Assignment 3

xw-zeng

2022-11-07

Problem sets.

- `4.1 (a)(b)(f)`
- `4.2 (a)(b)(c)`

List of optimization functions.

- `EM_Algorithm`: Expectation Maximization Algorithm.
- `EM_Aitken`: Aitken accelerated Expectation Maximization Algorithm.
- `Newton_Raphson`: Newton Raphson method.
- `Empirical_Information`: Compute the empirical information to estimate the covariance matrix of parameters estimated.

4.1

(a)

Unknown parameters of interst:

$$P = (P_C, P_I), P_T = 1 - P_C - P_I$$

Observed data:

$$X = (n_C, n_I, n_T, n_U)$$

Complete data:

$$Y = (n_{CC}, n_{CI}, n_{CT}, n_{II}, n_{IT}, n_{TT})$$

Likelihood function of complete data:

$$\begin{aligned} L(P \mid Y) =& \frac{n!}{n_{CC}!n_{CI}!n_{CT}!n_{II}!n_{IT}!n_{TT}!} \left({P_C}^2\right)^{n_{CC}} (2P_C P_I)^{n_{CI}} \\ & (2P_C P_T)^{n_{CT}} \left({P_I}^2\right)^{n_{II}} (2P_I P_T)^{n_{IT}} \left({P_T}^2\right)^{n_{TT}} \end{aligned}$$

Log likelihood function of complete data:

$$\begin{aligned} log f_Y(y \mid P) =& n_{CC} \log\left\{{P_C}^2\right\} + n_{CI} \log\left\{2P_C P_I\right\} + n_{CT} \log\left\{2P_C P_T\right\} \\ & + n_{II} \log\left\{{P_I}^2\right\} + n_{IT} \log\left\{2P_I P_T\right\} + n_{TT} \log\left\{{P_T}^2\right\} \\ & + log\begin{pmatrix} & & n & & & \\ n_{CC} & n_{CI} & n_{CT} & n_{II} & n_{IT} & n_{TT} \end{pmatrix} \end{aligned}$$

Conditional expectations:

$$E\left\{n_{CC} \mid n_C, n_I, n_T, n_U, P^{(t)}\right\} = n_{CC}^{(t)} = \frac{n_C\left(P_C^{(t)}\right)^2}{\left(P_C^{(t)}\right)^2 + 2P_C^{(t)}P_I^{(t)} + 2P_C^{(t)}P_T^{(t)}}$$

$$E\left\{n_{CI} \mid n_C, n_I, n_T, n_U, P^{(t)}\right\} = n_{CI}^{(t)} = \frac{2n_C P_C^{(t)}P_I^{(t)}}{\left(P_C^{(t)}\right)^2 + 2P_C^{(t)}P_I^{(t)} + 2P_C^{(t)}P_T^{(t)}}$$

$$E\left\{n_{CT} \mid n_C, n_I, n_T, n_U, P^{(t)}\right\} = n_{CT}^{(t)} = \frac{2n_C P_C^{(t)}P_T^{(t)}}{\left(P_C^{(t)}\right)^2 + 2P_C^{(t)}P_I^{(t)} + 2P_C^{(t)}P_T^{(t)}}$$

$$E\left\{n_{II} \mid n_C, n_I, n_T, n_U, P^{(t)}\right\} = n_{II}^{(t)} = \frac{n_I\left(P_I^{(t)}\right)^2}{\left(P_I^{(t)}\right)^2 + 2P_I^{(t)}P_T^{(t)}} + \frac{n_U\left(P_I^{(t)}\right)^2}{\left(P_I^{(t)}\right)^2 + 2P_I^{(t)}P_T^{(t)} + \left(P_T^{(t)}\right)^2}$$

$$E\left\{n_{IT} \mid n_C, n_I, n_T, n_U, P^{(t)}\right\} = n_{IT}^{(t)} = \frac{2n_I P_I^{(t)}P_T^{(t)}}{\left(P_I^{(t)}\right)^2 + 2P_I^{(t)}P_T^{(t)}} + \frac{2n_U P_I^{(t)}P_T^{(t)}}{\left(P_I^{(t)}\right)^2 + 2P_I^{(t)}P_T^{(t)} + \left(P_T^{(t)}\right)^2}$$

$$E\left\{n_{TT} \mid n_C, n_I, n_T, n_U, P^{(t)}\right\} = n_{TT}^{(t)} = n_T + \frac{n_U\left(P_T^{(t)}\right)^2}{\left(P_I^{(t)}\right)^2 + 2P_I^{(t)}P_T^{(t)} + \left(P_T^{(t)}\right)^2}$$

Q function:

$$
\begin{aligned}
Q\left(P \mid P^{(t)}\right) = & n_{CC}^{(t)} \log\left\{{P_C}^2\right\} + n_{CI}^{(t)} \log\left\{2P_C P_I\right\} + n_{CT}^{(t)} \log\left\{2P_C P_T\right\} \\
& + n_{II}^{(t)} \log\left\{{P_I}^2\right\} + n_{IT}^{(t)} \log\left\{2P_I P_T\right\} + n_{TT}^{(t)} \log\left\{{P_T}^2\right\} \\
& + k(n_C, n_I, n_T, P^{(t)})
\end{aligned}
$$

Derive the Q function:

$$
\begin{aligned}
\frac{dQ\left(P \mid P^{(t)}\right)}{dP_C} &= \frac{2n_{CC}^{(t)} + n_{CI}^{(t)} + n_{CT}^{(t)}}{P_C} - \frac{2n_{TT}^{(t)} + n_{CT}^{(t)} + n_{IT}^{(t)}}{1 - P_C - P_I} = 0 \\
\frac{dQ\left(P \mid P^{(t)}\right)}{dP_I} &= \frac{2n_{II}^{(t)} + n_{IT}^{(t)} + n_{CI}^{(t)}}{P_I} - \frac{2n_{TT}^{(t)} + n_{CT}^{(t)} + n_{IT}^{(t)}}{1 - P_C - P_I} = 0
\end{aligned}
$$

Update function:

$$
\begin{aligned}
P_C^{(t+1)} &= \frac{2n_{CC}^{(t)} + n_{CI}^{(t)} + n_{CT}^{(t)}}{2n} \\
P_I^{(t+1)} &= \frac{2n_{II}^{(t)} + n_{IT}^{(t)} + n_{CI}^{(t)}}{2n} \\
P_T^{(t+1)} &= \frac{2n_{TT}^{(t)} + n_{CT}^{(t)} + n_{IT}^{(t)}}{2n}
\end{aligned}
$$

(b)

Set the initial values.

```r
nc <- 85 ##carbonaria
ni <- 196 ##insularia
nt <- 341 ##typica
nu <- 578 ##insularia or typica
n <- nc + ni + nt + nu ##total
```

Define the conditional expectation function.

```r
expectation <- function(P_t){
  Pc <- P_t[1]; Pi <- P_t[2]; Pt <- P_t[3]

  ##carbonaria
  ncc <- nc * Pc ^ 2 / (Pc ^ 2 + 2 * Pc * Pi + 2 * Pc * Pt)
  nci <- 2 * nc * Pc * Pi / (Pc ^ 2 + 2 * Pc * Pi + 2 * Pc * Pt)
  nct <- 2 * nc * Pc * Pt / (Pc ^ 2 + 2 * Pc * Pi + 2 * Pc * Pt)

  ##insularia
  nii <- ni * Pi ^ 2 / (Pi ^ 2 + 2 * Pi * Pt) +
```

```
    nu * Pi ^ 2/ (Pi ^ 2 + 2 * Pi * Pt + Pt ^ 2)
  nit <- 2 * ni * Pi * Pt / (Pi ^ 2 + 2 * Pi * Pt) +
    2 * nu * Pi * Pt / (Pi ^ 2 + 2 * Pi * Pt + Pt ^ 2)

  ##typica
  ntt <- nt + nu * Pt ^ 2 / (Pi ^ 2 + 2 * Pi * Pt + Pt ^ 2)

  return (c(ncc, nci, nct, nii, nit, ntt))
}
```

Define the update function of EM.

```
updateP_EM <- function(P_t){
  ne <- expectation(P_t)

  ##update
  Pc_1 <- (2 * ne[1] + ne[2] + ne[3]) / (2 * n)
  Pi_1 <- (2 * ne[4] + ne[5] + ne[2]) / (2 * n)
  Pt_1 <- (2 * ne[6] + ne[3] + ne[5]) / (2 * n)

  return (c(Pc_1, Pi_1, Pt_1))
}
```

Define the `EM Algorithm` function.

```
EM_Algorithm <- function(start, criterion = 1e-6){

  ###INITIAL VALUES###
  P <- c(start, 1 - start[1] - start[2]); count <- 1

  ###MAIN###
  while (sqrt(sum(((updateP_EM(P) - P) / P) ^ 2)) >= criterion){
    P <- updateP_EM(P)
    count <- count + 1
  }
  P <- updateP_EM(P)

  ###OUTPUT###
  structure(list(P = P, itertime = count))
}
```

Compute the MLEs.

```
EM_Algorithm(c(1/3, 1/3))
```

```
## $P
## [1] 0.03606708 0.19579933 0.76813359
##
## $itertime
## [1] 27
```

(f)

Likelihood function of observed data:

$$\begin{aligned}L(P \mid X) =& \frac{n!}{n_C!n_I!n_T!n_U!}\left({P_C}^2 + 2P_CP_I + 2P_CP_T\right)^{n_C}\left({P_I}^2 + 2P_IP_T\right)^{n_I} \\ & \left({P_T}^2\right)^{n_T}\left({P_I}^2 + 2P_IP_T + {P_T}^2\right)^{n_U}\end{aligned}$$

Log likelihood function of observed data:

$$\begin{aligned}\ell(P \mid X) =& n_C \log\left\{P_C + 2P_CP_I + 2P_CP_T\right\} + n_I \log\left\{{P_I}^2 + 2P_IP_T\right\} + n_T \log\left\{{P_T}^2\right\} \\ & + n_U \log\left\{{P_I}^2 + 2P_IP_T + {P_T}^2\right\} + log\binom{n}{n_C \quad n_I \quad n_T \quad n_U}\end{aligned}$$

The approximation to the first derivative of log likelihood function:

$$\begin{aligned}\frac{d^2Q(P \mid P^{(t)})}{dP_CdP_C} &= -\frac{2n_{CC}^{(t)} + n_{CI}^{(t)} + n_{CT}^{(t)}}{{P_C}^2} - \frac{2n_{TT}^{(t)} + n_{CT}^{(t)} + n_{IT}^{(t)}}{\left(1 - P_C - P_I\right)^2} \\ \frac{d^2Q(P \mid P^{(t)})}{dP_CdP_I} &= -\frac{2n_{TT}^{(t)} + n_{CT}^{(t)} + n_{IT}^{(t)}}{\left(1 - P_C - P_I\right)^2} \\ \frac{d^2Q(P \mid P^{(t)})}{dP_IdP_I} &= -\frac{2n_{II}^{(t)} + n_{IT}^{(t)} + n_{CI}^{(t)}}{{P_I}^2} - \frac{2n_{TT}^{(t)} + n_{CT}^{(t)} + n_{IT}^{(t)}}{\left(1 - P_C - P_I\right)^2} \\ Q''(P \mid P^{(t)}) &= \begin{bmatrix} \frac{d^2Q(P|P^{(t)})}{dP_CdP_C} & \frac{d^2Q(P|P^{(t)})}{dP_CdP_I} \\ \frac{d^2Q(P|P^{(t)})}{dP_IdP_C} & \frac{d^2Q(P|P^{(t)})}{dP_IdP_I}\end{bmatrix} \\ \ell'(P \mid X) &= -Q''(P \mid P^{(t)})(\theta_{EM}^{(t+1)} - \theta^{(t)})\end{aligned}$$

First derivative of log likelihood function of observed data:

$$\begin{aligned}\frac{d\ell(P \mid X)}{dP_C} &= \frac{2n_C(1 - P_C)}{2P_C - {P_C}^2} - \frac{2n_I}{2 - P_I - 2P_C} - \frac{2n_T}{1 - P_C - P_I} - \frac{2n_U}{1 - P_C} \\ \frac{d\ell(P \mid X)}{dP_I} &= \frac{2n_I(1 - P_C - P_I)}{2P_I - {P_I}^2 - 2P_IP_C} - \frac{2n_T}{1 - P_C - P_I} \\ \ell'(P \mid X) &= \left[\frac{d\ell(P \mid X)}{dP_C}, \frac{d\ell(P \mid X)}{dP_I}\right]^\top\end{aligned}$$

Second derivative of log likelihood function of observed data:

$$\frac{d^2\ell(P\mid X)}{d{P_C}^2} = -\frac{4n_C(1-P_C)^2}{(2P_C-{P_C}^2)^2} - \frac{2n_C}{2P_C-{P_C}^2} - \frac{4n_I}{(2-P_I-2P_C)^2} - \frac{2n_T}{(1-P_C-P_I)^2} - \frac{2n_U}{(1-P_C)^2}$$

$$\frac{d^2\ell(P\mid X)}{dP_C dP_I} = -\frac{2n_I}{(2-P_I-2P_C)^2} - \frac{2n_T}{(1-P_C-P_I)^2}$$

$$\frac{d^2\ell(P\mid X)}{d{P_I}^2} = -\frac{4n_I(1-P_C-P_I)^2}{(2P_I-{P_I}^2-2P_IP_C)^2} - \frac{2n_I}{2P_I-{P_I}^2-2P_IP_C} - \frac{2n_T}{(1-P_C-P_I)^2}$$

$$\ell''(P\mid X) = \begin{bmatrix} \frac{d^2\ell(P|X)}{d{P_C}^2} & \frac{d^2\ell(P|X)}{dP_C dP_I} \\ \frac{d^2\ell(P|X)}{dP_I dP_C} & \frac{d^2\ell(P|X)}{d{P_I}^2} \end{bmatrix}$$

Update function of Newton Raphson method:

$$\begin{aligned} P^{(t+1)} =& P^{(t)} - \alpha \ell''(P^{(t)}\mid X)^{-1}\ell'(P^{(t)}\mid X) \\ =& P^{(t)} + \alpha \ell''(P^{(t)}\mid X)^{-1} Q''(P^{(t)}\mid X)(\theta_{EM}^{(t+1)} - \theta^{(t)}) \end{aligned}$$

Sometimes we cannot compute $\ell'(P^{(t)}\mid X)$ and $\ell''(P^{(t)}\mid X)$, so we need to do some approximations, such as using Louis method to approximate $\ell''(P^{(t)}\mid X)$ and using $Q'(P\mid P^{(t)})\mid_{P=P^{(t)}}$ to approximate $\ell'(P^{(t)}\mid X)$. In this problem, the latter method is applied and the former is unnecessary because we can write and derive the log likelihood function of observed data.

Define the update function of Newton Raphson method in Aitken Acceleration.

```
updateP_NR <- function(P_t, P_EM, alpha){
  Pc <- P_t[1]; Pi <- P_t[2]; Pt <- P_t[3]
  ne <- expectation(P_t)

  ##approximation to first derivative of log likelihood function
  dqdpcpc <- - (2 * ne[1] + ne[2] + ne[3]) / (Pc ^ 2) -
    (2 * ne[6] + ne[3] + ne[5]) / (Pt ^ 2)
  dqdpcpi <- - (2 * ne[6] + ne[3] + ne[5]) / (Pt ^ 2)
  dqdpipi <- - (2 * ne[4] + ne[5] + ne[2]) / (Pi ^ 2) -
    (2 * ne[6] + ne[3] + ne[5]) / (Pt ^ 2)
  q_2 <- matrix(c(dqdpcpc, dqdpcpi, dqdpcpi, dqdpipi), ncol = 2)

  ##second derivative of log likelihood function
  dldpcpc <- - 4 * nc * ((1 - Pc) ^ 2) / ((2 * Pc - Pc ^ 2) ^ 2) -
    2 * nc / (2 * Pc - Pc ^ 2) - 4 * ni / ((2 - Pi - 2 * Pc) ^ 2) -
    2 * nt / (Pt ^ 2) - 2 * nu / ((1 - Pc) ^ 2)
  dldpcpi <- - 2 * ni / ((2 - Pi - 2 * Pc) ^ 2) - 2 * nt / (Pt ^ 2)
  dldpipi <- - 4 * ni * (Pt ^ 2) / ((Pi ^ 2 + 2 * Pi * Pt) ^ 2) -
```

```
    2 * ni / (Pi ^ 2 + 2 * Pi * Pt) - 2 * nt / (Pt ^ 2)
  logL_2 <- matrix(c(dldpcpc, dldpcpi, dldpcpi, dldpipi), ncol = 2)

  ##output
  P_1 <- P_t[1:2] + alpha * solve(logL_2) %*% q_2 %*% (P_EM[1:2] - P_t[1:2])
  P_1 <- c(P_1, 1 - sum(P_1))
  return (P_1)
}
```

Define the `Aitken accelerated EM Algorithm` function (with step halving).

```
EM_Aitken <- function(start, alpha = 1, criterion = 1e-6){

  ###INITIAL VALUES###
  P <- c(start, 1 - start[1] - start[2]); count <- 1

  ###FUNCTIONS###
  ##Constant term is removed since it won't affect the maximization
  logL <- function(P){
    Pc <- P[1]; Pi <- P[2]; Pt <- P[3]
    out <- nc * log(2 * Pc - Pc ^ 2) + ni * log(Pi ^ 2 + 2 * Pi * Pt) +
      2 * nt * log(Pt) + 2 * nu * log(1 - Pc)
    return (out)
  }

  ###MAIN###
  L <- logL(P); L_1 <- -Inf
  while (sqrt(sum(((updateP_EM(P) - P) / P) ^ 2)) >= criterion){
    ##EM step
    P_EM <- updateP_EM(P)
    ##Newton step
    P_NR <- updateP_NR(P, P_EM, alpha)
    if (sum(P_NR > 0) == 3 & sum(P_NR < 1) == 3){L_1 <- logL(P_NR)}
    ##Step halving
    while (sum(P_NR > 0) != 3 | sum(P_NR < 1) != 3 | L_1 < L){
      alpha <- alpha / 2
      P_NR <- updateP_NR(P, P_EM, alpha)
      if (sum(P_NR > 0) == 3 & sum(P_NR < 1) == 3){L_1 <- logL(P_NR)}
    }
    P <- P_NR
```

```
    L <- L_1
    count <- count + 1
  }
  P <- updateP_EM(P)

  ###OUTPUT###
  structure(list(P = P, itertime = count))
}
```

Compute the MLEs.

```
EM_Aitken(c(1/3, 1/3))
```

```
## $P
## [1] 0.03606708 0.19579910 0.76813381
##
## $itertime
## [1] 5
```

As we can see, the iteration time of Aitken accelerated EM is only 5, which is far smaller than that of classic EM algorithm (27).

Digression

This part is somehow uncorrelated to the homework.

When I was working on this problem, one question came up that since we can figure out the first and second derivative of log likelihood function, what will happen if we use `Newton Raphson method` to solve this problem. So I tried to apply that to this problem.

Define the `Newton Raphson method` function.

```
Newton_Raphson <- function(start, itertime = 100, criterion = 1e-6){

    ###INITIAL VALUES###
    P <- c(start, 1 - start[1] - start[2]); count <- 1

    ###FUNCTIONS###
    logL_1 <- function(P_t){
      Pc <- P_t[1]; Pi <- P_t[2]; Pt <- P_t[3]
      dldpc <- 2 * nc * (1 - Pc) / (2 * Pc - Pc ^ 2) - 2 * ni / (2 - Pi - 2 * Pc) -
        2 * nt / (1 - Pc - Pi) - 2 * nu / (1 - Pc)
      dldpi <- 2 * ni * Pt / (Pi ^ 2 + 2 * Pi * Pt) - 2 * nt / Pt
      out <- matrix(c(dldpc, dldpi), ncol = 1)
      return (out)
    }
    logL_2 <- function(P_t){
      Pc <- P_t[1]; Pi <- P_t[2]; Pt <- P_t[3]
      dldpcpc <- - 4 * nc * ((1 - Pc) ^ 2) / ((2 * Pc - Pc ^ 2) ^ 2) -
        2 * nc / (2 * Pc - Pc ^ 2) - 4 * ni / ((2 - Pi - 2 * Pc) ^ 2) -
        2 * nt / (Pt ^ 2) - 2 * nu / ((1 - Pc) ^ 2)
      dldpcpi <- - 2 * ni / ((2 - Pi - 2 * Pc) ^ 2) - 2 * nt / (Pt ^ 2)
      dldpipi <- - 4 * ni * (Pt ^ 2) / ((Pi ^ 2 + 2 * Pi * Pt) ^ 2) -
        2 * ni / (Pi ^ 2 + 2 * Pi * Pt) - 2 * nt / (Pt ^ 2)
      out <- matrix(c(dldpcpc, dldpcpi, dldpcpi, dldpipi), ncol = 2)
      return (out)
    }
    delta <- function(P){solve(logL_2(P)) %*% logL_1(P)}
    epsilon <- function(x){sqrt(sum(x ^ 2))}

    ###MAIN###
    for (i in 1:itertime){
      d <- delta(P)
```

```r
      P[1:2] <- P[1:2] - d
      P[3] <- 1 - sum(P[1:2])
      if (epsilon(d) <= criterion){break}
      count <- count + 1
    }

    ###OUTPUT###
    structure(list(P = P, itertime = count))
}
```

Compute the MLEs.

```r
Newton_Raphson(c(1/3, 1/3)); Newton_Raphson(c(0.036, 0.19))
```

```
## $P
## [1] -5.225267e+29  1.813310e+29  3.411958e+29
##
## $itertime
## [1] 101
```

```
## $P
## [1] 0.03606708 0.19579915 0.76813377
##
## $itertime
## [1] 3
```

It turns out that only when the starting value is close to true value can the algorithm converge, but `EM algorithm` can always converge given any starting value. Therefore, `EM algorithm` may be a better choice for the problems with missing data or with restrict on unknown parameters.

4.2

(a)

Unknown parameters of interest:

$$\theta = (\alpha, \beta, \mu, \lambda)$$

Observed data:

$$X = (n_0, n_1, ..., n_{16}) \qquad N = \sum_{i=0}^{16} n_i$$

Complete data:

$$Y = (n_{z,0}, n_{t,0}, n_{t,1}, ..., n_{t,16}, n_{p,0}, n_{p,1}, .., n_{p,16})$$

Likelihood function of complete data:

$$L(\theta \mid Y) = \alpha^{n_{z,0}} \prod_{i=0}^{16} \left(\beta \frac{\mu^i e^{-\mu}}{i!}\right)^{n_{t,i}} \prod_{i=0}^{16} \left((1-\alpha-\beta)\frac{\lambda^i e^{-\lambda}}{i!}\right)^{n_{p,i}}$$

Log likelihood function of complete data:

$$\begin{aligned}
\ell(Y \mid \theta) =& n_{z,0} \log \alpha + \sum_{i=0}^{16} n_{t,i} \{\log \beta + i \log \mu - \mu - \log i!\} \\
&+ \sum_{i=0}^{16} n_{p,i} \{\log(1-\alpha-\beta) + i \log \lambda - \lambda - \log i!\} \\
=& n_{z,0} \log \alpha + \sum_{i=0}^{16} n_{t,i} \log \beta + \sum_{i=0}^{16} n_{t,i} \{i \log \mu - \mu\} + \sum_{i=0}^{16} n_{p,i} \log(1-\alpha-\beta) \\
&+ \sum_{i=0}^{16} n_{p,i} \{i \log \lambda - \lambda\} - \underbrace{\left\{\sum_{i=0}^{16} \left(n_{t,i} + n_{p,i}\right) \log i!\right\}}_{\text{constant}}
\end{aligned}$$

Conditional expectations:

$$\begin{aligned}
&\pi_i(\theta) = \alpha 1_{\{i=0\}} + \beta \mu^i e^{-\mu} + (1-\alpha-\beta)\lambda^i e^{-\lambda} \\
&z_0(\theta) = \frac{\alpha}{\pi_0(\theta)} \\
&t_i(\theta) = \frac{\beta \mu^i e^{-\mu}}{\pi_i(\theta)} \\
&p_i(\theta) = \frac{(1-\alpha-\beta)\lambda^i e^{-\lambda}}{\pi_i(\theta)} \\
&E\left\{n_{z,0} \mid n_i, \theta^{(t)}\right\} = n_{z,0}^{(t)} = n_0 z_0(\theta^{(t)}) \\
&E\left\{n_{t,i} \mid n_i, \theta^{(t)}\right\} = n_{t,i}^{(t)} = n_i t_i(\theta^{(t)}) \\
&E\left\{n_{p,i} \mid n_i, \theta^{(t)}\right\} = n_{p,i}^{(t)} = n_i p_i(\theta^{(t)})
\end{aligned}$$

Q function:

$$
\begin{aligned}
Q\left(\theta \mid \theta^{(t)}\right) = & n_{z,0}^{(t)} \log \alpha + \sum_{i=0}^{16} n_{t,i}^{(t)} \log \beta + \sum_{i=0}^{16} n_{t,i}^{(t)} \{i \log \mu - \mu\} \\
& + \sum_{i=0}^{16} n_{p,i}^{(t)} \log(1-\alpha-\beta) + \sum_{i=0}^{16} n_{p,i}^{(t)} \{i \log \lambda - \lambda\} + k\left(n_{t,i}, n_{p,i}, \theta^{(t)}\right)
\end{aligned}
$$

Derive the Q function:

$$
\begin{aligned}
\frac{dQ\left(\theta \mid \theta^{(t)}\right)}{d\alpha} &= \frac{n_{z,0}^{(t)}}{\alpha} - \frac{\sum_{i=0}^{16} n_{p,i}^{(t)}}{1-\alpha-\beta} = 0 \\
\frac{dQ\left(\theta \mid \theta^{(t)}\right)}{d\beta} &= \frac{\sum_{i=0}^{16} n_{t,i}^{(t)}}{\beta} - \frac{\sum_{i=0}^{16} n_{p,i}^{(t)}}{1-\alpha-\beta} = 0 \\
\frac{dQ\left(\theta \mid \theta^{(t)}\right)}{d\mu} &= \sum_{i=0}^{16} n_{t,i}^{(t)} \left(\frac{i}{\mu} - 1\right) = 0 \\
\frac{dQ\left(\theta \mid \theta^{(t)}\right)}{d\lambda} &= \sum_{i=0}^{16} n_{p,i}^{(t)} \left(\frac{i}{\lambda} - 1\right) = 0
\end{aligned}
$$

Update function:

$$
\begin{aligned}
\alpha^{(t+1)} &= \frac{n_0 z_0\left(\theta^{(t)}\right)}{N} \\
\beta^{(t+1)} &= \sum_{i=0}^{16} \frac{n_i t_i\left(\theta^{(t)}\right)}{N} \\
\mu^{(t+1)} &= \frac{\sum_{i=0}^{16} i n_i t_i\left(\theta^{(t)}\right)}{\sum_{i=0}^{16} n_i t_i\left(\theta^{(t)}\right)} \\
\lambda^{(t+1)} &= \frac{\sum_{i=0}^{16} i n_i p_i\left(\theta^{(t)}\right)}{\sum_{i=0}^{16} n_i p_i\left(\theta^{(t)}\right)}
\end{aligned}
$$

(b)

Set the initial values.

```r
n <- c(379, 299, 222, 145, 109, 95, 73, 59, 45, 30, 24, 12, 4, 2, 0, 1, 1)
N <- sum(n)
i <- 0:16
```

Define the update function.

```r
update_theta <- function(theta, n){
  alpha <- theta[1]; beta <- theta[2]; mu <- theta[3]; lambda <- theta[4]
  pi_i <- alpha * ifelse(i == 0, 1, 0) + beta * (mu ^ i) * exp(- mu) +
    (1 - alpha - beta) * (lambda ^ i) * exp(- lambda)
```

```
  ##probabilities of i risky encounters belong to the various groups
  z_0 <- alpha / pi_i[0 + 1]
  t_i <- beta * (mu ^ i) * exp(- mu) / pi_i
  p_i <- (1 - alpha - beta) * (lambda ^ i) * exp(- lambda) / pi_i

  ##compute the updated parameters
  alpha_1 <- n[0 + 1] * z_0 / N
  beta_1 <- sum(n * t_i / N)
  mu_1 <- sum(i * n * t_i) / sum(n * t_i)
  lambda_1 <- sum(i * n * p_i) / sum(n * p_i)

  ##output
  return (c(alpha_1, beta_1, mu_1, lambda_1))
}
```

Define the `EM Algorithm` function.

```
EM_Algorithm <- function(start, n, criterion = 1e-6){

  ###INITIAL VALUES###
  theta <- start; count <- 1

  ###MAIN###
  while (sqrt(sum(((update_theta(theta, n) - theta) / theta) ^ 2)) >= criterion){
    theta <- update_theta(theta, n)
    count <- count + 1
  }
  theta <- update_theta(theta, n)

  ###OUTPUT###
  structure(list(theta = theta, itertime = count))
}
```

Estimate the parameters of the model. Notice that μ is the parameter of typical group, while λ is the parameter of high-risk group, so it is natural that λ should be larger than μ (number of risky encounters in high-risk group should be larger than that in typical group). That's why I set the starting value of λ larger than μ.

```
EM_Algorithm(c(0.2, 0.4, 1, 5), n)
```

```
## $theta
## [1] 0.1221650 0.5625419 1.4674679 5.9388805
##
## $itertime
## [1] 118
```

(c)

I select two methods to estimate standard errors and pairwise correlations.

1.Bootstrap

Define the `Bootstrap` function.

```
Bootstrap <- function(t){
  sample_p <- n / N
  theta <- matrix(NA, t, 4)

  ##bootstrap for t times
  for (j in 1:t){
    ##sample pseudo-data at random with replacement
    temp <- sample(i, N, replace = TRUE, prob = sample_p)
    sample_n <- rep(NA, 17)
    for (k in i){sample_n[k + 1] <- length(which(temp == k))}
    ##calculate thetaj
    theta[j, ] <- EM_Algorithm(c(0.2, 0.4, 1, 5), sample_n)$theta
  }

  ##output
  out <- cov(theta)
  colnames(out) = rownames(out) = c('alpha', 'beta', 'mu', 'lambda')
  return (out)
}
```

Estimate the covariance matrix of parameter estimates.

```
set.seed(520)
co <- Bootstrap(10000); co
```

```
##                alpha          beta           mu      lambda
## alpha   0.0004320598 -2.018252e-04 1.738019e-03 0.001563121
## beta   -0.0002018252  4.786162e-04 3.856861e-05 0.001506322
## mu      0.0017380191  3.856861e-05 1.299612e-02 0.014093902
## lambda  0.0015631215  1.506322e-03 1.409390e-02 0.039763108
```

As a result, the standard errors of $(\alpha, \beta, \mu, \lambda)$ are respectively:

```r
se <- round(c(sqrt(co[1, 1]), sqrt(co[2, 2]), sqrt(co[3, 3]), sqrt(co[4, 4])), 5)
paste(se, collapse = ', ')
```

```
## [1] "0.02079, 0.02188, 0.114, 0.19941"
```

The pairwise correlation matrix is:

```r
corr <- co
for(j in 1:4){for(k in 1:4){corr[j, k] <- co[j, k] / (sqrt(co[j, j] * co[k, k]))}}
corr
```

```
##             alpha        beta          mu    lambda
## alpha   1.0000000 -0.44382294 0.73345895 0.3771210
## beta   -0.4438229  1.00000000 0.01546441 0.3452899
## mu      0.7334589  0.01546441 1.00000000 0.6199894
## lambda  0.3771210  0.34528994 0.61998940 1.0000000
```

2.Empirical Information

Individual scores for each observation:

$$
\begin{aligned}
&\pi_i(\theta) = \alpha 1_{\{i=0\}} + \beta \mu^i e^{-\mu} + (1-\alpha-\beta)\lambda^i e^{-\lambda} \\
&L(\theta \mid x_i) = \left[\frac{\pi_i(\theta)}{i!}\right]^{n_i} \\
&\ell(\theta \mid x_i) = n_i \log(\pi_i(\theta)) - n_i \log(i!) \\
&\frac{d\ell(\theta \mid x_i)}{d\alpha} = \frac{n_i(1_{\{i=0\}} - \lambda^i e^{-\lambda})}{\alpha 1_{\{i=0\}} + \beta \mu^i e^{-\mu} + (1-\alpha-\beta)\lambda^i e^{-\lambda}} \\
&\frac{d\ell(\theta \mid x_i)}{d\beta} = \frac{n_i(\mu^i e^{-\mu} - \lambda^i e^{-\lambda})}{\alpha 1_{\{i=0\}} + \beta \mu^i e^{-\mu} + (1-\alpha-\beta)\lambda^i e^{-\lambda}} \\
&\frac{d\ell(\theta \mid x_i)}{d\mu} = \frac{n_i \beta e^{-\mu} \mu^{i-1}(i-\mu)}{\alpha 1_{\{i=0\}} + \beta \mu^i e^{-\mu} + (1-\alpha-\beta)\lambda^i e^{-\lambda}} \\
&\frac{d\ell(\theta \mid x_i)}{d\lambda} = \frac{n_i (1-\alpha-\beta) e^{-\lambda} \lambda^{i-1}(i-\lambda)}{\alpha 1_{\{i=0\}} + \beta \mu^i e^{-\mu} + (1-\alpha-\beta)\lambda^i e^{-\lambda}} \\
&\ell'(\theta \mid x_i) = \left[\frac{d\ell(\theta \mid x_i)}{d\alpha}, \frac{d\ell(\theta \mid x_i)}{d\beta}, \frac{d\ell(\theta \mid x_i)}{d\mu}, \frac{d\ell(\theta \mid x_i)}{d\lambda}\right]^\top
\end{aligned}
$$

Total scores for adjustment:

$$
\begin{aligned}
L(\theta \mid X) &= \prod_{i=0}^{16} \left[\frac{\pi_i(\theta)}{i!}\right]^{n_i} \\
\ell(\theta \mid X) &= \sum_{i=0}^{16} n_i \log(\pi_i(\theta)) - \sum_{i=0}^{16} n_i \log(i!) \\
\frac{d\ell(\theta \mid X)}{d\alpha} &= \sum_{i=0}^{16} \frac{n_i(1_{\{i=0\}} - \lambda^i e^{-\lambda})}{\alpha 1_{\{i=0\}} + \beta\mu^i e^{-\mu} + (1-\alpha-\beta)\lambda^i e^{-\lambda}} \\
\frac{d\ell(\theta \mid X)}{d\beta} &= \sum_{i=0}^{16} \frac{n_i(\mu^i e^{-\mu} - \lambda^i e^{-\lambda})}{\alpha 1_{\{i=0\}} + \beta\mu^i e^{-\mu} + (1-\alpha-\beta)\lambda^i e^{-\lambda}} \\
\frac{d\ell(\theta \mid X)}{d\mu} &= \sum_{i=0}^{16} \frac{n_i \beta e^{-\mu} \mu^{i-1}(i-\mu)}{\alpha 1_{\{i=0\}} + \beta\mu^i e^{-\mu} + (1-\alpha-\beta)\lambda^i e^{-\lambda}} \\
\frac{d\ell(\theta \mid X)}{d\lambda} &= \sum_{i=0}^{16} \frac{n_i(1-\alpha-\beta) e^{-\lambda} \lambda^{i-1}(i-\lambda)}{\alpha 1_{\{i=0\}} + \beta\mu^i e^{-\mu} + (1-\alpha-\beta)\lambda^i e^{-\lambda}} \\
\ell'(\theta \mid X) &= \left[\frac{d\ell(\theta \mid X)}{d\alpha}, \frac{d\ell(\theta \mid X)}{d\beta}, \frac{d\ell(\theta \mid X)}{d\mu}, \frac{d\ell(\theta \mid X)}{d\lambda}\right]^\top
\end{aligned}
$$

Use the empirical information to estimate the covariance matrix.

$$
\begin{aligned}
\hat{Var}(\hat{\theta}) &= \left[\hat{I}(\hat{\theta})\right]^{-1} \\
\hat{I}(\hat{\theta}) &= \left[\frac{1}{n}\sum_{i=1}^{n} \ell'(\theta \mid x_i)\, \ell'(\theta \mid x_i)^\top - \frac{1}{n^2} \ell'(\theta \mid X)\, \ell'(\theta \mid X)^\top\right]\Bigg|_{\theta=\hat{\theta}}
\end{aligned}
$$

Define the `Empirical Information` function.

```
Empirical_Information <- function(theta){
  alpha <- theta[1]; beta <- theta[2]; mu <- theta[3]; lambda <- theta[4]
  pi_i <- alpha * ifelse(i == 0, 1, 0) + beta * (mu ^ i) * exp(- mu) +
    (1 - alpha - beta) * (lambda ^ i) * exp(- lambda)

  ##define individual scores for each observation
  dldalpha <- n * (c(1, rep(0, 16)) - lambda ^ i * exp(- lambda)) / pi_i
  dldbeta <- n * (mu ^ i * exp(- mu) - lambda ^ i * exp(- lambda)) / pi_i
  dldmu <- n * beta * exp(- mu) * mu ^ (i - 1) * (i - mu) / pi_i
  dldlambda <- n * (1 - alpha - beta) * exp(- lambda) * lambda ^ (i - 1) *
    (i - lambda) / pi_i

  ##output
  out <- matrix(0, 4, 4)
  for (j in i){
    l_i <- c(dldalpha[j + 1], dldbeta[j + 1], dldmu[j + 1], dldlambda[j + 1])
```

```r
    out <- out + l_i %*% t(l_i)}
  l <- c(sum(dldalpha), sum(dldbeta), sum(dldmu), sum(dldlambda))
  out <- out / length(n) - l %*% t(l) / (length(n) ^ 2)
  return (out)
}
```

Compute the empirical information matrix and estimate the covariance matrix.

```r
result <- EM_Algorithm(c(0.2, 0.4, 1, 5), n)
ei <- Empirical_Information(result$theta)
colnames(ei) = rownames(ei) = c('alpha', 'beta', 'mu', 'lambda')
co <- solve(ei); co
```

```
##                alpha          beta           mu       lambda
## alpha   2.343451e-05 -2.385271e-06 1.216210e-04 0.0001849732
## beta   -2.385271e-06  4.784850e-05 8.442457e-05 0.0003515634
## mu      1.216210e-04  8.442457e-05 1.108191e-03 0.0022303707
## lambda  1.849732e-04  3.515634e-04 2.230371e-03 0.0124838166
```

As a result, the standard errors of $(\alpha, \beta, \mu, \lambda)$ are respectively:

```r
se <- round(c(sqrt(co[1, 1]), sqrt(co[2, 2]), sqrt(co[3, 3]), sqrt(co[4, 4])), 5)
paste(se, collapse = ', ')
```

```
## [1] "0.00484, 0.00692, 0.03329, 0.11173"
```

The pairwise correlation matrix is:

```r
corr <- co
for(j in 1:4){for(k in 1:4){corr[j, k] <- co[j, k] / (sqrt(co[j, j] * co[k, k]))}}
corr
```

```
##              alpha        beta        mu    lambda
## alpha   1.00000000 -0.07123209 0.7546983 0.3419851
## beta   -0.07123209  1.00000000 0.3666296 0.4548789
## mu      0.75469826  0.36662958 1.0000000 0.5996477
## lambda  0.34198514  0.45487893 0.5996477 1.0000000
```

Compare the results of two methods.

- The standard error of bootstrap is larger than that of empirical information.
- The correlation matrix are quite similar except the correlation between β and α, μ, λ.

Great differences emerge, but I don't know it's a commonplace or it's because there's something wrong with my code that I fail to find. Could my dear TA tell me the answer? o(T ﹏ T)o