PH240C Assignment 3

Xiangyu Hu

November 5, 2014

Problem 1. Conditonal distribution of multinomial counts

Process of thinking:

First let's assume $X_k + X_{k'} = m$. When we are given that, we can reform the question and think of it as a binomial. Assume there are m trials, and we are interested in finding the probability $P(X_k = x_k)$. The probability

$$P(X_k = 1) = p$$

, and the probability

$$P(X_{k'} = 1) = 1 - p$$

Therefore, when given m, $X_k \sim binomial(m, p)$

Originally from multinomial distribution, we know that the probability $P(X_k = 1) = \pi_k$, and the probability $P(X_{k'} = 1) = \pi_{k'}$. Therefore, we can get

$$p = \frac{\pi_k}{\pi_k + \pi_{k'}}$$
$$1 - p = \frac{\pi_{k'}}{\pi_k + \pi_{k'}}$$

Since the conditional distribution of X_k given $X_k + X_{k'}$ is binomial(m, p). The conditonal expected value of X_k given $X_k + X_{k'}$ is:

$$\begin{aligned} E(X_k | X_k + X_{k'}) &= mp \\ &= (X_k + X_{k'}) \frac{\pi_k}{\pi_k + \pi_{k'}} \end{aligned}$$

To derive it mathematically:

$$\begin{aligned} P(X_k = x_k | X_k + X_{k'}) &= \frac{P(X_k = x_k, X_k + X_{k'} = m)}{P(X_k + X_{k'} = m)} \\ &= \frac{P(X_k = x_k, X_{k'} = m - x_k)}{P(X_k + X_{k'} = m)} \\ &= \frac{(1)}{(2)} \\ &= \frac{m!}{x_k!(m - x_k)!} \frac{\pi_k^{x_k} \pi_{k'}^{m - x_k}}{(\pi_k + \pi_{k'})^m} \\ &= \frac{m!}{x_k!(m - x_k)!} \frac{\pi_k^{x_k}}{(\pi_k + \pi_{k'})^{x_k}} \frac{\pi_{k'}^{m - x_k}}{(\pi_k + \pi_{k'})^{m - x_k}} \\ &= \frac{m!}{x_k!(m - x_k)!} \left(\frac{\pi_k}{\pi_k + \pi_{k'}}\right)^{x_k} \left(\frac{\pi_{k'}}{\pi_k + \pi_{k'}}\right)^{m - x_k} \\ &= \binom{m}{k} \left(\frac{\pi_k}{\pi_k + \pi_{k'}}\right)^{x_k} \left(\frac{\pi_{k'}}{\pi_k + \pi_{k'}}\right)^{m - x_k} \end{aligned}$$

$$P(X_k = x_k, X_{k'} = m - x_k) = \frac{n!}{x_k!(m - x_k)!(n - m)!} \pi_k^{x_k} \pi_{k'}^{m - x_k} (1 - \pi_k - \pi_{k'})^{n - m} \quad (1)$$

$$P(X_k + X_{k'} = m) = \frac{n!}{m!(n - m)!} (\pi_k + \pi_{k'})^m (1 - \pi_k - \pi_{k'})^{n - m} \quad (2)$$

Problem 2. Derivation of EM algorithm
Observed incomplete data: 4 phenotypes - A, B, AB, O. $Y = (Y_A, Y_B, Y_{AB}, Y_O) \ \sim multinomial(n, \mathbf{p})$, where $\mathbf{p} = (p_A = \pi_A^2 + 2\pi_A\pi_O, p_B = \pi_B^2 + 2\pi_B\pi_O, p_{AB} = 2\pi_A\pi_B, p_O = \pi_O^2)$

Unserved complete data: 6 unphased genotypes - AA, AO, BB, BO, AB, OO.
$X = (X_{AA}, X_{AO}, X_{BB}, X_{BO}, X_{AB}, X_{OO}) \ \sim multinomial(n, \mathbf{p}')$, where $\mathbf{p}' = (\pi_A^2, 2\pi_A\pi_O, \pi_B^2, 2\pi_B\pi_O, 2\pi_A\pi_B, \pi_O^2)$

For log-likelihood function, first let's look at the general log-likelihood function for multinomial distribution.
Likilihood function:

$$L(\pi_1, \ldots, \pi_K; n, X_1, \ldots, X_K) = \frac{n!}{\prod_{k=1}^K X_k!} \prod_{k=1}^K \pi_k^{X_k}$$

Log-Likilihood function:

$$l(\pi_1, \ldots, \pi_K; n, X_1, \ldots, X_K) = \log(\frac{n!}{\prod_{k=1}^K X_k!}) + \sum_{k=1}^K X_k \log \pi_k$$

Since the first term in the above equation doesn't involve $\pi_k s$. We can ignore it and re-write our log-likelihood function as follows:

$$l(\pi_1, \ldots, \pi_K; n, X_1, \ldots, X_K) = \sum_{k=1}^K X_k \log \pi_k$$

Now, if we plug in $\mathbf{p}$ for the corresponding data structure into the log-likelihood function, we will have:
Observed incomplete data structure log-likelihood:

$$\begin{aligned} l(\mathbf{p}; n, Y_A, Y_B, Y_{AB}, Y_O) &= Y_A \log(\pi_A^2 + 2\pi_A\pi_O) + Y_B \log(\pi_B^2 + 2\pi_B\pi_O) + Y_{AB} \log(2\pi_A\pi_B) + Y_O \log(\pi_O^2) \\ &= Y_A \log(\pi_A^2 + 2\pi_A\pi_O) + Y_B \log(\pi_B^2 + 2\pi_B\pi_O) + Y_{AB} \log(\pi_A\pi_B) + Y_O \log(\pi_O^2) \end{aligned}$$

Unobserved complete data structure log-likelihood:

$$\begin{aligned} l(\mathbf{p}'; n, X_{AA}, X_{AO}, X_{BB}, X_{BO}, X_{AB}, X_{OO}) &= X_{AA} \log(\pi_A^2) + X_{AO} \log(2\pi_A\pi_O) + X_{BB} \log(\pi_B^2) + X_{BO} \log(2\pi_B\pi_O) \\ &+ X_{AB} \log(2\pi_A\pi_B) + X_{OO} \log(\pi_O^2) \\ &= X_{AA} \log(\pi_A^2) + X_{AO} \log(\pi_A\pi_O) + X_{BB} \log(\pi_B^2) + X_{BO} \log(\pi_B\pi_O) \\ &+ X_{AB} \log(\pi_A\pi_B) + X_{OO} \log(\pi_O^2) \end{aligned}$$

The unobserved complete data structure log-likelihood is more tractable than observed incomplete data structure log-likelihood.
Main EM Q-function Here $\psi' = (\pi'_A, \pi'_B, \pi'_O)$, $\psi = (\pi_A, \pi_B, \pi_O)$. Also, $Y_A = X_{AA} + X_{AO}, Y_B = X_{BB} + X_{BO}, Y_{AB} = X_{AB}, Y_O = X_{OO}$

$$\begin{aligned} Q(\psi'|\psi) &= E[logf(X;\psi')|Y = y; \psi] \\ &= E[X_{AA} \log(\pi_A'^2) + X_{AO} \log(\pi'_A\pi'_O) + X_{BB} \log(\pi_B'^2) + X_{BO} \log(\pi'_B\pi'_O) + X_{AB} \log(\pi'_A\pi'_B) \\ &+ X_{OO} \log(\pi_O'^2)|Y_A = X_{AA} + X_{AO}, Y_B = X_{BB} + X_{BO}, Y_{AB} = X_{AB}, Y_O = X_{OO}, \pi_A, \pi_B, \pi_O] \\ &= \log(\pi_A'^2)E(X_{AA}|Y_A = X_{AA} + X_{AO}, \pi_A, \pi_B, \pi_O) + \log(\pi'_A\pi'_O)E(X_{AO}|Y_A = X_{AA} + X_{AO}, \pi_A, \pi_B, \pi_O) \\ &+ \log(\pi_B'^2)E(X_{BB}|Y_B = X_{BB} + X_{BO}, \pi_A, \pi_B, \pi_O) + \log(\pi'_B\pi'_O)E(X_{BO}|Y_B = X_{BB} + X_{BO}, \pi_A, \pi_B, \pi_O) \\ &+ Y_{AB} \log(\pi'_A\pi'_B) + Y_O \log(\pi_O'^2) \end{aligned}$$

E-step:

$$
\begin{aligned}
Q(\psi'|\psi) =& Y_A \frac{\pi_A^2}{\pi_A^2 + 2\pi_A\pi_O} 2\log(\pi'_A) + Y_A \frac{2\pi_A\pi_O}{\pi_A^2 + 2\pi_A\pi_O} \log(\pi'_A\pi'_O) + Y_B \frac{\pi_B^2}{\pi_B^2 + 2\pi_B\pi_O} 2\log(\pi'_B) \\
&+ Y_B \frac{2\pi_B\pi_O}{\pi_B^2 + 2\pi_B\pi_O} \log(\pi'_B\pi'_O) + Y_{AB}\log(\pi'_A\pi'_B) + Y_O 2\log(\pi'_O) \\
=& Y_A \frac{\pi_A^2}{\pi_A^2 + 2\pi_A\pi_O} 2\log(\pi'_A) + Y_A \frac{2\pi_A\pi_O}{\pi_A^2 + 2\pi_A\pi_O} (\log(\pi'_A) + \log(\pi'_O)) + Y_B \frac{\pi_B^2}{\pi_B^2 + 2\pi_B\pi_O} 2\log(\pi'_B) \\
&+ Y_B \frac{2\pi_B\pi_O}{\pi_B^2 + 2\pi_B\pi_O} (\log(\pi'_B) + \log(\pi'_O)) + Y_{AB}(\log(\pi'_A) + \log(\pi'_B)) + 2Y_O\log(\pi'_O) \\
=& Y_A \frac{2\pi_A^2}{\pi_A^2 + 2\pi_A\pi_O} \log(\pi'_A) + Y_A \frac{2\pi_A\pi_O}{\pi_A^2 + 2\pi_A\pi_O} \log(\pi'_A) + Y_{AB}\log(\pi'_A) \\
&+ Y_B \frac{2\pi_B^2}{\pi_B^2 + 2\pi_B\pi_O} \log(\pi'_B) + Y_B \frac{2\pi_B\pi_O}{\pi_B^2 + 2\pi_B\pi_O} \log(\pi'_B) + Y_{AB}\log(\pi'_B) \\
&+ Y_A \frac{2\pi_A\pi_O}{\pi_A^2 + 2\pi_A\pi_O} \log(\pi'_O) + Y_B \frac{2\pi_B\pi_O}{\pi_B^2 + 2\pi_B\pi_O} \log(\pi'_O) + 2Y_O\log(\pi'_O) \\
=& (Y_A \frac{2\pi_A^2}{\pi_A^2 + 2\pi_A\pi_O} + Y_A \frac{2\pi_A\pi_O}{\pi_A^2 + 2\pi_A\pi_O} + Y_{AB})\log(\pi'_A) \\
&+ (Y_B \frac{2\pi_B^2}{\pi_B^2 + 2\pi_B\pi_O} + Y_B \frac{2\pi_B\pi_O}{\pi_B^2 + 2\pi_B\pi_O} + Y_{AB})\log(\pi'_B) \\
&+ (Y_A \frac{2\pi_A\pi_O}{\pi_A^2 + 2\pi_A\pi_O} + Y_B \frac{2\pi_B\pi_O}{\pi_B^2 + 2\pi_B\pi_O} + 2Y_O)\log(\pi'_O)
\end{aligned}
$$

M-step:
Let

$$Z_A = Y_A \frac{2\pi_A^2}{\pi_A^2 + 2\pi_A\pi_O} + Y_A \frac{2\pi_A\pi_O}{\pi_A^2 + 2\pi_A\pi_O} + Y_{AB}$$

$$Z_B = Y_B \frac{2\pi_B^2}{\pi_B^2 + 2\pi_B\pi_O} + Y_B \frac{2\pi_B\pi_O}{\pi_B^2 + 2\pi_B\pi_O} + Y_{AB}$$

$$Z_O = Y_A \frac{2\pi_A\pi_O}{\pi_A^2 + 2\pi_A\pi_O} + Y_B \frac{2\pi_B\pi_O}{\pi_B^2 + 2\pi_B\pi_O} + 2Y_O$$

$$\sum_{i=1}^{3} Z_i = 2n$$

Then, the main Q-function becomes a multinomial log-likelihood: $Q(\psi'|\psi) = Z_A\log(\pi'_A) + Z_B\log(\pi'_B) + Z_O\log(\pi'_O)$
MLE of multinomial:

$$\hat{\pi'_A} = \frac{Z_A}{2n}$$

$$\hat{\pi'_B} = \frac{Z_B}{2n}$$

$$\hat{\pi'_O} = \frac{Z_O}{2n}$$

Problem 3. Software implementation of EM algorithm

```
EM = function(obsCount, start, stopping){

  obsLoglik = function(P, Y){
    Ya = Y[1]
    Yb = Y[2]
    Yab = Y[3]
    Yo = Y[4]
    Pa = P[1]
```

```r
  Pb = P[2]
  Po = P[3]
  LogLik = Ya*log(Pa^2 + 2*Pa*Po) + Yb*log(Pb^2 + 2*Pb*Po) + Yab*log(Pa*Pb) + Yo*log(Po^2)
  return(LogLik)
}

Q.mle = function(P, Y){
  Ya = Y[1]
  Yb = Y[2]
  Yab = Y[3]
  Yo = Y[4]
  Pa = P[1]
  Pb = P[2]
  Po = P[3]

  Za = Ya*((2*Pa^2)/(Pa^2 + 2*Pa*Po)) + Ya*((2*Pa*Po)/(Pa^2 + 2*Pa*Po)) + Yab
  Zb = Yb*((2*Pb^2)/(Pb^2 + 2*Pb*Po)) + Yb*((2*Pb*Po)/(Pb^2 + 2*Pb*Po)) + Yab
  Zo = Ya*((2*Pa*Po)/(Pa^2 + 2*Pa*Po)) + Yb*((2*Pb*Po)/(Pb^2 + 2*Pb*Po)) + 2*Yo

  Pa.new = Za/(2*sum(Y))
  Pb.new = Zb/(2*sum(Y))
  Po.new = Zo/(2*sum(Y))

  P.new = c(Pa.new, Pb.new, Po.new)
  return(P.new)
}
# maximize Q while it doesn't met the stoping rules
if (stopping == "MLE"){
  P.new = Q.mle(start,obsCount)
  P = start
  iteration=cbind(matrix(P.new, nrow=1), obsLoglik(P.new,obsCount))
  while(abs(obsLoglik(P.new, obsCount) - obsLoglik(P,obsCount)) > 0.0001){
    P = P.new
    P.new = Q.mle(P,obsCount)
    newRow = cbind(matrix(P.new, nrow=1), obsLoglik(P.new,obsCount))
    iteration = rbind(iteration, newRow)
  }
  colnames(iteration) = c("candidateMLE_Pa","candidateMLE_Pb","candidateMLE_Po", "obsLogLik")
  result = list("candidateMLE" = P.new, "obsLogLik" = obsLoglik(P.new, obsCount), "iter" = iteration)
  return(result)
}

else if (stopping == "FixMag"){
  P.new = Q.mle(start,obsCount)
  P = start
  iteration=cbind(matrix(P.new, nrow=1), obsLoglik(P.new,obsCount))
  while(dist(rbind(P,P.new), method = "euclidean") > 0.0001){
    P = P.new
    P.new = Q.mle(P,obsCount)
    newRow = cbind(matrix(P.new, nrow=1), obsLoglik(P.new,obsCount))
    iteration = rbind(iteration, newRow)
  }
  colnames(iteration) = c("candidateMLE_Pa","candidateMLE_Pb","candidateMLE_Po", "obsLogLik")
  result = list("candidateMLE" = P.new, "obsLogLik" = obsLoglik(P.new, obsCount), "iter" = iteration)
  return(result)
}

else if (stopping == "FlexThres"){
  P.new = Q.mle(start,obsCount)
```

```
    P = start
    iteration=cbind(matrix(P.new, nrow=1), obsLoglik(P.new,obsCount))
    while(abs(P.new[1]-P[1]) > 0.0001*(abs(P[1])+0.00001) & abs(P.new[2]-P[2]) > 0.0001*(abs(P[2])+0.00001) & 
      P = P.new
      P.new = Q.mle(P,obsCount)
      newRow = cbind(matrix(P.new, nrow=1), obsLoglik(P.new,obsCount))
      iteration = rbind(iteration, newRow)
    }
    colnames(iteration) = c("candidateMLE_Pa","candidateMLE_Pb","candidateMLE_Po", "obsLogLik")
    result = list("candidateMLE" = P.new, "obsLogLik" = obsLoglik(P.new, obsCount), "iter" = iteration)
    return(result)
  }

}
```

Problem 4. Application of EM algorithm

```
# here P only contain Pa, Pb two parameters
negobsLoglik = function(P, Y){
    Ya = Y[1]
    Yb = Y[2]
    Yab = Y[3]
    Yo = Y[4]
    Pa = P[1]
    Pb = P[2]
    Po = 1-Pa-Pb
    if (Pa + Pb < 1){
      LogLik = -(Ya*log(Pa^2 + 2*Pa*Po) + Yb*log(Pb^2 + 2*Pb*Po) + Yab*log(Pa*Pb) + Yo*log(Po^2))
    }
    else{
      LogLik = Inf
    }
    return(LogLik)
  }

obsLoglik2 = function(Pa,Pb, count){
    Ya = count[1]
    Yb = count[2]
    Yab = count[3]
    Yo = count[4]
    Po = 1-Pa-Pb
    if (Pa + Pb < 1){
      LogLik = Ya*log(Pa^2 + 2*Pa*Po) + Yb*log(Pb^2 + 2*Pb*Po) + Yab*log(Pa*Pb) + Yo*log(Po^2)
    }
    else{
      LogLik = - Inf
    }
    return(LogLik)
  }
# Apply the EM algorithm, and trace the progress of the EM algorithm
start = c(0.0001,0.0001,0.0001)
obsCount = c(186, 38, 13, 284)

EM_result = EM(obsCount, start, stopping="FlexThres")
EM_result

## $candidateMLE
## [1] 0.21360 0.05015 0.73626
##
## $obsLogLik
```

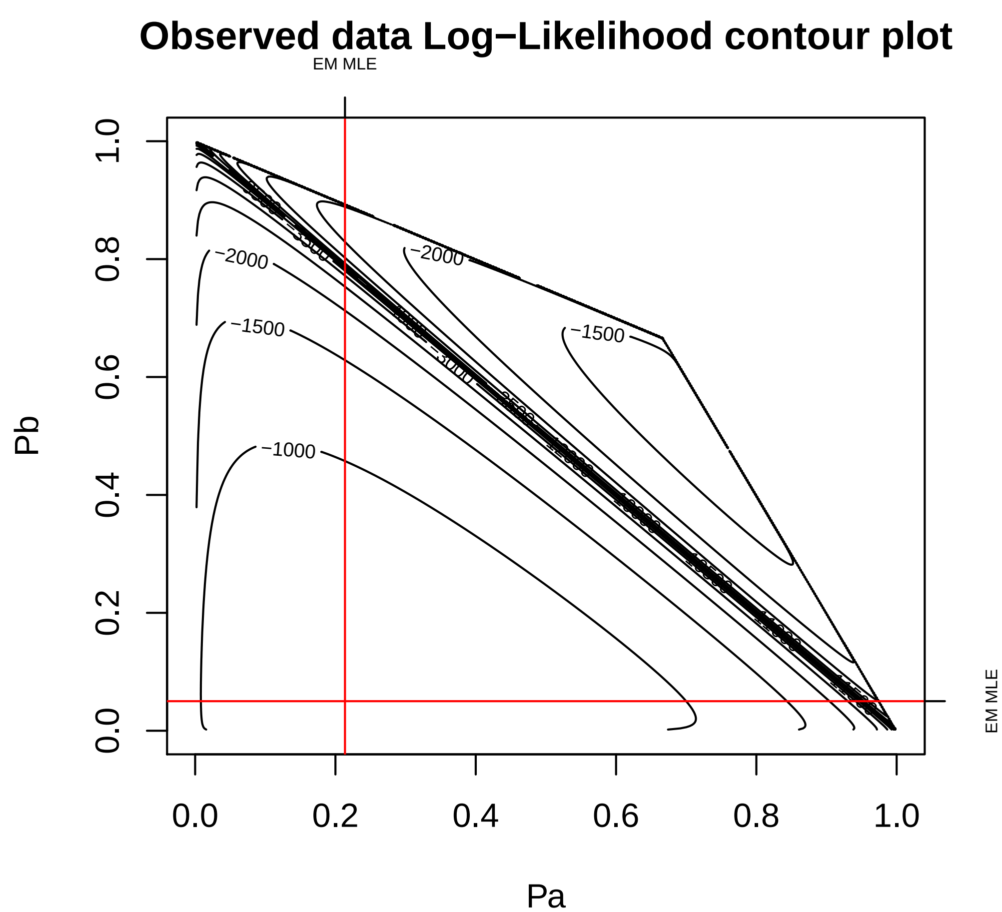
```
## [1] -520.6
##
## $iter
##      candidateMLE_Pa candidateMLE_Pb candidateMLE_Po obsLogLik
## [1,]          0.2505         0.06110          0.6884    -525.7
## [2,]          0.2185         0.05049          0.7311    -520.7
## [3,]          0.2142         0.05016          0.7357    -520.6
## [4,]          0.2137         0.05015          0.7362    -520.6
## [5,]          0.2136         0.05015          0.7363    -520.6

# graphical summaries
trace = EM_result$iter

# Pa + Pb + Po = 1, Pa + Pb < 1
Pa = seq(0,1, length.out=500)
Pb = seq(0,1, length.out=500)
z = outer(Pa,Pb, FUN=obsLoglik2, count = obsCount)

## Warning:  the condition has length > 1 and only the first element will be used
## Warning:  NaNs produced
## Warning:  NaNs produced

contour(Pa,Pb,z, nlevels=50, xlab="Pa", ylab="Pb", main="Observed data Log-Likelihood contour plot")
abline(v=EM_result$candidateMLE[1], h=EM_result$candidateMLE[2], col="red",xlab="EM MLE Pa" )
axis(3, at=EM_result$candidateMLE[1], labels="EM MLE", cex.axis=0.5)
axis(4, at=EM_result$candidateMLE[2], labels="EM MLE", cex.axis=0.5)
```



```r
# Comment on the EM algorithm performance
system.time(EM(obsCount, start, stopping="FlexThres"))

##    user  system elapsed
##   0.001   0.000   0.000

system.time(em2 <-EM(obsCount, c(0.05,0.3,0.001), stopping="FlexThres"))

##    user  system elapsed
##   0.000   0.000   0.001

system.time(em3 <-EM(obsCount, c(0.5,0.8,0.1), stopping="FlexThres"))

##    user  system elapsed
##   0.000   0.000   0.001

system.time(em4 <-EM(obsCount, c(0.8,0.03,0.05), stopping="FlexThres"))

##    user  system elapsed
##       0       0       0

system.time(em5 <-EM(obsCount, c(0.6,0.5,0.01), stopping="FlexThres"))

##    user  system elapsed
##   0.001   0.000   0.000

EM_result$candidateMLE

## [1] 0.21360 0.05015 0.73626

em2$candidateMLE

## [1] 0.21360 0.05015 0.73626

em3$candidateMLE

## [1] 0.21362 0.05015 0.73624

em4$candidateMLE

## [1] 0.21362 0.05015 0.73623

em5$candidateMLE

## [1] 0.21360 0.05015 0.73626

# Comment: The time for running the EM algorithm (rate of convergence) doesn't differ much when
#having different starting values. Also, the result obtained by these different start values are
#very identical. Therefore, the performance of the EM algorithm is really good.

# compare the result from my implementation of EM algorithm and the one from optim
start2 = c(0.0001, 0.0001)
optim(start2, negobsLoglik, Y = obsCount)$par

## [1] 0.21354 0.05016

# Po
1 - sum(optim(start2, negobsLoglik, Y = obsCount)$par)

## [1] 0.7363

# The results are identical
```