hw 9 solution

Statistical Computing, Jieun Shin

Autumn 2022

문제 1.

관측도수 $(x_1, x_2) = (117, 83)$가 주어졌을 때, $x_1, x_2 \sim \text{mult}(1-3p, 3p)$를 따르고 있다. $x_1 = y_1 + y_2, x_2 = y_2$로 놓으면 완비데이터는 $(y_1, y_2, y_3) \sim \text{mult}(1-4p, p, 3p)$라 할 수 있다. 완비 로그가능도는

$$\ell(p; x, y) = y_1 \log(1-4p) + y_2 \log p + y_3 \log 3p + \text{상수}$$

로 정의 가능하다.

p의 최대가능도 추정량은 완비 로그가능도를 미분하여 다음과 같이 구할 수 있다.

$$\begin{aligned}
&\frac{-4y_1}{1-4p} + \frac{y_2}{p} + \frac{y_3}{p} = 0 \\
&\Rightarrow -4y_1 p + (y_2 + y_3)(1-4p) = 0 \\
&\Rightarrow -4y_1 p + y_2 + y_3 - 4p(y_2 + y_3) = 0 \\
&\Rightarrow 4p(y_1 + y_2 + y_3) = y_2 + y_3 \\
&\therefore \hat{p} = \frac{y_2 + y_3}{4(y_1 + y_2 + y_3)}
\end{aligned}$$

여기서 y_2(혹은 y_1)은 관측이 안되는 값이기 때문에 조건부 기댓값으로 대신하여 알고리즘을 작동시킨다. y_2의 조건부 분포는

$$f(y_2; x, p_0) = \frac{f(x, y; p_0)}{f(x; p_0)} \propto \frac{(1-4p_0)^{y_1} p_0^{y_2} (3p_0)^{y_3}}{(1-3p_0)^{x_1} (3p_0)^{x_2}} = \frac{(1-4p_0)^{y_1} p_0^{y_2}}{(1-3p_0)^{x_1+x_2}} = \left(\frac{1-4p_0}{1-3p_0}\right)^{y_1} \left(\frac{p_0}{1-3p_0}\right)^{y_2}$$

이므로 $y_2|x, p_k \sim B(117, \frac{p_k}{1-3p_k}), k = 0, 1, \ldots$를 따른다. 따라서 y_2의 조건부 기댓값은 $\mathbb{E}[y_2|x, p_k] = \frac{117p_k}{1-3p_k}$이다 (그리고 $\mathbb{E}[y_1|x, p_k] = \frac{117(1-4p_k)}{1-3p_k}$가 된다).

기대화 단계에서는 완비 로그가능도의 기댓값을 아래와 같이 정의한다:

$$\begin{aligned}
Q(p_k, p_{k+1}) &= \mathbb{E}[\ell(p_k; x, y)] \\
&= \mathbb{E}[y_1|x, p_k] \log(1-4p_{k+1}) + \mathbb{E}[y_2|x, p_k] \log p_{k+1} + y_3 \log 3p_{k+1} + \text{상수}
\end{aligned}$$

최대화 단계에서는 $Q(p_k, p_{k+1})$의 최댓값을 구한다:

$$\begin{aligned}
&\frac{dQ(p_k, p_{k+1})}{dp_{k+1}} = \frac{-4\mathbb{E}[y_1|x, p_k]}{1-4_{k+1}} + \frac{\mathbb{E}[y_2|x, p_k]}{p_{k+1}} + \frac{y_3}{p_{k+1}} = 0 \\
&\therefore \hat{p}_k = \frac{\mathbb{E}[y_2|x, p_k] + y_3}{4(\mathbb{E}[y_1|x, p_k] + \mathbb{E}[y_2|x, p_k] + y_3)}
\end{aligned}$$

최적의 p는 기대화 단계와 최대화 단계를 수렴할 때까지 반복하여 구한다. 결과를 보면 약 10번의 반복에서 0.138로 추정되었다.

```r
y3 = 83

p = runif(1, 0, 1/3)  # initialize
k = 0
while(k < 100){
  k = k+1
  Ey1 = 117*(1-4*p) / (1-3*p)
  Ey2 = 117*p / (1-3*p)
  p_new = (Ey2+y3)/(Ey1+Ey2+y3)/4

  if(abs(p-p_new) < 0.0001) break
  p = p_new
}
cat("반복수 =",k, "p =",p)
```

```
## 반복수 = 15 p = 0.1384176
```

```python
import numpy as np
y3 = 83

p = np.random.random(1)  # initialize
k = 0
while k < 100:
  k = k+1
  Ey1 = 117*(1-4*p) / (1-3*p)
  Ey2 = 117*p / (1-3*p)
  p_new = (Ey2+y3)/(Ey1+Ey2+y3)/4

  if np.abs(p-p_new) < 0.0001: break
  p = p_new

print("반복수 =", k, "p=", p)
```

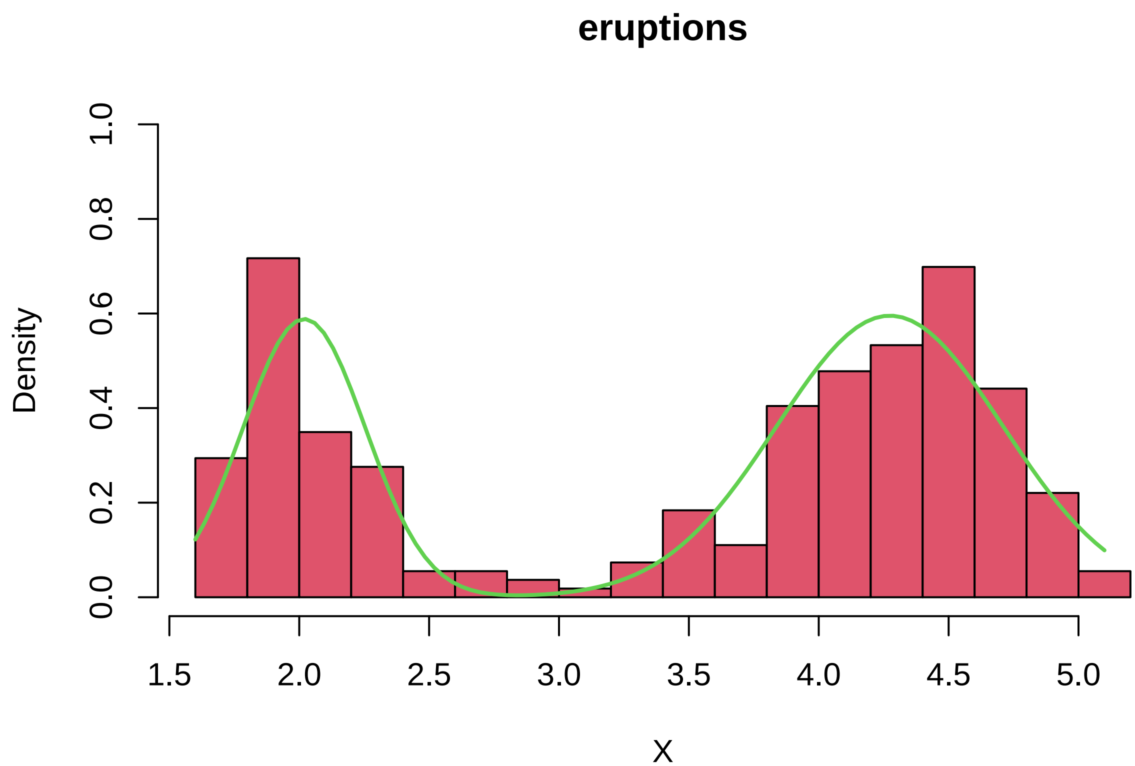
```
## 반복수 = 10 p= [0.13824609]
```

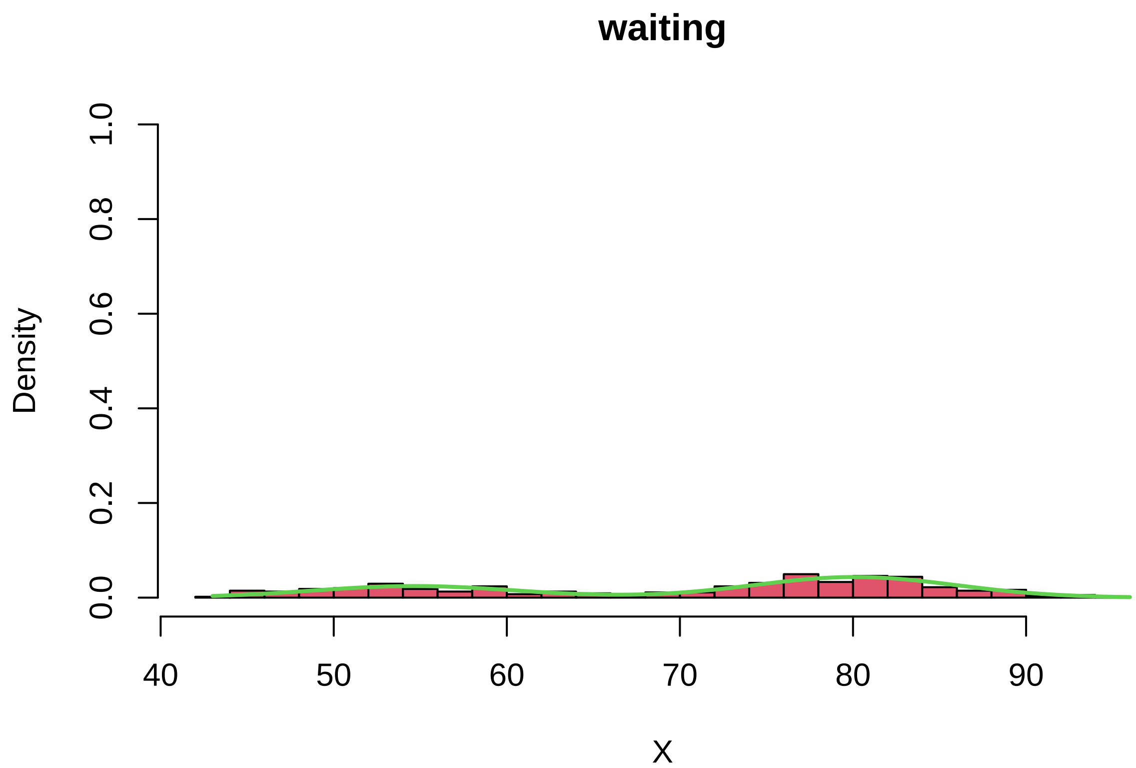
문제 2.

정규혼합분포 적합을 위한 코드는 강의자료의 코드를 사용하였다.

```r
## Example 3: normal mixture

Log.lik = function(x, R=2, mu, sigma, prior)
{
   lik = 0
   for (r in 1:R)
      lik = lik + prior[r] * dnorm(x, mean = mu[r], sd = sigma[r])
   return(sum(log(lik)))
}

Normal.Mixture = function(X, R=2, maxiter=100, eps=1e-5)
{
   X = as.vector(X)
   N = length(X)
   mu = sigma = prior = rep(0, R)
   gama = matrix(0, R, N)
   # find initial centroids using K-means clustering
```

```
    prior = rep(1/R, R)
    kmfit = kmeans(X, R)
    mu = kmfit$centers
    sigma = sqrt(kmfit$withinss /(kmfit$size - 1))
    old.lik = Log.lik(X, R, mu, sigma, prior)
    track.lik = as.vector(NULL)
    track.lik = c(old.lik)
    for (i in 1:maxiter)
    {
        for (r in 1:R)
                gama[r, ] = prior[r] * dnorm(X, mean = mu[r], sd = sigma[r])
        denom = apply(gama, 2, sum)
        for (r in 1:R)
        {
            gama[r, ] = gama[r, ] / denom
            mu[r] = t(gama[r, ]) %*% X / sum(gama[r, ])
            sigma[r] = sqrt(t(gama[r, ]) %*% (X - mu[r])^2 / sum(gama[r, ]))
        }
        prior = apply(gama, 1, sum) / N
        new.lik = Log.lik(X, R, mu, sigma, prior)
        if (abs(old.lik - new.lik) < eps * abs(old.lik))  break
        old.lik = new.lik
        track.lik = c(track.lik, old.lik)
    }
    return(list(mu = mu, sigma = sigma, prior = prior,
                    track = track.lik, resp = gama[r, ]))
}

Mixture.prob = function(x, mu, sigma, prior)
{
    R = length(mu)
    prob = 0
    for (r in 1:R)
        prob = prob + prior[r] * dnorm(x, mu[r], sigma[r])
    return(prob)
}

data = datasets::faithful

for(i in 1:2){
  X = data[,i]
  fit = Normal.Mixture(X)

  hist(X, nclass = 20, xlab = "X", freq = FALSE, col = 2, ylim = c(0, 1), main = colnames(data)[i])

  X.grid = seq(min(X), max(X), length=100)
  points(X.grid, Mixture.prob(X.grid, fit$mu, fit$sigma, fit$prior),
     type = "l", ylab = "density", xlab = "X",
     ylim = c(0, 1), lwd=2, col = 3)
}
```


eruptions
Density
0.0
0.2
0.4
0.6
0.8
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
X



waiting
Density
1.0
0.8
0.6
0.4
0.2
0.0
40
50
60
70
80
90
X