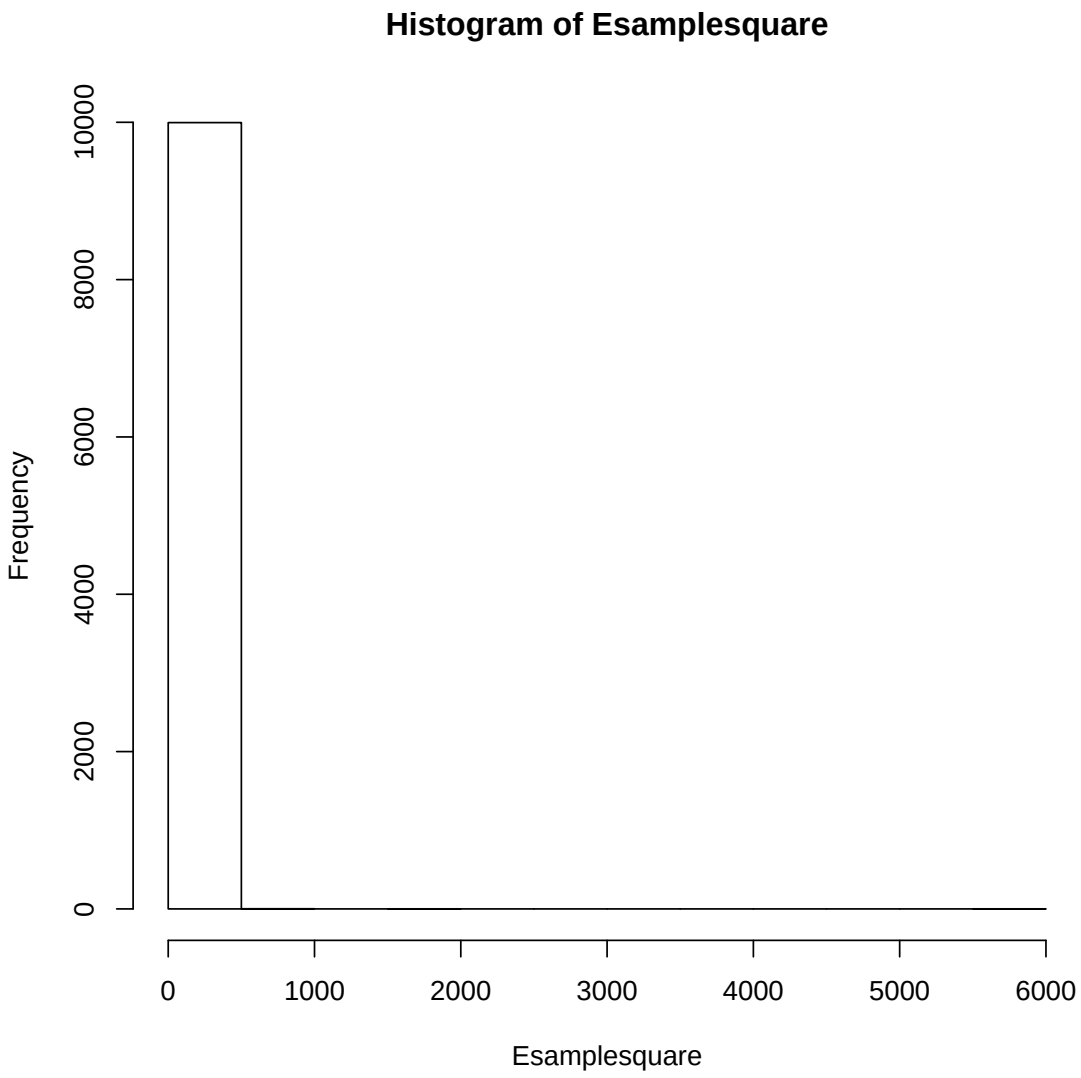
1.3 Problem c

```
#sample from exponential distribution(rate=1)
sample <- rexp(m, rate = 1)
samplesquare <- sample^2
fsample <- dpareto(sample + alpha, scale = alpha, shape = beta)
gsample <- dexp(sample, rate = 1)

#Estimate the mean of X and X^2, f/g
```

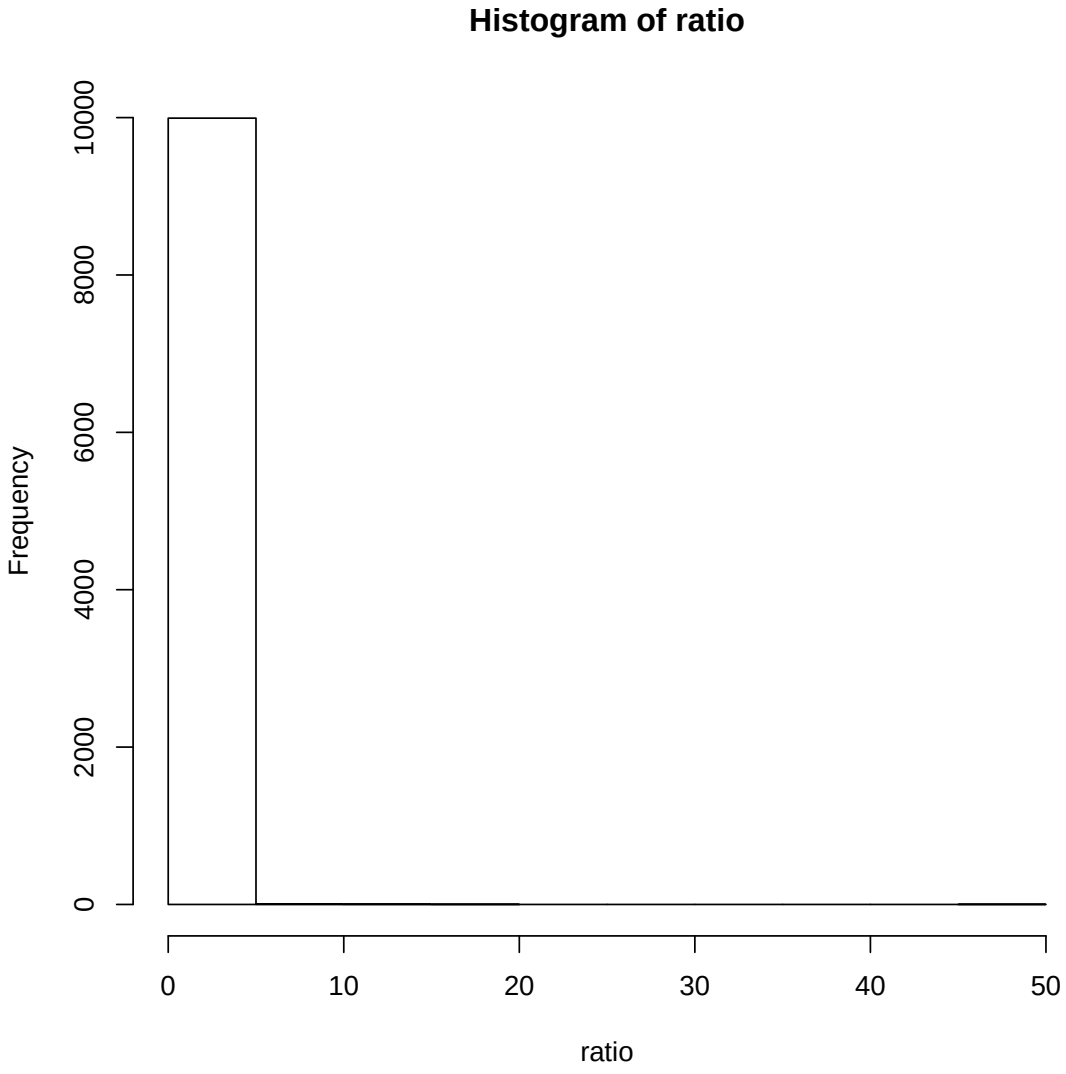
```
Esample <- sample*fsample/gsample
Esamplesquare <- samplesquare*fsample/gsample
ratio <- fsample/gsample

#draw histogram
hist(Esample)
```



```
hist(Esamplesquare)
```




```
hist(ratio)
```



There are severeal extreme small values in g(x), becuase the exponential distribution decays much more quickly than Pareto distribution. So when we take the reciprocal, we see that there are going to be extreme weights that have a very strong infludence on the estimator.

2 Problem 2

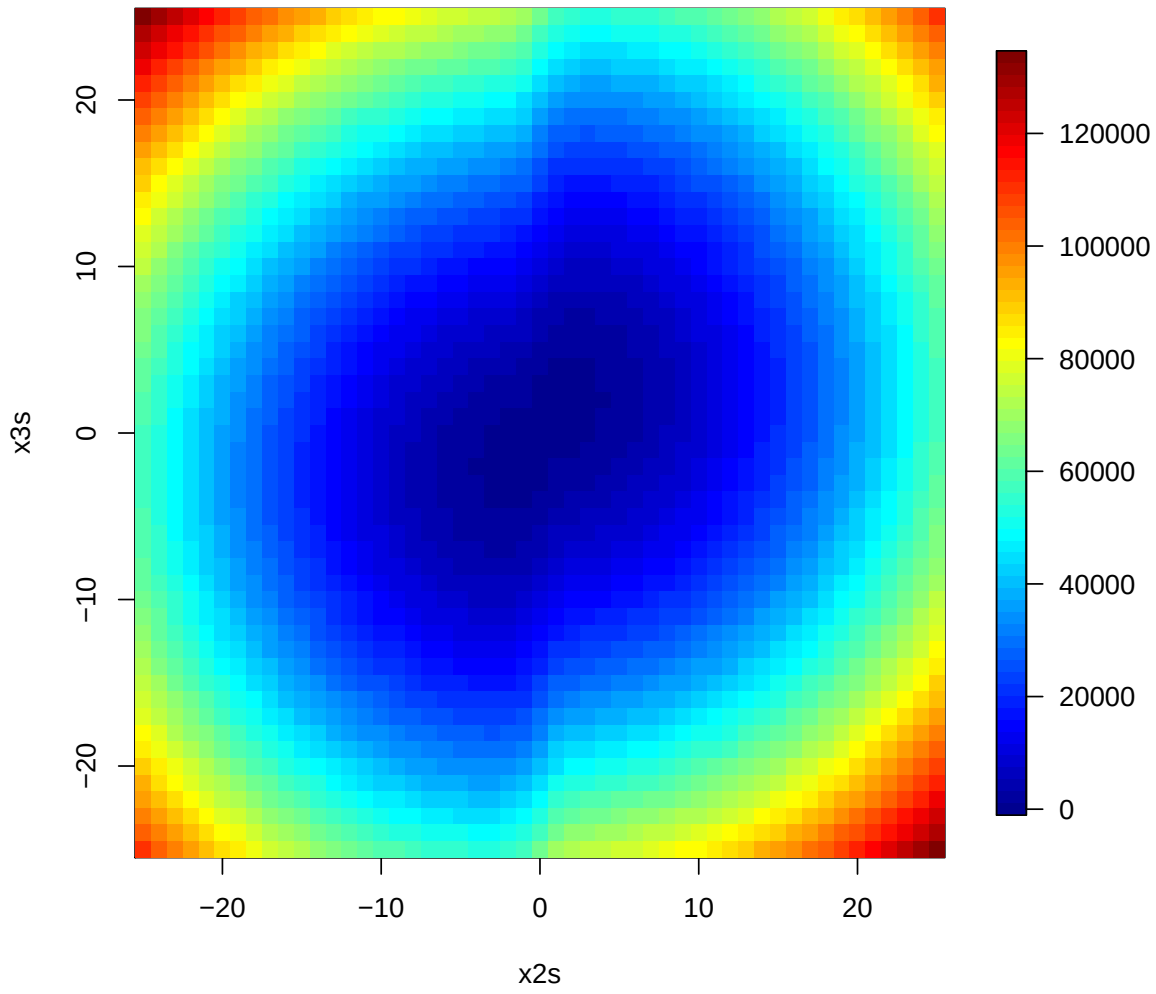
We first plot slicesof the function to get a sense for how it behaves. To be precise, we set one input be constant 1 and the other two input change from -25 to 25.

We then create 216 different starting points and use optimx() function and BFGS method to find the minimum of this function. Note that we find out it is possible for

us to find multiple local minima by using different starting points. Actually, we find 5 different local minima of the function.

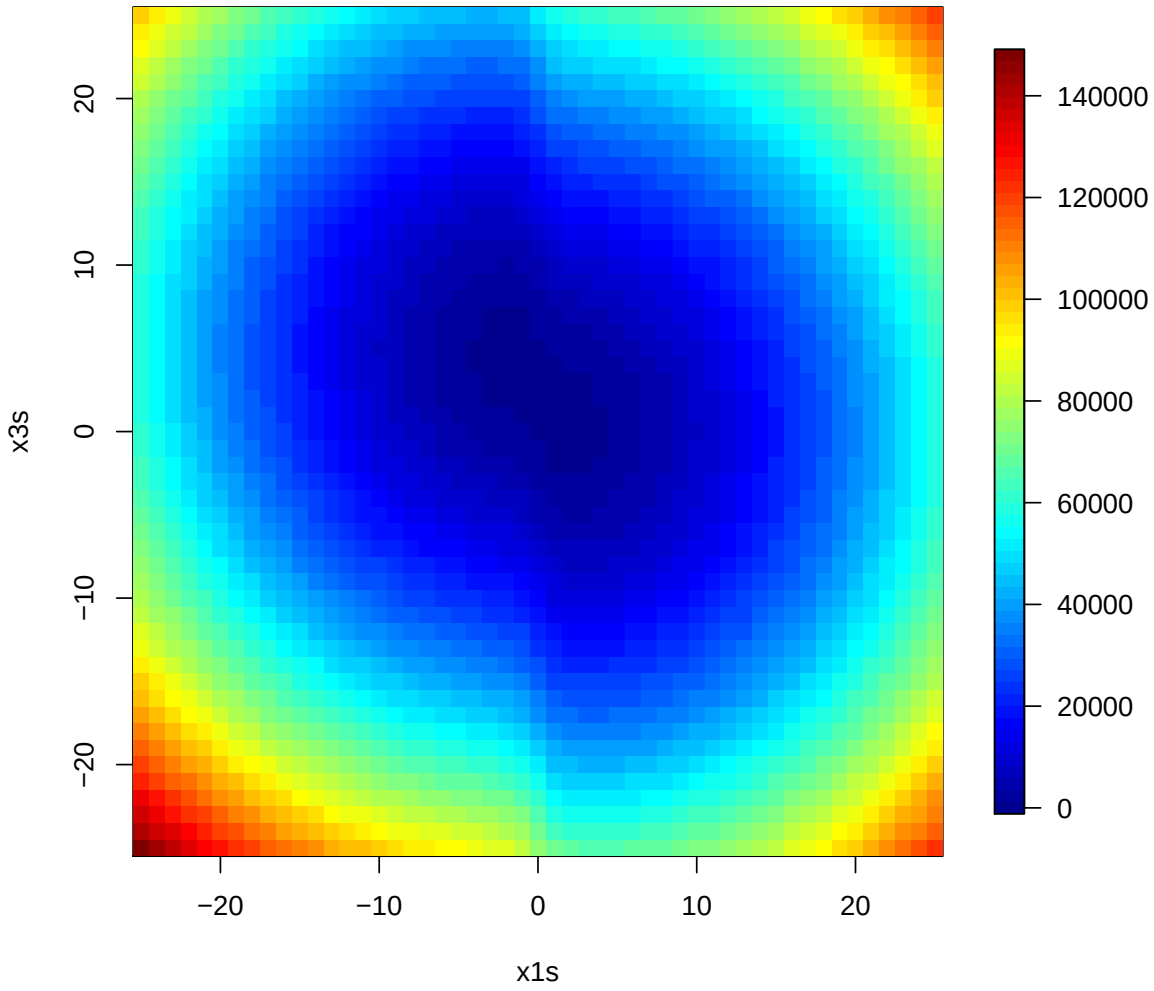
```
#helical valley
theta <- function(x1,x2) atan2(x2, x1)/(2*pi)

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)
}


library(fields)

## Loading required package:  spam
## Loading required package:  grid
## Spam version 1.3-0 (2015-10-24) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g.  'help( chol.spam)'.
##
## Attaching package:  'spam'
## The following object is masked from 'package:stats4':
##
##    mle
## The following objects are masked from 'package:base':
##
##    backsolve, forwardsolve
## Loading required package:  maps
##
##  # maps v3.1:  updated 'world':  all lakes moved to separate new #
##  # 'lakes' database.  Type '?world' or 'news(package="maps")'.  #

#set x1 as q and let x2,x3 vary from -25 to 25
x1s <- rep(1, 51)
x2s <- seq(-25, 25, len = 51)
```

```
x3s <- seq(-25, 25, len = 51)
x23s <- expand.grid(x2s, x3s)
xs <- cbind(rep(1,51^2), x23s)
fx <- apply(xs, 1, f)
image.plot(x2s, x3s, matrix(fx, 51,51))
```



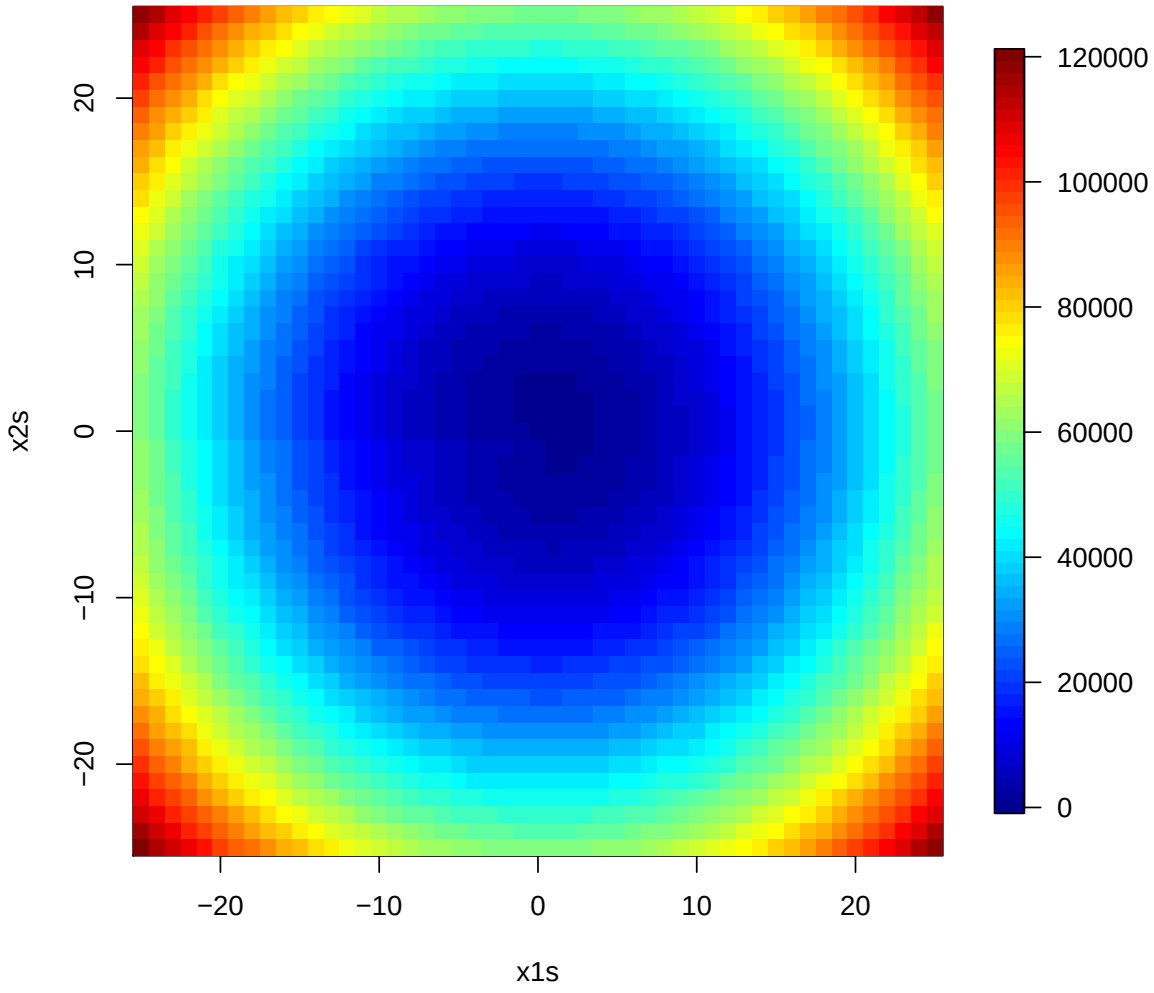

```
#set x2 as q and let x1,x3 vary from -25 to 25
x1s <- seq(-25, 25, len = 51)
x2s <- rep(1, 51)
x3s <- seq(-25, 25, len = 51)
x13s <- expand.grid(x1s, x3s)
```

```
xs <- cbind(x13s[,1], rep(1, 51^2), x13s[,2])
fx <- apply(xs, 1, f)
image.plot(x1s, x3s, matrix(fx, 51,51))
```




```
#set x3 as q and let x1,x2 vary from -25 to 25
x1s <- seq(-25, 25, len = 51)
x2s <- seq(-25, 25, len = 51)
x3s <- rep(1, 51)
x12s <- expand.grid(x1s, x2s)
xs <- cbind(x12s, rep(1, 51^2))
fx <- apply(xs, 1, f)
```

```
image.plot(x1s, x2s, matrix(fx, 51,51))
```




```
#testing the impact of different starting points
library(optimx)
#create different starting points
range <- seq(-20, 20, len = 6)
range3d <- expand.grid(range, range, range)

#find the unique local minimma using optimx()
findminimal <- function(startingpoint ,fun = f){optimx(startingpoint, fun, method = "BFGS")
optimalresult <- apply(range3d, MARGIN = 1, findminimal)
```

```
optimalpoint <- matrix(unlist(optimalresult), nrow =11)[1:3,]
optimalpoint <- round(optimalpoint, 2)
unioptpoint <- optimalpoint[,!duplicated(optimalpoint, MARGIN = 2)]
#every column of the matrix is a local minimal
unioptpoint
```

```
##      [,1]  [,2]  [,3]  [,4]  [,5]
## [1,]    1 -0.75 -1.94 -0.75 -1.94
## [2,]    0  0.00  0.00  0.00  0.00
## [3,]    0  7.83  5.93 -7.83 -5.93
```

3 Problem 3

3.1 Problem a

Let $Y_{obs,i}$ denote the observed data and Z_i denote the censored data, and in a given sample, we have c of the n observations will be cenosred. We then define some quantities which may simplify the calculation of $Q(\theta|\theta_t)$

$$\sigma^2_{\tau,i,t} = \sigma^2_t(1 + \tau^*_{i,t}\rho(\tau^*_{i,t}) - \rho(\tau^*_{i,t})^2)$$

$$\mu_i = \beta_0 + \beta_1 x_i$$

$$\mu_{\tau,i,t} = \beta_{0,t} + \beta_{1,t}x_i + \sigma_t\rho(\tau^*_{t,i})$$

$$\tau^*_{i,t} = (\tau - \beta_{0,t} - \beta_{1,t}x_i)/(\sigma_t)$$

Given the above notations, we can calculate $Q(\theta|\theta_t)$ as

$$\begin{aligned}
Q(\theta|\theta_t) &= E(\log \mathcal{L}(\theta|Y)|y_{obs}, \theta_t) \\
&= E(\log f(Y|\theta)|y_{obs}, \theta_t) \\
&= \sum_{i=1}^{n-c} E(\log f(Y_{obs,i}|\theta)|y_{obs}, \theta_t) + \sum_{i=n-c+1}^{n} E(\log f(Z_i|\theta)|y_{obs}, \theta_t) \\
&= \sum_{i=1}^{n-c}(-\log\sqrt{2\pi}\sigma - \frac{(y_{obs,i} - \beta_0 - \beta_1 x_i)^2}{2\sigma^2}) + \sum_{i=n-c+1}^{n} E(-\log(\sqrt{2\pi}\sigma) - \frac{(z_i - \beta_0 - \beta_1 x_i)^2}{2\sigma^2}|y_{obs}, \theta_t)
\end{aligned}$$

where

$$\sum_{i=1}^{n-c} E(\log f(Y_{obs,i}|\theta)|y_{obs},\theta_t) = \sum_{i=1}^{n-c}(-\log\sqrt{2\pi}\sigma - \frac{(y_{obs,i}-\beta_0-\beta_1 x_i)^2}{2\sigma^2})$$

and

$$\begin{aligned}\sum_{i=n-c+1}^{n} E(\log f(Z_i|\theta)|y_{obs},\theta_t) &= \sum_{i=n-c+1}^{n} E(-\log(\sqrt{2\pi}\sigma) - \frac{(z_i-\beta_0-\beta_1 x_i)^2}{2\sigma^2}|y_{obs},\theta_t)\\ &= \sum_{i=n-c+1}^{n}(-\log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2}E(Z_i^2 - 2(\beta_0+\beta_1 x_i)Z_i + (\beta_0+\beta_1 x_i)^2|y_{obs},\theta_t))\\ &= \sum_{i=n-c+1}^{n}(-\log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2}(\sigma_{\tau,i,t}^2 + \mu_{\tau,i,t}^2 - 2\mu_i\mu_{\tau,i,t} + \mu_i^2))\end{aligned}$$

In summary, $Q(\theta|\theta_t)$ can be expressed as

$$Q(\theta|\theta_t) = \sum_{i=1}^{n-c}(-\log\sqrt{2\pi}\sigma - \frac{(y_{obs,i}-\beta_0-\beta_1 x_i)^2}{2\sigma^2}) + \sum_{i=n-c+1}^{n}(-\log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2}(\sigma_{\tau,i,t}^2+\mu_{\tau,i,t}^2-2\mu_i\mu_{\tau,i,t}+\mu_i^2))$$

We then take derivative of $Q(\theta|\theta_t)$ with respect to $\beta_0, \beta_1, \sigma^2$.

$$\begin{aligned}\frac{\partial Q(\theta|\theta_t)}{\partial\beta_0} &= \sum_{i=1}^{n-c}\frac{(y_{obs,i}-\mu_i)}{\sigma^2} + \sum_{i=n-c+1}^{n}\frac{(\mu_{\tau,i,t}-\mu_i)}{\sigma^2}\\ \frac{\partial Q(\theta|\theta_t)}{\partial\beta_1} &= \sum_{i=1}^{n-c}\frac{(y_{obs,i,t}-\beta_0-\beta_1 x_i)}{\sigma^2}x_i + \sum_{i=n-c+1}^{n}\frac{(\mu_{\tau,i,t}-\mu_i)}{\sigma^2}x_i\\ \frac{\partial Q(\theta|\theta_t)}{\partial\sigma^2} &= -\frac{n}{2\sigma^2} + \sum_{i=1}^{n-c}\frac{(y_{obs,i-\mu_i})^2}{2(\sigma^2)^2} + \sum_{i=n-c+1}^{n}\frac{\sigma_{\tau,i,t}^2}{2(\sigma^2)^2} + \sum_{i=n-c+1}^{n}\frac{(\mu_{\tau,i,t}-\mu_i)^2}{2(\sigma^2)^2}\end{aligned}$$

Set the three derivatives equalo to 0. We have

$$\sigma^2 = \sum_{i=1}^{n-c}\frac{(y_{obs,i}-\mu_i)^2}{n} + \sum_{i=n-c+1}^{n}\frac{\sigma_{\tau,i,t}^2}{n} + \sum_{i=n-c+1}^{n}\frac{(\mu_{\tau,i,t}-\mu_i)^2}{n}$$

We can see that the estimator of σ^2 involes a ratio where the numerator is the usual sum of squares for the non-censored data, the deviation of censored data from the censored expectation, and the deviation of the censored expectation from the origianl expectation.

And

$$\begin{cases}\sum_{i=1}^{n-c}(y_{obs,i}-\beta_0+\beta_1 x_i) + \sum_{i=n-c+1}^{n}(\mu_{\tau,i,t}-\beta_0-\beta_1 x_i) = 0\\ \sum_{i=1}^{n-c}(y_{obs,i}-\beta_0+\beta_1 x_i)x_i + \sum_{i=n-c+1}^{n}(\mu_{\tau,i,t}-\beta_0-\beta_1 x_i)x_i = 0\end{cases}$$

We can see that β_0, β_1 can actually be expressed as the regression ceofficient of $\{Y_{obs}, \mu_{\tau,t}\}$ on $\{x\}$. So we can use the lm() function to update $\beta_{0,t}, \beta_{1,t}$ in the calculation.

3.2 Problem b

As we have all the information of non-censored data, we can use it to calculate a reasonable starting point for the parameters. The staring value for β_0 and β_1 should be the regression coefficient of $\{Y_{obs}\}$ on $\{x\}$. And the starting value for σ^2 should be the sample variance of observed data.

$$\begin{aligned}
\beta_1^{(0)} &= \frac{\sum_{i=1}^{n-c}(x_{obs,i} - \bar{x}_{obs})(y_{obs,i} - \bar{y}_{obs})}{\sum_{i=1}^{n-c}(x_{obs,i} - \bar{x}_{obs})^2} \\
\beta_0^{(0)} &= \bar{y}_{obs,i} - \beta_1^{(0)}\bar{x}_{obs} \\
\sigma^2_{(0)} &= \frac{\sum_{i=1}^{n-c}(y_{obs,i} - \beta_0^{(0)} - \beta_1^{(0)}x_{obs,i})^2}{n-c-1}
\end{aligned}$$

3.3 Problem c

```
###EM algorithm
set.seed(1)
n <- 100
beta0 <- 1
beta1 <- 2
sigma2 <- 6

X <- runif(n)
yComplete <- rnorm(n, beta0 + beta1*X, sqrt(sigma2))

## parameters chose such that signal in data is moderately strong
## estimate divided by std error is ~ 3
mod <- lm(yComplete ~ X)
summary(mod)$coef

##              Estimate Std. Error  t value    Pr(>|t|)
## (Intercept) 0.5607442  0.5041346 1.112290 0.268734381
## X           2.7650812  0.8657927 3.193699 0.001889262

#calculate \tau^*
gettaustar <- function(tau, mu, sigma2) (tau - mu)/sqrt(sigma2)
```

```
#calculate \rho(\tau^*)
getrhotaustar <- function(tau, mu, sigma2){
  dnorm(gettaustar(tau, mu, sigma2))/(1 - pnorm(gettaustar(tau, mu, sigma2)))
}

#calculate censored expectation
getEZ <- function(tau, mu, sigma2){
  mu + sqrt(sigma2) * getrhotaustar(tau,mu,sigma2)
}

#calculate cenosred variance
getVarZ <- function(tau, mu, sigma2){
  sigma2 * (1 + gettaustar(tau, mu, sigma2) * getrhotaustar(tau, mu, sigma2) - getrhotausta
}

#update theta usig lm() function
updatetheta <- function(thetat, Y, X, tau, censorindex){
  beta0t <- thetat[1]
  beta1t <- thetat[2]
  sigma2t <- thetat[3]
  mu <- beta0t + beta1t * X
  Y[censorindex] <- getEZ(tau, mu[censorindex], sigma2t)
  newsigma2 <- sum((Y-mu)^2)/n  + sum(getVarZ(tau, mu[censorindex], sigma2t))/n
  newbeta <- lm(Y ~ X)$coef
  newtheta <- c(newbeta, newsigma2)
  return(newtheta)
}


Y <- yComplete
#maximal iteration times
simu <- 1000
#stopping-iteration criteria
eplison <- 1e-6


#set tau so that the proportion of exceedances as about 0.2 and 0.8
```

```
Tau <- c(sort(yComplete)[(1-0.2)*n], sort(yComplete)[(1-0.8)*n])

for (taunum in 1:length(Tau)){
  censorindex <- which(yComplete>Tau[taunum])
  censoredsize <- length(censorindex)
  censorprob <- sum(Y>Tau[taunum])/n
  Yobs <- Y[-censorindex]
  Xobs <- X[-censorindex]
  Xcensor <- X[censorindex]
  #staring point
  beta1ini <- ((Xobs - mean(Xobs)) %*%  (Yobs - mean(Yobs)))/
    ((Xobs - mean(Xobs)) %*%  (Xobs - mean(Xobs)))
  beta0ini <- mean(Yobs) - beta1ini*mean(Xobs)
  sigma2ini <- sum((Yobs-beta0ini-beta1ini*Xobs)^2)/(length(Yobs)-1)
  thetaini <- c(beta0ini, beta1ini, sigma2ini)
  thetat <- thetaini
  for (i in 1:simu){
    if (sum(abs(thetat-updatetheta(thetat, Y, X, Tau[taunum], censorindex)))<eplison){
      break
    }
    else{
      thetat <- updatetheta(thetat, Y, X, Tau[taunum], censorindex)
    }
  }
  #the optimal point and times of iteration
  optresult <- c(thetat,i)
  print(optresult)
}

## (Intercept)           X
##   0.4562561   2.8219957   4.6099379  13.0000000
## (Intercept)           X
##   0.3816172   2.6588352   3.7986654 164.0000000
```

3.4 Problem d

We first calculat the log-likelihood in this case.

$$\begin{aligned}
\log \mathcal{L}(\theta|Y) &= \log \prod_{i=1}^{n-c} f(Y_{obs}|\theta) \prod_{i=n-c+1}^{n} P(Y_i > \tau) \\
&= \sum_{i=1}^{n-c} \log f(Y_{obs}|\theta) + \sum_{i=n-c+1}^{n} \log P(Y_i > \tau) \\
&= \sum_{i=1}^{n-c} (-\log\sqrt{2\pi} - \log\sigma - \frac{(y_{obs,i} - \mu_i)^2}{2\sigma^2}) + \sum_{i=n-c+1}^{n} \log(1 - \Phi(\frac{\tau - \mu_i}{\sigma}))
\end{aligned}$$

Note that in coding , we are going to use $\log\sigma$ as a parameter instead of σ^2 . This is because if we try to optimize with respect to σ^2, we'll have a problem whenever optim() tries to use a negative value since there is a log(sigma) term in the log-likelihood.

```
for (taunum in 1:length(Tau)){
  tau <- Tau[taunum]
  censorindex <- which(Y>tau)
  censoredsize <- length(censorindex)
  censorprob <- sum(Y>tau)/n
  Yobs <- Y[-censorindex]
  Xobs <- X[-censorindex]
  Xcensor <- X[censorindex]

  beta1ini <- ((Xobs - mean(Xobs)) %*%  (Yobs - mean(Yobs)))/((Xobs - mean(Xobs)) %*%  (Xob
  beta0ini <- mean(Yobs) - beta1ini*mean(Xobs)
  sigma2ini <- sum((Yobs-beta0ini-beta1ini*Xobs)^2)/(length(Yobs)-2)
  ini <- c(beta0ini,beta1ini, log(sqrt(sigma2ini)))

  minusloglikelihood <- function(theta){
    beta0 <- theta[1]
    beta1 <- theta[2]
    logsigma <- theta[3]
    muobs <- beta0 + beta1 * Xobs
    mucensor <- beta0 + beta1 * Xcensor
    loglikelihood <-  - (n-censoredsize)*(logsigma) - 1/(2*exp(2*logsigma))*sum((Yobs-muobs
    return(-loglikelihood)
  }
  output <- optimx(ini, minusloglikelihood, method = "BFGS")
  #the optimal point and times of iteration
```

```
  optresult <- c(output$p1,output$p2,exp(2*output$p3), output$fevals)
  print(optresult)
}

## [1]  0.4562535  2.8219885  4.6099156 27.0000000
## [1]  0.381620  2.658841  3.798683 41.000000
```

Based on the results above,we see that EM and BFGS converge to the same optimal point, but the iteration times are different(which is kind of related to the proportion of exceedances). In the modest case, EM takes less iterations , 13 instead of 27. But in a high proportion case, EM takes 164 times, while BFGS takes 41. I think this also related to how we decided when to stop the optimization in EM algorithm. If the criteria is more strict, then we may need more iteratons in EM algorithm