Solution 5: Statistical inference (II)

Jianhui Gao

July 14, 2025

EM and Newton-Raphson implementation

The ABO-gene or ABO-locus is on chromosome 9. It has 3 alleles (antigens) (A, B, O) and it determines 4 blood type (A, B, AB, O).

We have a large random sample obtained from Berlin (Bernstein 1925, Sham's book page 44):

- $n_A = 9123$ blood type A
- $n_B = 2987$ blood type B
- $n_{AB} = 1269$ blood type AB
- $n_O = 7725$ blood type O

For instance, $n_A = 9123 = n_{AA} + n_{AO}$: Among 9123 blood type A individuals, some have genotype AA and the others have genotype AO.

Our interest is to estimate the allele frequencies of alleles A, B, and O. i.e. $p =$ freq (allele A), $q =$ freq (allele B), $1 - p - q =$ freq (allele O).

1. Write out the log-likelihood $L(p, q)$.
2. Is there a closed-form solution of this log-likelihood function?
3. Formulate the problem as a missing data problem and use the Newton-Raphson algorithm to find the MLEs, $\hat{p}$ and $\hat{q}$, that maximize the log-likelihood, $\ln L(p, q)$.
4. (Advanced) Use the EM algorithm to find the Maximum Likelihood Estimates (MLEs) of parameters, $\hat{p}$ and $\hat{q}$.

Hint: Lei Sun's STA2080 Modern genetic statistics notes (link).

Solution

1. Let $X = (n_A, n_B, n_{AB}, n_O)$, X follows the multinomial distribution,

$$L(p,q) = \binom{n}{n_A, n_B, n_{AB}, n_O} \left(p^2 + 2p(1-p-q)\right)^{n_A} \left(q^2 + 2q(1-p-q)\right)^{n_B} (2pq)^{n_{AB}} \left((1-p-q)^2\right)^{n_O}$$

The general approach to estimate allele frequency is maximum likelihood estimation. To find MLE, take the derivatives of the log-likelihood function, and find the pair of (p, q) that set the derivatives to 0.

The log transformation is performed since finding a maximizer of $L(p, q)$ is equivalent to finding a maximizer of $\ln L(p, q)$. The log-likelihood is,

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

Take the partial derivatives of the log-likelihood and set them to 0,

$$\frac{\partial \ln L(p,q)}{\partial p} = \frac{2(1-p-q)}{p(2-p-2q)} n_A + \frac{2}{2p+q-2} n_B + \frac{1}{p} n_{AB} - \frac{2}{1-p-q} n_O = 0 \quad (2)$$

$$\frac{\partial \ln L(p,q)}{\partial q} = \frac{2}{p+2q-2} n_A + \frac{2(1-p-q)}{q(2-2p-q)} n_B + \frac{1}{q} n_{AB} - \frac{2}{1-p-q} n_O = 0 \quad (3)$$

2. It is hard to find the explicit form of the (p, q) from equation (2) and (3) **directly**. We consider solve the problem **iteratively**.

3. Newton-Raphson algorithm is another method that can be applied to estimate the ABO allele frequency. The algorithm is initially designed to find the roots (or zeros) of a real-valued function iteratively.

In the ABO blood type settings, maximizing the likelihood is equivalent to finding the roots of the derivative of log-likelihood function. Since it is hard to find the roots directly, one can approximate it iteratively. Here denote the parameters to be estimated as $\vec{\theta} = (p, q)$, denote the log-like hood shown in equation (1) as $f(\vec{\theta}) = \ln L(\theta)$, then the partial derivatives of the log-likelihood (score function) is $f'(\vec{\theta}) = f'(p, q)$, the second derivatives (Hessian matrix, or observed information $-I(\theta)$) as $f''(\vec{\theta}) = f''(p, q)$. The explicit forms of the score function and observed information are shown in the appendix.

4. Expectation-Maximum (EM) algorithm is a method for obtaining the Maximum likelihood estimates (MLE) of parameters iteratively. It usually contains two parts: **E-step (expectation)** computes an expected value of the log-likelihood using current estimate for the parameters; **M-step (Maximization)** calculates MLE based on the log likelihood in the E-step, then updates the estimates of the parameters.

The EM algorithms are often used when the model contains unobserved data. In the ABO settings, one could observe the phenotype counts (n_A, n_B, n_{AB}, n_O), while the genotype counts n_{AA} or n_{AO} in blood A group and n_{BB} or n_{BO} in blood A group and B group are missing. This leads to a problem when applying direct counting (Sham, 1998) to estimate p, q, the allele frequency of A and B respectively. The following steps will show how this missing data problem could be solved by the EM algorithm.

In each iteration k, firstly, the **E-step** computes the expected value of the log-likelihood $h(p, q)$ using the observed data $n_{\text{obs}} = (n_A, n_B, n_{AB}, n_O)$ and current parameter value $p^{(k)}, q^{(k)}$:

$$Q_k(p,q) = E[h_k(p,q) | n_{\text{obs}}, p^{(k)}, q^{(k)}] \quad (4)$$

and $h_k(p,q) \sim 2n_{AA} \log p^{(k)} + n_{AO} \log(2p^{(k)}(1-p^{(k)}-q^{(k)})) + 2n_{BB} \log q^{(k)} + n_{BO} \log(2p^{(k)}(1-p^{(k)}-q^{(k)})) + n_{AB} \log(2p^{(k)}q^{(k)}) + 2n_O \log(1-p^{(k)}-q^{(k)})$

Since the log-likelihood is linear w.r.t. the missing data, therefore, when taking the expectation for each component, the missing data can be imputed in this case. For example, since $E[2n_{AA} \log p^{(k)} | n_{\text{obs}}, p^{(k)}, q^{(k)}] = 2 \log p^{(k)} E[n_{AA} | n_{\text{obs}}, p^{(k)}, q^{(k)}]$, and under the HWE assumption,

$$E[n_{AA} | n_{\text{obs}}, p^{(k)}, q^{(k)}] = \frac{\text{freq}(AA)}{\text{freq}(AA) + \text{freq}(AO)} n_A = \frac{p^{(k)} p^{(k)}}{p^{(k)} p^{(k)} + 2p^{(k)} \left(1 - p^{(k)} - q^{(k)}\right)} n_A := n_{AA}^{(k)}$$

By the similar calculation, we can get the imputed data $n_{AA}^{(k)}, n_{AO}^{(k)}, n_{BB}^{(k)}, n_{BO}^{(k)}$ for each iteration.

Then the **M-step** computes the MLE based on the likelihood in equation (4). To be specific,

$$\frac{\partial Q_k(p,q)}{\partial p} = \frac{2n_{AA}^{(k)} + n_{AO}^{(k)} + n_{AB}}{p} - \frac{n_{AO}^{(k)} + n_{BO}^{(k)} + n_O}{1-p-q} = 0$$

$$\frac{\partial Q_k(p,q)}{\partial q} = \frac{2n_{BB}^{(k)} + n_{BO}^{(k)} + n_{AB}}{q} - \frac{n_{AO}^{(k)} + n_{BO}^{(k)} + n_O}{1-p-q} = 0$$

Thus, the updated values of parameters are

$$p^{(k+1)} = \frac{2n_{AA}^{(k)} + n_{AO}^{(k)} + n_{AB}}{2n} \qquad q^{(k+1)} = \frac{2n_{BB}^{(k)} + n_{BO}^{(k)} + n_{AB}}{2n}$$

```
max <- 10000
epsilon <- 1e-5
iter <- 0
p <- 0.3333333
q <- 0.3333333
diff1 <- 1
diff2 <- 1

ep <- NULL
eq <- NULL
elike <- NULL
ep[1] <- p
eq[1] <- q
elike[1] <- log_like(nA, nB, nAB, nO, p, q)

while (diff1 > epsilon & diff2 > epsilon & iter < max) {

  # E-step
  nAA <- nA * (p*p) / (p*p + 2*p*(1 - p - q))
  nAO <- nA * 2*p*(1 - p - q) / (p*p + 2*p*(1 - p - q))
  nBB <- nB * (q*q) / (q*q + 2*q*(1 - p - q))
  nBO <- nB * 2*q*(1 - p - q) / (q*q + 2*q*(1 - p - q))

  # M-step
  p.new <- (2*nAA + nAO + nAB) / (2*n)
  q.new <- (2*nBB + nBO + nAB) / (2*n)

  diff1 <- abs(p.new - p)
  diff2 <- abs(q.new - q)


  p <- p.new
  q <- q.new

  log_lik <- log_like(nA, nB, nAB, nO, p, q)


  iter <- iter + 1
  ep[iter+1] <- p
  eq[iter+1] <- q
```

```
  elike[iter+1] <- log_lik
}

knitr::kable(
  data.frame(c(0:(length(ep)-1)), ep, eq, elike),
  col.names = c("iteration $k$", "$p^{(k)}$", "$q^{(k)}$", "log likelihood"), booktabs = TRUE,
  align = "cccr",
  caption = 'Results for EM algorithm'
)
```

```
## Warning in attr(x, "align"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

```
## Warning in attr(x, "format"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 1: Results for EM algorithm

iteration k	$p^{(k)}$	$q^{(k)}$	log likelihood
0	0.3333333	0.3333333	-32186.43
1	0.3182572	0.1244235	-24998.46
2	0.2942165	0.1079404	-24827.44
3	0.2888920	0.1066936	-24822.90
4	0.2879007	0.1065736	-24822.76
5	0.2877236	0.1065579	-24822.76
6	0.2876923	0.1065555	-24822.76

```
df <- function(p, q) {
  dfp <- 2*(1 - p - q)*nA / (p*(2 - p - 2*q)) + 2*nB / (2*p + q - 2) + nAB/p - 2*nO / (1 - p - q)
  dfq <- 2*nA / (p + 2*q - 2) + 2*(1 - p - q)*nB / (q*(2 - 2*p - q)) + nAB/q - 2*nO / (1 - p - q)
  c(dfp, dfq)
}

d2f <- function(p, q) {
  pp <- -4*(1 - p - q)^2*nA / (p^2*(2 - p - 2*q)^2) - 2*nA / (p* (2 - p - 2*q)) -
    4*nB / ((2 - 2*p - q)^2) - nAB / (p^2) - 2*nO / ((1 - p - q)^2)
  pq <- 4*(1 - p - q)*nA / (p*(2 - p - 2*q)^2) - 2*nA /(p*(2 - p - 2*q)) -
    2*nB / ((2 - 2*p - q)^2) - 2*nO / ((1 - p - q)^2)
  qp <- -2*nA / ((2 - p - 2*q)^2) + 4*(1 - p - q)*nB / (q *(2 - 2*p - q)^2) -
    2*nB / (q* (2 - 2*p - q)) - 2*nO / ((1 - p - q)^2)
  qq <- -4*nA / ((2 - p - 2*q)^2) - 4*(1 - p - q)^2*nB /((q^2*(2 - 2*p - q)^2)) -
    2*nB / (q*(2 - 2*p - q)) - nAB / (q^2) - 2*nO / ((1 - p - q)^2)
  matrix(c(pp, pq, qp, qq), nrow = 2, ncol = 2, byrow = T)
}

max <- 10000
epsilon <- 1e-5
iter <- 0
p <- 0.3333333
q <- 0.3333333
diff1 <- 1
```

```r
diff2 <- 1
nA <- 9123
nB <- 2987
nAB <- 1269
nO <- 7725

theta <- c(p, q)
rp <- NULL
rq <- NULL
rlike <- NULL
rp[1] <- p
rq[1] <- q
rlike[1] <- log_like(nA, nB, nAB, nO, p, q)


while (diff1 > epsilon & diff2 > epsilon & iter < max) {
  p <- theta[1]
  q <- theta[2]

  theta.new <- theta - solve(d2f(p,q)) %*%  df(p, q)

  diff1 <- abs(theta[1] - theta.new[1])
  diff2 <- abs(theta[2] - theta.new[2])

  theta <- theta.new

  log_lik <- log_like(nA, nB, nAB, nO, theta[1], theta[2])

  iter <- iter + 1

  rp[iter+1] <- theta[1]
  rq[iter+1] <- theta[2]
  rlike[iter+1] <- log_lik

}


knitr::kable(
  data.frame(c(0:(length(rp)-1)), rp, rq, rlike),
  col.names = c("iteration $k$", "$p^{(k)}$", "$q^{(k)}$", "log likelihood"), booktabs = TRUE,
  align = "cccr",
  caption = 'Results for Newton-Raphson algorithm'
)
```

```
## Warning in attr(x, "align"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

```
## Warning in attr(x, "format"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 2: Results for Newton-Raphson algorithm

iteration k	$p^{(k)}$	$q^{(k)}$	log likelihood
0	0.3333333	0.3333333	-32186.43
1	0.4175144	0.0171487	-29913.64
2	0.2981320	0.0318471	-27106.11
3	0.3034715	0.0546993	-25646.39
4	0.2950282	0.0820775	-24968.81
5	0.2892635	0.1013210	-24828.66
6	0.2877549	0.1063252	-24822.77
7	0.2876857	0.1065546	-24822.76
8	0.2876856	0.1065550	-24822.76