Computational Biology Introduction to the EM algorithm

Predrag Pilipovic (predrag.pilipovic@grenoble-inp.org)

November 9, 2019

The EM algorithm

The EM algorithm is a method for estimating parameters in models with unobserved variables. Classical examples of applications are found in model-based clustering and sequence analysis. EM stands for *Expectation-Maximization*, and it describes an iterative method that maximizes an expected value at each iteration.

Problem statement. Assume that we observe data, $\mathbf{y} = (y_1, y_2, ..., y_n)$, from a probability distribution which is defined hierarchically, as follows

$$p(\mathbf{y} \mid \theta) = \int p(\mathbf{y} \mid \mathbf{z}, \theta) p(\mathbf{z} \mid \theta) d\mathbf{z} = \mathbb{E}[p(\mathbf{y} \mid \mathbf{z}, \theta) \mid \theta].$$

In this formula, θ is the parameter of interest, and z is an unobserved (hidden) variable (the parameter θ and the hidden variable z can have large dimensions). The integral symbol is a generic symbol that represents the summation symbol when $\mathbf{z}$ is a discrete variable and the multiple integration symbol when $\mathbf{z}$ is a continuous multidimensional variable.

A way to estimate θ is by maximizing the log-likelihood function, where the likelihood represents the probability distribution of the observed variables, $\mathbf{y}$, given θ. The solution of the optimization problem can be formalized as follows

$$\theta^* = \arg\max_\theta L(\theta) = \arg\max_\theta \log p(\mathbf{y} \mid \theta).$$

The concern with this approach is that the summation that appears in the formula of the log-likelihood

$$\log p(\mathbf{y} \mid \theta) = \log \left(\int p(\mathbf{y} \mid \mathbf{z}, \theta) p(\mathbf{z} \mid \theta) d\mathbf{z} \right).$$

is very difficult to evaluate in general.

Algorithm. To overcome the above problem, the EM algorithm repeats an iterative process that is guaranteed to increase the likelihood of the parameter at each iteration, and that converges to a local maximum of the likelihood function. Let θ^0 denote the current value of the parameter θ. The EM update rule replaces θ^0 by θ^1, the value of θ that maximizes the following quantity

$$Q(\theta, \theta^0) = \mathbb{E}[\log p(\mathbf{y}, \mathbf{z} \mid \theta) \mid \mathbf{y}, \theta^0] = \int \log p(\mathbf{y}, \mathbf{z} \mid \theta) p(\mathbf{z} \mid \mathbf{y}, \theta^0) d\mathbf{z},$$

and, more generally, we have

$$\theta^{t+1} = \arg\max_\theta Q(\theta, \theta^t).$$

The EM algorithm is useful when the quantity $\log p(\mathbf{y}, \mathbf{z} \mid \theta)$ has a simple expression, for example, a linear function of the hidden variable $\mathbf{z}$, and when the probability $p(\mathbf{z} \mid \mathbf{y}, \theta^0)$ can be easily obtained from the Bayes formula

$$p(\mathbf{z} \mid \mathbf{y}, \theta^0) \propto p(\mathbf{y} \mid \mathbf{z}, \theta^0) p(\mathbf{z} \mid \theta^0).$$

Exercise 1.

Basic arguments and remarks.

Answer the following questions.

1. **Find justifications for why the likelihood increases at each iteration and summarize the key arguments of the proof.**

   Let $\mathbf{Y}$ be a random discrete vector (just for the simplicity of proof, to give a good intuition). We want to find θ such that $p(\mathbf{y} \mid \theta) = P(\mathbf{Y} = \mathbf{y} \mid \theta)$ is maximized. This is known as MLE or Maximum Likelihood estimation for θ. Typically, it is easier to compute $\log p(\mathbf{y} \mid \theta)$, so we introduce *log-likelihood function*

$$L(\theta) = \log p(\mathbf{y} \mid \theta).$$

   This is justified with the fact that the function log is strictly increasing, so the value of θ which maximizes $p(\mathbf{y} = \mathbf{y} \mid \theta)$ also maximizes $L(\theta)$. Assume that after n^{th} iteration the current estimate for θ is θ^n. Since we want to maximize $L(\theta)$, we want to compute an updated estimate θ such that

$$L(\theta) > L(\theta^n).$$

   Equivalently, we want to maximize

$$L(\theta) - L(\theta^n) = \log p(\mathbf{y} \mid \theta) - \log p(\mathbf{y} \mid \theta_n).$$

   Now, we should consider unobserved or missing data. Denote the hidden random vector by $\mathbf{Z}$ and a given realization by $\mathbf{z}$. Note here, that we will assume that $\mathbf{Z}$ is again discrete. Using the total probability law we have

$$p(\mathbf{y} \mid \theta) = \sum_{\mathbf{z}} p(\mathbf{y} \mid \mathbf{z}, \theta) p(\mathbf{z} \mid \theta).$$

   We know that log function is concave so we can use Jensen's inequality which says

$$\log \sum_{i=1}^{n} \lambda_i x_i \geqslant \sum_{i=1}^{n} \lambda_i \log x_i,$$

   with constrains $\lambda_i \geqslant 0$, for all $i = 1, 2, ..., n$, and $\sum\limits_{i=1}^{n} \lambda_i = 1$. Consider letting the lambdas be $p(\mathbf{z} \mid \mathbf{y}, \theta_n)$. Since, it is a probability, we have that $p(\mathbf{z} \mid \mathbf{y}, \theta) \geqslant 0$ and $\sum\limits_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{y}, \theta^n) = 1$.

Applying the previous we have

$$\begin{aligned}
L(\theta) - (\theta^n) &= \log p(\mathbf{y} \mid \theta) - p(\mathbf{y} \mid \theta^n) \\
&= \log \sum_{\mathbf{z}} p(\mathbf{y} \mid \mathbf{z}, \theta) p(\mathbf{z} \mid \theta) - \log p(\mathbf{y} \mid \theta^n) \\
&= \log \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{y}, \theta_n) \cdot \frac{p(\mathbf{y} \mid \mathbf{z}, \theta) p(\mathbf{z} \mid \theta)}{p(\mathbf{z} \mid \mathbf{y}, \theta^n)} - \log p(\mathbf{y} \mid \theta^n) \\
&\geqslant \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{y}, \theta_n) \cdot \log \left(\frac{p(\mathbf{y} \mid \mathbf{z}, \theta) p(\mathbf{z} \mid \theta)}{p(\mathbf{z} \mid \mathbf{y}, \theta^n)} \right) - \log p(\mathbf{y} \mid \theta^n) \\
&= \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{y}, \theta^n) \log \left(\frac{p(\mathbf{y} \mid \mathbf{z}, \theta) p(\mathbf{z} \mid \theta)}{p(\mathbf{z} \mid \mathbf{y}, \theta^n) p(\mathbf{y} \mid \theta^n)} \right).
\end{aligned}$$

Let us define

$$\Delta(\theta \mid \theta^n) \coloneqq \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{y}, \theta^n) \log \left(\frac{p(\mathbf{y} \mid \mathbf{z}, \theta) p(\mathbf{z} \mid \theta)}{p(\mathbf{z} \mid \mathbf{y}, \theta^n) p(\mathbf{Y} \mid \theta^n)} \right).$$

Equivalently, we can write

$$L(\theta) \geqslant L(\theta^n) - \Delta(\theta \mid \theta^n).$$

And for convenience let us define

$$l(\theta \mid \theta^n) \coloneqq L(\theta^n) + \Delta(\theta \mid \theta^n).$$

So now we have

$$L(\theta) \geqslant l(\theta \mid \theta^n).$$

Let see what is happening with function $l(\theta \mid \theta^n)$ in point θ^n. We have

$$\begin{aligned}
\Delta(\theta^n \mid \theta^n) &= \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{y}, \theta^n) \cdot \log \left(\frac{p(\mathbf{y} \mid \mathbf{z}, \theta^n) p(\mathbf{z} \mid \theta^n)}{p(\mathbf{z} \mid \mathbf{y}, \theta^n) p(\mathbf{y} \mid \theta^n)} \right) \\
&= \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{y}, \theta^n) \cdot \cancelto{0}{\log \frac{p(\mathbf{y}, \mathbf{z} \mid \theta^n)}{p(\mathbf{y}, \mathbf{z} \mid \theta^n)}} = 0.
\end{aligned}$$

So, we can conclude that $l(\theta^n \mid \theta^n) = L(\theta^n)$. We have a function $l(\theta \mid \theta^n)$ which is bounded above by the log-likelihood function $L(\theta)$ and the equality holds for $\theta = \theta^n$. Therefore, any θ which increases $l(\theta \mid \theta^n)$ in turn increase $L(\theta)$. Our goal is to choose θ such that $L(\theta)$ is maximized. To achieve the greatest possible increase of $L(\theta)$, the EM algorithm suggests selecting θ which maximizes $l(\theta \mid \theta^n)$. In that way, we will ensure the increment of the function $L(\theta)$ like it is shown in Figure 1.

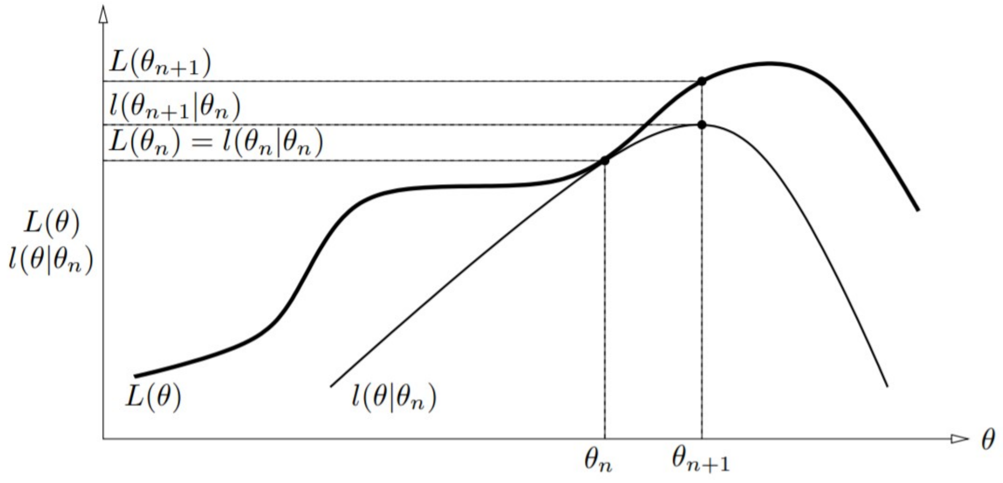

Figure 1: Single iteration of the EM algorithm

We showed that in this setting $L(\theta)$ is increasing in every iteration, but we did not show that this is the EM algorithm. We need to show that in this way of choosing θ^{n+1} we are doing the EM algorithm.

$$\begin{aligned}
\theta^{n+1} &= \arg\max_\theta l(\theta \mid \theta^n) \\
&= \arg\max_\theta \left\{ L(\theta^n) + \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{y}, \theta^n) \log \frac{p(\mathbf{y} \mid \mathbf{z}, \theta) p(\mathbf{z} \mid \theta)}{p(\mathbf{Y} \mid \theta^n) p(\mathbf{z} \mid \mathbf{y}, \theta^n)} \right\} \\
&\quad \text{(we drop terms which are constant with respect to } \theta) \\
&= \arg\max_\theta \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{y}, \theta^n) \log \left(p(\mathbf{y} \mid \mathbf{z}, \theta) p(\mathbf{z} \mid \theta) \right) \\
&= \arg\max_\theta \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{y}, \theta^n) \log \left(\frac{p(\mathbf{y}, \mathbf{z}, \theta) p(\mathbf{z}, \theta)}{p(\mathbf{z}, \theta) p(\theta)} \right) \\
&= \arg\max_\theta \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{y}, \theta^n) \log p(\mathbf{y}, \mathbf{z} \mid \theta) \\
&= \arg\max_\theta \mathbb{E} \left[\log p(\mathbf{y}, \mathbf{z} \mid \theta) \mid \mathbf{y}, \theta^n \right]
\end{aligned}$$

2. **There are obvious limitations to the EM algorithm. Describe two potential concerns with this method.**

   First of all, the EM algorithm is dependent on the initial value, bad parameter initial values can easy make the algorithm to converge to the local optimization points, instead of the global one. Secondly, although we saw that EM algorithm always converge, the convergence speed can be very slow.

Problem 1.

An EM algorithm for detecting frequency change points in a binary sequence.

We observe a sequence of binary data that consists of n observations, $\mathbf{y} = (y_1, y_2, ..., y_n)$, $y_i \in \{0, 1\}$. The n binary variables correspond to independent signals from a random source, for which the frequency of 1's is unknown and may be modified at an unknown time point from θ_1 to θ_2. For example, the sequence of observations can be as follows

$$\mathbf{y} = 01100...00110||1110011...1100111.$$

In this representation, the || symbol indicates that a change occurred at position z, which can represent any point between 2 and n. For $i = 1, 2, ..., z-1$, the frequency of 1's is θ_1, and for $i = z, ..., n$ the frequency of 1's is θ_2. By convention, $z = 1$ corresponds to the situation where no change occurs. In this case, the frequency of 1's is constant, and it is equal to θ_2.

Derivation of an EM algorithm. Let $\mathbf{Y} = (Y_1, Y_2, ..., Y_n)$ be independent discrete random variables which take values in $\{0, 1\}$. Let $\theta = (\theta_1, \theta_2)$ be a parameter we want to estimate. And, let Z be a random value that represents the change point in data.

For $z = 1$ we have

$$p(y_i \mid z, \theta) = P(Y_i = y_i \mid Z = 1, \theta_2) = \theta_2^{y_i}(1-\theta_2)^{1-y_i}, \quad i = 1, 2, ..., n,$$

and for $z = 2, 3, ..., n$, we have

$$p(y_i \mid z, \theta) = P(Y_i = y_i \mid Z = z, \theta_1) = \theta_1^{y_i}(1-\theta_1)^{1-y_i}, \quad i = 1, 2, ..., z-1,$$
$$p(y_i \mid z, \theta) = P(Y_i = y_i \mid Z = z, \theta_2) = \theta_2^{y_i}(1-\theta_2)^{1-y_i}, \quad i = z, z+1, ..., n.$$

A priori, we assume that the change point Z is sampled from the uniform distribution

$$p(z) = P(Z = z) = \frac{1}{n}, \quad z = 1, 2, ..., n.$$

We call this distribution the *prior distribution* on Z. The goal of this problem is to propose an EM algorithm for estimating the model parameter θ and for evaluating the conditional probabilities $p(z \mid \mathbf{y}) = P(Z = z \mid \mathbf{Y} = \mathbf{y})$ for all $z = 1, 2, ..., n$.

1. **Give a formula for the probability $p(\mathbf{y} \mid z, \theta)$ for $z = 1$, and for $z > 1$.**

   Knowing that $Y_1, Y_2, ..., Y_n$ are independent, we have

$$p(\mathbf{y} \mid z, \theta) = P(\mathbf{Y} = \mathbf{y} \mid Z = z, \theta) = \prod_{i=1}^{n} P(Y_i = y_i \mid Z = z, \theta).$$

   From previous we have

$$p(\mathbf{y} \mid z, \theta) = \begin{cases} \prod_{i=1}^{n} \theta_2^{y_i}(1-\theta_2)^{1-y_i}, & z = 1 \\ \prod_{i=1}^{z-1} \theta_1^{y_i}(1-\theta_1)^{1-y_i} \prod_{i=z}^{n} \theta_2^{y_i}(1-\theta_2)^{1-y_i}, & z > 1 \end{cases}$$

2. **Show that, for $z > 1$, we have**

$$\log p(\mathbf{y} \mid z, \theta) = \sum_{i=1}^{z-1} y_i \cdot \log \theta_1 + \left(z - 1 - \sum_{i=1}^{z-1} y_i \right) \cdot \log(1-\theta_1)$$
$$+ \sum_{i=z}^{n} y_i \cdot \log \theta_2 + \left(n - z + 1 - \sum_{i=z}^{n} y_i \right) \cdot \log(1-\theta_2).$$

We know $p(\mathbf{y} \mid z, \theta)$ so we can calculate

$$\begin{aligned}\log p(\mathbf{y} \mid z, \theta) &= \log \left(\prod_{i=1}^{z-1} \theta_1^{y_i} (1-\theta_1)^{1-y_i} \prod_{i=z}^{n} \theta_2^{y_i} (1-\theta_2)^{1-y_i} \right) \\ &= \sum_{i=1}^{z-1} y_i \cdot \log \theta_1 + \left(z - 1 - \sum_{i=1}^{z-1} y_i \right) \cdot \log(1-\theta_1) \\ &+ \sum_{i=z}^{n} y_i \cdot \log \theta_2 + \left(n - z + 1 - \sum_{i=z}^{n} y_i \right) \cdot \log(1-\theta_2).\end{aligned}$$

3. **Suppose θ_1 and θ_2 are known. Using the Bayes formula, show that**

$$R(z) := \frac{P(Z = z \mid \mathbf{Y} = \mathbf{y}, \theta)}{P(Z = 1 \mid \mathbf{Y} = \mathbf{y}, \theta)} = \prod_{i=1}^{z-1} \frac{P(Y_i = y_i \mid Z = z, \theta_1)}{P(Y_i = y_i \mid Z = 1, \theta_2)}, \quad \forall z > 1.$$

First, we can notice that $R(1) = 1$. Let calculate $R(z)$ for $z > 1$, using the Bayes formula

$$\begin{aligned}R(z) &= \frac{P(Z = z \mid \mathbf{Y} = \mathbf{y}, \theta)}{P(Z = 1 \mid \mathbf{Y} = \mathbf{y}, \theta)} \\ &= \frac{P(\mathbf{Y} = \mathbf{y} \mid Z = z, \theta) \cancelto{1/n}{P(Z = z)} P(\mathbf{Y} = \mathbf{y} \mid \theta)}{P(\mathbf{Y} = \mathbf{y} \mid Z = 1, \theta) P(\mathbf{Y} = \mathbf{y} \mid \theta) \cancelto{1/n}{P(Z = 1)}} \\ &= \frac{P(\mathbf{Y} = \mathbf{y} \mid Z = z, \theta)}{P(\mathbf{Y} = \mathbf{y} \mid Z = 1, \theta)} \\ &= \frac{\prod_{i=1}^{z-1} \theta_1^{y_i} (1-\theta_1)^{1-y_i} \prod_{i=z}^{n} \theta_2^{y_i} (1-\theta_2)^{1-y_i}}{\prod_{i=1}^{n} \theta_2^{y_i} (1-\theta_2)^{1-y_i}} \\ &= \prod_{i=1}^{z-1} \frac{\theta_1^{y_i} (1-\theta_1)^{1-y_i}}{\theta_2^{y_i} (1-\theta_2)^{1-y_i}} = \prod_{i=1}^{z-1} \frac{P(Y_i = y_i \mid Z = z, \theta_1)}{P(Y_i = y_i \mid Z = 1, \theta_2)}.\end{aligned}$$

4. **Show that there is a relationship between $R(z)$ and $R(z-1)$, for $z = 2, 3, ..., n$, and propose an algorithm for computing $p(z \mid \mathbf{y})$ for $z = 1, 2, ..., n$.**

From the last line in previous question we have

$$
\begin{aligned}
R(z) &= \prod_{i=1}^{z-1} \frac{\theta_1^{y_i}(1-\theta_1)^{1-y_i}}{\theta_2^{y_i}(1-\theta_2)^{1-y_i}} \\
&= \left(\frac{\theta_1}{\theta_2}\right)^{y_{z-1}} \left(\frac{1-\theta_1}{1-\theta_2}\right)^{1-y_{z-1}} \prod_{i=1}^{z-2} \frac{\theta_1^{y_i}(1-\theta_1)^{y_i}}{\theta_2^{y_i}(1-\theta_2)^{y_i}} \\
&= \left(\frac{\theta_1}{\theta_2}\right)^{y_{z-1}} \left(\frac{1-\theta_1}{1-\theta_2}\right)^{1-y_{z-1}} R(z-1),
\end{aligned}
$$

for $z = 2, 3, ..., n$. Having in mind that $R(1) = 1$, let calculate $p(z \mid \mathbf{y})$ using Bayes formula and the total probability law. We have

$$
\begin{aligned}
P(Z = z^* \mid \mathbf{Y} = \mathbf{y}) &= \frac{P(\mathbf{Y} = \mathbf{y} \mid Z = z^*)P(Z = z^*)}{P(\mathbf{Y} = \mathbf{y})} \\
&= \frac{P(\mathbf{Y} = \mathbf{y} \mid Z = z^*)\cancelto{1/n}{P(Z = z^*)}}{\sum\limits_{z=1}^{n} P(\mathbf{Y} = \mathbf{y} \mid Z = z)\cancelto{1/n}{P(Z = z)}} \\
&= \frac{P(\mathbf{Y} = \mathbf{y} \mid Z = z^*)}{P(\mathbf{Y} = \mathbf{y} \mid Z = 1) + \sum\limits_{z=2}^{n} P(\mathbf{Y} = \mathbf{y} \mid Z = z)} \\
&= \frac{\dfrac{P(\mathbf{Y} = \mathbf{y} \mid Z = z^*)}{P(\mathbf{Y} = \mathbf{y} \mid Z = 1)}}{1 + \sum\limits_{z=2}^{n} \dfrac{P(\mathbf{Y} = \mathbf{y} \mid Z = z)}{P(\mathbf{Y} = \mathbf{y} \mid Z = 1)}} \\
&= \frac{R(z^*)}{1 + \sum\limits_{z=2}^{n} R(z)} = \frac{R(z^*)}{\sum\limits_{z=1}^{n} R(z)},
\end{aligned}
$$

for all $z^* = 1, 2, ..., n$.

5. **Propose an algorithm for computing the expected value $\sum\limits_{i=1}^{Z-1} y_i$**

$$
E_1 = \mathbb{E}\left[\sum_{i=1}^{Z-1} y_i \mid \mathbf{y}, \theta^0\right]
$$

by averaging over all values of z. Apply the same approach to the three other quantities found in question 2. What is the complexity of the algorithm?

Because Z is discrete random variable, we can calculate the expectation by the formula

$$
\mathbb{E}[\varphi(Z) \mid \mathbf{Y} = \mathbf{y}, \theta^0] = \sum_{z=1}^{n} \varphi(z) P(Z = z \mid \mathbf{Y} = \mathbf{y}, \theta^0).
$$

In our case $\varphi(z) = \sum\limits_{i=1}^{z-1} y_i$, so we have

$$\begin{aligned}
E_1 &= \mathbb{E}\left[\sum_{i=1}^{Z-1} y_i \mid \mathbf{Y} = \mathbf{y}, \theta^0\right] = \sum_{z=1}^{n}\sum_{i=1}^{z-1} y_i \cdot P(Z = z \mid \mathbf{Y} = \mathbf{y}, \theta^0) \\
&= \sum_{z=1}^{n} \frac{R(z,\theta^0)}{\sum\limits_{z^*=1}^{n} R(z^*,\theta^0)} \sum_{i=1}^{z-1} y_i = \frac{1}{\widetilde{R}(\theta^0)} \sum_{z=1}^{n} R(z,\theta^0)\varphi(z).
\end{aligned}$$

Note here that R function depends on θ, but in 4. and 5. question we presumed that θ is known. That is the reason we put $R(z, \theta^0)$. Also, we introduce

$$\widetilde{R}(\theta^0) = \sum_{z=1}^{n} R(z, \theta^0)$$

for convenience. We know that $R(z) = 1$, and by the definition $\varphi(z) = 0$. For $R(z, \theta^0)$ and $\varphi(z)$ we have recursive formulas

$$\begin{aligned}
R(z) &= \left(\frac{\theta_1}{\theta_2}\right)^{y_{z-1}} \left(\frac{1-\theta_1}{1-\theta_2}\right)^{1-y_{z-1}} R(z-1) \\
\varphi(z) &= y_{z-1} + \varphi(z-1),
\end{aligned}$$

for $z > 1$. So, we can compute every $R(z)$ and $\varphi(z)$ in one `for` loop and put the values in corresponding arrays. That means we have time complexity $\mathcal{O}(n)$ and memory complexity $\mathcal{O}(n)$. Finally, for calculating $\widetilde{R}(\theta^0)$ we need n iteration, so it is again $\mathcal{O}(n)$, and we need another n iteration for summing $R(z, \theta^0)\varphi(z)$. To conclude we have in total

$$\mathcal{O}(n) + \mathcal{O}(n) + \mathcal{O}(n) + \mathcal{O}(n) = \mathcal{O}(n)$$

time complexity, and

$$\mathcal{O}(n) + \mathcal{O}(n) = \mathcal{O}(n)$$

memory complexity. Similarly, we can compute E_2, E_3 and E_4, and their time complexity will be also $\mathcal{O}(n)$. As far as memory complexity, we do not need new variables for φ_2, φ_3 and φ_4, so it is now $\mathcal{O}(1)$. We have

$$\begin{aligned}
E_2 &= \mathbb{E}\left[Z - 1 - \sum_{i=1}^{Z-1} y_i \mid \mathbf{Y} = \mathbf{y}, \theta^0\right] = \sum_{z=1}^{n}\left(z - 1 - \sum_{i=1}^{z-1} y_i\right) \cdot P(Z = z \mid \mathbf{Y} = \mathbf{y}, \theta^0) \\
&= \sum_{z=1}^{n} \frac{R(z,\theta^0)}{\widetilde{R}(\theta^0)}\left(z - 1 - \sum_{i=1}^{z-1} y_i\right) = \frac{1}{\widetilde{R}(\theta^0)} \sum_{z=1}^{n} R(z,\theta^0)(z - 1 - \varphi(z)), \\
E_3 &= \mathbb{E}\left[\sum_{i=Z}^{n} y_i \mid \mathbf{Y} = \mathbf{y}, \theta^0\right] = \sum_{z=1}^{n}\sum_{i=z}^{n} y_i \cdot P(Z = z \mid \mathbf{Y} = \mathbf{y}, \theta^0) \\
&= \sum_{z=1}^{n} \frac{R(z,\theta^0)}{\widetilde{R}(\theta^0)} \sum_{i=z}^{n} y_i = \frac{1}{\widetilde{R}(\theta^0)} \sum_{z=1}^{n} R(z,\theta^0)\left(\sum_{i=1}^{n} y_i - \varphi(z)\right),
\end{aligned}$$

and finally,

$$E_4 = \mathbb{E}\left[n - Z + 1 - \sum_{i=Z}^{n} y_i \mid \mathbf{Y} = \mathbf{y}, \theta^0\right] = \sum_{z=1}^{n}\left(n - z + 1 - \sum_{i=z}^{n} y_i\right) \cdot P(Z = z \mid \mathbf{Y} = \mathbf{y}, \theta^0)$$
$$= \sum_{z=1}^{n} \frac{R(z,\theta^0)}{\widetilde{R}(\theta^0)}\left(n - z + 1 - \sum_{i=z}^{n} y_i\right) = \frac{1}{\widetilde{R}(\theta^0)} \sum_{z=1}^{n} R(z,\theta^0)\left(n - z + 1 - \sum_{i=1}^{n} y_i + \varphi(z)\right).$$

6. **Describe the EM algorithm for estimating θ from $\mathbf{y}$.**

   Generally, EM algorithm has two steps: E step and M step.

   (E) For some initial value θ^0 we should calculate

   $$Q(\theta,\theta^0) = \mathbb{E}[\log p(\mathbf{y}, \mathbf{z} \mid \theta) \mid \mathbf{y}, \theta^0].$$

   Notice first that we calculated $\log p(\mathbf{y} \mid z, \theta)$, instead of $\log p(\mathbf{y}, z \mid \theta)$, but that is not a problem, because of the Bayes formula

   $$p(\mathbf{y}, z \mid \theta) = p(\mathbf{y} \mid z, \theta) \cdot p(z) = \frac{p(\mathbf{y} \mid z, \theta)}{n}.$$

   So, in our case we have

   $$\begin{aligned}Q(\theta,\theta^0) &= \mathbb{E}[\log P(\mathbf{Y} = \mathbf{y} \mid Z, \theta) - \log n \mid \mathbf{Y} = \mathbf{y}, \theta^0]\\ &= \mathbb{E}\Bigg[\sum_{i=1}^{Z-1} y_i \cdot \log\theta_1 + \left(Z - 1 - \sum_{i=1}^{Z-1} y_i\right) \cdot \log(1-\theta_1) + \sum_{i=Z}^{n} y_i \cdot \log\theta_2\\ &\qquad + \left(n - Z + 1 - \sum_{i=Z}^{n} y_i\right) \cdot \log(1-\theta_2) - \log n \mid \mathbf{Y} = \mathbf{y}, \theta^0\Bigg]\\ &= E_1 \cdot \log\theta_1 + E_2 \cdot \log(1-\theta_1) + E_3 \cdot \log\theta_2 + E_4 \cdot \log(1-\theta_2) - \log n.\end{aligned}$$

   (M) In M step we should maximize $Q(\theta,\theta^0)$ in terms of θ to find θ^1, i.e.

   $$\theta^1 = \arg\max_{\theta} Q(\theta,\theta^0).$$

   We can do that by finding the first derivative of $Q(\theta,\theta^1)$ in terms of θ_1 and θ_2. By solving the system

   $$\begin{cases} \dfrac{\partial Q}{\partial\theta_1} = \dfrac{E_1}{\theta_1} - \dfrac{E_2}{1-\theta_1} = 0 \\ \dfrac{\partial Q}{\partial\theta_2} = \dfrac{E_3}{\theta_2} - \dfrac{E_4}{1-\theta_2} = 0 \end{cases}$$

   we have next solution

   $$\theta_1^1 = \frac{E_1}{E_1 + E_2},$$
   $$\theta_2^1 = \frac{E_3}{E_3 + E_4},$$

   where $\theta^1 = (\theta_1^1, \theta_2^1)$.

7. **Generate simulated data for known values of θ and z. Apply the EM algorithm to the simulated data and evaluate the convergence of the algorithm by testing several values of θ^0. Plot histograms for $p(z \mid \mathbf{y})$ using `plot(., type = 'h')`.**

   For example, let $n = 500$, $z = 134$, and $\theta = (1/3, 3/4)$. Let set threshold

$$|\theta_i^{n+1} - \theta_i^n| < 10^{-8},$$

   for $i = 1, 2$. So, we want to stop algorithm when θ_i^{n+1} and θ_i^n are not different in first 8 decimal places. The EM algorithm estimates parameter θ as

$$\hat{\theta} = (0.3270314, 0.7554965).$$

   Below there is `R` code for EM algorithm.

```
set.seed(1)
n <- 500
z <- 134
theta1 <- 1/3
theta2 <- 3/4
theta <- c(theta1, theta2)
# Generating data
y1 <- sample(c(0,1), z, replace = T, prob = c(1-theta1, theta1))
y2 <- sample(c(0,1), n-z,replace = T, prob = c(1-theta2, theta2))
y <- c(y1, y2)
# Calculating R
R <- function(theta, y)
{
        n <- length(y)
        Rz <- 1:n
        for (z in 2:n)
        {
                Rz[z] <- Rz[z-1] * (theta[1]/theta[2])^y[z-1] *
                ((1 - theta[1])/(1 - theta[2]))^(1 - y[z-1])
        }
        return(Rz)
}
# Calculating Phi
Phi <- function(y)
{
        n <- length(y)
        phi1 <- rep(0, n)
        for (z in 2:n)
        {
                phi1[z] <- phi1[z-1] + y[z]
        }
        return(phi1)
}

```

```
# Calculating expected values
Exp <- function(theta, y)
{
      n <- length(y)
      E1 <- E2 <- E3 <- E4 <- 0
      r <- R(theta, y)
      Rsum <- sum(r)
      phi1 <- Phi(y)
      phi3 <- sum(y) - phi1
      for(z in 1:n)
      {
            E1 <- E1 + r[z]*phi1[z]
            E2 <- E2 + r[z]*(z - 1 - phi1[z])
            E3 <- E3 + r[z]*phi3[z]
            E4 <- E4 + r[z]*(n - z + 1 - phi3[z])
      }
      E <- c(E1, E2, E3, E4)
      E/Rsum
}
# Initializing theta and counter
theta_new <- c(0.1,0.9)
count <- 0
# Finally, here is iterative process of EM
repeat{
          theta_old <- theta_new
          E <- Exp(theta_old, y)
          theta_new1 <- E[1]/(E[1] + E[2])
          theta_new2 <- E[3]/(E[3] + E[4])
          theta_new <- c(theta_new1, theta_new2)
          count <- count + 1
          if(prod(round(theta_old, 8) == round(theta_new, 8)))
          {
                    print(count)
                    print(theta_new)
                    break
          }
}
```

In the next table we can see what happens when we change initial values for θ.

initial value of θ_1	initial value of θ_2	iter
0.33	0.75	5
0.33	0.05	10
0.99	0.75	8
0.01	0.01	8
0.99	0.99	8
0.33	0.99	8

Histogram of $p(z \mid \mathbf{y})$ can be seen in Figure 2.

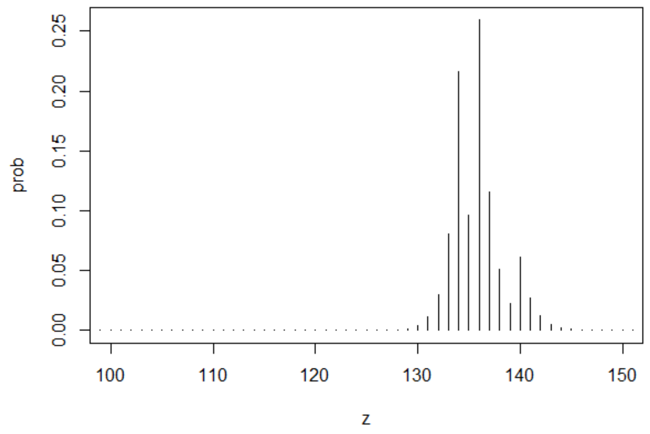

Figure 2: Histogram of distribution of the changing point Z

8. **Discuss the choice of a uniform prior distribution for Z. How could the EM algorithm be modified to account for informative prior distributions?**

   It is a natural thing to choose a uniform prior distribution for Z when we do not have any other information about Z. Without information, it does not make any sense to presume that Z is from binomial or geometrical distribution. But, if we could have some prior knowledge than the algorithm needs to be changed. First of all, we need to change $R(z)$. While calculating $R(z)$ we crossed $P(Z = z)$ and $P(Z = 1)$, but those probabilities are not $\frac{1}{n}$ now. So. we have

$$R(z) = \frac{P(Z = z)}{P(Z = 1)} \cdot \prod_{i=1}^{z-1} \frac{\theta_1^{y_i}(1-\theta_1)^{1-y_i}}{\theta_2^{y_i}(1-\theta_2)^{1-y_i}}.$$

   Similarly, when calculating $p(z^* \mid \mathbf{y})$, we crossed $P(Z = z^*)$, and $P(Z = z)$ and now we have

$$P(Z = z^* \mid \mathbf{Y} = \mathbf{y}) = \frac{R(z^*)P(Z = z^*)}{\sum\limits_{z=1}^{n} R(z)P(Z = z)}.$$

   According to this, we will change E_1, E_2, E_3 and E_4. For example

$$E_1 = \sum_{z=1}^{n} \frac{R(z, \theta^0)P(Z = z)}{\sum\limits_{z^*=1}^{n} R(z^*, \theta^0)P(Z = z^*)} \sum_{i=1}^{z-1} y_i.$$

   At the end we have difference with computing $p(\mathbf{y}, z \mid \theta)$, so we have

$$p(\mathbf{y}, z \mid \theta) = p(\mathbf{y} \mid z, \theta)p_Z(z),$$

   where $p_Z(z) = P(Z = z)$. Now, if we want to compute $Q(\theta, \theta^0)$ we will have

$$\begin{aligned}Q(\theta,\theta^0) &= \mathbb{E}\left[\log P(\mathbf{Y}=\mathbf{y}\mid Z,\theta)+\log P(Z=z)\mid \mathbf{Y}=\mathbf{y},\theta^0\right]\\ &= E_1\cdot\log\theta_1+E_2\cdot\log(1-\theta_1)+E_3\cdot\log\theta_2+E_4\cdot\log(1-\theta_2)\\ &\quad+\mathbb{E}[\log P(Z=z)\mid \mathbf{Y}=\mathbf{y},\theta^0].\end{aligned}$$

After this, we will find θ^1 by maximizing $Q(\theta,\theta^0)$.

Challenge and evaluation rule. Download the data from the following URL

`http://membres-timc.imag.fr/Olivier.Francois/sequence_2019.txt`

The data consists of a sequence of 399 binary items. The objective of the challenge is to provide a list of (one or more) change points with the following information.

1. Most likely change point position, z, in the range $[1,399]$.
2. Lower and upper values z_l and z_u, such that $P(Z\in[z_l,z_u])=0.75$.
3. Estimates of frequencies θ_1 and θ_2 before and after z.
4. A number of iterations of the EM algorithm.

The output file must be formatted as follows

```
number position lower upper theta1 theta2 iter
```

A `README` file including comments and describing the options used when analyzing the data is required. The results will be evaluated based on the

1) number of correct detections,
2) evaluation of uncertainty on each correctly detected position (i.e, the correctness of the difference `(upper - lower)`).

Solution. Instead of creating a new `README` file, here will be presented solutions to the challenge. We will upgrade our algorithm to solve this problem. EM algorithm performed on data `sequence_2019` will provide us a distribution of Z (Figure 3), estimation for θ_1 and θ_2, and a number of iterations.

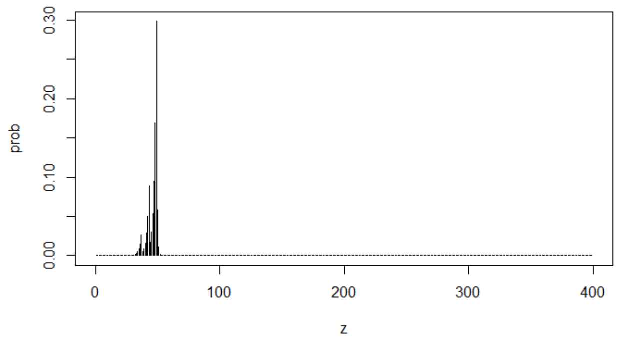

Figure 3: Histogram of distribution of the changing point Z

To find lower and upper values we will need to use cumulative distribution function. In Figure 4 we can see cumulative distribution function or CDF for Z. We put two horizontal lines through probabilities 0.125 and 0.875 to obtain 75% symmetric interval. Because Z is discrete we can not find z_u and z_l such that probability $P(Z \in [z_l, z_u]) = 0.75$. Firstly, we need to find points that are above and below our interval lines, in order to find z_u and z_l, respectively. That can be obtain with

```
zl<- max(which(cumsum(p)<0.125))
zu<- min(which(cumsum(p)>0.875)),
```

where p is a vector of probabilities for Z.

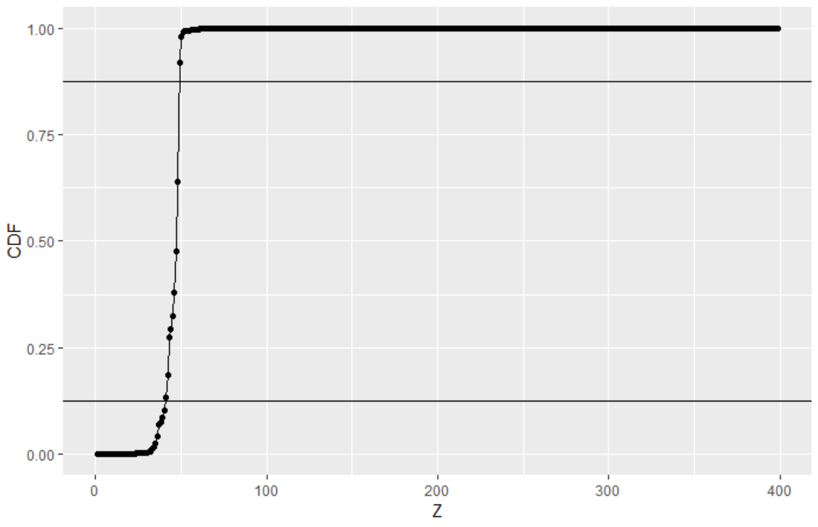

Figure 4: CDF of the changing point Z

We can see in Figure 3 that there are a lot of positions z such that probability is zero or close to zero. This can be our motivation to create subsequence from our original data. So we will eliminate all change-points close to zero. This we can do with function `almost.zero` from library `bazar`. The function `almost.zero` tests if values of the numeric vector `x` are equal to zero up to tolerance. The default tolerance is $1.5 \cdot 10^{-8}$. Our EM algorithm will return two different estimations for θ depending on the initial values. For example, for $\theta^0 = (0.1, 0.8)$ we have $\hat{\theta} = (0.3305813, 0.8581371)$ and most probable change point $z = 314$. On the other hand, for $\theta^0 = (0.1, 0.2)$ we have $\hat{\theta} = (0.09466104, 0.48813029)$ and most probable change point $z = 49$. After creating subsequences in those two scenarios we have next result

```
number  position  lower  upper    theta1      theta2    iter
   1       314     311    318   0.33058134  0.8581371    9
   2        20      14     23   0.32762683  0.7948283   18
   3        49      41     49   0.09466104  0.4881303    9
   4        45      30     45   0.09763800  0.3700609   14
```

In both cases, after having second subsequence, there was not any position for Z with probability close to zero, so every new subsequence would be the same as previous. Also, we can notice that most likely change point in the second case is the same as the upper bound. The reason for this is due to the distribution of Z. In Figure 4 we can see that most likely changing point has probability much greater than the rest of z. Also, distribution is shifted to the right, i.e. the peak is on the right side of the histogram. So when computing CDF this point will increase CDF so much, that it will be greater then 0.875. Finally, in order to understand the result given for subsequence, we need to see what are the correspondence between position in subsequence and position in original data. For example, our subsequence for the first scenario looks like

```
295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315
316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336
                        337 338 339 340 341 342 343 344 345
```

And 20. position in this data are actually 314 again. But for this we have different interval and also different estimation for θ. For the second case we have 45. position but again, it will correspond to 49. position in original data. In the end, let us look at the code.

```
# EM algorithm
EM <- function(theta_seq_new, data)
{
  count <- 0
  repeat{
         theta_seq_old <- theta_seq_new
         E <- Exp(theta_seq_old, data)
         theta_seq_new1 <- E[1]/(E[1] + E[2])
         theta_seq_new2 <- E[3]/(E[3] + E[4])
         theta_seq_new <- c(theta_seq_new1, theta_seq_new2)
         count <- count + 1
         if(prod(round(theta_seq_old, 8) == round(theta_seq_new, 8)))
            break
  }
  return(c(theta_seq_new, count))
}
# Function for having distribution of Z
z <- function(theta, y)
{
     n <- length(y)
     prob <- 0
     r <- R(theta, y)
     Rsum <- sum(r)
     prob <- r/Rsum
     plot(prob, type = 'h', xlim = c(0, 400), xlab = "z")
     return(prob)
}
# First call of EM algorithm with original data
theta_in1 <- c(0.1, 0.8)
EM_alg <- EM(theta_in1, sequence_2019)
theta_EM <- EM_alg[1:2]
iter <- EM_alg[3]
```

```r
p <- z(theta_EM, data)
z_max <- which.max(p)
zl <- max(which(cumsum(p)<0.125))
zu <- min(which(cumsum(p)>0.875))

# Creating data frame with all information
res <- data.frame(V1 = z_max, V2 = zl, V3 = zu,
                  V4 = theta_EM[1], V5 = theta_EM[2], V6 = iter)
colnames(res) <- c("position", "lower","upper", "theta1",
                   "theta2", "iter")
# Crating new subsequence
subsequence <- sequence_2019[!almost.zero(p)]
# Updating data frame with new information from subsequences
result <- function(data)
{
     for(i in 2:6)
     {
          EM_alg <- EM(c(0.1, 0.1), data)
          theta_EM <- EM_alg[1:2]
          iter <- EM_alg[3]
          p <- z(theta_EM, data)
          z_max <- which.max(p)
          zl <- max(which(cumsum(p)<0.125))
          zu <- min(which(cumsum(p)>0.875))
          res <- rbind(res, c(z_max, zl,  zu, theta_EM[1],
                              theta_EM[2], iter))
          data <- data[!almost.zero(p)]
     }
     return(res)
}
res <- result(theta_in1, subsequence)
# Adding new solution from the EM algorithm
theta_in2 <- c(0.1, 0.2)
EM_alg2 <- EM(theta_in2, sequence_2019)
theta_EM2 <- EM_alg2[1:2]
iter2 <- EM_alg2[3]
p2 <- z(theta_EM2, sequence_2019)
z_max2 <- which.max(p2)
zl2 <- max(which(cumsum(p2)<0.125))
zu2 <- min(which(cumsum(p2)>0.875))
res <- rbind(res, c(z_max2, zl2,  zu2, theta_EM2[1], theta_EM2[2],
                   iter2))
subsequence2 <- sequence_2019[!almost.zero(p2)]
res <- result(theta_in2, subsequence2)
# Result
result <- cbind(1:6, unique(res))
```

Problem 2.

Gaussian mixture models.

Consider a population P consisting of 2 subpopulations P_0 and P_1 with equal sizes. We sample n individuals from P, but their origins are not observed. A quantity y_i is measured for each individual. Grouping individuals based on the observations is called an *unsupervised clustering* task.

In order to group individuals into clusters, we assume that subpopulation labels are missing data. We want to estimate the cluster localization m_0 and m_1 for each group and the proportion of individuals sampled from subpopulation P_0 or P_1. In addition, we want to estimate the probability that each individual is sampled from population P_1 (or P_0).

Let $\theta = (m_0, m_1)$. We define a Gaussian mixture model as follows

$$p(y_i \mid \theta) = p \cdot p_1(y_i \mid \theta) + q \cdot p_0(y_i \mid \theta), \quad \forall y_i \in \mathbb{R},$$

where $p_k(yi \mid \theta) \sim \mathcal{N}(y_i \mid m_k, \sigma^2 = 1)$ is the Gaussian density function, for $k = 0, 1$. The probability p is the probability of sampling from P_1, and $q = 1 - p$.

1. **To begin, we consider that the variance σ^2 is a known parameter equal to $\sigma^2 = 1$, and $p = 1/2$. For all individuals, we consider hidden variables $z_i \in \{0, 1\}$ representing their unobserved label of source population. Show that**

   $$p(y_i \mid \theta) = p(z_i = 1)p(y_i \mid z_i = 1, \theta) + p(z_i = 0)p(y_i \mid z_i = 0, \theta),$$

   where $p(z_i = 1) = 1/2$.

   From the assumptions we have that

   $$\begin{aligned} p(y_i \mid \theta) = p \cdot p_1(y_i \mid \theta) + q \cdot p_0(y_i \mid \theta) &= \frac{1}{2} \cdot p_1(y_i \mid \theta) + \frac{1}{2} \cdot p_0(y_i \mid \theta) \\ &= \frac{1}{2} \cdot p(y_i \mid z_i = 1, \theta) + \frac{1}{2} \cdot p(y_i \mid z_i = 0, \theta) \\ &= p(z_i = 1)p(y_i \mid z_i = 1, \theta) + p(z_i = 0)p(y_i \mid z_i = 0, \theta). \end{aligned}$$

   We would have the same result if we applied law of total probability.

2. **Write a computer program for drawing samples from $p(y_i \mid \theta)$ of size $n = 200$ (for fixed values of (p, m_0, m_1)). Check if your program is correct by drawing a histogram of the simulated data.**

   Let us assume for example that $p = 0.27$, $m_0 = 3$ and $m_1 = 7$. The idea of how to generate data from mixture distribution lies in the fact that we need a random vector (X, Z), where Z will denote the class of X, i.e. $X \mid Z = 0 \sim \mathcal{N}(m_0, \sigma^2)$ and $X \mid Z = 1 \sim \mathcal{N}(m_1, \sigma^2)$. So firstly, we need to sample data randomly for Z, and then we need to use that data to sample X. This is really easy to implement in `R` because in function `rnorm` we can put a vector `mu[k]` with corresponding m_0 and m_1 depending on k. Below is `R` code for sampling. Also, in the Figure 5 we can see a histogram of sampled data.

```r
# Initializing values
n <- 200
m0 <- 3
m1 <- 7
mu <- c(m0, m1)
p <- 0.27
pi <- c(p, 1-p)
sigma <- c(1,1)

# Function for simulation
rnormix <- function(n, pi, mu, sigma)
{
     K <- length(pi)
     k <- sample(K, n, replace = TRUE, prob = pi)
     simulations <- rnorm(n, mu[k], sigma[k])
     return(simulations)
}

# Simulation
Data <- rnormix(n, pi, mu, sigma)

# Histogram and density function
hist(Data, probability = T, col = 'lightblue', main = "")
lines(density(Data), add = T, lwd = 2, col = 'coral')
```

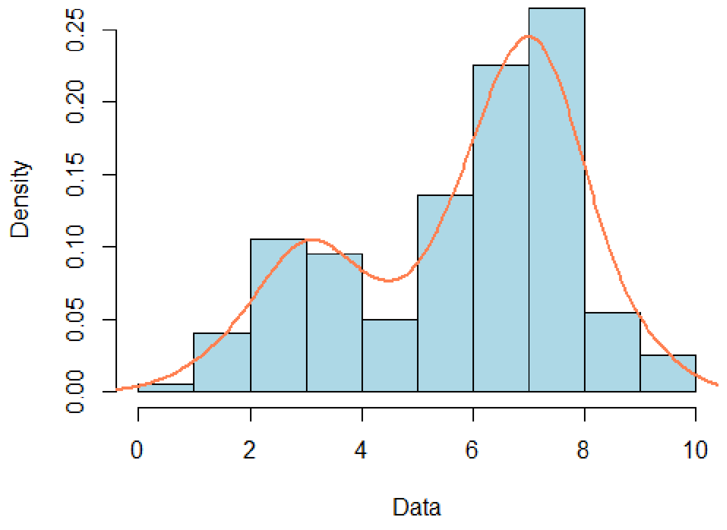

Figure 5: Histogram of distribution Gaussian mixture model

3. **Considering the unobserved vector $\mathbf{z} = (z_1, z_2, ..., z_n)$, show that**

$$\log p(\mathbf{y}, \mathbf{z} \mid \theta) = -\frac{1}{2} \sum_{i=1}^{n} \left((1 - z_i)(y_i - m_0)^2 + z_i(y_i - m_1)^2\right) + C_n.$$

Because $\mathbf{Z}$ is random vector of mutually independent random variables we have

$$p(\mathbf{y}, \mathbf{z} \mid \theta) = p(\mathbf{y} \mid \mathbf{z}, \theta) \cdot p(\mathbf{z}) = p(\mathbf{y} \mid \mathbf{z}, \theta) \cdot \prod_{i=1}^{n} p(z_i) = \frac{p(\mathbf{y} \mid \mathbf{z}, \theta)}{2^n}.$$

Having in mind that

$$\begin{aligned} p(y_i \mid z_i, \theta) &= \left(f_{\mathcal{N}(m_0,\sigma^2=1)}(y_i \mid \theta)\right)^{1-z_i} \cdot \left(f_{\mathcal{N}(m_1,\sigma^2=1)}(y_i \mid \theta)\right)^{z_i} \\ &= \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(y_i-m_0)^2}{2}}\right)^{1-z_i} \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(y_i-m_1)^2}{2}}\right)^{z_i}, \end{aligned}$$

and because of the independence of data we have

$$p(\mathbf{y} \mid \mathbf{z}, \theta) = \prod_{i=1}^{n} \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(y_i-m_0)^2}{2}}\right)^{1-z_i} \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(y_i-m_1)^2}{2}}\right)^{z_i}.$$

After applying logarithmic function on previous line we have

$$\begin{aligned} \log p(\mathbf{y} \mid \mathbf{z}, \theta) &= \sum_{i=1}^{n} \left((1 - z_i) \log \frac{1}{\sqrt{2\pi}} - (1 - z_i) \frac{(y_i - m_0)^2}{2} + z_i \log \frac{1}{\sqrt{2\pi}} - z_i \frac{(y_i - m_1)^2}{2}\right) \\ &= -\frac{1}{2} \sum_{i=1}^{n} \left((1 - z_i)(y_i - m_0)^2 + z_i(y_i - m_1)^2\right) - \frac{n}{2} \cdot \log 2\pi. \end{aligned}$$

Finally, we have

$$\begin{aligned} \log p(\mathbf{y}, \mathbf{z} \mid \theta) &= \log \frac{p(\mathbf{y} \mid \mathbf{z}, \theta)}{2^n} = \log p(\mathbf{y} \mid \mathbf{z}, \theta) - n \cdot \log 2 \\ &= -\frac{1}{2} \sum_{i=1}^{n} \left((1 - z_i)(y_i - m_0)^2 + z_i(y_i - m_1)^2\right) \underbrace{- \frac{n}{2} \log 2\pi - n \cdot \log 2}_{C_n} \end{aligned}$$

4. **Let $n_1 = \sum_{i=1}^{n} z_i$ and $n_0 = n - n_1$. Show that the above expression is maximized for**

$$\widehat{m}_1 = \frac{1}{n_1} \sum_{i=1}^{n} y_i z_i, \qquad \widehat{m}_0 = \frac{1}{n_0} \sum_{i=1}^{n} y_i(1 - z_i) \qquad (\star).$$

We need to find first derivatives of log-likelihood function in terms of θ, i.e. we need to find solution for the next system.

$$\begin{cases} \dfrac{\partial}{\partial m_0} \log p(\mathbf{y}, \mathbf{z} \mid \theta) = & -\dfrac{1}{2} \sum\limits_{i=1}^{n} (1 - z_i)(-2)(y_i - m_0) = \sum\limits_{i=1}^{n} (1 - z_i) y_i - m_0 \sum\limits_{i=1}^{n} (1 - z_i) = 0 \\ \dfrac{\partial}{\partial m_1} \log p(\mathbf{y}, \mathbf{z} \mid \theta) = & -\dfrac{1}{2} \sum\limits_{i=1}^{n} z_i(-2)(y_i - m_1) = \sum\limits_{i=1}^{n} z_i y_i - m_1 \sum\limits_{i=1}^{n} z_i = 0. \end{cases}$$

Knowing that $n_1 = \sum_{i=1}^n z_i$, and $n_0 = n - n_1 = n - \sum_{i=1}^n z_i = \sum_{i=1}^n (1 - z_i)$, we have solution

$$\widehat{m}_0 = \frac{1}{n_0} \sum_{i=1}^n y_i(1 - z_i),$$
$$\widehat{m}_1 = \frac{1}{n_1} \sum_{i=1}^n y_i z_i.$$

5. **We suppose $p(z_i = 1) = 1/2$. Show that the conditional probability of $z_i = 1$ given y_i and θ^0 is equal to**

$$p(z_i = 1 \mid y_i, \theta^0) = \frac{e^{-\frac{(y_i - m_1^0)^2}{2}}}{e^{-\frac{(y_i - m_0^0)^2}{2}} + e^{-\frac{(y_i - m_1^0)^2}{2}}}.$$

Using Bayes formula and law of total probability we have

$$\begin{aligned}
p(z_i = 1 \mid y_i, \theta^0) &= \frac{p(y_i \mid z_i = 1, \theta^0) p(z_i = 1)}{p(y_i \mid \theta^0)} \\
&= \frac{p(y_i \mid z_i = 1, \theta^0)\cancelto{1/2}{p(z_i = 1)}}{p(y_i \mid z_i = 1, \theta^0)\cancelto{1/2}{p(z_i = 1)} + p(y_i \mid z_i = 0, \theta^0)\cancelto{1/2}{p(z_i = 0)}} \\
&= \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{(y_i - m_1^0)^2}{2}}}{\frac{1}{\sqrt{2\pi}} e^{-\frac{(y_i - m_0^0)^2}{2}} + \frac{1}{\sqrt{2\pi}} e^{-\frac{(y_i - m_1^0)^2}{2}}} \\
&= \frac{e^{-\frac{(y_i - m_1^0)^2}{2}}}{e^{-\frac{(y_i - m_0^0)^2}{2}} + e^{-\frac{(y_i - m_1^0)^2}{2}}}.
\end{aligned}$$

6. **In equations ($\star$), replace the hidden variable z_i by $p(z_i = 1 \mid y_i, \theta^0)$, and show that this operation corresponds to writing the EM algorithm for estimating θ.**

In order to obtain EM algorithm, we need to calculate $Q(\theta, \theta^0)$, and after that to maximize it.

(E) We have already seen a few times how to compute $Q(\theta, \theta^0)$.

$$\begin{aligned}
Q(\theta, \theta^0) &= \mathbb{E}\left[\log P(\mathbf{Y} = \mathbf{y}, \mathbf{Z} \mid \theta) \mid \mathbf{Y} = \mathbf{y}, \theta^0\right] \\
&= \mathbb{E}\left[-\frac{1}{2}\sum_{i=1}^n (1 - Z_i)(y_i - m_0)^2 + Z_i(y_i - m_1)^2 + C_n \mid \mathbf{Y} = \mathbf{y}, \theta^0\right] \\
&= -\frac{1}{2}\sum_{i=1}^n (1 - \mathbb{E}[Z_i \mid Y_i = y_i, \theta^0])(y_i - m_0)^2 + \mathbb{E}[Z_i \mid Y_i = y_i, \theta^0](y_i - m_1)^2 + C_n \\
&= -\frac{1}{2}\sum_{i=1}^n (1 - p(z_i = 1 \mid y_i, \theta^0))(y_i - m_0)^2 + p(z_i = 1 \mid y_i, \theta^0)(y_i - m_1)^2 + C_n.
\end{aligned}$$

Note that the last line is true due the fact that $Z_i \in \{0, 1\}$, so the $\mathbb{E}[Z_i] = P(Z_i = 1)$.

(M) In order to maximize $Q(\theta, \theta^0)$ in terms of θ, we need to find derivatives, but we actually did that while computing $(\star)$, so it is sufficient replace z_i by $p(z_i = 1 \mid y_i, \theta^0)$, so we have

$$m_0^1 = \frac{1}{n_0} \sum_{i=1}^{n} y_i (1 - p(z_i = 1 \mid y_i, \theta^0)),$$
$$m_1^1 = \frac{1}{n_1} \sum_{i=1}^{n} y_i p(z_i = 1 \mid y_i, \theta^0).$$

7. **Write the EM algorithm in the `R` programming language. Generate simulated data that for known values of θ and z. Apply the EM algorithm to the simulated data, and evaluate the convergence of the algorithm by testing several initial values θ^0.**

We can continue with the same example. We already have data for $n = 200$, $p = 0.27$, and $\theta = (3, 7)$. Also, let keep same threshold

$$|\theta_i^{n+1} - \theta_i^n| < 10^{-8},$$

for $i = 1, 2$. The EM algorithm estimates parameter θ as

$$\hat{\theta} = (2.687662, 6.828413).$$

Below there is `R` code for EM algorithm.

```r
# Generating data
Mixture <- rnormix(n, pi, mu, sigma)
Data <- c(Mixture[[1]])   # This is y
Class <- Mixture[[2]] - 1 # This is z



# Function for p(zi = 1 | yi, theta0)
prob <- function(theta)
{
   exp(-1/2*(Data-theta[2])^2) /
      (exp(-1/2*(Data-theta[1])^2)+exp(-1/2*(Data-theta[2])^2))
}

# Initial values for theta and counter for iteration
theta_new <- c(3, 1)
count <- 0

# Finally, here is iterative process of EM
repeat{
        theta_old <- theta_new
        print(theta_old)
        p <- prob(theta_old)
        # Setting n0 and n1
        n1 <- sum(p)
        n0 <- n - n1
        theta_new1 <- 1/n0 * sum(Data*(1 - p))
        theta_new2 <- 1/n1 * sum(Data*p)
        theta_new <- c(theta_new1, theta_new2)
        print(theta_new)
        count <- count + 1
        if(prod(round(theta_old, 8) == round(theta_new, 8)))
        {
            print(count)
            print(theta_new)
            break
        }
}
```

In the next table we can see what happens when we change initial values for θ.

initial value of m_0	initial value of m_1	iter
3	1	25
10	7	29
1	7	24
1	1	2
3	7	23
30	70	NaN

Here, for $m_0^0 = m_1^0 = 1$, or generally $m_0^0 = m_1^0$ we have just two iterations since

$$p(z_i = 1 \mid \theta^0) = p(z_i = 1 \mid \theta^0) = 0.5.$$

Consequently, $n_0 = n_1 = 100$ which leads us to $m_0^1 = m_1^1$. In the next iteration, we will have $m_0^2 = m_1^2 = m_0^1 = m_1^1$, so the EM algorithm will stop. This does not mean that we found a solution in just two iterations, but that algorithm converged to local instead of the global maximum. On the other hand, for $m_0^0 = 30$ and $m_1^0 = 70$ we have $p(z_i = 1 \mid \theta^0) = 0$, for all z_i, so n_1 will be zero and algorithm will be stopped at the beginning. Also, for all $m_0^0 > m_1^0$ EM algorithm gives an estimation for θ

$$\widehat{\theta} = (6.828413, 2.687662),$$

which means that classes are just switched.

8. **Extend the EM algorithm to the case where the variance σ^2 is unknown, and p is arbitrary. Extend it further to the case where the two classes have unequal (unknown) variances, and p is arbitrary.**

Let extend algorithm in the most general manner, i.e. when everything is unknown and different. First of all, we need to see what is different from this algorithm we created. Starting from the beginning we have

$$\begin{aligned} p(y_i \mid z_i, \theta) &= \left(f_{\mathcal{N}(m_0,\sigma_0^2)}(y_i \mid \theta)\right)^{1-z_i} \cdot \left(f_{\mathcal{N}(m_1,\sigma_1^2)}(y_i \mid \theta)\right)^{z_i} \\ &= \left(\frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{(y_i-m_0)^2}{2\sigma_0^2}}\right)^{1-z_i} \left(\frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(y_i-m_1)^2}{2\sigma_1^2}}\right)^{z_i}, \end{aligned}$$

which leads us to

$$p(\mathbf{y} \mid \mathbf{z}, \theta) = \prod_{i=1}^{n} \left(\frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{(y_i-m_0)^2}{2\sigma_0^2}} \right)^{1-z_i} \left(\frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(y_i-m_1)^2}{2\sigma_1^2}} \right)^{z_i}.$$

After applying the logarithmic function on the previous line we have

$$\begin{aligned}\log p(\mathbf{y} \mid \mathbf{z}, \theta) &= \sum_{i=1}^{n} \left((1-z_i) \log \frac{1}{\sqrt{2\pi\sigma_0^2}} - (1-z_i)\frac{(y_i-m_0)^2}{2\sigma_0^2} + z_i \log \frac{1}{\sqrt{2\pi\sigma_1^2}} - z_i \frac{(y_i-m_1)^2}{2\sigma_1^2} \right) \\ &= -\frac{1}{2} \sum_{i=1}^{n} \left((1-z_i) \left(\log \sigma_0^2 + \frac{(y_i-m_0)^2}{\sigma_0^2} \right) + z_i \left(\log \sigma_1^2 + \frac{(y_i-m_1)^2}{\sigma_1^2} \right) \right) - \frac{n}{2} \cdot \log 2\pi.\end{aligned}$$

Another thing that changed is p, so we have $P(Z_i = z_i) = p^{z_i}(1-p)^{1-z_i}$. That means

$$\begin{aligned}p(\mathbf{y}, \mathbf{z} \mid \theta) &= p(\mathbf{y} \mid \mathbf{z}, \theta) \cdot p(\mathbf{z}) = p(\mathbf{y} \mid \mathbf{z}, \theta) \cdot \prod_{i=1}^{n} p(z_i) \\ &= p(\mathbf{y} \mid \mathbf{z}, \theta) \cdot p^{\sum_{i=1}^{n} z_i} \cdot (1-p)^{n-\sum_{i=1}^{n} z_i}.\end{aligned}$$

Finally, we have

$$\begin{aligned}\log p(\mathbf{y}, \mathbf{z} \mid \theta) = \log p(\mathbf{y} \mid \mathbf{z}, \theta) + \log p \cdot \sum_{i=1}^{n} z_i + \log(1-p) \cdot \left(n - \sum_{i=1}^{n} \right) \\ = -\frac{1}{2} \sum_{i=1}^{n} \left((1-z_i) \left(\log \sigma_0^2 + \frac{(y_i-m_0)^2}{\sigma_0^2} \right) + z_i \left(\log \sigma_1^2 + \frac{(y_i-m_1)^2}{\sigma_1^2} \right) \right) \\ - \frac{n}{2} \cdot \log 2\pi + \log p \cdot \sum_{i=1}^{n} z_i + \log(1-p) \cdot \left(n - \sum_{i=1}^{n} z_i \right).\end{aligned}$$

Now, again we need to find the first derivatives in terms of m_0, m_1, σ_0^2, σ_1^2 and p

$$\begin{cases} \frac{\partial}{\partial m_0} \log p(\mathbf{y}, \mathbf{z} \mid \theta) = & -\frac{1}{2} \sum_{i=1}^{n} \frac{(1-z_i)(-2)(y_i-m_0)}{\sigma_0^2} = \sum_{i=1}^{n} (1-z_i) y_i - m_0 \sum_{i=1}^{n} (1-z_i) = 0 \\ \frac{\partial}{\partial m_1} \log p(\mathbf{y}, \mathbf{z} \mid \theta) = & -\frac{1}{2} \sum_{i=1}^{n} \frac{z_i(-2)(y_i-m_1)}{\sigma_1^2} = \sum_{i=1}^{n} z_i y_i - m_1 \sum_{i=1}^{n} z_i = 0 \\ \frac{\partial}{\partial \sigma_0^2} \log p(\mathbf{y}, \mathbf{z} \mid \theta) = & -\frac{1}{2} \sum_{i=1}^{n} \left(\frac{1-z_i}{\sigma_0^2} - \frac{(1-z_i)(y_i-m_0)^2}{\sigma_0^4} \right) = 0 \\ \frac{\partial}{\partial \sigma_1^2} \log p(\mathbf{y}, \mathbf{z} \mid \theta) = & -\frac{1}{2} \sum_{i=1}^{n} \left(\frac{z_i}{\sigma_1^2} - \frac{z_i(y_i-m_1)^2}{\sigma_1^4} \right) = 0 \\ \frac{\partial}{\partial p} \log p(\mathbf{y}, \mathbf{z} \mid \theta) = & \frac{1}{p} \sum_{i=1}^{n} z_i - \frac{1}{1-p} \left(n - \sum_{i=1}^{n} z_i \right) = 0. \end{cases}$$

Solution of this system is

$$\begin{aligned}
\widehat{m}_0 &= \frac{1}{n_0}\sum_{i=1}^{n}(1-z_i)y_i \\
\widehat{m}_1 &= \frac{1}{n_1}\sum_{i=1}^{n} z_i y_i \\
\widehat{\sigma}_0^2 &= \frac{1}{n_0}\sum_{i=1}^{n}(1-z_i)(y_i-\widehat{m}_0)^2 \\
\widehat{\sigma}_1^2 &= \frac{1}{n_1}\sum_{i=1}^{n} z_i(y_i-\widehat{m}_1)^2 \\
\widehat{p} &= \frac{n_1}{n_1+n_0} = \frac{n_1}{n}.
\end{aligned}$$

Again, like the last time, if we change z_i with $p(z_i = 1 \mid y_i, \theta^0)$, with the same explanation we will have our upgrade step in the EM algorithm. So, to finalize this generalized EM algorithm we need to calculate $p(z_i = 1 \mid y_i, \theta_0)$. Let's do that

$$\begin{aligned}
p(z_i = 1 \mid y_i, \theta^0) &= \frac{p(y_i \mid z_i = 1, \theta^0)p(z_i = 1 \mid \theta^0)}{p(y_i \mid \theta^0)} \\
&= \frac{p(y_i \mid z_i = 1, \theta^0)p(z_i = 1 \mid \theta^0)}{p(y_i \mid z_i = 1, \theta^0)p(z_i = 1 \mid \theta^0) + p(y_i \mid z_i = 0, \theta^0)p(z_i = 0 \mid \theta^0)} \\
&= \frac{p^0 \cdot \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(y_i - m_1^0)^2}{2\sigma_1^2}}}{(1-p^0)\cdot \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{(y_i - m_0^0)^2}{2\sigma_0^2}} + p^0 \cdot \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(y_i - m_1^0)^2}{2\sigma_1^2}}} \\
&= \frac{\frac{p^0}{\sqrt{\sigma_1^2}} e^{-\frac{(y_i - m_1^0)^2}{2\sigma_1^2}}}{\frac{1-p^0}{\sqrt{\sigma_0^2}} e^{-\frac{(y_i - m_0^0)^2}{2\sigma_0^2}} + \frac{p^0}{\sqrt{\sigma_1^2}} e^{-\frac{(y_i - m_1^0)^2}{2\sigma_1^2}}}.
\end{aligned}$$

If we now have a case when $\sigma_0^2 = \sigma_1^2$, but still unknown, in previous formulas we just need to rewrite every σ_1^2 as σ_0^2. Another change would be

$$\frac{\partial}{\partial\sigma_0^2}\log p(\mathbf{y}, \mathbf{z} \mid \theta) = -\frac{1}{2}\sum_{i=1}^{n}\left(\frac{1-z_i}{\sigma_0^2} - \frac{(1-z_i)(y_i-m_0)^2}{\sigma_0^4} + \frac{z_i}{\sigma_0^2} - \frac{z_i(y_i-m_1)^2}{\sigma_0^4}\right) = 0,$$

which leads us to estimation

$$\widehat{\sigma}_0^2 = \frac{1}{n}\sum_{i=1}^{n}\left((1-z_i)(y_i-m_0)^2 + z_i(y_i-m_1)^2\right).$$

9. **Download the data from the following URL:**

   `http://membres-timc.imag.fr/Olivier.Francois/data2.txt`

   Apply the EM algorithm to the data. Evaluate the convergence of the algorithm by testing several initial values of θ^0. Report estimates for m_0 and m_1, and display $p(\mathbf{z} \mid \mathbf{y})$ by using the `barplot` command to visualize the probability matrix of size $n \times 2$.

   First of all, let us look at the data (Figure 6). We can see that m_0 should be around 0, and the m_1 should be around 3, so we will put those for initial values. Also, let put $\sigma_0^2 = 2$ and $\sigma_1^2 = 1$. Finally, it looks like p is around 0.5, so for the first run let us put $p = 0.5$.

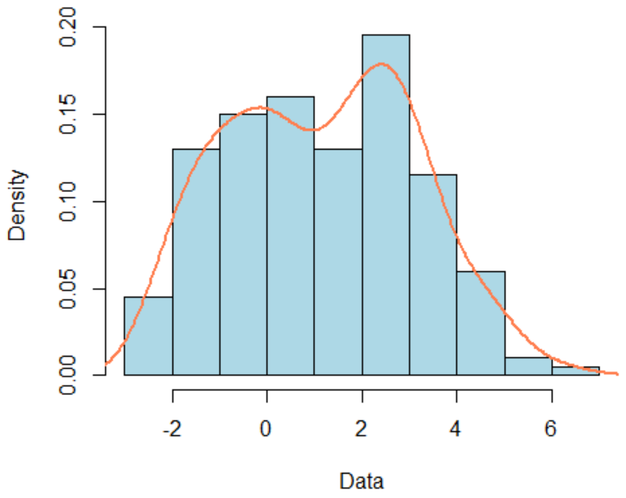

Figure 6: Histogram of data from Gaussian mixture model

Algorithm will return

$$\theta = (m_0, m_1, \sigma_0^2, \sigma_1^2, p) = (-0.8965767, 2.2779280, 0.7593524, 2.0207150, 0.6456336).$$

It looks pretty correct. Let test convergence by putting different initial values, and see how many steps EM algorithm need for convergence.

m_0	m_1	σ_0^2	σ_1^2	p	`iter`
0	3	2	1	0.5	379
-1	4	1	1	0.1	344
1	2	4	2	0.9	340
4	2	1	2	0.2	343
1	1	1	2	0.2	480
1	1	1	1	0.2	2

Again, for the last scenario, we have the same situation as before. But for the penultimate case, we do not have that problem, just more iterations. Also, if we put $m_0 > m_1$ we will get switched estimations. Below in Figure 7, we can see the `barplot` for probabilities $p(\mathbf{z} \mid \mathbf{y})$.

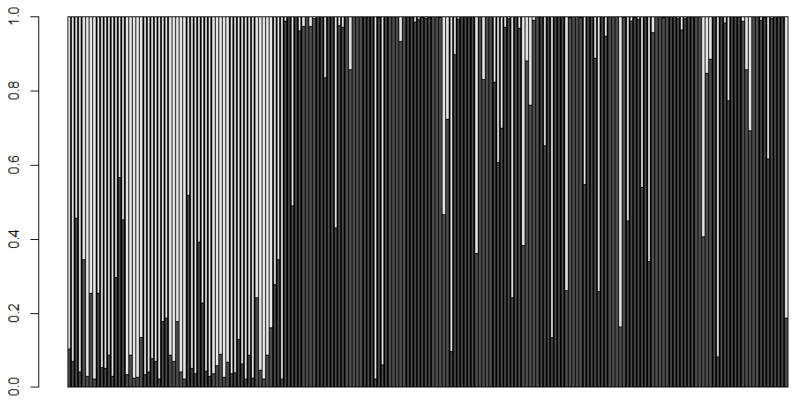

Figure 7: Barplot, where white (black) represents probabilities for z_i to be in class P_0 (P_1)

10. **Install the `R` package `mclust` from the CRAN web site. Look at the different options of the `Mclust` function (models and outputs), and run the `Mclust` command on the data for $G = 1$ to 5.**

    Firstly, let see what is this function, i.e. what `R` help says about it.

    ```
    Model-based clustering based on parameterized finite Gaussian mixture models.  Models
    are estimated by EM algorithm initialized by hierarchical model-based agglomerative
    clustering.  The optimal model is then selected according to BIC.
    ```

    Also, let look at some input and output variables for `Mclust` function.

    ```
    Mclust(data, G = NULL, modelNames = NULL,
          prior = NULL,
          control = emControl(),
          initialization = NULL,
          warn = mclust.options("warn"),
          x = NULL,
          verbose = interactive(), ...)
    ```

    where explanations for some of those input variables are listed below.

    - `data` - A numeric vector, matrix, or data frame of observations. Categorical variables are not allowed. If a matrix or data frame, rows correspond to observations (n) and columns correspond to variables (d).

- `G` - An integer vector specifying the numbers of mixture components (clusters) for which the BIC is to be calculated. The default is $G = 1 : 9$.
- `modelNames` - A vector of character strings indicating the models to be fitted in the EM phase of clustering. The default is:
  - for univariate data ($d = 1$): `c("E", "V")`
  - for multivariate data ($n > d$): all the models available in `mclust.options("emModelNames")`
  - for multivariate data ($n \leqslant d$): the spherical and diagonal models, i.e. `c("EII", "VII", "EEI", "EVI", "VEI", "VVI")`

  The help file for `mclustModelNames` describes the available models. Here are some of them
  - `univariate mixture`
    * `"E"` - equal variance (one-dimensional)
    * `"V"` - variable/unqual variance (one-dimensional)
  - `multivariate mixture`
    * `"EII"` - spherical, equal volume
    * `"VII"` - spherical, unequal volume
    * `"EEI"` - diagonal, equal volume and shape
    * `"VEI"` - diagonal, varying volume, equal shape
    * `"EVI"` - diagonal, equal volume, varying shape

An object of class `'Mclust'` providing the optimal (according to BIC) mixture model estimation.

The details of the output components are as follows:

- `call` - The matched call.
- `data` - The input data matrix.
- `modelName` - A character string denoting the model at which the optimal BIC occurs.
- `n` - The number of observations in the data.
- `d` - The dimension of the data.
- `G` - The optimal number of mixture components.
- `BIC` - All BIC values.
- `bic` - Optimal BIC value.
- `loglik` - The log-likelihood corresponding to the optimal BIC.
- `df` - The number of estimated parameters.
- `hypvol` - The hypervolume parameter for the noise component if required, otherwise set to NULL (see `hypvol`).
- `parameters` - A list with the following components:
  - `pro` - A vector whose k^{th} component is the mixing proportion for the k^{th} component of the mixture model. If missing, equal proportions are assumed.
  - `mean` - The mean for each component. If there is more than one component, this is a matrix whose k^{th} column is the mean of the k^{th} component of the mixture model.

  - `variance` - A list of variance parameters for the model. The components of this list depend on the model specification. See the help file for `mclustVariance` for details.
- `z` - A matrix whose $[i, k]^{\text{th}}$ entry is the probability that observation i in the test data belongs to the k^{th} class.
- `classification` - The classification corresponding to `z`, i.e. map(z).
- `uncertainty` - The uncertainty associated with the `classification`.

When calling `Mclust(data2)` function suggests that we have the model `("E", 2)`, which means that `modelNames` is `"E"`, i.e. equal variance in one dimension and `G` is 2, i.e. we have two classes. If we call `Mclust(data2)$BIC` we will have three best models based on BIC criteria, which are `("E", 2)`, `("V", 2)` and `("E", 3)`. Calling `Mclust(data2)$df` will tell us that we estimated 4 parameters, m_0, m_1, σ_0^2 and p. Reason for this is due to the fact that `Mclust(data2)` estimated that variances are equal, i.e. $\sigma_0^2 = \sigma_1^2$. So, the estimated parameters, which can be seen as `Mclust(data2)$parameters` are

$$\theta = (m_0, m_1, \sigma_0^2, p) = (-0.4962051, 2.7040097, 1.321432, 0.515339).$$

We see that this is not close to the results we have got from our algorithm. The reason for having different results are since we found maximum analytically, but `Mclust` does it numerically. Let's try to call `Mclust(data2, G = 2, modelNames = "V")$parameters`, which means we want 2 classes that have unequal variance. It will give us

$$\theta = (m_0, m_1, \sigma_0^2, \sigma_1^2, p) = (-0.8417746, 2.3495063, 0.807073, 1.903795, 0.6250671),$$

which is now more similar to our result. If we call `Mclust(data2, G = 2, modelNames = "V")$z` we will have a matrix of dimension $n \times 2$ with probabilities. In the first row, we have probabilities for every z_i be in the first class, and in the second row probabilities for a second class. We can make a `barplot` (Figure 8) to compere with the one from the previous question. In order to make `barplot` we need to transpose data with probabilities and subtract them from 1. Now, we have everything in order like in our EM algorithm. This new `barplot` seems pretty similar to one we had earlier.

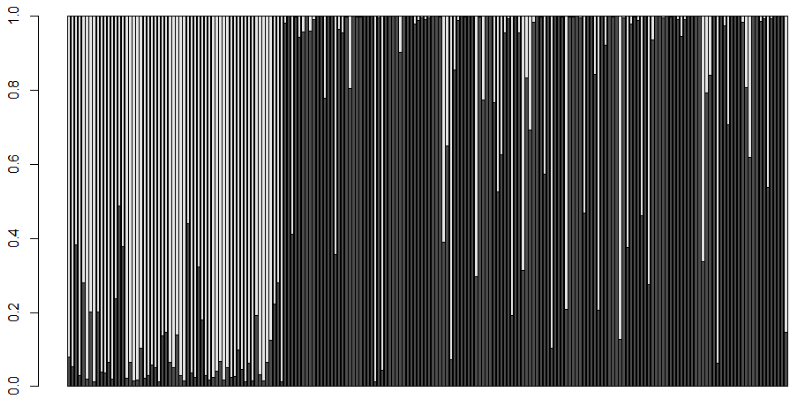

Figure 8: Barplot, where white (black) represents probabilities for z_i to be in class P_0 (P_1)

We can now see what happens if we change G. For `G = 1` we have a trivial case, where everything is in one class. So, the function will model data as a normal distribution, i.e. parameters will be

$$\theta = (m_0, \sigma_0^2) = (1.15299, 3.879366).$$

Note here, that p does not need to be estimated. It must be $p = 1$ because we have just one class. We have already seen what happened with the model for `G = 2`, so let put `G = 3`. Now, we will have 3 classes so it will not be enough to have just p, we need p_1, the probability that data is in the first class, and p_2, the probability that data is in the second class. Note here, that the probability that data is in the zero class is $1 - p_1 - p_2$. If we do not specify that `modelNames = "V"`, the function will fit model as `"E"`, i.e. the best model will be with equal variances. So, we have estimations

$$\begin{aligned}\theta &= (m_0, m_1, m_2, \sigma_0^2, p_1, p_2)\\ &= (-0.6694795, 2.2911847, 4.2693123, 0.9257204, 0.46541858, 0.09000615).\end{aligned}$$

Finally, for `G = 4` and `G = 5`, we will not provide the estimations, because there will be 8 and 10 parameters, respectively. That is if we fit model with equal variances. If the model is with different variances, there would be 11 and 14 parameters, respectively. Instead of this, we will provide fitted density functions for `G = 4` and `G = 5` in Figure 9. Here, also we let function decide should it be `modelNames = "E"` or `modelNames = "V"`. Function suggested being `"E"` for any number of classes `G`.

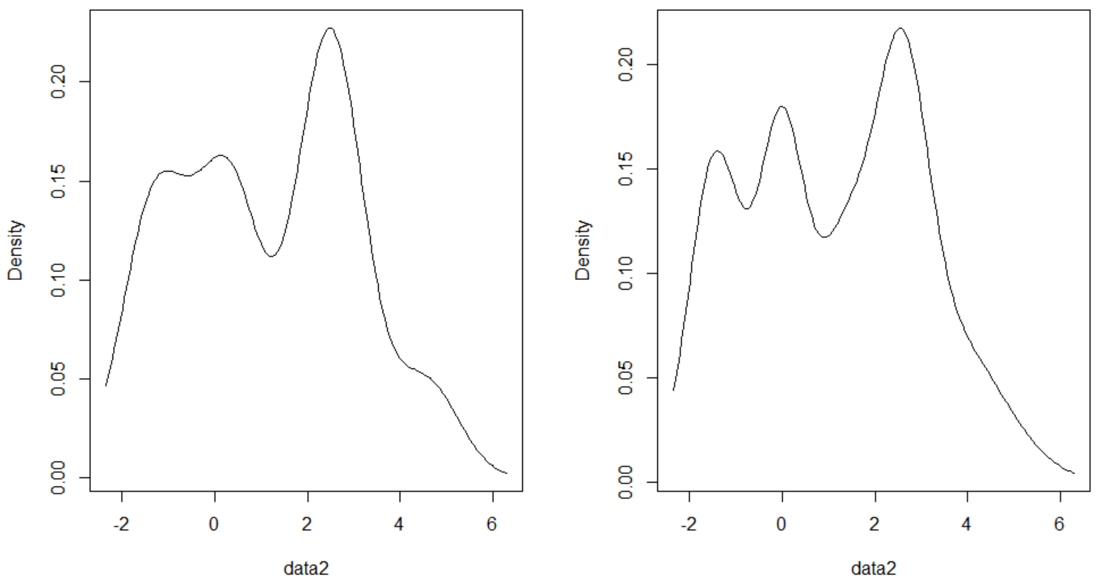

Figure 9: Estimated density for data for `G = 4` and `G = 5`, respectively

We can see that density functions are more or less same, it may suggest that we overfit our data, and maybe there is no reason for 5 classes.

11. **Find a definition of the Bayesian Information Criterion (BIC). Discuss the choice of a model for the data using the BIC.**

The Bayesian Information Criterion (BIC) is a criterion for model selection among a finite set of models. It is based, in part, on the log-likelihood function, and it is closely related to Akaike Information Criterion (AIC).

When fitting models, it is possible to increase the likelihood by adding parameters, but doing so may result in overfitting. The BIC resolves this problem by introducing a penalty term for the number of parameters in the model. The BIC is an asymptotic result derived under the assumption that the data distribution is in the exponential family.

Definition 1. Let x be the observed data, n be the number of data points in x, the number of observations, or equivalently, the sample size. Let k be the number of free parameters θ to be estimated, $p(x \mid \theta)$ be the probability of the observed data given the parameters; or, the likelihood of the parameters given the dataset. Finally, let $\widehat{L}$ be the maximized value of the likelihood function for the estimated model. We define BIC as

$$BIC = -2 \cdot \log \widehat{L} + k \cdot \log n.$$

Note. In library `mclust`, the sign of BIC is different from the definition, i.e. when calculating BIC `mclust` uses formula

$$BIC = 2 \cdot \log \widehat{L} - k \cdot \log n,$$

which means that we should look for the greatest values instead of the smallest one.

For our data, when calling `Mclust(data2)$BIC` we have

```
Bayesian Information Criterion (BIC):
              E          V
     1  -849.3064  -849.3064
     2  -842.3867  -843.8740
     3  -848.1377  -859.8732
     4  -854.5695  -869.2512
     5  -864.0680  -884.6254
     6  -874.6795  -899.7729
     7  -885.4543  -904.8113
     8  -896.0735  -922.4495
     9  -906.6151  -938.1552
Top 3 models based on the BIC criterion:
        E,2        V,2        E,3
  -842.3867  -843.8740  -848.1377
```

We can see from the above table that best models according to BIC are with 2 or 3 classes. For two classes it is better to assume that variances are equal because we have one parameter less to estimate, while from data we can see that σ_0^2 and σ_1^2 should be similar. We also can see that for more parameters BIC is smaller and smaller, so we definitely, should not overfit our model, like for example setting a number of classes to 5. Which we already know from the Figure 9.

Challenge and evaluation rule. Download the data from the following URL:

`http://membres-timc.imag.fr/Olivier.Francois/matrix_2019.txt`

The data consists of a matrix of 499 rows and 6421 columns with entries in 0, 1, 2. The objective of the challenge is to evaluate the number of clusters (for rows) in the data set and to assign a cluster label to each row. The result is a list of 499 cluster labels, one for each row. The output file must contain the resulting list formatted as a sequence of integer values separated by space characters as follows

```
12 12 1 6 7 6 6 3 11 11 ...
```

A `README` file describing the options used when analyzing the data is required. The results will be evaluated based on the confusion matrix and the number of wrongly classified rows.

Important comments: Use a dimension reduction algorithm such as principal component analysis or multidimensional scaling to reduce the dimension of the data set **before** applying model-based clustering algorithms. Then, prefer using the `Mclust` algorithm rather than reprogramming your EM method.

Solution. Like the previous challenge, instead of the `README` file, all methods will be discussed here. First of all, we need to reduce the dimension of 6421, our first attempt was with PCA (principal component analysis), so let us briefly introduce it. PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component, in turn, has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors (each being a linear combination of the variables and containing n observations) are an uncorrelated orthogonal basis set. In our case, it looks like this.

```r
matrix.pca <- prcomp(matrix_2019)

# We need proportion of variance to determine numbers of components
std_dev <- matrix.pca$sdev
pr_var <- std_dev^2
prop_varex <- pr_var/sum(pr_var)

# Plotting to see how many components we need
plot(cumsum(prop_varex), xlab = "Principal Component",
     ylab = "Cumulative Proportion of Variance Explained",
     type = "b")

# We can see that around 450 components explained approximately
# 95% of total variance
(cumsum(prop_varex))[450]
# It is 94.4%
```

We saw that we need 450 or maybe even 400 components out of 499 (Figure 10). It is not so good to reduce the dimension from 499 to 450. But actually, what happened is not a reduction from 499 to 450, but from 6421 to 450, because our new data looks like

$$Z = XA,$$

where X is original data of dimension 499×6421 and A is a rotation matrix obtained from PCA, i.e. `matrix.pca$rotation`, that has a dimension of 6421×450. That means our new data has a dimension of 499×450 which is much better then original data. On the other hand, if we want to do `Mclust` on new data we still have data of high dimension. For that reason, we will try with t-SNE instead. Then, we will have only 499×2 dimensions.

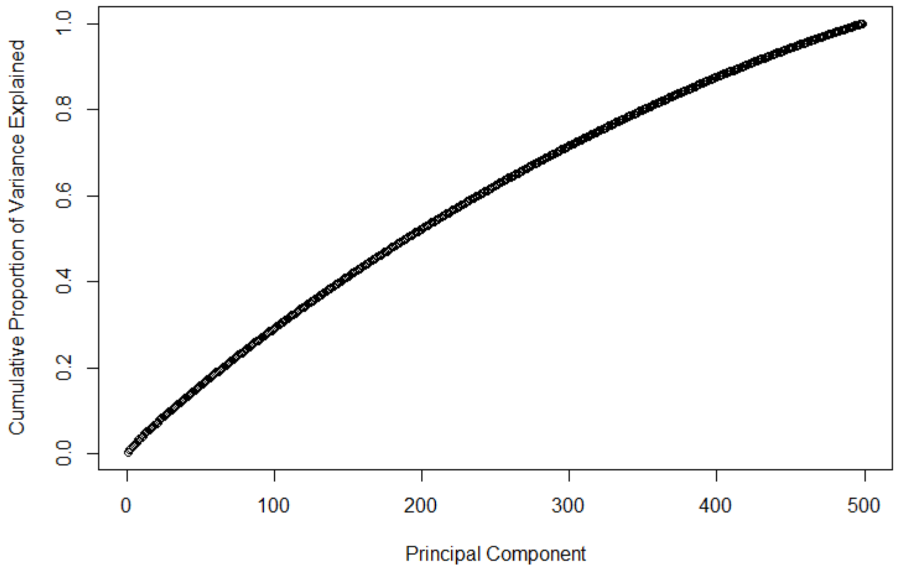

Figure 10: Cumulative proportion of variance explained for principal components

T-distributed Stochastic Neighbor Embedding (t-SNE) is a nonlinear dimensionality reduction technique well-suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions. Specifically, it models each high-dimensional object by a two or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability. The t-SNE algorithm comprises two main stages. First, t-SNE constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a high probability of being picked while dissimilar points have an extremely small probability of being picked. Second, t-SNE defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the Kullback-Leibler divergence between the two distributions with respect to the locations of the points on the map. While t-SNE plots often seem to display clusters, the visual clusters can be influenced strongly by the chosen parameterization and therefore a good understanding of the parameters for t-SNE is necessary. In library `Rtsne` there is `Rtsne()` function for doing t-SNE on given data. This function is different from `tsne` just because it is implemented in `C++`, which means it is faster. Some of the important parameters of this function are `X` - data matrix, `dims` - output dimensionality (defaulted `dims = 2`) and most importantly `perplexity`. The performance of t-SNE is fairly robust under different settings of the perplexity. The most appropriate value depends on the density of the data. Loosely speaking, one could say that a larger or denser dataset requires a larger perplexity. Typical values for the perplexity range between 5 and 50. The default value for perplexity is 30. We tied perplexity 2, 5, 10, 30, 50 and 100, and from the plot, we could see that not much is changed for bigger perplexities, so we left the default value. Let us look at the code now.

```
fit.tsne <- Rtsne(matrix_2019, max_iter = 5000)

tsne.data <- data.frame(fit.tsne$Y)
colnames(tsne.data) <- c("tsne1", "tsne2")

# Variable for data where we will add new column with class
info.data <- tsne.data

# Ploting classes we got from t-SNE
ggplot(tsne.data, aes(x = tsne1, y = tsne2)) +
      geom_point(alpha = 0.1) + theme_bw()

# Applying Mclust on data from t-SNE
# We let G to be any number from 1 to 25, and the optimal one is 15
mc <- Mclust(tsne.data, G = 1:25)

# Adding new variable for classification
info.data$mclust <- factor(mc$classification)

# Data frame having position on plane for every class.
# We need this for ploting a graph with label of every class
mc.cent <- info.data %>% group_by(mclust) %>%
            select(tsne1, tsne2) %>% summarize_all(mean)

# Ploting 2D plane with classes and label for each class
ggplot(info.data, aes(x = tsne1, y = tsne2, colour = mclust)) +
        geom_point(alpha = 0.3) +  theme_bw() +
        geom_label_repel(aes(label = mclust), data = mc.cent) +
        guides(colour = FALSE)
```

After applying previous we have a class for each row, which is represented in Figure 11. Also, the result which is needed, i.e. a vector of class for each row is this.

```
 1 2 3 3 3 4 1 7 6 7 8 5 3 9 7 2 4 10 11 10 12 8 7 6 2 7 11 11 4 7 6 6 13 1 9 14 9
13 14 13 2 10 4 10 4 9 14 15 13 2 10 10 1 12 12 1 5 13 2 9 3 8 6 2 13 7 2 1 14 14 9
1 11 7 11 7 5 5 12 6 9 11 8 7 8 6 12 10 2 1 1 8 11 5 11 5 1 9 2 8 9 6 5 3 5 9 15 13
10 3 7 10 9 13 13 6 13 10 10 1 4 8 10 10 8 9 7 3 9 12 7 6 5 10 12 2 7 6 12 1 4 11 2
```

```
11 9 12 15 6 10 9 6 2 13 2 7 9 2 1 9 7 1 3 14 13 8 3 13 8 7 6 1 6 11 15 5 8 10 11 1
14 8 10 14 5 11 2 5 1 6 14 13 4 2 5 2 10 9 5 1 10 11 2 6 14 9 12 5 5 1 6 13 6 6 9 8
 13 4 7 6 3 9 7 6 7 3 7 13 6 15 9 10 15 1 3 9 9 1 13 12 5 7 6 4 5 1 1 4 4 9 3 9 12
 10 4 1 9 5 4 6 13 9 1 12 3 9 8 6 2 10 15 9 4 1 13 14 8 15 11 9 12 7 4 14 9 9 3 1 8
 2 8 1 12 4 9 6 1 6 7 6 7 12 6 7 3 13 5 9 1 2 11 8 3 1 10 9 12 12 2 8 13 6 3 3 11 6
6 10 2 3 13 6 7 3 4 2 14 9 2 7 1 14 4 8 3 12 3 9 4 14 12 14 8 10 5 15 8 9 3 12 10 1
 6 1 9 5 9 11 4 10 12 6 9 14 2 2 7 13 5 12 13 4 4 12 11 13 4 3 1 6 1 13 2 13 2 11 1
12 9 5 5 6 2 5 13 8 6 3 2 1 5 9 4 5 9 5 10 14 10 9 14 11 13 8 2 2 4 14 14 8 4 1 9 3
 9 4 13 15 12 6 15 6 3 1 1 1 2 1 6 14 8 9 8 4 7 4 15 15 15 3 9 15 4 4 7 15 11 5 6 3
      4 7 15 2 15 3 7 14 12 10 6 14 10 2 12 12 12 4 6 12 6 5 9 6 6 13 9 8 8 8
```

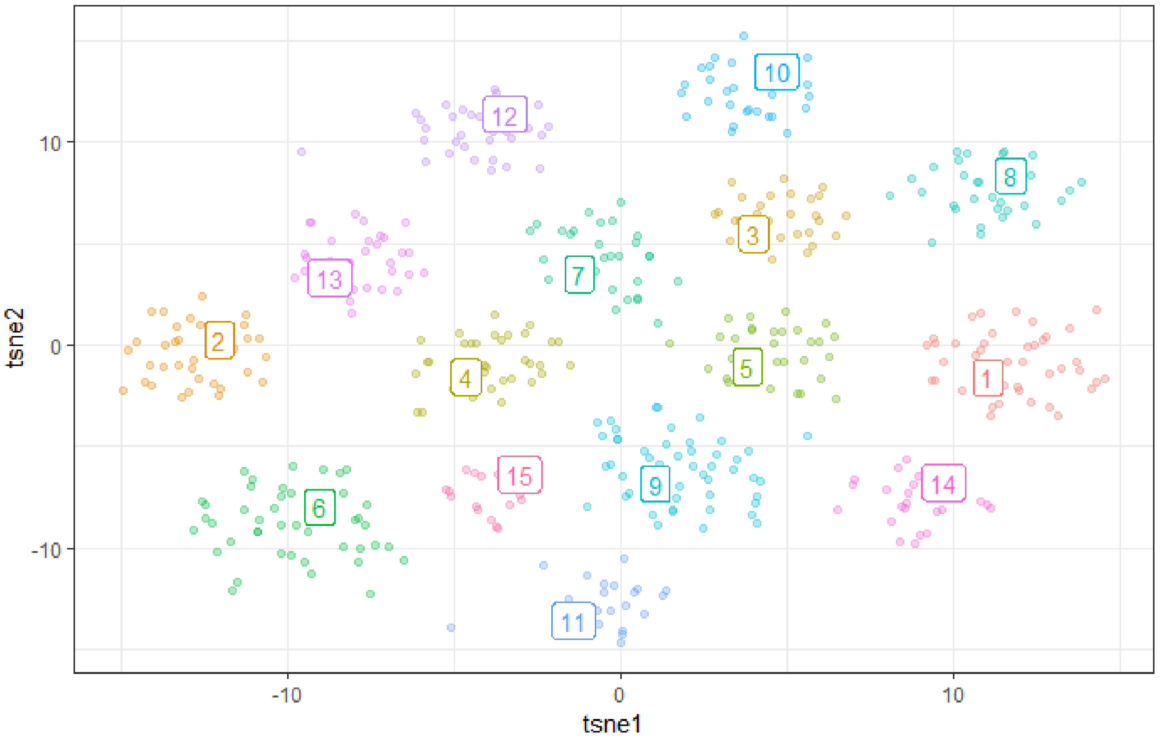

Figure 11: Classification on projected plane after t-SNE

Problem 3.

ABO groups and genetics

A geneticist studies genotypes at the *ABO* locus (alleles A, B, O) for blood samples from n individuals, and gets observations for phenotypes for each individual (blood groups). Four distinct phenotypes can be observed

- type A corresponds to genotypes A/A and A/O (sample size n_A),
- type B corresponds to genotypes B/B and B/O (sample size n_B),

- type AB corresponds to genotype A/B (sample size n_{AB}),
- type O corresponds to genotype O/O (sample size n_O)

where we suppose that A/O and O/A are the same genotype (the same for B), and

$$n = n_A + n_B + n_{AB} + n_O.$$

We say that types A and B are co-dominants, whereas type O is recessive and is observed only if an individual carries to copies of allele O. We want to estimate the frequency p_A of allele A in the sampled population.

1. **We first assume that all allele frequencies are known parameters. Using the Hardy-Weinberg principle, show that the expected number of genotypes A/A in a sample of size n is equal to**

$$n_{A/A} = n_A \cdot \frac{p_A^2}{p_A^2 + 2p_A p_O}.$$

According to the Hardy-Weinberg law, the genotype frequencies are

$$\begin{pmatrix} A/A & A/O & A/B & B/B & B/O & O/O \\ p_A^2 & 2p_A p_O & 2p_A p_B & p_B^2 & 2p_B p_O & p_O^2 \end{pmatrix}.$$

Note here that previous matrix represent distribution of genotypes so it must be

$$1 = p_A^2 + 2p_A p_O + 2p_A p_B + p_B^2 + 2p_B p_O + p_O^2 = (p_A + p_B + p_O)^2.$$

Knowing that type A can have either A/A or A/O genotypes, in our case, we have that frequency for A/A is given by

$$\frac{n_{A/A}}{n_A} = \frac{p_A^2}{p_A^2 + 2p_A p_O},$$

which leads us to

$$n_{A/A} = n_A \cdot \frac{p_A^2}{p_A^2 + 2p_A p_O}.$$

2. **Find a similar equation for $n_{A/O}$.**

We can calculate $n_{A/O}$ in the same way, or like

$$n_{A/O} = n_A - n_{A/A} = n_A - n_A \cdot \frac{p_A^2}{p_A^2 + 2p_A p_O} = n_A \cdot \frac{2p_A p_O}{p_A^2 + 2p_A p_O}.$$

3. **Suppose we know $n_{A/A}$, $n_{A/O}$, n_{AB}. Give an estimate $\widehat{p}_A$ of the frequency p_A.**

We will do this by the MLE algorithm. But firstly, we need to introduce some random variables. Let N_A, N_B, N_{AB}, and N_O be random variables corresponding to a number of people with certain blood type. That is our observed data. On the other hand, complete data is $N_{A/A}$, $N_{A/O}$, $N_{B/B}$, $N_{B/O}$, N_{AB} and N_O. Genotype counts

$$\mathbf{N} = (N_{A/A}, N_{A/O}, N_{B/B}, N_{B/O}, N_{AB}, N_O)$$

has a multinomial distribution with probability mass function

$$\begin{aligned} p(\mathbf{n} \mid \theta) = P(\mathbf{N} = \mathbf{n} \mid \theta) &= \binom{n}{n_{A/A}, n_{A/O}, n_{B/B}, n_{B/O}, n_{AB}, n_O} \\ &\cdot \left(p_A^2\right)^{n_{A/A}} (2p_A p_O)^{n_{A/O}} \left(p_B^2\right)^{n_{B/B}} (2p_B p_O)^{n_{B/O}} (2p_A p_B)^{n_{AB}} \left(p_O^2\right)^{n_O}, \end{aligned}$$

where $\mathbf{n} = (n_{A/A}, n_{A/O}, n_{B/B}, n_{B/O}, n_{AB}, n_O)$, and $\theta = (p_A, p_B, p_O)$. Also, the first thing in the previous formula is multinomial coefficient defined as

$$\binom{n}{n_{A/A}, n_{A/O}, n_{B/B}, n_{B/O}, n_{AB}, n_O} = \frac{n!}{n_{A/A}! n_{A/O}! n_{B/B}! n_{B/O}! n_{AB}! n_O!},$$

for shorter notation let's call it C_n. Our goal is to apply MLE for θ so we will compute log-likelihood function

$$\begin{aligned} \log p(\mathbf{n} \mid \theta) = \log C_n &+ 2n_{A/A} \cdot \log p_A + n_{A/O} \cdot \log(2p_A p_O) + 2n_{B/B} \cdot \log p_B \\ &+ n_{B/O} \cdot \log(2p_B p_O) + n_{AB} \cdot \log(2p_A p_B) + 2n_O \cdot \log p_O. \end{aligned}$$

We can think that this time, like every other time, we should find first derivatives in terms of θ and make them equal to zero. But we would be wrong. The problem is, in this case, we have an additional constraint

$$p_A + p_B + p_O = 1,$$

so this maximization problem needs a Lagrange multiplier to be properly solved. So, we need to maximize the next function

$$L(\theta, \lambda) = \log p(\mathbf{n} \mid \theta) + \lambda (1 - p_A - p_B - p_O).$$

For $L(\theta, \lambda)$ we need to find derivatives and equal them to zero

$$\begin{cases} \dfrac{\partial}{\partial p_A} L(\theta, \lambda) = & \dfrac{2n_{A/A}}{p_A} + \dfrac{n_{A/O}}{p_A} + \dfrac{n_{AB}}{p_A} - \lambda = 0 \\ \dfrac{\partial}{\partial p_B} L(\theta, \lambda) = & \dfrac{2n_{B/B}}{p_B} + \dfrac{n_{B/O}}{p_B} + \dfrac{n_{AB}}{p_B} - \lambda = 0 \\ \dfrac{\partial}{\partial p_O} L(\theta, \lambda) = & \dfrac{n_{A/O}}{p_O} + \dfrac{n_{B/O}}{p_O} + \dfrac{2n_O}{p_O} - \lambda = 0 \\ \dfrac{\partial}{\partial \lambda} L(\theta, \lambda) = & 1 - p_A - p_B - p_O = 0 \end{cases}$$

From the previous system we have

$$\begin{cases} p_A = & \dfrac{2n_{A/A} + n_{A/O} + n_{AB}}{\lambda} \\ p_B = & \dfrac{2n_{B/B} + n_{B/O} + n_{AB}}{\lambda} \\ p_O = & \dfrac{2n_O + n_{A/O} + n_{B/O}}{\lambda} \\ 1 = & p_A + p_B + p_O \end{cases}$$

After summing first three equations, we find that

$$\lambda = 2(n_{A/A} + n_{A/O} + n_{B/B} + n_{B/O} + n_O) = 2n,$$

so, we finally have estimations

$$\begin{aligned}\widehat{p}_A &= \frac{2n_{A/A} + n_{A/O} + n_{AB}}{2n}\\ \widehat{p}_B &= \frac{2n_{B/B} + n_{B/O} + n_{AB}}{2n}\\ \widehat{p}_O &= \frac{2n_O + n_{A/O} + n_{B/O}}{2n}\end{aligned}$$

4. **Use a circular (iterative) argument to compute an estimate $\widehat{p}_A$ from the data.**

   First, we should remember that we know n_A, n_B, n_{AB} and n_O. Our goal is to estimate p_A, p_B and p_O, but we saw in the previous question that we need also $n_{A/A}$, $n_{A/O}$, $n_{B/B}$, and $n_{B/O}$. We can choose initial values $\theta^0 = (p^0_A, p^0_B, p^0_O)$. From the first two questions, we know

$$\begin{aligned}n^0_{A/A} &= n_A \cdot \frac{\left(p^0_A\right)^2}{\left(p^0_A\right)^2 + 2p^0_A p^0_O}\\ n^0_{A/O} &= n_A \cdot \frac{2p^0_A p_O}{\left(p^0_A\right)^2 + 2p^0_A p^0_O}\\ n^0_{B/B} &= n_B \cdot \frac{\left(p^0_B\right)^2}{\left(p^0_B\right)^2 + 2p^0_B p^0_O}\\ n^0_{B/O} &= n_B \cdot \frac{2p^0_B p^0_O}{\left(p^0_B\right)^2 + 2p^0_B p^0_O}.\end{aligned}$$

   Now, putting the previous result in equations from question 3 we can compute

$$\begin{aligned}p^1_A &= \frac{2n^0_{A/A} + n^0_{A/O} + n_{AB}}{2n}\\ p^1_B &= \frac{2n^0_{B/B} + n^0_{B/O} + n_{AB}}{2n}\\ p^1_O &= \frac{2n_O + n^0_{A/O} + n^0_{B/O}}{2n}.\end{aligned}$$

   Now we can compute new values $n^1_{A/A}$, $n^1_{A/O}$, $n^1_{B/B}$ and $n^1_{B/O}$, and then p^2_A, p^2_B and p^2_O, (here 2 represents the second iteration and not square) and so on.

5. **Write an EM algorithm for estimating all allele frequencies.**

   Before programming the EM algorithm, let see what we will need.

   (E) We need to find function $Q(\theta, \theta^0)$. But, for that, we need to know what is incomplete data $\mathbf{Y}$ and what is missing data $\mathbf{Z}$. In our case incomplete data, let call it $\mathbf{N_y}$, is

$$\mathbf{N_y} = (N_A, N_B, N_{AB}, N_O),$$

and missing data $\mathbf{N_z}$ is

$$\mathbf{N_z} = (N_{A/A}, N_{A/O}, N_{B/B}, N_{B/O}).$$

Now, we compute $Q(\theta, \theta^0)$ by definition

$$\begin{aligned}
Q(\theta, \theta^0) &= \mathbb{E}\left[\log P(\mathbf{N_y} = \mathbf{n_y}, \mathbf{N_z} \mid \theta) \mid \mathbf{N_y} = \mathbf{n_y}, \theta^0\right] \\
&= \mathbb{E}\big[\log C_n + 2N_{A/A} \cdot \log p_A + N_{A/O} \log(2p_A p_O) + 2N_{B/B} \cdot \log p_B \\
&\qquad + N_{B/O} \log(2p_B p_O) + n_{AB} \cdot \log(2p_A p_B) + 2n_O \log p_O \mid \mathbf{N_y} = \mathbf{n_y}, \theta^0\big] \\
&= \mathbb{E}\left[2N_{A/A} \mid N_A = n_A, \theta^0\right] \cdot \log p_A + \mathbb{E}\left[N_{A/O} \mid N_A = n_A, \theta^0\right] \cdot \log(2p_A p_O) \\
&+ \mathbb{E}\left[2N_{B/B} \mid N_B = n_B, \theta^0\right] \cdot \log p_B + \mathbb{E}\left[N_{B/O} \mid N_B = n_B, \theta^0\right] \cdot \log(2p_B p_O) \\
&+ n_{AB} \cdot \log(2p_A p_B) + 2n_O \cdot \log p_O + \log C_n.
\end{aligned}$$

Now, we need to know distributions for $(N_{A/A} \mid N_A = n_A, \theta^0)$, $(N_{A/O} \mid N_A = n_A, \theta^0)$, and $(N_{B/B} \mid N_B = n_B, \theta^0)$, $(N_{B/O} \mid N_B = n_B, \theta^0)$ in order to compute expectation. Due the fact that $N_{A/A} + N_{A/O} = N_A$ and $N_{B/B} + N_{B/O} = N_B$ we know those are binomial distributions, i.e.

$$\begin{aligned}
N_{A/A} \mid N_A = n_A, \theta^0 &\sim \mathcal{B}\left(n_A, \frac{\left(p_A^0\right)^2}{\left(p_A^0\right)^2 + 2p_A^0 p_O^0}\right), \\
N_{A/O} \mid N_A = n_A, \theta^0 &\sim \mathcal{B}\left(n_A, \frac{2p_A^0 p_O^0}{\left(p_A^0\right)^2 + 2p_A^0 p_O^0}\right), \\
N_{B/B} \mid N_B = n_B, \theta^0 &\sim \mathcal{B}\left(n_B, \frac{\left(p_B^0\right)^2}{\left(p_B^0\right)^2 + 2p_B^0 p_O^0}\right), \\
N_{B/O} \mid N_B = n_B, \theta^0 &\sim \mathcal{B}\left(n_B, \frac{2p_B^0 p_O^0}{\left(p_B^0\right)^2 + 2p_B^0 p_O^0}\right).
\end{aligned}$$

We know that expectation of $\mathcal{B}(n, p)$ is np, which means that expected values we need for $Q(\theta, \theta^0)$ are actually $n_{A/A}^0$, $n_{A/O}^0$, $n_{B/B}^0$ and $n_{B/O}^0$, respectively. Now, we have

$$\begin{aligned}
Q(\theta, \theta^0) &= n_{A/A}^0 \cdot \log p_A + n_{A/O}^0 \cdot \log(2p_A p_O) + n_{B/B}^0 \cdot \log p_B \\
&\quad + n_{B/O}^0 \cdot \log(2p_B p_O) + n_{AB} \cdot \log(2p_A p_B) + 2n_O \cdot \log p_O + \log C_n.
\end{aligned}$$

(M) Note that we already maximized previous function (the only difference is we had $n_{A/A}$ instead of $n_{A/A}^0$, and so on). It means we have the next upgrade rule from EM algorithm

$$\begin{aligned}
p_A^1 &= \frac{2n_{A/A}^0 + n_{A/O}^0 + n_{AB}}{2n} \\
p_B^1 &= \frac{2n_{B/B}^0 + n_{B/O}^0 + n_{AB}}{2n} \\
p_O^1 &= \frac{2n_O + n_{A/O}^0 + n_{B/O}^0}{2n}.
\end{aligned}$$

We have our upgrade rule, so we can write an algorithm now.

```r
# Function for calculating nAA, nAO, nBB and nBO
find_n <- function(theta)
{
   nAA <- nA*theta[1]^2/(theta[1]^2 + 2*theta[1]*theta[3])
   nAO <- nA - nAA
   nBB <- nB*theta[2]^2/(theta[2]^2 + 2*theta[2]*theta[3])
   nBO <- nB - nBB
   n_new <- c(nAA, nAO, nBB, nBO)
}

# Initial values and counter for iterations
theta_new <- c(1/3, 1/3, 1/3)
count <- 0

# EM algorithm
repeat{
   theta_old <- theta_new
   print(theta_old)
   n_new <- find_n(theta_old)
   theta_new1 <- 1/(2*n) * (2*n_new[1] + n_new[2] + nAB)
   theta_new2 <- 1/(2*n) * (2*n_new[3] + n_new[4] + nAB)
   theta_new3 <- 1/(2*n) * (n_new[2] + n_new[4] + 2*nO)
   theta_new <- c(theta_new1, theta_new2, theta_new3)
   print(theta_new)
   count <- count + 1
   if(prod(round(theta_old, 8) == round(theta_new, 8)))
   {
     print(count)
     print(theta_new)
     break
   }
}
```

6. **Application: We observe $n = 521$ cases of peptic ulcer disease, for which $n_A = 186$, $n_B = 38$, $n_{AB} = 13$ and $n_O = 284$. Find estimates for p_A, p_B and p_O.**

   For this problem, our algorithm will return next estimation

$$\widehat{\theta} = (\widehat{p}_A, \widehat{p}_B, \widehat{p}_C) = (0.21359094, 0.05014533, 0.73626373).$$

   Below is a table of a number of iterations for different initial values.

`initial value of` p_A	`initial value of` p_B	`initial value of` p_B	`iter`
0.33	0.33	0.33	10
0.2	0.05	0.75	9
0.2	0.05	0.7	8
0.1	0.1	0.1	10
0.25	0.5	0.25	10
1	1	1	10

We could see there that even for some impossible initial values which sum is not equal to one, we will have a convergence to the same value.