Electronic Journal of Statistics
Vol. 9 (2015) 2293–2323
ISSN: 1935-7524
DOI: 10.1214/15-EJS1072

Approximately exact calculations for linear mixed models

Michael Lavine

Department of Mathematics and Statistics
UMass Amherst
Amherst, MA 01003
USA
e-mail: lavine@math.umass.edu

Andrew Bray

Reed College
3203 Southeast Woodstock Boulevard
Portland, Oregon 97202-8199
USA
e-mail: abray@reed.edu

and

Jim Hodges

Division of Biostatistics
School of Public Health
University of Minnesota
2221 University Ave SE, Suite 200
Minneapolis, MN 55414
USA
e-mail: hodge003@umn.edu

Abstract: This paper is about computations for linear mixed models having two variances, σ_e^2 for residuals and σ_s^2 for random effects, though the ideas can be extended to some linear mixed models having more variances. Researchers are often interested in either the restricted (residual) likelihood $\mathrm{RL}(\sigma_e^2, \sigma_s^2)$ or the joint posterior $\pi(\sigma_e^2, \sigma_s^2 \,|\, y)$ or their logarithms. Both $\log \mathrm{RL}$ and $\log \pi$ can be multimodal and computations often rely on either a general purpose optimization algorithm or MCMC, both of which can fail to find regions where the target function is high. This paper presents an alternative. Letting f stand for either RL or π, we show how to find a box B in the (σ_e^2, σ_s^2) plane such that

1. all local and global maxima of $\log f$ lie within B;
2. $\sup_{(\sigma_e^2,\sigma_s^2)\in B^c} \log f(\sigma_e^2, \sigma_s^2) \leq \sup_{(\sigma_e^2,\sigma_s^2)\in B} \log f(\sigma_e^2, \sigma_s^2) - M$ for a prespecified $M > 0$; and
3. $\log f$ can be estimated to within a prespecified tolerance ϵ everywhere in B with no danger of missing regions where $\log f$ is large.

Taken together these conditions imply that the (σ_e^2, σ_s^2) plane can be divided into two parts: B, where we know $\log f$ as accurately as we wish, and B^c, where $\log f$ is small enough to be safely ignored. We provide algorithms to find B and to evaluate $\log f$ as accurately as desired everywhere in B.

MSC 2010 subject classifications: Primary 62J05; secondary 62F99.

Received January 2015.

Contents

1. Introduction

Linear mixed models are an important class of statistical models. Books are written about them (e.g. Bryk and Raudenbush 2, Verbeke and Molenberghs 14, Hodges 6, West et al. 17), courses are taught about them, and they have many applications. Typical notation, which we adopt, is

$$y = X\beta + Zu + \epsilon \tag{1}$$

where y is a vector of n observations, X is a known $n \times p$ matrix, β is a vector of p unknown coefficients called fixed effects, Z is a known $n \times q$ matrix, u is a vector of q unknown coefficients called random effects, and ϵ is a vector of n errors. The term "mixed" is used when we treat u as a vector of random variables, thus mixing fixed and random effects in the same model. For linear mixed models where u and ϵ are modelled as Normal, researchers are often interested in the restricted likelihood function

$$\begin{aligned}\mathrm{RL}(\theta) = K|V(\theta)|^{-1/2}|X^tV^{-1}(\theta)X|^{-1/2} \\ \times \exp\left\{-\frac{1}{2}\left(y^tV^{-1}(\theta)y - \tilde{\beta}^t(\theta)X^tV^{-1}(\theta)X\tilde{\beta}(\theta)\right)\right\}\end{aligned} \tag{2}$$

where K is an unimportant constant, θ is a vector of unknown parameters in the covariance matrices of u and ϵ, $V(\theta)$ is the marginal covariance matrix of y implied by the covariance matrices of u and ϵ, and $\tilde{\beta}(\theta)$ is the generalized least-squares estimate of β, given $V(\theta)$. This manuscript deals with the special case in which we adopt the model

$$\epsilon \sim \mathrm{N}(0, \sigma_e^2\Sigma_e) \qquad u \sim \mathrm{N}(0, \sigma_s^2\Sigma_s)$$

where Σ_e and Σ_s are known matrices, often the identity, of the appropriate sizes and $\theta \equiv (\sigma_e^2, \sigma_s^2)$, two unknown variance parameters. The key for this manuscript is that θ contains only those two unknown variances and no others. Examples include random intercept models (including balanced and unbalanced one-way random effect models), additive models with one penalized spline, spatial models with one intrinsic conditional autoregression (ICAR) random effect, dynamic linear models with one system-level variance, and some multiple membership models (e.g. Browne et al. 1, McCaffrey et al. 9). [6] examines these and other examples and explains the importance of this special case.

[6] also unifies and generalizes [11] and [16] to show that in our special case, and a few others, $\log \mathrm{RL}(\sigma_e^2, \sigma_s^2)$ can be expressed as

$$\log \mathrm{RL}(\sigma_e^2, \sigma_s^2) = B - \frac{n_e}{2}\log(\sigma_e^2) - \frac{y^t \Gamma_c \Gamma_c^t y}{2\sigma_e^2} - \frac{1}{2}\sum_{j=1}^{s_z}\left[\log(a_j\sigma_s^2 + \sigma_e^2) + \frac{\hat{v}_j^2}{a_j\sigma_s^2 + \sigma_e^2}\right] \quad (3)$$

where

(1) B is an unimportant known constant;
(2) n_e is n minus the dimension of the space spanned by the columns of $[X|Z]$;
(3) Γ_c is $n \times n_e$ and spans the space orthogonal to $[X|Z]$ (so $y^t\Gamma_c\Gamma_c^t y$ is the residual sum of squares);
(4) s_z is the dimension of the space spanned by the columns of Z not already in the span of the columns of X; and
(5) the $\{a_j\}$ and $\{\hat{v}_j\}$ are known constants whose derivation is in the Appendix. All $a_j > 0$.

Thus the only unknowns are (σ_s^2, σ_e^2) and $\log \mathrm{RL}(\sigma_e^2, \sigma_s^2)$ is a function of just those two arguments.

As [6] further observes, if β is given an improper flat prior and σ_e^2 and σ_s^2 are given conjugate priors — say $\sigma_e^2 \sim \mathrm{InvGam}(\alpha_e, \beta_e)$ and $\sigma_s^2 \sim \mathrm{InvGam}(\alpha_s, \beta_s)$ — then

$$-(\alpha_e + 1)\log\sigma_e^2 - \beta_e/\sigma_e^2 - (\alpha_s + 1)\log\sigma_s^2 - \beta_s/\sigma_s^2$$

is added to (3) to yield the log posterior

$$\log \pi(\sigma_e^2, \sigma_s^2 \,|\, y) = B - \frac{n_e + 2\alpha_e + 2}{2}\log(\sigma_e^2) - \frac{y^t\Gamma_c\Gamma_c^t y + 2\beta_e}{2\sigma_e^2} - \frac{2\alpha_s + 2}{2}\log(\sigma_s^2) - \frac{2\beta_s}{2\sigma_s^2} - \frac{1}{2}\sum_{j=1}^{s_z}\left[\log(a_j\sigma_s^2 + \sigma_e^2) + \frac{\hat{v}_j^2}{a_j\sigma_s^2 + \sigma_e^2}\right]. \quad (4)$$

Equations (3) and (4) can both be written as a sum of multiples of logs and inverses of linear combinations $a_j\sigma_s^2 + b_j\sigma_e^2$, as in (5), where the summands with $j = s_z+1$ and $j = s_z+2$ are for the terms involving only σ_e^2 and σ_s^2, respectively,

and where we have dropped the irrelevant constant B. I.e.,

$$\log f(\sigma_e^2, \sigma_s^2) = -\frac{1}{2} \sum_{j=1}^{s_z+2} \left[c_j \log(a_j \sigma_s^2 + b_j \sigma_e^2) + \frac{d_j}{a_j \sigma_s^2 + b_j \sigma_e^2} \right] \tag{5}$$

where

— for $j = 1, \ldots, s_z$,

$$\begin{aligned} a_j &> 0 \text{ and is derived in the Appendix} \\ b_j &= 1 \\ c_j &= 1 \\ d_j &= \hat{v}_j^2 \text{ is a known function of } y \text{ derived in the Appendix} \end{aligned}$$

— for $j = s_z + 1$,

$$\begin{aligned} a_j &= 0 \\ b_j &= 1 \\ c_j &= \begin{cases} n_e & \text{for } \log \mathrm{RL} \text{ in (3)} \\ n_e + 2\alpha_e + 2 & \text{for the log posterior in (4)} \end{cases} \\ d_j &= \begin{cases} y^t \Gamma_c \Gamma_c^t y & \text{for } \log \mathrm{RL} \text{ in (3)} \\ y^t \Gamma_c \Gamma_c^t y + 2\beta_e & \text{for the log posterior in (4)} \end{cases} \end{aligned}$$

— for $j = s_z + 2$,

$$\begin{aligned} a_j &= 1 \\ b_j &= 0 \\ c_j &= \begin{cases} 0 & \text{for } \log \mathrm{RL} \text{ in (3)} \\ 2\alpha_s + 2 & \text{for the log posterior in (4)} \end{cases} \\ d_j &= \begin{cases} 0 & \text{for } \log \mathrm{RL} \text{ in (3)} \\ 2\beta_s & \text{for the log posterior in (4).} \end{cases} \end{aligned}$$

Our derivation proceeds from (5). We use f to denote the target function generically, either $\mathrm{RL}(\sigma_e^2, \sigma_s^2)$ or $\pi(\sigma_e^2, \sigma_s^2 \mid y)$. When the target function is $\mathrm{RL}(\sigma_e^2, \sigma_s^2)$, $c_{s_z+2} = d_{s_z+2} = 0$ and the upper limit of the sum in (5) is effectively $s_z + 1$.

It is known (e.g. Henn and Hodges 3) that $\log f(\sigma_e^2, \sigma_s^2)$ can have multiple maxima, though the incidence of multiple maxima is unknown. Multi-modal posterior distributions arise readily from conflict between the likelihood and prior; [8] explores an important such conflict in detail and [15] gives a naturally-occurring example explored further in [3]. Multiple maxima in restricted likelihoods have received far less attention and any statement about their incidence would be speculation. To our knowledge, two naturally-occurring cases have

been reported, in [16] and [12]; [3] report an artificial case and give a recipe for manufacturing examples.

As for numerical optimizers, it is known that existing general purpose algorithms for linear mixed models may fail to find all of the local maxima of $\log f(\sigma_e^2, \sigma_s^2)$, as shown by examples in [6], [3], and elsewhere. [10] examines 18 optimization functions available in R, tests them on 48 objective functions (admittedly more complicated than $\log f(\sigma_e^2, \sigma_s^2)$) and finds that even the best of them fail in over 10% of the cases. [3] examine conditions under which multiple maxima occur in posterior densities and conclude "...second maxima in posterior distributions therefore may be more common than reports in the literature would suggest." Thus, failure to find local and global maxima may be common, though with available tools it is extremely laborious to determine whether multiple maxima are present and where they are.

Our view is that it is important to find regions where f or $\log f$ is large relative to its maximum regardless of whether those regions contain local maxima. Points with large $\log f(\sigma_e^2, \sigma_s^2)$ are those that describe the data, or possibly prior information, well, at least compared to points with low $\log f(\sigma_e^2, \sigma_s^2)$. If f or $\log f$ is relatively flat and large over a region, it matters little whether the region contains small bumps that are, technically, local maxima. A good analysis should strive to find all points with large $\log f$. Therefore, the purpose of this paper is to introduce an algorithm that will, in finite time, divide the (σ_e^2, σ_s^2) plane into two parts: one where we know $\log f$ is small relative to its maximum and another where we know $\log f$ to within a pre-specified ϵ (hence the term "approximately exact"). The algorithm can also be used to find $(\hat{\sigma}_e^2, \hat{\sigma}_s^2) \equiv \text{argsup}_{\sigma_e^2, \sigma_s^2} \log f(\sigma_e^2, \sigma_s^2)$ (typically either the maximum restricted log likelihood estimate or the maximum *a posteriori* estimate), to within a prespecified tolerance without fear of missing regions of high f or $\log f$.

The technique relies on the partial derivatives of $\log f(\sigma_e^2, \sigma_s^2)$. Analysis of the partial derivatives allows us to satisfy two desiderata.

D1 For any prespecified constant $M > 0$ we can find a box B, a rectangle in the first quadrant of the (σ_s^2, σ_e^2) plane whose sides are parallel to the axes, such that

$$\text{all local maxima are in } B \text{ and } \sup_{(\sigma_e^2, \sigma_s^2) \in B^c} \log f(\sigma_e^2, \sigma_s^2) \leq \log f(\hat{\sigma}_e^2, \hat{\sigma}_s^2) - M. \tag{6}$$

In practice we will take M large enough to interpret (6) as meaning that we can restrict attention to B because values of $(\sigma_e^2, \sigma_s^2) \in B^c$ have $\log f(\sigma_e^2, \sigma_s^2)$ too low to be of further interest.

D2 For any box b with sides parallel to the axes we can quickly compute lower and upper bounds (L^b, U^b) satisfying

$$L^b \leq \inf_{(\sigma_e^2, \sigma_s^2) \in b} \log f(\sigma_e^2, \sigma_s^2) \qquad \text{and} \qquad U^b \geq \sup_{(\sigma_e^2, \sigma_s^2) \in b} \log f(\sigma_e^2, \sigma_s^2)$$

and such that $U^b - L^b \to 0$ as b shrinks. Therefore, partitioning the box B from **D1** allows us to know $\log f(\sigma_e^2, \sigma_s^2)$ everywhere in B to within a pre-

specified tolerance and also to locate argsup $\log f$ to within a pre-specified tolerance without fear of missing regions of high $\log f$.

D1 and **D2** allow us to divide the (σ_e^2, σ_s^2) plane into two parts: one where $\log f$ is at least M below its maximum and another where we know $\log f$ to within a pre-specified ϵ. The next section shows how the partial derivatives are used to satisfy **D1** and **D2**: the partial derivatives determine lines in the (σ_e^2, σ_s^2) plane; those lines determine a box B_1 containing all local maxima and a larger box $B \supset B_1$ satisfying **D1**; and those lines also determine upper and lower bounds on $\log f$ within any box b.

2. Satisfying the desiderata

2.1. Partial derivatives determine lines

The partial derivatives of $\log f(\sigma_e^2, \sigma_s^2)$ can be calculated from (5):

$$\begin{aligned}\frac{\partial \log f(\sigma_e^2, \sigma_s^2)}{\partial \sigma_s^2} &= -\frac{1}{2}\sum_{j=1}^{s_z+2}\left[\frac{a_j c_j}{a_j\sigma_s^2 + b_j\sigma_e^2} - \frac{a_j d_j}{(a_j\sigma_s^2 + b_j\sigma_e^2)^2}\right] \\ &= -\frac{1}{2}\sum_{j=1}^{s_z+2}\frac{a_j\left(a_j c_j \sigma_s^2 + b_j c_j \sigma_e^2 - d_j\right)}{(a_j\sigma_s^2 + b_j\sigma_e^2)^2} \qquad (7a)\end{aligned}$$

and

$$\begin{aligned}\frac{\partial \log f(\sigma_e^2, \sigma_s^2)}{\partial \sigma_e^2} &= -\frac{1}{2}\sum_{j=1}^{s_z+2}\left[\frac{b_j c_j}{a_j\sigma_s^2 + b_j\sigma_e^2} - \frac{b_j d_j}{(a_j\sigma_s^2 + b_j\sigma_e^2)^2}\right] \\ &= -\frac{1}{2}\sum_{j=1}^{s_z+2}\frac{b_j\left(a_j c_j \sigma_s^2 + b_j c_j \sigma_e^2 - d_j\right)}{(a_j\sigma_s^2 + b_j\sigma_e^2)^2}. \qquad (7b)\end{aligned}$$

We work with one term in (7)'s summations at a time; that is, one j at a time. For $j = 1, \ldots, s_z$, the j'th terms in (7) differ by a multiplicative constant $a_j/b_j = a_j$; they have the same sign as each other and the same sign as $(a_j c_j \sigma_s^2 + b_j c_j \sigma_e^2 - d_j) = (a_j\sigma_s^2 + \sigma_e^2 - d_j)$, which determines a line $\sigma_s^2 = d_j/a_j - \sigma_e^2/a_j$ — call it the j'th line — in the first quadrant of the (σ_s^2, σ_e^2) plane. The j'th line has positive intercept d_j/a_j and negative slope $-1/a_j$. For $j = s_z + 1$, $a_j = 0$, so (7a) $= 0$ and (7) determines a vertical line at $\sigma_e^2 = \sigma_e^{2*} \equiv d_{s_z+1}/c_{s_z+1}$. For $j = s_z + 2$, $b_j = 0$, so (7b) $= 0$ and, if the target function is (4), (7) determines a horizontal line at $\sigma_s^2 = \sigma_s^{2*} \equiv d_{s_z+2}/c_{s_z+2}$, while if the target function is (3), $c_{s_z+2} = d_{s_z+2} = 0$; the term for $j = s_z + 2$ effectively vanishes and there is no horizontal line.

For all j, both partial derivatives of the j'th term are nonnegative below or to the left of the j'th line, 0 on the line, and nonpositive above or to the right of the line, as indicated in Figure 1. The j'th term is constant on the j'th line and attains its maximum there.

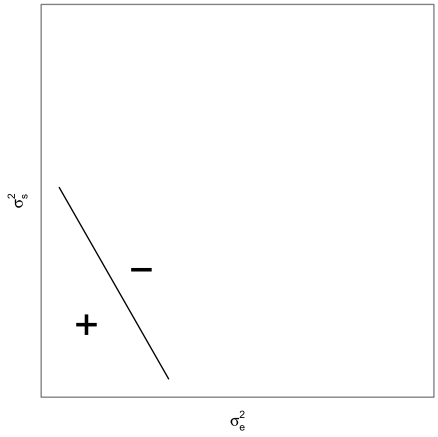


FIG 1. *For a single summand, i.e., a fixed j, in (5), the partial derivatives (7a) and (7b) are 0 on the line and nonnegative and nonpositive where indicated by "+" and "-".*

2.2. Lines determine a bounding box

Let σ_e^{2M} and σ_s^{2M} be the largest intercepts of the $s_z + 2$ lines on the σ_e^2 and σ_s^2 axes, respectively. I.e.,

$$\sigma_e^{2M} = \max\left\{\frac{d_1}{c_1}, \cdots, \frac{d_{s_z+1}}{c_{s_z+1}}\right\}$$

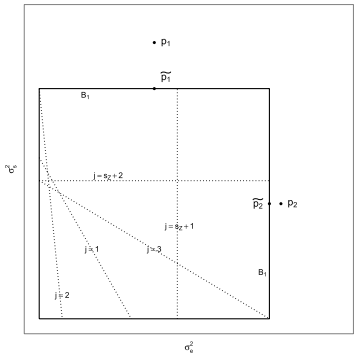
$$\sigma_s^{2M} = \begin{cases} \max\left\{\frac{d_1}{a_1}, \cdots, \frac{d_{s_z}}{a_{s_z}}\right\} & \text{if } \log f \text{ is (3)} \\ \max\left\{\frac{d_1}{a_1}, \cdots, \frac{d_{s_z}}{a_{s_z}}, \frac{d_{s_z+2}}{c_{s_z+2}}\right\} & \text{if } \log f \text{ is (4).} \end{cases}$$

Let B_1 be the box whose lower-left and upper-right corners are $(0, 0)$ and $(\sigma_e^{2M}, \sigma_s^{2M})$, respectively. Figure 2a shows B_1 and five lines — realistic data sets may have more — labelled $j = 1, 2, 3, s_z + 1, s_z + 2$.

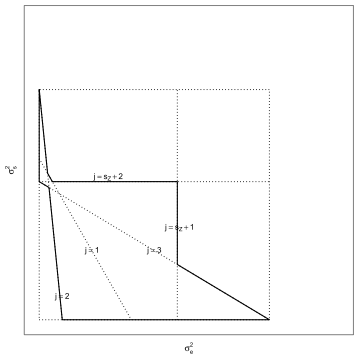
Claim. *All local maxima of* $\log f$ *lie within* B_1.

Proof. Let $p_1 = (\sigma_{e1}^2, \sigma_{s1}^2)$ be a point such that $\sigma_{s1}^2 > \sigma_s^{2M}$ and let $\widetilde{p_1} = (\sigma_{e1}^2, \sigma_s^{2M})$, as illustrated in Figure 2a. For $j = 1, \ldots, s_z, s_z + 2$, the partial derivatives (7b) are negative everywhere between p_1 and $\widetilde{p_1}$; for $j = s_z + 1$, the partial derivative is zero. Therefore, $\log f(\widetilde{p_1}) \geq \log f(p_1)$ and there can be no local maxima of $\log f$ above the line $\sigma_s^2 = \sigma_s^{2M}$. Let $p_2 = (\sigma_{e2}^2, \sigma_{s2}^2)$ be a point such that $\sigma_{e2}^2 > \sigma_e^{2M}$ and let $\widetilde{p_2} = (\sigma_e^{2M}, \sigma_{s2}^2)$. For $j = 1, \ldots, s_z, s_z+1$, the partial derivatives (7a) are negative everywhere between p_2 and $\widetilde{p_2}$; for $j = s_z+2$, the partial derivative is zero. Therefore, $\log f(\widetilde{p_2}) \geq \log f(p_2)$ and there can be no local maxima of $\log f$ to the right of the line $\sigma_e^2 = \sigma_e^{2M}$. □

By the claim, argsup $\log f$ must lie on or inside B_1, so B_1 could be passed to an optimizer such as R's optim or nlminb with potentially better results than using



(a) A rectangular region B_1 containing all maxima.



(b) A smaller region containing all maxima.

FIG 2. *all local and global maxima lie in the regions bounded by the solid dark line.*

those functions without bounds. However, even with known bounds, general purpose optimizers may still miss argsup log f and they emphasize single points of highest local log f while possibly ignoring large regions where log f is nearly as high [4, 6, Section 18.1.1].

We will next find a box $B \supset B_1$ satisfying desideratum **D1**. First, though, we pause to note that the region outlined in bold in Figure 2b is a subset of B_1 that must also contain argsup log f by the same reasoning used to prove the previous claim. But it is not rectangular, hence less convenient than B_1, so we don't pursue it further.

Claim. *For any positive number M, there exist positive numbers $\widetilde{\sigma_e^2} > \sigma_e^{2M}$ and $\widetilde{\sigma_s^2} > \sigma_s^{2M}$ that determine a box $B \supset B_1$ whose lower-left and upper-right corners are $(0,0)$ and $(\widetilde{\sigma_e^2}, \widetilde{\sigma_s^2})$, respectively, such that*

$$\sup_{B^c} \log f(\sigma_e^2, \sigma_s^2) \leq \log f(\hat{\sigma}_e^2, \hat{\sigma}_s^2) - M.$$

I.e., **D1** is satisfied.

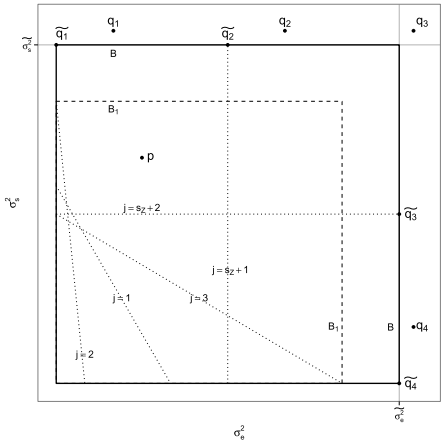


FIG 3. *Box B satisfies (6).*

Proof. Let p be an arbitrary point inside B_1, as illustrated in Figure 3, and define $L \equiv \log f(p)$; we use L as a lower bound on $\log f(\hat{\sigma}_e^2, \hat{\sigma}_s^2)$. Let $\widetilde{q_1} = (0, \widetilde{\sigma_s^2})$, the intercept of B and the σ_s^2 axis; let $\widetilde{q_2} = (\sigma_e^{2*}, \widetilde{\sigma_s^2})$; let $\widetilde{q_3} = (\widetilde{\sigma_e^2}, \sigma_s^{2*})$; let $\widetilde{q_4} = (\widetilde{\sigma_e^2}, 0)$, the intercept of B and the σ_e^2 axis; and let $\log f_j$ denote the j'th term of (5). The proof considers in turn four regions Q_1, Q_2, Q_3, and Q_4 whose union is B^c. Implicitly, B and B^c are functions of $\widetilde{\sigma_e^2}$ and $\widetilde{\sigma_s^2}$. The proof shows that $\widetilde{\sigma_e^2}$ and $\widetilde{\sigma_s^2}$ can be chosen large enough so that **D1** is satisfied on each region.

- $Q_1 : \sigma_e^2 \leq \sigma_e^{2*}$ and $\sigma_s^2 \geq \widetilde{\sigma_s^2}$, as illustrated by q_1 in Figure 3. In Q_1, by inspection of (5) and (7),
  - for $j \neq s_z + 1$,
    $$\frac{\partial \log f_j}{\partial \sigma_e^2} \leq 0 \qquad \text{and} \qquad \frac{\partial \log f_j}{\partial \sigma_s^2} \leq 0$$
    so for any $q \in Q_1$,
    $$\log f_j(q) \leq \log f_j(\widetilde{q_1});$$
  - for $j = s_z + 1$,
    $$\frac{\partial \log f_j}{\partial \sigma_e^2} > 0 \qquad \text{and} \qquad \frac{\partial \log f_j}{\partial \sigma_s^2} = 0$$
    so for any $q \in Q_1$,
    $$\log f_j(q) \leq \log f_j(\widetilde{q_2}).$$

  Therefore,
  $$\log f(\hat{\sigma}_e^2, \hat{\sigma}_s^2) - \log f(q) \geq L - \sum_{\substack{j=1 \\ j \neq s_z+1}}^{s_z+2} \log f_j(\widetilde{q_1}) - \log f_{s_z+1}(\widetilde{q_2}). \qquad (8)$$

The r.h.s. is larger than M if

$$\sum_{\substack{j=1 \\ j\neq s_z+1}}^{s_z+2} \log f_j(\widetilde{q_1}) < L - M - \log f_{s_z+1}(\widetilde{q_2})$$

$$= L - M + \frac{1}{2}\left[c_{s_z+1}\log\sigma_e^{2*} + \frac{d_{s_z+1}}{\sigma_e^{2*}}\right]. \tag{9}$$

Examination of (5) shows that for any fixed σ_e^2, in particular for $\sigma_e^2 = 0$, and for $j \neq s_z + 1$,
$\lim_{\sigma_s^2\to\infty} \log f_j(\sigma_e^2, \sigma_s^2) = -\infty$. Thus $\widetilde{\sigma_s^2}$ can be chosen large enough so that the summation on the l.h.s. of (9) is less than the r.h.s. of (9), and **D1** is satisfied.

- $Q_2 : \sigma_e^2 \geq \sigma_e^{2*}$ and $\sigma_s^2 \geq \widetilde{\sigma_s^2}$, as illustrated by q_2 and q_3 in Figure 3. In Q_2, for all j,

$$\frac{\partial \log f_j}{\partial \sigma_e^2} \leq 0 \qquad \text{and} \qquad \frac{\partial \log f_j}{\partial \sigma_s^2} \leq 0$$

so for any $q \in Q_2$,

$$\log f_j(q) \leq \log f_j(\widetilde{q_2}).$$

Therefore, $\log f(\hat{\sigma}_e^2, \hat{\sigma}_s^2) - \log f(q) \geq L - \log f(\widetilde{q_2})$. The latter expression is greater than M iff $\log f(\widetilde{q_2}) < L - M$. But (5) shows that for any fixed σ_e^2, $\lim_{\sigma_s^2\to\infty} \log f(\sigma_e^2, \sigma_s^2) = -\infty$ and therefore $\widetilde{\sigma_s^2}$ can be chosen large enough so that **D1** is satisfied.

- $Q_3 : \sigma_e^2 \geq \widetilde{\sigma_e^2}$ and $\sigma_s^2 \geq \sigma_s^{2*}$, as illustrated by q_3. In Q_3, for all j,

$$\frac{\partial \log f_j}{\partial \sigma_e^2} \leq 0 \qquad \text{and} \qquad \frac{\partial \log f_j}{\partial \sigma_s^2} \leq 0$$

so for any $q \in Q_3$,

$$\log f_j(q) \leq \log f_j(\widetilde{q_3}).$$

Therefore, $\log f(\hat{\sigma}_e^2, \hat{\sigma}_s^2) - \log f(q) \geq L - \log f(\widetilde{q_3})$. The latter expression is greater than M iff $\log f(\widetilde{q_3}) < L - M$. But for any fixed σ_s^2, $\lim_{\sigma_e^2\to\infty} \log f(\sigma_e^2, \sigma_s^2) = -\infty$ and therefore $\widetilde{\sigma_e^2}$ can be chosen large enough so that **D1** is satisfied.

- $Q_4 : \sigma_e^2 \geq \widetilde{\sigma_e^2}$ and $\sigma_s^2 \leq \sigma_s^{2*}$, as illustrated by q_4. In Q_4,
  - for $j \neq s_z + 2$,

$$\frac{\partial \log f_j}{\partial \sigma_e^2} < 0 \qquad \text{and} \qquad \frac{\partial \log f_j}{\partial \sigma_s^2} \leq 0$$

so for any $q \in Q_4$,

$$\log f_j(q) \leq \log f_j(\widetilde{q_4});$$

– for $j = s_z + 2$,

$$\frac{\partial \log f_j}{\partial \sigma_e^2} = 0 \qquad \text{and} \qquad \frac{\partial \log f_j}{\partial \sigma_s^2} \geq 0$$

so for any $q \in Q_4$,

$$\log f_j(q) \leq \log f_j(\widetilde{q_3}).$$

Therefore,

$$\log f(\hat{\sigma}_e^2, \hat{\sigma}_s^2) - \log f(q) \geq L - \sum_{j=1}^{s_z+1} \log f_j(\widetilde{q_4}) - \log f_{s_z+2}(\widetilde{q_3})$$

which is larger than M if

$$\sum_{j=1}^{s_z+1} \log f_j(\widetilde{q_4}) < L - M - \log f_{s_z+2}(\widetilde{q_3})$$
$$= L - M + \frac{1}{2}\left[c_{s_z+2} \log \sigma_s^{2*} + \frac{d_{s_z+2}}{\sigma_s^{2*}}\right]. \qquad (10)$$

For any fixed σ_s^2, in particular for $\sigma_s^2 = 0$, and for $j \neq s_z + 2$, $\lim_{\sigma_e^2 \to \infty} \log f_j(\sigma_e^2, \sigma_s^2) = -\infty$. Thus $\widetilde{\sigma_e^2}$ can be chosen large enough so that the summation on the l.h.s. of (10) is less than the r.h.s. of (10), and **D1** is satisfied. □

2.3. Lines determine bounds within boxes

Next we consider the relationship between the $\log f_j$'s and boxes as depicted in Figure 4, which shows a small box b and three lines. For lines such as $j = 2$ that

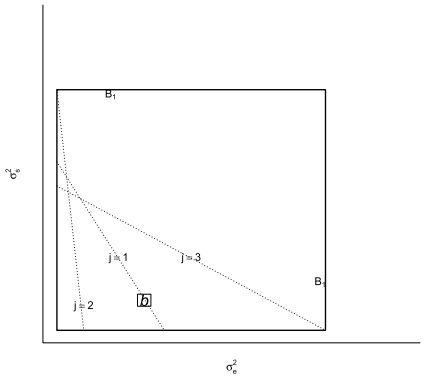


FIG 4. *For a box b, the minimum and maximum within b of* $\log f_j$ *occur at either the lower left corner, the upper right corner, or on the line.*

lie below or to the left of b, the partial derivatives (7a, 7b) are nonpositive, so the maximum and minimum of $\log f_j$ within b are attained at the lower left and upper right corners respectively. The situation is reversed for lines like $j = 3$ that lie above or to the right of b: the partial derivatives are nonnegative so the maximum and minimum are attained at the upper right and lower left corners respectively. For lines like $j = 1$ that pass through b, the minimum is attained at either the upper right or lower left corner while the maximum is attained on the line.

For any box b let $L_j^b = \inf_{p\in b} \log f_j(p)$ and $U_j^b = \sup_{p\in b} \log f_j(p)$. We have just shown that L_j^b and U_j^b are easily computable by evaluating $\log f_j$ at either two points (two corners for lines like $j = 2, 3$) or three points (two corners and one on the line for lines like $j = 1$). Armed with the L_j^b's and U_j^b's we can compute bounds on $\log f(\sigma_e^2, \sigma_s^2)$ within b:

$$L^b \equiv \sum_{j=1}^{s_z+2} L^b \le \inf_{(\sigma_s^2,\sigma_e^2)\in b} \log f(\sigma_e^2, \sigma_s^2) \le \sup_{(\sigma_s^2,\sigma_e^2)\in b} \log f(\sigma_e^2, \sigma_s^2) \le \sum_{j=1}^{s_z+2} U_j^b \equiv U^b. \quad (11)$$

Because $\log f$ is continuous, $U^b - L^b \to 0$ as b shrinks in both directions, thus satisfying desideratum **D2**.

3. An algorithm

Section 2.2 shows how to determine a box B_1 containing all local maxima of $\log f(\sigma_e^2, \sigma_s^2)$ and that we can find a bigger box $B \supseteq B_1$ satisfying **D1**. We are about to present an algorithm that uses Section 2.3 to show that any bounded box B^0 can be partitioned into finitely many smaller boxes $B_1^0, \ldots B_p^0$ such that, for each B_i^0 in the partition, we know for pre-specified constants $M, \epsilon > 0$, either (a) $\sup_{B_i^0} \log f < \sup_{B^0} \log f - M$ or (b) $\sup_{B_i^0} \log f - \inf_{B_i^0} \log f < \epsilon$. If B^0 satisfies **D1** and $B^0 \supseteq B_1$ then we have accomplished our goal of dividing the plane into one region where we know $\log f$ to be low and another where we know $\log f$ to within ϵ. One strategy would be to find B as outlined in Section 2.2 and set $B^0 \equiv B$. However, we have found that B_1 satisfies **D1** in all the examples we have tried and we therefore follow a different strategy. We set $B^0 \equiv B_1$ and run the following algorithm, during which we learn whether B_1 satisfies **D1**. If it does, we're done. If it doesn't, then we would expand B_1 by a factor of 2 in both dimensions; set B^0 to be the expanded box; rerun the algorithm; and continue to expand and rerun until we find a B^0 that does satisfy **D1**. Section 2.2 shows that such a bounded B^0 exists, and thus we are sure to find it in finite time.

Algorithm 1 sketches the basic procedure; details are in an appendix. Before giving examples, we mention some considerations in setting the tuning constants M and ϵ, which have to do with the interpretation of likelihood ratios. We

imagine a reference experiment in which a coin is chosen, either fair or two-headed, and tossed. If the coin is tossed twice and lands Heads both times, it produces a likelihood ratio of 4 in favor of the two-headed coin. That's only weak evidence, so we don't see the need to resolve likelihood functions much beyond a factor of four, or loglikelihood functions much beyond a factor of $\log(4) \approx 1.4$. In practice, we use $\epsilon = 1$ as a convenient default. If the coin were tossed 10 times and yielded 10 Heads, the likelihood ratio would be $2^{10} = 1024$. That's strong evidence, so we don't see the need to resolve $\log f$ where it is less than .001 times its maximum. $\log(1000) \approx 7$, so we use $M = 7$ as a convenient default.

Algorithm 1 Learn areas in B where $\log f$ is high to within ϵ

```
1:  function FINDF(B, ε, M)
2:      active ← list containing only B
3:      inactive ← empty list
4:      L ← −∞
5:      while length of active > 0 do
6:          for each element b in active do
7:              L^b, U^b ← get bounds on log f in b          ▷ see Section 2.3
8:          end for
9:          tmp ← max(L^b)
10:         L ← max(L, tmp)
11:         for each element b in active do
12:             if U^b − L^b < ε or U^b < L − M then
13:                 move b from active to inactive
14:             else
15:                 b_1, b_2, b_3, b_4 ← split b into 4 smaller boxes
16:                 remove b from active
17:                 add b_1, b_2, b_3, b_4 to active
18:             end if
19:         end for
20:     end while
21:     return inactive
22: end function
```

4. Examples

Section 4's examples use $\epsilon = 1$ and $M = 7$. Computations were performed on an iMac that was new in 2011, having a 2.7 GHz Intel Core i5 processor and 4GB of memory.

4.1. HMO premiums

4.1.1. Introduction to the data

Our first example is a traditional linear mixed model previously analyzed in [5], [15], [6], and [3]. [15] reported a bimodal log posterior density for (σ_e^2, σ_s^2). Quoting from [3],

> *. . . the HMO data set describes 341 HMOs [Health Maintenance Organizations] located in 45 states or similar political jurisdictions. Each jurisdiction had between 1 and 31 plans with a median of 5 plans. The data set originally was*

analysed to assess the cost of moving military retirees and dependents from a Department of Defense health plan to plans serving the US civil service. .... Specifically, the model is

$$y_{ij} = \alpha_i + \epsilon_{ij}$$
$$\alpha_i = \varrho_0 + \varrho_1 x_{1i} + \varrho_2 x_{2i} + \zeta_i,$$

where the fixed effects in α_i include an intercept, jurisdiction-average hospital expenses per admission (x_{1i}) and an indicator for plans in New England states (x_{2i}).

I.e., X is a 341×3 matrix with columns for the intercept and two fixed effects and Z is a 341×45 matrix whose columns are indicators of the 45 jurisdictions. The data are available at http://www.biostat.umn.edu/~hodges/RPLMBook/Datasets/09_HMO_premiums/Ex9.html. Because the span of Z's columns contains the span of X's, $s_z = 42$.

4.1.2. A log RL analysis

For a $\log \mathrm{RL}(\sigma_e^2, \sigma_s^2)$ analysis there are 43 lines, as shown in Figure 5. Running the algorithm on the box B_1 determined by the lines' maximum intercepts on the σ_e^2 and σ_s^2 axes results in the the output displayed in Table 1, which shows the state of the algorithm after iterations 1 through 15: the numbers of active and inactive boxes and the current value of the lower bound L on $\max \log f$. The run finished in less than 10 seconds. We see that boxes are steadily transferred from the active to the inactive list and that L increases monotonically. After 15 iterations there is a total of 9490 boxes, which are displayed in Figure 6.

Figure 6a shows the outlines of the 9490 boxes. The algorithm did not need to divide the boxes with large σ_e^2 or σ_s^2 as finely as those with small σ_e^2 and σ_s^2 because they more readily satisfy either $U^b < L - M$ or $U^b - L^b < \epsilon$, so become inactive. Figure 6b shows the same boxes on a logarithmic scale, shaded by L^b in each box. The red dot is the lower left corner of the box that maximizes

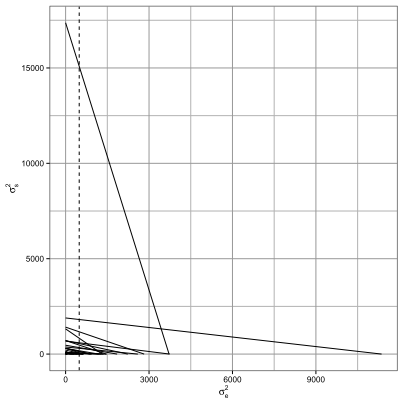


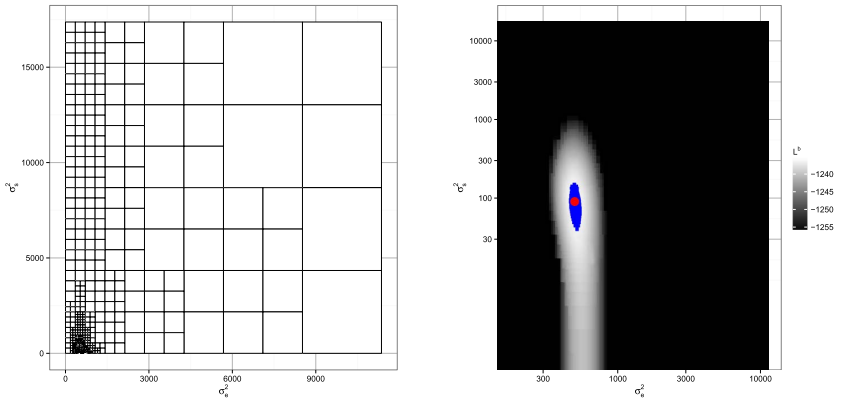
FIG 5. *The 43 lines, $j = 1, 2, \ldots, 43$, for the $\log RL$ analysis of the HMO data. The line for $j = s_z + 1 = 43$ is dashed.*

TABLE 1
The state of the algorithm after each of 15 iterations for the HMO data

Iteration	n active boxes	n inactive boxes	L
1	4	0	$-\infty$
2	16	0	-1618.84
3	40	6	-1509.24
4	72	28	-1408.12
5	144	64	-1325.99
6	68	191	-1265.01
7	116	230	-1256.35
8	144	310	-1243.51
9	340	369	-1240.05
10	920	479	-1237.87
11	2540	764	-1236.79
12	4380	2209	-1236.20
13	3524	5708	-1235.90
14	344	9146	-1235.90
15	0	9490	-1235.90

L^b. Boxes with $U^b \geq L \equiv \max L^b$ are outlined in blue; $(\hat{\sigma}_e^2, \hat{\sigma}_s^2)$ must lie within the blue region. For comparison, the standard REML analysis using R's lme function yields $\hat{\sigma}_e^2, \hat{\sigma}_s^2 \approx (495, 99)$ with 95% confidence intervals of $(421, 582)$ and $(39, 248)$.

Figure 6b depicts the same log RL as [3]'s Figures 2a (MCMC draws) and 2b (log RL contours), but their Figure 2a was produced by MCMC whereas our Figure 6b was produced by direct calculation. Their Figure 2a shows that the MCMC sampler did not sample any values of σ_s^2 less than about 10, whereas our Figure 6b and their Figure 2b show a region of high log RL extending down



(a) Locations of the boxes.

(b) Grayscale shows L^b in each box. Red dot shows $b^* \equiv \operatorname{argmax} L^b$. Boxes outlined in blue have $U^b \geq L^{b^*}$. Axes are logarithmic.

FIG 6. *The 9490 boxes produced in the* log RL *analysis of the HMO data.* maxit $= \infty; \epsilon = 1; \delta_e = \delta_s = 0;$ *and* $M = 7$.

to $\sigma_s^2 = 0$. In fact, $\log \mathrm{RL}(500, 0) \approx -1241.5$, only about 6 log units below $\log \mathrm{RL}(\hat{\sigma}_e^2, \hat{\sigma}_s^2) \approx -1235.7$. Further, about their Figure 2a, [3] say, "No change in contour shape indicative of a local maximum could be found in the ... region of $(500, 600) \times (10^{-3}, 1)$, regardless of contour resolution." I.e., they cannot be sure there are no undiscovered points with large $\log \mathrm{RL}$. In contrast, our algorithm guarantees there are no undiscovered points where $\log \mathrm{RL}$ is more than ϵ above L.

4.1.3. A Bayesian analysis

[5], [15], [6], and [3] report Bayesian analyses of the HMO data. Here we reproduce the analysis from [5] which used inverse Gamma priors for (σ_e^2, σ_s^2) with $\alpha_e = 1$; $\beta_e = 0$; $\alpha_s = 1.1$; and $\beta_s = 0.1$. (We don't defend the prior; we use it so we can compare to [5].)

For a Bayesian analysis there are 44 lines, as shown in Figure 7. Figure 7 differs from Figure 5 in that it includes a horizontal line for $j = s_z + 2$ and the position of the vertical line for $j = s_z + 1$ is slightly shifted.

Because the $\log \mathrm{RL}$ analysis in Figure 6b shows low values for $\sigma_e^2, \sigma_s^2 > 1000$, we run the Bayesian analysis on the box $B^0 = (0, 1000) \times (0, 1000)$. The algorithm finished in 22 iterations and took a little under 3 hours. Table 2 shows the output and Figure 8 shows the boxes. The Bayesian analysis in Figure 8 can be compared to the $\log \mathrm{RL}$ analysis in Figure 6. The peak of the log posterior is near $(600, .05)$, very far from the peak of the $\log \mathrm{RL}$ around $(500, 100)$. The posterior peak is due to the $\mathrm{InvGam}(1.1, 0.1)$ prior for σ_s^2, which has a mean of 1, an infinite variance, and a peak at 0.048. The region near $(500, 100)$, though having a lower posterior density, is part of a nearly flat plateau covering a large area. The MCMC draws depicted in [3] Figure 2c show that the plateau has significant posterior mass.

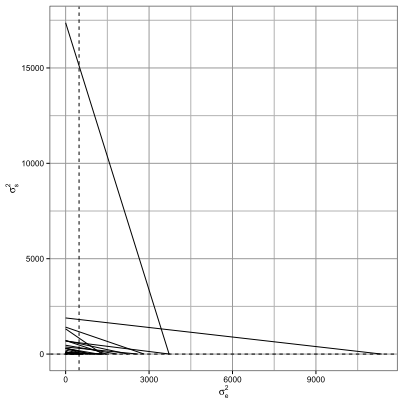
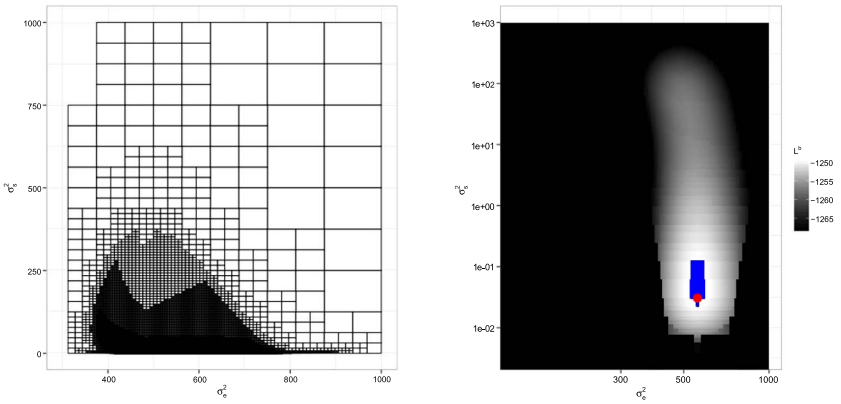


FIG 7. *The 44 lines, $j = 1, 2, \ldots, 44$, for the Bayesian analysis of the HMO data. Lines 43 and 44 are dashed.*

TABLE 2
The state of the algorithm after each of 22 iterations for the Bayesian analysis of the HMO data

Iteration	n active boxes	n inactive boxes	L
1	4	0	$-\infty$
2	16	0	-1313.42
3	52	3	-1285.05
4	132	22	-1270.76
5	264	88	-1263.98
6	628	195	-1260.47
7	1780	378	-1258.71
8	4188	1111	-1257.70
9	5676	3880	-1256.88
10	5136	8272	-1255.56
11	5528	12026	-1254.20
12	8236	15495	-1252.84
13	12488	20609	-1251.50
14	23852	27134	-1250.26
15	22740	45301	-1249.22
16	43228	57234	-1248.77
17	160208	60410	-1248.77
18	331900	137643	-1248.77
19	575196	325744	-1248.77
20	942860	665225	-1248.77
21	722412	1427482	-1248.77
22	0	2149894	-1248.77



(a) Locations of the boxes.

(b) Grayscale shows L^b in each box. Red dot shows $b^* \equiv \text{argmax}\, L^b$. Boxes outlined in blue have $U^b \geq L^{b^*}$. Axes are logarithmic.

FIG 8. *The 2,149,894 boxes in the Bayesian analysis of the HMO data.*

4.2. Global mean surface temperature

We reanalyze a data set in [6], global mean surface temperatures (GMST) from 1881 through 2005, depicted in Figure 9. As shown in [13], many splines can

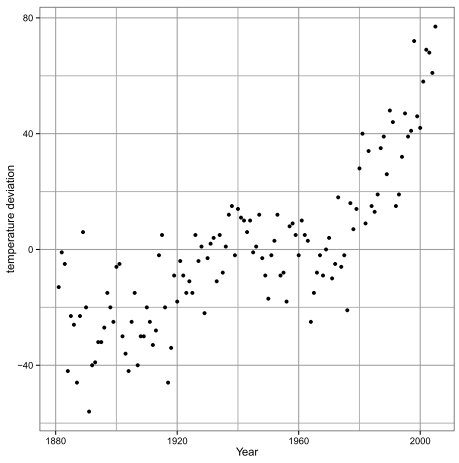


FIG 9. *Global mean surface temperature annually from 1881. The y-axis shows deviations from the overall mean in units of .01 degrees C.*

be written as linear mixed models. [6] fit a piecewise quadratic spline to the GMST data, though a piecewise cubic spline would look similar. Both splines can be formulated as linear mixed models. We follow [6]'s lead in fitting a quadratic spline with knots at 1880, 1884, 1888, ..., 2004. X has three columns: $1, \mathsf{year}, \mathsf{year}^2$. Z is 125×30: one row for each year; one column for each knot. Because we fit a quadratic spline, the entries in Z are squares. Σ_e and Σ_s are identity matrices of the appropriate dimension; see [6] for details. Following [6], we center and scale the year column of X, then compute the year^2 column of X and all the columns of Z from the transformed year, so Z becomes

$$Z = \begin{bmatrix} 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 10.97143 & 10.2521 & \dots & 0.01219048 \\ 11.15505 & 10.42971 & \dots & 0.01904762 \end{bmatrix}$$

Centering and scaling changes only the scale on which σ_s^2 is measured; we do it to more easily compare our result to [6]'s.

The column space of Z shares no dimensions with the column space of X so $s_z = 30$ and, for our log RL analysis, there are 31 lines in all, as shown in Figure 10. As usual, we use $B^0 = B_1$, the box determined by the largest intercepts of the 31 lines on the σ_e^2 and σ_s^2 axes. After 25 iterations and about 35 minutes of computing time, all boxes became inactive. Output is in Table 3; boxes are displayed in Figure 11.

The figure shows that the algorithm needed to divide boxes near the axes more finely than boxes away from the axes and that high log RL is found near $(200, 1000)$. The figure agrees with Figure 15.3 in [6].

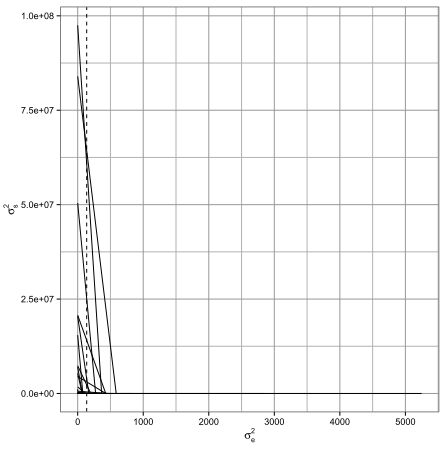


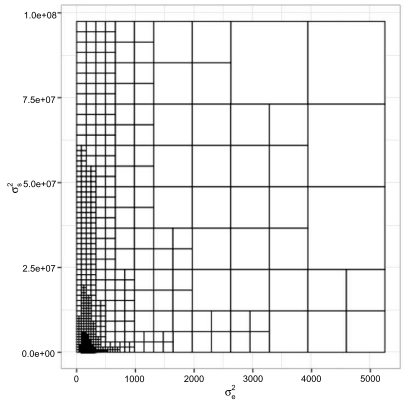
FIG 10. *The 31 lines for the* log *RL analysis of global mean surface temperatures.*

TABLE 3
The state of the algorithm after each of 25 iterations for the GMST data

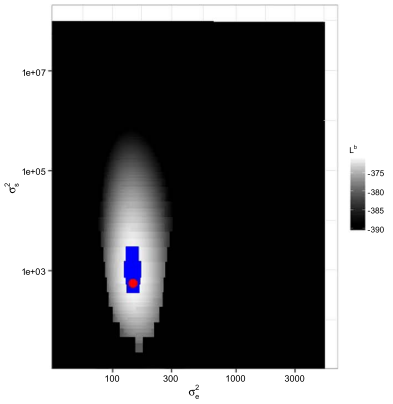
Iteration	n active boxes	n inactive boxes	L
1	4	0	$-\infty$
2	16	0	-560.82
3	48	4	-519.98
4	84	31	-480.58
5	152	77	-446.67
6	184	183	-419.47
7	176	323	-406.31
8	164	458	-395.36
9	284	551	-390.24
10	496	711	-385.84
11	984	961	-382.45
12	1792	1497	-379.65
13	2880	2569	-377.36
14	4668	4282	-375.37
15	7312	7122	-373.66
16	10336	11850	-372.29
17	10108	19659	-371.21
18	18596	25118	-370.82
19	62488	28092	-370.82
20	124240	59520	-370.70
21	287596	111861	-370.70
22	494292	275884	-370.70
23	646808	608474	-370.70
24	291944	1182296	-370.70
25	0	1474240	-370.70

5. Discussion

This paper explains and illustrates an algorithm that facilitates computation for linear mixed models with two variances. The algorithm finds all regions where either the restricted likelihood function or the joint posterior density of the



(a) Locations of boxes



(b) Grayscale shows L^b in each box. Red dot shows $b^* \equiv \text{argmax}\, L^b$. Boxes outlined in blue have $U^b \geq L^{b^*}$. Axes are logarithmic.

FIG 11. *The 1,474,240 boxes from the analysis of global mean surface temperature.*

variances is high and can evaluate the function there to arbitrary accuracy. A natural question to ask is *What about linear mixed models with more than two variances?* A partial answer is given by [6] who shows that some models with more than two variances can be re-expressed similarly to (3) but others can't. More complex models that can be re-expressed this way include, but are probably not limited to, models displaying general balance that are also orthogonal designs [all balanced ANOVAs plus other models; 7], models that are separable in a specific sense [6, Section 17.1.5], and miscellaneous other models [6, Section 17.1.5], e.g., a spatial model including random effects for heterogeneity and spatial clustering (an improper conditional autoregressive effect). We have not explored whether the re-expressible models can be analyzed by our algorithm; that's one direction for future work.

Another is to see whether the algorithm can be used to advantage even in non-re-expressible models. If a model has, say, three variances and is now analyzed by, say, MCMC, we can create an MCMC chain that alternates between draws of (σ_e^2, σ_s^2) and draws of the other variance. With the aid of our algorithm we may be able to draw more accurately from $[\sigma_e^2, \sigma_s^2 \,|\, \sigma^2_{\text{other}}]$. More generally, the conditional distribution of (σ_e^2, σ_s^2) given other parameters can now be analyzed more accurately than in the past. We have yet to explore how to exploit that accuracy. A third direction is the posterior $\pi(\sigma_e^2, \sigma_s^2 \,|\, y)$. We can identify a region B^c where the posterior density is low relative to its maximum and it would be of at least mild interest to find an upper bound for the posterior mass of B^c.

As written, our algorithm moves a box b to the inactive list if (a) $U^b < L - M$ or (b) $U^b - L^b < \epsilon$. But one could construct more elaborate rules. One appealing example is to apply criterion (a) if $U^b \leq L - \epsilon_2$ and apply criterion (b) if

$U^b > L - \epsilon_2$. Other rules are possible, too. We don't elaborate here in order to concentrate on the main ideas.

More generally, our algorithm differs from typical optimization algorithms in that it has a different goal: learning $\log f$ to specified accuracy wherever $\log f$ is high. The algorithm works by exploiting a re-expression of $\log f$ as a sum of simpler, easily analyzed functions. But there may be many other statistically interesting functions that can be so re-expressed. For example, many likelihood functions are products of terms from conditionally independent parts of the data. Posterior densities have the same terms, plus a term from the prior. We have not yet explored whether our algorithm, and more generally the idea of learning $\log f$ to specified accuracy, is useful outside the family of linear mixed models; that's another direction for future work.

In this paper we have taken the point of view that it is important to find all regions where $\log f$ is large without necessarily identifying all local maxima or even the global maximum, even though that point of view is at odds with common statistical estimators that maximize the likelihood, the restricted likelihood, or the posterior density. If two local maxima are close in height it hardly matters which is slightly higher than the other. And, as we said earlier, if there is a high plateau it hardly matters whether there are little bumps on that plateau.

Appendix A: Derivation of $\{a_j\}$ and $\{\hat{v}_j\}$ in (3), (4), and (5)

Our derivation follows [6], which contains more details. There are three steps.

1. **Make the covariance matrices proportional to the identity.** If Σ_e is not the identity matrix, transform the data to $\Sigma_e^{-.5} y$. The transformed data, which we shall still call y, has covariance proportional to the identity. Similarly, if Σ_s is not the identity matrix, re-parameterize the random effects to $\Sigma_s^{-.5} u$. The re-parameterized random effects, which we shall still call u, have covariance proportional to the identity.
2. **If the column spaces of X and Z have a non-trivial intersection, transform them.** Let $s_X = \text{rank}(X)$ and $s_Z = \text{rank}(X|Z) - s_X$. Let Γ_X be an $n \times s_X$ matrix whose columns are an orthonormal basis for the column space of X. Let Γ_Z be an $n \times s_Z$ matrix such that the columns of $[\Gamma_X|\Gamma_Z]$ are an orthonormal basis for the column space of $[X|Z]$. Let Γ_c be an $n \times (n - s_X - s_Z)$ matrix such that the columns of $[\Gamma_X|\Gamma_Z|\Gamma_c]$ are an orthonormal basis for $\mathbb{R}^n$. Define the matrix

$$M = \begin{bmatrix} M_{XX} & M_{XZ} \\ 0 & M_{ZZ} \end{bmatrix}$$

   by $[X|Z] = [\Gamma_X|\Gamma_Z]M$ where M_{XX} is $s_X \times p$ and M_{XZ} is $s_Z \times q$. Γ_X and Γ_Z are transformed versions of X and Z that have non-overlapping column spaces.

3. **Re-parameterize and diagonalize.** Let M_{ZZ} have the singular value decomposition $PA^{.5}L^t$. Now the linear mixed model (1) can be written as

$$\begin{aligned} y &= \left[X|Z\right]\begin{bmatrix}\beta\\ u\end{bmatrix} + \epsilon \\ &= \left[\Gamma_X|\Gamma_Z\right] M \begin{bmatrix}\beta\\ u\end{bmatrix} + \epsilon \\ &= \left[\Gamma_X|\Gamma_Z P\right]\begin{bmatrix}\beta^*\\ v\end{bmatrix} + \epsilon \end{aligned}$$

where $\beta^* = M_{XX}\beta + M_{XZ}u$ and $v = A^{.5}L^t u$. β^* contains the re-parametrized fixed effects while v contains the re-parametrized random effects. The corresponding design matrices Γ_X and $\Gamma_Z P$ are orthogonal to each other.

Finally, the $\{a_j\}$ in (3) are the diagonal elements of A, all of which are strictly positive, and the $\{\hat{v}_j\}$ in (3) are given by $\hat{v} = (\hat{v}_1, \ldots, \hat{v}_{s_Z})^t = P^t\Gamma_Z^t y$.

Appendix B: Details of the algorithm

With **D1** and **D2** the R function findf, sketched below, will evaluate, within a box B^0, $\log f$ to arbitrary accuracy everywhere $\log f$ is large by performing the following tasks: (a) accept as input $\{a_j, b_j, c_j, d_j\}$, B^0 and some tuning constants; (b) create a list of active boxes, initially consisting of just B^0; (c) create a list of inactive boxes, initially empty; and (d) for each active box B_j, find (L^{B_j}, U^{B_j}) and determine whether B_j needs to be subdivided. These notes explain some parts of the function in more detail.

1. A dataframe lines is an input to findf. lines contains the $\{a_j, b_j, c_j, d_j\}$ from Section 2.1. Details for computing them from y, X, Z, Σ_e, and Σ_s are in the Appendix.
2. startbox, or B^0, is an input to findf. As explained at the beginning of Section 3, startbox could be B_1, B, or any other box the user chooses.
3. Constants maxit, M, ϵ, δ_e, and δ_s are inputs to findf.
   (a) Though not mentioned in the main text, maxit is the maximum number (could be ∞) of iterations of findf's loop.
   (b) We will not further analyze regions of the plane where $\log f(\sigma_e^2, \sigma_s^2) < \log f(\hat{\sigma}_e^2, \hat{\sigma}_s^2) - M$. ($M$ could be ∞.)
   (c) We will evaluate $\log f$ to within an accuracy of ϵ (could be 0) inside B^0 unless evaluation is stopped by one of the other criteria.
   (d) Though not mentioned in the main text, the algorithm can be told not to distinguish values of σ_e^2 separated by less than δ_e (could be 0) nor values of σ_s^2 separated by less than δ_s (could be 0). Separation may be specified in either absolute or relative terms. (I.e. we look at either the difference in σ_e^2 or $\log\sigma_e^2$ (or σ_s^2 or $\log\sigma_s^2$) from one side of the box to the other.)

Setting maxit $= \infty$ and $\delta_e = \delta_s = 0$, as we have done in the examples, implies that B^0 will be partitioned so that either $\log f < \log f(\hat{\sigma}_e^2, \hat{\sigma}_s^2) - M$ or $\log f$ is known to within ϵ, for every B_i^0 in the partition.

4. A box b is a list consisting of
   - upper and lower limits on σ_e^2,
   - upper and lower limits on σ_s^2,
   - upper and lower bounds on $\log f$, U^b and L^b from (11),
   - indicators for whether this box lies above (or to the right of), below (or to the left of), or straddles each of the $s_z + 2$ lines.
5. killfunc is a function, defined within findf, that determines whether a box b should be divided more finely.
6. makebox is a function that takes limits on σ_e^2, σ_s^2 and returns a box structure.
7. splitbox is a function that takes a box $b = [\sigma_{e\,\text{low}}^2, \sigma_{e\,\text{high}}^2] \times [\sigma_{s\,\text{low}}^2, \sigma_{s\,\text{high}}^2]$ as input and returns the four boxes created by dividing each side of b at its midpoint.
8. getstatus is a function that calculates whether a box is above, below, or straddles each line.
9. getbounds is a function that computes $\{L_j^b, U_j^b\}$.
10. findlines is a function that finds the lines for $j = 1, \ldots, s_z + 1$ corresponding to $\log \text{RL}$.
11. addprior is a function that adjusts the $s_z + 1$ line and adds the $s_z + 2$ line to account for a conjugate prior.

The R code used in this manuscript is shown below. It can be obtained from github by issuing the following commands in R.

```
install.packages("devtools")
library(devtools)
install_github("andrewpbray/lmmoptim")
library(lmmoptim)
```

findf: The main function

```
#' Learn RLL or Log-posterior for two variance MM.
#'
#' Learns the shape of the objective function within epsilon.
#'
#' This is the primary function that implements the branch-bound-kill
#' algorithm described in "Approximately Exact Calculations for Linear
#' Mixed Models" by Lavine & Hodges (2015). This an interative algorithm
#' that can require substantial computation time. It is recommended that
#' the user start with a conservative \code{maxit} and do additional
#' computations as necessary.
#'
#' The arguments of this function include five settings to control when
#' to stop branching (subdividing) a box:
#' \enumerate{
#'   \item when the upper and lower bounds are within \code{eps} of one another.
#'   \item when the width of \eqn{\sigma^2_e} is less than \code{delE}.
#'   \item when the width of \eqn{\sigma^2_s} is less than \code{delS}.
#'   \item when the upper bound is more than \code{M} below the highest global
#'   lower bound.
#'   \item when the number of iterations reaches \code{maxit}.
```

```
#'   }
#'
#' @param lines a dataframe that contains the constants that define the
#'   line represented by each term in the sum. Created as output from
#'   \code{\link{findlines}}.
#' @param startbox a list of boxes and their bounds, possibly the output
#'   from a previous call to \code{findf}. If left empty, will create
#'   a new startbox from a call to \code{makebox}.
#' @param eps a non-negative numeric indicating the tolerance within which you
#'   would like to learn the function. Default is 0.
#' @param delE a non-negative numeric indicating the width of \eqn{\sigma^2_e}
#'   beyond which the algorithm will stop branching boxes. Default is 0.
#' @param delS a non-negative numeric indicating the width of \eqn{\sigma^2_s}
#'   beyond which the algorithm will stop branching boxes. Default is 0.
#' @param M a non-negative numeric indicating the size of the buffer between
#'   the highest global lower bound and the upper bound of a given box past
#'   that box will now be branched further.  Default is \code{Inf}.
#' @param maxit a positive integer indicating the maximum number of iterations
#'   of the algorithm.  Default is 10.
#' @param ratio a logical indicating if \code{delE} and \code{delS} are specified as
#'   ZZQ.  Default is \code{FALSE}.
#' @param lognote a string to append to the "log.out" log file to annotate
#'   the purpose of each run of the algorithm. For use in benchmarking computation
#'   time. Default is \code{"summary"}.
#'
#' @return A list of boxes including their limits in \eqn{\sigma^2_e} and
#'   \eqn{\sigma^2_e} as well as their bounds. Running this function also results
#'   in the creation of a log file called "log.out" containing box counts at every
#'   iteration.

findf <- function(lines, startbox, eps = 0, delE = 0, delS = 0, M = Inf, maxit = 10,
                  ratio = FALSE, lognote = "summary") {

    start_time <- Sys.time()

    if (missing(lines)) {
        print("please supply lines")
        return
    }
    if (missing(startbox)) {
        startbox <- makebox(lines = lines,
            lims.sigsqs = c(0, max(lines$int.sigsqs[is.finite(lines$int.sigsqs)])),
            lims.sigsqe = c(0, max(lines$int.sigsqe[is.finite(lines$int.sigsqe)])),
            status = rep("straddle", nrow(lines)))
    }
    inactive <- list()
    ninact <- 0
    # check whether startbox is a box or a list of boxes
    active <- ifelse(length(startbox) == 4 &&
        identical(names(startbox), c("lims.sigsqe", "lims.sigsqs", "status", "bounds")),
        list(startbox), startbox)
    nact <- length(active)

    lowbound <- -Inf
    iter <- 0

    # conditions under which a box becomes inactive
    killfunc <- function(box, lb, M, eps, delE, delS, ratio) {
        # lb is a lower bound on max f; it changes at each iteration. M, eps,
        # delE, delS stay constant throughout the iterations.
        cond.low <- box$bounds[2] < lb - M
        cond.eps <- diff(box$bounds) < eps
        cond.E <- ifelse(ratio, diff(log(box$lims.sigsqe)) < delE,
                                diff(box$lims.sigsqe) < delE)
        cond.S <- ifelse(ratio, diff(log(box$lims.sigsqs)) < delS,
                                diff(box$lims.sigsqs) < delS)
        return(cond.low || cond.eps || cond.E || cond.S)
    }
```

```
    while (nact > 0 && iter < maxit) {
        # Find the lower bound of each box and the maximum of the lower bounds.
        # For each active box, either make it inactive or divide it.
        low.act <- max(vapply(X = active, FUN = function(box) {
            box$bounds[1]
          }, FUN.VALUE = 0.1))
        lowbound <- max(lowbound, low.act)
        kill <- vapply(X = active, FUN = killfunc, FUN.VALUE = TRUE, lb = lowbound,
            M = M, eps = eps, delE - delE, delS = delS, ratio = ratio)
        nkill <- sum(kill)
        if (nkill > 0) {
            inactive[(ninact + 1):(ninact + nkill)] <- active[kill]
        }
        ninact <- length(inactive)
        kids <- list()
        nkids <- 0
        for (i in which(!kill)) {
            kids[(nkids + 1):(nkids + 4)] <-
                 splitbox(active[[i]], lines)  # boxes are split into 4 parts
            nkids <- nkids + 4
        }
        active <- kids
        nact <- length(active)

        iter <- iter + 1
        write(c("iteration", iter, "nact", nact, "ninact", ninact, "lowbound",
            lowbound), file = "log.out", ncolumns = 8, append = TRUE)
    }


    tmp <- t(vapply(X = c(active, inactive), FUN = function(box) {
        c(box$lims.sigsqs, box$lims.sigsqe, box$bounds)
    }, FUN.VALUE = c(sigsqs.lo = 0.1, sigsqs.hi = 0.1, sigsqe.lo = 0.1,
        sigsqe.hi = 0.1, rll.lower = 0.1, rll.upper = 0.1)))
    end_time <- Sys.time()
    write(paste(lognote, "runtime: ",
          round(end_time - start_time, digits = 3)),
          file = "log.out", ncolumns = 1, append = TRUE)
    return(data.frame(tmp))
}
```

makebox: make a box

```
#' Make a box with bounds
#'
#' Given the limits of the box, calculate its status relative to each line and
#' calculate the bounds on the objective function within that box.
#'
#' @param lims.sigsqe numeric vector of length 2 of the form \code{c(lower, upper)}
#'   containing the x-limits of the box.
#' @param lims.sigsqs numeric vector of length 2 of the form \code{c(lower, upper)}.
#'   containing the y-limits of the box.
#' @param status factor of length of \code{lines} denoting the location of the box
#'   relative to each line. If the box has a parent, these correspond to the status
#'   of that parent. The levels are \{above, below, straddle\}.
#' @param lines dataframe containing the constants that define the shape of each
#'   term in the sum. Output from \code{\link{findlines}}.
#'
#' @return A list containing:
#' \itemize{
#'   \item limits of the box in \eqn{\sigma^2_e} (numeric vector of length 2)
#'   \item limits of the box in \eqn{\sigma^2_s} (numeric vector of length 2)
#'   \item the status of the box relative to each line (factor of length of the
#'   number of terms in the sum)
#'   \item the bounds on the objective function (numeric vector of length 2 of
#'   the form \code{c(lower, upper)})
#'   }
```

```
makebox <- function(lims.sigsqe = NA, lims.sigsqs = NA, status = NA, lines) {

    # sanity checks
    if (missing(lims.sigsqs) || missing(lims.sigsqe))
        print("please supply lims.sigsqs and lims.sigsqe")
    if (missing(status))
        print("please supply status")
    if (missing(lines))
        print("please supply lines")

    # If the box has a parent then it inherits the parent's status.  But if the
    # parent straddles a line, we must check whether the child also straddles the
    # line.
    strad <- which(status == "straddle")
    status[strad] <- getstatus(lims.sigsqe = lims.sigsqe, lims.sigsqs = lims.sigsqs,
        lines = lines[strad, ])

    # we could get some of the bounds from the parent, but it's just as easy to
    # recalculate them
    bounds <- getbounds(lims.sigsqe = lims.sigsqe, lims.sigsqs = lims.sigsqs,
        status = status, lines = lines)

    return(list(lims.sigsqe = lims.sigsqe, lims.sigsqs = lims.sigsqs, status = status,
        bounds = colSums(bounds)))
}
```

splitbox: split a box into 4 children

```
#' Split a parent box into 4 children.
#'
#' Split a parent box into four equally sized boxes as part of the branching part
#' of the algorithm.
#'
#' @param box list containing the properties of a parent box. Created as output
#'   from \code{\link{makebox}}.
#' @param lines dataframe containing the constants that define the shape of each
#'   term in the sum. Output from \code{\link{findlines}}.
#'
#' @return a list of the four child boxes.

splitbox <- function(box, lines) {
    # split a box into four children
    NW <- with(box, makebox(lims.sigsqe = c(lims.sigsqe[1], mean(lims.sigsqe)),
        lims.sigsqs = c(mean(lims.sigsqs), lims.sigsqs[2]), status = status,
        lines = lines))
    NE <- with(box, makebox(lims.sigsqe = c(mean(lims.sigsqe), lims.sigsqe[2]),
        lims.sigsqs = c(mean(lims.sigsqs), lims.sigsqs[2]), status = status,
        lines = lines))
    SW <- with(box, makebox(lims.sigsqe = c(lims.sigsqe[1], mean(lims.sigsqe)),
        lims.sigsqs = c(lims.sigsqs[1], mean(lims.sigsqs)), status = status,
        lines = lines))
    SE <- with(box, makebox(lims.sigsqe = c(mean(lims.sigsqe), lims.sigsqe[2]),
        lims.sigsqs = c(lims.sigsqs[1], mean(lims.sigsqs)), status = status,
        lines = lines))
    return(list(NW, NE, SW, SE))
}
```

getstatus: determine where a box is, relative to the lines

```
#' Get the location of a box relative to lines.
#'
#' Computes the location of a box with given limits relative to each of
#' the lines.
#'
#' @param lims.sigsqe numeric vector of length 2 of the form \code{c(lower, upper)}
#'   containing the x-limits of the box.
```

```
#' @param lims.sigsqs numeric vector of length 2 of the form \code{c(lower, upper)}
#'   containing the y-limits of the box.
#' @param lines dataframe containing the constants that define the shape of each
#'   term in the sum. Output from \code{\link{findlines}}.
#'
#' @return A factor of length of \code{lines} denoting the location of the box
#'   relative to each line.

getstatus <- function(lims.sigsqe, lims.sigsqs, lines) {
    # Is a box above, below, or straddling the lines with these slopes and
    # intercepts?

    # value of the lines at the left side of the box
    tmp1 <- lines$int.sigsqs + lines$slope * lims.sigsqe[1]
    tmp1[is.infinite(tmp1)] <- NA

    # value of the lines at the right side of the box
    tmp2 <- lines$int.sigsqs + lines$slope * lims.sigsqe[2]
    tmp2[is.infinite(tmp2)] <- NA

    # where is the box relative to the lines?
    above <- with(lines, ifelse(slope > -Inf, lims.sigsqs[1] > tmp1,
                                              lims.sigsqe[1] > int.sigsqe))
    below <- with(lines, ifelse(slope > -Inf, lims.sigsqs[2] < tmp2,
                                              lims.sigsqe[2] < int.sigsqe))
    status <- rep("straddle", nrow(lines))
    status[above] <- "above"
    status[below] <- "below"

    return(status)
}
```

getbounds: compute $\{L_j^b, U_j^b\}$

```
#' Get bounds for a given box
#'
#' Computes the lower and upper bounds on every term in the sum of
#' the objective function within a box.
#'
#' @param lims.sigsqe numeric vector of length 2 of the form \code{c(lower, upper)}
#'   containing the x-limits of the box.
#' @param lims.sigsqs numeric vector of length 2 of the form \code{c(lower, upper)}
#'   containing the y-limits of the box.
#' @param lines dataframe containing the constants that define the shape of each
#'   term in the sum. Output from \code{\link{findlines}}.
#' @param status factor of length of \code{lines} denoting the location of the box
#'   relative to each line.
#'
#' @return A matrix with a row corresponding each term in the sum. The first column
#' contains the lower bound of that term within the box and the second column
#' contains the upper bound. Note that the \code{colSums} of this matrix yields the
#' \code{c(lower, upper)} bounds on the full objective function.

getbounds <- function(lims.sigsqe, lims.sigsqs, lines, status) {
    # small sanity check
    if (missing(lims.sigsqe) || missing(lims.sigsqs))
        print("Please supply lims.sigsqe and lims.sigsqs")
    if (missing(lines))
        print("Please supply lines")
    if (missing(status))
        print("Please supply status")
    if (length(status) != nrow(lines))
        print("length(status) != nrow(lines)")

    # evaluate each line at the upper-right corner of the box
    ur <- with(lines, a * lims.sigsqs[2] + b * lims.sigsqe[2])
    eval.ur <- with(lines, -0.5 * (multiplier.log * log(ur) + multiplier.inv/ur))
```

```
    # evaluate each line at the lower-left corner of the box
    ll <- with(lines, a * lims.sigsqs[1] + b * lims.sigsqe[1])
    eval.ll <- ifelse(ll == 0, -Inf, with(lines, -0.5 * (multiplier.log * log(ll) +
        multiplier.inv/ll)))

    bounds <- matrix(rep(eval.ur, 2), ncol = 2)
    # The next two lines of code are for lines that are not straddled.  'above'
    # means the box is above the line
    bounds[status != "above", 1] <- eval.ll[status != "above"] # lower
    bounds[status == "above", 2] <- eval.ll[status == "above"] # upper
    # now we'll take care of straddled lines
    strad <- status == "straddle"
    bounds[strad, 1] <- pmin(eval.ur[strad], eval.ll[strad]) # lower
    # for the upper bound, we can evaluate anywhere on the line, so we might as
    # well evaluate at (int.sigsqe,0)
    bounds[strad, 2] <- with(lines[strad, ], ifelse(is.na(int.sigsqe), -0.5 *
       (multiplier.log * log(int.sigsqs) + multiplier.inv/int.sigsqs), -0.5 *
       (multiplier.log * log(int.sigsqe) + multiplier.inv/int.sigsqe)))

    return(bounds)
}
```

findlines: find the $s_z + 1$ lines for $\log \mathbf{RL}$

```
#' Find the constants that define each line.
#'
#' Given X, Y, Z, find the constants {a_j, b_j, c_j, d_j} that define the shape
#' of each term in the sum.
#'
#' @param x a matrix with n rows corresponding to the fixed effects.
#' @param z a matrix with n rows corresponding to the random effects.
#' @param y a numeric vector of length n.
#' @param SigE an n x n covariance matrix for the random error.
#' @param SigS an n x n covariance matrix for the random effects.
#'
#' @return A dataframe of containing the constants that define the shape of each
#'   term in the sum.

findlines <- function(x, z, y, SigE, SigS) {
    if (!is.matrix(x))
        print("x should be a matrix")
    if (!is.matrix(z))
        print("z should be a matrix")
    if (!is.vector(y))
        print("y should be a vector")

    n <- length(y)
    nx <- ncol(x)
    nz <- ncol(z)

    if (nrow(x) != n)
        print("nrow(x) != length(y)")
    if (nrow(z) != n)
        print("nrow(z) != length(y)")

    if (!is.matrix(SigE))
        print("SigE should be a matrix")
    if (!is.matrix(SigS))
        print("SigS should be a matrix")
    if (nrow(SigE) != n || ncol(SigE) != n)
        print("SigE should be a square matrix to match length(y)")
    if (nrow(SigS) != nz || ncol(SigS) != nz)
        print("SigS should be a square matrix to match ncol(z)")

    # Is SigE the identity?  If not, make it so.
    if (!identical(SigE, diag(n))) {
        tmp <- inv.sqrt(SigE)
        y <- tmp %*% y
```

```
        x <- tmp %*% x
        z <- tmp %*% z
        SigE <- diag(n)
    }

    # Is SigS the identity?  If not, make it so.
    if (!identical(SigS, diag(nz))) {
        tmp <- sqrt.m(SigS)
        z <- z %*% tmp
        SigS <- diag(nz)
    }

    # sx, sz, Gamma_x, Gamma_z, Gamma_c # Is Gamma_c really needed?
    qrx <- qr(x, LAPACK = FALSE)  # use LINPACK to get rank(x)
    sx <- qrx$rank
    Gamx <- qr.Q(qrx)[, 1:sx]
    if (sx == 1)
        Gamx <- matrix(Gamx, ncol = 1)

    tmp <- qr.resid(qrx, z)
    qrz <- qr(tmp, LAPACK = FALSE)  # use LINPACK to get rank(x)
    sz <- qrz$rank
    Gamz <- qr.Q(qrz)[, 1:sz]
    if (sz == 1)
        Gamz <- matrix(Gamz, ncol = 1)

    M <- qr.solve(cbind(Gamx, Gamz), cbind(x, z))
    M.zz <- M[-(1:sx), -(1:ncol(x)), drop = FALSE]
    tmp <- svd(M.zz)
    a <- tmp$d^2  # follows Eq (15.1)
    v <- t(tmp$u) %*% t(Gamz) %*% y  # follows Eq (15.4)
    if (length(a) != sz)
        print("length(a) != sz")
    if (length(v) != sz)
        print("length(v) != sz")
    rss <- sum(resid(lm(y ~ cbind(Gamx, Gamz)))^2)

    lines <- data.frame(a = c(a, 0), v = c(v, sqrt(rss)), int.sigsqs = c(v^2/a,
        NA), int.sigsqe = c(v^2, rss/(n - (sx + sz))), slope = c(-1/a, -Inf),
        multiplier.log = c(rep(1, sz), n - (sx + sz)),
        multiplier.inv = c(v^2, rss), b = 1)
    return(lines)
}
```

addprior: adjust the $s_z + 1$ line and add the $s_z + 2$ line to account for a prior

```
#' Add a prior to the RLL
#'
#' Modifies the \code{lines} dataframe that is output from the
#' \code{\link{findlines}} function to add a term corresponding to the
#' prior for use in a Bayesian analysis.
#'
#' Working with the posterior density of \eqn{\sigma^2_e, \sigma^2_e} instead
#' of the RLL requires this incorporation of a prior distribution on those same
#' parameters. We work here with the inverse Gamma, which is the conjugate prior,
#' \enumerate{
#'  \item \eqn{\sigma^2_e ~ IG(\alpha_e, \beta_e)}
#'  \item \eqn{\sigma^2_s ~ IG(\alpha_s, \beta_s)}
#'  }
#' Given a dataframe output from \code{findlines},
#'
#' @param alpha_e a positive numeric. The shape parameter for the IG prior on
#'   \eqn{\sigma^2_e}.
#' @param beta_e a positive numeric. The scale parameter for the IG prior on
#'   \eqn{\sigma^2_e}.
#' @param alpha_s a positive numeric. The shape parameter for the IG prior on
```

```
#'   \eqn{\sigma^2_s}.
#' @param beta_s a positive numeric. The scale parameter for the IG prior on
#'   \eqn{\sigma^2_s}.
#'
#' @return Returns a new \code{lines} dataframe with the constants associated
#' with the prior term appended to the bottom.

addprior <- function(lines, alpha_e = 0, beta_e = 0, alpha_s = 0, beta_s = 0) {
    if ((alpha_e != 0) || (beta_e != 0)) {
        vert <- which(lines$slope == -Inf)
        lines$multiplier.log[vert] <- lines$multiplier.log[vert] + 2 * (alpha_e +
            1)
        lines$multiplier.inv[vert] <- lines$multiplier.inv[vert] + beta_e
        lines$v[vert] <- sqrt(lines$multiplier.inv[vert])
        lines$int.sigsqe[vert] <- lines$multiplier.inv[vert]/lines$multiplier.log[vert]
    }
    if ((alpha_s != 0) || (beta_s != 0)) {
        horiz <- data.frame(a = 1, multiplier.inv = 2 * beta_s, multiplier.log = 2 *
            (alpha_s + 1), v = sqrt(2 * beta_s), int.sigsqs = beta_s/(alpha_s + 1),
            int.sigsqe = NA, slope = 0, b = 0)
        lines <- rbind(lines, horiz)
    }
    return(lines)
}
```

References

[1] Browne, W., Goldstein, H., and Rasbash, J. (2001), "Multiple Membership Multiple Classification (MMMC) Models," *Statistical Modeling*, 1, 103–124.

[2] Bryk, A. S. and Raudenbush, S. W. (1992), *Hierarchical Linear Models: Applications and Data Analysis Methods*, Sage, Newbury Park.

[3] Henn, L. and Hodges, J. S. (2014), "Multiple Local Maxima in Restricted Likelihoods and Posterior Distributions for Mixed Linear Models," *International Statistical Review*, 82, 90–105.

[4] Hill, B. (1965), "Inference About Variance Components in the One-way Model," *Journal of the American Statistical Association*, 60, 806–825. MR0187319

[5] Hodges, J. H. (1998), "Some Algebra and Geometry for Hierarchical Models Applied to Diagnostics," *JRSS B*, 60, 497–536. MR1625954

[6] Hodges, J. S. (2013), *Richly Parameterized Linear Models: Additive, Time Series, and Spatial Models Using Random Effects*, CRC Press. MR3289097

[7] Houtman, A. and Speed, T. (1983), "Balance in Designed Experiments with Orthogonal Block Structure," *Annals of Statistics*, 11, 1069–1085. MR0720254

[8] Liu, J. and Hodges, J. S. (2003), "Posterior Bimodality in the Balanced One-way Random-effects Model," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65, 247–255.

[9] McCaffrey, D., Lockwood, J., Koretz, D., Louis, T., and Hamilton, L. (2004), "Models for Value-added Modeling of Teacher Effects," *Journal of Behavioral and Educational Statistics*, 29, 67–101.

[10] Mullen, K. M. (2014), "Continuous Global Optimization in R," *Journal of Statistical Software*, 60.

[11] Reich, B. and Hodges, J. (2008), "Identification of the Variance Components in the General Two-variance Linear Model," *JSPI*, 138, 1592–1604. MR2427291

[12] Reiss, P. T., Huang, L., Chen, Y.-H., Huo, L., Tarpey, T., and Mennes, M. (2014), "Massively Parallel Nonparametric Regression, with an Application to Developmental Brain Mapping," *Journal of Computational and Graphical Statistics*, 23, 232–248. MR3173769

[13] Ruppert, D., Wand, M. P., and Carroll, R. J. (2003), *Semiparametric Regression*, Cambridge University Press, Cambridge. MR1998720

[14] Verbeke, G. and Molenberghs, G. (2000), *Linear Mixed Models for Longitudinal Data*, Springer, first edn.

[15] Wakefield, J. (1998), "Comment on Some Algebra and Geometry for Hierarchical Models Applied to Diagnostics," *Journal of the Royal Statistical Society (Series B)*, 60, 497–536. MR1625954

[16] Welham, S. and Thompson, R. (2009), "A Note on Bimodality in the Log-likelihood Function for Penalized Spline Mixed Models," *Computational Statistics and Data Analysis*, 53, 920–931. MR2657058

[17] West, B. T., Welch, Kathleen, B., and Galecki, A. T. (2014), *Linear Mixed Models: A Practical Guide Using Statistical Software*, CRC Press, second edn.