Homework 3 - EM algorithm

Renjie Wei, rw2844

```
library(ggplot2)
library(tidyverse)
```

Problem 1

Recall the ABO blood type data, where we have Nobs = (NA,NB,NO,NAB) = (26, 27, 42, 7).

- Implementing the EM algorithm to estimate the allele frequencies, PA, PB and PO in R and present your results.

Answer:

Since we only observed the number of people with each phenotype $N_{\text{obs}} = (N_A, N_B, N_{AB}, N_O) = (26, 27, 42, 7)$, our goal is to estimate the frequencies of alleles A, B, and O, denoted by p_A, p_B, and p_O respectively. So we estimate the frequencies according to the Hardy-Weinberg law, the genotype frequencies are:

$$\begin{aligned}
\text{Prob}(\text{Genotype} = A/A) &= p_A^2 \\
\text{Prob}(\text{Genotype} = A/O) &= 2p_Ap_O \\
\text{Prob}(\text{Genotype} = B/B) &= p_B^2 \\
\text{Prob}(\text{Genotype} = B/O) &= 2p_Bp_O \\
\text{Prob}(\text{Genotype} = A/B) &= 2p_Ap_B \\
\text{Prob}(\text{Genotype} = O/O) &= p_O^2
\end{aligned}$$

Furthermore, genotype counts $N = (N_{A/A}, N_{A/O}, N_{B/B}, N_{B/O}, N_{A/B}, N_{O/O})$ are jointly multinomially distributed with log-likelihood function as shown below.

$$\begin{aligned}
\log L(p|N) &= N_{A/A}\log(p_A^2) + N_{A/O}\log(2p_Ap_O) + N_{B/B}\log(p_B^2) + N_{B/O}\log(2p_Bp_O) \\
&+ N_{A/B}\log(2p_Ap_B) + N_{O/O}\log(p_O^2) \\
&+ \log\Big(\frac{n!}{N_{A/A}!N_{A/O}!N_{B/B}!N_{B/O}!N_{A/B}!N_{O/O}!}\Big)
\end{aligned}$$

where $n = N_{A/A} + N_{A/O} + N_{B/B} + N_{B/O} + N_{A/B} + N_{O/O}$.

E-step

Note $N_{A/A} + N_{A/O} = N_A$ and $N_{B/B} + N_{B/O} = N_B$. Thus the conditional distribution of $N_{A/A}|N_A$ and $N_{B/B}|N_B$ are:

$$N_{A/A}|N_A \sim \text{Bin}\left(N_A, \frac{p_A^2}{p_A^2 + 2p_Ap_O}\right)$$

and

$$N_{B/B}|N_B \sim \text{Bin}\left(N_B, \frac{p_B^2}{p_B^2 + 2p_B p_O}\right)$$

respectively.

Therefore, the expectations in the k-th iteration can be easily calculated as follows.

$$N_{A/A}^{(k)} = E(N_{A/A}|N_{\text{obs}}, p^{(k)}) = N_A \times \frac{{p_A^{(k)}}^2}{{p_A^{(k)}}^2 + 2p_A^{(k)}p_O^{(k)}}$$

$$N_{A/O}^{(k)} = E(N_{A/O}|N_{\text{obs}}, p^{(k)}) = N_A \times \frac{2p_A^{(k)}p_O^{(k)}}{{p_A^{(k)}}^2 + 2p_A^{(k)}p_O^{(k)}}$$

$$N_{B/B}^{(k)} = E(N_{B/B}|N_{\text{obs}}, p^{(k)}) = N_B \times \frac{{p_B^{(k)}}^2}{{p_B^{(k)}}^2 + 2p_B^{(k)}p_O^{(k)}}$$

$$N_{B/O}^{(k)} = E(N_{B/O}|N_{\text{obs}}, p^{(k)}) = N_B \times \frac{2p_B^{(k)}p_O^{(k)}}{{p_B^{(k)}}^2 + 2p_B^{(k)}p_O^{(k)}}.$$

Moreover, it is obvious that

$$N_{A/B}^{(k)} = E(N_{A/B}|N_{\text{obs}}, p^{(k)}) = N_{A/B}$$

and

$$N_{O/O}^{(k)} = E(N_{O/O}|N_{\text{obs}}, p^{(k)}) = N_{O/O}.$$

M-step

Now consider maximizing $Q(p|p^{(k)})$ under the restriction $p_A + p_B + p_O = 1$. Introduce Lagrange multiplier λ and maximize

$$Q_L(p, \lambda|p^{(k)}) = Q(p|p^{(k)}) + \lambda(p_A + p_B + p_O - 1)$$

with respect to $p = (p_A, p_B, p_O)$ and λ.\

$$\frac{\partial Q_L(p, \lambda|p^{(k)})}{\partial p_A} = \frac{2N_{A/A}^{(k)}}{p_A} + \frac{N_{A/O}^{(k)}}{p_A} + \frac{N_{A/B}^{(k)}}{p_A} + \lambda \quad (1)$$

$$\frac{\partial Q_L(p, \lambda|p^{(k)})}{\partial p_B} = \frac{2N_{B/B}^{(k)}}{p_B} + \frac{N_{B/O}^{(k)}}{p_B} + \frac{N_{A/B}^{(k)}}{p_B} + \lambda \quad (2)$$

$$\frac{\partial Q_L(p, \lambda|p^{(k)})}{\partial p_O} = \frac{N_{A/O}^{(k)}}{p_O} + \frac{N_{B/O}^{(k)}}{p_O} + \frac{2N_{O/O}^{(k)}}{p_O} + \lambda \quad (3)$$

$$\frac{\partial Q_L(p, \lambda|p^{(k)})}{\partial p_\lambda} = p_A + p_B + p_C - 1 \quad (4)$$

Since $N_{A/A}^{(k)} + N_{A/O}^{(k)} + N_{B/B}^{(k)} + N_{B/O}^{(k)} + N_{A/B}^{(k)} + N_{O/O}^{(k)} = n$, from the above four functions, we get $\lambda = -2n$. By plugging $\lambda = -2n$ in and setting (1), (2), and (3) to be zero, update (p_A, p_B, p_O) as follows.

$$p_A^{(k+1)} = \frac{2N_{A/A}^{(k)} + N_{A/O}^{(k)} + N_{A/B}^{(k)}}{2n}$$

$$p_B^{(k+1)} = \frac{2N_{B/B}^{(k)} + N_{B/O}^{(k)} + N_{A/B}^{(k)}}{2n}$$

$$p_O^{(k+1)} = \frac{2N_{O/O}^{(k)} + N_{A/O}^{(k)} + N_{B/O}^{(k)}}{2n}$$

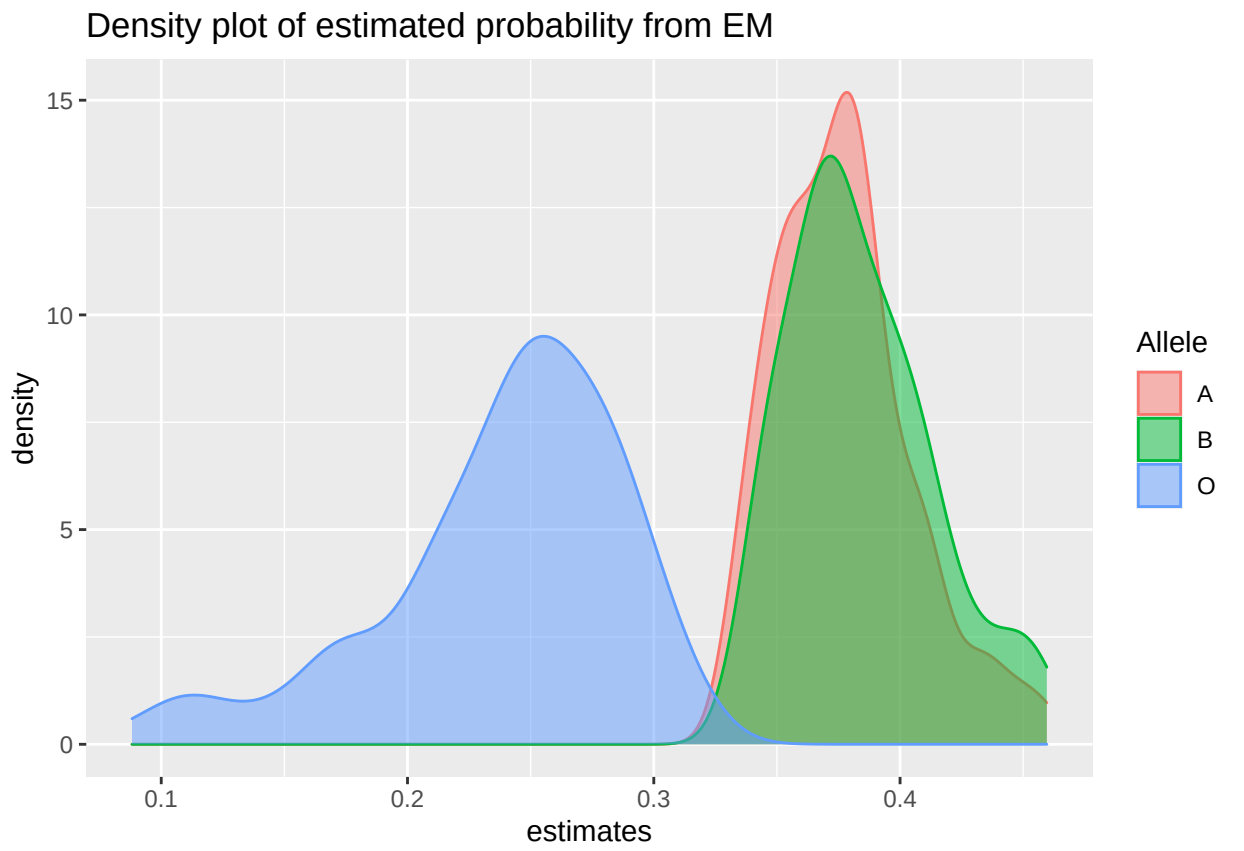
Repeat E-step and M-step until convergence.

R codes: I implement the EM algorithm in the following codes. And I generate 100 different starting values of P_A, P_B and P_O, then summary the results of estimated probabilities to a plot.

```
# E-step
# params: N - the obs data, pvec - now estimated c(pa, pb, po)
Q.abo <- function(N, pvec){
    pa = pvec[1]
    pb = pvec[2]
    po = pvec[3]

    Na = N[1]
    Nb = N[2]
    Nab = N[3]
    No = N[4]

    Naa = Na * (pa^2/(pa^2 + 2 * pa * po))
    Nao = Na * (2 * pa * po/(pa^2 + 2 * pa * po))
    Nbb = Nb * (pb^2/(pb^2 + 2 * pb * po))
    Nbo = Nb * (2 * pb * po/(pb^2 + 2 * pb * po))
    Nab = Nab
    Noo = No

    Qres <- c(Naa, Nao, Nbb, Nbo, Nab, Noo)
    return(Qres)
}

# M-step: Maximize posterior prob

M.abo <- function(N, Qres){
    n <- sum(N)
    Naa = Qres[1]
    Nao = Qres[2]
    Nbb = Qres[3]
    Nbo = Qres[4]
    Nab = Qres[5]
    Noo = Qres[6]
    pa <- (2 * Naa + Nao + Nab)/(2*n)
    pb <- (2 * Nbb + Nbo + Nab)/(2*n)
    po <- (2 * Noo + Nao + Nbo)/(2*n)
    Pres <- c(pa, pb, po)
    return(Pres)
}

EM.abo <- function(data, init_p = c(0.1, 0.1, 0.8), max_iter = 1e3, tol = 1e-5){
    i <- 0
    Q <- Q.abo(data, init_p)
    res <- c(0, init_p)
    error <- 1
    p_cur <- init_p
    p_old <- NULL
```

```
    while (i < max_iter & error > tol){
        i <- i + 1
        p_old <- p_cur
        p_cur <- M.abo(data, Q)
        error <- abs(sum(p_cur - p_old))
        Q <- Q.abo(data, p_cur)
        res <- rbind(res, c(i, p_cur))
    }
    return(res)
}
```

```
set.seed(2022)
dat.abo <- c(26, 27, 42, 7)
# generate sequence of starting probs
abo.start <- NULL
for(i in 1:100){
    set.seed(i)
    pa <- runif(1)
    pb <- runif(1)
    pc <- runif(1)
    sump <- pa+pb+pc
    abo.start <- rbind(abo.start, c(pa/sump, pb/sump, pc/sump))
}

res.abo <- NULL
for(j in 1:nrow(abo.start)){
    starts <- abo.start[j,]
    res <- EM.abo(dat.abo, starts)
    res.abo <- rbind(res.abo, res[nrow(res),])
}

colnames(res.abo) <- c("iteration", "PA", "PB", "PO")
res.abo <- data.frame(res.abo) %>% pivot_longer(cols = PA:PO, names_to = "Allele", values_to = "estimat

abo.plot <- ggplot(res.abo, aes(x = estimates, group = Allele, color = Allele, fill = Allele)) + geom_d
abo.plot
```




```
res.abo %>% group_by(Allele) %>% summarise(mean = mean(estimates), sd = sd(estimates)) %>% knitr::kable
```

Allele	mean	sd
A	0.3772045	0.0275550
B	0.3841941	0.0294350
O	0.2386013	0.0491643

By implementing EM algorithm using the observed data, it shows that given different starting probability vector of alleles, the probability of allele A centered around 0.377, the probability of allele B centered around 0.384 and the probability of allele O centered around 0.239. The density plots shows the estimated probability of allele O seems to have wider distribution than that of allele A and B.

Problem 2

Disease D is a chronic neurological condition that leads to fast deterioration of motor and cognitive functions and eventually leads to death. Based on a theoretical model, the survival time of a patient suffered from disease D very much depends on his or her disease onset age. To be specific, let Y as the survival time of a patient from their diagnosis, and X be the disease onset age, the conditional distribution for Y given $X = x$ is exponential with failure rate $0.01x$.

$$f(Y = t \mid X = x) = 0.01x \exp\{-0.01xt\}.$$

Since D is a chronic condition, its actual onset times are often unobserved. Suppose the disease onset ages in a population also follows an exponential with failure rate θ, where $\theta > 0$ is an unknown parameter. Suppose $\{Y_i,\ i = 1, ..., n\}$ are observed survival times of n patients with disease D in a population. The public health

researchers are interested in estimating the parameter θ in the population so that they could design disease prevention policies on target ages.

1. Write out the marginal distribution of Y, and the observed likelihood function of $\{Y_i,\, i = 1, ..., n\}$.

Answer:

The marginal distribution of Y is:

$$f_Y(y) = \int_0^\infty f_{XY}(x,y)dx = \int_0^\infty f_{Y|X}(y|x)f_X(x)dx = \int_0^\infty 0.01x\exp\{-0.01xy\}\times\theta\exp\{-\theta x\}dx = \frac{100\theta}{(100\theta+y)^2}$$

The observed likelihood is:

$$L_{\text{obs}}(\theta) = \prod_{i=1}^{n}\frac{100\theta}{(100\theta+y_i)^2} = 100^n\theta^n\prod_{i=1}^{n}(100\theta+y_i)^{-2}$$

2. Design a univariate optimization algorithm (e.g. Golden search or Newton's method) to find the MLE of the observed likelihood in (1), and specify each step of your algorithm. Implement the algorithm into an R function.

Answer:

I would like to develop a Newton's method to find the MLE in (1).

The likelihood function is:

$$L_{\text{obs}}(\theta) = 100^n\theta^n\prod_{i=1}^{n}(100\theta+y_i)^{-2}$$

In order to maximize the likelihood function, it is same to maximize the log-likelihood function:

$$l_{obs}(\theta) = n\log 100 + n\log\theta - 2\sum_{i=1}^{n}\log(100\theta+y_i)$$

According to Newton's method, we need to find the score function and the first derivative of the score function w.r.t θ:

$$S(\theta) = \frac{\partial l_{obs}(\theta)}{\partial\theta} = \frac{n}{\theta} - 2\sum_{i=1}^{n}100(100\theta+y_i)^{-1}$$

$$S'(\theta) = -\frac{n}{\theta^2} + 2\sum_{i=1}^{n}100^2(100\theta+y_i)^{-2}$$

The Newton's method told us to update θ_i each time to get the maximum of $L_{obs}(\theta)$ by doing this iteratively:

$$\theta_i = \theta_{i-1} - \frac{S(\theta_{i-1})}{S'(\theta_{i-1})}$$

R codes:

The implementation of this method is written in the function below:

```r
score.chronic <- function(Y, t){
    N <- length(Y)
    s <- N/t - 2*sum(100*((100 * t + Y)^(-1)))
    return(s)
}

dscore.chronic <- function(Y, t){
    N <- length(Y)
    ds <- 2 * sum((100^2)*((100 * t + Y)^(-2))) - (N/(t^2))
    return(ds)
}

newton.chronic <- function(Y, init, max_iter = 500, tol = 1e-10){
    i <- 0
    t <- init
    score <- score.chronic(Y, t)
    dscore <- dscore.chronic(Y, t)
    res <- c(i, t, score, dscore)
    while (i < max_iter & abs(score) > tol) {
        i <- i + 1
        score <- score.chronic(Y, t)
        dscore <- dscore.chronic(Y, t)
        t <- t - (score/dscore)
        res <- rbind(res, c(i, t, score, dscore))
    }
    return(res)
}
```

3. Write out the joint distribution of (Y, X), and design an EM algorithm to find the MLE of θ. Clearly write out the E-steps and M-steps in each iteration, and implement the algorithm into an R function.

Answer:

From the conditional distribution and the distribution of X, we can get the joint distribution of X, Y:

$$f_{XY}(x,y) = f_{Y|X}(y|x)f_X(x) = 0.01x\exp\{-0.01xy\} \times \theta\exp\{-\theta x\} = 0.01\theta x\exp\{-0.01xy - \theta x\}$$

The complete(joint) likelihood is:

$$L(\theta; X, Y) = \prod_{i=1}^{n} 0.01\theta x_i \exp\{-0.01x_i y_i - \theta x_i\} = 100^{-n}\theta^n \prod_{i=1}^{n} x_i \exp\{-0.01x_i y_i - \theta x_i\}$$

The log-likelihood is:

$$l(\theta; X, Y) = -n\log 100 + n\log\theta + \sum_{i=1}^{n}\{\log x_i - 0.01x_i y_i - \theta x_i\}$$

E-step

Since we cannot observe X_i, we replaced it by its expectation conditioning on our observed data Y_i:

$$\hat{x}_i^{(t)} = E(X_i|Y_i) = \int_0^\infty x f_{X|Y}(x|y)dx = \int_0^\infty x\frac{f_{XY}(x,y)}{f_Y(y)}dx = \int_0^\infty x\frac{0.01\theta x\exp\{-0.01xy - \theta x\} \times (100\theta + y)^2}{100\theta}dx = \frac{200}{y_i + 100\theta}$$

M-step

From the joint log-likelihood, we solve the MLE of $\hat{\theta}^{(t+1)}$:

$$\hat{\theta}^{(t+1)} = \frac{n}{\sum_{i=1}^{n} \hat{x}_i^{(t)}}$$

R codes:

```
Q.chronic <- function(Y, theta){
    Qres <- 200/(100*theta + Y)
}

M.chronic <- function(Y, Qres){
    N <- length(Y)
    Mres <- N/sum(Qres)
    return(Mres)
}

EM.chronic <- function(data, init_theta = 1, max_iter = 1e3, tol = 1e-5){
    i <- 0
    Q <- Q.chronic(data, init_theta)
    res <- c(0, init_theta)
    error <- 1
    theta_cur <- init_theta
    theta_old <- NULL
    while (i < max_iter & error > tol){
        i <- i + 1
        theta_old <- theta_cur
        theta_cur <- M.chronic(data, Q)
        error <- abs(sum(theta_cur - theta_old))
        Q <- Q.chronic(data, theta_cur)
        res <- rbind(res, c(i, theta_cur))
    }
    return(res)
}
```

4. Simulate data sets with true $\theta = 0.025$, and apply the optimization functions you developed in (2) and (3) to estimate θ, which algorithm is more efficient (comparing the number of iterations and computing times)?

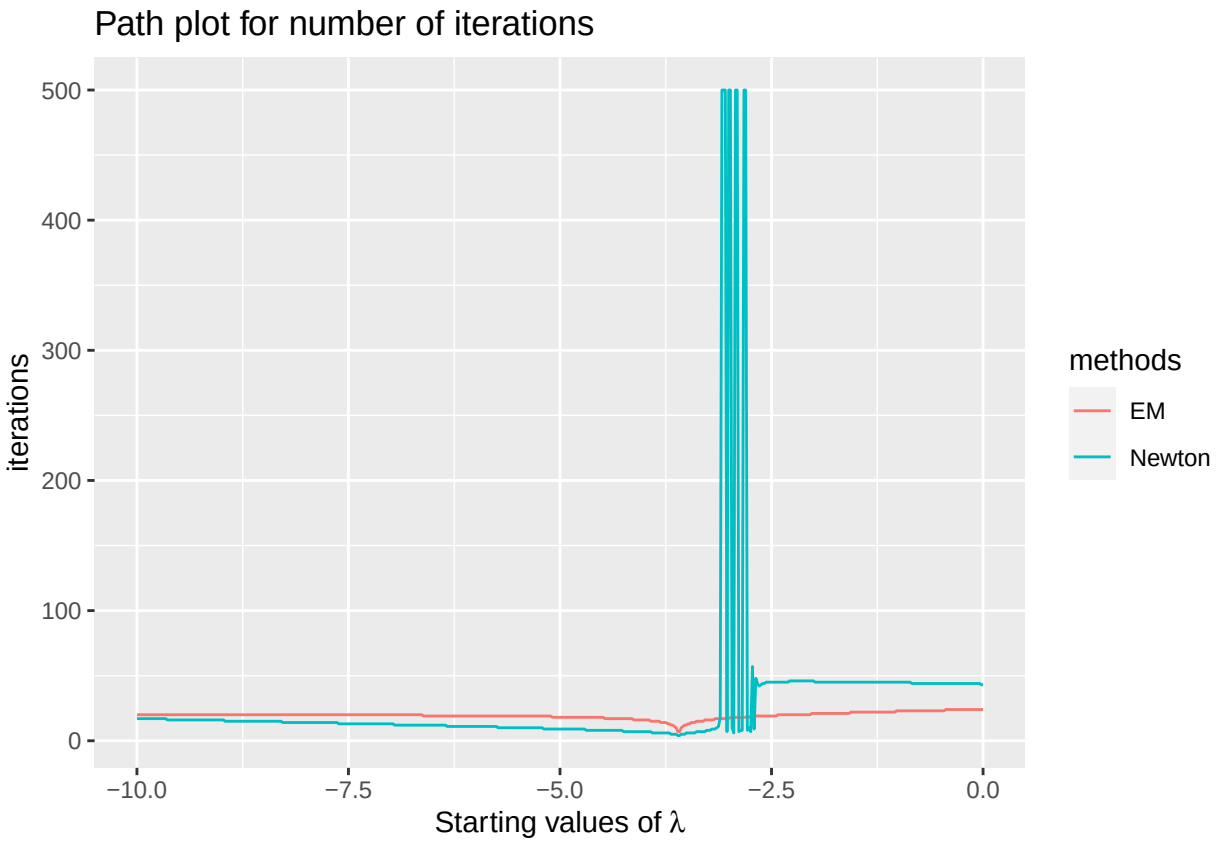
Answer:

I generated 1000 data to evaluate the performance of the algorithms. And I compared the algorithms performance under different starting values. The true $\theta = 0.025$.

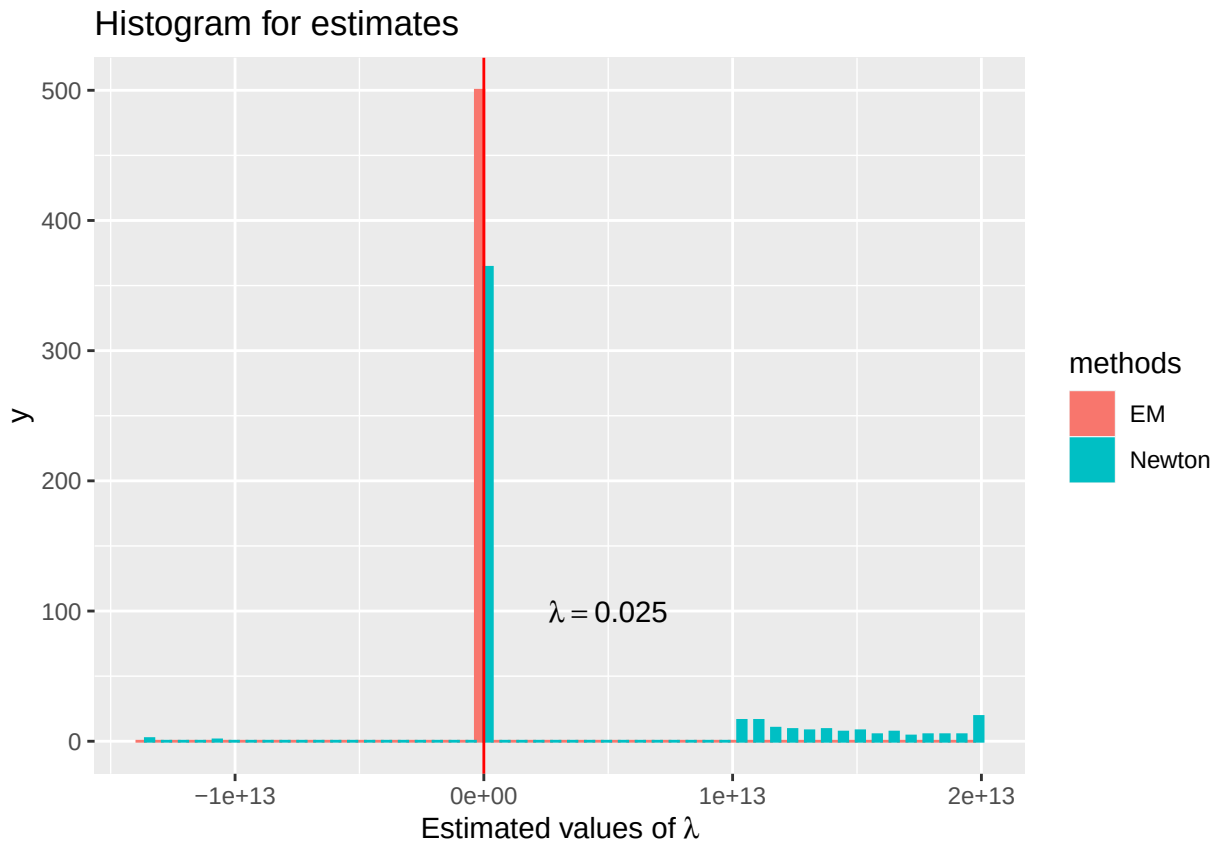
R codes:

```
chronic.data <- function(size = 500, theta = 0.025){
    X <- rexp(500, rate = 0.025)
    Y <- NULL
    for (i in 1:length(X)){
       Y[i] = rexp(1, 0.01 * X[i])
    }
```

```
    return(list(X = X, Y = Y))
}

set.seed(2022)
dat <- chronic.data(1000)


newton.chronic.res <- newton.chronic(dat$Y, 1)
EM.chronic.res <- EM.chronic(dat$Y, init_theta = 1)

theta.seq <- exp(seq(-10, 0, length = 500))

res <- NULL
for (i in 1:length(theta.seq)){
    cur <- theta.seq[i]
    newtonres <- newton.chronic(dat$Y, cur)
    EMres <- EM.chronic(dat$Y, cur)
    retn <- c(cur, newtonres[nrow(newtonres), 1][[1]],  newtonres[nrow(newtonres), 2][[1]], "Newton")
    rete <- c(cur, EMres[nrow(EMres), 1][[1]] , EMres[nrow(EMres), 2][[1]], "EM")
    res <- rbind(res, retn, rete)
}

res <- as.data.frame(res)
colnames(res) <- c("start", "iterations", "estimates", "methods")
res$start <- as.numeric(res$start)
res$iterations <- as.numeric(res$iterations)
res$estimates <- as.numeric(res$estimates)


iter.plot <- res %>% arrange(iterations) %>%
    ggplot(aes(x = log(start), y = iterations, group = methods, col = methods)) +
    geom_line() + xlab(bquote("Starting values of"~lambda)) +
    labs(title = 'Path plot for number of iterations')
iter.plot
```




```
res.plot <- res %>% arrange(estimates) %>%
    ggplot(aes(x = estimates, group = methods, col = methods, fill = methods)) +
    geom_histogram(bins = 50, position = "dodge") + geom_vline(xintercept = 0.025, color = "red") + xlal
    labs(title = 'Histogram for estimates') + annotate("text", x = 0.5e13, y = 100, parse = TRUE,
           label = "lambda == 0.025")

res.plot
```



The above plots shows the iterations and final estimates of two algorithms under different starting values. From the iteration plots, we can see in a specific range of starting values ($\theta_{\text{start}} < e^{-4}$), the Newton's method is more efficient than EM since it takes few iterations. However, when the starting value goes to a wider range, the Newton's method is not preferred anymore, since we can see some fluctuations in iterations and divergence in the final estimates. In some region which the starting values are close to 1, the Newton's method cannot converge (see the histogram of final estimates), both of them indicate that the Newton's method is not that stable. EM algorithm performs well under all starting values, the number of iterations are close and the final estimates are close too, which means it is stable.