Applied Statistical Programming - The EM Algorithm

Evan Jo, Annie Jarman, Jordon Newton, Cecilia Sui

4/13/2022

Write R and Rcpp code to answer the following questions. Write the code, and then show what the computer returns when that code is run. Thoroughly comment your solutions.

Complete this assignment before 10:00am on Wednesday, April 20. Submit the R implementation as an Rmarkdown and the knitted PDF to Canvas. Have one group member submit the activity with all group members listed at the top. The Rcpp portion will be given to you as your final assignment.

In-class Background: The Expectation-Maximization Algorithm

The goal of this in-class exercise is to implement an ensemble of models. You will combine forecasts of US presidential elections using ensemble Bayesian model averaging (EBMA). To do this, you must decide how to weight each component of the forecast in the prediction. The collection of these weighted forecasts form the ensemble, and you will use something called the EM (expectation-maximization) algorithm.

The task is to choose values w_k that maximize the following equation:

$$p(y|f_1^{s|t^\star}, ..., f_K^{s|t^\star}) = \sum_{k=1}^{N} w_k N(f_k^{t^\star}, \sigma^2) \tag{1}$$

For the remainder of this assignment, assume that the parameter σ^2 is known and that $\sigma^2 = 1$.

The first step of the EM algorithm is to estimate the latent quantity $\hat{z}_k^t$ that represents the probability that observation t was best predicted by model k.

$$\hat{z}_k^{(j+1)t} = \frac{\hat{w}_k^{(j)} N(y^t|f_k^t, 1)}{\sum_{k=1}^{N} \hat{w}_k^{(j)} N(y^t|f_k^t, 1)} \tag{2}$$

In this equation, j is the particular iteration of the EM algorithm, and $N(y^t|f_k^t, 1)$ is the normal cumulative distribution function evaluated at the observed election outcome (`dnorm(y, ftk, 1)`).

The second step of the EM algorithm is to estimate the expected value of the weights assuming that all $\hat{z}_k^t$ are correct.

$$\hat{w}_k^{(j+1)} = \frac{1}{n} \sum_t \hat{z}_k^{(j+1)t} \tag{3}$$

The estimation procedure is as follows:

1. Start with the assumption that all models are weighted equally.
2. Calculate $\hat{z}_k^{(j+1)t}$ for each model for each election.
3. Calculate $\hat{w}_k^{(j+1)}$ for each model.
4. Repeat steps 2-3 twenty times.

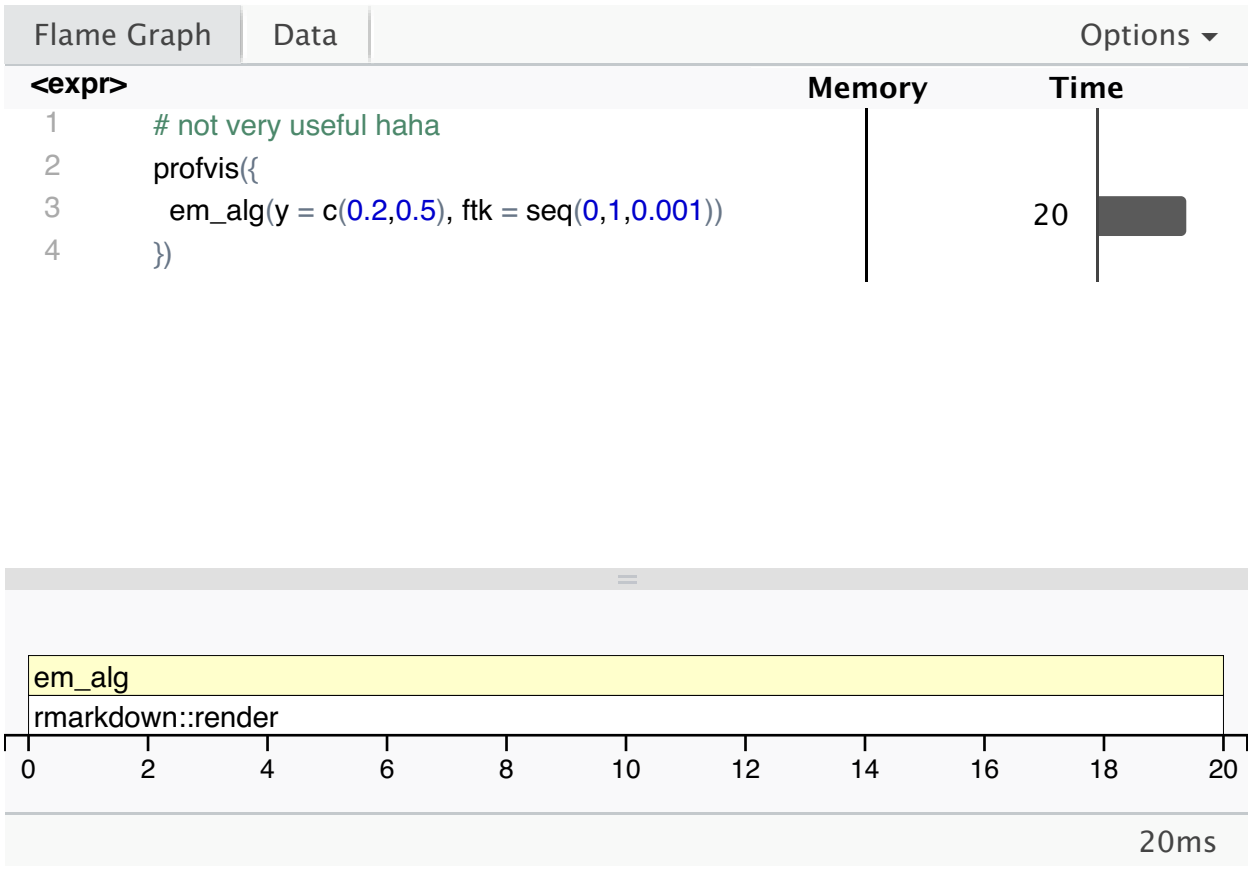
Complete the preceding tasks in `R` alone.

```
em_alg <- function(y, ftk) {

  # assume equal weights
  n <- length(y)
  m <- length(ftk)
  weights <- rep(1 / m, m)

  for (k in 1:20) {
    z_hat <- matrix(nrow = n, ncol = m)

    for (i in 1:n) {
      # store all values
      num_vec <- rep(0,m)

      for (j in 1:m) {
        num_vec[j] <- dnorm(y[i],ftk[j],1) * weights[j]
      }

      # update z_hat vector
      z_hat[i,] <- num_vec / sum(num_vec)
    }

    weights <- colMeans(z_hat)
  }
  return(weights)
}

em_alg(y = c(0.2,0.5), ftk = seq(0,1,0.1))
```

```
##  [1] 0.055922315 0.100265733 0.147846494 0.179292825 0.178817314 0.146673454
##  [7] 0.098943760 0.054893257 0.025046161 0.009398357 0.002900330
```

profiling function

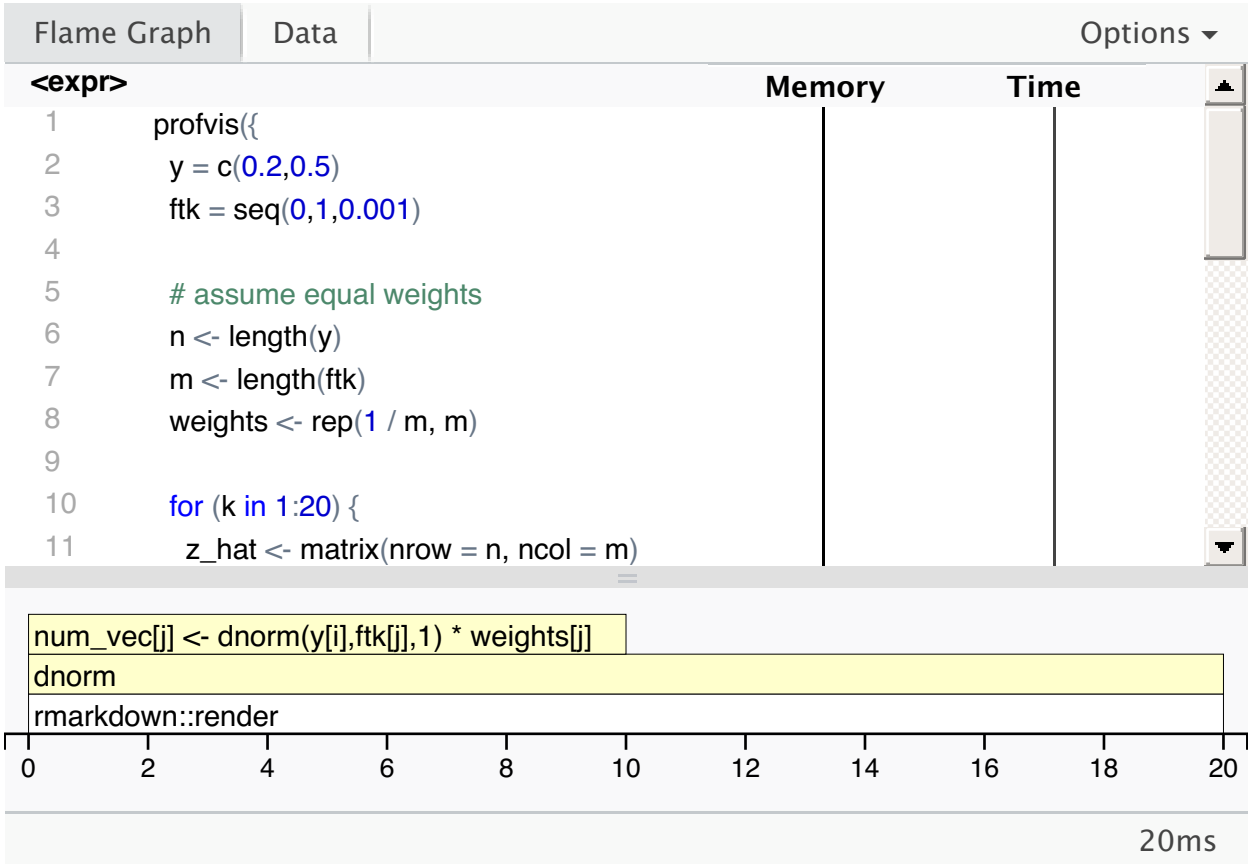
```
# not very useful haha
profvis({
  em_alg(y = c(0.2,0.5), ftk = seq(0,1,0.001))
})
```



profiling

```
profvis({
  y = c(0.2,0.5)
  ftk = seq(0,1,0.001)

  # assume equal weights
  n <- length(y)
  m <- length(ftk)
  weights <- rep(1 / m, m)

  for (k in 1:20) {
    z_hat <- matrix(nrow = n, ncol = m)

    for (i in 1:n) {
      # store all values
      num_vec <- rep(0,m)

      for (j in 1:m) {
        num_vec[j] <- dnorm(y[i],ftk[j],1) * weights[j]
      }

      # update z_hat vector
      z_hat[i,] <- num_vec / sum(num_vec)
```

```
    }

    weights <- colMeans(z_hat)
  }
})
```



Assignment: Rcpp Practice

1. Write an Rcpp function that will calculate the answer to Equation (2). The output will be a matrix.
2. Write an Rcpp function that will calculate the answer to Equation (3). The output will be a vector.
3. Write an Rcpp function that will complete the entire algorithm.