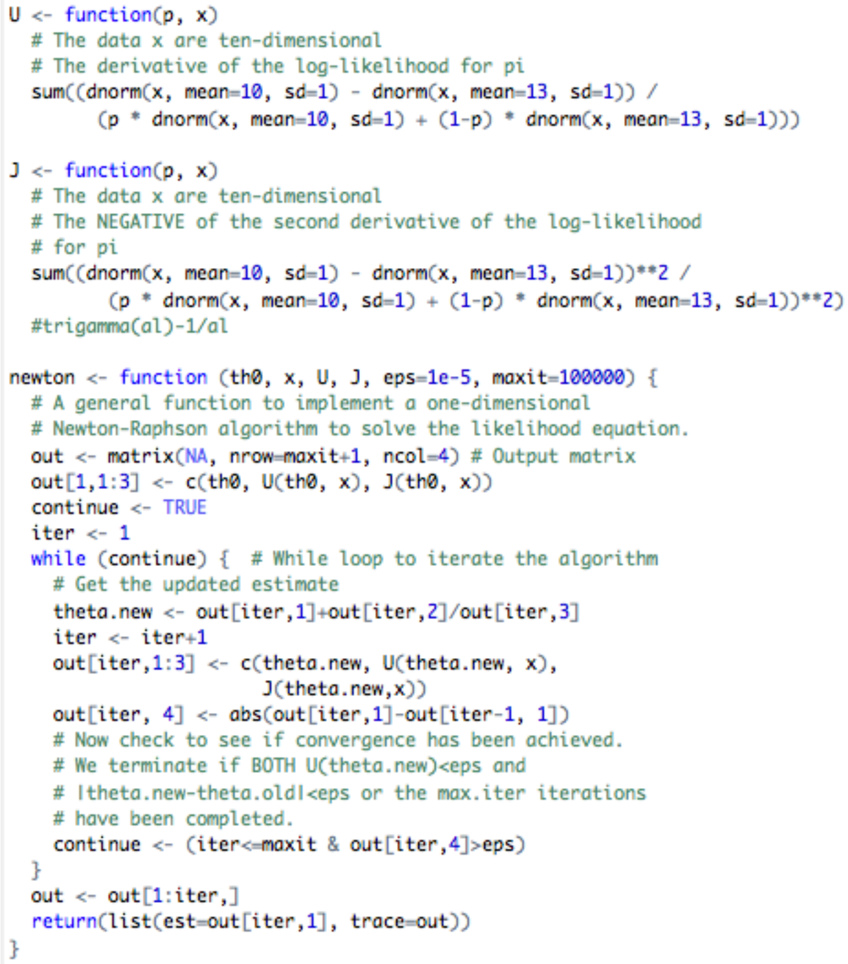
Newton-Raphson:

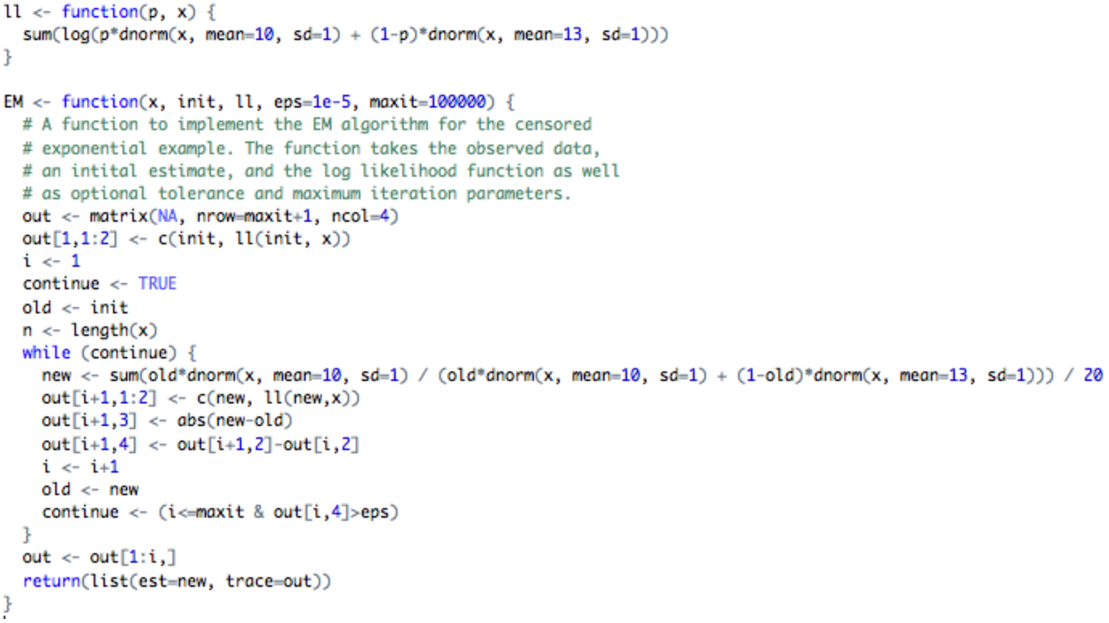


```
U <- function(p, x)
  # The data x are ten-dimensional
  # The derivative of the log-likelihood for pi
  sum((dnorm(x, mean=10, sd=1) - dnorm(x, mean=13, sd=1)) /
        (p * dnorm(x, mean=10, sd=1) + (1-p) * dnorm(x, mean=13, sd=1)))

J <- function(p, x)
  # The data x are ten-dimensional
  # The NEGATIVE of the second derivative of the log-likelihood
  # for pi
  sum((dnorm(x, mean=10, sd=1) - dnorm(x, mean=13, sd=1))**2 /
         (p * dnorm(x, mean=10, sd=1) + (1-p) * dnorm(x, mean=13, sd=1))**2)
  #trigamma(al)-1/al

newton <- function (th0, x, U, J, eps=1e-5, maxit=100000) {
  # A general function to implement a one-dimensional
  # Newton-Raphson algorithm to solve the likelihood equation.
  out <- matrix(NA, nrow=maxit+1, ncol=4) # Output matrix
  out[1,1:3] <- c(th0, U(th0, x), J(th0, x))
  continue <- TRUE
  iter <- 1
  while (continue) {  # While loop to iterate the algorithm
    # Get the updated estimate
    theta.new <- out[iter,1]+out[iter,2]/out[iter,3]
    iter <- iter+1
    out[iter,1:3] <- c(theta.new, U(theta.new, x),
                       J(theta.new,x))
    out[iter, 4] <- abs(out[iter,1]-out[iter-1, 1])
    # Now check to see if convergence has been achieved.
    # We terminate if BOTH U(theta.new)<eps and
    # |theta.new-theta.old|<eps or the max.iter iterations
    # have been completed.
    continue <- (iter<=maxit & out[iter,4]>eps)
  }
  out <- out[1:iter,]
  return(list(est=out[iter,1], trace=out))
}
```

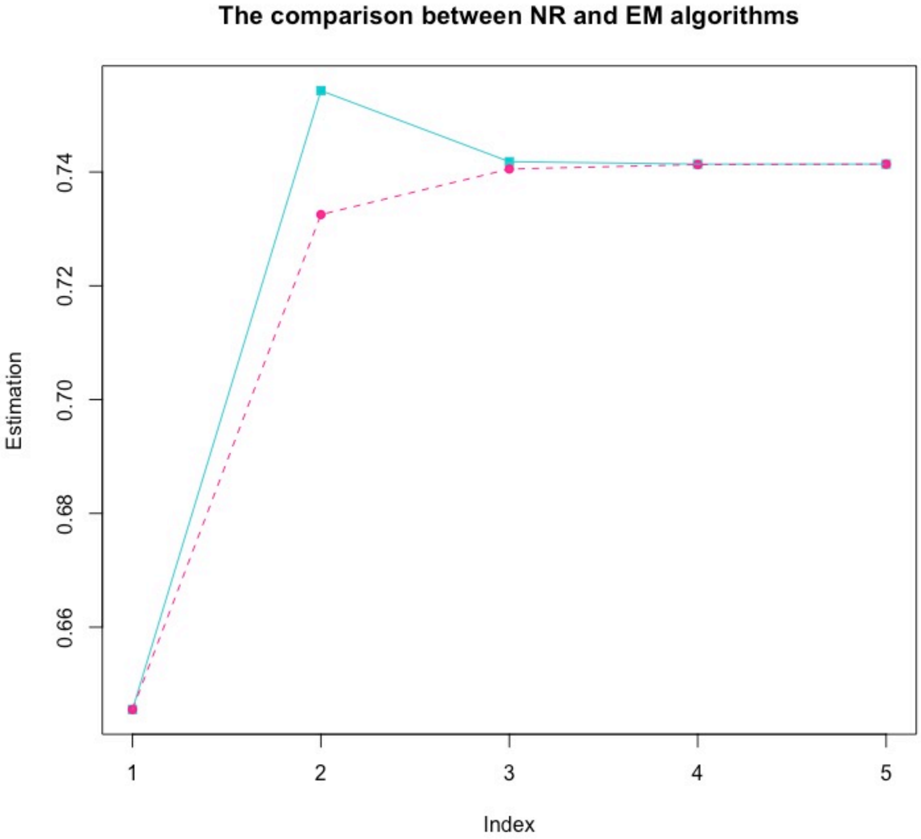
EM:



```
ll <- function(p, x) {
  sum(log(p*dnorm(x, mean=10, sd=1) + (1-p)*dnorm(x, mean=13, sd=1)))
}

EM <- function(x, init, ll, eps=1e-5, maxit=100000) {
  # A function to implement the EM algorithm for the censored
  # exponential example. The function takes the observed data,
  # an intital estimate, and the log likelihood function as well
  # as optional tolerance and maximum iteration parameters.
  out <- matrix(NA, nrow=maxit+1, ncol=4)
  out[1,1:2] <- c(init, ll(init, x))
  i <- 1
  continue <- TRUE
  old <- init
  n <- length(x)
  while (continue) {
    new <- sum(old*dnorm(x, mean=10, sd=1) / (old*dnorm(x, mean=10, sd=1) + (1-old)*dnorm(x, mean=13, sd=1))) / 20
    out[i+1,1:2] <- c(new, ll(new,x))
    out[i+1,3] <- abs(new-old)
    out[i+1,4] <- out[i+1,2]-out[i,2]
    i <- i+1
    old <- new
    continue <- (i<=maxit & out[i,4]>eps)
  }
  out <- out[1:i,]
  return(list(est=new, trace=out))
}
```

Comparison:


The comparison between NR and EM algorithms
Estimation
0.66
0.68
0.70
0.72
0.74
Index
1
2
3
4
5