CHL5224 Assignment 1

Fangming Liao, 1001374997

October 9, 2018

1.Introduction

In the ABO-blood example, we have 3 alleles: A, B, O and 6 genotypes: AA, AO, BB, BO, AB, and OO, and the genotype determines the blood type: A, B are dominant to O, O is recessive to A, B, and A, B are co-dominant. In other words, genotype AA, AO determines blood type A, genotype BB, BO determines blood type B, genotype AB determines blood type AB, and genotype OO determines blood type O.

In a large random sample from Berlin (Bernstein 1925, Sham's book page 44), the genotypes of participates were recorded: $n_A = 9133$ for blood type A, $n_B = 2987$ for blood type B, $n_{AB} = 1269$ for blood type AB, and $n_O = 7725$ for blood type O.

The question is how to estimate the allele frequencies of alleles A, B and O.

Let

$$p = freq(alleleA),$$
$$q = freq(alleleB),$$
$$1 - p - q = freq(alleleO).$$

Assuming HWE, we can represent the frequencies of 4 phenotypes using p and q:

$$freq(phenotype\ A) = freq(genotype\ AA) + freq(genotype\ AO) = p^2 + 2p(1-p-q)$$

$$freq(phenotype\ B) = freq(genotype\ BB) + freq(genotype\ BO) = q^2 + 2q(1-p-q)$$

$$freq(phenotype\ AB) = freq(genotype\ AB) = 2pq$$

$$freq(phenotype\ O) = freq(genotype\ OO) = (1-p-q)^2$$

In this particular sample, we can write the Likelihood function:

$$L(p,q) = [p^2 + 2p(1-p-q)]^{n_A}[q^2 + 2q(1-p-q)]^{n_B}(2pq)^{n_{AB}}[(1-p-q)^2]^{n_O} \tag{1}$$

Take log of the Likelihood function, we obtain our log-likelihood function:

$$l(p,q) = ln(L(p,q)) = n_A ln(p^2+2p(1-p-q)) + n_B ln(q^2+2q(1-p-q)) + n_{AB} ln(2pq) + n_O ln((1-p-q)^2) \tag{2}$$

By taking the derivative of the log-likelihood function w.r.t. p and q, we obtain:

$$\frac{\partial l(p,q)}{\partial p} = \frac{2n_A(1-p-q)}{p^2+2p(1-p-q)} - \frac{2n_B}{q+2(1-p-q)} + \frac{n_{AB}}{p} - \frac{2n_O}{1-p-q} \tag{3}$$

$$\frac{\partial l(p,q)}{\partial q} = \frac{2n_A}{p+2q-2} + \frac{2n_B(p+q-1)}{q^2-2q+2pq} + \frac{n_{AB}}{q} - \frac{2n_O}{1-p-q} \tag{4}$$

To get the maximum value of likelihood function, we set equation (3) and (4) to be 0 and solve for $\hat{p}, \hat{q}$. However, there is no closed formed solutions.

There were approximate solution (Bernstein, 1925) based on groupings of phenotypes. Alternatively, we can use numerical iterative approaches such as Newton-Raphson and EM algorithms.

2. Methodology

2.1 Newton-Raphson Algorithm

First, let's assume the allele types are evenly distributed in the population, i.e. $p = q = (1 - p - 1) = \frac{1}{3}$. Then we have the initial values for our iterative algorithm: $p^{(0)} = q^{(0)} = \frac{1}{3}$

Then, for k=1,2,... the updating function is

$$\begin{bmatrix} p^{(k)} \\ q^{(k)} \end{bmatrix} = \begin{bmatrix} p^{(k-1)} \\ q^{(k-1)} \end{bmatrix} - \begin{bmatrix} Dl(p^{(k-1)}, q^{(-1)}) \end{bmatrix}^{-1} \Delta l[p^{(k-1)}, q^{(k-1)}]$$

where

$$Dl[p^{(k-1)}, q^{(k-1)}] = \begin{bmatrix} \frac{\partial^2}{\partial p^2} l(p^{(k-1)}, q^{(k-1)}) & \frac{\partial^2}{\partial p \partial q} l(p^{(k-1)}, q^{(k-1)}) \\ \frac{\partial^2}{\partial q \partial p} l(p^{(k-1)}, q^{(k-1)}) & \frac{\partial^2}{\partial q^2} l(p^{(k-1)}, q^{(k-1)}) \end{bmatrix}$$

By equation (3), (4), we have:

$$\frac{\partial^2}{\partial p^2} l(p,q) = -\frac{2n_A(p^2 + 2p(1-p-q) + 2(1-p-q)^2)}{(p^2 + 2p(1-p-q))^2} - \frac{4n_B}{(q + 2(1-p-q))^2} - \frac{n_{AB}}{p^2} - \frac{2n_O}{(1-p-q)^2}$$

$$\frac{\partial^2}{\partial p \partial q} l(p,q) = \frac{\partial^2}{\partial q \partial p} l(p,q) = -\frac{2n_A}{(p+2q-2)^2} - \frac{2n_B}{(2p+q-2)^2} - \frac{2n_O}{(p+q-1)^2}$$

$$\frac{\partial^2}{\partial q^2} l(p^{(k-1)}, q^{(k-1)}) = -\frac{4n_A}{(p+2q-2)^2} + \frac{2n_B}{2pq + q^2 - 2q} - \frac{4n_B(p+q-1)^2}{q^2(2p+q-2)^2} - \frac{n_{AB}}{q^2} - \frac{2n_O}{(p+q-1)^2}$$

and

$$\Delta l[p^{(k-1)}, q^{(k-1)}] = \begin{bmatrix} \frac{\partial}{\partial p} l(p^{(k-1)}, q^{(k-1)}) \\ \frac{\partial}{\partial q} l(p^{(k-1)}, q^{(k-1)}) \end{bmatrix}$$

We implement this algorithm using R by writing a function as below:

```r
aboMLE <- function(x, p=0.3333, q=0.3333, eps = 1.e-5, max.iter = 100){
  n <- length(x) # x is the vector of observed frequencies of phenotypes

  p.old <- p
  q.old <- q

  no.conv <- T
  iter <- 0

  nA  <- x[1]
  nB  <- x[2]
  nAB <- x[3]
  nO  <- x[4]
```

```r
while(no.conv){

  # computing all the derivatives of previous step

  dldp <- (2*nA*(1-p.old-q.old))/(p.old^2+2*p.old*(1-p.old-q.old)) -
    (2*q.old*nB)/(q.old^2+2*q.old*(1-p.old-q.old)) +
    nAB/p.old -
    2*nO*(1-p.old-q.old)/((1-p.old-q.old)^2)

  dldq <- 2*nA/(p.old+2*q.old-2) +
    2*nB*(p.old+q.old-1)/(q.old^2-2*q.old+2*p.old*q.old)+
    nAB/q.old -
    2*nO/(1-p.old-q.old)

  ddldpp <- nA*(-2*(p.old^2+2*p.old*(1-p.old-q.old))-
    4*(1-p.old-q.old)^2)/((p.old^2+2*p.old*(1-p.old-q.old))^2) -
    (4*nB*q.old^2)/((q.old^2+2*q.old*(1-p.old-q.old))^2) -
    (nO*2*(1-p.old-q.old)^2)/((1-p.old-q.old)^4)

  ddldpq <- -(2*nA)/((p.old+2*q.old-2)^2) +
    (2*nB)/((2*p.old+q.old-2)^2) -
    (2*nO)/((p.old+q.old-1)^2)

  ddldqp <- -(2*nA)/((p.old+2*q.old-2)^2) +
    (2*nB)/((2*p.old+q.old-2)^2) -
    (2*nO)/((p.old+q.old-1)^2)

  ddldqq <- - (4*nA)/((p.old+2*q.old-2)^2) +
    (2*nB)/(2*p.old*q.old+q.old^2-2*q.old) -
    (4*nB*(p.old+q.old-1)^2)/(q.old^2*(2*p.old+q.old-2)^2) -
    (nAB)/(q.old^2) -
    (2*nO)/((p.old+q.old-1)^2)

  # computing the Heisssian matrix
  Dl <- matrix(c(ddldpp, ddldqp, ddldpq, ddldqq), nrow = 2)
  Dl.inv <- solve(Dl)

  # update p and q
  p <- p.old - (Dl.inv[1,1]*dldp + Dl.inv[1,2]*dldq)
  q <- q.old - (Dl.inv[2,1]*dldp + Dl.inv[2,2]*dldq)

  iter <- iter + 1

  # stop the iteration either p and q converges or it reaches maximum number of iteration
  if (max(c(abs(p-p.old), abs(q-q.old))) <= eps) no.conv <- F
  if (iter==max.iter) no.conv <- F

  p.old <- p
  q.old <- q
}

r <- list(p=p, q=q, iter=iter)
r
```

```
}
```

This algorithm stops either the parameter converges, or it reaches the maximum number of iterations.

```r
# run the N-R MLE Algo on given data
set.seed(999)
x <- c(9123, 2987, 1269, 7725)
mle <- aboMLE(x)
```

```
## Finished with 7 iterations

## Estimated p =  0.2876858

## Estimated q =  0.106555
```

2.2 EM Algorithm

Instead of Newton Raphson, we can use the Expectation-Maximization (EM) algorithm, which is a numerical iterative method for finding the Maximum Likelihood Estimates of parameters.

In our ABO-blood problem, we have our observed data: $n_A = n_{AA} + n_{AO}, n_B = n_{BB} + n_{BO}, n_{AB} = n_{AB}, n_O = n_O$.

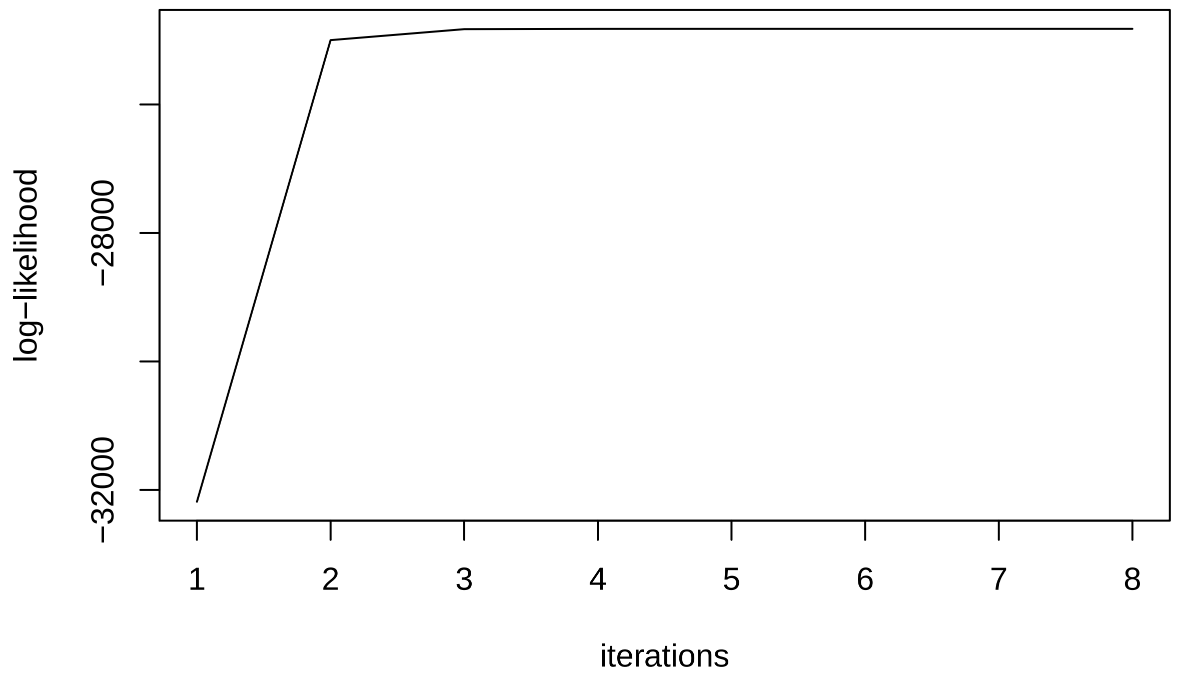
However, to find the maximized likelihood, we need our complete data: $n_{AA}, n_{AO}, n_{BB}, n_{BO}, n_{AB}, n_O$.

To do so, we only need to find our parameter of interest: $p = freq(alleleA), q = freq(alleleB)$ by EM algorithm implemented as follow:

```r
# Suppose we don't have any prior knowlegde about the frequencies,
# and assume alleles A, B, and O are evenly distributed among people,
# so we start the algo from p=q=1-p-q=1/3
aboEM <- function(x, p=0.3333, q=0.3333, em.iter=100, eps=1.e-5){
  n <- length(x) # x is the vector of observed frequencies of phenotypes

  p.old <- p
  q.old <- q

  # observed data
  nA  <- x[1]
  nB  <- x[2]
  nAB <- x[3]
  nO  <- x[4]

  no.conv <- T
  iter <- 0
  logliks <- vector("numeric")

  # initialize values of nAA, nAO, nBB, and nBO with initial values of p and q


  # update p, q and then update values of nAA, nAO, nBB, and nBO
  while(no.conv){

    ### E-step: calculate the expected value of log likelihood
    # estimate the missing data
    nAA <- (p.old*p.old*nA)/(p.old*p.old + 2*p.old*(1-p.old-q.old))
    nAO <- (2*p.old*(1-p.old-q.old)*nA)/(p.old*p.old+2*p.old*(1-p.old-q.old))
    nBB <- (q.old*q.old*nB)/(q.old*q.old + 2*q.old*(1-p.old-q.old))
    nBO <- (2*q.old*(1-p.old-q.old)*nB)/(q.old*q.old+2*q.old*(1-p.old-q.old))


    ### M-step: maximize the log-likelihood
    p <- (2*nAA+nAO+nAB)/(2*sum(x))
    q <- (2*nBB+nBO+nAB)/(2*sum(x))

    ### check if the log-likelihood is maximized, i.e. convergent
    # observed Log-likelihood computation
    if (iter != 0){
      logL.old <- logL
    }
    logL <- nA*log(p.old^2+2*p.old*(1-p.old-q.old))+
      nB*log(q.old^2+2*q.old*(1-p.old-q.old))+
      nAB*log(2*p.old*q.old)+
      nO*log((1-p.old-q.old)^2)

    # append the new log-likelihood to previous ones
    logliks <- c(logliks,logL)

    # iteration stops when the loglikelihood converges
    # or it reaches the limit of iterations
    if (iter != 0){
      if (abs(logL-logL.old) <= eps) no.conv <- F
    }
    if (iter==em.iter) no.conv <- F
    iter <- iter + 1

    p.old <- p
    q.old <- q
  }

  r <- list(p=p, q=q, logL=logliks, iter=iter)
  r
}
```

Different from Newton-Raphson, this algorithm stops either the log-likelihood converges, or it reaches the maximum number of iterations.

```
# run the EM Algo on given data
set.seed(999)
x = c(9123, 2987, 1269, 7725)
em <- aboEM(x)
# plot the loglikelihood by iterations
plot(em$logL, xlab = "iterations", ylab = "log-likelihood", type = "l")
```



```
## Finished with 8 iterations

## Estimated p =  0.2876858

## Estimated q =  0.106555
```

3.Conclusion

Both of the Newton-Raphson and EM algorithm supported the results from an approximate solution based on groupings of phenotypes (Bernstein 1925), which gave:

$$\hat{p} = 0.287552$$

$$\hat{q} = 0.106506$$

In addition, both algorithms converges pretty fast, N-R with 7 iterations, and EM with 8 iterations.

Reference

Bernstein, F., 1925. Zusammenfassende betrachtungen uber die erblichen blutstrukturen des menschen. Mol. General Genet., 37: 237-370.