Aster Models with Random Effects via Penalized Likelihood

By

Charles J. Geyer, Caroline E. Ridley, Robert G. Latta, Julie R. Etterson, and Ruth G. Shaw

Technical Report No. 692
School of Statistics
University of Minnesota
July 30, 2012

Abstract

This technical report works out details of approximate maximum likelihood estimation for aster models with random effects. Fixed and random effects are estimated by penalized log likelihood. Variance components are estimated by integrating out the random effects in the Laplace approximation of the complete data likelihood following Breslow and Clayton (1993), which can be done analytically, and maximizing the resulting approximate missing data likelihood. A further approximation treats the second derivative matrix of the cumulant function of the exponential family where it appears in the approximate missing data log likelihood as a constant (not a function of parameters). Then first and second derivatives of the approximate missing data log likelihood can be done analytically. Minus the second derivative matrix of the approximate missing data log likelihood is treated as approximate Fisher information and used to estimate standard errors.

1 Theory

Aster models (Geyer, Wagenius and Shaw, 2007; Shaw, Geyer, Wagenius, Hangelbroek, and Etterson, 2008) have attracted much recent attention. Several researchers have raised the issue of incorporating random effects in aster models, and we do so here.

1.1 Complete Data Log Likelihood

Although we are particularly interested in aster models (Geyer et al., 2007), our theory works for any exponential family model. The log likelihood can be written

$$l(\varphi) = y^T \varphi - c(\varphi),$$

where y is the canonical statistic vector, φ is the canonical parameter vector, and the cumulant function c satisfies

$$\mu(\varphi) = E_\varphi(y) = c'(\varphi) \tag{1}$$

$$W(\varphi) = \operatorname{var}_\varphi(y) = c''(\varphi) \tag{2}$$

where $c'(\varphi)$ denotes the vector of first partial derivatives and $c''(\varphi)$ denotes the matrix of second partial derivatives.

We assume a canonical affine submodel with random effects determined by

$$\varphi = a + M\alpha + Zb, \tag{3}$$

where a is a known vector, M and Z are known matrices, b is a normal random vector with mean vector zero and variance matrix D. The vector a is called the *offset vector* and the matrices M and Z are called the *model matrices* for fixed and random effects, respectively, in the terminology of the R function `glm`. (The vector a is called the *origin* in the terminology of the R function `aster`. *Design matrix* is alternative terminology for model matrix.) The matrix D is assumed to be diagonal, so the random effects are independent random variables. The diagonal components of D are called *variance components* in the classical terminology of random effects models (Searle et al., 1992). Typically the components of b are divided into blocks having the same variance (Searle et al., 1992, Section 6.1), so there are only a few variance components but many random effects, but nothing in this document uses this fact.

The unknown parameter vectors are α and ν, where ν is the vector of variance components. Thus D is a function of ν, although this is not indicated by the notation.

The "complete data log likelihood" (i. e., what the log likelihood would be if the random effect vector b were observed) is

$$l_c(\alpha, b, \nu) = l(a + M\alpha + Zb) - \tfrac{1}{2} b^T D^{-1} b - \tfrac{1}{2} \log \det(D) \tag{4}$$

in case none of the variance components are zero. We deal with the case of zero variance components in Sections 1.9, 1.10, and 1.11 below.

1.2 Missing Data Likelihood

Ideally, inference about the parameters should be based on the *missing data likelihood*, which is the complete data likelihood with random effects b integrated out

$$L_m(\alpha, \nu) = \int e^{l_c(\alpha, b, \nu)} \, db \tag{5}$$

Maximum likelihood estimates (MLE) of α and ν are the values that maximize (5). However MLE are hard to find. The integral in (5) cannot be done analytically, nor can it be done by numerical integration except in very simple cases. There does exist a large literature on doing such integrals by ordinary or Markov chain Monte Carlo (Penttinen, 1984; Thompson and Guo, 1991; Geyer and Thompson, 1992; Geyer, 1994; Shaw, Promislow, Tatar, Hughes, and Geyer, 1999; Shaw, Geyer and Shaw, 2002; Booth and Hobert, 1999; Sung and Geyer, 2007; Hunter, Handcock, Butts, Goodreau and Morris, 2008; Okabayashi and Geyer, 2011; Hummel, Hunter and Handcock, to appear), but these methods take a great deal of computing time and are difficult for ordinary users to apply. We wish to avoid that route if at all possible.

1.3 A Digression on Minimization

The theory of constrained optimization (Section 1.10 below) has a bias in favor of minimization rather than maximization. The explication below will be simpler if we switch now from maximization to minimization (minimizing minus the log likelihood) rather than switch later.

1.4 Laplace Approximation

Breslow and Clayton (1993) proposed to replace the integrand in (5) by its Laplace approximation, which is proportional to a normal probability density function so the random effects can be integrated out analytically. Let b^* denote the result of maximizing (4) considered as a function of b for fixed α and ν. Then $-\log L_m(\alpha, \nu)$ is approximated by

$$q(\alpha, \nu) = \tfrac{1}{2} \log \det[\kappa''(b^*)] + \kappa(b^*)$$

where

$$
\begin{aligned}
\kappa(b) &= -l_c(a + M\alpha + Zb) \\
\kappa'(b) &= -Z^T[y + \mu(a + M\alpha + Zb)] + D^{-1}b \\
\kappa''(b) &= Z^T W(a + M\alpha + Zb)Z + D^{-1}
\end{aligned}
$$

Hence

$$
\begin{aligned}
q(\alpha, \nu) &= -l_c(\alpha, b^*, \nu) + \tfrac{1}{2} \log\det\bigl[\kappa''(b^*)\bigr] \\
&= -l(a + M\alpha + Zb^*) + \tfrac{1}{2}(b^*)^T D^{-1} b^* + \tfrac{1}{2}\log\det(D) \\
&\qquad + \tfrac{1}{2}\log\det\bigl[Z^T W(a + M\alpha + Zb^*)Z + D^{-1}\bigr] \\
&= -l(a + M\alpha + Zb^*) + \tfrac{1}{2}(b^*)^T D^{-1} b^* \\
&\qquad + \tfrac{1}{2}\log\det\bigl[Z^T W(a + M\alpha + Zb^*)ZD + I\bigr]
\end{aligned}
\tag{6}
$$

where I denotes the identity matrix of the appropriate dimension (which must be the same as the dimension of D for the expression it appears in to make sense), where b^* is a function of α and ν and D is a function of ν, although this is not indicated by the notation, and where the last equality uses the rule sum of logs is log of product and the rule product of determinants is determinant of matrix product (Harville, 1997, Theorem 13.3.4).

Since minus the log likelihood of an exponential family is a convex function (Barndorff-Nielsen, 1978, Theorem 9.1) and the middle term on the right-hand side of (4) is a strictly convex function, it follows that (4) considered as a function of b for fixed α and ν is a strictly convex function. Moreover, this function has bounded level sets, because the first term on the right-hand side of (4) is bounded (Geyer, 2009, Theorems 4 and 6) and the second term has bounded level sets. It follows that there is unique global minimizer (Rockafellar and Wets, 2004, Theorems 1.9 and 2.6). Thus $b^*(\alpha, \nu)$ is well defined for all values of α and ν.

The key idea is to use (6) as if it were the log likelihood for the unknown parameters (α and ν), although it is only an approximation. However, this is also problematic. In doing likelihood inference using (6) we need first and second derivatives of it (to calculate Fisher information), but W is already the second derivative matrix of the cumulant function, so first derivatives of (6) would involve third derivatives of the cumulant function and second derivatives of (6) would involve fourth derivatives of the cumulant function. There are no published formulas for derivatives higher than second of the aster model cumulant function nor does software (the R package `aster`, Geyer, 2012) provide such — the derivatives do, of course, exist because every cumulant function of a regular exponential family is infinitely differentiable at every point of the canonical parameter space (Barndorff-Nielsen, 1978, Theorem 8.1) — they are just not readily available. Breslow and Clayton (1993) noted the same problem in the context of GLMM, and proceeded as if W were a constant function of its argument, so all derivatives of W were zero. This is not a bad approximation because "in asymptopia" the aster model log likelihood is exactly quadratic and W is a constant function, this being a general property of likelihoods (Geyer, in press). Hence we adopt this idea too, more because we are forced to by the difficulty of differentiating W than by our belief that we are "in asymptopia."

This leads to the following idea. Rather than basing inference on (6), we actually use

$$q(\alpha, \nu) = -l(a + M\alpha + Zb^*) + \tfrac{1}{2}(b^*)^T D^{-1} b^* + \tfrac{1}{2} \log \det\bigl[Z^T \widehat{W} Z D + I\bigr] \qquad (7)$$

where $\widehat{W}$ is a constant matrix (not a function of α and ν). This makes sense for any choice of $\widehat{W}$ that is symmetric and positive semidefinite, but we will choose $\widehat{W}$ that are close to $W(a + M\hat{\alpha} + Z\hat{b})$, where $\hat{\alpha}$ and $\hat{\nu}$ are the joint minimizers of (6) and $\hat{b} = b^*(\hat{\alpha}, \hat{\nu})$. Note that (7) is a redefinition of $q(\alpha, \nu)$. Hereafter we will no longer use the definition (6).

1.5 A Key Concept

Introduce

$$p(\alpha, b, \nu) = -l(a + M\alpha + Zb) + \tfrac{1}{2} b^T D^{-1} b + \tfrac{1}{2} \log \det\bigl[Z^T \widehat{W} Z D + I\bigr] \qquad (8)$$

where, as the left-hand side says, α, b, and ν are all free variables and, as usual, D is a function of ν, although the notation does not indicate this. Since the terms that contain b are the same in both (4) and (8), b^* can also be defined as the result of minimizing (8) considered as a function of b for fixed α and ν. Thus (7) is a profile of (8) and $(\hat{\alpha}, \hat{b}, \hat{\nu})$ is the joint minimizer of (8).

Since $p(\alpha, b, \nu)$ is a much simpler function than $q(\alpha, \nu)$, the latter having no closed form expression and requiring an optimization as part of each evaluation, it is much simpler to find $(\hat{\alpha}, \hat{b}, \hat{\nu})$ by minimizing the former rather than the latter.

1.6 A Digression on Partial Derivatives

Let $f(\alpha, b, \nu)$ be a scalar-valued function of three vector variables. We write partial derivative vectors using subscripts: $f_\alpha(\alpha, b, \nu)$ denotes the vector of partial derivatives with respect to components of α. Our convention is that we take this to be a column vector. Similarly for $f_b(\alpha, b, \nu)$. We also use this convention for partial derivatives with respect to single variables: $f_{\nu_k}(\alpha, b, \nu)$, which are, of course, scalars. We use this convention for any scalar-valued function of any number of vector variables.

We continue this convention for second partial derivatives: $f_{\alpha b}(\alpha, b, \nu)$ denotes the matrix of partial derivatives having i, j component that is the (mixed) second partial derivative of f with respect to α_i and b_j. Thus the row dimension of $f_{\alpha b}(\alpha, b, \nu)$ is the dimension of α, the column dimension is the dimension of b, and $f_{b\alpha}(\alpha, b, \nu)$ is the transpose of $f_{\alpha b}(\alpha, b, \nu)$.

This convention allows easy indication of points at which partial derivatives are evaluated. For example, $f_{\alpha b}(\alpha, b^*, \nu)$ indicates that b^* is plugged in for b in the expression for $f_{\alpha b}(\alpha, b, \nu)$.

We also use this convention of subscripts denoting partial derivatives with vector-valued functions. If $f(\alpha, b, \nu)$ is a column-vector-valued function of vector variables, then $f_\alpha(\alpha, b, \nu)$ denotes the matrix of partial derivatives having i, j component that is the partial derivative of the i-th component of $f_\alpha(\alpha, b, \nu)$ with respect to α_j. Thus the row dimension of $f_\alpha(\alpha, b, \nu)$ is the dimension of $f(\alpha, b, \nu)$ and the column dimension is the dimension of α.

1.7 First Derivatives

Start with (8). Its derivatives are

$$p_\alpha(\alpha, b, \nu) = -M^T \bigl[y - \mu(a + M\alpha + Zb)\bigr] \tag{9}$$

$$p_b(\alpha, b, \nu) = -Z^T \bigl[y - \mu(a + M\alpha + Zb)\bigr] + D^{-1} b \tag{10}$$

and

$$p_{\nu_k}(\alpha, b, \nu) = -\tfrac{1}{2} b^T D^{-1} E_k D^{-1} b + \tfrac{1}{2} \operatorname{tr}\Bigl(\bigl[Z^T \widehat{W} Z D + I \bigr]^{-1} Z^T \widehat{W} Z E_k \Bigr) \tag{11}$$

where

$$E_k = D_{\nu_k}(\nu) \tag{12}$$

is the diagonal matrix whose components are equal to one if the corresponding components of D are equal to ν_k by definition (rather than by accident when some other component of ν also has the same value) and whose components are otherwise zero. The formula for the derivative of a matrix inverse comes from Harville (1997, Chapter 15, Equation 8.15). The formula for the derivative of the log of a determinant comes from Harville (1997, Chapter 15, Equation 8.6).

The estimating equation for b^* can be written

$$p_b\bigl(\alpha, b^*, \nu\bigr) = 0 \tag{13}$$

and by the multivariate chain rule (Browder, 1996, Theorem 8.15) we have

$$\begin{aligned} q_\alpha(\alpha, \nu) &= p_\alpha(\alpha, b^*, \nu) + b^*_\alpha(\alpha, \nu)^T p_b(\alpha, b^*, \nu) \\ &= p_\alpha(\alpha, b^*, \nu) \end{aligned} \tag{14}$$

by (13), and

$$\begin{aligned} q_{\nu_k}(\alpha, \nu) &= b^*_{\nu_k}(\alpha, \nu)^T p_b(\alpha, b^*, \nu) + p_{\nu_k}(\alpha, b^*, \nu) \\ &= p_{\nu_k}(\alpha, b^*, \nu) \end{aligned} \tag{15}$$

again by (13).

1.8 Second Derivatives

By the multivariate chain rule (Browder, 1996, Theorem 8.15)

$$\begin{aligned} q_{\alpha\alpha}(\alpha, \nu) &= p_{\alpha\alpha}(\alpha, b^*, \nu) + p_{\alpha b}(\alpha, b^*, \nu) b^*_\alpha(\alpha, \nu) \\ q_{\alpha\nu}(\alpha, \nu) &= p_{\alpha\nu}(\alpha, b^*, \nu) + p_{\alpha b}(\alpha, b^*, \nu) b^*_\nu(\alpha, \nu) \\ q_{\nu\nu}(\alpha, \nu) &= p_{\nu\nu}(\alpha, b^*, \nu) + p_{\nu b}(\alpha, b^*, \nu) b^*_\nu(\alpha, \nu) \end{aligned}$$

The estimating equation (13) defines b^* implicitly. Thus derivatives of b^* are computed using the implicit function theorem (Browder, 1996, Theorem 8.29)

$$b^*_\alpha(\alpha, \nu) = -p_{bb}(\alpha, b^*, \nu)^{-1} p_{b\alpha}(\alpha, b^*, \nu) \tag{16}$$

$$b^*_\nu(\alpha, \nu) = -p_{bb}(\alpha, b^*, \nu)^{-1} p_{b\nu}(\alpha, b^*, \nu) \tag{17}$$

This theorem requires that $p_{bb}(\alpha, b^*, \nu)$ be invertible, and we shall see below that it is. Then the second derivatives above can be rewritten

$$
\begin{aligned}
q_{\alpha\alpha}(\alpha, \nu) &= p_{\alpha\alpha}(\alpha, b^*, \nu) - p_{\alpha b}(\alpha, b^*, \nu) p_{bb}(\alpha, b^*, \nu)^{-1} p_{b\alpha}(\alpha, b^*, \nu) \\
q_{\alpha\nu}(\alpha, \nu) &= p_{\alpha\nu}(\alpha, b^*, \nu) - p_{\alpha b}(\alpha, b^*, \nu) p_{bb}(\alpha, b^*, \nu)^{-1} p_{b\nu}(\alpha, b^*, \nu) \\
q_{\nu\nu}(\alpha, \nu) &= p_{\nu\nu}(\alpha, b^*, \nu) - p_{\nu b}(\alpha, b^*, \nu) p_{bb}(\alpha, b^*, \nu)^{-1} p_{b\nu}(\alpha, b^*, \nu)
\end{aligned}
$$

a particularly simple and symmetric form. If we combine all the parameters in one vector $\psi = (\alpha, \nu)$ and write $p(\psi, b)$ instead of $p(\alpha, b, \nu)$ we have

$$
q_{\psi\psi}(\psi) = p_{\psi\psi}(\psi, b^*) - p_{\psi b}\big(\psi, b^*\big) p_{bb}\big(\psi, b^*\big)^{-1} p_{b\psi}\big(\psi, b^*\big) \tag{18}
$$

This form is familiar from the conditional variance formula for normal distributions if

$$
\begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \tag{19}
$$

is the partitioned variance matrix of a partitioned normal random vector with components X_1 and X_2, then the variance matrix of the conditional distribution of X_1 given X_2 is

$$
\Sigma_{11} - \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21} \tag{20}
$$

assuming that X_2 is nondegenerate (Anderson, 2003, Theorem 2.5.1). Moreover, if the conditional distribution is degenerate, that is, if there exists a nonrandom vector v such that $\operatorname{var}(v^T X_1 \mid X_2) = 0$, then

$$
v^T X_1 = v^T \Sigma_{12} \Sigma_{22}^{-1} X_2
$$

with probability one, assuming X_1 and X_2 have mean zero (also by Anderson, 2003, Theorem 2.5.1), and the joint distribution of X_1 and X_2 is also degenerate. Thus we conclude that if the (joint) Hessian matrix of p is nonsingular, then so is the (joint) Hessian matrix of q given by (18).

The remaining work for this section is deriving second derivatives of p

$$
\begin{aligned}
p_{\alpha\alpha}(\alpha, b, \nu) &= M^T W(a + M\alpha + Zb) M \\
p_{\alpha b}(\alpha, b, \nu) &= M^T W(a + M\alpha + Zb) Z \\
p_{bb}(\alpha, b, \nu) &= Z^T W(a + M\alpha + Zb) Z + D^{-1} \\
p_{\alpha\nu_k}(\alpha, b, \nu) &= 0 \\
p_{b\nu_k}(\alpha, b, \nu) &= -D^{-1} E_k D^{-1} b \\
p_{\nu_j\nu_k}(\alpha, b, \nu) &= b^T D^{-1} E_j D^{-1} E_k D^{-1} b \\
&\qquad - \tfrac{1}{2} \operatorname{tr}\Big(\big[Z^T \widehat{W} Z D + I \big]^{-1} Z^T \widehat{W} Z E_j \\
&\qquad\qquad \big[Z^T \widehat{W} Z D + I \big]^{-1} Z^T \widehat{W} Z E_k \Big)
\end{aligned}
$$

This finishes the derivation of all the derivatives we need. Recall that in our use of the implicit function theorem we needed $p_{bb}(\alpha, b^*, \nu)$ to be invertible. From the explicit form given above we see that it is actually negative definite, because $W(a + M\alpha + Zb)$ is positive semidefinite by (2).

1.9 Zero Variance Components

When some variance components are zero, the corresponding diagonal components of D are zero, and the corresponding components of b are zero almost surely. However we deal with this situation, it must have the same effect as omitting those variance components and the corresponding random effects from the model.

Breslow and Clayton (1993, Section 2.3) suggest using the Moore-Penrose pseudoinverse (Harville, 1997, Chapter 20). Let D^+ denote the diagonal matrix whose diagonal components are the reciprocals of the diagonal components provided those are nonzero and whose diagonal components are zero otherwise. Then

$$q(\alpha, \nu) = -l(a + M\alpha + Zb^*) + \tfrac{1}{2}(b^*)^T D^+ b^* + \tfrac{1}{2} \log \det\bigl[Z^T \widehat{W} Z D + I\bigr] \qquad (21)$$

is an approximate log likelihood, but in the calculation of b^* constrained penalized maximum likelihood must be used: elements of b corresponding to zero variance components must be constrained to be zero, because (21) does not force them to be zero.

Although this proposal (Breslow and Clayton, 1993, Section 2.3) does deal with the situation where the zero variance components are somehow known, it does not adequately deal with estimating which variance components are zero. That is the subject of the following two sections.

1.10 The Theory of Constrained Optimization

1.10.1 Incorporating Constraints in the Objective Function

When zero variance components arise, optimization of (8) puts us in the realm of constrained optimization. The theory of constrained optimization (Rockafellar and Wets, 2004) has a notational bias towards minimization (Rockafellar and Wets, 2004, p. 5). Thus, as explained above (Section 1.3) we have switched from maximization to minimization.

Readers who are familiar with Karush-Kuhn-Tucker (Fletcher, 1987, Section 9.1; Nocedal and Wright, 1999, Section 12.2) theory should be warned that that theory is not adequate for the problem at hand, because the constraint set is not a closed set and so cannot be defined in terms of smooth constraint functions. Thus the need for the more general theory (Rockafellar and Wets, 2004).

The theory of constrained optimization incorporates constraints in the objective function by the simple device of defining the objective function (for a minimization problem) to have the value $+\infty$ off the constraint set (Rockafellar and Wets, 2004, Section 1A). Since no point where the objective function has the value $+\infty$ can minimize it, unless the the objective function has the value $+\infty$ everywhere, which is not the case in any application, the unconstrained minimizer of this sort of objective function is the same as the constrained minimizer.

Thus we need to impose constraints on our key function (8), requiring that each component of ν be nonnegative and when any component of ν is zero the corresponding components of b are also zero. However, the formula (8) does not make sense when components of ν are zero, so we proceed differently.

1.10.2 Lower Semicontinuous Regularization

Since all but the middle term on the right-hand side of (8) are actually defined on some neighborhood of each point of the constraint set and differentiable at each point of the constraint set, we only need to deal with the middle term. It is the sum of terms of the form b_i^2 / ν_k, where ν_k is the variance of b_i. Thus we investigate functions of this form

$$h(b, \nu) = b^2 / \nu \tag{22}$$

where, temporarily, b and ν are scalars rather than vectors (representing single components of the vectors). In case $\nu > 0$ we have derivatives

$$\begin{aligned}
h_b(b, \nu) &= 2b / \nu \\
h_\nu(b, \nu) &= - b^2 / \nu^2 \\
h_{b b}(b, \nu) &= 2 / \nu \\
h_{b \nu}(b, \nu) &= - 2 b / \nu^2 \\
h_{\nu \nu}(b, \nu) &= 2 b^2 / \nu^3
\end{aligned}$$

The Hessian matrix

$$h''(b, \nu) = \begin{pmatrix} 2 / \nu & - 2 b / \nu^2 \\ - 2 b / \nu^2 & 2 b^2 / \nu^3 \end{pmatrix}$$

has nonnegative determinants of its principal submatrices, since the diagonal components are positive and $\det\bigl(h''(b, \nu)\bigr)$ is zero. Thus the Hessian matrix is nonnegative definite (Harville, 1997, Theorem 14.9.11), which implies that h itself is convex (Rockafellar and Wets, 2004, Theorem 2.14) on the set where $\nu > 0$.

We then extend h to the whole of the constraint set (this just adds the origin to the points already considered) in two steps. First we define it to have the value $+\infty$ at all points not yet considered (those where any component of ν is nonpositive). This gives us an extended-real-valued convex function defined on all of $\mathbb{R}^2$. Second we take it to be the lower semicontinuous (LSC) regularization (Rockafellar and Wets, 2004, p. 14) of the function just defined. It is clear that

$$\liminf_{\substack{b \to \bar{b} \\ \nu \searrow 0}} h(b, \nu) = \begin{cases} 0, & \bar{b} = 0 \\ + \infty, & \text{otherwise} \end{cases}$$

Thus the LSC regularization is

$$h(b, \nu) = \begin{cases} b^2 / \nu, & \nu > 0 \\ 0, & \nu = 0 \text{ and } b = 0 \\ + \infty, & \text{otherwise} \end{cases} \tag{23}$$

The LSC regularization of a convex function is convex (Rockafellar and Wets, 2004, Proposition 2.32), so (23) defines an extended-real-valued convex function.

Note that $h(b, 0)$ considered as a function of b is minimized at $b = 0$ because that is the only point where this function is finite. Hence this does enforce the constraint that random effects corresponding to zero variance components must be zero.

Let k denote the map from indices for b to indices for ν that gives corresponding components: $\nu_{k(i)}$ is the variance of b_i. Let $\dim(b)$ denote the number of random effects. Then our objective function can be written

$$p(\alpha, b, \nu) = -l(a + M\alpha + Zb) + \tfrac{1}{2} \sum_{i=1}^{\dim(b)} h(b_i, \nu_{k(i)}) + \tfrac{1}{2} \log \det \bigl[Z^T \widehat{W} Z D + I \bigr] \tag{24}$$

where h is given by (23), provided all of the components of ν are nonnegative. The proviso is necessary because the third term on the right-hand side is not defined for all values of ν, only those such that the argument of the determinant is a positive definite matrix. Hence, we must separately define $p(\alpha, b, \nu) = +\infty$ whenever any component of ν is negative.

1.10.3 Subderivatives

In calculus we learn that the first derivative is zero at a local minimum and, therefore, to check points where the first derivative is zero. This is called Fermat's rule. This rule no longer works for nonsmooth functions, including those that incorporate constraints, such as (24). It does, of course, still work at points in the interior of the constraint set where (24) is differentiable. It does not work to check points on the boundary. There we need what Rockafellar and Wets (2004, Theorem 10.1) call *Fermat's rule, generalized:* at a local minimum the *subderivative function* is nonnegative.

For any extended-real-valued function f on $\mathbb{R}^d$, the *subderivative function*, denoted $df(x)$ is also an extended-real-valued function on $\mathbb{R}^d$ defined by

$$df(x)(\bar{w}) = \liminf_{\substack{\tau \searrow 0 \\ w \to \bar{w}}} \frac{f(x + \tau w) - f(x)}{\tau}$$

(Rockafellar and Wets, 2004, Definition 8.1). The notation on the left-hand side is read the subderivative of f at the point x in the direction $\bar{w}$. Fortunately, we do not have to use this definition to calculate subderivatives we want, because the calculus of subderivatives allows us to use simpler formulas in special cases. Firstly, there is the notion of subdifferential regularity (Rockafellar and Wets, 2004, Definition 7.25), which we can use without knowing the definition. The sum of regular functions is regular and the subderivative of a sum is the sum of the subderivatives (Rockafellar and Wets, 2004, Corollary 10.9). A smooth function is regular and the subderivative is given by

$$df(x)(w) = w^T f'(x), \tag{25}$$

where, as in Sections 1.1 and 1.4 above, $f'(x)$ denotes the gradient vector (the vector of partial derivatives) of f at the point x (Rockafellar and Wets, 2004, Exercise 8.20). Every LSC convex function is regular (Rockafellar and Wets, 2004, Example 7.27). Thus in computing subderivatives of (24) we may compute them term by term, and for the first and last terms, they are given in terms of the partial derivatives already computed by (25). For an LSC convex function

f, we have the following characterization of the subderivative (Rockafellar and Wets, 2004, Proposition 8.21). At any point x where $f(x)$ is finite, the limit

$$g(w) = \lim_{\tau \searrow 0} \frac{f(x + \tau w) - f(x)}{\tau}$$

exists and defines a sublinear function g, and then $df(x)$ is the LSC regularization of g. An extended-real-valued function g is *sublinear* if $g(0) = 0$ and

$$g(a_1 x_1 + a_2 x_2) \leq a_1 g(x_1) + a_2 g(x_2)$$

for all vectors x_1 and x_2 and positive scalars a_1 and a_2 (Rockafellar and Wets, 2004, Definition 3.18). The subderivative function of every regular LSC function is sublinear (Rockafellar and Wets, 2004, Theorem 7.26).

So let us proceed to calculate the subderivative of (23). In the interior of the constraint set, where this function is smooth, we can use the partial derivatives already calculated

$$dh(b, \nu)(u, v) = \frac{2bu}{\nu} - \frac{b^2 v}{\nu^2}$$

where the notation on the left-hand side means the subderivative of h at the point (b, ν) in the direction (u, v). On the boundary of the constraint set, which consists of the single point $(0, 0)$, we take limits. In case $v > 0$, we have

$$\lim_{\tau \searrow 0} \frac{h(\tau u, \tau v) - h(0, 0)}{\tau} = \lim_{\tau \searrow 0} \frac{\tau^2 u^2 / (\tau v)}{\tau} = \lim_{\tau \searrow 0} \frac{u^2}{v} = \frac{u^2}{v}$$

In case $v < 0$ or in case $v = 0$ and $u \neq 0$,

$$\frac{h(\tau u, \tau v) - h(0, 0)}{\tau} \tag{26}$$

is equal to $+\infty$ for all $\tau > 0$ so the limit inferior is $+\infty$. In case $v = 0$ and $u = 0$, (26) is equal to zero for all $\tau > 0$ so the limit inferior is zero. Thus we see that the limit inferior already defines an LSC function and

$$dh(0, 0)(u, v) = h(u, v).$$

1.10.4 Applying the Generalization of Fermat's Rule

The theory of constrained optimization tells us nothing we did not already know (from Fermat's rule) about smooth functions. The only way we can have $df(x)(w) = w^T f'(x) \geq 0$ for all vectors w is if $f'(x) = 0$. It is only at points where the function is nonsmooth, in the cases of interest to us, points on the boundary of the constraint set, where the theory of constrained optimization tells us things we did not know and need to know.

Even on the boundary, the conclusions of the theory about components of the state that are not on the boundary agree with what we already knew. We have

$$dp(\alpha, b, \nu)(s, u, v) = s^T p_\alpha(\alpha, b, \nu) + \text{terms not containing } s$$

and the only way this can be nonnegative for all s is if

$$p_\alpha(\alpha, b, \nu) = 0 \tag{27}$$

in which case $dp(\alpha, b, \nu)(s, u, v)$ is a constant function of s, or, what is the same thing in other words, the terms of $dp(\alpha, b, \nu)(s, u, v)$ that appear to involve s are all zero (and so do not actually involve s).

Similarly, $dp(\alpha, b, \nu)(s, u, v) \geq 0$ for all u_i and v_j such that $\nu_j > 0$ and $k(i) = j$ only if

$$\begin{aligned} p_{\nu_j}(\alpha, b, \nu) &= 0, \qquad j \text{ such that } \nu_j > 0 \\ p_{b_i}(\alpha, b, \nu) &= 0, \qquad i \text{ such that } \nu_{k(i)} > 0 \end{aligned} \tag{28}$$

in which case we conclude that $dp(\alpha, b, \nu)(s, u, v)$ is a constant function of such u_i and v_j.

Thus, assuming that we are at a point (α, b, ν) where (27) and (28) hold, and we do assume this throughout the rest of this section, $dp(\alpha, b, \nu)(s, u, v)$ actually involves only v_j and u_i such that $\nu_j = 0$ and $k(i) = j$. Define

$$\bar{p}(\alpha, b, \nu) = -l(a + M\alpha + Zb) + \tfrac{1}{2} \log \det\bigl[Z^T \widehat{W} Z D + I\bigr] \tag{29}$$

(the part of (24) consisting of the smooth terms). Then

$$\begin{aligned} dp(\alpha, b, \nu)(s, u, v) = \sum_{j \in J} \Biggl[& v_j \bar{p}_{\nu_j}(\alpha, b, \nu) \\ & + \sum_{i \in k^{-1}(j)} \biggl(u_i \bar{p}_{b_i}(\alpha, b, \nu) + h(u_i, v_j) \biggr) \Biggr] \end{aligned} \tag{30}$$

where J is the set of j such that $\nu_j = 0$, where $k^{-1}(j)$ denotes the set of i such that $k(i) = j$, and where h is defined by (23). Fermat's rule generalized says we must consider all of the terms of (30) together. We cannot consider partial derivatives, because the partial derivatives do not exist. To check that we are at a local minimum we need to show that (30) is nonnegative for all vectors u and v. Conversely, to verify that we are not at a local minimum, we need to find one pair of vectors u and v such that (30) is negative. Such a pair (u, v) we call a *descent direction*. Since Fermat's rule generalized is a necessary but not sufficient condition (like the ordinary Fermat's rule), the check that we are at a local minimum is not definitive, but the check that we are not is. If a descent direction is found, then moving in that direction away from the current value of (α, b, ν) will decrease the objective function (24).

So how do we find a descent direction? We want to minimize (30) considered as a function of u and v for fixed α, b, and ν. On further consideration, we can consider the terms of (30) for each j separately. If the minimum of

$$v_j \bar{p}_{\nu_j}(\alpha, b, \nu) + \sum_{i \in k^{-1}(j)} \biggl(u_i \bar{p}_{b_i}(\alpha, b, \nu) + h(u_i, v_j) \biggr) \tag{31}$$

over all vectors u and v is nonnegative, then the minimum is zero, because (31) has the value zero when $u = 0$ and $v = 0$. Thus we can ignore this j in calculating the descent direction.

On the other hand, if the minimum is negative, then the minimum does not occur at $v = 0$ and the minimum is actually $-\infty$ by the sublinearity of the subderivative, one consequence of sublinearity being positive homogeneity

$$df(x)(\tau w) = \tau df(x)(w), \qquad \tau \geq 0$$

which holds for any subderivative. Thus (as our terminology hints) we are only trying to find a descent *direction*, the length of the vector (u, v) does not matter, only its direction. Thus to get a finite minimum we can do a constrained minimization of (31), constraining (u, v) to lie in a ball. This is found by the well-known Karush-Kuhn-Tucker theory of constrained optimization (Fletcher, 1987, Section 9.1; Nocedal and Wright, 1999, Section 12.2) to be the minimum of the Lagrangian function

$$L(u, v) = \lambda v_j^2 + v_j \bar{p}_{\nu_j}(\alpha, b, \nu) + \sum_{i \in k^{-1}(j)} \left(\lambda u_i^2 + u_i \bar{p}_{b_i}(\alpha, b, \nu) + \frac{u_i^2}{v_j} \right) \tag{32}$$

where $\lambda > 0$ is the Lagrange multiplier, which would have to be adjusted if we were interested in constraining (u, v) to lie in a particular ball. Since we do not care about the length of (u, v) we can use any λ. We have replaced $h(u_i, v_i)$ by u_i^2 / v_j because we know that if we are finding an actual descent direction, then we will have $v_j > 0$. Now

$$\begin{aligned} L_{u_i}(u, v) &= 2\lambda u_i + \bar{p}_{b_i}(\alpha, b, \nu) + \frac{2u_i}{v_j}, \qquad i \in k^{-1}(j) \\ L_{v_j}(u, v) &= 2\lambda v_j + \bar{p}_{\nu_j}(\alpha, b, \nu) - \sum_{i \in k^{-1}(j)} \frac{u_i^2}{v_j^2} \end{aligned}$$

The minimum occurs where these are zero. Setting the first equal to zero and solving for u_i gives

$$\hat{u}_i(v_j) = -\frac{\bar{p}_{b_i}(\alpha, b, \nu)}{2(\lambda + 1/v_j)}$$

plugging this back into the second gives

$$L_{v_j}\bigl(\hat{u}(v), v\bigr) = 2\lambda v_j + \bar{p}_{\nu_j}(\alpha, b, \nu) - \frac{1}{4(\lambda v_j + 1)^2} \sum_{i \in k^{-1}(j)} \bar{p}_{b_i}(\alpha, b, \nu)^2$$

and we seek zeros of this. The right-hand is clearly an increasing function of v_j so it is negative somewhere only if it is negative when $v_j = 0$ where it has the value

$$\bar{p}_{\nu_j}(\alpha, b, \nu) - \frac{1}{4} \sum_{i \in k^{-1}(j)} \bar{p}_{b_i}(\alpha, b, \nu)^2 \tag{33}$$

So that gives us a test for a descent direction: we have a descent direction if and only if (33) is negative. Conversely, we appear to have $\hat{\nu}_j = 0$ if (33) is nonnegative.

That finishes our treatment of the theory of constrained optimization. We have to ask is all of this complication really necessary? It turns out that it is and it isn't. We can partially avoid it by a change of variables. But the cure is worse than the disease in some ways. This is presented in the following section.

1.11 Square Roots

We can avoid constrained optimization by the following change of parameter. Introduce new parameter variables by

$$\begin{aligned} \nu_j &= \sigma_j^2 \\ b &= Ac \end{aligned}$$

where A is diagonal and $A^2 = D$, so the i-th diagonal component of A is $\sigma_{k(i)}$. Then the objective function (8) becomes

$$\tilde{p}(\alpha, c, \sigma) = -l(a + M\alpha + ZAc) + \tfrac{1}{2} c^T c + \tfrac{1}{2} \log \det \bigl[Z^T \widehat{W} Z A^2 + I \bigr] \tag{34}$$

There are now no constraints and (34) is a continuous function of all variables.

The drawback is that by symmetry we must have $\tilde{p}_{\sigma_j}(\alpha, c, \sigma)$ equal to zero when $\sigma_j = 0$. Thus first derivatives become useless for checking for descent directions, and second derivative information is necessary. However, that is not the way unconstrained optimizers like the R functions `optim` and `nlminb` work. They do not expect such pathological behavior and do not deal with it correctly. If we want to use such optimizers to find local minima of (34), then we must provide starting points that have no component of ν equal to zero, and hope that the optimizer will never get any component of ν close to zero unless zero actually is a solution. But this is only a hope. The theory that guided the design of these optimizers does not provide any guarantees for this kind of objective function.

Moreover, optimizer algorithms stop when close to but not exactly at a solution, a consequence of inexactness of computer arithmetic. Thus when the optimizer stops and declares convergence with one or more components of ν close to zero, how do we know whether the true solution is exactly zero or not? We don't unless we return to the original parameterization and apply the theory of the preceding section. The question of whether the MLE of the variance components are exactly zero or not is of scientific interest, so it seems that the device of this section does not entirely avoid the theory of constrained optimization. We must change back to the original parameters and use (33) to determine whether or not we have $\nu_j = 0$.

Finally, there is another issue with this "square root" parameterization. For this new parameterization, the analogs of the second derivative formulas derived in Section 1.8 above are extraordinarily ill-behaved. The Hessian matrices are badly conditioned and sometimes turn out to be not positive definite when calculated by the computer's arithmetic (which is inexact) even though theory says they must be positive definite. We know this because at one point we thought that this "square root" parameterization was the answer to everything and tried to use it everywhere. Months of frustration ensued where it mostly worked, but failed on a few problems. It took us a long time to see that it is fundamentally wrong-headed. As we said above, the cure is worse than the disease.

Thus we concluded that, while we may use this "square root" parameterization to do unconstrained rather than constrained minimization, we should use it only for that. The test (33) should be used to determine whether variance components are exactly zero or not, and the formulas in Section 1.8 should be used to derive Fisher information.

1.11.1 First Derivatives

Some of R's optimization routines can use first derivative information, thus we derive first derivatives in this parameterization.

$$
\begin{aligned}
\tilde{p}_\alpha(\alpha, c, \sigma) &= -M^T [y - \mu(a + M\alpha + ZAc)] \qquad (35) \\
\tilde{p}_c(\alpha, c, \sigma) &= -AZ^T [y - \mu(a + M\alpha + ZAc)] + c \qquad (36) \\
\tilde{p}_{\sigma_j}(\alpha, c, \sigma) &= -c^T E_j Z^T [y - \mu(a + M\alpha + ZAc)] \\
&\qquad + \operatorname{tr}\Bigl(\bigl[Z^T \widehat{W} Z A^2 + I \bigr]^{-1} Z^T \widehat{W} Z A E_j \Bigr) \qquad (37)
\end{aligned}
$$

where E_j is given by (12).

1.12 Fisher Information

The observed Fisher information matrix is minus the second derivative matrix of the log likelihood. As we said above, we want to do this in the original parameterization.

Assembling stuff derived in preceding sections and introducing

$$
\begin{aligned}
\mu^* &= \mu\bigl(a + M\alpha + Zb^*(\alpha, \nu)\bigr) \\
W^* &= W\bigl(a + M\alpha + Zb^*(\alpha, \nu)\bigr) \\
H^* &= Z^T W^* Z + D^{-1} \\
\widehat{H} &= Z^T \widehat{W} Z D + I
\end{aligned}
$$

we obtain

$$
\begin{aligned}
q_{\alpha\alpha}(\alpha, \nu) &= M^T W^* M - M^T W^* Z (H^*)^{-1} Z^T W^* M \\
q_{\alpha\nu_j}(\alpha, \nu) &= M^T W^* Z (H^*)^{-1} D^{-1} E_j D^{-1} b^* \\
q_{\nu_j\nu_k}(\alpha, \nu) &= (b^*)^T D^{-1} E_j D^{-1} E_k D^{-1} b^* \\
&\qquad - \tfrac{1}{2} \operatorname{tr}\Bigl(\widehat{H}^{-1} Z^T \widehat{W} Z E_j \widehat{H}^{-1} Z^T \widehat{W} Z E_k \Bigr) \\
&\qquad - (b^*)^T D^{-1} E_j D^{-1} (H^*)^{-1} D^{-1} E_k D^{-1} b^*
\end{aligned}
$$

In all of these b^*, μ^*, W^*, and H^* are functions of α and ν even though the notation does not indicate this.

It is tempting to think expected Fisher information simplifies things because we "know" $E(y) = \mu$ and $\operatorname{var}(y) = W$, except we don't know that! What we do know is

$$
E(y \mid b) = \mu(a + M\alpha + Zb)
$$

but we don't know how to take the expectation of the right hand side (and similarly for the variance). Rather than introduce further approximations of dubious validity, it seems best to just use (approximate) observed Fisher information.

1.13 REML?

Breslow and Clayton (1993) do not maximize the approximate log likelihood (6), but make further approximations to give estimators motivated by REML (restricted maximum likelihood) estimators for linear mixed models (LMM). Breslow and Clayton (1993) concede that the argument that justifies REML estimators for LMM does not carry over to their REML-like estimators for generalized linear mixed models (GLMM). Hence these REML-like estimators have no mathematical justification. Even in LMM the widely used procedure of following REML estimates of the variance components with so-called BLUE estimates of fixed effects and BLUP estimates of random effects, which are actually only BLUE and BLUP if the variance components are assumed known rather than estimated, is obviously wrong: ignoring the fact that the variance components are estimated cannot be justified (and Breslow and Clayton say this in their discussion section). Hence REML is not justified even in LMM when fixed effects are the parameters of interest. In aster models, because components of the response vector are dependent and have distributions in different families, it is very unclear what REML-like estimators in the style of Breslow and Clayton (1993) might be. The analogy just breaks down. Hence, we do not pursue this REML analogy and stick with what we have described above.

2 Practice

Our goal is to minimize (6). We replace (6) with (7) in some steps because of our inability to differentiate (6), but our whole procedure must minimize (6).

2.1 Step 1

To get close to $(\hat{\alpha}, \hat{c}, \hat{\sigma})$ starting from far away we minimize

$$\begin{aligned} r(\sigma) = -l(a + M\tilde{\alpha} + ZA\tilde{c}) + \tfrac{1}{2}\tilde{c}^T\tilde{c} \\ + \tfrac{1}{2}\log\det\bigl[Z^T W(a + M\tilde{\alpha} + ZA\tilde{c})ZA^2 + I\bigr] \end{aligned} \tag{38}$$

where $\tilde{\alpha}$ and $\tilde{c}$ are the joint minimizers of (34) considered as a function of α and c for fixed σ. In (38), $\tilde{\alpha}$, $\tilde{c}$, and A are all functions of σ although the notation does not indicate this.

Because we cannot calculate derivatives of (38) we minimize using the R function `optim` with `method = "Nelder-Mead"`, the so-called Nelder-Mead simplex algorithm, a no-derivative method nonlinear optimization, not to be confused with the simplex algorithm for linear programming.

2.2 Step 2

Having found α, c, and σ close to the MLE values via the preceding step, we then switch to minimization of (34) for which we have the derivative formulas (35), (36), and (37). In this step we can use one of R's optimization functions that uses first derivative information: `nlm` or `nlminb` or `optim` with optional argument `method = "BFGS"` or `method = "CG"` or `method = "L-BFGS-B"`.

To define (34) we also need a $\widehat{W}$, and we take the value at the current values of α, c, and σ. Because W is typically a very large matrix ($n \times n$, where n is the number of nodes in complete aster graph, the number of nodes in the subgraph for a single individual times the number of individuals), we actually store $Z^T \widehat{W} Z$, which is only $r \times r$, where r is the number of random effects. We set

$$Z^T \widehat{W} Z = Z^T W(a + M\alpha + ZAc)Z \tag{39}$$

where α, c, and $A = A(\sigma)$ are the current values before we start minimizing $\tilde{p}(\alpha, c, \sigma)$ and this value of $Z^T \widehat{W} Z$ is fixed throughout the minimization, as is required by the definition of $\tilde{p}(\alpha, c, \sigma)$.

Having minimized $\tilde{p}(\alpha, c, \sigma)$ we are still not done, because now (39) is wrong. We held it fixed at the values of α, c, and σ we had before the minimization, and now those values have changed. Thus we should re-evaluate (39) and re-minimize, and continue doing this until convergence.

When this iteration terminates we are done with this step, and we have our point estimates $\hat{\alpha}$, $\hat{c}$, and $\hat{\sigma}$. We also have our point estimates $\hat{b}$ of the random effects on the original scale given by $A(\hat{\nu})\hat{c}$ and our point estimates $\nu_j = \sigma_j^2$ of the variance components.

2.3 Step 3

Having converted back to the original parameters, if any of the ν_j are close to zero we use the check (33) to determine whether or not they are exactly zero.

3 R Package Aster

We use the R statistical computing environment (R Development Core Team, 2012) in our analysis. It is free software and can be obtained from `http://cran.r-project.org`. Precompiled binaries are available for Windows, Macintosh, and popular Linux distributions. We use the contributed package `aster` (Geyer, 2012). If R has been installed, but this package has not yet been installed, do

```
install.packages("aster")
```

from the R command line (or do the equivalent using the GUI menus if on Apple Macintosh or Microsoft Windows). This may require root or administrator privileges.

Assuming the `aster` package has been installed, we load it. If we want to call R function `trust`, which this document does, we now have to load CRAN package `trust` (this used to be loaded automatically when package `aster` was loaded, but no longer is).

```
> library(aster)
> library(trust)
```

The version of the package used to make this document is 1.1-3. The version of R used to make this document is 4.3.2.

This entire document and all of the calculations shown were made using the R command `Sweave` and hence are exactly reproducible by anyone who has R and the R noweb (RNW) file from which it was created. Both the RNW file and and the PDF document produced from it will be made available at `http://www.stat.umn.edu/geyer/aster`. For further details on the use of Sweave and R see Chapter 1 of the technical report Shaw et al. (2007) available at the same web site.

Not only can one exactly reproduce the results in the printable document, one can also modify the parameters of the simulation and get different results. Anything at all can be changed once one has the RNW file.

In particular, we set the "seed" of the random number generator

```
> set.seed(42)
```

so that every time this RNW file is run it produces the same results. Changing the argument of set.seed or removing this chunk of R code will produce different results.

4 A Digression on Aster Models and Formulas

4.1 Observed Fitness

In an unconditional aster model (and all published examples in the literature are unconditional aster models except for Example 1 of Shaw et al. (2008) and that could have also been done using an unconditional aster model) the unconditional canonical parameter vector φ has a multivariate monotone relationship with the unconditional mean value parameter vector μ (Shaw and Geyer, 2010, Appendix). The exact relationship between φ and μ is very complicated. Geyer (2010) works through a simple example, and the formulas become very messy. So the only thing one can have any intuition about is the multivariate monotone relationship. The new functions added to the aster package to do random effects aster models only do unconditional aster models.

Now what unconditional mean values does one want to establish relationships with? Generally with those that are the best surrogates of overall fitness (total reproductive success of individuals over their lifetime) in the data, that is, generally, the last fitness component observed. In the data in the example in Geyer et al. (2007), that is head count (number of compound flowers observed). One might think that the earlier components of fitness are important too, but their effect is already incorporated in the last component of fitness (you can't have flowers if you are dead, and similarly for any other component of fitness).

Thus if one has predictors only affecting "fitness" nodes of the graph (those whose sum is the best surrogate of lifetime fitness), an unconditional aster model will do the right thing by adjusting the unconditional mean values of those nodes to fit the data. In recent papers we have included an indicator variable named fit that indicates these "fitness" nodes of the graph (it is one for those nodes and zero for other nodes) that helps in the modeling and we have done so for the examples in this paper.

The model in Geyer et al. (2007) that was deemed the best one for drawing scientific conclusions (their Model 2) was fit by the formula given in the paper

```
resp ~ varb + level:(nsloc + ewloc) + hdct * pop - pop
```

but that formula can be simplified to

```
resp ~ varb + level:(nsloc + ewloc) + hdct:pop
```

which fits the same model and is easier to understand. The term hdct:pop is best not thought of as an "interaction" although that is the way the R formula mini-language describes it. What actually happens is, because `pop` is categorical with 7 levels R makes 6 dummy variables (one for each category but throws one away because all 7 together would be confounded with the intercept dummy variable) then it multiplies each of these dummy variables by `hdct` (which, recall, is zero-or-one-valued), and this does exactly what is wanted: making those dummy variables apply to "fitness" nodes only. (It was just noticed that the examples on the help pages for the R functions `aster`, `anova.aster`, and `predict.aster` had the old-style formulas. Those help pages have now been fixed.)

The only difference between the example just discussed (which now matches the example on the help page for the `aster` function) and the ones in this technical report (other than being random effects models) is that where it says `hdct` we would now say `fit`, taking that for a conventional name of a dummy (indicator) variable that indicates "fitness" nodes of the graph.

Thus the dictum: every variable for which one wants to establish a relationship with (overall) fitness should enter every formula "interacted with" `fit` (but, as explained above, "interacted with" is a bad description, hence the scare quotes). And that "interacted with" must be the colon operator (`:`) in the R formula mini-language, not the star operator (`*`), as in `pop :  fit`.

4.2 Other Fitness Components

In general it makes no scientific sense to have terms without interaction in aster model formulas, except for `varb` if that is what one is calling the factor that indicates nodes of the graph (as it does in all of the examples in this technical report and earlier aster technical reports and in the aster papers these technical reports accompanied). The reason why one should not, for example, have `pop` by itself is that it makes no sense to have one parameter for the population effect on all fitness components (survival and fecundity, or survival, flowering indicator, and number of flowers). That is why the four models tested by Geyer et al. (2007, Table 1), which can be simplified (as discussed above) to

Model 1	`resp ~ varb + level:(nsloc + ewloc)`
Model 2	`resp ~ varb + level:(nsloc + ewloc) + fit:pop`
Model 3	`resp ~ varb + level:(nsloc + ewloc) + factor(fit)*pop`
Model 4	`resp ~ varb + level:(nsloc + ewloc) + level*pop`

have some "interaction" (again with scare quotes because they should not be interpreted as interactions) with either `fit` or `level` or `varb`, all of which are indicators for certain nodes of the graph or groups of nodes of the graph. These "interactions" make certain regression coefficients only apply to certain nodes of the graph. In model 1 there are no `pop` effects. In model 2 there are `pop` effects only for fitness nodes (the head count nodes that together are the best surrogate of observed fitness). In model 3 there are `pop` effects not only for fitness node but also for non-fitness nodes. We would now say this model doesn't really make scientific sense and Geyer et al. (2007) said the same thing ("it is difficult to interpret Model 3 scientifically ...") because non-fitness nodes includes two different fitness components (survival and flowering indicator) and these should have separate parameters not the same parameters. Thus we should not use Model 3 *even though it fits best according to purely statistical criteria.* In model

4 there are pop effects for each population and each “level” (which is shorthand for component of fitness), that is, there are separate pop effects for survival, for flowering, and for head count. And that also makes scientific sense. We could also have a model 5 in which the last term in the formula is varb*pop but that would be a lot of parameters.

Whenever possible, one wants to have fit interaction with all predictors with which one wants to establish a relation with fitness. As Geyer et al. (2007) discuss, models other than Model 2 are difficult to interpret. The fact that fit:pop is statistically significant has the direct interpretation that individuals having different parental populations have different fitness. Model 4, in contrast, says that all fitness components vary with respect to parental population, not necessarily in the same direction, and the effect on overall fitness is unclear.

5 Radish Data

We use data on the invasive California wild radish (*Raphanus sativus*) described by Ridley and Ellstrand (2010) and contained in the dataset radish in the R contributed package aster. For each individual, three response variables are observed, connected by the following graphical model

$$1 \xrightarrow{\text{Ber}} y_1 \xrightarrow{\text{0-Poi}} y_2 \xrightarrow{\text{Poi}} y_3$$

y_1 being an indicator of whether any flowers were produced, y_2 being the count of the number of flowers produced, y_3 being the count of the number of fruits produced, the unconditional distribution of y_1 being Bernoulli, the conditional distribution of y_2 given y_1 being zero-truncated Poisson, and the conditional distribution of y_3 given y_2 being Poisson.

We load the data

```
> data(radish)
> names(radish)
```

```
 [1] "Site"          "Block"         "Region"        "Pop"
 [5] "varb"          "resp"          "id"            "root"
 [9] "varbFlowering" "varbFlowers"   "fit"
```

```
> levels(radish$varb)
```

```
[1] "Flowering" "Flowers"   "Fruits"
```

This is a “long format” data frame produced by the R command reshape from “wide format” data. The variable varb indicates which components of the response vector (variable resp) corresponded to which original variables in the “wide format” data (components of fitness). The variable fit is the indicator of the best surrogate of fitness in these data, which is the last node (y_3) of the graph and the "Fruits" level of varb.

Then set up the graphical model

```
> pred <- c(0,1,2)
> fam <- c(1,3,2)
> sapply(fam.default(), as.character)[fam]
```

```
[1] "bernoulli"                         "truncated.poisson(truncation = 0)"
[3] "poisson"
```

These data come from a designed experiment started with seeds collected from three large wild populations of northern, coastal California wild radish and three populations of southern, inland California wild radish. Thus we have populations nested within region.

```
> reg2pop <- split(as.character(radish$Pop), as.character(radish$Region))
> reg2pop <- lapply(reg2pop, unique)
> reg2pop

$N
[1] "HLFMNGRVST" "SEARANCH"   "STYSITE"

$S
[1] "JHNSNRCH"   "NEWSW33HMT" "WATKINSUCR"
```

Plants were grown at two experimental sites, one northern, coastal California field site located at Point Reyes National Seashore and one southern, inland site located at the University of California Riverside Agricultural Experiment Station. Blocks were nested within site.

```
> sit2blk <- split(as.character(radish$Block), as.character(radish$Site))
> sit2blk <- lapply(sit2blk, unique)
> sit2blk

$`Point Reyes`
[1] "6"  "7"  "8"  "9"  "10"

$Riverside
[1] "1" "2" "3" "4" "5"
```

The issue of main scientific interest is the interaction of region and site, which is indicative of local adaptation when the pattern of mean values shows that each population does better in its home environment than in others. Testing significance of this interaction is complicated by the nesting of populations within region and blocks within site and the desire of scientists to treat these nested factors as random effects.

The best surrogate of fitness in these data is the `Fruits` component of the response vector. Thus we form the “interaction” with the indicator of this component and all scientifically interesting predictors (Section 4 above).

5.1 A Fixed Effects Model

The fixed effects model most closely connected with the random effects model of interest is

```
> aout <- aster(resp ~ varb + fit : (Site * Region + Block + Pop),
+     pred, fam, varb, id, root, data = radish)
> options(show.signif.stars = FALSE)
> summary(aout)
```

```
Call:
aster.formula(formula = resp ~ varb + fit:(Site * Region + Block +
    Pop), pred = pred, fam = fam, varvar = varb, idvar = id,
    root = root, data = radish)

                            Estimate Std. Error  z value Pr(>|z|)
(Intercept)               -4.671e+02  1.753e+00 -266.445  < 2e-16
varbFlowers                4.740e+02  1.755e+00  270.020  < 2e-16
varbFruits                 4.652e+02  1.760e+00  264.292  < 2e-16
fit:SitePoint Reyes        5.148e-01  1.300e-02   39.587  < 2e-16
fit:RegionS                3.507e-03  9.736e-03    0.360  0.71871
fit:Block2                 3.370e-01  1.466e-02   22.990  < 2e-16
fit:Block3                 9.366e-01  1.605e-02   58.370  < 2e-16
fit:Block4                 9.611e-01  1.594e-02   60.313  < 2e-16
fit:Block5                 1.220e+00  1.501e-02   81.321  < 2e-16
fit:Block6                 2.849e-01  1.652e-02   17.246  < 2e-16
fit:Block7                 1.975e-01  1.361e-02   14.518  < 2e-16
fit:Block8                 5.508e-02  1.895e-02    2.906  0.00366
fit:Block9                 1.680e-01  1.224e-02   13.725  < 2e-16
fit:PopJHNSNRCH           -2.565e-03  6.539e-03   -0.392  0.69492
fit:PopNEWSW33HMT         -1.918e-01  9.955e-03  -19.264  < 2e-16
fit:PopSEARANCH            1.879e-01  1.155e-02   16.273  < 2e-16
fit:PopSTYSITE            -1.932e-03  9.799e-03   -0.197  0.84371
fit:SiteRiverside:RegionS  5.005e-01  1.212e-02   41.311  < 2e-16

Original predictor variables dropped (aliased)
     fit:SiteRiverside
     fit:Block10
     fit:PopWATKINSUCR
```

Note: the variable `fit` is the same as the dummy variable `varbFruits` constructed by the aster model software.

The parameter of main scientific interest is the regression coefficient named (by R) `fit:SiteRiverside:RegionS`, which is the region by site interaction. A statistically significantly nonzero value of this parameter may indicate local adaptation (more on this in Section 5.3 below). In the fit above, this parameter is indeed highly statistically significant, but that P-value does not take the random effects story into account.

5.2 Random Effects Model

The traditional way to deal with a situation like this is to treat the population effects as random (within region) and the block effects as random (within site). We now do (an approximation to) this. To specify a random-effects aster model using the R function `reaster` one supplies either two formulas (when there is only one variance component) or one formula and a list of formulas (when there is more than one variance component). The first formula specifies the fixed effects model matrix (M in the notation of Section 1), and the second formula or list of formulas specifies the random effects model matrix (Z in the notation of

Section 1). When there is a list of formulas, each formula specifies the columns of the random effects model matrix that go with one variance component. The components of the list should be named, because the names are taken to name the variance components. The first formula (for fixed effects) is just like in an ordinary aster model (or a linear or generalized linear model). The other formulas (for random effects) are somewhat different in that they (1) do not need a response (that is specified by the fixed effects formula) and (2) should not have an intercept (the way to specify this is to prefix the formula with `0 +`). Hence the following. Note that we are following the dictum of Section 4.

```
> rout <- reaster(resp ~ varb + fit : (Site * Region),
+     list(block = ~ 0 + fit : Block, pop = ~ 0 + fit : Pop),
+     pred, fam, varb, id, root, data = radish)
> summary(rout)

Call:
reaster.formula(fixed = resp ~ varb + fit:(Site * Region), random = list(block = ~0 +
    fit:Block, pop = ~0 + fit:Pop), pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = radish)


Fixed Effects:
                            Estimate Std. Error  z value Pr(>|z|)
(Intercept)               -467.24221    1.75183 -266.717   <2e-16
varbFlowers                474.13812    1.75416  270.293   <2e-16
varbFruits                 466.11027    1.76038  264.779   <2e-16
fit:SitePoint Reyes         -0.03620    0.20781   -0.174    0.862
fit:RegionS                 -0.12249    0.07892   -1.552    0.121
fit:SiteRiverside:RegionS    0.49930    0.01211   41.223   <2e-16

Square Roots of Variance Components (P-values are one-tailed):
      Estimate Std. Error z value Pr(>|z|)/2
block  0.32820    0.07358   4.461   4.09e-06
pop    0.09619    0.02992   3.214   0.000654
```

The results are somewhat different from the fixed effects analysis. One fixed effect `fit:SitePoint Reyes`, which was statistically significant in the fixed effects model, is not statistically significant in the random effects model. The main scientific conclusions, however, do not change (Section 5.3 below).

We also try out some options of the `summary.reaster` function, not because they are particularly interesting here, but just to illustrate them

```
> summary(rout, standard.deviation = FALSE)

Call:
reaster.formula(fixed = resp ~ varb + fit:(Site * Region), random = list(block = ~0 +
    fit:Block, pop = ~0 + fit:Pop), pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = radish)
```

```
Fixed Effects:
                            Estimate Std. Error  z value Pr(>|z|)
(Intercept)               -467.24221    1.75183 -266.717   <2e-16
varbFlowers                474.13812    1.75416  270.293   <2e-16
varbFruits                 466.11027    1.76038  264.779   <2e-16
fit:SitePoint Reyes         -0.03620    0.20781   -0.174    0.862
fit:RegionS                 -0.12249    0.07892   -1.552    0.121
fit:SiteRiverside:RegionS    0.49930    0.01211   41.223   <2e-16

Variance Components (P-values are one-tailed):
      Estimate Std. Error z value Pr(>|z|)/2
block 0.107716   0.048297   2.230     0.0129
pop   0.009252   0.005757   1.607     0.0540
```

Now the estimates in the `Variance Components` section of the printout are variance components $\hat{\nu}_k = \hat{\sigma}_k^2$ rather than their square roots as we had before. We do not recommend this option `standard.deviation = FALSE` because the standard errors, derived from the delta method go to zero as ν_k goes to zero, and this is wrong (it is right "in asymptopia" but for sufficiently small ν_k whatever sample size one has is too small). Thus this option seems to provide worse estimates than the default.

5.3 Does this Reanalysis Change Biological Conclusions?

In short, no. Ridley and Ellstrand (2010) did not do a random effects aster analysis because it had not yet been invented. Nevertheless their conclusions hold up. The main conclusion of interest being local adaptation, which is indicated by the statistical significance of the fixed effect for region-site interaction and the pattern of mean values for different populations and different blocks, which we do not examine (this was done by Ridley and Ellstrand, 2010).

The fact that random effects analysis and fixed effects analysis agree qualitatively on this one example does not, of course, prove that they would agree on all examples. In these data the region-site interaction is very large and almost any sensible statistical analysis would show it. When the interaction is not so large, the analysis done will make a difference.

6 Bootstrapping the Radish Data

In this section we do a parametric bootstrap to check the standard errors provided by `reaster` and `summary`. First we store the (approximate) maximum likelihood estimates

```
> names(rout)

 [1] "obj"        "fixed"      "random"     "dropped"    "sigma"
 [6] "nu"         "c"          "b"          "alpha"      "zwz"
[11] "response"   "origin"     "iterations" "counts"     "deviance"
[16] "formula"    "call"
```

```
> alpha.hat <- rout$alpha
> sigma.hat <- rout$sigma
> nu.hat <- rout$nu
> b.hat <- rout$b
> c.hat <- rout$c
> sout <- summary(rout)
> se.alpha.hat <- sout$alpha[ , "Std. Error"]
> se.sigma.hat <- sout$sigma[ , "Std. Error"]
> se.nu.hat <- sout$nu[ , "Std. Error"]
> fixed <- rout$fixed
> random <- rout$random
> modmat.tot <- cbind(fixed, Reduce(cbind, random))
> nfix <- ncol(fixed)
> nrand <- sapply(random, ncol)
> a.hat <- rep(sigma.hat, times = nrand)
```

To simulate new aster data we first need to change from unconditional canonical parameters to conditional canonical parameters (because that's what the R function `raster` requires).

```
> c.star <- rnorm(sum(nrand))
> b.star <- a.hat * c.star
> eff.star <- c(alpha.hat, b.star)
> phi.star <- as.numeric(as.vector(rout$obj$origin) + modmat.tot %*% eff.star)
> theta.star <- astertransform(phi.star, rout$obj, to.cond = "conditional",
+     to.mean = "canonical")
> y.star <- raster(theta.star, pred, fam, rout$obj$root)
> y.star <- as.vector(y.star)
```

Now we need to redo the above analysis on the new data. We can take the simulation truth as starting values.

```
> rout.star <- reaster(y.star ~ varb + fit : (Site * Region),
+     list(block = ~ 0 + fit : Block, pop = ~ 0 + fit : Pop),
+     pred, fam, varb, id, root, data = radish,
+     effects = c(alpha.hat, c.star), sigma = sigma.hat)
> sout.star <- summary(rout.star)
> print(sout.star)

Call:
reaster.formula(fixed = y.star ~ varb + fit:(Site * Region),
    random = list(block = ~0 + fit:Block, pop = ~0 + fit:Pop),
    pred = pred, fam = fam, varvar = varb, idvar = id, root = root,
    data = radish, effects = c(alpha.hat, c.star), sigma = sigma.hat)


Fixed Effects:
                            Estimate Std. Error  z value Pr(>|z|)
(Intercept)               -465.37498    3.41631 -136.222   <2e-16
varbFlowers                472.27137    3.41732  138.199   <2e-16
varbFruits                 464.39985    3.41932  135.816   <2e-16
```

```
fit:SitePoint Reyes          0.04351    0.16525    0.263    0.792
fit:RegionS                 -0.09240    0.09720   -0.951    0.342
fit:SiteRiverside:RegionS    0.50187    0.01262   39.763   <2e-16

Square Roots of Variance Components (P-values are one-tailed):
      Estimate Std. Error z value Pr(>|z|)/2
block  0.26083    0.05863   4.448   4.32e-06
pop    0.11846    0.03604   3.287   0.000507
```

Now we want to get estimates and standard errors for this fit

```
> alpha.star <- rout.star$alpha
> sigma.star <- rout.star$sigma
> nu.star <- rout.star$nu
> as.vector(alpha.star - alpha.hat)
```

```
[1]  1.867230807 -1.866751889 -1.710425953  0.079703193  0.030085659
[6]  0.002567234
```

```
> as.vector(sigma.star - sigma.hat)
```

```
[1] -0.06737231  0.02227430
```

```
> as.vector(nu.star - nu.hat)
```

```
[1] -0.039684234  0.004781172
```

```
> se.alpha.star <- sout.star$alpha[ , "Std. Error"]
> se.sigma.star <- sout.star$sigma[ , "Std. Error"]
> se.nu.star <- sout.star$nu[ , "Std. Error"]
```

So this is the analysis we bootstrap. All we need to do is put it in a loop.

The bootstrap takes a long time (see below) if done with a reasonable sample size. Here we use bootstrap sample size

```
> nboot <- 199
```

Thus we save the results and restore them here

```
> suppressWarnings(foo <- try(load("radish-boot.rda"), silent = TRUE))
> done <- (! inherits(foo, "try-error")) &&
+     identical(.Random.seed, signature$seed) &&
+     identical(nboot, signature$nboot) &&
+     identical(alpha.hat, signature$alpha) &&
+     identical(sigma.hat, signature$sigma)
> done
```

```
[1] FALSE
```

If `done` is `TRUE` then the bootstrap is already done and its results restored. Otherwise we have to do it, either because it has never been done or because one or more of the values of the R objects that (partially) determine it has changed. The reason for using both the `suppressWarnings` function and the `silent = TRUE` argument to the `try` function is that when the file we are trying to load does not exist the `load` function gives both an error and a warning.

So now we are ready to try it out.

```
> if (! done) {
+     signature <- list(seed = .Random.seed, nboot = nboot, alpha = alpha.hat,
+         sigma = sigma.hat)
+     boot.start.time <- proc.time()
+
+     alpha.star <- matrix(NaN, nboot, length(alpha.hat))
+     sigma.star <- matrix(NaN, nboot, length(sigma.hat))
+     nu.star <- matrix(NaN, nboot, length(nu.hat))
+     se.alpha.star <- alpha.star
+     se.sigma.star <- sigma.star
+     se.nu.star <- nu.star
+
+     for (iboot in 1:nboot) {
+         c.star <- rnorm(sum(nrand))
+         b.star <- a.hat * c.star
+         eff.star <- c(alpha.hat, b.star)
+         phi.star <- as.numeric(as.vector(rout$obj$origin) +
+             modmat.tot %*% eff.star)
+         theta.star <- astertransform(phi.star, rout$obj,
+             to.cond = "conditional", to.mean = "canonical")
+         y.star <- raster(theta.star, pred, fam, rout$obj$root)
+         y.star <- as.vector(y.star)
+
+         rout.star <- reaster(y.star ~ varb + fit : (Site * Region),
+             list(block = ~ 0 + fit : Block, pop = ~ 0 + fit : Pop),
+             pred, fam, varb, id, root, data = radish,
+             effects = c(alpha.hat, c.star), sigma = sigma.hat)
+         sout.star <- suppressWarnings(summary(rout.star))
+
+         alpha.star[iboot, ] <- rout.star$alpha
+         sigma.star[iboot, ] <- rout.star$sigma
+         nu.star[iboot, ] <- rout.star$nu
+         se.alpha.star[iboot, ] <- sout.star$alpha[ , "Std. Error"]
+         se.sigma.star[iboot, ] <- sout.star$sigma[ , "Std. Error"]
+         se.nu.star[iboot, ] <- sout.star$nu[ , "Std. Error"]
+     }
+
+     boot.stop.time <- proc.time()
+     save.random.seed <- .Random.seed
+     save(signature, alpha.star, sigma.star, nu.star, se.alpha.star,
+         se.sigma.star, se.nu.star,
+         boot.start.time, boot.stop.time, save.random.seed,
+         file = "radish-boot.rda")
+ } else {
+     .Random.seed <- save.random.seed
+ }
```

The bootstrap with bootstrap sample size 199 took 0 hours, 5 minutes, and 50.7 seconds.

Occasionally, the summary function fails to calculate standard errors. The following tells us how many times this happened.

```
> sum(! is.finite(se.alpha.star[ , 1]))
```

```
[1] 1
```

Now we make standardized quantities

```
> z <- cbind(alpha.star, sigma.star)
> z <- sweep(z, 2, c(alpha.hat, sigma.hat))
> se.z <- cbind(se.alpha.star, se.sigma.star)
> z <- z / se.z
> apply(z, 2, mean)
```

```
[1] -0.19449646  0.19456491  0.19331165  0.17984055  0.01434750
[6]  0.01499696 -1.37949472 -1.25354773
```

```
> apply(z, 2, sd)
```

```
[1] 1.0189243 1.0190112 1.0146962 1.3651307 1.4327493 0.9472664
[7] 1.4910829 1.9438870
```

Not exactly standard normal (mean zero, standard deviation one). In fact, some seem quite far from standard normal.

If we apply more robust estimators of location and scale

```
> apply(z, 2, median)
```

```
[1] -0.02503285  0.02497086  0.01018879  0.10817474  0.08707737
[6]  0.09771285 -1.05151138 -0.86355658
```

```
> apply(z, 2, mad)
```

```
[1] 1.045318 1.045800 1.050378 1.239033 1.165617 1.007494 1.338282
[8] 1.306944
```

we see that some of the non-normality appears to be due to outliers, but the distributions are still nowhere near standard normal. This means if we really want accurate tests and confidence intervals, they cannot be based on the standard errors printed out by summary.reaster.

Suppose we want a 95% confidence interval for the coefficient for the fixed effect named "fit:SiteRiverside:RegionS". Instead of using

$$\text{point estimate} \pm 1.96 \times \text{standard error}$$

we should use critical values derived from the bootstrap distribution. Here's how to do that.

```
> conf.level <- 0.95
> n <- "fit:SiteRiverside:RegionS"
> colnames(z) <- c(names(alpha.hat), names(sigma.hat))
> myz <- z[ , n]
> crit <- quantile(myz, probs = c((1 - conf.level) / 2, (1 + conf.level) / 2))
> crit
```

```
     2.5%     97.5%
-1.769053  1.677903
```

```
> alpha.hat[n] - rev(crit) * se.alpha.hat[n]
```

```
    97.5%      2.5%
0.4789797 0.5207304
```

In this case, the bootstrap critical values are actually smaller than 1.96 so the bootstrap confidence interval is actually shorter than the confidence interval based on the standard error printed out by `summary.reaster`. But that won't be true in general.

Let us look at what seems to be the worst (most non-standard-normal) bootstrap distribution, that for ν_1.

```
> n <- "block"
> myz <- z[ , n]
> myz <- myz[! is.na(myz)]
```

makes the bootstrap z-scores and

```
> qqnorm(myz)
> qqline(myz)
```

makes a Q-Q plot (Figure 1).

We see from Fig. 1 that the distribution is not too far from normal in the middle (Tukey's law: *all* distributions look normal in the middle), but is centered in the wrong place (median = -1.052) has the wrong spread (1.4826 times median absolute deviation = 1.338), is skewed with a long left tail and short right tail.

For comparison we make the analogous plot for the corresponding variance component.

```
> n <- 1
> myz <- (nu.star[ , n] - nu.hat[n]) / se.nu.star[ , n]
> myz <- myz[! is.na(myz)]
> mean(myz)
```

```
[1] -1.848132
```

```
> sd(myz)
```

```
[1] 2.249301
```

```
> median(myz)
```

```
[1] -1.176664
```

```
> mad(myz)
```

```
[1] 1.535348
```

The following code

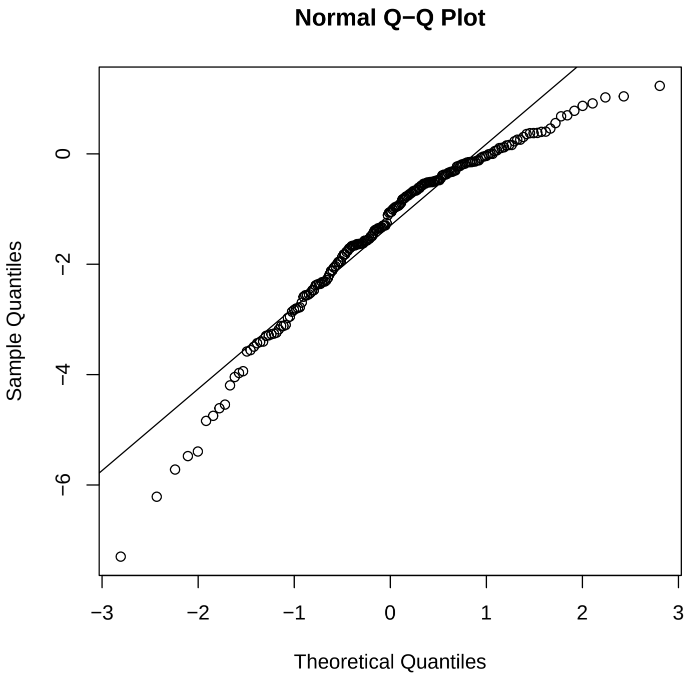

Figure 1: Q-Q plot of bootstrap z-scores for σ_1 (square root of variance component for blocks).

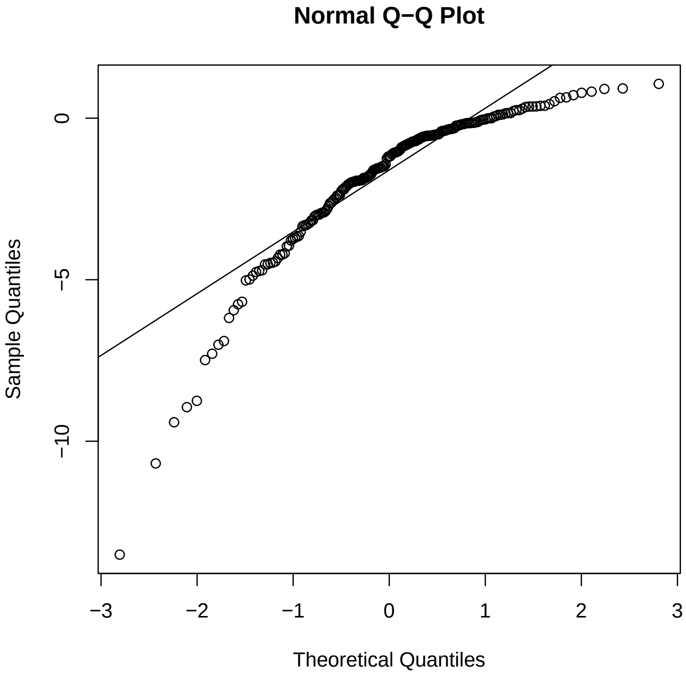

Figure 2: Q-Q plot of bootstrap z-scores for ν_1 (of variance component for blocks).

```
> qqnorm(myz)
> qqline(myz)
```

makes the Q-Q plot for this (Figure 2). We see that the bootstrap distribution of the variance component is even worse than the distribution of its square root. That is why we recommend using square roots of variance components.

That is the end of bootstrap analysis in this technical report. We see the asymptotic standard errors produced by `reaster` and `summary` are not perfect and not horrible. We will just use them from now on.

7 Oats Data

7.1 Data

We use data on the slender wild oat (*Avena barbata*) described by Latta (2009) and contained in the dataset `oats` in the R contributed package `aster`. For each individual, two response variables are observed, connected by the following graphical model

$$1 \xrightarrow{\text{Ber}} y_1 \xrightarrow{\text{0-Poi}} y_2$$

y_1 being an indicator of whether any spikelets (compound flowers) were produced, y_2 being the count of the number of spikelets produced, the unconditional distribution of y_1 being Bernoulli, and the conditional distribution of y_2 given y_1 being zero-truncated Poisson.

We load the data

```
> data(oats)
> names(oats)

 [1] "Plant.id" "Env"      "Gen"      "Fam"      "Site"     "Year"    
 [7] "varb"     "resp"     "id"       "root"     "fit"     

> levels(oats$varb)

[1] "Spike" "Surv" 
```

This is a "long format" data frame produced by the R command reshape from "wide format" data. The variable `varb` indicates which components of the response vector (variable `resp`) corresponded to which original variables in the "wide format" data (components of fitness). The variable `fit` is the indicator of the best surrogate of fitness in these data, which is the last node (y_2) of the graph and the `"Spike"` level of `varb`.

Then set up the graphical model

```
> pred <- c(0, 1)
> fam <- c(1, 3)
> sapply(fam.default(), as.character)[fam]

[1] "bernoulli"                         
[2] "truncated.poisson(truncation = 0)"
```

These data come from a designed experiment started with seeds collected in the 1980's in northern California of the xeric (found in drier regions) and mesic (found in less dry regions) ecotypes. The variable `Gen` is the ecotype (`"X"` or `"M"`). The variable `Fam` is the accession (nested within `Gen`). The variable `Site` is the site. The variable `Year` is the year (2003 to 2007). The experimental sites were at the Sierra Foothills Research and Extension Center (`Site == "SF"`), which is northeast of Sacramento on the east side of the Central Valley of California, and at the Hopland Research and Extension center (`Site = "Hop"`), which is in the California Coastal Ranges north of San Francisco. Hopland receives 30% more rainfall and has a less severe summer drought than Sierra foothills. The best surrogate of fitness in these data is the `Spike` component of the response vector. Thus we form the "interaction" with the indicator of this component and all scientifically interesting predictors (Section 4 above).

7.2 Random Effects Model

The random effects model of interest is complicated because interactions were statistically significant in the normal-normal (normal data, normal random effects) analysis, and we include them here too.

Effect	Type
Site	fixed
Year	random
Gen	fixed
Fam	random
Gen $*$ Site	fixed
Gen $*$ Year	random
Site $*$ Fam	random
Year $*$ Fam	random

Note that the interaction of a fixed effect and a random effect is a random effect. Each of these random effects is a vector whose components are assumed to have the same variance, so there is one variance component for each. And we need one random effects model matrix for each.

The following statements fit the random effects model.

```
> data(oats)
> pred <- c(0,1)
> fam <- c(1,3)
> rout <- reaster(resp ~ varb + fit : (Gen * Site),
+     list(year = ~ 0 + fit : Year, fam = ~ 0 + fit : Fam,
+         fam.site = ~ 0 + fit : Fam : Site, fam.year = ~ 0 + fit : Fam : Year,
+         gen.year = ~ 0 + fit : Gen : Year),
+         pred, fam, varb, id, root, data = oats)
> summary(rout)

Call:
reaster.formula(fixed = resp ~ varb + fit:(Gen * Site), random = list(year = ~0 +
    fit:Year, fam = ~0 + fit:Fam, fam.site = ~0 + fit:Fam:Site,
    fam.year = ~0 + fit:Fam:Year, gen.year = ~0 + fit:Gen:Year),
    pred = pred, fam = fam, varvar = varb, idvar = id, root = root,
    data = oats)


Fixed Effects:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)       2.86833    0.36873   7.779 7.31e-15
varbSurv        -15.15044    0.48600 -31.174  < 2e-16
fit:GenM          0.27250    0.13975   1.950  0.05118
fit:SiteSF       -0.32606    0.09609  -3.393  0.00069
fit:GenX:SiteSF   0.09138    0.14293   0.639  0.52259

Square Roots of Variance Components (P-values are one-tailed):
         Estimate Std. Error z value Pr(>|z|)/2
year      0.70794    0.25524   2.774    0.00277
fam       0.00000         NA      NA         NA
fam.site  0.17502    0.03013   5.810   3.13e-09
fam.year  0.18193    0.02538   7.168   3.80e-13
gen.year  0.10987    0.06078   1.807    0.03534
```

Again we have made an "interaction" with fit and all scientifically interesting effects, either fixed or random, as explained in Section 4.

We see that all variance components are significantly different from zero except for the fam random effect, which is estimated to be exactly zero. This happens in aster models with random effects, just like it happens in traditional normal-normal (normal data, normal random effects) random effects models. In this case, asymptotic normality does not hold. Part of the asymptotic distribution of the maximum likelihood estimator is an atom at zero, and any distribution having an atom is not normal. For this reason, the standard error for this variance component is reported as NA and similarly for the z-value and P-value. Anyway, there is no point in a P-value for testing whether this variance component is nonzero (when the data favor the null hypothesis, you don't need a P-value to accept the null hypothesis).

7.3 Does this Reanalysis Change Biological Conclusions?

In short, no. Latta (2009) did not do a random effects aster analysis because it had not yet been invented (instead he assumed normal response and did a conventional normal-normal random effects analysis). Nevertheless his conclusions hold up.

Local adaptation, which would have been shown by an interaction of ecotype with site was of interest, but was not found in the analysis by Latta (2009) using a conventional normal-normal random effects model. Instead it was found that the mesic ecotype had higher fitness (survived and reproduced better) in all environments. Our analysis here using aster models with random effects confirms this finding. This interaction is the coefficient named (by R) fit:GenX:SiteSF, which is the ecotype by site interaction. A statistically significantly nonzero value of this parameter would have indicated local adaptation if the pattern of mean values for ecotypes in the various sites had been as expected with local adaptation (each ecotype fitter in its home environment); these mean values are given in Latta (2009) (and do not show the pattern expected for local adaptation, so this test is moot, nevertheless we look at its P-value anyway). However this interaction is not statistically significant ($P = 0.523$). Like in our reanalysis, Latta (2009) did not find evidence of local adaptation. Instead, he found that the mesic ecotype had more fitness in all environments.

8 Partridge Pea Data

8.1 Data

We use data on the partridge pea (*Chamaecrista fasciculata*) described by Etterson (2004a,b) and Etterson and Shaw (2001) and contained in the dataset `chamae3` in the R contributed package `aster`. For each individual, two response variables are observed, connected by the following graphical model

$$1 \xrightarrow{\text{Ber}} y_1 \xrightarrow{\text{0-Poi}} y_2$$

y_1 being an indicator of whether any fruits were produced, y_2 being the count of the number of fruits produced, the unconditional distribution of y_1 being Bernoulli, and the conditional distribution of y_2 given y_1 being zero-truncated Poisson.

We load the data

```
> data(chamae3)
> names(chamae3)
```

```
 [1] "SIRE" "DAM"  "POP"  "SITE" "ROW"  "BLK"  "varb" "resp" "id"  
[10] "root" "fit" 
```

```
> levels(chamae3$varb)
```

```
[1] "fecund" "fruit" 
```

This is a "long format" data frame produced by the R command reshape from "wide format" data. The variable `varb` indicates which components of the response vector (variable `resp`) corresponded to which original variables in the "wide format" data (components of fitness). The variable `fit` is the indicator of the best surrogate of fitness in these data, which is the last node (y_2) of the graph and the `"fruit"` level of `varb`.

Then set up the graphical model

```
> pred <- c(0, 1)
> fam <- c(1, 3)
> sapply(fam.default(), as.character)[fam]
```

```
[1] "bernoulli"                          
[2] "truncated.poisson(truncation = 0)"
```

We show more information about variables in this dataset.

```
> levels(chamae3$POP)
```

```
[1] "1" "2" "3"

> levels(chamae3$SITE)

[1] "K" "M" "O"

> nlevels(chamae3$SIRE)

[1] 146

> nlevels(chamae3$DAM)

[1] 438

> head(chamae3$SIRE)

[1] 3030 3136 3004 1129 3175 1145
146 Levels: 1004 1021 1022 1026 1029 1030 1033 1034 1039 1040 ... 3199

> head(chamae3$DAM)

[1] 3149 3119 3190 1086 3008 1096
438 Levels: 1001 1002 1003 1005 1006 1007 1008 1009 1010 1011 ... 3201
```

C. fasciculata grows in the Great Plains of North America from southern Minnesota to Mexico. Three focal populations were sampled in the following locations

1. Kellog-Weaver Dunes, Wabasha County, Minnesota
2. Konza Prairie, Riley County, Kansas
3. Pontotoc Ridge, Pontotoc County, Oklahoma

(the numbers for the list items correspond to the levels of the variable `POP` in the dataset). These sites are progressively more arid from north to south and also differ in other characteristics. Seed pods were collected from 200 plants in each of these three natural populations, crosses were done, germinated, and raised in the greenhouse. The parent plants are indicated by the variables `SIRE` and `DAM` in the dataset. The seedlings from the greenhouse were planted using a randomized block design (Etterson, 2004b) in three field sites

`"O"` Robert S. Kerr Environmental Research Center, Ada, Oklahoma

`"K"` Konza Prairie Research Natural Area, Manhatten, Kansas

`"M"` University of Minnesota, St. Paul Minnesota

(the characters for the list items correspond to the levels of the variable `SITE` in the dataset). The Oklahoma field site was 30 km northwest of the Oklahoma natural population; the Kansas field site was 5 km from the Kansas natural population; the Minnesota field site was 110 km northwest of the Minnesota natural population.

These data are a subset of data previously analyzed by non-aster methods by Etterson (2004a,b) and Etterson and Shaw (2001) and by aster but not random effects aster methods by Shaw et al. (2008). Seed counts were also observed on up to three seed pods per plant and fecundity was estimated as average seed count × pod number with some exceptions. In some cases, especially Minnesota genotypes in the Oklahoma site, pods had dehisced and the plants senesced, in which case the number of pedicels that had remnant pod fragments still attached were counted and fecundity was imputed using the average seed count of the other full-sib replicate within the block or, if that was not available, the average seed count of the full-sib family across blocks. Because of the complexity of the seed count data, aster analysis that uses the seed counts is difficult (Shaw et al., 2008) and complicated and does not serve as a good example. Thus here we analyze only the pod number data (level `"fruit"` of variable `varb`), which does have straightforward aster analysis and serves as a better example, even though this makes our reanalysis not really comparable with the analysis in Etterson (2004b) which does use the seed counts. Shaw et al. (2008, p. E43) explain two alternative experimental designs that would have enabled straightforward aster analysis (including random effects aster models), but, of course, this experiment was done before aster models were invented, so there would have seemed no point to such designs at the time. Stanton-Geddes, Shaw and Tiffin (2012) used one of these designs, but their data do not address the questions we investigate here.

All individuals descended from all three natural populations were planted in all three field sites, so these data can address local adaptation and previous analyses Etterson (2004b, Discussion) did find local adaptation. But local adaptation is not the main point of interest for our analysis here. Instead we investigate whether sire effects, which we treat as random effects, as did the previous conventional quantitative genetics analysis (Etterson, 2004b) actually appear to be normally distributed. We focus on sire effects although dam effects are also treated as random effects because in this experimental design sire effects are expected to correspond closely to pure breeding values but dam effects will be confounded with maternal and dominance effects.

8.2 Analysis

First we do the aster analysis, analyzing each of the nine population-site pairs separately. Since these analyses take a long time we save the results and restore them here

```
> suppressWarnings(foo <- try(load("chamae-reaster.rda"), silent = TRUE))
> done <- (! inherits(foo, "try-error"))
> done
```

```
[1] FALSE
```

If `done` is `TRUE` then the analyses are already done and their results restored. Otherwise we have to do them. First we subset the data, making a list whose components are nine data frames (the data for the separate analyses).

```
> names(chamae3)
```

```
 [1] "SIRE" "DAM"  "POP"  "SITE" "ROW"  "BLK"  "varb" "resp" "id"
[10] "root" "fit"

> site <- as.character(chamae3$SITE)
> pop <- as.character(chamae3$POP)
> usite <- sort(unique(site))
> upop <- sort(unique(pop))
> usite

[1] "K" "M" "O"

> upop

[1] "1" "2" "3"

> rsite <- rep(usite, times = length(upop))
> rpop <- rep(upop, each = length(usite))
> cbind(rsite, rpop)

      rsite rpop
 [1,] "K"   "1"
 [2,] "M"   "1"
 [3,] "O"   "1"
 [4,] "K"   "2"
 [5,] "M"   "2"
 [6,] "O"   "2"
 [7,] "K"   "3"
 [8,] "M"   "3"
 [9,] "O"   "3"

> nsitepop <- paste(rsite, rpop, sep = "")
> nsitepop

[1] "K1" "M1" "O1" "K2" "M2" "O2" "K3" "M3" "O3"

> if (! done) {
+     subdata <- list()
+     for (i in seq(along = rsite))
+         subdata[[nsitepop[i]]] <- droplevels(subset(chamae3,
+             site == rsite[i] & pop == rpop[i]))
+ }
> length(subdata)

[1] 9

> sapply(subdata, nrow)

  K1   M1   O1   K2   M2   O2   K3   M3   O3
2108 2054 2034 2342 2256 2292 2020 1894 2062

> sapply(subdata, function(x) unique(x$SITE))
```

```
K1 M1 O1 K2 M2 O2 K3 M3 O3
 K  M  O  K  M  O  K  M  O
Levels: K M O

> sapply(subdata, function(x) unique(x$POP))

K1 M1 O1 K2 M2 O2 K3 M3 O3
 1  1  1  2  2  2  3  3  3
Levels: 1 2 3
```

We see we have successfully done the subsetting.

Then we do the analysis.

```
> if (! done) {
+     pea.analysis.time <- system.time(
+         subout <- lapply(subdata, function(x) reaster(resp ~ varb + fit:BLK,
+             list(sire = ~ 0 + fit:SIRE, dam = ~ 0 + fit:DAM),
+             pred, fam, varb, id, root, data = x))
+     )
+     sumout <- lapply(subout, summary)
+     save(subdata, subout, sumout, pea.analysis.time,
+         file = "chamae-reaster.rda")
+ }
```

Because the results of these analyses are voluminous, we put them in Appendix B. The nine invocations of the `reaster` took 0 hours, 13 minutes, and 54.9 seconds all together.

8.3 Significance Levels

```
> pp <- sapply(sumout, function(foo) foo$sigma["sire", "Pr(>|z|)/2"])
> round(pp, 4)

    K1     M1     O1     K2     M2     O2     K3     M3     O3
0.0009 0.5000 0.0406 0.0000 0.0565 0.0305 0.0068 0.3771 0.0001
```

We see that the sire variance components for the Minnesota and Oklahoma natural population are not close to statistically significant at the Minnesota field site. All the other sire variance components are at least borderline statistically significant.

8.4 Comparison of Breeding Values

Etterson (2004b, Figure 3) made scatterplots of breeding values (sire random effects) and we do the same. First we need to get the sire effects.

```
> subbreed <- lapply(subout, function(foo) foo$b)
> head(names(subbreed[[1]]))

[1] "fit:SIRE1004" "fit:SIRE1021" "fit:SIRE1022" "fit:SIRE1026"
[5] "fit:SIRE1029" "fit:SIRE1030"
```

```
> renames <- paste0("fit:SIRE", as.character(levels(chamae3$SIRE)))
> head(renames)
```

```
[1] "fit:SIRE1004" "fit:SIRE1021" "fit:SIRE1022" "fit:SIRE1026"
[5] "fit:SIRE1029" "fit:SIRE1030"
```

```
> subsire <- lapply(subout, function(foo) foo$b[renames])
> sapply(subsire, length)
```

```
 K1  M1  O1  K2  M2  O2  K3  M3  O3
146 146 146 146 146 146 146 146 146
```

```
> sapply(subsire, function(foo) sum(is.finite(foo)))
```

```
K1 M1 O1 K2 M2 O2 K3 M3 O3
48 48 48 50 50 50 48 48 48
```

The following code

```
> subsubsire <- subsire[paste0(c("M", "K", "O"), 1)]
> pairs(as.data.frame(subsubsire))
```

makes the pairwise scatter plots (Figure 3). We don't see anything interesting but that is perhaps because the breeding values for the Minnesota natural population are small.

We then repeat the same procedure for the other two natural populations. The code

```
> subsubsire <- subsire[paste0(c("M", "K", "O"), 2)]
> pairs(as.data.frame(subsubsire))
```

makes Figure 4, which is the Kansas natural population. The code

```
> subsubsire <- subsire[paste0(c("M", "K", "O"), 3)]
> pairs(as.data.frame(subsubsire))
```

makes Figure 5, which is the Oklahoma natural population.

8.5 Gaussianity of Breeding Values

Maximum likelihood makes the estimates of sire effects look normally distributed, even if the actual effects are not very normally distributed. This section looks at this issue.

The code

```
> qqnorm(subsire$K2)
> qqline(subsire$K2)
```

makes Figure 6, which is for the Kansas-Kansas (Kansas natural population and Kansas field site) sire effects. It shows them to be normally distributed.

These sire effects are penalized maximum likelihood estimators with quadratic penalty (b^* in the notation of Section 1.4 above) with penalty parameter (reciprocal variance component) estimated by approximate maximum likelihood (minimizing q defined in that section).

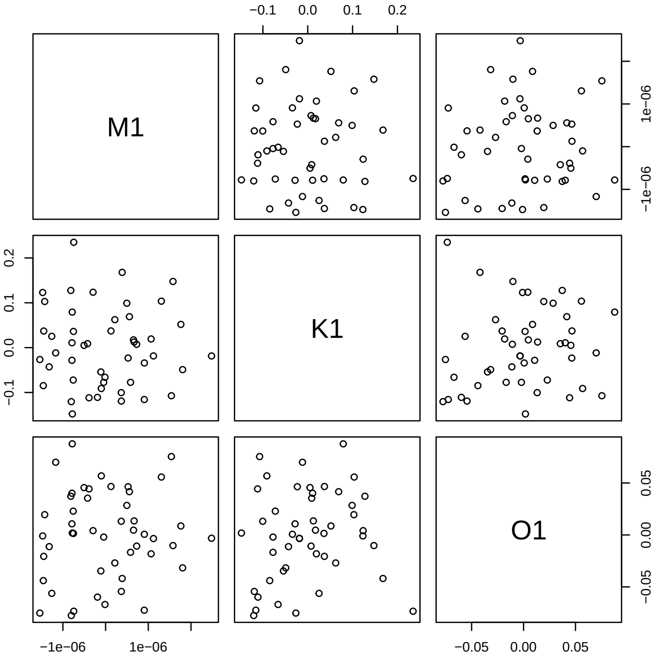

Figure 3: Pairwise scatter plots of sire effects for the Minnesota natural population at the various field sites (`M1` is Minnesota site, `K1` is Kansas site, and `O1` is Oklahoma site.

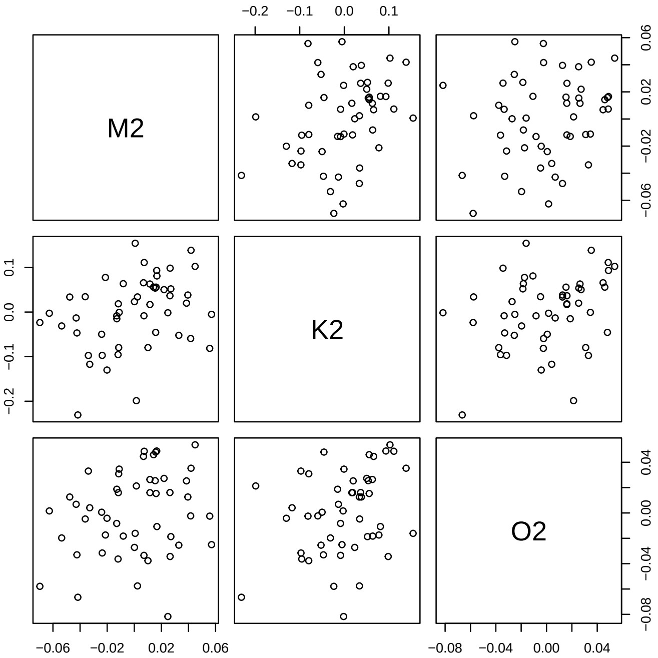

Figure 4: Pairwise scatter plots of sire effects for the Kansas natural population at the various field sites (`M2` is Minnesota site, `K2` is Kansas site, and `O2` is Oklahoma site.

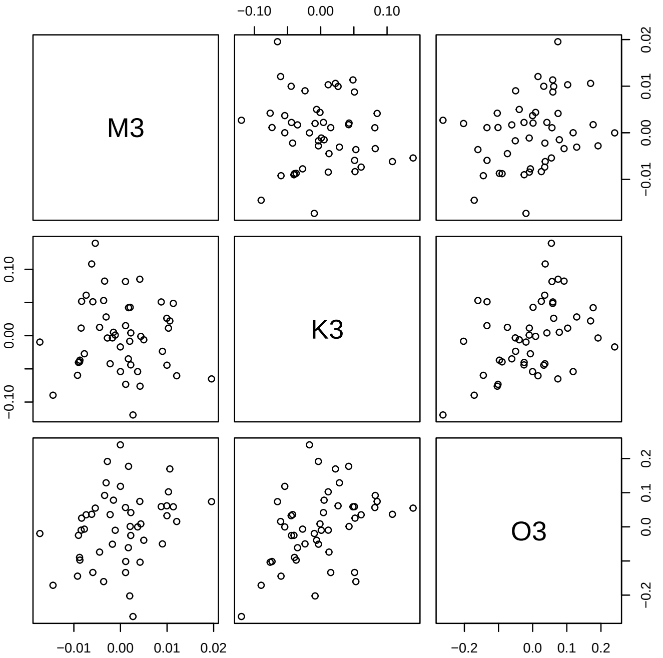

Figure 5: Pairwise scatter plots of sire effects for the Oklahoma natural population at the various field sites (`M3` is Minnesota site, `K3` is Kansas site, and `O3` is Oklahoma site.

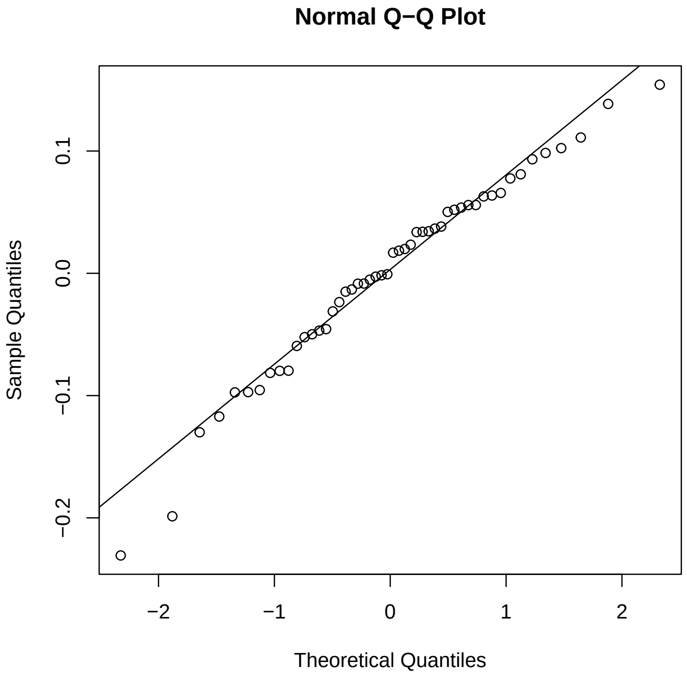

Figure 6: Q-Q plot of sire effects for the Kansas natural population at the Kansas field site.

If we broaden our context, such penalized maximum likelihood estimators are nothing new (Good and Gaskins, 1971; Hoerl and Kennard, 1970; Green, 1987; Hastie, Tibshirani and Friedman, 2009) and are widely used in statistics. What is different here is that elsewhere the penalty parameter (the multiplier of the penalty function) is chosen by cross-validation or AIC or some such device. The penalty parameter is not considered to be a parameter like other parameters, but as a constant to be adjusted to get good fit or good predictions.

If we take off our "normal random effects" hat and stop believing that there really random variables (the sire effects) that, although unobserved and unobservable even in principle, are exactly homoscedastically normally distributed, then we can still use penalized maximum likelihood. Let us see what happens when we use much smaller penalty (larger sire variance component) than the maximum likelihood penalty. We cannot use zero penalty because the sire effects are confounded with other effects (with the intercept and also with the dam effects). But we can use any nonzero penalty.

We try first with one-tenth the penalty (10 times the maximum likelihood sire variance component). There is not a nice function like `reaster` to do this. We have to use some of the "plumbing" functions that `reaster` calls.

```
> alpha.mle <- subout$K2$alpha
> sigma.mle <- subout$K2$sigma
> c.mle <- subout$K2$c
> sigma.penal10 <- c(sqrt(10), 1) * sigma.mle
> fixed <- subout$K2$fixed
> random <- subout$K2$random
> obj <- subout$K2$obj
> tout <- trust(objfun = penmlogl2, parinit = c.mle, rinit = 1, rmax = 10,
+     alpha = alpha.mle, sigma = sigma.penal10, fixed = fixed,
+     random = random, obj = obj)
> stopifnot(tout$converged)
> oldsire <- subsire$K2[is.finite(subsire$K2)]
> neweff <- tout$argument * tout$scale
```

The code

```
> newsire10 <- neweff[grep("SIRE", names(neweff))]
> plot(oldsire, newsire10, xlab = "MLE sire effects",
+     ylab = "sire effects with 1 / 10 the MLE penalty")
> abline(line(oldsire, newsire10))
```

makes Figure 7, which is still for the Kansas-Kansas sire effects but now plots the MLE effects against effects with smaller penalty. This doesn't show any non-normality. There is almost perfect linearity between the old and new sire effects. If one looks normal, then so will the other.

So try again with one-hundredth the penalty (100 times the maximum likelihood sire variance component).

```
> sigma.penal100 <- c(sqrt(100), 1) * sigma.mle
> tout <- trust(objfun = penmlogl2, parinit = tout$argument, rinit = 1, rmax = 10,
+     alpha = alpha.mle, sigma = sigma.penal100, fixed = fixed,
+     random = random, obj = obj)
```

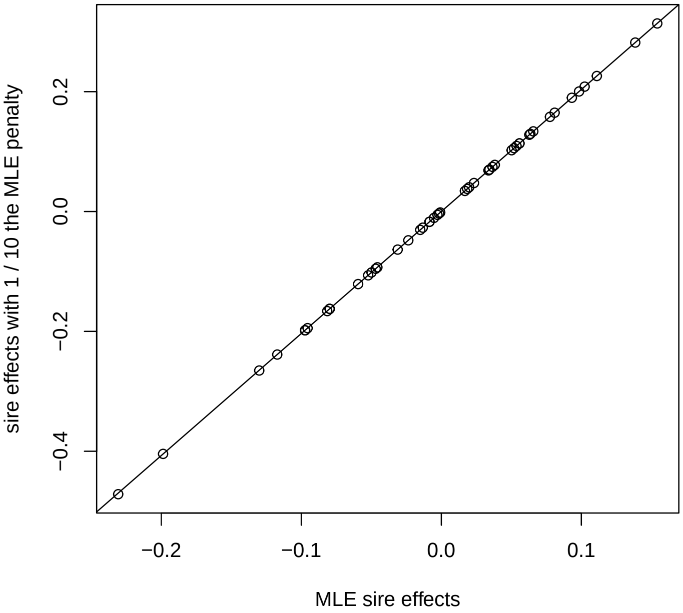

Figure 7: Plot of sire effects for the Kansas natural population at the Kansas field site estimated with the MLE penalty parameter (variance component) and 1 / 10 of that penalty.

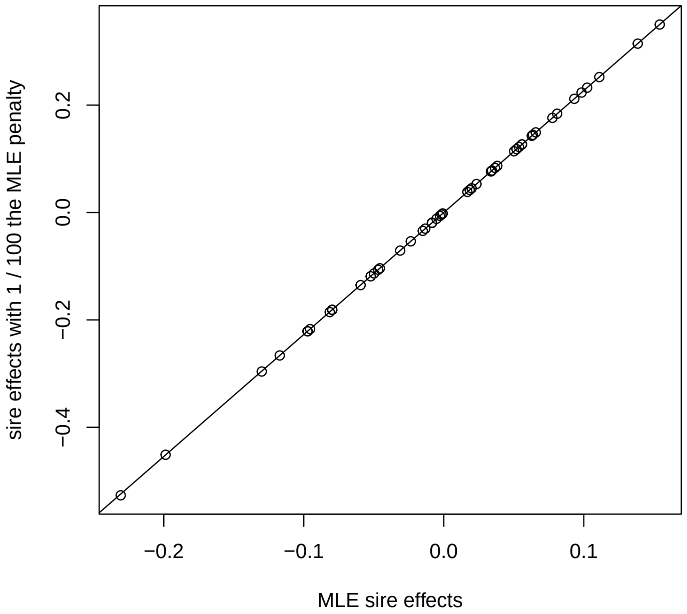

Figure 8: Plot of sire effects for the Kansas natural population at the Kansas field site estimated with the MLE penalty parameter (variance component) and 1 / 100 of that penalty.

```
> stopifnot(tout$converged)
> neweff <- tout$argument * tout$scale
```

The code

```
> newsire100 <- neweff[grep("SIRE", names(neweff))]
> plot(oldsire, newsire100, xlab = "MLE sire effects",
+     ylab = "sire effects with 1 / 100 the MLE penalty")
> abline(line(oldsire, newsire100))
```

makes Figure 8, which is still for the Kansas-Kansas sire effects but now plots the MLE effects against effects with even smaller penalty. This again doesn't show any non-normality.

So try again with one-millionth the penalty (10^6 times the maximum likelihood sire variance component).

```
> sigma.penal1e6 <- c(sqrt(1e6), 1) * sigma.mle
> tout <- trust(objfun = penmlogl2, parinit = tout$argument, rinit = 1, rmax = 10,
+     alpha = alpha.mle, sigma = sigma.penal1e6, fixed = fixed,
+     random = random, obj = obj)
> stopifnot(tout$converged)
> neweff <- tout$argument * tout$scale
```

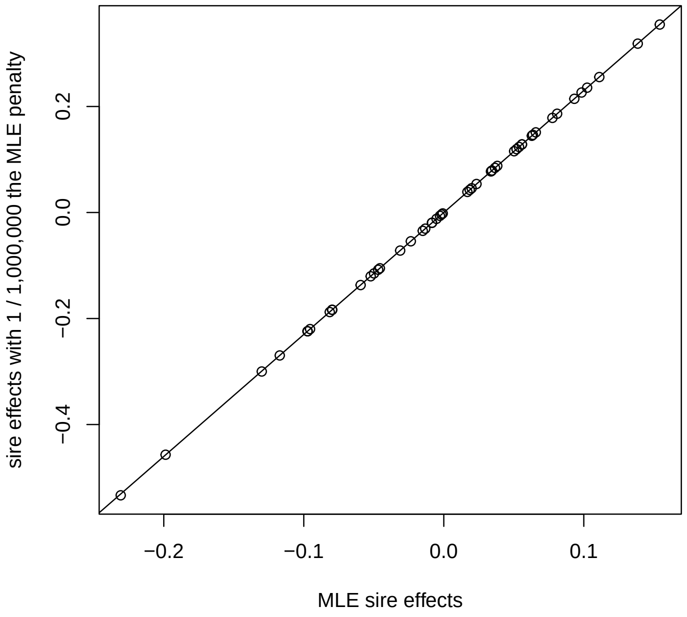

Figure 9: Plot of sire effects for the Kansas natural population at the Kansas field site estimated with the MLE penalty parameter (variance component) and 1 / 1,000,000 of that penalty.

The code

```
> newsire1e6 <- neweff[grep("SIRE", names(neweff))]
> plot(oldsire, newsire1e6, xlab = "MLE sire effects",
+     ylab = "sire effects with 1 / 1,000,000 the MLE penalty")
> abline(line(oldsire, newsire1e6))
```

makes Figure 9, which is still for the Kansas-Kansas sire effects but now plots the MLE effects against effects with a lot smaller penalty. This again doesn't show any non-normality.

In conclusion (of this section), there doesn't seem to be any evidence of non-normality in the particular population-site combination (Kansas-Kansas) we examined in detail. The other population-site combinations (not shown) are similar with only a few moderate outliers (points off the line, one outlier in Minnesota-Minnesota, two outliers in Kansas-Oklahoma, four outliers in Minnesota-Oklahoma, one outlier in Oklahoma-Oklahoma). None of these moderate outliers cause any apparent non-normality in Q-Q plots (not shown), nor do they have statistically significant non-normality as measured by the Shapiro-Wilk test (not shown, R function `shapiro.test`).

8.6 Mapping Breeding Values to Mean Values

The issue in the section title makes no sense without further clarification. The mapping between canonical and mean value parameter values in an aster model (or any exponential family model) is invertible. So there is no question that there is a one-to-one correspondence between canonical and mean value parameter values in the aster model. But this says nothing about random effects. It doesn't say anything about "effects" of any kind.

Since this one-to-one transformation is nonlinear, the phenomenon one has in linear models (where this transformation is the identity transformation, that is, canonical and mean value parameters are the same) that one can consider how much a change in one effect induces a change in mean values *without considering other effect values* does not occur. In generalized linear models and aster models, the amount of change depends on the values of the other effects. So one must carefully say what one is doing.

Here we are interested in mapping the sire effects to the mean value parameter scale. To do this we have specify the values of the other effects, both fixed and random. We set the other random effects (the dam effects) to zero (taking this to be a typical value), set the fixed effects to their maximum likelihood values, and predict for hypothetical individuals that are all in block 1 (so block effects do not influence the comparison of the sire effects).

Note that block 1 in one site has nothing to do with block 1 in another site, so this is arguably not the right thing to do, but it is not obvious that any other procedure is unarguably right either.

We want to make just two plots (merely to illustrate the method), the Kansas population in the Kansas site and in the Oklahoma site. These are the "K2" and "O2" elements of the various lists made above (which contain all nine site-population pairs).

```
> subsubout <- subout[c("K2", "O2")]
```

There is no function to do "prediction" for random effects. (It is not clear what such a function should do!) Thus we "predict" using the function `predict.aster`, which only understands fixed effects. We hand this function the object of class `"aster"` which is inside the object of class `"reaster"` produced by the R function `"reaster"`

```
> names(subsubout[[1]])

 [1] "obj"        "fixed"      "random"     "dropped"    "sigma"
 [6] "nu"         "c"          "b"          "alpha"      "zwz"
[11] "response"   "origin"     "iterations" "counts"     "deviance"
[16] "formula"    "call"
```

```
> class(subsubout[[1]]$obj)

[1] "aster"          "asterOrReaster"
```

```
> hoom <- predict(subsubout[[1]]$obj, subsubout[[1]]$alpha)
> hoom <- matrix(hoom, ncol = 2)
> hoom <- hoom[ , 2]
> unique(hoom)
```

```
[1] 158.5290 212.5000 260.4795 280.3276
```

These are the predicted values for the four blocks (not necessarily in order of block number). Let's restrict to block 1.

```
> hoom <- predict(subsubout[[1]]$obj, subsubout[[1]]$alpha)
> hoom <- matrix(hoom, ncol = 2)
> boom <- subsubout[[1]]$obj$modmat[ , , "fit:BLK1"] == 1
> unique(hoom[boom])
```

```
[1] 158.529
```

Our strategy will be to use the "prediction" (mean value parameter value) for any one of these individuals in block 1 and add to the block 1 effect the sire effect. First we find one of those individuals.

```
> idx <- subsubout[[1]]$obj$modmat[ , , "fit:BLK1"]
> idx <- matrix(idx, ncol = 2)
> idx <- idx[ , 2]
> idx <- seq(along = idx)[idx == 1]
> idx <- min(idx)
> idx
```

```
[1] 1
```

It turns out that block 1 comes first in the data (no surprise), so the first individual is in block 1.

Now get sire effects.

```
> subsubsire <- lapply(subsubout, function(x) x$b)
> subsubsire <- lapply(subsubsire, function(x) x[grep("SIRE", names(x))])
```

This means the following code gives mean value parameters for the first element of `subsubout`

```
> psire <- function(x) {
+     newcoef <- subsubout[[1]]$alpha
+     newcoef["fit:BLK1"] <- newcoef["fit:BLK1"] + x
+     hoom <- predict(subsubout[[1]]$obj, newcoef = newcoef)
+     hoom <- matrix(hoom, ncol = 2)
+     hoom[1, 2]
+ }
> musire1 <- unlist(Map(psire, subsubsire[[1]]))
```

And the following does the same for the second element

```
> psire <- function(x) {
+     newcoef <- subsubout[[2]]$alpha
+     newcoef["fit:BLK1"] <- newcoef["fit:BLK1"] + x
+     hoom <- predict(subsubout[[2]]$obj, newcoef = newcoef)
+     hoom <- matrix(hoom, ncol = 2)
+     hoom[1, 2]
+ }
```

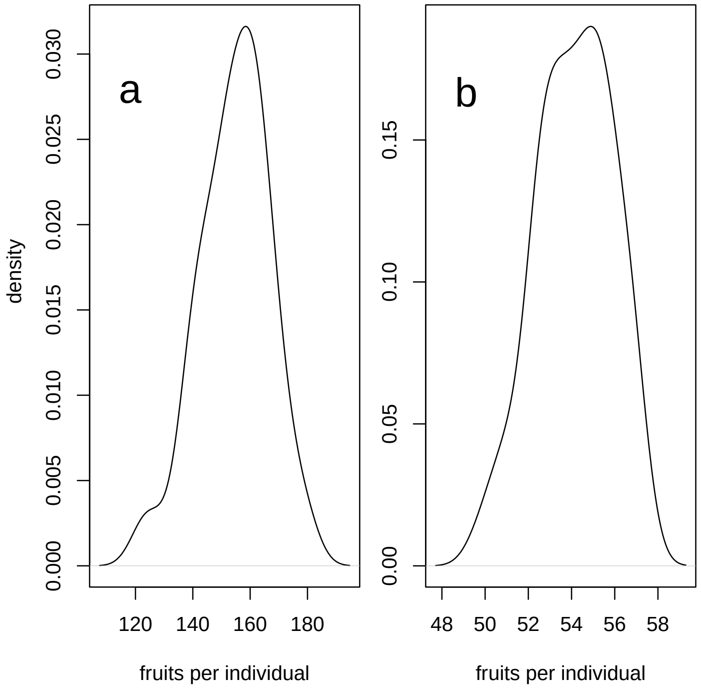

Figure 10: Plot of sire effects on the mean value parameter scale for an individual in block 1 having the various sire effects in the data. Panel a is the Kansas population in the Kansas field site. Panel b is the Kansas population in the Oklahoma field site.

```
> musire2 <- unlist(Map(psire, subsubsire[[2]]))
> musire <- list(musire1, musire2)
> names(musire) <- names(subsubout)
```

Now we make density plots

```
> subsubdens <- lapply(musire, density)
```

Then we do the plot. The code

```
> par(mar = c(5, 1.5, 1, 1) + 0.1, mfrow = c(1, 2), oma = c(0, 2.5, 0, 0))
> for (i in seq(along = subsubdens)) {
+     plot(subsubdens[[i]], ylab = "", xlab = "fruits per individual", main = "")
+     foo <- par("usr")
+     text(0.85 * foo[1] + 0.15 * foo[2], 0.15 * foo[3] + 0.85 * foo[4],
+         letters[i], cex = 2)
+ }
> mtext("density", side = 2, line = 1, outer = TRUE, at = 0.6)
```

makes Figure 10, which does both of these density plots in one figure.

The apparent non-Gaussianity of these plots is an artifact of small sample side. A Shapiro-Wilk test shows no statistically significant non-Gaussianity.

```
> lapply(subsubsire, shapiro.test)

$K2

        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.97181, p-value = 0.2738


$O2

        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.9729, p-value = 0.3025

> lapply(musire, shapiro.test)

$K2

        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.98499, p-value = 0.7713


$O2

        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.97541, p-value = 0.3782
```

A Cholesky

How do we calculate log determinants and derivatives thereof? R has a function `determinant` that calculates the log determinant. It uses *LU* decomposition.

An alternative method is to use Cholesky decomposition, but that only works when the given matrix is symmetric. This may be better because there is a sparse version (the `chol` function in the `Matrix` package) that may enable us to do much larger problems (perhaps after some other issues getting in the way of scaling are also fixed).

We need to calculate the log determinant that appears in (8) or (34), but the matrix is not symmetric. It can, however, be rewritten so as to be symmetric.

Assuming A is invertible

$$\begin{aligned}\det\bigl(Z^T \widehat{W} Z A^2 + I\bigr) &= \det\bigl(Z^T \widehat{W} Z A + A^{-1}\bigr) \det\bigl(A\bigr) \\ &= \det\bigl(A Z^T \widehat{W} Z A + I\bigr)\end{aligned}$$

If A is singular, we can see by continuity that the two sides must agree there too. That takes care of (34). The same trick works for (8); just replace A by $D^{1/2}$, which is the diagonal matrix whose diagonal components are the nonnegative square roots of the corresponding diagonal components of D.

Cholesky can also be used to efficiently calculate matrix inverses (done by the `chol2inv` function in the `Matrix` package). So we investigate whether we can use Cholesky to calculate derivatives.

A.1 First Derivatives

For the trace in the formula (37) for $\tilde{p}_{\sigma_j}(\alpha, c, \sigma)$ we have in case A is invertible

$$\begin{aligned}&\operatorname{tr}\Bigl([Z^T \widehat{W} Z A^2 + I]^{-1} Z^T \widehat{W} Z A E_j \Bigr) \\ &\qquad = \operatorname{tr}\Bigl([A^{-1}(A Z^T \widehat{W} Z A + I) A]^{-1} Z^T \widehat{W} Z A E_j \Bigr) \\ &\qquad = \operatorname{tr}\Bigl(A^{-1} [A Z^T \widehat{W} Z A + I]^{-1} A Z^T \widehat{W} Z A E_j \Bigr) \\ &\qquad = \operatorname{tr}\Bigl([A Z^T \widehat{W} Z A + I]^{-1} A Z^T \widehat{W} Z A E_j A^{-1} \Bigr) \\ &\qquad = \operatorname{tr}\Bigl([A Z^T \widehat{W} Z A + I]^{-1} A Z^T \widehat{W} Z E_j \Bigr)\end{aligned}$$

the next-to-last equality being $\operatorname{tr}(AB) = \operatorname{tr}(BA)$ and the last equality using the fact that A, E_j, and A^{-1} are all diagonal so they commute. Again we see that we get the same identity of the first and last expressions even when A is singular by continuity.

For the trace in the formula (11) for $p_{\nu_k}(\alpha, b, \nu)$ we have in case D is invertible

$$\begin{aligned}&\operatorname{tr}\Bigl(\bigl[Z^T \widehat{W} Z D + I\bigr]^{-1} Z^T \widehat{W} Z E_k \Bigr) \\ &\qquad = \operatorname{tr}\Bigl(D^{-1/2} \bigl[D^{1/2} Z^T \widehat{W} Z D^{1/2} + I\bigr]^{-1} D^{1/2} Z^T \widehat{W} Z E_k \Bigr) \\ &\qquad = \operatorname{tr}\Bigl(\bigl[D^{1/2} Z^T \widehat{W} Z D^{1/2} + I\bigr]^{-1} D^{1/2} Z^T \widehat{W} Z D^{-1/2} E_k \Bigr)\end{aligned}$$

This, of course, does not work when D is singular. We already knew we cannot differentiate $p(\alpha, b, \nu)$ on the boundary of the constraint set.

A.2 Second Derivatives

For the trace in the formula in Section 1.8 for $p_{\nu_j \nu_k}(\alpha, b, \nu)$ we have in case D is invertible

$$
\begin{aligned}
\operatorname{tr}\Bigl(\bigl[Z^T \widehat{W} Z D + I\bigr]^{-1} & Z^T \widehat{W} Z E_j \bigl[Z^T \widehat{W} Z D + I\bigr]^{-1} Z^T \widehat{W} Z E_k \Bigr) \\
&= \operatorname{tr}\Bigl(D^{-1/2} \bigl[D^{1/2} Z^T \widehat{W} Z D^{1/2} + I\bigr]^{-1} D^{1/2} Z^T \widehat{W} Z E_j \\
&\qquad D^{-1/2} \bigl[D^{1/2} Z^T \widehat{W} Z D^{1/2} + I\bigr]^{-1} D^{1/2} Z^T \widehat{W} Z E_k \Bigr) \\
&= \operatorname{tr}\Bigl(\bigl[D^{1/2} Z^T \widehat{W} Z D^{1/2} + I\bigr]^{-1} D^{1/2} Z^T \widehat{W} Z E_j D^{-1/2} \\
&\qquad\qquad \bigl[D^{1/2} Z^T \widehat{W} Z D^{1/2} + I\bigr]^{-1} D^{1/2} Z^T \widehat{W} Z E_k D^{-1/2} \Bigr)
\end{aligned}
$$

Again, this does not work when D is singular.

The same trace occurs in the expression for $q_{\nu_j \nu_k}(\alpha, \nu)$ given in Section 1.12 and can be calculated the same way.

B Partridge Pea Analysis Printout

```
> for (i in seq(along = sumout)) {
+     cat("\n\nSITE =", rsite[i], "and POP =", rpop[i], "\n")
+     print(sumout[[i]])
+ }

SITE = K and POP = 1

Call:
reaster.formula(fixed = resp ~ varb + fit:BLK, random = list(sire = ~0 +
    fit:SIRE, dam = ~0 + fit:DAM), pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = x)


Fixed Effects:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -18.48069    0.53273 -34.691  < 2e-16
varbfruit    22.35237    0.53469  41.804  < 2e-16
fit:BLK1     -0.29045    0.01335 -21.753  < 2e-16
fit:BLK2     -0.26231    0.01316 -19.929  < 2e-16
fit:BLK3      0.09342    0.01217   7.675 1.66e-14

Square Roots of Variance Components (P-values are one-tailed):
     Estimate Std. Error z value Pr(>|z|)/2
sire  0.14743    0.04733   3.115    0.00092
dam   0.34368    0.02552  13.467    < 2e-16


SITE = M and POP = 1
```

```
Call:
reaster.formula(fixed = resp ~ varb + fit:BLK, random = list(sire = ~0 +
    fit:SIRE, dam = ~0 + fit:DAM), pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = x)


Fixed Effects:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -71.365940   0.756004 -94.399  < 2e-16
varbfruit    76.380414   0.756820 100.923  < 2e-16
fit:BLK1     -0.373650   0.007369 -50.705  < 2e-16
fit:BLK2     -0.381537   0.007304 -52.235  < 2e-16
fit:BLK3     -0.053340   0.007102  -7.511 5.89e-14

Square Roots of Variance Components (P-values are one-tailed):
     Estimate Std. Error z value Pr(>|z|)/2
sire 0.000388   7.447842    0.00        0.5
dam  0.258927   0.018812   13.76     <2e-16


SITE = O and POP = 1

Call:
reaster.formula(fixed = resp ~ varb + fit:BLK, random = list(sire = ~0 +
    fit:SIRE, dam = ~0 + fit:DAM), pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = x)


Fixed Effects:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.61590    0.38842 -17.033  < 2e-16
varbfruit    9.37809    0.39191  23.929  < 2e-16
fit:BLK1     0.00149    0.02198   0.068 0.945964
fit:BLK2    -0.07552    0.02132  -3.542 0.000396
fit:BLK3    -0.05968    0.02149  -2.778 0.005478

Square Roots of Variance Components (P-values are one-tailed):
     Estimate Std. Error z value Pr(>|z|)/2
sire  0.09337    0.05353   1.744     0.0406
dam   0.30396    0.02391  12.712     <2e-16


SITE = K and POP = 2

Call:
reaster.formula(fixed = resp ~ varb + fit:BLK, random = list(sire = ~0 +
    fit:SIRE, dam = ~0 + fit:DAM), pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = x)
```

```
Fixed Effects:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -88.178261   1.429153  -61.70   <2e-16
varbfruit    93.779086   1.429651   65.60   <2e-16
fit:BLK1     -0.563040   0.005827  -96.63   <2e-16
fit:BLK2     -0.277439   0.005319  -52.16   <2e-16
fit:BLK3     -0.068424   0.005062  -13.52   <2e-16

Square Roots of Variance Components (P-values are one-tailed):
     Estimate Std. Error z value Pr(>|z|)/2
sire  0.12085    0.02999   4.029    2.8e-05
dam   0.23771    0.01700  13.986    < 2e-16


SITE = M and POP = 2

Call:
reaster.formula(fixed = resp ~ varb + fit:BLK, random = list(sire = ~0 +
    fit:SIRE, dam = ~0 + fit:DAM), pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = x)


Fixed Effects:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.330e+02  1.259e+00 -105.63   <2e-16
varbfruit    1.384e+02  1.260e+00  109.87   <2e-16
fit:BLK1     5.707e-02  5.351e-03   10.66   <2e-16
fit:BLK2    -9.622e-02  5.456e-03  -17.64   <2e-16
fit:BLK3     1.428e-01  5.176e-03   27.60   <2e-16

Square Roots of Variance Components (P-values are one-tailed):
     Estimate Std. Error z value Pr(>|z|)/2
sire  0.07164    0.04520   1.585     0.0565
dam   0.26114    0.01865  14.000     <2e-16


SITE = O and POP = 2

Call:
reaster.formula(fixed = resp ~ varb + fit:BLK, random = list(sire = ~0 +
    fit:SIRE, dam = ~0 + fit:DAM), pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = x)


Fixed Effects:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -32.74856    0.91077 -35.957  < 2e-16
varbfruit    36.81031    0.91162  40.379  < 2e-16
```

```
fit:BLK1     -0.07041    0.01084  -6.493 8.39e-11
fit:BLK2     -0.02198    0.01055  -2.084  0.03712
fit:BLK3     -0.03174    0.01061  -2.990  0.00279

Square Roots of Variance Components (P-values are one-tailed):
     Estimate Std. Error z value Pr(>|z|)/2
sire  0.06977    0.03725   1.873     0.0305
dam   0.22628    0.01673  13.526     <2e-16


SITE = K and POP = 3

Call:
reaster.formula(fixed = resp ~ varb + fit:BLK, random = list(sire = ~0 +
    fit:SIRE, dam = ~0 + fit:DAM), pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = x)


Fixed Effects:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -65.469538   0.999465  -65.50   <2e-16
varbfruit    71.077217   1.000407   71.05   <2e-16
fit:BLK1     -0.803528   0.006651 -120.81   <2e-16
fit:BLK2     -0.996898   0.007204 -138.37   <2e-16
fit:BLK3     -0.107690   0.005456  -19.74   <2e-16

Square Roots of Variance Components (P-values are one-tailed):
     Estimate Std. Error z value Pr(>|z|)/2
sire  0.10161    0.04118   2.467    0.00681
dam   0.27866    0.02050  13.596    < 2e-16


SITE = M and POP = 3

Call:
reaster.formula(fixed = resp ~ varb + fit:BLK, random = list(sire = ~0 +
    fit:SIRE, dam = ~0 + fit:DAM), pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = x)


Fixed Effects:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.326e+02 8.745e-01 -151.61   <2e-16
varbfruit    1.381e+02 8.755e-01  157.77   <2e-16
fit:BLK1    -3.068e-01 5.535e-03  -55.44   <2e-16
fit:BLK2    -2.965e-01 5.425e-03  -54.66   <2e-16
fit:BLK3    -5.793e-02 5.734e-03  -10.10   <2e-16

Square Roots of Variance Components (P-values are one-tailed):
     Estimate Std. Error z value Pr(>|z|)/2
sire  0.03736    0.11935   0.313      0.377
dam   0.31445    0.02350  13.379     <2e-16


SITE = 0 and POP = 3

Call:
reaster.formula(fixed = resp ~ varb + fit:BLK, random = list(sire = ~0 +
    fit:SIRE, dam = ~0 + fit:DAM), pred = pred, fam = fam, varvar = varb,
    idvar = id, root = root, data = x)


Fixed Effects:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -24.44572    0.73122 -33.431  < 2e-16
varbfruit    28.41596    0.73383  38.723  < 2e-16
fit:BLK1     -0.06119    0.01155  -5.296 1.18e-07
fit:BLK2      0.14785    0.01115  13.258  < 2e-16
fit:BLK3     -0.01166    0.01128  -1.034    0.301

Square Roots of Variance Components (P-values are one-tailed):
     Estimate Std. Error z value Pr(>|z|)/2
sire  0.16130    0.04273   3.775   7.99e-05
dam   0.32347    0.02431  13.307    < 2e-16
```

References

Anderson, T. W. (2003). *An Introduction to Multivariate Statistical Analysis*, 3rd ed. Hoboken: John Wiley & Sons.

Barndorff-Nielsen, O. (1978). *Information and Exponential Families.* Chichester: John Wiley & Sons.

Booth, J., and Hobert, J. P. (1999). Maximizing generalized linear mixed model likelihoods with an automated Monte Carlo EM algorithm. *Journal of the Royal Statistical Society*, Series B, **61**, 265–285.

Breslow, N. E., and Clayton, D. G. (1993). Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, **88**, 9–25.

Browder, A. (1996). *Mathematical Analysis: An Introduction.* New York: Springer-Verlag.

Etterson, J. R. (2004a). Evolutionary potential of *Chamaecrista fasciculata* in relation to climate change. I. Clinal patterns of selection along an environmental gradient in the great plains. *Evolution*, **58**, 1446–1458.

Etterson, J. R. (2004b). Evolutionary potential of *Chamaecrista fasciculata* in relation to climate change. II. Genetic architecture of three populations reciprocally planted along an environmental gradient in the great plains. *Evolution*, **58**, 1459–1471.

Etterson, J. R., and Shaw, R. G. (2001). Constraint to adaptive evolution in response to global warming. *Science*, **294**, 151–154.

Fletcher, R. (1987). *Practical Methods of Optimization*, 2nd ed. Chichester: John Wiley & Sons.

Geyer, C. J. (1994). On the convergence of Monte Carlo maximum likelihood calculations. *Journal of the Royal Statistical Society, Series B*, **56**, 261–274.

Geyer, C. J. (2009). Likelihood inference in exponential families and directions of recession. *Electronic Journal of Statistics*, **3**, 259–289.

Geyer, C. J. (2010). A Philosophical Look at Aster Models. Technical Report No. 676. School of Statistics, University of Minnesota. `http://purl.umn.edu/57163`.

Geyer, C. J. (2012). aster: Aster Models. R package version 0.8-12. `http://www.stat.umn.edu/geyer/aster/`.

Geyer, C. J. (in press). Asymptotics of maximum likelihood without the LLN or CLT or sample size going to infinity. To appear in *Festschrift for Morris L. Eaton*, G. Jones and X. Shen eds. Institute of Mathematical Statistics: Hayward, CA.

Geyer, C. J., and Thompson, E. A. (1992). Constrained Monte Carlo maximum likelihood for dependent data, (with discussion). *Journal of the Royal Statistical Society, Series B*, **54**, 657–699.

Geyer, C. J., Wagenius, S., and Shaw, R. G. (2007). Aster models for life history analysis. *Biometrika*, **94**, 415–426.

Good, I. J., and Gaskins, R. A. (1971). Nonparametric roughness penalties for probability densities. *Biometrika*, **58**, 255–277.

Green, P. J. (1987). Penalized likelihood for general semi-parametric regression models. *International Statistical Review*, **55**, 245–259.

Harville, D. A. (1997). Matrix Algebra From a Statistician's Perspective. New York: Springer.

Hastie, T., Tibshirani, R., and Friedman J. (2009). *Elements of Statistical Learning: Data Mining, Inference and Prediction*, 2nd ed. New York: Springer.

Hoerl, A. E., and Kennard, R. W. (1970). Ridge regression: applications to nonorthogonal problems. *Technometrics*, **12**, 69–82.

Hunter, D. R., Handcock, M. S., Butts, C. T., Goodreau, S. M., Morris, M. (2008). `ergm`: A package to fit, simulate and diagnose exponential-family models for networks. *Journal of Statistical Software*, **24**.

Hummel, R. M., Hunter, D. R., and Handcock, M. S. (to appear). Improving simulation-based algorithms for fitting EGRMs. To appear in *Journal of Statistical Software*.

Latta, R. G. (2009). Testing for local adaptation in *Avena barbata*, a classic example of ecotypic divergence. *Molecular Ecology*, **18**, 3781–3791.

Nocedal, J., and Wright, S. J. (1999). *Numerical Optimization*. New York: Springer.

Okabayashi, S., and Geyer, C. J. (2011) Gradient-based search for maximum likelihood in exponential families. *Electronic Journal of Statistics*, **6**, 123–147.

Penttinen, A. (1984). Modelling interations in spatial point patterns: parameter estimation by the maximum likelihood method. Jyväskylä Studies in Computer Science, Economics and Statistics, **7**.

R Development Core Team (2012). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. `http://www.R-project.org`.

Ridley, C. E., and Ellstrand, N. C. (2010). Rapid evolution of morphology and adaptive life history in the invasive California wild radish (*Raphanus sativus*) and the implications for management. *Evolutionary Applications*, **3**, 64–76.

Rockafellar, R. T., and Wets, R. J.-B. (2004). *Variational Analysis*, corr. 2nd printing. Berlin: Springer-Verlag.

Searle, S. R., Casella, G., and McCulloch, C. E. (1992). *Variance Components*. New York: John Wiley.

Shaw, F. H., Promislow, D. E. L., Tatar, M., Hughes, K. A. and Geyer, C. J. (1999). Towards reconciling inferences concerning genetic variation in senescence. *Genetics*, **152**, 553–566.

Shaw, F. H., Geyer, C. J., and Shaw, R. G. (2002). A Comprehensive Model of Mutations Affecting Fitness and Inferences for *Arabidopsis thaliana Evolution*, **56**, 453–463.

Shaw, R. G., Geyer, C. J., Wagenius, S., Hangelbroek, H. H., and Etterson, J. R. (2007). Supporting data analysis for "Unifying life history analysis for inference of fitness and population growth". University of Minnesota School of Statistics Technical Report No. 658 `http://www.stat.umn.edu/geyer/aster/`

Shaw, R. G., Geyer, C. J., Wagenius, S., Hangelbroek, H. H., and Etterson, J. R. (2008). Unifying life history analysis for inference of fitness and population growth. *American Naturalist*, **172**, E35–E47.

Shaw, R. G., and Geyer, C. J. (2010). Inferring fitness landscapes. *Evolution*, **64**, 2510–2520.

Stanton-Geddes, J., Shaw, R. G., and Tiffin, P. (2012). Interactions between soil habitat and geographic range affect plant fitness. *PLoS One*, **7**, e36015.

Sung, Y. J., and Geyer, C. J. (2007). Monte Carlo likelihood inference for missing data models. *Annals of Statistics*, **35**, 990–1011.

Thompson, E. A., and Guo, S. W. (1991). Evaluation of likelihood ratios for complex genetic models. *IMA J. Math. Appl. Med. Biol.*, **8**, 149–169.