Abstract

We survey the Monte Carlo EM algorithm, focusing on its implementation and a few alternative methods.

- Set standard terminology for the observed information matrix of the observed data likelihood. I think it's best to just call this the observed data information.
- When reviewing, check that every term and symbol has been defined before it is used.

1 Introduction

The EM algorithm (Dempster et al., 1977) is a very influential method for the analysis of missing data. In fact, the original paper detailing this method has ranked among the most cited papers both within the statistics literature (Ryan and Woodall, 2005), and among science as a whole (Van Noorden et al., 2014). The EM algorithm is an iterative algorithm which can be used when the observed data is framed as a partial observation from some unobserved 'complete' dataset. Unfortunately, there are certain technical barriers which can make the EM algorithm intractable in practice. One such challenge is how to compute conditional expectations given the observed data. A popular way to address this challenge is by replacing conditional expectations with Monte Carlo averages. We refer to such a method which makes this substitution as a Monte Carlo EM (MCEM) algorithm.

The practical application of the MCEM algorithm is plagued by numerous implementation details, including when to terminate and what Monte Carlo sample size to use. It is also necessary to devise a way to simulate from the appropriate conditional distribution. Numerous authors have proposed solutions to the termination and sample size problems, although there are is limited guidance for choosing between these solutions. There is also limited advice on choosing a simulation protocol.

In this work, we review various methods which have been proposed for implementing the MCEM algorithm. We also present some closely related alternatives to the MCEM algorithm and summarize what comparisons have been made between the various MCEM versions and alternatives. Next, we broadly review different strategies for simulating from arbitrary distributions. Finally, we give some conclusions and directions for future research.

The MCEM algorithm starts with the paper by Wei and Tanner (1990), which introduces the method and suggests some simple convergence diagnostics. Follow-up work by Chan and Ledolter (1995) uses a pilot study to quantify Monte Carlo uncertainty and inform the design of a follow-up analysis. Related work by Booth and Hobert (1999) and Caffo et al. (2005) focus on uncertainty quantification for the MCEM algorithm as an approximation to the EM algorithm at each step.

Some alternative methods also exist which solve the same problem as the MCEM algorithm but do not fit as neatly within the EM algorithm framework. One such method is

stochastic approximation (Gu and Li, 1998; Robbins and Monro, 1951), which, much like MCEM, is an iterative algorithm which requires that a Monte Carlo sample be generated at each iteration, but uses a different update formula from the EM algorithm. Another alternative is Monte Carlo maximum likelihood (Geyer, 1991), which uses Monte Carlo averaging to approximate the entire likelihood function of the observed data and estimates the parameter of interest by maximizing this approximate likelihood.

Numerous comparisons have been made between the methods introduced above. Some examples are comparisons made in the context of proposing a new method (Citation Needed), while others are full papers dedicated solely to comparing various methods (McCulloch, 1997; Citation Needed). We review these comparisons and give some guidance on how to synthesize the sometimes conflicting findings.

While the idea of the MCEM algorithm sounds promising—replace intractable conditional expectations with Monte Carlo averages—generating the required Monte Carlo samples can also be challenging. Fortunately, numerous methods exist for simulating from difficult distributions, such as importance sampling (Robert and Casella, 2004) and Markov chain Monte Carlo (Gelman et al., 2013). We focus our discussion of simulation strategies on these two methods, but also briefly touch on rejection sampling (Chopin and Papaspiliopoulos, 2020) and sequential Monte Carlo (Del Moral et al., 2006).

We illustrate all the the MCEM algorithms and related methods on a running example. Computation for these analyses is done in `Julia` (Bezanson et al., 2017). Our code is available in a GitHub repository (Ruth and Lockhart, 2023).

The rest of this paper is organized as follows. Section 2 contains a brief overview of the EM algorithm and some of its properties which are most relevant to the MCEM algorithm. Section 3 reviews different implementations of the MCEM algorithm. Section 4 presents alternatives to MCEM, and Section 5 discusses comparisons between the MCEM methods and their alternatives. Finally, Section 6 details some simulation strategies, and Section 7 gives our conclusions.

2 The EM Algorithm

We begin by setting-up the missing data framework and defining three distributions which will be central to our study of missing data problems. Let Y be the observed data and X be the missing data. Note that X need not correspond to any actual real-world process, but may instead be a conceptual device which facilitates analysis of the data that were actually observed. We refer to the distribution of Y as the "observed data distribution", and write f for its density (or mass function). We refer to the joint distribution of Y and X as the "complete data distribution", and write f_c for its density. We refer to the conditional distribution of X given Y as the "missing data distribution", and write f_m for its density. Note that the missing data distribution is not the marginal distribution of the missing data, but rather its conditional distribution given the observed data. We also write ℓ, ℓ_c and

ℓ_m for the log-likelihoods based on the observed, complete and missing data distributions respectively. Similarly, we use S, S_c, S_m for the scores (gradients of the corresponding ℓ's) and I, I_c, I_m for the observed information matrices (negative Hessians of the corresponding ℓ's). We emphasize that, in our notation, a subscript c denotes "complete" rather than "conditional", and a subscript m denotes "missing" rather than "marginal". We write $\theta \in \Theta \subseteq \mathbb{R}^p$ for the parameter of interest, and note that f, f_c and f_m are parameterized by the same θ (although in principle they need not all depend on every component of θ)

The EM algorithm is a method for analyzing incomplete data which was formalized by Dempster et al. (1977). See McLachlan and Krishnan (2008) for an excellent book-length overview of the EM algorithm. We first describe the EM algorithm, then give some of its properties. We also illustrate EM by analysing a toy problem of inferring genotype frequencies based on observed blood group phenotypes.

The EM algorithm consists of iterating two steps. First is the expectation, or "E", step, in which an objective function is constructed from the complete data likelihood. Second is the maximization, or "M", step, in which the previously computed objective function is maximized. These two steps are then alternated until some convergence criterion is met. Whatever value of θ the algorithm converges to is used as our parameter estimate. We now go into more detail on each of the two steps.

The E-step of the EM algorithm is where we construct the objective function which will be used to update our parameter estimate. This objective function is the conditional expectation of the complete data likelihood, given the observed data. If our complete data can be partitioned into an observed component, Y, and a missing component, X, then our objective function at iteration k is given by

$$Q(\theta|\theta_{k-1}) = \mathbb{E}_{\theta_{k-1}}[\ell_c(\theta; y, X)|Y = y] \tag{1}$$

where ℓ_c is the log-likelihood of the complete data model. Note that the conditional expectation uses our parameter estimate from the previous iteration.

The M-step of the EM algorithm consists of maximizing the objective function constructed in the previous E-step. That is, we define $\theta_k = \underset{\theta}{\operatorname{argmax}}\, Q(\theta|\theta_{k-1})$. Typically, this optimization must be performed numerically via, e.g., gradient ascent or the Newton-Raphson algorithm. See Nocedal and Wright (2006) for details and other optimization algorithms.

We can combine the E- and M-steps of the EM algorithm into a single "update function". We write $M(\theta_{k-1}) = \underset{\theta}{\operatorname{argmax}}\, Q(\theta|\theta_{k-1})$. The EM algorithm can thus be viewed as the iterative application of this update function, M. Note that EM performs local, but not necessarily global, optimization. Furthermore, it is common in the types of problems to which the EM algorithm is applied for the (observed data) likelihood surface to be multimodal (McLachlan and Krishnan, 2008). Thus, our choice of starting point can be very important.

2.1 Properties

In this section, we discuss some of the main properties of the EM algorithm. The EM algorithm literature is vast, so we present only a few of the highlights which will be most important to us later.

2.1.1 Ascent Property and Generalized EM

An important feature of the EM algorithm is its so-called "ascent property". This property says that an iteration of the EM algorithm never results in a decrease in the observed data likelihood. This is somewhat surprising to those unfamiliar with the EM algorithm, since updates are computed without ever evaluating the observed data likelihood.

Proposition 2.1 (Ascent Property of EM)**.** *Let $\theta \in \Theta$, and $\theta' = M(\theta)$ be the EM update from θ. Then $\ell(\theta') \geq \ell(\theta)$.*

Proof. We begin by noting that the following decomposition holds for any value of x:

$$\ell(\theta; y) = \ell_c(\theta; y, x) - \ell_m(\theta; y, x) \tag{2}$$

Subtracting the values of both sides at θ from their values at θ' and taking conditional expectations, we get

$$\begin{aligned} \ell(\theta'; y) - \ell(\theta; y) &= Q(\theta'|\theta) - Q(\theta|\theta) + \mathbb{E}_\theta[\ell_m(\theta; y, x) - \ell_m(\theta'; y, x)] && (3) \\ &= Q(\theta'|\theta) - Q(\theta|\theta) + \mathrm{KL}(\theta \to \theta') && (4) \end{aligned}$$

where the last term in line 4 is the Kullback-Leibler (KL) divergence from the missing data distribution with $\theta = \theta$ to the same distribution with $\theta = \theta'$ (van der Vaart, 1998). Note that KL divergences are always non-negative, so we get

$$\ell(\theta'; y) - \ell(\theta; y) \geq Q(\theta'|\theta) - Q(\theta|\theta) \tag{5}$$

Finally, since θ' maximizes $Q(\cdot|\theta)$, we have $\ell(\theta'; y) - \ell(\theta; y) \geq 0$. □

Note that, as long as our model is identifiable, the inequality in line 5 is strict when $\theta \neq \theta'$. Additionally, in our proof of Proposition 2.1, we only required that $Q(\theta'|\theta) \geq Q(\theta|\theta)$, not that θ' maximize $Q(\cdot|\theta)$. This observation leads to the definition of the "Generalized EM algorithm", which replaces the M-step with setting θ_k to any point in Θ such that $Q(\theta_k|\theta_{k-1}) \geq Q(\theta_{k-1}|\theta_{k-1})$.

2.1.2 Recovering Observed Data Likelihood Quantities

Under regularity conditions (see McLachlan and Krishnan, 2008), it is possible to compute both the score vector and the observed information matrix of the observed data likelihood using complete data quantities. These regularity conditions consist of being able to interchange the order of differentiation and integration for various functions.

Proposition 2.2. *Provided that differentiation and integration can be exchanged and that all given expectations are finite, the following identities hold:*

(i) $S(\theta; y) = \mathbb{E}_\theta[S_c(\theta; y, X)|Y = y]$

(ii) $I(\theta) = \mathcal{I}_c(\theta) - \mathcal{I}_m(\theta)$
where $\mathcal{I}_c(\theta) := -\mathbb{E}_\theta\left[\nabla^2 \ell_c(\theta; y, X)|Y = y\right]$ *and* $\mathcal{I}_m(\theta) := -\mathbb{E}_\theta\left[\nabla^2 \ell_m(\theta; y, X)|Y = y\right]$

Proof. We start with expression (i). Let Ω be the complete data sample space. Let $\mathcal{Y}$ and $\mathcal{X}$ be the observed and missing data sample spaces respectively. For every $y \in \mathcal{Y}$, let $\mathcal{X}(y) = \{x \in \mathcal{X} : (y, x) \in \Omega\}$. Note that $f(y; \theta) = \int_{\mathcal{X}(y)} f_c(y, x; \theta) dx$.

$$\begin{aligned}
\mathbb{E}_\theta[S_c(\theta; y, X)|Y = y] &= \int_{\mathcal{X}(y)} \nabla \ell_c(\theta; y, x) f_m(y, x; \theta) dx \\
&= \int_{\mathcal{X}(y)} \frac{f_m(y, x; \theta)}{f_c(y, x; \theta)} \nabla f_c(\theta; y, x) dx \\
&= \int_{\mathcal{X}(y)} \frac{1}{f(y; \theta)} \nabla f_c(\theta; y, x) dx \\
&= \frac{1}{f(y; \theta)} \int_{\mathcal{X}(y)} \nabla f_c(\theta; y, x) dx \\
&= \frac{1}{f(y; \theta)} \nabla \int_{\mathcal{X}(y)} f_c(\theta; y, x) dx \\
&= \frac{1}{f(y; \theta)} \nabla f(y; \theta) \\
&= S(\theta; y)
\end{aligned}$$

Proceeding now to (ii), we decompose the observed data log-likelihood as

$$\ell(\theta; y) = \ell_c(\theta; y, x) - \ell_m(\theta; y, x)$$

Differentiating twice and taking conditional expectations of both sides yields the required result. □

Note that the matrices $\mathcal{I}_c$ and $\mathcal{I}_m$ are not observed information matrices (negative Hessians), but conditional expectations of observed information matrices. An alternative to Proposition 2.2 part (ii), which involves only conditional expectations of complete data quantities, is given in the following proposition.

Proposition 2.3 (Louis' Identity)**.** *Let* $\hat{\theta}$ *be a critical point of the observed data log-likelihood. Assuming that differentiation and integration can be exchanged and that all given expectations are finite, we can write the observed information of the observed data distribution at* θ *as*

$$I(\theta) = \mathcal{I}_c(\theta) - \mathbb{E}_\theta[S_c(\theta) S_c(\theta)^T|Y = y] + S(\theta) S(\theta)^T \tag{6}$$

In particular, if $\hat{\theta}$ is a critical point of the observed data log-likelihood, then

$$I(\hat{\theta}) = \left(\mathcal{I}_c(\theta) - \mathbb{E}_\theta[S_c(\theta)S_c(\theta)^T|Y=y]\right)\Big|_{\theta=\hat{\theta}} \tag{7}$$

Proof. We follow the derivation of Louis (1982). For brevity, we write $f(\theta)$ and $f_c(\theta)$ for $f(y;\theta)$ and $f(y,x;\theta)$ respectively. Consider the following two Hessians:

$$\begin{aligned}
\nabla^2\ell(\theta) &= \nabla\left[\int_{\mathcal{X}(y)} \frac{\nabla f_c(\theta)dx}{f(\theta)}\right] && (8)\\
&= \int_{\mathcal{X}(y)} \frac{\nabla^2 f_c(\theta)}{f(\theta)}dx - \frac{1}{f(\theta)^2}\left(\int_{\mathcal{X}(y)} \nabla f_c(\theta)dx\right)\left(\int_{\mathcal{X}(y)} \nabla f_c(\theta)dx\right)^T && (9)\\
&= \mathbb{E}_\theta\left[\frac{\nabla^2 f_c(\theta)}{f_c(\theta)}\middle| Y=y\right] - \mathbb{E}_\theta\left[\frac{\nabla f_c(\theta)}{f_c(\theta)}\middle| Y=y\right]\mathbb{E}_\theta\left[\frac{\nabla f_c(\theta)}{f_c(\theta)}\middle| Y=y\right]^T && (10)\\
&= \mathbb{E}_\theta\left[\frac{\nabla^2 f_c(\theta)}{f_c(\theta)}\middle| Y=y\right] - S(\theta;y)S(\theta;y)^T && (11)\\
\nabla^2\ell_c(\theta) &= \nabla\left(\frac{\nabla f_c(\theta)}{f_c(\theta)}\right) && (12)\\
&= \frac{\nabla^2 f_c(\theta)}{f_c(\theta)} - S_c(\theta)S_c(\theta)^T && (13)
\end{aligned}$$

Combining lines 11 and 13, we get

$$\nabla^2\ell(\theta) = \mathbb{E}_\theta[\nabla^2\ell_c(\theta)|Y=y] + \mathbb{E}_\theta[S_c(\theta)S_c(\theta)^T|Y=y] - S(\theta;y)S(\theta;y)^T \tag{14}$$

Finally, evaluating line 14 at $\theta=\hat{\theta}$ makes the rightmost term vanish, thereby yielding the required expression. □

Proposition 2.3 is known as Louis' standard error formula. Other decompositions for the observed information matrix of the observed data likelihood do exist; see, e.g., Oakes (1999); McLachlan and Krishnan (2008). However, the one due to Louis will be most useful to us later.

2.2 Example: Gene Frequency Estimation

In this section, we describe an analysis which will be used as a running example on which to illustrate the methods we describe.

Consider the problem of estimating allele frequencies based on observed phenotypes. Often, a single phenotype can be encoded by multiple genotypes with different configurations of dominant and recessive alleles. This is sometimes referred to as the problem of

Table 1: Observed frequency and theoretical probability of each blood type (Fujita et al., 1978)

Blood Type	O	A	B	AB
Random Variable	Y_1	Y_2	Y_3	Y_4
Observed Frequency	10	16	7	1
Probability	r^2	$p^2 + 2pr$	$q^2 + 2qr$	$2pq$

gene frequency estimation. Our analysis closely follows Example 2.4 from McLachlan and Krishnan (2008)[1]

We investigate a simplified model for blood type which consists of only the ABO blood group. See Chapter 5 of Dean (2005), for a detailed introduction to blood types. There are three alleles for this gene: A, B and O. Allele O is recessive, while alleles A and B exhibit co-dominance. That is, genotypes AO and AA encode blood type A, genotypes BO and BB encode blood type B, genotype OO encodes blood type O, and genotype AB encodes blood type AB. Suppose that we seek to estimate the proportion of each allele within a population, based on a sample of individuals' phenotypes. Fujita et al. (1978) report blood types of 4,464,349 people in Japan collected between 1964 and 1975. This sample is so large that any standard errors are practically zero. To retain a reasonable level of uncertainty for the purposes of illustration, we focus on a single administrative division, Oto, in Nara Prefecture. See Figure 1 for details.

Let Y_1, Y_2, Y_3 and Y_4 be the number of people with blood type O, A, B and AB respectively, and $Y = (Y_1, Y_2, Y_3, Y_4)$. Let r, p and q be the proportions of alleles O, A and B respectively within the population of interest. Since $r + p + q = 1$, we let $\theta = (p, q)$ be our target of inference. Pretending that the population size is fixed and that independent, identically distributed (iid) sampling was employed, Y follows a multinomial distribution with $n = 34$. Assuming homogeneous genetic mixing, the probability vector for Y is $\pi = (r^2, p^2 + 2pr, q^2 + 2qr, 2pq)$.

Maximizing the likelihood in this model involves solving the score equation, a system of two 3rd-degree polynomials in p and q. This can be done numerically, and gives estimates $p = 0.299$ and $q = 0.128$. These values match the ones given by Fujita et al. (1978). The information matrix and asymptotic covariance (inverse information matrix) are given by

$$I(\hat{\theta}) = \begin{bmatrix} 276 & 84.8 \\ 84.8 & 584 \end{bmatrix} \tag{15}$$

$$\hat{\Sigma}_{\text{MLE}} = \begin{bmatrix} 3.79 \cdot 10^{-3} & -5.49 \cdot 10^{-4} \\ -5.49 \cdot 10^{-4} & 1.79 \cdot 10^{-3} \end{bmatrix} \tag{16}$$

[1] Although these authors do give data in their example, there is little context around this data, and we have been unable to locate more information from their references. We have opted instead to use a different dataset.

Table 2: Terminology and probabilities for our augmented version of the dataset in Fujita et al. (1978). We also give the blood type coded for be each genotype.

Genotype	OO	AO	AA	BO	BB	AB
Random Variable	X_1	X_2	X_3	X_4	X_5	X_6
Probability	r^2	$2pr$	p^2	$2qr$	q^2	$2pq$
Blood Type	O	A	A	B	B	AB

The asymptotic standard errors for our estimators are thus 0.062 and 0.042 for $\hat{p}$ and $\hat{q}$ respectively.

2.2.1 Complete Data

The problem of gene frequency estimation would be much simpler if we could observe individuals' genotypes. We consider augmenting the observed data Y by further classifying individuals by genotype. Let $X = (X_1, \ldots, X_6)$ be the genotypes of the individuals represented in Table 1. See Table 2.

Note that we can write Y in terms of X. Specifically, $Y_1 = X_1$, $Y_2 = X_2 + X_3$, $Y_3 = X_4 + X_5$ and $Y_4 = X_6$. This corresponds to summing components of X which correspond to the same blood type. The distribution of X is multinomial, with the same sample size as Y, and probability vector given in Table 2.

See Appendix A.2 for the complete data likelihood function and its derivatives.

2.2.2 EM Algorithm

The gene frequency estimation problem fits nicely into the EM algorithm framework. In this section, we present key quantities and results of our analysis. See Appendix A, especially parts A.3 and A.4, for more details.

The EM objective function at iteration k is

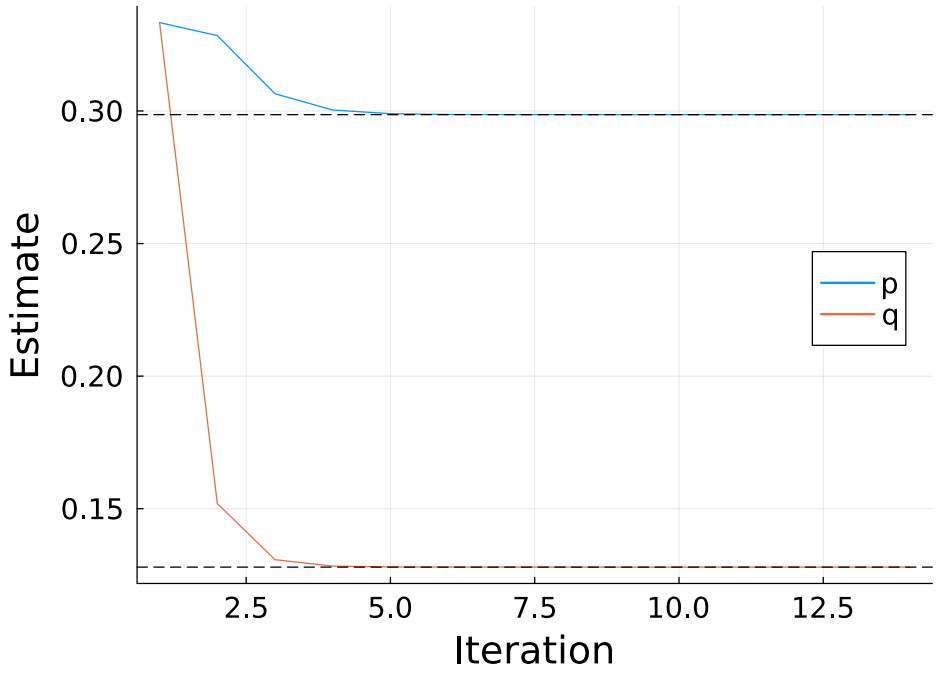
$$Q(\theta|\theta_{k-1}) \equiv \mathbb{E}_{\theta_{k-1}}(n_O|y)\log r + \mathbb{E}_{\theta_{k-1}}(n_A|y)\log p + \mathbb{E}_{\theta_{k-1}}(n_B|y)\log q \tag{17}$$

$$=: \nu_O^{(k-1)}\log r + \nu_A^{(k-1)}\log p + \nu_B^{(k-1)}\log q \tag{18}$$

where $\nu_O^{(k-1)}$, $\nu_A^{(k-1)}$ and $\nu_B^{(k-1)}$ are the expected number of O, A and B alleles respectively given $Y = y$ and $\theta = \theta_{k-1}$. See Appendix A.3 for explicit formulas. Maximizing this objective function corresponds to solving the system of equations given in Equations 74 and 75 of Appendix A.4.

Starting with $\theta_0 = (1/3, 1/3)$ corresponding to equal proportions of the three alleles, Figure 1 gives trajectories of the EM estimates for p and q using the data in Figure 1. These estimates converge quite quickly to the maximizer of the observed data likelihood, given by the horizontal dashed lines. Beyond computing the observed data MLE, we also need the

Figure 1: Trajectory of EM estimates for p and q for the blood type example. Horizontal dashed lines give the values of the MLE.



standard error of this estimator. To this end, we compute the observed data information matrix using Louis' Method (see Proposition 2.3). The asymptotic covariance matrix of our MLE is then the inverse of this information matrix. Omitting details (see Appendices A.2 and A.3), both the observed data information matrix and asymptotic covariance match those obtained from the observed data likelihood.

3 The Monte Carlo EM Algorithm

The Monte Carlo EM, or MCEM, algorithm was first proposed by Wei and Tanner (1990). See also Section 4.5 of Tanner (1996) for a textbook-level treatment of the basic idea with multiple examples. This method proceeds by replacing the conditional expectation in the E-step of the EM algorithm with a Monte Carlo average. More precisely, at each iteration we generate observations from the missing data distribution (i.e. the conditional distribution of the missing data given the observed data), and average the complete data likelihood over this Monte Carlo sample. Formally, at a given iteration of the MCEM algorithm, let $X_1, \ldots, X_M$ be a Monte Carlo sample (not necessarily iid) from the law of $X|Y = y$ with θ set to the value from the previous iteration, say θ_0. Write

$$\hat{Q}(\theta|\theta_0) = \sum_{i=1}^{M} w_i \ell_c(\theta; y, X_i) \tag{19}$$

where the w_i are sampling weights. We write $\hat{\mathbb{E}}\phi$ for the weighted average of a function ϕ based on an importance sample, so $\hat{Q}(\theta|\theta_0)$ can be re-written as $\hat{\mathbb{E}}\ell_c(\theta; y, X)$. Under iid sampling we simply get $w_i = M^{-1}$ for every i, but more intricate sampling schemes may have more complicated weights. In this section, we focus only on iid sampling. See Section 6 for discussions of some alternative sampling methods. The estimate of θ at iteration k is then the maximizer of the MCEM objective function: $\hat{\theta} = \text{argmax}_\theta \hat{Q}(\theta|\theta_{k-1})$. Write $\hat{\theta}_k$ for the kth MCEM estimate.

Provided that the MCEM algorithm converges to a critical point of the observed data likelihood, we can use Proposition 2.3 to estimate the observed data information matrix. Specifically, after declaring convergence, we generate a new Monte Carlo sample and use it to approximate the conditional expectations in Equation (7).

The MCEM algorithm has the advantage of circumventing the challenge of computing potentially intractable conditional expectations for the EM algorithm. However, this analytical simplification does come at the cost of introducing some new computational difficulties. In this section, we outline the main problems faced by the MCEM algorithm and present various solutions which have been proposed in the literature. We focus primarily on practical aspects of the MCEM algorithm; see Fort and Moulines (2003) for a thorough analysis of the convergence properties of MCEM, and Neath (2013) for a survey of the theory of MCEM.

Two connected problems which have received considerable attention in the literature are how to choose the Monte Carlo sample size at each iteration, and when to terminate the MCEM algorithm. These were identified early by Wei and Tanner (1990), but did not receive systematic treatment until later. We here give a brief overview of different authors' approaches to solving these two problems. The rest of this section goes into more detail on each of the methods. Wei and Tanner (1990) suggest examining a plot of the parameter estimates across iterations, and either terminating or increasing the Monte Carlo size (i.e. Monte Carlo sample size) when the plot appears to stabilize. Chan and Ledolter (1995) use a pilot study to choose the Monte Carlo sample size, and terminate when a confidence interval for the improvement in the observed data log-likelihood between successive iterations contains zero. Booth and Hobert (1999) frame each MCEM iteration as an M-estimation problem targeting the deterministic EM update. They increase the Monte Carlo size if an asymptotic confidence interval for the EM update contains the previous iteration's parameter estimate, and terminate when multiple consecutive iterations' estimates have sufficiently small relative error. Caffo et al. (2005) build confidence bounds for the increment in the EM objective function at each iteration of the MCEM algorithm. They increase the Monte Carlo size until the lower bound is positive and terminate when the upper bound is sufficiently small.

In the rest of this section, we give more detail on each of the implementations introduced above. We also illustrate each method on the blood type dataset described in Section 2.2. The relevant conditional distribution and likelihood calculations are described in Appendix A.

3.1 Early Work (Wei and Tanner, 1990)

In their seminal work, Wei and Tanner (1990) propose the MCEM algorithm and present a simple implementation. They illustrate that the complete data gradient and Hessian are easily obtained at each iteration from the Monte Carlo sample and, following Louis (1982), give an estimator for the observed data information matrix. Regarding convergence, Wei and Tanner recommend plotting the parameter estimates across iterations and stopping when the estimates appear to stabilize around some constant. When this stabilization is detected, one can either declare convergence and stop, or increase the Monte Carlo size and continue iterating until the estimate trajectory again stabilizes.

In order to apply the MCEM algorithm to estimate allele frequencies in the blood type problem, we must specify the number of iterations, K, and the Monte Carlo size for each iteration, M. Starting conservatively, we use $K = 50$ and $M = 100$. Figure 2a gives trajectories of the MCEM estimates of p and q. These estimates appear to converge quickly to a stationary mean, but there is still some uncertainty around this mean. As such, we run MCEM for another 20 iterations with $M = 1000$, staring with the final value from our first run. See Figure 2b. The trajectories from our second run are much more stable around their means. We use the final values from these trajectories as our estimates: $\hat{p} = 0.298$

and $\hat{q} = 0.128$. These values closely match the maximizer of the observed data likelihood.

As can be seen in Section 2.2, the statistical uncertainty in our problem is on the order of 10^{-2}. This is much larger than the Monte Carlo variability seen in either plot of Figure 2.

3.2 Running a Pilot Study (Chan and Ledolter, 1995)

Building on the ideas of Wei and Tanner, Chan and Ledolter (1995) develop a method for both choosing the Monte Carlo size and deciding when to terminate the MCEM algorithm. The method of Chan and Ledolter includes numerous choices for which they do not give specific guidance. We describe the procedure in general terms, but details for any particular implementation will need to be explored in the context of the dataset being analysed.

The algorithm presented by Chan and Ledolter is based on an identity which allows us to estimate observed data likelihood ratios by Monte Carlo averages of complete data likelihood ratios. More precisely, we write

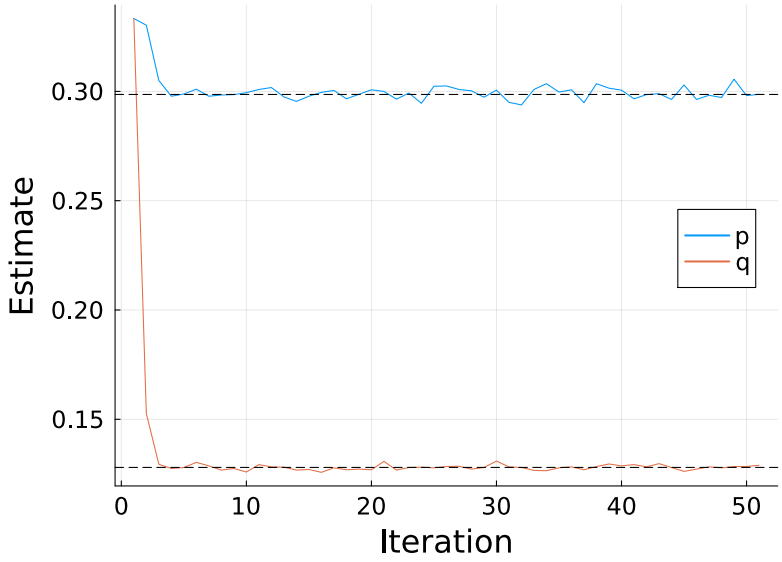
$$\frac{\mathcal{L}(\theta_1; y)}{\mathcal{L}(\theta_2; y)} = \mathbb{E}_{\theta_2}\left[\frac{\mathcal{L}_c(\theta_1; y, X)}{\mathcal{L}_c(\theta_2; y, X)} \middle| Y = y \right] \tag{20}$$

See Chan and Ledolter (1995) for a derivation. To apply Equation 20 to the MCEM algorithm, we replace the conditional expectation on the right-hand side with a Monte Carlo average from the corresponding conditional distribution. This adjustment allows us to estimate log-likelihood ratios from the observed data distribution, without ever directly evaluating the observed data likelihood.

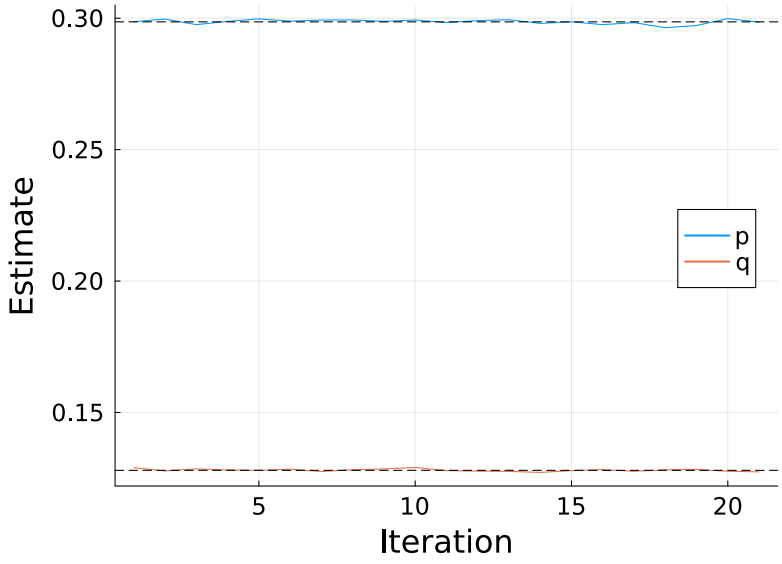
The method of Chan and Ledolter can be divided into two parts. The first part is a pilot study, in which we compute a standard error for our log-likelihood ratio estimator near the MLE, and determine what Monte Carlo size is required to get this standard error below a pre-specified threshold. Note that this standard error quantifies Monte Carlo uncertainty, not sampling variability of the observed data. In the second part, we increase the Monte Carlo size appropriately, and continue iterating until a confidence interval for the true observed data log-likelihood ratio contains zero. This two-part procedure reflects the suggestion of Wei and Tanner (1990) to run MCEM until it appears to stabilize, then increase the Monte Carlo sample size to get a more precise estimate.

The pilot study portion of Chan and Ledolter's method consists of running the MCEM algorithm with a fixed, "moderately large" Monte Carlo size (although they give no general guidance on exactly how large this should be). In addition to tracking the parameter estimates across iterations, we also record the estimated log-likelihood ratio of the current estimate relative to the starting point of the algorithm. This ratio is computed by keeping a running cumulative sum of all one-step log-likelihood ratios. We terminate our pilot study after a pre-specified number of iterations, and identify the estimate with largest estimated observed data log-likelihood ratio.

Figure 2: Trajectory of MCEM estimates of p and q for the blood type example. The horizontal lines correspond to maximum likelihood estimates.



(a) $M = 100$



(b) $M = 1000$

After concluding our pilot study, we select a few estimates from iterations near the maximizer (Chan and Ledolter suggest the 10 which follow the maximizer but give no justification for this number). All the selected estimates are thought of as approximately equivalent to the maximizer. We then perform a few single-iteration MCEM runs from each of the chosen estimates and pool information about the variability of a one-step change in the estimated observed data log-likelihood. See Section 2.3 of Chan and Ledolter (1995) for a more detailed description of their pilot study and variance estimator. Once we have an estimate of the Monte Carlo variance of our log-likelihood ratio estimator, we calculate what Monte Carlo sample size would be required to get that variance below some pre-specified threshold[2]. We then use this newly determined Monte Carlo size for a follow-up MCEM run.

More precisely, using our newly determined Monte Carlo size, we return to the optimal parameter estimate from our pilot study and continue iterating the MCEM algorithm. At each step now, we also construct a confidence interval for the true observed data log-likelihood ratio corresponding to the current parameter update (i.e., between two consecutive parameter values, not from the starting point to the current estimate). We terminate the algorithm when such a confidence interval contains zero. This corresponds to no evidence of an improvement in the observed data likelihood.

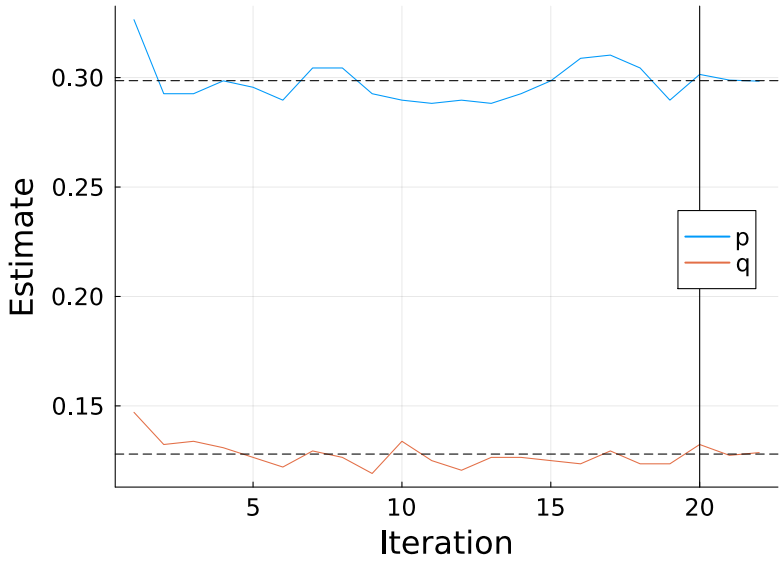
Note that the estimated observed data log-likelihood ratio computed at each iteration is by a Monte Carlo average. In order to avoid bias, we do not recycle the current iteration's Monte Carlo sample to estimate this ratio. Instead, we generate the Monte Carlo sample which will be used in the next iteration, and use this new sample to estimate the log-likelihood ratio[3]. We are then free to use this new Monte Carlo sample to compute the next iteration's parameter update.

Applying Chan and Ledolter's method to our blood type dataset, we get the parameter estimate trajectory shown in Figure 3. Figure 4 gives the trajectory of estimated observed data log-likelihood ratios relative to the starting point of the algorithm (starting with $\hat{p} = \hat{q} = 1/3$), along with pointwise 95% Wald-type confidence bands for the post-pilot study iterations. Recall that these confidence bands are used to assess convergence of the algorithm. We require that the standard error of our estimated log-likelihood increment be at most 10^{-3}, since this is much smaller than the statistical uncertainty given in Section 2.2. Note that this standard error only applies near the maximizer of the observed data MLE (i.e., we shouldn't take the CIs in Figure 4 too seriously until the trajectories in Figure 3 have stabilized). Indeed, we see that the Monte Carlo fluctuations in Figure 3 are much smaller than the statistical fluctuations in our problem. The estimated parameter

[2]Provided that the one-step observed data likelihood ratio is evaluated close to the MLE, Chan and Ledolter show that its Monte Carlo standard error scales like $1/M$ rather than the usual $1/\sqrt{M}$, where M is the Monte Carlo size.

[3]Due to our Monte Carlo sample using the new parameter estimate rather than the old, we must actually estimate the reciprocal of the likelihood ratio that we want, then multiply its logarithm by -1. This is reflected in the formulas of Chan and Ledolter but not discussed explicitly.

Figure 3: MCEM estimates of p and q for the blood type example, based on the method of Chan and Ledolter (1995). The vertical line shows the end of the pilot study. The horizontal dashed lines correspond to maximum likelihood estimates.

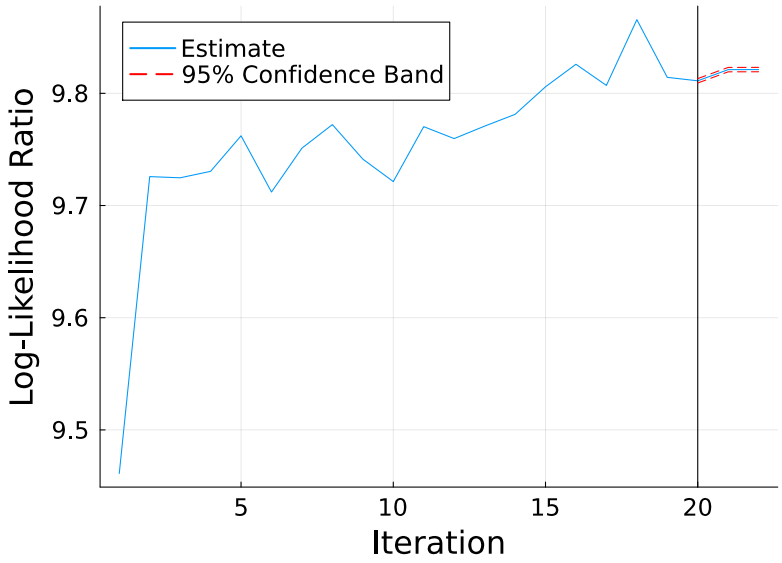


values are $\hat{p} = 0.298$ and $\hat{q} = 0.129$, which closely match the MLE.

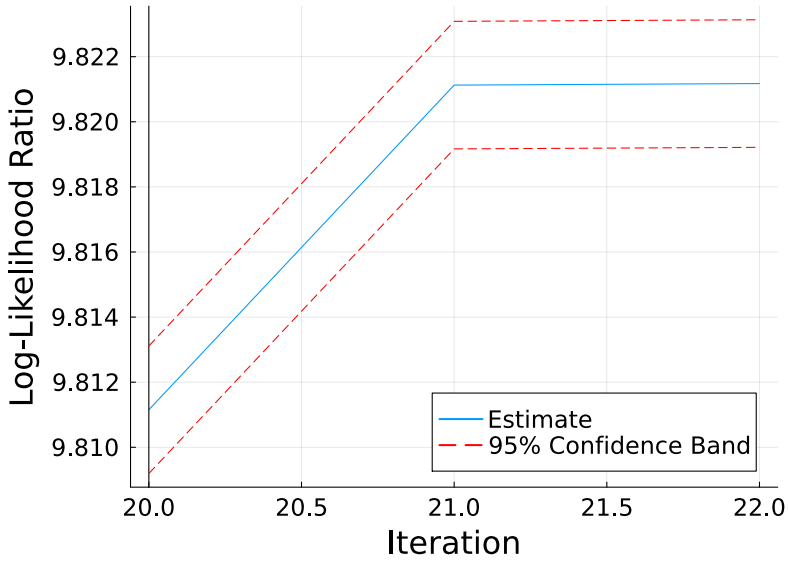
3.3 Uncertainty Quantification for the Parameter Estimate (Booth and Hobert, 1999)

Booth and Hobert (1999) use a somewhat different approach from either Wei and Tanner (1990) or Chan and Ledolter (1995) to understand the behaviour of the MCEM algorithm. The method of Booth and Hobert is based on quantifying Monte Carlo uncertainty of the MCEM update as an approximation to the update which would be made by the deterministic EM algorithm from the same starting point. They recommend starting the MCEM algorithm with a small Monte Carlo size, and adding more observations only when the parameter estimates are no longer changing discernibly across iterations. More formally, Booth and Hobert suggest building a confidence interval for the EM update based on the Monte Carlo variability of the MCEM update at each iteration. If this interval contains the previous iteration's parameter estimate, then the Monte Carlo variability is too large relative to the size of the parameter updates; thus, more samples are required. The authors similarly recommend assessing convergence by checking for small relative error in the parameter updates. To account for the possibility of Monte Carlo variability leading to two consecutive estimates being similar before the algorithm has 'converged', they suggest

Figure 4: Estimated observed data log-likelihood ratio, based on the method of Chan and Ledolter (1995). Red dashed lines give 95% pointwise confidence bands, and the vertical line shows the end of the pilot study.



(a) Full trajectory



(b) Post-pilot study iterations only

waiting until the relative error is small for three consecutive iterations.

The confidence interval used to quantify Monte Carlo uncertainty within an iteration is obtained by framing each step of the MCEM algorithm as the solution of an M-estimation problem. This allows us to inherit the desirable properties of M-estimators; specifically, asymptotic normality (see, e.g. van der Vaart, 1998). Following the usual M-estimator construction and assuming that the relevant regularity conditions hold, we can estimate the asymptotic variance of the MCEM parameter estimator at each iteration. Note that this standard error is based on the Monte Carlo variability within an iteration; it does not measure sampling variability due to the observed data.

More formally, write $\tilde{\theta}_k$ for the EM update based on $\hat{\theta}_{k-1}$. Note that $\hat{\theta}_{k-1}$ is held fixed (analysis of an MCEM update is done conditional on the previous iteration's estimate). Unless stated otherwise, all expectations are taken with $\theta = \hat{\theta}_{k-1}$, so we suppress this in our notation. Assuming sufficient smoothness and moment conditions, we get the following expression for the MCEM update:

$$\sqrt{M_k}(\hat{\theta}_k - \tilde{\theta}_k) = -\sqrt{M_k}\left[\nabla^2 Q(\tilde{\theta}_k|\hat{\theta}_{k-1})\right]^{-1}\left[\nabla\hat{Q}(\tilde{\theta}_k|\hat{\theta}_{k-1})\right] + o_p(1) \tag{21}$$

as $M_k \to \infty$, where M_k is the Monte Carlo size used to compute $\hat{\theta}_k$, ∇ denotes differentiation with respect to the left argument of Q or $\hat{Q}$ and $o_p(1)$ is a sequence which converges in probability to zero. Note that the first expression on the right-hand side is the inverse Hessian of the EM objective function (fixed) while the second is the gradient of the MCEM objective function (an average). Thus, $\hat{\theta}_k$ is asymptotically normal with asymptotic covariance

$$\left[\nabla^2 Q(\tilde{\theta}_k|\hat{\theta}_{k-1})\right]^{-1}\mathbb{V}\left[S_c(\tilde{\theta}_k)|Y=y\right]\left[\nabla^2 Q(\tilde{\theta}_k|\hat{\theta}_{k-1})\right]^{-1} \tag{22}$$

$$\approx \left[\nabla^2\hat{Q}(\hat{\theta}_k|\hat{\theta}_{k-1})\right]^{-1}\hat{\mathbb{E}}\left[S_c(\hat{\theta}_k)S_c(\hat{\theta}_k)^T|Y=y\right]\left[\nabla^2\hat{Q}(\hat{\theta}_k|\hat{\theta}_{k-1})\right]^{-1} \tag{23}$$

where S_c is the complete data score vector, and $\hat{\mathbb{E}}$ is the Monte Carlo average over the missing data with $\hat{\theta}_k$ held fixed. Note that there is no first moment term in the conditional variance of S_c because $\hat{\theta}_k$ is a maximizer of $\hat{\mathbb{E}}[\ell_c(\theta)|Y=y]$.

Based on the above discussion, we can build an asymptotic confidence interval for $\tilde{\theta}_k$. Booth and Hobert recommend checking whether this interval contains $\hat{\theta}_{k-1}$ and, if so, increasing the Monte Carlo size for the next iteration. Specifically, they suggest starting the next iteration with a sample of size $M_{k+1} = M_k(1 + 1/r)$, with $r = 3, 4$ or 5 working well in their examples.

To assess convergence of the MCEM algorithm, Booth and Hobert present two criteria. The first is a familiar measure of relative error in parameter estimates between consecutive iterations:

$$\max_j\left(\frac{\left|\hat{\theta}_{k,j} - \hat{\theta}_{k-1,j}\right|}{\left|\hat{\theta}_{k-1,j}\right| + \delta_1}\right) < \delta_2 \tag{24}$$

where δ_1 and δ_2 are small positive constants, and the subscript j ranges over components of θ. Booth and Hobert suggest using $\delta_1 = 10^{-3}$ and δ_2 between $2 \cdot 10^{-3}$ and $5 \cdot 10^{-3}$. See p. 436 of Searle et al. (2006) and the references therein for a discussion of the form of Equation 24.

Alternatively, since Booth and Hobert apply their method to the analysis of generalized linear mixed models, where pathologies may arise due to parameter estimates being too close to a boundary, they propose a second stopping rule:

$$\max_j \left(\frac{\left| \hat{\theta}_{k,j} - \hat{\theta}_{k-1,j} \right|}{\mathrm{SE}\left(\hat{\theta}_{k,j} \right) + \delta_1'} \right) < \delta_2' \tag{25}$$

where δ_1' and δ_2' are tolerances which may or may not differ from δ_1 and δ_2, and $\mathrm{SE}\left(\hat{\theta}_{k,j} \right)$ is the standard error at iteration k. The purpose of condition (25) is to detect when estimated variance components are very close to zero, whereupon the numerical precision needed to satisfy condition (24) requires a prohibitive amount of computation.

We apply the method of Booth and Hobert to our blood type example, with the settings recommended in their paper. Specifically, they suggest setting $\alpha = 0.25$, $k = 3$ (alternatively, 4 or 5), $\delta_1 = 0.001$, and $\delta_2 = 0.002$ (or as high as 0.005). We also start with a Monte Carlo size of 10. Figure 5 gives trajectories of the MCEM estimates, as well as the Monte Carlo size used to obtain each of these estimates. Note how the trajectories stabilize around the MLE as MC size increases. The final estimate from this method is $\hat{p} = 0.299$, $\hat{q} = 0.128$; again, very close to the MLE. The Monte Carlo fluctuations are also much smaller than the statistical uncertainty given in Equation (16).

3.4 Uncertainty Quantification for the Objective Function (Caffo et al., 2005)

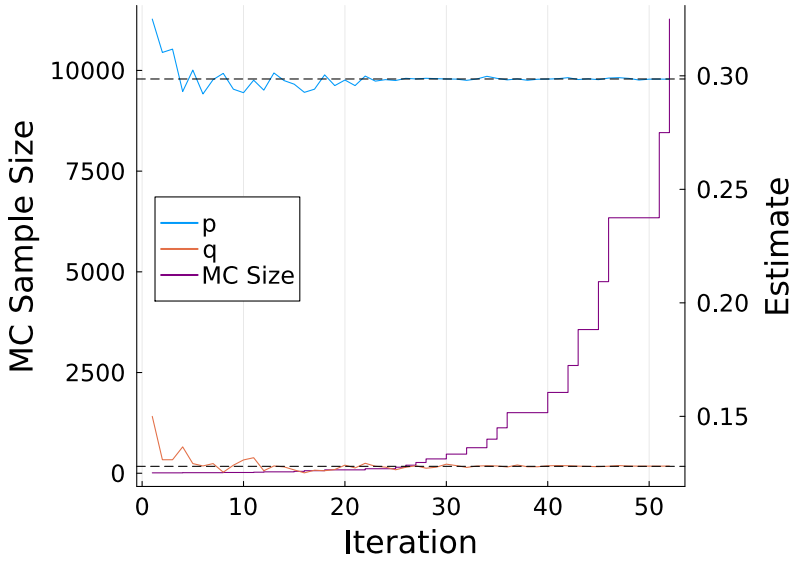
The approach of Caffo et al. (2005) is similar in spirit to that of Booth and Hobert (1999). Both methods quantify Monte Carlo uncertainty in the MCEM algorithm as an approximation to the EM algorithm. The difference is that where Booth and Hobert measure uncertainty in the parameter estimates, Caffo et al. focus on uncertainty in the objective function. Specifically, Caffo et al. base their analysis on asymptotic normality of the MCEM increment using the following:

Proposition 3.1. *Let* $\Delta\hat{Q}(\hat{\theta}_k|\hat{\theta}_{k-1}) = \hat{Q}(\hat{\theta}_{k-1}|\hat{\theta}_{k-1}) - \hat{Q}(\hat{\theta}_k|\hat{\theta}_{k-1})$. *Define* $\Delta Q(\hat{\theta}_k|\hat{\theta}_{k-1})$ *similarly. Let* M_k *be the Monte Carlo size at iteration* k. *Then*

$$\sqrt{M_k} \left[\Delta\hat{Q}(\hat{\theta}_k|\hat{\theta}_{k-1}) - \Delta Q(\hat{\theta}_k|\hat{\theta}_{k-1}) \right] \rightsquigarrow N(0, \Sigma_k) \tag{26}$$

as $M_k \to \infty$, *where* Σ_k *is an asymptotic covariance matrix.*

Figure 5: Trajectory of estimates for p and q, as well as Monte Carlo sample sizes, from the method of Booth and Hobert. Horizontal dashed lines give the maximum likelihood estimates.



See Caffo et al. for hypotheses and a proof sketch. Provided that we are able to estimate Σ_k, Proposition 3.1 allows us to build asymptotic confidence intervals for the EM increment, ΔQ. Recall that in Section 2.1.1, we defined the Generalized EM algorithm by requiring that $\Delta Q \geq 0$, and showed that this requirement guarantees the ascent property. While the stochastic nature of the MCEM algorithm makes it impossible to guarantee that the EM increment is positive, we are able to use Proposition 3.1 to construct asymptotic confidence bounds for ΔQ. Provided that we can estimate Σ_k, we can then be reasonably confident that $\Delta Q > 0$.

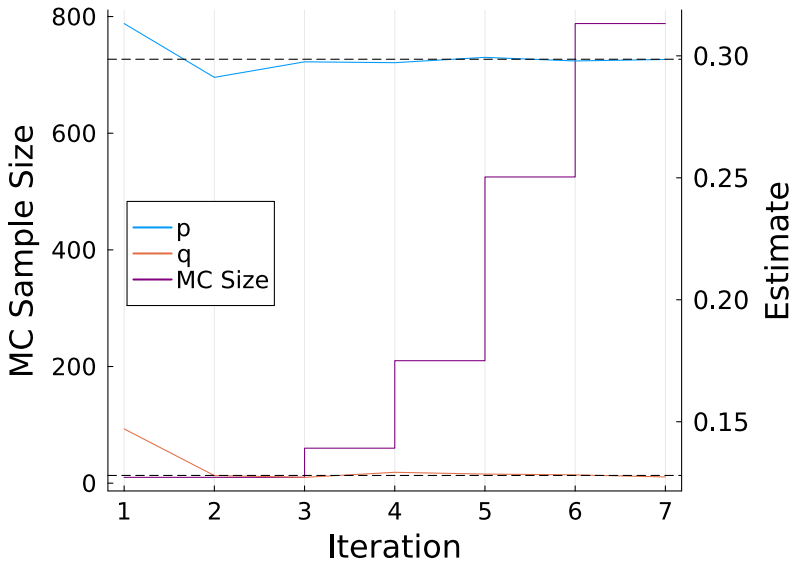
Estimating the asymptotic variance under direct or rejection sampling is fairly straightforward. Importance sampling however, is somewhat more complicated; particularly when a normalizing constant must be estimated (see Section 6.1 for more on importance sampling). Caffo et al. give a Delta Method-based formula for estimating Σ_k under importance sampling (see Chapter 3 of van der Vaart, 1998, for an overview of the Delta Method). They also give some guidance for calculating standard errors based on Markov chain Monte Carlo sampling, which we do not go into here. See Section 6.2 for more details on Markov chain Monte Carlo sampling.

We now return to the key MCEM problems of choosing the Monte Carlo size and when to terminate. For the former, Caffo et al. advise constructing a lower confidence limit for the EM increment, ΔQ. If this confidence limit is positive, then we proceed to the next iteration. If not, then we augment the Monte Carlo sample at the current iteration (with, say, M_k/r new points, for some small positive integer, r, as in Booth and Hobert, 1999), and compute a new confidence bound. At the next iteration, Caffo et al. advise using a starting Monte Carlo sample which is at least as large as the final sample from the previous iteration. In fact, we may find that a larger sample should be used based on extrapolation of Monte Carlo variability from the previous iteration. The paper gives a formula to check for whether we should increase the Monte Carlo size before starting the next iteration based on a normal approximation to increments in the MCEM objective function. In our sample analyses, increasing the Monte Carlo size between iterations is never called-for, so we omit this step from our presentation.

Caffo et al. base their termination criterion on stopping when there is evidence that the algorithm is no longer yielding sufficient improvement in the EM objective function. Specifically, they start by choosing a tolerance, $\tau > 0$, then calculate an upper confidence limit for the EM increment at each iteration. If this upper confidence limit is below τ, then we declare that there is little room for improvement left in the EM objective, and terminate our algorithm.

We now apply the method of Caffo et al. to our blood type example. This method has numerous tuning parameters, and the paper gives limited guidance on how to select them. As such, we choose values which appear to work reasonably well. Specifically, we use confidence levels of 80% when checking to augment the Monte Carlo size and 90% when checking for termination. Every time we augment the Monte Carlo size at iteration k, we add $M_k/2$ more points. We use a tolerance level of 10^{-3} to check for termination, and a

Figure 6: Trajectory of estimates for p and q, as well as Monte Carlo sample sizes, from the method of Caffo et al.. Horizontal dashed lines give the maximum likelihood estimates.



starting Monte Carlo size of 10. Figure 6 gives trajectories of the MCEM estimates, as well as the Monte Carlo size used to obtain each of these estimates. As with the method of Booth and Hobert (1999), the trajectory stabilizes as the Monte Carlo size increases. Our final estimate here is $\hat{p} = 0.299$ and $\hat{q} = 0.127$, which is very close to the MLE. As with our other methods, the Monte Carlo fluctuations here are much smaller than the statistical uncertainty given in Equation (16).

4 Alternatives to the MCEM Algorithm

In this section, we outline some alternatives to the MCEM algorithm for maximizing the likelihood of an incomplete dataset. Examples include stochastic approximation (Robbins and Monro, 1951; Lai, 2003) and the Monte Carlo maximum likelihood method (Gelfand and Carlin, 1993; Geyer, 1994). We are actively working on implementing the methods in this section on the blood type dataset.

4.1 Stochastic Approximation

This section is probably too long.

Stochastic approximation (SA) is a method originally proposed by Robbins and Monro

(1951) for finding roots of functions which can only be evaluated with noise. This method was expanded upon rapidly by, for example, Kiefer and Wolfowitz (1952) into a method for derivative-free optimization, and by Dvoretzky (1956) with a systematic theoretical framework. Since the mid-20th century, SA methods have grown into a thriving research area. See, e.g., Kushner and Yin (1997) or Borkar (2022) for textbook-length treatments, and Lai (2003) for a survey paper which focuses on applications to statistics.

The basic version of SA iteratively updates our estimate of the root, θ_*, of some unobservable function, ϕ, based on the value of a noisy realization of that function at the current estimate. Specifically, if θ_k is our estimate at iteration k and $\hat{\phi} \approx \phi$, then our estimate at iteration $k+1$ is $\theta_{k+1} = \theta_k + \alpha_k \hat{\phi}(\theta_k)$, where α_k is a sequence which goes to zero at a particular rate. Since our sequence of weights goes to zero with k, the update terms become negligible in the limit and our estimate of θ_* stabilizes. The precise requirement for these weights is that $\sum_{k=1}^{\infty} \alpha_k = \infty$ and $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$. A common choice is $\alpha_k = k^{-1}$. Numerous authors have studied convergence of the SA method in probability, almost surely and in $\mathcal{L}^1$ under various regularity conditions. See Lai (2003) for an excellent review of the history of stochastic approximation convergence theory.

Although the specific implementations of stochastic approximation are too numerous to cover here, we do give a particular application which is of great use to us. Suppose that we want to optimize a function, f, which we can only evaluate approximately. Assume further that we are able to approximately evaluate the gradient of f, ∇f. Setting $\phi = \nabla f$ and running the SA algorithm with $\hat{\phi} \approx \nabla f$ gives us a critical point for f. We can therefore use SA to optimize functions which cannot be evaluated exactly. This version of SA sees considerable use in the Machine Learning community under the name Stochastic Gradient Descent (Bottou, 2010). A related method developed by Kiefer and Wolfowitz (1952) involves approximating the gradient at each step with finite differences.

We now discuss specific applications of stochastic approximation to the missing data problem. We refer to such an application as a stochastic approximation EM (SAEM) algorithm. We turn first to the method of Gu and Li (1998), which follows the outline presented above for using stochastic approximation for optimization. That is, Gu and Li suggest setting ϕ to the observed data score, $\phi(\theta) = S(\theta; y)$. This function can be estimated using Proposition 2.2 (i) and the same Monte Carlo approach we use with MCEM. Iteratively applying the stochastic approximation update formula ultimately yields a critical point of the observed data score. In the case of a vector-valued parameter, Gu and Li also recommend pre-multiplying $\hat{\phi}$ by a matrix which converges to the inverse of the observed data information matrix. Such a sequence of matrices can be constructed using Louis' Identity (Proposition 2.3) and Monte Carlo. Note that we use the same sample of missing data to update our estimates of the parameter and the observed data information matrix at each iteration. Similar work by Gu and Kong (1998) extends the same SAEM construction to accommodate Markov chain Monte Carlo sampling. Gu and Zhu (2001) discuss how to apply the above methodology to the analysis of spatial models, and incorporate a second stage to the method in which estimates are averaged across iterations. This averaging

has been found to improve performance of SAEM, and of stochastic approximation more generally (Polyak and Juditsky, 1992; Delyon et al., 1999).

Deylon et al. (1999) present an SAEM implementation for estimation in exponential family models, in which stochastic approximation is used to estimate the EM objective function instead of working directly with θ. Here, the estimate being updated at each iteration is $\tilde{Q}$, an approximation to the EM objective function, Q. The update term is $\hat{phi}_k = \tilde{Q}_k - \hat{Q}_k$, where $\hat{Q}_k$ is the MCEM objective function based on the estimated value of θ from the previous iteration. Our updated parameter estimate is then obtained by maximizing over θ in the new stochastic approximation objective function, $\tilde{Q}_{k+1}(\theta) = \tilde{Q}_k(\theta) + \alpha_k[\tilde{Q}_k(\theta) - \hat{Q}_k(\theta)]$. Note that the updated objective function can be re-written as a convex combination: $\tilde{Q}_{k+1} = (1 - \alpha_k)\tilde{Q}_k + \alpha_k\hat{Q}_k$. Since $\alpha_k \to 0$, each iteration of stochastic approximation progressively puts more weight on the pre-existing objective function and less weight on the MCEM objective.

Keen observers will note that the update formula given above does not fit exactly into the stochastic approximation framework given earlier in this section. Specifically, the formula given by Deylon et al. (1999) does not directly update θ, but instead updates $\tilde{Q}$, which depends indirectly on θ, and is then used to infer an update for θ. In order to re-frame the algorithm of Deylon et al. as a stochastic approximation update, we must use an approximate sufficient statistic as our estimate of θ at each iteration. We also add an asymptotically negligible bias term to our update formula (as a theoretical device). See Section 5 of Deylon et al. (1999) and Chapter 5 of Kushner and Yin (2003) for details.

A subtly different line of research on the SAEM method has been developed by a group at the National Institute for Research in Digital Science and Technology (INRIA) in France (see, e.g., Celeux et al., 1995, for a discussion of some of their methods). The goal here is to augment the EM algorithm, rather than to facilitate the application of EM-type methods when ordinary EM is intractable. More precisely, methods from this group introduce a stochastic perturbation to the EM algorithm, with the goal of escaping fixed points which are locally, but not globally, optimal. Early work centered around a Stochastic EM (SEM) algorithm (Celeux and Diebolt, 1985), which is equivalent to the MCEM algorithm with a Monte Carlo size of one (Celeux and Diebolt, 1987; Celeux et al., 1995; see also Nielsen, 2000). Later, they also propose a method which they refer to as SAEM (although it does not quite fit into our framework), in which each iteration consists of first computing both the EM and MCEM updates from the previous iteration's estimate, then combining these two updates in a convex combination as the estimate for the current iteration. Here, as with the SEM algorithm, the MCEM update is computed with a Monte Carlo size of one (Celeux and Diebolt, 1992; Celeux et al., 1995).

An advantage of the SAEM algorithm over MCEM is that in SAEM we choose the Monte Carlo size once at the beginning and leave it fixed for every iteration. We can think of the method as automatically increasing the MC size since the estimate at each iteration is a weighted sum of all the estimates which came before it. A disadvantage of SAEM is that it requires us to select the sequence $\{\alpha_k\}$, commonly referred to as the "step

size". Choosing α_k too large will mean the algorithm takes a long time to stabilize, while choosing α_k too small causes the algorithm to stabilize before it reaches its limiting value (and will therefore take a long time to reach this limit). Jank (2006) gives some guidance on choosing this step size based on the goal of balancing bias with variance. Jank also presents a convergence diagnostic based on the ideas of Caffo et al. (2005) which allows for a posteriori assessment of whether the step size was too small.

We illustrate the SAEM methods of Gu and Li (1998) and Deylon et al. (1999) on our blood type example. For both methods, we use $\alpha_k = k^{-1}$ (a popular choice), a Monte Carlo sample size of 10 at each iteration, and 50 iterations. Since our parameters are constrained to lie between 0 and 1, we apply the method of Gu and Li on logit-scale, then back-transform before plotting[4]. See Figure 7 for trajectories from both SAEM methods. The final estimate from the Gu and Li method is $\hat{p} = 0.291$ and $\hat{q} = 0.127$, while the Deylon et al. method gives $\hat{p} = 0.301$ and $\hat{q} = 0.128$.

Both SAEM estimates are close to the MLE, but differ more than the MCEM methods presented in Section 3. It is worth noting that the number of Monte Carlo samples used to compute the final estimate under SAEM is comparable to that used by many MCEM methods. For SAEM however, this is the total number of Monte Carlo draws, whereas MCEM requires this many simulations at every iteration. Therefore, the total Monte Carlo cost of SAEM is much lower than for MCEM. Compare total number of MC samples across methods.

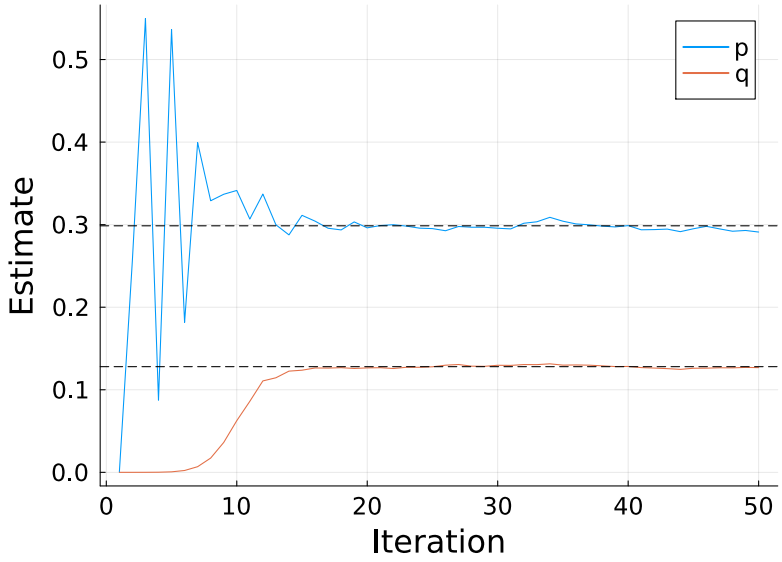
4.2 Monte Carlo Maximum Likelihood

The Monte Carlo maximum likelihood (MCML) method was developed to handle settings where the likelihood function cannot be evaluated exactly. The idea is to approximate the whole likelihood surface (up to an additive constant) using a single Monte Carlo sample. This approximate likelihood is then maximized, either numerically or analytically. The core idea of MCML is to choose a fixed reference parameter value, θ_*, and estimate likelihood ratios relative to θ_*. The likelihood ratio is written as an expectation with respect to the fixed reference value, and this expectation is approximated by Monte Carlo. A key feature of this methodology is that a single Monte Carlo sample can be re-used to estimate the likelihood ratio at any number of target parameter values. Finally, we maximize our estimated likelihood ratio as a proxy of the unknown likelihood function.

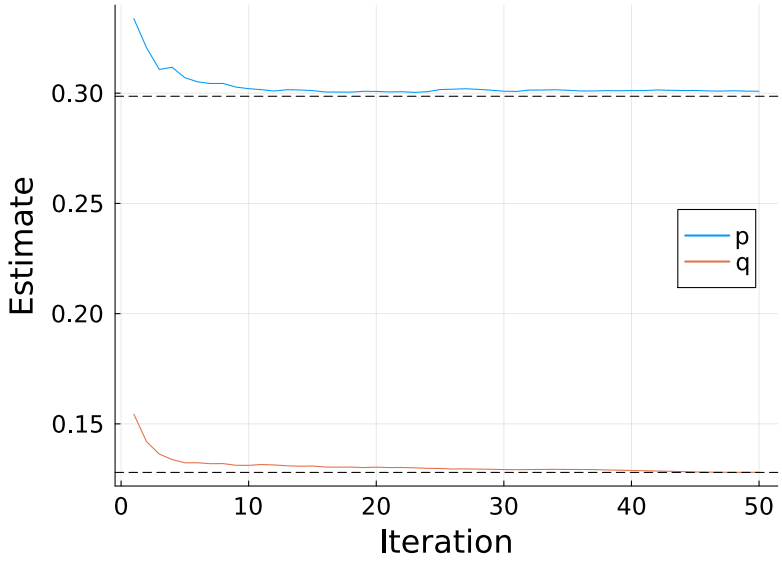
The most basic form of Monte Carlo maximum likelihood (Geyer, 1991) applies when we only know the likelihood up to a normalizing constant, say $f(y;\theta) = h(y;\theta)/c(\theta)$, where

[4]This logit transformation is not necessary for the method of Deylon et al. (1999), since there is always a maximizer of the estimated objective function which satisfies our parameter constraints.

Figure 7: Trajectory of estimates for p and q from two versions of stochastic approximation EM. Horizontal dashed lines give the maximum likelihood estimates.



(a) Gu and Li (1998)



(b) Deylon et al. (1999)

h is known but c is not. We note that $\int h(y;\theta)dy = c(\theta)$, and write

$$\log \frac{f(y;\theta)}{f(y;\theta_*)} = \log \frac{h(y;\theta)}{h(y;\theta_*)} - \log \frac{c(\theta)}{c(\theta_*)} \tag{27}$$

$$= \log \frac{h(y;\theta)}{h(y;\theta_*)} - \log \int \frac{f(y;\theta)}{c(\theta_*)} dy \tag{28}$$

$$= \log \frac{h(y;\theta)}{h(y;\theta_*)} - \log \int \frac{h(y;\theta)}{h(y;\theta_*)} f(y;\theta_*) dy \tag{29}$$

$$= \log \frac{h(y;\theta)}{h(y;\theta_*)} - \log \mathbb{E}_{\theta_*} \frac{h(y;\theta)}{h(y;\theta_*)} \tag{30}$$

$$\approx \log \frac{h(y;\theta)}{h(y;\theta_*)} - \log \frac{1}{M} \sum_{i=1}^{M} \frac{h(y_i;\theta)}{h(y_i;\theta_*)} \tag{31}$$

where the y_i are sampled iid from $f(y;\theta_*)$. The second term in line 31 may need to be modified if non-iid sampling is used.

An alternative formulation of the MCML method given by Gelfand and Carlin (1993) is more relevant for use with missing data (see also, Geyer, 1994). A similar derivation to the one given above shows that, in the presence of missing data, X,

$$\log \frac{f(y;\theta)}{f(y;\theta_*)} = \log \mathbb{E}_{\theta_*} \left[\frac{h(y,X;\theta)}{h(y,X;\theta_*)} \middle| Y = y \right] - \log \mathbb{E}_{\theta_*} \left[\frac{h(Y,X;\theta)}{h(Y,X;\theta_*)} \right] \tag{32}$$

$$\approx \log \left[\frac{1}{M_1} \sum_{i=1}^{M_1} \frac{h(y,X_i';\theta)}{h(y,X_i';\theta_*)} \right] - \log \left[\frac{1}{M_2} \sum_{j=1}^{M_2} \frac{h(Y_j,X_j;\theta)}{h(Y_j,X_j;\theta_*)} \right] \tag{33}$$

where the X_i' are generated from the conditional distribution of $X|Y = y$, the (Y_j, X_j) pairs are generated from the joint distribution of Y and X, and $h(y,x;\theta)$ is proportional to this joint distribution. Note that two Monte Carlo samples are required to evaluate (33). If the complete data likelihood is known exactly, then we can replace h with f in (32) and (33), and drop the second term (i.e. the one being subtracted).

The MCML procedure closely resembles importance sampling, with $f(y;\theta_*)$, $f_c(y,x;\theta_*)$ or $f_m(y,x;\theta_*)$ being the proposal distribution (see Section 6.1 for a discussion of importance sampling). Jank and Booth (2003) take this idea further, and suggest directly estimating $f(y;\theta)$ from the complete data density, $f_c(y,x;\theta)$, using importance sampling. That is, they write $f(y;\theta) = \int f_c(y,x;\theta)dx = \int [f_c(y,x;\theta)/g(x)]g(x)dx$, where g is an arbitrary density function. The final integral is then approximated by an average of samples drawn iid from g. While the method of Jank and Booth does present an interesting direction in which to generalize MCML, note that it is only applicable if f_c is known exactly (or, rather, up to a proportionality constant which does not depend on θ).

We apply the MCML method of Geyer (1994) on our blood type example. Using a Monte Carlo size of 1000 gives $\hat{p} = 0.298$ and $\hat{q} = 0.129$, which are both quite close to

the MLE. Comparisons with other methods have found that MCML is very sensitive to its starting point; see Section 5 or McCulloch (1997). We therefore repeat MCML, this time starting with the estimate obtained from its first application. Our second run gives $\hat{p} = 0.297$ and $\hat{q} = 0.128$, which is not appreciably closer to the MLE.

5 Comparisons Between Methods

Several authors have performed comparisons between methods discussed in Sections 3 and 4. In this section, we discuss these comparisons and their findings.

McCulloch (1997) use a simulation study to compare the Monte Carlo EM (MCEM) and Monte Carlo Maximum Likelihood (MCML) methods, along with a Monte Carlo version of the Newton-Raphson algorithm (MCNR). Their MCEM implementation starts with fixed Monte Carlo size, then increases this size at iterations 20 and 40. It is not clear how these jump points were selected, nor how convergence was assessed beyond examining plots. Their MCEM implementation thus most closely resembles that of Wei and Tanner (1990). A similar schedule of Monte Carlo sizes and termination was used for MCNR, whereas MCML used a much larger Monte Carlo size (the sample size for MCML was not increased). McCulloch also considers MCEM and MCNR followed by MCML (i.e. using MCEM or MCNR to choose the reference parameter for MCML). This comparison is made using a logit-normal mixed-effects model with one random effect and one fixed effect. McCulloch found that MCEM and MCNR perform better than MCML alone, but that following either MCEM or MCNR with MCML was even better. They did not find that following-up with multiple iterations of MCML was preferable to a single run of MCML.

In addition to presenting their stochastic approximation EM (SAEM) method, Gu and Li (1998) compare this method with MCEM. They give a comparative analysis on a dataset of motorette failure times (see Rain et al., 2016, for a diagram and explanation of what a motorette is). The model being fit is a linear regression with right-censoring. Their MCEM implementation is the same as that of Wei and Tanner (1990) (more precisely, they say their implementation "is from Tanner (1993)", which matches Wei and Tanner). It is not clear what Monte Carlo size they use to start, and it appears that this size is never augmented. Their SAEM implementation is as previously described in their paper (see Section 4.1), with an MC size of 1, step size at iteration k (i.e., α_k) of $1/k$, and pre-multiplying matrix chosen adaptively based on the current iteration, as given by Equation (13) of their paper. Gu and Li find that SAEM converges much more quickly than MCEM. In fact, based on their Figure 1, it is not clear that MCEM is converging to the MLE at all. These authors also give a heuristic argument that SAEM should converge much more quickly than MCEM based on the number of MC samples used at each iteration and the number of iterations required to converge to the MLE. However, this argument is based on a fixed Monte Carlo size at each iteration and is thus not directly relevant to the MCEM implementations discussed in Section 3.

Booth and Hobert (1999) compare their MCEM implementation with that of Wei and Tanner (1990) as presented in McCulloch (1997). They investigate performance on three datasets: the logit-normal mixed-effects model from McCulloch (1997), a dataset comparing smoking with lung cancer (Dorn, 1954) and the salamander dataset given in McCullagh and Nelder (1989). On McCulloch's dataset, Booth and Hobert ran their own method to convergence, then ran Wei and Tanner's method for the same amount of time. They found that their method converges more quickly to the observed data MLE than the method of Wei and Tanner does (the model here is sufficiently simple that the observed data MLE can be obtained directly). They also investigate "pure Monte Carlo error" by starting both methods at the observed data MLE (since the MLE is a fixed point of EM, any change here is error due to Monte Carlo variability), and find that their method performs better on this metric as well.

Note that the comparisons made by Booth and Hobert use rejection sampling or importance sampling for their own method and Markov chain Monte Carlo (MCMC) sampling for the method of Wei and Tanner (1990) (see Section 6 for a discussion of these simulation techniques). Of note is that MCMC typically requires a "burn-in" period to reach the required stationary distribution, so the results presented by Booth and Hobert may not show the best that we can expect from MCMC.

Booth et al. (2001) compare the MCEM, SAEM and MCML (referred to as stochastic maximum likelihood, or SML) methods. They perform their comparison on a simple one-way mixed-effects linear model, with known variance component and error variance. Their MCEM implementation matches that of Booth and Hobert (1999), while their SAEM implementation is that of Jank (2006) (or Delyon et al., 1999). Note that although the presentation of SAEM in Booth et al. appears different from ours, it is not hard to show that the two are equivalent. Their MCML implementation uses the missing data distribution with a fixed value for the unknown parameter as reference distribution. The comparison between these methods is done partly analytically and partly by simulation. The mean squared errors (MSEs) of MCEM and SAEM for reproducing the MLE can be obtained analytically. Booth et al. thus compare these two methods directly and find that MCEM performs better when the problem is harder (i.e. larger variance component). However, the MSE of MCML must be approximated by simulation. This MSE is estimated using 500 replicates, and a 95% Wald-type confidence interval is constructed. The upper and lower bounds of this confidence interval are then compared to the analytical MSE of MCEM (the authors do not compare MCML with SAEM). The result is that MCML is competitive with MCEM, and that MCML even performs better for some parameter settings (typically, when the variance component is small). Booth et al. point out that the comparison between MCEM and MCML is not entirely fair here though, because taking the variance component as known makes simulation for MCML unrealistically easy. In more serious problems, difficulty in choosing a proposal distribution for MCML will likely lead to worse performance.

Booth et al. (2001) also extend the heuristic argument in Gu and Li (1998) to account

for Monte Carlo sizes changing with MCEM iteration. They argue that, in the scalar case, MCEM should converge more quickly than SAEM when the fraction of missing information is larger than $\exp(-1)$. The fraction of missing information is defined as $\mathcal{I}_c(\hat{\theta})^{-1}\mathcal{I}_m(\hat{\theta})$, where $\hat{\theta}$ is the observed data MLE (see Proposition 2.2 for definitions of $\mathcal{I}_m$ and $\mathcal{I}_c$). This quantity is closely related to the convergence rate of the EM algorithm (Meng and Rubin, 1994; McLachlan and Krishnan, 2008). Booth et al. also find that MCML outperforms both MCEM and SAEM. However, they use a proposal distribution for MCML which contains information about the parameters, thereby giving this method an unfair advantage. This is consistent with the findings of McCulloch (1997) about MCML, where this method can perform very well, but is highly sensitive to the choice of proposal distribution.

Jank and Booth (2003) extend the work of Booth et al. (2001), specifically the comparison between MCEM and MCML. Jank and Booth focus on analytical comparisons; as such, they use a fixed Monte Carlo size across iterations to make their calculations more tractable. They also use an unrealistic proposal distribution for MCML which requires that we know the observed data MLE. The authors derive the asymptotic variance of MCEM and MCML, and investigate their asymptotic relative efficiency (ARE). It turns out that the ARE depends directly on the eigenvalues of the (matrix-valued) fraction of missing information defined above. In particular, the efficiency of MCEM relative to MCML goes to infinity as the fraction of missing information goes to 1. That is, as a problem gets harder in the sense that less information is available in the observed data, we expect MCEM to perform better relative to MCML. We also expect MCML to perform worse on real problems, since the above analysis is based on an inaccessible proposal distribution. Jank and Booth illustrate their analytical calculations on the one-way mixed-effects linear model from Booth et al. (2001) and a logistic-normal generalized linear mixed-effects model. The latter consists of a simulation study which compares, among other things, the effect of the choice of proposal distribution on MCML. They found that the average estimates from MCEM and MCML (i.e. averaged over simulation replicates) are fairly consistent, but that variability of the MCML estimates is much higher. In particular, entries in the empirical covariance matrix for MCML grow rapidly as the reference parameter value for the proposal distribution moves away from the MLE.

Caffo et al. (2005) compare their version of MCEM with that of Booth and Hobert (1999) on simulated data from the logit-normal mixed-effects model of McCulloch (1997). Specifically, Caffo et al. simulate 10000 datasets and compare their method to the one proposed by Booth and Hobert (1999), with the latter terminating after the change in estimated parameter is small for between 1 and 4 consecutive iterations. They compare methods on how many draws from the missing data distribution are used (a measure of computational cost), the fraction of time spent in the final iteration, and the relative error in estimating both the MLEs of the parameters and their covariance matrix. Caffo et al. find that, for a fixed amount of computing, their method performs a bit worse than Booth and Hobert (1999) on estimating the parameters, but that their estimates of the covariance matrix are more accurate. They also find that their method spends a much larger fraction

of its time in the final iteration. They argue that this is an advantage, since the Monte Carlo sample used at the final MCEM iteration can be used to estimate other moments of the missing data distribution.

5.1 Computational Cost

In our blood type example, all methods perform reasonably well recovering the observed data MLE. However, the amount of computing required differs wildly between methods. In order to better facilitate choosing between methods in practice, we present a comparison between the methods introduced in Sections 3 and 4 on several quantities. First, we give the relative error for estimating the observed data MLE. Next, we measure computational cost, both by wall time and by number of Monte Carlo samples drawn. Since each method is stochastic, we repeat the analyses 100 times on the same dataset and average results for each performance metric. Note that the MCEM method of Wei and Tanner (1990) and both SAEM methods require the analyst to use their judgement in assessing convergence. Since this is not practical in a benchmarking comparison, we instead select a moderately large number of iterations for each method. We see these as plausible values for first-pass analyses. See the discussion of each method in Sections 3 and 4 for specific choices of tuning parameters.

Table gives the results of our comparison.

5.2 Synthesis

Many of the comparisons we have discussed (McCulloch, 1997; Gu and Li, 1998; Jank and Booth, 2003) use unsophisticated implementations of MCEM. In particular, as far as we can tell, Gu and Li (1998) use a poor implementation which never increases the Monte Carlo size, so it is unsurprising that they found MCEM performs poorly. Jank and Booth (2003) also use a fixed Monte Carlo size, but theirs is sufficiently large that we can expect the behaviour of MCEM to be close to that of the deterministic EM algorithm. Similarly, McCulloch (1997) increase the Monte Carlo size at fixed iterations, and the final size is quite large (5000 for the final ten iterations). Only Booth and Hobert (1999) use an adaptive rule for choosing the Monte Carlo size.

Despite the above concerns about Monte Carlo size, MCEM tends to perform quite well in simulations and analytical comparisons. Only Gu and Li (1998) and Booth et al. (2001) found an instance where MCEM performed substantially worse than another method, and both of these simulation studies had features which biased results away from MCEM. Additionally, (McCulloch, 1997) found that following MCEM with MCML tends to improve performance over MCEM alone. We see this as an endorsement of MCEM, since MCML is known to be sensitive to how well the proposal distribution (i.e. the output of MCEM) approximates the target distribution. These findings suggest that the MCEM algorithm should be one of the first methods considered when approaching a missing data problem

in which the calculations required to implement EM are intractable.

When selecting which implementation of the MCEM algorithm to use, unfortunately, limited information is available. Caffo et al. (2005) offer one comparison between their method and that of Booth and Hobert (1999), although the findings are not conclusive for one method over the other. This problem is a clear gap in the MCEM literature, and its solution would make MCEM methodology more useful to statistics practitioners.

6 Simulation

An obstacle to implementing the MCEM algorithm which was not addressed in Section 3, and which is also relevant to many of the methods described in Section 4, is how to generate the necessary Monte Carlo samples. It is in general a hard problem to simulate from arbitrary conditional distributions. In this section, we discuss a few methods for simulating the required observations at each step of the MCEM algorithm. All the topics that we cover here have their own bodies of literature, which we cannot hope to cover in their entirety. We instead give only a brief overview, focusing on aspects which are particularly relevant to use with the MCEM algorithm, and direct the reader to other, more focused, reviews.

We start by discussing importance sampling, which is a method for using a sample from one distribution to compute moments of another distribution. Next, we cover Markov chain Monte Carlo sampling, which consists of constructing and sampling from a Markov chain whose stationary distribution matches the distribution from which we want to simulate. Finally, we briefly touch on rejection sampling, a way to simulate exactly from a target distribution at the expense of increased computation time, and sequential Monte Carlo, which generates a sequence of samples whose distributions converge to the target distribution. The latter two methods receive less attention because, to our knowledge, they have not been as widely used in the MCEM literature.

6.1 Importance Sampling

Broadly speaking, importance sampling is a framework for approximating intractable expectations. A typical use case is when we want to evaluate the expected value of some function, h, under a distribution $\mathbb{F}$, and this expectation is not just analytically intractable, but the distribution $\mathbb{F}$ is impossible (or impractical) to sample from. The latter restriction prevents us from using ordinary Monte Carlo integration. Instead, we select another distribution, $\mathbb{G}$, which is easier to work with, and observe that $\mathbb{F}h = \mathbb{G}[h \cdot (f/g)]$, where f and g are the densities of $\mathbb{F}$ and $\mathbb{G}$ respectively. Provided that $\mathbb{G}$ is easy to sample from, we can estimate this alternative expression for our target expectation via Monte Carlo integration with samples drawn from $\mathbb{G}$. We call $\mathbb{F}$ the target distribution and $\mathbb{G}$ the proposal distribution.

A classic reference on importance sampling and other Monte Carlo methods is the book by Robert and Casella (2004); particularly Chapters 3 and 4. Chapter 8 of the book by Chopin and Papaspiliopoulos (2020) gives a more current overview of importance sampling, with a focus on its application to Sequential Monte Carlo. Elvira and Martino (2022) give a survey of modern methods for extending the importance sampling framework, with an emphasis on two main approaches: multiple importance sampling and adaptive importance sampling. See Elvira et al. (2019) for a review of multiple importance sampling methods, and Bugallo et al. (2017) for more on adaptive importance sampling. Agapiou et al. (2017) give a survey paper level treatment of some more theoretical considerations for importance sampling.

In the rest of this section, we describe a few simple modifications which can ease implementation and improve performance when using importance sampling with the MCEM algorithm.

When using importance sampling with the MCEM algorithm, our target distribution is the missing data distribution (i.e. the conditional distribution of the missing data given the observed data). In some settings, this distribution may be difficult to describe exactly. However, the integrand is the (log-)likelihood of the complete data distribution, so it is reasonable to expect that we can evaluate this complete data density. From the definition of conditional probability, the missing data density is proportional to the complete data density, provided that we treat the latter as a function of the missing data and hold the observed data fixed. The proportionality constant here is the observed data density, so it is unlikely that we will be able to normalize the missing data density exactly (otherwise, we could just work directly with the observed data likelihood).

A simple modification of importance sampling, which circumvents the need for exact normalization, is to compute importance weights using un-normalized densities, then normalize them to sum to one. This is referred to as "auto-normalized", or "self-normalized", importance sampling (see, e.g., Elvira and Martino, 2022). The reason self-normalized importance sampling works is that the unknown normalizing constant cancels in the numerator and denominator of our normalized weights. In fact, our proposal distribution can also be un-normalized, and the ratio of the two normalizing constants cancels when we normalize our weights.

There are, however, disadvantages of self-normalized importance sampling compared to the exact importance sampling which we can do when both target and proposal densities are known exactly. One important limitation is that our estimator of $\mathbb{F}h$ is no longer unbiased, and is instead only asymptotically unbiased. In fact, as a ratio estimator, the standard error of a self-normalized importance sampling estimator can only be determined asymptotically. This is fine in problems where it is easy to sample from our proposal distribution, $\mathbb{G}$, but in high dimensional problems for example, even simulating from $\mathbb{G}$ may be costly, and more care must be taken with the discrepancy between asymptotic results and finite-sample behaviour.

It is well-known that the performance of an importance sampling estimator depends on

how closely the proposal distribution matches the target (Robert and Casella, 2004). One thing that can go wrong is if the importance weight (i.e. the likelihood ratio between these two distributions) does not have sufficiently many finite moments (Agapiou et al., 2017). A simple modification to our importance weights which guarantees infinitely many finite moments is to specify a threshold value, and truncate any weights which fall above the threshold (i.e., set weights which fall above this threshold equal to the threshold value). This "truncated importance sampling" method is proposed and analysed by Ionides (2008). Two strategies are proposed in this work for selecting the threshold: the first is to simply use the square root of the Monte Carlo size, $\sqrt{M}$, while the latter is based on unbiased risk estimation and gives a value better tailored to the specific problem. Note that these recommendations are based on exact importance sampling. If self-normalization is used, the general recommendation is instead to truncate at $\sqrt{M}$ times the mean of the un-normalized weights. Choosing the threshold level for truncated importance sampling requires managing the bias-variance trade-off (Hastie et al., 2009). Truncating weights reduces the variance of our importance sampling estimator, but also introduces bias. This trade-off highlights the importance of selecting an appropriate threshold value: too large and the variance reduction will be negligible, but too small and the bias will be unacceptable.

Vehtari et al. (2022) propose an alternative method for handling large importance weights, called Pareto Smoothed Importance Sampling (PSIS). The idea of this method is to fit a Generalized Pareto Distribution to the largest importance weights, then replace those large weights with quantiles of the fitted distribution. One of the parameter's fitted values also serves as a useful diagnostic for how closely our proposal distribution matches the target. An advantage of PSIS over the truncated importance sampling method of Ionides (2008) is that the bias of PSIS is smaller, although it is not clear in general which method has better mean-squared error (see Vehtari et al., 2022, for extensive numerical comparisons).

An approach presented by Quintana et al. (1999) and Levine and Casella (2001) seeks to use importance sampling to save computing time when running the MCEM algorithm. This computational efficiency is especially important in their context where Markov chain Monte Carlo sampling is used (see 6.2), so generating a single sample takes quite some time. Their idea is to generate a single sample from the missing data distribution with some reference value for θ, θ_{ref}. This sample is then re-used at every MCEM iteration, with conditional expectations under the current parameter value computed by taking importance ratios with respect to θ_{ref}. An important question here is whether a single proposal distribution can be adequate for every target distribution along the MCEM trajectory. To address this concern, both sets of authors suggest running MCEM for a few iterations with fresh importance samples and only drawing the sample which will be used for their method after a sufficiently long "burn-in" period (Quintana et al. use five MCEM iterations, while Levine and Casella run the algorithm for one minute).

6.2 Markov Chain Monte Carlo

The core idea of Markov chain Monte Carlo (MCMC) sampling is to construct a Markov chain from which we can simulate, and which has stationary distribution equal to the target distribution. Popular methods to construct such a Markov chain are Gibbs sampling and the Metropolis-Hastings algorithm. See Gelman et al. (2013) or Robert and Casella (2004) for excellent textbook-length overviews. A popular implementation of MCMC sampling is the `Stan` programming language (Stan Development Team, 2022), and its `R` interface, `RStan` (Stan Development Team, 2023).

Both Gibbs sampling and Metropolis-Hastings start with a random variable, $X = (X_1, \ldots, X_d)$, which we wish to simulate. Let f be the density of X. These methods proceed by iteratively simulating draws of the vector X from a distribution which depends on the draw from the previous iteration. Note that our sampler may need time to reach its stationary distribution. It is therefore common to use a "burn-in" period, which amounts to discarding some number of draws from the beginning of the sample.

Gibbs sampling is based on successively sampling each component of X conditional on all the other components. The order in which this conditioning is performed is a bit subtle, however. When generating X_i, we condition on the values of $X_1, \ldots, X_{i-1}$ from the current iteration and the values of $X_{i+1}, \ldots, X_d$ from the previous iteration. Once we reach the end of X, we start a new iteration. More generally, we can group the components of X, and simulate an entire group conditional on the others.

The Metropolis-Hastings algorithm closely resembles rejection sampling. We start each iteration by generating a candidate value of X from some proposal distribution (this distribution may depend on the previous iteration's value of X). Write $J(x|x_0)$ for the proposal density, where x_0 is the value of X from the previous iteration. Next, we define an acceptance probability, $r := f(x)J(x_0|x)/f(x_0)J(x|x_0)$, and accept the proposed value of X with probability $r \wedge 1$. With probability $(1 - r) \vee 0$, we reject the proposed x and instead set the current iteration's value to x_0. We then proceed to the next iteration. Note that rejecting still adds an observation to our Monte Carlo sample, this value just happens to be identical to the one proceeding it.

Limit theory for estimators based on MCMC sampling are more complicated than the similar theory for estimators based on iid or importance sampling (Geyer, 1991). This additional complexity stems from the dependence between draws from an MCMC sampler. While convergence for iid and importance sampling can be established using the Law of Large Numbers and the classical Central Limit Theorem, similar results for MCMC sampling make use of the Ergodic Theorem and Markov chain Central Limit Theorem. A challenge in implementing the Markov chain Central Limit Theorem is that the asymptotic variance depends on pairwise covariances between points in the chain with arbitrarily large lags. Estimation of this asymptotic variance is therefore challenging in practice. See, e.g., Chapters 6 and 7 of Robert and Casella (2004).

Levine and Casella (2001) propose a method to simplify the analysis of estimates based

on MCMC sampling. Their approach consists of subsampling the original chain using Poisson spacings in such a way that elements of the subsample are approximately independent. The result is that a target function averaged over the subsampled chain satisfies a classical Central Limit Theorem (i.e. one with no covariance terms in the asymptotic variance). We can estimate the mean and standard error of our subsample estimator using the entire chain, then construct confidence intervals for the mean of the target function based on our subsample. Levine and Casella apply this strategy for constructing confidence intervals along the same lines as Booth and Hobert (1999). Here, we construct a confidence interval for the gradient in the EM objective function, and increase the Monte Carlo size if the confidence interval for the current iteration contains the subsample estimate from the previous iteration. This work is modified to provide uncertainty quantification for the MCEM update instead of the EM score by Levine and Fan (2004). Furthermore, Levine and Fan give a principled argument for how much to increase the Monte Carlo sample size by when doing so is deemed necessary.

6.3 Other Sampling Methods

While importance sampling and MCMC are the most discussed sampling schemes in the context of the MCEM algorithm, some others do exist. Rejection sampling and sequential Monte Carlo are two such alternatives. However, we are not aware of many instances when these methods are applied to MCEM problems.

Rejection sampling closely resembles importance sampling, except instead of weighting each proposal by the likelihood ratio, we either accept or reject the proposed observation with probability proportional to the likelihood ratio (Chopin and Papaspiliopoulos, 2020). Typically, rejection sampling is continued until the number of accepted proposals reaches a desired sample size. These accepted points are then treated as an iid sample from the target distribution. Although the output of rejection sampling sounds ideal (much of the difficulty with importance sampling comes from having to account for simulated points not having been drawn from the target distribution), the cost comes in increased computation time. The number of draws from the proposal required to get a fixed number of accepted draws is random, and can be quite high if the proposal distribution does not closely match the target. Indeed, if we instead fix the number of draws from the proposal distribution, importance sampling can be shown to have lower variance than the corresponding rejection sampling scheme (see Section 8.8 of Chopin and Papaspiliopoulos, 2020). Booth and Hobert (1999) discuss the use of rejection sampling with their implementation of MCEM, but found that importance sampling was faster and gave similar results.

Sequential Monte Carlo (SMC), is a form of adaptive sampling in which a sequence of samples is generated such that the distribution of these samples converges to some target. The update from one sample to the next seeks to balance improving the proposal with maintaining computational efficiency. See Del Moral et al. (2006) for a survey paper, or Chopin and Papaspiliopoulos (2020) for a book-length overview of SMC. Moffa and Kuipers

(2014) use SMC in an MCEM analysis to sample from a truncated multivariate normal distribution with difficult truncation domain.

7 Conclusions

The EM algorithm is a very useful tool for the analysis of missing data. The MCEM algorithm and related methods allow us to apply ideas from the EM algorithm in contexts where analytical calculations are intractable. In this paper, we present several implementations of the MCEM algorithm, as well as some alternative methods which can be applied to missing data problems. We also discuss numerous comparisons between the MCEM algorithm and related methods. This gives both practitioners and researchers a firm basis for selecting a method to use in an analysis or for further investigation. In addition, we address the practical concern of how to generate the Monte Carlo samples required for our methods. Specifically, we discuss several methods for simulating from an arbitrary target distribution when direct sampling is not available.

Over the course of writing this review, we have identified a number of gaps in the literature. Addressing any of these problems would be a significant contribution to our understanding of the computational analysis of missing data models.

- Comparisons between methods. More work is required to identify when one method should be used over another. This is especially true of newer work like Caffo et al. (2005), which did not exist when other systematic comparisons were performed.
-

We hope that this review helps make the MCEM algorithm, and Monte Carlo methods for missing data more generally, more accessible. We also hope that our work generates increased interest and development in the important field of computational methods for missing data.

Appendix A Likelihood for Gene Frequency Estimation

In this appendix, we present details for the analysis of our example of estimating gene frequency. See Section 2.2 for formulation of the model and definition of notation.

A.1 Observed Data Likelihood, Score and Information

Let π_i be the probability of blood type i. The observed data log-likelihood for our model can be written as follows:

$$\ell(\theta; y) = \log \binom{n}{y} + \sum y_i \log \pi_i(\theta) \tag{34}$$

$$\equiv \sum y_i \log \pi_i \tag{35}$$

$$\equiv 2y_1 \log r + y_2 \log(p^2 + 2pr) + y_3 \log(q^2 + 2qr) + y_4 \log pq \tag{36}$$

where we use $\equiv$ to denote equality up to additive constants which do not depend on θ.

Differentiating ℓ with respect to θ and recalling that $r = 1 - p - q$, so $\partial_p r = \partial_q r = -1$, we get the following expression for the observed data score, S.

$$S(\theta; y) = \begin{pmatrix} \partial_p \ell(\theta; y) \\ \partial_q \ell(\theta; y) \end{pmatrix}, \text{where} \tag{37}$$

$$\partial_p \ell(\theta; y) = -\frac{2y_1}{r} + \frac{2ry_2}{p^2 + 2pr} - \frac{2qy_3}{q^2 + 2qr} + \frac{y_4}{p} \tag{38}$$

$$\partial_q \ell(\theta; Y) = -\frac{2y_1}{r} - \frac{2py_2}{p^2 + 2pr} + \frac{2ry_3}{q^2 + 2qr} + \frac{y_4}{q} \tag{39}$$

Solving the score equation, $S(\theta) = 0$, thus reduces to solving a system of two polynomials in p and q. Since p and q are proportions, we reject any roots outside the unit simplex.

Differentiating ℓ again and multiplying by -1 gives the observed data information matrix, I. To simplify notation, let $p_y = p^2 + 2pr$ and $q_y = q^2 + 2qr$.

$$I(\theta; y) = -\begin{bmatrix} \partial_p^2 \ell(\theta; y) & \partial_{p,q} \ell(\theta; y) \\ \partial_{p,q} \ell(\theta; y) & \partial_q^2 \ell(\theta; y) \end{bmatrix}, \text{where} \tag{40}$$

$$\partial_p^2 \ell(\theta; y) = \frac{2y_1}{r^2} + \frac{2y_2(p_y + 2r^2)}{p_y^2} + \frac{4y_3 q^2}{q_y^2} + \frac{y_4}{p^2} \tag{41}$$

$$\partial_{p,q} \ell(\theta; y) = \frac{2y_1}{r^2} + \frac{2y_2 p^2}{p_y^2} + \frac{2y_3 q^2}{q_y^2} \tag{42}$$

$$\partial_q^2 \ell(\theta; y) = \frac{y_1}{r^2} + \frac{4y_2 p^2}{p_y^2} + \frac{2y_3(q_y + 2r)}{q_y^2} + \frac{y_4}{q^2} \tag{43}$$

The asymptotic standard error of our MLE is I^{-1}, evaluated at the estimate.

A.2 Complete Data Likelihood, Score and Information

The complete data distribution for our model can be written as follows. Write ρ_i for the probability of genotype i. See Table 2 for the values of these probabilities.

$$\ell_c(\theta; y, x) = \log \binom{n}{x} + \sum x_i \log \rho_i(\theta) \tag{44}$$

$$\equiv \sum y_i \log \rho_i \tag{45}$$

$$\equiv 2x_1 \log r + x_2 \log pr + 2x_3 \log p + x_4 \log qr + 2x_5 \log q + x_6 \log pq \tag{46}$$

$$= (2x_1 + x_2 + x_4) \log r + (x_2 + 2x_3 + x_6) \log p + (x_4 + 2x_5 + x_6) \log q \tag{47}$$

$$= n_O \log r + n_A \log p + n_B \log q \tag{48}$$

where n_O, n_A and n_B are the number of times allele O, A and B arise respectively in the sampled genotypes. Note that ℓ_c depends on y only through x, so we suppress y from our notation for complete data quantities. The complete data score function is

$$S_c(\theta; x) = \begin{pmatrix} \partial_p \ell_c(\theta; x) \\ \partial_q \ell_c(\theta; x) \end{pmatrix}, \text{where} \tag{49}$$

$$\partial_p \ell_c(\theta; x) = \frac{x_2 + 2x_3 + x_6}{p} - \frac{2x_1 + x_2 + x_4}{r} = \frac{n_A}{p} - \frac{n_O}{r} \tag{50}$$

$$\partial_p \ell_c(\theta; x) = \frac{x_4 + 2x_5 + x_6}{q} - \frac{2x_1 + x_2 + x_4}{r} = \frac{n_B}{q} - \frac{n_O}{r} \tag{51}$$

Notice that the score is linear in x. To make this relationship explicit, we write $S_c(\theta; x) = \mathscr{S}(\theta)x$, where $\mathscr{S}(\theta) \in \mathbb{R}^{2\times 6}$ is a matrix consisting of the coefficients on x in (50) and (51). We will make use of this linearity in Section A.5.

Next, we give the information matrix for the complete data.

$$I_c(\theta; x) = -\begin{bmatrix} \partial_p^2 \ell_c(\theta; x) & \partial_{p,q} \ell_c(\theta; x) \\ \partial_{p,q} \ell_c(\theta; x) & \partial_q^2 \ell_c(\theta; x) \end{bmatrix}, \text{where} \tag{52}$$

$$\partial_p^2 \ell_c(\theta; x) = \frac{x_2 + 2x_3 + x_6}{p^2} + \frac{2x_1 + x_2 + x_4}{r^2} = \frac{n_A}{p^2} + \frac{n_O}{r^2} \tag{53}$$

$$\partial_{p,q} \ell_c(\theta; x) = \frac{2x_1 + x_2 + x_4}{r^2} = \frac{n_O}{r^2} \tag{54}$$

$$\partial_q^2 \ell_c(\theta; x) = \frac{x_4 + 2x_5 + x_6}{q^2} + \frac{2x_1 + x_2 + x_4}{r^2} = \frac{n_B}{q^2} + \frac{n_O}{r^2} \tag{55}$$

A.3 Missing Data Distribution

Many quantities which arise in the EM and MCEM algorithms depend on the missing data distribution (i.e. the conditional distribution of X given $Y = y$). This distribution is best described componentwise in X. First, note that $X_1 = y_1$ and $X_6 = y_4$. Next, we have that

$X_2 + X_3 = y_2$ and $X_4 + X_5 = y_3$. Thus, we can write $X_2|Y = y \sim \text{Bin}(y_2, 2pr/(p^2 + 2pr))$ and $X_4|Y = y \sim \text{Bin}(y_3, 2qr/(q^2 + 2qr))$. Finally, we recover X_3 and X_5 by subtracting X_2 from y_2 and X_4 from y_3 respectively.

We make frequent use of the first few conditional moments of X, so they are listed here for convenience. Let $\alpha_1 = 2pr/(p^2 + 2pr)$ be the probability parameter for the binomial distribution of X_2 given Y, and $\alpha_2 = 1 - \alpha_1$. Similarly, let $\beta_1 = 2qr/(q^2 + 2qr)$ correspond to X_4 and $\beta_2 = 1 - \beta_1$.

$$\mathbb{E}(X|Y = y) = \big(y_1, y_2\alpha_1, y_2\alpha_2, y_3\beta_1, y_3\beta_2, y_4\big)^T \tag{56}$$

$$=: \mu_m \tag{57}$$

$$\mathbb{V}(X|Y = y) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & y_2\alpha_1\alpha_2 & -y_2\alpha_1\alpha_2 & 0 & 0 & 0 \\ 0 & -y_2\alpha_1\alpha_2 & y_2\alpha_1\alpha_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & y_3\beta_1\beta_2 & -y_3\beta_1\beta_2 & 0 \\ 0 & 0 & 0 & -y_3\beta_1\beta_2 & y_3\beta_1\beta_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{58}$$

$$=: \Sigma_m \tag{59}$$

$$\mathbb{E}(XX^T|Y = y) = \Sigma_m + \mu_m\mu_m^T \tag{60}$$

Conditional expectations of the number of alleles of each kind will be of particular interest.

$$\nu_O := \mathbb{E}(n_O|y) \tag{61}$$
$$= 2y_1 + \frac{y_2 pr}{p^2 + 2pr} + \frac{y_3 qr}{q^2 + 2qr} \tag{62}$$
$$= 2y_1 + y_2\left(\frac{\rho_2}{\rho_2 + \rho_3}\right) + y_3\left(\frac{\rho_4}{\rho_4 + \rho_5}\right) \qquad \left(= 2y_1 + y_2\left(\frac{\rho_2}{\pi_2}\right) + y_3\left(\frac{\rho_4}{\pi_3}\right)\right) \tag{63}$$

$$\nu_A := \mathbb{E}(n_A|y) \tag{64}$$
$$= \frac{2y_2 pr}{p^2 + 2pr} + \frac{2y_2 p^2}{p^2 + 2pr} + y_4 \tag{65}$$
$$= y_2\left(\frac{\rho_2}{\rho_2 + \rho_3} + \frac{2\rho_3}{\rho_2 + \rho_3}\right) + y_4 \qquad \left(= y_2\left(\frac{\rho_2}{\pi_2} + \frac{2\rho_3}{\pi_2}\right) + y_4\right) \tag{66}$$
$$= y_2\left(1 + \frac{p^2}{p^2 + 2pr}\right) + y_4 \tag{67}$$

$$\nu_B := \mathbb{E}(n_B|y) \tag{68}$$
$$= \frac{2y_3 qr}{q^2 + 2qr} + \frac{2y_3 q^2}{q^2 + 2qr} + y_4 \tag{69}$$
$$= y_3\left(\frac{\rho_4}{\rho_4 + \rho_5} + \frac{2\rho_5}{\rho_4 + \rho_5}\right) + y_4 \qquad \left(= y_3\left(\frac{\rho_4}{\pi_3} + \frac{2\rho_5}{\pi_3}\right) + y_4\right) \tag{70}$$
$$= y_3\left(1 + \frac{q^2}{q^2 + 2qr}\right) + y_4 \tag{71}$$

A.4 EM Algorithm

In order to apply the EM algorithm, we must construct and optimize the EM objective function. That is, we must compute $Q(\theta|\theta_0) = \mathbb{E}_{\theta_0}\left[\ell_c(\theta; y, X)|Y = y\right]$. The EM objective function can be written as

$$Q(\theta|\theta_0) := \mathbb{E}_{\theta_0}[\ell_c(\theta; X)|Y = y] \tag{72}$$
$$\equiv \nu_O^{(0)} \log r + \nu_A^{(0)} \log p + \nu_B^{(0)} \log q \tag{73}$$

where a superscript zero denotes that the quantity is computed by taking an expectation under θ_0. Differentiating Q with respect to p and q and setting the result to zero, we get

the following system of equations:

$$\frac{\nu_A^{(0)}}{p} = \frac{\nu_O^{(0)}}{r} \tag{74}$$

$$\frac{\nu_B^{(0)}}{q} = \frac{\nu_O^{(0)}}{r} \tag{75}$$

This system of equations can be used to solve for a fixed point of the EM algorithm by evaluating ν_O, ν_A and ν_B at θ instead of θ_0. Note that the fixed point equations which result from this substitution exactly match the observed data score equations given by Equations 38 and 39. Indeed, this relationship holds in general under mild conditions (Wu, 1983).

A.5 Asymptotic Standard Error

Recall that the EM algorithm computes the MLE, which has asymptotic covariance matrix equal to the inverse Fisher information matrix evaluated at the true parameter value. In practice, we estimate this covariance with the inverse of the observed information matrix evaluated at the MLE. Using Proposition 2.3, we can calculate the observed information matrix using conditional expectations of quantities derived from the complete data likelihood.

To this end, we need to evaluate the conditional expectations in expression (7) of Proposition 2.3. It is convenient for us to write $S_c(\theta) =: \mathscr{S}(\theta)X$ (see Appendix A.2). Then

$$I_c(\hat{\theta}) = \begin{bmatrix} \frac{\nu_A}{p^2} + \frac{\nu_O}{r^2} & \frac{\nu_O}{r^2} \\ \frac{\nu_O}{r^2} & \frac{\nu_B}{q^2} + \frac{\nu_O}{r^2} \end{bmatrix}, \text{and} \tag{76}$$

$$\mathbb{E}_{\hat{\theta}}[S_c(\hat{\theta})S_c(\hat{\theta})^T | Y = y] = \mathscr{S}(\hat{\theta})\mathbb{E}_{\hat{\theta}}\left[XX^T | Y = y\right]\mathscr{S}(\hat{\theta}) \tag{77}$$

$$= \mathscr{S}(\hat{\theta})(\Sigma_m + \mu_M \mu_M^T)\mathscr{S}(\hat{\theta}) \tag{78}$$

$$\tag{79}$$

While it is possible to expand the above expressions, they quickly become too long to easily interpret. We instead leave these as computational formulas and use them as a guide for writing `R` or `Julia` code.

Check for "Citation Needed" before publishing.

References

S. Agapiou, O. Papaspiliopoulos, D. Sanz-Alonso, and A. M. Stuart. Importance sampling: Intrinsic dimension and computational cost. *Statistical Science*, 32(3), 2017.

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Vial B. Shah. Julia: a fresh approach to numerical computing. *SIAM Review*, 59(1), 2017.

James G. Booth and James P. Hobert. Maximizing generalized linear mixed model likelihoods with an automated monte carlo em algorithm. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 61(1), 1999.

James G. Booth, James P. Hobert, and Wolfgang Jank. A survey of Monte Carlo algorithms for maximizing the likelihood of a two-stage hierarchical model. *Statistical Modelling*, 1 (4), 2001.

Vivek S. Borkar. *Stochastic Approximation: A dynamical systems viewpoint*. Springer, 2nd edition, 2022.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, 2010.

Mónica F. Bugallo, Víctor Elvira, Luca Martino, David Luengo, Joaquín Míguez, and Petar M. Djurić. Adaptive importance sampling: The past, the present, and the future. *IEEE Signal Processing Magazine*, 34(4), 2017.

Brian S. Caffo, Wolfgang Jank, and Galin L. Jones. Ascent-based Monte Carlo expectation-maximization. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2), 2005.

G. Celeux and J. Diebolt. The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. *Computational Statistics Quarterly*, 2 (1), 1985.

Gilles Celeux and Jean Diebolt. The EM and SEM algorithms for mixtures: statistical and numerical aspects. Technical report, INRIA, 1987.

Gilles Celeux and Jean Diebolt. A stochastic approximation type EM algorithm for the mixture problem. *Stochastics and Stochastic Reports*, 41(1-2), 1992.

Gilles Celeux, Didier Chauveau, and Jean Diebolt. On stochastic versions of the EM algorithm. Technical report, INRIA, 1995.

K. S. Chan and Johannes Ledolter. Monte Carlo EM estimation for time series models involving counts. *Journal of the American Statistical Association*, 90(429), 1995.

Nicolas Chopin and Omiros Papaspiliopoulos. *An Introduction to Sequential Monte Carlo*. Springer, 2020.

Citation Needed.

Laura Dean. Blood groups and red cell antigens [internet]. Technical report, National Center for Biotechnology Information (US), 2005.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3), 2006.

Bernard Delyon, Marc Lavielle, and Eric Moulines. Convergence of a stochastic approximation version of the EM algorithm. *The Annals of Statistics*, 27(1), 1999.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1), 1977.

Bernard Deylon, Marc Lavielle, and Eric Moulines. Convergence of a stochastic approximation version of the EM algorithm. *The Annals of Statistics*, 27(1), 1999.

Harold F. Dorn. The relationship of cancer of the lung and the use of tobacco. *The American Statistician*, 8(5), 1954.

Aryeh Dvoretzky. On stochastic approximation. In Jerzy Neyman, editor, *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, 1956.

Víctor Elvira, Luca Martino, David Luengo, and Mónica Bugallo. Generalized multiple importance sampling. *Statistical Science*, 34(1), 2019.

Víctor Elvira and Luca Martino. Advances in importance sampling. *arXiv:2102.05407v3*, 2022.

Gersende Fort and Eric Moulines. Convergence of the Monte Carlo expectation maximization for curved exponential families. *The Annals of Statistics*, 31(4), 2003.

Yoshiko Fujita, Masako Tanimura, and Katumi Tanaka. The distribution of the ABO blood groups in Japan. *Japanese Journal of Human Genetics*, 23, 1978.

Alan E. Gelfand and Bradley P. Carlin. Maximum-likelihood estimation for constrained- or missing-data models. *The Canadian Journal of Statistics*, 21(3), 1993.

Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. CRC Press, third edition, 2013.

Charles J. Geyer. Markov chain Monte Carlo maximum likelihood. *Interface Foundation of North America*, 1991.

Charles J. Geyer. On the convergence of Monte Carlo maximum likelihood calculations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 56(1), 1994.

Ming Gao Gu and Fan Hui Kong. A stochastic approximation algorithm with Markov chain Monte-Carlo method for incomplete data estimation problems. *Proceedings of the National Academy of Sciences*, 95(13), 1998.

Ming Gao Gu and Shaolin Li. A stochastic approximation algorithm for maximum-likelihood estimation with incomplete data. *The Canadian Journal of Statistics*, 26 (4), 1998.

Ming Gao Gu and Hong-Tu Zhu. Maximum likelihood estimation for spatial models by Markov Chain Monte Carlo stochastic approximation. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 63(2), 2001.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, 2009.

Edward L. Ionides. Truncated importance sampling. *Journal of Computational and Graphical Statistics*, 17(2), 2008.

Wolfgang Jank. Implementing and diagnosing the stochastic approximation EM algorithm. *Journal of Computational and Graphical Statistics*, 15(4), 2006.

Wolfgang Jank and James Booth. Efficiency of Monte Carlo EM and simulated likelihood in two-stage hierarchical models. *Journal of Computational and Graphical Statistics*, 12 (1), 2003.

J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3), 1952.

Harold J. Kushner and G. George Yin. *Stochastic Approximation Algorithms and Applications*. Springer, 1997.

Harold J. Kushner and G. George Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, 2nd edition, 2003.

Tze Leung Lai. Stochastic approximation: invited paper. *The Annals of Statistics*, 31(2), 2003.

Richard A. Levine and George Casella. Implementations of the Monte Carlo EM algorithm. *Journal of Computational and Graphical Statistics*, 10(3), 2001.

Richard A. Levine and Juanjuan Fan. An automated (Markov chain) Monte Carlo EM algorithm. *Journal of Statistical Computation and Simulation*, 74(5), 2004.

Thomas A. Louis. Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 44(2), 1982.

P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman & Hall/CRC, 2nd edition, 1989.

Charles E. McCulloch. Maximum likelihood algorithms for generalized linear mixed models. *Journal of the American Statistical Association*, 92(437), 1997.

Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*. Wiley, 2nd edition, 2008.

Xiao-Li Meng and Donald B. Rubin. On the global and componentwise rates of convergence of the EM algorithm. *Linear Algebra and its Applications*, 199, 1994.

Giusi Moffa and Jack Kuipers. Sequential Monte Carlo EM for multivariate probit models. *Computational Statistics and Data Analysis*, 72, 2014.

Ronald C. Neath. On convergence properties of the Monte Carlo EM algorithm. *Advances in Modern Statistical Theory and Applications*, 10, 2013.

Søren Feodor Nielsen. The stochastic EM algorithm: estimation and asymptotic results. *Bernoulli*, 6(3), 2000.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

David Oakes. Direct calculation of the information matrix via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 61(2), 1999.

B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4), 1992.

Fernando A. Quintana, Jun S. Liu, and Guido E. del Pino. Monte Carlo EM with importance reweighting and its applications in random effects models. *Computational Statistics & Data Analysis*, 29, 1999.

P. Rain, F. Loubeau, A. Durieux, F. Le Strat, and F. Fresnet. Using motorettes for the experimental and numerical determinations of the pdiv in an electric motor. In *2016 IEEE International Conference on Dielectrics (ICD)*, volume 2, pages 967–970, 2016.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3), 1951.

Christian P. Robert and George Casella. *Monte Carlo statistical methods*. Springer, second edition, 2004.

William Ruth and Richard Lockhart. MCEM_Survey, 2023. URL `https://github.com/wruth1/MCEM_Survey`.

Thomas P. Ryan and William H. Woodall. The most-cited statistical papers. *Journal of Applied Statistics*, 32(5), 2005.

Shayle R. Searle, George Casella, and Charles E. McCulloch. *Variance Components*. Wiley Interscience, 2006.

Stan Development Team. Stan modelling language users guide and reference manual, 2022. URL `http://mc-stan.org/`. Version 2.32.

Stan Development Team. RStan: the R interface to Stan, 2023. URL `https://mc-stan.org/`. R package version 2.21.8.

Martin A. Tanner. *Tools for Statistical Inference*. Springer, 2nd edition, 1993.

Martin A. Tanner. *Tools for Statistical Inference*. Springer, 3rd edition, 1996.

A. W. van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 1998.

Richard Van Noorden, Brendan Maher, and Regina Nuzzo. The top 100 papers. *Nature News*, 514(7524):550, 2014.

Aki Vehtari, Daniel Simpson, Andrew Gelman, Yuling Yao, and Jonah Gabry. Pareto smoothed importance sampling, 2022.

Greg C. G. Wei and Martin A. Tanner. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85(411), 1990.

C. F. Jeff Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1), 1983.

Index