Design Document for Dimension Reduction in R Package glmm

Christina Knudson

July 16, 2020

Abstract

This design document describes the process of regularizing the random effects. For example, Sally and Tammy can share a random effect (and therefore be treated as the same salamander) if they have very similar random effects (such as 1.22 and 1.24). The goal of reducing the number of random effects is to reduce the computation time for MCLA.

1 Background

Let $y = (y_1, \ldots, y_n)'$ be a vector of observed data. Let u be a q-vector of unobserved, normally-distributed random effects centered at 0 with variance matrix D. Let β be a vector of p fixed effect parameters and let ν be a vector of T variance components for the random effects so that D depends on ν. Let $\theta = (\beta, \nu)'$ be a vector containing all unknown parameters. Then the data y are distributed conditionally on the random effects according to $f_\theta(y|u)$ and the random effects are distributed according to $f_\theta(u)$. Although $f_\theta(u)$ does not actually depend on β and $f_\theta(y|u)$ does not depend on ν, we write both densities with θ to keep notation simple in future equations.

Since u is unobservable, the log likelihood must be expressed by integrating out the random effects:

$$l(\theta) = \log \int f_\theta(y|u) f_\theta(u)\, du \tag{1}$$

For most datasets, this integral is intractible. In these cases, performing even basic inference on the likelihood is not possible. Rather than evaluating the integral, **?** suggest using a Monte Carlo approximation to the likelihood. Monte Carlo likelihood approximation (MCLA) uses an importance sampling distribution $\tilde{f}(u)$ to generate random effects $u_k, k = 1, \ldots, m$ where m is the Monte Carlo sample size. .

Then the Monte Carlo log likelihood approximation is

$$l_m(\theta) = \log \frac{1}{m} \sum_{k=1}^{m} f_\theta(y|u_k) \frac{f_\theta(u_k)}{\tilde{f}(u_k)} \tag{2}$$

$$= \log \frac{1}{m} \sum_{k=1}^{m} \frac{f_\theta(y, u_k)}{\tilde{f}(u_k)}. \tag{3}$$

If u_k were of length $q_2 < q$, this calculation would be less expensive. We therefore are trying to shorten the random effect vector u.

2 Overview

The package, in its current state has two main steps:

1. Use PQL to estimate the parameters and predict the random effects. (These predictions are then used to create the importance sampling distribution.)
2. Use MCLA to approximate the likelihood and conduct MCML.

I am figuring out one of these two options:

Plan A: Change PQL so that similar random effects are fused into a shared, single random effect.

Plan B: Keep the PQL optimization as it is, and add another step in between the two steps. At this time, the similar random effects are fused into a shared, single random effect.

I'm not sure, but it seems like Plan A is better than Plan B. It makes sense that we'd want to try fusing the random effects while making sure this simplification not ruin the fit of the model.

In the currently used version of PQL, there is an optimization within an optimization. The inner optimization estimates the fixed effects and predicts the random effects. The outer optimization works on the variance components. **I believe that adding a penalty term to the inner loop will fuse the random effects.** We want the random effects to be similar to one other, so the distance (or the square of the distance, if we prefer a ridge-like penalty) between the random effects must be controlled. This is almost exactly pairwise fusion lasso (Petry et. al. 2011), except we are fusing *random* effects instead of *fixed* effects.

3 PQL

This section on PQL requires a change in notation so that we can avoid constrained optimization. Recall that $D = \text{Var}(u)$ and is assumed to be diagonal. Let $A = D^{1/2}$ so that A has diagonal components that are positive or negative. Using A rather than D enables unconstrained optimization. If σ is a vector of the distinct standard deviations with components σ_t, we can write A as a function of σ by

$$A = \sum_t E_t \sigma_t.$$

Recall that E_t has a diagonal of indicators to show which random effects have the same variance components, and $\sum_{t=1}^{T} E_t$ is the identity matrix. PQL estimates the components contained on the diagonal of A. Taking the absolute value of those components provides the standard deviations (where the standard deviations are the square root of the variance components).

We also define s where $u = As$. The purpose of using s rather than u is to avoid $D^{-1/2}$ in the objective functon that we optimize.

There are two variations of PQL, both of which are described in the vignette `re.pdf` in the R package `aster` (Geyer, 2014). Both variations have an inner optimization and an outer optimization. I use the variation of PQL that is not the original version, but is pretty close and is better behaved. In this version, the inner optimization finds $\tilde{\beta}$ and $\tilde{s}$ given X, Z and A. Then, given $\tilde{\beta}$ and $\tilde{s}$, the outer optimization finds A. **I believe that adding a penalty term to the inner loop will fuse the random effects. We want the random effects to be similar to one other, so the distance (or the square of the distance) between the random effects must be controlled. In other words, if s has components $s_1, \ldots, s_q$, then we want to have penalty term $\sum_{k \neq j} |s_k - s_j|$ if we want a lasso-like penalty or $\sum_{k \neq j} (s_k - s_j)^2$ if we want a ridge-like penalty.**

Of course, PQL already has a penalty term in order to shrink the random effects toward 0. We will maintain this penalty term (I think) and add a second penalty (to group together similar random effects).

The **inner** optimization is done with the `trust` function in R. We chose to use `trust` because it requires two derivatives, which make the optimization more precise. We would like more accuracy in the inner maximization because any imprecision carries into the outer optimization.

The inner optimization maximizes the penalized log likelihood. Defining

$$\eta = X\beta + ZAs \tag{4}$$

we calculate the log likelihood as

$$l(\eta) = y'\eta - c(\eta) \tag{5}$$

and the penalized likelihood (pre-fusion) as:

$$l(\eta) - \frac{1}{2} s' s. \tag{6}$$

Then our doubly-penalized likelihood is

$$l(\eta) - \frac{1}{2}s's. - \lambda \sum_{k \neq j} (s_k - s_j)^2 \tag{7}$$

where λ is some number we can tweak to force more/less fusion.

I HAVE NOT REDONE THESE CALCULATIONS BELOW and it will kind of suck because of the new fusion penalty.

Since this function is maximized using `trust`, we need to provide derivatives with respect to s and β. We express these via the multivariate chain rule, taking advantage of η_i.

Create vector μ with components $\mu_i = c'(\eta_i)$. Since

$$l(\eta) = \sum_i Y_i \eta_i - c(\eta_i) \tag{8}$$

then

$$\frac{\partial l(\eta)}{\partial \eta_i} = Y_i - c'(\eta_i) = Y_i - \mu_i. \tag{9}$$

Now we can write the following expression:

$$\begin{aligned}
\frac{\partial}{\partial \beta_k}\left[l(\eta) - \frac{1}{2}s's\right] &= \frac{\partial l(\eta)}{\partial \beta_k} && (10)\\
&= \frac{\partial l(\eta)}{\partial \eta_i}\frac{\partial \eta_i}{\partial \beta_k} && (11)\\
&= \sum_i (y_i - \mu_i)\frac{\partial \eta_i}{\partial \beta_k} && (12)\\
&= \sum_i (y_i - \mu_i) X_{ik} && (13)
\end{aligned}$$

We find the function's derivative with respect to s as follows:

$$\begin{aligned}
\frac{\partial}{\partial s}\left[l(\eta) - \frac{1}{2}s's\right] &= \frac{\partial l(\eta)}{\partial s} - \frac{1}{2}\frac{\partial s's}{\partial s} && (14)\\
&= \frac{\partial l(\eta)}{\partial \eta}\frac{\partial \eta}{\partial s} - s && (15)\\
&= (y - \mu)'\left[\frac{\partial}{\partial s} ZAs\right] - s && (16)\\
&= (y - \mu)' ZA - s && (17)
\end{aligned}$$

This gives us the following derivatives of the penalized log likelihood:

$$\frac{\partial}{\partial \beta}\left[l(\eta) - (1/2)s's\right] = X'(y-\mu) \tag{18}$$
$$\frac{\partial}{\partial s}\left[l(\eta) - (1/2)s's\right] = AZ'(y-\mu) - s \tag{19}$$

Lastly, we need the Hessian of the penalized likelihood. This matrix can be broken down into four pieces:

1. $\dfrac{\partial^2}{\partial s^2}$
2. $\dfrac{\partial^2}{\partial \beta^2}$
3. $\dfrac{\partial^2}{\partial s\, \partial \beta}$
4. $\left(\dfrac{\partial^2}{\partial s\, \partial \beta}\right)' = \dfrac{\partial^2}{\partial \beta\, \partial s}$

We start at the top with $\dfrac{\partial^2}{\partial s^2}$, which is a $q \times q$ matrix.

$$\begin{aligned}
\frac{\partial^2}{\partial s^2}\left[l(\eta) - (1/2)s's\right] &= \frac{\partial}{\partial s}\left[(y - c'(\eta))'ZA - s\right] && (20)\\
&= \left[-\frac{\partial}{\partial s}c'(\eta)\right]' ZA - I_q && (21)\\
&= \left[-\frac{\partial c'(\eta)}{\partial \eta}\frac{\partial \eta}{\partial s}\right]' ZA - I_q && (22)\\
&= \left[-c''(\eta)ZA\right]' ZA - I_q && (23)\\
&= -AZ'\left[c''(\eta)\right] ZA - I_q && (24)
\end{aligned}$$

Note that $c''(\eta)$ is a $q \times q$ diagonal matrix with diagonal elements $c''(\eta_i)$. I_q is the identity matrix of dimension q. This makes $\dfrac{\partial^2}{\partial s^2}$ a $q \times q$ matrix.

Next is $\dfrac{\partial^2}{\partial \beta^2}$, which is a $p \times p$ matrix.

$$\begin{aligned}
\frac{\partial^2}{\partial \beta^2}\left[l(\eta) - (1/2)s's\right] &= \frac{\partial}{\partial \beta}\left[X'(y - c'(\eta))\right] && (25)\\
&= \frac{\partial}{\partial \beta}\left[X'y - X'(c'(\eta))\right] && (26)\\
&= \frac{\partial}{\partial \beta}\left[-X'(c'(\eta))\right] && (27)\\
&= -X\left[{}'\frac{\partial}{\partial \beta}c'(\eta)\right] && (28)\\
&= -X'\left[\frac{\partial c'(\eta)}{\partial \eta}\frac{\partial \eta}{\partial \beta}\right] && (29)\\
&= -X'\left[c''(\eta)\right]X && (30)
\end{aligned}$$

Next is the $p \times q$ mixed partial $\dfrac{\partial^2}{\partial \beta\, \partial s}$.

$$\begin{aligned}
\frac{\partial^2}{\partial \beta\, \partial s}\left[l(\eta) - (1/2)s's\right] &= \frac{\partial}{\partial \beta}\left\{\left[y - c'(\eta)\right]' ZA\right\} && (31)\\
&= \frac{\partial}{\partial \beta}\left\{y'ZA - \left[c'(\eta)\right]' ZA\right\} && (32)\\
&= -\frac{\partial}{\partial \beta}\left\{\left[c'(\eta)\right]' ZA\right\} && (33)\\
&= -\left[\frac{\partial c'(\eta)}{\partial \beta}\right]' ZA && (34)\\
&= -\left[\frac{\partial c'(\eta)}{\partial \eta}\frac{\partial \eta}{\partial \beta}\right]' ZA && (35)\\
&= -\left[c''(\eta)X\right]' ZA && (36)\\
&= -X'\left[c''(\eta)\right] ZA && (37)
\end{aligned}$$

And last we have the $q \times p$ mixed partial $\dfrac{\partial^2}{\partial \beta\, \partial s} = -AZ'[c''(\eta)]X$. These four pieces specify the hessian matrix for the penalized likelihood. Now `trust` can perform the inner optimization to find $\tilde{\beta}$ and $\tilde{s}$.

The **outer** optimization is done using `optim` with the default method of "`Nelder-Mead.`" This requires just the function value and no derivatives. This optimization method was chosen because the optimization function already contains second derivatives of the cumulant function; requiring derivatives of the optimization function would in turn require higher-order derivatives of the cumulant.

The default of `optim` is to minimize, but we'd like to do maximization. Reversing the sign of the optimization function will turn the maximization into minimization.

If $\tilde{\beta}$ and $\tilde{s}$ are available from previous calls to the inner optimization function, then they are used here. Otherwise, the values are taken to be 0 and 1. The outer optimization's function is evaluated by first defining

$$\begin{aligned}
\tilde{\eta} &= X\tilde{\beta} + ZA\tilde{s} && (38)\\
l(\tilde{\eta}) &= y'\tilde{\eta} - c(\tilde{\eta}) && (39)\\
A &= \sum_k E_k \sigma_k. && (40)
\end{aligned}$$

Let W be a diagonal matrix with elements $c''(\eta_i)$ on the diagonal. Then the quantity we would like to maximize is the penalized quasi-likelihood:

$$l(\tilde{\eta}) - \frac{1}{2}\tilde{s}'\tilde{s} - \frac{1}{2}\log|AZ'WZA + I| \tag{41}$$

Again, `optim` minimizes, so we have to minimize the negative value of the penalized quasi-likelihood in order to maximize the value of it.

We do need to be careful about the determinant. Let $AZ'WZA + I = LL'$ where L is the lower triangular matrix resulting from a Cholesky decomposition. Then

$$\begin{aligned}
\frac{1}{2}\log|AZ'WZA + I| &= \frac{1}{2}\log|LL'| && (42)\\
&= \frac{1}{2}\log(|L|)^2 && (43)\\
&= \log|L| && (44)
\end{aligned}$$

Since L is triangular, the determinant is just the product of the diagonal elements. Let l_i be the diagonal elements of L. Then

$$\begin{aligned}
\frac{1}{2}\log|AZ'WZA + I| &= \log\prod l_i && (45)\\
&= \sum \log l_i && (46)
\end{aligned}$$

If I become worried about all variance components being zero, I could implement an eigendecomposition using the R function `eigen`. This would be more numerically stable, but is slower. Let O be the matrix containing the eigenvectors and let Λ be a diagonal matrix with the eigenvalues λ_i on the diagonal. Then

$$AZ'WZA = O\Lambda O' \tag{47}$$

Then we can rewrite the argument of the determinant as follows:

$$AZ'WZA + I = O\Lambda O' + I = O\Lambda O + OO' = O(I + \Lambda)O' \tag{48}$$

This leads to the careful calculation of our determinant as follows:

$$|AZ'WZA + I| = 1 * \prod_{i=1}^{n}(1 + \lambda_i) * 1 \tag{49}$$

$$\Rightarrow \log|AZ'WZA + I| = \sum \log(1 + \lambda_i) \tag{50}$$

The last quantity can be accurately calculated using the `log1p` function in R.

The outer optimization uses $\tilde{\beta}$ and $\tilde{s}$ provided by the inner optimization, but it does not return them. To keep track of the most recent $\tilde{s}$, so store them in an environment that I call `cache`. The purpose of $\tilde{\beta}$ and $\tilde{s}$ is two-fold. First, if they are available from a previous iteration of the inner optimization, then they are used in the outer optimization of PQL. Second, after PQL is finished, $\tilde{s}$ is used to help center the generated random effects.

The point of utilizing PQL is to construct a decent importance sampling distribution. Thus, the estimates do not have to be perfect. It is possible that one of the σ_k will be 0 according to PQL. If this happens, then I just use $\sigma_k = .01$ for the importance sampling distribution.